# Table of Contents

AWS Prescriptive Guidance patterns ........................................................................................................... 1
Analytics .......................................................................................................................................................... 2
  Summary .................................................................................................................................................. 2
  Prerequisites and limitations ..................................................................................................................... 2
  Architecture .......................................................................................................................................... 3
  Tools ..................................................................................................................................................... 3
  Epics ..................................................................................................................................................... 4
  Related resources ................................................................................................................................. 6
Attachments ................................................................................................................................................. 6

Analyze Amazon Redshift data in Microsoft SQL Server Analysis Services ................................................. 6
  Summary ................................................................................................................................................ 6
  Prerequisites and limitations ................................................................................................................... 7
  Architecture ......................................................................................................................................... 7
  Tools ................................................................................................................................................... 8
  Epics ................................................................................................................................................... 8
  Related resources ................................................................................................................................. 9

Automate data loading from Amazon S3 to Amazon Redshift .................................................................... 9
  Summary .............................................................................................................................................. 9
  Prerequisites and limitations .................................................................................................................. 9
  Architecture ....................................................................................................................................... 10
  Tools ................................................................................................................................................ 10
  Epics ............................................................................................................................................... 11
  Related resources ............................................................................................................................. 12

Build an ETL pipeline from Amazon S3 to Amazon Redshift using AWS Glue ............................................. 12
  Summary ........................................................................................................................................... 12
  Prerequisites and limitations .................................................................................................................. 13
  Architecture .................................................................................................................................... 13
  Tools ............................................................................................................................................... 14
  Epics ............................................................................................................................................... 15
  Related resources ............................................................................................................................. 19
  Additional information .......................................................................................................................... 19

Convert NORMALIZE to Amazon Redshift SQL ....................................................................................... 19
  Summary ........................................................................................................................................... 20
  Prerequisites and limitations .................................................................................................................. 20
  Architecture .................................................................................................................................... 20
  Tools ............................................................................................................................................... 20
  Epics ............................................................................................................................................... 24
  Related resources ............................................................................................................................. 24

Convert RESET WHEN to Amazon Redshift SQL ...................................................................................... 25
  Summary ........................................................................................................................................... 25
  Prerequisites and limitations .................................................................................................................. 25
  Architecture .................................................................................................................................... 25
  Tools ............................................................................................................................................... 25
  Epics ............................................................................................................................................... 28
  Related resources ............................................................................................................................. 29

Ensure Amazon EMR logging to Amazon S3 ............................................................................................. 29
  Summary ........................................................................................................................................... 29
  Prerequisites and limitations .................................................................................................................. 30
  Architecture .................................................................................................................................... 30
  Tools ............................................................................................................................................... 31
  Epics ............................................................................................................................................... 31
  Related resources ............................................................................................................................. 33
Attachments ................................................................................................................................................. 6
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install SSM Agent on Amazon EKS worker nodes</td>
<td>219</td>
</tr>
<tr>
<td>Generate a static outbound IP address using Lambda and Amazon VPC</td>
<td>211</td>
</tr>
<tr>
<td>Deploy a gRPC-based application on Amazon EKS</td>
<td>203</td>
</tr>
<tr>
<td>Deploy a clustered application to Amazon ECS by using AWS Copilot</td>
<td>196</td>
</tr>
<tr>
<td>Deploy Kubernetes packages using Amazon EKS and Helm</td>
<td>184</td>
</tr>
<tr>
<td>Deploy Java microservices on Amazon ECS using Amazon ECR and AWS Fargate</td>
<td>174</td>
</tr>
<tr>
<td>Deploy Java microservices on Amazon ECS using Amazon ECR and load balancing</td>
<td>179</td>
</tr>
<tr>
<td>Deploy Java microservices on Amazon ECS using Amazon ECR and AWS Fargate</td>
<td>174</td>
</tr>
<tr>
<td>Related resources</td>
<td>173</td>
</tr>
<tr>
<td>Summary</td>
<td>174</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>174</td>
</tr>
<tr>
<td>Architecture</td>
<td>174</td>
</tr>
<tr>
<td>Tools</td>
<td>176</td>
</tr>
<tr>
<td>Epics</td>
<td>176</td>
</tr>
<tr>
<td>Related resources</td>
<td>179</td>
</tr>
<tr>
<td>Summary</td>
<td>180</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>180</td>
</tr>
<tr>
<td>Architecture</td>
<td>180</td>
</tr>
<tr>
<td>Tools</td>
<td>182</td>
</tr>
<tr>
<td>Epics</td>
<td>183</td>
</tr>
<tr>
<td>Related resources</td>
<td>184</td>
</tr>
<tr>
<td>Summary</td>
<td>184</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>184</td>
</tr>
<tr>
<td>Architecture</td>
<td>185</td>
</tr>
<tr>
<td>Tools</td>
<td>186</td>
</tr>
<tr>
<td>Epics</td>
<td>186</td>
</tr>
<tr>
<td>Related resources</td>
<td>191</td>
</tr>
<tr>
<td>Summary</td>
<td>191</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>192</td>
</tr>
<tr>
<td>Architecture</td>
<td>192</td>
</tr>
<tr>
<td>Tools</td>
<td>193</td>
</tr>
<tr>
<td>Epics</td>
<td>194</td>
</tr>
<tr>
<td>Related resources</td>
<td>195</td>
</tr>
<tr>
<td>Summary</td>
<td>195</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>197</td>
</tr>
<tr>
<td>Architecture</td>
<td>197</td>
</tr>
<tr>
<td>Tools</td>
<td>198</td>
</tr>
<tr>
<td>Epics</td>
<td>199</td>
</tr>
<tr>
<td>Related resources</td>
<td>203</td>
</tr>
<tr>
<td>Summary</td>
<td>203</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>204</td>
</tr>
<tr>
<td>Architecture</td>
<td>205</td>
</tr>
<tr>
<td>Tools</td>
<td>206</td>
</tr>
<tr>
<td>Epics</td>
<td>206</td>
</tr>
<tr>
<td>Related resources</td>
<td>209</td>
</tr>
<tr>
<td>Summary</td>
<td>209</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>209</td>
</tr>
<tr>
<td>Architecture</td>
<td>209</td>
</tr>
<tr>
<td>Tools</td>
<td>209</td>
</tr>
<tr>
<td>Epics</td>
<td>209</td>
</tr>
<tr>
<td>Additional information</td>
<td>6</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Generate a static outbound IP address using Lambda and Amazon VPC</td>
<td>211</td>
</tr>
<tr>
<td>Summary</td>
<td>211</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>211</td>
</tr>
<tr>
<td>Architecture</td>
<td>212</td>
</tr>
<tr>
<td>Tools</td>
<td>213</td>
</tr>
<tr>
<td>Epics</td>
<td>213</td>
</tr>
<tr>
<td>Related resources</td>
<td>219</td>
</tr>
<tr>
<td>Install SSM Agent on Amazon EKS worker nodes</td>
<td>219</td>
</tr>
<tr>
<td>Summary</td>
<td>220</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>220</td>
</tr>
<tr>
<td>Architecture</td>
<td>220</td>
</tr>
<tr>
<td>Tools</td>
<td>221</td>
</tr>
<tr>
<td>Epics</td>
<td>222</td>
</tr>
<tr>
<td>Related resources</td>
<td>223</td>
</tr>
<tr>
<td>Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands</td>
<td>223</td>
</tr>
<tr>
<td>Summary</td>
<td>223</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>223</td>
</tr>
<tr>
<td>Architecture</td>
<td>224</td>
</tr>
<tr>
<td>Tools</td>
<td>224</td>
</tr>
<tr>
<td>Epics</td>
<td>225</td>
</tr>
<tr>
<td>Related resources</td>
<td>228</td>
</tr>
<tr>
<td>Additional information</td>
<td>228</td>
</tr>
<tr>
<td>Optimize generated Docker images</td>
<td>229</td>
</tr>
<tr>
<td>Summary</td>
<td>229</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>229</td>
</tr>
<tr>
<td>Architecture</td>
<td>229</td>
</tr>
<tr>
<td>Tools</td>
<td>230</td>
</tr>
<tr>
<td>Epics</td>
<td>231</td>
</tr>
<tr>
<td>Related resources</td>
<td>235</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Replicate filtered Amazon ECR container images across accounts or Regions</td>
<td>235</td>
</tr>
<tr>
<td>Summary</td>
<td>235</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>236</td>
</tr>
<tr>
<td>Architecture</td>
<td>236</td>
</tr>
<tr>
<td>Tools</td>
<td>237</td>
</tr>
<tr>
<td>Epics</td>
<td>238</td>
</tr>
<tr>
<td>Related resources</td>
<td>245</td>
</tr>
<tr>
<td>Additional information</td>
<td>246</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Rotate container secrets without restarting</td>
<td>246</td>
</tr>
<tr>
<td>Summary</td>
<td>246</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>247</td>
</tr>
<tr>
<td>Architecture</td>
<td>247</td>
</tr>
<tr>
<td>Tools</td>
<td>249</td>
</tr>
<tr>
<td>Epics</td>
<td>250</td>
</tr>
<tr>
<td>Related resources</td>
<td>250</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Run Amazon ECS tasks on Amazon WorkSpaces</td>
<td>251</td>
</tr>
<tr>
<td>Summary</td>
<td>251</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>251</td>
</tr>
<tr>
<td>Architecture</td>
<td>251</td>
</tr>
<tr>
<td>Tools</td>
<td>252</td>
</tr>
<tr>
<td>Epics</td>
<td>253</td>
</tr>
<tr>
<td>Related resources</td>
<td>257</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Run an ASP.NET web API Docker container on AWS</td>
<td>258</td>
</tr>
<tr>
<td>Summary</td>
<td>258</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>258</td>
</tr>
<tr>
<td>Architecture</td>
<td>259</td>
</tr>
<tr>
<td>Tools</td>
<td>259</td>
</tr>
<tr>
<td>Epics</td>
<td>260</td>
</tr>
<tr>
<td>Related resources</td>
<td>265</td>
</tr>
<tr>
<td>Run stateful workloads with persistent data storage</td>
<td>265</td>
</tr>
<tr>
<td>Summary</td>
<td>265</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>266</td>
</tr>
<tr>
<td>Epics</td>
<td>348</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Related resources</td>
<td>353</td>
</tr>
<tr>
<td>Monitor Amazon Aurora for encryption</td>
<td>353</td>
</tr>
<tr>
<td>Summary</td>
<td>353</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>353</td>
</tr>
<tr>
<td>Architecture</td>
<td>354</td>
</tr>
<tr>
<td>Tools</td>
<td>354</td>
</tr>
<tr>
<td>Epics</td>
<td>355</td>
</tr>
<tr>
<td>Related resources</td>
<td>356</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Replicate data between Amazon RDS for MySQL and Amazon EC2</td>
<td>357</td>
</tr>
<tr>
<td>Summary</td>
<td>357</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>357</td>
</tr>
<tr>
<td>Architecture</td>
<td>357</td>
</tr>
<tr>
<td>Tools</td>
<td>358</td>
</tr>
<tr>
<td>Epics</td>
<td>358</td>
</tr>
<tr>
<td>Related resources</td>
<td>359</td>
</tr>
<tr>
<td>Schedule jobs for Amazon RDS and Aurora PostgreSQL</td>
<td>359</td>
</tr>
<tr>
<td>Summary</td>
<td>359</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>360</td>
</tr>
<tr>
<td>Architecture</td>
<td>360</td>
</tr>
<tr>
<td>Tools</td>
<td>361</td>
</tr>
<tr>
<td>Epics</td>
<td>362</td>
</tr>
<tr>
<td>Related resources</td>
<td>363</td>
</tr>
<tr>
<td>Database migration patterns by workload</td>
<td>364</td>
</tr>
<tr>
<td>IBM</td>
<td>364</td>
</tr>
<tr>
<td>Microsoft</td>
<td>364</td>
</tr>
<tr>
<td>N/A</td>
<td>365</td>
</tr>
<tr>
<td>Open-source</td>
<td>365</td>
</tr>
<tr>
<td>Oracle</td>
<td>366</td>
</tr>
<tr>
<td>SAP</td>
<td>367</td>
</tr>
<tr>
<td>More patterns</td>
<td>367</td>
</tr>
<tr>
<td>DevOps</td>
<td>370</td>
</tr>
<tr>
<td>Summary</td>
<td>371</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>371</td>
</tr>
<tr>
<td>Architecture</td>
<td>372</td>
</tr>
<tr>
<td>Tools</td>
<td>373</td>
</tr>
<tr>
<td>Epics</td>
<td>373</td>
</tr>
<tr>
<td>Related resources</td>
<td>374</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Automate backups from AWS CodeCommit to Amazon S3</td>
<td>374</td>
</tr>
<tr>
<td>Summary</td>
<td>375</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>375</td>
</tr>
<tr>
<td>Architecture</td>
<td>375</td>
</tr>
<tr>
<td>Tools</td>
<td>376</td>
</tr>
<tr>
<td>Epics</td>
<td>376</td>
</tr>
<tr>
<td>Related resources</td>
<td>378</td>
</tr>
<tr>
<td>Additional information</td>
<td>378</td>
</tr>
<tr>
<td>Automate static website deployment to Amazon S3</td>
<td>379</td>
</tr>
<tr>
<td>Summary</td>
<td>380</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>380</td>
</tr>
<tr>
<td>Architecture</td>
<td>380</td>
</tr>
<tr>
<td>Tools</td>
<td>381</td>
</tr>
<tr>
<td>Epics</td>
<td>381</td>
</tr>
<tr>
<td>Related resources</td>
<td>384</td>
</tr>
<tr>
<td>Additional information</td>
<td>384</td>
</tr>
</tbody>
</table>
Automatically attach a managed policy for Systems Manager to EC2 instance profiles ....................... 384
Summary ........................................................................................................................................ 385
Prerequisites and limitations ........................................................................................................ 385
Architecture .................................................................................................................................. 387
Tools ............................................................................................................................................ 388
Epics ............................................................................................................................................ 388
Related resources ........................................................................................................................ 395
Attachments ................................................................................................................................ 6
Automatically build CI/CD pipelines and Amazon ECS clusters for microservices ............................... 395
Summary ........................................................................................................................................ 395
Prerequisites and limitations ........................................................................................................ 395
Architecture .................................................................................................................................. 396
Tools ............................................................................................................................................ 397
Epics ............................................................................................................................................ 398
Related resources ........................................................................................................................ 402
Additional information .................................................................................................................. 402
Attachments ................................................................................................................................ 6
Build a loosely coupled architecture with microservices ................................................................. 403
Summary ........................................................................................................................................ 403
Prerequisites and limitations ........................................................................................................ 403
Architecture .................................................................................................................................. 404
Tools ............................................................................................................................................ 405
Epics ............................................................................................................................................ 405
Related resources ........................................................................................................................ 409
Additional information .................................................................................................................. 409
Build and test iOS apps with AWS services ..................................................................................... 409
Summary ........................................................................................................................................ 410
Prerequisites and limitations ........................................................................................................ 410
Architecture .................................................................................................................................. 410
Tools ............................................................................................................................................ 411
Epics ............................................................................................................................................ 412
Related resources ........................................................................................................................ 413
Check AWS CDK applications or CloudFormation templates for best practices by using rule packs ... 414
Summary ........................................................................................................................................ 414
Prerequisites and limitations ........................................................................................................ 414
Tools ............................................................................................................................................ 415
Epics ............................................................................................................................................ 415
Related resources ........................................................................................................................ 416
Configure cross-account Amazon DynamoDB access ........................................................................ 416
Summary ........................................................................................................................................ 416
Prerequisites and limitations ........................................................................................................ 417
Architecture .................................................................................................................................. 417
Tools ............................................................................................................................................ 418
Epics ............................................................................................................................................ 419
Related resources ........................................................................................................................ 426
Additional information .................................................................................................................. 426
Attachments ................................................................................................................................ 6
Configure mutual TLS for applications on Amazon EKS ................................................................. 428
Summary ........................................................................................................................................ 428
Prerequisites and limitations ........................................................................................................ 428
Architecture .................................................................................................................................. 429
Tools ............................................................................................................................................ 429
Epics ............................................................................................................................................ 429
Related resources ........................................................................................................................ 434
Attachments ................................................................................................................................ 6
Create a CI/CD pipeline for microservices with AWS Fargate and API Gateway ............................... 434
Summary ........................................................................................................................................ 434
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epics</td>
<td>561</td>
</tr>
<tr>
<td>Related resources</td>
<td>562</td>
</tr>
<tr>
<td><strong>Configure vRealize Automation to provision VMs on VMware Cloud on AWS</strong></td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>563</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>563</td>
</tr>
<tr>
<td>Architecture</td>
<td>563</td>
</tr>
<tr>
<td>Tools</td>
<td>564</td>
</tr>
<tr>
<td>Epics</td>
<td>565</td>
</tr>
<tr>
<td>Related resources</td>
<td>565</td>
</tr>
<tr>
<td>Deploy an SDDC by using VMware Cloud on AWS</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>569</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>570</td>
</tr>
<tr>
<td>Architecture</td>
<td>570</td>
</tr>
<tr>
<td>Tools</td>
<td>571</td>
</tr>
<tr>
<td>Epics</td>
<td>571</td>
</tr>
<tr>
<td>Related resources</td>
<td>574</td>
</tr>
<tr>
<td>Migrate VMs to VMware Cloud on AWS by using HCX OSAM</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>575</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>575</td>
</tr>
<tr>
<td>Architecture</td>
<td>576</td>
</tr>
<tr>
<td>Tools</td>
<td>576</td>
</tr>
<tr>
<td>Epics</td>
<td>577</td>
</tr>
<tr>
<td>Related resources</td>
<td>578</td>
</tr>
<tr>
<td>More patterns</td>
<td>578</td>
</tr>
<tr>
<td><strong>Infrastructure</strong></td>
<td>579</td>
</tr>
<tr>
<td>Check EC2 instances for mandatory tags at launch</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>579</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>579</td>
</tr>
<tr>
<td>Architecture</td>
<td>580</td>
</tr>
<tr>
<td>Tools</td>
<td>580</td>
</tr>
<tr>
<td>Epics</td>
<td>581</td>
</tr>
<tr>
<td>Related resources</td>
<td>582</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Deploy Sophos web proxy UTM on AWS</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>583</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>583</td>
</tr>
<tr>
<td>Architecture</td>
<td>584</td>
</tr>
<tr>
<td>Tools</td>
<td>586</td>
</tr>
<tr>
<td>Epics</td>
<td>586</td>
</tr>
<tr>
<td>Related resources</td>
<td>587</td>
</tr>
<tr>
<td>Deploy a Cassandra cluster on Amazon EC2 with private static IPs</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>587</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>588</td>
</tr>
<tr>
<td>Architecture</td>
<td>588</td>
</tr>
<tr>
<td>Epics</td>
<td>590</td>
</tr>
<tr>
<td>Related resources</td>
<td>593</td>
</tr>
<tr>
<td>Ensure that Amazon EC2 instances launch only in approved AWS Regions</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>593</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>593</td>
</tr>
<tr>
<td>Architecture</td>
<td>594</td>
</tr>
<tr>
<td>Tools</td>
<td>594</td>
</tr>
<tr>
<td>Epics</td>
<td>595</td>
</tr>
<tr>
<td>Related resources</td>
<td>596</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>Modernize your mainframe environment with Micro Focus</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>597</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>597</td>
</tr>
<tr>
<td>Related resources</td>
<td>598</td>
</tr>
</tbody>
</table>
Prerequisites and limitations ........................................................................................................... 753
Architecture ........................................................................................................................................ 754
Tools .................................................................................................................................................. 754
Epics .................................................................................................................................................. 755
Related resources ............................................................................................................................ 760
Create AWS CloudFormation templates for AWS DMS ................................................................. 760
Summary ........................................................................................................................................... 760
Prerequisites and limitations ............................................................................................................. 761
Architecture ....................................................................................................................................... 761
Tools .................................................................................................................................................. 762
Epics .................................................................................................................................................. 762
Related resources ............................................................................................................................ 763
Get started with automated portfolio discovery .................................................................................. 763
Summary ........................................................................................................................................... 764
Tools .................................................................................................................................................. 764
Epics .................................................................................................................................................. 764
Related resources ............................................................................................................................ 767
Additional information ....................................................................................................................... 768
Attachments ....................................................................................................................................... 768
Install CloudEndure Agent through AWS Systems Manager on Windows and Linux ...................... 768
Summary ........................................................................................................................................... 768
Prerequisites and limitations ............................................................................................................. 769
Architecture ....................................................................................................................................... 769
Tools .................................................................................................................................................. 770
Epics .................................................................................................................................................. 770
Related resources ............................................................................................................................ 773
Attachments ....................................................................................................................................... 773
Migrate from on-premises MySQL to Amazon RDS for MySQL ...................................................... 774
Summary ........................................................................................................................................... 774
Prerequisites and limitations ............................................................................................................. 774
Architecture ....................................................................................................................................... 774
Tools .................................................................................................................................................. 777
Epics .................................................................................................................................................. 777
Related resources ............................................................................................................................ 779
Move mainframe files to Amazon S3 using Transfer Family ............................................................. 779
Summary ........................................................................................................................................... 780
Prerequisites and limitations ............................................................................................................. 780
Architecture ....................................................................................................................................... 780
Tools .................................................................................................................................................. 781
Epics .................................................................................................................................................. 782
Related resources ............................................................................................................................ 787
Restart the CloudEndure Agent automatically without disabling SELinux ....................................... 787
Summary ........................................................................................................................................... 787
Prerequisites and limitations ............................................................................................................. 788
Architecture ....................................................................................................................................... 788
Tools .................................................................................................................................................. 788
Epics .................................................................................................................................................. 789
Related resources ............................................................................................................................ 791
Re-architect ........................................................................................................................................ 791
Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL .................... 792
Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL using AWS SCT and AWS DMS ...... 797
Migrate Oracle external tables to Amazon Aurora .............................................................................. 806
Migrate Oracle native functions to PostgreSQL .............................................................................. 811
Migrate a SQL Server database from Amazon EC2 to Amazon DocumentDB .................................. 816
Migrate a ThoughtSpot Falcon database to Amazon Redshift ............................................................ 822
Migrate from Amazon RDS for Oracle to MySQL ............................................................................ 830
Migrate from IBM Db2 to Aurora PostgreSQL .................................................................................. 836
# AWS Prescriptive Guidance Patterns

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate from Oracle 8i/9i to Amazon RDS for PostgreSQL using Quest SharePlex</td>
<td>840</td>
</tr>
<tr>
<td>Migrate from Oracle 8i/9i to Amazon RDS for PostgreSQL using materialized views</td>
<td>847</td>
</tr>
<tr>
<td>Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL</td>
<td>854</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon DocumentDB</td>
<td>860</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon DynamoDB</td>
<td>865</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon RDS for MariaDB</td>
<td>870</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon RDS for MySQL</td>
<td>876</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon RDS for PostgreSQL</td>
<td>880</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate</td>
<td>888</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon Redshift</td>
<td>891</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon Redshift using data extraction agents</td>
<td>899</td>
</tr>
<tr>
<td>Migrate from Oracle to Aurora MySQL</td>
<td>902</td>
</tr>
<tr>
<td>Migrate from Oracle to Aurora PostgreSQL</td>
<td>908</td>
</tr>
<tr>
<td>Migrate from Oracle with standby to Aurora PostgreSQL</td>
<td>915</td>
</tr>
<tr>
<td>Migrate from SAP ASE to Amazon RDS for SQL Server</td>
<td>922</td>
</tr>
<tr>
<td>Migrate from SAP ASE to Aurora MySQL</td>
<td>926</td>
</tr>
<tr>
<td>Migrate from SAP ASE to Aurora PostgreSQL</td>
<td>931</td>
</tr>
<tr>
<td>Migrate from SAP ASE to PostgreSQL EC2</td>
<td>936</td>
</tr>
<tr>
<td>Migrate from SQL Server to Amazon Redshift</td>
<td>940</td>
</tr>
<tr>
<td>Migrate from SQL Server to Amazon Redshift using data extraction agents</td>
<td>944</td>
</tr>
<tr>
<td>Migrate from Teradata to Amazon Redshift using data extraction agents</td>
<td>947</td>
</tr>
<tr>
<td>Migrate from Vertica to Amazon Redshift using data extraction agents</td>
<td>950</td>
</tr>
<tr>
<td>Migrate from on-premises MySQL to Aurora MySQL</td>
<td>953</td>
</tr>
<tr>
<td>Set up Oracle UTL_FILE functionality on Amazon Aurora</td>
<td>958</td>
</tr>
<tr>
<td>Ingest EC2 Windows instances into an AMS account</td>
<td>969</td>
</tr>
<tr>
<td>Automate pre-workload ingestion activities</td>
<td>974</td>
</tr>
<tr>
<td>Create an approval process for firewall requests during a migration</td>
<td>975</td>
</tr>
<tr>
<td>Ingest EC2 Windows instances into an AMS account</td>
<td>981</td>
</tr>
<tr>
<td>Migrate a Linux server to Amazon EC2</td>
<td>984</td>
</tr>
<tr>
<td>Migrate an Azure VM to Amazon EC2 using CloudEndure</td>
<td>991</td>
</tr>
<tr>
<td>Migrate an F5 BIG-IP workload to F5 BIG-IP VE</td>
<td>1000</td>
</tr>
<tr>
<td>Migrate an on-premises Go application to AWS Elastic Beanstalk</td>
<td>1007</td>
</tr>
<tr>
<td>Migrate data to Amazon S3 using AWS SFTP</td>
<td>1014</td>
</tr>
<tr>
<td>Migrate from Oracle GlassFish to AWS Elastic Beanstalk</td>
<td>1017</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon EC2</td>
<td>1023</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon EC2 using CloudEndure</td>
<td>1026</td>
</tr>
<tr>
<td>Migrate from Oracle to Amazon EC2 using Oracle Data Pump</td>
<td>1031</td>
</tr>
<tr>
<td>Migrate from PostgreSQL to Amazon EC2</td>
<td>1035</td>
</tr>
<tr>
<td>Migrate from SAP ASE to Amazon EC2</td>
<td>1038</td>
</tr>
<tr>
<td>Migrate from SAP ASE to Amazon EC2 using CloudEndure</td>
<td>1045</td>
</tr>
<tr>
<td>Migrate from SQL Server to Amazon EC2</td>
<td>1048</td>
</tr>
<tr>
<td>Migrate from SQL Server to Amazon EC2 using CloudEndure</td>
<td>1054</td>
</tr>
<tr>
<td>Migrate from on-premises MariaDB to Amazon EC2</td>
<td>1059</td>
</tr>
<tr>
<td>Migrate from on-premises MySQL to Amazon EC2</td>
<td>1067</td>
</tr>
<tr>
<td>Rehost on-premises workloads on AWS: migration checklist</td>
<td>1075</td>
</tr>
<tr>
<td>Replatform</td>
<td>1099</td>
</tr>
<tr>
<td>Migrate RDS for Oracle to another AWS Region and account</td>
<td>1100</td>
</tr>
<tr>
<td>Migrate VMware SDDC to VMware Cloud on AWS</td>
<td>1105</td>
</tr>
<tr>
<td>Migrate an Amazon RDS DB instance to another VPC or account</td>
<td>1108</td>
</tr>
<tr>
<td>Migrate an Amazon RDS for Oracle DB to another VPC</td>
<td>1111</td>
</tr>
<tr>
<td>Transport PostgreSQL databases between Amazon RDS DB instances</td>
<td>1114</td>
</tr>
<tr>
<td>Relocate</td>
<td>1124</td>
</tr>
<tr>
<td>Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker</td>
<td>1130</td>
</tr>
<tr>
<td>Migrate OpenText TeamSite workloads to AWS</td>
<td>1131</td>
</tr>
<tr>
<td>Migrate openText TeamSite workloads to AWS</td>
<td>1137</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Centralized logging and multiple-account security</td>
<td>1638</td>
</tr>
<tr>
<td>Summary</td>
<td>1639</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1639</td>
</tr>
<tr>
<td>Architecture</td>
<td>1640</td>
</tr>
<tr>
<td>Tools</td>
<td>1640</td>
</tr>
<tr>
<td>Epics</td>
<td>1642</td>
</tr>
<tr>
<td>Related resources</td>
<td>1643</td>
</tr>
<tr>
<td>Attachments</td>
<td>1647</td>
</tr>
<tr>
<td>Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version</td>
<td>1647</td>
</tr>
<tr>
<td>Summary</td>
<td>1647</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1648</td>
</tr>
<tr>
<td>Architecture</td>
<td>1648</td>
</tr>
<tr>
<td>Tools</td>
<td>1649</td>
</tr>
<tr>
<td>Epics</td>
<td>1649</td>
</tr>
<tr>
<td>Related resources</td>
<td>1651</td>
</tr>
<tr>
<td>Attachments</td>
<td>1655</td>
</tr>
<tr>
<td>Check for single-host network entries in security group ingress rules for IPv4 and IPv6</td>
<td>1651</td>
</tr>
<tr>
<td>Summary</td>
<td>1652</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1652</td>
</tr>
<tr>
<td>Architecture</td>
<td>1652</td>
</tr>
<tr>
<td>Tools</td>
<td>1652</td>
</tr>
<tr>
<td>Epics</td>
<td>1653</td>
</tr>
<tr>
<td>Related resources</td>
<td>1655</td>
</tr>
<tr>
<td>Attachments</td>
<td>1655</td>
</tr>
<tr>
<td>Enable GuardDuty using CloudFormation templates</td>
<td>1655</td>
</tr>
<tr>
<td>Summary</td>
<td>1655</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1655</td>
</tr>
<tr>
<td>Architecture</td>
<td>1656</td>
</tr>
<tr>
<td>Tools</td>
<td>1656</td>
</tr>
<tr>
<td>Epics</td>
<td>1657</td>
</tr>
<tr>
<td>Related resources</td>
<td>1658</td>
</tr>
<tr>
<td>Additional information</td>
<td>1658</td>
</tr>
<tr>
<td>Enable transparent data encryption in Amazon RDS for SQL Server</td>
<td>1660</td>
</tr>
<tr>
<td>Summary</td>
<td>1661</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1661</td>
</tr>
<tr>
<td>Architecture</td>
<td>1661</td>
</tr>
<tr>
<td>Tools</td>
<td>1662</td>
</tr>
<tr>
<td>Epics</td>
<td>1663</td>
</tr>
<tr>
<td>Related resources</td>
<td>1664</td>
</tr>
<tr>
<td>Ensure AWS CloudFormation stacks are launched from authorized S3 buckets</td>
<td>1664</td>
</tr>
<tr>
<td>Summary</td>
<td>1665</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1665</td>
</tr>
<tr>
<td>Architecture</td>
<td>1665</td>
</tr>
<tr>
<td>Tools</td>
<td>1666</td>
</tr>
<tr>
<td>Epics</td>
<td>1666</td>
</tr>
<tr>
<td>Related resources</td>
<td>1667</td>
</tr>
<tr>
<td>Additional information</td>
<td>1667</td>
</tr>
<tr>
<td>Attachments</td>
<td>1667</td>
</tr>
<tr>
<td>Ensure AWS load balancers use secure listener protocols</td>
<td>1667</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Monitor Amazon ElastiCache clusters for at-rest encryption</td>
<td>1696</td>
</tr>
<tr>
<td>Manage credentials with AWS Secrets Manager</td>
<td>1687</td>
</tr>
<tr>
<td>Help prevent scheduled KMS key deletion</td>
<td>1682</td>
</tr>
<tr>
<td>Ensure that new Amazon Redshift clusters are encrypted</td>
<td>1679</td>
</tr>
<tr>
<td>Ensure that an IAM profile is associated with an EC2 instance</td>
<td>1675</td>
</tr>
<tr>
<td>Ensure encryption for Amazon EMR data at rest</td>
<td>1671</td>
</tr>
<tr>
<td>Monitor Amazon EMR clusters for in-transit encryption at launch</td>
<td>1692</td>
</tr>
<tr>
<td>Monitor Amazon ElastiCache clusters for at-rest encryption</td>
<td>1696</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1730</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Architecture</td>
<td>1730</td>
</tr>
<tr>
<td>Tools</td>
<td>1732</td>
</tr>
<tr>
<td>Epics</td>
<td>1733</td>
</tr>
<tr>
<td>Related resources</td>
<td>1734</td>
</tr>
</tbody>
</table>

**Use Network Firewall to capture DNS domain names from outbound network traffic**

<table>
<thead>
<tr>
<th>Summary</th>
<th>1735</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites and limitations</td>
<td>1735</td>
</tr>
<tr>
<td>Architecture</td>
<td>1735</td>
</tr>
<tr>
<td>Tools</td>
<td>1736</td>
</tr>
<tr>
<td>Epics</td>
<td>1736</td>
</tr>
</tbody>
</table>

**More patterns**

<table>
<thead>
<tr>
<th>Serverless</th>
<th>1745</th>
</tr>
</thead>
</table>

**Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Kinesis Data Firehose**

<table>
<thead>
<tr>
<th>Summary</th>
<th>1747</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites and limitations</td>
<td>1748</td>
</tr>
<tr>
<td>Architecture</td>
<td>1748</td>
</tr>
<tr>
<td>Tools</td>
<td>1749</td>
</tr>
<tr>
<td>Related resources</td>
<td>1751</td>
</tr>
</tbody>
</table>

**Run Systems Manager automation tasks synchronously from Step Functions**

<table>
<thead>
<tr>
<th>Summary</th>
<th>1751</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites and limitations</td>
<td>1752</td>
</tr>
<tr>
<td>Architecture</td>
<td>1752</td>
</tr>
<tr>
<td>Tools</td>
<td>1753</td>
</tr>
<tr>
<td>Epics</td>
<td>1754</td>
</tr>
<tr>
<td>Related resources</td>
<td>1756</td>
</tr>
<tr>
<td>Additional information</td>
<td>1757</td>
</tr>
</tbody>
</table>

**More patterns**

<table>
<thead>
<tr>
<th>Tools</th>
<th>1759</th>
</tr>
</thead>
</table>

**Software development & testing**

<table>
<thead>
<tr>
<th>Run unit tests by using AWS CodeBuild</th>
<th>1761</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>1761</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1761</td>
</tr>
<tr>
<td>Architecture</td>
<td>1761</td>
</tr>
<tr>
<td>Tools</td>
<td>1762</td>
</tr>
<tr>
<td>Epics</td>
<td>1762</td>
</tr>
<tr>
<td>Related resources</td>
<td>1764</td>
</tr>
<tr>
<td>Additional information</td>
<td>1765</td>
</tr>
</tbody>
</table>

**More patterns**

<table>
<thead>
<tr>
<th>Tools</th>
<th>1767</th>
</tr>
</thead>
</table>

**Storage & backup**

<table>
<thead>
<tr>
<th>Allow EC2 instances write access to S3 buckets in AMS</th>
<th>1768</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>1768</td>
</tr>
<tr>
<td>Prerequisites and limitations</td>
<td>1768</td>
</tr>
<tr>
<td>Architecture</td>
<td>1769</td>
</tr>
<tr>
<td>Tools</td>
<td>1769</td>
</tr>
<tr>
<td>Epics</td>
<td>1769</td>
</tr>
<tr>
<td>Related resources</td>
<td>1771</td>
</tr>
</tbody>
</table>

**Automate data stream ingestion into a Snowflake database**

<table>
<thead>
<tr>
<th>Summary</th>
<th>1771</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites and limitations</td>
<td>1772</td>
</tr>
<tr>
<td>Architecture</td>
<td>1772</td>
</tr>
<tr>
<td>Tools</td>
<td>1773</td>
</tr>
<tr>
<td>Epics</td>
<td>1773</td>
</tr>
<tr>
<td>Related resources</td>
<td>1776</td>
</tr>
<tr>
<td>Additional information</td>
<td>1776</td>
</tr>
</tbody>
</table>

**Automatically encrypt EBS volumes**

<table>
<thead>
<tr>
<th>Summary</th>
<th>1779</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Attachments</td>
<td>6</td>
</tr>
<tr>
<td>More patterns</td>
<td>1859</td>
</tr>
<tr>
<td>Document history</td>
<td>1860</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns

August 2019

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

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

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

For more information about how to plan your migration to the AWS Cloud, see Mobilize Your Organization to Accelerate Large-Scale Migrations.
Enforce tagging of Amazon EMR clusters at launch

Created by Priyanka Chaudhary (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Technologies:</th>
<th>Analytics; Security, identity, compliance</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>AWS services:</th>
<th>Amazon EMR; AWS Lambda; Amazon CloudWatch Events</th>
</tr>
</thead>
</table>

Summary

This pattern provides a security control that ensures that Amazon EMR clusters are tagged when they are created.

Amazon EMR is an Amazon Web Services (AWS) service for processing and analyzing vast amounts of data. Amazon EMR offers an expandable, low-configuration service as an easier alternative to running in-house cluster computing. You can use tagging to categorize AWS resources in different ways, such as by purpose, owner, or environment. For example, you can tag your Amazon EMR clusters by assigning custom metadata to each cluster. A tag consists of a key and value that you define. We recommend that
you create a consistent set of tags to meet your organization's requirements. When you add a tag to an Amazon EMR cluster, the tag is also propagated to each active Amazon Elastic Compute Cloud (Amazon EC2) instance that is associated with the cluster. Similarly, when you remove a tag from an Amazon EMR cluster, that tag is removed from each associated, active EC2 instance as well.

The detective control monitors API calls and initiates an Amazon CloudWatch Events event for the RunJobFlow, AddTags, RemoveTags, and CreateTags APIs. The event calls AWS Lambda, which runs a Python script. The Python function gets the Amazon EMR cluster ID from the JSON input from the event and performs the following checks:

- Check if the Amazon EMR cluster is configured with tag names that you specify.
- If not, send an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information: the Amazon EMR cluster name, violation details, AWS Region, AWS account, and Amazon Resource Name (ARN) for Lambda that this notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the provided Lambda code. Or, you can create an S3 bucket for this purpose, as described in the Epics section.
- An active email address where you would like to receive violation notifications.
- A list of mandatory tags you want to check for.

Limitations

- This security control is regional. You must deploy it in each AWS Region that you want to monitor.

Product versions

- Amazon EMR release 4.8.0 and later.

Architecture

Workflow architecture

Automation and scale
If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **Amazon CloudWatch Events** - Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.

- **Amazon EMR** - Amazon EMR is a web service that simplifies running big data frameworks and processing vast amounts of data efficiently.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes the following attachments:

- **EMRTagValidation.zip** – The Lambda code for the security control.
- **EMRTagValidation.yml** – The CloudFormation template that sets up the event and Lambda function.

Epics

Set up the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket to host the Lambda code.zip file. This S3 bucket must be in the same AWS Region as the Amazon EMR cluster you want to monitor. An Amazon S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Upload the Lambda code.</td>
<td>Upload the Lambda code .zip file provided in the Attachments section to the S3 bucket.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

### Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the AWS CloudFormation template.</td>
<td>Open the <a href="https://aws.amazon.com/cloudformation/">AWS CloudFormation console</a> in the same AWS Region as your S3 bucket and deploy the template. For more information about deploying AWS CloudFormation templates, see Creating a stack on the AWS CloudFormation console in the CloudFormation documentation.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
| Complete the parameters in the template.  | When you launch the template, you’ll be prompted for the following information:  
  • **S3 bucket**: Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (.zip file).  
  • **S3 key**: Specify the location of the Lambda .zip file in your S3 bucket (for example, *filename.zip* or *controls/filename.zip*). Do not include leading slashes.  
  • **Notification email**: Provide an active email address where you want to receive Amazon SNS notifications.  
  • **Tagging key names**: Provide the tags you want to check for, in a comma-separated list (for example, ApplicationID, Environment, Owner). The CloudWatch Events event monitors the cluster for these tags and sends a notification if they aren't found.  
  • **Lambda logging level**: Specify the logging level and frequency for the Lambda function. Use Info to log                                                                                                                                                                                                                                                                                                                                 | Cloud architect   |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>detailed informational messages on progress, <strong>Error</strong> for error events that would still allow the deployment to continue, and <strong>Warning</strong> for potentially harmful situations.</td>
<td>Cloud architect</td>
<td></td>
</tr>
</tbody>
</table>

**Confirm the subscription**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email to the email address you provided. You must confirm this email subscription to start receiving violation notifications.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Related resources**

- AWS Lambda developer guide
- Tagging clusters in Amazon EMR

**Attachments**

attachment.zip

**Analyze Amazon Redshift data in Microsoft SQL Server Analysis Services**

*Created by Sunil Vora (AWS)*

<table>
<thead>
<tr>
<th>R Type: N/A</th>
<th>Source: Amazon Redshift</th>
<th>Target: Microsoft SQL Server Analysis Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Analytics</td>
</tr>
<tr>
<td>Workload: Microsoft</td>
<td>AWS services: Amazon Redshift</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes how to connect and analyze Amazon Redshift data in Microsoft SQL Server Analysis Services, by using the Intellisoft OLE DB Provider for database access.
Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. SQL Server Analysis Services is an online analytical processing (OLAP) tool that you can use to analyze data from data marts and data warehouses such as Amazon Redshift. You can use SQL Server Analysis Services to create OLAP cubes from your data for rapid, advanced data analysis.

**Prerequisites and limitations**

**Assumptions**

- This pattern describes how to set up SQL Server Analysis Services and Intellisoft OLE DB Provider for Amazon Redshift on an Amazon Elastic Compute Cloud (Amazon EC2) instance. Alternatively, you can install both on a host in your corporate data center.

**Prerequisites**

- An active AWS account
- An Amazon Redshift cluster with credentials

**Architecture**

**Source technology stack**

- An Amazon Redshift cluster

**Target technology stack**

- Microsoft SQL Server Analysis Services

**Source and target architecture**
Tools

- Microsoft Visual Studio 2019 (Community Edition)
- Intellisoft OLE DB Provider for Amazon Redshift (Trial)

Epics

Analyze tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the Amazon Redshift tables to be imported and their sizes.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Set up EC2 instance and install tools

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up an EC2 instance.</td>
<td>In your AWS account, create an EC2 instance in a private or public subnet.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Install the OLE DB Provider</td>
<td>Download and install the Intellisoft OLE DB Provider for Amazon Redshift. For a link, see the References section.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Install Visual Studio.</td>
<td>Download and install Visual Studio 2019 (Community Edition). For a link, see the References section.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Install extensions.</td>
<td>Install the “Microsoft Analysis Services Projects” extension in Visual Studio.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create a project.</td>
<td>Create a new tabular model project in Visual Studio to store your Amazon Redshift data. In Visual Studio, choose the “Analysis Services Tabular Project” option when creating your project.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Create data source and import tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon Redshift data source.</td>
<td>Create an Amazon Redshift data source by using the Intellisoft OLE DB Provider for Amazon Redshift</td>
<td>Redshift, DBA</td>
</tr>
</tbody>
</table>
### Related resources

- Amazon Redshift (AWS documentation)
- Install SQL Server Analysis Services (Microsoft documentation)
- Tabular Model Designer (Microsoft documentation)
- Overview of OLAP cubes for advanced analytics (Microsoft documentation)
- Microsoft Visual Studio 2019 (Community Edition)
- Intellisoft OLE DB Provider for Amazon Redshift (Trial)

### Automate data loading from Amazon S3 to Amazon Redshift using AWS Data Pipeline

*Created by Burada Kiran (AWS)*

<table>
<thead>
<tr>
<th>Created by</th>
<th>Environment</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>PoC or pilot</td>
<td>Storage &amp; backup; Databases; Analytics</td>
</tr>
<tr>
<td>AWS services</td>
<td>Amazon S3; Amazon Redshift</td>
<td></td>
</tr>
</tbody>
</table>

### Summary

This pattern walks you through the AWS data migration process from an Amazon Simple Storage Service (Amazon S3) bucket to Amazon Redshift using AWS Data Pipeline.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account
• An S3 source bucket with the right privileges

**Architecture**

**Source technology stack**
• An S3 bucket with CSV files

**Target technology stack**
• An Amazon Redshift cluster

**Data migration architecture**

**Tools**
• **Data Pipeline** - You can use AWS Data Pipeline to automate the movement and transformation of data. With Data Pipeline, you can define data-driven workflows so that tasks can proceed after the successful completion of previous tasks.
## Epics

### Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the version and engine of the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an outbound security group to source and target databases.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Prepare the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Extract users, roles, and grants list from the source.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Create users in the target database.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Apply roles from the previous step to the target database.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Review database options, parameters, network files, and database links from the source, and evaluate their applicability to the target database.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
</tbody>
</table>

### Configure the pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new pipeline in AWS Data Pipeline.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>For source, choose the option to load data from Amazon S3 into an Amazon Redshift template.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>For parameters, provide the source and target details.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Schedule and choose an AWS Data Pipeline activation.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>For Security/Access, leave the AWS Identity and Access</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>
Related resources

- Amazon S3 documentation
- AWS Data Pipeline documentation
- AWS Data Pipeline pricing
- Amazon Redshift pricing

Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue

Created by Rohan Jamadagni (AWS) and Arunabha Datta (AWS)

Created by: AWS  Environment: Production  Technologies: Data lakes; Storage & backup; Analytics

AWS services: Amazon Redshift; Amazon S3; AWS Glue; AWS Lambda

Summary

This pattern provides guidance on how to configure Amazon Simple Storage Service (Amazon S3) for optimal data lake performance, and then load incremental data changes from Amazon S3 into Amazon Redshift by using AWS Glue, performing extract, transform, and load (ETL) operations.

The source files in Amazon S3 can have different formats, including comma-separated values (CSV), XML, and JSON files. This pattern describes how you can use AWS Glue to convert the source files into a cost-optimized and performance-optimized format like Apache Parquet. You can query Parquet files
directly from Amazon Athena and Amazon Redshift Spectrum. You can also load Parquet files into Amazon Redshift, aggregate them, and share the aggregated data with consumers, or visualize the data by using Amazon QuickSight.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An S3 source bucket that has the right privileges and contains CSV, XML, or JSON files.

Assumptions

- The CSV, XML, or JSON source files are already loaded into Amazon S3 and are accessible from the account where AWS Glue and Amazon Redshift are configured.
- Best practices for loading the files, splitting the files, compression, and using a manifest are followed, as discussed in the Amazon Redshift documentation.
- The source file structure is unaltered.
- The source system is able to ingest data into Amazon S3 by following the folder structure defined in Amazon S3.
- The Amazon Redshift cluster spans a single Availability Zone. (This architecture is appropriate because AWS Lambda, AWS Glue, and Amazon Athena are serverless.) For high availability, cluster snapshots are taken at a regular frequency.

Limitations

- The file formats are limited to those that are currently supported by AWS Glue.
- Real-time downstream reporting isn’t supported.

Architecture

Source technology stack

- S3 bucket with CSV, XML, or JSON files

Target technology stack

- S3 data lake (with partitioned Parquet file storage)
- Amazon Redshift

Target architecture
Tools

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service. Amazon S3 can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

- **AWS Lambda** – AWS Lambda lets you run code without provisioning or managing servers. AWS Lambda is an event-driven service; you can set up your code to automatically initiate from other AWS services.
• **Amazon Redshift** – Amazon Redshift is a fully managed, petabyte-scale data warehouse service. With Amazon Redshift, you can query petabytes of structured and semi-structured data across your data warehouse and your data lake using standard SQL.

• **AWS Glue** – AWS Glue is a fully managed ETL service that makes it easier to prepare and load data for analytics. AWS Glue discovers your data and stores the associated metadata (for example, table definitions and schema) in the AWS Glue Data Catalog. Your cataloged data is immediately searchable, can be queried, and is available for ETL.

• **AWS Secrets Manager** – AWS Secrets Manager facilitates protection and central management of secrets needed for application or service access. The service stores database credentials, API keys, and other secrets, and eliminates the need to hardcode sensitive information in plaintext format. Secrets Manager also offers key rotation to meet security and compliance needs. It has built-in integration for Amazon Redshift, Amazon Relational Database Service (Amazon RDS), and Amazon DocumentDB. You can store and centrally manage secrets by using the Secrets Manager console, the command-line interface (CLI), or Secrets Manager API and SDKs.

• **Amazon Athena** – Amazon Athena is an interactive query service that makes it easy to analyze data that's stored in Amazon S3. Athena is serverless and integrated with AWS Glue, so it can directly query the data that's cataloged using AWS Glue. Athena is elastically scaled to deliver interactive query performance.

**Epics**

Create S3 buckets and folder structure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze source systems for data structure and attributes.</td>
<td>Perform this task for each data source that contributes to the Amazon S3 data lake.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Define the partition and access strategy.</td>
<td>This strategy should be based on the frequency of data captures, delta processing, and consumption needs. Make sure that S3 buckets are not open to the public and that access is controlled by specific service role-based policies only. For more information, see the Amazon S3 documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create separate S3 buckets for each data source type and a separate S3 bucket per source for the processed (Parquet) data.</td>
<td>Create a separate bucket for each source, and then create a folder structure that's based on the source system's data ingestion frequency; for example: s3://source-system-name/year/month/day/hour/. For the processed (converted to Parquet format) files, create a similar structure; for example: s3://source-processed-bucket/year/month/day/hour. For more information about creating S3</td>
<td>Data engineer</td>
</tr>
</tbody>
</table>
### Create a secret in AWS Secrets Manager

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new secret to store the Amazon Redshift user name and password in Secrets Manager.</td>
<td>This secret stores the credentials for the admin user as well as individual database service users. For instructions, see the Secrets Manager documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create an AWS Identity and Access Management (IAM) policy to restrict Secrets Manager access.</td>
<td>Restrict Secrets Manager access to only Amazon Redshift administrators and AWS Glue.</td>
<td>Data engineer</td>
</tr>
</tbody>
</table>

### Create a data warehouse in Amazon Redshift

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Amazon Redshift cluster with the appropriate parameter groups and maintenance and backup strategy.</td>
<td>Use the Secrets Manager database secret for admin user credentials while creating the Amazon Redshift cluster. For information about creating and sizing an Amazon Redshift cluster, see the Amazon Redshift documentation and the Sizing Cloud Data Warehouses whitepaper.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create and attach the IAM service role to the Amazon Redshift cluster.</td>
<td>This ensures access to Secrets Manager and the source S3 buckets. For more information, see the AWS documentation on authorization and adding a role.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create a database user with the appropriate roles and permissions to access the corresponding database schema objects.</td>
<td>When creating the database user, refer to the secret stored in Secrets Manager for the service user. This service user will be used by AWS Glue. For more information, see the Amazon Redshift documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create the database schema.</td>
<td>Follow Amazon Redshift best practices for table design. Based on the use case, choose the appropriate sort and distribution</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Configure workload management (WLM) queues, short query acceleration (SQA), or concurrency scaling, depending on your requirements.</td>
<td>For more information, see Implementing workload management in the Amazon Redshift documentation.</td>
<td>Data engineer</td>
</tr>
</tbody>
</table>

**Configure AWS Glue**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the AWS Glue Data Catalog, add a connection for Amazon Redshift.</td>
<td>For instructions, see the AWS Glue documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create and attach an IAM service role for AWS Glue to access Secrets Manager, Amazon Redshift, and S3 buckets.</td>
<td>For more information, see the AWS Glue documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Define the AWS Glue Data Catalog for the source.</td>
<td>This step involves creating a database and required tables in the AWS Glue Data Catalog. You can either use a crawler to catalog the tables in the AWS Glue database, or define them as Amazon Athena external tables. You can also access the external tables defined in Athena through the AWS Glue Data Catalog. See the AWS documentation for more information about defining the Data Catalog and creating an external table in Athena.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create an AWS Glue job to process source data.</td>
<td>The AWS Glue job can be a Python shell or PySpark to standardize, deduplicate, and cleanse the source data files. To optimize performance and avoid having to query the entire S3 source bucket, partition the S3 bucket by date, broken down by year, month, day, and hour as a pushdown predicate for the AWS Glue job. For more information, see the AWS Glue documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Create an AWS Glue job to load data into Amazon Redshift.</td>
<td>The AWS Glue job can be a Python shell or PySpark to load the data by upserting the data, followed by a complete refresh. For details, see the <a href="https://docs.aws.amazon.com/glue/latest/dg/">AWS Glue documentation</a> and the <em>Additional information</em> section.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>(Optional) Schedule AWS Glue jobs by using triggers as necessary.</td>
<td>The incremental data load is primarily driven by an Amazon S3 event that causes an AWS Lambda function to call the AWS Glue job. Use AWS Glue trigger-based scheduling for any data loads that demand time-based instead of event-based scheduling.</td>
<td>Data engineer</td>
</tr>
</tbody>
</table>

### Create a Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and attach an IAM service-linked role for AWS Lambda to access S3 buckets and the AWS Glue job.</td>
<td>Create an IAM service-linked role for AWS Lambda with a policy to read Amazon S3 objects and buckets, and a policy to access the AWS Glue API to start an AWS Glue job. For more information, see the Knowledge Center.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Create a Lambda function to run the AWS Glue job based on the defined Amazon S3 event.</td>
<td>The Lambda function should be initiated by the creation of the Amazon S3 manifest file. The Lambda function should pass the Amazon S3 folder location (for example, source_bucket/year/month/date/hour) to the AWS Glue job as a parameter. The AWS Glue job will use this parameter as a pushdown predicate to optimize file access and job processing performance. For more information, see the <a href="https://docs.aws.amazon.com/glue/latest/dg/">AWS Glue documentation</a>.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Create an Amazon S3 PUT object event to detect object creation, and call the respective Lambda function.</td>
<td>The Amazon S3 PUT object event should be initiated only by the creation of the manifest file. The manifest file controls the Lambda function and the AWS Glue job concurrency, and processes the load as a batch instead of processing individual files that arrive in a specific partition of the S3 source bucket. For more information, see the <a href="https://docs.aws.amazon.com/lambda/userguide/">Lambda documentation</a>.</td>
<td>Data engineer</td>
</tr>
</tbody>
</table>

**Related resources**

- Amazon S3 documentation
- AWS Glue documentation
- Amazon Redshift documentation
- AWS Lambda
- Amazon Athena
- AWS Secrets Manager

**Additional information**

**Detailed approach for upsert and complete refresh**

**Upsert:** This is for datasets that require historical aggregation, depending on the business use case. Follow one of the approaches described in [Updating and inserting new data](https://docs.aws.amazon.com/redshift/latest/dg/using-updating-data.html) (Amazon Redshift documentation) based on your business needs.

**Complete refresh:** This is for small datasets that don't need historical aggregations. Follow one of these approaches:

1. Truncate the Amazon Redshift table.
2. Load the current partition from the staging area

or:

1. Create a temporary table with current partition data.
2. Drop the target Amazon Redshift table.
3. Rename the temporary table to the target table.

**Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL**
Summary

NORMALIZE is a Teradata extension to the ANSI SQL standard. When a SQL table includes a column that has a PERIOD data type, NORMALIZE combines values that meet or overlap in that column, to form a single period that consolidates multiple, individual period values. To use NORMALIZE, at least one column in the SQL SELECT list must be of Teradata's temporal PERIOD data type. For more information about NORMALIZE, see the Teradata documentation.

Amazon Redshift doesn't support NORMALIZE, but you can implement this functionality by using native SQL syntax and the LAG window function in Amazon Redshift. This pattern focuses on using the Teradata NORMALIZE extension with the ON MEETS OR OVERLAPS condition, which is the most popular format. It explains how this feature works in Teradata and how it can be converted into Amazon Redshift native SQL syntax.

Prerequisites and limitations

Prerequisites

• Basic Teradata SQL knowledge and experience
• Amazon Redshift knowledge and experience

Architecture

Source technology stack

• Teradata data warehouse

Target technology stack

• Amazon Redshift

Target architecture

For a high-level architecture for migrating a Teradata database to Amazon Redshift, see the pattern Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents. The migration doesn't automatically convert the Teradata NORMALIZE phrase to Amazon Redshift SQL. You can convert this Teradata extension by following the guidelines in this pattern.

Tools

Code
To illustrate the concept and functionality of NORMALIZE, consider the following table definition in Teradata:

```sql
CREATE TABLE systest.project
(
  emp_id INTEGER,
  project_name VARCHAR(20),
  dept_id INTEGER,
  duration PERIOD(DATE)
);
```

Run the following SQL code to insert sample data into the table:

```sql
BEGIN TRANSACTION;
INSERT INTO systest.project VALUES (10, 'First Phase', 1000, PERIOD(DATE '2010-01-10', DATE '2010-03-20'));
INSERT INTO systest.project VALUES (10, 'First Phase', 2000, PERIOD(DATE '2010-03-20', DATE '2010-07-15'));
INSERT INTO systest.project VALUES (10, 'Second Phase', 2000, PERIOD(DATE '2010-06-15', DATE '2010-08-18'));
INSERT INTO systest.project VALUES (20, 'First Phase', 2000, PERIOD(DATE '2010-03-10', DATE '2010-07-20'));
INSERT INTO systest.project VALUES (20, 'Second Phase', 1000, PERIOD(DATE '2020-05-10', DATE '2020-09-20'));
END TRANSACTION;
```

Results:

```
select * from systest.project order by 1,2,3;
```

```sql
*** Query completed. 4 rows found. 4 columns returned.
*** Total elapsed time was 1 second.

<table>
<thead>
<tr>
<th>emp_id</th>
<th>project_name</th>
<th>dept_id</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>First Phase</td>
<td>1000</td>
<td>('10/01/10', '10/03/20')</td>
</tr>
<tr>
<td>10</td>
<td>First Phase</td>
<td>2000</td>
<td>('10/03/20', '10/07/15')</td>
</tr>
<tr>
<td>10</td>
<td>Second Phase</td>
<td>2000</td>
<td>('10/06/15', '10/08/18')</td>
</tr>
<tr>
<td>20</td>
<td>First Phase</td>
<td>2000</td>
<td>('10/03/10', '10/07/20')</td>
</tr>
<tr>
<td>20</td>
<td>Second Phase</td>
<td>1000</td>
<td>('20/05/10', '20/09/20')</td>
</tr>
</tbody>
</table>
```

**Teradata NORMALIZE use case**

Now add the Teradata NORMALIZE SQL clause to the SELECT statement:

```sql
SELECT NORMALIZE ON MEETS OR OVERLAPS emp_id, duration
FROM systest.project
ORDER BY 1,2;
```

This NORMALIZE operation is performed on a single column (emp_id). For emp_id=10, the three overlapping period values in duration coalesce into a single period value, as follows:

```sql
<table>
<thead>
<tr>
<th>emp_id</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>('10/01/10', '10/08/18')</td>
</tr>
<tr>
<td>20</td>
<td>('10/03/10', '10/07/20')</td>
</tr>
</tbody>
</table>
```
The following SELECT statement performs a **NORMALIZE** operation on project_name and dept_id. Note that the SELECT list contains only one PERIOD column, duration.

```
SELECT NORMALIZE project_name, dept_id, duration
FROM systest.project;
```

Output:

```
project_name      dept_id  duration
--------------------  -----------  ------------------------
First Phase        1000 ('10/01/10', '10/03/20')
Second Phase       1000 ('20/05/10', '20/09/20')
First Phase        2000 ('10/03/10', '10/07/20')
Second Phase       2000 ('10/06/15', '10/08/18')
```

**Amazon Redshift equivalent SQL**

Amazon Redshift currently doesn't support the PERIOD data type in a table. Instead, you need to divide a Teradata PERIOD data field into two parts: start_date, end_date, as follows:

```
CREATE TABLE systest.project
(   emp_id INTEGER,
    project_name VARCHAR(20),
    dept_id INTEGER,
    start_date DATE,
    end_date DATE
);
```

Insert sample data into the table:

```
BEGIN TRANSACTION;

INSERT INTO systest.project VALUES (10, 'First Phase', 1000, DATE '2010-01-10', DATE '2010-03-20');
INSERT INTO systest.project VALUES (10, 'First Phase', 2000, DATE '2010-03-20', DATE '2010-07-15');
INSERT INTO systest.project VALUES (10, 'Second Phase', 2000, DATE '2010-06-15', DATE '2010-08-18');
INSERT INTO systest.project VALUES (20, 'First Phase', 2000, DATE '2010-03-10', DATE '2010-07-20');
INSERT INTO systest.project VALUES (20, 'Second Phase', 1000, DATE '2020-05-10', DATE '2020-09-20');

END TRANSACTION;
```

Output:

```
emp_id | project_name | dept_id | start_date | end_date
--------+--------------+---------+------------+------------
10 | First Phase  | 1000 | 2010-01-10 | 2010-03-20
10 | First Phase  | 2000 | 2010-03-20 | 2010-07-15
10 | Second Phase | 2000 | 2010-06-15 | 2010-08-18
20 | First Phase  | 2000 | 2010-03-10 | 2010-07-20
20 | Second Phase | 1000 | 2020-05-10 | 2020-09-20
```

(5 rows)
To rewrite Teradata’s `NORMALIZE` clause, you can use the `LAG` window function in Amazon Redshift. This function returns the values for a row at a given offset above (before) the current row in the partition.

You can use the `LAG` function to identify each row that begins a new period by determining if a period meets or overlaps with the previous period (0 if yes and 1 if no). When this flag is cumulatively summed up, it provides a group identifier that can be used in the outer `Group By` clause to arrive at the desired result in Amazon Redshift.

Here’s a sample Amazon Redshift SQL statement that uses `LAG()`:

```sql
SELECT emp_id, start_date, end_date,
  (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY emp_id ORDER BY start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
FROM systest.project
ORDER BY 1,2;
```

Output:

<table>
<thead>
<tr>
<th>emp_id</th>
<th>start_date</th>
<th>end_date</th>
<th>groupstartflag</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2010-01-10</td>
<td>2010-03-20</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2010-03-20</td>
<td>2010-07-15</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2010-06-15</td>
<td>2010-08-18</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>2010-03-10</td>
<td>2010-07-20</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2020-05-10</td>
<td>2020-09-20</td>
<td>1</td>
</tr>
</tbody>
</table>

(5 rows)

The following Amazon Redshift SQL statement normalizes only on the `emp_id` column:

```sql
SELECT T2.emp_id, MIN(T2.start_date) as new_start_date, MAX(T2.end_date) as new_end_date
FROM
  ( SELECT T1.*, SUM(GroupStartFlag) OVER (PARTITION BY emp_id ORDER BY start_date ROWS UNBOUNDED PRECEDING) As GroupID
    FROM ( SELECT emp_id, start_date, end_date,
      (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY emp_id ORDER BY start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
    FROM systest.project ) T1
  ) T2
GROUP BY T2.emp_id, T2.GroupID
ORDER BY 1,2;
```

Output:

<table>
<thead>
<tr>
<th>emp_id</th>
<th>new_start_date</th>
<th>new_end_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2010-01-10</td>
<td>2010-08-18</td>
</tr>
<tr>
<td>20</td>
<td>2010-03-10</td>
<td>2010-07-20</td>
</tr>
<tr>
<td>20</td>
<td>2020-05-10</td>
<td>2020-09-20</td>
</tr>
</tbody>
</table>

(3 rows)

The following Amazon Redshift SQL statement normalizes on both the `project_name` and `dept_id` columns:

```sql
SELECT T2.project_name, T2.dept_id, MIN(T2.start_date) as new_start_date, MAX(T2.end_date) as new_end_date
FROM
```

(5 rows)
Output:

<table>
<thead>
<tr>
<th>project_name</th>
<th>dept_id</th>
<th>new_start_date</th>
<th>new_end_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Phase</td>
<td>1000</td>
<td>2010-01-10</td>
<td>2010-03-20</td>
</tr>
<tr>
<td>First Phase</td>
<td>2000</td>
<td>2010-03-10</td>
<td>2010-07-20</td>
</tr>
<tr>
<td>Second Phase</td>
<td>1000</td>
<td>2020-05-10</td>
<td>2020-09-20</td>
</tr>
<tr>
<td>Second Phase</td>
<td>2000</td>
<td>2010-06-15</td>
<td>2010-08-18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4 rows)</td>
<td></td>
</tr>
</tbody>
</table>

**Epics**

**Convert NORMALIZE to Amazon Redshift SQL**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create your Teradata SQL code.</td>
<td>Use the NORMALIZE phrase according to your needs.</td>
<td>SQL developer</td>
</tr>
<tr>
<td>Convert the code to Amazon Redshift SQL.</td>
<td>To convert your code, follow the guidelines in the &quot;Tools&quot; section of this pattern.</td>
<td>SQL developer</td>
</tr>
<tr>
<td>Run the code in Amazon Redshift.</td>
<td>Create your table, load data into the table, and run your code in Amazon Redshift.</td>
<td>SQL developer</td>
</tr>
</tbody>
</table>

**Related resources**

**References**

- Teradata NORMALIZE temporal feature (Teradata documentation)
- LAG window function (Amazon Redshift documentation)
- Migrate to Amazon Redshift (AWS website)
- Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents (AWS Prescriptive Guidance)
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL (AWS Prescriptive Guidance)

**Tools**

- AWS Schema Conversion Tool (AWS SCT)

**Partners**
Convert the Teradata RESET WHEN feature to Amazon Redshift SQL

*Created by Po Hong (AWS)*

<table>
<thead>
<tr>
<th>Source: Teradata data warehouse</th>
<th>Target: Amazon Redshift</th>
<th>R Type: Re-architect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: Production</td>
<td>Technologies: Analytics; Databases; Migration</td>
<td>Workload: All other workloads</td>
</tr>
<tr>
<td>AWS services: Amazon Redshift</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

**RESET WHEN** is a Teradata feature used in SQL analytical window functions. It is an extension to the ANSI SQL standard. **RESET WHEN** determines the partition over which an SQL window function operates based on some specified condition. If the condition evaluates to **TRUE**, a new, dynamic sub-partition is created inside the existing window partition. For more information about **RESET WHEN**, see the Teradata documentation.

Amazon Redshift doesn't support **RESET WHEN** in SQL window functions. To implement this functionality, you have to convert **RESET WHEN** to the native SQL syntax in Amazon Redshift, and use multiple, nested functions. This pattern demonstrates how you can use the Teradata **RESET WHEN** feature and how you can convert it to Amazon Redshift SQL syntax.

**Prerequisites and limitations**

**Prerequisites**

- Basic knowledge of the Teradata data warehouse and its SQL syntax
- Good understanding of Amazon Redshift and its SQL syntax

**Architecture**

**Source technology stack**

- Teradata data warehouse

**Target technology stack**

- Amazon Redshift
For a high-level architecture for migrating a Teradata database to Amazon Redshift, see the pattern Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents. The migration doesn't automatically convert the Teradata **RESET WHEN** phrase to Amazon Redshift SQL. You can convert this Teradata extension by following the guidelines in the next section.

### Tools

#### Code

To illustrate the concept of **RESET WHEN**, consider the following table definition in Teradata:

```sql
create table systest.f_account_balance
( account_id integer NOT NULL,
  month_id integer,
  balance integer )
unique primary index (account_id, month_id);
```

Run the following SQL code to insert sample data into the table:

```sql
BEGIN TRANSACTION;
Insert Into systest.f_account_balance values (1,1,60);
Insert Into systest.f_account_balance values (1,2,99);
Insert Into systest.f_account_balance values (1,3,94);
Insert Into systest.f_account_balance values (1,4,90);
Insert Into systest.f_account_balance values (1,5,80);
Insert Into systest.f_account_balance values (1,6,88);
Insert Into systest.f_account_balance values (1,7,90);
Insert Into systest.f_account_balance values (1,8,92);
Insert Into systest.f_account_balance values (1,9,10);
Insert Into systest.f_account_balance values (1,10,60);
Insert Into systest.f_account_balance values (1,11,80);
Insert Into systest.f_account_balance values (1,12,10);
END TRANSACTION;
```

The sample table has the following data:

<table>
<thead>
<tr>
<th>account_id</th>
<th>month_id</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>99</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>94</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>90</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>88</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>90</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>92</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>80</td>
</tr>
</tbody>
</table>
For each account, let’s say that you want to analyze the sequence of consecutive monthly balance increases. When one month’s balance is less than, or equal to, the previous month’s balance, the requirement is to reset the counter to zero and restart.

*Teradata RESET WHEN use case*

To analyze this data, Teradata SQL uses a window function with a nested aggregate and a `RESET WHEN` phrase, as follows:

```sql
SELECT account_id, month_id, balance,
( ROW_NUMBER() OVER (PARTITION BY account_id ORDER BY month_id
RESET WHEN balance <= SUM(balance) over (PARTITION BY account_id ORDER BY month_id ROWS
BETWEEN 1 PRECEDING AND 1 PRECEDING) ) -1 ) as balance_increase
FROM systest.f_account_balance
ORDER BY 1,2;
```

**Output:**

<table>
<thead>
<tr>
<th>account_id</th>
<th>month_id</th>
<th>balance</th>
<th>balance_increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>94</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>88</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>90</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>92</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>80</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The query is processed as follows in Teradata:

1. The `SUM(balance)` aggregate function calculates the sum of all balances for a given account in a given month.
2. We check to see if a balance in a given month (for a given account) is greater than the balance of the previous month.
3. If the balance increased, we track a cumulative count value. If the `RESET WHEN` condition evaluates to `false`, which means that the balance has increased over successive months, we continue to increase the count.
4. The `ROW_NUMBER()` ordered analytical function calculates the count value. When we reach a month whose balance is less than, or equal to, the balance of the previous month, the `RESET WHEN` condition evaluates to `true`. If so, we start a new partition and `ROW_NUMBER()` restarts the count from 1. We use `ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING` to access the value of the previous row.

5. We subtract 1 to ensure that the count value starts with 0.

**Amazon Redshift equivalent SQL**

Amazon Redshift doesn't support the `RESET WHEN` phrase in an SQL analytical window function. To produce the same result, you must rewrite the Teradata SQL using Amazon Redshift native SQL syntax and nested sub-queries, as follows:

```sql
SELECT account_id, month_id, balance,
     (ROW_NUMBER() OVER(PARTITION BY account_id, new_dynamic_part ORDER BY month_id) -1) as balance_increase
FROM
 ( SELECT account_id, month_id, balance, prev_balance,
     SUM(dynamic_part) OVER (PARTITION BY account_id ORDER BY month_id ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) As new_dynamic_part
     FROM ( SELECT account_id, month_id, balance, SUM(balance) over (PARTITION BY account_id ORDER BY month_id ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING) as prev_balance,
     (CASE When balance <= prev_balance Then 1 Else 0 END) as dynamic_part
     FROM systest.f_account_balance ) A
 ) B
ORDER BY 1,2;
```

Because Amazon Redshift doesn't support nested window functions in the `SELECT` clause of a single SQL statement, you must use two nested sub-queries.

- In the inner sub-query (alias A), a dynamic partition indicator (`dynamic_part`) is created and populated. `dynamic_part` is set to 1 if one month's balance is less than or equal to the preceding month's balance; otherwise, it's set to 0.
- In the next layer (alias B), a `new_dynamic_part` attribute is generated as the result of a `SUM` window function.
- Finally, you add `new_dynamic_part` as a new partition attribute (dynamic partition) to the existing partition attribute (`account_id`) and apply the same `ROW_NUMBER()` window function as in Teradata (and minus one).

After these changes, Amazon Redshift SQL generates the same output as Teradata.

**Epics**

**Convert RESET WHEN to Amazon Redshift SQL**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create your Teradata window function.</td>
<td>Use nested aggregates and the RESET WHEN phrase according to your needs.</td>
<td>SQL developer</td>
</tr>
<tr>
<td>Convert the code to Amazon Redshift SQL.</td>
<td>To convert your code, follow the guidelines in the &quot;Tools&quot; section of this pattern.</td>
<td>SQL developer</td>
</tr>
</tbody>
</table>
Run the code in Amazon Redshift. Create your table, load data into the table, and run your code in Amazon Redshift. SQL developer

Related resources

References

• RESET WHEN Phrase (Teradata documentation)
• RESET WHEN explanation (Stack Overflow)
• Migrate to Amazon Redshift (AWS website)
• Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents (AWS Prescriptive Guidance)
• Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL (AWS Prescriptive Guidance)

Tools

• AWS Schema Conversion Tool (AWS SCT)

Partners

• AWS Migration Competency Partners

Ensure Amazon EMR logging to Amazon S3 is enabled at launch

Created by Priyanka Chaudhary (AWS)

Environment: Production

Technologies: Security, identity, compliance; Serverless; Analytics

Workload: Open-source

AWS services: Amazon EMR; Amazon S3; Amazon SNS; Amazon CloudWatch

Summary

This pattern provides a security control that monitors logging configuration for Amazon EMR clusters running on Amazon Web Services (AWS).

Amazon EMR is an AWS tool for big data processing and analysis. Amazon EMR offers the expandable low-configuration service as an alternative to running in-house cluster computing. Amazon EMR provides two types of EMR clusters.
Transient Amazon EMR clusters: Transient Amazon EMR clusters automatically shut down and stop incurring costs when processing is finished.

Persistent Amazon EMR clusters: Persistent Amazon EMR clusters continue to run after the data processing job is complete.

Amazon EMR and Hadoop both produce log files that report status on the cluster. By default, these are written to the master node in the `/mnt/var/log/` directory. Depending on how you configure the cluster when you launch it, you can also save these logs to Amazon Simple Storage Service (Amazon S3) and view them through the graphical debugging tool. Note that Amazon S3 logging can be specified only when the cluster is launched. With this configuration, logs are sent from the primary node to the Amazon S3 location every 5 minutes. For transient clusters, Amazon S3 logging is important because the clusters disappear when processing is complete, and these log files can be used to debug any failed jobs.

The pattern uses an AWS CloudFormation template to deploy a security control that monitors for API calls and starts Amazon CloudWatch Events on "RunJobFlow." The trigger invokes AWS Lambda, which runs a Python script. The Lambda function retrieves the EMR cluster ID from the event JSON input and also checks for an Amazon S3 log URI. If an Amazon S3 URI is not found, the Lambda function sends an Amazon Simple Notification Service (Amazon SNS) notification detailing the EMR cluster name, violation details, AWS Region, AWS account, and the Lambda Amazon Resource Name (ARN) that the notification is sourced from.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- An S3 bucket for the Lambda code .zip file
- An email address where you want to receive the violation notification

**Limitations**

- This detective control is regional and must be deployed in the AWS Regions you intend to monitor.

**Product versions**

- Amazon EMR release 4.8.0 and later

**Architecture**

**Target technology stack**

- Amazon CloudWatch Events event
- Amazon EMR
- Lambda function
- S3 bucket
- Amazon SNS

**Target architecture**
Automation and scale

- If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up AWS resources using infrastructure as code.
- **AWS CloudWatch Events** – AWS CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- **Amazon EMR** – Amazon EMR is a managed cluster platform that simplifies running big data frameworks.
- **AWS Lambda** – AWS Lambda supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon S3** – Amazon S3 is a web services interface that you can use to store and retrieve any amount of data from anywhere on the web.
- **Amazon SNS** – Amazon SNS is a web service that coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses.

Code

- A .zip file of the project is available as an attachment.

Epics

Define the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>To host the Lambda code .zip file, choose or create an S3 bucket with a unique name that does not contain leading slashes.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same AWS Region as the Amazon EMR cluster that is being evaluated.</td>
<td></td>
</tr>
</tbody>
</table>

#### Upload the Lambda code to the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Upload the Lambda code .zip file that's provided in the &quot;Attachments&quot; section to the S3 bucket. The S3 bucket must be in the same Region as the Amazon EMR cluster that is being evaluated.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

#### Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On the AWS CloudFormation console, in the same Region as your S3 bucket, deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters. For more information about deploying AWS CloudFormation templates, see the “Related resources” section.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

#### Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Enter the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the Amazon S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, <code>&lt;directory&gt;/&lt;filename&gt;.zip</code>).</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
**Task** | **Description** | **Skills required**
---|---|---
Provide an email address. | Provide an active email address to receive Amazon SNS notifications. | Cloud Architect
Define the logging level. | Define the logging level and frequency for your Lambda function. "Info" designates detailed informational messages on the application's progress. "Error" designates error events that could still allow the application to continue running. "Warning" designates potentially harmful situations. | Cloud Architect

**Related resources**

AWS Lambda
Amazon EMR logging
Deploying AWS CloudFormation templates

**Attachments**

attachment.zip

**Launch a Spark job in a transient EMR cluster using a Lambda function**

*Created by Adnan Alvee (AWS)*

**Environment:** Production  **Technologies:** Analytics  **Workload:** Open-source
Summary

This pattern uses the Amazon EMR RunJobFlow API action to launch a transient cluster to run a Spark job from a Lambda function. A transient EMR cluster is designed to terminate as soon as the job is complete or if any error occurs. A transient cluster provides cost savings because it runs only during the computation time, and it provides scalability and flexibility in a cloud environment.

The transient EMR cluster is launched using the Boto3 API and the Python programming language in a Lambda function. The Lambda function, which is written in Python, provides the added flexibility of initiating the cluster when it is needed.

To demonstrate a sample batch computation and output, this pattern will launch a Spark job in an EMR cluster from a Lambda function and run a batch computation against the example sales data of a fictional company. The output of the Spark job will be a comma-separated values (CSV) file in Amazon Simple Storage Service (Amazon S3). The input data file, Spark .jar file, a code snippet, and an AWS CloudFormation template for a virtual private cloud (VPC) and AWS Identity and Access Management (IAM) roles to run the computation are provided as an attachment.

Prerequisites and limitations

Prerequisites

• An AWS account

Limitations

• Only one Spark job can be initiated from the code at a time.

Product versions

• Tested on EMR 5.0.0+

Architecture

Target technology stack

• Amazon EMR
• AWS Lambda
• Amazon S3
• Apache Spark
Tools

AWS services

- Amazon EMR – Amazon EMR is the industry-leading cloud big data platform for processing vast amounts of data using open source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi, and Presto.
- AWS Lambda – AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume; there is no charge when your code isn't running. With Lambda, you can run code for virtually any type of application or backend service with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and manages all the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) provides storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

Code

```python
""
Copy paste the following code in your Lambda function. Make sure to change the following key parameters for the API as per your account

- Name (Name of Spark cluster)
- LogUri (S3 bucket to store EMR logs)
- Ec2SubnetId (The subnet to launch the cluster into)
- JobFlowRole (Service role for EC2)
- ServiceRole (Service role for Amazon EMR)

The following parameters are additional parameters for the Spark job itself. Change the bucket name and prefix for the Spark job (located at the bottom).

""
```
```python
import json
import boto3

client = boto3.client('emr')

def lambda_handler(event, context):
    response = client.run_job_flow(
        Name= 'spark_job_cluster',
        LogUri= 's3://your-bucket-name/prefix/logs',
        ReleaseLabel= 'emr-6.0.0',
        Instances={
            'MasterInstanceType': 'm5.xlarge',
            'SlaveInstanceType': 'm5.large',
            'InstanceCount': 1,
            'KeepJobFlowAliveWhenNoSteps': False,
            'TerminationProtected': False,
            'Ec2SubnetId': 'subnet-XXXXXXXXXXXXX'
        },
        Applications = [ {'Name': 'Spark'} ],
        Configurations = [
            { 'Classification': 'spark-hive-site',
                'Properties': { 'hive.metastore.client.factory.class': 'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory' }
            },
        ],
        VisibleToAllUsers=True,
        JobFlowRole = 'EMRLambda-EMREC2InstanceProfile-XXXXXXXXX',
        ServiceRole = 'EMRLambda-EMRRole-XXXXXXXXX',
        Steps=[
            { 'Name': 'flow-log-analysis',
                'ActionOnFailure': 'TERMINATE_CLUSTER',
                'HadoopJarStep': { 'Jar': 'command-runner.jar',
                    'Args': [ 'spark-submit',
                        '--deploy-mode', 'cluster',
                        '--executor-memory', '6G',
                        '--num-executors', '1',
                        '--executor-cores', '2',
                        '--class', 'com.aws.emr.ProfitCalc',
                        's3://your-bucket-name/prefix/lambda-emr/SparkProfitCalc.jar',
                        's3://your-bucket-name/prefix/fake_sales_data.csv',
                        's3://your-bucket-name/prefix/outputs/report_1/'
                        ]
            }
        ]
    )
```

### Epics

#### Create the EMR and Lambda IAM roles and the VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If you already have Lambda and Amazon EMR IAM roles and a VPC, you can skip this step. To run the code, both the EMR cluster and the Lambda function require IAM roles. The EMR cluster also requires a VPC with a public subnet or a private subnet with a NAT gateway. To automatically create all the IAM roles and a VPC, deploy the attached AWS CloudFormation template as is, or you can create the roles and the VPC manually as specified in the Additional information section.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td></td>
<td>After the CloudFormation template has successfully deployed, navigate to the Outputs tab in the AWS CloudFormation console. Note the four output keys: LambdaExecutionRole, ServiceRole, JobFlowRole, and Ec2SubnetId. You will use the values from these keys when you create the Lambda function.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

#### Create the S3 bucket and upload the Spark .jar file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create an S3 bucket or use an existing bucket.</td>
<td>General AWS</td>
</tr>
<tr>
<td></td>
<td>Upload the Spark .jar file to the S3 bucket.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

#### Create the Lambda function to launch the EMR cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On the Lambda console, create a Python 3.8+ Lambda function</td>
<td>Cloud/Big Data Developer</td>
</tr>
</tbody>
</table>
## Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy and paste the code.</td>
<td>Replace the code in the <code>lambda_function.py</code> file with the code from the &quot;Tools&quot; section of this pattern.</td>
<td>Cloud/Big Data Developer</td>
</tr>
<tr>
<td>Change the parameters in the code.</td>
<td>Follow the comments in the code to change the parameter values to match your AWS account.</td>
<td>Cloud/Big Data Developer</td>
</tr>
<tr>
<td>Launch the function to initiate the cluster.</td>
<td>Launch the function to initiate the creation of a transient EMR cluster with the Spark .jar file provided. It will run the Spark job and terminate automatically when the job is complete.</td>
<td>Cloud/Big Data Developer</td>
</tr>
<tr>
<td>Check the EMR cluster status.</td>
<td>After the EMR cluster is initiated, it appears in the EMR console under the <strong>Clusters</strong> tab. Any errors while launching the cluster or running the job can be checked accordingly.</td>
<td>Cloud/Big Data Developer</td>
</tr>
</tbody>
</table>

### Set up and run the sample demo

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Spark .jar file.</td>
<td>Download the Spark .jar file from the <strong>Attachments</strong> section and upload it to the S3 bucket.</td>
<td>Cloud/Data Engineer</td>
</tr>
<tr>
<td>Upload the input dataset.</td>
<td>Upload the attached <code>fake_sales_data.csv</code> file to the S3 bucket.</td>
<td>Cloud/Big Data Developer</td>
</tr>
<tr>
<td>Paste the Lambda code and change the parameters.</td>
<td>Copy the code from the <strong>Tools</strong> section, and paste the code in a Lambda function, replacing the code <code>lambda_function.py</code> file. Change the parameter values to match your account.</td>
<td>Cloud/Big Data Developer</td>
</tr>
<tr>
<td>Launch the function and verify the output.</td>
<td>After the Lambda function initiates the cluster with the provided Spark job, it generates a CSV file in the S3 bucket.</td>
<td>Cloud/Big Data Developer</td>
</tr>
</tbody>
</table>
Related resources

- Building Spark
- Apache Spark and Amazon EMR
- Boto3 Docs run_job_flow documentation
- Apache Spark information and documentation

Additional information

To launch the EMR cluster in a Lambda function, a VPC and IAM roles are needed. You can set up the VPC and IAM roles by using the AWS CloudFormation template in the Attachments section of this pattern, or you can manually create them by using the links below.

The following IAM roles are required to run Lambda and Amazon EMR.

**Lambda execution role**

A Lambda function’s execution role grants it permission to access AWS services and resources.

**Service role for Amazon EMR**

The Amazon EMR role defines the allowable actions for Amazon EMR when provisioning resources and performing service-level tasks that are not performed in the context of an Amazon Elastic Compute Cloud (Amazon EC2) instance running within a cluster. For example, the service role is used to provision EC2 instances when a cluster launches.

**Service role for EC2 instances**

The service role for cluster EC2 instances (also called the EC2 instance profile for Amazon EMR) is a special type of service role that is assigned to every EC2 instance in an Amazon EMR cluster when the instance launches. Application processes that run on top of the Hadoop ecosystem assume this role for permissions to interact with other AWS services.

**VPC and subnet creation**

You can use the VPC wizard to create a new VPC from the VPC console.

Attachments

attachment.zip

Load data from Amazon S3 to Amazon Redshift using AWS Glue

*Created by Vijay Sahoo (AWS)*

<table>
<thead>
<tr>
<th>Created by:</th>
<th>AWS</th>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Storage &amp; backup; Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon Redshift; AWS Glue; Amazon S3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

This pattern describes the data migration process from an Amazon Simple Storage Service (Amazon S3) bucket to an Amazon Redshift cluster by using AWS Glue.

Prerequisites and limitations

Prerequisites

• An active AWS account
• An S3 source bucket with comma-separated values (CSV) files and the correct privileges
• An Amazon Redshift cluster on AWS

Architecture

Source technology stack

• An S3 bucket with CSV files

Target technology stack

• An Amazon Redshift cluster

Data migration architecture
## Tools

- **AWS Glue** - AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load data for analytics. It's designed to simplify the tasks of moving and transforming datasets for analysis, and provides the necessary scheduling, alerting, and triggering features as part of a wider data processing workflow.

## Epics

### Plan for the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create the outbound security group for the source and target databases.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Prepare the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the database user for the migration in the Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the AWS Identity and Access Management (IAM) service role for the Amazon Redshift cluster and give it read access to the S3 bucket data source.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Attach the role from the previous step to the target database.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
<tr>
<td>Review the target database options, parameters, and Amazon Redshift workload management (WLM) settings.</td>
<td></td>
<td>SysAdmin, DBA</td>
</tr>
</tbody>
</table>
Create a new job in AWS Glue

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a new database in AWS Glue.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Add a new table in the newly created AWS Glue database.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>In parameters, provide an Amazon S3 data source location and table column details.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create a new job in AWS Glue.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>In parameters, choose the IAM service role, which has permission to the Amazon S3 data source.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>For the data source, choose the Amazon S3 data source location.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>For the target, choose &quot;Create tables in your data target&quot; option. For the data store, choose Java Database Connectivity (JDBC).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Select the connection endpoint of the Amazon Redshift cluster and provide an Amazon Redshift database name.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Save and run the job in AWS Glue.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

Cut Over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete the job in AWS Glue after data loading or when the use case is complete.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

- Amazon S3 documentation
- AWS Glue documentation
- Amazon Redshift documentation
- AWS Glue pricing
- Amazon Redshift pricing
Migrate an ELK Stack to Elastic Cloud on AWS

Created by Battulga Purevragchaa (AWS), Uday Theepireddy (Elastic), and Antony Prasad Thevaraj (AWS)

<table>
<thead>
<tr>
<th>Environment</th>
<th>Production</th>
<th>Source</th>
<th>Elasticsearch</th>
<th>Target</th>
<th>Elastic Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type</td>
<td>Replatform</td>
<td>Workload</td>
<td>All other workloads</td>
<td>Technologies</td>
<td>Analytics; Security, identity, compliance</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon EC2; Amazon EC2 Auto Scaling; Elastic Load Balancing; Amazon S3; Amazon Route 53

**Summary**

Elastic has provided services for many years, with their users and customers typically managing Elastic themselves on premises. Elastic Cloud, the managed Elasticsearch service, provides a way to consume the Elastic Stack (ELK Stack) and solutions for enterprise search, observability, and security. You can access Elastic solutions with apps such as Logs, Metrics, APM (application performance monitoring), and SIEM (security information and event management). You can use integrated features such as machine learning, index lifecycle management, Kibana Lens (for drag-and-drop visualizations).

When you move from self-managed Elasticsearch to Elastic Cloud, the Elasticsearch service takes care of the following:

- Provisioning and managing the underlying infrastructure
- Creating and managing Elasticsearch clusters
- Scaling clusters up and down
- Upgrades, patching, and taking snapshots

This gives you more time to focus on solving other challenges.

This pattern defines how to migrate on-premises Elasticsearch 7.13 to Elasticsearch on Elastic Cloud on Amazon Web Services (AWS). Other versions might require slight modifications to the processes described in this pattern. For more information, contact your Elastic representative.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account with access to Amazon Simple Storage Service (Amazon S3) for snapshots
- A secure, sufficiently high-bandwidth private link for copying snapshot data files to Amazon S3
- Amazon S3 Transfer Acceleration
- Elastic Snapshot policies to ensure that data ingestion is archived regularly, either to a sufficiently large local data store or to remote storage (Amazon S3)

You must understand how large your snapshots and the lifecycle policies for accompanying indexes are on premises before initiating your migration. For more information, contact Elastic.
Roles and skills

The migration process also requires the roles and expertise described in the following table.

<table>
<thead>
<tr>
<th>Role</th>
<th>Expertise</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>App support</td>
<td>Familiarity with Elastic Cloud and Elastic on premises</td>
<td>All Elastic related tasks</td>
</tr>
<tr>
<td>Systems administrator or DBA</td>
<td>In-depth knowledge of the on-premises Elastic environment and its configuration</td>
<td>The ability to provision storage, install and use the AWS Command Line Interface (AWS CLI), and identify all data sources feeding Elastic on premises</td>
</tr>
<tr>
<td>Network administrator</td>
<td>Knowledge of on-premises to AWS network connectivity, security, and performance</td>
<td>Establishment of network links from on premises to Amazon S3, with an understanding of connectivity bandwidth</td>
</tr>
</tbody>
</table>

Limitations

- Elasticsearch on Elastic Cloud is available only in supported AWS Regions (September 2021).

Product versions

- Elasticsearch 7.13

Architecture

Source technology stack

On-premises Elasticsearch 7.13 or later:

- Cluster snapshots
- Index snapshots
- Beats configuration

Source technology architecture

The following diagram shows a typical on-premises architecture with different ingestion methods, node types, and Kibana. The different node types reflect the Elasticsearch cluster, authentication, and visualization roles.
1. Ingestion from Beats to Logstash
2. Ingestion from Beats to Apache Kafka messaging queue
3. Ingestion from Filebeat to Logstash
4. Ingestion from Apache Kafka messaging queue to Logstash
5. Ingestion from Logstash to an Elasticsearch cluster
6. Elasticsearch cluster
7. Authentication and notification node
8. Kibana and blob nodes

**Target technology stack**

Elastic Cloud is deployed to your software as a service (SaaS) account in multiple AWS Regions with cross-cluster replication.

- Cluster snapshots
- Index snapshots
- Beats configurations
- Elastic Cloud
- Network Load Balancer
- Amazon Route 53
- Amazon S3

**Target architecture**
The managed Elastic Cloud infrastructure is:

- Highly available, being present in multiple Availability Zones and multiple AWS Regions.
- Region failure tolerant because data (indexes and snapshots) is replicated using Elastic Cloud cross-cluster replication (CCR)
- Archival, because snapshots are archived in Amazon S3
- Network partition tolerant through a combination of Network Load Balancers and Route 53
- Data ingestion originating from (but not limited to) Elastic APM, Beats, Logstash

High-level migration steps

Elastic has developed its own prescriptive methodology for migrating on-premises Elastic Cluster to Elastic Cloud. The Elastic methodology is directly aligned and complementary to the AWS migration guidance and best practices, including Well-Architected Framework and AWS Migration Acceleration Program (MAP). Typically, the three AWS migration phases are the following:

- Assess
- Mobilize
- Migrate and modernize

Elastic follows similar migration phases with complementary terminology:
Elastic uses the Elastic Implementation Methodology to facilitate the delivery of project outcomes. This is inclusive by design to ensure that the Elastic, consulting teams, and customer teams work together with clarity to jointly deliver intended outcomes.

The Elastic methodology combines traditional waterfall phasing with Scrum within the implementation phase. Configurations of technical requirements are delivered iteratively in a collaborative manner while minimizing risk.

**Tools**

- **Elastic Load Balancing** – Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.
- **Amazon Route 53** – Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses an S3 bucket and Amazon S3 Transfer Acceleration.
- **Beats** – Beats ship data from Logstash or Elasticsearch
- **Elastic Cloud** – Elastic Cloud is a managed service for hosting Elasticsearch.
- **Elasticsearch** – Elasticsearch is a search and analytics engine that uses the Elastic Stack to centrally store your data for search and analytics that scale. This pattern also uses snapshot creation and cross-cluster replication.
- **Logstash** – Logstash is a server-side data processing pipeline that ingests data from multiple sources, transforms it, and then sends it to your data storage.

**Epics**

**Prepare the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify servers running the on-premises Elastic solution.</td>
<td>Confirm that Elastic migration is supported.</td>
<td>App owner</td>
</tr>
<tr>
<td>Understand the on-premises server configuration.</td>
<td>To understand the server configuration needed to drive</td>
<td>App Support</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Gather user and app account information.</td>
<td>Identify the user names and app names that are used by the on-premises Elastic environment.</td>
<td>Systems administrator, App support</td>
</tr>
<tr>
<td>Document Beats and data shipper configuration.</td>
<td>To document the configurations, look at existing data sources and sinks. For more information, see the Elastic documentation.</td>
<td>App support</td>
</tr>
<tr>
<td>Determine the velocity and volume of data.</td>
<td>Establish a baseline for how much data the cluster is handling.</td>
<td>Systems administrator, App support</td>
</tr>
<tr>
<td>Document RPO and RTO scenarios.</td>
<td>Document recovery point objective (RPO) and recovery time objective (RTO) scenarios in terms of outages and service level agreements (SLAs).</td>
<td>App owner, Systems administrator, App support</td>
</tr>
<tr>
<td>Determine the optimal snapshot lifecycle settings.</td>
<td>Define how often data needs to be secured by using Elastic snapshots during and after the migration.</td>
<td>App owner, Systems administrator, App support</td>
</tr>
<tr>
<td>Define post-migration performance expectations.</td>
<td>Generate metrics on current and expected screen refresh, query runtimes, and user interface behaviors.</td>
<td>Systems administrator, App support</td>
</tr>
<tr>
<td>Document internet access transport, bandwidth, and availability requirements.</td>
<td>Ascertain speed, latency, and resiliency of internet connections for copying snapshots to Amazon S3.</td>
<td>Network administrator</td>
</tr>
<tr>
<td>Document current costs of on-premises runtime for Elastic.</td>
<td>Ensure that the sizing of the AWS targeted environment is designed to be both high performing and cost effective.</td>
<td>DBA, Systems administrator, App support</td>
</tr>
<tr>
<td>Identify the authentication and authorization needs.</td>
<td>The Elastic Stack security features provide built-in realms such as Lightweight Directory Access Protocol (LDAP), Security Assertion Markup Language (SAML), and OpenID Connect (OIDC).</td>
<td>DBA, Systems administrator, App support</td>
</tr>
</tbody>
</table>
### Task 1: Understand the specific regulatory requirements based on the geographic location.

**Description:** Ensure that data is exported and encrypted according to your requirements and to any relevant national requirements.

**Skills required:** DBA, Systems administrator, App support

### Implement the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the staging area on Amazon S3.</td>
<td>To receive snapshots on Amazon S3, set up the Amazon S3 area by following the steps in the Elasticsearch documentation. Use the es-s3-Snapshot customer managed policy attached to a new es-Snapshot-user AWS Identity and Access Management (IAM) account. Keep the access key ID and secret access key secured. Enable Amazon S3 Transfer Acceleration on the bucket.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Install AWS CLI and the Amazon S3 plugin on premises.</td>
<td>On each Elasticsearch node, run the following command. Then reboot the node.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Configure Amazon S3 client access.</td>
<td>Use the keys from the IAM user es-Snapshot-user by running the following commands.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Register a snapshot repository for Elastic data</td>
<td>Use the Kibana Dev Tools to tell the on-premises local cluster which remote S3 bucket to write to.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Configure snapshot policy.</td>
<td>To configure snapshot lifecycle management, on the Kibana Policies tab, choose SLM policy, and define which times, data streams, or indexes should be</td>
<td>App support</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Configure a policy that</td>
<td>Configure a policy that takes frequent snapshots. Snapshots are incremental and make efficient use of storage. Match your readiness assessment decision. A policy can also specify a retention policy and automatically delete snapshots when they are no longer needed.</td>
<td>AWS administrator, App support,</td>
</tr>
<tr>
<td>included, and what names to use.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that snapshots work.</td>
<td>In Kibana Dev Tools, run the following command.</td>
<td>AWS administrator, App support,</td>
</tr>
<tr>
<td></td>
<td>GET _snapshot/ &lt;your_repo_name&gt;/_all</td>
<td></td>
</tr>
<tr>
<td>Deploy a new cluster on</td>
<td>Log in to Elastic, and choose a cluster for “observability, search or security” derived from your business findings in the readiness assessment.</td>
<td>AWS administrator, App support</td>
</tr>
<tr>
<td>Elastic Cloud.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set up cluster key store</td>
<td>The new cluster needs access to the S3 bucket that will store the snapshots. On the Elasticsearch Service Console, choose Security, and enter the access and secret IAM keys that you created earlier.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>access.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure the Elastic Cloud</td>
<td>Set up new cluster access to the previously created snapshot repository in Amazon S3. Using Kibana, do the following:</td>
<td>AWS administrator, App Support</td>
</tr>
<tr>
<td>hosted cluster to access</td>
<td>1. Choose Stack Management, Snapshot Settings, RegisterRepo.</td>
<td></td>
</tr>
<tr>
<td>Amazon S3.</td>
<td>2. In the Alias field, enter the name of the repository.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. For S3 Client name, choose secondary.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Add the S3 bucket that you created earlier to the repository.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Choose Compress snapshot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. For the Encryption settings, keep the default values.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Verify the new Amazon S3 repository.</td>
<td>Ensure that you can access your new repository hosted in the Elastic Cloud cluster.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Initialize the Elasticsearch service cluster.</td>
<td>On the Elasticsearch Service Console, initialize the Elasticsearch service cluster from the S3 snapshot. Run the following commands as POST.</td>
<td>App Support</td>
</tr>
<tr>
<td></td>
<td>*/_close? expand_wildcards=all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/_snapshot/&lt;your-repo-name&gt;/ &lt;your-snapshot-name&gt;/ _restore</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*/_open?expand_wildcards=all</td>
<td></td>
</tr>
</tbody>
</table>

**Complete the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify that the snapshot restore was successful.</td>
<td>Using Kibana Dev Tools, run the following command.</td>
<td>App support</td>
</tr>
<tr>
<td></td>
<td>GET _cat/indices</td>
<td></td>
</tr>
<tr>
<td>Redploy ingestion services.</td>
<td>Connect the endpoints for Beats and Logstash to the new Elasticsearch service endpoint.</td>
<td>App support</td>
</tr>
</tbody>
</table>

**Test the cluster environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the cluster environment.</td>
<td>After the on-premises Elastic cluster environment is migrated to AWS, you can connect to it and use your own user acceptance testing (UAT) tools to validate the new environment.</td>
<td>App support</td>
</tr>
</tbody>
</table>

**Related resources**

Elastic references
Related resources

• Elastic Cloud
• Managed Elasticsearch and Kibana on AWS
• Elastic enterprise search
• Elastic integrations
• Elastic observability
• Elastic security
• Beats
• Elastic APM
• Migrate to index lifecycle management
• Elastic subscriptions
• Contact Elastic

Elastic blog posts
• How to migrate from self-managed Elasticsearch to Elastic Cloud on AWS (blog post)
• Migrating to Elastic Cloud (blog post)

Elastic documentation
• Tutorial: Automate backups with SLM
• ILM: Manage the index lifecycle
• Logstash
• Cross-cluster replication (CCR)
• Ingest pipelines
• Run Elasticsearch API requests
• Snapshot retention

Elastic video and webinar
• Elastic cloud migration
• Elastic Cloud: Why are customers migrating (webinar)

AWS references
• Elastic Cloud on AWS Marketplace
• AWS Command Line Interface
• AWS Direct Connect
• AWS Migration Acceleration Program
• Network Load Balancers
• Regions and Availability Zones
• Amazon Route 53
• Amazon Simple Storage Service
• Amazon S3 Transfer Acceleration
• VPN connections
• Well-Architected Framework
Additional information

If you're planning to migrate complex workloads, engage Elastic Consulting Services. If you have basic questions related to configurations and services, contact the Elastic Support team.

Migrate data to the AWS Cloud by using Starburst

*Created by Antony Prasad Thevaraj (AWS), Shaun Van Staden (Starburst), and Suresh Veeragoni (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Analytics; Data lakes; Databases</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon EKS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

Starburst helps accelerate your data migration journey to Amazon Web Services (AWS) by providing an enterprise query engine that brings existing data sources together in a single access point. You can run analytics across multiple data sources to get valuable insights, before finalizing any migration plans. Without disrupting business-as-usual analytics, you can migrate the data by using the Starburst engine or a dedicated extract, transform, and load (ETL) application.

Prerequisites and limitations

**Prerequisites**

- An active AWS account
- A virtual private cloud (VPC)
- An Amazon Elastic Kubernetes Service (Amazon EKS) cluster
- An Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling group
- A list of current system workloads that need to be migrated
- Network connectivity from AWS to your on-premises environment

Architecture

**Reference architecture**

The following high-level architecture diagram shows the typical deployment of Starburst Enterprise in the AWS Cloud:

1. The Starburst Enterprise cluster runs inside your AWS account.
2. A user authenticates by using Lightweight Directory Access Protocol (LDAP) or Open Authorization (OAuth) and interacts directly with the Starburst cluster.
3. Starburst can connect to several AWS data sources, such as AWS Glue, Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), and Amazon Redshift. Starburst provides federated query capabilities across data sources in the AWS Cloud, on premises, or in other cloud environments.
4. You launch Starburst Enterprise in an Amazon EKS cluster by using Helm charts.
5. Starburst Enterprise uses Amazon EC2 Auto Scaling groups and Amazon EC2 Spot Instances to optimize infrastructure.

6. Starburst Enterprise connects directly to your existing on-premises data sources to read data real-time. In addition, if you have an existing Starburst Enterprise deployment in this environment, you can directly connect your new Starburst cluster in the AWS Cloud to this existing cluster.

Please note the following:

- Starburst is not a data virtualization platform. It is a SQL-based massively parallel processing (MPP) query engine that forms the basis of an overall data mesh strategy for analytics.
- When Starburst is deployed as part of a migration, it has direct connectivity to the existing on-premises infrastructure.
• Starburst provides several built-in enterprise and open-source connectors that facilitate connectivity to a variety of legacy systems. For a full list of connectors and their capabilities, see Connectors in the Starburst Enterprise user guide.

• Starburst can query data in real-time from on-premises data sources. This prevents interruptions of regular business operations while data is being migrated.

• If you are migrating from an existing on-premises Starburst Enterprise deployment, you can use a special connector, Starburst Stargate, to connect your Starburst Enterprise cluster in AWS directly to your on-premises cluster. This provides additional performance benefits when business users and data analysts are federating queries from the AWS Cloud to your on-premises environment.

**High-level process overview**

You can accelerate data migration projects by using Starburst because Starburst enables insights across all of your data, prior to migrating it. The following image shows the typical process for migrating data by using Starburst.

**Roles**

The following roles are typically required to complete a migration using Starburst:

• **Cloud administrator** – Responsible for making cloud resources available to run the Starburst Enterprise application

• **Starburst administrator** – Responsible for installing, configuring, managing, and supporting the Starburst application

• **Data engineer** – Responsible for:
  • Migrating the legacy data to the cloud
  • Building semantic views to support analytics
• **Solution or system owner** – Responsible for the overall solution implementation

## Tools

### AWS services

- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.

- **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service for running Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

### Other tools

- **Helm** – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.

- **Starburst Enterprise** – Starburst Enterprise is a SQL-based massively parallel processing (MPP) query engine that forms the basis of an overall data mesh strategy for analytics.

- **Starburst Stargate** – Starburst Stargate links catalogs and data sources in one Starburst Enterprise environment, such as a cluster in an on-premises data center, to the catalogs and data sources in another Starburst Enterprise environment, such as a cluster in the AWS Cloud.

## Epics

### Assess the data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify and prioritize your data.</td>
<td>Identify the data you want to move. Large, on-premises legacy systems can include core data that you want to migrate alongside data that you don’t want to move or can’t be moved because of compliance reasons. Starting with a data inventory helps you prioritize which data you should target first. For more information, see Get started with automated portfolio discovery.</td>
<td>Data engineer, DBA</td>
</tr>
<tr>
<td>Explore, inventory, and back up your data.</td>
<td>Validate the quality, quantity, and relevance of the data for your use case. Back up or create a snapshot of the data as needed, and finalize the target environment for the data.</td>
<td>Data engineer, DBA</td>
</tr>
</tbody>
</table>
### Set up the Starburst Enterprise environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Starburst Enterprise in the AWS Cloud.</td>
<td>While data is being catalogued, set up Starburst Enterprise in a managed Amazon EKS cluster. For more information see, Deploying with Kubernetes in the Starburst Enterprise reference documentation. This allows business-as-usual analytics while data migration is in process.</td>
<td>AWS administrator, App developer</td>
</tr>
<tr>
<td>Connect Starburst to the data sources.</td>
<td>After you have identified the data and set up Starburst Enterprise, connect Starburst to the data sources. Starburst reads data directly from the data source as a SQL query. For more information, see the Starburst Enterprise reference documentation.</td>
<td>AWS administrator, App developer</td>
</tr>
</tbody>
</table>

### Migrate the data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build and run the ETL pipelines.</td>
<td>Begin the data migration process. This activity can occur at the same time as business-as-usual analytics. For the migration, you can use a third-party product or Starburst. Starburst has the capability to both read and write data across different sources. For more information, see the Starburst Enterprise reference documentation.</td>
<td>Data engineer</td>
</tr>
<tr>
<td>Validate the data.</td>
<td>After the data has been migrated, validate the data to ensure all required data has been moved and is intact.</td>
<td>Data engineer, DevOps engineer</td>
</tr>
</tbody>
</table>

### Cut over and roll out

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut over the data.</td>
<td>After data migration and validation is complete, you can</td>
<td>Data engineer, Cutover lead</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut over the data. This involves changing the data connection links in Starburst. Instead of pointing at the on-premises sources, you point to the new cloud sources and update the semantic views. For more information, see Connectors in the Starburst Enterprise reference documentation.</td>
<td>Cutover lead, Data engineer</td>
</tr>
<tr>
<td>Roll out to users. Data consumers begin working off the migrated data sources. This process is invisible to the analytics end users.</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

**AWS Marketplace**
- Starburst Galaxy
- Starburst Enterprise
- Starburst Data JumpStart
- Starburst Enterprise with Graviton

**Starburst documentation**
- Starburst Enterprise user guide
- Starburst Enterprise reference documentation

**Other AWS documentation**
- Get started with automated portfolio discovery (AWS Prescriptive Guidance)
- Optimizing Cloud Infrastructure Cost and Performance with Starburst on AWS (blog post)

---

**Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions**

*Created by Sandip Gangapadhyay (AWS)*

**Environment:** Production  
**Technologies:** Analytics; Big data; Data lakes; DevOps; Serverless  
**AWS services:** Amazon Athena; AWS Glue; AWS Lambda; AWS Step Functions
Summary

This pattern describes how to build a serverless extract, transform, and load (ETL) pipeline to validate, transform, compress, and partition a large CSV dataset for performance and cost optimization. The pipeline is orchestrated by AWS Step Functions and includes error handling, automated retry, and user notification features.

When a CSV file is uploaded to an Amazon Simple Storage Service (Amazon S3) bucket source folder, the ETL pipeline starts to run. The pipeline validates the content and the schema of the source CSV file, transforms the CSV file to a compressed Apache Parquet format, partitions the dataset by year, month, and day, and stores it in a separate folder for analytics tools to process.

The code that automates this pattern is available on GitHub, in the ETL Pipeline with AWS Step Functions repository.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) installed and configured with your AWS account, so that you can create AWS resources by deploying an AWS CloudFormation stack. AWS CLI version 2 is recommended. For installation instructions, see Installing, updating, and uninstalling the AWS CLI version 2 in the AWS CLI documentation. For AWS CLI configuration instructions, see Configuration and credential file settings in the AWS CLI documentation.
- An Amazon S3 bucket.
- A CSV dataset with the correct schema. (The code repository included with this pattern provides a sample CSV file with the correct schema and data type that you can use.)
- A web browser that is supported for use with the AWS Management Console. (See the list of supported browsers.)
- AWS Glue console access.
- AWS Step Functions console access.

Limitations

- In AWS Step Functions, the maximum limit for keeping history logs is 90 days. For more information, see Quotas and Quotas for standard workflows in the AWS Step Functions documentation.

Product versions

- Python 3 for AWS Lambda
- AWS Glue version 2.0
The workflow illustrated in the diagram consists of these high-level steps:

1. The user uploads a CSV file into the source folder in Amazon S3.
2. An Amazon S3 notification event initiates an AWS Lambda function that starts the Step Functions state machine.
3. The Lambda function validates the schema and data type of the raw CSV file.
4. Depending on the validation results:
   a. If validation of the source file succeeds, the file moves to the stage folder for further processing.
   b. If validation fails, the file moves to the error folder, and an error notification is sent through Amazon Simple Notification Service (Amazon SNS).
5. An AWS Glue crawler creates the schema of the raw file from the stage folder in Amazon S3.
6. An AWS Glue job transforms, compresses, and partitions the raw file into Parquet format.
7. The AWS Glue job also moves the file to the transform folder in Amazon S3.
8. The AWS Glue crawler creates the schema from the transformed file. The resulting schema can be used by any analytics job. You can also use Amazon Athena to run ad-hoc queries.
9. If the pipeline completes without errors, the schema file is moved to the archive folder. If any errors are encountered, the file is moved to the error folder instead.
10. Amazon SNS sends a notification that indicates success or failure based on the pipeline completion status.

All the AWS resources used in this pattern are serverless. There are no servers to manage.

**Tools**

**AWS services**

- **AWS Glue** – AWS Glue is a fully managed ETL service that makes it easy for customers to prepare and load their data for analytics.
- **AWS Step Functions** – AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through the AWS Step Functions graphical console, you see your application’s workflow as a series of event-driven steps.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.
• **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications.

• **AWS Lambda** – AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

**Code**

The code for this pattern is available on GitHub, in the [ETL Pipeline with AWS Step Functions](https://github.com/aws-samples/etl-pipeline-with-aws-step-functions) repository. The code repository contains the following files and folders:

- **template.yml** – AWS CloudFormation template for creating the ETL pipeline with AWS Step Functions.
- **parameter.json** – Contains all parameters and parameter values. You update this file to change parameter values, as described in the **Epics** section.
- **myLayer/python** folder – Contains Python packages needed to create the required AWS Lambda layer for this project.
- **lambda** folder – Contains the following Lambda functions:
  - **move_file.py** – Moves the source dataset to the archive, transform, or error folder.
  - **check_crawler.py** – Checks the status of the AWS Glue crawler as many times as configured by the `RETRYLIMIT` environment variable before it sends a failure message.
  - **start_crawler.py** – Starts the AWS Glue crawler.
  - **start_step_function.py** – Starts AWS Step Functions.
  - **start_codebuild.py** – Starts the AWS CodeBuild project.
  - **validation.py** – Validates the input raw dataset.
  - **s3object.py** – Creates the required directory structure inside the S3 bucket.
  - **notification.py** – Sends success or error notifications at the end of the pipeline.

To use the sample code, follow the instructions in the **Epics** section.

**Epics**

**Prepare the source files**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Clone the sample code repository. | 1. Open the [ETL Pipeline with AWS Step Functions](https://github.com/aws-samples/etl-pipeline-with-aws-step-functions) repository.  
2. Choose Code on the main repository page, above the file list, and copy the URL listed under Clone with HTTPS.  
3. Change your working directory to the location where you want to store the sample files.  
4. At a terminal or command prompt, type the command: | Developer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>git clone &lt;repoURL&gt;</strong></td>
<td>where <code>&lt;repoURL&gt;</code> refers to the URL you copied in step 2.</td>
<td></td>
</tr>
</tbody>
</table>

**Update parameter values.**

In your local copy of the repository, edit the `parameter.json` file and update the default parameter values as follows:

- **pS3BucketName** — The name of the S3 bucket for storing the datasets. The template will create this bucket for you. The bucket name must be globally unique.
- **pSourceFolder** — The name of the folder inside the S3 bucket that will be used to upload the source CSV file.
- **pStageFolder** — The name of the folder inside the S3 bucket that will be used as the staging area during the process.
- **pTransformFolder** — The name of the folder inside the S3 bucket that will be used to store transformed and partitioned datasets.
- **pErrorFolder** — The folder inside the S3 bucket that the source CSV file will be moved to if it can't be validated.
- **pArchiveFolder** — The name of the folder inside the S3 bucket that will be used to archive the source CSV file.
- **pEmailforNotification** — A valid email address for receiving success/error notifications.
- **pDatasetSchema** — The dataset schema that the source file will be validated against. The Cerberus Python package is used for source dataset validation. For more information, see the Cerberus website.

**Developer**
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Upload the source code to the S3 bucket. | Before you deploy the CloudFormation template that automates the ETL pipeline, you must package the source files for the CloudFormation template and upload them to an S3 bucket. To do this, run the following AWS CLI command with your preconfigured profile:  
   ```
   aws cloudformation package --template-file template.yml --s3-bucket <bucket_name> --output-template-file packaged.template --profile <profile_name>
   ```  | Developer |

where:
- `<bucket_name>` is the name of an existing S3 bucket in the AWS Region where you want to deploy the stack. This bucket is used to store the source code package for the CloudFormation template.
- `<profile_name>` is a valid AWS CLI profile that you preconfigured when you set up AWS CLI.

### Create the stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the CloudFormation template. | To deploy the CloudFormation template, run the following AWS CLI command:  
   ```
   aws cloudformation deploy --stack-name <stack_name> --template-file packaged.template --parameter-overrides file://parameter.json --capabilities CAPABILITY_IAM --profile <profile_name>
   ```  | Developer |

where:
- `<stack_name>` is a unique identifier for the CloudFormation stack.
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Check progress.</strong></td>
<td>On the <a href="https://console.aws.amazon.com/cloudformation/home">AWS CloudFormation console</a>, check the progress of stack development. When the status is <code>CREATE_COMPLETE</code>, the stack has been deployed successfully.</td>
<td>Developer</td>
</tr>
<tr>
<td><strong>Note the AWS Glue database name.</strong></td>
<td>The <strong>Outputs</strong> tab for the stack displays the name of the AWS Glue database. The key name is <code>GlueDBOutput</code>.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

### Test the pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Start the ETL pipeline.** | 1. Navigate to the source folder (`source`, or the folder name you set in the `parameter.json` file) inside the S3 bucket.  
2. Upload a sample CSV file to this folder. (The code repository provides a sample file called `Sample_Bank_Transaction_Raw_Dataset.csv` that you can use.) Uploading the file will start the ETL pipeline through Step Functions.  
3. On the [Step Functions console](https://console.aws.amazon.com/stepfunctions/home), check the ETL pipeline status. | Developer |
| **Check for the partitioned dataset.** | When the ETL pipeline completes, verify that the partitioned dataset is available in the Amazon S3 transform folder (`transform`, or the folder name you set in the `parameter.json` file). | Developer |
| **Check for the partitioned AWS Glue database.** | 1. On the [AWS Glue console](https://console.aws.amazon.com/glue/home), select the AWS Glue database created by the stack (this is the database that you noted in the previous epic).  
2. Verify that the partitioned table is available in the AWS Glue Data Catalog. | Developer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run queries.</td>
<td>(Optional) Use Amazon Athena to run ad-hoc queries on the partitioned and transformed database. For instructions, see Running SQL Queries Using Amazon Athena in the AWS documentation.</td>
<td>Database analyst</td>
</tr>
</tbody>
</table>

**Related resources**

**AWS service documentation**

- AWS Step Functions
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon SNS

**Additional information**

The following diagram shows the AWS Step Functions workflow for a successful ETL pipeline, from the Step Functions Inspector panel.
The following diagram shows the AWS Step Functions workflow for an ETL pipeline that fails because of an input validation error, from the Step Functions Inspector panel.
Perform advanced analytics using Amazon Redshift ML
Summary

On the Amazon Web Services (AWS) Cloud, you can use Amazon Redshift machine learning (Amazon Redshift ML) to perform ML analytics on data stored in either an Amazon Redshift cluster or on Amazon Simple Storage Service (Amazon S3). Amazon Redshift ML supports supervised learning, which is typically used for advanced analytics. Use cases for Amazon Redshift ML include revenue forecasting, credit card fraud detection, and customer lifetime value (CLV) or customer churn predictions.

Amazon Redshift ML makes it easy for database users to create, train, and deploy ML models by using standard SQL commands. Amazon Redshift ML uses Amazon SageMaker Autopilot to automatically train and tune the best ML models for classification or regression based on your data, while you retain control and visibility.

All interactions between Amazon Redshift, Amazon S3, and Amazon SageMaker are abstracted away and automated. After the ML model is trained and deployed, it becomes available as a user-defined function (UDF) in Amazon Redshift and can be used in SQL queries.

This pattern complements the Create, train, and deploy ML models in Amazon Redshift using SQL with Amazon Redshift ML from the AWS Blog, and the Build, train, and deploy an ML model with Amazon SageMaker tutorial from the Getting Started Resource Center.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Existing data in an Amazon Redshift table

Skills

- Familiarity with terms and concepts used by Amazon Redshift ML, including machine learning, training, and prediction. For more information about this, see Training ML models in the Amazon Machine Learning (Amazon ML) documentation.
- Experience with Amazon Redshift user setup, access management, and standard SQL syntax. For more information about this, see Getting started with Amazon Redshift in the Amazon Redshift documentation.
- Knowledge and experience with Amazon S3 and AWS Identity and Access Management (IAM).
- Experience running commands in AWS Command Line Interface (AWS CLI) is also beneficial but not required.

Limitations

- The Amazon Redshift cluster and S3 bucket must be located in the same AWS Region.
- This pattern's approach only supports supervised learning models such as regression, binary classification, and multiclass classification.
The following steps explain how Amazon Redshift ML works with SageMaker to build, train, and deploy an ML model:

1. Amazon Redshift exports training data to an S3 bucket.
2. SageMaker Autopilot automatically preprocesses the training data.
3. After the `CREATE MODEL` statement is invoked, Amazon Redshift ML uses SageMaker for training.
4. SageMaker Autopilot searches for and recommends the ML algorithm and optimal hyper-parameters that optimize the evaluation metrics.
5. Amazon Redshift ML registers the output ML model as a SQL function in the Amazon Redshift cluster.
6. The ML model's function can be used in a SQL statement.

**Technology stack**

- Amazon Redshift
- SageMaker
- Amazon S3

**Tools**

- **Amazon Redshift** – Amazon Redshift is an enterprise-level, petabyte scale, fully managed data warehousing service.
- **Amazon Redshift ML** – Amazon Redshift machine learning (Amazon Redshift ML) is a robust, cloud-based service that makes it easy for analysts and data scientists of all skill levels to use ML technology.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- **Amazon SageMaker** – SageMaker is a fully managed ML service.
- **Amazon SageMaker Autopilot** – SageMaker Autopilot is a feature-set that automates key tasks of an automatic machine learning (AutoML) process.
Code

You can create a supervised ML model in Amazon Redshift by using the following code:

```
"CREATE MODEL customer_churn_auto_model
FROM (SELECT state,
    account_length,
    area_code,
    total_charge/account_length AS average_daily_spend,
    cust_serv_calls/account_length AS average_daily_cases,
    churn
FROM customer_activity
WHERE record_date < '2020-01-01'
)
TARGET churn
FUNCTION ml_fn_customer_churn_auto
IAM_ROLE 'arn:aws:iam::XXXXXXXXXXXX:role/Redshift-ML'
SETTINGS (
    S3_BUCKET 'your-bucket'
);
```

Note: The `SELECT state` can refer to Amazon Redshift regular tables, Amazon Redshift Spectrum external tables, or both.

Epics

Prepare a training and test dataset

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare a training and test dataset.</td>
<td>Sign in to the AWS Management Console and open the Amazon SageMaker console. Follow the instructions from the <a href="https://docs.aws.amazon.com/sagemaker/latest/dg/bt-dml.html">Build, train, and deploy a machine learning model</a> tutorial to create a .csv or Apache Parquet file that has a label column (supervised training) and no header. Note: We recommend that you shuffle and split the raw dataset into a training set for the model's training (70 percent) and a test set for the model's performance evaluation (30 percent).</td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

Prepare and configure the technology stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and configure an Amazon Redshift cluster.</td>
<td>On the Amazon Redshift console, create a cluster</td>
<td>DBA, Cloud architect</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket to store training data and model artifacts.</td>
<td>On the Amazon S3 console, create an S3 bucket for the training and test data. For more information about creating an S3 bucket, see Create an S3 bucket from AWS Quick Starts. <strong>Important</strong>: Make sure that your Amazon Redshift cluster and S3 bucket are in the same Region.</td>
<td>DBA, Cloud architect</td>
</tr>
<tr>
<td>Create and attach an IAM policy to the Amazon Redshift cluster.</td>
<td>Create an IAM policy to allow the Amazon Redshift cluster to access SageMaker and Amazon S3. For instructions and steps, see Cluster setup for using Amazon Redshift ML in the Amazon Redshift documentation.</td>
<td>DBA, Cloud architect</td>
</tr>
<tr>
<td>Allow Amazon Redshift users and groups to access schemas and tables.</td>
<td>Grant permissions to allow users and groups in Amazon Redshift to access internal and external schemas and tables. For steps and instructions, see Managing permissions and ownership in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

---

### Create and train the ML model in Amazon Redshift

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and train the ML model in Amazon Redshift.</td>
<td>Create and train your ML model in Amazon Redshift ML. For more information, see the CREATE MODEL statement in the Amazon Redshift documentation.</td>
<td>Developer, Data scientist</td>
</tr>
</tbody>
</table>
Perform batch inference and prediction in Amazon Redshift

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform inference using the generated ML model function.</td>
<td>For more information about performing inference by using the generated ML model function, see Prediction in the Amazon Redshift documentation.</td>
<td>Data scientist, Business intelligence user</td>
</tr>
</tbody>
</table>

Related resources

Prepare a training and test dataset
- Building, training, and deploying a machine learning model with Amazon SageMaker

Prepare and configure the technology stack
- Creating an Amazon Redshift cluster
- Choosing Amazon Redshift cluster maintenance tracks
- Creating an S3 bucket
- Setting up an Amazon Redshift cluster for using Amazon Redshift ML
- Managing permissions and ownership in Amazon Redshift

Create and train the ML model in Amazon Redshift
- CREATE MODEL statement in Amazon Redshift

Perform batch inference and prediction in Amazon Redshift
- Prediction in Amazon Redshift

Other resources
- Getting started with Amazon Redshift ML
- Creating, training, and deploying ML models in Amazon Redshift using SQL with Amazon Redshift ML
- Amazon Redshift partners
- AWS machine learning competency partners

Set up SSO for QuickSight by using AWS SSO and identity federation
Summary

This pattern shows you how to set up single sign-on (SSO) for Amazon QuickSight applications in the Amazon Web Services (AWS) Cloud by using AWS Single Sign-On (AWS SSO) and identity federation with Security Assertion Markup Language (SAML) 2.0. You can deliver a better user experience and improve security by centrally managing accounts and access control from AWS SSO. Your users can sign in to QuickSight applications by using their AWS SSO accounts, and they won't be prompted for credentials each time.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An active QuickSight subscription
- An organization in AWS Organizations (Note: If you don’t already have an organization, AWS SSO automatically creates an organization for you when you set up AWS SSO.)
- An AWS SSO identity source with a built-in identity store (Note: You can also use AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft AD, or an on-premises Active Directory (AD) that uses AD Connector as the identity store.)

Product versions

This pattern is tested on AWS Managed Microsoft AD with Active Directory Federation Services, but it will work the same way with other identity federation services.

Architecture

The following diagram shows how users get SSO access to a QuickSight application through an architecture that uses AWS SSO and identity federation with SAML 2.0.
The diagram shows the following workflow:

1. The user provides AD authentication credentials in the web endpoint URL.
2. AWS Managed Microsoft AD authenticates users and then posts the SAML assertion to the AWS SSO sign-in endpoint.
3. The sign-in endpoint receives the SAML request, processes the request, and then creates temporary credentials.
4. The temporary credentials are validated in the member account by the AWS Identity and Access Management (IAM) roles that are provisioned by AWS SSO.
5. The IAM roles authenticate the permissions and pass them to QuickSight.

**Technology Stack**

- AWS Identity and Access Management (IAM)
- AWS Managed Microsoft AD
- AWS Organizations
- AWS Single Sign-On
- Amazon QuickSight

**Tools**

- **AWS Identity and Access Management (IAM)** – IAM is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- **AWS Managed Microsoft AD** – AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft Active Directory, enables your directory-aware workloads and AWS resources to use managed AD in AWS.
- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.
• **AWS Single Sign-On** – AWS Single Sign-On is a cloud-based SSO service that makes it easy to centrally manage SSO access to all of your AWS accounts and cloud applications.

• **Amazon QuickSight** – Amazon QuickSight is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service built for the cloud.

## Epics

### Configure AWS SSO in the AWS Organizations management account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add QuickSight as an application.         | 1. Sign in to the AWS Management Console and then open the AWS SSO console.  
2. In the navigation pane, choose Applications and then choose Add a new application.  
3. In the AWS SSO Application Catalog section, choose Amazon QuickSight from the catalog.  
4. On the Configure Amazon QuickSight page, under Details, enter a unique display name for your application for Display name (for example, Amazon QuickSight).  
5. In the AWS SSO metadata section, choose Download for AWS SSO SAML metadata file.  
7. Choose Save changes.                                                                 | Cloud administrator   |
| Configure attributes in the SSO application. | 1. Choose the Attribute mappings tab and then choose Add a new attribute mapping.  
2. For User attribute in the application, enter `https://aws.amazon.com/SAML/Attributes/Role`.  
3. For Maps to this string value or user attribute                                                                 | Cloud administrator   |
### Task: Assign users to the application

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>in AWS SSO</strong>, enter <code>&lt;IDP ARN&gt;</code>, <code>&lt;Role ARN&gt;</code>. <strong>Note:</strong> These Amazon Resource Names (ARNs) are intended for the identity provider and IAM role that you created earlier. 4. Choose <strong>Save changes.</strong></td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

**Note:** If no users or groups are available, you can create a user and group for testing purposes. Open the AWS SSO console, choose **Users** from the navigation pane, and then create a user. For example, create a user named **Analyst01**. To create a group, choose **Groups** in the navigation pane, and then create a group called **Analyst**. Then, you can assign the **Analyst01** user to the **Analyst** group.

### Configure the IAM role for the member account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a SAML identity provider.</strong> 1. Sign in to the AWS Management Console and then open the IAM console. 2. In the navigation pane, choose <strong>Identity providers</strong>, and then choose <strong>Add provider</strong>. 3. For <strong>Provider type</strong>, choose <strong>SAML</strong>. 4. For <strong>Provider name</strong>, enter a name for the identity provider. 5. For <strong>Metadata document</strong>, choose <strong>Choose file</strong>, and</td>
<td>Cloud administrator</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>then choose the SAML metadata document that you downloaded earlier.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>(Optional) For <strong>Add tags</strong>, add key–value pairs to help you identify and organize your identity providers.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Note the ARN of the identity provider. You will use it later when you configure attributes in the SSO application.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Choose <strong>Add provider</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Create a SAML 2.0 federation IAM role.</strong></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>Open the <a href="https://aws.amazon.com/iam/">IAM console</a>.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>2.</td>
<td>In the navigation pane, choose <strong>Roles</strong>, and then choose <strong>Create role</strong>.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>For <strong>Trusted entity type</strong>, select <strong>SAML 2.0 federation</strong>.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>For <strong>Choose a SAML 2.0 provider</strong>, select the SAML provider that you created earlier, and then choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>On the <strong>Add Permissions page</strong>, attach the appropriate policies to limit the role permissions to QuickSight only by attaching an inline policy to the role. For an example policy, see <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id_policies-examples.html">IAM policy</a> in the Additional information section of this pattern.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>On the <strong>Name, review, and create</strong> page, under <strong>Role details</strong>, enter a name for <strong>Role name</strong>.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>(Optional) For <strong>Add tags</strong>, add key–value pairs to help you identify and organize your roles.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Note the ARN of this role. You will use it later when you configure attributes in your SSO application.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Choose <strong>Create role</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

**Additional information**

**IAM policy**
The following IAM policy allows users to access QuickSight and create both author accounts and reader accounts.

```
{
    "Statement": [
      {
        "Action": [
          "quicksight:CreateUser"
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:quicksight::<YOUR_AWS_ACCOUNT_ID>:user/${aws:userid}"
        ]
      },
      "Version": "2012-10-17"
    ]
}
```

---

**Subscribe a Lambda function to event notifications from S3 buckets in different AWS Regions**

*Created by Suresh Konathala (AWS) and Arindom Sarkar (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Analytics</th>
<th>AWS services: AWS Lambda; Amazon S3; Amazon SNS; Amazon SQS</th>
</tr>
</thead>
</table>

**Summary**

Amazon Simple Storage Service (Amazon S3) Event Notifications publishes notifications for certain events in your S3 bucket (for example, object created events, object removal events, or restore object events). You can use an AWS Lambda function to process these notifications according to your application's requirements. However, the Lambda function can't directly subscribe to notifications from S3 buckets that are hosted in different AWS Regions.

This pattern's approach deploys a fanout scenario to process Amazon S3 notifications from cross-Region S3 buckets by using an Amazon Simple Notification Service (Amazon SNS) topic for each Region. These Regional SNS topics send the Amazon S3 event notifications to an Amazon Simple Queue Service (Amazon SQS) queue in a central Region that also contains your Lambda function. The Lambda function subscribes to this SQS queue and processes the event notifications according to your organization's requirements.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- Existing S3 buckets in multiple Regions, including a central Region to host the Amazon SQS queue and Lambda function.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
Familiarity with the fanout scenario in Amazon SNS. For more information about this, see Common Amazon SNS scenarios in the Amazon SNS documentation.

Architecture

The following diagram shows the architecture for this pattern’s approach.

The diagram shows the following workflow:

1. Amazon S3 sends event notifications about S3 buckets (for example, object created, object removed, or object restored) to an SNS topic in the same Region.
2. The SNS topic publishes the event to an SQS queue in the central Region.
3. The SQS queue is configured as the event source for your Lambda function and buffers the event messages for the Lambda function.
4. The Lambda function polls the SQS queue for messages and processes the Amazon S3 event notifications according to your application’s requirements.

Technology stack

- Lambda
- Amazon SNS
- Amazon SQS
- Amazon S3

Tools

- AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS
CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

- **Amazon SQS** – Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components. Amazon SQS supports both standard and FIFO queues.

## Epics

### Create the SQS queue and Lambda function in your central Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an SQS queue with a Lambda trigger. | Sign in to the AWS Management Console and use the instructions from the tutorial Using Lambda with Amazon SQS in the AWS Lambda documentation to create the following resources in your central Region:  
  - A Lambda execution role  
  - A Lambda function to process the Amazon S3 events  
  - An SQS queue  
  
  **Note:** Make sure that you configure the SQS queue as the event source for your Lambda function. | AWS DevOps, Cloud architect |

### Create an SNS topic and set up event notifications for the S3 buckets in each required Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SNS topic to receive Amazon S3 event notifications.</td>
<td>Create an SNS topic in a Region that you want to receive Amazon S3 event notifications from. For</td>
<td>AWS DevOps, Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Subscribe the SNS topic to the central SQS queue.</td>
<td>Subscribe your SNS topic to the SQS queue hosted by your central Region. For more information about this, see Subscribing to an SNS topic in the Amazon SNS documentation.</td>
<td>AWS DevOps, Cloud architect</td>
</tr>
</tbody>
</table>
| Update the SNS topic's access policy. | 1. Open the Amazon SNS console, choose Topics, and then choose the SNS topic that you created earlier.  
2. Choose Edit and then expand the Access policy - optional section.  
3. Attach the following access policy to your SNS topic to allow sns:publish permission for Amazon S3 and then choose Save:  

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "0",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-west-2::s3Events-SNSTopic-us-west-2"
        }
    ]
}
``` | AWS DevOps, Cloud architect |
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up notifications for each S3 bucket in the Region.</td>
<td>Set up event notifications for each S3 bucket in the Region. For more information about this, see Enabling and configuring event notifications using the Amazon S3 console in the Amazon S3 documentation. Note: In the Destination section, choose SNS topic and specify the ARN of the SNS topic that you created earlier.</td>
<td>AWS DevOps, Cloud architect</td>
</tr>
<tr>
<td>Repeat this epic for all required Regions.</td>
<td>Important: Repeat the tasks in this epic for each Region that you want to receive Amazon S3 event notifications from, including your central Region.</td>
<td>AWS DevOps, Cloud architect</td>
</tr>
</tbody>
</table>

### Related resources

- Configuring an access policy (Amazon SQS documentation)
- Configuring an SQS queue as an event source (AWS Lambda documentation)
- Configuring an SQS queue to initiate a Lambda function (Amazon SQS documentation)
- AWS::Lambda::Function resource (AWS CloudFormation documentation)

### Three AWS Glue ETL job types for converting data to Apache Parquet

*Created by Adnan Alvee (AWS)*

**Environment:** PoC or pilot  
**Technologies:** Analytics  
**Workload:** All other workloads  
**AWS services:** AWS Glue

### Summary

On the Amazon Web Services (AWS) Cloud, AWS Glue is a fully managed extract, transform, and load (ETL) service. AWS Glue makes it cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.

This pattern provides different job types in AWS Glue and uses three different scripts to demonstrate authoring ETL jobs.

You can use AWS Glue to write ETL jobs in a Python shell environment. You can also create both batch and streaming ETL jobs by using Python (PySpark) or Scala in a managed Apache Spark environment.
To get you started with authoring ETL jobs, this pattern focuses on batch ETL jobs using Python shell, PySpark, and Scala. Python shell jobs are meant for workloads requiring lesser compute power. The managed Apache Spark environment is meant for workloads requiring high compute power.

Apache Parquet is built to support efficient compression and encoding schemes. It can speed up your analytics workloads because it stores data in a columnar fashion. Converting data to Parquet can save you both storage space, cost and time in the longer run. To learn more about Parquet, see the blog post Apache Parquet: How to be a hero with the open-source columnar data format on Google, Azure, and Amazon cloud.

Prerequisites and limitations

Prerequisites

- AWS Identity and Access Management (IAM) role

Architecture

Target technology stack

- AWS Glue
- Amazon Simple Storage Service (Amazon S3)
- Apache Parquet

Automation and scale

AWS Glue workflows support full automation of an ETL pipeline.

You can change the number of data processing units (DPUs), or worker types, to scale horizontally and vertically.

Tools

Tools

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) provides storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- AWS Glue – AWS Glue is a fully managed ETL service for categorizing, cleaning, enriching, and moving your data between various data stores and data streams.

Configuration

Following are the settings for configuring the compute power of AWS Glue ETL. To reduce costs, use the minimal settings when you run the workload that is provided in this pattern.

- **Python shell** – You can use 1 DPU to utilize 16 GB of memory or 0.0625 DPU to utilize 1 GB of memory. This pattern uses 0.0625 DPU, which is the default in the AWS Glue console.
- **Python or Scala for Spark** – If you choose the Spark-related job types in the console, AWS Glue by default uses 10 workers and the G.1X worker type. This pattern uses two workers, which is the minimum number allowed, with the standard worker type, which is sufficient and cost effective.
The following table displays the different AWS Glue worker types for the Apache Spark environment. Because a Python shell job does not use the Apache Spark environment to run Python, it is not included in the table.

<table>
<thead>
<tr>
<th></th>
<th>Standard</th>
<th>G.1X</th>
<th>G.2X</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCPU</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Memory</td>
<td>16 GB</td>
<td>16 GB</td>
<td>32 GB</td>
</tr>
<tr>
<td>Disk space</td>
<td>50 GB</td>
<td>64 GB</td>
<td>128 GB</td>
</tr>
<tr>
<td>Executor per worker</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Code

When you create the AWS Glue jobs, you can use either the IAM role that is attached or an existing role.

#### AWS Glue Python shell

The Python code uses the Pandas and PyArrow libraries to convert data to Parquet. The Pandas library is already available. The PyArrow library is downloaded when you run the pattern, because it is a one-time run. You can use wheel files to convert PyArrow to a library and provide the file as a library package. For more information about packaging wheel files, see Providing your own Python library.

```python
from io import BytesIO
import pandas as pd
import boto3
import os
import io
import site
from importlib import reload
from setuptools.command import easy_install
install_path = os.environ['GLUE_INSTALLATION']
easy_install.main( ["--install-dir", install_path, "pyarrow"] )
reload(site)
import pyarrow

input_loc = "bucket-name/prefix/sample_data.csv"
output_loc = "bucket-name/prefix/"

input_bucket = input_loc.split('/', 1)[0]
object_key = input_loc.split('/', 1)[1]

output_loc_bucket = output_loc.split('/', 1)[0]
output_loc_prefix = output_loc.split('/', 1)[1]

s3 = boto3.client('s3')
obj = s3.get_object(Bucket=input_bucket, Key=object_key)
df = pd.read_csv(io.BytesIO(obj['Body'].read()))

parquet_buffer = BytesIO()
s3_resource = boto3.resource('s3')
df.to_parquet(parquet_buffer, index=False)
s3_resource.Object(output_loc_bucket, output_loc_prefix + 'data' + '.parquet').put(Body=parquet_buffer.getvalue())
```
**AWS Glue Spark job with Python**

To use an AWS Glue Spark job type with Python, choose Spark as the job type. Choose Spark 2.4, Python 3 with improved job startup time (Glue Version 2.0) as the AWS Glue version.

```python
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
from awsglue.dynamicframe import DynamicFrame
from awsglue.utils import getResolvedOptions
from awsglue.job import Job

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

input_loc = "bucket-name/prefix/sample_data.csv"
output_loc = "bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(
    connection_type = "s3",
    connection_options = {
        "paths": [input_loc],
        "format": "csv",
        "format_options": {
            "withHeader": True,
            "separator": "",
        }
    }
)

outputDF = glueContext.write_dynamic_frame.from_options(
    frame = inputDyf,
    connection_type = "s3",
    connection_options = {
        "path": output_loc,
        "format": "parquet"
    }
)
```

**AWS Glue Spark job with Scala**

To use an AWS Glue Spark job type with Scala, choose Spark as the job type. Choose Spark 2.4, Scala 2 with improved job startup time (Glue Version 2.0) as the AWS Glue version. To save on storage space, the following AWS Glue with Scala sample also uses the applyMapping feature to convert data types.

```scala
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.DynamicFrame
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters

object GlueScalaApp {
  def main(sysArgs: Array[String]) {
    @transient val spark: SparkContext = SparkContext.getOrCreate()
    val glueContext: GlueContext = new GlueContext(spark)
    // Additional code to process data...
  }
}
```
val inputLoc = "s3://bucket-name/prefix/sample_data.csv"
val outputLoc = "s3://bucket-name/prefix/"

val readCSV = glueContext.getSource("csv", JsonOptions(Map("paths" -> Set(inputLoc)))).getDynamicFrame()

val applyMapping = readCSV.applyMapping(mappings = Seq(("_c0", "string", "date", "string"),
("_c1", "string", "sales", "long"),
("_c2", "string", "profit", "double")), caseSensitive = false)

val formatPartition = applyMapping.toDF().coalesce(1)
val dynamicFrame = DynamicFrame(formatPartition, glueContext)
val dataSink = glueContext.getSinkWithFormat(
connectionType = "s3",
options = JsonOptions(Map("path" -> outputLoc )),
transformationContext = "dataSink", format = "parquet").writeDynamicFrame(dynamicFrame)
}

Epics

Upload the data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the data into a new or existing S3 bucket.</td>
<td>Create or use an existing S3 bucket in your account. Upload the sample_data.csv file from the Attachments section, and note the S3 bucket and prefix location.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Create and run the AWS Glue job

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS Glue job.</td>
<td>Under the ETL section of the AWS Glue console, add an AWS Glue job. Select the appropriate job type, AWS Glue version, and the corresponding DPU/Worker type and number of workers. For details, see the Configuration section.</td>
<td>Developer, cloud or data</td>
</tr>
<tr>
<td>Change the input and output locations.</td>
<td>Copy of the code corresponding to your Glue job and change the input and output location that you noted in the Upload the data epic.</td>
<td>Developer, cloud or data</td>
</tr>
<tr>
<td>Run the ETL job.</td>
<td>Run your job and check the output. Note how much space</td>
<td>Developer, cloud or data</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
--- | --- | ---
was reduced from the original file. | | |

### Related resources

#### References
- Apache Spark
- AWS Glue: How it works
- AWS Glue pricing

#### Tutorials and videos
- What is AWS Glue?

### Additional information

#### Parameter configuration

You can use the following snippets to set parameters for your ETL job. AWS Glue uses four argument names internally:

- --conf
- --debug
- --mode
- --JOB_NAME

The --JOB_NAME parameter must be explicitly entered on the AWS Glue console. Choose Jobs, Edit Job, Security configuration, script libraries, and job parameters (optional). Enter --JOB_NAME as the key and provide a value. You can also use the AWS Command Line Interface (AWS CLI) or the AWS Glue API to set this parameter. The --JOB_NAME parameter is used by Spark and is not needed in a Python shell environment job.

You must add -- before every parameter name; otherwise, the code will not work. For example, for the following code snippets, the location parameters must be invoked by --input_loc and --output_loc.

**AWS Glue Python shell**

```python
from awsglue.utils import getResolvedOptions
args = getResolvedOptions(sys.argv, ['input_loc', 'output_loc'])
```

**AWS Glue Python**

```python
from awsglue.utils import getResolvedOptions
```
Visualize Amazon Redshift audit logs using Amazon Athena and Amazon QuickSight

Created by Sanket Sirsikar (AWS) and Gopal Krishna Bhatia (AWS)

| Environment: | PoC or pilot | Technologies: | Analytics; Big data; Data lakes | AWS services: | Amazon Athena; Amazon Redshift; Amazon S3; Amazon QuickSight |

**Summary**

Security is an integral part of database operations on the Amazon Web Services (AWS) Cloud. Your organization should ensure that it monitors database user activities and connections to detect potential security incidents and risks. This pattern helps you monitor your databases for security and troubleshooting purposes, which is a process known as database auditing.

This pattern provides a SQL script that automates the creation of an Amazon Athena table and views for a reporting dashboard in Amazon QuickSight that helps you audit Amazon Redshift logs. This ensures that users responsible for monitoring database activities have convenient access to data security features.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- An existing Amazon Redshift cluster. For more information about this, see [Create an Amazon Redshift cluster](https://docs.aws.amazon.com/redshift/latest/guides/cluster-creation.html) in the Amazon Redshift documentation.
- Access to an existing Athena workgroup. For more information, see [How workgroups work](https://docs.aws.amazon.com/athena/latest/ug/workgroup.html) in the Amazon Athena documentation.
- An existing Amazon Simple Storage Service (Amazon S3) source bucket with the required AWS Identity and Access Management (IAM) permissions. For more information, see [Bucket permissions for Amazon Redshift audit logging](https://docs.aws.amazon.com/redshift/latest/gsg/audit-trail-log-permissions.html) from [Database audit logging](https://docs.aws.amazon.com/redshift/latest/gsg/audit-trail-log-permissions.html) in the Amazon Redshift documentation.
Architecture

Technology stack

• Athena
• Amazon Redshift
• Amazon S3
• QuickSight

Tools

• Amazon Athena – Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.
• Amazon QuickSight – QuickSight is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service.
• Amazon Redshift – Amazon Redshift is an enterprise-level, petabyte scale, fully managed data warehousing service.
• Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

Epics

Configure the Amazon Redshift cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Enable audit logging for the Amazon Redshift cluster. | 1. Sign in to the AWS Management Console, open the Amazon Redshift console, choose CLUSTERS, and then choose the cluster that you want to enable logging for.  
2. Choose the Properties tab and then enable auditing by following the instructions from Configuring auditing using the console | DBA, Data engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable logging in the Amazon Redshift cluster parameter group.</td>
<td>You can enable auditing of connection logs, user logs, and user activity logs at the same time by using the AWS Management Console, the Amazon Redshift API reference, or AWS Command Line Interface (AWS CLI). For auditing of user activity logs, you must enable the <code>enable_user_activity_logging</code> database parameter. If you only enable the audit logging feature but not the associated parameter, the database audit logs the logging information for the connection and user logs but not for the user activity logs. The <code>enable_user_activity_logging</code> parameter is not enabled by default, but you can enable it by changing it from <code>false</code> to <code>true</code>. <strong>Important:</strong> You need to create a new cluster parameter group with the <code>user_activity_logging</code> parameter enabled and attach it to your Amazon Redshift cluster. For more information about this, see <a href="https://docs.aws.amazon.com/redshift/latest/userguide/cluster-modify.html">Modifying a cluster</a> in the Amazon Redshift documentation. For more information about this task, see [Amazon Redshift parameter groups](<a href="https://docs.aws.amazon.com/redshift/latest/userguide/redshift-parameter-group">https://docs.aws.amazon.com/redshift/latest/userguide/redshift-parameter-group</a> onCreateOptionsMenu.html) and <a href="https://docs.aws.amazon.com/redshift/latest/userguide/redshift-console.html">Configuring auditing using the console</a> in the Amazon Redshift documentation.</td>
<td>DBA, Data engineer</td>
</tr>
</tbody>
</table>
## Configure S3 bucket permissions for Amazon Redshift cluster logging

**Task**: Configure S3 bucket permissions for Amazon Redshift cluster logging.

**Description**: When you enable logging, Amazon Redshift collects logging information and uploads it to log files stored in an S3 bucket. You can use an existing S3 bucket or create a new bucket.

**Important**: Make sure that Amazon Redshift has the required IAM permissions to access the S3 bucket. For more information about this, see [Bucket permissions for Amazon Redshift audit logging](https://docs.aws.amazon.com/redshift/latest/aarch2/guides/audit-log-permissions.html) from Database audit logging in the Amazon Redshift documentation.

**Skills required**: DBA, Data engineer

### Create the Athena table and views

**Task**: Create the Athena table and views to query Amazon Redshift audit log data from the S3 bucket.

**Description**: Open the Amazon Athena console and use the data definition language (DDL) query from the `AuditLogging.sql` SQL script (attached) to create the table and views for user activity logs, user logs, and connection logs.

For more information and instructions, see the [Create tables and run queries](https://docs.aws.amazon.com/athena/latest/ug/workshop-athena-table-query.html) tutorial from the Amazon Athena Workshop.

**Skills required**: Data engineer

### Set up log monitoring in the QuickSight dashboard

**Task**: Set up log monitoring in the QuickSight dashboard.

**Description**: Open the Amazon QuickSight console and create a QuickSight dashboard by following the instructions in the [Visualize with QuickSight using Athena](https://docs.aws.amazon.com/athena/latest/ug/workshop-quick-sight-analytics.html) tutorial from the Amazon Athena Workshop.

**Skills required**: DBA, Data engineer
Related resources

- Create tables and run queries in Athena
- Visualize with QuickSight using Athena

Attachments

attachment.zip

More patterns

- Automate data ingestion from AWS Data Exchange into Amazon S3 (p. 278)
- Automatically extract content from PDF files using Amazon Textract (p. 643)
- Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code (p. 281)
- Embed an Amazon QuickSight dashboard in a local Angular application (p. 1848)
- Ensure an Amazon Redshift cluster is encrypted upon creation (p. 1679)
- Ensure encryption for Amazon EMR data at rest is enabled at launch (p. 1671)
- Extract and query AWS IoT SiteWise metadata attributes in a data lake (p. 630)
- Identify and alert when Amazon Kinesis Data Firehose resources are not encrypted with an AWS KMS key (p. 684)
- Migrate a self-hosted MongoDB environment to MongoDB Atlas on the AWS Cloud (p. 1198)
- Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT (p. 891)
- Migrate data from an on-premises Hadoop environment to Amazon S3 using DistCp with AWS PrivateLink for Amazon S3 (p. 1816)
- Migrate from Couchbase Server to Couchbase Cloud on AWS (p. 1164)
- Monitor Amazon EMR clusters for in-transit encryption at launch (p. 1692)
- Verify that new Amazon Redshift clusters have required SSL endpoints (p. 1563)
- Verify that new Amazon Redshift clusters launch in a VPC (p. 1570)
- Visualize AI/ML model results using Flask and AWS Elastic Beanstalk (p. 672)
Cloud-native

Topics
- Build a video processing pipeline by using Amazon Kinesis Video Streams and AWS Fargate (p. 93)
- Copy data from an S3 bucket in one account and Region to another account and Region (p. 100)
- Successfully import an S3 bucket as an AWS CloudFormation stack (p. 103)
- More patterns (p. 111)

Build a video processing pipeline by using Amazon Kinesis Video Streams and AWS Fargate

Created by Piotr Chotkowski (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Cloud-native; Software development &amp; testing; Media services</td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS Fargate; Amazon Kinesis; Amazon S3</td>
</tr>
</tbody>
</table>

Summary

This pattern demonstrates how to use Amazon Kinesis Video Streams and AWS Fargate to extract frames from a video stream and store them as image files for further processing in Amazon Simple Storage Service (Amazon S3).

The pattern provides a sample application in the form of a Java Maven project. This application defines the AWS infrastructure by using the AWS Cloud Development Kit (AWS CDK). Both the frame processing logic and the infrastructure definitions are written in the Java programming language. You can use this sample application as a basis for developing your own real-time video processing pipeline or to build the video preprocessing step of a machine learning pipeline.

Prerequisites and limitations

Prerequisites
- An active AWS account
- Java SE Development Kit (JDK) 11, installed
- Apache Maven, installed
- AWS Cloud Development Kit (AWS CDK), installed
- AWS Command Line Interface (AWS CLI) version 2, installed
- Docker (required for building Docker images to use in AWS Fargate task definitions), installed

Limitations

This pattern is intended as a proof of concept, or as a basis for further development. It should not be used in its current form in production deployments.
Product versions

- This pattern was tested with the AWS CDK version 1.77.0 (see AWS CDK versions)
- JDK 11
- AWS CLI version 2

Architecture

Target technology stack

- Amazon Kinesis Video Streams
- AWS Fargate task
- Amazon Simple Queue Service (Amazon SQS) queue
- Amazon S3 bucket

Target architecture

The user creates a Kinesis video stream, uploads a video, and sends a JSON message that contains details about the input Kinesis video stream and the output S3 bucket to an SQS queue. AWS Fargate, which is running the main application in a container, pulls the message from the SQS queue and starts extracting frames. Each frame is saved in an image file and stored in the target S3 bucket.

Automation and scale

The sample application can scale both horizontally and vertically within a single AWS Region. Horizontal scaling can be achieved by increasing the number of deployed AWS Fargate tasks that read from the SQS queue. Vertical scaling can be achieved by increasing the number of frame-splitting and image-publishing threads in the application. These settings are passed as environment variables to the application in the definition of the QueueProcessingFargateService resource in the AWS CDK. Due to the nature of AWS CDK stack deployment, you can deploy this application in multiple AWS Regions and accounts with no additional effort.

Tools
• **AWS CDK** – The AWS Cloud Development Kit (AWS CDK) is a software development framework for defining your cloud infrastructure and resources by using programming languages such as TypeScript, JavaScript, Python, Java, and C#/.Net.

• **Amazon Kinesis Video Streams** – Amazon Kinesis Video Streams is a fully managed AWS service that you can use to stream live video from devices to the AWS Cloud, or build applications for real-time video processing or batch-oriented video analytics.

• **AWS Fargate** – AWS Fargate is a serverless compute engine for containers. Fargate removes the need to provision and manage servers, and lets you focus on developing your applications.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance.

• **Amazon SQS** – Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

**Code**

- A .zip file of the sample application project (frame-splitter-code.zip) is attached.

**Epics**

**Deploy the infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the Docker daemon.</td>
<td>Start the Docker daemon on your local system. The AWS CDK uses Docker to build the image that is used in the AWS Fargate task. You must run Docker before you proceed to the next step.</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td>Build the project.</td>
<td>Download the frame-splitter-code sample application (attached) and extract its contents into a folder on your local machine. Before you can deploy the infrastructure, you have to build the Java Maven project. At a command prompt, navigate to the root directory of the project, and build the project by running the command: mvn clean install</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td>Bootstrap the AWS CDK.</td>
<td>(First-time AWS CDK users only) If this is the first time you’re using the AWS CDK, you might have to bootstrap the environment by running the AWS CLI command:</td>
<td>Developer, DevOps engineer</td>
</tr>
</tbody>
</table>
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CDK stack.</td>
<td>In this step, you create the required infrastructure resources (SQS queue, S3 bucket, AWS Fargate task definition) in your AWS account, build the Docker image that is required for the AWS Fargate task, and deploy the application. At a command prompt, navigate to the root directory of the project, and run the command: <code>cdk deploy --profile &quot;$AWS_PROFILE_NAME&quot; --all</code> where <code>$AWS_PROFILE_NAME</code> holds the name of the AWS profile from your AWS credentials. Or, you can remove this parameter to use the default profile. Confirm the deployment. Note the <strong>QueueUrl</strong> and <strong>Bucket</strong> values from the CDK deployment output; you will need these in later steps. The AWS CDK creates the assets, uploads them to your AWS account, and creates all infrastructure resources. You can observe the resource creation process in the AWS CloudFormation console at <a href="https://console.aws.amazon.com/cloudformation/">https://console.aws.amazon.com/cloudformation/</a>. For more information, see the <a href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide">AWS CloudFormation documentation</a> and the <a href="https://docs.aws.amazon.com/cdk/latest/guide">AWS CDK documentation</a>.</td>
<td>Developer, DevOps engineer</td>
</tr>
</tbody>
</table>

<p>| CDK bootstrap --profile &quot;$AWS_PROFILE_NAME&quot; | where <code>$AWS_PROFILE_NAME</code> holds the name of the AWS profile from your AWS credentials. Or, you can remove this parameter to use the default profile. For more information, see the <a href="https">AWS CDK documentation</a>. |  |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a video stream.</td>
<td>In this step, you create a Kinesis video stream that will serve as an input stream for video processing. Make sure that you have the AWS CLI installed and configured. In the AWS CLI, run: <code>aws kinesisvideo --profile &quot;$AWS_PROFILE_NAME&quot; create-stream --stream-name &quot;$STREAM_NAME&quot; --data-retention-in-hours &quot;24&quot;</code> where <code>$AWS_PROFILE_NAME</code> holds the name of the AWS profile from your AWS credentials (or remove this parameter to use the default profile) and <code>$STREAM_NAME</code> is any valid stream name. Alternatively, you can create a video stream by using the Kinesis console by following the steps in the Kinesis Video Streams documentation. Note the AWS Resource Name (ARN) of the created stream; you will need it later.</td>
<td>Developer, DevOps engineer</td>
</tr>
</tbody>
</table>

**Run an example**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the video to the stream.</td>
<td>In the project folder for the sample frame-splitter-code application, open the <code>ProcessingTaskTest.java</code> file in the <code>src/test/java/amazon/awscdk/examples/splitter</code> folder. Replace the <code>profileName</code> and <code>streamName</code> variables with the values you used in the previous steps. To upload the example video to the Kinesis video stream you created in the previous step, run: <code>amazon.awscdk.examples.splitter.ProcessingTaskTest#testExample</code> Alternatively, you can upload your video by using one of the</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Initiate video processing.</td>
<td>Now that you have uploaded a video to the Kinesis video stream, you can start processing it. To initiate the processing logic, you have to send a message with details to the SQS queue that the AWS CDK created during deployment. To send a message by using the AWS CLI, run:</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td><strong>aws sqs --profile &quot;$AWS_PROFILE_NAME&quot; send-message --queue-url QUEUE_URL --message-body MESSAGE</strong> where $AWS_PROFILE_NAME holds the name of the AWS profile from your AWS credentials (remove this parameter to use the default profile), QUEUE_URL is the QueueUrl value from the AWS CDK output, and MESSAGE is a JSON string in the following format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{ &quot;streamARN&quot;: &quot;STREAM_ARN&quot;, &quot;bucket&quot;: &quot;BUCKET_NAME&quot;, &quot;s3Directory&quot;: &quot;test-output&quot; } where STREAM_ARN is the ARN of the video stream you created in an earlier step and BUCKET_NAME is the Bucket value from the AWS CDK output. Sending this message initiates video processing. Alternatively, you can send a message by using the Amazon SQS console, as described in the Amazon SQS documentation.</td>
<td></td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns

Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>View images of the video frames.</td>
<td>You can see the resulting images in the S3 output bucket s3://BUCKET_NAME/test-output where BUCKET_NAME is the Bucket value from the AWS CDK output.</td>
<td>Developer, DevOps engineer</td>
</tr>
</tbody>
</table>

Related resources

- AWS CDK documentation
- AWS CDK API reference
- AWS CDK introductory workshop
- Amazon Kinesis Video Streams documentation
- Example: Identifying Objects in Video Streams Using SageMaker
- Example: Parsing and Rendering Kinesis Video Streams Fragments
- Analyze live video at scale in real time using Amazon Kinesis Video Streams and Amazon SageMaker (AWS Machine Learning blog post)
- AWS Fargate Getting Started

Additional information

Choosing an IDE

We recommend that you use your favorite Java IDE to build and explore this project.

Cleaning up

After you finish running this example, remove all deployed resources to avoid incurring additional AWS infrastructure costs.

To remove the infrastructure and the video stream, use these two commands in the AWS CLI:

```bash
cdk destroy --profile "$AWS_PROFILE_NAME" --all
aws kinesisvideo --profile "$AWS_PROFILE_NAME" delete-stream --stream-arn "$STREAM_ARN"
```

Alternatively, you can remove the resources manually by using the AWS CloudFormation console to remove the AWS CloudFormation stack, and the Kinesis console to remove the Kinesis video stream. Note that `cdk destroy` doesn't remove the output S3 bucket or the images in Amazon Elastic Container Registry (Amazon ECR) repositories (aws-cdk/assets). You have to remove them manually.

Attachments

attachment.zip
Copy data from an S3 bucket in one account and Region to another account and Region

*Created by Appasaheb Bagali (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Cloud-native; Security, identity, compliance; Storage &amp; backup; Migration</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
</table>

**AWS services:** Amazon S3

**Summary**

This pattern describes how to copy data from an Amazon Simple Storage Service (Amazon S3) bucket in an Amazon Web Services (AWS) account and AWS Region to an S3 bucket in another account and Region.

This pattern uses a source account and a destination account in different Regions. You attach a bucket policy to your source S3 bucket that grants the destination account access through AWS Identity and Access Management (IAM). You then create an IAM policy in your destination account that allows a user to perform `PutObject` and `GetObject` actions on the source S3 bucket. Finally, you run `copy` and `sync` commands to transfer data from the source S3 bucket to the destination S3 bucket.

Accounts own the objects that they upload to S3 buckets. If you copy objects across different accounts and Regions, you grant the destination account ownership of the copied objects. You can change the ownership of an object by changing its access control list (ACL) to `bucket-owner-full-control`. However, we recommend that you grant programmatic cross-account permissions to the destination account because ACLs can be difficult to manage for multiple objects.

**Prerequisites and limitations**

**Prerequisites**

- Two active AWS accounts in different AWS Regions.
- An existing S3 bucket in the source account.
- AWS Command Line Interface (AWS CLI), installed and configured.
- If your source or destination S3 bucket has default encryption enabled, you must modify the AWS Key Management Service (AWS KMS) key permissions. For more information, see Allow users to download from and upload to an S3 bucket with default encryption in the AWS Knowledge Center.
- Familiarity with cross-account permissions.
Architecture

Technology stack

- Amazon S3
- IAM
- AWS CLI

Tools

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- AWS CLI – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services.
- IAM – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.

Code

Copy

```
aws s3 cp s3:// DOC-EXAMPLE-BUCKET-SOURCE /\n   s3:// DOC-EXAMPLE-BUCKET-TARGET /\n   --recursive --source-region SOURCE-REGION-NAME --region DESTINATION-REGION-NAME
```

Synchronize

```
aws s3 sync s3:// DOC-EXAMPLE-BUCKET-SOURCE /\ns3:// DOC-EXAMPLE-BUCKET-TARGET /\n   --source-region SOURCE-REGION-NAME --region DESTINATION-REGION-NAME
```
## Epics

Create and attach the S3 bucket policy in the source account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and attach an S3 bucket policy.</td>
<td>Sign in to the AWS Management Console for your source account and open the Amazon S3 console. Choose your source S3 bucket and then choose Permissions. Under Bucket policy, choose Edit and then paste the bucket policy from the sourcebucket-policy.json file (attached). Choose Save. <strong>Important:</strong> Make sure that you include the AWS account ID for the destination account and configure the bucket policy template according to your requirements.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

Configure the destination account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a destination S3 bucket.</td>
<td>Sign in to the AWS Management Console for your destination account, open the Amazon S3 console, and then choose Create bucket. Create an S3 bucket according to your requirements. For more information, see Creating a bucket in the Amazon S3 documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an IAM policy for the IAM user.</td>
<td>Open the IAM console, choose Policies, and then choose Create policy. Choose the JSON tab and paste the destination-IAM-user-policy.json file (attached). Choose Review policy, enter a policy name and description, and then choose Create policy. Choose either Programmatic access or AWS Management Console access. We recommend that you choose Programmatic access if users require access to the API, AWS CLI, or tools.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
## Task Description

For Windows PowerShell. This creates an access key for each new user. You can view or download the access keys when you are on the confirmation page.

Choose Next: Permissions and then choose Attach existing policies directly. Attach the policy that you created earlier. Choose Next: Tags, and then choose Next: Review to finalize the user configuration.

### Related resources

- Creating an S3 bucket
- Amazon S3 bucket policies and user policies
- IAM identities (users, groups, and roles)

### Attachments

attachment.zip

## Successfully import an S3 bucket as an AWS CloudFormation stack

*Created by Ram Kandaswamy (AWS)*
Summary

If you use Amazon Web Services (AWS) resources, such as Amazon Simple Storage Service (Amazon S3) buckets, and want to use an infrastructure as code (IaC) approach, then you can import your resources into AWS CloudFormation and manage them as a stack.

This pattern provides steps to successfully import an S3 bucket as an AWS CloudFormation stack. By using this pattern's approach, you can avoid possible errors that might occur if you import your S3 bucket in a single action.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing S3 bucket and S3 bucket policy. For more information about this, see What S3 bucket policy should I use to comply with the AWS Config rule s3-bucket-ssl-requests-only in the AWS Knowledge Center.
- An existing AWS Key Management Service (AWS KMS) key and its alias. For more information about this, see Working with aliases in the AWS KMS documentation.
- The sample CloudFormation-template-S3-bucket AWS CloudFormation template (attached), downloaded to your local computer.

Architecture

The diagram shows the following workflow:

1. The user creates a JSON or YAML-formatted AWS CloudFormation template.
2. The template creates an AWS CloudFormation stack to import the S3 bucket.
3. The AWS CloudFormation stack manages the S3 bucket that you specified in the template.
**Technology stack**

- AWS CloudFormation
- AWS Identity and Access Management (IAM)
- AWS KMS
- Amazon S3

**Tools**

- **AWS CloudFormation** – AWS CloudFormation helps you to create and provision AWS infrastructure deployments predictably and repeatedly.
- **AWS Identity and Access Management (IAM)** – IAM is a web service for securely controlling access to AWS services.
- **AWS KMS** – AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the Internet.

**Epics**

**Import an S3 bucket with CMK-based encryption as an AWS CloudFormation stack**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a template to import the S3 bucket and CMK.</td>
<td>On your local computer, create a template to import your S3 bucket and CMK by using the following sample template:</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

```
AWSTemplateFormatVersion: 2010-09-09

Parameters:
  bucketName: Type: String

Resources:
  S3Bucket:
    Type: 'AWS::S3::Bucket'
    DeletionPolicy: Retain
    Properties:
      BucketName: !Ref bucketName
      BucketEncryption: ServerSideEncryptionConfiguration:
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerSideEncryptionByDefault:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSEAlgorithm:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'aws:kms'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KMSMasterKeyID: !GetAtt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KMSS3Encryption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Arn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KMSS3Encryption:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type: 'AWS::KMS::Key'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeletionPolicy: Retain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Properties:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enabled: true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KeyPolicy: !Sub</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>`-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`{</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Id&quot;: &quot;key-consolepolicy-3&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Version&quot;: &quot;2012-10-17&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Statement&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

### Create the stack.

1. Sign in to the AWS Management Console, open the AWS CloudFormation console, choose **View stack**, choose **Create stack**, and then choose **With existing resources (import resources)**.

2. Choose **Upload a template file** and then upload the template file that you created earlier.

3. Enter a name for your stack and configure the remaining options according to your requirements.

4. Choose **Create stack** and wait for the stack’s status to change to **IMPORT_COMPLETE**.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the KMS key alias. | 1. On the AWS CloudFormation console, choose **Stacks**, choose the name of the stack that you created earlier, choose the **Template** pane, and then choose **View in Designer**.  
2. Add the following snippet to the **Resource** section of your template, and then choose **Create stack** and complete the wizard:  

```yaml
KMSS3EncryptionAlias:
  Type: 'AWS::KMS::Alias'
  DeletionPolicy: Retain
  Properties:
    AliasName: alias/S3BucketKey
    TargetKeyId: !Ref KMSS3Encryption
```

For more information about this, see **AWS CloudFormation stack updates** in the AWS CloudFormation documentation. | AWS DevOps |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Update the stack to include the S3 bucket policy. | 1. On the AWS CloudFormation console, choose **Stacks**, choose the name of the stack that you created earlier, choose the **Template** pane, and then choose **View in Designer**.  
2. Add the following snippet to the **Resource** section of the template, and then choose **Create stack** and complete the wizard: | AWS DevOps       |

```json
S3BucketPolicy:
  Type: 'AWS::S3::BucketPolicy'
  Properties:
    Bucket: !Ref S3Bucket
    PolicyDocument: !Sub |
      |
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Condition&quot;: {</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>&quot;Bool&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;aws:SecureTransport&quot;: &quot;false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>This S3 bucket policy has a deny statement that restricts API calls that</td>
<td></td>
</tr>
<tr>
<td></td>
<td>are not secure.</td>
<td></td>
</tr>
<tr>
<td>Update the key policy.</td>
<td>1. On the AWS CloudFormation console, choose Stacks, choose the name of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stack that you created earlier, choose the Template pane, and then choose</td>
<td></td>
</tr>
<tr>
<td></td>
<td>View in Designer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Modify the template's KMS resource to include the key policy that allows</td>
<td></td>
</tr>
<tr>
<td></td>
<td>administrators to administer the CMK.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Choose Create stack, choose Next, and then complete the wizard according</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to your requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about this, see Using key policies in AWS KMS and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Allowing key administrators to administer the CMK in the AWS KMS documentation.</td>
<td></td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add resource-level tags. | 1. On the AWS CloudFormation console, choose **Stacks**, choose the name of the stack that you created earlier, choose the **Template** pane, and then choose **View in Designer**.  
2. Add the following snippet to the Amazon S3 resource **Properties** section of the template, and then choose **Create stack** and complete the wizard:  

Tags:  
- Key: createdBy  
  Value: Cloudformation | AWS DevOps |

### Related resources

- Bringing existing resources into AWS CloudFormation management
- AWS re:Invent 2017: Deep dive on AWS CloudFormation (video)

### Attachments

attachment.zip

### More patterns

- Automate adding or updating Windows registry entries using AWS Systems Manager (p. 687)
- Automate the creation of AppStream 2.0 resources using AWS CloudFormation (p. 554)
- Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline (p. 156)
- Automatically create an RFC in AMS using Python (p. 1554)
- Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows (p. 691)
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager (p. 1345)
- Cache secrets using AWS Lambda extensions (p. 1391)
- Check EC2 instances for mandatory tags at launch (p. 579)
- Create alarms for custom metrics using Amazon CloudWatch anomaly detection (p. 705)
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS (p. 166)
- Deploy a clustered application to Amazon ECS by using AWS Copilot (p. 196)
• Deploy a React-based single-page application to Amazon S3 and CloudFront (p. 1844)
• Deploy Lambda functions with container images (p. 191)
• Enforce automatic tagging of Amazon RDS databases at launch (p. 319)
• Help enforce DynamoDB tagging (p. 322)
• Identify and alert when Amazon Kinesis Data Firehose resources are not encrypted with an AWS KMS key (p. 684)
• Implement the serverless saga pattern by using AWS Step Functions (p. 1448)
• Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK (p. 713)
• Ingest and migrate EC2 Windows instances into an AWS Managed Services account (p. 984)
• Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions (p. 728)
• Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS (p. 816)
• Monitor Amazon ElastiCache clusters for at-rest encryption (p. 1696)
• Monitor Amazon EMR clusters for in-transit encryption at launch (p. 1692)
• Monitor ElastiCache clusters for security groups (p. 1566)
• Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization (p. 709)
• Set up CI/CD for AWS AppSync GraphQL API updates (p. 1482)
Containers & microservices

Topics

• Deploy Java microservices on Amazon EKS by using Amazon ECR and eksctl (p. 113)
• Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer (p. 117)
• Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer (p. 128)
• Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer (p. 138)
• Activate mTLS in AWS App Mesh using ACM Private CA on Amazon EKS (p. 143)
• Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch (p. 148)
• Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline (p. 156)
• Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS (p. 166)
• Deploy Java microservices on Amazon ECS using AWS Fargate (p. 169)
• Deploy Java microservices on Amazon ECS using Amazon ECR and AWS Fargate (p. 174)
• Deploy Java microservices on Amazon ECS using Amazon ECR and load balancing (p. 179)
• Deploy Kubernetes resources and packages using Amazon EKS and a Helm chart repository in Amazon S3 (p. 184)
• Deploy Lambda functions with container images (p. 191)
• Deploy a clustered application to Amazon ECS by using AWS Copilot (p. 196)
• Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer (p. 203)
• Generate a static outbound IP address using a Lambda function, Amazon VPC, and a serverless architecture (p. 211)
• Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet (p. 219)
• Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands (p. 223)
• Optimize AWS App2Container generated Docker images (p. 229)
• Replicate filtered Amazon ECR container images across accounts or Regions (p. 235)
• Rotate credentials without restarting containers (p. 246)
• Run Amazon ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere (p. 251)
• Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance (p. 258)
• Run stateful workloads with persistent data storage by using Amazon EFS on Amazon EKS with AWS Fargate (p. 265)
• More patterns (p. 276)

Deploy Java microservices on Amazon EKS by using Amazon ECR and eksctl

Created by Vijay Thompson (AWS)
Summary

This pattern describes how to deploy Java microservices as containerized applications in Amazon Elastic Kubernetes Service (Amazon EKS) by using the eksctl command line utility and Amazon Elastic Container Registry (Amazon ECR).

Prerequisites and limitations

Prerequisites

- An existing Java microservices on-premises application running on Docker.
- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows.
- The eksctl command line utility, installed and configured on macOS, Linux, or Windows. For more information, see Getting started with eksctl.
- The kubectl command line utility, installed and configured on macOS, Linux, or Windows. For more information, see Installing kubectl.

Architecture
• Amazon ECR
• Amazon EKS

**Tools**

• **AWS CLI** – AWS Command Line Interface (CLI) is a unified tool to manage your AWS services.
• **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images.
• **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane.
• **eksctl** – eksctl creates clusters on Amazon EKS.
• **kubectl** – kubectl runs commands against Kubernetes clusters.
• **Docker** – Docker helps you build, test, and deliver applications in packages called containers.

**Epics**

Create an Amazon EKS cluster by using eksctl

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon EKS cluster.</td>
<td>Use the eksctl “Create cluster” command. See the “Additional information” section for the command syntax. This command creates an Amazon EKS cluster in your default AWS Region and uses a single t2.small Amazon Elastic Compute Cloud (Amazon EC2) instance as the node. The process might take between 15 to 20 minutes to create the Amazon EKS cluster. After the cluster is created, the appropriate Kubernetes configuration will be added to your kubeconfig file.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Verify the Amazon EKS cluster.</td>
<td>After the cluster is created, Amazon EKS connects kubectl to the cluster. Run the &quot;kubectl get nodes&quot; command to verify that you have connected to the Amazon EKS cluster.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Create an Amazon ECR repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon ECR repository.</td>
<td>Open the Amazon ECR console. In the navigation pane, choose</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>
“Repositories,” and then choose “Create repository.” For help with this and other stories, see the “Related resources” section.

Upload the project.

Open the Amazon ECR repository, choose “View push commands,” and upload your project. After the upload is complete, copy the URL of the latest build in the Amazon ECR repository.

Deploy the Java microservices

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a YAML-formatted file.</td>
<td>Create a YAML file. Make sure you use the URL you copied earlier as the path of the image file for the Amazon ECR repository. For a sample YAML file, see the “Additional information” section.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Deploy the Java microservices on the Amazon EKS cluster.</td>
<td>After the YAML file is configured, run the &quot;kubectl apply -f &lt;filename&gt;.yaml&quot; command. This command deploys the YAML file on the Amazon EKS cluster.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Verify the status of the pods.</td>
<td>Verify the status of the pods by running the &quot;kubectl get pods&quot; command. After the application is up and running, the application will be in Ready status.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Test the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test and verify the application.</td>
<td>Test and verify the application by using the Classic Load Balancer that you configured in the YAML file.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Related resources

- Creating an Amazon ECR repository
- Pushing Docker images to Amazon ECR
Creating a cluster:

```bash
ekubectl create cluster \   --name <your-cluster-name> \   --version <version-number> \   --nodes=1 \   --node-type=t2.small
```

Sample YAML file:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: microservice-deployment
  labels:
    app: java-microservice
spec:
  replicas: 1
  selector:
    matchLabels:
      app: java-microservice
template:
  metadata:
    labels:
      app: java-microservice
  spec:
    containers:
    - name: java-microservice-container
      image: 123456789123.dkr.ecr.ap-south-1.amazonaws.com/java-microservice:latest
      ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: java-microservice-service
spec:
  # Creating a service of type load balancer. Load balancer gets created but takes time to reflect
  type: LoadBalancer
  selector:
    app: java-microservice
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer

*Created by Kirankumar Chandrashekar (AWS)*
Summary

This pattern describes how to privately host a Docker container application on Amazon Elastic Container Service (Amazon ECS) behind a Network Load Balancer, and access the application by using AWS PrivateLink. You can then use a private network to securely access services on the Amazon Web Services (AWS) Cloud. Amazon Relational Database Service (Amazon RDS) hosts the relational database for the application running on Amazon ECS with high availability (HA). Amazon Elastic File System (Amazon EFS) is used if the application requires persistent storage.

The Amazon ECS service running the Docker applications, with a Network Load Balancer at the front end, can be associated with a virtual private cloud (VPC) endpoint for access through AWS PrivateLink. This VPC endpoint service can then be shared with other VPCs by using their VPC endpoints.

You can also use AWS Fargate instead of an Amazon EC2 Auto Scaling group. For more information, see Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows
- Docker, installed and configured on Linux, macOS, or Windows
- An application running on Docker
Technology stack

- Amazon CloudWatch
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon EC2 Auto Scaling
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- Amazon RDS
- Amazon Simple Storage Service (Amazon S3)
- AWS Lambda
- AWS PrivateLink
- AWS Secrets Manager
- Application Load Balancer
- Network Load Balancer
- VPC

Automation and scale

- You can use AWS CloudFormation to create this pattern by using Infrastructure as Code.

Tools

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.
- Amazon EC2 Auto Scaling – Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage containers on a cluster.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a managed AWS container image registry service that is secure, scalable, and reliable.
- Amazon EFS – Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources.
- AWS Lambda – Lambda is a compute service for running code without provisioning or managing servers.
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.
- AWS Secrets Manager – Secrets Manager helps you replace hardcoded credentials in your code, including passwords, by providing an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon VPC – Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you’ve defined.
- Elastic Load Balancing – Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in multiple Availability Zones.
• **Docker** – Docker helps developers to pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

## Epics

### Create networking components

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC.</td>
<td>Create a VPC. Sign in to the AWS Management Console and open the Amazon VPC console. Choose “Launch VPC Wizard,” choose “VPC with a single public subnet,” and then choose “Select.” On the configuration page, enter a name for your VPC, and choose an appropriate CIDR block range. Configure the remaining options to your specifications. Choose “Create VPC,” and then choose “OK” to close the status window.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create the private subnets.</td>
<td>Create the private subnets. On the Amazon VPC console, choose “Subnets,” and then choose “Create subnets.” Specify the subnet details as necessary and choose “Create.” Repeat these steps to create three additional subnets. Two subnets are for Amazon ECS tasks, and two subnets are for Amazon RDS databases.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Create the load balancers

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Network Load Balancer.</td>
<td>Create a Network Load Balancer. Open the Amazon EC2 console and choose the AWS Region that contains your VPC. Under “Load balancing,” choose “Load balancers,” and choose “Create load balancer.” Choose “Network Load Balancer,” and choose “Create.” On the “Configure load balancer” page, configure your Network Load Balancer and listener. Important: Make sure you choose your Network Load Balancer’s scheme as “Internal.” Then choose the</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Create an Application Load Balancer.</strong></td>
<td>On the Amazon EC2 console, choose the same Region that contains your VPC. Under “Load balancing,” choose “Load balancers,” and choose “Create load balancer.” Choose “Application Load Balancer,” and choose “Create.” Configure your Application Load Balancer and its listener. Important: Make sure you choose your Application Load Balancer’s scheme as “Internal.” Then choose the applicable security settings, configure a security group and a target group. Choose &quot;Instance&quot; or &quot;IP&quot; as the &quot;Target type&quot; in the &quot;Configure routing&quot; section. Make sure you do not register a target. When you have configured all the settings, choose &quot;Next: Review,&quot; and then choose &quot;Create.&quot;</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create an Amazon EFS file system

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an Amazon EFS file system.</strong></td>
<td>Open the Amazon EFS console and choose “Create file system.” In the “Create file system” dialog box, enter a name for your file system, and choose your VPC. Choose “Create” to create the file system. Set up and configure your Amazon EFS file system.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td><strong>Mount targets for the subnets.</strong></td>
<td>Return to the Amazon EFS console, and choose “File systems.” The “File systems” page shows the Amazon EFS file systems in your account. Choose the file system that you created,</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Verify that the subnets are mounted as targets.</td>
<td>On the Amazon EFS console, choose “File systems.” Choose “Network” to display the list of existing mount targets. Make sure that these include the four subnets that you created.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Create an S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket.</td>
<td>Open the Amazon S3 console and create an S3 bucket to store your application’s static assets, if required.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Create a Secrets Manager secret

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS KMS key to encrypt the Secrets Manager secret.</td>
<td>Open the AWS Key Management Service (AWS KMS) console and create a KMS key.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create a Secrets Manager secret to store the Amazon RDS password.</td>
<td>Open the AWS Secrets Manager console and create a new secret by choosing “Store a new secret.” Choose the KMS key that you created, and store your new secret.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Create an Amazon RDS instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a DB subnet group.</td>
<td>Open the Amazon RDS console and choose “Subnet groups.” Choose “Create DB subnet group,” and enter a name and description for your DB subnet group. Choose the VPC that you created earlier, and choose the Availability Zones and subnets. Then choose “Create.”</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
### Create an Amazon RDS instance

**Task**: Create an Amazon RDS instance.

**Description**: Create and configure an Amazon RDS instance within the private subnets. Make sure that "Multi-AZ" is turned on for HA.

**Skills required**: Cloud Administrator

### Load data to the Amazon RDS instance

**Task**: Load data to the Amazon RDS instance.

**Description**: Load the relational data required by your application into your Amazon RDS instance. This process will vary depending on your application's needs, as well as how your database schema is defined and designed.

**Skills required**: Cloud Administrator, DBA

### Create the Amazon ECS components

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an ECS cluster.</td>
<td>Open the Amazon ECS console and choose “Clusters.” Then choose &quot;Create clusters,&quot; and set up an ECS cluster according to your required specifications.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create the Docker images.</td>
<td>Create the Docker images by following the instructions in the “Related resources” section.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create Amazon ECR repositories.</td>
<td>On the Amazon ECR console, choose “Repositories.” Choose “Create repository” and enter a unique name for your repository. Configure the repository according to your specifications, including AWS KMS encryption if required.</td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Authenticate your Docker client for the Amazon ECR repository.</td>
<td>To authenticate your Docker client for the Amazon ECR repository, run the “aws ecr get-login-password” command in the AWS CLI.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Push the Docker images to the Amazon ECR repository.</td>
<td>Identify the Docker image you want to push and run the “docker images” command in the AWS CLI. Tag your image with the Amazon ECR registry, repository, and optional image tag name combination. Then push the Docker image by running the “docker push” command. Repeat this step for all required images.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Create an Amazon ECS task definition.</td>
<td>A task definition is required to run Docker containers in Amazon ECS. Return to the Amazon ECS console and choose “Task definitions,” and then choose “Create new task definition.” On the “Select compatibilities” page, select the launch type that your task should use and choose “Next step.” For help with setting up your task definition, see “Creating a task definition” in the “Related resources” section. Important: Make sure you provide the Docker images that you pushed to Amazon ECR.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create an Amazon ECS service.</td>
<td>Create an Amazon ECS service by using the ECS cluster you created earlier. Make sure you choose Amazon EC2 as the launch type, and choose the task definition created in the previous step, as well as the target group of the Application Load Balancer.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create an Amazon EC2 Auto Scaling group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a launch configuration.</td>
<td>Open the Amazon EC2 console and create a launch configuration. Make sure that the user data has the code to allow the EC2 instances to join the desired ECS cluster. For an example of the code required, see the “Related resources” section.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create an Amazon EC2 Auto Scaling group.</td>
<td>Return to the Amazon EC2 console and under “Auto Scaling,” choose “Auto Scaling groups.” Set up an Amazon EC2 Auto Scaling group. Make sure you choose the private subnets and launch configuration that you created earlier.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
Set up AWS PrivateLink

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the AWS PrivateLink endpoint.</td>
<td>On the Amazon VPC console, create an AWS PrivateLink endpoint. Associate this endpoint with the Network Load Balancer, which makes the application hosted on Amazon ECS available privately to customers. For more information, see the “Related resources” section.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create a VPC endpoint

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC endpoint.</td>
<td>Create a VPC endpoint for the AWS PrivateLink endpoint that you created earlier. The VPC endpoint Fully Qualified Domain Name (FQDN) will point to the AWS PrivateLink endpoint FQDN. This creates an elastic network interface to the VPC endpoint service that the DNS endpoints can access.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create the Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Lambda function.</td>
<td>On the AWS Lambda console, create a Lambda function to update the Application Load Balancer IP addresses as targets for the Network Load Balancer. For more information on this, see the &quot;Using static IP addresses for Application Load Balancers&quot; blog post in the “Related resources” section.</td>
<td>Application Developer</td>
</tr>
</tbody>
</table>

Related resources

Create the load balancers:
- Create a Network Load Balancer
• Create an Application Load Balancer

Create an Amazon EFS file system:
• Create an Amazon EFS file system
• Create mount targets in Amazon EFS

Create an S3 bucket:
• Create an S3 bucket

Create a Secrets Manager secret:
• Create keys in AWS KMS
• Create a secret in AWS Secrets Manager

Create an Amazon RDS instance:
• Create an Amazon RDS DB instance

Create the Amazon ECS components:
• Create an Amazon ECS cluster
• Create a Docker image
• Create an Amazon ECR repository
• Authenticate Docker with Amazon ECR repository
• Push an image to an Amazon ECR repository
• Create Amazon ECS task definition
• Create an Amazon ECS service

Create an Amazon EC2 Auto Scaling group:
• Create a launch configuration
• Create an Auto Scaling group using a launch configuration
• Bootstrap container instances with Amazon EC2 user data

Set up AWS PrivateLink:
• VPC endpoint services (AWS PrivateLink)

Create a VPC endpoint:
• Interface VPC endpoints (AWS PrivateLink)
Create the Lambda function:

- Create a Lambda function

Other resources:

- Using static IP addresses for Application Load Balancers
- Securely accessing services over AWS PrivateLink

Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer

Created by Kirankumar Chandrashekar (AWS)

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Containers &amp; microservices; Networking; Security, identity, compliance; Websites &amp; web apps</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services: AWS EC2 Container Registry; Amazon ECS; Amazon EFS; Amazon RDS; Amazon VPC; Elastic Load Balancing; AWS Lambda</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to privately host a Docker container application on the Amazon Web Services (AWS) Cloud by using Amazon Elastic Container Service (Amazon ECS) with an AWS Fargate launch type, behind a Network Load Balancer, and access the application by using AWS PrivateLink. Amazon Relational Database Service (Amazon RDS) hosts the relational database for the application running on Amazon ECS with high availability (HA). You can use Amazon Elastic File System (Amazon EFS) if the application requires persistent storage.

This pattern uses a Fargate launch type for the Amazon ECS service running the Docker applications, with a Network Load Balancer at the front end. It can then be associated with a virtual private cloud (VPC) endpoint for access through AWS PrivateLink. This VPC endpoint service can then be shared with other VPCs by using their VPC endpoints.

You can use Fargate with Amazon ECS to run containers without having to manage servers or clusters of Amazon Elastic Compute Cloud (Amazon EC2) instances. You can also use an Amazon EC2 Auto Scaling group instead of Fargate. For more information, see Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer.

Prerequisites and limitations

Prerequisites
• An active AWS account
• AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows
• Docker, installed and configured on Linux, macOS, or Windows
• An application running on Docker
Architecture

AWS Prescriptive Guidance Patterns

Architecture

Diagram showing the architecture with components such as AWS Lambda function, Network Load Balancer, Application Load Balancer, Amazon ECS, Amazon RDS, and others.
Technology stack

- Amazon CloudWatch
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- Amazon EFS
- Amazon RDS
- Amazon Simple Storage Service (Amazon S3)
- AWS Fargate
- AWS Lambda
- AWS PrivateLink
- AWS Secrets Manager
- Application Load Balancer
- Network Load Balancer
- VPC

Automation and scale

- You can use AWS CloudFormation to create this pattern by using Infrastructure as Code.

Tools

- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage containers on a cluster.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a managed AWS container image registry service that is secure, scalable, and reliable.
- Amazon EFS – Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources.
- AWS Fargate – AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances.
- AWS Lambda – Lambda is a compute service that lets you run code without provisioning or managing servers.
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.
- AWS Secrets Manager – Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon VPC – Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you’ve defined.
- Elastic Load Balancing – Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in multiple Availability Zones.
- Docker – Docker helps developers to easily pack, ship, and run any application as a lightweight, portable, and self-sufficient container.
# Epics

## Create networking components

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC.</td>
<td>Sign in to the AWS Management Console and open the Amazon VPC console. Choose “Launch VPC Wizard,” choose “VPC with a single public subnet,” and then choose “Select.” On the configuration page, enter a name for your VPC, and choose an appropriate CIDR block range. Configure the remaining options to your specifications. Choose “Create VPC,” and then choose “OK” to close the status window.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create the private subnets.</td>
<td>On the Amazon VPC console, choose “Subnets,” and then choose “Create subnets.” Specify the subnet details as necessary and choose “Create.” Repeat these steps to create three additional subnets. Two subnets are for Amazon ECS tasks, and two subnets are for Amazon RDS databases.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

## Create the load balancers

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Network Load Balancer.</td>
<td>Open the Amazon EC2 console and choose the AWS Region that contains your VPC. Under “Load balancing,” choose “Load balancers,” and choose “Create load balancer.” Choose “Network Load Balancer,” and choose “Create.” On the “Configure load balancer” page, configure your Network Load Balancer and listener. Important: Make sure you choose your Network Load Balancer’s scheme as “Internal.” Then choose the applicable security settings, configure a security group and a target group. Choose “IP” as the “Target type” in the “Configure routing”</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
### Create an Application Load Balancer.

**Task** | **Description** | **Skills required**  
---|---|---  
Create an Application Load Balancer. | On the Amazon EC2 console, choose the same Region that contains your VPC. Under "Load balancing," choose "Load balancers," and choose "Create load balancer." Choose "Application Load Balancer," and choose "Create." Configure your Application Load Balancer and its listener. Important: Make sure you choose your Application Load Balancer’s scheme as "Internal." Then choose the applicable security settings, configure a security group and a target group. Choose "IP" as the "Target type" in the "Configure routing" section. Make sure you do not register a target. When you have configured all the settings, choose "Next: Review," and then choose "Create." | Cloud Administrator  

### Create an Amazon EFS file system

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an Amazon EFS file system. | Open the Amazon EFS console and choose “Create file system.” In the “Create file system” dialog box, enter a name for your file system, and choose your VPC. Choose “Create” to create the file system. Set up and configure your Amazon EFS file system. | Cloud Administrator  

Mount targets for the subnets. | Return to the Amazon EFS console and choose “File systems.” The “File systems” page shows the Amazon EFS file systems in your account. Choose the file system that you created, and choose “Manage” to display the “Availability Zone.” To add a mount target, choose “Add” | Cloud Administrator  

133
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mount target,&quot; and add the four private subnets that you created.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Verify that the subnets are mounted as targets.</td>
<td>On the Amazon EFS console, choose “File systems.” Choose “Network” to display the list of existing mount targets. Make sure that these include the four subnets that you created.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create an S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket.</td>
<td>Open the Amazon S3 console and create an S3 bucket to store your application's static assets, if required.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create a Secrets Manager secret

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS KMS key to encrypt the Secrets Manager secret.</td>
<td>Open the AWS Key Management Service (AWS KMS) console and create a KMS key.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create a Secrets Manager secret to store the Amazon RDS password.</td>
<td>Open the AWS Secrets Manager console and create a new secret by choosing “Store a new secret.” Choose the KMS key that you created, and store your new secret.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create an Amazon RDS instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a DB subnet group.</td>
<td>Open the Amazon RDS console and choose “Subnet groups.” Choose “Create DB subnet group,” and enter a name and description for your DB subnet group. Choose the VPC that you created earlier, and choose the Availability Zones and subnets. Then choose “Create.”</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create an Amazon RDS instance.</td>
<td>Create and configure an Amazon RDS instance within the private Cloud Administrator.</td>
<td></td>
</tr>
</tbody>
</table>
### Create the Amazon ECS components

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an ECS cluster.</strong></td>
<td>Open the Amazon ECS console and choose “Clusters.” Then choose “Create clusters,” and set up an ECS cluster according to your required specifications.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td><strong>Create the Docker images.</strong></td>
<td>Create the Docker images by following the instructions in the “Related resources” section.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td><strong>Create an Amazon ECR repository.</strong></td>
<td>Open the Amazon ECR console and choose “Repositories.” Choose “Create repository” and enter a unique name for your repository. Configure the repository according to your specifications, including AWS KMS encryption if required.</td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td><strong>Push the Docker images to the Amazon ECR repository.</strong></td>
<td>Identify the Docker image you want to push and run the “docker images” command in AWS CLI. Tag your image with the Amazon ECR registry, repository, and optional image tag name combination. Then push the Docker image by running the “docker push” command. Repeat this step for all required images.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td><strong>Create an Amazon ECS task definition.</strong></td>
<td>A task definition is required to run Docker containers in Amazon ECS. Return to the Amazon ECS console and choose “Task definitions,” and then choose “Create new task definition.” On the “Select compatibilities” page, select</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
Create an ECS service and choose Fargate as the launch type.

Create an Amazon ECS service by using the ECS cluster you created earlier. Make sure you choose “Fargate” as the launch type, then choose the task definition created in the previous step, and choose the target group of the Application Load Balancer.

Skills required

Cloud Administrator

### Set up AWS PrivateLink

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the AWS PrivateLink endpoint.</td>
<td>Open the Amazon VPC console and create an AWS PrivateLink endpoint. Associate this endpoint with the Network Load Balancer, which makes the application hosted on Amazon ECS available privately to customers. For more information, see the “Related resources” section.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Create a VPC endpoint

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC endpoint.</td>
<td>Create a VPC endpoint for the AWS PrivateLink endpoint that you created earlier. The VPC endpoint Fully Qualified Domain Name (FQDN) will point to the AWS PrivateLink endpoint FQDN. This creates an elastic network interface to the VPC endpoint service that the Domain Name Service endpoints can access.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
Create the Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Lambda function.</td>
<td>Open the Lambda console and create a Lambda function to update the Application Load Balancer IP addresses as targets for the Network Load Balancer. For more information on this, see the &quot;Using static IP addresses for Application Load Balancers&quot; blog post in the “Related resources” section.</td>
<td>Application Developer</td>
</tr>
</tbody>
</table>

Related resources

Create the load balancers:

- Create a Network Load Balancer
- Create an Application Load Balancer

Create an Amazon EFS file system:

- Create an Amazon EFS file system
- Create mount targets in Amazon EFS

Create an S3 bucket:

- Create an S3 bucket

Create a Secrets Manager secret:

- Create keys in AWS KMS
- Create a secret in AWS Secrets Manager

Create an Amazon RDS instance:

- Create an Amazon RDS DB instance

Create the Amazon ECS components:

- Create an Amazon ECS cluster
- Create a Docker image
- Create an Amazon ECR repository
- Authenticate Docker with Amazon ECR repository
- Push an image to an Amazon ECR repository
• Create Amazon ECS task definition
• Create an Amazon ECS service

Set up AWS PrivateLink:
• VPC endpoint services (AWS PrivateLink)

Create a VPC endpoint:
• Interface VPC endpoints (AWS PrivateLink)

Create the Lambda function:
• Create a Lambda function

Other resources:
• Using static IP addresses for Application Load Balancers
• Securely accessing services over AWS PrivateLink

Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer

Created by Kirankumar Chandrashekar (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Containers &amp; microservices; DevOps; Modernization; Security, identity, compliance</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon EKS; Amazon VPC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to privately host a Docker container application on Amazon Elastic Kubernetes Service (Amazon EKS) behind a Network Load Balancer, and access the application by using AWS PrivateLink. You can then use a private network to securely access services on the Amazon Web Services (AWS) Cloud.

The Amazon EKS cluster running the Docker applications, with a Network Load Balancer at the front end, can be associated with a virtual private cloud (VPC) endpoint for access through AWS PrivateLink. This VPC endpoint service can then be shared with other VPCs by using their VPC endpoints.
The setup described by this pattern is a secure way to share application access among VPCs and AWS accounts. It requires no special connectivity or routing configurations, because the connection between the consumer and provider accounts is on the global AWS backbone and doesn't traverse the public internet.

Prerequisites and limitations

Prerequisites

- Docker, installed and configured on Linux, macOS, or Windows.
- An application running on Docker.
- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows.
- An existing Amazon EKS cluster with tagged private subnets and configured to host applications. For more information, see Subnet tagging in the Amazon EKS documentation.
- Kubectl, installed and configured to access resources on your Amazon EKS cluster. For more information, see Installing kubectl in the Amazon EKS documentation.

Architecture

Technology stack

- Amazon EKS
- AWS PrivateLink
- Network Load Balancer

Automation and scale
• Kubernetes manifests can be tracked and managed on a Git-based repository (for example, on AWS CodeCommit), and deployed by using continuous integration and continuous delivery (CI/CD) in AWS CodePipeline.

• You can use AWS CloudFormation to create this pattern by using infrastructure as code (IaC).

**Tools**

• **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.

• **Elastic Load Balancing** – Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses, in one or more Availability Zones.

• **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

• **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined.

• **Kubectl** – Kubectl is a command line utility for running commands against Kubernetes clusters.

**Epics**

**Deploy the Kubernetes deployment and service manifest files**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Kubernetes deployment manifest file.</td>
<td>Create a deployment manifest file by modifying the following sample file according to your requirements.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/z9d2n7e1/nginx:1.19.5
          ports: - name: http
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the Kubernetes deployment manifest file.                    | Run the following command to apply the deployment manifest file to your Amazon EKS cluster:  
  ```bash
  kubectl apply -f <your_deployment_file_name>
  ``` | DevOps engineer                                                                                                                                  |
| Create the Kubernetes service manifest file.                       | Create a service manifest file by modifying the following sample file according to your requirements.                                                                                                      | DevOps engineer |
|                                                                     | ```yaml
  apiVersion: v1
class: Service
  metadata:
   name: sample-service
  annotations:
   service.beta.kubernetes.io/aws-load-balancer-type: nlb
   service.beta.kubernetes.io/aws-load-balancer-internal: "true"
  spec:
   ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      type: LoadBalancer
    selector:
     app: nginx
  ``` |  |
| Important: Make sure that you included the following annotations to define an internal Network Load Balancer: | ```yaml
  service.beta.kubernetes.io/aws-load-balancer-type: nlb
  service.beta.kubernetes.io/aws-load-balancer-internal: "true"
  ``` |  |
### Create the endpoints

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Deploy the Kubernetes service manifest file.** | Run the following command to apply the service manifest file to your Amazon EKS cluster:  
```bash
kubectl apply -f <your_service_file_name>
```                                                                                                                                   | DevOps engineer         |
| **Record the Network Load Balancer’s name.** | Run the following command to retrieve the name of the Network Load Balancer:  
```bash
kubectl get svc sample-service -o wide
```  
Record the Network Load Balancer’s name, which is required to create an AWS PrivateLink endpoint.                                                                 | DevOps engineer         |
| **Create an AWS PrivateLink endpoint.** | Sign in to the AWS Management Console, open the Amazon VPC console, and then create an AWS PrivateLink endpoint. Associate this endpoint with the Network Load Balancer, this makes the application privately available to customers. For more information, see VPC endpoint services (AWS PrivateLink) in the Amazon VPC documentation.  
**Important:** If the consumer account requires access to the application, the consumer account’s AWS account ID must be added to the allowed principals list for the AWS PrivateLink endpoint configuration. For more information, see Adding and removing permissions for your endpoint service in the Amazon VPC documentation. | Cloud administrator     |
| **Create a VPC endpoint.** | On the Amazon VPC console, choose **Endpoint Services**, and then choose **Create Endpoint Service**. Create a VPC endpoint for the AWS PrivateLink endpoint.                                                                 | Cloud administrator     |
The VPC endpoint’s fully qualified domain name (FQDN) points to the FQDN for the AWS PrivateLink endpoint. This creates an elastic network interface to the VPC endpoint service that the DNS endpoints can access.

### Related resources

- Using the official NGINX Docker image
- Network load balancing on Amazon EKS
- Creating VPC endpoint services (AWS PrivateLink)
- Adding and removing permissions for your endpoint service

## Activate mTLS in AWS App Mesh using ACM Private CA on Amazon EKS

*Created by Omar Kahil (AWS), Emmanuel Saliu (AWS), and Muhammad Shahzad (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Containers &amp; microservices</td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS App Mesh; Amazon EKS; AWS Certificate Manager</td>
</tr>
</tbody>
</table>

### Summary

This pattern shows how to implement Mutual Transport Layer Security (mTLS) on Amazon Web Services (AWS) using certificates from AWS Certificate Manager Private Certificate Authority (ACM Private CA) in AWS App Mesh. It uses the Envoy secret discovery service (SDS) API through the Secure Production Identity Framework for Everyone (SPIFFE). SPIFFE is a Cloud Native Computing Foundation (CNCF) open-source project with wide community support that provides fine-grained and dynamic workload identity management. To implement SPIFFE standards, use the SPIRE SPIFFE runtime environment.

Using mTLS in App Mesh offers two-way peer authentication, because it adds a layer of security over TLS and allows services in the mesh to verify the client that's making the connection. The client in the client-server relationship also provides an X.509 certificate during the session negotiation process. The server uses this certificate to identify and authenticate the client. This helps to verify if the certificate is issued by a trusted certificate authority (CA) and if the certificate is a valid one.

### Prerequisites and limitations

#### Prerequisites

- An Amazon Elastic Kubernetes Service (Amazon EKS) cluster with self-managed or managed node groups
- App Mesh controller deployed on the cluster with SDS activated
• A private certificate from AWS Certificate Manager (ACM) that is issued by an AWS Certificate Manager Private Certificate Authority (ACM Private CA)

Limitations

• SPIRE cannot be installed on AWS Fargate because the SPIRE Agent must be run as a Kubernetes DaemonSet.

Product versions

• AWS App Mesh Controller chart 1.3.0 or later

Architecture

The following diagram shows the EKS cluster with App Mesh in the VPC. The SPIRE server in one worker node communicates with the SPIRE Agents in other worker nodes, and with ACM Private CA. Envoy is used for mTLS communication between the SPIRE Agent worker nodes.

1. Certificate is issued.
2. Request cert signing and certificate.

Tools

AWS services

• ACM Private CA – AWS Certificate Manager Private Certificate Authority (ACM Private CA) enables creation of private certificate authority (CA) hierarchies, including root and subordinate CAs, without the investment and maintenance costs of operating an on-premises CA.
• **AWS App Mesh** – AWS App Mesh is a service mesh that makes it easier to monitor and control services. App Mesh standardizes how your services communicate, giving you consistent visibility and network traffic controls for every service in an application.

• **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

**Other tools**

• **Helm** – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster. This pattern uses Helm to deploy AWS App Mesh Controller.

• **AWS App Mesh Controller chart** – AWS App Mesh Controller chart is used by this pattern to enable AWS App Mesh on Amazon EKS.

**Epics**

**Set up the environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up App Mesh with Amazon EKS.</td>
<td>Follow base deployment steps that are provided in the repository.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Install SPIRE.</td>
<td>Install SPIRE on the EKS cluster by using spire_setup.yaml.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Install the ACM Private CA certificate.</td>
<td>Create and install a certificate for your private root CA by following the instructions in the AWS documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Grant permissions to the cluster node instance role.</td>
<td>To attach policies to the cluster node instance role, use the code that's in the Additional information section.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Add the SPIRE plugin for ACM Private CA.</td>
<td>To add the plugin to the SPIRE server configuration, use the code that's in the Additional information section. Replace the certificate_authority_arn Amazon Resource Name (ARN) to your private CA ARN. The signing algorithm used must be the same as the signing algorithm on the PCA. Replace your_region with your AWS Region. For more information about the plugin, see Server plugin: UpstreamAuthority &quot;aws_pca&quot;.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
## Update bundle.cert.

After you create the SPIRE server, a spire-bundle.yaml file will be created. Change the bundle.crt value in the spire-bundle.yaml file from the private CA to the public certificate.

**Skills required**

DevOps engineer

---

### Deploy and register the workloads

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register node and workload entries with SPIRE.</td>
<td>To register node and workload (services) with SPIRE Server, use the code in the repository.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Create a mesh in App Mesh with mTLS activated.</td>
<td>Create a new mesh in App Mesh with all the components for your microservices application (for example, virtual service, virtual router, and virtual nodes).</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Inspect the registered entries.</td>
<td>You can inspect the registered entries for your nodes and workloads by running the following command.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

```bash
cubectl exec -n spire spire-server-0 -- /opt/spire/bin/spire-server entry show
```

This will show the entries for the SPIRE Agents.

### Verify mTLS traffic

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Verify mTLS traffic. | 1. From the frontend service, send an HTTP header to the backend service, and verify a successful response with the services that are registered in SPIRE.  
2. For mutual TLS authentication, you can inspect the ssl.handshake statistic by running the following command. | DevOps engineer |
## Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>`kubectl exec -it $POD -n $NAMESPACE -- curl <a href="http://localhost:9901/stats">http://localhost:9901/stats</a></td>
<td>grep ssl.handshake`</td>
</tr>
<tr>
<td>After running the previous command, you should see the listener <code>ssl.handshake</code> count, which will look similar to the following example:</td>
<td></td>
</tr>
<tr>
<td><code>listener.0.0.0.0_15000.ssl.handshake: 2</code></td>
<td></td>
</tr>
<tr>
<td>Verify that certificates are being issued from ACM Private CA.</td>
<td>You can check that the plugins have been configured correctly and certificates are being issued from your upstream private CA by viewing the logs in your SPIRE server. Run the following command.</td>
</tr>
<tr>
<td><code>kubectl logs spire-server-0 -n spire</code></td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Then view the logs that are produced. This code assumes that your server is named <code>spire-server-0</code> and is hosted in your <code>spire</code> namespace. You should see successful loading of the plugins and a connection being made to your upstream private CA.</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

- Using mTLS with SPIFFE/SPHERE in AWS App Mesh on Amazon EKS
- Enabling mTLS in AWS App Mesh using SPIFFE/SPHERE in a multi-account Amazon EKS environment
- Walkthrough used in this pattern
- Server plugin: UpstreamAuthority "aws_pca"
- Quickstart for Kubernetes

### Additional information

Attach permissions to the cluster node instance role

```
Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch

*Created by Kirankumar Chandrashekar (AWS)*

<table>
<thead>
<tr>
<th>Environment</th>
<th>Technologies</th>
<th>Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoC or pilot</td>
<td>Containers &amp; microservices; Databases; DevOps</td>
<td>All other workloads</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AWS services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon RDS; AWS Batch; Amazon CloudWatch; AWS Lambda; Amazon S3</td>
</tr>
</tbody>
</table>

**Summary**

Backing up your PostgreSQL databases is an important task and can typically be completed with the `pg_dump` utility, which uses the COPY command to create a schema and data dump of a PostgreSQL database. However, this process can become repetitive if you require regular backups for multiple PostgreSQL databases. This pattern describes how to automate regular backups for Amazon Relational Database Service (Amazon RDS) for PostgreSQL DB instances.
A time-based Amazon CloudWatch Events event initiates an AWS Lambda function that searches for specific backup tags applied to the metadata of the PostgreSQL DB instances on Amazon RDS. If the PostgreSQL DB instances have the **bkp:AutomatedDBDump = Active** tag and other required backup tags, then the Lambda function submits individual jobs for each database backup to AWS Batch.

AWS Batch processes these jobs and uploads the backup data to an Amazon Simple Storage Service (Amazon S3) bucket. This pattern uses a Dockerfile and an entrypoint.sh file to build a Docker container image that is used to make backups in the AWS Batch job. After the backup process is complete, AWS Batch records the backup details to an inventory table on Amazon DynamoDB. As an additional safeguard, a CloudWatch Events event initiates an Amazon Simple Notification Service (Amazon SNS) notification if a job fails in AWS Batch.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account.
- An existing managed or unmanaged compute environment. For more information, see Managed and unmanaged compute environments in the AWS Batch documentation.
- **AWS Command Line Interface (CLI)** version 2 Docker image, installed and configured.
- Existing Amazon RDS for PostgreSQL DB instances.
- An existing S3 bucket.
- **Docker**, installed and configured on Linux, macOS, or Windows.
- Familiarity with coding in Lambda.

**Architecture**

**Technology stack**

- Amazon CloudWatch Events
- Amazon DynamoDB
Your PostgreSQL DB instances on Amazon RDS must have tags applied to their metadata. The Lambda function searches for tags to identify DB instances that should be backed up, and the following tags are typically used.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bkp:AutomatedDBDump = Active</td>
<td>Identifies an Amazon RDS DB instance as a candidate for backups.</td>
</tr>
<tr>
<td>bkp:AutomatedBackupSecret = &lt;secret_name&gt;</td>
<td>Identifies the Secrets Manager secret that contains the Amazon RDS login credentials.</td>
</tr>
<tr>
<td>bkp:AutomatedDBDumpS3Bucket = &lt;s3_bucket_name&gt;</td>
<td>Identifies the S3 bucket to send backups to.</td>
</tr>
<tr>
<td>bkp:AutomatedDBDumpFrequency</td>
<td>Identify the frequency and times when databases should be backed up.</td>
</tr>
</tbody>
</table>
Epics

Create an inventory table in DynamoDB

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a table in DynamoDB.</td>
<td>Sign in to the AWS Management Console, open the Amazon DynamoDB console, and create a table. For help with this and other stories, see the “Related resources” section.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Confirm that the table was created.</td>
<td>Run the “aws dynamodb describe-table --table-name &lt;table-name&gt;</td>
<td>grep TableStatus” command. If the table exists, the command will return the ““TableStatus”: &quot;ACTIVE”,” result.</td>
</tr>
</tbody>
</table>

Create an SNS topic for failed job events in AWS Batch

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SNS topic.</td>
<td>Open the Amazon SNS console, choose “Topics,” and create an SNS topic with the name “JobFailedAlert.” Subscribe an active email address to the topic, and check your email inbox to confirm the SNS subscription email from AWS Notifications.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create a failed job event rule for AWS Batch.</td>
<td>Open the Amazon CloudWatch console, choose “Events,” and then choose “Create rule.” Choose “Show advanced options,” and choose “Edit.” For “Build a pattern that selects events for processing by your targets,” replace the existing text with the “Failed job event” code from the “Additional information” section. This code defines a CloudWatch Events rule that initiates when AWS Batch has a “Failed” event.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
## Add event rule target.

In “Targets,” choose “Add targets,” and choose the “JobFailedAlert” SNS topic. Configure the remaining details and create the Cloudwatch Events rule.

**Skills required:** Cloud Administrator

## Build a Docker image and push it to an Amazon ECR repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon ECR repository.</td>
<td>Open the Amazon ECR console and choose the AWS Region in which you want to create your repository. Choose “Repositories,” and then choose “Create repository.” Configure the repository according to your requirements.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Write a Dockerfile.</td>
<td>Sign in to Docker and use the “Sample Dockerfile” and “Sample entrypoint.sh file” from the “Additional information” section to build a Dockerfile.</td>
<td>DevOps Engineer</td>
</tr>
<tr>
<td>Create a Docker image and push it to the Amazon ECR repository.</td>
<td>Build the Dockerfile into a Docker image and push it to the Amazon ECR repository. For help with this story, see the “Related resources” section.</td>
<td>DevOps Engineer</td>
</tr>
</tbody>
</table>

## Create the AWS Batch components

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS Batch job definition.</td>
<td>Open the AWS Batch console and create a job definition that includes the Amazon ECR repository’s Uniform Resource Identifier (URI) as the property “Image.”</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Configure the AWS Batch job queue.</td>
<td>On the AWS Batch console, choose “Job queues,” and then choose “Create queue.” Create a job queue that will store jobs until AWS Batch runs them on the resources within your compute environment. Important: Make sure you write logic for AWS Batch to</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
Create and schedule a Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Lambda function to search for tags.</td>
<td>Create a Lambda function that searches for tags on your PostgreSQL DB instances and identifies backup candidates. Make sure your Lambda function can identify the “bkp:AutomatedDBDump = Active” tag and all other required tags. Important: The Lambda function must also be able to add jobs to the AWS Batch job queue.</td>
<td>DevOps Engineer</td>
</tr>
<tr>
<td>Create a time-based CloudWatch Events event.</td>
<td>Open the Amazon CloudWatch console and create a CloudWatch Events event that uses a cron expression to run your Lambda function on a regular schedule. Important: All scheduled events use the UTC time zone.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Test the backup automation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon KMS key.</td>
<td>Open the Amazon KMS console and create a KMS key that can be used to encrypt the Amazon RDS credentials stored in AWS Secrets Manager.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create an AWS Secrets Manager secret.</td>
<td>Open the AWS Secrets Manager console and store your Amazon RDS for PostgreSQL database credentials as a secret.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Add the required tags to the PostgreSQL DB instances.</td>
<td>Open the Amazon RDS console and add tags to the PostgreSQL DB instances that you want to automatically back up. You can use the tags from the table in the “Tools” section. If you require backups from multiple PostgreSQL databases.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the backup automation.</td>
<td>To verify the backup automation, you can either invoke the Lambda function or wait for the backup schedule to begin. After the backup process is complete, check that the DynamoDB inventory table has a valid backup entry for your PostgreSQL DB instances. If they match, then the backup automation process is successful.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Related resources

**Create an inventory table in DynamoDB**
- Create an Amazon DynamoDB table

**Create an SNS topic for failed job events in AWS Batch**
- Create an Amazon SNS topic
- Send SNS alerts for failed job events in AWS Batch

**Build a Docker image and push it to an Amazon ECR repository**
- Create an Amazon ECR repository
- Write a Dockerfile, create a Docker image, and push it to Amazon ECR

**Create the AWS Batch components**
- Create an AWS Batch job definition
- Configure your compute environment and AWS Batch job queue
- Create a job queue in AWS Batch

**Create a Lambda function**
• Create a Lambda function and write code
• Use Lambda with DynamoDB

Create a CloudWatch Events event
• Create a time-based CloudWatch Events event
• Use cron expressions in CloudWatch Events

Test the backup automation
• Create an Amazon KMS key
• Create a Secrets Manager secret
• Add tags to an Amazon RDS instance

Additional information

Failed job event:

```
{
  "detail-type": [
    "Batch Job State Change"
  ],
  "source": [
    "aws.batch"
  ],
  "detail": {
    "status": [
      "FAILED"
    ]
  }
}
```

Sample Dockerfile:

```
FROM alpine:latest
RUN apk --update add py-pip postgresql-client jq bash && 
    pip install awscli && 
    rm -rf /var/cache/apk/*
ADD entrypoint.sh /usr/bin/
RUN chmod +x /usr/bin/entrypoint.sh
ENTRYPOINT ["entrypoint.sh"]
```

Sample entrypoint.sh file:

```
#!/bin/bash
set -e
DATETIME=`date +"%Y-%m-%d_%H_%M"`
FILENAME=RDS_PostGres_dump_${RDS_INSTANCE_NAME}
FILE=${FILENAME}_${DATETIME}
aws configure --profile new-profile set role_arn arn:aws:iam::${TargetAccountId}:role/
  @${TargetAccountRoleName}
```
Automatically build and deploy a Java application to Amazon EKS

Created by MAHESH RAGHUNANDANAN (AWS) and Jomcy Pappachen (AWS)
Summary

This pattern describes how to create a continuous integration and continuous delivery (CI/CD) pipeline that automatically builds and deploys a Java application to an Amazon Elastic Kubernetes Service (Amazon EKS) cluster on the Amazon Web Services (AWS) Cloud. This pattern uses a greeting application developed with a Spring Boot Java framework and that uses Apache Maven.

You can use this pattern's approach to build the code for a Java application, package the application artifacts as a Docker image, security scan the image, and upload the image as a workload container on Amazon EKS. This pattern's approach is useful if you want to migrate from a tightly coupled monolithic architecture to a microservices architecture. The approach also helps you to monitor and manage the entire lifecycle of a Java application, which ensures a higher level of automation and helps avoid errors or bugs.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI version 2 in the AWS CLI documentation.
- AWS CLI version 2 must be configured with the same IAM user that creates the Amazon EKS cluster because only they are authorized to add other IAM users, groups, or roles to the aws-auth ConfigMap. For information and steps to configure AWS CLI, see Configuration basics in the AWS CLI documentation.
- AWS Identity and Access Management (IAM) roles and permissions with full access to AWS CloudFormation. For more information about this, see Controlling access with IAM in the AWS CloudFormation documentation.
- An existing Amazon EKS cluster.
- Kubernetes Cluster Autoscaler, installed and configured in your Amazon EKS cluster. For more information about this, see Cluster Autoscaler in the Amazon EKS documentation.
- The java-cicd-eks-cfn.zip file (attached), downloaded to your local computer.

Limitations

- This pattern's approach doesn't implement container-specific logging and monitoring tools, such as Fluentd, Amazon CloudWatch, Grafana, or Prometheus.
- This approach doesn't deploy containers to Amazon EKS clusters across multiple accounts.
Product versions

- Helm version 3.4.2 or later
- Apache Maven version 3.6.3 or later

Architecture

The diagram shows the following workflow:

1. Users commit the Java application code changes to an AWS CodeCommit repository.
2. An Amazon CloudWatch Events event is generated by the new commit.
3. The CloudWatch Events event initiates AWS CodePipeline.
4. CodePipeline runs the build phase (continuous integration).
5. CodeBuild builds the artifact, packages the artifact to a Docker image, scans the image for security vulnerabilities by using Aqua Security Trivy, and stores the image in Amazon Elastic Container Registry (Amazon ECR).
6. After the continuous integration phases are complete, CodePipeline enters the deployment phase (continuous delivery).
7. The Docker image is deployed to Amazon EKS as a container workload using Helm charts.

Tools

- **AWS CloudFormation**— AWS CloudFormation is a service that helps you model and set up your AWS resources.
- **AWS CodeBuild**— CodeBuild is a fully managed build service in the cloud.
- **AWS CodeCommit**— CodeCommit is a version control service hosted that you can use to privately store and manage assets.
- **AWS CodePipeline**— CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.
• **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.

• **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

• **AWS Identity and Access Management** – IAM is a web service that helps you securely control access to AWS resources.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

• **Aqua Security Trivy** – Trivy is a simple and comprehensive scanner for vulnerabilities in container images, file systems, and Git repositories, in addition to configuration issues.

• **Apache Maven** – Apache Maven is a software project management and comprehension tool.

• **Helm** – Helm helps you manage Kubernetes applications.

**Code**

The `java-cicd-eks-cfn.zip` file (attached) contains the build scripts, Helm chart configuration files, and IAM roles and policies required for this pattern. The code has the following structure, with 15 directories and 22 files.

```plaintext
### README
### app_code (Java application code)
# # # app
# # # Dockerfile (the Docker file to package the application artifact)
# # # pom.xml
# # # src
# # # main
# # # java
# # # # org
# # # # aws
# # # # samples
# # # # greeting
# # # # Application.java
# # # # GreetingController.java
# # # resources
# # # Images
# # # # aws_proserve.jpg
# # # app_code.zip (.zip format of the code to be uploaded to an S3 bucket, in case you need to create a CodeCommit repository and upload the code automatically by using cf_templates/codecommit.yaml)
# # # aws-proserve-java-greeting (the application name and Helm charts)
# # # Chart.yaml
```
# AWS Prescriptive Guidance Patterns

## Epics

### Create the AWS resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an S3 bucket. | 1. Sign in to the AWS Management Console, open the Amazon S3 console, and then create an S3 bucket.  
2. Create a folder in the S3 bucket. We recommend naming this folder “Code.”  
3. Open the `java-cicd-eks-cfn.zip` file (attached) and upload the `app_code/app_code.zip` to the Code folder that you created in the S3 bucket.  
   For more information about this, see [Creating a bucket](https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-a-bucket.html) in the Amazon S3 documentation. | AWS DevOps, DevOps |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an AWS CloudFormation stack. | 1. Open the AWS CloudFormation console and choose Create stack.  
2. In Specify template, choose Upload a template file, upload the cf_templates/codecommit.yaml file, and then choose Next.  
3. In Specify stack details, enter the stack name, and then provide the following input parameter values:  
   - CodeCommitRepositoryBranchName – The default value is the main branch name but any other branch name can be provided.  
   - CodeCommitRepositoryName – The name for your CodeCommit repository.  
   - CodeCommitRepositoryS3Bucket – The name of the S3 bucket that you created earlier.  
   - CodeCommitRepositoryS3BucketObjKey – The value for this is code/app_code.zip.  
4. Choose Next, use the default settings for the Configure stack options, and then choose Next.  
5. In the Review section, verify the template and stack details, and then choose Create stack. The stack is then created and this also creates the CodeCommit and Amazon ECR repositories. | AWS DevOps, DevOps |
| Delete the S3 bucket. | Empty and delete the S3 bucket that you created earlier. For more information about this, see Deleting a bucket in the Amazon S3 documentation. | AWS DevOps, DevOps |
## Configure the Helm charts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure the Helm charts of your Java application. | Open the `app_code/aws-proserve-java-greeting` folder. The `values.dev.yaml` file is the input file that can be modified for your container deployments to Amazon EKS. Change the Docker repository with the following parameter by replacing AWS account ID, AWS Region, and Amazon ECR repository information:  

```yaml
image:
  repository: <account-id>.dkr.ecr.<region>.amazonaws.com/aws-proserve-java-docker
```

The Java pod's service type is set to `LoadBalancer`. If you need to change it to another service (for example, `NodePort`), you can modify the following parameters:

```yaml
service:
  type: LoadBalancer
  port: 80
  targetPort: 8080
  path: /hello
```

The **Kubernetes Horizontal Pod Autoscaler** can be turned on by modifying the following parameters:

```yaml
autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80
```

This means that you can turn on different features for the Kubernetes workloads by changing the values in the `{ENV}.yaml` file, in which `{ENV}` can be your development,
Set up the CI/CD pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the CI/CD pipeline.</td>
<td>1. Open the AWS CloudFormation console and choose Create stack. 2. In Specify template, choose Upload a template file, upload the cf_templates/build_deployment.yaml template, and then choose Next. 3. In Specify stack details, specify the Stack name, and then provide the following values for the input parameters:  - CodeBranchName – The default value is the main branch where the Java application code, Docker file, and Helm charts are located.  - EKSClusterName – The name of the Amazon EKS cluster.  - EKSCloneAppNane – The name of the Java application. This pattern uses aws-proserve-java-greeting.  - EcrDockerRepository – The name of your Amazon ECR repository.  - EnvType – Your environment type.  - SourceRepoName – The name of your CodeCommit repository. 4. Choose Next. Use the default settings in Configure stack options and then choose Next. 5. In the Review section, verify the AWS CloudFormation template and stack details.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
### Allow CodeBuild to run Helm or kubectl commands

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Allow CodeBuild to run Helm or kubectl commands in the Amazon EKS cluster. | 1. CodeBuild needs to authenticate with the Amazon EKS cluster to use Helm or kubectl commands. The `aws-auth` ConfigMap in the Amazon EKS cluster is responsible for authenticating IAM users, groups, or roles, which must be added to the `aws-auth` ConfigMap.  
  2. Open the `cf_templates/kube_aws_auth_configmap_patch.sh` shell script in your Amazon Linux or macOS environment.  
  3. Authenticate to the Amazon EKS cluster by running the following command:  
     ```bash
     aws eks --region <aws-region> update-kubeconfig --name <eks-cluster-name>
     ```  
  4. Run the shell script with the following command:  
     ```bash
     bash cf_templates/kube_aws_auth_configmap_patch.sh <rolearn-eks-codebuild-kubectl>
     ```  
  **Note:** Replace `<rolearn-eks-codebuild-kubectl>` with the ARN value of `EksCodeBuildkubRoleARN` that you recorded earlier.  
  5. The `aws_auth` ConfigMap is configured and access is granted. | DevOps                                                             |
### Validate the CI/CD pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Verify that the CI/CD pipeline automatically initiates. | The build phase in the pipeline will initially fail when Trivy scans the Docker image for vulnerabilities. This is intentional and is caused by an old version of the spring-boot-starter-parent in the app_code/app/pom.xml file that validates that the Docker image is tested against well-known application and Docker common vulnerabilities and exposures (CVEs). Use the following steps to resolve this build failure:  

1. Update the spring-boot-starter-parent version to the most recent version in the file app_code/app/pom.xml file in your CodeCommit repository. You can find the most recent version in the Maven Spring Boot Starter Parent repository.  
2. Commit the changes to the CodeCommit repository.  
3. The pipeline automatically initiates and the build phase is successful.  

For more information about this, see Start a pipeline execution in CodePipeline, Start a pipeline manually, and Start a pipeline on a schedule from the AWS CodePipeline documentation. | DevOps |
| Approve the deployment. | After the build phase is complete, there is a deployment approval gate. The reviewer or a release manager should inspect the build and, if all requirements are met, approve it.  

When approved, the pipeline implements the deployment phase. | DevOps |
Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS

Created by Durga Prasad Cheepuri (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Containers &amp; microservices; Cloud-native; Management &amp; governance; Storage &amp; backup; Websites &amp; web apps</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon ECS; Amazon EFS</td>
</tr>
</tbody>
</table>

Summary

This pattern provides code samples and steps to create an Amazon Elastic Container Service (Amazon ECS) task definition that runs on Amazon Elastic Compute Cloud (Amazon EC2) instances in the Amazon Web Services (AWS) Cloud, while using Amazon Elastic File System (Amazon EFS) to mount a file system on those EC2 instances. Amazon ECS tasks that use Amazon EFS automatically mount the file systems that you specify in the task definition and make these file systems available to the task's containers across all Availability Zones in an AWS Region.

To meet your persistent storage and shared storage requirements, you can use Amazon ECS and Amazon EFS together. For example, you can use Amazon EFS to store persistent user data and application data for your applications with active and standby ECS container pairs running in different Availability Zones for high availability. You can also use Amazon EFS to store shared data that can be accessed in parallel by ECS containers and distributed job workloads.

To use Amazon EFS with Amazon ECS, you can add one or more volume definitions to a task definition. A volume definition includes an Amazon EFS file system ID, access point ID, and a configuration for AWS Identity and Access Management (IAM) authorization or Transport Layer Security (TLS) encryption in transit. You can use container definitions within task definitions to specify the task definition volumes that get mounted when the container runs. When a task that uses an Amazon EFS file system runs, Amazon ECS ensures that the file system is mounted and available to the containers that need access to it.

Prerequisites and limitations

Prerequisites
• An active AWS account
• A virtual private cloud (VPC) with a virtual private network (VPN) endpoint or a router
• (Recommended) Amazon ECS container agent 1.38.0 or later for compatibility with Amazon EFS access points and IAM authorization features (For more information, see the AWS blog post New for Amazon EFS – IAM Authorization and Access Points.)

Limitations

• Amazon ECS container agent versions earlier than 1.35.0 don’t support Amazon EFS file systems for tasks that use the EC2 launch type.

Architecture

The following diagram shows an example of an application that uses Amazon ECS to create a task definition and mount an Amazon EFS file system on EC2 instances in ECS containers.

The diagram shows the following workflow:

1. Create an Amazon EFS file system.
2. Create a task definition with a container.
3. Configure the container instances to mount the Amazon EFS file system. The task definition references the volume mounts, so the container instance can use the Amazon EFS file system. ECS tasks have access to the same Amazon EFS file system, regardless of which container instance those tasks are created on.
4. Create an Amazon ECS service with three instances of the task definition.

Technology stack

• Amazon EC2
• Amazon ECS
• Amazon EFS
Tools

- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of EC2 instances that you manage.
- **Amazon EFS** – Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources.
- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

Epics

Create an Amazon EFS file system

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon EFS file system by using the AWS Management Console.</td>
<td>1. Create an Amazon EFS file system and choose the VPC that includes your containers. <strong>Note</strong>: If you use a different VPC, set up a VPC peering connection. 2. Note the file system ID.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

Create an Amazon ECS task definition by using either an Amazon EFS file system or the AWS CLI

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a task definition using an Amazon EFS file system.</td>
<td>Create a task definition by using the new Amazon ECS console or classic Amazon ECS console with the following configurations: - If you use the new console, choose Amazon EC2 instances for App environment. If you use the classic console, choose EC2 as the launch type. - Add a volume. Enter a name for the volume, choose EFS for volume type, and then choose the file system ID that</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Create a task definition using the AWS CLI. | 1. To create a JSON template with input parameter placeholders for your task definition, run the following command:  

```bash
aws ecs register-task-definition --generate-cli-skeleton
```

2. To create the task definition with the JSON template, run the following command:

```bash
aws ecs register-task-definition --cli-input-json file://<path_to_your_json_file>
```

3. Enter the input parameters in your JSON template based on the `task_definition_parameters.json` file (attached). **Note:** For more information on input parameters, see Task definition parameters (Amazon ECS documentation) and register-task-definition (AWS CLI Command Reference). | AWS DevOps |

**Related resources**

- Amazon ECS task definitions
- Amazon EFS volumes

**Attachments**

attachment.zip

**Deploy Java microservices on Amazon ECS using AWS Fargate**
Summary

This pattern provides guidance for deploying containerized Java microservices on Amazon Elastic Container Service (Amazon ECS) using AWS Fargate. The pattern doesn’t use Amazon Elastic Container Registry (Amazon ECR) for container management; instead, Docker images are pulled in from the on-premises environment.

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

Amazon ECS is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS.

AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters.

Prerequisites and limitations

Prerequisites

• An existing Java microservices application on a Docker hub
• A public Docker repository
• An active AWS account
• Familiarity with AWS services, including Amazon ECS and Fargate
• Docker, Java, and Spring Boot framework
• Amazon Relational Database Service (Amazon RDS) up and running (optional)
• A virtual private cloud (VPC) if the application requires Amazon RDS (optional)

Limitations

• Fargate is currently supported only in Amazon ECS.

Architecture

Source technology stack

• Java microservices (for example, implemented in Spring Boot) and deployed on Docker
Source architecture

Target technology stack
- An Amazon ECS cluster that hosts each microservice using Fargate
- A VPC network to host the Amazon ECS cluster and associated security groups
- A cluster/task definition for each microservice that spins up containers using Fargate

Target architecture

Tools
- **Amazon ECS** – Amazon ECS eliminates the need to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.

- **AWS Fargate** – With the AWS Fargate compute engine, you no longer have to provision, configure, and scale clusters of virtual machines to run containers, choose server types, decide when to scale your clusters, or optimize cluster packing.

- **Docker file** – Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run, including libraries, system tools, code, and runtime.

### Docker code

The following DockerFile specifies the Java Development Kit (JDK) version that is used, where the Java archive (JAR) file exists, the port number that is exposed, and the entry point for the application.

```
FROM openjdk:8
ADD target/Spring-docker.jar Spring-docker.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","Spring-docker.jar"]
```

### Epics

#### Set up a VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) in your AWS account.</td>
<td></td>
<td>System Admin, Developer</td>
</tr>
<tr>
<td>Create subnets inside the VPC.</td>
<td>A minimum of two subnets are required.</td>
<td>System Admin, Developer</td>
</tr>
<tr>
<td>Create an Amazon RDS DB instance (optional).</td>
<td>If you require Amazon RDS, create it in one of your private subnets.</td>
<td>System Admin, Developer</td>
</tr>
<tr>
<td>Ensure that the security group allows access to Amazon RDS (optional).</td>
<td>If you require Amazon RDS, ensure that the security group allows access to Amazon RDS.</td>
<td>System Admin, Developer</td>
</tr>
</tbody>
</table>

#### Create new task definitions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a task definition.</td>
<td>Running a Docker container in Amazon ECS requires a task definition. Open the Amazon ECS console at <a href="https://console.aws.amazon.com/ecs/">https://console.aws.amazon.com/ecs/</a>, choose &quot;Task definitions,&quot; and create a new task definition.</td>
<td>System Admin, Developer</td>
</tr>
</tbody>
</table>
Choose launch type. | Choose Fargate as the launch type. | System Admin, Developer

Configure the task. | Define a task name and configure the application with the appropriate amount of task memory and CPU. | System Admin, Developer

Define the container. | Specify the container name. For the image, enter the Docker site name, the repository name, and the tag name of the Docker image. Set memory limits for the application, and set port mappings for the allowed ports. | System Admin, Developer

Create the task. | When the task and container configurations are in place, create the task. For detailed instructions, see the links in the References section. | System Admin, Developer

**Configure the cluster**

Create and configure a cluster. | Select networking as the cluster type, configure the name, and then create the cluster. | System Admin, Developer

Configure tasks. | Go to Tasks and choose "Run new task." Configure the launch type as Fargate and select the task and the cluster you previously created. Configure the VPC, subnet, and security group to allow access. | System Admin, Developer

**Cut over to Amazon ECS**

Test the application. | Test the application using the public DNS, and then cut over to Amazon ECS. | System Admin, Developer

**Related resources**

- Docker Basics for Amazon ECS (Amazon ECS documentation)
- Amazon ECS on AWS Fargate (Amazon ECS documentation)
Deploy Java microservices on Amazon ECS using Amazon ECR and AWS Fargate

Created by Vijay Thompson (AWS) and Sandeep Bondugula (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Containers</th>
<th>Target:</th>
<th>Amazon ECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>N/A</td>
<td>Technologies:</td>
<td>Containers &amp; microservices; Websites &amp; web apps</td>
<td>AWS services:</td>
<td>Amazon ECS</td>
</tr>
</tbody>
</table>

**Summary**

This pattern guides you through the steps for deploying Java microservices as containerized applications in Amazon Elastic Container Service (Amazon ECS). The pattern also uses Amazon Elastic Container Registry (Amazon ECR) to manage your container, and AWS Fargate to run your container.

**Prerequisites and limitations**

**Prerequisites**

- An existing Java microservices application running on premises on Docker
- An active AWS account
- Familiarity with Amazon ECR, Amazon ECS, AWS Fargate, and AWS Command Line Interface (AWS CLI)
- Familiarity with Java and Docker software

**Product versions**

- AWS CLI version 1.7 or later

**Architecture**

**Source technology stack**

- Java microservices (for example, developed using Spring Boot) and deployed on premises
- Docker

**Source architecture**
Target technology stack

- Amazon ECR
- Amazon ECS
- AWS Fargate

Target architecture
Tools

Tools

- **Amazon ECR** - Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS to simplify your development-to-production workflow. Amazon ECR hosts your images in a highly available and scalable architecture so you can reliably deploy containers for your applications. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.

- **Amazon ECS** - Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need for you to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.

- **AWS Fargate** - AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.

- **Docker** - Docker is a platform that lets you build, test, and deliver applications in packages called containers.

Code

The following DockerFile specifies the Java Development Kit (JDK) version that is used, where the Java archive (JAR) file exists, the port number that is exposed, and the entry point for the application.

```
FROM openjdk:8
ADD target/Spring-docker.jar Spring-docker.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","Spring-docker.jar"]
```

Epics

Create an Amazon ECR repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a repository.</td>
<td>Sign in to the AWS Management Console, and open the Amazon ECR console at <a href="https://console.aws.amazon.com/ecr/repositories">https://console.aws.amazon.com/ecr/repositories</a>. Create a private repository. For instructions, see Creating a private repository in the Amazon ECR documentation.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Upload the project.</td>
<td>Open the repository and choose View push commands. Follow the steps displayed to upload the project. (These steps work only when you use AWS CLI version 1.7 or later.) When the upload is complete, copy the URL of the build in the</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>
### Create and spin up the container

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a task definition.</td>
<td>Running a Docker container in Amazon ECS requires a task definition. Open the Amazon ECS console at <a href="https://console.aws.amazon.com/ecs/">https://console.aws.amazon.com/ecs/</a>, choose <strong>Task definitions</strong>, and create a new task definition. For more information, see Creating a task definition in the Amazon ECS documentation.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Choose the launch type.</td>
<td>Choose Fargate as the launch type.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Configure the task.</td>
<td>Define a task name and configure the application with the appropriate amount of task memory and CPU.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Define the container.</td>
<td>Add the container, providing a name, the URL of the Amazon ECR repository, memory limits, and port mapping. Ports 8080 and 80 are configured for port mappings. Configure the remaining settings based on your application requirements.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Create the task.</td>
<td>When the task and container configurations are in place, create the task. For detailed instructions, see the links in the Related resources (p. 179) section.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

### Create an Amazon ECS cluster and configure a service

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create or choose a cluster.</td>
<td>An Amazon ECS cluster provides a logical grouping of tasks or services. You can opt to use an existing cluster or create a new cluster. If you decide to create a new cluster, choose</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Create a service.</td>
<td>Inside the cluster, choose <strong>Create service</strong>.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Choose the launch type.</td>
<td>Choose Fargate as the launch type.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Choose task definition, revision, and platform version.</td>
<td>Choose the task that you want to run, followed by the revision of the task definition and the platform version.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Select the cluster.</td>
<td>Select the cluster in which to create your service from the dropdown list.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Provide a service name.</td>
<td>Provide a unique name for the service that you are creating.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Specify the number of tasks.</td>
<td>Configure the number of tasks that should run when the service launches. If you're launching with two or more tasks, a load balancer is required to balance the tasks. The minimum number of tasks to be configured is <strong>One</strong>.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Set the minimum and maximum healthy percentages.</td>
<td>Configure the minimum and maximum healthy percentages for the application, or accept the default option that is provided.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Configure deployment settings.</td>
<td>Choose the deployment type based on your requirements. You can choose a rolling update or a blue/green deployment.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Configure the cluster VPC, subnets, and security groups.</td>
<td>Configure the cluster VPC, the subnets on which you want to deploy the application, and the security groups (HTTP, HTTPS, and port 8080) for providing access to inbound/outbound connections.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Configure public IP settings.</td>
<td>Enable or disable the public IP, depending on whether you want to use a public IP address for Fargate tasks.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>
Configure load balancing.

<table>
<thead>
<tr>
<th>Description</th>
<th>Configure the load balancer, if you’re launching the service with more than one task. You must create a load balancer and its target group before you launch the service.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skills required</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Configure automatic scaling.

<table>
<thead>
<tr>
<th>Description</th>
<th>Configure your service to use Amazon ECS Service Auto Scaling to adjust the desired number of tasks up or down, depending on your requirements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skills required</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Review settings and create the service.

<table>
<thead>
<tr>
<th>Description</th>
<th>Review your service settings, and then choose Create service.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skills required</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test your application.</td>
<td>Test the application by using the public DNS that’s created when the task is deployed. If the application has a load balancer, test the application by using it and then cut over.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

Related resources

- Docker basics for Amazon ECS (Amazon ECS documentation)
- Amazon ECS on AWS Fargate (Amazon ECS documentation)
- Creating a private repository (Amazon ECR documentation)
- Creating a task definition (Amazon ECS documentation)
- Container definitions (Amazon ECS documentation)
- Creating a cluster (Amazon ECS documentation)
- Configuring basic service parameters (Amazon ECS documentation)
- Configuring a network (Amazon ECS documentation)
- Configuring your service to use a load balancer (Amazon ECS documentation)
- Configuring your service to use Service Auto Scaling (Amazon ECS documentation)
- How to Deploy Java Spring Boot on AWS Fargate (blog post)

Deploy Java microservices on Amazon ECS using Amazon ECR and load balancing

*Created by Durga Prasad Cheepuri (AWS)*
Summary

This pattern outlines steps for deploying a containerized Java microservices architecture on Amazon Elastic Container Service (Amazon ECS) to make it easier to scale and faster to develop your applications. This helps enable innovation and accelerates time-to-market for new features.

The pattern also uses Amazon Elastic Container Registry (Amazon ECR) to store and manage the Docker-based containers, and an AWS CloudFormation template with a Python script to automate the setup of your infrastructure. The pattern is based on the post Deploying Java Microservices on Amazon Elastic Container Service, which is published on the AWS Compute blog.

Microservices provide an architectural and organizational approach to software development, where software is composed of small, independent services that communicate over well-defined application programming interfaces (APIs). Small, self-contained teams own these services.

Amazon ECS is a highly scalable, high-performance container orchestration service. It supports Docker containers and enables you to run and scale containerized applications on AWS quickly. With Amazon ECS, you no longer have to install and operate your container orchestration software, manage and scale a cluster of virtual machines (VMs), or schedule containers on those VMs.

With simple API calls, you can launch and stop Docker-enabled applications, query the complete state of your request, and access many natural features, such as AWS Identity and Access Management (IAM) roles, security groups, load balancers, Amazon CloudWatch Events, AWS CloudFormation templates, and AWS CloudTrail logs.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Java microservices source code, with Java Development Kit version 1.7 or later
- An access key and secret access key for a user in the account
- AWS Command Line Interface (AWS CLI)
- Java, AWS Software Development Kit (SDK) for Python (Boto3), and Docker software
- Familiarity with the use of the preceding technologies
- Familiarity with AWS services such as Amazon ECS, AWS CloudFormation, and Elastic Load Balancing

Architecture

Source technology stack

- Microservices implemented in Java and deployed on Apache Tomcat in an on-premises environment

Target technology stack
• The Application Load Balancer that inspects the client request. Based on routing rules, the load balancer directs the request to an instance and port from the target group that matches the state.

• A target group for each microservice. The target groups are used by the corresponding services to register available container instances. Each target group has a path, so when you call the way for a particular microservice, it maps to the correct target group. This enables you to use one Application Load Balancer to serve all the microservices, accessed by the path. For example, https://owner/* would map and direct to the Owner microservice.

• An Amazon ECS cluster that hosts the containers for each microservice.

• An Amazon Virtual Private Cloud (Amazon VPC) network for hosting the Amazon ECS cluster and associated security groups.

• An Amazon Elastic Container Registry (Amazon ECR) repository for each microservice.

• A service or task definition for each microservice, which spins up containers on the instances of the Amazon ECS cluster.

**Target architecture**
Tools

- Amazon ECS – Amazon ECS lets you launch and stop container-based applications with simple API calls, enables you to get the state of your cluster from a centralized service, and gives you access to many familiar Amazon Elastic Compute Cloud (Amazon EC2) features.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS to simplify your development-to-production workflow. Amazon ECR
hosts your images in a highly available and scalable architecture so you can reliably deploy containers for your applications. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.

Epics

Create an AWS CloudFormation template to set up an Amazon ECS cluster to host the Java microservices

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provision an Amazon EC2 Linux instance, install Docker, and create a Docker file for each microservice.</td>
<td>Use the Dockerfile for the image to push, build the image, and tag it for your new repository. Do the same for each microservice. Push the newly tagged images to the repository.</td>
<td>Ops</td>
</tr>
<tr>
<td>Set up Docker images on Amazon ECR.</td>
<td></td>
<td>Ops</td>
</tr>
<tr>
<td>Create an AWS CloudFormation template.</td>
<td>Create an AWS CloudFormation template to provision the virtual private cloud (VPC), Amazon ECS cluster, and Amazon Relational Database Service (Amazon RDS).</td>
<td>Ops</td>
</tr>
</tbody>
</table>

Provision AWS services

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS infrastructure by using the CloudFormation template you created earlier.</td>
<td>Use the Python script at <a href="https://github.com/awslabs/amazon-ecs-java-microservices/blob/master/2_ECS_Java_Spring_PetClinic_Microservices/setup.py">https://github.com/awslabs/amazon-ecs-java-microservices/blob/master/2_ECS_Java_Spring_PetClinic_Microservices/setup.py</a> to invoke the AWS CloudFormation template you created earlier. This template creates the AWS infrastructure you need for the target environment.</td>
<td>Ops</td>
</tr>
<tr>
<td>Create Amazon ECR repositories, tasks, services, the Application Load Balancer, and target groups.</td>
<td>The Python script reads the outputs of the AWS CloudFormation template and uses BOTO3 API calls to create Amazon ECR repositories, tasks, services, the Application Load Balancer, and target groups.</td>
<td>Ops</td>
</tr>
</tbody>
</table>
Deploy Kubernetes resources and packages using Amazon EKS and a Helm chart repository in Amazon S3

Created by Sagar Panigrahi (AWS)

| Environment: | PoC or pilot | Technologies: | Containers & microservices; DevOps | AWS services: | Amazon EKS |

Summary

This pattern helps you to manage Kubernetes applications efficiently, regardless of their complexity. The pattern integrates Helm into your existing continuous integration and continuous delivery (CI/CD) pipelines to deploy applications into a Kubernetes cluster. Helm is a Kubernetes package manager that helps you manage Kubernetes applications. Helm charts help to define, install, and upgrade complex Kubernetes applications. Charts can be versioned and stored in Helm repositories, which improves mean time to restore (MTTR) during outages.

This pattern uses Amazon Elastic Kubernetes Service (Amazon EKS) for the Kubernetes cluster. It uses Amazon Simple Storage Service (Amazon S3) as a Helm chart repository, so that the charts can be centrally managed and accessed by developers across the organization.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account with a virtual private cloud (VPC)
- An Amazon EKS cluster
- Worker nodes set up within the Amazon EKS cluster and ready to take workloads
- Kubectl for configuring the Amazon EKS kubeconfig file for the target cluster in the client machine
- AWS Identity and Access Management (IAM) access to create the S3 bucket
- IAM (programmatic or role) access to Amazon S3 from the client machine
• Source code management and a CI/CD pipeline

Limitations
• There is no support at this time for upgrading, deleting, or managing custom resource definitions (CRDs).
• If you are using a resource that refers to a CRD, the CRD must be installed separately (outside of the chart).

Product versions
• Helm v3.6.3

Architecture

Target technology stack
• Amazon EKS
• Amazon VPC
• Amazon S3
• Source code management
• Helm
• Kubectl

Target architecture

Automation and scale
• AWS CloudFormation can be used to automate the infrastructure creation. One option is to use the Modular and Scalable Amazon EKS Architecture Quick Start.
• Helm is to be incorporated into your existing CI/CD automation tool to automate the packaging and
versioning of Helm charts (out of scope for this pattern).
• GitVersion or Jenkins build numbers can be used to automate the versioning of charts.

**Tools**

**Tools**

• **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service for running
Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane.
Kubernetes is an open-source system for automating the deployment, scaling, and management of
containerized applications.
• **Helm** – Helm is a package manager for Kubernetes that helps you install and manage applications on
your Kubernetes cluster.
• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use
Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
• **Kubectl** – Kubectl is a command line utility for running commands against Kubernetes clusters.

**Code**

The example code is attached.

**Epics**

**Configure and initialize Helm**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the Helm client.</td>
<td>To download and install the Helm client on your local system, use the following command.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bash</td>
</tr>
<tr>
<td>Validate the Helm installation.</td>
<td>To validate that Helm is able to communicate with the Kubernetes API server within the Amazon EKS cluster, run helm version.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

**Create and install a Helm chart in the Amazon EKS cluster**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Helm chart for NGINX.</td>
<td>To create a helm chart named my-nginx on the client</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>machine, run helm create my-nginx.</strong></td>
<td><strong>Review the structure of the chart.</strong></td>
<td><strong>DevOps engineer</strong></td>
</tr>
<tr>
<td><strong>review the structure of the chart, run the tree command tree my-nginx/</strong>.**</td>
<td><strong>Deactivate service account creation in the chart.</strong></td>
<td><strong>DevOps engineer</strong></td>
</tr>
<tr>
<td><strong>In values.yaml, under the serviceAccount section, set the create key to false. This is turned off because there is no requirement to create a service account for this pattern.</strong></td>
<td><strong>Validate (lint) the modified chart for syntactical errors.</strong></td>
<td><strong>DevOps engineer</strong></td>
</tr>
<tr>
<td><strong>To validate the chart for any syntactical error before installing it in the target cluster, run helm lint my-nginx/</strong>.**</td>
<td><strong>Install the chart to deploy Kubernetes resources.</strong></td>
<td><strong>DevOps engineer</strong></td>
</tr>
<tr>
<td><strong>To run the Helm chart installation, use the following command.</strong></td>
<td><strong>helm install --name my-nginx-release --debug my-nginx/ --namespace helm-space</strong></td>
<td><strong>The optional debug flag outputs all debug messages during the installation. The namespace flag specifies the namespace in which the resources part of this chart will be created.</strong></td>
</tr>
<tr>
<td><strong>To review the resources that were created as part of the Helm chart in the helm-space namespace, use the following command.</strong></td>
<td><strong>kubectl get all -n helm-space</strong></td>
<td><strong>DevOps engineer</strong></td>
</tr>
</tbody>
</table>

**Roll back to a previous version of a Kubernetes application**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modify and upgrade the release.</strong></td>
<td><strong>To modify the chart, in values.yaml, change the replicaCount value to 2. Then upgrade the already installed release by running the following command.</strong></td>
<td><strong>DevOps engineer</strong></td>
</tr>
</tbody>
</table>

---

187
## AWS Prescriptive Guidance Patterns
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Role the history of the Helm release. | To list all the revisions for a specific release that have been installed using Helm, run the following command.  

```bash
helm history my-nginx-release
```
| DevOps engineer |
|------|----------------|
| Review the details for a specific revision. | Before switching or rolling back to a working version, and for an additional layer of validation before installing a revision, view which values were passed to each of the revisions by using the following command.  

```bash
helm get --revision=2 my-nginx-release
```
| DevOps engineer |
| Roll back to a previous version. | To roll back to a previous revision, use the following command.  

```bash
helm rollback my-nginx-release 1
```
| DevOps engineer |

### Initialize an S3 bucket as a Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket for Helm charts.</td>
<td>Create a unique S3 bucket. In the bucket, create a folder called charts. The example in this pattern uses <code>s3://my-helm-charts/charts</code> as the target chart repository.</td>
<td></td>
</tr>
<tr>
<td>Cloud administrator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Install the Helm plugin for Amazon S3. | To install the helm-s3 plugin on your client machine, use the following command.  

```bash
helm plugin install https://github.com/hypnoglow/helm-s3.git --version 0.10.0
```
| DevOps engineer |
### Initialize the Amazon S3 Helm repository.

**Task**
- To initialize the target folder as a Helm repository, use the following command.

```
helm s3 init s3://my-helm-charts/charts
```

**Skills required**
- DevOps engineer

*Note: Helm V3 support is available with plugin version 0.9.0 and above.*

*The command creates an `index.yaml` file in the target to track all the chart information that is stored at that location.*

### Add the Amazon S3 repository to Helm.

**Task**
- To add the repository in the client machine, use the following command.

```
helm repo add my-helm-charts s3://my-helm-charts/charts
```

**Skills required**
- DevOps engineer

*This command adds an alias to the target repository in the Helm client machine.*

### Review the repository list.

**Task**
- To view the list of repositories in the Helm client machine, run `helm repo list`.

**Skills required**
- DevOps engineer

### Package and store charts in the Amazon S3 Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package the chart.</td>
<td>To package the <code>my-nginx</code> chart that you created, run <code>helm package ./my-nginx/</code>. The command packages all the contents of the <code>my-nginx</code> chart folder into an archive file, which is named using the version number that is mentioned in the <code>Chart.yaml</code> file.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Store the package in the Amazon S3 Helm repository.</td>
<td>To upload the package to the Helm repository in Amazon S3, run the following command, using the correct name of the <code>.tgz</code> file.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
### Modify, version, and package a chart

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Modify and package the chart.             | In `values.yaml`, set the `replicaCount` value to 1. Then package the chart by running `helm package ./my-nginx/`, this time changing the version in `Chart.yaml` to 0.1.1.  

The versioning is ideally updated through automation using tools such as GitVersion or Jenkins build numbers in a CI/CD pipeline. Automating the version number is out of scope for this pattern. | DevOps engineer |
| Push the new version to the Helm repository in Amazon S3. | To push the new package with version of 0.1.1 to the `my-helm-charts` Helm repository in Amazon S3, run the following command.  

```
helm s3 push ./my-nginx-0.1.1.tgz my-helm-charts
```

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Search for the Helm chart.                | To confirm that the chart appears both locally and in the Helm repository in Amazon S3, run the following command.  

```
helm search repo my-nginx
```

### Search for and install a chart from the Amazon S3 Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Search for all versions of the my-nginx chart. | To view all the available versions of a chart, run the following command with the `--versions` flag.  

```
helm search repo my-nginx
```

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>helm search repo my-nginx --versions</td>
<td>Without the flag, Helm by default displays the latest uploaded version of a chart.</td>
<td></td>
</tr>
<tr>
<td>Install a chart from the Amazon S3 Helm repository.</td>
<td>The search results from the previous task show the multiple versions of the my-nginx chart. To install the new version (0.1.1) from the Amazon S3 Helm repository, use the following command.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>helm upgrade my-nginx-release my-helm-charts/my-nginx --version 0.1.1 --namespace helm-space</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

- HELM documentation
- helm-s3 plugin (MIT License)
- HELM client binary
- Amazon EKS documentation

### Attachments

attachment.zip

## Deploy Lambda functions with container images

*Created by Ram Kandaswamy (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Containers &amp; microservices; Business productivity; Cloud-native; Cost management; Software development &amp; testing; Modernization; Serverless</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
</table>

| AWS services: AWS EC2 | Container Registry; AWS Lambda | |
Summary

AWS Lambda supports containers images as a deployment model. This pattern shows how to deploy Lambda functions through container images.

Lambda is a serverless, event-driven compute service that you can use to run code for virtually any type of application or backend service without provisioning or managing servers. With container image support for Lambda functions, you get the benefits of up to 10 GB of storage for your application artifact and the ability to use familiar container image development tools.

Prerequisites and limitations

Prerequisites

- Amazon Elastic Container Registry (Amazon ECR) activated
- Application code
- Docker images with the runtime interface client

Limitations

- Maximum image size supported is 10 GB.
- Maximum runtime for a Lambda based container deployment is 15 minutes.

Architecture

Target technology stack

- AWS CodeBuild
- AWS CodeCommit
- Docker image
- Amazon Elastic Container Registry (Amazon ECR)
- AWS Identity and Access Management (IAM)
- AWS Lambda
- Amazon CloudWatch Logs

Target architecture
1. You create a repository and commit the application code using CodeCommit.
2. The CodeBuild project is created, using CodeCommit as the source provider.
3. The CodeBuild run creates the Docker image.
4. CodeBuild publishes the image to Amazon ECR.
5. You create the Lambda function using the image in Amazon ECR.

**Automation and scale**

This pattern can be automated by using AWS CloudFormation, AWS Cloud Development Kit (AWS CDK), or API operations from an SDK. Lambda can automatically scale based on the number of requests, and you can tune it by using the concurrency parameters.

**Tools**

- **AWS CloudFormation Designer** integrated JSON and YAML editor – AWS CloudFormation Designer provides an integrated JSON and YAML editor for viewing and editing template details. For example, you can use the integrated editor to define the properties of a resource or to change a template parameter.
- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeCommit** – AWS CodeCommit is a version control service hosted by AWS. You can use CodeCommit to privately store and manage assets (such as documents, source code, and binary files) in the cloud.
- **AWS CodeStar** or another development environment – AWS CodeStar is a cloud-based service for creating, managing, and working with software development projects on AWS.
- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
**Docker** – Docker is a software platform that for building, testing, and deploying applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run, including libraries, system tools, code, and runtime.

# Epics

Create a project in CodeBuild

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CodeCommit repository.</td>
<td>The following approach uses Python as the underlying programming language, but you can use other languages, such as Java, Node.js, or Go. To store the configuration file and the files that will contain the code, create a CodeCommit repository.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create a CodeBuild project.</td>
<td>On the CodeBuild console, create a new project. Provide values for project name and description. Specify a source provider. This example uses CodeCommit. Other options include GitHub, Bitbucket, or Amazon Simple Storage Service (Amazon S3).</td>
<td>Developer</td>
</tr>
<tr>
<td>Edit the Dockerfile.</td>
<td>The Dockerfile should be located in the top-level directory where you're developing the application. The Python code should be in the <code>src</code> folder. When you create the image, use the official Lambda supported images. Otherwise, a bootstrap error will occur, making the packing process more difficult. For details, see the Additional information section.</td>
<td>Developer</td>
</tr>
<tr>
<td>Add the image in Amazon ECR.</td>
<td>Build the image and publish it to the image repository. Use CodeBuild to perform the image build process. Then push the image to Amazon ECR registry. For details and the code, see the Additional information section.</td>
<td>Developer</td>
</tr>
<tr>
<td>Verify that the image is in the repository.</td>
<td>To verify that the image is in the repository, on the Amazon ECR</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Related resources

- Runtime support for Lambda container images

#### Additional information

**Edit the Dockerfile**

The following screenshot shows the commands for editing the Dockerfile.

```
[cloudshell-user@ip-10-1-157-103 python-hello-docker]$ cat app/dockerfile

# first stage
FROM public.ecr.aws/lambda/python:3.8
COPY requirements.txt .

# install dependencies to the local user directory (eg. /root/.local)
RUN pip install --user -r requirements.txt

# Copy function code
COPY src/server.py ${LAMBDA_TASK_ROOT}

# Set the CMD to your handler (could also be done as a parameter override outside of the Dockerfile)
CMD [ "server.lambda_handler" ]
```

The **FROM** command value corresponds to the Python 3.8 base image that is using the Lambda function in the public Amazon ECR image repository.

The **COPY requirements.txt** command captures the dependencies necessary for the code.

The **RUN pip install --user -r requirements.txt** command installs the dependencies to the local user directory.

The **COPY src/server.py $(LAMBDA_TASK_ROOT)** command copies the code to the task root directory, which the Lambda function will use. This command uses the environment variable so we don't have to worry about the actual path. The function to be run is passed as an argument to the **CMD [ "server.lambda_handler" ]** command.
Add the image in Amazon ECR

In the following code, replace `11111111` with the account number, and replace `us-east-1` if you are using a different Region. The `buildspec` file uses the CodeBuild build number to uniquely identify image versions as a tag value. You can change this to fit your requirements.

**Buildspec custom code**

```yaml
phases:
  install:
    runtime-versions:
      python: 3.8
  pre_build:
    commands:
      - python3 --version
      - pip3 install --upgrade pip
      - pip3 install --upgrade awscli
      - docker info
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - ls
      - cd app
      - docker build -t cf-demo:$CODEBUILD_BUILD_NUMBER .
      - docker ls
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 111111111.dkr.ecr.us-east-1.amazonaws.com
      - docker tag cf-demo:$CODEBUILD_BUILD_NUMBER 111111111.dkr.ecr.us-east-1.amazonaws.com/cf-demo:$CODEBUILD_BUILD_NUMBER
      - docker push 111111111.dkr.ecr.us-east-1.amazonaws.com/cf-demo:$CODEBUILD_BUILD_NUMBER
```

Deploy a clustered application to Amazon ECS by using AWS Copilot

*Created by Jean-Baptiste Guillois (AWS), Mathew George (AWS), and Thomas Scott (AWS)*

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment:</th>
<th>Technologies:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Clustered Sample Application demo</td>
<td>Production</td>
<td>Containers &amp; microservices; Business productivity; Cloud-native; Software development &amp; testing</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon ECS; AWS Fargate; Amazon ECR

**Summary**

This pattern shows how to deploy containers in an Amazon Elastic Container Service (Amazon ECS) cluster in two ways—by using the Amazon Web Services (AWS) Management Console, and by using AWS Copilot—to demonstrate how AWS Copilot simplifies deployment tasks.
Amazon ECS is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.

The AWS Copilot command line interface (CLI) commands simplify building, releasing, and operating production-ready containerized applications on Amazon ECS from a local development environment. The AWS Copilot CLI aligns with developer workflows that support modern application best practices: from using infrastructure as code to creating a continuous integration and continuous delivery (CI/CD) pipeline provisioned on behalf of a user. You can use the AWS Copilot CLI as part of your everyday development and testing cycle as an alternative to the AWS Management Console.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) locally installed and configured to use your AWS account (see installation instructions and the configuration instructions in the AWS CLI documentation)
- AWS Copilot locally installed (see the installation instructions in the Amazon ECS documentation)
- Docker installed on your local machine (see the Docker documentation)

Limitations

- Docker enforces pull limits of 100 container images per 6 hours per IP address on the free plan.

Architecture

Target technology stack

- AWS environment set up with a virtual private cloud (VPC), public and private subnets, and security groups
- Amazon ECS cluster
- Amazon ECS service and task definition
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon DynamoDB
- Application Load Balancer
- AWS Fargate
- Amazon Identity and Access Management (IAM)
- Amazon CloudWatch
- AWS CloudTrail

Target architecture

When you deploy the sample application for this pattern, multiple tasks are created and deployed in separate Availability Zones. Each task stores data in Amazon DynamoDB. When you access the webpage for a task, you can view the data from all other tasks.
Tools

AWS services

- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private repositories with resource-based permissions using IAM.

- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.

- **AWS Copilot** – AWS Copilot provides a command line interface that helps you launch and manage containerized applications on AWS, including pushing to a registry, creating a task definition, and creating a cluster.

- **AWS Fargate** – AWS Fargate is a serverless, pay-as-you-go compute engine that lets you focus on building applications without managing servers. AWS Fargate is compatible with both Amazon ECS and Amazon Elastic Kubernetes Service (Amazon EKS). When you run your Amazon ECS tasks and services with the Fargate launch type or a Fargate capacity provider, you package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application. Each Fargate task has its own isolation boundary and doesn’t share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.

- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

- **Elastic Load Balancing (ELB)** – Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets.
Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

**Tools**

- Docker Command Line Interface
- AWS Command Line Interface (AWS CLI)
- AWS Copilot command line interface

**Code**

The code for the sample application used in this pattern is available on GitHub, in the [Cluster Sample Application](https://github.com/aws-samples/cluster-sample-app) repository. Follow the instructions in the next section to use the sample files.

**Epics**

**Deploy the application stack - option 1 (AWS Management Console)**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the GitHub repository.</td>
<td>Clone the sample code repository by using the command: <code>git clone https://github.com/aws-samples/cluster-sample-app cluster-sample-app &amp;&amp; cd cluster-sample-app</code></td>
<td>App developer, AWS DevOps</td>
</tr>
</tbody>
</table>
| Create your Amazon ECR repository. | 1. Sign in to the AWS Management Console and open the Amazon ECR console at [https://console.aws.amazon.com/ecr/repositories](https://console.aws.amazon.com/ecr/repositories). 2. Choose **Create repository**. 3. For the repository name, enter **cluster-sample-app**. 4. For all other settings, keep the default values. 5. Choose **Create repository**.  
For more information, see [Creating a private repository](https://docs.aws.amazon.com/AmazonECR/latest/userguide/creating-private-repository.html) in the Amazon ECR documentation. | App developer, AWS DevOps |
<p>| Build, tag, and push your Docker image to your Amazon ECR repository. | 1. Select the repository you just created and choose <strong>View push commands</strong>. 2. Copy the commands that are displayed and run them locally to build, tag, and push | App developer, AWS DevOps |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>your docker image. These commands will be similar to the following.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To authenticate your Docker client to the registry:</td>
<td>aws ecr get-login-password --region &lt;YOUR_AWS_REGION&gt;</td>
</tr>
<tr>
<td></td>
<td>To build your Docker image:</td>
<td>docker build -t cluster-sample-app .</td>
</tr>
<tr>
<td></td>
<td>To push the Docker image to your repository:</td>
<td>docker push &lt;YOUR_AWS_ACCOUNT&gt;.dkr.ecr.&lt;YOUR_AWS_REGION&gt;.amazonaws.com/cluster-sample-app:latest</td>
</tr>
</tbody>
</table>
### Deploy the application stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the application stack. | 1. Open the AWS CloudFormation console at [https://console.aws.amazon.com/cloudformation/](https://console.aws.amazon.com/cloudformation/).  
2. Choose **Create stack**.  
3. In the **Prepare template** section, choose **Template is ready**.  
4. In the **Specify template** section, choose **Upload a template file**.  
5. Choose the local file `cluster-sample-app-stack.yml` that you cloned from the GitHub repository as the CloudFormation template, and then choose **Next**.  
6. Enter a name for your stack, and then choose **Next**.  
7. Keep all default options, and then choose **Next**.  
8. Review all options, acknowledge the creation of IAM resources, and then choose **Create stack**.  
9. When your application stack has been deployed, choose the **Output** tab, copy the URL, and open it in your browser to access the application.  

For more information about deploying CloudFormation templates, see [Creating a stack](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/getting-started-deploying/templates.html) in the AWS CloudFormation documentation. | AWS DevOps, App developer |

### Deploy the application stack – option 2 (AWS Copilot CLI)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Clone the GitHub repository. | Clone the sample code repository by using the command:  
```bash  
git clone https://github.com/aws-samples/cluster-sample-appcluster-  ``` | App developer, AWS DevOps |
### Deploy your container image to AWS by using the AWS Copilot CLI.

**Task**

Deploy the application in one step by using the following command in the root directory of your project:

```
copilot init --app cluster-sample-app --name demo --type "Load Balanced Web Service" --dockerfile ./Dockerfile --port 8080 --deploy
```

**Skills required**

App developer, AWS DevOps

You should then be able to access the application by using the DNS name provided as output.

### Delete the created resources

#### Delete the resources created through the AWS Management Console.

**Task**

If you used option 1 (the AWS Management Console) to deploy the application stack, follow these steps when you’re ready to delete the resources you created:

2. Select the stack you created, and then choose **Delete**.
4. Select the repository you created, and then choose **Delete**.

**Skills required**

App developer, AWS DevOps

#### Delete the resources created by AWS Copilot.

**Task**

If you used option 2 (the AWS Copilot CLI) to deploy the application stack, run the following command from the root directory of your project when you’re ready to delete the resources you created:

```
sample-app && cd cluster-sample-app
```

**Skills required**

App developer, AWS DevOps
Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer

*Created by Kirankumar Chandrashekar (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Containers &amp; microservices; Content delivery; Web hosting; Websites &amp; web apps</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon EKS; Elastic Load Balancing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes how to host a gRPC-based application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster and securely access it through an Application Load Balancer.

gRPC is an open-source remote procedure call (RPC) framework that can run in any environment. You can use it for microservice integrations and client-server communications. For more information about gRPC, see Application Load Balancer support for end-to-end HTTP/2 and gRPC on the Amazon Web Services (AWS) Blog.

This patterns shows you how to host a gRPC-based application that runs as Kubernetes pods on Amazon EKS. The gRPC client connects to an Application Load Balancer through the HTTP/2 protocol with a Secure Sockets Layer/Transport Layer Security (SSL/TLS) encrypted connection. The Application Load Balancer forwards traffic to the gRPC application that runs on Amazon EKS pods. The number of gRPC pods can be automatically scaled based on traffic by using the Kubernetes Horizontal Pod Autoscaler. The Application Load Balancer’s target group performs health checks on the Amazon EKS nodes, evaluates if the target is healthy, and only forwards traffic to healthy nodes.

---

**Related resources**

- Installing or updating the latest version of the AWS CLI ([AWS CLI documentation](#))
- Using the AWS Copilot command line interface ([Amazon ECS documentation](#))
- Amazon ECS on AWS Fargate ([Amazon ECR documentation](#))
- Amazon ECS documentation
- Amazon ECR documentation
- Amazon CloudFormation documentation
- Docker Desktop ([Docker documentation](#))
Prerequisites and limitations

Prerequisites

- **Docker**, installed and configured on Linux, macOS, or Windows.
- An active AWS account.
- **AWS Command Line Interface (AWS CLI) version 2**, installed and configured on Linux, macOS, or Windows.
- An existing Amazon EKS cluster with tagged private subnets, public subnets, and configured to host applications. For more information, see Subnet tagging in the Amazon EKS documentation.
- **kubectl**, installed and configured to access resources on your Amazon EKS cluster. For more information, see Installing kubectl in the Amazon EKS documentation.
- **gRPCurl**, installed and configured.
- **AWS Load Balancer Controller**, provisioned in the Amazon EKS cluster.
- An existing Domain Name System (DNS) host name with a valid SSL or SSL/TLS certificate. You can obtain a certificate for your domain by using AWS Certificate Manager (ACM) or uploading an existing certificate to ACM. For more information about these two options, see Requesting a public certificate and Importing certificates into AWS Certificate Manager in the ACM documentation.
The following diagram shows a workflow where SSL/TLS traffic is received from a gRPC client that offloads to an Application Load Balancer. Traffic is forwarded in plaintext to the gRPC server because it comes from a virtual private cloud (VPC).
Tools

- **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command line shell.
- **Elastic Load Balancing** – Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses, in one or more Availability Zones.
- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is a fully managed Docker container registry.
- **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- **kubectl** – kubectl is a command line utility for running commands against Kubernetes clusters.
- **AWS Load Balancer Controller** – AWS Load Balancer Controller manages Elastic Load Balancers for a Kubernetes cluster.
- **gRPCurl** – gRPCurl is a command line tool that helps you interact with gRPC services.

Epics

**Build and push the gRPC server’s Docker image to Amazon ECR**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon ECR repository.</td>
<td>Sign in to the AWS Management Console, open the Amazon ECR console, and then create an Amazon ECR repository. For more information about this, see Creating a repository in the Amazon ECR documentation. Make sure that you record the Amazon ECR repository’s URL. You can also create an Amazon ECR repository with AWS CLI by running the following command: <code>aws ecr create-repository --repository-name helloworld-grpc</code></td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Build the Docker image.</td>
<td>Download the helloworld_grpc_alb file (attached) and open the directory. From the root of the directory contents, make sure that the Dockerfile exists and then run the following command to build the Docker image: <code>docker build -t &lt;amazon_ecr_repository_url&gt;:&lt;Tag&gt; .</code></td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
### Push the Docker image to Amazon ECR.

Run the following command to log in to the Amazon ECR repository:

```bash
aws ecr get-login-password --region us-east-1 --no-cli-auto-prompt | docker login --username AWS --password-stdin
```

0123456789.dkr.ecr.us-east-1.amazonaws.com

Push the Docker image to the Amazon ECR repository by running the following command:

```bash
docker push <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com/helloworld-grpc:1.0
```

**Important:** Make sure that you replace `<your_aws_account_id>` with your AWS account ID.

---

**Deploy the Kubernetes manifests to the Amazon EKS cluster**

### Modify the values in the Kubernetes manifest file.

Modify the grpc-sample.yaml Kubernetes manifest file in the Kubernetes directory according to your requirements. You must modify the annotations and host name in the ingress resource. For a sample ingress resource, see the Additional information section. For more information about ingress annotations, see Ingress annotations in the Kubernetes documentation.

In the Kubernetes deployment resource, change the deployment resource's image to the uniform resource identifier (URI) for the Amazon ECR repository that you pushed the Docker image to. For a sample

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push the Docker image to Amazon ECR.</td>
<td>Run the following command to log in to the Amazon ECR repository:</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
|                                                                        | ```bash
|                                                                        |   aws ecr get-login-password --region us-east-1 --no-cli-auto-prompt | docker login --username AWS --password-stdin
|                                                                        |   0123456789.dkr.ecr.us-east-1.amazonaws.com
| Push the Docker image to the Amazon ECR repository by running the following command: | ```bash
docker push <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com/helloworld-grpc:1.0
|                                                                        | **Important:** Make sure that you replace `<your_aws_account_id>` with your AWS account ID.                                                                                                           |                       |
| Modify the values in the Kubernetes manifest file.                  | Modify the grpc-sample.yaml Kubernetes manifest file in the Kubernetes directory according to your requirements. You must modify the annotations and host name in the ingress resource. For a sample ingress resource, see the Additional information section. For more information about ingress annotations, see Ingress annotations in the Kubernetes documentation. |
|                                                                        | In the Kubernetes deployment resource, change the deployment resource's image to the uniform resource identifier (URI) for the Amazon ECR repository that you pushed the Docker image to. For a sample |
|                                                                        | DevOps engineer                                                                                                                             |                       |
### Task 1: Deploy the Kubernetes manifest file

**Description:**
- Deploy the `grpc-sample.yaml` file to the Amazon EKS cluster by running the following `kubectl` command:
  ```yaml
  kubectl apply -f ./kubernetes/grpc-sample.yaml
  ```

**Skills required:** DevOps engineer

### Task 2: Create the DNS record for the Application Load Balancer's FQDN

**Description:**
- Record the FQDN for the Application Load Balancer.
- Run the following `kubectl` command to describe the Kubernetes ingress resource that manages the Application Load Balancer:
  ```bash
  kubectl get ingress -n grpcserver
  ```
  
  A sample output is provided in the *Additional information* section. In the output, the `HOSTS` field displays the DNS host name that the SSL certificates were created for.
  - Record the Application Load Balancer's fully qualified domain name (FQDN) from the `Address` field.
  - Create a DNS record pointing to the Application Load Balancer's FQDN. If your DNS provider is Amazon Route 53, you can create an alias record that points to the Application Load Balancer's FQDN. For more information about this, see [Choosing between alias and non-alias records](#) in the Route 53 documentation.
## Test the solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the gRPC server.</td>
<td>Use gRPCurl to test the endpoint by running the following command:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td><code>grpcurl grpc.example.com:443</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>list grpc.reflection.v1alpha.ServerReflection helloworld.helloworld</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Replace <code>grpc.example.com</code> with your DNS name.</td>
<td></td>
</tr>
<tr>
<td>Test the gRPC server using a gRPC client.</td>
<td>In the helloworld_client_ssl.py sample gRPC client, replace the host name from <code>grpc.example.com</code> with the host name used for the gRPC server.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>The following code sample shows the response from the gRPC server for the client's request:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>python ./app/helloworld_client_ssl.py</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>message: &quot;Hello to gRPC server from Client&quot;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>message: &quot;Thanks for talking to gRPC server!! Welcome to hello world. Received message is \&quot;Hello to gRPC server from Client\&quot;&quot;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>received: true</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This shows that the client can talk to the server and that the connection is successful.</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources
- Network load balancing on Amazon EKS
- Target groups for your Application Load Balancers

### Additional information

**Sample ingress resource:**
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
    annotations:
      alb.ingress.kubernetes.io/actions.ssl-redirect: '{"Type": "redirect", "RedirectConfig":
        {
          "Protocol": "HTTPS", "Port": "443", "StatusCode": "HTTP_301"}}'
      alb.ingress.kubernetes.io/backend-protocol-version: GRPC
      alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS":443}]'
      alb.ingress.kubernetes.io/scheme: internet-facing
      alb.ingress.kubernetes.io/target-type: ip
      kubernetes.io/ingress.class: alb
      alb.ingress.kubernetes.io/subnets: subnet-XXXXXX,subnet-YYYYYY  # <----- Replace with the Public Subnets within the VPC that hosts the Amazon EKS cluster
      alb.ingress.kubernetes.io/security-groups: sg-ZZZZZZZZ.  # <----- Replace with a security group ID from the VPC where the Amazon EKS cluster is deployed. Make sure that this security group is allowed to access the nodes
      alb.ingress.kubernetes.io/healthcheck-path: /
      alb.ingress.kubernetes.io/healthcheck-protocol: HTTP
      alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-west-2:xxxxx:certificate/xxxxxxx  #<----- Replace with the ACM certificate ARN for the DNS hostname
labels:
    app: grpcserver
    environment: dev
    name: grpcserver
    namespace: grpcserver
spec:
    rules:
    - host: grpc.example.com  # <----- Replace as required by your host name that the SSL certificate is available for in ACM
      http:
        paths:
          - backend:
              serviceName: ssl-redirect
              servicePort: use-annotation
              path: /*
          - backend:
              serviceName: grpcserver
              servicePort: 9000

Sample deployment resource:

apiVersion: apps/v1
kind: Deployment
metadata:
    name: grpcserver
    namespace: grpcserver
spec:
    selector:
        matchLabels:
            app: grpcserver
    replicas: 1
    template:
        metadata:
            labels:
                app: grpcserver
        spec:
            containers:
            - name: grpc-demo
              image: 00123456789.dkr.ecr.us-east-1.amazonaws.com/helloworld-grpc:1.0  #<------- Change to the URI that the Docker image is pushed to
              imagePullPolicy: Always
              ports:
              - name: grpc-api
                containerPort: 9000
Generate a static outbound IP address using a Lambda function, Amazon VPC, and a serverless architecture

Created by Thomas Scott (AWS)

Environment: Production  Technologies: Containers & microservices; Software development & testing  AWS services: AWS Lambda

Summary

This pattern describes how to generate a static outbound IP address in the Amazon Web Services (AWS) Cloud by using a serverless architecture. Your organization can benefit from this approach if it wants to send files to a separate business entity by using Secure File Transfer Protocol (SFTP). This means that the business entity must have access to an IP address that allows files through its firewall.

The pattern’s approach helps you create an AWS Lambda function that uses an Elastic IP address as the outbound IP address. By following the steps in this pattern, you can create a Lambda function and a virtual private cloud (VPC) that routes outbound traffic through an internet gateway with a static IP address. To use the static IP address, you attach the Lambda function to the VPC and its subnets.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Identity and Access Management (IAM) permissions to create and deploy a Lambda function, and to create a VPC and its subnets. For more information about this, see Execution role and user permissions in the AWS Lambda documentation.
If you plan to use infrastructure as code (IaC) to implement this pattern's approach, you need an integrated development environment (IDE) such as AWS Cloud9. For more information about this, see What is AWS Cloud9? in the AWS Cloud9 documentation.

**Architecture**

The following diagram shows the serverless architecture for this pattern.

The diagram shows the following workflow:

1. Outbound traffic leaves NAT gateway 1 in Public subnet 1.
2. Outbound traffic leaves NAT gateway 2 in Public subnet 2.
3. The Lambda function can run in Private subnet 1 or Private subnet 2.
4. Private subnet 1 and Private subnet 2 route traffic to the NAT gateways in the public subnets.
5. The NAT gateways send outbound traffic to the internet gateway from the public subnets.
6. Outbound data is transferred from the internet gateway to the external server.

**Technology stack**

- Lambda
- Amazon Virtual Private Cloud (Amazon VPC)

**Automation and scale**

You can ensure high availability (HA) by using two public and two private subnets in different Availability Zones. Even if one Availability Zone becomes unavailable, the pattern's solution continues to work.
Tools

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Create a new VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new VPC.</td>
<td>Sign in to the AWS Management Console, open the Amazon VPC console, and then create a VPC named Lambda VPC that has 10.0.0.0/25 as the IPv4 CIDR range. For more information about creating a VPC, see Getting started with Amazon VPC in the Amazon VPC documentation.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>

Create two public subnets

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the first public subnet. | 1. On the Amazon VPC console, choose **Subnets** and then choose **Create Subnet**.  
2. For **Name tag**, enter **public-one**.  
3. For **VPC**, choose **Lambda VPC**.  
4. Choose an **Availability Zone** and record it.  
5. For **IPv4 CIDR block**, enter 10.0.0.0/28 and then choose **Create subnet**. | AWS administrator       |

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the second public subnet. | 1. On the Amazon VPC console, choose **Subnets** and then choose **Create Subnet**.  
2. For **Name tag**, enter **public-two**. | AWS administrator       |
### Create two private subnets

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the first private subnet.</td>
<td>1. On the Amazon VPC console, choose Subnets and then choose Create Subnet.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>2. For Name tag, enter private-one.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. For VPC, choose Lambda VPC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Choose the Availability Zone that contains the public-one subnet that you created earlier.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. For IPv4 CIDR block, enter 10.0.0.32/28 and then choose Create subnet.</td>
<td></td>
</tr>
<tr>
<td>Create the second private subnet.</td>
<td>1. On the Amazon VPC console, choose Subnets and then choose Create Subnet.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>2. For Name tag, enter private-two.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. For VPC, choose Lambda VPC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Choose the same Availability Zone that contains the public-two subnet that you created earlier.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. For IPv4 CIDR block, enter 10.0.0.64/28 and then choose Create subnet.</td>
<td></td>
</tr>
</tbody>
</table>
## Create two Elastic IP addresses for your NAT gateways

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the first Elastic IP address. | 1. On the Amazon VPC console, choose **Elastic IPs** and then choose **Allocate new address**.  
2. Choose **Allocate** and record the **Allocation ID** for your newly created Elastic IP address.  
**Note**: This Elastic IP address is used for your first NAT gateway. | AWS administrator |
| Create the second Elastic IP address. | 1. On the Amazon VPC console, choose **Elastic IPs** and then choose **Allocate new address**.  
2. Choose **Allocate** and record the **Allocation ID** for this second Elastic IP address.  
**Note**: This Elastic IP address is used for your second NAT gateway. | AWS administrator |

## Create an internet gateway

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an internet gateway. | 1. On the Amazon VPC console, choose **Internet Gateways** and then choose **Create internet gateway**.  
2. Enter Lambda internet gateway as the name and then choose **Create internet gateway**. Make sure that you record the internet gateway ID. | AWS administrator |
| Attach the internet gateway to the VPC. | Select the internet gateway that you just created, and then choose **Actions, Attach to VPC**. | AWS administrator |
Create two NAT gateways

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the first NAT gateway. | 1. On the Amazon VPC console, choose **NAT Gateways** and then choose **Create NAT Gateway**.  
2. Enter nat-one as the NAT gateway name.  
3. Choose public-one as the subnet to create the NAT gateway in.  
4. For **Connectivity type**, choose Public.  
5. For **Elastic IP allocation ID**, choose the first Elastic IP address that you created earlier and associate it with the NAT gateway.  
6. Choose **Create NAT gateway**. | AWS administrator |
| Create the second NAT gateway. | 1. On the Amazon VPC console, choose **NAT Gateways** and then choose **Create NAT Gateway**.  
2. Enter nat-two as the NAT gateway name.  
3. Choose public-two as the subnet to create the NAT gateway in.  
4. For **Connectivity type**, choose Public.  
5. For **Elastic IP allocation ID**, choose the second Elastic IP address that you created earlier and associate it with the NAT gateway.  
6. Choose **Create NAT gateway**. | AWS administrator |

Create route tables for your public and private subnets

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the route table for the public-one subnet. | 1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**.  
2. Enter public-one-subnet as the route table name and then choose **Create route table**. | AWS administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>Choose the public-one-subnet route table, choose Edit routes, and then choose Add route.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>4.</td>
<td>Specify 0.0.0.0 in the Destination box and then choose the internet gateway ID in the Target list.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>On the Subnet associations tab, choose Edit subnet associations, choose the public-one subnet with the 10.0.0.0/28 CIDR range, and then choose Save associations.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Choose Save Changes.</td>
<td></td>
</tr>
<tr>
<td>Create the route table for the public-two subnet.</td>
<td>1. On the Amazon VPC console, choose Route Tables and then choose Create route table.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Enter public-two-subnet as the route table name and then choose Create route table.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Choose the public-two-subnet route table, choose Edit routes, and then choose Add route.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Specify 0.0.0.0 in the Destination box and then choose the internet gateway ID in the Target list.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. On the Subnet associations tab, choose Edit subnet associations, choose the public-two subnet with the 10.0.0.16/28 CIDR range, and then choose Save associations.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Choose Save Changes.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Create the route table for the private-one subnet. | 1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**.  
2. Enter `private-one-subnet` as the route table name and then choose **Create route table**.  
3. Choose the `private-one-subnet` route table, choose **Edit routes**, and then choose **Add route**.  
4. Specify `0.0.0.0` in the **Destination** box and then choose the NAT gateway in the public-one subnet in the **Target** list.  
5. On the **Subnet associations** tab, choose **Edit subnet associations**, choose the private-one subnet with the `10.0.0.32/28` CIDR range, and then choose **Save associations**.  
6. Choose **Save Changes**. | AWS administrator |

| Create the route table for the private-two subnet. | 1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**.  
2. Enter `private-two-subnet` as the route table name and then choose **Create route table**.  
3. Choose the `private-two-subnet` route table, choose **Edit routes**, and then choose **Add route**.  
4. Specify `0.0.0.0` in the **Destination** box and then choose the NAT gateway in the public-two subnet in the **Target** list.  
5. On the **Subnet associations** tab, choose **Edit subnet associations**, choose the private-two subnet with the `10.0.0.64/28` CIDR range, and then choose **Save associations**.  
6. Choose **Save Changes**. | AWS administrator |
Create the Lambda function, add it to the VPC, and test the solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a new Lambda function. | 1. Open the AWS Lambda console and choose Create function.  
2. Under Basic information, enter Lambda test under Function name and then choose the language of your choice under Runtime.  
3. Choose Create function. | AWS administrator |
| Add the Lambda function to your VPC. | 1. On the AWS Lambda console, choose Functions and then choose the function that you created earlier.  
2. Choose Configuration and then choose VPC.  
3. Choose Edit and then choose Lambda VPC and both private subnets.  
4. Choose Default security group for testing purposes and then choose Save. | AWS administrator |
| Write code to call an external service. | 1. In the programming language of your choice, write code to call an external service that returns your IP address.  
2. Verify that the returned IP address matches one of your Elastic IP addresses. | AWS administrator |

Related resources

- Configuring a Lambda function to access resources in a VPC

Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet

Created by Mahendra Siddappa (AWS)

Environment: PoC or pilot  
Technologies: Containers & microservices; DevOps; Infrastructure  
AWS services: Amazon EKS; AWS Systems Manager
Summary

Note, September 2021: The latest Amazon EKS optimized AMIs install SSM Agent automatically. For more information, see the release notes for the June 2021 AMIs.

In Amazon Elastic Kubernetes Service (Amazon EKS), because of security guidelines, worker nodes don't have Secure Shell (SSH) key pairs attached to them. This pattern shows how you can use the Kubernetes DaemonSet resource type to install AWS Systems Manager Agent (SSM Agent) on all worker nodes, instead of installing it manually or replacing the Amazon Machine Image (AMI) for the nodes. DaemonSet uses a CronJob on the worker node to schedule the installation of SSM Agent. You can also use this pattern to install other packages on worker nodes.

When you're troubleshooting issues in the cluster, installing SSM Agent on demand enables you to establish an SSH session with the worker node, to collect logs or to look into instance configuration, without SSH key pairs.

Prerequisites and limitations

Prerequisites

- An existing Amazon EKS cluster with Amazon Elastic Compute Cloud (Amazon EC2) worker nodes.
- Container instances should have the required permissions to communicate with the SSM service. The AWS Identity and Access Management (IAM) managed role AmazonSSMManagedInstanceIdCore provides the required permissions for SSM Agent to run on EC2 instances. For more information, see the AWS Systems Manager documentation.

Limitations

- This pattern isn't applicable to AWS Fargate, because DaemonSets aren't supported on the Fargate platform.
- This pattern applies only to Linux-based worker nodes.
- The DaemonSet pods run in privileged mode. If the Amazon EKS cluster has a webhook that blocks pods in privileged mode, the SSM Agent will not be installed.

Architecture
Tools

Tools

- **kubectl** – kubectl is a command-line utility that is used to interact with an Amazon EKS cluster. This pattern uses kubectl to deploy a DaemonSet on the Amazon EKS cluster, which will install SSM Agent on all worker nodes.

- **Amazon EKS** – Amazon EKS makes it easy for you to run Kubernetes on AWS without having to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

- **Session Manager** – AWS Systems Manager Session Manager lets you manage your EC2 instances, on-premises instances, and virtual machines (VMs) through an interactive, one-click, browser-based shell or through the AWS Command Line Interface (AWS CLI).

Code

Use the following code to create a DaemonSet configuration file that will install SSM Agent on the Amazon EKS cluster.

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    k8s-app: ssm-installer
dapiVersion: apps/v1
dmetadata:
  labels:
    k8s-app: ssm-installer
dapiVersion: apps/v1
dmetadata:
  labels:
    k8s-app: ssm-installer
dapiVersion: apps/v1
dmetadata:
  labels:
    k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
```
### Epics

**Set up kubectl**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install and configure kubectl to access the EKS cluster.</td>
<td>If kubectl isn't already installed and configured to access the Amazon EKS cluster, see <a href="#">Installing kubectl</a> in the Amazon EKS documentation.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

**Deploy the DaemonSet**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the DaemonSet configuration file.</td>
<td>Use the code in the Code section to create a DaemonSet configuration file called ssm_daemonset.yaml, which will be deployed to the Amazon EKS cluster. The pod launched by DaemonSet will have a main container and an init container. Main container will just have sleep command and Init container will have command section which creates a cron job file to install SSM Agent at the path /etc/cron.d/. The cron job runs only once, and the file it creates is automatically deleted after the cron job is complete. Once the init container has finished the main container will wait for 60 minutes before exiting. After 60 minutes new pod is launched which will either install SSM agent if its missing or update to the latest version. If required, sleep command can be modified to make the pod restart once a day or to run more often.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Deploy the DaemonSet on the Amazon EKS cluster.</td>
<td>To deploy the DaemonSet configuration file you created in the previous step on the Amazon</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands

**Summary**

This pattern provides code samples and steps to install the AWS Systems Manager Agent (SSM Agent) and Amazon CloudWatch agent on Amazon Elastic Kubernetes Service (Amazon EKS) worker nodes in the Amazon Web Services (AWS) Cloud during Amazon EKS cluster creation. You can install the SSM Agent and CloudWatch agent by using the `preBootstrapCommands` property from the `eksctl config file schema` (Weaveworks documentation). Then, you can use the SSM Agent to connect to your worker nodes without an Amazon Elastic Compute Cloud (Amazon EC2) key pair. Additionally, you can use the CloudWatch agent to monitor memory and disk utilization on your Amazon EKS worker nodes.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- The `eksctl` command line utility, installed and configured on macOS, Linux, or Windows
- The `kubectl` command line utility, installed and configured on macOS, Linux, or Windows
Limitations

- We recommend that you avoid adding long-running scripts to the `preBootstrapCommands` property, because this delays the node from joining the Amazon EKS cluster during scaling activities. We recommend that you create a custom Amazon Machine Image (AMI) instead.
- This pattern applies to Amazon EC2 Linux instances only.

Architecture

The following diagram shows an example of the SSM Agent connecting to Amazon EKS worker nodes by using the `eksctl` config file schema.

![Diagram showing SSM Agent connecting to Amazon EKS worker nodes](image)

The diagram shows the following workflow:

1. The user creates an Amazon EKS cluster by using the `eksctl` configuration file with the `preBootstrapCommands` property, which installs the SSM Agent and CloudWatch agent.
2. Any new instances that join the cluster later due to scaling activities get created with the pre-installed SSM Agent and CloudWatch agent.
3. The user connects to Amazon EC2 by using the SSM Agent and then monitors memory and disk utilization by using the CloudWatch agent.

Tools

- **Amazon CloudWatch** – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables that you can measure for your resources and applications.
- **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane.
- **AWS Systems Manager Parameter Store** – Parameter Store, a capability of AWS Systems Manager, provides secure, hierarchical storage for configuration data management and secrets management.
• **AWS Systems Manager Session Manager** – AWS Systems Manager Session Manager lets you manage your EC2 instances, on-premises instances, and virtual machines through an interactive, one-click, browser-based shell or through the AWS Command Line Interface (AWS CLI).

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

• **kubectl** – kubectl is a command-line utility for communicating with the cluster API server.

# Epics

## Create an Amazon EKS cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Store the CloudWatch agent configuration file. | Store the CloudWatch agent configuration file in the AWS Systems Manager Parameter Store in the AWS Region where you want to create your Amazon EKS cluster. To do this, create a parameter in AWS Systems Manager Parameter Store and note the name of the parameter (for example, AmazonCloudwatch-linux). In the following example, the CloudWatch agent is configured to monitor disk and memory utilization on Amazon Linux instances:  

```json
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "cwagent"
    },
    "metrics": {
        "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}",
            "metrics_collected": {
                "disk": {
                    "measurement": [
``` | DevOps engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;used_percent&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;metrics_collection_interval&quot;: 60,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;resources&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;*&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>],</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;mem&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;measurement&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;mem_used_percent&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;metrics_collection_interval&quot;: 60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Create the eksctl configuration file and cluster. | Create an eksctl configuration file that includes the CloudWatch agent and SSM Agent installation steps:  
```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: test
  region: us-east-2
  version: "1.19"
managedNodeGroups:
  - name: test
    minSize: 2
    maxSize: 4
    desiredCapacity: 2
    volumeSize: 20
    instanceType: t3.medium
    preBootstrapCommands:
      - sudo yum install amazon-ssm-agent -y
      - sudo systemctl enable amazon-ssm-agent
      - sudo systemctl start amazon-ssm-agent
      - sudo yum install amazon-cloudwatch-agent -y
      - sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c
      - ssm:AmazonCloudwatch-linux
      - iam:
        - attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
Next, create a cluster by running the following eksctl command:  
```eksctl create cluster -f cluster.yaml

AWS DevOps
Verify that the SSM Agent and CloudWatch agent work

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the SSM Agent.</td>
<td>Use SSH to connect to your Amazon EKS cluster nodes by using any of the methods covered in Start a session from the AWS Systems Manager documentation.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Test the CloudWatch agent.</td>
<td>Use the CloudWatch console to validate the CloudWatch agent: 1. Sign in to the AWS Management Console and open the CloudWatch console. 2. On the navigation pane, expand Metrics and then choose All metrics. 3. In the search box on the Browse tab, enter and then choose CWAgent metrics to see the memory and disk metrics.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

Related resources

- Installing and running the CloudWatch agent on your servers [Amazon CloudWatch documentation](#)  
- Create a Systems Manager parameter (console) [AWS Systems Manager documentation](#)  
- Create the CloudWatch agent configuration file [Amazon CloudWatch documentation](#)  
- Starting a session (AWS CLI) [AWS Systems Manager documentation](#)  
- Starting a session (Amazon EC2 console) [AWS Systems Manager documentation](#)  

Additional information

- In the last line of the `preBootstrapCommands` property, `AmazonCloudwatch-linux` is the name of the parameter created in AWS System Manager Parameter Store. You must include `AmazonCloudwatch-linux` in Parameter Store in the same AWS Region where you created the Amazon EKS cluster. You can also specify a file path, but we recommend using Systems Manager for easier automation and reusability.
- If you use `preBootstrapCommands` in the `eksctl` configuration file, you see two launch templates in the AWS Management Console. The first launch template includes the commands specified in `preBootstrapCommands`. The second template includes the commands specified in `preBootstrapCommands` and default Amazon EKS user data, which is required to get the nodes to join the cluster. The node group’s Auto Scaling group uses this user data to spin up new instances.
- If you use the `iam` attribute in the `eksctl` configuration file, you must list the default Amazon EKS policies with any additional policies required in your attached AWS Identity and Access Management (IAM) policies. In the code snippet from the Create the eksctl configuration file and cluster step, `CloudWatchAgentServerPolicy` and `AmazonSSMManagedInstanceCore`
are additional policies added to make sure that the CloudWatch agent and SSM Agent work as expected. The `AmazonEKSWorkerNodePolicy`, `AmazonEKS_CNI_Policy`, and `AmazonEC2ContainerRegistryReadOnly` policies are mandatory policies required for the Amazon EKS cluster to function correctly.

Optimize AWS App2Container generated Docker images

*Created by Varun Sharma (AWS)*

| Environment: PoC or pilot | Technologies: Containers & microservices; Modernization; DevOps | AWS services: Amazon ECS |

Summary

AWS App2Container is a command line tool that helps transform existing applications running on premises or on virtual machines into containers, without needing code changes.

Based on application type, App2Container takes a conservative approach to identify dependencies. For process mode, all non-system files on the application server are included in the container image. In such cases, a fairly large image might be generated.

This pattern provides an approach for optimizing the container images generated by App2Container. It is applicable for all Java applications discovered by App2Container in process mode. The workflow defined in the pattern is designed to be run on the application server.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Java application running on an application server on a Linux server
- App2Container installed and set up, with all prerequisites met, on the Linux server

Architecture

Source technology stack

- A Java application running on a Linux server

Target technology stack

- A Docker image generated by App2Container

Target architecture flow
1. Discover the applications that are running on the application server, and analyze the applications.
2. Containerize the applications.
3. Evaluate the size of the Docker image. If the image is too large, continue to step 4.
4. Use the shell script (attached) to identify large files.
5. Update the appExcludedFiles and appSpecificFiles lists in the analysis.json file.

Tools

Tools

- AWS App2Container – AWS App2Container (A2C) is a command line tool to help you lift and shift applications that run in your on-premises data center or on virtual machines, so that they run in containers that are managed by Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

Code

The optimizeImage.sh shell script and an example analysis.json file are attached.

The optimizeImage.sh file is a utility script for reviewing the contents of the App2Container generated file, ContainerFiles.tar. The review identifies files or subdirectories that are large and can be excluded. The script is a wrapper for the following tar command.

```
tar -Ptfv <path>|tr -s ' '|cut -d ' ' -f3,6| awk '$2 ~/<filetype>/'| awk '$2 ~/^<toplevel>/ '|cut -f1-<depth> -d'/|awk '{ if ($1>=$size) arr[$2]+=$1 } END { for (key in arr) { printf("%-50s%-50s\n", key, arr[key]) } } ' |sort -k2 -nr
```
In the tar command, the script uses the following values:

<table>
<thead>
<tr>
<th>path</th>
<th>The path to ContainerFiles.tar</th>
</tr>
</thead>
<tbody>
<tr>
<td>filetype</td>
<td>The file type to match</td>
</tr>
<tr>
<td>toplevel</td>
<td>The top-level directory to match</td>
</tr>
<tr>
<td>depth</td>
<td>The depth of the absolute path</td>
</tr>
<tr>
<td>size</td>
<td>The size for each file</td>
</tr>
</tbody>
</table>

The script does the following:

1. It uses tar -Ptvf to list the files without extracting them.
2. It filters the files by file type, starting with the top-level directory.
3. Based on the depth, it generates the absolute path as an index.
4. Based on the index and stores, it provides the total size of the subdirectory.
5. It prints the size of the subdirectory.

You can also replace the values manually in the tar command.

**Epics**

**Discover, analyze, and containerize applications**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover the on-premises Java applications.</td>
<td>To discover all applications running on the application server, run the following command.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td><strong>sudo app2container inventory</strong></td>
<td></td>
</tr>
<tr>
<td>Analyze the discovered applications.</td>
<td>To analyze each application by using the application-id that was obtained in the inventory stage, run the following command.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td><strong>sudo app2container analyze --application-id &lt;java-app-id&gt;</strong></td>
<td></td>
</tr>
<tr>
<td>Containerize the analyzed applications.</td>
<td>To containerize an application, run the following command.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td><strong>sudo app2container containerize --application-id &lt;application-id&gt;</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The command generates the Docker image along with a tar bundle in the workspace location.</strong></td>
<td><strong>If the Docker image is too large, proceed to the next step.</strong></td>
<td><strong>AWS DevOps</strong></td>
</tr>
</tbody>
</table>

### Identify appExcludedFiles and appSpecificFiles from the App2Container extracted tar file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identify the Artifacts tar file size.</strong></td>
<td>Identify the Artifacts.tar file in {workspace}/{java-app-id}/Artifacts, where workspace is the App2Container workspace and java-app-id is the application ID.</td>
<td><strong>AWS DevOps</strong></td>
</tr>
</tbody>
</table>

```
./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 0 -t / -v
```

This is the total size of the tar file after optimization.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>List the subdirectories under the / directory and their sizes.</strong></td>
<td>To identify the sizes of the major subdirectories under the / top-level directory, run the following command.</td>
<td><strong>AWS DevOps</strong></td>
</tr>
</tbody>
</table>

```
./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 1 -t / -s 1000000 -v
```

```
/var 554144711
/usr 2097300819
/tmp 18579660
/root 43645397
/opt 222320534
/home 65212518
/etc 11357677
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Identify large subdirectories under the / directory. | For each major subdirectory that is listed in the previous command, identify the sizes of its subdirectories. Use `-d` to increase the depth and `-t` to indicate the top-level directory.  

For example, use `/var` as the top-level directory. Under `/var`, identify all the large subdirectories and their sizes.  

```bash
./optimizeImage.sh -p / {workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 2 -t /var -s 1000000 -v
```

Repeat this process for each subdirectory listed in the previous step (for example, `/usr`, `/tmp`, `/opt`, and `/home`). | AWS DevOps |
| Analyze the large folder in each subdirectory under the / directory. | For each subdirectory that is listed in the previous step, identify any folders that are required to run the application.  

For example, using the subdirectories from the previous step, list all the subdirectories in the `/var` directory and their sizes. Identify any subdirectories that are needed by the application.  

```
/var/tmp
  237285851
/var/lib
  24489984
/var/cache
  237285851
```

To exclude subdirectories that are not needed by the application, in the `analysis.json` file, add those subdirectories to the `appExcludedFiles` section under `containerParameters`.  

An example `analysis.json` file is attached. | AWS DevOps |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identify files that are needed from the appExcludes list.</strong></td>
<td>For each subdirectory that is added to the appExcludes list, identify any files in that subdirectory that are required by the application. In the analysis.json file, add the specific files or subdirectories in the appSpecificFiles section under containerParameters. For example, if the /usr/lib directory is added to the exclude list, but /usr/lib/jvm is needed by the application, add /usr/lib/jvm to the appSpecificFiles section.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

**Extract and containerize the application again**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Containerize the analyzed application.</strong></td>
<td>To containerize the application, run the following command.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td><code>sudo app2container containerize --application-id &lt;application-id&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The command generates the Docker image along with a tar bundle in the workspace location.</td>
<td></td>
</tr>
<tr>
<td><strong>Identify the Artifacts tar file size.</strong></td>
<td>Identify the <code>ContainerFiles.tar</code> file in <code>{workspace}/{java-app-id}/Artifacts</code>, where workspace is the App2Container workspace and java-app-id is the application ID.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td><code>./optimizeImage.sh -p /{workspace}/{java-app-id}/Artifacts/ContainerFiles.tar -d 0 -t / -v</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is the total size of the tar file after optimization.</td>
<td></td>
</tr>
<tr>
<td><strong>Run the Docker image.</strong></td>
<td>To verify that the image starts without errors, run the</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
Replicate filtered Amazon ECR container images across accounts or Regions

Created by Abdal Garuba (AWS)

| Environment: Production | Technologies: Containers & microservices; DevOps | AWS services: AWS EC2 Container Registry; Amazon CloudWatch; AWS CodeBuild; AWS Identity and Access Management; AWS CLI |

Summary

This pattern describes how to replicate container images that are stored in Amazon Elastic Container Registry (Amazon ECR) across Amazon Web Services (AWS) accounts and AWS Regions, based on image tag patterns. The pattern uses Amazon CloudWatch Events to listen for push events for images that have a predefined, custom tag. A push event starts an AWS CodeBuild project and passes the image details to it. The CodeBuild project copies the images from the source Amazon ECR registry to the destination registry based on the details provided.

Amazon ECR supports cross-Region and cross-account replication for images. Both options replicate all new images that are pushed to the source registry. (For more information, see Private image replication in the Amazon ECR documentation.) However, there is no way to filter the images copied across AWS Regions or accounts based on any criteria.

This pattern copies images that have specific tags across accounts. For example, you can use this pattern to copy only production-ready, secure images to the production AWS account. In the development
account, after images are thoroughly tested, you can add a predefined tag to the secure images and use the steps in this pattern to copy the marked images to the production account.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account for source and destination Amazon ECR registries
- Administrative permissions for the tools used in this pattern
- Docker installed on your local machine for testing
- AWS Command Line Interface (AWS CLI), for authenticating into Amazon ECR

**Limitations**

- This pattern watches the push events of the source registry in only one AWS Region. You can deploy this pattern to other Regions to watch registries in those Regions.
- In this pattern, one Amazon CloudWatch Events rule listens for a single image tag pattern. If you want to check for multiple patterns, you can add events to listen for additional image tag patterns.

**Architecture**

**Target architecture**

Automation and scale

This pattern can be automated with an infrastructure as code (IaC) script and deployed at scale. To use AWS CloudFormation templates to deploy this pattern, download the attachment and follow the instructions in the Additional information (p. 246) section.

You can point multiple Amazon CloudWatch Events events (with different custom event patterns) to the same AWS CodeBuild project to replicate multiple image tag patterns, but you will need to update the secondary validation in the buildspec.yaml file (which is included in the attachment and in the Tools (p. 237) section) as follows to support multiple patterns.
if [[ ${IMAGE_TAG} != release-* ]]; then
...

Tools

Amazon services

- **IAM** – AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. In this pattern, you would need to create the cross-account IAM role that AWS CodeBuild will assume when pushing container images to the destination registry.
- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is a fully managed container registry that makes it easy to store, manage, share, and deploy your container images and artifacts anywhere. Image push actions to the source registry send system event details to the event bus that is picked up by Amazon CloudWatch Events.
- **AWS CodeBuild** – AWS CodeBuild is a fully managed continuous integration service that provides compute power to perform jobs such as compiling source code, running tests, and producing artifacts that are ready to be deployed. This pattern uses AWS CodeBuild to perform the copy action from the source Amazon ECR registry to the destination registry.
- **CloudWatch Events** – Amazon CloudWatch Events delivers a stream of system events that describe changes in AWS resources. This pattern uses rules to match Amazon ECR push actions with a specific image tag pattern.

Tools

- **Docker CLI** – Docker is a tool that makes it easier to create and manage containers. Containers pack an application and all its dependencies into one unit or package that can easily be deployed on any platform that supports the container runtime.

Code

You can implement this pattern in two ways:

- **Automated setup**: Deploy the two AWS CloudFormation templates provided in the attachment. For instructions, see the Additional information (p. 246) section.
- **Manual setup**: Follow the steps in the Epics (p. 238) section.

*Sample buildspec.yaml*

If you're using the CloudFormation templates that are provided with this pattern, the buildspec.yaml file is included in the CodeBuild resources.

```yaml
version: 0.2
env:
  shell: bash
phases:
  install:
    commands:
    - export CURRENT_ACCOUNT=$(echo ${CODEBUILD_BUILD_ARN} | cut -d':' -f5)
    - export CURRENT_ECR_REGISTRY=${CURRENT_ACCOUNT}.dkr.ecr.${AWS_REGION}.amazonaws.com
    - export DESTINATION_ECR_REGISTRY=${DESTINATION_ACCOUNT}.dkr.ecr.${DESTINATION_REGION}.amazonaws.com
  pre_build:
    on-failure: ABORT
    commands:
```

237
- echo "Validating Image Tag ${IMAGE_TAG}"
- 
if [[ ${IMAGE_TAG} != release-* ]]; then
    aws codebuild stop-build --id ${CODEBUILD_BUILD_ID}
sleep 60
    exit 1
fi
- aws ecr get-login-password --region ${AWS_REGION} | docker login -u AWS --password-stdin ${CURRENT_ECR_REGISTRY}
- docker pull ${CURRENT_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
build:
  commands:
  - echo "Assume cross-account role"
  - CREDENTIALS=$(aws sts assume-role --role-arn ${CROSS_ACCOUNT_ROLE_ARN} --role-session-name Rolesession)
  - export AWS_DEFAULT_REGION=${DESTINATION_REGION}
  - export AWS_ACCESS_KEY_ID=$(echo ${CREDENTIALS} | jq -r '.Credentials.AccessKeyId')
  - export AWS_SECRET_ACCESS_KEY=$(echo ${CREDENTIALS} | jq -r '.Credentials.SecretAccessKey')
  - export AWS_SESSION_TOKEN=$(echo ${CREDENTIALS} | jq -r '.Credentials.SessionToken')
  - echo "Logging into cross-account registry"
  - aws ecr get-login-password --region ${DESTINATION_REGION} | docker login -u AWS --password-stdin ${DESTINATION_ECR_REGISTRY}
  - echo "Check if Destination Repository exists, else create"
  - aws ecr describe-repositories --repository-names ${REPO_NAME} --region ${DESTINATION_REGION} \
  || aws ecr create-repository --repository-name ${REPO_NAME} --region ${DESTINATION_REGION}
  - echo "retag image and push to destination"
  - docker tag ${CURRENT_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG} ${DESTINATION_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
  - docker push ${DESTINATION_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}

Epics

Create IAM roles

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudWatch Events role.</td>
<td>In the source AWS account, create an IAM role for Amazon CloudWatch Events to assume. The role should have permissions to start a AWS CodeBuild project. To create the role by using the AWS CLI, follow the instructions in the IAM documentation. Example trust policy (trustpolicy.json):</td>
<td>AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer</td>
</tr>
</tbody>
</table>

```json
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
```
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{ &quot;Principal&quot;: { &quot;Service&quot;: &quot;ec2.amazonaws.com&quot; }, &quot;Action&quot;: &quot;sts:AssumeRole&quot; }</td>
<td></td>
</tr>
</tbody>
</table>

Example permission policy (permissionpolicy.json):

```json
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "<CodeBuild Project ARN>"
   }
}
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a CodeBuild role. | Create an IAM role for AWS CodeBuild to assume, by following the instructions in the IAM documentation. The role should have the following permissions:  
  • Permission to assume the destination cross-account role  
  • Permission to create log groups and log streams, and to put log events  
  • Read-only permissions to all Amazon ECR repositories, by adding the AmazonEC2ContainerRegistryReadOnly managed policy to the role  
  • Permission to stop CodeBuild | AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer |

Example trust policy (trustpolicy.json):

```json
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": 
    "sts:AssumeRole"
  }
  ]
}
```

Example permission policy (permissionpolicy.json):

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:Get*",
        "codebuild:List*",
        "codebuild:BatchGet*"
      ],
    }
  ]
}
```
### Attach the managed policy

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attach the managed policy AmazonEC2ContainerRegistryReadOnly to the CLI command as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>~$ aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly --role-name &lt;name of CodeBuild Role&gt;</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Create a cross-account role.</td>
<td>In the destination AWS account, create an IAM role for the AWS CodeBuild role for the source account to assume. The cross-account role should allow container images to create a new repository and upload container images to Amazon ECR. To create the IAM role by using the AWS CLI, follow the instructions in the IAM documentation. To allow the AWS CodeBuild project from the previous step, use the following trust policy:</td>
<td>AWS administrator, AWS DevOps, Cloud administrator, Cloud architect, DevOps engineer, AWS systems administrator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To allow the AWS CodeBuild project from the previous step to save images in the destination registry, use the following permission policy:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Create the CodeBuild project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a CodeBuild project. | Create a AWS CodeBuild project in the source account by following the instructions in the AWS CodeBuild documentation. The project should be in the same Region as the source registry. Configure the project as follows:  
  - Environment type: LINUX CONTAINER  
  - Service role: CodeBuild Role  
  - Privileged mode: true  
  - Environment image: aws/codebuild/standard:x.x (use the latest image available)  
  - Environment variables:  
    - CROSS_ACCOUNT_ROLE_ARN: The Amazon Resource Name (ARN) of the cross-account role  
    - DESTINATION_REGION: The name of the cross-account Region  
    - DESTINATION_ACCOUNT: The number of the destination account  
  - Build specifications: Use the buildspec.yaml file listed in the Tools (p. 237)section. | AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer |
## Create the event

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a CloudWatch Events rule. | Because the pattern uses the content filtering feature, you need to create the event by using Amazon EventBridge. Create the event and target by following the instructions in the EventBridge documentation, with a few modifications:  
  - For **Define pattern**, choose **Event Pattern**, and then choose **Custom pattern**.  
  - Copy the following custom events pattern sample code into the text box provided:  
    ```json
    {
      "source": ["aws.ecr"],
      "detail-type": ["ECR Image Action"],
      "detail": {
        "action-type": ["PUSH"],
        "result": ["SUCCESS"],
        "image-tag": [
          { "prefix": "release-" }
        ]
      }
    }
    ```  
  - For **Select targets**, choose the AWS CodeBuild project, and paste the ARN for the AWS CodeBuild project that you created in the previous epic.  
  - For **Configure Input**, choose **Input Transformer**.  
    - In the **Input Path** text box, paste:  
      ```json
      {"IMAGE_TAG":"$.detail.image-tag","REPO_NAME":"$.detail.repository-name"}
      ```  
    - In the **Input Template** text box, paste:  
      ```json
      {"environmentVariablesOverride": [  
        {
          "name": "IMAGE_TAG",
          "value": "<IMAGE_TAG>"  
        },  
        {
          "name": "REPO_NAME",
          "value": "<REPO_NAME>"  
        }
      ]}
      ```  
  - Choose **Use existing role**, and choose the name of the CloudWatch Events role you | AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer |
### Validate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authenticate with Amazon ECR.</strong></td>
<td>Authenticate to both source and destination registries by following the steps in the Amazon ECR documentation.</td>
<td>AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, DevOps engineer, Cloud architect</td>
</tr>
</tbody>
</table>

You can monitor the progress of the CodeBuild project in the [CodeBuild console](#).

After the CodeBuild project has completed successfully, sign in to the destination AWS account, open the Amazon ECR console and confirm that the image exists in the destination ECR registry.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test image replication.</strong></td>
<td>In your source account, push a container image to a new or existing Amazon ECR source repository with an image tag prefixed with <code>release-</code>. To push the image, follow the steps in the Amazon ECR documentation. You can monitor the progress of the CodeBuild project in the CodeBuild console. After the CodeBuild project has completed successfully, sign in to the destination AWS account, open the Amazon ECR console and confirm that the image exists in the destination ECR registry.</td>
<td>AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test image exclusion.</strong></td>
<td>In your source account, push a container image to a new or existing Amazon ECR source repository with an image tag that doesn’t have the custom prefix. Confirm that the CodeBuild project isn’t started, and that no container images appear in the destination registry.</td>
<td>AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer</td>
</tr>
</tbody>
</table>

### Related resources

- Getting started with CodeBuild
- Getting started with Amazon EventBridge
- Content-based filtering in Amazon EventBridge event patterns
- Delegate access across AWS accounts using IAM roles
Additional information

To automatically deploy the resources for this pattern, follow these steps:

1. Download the attachment and extract the two CloudFormation templates: `part-1-copy-tagged-images.yaml` and `part-2-destination-account-role.yaml`.
2. Log in to the AWS CloudFormation console, and deploy `part-1-copy-tagged-images.yaml` in the same AWS account and Region as the source Amazon ECR registries. Update the parameters as needed. The template deploys the following resources:
   - Amazon CloudWatch Events IAM role
   - AWS CodeBuild project IAM role
   - AWS CodeBuild project
   - AWS CloudWatch Events rule
3. Take note of the value of `SourceRoleName` in the Outputs tab. You will need this value in the next step.
4. Deploy the second CloudFormation template, `part-2-destination-account-role.yaml`, in the AWS account that you want to copy the Amazon ECR container images to. Update the parameters as needed. For the `SourceRoleName` parameter, specify the value from step 3. This template deploys the cross-account IAM role.
5. Validate image replication and exclusion, as described in the last step of the Epics (p. 238) section.

Attachments

attachment.zip

Rotate credentials without restarting containers

Created by Josh Joy (AWS)

| Environment: Production | Technologies: Containers & microservices; Databases; DevOps; Infrastructure; Security, identity, compliance; Management & governance | AWS services: Amazon ECS; Amazon Aurora; AWS Fargate; AWS Secrets Manager; Amazon VPC |

Summary

On the Amazon Web Services (AWS) Cloud, you can use AWS Secrets Manager to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to the Secrets Manager API, removing the need to hardcode sensitive information in plaintext.

If you're using containers for microservice workloads, you can securely store credentials in AWS Secrets Manager. To separate out configuration from code, these credentials are commonly injected into the container. However, it's important to rotate your credentials periodically and automatically. It's also important to support the ability to refresh credentials after revocation. At the same time, applications require the ability to rotate credentials while reducing any potential downstream availability impact.
This pattern describes how to rotate your secrets that are secured with AWS Secrets Manager within your containers without requiring your containers to restart. In addition, this pattern reduces the number of credential lookups to Secrets Manager by using the Secrets Manager client-side caching component. When you use the client-side caching component to refresh the credentials within the application, the container doesn't need to be restarted to fetch a rotated credential.

This approach works for Amazon Elastic Kubernetes Service (Amazon EKS) and Amazon Elastic Container Service (Amazon ECS).

Two scenarios are covered. In the single-user scenario, the database credential is refreshed on secret rotation by detecting the expired credential. The credential cache is instructed to refresh the secret, and then the application re-establishes the database connection. The client-side caching component caches the credential within the application and helps avoid reaching out to Secrets Manager for each credential lookup. The credential is rotated within the application without the need to force the credential refresh by restarting the container.

The second scenario rotates the secret by alternating between two users. Having two active users reduces the potential for downtime, because one user’s credentials are always active. Two-user credential rotation is helpful when you have a large deployment with clusters in which there might be a small propagation delay of credential updates.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- An application running in a container in Amazon EKS or Amazon ECS.
- Credentials stored in Secrets Manager, with rotation enabled.
- A second set of credentials stored in Secrets Manager, if deploying the two-user solution. Code examples can be found in the GitHub repo aws-secrets-manager-rotation-lambdas.
- An Amazon Aurora database.

**Limitations**

- This example is targeted for Python applications. For Java applications, you can use the Java client-side caching component or the JDBC client-side caching library for Secrets Manager.

**Architecture**

**Target architecture**

*Scenario 1 – Rotation of a credential for a single user*
In the first scenario, a single database credential is periodically rotated by Secrets Manager. The application container runs in Fargate. When the first database connection is established, the application container fetches the database credential for Aurora. The Secrets Manager caching component then caches the credential for future connection establishment. When rotation period has elapsed, the credential expires and the database returns an authentication error. The application then fetches the rotated credential, invalidates the cache, and updates the credential cache via the Secrets Manager client-side caching component.

In this scenario, there might be a minimal disruption while the credential is being rotated and stale connections are using the outdated credential. This concern can be addressed by using the two-user scenario.

**Scenario 2 – Rotation of credentials for two users**

In the second scenario, two database user credentials (Alice’s and Bob’s) are periodically rotated by Secrets Manager. The application container runs in a Fargate cluster. When the first database connection is established, the application container fetches the Aurora database credential for the first user (Alice). The Secrets Manager caching component then caches the credential for future connection establishment. Although there are two users and credentials, one only active credential is managed by Secrets Manager. The caching component periodically expires and fetches the latest credential. If the Secrets Manager rotation period is longer than the cache timeout, the caching component picks up the rotated credential for the second user (Bob). For example, if the cache expiration is measured in minutes and the rotation period is measured in days, the caching component fetches the new credential as part of its periodic cache refresh. In this way, the downtime is minimized because each user’s credential is active for one Secrets Manager rotation.

**Automation and scale**

You can use AWS CloudFormation to deploy this pattern by using infrastructure as code. This builds and creates the application container, creates the Fargate task, deploys the container into Fargate, and
sets up and configure Secrets Manager with Aurora. For step-by-step deployment instructions, see the readme file.

Tools

- **Secrets Manager** – AWS Secrets Manager enables the replacement of hardcoded credentials, including passwords, with an API call to Secrets Manager to retrieve the secret. Because Secrets Manager can automatically rotate the secret according to a schedule, you can replace long-term secrets with short-term ones, reducing the risk of compromise.
- **Docker** – Docker helps developers to pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

Code

*Example Python code*

This pattern uses the Python client-side caching component for Secrets Manager to retrieve the authentication credentials when establishing the database connection. The client-side caching component helps avoid reaching out to Secrets Manager each time.

Now, when the rotation period elapses, the cached credential will be expired, and connecting to the database will result in an authentication error. For MySQL, the authentication error code is 1045. This example uses Amazon Aurora for MySQL, though you could use another engine such as PostgreSQL. Upon the authentication error, the database connection exception handling code catches the error. It then informs the Secrets Manager client-side caching component to refresh the secret, then to reauthenticate and re-establish the database connection. If you are using PostgreSQL or another engine, you must look up the corresponding authentication error code.

The container application can now update the database password with the rotated password without restarting the container.

Place the following code in your application code that handles database connections. This example uses Django, and it subclasses the database backend with a database wrapper for connections. If you are using a different programming language or database connection library, see your database connection library to review how to subclass database connection retrieval.

```python
def get_new_connection(self, conn_params):
    try:
        logger.info("get connection")
        databasecredentials.get_conn_params_from_secrets_manager(conn_params)
        conn = super(DatabaseWrapper, self).get_new_connection(conn_params)
        return conn
    except MySQLdb.OperationalError as e:
        error_code = e.args[0]
        if error_code != 1045:
            raise e

        logger.info("Authentication error. Going to refresh secret and try again.")
        databasecredentials.refresh_now()
        databasecredentials.get_conn_params_from_secrets_manager(conn_params)
        conn = super(DatabaseWrapper, self).get_new_connection(conn_params)
        logger.info("Successfully refreshed secret and established new database connection.")
        return conn
```

AWS Prescriptive Guidance Patterns

Tools

AWS CloudFormation and Python code
Epics

Maintain application availability during credential rotation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the caching component.</td>
<td>Download and install the Secrets Manager client-side caching component for Python. For the download link, see the Related resources section.</td>
<td>Developer</td>
</tr>
<tr>
<td>Cache the working credential.</td>
<td>Use the Secrets Manager client-side caching component to cache the working credential locally.</td>
<td>Developer</td>
</tr>
<tr>
<td>Update the application code to refresh the credential upon the unauthorized error from the database connection.</td>
<td>Update the application code to use Secrets Manager to fetch and refresh database credentials. Add the logic to handle unauthorized error codes, and then fetch the newly rotated credential. See the Example Python code section.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

Related resources

Create a Secrets Manager secret

- Create keys in AWS KMS
- Create a secret in AWS Secrets Manager

Create an Amazon Aurora cluster

- Creating an Amazon RDS DB instance

Create the Amazon ECS components

- Create an Amazon ECS cluster
- Create a Docker image
- Create an Amazon ECR repository
- Authenticate Docker with Amazon ECR repository
- Push an image to an Amazon ECR repository
- Create Amazon ECS task definition
- Create an Amazon ECS service

Download and install the Secrets Manager client-side caching component
Run Amazon ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere

*Created by Akash Kumar (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Containers &amp; microservices; Modernization</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon ECS; Amazon WorkSpaces; AWS Directory Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

Amazon Elastic Container Service (Amazon ECS) Anywhere supports the deployment of Amazon ECS tasks in any environment, including Amazon Web Services (AWS) managed infrastructure and customer managed infrastructure. You can do this while using a fully AWS managed control plane that's running in the cloud and always up to date.

Enterprises often use Amazon WorkSpaces for developing container-based applications. This has required Amazon Elastic Compute Cloud (Amazon EC2) or AWS Fargate with an Amazon ECS cluster to test and run ECS tasks. Now, by using Amazon ECS Anywhere, you can add Amazon WorkSpaces as external instances directly to an ECS cluster, and you can run your tasks directly. This reduces your development time, because you can test your container with an ECS cluster locally on Amazon WorkSpaces. You can also save the cost of using EC2 or Fargate instances for testing your container applications.

This pattern showcases how to deploy ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere. It sets up the ECS cluster and uses AWS Directory Service Simple AD to launch the WorkSpaces. Then the example ECS task launches NGINX in the WorkSpaces.

**Prerequisites and limitations**

- An active AWS account
- AWS Command Line Interface (AWS CLI)
- AWS credentials configured on your machine

**Architecture**

**Target technology stack**

- A virtual private cloud (VPC)
- An Amazon ECS cluster
AWS Prescriptive Guidance Patterns

Tools

- Amazon WorkSpaces
- AWS Directory Service with Simple AD

Target architecture

The architecture includes the following services and resources:

- An ECS cluster with public and private subnets in a custom VPC
- Simple AD in the VPC to provide user access to Amazon WorkSpaces
- Amazon WorkSpaces provisioned in the VPC using Simple AD
- AWS Systems Manager activated for adding Amazon WorkSpaces as managed instances
- Using Amazon ECS and AWS Systems Manager Agent (SSM Agent), Amazon WorkSpaces added to Systems Manager and the ECS cluster
- An example ECS task to run in the WorkSpaces in the ECS cluster

Tools

- Simple AD – AWS Directory Service Simple Active Directory (Simple AD) is a standalone managed directory powered by a Samba 4 Active Directory Compatible Server. Simple AD provides a subset of the features offered by AWS Managed Microsoft AD, including the ability to manage user accounts and to securely connect to Amazon EC2 instances.
- Amazon ECS – Amazon Elastic Container Service is a highly scalable, fast container management service that for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon EC2 instances that you manage.
- AWS Identity and Access Management – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users,
security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS Systems Manager** – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

- **Amazon WorkSpaces** – Amazon WorkSpaces enables the provisioning of virtual, cloud-based Microsoft Windows or Amazon Linux desktops for your users, known as WorkSpaces. WorkSpaces eliminates the need to procure and deploy hardware or install complex software.

## Epics

### Set up the ECS cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create and configure the ECS cluster. | To create the ECS cluster, follow the instructions in the AWS documentation, including the following steps:
  - For Select cluster compatibility, choose Networking only, which will support an Amazon WorkSpace as an external instance to the ECS cluster.
  - Choose to create a new VPC. | Cloud architect |

### Launch Amazon WorkSpaces

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up Simple AD and launch Amazon WorkSpaces.</td>
<td>To provision a Simple AD directory for your newly created VPC and launch Amazon WorkSpaces, follow the instructions in the AWS documentation.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

### Set up AWS Systems Manager for a hybrid environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the attached scripts.</td>
<td>On your local machine, download the ssm-trust-policy.json and ssm-activation.json files that are in the Attachments section.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Add the IAM role.</td>
<td>Add environment variables based on your business requirements.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>export AWS_DEFAULT_REGION=${AWS_REGION_ID}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>export ROLE_NAME=${ECS_TASK_ROLE}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>export CLUSTER_NAME=${ECS_CLUSTER_NAME}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>export SERVICE_NAME=${ECS_CLUSTER_SERVICE_NAME}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws iam create-role --role-name $ROLE_NAME --assume-role-policy-document file://ssm-trust-policy.json</td>
<td></td>
</tr>
<tr>
<td>Add the AmazonSSMManagedInstanceCore policy to the IAM role.</td>
<td>Run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>/aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore</td>
<td></td>
</tr>
<tr>
<td>Add the AmazonEC2ContainerServiceforEC2Role policy to IAM role.</td>
<td>Run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role</td>
<td></td>
</tr>
<tr>
<td>Verify the IAM role.</td>
<td>To verify the IAM role, run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>aws iam list-attached-role-policies --role-name $ROLE_NAME</td>
<td></td>
</tr>
<tr>
<td>Activate Systems Manager.</td>
<td>Run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>aws ssm create-activation --iam-role $ROLE_NAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tee ssm-activation.json</td>
<td></td>
</tr>
</tbody>
</table>
## Add WorkSpaces to the ECS cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to your WorkSpaces.</td>
<td>To connect to and set up your Workspaces, follow the instructions in the <a href="https://aws.amazon.com">AWS documentation</a>.</td>
<td>App developer</td>
</tr>
<tr>
<td>Download the ecs-anywhere install script.</td>
<td>At the command prompt, run the following command.</td>
<td>App developer</td>
</tr>
<tr>
<td>Check integrity of the shell script.</td>
<td>(Optional) Run the following command.</td>
<td>App developer</td>
</tr>
<tr>
<td>Add an EPEL repository on Amazon Linux.</td>
<td>To add an Extra Packages for Enterprise Linux (EPEL) repository, run the command <code>sudo amazon-linux-extras install epel -y</code>.</td>
<td>App developer</td>
</tr>
<tr>
<td>Install Amazon ECS Anywhere.</td>
<td>To run the install script, use the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>sudo ./ecs-anywhere-install.sh --cluster $CLUSTER_NAME --activation-id $ACTIVATION_ID --activation-code $ACTIVATION_CODE --region $AWS_REGION</code></td>
<td></td>
</tr>
<tr>
<td>Check instance information from the ECS cluster.</td>
<td>To check the Systems Manager and ECS cluster instance information and validate that WorkSpaces were added on</td>
<td>App developer</td>
</tr>
</tbody>
</table>
Add an ECS task for the WorkSpaces

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a task execution IAM role.</td>
<td>Download task-execution-assume-role.json and external-task-definition.json from the Attachments section.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>On your local machine, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws iam --region $AWS_DEFAULT_REGION create-role --role-name $ECS_TASK_EXECUTION_ROLE --assume-role-policy-document file://task-execution-assume-role.json</td>
<td></td>
</tr>
<tr>
<td>Add the policy to the execution role.</td>
<td>Run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>aws iam --region $AWS_DEFAULT_REGION attach-role-policy --role-name $ECS_TASK_EXECUTION_ROLE --policy-name arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy</td>
<td></td>
</tr>
<tr>
<td>Create a task role.</td>
<td>Run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>aws iam --region $AWS_DEFAULT_REGION create-role --role-name $ECS_TASK_EXECUTION_ROLE --assume-role-policy-document file://task-execution-assume-role.json</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Register the task definition to the cluster.</td>
<td>On your local machine, run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td><code>aws ecs register-task-definition --cli-input-json file://external-task-definition.json</code></td>
<td></td>
</tr>
<tr>
<td>Run the task.</td>
<td>On your local machine, run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td><code>aws ecs run-task --cluster $CLUSTER_NAME --launch-type EXTERNAL --task-definition nginx</code></td>
<td></td>
</tr>
<tr>
<td>Validate the task running state.</td>
<td>To fetch the task ID, run the following command.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>`export TEST_TASKID=$(aws ecs list-tasks --cluster $CLUSTER_NAME</td>
<td>jq -r '.taskArns[0]')`</td>
</tr>
<tr>
<td></td>
<td>With the task ID, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>aws ecs describe-tasks --cluster $CLUSTER_NAME --tasks ${TEST_TASKID}</code></td>
<td></td>
</tr>
<tr>
<td>Verify the task on the WorkSpace.</td>
<td>To check that NGINX is running on the WorkSpace, run the command <code>curl http://localhost:8080</code>.</td>
<td>App developer</td>
</tr>
</tbody>
</table>

**Related resources**

- ECS clusters
- Setting up a hybrid environment
- Amazon WorkSpaces
- Simple AD

**Attachments**

attachment.zip
Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance

Created by Vijai Anand Ramalingam (AWS) and Sreelaxmi Pai (AWS)

**Environment:** PoC or pilot  
**Technologies:** Containers & microservices; Software development & testing; Websites & web apps  
**Workload:** Microsoft

### Summary

This pattern is for people who are starting to containerize their applications on the Amazon Web Services (AWS) Cloud. When you begin to containerize apps on cloud, usually there are no container orchestrating platforms set up. This pattern helps you quickly set up infrastructure on AWS to test your containerized applications without needing an elaborate container orchestrating infrastructure.

The first step in the modernization journey is to transform the application. If it's a legacy .NET Framework application, you must first change the runtime to ASP.NET Core. Then do the following:

- Create the Docker container image
- Run the Docker container using the built image
- Validate the application before deploying it on any container orchestration platform, such as Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

This pattern covers the build, run, and validate aspects of modern application development on an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance.

### Prerequisites and limitations

**Prerequisites**

- An active Amazon Web Services (AWS) account
- An AWS Identity and Access Management (IAM) role with sufficient access to create AWS resources for this pattern
- Visual Studio Community 2022 or later downloaded and installed
- A .NET Framework project modernized to ASP.NET Core
- A GitHub repository

**Product versions**

- Visual Studio Community 2022 or later
Architecture

Target architecture

This pattern uses an AWS CloudFormation template to create the highly available architecture shown in the following diagram. An Amazon EC2 Linux instance is launched in a private subnet. AWS Systems Manager Session Manager is used to access the private Amazon EC2 Linux instance and to test the API running in the Docker container.

Tools

AWS services

- AWS Command Line Interface – AWS Command Line Interface (AWS CLI) is an open source tool for interacting with AWS services through commands in your command line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console.

- AWS Management Console – The AWS Management Console is a web application that comprises and refers to a broad collection of service consoles for managing AWS resources. When you first sign in, you see the console home page. The home page provides access to each service console and offers a single place to access the information you need to perform your AWS related tasks.

- AWS Systems Manager Session Manager – Session Manager is a fully managed AWS Systems Manager capability. With Session Manager, you can manage your Amazon Elastic Compute Cloud (Amazon EC2) instances. Session Manager provides secure and auditable node management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys.

Other tools

- Visual Studio 2022 – Visual Studio 2022 is an integrated development environment (IDE).

- Docker – Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
Epics

Develop the ASP.NET Core web API

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an example ASP.NET Core web API using Visual Studio. | To create an example ASP.NET Core web API, do the following:  
2. Choose Create a new project.  
3. Select the ASP.NET Core Web API project template, and choose Next.  
4. For the project name, enter DemoNetCoreWebAPI, and choose Next.  
5. Choose Create.  
6. To run the project locally, press F5.  
7. Verify that the default WeatherForecast API endpoint is returning the results using Swagger.  
8. Open the command prompt, navigate to the .csproj project folder, and run the following commands to push the new web API to your GitHub repository. | App developer |

```bash
git add --all
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| git commit -m “Initial Version”  
git push | Create a Dockerfile.  
To create a Dockerfile, do one of the following:  
- Create the Dockerfile manually using the sample Dockerfile in the *Code* section. Based on the requirements, select the appropriate .NET base image. For information about .NET and ASP.NET Core related images, see *Docker hub*.  
- Create the Dockerfile using Visual Studio and *Docker Desktop*. In the solution explorer, right click on the project, choose *Add* -> *Docker Support*. For *Target OS*, select *Linux*. Ensure that the new Dockerfile is in the same path as the solution file (.sln).  
To push the changes to your GitHub repository, run the following command.  
```  
git add --all  
git commit -m “Dockerfile added”  
git push  
``` | App developer |

**Set up the Amazon EC2 Linux instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Set up the infrastructure. | Launch the *AWS CloudFormation template* to create the infrastructure, which includes the following:  
- A virtual private cloud (VPC), using the *AWS VPC Quick Start*, with two public and two private subnets spanning two Availability Zones.  
- The required IAM role to enable AWS Systems Manager. | App developer, AWS administrator, AWS DevOps |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>• In one of the private subnets, an Amazon Linux 2 demo instance with the latest SSM Agent. Although this instance doesn’t have any direct connectivity from the internet, it can be accessed securely by using AWS Systems Manager Session Manager without requiring a bastion host.</td>
<td></td>
<td>App developer</td>
</tr>
<tr>
<td>To learn more about accessing a private Amazon EC2 instance using Session Manager without requiring a bastion host, see the Toward a bastion-less world blog post.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log in to the Amazon EC2 Linux instance.</td>
<td>To connect to the Amazon EC2 Linux instance in the private subnet, do the following: 1. Open the Amazon EC2 console. 2. In the navigation pane, choose Instances. 3. Select the Amazon Linux 2 demo instance, and choose Connect. 4. Choose Session Manager. 5. Choose Connect to open a new terminal window. 6. Run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sudo su</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Install and start Docker.</td>
<td>To install and start Docker in the Amazon EC2 Linux instance, do the following:</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>1. To install Docker, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>yum install -y docker</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. To start the Docker service, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>service docker start</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. To verify Docker installation, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>docker info</code></td>
<td></td>
</tr>
<tr>
<td>Install Git and clone the repository.</td>
<td>To install Git on the Amazon EC2 Linux instance and clone the repository from GitHub, do the following.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>1. To install Git, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>yum install git -y</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. To clone the repository, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>git clone https://github.com/&lt;username&gt;/&lt;repo-name&gt;.git</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. To navigate to the Dockerfile, run the following command.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>cd &lt;repo-name&gt;/DemoNetCoreWebAPI/</code></td>
<td></td>
</tr>
</tbody>
</table>
### Task 1: Build and run the Docker container.

**Description:** To build the Docker image and run the container inside the Amazon EC2 Linux instance, do the following:

1. To create the Docker image, run the following command.
   ```bash
docker build -t aspnetcorewebapiimage -f Dockerfile .
   ```

2. To view all the Docker images, run the following command.
   ```bash
docker images
   ```

3. To create and run the container, run the following command.
   ```bash
docker run -d -p 80:80 --name aspnetcorewebapicontainer aspnetcorewebapiimage
   ```

**Skills required:** App developer, AWS administrator, AWS DevOps

---

### Task 2: Test the web API

**Description:** Test the web API using the curl command.

To test the web API, run the following command.

```bash
curl -X GET "http://localhost/WeatherForecast" -H "accept: text/plain"
```

Verify the API response.

**Note:** You can get the curl commands for each endpoint from Swagger when you are running it locally.

**Skills required:** App developer

---

### Task 3: Clean up resources

**Description:** Delete all resources.

Delete the stack to remove all the resources. This ensures

**Skills required:** AWS administrator, AWS DevOps
Run stateful workloads with persistent data storage by using Amazon EFS on Amazon EKS with AWS Fargate

Created by Ricardo Morais (AWS) and Lucio Pereira (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Containers &amp; microservices; Storage &amp; backup</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon EFS; Amazon EKS; AWS Fargate</td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for enabling Amazon Elastic File System (Amazon EFS) as a storage device for containers that are running on Amazon Elastic Kubernetes Service (Amazon EKS), using AWS Fargate to provision your compute resources.

The setup described in this pattern follows security best practices and provides security at rest and security in transit by default. To encrypt your Amazon EFS file system, it uses an AWS Key Management Service (AWS KMS) key, but you can also specify a key alias that dispatches the process of creating a KMS key.

You can follow the steps in this pattern to create a namespace and Fargate profile for a proof-of-concept (PoC) application, install the Amazon EFS Container Storage Interface (CSI) driver that is used to integrate the Kubernetes cluster with Amazon EFS, configure the storage class, and deploy the PoC application. These steps result in an Amazon EFS file system that is shared among multiple Kubernetes workloads, running over Fargate. The pattern is accompanied by scripts that automate these steps.

You can use this pattern whenever you want to ensure data persistence in your containerized applications, and avoid data loss in scale-in or scale-out operations. For example:

- DevOps tools – A common scenario is to use Jenkins as a continuous integration and continuous delivery (CI/CD) tool. In this case, you can use Amazon EFS as a shared file system to store configurations among different instances of the CI/CD tool or to store a cache (for example, an Apache Maven repository) for pipeline stages among different instances of the CI/CD tool.
• Web servers – A common scenario is to use Apache as an HTTP web server. You can use Amazon EFS as a shared file system to store static files that are shared among different instances of the web server. In this example scenario, modifications are applied directly to the file system instead of static files being baked into a Docker image.

Prerequisites and limitations

Prerequisites

• An active AWS account
• An existing Amazon EKS cluster with Kubernetes version 1.17 or later
• Cluster administration permissions
• Context configured to point to the desired Amazon EKS cluster

Limitations

• You have to create the Amazon EFS file system before you can use the CSI driver to mount it inside a container as a persistent volume. Currently, the CSI driver doesn't provision the Amazon EFS file system automatically.
• There are some limitations to consider when you're using Amazon EKS with Fargate. For example, daemonsets and privileged containers aren't supported. For more information, see AWS Fargate considerations in the Amazon EKS documentation.
• The code provided with this pattern supports workstations that are running Linux or macOS.

Product versions

• AWS Command Line Interface (AWS CLI) version 2 or later
• Amazon EFS CSI driver version 1.0 or later
• ekactl version 0.24.0 or later
• jq version 1.6 or later
• kubectl version 1.17 or later
• Kubernetes version 1.17 or later
The target architecture includes the following services and components, and follows AWS Well-Architected Framework best practices:

- Amazon EFS, which provides a simple, scalable, fully managed elastic NFS file system. This is used as a shared file system among all replications of the PoC application that are running in pods, which are distributed in the private subnets of the chosen Amazon EKS cluster.

- An Amazon EFS mount target for each private subnet. This provides redundancy per Availability Zone within the virtual private cloud (VPC) of the cluster.

- Amazon EKS, which runs the Kubernetes workloads. You must provision an Amazon EKS cluster before you use this pattern, as described in the Prerequisites section.
AWS Prescriptive Guidance Patterns
Tools

- AWS KMS, which provides encryption at rest for the content that’s stored in the Amazon EFS file system.
- Fargate, which manages the compute resources for the containers so that you can focus on business requirements instead of infrastructure burden. The Fargate profile is created for all private subnets. It provides redundancy per Availability Zone within the virtual private cloud (VPC) of the cluster.
- Kubernetes pods, for validating that content can be shared, consumed, and written by different instances of an application.

Tools

- AWS CLI – AWS Command Line Interface (AWS CLI) is an open-source tool that you can use to interact with AWS services from the command line.
- eksctl – eksctl is a command-line utility for creating and managing Kubernetes clusters on Amazon EKS.
- kubectl – kubectl is a command-line utility for communicating with the cluster API server.
- jq – jq is a command-line tool for parsing JSON.

AWS services

- Amazon EFS – Amazon Elastic File System (Amazon EFS) manages file storage in the AWS Cloud. In this pattern, it provides a simple, scalable, fully managed, and shared file system for use with Amazon EKS.
- Amazon EKS – Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or operate your own clusters.
- AWS Fargate – AWS Fargate is a serverless compute engine for Amazon EKS. It creates and manages compute resources for your Kubernetes applications.
- AWS KMS – AWS Key Management Service (AWS KMS) is an encryption and key management service that helps you protect your application data.

Code

The code for this pattern is provided in a GitHub repo. The scripts are organized by epic, in the folders epic01 through epic06, corresponding to the order in the following section. For information about how to use the scripts to automate the epics in this pattern, see the Additional information section.

Epics

Create a Kubernetes namespace and a linked Fargate profile

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Kubernetes namespace for application workloads.</td>
<td>Create a namespace for receiving the application workloads that interact with Amazon EFS. To run a script to automate the steps in this epic, see the Additional information section.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Create a custom Fargate profile.</td>
<td>Create a custom Fargate profile that is linked to the namespace you created.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Create an Amazon EFS file system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate an unique token.</td>
<td>Amazon EFS requires a creation token to ensure idempotent operation (calling the operation with the same creation token has no effect). To meet this requirement, you must generate an unique token through an available technique. For example, you can generate an universally unique identifier (UUID) to use as a creation token. After you create a token, you can run a script to automate the remaining steps in this epic; see the Additional information section.</td>
<td>System administrator</td>
</tr>
<tr>
<td>Create a KMS key.</td>
<td>(Optional) Create a unique, symmetric AWS KMS key in your AWS account and AWS Region. This key is used for file system encryption, so that the pattern can use a KMS key for handling encryption at rest. If it is not specified, an AWS managed key is used.</td>
<td>System administrator</td>
</tr>
<tr>
<td>Create an Amazon EFS file system.</td>
<td>Create the file system for receiving the data files that are read and written by the application workloads. You can create an encrypted or non-encrypted file system. (As a best practice, the code for this pattern creates an encrypted system to enable encryption at rest by default.)</td>
<td>System administrator</td>
</tr>
<tr>
<td>Create a security group.</td>
<td>Create a security group to allow the Amazon EKS cluster to access the Amazon EFS file system.</td>
<td>System administrator</td>
</tr>
<tr>
<td>Update the inbound rule for the security group.</td>
<td>Update the inbound rules of the security group to allow incoming traffic for the following settings:</td>
<td>System administrator</td>
</tr>
</tbody>
</table>
## Install Amazon EFS components into the Kubernetes cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the Amazon EFS CSI driver.</td>
<td>Deploy the Amazon EFS CSI driver into the cluster. The driver will provision storage according to persistent volume claims created by applications. To run a script to automate the steps in this epic, see the <em>Additional information</em> section.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Deploy the storage class.</td>
<td>Deploy the storage class into the cluster for the Amazon EFS provisioner (efs.csi.aws.com).</td>
<td>Kubernetes user with granted permissions</td>
</tr>
</tbody>
</table>

## Install the PoC application into the Kubernetes cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the persistent volume.</td>
<td>Deploy the persistent volume, and link it to the created storage class and to the ID of the Amazon EFS file system. The application uses the persistent volume to read and write content. You can specify any size for the persistent volume in the storage field. Kubernetes requires this field, but because Amazon EFS is an elastic file system, it does not enforce any file system capacity. You can deploy the persistent volume with or without encryption. (The Amazon EFS CSI driver enables encryption by default, as a best practice.) To run a script to automate the steps in this epic,</td>
<td>Kubernetes user with granted permissions</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>see the <em>Additional information</em> section.</td>
<td></td>
</tr>
<tr>
<td>Deploy the persistent volume claim requested by the application.</td>
<td>Deploy the persistent volume claim requested by the application, and link it to the storage class. Use the same access mode as the persistent volume you created previously. You can specify any size for the persistent volume claim in the storage field. Kubernetes requires this field, but because Amazon EFS is an elastic file system, it does not enforce any file system capacity.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Deploy workload 1.</td>
<td>Deploy the pod that represents workload 1 of the application. This workload writes content to the file /data/out1.txt.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Deploy workload 2.</td>
<td>Deploy the pod that represents workload 2 of the application. This workload writes content to the file /data/out2.txt.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
</tbody>
</table>

#### Validate file system persistence, durability, and shareability

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate that workload 1 can write to the file system.</td>
<td>Validate that workload 1 of the application is writing to the /data/out1.txt file in the Amazon EFS file system. For commands and scripts that automate the steps in this epic, see the <em>Additional information</em> section.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Validate that workload 2 can write to the file system.</td>
<td>Validate that workload 2 of the application is writing to the /data/out2.txt file in the Amazon EFS file system.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Validate that workload 1 can read the file written by workload 2.</td>
<td>Validate that workload 1 of the application is able to read the file /data/out2.txt, written by workload 2 of the application, from the file system.</td>
<td>Kubernetes user with granted permissions</td>
</tr>
<tr>
<td>Validate that workload 2 can read the file written by workload 1.</td>
<td>Validate that workload 2 of the application is able to read the file /data/out1.txt, written by</td>
<td>Kubernetes user with granted permissions</td>
</tr>
</tbody>
</table>
## Monitor operations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor application logs.</td>
<td>As part of a day-two operation, ship the application logs to Amazon CloudWatch for monitoring.</td>
<td>Kubernetes user with granted permissions, System administrator</td>
</tr>
<tr>
<td>Monitor Amazon EKS and Kubernetes containers with Container Insights.</td>
<td>As part of a day-two operation, monitor the Amazon EKS and Kubernetes systems by using Amazon CloudWatch Container Insights. This tool collects, aggregates, and summarizes metrics from containerized applications at different levels and dimensions. For more information, see the links in the Related resources section.</td>
<td>Kubernetes user with granted permissions, System administrator</td>
</tr>
<tr>
<td>Monitor Amazon EFS with CloudWatch.</td>
<td>As part of a day-two operation, monitor the file systems using Amazon CloudWatch, which collects and processes raw data from Amazon EFS into readable, near real-time metrics. For more information, see the Related resources section.</td>
<td>System administrator</td>
</tr>
</tbody>
</table>

## Clean up resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean up all created resources for the pattern.</td>
<td>After you complete this pattern, clean up all resources, to avoid incurring AWS charges. To run a script to automate the cleanup</td>
<td>Kubernetes user with granted permissions, System administrator</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns

Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>process, see the Additional information section.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

References

- New – AWS Fargate for Amazon EKS now supports Amazon EFS (announcement)
- How to capture application logs when using Amazon EKS on AWS Fargate (blog post)
- Using Container Insights (Amazon CloudWatch documentation)
- Setting Up Container Insights on Amazon EKS and Kubernetes (Amazon CloudWatch documentation)
- Amazon EKS and Kubernetes Container Insights metrics (Amazon CloudWatch documentation)
- Monitoring Amazon EFS with Amazon CloudWatch (Amazon EFS documentation)

GitHub tutorials and examples

- Static provisioning
- Encryption in transit
- Accessing the file system from multiple pods
- Consuming Amazon EFS in StatefulSets
- Mounting subpaths
- Using Amazon EFS access points

Required tools

- Installing the AWS CLI version 2
- Installing eksctl
- Installing kubectl
- Installing jq

Additional information

The scripts that automate the epics and stories for this pattern are included in a GitHub repo and described in the following sections.

Creating a Kubernetes namespace and a linked Fargate profile

Run the following script to implement these two steps:

```bash
./scripts/epic01/create-k8s-ns-and-linked-fargate-profile.sh \ -c "MY_CLUSTERS_NAME"
```

where MY_CLUSTER_NAME is the name of your Amazon EKS cluster.

Creating an Amazon EFS file system

Use the create-efs.sh script to create an encrypted or non-encrypted Amazon EFS file system, after you generate a unique token for Amazon EFS.
With encryption at rest, without a KMS key:

```
./scripts/epic02/create-efs.sh \
  -c "MY_CLUSTER_NAME" \
  -t "MY_EFS_CREATION_TOKEN"
```

where `MY_CLUSTER_NAME` is the name of your Amazon EKS cluster and `MY_EFS_CREATION_TOKEN` is a unique creation token for the file system.

With encryption at rest, with a KMS key:

```
./scripts/epic02/create-efs.sh \
  -c "MY_CLUSTER_NAME" \
  -t "MY_EFS_CREATION_TOKEN" \
  -k "MY_KMS_KEY_ALIAS"
```

where `MY_CLUSTER_NAME` is the name of your Amazon EKS cluster, `MY_EFS_CREATION_TOKEN` is a unique creation token for the file system, and `MY_KMS_KEY_ALIAS` is the alias for the KMS key.

Without encryption:

```
./scripts/epic02/create-efs.sh -d \
  -c "MY_CLUSTER_NAME" \
  -t "MY_EFS_CREATION_TOKEN"
```

where `MY_CLUSTER_NAME` is the name of your Amazon EKS cluster, `MY_EFS_CREATION_TOKEN` is a unique creation token for the file system, and `–d` disables encryption at rest.

Installing Amazon EFS components into the Kubernetes cluster

Run the following script to deploy the Amazon EFS CSI driver and the storage class into the cluster:

```
./scripts/epic03/create-k8s-efs-csi-sc.sh
```

This script uses the `kubectl` utility, so make sure that the context has been configured to point to the desired Amazon EKS cluster.

Installing the PoC application

Run the `deploy-poc-app.sh` script to deploy the persistent volume, the persistent volume claim, and the two workloads.

With encryption in transit:

```
./scripts/epic04/deploy-poc-app.sh \
  -t "MY_EFS_CREATION_TOKEN"
```

where `MY_EFS_CREATION_TOKEN` is the unique creation token for the file system.

Without encryption in transit:

```
./scripts/epic04/deploy-poc-app.sh -d \
  -t "MY_EFS_CREATION_TOKEN"
```

where `MY_EFS_CREATION_TOKEN` is the unique creation token for the file system, and `–d` disables encryption in transit.
Validating file system persistence, durability, and shareability

To validate that workload 1 is writing to /data/out1.txt:

```
kubectl exec -ti poc-app1 -n poc-efs-eks-fargate -- tail -f /data/out1.txt
```

The results will be similar to the following:

```
...
Thu Sep 3 15:25:07 UTC 2020 - PoC APP 1
Thu Sep 3 15:25:12 UTC 2020 - PoC APP 1
Thu Sep 3 15:25:17 UTC 2020 - PoC APP 1
...
```

To validate that workload 2 is writing to /data/out2.txt:

```
kubectl exec -ti poc-app2 -n poc-efs-eks-fargate -- tail -f /data/out2.txt
```

The results will be similar to the following:

```
...
Thu Sep 3 15:26:48 UTC 2020 - PoC APP 2
Thu Sep 3 15:26:53 UTC 2020 - PoC APP 2
Thu Sep 3 15:26:58 UTC 2020 - PoC APP 2
...
```

To validate that workload 1 can read the file written by workload 2:

```
kubectl exec -ti poc-app1 -n poc-efs-eks-fargate -- tail -n 3 /data/out2.txt
```

The results will be similar to the following:

```
...
Thu Sep 3 15:26:48 UTC 2020 - PoC APP 2
Thu Sep 3 15:26:53 UTC 2020 - PoC APP 2
Thu Sep 3 15:26:58 UTC 2020 - PoC APP 2
...
```

To validate that workload 2 can read the file written by workload 1:

```
kubectl exec -ti poc-app2 -n poc-efs-eks-fargate -- tail -n 3 /data/out1.txt
```

The results will be similar to the following:

```
...
Thu Sep 3 15:29:22 UTC 2020 - PoC APP 1
Thu Sep 3 15:29:27 UTC 2020 - PoC APP 1
Thu Sep 3 15:29:32 UTC 2020 - PoC APP 1
...
```

To validate that files are retained after you remove application components:

```
./scripts/epic05/validate-efs-content.sh \
  -t "MY_EFS_CREATION_TOKEN"
```

275
where `MY_EFS_CREATION_TOKEN` is the unique creation token for the file system.

The results will be similar to the following:

```
pod/poc-app-validation created
Waiting for pod get Running state...
Waiting for pod get Running state...
Waiting for pod get Running state...
Results from execution of 'find /data' on validation process pod:
/data
/data/out2.txt
/data/out1.txt
```

**Cleaning up resources**

Run the `clean-up-resources.sh` script to remove all resources after you have finished using the PoC application.

*With encryption at rest, with a KMS key:*

```
./scripts/epic06/clean-up-resources.sh \
  -c "MY_CLUSTER_NAME" \
  -t "MY_EFS_CREATION_TOKEN" \
  -k "MY_KMS_KEY_ALIAS"
```

where `MY_CLUSTER_NAME` is the name of your Amazon EKS cluster, `MY_EFS_CREATION_TOKEN` is the creation token for the file system, and `MY_KMS_KEY_ALIAS` is the alias for the KMS key.

*Without encryption at rest:*

```
./scripts/epic06/clean-up-resources.sh \
  -c "MY_CLUSTER_NAME" \
  -t "MY_EFS_CREATION_TOKEN"
```

where `MY_CLUSTER_NAME` is the name of your Amazon EKS cluster and `MY_EFS_CREATION_TOKEN` is the creation token for the file system.

---

**More patterns**

- Assess application readiness for migration to the AWS Cloud by using CAST Highlight (p. 1321)
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK (p. 395)
- Create a CI/CD pipeline to deploy microservices with AWS Fargate and Amazon API Gateway (p. 434)
- Create a custom log parser for Amazon ECS using a Firelens log router (p. 441)
- Deploy a CI/CD pipeline for Java microservices on Amazon ECS (p. 457)
- Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile (p. 468)
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK (p. 1459)
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk (p. 1026)
- Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS (p. 1204)
- Modernize ASP.NET Web Forms applications on AWS (p. 1466)
- Monitor Amazon ECR repositories for wildcard permissions using AWS CloudFormation and AWS Config (p. 507)
• Set up a Helm v3 chart repository in Amazon S3 (p. 524)
• Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let's Encrypt (p. 530)
• Train and deploy a custom GPU-supported ML model on Amazon SageMaker (p. 648)
Data lakes

Topics
- Automate data ingestion from AWS Data Exchange into Amazon S3 (p. 278)
- Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code (p. 281)
- Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator (p. 286)
- More patterns (p. 292)

Automate data ingestion from AWS Data Exchange into Amazon S3

*Created by Adnan Alvee (AWS)*

<table>
<thead>
<tr>
<th>Technologies:</th>
<th>Environment:</th>
<th>AWS services:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics; Data lakes</td>
<td>Production</td>
<td>Amazon S3; Amazon CloudWatch; AWS Lambda; Amazon SNS</td>
</tr>
</tbody>
</table>

Summary

This pattern provides an AWS CloudFormation template that enables you to automatically ingest data from AWS Data Exchange into your data lake in Amazon Simple Storage Service (Amazon S3).

AWS Data Exchange is a service that makes it easy to securely exchange file-based data sets in the AWS Cloud. AWS Data Exchange data sets are subscription-based. As a subscriber, you can also access data set revisions as providers publish new data.

The AWS CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event watches for any updates to the data set you have subscribed to. If there is an update, CloudWatch initiates a Lambda function, which copies the data over to the S3 bucket you specify. When the data has been copied successfully, Lambda sends you an Amazon Simple Notification Service (Amazon SNS) notification.

Prerequisites and limitations

**Prerequisites**
- An active AWS account
- Subscription to a data set in AWS Data Exchange

**Limitations**
- The AWS CloudFormation template must be deployed separately for each subscribed data set in AWS Data Exchange.
Architecture

Target technology stack

- AWS Lambda
- Amazon S3
- AWS Data Exchange
- Amazon CloudWatch
- Amazon SNS

Target architecture

Automation and scale

You can use the AWS CloudFormation template multiple times for the data sets you want to ingest into the data lake.

Tools

- AWS Data Exchange – A service that makes it easy for AWS customers to securely exchange file-based data sets in the AWS Cloud. As a subscriber, you can find and subscribe to hundreds of products from qualified data providers. Then, you can quickly download the data set or copy it to Amazon S3 for use across a variety of AWS analytics and machine learning services. Anyone with an AWS account can be an AWS Data Exchange subscriber.
- AWS Lambda – A compute service that lets you run code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume; there is no charge when your code isn't running. With AWS Lambda, you can run code for virtually any type of application or backend service with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and manages all the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging.
• **Amazon S3** – Storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

• **Amazon CloudWatch Events** – Delivers a near real-time stream of system events that describe changes in AWS resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams. CloudWatch Events becomes aware of operational changes as they occur. It responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information. You can also use CloudWatch Events to schedule automated actions that self-initiate at certain times using **cron** or **rate** expressions.

• **Amazon SNS** – A web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud. Amazon SNS provides topics (communication channels) for high-throughput, push-based, many-to-many messaging. Using Amazon SNS topics, publishers can distribute messages to a large number of subscribers for parallel processing, including Amazon Simple Queue Service (Amazon SQS) queues, AWS Lambda functions, and HTTP/S webhooks. You can also use Amazon SNS to send notifications to end users using mobile push, SMS, and email.

## Epics

### Subscribe to a data set

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to a data set.</td>
<td>In the AWS Data Exchange console, subscribe to a dataset. For instructions, see the link in the &quot;Related resources&quot; section.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Note the data set attributes.</td>
<td>Note the AWS Region, ID, and revision ID for the data set. You will need this for the AWS CloudFormation template in the next step.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

### Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket and folder.</td>
<td>If you already have a data lake in Amazon S3, create a folder to store the data to ingest from AWS Data Exchange. If you are deploying the template for testing purposes, create a new S3 bucket, and note the bucket name and folder prefix for the next step.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>Deploy the AWS CloudFormation template that's provided as an attachment to this pattern. Configure the following parameters to correspond to your AWS account, data set, and</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
### Task Description

**S3 bucket settings:** Dataset AWS Region, Dataset ID, Revision ID, S3 Bucket Name (for example, DOC-EXAMPLE-BUCKET), Folder Prefix (for example, myfolder/), and Email for SNS Notification. You can set the Dataset Name parameter to any name. When you deploy the template, it runs a Lambda function to automatically ingest the first set of data available in the data set. Subsequent ingestion then takes place automatically, as new data arrives in the data set.

### Related resources

- [Subscribing to data products on AWS Data Exchange](https://aws.amazon.com/data-exchange/) (AWS Data Exchange documentation)

### Attachments

attachment.zip

### Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code

*Created by Kirankumar Chandrashekar (AWS) and Abdel Jaidi (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Data lakes; Analytics; Serverless; DevOps</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services:</strong></td>
<td>Amazon S3; Amazon SQS; AWS CloudFormation; AWS Glue; Amazon CloudWatch; AWS Lambda; AWS Step Functions; Amazon DynamoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Summary

This pattern describes how to use serverless computing and infrastructure as code (IaC) to implement and administer a data lake on the Amazon Web Services (AWS) Cloud. This pattern is based on the serverless data lake framework (SDLF) workshop developed by AWS.
SDLF is a collection of reusable resources that accelerate the delivery of enterprise data lakes on the AWS Cloud and helps with faster deployment to production. It is used to implement the foundational structure of a data lake by following best practices.

SDLF implements a continuous integration / continuous deployment (CI/CD) process throughout the code and infrastructure deployment by using AWS services such as AWS CodePipeline, AWS CodeBuild, and AWS CodeCommit.

This pattern uses multiple AWS serverless services to simplify data lake management. These include Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB for storage, AWS Lambda and AWS Glue for computing, and Amazon CloudWatch Events, Amazon Simple Queue Service (Amazon SQS), and AWS Step Functions for orchestration.

AWS CloudFormation and AWS code services act as the IaC layer to provide reproducible and fast deployments with easy operations and administration.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured.
- A Git client, installed and configured.
- The SDLF workshop, open in a web browser window and ready to use.
The architecture diagram illustrates an event-driven process with the following steps.

1. After a file is added to the raw data S3 bucket, an Amazon S3 event notification is placed in an SQS queue. Each notification is delivered as a JSON file, which contains metadata such as the S3 bucket name, object key, or timestamp.

2. This notification is consumed by a Lambda function that routes the event to the correct extraction, transformation, and loading (ETL) process based on the metadata. The Lambda function can also use contextual configurations stored in an Amazon DynamoDB table. This step enables decoupling and scaling to multiple applications in the data lake.

3. The event is routed to the first Lambda function in the ETL process, which transforms and moves data from the raw data area to the staging area for the data lake. The first step is to update the comprehensive catalog. This is a DynamoDB table that contains all the file metadata of the data lake. Each row in this table holds operational metadata about a single object stored in Amazon S3. A synchronous call is made to a Lambda function that performs a light transformation, which is a computationally inexpensive operation (such as converting a file from one format to another), on the S3 object. Because a new object has been added to the staging S3 bucket, the comprehensive catalog is updated and a message is sent to the SQS queue waiting for the next phase in the ETL.

4. A CloudWatch Events rule triggers a Lambda function every 5 minutes. This function checks if messages were delivered to the SQS queue from the previous ETL phase. If a message was delivered, the Lambda function begins the second function from AWS Step Functions in the ETL process.
5. A heavy transformation is then applied on a batch of files. This heavy transformation is a computationally expensive operation, such as a synchronous call to an AWS Glue job, AWS Fargate task, Amazon EMR step, or Amazon SageMaker notebook. Table metadata is extracted from the output files by using an AWS Glue crawler, which updates the AWS Glue catalog. File metadata is also added to the comprehensive catalog table in DynamoDB. Finally, a data quality step leveraging Deequ is also run.

Technology stack
- Amazon CloudWatch Events
- AWS CloudFormation
- AWS CodePipeline
- AWS CodeBuild
- AWS CodeCommit
- Amazon DynamoDB
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon SQS
- AWS Step Functions

Tools
- Amazon CloudWatch Events – CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS CloudFormation – CloudFormation helps create and provision AWS infrastructure deployments predictably and repeatedly.
- AWS CodeBuild – CodeBuild is a fully managed build service that compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit – CodeCommit is a version control service hosted by AWS that you can use to privately store and manage assets (such as source code and binary files).
- AWS CodePipeline – CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software changes continuously.
- Amazon DynamoDB – DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with scalability.
- AWS Glue – AWS Glue is a fully managed ETL service that makes it easier to prepare and load data for analytics.
- AWS Lambda – Lambda supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service. Amazon S3 can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- AWS Step Functions - AWS Step Functions is a serverless function orchestrator that makes it easy to sequence AWS Lambda functions and multiple AWS services into business-critical applications.
- Amazon SQS – Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that helps you decouple and scale microservices, distributed systems, and serverless applications.
- **Deequ** – Deequ is a tool that helps you compute data quality metrics for large datasets, define and verify data quality constraints, and stay informed about changes in the data distribution.

**Code**

The source code and resources for the SDLF are available in the [AWS Labs GitHub repository](https://github.com/awslabs/sdlf).

**Epics**

**Set up the CI/CD pipeline to provision IaC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the CI/CD pipeline to manage IaC for the data lake.</td>
<td>Sign in to the AWS Management Console and follow the steps from the <em>Initial setup</em> section of the SDLF workshop. This creates the initial CI/CD resources, such as CodeCommit repositories, CodeBuild environments, and CodePipeline pipelines that provision and manage IaC for the data lake.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

**Version-control the IaC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the CodeCommit repository on your local machine.</td>
<td>Follow the steps from the <em>Deploying the foundations</em> section of the SDLF workshop. This helps you clone the Git repository that hosts IaC into your local environment. For more information, see <em>Connecting to CodeCommit repositories</em> from the CodeCommit documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Modify the CloudFormation templates.</td>
<td>Use your local workstation and a code editor to modify the CloudFormation templates according to your use cases or requirements. Commit them to the locally cloned Git repository. For more information, see <em>Working with AWS CloudFormation templates</em> from the AWS CloudFormation documentation.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
---|---|---
Push the changes to the CodeCommit repository. | Your infrastructure code is now under version control and modifications to your code base are tracked. When you push a change to the CodeCommit repository, CodePipeline automatically applies it to your infrastructure and delivers it to CodeBuild. | DevOps engineer

Important: If you use the AWS SAM CLI in CodeBuild, run the `sam package` and `sam deploy` commands. If you use AWS CLI, run the `aws cloudformation package` and `aws cloudformation deploy` commands.

Related resources

Set up the CI/CD pipeline to provision IaC
- SDLF workshop – Initial setup

Version-control the IaC
- SDLF workshop – Deploying the foundations
- Connecting to CodeCommit repositories
- Working with AWS CloudFormation templates

Other resources
- AWS serverless data analytics pipeline reference architecture
- SDLF documentation

Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator

Created by Tony Velcich

Source: On-premises Hadoop cluster  |  Target: Amazon S3  |  R Type: Rehost
Summary

This pattern describes the process for migrating Apache Hadoop data from a Hadoop Distributed File System (HDFS) to Amazon Simple Storage Service (Amazon S3). It uses WANdisco LiveData Migrator to automate the data migration process.

Prerequisites and limitations

Prerequisites

- Hadoop cluster edge node where LiveData Migrator will be installed. The node should meet the following requirements:
  - Minimum specification: 4 CPUs, 16 GB RAM, 100 GB storage.
  - 2 Gbps minimum network.
  - Port 8081 accessible on your edge node to access the WANdisco UI.
  - Java 1.8 64-bit.
  - Hadoop client libraries installed on the edge node.
  - Ability to authenticate as the HDFS superuser (for example, "hdfs").
  - If Kerberos is enabled on your Hadoop cluster, a valid keytab that contains a suitable principal for the HDFS superuser must be available on the edge node.
  - See the release notes for a list of supported operating systems.
- An active AWS account with access to an S3 bucket.
- An AWS Direct Connect link established between your on-premises Hadoop cluster (specifically the edge node) and AWS.

Product versions

- LiveData Migrator 1.8.6
- WANdisco UI (OneUI) 5.8.0

Architecture

Source technology stack

- On-premises Hadoop cluster

Target technology stack

- Amazon S3

Architecture

The following diagram shows theLiveData Migrator solution architecture.
The workflow consists of four primary components for data migration from on-premises HDFS to Amazon S3.

- **LiveData Migrator** – Automates the migration of data from HDFS to Amazon S3, and resides on an edge node of the Hadoop cluster.
- **HDFS** – A distributed file system that provides high-throughput access to application data.
- **Amazon S3** – An object storage service that offers scalability, data availability, security, and performance.
- **AWS Direct Connect** – A service that establishes a dedicated network connection from your on-premises data centers to AWS.

### Automation and scale

You will typically create multiple migrations so that you can select specific content from your source file system by path or directory. You can also migrate data to multiple, independent file systems at the same time by defining multiple migration resources.

### Epics

#### Configure Amazon S3 storage in your AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to your AWS account.</td>
<td>Sign in to the AWS Management Console and open the Amazon S3 console at <a href="https://console.aws.amazon.com/s3/">https://console.aws.amazon.com/s3/</a>.</td>
<td>AWS experience</td>
</tr>
<tr>
<td>Create an S3 bucket.</td>
<td>If you don't already have an existing S3 bucket to use as the target storage, choose the “Create bucket” option on the Amazon S3 console, and specify a bucket name, AWS Region, and bucket settings for block public access. AWS and WANdisco recommend that you enable</td>
<td>AWS experience</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>the block public access options for the S3 bucket, and set up the bucket access and user permission policies to meet your organization's requirements. An AWS example is provided at <a href="https://docs.aws.amazon.com/AmazonS3/latest/dev/example-walkthroughs-managing-access-example1.html">https://docs.aws.amazon.com/AmazonS3/latest/dev/example-walkthroughs-managing-access-example1.html</a>.</td>
<td></td>
</tr>
</tbody>
</table>

### Install LiveData Migrator

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Download the LiveData Migrator installer and upload it to the Hadoop edge node. You can download a free trial of LiveData Migrator at <a href="https://www2.wandisco.com/ldm-trial">https://www2.wandisco.com/ldm-trial</a>. You can also obtain access to LiveData Migrator from AWS Marketplace, at <a href="https://aws.amazon.com/marketplace/pp/B07B8SZND9">https://aws.amazon.com/marketplace/pp/B07B8SZND9</a>.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
<tr>
<td></td>
<td>Use the downloaded installer and install LiveData Migrator as the HDFS superuser on an edge node in your Hadoop cluster. See the &quot;Additional information&quot; section for the installation commands.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
<tr>
<td></td>
<td>Check the status of LiveData Migrator, Hive migrator, and WANdisco UI by using the commands provided in the &quot;Additional information&quot; section.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
</tbody>
</table>

### Configure storage through the WANdisco UI

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Log in to the WANdisco UI through a web browser on port 8081 (on the Hadoop edge node) and provide your details for registration. For example, if you are running LiveData Migrator on a host named</td>
<td>Application owner</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>Configure your source HDFS storage.</td>
<td>Provide the configuration details needed for your source HDFS storage. This will include the &quot;fs.defaultFS&quot; value and a user-defined storage name. If Kerberos is enabled, provide the principal and keytab location for LiveData Migrator to use. If NameNode HA is enabled on the cluster, provide a path to the core-site.xml and hdfs-site.xml files on the edge node.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
<tr>
<td>Configure your target Amazon S3 storage.</td>
<td>Add your target storage as the S3a type. Provide the user-defined storage name and the S3 bucket name. Enter &quot;org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider&quot; for the Credentials Provider option, and provide the AWS access and secret keys for the S3 bucket. Additional S3a properties will also be needed. For details, see the &quot;S3a Properties&quot; section in the LiveData Migrator documentation at <a href="https://docs.wandisco.com/live-data-migrator/docs/command-reference/#filesystem-add-s3a">https://docs.wandisco.com/live-data-migrator/docs/command-reference/#filesystem-add-s3a</a>.</td>
<td>AWS, Application owner</td>
</tr>
</tbody>
</table>

**Prepare for the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add exclusions (if needed).</td>
<td>If you want to exclude specific datasets from migration, add exclusions for the source HDFS storage. These exclusions can be based on file size, file names (based on regex patterns), and modification date.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
</tbody>
</table>
Create and start the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and configure the migration.</td>
<td>Create a migration in the dashboard of the WANdisco UI. Choose your source (HDFS) and target (the S3 bucket). Add new exclusions that you have defined in the previous step. Select either the &quot;Overwrite&quot; or the &quot;Skip if Size Match&quot; option. Create the migration when all fields are complete.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
<tr>
<td>Start the migration.</td>
<td>On the dashboard, select the migration you created. Click to start the migration. You can also start a migration automatically by choosing the auto-start option when you create the migration.</td>
<td>Application owner</td>
</tr>
</tbody>
</table>

Manage bandwidth (optional)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set a network bandwidth limit between the source and target.</td>
<td>In the Storages list on the dashboard, select your source storage and select &quot;Bandwidth Management&quot; in the Grouping list. Clear the unlimited option, and define the maximum bandwidth limit and unit. Choose &quot;Apply.&quot;</td>
<td>Application owner, Networking</td>
</tr>
</tbody>
</table>

Monitor and manage migrations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>View migration information using the WANdisco UI.</td>
<td>Use the WANdisco UI to view license, bandwidth, storage and migration information. The UI also provides a notification system so you can receive notifications about errors, warnings, or important milestones in your usage.</td>
<td>Hadoop administrator, Application owner</td>
</tr>
<tr>
<td>Stop, resume, and delete migrations.</td>
<td>You can stop a migration from transferring content to its target by placing it in the STOPPED state. Stopped migrations can</td>
<td>Hadoop administrator, Application owner</td>
</tr>
</tbody>
</table>
be resumed. Migrations in the STOPPED state can also be deleted.

### Related resources

- LiveData Migrator documentation
- LiveData Migrator in AWS Marketplace
- WANdisco support community
- WANdisco LiveData Migrator demonstration (video)

### Additional information

**Installing LiveData Migrator**

You can use the following commands to install LiveData Migrator, assuming that the installer is inside your working directory:

```
su - hdfs
chmod +x livedata-migrator.sh && sudo ./livedata-migrator.sh
```

**Checking the status of LiveData Migrator and other services after installation**

Use the following commands to check the status of LiveData Migrator, Hive migrator, and WANdisco UI:

```
service livedata-migrator status
service hivemigrator status
service livedata-ui status
```

### More patterns

- Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue (p. 12)
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Kinesis Data Firehose with AWS CDK (p. 1747)
- Ensure an Amazon Redshift cluster is encrypted upon creation (p. 1679)
- Migrate data to the AWS Cloud by using Starburst (p. 53)
- Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions (p. 58)
- Verify that new Amazon Redshift clusters have required SSL endpoints (p. 1563)
- Visualize Amazon Redshift audit logs using Amazon Athena and Amazon QuickSight (p. 88)
Databases

Topics

- Access on-premises Microsoft SQL Server tables from Microsoft SQL Server on Amazon EC2 using linked servers (p. 293)
- Copy Amazon DynamoDB tables across accounts using AWS Backup (p. 297)
- Copy Amazon DynamoDB tables across accounts using a custom implementation (p. 302)
- Deploy DataStax Enterprise on AWS (p. 309)
- Determine whether your Microsoft SQL Server database can be downgraded from Enterprise to Standard edition (p. 311)
- Encrypt an existing Amazon RDS for PostgreSQL DB instance (p. 313)
- Enforce automatic tagging of Amazon RDS databases at launch (p. 319)
- Help enforce DynamoDB tagging (p. 322)
- Implement cross-Region disaster recovery with AWS DMS and Amazon Aurora (p. 326)
- Migrate SAP HANA to AWS using SAP HSR with the same hostname (p. 336)
- Migrate SQL Server to AWS using distributed availability groups (p. 346)
- Monitor Amazon Aurora for instances without encryption (p. 353)
- Replicate data between Amazon RDS for MySQL and MySQL on Amazon EC2 (p. 357)
- Schedule jobs for Amazon RDS and Aurora PostgreSQL using Lambda and Secrets Manager (p. 359)
- Database migration patterns by workload (p. 364)
- More patterns (p. 367)

Access on-premises Microsoft SQL Server tables from Microsoft SQL Server on Amazon EC2 using linked servers

Created by Tirumala Dasari (AWS)

Created by: AWS  Environment: PoC or pilot  Technologies: Databases  Workload: Microsoft

Summary

This pattern describes how to access on-premises Microsoft SQL Server database tables running on Microsoft Windows, from Microsoft SQL Server databases running on Amazon Elastic Compute Cloud (Amazon EC2) Windows or Linux instances by using linked servers.
Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon EC2 with Microsoft SQL Server running on Amazon Linux AMI (Amazon Machine Image)
- AWS Direct Connect between the on-premises Microsoft SQL Server (Windows) server and the Linux EC2 instance

Architecture

Source technology stack

- On-premises Microsoft SQL Server database running on Windows
- Amazon EC2 with Microsoft SQL Server running on Amazon Linux AMI

Target technology stack

- Amazon EC2 with Microsoft SQL Server running on Amazon Linux AMI
- Amazon EC2 with Microsoft SQL Server running on Windows AMI

Source and target database architecture

Tools

- **SSMS** - Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing a SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.
Epics

Change authentication mode to Windows and SQL Server in Windows SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to Windows SQL Server through SSMS.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Change the authentication mode to Windows and SQL Server from the context (right-click) menu for the Windows SQL Server instance.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Restart the Windows MSSQL service

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the context (right-click) menu for the Windows SQL Server instance and select Restart.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Create new login and choose databases to access in Windows SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the Security tab, open the context (right-click) menu for Login and select a new login.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>In the General tab, choose SQL Server authentication, enter a user name, enter the password, and then confirm the password and clear the option for changing the password at the next login.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>In the Server Roles tab, choose Public.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>In the User Mapping tab, choose the database and schema you want to access, and then highlight the database to select database roles.</td>
<td>Select public and db_datarreader to access data from the database tables.</td>
<td>DBA</td>
</tr>
<tr>
<td>Choose OK to create a user.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>
Add Windows SQL Server IP to Linux SQL Server host file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Connect to the Linux SQL Server box through the terminal window.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Open the /etc/hosts file and add the IP address of the Windows machine with SQL Server.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Save the hosts file.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Create linked server on Linux SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a linked server by using the stored procedures master.sys.sp_addlinkedserver and master.dbo.sp_addlinkedsrvlogin. For more information about using these stored procedures, see the Additional information section.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

Verify the created linked server and databases in SSMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In Linux SQL Server in SSMS, go to Linked Servers and refresh.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Expand the created linked servers and catalogs in the left pane. You'll see the selected SQL Server databases with tables and views.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Verify that you can access Windows SQL Server database tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In the SSMS query window, run the query: &quot;select top 3 * from [sqllin].dms_sample_win.dbo.mlb_data&quot;. Note that the FROM clause uses a four-part syntax: computer.database.schema.table (e.g., SELECT name &quot;SQL2 databases&quot; FROM [sqllin].master.sys.databases). In our example, we created an alias for SQL2 in the hosts file, so you don’t need to enter the actual NetBIOS name between the square brackets. If you do use</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>
**Related resources**

- Release notes for SQL Server on Linux

**Additional information**

**Using stored procedures to create linked servers**

SSMS doesn't support the creation of linked servers for Linux SQL Server, so you have to use these stored procedures to create them:

```sql
EXEC master.sys.sp_addlinkedserver @server= 'N'SQLLIN' , @srvproduct= 'N'SQL Server'
EXEC master.dbo.sp_addlinkededsrvlogin @rmtsrvname=N'SQLLIN',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassword='Test123$
```

Note 1: Enter the user name and password that you created earlier in Windows SQL Server in the stored procedure `master.dbo.sp_addlinkededsrvlogin`.

Note 2: @server name SQLLIN and host file entry name 172.12.12.4 SQLLIN should be the same.

You can use this process to create linked servers for the following scenarios:

- Linux SQL Server to Windows SQL Server through a linked server (as specified in this pattern)
- Windows SQL Server to Linux SQL Server through a linked server
- Linux SQL Server to another Linux SQL Server through a linked server

**Copy Amazon DynamoDB tables across accounts using AWS Backup**

*Created by Ramkumar Ramanujam (AWS)*

**Environment:** PoC or pilot  
**Technologies:** Databases; Migration  
**AWS services:** Amazon DynamoDB; AWS Backup
Summary

When working with Amazon DynamoDB on Amazon Web Services (AWS), a common use case is to copy or sync DynamoDB tables in development, testing, or staging environments with the table data that is in the production environment. As a standard practice, each environment uses a different AWS account.

AWS Backup supports cross-Region and cross-account backup and restore of data for DynamoDB, Amazon Simple Storage Service (Amazon S3), and other AWS services. This pattern provides the steps for using AWS Backup cross-account backup and restore to copy DynamoDB tables between AWS accounts.

Prerequisites and limitations

Prerequisites

- Two active AWS accounts that belong to the same AWS Organizations organization
- DynamoDB tables in both the accounts.
- AWS Identity and Access Management (IAM) permissions to create and use AWS backup vaults

Limitations

- Source and target AWS accounts should be part of the same AWS Organizations organization.

Architecture

Target technology stack

- AWS Backup
- Amazon DynamoDB

Target architecture
1. Create the DynamoDB table backup in the AWS Backup backup vault in the source account.
2. Copy the backup to the backup vault in the target account.
3. Restore the DynamoDB table in the target account using the backup from the target account backup vault.

Automation and scale

You can use AWS Backup to schedule backups to run at specific intervals.

Tools

- **AWS Backup** – AWS Backup is a fully-managed service for centralizing and automating data protection across AWS services, in the cloud, and on premises. Using this service, you can configure backup policies and monitor activity for your AWS resources in one place. It allows you to automate and consolidate backup tasks that were previously performed service-by-service, and removes the need to create custom scripts and manual processes.
- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
Epics

Turn on AWS Backup features in the source and target accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Turn on advanced features for DynamoDB and cross-account backup. | In both the source and the target AWS accounts, do the following:  
1. On the AWS Management Console, open the AWS Backup console.  
2. Choose Settings.  
3. Under Advanced features for Amazon DynamoDB backups, confirm that Advanced features is enabled, or choose Enable.  
4. Under Cross-account management, for Cross-account backup, choose Enable. | AWS DevOps, Migration engineer      |

Create backup vaults in the source and target accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create backup vaults. | In both the source and the target AWS accounts, do the following:  
1. On the AWS Backup console, choose Backup vaults.  
2. Choose Create Backup vault.  
3. Copy the Amazon Resource Name (ARN) of the backup vault and save it.  

The ARNs of both the source and the target backup vaults will be required when you copying the DynamoDB table backup between the source account and the target account. | AWS DevOps, Migration engineer      |
## Perform backup and restore using backup vaults

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| In the source account, create a DynamoDB table backup. | To create a backup for the DynamoDB table in the source account, do the following:  
1. On the AWS Backup Dashboard page, choose Create on-demand backup.  
2. In the Settings section, for Resource type, select DynamoDB, and then select the table name.  
3. In the Backup vault dropdown list, select the backup vault that you created in the source account.  
4. Select the Retention period that you want.  
5. Choose Create on-demand backup.  

A new backup job is created.  
To monitor the status of the backup job, on the AWS Backup Jobs page, choose the Backup Jobs tab. All active, in-progress, and completed backup jobs are listed in this tab. | AWS DevOps, DBA, Migration engineer |
| Copy the backup from the source account to the target account. | After the backup job is completed, copy the DynamoDB table backup from the backup vault in the source account to the backup vault in target account.  
To copy the backup vault, in the source account, do the following:  
1. On the AWS Backup console, choose Backup vaults.  
2. Under Backups, choose the DynamoDB table backup.  
3. Choose Actions, Copy.  
4. Enter the AWS Region of the target account.  
5. For External vault ARN, enter the ARN of the backup vault that you created in the target account. | AWS DevOps, Migration engineer, DBA |
6. To copy backups from the source account to the target account, in the target account backup vault, enable access from a different account.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6. To copy backups from the source account to the target account, in the target account backup vault, enable access from a different account.</td>
<td>AWS DevOps, DBA, Migration engineer</td>
</tr>
</tbody>
</table>

Restore the backup in the target account.

In the target AWS account, do the following:

1. On the AWS Backup console, choose **Backup vaults**.
2. Under **Backups**, select the backup that you copied from the source account.
3. Choose **Actions**, **Restore**.
4. Enter the name of the target DynamoDB table that you want to restore.

### Related resources

- Using AWS Backup with DynamoDB
- Creating backup copies across AWS accounts
- AWS Backup pricing

### Copy Amazon DynamoDB tables across accounts using a custom implementation

*Created by Ramkumar Ramanujam (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Source:</th>
<th>Amazon DynamoDB</th>
<th>Target:</th>
<th>Amazon DynamoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>N/A</td>
<td>Workload:</td>
<td>All other workloads</td>
<td>Technologies:</td>
<td>Databases</td>
</tr>
<tr>
<td><strong>AWS services:</strong></td>
<td>Amazon DynamoDB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Summary

When working with Amazon DynamoDB on Amazon Web Services (AWS), a common use case is to copy or sync DynamoDB tables in development, testing, or staging environments with the table data that are in the production environment. As a standard practice, each environment uses a different AWS account.

DynamoDB now supports cross-account backup using AWS Backup. For information about associated storage costs when using AWS Backup, see AWS Backup pricing. When you use AWS Backup to copy across accounts, the source and target accounts must be part of an AWS Organizations organization. There are other solutions for cross-account backup and restore using AWS services such as AWS Data...
Pipeline or AWS Glue. Using those solutions, however, increases the application footprint, because there are more AWS services to deploy and maintain.

You can also use Amazon DynamoDB Streams to capture table changes in the source account. Then you can initiate an AWS Lambda function, and make the corresponding changes in the target table in the target account. But that solution applies to use cases in which source and target tables must always be kept in sync. It might not apply to development, testing, and staging environments where data are updated frequently.

This pattern provides steps to implement a custom solution to copy a Amazon DynamoDB table from one account to another. This pattern can be implemented using common programming languages such as C#, Java, and Python. We recommend using a language that is supported by an AWS SDK.

**Prerequisites and limitations**

**Prerequisites**

- Two active AWS accounts
- DynamoDB tables in both the accounts
- Knowledge of AWS Identity and Access Management (IAM) roles and policies
- Knowledge of how to access Amazon DynamoDB tables using any common programming language, such as C#, Java, or Python

**Limitations**

This pattern applies to DynamoDB tables that are around 2 GB or smaller. With additional logic to handle connection or session interruptions, throttling, and failures and retries, it can be used for larger tables.

The DynamoDB scan operation, which reads items from the source table, can fetch only up to 1 MB of data in a single call. For larger tables, greater than 2 GB, this limitation can increase the total time to perform a full table copy.

**Architecture**

**Automation and scale**

This pattern applies to DynamoDB tables that are smaller in size, around 2 GB.

To apply this pattern for larger tables, address the following issues:
During the table copy operation, two active sessions are maintained, using different security tokens. If the table copy operation takes longer than the token expiration times, you must put in place logic to refresh the security tokens.

If enough read capacity units (RCUs) and write capacity units (WCUs) are not provisioned, reads or writes on the source or target table might get throttled. Be sure to catch and handle these exceptions.

Handle any other failures or exceptions and put a retry mechanism in place to retry or continue from where the copy operation failed.

Tools

Amazon DynamoDB – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

The additional tools required will differ based on the programming language that you choose for the implementation. For example, if you use C#, you will need Microsoft Visual Studio and the following NuGet packages:

- AWSSDK
- AWSSDK.DynamoDBv2

Code

The following Python code snippet deletes and recreates a DynamoDB table using the Boto3 library.

```python
import boto3
import sys
import json

#args = input-parameters = GLOBAL_SEC_INDEXES_JSON_COLLECTION, ATTRIBUTES_JSON_COLLECTION, TARGET_DYNAMODB_NAME, TARGET_REGION, ...

#Input param: GLOBAL_SEC_INDEXES_JSON_COLLECTION
#[("IndexName":"Test-index","KeySchema":[{"AttributeName":"AppId","KeyType":"HASH"},
{"AttributeName":"AppType","KeyType":"RANGE"}],"Projection":
{"ProjectionType":"INCLUDE","NonKeyAttributes":["PK","SK","OwnerName","AppVersion"]})]

#Input param: ATTRIBUTES_JSON_COLLECTION
#[("AttributeName":"PK","AttributeType":"S"),{"AttributeName":"SK","AttributeType":"S"},
{"AttributeName":"AppId","AttributeType":"S"},
{"AttributeName":"AppType","AttributeType":"N"}]

region = args['TARGET_REGION']
target_ddb_name = args['TARGET_DYNAMODB_NAME']

global_secondary_indexes = json.loads(args['GLOBAL_SEC_INDEXES_JSON_COLLECTION'])
attribute_definitions = json.loads(args['ATTRIBUTES_JSON_COLLECTION'])

# Drop and create target DynamoDB table
dynamodb_client = boto3.Session(  
    aws_access_key_id=args['AWS_ACCESS_KEY_ID'],
    aws_secret_access_key=args['AWS_SECRET_ACCESS_KEY'],
    aws_session_token=args['TEMPORARY_SESSION_TOKEN'],
).client('dynamodb')

# Delete table
print('Deleting table: ' + target_ddb_name + ' ...')

try:
    ...
Epics

Set up DynamoDB tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create DynamoDB tables.</td>
<td>Create DynamoDB tables, with indexes, in both source and target AWS accounts.</td>
<td>App developer, DBA, Migration engineer</td>
</tr>
<tr>
<td></td>
<td>Set the capacity provisioning as on-demand mode, which allows DynamoDB to scale read/write capacities dynamically based on the workload.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alternatively, you can use provisioned capacity with 4000 RCU and 4000 WCU.</td>
<td></td>
</tr>
<tr>
<td>Populate the source table.</td>
<td>Populate the DynamoDB table in the source account with test data. Having at least 50 MB or more of test data helps you to see the peak and average RCUs consumed during table copy. You can then change the capacity provisioning as needed.</td>
<td>App developer, DBA, Migration engineer</td>
</tr>
</tbody>
</table>
### Set up credentials to access the DynamoDB tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create IAM users to access the source and target DynamoDB tables.</td>
<td>Create an IAM user in the source account with permissions to access (read) the DynamoDB table in the source account. Create an IAM user in the target account with permissions to access (create, read, update, delete) the DynamoDB table in the target account. Note the AccessKeyId and SecretAccessKey for each user.</td>
<td>App developer, AWS DevOps</td>
</tr>
</tbody>
</table>

### Copy table data from one account to another

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add configuration for the source and target accounts. | In the implementation application's .config file, add entries for AccessKeyId and the path of the file that contains SecretAccessKey. Add these entries for both source and target account users. Secrets (for example, Password and SecretAccessKey) should not be stored in the app .config file. Use one of the following alternatives:  
• Store secrets in a flat file on the server where the code runs and set the file path in the app .config file.  
• Store secrets in AWS Systems Manager Parameter Store, and set the Parameter Store key in the app .config file. | App developer, Migration engineer |
<p>| Initialize the DynamoDB clients for source and target DynamoDB access. | Initialize the DynamoDB clients, which are provided by the AWS SDK, for source and target DynamoDB tables. The application .config file has separate entries for AccessKeyId and SecretAccessKey_FilePath | App developer |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>For the source and target accounts.</td>
<td>Because <code>SecretAccessKey</code> is not directly saved in the <code>.config</code> file, read the secret access keys for source and target accounts from the file paths mentioned in the <code>.config</code> file.</td>
<td><strong>Skills required</strong></td>
</tr>
<tr>
<td></td>
<td>• For the source DynamoDB client, use the <code>AccessKeyId</code> and <code>SecretAccessKey</code> of the source account.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For the target DynamoDB client, use the <code>AccessKeyId</code> and <code>SecretAccessKey</code> of the target account.</td>
<td></td>
</tr>
<tr>
<td>Drop and recreate the target table.</td>
<td>Delete and recreate the target DynamoDB table (along with indexes) in the target account, using the target account DynamoDB client.</td>
<td>App developer</td>
</tr>
<tr>
<td></td>
<td>Deleting all records from a DynamoDB table is a costly operation because it consumes provisioned WCUs. Deleting and recreating the table avoids those extra costs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can add indexes to a table after you create it, but this takes 2–5 minutes longer. Creating indexes during table creation, by passing the indexes collection to the <code>createTable</code> call, is more efficient.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Perform the table copy.</td>
<td>Repeat the following steps until all data are copied:</td>
<td>App developer</td>
</tr>
<tr>
<td></td>
<td>• Perform a scan on the table in the source account, using the source DynamoDB client. Each DynamoDB scan retrieves only 1 MB of data from the table, so you must repeat this operation until all items, or records, are read.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• For each set of scanned items, write the items to the table in the target account, with the target DynamoDB client, using the BatchWriteItem call in AWS SDK for DynamoDB. This reduces the number of PutItem requests made to DynamoDB.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• BatchWriteItem has a limitation of 25 writes or puts, or up to 16 MB. You must add logic to accumulate scanned items in counts of 25 before calling BatchWriteItem. BatchWriteItem returns a list of items that could not be successfully copied. Using this list, add retry logic to perform another BatchWriteItem call with only those items that did not succeed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see the reference implementation in C# (for dropping, creating, and populating tables) in the Attachments section. An example table config JavaScript Object Notation (JSON) file is also attached.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- Amazon DynamoDB documentation
- Creating an IAM user in your AWS account
- AWS SDKs
Additional information

This pattern was implemented using C# to copy a DynamoDB table with 200,000 items (average item size of 5 KB and table size of 250 MB). The target DynamoDB table was set up with provisioned capacity of 4000 RCUs and 4000 WCUs.

The complete table copy operation (from source account to target account), including dropping and recreating the table, took 5 minutes. Total capacity units consumed: 30,000 RCUs and approximately 400,000 WCUs.

For more information on DynamoDB capacity modes, see Read/Write capacity mode in the AWS documentation.

Attachments

attachment.zip

Deploy DataStax Enterprise on AWS

Created by Baji Shaik (AWS)

Summary

This pattern describes how you can deploy DataStax Enterprise (DSE) automatically into an Amazon Web Services (AWS) Cloud configuration of your choice.

DSE is the always-on data platform for cloud applications powered by Apache Cassandra. The DSE platform is designed to handle big data workloads across multiple nodes with no single point of failure. DSE addresses the problem of failures by employing a peer-to-peer distributed system across homogeneous nodes where data is distributed among all nodes in the cluster. DSE offers advanced functionality designed to accelerate your ability to create intelligent and compelling cloud applications. Integrated within each node of DSE is powerful indexing, search, analytics, and graph functionality, provided by combining Cassandra with Apache Solr, Apache Spark, and DSE Graph. You can write data once, and access it using a variety of workloads or access patterns, all from a single cohesive solution.

This pattern leverages the AWS Quick Start developed by DataStax in collaboration with Amazon Web Services (AWS). DataStax is an AWS Partner Network (APN) Partner.

Prerequisites and limitations

- An active AWS account

Architecture

Source technology stack

- Not applicable
Target technology stack

This target architecture incorporates the following components:

- A highly available architecture that spans three Availability Zones.
- A virtual private cloud (VPC) configured with public and private subnets, to provide you with your own virtual network.
- An internet gateway to allow access to the internet.
- Managed network address translation (NAT) gateways to allow outbound internet access for resources in the private subnets.
- One Amazon Elastic Compute Cloud (Amazon EC2) instance running DSE OpsCenter, which is the DSE cluster-management web console.
- Additional Amazon EC2 instances for DSE data centers and nodes. A DSE data center is a logical grouping of nodes for workload separation.
- One Amazon Elastic Block Store (Amazon EBS) data volume per node instance deployed.
Tools

• Quick Start: DataStax Enterprise on AWS

Epics

Access and deploy the Quick Start

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Quick Start, if it meets your needs.</td>
<td>See the Quick Start deployment guide (see the References and Help section) for any pre-deployment instructions, and then launch the Quick Start from the link provided.</td>
<td></td>
</tr>
<tr>
<td>Customize and launch the Quick Start, if you have additional requirements.</td>
<td>Download the AWS CloudFormation templates from the GitHub repository (see the References and Help section), modify them to meet your needs, and then launch the customized templates.</td>
<td>Default areas</td>
</tr>
<tr>
<td>Validate the deployment.</td>
<td>See the Quick Start deployment guide for any post-deployment and testing instructions.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

DSE

• DSE Documentation
• DataStax Academy online courses and tutorials

AWS Quick Starts

• Quick Start: DataStax Enterprise on AWS (data sheet and deployment guide)
• Quick Start: DataStax Enterprise on AWS (source code in GitHub repository)
• AWS Quick Start catalog

Determine whether your Microsoft SQL Server database can be downgraded from Enterprise to Standard edition
Summary

This pattern describes how to determine whether you can downgrade from Microsoft SQL Server Enterprise edition to Standard edition on Amazon Relational Database Service (Amazon RDS) to reduce costs. It doesn’t cover the process for downgrading, but helps you determine whether your database is a downgrade candidate. For downgrading instructions, see the SQL Server documentation from Microsoft.

There are multiple migration paths for reducing SQL Server licensing costs, but this simple process provides a quick win. By using the SQL statement provided in this pattern, you can quickly check whether your Microsoft SQL Server on Amazon RDS edition is a downgrade candidate.

Prerequisites and limitations

Prerequisites

- You have access to SQL Server using SQL tools such as Microsoft SQL Server Management Studio (SSMS)
- The Microsoft SQL Server login user has VIEW DATABASE STATE permissions

Product versions

- Microsoft SQL Server 2008 through 2016

Architecture

You can use the following SQL statement to determine whether your Amazon RDS for Microsoft SQL Server database is using Enterprise edition-specific features:

```sql
SELECT feature_name FROM sys.dm_db_persisted_sku_features;
GO
```

If the view returns no rows, as shown in the following screen illustration, your SQL Server database can be a candidate for downgrading from Enterprise edition to Standard edition.
**Tools**

- **SSMS** – Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

**Epics**

**Validate SQL Server Enterprise edition feature usage**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run a SQL query against SQL Server to determine Enterprise feature usage.</td>
<td>Run the SQL query shown in the Architecture section. If the query returns no rows, you can downgrade your SQL Server database from Enterprise edition to Standard edition.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

**Related resources**

- `sys.dm_db_persisted_sku_features` (Transact-SQL)
- Amazon RDS for Microsoft SQL Server documentation

**Encrypt an existing Amazon RDS for PostgreSQL DB instance**

*Created by Piyush Goyal (AWS)*
Summary

This pattern explains how to encrypt an existing Amazon Relational Database Service (Amazon RDS) for PostgreSQL DB instance in the Amazon Web Services (AWS) Cloud with minimal downtime. (This process works for Amazon RDS for MySQL DB instances as well.)

You can enable encryption for an Amazon RDS DB instance when you create it, but not after it’s created. However, you can add encryption to an unencrypted DB instance by creating a snapshot of your DB instance, and then creating an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot to get an encrypted copy of your original DB instance. The pattern uses AWS Database Migration Service (AWS DMS) to migrate data and AWS Key Management Service (AWS KMS) for encryption.

Amazon RDS encrypted DB instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS DB instances. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently, with minimal impact on performance. You don’t need to modify your database client applications to use encryption.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An unencrypted Amazon RDS for PostgreSQL DB instance
- Experience working with (creating, modifying, or stopping) AWS DMS tasks (see Working with AWS DMS tasks in the AWS DMS documentation)
- Familiarity with AWS KMS for encrypting databases (see the AWS KMS documentation)

Limitations

- You can enable encryption for an Amazon RDS DB instance only when you create it, not after the DB instance is created.
- You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance.
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
- Sequences aren't migrated to the new, encrypted DB instance.

For more information, see Limitations of Amazon RDS encrypted DB instances in the Amazon RDS documentation.

Architecture

Source architecture

- Unencrypted RDS DB instance

Target architecture
• Encrypted RDS DB instance
  • The destination RDS DB instance is created by restoring the DB snapshot copy of the source RDS DB instance.
  • An AWS KMS key is used for encryption while restoring the snapshot.
  • An AWS DMS replication task is used to migrate the data.

Tools

Tools used to enable encryption:
  • AWS KMS key for encryption – When you create an encrypted DB instance, you can choose a customer managed key or the AWS managed key for Amazon RDS to encrypt your DB instance. If you don’t
specify the key identifier for a customer managed key, Amazon RDS uses the AWS managed key for your new DB instance. Amazon RDS creates an AWS managed key for Amazon RDS for your AWS account. Your AWS account has a different AWS managed key for Amazon RDS for each AWS Region. For more information about using KMS keys for Amazon RDS encryption, see Encrypting Amazon RDS Resources.

Tools used for ongoing replication:
- AWS DMS – You can use AWS Database Migration Service (AWS DMS) to replicate changes from the source DB to the target DB. It is important to keep the source and target DB in sync to keep downtime to a minimum. For information about setting up AWS DMS and creating tasks, see the AWS DMS documentation.

Epics
Create a snapshot of the source DB instance and encrypt it

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check the details for the source PostgreSQL DB instance.</td>
<td>On the Amazon RDS console, choose the source PostgreSQL DB instance. On the Configuration tab, make sure that encryption isn’t enabled for the instance. For a screen illustration, see the Additional information section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create the DB snapshot.</td>
<td>Create a DB snapshot of the instance you want to encrypt. The amount of time it takes to create a snapshot depends on the size of your database. For instructions, see Creating a DB snapshot in the Amazon RDS documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Encrypt the snapshot.</td>
<td>In the Amazon RDS console navigation pane, choose Snapshots, and select the DB snapshot you created. For Actions, choose Copy Snapshot. Provide the destination AWS Region and the name of the DB snapshot copy in the corresponding fields. Select the Enable Encryption checkbox. For Master Key, specify the AWS KMS key identifier to use to encrypt the DB snapshot copy. Choose Copy Snapshot. For more information, see Copying a snapshot in the Amazon RDS documentation.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
**Prepare the target DB instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restore the DB snapshot.</td>
<td>On the Amazon RDS console, choose <strong>Snapshots</strong>. Choose the encrypted snapshot that you created. For <strong>Actions</strong>, choose <strong>Restore Snapshot</strong>. For <strong>DB Instance Identifier</strong>, provide a unique name for the new DB instance. Review the instance details, and then choose <strong>Restore DB Instance</strong>. A new, encrypted DB Instance will be created from your snapshot. For more information, see <em>Restoring from a DB snapshot</em> in the Amazon RDS documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Migrate data by using AWS DMS.</td>
<td>On the AWS DMS console, create an AWS DMS task. For <strong>Migration type</strong>, choose <strong>Migrate existing data and replicate ongoing changes</strong>. In <strong>Task Settings</strong>, for <strong>Target table preparation mode</strong>, choose <strong>Truncate</strong>. For more information, see <em>Creating a task</em> in the AWS DMS documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Enable data validation.</td>
<td>In <strong>Task Settings</strong>, choose <strong>Enable validation</strong>. This enables you to compare the source data to the target data to verify that the data was migrated accurately.</td>
<td>DBA</td>
</tr>
<tr>
<td>Disable constraints on the target DB instance.</td>
<td>Disable any triggers and foreign key constraints on the target DB instance, and then start the AWS DMS task.</td>
<td>DBA</td>
</tr>
<tr>
<td>Verify data.</td>
<td>After the full load is complete, verify the data on the target DB instance to see if it matches the source data. For more information, see <em>AWS DMS data validation</em> in the AWS DMS documentation.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Cut over to the target DB instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop write operations on the source DB instance.</td>
<td>Stop the write operations on the source DB instance so that</td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>application downtime can begin. Verify that AWS DMS has completed the replication for the data in the pipeline. Enable triggers and foreign keys on the target DB instance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configure the application endpoint.</td>
<td>DBA, Application owner</td>
</tr>
<tr>
<td></td>
<td>Configure your application connections to use the new Amazon RDS DB instance endpoints. The DB instance is now encrypted.</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

- Creating an AWS DMS task
- Monitoring replication tasks using Amazon CloudWatch
- Monitoring AWS DMS tasks

### Additional information

Checking the encryption for the source PostgreSQL DB instance:

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DB identifier</td>
<td>CPU</td>
<td>Status</td>
<td>Class</td>
</tr>
<tr>
<td>rds-test</td>
<td>3.17%</td>
<td>Available</td>
<td>db.t2.micro</td>
</tr>
<tr>
<td>Role</td>
<td>Current activity</td>
<td>Engine</td>
<td>Region &amp; AZ</td>
</tr>
<tr>
<td>Instance</td>
<td>0 Sessions</td>
<td>PostgreSQL</td>
<td>us-east-1a</td>
</tr>
</tbody>
</table>

**Instance**

- Configuration
  - DB instance id: rds-test
  - Instance class: db.t2.micro

**Storage**

- Encryption: Not Enabled

**Performance Insights**

- Performance Insights enabled: Yes

Additional notes for this pattern:

- Enable replication on PostgreSQL by setting the `rds.logical_replication` parameter to 1.

**Important note:** Replication slots retain the write ahead log (WAL) files until the files are externally consumed—for example, by `pg_recvlogical`; by extract, transform, and load (ETL) jobs; or by AWS DMS. When you set the `rds.logical_replication` parameter value to 1, AWS DMS sets the `wal_level`, `max_wal_senders`, `max_replication_slots`, and `max_connections` parameters. If logical replication slots are present but there is no consumer for the WAL files retained by the replication slot, you might see an increase in the transaction log disk usage and a constant decrease in free storage space. For more information and steps to resolve this issue, see the article How can I identify what is
causing the "No space left on device" or "DiskFull" error on Amazon RDS for PostgreSQL? in the AWS Support Knowledge Center.

- Any schema changes that you make to the source DB instance after you create the DB snapshot will not be present on the target DB instance.
- After you create an encrypted DB instance, you can't change the CMK used by that DB instance. Be sure to determine your CMK requirements before you create your encrypted DB instance.
- You must disable triggers and foreign keys on the target DB instance before you run the AWS DMS task. You can re-enable these when the task is complete.

Enforce automatic tagging of Amazon RDS databases at launch

Created by Susanne Kangnoh (AWS)

| Environment: Production | Technologies: Databases; Cloud-native; Security, identity, compliance | AWS services: Amazon RDS; Amazon SNS; AWS CloudTrail; Amazon CloudWatch |

Summary

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the Amazon Web Services (AWS) Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

You can use tagging to categorize your AWS resources in different ways. Relational database tagging is useful when you have many resources in your account and you want to quickly identify a specific resource based on the tags. You can use Amazon RDS tags to add custom metadata to your RDS DB instances. A tag consists of a user-defined key and value. We recommend that you create a consistent set of tags to meet your organization's requirements.

This pattern provides an AWS CloudFormation template to help you monitor and tag RDS DB instances. The template creates an Amazon CloudWatch Events event that watches for the AWS CloudTrail CreateDBInstance event. (CloudTrail captures API calls for Amazon RDS as events.) When it detects this event, it calls an AWS Lambda function that automatically applies tag keys and values that you define. The template also sends out a notification that the instance has been tagged, by using Amazon Simple Notification Service (Amazon SNS).

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the Lambda code.
- An email address where you would like to receive tagging notifications.
Limitations

• The solution supports CloudTrail **CreateDBInstance** events. It does not create notifications for any other events.

Architecture

Workflow architecture

Automation and scale

• You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run the template only once in each Region or account.

Tools

AWS services

• **AWS CloudTrail** – AWS CloudTrail is an AWS service that helps you with governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.

• **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

• **AWS Lambda** – AWS Lambda is a compute service that supports running code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume —there is no charge when your code is not running.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

• **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud.

Code

This pattern includes an attachment with two files:

• **index.zip** is a compressed file that includes the Lambda code for this pattern.

• **rds.yaml** is a CloudFormation template that deploys the Lambda code.

See the Epics section for information about how to use these files.
## Epics

### Deploy the Lambda code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the code to an S3 bucket.</td>
<td>Create a new S3 bucket or use an existing S3 bucket to upload the attached <code>index.zip</code> file (Lambda code). This bucket must be in the same AWS Region as the resources (RDS DB instances) that you want to monitor.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the <code>rds.yaml</code> file that's provided in the attachment. In the next epic, provide values for the template parameters.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

### Complete the parameters in the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the RDS DB instances that you want to monitor.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, <code>index.zip</code> or <code>controls/index.zip</code>).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address where you want to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Specify a logging level.</td>
<td>Specify the logging level and verbosity. <strong>Info</strong> designates detailed informational messages on the application's progress and should be used only for debugging. <strong>Error</strong> designates error events that could still allow the application to continue.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Enter the tag keys and values for your RDS DB instances.

Enter the required tag keys and values that you want to automatically apply to the RDS instance. For more information, see Tagging Amazon RDS resources in the AWS documentation.

Cloud architect

Confirm the subscription

Confirm the email subscription.

When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications when your instances are tagged, you must confirm this email subscription.

Cloud architect

Related resources

- Creating a bucket (Amazon S3 documentation)
- Tagging Amazon RDS resources (Amazon Aurora documentation)
- Uploading objects (Amazon S3 documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (Amazon CloudWatch documentation)

Attachments

attachment.zip

Help enforce DynamoDB tagging

Created by Mansi Suratwala (AWS)

Environment: Production
Technologies: Cloud-native; Security, identity, compliance; Databases
Workload: All other workloads

AWS services: Amazon CloudWatch; Amazon
Summary

This pattern sets up automatic notifications when a predefined Amazon DynamoDB tag is missing or removed from a DynamoDB resource on the Amazon Web Services (AWS) Cloud.

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database. When you use DynamoDB, you don’t have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.

The pattern uses an AWS CloudFormation template, which creates an Amazon CloudWatch Events event and an AWS Lambda function. The event watches for any new or existing DynamoDB tagging information via AWS CloudTrail. If a predefined tag is missing or removed, CloudWatch triggers a Lambda function, which sends you an Amazon Simple Notification Service (Amazon SNS) notification informing you of the violation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Simple Storage Service (Amazon S3) bucket for the Lambda .zip file that contains the Python script for running the Lambda function

Limitations

- The solution works only when the TagResource or UntagResource CloudTrail events occur. It does not create notifications for any other events.

Architecture

Target technology stack

- Amazon DynamoDB
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon S3
- Amazon SNS

Target architecture
Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run the template only once in each Region or account.

Tools

- Amazon DynamoDB – DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with scalability.
- AWS CloudTrail – CloudTrail is an AWS service that helps you with governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near-real time stream of system events that describe changes in AWS resources.
- AWS Lambda – Lambda is a compute service that supports running code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud.

Code

- A .zip file of the project is available as an attachment.

Epics

Define the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. This</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
### Upload the Lambda code to the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code to the S3 bucket.</td>
<td>Upload the Lambda code .zip file provided in the “Attachments” section to the S3 bucket. The S3 bucket must be in the same Region as the DynamoDB resource that is being monitored.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

### Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>On the AWS CloudFormation console, deploy the AWS CloudFormation template that's provided in the “Attachments” section. In the next epic, provide values for the parameters.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

### Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Enter the name of the S3 bucket that you created or chose in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the Amazon S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, &lt;folder&gt;/&lt;filename&gt;.zip).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide an email address</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. “Info” designates</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td></td>
</tr>
<tr>
<td>Enter the required DynamoDB tag keys.</td>
<td>Be sure that the tags are separated by commas, with no spaces between them (for example, ApplicationId,CreatedBy,Environment,Organization). The CloudWatch Events event searches for these tags and sends a notification if they are not found.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Confirm the subscription.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email to the email address that you provided. To receive violation notifications, you must confirm this email subscription.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating an S3 bucket
- Uploading files to an S3 bucket
- Tagging resources in DynamoDB
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail

Attachments

attachment.zip

Implement cross-Region disaster recovery with AWS DMS and Amazon Aurora

*Created by Mark Hudson (AWS)*
Summary

Natural or human-induced disasters can occur at any time and can impact the availability of services and workloads running in a given Amazon Web Services (AWS) Region. To mitigate the risks, you must develop a disaster recovery (DR) plan that incorporates the built-in cross-Region capabilities of AWS services. For AWS services that do not inherently provide cross-Region functionality, the DR plan must also provide a solution to handle their failover across AWS Regions.

This pattern guides you through a disaster recovery setup involving two Amazon Aurora MySQL-Compatible Edition database clusters in a single Region. To meet DR requirements, the database clusters are configured to use the Amazon Aurora global database feature, with a single database spanning multiple AWS Regions. An AWS Database Migration Service (AWS DMS) task replicates data between the clusters in the local Region. AWS DMS, however, currently doesn't support cross-Region tasks. This pattern includes the steps required to work around that limitation and independently configure AWS DMS in both Regions.

Prerequisites and limitations

Prerequisites

- Selected primary and secondary AWS Regions that support Amazon Aurora global databases.
- Two independent Amazon Aurora MySQL-Compatible Edition database clusters in a single account in the primary Region.
- Database instance class db.r5 or higher (recommended).
- An AWS DMS task in the primary Region performing ongoing replication between the existing database clusters.
- DR Region resources in place to meet requirements for creating database instances. For more information, see Working with a DB instance in a VPC.

Limitations

- For the full list of Amazon Aurora global database limitations, see Limitations of Amazon Aurora global databases.

Product versions

- Amazon Aurora MySQL-Compatible Edition 5.6, 5.7, or 8.0. For more information, see Aurora global databases.

Architecture

Target technology stack

- Amazon Aurora MySQL-Compatible Edition global database cluster
- AWS DMS

Target architecture
Automation and scale

You can use AWS CloudFormation to create the prerequisite infrastructure in the secondary Region, such as the virtual private cloud (VPC), subnets, and parameter groups. You can also use AWS CloudFormation to create the secondary clusters in the DR Region and add them to the global database. If you used CloudFormation templates to create the database clusters in the primary Region, you can update or augment them with an additional template to create the global database resource. For more information, see Creating an Amazon Aurora DB cluster with two DB instances and Creating a global database cluster for Aurora MySQL-Compatible.

Finally, you can create the AWS DMS tasks in the primary and secondary Regions using CloudFormation after failover and failback events occur. For more information, see AWS::DMS::ReplicationTask.

Tools

- **Amazon Aurora** - Amazon Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. This pattern uses Amazon Aurora MySQL-Compatible Edition.

- **Amazon Aurora global databases** - Amazon Aurora global databases are designed for globally distributed applications. A single Amazon Aurora global database can span multiple AWS Regions. It replicates your data with no impact on database performance. It also enables fast local reads with low latency in each Region, and it provides disaster recovery from Region-wide outages.

- **AWS DMS** - AWS Database Migration Service (AWS DMS) provides one-time migration or on-going replication. An on-going replication task keeps your source and target databases in sync. After it is set up, the on-going replication task continuously applies source changes to the target with minimal latency. All AWS DMS features, such as data validation and transformations, are available for any replication task.
## Epics

### Prepare the existing database clusters in the primary Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify the database cluster parameter group.</td>
<td>In the existing database cluster parameter group, activate row-level binary logging by setting the <code>binlog_format</code> parameter to a value of <code>row</code>. AWS DMS requires row-level binary logging for MYSQL-compatible databases when performing ongoing replication or change data capture (CDC). For more information, see Using an AWS managed MySQL-compatible database as a source for AWS DMS.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Update the database binary log retention period.</td>
<td>Using a MySQL client installed on your end-user device or an Amazon Elastic Compute Cloud (Amazon EC2) instance, run the following stored procedure provided by Amazon Relational Database Service (Amazon RDS) on the main database cluster's writer node, where <code>XX</code> is the number of hours to retain the logs. MySQL-compatible databases managed by AWS purge the binary logs as soon as possible. Therefore, the retention period must be long enough to ensure that the logs are not purged before the AWS DMS task runs. A value of 24 hours is usually sufficient, but the value should be based on the time required to</td>
<td>DBA</td>
</tr>
</tbody>
</table>

```sql
call mysql.rds_set_configuration('binlog_retention_hours', XX)
```

Confirm the setting by running the following command.

```sql
call mysql.rds_show_configuration;
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>set up the AWS DMS task in the DR Region.</td>
<td></td>
</tr>
</tbody>
</table>

**Update the existing AWS DMS task in the primary Region**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record the AWS DMS task ARN.</td>
<td>Use the Amazon Resource Name (ARN) to obtain the AWS DMS task name for later use. To retrieve the AWS DMS task ARN, view the task in the console or run the following command.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>\texttt{aws dms describe-replication-tasks}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An ARN looks like the following.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\texttt{arn:aws:dms:us-east-1:&lt;accountid&gt;:task:AN6HFFMMPM246XOZVEUHCNS0VF7MQCL70ZUIRAMY}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The characters after the last colon correspond to the task name used in a later step.</td>
<td></td>
</tr>
<tr>
<td>Modify the existing AWS DMS task to record the checkpoint.</td>
<td>AWS DMS creates checkpoints that contain information so that the replication engine knows the recovery point for the change stream. To record checkpoint information, perform the following steps in the console:</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>1. Stop the AWS DMS task.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Use the JSON editor in the task to set the TaskRecoveryTableEnabled parameter to \texttt{true}.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Start the AWS DMS task.</td>
<td></td>
</tr>
<tr>
<td>Validate checkpoint information.</td>
<td>Using a MySQL client connected to the writer endpoint for the cluster, query the new metadata table in the reporter database cluster to verify that it exists and contains the replication state information. Run the following command.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>\texttt{select * from awsdms_control.awsdms_txn_state;}</td>
<td></td>
</tr>
</tbody>
</table>
## Expand both Amazon Aurora clusters to a DR Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create base infrastructure in the DR Region.</td>
<td>Create the base components required for the creation of and access to the Amazon Aurora clusters: • Virtual private cloud (VPC) • Subnets • Security group • Network access control lists • Subnet group • DB parameter group • DB cluster parameter group</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>Ensure that the configuration of both parameter groups matches the configuration in the primary Region.</td>
<td></td>
</tr>
<tr>
<td>Add the DR Region to both Amazon Aurora clusters.</td>
<td>Add a secondary Region (the DR Region) to the main and reporter Amazon Aurora clusters. For more information, see <a href="https://docs.aws.amazon.com/AmazonAurora/latest/ug/working-with-regions.html">Adding an AWS Region to an Amazon Aurora global database</a></td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>

## Perform failover

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop the AWS DMS task.</td>
<td>The AWS DMS task in the primary Region will not function properly after failover occurs and should be stopped to avoid errors.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Perform a managed failover.</td>
<td>Perform a managed failover of the main database cluster to the DR Region. For instructions, see <a href="https://docs.aws.amazon.com/AmazonAurora/latest/ug/working-with-regions.html">Performing managed planned failovers for Amazon Aurora global databases</a>. After failover on the main database</td>
<td>AWS administrator, DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Cluster is complete, perform the same activity on the reporter database cluster.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load data into the main database.</td>
<td>Insert test data into writer node of the main database in the DR database cluster. This data will be used to validate that replication is functioning properly.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create the AWS DMS replication instance.</td>
<td>To create the AWS DMS replication instance in the DR Region, see Creating a replication instance.</td>
<td>AWS administrator, DBA</td>
</tr>
<tr>
<td>Create the AWS DMS source and target endpoints.</td>
<td>To create the AWS DMS source and target endpoints in the DR Region, see Creating source and target endpoints. The source should point to the writer instance of the main database cluster. The target should point to the writer instance of the reporter database cluster.</td>
<td>AWS administrator, DBA</td>
</tr>
</tbody>
</table>
| Obtain the replication checkpoint. | To obtain the replication checkpoint, use a MySQL client to query the metadata table by running the following against the writer node in the reporter database cluster in the DR Region. 

```sql
select * from awsdms_control.awsdms_txn_state;
```

In the table, find the task_name value that corresponds to the AWS DMS task's ARN that exists in the primary Region that you obtained in the second epic. | DBA |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an AWS DMS task. | Using the console, create an AWS DMS task in the DR Region. In the task, specify a migration method of **Replicate data changes only**. For more information, see *Creating a task*.  
1. In the task settings, use the wizard to specify the following:  
   - **CDC start mode for source transactions** – Enable custom CDC start mode  
   - **Custom CDC start point for source transactions** – Specify a recovery checkpoint  
2. In the **Recovery checkpoint** box, enter the replication checkpoint value previously obtained through the database query on the `awsdms_txn_state` table.  
3. In the task settings section, select the JSON editor, and set the **TaskRecoveryTableEnabled** parameter to **true**.  
   Set the AWS DMS task **Start migration task** setting to **Automatically on create**. | AWS administrator, DBA |
| Record the AWS DMS task ARN. | Use the ARN to obtain the AWS DMS task name for later use. To retrieve the AWS DMS task ARN, run the following command.  
```bash  
aws dms describe-replication-tasks  
``` | AWS administrator, DBA |
| Validate the replicated data. | Query the reporter database cluster in the DR Region to confirm that the test data that you loaded into the main database cluster has been replicated. | DBA |
### Perform failback

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop the AWS DMS task.</td>
<td>The AWS DMS task in the DR Region will not function properly after failback occurs and should be stopped to avoid errors.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Perform a managed failback.</td>
<td>Fail back the main database cluster to the primary Region. For instructions, see <a href="https://docs.aws.amazon.com/databasemigration/latest/dg/managed-planned-failovers.html">Performing managed planned failovers for Amazon Aurora global databases</a>. After the failback on the main database cluster is complete, perform the same activity on the reporter database cluster.</td>
<td>AWS administrator, DBA</td>
</tr>
<tr>
<td>Obtain the replication checkpoint.</td>
<td>To obtain the replication checkpoint, use a MySQL client to query the metadata table by running the following against the writer node in the reporter database cluster in the DR Region.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
|                                               | ```sql
select * from awsdms_control.awsdms_txn_state;
```
<p>|                                               | In the table, find the <code>task_name</code> value that corresponds to the AWS DMS task's ARN that exists in the DR Region that you obtained in the fourth epic.                                                         |                               |
| Update the AWS DMS source and target endpoints.| After the database clusters have failed back, check the clusters in the primary Region to determine which nodes are the writer instances. Then verify the existing AWS DMS source and target endpoints in the primary Region are pointing to the writer instances. If not, update the endpoints with the writer instance Domain Name System (DNS) names. | AWS administrator             |
| Create an AWS DMS task.                        | Using the console, create an AWS DMS task in the primary Region. In the task, specify a migration method of <strong>Replicate data changes only</strong>. For more information, see <a href="https://docs.aws.amazon.com/databasemigration/latest/dg/creating-tasks.html">Creating a task</a>. | AWS administrator, DBA       |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the task settings, use the wizard and specify the following:</td>
<td></td>
<td>AWS administrator, DBA</td>
</tr>
<tr>
<td>• <strong>CDC start mode for source transactions</strong> – Enable custom CDC start mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• <strong>Custom CDC start point for source transactions</strong> – Specify a recovery checkpoint</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. In the <strong>Recovery checkpoint</strong> box, enter the replication checkpoint value previously obtained through the database query on the <code>awsdms_txn_state</code> table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Also within the task settings section, select the JSON editor and set the <strong>TaskRecoveryTableEnabled</strong> parameter to <code>true</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Finally, set the AWS DMS task <strong>Start migration task</strong> setting to <strong>Automatically on create</strong>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Record the AWS DMS task Amazon Resource Name (ARN).**

Use the ARN to obtain the AWS DMS task name for later use. To retrieve the AWS DMS task ARN, run the following command:

```
aws dms describe-replication-tasks
```

The task name will be needed when performing another managed failover or during a DR scenario.

**Delete AWS DMS tasks.**

Delete the original (currently stopped) AWS DMS task in the primary Region and the existing AWS DMS task (currently stopped) in the secondary Region.

**Related resources**

- Configuring your Amazon Aurora DB cluster
- Using Amazon Aurora global databases
- Working with Amazon Aurora MySQL-Compatible Edition
• Working with an AWS DMS replication instance
• Working with AWS DMS endpoints
• Working with AWS DMS tasks
• What is AWS CloudFormation?

Additional information

Amazon Aurora global databases are used in this example for DR because they provide an effective recovery time objective (RTO) of 1 second and a recovery point objective (RPO) of less than 1 minute, both lower than traditional replicated solutions and ideal for DR scenarios.

Amazon Aurora global databases offer many other advantages, including the following:

• Global reads with local latency – Global consumers can access information in a local Region, with local latency.
• Scalable secondary Amazon Aurora DB clusters – Secondary clusters can be scaled independently, adding up to 16 read-only replicas.
• Fast replication from primary to secondary Amazon Aurora DB clusters – Replication has little performance impact on the primary cluster. It occurs at the storage layer, with typical cross-Region replication latencies of less than 1 second.

This pattern also uses AWS DMS for replication. Amazon Aurora databases provide the ability to create read replicas, which can simplify the replication process and the DR setup. However, AWS DMS is often used to replicate when data transformations are required or when the target database requires additional indexes that the source database does not have.

Migrate SAP HANA to AWS using SAP HSR with the same hostname

*Created by Pradeep Puliyampatta (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Source:</th>
<th>SAP HANA DB on-premises</th>
<th>Target:</th>
<th>SAP HANA DB on AWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Rehost</td>
<td>Workload:</td>
<td>SAP</td>
<td>Technologies:</td>
<td>Databases; Migration</td>
</tr>
</tbody>
</table>

**AWS services:** AWS Client VPN; AWS Direct Connect; Amazon EBS

Summary

SAP HANA migrations to Amazon Web Services (AWS) can be performed using multiple options, including backup and restore, export and import, and SAP HANA System Replication (HSR). The selection of a particular option depends on the network connectivity between source and target SAP HANA databases, the size of the source database, downtime considerations, and other factors.
The SAP HSR option for migrating SAP HANA workloads to AWS works well when there is a stable network between the source and target systems and the entire database (SAP HANA DB replication snapshot) can be completely replicated within 1 day, as stipulated by SAP for network throughput requirements for SAP HSR. The downtime requirements with this approach are limited to performing the takeover on the target AWS environment, SAP HANA DB backup, and post-migration tasks.

SAP HSR supports the use of different hostnames (hostnames mapped to different IP addresses) for replication traffic between the primary, or source, and secondary, or target, systems. You can do this by defining those specific sets of hostnames under the `[system_replication_hostname_resolution]` section in `global.ini`. In this section, all hosts of the primary and the secondary sites must be defined on each host. For detailed configuration steps, see the SAP documentation.

One key takeaway from this setup is that the hostnames in the primary system must be different from the hostnames in the secondary system. Otherwise, the following errors can be observed.

- "each site must have a unique set of logical hostnames"
- "remoteHost does not match with any host of the source site. All hosts of source and target site must be able to resolve all hostnames of both sites correctly"

However, the number of post-migration steps can be reduced by using the same SAP HANA DB hostname on the target AWS environment.

This pattern provides a workaround for using the same hostname on source and target environments when using the SAP HSR option. With this pattern, you can use the SAP HANA Hostname Rename option. You assign a temporary hostname to the target SAP HANA DB to facilitate hostname uniqueness for SAP HSR. After the migration completes the takeover milestone on the target SAP HANA environment, you can revert the target system hostname back to the hostname of the source system.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- A virtual private cloud (VPC) with a virtual private network (VPN) endpoint or a router.
- AWS Client VPN or AWS Direct Connect configured to transfer files from the source to the target.
- SAP HANA databases in both the source and the target environment. The target SAP HANA DB patch level should be equal to or higher than the source SAP HANA DB patch level, within the same SAP HANA Platform edition. For example, replication cannot be set up between HANA 1.0 and HANA 2.0 systems. For more information, see question 15 in SAP Note: 1999880 – FAQ: SAP HANA System Replication.
- SAP application servers in the target environment.
- Amazon Elastic Block Store (Amazon EBS) volumes in the target environment.

**Limitations**

The following list of SAP documents covers known issues that are related to this workaround, including constraints regarding SAP HANA dynamic tiering and scale-out migrations:

- 2956397 – Renaming of SAP HANA Database System failed
- 2222694 – When trying to rename the HANA system, the following error appears "Source files are not owned by the original sidadm user (uid = xxxx)"
- 2607227 – hdblcm: register_rename_system: Renaming SAP HANA instance failed
- 2630562 – HANA Hostname Rename failed and HANA does not start up
• 2935639 – sr_register is not using the hostname that is specified under system_replication_hostname_resolution in the global.ini section
• 2710211 – Error: source system and target system have overlapping logical hostnames
• 2693441 – Failed to rename an SAP HANA System due to error
• 2519672 – HANA Primary and Secondary has different system PKI SSFS data and key or unable to check
• 2457129 – SAP HANA System Host Rename is not Permitted when Dynamic Tiering is Part of Landscape
• 2473002 – Using HANA System Replication to migrate scale out system (There are no restrictions provided by SAP in using this hostname rename approach for scale-out SAP HANA systems. However, the procedure must be repeated on each individual host. Other scale-out migration limitations also apply to this approach.)

Product versions
• This solution applies to SAP HANA DB platform edition 1.0 and 2.0.

Architecture

Source setup

An SAP HANA database is installed on the source environment. All the SAP application server connections and DB interfaces use the same hostname for client connections. The following diagram shows the example source hostname hdbhost and its corresponding IP address.

<table>
<thead>
<tr>
<th>Source details</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.2.1</td>
</tr>
<tr>
<td>hdbhost</td>
</tr>
</tbody>
</table>

Target setup

The AWS Cloud target environment uses the same hostname to run an SAP HANA database. The target environment on AWS includes the following:
**AWS Prescriptive Guidance Patterns**

**Architecture**

- SAP HANA database
- SAP application servers
- EBS volumes

**Intermediate configuration**

In the following diagram, the hostname on the AWS target environment is temporarily renamed as `temp-host` so that the hostnames on the source and target are unique. After the migration completes the takeover milestone on the target environment, the target system virtual hostname is renamed using the original name, `hdbhost`.

The intermediate configuration includes one of the following options:

- AWS Client VPN with a Client VPN endpoint
- AWS Direct Connect connecting to a router
SAP application servers on the AWS target environment can be installed either before replication setup or after the takeover. However, installing the application servers before replication setup can help with reduction of downtime during installation, configuration of high availability, and backups.

**Tools**

**AWS services**

- **AWS Client VPN** – AWS Client VPN is a managed client-based VPN service that enables you to securely access AWS resources and resources in your on-premises network.

- **AWS Direct Connect** – AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services, bypassing internet service providers in your network path.

- **Amazon EBS** – Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances.

**Other tools**

- **SAP application servers** – SAP application servers provide programmers with a way to express business logic. The SAP application server performs the data processing based on the business logic. The actual data is stored in a database, which is a separate component.

- **SAP HANA cockpit** and **SAP HANA Studio** – Both SAP HANA cockpit and SAP HANA Studio provide an administrative interface to the SAP HANA database. In SAP HANA Studio, the SAP HANA Administration console is the system view that provides relevant content for SAP HANA database administration.

- **SAP HANA System Replication** – SAP HANA System Replication (SAP HSR) is the standard procedure provided by SAP for replicating SAP HANA databases. The required executables for SAP HSR are part of the SAP HANA server kernel itself.
## Epics

### Prepare the source and target environments

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Install and configure the SAP HANA databases.</strong></td>
<td>In the source and target environments, ensure that the SAP HANA DB is installed and configured according to SAP HANA on AWS best practices. For more information, see <a href="https://aws.amazon.com/sap-hana/">SAP HANA on AWS</a>.</td>
<td>SAP Basis administration</td>
</tr>
</tbody>
</table>
| **Map the IP address.** | In the target environment, ensure that the temporary hostname is assigned to an internal IP address.  
1. Assign a secondary IPv4 address to the Amazon Elastic Compute Cloud (Amazon EC2) instance on the AWS Management Console by navigating to **EC2, Instance, Actions, Networking, Manage IP address, Assign new IP address.**  
2. To assign the same address to the EC2 network adaptor (NIC), from the operating system, as root user, run the command `ip addr add <IP>/32 dev eth0`, replacing `<IP>` with the IP address from step 1. | AWS administration |
<p>| <strong>Resolve target hostnames.</strong> | On the secondary SAP HANA DB, confirm that both hostnames (<code>hdbhost</code> and <code>temp-host</code>) are resolved for the SAP HANA replication networks by updating the relevant hostnames in the <code>/etc/hosts</code> file. | Linux administration |
| <strong>Back up the source and target SAP HANA databases.</strong> | Use SAP HANA Studio or the SAP HANA cockpit to perform backups on the SAP HANA databases. | SAP Basis administration |
| <strong>Exchange system PKI certificates.</strong> | (Applies only to SAP HANA 2.0 and later) Exchange certificates in the system public key infrastructure (PKI) secure store in the file system (SSFS) store between the primary | SAP Basis administration |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rename the target SAP HANA DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop target client connections.</td>
<td>In the target environment, shut down the SAP application servers and other client connections.</td>
<td>SAP Basis administration</td>
</tr>
</tbody>
</table>
| Rename the target SAP HANA DB to the temporary hostname. | 1. As root user, rename the target SAP HANA DB hostname to the temporary hostname by using resident hdblcm.  

```bash
root #> cd /hana/shared/<SID>/hdblcm
root #> ./hdblcm
```

2. Choose option 9 | rename_system | Rename the SAP HANA Database System.  
3. Provide the new name: temp-host.  
4. You can validate other options as needed. However, be sure that you don’t mix up the host rename with a SID change (SAP Note 2598814 – hdblcm: SID rename fails).  

The SAP HANA DB stop and start will be controlled by hdblcm. | SAP Basis administration |
<p>| Assign replication networks.             | In the global.ini file of the source system, under the [system_replication_hostname_resolution] header, provide the source and target replication network details. Then copy the entries to the global.ini file on the target system. | SAP Basis administration |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable replication on primary.</td>
<td>To enable replication on the source SAP HANA DB, run the following command.</td>
<td>SAP Basis administration</td>
</tr>
<tr>
<td></td>
<td><code>hdbnsutil -sr_enable --name=siteA</code></td>
<td></td>
</tr>
<tr>
<td>Register the target SAP HANA DB as a secondary system.</td>
<td>To register the target SAP HANA DB as a secondary system to source for SAP HSR, choose async replication.</td>
<td>SAP Basis administration</td>
</tr>
<tr>
<td></td>
<td><code>(sid)adm $&gt; HDB stop</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>(sid)adm $&gt; hdbnsutil -sr_register --name=siteB --remotehost=hdbhost / --remoteInstance=00 --replicationMode=async --operationMode=logreplay</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>(sid)adm $&gt; HDB start</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alternatively, you can choose the --online option to register. In that case, you don't need to stop and start the SAP HANA DB.</td>
<td></td>
</tr>
<tr>
<td>Validate synchronization.</td>
<td>On the source SAP HANA DB, verify that all the logs are applied on the target system (because it is async replication).</td>
<td>SAP Basis administration</td>
</tr>
<tr>
<td></td>
<td>To verify the replication, on the source, run the following commands.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>(sid)adm $&gt; cdpy</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>(sidadm $&gt; python systemReplicationStatus.py</code></td>
<td></td>
</tr>
<tr>
<td>Shut down the source SAP application and SAP HANA DB.</td>
<td>During the migration cutover, perform a shutdown of the source system (the SAP application and SAP HANA database.</td>
<td>SAP Basis administration</td>
</tr>
<tr>
<td>Perform a takeover at the target.</td>
<td>To perform a takeover at the target on AWS, run the command hdbnsutil -sr_takeover.</td>
<td>SAP Basis administration</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| On the target SAP HANA DB, turn off replication. | To clear the replication metadata, stop replication on the target system by running the command `hdbnsutil -sr_disable`.  
**Note:** This is in accordance with SAP Note 2693441 – Failed to rename an SAP HANA System due to error. | SAP Basis administration |
| Back up the target SAP HANA DB. | After the takeover is successful, we recommend performing a full SAP HANA DB backup. | SAP Basis administration |

### Revert to the original hostname in the target system

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Revert the target SAP HANA DB hostname to the original. | 1. To revert the target SAP HANA DB hostname to the original virtual hostname, use resident hdblcm.  
```bash
root #> cd /hana/shared/<SID>/hdblcm
root #> ./hdblcm
```
2. Choose option 9 | rename_system | Rename the SAP HANA Database System.  
3. Provide the new name: hdbhost.  
You can validate other options as needed. However, be sure that you don’t mix up the host rename with a SID change (SAP Note 2598814 – hdblcm: SID rename fails). | SAP Basis administration |
| Adjust hdbuserstore. | Adapt the hdbuserstore details pointing to the source schema/user details. For detailed steps, see the SAP documentation.  
To validate this step, run the command `R3trans -d`. The result should reflect a successful connection to the SAP HANA database. | SAP Basis administration |
## Related resources

### SAP references

SAP documentation references are frequently updated by SAP. To stay up to date, see SAP Note 2407186 – How-To Guides & Whitepapers For SAP HANA High Availability.

**Additional SAP notes**

- 2550327 – How-To Rename an SAP HANA System
- 1999880 – FAQ: SAP HANA System Replication
- 2078425 – Troubleshooting note for SAP HANA platform lifecycle management tool hdb1cm
- 2592227 – FQDN suffix change in HANA systems
- 2048681 – Performing SAP HANA platform lifecycle management administration tasks on multiple-host systems without SSH or root credentials

### SAP documents

- Network Required for SAP HANA System Replication
- Network Configuration for SAP HANA System Replication
- Host Name Resolution for System Replication

### AWS references

- Migrating SAP HANA from Other Platforms to AWS

### Additional information

The changes performed by hdb1cm as part of the hostname rename activity are consolidated in the following verbose log.
Migrate SQL Server to AWS using distributed availability groups

*Created by Praveen Marthala (AWS)*

**Source:** SQL Server On-Premises  
**Target:** SQL Server on EC2  
**R Type:** Rehost

**Environment:** PoC or pilot  
**Technologies:** Databases; Migration  
**Workload:** Microsoft

**AWS services:** Amazon EC2

**Summary**

Microsoft SQL Server Always On availability groups provide a high availability (HA) and disaster recovery (DR) solution for SQL Server. An availability group consists of a primary replica that accepts read/write traffic, and up to eight secondary replicas that accept read traffic. An availability group is configured on a Windows Server Failover Cluster (WSFC) with two or more nodes.

Microsoft SQL Server Always On distributed availability groups provide a solution to configure two separate availability groups between two independent WFSCs. The availability groups that are part of the distributed availability group don't have to be in the same data center. One availability group can be on premises, and the other availability group can be on the Amazon Web Services (AWS) Cloud on Amazon Elastic Compute Cloud (Amazon EC2) instances in a different domain.

This pattern outlines steps for using a distributed availability group to migrate on-premises SQL Server databases that are part of an existing availability group to SQL Server with availability groups set up on...
Amazon EC2. By following this pattern, you can migrate the databases to the AWS Cloud with minimal downtime during cutover. The databases are highly available on AWS immediately after the cutover. You can also use this pattern to change the underlying operating system from on-premises to AWS while keeping the same version of SQL Server.

**Prerequisites and limitations**

**Prerequisites**
- An active AWS account
- AWS Direct Connect or AWS Site-to-Site VPN
- The same version of SQL Server installed on-premises and on the two nodes on AWS

**Product versions**
- SQL Server version 2016 and later
- SQL Server Enterprise Edition

**Architecture**

**Source technology stack**
- Microsoft SQL Server database with Always On availability groups on premises

**Target technology stack**
- Microsoft SQL Server database with Always On availability groups on Amazon EC2 on the AWS Cloud

**Migration architecture**

**Terminology**
- WSFC 1 – WSFC on premises
- WSFC 2 – WSFC on the AWS Cloud
• AG 1 – First availability group, which is in WSFC 1
• AG 2 – Second availability group, which is in WSFC 2
• SQL Server primary replica – Node in AG 1 that is considered the global primary for all writes
• SQL Server forwarder – Node in AG 2 that receives data asynchronously from the SQL Server primary replica
• SQL Server secondary replica – Nodes in AG 1 or AG 2 that receive data synchronously from the primary replica or the forwarder

Tools

• AWS Direct Connect – AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services, bypassing internet service providers in your network path.
• Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
• AWS Site-to-Site VPN – AWS Site-to-Site VPN supports creating a site-to-site virtual private network (VPN). You can configure the VPN to pass traffic between instances that you launch on AWS and your own remote network.
• Microsoft SQL Server Management Studio – Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

Epics

Set up a second availability group on AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a WSFC on AWS.</td>
<td>Create WSFC 2 on Amazon EC2 instances with two nodes for HA. You will use this failover cluster to create the second availability group (AG 2) on AWS.</td>
<td>Systems administrator, SysOps administrator</td>
</tr>
<tr>
<td>Create the second availability group on WSFC 2.</td>
<td>Using SSMS, create AG 2 on two nodes in WSFC 2. The first node in WSFC 2 will act as the forwarder. The second node in WSFC 2 will act as the secondary replica of AG 2. At this stage, no databases are available in AG 2. This is the starting point for setting up the distributed availability group.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Create databases with no recovery option on AG 2.</td>
<td>Back up databases on the on-premises availability group (AG 1). Restore the databases to both the forwarder and the secondary</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>
### Configure the distributed availability group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the distributed availability group on AG 1. | To create the distributed availability group on AG 1, use the `CREATE AVAILABILITY GROUP` with the `DISTRIBUTED` option.  
  1. Use `LISTENER_URL` endpoint addresses for AG 1 and AG 2.  
  2. For `AVAILABILITY-MODE`, use `ASYNCHRONOUS_COMMIT` to avoid network latency, if any. This will not impact the performance of the database.  
  3. For `FAILOVER_MODE`, use `MANUAL`. It is the only availability mode that works with distributed availability groups.  
  4. To restore the databases manually on AG 2 and have more control on larger databases, use `MANUAL` for `SEEDING_MODE`. | DBA, Developer     |
| Create the distributed availability group on AG 2. | To create the distributed availability group on AG 2, use `ALTER AVAILABILITY GROUP` with the `DISTRIBUTED` option.  
  1. Use `LISTENER_URL` endpoint addresses for AG 1 and AG 2.  
  2. For `AVAILABILITY-MODE`, use `ASYNCHRONOUS_COMMIT` to avoid network latency, if any. This will not impact the performance of the database. | DBA, Developer     |
**AWS Prescriptive Guidance Patterns**

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>**3. For <strong>FAILOVER_MODE, use <strong>MANUAL</strong>. It is the only availability mode that works with distributed availability groups.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>4. To restore the databases manually on AG 2 and have more control on larger databases, use <strong>MANUAL</strong> for <strong>SEEDING_MODE</strong>.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The distributed availability group is created between AG 1 and AG 2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The databases in AG 2 are not yet configured to take part in the data flow from AG 1 to AG 2.</td>
<td></td>
</tr>
<tr>
<td>Add databases to the forwarder and secondary replica on AG 2.</td>
<td><strong>Add the databases to the distributed availability group by using <strong>ALTER DATABASE</strong> with the <strong>SET HADR AVAILABILITY GROUP</strong> option in both the forwarder and the secondary replica on AG 2.</strong></td>
<td><strong>DBA, Developer</strong></td>
</tr>
<tr>
<td></td>
<td>This starts asynchronous data flow between databases on AG 1 and AG 2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The global primary takes writes, sends data synchronously to the secondary replica on AG 1, and sends data asynchronously to the forwarder on AG 2. The forwarder on AG 2 sends data synchronously to the secondary replica on AG 2.</td>
<td></td>
</tr>
</tbody>
</table>

**Monitor asynchronous data flow between AG 1 and AG 2**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use DMVs and SQL Server logs.</td>
<td><strong>Monitor the status of the data flow between two availability groups by using dynamic management views (DMVs) and SQL Server logs.</strong> <strong>DMVs that are of interest for monitoring include</strong> <code>sys.dm_hadr_availability_replica_states</code></td>
<td><strong>DBA, Developer</strong></td>
</tr>
</tbody>
</table>
Perform cutover activities for final migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop all traffic to the primary replica.</td>
<td>Stop incoming traffic to the primary replica in AG 1 so that no write activity occurs on the databases and the databases are ready for migration.</td>
<td>App owner, Developer</td>
</tr>
<tr>
<td>Change the availability mode of the distributed availability group on AG 1.</td>
<td>On the primary replica, set the availability mode of the distributed availability group to synchronous. After you change the availability mode to synchronous, the data are sent synchronously from the primary replica in AG 1 to the forwarder in AG 2.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Check the LSNs in both availability groups.</td>
<td>Check the last Log Sequence Numbers (LSNs) in both AG 1 and AG 2. Because no writes are happening in the primary replica in AG 1, the data are synchronized, and last LSNs for both availability groups should match.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Update AG 1 to the secondary role.</td>
<td>When you update AG 1 to the secondary role, AG 1 loses the primary replica role and doesn't accept writes, and the data flow between two availability groups stops.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

Fail over to the second availability group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manually fail over to AG 2.</td>
<td>On the forwarder in AG 2, alter the distributed availability group to allow data loss. Because you already checked and confirmed</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
|      | that the last LSNs on AG 1 and AG 2 match, data loss is not a concern. When you allow data loss on the forwarder in AG 2, the roles of AG 1 and AG 2 change:  
- AG 2 becomes the availability group with the primary replica and secondary replica.  
- AG 1 becomes the availability group with the forwarder and secondary replica. | DBA, Developer |
| Change the availability mode of the distributed availability group on AG 2. | On the primary replica in AG 2, change the availability mode to asynchronous. This changes the data movement from AG 2 to AG 1, from synchronous to asynchronous. This step is required to avoid network latency between AG 2 and AG 1, if any, and will not impact the performance of the database. | |
| Start sending traffic to the new primary replica. | Update the connection string to use the listener URL endpoint on AG 2 for sending traffic to the databases. AG 2 now accepts writes and sends data to the forwarder in AG 1, along with sending data to its own secondary replica in AG 2. Data moves asynchronously from AG 2 to AG 1. | App owner, Developer |

### Perform post-cutover activities

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop the distributed availability group on AG 2.</td>
<td>Monitor the migration for the planned amount of time. Then drop the distributed availability group on AG 2 to remove distributed availability group setup between AG 2 and AG 1. This removes the distributed availability group configuration, and data flow from AG 2 to AG 1 stops.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>
At this point, AG 2 is highly available on AWS, with a primary replica that takes writes and a secondary replica in the same availability group.

Decommission the on-premises servers.

Decommission the on-premises servers in WSFC 1 that are part of AG 1.

Skills required:
- Systems administrator, SysOps administrator

Related resources
- Distributed availability groups
- SQL Docs: Distributed availability groups
- SQL Docs: Always On availability groups: a high-availability and disaster-recovery solution

Monitor Amazon Aurora for instances without encryption

Created by Mansi Suratwala (AWS)

Environment: Production
Technologies: Security, identity, compliance; Storage & backup; Databases
Workload: Open-source; All other workloads

AWS services: Amazon SNS; Amazon Aurora; AWS CloudTrail; Amazon CloudWatch; AWS Lambda

Summary
This pattern provides an Amazon Web Services (AWS) CloudFormation template that you can deploy to set up automatic notifications when an Amazon Aurora instance is created without encryption turned on.

Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.

The CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event uses AWS CloudTrail to monitor for any Aurora instance creation or a point in time restoration of an existing instance. The Cloudwatch Events event initiates the Lambda function, which checks whether encryption is enabled. If encryption is not turned on, the Lambda function sends an Amazon Simple Notification Service (Amazon SNS) notification informing you of the violation.

Prerequisites and limitations

Prerequisites
• An active AWS account

Limitations

• This service control works with Amazon Aurora instances only. It does not support other Amazon Relational Database Service (Amazon RDS) instances.
• The CloudFormation template must be deployed for CreateDBInstance and RestoreDBClusterToPointInTime only.

Product versions

• PostgreSQL versions that are supported in Amazon Aurora
• MySQL versions that are supported in Amazon Aurora

Architecture

Target technology stack

• Amazon Aurora
• AWS CloudTrail
• Amazon CloudWatch
• AWS Lambda
• Amazon Simple Storage Service (Amazon S3)
• Amazon SNS

Target architecture

Automation and scale

You can use the CloudFormation template multiple times for different Regions and accounts. You need to run it only once in each Region or account.

Tools

Tools

• Amazon Aurora – Amazon Aurora is a fully managed relational database engine that’s compatible with MySQL and PostgreSQL.
- **AWS CloudTrail** – AWS CloudTrail helps you manage governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, a role, or an AWS service are recorded as events in CloudTrail.
- **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery using Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

**Code**

A .zip file of the project is available as an attachment.

**Epics**

**Create the S3 bucket for the Lambda script**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>Open the Amazon S3 console, and choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. Your S3 bucket needs to be in the same Region as Aurora. The S3 bucket name cannot contain leading slashes.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Upload the Lambda code to the S3 bucket**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code.</td>
<td>Upload the Lambda code .zip file provided in the <strong>Attachments</strong> section to the S3 bucket that you defined.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Deploy the CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>On the CloudFormation console, deploy the RDS_Aurora_Encryption_At_Rest.yml CloudFormation template that's</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or chose in the first epic.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, &lt;directory&gt;/&lt;filename&gt;.zip).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. Info designates detailed informational messages on the application's progress. Error designates error events that could still allow the application to continue running. Warning designates potentially harmful situations.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Complete the parameters in the CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email message to the email address provided. To receive notifications, you must confirm this email subscription.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Related resources**

- Creating an S3 bucket
- Uploading files to an S3 bucket
- Creating an Amazon Aurora DB cluster
Replicate data between Amazon RDS for MySQL and MySQL on Amazon EC2

Created by Rajesh Madiwale (AWS)

| R Type: N/A | Source: Databases: Relational | Target: Amazon EC2 MySQL |
| Created by: AWS | Environment: PoC or pilot | Technologies: Databases |
| Workload: Open-source |

Summary

This pattern describes how to set up data replication between an Amazon Relational Database Service (Amazon RDS) for MySQL DB instance and a MySQL database on an Amazon Elastic Compute Cloud (Amazon EC2) instance by using native global transaction identifier (GTID) replication.

Prerequisites and limitations

Prerequisites

- Make sure you have access to the Amazon RDS for MySQL DB instance with replication permissions.
- Exclude default Amazon RDS databases.
- Use similar memory and storage for source and target databases.

Restrictions

- This setup requires an internal team to run the read-only queries.
- The source and target MySQL versions should be the same.
- Replication is set up in the same AWS Region and virtual private cloud (VPC).

Product versions

- MySQL versions 5.5, 5.6, 5.7, 8.0. For the latest list of MySQL versions and features supported on Amazon RDS, see MySQL on Amazon RDS in the AWS documentation.

Architecture

Source technology stack
• Amazon RDS for MySQL DB instance

**Target technology stack**

• MySQL database on an EC2 instance

**Data migration architecture**

**Tools**

• Native MySQL GTID replication
• Native MySQL tools: `mysqldump` and `mysql`

**Epics**

**Replication setup**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install and configure the Amazon EC2 instance.</td>
<td></td>
<td>SYS ADMIN</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Install and configure MySQL binaries on the EC2 instance.</td>
<td>In the source database, use mysqldump to create an output file that contains database objects and data. In the target database, use mysql to restore the data.</td>
<td>DBA</td>
</tr>
<tr>
<td>Export the data from the Amazon RDS for MySQL DB instance, and then restore it on the EC2 instance.</td>
<td>For information about GTID-based replication, see the MySQL documentation link in the References and Help section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set up replication using the GTID.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Check the replication status.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Validate the data in the source and target databases.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Related resources**

- Amazon EC2 User Guide (AWS documentation)
- Installing MySQL on Linux (MySQL documentation)
- Replication with Global Transaction Identifiers (MySQL documentation)

---

**Schedule jobs for Amazon RDS and Aurora PostgreSQL using Lambda and Secrets Manager**

*Created by Yaser Raja (AWS)*

<table>
<thead>
<tr>
<th>R Type:</th>
<th>N/A</th>
<th>Source: Databases: Relational</th>
<th>Target: PostgreSQL on AWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
<td>AWS services: Amazon RDS; AWS Lambda; AWS Secrets Manager</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

For on-premises databases and databases that are hosted on Amazon Elastic Compute Cloud (Amazon EC2) instances, database administrators often use the `cron` utility to schedule jobs.

For example, a job for data extraction or a job for data purging can easily be scheduled using `cron`. For these jobs, database credentials are typically either hard-coded or stored in a properties file. However,
when you migrate to Amazon Relational Database Service (Amazon RDS) or Amazon Aurora PostgreSQL-Compatible Edition, you lose the ability to log in to the host instance to schedule cron jobs.

This pattern describes how to use AWS Lambda and AWS Secrets Manager to schedule jobs for Amazon RDS for PostgreSQL and Aurora PostgreSQL-Compatible databases after migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database

Limitations

- A job must complete within 15 minutes, which is the Lambda function timeout limit. For other limits, see the AWS Lambda documentation.
- Job code must be written in a language supported by Lambda.

Architecture

Source technology stack

This stack features jobs written in languages such as Bash, Python, and Java. Database credentials are stored in the properties file, and the job is scheduled using Linux cron.

Target technology stack

This stack has a Lambda function that uses the credentials stored in Secrets Manager to connect to the database and to perform the activity. The Lambda function is initiated at the scheduled interval by using Amazon CloudWatch Events.

Target architecture
Tools

- **AWS Lambda** – AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume; there is no charge when your code is not running. With AWS Lambda, you can run code for virtually any type of application or backend service with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and manages all the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging. All you need to do is provide your code in one of the languages that AWS Lambda supports.

- **CloudWatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams. CloudWatch Events becomes aware of operational changes as they occur. It responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information. You can also use CloudWatch Events to schedule automated actions that self-initiate at certain times using cron or rate expressions.

- **Secrets Manager** – AWS Secrets Manager helps you protect secrets for accessing your applications, services, and IT resources. You can easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets by calling Secrets Manager APIs, which eliminates the need to hard-code sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB. The service is extensible to other types of secrets, including API keys and OAuth tokens. Secrets Manager enables you to control access to secrets using fine-grained permissions and to audit secret rotation centrally for resources in the AWS Cloud, third-party services, and on-premises.
## Epics

### Store database credentials in Secrets Manager

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a database user for the Lambda function.</td>
<td>It is a good practice to use separate database users for different parts of your application. If a separate database user already exists for your cron jobs, use that; otherwise, create a new database user. For more information about users and permissions, see the link in the References and Help section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Store database credentials as a secret in Secrets Manager.</td>
<td>Follow the steps in the Secrets Manager tutorial. See the &quot;Related resources&quot; section for a link.</td>
<td>DBA, DevOps</td>
</tr>
</tbody>
</table>

### Author the code for the Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose a programming language supported by AWS Lambda.</td>
<td>See the &quot;Related resources&quot; section for a link to a list of supported languages.</td>
<td>Developer</td>
</tr>
<tr>
<td>Write the logic to fetch the database credentials from Secrets Manager.</td>
<td>For sample code, see the link in the &quot;Related resources&quot; section.</td>
<td>Developer</td>
</tr>
<tr>
<td>Write the logic to perform the scheduled database activity.</td>
<td>Migrate your existing code for the scheduling job that you're using on premises to the AWS Lambda function.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

### Deploy the code and create the Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Lambda function deployment package.</td>
<td>This package contains the code and its dependencies. For details, see the link in the &quot;Related resources&quot; section.</td>
<td>Developer</td>
</tr>
</tbody>
</table>
Create the Lambda function.
In the AWS Lambda console, choose "Create function," enter a function name, choose the runtime environment, and then choose "Create function."
DevOps

Upload the deployment package.
Choose the Lambda function you created to open its configuration. You can write your code directly in the code section or upload your deployment package. To upload your package, go to the "Function code" section, choose the "Code entry type" to upload a .zip file, and then select the package.
DevOps

Configure the Lambda function per your requirements.
For example, you can set the "Timeout" parameter to the duration you expect your Lambda function to take. For more information about Lambda configurations, see the link in the "Related resources" section.
DevOps

Set permissions for the Lambda function role to access Secrets Manager.
For instructions, see the link in the "Related resources" section.
DevOps

Test the Lambda function.
Initiate the function manually to make sure it works as expected.
DevOps

Schedule the Lambda function by using CloudWatch Events

Create a rule to run your Lambda function on a schedule.
Schedule the Lambda function by using CloudWatch Events. For instructions, see the link in the "Related resources" section.
DevOps

Related resources

Instructions for epics and tasks

- Creating users and setting permissions: Managing PostgreSQL users and roles (blog post)
- Storing database credentials: Create and Store Your Secret in AWS Secrets Manager
- Supported programming languages for Lambda: AWS Lambda Runtimes
- Fetching database credentials from Secrets Manager: How to securely provide database credentials to Lambda functions by using AWS Secrets Manager (blog post)
- Creating a Lambda deployment package: Deployment Package
- Configuring a Lambda function: Configuring AWS Lambda Functions
- Accessing Secrets Manager: Set permissions for the Lambda function role to access Secrets Manager
- Scheduling a Lambda function: Tutorial: Schedule AWS Lambda Functions Using CloudWatch Events

Additional resources
- AWS Secrets Manager
- Create a Lambda Function with the Console
- Creating a CloudWatch Events Rule That Triggers on an Event
- AWS Lambda Limits
- Query your AWS database from your serverless application (blog post)

Database migration patterns by workload

Topics
- IBM (p. 364)
- Microsoft (p. 364)
- N/A (p. 365)
- Open-source (p. 365)
- Oracle (p. 366)
- SAP (p. 367)

IBM
- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 836)
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 (p. 1182)
- Move mainframe files directly to Amazon S3 using Transfer Family (p. 779)

Microsoft
- Access on-premises Microsoft SQL Server tables from Microsoft SQL Server on Amazon EC2 using linked servers (p. 293)
- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition (p. 1291)
- Configure SAML SSO from Microsoft Azure AD to CloudEndure Migration (p. 752)
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python (p. 760)
- Determine whether your Microsoft SQL Server database can be downgraded from Enterprise to Standard edition (p. 311)
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account (p. 984)
- Install CloudEndure Agent through AWS Systems Manager on Windows and Linux (p. 768)
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS (p. 1155)
- Migrate a Microsoft Azure VM to Amazon EC2 using CloudEndure (p. 1000)
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS (p. 816)
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT (p. 1257)
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk (p. 1193)
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 (p. 1067)
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using CloudEndure (p. 1073)
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server (p. 1245)
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using Amazon S3 and SSMS (p. 1249)
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers (p. 1253)
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS (p. 940)
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents (p. 944)
- Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux (p. 1242)
- Migrate data from Microsoft Azure Blob storage and Backblaze B2 Cloud Storage to Amazon S3 by using Rclone (p. 1159)
- Migrate SQL Server to AWS using distributed availability groups (p. 346)
- Migrate Windows SSL certificates to an Application Load Balancer using ACM (p. 1150)

N/A

- Create an approval process for firewall requests during a rehost migration to AWS (p. 981)
- Encrypt an existing Amazon RDS for PostgreSQL DB instance (p. 313)
- Implement cross-Region disaster recovery with AWS DMS and Amazon Aurora (p. 326)

Open-source

- Migrate an on-premises Linux server to an Amazon EC2 Linux instance using AWS SMS (p. 991)
- Migrate an on-premises MariaDB database to Amazon EC2 using MySQL tools (p. 1081)
- Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools (p. 1264)
- Migrate an on-premises MySQL database to Amazon EC2 (p. 1086)
- Migrate an on-premises MySQL database to Amazon RDS for MySQL (p. 774)
- Migrate an on-premises MySQL database to Aurora MySQL (p. 953)
- Migrate an on-premises PostgreSQL database to Amazon EC2 (p. 1048)
- Migrate an on-premises PostgreSQL database to Amazon RDS for PostgreSQL (p. 1231)
- Migrate an on-premises PostgreSQL database to Aurora PostgreSQL (p. 1237)
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling (p. 1188)
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk (p. 1026)
- Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical (p. 1228)
- Migrate on-premises Java applications to AWS using AWS App2Container (p. 1278)
- Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3 (p. 1268)
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible Edition (p. 806)
- Monitor Amazon Aurora for instances without encryption (p. 353)
- Replicate data between Amazon RDS for MySQL and MySQL on Amazon EC2 (p. 357)
Restart the CloudEndure Agent automatically without disabling SELinux after rebooting a RHEL source server (p. 787)
Schedule jobs for Amazon RDS and Aurora PostgreSQL using Lambda and Secrets Manager (p. 359)
Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport (p. 1124)

Oracle

Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT (p. 792)
Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation (p. 797)
Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication (p. 1100)
Migrate an Amazon RDS for Oracle database to another VPC (p. 1111)
Migrate an on-premises Oracle database to Amazon EC2 using CloudEndure (p. 1038)
Migrate an on-premises Oracle database to Amazon EC2 using Oracle Data Pump (p. 1045)
Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash (p. 1214)
Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 876)
Migrate an on-premises Oracle database to Amazon RDS for Oracle (p. 1219)
Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump (p. 1224)
Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL using an Oracle bystander and AWS DMS (p. 880)
Migrate an on-premises Oracle database to Amazon Redshift using AWS SCT data extraction agents (p. 899)
Migrate an on-premises Oracle database to Oracle on Amazon EC2 (p. 1031)
Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT (p. 870)
Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS (p. 1210)
Migrate an Oracle database to Amazon Aurora MySQL using AWS DMS and AWS SCT (p. 902)
Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT (p. 891)
Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 908)
Migrate data from an on-premises Oracle database to Aurora PostgreSQL (p. 915)
Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL (p. 830)
Migrate from an on-premises Oracle database or Amazon RDS for Oracle to Amazon DynamoDB using AWS DMS and AWS SCT (p. 865)
Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS (p. 847)
Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS (p. 840)
Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 854)
Migrate from Oracle to Amazon DocumentDB using AWS DMS (p. 860)
Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate (p. 888)
Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS (p. 1204)
Migrate Oracle native functions to PostgreSQL using extensions (p. 811)
Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible (p. 958)
Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL (p. 969)
AWS Prescriptive Guidance Patterns

SAP

- Migrate an on-premises SAP ASE database to Amazon EC2 (p. 1054)
- Migrate an SAP ASE database to Aurora MySQL using AWS DMS (p. 926)
- Migrate an SAP ASE database to Aurora PostgreSQL using AWS DMS (p. 931)
- Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS (p. 922)
- Migrate from SAP ASE to PostgreSQL on Amazon EC2 using AWS DMS (p. 936)
- Migrate SAP HANA to AWS using SAP HSR with the same hostname (p. 336)
- Migrate SAP workloads to an SAP database on Amazon EC2 using CloudEndure (p. 1059)

More patterns

- Allow EC2 instances write access to S3 buckets in AMS accounts (p. 1768)
- Authenticate Microsoft SQL Server on Amazon EC2 using AWS Directory Service (p. 1579)
- Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch (p. 148)
- Automate data loading from Amazon S3 to Amazon Redshift using AWS Data Pipeline (p. 9)
- Automatically archive items to Amazon S3 using DynamoDB TTL (p. 1334)
- Automatically remediate unencrypted Amazon RDS DB instances and clusters (p. 1608)
- Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows (p. 691)
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9 (p. 403)
- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition (p. 1291)
- Configure cross-account access to Amazon DynamoDB (p. 416)
- Convert and unpack data from EBCDIC to ASCII (p. 1398)
- Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL (p. 19)
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL (p. 25)
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python (p. 760)
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Kinesis Data Firehose with AWS CDK (p. 1747)
- Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing (p. 587)
- Enable transparent data encryption in Amazon RDS for SQL Server (p. 1660)
- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT (p. 792)
- Manage credentials using AWS Secrets Manager (p. 1687)
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS (p. 816)
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT (p. 1257)
- Migrate a self-hosted MongoDB environment to MongoDB Atlas on the AWS Cloud (p. 1198)
- Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents (p. 947)
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation (p. 797)
- Migrate an Amazon RDS DB instance to another VPC or account (p. 1108)
- Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication (p. 1100)
• Migrate an Amazon RDS for Oracle DB instance to another VPC (p. 1111)
• Migrate an Amazon Redshift cluster to an AWS Region in China (p. 1114)
• Migrate an on-premises MariaDB database to Amazon EC2 using MySQL tools (p. 1081)
• Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools (p. 1264)
• Migrate an on-premises Microsoft SQL Server database to Amazon EC2 (p. 1067)
• Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using CloudEndure (p. 1073)
• Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server (p. 1245)
• Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using Amazon S3 and SSMS (p. 1249)
• Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers (p. 1253)
• Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS (p. 940)
• Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents (p. 944)
• Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux (p. 1242)
• Migrate an on-premises MySQL database to Amazon EC2 (p. 1086)
• Migrate an on-premises MySQL database to Amazon RDS for MySQL (p. 774)
• Migrate an on-premises MySQL database to Aurora MySQL (p. 953)
• Migrate an on-premises Oracle database to Amazon EC2 using CloudEndure (p. 1038)
• Migrate an on-premises Oracle database to Amazon EC2 using Oracle Data Pump (p. 1045)
• Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash (p. 1214)
• Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 876)
• Migrate an on-premises Oracle database to Amazon RDS for Oracle (p. 1219)
• Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump (p. 1224)
• Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL using an Oracle bystander and AWS DMS (p. 880)
• Migrate an on-premises Oracle database to Amazon Redshift using AWS SCT data extraction agents (p. 899)
• Migrate an on-premises Oracle database to Oracle on Amazon EC2 (p. 1031)
• Migrate an on-premises PostgreSQL database to Amazon EC2 (p. 1048)
• Migrate an on-premises PostgreSQL database to Amazon RDS for PostgreSQL (p. 1231)
• Migrate an on-premises PostgreSQL database to Aurora PostgreSQL (p. 1237)
• Migrate an on-premises SAP ASE database to Amazon EC2 (p. 1054)
• Migrate an on-premises ThoughtSpot Falcon database to Amazon Redshift (p. 822)
• Migrate an on-premises Vertica database to Amazon Redshift using AWS SCT data extraction agents (p. 950)
• Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT (p. 870)
• Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS (p. 1210)
• Migrate an Oracle database to Amazon Aurora MySQL using AWS DMS and AWS SCT (p. 902)
• Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT (p. 891)
• Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 908)
• Migrate an SAP ASE database to Aurora MySQL using AWS DMS (p. 926)
• Migrate an SAP ASE database to Aurora PostgreSQL using AWS DMS (p. 931)
• Migrate data from an on-premises Oracle database to Aurora PostgreSQL (p. 915)
• Migrate data to the AWS Cloud by using Starburst (p. 53)
• Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL (p. 830)
• Migrate from an on-premises Oracle database or Amazon RDS for Oracle to Amazon DynamoDB using AWS DMS and AWS SCT (p. 865)
• Migrate from Couchbase Server to Couchbase Cloud on AWS (p. 1164)
• Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 836)
• Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS (p. 847)
• Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS (p. 840)
• Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 854)
• Migrate from Oracle to Amazon DocumentDB using AWS DMS (p. 860)
• Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate (p. 888)
• Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical (p. 1228)
• Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS (p. 922)
• Migrate from SAP ASE to PostgreSQL on Amazon EC2 using AWS DMS (p. 936)
• Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3 (p. 1268)
• Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible Edition (p. 806)
• Migrate Oracle native functions to PostgreSQL using extensions (p. 811)
• Migrate SAP workloads to an SAP database on Amazon EC2 using CloudEndure (p. 1059)
• Monitor Amazon ElastiCache clusters for at-rest encryption (p. 1696)
• Monitor ElastiCache clusters for security groups (p. 1566)
• Perform cross-Region failover and failback by using CloudEndure Disaster Recovery (p. 1826)
• Rotate credentials without restarting containers (p. 246)
• Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible (p. 958)
• Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport (p. 1124)
• Use CloudEndure for disaster recovery of an on-premises database (p. 1834)
• Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL (p. 969)
• Verify that new Amazon Redshift clusters launch in a VPC (p. 1570)
DevOps

Topics

• Deploy an AWS Glue job with an AWS CodePipeline CI/CD pipeline (p. 371)
• Automate event-driven backups from CodeCommit to Amazon S3 using CodeBuild and CloudWatch Events (p. 374)
• Automate static website deployment to Amazon S3 (p. 379)
• Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK (p. 384)
• Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK (p. 395)
• Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9 (p. 403)
• Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm (p. 409)
• Check AWS CDK applications or CloudFormation templates for best practices by using cdk-nag rule packs (p. 414)
• Configure cross-account access to Amazon DynamoDB (p. 416)
• Configure mutual TLS authentication for applications running on Amazon EKS (p. 428)
• Create a CI/CD pipeline to deploy microservices with AWS Fargate and Amazon API Gateway (p. 434)
• Create a custom log parser for Amazon ECS using a Firelens log router (p. 441)
• Create a pipeline and AMI using CodePipeline and HashiCorp Packer (p. 447)
• Create a pipeline and deploy artifact updates to on-premises EC2 instances using CodePipeline (p. 452)
• Deploy a CI/CD pipeline for Java microservices on Amazon ECS (p. 457)
• Use AWS CodeCommit and AWS CodePipeline to deploy a CI/CD pipeline in multiple AWS accounts (p. 464)
• Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile (p. 468)
• Deploy code in multiple AWS Regions using AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild (p. 475)
• Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere (p. 483)
• Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit (p. 487)
• Integrate a Bitbucket repository with AWS Amplify using AWS CloudFormation (p. 495)
• Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function (p. 500)
• Monitor Amazon ECR repositories for wildcard permissions using AWS CloudFormation and AWS Config (p. 507)
• Perform a canary-based deployment using the blue/green strategy and AWS Lambda (p. 511)
• Perform custom actions from AWS CodeCommit events (p. 517)
• Publish Amazon CloudWatch metrics to a CSV file (p. 520)
Deploy an AWS Glue job with an AWS CodePipeline CI/CD pipeline

Created by Bruno Klein (AWS) and Luis Henrique Massao Yamada (AWS)

**Environment:** Production  
**Technologies:** DevOps; Big data  
**AWS services:** AWS Glue; AWS CodeCommit; AWS CodePipeline; AWS Lambda

### Summary

This pattern demonstrates how you can integrate Amazon Web Services (AWS) CodeCommit and AWS CodePipeline with AWS Glue, and use AWS Lambda to launch jobs as soon as a developer pushes their changes to a remote AWS CodeCommit repository.

When a developer submits a change to an extract, transform, and load (ETL) repository and pushes the changes to AWS CodeCommit, a new pipeline is invoked. The pipeline initiates a Lambda function that launches an AWS Glue job with these changes. The AWS Glue job performs the ETL task.

This solution is helpful in the situation where businesses, developers, and data engineers want to launch jobs as soon as changes are committed and pushed to the target repositories. It helps achieve a higher level of automation and reproducibility, therefore avoiding errors during the job launch and lifecycle.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account  
- Git installed on the local machine  
- Amazon Cloud Development Kit (Amazon CDK) installed on the local machine  
- Python installed on the local machine  
- The code in the *Attachments* section

**Limitations**

- The pipeline is finished as soon as the AWS Glue job is successfully launched. It does not wait for the conclusion of the job.
• The code provided in the attachment is intended for demo purposes only.

Architecture

Target technology stack

• AWS Glue
• AWS Lambda
• AWS CodePipeline
• AWS CodeCommit

Target architecture

The process consists of these steps:

1. The developer or data engineer makes a modification in the ETL code, commits, and pushes the change to AWS CodeCommit.
2. The push initiates the pipeline.
3. The pipeline initiates a Lambda function, which calls codecommit:GetFile on the repository and uploads the file to Amazon Simple Storage Service (Amazon S3).
4. The Lambda function launches a new AWS Glue job with the ETL code.
5. The Lambda function finishes the pipeline.

**Automation and scale**

The sample attachment demonstrates how you can integrate AWS Glue with AWS CodePipeline. It provides a baseline example that you can customize or extend for your own use. For details, see the Epics section.

**Tools**

- **AWS CodePipeline** – AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates.
- **AWS CodeCommit** – AWS CodeCommit is a fully managed source control service that hosts secure, Git-based repositories.
- **AWS Lambda** – AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers.
- **AWS Glue** – AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development.
- **Git client** – Git provides GUI tools, or you can use the command line or a desktop tool to check out the required artifacts from GitHub.
- **AWS CDK** – The AWS CDK is an open source software development framework that helps you define your cloud application resources by using familiar programming languages.

**Epics**

**Deploy the sample code**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the AWS CLI.</td>
<td>Configure the AWS Command Line Interface (AWS CLI) to target and authenticate with your current AWS account. For instructions, see the AWS CLI documentation.</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td>Extract the sample project files.</td>
<td>Extract the files from the attachment to create a folder that contains the sample project files.</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td>Deploy the sample code.</td>
<td>After you extract the files, run the following commands from the extract location to create a baseline example: cdk bootstrap, cdk deploy, git init</td>
<td>Developer, DevOps engineer</td>
</tr>
</tbody>
</table>
Automate event-driven backups from CodeCommit to Amazon S3 using CodeBuild and CloudWatch Events

Created by Kirankumar Chandrashekar (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>DevOps; Storage &amp; backup</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon S3; Amazon CloudWatch; AWS CodeBuild; AWS CodeCommit</td>
</tr>
<tr>
<td>Workload:</td>
<td>All other workloads</td>
</tr>
</tbody>
</table>
Summary

On the Amazon Web Services (AWS) Cloud, you can use AWS CodeCommit to host secure Git-based repositories. CodeCommit is a fully managed source control service. However, if a CodeCommit repository is accidentally deleted, its contents are also deleted and cannot be restored.

This pattern describes how to automatically back up a CodeCommit repository to an Amazon Simple Storage Service (Amazon S3) bucket after a change is made to the repository. If the CodeCommit repository is later deleted, this backup strategy provides you with a point-in-time recovery option.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing CodeCommit repository, with user access configured according to your requirements. For more information, see Setting up for AWS CodeCommit in the CodeCommit documentation.
- An S3 bucket for uploading the CodeCommit backups.

Limitations

- This pattern automatically backs up all of your CodeCommit repositories. If you want to back up individual CodeCommit repositories, you must modify the Amazon CloudWatch Events rule.

Architecture

The following diagram illustrates the workflow for this pattern.

The workflow consists of the following steps:

1. Code is pushed to a CodeCommit repository.
2. The CodeCommit repository notifies CloudWatch Events of a repository change (for example, a git push command).
3. CloudWatch Events invokes AWS CodeBuild and sends it the CodeCommit repository information.
4. CodeBuild clones the entire CodeCommit repository and packages it into a .zip file.
5. CodeBuild uploads the .zip file to an S3 bucket.

Technology stack

- CloudWatch Events
- CodeBuild
- CodeCommit
- Amazon S3

Tools

- Amazon CloudWatch Events – CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS CodeBuild – CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.
- AWS CodeCommit – CodeCommit is a fully managed source control service that hosts secure Git-based repositories.
- AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

Epics

Create a CodeBuild project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CodeBuild service role.</td>
<td>Sign in to the AWS Management Console and open the IAM console. Choose Roles, and choose Create role. Create a service role for CodeBuild to clone the CodeCommit repository, upload files to the S3 bucket, and send logs to Amazon CloudWatch. For more information, see Create a CodeBuild service role in the CodeBuild documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create a CodeBuild project.</td>
<td>On the CodeBuild console, choose Create CodeBuild project. Create a CodeBuild project by using the buildspec.yml template.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Create and configure the CloudWatch Events rule</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Create an IAM role for CloudWatch Events.</strong></td>
<td>On the IAM console, choose Roles and create an IAM role for CloudWatch Events. For more information about this, see CloudWatch Events IAM role in the IAM documentation. <strong>Important:</strong> You must add codebuild:StartBuild permissions to the IAM role for CloudWatch Events.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
| **Create a CloudWatch Events rule.** | 1. On the CloudWatch console, choose Events and then choose Rules. Choose Create rule, and use the CloudWatch Events rule from the Additional information section. This creates a rule that listens for event changes (for example, git push or git commit commands) in your CodeCommit repositories. For more information, see Create a CloudWatch Events rule for a CodeCommit source in the AWS CodePipeline documentation.  
2. Choose Targets, choose Topic, and then choose Configure input. Choose Input transformer, and use the input path and input template from the Additional information section. This ensures that your CodeCommit repository details are parsed and sent as environment variables to the CodeBuild project. For more information, see the input transformer tutorial in the CloudWatch documentation. | Cloud administrator |
### Task

3. Choose **Configure details**, and enter a name and description for the rule. Choose **Create rule**.

**Important:** This CloudWatch Events rule describes changes in all your CodeCommit repositories. You must modify the CloudWatch Events rule if you want to back up individual CodeCommit repositories or use separate S3 buckets for different repository backups.

### Related resources

**Creating a CodeBuild project**

- Create a CodeBuild service role
- Create a CodeBuild project
- Required permissions for Git client commands

**Creating and configuring a CloudWatch Events rule**

- Create a CloudWatch Events rule for a CodeCommit source
- Use input transformer to customize what is passed to the event target
- Create a CloudWatch Events rule that initiates on an event
- Create a CloudWatch Events IAM role

### Additional information

**CodeBuild buildspec.yml template**

```yaml
version: 0.2
phases:
  install:
    commands:
      - pip install git-remote-codecommit
  build:
    commands:
      - env
      - git clone -b $REFERENCE_NAME codecommit::REPO_REGION://$REPOSITORY_NAME
      - dt=$(date '+%d-%m-%Y-%H:%M:%S');
      - echo "$dt"
      - zip -yr $dt-$REPOSITORY_NAME-backup.zip ./
      - aws s3 cp $dt-$REPOSITORY_NAME-backup.zip s3:// #substitute a valid S3 Bucket Name here

CloudWatch Events rule
```
Automate static website deployment to Amazon S3

Sample input transformer for the CloudWatch Events rule target

Input path:

{"referenceType":"$.detail.referenceType","region":"$.region","repositoryName":"$.detail.repositoryName"}

Input template:

{"environmentVariablesOverride": [
    {
        "name": "REFERENCE_NAME",
        "value": <referenceName>
    },
    {
        "name": "REFERENCE_TYPE",
        "value": <referenceType>
    },
    {
        "name": "REPOSITORY_NAME",
        "value": <repositoryName>
    },
    {
        "name": "REPO_REGION",
        "value": <region>
    },
    {
        "name": "ACCOUNT_ID",
        "value": <account>
    }
]}

Automate static website deployment to Amazon S3

Created by Sankar Sangubotla (AWS)

| R Type: | N/A | Source: Application Development | Target: S3 |
Summary

This pattern describes the steps required to add a continuous integration and continuous delivery (CI/CD) pipeline to an existing bucket in Amazon Simple Storage Service (Amazon S3) on the Amazon Web Services (AWS) Cloud. This pattern uses GitHub as a source provider. The pipeline is initiated when new items are committed, and the changes are then reflected in the S3 bucket.

Prerequisites and limitations

**Prerequisites**

- An active AWS account
- Knowledge of Amazon S3 and AWS CodePipeline
- A static website, including output/source files such as HTML 4/5, CSS 2/4, images, fonts, and icons
- A GitHub repository

**Limitations**

- This process is recommended for displaying read-only content. It isn't recommended for collecting or transferring sensitive information, because Amazon S3 uses the HTTP protocol.
- Websites built using PHP, JSP, or APS.NET are not supported, because Amazon S3 doesn't support server-side scripts.

Architecture

**Source architecture**

[Diagram showing GitHub, Developer, Web server, and Website]

**Target technology stack**

- AWS CodePipeline
- AWS CodeStar
- Amazon S3
- Any web server
Tools

AWS CodePipeline – A continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.

AWS CodeStar – AWS CodeStar is a cloud-based service for creating, managing, and working with software development projects on AWS.

Amazon S3 – A highly scalable object storage service. It can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

Epics

Create an Amazon S3 bucket and upload content

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket.</td>
<td>Open the Amazon S3 console and choose Create bucket to create an S3 bucket for hosting the website and uploading content.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Enter the bucket name.</td>
<td>Enter a unique DNS-compliant name for your bucket. An S3</td>
<td>AWS General, Developer</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select a Region.</strong></td>
<td>Choose either the default AWS Region or select a specific Region for where your bucket will be based.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td><strong>Set permissions.</strong></td>
<td>Allow public access to your bucket by choosing Permissions and then choosing Edit. Choose Off for Block all public access. By default, this check box is selected for security purposes. Choose Save and review the information before choosing Create bucket. This closes the pop-up and creates the bucket.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td><strong>Upload folders and files.</strong></td>
<td>Upload your folders and files by following the instructions in the Amazon S3 documentation. For links, see the Related resources section.</td>
<td>AWS General, Developer</td>
</tr>
</tbody>
</table>

**Configure the S3 bucket for website hosting**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable static website hosting.</strong></td>
<td>In the Properties tab for the S3 bucket, choose Static website hosting, and then Use this bucket to host a website.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td><strong>Define home and error pages.</strong></td>
<td>Specify file names and extensions for the home page and error page (for example, index.html and error.html). Make sure that the root folder contains these files and that they serve as landing pages.</td>
<td>AWS General, Developer</td>
</tr>
</tbody>
</table>

**Create a bucket policy**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attach bucket policy.</strong></td>
<td>Create a bucket policy so that other AWS applications</td>
<td>AWS General, Developer</td>
</tr>
</tbody>
</table>
### Access and test the website.

Enter the endpoint URL to access the website; for example: http://www.spsbuddy.xyz.s3-website-us-east-1.amazonaws.com. The endpoint follows two formats, depending on your Region. For more information, see the Related resources section.

**Skills required:** AWS General, Developer

### Create a pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a CodePipeline name.</td>
<td>Sign in to the AWS Management Console and open the AWS CodePipeline console. Choose Create pipeline and enter a name for the project.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Create a service role.</td>
<td>Create a service role or select from the list of service roles if provided.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Select an artifact store.</td>
<td>Choose a custom location and specify an existing bucket if you want Amazon S3 to create a bucket and store the artifacts in it. If not, use the default location.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Choose a source provider.</td>
<td>Choose GitHub (Version 2) from the list of source providers.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Connect to GitHub.</td>
<td>Authorize GitHub to grant CodePipeline access to your GitHub repository by following the instructions from Step 2: Create a connection to GitHub in GitHub connections from the AWS CodePipeline documentation.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Create repository and branch.</td>
<td>Choose the repository and branch to create as part of the commit step.</td>
<td>AWS General, Developer</td>
</tr>
</tbody>
</table>
### Related resources

**References**

- Hosting a static website on Amazon S3
- Amazon S3 website endpoints
- How to create an S3 bucket
- How to upload files and folders to an S3 bucket
- How to configure an S3 bucket for static website hosting

### Additional information

To grant public read access for your website, copy the following bucket policy, and paste it in the bucket policy editor:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::example.com/*"]
    }
  ]
}
```

Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK
Summary

You can integrate Amazon Elastic Compute Cloud (Amazon EC2) instances with AWS Systems Manager to automate operational tasks and provide more visibility and control. To integrate with Systems Manager, EC2 instances must have an installed AWS Systems Manager Agent (SSM Agent) and an AmazonSSMManagedInstanceCore AWS Identity and Access Management (IAM) policy attached to their instance profiles.

However, if you want to ensure that all EC2 instance profiles have the AmazonSSMManagedInstanceCore policy attached, you can face challenges updating new EC2 instances that don’t have instance profiles or EC2 instances that have an instance profile but don’t have the AmazonSSMManagedInstanceCore policy. It can also be difficult to add this policy across multiple Amazon Web Services (AWS) accounts and AWS Regions.

This pattern helps solve these challenges by deploying three Cloud Custodian policies in your AWS accounts:

- The first Cloud Custodian policy checks for existing EC2 instances that have an instance profile but don’t have the AmazonSSMManagedInstanceCore policy. The AmazonSSMManagedInstanceCore policy is then attached.
- The second Cloud Custodian policy checks for existing EC2 instances without an instance profile and adds a default instance profile that has the AmazonSSMManagedInstanceCore policy attached.
- The third Cloud Custodian policy creates AWS Lambda functions in your accounts to monitor the creation of EC2 instances and instance profiles. This ensures that the AmazonSSMManagedInstanceCore policy is automatically attached when an EC2 instance is created.

This pattern uses AWS DevOps tools to achieve a continuous, at-scale deployment of the Cloud Custodian policies to a multi-account environment, without provisioning a separate compute environment.

Prerequisites and limitations

Prerequisites

- Two or more active AWS accounts. One account is the security account and the others are member accounts.
- An existing IAM user or role in the security account with permissions to provision AWS resources. This pattern uses administrator permissions, but you should grant permissions according to your organization’s requirements and policies.
- Ability to assume an IAM role from the security account to member accounts and create the required IAM roles. For more information about this, see Delegate access across AWS accounts using IAM roles in the IAM documentation.

- AWS Command Line Interface (AWS CLI), installed and configured. For testing purposes, you can configure AWS CLI by using the `aws configure` command or setting environment variables. **Important:** This isn't recommended for production environments and we recommend that this account is only granted least privilege access. For more information about this, see Grant least privilege in the IAM documentation.

- The `devops-cdk-cloudcustodian.zip` file (attached), downloaded to your local computer.
- Familiarity with Python.
- The required tools (Node.js, AWS Cloud Development Kit (AWS CDK), and Git), installed and configured. You can use the `install-prerequisites.sh` file in the `devops-cdk-cloudcustodian.zip` file to install these tools. Make sure you run this file with root privileges.

**Limitations**

- Although this pattern can be used in a production environment, make sure that all IAM roles and policies meet your organization’s requirements and policies.

**Package versions**

- Cloud Custodian version 0.9 or later
- TypeScript version 3.9.7 or later
- Node.js version 14.15.4 or later
- npm version 7.6.1 or later
- AWS CDK version 1.96.0 or later
The diagram shows the following workflow:

1. Cloud Custodian policies are pushed to an AWS CodeCommit repository in the security account. An Amazon CloudWatch Events rule automatically initiates the AWS CodePipeline pipeline.

2. The pipeline fetches the most recent code from CodeCommit and sends it to the continuous integration part of the continuous integration and continuous delivery (CI/CD) pipeline handled by AWS CodeBuild.

3. CodeBuild performs the complete DevSecOps actions, including policy syntax validation on the Cloud Custodian policies, and runs these policies in --dryrun mode to check which resources are identified.

4. If there are no errors, the next task alerts an administrator to review the changes and approve the deployment into the member accounts.

**Technology stack**

- AWS CDK
- CodeBuild
- CodeCommit
- CodePipeline
- IAM
- Cloud Custodian

**Automation and scale**
The AWS CDK pipelines module provisions a CI/CD pipeline that uses CodePipeline to orchestrate the building and testing of source code with CodeBuild, in addition to the deployment of AWS resources with AWS CloudFormation stacks. You can use this pattern for all member accounts and Regions in your organization. You can also extend the Roles creation stack to deploy other IAM roles in your member accounts.

**Tools**

- **AWS CDK** – AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
- **AWS CodeBuild** – CodeBuild is a fully managed build service in the cloud.
- **AWS CodeCommit** – CodeCommit is a version control service that you can use to privately store and manage assets.
- **AWS CodePipeline** – CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.
- **AWS Identity and Access Management** – IAM is a web service that helps you securely control access to AWS resources.
- **Cloud Custodian** – Cloud Custodian is a tool that unifies the dozens of tools and scripts most organizations use for managing their public cloud accounts into one open-source tool.
- **Node.js** – Node.js is a JavaScript runtime built on Google Chrome's V8 JavaScript engine.

**Code**

For a detailed list of modules, account functions, files, and deployment commands used in this pattern, see the README file in the devops-cdk-cloudcustodian.zip file (attached).

**Epics**

Set up the pipeline with AWS CDK

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the CodeCommit</td>
<td>1. Unzip the devops-cdk-cloudcustodian.zip file (attached) in the working</td>
<td>Developer</td>
</tr>
<tr>
<td>repository.</td>
<td>directory on your local computer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Sign in to the AWS Management Console for your security account, open the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CodeCommit console, and then create a new devops-cdk-cloudcustodian</td>
<td></td>
</tr>
<tr>
<td></td>
<td>repository.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Change into the project directory and set up the CodeCommit repository</td>
<td></td>
</tr>
<tr>
<td></td>
<td>as the origin, commit the changes, and then push them to the origin branch</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>by running the following commands:</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>• cd devops-cdk-cloudcustodian</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• git init --initial-branch=main</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• git add . git commit -m 'initial commit'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• git remote add origin <a href="https://git-codecommit.us-east-1.amazonaws.com/v1/devops-cdk-cloudcustodian">https://git-codecommit.us-east-1.amazonaws.com/v1/devops-cdk-cloudcustodian</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• git push origin main</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about this, see <a href="https://aws.amazon.com">Creating a CodeCommit repository</a> in the AWS CodeCommit documentation.</td>
<td></td>
</tr>
<tr>
<td>Install the required tools.</td>
<td>Use the <code>install-prerequisites.sh</code> file to install all the required tools on Amazon Linux. This doesn’t include AWS CLI because it comes pre-installed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about this, see the <a href="https://aws.amazon.com">Prerequisites</a> section of <a href="https://aws.amazon.com">Getting started with the AWS CDK</a> in the AWS CDK documentation.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Install the required AWS CDK packages. | 1. Set up your virtual environment by running the following command in AWS CLI: `$ python3 -m venv .env`  
2. Activate your virtual environment by running the following command: `$ source .env/bin/activate`  
3. After the virtual environment is activated, install the required dependencies by running the following command: `$ pip install -r requirements.txt`  
4. To add additional dependencies (for example, other AWS CDK libraries), add them to the requirements.txt file, and then run the following command: `pip install -r requirements.txt` | Developer |

The following packages are required by AWS CDK and are included in the requirements.txt file:

- `aws-cdk.aws-cloudwatch`
- `aws-cdk.aws-codebuild`
- `aws-cdk.aws-codecommit`
- `aws-cdk.aws-codedeploy`
- `aws-cdk.aws-codepipeline`
- `aws-cdk.aws-codepipeline-actions`
- `aws-cdk.aws-events`
- `aws-cdk.aws-events-targets`
- `aws-cdk.aws-iam`
- `aws-cdk.aws-logs`
- `aws-cdk.aws-s3`
- `aws-cdk.aws-sns`
- `aws-cdk.aws-sns-subscriptions`
- `aws-cdk.aws-sqs`
- `aws-cdk.core`
Configure your environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the required variables.</td>
<td>Open the <code>vars.py</code> file in the root folder of your CodeCommit repository and update the following variables:</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>• Update <code>var_deploy_region = 'us-east-1'</code> with the AWS Region where you want the pipeline to be deployed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update <code>var_codecommit_repo_name = &quot;cdk-cloudcustodian&quot;</code> with the name of your CodeCommit repository.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update <code>var_codecommit_branch_name = &quot;main&quot;</code> with name of the CodeCommit branch.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update `var_adminEmail=<a href="mailto:notifyadmin@email.com">notifyadmin@email.com</a>' with the email address for the administrator that approves changes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update <code>var_slackWebHookUrl = https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX</code> with the Slack webhook used to send Cloud Custodian notifications when changes are made.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update <code>var_orgId = 'o-yyyyMMdd'</code> with your organization ID.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update <code>security_account = '123456789011'</code> with the AWS account ID for the account where the pipeline is deployed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Update <code>member_accounts = ['1111111111', '11111111112', '11111111113']</code> with the member accounts where you want to bootstrap</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Update the account.yml file with the member account information.</td>
<td>To run the <code>c7n-org Cloud Custodian</code> tool against multiple accounts, you must place the <code>accounts.yml</code> config file in the root of the repository. The following is a sample Cloud Custodian config file for AWS:</td>
<td>Developer</td>
</tr>
</tbody>
</table>

```yaml
accounts:
- account_id: '123123123123'
  name: account-1
  regions:
  - us-east-1
  - us-west-2
  role:
    arn:aws:iam::123123123123:role/CloudCustodian
  vars:
    charge_code: xyz
  tags:
    - type:prod
    - division:some division
    - partition:us
    - scope:pci
```

- Set `cdk_bootstrap_member_accounts = True` to `True` if you want the pipeline to automatically bootstrap the AWS CDK to your member accounts. If set to `True` this also requires the name of an existing IAM role in the member accounts that can be assumed from the security account. This IAM role must also have the required permissions to bootstrap the AWS CDK.

- Update `cdk_bootstrap_role = ‘AWSControlTowerExecution’` with the existing IAM role in the member accounts that can be assumed from the security account. This role must also permission to bootstrap the AWS CDK.

  **Note:** This only applies if `cdk_bootstrap_member_accounts` is set to `True`.

- The AWS CDK stack and deploy the required IAM roles.
## Bootstrap the AWS accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Bootstrap the security account. | Bootstrap the deploy_account with the cloudcustodian_stack application by running the following command:  

```bash
cdk bootstrap -a 'python3 cloudcustodian/cloudcustodian_stack.py'  
```
<table>
<thead>
<tr>
<th>Developer</th>
</tr>
</thead>
</table>

| Option 1 - Automatically bootstrap the member accounts. | If the cdk_bootstrap_member_accounts variable is set to True in the vars.py file, the accounts specified in the member_accounts variable are automatically bootstrapped by the pipeline.  

If required, you can update *cdk_bootstrap_role* with an IAM role that you can assume from the security account and that has the required permissions to bootstrap the AWS CDK.  

New accounts added to the member_accounts variable are automatically bootstrapped by the pipeline so that the required roles can be deployed. | Developer |
|--------|-------------------------------------------------|----------------|

| Option 2 - Manually bootstrap the member accounts. | Although we don't recommend using this approach, you can set the value of cdk_bootstrap_member_accounts to False and perform this step manually by running the following command:  

```bash
$ cdk bootstrap -a 'python3 cloudcustodian/member_account_roles_stack.py' \  
--trust \  
{security_account_id} \  
```
<table>
<thead>
<tr>
<th>Developer</th>
</tr>
</thead>
</table>

---

393
### Deploy the AWS CDK stacks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create the IAM roles in the member accounts.</strong></td>
<td>Run the following command to deploy the member_account_roles_stack stack and create the IAM roles in the member accounts:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cdk deploy --all -a 'python3 cloudcustodian/member_account_roles_stack.py'</td>
<td>Developer</td>
</tr>
<tr>
<td><strong>Deploy the Cloud Custodian pipeline stack.</strong></td>
<td>Run the following command to create the Cloud Custodian stack.</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline that is deployed into the security account:</td>
<td>cdk deploy -a 'python3 cloudcustodian/cloudcustodian_stack.py'</td>
</tr>
</tbody>
</table>

### Related resources

- Getting started with the AWS CDK

### Attachments

attachment.zip

### Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK

*Created by Varsha Raju (AWS)*

| Environment: PoC or pilot | Technologies: DevOps; Containers & microservices; Modernization; Infrastructure | AWS services: AWS CodeBuild; AWS CodeCommit; AWS CodePipeline; Amazon ECS; AWS CDK |

### Summary

This pattern describes how to automatically create the continuous integration and continuous delivery (CI/CD) pipelines and underlying infrastructure for building and deploying microservices on Amazon Elastic Container Service (Amazon ECS). You can use this approach if you want to set up proof-of-concept CI/CD pipelines to show your organization the benefits of CI/CD, microservices, and DevOps. You can also use this approach to create initial CI/CD pipelines that you can then customize or change according to your organization’s requirements.

The pattern’s approach creates a production environment and non-production environment that each have a virtual private cloud (VPC) and an Amazon ECS cluster configured to run in two Availability Zones. These environments are shared by all your microservices and you then create a CI/CD pipeline for each microservice. These CI/CD pipelines pull changes from a source repository in AWS CodeCommit, automatically build the changes, and then deploy them into your production and non-production environments. When a pipeline successfully completes all of its stages, you can use URLs to access the microservice in the production and non-production environments.

### Prerequisites and limitations

**Prerequisites**
• An active Amazon Web Services (AWS) account.
• An existing Amazon Simple Storage Service (Amazon S3) bucket that contains the starter-code.zip file (attached).
• AWS Cloud Development Kit (AWS CDK), installed and configured in your account. For more information about this, see Getting started with the AWS CDK in the AWS CDK documentation.
• Python 3 and pip, installed and configured. For more information about this, see the Python documentation.
• Familiarity with AWS CDK, AWS CodePipeline, AWS CodeBuild, CodeCommit, Amazon Elastic Container Registry (Amazon ECR), Amazon ECS, and AWS Fargate.
• Familiarity with Docker.
• An understanding of CI/CD and DevOps.

Limitations
• General AWS account limits apply. For more information about this, see AWS service quotas in the AWS General Reference documentation.

Product versions
• The code was tested using Node.js version 16.13.0 and AWS CDK version 1.132.0.

Architecture

The diagram shows the following workflow:
1. An application developer commits code to a CodeCommit repository.
2. A pipeline is initiated.
3. CodeBuild builds and pushes the Docker image to an Amazon ECR repository.
4. CodePipeline deploys a new image to an existing Fargate service in a non-production Amazon ECS cluster.
5. Amazon ECS pulls the image from the Amazon ECR repository into a non-production Fargate service.
6. Testing is performed using a non-production URL.
7. The release manager approves the production deployment.
8. CodePipeline deploys the new image to an existing Fargate service in a production Amazon ECS cluster.
9. Amazon ECS pulls the image from the Amazon ECR repository into the production Fargate service.
10. Production users access your feature by using a production URL.

**Technology stack**

- AWS CDK
- CodeBuild
- CodeCommit
- CodePipeline
- Amazon ECR
- Amazon ECS
- Amazon VPC

**Automation and scale**

You can use this pattern's approach to create pipelines for microservices deployed in a shared AWS CloudFormation stack. The automation can create more than one Amazon ECS cluster in each VPC and also create pipelines for microservices deployed in a shared Amazon ECS cluster. However, this requires that you provide new resource information as inputs to the pipeline stack.

**Tools**

- **AWS CDK** – AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeCommit** – AWS CodeCommit is a version control service that enables you to privately store and manage Git repositories in the AWS Cloud. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure.
- **AWS CodePipeline** – AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.
- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that is used for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.
- **Docker** – Docker helps developers to pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

**Code**

The code for this pattern is available in the cicdstarter.zip and starter-code.zip files (attached).
Epics

Set up your environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Set up the working directory for AWS CDK. | 1. Create a directory named cicdproject on your local machine.  
2. Download the cicdstarter.zip file (attached) into the cicdproject directory and unzip it. This creates a folder named cicdstarter.  
3. Run the cd <user-home>/cicdproject/cicdstarter command.  
4. Set up the Python virtual environment by running the python3 -m venv .venv command.  
5. Run the source ./.venv/bin/activate command.  
6. Configure your AWS environment by running the aws configure command or by using the following environment variables:  
   • AWS_ACCESS_KEY_ID  
   • AWS_SECRET_ACCESS_KEY  
   • AWS_DEFAULT_REGION | AWS DevOps, Cloud infrastructure |

Create the shared infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the shared infrastructure. | 1. In your working directory, run the cd cicdvpcecs command.  
2. Run the pip3 install -r requirements.txt command to install all required Python dependencies  
3. Run the cdk bootstrap command to set the AWS environment for the AWS CDK.  
4. Run the cdk synth --context | AWS DevOps, Cloud infrastructure |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Sign in to the AWS Management Console, open the AWS CloudFormation console, and then choose the cicd-vpc-ecs stack from the list.</td>
<td>AWS DevOps, Cloud infrastructure</td>
</tr>
<tr>
<td>2.</td>
<td>In the stack details pane, choose the <strong>Events</strong> tab and monitor the progress of your stack creation.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Run the <code>cdk deploy --context aws_account=&lt;aws_account_ID&gt; --context aws_region=&lt;aws_region&gt;</code> command.</td>
<td></td>
</tr>
</tbody>
</table>
| 6. | The AWS CloudFormation stack creates the following infrastructure:  
  - A non-production VPC named `cicd-vpc-ecs/cicd-vpc-nonprod`  
  - A production VPC named `cicd-vpc-ecs/cicd-vpc-prod`  
  - A non-production Amazon ECS cluster named `cicd-ecs-nonprod`  
  - A production Amazon ECS cluster named `cicd-ecs-prod` | |

Monitor the AWS CloudFormation stack.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Test the AWS CloudFormation stack. | 1. After the cicd-vpc-ecs AWS CloudFormation stack is created, ensure that the cicd-vpc-ecs/cicd-vpc-nonprod and cicd-vpc-ecs/cicd-vpc-prod VPCs are created.  
2. Ensure that the cicd-ecs-nonprod and cicd-ecs-prod Amazon ECS clusters are created. | AWS DevOps, Cloud infrastructure |

**Important:** Make sure that you record the IDs for the two VPCs and the security group IDs for the default security groups in both VPCs.

---

Create a CI/CD pipeline for a microservice

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the infrastructure for the microservice. | 1. Name your microservice. For example, this pattern uses myservice1 as the microservice's name.  
2. In your working directory run the cd <working-directory>/cdkpipeline command.  
3. Run the pip3 install -r requirements.txt command.  
4. Run the full cdk synth command that is available in the Additional information section of this pattern.  
5. Run the full cdk deploy command that is available in the Additional information section of this pattern. | AWS DevOps, Cloud infrastructure |

**Note:** You can also provide the values for both commands by using the cdk.json file in the directory.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor the AWS CloudFormation stack.</td>
<td>Open the AWS CloudFormation console and monitor the progress of the myservice1-cicd-stack stack. Eventually,</td>
<td>AWS DevOps, Cloud infrastructure</td>
</tr>
</tbody>
</table>
### Task: Test the AWS CloudFormation stack.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>the status changes to CREATE_COMPLETE.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the AWS CodeCommit console, verify that a repository named myservice1 exists and contains the starter code.</td>
</tr>
<tr>
<td>2. On the AWS CodeBuild console, verify that a build project named myservice1 exists.</td>
</tr>
<tr>
<td>3. On the Amazon ECR console, verify that an Amazon ECR repository named myservice1 exists.</td>
</tr>
<tr>
<td>4. On the Amazon ECS console, verify that a Fargate service named myservice1 exists in both a non-production and production Amazon ECS cluster.</td>
</tr>
<tr>
<td>5. On the Amazon Elastic Compute Cloud (Amazon EC2) console, verify that the non-production and production Application Load Balancers are created. Record the DNS names of the ALBs.</td>
</tr>
<tr>
<td>6. On the AWS CodePipeline console, verify that a pipeline named myservice1 exists. It must have Source, Build, Deploy-NonProd, and Deploy-Prod stages. The pipeline should also have an in progress status.</td>
</tr>
<tr>
<td>7. Monitor the pipeline until all stages are complete.</td>
</tr>
<tr>
<td>8. Manually approve it for production.</td>
</tr>
<tr>
<td>9. In a browser window, enter the DNS names of the ALBs.</td>
</tr>
<tr>
<td>10. The application should display Hello World in the non-production and production URLs.</td>
</tr>
</tbody>
</table>
Use the pipeline.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the pipeline.</td>
<td>1. Open the CodeCommit repository that you created earlier and open the index.js file. 2. Replace Hello World with Hello CI/CD. 3. Save and commit the changes to the main branch. 4. Verify that the pipeline initiates and that the change goes through the Build, Deploy-NonProd, and Deploy-Prod stages. 5. Manually approve the production. 6. Both production and non-production URLs should now display Hello CICD.</td>
<td>AWS DevOps, Cloud infrastructure</td>
</tr>
</tbody>
</table>

Repeat this epic for each microservice.

Repeat the tasks in this epic to create a CI/CD pipeline for each of your microservices.

Related resources

- Using Python with AWS CDK
- AWS CDK Python reference
- Creating an AWS Fargate service using the AWS CDK

Additional information

cdk synth command

cdk synth --context aws_account=<aws_account_number> --context aws_region=<aws_region> --context vpc_nonprod_id=<id_of_non_production_VPC> --context ecssg_nonprod_id=<default_security_group_id_of_non-production_VPC> --context code_commit_s3_bucket_for_code=<S3 bucket name> --context code_commit_s3_object_key_for_code=<Object_key_of_starter_code> --context microservice_name=<name_of_microservice>

cdk deploy command

cdk deploy --context aws_account=<aws_account_number> --context aws_region=<aws_region> --context vpc_nonprod_id=<id_of_non_production_VPC> --context vpc_prod_id=<id_of_production_VPC> --context ecssg_nonprod_id=<default_security_group_id_of_non-production_VPC> --context ecssg_prod_id=<default_security_group_id_of_production_VPC> --context code_commit_s3_bucket_for_code=<S3 bucket name> --context
Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9

Created by Alexandre Nardi (AWS)

Environment: PoC or pilot
Technologies: DevOps; Serverless; Websites & web apps; Databases
AWS services: AWS Cloud9; AWS CloudFormation; AWS CodePipeline; Amazon DynamoDB; AWS CodeCommit

Summary

This pattern demonstrates how to develop a typical web application in a serverless architecture, for developers and development leads who are beginning to test DevOps practices on Amazon Web Services (AWS). It builds a sample application that creates a storefront and backend for browsing and purchasing books, and provides a microservice that can be developed independently. The pattern uses AWS Cloud9 as a development environment, an Amazon DynamoDB database as a data store, and AWS services such as AWS CodePipeline and AWS CodeBuild for continuous integration and continuous deployment (CI/CD) functionality.

The pattern guides you through the following development activities:

• Creating a standard AWS Cloud9 development environment
• Using AWS CloudFormation templates to create a web application and a microservice for books
• Using AWS Cloud9 to modify the front-end, commit changes, and test changes
• Creating and testing a CI/CD pipeline to the microservice
• Automating unit tests

The code for this pattern is provided in GitHub, in the AWS DevOps End-to-End Workshop repository.

Prerequisites and limitations

Prerequisites

• An active AWS account
• Files from the AWS DevOps End-to-End Workshop downloaded to your computer
Important: Building this demo application in your AWS account creates and consumes AWS resources. You are responsible for the cost of the AWS services and resources used to create and run the application. After you finish your work, be sure to remove all resources to avoid ongoing charges. For cleanup instructions, see the Epics section.

Limitations

This walkthrough is intended for demonstration and development purposes only. To use it in a production environment, see Security best practices in the AWS Identity and Access Management (IAM) documentation, and make the necessary changes to IAM roles, Amazon DynamoDB, and other services used. The web application is derived from the AWS Bookstore Demo App; for additional considerations, see the Known limitations section of the README file.

Architecture

The architecture of the bookstore application is illustrated in the Architecture section of the README file for the AWS Bookstore Demo App.

From a deployment perspective, the Bookstore Demo App uses a single CloudFormation template to deploy all services and objects in one stack. This pattern makes a few changes to demonstrate how a particular developer or team could work in a specific product (Books), and update it independently from the rest of the application. For this reason, the code for this pattern separates the AWS Lambda functions and related objects for the Books microservice into a second CloudFormation template, which creates a Books stack. That makes it possible to see the microservice being updated by using CI/CD practices. In the following diagram, the dashed border identifies the Books microservice.
Tools

Tools

- Jest framework for JavaScript testing
- Python 3.8

Code

The source code and templates for this pattern are available on GitHub, in the AWS DevOps End-to-End Workshop repository. Before following the steps in the Epics section, download all the files from the repository to your computer.

Note: The Epics section provides the high-level steps for this walkthrough, to give you general information about the process. To complete each step, see the README file in the AWS DevOps End-to-End Workshop repository for detailed instructions.

The AWS DevOps End-to-End Workshop repository extends the AWS Bookstore Demo App repository and uses a modified version of the AWS Cloud9 Bootstrapping code to create the AWS Cloud9 IDE.

Epics

Build the Bookstore web application and the Books microservice

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the front-end and Lambda functions for the Bookstore app. | 1. Log in to the CloudFormation console, and deploy the DemoBookstoreMainTemplate.yml template to create the DemoBookStoreStack stack. This creates the front-end and Lambda functions that are outside the Books microservice.  
2. In the Outputs tab of the stack, note the website URL under the WebApplication label. | Developer |
| Create the Books microservice. | On the CloudFormation console, deploy the DemoBookstoreBooksServiceTemplate.yml template to create the DemoBooksServiceStack stack. | Developer |
| Test your application. | Use the website URL from the DemoBookStoreStack stack to access the Bookstore application. | Developer |
Use the Cloud9 environment to maintain your application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS Cloud9 IDE.</td>
<td>On the CloudFormation console, deploy the C9EnvironmentTemplate.yml template to create an AWS Cloud9 environment.</td>
<td>Developer, Developer lead</td>
</tr>
</tbody>
</table>
| Create CodeCommit repositories.  | 1. Log in to the AWS CodeCommit console, and verify that you have a demobookstore-WebAssets repository, which contains the code for the front-end application.  
   2. Create a repository for the Books microservice called demobookstore-BooksService.  
   3. Clone the two repositories in AWS Cloud9 (demobookstore-WebAssets and demobookstore-BooksService) by using the git clone command. | Developer                        |
| Change the code in the front-end and check the pipeline. | 1. Using AWS Cloud9, make some code changes in a webpage. This will update the demobookstore-WebAssets repository.  
   2. On the AWS CodePipeline console, verify that demobookstore-Assets-Pipeline is running.  
   3. Test your web application by refreshing it from the browser (Ctrl+F5 on Firefox). | Developer                        |

Implement a CI/CD pipeline for the Books microservice

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add the build and service update YAML files. | 1. In AWS Cloud9, upload the buildspec.yml and DemoBookstoreBooksServiceUpdateTemplate.yml files.  
   • buildspec.yml has building instructions, and also includes testing instructions for automated tests. They are commented | Developer                        |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an S3 bucket for the build pipeline. | To create an S3 bucket, follow the instructions in the Amazon S3 documentation.  
- The bucket name must be globally unique; for example, demobookstore-books-service-pipeline-bucket-<YYYYMMDDHHMM>.  
- Clear the **Block all public access** check box, and select the **I acknowledge...** check box. | Developer |
| Use IAM to create a role for CloudFormation deployment. | Create a demobookstore-CloudFormation-role role and attach the AdministratorAccess policy. In the next epic, you can reconfigure this role for minimum permissions. | Developer |
| Create a new pipeline to automate building and deploying the Books microservice. | Create a pipeline (for example, demobookstore-BooksService-Pipeline) with Commit, Build, and Deploy stages, as described in the README file. | Developer |
| Test your microservice in AWS Cloud9. | Make a change in the ListBooks function and see the pipeline working. | Developer |
| Automate the unit test for the ListBooks Lambda function. | In the AWS Cloud9 IDE, enable the build to run unit tests, and check the test results. For instructions, see the README file. | Developer |

**(Optional) Implement additional functionality**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make your solution secure.</td>
<td>Configure demobookstore-CloudFormation-role to</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have minimum permissions, and check other used roles as well.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Eliminate dependencies in the CloudFormation templates.</td>
<td>The method for exchanging information between the DemoBookstoreMainTemplate.yml template and the DemoBookstoreBooksServiceTemplate.yml template is based on outputs and imports. Passing values between these two templates adds dependencies. To eliminate the dependencies, consider using AWS Systems Manager Parameter Store.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create a Cart microservice.</td>
<td>Use the Books microservice as an example for taking shopping cart-related functions out of the DemoBookstoreMainTemplate.yml template and creating a Cart microservice.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

#### Clean up

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete the S3 buckets.</td>
<td>On the Amazon S3 console, delete the buckets associated with the sample web application:</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>* Two buckets created for the AWS Bookstore Demo App. The buckets names start with the stack name you provided for AWS CloudFormation when you created the front-end; for example, DemoBookStoreStack.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* One bucket for the build pipeline; for example, demobookstore-books-service-pipeline-bucket-&lt;YYYYMMDDHHMM&gt;.</td>
<td></td>
</tr>
<tr>
<td>Delete the stacks.</td>
<td>On the CloudFormation console, delete the stacks associated with the sample web application:</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>* DemoBooksServiceStack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* DemoBookStoreStack</td>
<td></td>
</tr>
</tbody>
</table>
The removal could take more than 90 minutes. If the removal fails, delete them again, and also delete any manual resources (for example, the VPC or network interfaces) based on notifications.

Delete the IAM roles.

On the IAM console, delete the following roles:
- demobookstore-Cloudformation-role
- demobookstore-BooksService-BuildProject-service-role

For step-by-step instructions, see the IAM documentation.

Related resources

- AWS Bookstore Demo App
- AWS Cloud9 Bootstrapping Example
- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Creating a bucket (Amazon S3 documentation)

Additional information

For detailed, step-by-step instructions, see the README file in the AWS DevOps End-to-End Workshop GitHub repository.

Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm

Created by Abdullahi Olaoye (AWS)

| R Type: N/A | Source: On-premises DevOps processes | Target: CI/CD pipeline for iOS apps development on AWS |
| Created by: AWS | Environment: PoC or pilot | Technologies: Websites & web apps; DevOps |
| AWS services: AWS CodeCommit; AWS CodePipeline; AWS Device Farm |
Summary

This pattern outlines the steps for creating a continuous integration and continuous delivery (CI/CD) pipeline that uses AWS CodePipeline to build and test iOS applications on real devices on AWS. The pattern uses AWS CodeCommit to store the application code, the Jenkins open-source tool to build the iOS application, and AWS Device Farm to test the built application on real devices. These three phases are orchestrated together in a pipeline by using AWS CodePipeline.

This pattern is based on the post Building and testing iOS and iPadOS apps with AWS DevOps and mobile services on the AWS DevOps blog. For detailed instructions, see the blog post.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Apple developer account
- Build server (macOS)
- Xcode version 11.3 (installed and set up on the build server)
- AWS Command Line Interface (AWS CLI) installed and configured on the workstation
- Basic knowledge of Git

Limitations

- The application build server must be running macOS.
- The build server must have a public IP address, so CodePipeline can connect to it remotely to initiate builds.

Architecture

Source technology stack

- An on-premises iOS application build process that involves using a simulator or manual test on physical devices

Target technology stack

- An AWS CodeCommit repository for storing application source code
- A Jenkins server for application builds using Xcode
- An AWS Device Farm device pool for testing applications on real devices

Target architecture

When a user commits changes to the source repository, the pipeline (AWS CodePipeline) fetches the code from the source repository, initiates a Jenkins build, and passes the application code to Jenkins. After the build, the pipeline retrieves the build artifact and starts an AWS Device Farm job to test the application against a device pool.
AWS CodePipeline – AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define.

AWS CodeCommit – AWS CodeCommit is a fully managed source control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. CodeCommit eliminates the need to operate your own source control system or worry about scaling its infrastructure.

AWS Device Farm – AWS Device Farm is an application testing service that lets you improve the quality of your web and mobile apps by testing them across an extensive range of desktop browsers and real mobile devices, without having to provision and manage any testing infrastructure.

Jenkins – Jenkins is an open-source automation server that enables developers to build, test, and deploy their software.
Epics

Set up the build environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Jenkins on the build server that's running macOS.</td>
<td>Jenkins will be used for building the application, so you must first install it on the build server. To get detailed instructions for this and subsequent tasks, see the blog post and other resources in the &quot;Related resources&quot; section at the end of this pattern.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Configure Jenkins.</td>
<td>Follow the on-screen instructions to configure Jenkins.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Install the AWS CodePipeline plugin for Jenkins.</td>
<td>This plugin must be installed on the Jenkins server in order for Jenkins to interact with the AWS CodePipeline service.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Create a Jenkins freestyle project.</td>
<td>In Jenkins, create a freestyle project. Configure the project to specify triggers and other build configuration options.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

Configure AWS Device Farm

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Device Farm project.</td>
<td>Open the AWS Device Farm console. Create a project and a device pool for testing. For instructions, see the blog post.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

Configure the source repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CodeCommit repository.</td>
<td>Create a repository where the source code will be stored.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Commit your application code to the repository.</td>
<td>Connect to the CodeCommit repository you created. Push the code from your local machine to the repository.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
Configure the pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a pipeline in AWS CodePipeline.</td>
<td>Open the AWS CodePipeline console, and create a pipeline. The pipeline orchestrates all the phases of the CI/CD process. For instructions, see the blog post.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Add a test stage to the pipeline.</td>
<td>Edit the pipeline to add a test stage and to integrate it with AWS Device Farm.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Initiate the pipeline.</td>
<td>Choose &quot;Release change&quot; to start the pipeline and the CI/CD process.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

View application test results

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review test results.</td>
<td>In the AWS Device Farm console, select the project you created, and review the results of the tests. The console will show the details of each test.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

Related resources

Step-by-step instructions for this pattern

- Building and testing iOS and iPadOS apps with AWS DevOps and mobile services (AWS DevOps blog post)

Configure AWS Device Farm

- AWS Device Farm console

Configure the source repository

- Create an AWS CodeCommit repository
- Connect to an AWS CodeCommit repository

Configure the pipeline

- AWS CodePipeline console

Additional resources

- AWS CodePipeline documentation
Check AWS CDK applications or CloudFormation templates for best practices by using cdk-nag rule packs

Created by Arun Donti

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>DevOps; Security, identity, compliance</th>
<th>Workload:</th>
<th>Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS CDK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern explains how you can use the cdk-nag utility to check AWS Cloud Development Kit (AWS CDK) applications for best practices by using a combination of rule packs. cdk-nag is an open-source project that was inspired by cfn_nag. It implements rules in evaluation packs such as AWS Solutions Library, Health Insurance Portability and Accountability Act (HIPAA), and National Institute of Standards and Technology (NIST) 800-53 by using AWS CDK Aspects. You can check your AWS CDK applications for best practices by using the rules in these packs, detect and remediate code based on best practices, and suppress the rules that you don’t want to use in your evaluations.

You can also use cdk-nag to check your AWS CloudFormation templates by using the cloudformation-include module.

For information about all available packs, see the Rules section of the cdk-nag repository. Evaluation packs are available for:

- AWS Solutions Library
- HIPAA security
- NIST 800-53 rev 4
- NIST 800-53 rev 5
- Payment Card Industry Data Security Standard (PCI DSS) 3.2.1

Prerequisites and limitations

Prerequisites

- An application that uses the AWS CDK
**Tools**

- **AWS CDK** – Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and you can launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

**Epics**

**Integrate cdk-nag with your AWS CDK application**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn about cdk-nag.</td>
<td>Navigate to the cdk-nag GitHub repository and read through the documentation.</td>
<td>App developer</td>
</tr>
<tr>
<td>Install the cdk-nag package in your AWS CDK application.</td>
<td>To use cdk-nag in your AWS CDK application, you must install it first. cdk-nag is available to download from PyPI, npm, NuGet, and Apache Maven. For the latest information about available versions and download locations, see the Readme file in the repository.</td>
<td>App developer</td>
</tr>
<tr>
<td>Choose your NagPacks.</td>
<td>cdk-nag has different packs of rules called NagPacks. Each NagPack contains rules that conform to a specific standard. For example, the AWS Solutions NagPack contains general best practices, and the NIST 800-53 rev 5 NagPack can help with compliance. You can apply multiple NagPacks to your application, and you can add and remove packs as necessary. For a list of available packs, see the Readme file in the GitHub repository. For information about the individual rules in each pack, see the Rules section of the GitHub repository.</td>
<td>App developer</td>
</tr>
<tr>
<td>Integrate cdk-nag into your AWS CDK application.</td>
<td>You can integrate cdk-nag into your application on an applicationwide level, or integrate it into individual stages or stacks</td>
<td>App developer</td>
</tr>
</tbody>
</table>
in your application. For example, to integrate the AWS Solutions and HIPAA security NagPacks into an AWS CDK v2 TypeScript application on an applicationwide level, you can use the following code:

```javascript
import { App, Aspects } from 'aws-cdk-lib';
import { CdkTestStack } from '../lib/cdk-test-stack';
import { AwsSolutionsChecks, HIPAASecurityChecks } from 'cdk-nag';

const app = new App();
new CdkTestStack(app, 'CdkNagDemo');
// Simple rule informational messages
Aspects.of(app).add(new AwsSolutionsChecks());
// Additional explanations on the purpose of triggered rules
Aspects.of(app).add(new HIPAASecurityChecks({ verbose: true }));
```

### Related resources

- cdk-nag code repository
- cdk-nag in Construct Hub

### Configure cross-account access to Amazon DynamoDB

*Created by Shashi Dalmia (AWS)*

**Environment:** Production  
**Technologies:** DevOps; Databases; Security, identity, compliance  
**AWS services:** Amazon DynamoDB; AWS Identity and Access Management; AWS Lambda

### Summary

This pattern explains the steps for configuring cross-account access to Amazon DynamoDB. Amazon Web Services (AWS) services can access DynamoDB tables that are in the same AWS account if the service
has the appropriate AWS Identity and Access Management (IAM) permissions set up in the database. However, access from a different AWS account requires setting up IAM permissions and establishing a trust relationship between the two accounts.

This pattern provides steps and sample code to demonstrate how you can configure AWS Lambda functions in one account to read and write to a DynamoDB table in a different account.

**Prerequisites and limitations**

- Two active AWS accounts. This pattern refers to these accounts as **Account A** and **Account B**.
- AWS Command Line Interface (AWS CLI) installed and configured to access Account A, to create the DynamoDB database. The other steps in this pattern provide instructions for using the IAM, DynamoDB, and Lambda consoles. If you’re planning to use AWS CLI instead, configure it to access both accounts.

**Architecture**

In the following diagram, AWS Lambda, Amazon EC2, and DynamoDB are all in the same account. In this scenario, Lambda functions and Amazon Elastic Compute Cloud (Amazon EC2) instances can access DynamoDB.

If resources in a different AWS account try to access DynamoDB, they require setting up cross-account access and a trust relationship. For example, in the following diagram, to enable access between DynamoDB in Account A and the Lambda function in Account B, you must create a trust relationship between the accounts and grant appropriate access to the Lambda service and users, as described in the Epics (p. 419) section.
Tools

AWS services

- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
Code

This pattern includes sample code to illustrate how you can configure a Lambda function in Account B to read and write to the DynamoDB table in Account A. The code is provided for illustration and testing purposes only. If you’re implementing this pattern in a production environment, use the code as a reference and customize it for your own environment.

This pattern illustrates cross-account access with Lambda and DynamoDB. You can use the same steps for other AWS services as well, but make sure that you grant and configure the appropriate permissions in both accounts. For example, if you want to grant access to an Amazon Relational Database Service (Amazon RDS) database in Account A, create a role for that database and bind it with a trust relationship. In Account B, if you want to use Amazon EC2 instead of AWS Lambda, create the respective IAM policy and role, and then attach them to the EC2 instance.

Epics

Create a DynamoDB table in Account A

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a DynamoDB table in Account A.</td>
<td>After you configure AWS CLI for Account A, use the following AWS CLI command to create a DynamoDB table:</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>aws dynamodb create-table \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--table-name Table-Account-A \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--attribute-definitions \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeName=category,AttributeType=S \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeName=item,AttributeType=S \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--key-schema \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeName=category,KeyType=HASH \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeName=item,KeyType=RANGE \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--provisioned-throughput \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ReadCapacityUnits=5,WriteCapacityUnits=5</td>
<td></td>
</tr>
</tbody>
</table>

For more information about creating tables, see the DynamoDB documentation.
# Create a role in Account A

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a role in Account A.</td>
<td>This role will be used by Account B to gain permissions to access Account A. To create the role:</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>2. Open the IAM console at <a href="https://console.aws.amazon.com/iam/">https://console.aws.amazon.com/iam/</a>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. In the navigation pane of the console, choose Roles, and then choose Create role.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. For Select type of trusted entity, choose Another AWS account.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. For Account ID, type the ID for Account B.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Choose Next: Permissions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7. In the Filter policies box, type DynamoDB.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8. In the list of DynamoDB policies, select AmazonDynamoDBFullAccess. This policy allows all actions on DynamoDB. For a list of other policies you can choose instead, see Example policies in the IAM documentation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9. Choose Next: Tags.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10(Optional) Add metadata to the role by attaching tags as key-value pairs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11. Choose Next: Review. The Review page displays the account number for Account A, the AmazonDynamoDBFullAccess policy, and any tags you entered.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12. For Role name, type a unique name for your role (for example, DynamoDB-FullAccess-For-Account-B), and add an optional role description.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13. Choose Create role.</td>
<td></td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information about creating roles, see the IAM documentation.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
| Note the ARN for the role in Account A. | 1. In the navigation pane of the IAM console, choose Roles.  
2. In the search box, type `DynamoDB-FullAccess-For-Account-B` (or the role name you assigned in the previous story), and choose the role.  
3. In the summary page for the role, copy the Amazon Resource Name (ARN). You'll use the ARN when setting up Account B and in the Lambda code. | AWS DevOps |

Configure access to Account A from Account B

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
2. Open the IAM console at https://console.aws.amazon.com/iam/.  
3. In the navigation pane of the console, choose Policies, and then choose Create Policy.  
4. Choose the JSON tab.  
5. Type or paste the following JSON document:  

```json
{
    "Version": "2012-10-17", 
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
        }
    ]
}
``` | AWS DevOps |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>where the <code>Resource</code> property contains the ARN of the role you created in the previous story in Account A.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Choose <strong>Next: Tags</strong>.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>(Optional) Add metadata to the policy by attaching tags as key-value pairs.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Choose <strong>Next: Review</strong>.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>For <strong>Policy name</strong>, type a unique name for your policy (for example, <code>DynamoDB-FullAccess-Policy-in-Account-A</code>), and add an optional policy description.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Choose <strong>Create policy</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

For more information about creating policies, see the [IAM documentation](https://docs.aws.amazon.com/IAM/latest/UserGuide/index.html).
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a role based on the policy. | This role is used by the Lambda functions in Account B to read from and write to the DynamoDB table in Account A.  
1. In Account B, in the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.  
2. For **Select type of trusted entity**, choose **AWS service**.  
3. For use case, choose **Lambda**.  
4. Choose **Next: Permissions**.  
5. In the **Filter policies** box, type **DynamoDB**.  
6. In the list of DynamoDB policies, select **DynamoDB-FullAccess-Policy-in-Account-A**, which you created in the previous story.  
7. Choose **Next: Tags**.  
8. (Optional) Add metadata to the role by attaching tags as key-value pairs.  
9. Choose **Next: Review**.  
   The **Review** page displays **AWS service: lambda.amazonaws.com** as the trusted entity, **DynamoDB-FullAccess-Policy-in-Account-A** as policy, and any tags you entered.  
10. For **Role name**, type a unique name for your role (for example, **DynamoDB-FullAccess-in-Account-A**), and add an optional role description.  
11. Choose **Create role**.  
   You can now attach this role to the Lambda functions in the next epic.  
   For more information about creating roles, see the **IAM documentation**. | AWS DevOps |
## Create Lambda functions in Account B

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Create a Lambda function to write data to DynamoDB.** | 1. Sign in to Account B at https://account-ID-for-Account-B.signin.aws.amazon.com/console.  
3. In the navigation pane of the console, choose Functions, and then choose Create function.  
4. For Name, type lambda_write_function.  
5. For Runtime, choose Python 3.8 or later.  
6. For Permissions, Change default execution role, choose Use an existing role.  
7. For Existing role, choose DynamoDB-FullAccess-in-Account-A.  
8. Choose Create function.  
9. In the Code tab, paste the contents of the lambda_write_function.py file, which is provided in the attachment. Make sure to provide the correct ARN information (from the epic Create a role in Account A) for the RoleArn field, and change region_name according to your requirements.  
10. Run the function by choosing the Test button.  
11. Check the output from the function. It should be similar to the output shown in the Lambda write function section of Additional information (p. 426). This output indicates that the function accessed the DynamoDB table in Account A and was able to write data to it. | AWS DevOps |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Create a Lambda function to read data from DynamoDB.** | 1. In the navigation pane of the Lambda console, choose **Functions**, and then choose **Create function**.  
2. For **Name**, type **lambda_read_function**.  
3. For **Runtime**, choose **Python 3.8** or later.  
4. For **Permissions**, **Change default execution role**, choose **Use an existing role**.  
5. For **Existing role**, choose **DynamoDB-FullAccess-in-Account-A**.  
6. Choose **Create function**.  
7. In the **Code** tab, paste the contents of the **lambda_read_function.py** file, which is provided in the attachment. Make sure to provide the correct ARN information (from the epic **Create a role in Account A**) for the **RoleArn** field, and change **region_name** according to your requirements.  
8. Run the function by choosing the **Test** button.  
9. Check the output from the function. It should be similar to the output shown in the **Lambda read function** section of **Additional information** (p. 426).  
   This output indicates that the function accessed the DynamoDB table in Account A and was able to read the data you added to the table. | AWS DevOps

For more information about creating Lambda functions, see the [Lambda documentation](https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html).
## Clean up resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete the resources you created.</td>
<td>If you’re running this pattern in a testing or proof of concept (PoC) environment, delete the resources you created to avoid incurring costs.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>1. In Account B, delete the two Lambda functions and other resources you created to connect to DynamoDB.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. In Account A, delete the DynamoDB table you created.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. IAM policies do not cost anything, so you can keep them as is. However, for security, we recommend that you delete the following roles and policies you created for this pattern:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Account A: DynamoDB-Full-Access-for-Account-A role</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Account B: DynamoDB-FullAccess-in-Account-A role</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Account B: DynamoDB-FullAccess-Policy-in-Account-A policy</td>
<td></td>
</tr>
</tbody>
</table>

## Related resources

- Getting started with the AWS CLI (AWS CLI documentation)
- Configuring the AWS CLI (AWS CLI documentation)
- Getting started with DynamoDB (DynamoDB documentation)
- Getting started with Lambda (AWS Lambda documentation)
- Creating a role to delegate permissions to an IAM user (IAM documentation)
- Creating IAM policies (IAM documentation)
- Cross-account policy evaluation logic (IAM documentation)
- IAM JSON policy elements reference (IAM documentation)

## Additional information

**Lambda write function**

This code is provided in the `lambda_write_function.py` file in the attachment.
Lambda read function

This code is provided in the lambda_read_function.py file in the attachment.
Configure mutual TLS authentication for applications running on Amazon EKS

Created by Mahendra Siddappa (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>DevOps; Security, identity, compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon EKS; Amazon Route 53</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

Certificate-based mutual Transport Layer Security (TLS) is an optional TLS component that provides two-way peer authentication between servers and clients. With mutual TLS, clients must provide an X.509 certificate during the session negotiation process. The server uses this certificate to identify and authenticate the client.

Mutual TLS is a common requirement for Internet of Things (IoT) applications and can be used for business-to-business applications or standards such as Open Banking.

This pattern describes how to configure mutual TLS for applications running on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster by using an NGINX ingress controller. You can enable built-in mutual TLS features for the NGINX ingress controller by annotating the ingress resource. For more information about mutual TLS annotations on NGINX controllers, see Client certificate authentication in the Kubernetes documentation.

Important: This pattern uses self-signed certificates. We recommend that you use this pattern only with test clusters, and not in production environments. If you want to use this pattern in a production environment, you can use AWS Certificate Manager Private Certificate Authority or your existing public key infrastructure (PKI) standard to issue private certificates.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account.
- An existing Amazon EKS cluster.
- AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows.
- The `kubectl` command line utility, installed and configured to access the Amazon EKS cluster. For more information about this, see Installing kubectl in the Amazon EKS documentation.
- An existing Domain Name System (DNS) name to test the application.

Limitations

- This pattern uses self-signed certificates. We recommend that you use this pattern only with test clusters, and not in production environments.
Architecture

Technology stack

- Amazon EKS
- Amazon Route 53
- Kubectl

Tools

- **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) makes it easy for you to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- **Amazon Route 53** – Route 53 is a highly available and scalable DNS web service.
- **Kubectl** – `kubectl` is a command line utility that you use to interact with an Amazon EKS cluster.

Epics

Download the Kubernetes configuration files

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download and save the Kubernetes configuration.</td>
<td>Download the ingress.yaml and mtls.yaml files (attached) and save them to a local path on your computer.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
### Generate the self-signed certificates

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate the CA key and certificate.</td>
<td>Generate the certificate authority (CA) key and certificate by running the following command:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes -subj '/CN=Test Cert Authority'</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Generate the server key and certificate, and sign with the CA certificate.</td>
<td>Generate the server key and certificate, and sign with the CA certificate by running the following command:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj '/CN= &lt;your_domain_name&gt;' &amp;&amp; openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td><strong>Important</strong>: Make sure sure that you replace &lt;your_domain_name&gt; with your existing domain name.</td>
<td></td>
</tr>
<tr>
<td>Generate the client key and certificate, and sign with the CA certificate.</td>
<td>Generate the client key and certificate, and sign with the CA certificate by running the following command:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj '/CN=Test' &amp;&amp; openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 02 -out client.crt</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
# Deploy the NGINX ingress controller

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the NGINX ingress controller in your Amazon EKS cluster. | Deploy the NGINX ingress controller by using the following command:  
`kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-0.32.0/deploy/static/provider/aws/deploy.yaml` | DevOps engineer |
| Verify that the NGINX ingress controller service is running. | Verify that the NGINX ingress controller service is running by using the following command:  
`kubectl get svc -n ingress-nginx`  
**Important**: Make sure that the field of service address contains the Network Load Balancer’s domain name. | DevOps engineer |

## Create a namespace in the Amazon EKS cluster to test mutual TLS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a namespace in the Amazon EKS cluster.</td>
<td>Run the <code>kubectl create ns mtls</code> command to create a namespace called <code>mtls</code> in your Amazon EKS cluster. This deploys the sample application to test mutual TLS.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

## Create the deployment and service for the sample application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Kubernetes deployment and service in the <code>mtls</code> namespace.</td>
<td>Create the Kubernetes deployment and service in the <code>mtls</code> namespace by running the <code>kubectl create -f mtls.yaml -n mtls</code> command.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Verify that the Kubernetes deployment is created.</td>
<td>Run the following command to verify that the deployment</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>is created and has one pod in available status:</td>
<td>kubectl get deploy -n mtls</td>
<td></td>
</tr>
<tr>
<td>Verify that the Kubernetes service is created.</td>
<td>Verify that the Kubernetes service is created by running the following command:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>kubectl get service -n mtls</td>
<td></td>
</tr>
</tbody>
</table>

Create a secret in the mtls namespace

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a secret for the ingress resource.</td>
<td>Run the following command to create a secret for the NGINX ingress controller using the certificates that you created earlier:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>kubectl create secret generic mtls-certs --from-file=tls.crt=server.crt --from-file=tls.key=server.key --from-file=ca.crt=ca.crt -n mtls</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Your secret has a server certificate for the client to identify the server and a CA certificate for the server to verify the client certificates.</td>
<td></td>
</tr>
</tbody>
</table>

Create the ingress resource in the mtls namespace

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the ingress resource in the mtls namespace.</td>
<td>Open the ingress.yaml file and replace &lt;your_domain_name&gt; with your existing domain name.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>Create the ingress resource in the mtls namespace by running the following command:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>kubectl create -f ingress.yaml -n mtls</td>
<td></td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This means that the NGINX ingress controller can route traffic to your sample application.</td>
<td></td>
</tr>
<tr>
<td>Verify that the ingress resource is created.</td>
<td>Verify that the ingress resource is created by running the following command:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>kubectl get ing -n mtls</td>
<td></td>
</tr>
<tr>
<td>Important: Make sure that the address of the ingress resource shows the load balancer created for the NGINX ingress controller.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Configure DNS to point the hostname to the load balancer

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create CNAME record that points to the load balancer for the NGINX ingress controller.</td>
<td>Sign in to the AWS Management Console, open the Amazon Route 53 console, and create a Canonical Name (CNAME) record that points mtls.&lt;your_domain_name&gt; to the load balancer for the NGINX ingress controller. For more information about this, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/creating-records-console.html">Creating records by using the Route 53 console</a> in the Route 53 documentation.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

Test the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test mutual TLS setup without certificates.</td>
<td>Run the <code>curl -k https://mtls.&lt;your_domain_name&gt;</code> command. You should receive the &quot;400 No required SSL certificate was sent&quot; error response.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Test mutual TLS setup with certificates.</td>
<td>Run the <code>curl -k https://mtls.&lt;your_domain_name&gt; --cert client.crt --key client.key</code> command. You should receive the &quot;mTLS is working&quot; response.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
Related resources

- Creating records by using the Amazon Route 53 console
- Using a Network Load Balancer with the NGINX ingress controller on Amazon EKS
- Client Certificate Authentication

Attachments

attachment.zip

Create a CI/CD pipeline to deploy microservices with AWS Fargate and Amazon API Gateway

Created by Jonathan Cardoso (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>DevOps; Containers &amp; microservices</th>
<th>AWS services:</th>
<th>AWS CodeBuild; AWS CodeCommit; AWS CodeDeploy; AWS CodePipeline; AWS Fargate</th>
</tr>
</thead>
</table>

Summary

This pattern describes how to create a continuous integration and continuous deployment (CI/CD) pipeline to deploy microservices in the Amazon Web Services (AWS) Cloud. Traditionally, building a modern application involves decoupling and optimizing business logic into a microservices architecture to reduce complexity and to improve speed and alignment. APIs, which abstract communications among services, are the foundation of microservices. Having a CI/CD pipeline for deploying microservices and APIs can accelerate the development of modern applications and help maintain system stability and security.

This pattern uses AWS Fargate to provision compute resources for microservices and Amazon API Gateway to manage the APIs that handle communications among the microservices efficiently. The CI/CD pipeline is set up by implementing three AWS services: AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy. The pattern provides AWS CloudFormation templates and Python code to automate most of the steps, and includes a sample application.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2 or later (see Installing, updating, and uninstalling the AWS CLI version 2 in the AWS documentation)

Architecture

The pattern uses the following AWS services, as illustrated in the following diagram:
• In the private subnets, AWS Fargate manages the compute resources for microservices, to remove the infrastructure burden.
• API Gateway handles API calls and makes it easier to create, maintain, and secure APIs at any scale.
• AWS KMS creates and controls the keys for encrypting content such as artifacts and reports.
• Amazon Simple Storage Service (Amazon S3) is used as a repository to store artifacts and reports that are generated by the pipeline.

The following diagram illustrates the CI/CD workflow.
Tools

AWS services

- **API Gateway** – Amazon API Gateway helps create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs for use in your applications and microservices.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that you can use to interact with AWS services from the command line.
- **CodeCommit** – AWS CodeCommit is a fully managed source control service that stores and manages documents, source code, and binary files in the AWS Cloud.
- **CodePipeline** – AWS CodePipeline is a continuous delivery service that helps you model and configure the different stages of a software release process.
- **CodeBuild** – AWS CodeBuild is a build service that compiles source code, runs unit tests, and produces software packages that are ready to deploy.
- **CodeDeploy** – AWS CodeDeploy is a software deployment service that automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) instances, on-premises instances, serverless AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- **Fargate** – AWS Fargate is a serverless compute engine that works with Amazon ECS and provides on-demand, right-sized compute capacity for containers.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that lets you store and retrieve any amount of data at any time, from anywhere on the web.

Code

The code for this pattern is attached. You can download and unzip the project on your computer to use the files that are referenced in the Epics section.
Epics

Configure your credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM user.</td>
<td>In your AWS account, create an AWS Identity and Access Management (IAM) user. For instructions, see the <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id%E4%B8%BA%E4%B8%80%E4%BD%93_iam-howto_create-user.html">IAM documentation</a>.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Configure your credentials.</td>
<td>Assign the IAM user programmatic access, so they can use the AWS CLI. For instructions, see the <a href="https://docs.aws.amazon.com/cli/latest/userguide/awscli-howto-iam.html">AWS CLI documentation</a>.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Test your access.</td>
<td>To validate that you configured the IAM user correctly, use the following AWS CLI command:</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>```python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws sts get-caller-identity --query Account --output text --profile &lt;profile&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>where <code>&lt;profile&gt;</code> specifies a profile from your credential file. The command outputs your account ID.</td>
<td></td>
</tr>
<tr>
<td>Set up the AWS infrastructure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run the CloudFormation template to deploy the infrastructure.</td>
<td>In your AWS account, create a virtual private cloud (VPC) that is configured with public and private subnets across two Availability Zones and an Application Load Balancer. To deploy this infrastructure, use the following AWS CLI command:</td>
<td>Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>```python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws cloudformation deploy \ --stack-name foundation \ --template-file foundation/full.yaml \ --capabilities CAPABILITY_IAM \ --region &lt;region&gt; \ --profile &lt;profile&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Create and deploy the CI/CD pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a pipeline</td>
<td>Open the Cloudformation console and create a new stack by launching the pipeline.yaml template, which is in the attachment. Configure the following parameters:</td>
<td>Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>• Stack name represents the pipeline for the application (default is myapp).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ServiceName is the application name, and should match the service name in myapp/templates/service.yaml (default is myapp).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• BranchName specifies a branch in your code repository (default is main).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Email is the email address to use for notifications when manual approval is enabled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ManualApproval indicates whether manual approval is required for pipeline operations (default is true).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This stack creates the following resources:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• A CodeCommit repository, using the ServiceName parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• An Amazon Elastic Container Registry (Amazon ECR) repository to store your Docker images</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• A CodePipeline pipeline for your application</td>
<td></td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>• If Manual Approval is set to true, an Amazon Simple Notification Service (Amazon SNS) topic that notifies the email address when there's a commit operation in the pipeline (you receive a confirmation email to confirm your subscription to the SNS topic)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The created pipeline contains these stages:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Source: Monitors any changes in the CodeCommit repository.</td>
<td></td>
<td>Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>• Continuous Integration: Tests your application by using the tools git-secrets, unitest, cfnnag, hadolint, Dependency-Check, Bandit, Safety, and Trivy.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Publish: Builds and pushes the Docker image to the Docker image repository.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Deploy: After manual approval (if enabled), launches the CloudFormation template to deploy this new service version.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check the deployment.</td>
<td>You can check each step of your pipeline that was created, from source to deployment, in the CodePipeline console.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The pipeline won't run yet because the repository is empty. Next, you set up your sample application and start the pipeline.</td>
<td></td>
</tr>
<tr>
<td>Publish the application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>Clone your CodeCommit repository.</td>
<td>Open the AWS CodeCommit console, select the repository created by the pipeline, and then choose Clone URL and HTTPS (as the protocol) to copy the URL.</td>
<td>Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>At the terminal or command prompt, clone the repository by using the command:</td>
<td><strong>git clone &lt;repoURL&gt;</strong>&lt;br&gt;where <code>&lt;repoURL&gt;</code> refers to the clone URL you obtained from CodeCommit. For more information, see the CodeCommit documentation.</td>
<td></td>
</tr>
<tr>
<td>Copy the sample application files.</td>
<td>After you clone your repository, copy the content in the attachment, and place it in your new repository. You can use the following command on Linux or macOS:&lt;br&gt;*<em>cp -rpf ../aws-cicd-microservices-api/myapp/</em> &lt;reponame&gt;/**&lt;br&gt;where <code>&lt;reponame&gt;</code> refers to the name of your CodeCommit repository.</td>
<td>Developer</td>
</tr>
<tr>
<td>Configure your application parameters.</td>
<td>Change the following two files to create your microservice.&lt;br&gt;In the <strong>templates/service.yaml</strong> file:&lt;br&gt;• <strong>ServiceName</strong>: <code>myapp</code> (same as in pipeline.yaml)&lt;br&gt;• <strong>ServicePath</strong>: <code>/myapp</code> (same as in app/main.py)&lt;br&gt;• <strong>BranchName</strong>: <code>main</code>&lt;br&gt;• <strong>AlbRulePriority</strong>: 2 (each microservice should have a unique AlbRulePriority number; increase this number by one for each microservice)&lt;br&gt;&lt;br&gt;In the <strong>app/main.py</strong> file:&lt;br&gt;<code>mypath=&quot;myapp&quot;</code></td>
<td>Developer</td>
</tr>
</tbody>
</table>
Create a custom log parser for Amazon ECS using a Firelens log router

Created by Varun Sharma (AWS)

**Environment:** Production  
**Technologies:** DevOps; Containers & microservices  
**Workload:** All other workloads  
**AWS services:** Amazon ECS

**Summary**

Firelens is a log router for Amazon Elastic Container Service (Amazon ECS) and AWS Fargate. You can use Firelens to route container logs from Amazon ECS to Amazon CloudWatch and other destinations (for example, Splunk or Sumo Logic). Firelens works with Fluentd or Fluent Bit as the logging agent, which means that you can use Amazon ECS task definition parameters to route logs.

By choosing to parse logs at the source level, you can analyze your logging data and perform queries to more efficiently and effectively respond to operational issues. Because different applications have
different logging patterns, you need to use a custom parser that structures the logs and makes searching easier at your end destination.

This pattern uses a Firelens log router with a custom parser to push logs to CloudWatch from a sample Spring Boot application running on Amazon ECS. You can then use Amazon CloudWatch Logs Insights to filter the logs based on custom fields that are generated by the custom parser.

**Prerequisites and limitations**

**Prerequisites**

- An active Amazon Web Services (AWS) account.
- AWS Command Line Interface (AWS CLI), installed and configured on your local machine.
- Docker, installed and configured on your local machine.
- An existing Spring Boot-based containerized application on Amazon Elastic Container Registry (Amazon ECR).

**Architecture**

**Technology stack**

- CloudWatch
- Amazon ECR
- Amazon ECS
- Fargate
- Docker
- Fluent Bit

**Tools**

- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster.
• **AWS Identity and Access Management (IAM)** – IAM is a web service for securely controlling access to AWS services.
• **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
• **Docker** – Docker is an open platform for developing, shipping, and running applications.

**Epics**

**Create a custom Fluent Bit image**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon ECR repository.</td>
<td>Sign in to the AWS Management Console, open the Amazon ECR console, and create a repository called fluentbit_custom. For more information about this, see Creating a repository in the Amazon ECR documentation.</td>
<td>Systems administrator, Developer</td>
</tr>
</tbody>
</table>
| Unzip the customFluentBit.zip package. | 1. Download the customFluentBit.zip package (attached) to your local machine.  
2. Unzip to the customFluentBit directory by running the following command: `unzip -d customFluentBit.zip`  
3. The directory contains the following files that are required for adding the custom parsing and configurations:  
  - parsers/springboot_parser.conf – Contains the parser directive and defines the regular expression (regex) pattern for the custom parser. You can add the regex pattern for your specific parser. | |

**Code**

The following files are attached to this pattern:

- customFluentBit.zip – Contains the files to add the custom parsing and configurations.
- firelens_policy.json – Contains the policy document to create an IAM policy.
- Task.json – Contains a sample task definition for Amazon ECS.
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Create the custom Docker image.** | 1. Change the directory to `customFluentBit`.  
2. Open the Amazon ECR console, choose the `fluentbit_custom` repository, and then choose **View push commands**.  
3. Upload your project.  
4. After the upload is complete, copy the build's URL. This URL is required when you create a container in Amazon ECS.  
For more information about this, see **Pushing a Docker image** in the Amazon ECR documentation. | **Systems administrator, Developer** |

**Set up the Amazon ECS cluster**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Create an Amazon ECS cluster.** | Create an Amazon ECS cluster by following the instructions from the **Networking only template** section of **Creating a cluster** in the Amazon ECS documentation.  
**Note:** Make sure that you choose **Create VPC** to create a new virtual private cloud (VPC) for your Amazon ECS cluster. | **Systems administrator, Developer** |

**Set up the Amazon ECS task**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set up the Amazon ECS task execution IAM role.</strong></td>
<td>Create an Amazon ECS task execution IAM role by using the <code>AmazonECSTaskExecutionRolePolicy</code> managed policy. For more information about this, see <strong>Amazon ECS task execution</strong></td>
<td><strong>Systems administrator, Developer</strong></td>
</tr>
</tbody>
</table>
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM role in the Amazon ECS documentation.</td>
<td><strong>Note:</strong> Make sure that you record the IAM role's Amazon Resource Name (ARN).</td>
<td>Systems administrator, Developer</td>
</tr>
</tbody>
</table>
| Attach the IAM policy to the Amazon ECS task execution IAM role. | 1. Create an IAM policy by using the `firelens_policy.json` (attached) policy document. For more information about this, see Creating policies on the JSON tab in the IAM documentation.  
2. Attach this policy to the Amazon ECS task execution IAM role that you created earlier. For more information about this, see Adding IAM policies (AWS CLI) in the IAM documentation. | Systems administrator, Developer |
## Set up the Amazon ECS task definition.

1. Update the following sections in the `Task.json` sample task definition (attached):
   - Update the `executionRoleArn` and `taskRoleArn` with the ARN of the task execution IAM role
   - Update the image in `containerDefinitions` with the custom Fluent Bit Docker image that you created earlier
   - Update the image in `containerDefinitions` with your application image's name

2. Open the Amazon ECS console, choose **Task Definitions**, choose **Create new task definition**, and then choose **Fargate** on the **Select compatibilities** page.

3. Choose **Configure via Json**, paste the updated `Task.json` file into the text area, and then choose **Save**.

4. Create the task definition.

For more information about this, see [Creating a task definition](https://docs.aws.amazon.com/ecs/latest/userguide/task-definitions.html) in the Amazon ECS documentation.

### Run the Amazon ECS task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the Amazon ECS task.</td>
<td>On the Amazon ECS console, choose <strong>Clusters</strong>, choose the cluster that you created earlier, and then run the standalone task.</td>
<td>Systems administrator, Developer</td>
</tr>
</tbody>
</table>

For more information about this, see [Run a standalone task](https://docs.aws.amazon.com/ecs/latest/userguide/running-standalone-tasks.html) in the Amazon ECS documentation.
Verify the CloudWatch logs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Verify the logs. | 1. Open the CloudWatch console, choose **Log groups**, and then choose /aws/ecs/containerinsights/{{cluster_ARN}}/firelens/application.  
2. Verify the logs, particularly the custom fields added by the custom parser.  
3. Use CloudWatch to filter logs based on the custom fields. | Systems administrator, Developer |

Related resources

- Docker basics for Amazon ECS
- Amazon ECS on AWS Fargate
- Configuring basic service parameters

Attachments

attachment.zip

Create a pipeline and AMI using CodePipeline and HashiCorp Packer

*Created by Akash Kumar (AWS)*

<table>
<thead>
<tr>
<th>Environment: PoC or pilot</th>
<th>Source: DevOps</th>
<th>Target: Amazon Machine Images(AMI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type: Rehost</td>
<td>Workload: All other workloads</td>
<td>Technologies: DevOps; Modernization; Websites &amp; web apps</td>
</tr>
</tbody>
</table>

Summary

This pattern provides code samples and steps to create both a pipeline in the Amazon Web Services (AWS) Cloud by using AWS CodePipeline and an Amazon Machine Image (AMI) by using HashiCorp Packer. The pattern is based on the **continuous integration** practice, which automates the building and testing of code with a Git-based version control system. In this pattern, you create and clone a code repository by using AWS CodeCommit. Then, create a project and configure your source code by using AWS CodeBuild. Finally, create an AMI that gets committed to your repository.
Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Linux AMI for launching Amazon Elastic Compute Cloud (Amazon EC2) instances
- HashiCorp Packer 0.12.3 or later
- Amazon CloudWatch Events (optional)
- Amazon CloudWatch Logs (optional)

Architecture

The following diagram shows an example of application code that automates the creation of an AMI by using this pattern’s architecture.

The diagram shows the following workflow:

1. The developer commits code changes to a private CodeCommit Git repository. Then, CodePipeline uses CodeBuild to initiate the build and add new artifacts that are ready for deployment to the Amazon Simple Storage Service (Amazon S3) bucket.

2. CodeBuild uses Packer to bundle and package the AMI based on a JSON template. If enabled, CloudWatch Events can automatically start the pipeline when a change occurs in the source code.

Technology stack

- CodeBuild
- CodeCommit
- CodePipeline
- CloudWatch Events (optional)
Tools

- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeCommit** – AWS CodeCommit is a version control service that enables you to privately store and manage Git repositories in the AWS Cloud. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure.
- **AWS CodePipeline** – AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software.
- **HashiCorp Packer** – HashiCorp Packer is an open-source tool for automating the creation of identical machine images from a single source configuration. Packer is lightweight, runs on every major operating system, and creates machine images for multiple platforms in parallel.

Code

This pattern includes the following attachments:

- `buildspec.yml` – This file uses CodeBuild to build and create an artifact for deployment.
- `amazon-linux_packer-template.json` – This file uses Packer to create an Amazon Linux AMI.

Epics

Set up the code repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the repository.</td>
<td>Create a CodeCommit repository.</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>Clone the repository.</td>
<td>Connect to the CodeCommit repository by cloning the repository.</td>
<td>App developer</td>
</tr>
<tr>
<td>Push the source code to the remote repository.</td>
<td>1. Create a commit to add the <code>buildspec.yml</code> and <code>amazon-linux_packer-template.json</code> files to your local repository.  &lt;br&gt; 2. Push the commit from your local repository to the remote CodeCommit repository.</td>
<td>App developer</td>
</tr>
</tbody>
</table>

Create a CodeBuild project for the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a build project.</td>
<td>1. Sign in to the AWS Management console, open the AWS CodeBuild console, and then choose Create build project.</td>
<td>App developer, AWS systems administrator</td>
</tr>
</tbody>
</table>
### Set up the pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline name</td>
<td></td>
<td>App developer, AWS systems administrator</td>
</tr>
<tr>
<td></td>
<td>1. Sign in to the AWS Management console, open the <strong>AWS CodePipeline</strong></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1.</td>
<td>console, and then choose Create pipeline.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>For Pipeline name, enter a name for the pipeline.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>For Service role, choose New service role or Existing service role.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>For Role name, enter a name for your role.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>In the Advanced settings section, for Artifact store, choose Default location if you want Amazon S3 to create a bucket and store the artifacts in the bucket. To use an existing S3 bucket, choose Custom location. Choose Next.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>For Source provider, choose AWS CodeCommit.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>For Repository name, choose the repository that you cloned earlier. For Branch name, choose your source code branch.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>For Change detection options, choose Amazon CloudWatch Events (recommended) to start the pipeline or AWS CodePipeline to periodically check for changes. Choose Next.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>For Build provider, choose AWS CodeBuild.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>For Project Name, choose the build project that you created in the Create a CodeBuild project for the application epic.</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Choose your build options and then choose Next.</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>Choose Skip deploy stage.</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Choose Create pipeline.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- Working with repositories in AWS CodeCommit
- Working with build projects
- Working with pipelines in CodePipeline

451
Create a pipeline and deploy artifact updates to on-premises EC2 instances using CodePipeline

**Created by Akash Kumar (AWS)**

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Source:</th>
<th>Target:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoC or pilot</td>
<td>DevOps</td>
<td>Amazon EC2/On-Premises</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Technologies:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rehost</td>
<td>DevOps; Modernization; Websites &amp; web apps</td>
</tr>
</tbody>
</table>

**Summary**

This pattern provides code samples and steps to create a pipeline in the Amazon Web Services (AWS) Cloud and deploy updated artifacts to on-premises Amazon Elastic Compute Cloud (Amazon EC2) instances in AWS CodePipeline. The pattern is based on the continuous integration practice, which automates the building and testing of code with a Git-based version control system. In this pattern, you create and clone a code repository by using AWS CodeCommit. Then, create a project and configure your source code by using AWS CodeBuild. Finally, create your application and configure its target environment for on-premises EC2 instances by using AWS CodeDeploy.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- User-defined tags to identify EC2 instances during deployment
- CodeDeploy agent, installed on EC2 instances
- Your required runtime software, installed on EC2 instances
- Amazon Corretto 8 for the Java Development Kit
- Apache Tomcat web server, installed
- Amazon CloudWatch Events (optional)
- A key pair to log in to the web server (optional)

**Architecture**

The following diagram shows an example Java web application that’s deployed to on-premises EC2 instances by using this pattern’s architecture.
The diagram shows the following workflow:

1. The developer commits code changes to a private CodeCommit Git repository.
2. CodePipeline uses CodeBuild to initiate the build and add new artifacts that are ready for deployment in the Amazon Simple Storage Service (Amazon S3) bucket.
3. CodePipeline uses the CodeDeploy agent to pre-install any dependencies required for the deployment artifact changes.
4. CodePipeline uses the CodeDeploy agent to deploy the artifacts from the S3 bucket to target EC2 instances. If enabled, CloudWatch Events can automatically start the pipeline when a change occurs in the source code.

Technology stack

- CodeBuild
- CodeCommit
- CodeDeploy
- CodePipeline
- CloudWatch Events (optional)

**Tools**

- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeCommit** – AWS CodeCommit is a version control service that enables you to privately store and manage Git repositories in the AWS Cloud. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure.
- **AWS CodeDeploy** – AWS CodeDeploy is a deployment service that automates application deployments to EC2 instances, on-premises instances, serverless AWS Lambda functions, or Amazon Elastic
Container Service (Amazon ECS) services. CodeDeploy works with various systems for configuration management, source control, continuous integration, continuous delivery, and continuous deployment.

- AWS CodePipeline – AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software.

### Code

This pattern includes the following attachments:

- `buildspec.yml` – This file specifies the actions that CodeBuild requires to build and create an artifact for deployment.
- `appspec.yml` – This file specifies the actions that CodeDeploy requires to create an application and configure a target environment for on-premises EC2 instances.
- `install_dependencies.sh` – This file installs dependencies for the Apache Tomcat web server.
- `start_server.sh` – This file starts the Apache Tomcat web server.
- `stop_server.sh` – This file stops the Apache Tomcat web server.

### Epics

#### Set up the code repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the repository.</td>
<td>Create a CodeCommit repository.</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>Clone the repository.</td>
<td>Connect to the CodeCommit repository by cloning the repository.</td>
<td>App developer</td>
</tr>
<tr>
<td>Push the source code to the remote repository.</td>
<td>1. Create a commit to add the <code>buildspec.yml</code> and <code>appspec.yml</code> files to your local repository. 2. Push the commit from your local repository to the remote CodeCommit repository.</td>
<td>App developer</td>
</tr>
</tbody>
</table>

#### Create a CodeBuild project for the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a build project.</td>
<td>1. Sign in to the AWS Management console, open the AWS CodeBuild console, and then choose Create build project. 2. For Project name, enter the name of your project. 3. For Source provider, choose AWS CodeCommit.</td>
<td>AWS administrator, App developer</td>
</tr>
</tbody>
</table>
## Configure artifact deployment for on-premises EC2 instances

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create the application.</strong></td>
<td></td>
<td>AWS systems administrator, App developer</td>
</tr>
<tr>
<td>1. Sign in to the AWS Management console, open the AWS CodeDeploy console, and then choose Create application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. For Application name, enter a name for your application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. For Compute platform, choose EC2/On-premises.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Choose Create application and then choose Create deployment group.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. For Deployment group name, enter a name.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>Create a service role for CodeDeploy. <strong>Note:</strong> The service role must have permissions to grant CodeDeploy access to your target environment.</td>
<td>AWS systems administrator, App developer</td>
</tr>
<tr>
<td>7.</td>
<td>For <strong>Service role</strong>, choose the service role that you created in step 6.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>For <strong>Deployment type</strong>, choose either <strong>In-place</strong> or <strong>Blue/green</strong> based on your business requirements.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>For <strong>Environment configuration</strong>, choose the options that meet your business requirements.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>(Optional) <strong>Create a target group</strong> for your load balancer separately in the Amazon EC2 console, and then go back to the <strong>Create deployment group</strong> page of the AWS CodeDeploy console to choose your load balancer and target group.</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Choose <strong>Create deployment group</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

### Set up the pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Sign in to the AWS Management console, open the AWS CodePipeline console, and then choose <strong>Create pipeline</strong>.</td>
<td>AWS systems administrator, App developer</td>
</tr>
<tr>
<td>the pipeline.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>For <strong>Pipeline name</strong>, enter a name for the pipeline.</td>
<td></td>
</tr>
<tr>
<td>the pipeline.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>For <strong>Service role</strong>, choose <strong>New service role</strong> or <strong>Existing service role</strong>.</td>
<td></td>
</tr>
<tr>
<td>the pipeline.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>For <strong>Role name</strong>, enter a name for your role.</td>
<td></td>
</tr>
<tr>
<td>the pipeline.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>In the <strong>Advanced settings</strong> section, for <strong>Artifact store</strong>, choose <strong>Default location</strong> if you want Amazon S3 to create a bucket and store the artifacts in the bucket. To use</td>
<td></td>
</tr>
<tr>
<td>the pipeline.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>an existing S3 bucket, choose <strong>Custom location</strong>. Choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>For <strong>Source provider</strong>, choose <strong>AWS CodeCommit</strong>.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>For <strong>Repository name</strong>, choose the repository that you cloned earlier. For <strong>Branch name</strong>, choose your source code branch.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>For <strong>Change detection options</strong>, choose <strong>Amazon CloudWatch Events</strong> <em>(recommended)</em> or <strong>AWS CodePipeline</strong>. Choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>For <strong>Build provider</strong>, choose <strong>AWS CodeBuild</strong>.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>For <strong>Project Name</strong>, choose the build project that you created in the <em>Create a CodeBuild project for the application epic</em>.</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Choose your build options and then choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>For <strong>Deploy provider</strong>, choose <strong>AWS CodeDeploy</strong>.</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Choose an application name and deployment group, and then choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>Choose <strong>Create pipeline</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- Working with repositories in AWS CodeCommit
- Working with build projects
- Working with applications in CodeDeploy
- Working with pipelines in CodePipeline

**Attachments**

attachment.zip

**Deploy a CI/CD pipeline for Java microservices on Amazon ECS**
Summary

This pattern guides you through the steps for deploying a CI/CD (continuous integration and continuous delivery) pipeline for Java microservices on an existing Amazon Elastic Container Service (Amazon ECS) cluster using AWS CodeBuild. When the developer commits changes, the CI/CD pipeline is triggered. The build process starts in CodeBuild, and when it’s complete, the artifact gets pushed to Amazon Elastic Container Registry (Amazon ECR). The latest build from Amazon ECR is picked up and pushed to the Amazon ECS service.

Prerequisites and limitations

Prerequisites

- An existing Java microservices application running on Amazon ECS
- Familiarity with AWS CodeBuild and AWS CodePipeline

Architecture

Source technology stack

- Java microservices running on Amazon ECS
- Code repository in Amazon ECR

Source architecture
Target technology stack

- Amazon ECR
- Amazon ECS
- AWS Fargate

Target architecture
CodeBuild buildspec.yml file:

```yaml
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=build-$(echo $CODEBUILD_BUILD_ID | awk -F:" '{print $2}"
build:
  commands:
    - echo Build started on `date`
    - mvn clean install
    - docker build -t $REPOSITORY_URI:latest .
    - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
```

Automation and scale
Tools

- **Amazon ECR** - Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS to simplify your development-to-production workflow. Amazon ECR hosts your images in a highly available and scalable architecture so you can reliably deploy containers for your applications. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.

- **Amazon ECS** - Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need for you to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.

- **AWS Fargate** - AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.

- **Docker** - Docker is a platform that lets you build, test, and deliver applications in packages called containers.

- **CodeBuild** - AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. AWS CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left in the queue.

- **CodePipeline** - AWS CodePipeline is a fully managed continuous delivery service that helps you automate release pipelines for fast and reliable application and infrastructure updates. AWS CodePipeline automates the build, test, and deploy phases of the release process every time there is a code change. With AWS CodePipeline, you can rapidly and reliably deliver features and updates. You can integrate AWS CodePipeline with third-party services like GitHub, or use an AWS services such as AWS CodeCommit or Amazon ECR.

- **Git** - Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. You can also use AWS CodeCommit as an alternative to Git.

Code

- A .zip file of the project is available as an attachment.
# Epics

## Set up the build project in AWS CodeBuild

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CodeBuild build project.</td>
<td>In the AWS CodeBuild console at <a href="https://console.aws.amazon.com/codesuite/codebuild/home">https://console.aws.amazon.com/codesuite/codebuild/home</a>, create a build project, and specify its name.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Select the source.</td>
<td>This pattern uses Git for the code repository, so choose Git from the available options.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Select a repository.</td>
<td>Select the repository from which you want to build the code.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Select the environment.</td>
<td>You can select from a list of managed images or opt for a custom image using Docker. This pattern uses a managed image (Ubuntu).</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Choose a service role.</td>
<td>You can create a service role or select from a list of existing roles.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Create a buildspec file.</td>
<td>You can create a buildspec.yml file and add the configuration, or use the online buildspec editor to configure the changes.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Opt for artifacts (optional).</td>
<td>Configure the build project for artifacts, if required.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Configure Amazon CloudWatch logs.</td>
<td>Configure the build project Amazon CloudWatch logs, if required. This step is optional but recommended. Charges are applicable.</td>
<td>Developer, System Admin</td>
</tr>
<tr>
<td>Opt for Amazon S3 logs (optional).</td>
<td>Configure the build project for Amazon Simple Storage Service (Amazon S3) logs, if you want to store the logs.</td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>

## Configure the pipeline in AWS CodePipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a pipeline.</td>
<td>In the AWS CodePipeline console at <a href="https://console.aws.amazon.com/">https://console.aws.amazon.com/</a></td>
<td>Developer, System Admin</td>
</tr>
</tbody>
</table>
### Task | Description | Skills required
--- | --- | ---
| codesuite/codepipeline/home, create a pipeline and specify its name. | Select a service role. | Developer, System Admin
| Create a service role or select from the list of existing service roles. | Opt for an artifact store. | Developer, System Admin
| If you want Amazon S3 to create a bucket and store the artifacts in it, use the default location. Or, select a custom location and specify an existing bucket. | Specify the source provider. | Developer, System Admin
| Choose GitHub from the list of source providers. | Select the repository and branch of the code. | Developer, System Admin
| The options are GitHub Webhooks and AWS CodePipeline. This pattern uses GitHub Webhooks. | Change detection options. | Developer, System Admin
| Select AWS CodeBuild as the build provider, then select the project name. You can choose Jenkins if your code was built using Jenkins. | Select a build provider. | Developer, System Admin
| Select Amazon ECS as the deploy provider from the list of available options. Select the cluster name, the service name, the image definitions file, if any, and a deployment timeout value, if required. Choose “Create pipeline.” | Choose a deploy provider. | Developer, System Admin

### Related resources
- AWS ECS documentation
- AWS ECR documentation
- AWS CodeBuild documentation
- AWS CodeCommit documentation
- AWS CodePipeline documentation
- Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source (blog post)

### Attachments
- attachment.zip
Use AWS CodeCommit and AWS CodePipeline to deploy a CI/CD pipeline in multiple AWS accounts

Created by Kirankumar Chandrashekar (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>DevOps</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS CodeCommit; AWS CodePipeline</td>
<td>AWS CodePipeline</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern shows you how to deploy a continuous integration / continuous delivery (CI/CD) pipeline for your application code workloads in separate Amazon Web Services (AWS) accounts for DevOps, developer, staging, and production workflows.

You can use a multiple AWS account strategy to provide a high level of resource or security isolation, optimize costs, and separate out your production workflow.

Your application's code remains identical in all these separate AWS accounts and is maintained on a central AWS CodeCommit repository hosted by your DevOps account. Your developer, staging, and production accounts have separate Git branches in this CodeCommit repository.

For example, when code is committed to the developer Git branch in your central CodeCommit repository, Amazon EventBridge in your DevOps account notifies EventBridge in your developer account of the repository changes. In your developer account, AWS CodePipeline and the source stage go into InProgress status. The source stage is configured from the developer Git branch in the central CodeCommit repository and CodePipeline assumes a service role for the DevOps account.

The contents of the CodeCommit repository in the developer branch are uploaded to an artifact store in an Amazon Simple Storage Service (Amazon S3) bucket and encrypted with an AWS Key Management Service (AWS KMS) key. After the source stage's status changes to Succeeded in CodePipeline, the code will be transitioned to the next stage of the pipeline execution.

**Prerequisites and limitations**

**Prerequisites**

- Existing AWS accounts for each required environment (DevOps, developer, staging, and production). These accounts can be hosted by AWS Organizations.
- AWS Command Line Interface (AWS CLI), installed and configured.
Architecture

Technology stack

- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- Amazon EventBridge
- AWS Identity and Access Management (IAM)
- AWS KMS
- AWS Organizations
- Amazon S3

Tools

- AWS CodeBuild – CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.
- AWS CodeCommit – CodeCommit is a fully-managed source-control service that hosts secure Git-based repositories
- AWS CodePipeline – CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates.
- Amazon EventBridge – EventBridge is a serverless event bus service for connecting your applications with data from a variety of sources.
- AWS Identity and Access Management (IAM) – IAM helps you to manage access to AWS services and resources securely.
- **AWS KMS** – AWS Key Management Service (AWS KMS) helps you create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

## Epics

### Create resources in your DevOps AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CodeCommit repository.</td>
<td>Sign in to the AWS Management Console for your DevOps account, and open the CodeCommit console. Create a repository and set up all the required Git branches for your developer, staging, and production AWS accounts. For help with this and other stories, see the &quot;Related resources&quot; section.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Create access credentials for the CodeCommit repository.</td>
<td>On the IAM console, create access credentials to allow application developers to push and pull the application's code base from the CodeCommit repository.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Create an IAM role for CodePipeline service roles.</td>
<td>On the IAM console, create an IAM role that can be used by all your CodePipeline service roles to access the central CodeCommit repository.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Set up the EventBridge rules for your other AWS accounts.</td>
<td>On the Amazon EventBridge console, set up rules to send notifications about relevant CodeCommit repository changes to EventBridge in the individual developer, staging, and production AWS accounts.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an AWS KMS key.</td>
<td>On the AWS KMS console, create a KMS key that allows CodePipeline in your individual developer, staging, and production AWS accounts to encrypt and decrypt artifacts.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
Create resources in your other AWS accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up EventBridge to receive events from the DevOps AWS account.</td>
<td>Sign in to the AWS Management Console for one of your individual AWS accounts (developer, staging, or production). On the Amazon EventBridge console, set up EventBridge to receive CodeCommit repository change events from your DevOps account.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an S3 bucket.</td>
<td>On the Amazon S3 console, create an S3 bucket to store CodePipeline artifacts.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create all required AWS resources for CodePipeline stages.</td>
<td>Create all the other AWS resources that will be required by the CodePipeline stages. These resources will vary depending on the role of each AWS account in your CI/CD pipeline.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an IAM role.</td>
<td>On the IAM console, create an IAM role for the CodePipeline service role. This service role must be able to assume the IAM role in the DevOps account to access the CodeCommit repository.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create a pipeline in CodePipeline.</td>
<td>On the CodePipeline console, create a pipeline. Then create a source stage that points to the CodeCommit repository in the DevOps account for its individual Git branch.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Repeat the steps for all your AWS accounts.</td>
<td>Repeat these steps for all the AWS accounts that are required as part of your CI/CD strategy.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

Related resources

Create resources in your DevOps AWS account

- Create a CodeCommit repository
- Set up a CodeCommit repository
- Create and share a branch in your CodeCommit repository
- Create access credentials for the CodeCommit repository
- Create an IAM role for CodePipeline service roles
Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile

**Created by Sagar Panigrahi (AWS)**

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: DevOps; Containers &amp; microservices</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services:</strong> Amazon EKS; AWS Cloud9; AWS Identity and Access Management; AWS CloudFormation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes how to use AWS Cloud9 and AWS CloudFormation to create an Amazon Elastic Kubernetes Service (Amazon EKS) cluster that can be operated without enabling programmatic access for users in your Amazon Web Services (AWS) account.

AWS Cloud9 is a cloud-based integrated development environment (IDE) that helps you write, run, and debug your code by using a browser. AWS Cloud9 is used as a control center that provisions an Amazon EKS cluster by using Amazon Elastic Compute Cloud (Amazon EC2) instance profiles and AWS CloudFormation templates.

You can use this pattern if you don’t want to create AWS Identity and Access Management (IAM) users and want to use IAM roles instead. Role-based access control (RBAC) regulates access to resources based
on the roles of individual users. This pattern demonstrates how to update RBAC within an Amazon EKS cluster to allow access to a specific IAM role.

The pattern’s setup also helps your DevOps team use AWS Cloud9 features to maintain and develop infrastructure as code (IaC) resources for creating Amazon EKS infrastructure.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Permissions to create IAM roles and policies for the account. The IAM role for the user must include the AWSCloud9Administrator policy. The AWSServiceRoleForAmazonEKS and eksNodeRoles roles must also be created because they are required to create an Amazon EKS cluster.
- Knowledge of Kubernetes concepts.

Limitations

- This pattern describes how to create a basic Amazon EKS cluster. For production clusters, you must update the AWS CloudFormation template.
- The pattern doesn’t deploy additional Kubernetes components (for example, Fluentd, ingress controllers, or storage controllers).

Architecture

Technology stack
• AWS Cloud9
• AWS CloudFormation
• Amazon EKS
• IAM

Automation and scale

You can expand this pattern and incorporate it into continuous integration and continuous deployment (CI/CD) pipelines to automate the complete provisioning of Amazon EKS.

Tools

• AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications.
• AWS Cloud9 – AWS Cloud9 offers a rich code-editing experience with support for several programming languages and runtime debuggers, and a built-in terminal.
• AWS CLI – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
• Kubectl – kubectl is a command line utility that you can use to interact with an Amazon EKS cluster.

Epics

Create the IAM roles for the EC2 instance profile

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the IAM policy.</td>
<td>Sign in to the AWS Management Console, open the IAM console, choose Policies, and then choose Create policy. Choose the JSON tab and paste the contents from the policy-role-eks-instance-profile-for-cloud9.json file (attached). Resolve any security warnings, errors, or general warnings generated during the policy validation, and then choose Review policy. Enter a Name for the policy. We recommend that you use eks-instance-profile-for-cloud9 for the policy name. Review the policy Summary to see the permissions that are granted by your policy. Then choose Create policy.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Create an IAM role using the policy.</td>
<td>On the IAM console, choose Roles and then choose Create role. Choose AWS Service and then choose EC2 from the list. Choose Next: Permissions and search for the IAM policy that you created earlier. Choose the appropriate tags for your requirements. In the Review section, enter a name for the role. We recommend that you use role-eks-instance-profile-for-cloud9 for the role name. Then choose Create role.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

### Create an IAM policy and role for the Amazon EKS RBAC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the IAM policy.</td>
<td>On the IAM console, choose Policies and then choose Create policy. Choose the JSON tab and paste the contents from the policy-for-eks-rbac.json file (attached). Resolve any security warnings, errors, or general warnings generated during the policy validation, and then choose Review policy. Enter a Name for the policy. We recommend that you use policy-for-eks-rbac for the policy name. Review the policy Summary to see the permissions that are granted by your policy. Then choose Create policy.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an IAM role using the policy.</td>
<td>On the IAM console, choose Roles and then choose Create role. Choose AWS Service and then choose EC2 from the list. Choose Next: Permissions and search for the IAM policy that you created earlier. Choose the appropriate tags for your requirements. In the Review section, enter a name for the role. We</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
Create the AWS Cloud9 environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS Cloud9 environment.</td>
<td>Open the AWS Cloud9 console and choose Create environment. On the Name environment page, enter a name for your environment. We recommend that you use eks-management-env for the environment name. Configure the remaining settings according to your requirements and then choose Next step. On the Review page, choose Create environment. Wait while AWS Cloud9 creates your environment. This can take several minutes. For more information about the available configuration options, see Creating an EC2 environment in the AWS Cloud9 documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Remove the temporary IAM credentials for AWS Cloud9.</td>
<td>After your AWS Cloud9 environment is provisioned, choose Settings in the gear icon. Under Preferences, choose AWS settings and then choose Credentials. Turn off AWS managed temporary credentials and close the tab.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Attach the EC2 instance profile to the underlying EC2 instance.</td>
<td>Open the Amazon EC2 console and choose the EC2 instance that matches your environment in AWS Cloud9. If you used the name that we recommended, the EC2 instance is called aws-cloud9-eks-management-env. Choose the EC2 instance, choose Actions, and then choose</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns

Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance settings.</strong> Choose <strong>Attach/replace IAM role.</strong> Search for role-eks-instance-profile-for-cloud9 or the name of the IAM role that you created earlier, and then choose <strong>Apply.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Create the Amazon EKS cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create the Amazon EKS cluster.</strong></td>
<td>Download and open the eks-cfn.yaml (attached) template for AWS CloudFormation. Edit the template according to your requirements. Open the AWS Cloud9 environment and choose <strong>New file.</strong> Paste the AWS CloudFormation template that you created earlier into the field. We recommend that you use eks-cfn.yaml for the template name. In the AWS Cloud9 terminal, run the following command to create the Amazon EKS cluster: aws cloudformation create-stack --stack-name eks-cluster --template-body file://eks-cfn.yaml --region &lt;your_AWS_Region&gt; If the AWS CloudFormation call is successful, you receive the AWS CloudFormation stack's Amazon Resource Name (ARN) in your output. The stack creation can take between 10 to 20 minutes.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td><strong>Verify the Amazon EKS cluster’s status.</strong></td>
<td>On the AWS CloudFormation console, open the <strong>Stacks</strong> page and then choose the stack name. The stack is created when the stack status code shows <strong>CREATE_COMPLETE.</strong> For more information, see Viewing AWS CloudFormation stack data</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Access the Kubernetes resources in the Amazon EKS cluster</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install kubectl in the AWS Cloud9 environment.</td>
<td>Install <code>kubectl</code> in your AWS Cloud9 environment by following the instructions from Installing kubectl in the Amazon EKS documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Update the new Amazon EKS configuration in AWS Cloud9.</td>
<td>Run the following command in the AWS Cloud9 terminal to update the <code>kubeconfig</code> from the Amazon EKS cluster to the AWS Cloud9 environment:</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
| | ```
aws eks update-kubeconfig --name EKS-DEV2 --region <your_AWS_Region>
``` | |
| | **Important:** EKS-DEV2 is the name of the Amazon EKS cluster in the AWS CloudFormation template that you used to create the cluster. | |
| | Run the `kubectl get all -A` command to view all Kubernetes resources. | |
| Add the administrator IAM role to the Kubernetes RBAC. | Run the following command in your AWS Cloud9 terminal to open the RBAC configuration map for Amazon EKS in edit mode: | Cloud administrator |
| | ```
kubectl edit cm/aws-auth -n kube-system
``` | |
| | Append the following lines under the `mapRoles` section: | |
| | ```
- groups: 
- system:masters
rolearn: <ARN_of_IAM_role_from_second_epic>
username: eksadmin
``` | |
| | Lint the YAML-formatted file to avoid syntax errors. Save the file | |
## Related resources

### References
- Modular and scalable Amazon EKS architecture *(Quick Start)*
- Managing users or IAM roles for your Amazon EKS cluster
- AWS CloudFormation template to create a new Amazon EKS control plane

### Attachments
attachment.zip

**Deploy code in multiple AWS Regions using AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild**

*Created by Rama Anand Krishna Varanasi (AWS)*

<table>
<thead>
<tr>
<th>Created by:</th>
<th>AWS services: AWS CodeCommit; AWS CodePipeline; AWS CodeBuild</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: PoC or pilot</td>
<td>Technologies: Management &amp; governance; DevOps</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>using <em>vi</em> commands and then exit the file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By adding this section, you inform the Kubernetes RBAC that <code>&lt;ARN_of_IAM_role_from_second_epic&gt;</code> is to receive full administrator access on the Amazon EKS cluster. This means that the identified IAM role can carry out administrative actions on the Kubernetes cluster. AWS adds the existing section under <code>mapRoles</code> while the Amazon EKS cluster is provisioned.</td>
<td></td>
</tr>
</tbody>
</table>
Summary

This pattern demonstrates how to build infrastructure or architecture across multiple Amazon Web Services (AWS) Regions by using AWS CloudFormation. It includes continuous integration (CI)/continuous deployment (CD) across multiple AWS Regions for faster deployments. The steps in this pattern have been tested for the creation of an AWS CodePipeline job to deploy to three AWS Regions as an example. You can change the number of Regions based on your use case.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Two AWS Identity and Access Management (IAM) roles for AWS CodeBuild and AWS CloudFormation with proper policies for CodeBuild to perform the CI tasks of testing, bundling, packaging the artifacts, and deploying to multiple AWS Regions in parallel. **Note:** Cross-check the policies created by CodePipeline to verify that CodeBuild and AWS CloudFormation have proper permissions in the CI and CD phases.
- A CodeBuild role with the `AmazonS3FullAccess` and `CloudWatchFullAccess` policies. These policies give CodeBuild access to watch events of AWS CodeCommit through Amazon CloudWatch and to use Amazon Simple Storage Service (Amazon S3) as an artifact store.
- An AWS CloudFormation role with the following policies, which give AWS CloudFormation, in the final Build stage, the ability to create or update AWS Lambda functions, push or watch Amazon CloudWatch logs, and to create and update change sets.
  - `AWSLambdaFullAccess`
  - `AWSCodeDeployFullAccess`
  - `CloudWatchFullAccess`
  - `AWSCloudFormationFullAccess`
  - `AWSCodePipelineFullAccess`

Architecture

This pattern’s multiple-Region architecture and workflow comprise the following steps.

1. You send your code to a CodeCommit repository.
2. Upon receiving any code update or commit, CodeCommit invokes a CloudWatch event, which in turn starts a CodePipeline job.

3. CodePipeline engages the CI that is handled by CodeBuild. The following tasks are performed.
   - Testing of the AWS CloudFormation templates (optional)
   - Packaging of the AWS CloudFormation templates for each Region included in the deployment. For example, this pattern deploys in parallel to three AWS Regions, so CodeBuild packages the AWS CloudFormation templates into three S3 buckets, one in each specified Region. The S3 buckets are used by CodeBuild as artifact repositories only.

4. CodeBuild packages the artifacts as input for next Deploy phase, which runs in parallel in the three AWS Regions. If you specify a different number of Regions, CodePipeline will deploy to those Regions.

Tools

- AWS CodePipeline – CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software changes continuously.
- AWS CodeBuild – CodeBuild is a fully managed build service that compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit – CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as source code and binary files) in the cloud.
- AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.
- AWS Identity and Access Management – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.

Code

The following sample code is for the `BuildSpec.yaml` file (Build phase).

```yaml
---
artifacts:
discard-paths: true
files:
  - packaged-first-region.yaml
  - packaged-second-region.yaml
  - packaged-third-region.yaml
phases:
  build:
    commands:
      - echo "********BUILD PHASE - CF PACKAGING**********"
      - "aws cloudformation package --template-file sam-template.yaml --s3-bucket $S3_FIRST_REGION --output-template-file packaged-first-region.yaml --region $FIRST_REGION"
      - "aws cloudformation package --template-file sam-template.yaml --s3-bucket $S3_SECOND_REGION --output-template-file packaged-second-region.yaml --region $SECOND_REGION"
      - "aws cloudformation package --template-file sam-template.yaml --s3-bucket $S3_THIRD_REGION --output-template-file packaged-third-region.yaml --region $THIRD_REGION"
  install:
    commands:
      - echo "********BUILD PHASE - PYTHON SETUP**********"
runtime-versions:
```
Epics

Prepare the code and the CodeCommit repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the primary AWS Region for the deployment.</td>
<td>Sign in to your AWS account and choose the primary Region for the deployment. The CodeCommit repository will be in the primary Region.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Create the CodeCommit repository.</td>
<td>Create the CodeCommit repository, and push the required code into it. The code generally includes the AWS CloudFormation or AWS SAM templates, Lambda code if any, and the CodeBuild buildspec.yaml files as input to the AWS CodePipeline.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Push the code into the CodeCommit repository.</td>
<td>In the Attachments section, download the code for this example, and then push the required code into it. Generally, the code can include AWS CloudFormation or AWS SAM templates, Lambda code and the CodeBuild buildspec.yaml files as input to the pipeline.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

Source phase: Create the pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the CodePipeline job.</td>
<td>On the CodePipeline console, choose Create pipeline.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Name the CodePipeline job and choose the service role setting.</td>
<td>Enter a name for the job, and keep the default service role setting so that CodePipeline creates the role with the necessary policies attached.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Specify the location for the artifact store.</td>
<td>Under <strong>Advanced settings</strong>, keep the default option so that CodePipeline creates an S3 bucket to use for code artifact storage. If you use an existing S3 bucket instead, the bucket must be in the primary Region that you specified in the first epic.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the encryption key.</td>
<td>Keep the default option, <strong>Default AWS Managed Key</strong>, or choose to use your own AWS Key Management Service (AWS KMS) customer managed key.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the source provider.</td>
<td>Under <strong>Source provider</strong>, choose <strong>AWS CodeCommit</strong>.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the repository.</td>
<td>Choose the CodeCommit repository that you created in the first epic. If you placed the code in a branch, choose the branch.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify how code changes are detected.</td>
<td>Keep the default, <strong>Amazon CloudWatch Events</strong>, as the change trigger for CodeCommit to start the CodePipeline job.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

**Build phase: Configure the pipeline**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the build provider.</td>
<td>For the build provider, choose <strong>AWS CodeBuild</strong>.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the AWS Region.</td>
<td>Choose the primary Region, which you specified in the first epic.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

**Build phase: Create and configure the project**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the project</td>
<td>Choose <strong>Create project</strong>, and enter a name for the project.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the environment image.</td>
<td>For this pattern demonstration, use the default CodeBuild managed image. You also have the option to use a custom Docker image if you have one.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Specify the operating system.</td>
<td>Choose either Amazon Linux 2 or Ubuntu.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the service role.</td>
<td>Choose the role you created for CodeBuild before you started to create the CodePipeline job. (See the Prerequisites section.)</td>
<td>DevOps</td>
</tr>
<tr>
<td>Set additional options.</td>
<td>For <strong>Timeout</strong> and <strong>Queued timeout</strong>, keep the default values. For certificate, keep the default setting unless you have a custom certificate that you want to use.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Create the environment variables.</td>
<td>For each AWS Region that you want to deploy to, create environment variables by providing the S3 bucket name and the Region name (for example, us-east-1).</td>
<td>DevOps</td>
</tr>
<tr>
<td>Provide the buildspec file name, if it is not buildspec.yml.</td>
<td>Keep this field blank if the file name is the default, <code>buildspec.yml</code>. If you renamed the buildspec file, enter the name here. Make sure it matches the name of the file that is in the CodeCommit repository.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify logging.</td>
<td>To see logs for Amazon CloudWatch Events, keep the default setting. Or you can define any specific group or logger names.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

**Skip the Deploy phase**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip the deploy phase and complete the creation of the pipeline.</td>
<td>When you set up the pipeline, CodePipeline allows you to create only one stage in the Deploy phase. To deploy to multiple AWS Regions, skip this phase. After the pipeline is created, you can add multiple Deploy phase stages.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
### Deploy phase: Configure the pipeline for deployment to the first Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a stage to the Deploy phase.</td>
<td>Edit the pipeline and choose <strong>Add stage</strong> in the Deploy phase. This first stage is for the primary Region.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Provide an action name for the stage.</td>
<td>Enter a unique name that reflects the first (primary) stage and Region. For example, enter <code>primary_&lt;region&gt;_deploy</code>.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the action provider.</td>
<td>For <strong>Action provider</strong>, choose AWS CloudFormation.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Configure the Region for the first stage.</td>
<td>Choose the first (primary) Region, the same Region where CodePipeline and CodeBuild are set up. This is the primary Region where you want to deploy the stack.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the input artifact.</td>
<td>Choose <strong>BuildArtifact</strong>. This is the output of the build phase.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the action to take.</td>
<td>For <strong>Action mode</strong>, choose <strong>Create or update a stack</strong>.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Enter a name for the CloudFormation stack.</td>
<td></td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the template for the first Region.</td>
<td>Select the Region-specific package name that was packaged by CodeBuild and dumped into the S3 bucket for the first (primary) Region.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the capabilities.</td>
<td>Capabilities are required if the stack template includes IAM resources or if you create a stack directly from a template that contains macros. For this pattern, use CAPABILITY_IAM, CAPABILITY_NAMED_IAM, CAPABILITY_AUTO_EXPAND.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Deploy phase: Configure the pipeline for deployment to the second Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add the second stage to the Deploy phase.</td>
<td>To add a stage for the second Region, edit the pipeline and choose <strong>Add stage</strong> in the Deploy phase. Important: The process</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>of creating the second Region is the same as that of the first Region, except for the following values.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide an action name for the second stage.</td>
<td>Enter a unique name that reflects the second stage and the second Region.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Configure the Region for the second stage.</td>
<td>Choose the second Region where you want to deploy the stack.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the template for the second Region.</td>
<td>Select the Region-specific package name that was packaged by CodeBuild and dumped into the S3 bucket for the second Region.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

**Deploy phase: Configure the pipeline for deployment to the third Region**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add the third stage to the Deploy phase.</td>
<td>To add a stage for the third Region, edit the pipeline and choose Add stage in the Deploy phase. Important: The process of creating the second Region is the same as that of the previous two Regions, except for the following values.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Provide an action name for the third stage.</td>
<td>Enter a unique name that reflects the third stage and the third Region.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Configure the Region for the third stage.</td>
<td>Choose the third Region where you want to deploy the stack.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the template for the third Region.</td>
<td>Select the Region-specific package name that was packaged by CodeBuild and dumped into the S3 bucket for the third Region.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

**Clean up the deployment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete the AWS resources.</td>
<td>To clean up the deployment, delete the CloudFormation stacks in each Region. Then delete the CodeCommit, CodeBuild, and CodePipeline</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
Related resources

- What is AWS CodePipeline?
- AWS Serverless Application Model
- AWS CloudFormation
- AWS CloudFormation architecture structure reference for AWS CodePipeline

Attachments

attachment.zip

Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere

Created by Lucas Nation (AWS) and Freddie Wilson (AWS)

Environment: Production
Technologies: DevOps; Management & governance; Security, identity, compliance
Workload: Microsoft; Open-source

AWS services: AWS Config; AWS CloudFormation

Summary

Many organizations use AWS Config managed rules to evaluate the compliance of their Amazon Web Services (AWS) resources against common best practices. However, these rules can be time consuming to maintain and this pattern helps you leverage Troposphere, a Python library, to generate and manage AWS Config managed rules.

The pattern helps you to manage your AWS Config managed rules by using a Python script to convert a Microsoft Excel spreadsheet containing AWS managed rules into an AWS CloudFormation template. Troposphere acts as the infrastructure as code (IaC) and this means that you can update the Excel spreadsheet with managed rules, instead of using a JSON or YAML-formatted file. You then use the template to launch an AWS CloudFormation stack that creates and updates the managed rules in your AWS account.

The AWS CloudFormation template defines each AWS Config managed rule by using the Excel spreadsheet and helps you to avoid manually creating individual rules in the AWS Management
Prerequisites and limitations

**Prerequisites**

- An active AWS account.
- Familiarity with using AWS CloudFormation templates to create AWS Config managed rules. For more information about this, see [Creating AWS Config managed rules with AWS CloudFormation templates](#) in the AWS Config documentation.
- Python 3, installed and configured. For more information about this, see the [Python documentation](#).
- An existing integrated development environment (IDE) such as AWS Cloud9. For more information about this, see [What is AWS Cloud9?](#) in the AWS Cloud9 documentation.
- Identify your organizational units (OUs) in a column in the sample `excel_config_rules.xlsx` Excel spreadsheet (attached).

**Epics**

Customize and configure the AWS Config managed rules

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the sample Excel spreadsheet.</td>
<td>Download the sample <code>excel_config_rules.xlsx</code> Excel spreadsheet (attached) and label as Implemented the AWS Config managed rules that you want to use. Rules marked as Implemented will be added to the AWS CloudFormation template.</td>
<td>Developer</td>
</tr>
<tr>
<td>(Optional) Update the <code>config_rules_params.json</code> file with AWS Config rule parameters.</td>
<td>Some AWS Config managed rules require parameters and should be passed to the Python script as a JSON file by using the <code>--param-file</code> option. For example, the <code>access-keys-rotated</code> managed rule uses the following <code>maxAccessKeyAge</code> parameter:</td>
<td>Developer</td>
</tr>
</tbody>
</table>

```json
{
  "access-keys-rotated": {
    "InputParameters": {
      "maxAccessKeyAge": 90
    }
  }
}
```
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In this sample parameter, the maxAccessKeyAge is set to 90 days. The script reads the parameter file and adds any InputParameters that it finds.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(Optional) Update the config_rules_params.json file with AWS Config ComplianceResourceTypes.</strong></td>
<td>By default, the Python script retrieves the ComplianceResourceTypes from AWS defined templates. If you want to override the scope of a specific AWS Config managed rule, then you need to pass it to the Python script as a JSON file using the --param-file option. For example, the following sample code shows how the ComplianceResourceTypes for ec2-volume-inuse-check is set to the [&quot;AWS::EC2::Volume&quot;] list:</td>
<td>Developer</td>
</tr>
</tbody>
</table>

```
{
    "ec2-volume-inuse-check": {
        "Scope": {
            "ComplianceResourceTypes": [
                "AWS::EC2::Volume"
            ]
        }
    }
}
```

**Run the Python script**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the pip packages from the requirements.txt file.</td>
<td>Download the requirements.txt file (attached) and run the following command in your IDE to install the Python packages: pip3 install -r requirements.txt</td>
<td>Developer</td>
</tr>
<tr>
<td>Run the Python script.</td>
<td>1. Download the aws_config_rules.py</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### Deploy the AWS Config managed rules

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Launch the AWS CloudFormation stack. | 1. Sign in to the AWS Management Console, open the AWS CloudFormation console, and then choose Create stack.  
2. On the Specify template page, choose Upload a template file and then upload your AWS CloudFormation template.  
3. Specify a stack name and then choose Next. | Developer |

You can also add the following optional parameters:

- `--config-rule-option` – Defines the rules to choose from the Excel spreadsheet. The default is the Implemented parameter.
- `--excel-file` – The path for the Excel spreadsheet. The default is `aws_config_rules.xlsx`.
- `--param-file` – The path of the parameter JSON file. The default is `config_rules_params.json`.
- `--max-execution-frequency` – Defines how often the AWS Config managed rules are evaluated. The choices are One_Hour, Three_Hours, Six_Hours, Twelve_Hours, or TwentyFour_Hours. The default is TwentyFour_Hours.
Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit

Created by Helton Henrique Ribeiro (AWS) and Ricardo Morais (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment:</th>
<th>Technologies:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• AWS CodeCommit monorepo multi-pipeline triggers</td>
<td>PoC or pilot</td>
<td>DevOps; Infrastructure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Workload:</th>
<th>AWS services:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>AWS CodeCommit; AWS CodePipeline; AWS Lambda</td>
</tr>
</tbody>
</table>

Summary

This pattern helps you automatically detect changes to the source code of a monorepo-based application in AWS CodeCommit and then initiate a pipeline in AWS CodePipeline that runs the continuous integration and continuous delivery (CI/CD) automation for each microservice.

This approach means that each microservice in your monorepo-based application can have a dedicated CI/CD pipeline, which ensures better visibility, easier sharing of code, and improved collaboration, standardization, and discoverability.

The pattern uses AWS Cloud9 as the integrated development environment (IDE) and AWS Cloud Development Kit (AWS CDK) to define an infrastructure using two AWS CloudFormation stacks: MonoRepoStack and PipelinesStack. The MonoRepoStack stack creates the monorepo in AWS CodeCommit and the AWS Lambda function that initiates the CI/CD pipelines. The PipelinesStack stack defines your pipeline infrastructure.

Important: This pattern's workflow is a proof of concept (POC) and we recommend that you only use it in a test environment. If you want to use this pattern's approach in a production environment, see Security best practices in IAM in the IAM documentation and make the required changes to your IAM roles and AWS services.

Prerequisites and limitations

Prerequisites
• An active AWS account.
• AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
• Python 3 and pip, installed on your local machine. For more information about this, see the Python documentation.
• AWS CDK, installed and configured. For more information about this, see Getting started with the AWS CDK in the AWS CDK documentation.
• An AWS Cloud9 IDE, installed and configured. For more information about this, see Setting up AWS Cloud9 in the AWS Cloud9 documentation.
• The GitHub AWS CodeCommit monorepo multi-pipeline triggers repository, cloned to your local machine.
• An existing directory containing application code that you want to build and deploy with CodePipeline.
• Familiarity and experience with DevOps best practices on the AWS Cloud. To increase your familiarity with DevOps, you can use the pattern Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9 on the AWS Prescriptive Guidance website.

Limitations

• This pattern's workflow is a proof of concept (POC) and we recommend that you only use it in a test environment. If you want to use this pattern's approach in a production environment, see Security best practices in IAM in the IAM documentation and make the required changes to your IAM roles and AWS services.

Architecture

The following diagram shows how to use AWS CDK to define an infrastructure with two AWS CloudFormation stacks: MonoRepoStack and PipelinesStack.
The diagram shows the following workflow:

1. The bootstrap process uses AWS CDK to create the MonoRepoStack and PipelinesStack AWS CloudFormation stacks.
2. The MonoRepoStack stack creates the CodeCommit repository for your application and the monorepo-event-handler Lambda function that is initiated after each commit.
3. The PipelinesStack stack creates the pipelines in CodePipeline that are initiated by the Lambda function. Each microservice must have a defined infrastructure pipeline.
4. The pipeline for microservice-n is initiated by the Lambda function and starts its isolated CI/CD stages that are based on the source code in CodeCommit.
5. The pipeline for microservice-1 is initiated by the Lambda function and starts its isolated CI/CD stages that are based on the source code in CodeCommit.

The following diagram shows the deployment of the MonoRepoStack and PipelinesStack AWS CloudFormation stacks in an account.
1. A user changes code in one of the application’s microservices.
2. The user pushes the changes from a local repository to a CodeCommit repository.
3. The push activity initiates the Lambda function that receives all pushes to the CodeCommit repository.
4. The Lambda function reads a parameter in AWS Systems Manager Parameter Store to retrieve the most recent commit ID. The parameter has the /MonoRepoTrigger/{repository}/{branch_name}/LastCommit naming format. If the parameter is not found, the Lambda function reads the last commit ID from the CodeCommit repository and saves the returned value in Parameter Store.
5. After identifying the commit ID and the changed files, the Lambda function identifies the pipelines for each microservice directory and initiates the required CodePipeline pipeline.

**Tools**

- **AWS CDK** – AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- **Python** – Python is a programming language that lets you work quickly and integrate systems more effectively.

**Code**

The source code and templates for this pattern are available in the Github AWS CodeCommit monorepo multi-pipeline triggers repository.
## Epics

### Set up the environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a virtual Python environment.     | In your AWS Cloud9 IDE, create a virtual Python environment and install the required dependencies by running the following command:  
  make install                                                                 | Developer       |
| Bootstrap the account and AWS Region for the AWS CDK. | Bootstrap the required AWS account and Region by running the following command:  
  make bootstrap account-id=<your-AWS-account-ID> region=<required-region> | Developer       |

### Add a new pipeline for a microservice

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add your sample code to your application directory.</td>
<td>Add the directory containing your sample application code to the monorepo-sample directory in the cloned AWS CodeCommit monorepo multi-pipeline triggers repository.</td>
<td>Developer</td>
</tr>
<tr>
<td>Edit the monorepo-main.json file.</td>
<td>Add the directory name of your application's code and the pipeline's name to the monorepo-main.json file in the cloned AWS CodeCommit monorepo multi-pipeline triggers repository.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the pipeline.</td>
<td>In the Pipelines directory for the AWS CodeCommit monorepo multi-pipeline triggers repository, add the pipeline class for your application. The directory contains two sample files, pipeline_hotsite.py and pipeline_demo.py, that each have three stages: source, build, and deploy.</td>
<td>Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Edit the monorepo_config.py file.</td>
<td>You can copy one of the files and makes changes to it according to your application's requirements.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>In <code>service_map</code>, add the directory name for your application and the class that you created for the pipeline. For example, the following code shows a pipeline definition in the <code>Pipelines</code> directory that uses a file named <code>pipeline_mysample.py</code> with a <code>MySamplePipeline</code> class:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>... # Pipeline definition imports from pipelines.pipeline_demo import DemoPipeline from pipelines.pipeline_hotsite import HotsitePipeline from pipelines.pipeline_mysample import MySamplePipeline $$$ Add your pipeline configuration here service_map: Dict[str, ServicePipeline] = { # folder-name -&gt; pipeline-class 'demo': DemoPipeline(), 'hotsite': HotsitePipeline(), 'mysample': MySamplePipeline() }</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy the MonoRepoStack stack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deploy the AWS CloudFormation stack.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Deploy the MonoRepoStack AWS CloudFormation stack with default parameter values in the root directory of the cloned AWS CodeCommit monorepo multi-pipeline triggers repository by running the make deploy-core command.</td>
<td></td>
</tr>
</tbody>
</table>
## Validate the CodeCommit repository.

Validate that your resources were created by running the `aws codecommit get-repository --repository-name <repo_name>` command.

**Important:** Because the AWS CloudFormation stack creates the CodeCommit repository where the monorepo is stored, don’t run the `cdk destroy MonoRepoStack` command if you have started to push modifications into it.

## Validate the AWS CloudFormation stack results.

Validate that the MonoRepoStack AWS CloudFormation stack is correctly created and configured by running the following command:

```bash
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE --query 'StackSummaries[?StackName == `MonoRepoStack`]'
```

## Deploy the PipelinesStack stack

Deploy the AWS CloudFormation stack.

The PipelinesStack AWS CloudFormation stack must be deployed after you deploy the MonoRepoStack stack. The stack increases in size when new microservices are added to the monorepo’s code base and is redeployed when a new microservice is onboarded.
## Task | Description | Skills required
--- | --- | ---
**Deploy the PipelinesStack stack**<br>by running the `make deploy-pipelines` command.  
**Note:** You can also deploy simultaneously deploy both pipelines by running the `make deploy monorepo-name=<repo_name>` command.  
The following sample output shows how the PipelinesStacks deployment prints the URLs for the microservices at the end of the implementation:  

| Outputs: |  |
| PipelinesStack.demourl | = .cloudfront.net |
| PipelinesStack.hotsiteurl | = .cloudfront.net |

**Validate the AWS CloudFormation stack results.**
Validate that the PipelinesStacks AWS CloudFormation stack is correctly created and configured by running the following command:

```
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE UPDATE_COMPLETE --query 'StackSummaries[?StackName == `PipelinesStack`]'
```

**Clean up resources**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete your AWS CloudFormation stacks.</strong>&lt;br&gt;Run the <code>make destroy</code> command.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>
| **Delete the S3 buckets for your pipelines.**<br>1. Sign in to the AWS Management Console and open the Amazon Simple Storage Service (Amazon S3) console.  
2. Delete the S3 buckets associated with your pipelines and that use the following | Developer |
Integration patterns

Related resources

- Continuous integration and delivery (CI/CD) using CDK Pipelines (AWS CDK documentation)
- aws-cdk/pipelines module (AWS CDK API reference)

Integrate a Bitbucket repository with AWS Amplify using AWS CloudFormation

Created by Alwin Abraham (AWS)

Environment: Production  Technologies: DevOps  AWS services: AWS Amplify; AWS CloudFormation

Summary

AWS Amplify helps you to quickly deploy and test static websites without having to set up the infrastructure that is typically required. You can deploy this pattern's approach if your organization wants to use Bitbucket for source control, whether to migrate existing application code or build a new application. By using AWS CloudFormation to automatically set up Amplify, you provide visibility into the configurations that you use.

This pattern describes how to create a front-end continuous integration and continuous deployment (CI/CD) pipeline and deployment environment by using AWS CloudFormation to integrate a Bitbucket repository with AWS Amplify. The pattern's approach means that you can build an Amplify front-end pipeline for repeatable deployments.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- An active Bitbucket account with administrator access
- Access to a terminal that uses cURL or the Postman application
- Familiarity with Amplify
- Familiarity with AWS CloudFormation
- Familiarity with YAML-formatted files
Architecture

Technology stack

- Amplify
- AWS CloudFormation
- Bitbucket

Tools

- **AWS Amplify** – Amplify helps developers to develop and deploy cloud-powered mobile and web apps.
- **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.
- **Bitbucket** – Bitbucket is a Git repository management solution designed for professional teams. It gives you a central place to manage Git repositories, collaborate on your source code, and guide you through the development flow.

Code

The `bitbucket-amplify.yml` file (attached) contains the AWS CloudFormation template for this pattern.

Epics

Configure the Bitbucket repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Optional) Create a Bitbucket repository.</td>
<td>1. Sign in to your Bitbucket account and create a new repository. For more information about this, see Create a Git repository in the Bitbucket documentation.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
### Task | Description | Skills required
--- | --- | ---
2. Record the workspace’s name. | **Note:** You can also use an existing Bitbucket repository. | DevOps engineer

### Open the workspace settings.
1. Open the workspace and choose the **Repository** tab.
2. Choose the repository that you want to integrate with Amplify.
3. Choose the name of the workspace that is above the repository’s name.
4. On the sidebar, choose **Settings**.

### Create an OAuth consumer.
1. In the **Apps and Features** section, choose **OAuth consumers**, and then choose **Add consumer**.
2. Enter a name for your consumer, for example, **Amplify Integration**.
3. Enter a callback URL. Although this field is a required input, it’s not used to complete the integration so the value could be `http://localhost:3000`
4. Check the box for **This is a private consumer**.
5. Choose the following permissions:
   - **Project** – **Read**
   - **Repositories** – **Admin**
   - **Pull requests** – **Read**
   - **Webhooks** – **Read and Write**
6. Leave the default choices for all the other fields and choose **Submit**.
7. Record the key and secret that are generated.
### Obtain OAuth access token.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Obtain OAuth access token. | 1. Open a terminal window and run the following command: 
   ```
curl -X POST -u "KEY:SECRET" https://bitbucket.org/site/oauth2/access_token -d grant_type=client_credentials
   ```
   **Important:** Replace_KEY_ and_SECRET_ with the key and secret that you recorded earlier. 
   2. Record the access token without using the quotation marks. The token is only valid for a limited time and the default time is two hours. You must run the AWS CloudFormation template in this timeframe. | DevOps engineer |

### Create and deploy the AWS CloudFormation stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the AWS CloudFormation template.</td>
<td>Download the bitbucket-amplify.yml AWS CloudFormation template (attached). This template creates the CI/CD pipeline in Amplify, in addition to the Amplify project and branch.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create and deploy the AWS CloudFormation stack. | 1. Sign in to the AWS Management Console in the AWS Region that you want to deploy in and open the AWS CloudFormation console. 
   2. Choose **Create Stack (with new resources)** and then choose **Upload a Template File**. 
   3. Upload the bitbucket-amplify.yml file. 
   4. Choose **Next**, enter a stack name, and then enter the following parameters: 
     - **Access token:** Paste the OAuth access token that you created earlier. | DevOps engineer |
Test the CI/CD pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the code to the branch in your repository. | 1. Clone your Bitbucket repository by running the following command: git clone https://bitbucket.org/ <WORKSPACE_NAME>/ <REPO_NAME>  
2. Check out the branch name that was used when running the AWS CloudFormation script. To create and check out a new branch, run the git checkout -b <BRANCH_NAME> command. To check out an existing branch, run the git checkout <BRANCH_NAME> command  
3. Commit the code into the branch and push it to the remote branch by running the git commit and git push commands. | App developer       |
### Related resources

**Authentication methods** *(Atlassian documentation)*

**Attachments**

attachment.zip

---

**Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function**

*Created by Richard Milner-Watts (AWS)*

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: Production</th>
<th>Technologies: DevOps; Management &amp; governance; Operations; Serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Cross-Account CodeBuild Proxy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AWS services:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CodeBuild; AWS Lambda; AWS Step Functions; AWS X-Ray; AWS CloudFormation</td>
<td></td>
</tr>
</tbody>
</table>

---

**Summary**

This pattern demonstrates how to asynchronously launch an Amazon Web Services (AWS) CodeBuild project across AWS accounts by using AWS Step Functions and an AWS Lambda proxy function. You can use the pattern’s sample Step Functions state machine to test the success of your CodeBuild project.

CodeBuild helps you launch operational tasks using the AWS Command Line Interface (AWS CLI) from a fully-managed Amazon Linux 2 runtime environment. You can change the behavior of your CodeBuild project at runtime by overriding environment variables. You can also use AWS CodePipeline to launch operational tasks, but you can’t override environment variables with CodePipeline. Additionally, you can use CodeBuild to manage workflows. For more information, see Service Catalog Tools on the AWS Workshop website and Schedule jobs in Amazon RDS for PostgreSQL using AWS CodeBuild and Amazon EventBridge on the AWS Database Blog.

---

**Prerequisites and limitations**

**Prerequisites**
Two active AWS accounts: a source account for invoking a Lambda proxy function with Step Functions and a target account for building a remote CodeBuild sample project

Limitations

- This pattern cannot be used to copy artifacts between accounts.

Architecture

The following diagram shows the architecture that this pattern builds.

The diagram shows the following workflow:

1. The Step Functions state machine invokes the Lambda proxy function (codebuild-proxy-lambda).
2. The Lambda proxy function uses AWS Security Token Service (AWS STS) to assume an IAM proxy role (codebuild-proxy-role) and use an IAM policy (codebuild-proxy-policy) in the target account.
3. The Lambda function launches the CodeBuild project and returns the CodeBuild job ID.
4. The Step Functions state machine loops and polls the CodeBuild job until receiving a success or failure status.

Technology stack

- AWS CloudFormation
- CodeBuild
- IAM
- Lambda
- Step Functions
- X-Ray
Tools

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **AWS CloudFormation Designer** – AWS CloudFormation Designer provides an integrated JSON and YAML editor for viewing and editing template details. For example, you can use the integrated editor to define the properties of a resource or to change a template parameter.

- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS Step Functions** – AWS Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications. Through the Step Functions graphical console, you see your application's workflow as a series of event-driven steps.

- **AWS X-Ray** – AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture.

- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

Code

The code for this pattern is available in the GitHub **Cross Account CodeBuild Proxy** repository.

Epics

Create the Lambda proxy function and associated IAM role in the source account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record the AWS account IDs.</td>
<td>AWS account IDs are required to set up access across accounts. Record the AWS account ID for your source and target accounts. For more information, see <a href="#">Finding your AWS account ID in the IAM documentation</a>.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Download the AWS CloudFormation templates.</td>
<td>1. Download the <em>sample_target_codebuild_template.yaml</em> AWS CloudFormation template from the <a href="#">GitHub repository</a> for this pattern. 2. Download the <em>codebuild_lambda_proxy_template.yaml</em> AWS CloudFormation template from the <a href="#">GitHub repository</a> for this pattern.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Note:</strong> In the AWS CloudFormation templates, <code>&lt;SourceAccountId&gt;</code> is the AWS account ID for the source account, and <code>&lt;TargetAccountId&gt;</code> is the AWS account ID for the target account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Create and deploy the AWS CloudFormation stack. | 1. Sign in to the AWS Management Console for your source account, open the AWS CloudFormation console, and then choose **Stacks**.  
2. Choose **Create Stack** and then choose **With new resources (standard)**.  
3. For **Template source**, choose **Upload a template file**.  
4. For **Upload a template file**, choose **file**, and then choose your downloaded `codebuild_lambda_proxy_template.yaml` file. Choose **Next**.  
5. For **Stack name**, enter a name for the stack (for example, `codebuild-lambda-proxy`).  
6. Replace the `crossAccountTargetRoleArn` parameter with your `<TargetAccountId>` (for example, `<arn:aws:iam::123456789012:role/proxy-lambda-codebuild-role>`). **Note**: You aren’t required to update the default value for the `targetCodeBuildProject` parameter.  
7. Choose **Next**, accept the default stack creation options, and then choose **Next**.  
8. Choose the **I acknowledge that AWS CloudFormation might create IAM resources with custom names** check box, and then choose **Create stack**.  

**Note:** You must create the AWS CloudFormation stack for the proxy Lambda function before creating any other resources in the source account. For more information, see [Creating a role to delegate permissions to an IAM user](https://docs.aws.amazon.com/iam/latest/UserGuide/what-is-user.html) in the IAM documentation. | AWS DevOps |
### Confirm the creation of the proxy function and state machine.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Confirm the creation of the proxy function and state machine.       | 1. Wait for the AWS CloudFormation stack to reach **CREATE_COMPLETE** status. This should take less than one minute.  
2. Open the AWS Lambda console, choose **Functions**, and then find the lambda-proxy-ProxyLambda-<GUID> function.  
3. Open the AWS Step Functions console, choose **state machines**, and then find the sample-crossaccount-codebuild-state-machine state machine.                                                                                                                                   | AWS DevOps       |

### Create an IAM role in the target account and launch a sample CodeBuild project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create and deploy the AWS CloudFormation stack.                      | 1. Sign in to the AWS Management Console for your target account, open the AWS CloudFormation console, and then choose **Stacks**.  
2. Choose **Create Stack** and then choose **With new resources (standard)**.  
3. For **Template source**, choose **Upload a template file**.  
4. For **Upload a template file**, choose **Choose file**, and then choose the sample_target_codebuild_template.yaml file. Choose **Next**.  
5. For **Stack name**, enter a name for the stack (for example: sample-codebuild-stack).  
6. Replace the **crossAccountSourceRoleArn** parameter with your <SourceAccountId> (for example, <arn:aws:iam::123456789012:role/codebuild-proxy-lambda-role>).  
7. Choose **Next**, accept the default stack creation options, and then choose **Next**.                                                                                                                                  | AWS DevOps       |
8. Choose the I acknowledge that AWS CloudFormation might create IAM resources with custom names check box, and then choose Create stack.

Verify the creation of the sample CodeBuild project.

1. Wait for the AWS CloudFormation stack to reach CREATE_COMPLETE status. This should take less than one minute.
2. Open the AWS CodeBuild console and then find the sample-codebuild-project project.

Test the cross-account Lambda proxy function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Launch the state machine. | 1. Sign in to the AWS Management Console for your source account, open the AWS Step Functions console, and then choose State machines.  
2. Choose the sample-crossaccount-codebuild-state-machine state machine and then choose Start execution.  
3. In the Input editor, enter the following key-value pairs: { "SampleValue1": "Value1", "SampleValue2": "Value2" }. Note: The key-value pairs are passed as environment variables from the function in the source account to the CodeBuild project in the target account.  
5. On the Details tab of the state machine page, check if Execution Status is set to Succeeded. This confirms that your state machine is running. Note: It can take around 30 seconds for the state machine to reach Succeeded status. | AWS DevOps       |
## AWS Prescriptive Guidance Patterns

### Monitor Amazon ECR repositories for wildcard permissions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6. To see the output and input of a step in the state machine, expand that step in the <strong>Execution event history</strong> section. For example, expand the <strong>Lambda - CodeBuild Proxy – Start</strong> step. The output includes details on the overridden environment variables, the original payload, and the CodeBuild job ID.</td>
<td></td>
</tr>
</tbody>
</table>

| Validate the environment variables. | 1. Sign in to the AWS Management Console for your target account.  
2. Open the [AWS CodeBuild console](https://aws.amazon.com/codebuild/), expand **Build**, and then choose **Build projects**.  
3. Choose the `sample-codebuild-project` project and then choose **View details**.  
4. On the **Build history** tab, choose the most recent build of the project and then choose **View logs**.  
5. In the log output, verify that the environment variables printed to STDOUT match the environment variables from the Step Functions sample state machine. | AWS DevOps |

---

## Monitor Amazon ECR repositories for wildcard permissions using AWS CloudFormation and AWS Config

*Created by Vikrant Telkar (AWS), Sajid Momin (AWS), and Wassim Benhallam (AWS)*

| Environment: Production | Technologies: DevOps; Containers & microservices | AWS services: AWS CloudFormation; AWS Config; Amazon ECR; Amazon SNS; AWS Lambda |
Summary

On the Amazon Web Services (AWS) Cloud, Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that supports private repositories with resource-based permissions using AWS Identity and Access Management (IAM).

IAM supports the "*" wildcard in both the resource and action attributes, which makes it easier to automatically choose multiple matching items. In your testing environment, you can allow all authenticated AWS users to access an Amazon ECR repository by using the `ecr:*` wildcard permission in a principal element for your repository policy statement. The `ecr:*` wildcard permission can be useful when developing and testing in development accounts that can't access your production data.

However, you must make sure that the `ecr:*` wildcard permission is not used in your production environments because it can cause serious security vulnerabilities. This pattern's approach helps you to identify Amazon ECR repositories that contain the `ecr:*` wildcard permission in repository policy statements. The pattern provides steps and an AWS CloudFormation template to create a custom rule in AWS Config. An AWS Lambda function then monitors your Amazon ECR repository policy statements for `ecr:*` wildcard permissions. If it finds non-compliant repository policy statements, Lambda notifies AWS Config to send an event to Amazon EventBridge and EventBridge then initiates an Amazon Simple Notification Service (Amazon SNS) topic. The SNS topic notifies you by email about the non-compliant repository policy statements.

Prerequisites and limitations

**Prerequisites**

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
- An existing Amazon ECR repository with an attached policy statement, installed and configured in your testing environment. For more information about this, see Creating a private repository and Setting a repository policy statement in the Amazon ECR documentation.
- AWS Config, configured in your preferred AWS Region. For more information about this, see Getting started with AWS Config in the AWS Config documentation.
- The `aws-config-cloudformation.template` file (attached), downloaded to your local machine.

**Limitations**

- This pattern's solution is Regional and your resources must be created in the same Region.

Architecture

The following diagram shows how AWS Config evaluates Amazon ECR repository policy statements.
The diagram shows the following workflow:

1. AWS Config initiates a custom rule.
2. The custom rule invokes a Lambda function to evaluate the compliance of the Amazon ECR repository policy statements. The Lambda function then identifies non-compliant repository policy statements.
3. The Lambda function sends the non-compliance status to AWS Config.
4. AWS Config sends an event to EventBridge.
5. EventBridge publishes the non-compliance notifications to an SNS topic.
6. Amazon SNS sends an email alert to you or an authorized user.

**Automation and scale**

This pattern’s solution can monitor any number of Amazon ECR repository policy statements, but all resources that you want to evaluate must be created in the same Region.

**Tools**

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- **AWS Config** – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.
- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private repositories with resource-based permissions using IAM.
Amazon EventBridge – Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your applications, software as a service (SaaS) applications, and AWS services to targets such as AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other accounts.

AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

The code for this pattern is available in the `aws-config-cloudformation.template` file (attached).

Epics

Create the AWS CloudFormation stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS CloudFormation stack.</td>
<td>Create an AWS CloudFormation stack by running the following command in AWS CLI:</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

```bash
$ aws cloudformation create-stack --stack-name=AWSConfigECR --template-body file://aws-config-cloudformation.template --parameters ParameterKey=<email>,ParameterValue=<myemail@example.com> --capabilities CAPABILITY_NAMED_IAM
```

Test the AWS Config custom rule

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Test the AWS Config custom rule. | 1. Sign in to the AWS Management Console, open the AWS Config console, and then choose Resources.  
2. On the Resource inventory page, you can filter by resource category, resource type, and compliance status.  
3. An Amazon ECR repository that contains `ecr:*` is NON- |

AWS DevOps

510
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLIANT? and an Amazon ECR repository that doesn't contain ecr:* is COMPLIANT. 4. The email address subscribed to the SNS topic receives notifications if an Amazon ECR repository contains non-compliant policy statements.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Attachments**

attachment.zip

**Perform a canary-based deployment using the blue/green strategy and AWS Lambda**

*Created by Raju Banerjee (AWS)*

<table>
<thead>
<tr>
<th>Created by: AWS</th>
<th>Environment: PoC or pilot</th>
<th>Technologies: DevOps</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services: AWS CodeDeploy; AWS Lambda</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern guides you through the steps to set up a canary-based blue/green deployment using AWS CodeDeploy. CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon Elastic Compute Cloud (Amazon EC2), AWS Lambda, and your on-premises servers. Manually deploying applications to servers can take a long time and often results in errors that lead to downtime.

The purpose of a canary deployment is to reduce the risk of deploying a new version that impacts the workload. The method covered by this pattern incrementally deploys the new version, making it visible to new users in a slow fashion. As you gain confidence in the deployment, you can deploy it to replace the current version in its entirety.

By following this pattern, you can test both old and new application versions. You can use AWS cloud-native tools, with no need to maintain separate tools for deployment and no extra Amazon EC2 machine cost. The pattern supports AWS Identity and Access Management (IAM) authentication and audit features that use AWS CloudTrail to manage user activities.

**Prerequisites and limitations**

**Prerequisites**

- An application, with code defined in the AppSpec file
- An active AWS account
• A virtual private cloud (VPC)
• An Amazon EC2 Auto Scaling group
• A key pair (optional)
• An IAM role
• An Amazon Simple Notification Service (Amazon SNS) topic

Limitations

• The old version of code will not provide the automatic scaling feature.
• Cleaning up the old environment will require a separate Lambda function. Or, wait 48 hours, and the environment will be cleaned by default.

Architecture

Target technology stack

• Amazon CloudWatch Events
• IAM
• Amazon SNS
• AWS CodeBuild
• AWS CodeDeploy
• Amazon EC2
• Application Load Balancer
• AWS Lambda

Target architecture
Automation and scale

AWS CloudFormation can be used to automate the infrastructure creation.

Cleanup of the pilot environment can be automated by using a Python (Boto) script and the tags of the environment.

Tools

Tools

- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeDeploy** – AWS CodeDeploy is a deployment service that automates application deployments to EC2 instances, on-premises instances, serverless Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services. You do not need to make changes to your existing code before you can use CodeDeploy.
- **AWS Lambda** – AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- **AWS SNS** – Amazon SNS is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. Subscribers, including Lambda functions, consume or receive the message or notification over one of the supported protocols when they are subscribed to the topic.
- **A launch template** – A launch template specifies instance configuration information. Included are the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instances.

Code

Python code for the Lambda function:

```python
import boto3
codeploy = boto3.client('codedeploy')
autoscaling = boto3.client('autoscaling')

def lambda_handler(event, context):
    # Define the code-deploy related parameters here or pass it using lambda parameters.
    codedeployappname = 'pattern-test'
    codedeploymentgroup = 'pattern-testing'
    albtargetgrouparn = 'arn:aws:elasticloadbalancing:us-east-1:************:targetgroup/green-target-group/0614b53e*******'
    
    response = codeploy.list_deployments(
        applicationName=codedeployappname,
        deploymentGroupName=codedeploymentgroup,
        includeOnlyStatuses=['Succeeded'],
    )

    # Get the latest deployment event and fetch the id and ignore any autoscale events.
    for i in range(len(response['deployments'])):
        response3 = codeploy.get_deployment(deploymentId=response['deployments'][i])
        type01 = response3['deploymentInfo']['creator']
        if type01 == 'user':
            deploymentinfo = response['deployments'][i]
            response4 = codeploy.get_deployment(deploymentId=deploymentinfo['deploymentId'])
```

513
response5 = response4['deploymentInfo']['targetInstances']['autoScalingGroups'][0]
autosg.append(response5)
print(autosg[1])
response = clientasg.attach_load_balancer_target_groups(
    AutoScalingGroupName=autosg[1],
    TargetGroupARNs=[
        albtargetgrouparn,
    ]
)

Epics

Configure the Auto Scaling group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AMI.</td>
<td>After the deployment of the first version is complete, create an AMI of the EC2 instance.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create a launch template.</td>
<td>Create a template for setting up the Auto Scaling group for application. The launch template uses the AMI ID of the AMI that you created in the first task. For more information about the commands and steps described in this and other epics, see the “Related resources” section at the end of this pattern.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Set up automatic scaling.</td>
<td>Use the launch template to create the Auto Scaling group.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create a load balancer for the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create target groups for the Application Load Balancer.</td>
<td>Create two target groups: a Blue target group and a Green target group.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create the Application Load Balancer.</td>
<td>To divide traffic equally between the Blue and Green environments, add a forward action to the listener. You can also enable group level stickiness to maintain session. If a target group is sticky, requests routed to it remain on that target group for the duration of the session. The duration of stickiness can be between 1 second and 7 days.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
### Configure CodeDeploy for the deployment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a service role.</td>
<td>If you are configuring CodeDeploy for the first time, you must create an IAM role so that CodeDeploy can assume the role and perform operations.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create an application</td>
<td>Choose &quot;Deploy&quot; to start creating the application. Provide a name for the application, and choose &quot;EC2/On-premises.&quot;</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create the deployment group.</td>
<td>On the application page, create the deployment group, using the same name as your application.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Configure the deployment group.</td>
<td>Under &quot;Service role&quot; choose the service role that you created in the first task. Under &quot;Deployment type,&quot; choose “Blue/green.” Under &quot;Environment configuration,&quot; choose “Automatically copy Amazon EC2 Auto Scaling group.” Under &quot;Deployment settings,&quot; choose &quot;Reroute traffic immediately.&quot; If your testing time takes more than two days, choose &quot;Keep the original instances in the deployment group running.&quot;</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Specify the target group for the Application Load Balancer.</td>
<td>Under &quot;Load balancer,&quot; choose the Blue target group that you created.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Configure the deployment trigger.</td>
<td>Expand the &quot;Advanced - optional&quot; section, and choose &quot;Create trigger.&quot; Enter a name. In the &quot;Events&quot; field, choose &quot;Deployment succeeds,&quot; and then choose the SNS topic.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

### Configure the Lambda function for the automatic scaling group update

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new IAM role.</td>
<td>Create an IAM role that has permissions to update the Auto Scaling group and write logs to CloudWatch.</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
Create the Lambda function.
Under "Runtime," choose the latest version of Python.
Cloud Administrator

Configure the Lambda function.
Update the function by using the Python code in the "Code" section of this pattern. For the trigger source, use the same SNS topic that you used when configuring the CodeDeploy deployment group.
Cloud Administrator

Clean up the environment after the pilot is complete

Clean up the automatic scaling environment.
Copy the Deployment identifier from the CodeDeploy console and use it to search for the Auto Scaling group that is associated with the identifier. Delete the scaling group to clean up all the EC2 instances that are associated with this setup.
Cloud Administrator

Delete the load balancer and the target groups.
Cloud Administrator

Delete the Lambda function.
Delete the Lambda function unless you plan to use it later for another purpose.
Cloud Administrator

Related resources

Create a load balancer
- Forward actions

Configure the Auto Scaling group
- Creating a launch template for an Auto Scaling group
- Creating an Auto Scaling Group using a launch template

Configure CodeDeploy
- Step 3: Create a service role for CodeDeploy

Additional references
- AWS CodeDeploy
- AppSpec file example
Perform custom actions from AWS CodeCommit events

Created by Abdullahi Olaoye (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>N/A</th>
<th>Source: AWS CodeCommit Repository</th>
<th>Target: Integration with other AWS Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Management &amp; governance; DevOps</td>
</tr>
</tbody>
</table>

| AWS services: | AWS CodeCommit; Amazon CloudWatch |

Summary

When you use an AWS CodeCommit repository to store code, you might want to monitor the repository and initiate a workflow of actions when specific events occur. For example, you might want to send an email notification when a user comments on a line of code in a commit, or initiate an AWS Lambda function to perform security scans on repository contents after a commit. This pattern outlines the steps for configuring a CodeCommit repository for custom actions. The pattern uses AWS CodeCommit notification rules to capture the events of interest, and then sends these events to a configured target.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Familiarity with Git commands
AWS CodeCommit – AWS CodeCommit is a fully-managed source control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. CodeCommit eliminates the need to operate your own source control system or worry about scaling its infrastructure.

Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud. Amazon SNS provides topics (communication channels) for high-throughput, push-based, many-to-many messaging. Using Amazon SNS topics, publishers can distribute messages to a large number of subscribers for parallel processing, including Amazon Simple Queue Service (Amazon SQS) queues, AWS Lambda functions, and HTTP/S webhooks. You can also use Amazon SNS to send notifications to end users using mobile push, SMS, and email.

Epics

Set up AWS CodeCommit

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete the initial setup required for CodeCommit.</td>
<td>This task involves creating and configuring an AWS Identity and Access Management (IAM) user for CodeCommit and configuring your local computer for access. You should also install the AWS Command Line Interface (AWS CLI) to manage CodeCommit. For instructions and more information about this task and other epics, see the links in the &quot;Related resources&quot; section.</td>
<td>DevOps Engineer</td>
</tr>
</tbody>
</table>
### Task | Description | Skills required
--- | --- | ---
Create a CodeCommit repository. | Use the CodeCommit console or the AWS CLI to create a CodeCommit repository. | DevOps Engineer
Push content to the CodeCommit repository. | After you create a repository, you can add content to it by using Git commands. You can migrate the contents of an existing Git repository or local, unversioned content from your computer. | DevOps Engineer

### Set up Amazon SNS

### Task | Description | Skills required
--- | --- | ---
Create an SNS topic. | This SNS topic will receive the events from CodeCommit. | Cloud Engineer/Architect
Create a resource for a custom action. | For the custom action to be performed, you must create the corresponding resource. For example, if your custom action is to run Lambda code and send messages to an SQS queue, you must create the Lambda function and the SQS queue. Actions such as email and SMS notifications do not require resources. | Cloud Engineer/Architect
Subscribe the custom action resource to the SNS topic. | Depending on the custom action, you create a subscription for the appropriate protocol: for example, you subscribe an email address for email notification, a Lambda function to run custom code, or an SQS queue to send events to Amazon SQS. For subscription protocols like email and SMS, you need to confirm the subscription from the link that is sent to the email or telephone number, respectively. | Cloud Engineer/Architect

### Configure notification rules

### Task | Description | Skills required
--- | --- | ---
Create the notification rule for the CodeCommit repository. | When you create the notification rule, you select the Git events that should initiate the | DevOps Engineer
## Task Description

**Test custom actions.**

Perform one of the events that was configured to initiate the notification. For example, create a pull request if you selected that event as a trigger. You should see your custom action being performed. For example, if you subscribed an email address to the SNS topic, you should receive an email notification.

**Skills required:** DevOps Engineer

---

## Related resources

### Set up AWS CodeCommit
- Getting started with Git and AWS CodeCommit
- Migrate to AWS CodeCommit

### Set up Amazon SNS
- Tutorial: Creating an SNS Topic
- Tutorial: Subscribing an endpoint to an Amazon SNS topic
- Tutorial: Subscribing an Amazon SQS queue to an Amazon SNS topic

### Configure notification rules
- Create a notification rule

### More resources
- AWS CodeCommit Documentation
- Amazon SNS Documentation
- Git Documentation

---

## Publish Amazon CloudWatch metrics to a CSV file

*Created by Abdullahi Olaoye (AWS)*

**Environment:** PoC or pilot  
**Technologies:** DevOps  
**AWS services:** Amazon CloudWatch
Summary

This pattern uses a Python script to retrieve Amazon CloudWatch metrics and to convert the metrics information into a comma-separated values (CSV) file for improved readability. The script takes the AWS service whose metrics should be retrieved as a required argument. The AWS Region and AWS credential profile can be passed as optional arguments. If those optional arguments are not passed, the script uses the configured default Region and the default AWS credential that are configured for the workstation where the script is run. After the script runs, it generates and stores a CSV file in the same directory.

Prerequisites and limitations

Prerequisites

- Python 3.x
- AWS Command Line Interface (AWS CLI)

Limitations

The script currently supports the following AWS services:

- AWS Lambda
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Relational Database Service (Amazon RDS)
  - However, the script doesn't support Amazon Aurora
- Application Load Balancer
- Network Load Balancer
- Amazon API Gateway

Tools

- Amazon CloudWatch – Amazon CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers. CloudWatch provides data and actionable insights to help you monitor your applications, respond to systemwide performance changes, optimize resource utilization, and get a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, and provides a unified view of AWS resources, applications, and services that run on AWS and on-premises servers.

Epics

Install and configure the prerequisites

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the prerequisites.</td>
<td>Run the following command: $ pip3 install -r requirements.txt</td>
<td>Developer</td>
</tr>
<tr>
<td>Configure the AWS CLI.</td>
<td>Run the following command:</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### Configure the Python script

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the script.</td>
<td>To change the default configuration of the script, open <code>metrics.yaml</code>.</td>
<td>Developer</td>
</tr>
</tbody>
</table>
| Set the period for the script.            | This is the time period to fetch. The default period is 5 minutes (300 seconds). You can change the period, but note the following limitations:  
  - If the hours value that you specify is between 3 hours and 15 days ago, use a multiple of 60 seconds (1 minute) for the period.  
  - If the hours value that you specify is between 15 hours and 63 days ago, use a multiple of 300 seconds (5 minutes) for the period.  
  - If the hours value that you specify is greater than 63 days ago, use a multiple of 3,600 seconds (1 hour) for the period.  
  Otherwise, the API operation won't return any data points. | Developer       |
| Set the hours for the script.             | This value specifies how many hours of metrics you want to fetch. The default is 1 hour. To retrieve multiple days of metrics, provide the value in hours. For example, for 2 days, specify 48. | Developer       |
| Change statistics values for the script.  | (Optional) The global statistics value is `Average`, which is used when fetching metrics that do not have a specific statistics value assigned. The script supports the statistics values `Maximum`, `SampleCount`, and `Sum`. | Developer       |
Run the Python script

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the script.</td>
<td>Use the following command:</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>$ python cwreport.py &lt;service&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To see a list of service values and the optional region and profile parameters, run the following command:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$ python cwreport.py -h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about the optional parameters, see the Additional information section.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

- Configuring the AWS CLI
- Using Amazon CloudWatch metrics
- Amazon CloudWatch documentation

Additional information

Script usage

$ python cwreport.py -h

Example syntax

python cwreport.py <service> [--region=Optional Region] [--profile=Optional credential profile]

Parameters

- **service (required)** - The service you want to run the script against. The script currently supports these services: AWS Lambda, Amazon EC2, Amazon RDS, Application Load Balancer, Network Load Balancer, and API Gateway.
- **region (optional)** - The AWS Region to fetch metrics from. The default Region is ap-southeast-1.
- **profile (optional)** - The AWS CLI named profile to use. If specified, the default configured credential isn't used.

Examples

- Using default Region ap-southeast-1 and default configured credentials to fetch Amazon EC2 metrics: $ python cwreport.py ec2
• Specifying a Region and fetching Amazon API Gateway metrics: $ python cwreport.py apigateway --region us-east-1
• Specifying an AWS profile: $ python cwreport.py ec2 --profile testprofile
• Specifying both Region and profile: $ python cwreport.py ec2 --region us-east-1 --profile testprofile

## Attachments
attachment.zip

### Set up a Helm v3 chart repository in Amazon S3

*Created by Abhishek Sharma (AWS)*

<table>
<thead>
<tr>
<th><strong>Environment:</strong></th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technologies:</strong></td>
<td>DevOps; Containers &amp; microservices; Modernization</td>
</tr>
<tr>
<td><strong>Workload:</strong></td>
<td>All other workloads</td>
</tr>
</tbody>
</table>

| **AWS services:** | Amazon S3 |

### Summary

This pattern helps you to manage Helm v3 charts efficiently by integrating the Helm v3 repository into Amazon Simple Storage Service (Amazon S3) on the Amazon Web Services (AWS) Cloud. To use this pattern, you must be familiar with Kubernetes and with Helm, which is a Kubernetes package manager. Using Helm repositories to store charts and control chart versions can improve mean time to restore (MTTR) during outages.

This pattern uses AWS CodeCommit for Helm repository creation, and it uses an S3 bucket as a Helm chart repository, so that the charts can be centrally managed and accessed by developers across the organization.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account
- Python version 2.7.12 or later
- pip
- A virtual private cloud (VPC) with subnets and an Amazon Elastic Compute Cloud (Amazon EC2) instance
- Git installed on the EC2 instance
- AWS Identity and Access Management (IAM) access to create the S3 bucket
- IAM (programmatic or role) access to Amazon S3 from the client machine
- AWS CodeCommit repository
- AWS Command Line Interface (AWS CLI)

**Product versions**
• Helm v3
• Python version 2.7.12 or later

Architecture

Target technology stack

• Amazon S3
• AWS CodeCommit
• Helm
• Kubectl
• Python and pip
• Git
• helm-s3 plugin

Target architecture

Automation and scale

• You can incorporate Helm into your existing continuous integration/continuous delivery (CI/CD) automation tool to automate the packaging and version control of Helm charts (out of scope for this pattern).
• GitVersion or Jenkins build numbers can be used to automate version control of the charts.

Tools

• Helm – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
Epics

Install and validate Helm v3

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the Helm v3 client.</td>
<td>To download and install the Helm client on your local system, run the following command: sudo curl <a href="https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3">https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3</a></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Validate the Helm installation.</td>
<td>To validate the Helm client, run the following command: helm version --short</td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
</tbody>
</table>

Initialize an S3 bucket as a Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket for Helm charts.</td>
<td>Create a unique S3 bucket. In the bucket, create a folder called stable/myapp. The example in this pattern uses s3://my-helm-charts/stable/myapp as the target chart repository.</td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Install the helm-s3 plugin for Amazon S3.</td>
<td>To install the helm-s3 plugin on your client machine, run the following command: helm plugin install <a href="https://github.com/hypnoglow/helm-s3.git">https://github.com/hypnoglow/helm-s3.git</a></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Initialize the Amazon S3 Helm repository.</td>
<td>To initialize the target folder as a Helm repository, use the following command: helm s3 init s3://my-helm-charts/stable/myapp The command creates an index.yaml file in the target to track all the chart information that is stored at that location.</td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
</tbody>
</table>
### Task 1: Verify the newly created Helm repository.

**Description:** To verify that the `index.yaml` file was created, run the following command: `aws s3 ls s3://my-helm-charts/stable/myapp/`

**Skills required:** Cloud Administrator, DevOps Engineer

### Task 2: Add the Amazon S3 repository to Helm on the client machine.

**Description:** To add the target repository alias to the Helm client machine, use the following command: `helm repo add stable-myapp s3://my-helm-charts/stable/myapp/`

**Skills required:** Cloud Administrator, DevOps Engineer

---

### Package and publish charts in the Amazon S3 Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task 1: Clone your Helm charts:</strong></td>
<td>If no local Helm charts are present on in your CodeCommit repository, clone them from your GitHub repo by running the following command: <code>git clone &lt;url_of_your_helm_source_code&gt;.git</code></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
</tbody>
</table>
| **Task 2: Package the local Helm chart:** | To package the chart that you created or cloned, use the following command: `helm package ./my-app`

   As an example, this pattern uses the `my-app` chart. The command packages all the contents of the `my-app` chart folder into an archive file, which is named using the version number that is mentioned in the `Chart.yaml` file. | Cloud Administrator, DevOps Engineer |
| **Task 3: Store the local package in the Amazon S3 Helm repository:** | To upload the local package to the Helm repository in Amazon S3, run the following command: `helm s3 push ./my-app-0.1.0.tgz stable-myapp`

   In the command, `my-app` is your chart folder name, `0.1.0` is the chart version mentioned in `Chart.yaml`, and `stable-myapp` is the target repository alias. | Cloud Administrator, DevOps Engineer |
### Upgrade your Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for the Helm chart.</td>
<td>To confirm that the chart appears both locally and in the Amazon S3 Helm repository, run the following command: <code>helm search repo stable-myapp</code></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Modify and package the chart.</td>
<td>In <code>values.yaml</code>, set the <code>replicaCount</code> value to 1, and then package the chart, this time changing the version in <code>Chart.yaml</code> to 0.1.1. Version control is ideally achieved through automation by using tools like GitVersion or Jenkins build numbers in a CI/CD pipeline. Automating the version number is out of scope for this pattern. To package the chart, run the following command: <code>helm package ./my-app/</code></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Push the new version to the Helm repository in Amazon S3.</td>
<td>To push the new package, version of 0.1.1, to the <code>my-helm-charts</code> Helm repository in Amazon S3, run the following command: <code>helm s3 push ./my-app-0.1.1.tgz stable-myapp</code></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
<tr>
<td>Verify the updated Helm chart.</td>
<td>To confirm that the updated chart appears both locally and in the Amazon S3 Helm repository, run the following commands. &lt;br&gt; <code>helm repo update</code>&lt;br&gt; <code>helm search repo stable-myapp</code></td>
<td>Cloud Administrator, DevOps Engineer</td>
</tr>
</tbody>
</table>

### Search for and install a chart from the Amazon S3 Helm repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for all versions of the my-app chart.</td>
<td>To view all the available versions of a chart, run the following command with the <code>--versions</code> flag: <code>helm search repo my-app --versions</code></td>
<td>DevOps Engineer</td>
</tr>
</tbody>
</table>
### Task 1: Install a chart from the Amazon S3 Helm repository.

**Description:**
Without the flag, Helm by default displays the latest uploaded version of a chart. Automated installation is out of scope for this pattern, but you can manually install. The search results from the previous task show the multiple versions of the `my-app` chart. To install the new version (0.1.1) from the Amazon S3 Helm repository, use the following command:

```
helm upgrade --install my-app-release stable-myapp/my-app --version 0.1.1 --namespace dev
```

**Skills required:**
DevOps Engineer

### Task 2: Roll back to a previous version by using Helm

**Description:**
Automated rollback is out of scope for this pattern, but you can roll back to an earlier version manually. Before you switch or roll back to a working version, and for an additional layer of validation before installing a revision, view which values were passed to each of the revisions by using the following command:

```
helm get values --revision=2 my-app-release
```

**Description:**
Automated rollback is out of scope for this pattern. To manually roll back to a previous revision, use the following command:

```
helm rollback my-app-release 1
```

This example is rolling back to revision number 1.

**Skills required:**
DevOps Engineer

---

### Related resources

- HELM documentation
- helm-s3 plugin (MIT License)
- Amazon S3

---

529
Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let's Encrypt

Created by Mahendra Siddappa (AWS) and Vasanth Jeyaraj (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: PoC or pilot</th>
<th>Technologies: DevOps; Containers &amp; microservices; Security, identity, compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>• End-to-end encryption on Amazon EKS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AWS services: Amazon EKS; Amazon Route 53

Summary

Implementing end-to-end encryption can be complex and you need to manage certificates for each asset in your microservices architecture. Although you can terminate the Transport Layer Security (TLS) connection at the edge of the Amazon Web Services (AWS) network with a Network Load Balancer or Amazon API Gateway, some organizations require end-to-end encryption.

This pattern is intended for organizations that require mutual authentication between all microservices in their applications. Mutual TLS reduces the burden of maintaining user names or passwords and can also use the turnkey security framework. This pattern's approach is compatible if your organization has a large number of connected devices or must comply with strict security guidelines.

This pattern helps increase your organization's security posture by implementing end-to-end encryption for applications running on Amazon Elastic Kubernetes Service (Amazon EKS). This pattern provides a sample application and code in the GitHub End-to-end encryption on Amazon EKS repository to show how a microservice runs with end-to-end encryption on Amazon EKS. The pattern's approach uses cert-manager, an add-on to Kubernetes, with Let's Encrypt as the certificate authority (CA). Let's Encrypt is a cost-effective solution to manage certificates and provides free certificates that are valid for 90 days. Cert-manager automates the on-demand provisioning and rotating of certificates when a new microservice is deployed on Amazon EKS.

Prerequisites and limitations

Prerequisites

• An active AWS account.
• An existing Amazon EKS cluster.
• AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows.
• The kubectl command line utility, installed and configured to access the Amazon EKS cluster. For more information about this, see Installing kubectl in the Amazon EKS documentation.
• An existing Domain Name System (DNS) name to test the application. For more information about this, see Registering domain names using Amazon Route 53 in the Amazon Route 53 documentation.
• The latest Helm version, installed on your local machine. For more information about this, see Using Helm with Amazon EKS in the Amazon EKS documentation and the GitHub Helm repository.
• The GitHub End-to-end encryption on Amazon EKS repository, cloned to your local machine.
• Replace the following values in the policy.json and trustpolicy.json files from the cloned GitHub End-to-end encryption on Amazon EKS repository:
• `<account_number>` – Replace with the AWS account ID for the account that you want to deploy the solution in.
• `<zone_id>` – Replace with the domain name’s Route 53 zone ID.
• `<node_group_role>` – Replace with the name of the AWS Identity and Access Management (IAM) role associated with the Amazon EKS nodes.
• `<namespace>` – Replace with the Kubernetes namespace in which you deploy the NGINX Ingress Controller and the sample application.
• `<application-domain-name>` – Replace with the DNS domain name from Route 53.

Limitations

• This pattern doesn’t describe how to rotate certificates and only demonstrates how to use certificates with microservices on Amazon EKS.

Architecture

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

The diagram shows the following workflow:

1. A client sends a request to access the application to the DNS name.
2. The Route 53 record is a CNAME to the Network Load Balancer.
3. The Network Load Balancer forwards the request to the NGINX Ingress Controller that is configured with a TLS listener. Communication between the NGINX Ingress Controller and the Network Load Balancer follows HTTPS protocol.
4. The NGINX Ingress Controller carries out path-based routing based on the client's request to the application service.
5. The application service forwards the request to the application pod. The application is designed to use the same certificate by calling secrets.
6. Pods run the sample application using the cert-manager certificates. The communication between the NGINX Ingress Controller and the pods uses HTTPS.
**Note:** Cert-manager runs in its own namespace. It uses a Kubernetes cluster role to provision certificates as secrets in specific namespaces and that can then be attached to application pods and NGINX Ingress Controller.

**Tools**

- **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- **Elastic Load Balancing** – Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, containers, and IP addresses.
- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- **Amazon Route 53** – Route 53 is a highly available and scalable Domain Name System (DNS) web service.
- **Cert-manager** – Cert-manager is an add-on to Kubernetes that requests certificates, distributes them to Kubernetes containers, and automates certificate renewal.
- **NGINX Ingress Controller** – NGINX Ingress Controller is a traffic management solution for cloud-native apps in Kubernetes and containerized environments.

**Epics**

Create and configure a public hosted zone with Route 53

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a public hosted zone in Route 53.</td>
<td>Sign in to the AWS Management Console, open the Amazon Route 53 console, choose <strong>Hosted zones</strong>, and then choose <strong>Create hosted zone</strong>. Create a public hosted zone and record the zone ID. For more information about this, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/creating-hosted-zone.html">Creating a public hosted zone</a> in the Amazon Route 53 documentation. <strong>Note:</strong> ACME DNS01 uses the DNS provider to post a challenge for cert-manager to issue the certificate.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

Configure an IAM role to allow cert-manager to access the public hosted zone

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the IAM policy for cert-manager.</td>
<td>An IAM policy is required to provide cert-manager with permission to validate that you own the Route 53 domain.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **The policy.json sample IAM policy is provided in the 1-IAMRole directory in the cloned GitHub End-to-end encryption on Amazon EKS repository.** | Run the following command in AWS CLI to create the IAM policy:  
```bash
aws iam create-policy --policy-name PolicyForCertManager --policy-document file://policy.json
``` |  |
| **Create the IAM role for cert-manager.** | After you create the IAM policy, you must create an IAM role. The trustpolicy.json sample IAM role is provided in the 1-IAMRole directory.  
Run the following command in AWS CLI to create the IAM role:  
```bash
aws iam create-role --role-name RoleForCertManager --assume-role-policy-document file://trustpolicy.json
``` | AWS DevOps |
| **Attach the policy to the role.** | Run the following command in AWS CLI to attach the IAM policy to the IAM role:  
```bash
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/PolicyForCertManager --role-name RoleForCertManager
``` | AWS DevOps |

### Set up the NGINX Ingress Controller in Amazon EKS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deploy the NGINX Ingress Controller.</strong></td>
<td>Install the most recent version of nginx-ingress using Helm. You can modify the nginx-ingress configuration according to your requirements before deploying it. This pattern uses an annotated, internal-facing Network Load Balancer</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>and that is available in the 5-Nginx-Ingress-Controller directory.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>Install the NGINX Ingress Controller by running the following Helm command from the 5-Nginx-Ingress-Controller directory:</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>helm install test-nginx nginx-stable/nginx-ingress -f 5-Nginx-Ingress-Controller/values_internal_nlb.yaml</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>Verify that the NGINX Ingress Controller is installed.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>Run the helm list command and the output should show that the NGINX Ingress Controller is installed.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Create a Route 53 A record.</td>
<td>The A record points to the Network Load Balancer created by NGINX Ingress Controller.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>1. On the Amazon Elastic Compute Cloud (Amazon EC2) console, choose <strong>Load Balancer</strong>, and then copy the Network Load Balancer's DNS name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. On the Amazon Route 53 console, choose <strong>Public Hosted Zone</strong>, choose <strong>Create record</strong>, and then enter a name for the record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Choose <strong>A - Routes traffic to IPv4 and some AWS resources</strong> as the <strong>Record type</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Enable alias and create an A record alias that points to the Network Load Balancer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. On the Amazon Route53 console, choose <strong>Public Hosted Zone</strong>, choose <strong>Create record</strong>, and then choose <strong>Supply record name</strong>. Choose the <strong>A - Routes traffic to IPv4 and some AWS resources</strong> type and enable alias.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Under <strong>Route traffic</strong>, choose the Network Load Balancer and AWS Region, and then choose the DNS for the Network Load Balancer.</td>
<td></td>
</tr>
</tbody>
</table>

**Set up NGINX VirtualServer on Amazon EKS**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy NGINX VirtualServer.</td>
<td>The NGINX VirtualServer resource is a load balancing configuration that is an alternative to the ingress resource. The configuration to create the NGINX VirtualServer resource is available in the nginx_virtualserver.yaml file in the 6-Nginx-Virtual-Server directory. Run the following command in</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
<tr>
<td>kubectl to create the NGINX VirtualServer resource: kubectl apply -f nginx_virtualserver.yaml</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Important:</strong> Make sure that you update the application domain name, certificate secret, and application service name in the nginx_virtualserver.yaml file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify that NGINX VirtualServer is created.</td>
<td>Run the following command in kubectl to verify that the NGINX VirtualServer resource was successfully created: kubectl get virtualserver</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td><strong>Note:</strong> Verify that the Host column matches your application's domain name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy the NGINX web server with TLS enabled.</td>
<td>This pattern uses a NGINX web server with TLS enabled as the application for testing end-to-end encryption. The configuration files required to deploy the test application are available in the demo-webserver directory. Run the following command in kubectl to deploy the test application: kubectl apply -f nginx-tls-ap.yaml</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Verify that the test application resources are created.</td>
<td>Run the following commands in kubectl to verify that the required resources are created for the test application: • kubectl get deployments Note: Validate the Ready column and Available column • kubectl get pods</td>
<td>grep -i example-deploy Note: Pods should be in running state • kubectl get configmap • kubectl get svc</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Validate the application. | 1. Run the following command by replacing the `<application-domain-name>` with the Route53 DNS name that you created earlier: `curl --verbose https://<application-domain-name>`  
2. Verify that you can access the application. | AWS DevOps       |

### Related resources

- Creating records by using the Amazon Route 53 console ([Amazon Route 53 documentation](#))
- Using a Network Load Balancer with the NGINX ingress controller on Amazon EKS ([Amazon Open Source Blog](#))
- Cert-manager documentation

### Subscribe multiple email endpoints to an SNS topic by using a custom resource

*Created by Ricardo Morais (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>DevOps</th>
<th>AWS services:</th>
<th>Amazon SNS; AWS CloudFormation; AWS Lambda</th>
</tr>
</thead>
</table>

### Summary

This pattern describes how to subscribe multiple email addresses to receive notifications from an Amazon Simple Notification Service (Amazon SNS) topic. It uses an AWS Lambda function as a custom resource in an AWS CloudFormation template. The Lambda function is associated with an input parameter that specifies the email endpoints for the SNS topic.

Currently, you can use the AWS CloudFormation template objects `AWS::SNS::Topic` and `AWS::SNS::Subscription` to subscribe single endpoints to SNS topics. To subscribe multiple endpoints, you have to invoke the object multiple times. By using the Lambda function as a custom resource, you can subscribe multiple endpoints through an input parameter. You can use this Lambda function as a custom resource in any AWS CloudFormation template.

### Prerequisites and limitations

**Prerequisites**
• An active AWS account.
• An AWS profile configured in your local environment with an access key and secret key. You can also run this code from AWS Cloud9.
• Permissions for the following:
  • AWS Identity and Access Management (IAM) role and policy
  • AWS Lambda function
  • Amazon Simple Storage Service (Amazon S3) for uploading the Lambda function
  • Amazon SNS topic and policy
  • AWS CloudFormation stacks

Limitations
• The code supports Linux and macOS workstations.

Product versions
• AWS Command Line Interface (AWS CLI) version 2 or later.

Architecture

Target technology stack
• AWS CloudFormation
• Amazon SNS
• AWS Lambda

Tools

Tools
• AWS CLI version 2

Code

The attachment includes the following files:
• Lambda function: lambda_function.py
• AWS CloudFormation template: template.yaml
• Two parameter files to handle multiple or single email endpoint subscriptions: parameters-multiple-values.json (used as the default) and parameters-one-value.json

To deploy the stack, you can use either parameter file. To specify multiple email endpoints:

```bash
./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION>
```

To specify a single email endpoint:
./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION> -f parameters-one-value.json

Epics

Option 1 - Deploy an SNS topic with one email subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the email endpoint for SNS topic subscriptions.</td>
<td>Edit the file parameters-one-value.json (attached), and change the value of the pSNSNotificationsEmail parameter to reflect the email address you want to use, such as <a href="mailto:someone@example.com">someone@example.com</a>.</td>
<td></td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation stack that creates the resources and subscription.</td>
<td>Run the “deploy.sh” command with your AWS profile name, AWS Region, and the parameters-one-value.json file, as shown in the Code section.</td>
<td>IAM role with proper permissions</td>
</tr>
</tbody>
</table>

Option 2 - Deploy an SNS topic with two or more email subscriptions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the email endpoints for SNS topic subscriptions.</td>
<td>Edit the file parameters-multiple-values.json (attached), and change the value of the pSNSNotificationsEmail parameter to reflect the email addresses you want to use, separated by commas, as follows: <a href="mailto:someone1@example.com">someone1@example.com</a>, <a href="mailto:someone2@example.com">someone2@example.com</a>.</td>
<td></td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation stack that creates the resources and subscription.</td>
<td>Run the “deploy.sh” command with your AWS profile name and AWS Region, as shown in the Code section. You don't have to specify the parameters-multiple-values.json file because it's used by default.</td>
<td>IAM role with proper permissions</td>
</tr>
</tbody>
</table>

Option 3 - Deploy an SNS topic through an AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SNS topic.</td>
<td>Create an SNS topic through an AWS CloudFormation</td>
<td>IAM role with proper permissions</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>template, without specifying subscription endpoints in the AWS::SNS::Topic template object. You can use template.yaml in the attachment as a starting point.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an SNS topic policy.</td>
<td>Create an SNS topic policy in the AWS CloudFormation template.</td>
<td>IAM role with proper permissions</td>
</tr>
<tr>
<td>Subscribe the email endpoints list to the SNS topic.</td>
<td>Based on the list of email endpoints (one or more), subscribe the endpoints to the SNS topic you created.</td>
<td>IAM role with proper permissions</td>
</tr>
</tbody>
</table>

Related resources

References

- AWS CloudFormation custom resources (AWS documentation)
- AWS CloudFormation custom resource creation with Python, AWS Lambda, and crhelper (blog post)

Required tools

- AWS CLI version 2

Attachments

attachment.zip

Use Serverspec for test-driven development of infrastructure code

Created by Sushant Jagdale (AWS)

Environment: PoC or pilot

Technologies: DevOps; Infrastructure; Hybrid cloud

AWS services: Amazon EC2; AWS CodeBuild; AWS CodeDeploy

Summary

This pattern shows you how to use Serverspec to use test-driven development (TDD) when writing infrastructure code on the Amazon Web Services (AWS) Cloud. The pattern also covers automation
with AWS CodePipeline. TDD will focus attention on what the infrastructure code must do and sets a clear definition of done. You can use Serverspec to test infrastructure created by tools such as AWS CloudFormation, Terraform by HashiCorp, and Ansible.

Serverspec helps with refactoring infrastructure code. With Serverspec, you can write RSpec tests to check installation of various packages and software, run commands, check for running processes and ports, check file permission settings, and so forth. Serverspec checks whether your servers are configured correctly. You install only Ruby on your servers. You don't need to install any agent software.

Test-driven infrastructure provides the following benefits:

- Cross-platform testing
- Validation of expectations
- Confidence in your automation
- Infrastructure consistency and stability
- Fail early

You can use this pattern to run Serverspec unit tests for Apache software and check file permission settings during Amazon Machine Image (AMI) creation. An AMI will be created only if all the test cases pass. Serverspec will perform following tests:

- Apache process is running.
- Apache port is running.
- Apache configuration files and directories exist at certain locations, and so forth.
- File permissions are correctly configured.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- A virtual private cloud (VPC) with a public subnet
- Installation of AWS Command Line Interface (AWS CLI) and Git

Product versions

- HashiCorp Packer version: 1.6.6
- Ruby version: 2.5.1 and later
- AWS CLI version: 1.18.185

Architecture

Target architecture
1. When you push the code to the CodeCommit repository, an Amazon CloudWatch Events event engages the CodePipeline. In the first stage of the pipeline, the code is fetched from CodeCommit.

2. The second pipeline stage runs CodeBuild, which validates and builds the Packer template.

3. As a part of the Packer build provisioner, Packer installs Apache and Ruby software. Then the provisioner calls a shell script that uses Serverspec to unit test the Apache process, port, files, and directories. The Packer post-processor writes a JavaScript Object Notation (JSON) file with a list of all the artifacts produced by Packer during a run.

4. Finally, an Amazon Elastic Compute Cloud (Amazon EC2) instance is created using the AMI ID produced by Packer.

**Tools**

- **AWS CLI** – Amazon Command Line Interface (AWS CLI) is an open source tool for interacting with AWS services using commands in your command line shell.
- **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in Amazon Web Services (AWS) resources.
- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeCommit** – AWS CodeCommit is a version control service hosted by Amazon Web Services. You can use CodeCommit to privately store and manage assets (such as documents, source code, and binary files) in the cloud.
- **AWS CodePipeline** – AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process.
- **HashiCorp Packer** – HashiCorp Packer is a tool for automating the creation of identical machine images from a single source configuration.
- **Serverspec** – Serverspec runs RSpec tests to check server configuration. Serverspec uses Ruby, and you don’t need to install agent software.

**Code**

The code is attached. The code uses the following structure, with three directories and eight files.

```bash
### amazon-linux_packer-template.json (Packer template)
### buildspec.yaml (CodeBuild .yaml file)
### pipeline.yaml (AWS CloudFormation template to automate CodePipeline)
### rspec_tests (RSpec required files and spec)
  #  #### Gem-file
  #  #### Rakefile
  #  #### spec
```
Epics

Configure AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM user.</td>
<td>Create an AWS Identity and Access Management (IAM) user with programmatic and console access. For more information, see the AWS documentation.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Configure AWS credentials.</td>
<td>On your local computer or in your environment, configure AWS credentials for the IAM user. For instructions, see the AWS documentation.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Test your credentials.</td>
<td>To validate the configured credentials, run the following command.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>aws sts get-caller-identity --profile &lt;profile&gt;</td>
<td></td>
</tr>
</tbody>
</table>

AWS CodePipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CodeCommit repository.</td>
<td>To create a CodeCommit repository, run the following command.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>aws codecommit create-repository --repository-name &quot;&lt;provide repository-name&gt;&quot; --repository-description &quot;repository to unit test the infrastructure code&quot;</td>
<td></td>
</tr>
<tr>
<td>Write RSpec tests.</td>
<td>Create RSpec test cases for your infrastructure. For more information, see the Additional information section.</td>
<td>Developer, DevOps engineer</td>
</tr>
<tr>
<td>Push code to the CodeCommit repository.</td>
<td>To push the attached code to the CodeCommit repository, run the following commands.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Create the pipeline.</td>
<td>To create the pipeline, run the AWS CLI command that is in the Additional information section.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Start the pipeline.</td>
<td>Commit code to the CodeCommit repository. Any commit to the repository will initiate the pipeline.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td>Test the Apache URL.</td>
<td>To test the AMI installation, use the following URL.</td>
<td>Developer, Systems administrator, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>http://&lt;your instance public ip&gt;/hello.html</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The page will show a &quot;Hello from Apache&quot; message.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

- HashiCorp
- HashiCorp Packer
- Serverspec
- Introduction to ServerSpec: What is ServerSpec and how do we use it at Stelligent? (external blog post)
- Test-driven development of infrastructure code (external blog post)
- Image creation and testing with HashiCorp Packer and ServerSpec (external article)

Additional information

Write RSpec tests

The RSpec test for this pattern is located at <repository folder>/rspec_tests/spec/apache_spec.rb.

```ruby
require 'spec_helper'

describe service('httpd') do
  it { should be_enabled }
  it { should be_running }
end

describe port(80) do
```
it { should be_listening }
end

describe file('/etc/httpd/conf/httpd.conf') do
  it { should exist }
  it { should be_owned_by 'root' }
  it { should contain 'ServerName www.example.com' }
end

describe file('/etc/httpd/conf/httpd.conf') do
  its(:content) { should match /ServerName www.example.com/ }
end

describe file('/var/www/html/hello.html') do
  it { should exist }
  it { should be_owned_by 'ec2-user' }
end

describe file('/var/log/httpd') do
  it { should be_directory }
end

describe file('/etc/sudoers') do
  it { should be_mode 440 }
end

describe group('root') do
  it { should have_gid 0 }
end

You can add your own tests to the /spec directory.

Create the pipeline

```
aws cloudformation create-stack --stack-name myteststack --template-body file://
pipeline.yaml --parameters ParameterKey=RepositoryName,ParameterValue=<provide
repository-name> ParameterKey=ApplicationName,ParameterValue=<provide
application-name> ParameterKey=SecurityGroupId,ParameterValue=<provide
SecurityGroupId> ParameterKey=VpcId,ParameterValue=<provide VpcId>
ParameterKey=SubnetId,ParameterValue=<provide SubnetId> ParameterKey=Region,ParameterValue=<provide Region
AccountId> --capabilities CAPABILITY_NAMED_IAM
```

Parameter details

repository-name – The name of the AWS CodeCommit repository

application-name – The Amazon Resource Name (ARNs) are linked with ApplicationName; provide any name

SecurityGroupId – Any security group ID from your AWS account that has port 80 open

VpcId – The ID of your VPC

SubnetId – The ID of a public subnet in your VPC

Region – The AWS Region where you are running this pattern

Keypair – The Secure Shell (SSH) key name to log in to the EC2 instance
AccountId – Your AWS account ID

You can also create a CodePipeline pipeline by using the AWS Management Console and passing the same parameters that are in the previous command line.

Attachments

attachment.zip

Use third-party Git source repositories in AWS CodePipeline

Created by Kirankumar Chandrashekar (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>DevOps</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
</tr>
</tbody>
</table>

| AWS services: | AWS CodeBuild; AWS CodePipeline; AWS Lambda |

Summary

This pattern describes how to use AWS CodePipeline with third-party Git source repositories.

AWS CodePipeline is a continuous delivery service that automates tasks for building, testing, and deploying your software. The service currently supports Git repositories managed by GitHub, AWS CodeCommit, and Atlassian Bitbucket. However, some enterprises use third-party Git repositories that are integrated with their single sign-on (SSO) service and Microsoft Active Directory for authentication. You can use these third-party Git repositories as sources for CodePipeline by creating custom actions and webhooks.

A webhook is an HTTP notification that detects events in another tool, such as a GitHub repository, and connects those external events to a pipeline. When you create a webhook in CodePipeline, the service returns a URL that you can use in your Git repository webhook. If you push code to a specific branch of the Git repository, the Git webhook initiates the CodePipeline webhook through this URL, and sets the source stage of the pipeline to In Progress. When the pipeline is in this state, a job worker polls CodePipeline for the custom job, runs the job, and sends a success or failure status to CodePipeline. In this case, because the pipeline is in the source stage, the job worker gets the contents of the Git repository, zips the contents, and uploads it to the Amazon Simple Storage Service (Amazon S3) bucket where artifacts for the pipeline are stored, using the object key provided by the polled job. You can also associate a transition for the custom action with an event in Amazon CloudWatch, and initiate the job worker based on the event. This setup enables you to use third-party Git repositories that the service doesn’t natively support as sources for CodePipeline.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Git repository that supports webhooks and can connect to a CodePipeline webhook URL through the internet
AWS Command Line Interface (AWS CLI) installed and configured to work with the AWS account

Architecture

The pattern involves these steps:

1. The user commits code to a Git repository.
2. The Git webhook is called.
3. The CodePipeline webhook is called.
4. The pipeline is set to In Progress, and the source stage is set to the In Progress state.
5. The source stage action initiates a CloudWatch Events rule, indicating that it was started.
6. The CloudWatch event initiates a Lambda function.
7. The Lambda function gets the details of the custom action job.
8. The Lambda function initiates AWS CodeBuild and passes it all the job-related information.
9. CodeBuild gets the public SSH key or user credentials for HTTPS Git access from Secrets Manager.
10. CodeBuild clones the Git repository for a specific branch.
11. CodeBuild zips the archive and uploads it to the S3 bucket that serves as the CodePipeline artifact store.

Tools

- AWS CodePipeline – AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deployment phases of your release process for each code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin.
AWS Prescriptive Guidance Patterns
Epics

- **AWS Lambda** – AWS Lambda lets you run code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service with no administration necessary. You upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically initiate from other AWS services or call it directly from any web or mobile app.

- **AWS CodeBuild** – AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools.

- **AWS Secrets Manager** – AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets by calling Secrets Manager APIs, without having to hardcode sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon Relational Database Service (Amazon RDS), Amazon Redshift, and Amazon DocumentDB. The service can be extended to support other types of secrets, including API keys and OAuth tokens. In addition, Secrets Manager lets you control access to secrets by using fine-grained permissions, and audit secret rotation centrally for resources in the AWS Cloud, third-party services, and on-premises environments.

- **Amazon CloudWatch** – Amazon CloudWatch is a monitoring and observation service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers. CloudWatch provides you with data and actionable insights to monitor your applications, respond to systemwide performance changes, optimize resource utilization, and get a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, providing you with a unified view of AWS resources, applications, and services that run on AWS and on-premises servers. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that lets you store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features to help you organize your data and configure finely tuned access controls to meet your specific business, organizational, and compliance requirements.

## Epics

**Create a custom action in CodePipeline**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a custom action using AWS CLI or AWS CloudFormation.</td>
<td>This step involves creating a custom source action that can be used in the source stage of a pipeline in your AWS account in a particular region. You must use AWS CLI or AWS CloudFormation (not the console) to create the custom source action. For more information about the commands and steps described in this and other epics, see the &quot;Related resources&quot;</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
section at the end of this pattern. In AWS CLI, use the create-custom-action-type command. Use --configuration-properties to provide all the parameters required for the job worker to process when it polls CodePipeline for a job. Make sure to note the values provided to the --provider and --action-version options, so that you can use the same values when creating the pipeline with this custom source stage. You can also create the custom source action in AWS CloudFormation by using the resource type AWS::CodePipeline::CustomActionType.

Set up authentication

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SSH key pair.</td>
<td>Create a Secure Shell (SSH) key pair. For instructions, see the GitHub documentation.</td>
<td>Systems/DevOps engineer</td>
</tr>
<tr>
<td>Create a secret in AWS Secrets Manager.</td>
<td>Copy the contents of the private key from the SSH key pair and create a secret in AWS Secrets Manager. This secret is used for authentication when accessing the Git repository.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Add the public key to the Git repository.</td>
<td>Add the public key from the SSH key pair to the Git repository account settings, for authentication against the private key.</td>
<td>Systems/DevOps engineer</td>
</tr>
</tbody>
</table>

Create a pipeline and webhook

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a pipeline that includes the custom source action.</td>
<td>Create a pipeline in CodePipeline. When you configure the source stage, choose the custom source action that you created previously. You can do this in the AWS CodePipeline console or in AWS</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th><strong>Create a CodePipeline webhook.</strong></th>
<th><strong>Description</strong></th>
<th><strong>Skills required</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a webhook for the pipeline you created with the custom source action. You must use AWS CLI or AWS CloudFormation (not the console) to create the webhook. In AWS CLI, run the put-webhook command and provide the appropriate values for the webhook options. Make a note of the webhook URL that the command returns. If you're using AWS CloudFormation to create the webhook, use the resource type AWS::CodePipeline::Webhook. Make sure to output the webhook URL from the created resource, and make a note of it.</td>
<td>General AWS</td>
<td><strong>Skills required</strong></td>
</tr>
</tbody>
</table>

CLI. CodePipeline prompts you for the configuration properties that you set on the custom action. This information is required for the job worker to process the job for the custom action. Follow the wizard and create the next stage for the pipeline.
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Lambda function and CodeBuild project.</td>
<td>In this step, you use Lambda and CodeBuild to create a job worker that will poll CodePipeline for job requests for the custom action, run the job, and return the status result to CodePipeline. Create a Lambda function that is initiated by an Amazon CloudWatch Events rule when the custom source action stage of the pipeline transitions to &quot;In Progress.&quot; When the Lambda function is initiated, it should get the custom action job details by polling for jobs. You can use the PollForJobs API to return this information. After the polled job information is obtained, the Lambda function should return an acknowledgment, and then process the information with the data it obtains from the configuration properties for the custom action. When the worker is ready to talk to the Git repository, you might initiate a CodeBuild project, because it's convenient to handle Git tasks by using the SSH client.</td>
<td>General AWS, code developer</td>
</tr>
</tbody>
</table>

### Create an event in CloudWatch

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudWatch Events rule.</td>
<td>Create a CloudWatch Events rule that initiates the Lambda function as a target whenever the pipeline's custom action stage transitions to &quot;In Progress.&quot;</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

### Related resources

**Creating a custom action in CodePipeline**

- Create and add a custom action in CodePipeline
- AWS::CodePipeline::CustomActionType resource

**Setting up authentication**
• Creating and Managing Secrets with AWS Secrets Manager

Creating a pipeline and webhook
• Create a Pipeline in CodePipeline
• put-webhook command reference
• AWS::CodePipeline::Webhook resource
• PollForJobs API reference
• Create and Add a Custom Action in CodePipeline
• Create a build project in AWS CodeBuild

Creating an event
• Detect and react to changes in pipeline state with Amazon CloudWatch Events

Additional references
• Working with pipelines in CodePipeline
• AWS Lambda developer guide

More patterns
• Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer (p. 138)
• Automate adding or updating Windows registry entries using AWS Systems Manager (p. 687)
• Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch (p. 148)
• Automate deployment of nested applications using AWS SAM (p. 1416)
• Automate RabbitMQ configuration in Amazon MQ (p. 742)
• Automate the creation of AppStream 2.0 resources using AWS CloudFormation (p. 554)
• Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline (p. 156)
• Automatically validate and deploy IAM policies and roles in an AWS account by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros (p. 1621)
• Back up Sun SPARC servers in the Stromasys Charon-SSP emulator on the AWS Cloud (p. 1785)
• Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager (p. 1345)
• Create alarms for custom metrics using Amazon CloudWatch anomaly detection (p. 705)
• Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code (p. 281)
• Deploy Kubernetes resources and packages using Amazon EKS and a Helm chart repository in Amazon S3 (p. 184)
• Deploy multiple-stack applications using AWS CDK with TypeScript (p. 1408)
• Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates (p. 1655)
• Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK (p. 713)
• Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet (p. 219)
• Mainframe modernization: DevOps on AWS with Micro Focus (p. 597)
• Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK (p. 1459)
• Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker using AWS Developer Tools (p. 1131)
• Optimize AWS App2Container generated Docker images (p. 229)
• Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions (p. 58)
• Replicate filtered Amazon ECR container images across accounts or Regions (p. 235)
• Rotate credentials without restarting containers (p. 246)
• Run AWS Systems Manager automation tasks synchronously from AWS Step Functions (p. 1751)
• Set up CI/CD for AWS AppSync GraphQL API updates (p. 1482)
• Visualize AI/ML model results using Flask and AWS Elastic Beanstalk (p. 672)
End-user computing

Topics
• Automate the creation of AppStream 2.0 resources using AWS CloudFormation (p. 554)
• More patterns (p. 558)

Automate the creation of AppStream 2.0 resources using AWS CloudFormation

Created by Ram Kandaswamy (AWS)

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: End-user computing; Cloud-native; Cost management; DevOps; SaaS</th>
<th>Workload: Microsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services:</strong> Amazon AppStream 2.0; AWS CloudFormation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides code samples and steps to automate the creation of Amazon AppStream 2.0 resources in the Amazon Web Services (AWS) Cloud by using an AWS CloudFormation template. The pattern shows you how to use an AWS CloudFormation stack to automate the creation of your AppStream 2.0 application resources, including an image builder, image, fleet instance, and stack. You can stream your AppStream 2.0 application to end users on an HTML5-compliant browser by using either the desktop or application delivery mode.

Prerequisites and limitations

Prerequisites
• An active AWS account
• An acceptance of AppStream 2.0 terms and conditions
• Basic knowledge of AppStream resources, such as stacks, fleets, and image builders

Limitations
• You can’t modify the AWS Identity and Access Management (IAM) role associated with an AppStream 2.0 instance after that instance is created.
• You can’t modify properties (such as the subnet or security group) on the AppStream 2.0 image builder instance after that image builder is created.
The following diagram shows you how to automate the creation of AppStream 2.0 resources by using an AWS CloudFormation template.

The diagram shows the following workflow:

1. You create an AWS CloudFormation template based on the YAML code in the Additional information section of this pattern.
2. The AWS CloudFormation template creates an AWS CloudFormation test stack.
   a. (Optional) You create an image builder instance by using AppStream 2.0.
   b. (Optional) You create a Windows image by using your custom software.
3. The AWS CloudFormation stack creates an AppStream 2.0 fleet instance and stack.
4. You deploy your AppStream 2.0 resources to end users on an HTML5-compliant browser.

Technology stack

- Amazon AppStream 2.0
- AWS CloudFormation

Tools

- **Amazon AppStream 2.0** – Amazon AppStream 2.0 is a fully managed application streaming service that provides you with instant access to your desktop applications from anywhere. AppStream 2.0 manages the AWS resources required to host and run your applications, scales automatically, and provides access to your users on demand.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together.
as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

**Epics**

(Optional) Create an AppStream 2.0 image

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Install custom software and create an image. | 1. Install the AppStream 2.0 application that you plan to deploy to your users.  
2. Use the Photon create image agent or a PowerShell script to create a new Windows image for your custom software.  

**Note:** Consider using the Windows AppLocker feature to further lock down the image. | AWS DevOps, Cloud architect |

Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Update the AWS CloudFormation template. | 1. Save the code in the Additional information section of this pattern as a YAML file.  
2. Update the YAML file with the required values for the parameters in your environment. | AWS systems administrator, Cloud administrator, Cloud architect, General AWS, AWS administrator |
| Create an AWS CloudFormation stack using the template. | 1. Sign in to the AWS Management Console and open the AWS CloudFormation console.  
2. On the navigation pane, choose Stacks.  
3. Choose Create stack and then choose With new resources (standard).  
4. In the Prerequisite – Prepare template section, choose Template is ready.  
5. In the Specify template section, choose Upload a template file. | App owner, AWS systems administrator, Windows Engineer |
Related resources

References

- Get Started with Amazon AppStream 2.0: Set Up With Sample Applications
- Create an AppStream 2.0 Fleet and Stack

Tutorials and videos

- Amazon AppStream 2.0 User Workflow
- How to Migrate a Legacy Windows Forms App to Amazon AppStream 2.0
- AWS re:Invent 2018: Securely Deliver Desktop Applications with Amazon AppStream 2.0 (BAP201)

Additional information

The following code is an example of an AWS CloudFormation template that allows you to automatically create AppStream 2.0 resources.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  SubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
  testSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup::Id'
  imageName:
    Type: String
Resources:
  AppStreamFleet:
    Type: 'AWS::AppStream::Fleet'
    Properties:
      ComputeCapacity:
        DesiredInstances: 5
      InstanceType: stream.memory.largea
      Name: appstream-test-fleet
      DisconnectTimeoutInSeconds: 1200
      FleetType: ON_DEMAND
      IdleDisconnectTimeoutInSeconds: 1200
      ImageName: !Ref imageName
      MaxUserDurationInSeconds: 345600
      VpcConfig:
        SecurityGroupIds:
        - !Ref testSecurityGroup
        SubnetIds: !Ref SubnetIds
  AppStreamStack:
    Type: 'AWS::AppStream::Stack'
    Properties:
```

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>Choose Choose file and then choose your updated AWS CloudFormation template.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Complete the rest of the steps in the wizard to create your stack.</td>
<td></td>
</tr>
</tbody>
</table>
Description: AppStream stack for test
DisplayName: AppStream test Stack
Name: appstream-test-stack
StorageConnectors:
  - ConnectorType: HOMEFOLDERS
UserSettings:
  - Action: CLIPBOARD_COPY_FROM_LOCAL_DEVICE
    Permission: ENABLED
  - Action: CLIPBOARD_COPY_TO_LOCAL_DEVICE
    Permission: ENABLED
  - Action: FILE_DOWNLOAD
    Permission: ENABLED
  - Action: PRINTING_TO_LOCALDEVICE
    Permission: ENABLED
AppStreamFleetAssociation:
  Type: 'AWS::AppStream::StackFleetAssociation'
  Properties:
    FleetName: appstream-test-fleet
    StackName: appstream-test-stack
  DependsOn:
    - AppStreamFleet
    - AppStreamStack

More patterns

- Run AWS Systems Manager automation tasks synchronously from AWS Step Functions (p. 1751)
Hybrid cloud

Topics
- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode (p. 559)
- Configure VMware vRealize Automation to provision VMs on VMware Cloud on AWS (p. 563)
- Deploy a VMware SDDC on AWS by using VMware Cloud on AWS (p. 569)
- Migrate VMs to VMware Cloud on AWS by using HCX OS Assisted Migration (p. 575)
- More patterns (p. 578)

Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode

Created by Deepak Kumar (AWS)

<table>
<thead>
<tr>
<th>Environment</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies</td>
<td>Hybrid cloud; Infrastructure; Migration</td>
</tr>
<tr>
<td>Workload</td>
<td>All other workloads</td>
</tr>
<tr>
<td>AWS services</td>
<td>AWS Direct Connect</td>
</tr>
</tbody>
</table>

Summary

This pattern describes how you can use Hybrid Linked Mode to view and manage inventories in an on-premises data center and a VMware Cloud on AWS software-defined data center (SDDC) by using a single VMware vSphere Client interface.

By configuring Hybrid Linked Mode, you can migrate your on-premises virtual machines (VMs) and applications to the cloud SDDC. Your IT teams can then manage your cloud-based resources with familiar VMware tools and without requiring any new tools. You can also ensure consistent operations and simplified administration by using the VMware Cloud Gateway Appliance.

This pattern provides two options for configuring Hybrid Linked Mode, but you can only use one option at a time. The first option installs the Cloud Gateway Appliance and uses it to link from the on-premises vCenter Server to the cloud SDDC. The second option configures Hybrid Linked Mode from the cloud SDDC.

Prerequisites and limitations

Prerequisites (both options)
- An existing on-premises data center and a cloud SDDC.
- An existing connection between the on-premises data center and the cloud SDDC, using AWS Direct Connect, a VPN, or both.
- The on-premises data center and cloud SDDC are synchronized with network time protocol (NTP) or another authoritative time source.
- The maximum latency of a round-trip time between the on-premises data center and the cloud SDDC doesn't exceed 100ms.
- Cloud administrators with access to your on-premises environment.
• The vCenter Server’s fully qualified domain name (FQDN) must resolve to a private IP address.

**Prerequisites for Option 1**

• The on-premises environment should run on vSphere 6.5.0d or later.
• The Cloud Gateway Appliance and vCenter Server can communicate over AWS Direct Connect, a VPN, or both.
• The Cloud Gateway Appliance meets hardware requirements.
• Firewall ports are open.

**Prerequisites for Option 2**

• The on-premises vCenter Server runs on vSphere 6.0 Update 3 or later, or on vSphere 6.5.0d or later.
• Login credentials are available for the on-premises vSphere single sign-on (SSO) domain.
• Users in the on-premises environment have read-only access to the base distinguished name (Base DN).
• The on-premises Domain Name System (DNS) server is configured for VMware Management Gateway.
• Implement network connectivity tests using the VMware Connectivity Validator.
• Firewall ports are open.

**Limitations**

• Hybrid Linked Mode can only connect one on-premises vCenter Server Enhanced Linked Mode domain.
• Hybrid Linked Mode only supports on-premises vCenter Server running version 6.7 or later.

**Architecture**

The following diagram shows both options for configuring Hybrid Linked Mode.
Migrating different workload types using Hybrid Linked Mode

Hybrid Linked Mode supports migrating workloads between an on-premises data center and a cloud SDDC by using either a cold migration or a live migration with VMware vSphere vMotion. Factors that must be considered when choosing the migration method include the virtual switch type and version, the connection type to the cloud SDDC, and the virtual hardware version.

A cold migration is appropriate for VMs that experience downtime. You can shut down the VMs, migrate them, and then turn them back on. The migration time is faster because there is no need to copy active memory. We recommend using a cold migration for applications that accept downtime (for example, tier 3 applications or development and testing workloads). If your VMs cannot experience downtime, you should consider a live migration using vMotion for your mission-critical applications.

The following diagram provides an overview of the different workload migration types using Hybrid Linked Mode.

Tools

- **VMware Cloud on AWS** – VMware Cloud on AWS is an integrated cloud offering jointly developed by Amazon Web Services (AWS) and VMware.
- **VMware Cloud Gateway Appliance** – Cloud Gateway Appliance enables a number of hybrid cloud use cases where on-premises resources are connected to cloud resources.
- **VMware vSphere** – vSphere is VMware’s virtualization platform, which transforms data centers into aggregated computing infrastructures that include CPU, storage, and networking resources.

Epics

*(Option 1) Use Hybrid Linked Mode with the Cloud Gateway Appliance*

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure the Cloud Gateway Appliance. | 1. Log in to the VMware Cloud on AWS console and download the Cloud Gateway Appliance.  
2. Install the Cloud Gateway Appliance in your on-premises | Cloud administrator |
(Option 2) Use Hybrid Linked Mode from the cloud SDDC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure Hybrid Linked Mode from the cloud SDDC. | 1. Log in to the VMware Cloud on AWS console and use the Connectivity Validator to check all required network connectivity. For more information about this, see Validate network connectivity for Hybrid Linked Mode in the VMware documentation.  
2. Log in to the vSphere Client of the cloud SDDC, choose Menu, choose Administration, and then choose Domains.  
3. In the Hybrid Cloud section, choose Linked Domains and then connect to your on-premises vCenter Server.  
4. Add an identity source to the cloud SDDC Lightweight Directory Access Protocol (LDAP) domain. For more information about this, see Add an Identity Source to the SDDC LDAP Domain in the VMware documentation. | Cloud administrator |
Configure VMware vRealize Automation to provision VMs on VMware Cloud on AWS

**Summary**

VMware vRealize Automation is automation software that you can use to request and manage IT resources. By choosing to configure vRealize Automation with VMware Cloud on AWS, you can automate the delivery of virtual machines (VMs), applications, and IT services across multiple data centers and cloud environments.

Your IT teams can then create catalog items to configure service provisioning and operational capabilities that your users can request and use with their existing vRealize Automation tools. You can also improve your IT agility and efficiency by integrating VMware Cloud on AWS with vRealize Automation Cloud Assembly.

This pattern describes how to configure VMware vRealize Automation to automatically build VMs or application capabilities on VMware Cloud on AWS.

**Prerequisites and limitations**

**Prerequisites**

- An existing on-premises data center and a VMware Cloud on AWS software-defined data center (SDDC). For more information about the cloud SDCC, see About Software-Defined Data Centers in the VMware documentation.
- An existing connection between the on-premises data center and the cloud SDDC, using AWS Direct Connect, a VPN (route or policy-based), or both.
- The on-premises data center and cloud SDDC are synchronized with network time protocol (NTP) or another authoritative time source.
- The maximum latency of a round-trip time between the on-premises data center and the cloud SDDC doesn’t exceed 100ms.
- The vCenter Server’s fully qualified domain name (FQDN) must resolve to a private IP address.
- Cloud SDDC users with access to your on-premises environment.
- Organization owner access in the vRealize Automation Cloud Assembly service role.
- End users with permission in vRealize Automation Service Broker to consume service.
- The on-premises data center’s Classless Inter-Domain Routing (CIDR) range must be open for the generating of API tokens from the VMware Cloud on AWS console. The following list provides the minimum roles required to generate API tokens:

---

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Hybrid cloud; Infrastructure</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS Direct Connect; AWS Site-to-Site VPN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Created by Deepak Kumar (AWS)
• Organization member
• Organization owner
• Service Roles - VMware Cloud on AWS
• Administrator
• NSX Cloud Administrator
• NSX Cloud Auditor

For more information about this, see Connectivity Options for VMware Cloud on AWS SDDCs from the AWS Partner Network Blog.

Limitations

• You can only configure 20 VMware Cloud accounts with public endpoints in one vRealize Automation. For more information about this, see Scalability and concurrency maximums in the VMware documentation.

Product versions

• vRealize Automation version 8.x or later
• VMware vRealize Identity Manager version 3.x or later
• VMware vRealize Suite Lifecycle Manager version 8.x or later

Architecture

The following diagram shows the vRealize Automation services that can use infrastructure from both on-premises and VMware Cloud on AWS environments.

![vRealize Automation services diagram]

VMware Cloud Assembly components

VMware Cloud Assembly is a core component of vRealize Automation and you can use it to deploy and provision VMs and compute resources. The following table describes VMware Cloud Assembly components that must be configured for provisioning VMs on VMware Cloud on AWS.

<table>
<thead>
<tr>
<th>Components</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Account</td>
<td>The Cloud Account provides connection details (for example, server name, user name and password, access key, and API token). VMware Cloud Assembly uses the Cloud Account to collect an inventory of your resources.</td>
</tr>
</tbody>
</table>
Cloud zones

Cloud zones identify resource boundaries in the Cloud Account (for example, AWS Regions and the cloud SDDC). Cloud zones associate compute resources with the Cloud Assembly project.

Projects

A project is a logical entity that consists of users and resources such as cloud zones. It also consists of resource quotas and VM naming policies that are used when building the VM.

Flavor mappings

Flavor mapping provides information about the VM's capacity (for example, number of CPUs and amount of memory) that are used in the Cloud Template.

Image mappings

Image mapping maps the VMware vSphere VM template and Amazon Web Services (AWS) image that are used in the Cloud Template. For more information about this, see Learn more about image mappings in vRealize Automation in the VMware documentation.

Network profile

Network profile controls the placement decision to choose a network during VM provisioning.

Storage profile

Storage profile controls the placement decision to choose storage during VM provisioning.

Cloud Templates

VMware Cloud Templates are an important component of vRealize Automation because they define cloud infrastructure provisioning and orchestration. The Cloud Templates are specifications for the resources and include the resource type, resource properties, and input to be collected from users.

Tools

- VMware vRealize Automation – vRealize Automation is an infrastructure automation platform with event-driven state management and compliance. It is designed to help organizations control and secure self-service clouds, multi-cloud automation with governance, and DevOps-based infrastructure delivery.
- VMware Cloud on AWS – VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware.

Epics

Generate the API tokens

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate the API tokens from your VMware Cloud on AWS account.</td>
<td>1. Sign in to the VMware Cloud Console.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>2.</td>
<td>On the VMware Cloud Services toolbar, choose <strong>My Account</strong> and then choose <strong>API Token</strong>.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Enter a name for your API token, provide the required lifespan, and define the scopes for the token.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Choose the <strong>Open ID</strong> check box and then choose <strong>Generate</strong>.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Record the API token's credentials.</td>
<td></td>
</tr>
</tbody>
</table>

For more information about this, see [How do I generate API tokens in the VMware documentation](#).

### Install vRealize Automation in your on-premises data center

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the required software.</td>
<td>Download the VMware vRealize Suite ISO file from the My VMware Portal. This package contains vRealize Suite Lifecycle Manager, VMware Identity Manager, and vRealize Automation.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

| Install the software. | Install the software and connect to your cloud SDCC by following the instructions from [Installing vRealize Suite Lifecycle Manager with Easy Installer for vRealize Automation and VMware Identity Manager](#) in the VMware documentation. | Cloud administrator, Cloud architect |

**Important:** Make sure that the following are available for your installation:

- The on-premises VMware vCenter Server setup and login credentials
- The network details for the vRealize Automation IP and subnet
- The vRealize Automation license key
## Connect VMware Cloud on AWS with VMware Cloud Assembly

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Configure your Cloud Accounts.** | 1. On the VMware Cloud Console, open the **Infrastructure** tab, choose **Manage – Cloud Accounts**, and then choose **Add Cloud Accounts**.  
2. Choose **VMware Cloud on AWS** as the type.  
3. Paste the API token information that you recorded earlier. This populates all available cloud SDDCs in your VMware Cloud on AWS organization.  
4. Choose the required cloud SDCC and then provide the vCenter user name and password for the SDDC.  
5. After you are successfully authenticated, you can view the integrated VMware Cloud on AWS account with an **OK** status.  

For more information about this, see Create a VMware Cloud on AWS cloud account in vRealize Automation in the VMware documentation. | Cloud architect, Cloud administrator |
| **Configure the project.**     | 1. On the VMware Cloud Console, open the **Projects** tab and then choose **New project**.  
2. Enter the name of your project.  
3. Open the **Cloud Zones** tab and choose **default VMware Cloud on AWS Cloud Account**. | Cloud administrator |
| **Configure cloud zone.**      | 1. On the VMware Cloud Console, open **Cloud Zones** and choose the cloud zone for your SDDC data center.  
2. By default, `cloudadmin@vmc.local` (this is the default local user ID for the cloud SDDC's vCenter) only has access to | Cloud administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>provision in the Compute-ResourcePool. 3. Open the <strong>Compute</strong> tab under <strong>Cloud Zones</strong> and then choose <strong>Compute-ResourcePool</strong>.</td>
<td></td>
</tr>
<tr>
<td>Configure flavor mapping.</td>
<td>1. Open the <strong>Flavor Mappings</strong> tab and create a new flavor mapping. 2. Enter the flavor name, choose the VMware Cloud on AWS account, and then provide the number of vCPUs and amount of memory.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Configure image mapping.</td>
<td>1. Open <strong>Image Mappings</strong> and create a new image mapping. 2. Enter the image name. 3. Choose the VMware Cloud on AWS account and provide the Cloud Account templates that are required.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Configure network profile.</td>
<td>1. Open <strong>Network Profile</strong> and create a new network profile. 2. Enter the network profile name. 3. Open the <strong>Network</strong> tab and choose the existing network that you want to use for provisioning.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Configure storage profile.</td>
<td>1. Open <strong>Storage Profile</strong> and choose <strong>New Storage Profile</strong>. 2. Enter the storage profile's name. 3. In the <strong>Policies</strong> section, create a new policy. 4. Choose <strong>Workload Datastore</strong>. By default <code>cloudadmin@vmc.local</code> only has access to provision in the workload's datastore.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Create the Cloud Template.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Open the Design tab, choose <strong>Cloud Templates</strong>, and then choose <strong>New From</strong> and <strong>Blank Canvas</strong>.</td>
</tr>
<tr>
<td>2. Provide the name and description of the Cloud Template.</td>
</tr>
<tr>
<td>3. Choose the project that you created earlier.</td>
</tr>
<tr>
<td>4. From the Cloud Template resources design page, drag components into the blank canvas according to your requirements.</td>
</tr>
<tr>
<td>5. Choose <strong>Test</strong> to test the template and fix any issues.</td>
</tr>
<tr>
<td>6. Choose <strong>Deployment</strong> and provide the deployment name to deploy the VMs.</td>
</tr>
</tbody>
</table>

For more information about this, see [Create a basic cloud template in the VMware documentation](https://www.vmware.com/support/pubs/vmware-cloud-on-aws.html).

### Skills required

- Cloud administrator

### Related resources

- Connect vRealize Automation version 8.x to your SDDC: [VMware Cloud on AWS](https://www.vmware.com/support/pubs/vmware-cloud-on-aws.html)
- Deploy a SDDC from the VMC Console
- AWS Direct Connect Integration with VMware Cloud on AWS

### Deploy a VMware SDDC on AWS by using VMware Cloud on AWS

*Created by Deepak Kumar (AWS)*

**Environment:** Production  
**Technologies:** Hybrid cloud; Infrastructure  
**Workload:** All other workloads  
**AWS services:** Amazon VPC

### Summary

This pattern describes how to create a VMware-based Software-Defined Data Center (SDDC) that’s hosted in the Amazon Web Services (AWS) Cloud. You can deploy an SDDC to migrate your VMware
vSphere-based workloads to the AWS Cloud and take advantage of AWS services while you use your existing VMware tools and skills. You can use this SDDC to run your production applications across VMware vSphere-based private, public, and hybrid cloud environments, with optimized access to AWS services. For example, you can use the SDDC as a secondary site for disaster recovery or to extend your data center to different geographical locations.

VMware Cloud on AWS is a pay-as-you-go (on-demand) service that enables enterprises of all sizes to run workloads across VMware vSphere-based cloud environments by using a wide range of AWS services. You can start with a minimum of 2 hosts per SDDC cluster and scale up to 16 hosts per cluster in your production environment. For more information, see the VMware Cloud on AWS website. To learn more about SDDCs, see About Software-Defined Data Centers in the VMware documentation.

Prerequisites and limitations

Prerequisites

- Sign up for a MyVMware account and fill out all fields.
- Sign up for an AWS account. For instructions, see the AWS Knowledge Center.
- Sign up for a MyVMware Cloud on AWS account. An activation link is sent to the email address you specify when you sign up.

Limitations

- See VMware Cloud on AWS configuration limits pages on the VMware website.

Product versions

- See VMware Cloud on AWS Release Notes in the VMware documentation.

Architecture

Target technology stack

The following diagram shows the VMware software stack, including vSphere, vCenter, vSAN, and NSX-T, running on AWS bare-metal dedicated infrastructure. You can manage your VMware-based resources and tools on AWS with seamless integration with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon Redshift, AWS Direct Connect, Amazon Relational Database Service (Amazon RDS), and Amazon DynamoDB.

![Architecture Diagram](image-url)
The basic entity of VMware Cloud on AWS is an SDDC, which includes the following components:

- **Compute**: The compute component is the lowest layer of the VMware Cloud on AWS SDDC. VMware Cloud on AWS runs on Amazon EC2 bare metal instance types. These include i3.metal and i3en.metal and provide direct access to physical resources such as processors and memory.

- **Storage**: SDDC clusters support VMware vSAN with an all-flash configuration for storage using non-volatile memory express (NVMe) flash storage, which provides fast and high-performance storage.

- **Networking**: Networking capabilities and policies are managed by using VMware NSX-T in the SDDC cluster. Multi-tier virtual networks are created in the SDDC cluster to separate network resources from physical equipment. This enables VMware Cloud on AWS users to create logical, software-defined networks.

**Tools**

- **VMware Cloud on AWS** – VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware.

**Epics**

Create a VPC and subnet in your AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to your AWS account.</td>
<td>Sign in to your AWS account with an IAM user that has administrator permissions.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create a new VPC.</td>
<td>In this step, you define a virtual private cloud (VPC) that links to the SDDC. If you already have a VPC you want to use for the SDDC, skip this step.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>1. Choose the AWS Region to deploy your VMware Cloud on AWS SDDC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Open the Amazon VPC console at <a href="https://console.aws.amazon.com/vpc/">https://console.aws.amazon.com/vpc/</a>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. In the navigation pane, choose Your VPCs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Choose Create VPC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Specify VPC settings such as the VPC name tag, IPv4 CIDR block, Tenancy (keep as Default), and then choose Create VPC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. When the VPC has been created, choose Close.</td>
<td></td>
</tr>
</tbody>
</table>
### Create a private subnet.

You will now create a private subnet for the elastic network interface (ENI) for each Availability Zone. We recommend that you use a subnet without an internet gateway attached.

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Choose Create Subnet.
4. On the Create Subnet page, choose the VPC that you created earlier.
5. Complete the settings for the subnet, including a subnet name, Availability Zone, and IPv4 CIDR block.
6. Choose Create Subnet.

Repeat these steps to create subnets for each Availability Zone in the Region.

### Activate VMware Cloud on AWS

#### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate the service.</td>
<td>When you sign up for a MyVMware account, VMware sends you a welcome email and activation link to the email address you specified.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

1. Open the **Activate Service** link from the welcome email in your browser.
2. Log in with MyVMware credentials.
3. Review and accept the terms and conditions for the use of services.
## AWS Prescriptive Guidance Patterns

### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>Complete the account activation process. You will be redirected to the VMware Cloud on AWS console. (Note: VMware Cloud on AWS accounts are based on an organization, which represents a group or line of business subscribed to the account. This organization has no relationship to AWS Organizations.)</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>5.</td>
<td>On the <strong>Select or Create Organization</strong> page, create an organization that is linked to the MyVMware account.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>6.</td>
<td>Enter the <strong>Organization Name</strong> and <strong>Address</strong> for logical distinction.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>7.</td>
<td>Select <strong>Create Organization</strong> to complete the process.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

For more information about this process, see **SDDC Deployment and Best Practices Guide on AWS** in the AWS documentation.

### Assign IAM roles.

When the organization has been created, assign privileged access to specific users to access the Cloud Services and SDDC console, SDDC, and NSX components. For instructions, see **Assign a VMC Service Role to an Organization Member** in the VMware documentation.

There are two types of organization roles:

- Organization owners can add, remove, and modify users and access all cloud resources.
- Organization members can access cloud resources only.

### Deploy an SDDC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy an SDDC in your VMware Cloud on AWS account.</td>
<td>1. Log in to the VMC console at <a href="https://vmc.vmware.com">https://vmc.vmware.com</a>.</td>
<td>Cloud administrator, Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>2.</td>
<td>Choose <strong>VMware Cloud on AWS Service</strong> from the available services.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Choose <strong>Create SDDC</strong>.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Enter SDDC properties such as <strong>AWS Region</strong>, <strong>Deployment</strong> (Single Host, Multi-Host, or Stretched Cluster), <strong>Host Type</strong>, <strong>SDDC Name</strong>, <strong>Number of Hosts</strong>, <strong>Host Capacity</strong>, and <strong>Total Capacity</strong>, and then choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Connect to your AWS account, and then choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Select your previously configured VPC and subnet, and then choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Enter the management subnet CIDR block for the SDDC, and then choose <strong>NEXT</strong>. For more information, see <strong>Selecting IP Subnets and Connectivity for your SDDC</strong> on the VMware Cloud Blog.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Select the two check boxes to acknowledge that you take responsibility for the costs for deploying an SDDC, and then choose <strong>Deploy SDDC</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

You’ll be charged when you choose **Deploy SDDC**. You won’t be able to pause or cancel the deployment process, which takes some time to complete.

For more information about creating an SDDC, see **Deploy an SDDC from the VMC Console** in the VMware documentation.

**Related resources**

- **Deploying and Managing a Software-Defined Data Center** *(VMware documentation)*
- **VMware Cloud on AWS features** *(AWS website)*
- **Accelerate Cloud Migration and Modernization with VMware Cloud on AWS** *(video)*
Migrate VMs to VMware Cloud on AWS by using HCX OS Assisted Migration

Summary

This pattern describes how to migrate a virtual machine (VM) from a non-vSphere environment to VMware Cloud on Amazon Web Services (AWS) by using OS Assisted Migration (OSAM).

OSAM is part of VMware Hybrid Cloud Extension (HCX), which is included with VMware Cloud on AWS. You can use OSAM to migrate a non-vSphere environment such as VMware KVM or Hyper-V to VMware Cloud on AWS. OSAM uses Sentinel software, which you install on a Windows or Linux guest VM to assist in replicating the VM from your on-premises environment to a Software-Defined Data Center (SDDC) on VMware Cloud on AWS.

This pattern explains how to enable OSAM, install Sentinel software on a Windows VM, connect and register with an HCX Sentinel Gateway (SGW) appliance at the source site, and establish a forwarding connection with an HCX Sentinel Data Receiver (SDR) appliance at the destination site to initiate migration.

For more information about OSAM, see the VMware documentation.

Prerequisites and limitations

Prerequisites

- Install HCX in your source and target environments. For HCX prerequisites, see Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX in the AWS Prescriptive Guidance documentation.
- For OSAM prerequisites, see the installation checklist in the VMware documentation.
- For OSAM port information, see VMware HCX port requirements on the VMware Ports and Protocols website.

Limitations

- VMware HCX 4.2.0 Configuration Limits
- Considerations for OSAM Deployment
- Supported Guest Operating Systems
- Guest Operating System Considerations

Product versions

575
• VMware HCX 4.2.0
• VMware SDDC 1.12

Architecture

The following diagram shows how HCX OSAM works with the Sentinel software to replicate non-vSphere VMs from your on-premises environment to VMware Cloud on AWS.

OSAM consists of three components:

• The Sentinel Gateway (SGW) appliance, which is used to connect and forward workloads and applications in the source VMware-based environment
• The Sentinel Data Receiver (SDR), which is used in the destination VMware Cloud on AWS environment to receive migrated workloads from the source
• Sentinel software, which must be installed on each guest VM that you want to migrate

OSAM uses the Sentinel software that is installed on Windows or Linux guest VMs to assist in replicating a VM from on premises to a VMware SDDC. The Sentinel software that you install on guest VMs collects the system configurations from the guest VM and assists with the data replication. This information is also used to create the inventory of guest VMs for migration and helps prepare the disks on the replica VM for replication and migration purposes.

Tools

• VMware HCX 4.2.0
• VMware Cloud on AWS SDDC
## Epics

### Configure HCX

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy HCX Cloud and HCX Connector.</td>
<td>Follow the instructions in [HCX Connector and HCX Cloud Installations](VMware documentation)</td>
<td>Cloud administrator, Systems administrator</td>
</tr>
</tbody>
</table>

### Configure OSAM and migrate VMs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install HCX Sentinel.</td>
<td>To install Sentinel on Linux: 1. In the vCenter Server for the HCX Connector, choose Interconnect, Multi-Site Service Mesh, Sentinel Management. 2. Choose Download Linux Bundle. 3. Install the Sentinel agent on a Linux machine. For more information, see [Downloading and Installing HCX Sentinel Agent software](VMware documentation)</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Migrate VMs.</td>
<td>To migrate your VMs in groups (called mobility groups), follow these steps: 1. In the vSphere Client, from the HCX plug-in, choose Services, Migration. 2. Choose Migrate. 3. Choose Non vSphere Inventory, Remote connections. This will show the list VMs that you installed HCX Sentinel on. 4. For Group name, enter the name of the mobility group you want to create for the VMs. 5. Choose the VMs you want to migrate, and then choose Add to add them to the mobility group.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
6. For each VM:
   a. Select the destination compute container.
   b. Select the destination storage.
   c. Select the migration profile.
   d. Select the destination folder.

7. To start the migration process, choose Go.

HCX validates your VM selections before migration starts.

For more information, see Migrating Virtual Machines with Mobility Groups and Monitoring and Estimating Migration with Mobility Groups in the VMware documentation.

---

### Related resources

VMware documentation:

- VMware HCX User Guide
- Install Checklist B - HCX with a VMC SDDC Destination Environment
- VMware HCX in the VMware Cloud on AWS
- HCX OS Assisted Migration for VMware Cloud on AWS
- VMware HCX 4.2.1 Release Notes

### More patterns

- Automate SAML 2.0 federation for AWS multi-account environments that use Azure AD (p. 1583)
- Automate the setup of inter-Region peering with AWS Transit Gateway (p. 1514)
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK (p. 1459)
- Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator (p. 286)
- Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS (p. 1538)
- Use Serverspec for test-driven development of infrastructure code (p. 540)
**Infrastructure**

**Topics**
- Check EC2 instances for mandatory tags at launch (p. 579)
- Deploy Sophos web proxy UTM and Outbound Gateway on AWS (p. 583)
- Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing (p. 587)
- Ensure that Amazon EC2 instances launch only in approved AWS Regions (p. 593)
- Mainframe modernization: DevOps on AWS with Micro Focus (p. 597)
- Set up integrated DNS resolution for hybrid networks in Amazon Route 53 (p. 613)
- Use consistent Availability Zones in VPCs across different AWS accounts (p. 617)
- More patterns (p. 622)

**Check EC2 instances for mandatory tags at launch**

*Created by Susanne Kangnoh (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technologies:</strong></td>
<td>Infrastructure; Management &amp; governance; Security, identity, compliance; Cloud-native</td>
</tr>
<tr>
<td><strong>AWS services:</strong></td>
<td>Amazon EC2; AWS CloudTrail; Amazon CloudWatch; Amazon SNS</td>
</tr>
</tbody>
</table>

**Summary**

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

You can use tagging to categorize your AWS resources in different ways. EC2 instance tagging is useful when you have many resources in your account and you want to quickly identify a specific resource based on the tags. You can assign custom metadata to your EC2 instances by using tags. A tag consists of a user-defined key and value. We recommend that you create a consistent set of tags to meet your organization's requirements.

This pattern provides an AWS CloudFormation template to help you monitor EC2 instances for specific tags. The template creates an Amazon CloudWatch Events event that watches for the AWS CloudTrail **TagResource** or **UntagResource** events, to detect new EC2 instance tagging or tag removal. If a predefined tag is missing, it calls an AWS Lambda function, which sends out a violation message to an email address that you provide, by using Amazon Simple Notification Service (Amazon SNS).

**Prerequisites and limitations**

**Prerequisites**
- An active AWS account.
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the provided Lambda code.
- An email address where you would like to receive violation notifications.

**Limitations**

- This solution supports CloudTrail **TagResource** or **UntagResource** events. It does not create notifications for any other events.
- This solution checks only for tag keys. It does not monitor key values.

**Architecture**

**Workflow architecture**

**Automation and scale**

- You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run the template only once in each Region or account.

**Tools**

**AWS services**

- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.
- **AWS CloudTrail** – CloudTrail is an AWS service that helps you with governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or AWS service are recorded as events in CloudTrail.
- **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- **AWS Lambda** – Lambda is a compute service that supports running code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud.
Code

This pattern includes an attachment with two files:

- *index.zip* is a compressed file that includes the Lambda code for this pattern.
- *ec2-require-tags.yaml* is a CloudFormation template that deploys the Lambda code.

See the *Epics* section for information about how to use these files.

Epics

Deploy the Lambda code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the code to an S3 bucket.</td>
<td>Create a new S3 bucket or use an existing S3 bucket to upload the attached <em>index.zip</em> file (Lambda code). This bucket must be in the same AWS Region as the resources (EC2 instances) that you want to monitor.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the <em>ec2-require-tags.yaml</em> file that's provided in the attachment. In the next epic, provide values for the template parameters.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Complete the parameters in the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the EC2 instances that you want to monitor.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, <em>index.zip</em> or <em>controls/index.zip</em>).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address where you want to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Define a logging level.</td>
<td>Specify the logging level and verbosity. Info designates detailed informational messages on the application’s progress and should be used only for debugging. Error designates error events that could still allow the application to continue running. Warning designates potentially harmful situations.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Enter the required tag keys.</td>
<td>Enter the tag keys that you want to check for. If you want to specify multiple keys, separate them with commas, without spaces. (For example, ApplicationId,CreatedBy,Environment,Organization searches for four keys.) The CloudWatch Events event searches for these tag keys and sends a notification if they are not found.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the email subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating a bucket (Amazon S3 documentation)
- Uploading objects (Amazon S3 documentation)
- Tag your Amazon EC2 resources (Amazon EC2 documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (Amazon CloudWatch documentation)

Attachments

attachment.zip
Deploy Sophos web proxy UTM and Outbound Gateway on AWS

Summary

Amazon Elastic Compute Cloud (Amazon EC2) instances often require access to external resources, such as software repositories and web services. Many organizations require restricting internet connections to authorized websites. Web filtering proxies are commonly used to enforce web policies for internet access.

This pattern uses the Sophos Unified Threat Management (UTM) virtual appliance, which is available in AWS Marketplace, to provide a transparent outbound proxy for EC2 instances. Alternative web proxy solutions may use open-source solutions such as Squid or Apache Traffic Server, which are beyond the scope of this pattern.

Sophos UTM provides multiple security functions, including firewall, intrusion prevention system (IPS), virtual private network (VPN), and web filtering. Sophos Outbound Gateway provides a distributed, fault-tolerant architecture to provide visibility, policy enforcement, and elastic scalability to outbound web traffic.

This pattern provides guidance for one use case: allowing AWS API calls from a virtual private cloud (VPC). This pattern does not cover general installation and software configuration tasks for Sophos UTM. For general guidance and best practices, consult the Sophos UTM on AWS Administration Guide.

This pattern uses the following Sophos software:

- Sophos UTM 9 virtual appliance – Sophos UTM is a security platform that helps you secure your infrastructure in AWS. Sophos UTM provides multiple security tools, such as Next-Gen Firewall (NGFW), Web Application Firewall (WAF), Intrusion Prevention System (IPS), and Advanced Threat Protection (ATP).
- Sophos UTM Controller (Queen) – The Controller is a UTM instance that provides administrative control and configuration management for UTM Workers.
- Sophos UTM Workers – The UTM Workers terminate the Generic Routing Encapsulation (GRE) tunnels from the Outbound Gateways and proxy the traffic to the destination based on the policy configured within the Controller.
- Sophos Outbound Gateway (OGW) on AWS – The OGW is an instance that resides within an Availability Zone where clients need to connect out through the proxy.

This pattern leverages the AWS Quick Start that was developed by Sophos in collaboration with AWS. Sophos is an AWS APN Partner.

Prerequisites and limitations

Prerequisites
• Determine the licensing model you want to use for Sophos UTM. Available options are hourly and Bring Your Own License Model (BYOL). If you’re using the BYOL option, you’ll need your license file.

• Confirm that you have an active AWS account, and that your account limits allow you to provision two VPCs and one Elastic IP address.

• Identify the CIDR ranges you want to use for the proxy and application VPCs and subnets.

• In order to administer the Sophos UTM, you will need access to TCP port 4444 from your browser.

Outbound proxy solution considerations

Although outbound proxy solutions are widely used, there are implications that you should consider before you implement proxies in your architecture. The Sophos web filtering engine operates in transparent or standard mode. Regardless of the deployment mode, your client applications either need to support the Server Name Indicator (SNI) specification to securely access Secure Sockets Layer / Transport Layer Security (SSL/TLS) endpoints, or you will need to employ SSL decryption and distribute the signing certificate authority (CA) to your clients, as discussed in the Sophos Knowledge Base article. Additionally in standard mode, your client applications will need to be “proxy aware” and explicitly direct requests through the Sophos Outbound Gateway. Sophos provides additional options to control traffic, such as the ability to bypass filtering by source or destination IP. To learn more about this and other considerations, see the Sophos UTM Administration Guide.

Architecture

Source technology stack

• On-premises Sophos web proxy UTM

Target technology stack

• Sophos web proxy UTM on AWS

Source and target architecture

This is a simple software relocation from an on-premises data center host to an EC2 instance. The following diagram shows how the Sophos Outbound Gateway uses GRE to tunnel outbound network traffic between VPCs.
The following diagram provides a detailed view of the architecture on AWS. This pattern uses a highly available architecture that includes a Sophos UTM Controller, Sophos UTM Workers, and Sophos Outbound Gateways on AWS. In total, it deploys seven instances, including one Controller, two Workers in an Auto Scaling group, two Outbound Gateways, one bastion host, and one client test EC2 instance. To centralize the proxy service for the clients, the Controller and Workers are deployed into a dedicated VPC.

This pattern utilizes one additional application VPC for proxy clients. The Outbound Gateway on AWS is also deployed into the application VPC to support the connections from the clients. Finally, the bastion host and tester instances are deployed into the application VPC. You can use these instances to test and become familiar with the outbound web proxy functionality.
Tools

- Quick Start: Sophos outbound web proxy on AWS
- Sophos UTM console

Epics

Assess and deploy the Quick Start

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Quick Start, if it meets your needs.</td>
<td>See the Quick Start deployment guide (see the References and Help section) for any pre-deployment instructions, and</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>then launch the Quick Start from the link provided.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customize and launch the Quick Start, if you have additional requirements.</td>
<td>Download the AWS CloudFormation templates from the GitHub repository (see the References and Help section), modify them to meet your needs, and launch the customized templates.</td>
<td></td>
</tr>
<tr>
<td>Validate the deployment.</td>
<td>See the Quick Start deployment guide for any post-deployment and testing instructions.</td>
<td></td>
</tr>
</tbody>
</table>

## Related resources

**Sophos**
- Sophos UTM on AWS Administration Guide
- Sophos Knowledge Base
- Sophos Technical Support

**AWS Quick Starts**
- Quick Start: Sophos Outbound Web Proxy on AWS (data sheet and deployment guide)
- Quick Start: Sophos Outbound Web Proxy on AWS (source code in GitHub repository)
- AWS Quick Start catalog

### Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing

*Created by Dipin Jain (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Source: On-premises VM</th>
<th>Target: Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Workload: Open-source</td>
<td>Technologies: Infrastructure; Databases; Migration</td>
</tr>
<tr>
<td>AWS services:</td>
<td></td>
<td>Amazon EC2</td>
</tr>
</tbody>
</table>

### Summary

The private IP of an Amazon Elastic Compute Cloud (Amazon EC2) instance is retained throughout its lifecycle. However, the private IP might change during a planned or unplanned system crash; for example, during an Amazon Machine Image (AMI) upgrade. In some scenarios, retaining a private static
IP can enhance the performance and recovery time of workloads. For example, using a static IP for an Apache Cassandra seed node prevents the cluster from incurring a rebalancing overhead.

This pattern describes how to attach a secondary elastic network interface to EC2 instances to keep the IP static during rehosting. The pattern focuses on Cassandra clusters, but you can use this implementation for any architecture that benefits from private static IPs.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Service (AWS) account

Product versions

- DataStax version 5.11.1
- Operating system: Ubuntu 16.04.6 LTS

Architecture

Source architecture

The source could be a Cassandra cluster on an on-premises virtual machine (VM) or on EC2 instances in the AWS Cloud. The following diagram illustrates the second scenario. This example includes four cluster nodes: three seed nodes and one management node. In the source architecture, each node has a single network interface attached.
Target architecture

The destination cluster is hosted on EC2 instances with a secondary elastic network interface attached to each node, as illustrated in the following diagram.
Automation and scale

You can also automate attaching a second elastic network interface to an EC2 Auto Scaling group, as described in an AWS Knowledge Center video.

Epics

Configure a Cassandra cluster on Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch EC2 nodes to host a Cassandra cluster.</td>
<td>On the Amazon EC2 console, launch four EC2 instances for your Ubuntu nodes in your AWS account. Three (seed) nodes are used for the Cassandra cluster, and the fourth node acts as a cluster management node where you will install DataStax Enterprise (DSE) OpsCenter. For</td>
<td>Cloud engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>instructions, see the Amazon EC2 documentation.</td>
<td></td>
</tr>
<tr>
<td>Confirm node communications.</td>
<td>Make sure that the four nodes can communicate with one another over the database and cluster management ports.</td>
<td>Network engineer</td>
</tr>
<tr>
<td>Install DSE OpsCenter on the management node.</td>
<td>Install DSE OpsCenter 6.1 from the Debian package on the management node. For instructions, see the DataStax documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create a secondary network interface.</td>
<td>Cassandra generates a universal unique identifier (UUID) for each node based on the IP address of the EC2 instance for that node. This UUID is used for distributing virtual nodes (vnodes) on the ring. When Cassandra is deployed on EC2 instances, IP addresses are assigned automatically to the instances as they are created. In the event of a planned or unplanned outage, the IP address for the new EC2 instance changes, the data distribution changes, and the entire ring has to be rebalanced. This is not desirable. To preserve the assigned IP address, use a secondary elastic network interface with a fixed IP address.</td>
<td>Cloud Engineer</td>
</tr>
<tr>
<td></td>
<td>1. On the Amazon EC2 console, choose Network Interfaces, Create network interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. For Subnet, select the subnet that you created the EC2 instance in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. For Private IPv4 address, choose Auto-assign.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. For Security groups, select a security group, and then choose Create network interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about creating a network interface, see the Amazon EC2 documentation.</td>
<td></td>
</tr>
</tbody>
</table>
### Attach the secondary network interface to cluster nodes

**Task:** Attach the secondary network interface to cluster nodes.

**Description:**
1. On the Amazon EC2 console, choose **Instances**.
2. Select the checkbox for the EC2 instance you created earlier.
3. Choose **Actions, Networking, Attach network interface**.
4. Select the network interface you created in the previous step, and then choose **Attach**.

For more information about attaching a network interface, see the Amazon EC2 documentation.

**Skills required:** Cloud engineer

### Add routes in Amazon EC2 to address asymmetric routing

**Task:** Add routes in Amazon EC2 to address asymmetric routing.

**Description:** When you attach the second network interface, the network will very likely perform asymmetric routing. To avoid this, you can add routes for the new network interfaces.

For an in-depth explanation and remediation of asymmetric routing, see the AWS Knowledge Center video or Overcoming Asymmetric Routing on Multi-Home Servers (article in *Linux Journal* by Patrick McManus, April 5, 2004).

**Skills required:** Network engineer

### Update DNS entries to point to the secondary network interface IP

**Task:** Update DNS entries to point to the secondary network interface IP.

**Description:** Point the fully qualified domain name (FQDN) of the node to the IP of the secondary network interface.

**Skills required:** Network engineer

### Install and configure the Cassandra cluster by using DSE OpsCenter

**Task:** Install and configure the Cassandra cluster by using DSE OpsCenter.

**Description:** When the cluster nodes are ready with the secondary network interfaces, you can install and configure the Cassandra cluster.

**Skills required:** DBA

---

## Recover cluster from node failure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AMI for the cluster seed node.</td>
<td>Make a backup of the nodes so you can restore them with database binaries in case of node failure. For instructions,</td>
<td>Backup administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Recover from node failure.</td>
<td>Replace the failed node with a new EC2 instance launched from the AMI, and attach the secondary network interface of the failed node.</td>
<td>Backup administrator</td>
</tr>
<tr>
<td>Verify that the Cassandra cluster is healthy.</td>
<td>When the replacement node is up, verify cluster health in DSE OpsCenter.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Related resources**

- Installing DSE OpsCenter 6.1 from the Debian package (DataStax documentation)
- How to make a secondary network interface work in an Ubuntu EC2 instance (AWS Knowledge Center video)
- Best Practices for Running Apache Cassandra on Amazon EC2 (AWS blog post)

**Ensure that Amazon EC2 instances launch only in approved AWS Regions**

*Created by Gaurav Gupta (AWS)*

**Environment:** Production  
**Technologies:** Security, identity, compliance; Infrastructure  
**AWS services:** Amazon EC2

**Summary**

This pattern provides an AWS CloudFormation template to ensure that an Amazon Elastic Compute Cloud (Amazon EC2) instance launches only in AWS Regions that you specify. This check is useful if you have a security or compliance requirement for deploying your EC2 instances in specific Regions.

Amazon CloudWatch Events initiates this check when AWS CloudTrail logs Amazon EC2 API calls based on the `ModifyInstanceAttribute` and `RunInstances` actions. This trigger calls an AWS Lambda function, which uses a CloudWatch Events event and compares the deployment Regions with the Regions that you specified in AWS CloudFormation.

If the EC2 instance launches in an unauthorized Region, the Lambda function initiates an Amazon Simple Notification Service (Amazon SNS) email notification that contains the EC2 instance ID, account number, Region, and the Amazon Resource Name (ARN) of the Lambda function.

**Prerequisites and limitations**

**Prerequisites**
• An Amazon Simple Storage Service (Amazon S3) bucket to upload the Lambda code
• An active email address to receive Amazon SNS notifications

Limitations

• A CloudWatch Events rule must be created for each Region you want to monitor. This rule must also have an associated Lambda function for that Region.
• This process doesn't check for existing EC2 instances unless you change their attributes, which then calls ModifyInstanceAttribute.

Automation and scale

• If you use AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in Regions that you want to monitor.

Architecture

Target technology stack

• AWS Lambda
• Amazon SNS
• CloudWatch Events

Target architecture

Tools

Tools

• AWS CloudFormation - Allows you to use programming languages or a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all AWS Regions and accounts.
• Amazon CloudWatch Events - Provides you with data and actionable insights to monitor your applications, respond to systemwide performance changes, optimize resource utilization, and get a unified view of operational health.
• AWS Lambda - Lets you run code without provisioning or managing servers.
• Amazon SNS - Enables applications, end-users, and devices to instantly send and receive notifications from the cloud.
• Amazon S3 - A highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
Epics

Define the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>Open the Amazon S3 console and choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. Your S3 bucket needs to be in the same Region as the EC2 instance being evaluated. The S3 bucket's name cannot contain leading slashes.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Upload the Lambda code to the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code to the S3 bucket.</td>
<td>Upload the Lambda code provided in the &quot;Attachments&quot; section to the S3 bucket. The S3 bucket must be in the same Region as the EC2 instance being evaluated.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>Open the AWS CloudFormation console in the same Region as your S3 bucket and deploy the template. For more information about deploying AWS CloudFormation templates, see the “Related resources” section.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Give your S3 bucket a unique name. An Amazon S3 bucket name is globally unique, and the namespace is shared by all AWS accounts.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket (for example: <code>&lt;filename&gt;.zip</code>). Do not use leading slashes.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define logging level.</td>
<td>Define the logging level and frequency for your Lambda function. “Info” designates detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define authorized Regions.</td>
<td>Provide a comma-separated list of authorized Regions.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

#### Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email to the email address provided. You must confirm this email subscription to receive violation notifications.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

### Related resources

- Deploying AWS CloudFormation templates

### Attachments

attachment.zip
Mainframe modernization: DevOps on AWS with Micro Focus

Created by Kevin Yung (AWS)

<table>
<thead>
<tr>
<th>Source: IBM z/OS Mainframe</th>
<th>Target: AWS</th>
<th>R Type: N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: PoC or pilot</td>
<td>Technologies: DevOps; Infrastructure</td>
<td>AWS services: Amazon EC2; AWS CloudFormation; AWS CodeBuild; AWS CodeCommit; AWS CodeDeploy; AWS Systems Manager; AWS CodePipeline</td>
</tr>
</tbody>
</table>

Summary

Customer challenges

Organizations that run core applications on mainframe hardware usually encounter a few challenges when the hardware needs to scale up to meet the demands of digital innovations. These challenges include the following constraints.

- Mainframe development and test environments are unable to scale due to the inflexibility of mainframe hardware components and the high cost of changing.
- Mainframe development is facing skill shortages, because new developers are not familiar and not interested in the traditional mainframe development tools. Modern technology such as containers, continuous integration/continuous delivery (CI/CD) pipelines, and modern test frameworks are not available in mainframe development.

Pattern outcomes

To address these challenges, Amazon Web Services (AWS) and Micro Focus, an AWS Partner Network (APN) Partner, have collaborated to create this pattern. The solution is designed to help you achieve the following outcomes.

- Improved developer productivity. Developers can be given new mainframe development instances within minutes.
- Use of the AWS Cloud to create new mainframe test environments with virtually unlimited capacity.
- Rapid provisioning of new mainframe CI/CD infrastructure. Provisioning on AWS can be completed within an hour by using AWS CloudFormation and AWS Systems Manager.
- Native use of AWS DevOps tools for mainframe development, including AWS CodeBuild, AWS CodeCommit, AWS CodePipeline, AWS CodeDeploy, and Amazon Elastic Container Registry (Amazon ECR).
- Transform traditional waterfall development to agile development in mainframe projects.

Technologies summary

In this pattern, the target stack contains the following components.
### Prerequisites and limitations

#### Prerequisites

<table>
<thead>
<tr>
<th>Logical components</th>
<th>Implementation solutions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source code repositories</td>
<td>Micro Focus AccuRev Server, CodeCommit, Amazon ECR</td>
<td>Source code management – The solution uses two types of source code. - Mainframe source code, for example COBOL, JCL, etc. - AWS infrastructure templates and automation scripts</td>
</tr>
<tr>
<td>Enterprise developer instances</td>
<td>Amazon Elastic Compute Cloud (Amazon EC2), Micro Focus Enterprise Developer for Eclipse</td>
<td>Both types of source code need version control, but they are managed in different SCMs. Source code deployed into mainframe or Micro Focus Enterprise Servers is managed in Micro Focus AccuRev Server. AWS templates and automation scripts are managed in CodeCommit. Amazon ECR is used for the Docker image repositories.</td>
</tr>
<tr>
<td>Micro Focus license management</td>
<td>Micro Focus License Manager</td>
<td>For centralized Micro Focus license management and governance, the solution uses Micro Focus License Manager to host the required license.</td>
</tr>
<tr>
<td>CI/CD pipelines</td>
<td>CodePipeline, CodeBuild, CodeDeploy, Micro Focus Enterprise Developer in a container, Micro Focus Enterprise Test Server in a container, Micro Focus Enterprise Server</td>
<td>Mainframe development teams need CI/CD pipelines to perform code compilation, integration tests, and regression tests. In AWS, CodePipeline and CodeBuild can work with Micro Focus Enterprise Developer and Enterprise Test Server in a container natively.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>py3270</td>
<td>py3270 is a Python interface to x3270, an IBM 3270 terminal emulator. It provides an API to a x3270 or s3270 subprocess.</td>
<td></td>
</tr>
<tr>
<td>x3270</td>
<td>x3270 is an IBM 3270 terminal emulator for the X Window System and Windows. This can be used by developer for unit testing locally.</td>
<td></td>
</tr>
<tr>
<td>Robot-Framework-3270-Library</td>
<td>Mainframe3270 is a library for Robot Framework based on py3270 project.</td>
<td></td>
</tr>
<tr>
<td>Micro Focus Verastream</td>
<td>Micro Focus Verastream is an integration platform that enables testing mainframe assets the way that mobile apps, web applications, and SOA web services are tested.</td>
<td></td>
</tr>
<tr>
<td>Micro Focus Unified Functional Testing (UFT) installer and license</td>
<td>Micro Focus Unified Functional Testing is software that provides functional and regression test automation for software applications and environments.</td>
<td></td>
</tr>
<tr>
<td>Micro Focus Enterprise Server installer and license</td>
<td>Enterprise Server provides the runtime environment for mainframe applications.</td>
<td></td>
</tr>
<tr>
<td>Micro Focus Enterprise Test Server installer and license</td>
<td>Micro Focus Enterprise Test Server is an IBM mainframe application test environment</td>
<td></td>
</tr>
<tr>
<td>Micro Focus AccuRev installer and license for Server, and Micro Focus AccuRev installer and license for Windows and Linux operating systems</td>
<td>AccuRev provides source code management (SCM). The AccuRev system is designed for use by a team of people who are developing a set of files.</td>
<td></td>
</tr>
<tr>
<td>Micro Focus Enterprise Developer for Eclipse installer, patch and license</td>
<td>Enterprise Developer provide mainframe developer a platform to develop and maintain the core mainframe online and batch applications.</td>
<td></td>
</tr>
</tbody>
</table>

**Limitations**

- Building a Windows Docker image is not supported in CodeBuild. This reported issue needs support from Windows Kernel/HCS and Docker teams. The work-around is to create a Docker image build runbook by using Systems Manager. This pattern uses the work-around to build Micro Focus Enterprise Developer for Eclipse and Micro Focus Enterprise Test Server Container images.
- Virtual private cloud (VPC) connectivity from CodeBuild is not supported in Windows yet, so the pattern does not use Micro Focus License Manager to manage licenses in Micro Focus Enterprise Developer and Micro Focus Enterprise Test Server containers.

**Product versions**

- Micro Focus Enterprise Developer 5.5 or later
- Micro Focus Enterprise Test Server 5.5 or later
- Micro Focus Enterprise Server 5.5 or later
- Micro Focus AccuRev 7.x or later
- Windows Docker base image for Micro Focus Enterprise Developer and Enterprise Test Server: microsoft/dotnet-framework-4.7.2-runtime
• Linux Docker base image for AccuRev client: `amazonlinux:2`

## Architecture

### Mainframe environment

In conventional mainframe development, the developers need to use mainframe hardware to develop and test programs. They face capacity limitations, for example restricted million instructions per second (MIPS) for the dev/test environment, and they must rely on the tools that are available on the mainframe computers.

In many organizations, mainframe development follows the waterfall development methodology, with teams relying on long cycles to release changes. These release cycles are usually longer than digital product development.

The following diagram shows multiple mainframe projects sharing mainframe hardware for their development. In mainframe hardware, it is expensive to scale out a development and test environment for more projects.
This pattern extends mainframe development to the AWS Cloud. First, it uses Micro Focus AccuRev SCM to host the mainframe source code on AWS. Then it makes Micro Focus Enterprise Developer and Micro Focus Enterprise Test Server available for building and testing the mainframe code on AWS.

The following sections describe the pattern's three major components.

1. SCM

In AWS, the pattern uses Micro Focus AccuRev to create a set of SCM workspaces and version control for the mainframe source code. Its stream-based architecture enables parallel mainframe development for
multiple teams. To merge a change, AccuRev uses the promote concept. To add that change to other workspaces, AccuRev uses the update concept.

At the project level, each team can create one or more streams in AccuRev to track project level changes. These are called project streams. These project streams are inherited from the same parent stream. The parent stream is used to merge the changes from different project streams.

Each project stream can promote code to AccuRev, and a promote post trigger is set up to initiate the AWS CI/CD pipeline. The successful build for a project stream change can be promoted to its parent stream for more regression tests.

Usually, the parent stream is called the system integration stream. When there is a promotion from a project stream to a system integration stream, a post promotion trigger initiates another CI/CD pipeline to run regression tests.

In addition to mainframe code, this pattern includes AWS CloudFormation templates, Systems Manager Automation documents, and scripts. Following infrastructure-as-code best practices, they are version-controlled in AWS CodeCommit.

If you need to synchronize mainframe code back to a mainframe environment for deployment, Micro Focus provides the Enterprise Sync solution, which synchronizes code from the AccuRev SCM back to the mainframe SCM.

2. Developer and test environments

In a large organization, scaling more than a hundred or even more than a thousand mainframe developers is challenging. To address this constraint, the pattern uses Amazon EC2 Windows instances for development. On the instances, Micro Focus Enterprise Developer for Eclipse tools are installed. The developer can perform all mainframe code test and debugging locally on the instance.

AWS Systems Manager State Manager and Automation documents are used to automate the developer instance provisioning. The average time to create a developer instance is within 15 minutes. The following software and configurations are prepared.

- AccuRev Windows client for checking out and committing source code into AccuRev
- Micro Focus Enterprise Developers for Eclipse tool, for writing, testing, and debugging mainframe code locally
- Open source testing frameworks Python behavior-driven development (BDD) test framework Behave, py3270, and the x3270 emulator for creating scripts to test applications
- A Docker developer tool for building the Enterprise Test Server Docker image and testing the application in the Enterprise Test Server Docker container

In the development cycle, developers use the EC2 instance to develop and test mainframe code locally. When the local changes are tested successfully, developers promote the change into the AccuRev server.

3. CI/CD pipelines

In the pattern, CI/CD pipelines are used for integration tests and regression tests before deployment to the production environment.

As explained in the SCM section, AccuRev uses two types of streams: a project stream and an integration stream. Each stream is hooked up with CI/CD pipelines. To perform the integration between the AccuRev server and AWS CodePipeline, the pattern uses AccuRev post promotion script to create an event to initiate CI/CD.

For example, when a developer promotes a change to a project stream in AccuRev, it initiates a post promotion script to run in AccuRev Server. Then the script uploads the metadata of the change into
an Amazon Simple Storage Service (Amazon S3) bucket to create an Amazon S3 event. This event will initiate a CodePipeline configured pipeline to run.

The same event-initiating mechanism is used for the integration stream and its associated pipelines.

In the CI/CD pipeline, CodePipeline uses CodeBuild with the Micro Focus AccuRev Linux client container to check out the latest code from the AccuRev streams. Then the pipeline starts CodeBuild to use the Micro Focus Enterprise Developer Windows container to compile the source code, and to use the Micro Focus Enterprise Test Server Windows container in CodeBuild to test mainframe applications.

The CI/CD pipelines are built using AWS CloudFormation templates, and the blueprint will be used for new projects. By using the templates, it takes less than an hour for a project to create a new CI/CD pipeline in AWS.

To scale your mainframe test capability on AWS, the pattern builds out the Micro Focus DevOps test suite, Micro Focus Verastream and Micro Focus UFT server. By using the modern DevOps tools, you can run as many tests on AWS as you need.

An example mainframe development environment with Micro Focus on AWS is shown in the following diagram.

**Target technology stack**

This section provides a closer look at the architecture of each component in the pattern.

1. **Source code repository – AccuRev SCM**

   Micro Focus AccuRev SCM is set up to manage mainframe source code versions. For high availability, AccuRev supports primary and replica modes. Operators can fail over to the replica when performing maintenance on the primary node.

   To speed up the response of the CI/CD pipeline, the pattern uses Amazon CloudWatch Events to detect source code changes and initiate the start of the pipeline.

   1. The CodePipeline is set up to use an Amazon S3 source.
   2. A CloudWatch Events rule is set up to capture S3 events from a source S3 bucket.
   3. The CloudWatch Events rule sets a target to the pipeline.
   4. AccuRev SCM is configured to run a post promotion script locally after promotion is complete.
5. AccuRev SCM generates an XML file that contains the metadata of the promotion, and the script uploads the XML file to the source S3 bucket.

6. After the upload, the source S3 bucket sends events to match the CloudWatch Events rule, and the CloudWatch Events rule initiates the CodePipeline to run.

When the pipeline runs, it kicks off a CodeBuild project to use an AccuRev Linux client container to check out the latest mainframe code from an associated AccuRev stream.

The following diagram shows an AccuRev Server setup.

---

2. **Enterprise Developer template**

The pattern uses Amazon EC2 templates to simplify creation of the developer instance. By using State Manager, it can apply software and license settings to EC2 instances consistently.

The Amazon EC2 template builds in its VPC context settings and default instance settings, and it follows enterprise tagging requirements. By using a template, a team can create their own new development instances.

When a developer instance starts, by associating with tags, Systems Manager uses State Manager to apply automation. The automation includes the following general steps.

1. Install Micro Focus Enterprise Developer software and install patches.
2. Install the Micro Focus AccuRev client for Windows.
3. Install the pre-configured script for developers to join the AccuRev stream. Initialize Eclipse workspaces.
4. Install development tools, including x3270, py3270, and Docker.
5. Configure license settings to point to a Micro Focus License Manager load balancer.

The following diagram shows an Enterprise developer instance created by the Amazon EC2 template, with software and configuration applied to the instance by State Manager. Enterprise developer instances connect to Micro Focus License Manager to activate their license.

3. CI/CD pipelines

As explained in AWS architecture section, in the pattern, there are project-level CI/CD pipelines and system integration pipelines. Each mainframe project team creates a pipeline or multiple CI/CD pipelines for building the programs that they are developing in a project. These project CI/CD pipelines check out source code from an associated AccuRev stream.

In a project team, developers promote their code in the associated AccuRev stream. Then the promotion initiates the project pipeline to build the code and run and integration tests.

Each project CI/CD pipeline uses CodeBuild projects with the Micro Focus Enterprise Developer tool Amazon ECR image and Micro Focus Enterprise Test Server tool Amazon ECR image.

CodePipeline and CodeBuild are used to create the CI/CDs pipelines. Because CodeBuild and CodePipeline have no upfront fees or commitments, you pay only for what you use. Compared to mainframe hardware, the AWS solution greatly reduces hardware provisioning lead time and lowers the cost of your testing environment.

In modern development, multiple test methodologies are used. For example, test-driven development (TDD), BDD, and Robot Framework. With this pattern, developers can use these modern tools for mainframe testing. For example, by using x3270, py3270 and the Behave python test tool, you can
define an online application's behavior. You can also use build mainframe 3270 robot framework in these CI/CD pipelines.

The following diagram shows the team stream CI/CD pipeline.

The following diagrams shows the project CI/CD test reports produced by CodePipeline in Mainframe3270 Robot Framework and Behave.
MicroFocus-Cicd-Master-PipelineRpaTestBankDemo-bankdemo-rpatest-report:d7248206-40f2-4098-9712-c33f158f7382

Summary

Pass rate
100%

Report duration
2.662 seconds

Created
16 days ago

Details

Test cases

<table>
<thead>
<tr>
<th>Test case</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>User login to demo</td>
<td>Succeeded</td>
</tr>
<tr>
<td>User Calculate home loan</td>
<td>Succeeded</td>
</tr>
<tr>
<td>User Disconnect Session</td>
<td>Succeeded</td>
</tr>
</tbody>
</table>

Test Report - Mainframe3270 Robot Framework
After project level tests are passed successfully, the tested code is manually promoted to the integration stream in AccuRev SCM. You can automate this step after the teams have a confidence on the tests coverage of their project pipeline.

When code is promoted, the system integration CI/CD pipeline checks out the merged code and performs regression tests. The merged code is promoted from all parallel project streams.

Depending on how fine grain the test environment are required, customers can have more system integration CI/CD pipelines in different environment, for example UAT, Pre-Production.

In the pattern, the tools used in the system integration pipeline are Micro Focus Enterprise Test Server, Micro Focus UFT Server, and Micro Focus Verastream. All these tools can be deployed into the Docker container and used with CodeBuild.

After successfully testing of the mainframe programs, the artifact is stored, with version control, in an S3 bucket.

The following diagram shows a system integration CI/CD pipeline.
After the artifact has been successfully tested in the system integration CI/CD pipelines, it can be promoted for production deployment.

If you need to deploy source code back to the mainframe, Micro Focus offers the Enterprise Sync solution to synchronize source code from AccuRev back to Mainframe Endeavour.

The following diagram shows a production CI/CD pipeline deploying the artifact into Micro Focus Enterprise Servers. In this example, CodeDeploy orchestrates the deployment of the tested mainframe artifact into Micro Focus Enterprise Server.

In addition to the architecture walkthrough of the CI/CD pipeline, you can also read the AWS DevOps blog post Automate thousands of mainframe tests on AWS with the Micro Focus Enterprise Suite for more information on testing mainframe applications in CodeBuild and CodePipeline. Refer to the blog post for the best practices and details of doing mainframe tests on AWS.

**Tools**
AWS automation tools

- AWS CloudFormation
- Amazon CloudWatch Events
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline
- Amazon ECR
- Amazon S3
- AWS Secrets Manager
- AWS Systems Manager

Micro Focus tools

- Micro Focus Enterprise Developer for Eclipse
- Micro Focus Enterprise Test Server
- Micro Focus Enterprise Server (production deployment)
- Micro Focus AccuRev
- Micro Focus License Manager
- Micro Focus Verastream Host Integrator
- Micro Focus UFT One

Other tools

- x3270
- py3270
- Robot-Framework-3270-Library

Epics

Create the AccuRev SCM infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy a primary AccuRev SCM server by using AWS CloudFormation.</td>
<td>Log in to AccuRev SCM Server, and run the CLI command to create an Administrator user.</td>
<td>AWS CloudFormation</td>
</tr>
<tr>
<td>Create the AccuRev Administrator user.</td>
<td>Create AccuRev streams that inherit from upper streams in sequence: Production, System Integration, Team streams.</td>
<td>AccuRev SCM Administrator</td>
</tr>
<tr>
<td>Create AccuRev streams.</td>
<td>Use AccuRev SCM CLI commands to create AccuRev users login</td>
<td>AccuRev SCM Administrator</td>
</tr>
<tr>
<td>Create the developer AccuRev login accounts.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Create the Enterprise Developer Amazon EC2 launch template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>accounts for mainframe developers.</td>
<td></td>
</tr>
</tbody>
</table>

Create the Enterprise Developer instance from the Amazon EC2 template.

Create the Micro Focus Enterprise Developer tool Docker image

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use the Docker command and the Micro Focus Enterprise Developer tool Dockerfile to create the Docker image.</td>
<td>Docker</td>
</tr>
<tr>
<td></td>
<td>On the Amazon ECR console, create the repository for the Micro Focus Enterprise Developer Docker image.</td>
<td>Amazon ECR</td>
</tr>
<tr>
<td></td>
<td>Run the Docker push command to push the Enterprise Developer tool Docker image to save it in the Docker repository in Amazon ECR.</td>
<td>Docker</td>
</tr>
</tbody>
</table>

Create the Micro Focus Enterprise Test Server Docker image

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use the Docker command and the Micro Focus Enterprise Test Server Dockerfile to create the Docker image.</td>
<td>Docker</td>
</tr>
<tr>
<td></td>
<td>On the Amazon ECR console, create the Amazon ECR</td>
<td>Amazon ECR</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>repository for the Micro Focus Enterprise Test Server Docker image.</td>
<td>Docker</td>
</tr>
<tr>
<td>Push the Micro Focus Enterprise Test Server Docker image to Amazon ECR.</td>
<td>Run the Docker push command to push and save the Enterprise Test Server Docker image in Amazon ECR.</td>
<td>Docker</td>
</tr>
</tbody>
</table>

**Create the team stream CI/CD pipeline**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS CodeCommit repository.</td>
<td>On the CodeCommit console, create a Git-based repository for infrastructure and AWS CloudFormation code.</td>
<td>AWS CodeCommit</td>
</tr>
<tr>
<td>Upload the AWS CloudFormation template and the automation code into the CodeCommit repository.</td>
<td>Run the Git push command to upload AWS CloudFormation template and automation code into the repository.</td>
<td>Git</td>
</tr>
<tr>
<td>Deploy the team stream CI/CD pipeline via CloudFormation.</td>
<td>Use the prepared AWS CloudFormation template to deploy a team stream CI/CD pipeline.</td>
<td>AWS CloudFormation</td>
</tr>
</tbody>
</table>

**Create the system integration CI/CD pipeline**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Micro Focus UFT Docker image.</td>
<td>Use the Docker command and the Micro Focus UFT Dockerfile to create the Micro Focus Docker image.</td>
<td>Docker</td>
</tr>
<tr>
<td>Create the Docker repository in Amazon ECR for the Micro Focus UFT image.</td>
<td>On the Amazon ECR console, create the Docker repository for the Micro Focus UFT image.</td>
<td>Amazon ECR</td>
</tr>
<tr>
<td>Push the Micro Focus UFT Docker image to Amazon ECR.</td>
<td>Run the Docker push command to push and save the Enterprise Test Server Docker image in Amazon ECR.</td>
<td>Docker</td>
</tr>
<tr>
<td>Create the Micro Focus Verastream Docker image.</td>
<td>Use the Docker command and the Micro Focus Verastream Dockerfile to create the Docker image.</td>
<td>Docker</td>
</tr>
<tr>
<td>Create the Docker repository in Amazon ECR for the Micro Focus Verastream image.</td>
<td>On the Amazon ECR console, create the Docker repository</td>
<td>Amazon ECR</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the system integration CI/CD pipeline via CloudFormation.</td>
<td>Use the prepared AWS CloudFormation template to deploy a system integration CI/CD pipeline.</td>
<td>AWS CloudFormation</td>
</tr>
</tbody>
</table>

### Create production deployment CI/CD pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy Micro Focus Enterprise Server by using the AWS Quick Start.</td>
<td>To deploy Micro Focus Enterprise Server by using AWS CloudFormation, launch the Micro Focus Enterprise Server on AWS Quick Start.</td>
<td>AWS CloudFormation</td>
</tr>
<tr>
<td>Deploy a production deployment CI/CD pipeline.</td>
<td>On the AWS CloudFormation console, use the AWS CloudFormation template to deploy a production deployment CI/CD pipeline.</td>
<td>AWS CloudFormation</td>
</tr>
</tbody>
</table>

### Related resources

**References**
- AWS DevOps Blog - Automate thousands of mainframe tests on AWS with the Micro Focus Enterprise Suite
- py3270/py3270 GitHub repository
- Altran-PT-GDC/Robot-Framework-Mainframe-3270-Library GitHub repository
- Welcome to behave!
- APN Partner Blog - Tag: Micro Focus
- Launching an instance from a launch template

**AWS Marketplace**
- Micro Focus UFT One

**AWS Quick Start**
- Micro Focus Enterprise Server on AWS

---

**Set up integrated DNS resolution for hybrid networks in Amazon Route 53**

---

613
Summary

This pattern describes how to set up a fully hybrid Domain Name System (DNS) architecture that enables end-to-end DNS resolution of on-premise resources, AWS resources, and internet DNS queries, without administrative overhead. The pattern describes how to set up Amazon Route 53 Resolver forwarding rules that determine where a DNS query that originates from AWS should be sent, based on the domain name. DNS queries for on-premises resources are forwarded to on-premises DNS resolvers. DNS queries for AWS resources and internet DNS queries are resolved by Route 53 Resolver.

Prerequisites and limitations

Prerequisites

- An AWS account
- A virtual private cloud (VPC) in your AWS account
- A network connection between the on-premises environment and your VPC, through AWS Virtual Private Network (AWS VPN) or AWS Direct Connect
- IP addresses of your on-premises DNS resolvers (reachable from your VPC)
- Domain/subdomain name to forward to on-premises resolvers (for example, onprem.mydc.com)
- Domain/subdomain name for the AWS private hosted zone (for example, myvpc.cloud.com)

Architecture

Target technology stack

- Amazon Route 53 private hosted zone
- Amazon Route 53 Resolver
- Amazon VPC
- AWS VPN or Direct Connect

Target architecture
Tools

- **Amazon Route 53 Resolver** – Amazon Route 53 Resolver makes hybrid cloud easier for enterprise customers by enabling seamless DNS query resolution across your entire hybrid cloud. You can create DNS endpoints and conditional forwarding rules to resolve DNS namespaces between your on-premises data center and your VPCs.

- **Amazon Route 53 private hosted zone** – A private hosted zone is a container that holds information about how you want Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs that you create with the Amazon VPC service.

Epics

Configure a private hosted zone

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Route 53 private hosted zone for an AWS reserved domain name such as myvpc.cloud.com.</td>
<td>This zone holds the DNS records for AWS resources that should be resolved from the on-premises environment. For instructions, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zone-private-creating.html">https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zone-private-creating.html</a>.</td>
<td>Network admin, System admin</td>
</tr>
</tbody>
</table>

Associate the private hosted zone with your VPC. To enable resources in your VPC to resolve DNS records in this private hosted zone, you must associate your VPC with the hosted zone. For instructions, see https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zone-private-creating.html. | Network admin, System admin       |
### Set up Route 53 Resolver endpoints

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an inbound endpoint.</td>
<td>Route 53 Resolver uses the inbound endpoint to receive DNS queries from on-premises DNS resolvers. For instructions, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-forwarding-inbound-queries.html">https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-forwarding-inbound-queries.html</a>. Make a note of the inbound endpoint IP address.</td>
<td>Network admin, System admin</td>
</tr>
<tr>
<td>Create an outbound endpoint.</td>
<td>Route 53 Resolver uses the outbound endpoint to send DNS queries to on-premises DNS resolvers. For instructions, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-forwarding-outbound-queries.html">https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-forwarding-outbound-queries.html</a>. Make a note of the output endpoint ID.</td>
<td>Network admin, System admin</td>
</tr>
</tbody>
</table>

### Set up a forwarding rule and associate it with your VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a forwarding rule for the on-premises domain.</td>
<td>This rule will instruct Route 53 Resolver to forward any DNS queries for on-premises domains (such as onprem.mydc.com) to on-premises DNS resolvers. To create this rule, you will need the IP addresses of the on-premises DNS resolvers and the outbound endpoint ID for Route 53 Resolver. For instructions, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-rules-managing.html">https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-rules-managing.html</a>.</td>
<td>Network admin, System admin</td>
</tr>
<tr>
<td>Associate the forwarding rule with your VPC.</td>
<td>For the forwarding rule to take effect, you must associate the rule with your VPC. Route 53 Resolver then takes the rule into consideration when resolving a domain. For instructions, see <a href="https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-rules-managing.html">https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-rules-managing.html</a>.</td>
<td>Network admin, System admin</td>
</tr>
</tbody>
</table>
Configure on-premises DNS resolvers

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure conditional forwarding in the on-premise DNS resolvers.</td>
<td>For DNS queries to be sent to the Route 53 private hosted zone from the on-premises environment, you must configure conditional forwarding in the on-premises DNS resolvers. This instructs the DNS resolvers to forward all DNS queries for the AWS domain (for example, for myvpc.cloud.com) to the inbound endpoint IP address for Route 53 Resolver.</td>
<td>Network admin, System admin</td>
</tr>
</tbody>
</table>

Test end-to-end DNS resolution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test DNS resolution from AWS to the on-premises environment.</td>
<td>From a server in the VPC, perform a DNS query for an on-premises domain (such as server1.onprem.mydc.com).</td>
<td>Network admin, System admin</td>
</tr>
<tr>
<td>Test DNS resolution from the on-premises environment to AWS.</td>
<td>From an on-premises server, perform DNS resolution for an AWS domain (such as server1.myvpc.cloud.com).</td>
<td>Network admin, System admin</td>
</tr>
</tbody>
</table>

Related resources

- Centralized DNS management of hybrid cloud with Amazon Route 53 and AWS Transit Gateway (AWS Networking & Content Delivery blog)
- Simplify DNS management in a multi-account environment with Route 53 Resolver (AWS Security blog)
- Working with private hosted zones (Route 53 documentation)
- Getting started with Route 53 Resolver (Route 53 documentation)

Use consistent Availability Zones in VPCs across different AWS accounts

Created by Adam Spicer (AWS)
Summary

On the Amazon Web Services (AWS) Cloud, an Availability Zone has a name that can vary between your AWS accounts and an Availability Zone ID (AZ ID) that identifies its location. If you use AWS CloudFormation to create virtual private clouds (VPCs), you must specify the Availability Zone's name when creating the subnets. If you create VPCs in multiple accounts, the Availability Zone name is randomized and this means that subnets use different Availability Zones in each account.

To use the same Availability Zone across your accounts, you must map the Availability Zone name in each account to the same AZ ID. For example, the following diagram shows that the use1-az6 AZ ID is named us-east-1a in AWS account A and us-east-1c in AWS account Z.

This pattern helps ensure zonal consistency by providing a cross-account, scalable solution for using the same Availability Zones in your subnets. Zonal consistency ensures that your cross-account network traffic avoids cross-Availability Zone network paths, which helps reduce data transfer costs and lower network latency between your workloads.

Prerequisites and limitations

Prerequisites

- At least two active AWS accounts in the same AWS Region.
- Evaluate how many Availability Zones are needed to support your VPC requirements in the Region.
- Identify and record the AZ ID for each Availability Zone that you need to support. For more information about this, see Availability Zone IDs for your AWS resources in the AWS Resource Access Manager documentation.
- An ordered, comma-separated list of your AZ IDs. For example, the first Availability Zone on your list is mapped as az1, the second Availability Zone is mapped as az2, and this mapping structure continues until your comma-separated list is fully mapped. There is no maximum number of AZ IDs that can be mapped.
- The az-mapping.yaml file from the GitHub Multi-account Availability Zone mapping repository, copied to your local machine.
Architecture

The following diagram shows the architecture that is deployed in an account and that creates AWS Systems Manager Parameter Store values. These Parameter Store values are consumed when you create a VPC in the account.

The diagram shows the following workflow:

1. This pattern's solution is deployed to all accounts that require zonal consistency for a VPC.
2. The solution creates Parameter Store values for each AZ ID and stores the new Availability Zone name.
3. The AWS CloudFormation template uses the Availability Zone name stored in each Parameter Store value and this ensures zonal consistency.

The following diagram shows the workflow for creating a VPC with this pattern's solution.
The diagram shows the following workflow:

1. Submit a template for creating a VPC to AWS CloudFormation.
2. AWS CloudFormation resolves the Parameter Store values for each Availability Zone and returns the Availability Zone name for each AZ ID.
3. A VPC is created with the correct AZ IDs required for zonal consistency.

After you deploy this pattern's solution, you can create subnets that reference the Parameter Store values. If you use AWS CloudFormation, you can reference the Availability Zone mapping parameter values from the following YAML-formatted sample code:

```yaml
Resources:
  PrivateSubnet1AZ1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Ref PrivateSubnetAZ1CIDR
      AvailabilityZone: !Join
        - '
        - - {{resolve:ssm:/az-mapping/az1:1}}'
```

This sample code is contained in the `vpc-example.yaml` file from the GitHub Multi-account Availability Zone mapping repository. It shows you how to create a VPC and subnets that align to the Parameter Store values for zonal consistency.

**Technology stack**

- AWS CloudFormation
- AWS Lambda
- AWS Systems Manager Parameter Store
Automation and scale

You can deploy this pattern to all your AWS accounts by using AWS CloudFormation StackSets or the Customizations for AWS Control Tower solution. For more information about this, see Working with AWS CloudFormation StackSets in the AWS Cloudformation documentation and Customizations for AWS Control Tower in the AWS Solutions Library.

After you deploy the AWS CloudFormation template, you can update it to use the Parameter Store values and deploy your VPCs in pipelines or according to your requirements.

Tools

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS Systems Manager Parameter Store** – Parameter Store, a capability of AWS Systems Manager, provides secure, hierarchical storage for configuration data management and secrets management.

Code

The code for this pattern is provided in the GitHub Multi-account Availability Zone mapping repository.

Epics

**Deploy the az-mapping.yaml file**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the required Availability Zones for the Region.</td>
<td>1. Determine the AZ IDs that must be consistently used in your Region. 2. Record these AZ IDs in a comma-separated list and in the order that you want them applied in. For example, the first Availability Zone on your list is mapped as az1 and the second is mapped as az2. There is no maximum number of AZ IDs that can be mapped.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the az-mapping.yaml file.</td>
<td>Use the az-mapping.yaml file to create an AWS CloudFormation stack in all required AWS accounts. In the AZIds parameter, use the comma-separated list that you created earlier.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
We recommend that you use AWS CloudFormation StackSets or the Customizations for AWS Control Tower Solution.

### Related resources

- Availability Zone IDs for your AWS resources (AWS Resource Access Manager documentation)
- AWS::EC2::Subnet (AWS CloudFormation documentation)

### More patterns

- Automate adding or updating Windows registry entries using AWS Systems Manager (p. 687)
- Automate RabbitMQ configuration in Amazon MQ (p. 742)
- Automate SAML 2.0 federation for AWS multi-account environments that use Azure AD (p. 1583)
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK (p. 384)
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK (p. 395)
- Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit (p. 487)
- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config (p. 1601)
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager (p. 1345)
- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode (p. 559)
• Configure SAML SSO from Microsoft Azure AD to CloudEndure Migration (p. 752)
• Configure VMware vRealize Automation to provision VMs on VMware Cloud on AWS (p. 563)
• Deploy a VMware SDDC on AWS by using VMware Cloud on AWS (p. 569)
• Ensure that an IAM profile is associated with an EC2 instance (p. 1675)
• Identify and alert when Amazon Kinesis Data Firehose resources are not encrypted with an AWS KMS key (p. 684)
• Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet (p. 219)
• Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands (p. 223)
• Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions (p. 728)
• Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK (p. 1459)
• Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX (p. 1105)
• Monitor Amazon ElastiCache clusters for at-rest encryption (p. 1696)
• Monitor ElastiCache clusters for security groups (p. 1566)
• Rotate credentials without restarting containers (p. 246)
• Send a notification when an IAM user is created (p. 1704)
• Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization (p. 709)
• Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible (p. 958)
• Simplify private certificate management by using ACM Private CA and AWS RAM (p. 1721)
• Tag Transit Gateway attachments automatically using AWS Organizations (p. 1542)
• Use Serverspec for test-driven development of infrastructure code (p. 540)
Configure logging and monitoring for security events in your AWS IoT environment

Created by Prateek Prakash (AWS)

**Environment:** Production  
**Technologies:** IoT; Security, identity, compliance; Operations  
**Workload:** All other workloads

**AWS services:** Amazon CloudWatch; Amazon OpenSearch Service; Amazon GuardDuty; AWS IoT Core; AWS IoT Device Defender; AWS IoT Device Management; Amazon CloudWatch Logs

**Summary**

Ensuring that your Internet of Things (IoT) environments are secure is an important priority, particularly because organizations are connecting billions of devices to their IT environments. This pattern provides a reference architecture that you can use to implement logging and monitoring for security events across your IoT environment on the Amazon Web Services (AWS) Cloud. Typically, an IoT environment on the AWS Cloud has the following three layers:

- IoT devices that generate relevant telemetry data.
- AWS IoT services (for example, AWS IoT Core, AWS IoT Device Management, or AWS IoT Device Defender) that connect your IoT devices to other devices and AWS services.
- Backend AWS services that help process telemetry data and provide useful insights for your different business use cases.

The best practices provided by the AWS IoT Lens - AWS Well-Architected Framework whitepaper can help you review and improve your cloud-based architecture and better understand the business impact of your design decisions. An important recommendation is that you analyze application logs and metrics on your devices and in the AWS Cloud. You can achieve this by leveraging different approaches and techniques (for example, threat modeling) to identify metrics and events that must be monitored to detect potential security issues.

This pattern describes how to use AWS IoT and security services to design and implement a security logging and monitoring reference architecture for an IoT environment on the AWS Cloud. This architecture builds on existing AWS security best practices and applies them to your IoT environment.
Prerequisites and limitations

Prerequisites

- An existing landing zone environment. For more information about this, see the guide Setting up a secure and scalable multi-account AWS environment on the AWS Prescriptive Guidance website.
- The following accounts must be available in your landing zone:
  - **Log Archive account** – This account is for users that need to access the logging information for accounts in your landing zone’s organizational units (OUs). For more information about this, see the Security OU – Log Archive account section of the guide AWS Security Reference Architecture on the AWS Prescriptive Guidance website.
  - **Security account** – Your security and compliance teams use this account for auditing or to perform emergency security operations. This account is also designated as the administrator account for Amazon GuardDuty. Users from the administrator account can configure GuardDuty, in addition to viewing and managing GuardDuty findings for their own account and all member accounts. For more information about this, see Managing multiple accounts in GuardDuty in the Amazon GuardDuty documentation.

Architecture

This pattern extends the Centralized Logging solution from the AWS Solutions Library to collect and process security-related IoT events. The Centralized Logging solution is deployed in the Security account and helps collect, analyze, and display Amazon CloudWatch logs in a single dashboard. This solution consolidates, manages, and analyzes log files from multiple sources. Finally, the Centralized Logging solution also uses Amazon OpenSearch Service (successor to Amazon Elasticsearch Service) and OpenSearch Dashboards to show a unified view of all log events.

The following architecture diagram shows the key components of an IoT security logging and reference architecture on the AWS Cloud.

The diagram shows the following workflow:
1. IoT things are the devices that must be monitored for anomalous security events. These devices run an agent to publish security events or metrics to AWS IoT Core and AWS IoT Device Defender.

2. When AWS IoT logging is enabled, AWS IoT sends progress events about each message as it passes from your devices through the message broker and rules engine to Amazon CloudWatch Logs. You can use CloudWatch Logs subscriptions to push events to a Centralized Logging solution. For more information about this, see AWS IoT metrics and dimensions in the AWS IoT Core documentation.

3. AWS IoT Device Defender helps monitor insecure configurations and security metrics for your IoT devices. When an anomaly is detected, alarms notify Amazon Simple Notification Service (Amazon SNS), which has an AWS Lambda function as a subscriber. The Lambda function sends the alarm as a message to CloudWatch Logs. You can use CloudWatch Logs subscriptions to push events to your Centralized Logging solution. For more information about this, see Audit checks, Device-side metrics, and Cloud-side metrics in the AWS IoT Core documentation.

4. AWS CloudTrail logs AWS IoT Core control plane actions that make changes (for example, creating, updating, or attaching APIs). When CloudTrail is set up as part of a landing zone implementation, it sends events to CloudWatch Logs and you can use subscriptions to push events to your Centralized Logging solution.

5. AWS Config managed rules or custom rules evaluate resources that are part of your IoT environment. Monitor your compliance change notifications using CloudWatch Events with CloudWatch Logs as the target. After compliance change notifications are sent to CloudWatch Logs, you can use subscriptions to push events to your Centralized Logging solution.

6. Amazon GuardDuty continuously analyzes CloudTrail management events and helps identify API calls made to AWS IoT Core endpoints from known malicious IP addresses, unusual geolocations, or anonymizing proxies. Monitor GuardDuty notifications using Amazon CloudWatch Events with log groups in CloudWatch Logs as the target. When GuardDuty notifications are sent to CloudWatch Logs, you can use subscriptions to push events to your Centralized Monitoring solution or use the GuardDuty console in your Security account to view the notifications.

7. AWS Security Hub monitors your IoT account by using security best practices. Monitor Security Hub notifications by using CloudWatch Events with log groups in CloudWatch Logs as the target. When Security Hub notifications are sent to CloudWatch Logs, use subscriptions to push events to your Centralized Monitoring solution or use the Security Hub console in your Security account to view the notifications.

8. Amazon Detective evaluates and analyzes information to isolate the root cause and take action on security findings for unusual calls to AWS IoT endpoints or other services in your IoT architecture.

9. Amazon Athena queries logs stored in your Log Archive account to enhance your understanding of security findings and identify trends and malicious activities.

Tools

- **Amazon Athena** – Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL.
- **AWS CloudTrail** – AWS CloudTrail helps you enable governance, compliance, and operational and risk auditing of your AWS account.
- **Amazon CloudWatch** – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.
- **Amazon CloudWatch Logs** – Amazon CloudWatch Logs centralizes the logs from all your systems, applications, and AWS services that you use. You can view and monitor the logs, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis.
- **AWS Config** – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account.
- **Amazon Detective** – Amazon Detective makes it easy to analyze, investigate, and quickly identify the root cause of security findings or suspicious activities.

- **AWS Glue** – AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.

- **Amazon GuardDuty** – Amazon GuardDuty is a continuous security monitoring service.

- **AWS IoT Core** – AWS IoT Core provides secure, bi-directional communication for Internet-connected devices (such as sensors, actuators, embedded devices, wireless devices, and smart appliances) to connect to the AWS Cloud over MQTT, HTTPS, and LoRaWAN.

- **AWS IoT Device Defender** – AWS IoT Device Defender is a security service that allows you to audit the configuration of your devices, monitor connected devices to detect abnormal behavior, and mitigate security risks.

- **Amazon OpenSearch Service** – Amazon OpenSearch Service (successor to Amazon Elasticsearch Service) is a managed service that makes it easy to deploy, operate, and scale OpenSearch clusters in the AWS Cloud.

- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

- **AWS Security Hub** – AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.

- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

## Epics

**Set up an IoT account in your landing zone environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IoT account in your landing zone.</td>
<td>Create an AWS account in your landing zone that meets your organization’s requirements and that you can use for your IoT environment.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Validate the security guardrails in the IoT account.</td>
<td>Validate that the guardrails for CloudTrail, AWS Config, GuardDuty, and Security Hub are enabled in your IoT account.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>
| Validate that your IoT account is configured as a member account of your Security account. | Validate that your IoT account is configured and associated as a member account for GuardDuty and Security Hub in your Security account.  
For more information about this, see Managing GuardDuty accounts with AWS Organizations in the Amazon GuardDuty documentation and Managing administrator and | AWS administrator        |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>member accounts in the AWS Security Hub documentation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validate log archiving.</td>
<td>Validate that CloudTrail, AWS Config, and VPC Flow Logs are stored in the Log Archive account.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>

### Set up the Centralized Logging solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the Centralized Logging solution in your Security account.</td>
<td>Sign in to the AWS Management Console for your Security account and set up the <a href="#">Centralized Logging solution</a> from the AWS Solutions Library to collect, analyze, and display CloudWatch Logs in Amazon OpenSearch Service and OpenSearch Dashboards. For more information about this, see <a href="#">Collect, analyze, and display Amazon CloudWatch Logs in a single dashboard with the Centralized Logging solution</a> from the Centralized Logging implementation guide in the AWS Solutions Library.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>

### Set up and configure AWS resources in your IoT account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up AWS IoT logging.</td>
<td>Sign in to the AWS Management Console for your IoT account. Set up and configure AWS IoT Core to send logs to CloudWatch Logs. For more information about this, see <a href="#">Configure AWS IoT logging and Monitor AWS IoT using CloudWatch Logs</a> in the AWS IoT Core documentation.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Set up AWS IoT Device Defender.</td>
<td>Set up AWS IoT Device Defender to audit your IoT resources and detect anomalies. For more information about this, see <a href="#">Getting started with AWS</a></td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| IoT Device Defender | Set up CloudTrail to send events to CloudWatch Logs.  
For more information about this, see Sending events to CloudWatch Logs in the AWS CloudTrail documentation. | AWS administrator |
| Set up CloudTrail. | Set up AWS Config and AWS Config rules.  
For more information about this, see Setting up AWS Config with the console and Setting up AWS Config rules with the console in the AWS Config documentation. | AWS administrator |
| Set up AWS Config and AWS Config rules. | Set up GuardDuty to send findings to Amazon CloudWatch Events with log groups in CloudWatch Logs as the target.  
For more information about this, see Creating custom responses to GuardDuty findings with Amazon CloudWatch Events in the Amazon GuardDuty documentation. | AWS administrator |
For more information about this, see Automated response and remediation in the AWS Security Hub documentation. | AWS administrator |
| Set up Security Hub. | Set up Amazon Detective to facilitate analysis of security findings  
For more information about this, see Setting up Amazon Detective in the Amazon Detective documentation. | AWS administrator |
| Set up Amazon Detective. | | |
Set up Amazon Athena and AWS Glue.

Set up Athena and AWS Glue to query the AWS service logs that conduct security incident investigations.

For more information about this, see Querying AWS service logs in the Amazon Athena documentation.

**Related resources**

- What is a landing zone?

**Extract and query AWS IoT SiteWise metadata attributes in a data lake**

*Created by Ambarish Dongaonkar (AWS)*

**Environment:** Production  
**Technologies:** IoT; Analytics; Big data  
**AWS services:** AWS IoT SiteWise; AWS Lambda; AWS Glue

**Summary**

AWS IoT SiteWise uses asset models and hierarchies to represent your industrial equipment, processes, and facilities. Each model or asset can have multiple attributes that are specific to your environment. Example metadata attributes include the site or physical location of the asset, plant details, and equipment identifiers. These attribute values complement asset measurement data to maximize the business value. Machine learning (ML) can provide additional insights into this metadata and streamline engineering tasks.

However, metadata attributes can’t be queried directly from the AWS IoT SiteWise service. To make the attributes queryable, you must extract and ingest them into a data lake. This pattern uses a Python script to extract the attributes for all AWS IoT SiteWise assets and ingest them into a data lake in an Amazon Simple Storage Service (Amazon S3) bucket. When you have completed this process, you can use SQL queries in Amazon Athena to access the AWS IoT SiteWise metadata attributes and other datasets, such as measurement datasets. The metadata attribute information is also useful when working with AWS IoT SiteWise monitors or dashboards. You can also build an AWS QuickSight dashboard by using the extracted attributes in the S3 bucket.

The pattern has reference code, and you can implement the code by using the best compute services for your use case, such as AWS Lambda or AWS Glue.

**Prerequisites and limitations**

**Prerequisites**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up Amazon Athena and AWS Glue.</td>
<td>Set up Athena and AWS Glue to query the AWS service logs that conduct security incident investigations.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>

For more information about this, see Querying AWS service logs in the Amazon Athena documentation.
• An active AWS account.
• Permissions to set up AWS Lambda functions or AWS Glue jobs.
• An Amazon S3 bucket.
• The asset models and hierarchies are set up in AWS IoT SiteWise. For more information, see Creating asset models (AWS IoT SiteWise documentation).

Architecture

You can use a Lambda function or an AWS Glue job to complete this process. We recommend using Lambda if you have less than 100 models and each model has an average of 15 or fewer attributes. For all other use cases, we recommend using AWS Glue.

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

1. The scheduled AWS Glue job or Lambda function runs. It extracts the asset metadata attributes from AWS IoT SiteWise and ingests them into an S3 bucket.
2. An AWS Glue crawler crawls the extracted data in the S3 bucket and creates tables in an AWS Glue Data Catalog.
3. Using standard SQL, Amazon Athena queries the tables in the AWS Glue Data Catalog.

Automation and scale

You can schedule the Lambda function or AWS Glue job to run daily or weekly, according to the update frequency of your AWS IoT SiteWise asset configurations.

There is no limit to the number of AWS IoT SiteWise assets that the sample code can process, but a large number of assets can increase the amount of time required to complete the process.

Tools

• Amazon Athena – Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL.
• AWS Glue – AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.
• **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

• **AWS IoT SiteWise** - AWS IoT SiteWise is a managed service designed to collect, model, analyze, and visualize data from industrial equipment at scale. AWS IoT SiteWise provides an asset modeling framework for building representations of your industrial devices, processes, and facilities.

• **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code isn’t running.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

• **SDK for Python** – AWS SDK for Python (Boto3) allows you to integrate your Python application, library, or script with AWS services.

### Epics

**Set up the job or function**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure permissions in IAM.</td>
<td>In the IAM console, grant permissions to the IAM role assumed by the Lambda function or AWS Glue job to do the following:&lt;br&gt;• Read from the AWS IoT SiteWise service&lt;br&gt;• Write to the S3 bucket&lt;br&gt;For more information, see <a href="https://docs.aws.amazon.com/IAM/latest/userguide/iam-service-authorization.html">Creating a role for an AWS service</a> (IAM documentation).</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create the Lambda function or AWS Glue job.</td>
<td>If you are using Lambda, create a new Lambda function. For <strong>Runtime</strong>, choose <strong>Python</strong>. For more information, see <a href="https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html">Building Lambda functions with Python</a> (Lambda documentation).&lt;br&gt;If you are using AWS Glue, create a new Python shell job in the AWS Glue console. For more information, see <a href="https://docs.aws.amazon.com/glue/latest/dg/python-shells.html">Adding Python shell jobs</a> (AWS Glue documentation).</td>
<td>General AWS</td>
</tr>
<tr>
<td>Update the Lambda function or AWS Glue job.</td>
<td>Modify the new Lambda function or AWS Glue job, and enter the code sample in the [Additional information](p. 634)</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
## Task

### Run the job or function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the Lambda function or AWS Glue job.</td>
<td>Run the Lambda function or AWS Glue job. For more information, see <a href="https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-functions.html">Invoke the Lambda function</a> or <a href="https://docs.aws.amazon.com/glue/latest/dg/creating-crawler-workflow.html">Starting jobs using triggers</a>. This extracts the metadata attributes for the assets and models in the AWS IoT SiteWise hierarchy and stores them in the specified S3 bucket.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Set up an AWS Glue crawler.</td>
<td>Set up an AWS Glue crawler with the necessary format classifier for a CSV-formatted file. Use the S3 bucket and prefix details used in the Lambda function or AWS Glue job. For more information, see <a href="https://docs.aws.amazon.com/glue/latest/dg/crawling.html">Defining crawlers</a>.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Run the AWS Glue crawler.</td>
<td>Run the crawler to process the data file created by the Lambda function or AWS Glue job. The crawler creates a table in the specified AWS Glue Data Catalog. For more information, see <a href="https://docs.aws.amazon.com/glue/latest/dg/crawling.html">Starting crawlers using triggers</a>.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Query the metadata attributes.</td>
<td>Using Amazon Athena, use standard SQL to query the AWS Glue Data Catalog as needed for your use case. You can join the metadata attribute table with other databases and tables. For more information, see <a href="https://docs.aws.amazon.com/athena/latest/ug/getting-started.html">Getting Started</a>.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
Related resources

- Amazon Athena documentation
- AWS Glue documentation
- AWS IoT SiteWise API reference
- AWS IoT SiteWise user guide
  - Getting started
  - Modeling industrial assets
  - Defining relationships between asset models (hierarchies)
  - Associating and disassociating assets
  - Creating the AWS IoT SiteWise demo
- IOTSiteWise (SDK for Python documentation)
- Lambda documentation

Additional information

Code

The sample code provided is for reference, and you can customize this code as needed for your use case.

```python
# Following code can be used in an AWS Lambda function or in an AWS Glue Python shell job.
# IAM roles used for this job need read access to the AWS IoT SiteWise service and write
# access to the S3 bucket.
sw_client = boto3.client('iotsitewise')
s3_client = boto3.client('s3')
output = io.StringIO()
attribute_list = []
bucket = '{3_bucket name}'
prefix = '{s3_bucket prefix}'
output.write("model_id,model_name,asset_id,asset_name,attribuet_id,attribute_name,attribute_value
")
m_resp = sw_client.list_asset_models()
for m_rec in m_resp['assetModelSummaries']:
    model_id = m_rec['id']
    model_name = m_rec['name']
    attribute_list.clear()
    dam_response = sw_client.describe_asset_model(assetModelId=model_id)
    for rec in dam_response['assetModelProperties']:
        if 'attribute' in rec['type']:
            attribute_list.append(rec['name'])
    response = sw_client.list_assets(assetModelId=model_id, filter='ALL')
    for asset in response['assetSummaries']:
        asset_id = asset['id']
        asset_name = asset['name']
        resp = sw_client.describe_asset(assetId=asset_id)
        for rec in resp['assetProperties']:
            if rec['name'] in attribute_list:
                p_resp = sw_client.get_asset_property_value(assetId=asset_id,
                                                  propertyId=rec['id'])
                if 'propertyValue' in p_resp:
                    if p_resp['propertyValue']['value']:
                        if 'stringValue' in p_resp['propertyValue']['value']:
...
```
output.write(model_id + "," + model_name + "," + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," + str(p_resp['propertyValue']["stringValue"])) + "\n"
        if 'doubleValue' in p_resp['propertyValue']['value']:
            output.write(model_id + "," + model_name + "," + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," + str(p_resp['propertyValue']["doubleValue"])) + "\n"
        if 'integerValue' in p_resp['propertyValue']['value']:
            output.write(model_id + "," + model_name + "," + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," + str(p_resp['propertyValue']["integerValue"])) + "\n"
        if 'booleanValue' in p_resp['propertyValue']['value']:
            output.write(model_id + "," + model_name + "," + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," + str(p_resp['propertyValue']["booleanValue"])) + "\n"
    output.seek(0)
s3_client.put_object(Bucket=bucket, Key= prefix + '/data.csv', Body=output.getvalue())
    output.close()
Machine learning & AI

Analyze speech in real time using Amazon Transcribe and Amazon Comprehend (p. 636)
Automatically extract content from PDF files using Amazon Textract (p. 643)
Train and deploy a custom GPU-supported ML model on Amazon SageMaker (p. 648)
Use SageMaker Processing for distributed feature engineering of terabyte-scale ML datasets (p. 661)
Visualize AI/ML model results using Flask and AWS Elastic Beanstalk (p. 672)
More patterns (p. 683)

Analyze speech in real time using Amazon Transcribe and Amazon Comprehend

Created by Suresh Konathala (AWS)

| Environment: PoC or pilot | Technologies: Machine learning & AI; Websites & web apps; Security, identity, compliance | AWS services: Amazon Comprehend; Amazon S3; Amazon Transcribe; Amazon CloudFront |

Summary

Many organizations need to process and analyze conversations with their customers but audio data can be difficult to search and analyze. This pattern describes how to use Amazon Transcribe’s real-time transcription to convert speech into text and then analyze it using Amazon Comprehend to recognize and extract entities such as names, locations, or dates and quantities.

The front-end user interface (UI) is deployed as a static website in an Amazon Simple Storage Service (Amazon S3) bucket and accessed with Amazon CloudFront. When a user speaks, the audio data stream is sent over a WebSocket connection to an Amazon Transcribe streaming API.

This pattern also describes how to use an AWS Identity and Access Management (IAM) role with an attached policy to invoke Amazon Transcribe and Amazon Comprehend.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account.
• IAM permissions to invoke Amazon Transcribe and Amazon Comprehend.
• The `frontend-app.zip` and `lambda_function.zip` files (attached), downloaded to your local computer.

Limitations

• You can improve the transcription's accuracy by providing custom vocabulary files. For more information about this, see Custom vocabularies in the Amazon Transcribe documentation.
• The built-in version of Amazon Comprehend only recognizes entities such as names of people, locations, or dates. You can use Amazon Comprehend's custom entity recognition to extract business-specific entities that fit your use case (for example, product codes). For more information about this, see Custom entity recognition in the Amazon Comprehend documentation.

Architecture

The diagram shows the following workflow:

1. The contents of a static website are hosted on Amazon S3 and served using a CloudFront distribution. A user accesses the website and the following webpage appears:
2. After the user clicks the microphone icon, the browser makes a REST API call to Amazon API Gateway. API Gateway then invokes an AWS Lambda function.

3. The Lambda function assumes an IAM role with an attached policy that has permission to invoke Amazon Transcribe and Amazon Comprehend. The Lambda function invokes the AWS Security Token Service (AWS STS) API to fetch the temporary access key ID and secret access key credentials. These temporary access credentials are sent to the browser in the REST API call response.

4. The front-end UI captures the microphone's live audio by using a microphone stream npm module. The UI sends the audio stream to an Amazon Transcribe WebSocket API. This API responds with a stream of transcribed text.

5. The transcribed text is sent to Amazon Comprehend for analysis and entity recognition. Amazon Comprehend responds with a list of identified entities (for example, names of people, locations, and specific dates or quantities). The following image shows how the UI uses this information to color code the text and display it:
Tools

- **Amazon API Gateway** – API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- **Amazon Comprehend** – Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents.
- **AWS Identity and Access Management (IAM)** – IAM is a web service that helps you securely control access to AWS resources.
- **AWS Lambda** – Lambda supports running code without provisioning or managing servers.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, backups, and data lakes.
- **AWS STS** – AWS Security Token Service (AWS STS) is a web service that enables you to request temporary, limited-privilege credentials for IAM users or for users that you authenticate.
- **Amazon Transcribe** – Amazon Transcribe uses a deep learning process called automatic speech recognition (ASR) to convert speech to text quickly and accurately.

Code

The `frontend-app.zip` and `lambda_function.zip` files (attached) contain the source code for this pattern.

Epics

Create an API to receive temporary credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the IAM policy.</td>
<td>Sign in to the AWS Management Console, open the IAM console, and create an IAM policy by using the following code:</td>
<td>App developer, AWS DevOps, General AWS</td>
</tr>
</tbody>
</table>

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Principal": {
        "Service": "viewer.amazonaws.com"
      },
      "Resource": "arn:aws:s3:::examplebucket/*"
    }
  ]
}
```
AWS Prescriptive Guidance Patterns

Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Effect&quot;:</td>
<td>&quot;Action&quot;: [</td>
</tr>
<tr>
<td></td>
<td>&quot;Allow&quot;,</td>
<td>&quot;comprehend:DescribeSentimentDetectionJob&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeTopicsDetectionJob&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DetectSentiment&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeEntityRecognizer&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeDominantLanguageDetectionJob&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeDocumentClassificationJob&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:BatchDetectSentiment&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:BatchDetectEntities&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:BatchDetectKeyPhrases&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DetectDominantLanguage&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeEntitiesDetectionJob&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:ClassifyDocument&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DetectSyntax&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeDocumentClassifier&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:BatchDetectSyntax&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;transcribe:StartStreamTranscription&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:BatchDetectDominantLanguage&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeEndpoint&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DetectEntities&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;transcribe:StartStreamTranscriptionWebSocket&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DetectKeyPhrases&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;comprehend:DescribeKeyPhrasesDetectionJob&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>], &quot;Resource&quot;: &quot;*&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>] }</td>
<td></td>
</tr>
</tbody>
</table>

For more information about this, see Creating IAM policies in the IAM documentation.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create and attach an IAM role to the IAM policy. | 1. On the IAM console, create an IAM role. For more information about this, see Creating IAM roles in the IAM documentation. <br>2. After you create the IAM role, choose Roles in the navigation pane of the IAM console. Choose the name of the role that created earlier and then choose the Trust relationships tab. Choose Edit trust relationship and paste the following code under Policy Document:  

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iam.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<account_number>:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Choose Update Trust Policy. | App developer, AWS DevOps, General AWS |

Create the Lambda function. | Open the Lambda console and create a Lambda function with a Python 3.7 runtime environment by using the lambda_function.py file (attached). <br>For more information about this, see Create a Lambda function with the console in the AWS Lambda documentation. | App developer, General AWS |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an API Gateway endpoint.</strong></td>
<td>Expose the Lambda function as a REST endpoint using API Gateway by following the instructions from the Build a Hello World REST API with Lambda proxy integration tutorial in the Amazon API Gateway documentation. <strong>Important:</strong> Record the API Gateway endpoint’s URL.</td>
<td>General AWS, App developer</td>
</tr>
</tbody>
</table>

**Create a static website IaC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an S3 bucket with a bucket policy set for CloudFront origin access identity.</strong></td>
<td>On the AWS CloudFormation console, use the template from the staticwebsite.yaml file (attached) to create a stack that creates an S3 bucket with a bucket policy for CloudFront origin access identity.</td>
<td>General AWS, App developer</td>
</tr>
</tbody>
</table>

**Create the front-end UI**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Update the API Gateway endpoint URL.** | 1. Extract the source code from the frontend-app.zip file (attached) and open the getTranscribeCredentials.js file.  
2. Replace <GET CREDENTIALS API GATEWAY ENDPOINT> with the API Gateway endpoint URL that you recorded earlier. | App developer                         |
| **Deploy the application.**               | Navigate to the extracted source code directory and run the following commands to build and deploy your application:  
• npm install  
• npm run build  
• aws s3 cp - recursive ./build s3:// <bucket_name> | App developer                         |
Automatically extract content from PDF files using Amazon Textract

**Creating by Tianxia Jia (AWS)**

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies: Machine learning &amp; AI; Analytics; Big data</th>
<th>AWS services: Amazon S3; Amazon Textract; Amazon SageMaker</th>
</tr>
</thead>
</table>

**Summary**

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

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

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

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

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
• An existing Amazon Simple Storage Service (Amazon S3) bucket to store the PDF files after they are converted to JPEG format for processing by Amazon Textract. For more information about S3 buckets, see Buckets overview in the Amazon S3 documentation.
• The Textract_PostProcessing.ipynb Jupyter notebook (attached), installed and configured. For more information about Jupyter notebooks, see Create a Jupyter notebook in the Amazon SageMaker documentation.
• Existing PDF files that have an identical format.
• An understanding of Python.

Limitations

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

Architecture

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

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

The diagram shows the following workflow for this pattern:

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

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

**Target technology stack**

- Amazon SageMaker
- Amazon S3
- Amazon Textract

**Automation and scale**

You can automate the *Repeat run* workflow by using an AWS Lambda function that initiates Amazon Textract when a new PDF file is added to Amazon S3. Amazon Textract then runs the processing scripts and the final output can be saved to a storage location. For more information about this, see [Using an Amazon S3 trigger to invoke a Lambda function](https://docs.aws.amazon.com/lambda/latest/dg/APIs-supported.html) in the Lambda documentation.

**Tools**

- **Amazon SageMaker** – SageMaker is a fully managed ML service that helps you to quickly and easily build and train ML models, and then directly deploy them into a production-ready hosted environment.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- **Amazon Textract** – Amazon Textract makes it easy to add document text detection and analysis to your applications.

**Epics**

**First-time run**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert the PDF file.</td>
<td>Prepare the PDF file for your first-time run by splitting it into a single page and converting it into JPEG format for the Amazon Textract synchronous operation <em>(Syn API)</em>.</td>
<td>Data scientist, Developer</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can also use the Amazon Textract asynchronous operation <em>(Asyn API)</em> for multipage PDF files.</td>
<td></td>
</tr>
<tr>
<td>Parse the Amazon Textract response JSON.</td>
<td>Open the Textract_PostProcessing.ipynb Jupyter notebook (attached) and call the Amazon Textract API by using the following code:</td>
<td>Data scientist, Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>response = textract.analyze_document(</td>
<td>Data scientist, Developer</td>
</tr>
<tr>
<td></td>
<td>Document={</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'S3Object': {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'Bucket':</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BUCKET,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'Name':</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'{}' .format(filename)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FeatureTypes=['TABLES',</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'FORMS'])</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parse the response JSON into</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a form and table by using the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>following code:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parseformKV=form_kv_from_JSON(response)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parseformTables=get_tables_fromJSON(response)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Edit the TemplateJSON file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Edit the parsed JSON for each</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KeyName and corresponding</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DataType (for example, string,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>float, integer, or date), and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>table headers (for example,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ColumnNames and RowNames).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This template is used for each</td>
<td></td>
</tr>
<tr>
<td></td>
<td>individual PDF file type, which</td>
<td></td>
</tr>
<tr>
<td></td>
<td>means that the template can be</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reused for PDF files that have an</td>
<td></td>
</tr>
<tr>
<td></td>
<td>identical format.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define the post-processing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>correction functions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The values in Amazon Textract's</td>
<td></td>
</tr>
<tr>
<td></td>
<td>response for the TemplateJSON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>file are strings. There is no</td>
<td></td>
</tr>
<tr>
<td></td>
<td>differentiation for date, float,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>integer, or currency. These</td>
<td></td>
</tr>
<tr>
<td></td>
<td>values must be converted to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the correct data type for your</td>
<td></td>
</tr>
<tr>
<td></td>
<td>downstream use case.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Correct each data type according</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to the TemplateJSON file by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>using the following code:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>finalJSON=postprocessingCorrection(parsedJSON,templateJSON)</td>
<td></td>
</tr>
</tbody>
</table>
## Repeat run

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the PDF files.</td>
<td>Prepare the PDF files by splitting them into a single page and converting them into JPEG format for the Amazon Textract <a href="https://aws.amazon.com/textract/">synchronous operation</a> (<em>Syn API</em>). Note: You can also use the Amazon Textract <a href="https://aws.amazon.com/textract/">asynchronous operation</a> (<em>Asyn API</em>) for multipage PDF files.</td>
<td>Data scientist, Developer</td>
</tr>
<tr>
<td>Call the Amazon Textract API.</td>
<td>Call the Amazon Textract API by using the following code:</td>
<td>Data scientist, Developer</td>
</tr>
</tbody>
</table>
|                                   | ```python
response =
textract.analyze_document(
    Document={
        'S3Object': {
            'Bucket': BUCKET,
            'Name': '{}'.format(filename)
        }
    ),
    FeatureTypes=['TABLES', "FORMS"]
)```                                                                                                                                                                                                                       |                          |
| Parse the Amazon Textract response JSON. | Parse the response JSON into a form and table by using the following code:                                                                                                                                                                                                                                                                 | Data scientist, Developer |
|                                   | ```python
parseformKV=form_kv_from_JSON(response)
parseformTables=get_tables_fromJSON(response)
```                                                                                                                                                                                                                                                                               |                          |
| Load the TemplateJSON file and match it with the parsed JSON. | Use the TemplateJSON file to extract the correct key-value pairs and table by using the following commands:                                                                                                                                                                                                                           | Data scientist, Developer |
|                                   | ```python
form_kv_corrected=form_kv_correction(parseformKV,templateJSON)
form_table_corrected=form_Table_correction(parseformTables, templateJSON)
form_kv_table_corrected_final={**form_kv_corrected , **form_table_corrected}
```                                                                                                                                                                                                                   |                          |
Train and deploy a custom GPU-supported ML model on Amazon SageMaker

*Created by Ankur Shukla (AWS)*

**Environment:** PoC or pilot  
**Technologies:** Machine learning & AI; Containers & microservices  
**AWS services:** Amazon ECS; Amazon SageMaker

### Summary

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

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

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

### Prerequisites and limitations

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

**Limitations**

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

**Architecture**

**Technology stack**

• SageMaker
• Amazon ECR

**Tools**

Tools
• **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.

• **Amazon SageMaker** – SageMaker is a fully managed ML service.

• **Docker** – Docker is a software platform for building, testing, and deploying applications quickly.

• **Python** – Python is a programming language.

**Code**

The code for this pattern is available on the GitHub [Implementing a review classification model with Catboost and SageMaker](https://github.com/aws-samples/Implementing-a-review-classification-model-with-Catboost-and-SageMaker) repository.

**Epics**

**Prepare the data**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an IAM role and attach the required policies. | Sign in to the AWS Management Console, open the IAM console, and create a new IAM role. Attach the following policies to the IAM role:  
  - AmazonEC2ContainerRegistryFullAccess  
  - AmazonS3FullAccess  
  - AmazonSageMakerFullAccess  

For more information about this, see [Create a notebook instance](https://docs.aws.amazon.com/sagemaker/latest/dg/notebook-instances.html) in the Amazon SageMaker documentation. | Data scientist |
|-------------------------------------------|-----------------------------------------------------------------------------|-------------------|
| Create the SageMaker notebook instance.   | Open the SageMaker console, choose **Notebook instances**, and then choose **Create notebook instance**. For **IAM role**, choose the IAM role that you created earlier. Configure the notebook instance according to your requirements and then choose **Create notebook instance**.  

For detailed steps and instructions, see [Create a notebook instance](https://docs.aws.amazon.com/sagemaker/latest/dg/notebook-instances.html) in the Amazon SageMaker documentation. | Data scientist |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the repository.</td>
<td>Open the terminal in the SageMaker notebook instance and clone the GitHub <a href="https://github.com/aws-samples/Implementing-a-review-classification-model-with-Catboost-and">Implementing a review classification model with Catboost and</a></td>
<td></td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the Jupyter notebook.</td>
<td>Start the Review classification model with Catboost and SageMaker.ipynb Jupyter notebook, which contains the predefined steps.</td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

#### Feature engineering

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run commands in Jupyter notebook.</td>
<td>Open the Jupyter notebook and run the commands from the following stories to prepare the data to train your ML model.</td>
<td>Data scientist</td>
</tr>
<tr>
<td>Read the data from the S3 bucket.</td>
<td><strong>import pandas as pd</strong>&lt;br&gt;<strong>import csv</strong>&lt;br&gt;<strong>fname = 's3://amazon-reviews-pds/tsv/amazon_reviews_us_Digital_Video_Download_v1_00.tsv.gz'</strong>&lt;br&gt;<strong>df = pd.read_csv(fname, sep='\t', delimiter='\t', error_bad_lines=False)</strong></td>
<td>Data scientist</td>
</tr>
<tr>
<td>Preprocess the data.</td>
<td><strong>import numpy as np</strong>&lt;br&gt;<strong>def pre_process(df):</strong>&lt;br&gt;  <strong>df.fillna(value={'review_body': '', 'review_headline': ''}, inplace=True)</strong>&lt;br&gt;  <strong>df.fillna( value={'verified_purchase': 'Unk'}, inplace=True)</strong>&lt;br&gt;  <strong>df.fillna(0, inplace=True)</strong>&lt;br&gt;  <strong>return df</strong>&lt;br&gt;  <strong>df = pre_process(df)</strong>&lt;br&gt;  <strong>df.review_date = pd.to_datetime(df.review_date)</strong>&lt;br&gt;  <strong>df['target'] = np.where(df['star_rating']&gt;=4,1,0)</strong></td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

**Note:** This code replaces null values in the...
### Task | Description | Skills required
--- | --- | ---
| | 'review_body' with an empty string and replaces the 'verified_purchase' column with 'Unk', which means “unknown.” | Data scientist
| | Split the data into training, validation, and test datasets. To keep the distribution of the target label identical across the split sets, you must stratify the sampling by using the scikit-learn library. | Data scientist

```python
from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=2, test_size=0.10, random_state=0)
sss.get_n_splits(df, df['target'])
for train_index, test_index in sss.split(df, df['target']):
    X_train_vallid, X_test = df.iloc[train_index], df.iloc[test_index]
    sss.get_n_splits(X_train_vallid, X_train_vallid['target'])
    for train_index, test_index in sss.split(X_train_vallid, X_train_vallid['target']):
        X_train, X_valid = X_train_vallid.iloc[train_index], X_train_vallid.iloc[test_index]
```

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

### Task | Description | Skills required
--- | --- | ---
| | Build, run, and push the Docker image to Amazon ECR. | ML engineer
| Prepare and push the Docker image. | In the Jupyter notebook, run the commands from the following stories to prepare the Docker image and push it to Amazon ECR. | ML engineer
| Create a repository in Amazon ECR. | | ML engineer

%%sh
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Build a Docker image locally. | **algorithm_name=sagemaker-catboost-github-gpu-img**<br>chmod +x code/train<br>chmod +x code/serve<br>account=$(aws sts get-caller-identity --query Account --output text)<br>  # Get the region defined in the current configuration (default to us-west-2 if none defined<br>  region=$(aws configure get region)<br>  region=${region:-us-east-1}<br>fullname="${account}.dkr.ecr.${region}.amazonaws.com/${algorithm_name}:latest"
aws ecr create-repository --repository-name "${algorithm_name}" > /dev/null | **ML engineer**<br>**Data scientist**<br>**Data scientist**

| Run the Docker image and push it to Amazon ECR. | docker build -t "${algorithm_name}" .
docker tag ${algorithm_name} ${fullname} | **ML engineer**

### Training

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a SageMaker hyperparameter tuning job. | In the Jupyter notebook, run the commands from the following stories to create a SageMaker hyperparameter tuning job using your Docker image. | **Data scientist**

| Create a SageMaker estimator. | Create a **SageMaker estimator** by using the Docker image's name. | **Data scientist**

| | import sagemaker as sage from time import gmtime, strftime<br>sess = sage.Session() | **Data scientist**

---

653
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, HyperparameterTuner
account = sess.boto_session.client('sts').get_caller_identity()['Account']
region = sess.boto_session.region_name
image = '{}.dkr.ecr.{}.amazonaws.com/sagemaker-catboost-github-gpu-img:latest'.format(account, region)
tree_hpo = sage.estimator.Estimator(image, role, 1, 'ml.p3.16xlarge',
train_volume_size = 100,
output_path="s3://{}/sagemaker/DEMO-GPU-Catboost/output".format(bucket),
sagemaker_session=sess) | |

654
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an HPO job.</td>
<td>Create a hyperparameter optimization (HPO) tuning job with parameter ranges and pass the train and validation sets as parameters to the function.</td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

```python
hyperparameter_ranges = {'iterations': IntegerParameter(80000, 130000), 'max_depth': IntegerParameter(6, 10), 'max_ctr_complexity': IntegerParameter(4, 10), 'learning_rate': ContinuousParameter(0.01, 0.5)}

objective_metric_name = 'auc'
metric_definitions = [{'Name': 'auc', 'Regex': 'auc: ([0-9.]+)'}]

tuner = HyperparameterTuner(tree_hpo, objective_metric_name, hyperparameter_ranges, metric_definitions, objective_type='Maximize', max_jobs=50, max_parallel_jobs=2)
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the HPO job.</td>
<td><code>train_location = 's3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/train/'</code>&lt;br&gt;<code>valid_location = 's3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/valid/'</code>&lt;br&gt;<code>tuner.fit({'train': train_location, 'validation': valid_location })</code></td>
<td>Data scientist</td>
</tr>
<tr>
<td>Receive the best performing training job.</td>
<td><code>import sagemaker as sage&lt;br&gt;from time import gmtime, strftime&lt;br&gt;sess = sage.Session()&lt;br&gt;best_job = tuner.best_training_job()</code></td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

**Batch transform**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a SageMaker batch transform job on test data for model prediction.</td>
<td>In the Jupyter notebook, run the commands from the following stories to create the model from your SageMaker hyperparameter tuning job and submit a SageMaker batch transform job on the test data for model prediction.</td>
<td>Data scientist</td>
</tr>
<tr>
<td>Create the SageMaker model.</td>
<td>Create a model in SageMaker model using the best training job.</td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

```
attached_estimator = sage.estimator.Estimator.attach(best_job)
output_path = 's3://'+bucket+'/'+sagemaker/DEMO-GPU-Catboost/data/test-predictions/'
input_path = 's3://'+bucket+'/'+sagemaker/DEMO-GPU-Catboost/data/test/'
```
## Create batch transform job.

Create batch transform job on the test data set.

```python
transformer.transform(input_path,
content_type='text/csv',
split_type='Line')
```

### Skills required
- **Data scientist**

## Analyze the results

### Task

**Read the results and evaluate the model's performance.**

In the Jupyter notebook, run the commands from the following stories to read the results and evaluate the performance of the model on Area Under the ROC Curve (ROC-AUC) and Area Under the Precision Recall Curve (PR-AUC) model metrics.

For more information about this, see Amazon Machine Learning key concepts in the Amazon Machine Learning (Amazon ML) documentation.

### Skills required
- **Data scientist**

**Read the batch transform job results.**

Read the batch transform job results into a data frame.

### Skills required
- **Data scientist**
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>file_name = 's3://'+bucket +'/sagemaker/DEMO-GPU-Catboost/data/test-predictions/file_1.out'</code></td>
<td><code>results = pd.read_csv(file_name, names=['review_id','target','score'], sep='\t', escapechar = '\', quoting=csv.QUOTE_NONE, lineterminator='\n', quotechar='''').dropna()</code></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Evaluate the performance metrics.</td>
<td>Evaluate the performance of the model on ROC-AUC and PR-AUC.</td>
<td>Data scientist</td>
</tr>
</tbody>
</table>

```python
from sklearn import metrics
import matplotlib
import pandas as pd
matplotlib.use('agg', warn=False, force=True)
from matplotlib import pyplot as plt
%matplotlib inline
def analyze_results(labels, predictions):
    precision, recall, thresholds = metrics.precision_recall_curve(labels, predictions)
    auc = metrics.auc(recall, precision)

    fpr, tpr, _ = metrics.roc_curve(labels, predictions)
    roc_auc_score = metrics.roc_auc_score(labels, predictions)

    print('Neural-Nets: ROC auc=%.3f' % (roc_auc_score))
    plt.plot(fpr, tpr, label="data 1, auc=") + str(roc_auc_score))
    plt.xlabel('1-Specificity')
    plt.legend(loc=4)
    plt.show()

    lr_precision, lr_recall, _ = metrics.precision_recall_curve(labels, predictions)
    lr_auc = metrics.auc(lr_recall, lr_precision)
    print('Neural-Nets: PR auc=%.3f' % (lr_auc))
    # plot the precision-recall curves
    no_skill = len(labels[labels==1.0]) / len(labels)
```
### Related resources

- Train and host Scikit-Learn models in Amazon SageMaker by building a Scikit Docker container

### Additional information

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

#### Install Python with aws-cli.

```
FROM amazonlinux:1

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

#### Install the Python packages

```
RUN pip-3.6 install --no-cache-dir --upgrade pip && \
    pip3 install --no-cache-dir --upgrade setuptools && \
    pip3 install Cython && \
    pip3 install --no-cache-dir numpy==1.16.0 scipy==1.4.1 scikit-learn==0.20.3 pandas==0.24.2 \
    flask gevent gunicorn boto3 s3fs matplotlib joblib catboost==0.20.2
```

#### Install CUDA and CuDNN

---

```python
import matplotlib.pyplot as plt
def analyze_results(results['target'].values, results['score'].values):
    lr_recall = # Calculate recall values
    lr_precision = # Calculate precision values
    auc = # Calculate AUC
    plt.plot([0, 1],
              [no_skill, no_skill],
              linestyle='--', label='No Skill')
    plt.plot(lr_recall,
              lr_precision, marker='.',
              label='Neural-Nets')
    # axis labels
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    # show the legend
    plt.legend()
    # show the plot
    plt.show()
    return auc
```

---

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')</td>
<td>plt.plot(lr_recall, lr_precision, marker='.', label='Neural-Nets') # axis labels plt.xlabel('Recall') plt.ylabel('Precision') # show the legend plt.legend() # show the plot plt.show() return auc analyze_results(results['target'].values, results['score'].values)</td>
</tr>
</tbody>
</table>
Use SageMaker Processing for distributed feature engineering of terabyte-scale ML datasets

**Environment:** Production  
**Technologies:** Machine learning & AI, Big data  
**AWS services:** Amazon SageMaker

**Summary**

Many terabyte-scale or larger datasets often consist of a hierarchical folder structure, and the files in the dataset sometimes share interdependencies. For this reason, machine learning (ML) engineers and data scientists must make thoughtful design decisions to prepare such data for model training and inference. This pattern demonstrates how you can use manual macrosharding and microsharding techniques in combination with Amazon SageMaker Processing and virtual CPU (vCPU) parallelization to efficiently scale feature engineering processes for complicated big data ML datasets.
This pattern defines **macrosharding** as the splitting of data directories across multiple machines for processing, and **microsharding** as the splitting of data on each machine across multiple processing threads. The pattern demonstrates these techniques by using Amazon SageMaker with sample time-series waveform records from the **PhysioNet MIMIC-III** dataset. By implementing the techniques in this pattern, you can minimize the processing time and costs for feature engineering while maximizing resource utilization and throughput efficiency. These optimizations rely on distributed SageMaker Processing on Amazon Elastic Compute Cloud (Amazon EC2) instances and vCPUs for similar, large datasets, regardless of data type.

**Prerequisites and limitations**

**Prerequisites**

- Access to SageMaker notebook instances or SageMaker Studio, if you want to implement this pattern for your own dataset. If you are using Amazon SageMaker for the first time, see Get started with Amazon SageMaker in the AWS documentation.
- SageMaker Studio, if you want to implement this pattern with the **PhysioNet MIMIC-III** sample data.
- The pattern uses SageMaker Processing, but doesn’t require any experience running SageMaker Processing jobs.

**Limitations**

- This pattern is well suited to ML datasets that include interdependent files. These interdependencies benefit the most from manual macrosharding and running multiple, single-instance SageMaker Processing jobs in parallel. For datasets where such interdependencies do not exist, the **ShardedByS3Key** feature in SageMaker Processing might be a better alternative to macrosharding, because it sends sharded data to multiple instances that are managed by the same Processing job. However, you can implement this pattern’s microsharding strategy in both scenarios to best utilize instance vCPUs.

**Product versions**

- Amazon SageMaker Python SDK version 2

**Architecture**

**Target technology stack**

- Amazon Simple Storage Service (Amazon S3)
- Amazon SageMaker

**Target architecture**

*Macrosharding and distributed EC2 instances*

The 10 parallel processes represented in this architecture reflect the structure of the MIMIC-III dataset. (Processes are represented by ellipses for diagram simplification.) A similar architecture applies to any dataset when you use manual macrosharding. In the case of MIMIC-III, you can use the dataset's raw structure to your advantage by processing each patient group folder separately, with minimal effort. In the following diagram, the record groups block appears on the left (1). Given the distributed nature of the data, it makes sense to shard by patient group.
However, manually sharding by patient group means that a separate Processing job is required for each patient group folder, as you can see in the middle section of the diagram (2), instead of a single Processing job with multiple EC2 instances. Because MIMIC-III's data includes both binary waveform files and matching text-based header files, and there is a required dependency on the wfdb library for binary data extraction, all the records for a specific patient must be made available on the same instance. The only way to be certain that each binary waveform file's associated header file is also present is to implement manual sharding to run each shard within its own Processing job, and to specify `s3_data_distribution_type='FullyReplicated'` when you define the Processing job input.

Alternatively, if all data were available in a single directory and no dependencies existed between files,
A more suitable option might be to launch a single Processing job with multiple EC2 instances and `s3_data_distribution_type='ShardedByS3Key'` specified. Specifying ShardedByS3Key as the Amazon S3 data distribution type directs SageMaker to manage data sharding automatically across instances.

Launching a Processing job for each folder is a cost-efficient way to preprocess the data, because running multiple instances concurrently saves time. For additional cost and time savings, you can use microsharding within each Processing job.

**Microsharding and parallel vCPUs**

Within each Processing job, the grouped data is further divided to maximize use of all available vCPUs on the SageMaker fully managed EC2 instance. The blocks in the middle section of the diagram (2) depict what happens within each primary Processing job. The contents of the patient record folders are flattened and divided evenly based on the number of available vCPUs on the instance. After the folder contents are divided, the evenly sized set of files are distributed across all vCPUs for processing. When processing is complete, the results from each vCPU are combined into a single data file for each Processing job.

In the attached code, these concepts are represented in the following section of the `src/feature-engineering-pass1/preprocessing.py` file.

```python
def chunks(lst, n):
    """
    Yield successive n-sized chunks from lst.
    
    :param lst: list of elements to be divided
    :param n: number of elements per chunk
    :type lst: list
    :type n: int
    :return: generator comprising evenly sized chunks
    :rtype: class 'generator'
    """
    for i in range(0, len(lst), n):
        yield lst[i:i + n]

# Generate list of data files on machine
data_dir = input_dir
d_subs = next(os.walk(os.path.join(data_dir, '.')))[1]
file_list = []
for ds in d_subs:
    file_list.extend(os.listdir(os.path.join(data_dir, ds, '.')))
dat_list = [os.path.join(re.split('_|\.', f)[0].replace('n', ''), f[:-4]) for f in file_list if f[-4:] == '.dat']

# Split list of files into sub-lists
cpu_count = multiprocessing.cpu_count()
splits = int(len(dat_list) / cpu_count)
if splits == 0: splits = 1
dat_chunks = list(chunks(dat_list, splits))

# Parallelize processing of sub-lists across CPUs
ws_df_list = Parallel(n_jobs=-1, verbose=0)(delayed(run_process)(dc) for dc in dat_chunks)

# Compile and pickle patient group dataframe
ws_df_group = pd.concat(ws_df_list).reset_index().rename(columns={'index': 'signal'}).to_json(os.path.join(output_dir, group_data_out))
```

A function, `chunks`, is first defined to consume a given list by dividing it into evenly sized chunks of length `n` and by returning these results as a generator. Next, the data is flattened across patient folders by compiling a list of all binary waveform files that are present. After this is done, the number of vCPUs
available on the EC2 instance is obtained. The list of binary waveform files is evenly divided across these vCPUs by calling chunks, and then each waveform sublist is processed on its own vCPU by using joblib's Parallel class. Results are automatically combined into a single list of dataframes by the Processing job, which SageMaker then processes further before writing it to Amazon S3 upon job completion. In this example, there are 10 files written to Amazon S3 by the Processing jobs (one for each job).

When all the initial Processing jobs are complete, a secondary Processing job, which is shown in the block to the right of the diagram (3) combines the output files produced by each primary Processing job and writes the combined output to Amazon S3 (4).

Tools

Tools

- **Python** – The sample code used for this pattern is Python (version 3).
- **SageMaker Studio** – Amazon SageMaker Studio is a web-based, integrated development environment (IDE) for machine learning that lets you build, train, debug, deploy, and monitor your machine learning models. You run SageMaker Processing jobs by using Jupyter notebooks inside SageMaker Studio.
- **SageMaker Processing** – Amazon SageMaker Processing provides a simplified way to run your data processing workloads. In this pattern, the feature engineering code is implemented at scale by using SageMaker Processing jobs.

Code

The attached .zip file provides the complete code for this pattern. The following section describes the steps to build the architecture for this pattern. Each step is illustrated by sample code from the attachment.

Epics

Set up your SageMaker Studio environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Amazon SageMaker Studio.</td>
<td>Onboard to SageMaker Studio in your AWS account by following the directions provided in the Amazon SageMaker documentation.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>
| Install the wget utility.                 | Install wget if you onboarded with a new SageMaker Studio configuration or if you’ve never used these utilities in SageMaker Studio before. To install, open a terminal window in the SageMaker Studio console and run the following command:  

```
sudo yum install wget
```
| Data scientist, ML engineer              |
| Download and unzip the sample code.      | Download the attachments.zip file in | Data scientist, ML engineer |
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachments section. In a terminal window, navigate to the folder where you downloaded the file and extract its contents:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```shell
unzip attachment.zip
```

Navigate to the folder where you extracted the .zip file, and extract the contents of the Scaled-Processing.zip file.

```shell
unzip Scaled-Processing.zip
```

Download the sample dataset from physionet.org and upload it to Amazon S3.

Run the `get_data.ipynb` Jupyter notebook within the folder that contains the Scaled-Processing files. This notebook downloads a sample MIMIC-III dataset from physionet.org and uploads it to your SageMaker Studio session bucket in Amazon S3.

Data scientist, ML engineer

### Configure the first preprocessing script

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flatten the file hierarchy across all subdirectories.</td>
<td>In large datasets such as MIMIC-III, files are often distributed across multiple subdirectories even within a logical parent group. Your script should be configured to flatten all group files across all subdirectories, as the following code demonstrates.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>

```python
# Generate list of .dat files on machine
data_dir = input_dir
d_subs = next(os.walk(os.path.join(data_dir, '.')))[1]
file_list = []
for ds in d_subs:
    file_list.extend(os.listdir(os.path.join(data_dir, ds, '.')))
dat_list = [os.path.join(re.split('_|\.', f)[0].replace('n', ''), 'dt')]
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Divide files into subgroups based on vCPU count.</strong></td>
<td>Files should be divided into evenly sized subgroups, or chunks, depending on the number of vCPUs present on the instance that runs the script. For this step, you can implement code similar to the following.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>
| | # Split list of files into sub-lists  
<code>cpu_count = multiprocessing.cpu_count()
splits = int(len(dat_list) / cpu_count)
if splits == 0: splits = 1
dat_chunks = list(chunks(dat_list, splits))</code> | |
| **Parallelize processing of subgroups across vCPUs.** | Script logic should be configured to process all subgroups in parallel. To do this, use the Joblib library's Parallel class and delayed method as follows. | Data scientist, ML engineer |
| | # Parallelize processing of sub-lists across CPUs  
<code>ws_df_list = Parallel(n_jobs=-1, verbose=0)(delayed(run_process)(dc) for dc in dat_chunks)</code> | |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save single file group output to Amazon S3.</td>
<td>When parallel vCPU processing is complete, the results from each vCPU should be combined and uploaded to the file group's S3 bucket path. For this step, you can use code similar to the following.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>
| | # Compile and pickle patient group dataframe  
ws_df_group =  
pd.concat(ws_df_list)  
ws_df_group =  
ws_df_group.reset_index().rename(columns={'index': 'signal'})  
ws_df_group.to_json(os.path.join(output_dir, group_data_out)) | |

### Configure the second preprocessing script

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combine data files produced across all Processing jobs that ran the first script.</td>
<td>The previous script outputs a single file for each SageMaker Processing job that processes a group of files from the dataset. Next, you need to combine these output files into a single object and write a single output dataset to Amazon S3. This is demonstrated in the src/feature-engineering-pass1p5/preprocessing.py file, which is provided in the attachment, as follows.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>
| | def write_parquet(wavs_df, path):  
"""  
Write waveform summary dataframe to S3 in parquet format.  
"""  
:param wavs_df: waveform summary dataframe  
:param path: S3 directory prefix  
:type wavs_df: pandas dataframe  
:type path: str  
:return: None  
"""  
extra_args =  
{"ServerSideEncryption": "aws:kms"}  
wr.s3.to_parquet( | |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=wavs_df, path=path, compression='snappy', s3_additional_kwargs=extra_args)</td>
<td>def combine_data(): &quot;&quot; Get combined data and write to parquet. :return: waveform summary dataframe :rtype: pandas dataframe &quot;&quot; wavs_df = get_data() wavs_df = normalize_signal_names(wavs_df) write_parquet(wavs_df, &quot;s3://{{}}/{{}}.format(bucket_xform, dataset_prefix, pass1p5out_data)) return wavs_df</td>
<td>Data scientist, ML engineer</td>
</tr>
<tr>
<td>wavs_df = combine_data()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Run Processing jobs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the first Processing job.</td>
<td>To perform macrosharding, run a separate Processing job for each file group. Microsharding is performed inside each Processing job, because each job runs your first script. The following code demonstrates how to launch a Processing job for each file group directory in the following snippet (included in notebooks/FeatExtract_Pass1.ipynb).</td>
<td>Data scientist, ML engineer</td>
</tr>
<tr>
<td>pat_groups = list(range(30,40)) ts = str(int(time.time())) for group in pat_groups: sklearn_processor = SKLearnProcessor(framework_version='0.20.0', role=role,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
|      | `instance_type='ml.m5.4xlarge',
     instance_count=1,
     volume_size_in_gb=5)
     sklearn_processor.run(
     code='..../src/
     feature-engineering-pass1/
     preprocessing.py',
     job_name='--'.join(['scaled-
     processing-p1', str(group),
     ts]),
     arguments=[
     "input_path", "/
     opt/ml/processing/input",
     "output_path",
     "/opt/ml/processing/
     output",
     "group_data_out",
     "ws_df_group.json"
     ],
     inputs=
     [
     ProcessingInput(
     source=f's3://
     {sess.default_bucket()}/
     data_inputs/{group}',
     destination="/opt/ml/
     processing/input",
     s3_data_distribution_type='FullyReplicated'
     ),
     ],
     outputs=
     [
     ProcessingOutput(
     source="/opt/ml/processing/output",
     destination=f's3://
     {sess.default_bucket()}/
     data_outputs/{group}"
     ),
     ],
     wait=False |

670
**Task**

Run the second Processing job.

**Description**

To combine the outputs generated by the first set of processing jobs and perform any additional computations for preprocessing, you run your second script by using a single SageMaker Processing job. The following code demonstrates this (included in notebooks/FeatExtract_Passip5.ipynb).

```
import time

ts = str(int(time.time()))
bucket = sess.default_bucket()

sklearn_processor = SKLearnProcessor(framework_version='0.20.0',
                                       role=role,
                                       instance_type='ml.t3.2xlarge',
                                       instance_count=1,
                                       volume_size_in_gb=5)

sklearn_processor.run(
    code='../src/feature-engineering-pass1p5/preprocessing.py',
    job_name='-'.join(['scaled-processing', 'p1p5', ts]),
    arguments=['bucket', bucket,
               'pass1out_prefix', 'data_outputs',
               'pass1out_data', 'ws_df_group.json',
               'passip5out_data', 'waveform_summary.parquet',
               'statsdata_name', 'signal_stats.csv'],
    wait=True)
```

**Skills required**

Data scientist, ML engineer

---

**Related resources**

- Onboard to Amazon SageMaker Studio Using Quick Start (SageMaker documentation)
- Process Data (SageMaker documentation)
- Data Processing with scikit-learn (SageMaker documentation)
- joblib.Parallel documentation
Visualize AI/ML model results using Flask and AWS Elastic Beanstalk

Created by Chris Caudill (AWS) and Durga Sury (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Machine learning &amp; AI; Analytics; DevOps; Websites &amp; web apps</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon Comprehend; AWS Elastic Beanstalk</td>
</tr>
</tbody>
</table>

Summary

Visualizing output from artificial intelligence and machine learning (AI/ML) services often requires complex API calls that must be customized by your developers and engineers. This can be a drawback if your analysts want to quickly explore a new dataset.

You can enhance the accessibility of your services and provide a more interactive form of data analysis by using a web-based user interface (UI) that enables users to upload their own data and visualize the model results in a dashboard.

This pattern uses Flask and Plotly to integrate Amazon Comprehend with a custom web application and visualize sentiments and entities from user-provided data. The pattern also provides the steps to deploy an application by using AWS Elastic Beanstalk. You can adapt the application by using Amazon Web Services (AWS) AI services or with a custom trained model hosted on an endpoint (for example, an Amazon SageMaker endpoint).

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured on your local machine. For more information about this, see Configuration basics in the AWS CLI documentation. You can also use

Attachments

attachment.zip


• MIMIC-III Waveform Database license
an AWS Cloud9 integrated development environment (IDE); for more information about this, see Python tutorial for AWS Cloud9 and Previewing running applications in the AWS Cloud9 IDE in the AWS Cloud9 documentation.

- An understanding of Flask's web application framework. For more information about Flask, see the Quickstart in the Flask documentation.
- Python version 3.6 or later, installed and configured. You can install Python by following the instructions from Setting up your Python development environment in the AWS Elastic Beanstalk documentation.
- Elastic Beanstalk Command Line Interface (EB CLI), installed and configured. For more information about this, see Install the EB CLI and Configure the EB CLI from the AWS Elastic Beanstalk documentation.

Limitations

- This pattern's Flask application is designed to work with .csv files that use a single text column and are restricted to 200 rows. The application code can be adapted to handle other file types and data volumes.
- The application doesn't consider data retention and continues to aggregate uploaded user files until they are manually deleted. You can integrate the application with Amazon Simple Storage Service (Amazon S3) for persistent object storage or use a database such as Amazon DynamoDB for serverless key-value storage.
- The application only considers documents in the English language. However, you can use Amazon Comprehend to detect a document's primary language. For more information about the supported languages for each action, see API reference in the Amazon Comprehend documentation.
- A troubleshooting list that contains common errors and their solutions is available in the Additional information section.

Architecture

Flask application architecture

Flask is a lightweight framework for developing web applications in Python. It is designed to combine Python’s powerful data processing with a rich web UI. The pattern's Flask application shows you how to build a web application that enables users to upload data, sends the data to Amazon Comprehend for inference, and then visualizes the results. The application has the following structure:

- static – Contains all the static files that support the web UI (for example, JavaScript, CSS, and images)
- templates – Contains all of the application's HTML pages
- userData – Stores uploaded user data
- application.py – The Flask application file
- comprehend_helper.py – Functions to make API calls to Amazon Comprehend
- config.py – The application configuration file
- requirements.txt – The Python dependencies required by the application

The application.py script contains the web application's core functionality, which consists of four Flask routes. The following diagram shows these Flask routes.
• `/` is the application's root and directs users to the `upload.html` page (stored in the `templates` directory).

• `/saveFile` is a route that is invoked after a user uploads a file. This route receives a POST request via an HTML form, which contains the file uploaded by the user. The file is saved in the `userData` directory and the route redirects users to the `/dashboard` route.

• `/dashboard` sends users to the `dashboard.html` page. Within this page's HTML, it runs the JavaScript code in `static/js/core.js` that reads data from the `/data` route and then builds visualizations for the page.

• `/data` is a JSON API that presents the data to be visualized in the dashboard. This route reads the user-provided data and uses the functions in `comprehend_helper.py` to send the user data to Amazon Comprehend for sentiment analysis and named entity recognition (NER). Amazon Comprehend's response is formatted and returned as a JSON object.

**Deployment architecture**
For more information about design considerations for applications deployed using Elastic Beanstalk on the AWS Cloud, see Design considerations in the AWS Elastic Beanstalk documentation.

**Technology stack**

- Amazon Comprehend
- Elastic Beanstalk
- Flask

**Automation and scaling**

Elastic Beanstalk deployments are automatically set up with load balancers and auto scaling groups. For more configuration options, see Configuring Elastic Beanstalk environments in the AWS Elastic Beanstalk documentation.

**Tools**

- **AWS Command Line Interface (AWS CLI)** – AWS CLI is a unified tool that provides a consistent interface for interacting with all parts of AWS.
- **Amazon Comprehend** - Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents without requiring special preprocessing.
- **AWS Elastic Beanstalk** – Elastic Beanstalk helps you quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications.
- **Elastic Beanstalk CLI (EB CLI)** – EB CLI is a command line interface for AWS Elastic Beanstalk that provides interactive commands to simplify creating, updating, and monitoring environments from a local repository.
- **Flask** – The Flask framework performs data processing and API calls using Python and offers interactive web visualization with Plotly.

**Code**

675
The code for this pattern is available in the GitHub Visualize AI/ML model results using Flask and AWS Elastic Beanstalk repository.

## Epics

### Set up the Flask application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Clone the GitHub repository. | Pull the application code from the GitHub Visualize AI/ML model results using Flask and AWS Elastic Beanstalk repository by running the following command:  
  
git clone git@github.com:aws-samples/aws-comprehend-elasticbeanstalk-for-flask.git  

**Note:** Make sure that you configure your SSH keys with GitHub.                                                                                                                                                                                                                     | Developer                 |
| Install the Python modules.  | After you clone the repository, a new local aws-comprehend-elasticbeanstalk-for-flask directory is created. In that directory, the requirements.txt file contains the Python modules and versions that run the application. Use the following commands to install the modules:  
  
cd aws-comprehend-elasticbeanstalk-for-flask
  pip install -r requirements.txt  

| Test the application locally.| Start the Flask server by running the following command:  
  
python application.py  

This returns information about the running server. You should be able to access the application by opening a browser and visiting http://localhost:5000  

**Note:** If you’re running the application in an AWS Cloud9 | Python developer          |
### Deploy the Elastic Beanstalk application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Elastic Beanstalk application.</td>
<td>To launch your project as an Elastic Beanstalk application, run the following command from your application's root directory:</td>
<td>Architect, Developer</td>
</tr>
<tr>
<td></td>
<td><code>eb init -p python-3.6 comprehend_flask --region us-east-1</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>comprehend_flask</code> is the name of the Elastic Beanstalk application and can be changed according to your requirements.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You can replace the AWS Region with a Region of your choice. The default Region in AWS CLI is used if you don't specify a Region.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The application was built with Python version 3.6. You might encounter errors if you use other Python versions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Run the <code>eb init -i</code> command for more deployment configuration options.</td>
<td></td>
</tr>
<tr>
<td>Deploy the Elastic Beanstalk environment.</td>
<td>Run the following command from the application's root directory:</td>
<td>Architect, Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
</tbody>
</table>
| **Authorize your deployment to use Amazon Comprehend.**              | **Note**: comprehend-flask-env is the name of the Elastic Beanstalk environment and can be changed according to your requirements. The name can only contain letters, numbers, and dashes. Although your application might be successfully deployed, you should also provide your deployment with access to Amazon Comprehend. **ComprehendFullAccess** is an AWS managed policy that provides the deployed application with permissions to make API calls to Amazon Comprehend. **Important**: aws-elasticbeanstalk-ec2-role is created when your application deploys. You must complete the deployment process before you can attach the AWS Identity and Access Management (IAM) policy.  

Attach the **ComprehendFullAccess** policy to **aws-elasticbeanstalk-ec2-role** (this role is automatically created for your deployment's Amazon Elastic Compute Cloud (Amazon EC2) instances) by running the following command: 

```bash
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/ComprehendFullAccess --role-name aws-elasticbeanstalk-ec2-role
```

**Important**: aws-elasticbeanstalk-ec2-role is created when your application deploys. You must complete the deployment process before you can attach the AWS Identity and Access Management (IAM) policy. | Developer, Security architect |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit your deployed application.</td>
<td>After your application successfully deploys, you can visit it by running the <code>eb open</code> command. You can also run the <code>eb status</code> command to receive details about your deployment. The deployment URL is listed under CNAME.</td>
<td>Architect, Developer</td>
</tr>
</tbody>
</table>

### (Optional) Customize the application to your ML model

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorize Elastic Beanstalk to access the new model.</td>
<td>Make sure that Elastic Beanstalk has the required access permissions for your new model endpoint. For example, if you use an Amazon SageMaker endpoint, your deployment needs to have permission to invoke the endpoint. For more information about this, see <code>InvokeEndpoint</code> in the Amazon SageMaker documentation.</td>
<td>Developer, Security architect</td>
</tr>
</tbody>
</table>
| Send the user data to a new model. | To change the underlying ML model in this application, you must change the following files:  
  • `comprehend_helper.py` – This is the Python script that connects with Amazon Comprehend, processes the response, and returns the final result to the application. In this script, you can either route the data to another AI service on the AWS Cloud or you can send the data to a custom model endpoint. We recommend that you also format the results in this script for logical separation and the reusability of this pattern.  
  • `application.py` – If you change the name of the `comprehend_helper.py` script or functions, you need to update the application | Data scientist |
### Task 1: Update the dashboard visualizations

Typically, incorporating a new ML model means that visualizations must be updated to reflect the new results. These changes are made in the following files:

- **templates/dashboard.html** – The prebuilt application only accounts for two basic visualizations. The entire layout of the page can be adjusted in this file.
- **static/js/core.js** – This script captures the formatted output of the Flask server's `/data` route and uses Plotly to create visualizations. You can add or update the page's charts.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the dashboard visualizations.</td>
<td>Typically, incorporating a new ML model means that visualizations must be updated to reflect the new results. These changes are made in the following files:</td>
<td>Web developer</td>
</tr>
</tbody>
</table>

### (Optional) Deploy the updated application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update your application's requirements file.</td>
<td>Before sending changes to Elastic Beanstalk, update the <code>requirements.txt</code> file to reflect any new Python modules by running the following command in your application's root directory:</td>
<td>Python developer</td>
</tr>
<tr>
<td></td>
<td><code>pip freeze &gt; requirements.txt</code></td>
<td></td>
</tr>
<tr>
<td>Redeploy the Elastic Beanstalk environment.</td>
<td>To ensure that your application changes are reflected in your Elastic Beanstalk deployment, navigate to your application's root directory and run the following command:</td>
<td>Systems administrator, Architect</td>
</tr>
<tr>
<td></td>
<td><code>eb deploy</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This sends the most recent version of the application's code to your existing Elastic Beanstalk deployment.</td>
<td></td>
</tr>
</tbody>
</table>
Related resources

- Call an Amazon SageMaker model endpoint using Amazon API Gateway and AWS Lambda
- Deploying a Flask application to Elastic Beanstalk
- EB CLI command reference
- Setting up your Python development environment

Additional information

Troubleshooting list

The following are six common errors and their solutions.

**Error 1**

Unable to assume role "arn:aws:iam::xxxxxxxxxx:role/aws-elasticbeanstalk-ec2-role". Verify that the role exists and is configured correctly.

**Solution:** If this error occurs when you run `eb create`, create a sample application on the Elastic Beanstalk console to create the default instance profile. For more information about this, see Creating an Elastic Beanstalk environment in the AWS Elastic Beanstalk documentation.

**Error 2**

Your WSGIPath refers to a file that does not exist.

**Solution:** This error occurs in deployment logs because Elastic Beanstalk expects the Flask code to be named `application.py`. If you chose a different name, run `eb config` and edit the WSGIPath as shown in the following code sample:

```python
aws:elasticbeanstalk:container:python:
    NumProcesses: '1'
    NumThreads: '15'
    StaticFiles: /static/=static/
    WSGIPath: application.py
```

Make sure that you replace `application.py` with your file name.

You can also leverage Gunicorn and a Procfile. For more information about this approach, see Configuring the WSGI server with a Procfile in the AWS Elastic Beanstalk documentation.

**Error 3**

Target WSGI script '/opt/python/current/app/application.py' does not contain WSGI application 'application'.

**Solution:** Elastic Beanstalk expects the variable that represents your Flask application to be named `application`. Make sure that the `application.py` file uses `application` as the variable name:

```python
application = Flask(__name__)
```
Error 4

The EB CLI cannot find your SSH key file for keyname

Solution: Use the EB CLI to specify which key pair to use or to create a key pair for your deployment’s EC2 instances. To resolve the error, run `eb init -i` and one of the options will ask:

Do you want to set up SSH for your instances?

Respond with Y to either create a key pair or specify an existing key pair.

Error 5

I’ve updated my code and redeployed but my deployment is not reflecting my changes.

Solution: If you’re using a Git repository with your deployment, make sure that you add and commit your changes before redeploying.

Error 6

You are previewing the Flask application from an AWS Cloud9 IDE and run into errors.

Solution: For more information about this, see Previewing running applications in the AWS Cloud9 IDE in the AWS Cloud9 documentation.

Natural language processing using Amazon Comprehend

By choosing to use Amazon Comprehend, you can detect custom entities in individual text documents by running real-time analysis or asynchronous batch jobs. Amazon Comprehend also enables you to train custom entity recognition and text classification models that can be used in real time by creating an endpoint.

This pattern uses asynchronous batch jobs to detect sentiments and entities from an input file that contains multiple documents. The sample application provided by this pattern is designed for users to upload a .csv file containing a single column with one text document per row. The `comprehend_helper.py` file in the GitHub Visualize AI/ML model results using Flask and AWS Elastic Beanstalk repository reads the input file and sends the input to Amazon Comprehend for processing.

BatchDetectEntities

Amazon Comprehend inspects the text of a batch of documents for named entities and returns the detected entity, location, type of entity, and a score that indicates Amazon Comprehend’s level of confidence. A maximum of 25 documents can be sent in one API call, with each document smaller than 5,000 bytes in size. You can filter the results to show only certain entities based on the use case. For example, you could skip the ‘quantity’ entity type and set a threshold score for the detected entity (for example, 0.75). We recommend that you explore the results for your specific use case before choosing a threshold value. For more information about this, see BatchDetectEntities in the Amazon Comprehend documentation.

BatchDetectSentiment

Amazon Comprehend inspects a batch of incoming documents and returns the prevailing sentiment for each document (POSITIVE, NEUTRAL, MIXED, or NEGATIVE). A maximum of 25 documents can be sent in one API call, with each document smaller than 5,000 bytes in size. Analyzing the sentiment is
straightforward and you choose the sentiment with the highest score to be displayed in the final results. For more information about this, see BatchDetectSentiment in the Amazon Comprehend documentation.

**Flask configuration handling**

Flask servers use a series of configuration variables to control how the server runs. These variables can contain debug output, session tokens, or other application settings. You can also define custom variables that can be accessed while the application is running. There are multiple approaches for setting configuration variables.

In this pattern, the configuration is defined in config.py and inherited within application.py.

- config.py contains the configuration variables that are set up on the application’s startup. In this application, a DEBUG variable is defined to tell the application to run the server in debug mode.
  
  **Note:** Debug mode should not be used when running an application in a production environment.

- UPLOAD_FOLDER is a custom variable that is defined to be referenced later in the application and inform it where uploaded user data should be stored.

- application.py initiates the Flask application and inherits the configuration settings defined in config.py. This is performed by the following code:

```python
application = Flask(__name__)
application.config.from_pyfile('config.py')
```

**More patterns**

- Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker using AWS Developer Tools (p. 1131)
- Perform advanced analytics using Amazon Redshift ML (p. 67)
Management & governance

Topics

- Identify and alert when Amazon Kinesis Data Firehose resources are not encrypted with an AWS KMS key (p. 684)
- Automate adding or updating Windows registry entries using AWS Systems Manager (p. 687)
- Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows (p. 691)
- Copy AWS Service Catalog products across different AWS accounts and AWS Regions (p. 699)
- Create alarms for custom metrics using Amazon CloudWatch anomaly detection (p. 705)
- Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization (p. 709)
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK (p. 713)
- Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions (p. 728)
- Migrate an AWS member account from AWS Organizations to AWS Control Tower (p. 733)
- More patterns (p. 741)

Identify and alert when Amazon Kinesis Data Firehose resources are not encrypted with an AWS KMS key

Created by Ram Kandaswamy (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Management &amp; governance; Analytics; Big data; Cloud-native; Infrastructure; Security, identity, compliance</td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS CloudTrail; Amazon CloudWatch; AWS Identity and Access Management; Amazon Kinesis; AWS Lambda; Amazon SNS</td>
</tr>
</tbody>
</table>

Summary

For compliance, some organizations must have encryption enabled on data delivery resources such as Amazon Kinesis Data Firehose. This pattern shows a way to monitor, detect, and notify when resources are out of compliance.

To maintain the encryption requirement, this pattern can be used on Amazon Web Services (AWS) to provide automated monitoring and detection of Kinesis Data Firehose delivery resources that are not encrypted with AWS Key Management Service (AWS KMS) key. The solution sends alert notifications, and it can be extended to perform automatic remediation. This solution can be applied to an individual account or a multiple-account environment, such as an environment using AWS Landing Zone or AWS Control Tower.

Prerequisites and limitations

Prerequisites
• Firehose delivery stream
• Sufficient permissions and familiarity with AWS CloudFormation, which is used in this infrastructure automation

Limitations
The solution is not real time because it uses AWS CloudTrail events for detection, and there is a delay between the time an unencrypted resource is created and the notification is sent.

Architecture

Target technology stack
The solution uses serverless technology and the following services:
• AWS CloudTrail
• Amazon CloudWatch
• AWS Command Line Interface (AWS CLI)
• AWS Identity and Access Management (IAM)
• Amazon Kinesis Data Firehose
• AWS Lambda
• Amazon Simple Notification Service (Amazon SNS)

Target architecture

1. A user creates or modifies Kinesis Data Firehose.
2. A CloudTrail event is detected and matched.
3. Lambda is invoked.
4. Noncompliant resources are identified.
5. Email notification is sent.

Automation and scale
Using AWS CloudFormation StackSets, you can apply this solution to multiple AWS Regions or accounts with a single command.
Tools

- **AWS CloudTrail** – AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and API operations.

- **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources.

- **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands in your command line shell.

- **IAM** – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

- **Amazon Kinesis Data Firehose** – Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data. With Kinesis Data Firehose, you don't need to write applications or manage resources. You configure your data producers to send data to Kinesis Data Firehose, and it automatically delivers the data to the destination that you specified.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).

Epics

Enforce encryption for compliance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy AWS CloudFormation</td>
<td>In the AWS CLI, use the firehose-encryption-checker.yaml template (attached) to create the stack set by running the following command. Provide a valid Amazon SNS topic Amazon Resource Name (ARN) for the parameter. The deployment should successfully create CloudWatch Events rules, the Lambda function, and an IAM role with the necessary permissions as described in the template.</td>
<td></td>
</tr>
</tbody>
</table>
Create stack instances. | Stacks need to be created in the AWS Regions of your choice as well as in one or more accounts. To create stack instances, run the following command, replacing the stack name, account numbers, and Regions with your own.

```
aws cloudformation create-stack-instances
--stack-set-name my-stack-set
--accounts 123456789012 223456789012
--regions us-east-1 us-east-2 us-west-1 us-west-2
--operation-preferences FailureToleranceCount=1
```

Skills required: Cloud architect, Systems administrator

---

Related resources

- Working with AWS CloudFormation StackSets
- What is Amazon CloudWatch Events?

---

Additional information

AWS Config does not support the Firehose delivery stream resource type, so an AWS Config rule cannot be used in the solution.

---

Attachments

attachment.zip

---

Automate adding or updating Windows registry entries using AWS Systems Manager

Created by Appasaheb Bagali (AWS)

---

Created by: AWS

Environment: PoC or pilot

Technologies: Cloud-native; DevOps; Infrastructure; Modernization; Security, identity, compliance; Management & governance

Workload: Microsoft

AWS services: AWS Systems Manager
Summary

AWS Systems Manager is a remote management tool for Amazon Elastic Compute Cloud (Amazon EC2) instances. Systems Manager provides visibility and control over your infrastructure on Amazon Web Services. This versatile tool can be used to fix Windows registry changes that are identified as vulnerabilities by the security vulnerability scan report.

This pattern covers the steps to keep your EC2 instances running Windows operating system secure by automating registry changes that are recommended for the safety of your environment. The pattern uses the Run command to run a Command document. The code is attached, and a portion of it is included in the Code section.

Prerequisites and limitations

- An active AWS account
- An AWS Identity and Access Management (IAM) user with access to the EC2 instance and Systems Manager

Architecture

Target technology stack

- A virtual private cloud (VPC), with two subnets and a network address translation (NAT) gateway
- A Systems Manager Command document to add or update the registry name and value
- Systems Manager Run Command to run the Command document on the specified EC2 instances

Target architecture

![Architecture Diagram]
Tools

IAM policies and roles – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

Amazon Simple Storage Service – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers. In this pattern, an S3 bucket is used to store the Systems Manager logs.

AWS Systems Manager – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting (or taking corrective action on) any policy violations it detects.

AWS Systems Manager Command document – AWS Systems Manager Command documents are used by Run Command. Most Command documents are supported on all Linux and Windows Server operating systems supported by Systems Manager.

AWS Systems Manager Run Command – AWS Systems Manager Run Command gives you a way to manage the configuration of your managed instances remotely and securely. Using Run Command, you can automate common administrative tasks and perform one-time configuration changes at scale.

Code

You can use the following example code to add or update a Microsoft Windows registry name to Version, registry path to HKCU:\Software\ScriptingGuys\Scripts, and value to 2.

```powershell
#Windows registry path which needs to add/update
$registryPath = 'HKCU:\Software\ScriptingGuys\Scripts'
#Windows registry Name which needs to add/update
$Name = 'Version'
#Windows registry value which needs to add/update
$value = 2
# Test-Path cmdlet to see if the registry key exists.
IF(!(Test-Path $registryPath))
{
    New-Item -Path $registryPath -Force | Out-Null
    New-ItemProperty -Path $registryPath -Name $name -Value $value ` -PropertyType DWORD -Force | Out-Null
}
ELSE {
    New-ItemProperty -Path $registryPath -Name $name -Value $value ` -PropertyType DWORD -Force | Out-Null
}
echo 'Registry Path:'$registryPath
echo 'Registry Name:'$registryPath
echo 'Registry Value:'(Get-ItemProperty -Path $registryPath -Name $Name).version
```

The full Systems Manager Command document JavaScript Object Notation (JSON) code example is attached.
## Epics

### Set up a VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC.</td>
<td>On the AWS Management Console, create a VPC that has public and private subnets and a NAT gateway. For more information, see the AWS documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create security groups.</td>
<td>Ensure that each security group allows access for Remote Desktop Protocol (RDP) from the source IP address.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

### Create an IAM policy and an IAM role

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM policy.</td>
<td>Create an IAM policy that provides access to Amazon S3, Amazon EC2, and Systems Manager.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an IAM role.</td>
<td>Create an IAM role, and attach the IAM policy that provides access to Amazon S3, Amazon EC2, and Systems Manager.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

### Run the automation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Systems Manager Command document.</td>
<td>Create a Systems Manager Command document that will deploy the Microsoft Windows registry changes to add or update.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Run the Systems Manager Run Command.</td>
<td>Run the Systems Manager Run Command, selecting the Command document and the Systems Manager target instances. This pushes the Microsoft Windows registry change in the selected Command document to the target instances.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
Related resources

- AWS Systems Manager
- AWS Systems Manager documents
- AWS Systems Manager Run Command

Attachments

attachment.zip

Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows

Created by Ashita Dsilva (AWS)

| Environment | Production | Technologies | Management & governance; Cost management; Databases; Cloud-native | AWS services | AWS Systems Manager; Amazon RDS |

Summary

This pattern demonstrates how to automatically stop and start an Amazon Relational Database Service (Amazon RDS) DB instance on a specific schedule (for example, shutting down a DB instance outside of business hours to reduce costs) by using AWS Systems Manager Maintenance Windows.

AWS Systems Manager Automation provides the AWS-StopRdsInstance and AWS-StartRdsInstance runbooks to stop and start Amazon RDS DB instances. This means that you don’t need to write custom logic with AWS Lambda functions or create an Amazon CloudWatch Events rule.

AWS Systems Manager provides two capabilities for scheduling tasks: State Manager and Maintenance Windows. State Manager sets and maintains the required state configuration for resources in your Amazon Web Services (AWS) account one time or on a specific schedule. Maintenance Windows runs tasks on the resources in your account during a specific time window. Although you can use this pattern’s approach with State Manager or Maintenance Windows, we recommend that you use Maintenance Windows because it can run one or more tasks based on assigned priority and can also run AWS Lambda functions and AWS Step Functions tasks. For more information about State Manager and Maintenance Windows, see Choosing between State Manager and Maintenance Windows in the AWS Systems Manager documentation.

This pattern provides detailed steps to configure two separate maintenance windows that use cron expressions to stop and then start an Amazon RDS DB instance.

Prerequisites and limitations

**Prerequisites**

- An active AWS account.
• An existing Amazon RDS DB instance that you want to stop and start on a specific schedule.
• Cron expressions for your required schedule. For example, the (0 9 * * 1-5) cron expression runs in the morning at 09:00 on Monday through Friday.
• Familiarity with Systems Manager.

Limitations
• An Amazon RDS DB instance can be stopped for up to seven days at one time. After seven days, the DB instance automatically restarts to ensure that it receives any required maintenance updates.
• You can’t stop a DB instance that is a read replica or that has a read replica.
• You can’t stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.
• Service quotas apply to Maintenance Windows and Systems Manager Automation. For more information about service quotas, see AWS Systems Manager endpoints and quotas in the AWS General Reference documentation.

Architecture

The following diagram shows the workflow to automatically stop and start an Amazon RDS DB instance.

The workflow has the following steps:

1. Create a maintenance window and use cron expressions to define the stop and start schedule for your Amazon RDS DB instances.
2. Register a Systems Manager Automation task to the maintenance window by using the AWS-StopRdsInstance or AWS-StartRdsInstance runbook.
3. Register a target with the maintenance window by using a tag-based resource group for your Amazon RDS DB instances.

Technology stack
• AWS CloudFormation
• AWS Identity and Access Management (IAM)
• Amazon RDS
• Systems Manager

Automation and scale

You can stop and start multiple Amazon RDS DB instances at the same time by tagging the required Amazon RDS DB instances, creating a resource group that includes all the tagged DB instances, and registering this resource group as a target for the maintenance window.

Tools

• AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your AWS resources.
• AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources.
• Amazon RDS – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
• AWS Resource Groups – Resource Groups helps you organize AWS resources into groups, tag resources, and manage, monitor, and automate tasks on grouped resources.
• AWS Systems Manager – Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS.
• AWS Systems Manager Automation – Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon Elastic Compute Cloud (Amazon EC2) instances and other AWS resources.
• AWS Systems Manager Maintenance Windows – Maintenance Windows helps you define a schedule for when to perform potentially disruptive actions on your instances.

Epics

Create and configure the IAM service role for Systems Manager Automation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure the IAM service role for Systems Manager Automation. | Sign in to the AWS Management Console and create a service role for Systems Manager Automation. You can use one of the following two methods to create this service role:  
  • Use AWS CloudFormation to configure a service role for Systems Manager Automation  
  • Use IAM to configure roles for Systems Manager Automation | AWS administrator |

The Systems Manager Automation workflow invokes Amazon RDS by using a service role to perform start and stop actions on the Amazon RDS DB instance.
### Task 1: Configuring the Service Role

The service role must be configured with the following **inline policy** that has permissions to start and stop the Amazon RDS DB instance:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RdsStartStop",
            "Effect": "Allow",
            "Action": [
                "rds:StopDBInstance",
                "rds:StartDBInstance"
            ],
            "Resource": "<RDS_Instance_ARN>"
        },
        {
            "Sid": "RdsDescribe",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBInstances",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

Make sure that you replace `<RDS_Instance_ARN>` with your Amazon RDS DB instance's Amazon Resource Name (ARN).

**Important**: Make sure that you record the service role's ARN.

### Create a Resource Group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag the Amazon RDS DB instances.</td>
<td>Open the Amazon RDS console and tag the Amazon RDS DB instances that you want to add to the resource group. A tag is metadata assigned to an AWS resource and consists of a key-value pair. We recommend that you use <strong>Action</strong> as the <strong>Tag key</strong> and <strong>StartStop</strong> as the <strong>Value</strong>.</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a resource group for your tagged Amazon RDS DB instances.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information about this, see Adding, listing, and removing tags in the Amazon RDS documentation.

Open the AWS Resource Groups console and create a resource group based on the tag that you created for your Amazon RDS DB instances.

Under Grouping Criteria, make sure that you choose AWS::RDS::DBInstance for the resource type and then provide the tag's key-value pair (for example, "Action-StartStop"). This ensures that the service only checks for Amazon RDS DB instances and not other resources that have this tag. Make sure that you record the resource group's name.

For more information and detailed steps, see Build a tag-based query and create a group in the AWS Resource Groups documentation.

Skills required: AWS administrator

---

<table>
<thead>
<tr>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure a maintenance window to stop the Amazon RDS DB instances</td>
</tr>
</tbody>
</table>

Create a maintenance window.

1. Open the AWS Systems Manager console, choose Maintenance Windows, and then choose Create a maintenance window. Provide a name for your maintenance window (for example, "StopRdsInstance"), enter a description, and then uncheck Allow unregistered targets.

2. Choose CRON/Rate expression and provide the schedule expression to define when the Amazon RDS DB instances should be stopped. Enter 1 for the Duration and 0 for Stop initiating tasks. By default, the Time zone shows

Skills required: AWS administrator
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTC. You can change the time zone to initiate the maintenance window based on the timestamp defined in your cron expression. 3. Choose <strong>Create maintenance window</strong>. The system returns you to the maintenance window page and the state of your maintenance window is <strong>Enabled</strong>. <strong>Important</strong>: The task to stop the DB instance runs almost instantly when initiated and doesn’t span the entire duration of the maintenance window. This pattern provides the minimum values for <strong>Duration</strong> and <strong>Stop initiating tasks</strong> because they are the required parameters for a maintenance window. For more information and detailed steps, see <strong>Create a maintenance window (console)</strong> in the AWS Systems Manager documentation.</td>
<td></td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Assign a target to the maintenance window.</td>
<td>1. On the <strong>AWS Systems Manager console</strong>, choose <strong>Maintenance Windows</strong>, choose <strong>Actions</strong>, and then choose <strong>Register targets</strong>. 2. In the <strong>Targets</strong> area, specify <strong>Choose a resource group</strong> and then choose the name of an existing resource group in your account. 3. For <strong>Resource types</strong>, choose <strong>AWS::RDS::DBInstance</strong> and then choose <strong>Register target</strong>. For more information and detailed steps, see <strong>Assign targets to a maintenance window (console)</strong> in the AWS Systems Manager documentation.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Assign a task to the maintenance window. | 1. On the *AWS Systems Manager* console, choose **Maintenance Windows** and then choose your maintenance window. Choose **Actions** and then choose **Register Automation task**.  
2. For **Document**, choose **AWS-StopRdsInstance**.  
3. In the **Targets** section, choose **Selecting registered target groups** and then choose the maintenance window target that you registered with the current maintenance window.  
4. For **Rate control**, specify 100 percent for **Concurrency** and **Error threshold**. You can change the **Rate control** values according to your requirements for task concurrency and error threshold. For more information about this, see *About concurrency and error thresholds* in the AWS Systems Manager documentation.  
5. In the IAM service role area, you can choose **Create and use a service-linked role for Systems Manager** or choose **Use a custom service role**.  
6. In the **Input Parameters** section, specify the following parameters for the runbook:  
   - **InstanceId**:  
     ```bash
     {{RESOURCE_ID}}
     ```  
   - **AutomationAssumeRole**:  
     Provide the ARN of the service role that you created for Systems Manager Automation.  
   - **Note**: For **InstanceId**, a pseudo parameter is used to extract the Amazon RDS DB resource ID from the ARN. To learn more about pseudo parameters, see *About pseudo parameters* in the AWS Systems Manager documentation. | **AWS administrator** |
### Configure a maintenance window to start the Amazon RDS DB instances

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure a maintenance window to start the Amazon RDS DB instances.</td>
<td>Repeat the steps from the <em>Configure a maintenance window to stop the Amazon RDS DB instances</em> epic to configure another maintenance window to start the Amazon RDS DB instances at a scheduled time.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> You must make the following changes when you configure the maintenance window to start the DB instances:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Use a new name for the maintenance window (for example, &quot;StartRdsInstance&quot;).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Replace the cron expression with the cron expression that you want to use to start the DB instances.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Replace the <code>AWS-StopRdsInstance</code> runbook with <code>AWS-StartRdsInstance</code> in <em>Task</em>.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>
Related resources

- Use Systems Manager Automation documents to manage instances and cut costs off-hours (AWS blog post)

Copy AWS Service Catalog products across different AWS accounts and AWS Regions

Created by Sachin Vighe (AWS) and Santosh Kale (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Management &amp; governance; Serverless</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS Service Catalog; AWS Lambda</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

AWS Service Catalog is a Regional service and this means that AWS Service Catalog portfolios and products are only visible in the AWS Region where they are created. If you set up an AWS Service Catalog hub in a new Region, you must recreate your existing products and this can be a time-consuming process.

This pattern's approach helps simplify this process by describing how to copy products from an AWS Service Catalog hub in a source AWS account or Region to a new hub in a destination account or Region. For more information about the AWS Service Catalog hub and spoke model, see AWS Service Catalog hub and spoke model: How to automate the deployment and management of AWS Service Catalog to many accounts on the AWS Management and Governance Blog.

The pattern also provides the separate code packages required to copy AWS Service Catalog products across accounts or to other Regions. By using this pattern, your organization can save time, make existing and previous product versions available in a new AWS Service Catalog hub, minimize the risk of manual errors, and scale the approach across multiple accounts or Regions.

Note: This pattern's Epics section provides two options for copying products. You can use Option 1 to copy products across accounts or choose Option 2 to copy products across Regions.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Existing AWS Service Catalog products in a source account or Region.
- An existing AWS Service Catalog hub in a destination account or Region.
- If you want to copy products across accounts, you must share and then import the AWS Service Catalog portfolio containing the products into your destination account. For more information about this, see Sharing and importing portfolios in the AWS Service Catalog documentation.

Limitations
AWS Service Catalog products that you want to copy across Regions or accounts cannot belong to more than one portfolio.

Architecture

The following diagram shows the copying of AWS Service Catalog products from a source account to a destination account.

The following diagram shows the copying of AWS Service Catalog products from a source Region to a destination Region.
Technology stack

- Amazon CloudWatch
- AWS Identity and Access Management (IAM)
- AWS Lambda
- AWS Service Catalog

Automation and scale

You can scale this pattern's approach by using a Lambda function that can be scaled depending on the number of requests received or how many AWS Service Catalog products you need to copy. For more information about this, see Lambda function scaling in the AWS Lambda documentation.

Tools

- AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.
- Amazon CloudWatch – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.
- IAM – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- AWS Service Catalog – AWS Service Catalog helps you create, manage, and distribute portfolios of approved products to end users, who can then access the products they need in a personalized portal. Typical products include servers, databases, websites, or applications that are deployed using AWS resources.

Code

You can use the cross-account-copy package (attached) to copy AWS Service Catalog products across accounts or the cross-region-copy package (attached) to copy products across Regions.

The cross-account-copy package contains the following files:

- copyconf.properties – The configuration file that contains the Region and AWS account ID parameters for copying products across accounts.
- scProductCopyLambda.py – The Python function for copying products across accounts.
- createDestAccountRole.sh – The script to create an IAM role in the destination account.
- createSrcAccountRole.sh – The script to create an IAM role in the source account.
- copyProduct.sh – The script to create and invoke the Lambda function for copying products across accounts.

The cross-region-copy package contains the following files:

- copyconf.properties – The configuration file that contains the Region and AWS account ID parameters for copying products across Regions.
• `scProductCopyLambda.py` – The Python function for copying products across Regions.
• `copyProduct.sh` – The script to create an IAM role and create and invoke the Lambda function for copying products across Regions.

## Epics

### Option 1 – Copy AWS Service Catalog products across accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the configuration file.</td>
<td>1. Download the cross-account-copy package (attached) to your local machine.</td>
<td>AWS administrator, AWS systems administrator, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>2. Update the <code>copyconf.properties</code> configuration file with the following values:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>srcRegion</code> – Provide the source Region that contains the products.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>destRegion</code> – Provide the destination Region for the products.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>sourceAccountId</code> – Provide the AWS account ID for your source account.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <code>destAccountId</code> – Provide the AWS account ID for your destination account.</td>
<td></td>
</tr>
<tr>
<td>Configure your credentials for AWS CLI in the destination account.</td>
<td>Configure your credentials to access AWS CLI in your destination account by running the <code>aws configure</code> command and providing the following values:</td>
<td>AWS administrator, AWS systems administrator, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td><code>$aws configure</code> AWS Access Key ID [None]: &lt;your_access_key_id&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Secret Access Key [None]: &lt;your_secret_access_key&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default region name [None]: Region</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default output format [None]:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about this, see <a href="#">Configuration basics</a> in the AWS Command Line Interface documentation.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Configure your credentials for AWS CLI in the source account.</td>
<td>Configure your credentials to access AWS CLI in your source account by running the <code>aws configure</code> command and providing the following values:</td>
<td>AWS administrator, AWS systems administrator, Cloud administrator</td>
</tr>
<tr>
<td><img src="#" alt="Code" /></td>
<td><img src="#" alt="Description" /></td>
<td><img src="#" alt="Skills" /></td>
</tr>
<tr>
<td>Create a Lambda execution role in your destination account.</td>
<td>Run the <code>createDestAccountRole.sh</code> script in your destination account. The script implements the following actions:</td>
<td>AWS administrator, AWS systems administrator, Cloud administrator</td>
</tr>
<tr>
<td><img src="#" alt="Code" /></td>
<td><img src="#" alt="Description" /></td>
<td><img src="#" alt="Skills" /></td>
</tr>
<tr>
<td>Create the cross-account IAM role in your source account.</td>
<td>Run the <code>createSrcAccountRole.sh</code> script in your source account. The script implements the following actions:</td>
<td>AWS administrator, AWS systems administrator, Cloud administrator</td>
</tr>
<tr>
<td><img src="#" alt="Code" /></td>
<td><img src="#" alt="Description" /></td>
<td><img src="#" alt="Skills" /></td>
</tr>
</tbody>
</table>
### Option 2 – Copy AWS Service Catalog products from a source Region to a destination Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
</tbody>
</table>
| Update the configuration file. | 1. Download the cross-region-copy package (attached) to your local machine.  
2. Update the `copyconf.properties` configuration file with the following values:  
• `srcRegion` – Provide the source Region that contains the products.  
• `destRegion` – Provide the destination Region for the products.  
• `accountId` – Provide your AWS account ID. | AWS systems administrator, Cloud administrator, AWS administrator |
| Configure your credentials for AWS CLI. | Configure your credentials to access AWS CLI in your environment by running the `aws configure` command and providing the following values:  
```
$aws configure
AWS Access Key ID [None]: <your_access_key_id>
AWS Secret Access Key [None]: <your_secret_access_key>
Default region name [None]: Region
Default output format [None]:
```
For more information about this, see Configurations basics in the | AWS administrator, AWS systems administrator, Cloud administrator |
### Related resources

- Create a Lambda execution role ([AWS Lambda documentation](#))
- Create a Lambda function ([AWS Lambda documentation](#))
- AWS Service Catalog API reference
- AWS Service Catalog documentation

### Attachments

attachment.zip

### Create alarms for custom metrics using Amazon CloudWatch anomaly detection

*Created by Ram Kandaswamy (AWS) and Raheem Jiwani (AWS)*

#### Environment:
Production

#### Technologies:
Management & governance; DevOps; Operations; Cloud-native

#### AWS services:
Amazon CloudWatch

### Summary

On the Amazon Web Services (AWS) Cloud, you can use Amazon CloudWatch to create alarms that monitor metrics and send notifications or automatically make changes if a threshold is breached.

To avoid being limited by static thresholds, you can create alarms based on past patterns and that notify you if specific metrics are outside the normal operating window. For example, you could monitor your
API's response times from Amazon API Gateway and receive notifications about anomalies that prevent you from meeting a service-level agreement (SLA).

This pattern describes how to use CloudWatch anomaly detection for custom metrics. The pattern shows you how to create a custom metric in Amazon CloudWatch Logs Insights or publish a custom metric with an AWS Lambda function, and then set up anomaly detection and create notifications using Amazon Simple Notification Service (Amazon SNS).

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- An existing SNS topic, configured to send email notifications. For more information about this, see Getting started with Amazon SNS in the Amazon SNS documentation.
- An existing application, configured with CloudWatch Logs.

**Limitations**

- CloudWatch metrics don't support millisecond time intervals. For more information about the granularity of regular and custom metrics, see the Amazon CloudWatch FAQs.

**Architecture**

The diagram shows the following workflow:

1. Logs that use metrics created and updated by CloudWatch Logs are streamed to CloudWatch.
2. An alarm initiates based on thresholds and sends an alert to an SNS topic.
3. Amazon SNS sends you an email notification.

**Technology stack**

- Cloudwatch
• AWS Lambda
• Amazon SNS

Tools

• **Amazon Cloudwatch** – CloudWatch provides a reliable, scalable, and flexible monitoring solution.
• **AWS Lambda** – Lambda is a compute service that helps you run code without provisioning or managing servers.
• **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.

Epics

Set up anomaly detection for a custom metric

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1 - Create a custom metric with a Lambda function.</td>
<td>Download the lambda_function.py file (attached) and then replace the sample lambda_function.py file in the aws-lambda-developer-guide repository on the AWS Documentation GitHub. This provides you with a sample Lambda function that sends custom metrics to CloudWatch Logs. The Lambda function uses the Boto3 API to integrate with CloudWatch. After you run the Lambda function, you can sign in to the AWS Management Console, open the CloudWatch console, and the published metric is available under your published namespace.</td>
<td>DevOps engineer, AWS DevOps</td>
</tr>
<tr>
<td>Option 2 – Create custom metrics from CloudWatch log groups.</td>
<td>Sign in to the AWS Management Console, open the CloudWatch console, and then choose Log groups. Choose the log group that you want to create a metric for. Choose Actions and then choose Create metric filter. For Filter pattern, enter the filter pattern that you want to use. For more information, see Filter and pattern syntax in the CloudWatch documentation.</td>
<td>DevOps engineer, AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>To test your filter pattern, enter one or more log events under <strong>Test Pattern</strong>. Each log event must be within one line, because line breaks are used to separate log events in the <strong>Log event</strong> messages box. After you test the pattern, you can enter a name and value for your metric under <strong>Metric details</strong>.</td>
<td>DevOps engineer, AWS DevOps</td>
<td></td>
</tr>
<tr>
<td>For more information and steps to create a custom metric, see <a href="https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CreateMetricFilter.html">Create a metric filter for a log group</a> in the CloudWatch documentation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Create an alarm for your custom metric. | On the CloudWatch console, choose **Alarms** and then choose **Create Alarm**. Choose **Select metric** and enter the name of the metric that you created earlier into the search box. Choose the **Graphed metrics** tab and configure the options according to your requirements. Under **Conditions**, choose **Anomaly detection** instead of **Static thresholds**. This shows you a band based on two standard default deviations. You can set up thresholds and adjust them according to your requirements. Choose **Next**.  
  **Note**: The band is dynamic and depends on the quality of the datapoints. When you begin aggregating more data, the band and thresholds are automatically updated. |                                     |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up SNS notifications</td>
<td>Under <strong>Notification</strong>, choose the SNS topic to notify when the alarm is in ALARM state, OK state, or INSUFFICIENT_DATA state. To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose <strong>Add notification</strong>. Choose Next. Enter a name and description for the alarm. The name must only contain ASCII characters. Then choose Next. Under <strong>Preview and create</strong>, confirm that the information and conditions are correct, and then choose <strong>Create alarm</strong>.</td>
<td>DevOps engineer, AWS DevOps</td>
</tr>
</tbody>
</table>

**Related resources**

- Publishing custom metrics to CloudWatch
- Using CloudWatch anomaly detection
- Alarm events and Amazon EventBridge
- What are the best practices to follow while pushing custom metrics to Cloud Watch? (video)
- Introduction to CloudWatch Application Insights (video)
- Detect anomalies with CloudWatch (video)

**Attachments**

attachment.zip

**Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization**

*Created by Ram Kandaswamy (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Management &amp; governance; Cloud-native; Infrastructure; Operations; Modernization</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services: Amazon SNS; AWS Config; AWS Lambda; AWS CloudFormation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

Customers on Amazon Web Services (AWS) are often looking for an efficient way to detect resource configuration mismatches, including drift in AWS CloudFormation stacks, and fix them as soon as possible. This is especially the case when AWS Control Tower or AWS Landing Zone solutions are used.

This pattern provides a prescriptive solution that efficiently solves the problem by using consolidated resource configuration changes and acting on those changes to generate results. The solution is designed for scenarios where there are several CloudFormation stacks created in more than one Region or more than one account or a combination of both. The goals of the solution are the following:

- Simplify the drift detection process
- Set up notification and alerting
- Set up consolidated reporting

Prerequisites and limitations

Prerequisites

- AWS Config enabled in all the Regions and accounts that must be monitored

Limitations

- The report generated supports only the .csv or .json output formats.

Architecture

Target technology stack

The current guidance will help organizations achieve the goal by using a combination of the following services:

- AWS Config rule
- Amazon CloudWatch rule
- AWS Identity and Access Management (IAM)
- AWS Lambda
- Amazon Simple Notification Service (Amazon SNS)
1. The AWS Config rule detects drift.
2. Drift detection results in other accounts are sent to the management account.
3. The CloudWatch rule calls Lambda.
4. Lambda queries the AWS Config rule for aggregated results.
5. Lambda notifies Amazon SNS, which sends email notification of the drift.

**Automation and scale**

The solution presented here can scale for both additional Regions and accounts.

**Tools**

**AWS Config** – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time. With AWS Config, you can assess, audit, and evaluate the configurations of your AWS resources.

**Amazon CloudWatch** – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

**AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

**Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).
## Epics

**Automate drift detection for CloudFormation**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the aggregator.</td>
<td>On the AWS Config console, create an aggregator in the management account. Ensure that data replication is turned on so that AWS Config can fetch data from the source accounts. Also, select all applicable Regions and accounts. You can select accounts based on organizations. This is the recommended approach because new accounts in the organization are automatically part of the aggregator.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Create an AWS managed rule.</td>
<td>Add the <code>cloudformation-stack-drift-detection-check</code> AWS managed rule. The rule needs one parameter value: <code>cloudformationArn</code>. Enter the IAM role Amazon Resource Name (ARN) that has permissions to detect stack drift. In addition, the role must have a trust policy that enables AWS Config to assume the role.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
| Create the advanced query section of the aggregator. | To fetch drifted stacks from multiple sources, create the following query:  

```
SELECT resourceId, configuration.driftInformation.stackDriftStatus  
WHERE resourceType = 'AWS::CloudFormation::Stack'  
AND  
configuration.driftInformation.stackDriftStatus IN ('DRIFTED')
```  

<table>
<thead>
<tr>
<th>Automate running the query and publish.</th>
<th>Create a Lambda function using the code that is attached. Lambda will publish the results to an Amazon SNS topic that is provided as an environment variable in the Lambda function. Also, to receive alerts, create an email subscription to an existing Amazon SNS topic.</th>
<th>Cloud architect, Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudWatch rule.</td>
<td>Create a schedule-based CloudWatch rule to call the</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Related resources

Resources

- What Is AWS Config?
- Concepts: multi-account multi-Region data aggregation
- Multi-account multi-Region data aggregation
- Detecting unmanaged configuration changes to stacks and resources
- IAM: Pass an IAM role to a specific AWS service
- What is Amazon SNS?

Additional information

Considerations

Using custom solutions that involve API calls at specific intervals to initiate drift detection on each CloudFormation stack or on stack sets is not optimal. It leads to a large number API calls and affects the performance. Because of the number of API calls, throttling can happen. Another potential issue is a delay in detection if resource changes are identified based on schedule only.

FAQ

Q. Should I use an add-on based solution with AWS Landing Zone?

A. With the availability of the advanced queries feature in AWS Config, along with the aggregator, the recommendation is to use AWS Config instead of an add-on.

Q. How does this solution address CloudFormation StackSets?

A. Because stack sets are made of stacks, you can use this solution. Stack instance details are also available as part of the solution.

Attachments

attachment.zip

Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK

Created by Rahul Sharad Gaikwad (AWS)
Summary

This pattern demonstrates the steps to enable the Amazon DevOps Guru service across multiple Amazon Web Services (AWS) Regions, accounts, and organizational units (OUs) by using the AWS Cloud Development Kit (AWS CDK) in TypeScript. You can use AWS CDK stacks to deploy AWS CloudFormation StackSets from the administrator (primary) AWS account to enable Amazon DevOps Guru across multiple accounts, instead of logging into each account and enabling DevOps Guru individually for each account.

Amazon DevOps Guru provides artificial intelligence operations (AIOps) features to help you improve the availability of your applications and resolve operational issues faster. DevOps Guru reduces your manual effort by applying machine learning (ML) powered recommendations, without requiring any ML expertise. DevOps Guru analyzes your resources and operational data. If it detects any anomalies, it provides metrics, events, and recommendations to help you address the issue.

This pattern describes three deployment options for enabling Amazon DevOps Guru:

- For all stack resources across multiple accounts and Regions
- For all stack resources across OUs
- For specific stack resources across multiple accounts and Regions

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. (See Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.)
- AWS CDK Toolkit, installed and configured. (See AWS CDK Toolkit in the AWS CDK documentation.)
- Node Package Manager (npm), installed and configured for the AWS CDK in TypeScript. (See Downloading and installing Node.js and npm in the npm documentation.)
- Python3 installed and configured, for running a Python script to inject traffic into the sample serverless application. (See Python Setup and Usage in the Python documentation.)
- Pip, installed and configured to install the Python requests library. (See the pip installation instructions on the PyPI website.)

Product versions

- AWS CDK Toolkit version 1.107.0 or later
- npm version 7.9.0 or later
- Node.js version 15.3.0 or later
Architecture

Technologies

The architecture for this pattern includes the following services:

- Amazon DevOps Guru
- AWS CloudFormation
- Amazon API Gateway
- AWS Lambda
- Amazon DynamoDB
- Amazon CloudWatch
- AWS CloudTrail

AWS CDK stacks

The pattern uses the following AWS CDK stacks:

- CdkStackSetAdminRole – Creates an AWS Identity and Access management (IAM) administrator role to establish a trust relationship between the administrator and target accounts.
- CdkStackSetExecRole – Creates an IAM role to trust the administrator account.
- CdkDevOpsGuruStackMultiAccReg – Enables DevOps Guru across multiple AWS Regions and accounts for all stacks, and sets up Amazon Simple Notification Service (Amazon SNS) notifications.
- CdkDevOpsGuruStackMultiAccRegSpecStacks – Enables DevOps Guru across multiple AWS Regions and accounts for specific stacks, and sets up Amazon SNS notifications.
- CdkDevopsGuruStackOrgUnit – Enables DevOps Guru across OUs, and sets up Amazon SNS notifications.
- CdkInfrastructureStack – Deploys sample serverless application components such as API Gateway, Lambda, and DynamoDB in the administrator account to demonstrate fault injection and insights generation.

Sample application architecture

The following diagram illustrates the architecture of a sample serverless application that has been deployed across multiple accounts and Regions. The pattern uses the administrator account to deploy all the AWS CDK stacks. It also uses the administrator account as one of the target accounts for setting up DevOps Guru.

1. When DevOps Guru is enabled, it first baselines each resource's behavior and then ingests operational data from CloudWatch vended metrics.
2. If it detects an anomaly, it correlates it with the events from CloudTrail, and generates an insight.
3. The insight provides a correlated sequence of events along with prescribed recommendations to enable the operator to identify the culprit resource.
4. Amazon SNS sends notification messages to the operator.
Automation and scale

The GitHub repository provided with this pattern uses the AWS CDK as an infrastructure as code (IaC) tool to create the configuration for this architecture. AWS CDK helps you orchestrate resources and enable DevOps Guru across multiple AWS accounts, Regions, and OUs.

Tools

AWS services

- **AWS CDK** – AWS Cloud Development Kit (AWS CDK) helps you define your cloud infrastructure as code in one of five supported programming languages: TypeScript, JavaScript, Python, Java, and C#.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent command-line interface for interacting with AWS services and resources.

Code

The source code for this pattern is available on GitHub, in the Amazon DevOps Guru CDK Samples repository. The AWS CDK code is written in TypeScript. To clone and use the repository, follow the instructions in the next section.

**Important:** Some of the stories in this pattern include AWS CDK and AWS CLI command examples that are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) continuation character at the end of each line with a caret (^).
# Epics

Prepare the AWS resources for deployment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure AWS named profiles.</td>
<td>Set up your AWS named profiles as follows to deploy stacks in a multi-account environment.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>For the administrator account:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$aws configure --profile administrator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Access Key ID [****]: &lt;your-administrator-access-key-ID&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Secret Access Key [****]: &lt;your-administrator-secret-access-key&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default region name [None]: &lt;your-administrator-region&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default output format [None]: json</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For the target account:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$aws configure --profile target</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Access Key ID [****]: &lt;your-target-access-key-ID&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Secret Access Key [****]: &lt;your-target-secret-access-key&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default region name [None]: &lt;your-target-region&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default output format [None]: json</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see Named profiles in the AWS CLI documentation.</td>
<td></td>
</tr>
<tr>
<td>Verify AWS profile</td>
<td>(Optional) You can verify your AWS profile configurations in the credentials and config files by following the instructions in Set and view configuration settings in the AWS CLI documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>configurations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify the AWS CDK version.</td>
<td>Verify the version of the AWS CDK Toolkit by running the following command:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>$cdk --version</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| Clone the project code. | Clone the GitHub repository for this pattern by using the command:  
```bash
$git clone https://github.com/aws-samples/amazon-devopsguru-cdk-samples.git
```
| DevOps engineer |
| Install package dependencies and compile the TypeScript files. | Install the package dependencies and compile the TypeScript files by running the following commands:  
```bash
$cd amazon-devopsguru-cdk-samples  
$npm install  
$npm fund
```
| DevOps engineer |

These commands install all the packages from the sample repository.  
**Important:** If you get any errors about missing packages, use one of the following commands:  
```bash
$npm ci
```

— or —
```bash
$npm install -g @aws-cdk/<package-name>
```

You can find the list of package names and versions in the Dependencies section of the  
`/amazon-devopsguru-cdk-samples/package.json` file. For more information, see `npm ci` and `npm install` in the npm documentation.
# Build (synthesize) the AWS CDK stacks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure an email address for Amazon SNS notifications. | Follow these steps to provide an email address for Amazon SNS notifications:  
2. In the `DevOpsGuruTopicSubscription` section, update the `Endpoint` parameter with your email address.  
3. Save and close the files. | DevOps engineer |
| Build the project code. | Build the project code and synthesize the stacks by running the command: 
```bash
npm run build && cdk synth
```
You should see output similar to the following: 
```bash
$npm run build && cdk synth  
> cdk-devopsguru@0.1.0 build  
> tsc  
Successfully synthesized to ~/amazon-devopsguru-cdk-samples/cdk.out  
Supply a stack id (CdkDevopsGuruStackMultiAccReg, CdkDevopsGuruStackMultiAccRegSpecStacks, CdkDevopsGuruStackOrgUnit, CdkInfrastructureStack, CdkStackSetAdminRole, CdkStackSetExecRole) to display its template.  
```
For more information and steps, see [Your first AWS CDK app](https://docs.aws.amazon.com/cdk/latest/guide/getting-started.html) in the AWS CDK documentation. | DevOps engineer |
| List the AWS CDK stacks. | Run the following command to list all AWS CDK stacks: 
```bash
$cdk list
``` | DevOps engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The command displays the following list:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CdkDevopsGuruStackMultiAccReg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CdkDevopsGuruStackMultiAccRegSpecStacks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CdkDevopsGuruStackOrgUnit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CdkInfrastructureStack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CdkStackSetAdminRole</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CdkStackSetExecRole</td>
<td></td>
</tr>
</tbody>
</table>

**Option 1 - Enable DevOps Guru for all stack resources across multiple accounts**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CDK stacks for creating IAM roles.</td>
<td>This pattern uses <a href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stack-sets.html">AWS CloudFormation StackSets</a> to perform stack operations across multiple accounts. If you are creating your first stack set, you must create the following IAM roles to get the required permissions set up in your AWS accounts:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>• AWSCloudFormationStackSetAdministrationRole</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AWSCloudFormationStackSetExecutionRole</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The roles must have these exact names.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Create the IAM AWSCloudFormationStackSetAdministrationRole role in the administrator (primary) account by running the following CLI command:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```sh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$cdk deploy CdkStackSetAdminRole --profile administrator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Create the IAM AWSCloudFormationStackSetExecutionRole role in all target accounts where you want to run the stack instances. To create this role, run these CLI commands:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```sh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$cdk deploy CdkStackSetExecRole \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--parameters AdministratorAccountId=&lt;administrator-account-ID&gt; \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--profile administrator</td>
<td></td>
</tr>
</tbody>
</table>
### Deploy the AWS CDK stack for enabling DevOps Guru across multiple accounts.

The AWS CDK CdkDevOpsGuruStackMultiAccReg stack creates stack sets to deploy stack instances across multiple accounts and Regions. To deploy the stack, run the following CLI command with the specified parameters:

```bash
$cdk deploy
CdkDevopsGuruStackMultiAccReg
  --profile administrator
  --parameters
  AdministratorAccountId=<administrator-account-ID>
  --parameters
  TargetAccountId=<target-account-ID>
  --parameters
  RegionIds="<region-1>,<region-2>"
```

Currently Amazon DevOps Guru is available in the AWS Regions listed in the [DevOps Guru FAQ](https://docs.aws.amazon.com/autoscaling/latest/userguide/what-is-devops-guru.html).

### Option 2 - Enable DevOps Guru for all stack resources across OUs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract OU IDs.</td>
<td>On the <a href="https://console.aws.amazon.com/org">AWS Organizations</a> console, identify the IDs of the organizational units where you want to enable DevOps Guru.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Enable service-managed permissions for OUs.</td>
<td>If you're using AWS Organizations for account management, you must grant service-managed permissions to enable DevOps Guru. Instead of creating the IAM roles manually,</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>use organization-based trusted access and service-linked roles (SLRs).</td>
<td>Deploy the AWS CDK stack for enabling DevOps Guru across OUs.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

The AWS CDK CdkDevopsGuruStackOrgUnit stack enables DevOps Guru service across OUs. To deploy the stack, run the following command with the specified parameters:

```bash
$cdk deploy CdkDevopsGuruStackOrgUnit --profile administrator --parameters RegionIds="<region-1>,<region-2>" --parameters OrganizationalUnitIds="<OU-1>,<OU-2>"
```

### Option 3 - Enable DevOps Guru for specific stack resources across multiple accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the AWS CDK stacks for creating IAM roles. | If you haven't already created the required IAM roles shown in the first option, do that first:  
1. Create the IAM AWSCloudFormationStackSetAdministrationRole role in the administrator (primary) account by running the following CLI command:  
   ```bash
   $cdk deploy CdkStackSetAdminRole --profile administrator
   ```  
2. Create the IAM AWSStackSetSetExecutionRole role in all target accounts where you want to run the stack instances. To create this role, run the CLI commands:  
   ```bash
   $cdk deploy CdkStackSetExecRole --parameters AdministratorAccountId=<administrator-account-ID> --profile administrator
   ``` | DevOps engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
|                             | $cdk deploy CdkStackSetExecRole  
|                               |   --parameters AdministratorAccount=Id=<administrator-account-ID>  
|                               |   --profile target                                                                                                                  | DevOps engineer         |
| Delete existing stacks.      | If you already used the first option to enable DevOps Guru for all stack resources, you can delete the old stack by using the following command:                                                                                          | DevOps engineer         |
|                             | $cdk destroy CdkDevopsGuruStackMultiAccReg  
<p>|                               |   --profile administrator                                                                                                              |                          |
|                             | Or, you can change the RegionIds parameter when you redeploy the stack to avoid a Stacks already exist error.                                                                                                  |                          |
| Update the AWS CDK stack with | 1. Edit the file /amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-multi-acc-reg-spec-stack.ts.                                                                                                           | Data engineer           |
| a stack list.                | 2. Under Resources, CloudFormation, StackNames, list the stacks for which you want to enable DevOps Guru. For demonstration purposes, the parameter specifies the CdkInfrastructureStack stack, but you can edit this entry based on your requirements. |                          |
|                             | 3. Save and close the file.                                                                                                                                                                             |                          |
|                             | 4. To synthesize and update the stack template, run:                                                                                                                                            |                          |
|                             | $cdk synth                                                                                                                                   |                          |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the AWS CDK stack for enabling DevOps Guru for specific stack resources across multiple accounts. | The AWS CDK CdkDevopsGuruStackMultiAccRegSpecStacks stack enables DevOps Guru for specific stack resources across multiple accounts. To deploy the stack, run the following command:  

```
cdk deploy  
CdkDevopsGuruStackMultiAccRegSpecStacks  
--profile administrator  
--parameters  
AdministratorAccountId=<administrator-account-ID>  
--parameters  
TargetAccountId=<target-account-ID>  
--parameters  
RegionIds="<region-1>,<region-2>"
```

**Note:** If you previously deployed this stack for option 1, change the RegionIds parameter (making sure to choose from available Regions) to avoid a *Stacks already exist* error. | DevOps engineer |

### Deploy the AWS CDK infrastructure stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the sample serverless infrastructure stack. | The AWS CDK CdkInfrastructureStack stack deploys serverless components such as API Gateway, Lambda, and a DynamoDB table to demonstrate DevOps Guru insights. To deploy the stack, run the following command:  

```
cdk deploy  
CdkInfrastructureStack --profile administrator
```

| Insert sample records in DynamoDB. | Run the following command to populate the DynamoDB table with sample records. Provide the correct path for the populate-shops-dynamodb-table.json script. | DevOps engineer |
## AWS Prescriptive Guidance Patterns

### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$aws dynamodb batch-write-item \ --request-items file://scripts/populate-shops-dynamodb-table.json \ --profile administrator</td>
<td></td>
</tr>
</tbody>
</table>
| Verify inserted records in DynamoDB. | The command displays the following output:  

```json
{
    "UnprocessedItems": {}
}
``` | DevOps engineer |
| | To verify that the DynamoDB table includes the sample records from the populate-shops-dynamodb-table.json file, access the URL for the ListRestApiEndpointMonitorOperator API, which is published as an output of the AWS CDK stack. You can also find this URL in the Outputs tab of the AWS CloudFormation console for the CdkInfrastructureStack stack. The AWS CDK output would look similar to the following:  

```text
CdkInfrastructureStack.CreateRestApiMonitorOperatorEndpointD1D00045 = https://oure17c5vob.execute-api.<your-region>.amazonaws.com/prod/
``` | |
| Wait for resources to complete baselining. | This serverless stack has a few resources. We recommend that you wait for 2 hours before you carry out the next steps. If you deployed this stack in a production environment, it might take up to 24 hours to complete baselining, depending on the number of resources you selected to monitor in DevOps Guru. | DevOps engineer |
### Generate DevOps Guru insights

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Update the AWS CDK infrastructure stack. | To try out DevOps Guru insights, you can make some configuration changes to reproduce a typical operational issue.  
1. Edit the file `/amazon-devopsguru-cdk-samples/lib/infrastructure-stack.ts`.  
2. In the DDB Table section, change the read capacity for the DynamoDB table from 5 to 1.  
3. Save and close the file.  
4. Run the following commands to synthesize and deploy the updated AWS CDK infrastructure stack:  
   ```bash  
   $ cdk synth  
   $ cdk deploy  
   CdkInfrastructureStack --profile administrator  
   ``` | DevOps engineer |
| Inject HTTP requests on the API. | Inject ingress traffic in the form of HTTP requests on the `ListRestApiMonitorOperatorEndpointxxxx` API:  
1. Edit the Python script `/amazon-devopsguru-cdk-samples/scripts/sendAPIRequest.py`.  
2. Update the `url` variable with the API link for `ListRestApiMonitorOperatorEndpointxxxx`. You can find this URL in the output of the AWS CDK `deploy` command or on the AWS Cloudformation console, in the `Outputs` tab for the stack.  
3. Save and close the file.  
4. Run the Python script by using the command:  
   ```bash  
   $ python sendAPIRequest.py  
   ```  
5. Make sure that you get a 200 status code. | DevOps engineer |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>You might need to run the script through multiple (preferably four) terminals to inject traffic at a high rate.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>After the script runs approximately 10 minutes in a loop, you can see an operational insight on the DevOps Guru console.</td>
<td></td>
</tr>
</tbody>
</table>

#### Review DevOps Guru insights.
Under standard conditions, the DevOps Guru dashboard displays zero in the ongoing insights counter. If it detects an anomaly, it raises an alert in the form of an insight. In the navigation pane, choose **Insights** to see the details of the anomaly, including an overview, aggregated metrics, relevant events, and recommendations. For more information about reviewing insights, see the *Gaining operational insights with AIOps using Amazon DevOps Guru* blog post.

**DevOps engineer**

#### Clean up

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean up and delete resources.</td>
<td>After you walk through this pattern, you should remove the resources you created to avoid incurring any further charges. Run these commands:</td>
<td></td>
</tr>
</tbody>
</table>

```bash
$cdk destroy CdkDevopsGuruStackMultiAccReg --profile administrator
$cdk destroy CdkDevopsguruStackOrgUnit --profile administrator
$cdk destroy CdkDevopsGuruStackMultiAccRegSpecStacks --profile administrator
$cdk destroy CdkInfrastructureStack --profile administrator
$cdk destroy CdkStackSetAdminRole --profile administrator
$cdk destroy CdkStackSetExecRole --profile administrator
```

<table>
<thead>
<tr>
<th></th>
<th>DevOps engineer</th>
</tr>
</thead>
</table>

**DevOps engineer**
Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions

**Created by Ram Kandaswamy (AWS)**

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Management &amp; governance; Cloud-native; Infrastructure; Modernization</td>
</tr>
<tr>
<td>Workload:</td>
<td>All other workloads</td>
</tr>
</tbody>
</table>

| AWS services: | AWS Service Catalog; AWS CloudFormation |

**Summary**

Amazon Web Services (AWS) Service Catalog simplifies and accelerates the governance and distribution of infrastructure as code (IaC) templates for enterprises. You use AWS CloudFormation templates to define a collection of AWS resources (stacks) required for a product. AWS CloudFormation StackSets extends this functionality by enabling you to create, update, or delete stacks across multiple accounts and AWS Regions with a single operation.

AWS Service Catalog administrators create products by using CloudFormation templates that are authored by developers, and publish them. These products are then associated with a portfolio, and constraints are applied for governance. To make your products available to users in other AWS accounts or organizational units (OUs), you typically share your portfolio with them. This pattern describes an alternative approach for managing AWS Service Catalog product offerings that is based on AWS CloudFormation StackSets. Instead of sharing portfolios, you use stack set constraints to set AWS Regions and accounts where your product can be deployed and used. By using this approach, you can provision your AWS Service Catalog products in multiple accounts, OUs, and AWS Regions, and manage them from a central location, while meeting your governance requirements.

Benefits of this approach:

- The product is provisioned and managed from the primary account, and not shared with other accounts.
- This approach provides a consolidated view of all provisioned products (stacks) that are based on a specific product.
- Configuration with AWS Service Management Connector is easier, because it targets only one account.
• It's easier to query and use products from AWS Service Catalog.

Prerequisites and limitations

Prerequisites

• AWS CloudFormation templates for IaC and versioning
• Multi-account setup and AWS Service Catalog for provisioning and managing AWS resources

Limitations

• This approach uses AWS CloudFormation StackSets, and the limitations of StackSets apply:
  • StackSets doesn't support CloudFormation template deployment through macros. If you're using a macro to preprocess the template, you won't be able to use a StackSets-based deployment.
  • StackSets provides the ability to disassociate a stack from the stack set, so you can target a specific stack to fix an issue. However, a disassociated stack cannot be re-associated with the stack set.
  • AWS Service Catalog autogenerates StackSet names. Customization isn't currently supported.

Architecture

Target architecture

1. The user creates an AWS CloudFormation template to provision AWS resources, in JSON or YAML format.
2. The CloudFormation template creates a product in AWS Service Catalog, which is added to a portfolio.
3. The user creates a provisioned product, which creates CloudFormation stacks in the target accounts.
4. Each stack provisions the resources specified in the CloudFormation templates.

Tools

AWS services

• AWS Service Catalog – AWS Service Catalog helps you create, manage, and distribute portfolios of approved products to end users, who can then access the products they need in a personalized portal. Typical products include servers, databases, websites, or applications that are deployed using AWS resources.
• AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a
template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

## Epics

### Provision products across accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a portfolio.</td>
<td>A portfolio is a container that includes one or more products that are grouped together based on specific criteria. Using a portfolio for your products helps you apply common constraints across your product set. To create a portfolio, follow the instructions in the AWS Service Catalog documentation. If you're using the AWS CLI, here's an example command: aws servicecatalog create-portfolio --provider-name my-provider --display-name my-portfolio For more information, see the AWS CLI documentation.</td>
<td>AWS Service Catalog, IAM</td>
</tr>
<tr>
<td>Create a CloudFormation template.</td>
<td>Create a CloudFormation template that describes the resources. Resource property values should be parameterized where applicable.</td>
<td>AWS CloudFormation, JSON/YAML</td>
</tr>
<tr>
<td>Create a product with version information.</td>
<td>The CloudFormation template becomes a product when you publish it in the AWS Service Catalog. Provide values for the optional version detail parameters, such as version title and description; this will be helpful for querying for the product later.</td>
<td>AWS Service Catalog</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| To create a product, follow the instructions in the [AWS Service Catalog documentation](https://aws.amazon.com/documentation/servicecatalog/). If you're using the AWS CLI, an example command is:  
```bash
aws servicecatalog create-product --cli-input-json file://create-product-input.json
```
<table>
<thead>
<tr>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>where <code>create-product-input.json</code> is the file that passes the parameters for the product. For an example of this file, see the <em>Additional information</em> section. For more information, see the <a href="https://docs.aws.amazon.com/cli/latest/index.html">AWS CLI documentation</a>.</td>
</tr>
</tbody>
</table>

#### Apply constraints.  
Apply stack set constraints to the portfolio, to configure product deployment options such as multiple AWS accounts, Regions, and permissions. For instructions, see the [AWS Service Catalog documentation](https://aws.amazon.com/documentation/servicecatalog/).  

#### Add permissions.  
Provide permissions to users so that they can launch the products in the portfolio. For console instructions, see the [AWS Service Catalog documentation](https://aws.amazon.com/documentation/servicecatalog/). If you're using the AWS CLI, here's an example command:  
```bash
aws servicecatalog associate-principal-with-portfolio 
   --portfolio-id port-2s6abcdefwdh4 
   --principal-arn arn:aws:iam::444455556666:role/Admin 
   --principal-type IAM
```
<table>
<thead>
<tr>
<th>AWS Service Catalog, IAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the <a href="https://docs.aws.amazon.com/cli/latest/index.html">AWS CLI documentation</a>.</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provision the product.</td>
<td>A provisioned product is a resourced instance of a product. Provisioning a product based on a CloudFormation template launches a CloudFormation stack and its underlying resources. Provision the product by targeting the applicable AWS Regions and accounts, based on stack set constraints. In the AWS CLI, here’s an example command: aws servicecatalog provision-product \   --product-id prod-abcdf23syn2rg \   --provisioning-artifact-id pa-abc347pcsccfm \   --provisioned-product-name &quot;mytestppname3&quot;</td>
<td>AWS Service Catalog</td>
</tr>
</tbody>
</table>

For more information, see the AWS CLI documentation.

### Related resources

#### References

- Overview of AWS Service Catalog
- Using AWS CloudFormation StackSets

#### Tutorials and videos


### Additional information

When you use the `create-product` command, the `cli-input-json` parameter points to a file that specifies information such as product owner, support email, and CloudFormation template details. Here’s an example of such a file:

```json
{
    "Owner": "Test admin",
    "SupportDescription": "Testing",
    "Name": "SNS",
    "SupportEmail": "example@example.com",
    "ProductType": "CLOUDFORMATION_TEMPLATE",
    "AcceptLanguage": "en",
    "ProvisioningArtifactParameters": {
        "Description": "SNS product",
```
Migrate an AWS member account from AWS Organizations to AWS Control Tower

Created by Rodolfo Jr. Cerrada (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Management &amp; governance; Modernization</td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS Organizations; AWS Control Tower</td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an Amazon Web Services (AWS) account from AWS Organizations, where it is a member account that's governed by a management account, to AWS Control Tower. By enrolling the account in AWS Control Tower, you can take advantage of preventive and detective guardrails and features that streamline your account governance. You might also want to migrate your member account if your AWS Organizations management account has been compromised, and you want to move member accounts to a new organization that is governed by AWS Control Tower.

AWS Control Tower provides a framework that combines and integrates the capabilities of several other AWS services, including AWS Organizations, and ensures consistent compliance and governance across your multi-account environment. With AWS Control Tower, you can follow a set of prescribed rules and definitions that extend the capabilities of AWS Organizations. For example, you can use guardrails to ensure that security logs and necessary cross-account access permissions are created, and not altered.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Control Tower set up in your target organization in AWS Organizations (for instructions, see Setting up in the AWS Control Tower documentation)
- Administrator credentials for AWS Control Tower (member of the AWSControlTowerAdmins group)
- Administrator credentials for the source AWS account

Limitations

- The source management account in AWS Organizations must be different from the target management account in AWS Control Tower.
Product versions

- AWS Control Tower version 2.3 (February 2020) or later (see release notes)

Architecture

The following diagram illustrates the migration process and reference architecture. This pattern migrates the AWS account from the source organization to a target organization that is governed by AWS Control Tower.

The enrollment process consists of these steps:

1. The account leaves the source organization in AWS Organizations.
2. The account becomes a standalone account. This means that it doesn't belong to any organization, so governance and billing are managed independently by account administrators.
3. The target organization sends an invitation for the account to join the organization.
4. The standalone account accepts the invitation and becomes a member of the target organization.
5. The account is enrolled in AWS Control Tower and moved to a registered organizational unit (OU). (We recommend that you check the AWS Control Tower dashboard to confirm the enrollment.) At this point, all guardrails that are enabled in the registered OU take effect.

Tools

AWS services

- AWS Organizations – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into a single entity (an organization) that you create and centrally manage.
- AWS Control Tower – AWS Control Tower integrates the capabilities of other services, including AWS Organizations, AWS Single Sign-on (AWS SSO), and AWS Service Catalog, to help you enforce and manage governance rules for security, operations, and compliance at scale across all your organizations and accounts in the AWS Cloud.
## Epics

Remove the member account from the source organization

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify that the member account can run as a stand-alone account.</td>
<td>Confirm that the member account that will leave the source organization has the information that is required for it to operate as a standalone account. For example, if the member account doesn't have billing information, it can't operate as a standalone account, because AWS uses the payment information to charge for any billable AWS activity that occurs while the account isn't attached to an organization. Typically, if you created the member account by using the AWS Organizations console, API, or AWS Command Line Interface (CLI) commands, the information required of standalone accounts isn't automatically collected. To add this information, sign in to the account, and specify a support plan, contact information, and a payment method. For more information about what you need to know before removing an account from an organization, see Before removing an account from organization in the AWS Organizations documentation.</td>
<td>Account administrator</td>
</tr>
<tr>
<td>Remove the member account from its source organization.</td>
<td>Follow the instructions in the AWS Organizations documentation to remove a member account from an organization. You can sign in to the organization's management account and remove the member account, or sign in to the member account and leave the organization. If you don't have administrator-level credentials to remove or leave the account, ask</td>
<td>Management account administrator or account administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>for assistance from your organization's administrator. If the member account is missing a support plan, contact information, or payment information, you will be prompted to provide and verify that information. When you leave the organization, you are redirected to the <strong>Getting Started</strong> page of the AWS Organizations console, where you can view invitations for your account to join other organizations. <strong>Important</strong>: At this point, your account is a standalone account. If you are running workloads that aren't covered by AWS Free Tier, you will be charged according to the payment and billing information you provided for the account.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In the AWS Organizations console, you should no longer see the <strong>Leave organization</strong> button. Instead, you should see pending invitations, if any, from other organizations.</td>
<td>Account administrator</td>
</tr>
<tr>
<td></td>
<td>When you remove the account from the source organization, AWS Identity and Access Management (IAM) roles created by AWS Organizations or by administrators aren't automatically deleted. To terminate access from the source organization's management account, you must manually delete the IAM roles. For more information, see <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_manage.html">Deleting roles or instance profiles</a> in the IAM documentation. When a member account leaves an organization, all tags that were attached to the account are deleted. Standalone accounts do not support tags.</td>
<td>Account administrator</td>
</tr>
</tbody>
</table>
### Invite the account to join the new organization with AWS Control Tower

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to AWS Control Tower.</td>
<td>Sign in to the AWS Control Tower console as an administrator.</td>
<td>AWS Control Tower administrator</td>
</tr>
<tr>
<td></td>
<td>Currently, there is no direct way to move an AWS account from a source organization to an organization in an OU that's governed by AWS Control Tower. However, you can extend AWS Control Tower governance to an existing AWS account when you enroll it into an OU that's already governed by AWS Control Tower. That's why you have to log in to AWS Control Tower for this step.</td>
<td></td>
</tr>
</tbody>
</table>
| Invite the member account.        | 1. Sign in to the AWS Organizations console, and navigate to the AWS Accounts page.  
2. On the Add an AWS account page, choose Invite an existing AWS account.  
3. Complete the account information, including the 12-digit account number (without dashes) and the optional description and tags, and then choose Send invitation.  

**Important:** Verify that no applications or network connectivity will be affected by the account transfer.  
This action sends an invitation email with a link to the member account. When the account administrator follows the link and accepts the invitation, the member account appears in the AWS accounts page. For more information, see Inviting an AWS account to join your organization in the AWS Organizations documentation. | AWS Control Tower administrator                      |
| Test applications and connectivity.| When the member account has been registered into the                                                                                                                                                                                                                                                                                                                                                   | AWS Control Tower administrator, Member account       |
### Prepare the account for enrollment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review guardrails and fix any violations.</td>
<td>Review the guardrails that are defined in the target OU, especially the preventive guardrails, and fix any violations.</td>
<td>AWS Control Tower administrator, Member account administrator</td>
</tr>
</tbody>
</table>

A number of **mandatory, preventive guardrails** are enabled by default when you set up your AWS Control Tower landing zone. These can't be disabled. You must review these mandatory guardrails and fix the member account (manually or by using a script) before you enroll the account.

**Note:** Preventive guardrails keep AWS Control Tower registered accounts compliant and prevent policy violations. Any violation of preventive guardrails might affect...
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrollment. Detective guardrail violations appear in the AWS Control Tower dashboard, if detected, after successful enrollment. They do not affect the enrollment process. For more information, see Guardrails in AWS Control Tower in the AWS documentation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check for connectivity issues after fixing guardrail violations.</td>
<td>In some cases, you might have to close specific ports or disable services to fix guardrail violations. Make sure that applications that use those ports and services are remediated before you enroll the account.</td>
<td>Application owner</td>
</tr>
</tbody>
</table>

**Enroll the account into AWS Control Tower**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the AWS Control Tower console.</td>
<td>Use the administrator account to sign in to AWS Control Tower. Do not use root user (management account) credentials to enroll an AWS Organizations account. This will display an error message.</td>
<td>AWS Control Tower administrator</td>
</tr>
<tr>
<td>Enroll the account.</td>
<td>1. From the Account Factory page in AWS Control Tower, choose Enroll account. 2. Fill in the details, including the email address associated with the account you want to enroll, the display name that will appear in AWS Control Tower, the AWS SSO email address, the first and last name of the account owner, and the OU in which you would like to enroll the account. The AWS SSO email address is your preferred user email address. You can use the same email address as the account email. 3. Choose Enroll account.</td>
<td>AWS Control Tower administrator</td>
</tr>
</tbody>
</table>

For more information, see Enroll an existing account in the AWS Control Tower documentation.
Verify the account after enrollment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the account.</td>
<td>From AWS Control Tower, choose Accounts. The account that you just enrolled has an initial state of Enrolling. When enrollment is complete, its state changes to Enrolled.</td>
<td>AWS Control Tower administrator, Member account administrator</td>
</tr>
<tr>
<td>Check for guardrail violations.</td>
<td>Guardrails defined in the OU will automatically apply to the enrolled member account. Monitor the AWS Control Tower dashboard for violations and fix them accordingly. For more information, see Guardrails in AWS Control Tower in the AWS documentation.</td>
<td>AWS Control Tower administrator, Member account administrator</td>
</tr>
</tbody>
</table>

Related resources

Documentation

- AWS Organizations terminology and concepts (AWS Organizations documentation)
- What is AWS Control Tower? (AWS Control Tower documentation)
- Removing a member account from your organization (AWS Organizations documentation)
- Creating an administrator account in AWS Control Tower (AWS Control Tower documentation)

Tutorials and videos

- AWS Control Tower Workshop (self-paced workshop)
- What is AWS Control Tower? (video)
- Provisioning Users in AWS Control Tower using AWS SSO (video)
- Enable AWS Control Tower for Existing Organization (video)

Additional information

Troubleshooting

Here are two errors you might encounter in AWS Control Tower and their resolutions:

- An unknown error occurred. Try again later, or contact AWS Support. This error occurs when you use root user credentials (management account) in AWS Control Tower to enroll a new account. AWS Service Catalog can’t map the Account Factory Portfolio or product to the root user, which results in the error message. To remediate this error, use non-root, full-access user (administrator) credentials to enroll the new account. For more information about how to create an administrator user, see Create an IAM user in the AWS Control Tower documentation.
The AWS Control Tower Activities page displays a Get Catastrophic Drift action. This action reflects a drift check of the service and does not indicate any issues with the AWS Control Tower setup. No action is required.

More patterns

- Automate SAML 2.0 federation for AWS multi-account environments that use Azure AD (p. 1583)
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK (p. 384)
- Centralized logging and multiple-account security guardrails (p. 1638)
- Check EC2 instances for mandatory tags at launch (p. 579)
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS (p. 166)
- Deploy code in multiple AWS Regions using AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild (p. 475)
- Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere (p. 483)
- Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function (p. 500)
- Migrate Windows SSL certificates to an Application Load Balancer using ACM (p. 1150)
- Perform custom actions from AWS CodeCommit events (p. 517)
- Populate your CMDB after integrating AWS Config with ServiceNow (p. 1285)
- Rotate credentials without restarting containers (p. 246)
- Tag Transit Gateway attachments automatically using AWS Organizations (p. 1542)
### Messaging & communications

**Topics**
- Automate RabbitMQ configuration in Amazon MQ (p. 742)
- More patterns (p. 746)

---

### Automate RabbitMQ configuration in Amazon MQ

*Created by Yogesh Bhatia (AWS)*

<table>
<thead>
<tr>
<th>Environment</th>
<th>Technologies</th>
<th>AWS services</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoC or pilot</td>
<td>Messaging &amp; communications; DevOps; Infrastructure</td>
<td>Amazon MQ; AWS CloudFormation</td>
</tr>
</tbody>
</table>

#### Summary

Amazon MQ is a managed message broker service that provides compatibility with many popular message brokers. Using Amazon MQ with RabbitMQ provides a robust RabbitMQ cluster managed in the Amazon Web Services (AWS) Cloud with multiple brokers and configuration options. Amazon MQ provides a highly available, secure, and scalable infrastructure, and can process a large number of messages per second with ease. Multiple applications can use the infrastructure with different virtual hosts, queues, and exchanges. However, managing these configuration options or creating the infrastructure manually can require time and effort. This pattern describes a way to manage configurations for RabbitMQ in one step, through a single file. You can embed the code provided with this pattern within any continuous integration (CI) tool such as Jenkins or Bamboo.

You can use this pattern to configure any RabbitMQ cluster. All it requires the connectivity to the cluster. Although there are many other ways to manage RabbitMQ configurations, this solution creates entire application configurations in one step, so you can manage queues and other details easily.

#### Prerequisites and limitations

**Prerequisites**

- AWS Command Line Interface (AWS CLI) installed and configured to point to your AWS account (for instructions, see [AWS CLI documentation](#))
- Ansible installed, so you can run playbooks to create the configuration
- `rabbitmqadmin` installed (for instructions, see the [RabbitMQ documentation](#))
- A RabbitMQ cluster in Amazon MQ, created with healthy Amazon CloudWatch metrics

**Additional requirements**

- Make sure to create the configurations for virtual hosts and users separately and not as part of JSON.
- Make sure that the configuration JSON is part of the repository and is version-controlled.
- The version of the `rabbitmqadmin` CLI has to be the same as the version of the RabbitMQ server, so the best option is to download the CLI from the RabbitMQ console.
- As part of the pipeline, make sure that JSON syntax is validated before each run.
Product versions

- AWS CLI version 1.18.147
- Ansible version 2.9.13
- `rabbitmqadmin` version 3.8.6 (must be the same as the RabbitMQ server version)

Architecture

Source technology stack

- An RabbitMQ cluster running on an existing on-premises virtual machine (VM) or a Kubernetes cluster (on premises or in the cloud)

Target technology stack

- Automated RabbitMQ configurations on Amazon MQ for RabbitMQ

Target architecture

There are many ways to configure RabbitMQ. This pattern uses the import configuration functionality, where a single JSON file contains all the configurations. This file applies all settings and can be managed by a version-control system such as Bitbucket or Git. This pattern uses Ansible to implement the configuration through the `rabbitmqadmin` CLI.

Tools

Tools

- `rabbitmqadmin` – A command line tool for the RabbitMQ HTTP-based API, used to manage and monitor RabbitMQ nodes and clusters.
- Ansible – An open-source tool for automating applications and IT infrastructure.
- AWS CLI – An open source tool that enables you to interact with AWS services by using commands in a command-line shell.
AWS services

- **Amazon MQ** – A managed message broker service that makes it easy to set up and operate message brokers in the cloud.
- **AWS CloudFormation** – A service that helps you set up your AWS infrastructure and speed up cloud provisioning with infrastructure as code.

Code

The JSON configuration file used in this pattern and a sample Ansible playbook are provided in the attachment.

Epics

Create your AWS infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a RabbitMQ cluster on AWS.</td>
<td>If you don't already have a RabbitMQ cluster, you can use AWS CloudFormation to create the stack on AWS. Or, you can use the Cloudformation module in Ansible to create the stack. With the latter approach, you can use Ansible for both tasks: to create the RabbitMQ infrastructure and to manage configurations.</td>
<td>AWS CloudFormation, Ansible</td>
</tr>
</tbody>
</table>

Create the Amazon MQ for RabbitMQ configuration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a properties file.</td>
<td>Download the JSON configuration file (rabbitmqconfig.json) in the attachment, or export it from the RabbitMQ console. Modify it to configure queues, exchanges, and bindings. This configuration file demonstrates the following:</td>
<td>JSON</td>
</tr>
<tr>
<td></td>
<td>- Creates two queues: sample-queue1 and sample-queue2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Creates two exchanges: sample-exchange1 and sample-exchange2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Implements the binding between the queues and exchanges</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Retrieve the details of the Amazon MQ for RabbitMQ infrastructure.</td>
<td>These configurations are performed under the root (/) virtual host, as required by rabbitmqadmin.</td>
<td>AWS CLI, Amazon MQ</td>
</tr>
<tr>
<td></td>
<td>Retrieve the following details for the RabbitMQ infrastructure on AWS: • Broker name • RabbitMQ host • RabbitMQ user name (the administrator user created during cluster creation) • RabbitMQ password</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can use the AWS Management Console or the AWS CLI to retrieve this information. These details enable the Ansible playbook to connect to your AWS account and use the RabbitMQ cluster to run commands.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Important The computer that runs the Ansible playbook must be able to access your AWS account, and AWS CLI must already be configured, as described in the Prerequisites section.</td>
<td></td>
</tr>
<tr>
<td>Create the hosts_var file.</td>
<td>Create the hosts_var file for Ansible and make sure that all the variables are defined in the file. Consider using Ansible Vault to store the password. You can configure the hosts_var file as follows (replace the asterisks with your information):</td>
<td>Ansible</td>
</tr>
<tr>
<td></td>
<td>RABBITMQ_HOST: &quot;***********.mq.us-east-2.amazonaws.com&quot; RABBITMQ_VHOST: &quot;/&quot; RABBITMQ_USERNAME: &quot;admin&quot; RABBITMQ_PASSWORD: &quot;*******&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Create an Ansible playbook.

**Description**
For a sample playbook, see `ansible-rabbit-config.yaml` in the attachment. Download and save this file. The Ansible playbook imports and manages all RabbitMQ configurations, such as queues, exchanges, and bindings, that applications require.

Follow best practices for Ansible playbooks, such as securing passwords. Use Ansible Vault for password encryption, and retrieve the RabbitMQ password from the encrypted file.

**Skills required**
Ansible

---

**Deploy the configuration**

**Task**
Run the playbook.

**Description**
Run the Ansible playbook that you created in the previous epic:

```
ansible-playbook ansible-rabbit-config.yaml
```

You can verify the new configurations on the RabbitMQ console.

**Skills required**
RabbitMQ, Amazon MQ, Ansible

---

**Related resources**
- Migrating from RabbitMQ to Amazon MQ (AWS blog post)
- Management Command Line Tool (RabbitMQ documentation)
- Create or delete an AWS CloudFormation stack (Ansible documentation)
- Migrating message driven applications to Amazon MQ for RabbitMQ (AWS blog post)

**Attachments**

attachment.zip

---

**More patterns**
- Decompose monoliths into microservices by using CQRS and event sourcing (p. 1491)
Migration

Topics
- Automate migration strategy identification and planning using AppScore (p. 747)
- Configure SAML SSO from Microsoft Azure AD to CloudEndure Migration (p. 752)
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python (p. 760)
- Get started with automated portfolio discovery (p. 763)
- Install CloudEndure Agent through AWS Systems Manager on Windows and Linux (p. 768)
- Migrate an on-premises MySQL database to Amazon RDS for MySQL (p. 774)
- Move mainframe files directly to Amazon S3 using Transfer Family (p. 779)
- Restart the CloudEndure Agent automatically without disabling SELinux after rebooting a RHEL source server (p. 787)
- Re-architect (p. 791)
- Rehost (p. 974)
- Relocate (p. 1099)
- Replatform (p. 1130)
- Migration patterns by workload (p. 1311)
- More patterns (p. 1313)

Automate migration strategy identification and planning using AppScore

Created by Lech Migdal (AWS) and Geoff Davies (AppScore Technology Limited)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Source:</th>
<th>All workloads</th>
<th>Target:</th>
<th>AWS Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>N/A</td>
<td>Workload:</td>
<td>All other workloads</td>
<td>Technologies:</td>
<td>Migration; Modernization; Websites &amp; web apps; SaaS</td>
</tr>
</tbody>
</table>

AWS services: AWS Application Discovery Service; AWS Migration Hub

Summary

On-premises applications require a transformative approach to help unlock the benefits of the Amazon Web Services (AWS) Cloud. The seven common migration strategies (7 Rs) provide you with transformation options, which vary from making technology changes in on-premises database servers to rebuilding an application by using a cloud-native microservices architecture.
Choosing to use the full 7 Rs model means that you operate at the application and business level instead of only evaluating and preparing the servers for migration. Although you can obtain server data by using tools such as AWS Migration Evaluator, other application information is often not recorded (for example, roadmap status, required recovery time objective (RTO) and recovery point objective (RPO), or data privacy requirements).

This pattern describes how to use AppScore to avoid these challenges by using an application-centric view of your portfolio. This includes a recommended transformation route to the AWS Cloud for each application against the full 7 Rs model. AppScore helps you capture application information, determine the ideal transformation route, identify the risk, complexity and benefits of cloud adoption, and quickly define the migration scopes, move groups, and schedules.

This pattern was created by AWS and AppScore Technology Limited, an AWS Partner.

Prerequisites and limitations

Prerequisites

- Existing applications that you want to migrate to the AWS Cloud.
- Existing server inventory information from a tool such as AWS Migration Evaluator. You can also import this data at a later stage in your migration.
- An existing AppScore account with Power User privileges. For more information about AppScore user accounts, see How do I assign role-based access control (RBAC) to users? in the AppScore documentation.
- An understanding of how to assign RBAC roles in AppScore. AppScore provides three subject matter expert (SME) roles that align to the questions asked in the Scoring stage. This means that an SME only answers questions relevant to their expertise and role. For more information about this, see How do I assign role-based access control (RBAC) to users? in the AppScore documentation.
- An understanding of AppScore’s recommendations, which are based on the following three categories of application attributes:
  - **Risk** – The business criticality of the application, whether it contains confidential data, data sovereignty requirements, and the number of application users or interfaces
  - **Complexity** – The application’s development language (for example, COBOL has a higher score than .NET or PHP), age, UI, or number of interfaces
  - **Benefit** – The batch processing demand, application profile, disaster recovery model, development and test environment use
- An understanding of AppScore’s four phases of iterative data capture:
  - **Signposting** – Questions that are combined with server data to produce the 7 Rs assessments. For more information, see How to signpost and score applications in the AppScore documentation.
  - **Scoring** – Questions that produce scores for risk, benefit, and complexity.
  - **Current State Assessment** – Questions that provide a current state assessment of the application.
  - **Transformation** – Questions that comprehensively evaluate the application for future state design.

Important: Only the Signposting and Scoring stages are required to receive application scores, 7 Rs assessments, and enable group planning. After you group applications and form scopes, you can complete the Current State Assessment and Transformation stages to build a more detailed overview of your application.

Architecture

The following diagram shows the AppScore workflow that uses application and server data to create a recommendation for your migration strategy and transformation plan.
Tools

- **AppScore** – AppScore helps you bridge the gap between discovery and migration implementation by providing an application-centric view of your portfolio with a recommended route to the cloud for each application against the full 7 Rs model.
- **AWS Migration Evaluator** – AWS Migration Evaluator is a migration assessment service that helps you create a directional business case for planning and migration.

Epics

Create and load the initial application list

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the list of applications</td>
<td>Log in to the AppScore portal with your user credentials. Download the Import Template from the Application page and then update the Import Template with your application’s non-technical attributes (for example, data classification or a list of attributes that can be customized). For more information about this, see How do I alter the AppScore application and business questionnaires in the AppScore documentation. <strong>Note</strong>: You can also manually add an application by choosing New Application on the Application page. You can then enter the non-technical attributes of the application.</td>
<td>Migration engineer</td>
</tr>
</tbody>
</table>
### Import the application data

**Task**
- Import the application data.

**Description**
- On the **Application** page, choose **Import Applications** to import your application data.

**Skills required**
- Migration engineer

### Capture the application and business data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review and answer Signposting and Scoring questions.</td>
<td>Open the <strong>Servers</strong> page and choose <strong>Import Servers</strong>. Choose the .csv file that contains your server data. The file can include attributes such as name, data center, operating system, virtual or physical, application name, role, database technology, environment, CPU core count and utilization, RAM size and utilization, disk size and utilization, matched machine type, and current and projected monthly costs. Confirm column mapping and choose <strong>Confirm and Import</strong>. Missing information in the imported data is highlighted on the <strong>Server</strong> page. You can resolve these gaps on this page or by using the <strong>Bulk Edit</strong> option. Servers are associated with the relevant application. However, if applications don't exist in AppScore, they are automatically created and the servers are then associated. You can also use an API connection to retrieve the data with AWS Migration Hub. For more information about this, see <a href="#">How do I import servers from AWS Migration Hub via API?</a> in the AppScore documentation. <strong>Note:</strong> If you used a discovery tool (for example, AWS Migration Evaluator) to capture performance over time, you must load an early extract of the server data as soon as possible and refresh the data.</td>
<td>App owner</td>
</tr>
</tbody>
</table>
### Create the migration schedule

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the applications for the move group.</td>
<td>Open the Planning page, choose Group Builder, and then create application move groups according to your requirements. You can add or remove attributes from the application list in the Columns section. You can also use application attributes in the Filters section to choose specific applications, which includes filtering out all applications that are already part of existing move groups.</td>
<td>Migration engineer</td>
</tr>
<tr>
<td>Create the move group.</td>
<td>Choose Group Selected, enter a name for your move group, choose the applications that you want to include in your move group, and then choose Add to Group.</td>
<td>Migration engineer</td>
</tr>
<tr>
<td>Schedule the migration.</td>
<td>On the Transformation Schedules page, AppScore</td>
<td>Migration engineer</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
--- | --- | ---
 | provides an estimated transformation duration, effort, and cost for your move group. The move group is automatically added into the overall transformation schedule.  
**Note:** You can customize the assumptions behind the effort estimation in the Planning Settings page. This helps align them with your organization's requirements. For more information about this, see How do I configure the planning settings in the AppScore documentation. |  
 | Generate the complete transformation report. | Open the Group Manager page and choose Create Application Transformation Report Doc. Choose the move groups and then choose Export. This generates a .docx file that summarizes the transformation, including the details for each move group.  
For a sample application transformation report, see Sample application transformation report from the AppScore website. | Migration engineer

Related resources

- What are the 7 Rs of an application migration?
- A closer look at AppScore
- AppScore in the AWS Marketplace

Configure SAML SSO from Microsoft Azure AD to CloudEndure Migration

*Created by Chandra Sekhar Yaratha (AWS)*

**Environment:** Production  
**Technologies:** Migration; Infrastructure; Security, identity, compliance  
**Workload:** Microsoft
Summary

This pattern helps you integrate CloudEndure Migration with Microsoft Azure Active Directory (Azure AD) by configuring Security Assertion Markup Language (SAML) single sign-on (SSO) authentication. This approach means that you can use your Azure credentials through the SAML SSO integration and federate Azure AD users with CloudEndure Migration logins.

Using SSO provides several benefits, such as signing in one-time with one account to access domain-joined devices, company resources, software as a service (SaaS) applications, and web applications. After signing in, users can launch applications from the Microsoft Office 365 or My Apps portals. Administrators can also centralize user account management and automatically add or remove user access to applications based on group membership.

Note: AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the Amazon Web Services (AWS) Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

Prerequisites and limitations

Prerequisites

- An existing CloudEndure Migration user account with administrator privileges. For more information about this, see User Management in the CloudEndure documentation.
- An existing Azure account with access to Azure AD and sufficient permissions to register a new Azure enterprise application. For more information about this, see the Microsoft Azure Active Directory documentation.

Limitations

- AWS only supports SAML 2.0
- The Azure AD Free edition doesn’t authenticate user credentials for more than ten SaaS applications and has a limit of 500,000 objects for directory objects.
The diagram shows the following workflow:

1. A user attempts to access CloudEndure Migration.
2. The user is redirected to Azure AD and enters their user name and password.
3. Azure AD encrypts the user name and password and then places them in a queue.
4. The on-premises Azure AD agent retrieves the request from the queue.
5. The agent decrypts the password using a private key.
6. The agent validates the user name and password against Azure AD.
7. Azure AD returns the results to the agent.
8. The agent returns the response to Azure AD.
9. Azure AD completes the sign-in process. If the user is successfully authenticated, they can access CloudEndure Migration.

**Technology stack**

- Azure AD
- CloudEndure Migration

**Tools**

- **Microsoft Azure Active Directory** – Azure AD is a cloud-based identity and access management service that helps users sign in and access external and internal resources.
- **CloudEndure Migration** – CloudEndure Migration is an agent-based tool that rehosts your applications on the AWS Cloud.
## Epics

### Initial setup

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add CloudEndure Migration to the enterprise applications in Azure AD. | 1. Log in to your Azure account and open the Azure AD console.  
2. Choose **Enterprise applications**.  
3. In the **All applications** pane, choose **New application**, and then choose **Non gallery application**.  
4. Enter "CloudEndure" as the application name.  
5. Edit the properties according to your requirements and then choose **Create**.  
6. The **Getting started** page then displays options for configuring the application for your organization. | Cloud administrator, Cloud engineer |
| Add an Azure AD user and group to the CloudEndure application. | 1. On the Azure AD console, choose **Users**, choose **New user**, and then create or invite a user.  
2. Return to the Azure AD console, choose **Groups**, choose **New Group**, and then enter the group information as required.  
3. Return to the Azure AD console, choose **Enterprise applications**, choose **CloudEndure**, and then choose **Assign users and groups** in the **Getting Started** section.  
4. In the **Name of the Application Users and Groups** window, choose **Add user**.  
5. In the **Add Assignment** window, choose **Users and groups**.  
6. Choose the group and users that you created earlier and choose **Select** to authenticate CloudEndure for SSO in your Azure environment. | Cloud administrator, Cloud engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information about this, see Assign users to an app that is using Azure AD as an identity provider in the Microsoft Azure documentation.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Configure SAML for your CloudEndure application. | 1. Open the CloudEndure application overview section, choose **Set up Single Sign On**, and then choose SAML.  
2. Record the login URL and Azure AD identifier.  
3. In the **SAML Signing Certificate** section, download the Federation Metadata file.  
4. Open a new tab in your web browser and log in to the CloudEndure user console.  
5. Choose the account icon and then choose **Configure SAML**.  
6. In the **Configure SAML** section, there is a hyperlink called **Bookmark this link**. You must bookmark this rescue link to regain access to the CloudEndure account if it is locked. The link is in the following format: https://console.cloudendure.com/api/v5/accounts/<account identifier>/access?username=<your email>  
**Note:** You are no longer able to sign into the CloudEndure user console with your user name and password after you configure SAML.  
7. Save the CloudEndure account identifier. This identifier is required for further configurations in Azure AD.  
8. Use the following values in the **Configure SAML** section and then choose **Save configuration**:  
   - **Identity Provider ID** – Azure AD identifier  
   - **Identity Provider URL** – Login URL  
   - **Identity Provider Certificate** – Open the Federation Metadata file and copy and paste the content that is between the | Cloud administrator, Cloud engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the CloudEndure Migration account identifier on Azure AD and test the application.</td>
<td>Configure the CloudEndure Migration account identifier on Azure AD and test the application.</td>
<td>Cloud administrator, Cloud engineer</td>
</tr>
<tr>
<td>1. In the CloudEndure application overview section of the Azure AD portal, enter the following values in the Single Sign-On section of the Basic SAML Configuration:</td>
<td>1. In the CloudEndure application overview section of the Azure AD portal, enter the following values in the Single Sign-On section of the Basic SAML Configuration:</td>
<td>Cloud administrator, Cloud engineer</td>
</tr>
</tbody>
</table>
| - Key: Identifier (Entity ID)  
  Value: https://console.cloudendure.com | - Key: Identifier (Entity ID)  
  Value: https://console.cloudendure.com | Cloud administrator, Cloud engineer  |
| - Key: Reply URL (Assertion Consumer Service URL)  
  Value: https://console.cloudendure.com/api/v5/assertionConsumerService | - Key: Reply URL (Assertion Consumer Service URL)  
  Value: https://console.cloudendure.com/api/v5/assertionConsumerService | Cloud administrator, Cloud engineer  |
| - Key: Relay State  
  Value: https://console.cloudendure.com/#/signin;<cloudendure account identifier> | - Key: Relay State  
  Value: https://console.cloudendure.com/#/signin;<cloudendure account identifier> | Cloud administrator, Cloud engineer  |
| 2. Choose Save changes. | 2. Choose Save changes. | Cloud administrator, Cloud engineer  |
| 3. In the User Attributes & Claims section, set the Unique User Identifier as the attribute that corresponds to the email that you registered in the CloudEndure user console. The attribute depends on the directory and could be the user principal name or another attribute, such as user.mail. | 3. In the User Attributes & Claims section, set the Unique User Identifier as the attribute that corresponds to the email that you registered in the CloudEndure user console. The attribute depends on the directory and could be the user principal name or another attribute, such as user.mail. | Cloud administrator, Cloud engineer  |
| 4. Remove all other claims and add a “user name” claim with the same value as user.mail. | 4. Remove all other claims and add a “user name” claim with the same value as user.mail. | Cloud administrator, Cloud engineer  |
| 5. In the Test Single Sign-on with CloudEndure, choose Sign in as current user and choose a user account. You are then automatically redirected to the CloudEndure console. | 5. In the Test Single Sign-on with CloudEndure, choose Sign in as current user and choose a user account. You are then automatically redirected to the CloudEndure console. | Cloud administrator, Cloud engineer  |
## Add new users and grant permissions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add new users and grant permissions to the CloudEndure application in Azure AD. | 1. Open the Azure AD console, choose **Manage**, choose **Users**, choose **New user**, and then choose either **Create user** or **Invite user**. Configure the remaining options according to your requirements.  
2. Grant the new user permissions to the CloudEndure application by opening the Azure AD console and choosing **Enterprise applications**. Choose **CloudEndure** and then choose **Assign users and groups**. Choose **Add user**, choose **Users**, and then choose the user that you created earlier and grant them access to the application.  
3. Open the CloudEndure user console, choose the account icon, and then choose **Manage Users**.  
4. Choose **Create User** and configure the user to receive an e-mail invitation. Ensure the account status changes from **Pending** to **Confirmed**.  

For more information about this, see [Assign users to an app that is using Azure AD as an identity provider](https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-app-inject) in the Microsoft Azure documentation. | Cloud administrator, Cloud engineer |
| User accesses CloudEndure Migration. | 1. The user opens the login page for CloudEndure Migration, chooses the **Use corporate credentials (SSO)** option, and uses the account identifier from the bookmark that you created earlier.  
2. After the user logs in, they must accept the CloudEndure terms of service. | Cloud architect, Cloud engineer |
3. After the terms of service are accepted, the user status in the CloudEndure user console updates from Pending status to Confirmed. The user can now easily login to the CloudEndure portal from the Azure application by using the following link: https://myapps.microsoft.com

Note: Each user from your account must accept the CloudEndure terms of service when they log in for the first time.

Related resources

- Azure AD SSO integration with AWS
- Azure AD pass-through authentication
- SAML Single Sign-On in CloudEndure
- Set up SAML-based SSO for an application in your Azure AD tenant
- Tutorials for integrating SaaS applications with Azure AD

Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python

Created by Venkata Naveen Koppula (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>N/A</th>
<th>Source:</th>
<th>Automation</th>
<th>Target:</th>
<th>Database in AWS Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Microsoft</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern outlines steps for automatically creating AWS CloudFormation templates for AWS Database Migration Service (AWS DMS) using Microsoft Excel and Python.

Migrating databases using AWS DMS often involves creation of AWS CloudFormation templates to provision AWS DMS tasks. Previously, creating AWS CloudFormation templates required knowledge of...
the JSON or YAML programming language. With this tool, you only need basic knowledge of Excel and how to run a Python script using a terminal or command window.

As input, the tool takes an Excel workbook that includes the names of the tables to be migrated, Amazon Resource Names (ARNs) of AWS DMS endpoints, and AWS DMS replication instances. The tool then generates AWS CloudFormation templates for the required AWS DMS tasks.

For detailed steps and background information, see the blog post Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel in the AWS Database blog.

**Prerequisites and limitations**

**Prerequisites**

- Microsoft Excel version 2016 or later
- Python version 2.7 or later
- The `xlrd` Python module (installed at a command prompt with the command: `pip install xlrd`)
- AWS DMS source and target endpoints and AWS DMS replication instance

**Limitations**

- The names of schemas, tables, and associated columns are transformed into lowercase characters at the destination endpoints.
- This tool doesn’t address the creation of AWS DMS endpoints and replication instances.
- Currently, the tool supports only one schema for each AWS DMS task.

**Architecture**

**Source technology stack**

- An on-premises database
- Microsoft Excel

**Target technology stack**

- AWS CloudFormation templates
- A database in the AWS Cloud
Tools

- Pycharm IDE, or any integrated development environment (IDE) that supports Python version 3.6
- Microsoft Office 2016 (for Microsoft Excel)

Epics

Register an AWS account, subscription, and license

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS account.</td>
<td><a href="https://aws.amazon.com">https://aws.amazon.com</a></td>
<td>General AWS</td>
</tr>
<tr>
<td>Select an AWS Region.</td>
<td>Choose the AWS Region where you want to deploy on AWS.</td>
<td>General AWS</td>
</tr>
<tr>
<td>If necessary, request a service quota increase.</td>
<td>Request a service quota increase for the AWS DMS tasks if needed.</td>
<td></td>
</tr>
</tbody>
</table>

Configure the network, AWS DMS replication instance, and endpoints

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the AWS Region, virtual private clouds (VPCs),</td>
<td></td>
<td>General AWS</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Related resources**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDR ranges, Availability Zones, and subnets.</td>
<td>The AWS DMS replication instance can connect to both on-premises and AWS databases.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Configure the AWS DMS replication instance.</td>
<td>Configure endpoints for both the source and target databases.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Configure AWS DMS endpoints.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Prepare the worksheets for AWS DMS tasks and tags

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the tables list.</td>
<td>List all tables involved in the migration.</td>
<td>Database</td>
</tr>
<tr>
<td>Prepare the tasks worksheet.</td>
<td>Prepare the Excel worksheet using the tables list you configured.</td>
<td>General AWS, Microsoft Excel</td>
</tr>
<tr>
<td>Prepare the tags worksheet.</td>
<td>Detail the AWS resource tags to attach to the AWS DMS tasks.</td>
<td>General AWS, Microsoft Excel</td>
</tr>
</tbody>
</table>

### Download and run the tool

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download and extract the template generation tool from the GitHub repository.</td>
<td>GitHub repository: <a href="https://github.com/aws-samples/dms-cloudformation-templates-generator/">https://github.com/aws-samples/dms-cloudformation-templates-generator/</a></td>
<td></td>
</tr>
<tr>
<td>Run the tool.</td>
<td>Follow the detailed instructions in the blog post listed under “References and help.”</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel (blog post)
- DMS CloudFormation Templates Generator (GitHub repository)
- Python documentation
- xlrd description and download
- AWS DMS documentation
- AWS CloudFormation documentation

### Get started with automated portfolio discovery
Summary

Assessing the portfolio and collecting metadata is a critical challenge when migrating applications and servers to the Amazon Web Services (AWS) Cloud, especially for large migrations that have more than 300 servers. Using an automated portfolio discovery tool can help you collect information about your applications, such as the number of users, frequency of use, dependencies, and information about the application’s infrastructure. This information is essential when planning migration waves so that you can properly prioritize and group applications with similar traits. Using a discovery tool streamlines communication between the portfolio team and the application owners because the portfolio team can validate the results of the discovery tool rather than manually collecting the metadata. This pattern discusses key considerations for selecting an automated discovery tool and information about how to deploy and test one in your environment.

This pattern includes a template, which is a starting point for building your own checklist of high-level activities. Next to the checklist is template for a responsible, accountable, consulted, informed (RACI) matrix. You can use this RACI matrix to determine who is responsible for each task in your checklist.

Tools

For a comprehensive list of automated portfolio discovery tools, see Discovery migration tool comparison.

Epics

Select a discovery tool

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine whether a discovery tool is appropriate for your use case.</td>
<td>A discovery tool might not be the best solution for your use case. Consider the amount of time required to select, procure, prepare, and deploy a discovery tool. It can take 4–8 weeks to set up the scanning appliance for an agentless discovery tool in your environment or to install agents to all in-scope workloads. Once deployed, you must allow 4–12 weeks for the discovery tool to collect metadata by scanning the application workloads and performing application stack analysis. If you are migrating fewer than 100 servers, you might be able to manually collect the metadata and analyze dependencies faster.</td>
<td>Migration lead, Migration engineer</td>
</tr>
</tbody>
</table>
## Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>than the time required to deploy and collect metadata with an automated discovery tool.</td>
<td></td>
</tr>
<tr>
<td>Select a discovery tool.</td>
<td>Review the <strong>Considerations for selecting an automated discovery tool</strong> in the Additional information (p. 768) section. Determine the appropriate criteria for selecting a discovery tool for your use case, and then evaluate each tool against those criteria. For a comprehensive list of automated discovery tools, see <strong>Discovery migration tool comparison</strong>.</td>
<td>Migration lead, Migration engineer</td>
</tr>
</tbody>
</table>

### Prepare for installation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the pre-deployment checklist.</td>
<td>Create a checklist of the tasks you must complete before deploying the tool. For an example, see <strong>Predeployment Checklist</strong> on the Flexera documentation website.</td>
<td>Build lead, Migration engineer, Migration lead, Network administrator</td>
</tr>
<tr>
<td>Prepare the network requirements.</td>
<td>Provision the ports, protocols, IP addresses, and routing necessary for the tool to run and access the target servers. For more information, see the installation guide for your discovery tool. For an example, see <strong>Deployment Requirements</strong> on the Flexera documentation website.</td>
<td>Migration engineer, Network administrator, Cloud architect</td>
</tr>
<tr>
<td>Prepare the account and credential requirements.</td>
<td>Identify the credentials you need to access the target servers and to install all of the tool’s components.</td>
<td>Cloud administrator, General AWS, Migration engineer, Migration lead, Network administrator, AWS administrator</td>
</tr>
<tr>
<td>Prepare the appliances on which you will install the tool.</td>
<td>Ensure that the appliances on which you will install the tool components meet the specifications and platform requirements for the tool.</td>
<td>Migration engineer, Migration lead, Network administrator</td>
</tr>
<tr>
<td>Prepare the change orders.</td>
<td>According to the change management process in your organization, prepare the any change orders needed, and</td>
<td>Build lead, Migration lead</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>ensure these change orders are approved.</td>
<td><strong>Send requirements to stakeholders.</strong> Send the pre-deployment checklist and network requirements to the stakeholders. Stakeholders should review, evaluate, and prepare the necessary requirements before proceeding with the deployment.</td>
<td>Build lead, Migration lead</td>
</tr>
</tbody>
</table>

### Deploy the tool

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the installer.</td>
<td>Download the installer or the virtual machine image. Virtual machine images typically come in Open Virtualization Format (OVF).</td>
<td>Build lead, Migration lead</td>
</tr>
<tr>
<td>Extract the files.</td>
<td>If you are using an installer, you must download and run the installer on an on-premises server.</td>
<td>Build lead, Migration lead</td>
</tr>
</tbody>
</table>
| Deploy the tool on the servers. | Deploy the discovery tool on the target, on-premises servers as follows:  
  • If your source file is a virtual machine image, deploy it into your virtual machine environment, such as VMware.  
  • If your source file is an installer, run the installer to install and set up the tool. | Build lead, Migration lead, Network administrator |
| Log in to the discovery tool. | Follow the on-screen prompts, and log in to get started with the tool. | Migration lead, Build lead |
| Activate the product. | Enter your license key. | Build lead, Migration lead |
| Configure the tool. | Enter any credentials necessary to access the target servers, such as credentials for Windows, VMware, Simple Network Management Protocol (SNMP), and Secure Shell Protocol (SSH), or databases. | Build lead, Migration lead |
Test the tool

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select test servers.</td>
<td>Identify a small set of non-production subnets or IP addresses that you can use to test the discovery tool. This helps you validate the scans quickly, identify and troubleshoot any errors quickly, and isolate your tests from production environments.</td>
<td>Build lead, Migration lead, Network administrator</td>
</tr>
<tr>
<td>Start scanning the selected test servers.</td>
<td>For an agentless discovery tool, enter the subnets or IP addresses for the selected test servers in the discovery tool console, and start the scan. For an agent-based discovery tool, install the agent on the selected test servers.</td>
<td>Build lead, Migration lead, Network administrator</td>
</tr>
<tr>
<td>Review the scan results.</td>
<td>Review the scan results for the test servers. If any errors are found, troubleshoot and fix the errors. Document the errors and solutions. You reference this information in the future, and you can add this information to your portfolio runbook.</td>
<td>Build lead, Migration lead, Network administrator</td>
</tr>
<tr>
<td>Rescan the test servers.</td>
<td>Once the rescan is complete, repeat the scan until there are no errors.</td>
<td>Build lead, Migration lead, Network administrator</td>
</tr>
</tbody>
</table>

Related resources

AWS resources for portfolio assessment and discovery tools

- Application portfolio assessment guide for AWS Cloud migration
- Discovery migration tool comparison

Deployment guides for commonly selected discovery tools

- Deploy the RN150 virtual appliance (Flexera documentation)
- FlexDeploy Getting Started Guide (Flexera documentation)
- Gatherer Installation (modelizeIT documentation)
- On-Prem Analysis Server Installation (modelizeIT documentation)
## Additional information

**Considerations for selecting an automated discovery tool**

Each discovery tool has benefits and limitations. When selecting the appropriate tool for your use case, consider the following:

- Select a discovery tool that can collect most, if not all, of the metadata you need to achieve your portfolio assessment goal.
- Identify any metadata you need to gather manually because the tool doesn’t support it.
- Provide the discovery tool requirements to stakeholders so they can review and assess the tool based on their internal security and compliance requirements, such as server, network, and credential requirements.
  - Does the tool require that you install an agent in the in-scope workload?
  - Does the tool require that you set up a virtual appliance in your environment?
- Determine your data residency requirements. Some organizations don’t want to store their data outside of their environment. To address this, you might need to install some components of the tool in the on-premises environment.
- Make sure the tool supports the operating system (OS) and OS version of the in-scope workload.
- Determine whether your portfolio includes mainframe, mid-range, and legacy servers. Most of the discovery tools can detect these workloads as dependencies, but some tools might not be able to get device details, such as utilization and server dependencies. Device42 and modernizeIT discovery tools both support mainframe and mid-range servers.

## Attachments

attachment.zip

## Install CloudEndure Agent through AWS Systems Manager on Windows and Linux

*Created by Srikanth Rangavajhala (AWS)*

<table>
<thead>
<tr>
<th><strong>Source:</strong> Windows or Linux servers (on premises, hybrid, or cloud)</th>
<th><strong>Target:</strong> Amazon EC2</th>
<th><strong>R Type:</strong> N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environment:</strong> Production</td>
<td><strong>Technologies:</strong> Migration</td>
<td><strong>Workload:</strong> Microsoft; All other workloads</td>
</tr>
<tr>
<td><strong>AWS services:</strong> Amazon EC2; AWS Systems Manager</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Summary

CloudEndure Migration uses an agent-based process for migrating your on-premises or cloud-native servers and applications to the Amazon Web Services (AWS) Cloud. The migration process requires
that you install CloudEndure Agent on the source machines that you want to migrate. You can install CloudEndure Agent through AWS Systems Manager, which helps automate operational tasks. This pattern describes steps and provides scripts for automating the CloudEndure Agent installation on one or multiple Microsoft Windows or Linux machines in a single step.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

### Prerequisites and limitations

**Prerequisites**

- AWS Systems Manager Agent (SSM Agent) installed on the source computers, and the Systems Manager role attached to the source computer (see the ssmrole.json file in the attachment)
- A CloudEndure user with the appropriate AWS Identity and Access Management (IAM) policy for migration

**Versions**

- You can use this pattern for the Windows and Linux operating systems and versions that CloudEndure supports. For a complete list, see Supported Operating Systems on the CloudEndure website.

### Architecture

**Source technology stack**

- Windows machines
- Linux machines

**Target technology stack**

- CloudEndure Migration
- AWS Systems Manager

**CloudEndure network architecture**
Tools

- **CloudEndure Migration** – CloudEndure Migration simplifies, expedites, and reduces the cost of cloud migration by automating large-scale migrations to AWS. Continuous data replication takes place in the background, without application disruption or performance impact, which ensures that data is synchronized in real time and minimizes cutover windows. When migration cutover is initiated, CloudEndure runs a highly automated machine conversion and orchestration process, which enables even the most complex applications and databases to run natively on AWS, without compatibility issues, and requires only minimal IT skills.

- **AWS Systems Manager** – AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. With Systems Manager, you can group resources, such as Amazon EC2 instances, Amazon Simple Storage Service (Amazon S3) buckets, or Amazon Relational Database Service (Amazon RDS) instances, by application, view operational data for monitoring and troubleshooting, and take action on your groups of resources. Systems Manager simplifies resource and application management, shortens the time to detect and resolve operational problems, and makes it easy to operate and manage your infrastructure securely at scale.

Epics

Sign up for CloudEndure through AWS Marketplace

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In AWS Marketplace, subscribe to CloudEndure Migration.</td>
<td>In AWS Marketplace, open the CloudEndure Migration listing, and subscribe to it.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Register an account, and confirm your email address.</td>
<td>Your CloudEndure account is created after you confirm your email address.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Set the account password, and accept terms and conditions.</td>
<td>The password must be at least 8 characters long and must contain one uppercase letter, one lowercase letter, one digit, and one special character.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

**Create a project in CloudEndure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure user console.</td>
<td>Navigate to <a href="https://console.cloudendure.com">https://console.cloudendure.com</a>. Sign in with the credentials you created in the previous step.</td>
<td>Systems administrator, CloudEndure administrator</td>
</tr>
<tr>
<td>Create a new project.</td>
<td>To create a new project, choose the plus (+) button in the upper-left corner of the CloudEndure user console.</td>
<td>Systems administrator, CloudEndure administrator</td>
</tr>
<tr>
<td>Enter a project name, and select a project type and license package.</td>
<td>Choose Live Migration as the project type. You can acquire a license through AWS Marketplace.</td>
<td>Systems administrator, CloudEndure administrator</td>
</tr>
</tbody>
</table>

**Configure replication settings**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose replication settings.</td>
<td>On the CloudEndure user console, navigate to Setup &amp; Info, and then choose the Replication settings tab. For detailed instructions, see the CloudEndure documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Set your source and target infrastructure.</td>
<td>For Live Migration Source, choose Other Infrastructure. For Target, choose the AWS Region you want to replicate your data to.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Define your replication servers.</td>
<td>In the Replication Settings tab: 1. Choose the subnet where the replication servers will be launched (customer-application-x subnet).</td>
<td>Systems administrator</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Choose the security groups to apply to the replication servers. Choose both Sentinel security groups <em>(Private Only and EgressAll)</em>.</td>
<td></td>
</tr>
</tbody>
</table>

### Install CloudEndure Agents through Systems Manager

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve the installation token.</td>
<td>On the CloudEndure user console, navigate to <strong>Machines, Machine Actions, Add Machines</strong>. Copy the installation token that is displayed on the screen. This token is automatically generated for you when your CloudEndure account is activated. For more information, see the CloudEndure documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Access Systems Manager from your AWS account.</td>
<td>From the AWS account that has the correct IAM policy, open the Systems Manager console.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Create an SSM document.</td>
<td>1. In the Systems Manager navigation pane, in the <strong>Shared Resources</strong> section, choose <strong>Documents</strong>. 2. Choose <strong>Create command or session</strong> to open the <strong>Create document</strong> page. 3. Complete the document details. For content, choose <strong>JSON</strong>, paste the contents of the CEAgentInstFromSSM.json file (in the attachment), and then choose <strong>Create document</strong>.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Run the SSM document.</td>
<td>1. In the Systems Manager navigation pane, in the <strong>Node Management</strong> section, choose <strong>Run Command</strong>. For <strong>Command document</strong>, choose the SSM document you created in the previous step. 2. In the <strong>Command parameters</strong> section, specify the CloudEndure Agent</td>
<td>Systems administrator</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>installation token you copied from the CloudEndure console.</td>
<td></td>
</tr>
<tr>
<td>3. In the <strong>Targets</strong> section, choose <strong>Choose instances manually</strong>.</td>
<td></td>
</tr>
<tr>
<td>4. In the <strong>Instances</strong> section, choose the Windows and Linux servers that you want to install the CloudEndure Agent on. You can choose one or multiple servers.</td>
<td></td>
</tr>
<tr>
<td>5. Apply other options as necessary, and then choose <strong>Run</strong> to install the CloudEndure Agent.</td>
<td></td>
</tr>
<tr>
<td>6. Check the status of the installation by using the <strong>Refresh</strong> button.</td>
<td>Systems administrator</td>
</tr>
</tbody>
</table>

**Verify the Agent installation.**

On the CloudEndure user console, switch to project view. You should see the servers where Systems Manager installed the CloudEndure Agent.

<table>
<thead>
<tr>
<th>Related resources</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>References</strong></td>
<td></td>
</tr>
<tr>
<td>• AWS Systems Manager</td>
<td></td>
</tr>
<tr>
<td>• Working with SSM Agent</td>
<td></td>
</tr>
<tr>
<td>• AWS Systems Manager FAQs</td>
<td></td>
</tr>
<tr>
<td>• CloudEndure Migration</td>
<td></td>
</tr>
<tr>
<td><strong>AWS Marketplace</strong></td>
<td></td>
</tr>
<tr>
<td>• CloudEndure Migration</td>
<td></td>
</tr>
<tr>
<td><strong>Tutorials and videos</strong></td>
<td></td>
</tr>
<tr>
<td>• CloudEndure Troubleshooting Playbook</td>
<td></td>
</tr>
<tr>
<td>• CloudEndure FAQ and Troubleshooting</td>
<td></td>
</tr>
<tr>
<td>• CloudEndure Video Guide</td>
<td></td>
</tr>
</tbody>
</table>

**Attachments**

attachment.zip
Migrate an on-premises MySQL database to Amazon RDS for MySQL

Summary

This pattern provides guidance for migrating an on-premises MySQL database to Amazon Relational Database Service (Amazon RDS) for MySQL. The pattern discusses the use of AWS Database Migration Service (AWS DMS) or native MySQL tools such as `mysqldbcopy` and `mysqldump` for the migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A MySQL source database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- MySQL versions 5.5, 5.6, 5.7, 8.0. For the latest list of supported versions, see MySQL on Amazon RDS in the AWS documentation. If you're using AWS DMS, see also Using a MySQL-Compatible Database as a Target for AWS DMS for MySQL versions supported by AWS DMS.

Architecture

Source technology stack

- An on-premises MySQL database

Target technology stack

- An Amazon RDS DB instance running MySQL
AWS data migration architecture

*Using AWS DMS:*
Using native MySQL tools:
AWS Prescriptive Guidance Patterns

Tools

- **AWS DMS** - AWS Database Migration Service (AWS DMS) supports several source and target databases. For information about MySQL source and target databases supported by AWS DMS, see Migrating MySQL-Compatible Databases to AWS. If your source database isn't supported by AWS DMS, you must choose another method to migrate your data.

- **Native MySQL tools** - `mysqldbcopy` and `mysqldump`

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose the proper instance type based on capacity, storage features, and network features.</strong></td>
<td><strong>Identify the network access security requirements for source and target databases.</strong></td>
<td><strong>Identify the application migration strategy.</strong></td>
</tr>
<tr>
<td><strong>DBA, SysAdmin</strong></td>
<td><strong>DBA, SysAdmin</strong></td>
<td><strong>DBA, SysAdmin, App owner</strong></td>
</tr>
</tbody>
</table>

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a virtual private cloud (VPC).</strong></td>
<td><strong>Create security groups.</strong></td>
<td><strong>Configure and start an Amazon RDS DB instance running MySQL.</strong></td>
</tr>
<tr>
<td><strong>SysAdmin</strong></td>
<td><strong>SysAdmin</strong></td>
<td><strong>For instructions, see <a href="https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateInstance.html">https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateInstance.html</a>.</strong></td>
</tr>
</tbody>
</table>

### Migrate data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use native MySQL tools or third-party tools to migrate database objects and data.</strong></td>
<td>For instructions, see the documentation for MySQL tools such as mysqldbcopy and mysqldump.</td>
<td><strong>DBA</strong></td>
</tr>
</tbody>
</table>

### Migrate data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Migrate data with AWS DMS.</strong></td>
<td></td>
<td><strong>DBA</strong></td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Follow the application migration strategy.</strong></td>
<td></td>
<td><strong>DBA, SysAdmin, App owner</strong></td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients over to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

References

- AWS DMS website
- Amazon RDS Pricing
- VPCs and Amazon RDS
- Amazon RDS Multi-AZ Deployments

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon RDS

Move mainframe files directly to Amazon S3 using Transfer Family

Created by Luis Gustavo Dantas (AWS)

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Source: Mainframe</th>
<th>Target: Amazon S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type: N/A</td>
<td>Workload: IBM</td>
<td>Technologies: Migration; Storage &amp; backup</td>
</tr>
</tbody>
</table>
Summary

As part of the modernization journey, you can face the challenge of transferring files between your on-premises servers and the Amazon Web Services (AWS) Cloud. Transferring data from mainframes can be a significant challenge because mainframes typically can't access modern data stores like Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), or Amazon Elastic File System (Amazon EFS).

Many customers use intermediate staging resources, such as on-premises Linux, Unix, or Windows servers, to transfer files to the AWS Cloud. You can avoid this indirect method by using AWS Transfer Family with the Secure Shell (SSH) File Transfer Protocol (SFTP) to upload mainframe files directly to Amazon S3.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) with a subnet that’s reachable by your legacy platform
- A Transfer Family endpoint for your VPC
- Mainframe Virtual Storage Access Method (VSAM) files converted to sequential, fixed-length files (IBM documentation)

Limitations

- SFTP transfers files in binary mode by default, which means that files are uploaded to Amazon S3 with EBCDIC encoding preserved. If your file doesn't contain binary or packed data, then you can use the `sftp ascii subcommand` (IBM documentation) to convert your files to text during the transfer.
- You must unpack mainframe files (AWS Prescriptive Guidance) that contain packed and binary content to use these files in your target environment.
- Amazon S3 objects can range in size from a minimum of 0 bytes to a maximum of 5 TB. For more information about Amazon S3 capabilities, see Amazon S3 FAQs.

Architecture

Source technology stack

- Job control language (JCL)
- z/OS Unix shell and ISPF
- SFTP
- VSAM and flat files

Target technology stack

- Transfer Family
- Amazon S3
- Amazon Virtual Private Cloud (Amazon VPC)
**Target architecture**

The following diagram shows a reference architecture for using Transfer Family with SFTP to upload mainframe files directly to an S3 bucket.

![Diagram](image)

The diagram shows the following workflow:

1. You use a JCL job to transfer your mainframe files from the legacy mainframe to the AWS Cloud through Direct Connect.
2. Direct Connect enables your network traffic to remain on the AWS global network and bypass the public internet. Direct Connect also enhances the network speed, starting at 50 Mbps and scaling up to 100 Gbps.
3. The VPC endpoint enables connections between your VPC resources and the supported services without using the public internet. Access to Transfer Family and Amazon S3 achieves high availability by taking place through the elastic network interfaces located in two private subnets and Availability Zones.
4. Transfer Family authenticates users and uses SFTP to receive your files from the legacy environment and move them to an S3 bucket.

**Automation and scale**

After the Transfer Family service is in place, you can transfer an unlimited number of files from the mainframe to Amazon S3 by using a JCL job as the SFTP client. You can also automate the file transfer by using a mainframe batch job scheduler to run the SFTP jobs when you're ready to transfer the mainframe files.

**Tools**

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This
virtual network closely resembles a traditional network that you’d operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

- **AWS Transfer Family** – AWS Transfer Family enables you to securely scale your recurring business-to-business file transfers to Amazon S3 and Amazon EFS using SFTP, FTPS, and FTP protocols.

## Epics

### Create the S3 bucket and the access policy

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the S3 bucket.</td>
<td>Create an S3 bucket to host the files that you transfer from your legacy environment.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create the IAM role and policy.</td>
<td>Transfer Family uses your AWS Identity and Access Management (IAM) role to grant access to the S3 bucket that you created earlier.</td>
<td>General AWS</td>
</tr>
<tr>
<td></td>
<td><strong>Create an IAM role</strong> that includes the following <strong>IAM policy:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Version&quot;: &quot;2012-10-17&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Statement&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Sid&quot;: &quot;UserFolderListing&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:ListBucket&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:GetBucketLocation&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>],</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Effect&quot;: &quot;Allow&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Resource&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;arn:aws:s3:::&lt;your-bucket-name&gt;&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Sid&quot;: &quot;HomeDirObjectAccess&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Effect&quot;: &quot;Allow&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:PutObject&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:GetObjectAcl&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:GetObject&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:DeleteObjectVersion&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;s3:DeleteObject&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>


### Define the transfer service

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the SFTP server.| 1. Sign in to the AWS Management console, open the Transfer Family console, and then choose Create server.  
2. Choose only SFTP (SSH File Transfer Protocol) - file transfer over Secure Shell protocol and then choose Next.  
3. For Identity provider, choose Service managed and then choose Next.  
4. For Endpoint type, choose VPC hosted.  
5. For Access, choose Internal.  
6. For VPC, choose your VPC.  
7. In the Availability Zones section, choose your Availability Zones and subnets.  
8. In the Security Groups section, choose your security group, and then choose Next.  
9. For Domain, choose Amazon S3 and then choose Next.  
10. Leave the default options on the Configure additional details page and then choose Next.  
11. Choose Create server. | General AWS |

Note: You must choose the Transfer use case when you create the IAM role.
### Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> For more information about how to set up an SFTP server, see <a href="https://docs.aws.amazon.com/transfer/latest/userguide/create-sftp-enabled-server.html">Create an SFTP-enabled server</a> (AWS Transfer Family User Guide).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Get the server address.

1. Open the Transfer Family console and choose your server ID in the **Server ID** column.
2. In the **Endpoint details** section, for **Endpoint type**, choose the endpoint ID. This takes you to the Amazon VPC console.
3. On the **Details** tab of the Amazon VPC console, find the DNS names next to **DNS names**.

**General AWS**

#### Create the SFTP client key pair.

Create an SSH key pair for either **Microsoft Windows** or **macOS/Linux/UNIX**.

**General AWS, SSH**

#### Create the SFTP user.

1. Open the Transfer Family console, choose **Servers** from the navigation pane, and then select your server.
2. In the **Server ID** column, choose the server ID for your server and then choose **Add user**.
3. For **Username**, enter a user name that matches your SSH key pair user name.
4. For **Role**, choose the IAM role that you created earlier.
5. For **Home directory**, choose the S3 bucket that you created earlier.
6. For **SSH public keys**, enter the key pair that you created earlier.
7. Choose **Add**.

**General AWS**

### Transfer the mainframe file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Send the SSH private key to the mainframe.</strong></td>
<td>Use SFTP or SCP to send the SSH private key to the legacy environment.</td>
<td><strong>Mainframe, z/OS Unix shell, FTP, SCP</strong></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>SFTP example:</td>
<td>sftp [USERNAME@mainframeIP] [password] cd [/u/USERNAME] put [your-key-pair-file]</td>
<td></td>
</tr>
<tr>
<td>SCP example:</td>
<td>scp [your-key-pair-file] [USERNAME@MainframeIP]:/[u/USERNAME]</td>
<td></td>
</tr>
</tbody>
</table>

Next, store the SSH key in the z/OS Unix file system under the user name that will later run the file transfer batch job (for example, /u/CONTROLM).

**Note:** For more information about z/OS Unix shell, see *An introduction to the z/OS shells* (IBM documentation).
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create the JCL SFTP client. | Because mainframes don't have a native SFTP client, you must use the BPXBATCH utility to run the SFTP client from the z/OS Unix shell. In the ISPF editor, create the JCL SFTP client. For example:  
  ```  
  //JOBNAME JOB ...
  //
  **********************************************************************
  //SFTP EXEC
  PGM=BPXBATCH,REGION=0M
  //STDPARM DD *
  SH cp "//'MAINFRAME.FILE.NAME'" filename.txt;
  echo 'put filename.txt' > uplcmd;
  sftp -b uplcmd -i ssh_private_key_file ssh_username@<transfer service ip or DNS>;
  //SYSPRINT DD SYSOUT=*  
  //STDOUT DD SYSOUT=*  
  //STDENV DD *
  //STDERR DD SYSOUT=*  
  ``` | JCL, Mainframe, z/OS Unix shell |
| Run the JCL SFTP client. | 1. In the ISPF editor, enter SUB, and then press the ENTER key after the JCL job is created.  
2. Monitor the mainframe's file transfer batch job activity in SDSF. | Mainframe, JCL, ISPF |

Note: For more information about how to run a command in the z/OS Unix shell, see The BPXBATCH utility (IBM documentation). For more information about how to create or edit JCL jobs in z/OS, see What is ISPF? and The ISPF editor (IBM documentation).
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Validate the file transfer. | 1. Sign in to the AWS Management console, open the Amazon S3 console, and then choose Buckets from the navigation pane.  
2. Choose the bucket that's associated with your Transfer Family.  
3. In the Objects section of the Objects tab, find the file that you transferred from the mainframe.                                                                                                                   | General AWS      |
| Automate the JCL SFTP client.| Use job scheduler to automatically trigger the JCL SFTP client.  
**Note:** You can use mainframe job schedulers, such as BMC Control-M or CA Workload Automation, to automate batch jobs for file transfers based on time and other batch job dependencies.                                                                                                             | Job scheduler    |

## Related resources

- How AWS Transfer Family works
- Mainframe Modernization with AWS

### Restart the CloudEndure Agent automatically without disabling SELinux after rebooting a RHEL source server

*Created by Anil Kunapareddy (AWS)*

**Environment:** Production  
**Technologies:** Migration  
**Workload:** Open-source  
**AWS services:** Amazon EC2

## Summary

CloudEndure Migration helps simplify, expedite, and automate the migration of your Red Hat Enterprise Linux (RHEL) workload to the Amazon Web Services (AWS) Cloud. CloudEndure provides real-time, asynchronous, block-level replication. This means that you can continue normal IT operations during the entire replication process.
These IT operations might require that you reboot or restart your RHEL source server during the migration. If this happens, the CloudEndure Agent will not restart automatically and your data replication will stop. Typically, you can set Security-Enhanced Linux (SELinux) to disabled or permissive mode to avoid a restart. However, your organization’s security policies might prohibit disabling SELinux, and you might also have to relabel your files.

This pattern describes how to automatically restart the CloudEndure Agent without turning off SELinux when your RHEL source server reboots or restarts during a migration.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

## Prerequisites and limitations

### Prerequisites

- An on-premises RHEL workload being migrated to the AWS Cloud.
- An active CloudEndure account.
- An existing CloudEndure migration project for your RHEL migration, configured according to your requirements. Make sure you choose **Live migration** as your project type.
- An existing CloudEndure Agent Installation Token, which is provided to you when your CloudEndure account is activated.
- An active AWS account.
- An existing AWS Identity and Access Management (IAM) policy for CloudEndure.

### Architecture

#### Technology stack

- RHEL version 7 or later

### Tools

- **CloudEndure Migration** - CloudEndure helps you simplify, expedite, and automate large-scale migration to AWS.

The following table provides a list of Linux commands that you will run on your RHEL source server. These are also described in the epics and stories for this pattern.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#systemctl --version</td>
<td>Identifies system version</td>
</tr>
<tr>
<td>#systemctl list-units --type=service</td>
<td>Lists all active services available on RHEL server</td>
</tr>
<tr>
<td>#systemctl list-units --type=service</td>
<td>grep running</td>
</tr>
<tr>
<td>#systemctl list-units --type=service</td>
<td>grep failed</td>
</tr>
<tr>
<td>restorecon -Rv /etc/rc.d/init.d/ce-agent</td>
<td>Changes the context to “cloudendure-agent.”</td>
</tr>
</tbody>
</table>
yum install policycoreutils*  
Contains the policy core utilities required for the operation of a SELinux system

ausearch -c "insmod" --raw | audit2allow -M my-modprobe  
Installs the policy

semodule -i my-modprobe.pp  
Makes the policy active

cat my-modprobe.te  
Displays the contents of "my-modprobe.te"

semodule -l | grep my-modprobe  
Loads the policy to the SELinux module

### Epics

**Install the CloudEndure Agent and restart or reboot your RHEL source server**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the CloudEndure Agent.</td>
<td>Sign in to the CloudEndure User Console, choose your existing RHEL migration project, and install the CloudEndure Agent. For more information, see the &quot;Related resources&quot; section. After the CloudEndure Agent is installed, the RHEL source server will appear in the CloudEndure console and the initial replication process will begin.</td>
<td>CloudEndure administrator, OS administrator</td>
</tr>
<tr>
<td>Restart or reboot the RHEL source server.</td>
<td>Restart or reboot your RHEL source server after it enters &quot;Continuous data replication&quot; mode.</td>
<td>CloudEndure administrator, OS administrator</td>
</tr>
<tr>
<td>Check data replication status.</td>
<td>Wait for one hour and then check the status of the data replication task by navigating to &quot;Data replication progress&quot; on the CloudEndure console. You should see a &quot;Communication with the source is lost&quot; message.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

**Check CloudEndure Agent status in the RHEL source server**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify system version.</td>
<td>Open the CLI for your RHEL source server and run the &quot;#systemctl --version&quot; command to identify the system version.</td>
<td>OS administrator</td>
</tr>
</tbody>
</table>
### List all active services.
- Run the `"#systemctl list-units --type=service"` command to list all active services available on the RHEL server.
- **Skills required:** OS administrator

### List all services running on the RHEL server.
- Run the `"#systemctl list-units --type=service | grep running"` command to list all services currently running on the RHEL server.
- **Skills required:** OS administrator

### List all services that failed to load.
- Run the `"#systemctl list-units --type=service | grep failed"` command to list all services that failed to load after the RHEL server rebooted or restarted.
- **Skills required:** OS administrator

### Create and run the SELinux module

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the “restorecon -Rv /etc/rc.d/init.d/ce-agent&quot; command.</td>
<td>In your RHEL source server's CLI, run the “restorecon -Rv /etc/rc.d/init.d/ce-agent&quot; command to change the context to “cloudendure-agent.”</td>
<td>OS administrator</td>
</tr>
<tr>
<td>Run the “yum install policycoreutils*&quot; command.</td>
<td>Run the “yum install policycoreutils*&quot; command that contains the policy core utilities required for the operation of a SELinux system.</td>
<td>OS administrator</td>
</tr>
<tr>
<td>Run the “ausearch -c &quot;insmod&quot; --raw</td>
<td>audit2allow -M my-modprobe&quot; command.</td>
<td>Run the “ausearch -c &quot;insmod&quot; --raw</td>
</tr>
<tr>
<td>Run the “cat my-modprobe.te&quot; command</td>
<td>Run the “cat my-modprobe.te&quot; command to display the contents of “my-modprobe.te&quot;</td>
<td>OS administrator</td>
</tr>
<tr>
<td>Run the “semodule -i my-modprobe.pp&quot; command.</td>
<td>Run the “semodule -i my-modprobe.pp&quot; command to make the policy package active.</td>
<td>OS administrator</td>
</tr>
<tr>
<td>Run the “semodule -l</td>
<td>grep my-modprobe&quot; command.</td>
<td>Run the “semodule -l</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“permissive” mode during your migration.</td>
<td>CloudEndure administrator, OS administrator</td>
</tr>
<tr>
<td>Reboot or restart the RHEL source server and verify the data replication status.</td>
<td>Open the CloudEndure User Console, navigate to “Data replication progress,” and then reboot or restart your RHEL source server. Data replication should now resume automatically after the RHEL source server reboots or restarts.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

- Install the CloudEndure Agent on your source server
- IAM policy for CloudEndure
- CloudEndure troubleshooting playbook

Re-architect

Topics

- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT (p. 792)
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation (p. 797)
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible Edition (p. 806)
- Migrate Oracle native functions to PostgreSQL using extensions (p. 811)
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS (p. 816)
- Migrate an on-premises ThoughtSpot Falcon database to Amazon Redshift (p. 822)
- Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL (p. 830)
- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 836)
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS (p. 840)
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS (p. 847)
- Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 854)
- Migrate from Oracle to Amazon DocumentDB using AWS DMS (p. 860)
- Migrate from an on-premises Oracle database or Amazon RDS for Oracle to Amazon DynamoDB using AWS DMS and AWS SCT (p. 865)
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT (p. 870)
- Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 876)
- Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL using an Oracle bystander and AWS DMS (p. 880)
• Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate (p. 888)
• Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT (p. 891)
• Migrate an on-premises Oracle database to Amazon Redshift using AWS SCT data extraction agents (p. 899)
• Migrate an Oracle database to Amazon Aurora MySQL using AWS DMS and AWS SCT (p. 902)
• Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 908)
• Migrate data from an on-premises Oracle database to Aurora PostgreSQL (p. 915)
• Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS (p. 922)
• Migrate an SAP ASE database to Aurora MySQL using AWS DMS (p. 926)
• Migrate an SAP ASE database to Aurora PostgreSQL using AWS DMS (p. 931)
• Migrate from SAP ASE to PostgreSQL on Amazon EC2 using AWS DMS (p. 936)
• Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS (p. 940)
• Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents (p. 944)
• Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents (p. 947)
• Migrate an on-premises Vertica database to Amazon Redshift using AWS SCT data extraction agents (p. 950)
• Migrate an on-premises MySQL database to Aurora MySQL (p. 953)
• Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible (p. 958)
• Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL (p. 969)

Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT

Created by Pinesh Singal (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon RDS PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Re-architect</td>
<td>Workload:</td>
<td>Oracle; Open-source</td>
<td>Technologies:</td>
<td>Migration; Databases; Modernization</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon EC2;</td>
<td></td>
<td>Amazon RDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

Many migration strategies and approaches run in multiple phases that can last from a few weeks to several months. During this time, you can experience delays because of patching or upgrades in the source Oracle DB instances that you want to migrate to PostgreSQL DB instances. To avoid this situation, we recommend that you incrementally migrate the remaining Oracle database code to PostgreSQL database code.

This pattern provides an incremental migration strategy with no downtime for a multi-terabyte Oracle DB instance that has a high number of transactions performed after your initial migration and that
must be migrated to a PostgreSQL database. You can use this pattern's step-by-step approach to incrementally migrate an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance without signing in to the Amazon Web Services (AWS) Management Console.

The pattern uses Oracle SQL Developer to find the differences between two schemas in the source Oracle database. You then use AWS Schema Conversion Tool (AWS SCT) to convert the Amazon RDS for Oracle database schema objects to Amazon RDS for PostgreSQL database schema objects. You can then run a Python script in the Windows Command Prompt to create AWS SCT objects for the incremental changes to the source database objects.

**Note:** Before you migrate your production workloads, we recommend that you run a proof of concept (PoC) for this pattern's approach in a testing or non-production environment.

### Prerequisites and limitations

#### Prerequisites

- An active AWS account.
- An existing Amazon RDS for Oracle DB instance.
- An existing Amazon RDS for PostgreSQL DB instance.
- AWS SCT, installed and configured with JDBC drivers for Oracle and PostgreSQL database engines. For more information about this, see [Installing AWS SCT](#) and [Installing the required database drivers](#) in the AWS SCT documentation.
- Oracle SQL Developer, installed and configured. For more information about this, see the [Oracle SQL Developer](#) documentation.
- The `incremental-migration-sct-sql.zip` file (attached), downloaded to your local computer.

#### Limitations

- The minimum requirements for your source Amazon RDS for Oracle DB instance are:
  - Oracle versions 10.2 and later (for versions 10.x), 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c for the Enterprise, Standard, Standard One, and Standard Two editions.
- The minimum requirements for your target Amazon RDS for PostgreSQL DB instance are:
  - PostgreSQL versions 9.4 and later (for versions 9.x), 10.x, and 11.x
- This pattern uses Oracle SQL Developer. Your results might vary if you use other tools to find and export schema differences.
- The SQL scripts generated by Oracle SQL Developer can raise transformation errors, which means that you need to perform a manual migration.
- If the AWS SCT source and target test connections fail, make sure that you configure the JDBC driver versions and inbound rules for the virtual private cloud (VPC) security group to accept incoming traffic.

#### Product versions

- Amazon RDS for Oracle DB instance version 12.1.0.2 (version 10.2 and later)
- Amazon RDS for PostgreSQL DB instance version 11.5 (version 9.4 and later)
- Oracle SQL Developer version 19.1 and later
- AWS SCT version 1.0.632 and later
Architecture

Source technology stack

- Amazon RDS for Oracle DB instance

Target technology stack

- Amazon RDS for PostgreSQL DB instance

Source and target architecture

The following diagram shows the migration of an Amazon RDS for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance.

The diagram shows the following migration workflow:

1. Open Oracle SQL Developer and connect to the source and target databases.
2. Generate a diff report and then generate the SQL scripts file for the schema difference objects. For more information about diff reports, see Detailed diff reports in the Oracle documentation.
3. Configure AWS SCT and run the Python code.
4. The SQL scripts file converts from Oracle to PostgreSQL.
5. Run the SQL scripts file on the target PostgreSQL DB instance.

Automation and scale

You can automate this migration by adding additional parameters and security-related changes for multiple functionalities in a single program to your Python script.

Tools

- AWS SCT – AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another.
• Oracle SQL Developer – Oracle SQL Developer is an integrated development environment (IDE) that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments.

Code

The incremental-migration-sct-sq1.zip file (attached) contains the complete source code for this pattern.

Epics

Create the SQL scripts file for the source database schema differences

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Run Database Diff in Oracle SQL Developer. | 1. Sign in to your source Oracle DB instance, choose Tools, and then choose Database Diff.  
2. Choose your source database in Source Connection.  
3. Choose the updated or patched source database in Destination Connection.  
4. Configure the remaining options according to your requirements, choose Next, and then choose Finish to generate the diff report. | DBA |
| Generate the SQL scripts file. | Choose Generate Script to generate the differences in the SQL files.  
This generates the SQL scripts file that AWS SCT uses to convert your database from Oracle to PostgreSQL. | DBA |

Use the Python script to create the target DB objects in AWS SCT

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure AWS SCT with the Windows Command Prompt. | 1. Copy the AWSSchemaConversionToolBatch.jar file from your pre-installed AWS SCT folder and paste it into your working directory.  
2. Deploy the Python code from the run_aws_sct_sql.py file from the incremental- | DBA |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>migration-sct-sql.zip folder (attached). This creates .xml files and .sct files in the projects directory with your source and target database environment configuration details. It also reads the SQL scripts file that you generated in Oracle SQL Developer. Finally, it creates .sql file objects in the output directory.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Configure the source and target environment configuration details in the database_migration.txt file by using the following format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>#source_vendor,source_hostname,source_dbname,source_user,source_pwd,source_schema,source_port,target_vendor,target_hostname,target_user,target_pwd,target_dbname,target_port</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORACLE,myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com,ORCL,orcl,orcl1234,orcl,1521,ORCL,POSTGRESQL,mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com,pguser,pgpassword,pgdb,5432</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Modify the AWS SCT configuration parameters according to your requirements and then copy the SQL scripts file into your working directory in the input subdirectory.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Run the Python script.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Run the Python script by using the following command: $ python run_aws_sct_sql.py database_migration.txt</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>2. This creates the DB objects SQL file. Non-converted codes with transformation errors can be manually converted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create the objects in Amazon RDS for PostgreSQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Run the SQL files and create objects in your Amazon RDS for PostgreSQL DB instance.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Related resources**

- Oracle on Amazon RDS
- PostgreSQL on Amazon RDS
- Using the AWS SCT user interface
- Using Oracle as a source for AWS SCT
Attachments
attachment.zip

Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation

Created by Pinesh Singal (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Amazon RDS for Oracle</th>
<th>Target:</th>
<th>Amazon RDS for PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Re-architect</td>
<td>Workload:</td>
<td>Oracle; Open-source</td>
<td>Technologies:</td>
<td>Migration; Databases</td>
</tr>
</tbody>
</table>

AWS services: AWS DMS; Amazon RDS; AWS SCT

Summary

This pattern provides an approach with minimal downtime for migrating a multi-terabyte Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance, without signing in to the Amazon Web Services (AWS) Management Console.

This approach helps your organization avoid manual configurations and individual migrations using the AWS Schema Conversion Tool (AWS SCT) and AWS Database Migration Service (AWS DMS) consoles. Instead, you set up a one-time configuration for multiple databases and perform the migrations using AWS SCT and AWS DMS on the AWS Command Line Interface (AWS CLI).

The pattern uses AWS SCT to convert database schema objects from Amazon RDS for Oracle to Amazon RDS for PostgreSQL and then uses AWS DMS to migrate the data. Using Python scripts in AWS CLI, you create AWS SCT objects and AWS DMS tasks with an AWS CloudFormation template.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing Amazon RDS for Oracle DB instance.
- An existing Amazon RDS for PostgreSQL DB instance.
- An understanding of the following AWS DMS migration task types: full-load, cdc, full-load-and-cdc. For more information about this, see Creating a task in the AWS DMS documentation.
- AWS SCT, installed and configured with Java Database Connectivity (JDBC) drivers for Oracle and PostgreSQL database engines. For more information about this, see Installing AWS SCT and Installing the required database drivers in the AWS SCT documentation.
- The AWSSchemaConversionToolBatch.jar file from the installed AWS SCT folder, copied to your working directory.
AWS Prescriptive Guidance Patterns
Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL using AWS SCT and AWS DMS

- The cli-sct-dms-cft.zip file (attached), downloaded and extracted in your working directory.

- The most recent AWS DMS replication instance engine version. For more information about this, see How do I create an AWS DMS replication instance in the AWS Support documentation and AWS DMS 3.4.4 release notes in the AWS DMS documentation.

- AWS CLI version 2, installed and configured with your access key ID, secret access key, and default AWS Region name for the Amazon Elastic Compute Cloud (Amazon EC2) instance or operating system (OS) where the scripts are run. For more information about this, see Installing, updating, and uninstalling the AWS CLI version 2 and Configuring the AWS CLI in the AWS CLI documentation.

- Familiarity with AWS CloudFormation templates. For more information about this, see AWS CloudFormation concepts in the AWS CloudFormation documentation.

- Python, installed and configured on the Amazon EC2 instance or OS where the scripts are run. For more information about this, see the Python documentation.

Limitations

- The minimum requirements for your source Amazon RDS for Oracle DB instance are:
  - Oracle versions 12c (v12.1.0.2, v12.2.0.1), 18c (v18.0.0.0) and 19c (v19.0.0.0) for the Enterprise, Standard, Standard One, and Standard Two editions.
  - Although Amazon RDS supports Oracle 18c (v18.0.0.0), this version is on a deprecation path because Oracle no longer provide patches for 18c after the end-of-support date. For more information about this, see Oracle on Amazon RDS in the Amazon RDS documentation.
  - Amazon RDS for Oracle 11g is no longer supported.
- The minimum requirements for your target Amazon RDS for PostgreSQL DB instance are:
  - PostgreSQL versions 9 (versions 9.5 and 9.6), 10.x, 11.x, 12.x, and 13.x

Product versions

- Amazon RDS for Oracle DB instance version 12.1.0.2 and later
- Amazon RDS for PostgreSQL DB instance version 11.5 and later
- AWS CLI version 2
- AWS SCT most recent version

Architecture

Source technology stack

- Amazon RDS for Oracle

Target technology stack

- Amazon RDS for PostgreSQL

Source and Target architecture

The following diagram shows the migration of an Amazon RDS for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance using AWS DMS and Python scripts.
The diagram shows the following migration workflow using AWS CLI:

1. The Python script uses AWS SCT to connect to the source and target DB instances.
2. The user starts AWS SCT with the Python script, converts the Oracle code to PostgreSQL code, and runs it on the target DB instance.
3. The Python script creates AWS DMS replication tasks for the source and target DB instances.
4. The user deploys Python scripts to start the AWS DMS tasks and then stops the tasks after the data migration is complete.

**Automation and scale**

You can automate this migration by adding additional parameters and security-related changes for multiple functionalities in a single program to your Python script.

**Tools**

- **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
- **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. This pattern converts the .csv input file to a .json input file using a Python script. The .json file is used in AWS CLI commands to create an AWS CloudFormation stack that creates multiple AWS DMS replication tasks with Amazon Resource Names (ARNs), migration types, task settings, and table mappings.
AWS Prescriptive Guidance Patterns
Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL using AWS SCT and AWS DMS

- **AWS DMS** – AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. This pattern uses AWS DMS to create, start, and stop tasks with a Python script run over the command-line and create the AWS CloudFormation template.

- **AWS SCT** – AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another. This pattern requires the AWS-SchemaConversionToolBatch.jar file from the installed AWS SCT directory.

**Code**

The cli-sct-dms-cft.zip file (attached) contains the complete source code for this pattern.

**Epics**

Configure AWS SCT and create database objects in AWS CLI

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure AWS SCT to run from AWS CLI.</td>
<td>1. Configure the source and target environment configuration details in the database_migration.txt file by using the following format: #source_vendor,source_hostnam,source_dbname,source_user,source_pwd,source_port,target_vendor,target_hostname,target_user,target_pwd,target_dbname,target_port ORACLE,myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com,ORCL,orcl,orcl1234,1521,POSTGRESQL,mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com,puser,pgpassword,pgdb,5432. 2. Modify the AWS SCT configuration parameters according to your requirements in the following files: project_settings.xml, Oracle_PG_Test_Batch.xml, and ORACLE-orcl-to-POSTGRESQL.xml.</td>
<td>DBA</td>
</tr>
<tr>
<td>Run the run_aws_sct.py Python script.</td>
<td>Run the run_aws_sct.py Python script by using the following command: $ python run_aws_sct.py database_migration.txt. The Python script converts the database objects from Oracle to PostgreSQL and creates SQL files in PostgreSQL format. The script also creates the Database migration assessment report.pdf file that provides you with detailed recommendations and conversion statistics for database objects.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL using AWS SCT and AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create objects in Amazon RDS for PostgreSQL. | 1. Manually modify the SQL files generated by AWS SCT, if required.  
2. Run the SQL files and create objects in your Amazon RDS for PostgreSQL DB instance. | DBA |

**Configure and create AWS DMS tasks using AWS CLI and AWS CloudFormation**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an AWS DMS replication instance. | Sign in to the AWS Management Console, open the AWS DMS console, and create a replication instance that is configured according to your requirements.  
For more information about this, see [Creating a replication instance](https://docs.aws.amazon.com/dms/latest/userguide/ug-tasks-dms-configure-create-replication-instance.html) in the AWS DMS documentation and [How do I create an AWS DMS replication instance](https://docs.aws.amazon.com/dms/latest/ug/tasks-dms-create-replication-instance.html) in the AWS Support documentation. | DBA |
| Create the source endpoint. | On the AWS DMS console, choose **Endpoints** and then create a source endpoint for the Oracle database according to your requirements.  
**Note:** The extra connection attribute must be `numberDataTypeScale` with a `-2` value.  
For more information about this, see [Creating source and target endpoints](https://docs.aws.amazon.com/dms/latest/userguide/ug-tasks-dms-create-endpoint.html) in the AWS DMS documentation. | DBA |
| Create the target endpoint. | On the AWS DMS console, choose **Endpoints** and then create a target endpoint for the PostgreSQL database according to your requirements.  
For more information about this, see [Creating source and target endpoints](https://docs.aws.amazon.com/dms/latest/userguide/ug-tasks-dms-create-endpoint.html) in the AWS DMS documentation. | DevOps engineer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure the AWS DMS replication details to run from AWS CLI. | Configure the AWS DMS source and target endpoints and replication details in the dms-arn-list.txt file with the source endpoint ARN, target endpoint ARN, and the replication instance ARN by using following format:   

#sourceARN,targetARN,repARN
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Run the `dms-create-task.py` Python script to create the AWS DMS tasks. | 1. Run the `dms-create-task.py` Python script by using the following command: 

```bash
$ python dms-create-task.py database_migration.txt dms-arn-list.txt <cft-stack-name> <migration-type>
```

- `database_migration.txt` is the database migration text file
- `dms-arn-list.txt` is the ARN list for AWS DMS
- `<cft-stack-name>` is the user-defined AWS CloudFormation stack name
- `<migration-type>` is the migration type (full-load, cdc, or full-load-and-cdc)

2. Depending on your migration type, you can use the following commands to create three types of AWS DMS tasks:

```bash
$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack full-load
```

```
$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack cdc
```

```
$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack full-load-and-cdc
```

3. The AWS CloudFormation stack and AWS DMS tasks are created | DBA |
### AWS Prescriptive Guidance Patterns

Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL using AWS SCT and AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check that AWS DMS tasks are ready.</td>
<td>On the AWS console, check that your AWS DMS tasks are in Ready status in the Status section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Start and stop the AWS DMS tasks using AWS CLI

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Start the AWS DMS tasks. | Run the `dms-start-task.py` Python script by using the following command:  
  
  ```
  $ python dms-start-task.py start `<cdc-start-datetime>`
  ```
  
  **Note:** The start date and time must be in the 'DD-MON-YYYY' or 'YYYY-MM-DDTHH:MI:SS' timestamp data type formats (for example, '01-Dec-2019' or '2018-03-08T12:12:12')
  
  You can review the AWS DMS task status in the **Table statistics** tab of your migration tasks on the **Tasks** page of the AWS DMS console. | DBA |

| Validate the data. | 1. After the full-load migration is complete, the task is continuously kept running for continuous data change (CDC).  
  2. When CDC is complete or no more changes need to be migrated, review and validate the migration task results and data in your Oracle and PostgreSQL databases.  
  3. You can validate your data by checking status and count columns (Validation state, Validation pending, Validation failed, Validation suspended, and Validation details) in the **Table statistics** tab of your database migration task on the **Tasks** page of the AWS DMS console. | DBA |
AWS Prescriptive Guidance Patterns
Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL using AWS SCT and AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop the AWS DMS tasks.</td>
<td>Run the Python script by using the following command:</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>$ python dms-start-task.py stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: AWS DMS tasks might stop with a <em>failed</em> status, depending on the validation status. For more information about this, see the troubleshooting table in the <em>Additional information</em> section.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

- Installing AWS SCT
- Introduction to AWS DMS (video)
- Using the AWS CLI in AWS CloudFormation
- Using the AWS SCT user interface
- Using an Oracle database as a source for AWS DMS
- Using Oracle as a source for AWS SCT
- Using a PostgreSQL database as a target for AWS DMS
- Sources for data migration in AWS DMS
- Targets for data migration in AWS DMS
- cloudformation (AWS CLI documentation)
- cloudformation create-stack (AWS CLI documentation)
- dms (AWS CLI documentation)

Additional information

Troubleshooting

The following table provides guidance about troubleshooting this pattern.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SCT source and target test connections fail</td>
<td>Configure the JDBC driver versions and VPC security group inbound rules to accept the incoming traffic.</td>
</tr>
<tr>
<td>Source or target endpoint test run fails</td>
<td>Check if the endpoint settings and replication instance is in <em>Available</em> status. Check if the endpoint connection status is <em>Successful</em>.</td>
</tr>
</tbody>
</table>
Full-load run fails

Check if the source and target databases have matching data types and sizes.

For more information about this, see Troubleshooting migration tasks in AWS DMS in the AWS DMS documentation.

Validation run errors

Check if the table has a primary key because non-primary key tables are not validated.

If the table has a primary key and errors, check that the extra connection attribute in the source endpoint has `numberDataTypeScale=-2`.

For more information about this, see Extra connection attributes when using Oracle as a source for AWS DMS, OracleSettings, and Troubleshooting in the AWS DMS documentation.

---

### Attachments

attachment.zip

Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible Edition

*Created by anuradha chintha (AWS) and Rakesh Raghav (AWS)*

<table>
<thead>
<tr>
<th>Source: Databases: Relational</th>
<th>Target: Databases: Relational</th>
<th>R Type: Re-architect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: PoC or pilot</td>
<td>Technologies: Migration; Databases</td>
<td>Workload: Open-source</td>
</tr>
</tbody>
</table>

AWS services: Amazon S3

### Summary

External tables give Oracle the ability to query data that is stored outside the database in flat files. You can use the ORACLE_LOADER driver to access any data stored in any format that can be loaded by the SQL*Loader utility. You can't use Data Manipulation Language (DML) on external tables, but you can use external tables for query, join, and sort operations.
Amazon Aurora PostgreSQL-Compatible Edition doesn't provide functionality similar to external tables in Oracle. Instead, you must use modernization to develop a scalable solution that meets functional requirements and is frugal.

This pattern provides steps for migrating different types of Oracle external tables to Aurora PostgreSQL-Compatible Edition on the Amazon Web Services (AWS) Cloud by using the `aws_s3` extension.

**Prerequisites and limitations**

**Prerequisites**
- An active AWS account
- AWS Command Line Interface (AWS CLI)
- An available Aurora PostgreSQL-Compatible database instance
- An on-premises Oracle database with an external table
- pg.Client API
- Data files

**Limitations**
- This pattern doesn't provide the functionality to act as a replacement for Oracle external tables. However, the steps and sample code can be enhanced further to achieve your database modernization goals.
- Files should not contain the character that is passing as a delimiter in `aws_s3` export and import functions.

**Architecture**

**Source technology stack**
Oracle

**Source Architecture**

**Target technology stack**
- Amazon Aurora PostgreSQL-Compatible
- Amazon CloudWatch
- AWS Lambda
- AWS Secrets Manager
• Amazon Simple Notification Service (Amazon SNS)
• Amazon Simple Storage Service (Amazon S3)

Target architecture

The following diagram shows a high-level representation of the solution.

Automation and scale

Any additions or changes to the external tables can be handled with metadata maintenance.

Tools

• Amazon Aurora PostgreSQL-Compatible – Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine that combines the speed and reliability of high-end commercial databases with the cost-effectiveness of open-source databases.
• AWS CLI – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With only one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
• Amazon CloudWatch – Amazon CloudWatch monitors Amazon S3 resources and utilization.
• AWS Lambda – AWS Lambda is a serverless compute service that supports running code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes. In this pattern, Lambda runs the database function whenever a file is uploaded to Amazon S3.
• AWS Secrets Manager – AWS Secrets Manager is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
• Amazon S3 – Amazon Simple Storage Service (Amazon S3) provides a storage layer to receive and store files for consumption and transmission to and from the Aurora PostgreSQL-Compatible cluster.
• aws_s3 – The aws_s3 extension integrates Amazon S3 and Aurora PostgreSQL-Compatible.
• Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients. In this pattern, Amazon SNS is used to send notifications.
Code

Whenever a file is placed in the S3 bucket, a DB function must be created and called from the processing application or the Lambda function. For details, see the code (attached).

Epics

Create an external file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add an external file to the source database.</td>
<td>Create an external file, and move it to the oracle directory.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Configure the target (Aurora PostgreSQL-Compatible)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Aurora PostgreSQL database.</td>
<td>Create a DB instance in your Amazon Aurora PostgreSQL-Compatible cluster.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create a schema and tables.</td>
<td>The tables include actual tables, staging tables, error and log tables, and a metatable.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Create the DB function.</td>
<td>To create the DB function, use the load_external_table_latest.sql file (attached).</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

Create and configure the Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a role.</td>
<td>Create a role with permissions to access Amazon S3 and Amazon Relational Database Service (Amazon RDS). This role will be assigned to Lambda for running the pattern.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create the Lambda function.</td>
<td>To create a Lambda function, use the attached code and the role that you created.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure an S3 bucket event trigger.</td>
<td>Configure a mechanism to call the Lambda function for all object creation events in the S3 bucket.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create a secret.</td>
<td>Create a secret name for the database credentials using Secrets Manager. Pass the secret in the Lambda function.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Add integration with Amazon S3

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket.</td>
<td>On the Amazon S3 console, create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create IAM policies.</td>
<td>To create the AWS Identity and Access Management (IAM) policies, use s3bucketpolicy_for_import.json (attached).</td>
<td>DBA</td>
</tr>
<tr>
<td>Create roles.</td>
<td>Create two roles for Aurora PostgreSQL-Compatible, one role for Import and one role for Export. Assign the corresponding policies to the roles.</td>
<td>DBA</td>
</tr>
<tr>
<td>Attach the roles to the Aurora PostgreSQL-Compatible cluster.</td>
<td>Under Manage roles, attach the Import and Export roles to the Aurora PostgreSQL cluster.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create supporting objects for Aurora PostgreSQL-Compatible.</td>
<td>Use ext_tbl_scripts.sql (attached) for table scripts. Use load_external_table_latest.sql (attached) for the custom function.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Process a test file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload a file into the S3 bucket.</td>
<td>To upload a test file into the S3 bucket, use the console or the following command in AWS CLI.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

```bash
aws s3 cp /Users/Desktop/ukpost/xtt1t/*testing
```
Task | Description | Skills required
--- | --- | ---
files"/aps s3://s3importtest/inputext/aps | As soon as the file is uploaded, a bucket event initiates the Lambda function, which runs the Aurora PostgreSQL-Compatible function. | 
Check the data and the log and error files. | The Aurora PostgreSQL-Compatible function loads the files into the main table, and it creates .log and .bad files in the S3 bucket. | DBA
Monitor the solution. | In the Amazon CloudWatch console, monitor the Lambda function. | DBA

**Related resources**
- Amazon S3 integration
- Amazon S3
- Working with Amazon Aurora PostgreSQL-Compatible Edition
- AWS Lambda
- Amazon CloudWatch
- AWS Secrets Manager
- Setting up Amazon SNS notifications

**Attachments**

attachment.zip

**Migrate Oracle native functions to PostgreSQL using extensions**

*Created by Pinesh Singal (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon RDS PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Re-architect</td>
<td>Workload:</td>
<td>Oracle; Open-source</td>
<td>Technologies:</td>
<td>Migration; Databases</td>
</tr>
</tbody>
</table>

**AWS services:**
Amazon EC2; Amazon RDS
Summary

This migration pattern provides step-by-step guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle database instance to an Amazon RDS for PostgreSQL database by modifying the `aws_oracle_ext` and `orafce` extensions to PostgreSQL (psql) native built-in code. This will save processing time.

The pattern describes an offline manual migration strategy with no downtime for a multi-terabyte Oracle source database with a high number of transactions.

The migration process uses AWS Schema Conversion Tool (AWS SCT) with the `aws_oracle_ext` and `orafce` extensions to convert an Amazon RDS for Oracle database schema to an Amazon RDS for PostgreSQL database schema. Then the code is manually changed to PostgreSQL supported native `psql` built-in code. This is because the extension calls impact code processing on the PostgreSQL database server, and not all the extension code is fully complaint or compatible with PostgreSQL code.

This pattern primarily focuses on manually migrating SQL codes using AWS SCT and the extensions `aws_oracle_ext` and `orafce`. You convert the already used extensions into native PostgreSQL (psql) built-ins. Then you remove all references to the extensions and convert the codes accordingly.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Operating system (Windows or Mac) or Amazon EC2 instance (up and running)
- Orafce

Limitations

Not all Oracle functions using `aws_oracle_ext` or `orafce` extensions can be converted to native PostgreSQL functions. It might need manual rework so as to compile it with PostgreSQL libraries.

One drawback of using AWS SCT extensions is its slow performance in running and fetching the results. Its cost can be understood from simple PostgreSQL EXPLAIN plan (execution plan of a statement) on the Oracle `SYSDATE` function migration to the PostgreSQL `NOW()` function between all three codes (`aws_oracle_ext`, `orafce`, and `psql` default), as explained in the Performance comparison check section in the attached document.

Product versions

- **Source**: Amazon RDS for Oracle database 10.2 and later (for 10.x), 11g (11.2.0.3.v1 and later) and up to 12.2, and 18c for the Enterprise, Standard, Standard One, and Standard Two editions
- **Target**: Amazon RDS for PostgreSQL database 9.4 and later (for 9.x), 10.x, and 11.x
- **AWS SCT**: Latest version (this pattern was tested with 1.0.632)
- **Orafce**: Latest version (this pattern was tested with 3.9.0)

Architecture

Source technology stack

- An Amazon RDS for Oracle database instance with version 12.1.0.2.v18
**Target technology stack**

- An Amazon RDS for PostgreSQL database instance with version 11.5

**Database migration architecture**

The following diagram represents the database migration architecture between the source Oracle and target PostgreSQL databases. The architecture involves AWS Cloud, a virtual private cloud (VPC), Availability Zones, a private subnet, an Amazon RDS for Oracle database, AWS SCT, an Amazon RDS for PostgreSQL database, extensions for Oracle (`aws_oracle_ext` and `orafce`), and structured query language (SQL) files.

1. Launch Amazon RDS for Oracle DB instance (source DB).
2. Use AWS SCT with the `aws_oracle_ext` and `orafce` extension packs to convert the source code from Oracle to PostgreSQL.
3. The conversion produces PostgreSQL-supported migrated .sql files.
4. Manually convert the non-converted Oracle extension codes to PostgreSQL (`psql`) codes.
5. The manual conversion produces PostgreSQL-supported converted .sql files.
6. Run these .sql files on your Amazon RDS for PostgreSQL DB instance (target DB).

**Tools**

**Tools**

*AWS services*

- **AWS SCT** - AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another. You can convert relational Online Transactional Processing (OLTP) schema, or data warehouse schema. Your converted schema is suitable for an Amazon RDS MySQL DB instance, an Amazon Aurora DB cluster, an Amazon RDS PostgreSQL DB instance, or an Amazon Redshift cluster. The converted schema can also be used with a database on an Amazon EC2 instance or stored as data on an Amazon S3 bucket. AWS SCT provides a project-based user interface to automatically convert the database schema of your source database into a format compatible with your target Amazon RDS instance. You can use AWS SCT to do migration from an Oracle source database to any of the targets listed preceding. Using AWS SCT, you can export the source database object definitions such as schema, views, stored procedures, and functions. You can use AWS SCT to convert data from Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition. In this pattern, you use AWS SCT to convert and migrate Oracle code into PostgreSQL using the extensions `aws_oracle_ext` and `orafce`, and manually migrating the extension codes into `psql` default or native built-in code.
• The AWS SCT extension pack is an add-on module that emulates functions present in the source database that are required when converting objects to the target database. Before you can install the AWS SCT extension pack, you need to convert your database schema. When you convert your database or data warehouse schema, AWS SCT adds an additional schema to your target database. This schema implements SQL system functions of the source database that are required when writing your converted schema to your target database. This additional schema is called the extension pack schema. The extension pack schema for OLTP databases is named according to the source database. For Oracle databases, the extension pack schema is AWS_ORACLE_EXT.

Other tools
• Orafce – Orafce is a module that implements Oracle compatible functions, data types, and packages. It's an open-source tool with a Berkeley Source Distribution (BSD) license so that anyone can use it. The orafce module is useful for migrating from Oracle to PostgreSQL because it has many Oracle functions implemented in PostgreSQL.

Code
For a list of all commonly used and migrated code from Oracle to PostgreSQL to avoid AWS SCT extension code usage, see the attached document.

Epics
Configure the Amazon RDS for Oracle source database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Oracle database instance.</td>
<td>Create an Amazon RDS for Oracle database instance from the Amazon RDS console.</td>
<td>General AWS, DBA</td>
</tr>
<tr>
<td>Configure the security groups.</td>
<td>Configure inbound and outbound security groups.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create the database.</td>
<td>Create the Oracle database with needed users and schemas.</td>
<td>General AWS, DBA</td>
</tr>
<tr>
<td>Create the objects.</td>
<td>Create objects and insert data in schema.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Configure the Amazon RDS for PostgreSQL target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the PostgreSQL database instance.</td>
<td>Create an Amazon RDS for PostgreSQL database instance from the Amazon RDS console.</td>
<td>General AWS, DBA</td>
</tr>
<tr>
<td>Configure the security groups.</td>
<td>Configure inbound and outbound security groups.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create the database.</td>
<td>Create the PostgreSQL database with needed users and schemas.</td>
<td>General AWS, DBA</td>
</tr>
</tbody>
</table>
### Validate the extensions.

**Make sure that** `aws_oracle_ext` and `orafce` are installed and configured correctly in the PostgreSQL database.

**Skills required:** DBA

### Verify that the PostgreSQL database is available.

**Make sure that the PostgreSQL database is up and running.**

**Skills required:** DBA

---

### Migrate the Oracle schema into PostgreSQL using AWS SCT and the extensions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install AWS SCT.</td>
<td>Install the latest version of AWS SCT.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure AWS SCT.</td>
<td>Configure AWS SCT with Java Database Connectivity (JDBC) drivers for Oracle (<code>ojdbc8.jar</code>) and PostgreSQL (<code>postgresql-42.2.5.jar</code>).</td>
<td>DBA</td>
</tr>
<tr>
<td>Enable the AWS SCT extension pack or template.</td>
<td>Under AWS SCT <strong>Project Settings</strong>, enable built-in function implementation with the <code>aws_oracle_ext</code> and <code>orafce</code> extensions for the Oracle database schema.</td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the schema.</td>
<td>In AWS SCT, choose <strong>Convert Schema</strong> to convert the schema from Oracle to PostgreSQL and generate the <code>.sql</code> files.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

---

### Convert AWS SCT extension code to `psql` code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manually convert the code.</td>
<td>Manually convert each line of extension-supported code into <code>psql</code> default built-in code, as detailed in the attached document. For example, change <code>AWS_ORACLE_EXT.SYSDATE()</code> or <code>ORACLE.SYSDATE()</code> to <code>NOW()</code>.</td>
<td>DBA</td>
</tr>
<tr>
<td>Validate the code</td>
<td>(Optional) Validate each line of code by temporary running it in the PostgreSQL database.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create objects in the PostgreSQL database.</td>
<td>To create objects in the PostgreSQL database, run</td>
<td>DBA</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
---|---|---
| | the .sql files that were generated by AWS SCT and modified in the previous two steps. | |

**Related resources**

- Database
  - Oracle on Amazon RDS
  - PostgreSQL on Amazon RDS
  - PostgreSQL EXPLAIN plan
- AWS SCT
  - AWS Schema Conversion Tool Overview
  - AWS SCT User Guide
  - Using the AWS SCT user interface
  - Using Oracle Database as a source for AWS SCT
- Extensions for AWS SCT
  - Using the AWS SCT extension pack
  - Oracle functionality (en)
  - PGXN orafce
  - GitHub orafce

**Additional information**

For more information, follow the detailed commands, with syntax and examples, for manually converting code in the attached document.

**Attachments**

attachment.zip

**Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS**

*Created by Uma Maheswara Rao Nooka (AWS)*

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>R Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server on Amazon EC2</td>
<td>Amazon DocumentDB</td>
<td>Re-architect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment</th>
<th>Technologies</th>
<th>Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoC or pilot</td>
<td>Cloud-native; Databases; Migration</td>
<td>Microsoft</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon EC2; Amazon DocumentDB
Summary

This pattern describes how to use AWS Database Migration Service (AWS DMS) to migrate a Microsoft SQL Server database hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon DocumentDB (with MongoDB compatibility) database.

The AWS DMS replication task reads the table structure of the SQL Server database, creates the corresponding collection in Amazon DocumentDB, and performs a full-load migration.

You can also use this pattern to migrate an on-premises SQL Server or an Amazon Relational Database Service (Amazon RDS) for SQL Server DB instance to Amazon DocumentDB. For more information, see the guide Migrating Microsoft SQL Server databases to the AWS Cloud on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing SQL Server database on an EC2 instance.
- Fixed database (db_owner) role assigned to AWS DMS in the SQL Server database. For more information, see Database-level roles in the SQL Server documentation.
- Familiarity with using the `mongodump`, `mongorestore`, `mongoexport`, and `mongoimport` utilities to move data in and out of an Amazon DocumentDB cluster.
- Microsoft SQL Server Management Studio, installed and configured.

Limitations

- The cluster size limit in Amazon DocumentDB is 64 TB. For more information, see Cluster limits in the Amazon DocumentDB documentation.
- AWS DMS doesn't support the merging of multiple source tables into a single Amazon DocumentDB collection.
- If AWS DMS processes any changes from a source table without a primary key, it will ignore large object (LOB) columns in the source table.

Architecture

Source technology stack

- Amazon EC2

Target architecture
Target technology stack

- Amazon DocumentDB

Tools

- AWS DMS – AWS Database Migration Service (AWS DMS) helps you migrate databases easily and securely.
- Amazon DocumentDB – Amazon DocumentDB (with MongoDB compatibility) is a fast, reliable, and fully managed database service.
- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.
- Microsoft SQL Server – SQL Server is a relational database management system.
- SQL Server Management Studio (SSMS) – SSMS is a tool for managing SQL Server, including accessing, configuring, and administering SQL Server components.

Epics

Create and configure a VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC.</td>
<td>Sign in to the AWS Management Console and open the Amazon VPC console. Create a virtual private cloud (VPC) with an IPv4 CIDR block range.</td>
<td>System administrator</td>
</tr>
<tr>
<td>Create security groups and network ACLs.</td>
<td>On the Amazon VPC console, create security groups and</td>
<td>System administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>network access control lists (network ACLs) for your VPC, according to your requirements. You can also use the default settings for these configurations. For more information about this and other stories, see the “Related resources” section.</td>
<td></td>
</tr>
</tbody>
</table>

Create and configure the Amazon DocumentDB cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create an Amazon DocumentDB cluster.</td>
<td>System administrator</td>
</tr>
<tr>
<td></td>
<td>Open the Amazon DocumentDB console and choose “Clusters.” Choose “Create,” and create an Amazon DocumentDB cluster with one instance. Important: Make sure you configure this cluster with your VPC’s security groups.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Install the mongo shell.</td>
<td>System administrator</td>
</tr>
<tr>
<td></td>
<td>The mongo shell is a command-line utility that you use to connect to and query your Amazon DocumentDB cluster. To install it, run the “/etc/yum.repos.d/mongodb-org-3.6.repo” command to create the repository file. Run the “sudo yum install -y mongoddb-org-shell” command to install the mongo shell. To encrypt data in transit, download the public key for Amazon DocumentDB, and then connect to your Amazon DocumentDB instance. For more information about these steps, see the “Related resources” section.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create a database in the Amazon DocumentDB cluster.</td>
<td>System administrator</td>
</tr>
<tr>
<td></td>
<td>Run the “use” command with the name of your database to create a database in your Amazon DocumentDB cluster.</td>
<td></td>
</tr>
</tbody>
</table>

Create and configure the AWS DMS replication instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS DMS replication instance.</td>
<td>Open the AWS DMS console and choose “Create replication”</td>
<td>System administrator</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate a SQL Server database from Amazon EC2 to Amazon DocumentDB**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure the SQL Server database.</strong></td>
<td>Log in to Microsoft SQL Server and add an inbound rule for communication between the source endpoint and the AWS DMS replication instance. Use the replication instance's private IP address as the source. Important: The replication instance and target endpoint should be on the same VPC. Use an alternative source in the security group if the VPCs are different for the source and replication instances.</td>
<td>System administrator</td>
</tr>
</tbody>
</table>

**Create and test the source and target endpoints in AWS DMS**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create the source and target database endpoints.</strong></td>
<td>Open the AWS DMS console and choose “Connect source and target database endpoints.” Specify the connection information for the source and target databases. If required, choose the “Advanced” tab to set values for “Extra connection attributes.” Download and use the certificate bundle in your endpoint configuration.</td>
<td>System administrator</td>
</tr>
<tr>
<td><strong>Test the endpoint connection.</strong></td>
<td>Choose “Run test” to test the connection. Troubleshoot any error messages by verifying the security group settings and the connections to the AWS DMS replication instance from both the source and target database instances.</td>
<td>System administrator</td>
</tr>
</tbody>
</table>
Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS DMS migration task.</td>
<td>On the AWS DMS console, choose “Tasks,” “Create task.” Specify the task options, including the source and destination endpoint names, and replication instance names. Under “Migration type” choose “Migrate existing data,” and “Replicate data changes only.” Choose “Start task.”</td>
<td>System administrator</td>
</tr>
<tr>
<td>Run the AWS DMS migration task.</td>
<td>Under “Task settings,” specify the settings for the table preparation mode, such as “Do nothing,” “Drop tables on target,” “Truncate,” and “Include LOB columns in replication.” Set a maximum LOB size that AWS DMS will accept and choose “Enable logging.” Leave the “Advanced settings” at their default values and choose “Create task.”</td>
<td>System administrator</td>
</tr>
<tr>
<td>Monitor the migration.</td>
<td>On the AWS DMS console, choose “Tasks” and choose your migration task. Choose “Task monitoring” to monitor your task. The task stops when the full-load migration is complete and cached changes are applied.</td>
<td>System administrator</td>
</tr>
</tbody>
</table>

Test and verify the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to the Amazon DocumentDB cluster by using the mongo shell.</td>
<td>Open the Amazon DocumentDB console, choose your cluster under “Clusters.” In the “Connectivity and Security” tab, choose “Connect to this cluster with the mongo shell.”</td>
<td>System administrator</td>
</tr>
<tr>
<td>Verify the results of your migration.</td>
<td>Run the “use” command with the name of your database and then run the “show collections” command. Run the “db. .count();” command with the name of your database. If the results match your source</td>
<td>System administrator</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Migrate a ThoughtSpot Falcon
database to Amazon Redshift

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>database, then your migration was successful.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

Create and configure a VPC
- Create a security group for your VPC
- Create a network ACL

Create and configure the Amazon DocumentDB cluster
- Create an Amazon DocumentDB cluster
- Install the mongo shell for Amazon DocumentDB
- Connect to your Amazon DocumentDB cluster

Create and configure the AWS DMS replication instance
- Use public and private replication instances

Create and test the source and target endpoints in AWS DMS
- Use Amazon DocumentDB as a target for AWS DMS
- Use a SQL Server database as a source for AWS DMS
- Use AWS DMS endpoints

Migrate data
- Migrate to Amazon DocumentDB

Other resources
- Limitations on using SQL Server as a source for AWS DMS
- How to use Amazon DocumentDB to build and manage applications at scale

Migrate an on-premises ThoughtSpot Falcon database to Amazon Redshift
Summary

On-premises data warehouses require significant administration time and resources, particularly for large datasets. The financial cost of building, maintaining, and growing these warehouses is also very high. To help manage costs, keep extract, transform, and load (ETL) complexity low, and deliver performance as your data grows, you must constantly choose which data to load and which data to archive.

By migrating your on-premises ThoughtSpot Falcon databases to the Amazon Web Services (AWS) Cloud, you can access cloud-based data lakes and data warehouses that increase your business agility, security, and application reliability, in addition to reducing your overall infrastructure costs. Amazon Redshift helps to significantly lower the cost and operational overhead of a data warehouse. You can also use Amazon Redshift Spectrum to analyze large amounts of data in its native format without data loading.

This pattern describes the steps and process for migrating a ThoughtSpot Falcon database from an on-premises data center to an Amazon Redshift database on the AWS Cloud.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A ThoughtSpot Falcon database hosted in an on-premises data center

Product versions

- ThoughtSpot version 7.0.1
## Architecture

The diagram shows the following workflow:

1. Data is hosted in an on-premises relational database.
2. AWS Schema Conversion Tool (AWS SCT) converts the data definition language (DDL) that is compatible with Amazon Redshift.
3. After the tables are created, you can migrate the data by using AWS Database Migration Service (AWS DMS).
4. The data is loaded into Amazon Redshift.
5. The data is stored in Amazon Simple Storage Service (Amazon S3) if you use Redshift Spectrum or already host the data in Amazon S3.

## Tools

- **AWS DMS** – AWS Data Migration Service (AWS DMS) helps you quickly and securely migrate databases to AWS.
- **Amazon Redshift** – Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools.
- **AWS SCT** – AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another.

## Epics

**Prepare for the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the appropriate Amazon Redshift configuration.</td>
<td>Identify the appropriate Amazon Redshift cluster configuration based on your requirements and data volume.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
## Migrate a ThoughtSpot Falcon database to Amazon Redshift

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Research Amazon Redshift to evaluate if it meets your requirements.</strong></td>
<td>Use the Amazon Redshift FAQs to understand and evaluate whether Amazon Redshift meets your requirements.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Prepare the target Amazon Redshift cluster</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an Amazon Redshift cluster.</strong></td>
<td>Sign in to the AWS Management Console, open the Amazon Redshift console, and then create an Amazon Redshift cluster in a virtual private cloud (VPC). For more information about this, see Creating a cluster in a VPC in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Conduct a PoC for your Amazon Redshift database design.</strong></td>
<td>Follow Amazon Redshift best practices by conducting a proof of concept (PoC) for your database design. For more information about this, see Conducting a proof of concept for Amazon Redshift in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Create database users.</strong></td>
<td>Create the users in your Amazon Redshift database and grant the appropriate roles for access to the schema and tables. For more information about this, see Grant access privileges for a user or user group in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Apply configuration settings to the target database.</strong></td>
<td>Apply configuration settings to the Amazon Redshift database according to your requirements. For more information about enabling database, session, and server-level parameters, see</td>
<td>DBA</td>
</tr>
</tbody>
</table>
## Migrate a ThoughtSpot Falcon database to Amazon Redshift

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create objects in the Amazon Redshift cluster</td>
<td><strong>Task</strong> Create objects in the Amazon Redshift cluster. <strong>Description</strong> Open the <strong>Configuration reference</strong> in the Amazon Redshift documentation. <strong>Skills required</strong> DBA.</td>
<td></td>
</tr>
<tr>
<td>Manually create tables with DDL in Amazon Redshift.</td>
<td><strong>Task</strong> Manually create tables with DDL in Amazon Redshift. <strong>Description</strong> (Optional) If you use AWS SCT, the tables are automatically created. However, if there are failures when replicating DDLs, you have to manually create the tables. <strong>Skills required</strong> DBA.</td>
<td></td>
</tr>
<tr>
<td>Create external tables for Redshift Spectrum.</td>
<td><strong>Task</strong> Create external tables for Redshift Spectrum. <strong>Description</strong> Create an external table with an external schema for Amazon Redshift Spectrum. To create external tables, you must be the owner of the external schema or a database superuser. For more information about this, see Creating external tables for Amazon Redshift Spectrum in the Amazon Redshift documentation. <strong>Skills required</strong> DBA.</td>
<td></td>
</tr>
<tr>
<td>Migrate data using AWS DMS</td>
<td><strong>Task</strong> Migrate data using AWS DMS. <strong>Description</strong> Use AWS DMS to migrate the data. After you create the DDL of the tables in the Amazon Redshift database, migrate your data to Amazon Redshift using AWS DMS. For detailed steps and instructions about this, see Using an Amazon Redshift database as a target for AWS DMS in the AWS DMS documentation. <strong>Skills required</strong> DBA.</td>
<td></td>
</tr>
<tr>
<td>Use the COPY command to load the data.</td>
<td>Use the Amazon Redshift COPY command to load the data from Amazon S3 to Amazon Redshift. For more information about this, see Using the COPY command to load from Amazon. <strong>Skills required</strong> DBA.</td>
<td></td>
</tr>
</tbody>
</table>

826
**AWS Prescriptive Guidance Patterns**

**Migrate a ThoughtSpot Falcon database to Amazon Redshift**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3 in the Amazon Redshift documentation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Validate the Amazon Redshift cluster**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target records.</td>
<td>Validate the table count for the source and target records that were loaded from your source system.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
| Implement Amazon Redshift best practices for performance tuning. | Implement Amazon Redshift best practices for table and database design.  
For more information about this, see [Top 10 performance tuning techniques for Amazon Redshift](https://aws.amazon.com/blogs/big-data/) from the AWS Big Data Blog. | DBA |
| Optimize query performance. | Amazon Redshift uses SQL-based queries to interact with data and objects in the system.  
Data manipulation language (DML) is the subset of SQL that you can use to view, add, change, and delete data.  
Data definition language (DDL) is the subset of SQL that you use to add, change, and delete database objects such as tables and views.  
For more information about this, see [Tuning query performance](https://aws.amazon.com/blogs/machine-learning/) in the Amazon Redshift documentation. | DBA |
| Implement WLM. | You can use workload management (WLM) to define multiple query queues and route queries to appropriate queues at runtime.  
For more information about this, see [Implementing workload management](https://aws.amazon.com/blogs/machine-learning/) in the Amazon Redshift documentation. | DBA |
<p>| Work with concurrency scaling. | By using the Concurrency Scaling feature, you can support virtually unlimited concurrent users and concurrent queries, with consistently fast query performance. | DBA |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Amazon Redshift best practices for table design.</td>
<td>When you plan your database, certain important table design decisions can strongly influence overall query performance. For more information about choosing the most appropriate table design option, see Amazon Redshift best practices for designing tables in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create materialized views in Amazon Redshift.</td>
<td>A materialized view contains a precomputed results set based on an SQL query over one or more base tables. You can issue SELECT statements to query a materialized view in the same way that you query other tables or views in the database. For more information about this, see Creating materialized views in Amazon Redshift in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Define joins between the tables.</td>
<td>To search more than one table at the same time in ThoughtSpot, you must define joins between the tables by specifying columns that contain matching data across two tables. These columns represent the primary key and foreign key of the join. You can define them by using the ALTER TABLE command in Amazon Redshift or ThoughtSpot. For more information about this, see ALTER TABLE in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Add an Amazon Redshift connection.</td>
<td>Add an Amazon Redshift connection to your on-premises ThoughtSpot Falcon database. For more information about this, see Add an Amazon Redshift connection in the ThoughtSpot documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Edit the Amazon Redshift connection.</td>
<td>You can edit the Amazon Redshift connection to add tables and columns. For more information about this, see Edit an Amazon Redshift connection in the ThoughtSpot documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Remap the Amazon Redshift connection.</td>
<td>Modify the connection parameters by editing the source mapping .yaml file that was created when you added the Amazon Redshift connection. For example, you can remap the existing table or column to a different table or column in an existing database connection. ThoughtSpot recommends that you check the dependencies before and after you remap a table or column in a connection to ensure that they display as required. For more information about this, see Remap an Amazon Redshift connection in the ThoughtSpot documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Delete a table from the Amazon Redshift connection.</td>
<td>(Optional) If you attempt to remove a table in an Amazon Redshift connection, ThoughtSpot checks for dependencies and shows a list of dependent objects. You can choose the listed objects to delete them or remove the dependency. You can then remove the table. For more information about this, see Delete a table from an Redshift connection.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Task Description Skills required

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete a table with dependent objects from an Amazon Redshift connection.</td>
<td>(Optional) If you try to delete a table with dependent objects, the operation is blocked. A <strong>Cannot delete window</strong> appears, with a list of links to dependent objects. When all the dependencies are removed, you can then delete the table. For more information about this, see <strong>Delete a table with dependent objects from an Amazon Redshift connection</strong> in the ThoughSpot documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Delete an Amazon Redshift connection.</td>
<td>(Optional) Because a connection can be used in multiple data sources or visualizations, you must delete all of the sources and tasks that use that connection before you can delete the Amazon Redshift connection. For more information about this, see <strong>Delete an Amazon Redshift connection</strong> in the ThoughSpot documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Check connection reference for Amazon Redshift.</td>
<td>Make sure that you provide the required information for your Amazon Redshift connection by using the <strong>Connection reference</strong> from the ThoughSpot documentation.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Additional information
- **AI-driven analytics at any scale with ThoughtSpot and Amazon Redshift**
- **Amazon Redshift pricing**
- **Getting started with AWS SCT**
- **Getting started with Amazon Redshift**
- **Using data extraction agents**
- **Chick-fil-A improves speed to insight with ThoughtSpot and AWS**

### Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL
Summary

This pattern provides guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to an Amazon RDS for MySQL DB instance by using AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT).

The pattern covers best practices for handling the migration of stored procedures and code changes to support the application layer.

Additionally, this pattern uses AWS Workload Qualification Framework (AWS WQF) to estimate efforts required to carry out this migration. AWS WQF classifies this migration as workload category 1.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Amazon RDS for Oracle source database.
- An Amazon RDS for MySQL target database. Source and target databases should be in the same virtual private cloud (VPC), or you must have the required access permissions if you're using multiple VPCs.
- Security groups that allow connectivity between the source and target databases, AWS SCT, the application server, and AWS DMS.
- A user account with the required privilege to run AWS SCT on the source database.
- Supplemental logging enabled, to run AWS DMS on the source database.

Limitations

- The source and target Amazon RDS database size limit is 64 TB.
- Oracle is case-insensitive for database objects, but MySQL is not. AWS SCT can handle this issue while creating an object; however, some manual work is required to support full case insensitivity.
- This migration doesn't use MySQL extensions to enable Oracle-native functions. AWS SCT handles most of the conversion, but some work is required to change code manually.
- Java Database Connectivity (JDBC) driver changes are required in the application.

Product versions

- Amazon RDS for Oracle 11.2.0.4.v20
- Amazon RDS for MySQL 8.0.15
- AWS DMS version 3.3.0
- AWS SCT version 1.0.628
Architecture

Source technology stack

- Amazon RDS for Oracle. For more information, see Using an Oracle Database as a Source for AWS DMS in the AWS DMS documentation.

Target technology stack

- Amazon RDS for MySQL. For more information, see Using a MySQL-Compatible Database as a Target for AWS DMS in the AWS DMS documentation.

Migration architecture

Tools

- **AWS Schema Conversion Tool** – AWS SCT makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database.

- **AWS Data Migration Service** – AWS DMS can migrate your data to and from most widely used commercial and open-source databases. AWS DMS supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora.
AWS Prescriptive Guidance Patterns
Migrate from Amazon RDS for Oracle to MySQL

- **AWS Workload Qualification Framework** – AWS WQF uses AWS SCT to collect information to model existing Oracle and Microsoft SQL Server database workloads and provides instructions to convert them to an AWS database.

**Epics**

**Prepare for migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions and engines.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Use AWS WQF to estimate efforts required for this migration.</td>
<td></td>
<td>DBA, Sysadmin</td>
</tr>
<tr>
<td>Identify hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity, storage features, network features).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network-access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose an application migration strategy.</td>
<td>Consider whether you want full downtime or partial downtime for cutover activities.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Configure infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC and subnets.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups and network access control lists (ACLs).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon RDS for Oracle instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon RDS for MySQL instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Prepare a test case for validation of code conversion.</td>
<td>This will help in unit-testing for the converted code.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Configure the AWS DMS instance.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

833
## Migrate from Amazon RDS for Oracle to MySQL

### Configure source and target endpoints in AWS DMS.

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate the target database script</td>
<td>Check the accuracy of the code that was converted by AWS SCT. Some manual</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td></td>
<td>work will be required.</td>
<td></td>
</tr>
<tr>
<td>In AWS SCT, choose the &quot;Case Insensitive&quot; setting.</td>
<td>In AWS SCT, choose Project Settings, Target Case Sensitivity, Case Insensitive.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>In AWS SCT, choose not to use the Oracle native function.</td>
<td>In Project Settings, check the functions TO_CHAR/TO_NUMBER/TO_DATE.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Make changes for &quot;sql %notfound&quot; code.</td>
<td>You might have to convert the code manually.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Query on tables and objects in stored</td>
<td></td>
<td></td>
</tr>
<tr>
<td>procedures (use lowercase queries).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create the primary script after all</td>
<td></td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>changes are made, and then deploy the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>primary script on the target database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit-test stored procedures and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application calls using sample data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clean up data that was created during</td>
<td></td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>unit testing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drop foreign key constraints on</td>
<td>This step is required to load initial data. If you don't want to drop the</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>the target database.</td>
<td>foreign key constraints, you must create a migration task for data specific</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to the primary and secondary tables.</td>
<td></td>
</tr>
<tr>
<td>Drop primary keys and unique keys on the</td>
<td>This step results in better performance for the initial load.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>target database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable supplemental logging on the</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>source database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a migration task for the initial</td>
<td>Choose the option to migrate existing data.</td>
<td>DBA</td>
</tr>
<tr>
<td>load in AWS DMS, and then run it.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Migrate from Amazon RDS for Oracle to MySQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add the primary keys and foreign keys to the target database.</td>
<td>Constraints need to be added after the initial load.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Create a migration task for ongoing replication.</td>
<td>Ongoing replication keeps the target database synchronized with the source database.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Migrate applications**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace Oracle native functions with MySQL native functions.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Make sure that only lowercase names are used for database objects in SQL queries.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Cut over to the target database**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the application server.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Validate that the source and target databases are in sync.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Stop the Amazon RDS for Oracle DB instance.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Stop the migration task.</td>
<td>This will stop automatically after you complete the previous step.</td>
<td>DBA</td>
</tr>
<tr>
<td>Change the JDBC connection from Oracle to MySQL.</td>
<td></td>
<td>App owner, DBA</td>
</tr>
<tr>
<td>Start the application.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Close the project**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Migrate from IBM Db2 to Aurora PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop and delete AWS DMS instances.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Remove the source and target endpoints.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Remove migration tasks.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Take a snapshot of the Amazon RDS for Oracle DB instance.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Delete the Amazon RDS for Oracle DB instance.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Shut down and delete any other temporary AWS resources you used.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Related resources

- AWS DMS
- AWS SCT
- AWS WQF
- Amazon RDS Pricing
- Getting Started with AWS DMS
- Getting Started with Amazon RDS

Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL using AWS DMS and AWS SCT

Created by Sirsendu Halder (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Aurora PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>IBM</td>
<td>AWS services:</td>
<td>Amazon Aurora</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating an IBM Db2 database on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon Aurora PostgreSQL-Compatible Edition DB instance. This pattern uses AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT) for data migration and schema conversion.
The pattern targets an online migration strategy with little or no downtime for a multi-terabyte IBM Db2 database that has a high number of transactions. We recommend that you convert the columns in primary keys (PKs) and foreign keys (FKs) with the data type NUMERIC to INT or BIGINT in PostgreSQL for better performance.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source IBM Db2 database on an EC2 instance

Product versions

- DB2/LINUXX8664 version 11.1.4.4 and later

Architecture

Source technology stack

- A Db2 database on an EC2 instance

Target technology stack

- An Aurora PostgreSQL-Compatible DB instance

Database migration architecture
Tools

- **AWS DMS** - *AWS Database Migration Service* (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. You can use AWS DMS to migrate your data to and from the most widely used commercial and open-source databases. AWS DMS supports heterogeneous migrations between different database platforms, such as IBM Db2 to Aurora PostgreSQL-Compatible. For details, see *Sources for Data Migration* and *Targets for Data Migration* in the AWS DMS documentation.

- **AWS SCT** - *AWS Schema Conversion Tool* (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format that's compatible with the target database. Any objects that are not automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan the application source code for embedded SQL statements and convert them.

Epics

Migrate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Aurora PostgreSQL-Compatible DB instance.</td>
<td>The Aurora PostgreSQL-Compatible DB instance should be in the same virtual private cloud (VPC) as the source IBM Db2 database.</td>
<td>Amazon RDS</td>
</tr>
</tbody>
</table>
## Task descriptions and related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a replication instance and endpoints for IBM Db2.</strong></td>
<td>You must create separate endpoints for the source and target databases. For Aurora PostgreSQL-Compatible, the port will be 5432, and you'll need to get the server name from the DB instance's endpoint. For the source database, the server name must be the public DNS of the EC2 instance where Db2 is running. For user name, you can use db2inst1 followed by the port, which will be 5000 for IBM Db2. You then create a replication instance. For the VPC security group, select both the EC2 instance for Db2 and the Aurora PostgreSQL-Compatible DB instance. This replication instance must be in the same VPC as the source and target DB instances.</td>
<td>AWS DMS</td>
</tr>
<tr>
<td><strong>Create an AWS DMS task to fully load the source IBM Db2 tables to the target PostgreSQL schema.</strong></td>
<td>For source and target, use the source and destination endpoint names. The type can be full load. For the schema rule, you can use the Db2 database's &quot;inst1&quot; schema. For the table name, specify &quot;%&quot; to migrate all tables. When the load is complete, you'll see the Db2 tables of the &quot;inst1&quot; schema appearing in the Aurora PostgreSQL-Compatible database.</td>
<td>AWS DMS</td>
</tr>
<tr>
<td><strong>Use AWS SCT to migrate all other objects.</strong></td>
<td>Configure AWS SCT to connect to the source and target DB instances. In AWS SCT, select the views, indexes, procedures, and functions from the &quot;inst1&quot; schema in the source Db2 database, to migrate them to the target database.</td>
<td>AWS SCT</td>
</tr>
</tbody>
</table>

### Related resources

**References**

- *Amazon Aurora Features: PostgreSQL-Compatible Edition*
- *PostgreSQL 9.5.18 documentation*
Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS

Created by Kumar Babu P G (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for PostgreSQL/Amazon Aurora PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Re-architect</td>
<td>Workload: Oracle</td>
<td>Technologies: Migration; Databases</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon RDS;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amazon Aurora</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an on-premises Oracle 8i or 9i database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL. AWS Database Migration Service (AWS DMS) doesn't support Oracle 8i or 9i as a source, so Quest SharePlex replicates data from an on-premises 8i or 9i database to an intermediate Oracle database (Oracle 10g or 11g), which is compatible with AWS DMS.

From the intermediate Oracle instance, the schema and data are migrated to the PostgreSQL database on AWS by using AWS Schema Conversion Tool (AWS SCT) and AWS DMS. This method helps achieve continuous streaming of data from the source Oracle database to the target PostgreSQL DB instance with minimum replication lag. In this implementation, the downtime is limited to the length of time it takes to create or validate all the foreign keys, triggers, and sequences on the target PostgreSQL database.

The migration uses an Amazon Elastic Compute Cloud (Amazon EC2) instance with Oracle 10g or 11g installed to host the changes from the source Oracle database. AWS DMS uses this intermediate Oracle instance as the source to stream the data to Amazon RDS for PostgreSQL or Aurora PostgreSQL. Data replication can be paused and resumed from the on-premises Oracle database to the intermediate Oracle instance. It can also be paused and resumed from the intermediate Oracle instance to the target PostgreSQL database so you can validate the data by using either AWS DMS data validation or a custom data validation tool.

Prerequisites and limitations

Prerequisites
• An active AWS account
• A source Oracle 8i or 9i database in an on-premises data center
• AWS Direct Connect configured between the on-premises data center and AWS
• Java Database Connectivity (JDBC) drivers for AWS SCT connectors installed either on a local machine or on the EC2 instance where AWS SCT is installed
• Familiarity with using an Oracle database as an AWS DMS source
• Familiarity with using a PostgreSQL database as an AWS DMS target
• Familiarity with Quest SharePlex data replication

Limitations
• The database size limit is 64 TB
• The on-premises Oracle database must be Enterprise Edition

Product versions
• Oracle 8i or 9i for the source database
• Oracle 10g or 11g for the intermediate database
• PostgreSQL 9.6 or later

Architecture

Source technology stack
• Oracle 8i or 9i database
• Quest SharePlex

Target technology stack
• Amazon RDS for PostgreSQL or Aurora PostgreSQL

Source and target architecture
Tools

- **AWS DMS** – AWS Database Migration Service (AWS DMS) helps you migrate databases quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.

- **AWS SCT** – AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database. Objects that cannot be automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project. During this process, AWS SCT performs cloud-native code optimization by converting legacy Oracle and SQL Server functions to their AWS equivalents, to help you modernize your applications while migrating your databases. When schema conversion is complete, AWS SCT can help migrate data from a range of data warehouses to Amazon Redshift by using built-in data migration agents.

- **Quest SharePlex** – Quest SharePlex is an Oracle-to-Oracle data replication tool for moving data with minimal downtime and no data loss.

Epics

Create the EC2 instance and install Oracle

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the network for Amazon EC2.</td>
<td>Create the virtual private cloud (VPC), subnets, internet gateway, route tables, and security groups.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create the new EC2 instance.</td>
<td>Select the Amazon Machine Image (AMI) for the EC2</td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate from Oracle 8i/9i to Amazon RDS for PostgreSQL using Quest SharePlex**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create instance.</strong></td>
<td>Choose the instance size and configure instance details: the number of instances (1), the VPC and subnet from the previous step, auto-assign public IP, and other options. Add storage, configure security groups, and launch the instance. When prompted, create and save a key pair for the next step.</td>
<td></td>
</tr>
<tr>
<td><strong>Install Oracle on the EC2 instance.</strong></td>
<td>Acquire the licenses and the required Oracle binaries, and install Oracle 10g or 11g on the EC2 instance.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Set up SharePlex on an EC2 instance and configure data replication</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Set up SharePlex.</strong></td>
<td>Create an Amazon EC2 instance and install the SharePlex binaries that are compatible with Oracle 8i or 9i.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td><strong>Configure data replication.</strong></td>
<td>Follow SharePlex best practices to configure data replication from an on-premises Oracle 8i/9i database to an Oracle 10g/11g instance.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Convert the Oracle database schema to PostgreSQL</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Set up AWS SCT.</strong></td>
<td>Create a new report, and then connect to Oracle as the source and PostgreSQL as the target. In project settings, open the SQL Scripting tab and change the target SQL script to Multiple Files.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Convert the Oracle database schema.</strong></td>
<td>In the Action tab, choose Generate Report, Convert Schema, and then Save as SQL.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Modify the SQL scripts generated by AWS SCT.</strong></td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>
## Create and configure the Amazon RDS DB instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Amazon RDS DB instance.</td>
<td>In the Amazon RDS console, create a new PostgreSQL DB instance.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure the DB instance.</td>
<td>Specify the DB engine version, DB instance class, Multi-AZ deployment, storage type, and allocated storage. Enter the DB instance identifier, a master user name, and a master password.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure network and security.</td>
<td>Specify the VPC, subnet group, public accessibility, Availability Zone preference, and security groups.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure database options.</td>
<td>Specify the database name, port, parameter group, encryption, and master key.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure backups.</td>
<td>Specify the backup retention period, backup window, start time, duration, and whether to copy tags to snapshots.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure monitoring options.</td>
<td>Enable or disable enhanced monitoring and performance insights.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure maintenance options.</td>
<td>Specify auto minor version upgrade, maintenance window, and the start day, time, and duration.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Run the pre-migration scripts from AWS SCT.</td>
<td>On the Amazon RDS instance, run these scripts: create_database.sql, create_sequence.sql, create_table.sql, create_view.sql, and create_function.sql.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
</tbody>
</table>

## Migrate data by using AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a replication instance in AWS DMS.</td>
<td>Complete the fields for the name, instance class, VPC (same as for the EC2 instance), Multi-AZ, and public accessibility. In the advanced configuration section, specify allocated storage, subnet group, Availability Zone, VPC</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Create the source database endpoint.</td>
<td>Specify the endpoint name, type, source engine (Oracle), server name (Amazon EC2 private DNS name), port, SSL mode, user name, password, SID, VPC (specify the VPC that has the replication instance), and replication instance. To test the connection, choose Run Test, and then create the endpoint. You can also configure the following advanced settings: maxFileSize and numberDataTypeScale.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the AWS DMS replication task.</td>
<td>Specify the task name, replication instance, source and target endpoints, and replication instance. For migration type, choose &quot;Migrate existing data and replicate ongoing changes.&quot; Clear the &quot;Start task on create&quot; check box.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure the AWS DMS replication task settings.</td>
<td>For target table preparation mode, choose &quot;Do nothing.&quot; Stop the task after the full load completes to create primary keys. Specify limited or full LOB mode, and enable control tables. Optionally, you can configure the CommitRate advanced setting.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure the table mappings.</td>
<td>In the table mappings section, create an Include rule for all tables in all schemas included in the migration, and then create an Exclude rule. Add three transformation rules to convert the schema, table, and column names to lowercase, and add any other rules needed for this specific migration.</td>
<td>DBA</td>
</tr>
<tr>
<td>Start the task.</td>
<td>Start the replication task. Make sure that the full load is running. Run ALTER SYSTEM SWITCH LOGFILE on the primary Oracle database to kick-start the task.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Task

#### Description

- **Run the mid-migration scripts from AWS SCT.**
  In Amazon RDS for PostgreSQL, run these scripts: `create_index.sql` and `create_constraint.sql.`

- **Restart the task to continue change data capture (CDC).**
  In the Amazon RDS for PostgreSQL DB instance, run VACUUM, and restart the AWS DMS task to apply the cached CDC changes.

#### Skills required

- **DBA**

### Cut over to the PostgreSQL database

#### Task

#### Description

- **Check the AWS DMS logs and metadata tables.**
  Validate any errors and fix if required.

- **Stop all Oracle dependencies.**
  Shut down listeners on the Oracle database and run `ALTER SYSTEM SWITCH LOGFILE`. Stop the AWS DMS task when it shows no activity.

- **Run the post-migration scripts from AWS SCT.**
  In Amazon RDS for PostgreSQL, run these scripts: `create_foreign_key_constraint.sql` and `create_triggers.sql.`

- **Complete any additional Amazon RDS for PostgreSQL steps.**
  Increment sequences to match Oracle if needed, run VACUUM and ANALYZE, and take a snapshot for compliance.

- **Open the connections to Amazon RDS for PostgreSQL.**
  Remove the AWS DMS security groups from Amazon RDS for PostgreSQL, add production security groups, and point your applications to the new database.

- **Clean up AWS DMS resources.**
  Remove the endpoints, replication tasks, replication instances, and the EC2 instance.

#### Skills required

- **DBA**
- **SysAdmin, DBA**

### Related resources

- AWS DMS documentation
- AWS SCT documentation
- Amazon RDS for PostgreSQL pricing
- Using an Oracle database as a source for AWS DMS
- Using a PostgreSQL database as a target for AWS DMS
Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS

Created by Kumar Babu P G (AWS)

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for PostgreSQL/Amazon Aurora PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Oracle</td>
<td>AWS services: Amazon RDS; Amazon Aurora</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes how to migrate an on-premises legacy Oracle 8i or 9i database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora for PostgreSQL. AWS Database Migration Service (AWS DMS) doesn't support Oracle 8i or 9i as a source, so this pattern uses an intermediate Oracle database instance that's compatible with AWS DMS, such as Oracle 10g or 11g. It also uses the materialized views feature to migrate data from the source Oracle 8i/9i instance to the intermediate Oracle 10g/11g instance.

AWS Schema Conversion Tool (AWS SCT) converts the database schema, and AWS DMS migrates the data to the target PostgreSQL database.

This pattern helps users who want to migrate from legacy Oracle databases with minimum database downtime. In this implementation, the downtime would be limited to the length of time it takes to create or validate all the foreign keys, triggers, and sequences on the target database.

The pattern uses Amazon Elastic Compute Cloud (Amazon EC2) instances with an Oracle 10g/11g database installed to help AWS DMS stream the data. You can temporarily pause streaming replication from the on-premises Oracle database to the intermediate Oracle instance to enable AWS DMS to catch up on data validation or to use another data validation tool. The PostgreSQL DB instance and intermediate Oracle database will have the same data when AWS DMS has finished migrating current changes.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- A source Oracle 8i or 9i database in an on-premises data center
- AWS Direct Connect configured between the on-premises data center and AWS
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors installed either on a local machine or on the EC2 instance where AWS SCT is installed
- Familiarity with using an Oracle database as an AWS DMS source
- Familiarity with using a PostgreSQL database as an AWS DMS target
Limitations

- The database size limit is 64 TB

Product versions

- Oracle 8i or 9i for the source database
- Oracle 10g or 11g for the intermediate database
- PostgreSQL 9.6 or later

Architecture

Source technology stack

- Oracle 8i or 9i database

Target technology stack

- Amazon RDS for PostgreSQL or Aurora PostgreSQL

Target architecture

Tools

- AWS DMS – AWS Database Migration Service (AWS DMS) helps migrate databases quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.
• **AWS SCT** – AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database. Objects that cannot be automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project. During this process, AWS SCT performs cloud-native code optimization by converting legacy Oracle and SQL Server functions to their AWS equivalents, to help you modernize your applications while migrating your databases. When schema conversion is complete, AWS SCT can help migrate data from a range of data warehouses to Amazon Redshift by using built-in data migration agents.

**Epics**

**Install Oracle on an EC2 instance and create materialized views**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the network for the EC2 instance.</td>
<td>Create the virtual private cloud (VPC), subnets, internet gateway, route tables, and security groups.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create the EC2 instance.</td>
<td>Select the Amazon Machine Image (AMI) for the EC2 instance. Choose the instance size and configure instance details: the number of instances (1), the VPC and subnet from the previous step, auto-assign public IP, and other options. Add storage, configure security groups, and launch the instance. When prompted, create and save a key pair for the next step.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Install Oracle on the EC2 instance.</td>
<td>Acquire the licenses and the required Oracle binaries, and install Oracle 10g or 11g on the EC2 instance.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure Oracle networking.</td>
<td>Modify or add entries in listener.ora to connect to the on-premises source Oracle 8i/9i database, and then create the database links.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create materialized views.</td>
<td>Identify the database objects to replicate in the source Oracle 8i/9i database, and then create materialized views for all the objects by using the database link.</td>
<td>DBA</td>
</tr>
<tr>
<td>Deploy scripts to refresh materialized views at required intervals.</td>
<td>Develop and deploy scripts to refresh materialized views at required intervals on the Amazon EC2 Oracle 10g/11g</td>
<td>DBA</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Migrate from Oracle 8i/9i to Amazon RDS
for PostgreSQL using materialized views

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>instance. Use the incremental refresh option to refresh materialized views.</td>
<td></td>
</tr>
</tbody>
</table>

Convert the Oracle database schema to PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up AWS SCT.</td>
<td>Create a new report, and then connect to Oracle as the source and PostgreSQL as the target. In project settings, open the SQL Scripting tab. Change the target SQL script to Multiple Files. (AWS SCT doesn't support Oracle 8i/9i databases, so you have to restore the schema-only dump on the intermediate Oracle 10g/11g instance and use it as a source for AWS SCT.)</td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the Oracle database schema.</td>
<td>In the Action tab, choose Generate Report, Convert Schema, and then Save as SQL.</td>
<td>DBA</td>
</tr>
<tr>
<td>Modify the SQL scripts.</td>
<td>Make modifications based on best practices. For example, switch to suitable data types and develop PostgreSQL equivalents for Oracle-specific functions.</td>
<td>DBA, DevDBA</td>
</tr>
</tbody>
</table>

Create and configure the Amazon RDS DB instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Amazon RDS DB instance.</td>
<td>In the Amazon RDS console, create a new PostgreSQL DB instance.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure the DB instance.</td>
<td>Specify the DB engine version, DB instance class, Multi-AZ deployment, storage type, and allocated storage. Enter the DB instance identifier, a master user name, and a master password.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure network and security.</td>
<td>Specify the virtual private cloud (VPC), subnet group, public accessibility, Availability Zone preference, and security groups.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from Oracle 8i/9i to Amazon RDS for PostgreSQL using materialized views

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure database options.</td>
<td>Specify the database name, port, parameter group, encryption, and master key.</td>
<td>DBA, AWS SysAdmin</td>
</tr>
<tr>
<td>Configure backups.</td>
<td>Specify the backup retention period, backup window, start time, duration, and whether to copy tags to snapshots.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure monitoring options.</td>
<td>Enable or disable enhanced monitoring and performance insights.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure maintenance options.</td>
<td>Specify auto minor version upgrade, maintenance window, and the start day, time, and duration.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Run the pre-migration scripts from AWS SCT.</td>
<td>On the target Amazon RDS for PostgreSQL instance, create the database schema by using the SQL scripts from AWS SCT with other modifications. These might include running multiple scripts and including user creation, database creation, schema creation, tables, views, functions, and other code objects.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
</tbody>
</table>

### Migrate data by using AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a replication instance in AWS DMS.</td>
<td>Complete the fields for the name, instance class, VPC (same as for the EC2 instance), Multi-AZ, and public accessibility. In the advanced configuration section, specify allocated storage, subnet group, Availability Zone, VPC security groups, and AWS Key Management Service (AWS KMS) master key.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the source database endpoint.</td>
<td>Specify the endpoint name, type, source engine (Oracle), server name (the EC2 instance's private DNS name), port, SSL mode, user name, password, SID, VPC (specify the VPC that has the replication instance), and replication instance. To test the connection, choose</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Connect AWS DMS to Amazon RDS for PostgreSQL.</td>
<td>Create a migration security group for connections across VPCs, if your PostgreSQL database is in another VPC.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the target database endpoint.</td>
<td>Specify the endpoint name, type, source engine (PostgreSQL), server name (Amazon RDS endpoint), port, SSL mode, user name, password, database name, VPC (specify the VPC that has the replication instance), and replication instance. To test the connection, choose Run Test, and then create the endpoint. You can also configure the following advanced settings: maxFileSize and numberDataTypeScale.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the AWS DMS replication task.</td>
<td>Specify the task name, replication instance, source and target endpoints, and replication instance. For migration type, choose &quot;Migrate existing data and replicate ongoing changes.&quot; Clear the &quot;Start task on create&quot; checkbox.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure the AWS DMS replication task settings.</td>
<td>For target table preparation mode, choose &quot;Do nothing.&quot; Stop the task after full load completes (to create primary keys). Specify limited or full LOB mode, and enable control tables. Optionally, you can configure the CommitRate advanced setting.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure the table mappings.</td>
<td>In the Table mappings section, create an Include rule for all tables in all schemas included in the migration, and then create an Exclude rule. Add three transformation rules to convert the schema, table, and column names to lowercase, and add any other rules you need for this specific migration.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Migrate from Oracle 8i/9i to Amazon RDS for PostgreSQL using materialized views

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the task.</td>
<td>Start the replication task. Make sure that the full load is running. Run <code>ALTER SYSTEM SWITCH LOGFILE</code> on the primary Oracle database to kick-start the task.</td>
<td>DBA</td>
</tr>
<tr>
<td>Run the mid-migration scripts from AWS SCT.</td>
<td>In Amazon RDS for PostgreSQL, run the following scripts: <code>create_index.sql</code> and <code>create_constraint.sql</code> (if the complete schema wasn't initially created).</td>
<td>DBA</td>
</tr>
<tr>
<td>Resume the task to continue change data capture (CDC).</td>
<td>Run <code>VACUUM</code> on the Amazon RDS for PostgreSQL DB instance, and restart the AWS DMS task to apply cached CDC changes.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Cut over to the PostgreSQL database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check the AWS DMS logs and validation tables.</td>
<td>Check and fix any replication or validation errors.</td>
<td>DBA</td>
</tr>
<tr>
<td>Stop using the on-premises Oracle database and its dependencies.</td>
<td>Stop all Oracle dependencies, shut down listeners on the Oracle database, and run <code>ALTER SYSTEM SWITCH LOGFILE</code>. Stop the AWS DMS task when it shows no activity.</td>
<td>DBA</td>
</tr>
<tr>
<td>Run the post-migration scripts from AWS SCT.</td>
<td>In Amazon RDS for PostgreSQL, run these scripts: <code>create_foreign_key_constraint.sql</code> and <code>create_triggers.sql</code>. Make sure the sequences are up-to-date.</td>
<td>DBA</td>
</tr>
<tr>
<td>Complete additional Amazon RDS for PostgreSQL steps.</td>
<td>Increment sequences to match Oracle if needed, run <code>VACUUM</code> and <code>ANALYZE</code>, and take a snapshot for compliance.</td>
<td>DBA</td>
</tr>
<tr>
<td>Open the connections to Amazon RDS for PostgreSQL.</td>
<td>Remove the AWS DMS security groups from Amazon RDS for PostgreSQL, add production security groups, and point your applications to the new database.</td>
<td>DBA</td>
</tr>
<tr>
<td>Clean up the AWS DMS objects.</td>
<td>Remove the endpoints, replication tasks, replication instances, and the EC2 instance.</td>
<td>SysAdmin, DBA</td>
</tr>
</tbody>
</table>
Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT

Created by Anil Kunapareddy (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

Managing Oracle databases on Amazon Elastic Compute Cloud (Amazon EC2) instances requires resources and can be costly. Moving these databases to an Amazon Relational Database Service (Amazon RDS) for MySQL DB instance will ease your job by optimizing the overall IT budget. Amazon RDS for MySQL also provides features like Multi-AZ, scalability, and automatic backups.

This pattern walks you through the migration of a source Oracle database on Amazon EC2 to a target Amazon RDS for MySQL DB instance. It uses AWS Database Migration Service (AWS DMS) to migrate the data, and AWS Schema Conversion Tool (AWS SCT) to convert the source database schema and objects to a format that's compatible with Amazon RDS for MySQL.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source database with instance and listener services running, in ARCHIVELOG mode
- A target Amazon RDS for MySQL database, with sufficient storage for data migration

Limitations

- AWS DMS does not create a schema on the target database; you have to do that. The schema name must already exist for the target. Tables from the source schema are imported to user/schema, which AWS DMS uses to connect to the target instance. You must create multiple replication tasks if you have to migrate multiple schemas.

Product versions
• All Oracle database editions for versions 10.2 and later, 11g and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and Using a MySQL-Compatible Database as a Target for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.

• AWS DMS supports versions 5.5, 5.6, and 5.7 of MySQL.

Architecture

Source technology stack

• An Oracle database on an EC2 instance

Target technology stack

• Amazon RDS for MySQL DB instance

Data migration architecture

Source and target architecture
**Tools**

- **AWS DMS** - AWS Database Migration Service (AWS DMS) is a web service you can use to migrate data from your database that is on-premises, on an Amazon RDS DB instance, or in a database on an EC2 instance, to a database on an AWS service such as Amazon RDS for MySQL or an EC2 instance. You can also migrate a database from an AWS service to an on-premises database. You can migrate data between heterogeneous or homogenous database engines.

- **AWS SCT** - AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format that's compatible with the target database. After converting your database schema and code objects using AWS SCT, you can use AWS DMS to migrate data from the source database to the target database to complete your migration projects.

**Epics**

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the source and target database versions and engines.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify the DMS replication instance.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify storage requirements such as storage type and capacity.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify network requirements such as latency and bandwidth.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify hardware requirements for the source and target server instances (based on Oracle compatibility list and capacity requirements).</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Identify network access security requirements for source and target databases.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Install AWS SCT and Oracle drivers.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Determine a backup strategy.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Determine availability requirements.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify application migration and switch-over strategy.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Select the proper DB instance type based on capacity, storage, and network features.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
</tbody>
</table>

**Configure the environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC). The source, target, and replication instance should be in the same VPC. It is also good to have these in the same Availability Zone.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Create the necessary security groups for database access.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Generate and configure a key pair.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Configure subnets, Availability Zones, and CIDR blocks.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>

**Configure the source: Oracle database on EC2 instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Oracle Database on Amazon EC2 with required users and roles.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Perform the three steps in the next column to access Oracle from outside the EC2 instance.</td>
<td>(1) Change the local host in 'tnsnames' to the Amazon EC2 public DNS; (2) Change the local host in 'listener' to the Amazon EC2 public DNS; (3) Stop and restart the listener.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>When Amazon EC2 is restarted, the public DNS changes. Make sure to update Amazon EC2 public DNS in 'tnsnames' and 'listener' or use an Elastic IP address.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Configure the EC2 instance security group so that the replication instance and required clients can access the source database.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
</tbody>
</table>

**Configure the target: Amazon RDS for MySQL**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure and start the Amazon RDS for MySQL DB instance.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Create the necessary tablespace in the Amazon RDS for MySQL DB instance.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Configure the security group so that the replication instance and required clients can access the target database.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>

**Configure AWS SCT and create a schema in the target database**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install AWS SCT and Oracle drivers.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Enter the appropriate parameters and connect to the source and target.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Generate a schema conversion report.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Correct the code and schema as necessary, especially tablespaces and quotes, and run on the target database.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Validate the schema on source vs. target before migrating data.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>
# Migrate data using AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For full-load and change data capture (CDC) or just CDC, you must set an extra connection attribute.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>The user specified in the AWS DMS source Oracle database definitions must be granted all the required privileges. For a complete list, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Self-Managed">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Self-Managed</a></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td></td>
<td>Enable supplemental logging in the source database.</td>
<td>DBA/Developer</td>
</tr>
<tr>
<td></td>
<td>For full-load and change data capture (CDC) or just CDC, enable ARCHIVELOG mode in the source database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Create source and target endpoints, and test the connections.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>When the endpoints are connected successfully, create a replication task.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Select CDC only (or) full load plus CDC in the task to capture changes for continuous replication only (or) full load plus ongoing changes, respectively.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Run the replication task and monitor Amazon CloudWatch logs.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Validate the data in the source and target databases.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

# Migrate your application and cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Follow the steps for your application migration strategy.</td>
<td>DBA, Developer, App owner</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
--- | --- | ---
Follow the steps for your application cutover/switch-over strategy. |  | DBA, Developer, App owner

Close the project

Task | Description | Skills required
--- | --- | ---
Validate the schema and data in source vs. target databases. |  | DBA/Developer
Gather metrics around time to migrate, percent of manual vs. tool, cost savings, etc. |  | DBA/Developer/AppOwner
Review the project documents and artifacts. |  | DBA/Developer/AppOwner
Shut down temporary AWS resources. |  | DBA/Developer
Close out the project and provide feedback. |  | DBA/Developer/AppOwner

Related resources

- AWS DMS documentation
- AWS DMS website
- AWS DMS blog posts
- Strategies for Migrating Oracle Database to AWS
- Amazon RDS for Oracle FAQs
- Oracle FAQ
- Amazon EC2
- Amazon EC2 FAQs
- Licensing Oracle Software in the Cloud Computing Environment

Migrate from Oracle to Amazon DocumentDB using AWS DMS

*Created by Sashikanta Pattanayak (AWS)*

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon DocumentDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
</tbody>
</table>
Summary

This pattern provides guidance for migrating an Oracle database to an Amazon DocumentDB (with MongoDB compatibility) database by using AWS Database Migration Service (AWS DMS). This approach can be applied to an on-premises Oracle source database as well as an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance. This pattern uses an Amazon RDS Oracle DB source instance as an example.

Amazon DocumentDB (with MongoDB compatibility) is a fully managed, MongoDB-compatible document database service that makes it easy to store, query, and index JSON data.

The use case for this pattern is one-to-one replication of an Oracle database table to an Amazon DocumentDB collection. The pattern uses AWS DMS replication tasks to read the table structure of the Oracle database, create the corresponding collection in Amazon DocumentDB, and perform a full load migration. You can view and query your data in Amazon DocumentDB, the same as you would in MongoDB.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Familiarity with using Oracle databases
- Familiarity with using Amazon DocumentDB
- For the Oracle user, SELECT ANY TABLE privilege
- For the Amazon DocumentDB use, the privilege required to dump data

Limitations

The following limitations apply when using Amazon DocumentDB as a target for AWS DMS:

- In Amazon DocumentDB, collection names can't contain the dollar symbol ($). In addition, database names can't contain any Unicode characters.
- AWS DMS doesn't support merging of multiple source tables into a single Amazon DocumentDB collection.
- When AWS DMS processes changes from a source table that doesn't have a primary key, any large binary object (LOB) columns in that table are ignored.
- If the Change table option is enabled and AWS DMS encounters a source column named "_id", that column appears as "__id" (two underscores) in the change table.
- If you choose Oracle as a source endpoint, the Oracle source must have full supplemental logging enabled. Otherwise, if there are columns at the source that weren't changed, the data is loaded into Amazon DocumentDB as null values.

Product versions

- Amazon RDS for Oracle version 11.2.0.3 or later
• AWS DMS version 3.1.3 or later (for the latest version information, see Using Amazon DocumentDB as a Target for AWS DMS in the AWS DMS documentation)

Architecture

Source technology stack

• Amazon RDS for Oracle DB instance

Target technology stack

• Amazon DocumentDB

Source and target architecture

Tools

• AWS DMS – AWS Database Migration Service (AWS DMS) is a web service that you can use to migrate data from a source data store to a target data store. The AWS DMS User Guide specifies the Oracle source database versions and editions that are supported for use with AWS DMS. For additional information relevant to this pattern, see Using Amazon DocumentDB as a Target for AWS DMS.

• Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS cloud. Your Amazon DocumentDB cluster should be running in your default virtual private cloud (VPC). To interact with your Amazon DocumentDB cluster, you must launch an EC2 instance into your default VPC, in the same AWS Region where you created your Amazon DocumentDB cluster. For details, refer to Launch an Amazon EC2 instance in the Amazon DocumentDB documentation.
## Epics

### Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions and engines.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity, storage features, network features).</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Identify network/host access security requirements for the source and target databases.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Create an outbound security group to the source and target databases.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Create and configure an EC2 instance for Amazon DocumentDB.</td>
<td></td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>

### Configure infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC and subnets.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Create security groups and network access control lists (ACLs).</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Configure and start the source Amazon RDS for Oracle instance.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Configure and start the Amazon DocumentDB instance.</td>
<td></td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>

### Prepare the source database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify that the Oracle database can be connected using the connection details.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Verify that the Oracle user has the SELECT ANY TABLE privilege.</td>
<td></td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>
## Prepare the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Amazon DocumentDB cluster by choosing the proper instance class and number of instances.</td>
<td></td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>

## Configure Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the EC2 instance.</td>
<td>To interact with your Amazon DocumentDB cluster, you must launch an EC2 instance into your default VPC, in the same AWS Region where you created your Amazon DocumentDB cluster. Configure the AWS Region, VPCs, Availability Zones, and subnets for the EC2 instance.</td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Configure the key pair.</td>
<td>A public/private key pair allows you to connect securely to the EC2 instance after it launches.</td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Set the bastion host CIDR ranges (optional).</td>
<td>Set the CIDR IP range that is allowed for external Secure Shell (SSH) access to the bastion host instances.</td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>

## Migrate data – full load

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS DMS replication instance.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Create source and target endpoints.</td>
<td></td>
<td>AWS Admin</td>
</tr>
<tr>
<td>Create AWS DMS replication tasks for a full load.</td>
<td></td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>

## Test the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to the Amazon DocumentDB cluster through the EC2 instance.</td>
<td></td>
<td>AWS Admin</td>
</tr>
</tbody>
</table>
**Task** | **Description** | **Skills required**
---|---|---
Connect to the cluster using the mongo shell. | For instructions, see the Amazon DocumentDB links in the References and Help section. | AWS Admin
Verify the results of the migration. |  | AWS Admin

**Related resources**
- How AWS DMS Works
- Migrating to Amazon DocumentDB
- Using Amazon DocumentDB as a Target for AWS DMS
- Amazon DocumentDB overview
- Access and Use Your Amazon DocumentDB cluster Using the mongo Shell
- Migrate from MongoDB to Amazon DocumentDB using the offline method (blog post)
- How to use Amazon DocumentDB (with MongoDB compatibility) to build and manage applications at scale (blog post)

**Migrate from an on-premises Oracle database or Amazon RDS for Oracle to Amazon DynamoDB using AWS DMS and AWS SCT**

*Created by Rambabu Karnena (AWS)*

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon DynamoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services:</td>
<td>Amazon DynamoDB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern walks you through the steps for migrating a source, on-premises Oracle database or Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to a target Amazon DynamoDB database using AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT).

**Prerequisites and limitations**

**Prerequisites**
- An active AWS account

**Limitations**
• Oracle database objects, such as procedures, functions, packages, and triggers, are not considered for migration because Amazon DynamoDB does not support these database objects.

Product versions

• All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and Using an Amazon DynamoDB Database as a Target for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.

Architecture

Source technology stack

• Amazon RDS for Oracle DB instances or on-premises Oracle databases

Target technology stack

• Amazon DynamoDB

Source and target architecture

![Source and target architecture diagram](image_url)
AWS Prescriptive Guidance Patterns
Migrate from Oracle to Amazon DynamoDB

Tools

- **AWS DMS** - AWS Database Migration Service (AWS DMS) helps you migrate your data to and from widely used commercial and open-source databases, including Oracle, MySQL, and PostgreSQL. The service supports homogeneous migrations such as Oracle to Oracle, and heterogeneous migrations such as Oracle to PostgreSQL or MySQL to Oracle.

- **AWS SCT** - AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations easy by automatically converting the source database schema and a majority of the custom code to a format compatible with the target database.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Determine the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
# Migrate from Oracle to Amazon DynamoDB

## Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify network access security requirements for source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements, storage type, and capacity.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify hardware requirements for source and target server instances.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

## Configure the infrastructure

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets in your AWS account.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups and network access control lists (ACLs).</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon RDS for Oracle DB instance.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

## Migrate data

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the AWS Identity and Access Management (IAM) console, create a role to access DynamoDB.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create source and target endpoints in AWS DMS.</td>
<td>AWS DMS</td>
</tr>
<tr>
<td>For the source database endpoint, choose the endpoint from the Amazon RDS console, Databases, DB identifier, Connectivity &amp; Security. Alternatively, you can choose “Select RDS DB instance” from the AWS DMS console. For the target database endpoint, choose the role ARN from the previous task to access DynamoDB.</td>
<td>AWS DMS</td>
</tr>
<tr>
<td>Create an AWS DMS replication instance for migration.</td>
<td>AWS DMS</td>
</tr>
<tr>
<td>The replication instance should be in the same Availability Zone</td>
<td></td>
</tr>
</tbody>
</table>
Create an AWS DMS task to load the source Oracle database tables to DynamoDB. Choose the source and destination endpoint names and replication instance from the previous steps. The type can be full load. Choose the Oracle schema and specify % to select all tables.

Validate the tables in DynamoDB. To view the migration results, choose Tables from the left navigation pane in the DynamoDB console.

Migrate the application

Follow the application migration strategy.

Use AWS SCT to analyze and convert the SQL code inside the application code. For more information, see https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Converting.App.html.

Cut over

Switch the application clients over to use DynamoDB.

Close the project

Shut down temporary AWS resources.

Review and validate the project documents.

Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.

Close out the project and provide feedback.
Related resources

- AWS DMS website
- AWS DMS documentation
- Amazon DynamoDB documentation
- Amazon DynamoDB migration information
- AWS SCT documentation
- Using an Oracle Database as a Source for AWS DMS
- Amazon DynamoDB documentation
- Using an Amazon DynamoDB Database as a Target for AWS DMS
- Best Practices for Migrating from RDBMS to Amazon DynamoDB

Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT

Created by Veeranjaneyulu Grandhi (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for MariaDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern walks you through the steps for migrating an Oracle database on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon Relational Database Service (Amazon RDS) for MariaDB DB instance. The pattern uses AWS Data Migration Service (AWS DMS) for data migration and AWS Schema Conversion Tool (AWS SCT) for schema conversion.

Managing Oracle databases on EC2 instances requires more resources and is more costly than using a database on Amazon RDS. Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud. Amazon RDS provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching, and backups.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A source Oracle database with instance and listener services up and running. This database should be in ARCHIVELOG mode.
- Familiarity with Using an Oracle Database as a Source for AWS DMS.
AWS Prescriptive Guidance Patterns
Migrate from Oracle to Amazon RDS for MariaDB

- Familiarity with Using Oracle as a Source for AWS SCT.

Limitations
- Database size limit: 64 TB

Product versions
- All Oracle database editions for versions 10.2 and later, 11g and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and the AWS SCT version table in the AWS documentation.
- Amazon RDS supports MariaDB Server versions 10.0, 10.1, 10.2, and 10.3 (see the Amazon RDS documentation for the latest list of supported versions).

Architecture

Source technology stack
- An Oracle database on an EC2 instance

Target technology stack
- Amazon RDS for MariaDB

Data migration architecture

Target architecture
Tools

- **AWS SCT** - AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects – including views, stored procedures, and functions – to a format compatible with the target database. After converting your database schema and code objects using AWS SCT, you can use AWS DMS to migrate data from the source database to the target database to complete your migration projects. For more information, see Using Oracle as a Source for AWS SCT in the AWS SCT documentation.

- **AWS DMS** - AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases. AWS DMS supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora. To learn more about migrating Oracle databases, see Using an Oracle Database as a Source for AWS DMS in the AWS DMS documentation.

Epics

Plan for the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the source and target database versions and engines.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify the AWS DMS replication instance.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify network requirements (latency, bandwidth).</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Identify hardware requirements for the source and target server instances (based on the Oracle compatibility list and capacity requirements).</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Identify network-access security requirements for the source and target databases.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Install the latest AWS SCT and Oracle drivers.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Determine a backup strategy.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Determine availability requirements.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Choose an application migration/switchover strategy.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Select the proper instance type based on capacity, storage, and network features.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
</tbody>
</table>

**Configure the environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>The source, target, and replication instances should be in the same VPC and (recommended) in the same Availability Zone.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the necessary security groups for database access.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Generate and configure a key pair.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Configure subnets, Availability Zones, and CIDR blocks.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>

**Configure the source**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the EC2 instance.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Install the Oracle database on the EC2 instance, with required users and roles.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Follow the steps in the task description to access Oracle from outside of the EC2 instance.</td>
<td>1. Change the local host in &quot;tnsnames&quot; to the Amazon EC2 public DNS; 2. Change the local host in 'listener' to the Amazon EC2 public DNS; 3. Stop and restart the listener.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Configure the target Amazon RDS for MariaDB environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure and start the Amazon RDS for MariaDB DB instance.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Create any necessary tablespaces in the Amazon RDS MariaDB database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Configure a security group so the replication instance and required clients can access the target database.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>

### Configure AWS SCT

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the latest AWS SCT and Oracle drivers.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Enter appropriate parameters and then connect to the source and target.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Generate an AWS SCT schema conversion report.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Make any necessary corrections to the code and schema (especially tablespaces and quotation marks).</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Validate the schema on the source versus the target before loading data.</td>
<td></td>
<td>Developer</td>
</tr>
</tbody>
</table>
# Migrate from Oracle to Amazon RDS for MariaDB

## Migrate data using AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>For full-load and change data capture (CDC)</td>
<td>For full-load and change data capture (CDC) (or just for CDC), set an extra connection attribute.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>See <a href="https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MariaDB.html">https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MariaDB.html</a></td>
<td></td>
</tr>
<tr>
<td>Enable supplemental logging on the source</td>
<td>Enable supplemental logging on the source database.</td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For full-load and CDC (or just for CDC)</td>
<td>For full-load and CDC (or just for CDC), enable archive log mode on the source database.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create source and target endpoints and test</td>
<td>Create source and target endpoints and test the connections.</td>
<td>Developer</td>
</tr>
<tr>
<td>the connections.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>When the endpoints are connected successfully,</td>
<td>When the endpoints are connected successfully, create a replication task.</td>
<td>Developer</td>
</tr>
<tr>
<td>create a replication task.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select “CDC only” or “Full load + CDC” in the</td>
<td>Select “CDC only” or “Full load + CDC” in the task to capture changes for continuous replication only, or for full load and ongoing changes, respectively.</td>
<td>Developer</td>
</tr>
<tr>
<td>task to capture changes for continuous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>replication only, or for full load and ongoing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>changes, respectively.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start the replication task and monitor</td>
<td>Start the replication task and monitor Amazon CloudWatch Logs.</td>
<td>Developer</td>
</tr>
<tr>
<td>Amazon CloudWatch Logs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validate the data in the source and target</td>
<td>Validate the data in the source and target databases.</td>
<td>Developer</td>
</tr>
<tr>
<td>databases.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Migrate applications and cut over to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the chosen application migration</td>
<td>Follow the chosen application migration strategy.</td>
<td>DBA, Developer, App owner</td>
</tr>
<tr>
<td>strategy.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Follow the chosen application cutover/switch</td>
<td>Follow the chosen application cutover/switch strategy.</td>
<td>DBA, Developer, App owner</td>
</tr>
<tr>
<td>over strategy.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure that the schema and data are validated</td>
<td>Ensure that the schema and data are validated successfully in the</td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>successfully in the source database.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>source versus the target before project closure.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.</td>
<td></td>
<td>DBA, Developer, App owner</td>
</tr>
<tr>
<td>Review the project documents and artifacts.</td>
<td></td>
<td>DBA, Developer, App owner</td>
</tr>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>DBA, Developer, App owner</td>
</tr>
</tbody>
</table>

### Related resources

- MariaDB Amazon RDS overview
- Amazon RDS for MariaDB product details
- Using an Oracle Database as a Source for AWS DMS
- Strategies for Migrating Oracle Databases to AWS
- Licensing Oracle Software in the Cloud Computing Environment
- Amazon RDS for Oracle FAQs
- AWS DMS overview
- AWS DMS blog posts
- Amazon EC2 overview
- Amazon EC2 FAQs
- AWS SCT documentation

### Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT

*Created by Sergey Dmitriev (AWS)*

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Oracle</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

### Summary

This pattern walks you through the migration of an on-premises Oracle database to an Amazon Relational Database Service (Amazon RDS) for MySQL DB instance. It uses AWS Database Migration...
Service (AWS DMS) to migrate the data, and AWS Schema Conversion Tool (AWS SCT) to convert the source database schema and objects to a format that's compatible with Amazon RDS for MySQL.

**Prerequisites and limitations**

### Prerequisites

- An active AWS account
- A source Oracle database in an on-premises data center

### Limitations

- Database size limit: 64 TB

### Product versions

- All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- AWS DMS currently supports MySQL versions 5.5, 5.6, and 5.7. For the latest list of supported versions, see Using a MySQL-Compatible Database as a Target for AWS DMS in the AWS documentation.

### Architecture

**Source technology stack**

- On-premises Oracle database

**Target technology stack**

- Amazon RDS for MySQL DB instance

**Data migration architecture**
AWS Prescriptive Guidance Patterns
Migrate from Oracle to Amazon RDS for MySQL

**Tools**

- **AWS DMS** - *AWS Database Migration Services* (AWS DMS) helps you migrate relational databases, data warehouses, NoSQL databases, and other types of data stores. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.

- **AWS SCT** - *AWS Schema Conversion Tool* (AWS SCT) is used to convert your database schema from one database engine to another. The custom code that the tool converts includes views, stored procedures, and functions. Any code that the tool cannot convert automatically is clearly marked so that you can convert it yourself.

**Epics**

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from Oracle to Amazon RDS for MySQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create the security groups and network access control lists (ACLs).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an Amazon RDS DB instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the database schema by using AWS SCT.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Migrate data by using AWS DMS.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use AWS SCT to analyze and convert the SQL code inside the application code.</td>
<td>For more information, see <a href="https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Converting.App.html">https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Converting.App.html</a>.</td>
<td>App owner</td>
</tr>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
## Migrate from Oracle to Amazon RDS for PostgreSQL

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients over to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

**References**

- AWS DMS documentation
- AWS SCT documentation
- Amazon RDS Pricing

**Tutorial and videos**

- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS DMS (video)
- Amazon RDS (video)

### Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL using an Oracle bystander and AWS DMS

*Created by Cady Motyka (AWS)*

<table>
<thead>
<tr>
<th>Environment: PoC or pilot</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for PostgreSQL/Amazon Aurora PostgreSQL</th>
</tr>
</thead>
</table>

---

880
Summary

This pattern describes how you can migrate an on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL, using AWS Database Migration Service (AWS DMS) to migrate the data, AWS Schema Conversion Tool (AWS SCT) to convert the database schema, and an Oracle bystander to help manage the migration.

This pattern helps users who want to minimize database downtime. In this implementation, the downtime would be restricted to the length of time it takes to create or validate all the foreign keys on the database.

The pattern uses Amazon Elastic Compute Cloud (Amazon EC2) instances with an Oracle bystander database to help control the stream of data through AWS DMS. You can temporarily pause streaming replication from the on-premises Oracle database to the Oracle bystander to enable AWS DMS to catch up on data validation or to use another data validation tool. The Amazon RDS for PostgreSQL DB instance and the bystander database will have the same data when AWS DMS has finished migrating current changes.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database in an on-premises data center with an Active Data Guard standby database configured
- AWS Direct Connect configured between the on-premises data center and AWS Secrets Manager for storing the database secrets
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed either on a local machine or on the EC2 instance where AWS SCT is installed
- Familiarity with using an Oracle database as a source for AWS DMS
- Familiarity with using a PostgreSQL database as a target for AWS DMS

Limitations

- Database size limit: 64 TB

Product versions

- All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and Using a MySQL-Compatible Database as a Target for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- AWS DMS supports PostgreSQL versions 9.4 and later (for 9.x), 10.x, and 11.x. For the latest information, see Using a PostgreSQL Database as a Target for AWS DMS in the AWS documentation.
Architecture

Source technology stack

- An on-premises Oracle database
- An EC2 instance that holds a bystander for the Oracle database

Target technology stack

- Amazon RDS for PostgreSQL, PostgreSQL 9.3 and later

Database migration architecture

Tools

- **AWS DMS** - AWS Database Migration Service (DMS) helps you migrate databases quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.

- **AWS SCT** - AWS Schema Conversion Tool (SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database. Objects that cannot be automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project. During this process, AWS SCT performs cloud-native code optimization by converting legacy Oracle and SQL Server functions to their AWS equivalents, to help you modernize your applications while migrating your databases. When schema conversion is complete, AWS SCT can help migrate data from a range of data warehouses to Amazon Redshift by using built-in data migration agents.

- **Amazon RDS** - Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while
automating time-consuming administration tasks such as hardware provisioning, database setup, patching, and backups.

**Epics**

**Convert the Oracle database schema to PostgreSQL**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up AWS SCT.</td>
<td>Create a new report, and connect to Oracle as the source and PostgreSQL as the target. In Project Settings, go to the SQL Scripting tab. Change the Target SQL Script to Multiple Files.</td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the Oracle database schema.</td>
<td>In the Action tab, choose Generate Report, Convert Schema, and then Save as SQL.</td>
<td>DBA</td>
</tr>
<tr>
<td>Modify the scripts.</td>
<td>For example, you might want to modify the script if a number in the source schema has been converted to numeric format in PostgreSQL, but you want to use BIGINT instead for better performance.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Create and configure the Amazon RDS DB instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Amazon RDS DB instance.</td>
<td>In the correct AWS Region, in the Amazon RDS console, create a new PostgreSQL DB instance.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure DB instance specifications.</td>
<td>Specify the DB engine version, DB instance class, Multi-AZ deployment, storage type, and allocated storage. Enter the DB instance identifier, a master user name, and a master password.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure network and security.</td>
<td>Specify the virtual private cloud (VPC), subnet group, public accessibility, Availability Zone preference, and security groups.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Configure database options.</td>
<td>Specify the database name, port, parameter group, encryption, and KMS key.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure backups.</td>
<td>Specify the backup retention period, backup window, start</td>
<td>AWS SysAdmin, DBA</td>
</tr>
</tbody>
</table>
## Configure monitoring options
Enable or disable enhanced monitoring and performance insights.

**Skills required:** AWS SysAdmin, DBA

## Configure maintenance options
Specify auto minor version upgrade, maintenance window, and start day, time, and duration.

**Skills required:** AWS SysAdmin, DBA

## Run the pre-migration scripts from AWS SCT
On the Amazon RDS instance, run the following scripts: `create_database.sql`, `create_sequence.sql`, `create_table.sql`, `create_view.sql`, and `create_function.sql`.

**Skills required:** AWS SysAdmin, DBA

### Configure the Oracle bystander in Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the network for Amazon EC2.</td>
<td>Create the new VPC, subnets, internet gateway, route tables, and security groups.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create the EC2 instance.</td>
<td>In the appropriate AWS Region, create a new EC2 instance. Select the Amazon Machine Image (AMI), choose the instance size, and configure instance details: number of instances (1), the VPC and subnet you created in the previous task, auto-assign public IP, and other options. Add storage, configure security groups, and launch. When prompted, create and save a key pair for the next step.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Connect the Oracle source database to the EC2 instance.</td>
<td>Copy the IPv4 public IP address and DNS to a text file and connect by using SSH as follows: <code>ssh -i &quot;your_file.pem&quot; ec2-user@&lt;your-IP-address-or-public-DNS&gt;</code>.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Set up the initial host for a bystander in Amazon EC2.</td>
<td>Set up SSH keys, bash profile, ORATAB, and symbolic links. Create Oracle directories.</td>
<td>AWS SysAdmin, Linux Admin</td>
</tr>
<tr>
<td>Set up the database copy for a bystander in Amazon EC2</td>
<td>Use RMAN to create a database copy, enable supplemental logging, and create the standby.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
</tbody>
</table>
### Migrate from Oracle to Amazon RDS for PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prep the Oracle database to sync shipping.</td>
<td>Add the standby log files and change the recovery mode. Change the log shipping to SYNCHAFFIRM on both the source primary and the source standby. Switch logs on primary, confirm via the Amazon EC2 bystander alert log that you are using the standby log files, and confirm that the redo stream is flowing in SYNCH.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Set up Oracle Data Guard.</td>
<td>Modify your listener.ora file and start the listener. Set up a new archive destination. Place the bystander in recovery mode, replace temporary files to avoid future corruption, install a crontab if necessary to prevent the archive directory from running out of space, and edit the manage-trclog-files-oracle.cfg file for the source and standby.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Migrate data with AWS DMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Create a replication instance in AWS DMS.</td>
<td>Complete the fields for the name, instance class, VPC (same as the Amazon EC2 instance), Multi-AZ, and public accessibility. Under Advance, specify allocated storage, subnet group, Availability Zone, VPC security groups, and AWS Key Management Service (AWS KMS) key.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the source database endpoint.</td>
<td>Specify the endpoint name, type, source engine (Oracle), server name (Amazon EC2 private DNS name), port, SSL mode, user name, password, SID, VPC (specify the VPC that has the replication instance), and replication instance. To test the connection, choose Run Test, and then create the</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Connect AWS DMS to Amazon RDS for PostgreSQL.</td>
<td>Create a migration security group for connections across VPCs.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the target database endpoint.</td>
<td>Specify the endpoint name, type, source engine (PostgreSQL), server name (Amazon RDS endpoint), port, SSL mode, user name, password, database name, VPC (specify the VPC that has the replication instance), and replication instance. To test the connection, choose Run Test, and then create the endpoint. You can also configure the following advanced settings: maxFileSize and numberDataTypeScale.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Create the AWS DMS replication task.</td>
<td>Specify the task name, replication instance, source and target endpoints, and replication instance. For migration type, choose &quot;Migrate existing data and replicate ongoing changes.&quot; Clear the &quot;Start task on create&quot; checkbox.</td>
<td>AWS SysAdmin, DBA</td>
</tr>
<tr>
<td>Configure the AWS DMS replication task settings.</td>
<td>For target table preparation mode, choose &quot;Do nothing.&quot; Stop task after full load completes (to create primary keys). Specify limited or full LOB mode, and enable control tables. Optionally, you can configure the CommitRate advance setting.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure table mappings.</td>
<td>In the Table mappings section, create an Include rule for all tables in all schemas included in the migration, and then create an Exclude rule. Add three transformation rules to convert the schema, table, and column names to lowercase, and add any other rules needed for this specific migration.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Migrate from Oracle to Amazon RDS for PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start the task.</strong></td>
<td>Start the replication task. Make sure that the full load is running. Run <code>ALTER SYSTEM SWITCH LOGFILE</code> on the primary Oracle database to kick-start the task.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Run the mid-migration scripts from AWS SCT.</strong></td>
<td>In Amazon RDS for PostgreSQL, run the following scripts: <code>create_index.sql</code> and <code>create_constraint.sql</code>.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Restart the task to continue change data capture (CDC).</strong></td>
<td>Run <code>VACUUM</code> on the Amazon RDS for PostgreSQL DB instance, and restart the AWS DMS task to apply cached CDC changes.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Cut over to the PostgreSQL database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Review the AWS DMS logs and validation tables for any errors.</strong></td>
<td>Check and fix any replication or validation errors.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Stop all Oracle dependencies.</strong></td>
<td>Stop all Oracle dependencies, shut down listeners on the Oracle database, and run <code>ALTER SYSTEM SWITCH LOGFILE</code>. Stop the AWS DMS task when it shows no activity.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Run the post-migration scripts from AWS SCT.</strong></td>
<td>In Amazon RDS for PostgreSQL, run these scripts: <code>create_foreign_key_constraint.sql</code> and <code>create_triggers.sql</code>.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Complete additional Amazon RDS for PostgreSQL steps.</strong></td>
<td>Increment sequences to match Oracle if needed, run <code>VACUUM</code> and <code>ANALYZE</code>, and take a snapshot for compliance.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Open the connections to Amazon RDS for PostgreSQL.</strong></td>
<td>Remove the AWS DMS security groups from Amazon RDS for PostgreSQL, add production security groups, and point your applications to the new database.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Clean up AWS DMS objects.</strong></td>
<td>Remove the endpoints, replication tasks, replication instances, and the EC2 instance.</td>
<td>SysAdmin, DBA</td>
</tr>
</tbody>
</table>

### Related resources
Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate

Created by Dhairya Jindani (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an Oracle database running on premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance to Amazon Relational Database Service (Amazon RDS) for PostgreSQL by using Oracle GoldenGate.

Oracle GoldenGate is a tool for replicating data between a source database and one or more destination databases with minimal downtime.

Prerequisites and limitations

Prerequisites

- An Oracle GoldenGate license
- Knowledge of the fundamental building blocks of Oracle GoldenGate
- GoldenGate software on an EC2 instance
- GoldenGate configured on the source Oracle database
- GoldenGate configured on the target PostgreSQL database (including Amazon EC2 setup and the JDBC driver to connect to the PostgreSQL database)
- Schema and tables using the AWS Schema Conversion Tool (AWS SCT) on the target PostgreSQL database

Restrictions

- GoldenGate is used only to replicate existing table data (initial load) and ongoing changes (change data capture or CDC)

Product versions

- Oracle Database Enterprise Edition, 10g, 11g, 12c, or later version
- GoldenGate for Oracle version 12.2.0.1.1
AWS Prescriptive Guidance Patterns
Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate

- GoldenGate for PostgreSQL version 12.2.0.1.1

Architecture

Source technology stack
- Oracle database, either on premises or on an EC2 instance

Target technology stack
- Amazon RDS for PostgreSQL

Target architecture

Tools
- **Oracle GoldenGate**  - Oracle GoldenGate enables the exchange and manipulation of data at the transaction level among multiple, heterogeneous platforms across the enterprise. It moves committed transactions with transaction integrity and minimal overhead on an existing infrastructure.

Epics

Set up GoldenGate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download GoldenGate for Oracle.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Download GoldenGate for PostgreSQL.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install GoldenGate for Oracle on the Oracle database server.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Install GoldenGate for PostgreSQL database on the EC2 instance.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Prepare Oracle for GoldenGate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up GoldenGate for the Oracle database.</td>
<td>Set up includes supplemental logging, the GoldenGate user, and required grants and permissions.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure GoldenGate on Oracle.</td>
<td>Create and set up the parameter file, manager setup, GoldenGate directory, GLOBALS file, and wallet.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Configure GoldenGate on PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up GoldenGate for PostgreSQL.</td>
<td>Includes the creation and setup of GoldenGate manager, GLOBALS file, and wallet.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Change the data capture configuration for GoldenGate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the extract process in the source database.</td>
<td>Create an extract file on the source Oracle database to extract data. The extract file includes the creation of the extract parameter file and trail file directory.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set up a data pump to transfer the trail file from the source to the target database.</td>
<td>Setup includes creating the data pump parameter file and the trail file directory.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set up replication on the EC2 instance.</td>
<td>Setup includes creating the replication parameter file, creating the trail file directory, and adding a checkpoint table entry in the GLOBALS file at the target.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
Configure the initial load

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the parameter file at the source side to extract data for the initial load.</td>
<td>Make sure that the manager is running on the target.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set up the Replicat process for the initial load at the target.</td>
<td>Create a parameter file at the target to replicate data for the initial load. Add and start the Replicat process.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop replication on the task by ensuring that the source and target are in sync.</td>
<td>Make sure that data validation is successful.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create trigger, sequence, synonyms, and referential keys on the target.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Related resources

- Amazon RDS for PostgreSQL documentation
- Amazon EC2 documentation
- Oracle GoldenGate documentation

Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT

Created by Piyush Goyal (AWS)

<table>
<thead>
<tr>
<th>Source: Oracle</th>
<th>Target: Redshift</th>
<th>R Type: Re-architect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: Production</td>
<td>Technologies: Migration; Analytics; Databases</td>
<td>Workload: Oracle</td>
</tr>
<tr>
<td>AWS services: Amazon Redshift; AWS DMS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating Oracle databases to an Amazon Redshift cloud data warehouse in the Amazon Web Services (AWS) Cloud by using AWS Database Migration Service (AWS).
DMS) and AWS Schema Conversion Tool (AWS SCT). The pattern covers source Oracle databases that are on premises or installed on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It also covers Amazon Relational Database Service (Amazon RDS) for Oracle databases.

Prerequisites and limitations

Prerequisites

- An Oracle database that is running in an on-premises data center or in the AWS Cloud
- An active AWS account
- Familiarity with using an Oracle database as a source for AWS DMS
- Familiarity with using an Amazon Redshift database as a target for AWS DMS
- Knowledge of Amazon RDS, Amazon Redshift, the applicable database technologies, and SQL
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, where AWS SCT is installed

Product versions

- For self-managed Oracle databases, AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c. For Amazon RDS for Oracle databases that AWS manages, AWS DMS supports all Oracle database editions for versions 11g (versions 11.2.0.4 and later) and up to 12.2, 18c, and 19c. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture

Source technology stack

One of the following:

- An on-premises Oracle database
- An Oracle database on an EC2 instance
- An Amazon RDS for Oracle DB instance

Target technology stack

- Amazon Redshift

Target architecture

*From an Oracle database running in the AWS Cloud to Amazon Redshift:*
From an Oracle database running in an on-premises data center to Amazon Redshift:
Tools

- **AWS DMS** - AWS Data Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from most widely used commercial and open-source databases.

- **AWS SCT** - AWS Schema Conversion Tool (AWS SCT) can be used to convert your existing database schema from one database engine to another. It supports various database engines, including Oracle, SQL Server, and PostgresSQL, as sources.

Epics

Prepare for the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the database versions.</td>
<td>Validate the source and target database versions and make sure they are supported by AWS DMS. For information about supported Oracle Database versions, see Using an Oracle database as a source for AWS DMS. For information about using Amazon Redshift as a target, see Using an Amazon</td>
<td>DBA</td>
</tr>
</tbody>
</table>
## Migrate from Oracle to Amazon Redshift

### Task | Description | Skills required
---|---|---
Create a VPC and security group. | In your AWS account, create a virtual private cloud (VPC), if it doesn’t exist. Create a security group for outbound traffic to source and target databases. For more information, see the Amazon Virtual Private Cloud (Amazon VPC) documentation. | Systems administrator
Install AWS SCT. | Download and install the latest version of AWS SCT and its corresponding drivers. For more information, see Installing, verifying, and updating the AWS SCT. | DBA
Create a user for the AWS DMS task. | Create an AWS DMS user in the source database and grant it READ privileges. This user will be used by both AWS SCT and AWS DMS. | DBA
Test the DB connectivity. | Test the connectivity to the Oracle DB instance. | DBA
Create a new project in AWS SCT. | Open the AWS SCT tool and create a new project. | DBA
Analyze the Oracle schema to be migrated. | Use AWS SCT to analyze the schema to be migrated, and generate a database migration assessment report. For more information, see Creating a database migration assessment report in the AWS SCT documentation. | DBA
Review the assessment report. | Review the report for migration feasibility. Some DB objects might require manual conversion. For more information about the report, see Viewing the assessment report in the AWS SCT documentation. | DBA

Prepare the target database

### Task | Description | Skills required
---|---|---
Create an Amazon Redshift cluster. | Create an Amazon Redshift cluster within the VPC that | DBA
### AWS Prescriptive Guidance Patterns

**Migrate from Oracle to Amazon Redshift**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create database users.</strong></td>
<td>Extract the list of users, roles, and grants from the Oracle source database. Create users in the target Amazon Redshift database and apply the roles from the previous step.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Evaluate database parameters.</strong></td>
<td>Review the database options, parameters, network files, and database links from the Oracle source database, and evaluate their applicability to the target.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Apply any relevant settings to the target.</strong></td>
<td>For more information about this step, see Configuration reference in the Amazon Redshift documentation.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Create objects in the target database**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS DMS user in the target database.</td>
<td>Create an AWS DMS user in the target database and grant it read and write privileges. Validate the connectivity from AWS SCT.</td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the schema, review the SQL report, and save any errors or warnings.</td>
<td>For more information, see Converting database schemas using the AWS SCT in the AWS SCT documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Apply the schema changes to the target database or save them as a .sql file.</td>
<td>For instructions, see Saving and applying your converted schema in the AWS SCT in the AWS SCT documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Validate the objects in the target database.</td>
<td>Validate the objects that were created in the previous step in the target database. Rewrite or redesign any objects that weren’t successfully converted.</td>
<td>DBA</td>
</tr>
<tr>
<td>Disable foreign keys and triggers.</td>
<td>Disable any foreign key and triggers. These can cause data loading issues during the full</td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>load process when running AWS DMS.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Migrate data using AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS DMS replication instance.</td>
<td>Sign in to the AWS Management Console, and open the AWS DMS console. In the navigation pane, choose Replication instances, Create replication instance. For detailed instructions, see step 1 in Getting started with AWS DMS in the AWS DMS documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create source and target endpoints.</td>
<td>Create source and target endpoints, Test the connection from the replication instance to both source and target endpoints. For detailed instructions, see step 2 in Getting started with AWS DMS in the AWS DMS documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create a replication task.</td>
<td>Create a replication task and select the appropriate migration method. For detailed instructions, see step 3 in Getting started with AWS DMS in the AWS DMS documentation.</td>
<td>DBA</td>
</tr>
<tr>
<td>Start the data replication.</td>
<td>Start the replication task and monitor the logs for any errors.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate your application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create application servers.</td>
<td>Create the new application servers on AWS.</td>
<td>Application owner</td>
</tr>
<tr>
<td>Migrate the application code.</td>
<td>Migrate the application code to the new servers.</td>
<td>Application owner</td>
</tr>
<tr>
<td>Configure the application server.</td>
<td>Configure the application server for the target database and drivers.</td>
<td>Application owner</td>
</tr>
<tr>
<td>Optimize the application code.</td>
<td>Optimize the application code for the target engine.</td>
<td>Application owner</td>
</tr>
</tbody>
</table>
Cut over to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate users.</td>
<td>In the target Amazon Redshift database, validate users and grant them roles and privileges.</td>
<td>DBA</td>
</tr>
<tr>
<td>Validate that the application is locked.</td>
<td>Make sure that the application is locked, to prevent further changes.</td>
<td>Application owner</td>
</tr>
<tr>
<td>Validate the data.</td>
<td>Validate the data in the target Amazon Redshift database.</td>
<td>DBA</td>
</tr>
<tr>
<td>Enable foreign keys and triggers.</td>
<td>Enable foreign keys and triggers in the target Amazon Redshift database.</td>
<td>DBA</td>
</tr>
<tr>
<td>Connect to the new database.</td>
<td>Configure the application to connect to the new Amazon Redshift database.</td>
<td>Application owner</td>
</tr>
<tr>
<td>Perform final checks.</td>
<td>Perform a final, comprehensive system check before going live.</td>
<td>DBA, Application owner</td>
</tr>
<tr>
<td>Go live.</td>
<td>Go live with the target Amazon Redshift database.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Close the migration project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td>Shut down temporary AWS resources such as the AWS DMS replication instance and the EC2 instance used for AWS SCT.</td>
<td>DBA, Systems administrator</td>
</tr>
<tr>
<td>Review documents.</td>
<td>Review and validate the migration project documents.</td>
<td>DBA, Systems administrator</td>
</tr>
<tr>
<td>Gather metrics.</td>
<td>Collect information about the migration project, such as the time to migrate, the percentage of manual versus tool tasks, and total cost savings.</td>
<td>DBA, Systems administrator</td>
</tr>
<tr>
<td>Close out the project.</td>
<td>Close out the project and provide feedback.</td>
<td>DBA, Systems administrator</td>
</tr>
</tbody>
</table>

Related resources

References

- AWS DMS user guide
Migrate an on-premises Oracle database to Amazon Redshift using AWS SCT data extraction agents

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services: Amazon Redshift</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating an on-premises Oracle database to Amazon Redshift using AWS Schema Conversion Tool (AWS SCT) data extraction agents. An agent is an external program that is integrated with AWS SCT but performs data transformation elsewhere and interacts with other AWS services on your behalf.

Prerequisites and limitations

Prerequisites

- An Oracle source database used for the data warehouse workload in an on-premises data center
- An active AWS account

Product versions

- Oracle version 10 or later. For the latest list of supported versions, see the AWS SCT documentation.

Architecture

Source technology stack

- An on-premises Oracle database

Target technology stack

- Amazon Redshift
Data migration architecture

Tools

- **AWS SCT** – AWS Schema Conversion Tool (AWS SCT) handles heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. When the source and target databases are very different from one another, you can use an AWS SCT agent to perform additional data transformation. For more information, see Migrating Data from an On-Premises Data Warehouse to Amazon Redshift in the AWS documentation.

Epics

Prepare for migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions and engines.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity, storage features, network features).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network-access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose an application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
## Configure infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

## Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data using the AWS SCT data extraction agents.</td>
<td>For detailed information on using AWS SCT data extraction agents, see the links in the References and Help section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

## Migrate applications

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the chosen application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

## Cut over to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over the application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

## Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
### Related resources

#### References
- AWS SCT User Guide
- Using Data Extraction Agents
- Amazon Redshift Pricing

#### Tutorials and videos
- Getting Started with the AWS Schema Conversion Tool
- Getting Started with Amazon Redshift

### Migrate an Oracle database to Amazon Aurora MySQL using AWS DMS and AWS SCT

*Created by Mark Szalkiewicz (AWS)*

<table>
<thead>
<tr>
<th>R Type</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Aurora MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services: Amazon Aurora</td>
</tr>
</tbody>
</table>

### Summary

This pattern provides guidance for migrating an Oracle database to Amazon Aurora MySQL-Compatible Edition using AWS Data Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT).

This pattern covers three types of source databases: on-premises Oracle databases, Oracle databases on Amazon Elastic Compute Cloud (Amazon EC2) instances, and Amazon Relational Database Service (Amazon RDS) for Oracle DB instances. The target is an Aurora MySQL-Compatible database.

### Prerequisites and limitations

#### Prerequisites
- An active AWS account
- SQL clients installed either on a local machine or on an EC2 instance
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed either on a local machine or on an EC2 instance where AWS SCT is installed
- Familiarity with Using an Oracle Database as a Source for AWS DMS
- Familiarity with Using a MySQL-Compatible Database as a Target for AWS DMS
Limitations

- Database size limit: 64 TB

Product versions

- All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and Using a MySQL-Compatible Database as a Target for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- MySQL versions 5.5, 5.6, and 5.7

Architecture

Source technology stack

One of the following:

- An on-premises Oracle database
- An Oracle database on an EC2 instance
- An Amazon RDS for Oracle DB instance

Target technology stack

- Aurora MySQL-Compatible DB instance

Data migration architecture

For an Oracle source database running in the AWS Cloud:

For an on-premises Oracle source database:
Tools

- **AWS DMS** - *AWS Database Migration Services* (AWS DMS) helps you migrate relational databases, data warehouses, NoSQL databases, and other types of data stores. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.

- **AWS SCT** - *AWS Schema Conversion Tool* (AWS SCT) is used to convert your database schema from one database engine to another.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an outbound security group to the source and target databases.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create and configure an EC2 instance for AWS SCT, if required.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Download AWS SCT and drivers.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Add and validate prerequisite users and grants in the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Create an AWS SCT project for the workload and connect to the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Generate an assessment report and evaluate feasibility.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Prepare the target database**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon RDS target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Extract the lists for users, roles, and grants from the source.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Map existing database users to new database users.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Create users in the target database.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Apply roles from the preceding step to the target Amazon RDS database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Review the source database options, parameters, network files, and database links, and evaluate their applicability to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Apply any relevant settings to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Transfer objects**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure AWS SCT connectivity to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the schema in AWS SCT.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Review the generated SQL code, and save the error and warning objects.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Apply the automated schema changes to the target, or save them as a .sql file.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>
## Migrate from Oracle to Aurora MySQL

### Validate AWS SCT object creation in the target database.
- **Skills required:** DBA

### Manually rewrite, reject, or redesign any items that failed to convert automatically.
- **Skills required:** DBA, App owner

### Apply the role and user grants that were generated, and review exceptions.
- **Skills required:** DBA

### Migrate the database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the migration method.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a replication instance in the AWS DMS console.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create source and target endpoints.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a replication task.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Run the replication task and monitor the logs.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use AWS SCT to analyze and convert the SQL code inside the application code.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Create the new application servers on AWS.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Migrate the application code to the new servers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Configure the application server for the target database and drivers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Fix any source database engine-specific code in the application.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Optimize the application code for the target database engine.</td>
<td></td>
<td>App owner</td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administer any new users, grants, and code changes in the target database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Lock the application from any further changes.</td>
<td>App owner</td>
<td></td>
</tr>
<tr>
<td>Validate that all changes have been propagated to the target database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Point the new application server to the target database.</td>
<td>App owner</td>
<td></td>
</tr>
<tr>
<td>Perform sanity checks.</td>
<td>App owner</td>
<td></td>
</tr>
<tr>
<td>Go live.</td>
<td>App owner</td>
<td></td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources (AWS DMS replication instance, EC2 instance used for AWS SCT).</td>
<td>DBA, App owner</td>
<td></td>
</tr>
<tr>
<td>Update feedback on the AWS DMS process to internal teams.</td>
<td>DBA, App owner</td>
<td></td>
</tr>
<tr>
<td>Revise the AWS DMS process and improve the template.</td>
<td>DBA, App owner</td>
<td></td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td>DBA, App owner</td>
<td></td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td>DBA, App owner</td>
<td></td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td>DBA, App owner</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

References

- Migrating Your Databases to Amazon Aurora
- Migrating an Amazon RDS for Oracle database to Amazon Aurora MySQL-Compatible Edition (using AWS DMS)
- AWS DMS website
Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT

Created by Mark Szalkiewicz (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Aurora PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>AWS services: Amazon Aurora</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an Oracle database to Amazon Aurora PostgreSQL-Compatible Edition by using AWS Data Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT).

The pattern covers source Oracle databases that are on premises, Oracle databases that are installed on Amazon Elastic Compute Cloud (Amazon EC2) instances, and Amazon Relational Database Service (Amazon RDS) for Oracle databases. The pattern converts these databases to Aurora PostgreSQL-Compatible.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Oracle database in an on-premises data center or in the AWS Cloud.
- SQL clients installed either on a local machine or on an EC2 instance.
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed on either a local machine or an EC2 instance where AWS SCT is installed.
- If the source database supports a commercial off-the-shelf (COTS) application or is vendor-specific, you might not be able to convert it to another database engine. Before using this pattern, confirm that the application supports Aurora PostgreSQL-Compatible.
Limitations

- Database size limit: 64 TB

Product versions

- All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and Using a PostgreSQL Database as a Target for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- Aurora supports the PostgreSQL versions listed in Database Engine Versions for Aurora PostgreSQL-Compatible.

Architecture

Source technology stack

One of the following:

- An on-premises Oracle database
- An Oracle database on an EC2 instance
- An Amazon RDS for Oracle DB instance

Target technology stack

- Aurora PostgreSQL-Compatible

Target architecture
**Data migration architecture**

- *From an Oracle database running in the AWS Cloud*

- *From an Oracle database running in an on-premises data center*
Tools

- **AWS DMS** - *AWS Data Migration Service* (AWS DMS) helps you migrate your data to and from widely used commercial and open-source databases, including Oracle, MySQL, and PostgreSQL. The service supports homogeneous migrations such as Oracle to Oracle, and heterogeneous migrations such as Oracle to PostgreSQL or MySQL to Oracle.

- **AWS SCT** - *AWS Schema Conversion Tool* (AWS SCT) makes heterogeneous database migrations easy by automatically converting the source database schema and a majority of the custom code to a format compatible with the target database.

Epics

**Prepare for the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an outbound security group to the source and target databases.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create and configure an EC2 instance for AWS SCT, if required.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Download the latest version of AWS SCT and associated drivers.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Add and validate the prerequisite users and grants in the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an AWS SCT project for the workload and connect to the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Generate an assessment report and evaluate feasibility.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Prepare the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a target Amazon RDS DB instance, using Amazon Aurora as the database engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Extract the list of users, roles, and grants from the source.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Map the existing database users to the new database users.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Create users in the target database.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Apply roles from the previous step to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Review the database for options, parameters, network files, and database links from the source and then evaluate applicability to the target.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Apply any relevant settings to the target.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Transfer objects

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure AWS SCT connectivity to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the schema using AWS SCT.</td>
<td>AWS SCT automatically converts the source database schema and most of the custom code to a format that is compatible with the target database. Any code that the tool cannot convert automatically is clearly</td>
<td>DBA</td>
</tr>
</tbody>
</table>
## Migrate from Oracle to Aurora PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Review the generated SQL report and save any errors and warnings.</strong></td>
<td>marked so that you can convert it yourself.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Apply automated schema changes to the target or save them as a .sql file.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Validate that AWS SCT created the objects on the target.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Manually rewrite, reject, or redesign any items that failed to convert automatically.</strong></td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td><strong>Apply the generated role and user grants and review any exceptions.</strong></td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Determine the migration method.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Create a replication instance from the AWS DMS console.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Create the source and target endpoints.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Create a replication task.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Start the replication task and monitor the logs.</strong></td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use AWS SCT to analyze and convert the SQL items within the application code.</strong></td>
<td>When you convert your database schema from one engine to another, you also need to update the SQL code in your applications to interact with the new database engine instead of the old one. You can view, analyze, edit, and save the converted SQL code.</td>
<td>App owner</td>
</tr>
</tbody>
</table>
### Migrate from Oracle to Aurora PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the new application servers on AWS.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Migrate the application code to the new servers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Configure the application server for the target database and drivers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Fix any code specific to the source database engine in the application.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Optimize the application code for the target engine.</td>
<td></td>
<td>App owner</td>
</tr>
</tbody>
</table>

**Cut over**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform the cutover.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Lock the application from any further changes.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Validate that all changes were propagated to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Point the new application server to the target database.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Perform a final comprehensive system check.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Go live.</td>
<td></td>
<td>App owner</td>
</tr>
</tbody>
</table>

**Close the project**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources (AWS DMS replication instance and EC2 instance used for AWS SCT).</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Update feedback on the AWS DMS process for internal teams.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Revise the AWS DMS process and improve the template if necessary.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
--- | --- | ---
Review and validate the project documents. |  | DBA, App owner
Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on. |  | DBA, App owner
Close the project and provide any feedback. |  | DBA, App owner

Related resources

References
- Using an Oracle Database as a Source for AWS DMS
- Using a PostgreSQL Database as a Target for AWS Database Migration Service
- Oracle Database 11g/12c to Amazon Aurora with PostgreSQL Compatibility (9.6.x) Migration Playbook
- Migrating an Amazon RDS for Oracle database to Amazon Aurora MySQL-Compatible Edition
- AWS Data Migration Service
- AWS Schema Conversion Tool
- Migrate from Oracle to Amazon Aurora
- Amazon RDS pricing

Tutorials and videos
- AWS DMS Step-by-Step Walkthroughs
- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS Data Migration Service (video)

Migrate data from an on-premises Oracle database to Aurora PostgreSQL

Created by Michelle Deng (AWS) and Shunan Xiang (AWS)

Environment: PoC or pilot  |  Source: Oracle  |  Target: Aurora PostgreSQL-Compatible
R Type: Re-architect  |  Workload: Oracle  |  Technologies: Migration; Databases
AWS services: Amazon Aurora; AWS DMS; AWS SCT
Summary

This pattern provides guidance for data migration from an on-premises Oracle database to Amazon Aurora PostgreSQL-Compatible Edition. It targets an online data migration strategy with a minimal amount of downtime for multi-terabyte Oracle databases that contain large tables with high data manipulation language (DML) activities. An Oracle Active Data Guard standby database is used as the source to offload data migration from the primary database. The replication from the Oracle primary database to standby can be suspended during the full load to avoid ORA-01555 errors.

Table columns in primary keys (PKs) or foreign keys (FKs), with data type NUMBER, are commonly used to store integers in Oracle. We recommend that you convert these to INT or BIGINT in PostgreSQL for better performance. You can use the AWS Schema Conversion Tool (AWS SCT) to change the default data type mapping for PK and FK columns. (For more information, see the AWS blog post Convert the NUMBER data type from Oracle to PostgreSQL.) The data migration in this pattern uses AWS Database Migration Service (AWS DMS) for both full load and change data capture (CDC).

You can also use this pattern to migrate an on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL, or an Oracle database that's hosted on Amazon Elastic Compute Cloud (Amazon EC2) to either Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Oracle source database in an on-premises data center with Active Data Guard standby configured
- AWS Direct Connect configured between the on-premises data center and the AWS Cloud
- Familiarity with using an Oracle database as a source for AWS DMS
- Familiarity with using a PostgreSQL database as a target for AWS DMS

Limitations

- Amazon Aurora database clusters can be created with up to 128 TiB of storage. Amazon RDS for PostgreSQL database instances can be created with up to 64 TiB of storage. For the latest storage information, see Amazon Aurora storage and reliability and Amazon RDS DB instance storage in the AWS documentation.

Product versions

- AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS in the AWS documentation.

Architecture

Source technology stack

- On-premises Oracle databases with Oracle Active Data Guard standby configured

Target technology stack

- Aurora PostgreSQL-Compatible

Data migration architecture
Tools

- **AWS DMS - AWS Database Migration Service** (AWS DMS) supports several source and target databases. See [Using an Oracle Database as a Source for AWS DMS](#) in the AWS DMS documentation for a list of supported Oracle source and target database versions and editions. If the source database is not supported by AWS DMS, you must select another method for migrating the data in Phase 6 (in the *Epics* section). **Important note:** Because this is a heterogeneous migration, you must first check to see whether the database supports a commercial off-the-shelf (COTS) application. If the application is COTS, consult the vendor to confirm that Aurora PostgreSQL-Compatible is supported before proceeding. For more information, see [AWS DMS Step-by-Step Migration Walkthroughs](#) in the AWS documentation.

- **AWS SCT - The AWS Schema Conversion Tool** (AWS SCT) facilitates heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. The custom code that the tool converts includes views, stored procedures, and functions. Any code that the tool cannot convert automatically is clearly marked so that you can convert it yourself.

**Epics**

**Plan the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Install AWS SCT and drivers.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Add and validate the AWS SCT prerequisite users and grants-source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an AWS SCT project for the workload, and connect to the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Generate an assessment report and evaluate feasibility.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>
Prepare the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Aurora PostgreSQL-Compatible target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Extract users, roles, and grants list from the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Map the existing database users to the new database users.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Create users in the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Apply roles from the previous step to the target Aurora PostgreSQL-Compatible database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Review database options, parameters, network files, and database links from the source database, and evaluate their applicability to the target database.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Apply any relevant settings to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Prepare for database object code conversion

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure AWS SCT connectivity to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Convert the schema in AWS SCT, and save the converted code as a .sql file.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Manually convert any database objects that failed to convert automatically.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Optimize the database code conversion.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Separate the .sql file into multiple .sql files based on the object type.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Validate the SQL scripts in the target database.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>
### Prepare for data migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS DMS replication instance.</td>
<td></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Create the source and target endpoints.</td>
<td>If the data type of the PKs and FKs is converted from NUMBER in Oracle to BIGINT in PostgreSQL, consider specifying the connection attribute numberDataTypeScale=-2 when you create the source endpoint.</td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>

### Migrate data – full load

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the schema and tables in the target database.</td>
<td></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Create AWS DMS full-load tasks by either grouping tables or splitting a big table based on the table size.</td>
<td></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Stop the applications on the source Oracle databases for a short period.</td>
<td></td>
<td>App owner</td>
<td></td>
</tr>
<tr>
<td>Verify that the Oracle standby database is synchronous with the primary database, and stop the replication from the primary database to the standby database.</td>
<td></td>
<td>DBA, App owner</td>
<td></td>
</tr>
<tr>
<td>Start applications on the source Oracle database.</td>
<td></td>
<td>App owner</td>
<td></td>
</tr>
<tr>
<td>Start the AWS DMS full-load tasks in parallel from the Oracle standby database to the Aurora PostgreSQL-Compatible database.</td>
<td></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Create PKs and secondary indexes after the full load is complete.</td>
<td></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Validate the data.</td>
<td></td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>
Migrate data – CDC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create AWS DMS ongoing replication tasks by specifying a custom CDC start time or system change number (SCN) when the Oracle standby was synchronized with the primary database, and before the applications were restarted in the previous task.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Start AWS DMS tasks in parallel to replicate ongoing changes from the Oracle standby database to the Aurora PostgreSQL-Compatible database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Re-establish the replication from the Oracle primary database to the standby database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Monitor the logs and stop the applications on the Oracle database when the target Aurora PostgreSQL-Compatible database is almost synchronous with the source Oracle database.</td>
<td>DBA, App owner</td>
</tr>
<tr>
<td></td>
<td>Stop the AWS DMS tasks when the target is fully synchronized with the source Oracle database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Create FKs and validate the data in the target database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Create functions, views, triggers, sequences, and other object types in the target database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Apply role grants in the target database.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use AWS SCT to analyze and convert the SQL statements inside the application code.</td>
<td>App owner</td>
</tr>
<tr>
<td></td>
<td>Create new application servers on AWS.</td>
<td>App owner</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Migrate from Oracle with standby to Aurora PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the application code to the new servers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Configure the application server for the target database and drivers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Fix any code that's specific to the source database engine in the application.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Optimize the application code for the target database.</td>
<td></td>
<td>App owner</td>
</tr>
</tbody>
</table>

**Cut over**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point the new application server to the target database.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Perform sanity checks.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Go live.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

**Close the project**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, Systems administrator</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Gather metrics for time to migrate, percentage of manual versus tool use, cost savings, and similar data.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

**Related resources**

**References**

- Oracle Database to Aurora PostgreSQL-Compatible: Migration Playbook
- Migrating an Amazon RDS for Oracle Database to Amazon Aurora MySQL
- AWS DMS website
- AWS DMS documentation
Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for SQL Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>SAP</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating an SAP Adaptive Server Enterprise (ASE) database to an Amazon Relational Database Service (Amazon RDS) DB instance that’s running Microsoft SQL Server. The source database can be located in an on-premises data center or on an Amazon Elastic Compute Cloud (Amazon EC2) instance. The pattern uses AWS Database Migration Service (AWS DMS) to migrate data and (optionally) computer-aided software engineering (CASE) tools to convert the database schema.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An SAP ASE database in an on-premises data center or on an EC2 instance

Limitations

- Database size limit: 64 TB

Product versions

- SAP ASE version 15.x or 16.x and later. For the latest information, see Using an SAP Database as a Source for AWS DMS.
about supported versions, see the AWS DMS documentation. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture

Source technology stack

- An SAP ASE database that's on premises or on an Amazon EC2 instance

Target technology stack

- An Amazon RDS for SQL Server DB instance

Source and target architecture

*From an SAP ASE database on Amazon EC2 to an Amazon RDS for SQL Server DB instance:*

*From an on-premises SAP ASE database to an Amazon RDS for SQL Server DB instance:*

923
Tools

- **AWS DMS** - AWS Database Migration Service (AWS DMS) is a web service you can use to migrate data from your database that is on-premises, on an Amazon RDS DB instance, or in a database on an EC2 instance, to a database on an AWS service such as Amazon RDS for SQL Server or an EC2 instance. You can also migrate a database from an AWS service to an on-premises database. You can migrate data between heterogeneous or homogenous database engines.

- **CASE tools** - For schema conversions, you can optionally use erwin Data Modeler or SAP PowerDesigner.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Migrate from SAP ASE to Amazon RDS for SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identify the network access security requirements for the source and target databases.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Identify the application migration strategy.</strong></td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Configure the infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a virtual private cloud (VPC) and subnets.</strong></td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td><strong>Create security groups and network access control lists (ACLs).</strong></td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td><strong>Configure and start an Amazon RDS DB instance.</strong></td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

**Migrate data - option 1**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Migrate the database schema manually or use a CASE tool such as erwin Data Modeler or SAP PowerDesigner.</strong></td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Migrate data - option 2**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Migrate data using AWS DMS.</strong></td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Migrate the application**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Follow the application migration strategy.</strong></td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application</td>
<td>Switch the application clients over to the new infrastructure.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>clients</td>
<td></td>
<td></td>
</tr>
<tr>
<td>over to the new</td>
<td></td>
<td></td>
</tr>
<tr>
<td>infrastructure.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary</td>
<td>Shut down the temporary AWS resources.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>AWS resources.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review and validate the</td>
<td>Review and validate the project documents.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>project documents.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gather metrics around</td>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings,</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>time to migrate, % of</td>
<td>etc.</td>
<td></td>
</tr>
<tr>
<td>manual vs. tool, cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>savings, etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close out the project</td>
<td>Close out the project and provide feedback.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>and provide feedback.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Related resources

References

- AWS DMS website
- Amazon RDS Pricing
- Using a SAP ASE Database as a Source for AWS DMS

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS DMS (video)
- Amazon RDS (video)

Migrate an SAP ASE database to Aurora MySQL using AWS DMS

Created by Sergey Dmitriev (AWS)

R Type: Re-architect  Source: Databases: Relational  Target: Amazon Aurora MySQL
Summary

This pattern provides guidance for migrating SAP Adaptive Server Enterprise (ASE) databases to Amazon Aurora MySQL-Compatible Edition DB instances using AWS Database Migration Service (AWS DMS). The pattern applies to SAP ASE source databases in an on-premises data center or on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An SAP ASE source database in an on-premises data center or on an EC2 instance

Limitations

- Database size must be less than 64 TB

Product versions

- SAP ASE version 15.x or later (for the latest information, see Using an SAP Database as a Source for AWS DMS)
- MySQL 5.x or later (for details, see Using a MySQL-Compatible Database as a Target for AWS DMS)
- Amazon Aurora 1.x or later (for the latest information, see Database Engine Updates for Aurora MySQL-Compatible)

Architecture

Source technology stack

- SAP ASE database (on premises or on an EC2 instance)

Target technology stack

- Aurora MySQL-Compatible DB instance

Data migration architecture

From an SAP ASE database on Amazon EC2 to an Aurora MySQL-Compatible DB instance:
From an on-premises SAP ASE database to an Aurora MySQL-Compatible DB instance:
Tools

- **AWS DMS** - AWS Database Migration Service (AWS DMS) supports several different source and target databases. For details, see Sources for Data Migration and Targets for Data Migration. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Epics

Prepare for the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity, storage features, network features).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from SAP ASE to Aurora MySQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose an application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Configure infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td>A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. In AWS DMS, if you use your own security group, egress must, at a minimum, be permitted to the source and target endpoints on the respective database ports.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an Aurora MySQL-Compatible DB instance.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate data using AWS DMS.</td>
<td>We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate applications

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the chosen application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Cut over to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

References

- AWS Database Migration Service
- Amazon Aurora Pricing
- Best Practices with Amazon Aurora MySQL-Compatible Edition

Tutorials and videos

- Getting Started with AWS Database Migration Service
- Using an SAP ASE Database as a Source for AWS DMS
- AWS Database Migration Service (DMS) (video)

Migrate an SAP ASE database to Aurora PostgreSQL using AWS DMS

*Created by Sergey Dmitriev (AWS)*

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Aurora PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: SAP</td>
<td>AWS services: Amazon Aurora</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an SAP Adaptive Server Enterprise (ASE) database to an Amazon Aurora PostgreSQL-Compatible Edition instance by using AWS Database Migration Service (AWS DMS).
The pattern applies to SAP ASE source databases in an on-premises data center or on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

**Prerequisites and limitations**

**Prerequisites**
- An active AWS account
- An SAP ASE database in an on-premises data center or on an EC2 instance

**Limitations**
- The source database must be less than 64 TB

**Product versions**
- SAP ASE version 15.x and 16.x or later (for the latest information, see Using an SAP Database as a Source for AWS DMS)
- PostgreSQL 9.6 or later (for details, see Using a PostgreSQL Database as a Target for AWS DMS)
- Amazon Aurora 1.x or later (for the latest information, see Aurora PostgreSQL-Compatible releases and engine versions)

**Architecture**

**Source technology stack**
- SAP ASE database (on premises or on an EC2 instance)

**Target technology stack**
- Aurora PostgreSQL-Compatible DB instance

**Data migration architecture**

*From an SAP ASE database on Amazon EC2 to Aurora PostgreSQL-Compatible*
From an on-premises SAP ASE database to Aurora PostgreSQL-Compatible
Tools

- AWS DMS - AWS Data Migration Service (AWS DMS) supports several different source and target databases. For more information, see Sources for Data Migration and Targets for Data Migration. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

- An optional, third-party CASE (computer-aided software engineering) tool.

Epics

Analyze the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the storage requirements for the storage type and capacity.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type, capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from SAP ASE to Aurora PostgreSQL

### Identify the network access security requirements for the source and target databases.
- **Skills required**: DBA, SysAdmin

### Identify the application migration strategy.
- **Skills required**: DBA, SysAdmin, App owner

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon Aurora cluster.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate the data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data by using AWS DMS.</td>
<td>We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over the application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Validate and review the</td>
<td>Validate and review the project documents.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>project documents.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gather metrics around</td>
<td>Gather metrics around time to migrate, percent of manual versus tool cost</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>time to migrate, percent</td>
<td>savings, and so on.</td>
<td></td>
</tr>
<tr>
<td>of manual versus tool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cost savings, and so on.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close the project and</td>
<td>Close the project and provide any feedback.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>provide any feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

**References**

- AWS Database Migration Service
- Amazon Aurora pricing
- Best practices with Amazon Aurora PostgreSQL-Compatible Edition

**Tutorials and videos**

- Getting Started with AWS Database Migration Service
- Using an SAP ASE Database as a Source for AWS DMS
- AWS Database Migration Service (video)

**Migrate from SAP ASE to PostgreSQL on Amazon EC2 using AWS DMS**

*Created by Prashant Borse (AWS)*

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: PostgreSQL on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: SAP</td>
<td>AWS services: Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern provides guidance for migrating an on-premises SAP Adaptive Server Enterprise (ASE) database to a PostgreSQL database on an Amazon Elastic Compute Cloud (Amazon EC2) instance using AWS Database Migration Service (AWS DMS).

**Prerequisites and limitations**

**Prerequisites**
• An active AWS account
• An SAP ASE database in an on-premises data center

limitations
• Database size limit: 64 TB

Product versions
• SAP ASE version 15.x or 16.x and later. For the latest information, see Using an SAP Database as a Source for AWS DMS.
• AWS DMS supports PostgreSQL version 9.4 and later (for 9.x), 10.x, and 11.x. For the latest information about supported versions, see the AWS DMS documentation. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture
Source technology stack
• An on-premises SAP ASE database

Target technology stack
• A PostgreSQL database on an EC2 instance

Database migration architecture
Tools

- **AWS DMS - AWS Database Migration Service** (AWS DMS) is a web service you can use to migrate data from your database that is on-premises, on an Amazon RDS DB instance, or in a database on an EC2 instance, to a database on an AWS service such as Amazon RDS for SQL Server or an EC2 instance. You can also migrate a database from an AWS service to an on-premises database. You can migrate data between heterogeneous or homogenous database engines.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network requirements (latency, bandwidth).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from SAP ASE to PostgreSQL on Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose the proper instance type</strong> based on capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td><strong>Identify the network access security requirements for the source and target databases.</strong></td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td><strong>Determine the backup strategy for the target database.</strong></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td><strong>Determine availability and replication requirements.</strong></td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td><strong>Identify the application migration strategy.</strong></td>
<td>DBA, SysAdmin, App owner</td>
<td></td>
</tr>
</tbody>
</table>

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a virtual private cloud (VPC).</strong></td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td><strong>Create security groups.</strong></td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td><strong>Configure and start an EC2 instance.</strong></td>
<td>SysAdmin</td>
<td></td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Migrate data using AWS DMS.</strong></td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Follow the application migration strategy.</strong></td>
<td>DBA, SysAdmin, App owner</td>
<td></td>
</tr>
</tbody>
</table>

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Switch the application clients over to the new infrastructure.</strong></td>
<td>DBA, SysAdmin, App owner</td>
<td></td>
</tr>
</tbody>
</table>

939
Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

References

- Amazon EC2 website
- AWS DMS website
- Amazon EC2 Pricing

Tutorials and videos

- Getting Started with AWS DMS
- Using a SAP ASE Database as a Source for AWS DMS
- AWS DMS (video)

Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS

Created by Mark Szalkiewicz (AWS)

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Microsoft</td>
<td>AWS services: Amazon Redshift</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating an on-premises Microsoft SQL Server database to Amazon Redshift using AWS Data Migration Service (AWS DMS).
Prerequisites and limitations

Prerequisites

• An active AWS account
• A source Microsoft SQL Server database in an on-premises data center

Product versions

• SQL Server 2005-2017, Enterprise, Standard, Workgroup, and Developer editions. For the latest list of supported versions, see Using a Microsoft SQL Server Database as a Source for AWS DMS in the AWS documentation/

Architecture

Source technology stack

• An on-premises Microsoft SQL Server database

Target technology stack

• Amazon Redshift

Data migration architecture
Tools

- **AWS DMS** - *AWS Database Migration Services* (AWS DMS) supports several types of source and target databases. For information about the Microsoft SQL Server database versions and editions that are supported for use with AWS DMS, see *Using a Microsoft SQL Server Database as a Source for AWS DMS* in the AWS DMS documentation. If AWS DMS doesn't support your source database, you must select an alternative method for data migration.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>See <a href="https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html">https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html</a> for how to work with a DB instance in a VPC.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an Amazon Redshift cluster.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data from the Microsoft SQL Server database by using AWS DMS.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients over to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

References

- AWS DMS documentation
- Amazon Redshift documentation
- Amazon Redshift Pricing

Tutorials and videos

- Getting Started with AWS DMS
Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents

Created by Mark Szalkiewicz (AWS)

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Microsoft</td>
<td>AWS services: Amazon Redshift</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern outlines steps for migrating an on-premises Microsoft SQL Server source database to an Amazon Redshift target database using AWS Schema Conversion Tool (SCT) data extraction agents. An agent is an external program that is integrated with AWS SCT but performs data transformation elsewhere and interacts with other AWS services on your behalf.

**Prerequisites and limitations**

**Prerequisites**

- A Microsoft SQL Server source database used for the data warehouse workload in an on-premises data center
- An active AWS account

**Product versions**

- Microsoft SQL Server version 2008 or later. For the latest list of supported versions, see AWS SCT documentation.

**Architecture**

**technology stackSource**

- An on-premises Microsoft SQL Server database

**technology stackTarget**

- Amazon Redshift

Data migration architecture
Tools

- **AWS SCT** – *AWS Schema Conversion Tool* (AWS SCT) handles heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. When the source and target databases are very different from one another, you can use an AWS SCT agent to perform additional data transformation. For more information, see *Migrating Data from an On-Premises Data Warehouse to Amazon Redshift* in the AWS documentation.

Epics

**Prepare for migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions and engines.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity, storage features, network features).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose an application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
### Configure infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data using the AWS SCT data extraction agents.</td>
<td>For detailed information on using AWS SCT data extraction agents, see the links in the References and Help section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate applications

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the chosen application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Cut over to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
Related resources

References

- AWS SCT User Guide
- Using Data Extraction Agents
- Amazon Redshift Pricing

Tutorials and videos

- Getting Started with the AWS Schema Conversion Tool
- Getting Started with Amazon Redshift

Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
</tr>
</tbody>
</table>

AWS services: Amazon Redshift

Summary

This pattern walks you through the steps for migrating a Teradata database, used as a data warehouse in an on-premises data center, to an Amazon Redshift database. The pattern uses AWS Schema Conversion Tool (AWS SCT) data extraction agents. An agent is an external program that is integrated with AWS SCT but performs data transformation elsewhere and interacts with other AWS services on your behalf.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Teradata source database in an on-premises data center

Product versions

- Teradata version 13 and later. For the latest list of supported versions, see the AWS SCT documentation.

Architecture

Source technology stack

- On-premises Teradata database
Target technology stack

- Amazon Redshift cluster

Data migration architecture

Tools

- **AWS SCT** – **AWS Schema Conversion Tool** (AWS SCT) handles heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. When the source and target databases are very different from one another, you can use an AWS SCT agent to perform additional data transformation. For more information, see Migrating Data from an On-Premises Data Warehouse to Amazon Redshift in the AWS documentation.

Epics

Prepare for migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database</td>
<td>versions and engines.</td>
<td>DBA</td>
</tr>
<tr>
<td>Identify hardware requirements for the</td>
<td>target server instance.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>target server instance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify storage requirements</td>
<td>(storage type and capacity).</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity,</td>
<td>capacity, storage features, network features).</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>network features).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify network-access security</td>
<td>requirements for the source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose an application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
### Configure infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate data by using AWS SCT data extraction agents.</td>
<td>For detailed information on using AWS SCT data extraction agents, see the links in the References and Help section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate applications

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the chosen application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Cut over to the target Amazon Redshift database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Related resources

References

- AWS SCT User Guide
- Using Data Extraction Agents
- Amazon Redshift Pricing
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL (AWS Prescriptive Guidance)
- Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL (AWS Prescriptive Guidance)

Tutorials

- Getting Started with the AWS Schema Conversion Tool
- Getting Started with Amazon Redshift

Migrate an on-premises Vertica database to Amazon Redshift using AWS SCT data extraction agents

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
</tbody>
</table>

| AWS services: | Amazon Redshift |

Summary

This pattern provides guidance for migrating an on-premises Vertica database to an Amazon Redshift cluster using AWS Schema Conversion Tool (AWS SCT) data extraction agents. An agent is an external program that is integrated with AWS SCT but performs data transformation elsewhere and interacts with other AWS services on your behalf.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Vertica source database used for the data warehouse workload in an on-premises data center
- An Amazon Redshift target cluster

Product versions

- Vertica version 7.2.2 and later. For the latest list of supported versions, see the AWS SCT documentation.
Architecture

Source technology stack

- An on-premises Vertica database

Target technology stack

- An Amazon Redshift cluster

Data migration architecture

Tools

- AWS SCT - AWS Schema Conversion Tool (AWS SCT) handles heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. When the source and target databases are very different from one another, you can use an AWS SCT agent to perform additional data transformation. For more information, see Migrating Data from an On-Premises Data Warehouse to Amazon Redshift in the AWS documentation.

Epics

Prepare for migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type (capacity, storage features, network features).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Identify the network-access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose an application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Configure infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an Amazon Redshift cluster.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

**Migrate data**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data using the AWS SCT data extraction agents.</td>
<td>For detailed information on using AWS SCT data extraction agents, see the links in the References and Help section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Migrate applications**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the chosen application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Cut over to the target database**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Close the project**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
Migrate an on-premises MySQL database to Aurora MySQL

Created by Mark Szalkiewicz (AWS)

<table>
<thead>
<tr>
<th>R Type: Re-architect</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon Aurora MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: Production</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Open-source</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern explains how to migrate an on-premises MySQL source database to Amazon Aurora MySQL. It describes two options for migration: using AWS Database Migration Service (AWS DMS) or using native MySQL tools such as mysqldbcopy and mysqldump.

Prerequisites and limitations

Prerequisites

• An active AWS account
• A source MySQL database in an on-premises data center

Limitations

• Database size limit: 64 TB

Product versions

• MySQL versions 5.5, 5.6, 5.7, 8.0. For the latest list of supported versions, see MySQL on Amazon RDS in the AWS documentation. If you’re using AWS DMS, see also Using a MySQL-Compatible Database as a Target for AWS DMS for MySQL versions supported by AWS DMS.

Architecture

Source technology stack

• An on-premises MySQL database

Target technology stack

• Amazon Aurora MySQL

Target architecture
Data migration architecture

*Using AWS DMS:*

*Using native MySQL tools:*
AWS Prescriptive Guidance Patterns  
Migrate from on-premises MySQL to Aurora MySQL

Tools

- **AWS DMS - AWS Database Migration Service** (AWS DMS) supports several source and target databases. For information about MySQL source and target databases supported by AWS DMS, see Migrating MySQL-Compatible Databases to AWS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.
- **Native MySQL tools** - `mysqldbcopy` and `mysqldump`.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose proper instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
# Migrate from on-premises MySQL to Aurora MySQL

## Task
<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the network access security requirements for source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

## Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Create security groups.</td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Configure and start an Amazon RDS for MySQL DB instance.</td>
<td>SysAdmin</td>
<td></td>
</tr>
</tbody>
</table>

## Migrate data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native MySQL tools or third-party tools to migrate database objects and data.</td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>

## Migrate data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate data with AWS DMS.</td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>

## Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
<td></td>
</tr>
</tbody>
</table>

## Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients over to the new infrastructure.</td>
<td>DBA, SysAdmin, App owner</td>
<td></td>
</tr>
</tbody>
</table>
Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Related resources

References

- Migrating Your Databases to Amazon Aurora
- AWS DMS website
- AWS DMS documentation
- Amazon Aurora Pricing
- Amazon Virtual Private Cloud VPCs and Amazon RDS
- Amazon Aurora documentation

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon Aurora

Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible

Created by Rakesh Raghav (AWS) and anuradha chintha (AWS)

Environment: PoC or pilot  Source: Oracle  Target: Aurora PostgreSQL

R Type: Re-architect  Workload: Oracle  Technologies: Migration; Infrastructure; Databases

AWS services: Amazon S3; Amazon Aurora

Summary

As part of your migration journey from Oracle to Amazon Aurora PostgreSQL-Compatible Edition on the Amazon Web Services (AWS) Cloud, you might encounter multiple challenges. For example, migrating
code that relies on the Oracle UTL_FILE utility is always a challenge. In Oracle PL/SQL, the UTL_FILE package is used for file operations, such as read and write, in conjunction with the underlying operating system. The UTL_FILE utility works for both server and client machine systems.

Amazon Aurora PostgreSQL-Compatible is a managed database offering. Because of this, it isn't possible to access files on the database server. This pattern walks you through the integration of Amazon Simple Storage Service (Amazon S3) and Amazon Aurora PostgreSQL-Compatible to achieve a subset of UTL_FILE functionality. Using this integration, we can create and consume files without using third-party extract, transform, and load (ETL) tools or services.

Optionally, you can set up Amazon CloudWatch monitoring and Amazon SNS notifications.

We recommend thoroughly testing this solution before implementing it in a production environment.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Database Migration Service (AWS DMS) expertise
- Expertise in PL/pgSQL coding
- An Amazon Aurora PostgreSQL-Compatible cluster
- An S3 bucket

Limitations

This pattern doesn't provide the functionality to act as a replacement for the Oracle UTL_FILE utility. However, the steps and sample code can be enhanced further to achieve your database modernization goals.

Product versions

- Amazon Aurora PostgreSQL-Compatible Edition 11.9

Architecture

Target technology stack

- Amazon Aurora PostgreSQL-Compatible
- Amazon CloudWatch
- Amazon Simple Notification Service (Amazon SNS)
- Amazon S3

Target architecture

The following diagram shows a high-level representation of the solution.
1. Files are uploaded from the application into the S3 bucket.
2. The `aws_s3` extension accesses the data, using PL/pgSQL, and uploads the data to Aurora PostgreSQL-Compatible.

**Tools**

- **Amazon Aurora PostgreSQL-Compatible** – Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine. It combines the speed and reliability of high-end commercial databases with the cost-effectiveness of open-source databases.
- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With only one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
- **Amazon CloudWatch** – Amazon CloudWatch monitors Amazon S3 resources and use.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet. In this pattern, Amazon S3 provides a storage layer to receive and store files for consumption and transmission to and from the Aurora PostgreSQL-Compatible cluster.
- **aws_s3** – The `aws_s3` extension integrates Amazon S3 and Aurora PostgreSQL-Compatible.
- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients. In this pattern, Amazon SNS is used to send notifications.
- **pgAdmin** – pgAdmin is an open-source management tool for Postgres. pgAdmin 4 provides a graphical interface for creating, maintaining, and using database objects.

**Code**

To achieve the required functionality, the pattern creates multiple functions with naming similar to `UTL_FILE`. The Additional information section contains the code base for these functions.
In the code, replace `testaurorabucket` with the name of your test S3 bucket. Replace `us-east-1` with the AWS Region where your test S3 bucket is located.

**Epics**

**Integrate Amazon S3 and Aurora PostgreSQL-Compatible**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up IAM policies.</td>
<td>Create AWS Identity and Access Management (IAM) policies that grant access to the S3 bucket and objects in it. For the code, see the Additional information section.</td>
<td>AWS administrator, DBA</td>
</tr>
</tbody>
</table>
| Add Amazon S3 access roles to Aurora PostgreSQL. | Create two IAM roles: one role for read and one role for write access to Amazon S3. Attach the two roles to the Aurora PostgreSQL-Compatible cluster:  
  - One role for the S3Export feature  
  - One role for the S3Import feature  
  For more information, see the Aurora PostgreSQL-Compatible documentation on importing and exporting data to Amazon S3. | AWS administrator, DBA           |

**Set up the extensions in Aurora PostgreSQL-Compatible**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the aws_commons extension.</td>
<td>The <code>aws_commons</code> extension is a dependency of the <code>aws_s3</code> extension.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Create the aws_s3 extension.</td>
<td>The <code>aws_s3</code> extension interacts with Amazon S3.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

**Validate Amazon S3 and Aurora PostgreSQL-Compatible integration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test importing files from Amazon S3 into Aurora PostgreSQL.</td>
<td>To test importing files into Aurora PostgreSQL-Compatible, create a sample CSV file and upload it into the S3 bucket.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Create a table definition based on the CSV file, and load the file into the table by using the aws_s3.table_import_from_s3 function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test exporting files from Aurora PostgreSQL to Amazon S3.</td>
<td>To test exporting files from Aurora PostgreSQL-Compatible, create a test table, populate it with data, and then export the data by using the aws_s3.query_export_to_s3 function.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

To mimic the UTL_FILE utility, create wrapper functions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the utl_file_utility schema.</td>
<td>The schema keeps the wrapper functions together. To create the schema, run the following command.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td></td>
<td>CREATE SCHEMA utl_file_utility;</td>
<td></td>
</tr>
<tr>
<td>Create the file_type type.</td>
<td>To create the file_type type, use the following code.</td>
<td>DBA/Developer</td>
</tr>
<tr>
<td></td>
<td>CREATE TYPE utl_file_utility.file_type AS ( p_path character varying(30), p_file_name character varying );</td>
<td></td>
</tr>
<tr>
<td>Create the init function.</td>
<td>The init function initializes common variable such as bucket or region. For the code, see the Additional information section.</td>
<td>DBA/Developer</td>
</tr>
<tr>
<td>Create the wrapper functions.</td>
<td>Create the wrapper functions fopen, put_line, and fclose. For code, see the Additional information section.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>
Test the wrapper functions

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the wrapper functions in write mode.</td>
<td>To test the wrapper functions in write mode, use the code provided in the Additional information section.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Test the wrapper functions in append mode.</td>
<td>To test the wrapper functions in append mode, use the code provide in the Additional information section.</td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

Related resources

- Amazon S3 integration
- Amazon S3
- Aurora
- Amazon CloudWatch
- Amazon SNS

Additional information

Set up IAM policies

Create the following policies.

<table>
<thead>
<tr>
<th>Policy name</th>
<th>JSON</th>
</tr>
</thead>
</table>
| S3IntRead   | ```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3integrationtest",
            "Effect": "Allow",
            "Action": ["s3:GetObject", "s3:ListBucket"],
            "Resource": [
                "arn:aws:s3:::testaurorabucket/",
                "arn:aws:s3:::testaurorabucket"
            ]
        }
    ]
}
``` |
| S3IntWrite  | ```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3integrationtest",
            "Effect": "Allow",
            "Action": ["s3:GetObject", "s3:ListBucket"],
            "Resource": [
                "arn:aws:s3:::testaurorabucket/",
                "arn:aws:s3:::testaurorabucket"
            ]
        }
    ]
}
``` |
Create the init function

To initialize common variables, such as bucket or region, create the init function by using the following code.

```sql
CREATE OR REPLACE FUNCTION utl_file_utility.init(
 )
RETURNS void
LANGUAGE 'plpgsql'
COST 100
VOLATILE
AS $BODY$
BEGIN
  perform set_config
  ( format( '%s.%s','UTL_FILE_UTILITY', 'region' ),
    'us-east-1':text,
    false );
  perform set_config
  ( format( '%s.%s','UTL_FILE_UTILITY', 's3bucket' ),
    'testaurorabucket':text,
    false );
END;
$BODY$;
```

Create the wrapper functions

Create the fopen, put_line, and fclose wrapper functions.

**fopen**

```sql
CREATE OR REPLACE FUNCTION utl_file_utility.fopen( 
  p_file_name character varying,
  p_path character varying,
  p_mode character DEFAULT 'W'::bpchar,
  OUT p_file_type utl_file_utility.file_type)
RETURNS utl_file_utility.file_type
LANGUAGE 'plpgsql'
COST 100
VOLATILE
AS $BODY$
declare
  v_sql character varying;
  v_cnt_stat integer;
  v_cnt integer;
  v_tabname character varying;
```
BEGIN
/*initialize common variable */
PERFORM utl_file_utility.init();

v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 'region' ) );
v_bucket := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 's3bucket' ) );

/* set tablename*/
v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then length(p_file_name) else strpos(p_file_name,'.') - 1 end );
v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else concat_ws('/',p_path,p_file_name) end ;
raise notice 'v_bucket %, v_filewithpath % , v_region %', v_bucket,v_filewithpath, v_region;

/* APPEND MODE HANDLING; RETURN EXISTING FILE DETAILS IF PRESENT ELSE CREATE AN EMPTY FILE */
IF p_mode = 'A' THEN
v_sql := concat_ws('','create temp table if not exists ', v_tabname,' (col1 text)');
execute v_sql;
begin
PERFORM aws_s3.table_import_from_s3
( v_tabname,
'','DELIMITER AS ''#''
,aws_commons.create_s3_uri
( v_bucket,
 v_filewithpath ,
 v_region)
);
exception
when others then
raise notice 'File load issue ,%',sqlerrm;
raise;
end;
execute concat_ws('','select count(*) from ',v_tabname) into v_cnt;
IF v_cnt > 0 then
p_file_type.p_path := p_path;
p_file_type.p_file_name := p_file_name;
else
PERFORM aws_s3.query_export_to_s3('select ''''
,aws_commons.create_s3_uri(v_bucket, v_filewithpath, v_region)
);

p_file_type.p_path := p_path;
p_file_type.p_file_name := p_file_name;
end if;
v_sql := concat_ws('','drop table ', v_tabname);
execute v_sql;
ELSEIF p_mode = 'W' THEN
PERFORM aws_s3.query_export_to_s3('select ''''
,aws_commons.create_s3_uri(v_bucket, v_filewithpath, v_region)
);

p_file_type.p_path := p_path;
p_file_type.p_file_name := p_file_name;
END IF;
EXCEPTION
  when others then
    p_file_type.p_path := p_path;
    p_file_type.p_file_name := p_file_name;
    raise notice 'fopenerror,%',sqlerrm;
    raise;
END;
#BODY$;

put_line

CREATE OR REPLACE FUNCTION utl_file_utility.put_line(
  p_file_name character varying,
  p_path character varying,
  p_line text,
  p_flag character DEFAULT 'W'::bpchar)
RETURNS boolean
LANGUAGE 'plpgsql'
COST 100
VOLATILE
AS #BODY#

/**************************************************************************
* Write line, p_line in windows format to file, p_fp - with carriage return
* added before new line.
**************************************************************************/
declare
  v_sql varchar;
  v_ins_sql varchar;
  v_cnt INTEGER;
  v_filewithpath character varying;
  v_tabname character varying;
  v_bucket character varying;
  v_region character varying;
BEGIN
  PERFORM utl_file_utility.init();

  /* check if temp table already exist */
  v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
    length(p_file_name) else strpos(p_file_name,'.') - 1 end );
  v_sql := concat_ws('','select count(1) FROM pg_catalog.pg_class c LEFT JOIN
    pg_catalog.pg_namespace n ON n.oid = c.relnamespace where n.nspname like ''pg_temp_%''
    ,'' AND pg_catalog.pg_table_is_visible(c.oid) AND Upper(relname) =
    Upper( ''
    , v_tabname ,'' ) ');
  execute v_sql into v_cnt;
  IF v_cnt = 0 THEN
    v_sql := concat_ws('','create temp table ',v_tabname,' (col text)');
    execute v_sql;
    /* CHECK IF APPEND MODE */
    IF upper(p_flag) = 'A' THEN
      PERFORM utl_file_utility.init();
      v_region := current_setting( format( '%s.%s', 'UTL_FILEUTILITY', 'region' ) );
      v_bucket := current_setting( format( '%s.%s', 'UTL_FILEUTILITY',
        's3bucket' ) );
      /* set tablename*/
      v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
        concat_ws('/',p_path,p_file_name) end ;
    END IF;
  END IF;

  execute v_sql into v_cnt;
  IF v_cnt = 0 THEN
    v_sql := concat_ws('','create temp table ',v_tabname,' (col text)');
    execute v_sql;
    /* CHECK IF APPEND MODE */
    IF upper(p_flag) = 'A' THEN
      PERFORM utl_file_utility.init();
      v_region := current_setting( format( '%s.%s', 'UTL_FILEUTILITY', 'region' ) );
      v_bucket := current_setting( format( '%s.%s', 'UTL_FILEUTILITY',
        's3bucket' ) );
      /* set tablename*/
      v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
        concat_ws('/',p_path,p_file_name) end ;
    END IF;
  END IF;
*/
begin
    PERFORM aws_s3.table_import_from_s3
    ( v_tabname,
      'DELIMITER AS ''#''',
      aws_commons.create_s3_uri
      ( v_bucket,
        v_filewithpath,
        v_region   )
    );
exception
    when others then
      raise notice 'Error Message : %',sqlerrm;
      raise;
end;
END IF;
END IF;
/* INSERT INTO TEMP TABLE */
v_ins_sql := concat_ws('','insert into ',v_tabname,' values(''',p_line,''')');
execute v_ins_sql;
RETURN TRUE;
exception
    when others then
      raise notice 'Error Message : %',sqlerrm;
      raise;
END;
$BODY$;
fclose

CREATE OR REPLACE FUNCTION utl_file_utility.fclose(
    p_file_name character varying,
    p_path character varying)
RETURNS boolean
LANGUAGE 'plpgsql'
COST 100
VOLATILE
AS $BODY$
DECLARE
    v_filewithpath character varying;
    v_bucket character varying;
    v_region character varying;
    v_tabname character varying;
    v_sql character varying;
BEGIN
    PERFORM utl_file_utility.init();

    v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 'region' ) );
    v_bucket := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 's3bucket' ) );

    v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
        length(p_file_name) else strpos(p_file_name,'.') - 1 end );
    v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
        concat_ws('/',p_path,p_file_name) end ;

    raise notice 'v_bucket %, v_filewithpath %, v_region %', v_bucket,v_filewithpath,
        v_region ;

    /* exporting to s3 */
    perform aws_s3.query_export_to_s3
        (concat_ws('','select * from ',v_tabname,' order by ctid asc'),
        aws_commons.create_s3_uri(v_bucket, v_filewithpath, v_region) );
    v_sql := concat_ws('','drop table ', v_tabname);

    /* INSERT INTO TEMP TABLE */

    v_ins_sql := concat_ws('','insert into ',v_tabname,' values(''',p_line,''')');
execute v_ins_sql;
RETURN TRUE;
exception
    when others then
      raise notice 'Error Message : %',sqlerrm;
      raise;
END;
$BODY$;
execute v_sql;
RETURN TRUE;
EXCEPTION
when others then
raise notice 'error fclose %',sqerrm;
RAISE;
END;
#BODY$;

Test your setup and wrapper functions

Use the following anonymous code blocks to test your setup.

Test the write mode

The following code writes a file named s3inttest in the S3 bucket.

do $$
declare
l_file_name varchar := 's3inttest' ;
l_path varchar := 'integration_test' ;
l_mode char(1) := 'W';
l_fs utl_file_utility.file_type ;
l_status boolean;
begin
select * from
utl_file_utility.fopen( l_file_name, l_path , l_mode ) into l_fs ;
raise notice 'fopen : l_fs : %', l_fs;
select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket: for test purpose', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;
select * from utl_file_utility.fclose( l_file_name , l_path ) into l_status ;
raise notice 'fclose : l_status %', l_status;
end;
$$

Test the append mode

The following code appends lines onto the s3inttest file that was created in the previous test.

do $$
declare
l_file_name varchar := 's3inttest' ;
l_path varchar := 'integration_test' ;
l_mode char(1) := 'A';
l_fs utl_file_utility.file_type ;
l_status boolean;
begin
select * from
utl_file_utility.fopen( l_file_name, l_path , l_mode ) into l_fs ;
raise notice 'fopen : l_fs : %', l_fs;
select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket: for test purpose : append 1', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

```
select * from utl_file_utility.put_line( l_file_name, l_path , 'this is test file:in s3bucket : for test purpose : append 2', l_mode ) into l_status;
raise notice 'put_line : l_status %', l_status;

select * from utl_file_utility.fclose( l_file_name, l_path ) into l_status;
raise notice 'fclose : l_status %', l_status;
end;
##

Amazon SNS notifications

Optionally, you can set up Amazon CloudWatch monitoring and Amazon SNS notifications on the S3 bucket. For more information, see [Monitoring Amazon S3](https://docs.aws.amazon.com/AmazonS3/latest/userguide/MonitoringAmazonS3.html) and [Setting up Amazon SNS Notifications](https://docs.aws.amazon.com/AmazonSNS/latest/dg/NotificationTypes.html).

Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL

*Created by Venkatramana Chintha (AWS)*

<table>
<thead>
<tr>
<th><strong>R Type:</strong></th>
<th>Re-architect</th>
<th><strong>Source:</strong></th>
<th>Relational</th>
<th><strong>Target:</strong></th>
<th>Amazon Aurora PostgreSQL, Amazon RDS for PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Created by:</strong></td>
<td>AWS</td>
<td><strong>Environment:</strong></td>
<td>PoC or pilot</td>
<td><strong>Technologies:</strong></td>
<td>Databases; Migration</td>
</tr>
<tr>
<td><strong>Workload:</strong></td>
<td>Oracle</td>
<td><strong>AWS services:</strong></td>
<td>Amazon Aurora</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes a step-by-step approach to validate objects after migrating an Oracle database to Amazon Aurora PostgreSQL-Compatible Edition.

This pattern outlines usage scenarios and steps for database object validation; for more detailed information, see [Validating database objects after migration using AWS SCT and AWS DMS](https://aws.amazon.com/blogs/database/validating-database-objects-after-migration-using-aws-sct-and-aws-dms/) on the AWS Database blog.

Prerequisites and limitations

**Prerequisites:**

- An active AWS account.
- An on-premises Oracle database that was migrated to an Aurora PostgreSQL-Compatible database.
- This pattern uses the [query editor for Aurora Serverless DB clusters](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraServerlessGuide/userguide.html), which is available in the Amazon Relational Database Service (Amazon RDS) console. However, you can use this pattern with any other query editor.
**Limitations:**

- Oracle SYNONYM objects are not available in PostgreSQL but can be partially validated through **views** or **SET search_path** queries.
- The Amazon RDS query editor is available only in **certain AWS Regions and for certain MySQL and PostgreSQL versions**.

**Architecture**

![Architecture Diagram]

**Tools**

- **Amazon Aurora PostgreSQL-Compatible Edition** – Aurora PostgreSQL-Compatible is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases.
- **Amazon RDS** – Amazon Relational Database Service (Amazon RDS) makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.
- **Query Editor for Aurora Serverless** – Query editor helps you run SQL queries in the Amazon RDS console. You can run any valid SQL statement on the Aurora Serverless DB cluster, including data manipulation and data definition statements.

To validate the objects, use the full scripts in the "Object validation scripts" file in the “Attachments” section. Use the following table for reference.

<table>
<thead>
<tr>
<th>Oracle object</th>
<th>Script to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packages</td>
<td>Query 1</td>
</tr>
</tbody>
</table>
Epics

Validate objects in the source Oracle database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the “packages” validation query in the source Oracle database.</td>
<td>Download and open the “Object validation scripts” file from the “Attachments” section. Connect to the source Oracle database through your client program. Run the “Query 1” validations script from the “Object validation scripts” file. Important: Enter your Oracle user name instead of “your_schema” in the queries. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>
## Task | Description | Skills required
--- | --- | ---
Run the “tables” validation query. | Run the “Query 3” script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “views” validation query. | Run the “Query 5” script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “sequences” count validation. | Run the “Query 7” script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “triggers” validation query. | Run the “Query 9” script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “primary keys” validation query. | Run the “Query 11” script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “indexes” validation query. | Run the “Query 13” validation script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “check constraints” validation query. | Run the “Query 15” script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA
Run the “foreign keys” validation query. | Run the “Query 17” validation script from the “Object validation scripts” file. Make sure you record your query results. | Developer, DBA

### Validate objects in the target Aurora PostgreSQL-Compatible database

## Task | Description | Skills required
--- | --- | ---
Connect to the target Aurora PostgreSQL-Compatible database by using the query editor. | Sign in to the AWS Management Console and open the Amazon RDS console. In the upper-right corner, choose the AWS Region in which you created the Aurora PostgreSQL-Compatible database. In the navigation pane, choose “Databases,” and choose the target Aurora PostgreSQL-Compatible database. In “Actions,” choose | Developer, DBA
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the “packages” validation query</td>
<td>Run the “Query 2” script from the “Object validation scripts” file in the “Attachments” section. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “tables” validation query</td>
<td>Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the “Query 4” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “views” validation query</td>
<td>Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the “Query 6” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “sequences” count validation</td>
<td>Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the “Query 8” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “triggers” validation query</td>
<td>Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the “Query 10” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “primary keys” validation query</td>
<td>Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the “Query 12” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Rehost**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the “indexes” validation query.</td>
<td>Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the “Query 14” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “check constraints” validation query.</td>
<td>Run the “Query 16” script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Run the “foreign keys” validation query.</td>
<td>Run the “Query 18” validation script from the “Object validation scripts” file. Make sure you record your query results.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>

**Compare source and target database validation records**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare and validate both query results.</td>
<td>Compare the query results of the Oracle and Aurora PostgreSQL-Compatible databases to validate all objects. If they all match, then all objects have been successfully validated.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>

**Related resources**

- Validating database objects after a migration using AWS SCT and AWS DMS
- Amazon Aurora Features: PostgreSQL-Compatible Edition

**Attachments**

attachment.zip

---

**Rehost**

**Topics**

- Automate pre-workload ingestion activities for AWS Managed Services on Windows (p. 975)
- Create an approval process for firewall requests during a rehost migration to AWS (p. 981)
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account (p. 984)
- Migrate an on-premises Linux server to an Amazon EC2 Linux instance using AWS SMS (p. 991)
- Migrate a Microsoft Azure VM to Amazon EC2 using CloudEndure (p. 1000)
- Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud (p. 1007)
• Migrate an on-premises Go web application to AWS Elastic Beanstalk by using the binary method (p. 1014)
• Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP (p. 1017)
• Migrate small sets of data from on premises to Amazon S3 using AWS SFTP (p. 1023)
• Migrate from Oracle GlassFish to AWS Elastic Beanstalk (p. 1026)
• Migrate an on-premises Oracle database to Oracle on Amazon EC2 (p. 1031)
• Migrate an on-premises Oracle database to Amazon EC2 using CloudEndure (p. 1038)
• Migrate an on-premises Oracle database to Amazon EC2 using Oracle Data Pump (p. 1045)
• Migrate an on-premises PostgreSQL database to Amazon EC2 (p. 1048)
• Migrate an on-premises SAP ASE database to Amazon EC2 (p. 1054)
• Migrate SAP workloads to an SAP database on Amazon EC2 using CloudEndure (p. 1059)
• Migrate an on-premises Microsoft SQL Server database to Amazon EC2 (p. 1067)
• Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using CloudEndure (p. 1073)
• Migrate an on-premises MariaDB database to Amazon EC2 using MySQL tools (p. 1081)
• Migrate an on-premises MySQL database to Amazon EC2 (p. 1086)
• Rehost on-premises workloads in the AWS Cloud: migration checklist (p. 1092)

Automate pre-workload ingestion activities for AWS Managed Services on Windows

Created by Jacob Zhang (AWS), Calvin Yeh (AWS), and Dwayne Bordelon (AWS)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target: AWS Managed Services</th>
<th>R Type: Rehost</th>
<th>Technologies: Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CloudFormation; AWS Managed Services; AWS Systems Manager; Amazon S3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

On the Amazon Web Services (AWS) Cloud, AWS Managed Services (AMS) uses AMS workload ingest (WIGS) to move existing workloads into an AMS managed VPC. This pattern describes a solution to automate common pre-workload ingestion activities, such as upgrading .NET and Windows PowerShell and running Windows WIGS pre-ingestion validation maintained by AMS. The pattern also provides a unified user interface for the run results. It packages an AWS Systems Manager Command document, which performs the pre-ingestion activities, into an AWS CloudFormation template. The template can be deployed repeatedly without requiring access to Systems Manager itself or conflicting with automations from AMS.

Business background

Migrations to AMS require the provision of new Amazon Elastic Compute Cloud (Amazon EC2) instances using AMS managed Amazon Machine Images (AMIs) that include AMS components. Any workloads or applications running in existing data centers must be redeployed to fresh EC2 instances launched from
these AMS AMIs. To avoid the potentially massive amount of manual work during the process, the AMS team built the AMS workload ingest (WIGS) workflow to onboard your custom images to AMS.

Windows instances must satisfy a few prerequisites before the WIGS process takes place. Windows PowerShell scripts are usually used to perform the necessary preparations (WIGS prep) and check if the instances are ready for WIGs (WIGS pre-ingestion validation). The prep and validation processes require an engineer to spend 15–30 minutes on each server, manually logging in and running the scripts one by one.

**Business driver**

Traditionally, using Systems Manager, you can automate operational tasks such as running Windows PowerShell scripts. However, because of elevated risks and frequent conflicts between automations from AMS and those from the users, AMS does not usually grant its users access to Systems Manager.

For mass migrations using AWS Application Migration Service (AWS MGN), Windows PowerShell scripts in the `C:\Program Files (x86)\AWS Replication Agent\post_launch` folder usually run automatically when a test or cutover instance is launched. However, these scripts, if run immediately during an instance launch, frequently conflict with automations from AMS. As a result, the launch might fail without providing the run results that you need to troubleshoot the failure.

This pattern tackles these problems and provides a working automated solution.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account with AMS onboarding completed.
- An Amazon Simple Storage Service (Amazon S3) bucket in the AWS account. If there is no S3 bucket over which you have control in the account, use a request for change (RFC) to create one.
- The PreWIGs_CFN.json template downloaded from the ams-auto-prewigs-windows repository.
- A server to which you apply this pattern must meet the following requirements:
  - Run Windows Server 2012 or later.
  - Be launched or ready to launch in the sandbox VPC migration subnet.
  - Have an AWS Systems Manager Agent (SSM Agent) installed.
  - Have an AWS Identity and Access Management (IAM) instance profile attached. The instance profile must have permissions to download files from S3 buckets in the same AWS account. An instance profile that satisfies the above-mentioned requirement is usually already established during earlier setups of a migration.
  - Be viewable from AWS Systems Manager Fleet Manager.

**Limitations**

- Pre-WIGS activities vary depending on your environment and business requirements. You might need to make minor modifications to this pattern to suit your specific needs.

**Product versions**


**Architecture**

The architecture diagram shows the following:
1. A sandbox VPC with a migration subnet containing servers that have not been prepped.
2. The S3 bucket that stores scripts that are used by the CloudFormation template.
3. The CloudFormation template deploys the Systems Manager Command document. The process iterates until the steps complete.
4. The instances are prepped and RFCs for WIGS are made.
5. In the AMS managed VPC, the AMS managed subnet contains the servers after workload ingestion.

**How it works**

- This pattern is packaged into an AWS CloudFormation template that allows infrastructure as code (IaC) repeatable deployments. You need to deploy this template only one time for each AWS account that requires this automation.
- The automation is applied to all EC2 instances with a tag key `AutoPreWIGs` in the AWS account where this pattern is deployed. The first time an Amazon EC2 Windows instance with the tag key `AutoPreWIGs` starts, the automation performs the following tasks.
  1. Upgrades Windows PowerShell to version 5.1 and .NET to version 4.5.2. The instance might reboot several times, depending on its existing Windows PowerShell and .NET versions. After each reboot, the upgrades continue until they are complete. This step uses embedded code in the CloudFormation template modified from a Windows PowerShell script, as well as specific Systems Manager guidance on server reboots.
  2. Downloads from Amazon S3 and runs a Windows PowerShell script that you have customized to prepare the Amazon EC2 Windows instance for WIGS. For more information see the Epics section.
  3. Installs the Windows WIGS pre-ingestion validation PowerShell module from AWS.
  4. Runs the Windows WIGS pre-ingestion validation and makes the results viewable in Systems Manager State Manager.
Tools

- **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your AWS resources. You can use a template that describes all the AWS resources that you want and their dependencies, so that you can launch and configure those resources as a stack. This pattern uses a CloudFormation template to automate deployment of the resources in this pattern.

- **AWS Managed Services** – AWS Managed Services (AMS) is an enterprise service that provides ongoing management of your AWS infrastructure. Changes made to the infrastructure in an AMS environment must be made through an RFC.

- **AWS Systems Manager** – AWS Systems Manager (formerly known as SSM) is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. This pattern uses Systems Manager to run and view the run results of the pre-WIGS activities.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This pattern uses Amazon S3 to store the CloudFormation template and a Windows PowerShell script that is downloaded.

Epics

Create a custom Windows PowerShell script to automate additional tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform necessary changes to the servers based on business needs.</td>
<td>If you need changes automatically applied to your servers before their ingestions, create a Windows PowerShell script named ingestion-prep.ps1. <strong>Important</strong>: The script must not contain instructions to reboot the server, and it must not require administrator privileges.</td>
<td>PowerShell scripting</td>
</tr>
<tr>
<td>Remove software that isn’t supported by AMS.</td>
<td>AMS requires certain software, such as antivirus applications and VMware Tools removed before WIGS runs. Include the uninstallation in the ingestion-prep.ps1 script. For more information about software that isn’t supported, see the AWS documentation.</td>
<td>PowerShell scripting</td>
</tr>
</tbody>
</table>
**Upload the CloudFormation template and the optional Windows PowerShell script to Amazon S3**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a folder in S3.</td>
<td>In an S3 bucket in the same AWS account where you deploy this pattern, create a folder.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Upload the scripts.</td>
<td>Upload the <code>PreWIGs_CFN.json</code> CloudFormation template and the <code>ingestion-prep.ps1</code> Windows PowerShell script, which you created in the previous epic, to the Amazon S3 folder.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

**Deploy the CloudFormation stack**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the change type.</td>
<td>Navigate to the AMS console to create an RFC. Use the <strong>Create Stack from CloudFormation (CFN) Template</strong> change type.</td>
<td>General AMS</td>
</tr>
<tr>
<td>Set run parameters for the path to the CloudFormation template.</td>
<td>In the <strong>Execution configuration</strong> section, expand <strong>Additional configuration</strong>. In the <strong>CloudFormation template S3 endpoint</strong> box, paste the URL to the CloudFormation template.</td>
<td>General AMS</td>
</tr>
<tr>
<td>Specify the path to the Amazon S3 folder.</td>
<td>Under <strong>Parameters</strong>, use <strong>ScriptSource</strong> as the <strong>Name</strong>. For <strong>Value</strong>, enter the path to the S3 folder that contains the Windows PowerShell scripts. Make sure that you use the <code>https://xxx</code> URL instead of the <code>s3://xxx</code> URI, and include the <code>/</code> at the end.</td>
<td>General AMS</td>
</tr>
<tr>
<td>Deploy the stack.</td>
<td>To deploy the stack, choose <strong>Create</strong>.</td>
<td>General AMS</td>
</tr>
<tr>
<td>Escalate the RFC to AMS Ops.</td>
<td>The RFC must be implemented manually by the AMS Ops team because it uses Systems Manager to deploy resources with and requires a security review. As soon as you create the RFC, it will be automatically rejected by the system. Choose the RFC, and add a correspondence to the</td>
<td>General AMS</td>
</tr>
</tbody>
</table>
### Apply the automation to the instances

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add the AutoPreWIGs tag to instances.</td>
<td>Note IDs of all instances to which you want to apply this automation and wait at least 30 minutes for the instance to finish the automations implemented by AMS. Submit an automated RFC to add the tag with <strong>AutoPreWIGs</strong> as the key and any string, such as 1, as the value. The automation will be applied a few minutes after you add the tag.</td>
<td>General AMS</td>
</tr>
<tr>
<td>Verify the automation results.</td>
<td>Open the Systems Manager console, and choose <strong>State Manager</strong>. Choose the <strong>Association ID</strong> with the name <strong>AMS-PreWIG-Prep-and-Validation-Association</strong>. On the <strong>Execution history</strong> tab, you can see the results of the automation.</td>
<td>General AMS</td>
</tr>
<tr>
<td>Fix any errors.</td>
<td>If the automation fails, choose its <strong>Execution ID</strong>. You can see the run results for each EC2 instance. To see the details for each step of the automation, choose <strong>Output</strong>. If a particular step fails, use the information in the <strong>Output</strong> and the <strong>Error</strong> sections to diagnose the problem.</td>
<td>Migration engineer</td>
</tr>
<tr>
<td>Remove the AutoPreWIGs tag.</td>
<td><strong>Important:</strong> After you fix the errors, if any, submit an automated RFC to remove the <strong>AutoPreWIGs</strong> tag. WIGS will fail if you don't remove the tag.</td>
<td>General AMS</td>
</tr>
</tbody>
</table>
Create an approval process for firewall requests during a rehost migration to AWS

Created by Srikanth Rangavajhala (AWS)

**R Type:** Rehost  **Environment:** Production  **Technologies:** Migration

**Source:** On premises  **Target:** AWS Cloud

**Summary**

If you want to use CloudEndure Migration or the CloudEndure Migration factory (CEMF) solution for a rehost migration to the Amazon Web Services (AWS) Cloud, one of the prerequisites is that you must keep TCP ports 443 and 1500 open. Typically, opening these firewall ports requires approval from your information security (InfoSec) team.

This pattern outlines the process to obtain a firewall request approval from an InfoSec team during a rehost migration to the AWS Cloud. You can use this process to avoid rejections of your firewall request by the InfoSec team, which can become expensive and time consuming. The firewall request process has two review and approval steps between AWS migration consultants and leads who work with your InfoSec and application teams to open the firewall ports.

This pattern assumes that you are planning a rehost migration with AWS consultants or migration specialists from your organization. The pattern can be used if your organization doesn’t have a firewall approval process or firewall request form. For more information about this, see the Limitations section.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

**Prerequisites and limitations**

**Prerequisites**

---

**Ingest the prepared instances**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit RFCs for WIGS.</td>
<td>Now that the instances are ready for workload ingestion, submit the RFCs for WIGS.</td>
<td>General AMS</td>
</tr>
</tbody>
</table>

**Related resources**

- AMS Workload Ingest (WIGS)
- Migrating workloads: Windows pre-ingestion validation
- AWS Application Migration Service quick start guide
- Getting started with AWS CloudFormation
- Setting up AWS Systems Manager
Create an approval process for firewall requests during a migration

- The pattern assumes that you are planning a rehost migration with AWS consultants or migration specialists from your organization.
- The required port and IP information to migrate the stack.
- Existing and future state architecture diagrams.
- Firewall information about the on-premises and destination infrastructure, ports, and zone-to-zone traffic flow.
- A firewall request review checklist (attached).
- A firewall request document, configured according to your organization's requirements.
- A contact list for the firewall reviewers and approvers. The following table describes the roles in this process.

<table>
<thead>
<tr>
<th>Firewall request submitter</th>
<th>AWS migration specialist or consultant. The firewall request submitter can also be a migration specialist from your organization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall request reviewer</td>
<td>Typically, this is the single point of contact (SPOC) from AWS.</td>
</tr>
<tr>
<td>Firewall request approver</td>
<td>An InfoSec team member.</td>
</tr>
</tbody>
</table>

Limitations

- This pattern describes a generic firewall request approval process. Requirements can vary for individual organizations.
- Make sure that you track changes to your firewall request document.

The following table shows the use cases for this pattern.

<table>
<thead>
<tr>
<th>Does your organization have an existing firewall approval process?</th>
<th>Does your organization have an existing firewall request form?</th>
<th>Suggested action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Collaborate with AWS consultants or your migration specialists to implement your organization's process.</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Use this pattern's firewall approval process. Use either an AWS consultant or a migration specialist from your organization to submit the firewall request form.</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Use this pattern's firewall approval process. Use either an AWS consultant or a migration specialist from your organization to submit the firewall request form.</td>
</tr>
</tbody>
</table>
### Architecture

The following diagram shows the steps for the firewall request approval process.

#### Tools

You can use scanner tools such as **Palo Alto Networks** or **SolarWinds** to analyze and validate firewalls and IP addresses.

#### Epics

**Analyze the firewall request**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze the ports and IP addresses.</td>
<td>The firewall request submitter completes an initial analysis to understand the required firewall ports and IP addresses. After this is complete, they request that your InfoSec team opens the required ports and maps the IP addresses.</td>
<td>AWS Cloud engineer, CloudEndure associate, migration specialist</td>
</tr>
</tbody>
</table>

**Validate the firewall request**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the firewall information.</td>
<td>The AWS Cloud engineer or CloudEndure associate schedules a meeting with your InfoSec team. During this meeting, they examine and validate the firewall request information. Typically, the firewall request submitter is the same person</td>
<td>AWS Cloud engineer, CloudEndure associate, migration specialist</td>
</tr>
</tbody>
</table>
# AWS Prescriptive Guidance Patterns

## Ingest EC2 Windows instances into an AMS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>as the firewall requester.</td>
<td>This validation phase can become iterative based on the feedback given by the approver if anything is observed or recommended.</td>
<td>AWS Cloud engineer, CloudEndure associate, migration specialist</td>
</tr>
<tr>
<td>Update the firewall request</td>
<td>After the InfoSec team shares their feedback, the firewall request document is edited, saved, and re-uploaded. This document is updated after each iteration.</td>
<td>document.</td>
</tr>
<tr>
<td>document.</td>
<td>We recommend that you store this document in a version-controlled storage folder. This means that all changes are tracked and correctly applied.</td>
<td></td>
</tr>
</tbody>
</table>

**Submit the firewall request**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit the firewall request.</td>
<td>After the firewall request approver has approved the request, the AWS Cloud engineer submits the firewall request. The request specifies the ports that must be open and IP addresses that are required to map and update the AWS account.</td>
<td>AWS Cloud engineer, CloudEndure associate, migration specialist</td>
</tr>
<tr>
<td></td>
<td>You can make suggestions or provide feedback after the firewall request is submitted. We recommend that you automate this feedback process and send any edits through a defined workflow mechanism.</td>
<td></td>
</tr>
</tbody>
</table>

**Attachments**

attachment.zip

**Ingest and migrate EC2 Windows instances into an AWS Managed Services account**

*Created by Anil Kunapareddy (AWS) and Venkatramana Chinthia (AWS)*
Ingest EC2 Windows instances into an AMS account

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Source:</th>
<th>VPC in AWS Cloud</th>
<th>Target:</th>
<th>VPC Managed by AWS Managed Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Rehost</td>
<td>Workload:</td>
<td>Microsoft</td>
<td>Technologies: Migration; Operations; Security, identity, compliance; Cloud-native</td>
<td></td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS Managed Services</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern explains the step-by-step process of migrating and ingesting Amazon Elastic Compute Cloud (Amazon EC2) Windows instances into an Amazon Web Services (AWS) Managed Services (AMS) account. AMS can help you manage the instance more efficiently and securely. AMS provides operational flexibility, enhances security and compliance, and helps you optimize capacity and reduce costs.

This pattern starts with an EC2 Windows instance that you have migrated to a staging subnet in your AMS account. A variety of migration services and tools are available to perform this task, such as AWS Application Migration Service or AWS Server Migration Service (SMS).

To make a change to your AMS-managed environment, you create and submit a request for change (RFC) for a particular operation or action. Using an AMS workload ingest (WIGS) RFC, you ingest the instance into the AMS account and create a custom Amazon Machine Image (AMI). You then create the AMS-managed EC2 instance by submitting another RFC to create an EC2 stack. For more information, see AMS Workload Ingest in the AMS documentation.

Prerequisites and limitations

Prerequisites

- An active, AMS-managed AWS account
- An existing landing zone
- An AWS Identity and Access Management (IAM) user with permissions to make changes in the AMS-managed VPC
- An Amazon EC2 Windows instance in a staging subnet in your AMS account
- Completion of the general prerequisites for migrating workloads using AMS WIGS
- Completion of the Windows prerequisites for migrating workloads using AMS WIGS

Limitations

- This pattern is for EC2 instances operating Windows Server. This pattern doesn’t apply to instances running other operating systems, such as Linux.

Architecture

Source technology stack

Amazon EC2 Windows instance in a staging subnet in your AMS account

Target technology stack
Ingest EC2 Windows instances into an AMS account

Amazon EC2 Windows instance managed by AWS Managed Services (AMS)

**Target architecture**

**Tools**

**AWS services**

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AMS** – AWS Managed Services (AMS) helps you to operate your cloud-based infrastructure more efficiently and securely by providing ongoing management of your Amazon Web Services (AWS) infrastructure, including monitoring, incident management, security guidance, patch support, and backup for AWS workloads.

**Other services**

- **PowerShell** – PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.

**Epics**

**Configure settings on the instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Change the DNS Client settings. | 1. On the source EC2 instance, open Command Prompt as an administrator, type `gpedit.msc`, and then press `Enter`.  
2. In the Local Group Policy Editor, navigate to *Computer Configuration*, *Administrative Templates*, *Network*, *DNS Client*.  
3. For **Primary DNS suffix**, choose **Not configured**.  
4. For **Primary DNS suffix devolution**, choose **Not configured**. | Migration engineer |

| Change the Windows Update settings. | 1. In the Local Group Policy Editor, navigate to *Computer Configuration*, *Administrative Templates*, *Windows Components*, *Windows Update*.  
2. For **Specify intranet Microsoft update service location**, choose **Not configured**.  
3. For **Configure Automatic Updates**, choose **Not configured**.  
4. For **Automatic Updates detection frequency**, choose **Not configured**.  
5. Close the Local Group Policy Editor. | Migration engineer |
### AWS Prescriptive Guidance Patterns

#### Ingest EC2 Windows instances into an AMS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Enable the firewall.        | 1. On the source EC2 instance, open Command Prompt as an administrator, type `services.msc`, and then press **Enter**.  
2. In Windows Services, enable **Firewall**.  
3. Close Windows Services. | Migration engineer         |

#### Prepare the instance for AMS WIGS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Clean up and prepare the instance. | 1. Using a bastion host and local credentials, create a Remote Desktop Protocol (RDP) connection to the EC2 instance in the staging subnet.  
2. Remove all legacy software, antivirus software, and backup solutions that aren't required in AMS. | Migration engineer     |
| Repair the sppnp.dll file.  | 1. Go to `C:\Windows \System32\sppnp.dll`.  
2. Rename `sppnp.dll` to `sppnp_old.dll`.  
3. Using PowerShell and administrator credentials, enter the following commands:  
```
dism /online /cleanup-image /restorehealth  
sfc /scannow  
```
4. Restart the EC2 Windows instance. | Migration engineer          |
| Run the pre-WIG validation script. | 1. Download the Windows WIGS Pre-ingestion Validation zip file (`windows-prewings-validation.zip`) from [Migrating workloads: Windows pre-ingestion validation](https://aws.amazon.com/ams/docs/ams-migration-workloads-windows) in the AMS documentation.  
2. Run the Windows pre-WIG validation script and verify the results. | Migration engineer       |
### AWS Prescriptive Guidance Patterns

**Ingest EC2 Windows instances into an AMS account**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
</tbody>
</table>
| Create the failsafe AMI. | After the pre-WIG validation passes, create a pre-ingestion AMI as follows:  
1. Choose **Deployment**, **Advanced stack components**, **AMI**, **Create**.  
2. During creation, add a tag **Key=Name**, **Value=APPLICATION-ID_IngestReady**.  
3. Wait until AMI is created before proceeding. | Migration engineer |
| Ingest and validate the instance | | |
| Submit the RFC to create the workload ingest stack. | Submit a request for change (RFC) to start the AMS WIGS. For instructions, see Workload Ingest Stack: Creating in the AMS documentation. This starts the workload ingestion and installs all the software required by AMS, including backup tools, Amazon EC2 management software, and antivirus software. | Migration engineer |
| Validate successful migration. | After the workload ingestion is complete, you can see the AMS-managed instance and AMS-ingested AMI.  
1. Log in to the AMS-managed instance with domain credentials.  
2. Validate the domain joining as follows:  
   a. In Windows Explorer, right-click **This PC**, and then choose **Properties**.  
   b. In the Device Specification section, confirm that the | Migration engineer |
AWS Prescriptive Guidance Patterns
Ingest EC2 Windows instances into an AMS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>domain appears in the <strong>Full device name</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Validate the source and target disk drives.</td>
<td></td>
</tr>
</tbody>
</table>

Launch the instance in the target AMS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Submit the RFC to create an EC2 stack.</td>
<td>Migration engineer</td>
</tr>
<tr>
<td></td>
<td>1. Using the AMS-ingested AMI of the Windows instance, prepare an RFC for an EC2 stack according to the instructions in Create EC2 stack instance in AMS documentation. In the EC2 stack RFC, provide all the parameters, including the server name, tags, target VPC, target subnet, instance type, target security groups, ingestion AMI, and role.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Submit the RFC for the EC2 stack, and then wait for the instance to be successfully created.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

**Marketing resources**
- AWS Managed Services
- AWS Managed Services FAQs
- AWS Managed Services Resources
- AWS Managed Services Features

**AWS Prescriptive Guidance**
- Automate pre-workload ingestion activities for AWS Managed Services on Windows
- Automatically create an RFC in AMS using Python

**AMS documentation**
- AMS Workload Ingest
- How Migration Changes Your Resource
- Migrating Workloads: Standard Process
Migrate an on-premises Linux server to an Amazon EC2 Linux instance using AWS SMS

Created by Mark Szalkiewicz (AWS)

R Type: Rehost  Source: Operating Systems  Target: Amazon EC2

Created by: AWS  Environment: PoC or pilot  Technologies: Websites & web apps; Operating systems; Migration

Workload: Open-source

Summary

This pattern walks you through the steps for rehosting ("lift-and-shift") an on-premises Linux server by migrating it to a supported Amazon Elastic Compute Cloud (Amazon EC2) Linux instance, using AWS Server Migration Service (AWS SMS).

For information about migrating databases (such as Oracle) that are associated with the servers and applications you are migrating, see the database migration patterns in this catalog.

Prerequisites and limitations

Prerequisites

General requirements:

- You must have an active AWS account.
- Disable any antivirus or intrusion detection software on the virtual machine (VM).
- Disconnect any CD-ROM drives (virtual or physical) connected to the VM.

Linux VMs:

- Enable Secure Shell (SSH) for remote access, and enable host firewall access to SSH.
- Make sure that your Linux VM uses GRUB (GRUB legacy) or GRUB 2 as its bootloader.
- The root volume of your Linux VM must use one of the following file systems: ext2, ext3, ext4, Btrfs, JFS, XFS.

Hardware requirements:

- VMware vCenter version 5.5 or higher (validated up to 6.5)
- ESXi 5.1 or higher (validated up to 6.5)
- Minimum 4 GB RAM
- Minimum available disk storage of 20 GB (thin-provisioned) or 250 GB (thick-provisioned)

Software requirements:

- If VMware vCenter Server is configured to use a non-default port, enter the vCenter hostname and port, separated by a colon (for example, HOSTNAME: PORT or IP: PORT) in the vCenter Service Account page in Connector setup.
AWS Prescriptive Guidance Patterns
Migrate a Linux server to Amazon EC2

- Make sure that your system supports the following network services (you might need to reconfigure your firewall to permit stateful outbound connections from the connector to these services):
  - Domain Name System (DNS) - Allow the connector to initiate connections to port 53 for name resolution.
  - HTTPS on vCenter - Allow the connector to initiate secure web connections to port 443 of vCenter. You can also configure a non-default port at your discretion.
  - HTTPS on ESXi - Allow the connector to initiate secure web connections to port 443 of the ESXi hosts that contain the VMs you intend to migrate.
  - Internet Control Message Protocol (ICMP) - Allow the connector to initiate connections using ICMP.
  - Network Time Protocol (NTP) - The connector must be able to reach a time server on port 123.
- Allow outbound connections from the connector to the following URL ranges:
  - *.amazonaws.com
  - *.aws.amazon.com
  - *.ntp.org (optional; used only to validate that connector time is in sync with NTP)

For additional information, see AWS SMS Requirements in the AWS SMS documentation.

**Linux licensing options:**

When you create a new replication job, the AWS SMS console provides a License type option that provides these values:

- **Auto** (default) - Detects the source system operating system (OS) and applies the appropriate license to the migrated VM.
- **AWS** - Replaces the source system license with an AWS license, if appropriate, on the migrated VM.
- **BYOL** - Retains the source system license, if appropriate, on the migrated VM.

If you choose a license type that is incompatible with your VM, the replication job fails with an error message.

Linux operating systems support only BYOL licenses, so if you choose Auto (the default), AWS SMS will use a BYOL license.

Migrated Red Hat Enterprise Linux (RHEL) VMs must use Cloud Access (BYOL) licenses. For more information, see Red Hat Cloud Access on the Red Hat website.

Migrated SUSE Linux Enterprise Server (SLES) VMs must use SUSE Public Cloud Program (BYOS) licenses. For more information, see SUSE Public Cloud Program—Bring Your Own Subscription on the SUSE website.

**Limitations**

- The migration source must be a VMware server. The use of AWS SMS is limited as follows:
  - 50 concurrent VM migrations per account
  - 90 days of service usage per VM (not per account), beginning with the initial replication of a VM
- The target operating system must be supported by Amazon EC2. Supported systems include RHEL, SLES, CentOS, Ubuntu, Oracle Linux, Fedora, and Debian Linux. RHEL 6.0 is not supported. For a complete list, see Amazon EC2 FAQs.
- Linux/Unix (64-bit) volume types and file systems: MBR-partitioned volumes that are formatted using the ext2, ext3, ext4, Btrfs, JFS, or XFS file system.
- For operating system licensing policies and limitations, see AWS SMS Requirements.
- AWS SMS partially supports vMotion, Storage vMotion, and other features based on virtual machine migration, with the following limitations:
• Migrating a virtual machine to a new ESXi host or datastore after one replication run ends, and before the next replication run begins, is supported as long the Server Migration Connector’s vCenter service account has sufficient permissions on the destination ESXi host, datastores, and data center, and on the virtual machine itself at the new location.

• Migrating a virtual machine to a new ESXi host, datastore, or data center while a replication run is active—that is, while a virtual machine upload is in progress—is not supported.

• Cross vCenter vMotion is not supported for use with AWS SMS.

Architecture

Source technology stack

On-premises application/web servers:

• Red Hat Enterprise Linux (RHEL)
• SUSE Linux Enterprise Server (SLES)
• CentOS
• Ubuntu
• Oracle Linux
• Fedora
• Debian Linux

Target technology stack

EC2 instances running the following operating systems (supporting migrations from the same system on premises):

• RHEL
• SLES
• CentOS
• Ubuntu
• Oracle Linux
• Fedora
• Debian Linux

Target architecture
Tools

AWS SMS - AWS Server Migration Service (AWS SMS) automates the migration of virtual machines to AWS. It provides the following features:

- Automated migration of an on-premises VMware server fleet to the AWS Cloud.
- Incremental replication of VMs to Amazon Machine Images (AMIs). AWS SMS transfers only the delta to the cloud, so you can test small changes iteratively and save network bandwidth.
- Multi-server migration to AWS, including scheduling and periodic replication for a group of servers.

To migrate VMs, you need to set up the AWS Server Migration Connector in your on-premises virtualization environment. To deploy the Server Migration Connector, choose Get Started from the AWS SMS console.

- Schedule the connector download. The connector is an OVA image that's nearly 10 GB in size, so it could take a long time to download, depending on the speed of your internet connection. We recommend that you schedule an appropriate time to download the appliance, based on your connection speed and traffic priority. The downloaded OVA should be made available to vCenter for later deployment. For step-by-step instructions, see the AWS SMS documentation.

- Create a vCenter account. The connector requires a vCenter service account with (at the minimum) Create Snapshot and Delete Snapshot permissions on VMs that you plan to migrate to AWS. We recommend that you create a service account that has access only to the vCenter data centers and ESXi hosts, folders, and datastores that you plan to migrate.

- Create an IAM user. The connector requires AWS Identity and Access Management (IAM) credentials to communicate with AWS. Use the following AWS CloudFormation template to create an IAM user with the ServerMigrationConnector AWS managed policy attached. After you create the IAM user, navigate to the IAM console and search for the IAM user ServerMigrationServiceConnector. Create an IAM access key for this user, and then download and save the access key securely. The credential will be used in the connector configuration.

```yaml
AWSTemplateFormatVersion: '2010-09-09' Description: |

# Copyright 2017 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# Licensed under the Amazon Software License (the "License").
# You may not use this file except in compliance with the
# License. A copy of the License is located at
# http://aws.amazon.com/asl/
# or in the "license" file accompanying this file. This file is
# distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
# CONDITIONS OF ANY KIND, express or implied. See the License
# for the specific language governing permissions and
# limitations under the License.
#
# IAM user for AWS Server Migration Service (SMS) Connector

Resources:
SmsConnector:
  Properties:
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/ServerMigrationConnector
    UserName: ServerMigrationServiceConnector

Type: AWS::IAM:: ser
```
• **Create an IAM role.** AWS SMS requires a service role to access AWS resources. Use the following AWS CloudFormation template to create a service role.

```yaml
AWSTemplateFormatVersion: '2010-09-09' Description: |

# Copyright 2017 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Amazon Software License (the "License").
# You may not use this file except in compliance with the
# License. A copy of the License is located at
#
# http://aws.amazon.com/asl/
#
# or in the "license" file accompanying this file. This file is
# distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
# CONDITIONS OF ANY KIND, express or implied. See the License
# for the specific language governing permissions and
# limitations under the License.
#
# IAM Role for AWS Server Migration Service (SMS)

Resources:
smsRole:
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Sid: SmsServiceRoleAssumeRolePolicyDocument
          Effect: Allow
          Principal:
            Service: sms.amazonaws.com
          Action: sts:AssumeRole
          Condition:
            StringEquals:
              sts:ExternalId: sms
          RoleName: sms
          Type: AWS::IAM::Role
    policy:
      Properties:
        Description: IAMManagedPolicyForSmsServiceRole
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Sid: SmsServiceRolePolicyDocument
              Effect: Allow
              Resource: "*"
              Action:
                - ec2:ModifySnapshotAttribute
                - ec2:CopySnapshot
                - ec2:CopyImage
                - ec2:DescribeImages
                - ec2:DescribeSnapshots
                - ec2:DeleteSnapshot
                - ec2:DeregisterImage
                - ec2:CreateTags
                - ec2:DeleteTags
              Roles:
                - Ref: smsRole
      Type: AWS::IAM::ManagedPolicy
```

995
• **Plan your firewall configurations.** When the connector operates, it needs to connect to in-scope vCenter and ESXi hosts on the HTTPS port (443), NTP servers, DNS servers, and AWS endpoints that have the patterns *.amazonaws.com and *.aws.amazon.com. Please plan your network configurations, including IP allocation, firewall change, and monitoring updates. You might need to submit change requests to get approvals for these changes in your organization.

• **Prepare storage space and VM resources.** When deploying the connector, the appliance requires 20 GB of disk space and 8 GB of RAM at the minimum. The disk in each connector may grow up to 250 GB if you choose thin provisioning, so make sure that your VMware storage has this capacity.

• **Install additional connectors if necessary.** If you're migrating multiple data centers, we recommend that you install at least one connector in each data center. For step-by-step installation instructions, see the AWS SMS documentation. For information about configuring vCenter, see the blog post [AWS Server Migration Service – Server Migration to the Cloud Made Easy](https://aws.amazon.com/blogs/migration/aws-server-migration-service-server-migration-to-the-cloud-made-easy/).

• **Validate after installation.** After deploying the connector, verify the registration by signing in to the AWS Management console and opening the AWS SMS console. You should see the connector registered on the Connectors page.

## Epics

### Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check the current state footprint and performance baseline (application discovery).</td>
<td>Assess the Linux server before rehosting.</td>
<td>BA, Migration Lead</td>
</tr>
<tr>
<td>Validate the source and target OS instance.</td>
<td>Confirm that the Linux server version is compatible with the target EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance.</td>
<td>Confirm that Linux server hardware is compatible with the targeted EC2 bare metal instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td>Validate the Linux server storage capacity and compatibility with the targeted EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td>Make sure that the targeted EC2 instance is compatible with the existing Linux server profile.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td>Confirm that the security configuration for the target EC2 instance is set up properly.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create an outbound security group to the source and target databases.</td>
<td>Make sure that the new security group maps correctly from the Linux server to the target EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Complete the migration design and migration guide for the application.</td>
<td>Confirm that the migration design and migration guide map correctly to the Linux server and target EC2 instance.</td>
<td>Build Lead, Migration Lead</td>
</tr>
</tbody>
</table>
Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete the application migration runbook.</td>
<td>Assess the migration strategy that has been mapped.</td>
<td>Build Lead, Cutover Lead, Testing Lead, Migration Lead</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>Set up a VPC for the Linux server that is being migrated to the targeted EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td>Establish security groups that map to the Linux server that is being migrated to the targeted EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>File a change request for firewall rule changes.</td>
<td>Make sure that the firewall for the targeted EC2 instance maps to the Linux server that is being migrated.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>File a change request for DNS changes.</td>
<td>Make sure that the DNS for the targeted EC2 instance maps to the Linux server that is being migrated.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Download the AWS Service Migration Connector.</td>
<td>Confirm that the AWS Service Migration Connector has been downloaded and prepared for Linux server migration to the targeted EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Install the AWS Server Migration Connector.</td>
<td>Set up the connector to migrate the Linux server to the targeted EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Verify that the connector is displayed in the AWS SMS console.</td>
<td>If the connector is properly configured, it will show up in the AWS SMS console when you choose Connectors from the navigation pane.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

Migrate your application using AWS SMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a replication job.</td>
<td>Establish the replication environment.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure server-specific settings (select the license type).</td>
<td>Confirm that the server configuration maps to the targeted instance.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>
## Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure replication job settings with the appropriate IAM role (the output should be an AMI).</td>
<td>Manage access to the replication environment that you have set up.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Enable subsequent replication runs to occur every 12 to 24 hours (12 hours preferred).</td>
<td>Establish a continuous schedule for replication runs.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Complete the setup.</td>
<td>Follow the procedure described at <a href="https://aws.amazon.com/blogs/apn/aws-server-migration-service-server-migration-to-the-cloud-made-easy/">https://aws.amazon.com/blogs/apn/aws-server-migration-service-server-migration-to-the-cloud-made-easy/</a> to complete the setup.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

## Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicate any changes or outages to business and application users and stakeholders.</td>
<td>Make sure that everyone is up to date on the status of migration.</td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Update production DNS entries for the servers being replicated via AWS SMS to point to the sitedown/maintenance URL, and set the DNS TTL to 1 minute.</td>
<td>This is the final step before shutting down and stopping source servers.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Shut down and stop source servers.</td>
<td>Commit to cloud operations.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Rename source servers to append &quot;DO NOT TURN ON&quot;.</td>
<td>Do not re-engage source servers.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Run the AWS SMS VM migration task to copy final data sync of source servers to the production AWS account.</td>
<td>Complete the final data sync of source servers.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Run AWS CloudFormation scripts to build and apply any infrastructure configuration (for example, security group, firewall ports allowed, IAM roles, permissions).</td>
<td>Use AWS CloudFormation templates to ensure that correct procedures are followed and everything is in order for cloud operations.</td>
<td>SysAdmin, Migration Engineer</td>
</tr>
<tr>
<td>Make application-specific changes to map to new IP addresses, and make any other required configuration changes, as specified in the application migration runbook.</td>
<td>Make sure that necessary changes are made, mapping to what is detailed in the application migration runbook.</td>
<td>SysAdmin, Migration Engineer, Application Owner</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Update production DNS entries for source servers to point to the EC2 application instance.</td>
<td>Map DNS entries to targeted EC2 instance.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Start the application and its associated databases. Validate the application in accordance with product verification tests (PVTs), build verification tests (BVTs), and other test regimens, as specified in the application migration runbook.</td>
<td>Make sure that the applications that were migrated to AWS are validated according to the test regimen described in the application migration runbook.</td>
<td>App Owner, QA Team</td>
</tr>
<tr>
<td>Obtain sign-off from the TSO or business owner that application testing is complete.</td>
<td>Get final approval and validation from the TSO or business owner that the application works as expected.</td>
<td>Cutover Lead, App Owner</td>
</tr>
<tr>
<td>Update the configuration management database (CMDB) with details on the migrated VMs/hosts and new EC2 instances.</td>
<td>Make sure that the CMDB is updated to reflect the latest information.</td>
<td>Cutover Lead, App Owner</td>
</tr>
<tr>
<td>Communicate project completion and new environment availability to business and application users and stakeholders.</td>
<td>Make sure that all concerned parties are aware that changes have been completed.</td>
<td>Cutover Lead</td>
</tr>
<tr>
<td><strong>Close the project</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shut down temporary AWS resources</td>
<td>Stop using the temporary AWS resources that were set up for migration.</td>
<td>SysAdmin, Migration Engineer</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td>Check project results against plans, post-migration.</td>
<td>Migration Lead, SysAdmin, Application Owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td>Report on how the migration process progressed and was completed.</td>
<td>Migration Lead, DBA, SysAdmin, Application Owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td>Include details on anything that wasn’t previously mentioned.</td>
<td>Migration Lead, DBA, SysAdmin, Application Owner</td>
</tr>
</tbody>
</table>

**Related resources**

- AWS SMS documentation
- AWS Server Migration Service – Server Migration to the Cloud Made Easy (blog post)
Migrate a Microsoft Azure VM to Amazon EC2 using CloudEndure

Created by Pavel Rubin (AWS)

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Source: Compute</th>
<th>Target: Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R Type:</strong> Rehost</td>
<td><strong>Workload:</strong> Microsoft</td>
<td><strong>Technologies:</strong> Migration</td>
</tr>
</tbody>
</table>

**Summary**

This pattern provides guidance for migrating a Microsoft Azure virtual machine (VM) to an Amazon Elastic Compute Cloud (Amazon EC2) instance using CloudEndure Migration.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the Amazon Web Services (AWS) Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

**Prerequisites and limitations**

**Prerequisites**

- An active CloudEndure account
- An active AWS account

**Limitations**

- For a list of supported operating systems, see the CloudEndure website

**Architecture**

**Source technology stack**

- Azure VM

**Target technology stack**

- Amazon EC2

**Target architecture**

*Network architecture:*
Network Architecture

Migration architecture:
**Tools**

- **CloudEndure Migration** - Helps you simplify, expedite, and automate large-scale migrations to AWS. Continuous data replication takes place in the background, without application disruption or performance impact, which ensures that data is synced in real time and minimizes cutover windows.

**Epics**

**Subscribe to CloudEndure Migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register your CloudEndure account and confirm your email address.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Set your CloudEndure account password, and accept terms and conditions.</td>
<td>The password must be at least 8 characters long and must contain one uppercase letter, one lowercase letter, one digit, and one special character.</td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
### Sign in to the CloudEndure user console and create a project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure user console at <a href="https://console.cloudendure.com">https://console.cloudendure.com</a>.</td>
<td>Sign in with the credentials you created in the previous step.</td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Choose the plus button in the upper-left corner of the console to create a project.</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Enter a project name, choose &quot;Live Migration&quot; as the project type, and choose a license package to associate with the project.</td>
<td>You can acquire a license through AWS Marketplace.</td>
<td>CloudEndure Admin</td>
</tr>
</tbody>
</table>

### Generate and use AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the AWS Identity and Access Management (IAM) console, generate an access key ID and secret access key.</td>
<td>The IAM console is at <a href="https://console.aws.amazon.com/iam/">https://console.aws.amazon.com/iam/</a>.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>In the IAM console, choose &quot;Policies,&quot; and then choose &quot;Create policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the JSON tab, copy the CloudEndure IAM policy document into the field, and then choose &quot;Review policy.&quot;</td>
<td>The policy can be found at <a href="https://console.cloudendure.com/IAMPolicy.json">https://console.cloudendure.com/IAMPolicy.json</a>.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Enter a name and description for the policy, and then choose &quot;Create policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create a new user by navigating to &quot;Users&quot; in the IAM console.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose &quot;Add user,&quot; give the user a user name, and set the access type to &quot;Programmatic access.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose &quot;Next: Permissions,&quot; and on the Set permissions page, choose &quot;Attach existing policies.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Locate the policy you created, check the box to the left of the policy name, and choose &quot;Next: Review.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
### Migrate an Azure VM to Amazon EC2 using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify that the fields are correct, and then choose “Create user.”</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy the access key ID and secret access key. Store them in a safe location.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Sign in to the CloudEndure user console, and select your migration project. The CloudEndure console is at <a href="https://console.cloudendure.com">https://console.cloudendure.com</a>.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the “Setup &amp; Info” tab from the left menu, and navigate to &quot;AWS Credentials.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy and paste your AWS access key ID and secret access key into the corresponding fields, and then choose &quot;Save.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>

#### Configure replication settings

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure user console, navigate to “Setup &amp; Info,” and then choose &quot;Replication settings.&quot;</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>For &quot;Live Migration Source,&quot; choose &quot;Other Infrastructure.&quot; &quot;Other Infrastructure&quot; includes all infrastructures supported by CloudEndure.</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Define your replication servers. See the CloudEndure documentation links in the &quot;References and Help&quot; section for detailed instructions.</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
</tbody>
</table>

#### Install CloudEndure agents on source machines

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure user console, navigate to &quot;Machines,&quot; &quot;Machine Actions,&quot; &quot;Add Machines.&quot;</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Locate your agent installation token. When you run the installer file on a source machine, you are first asked to enter your installation token. The token is a unique string of characters that is automatically generated</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
</tbody>
</table>
### Migrate an Azure VM to Amazon EC2 using CloudEndure

#### Task Description

**Skills required**

**Task**

1. **For Linux machines**, copy the installer command, log in to your source machines, and run the installer.  
   **Description**: For detailed instructions, see the CloudEndure documentation in the "References and Help" section.  
   **Skills required**: CloudEndure Admin, OS Admin

2. **For Windows machines**, download the installer file to each machine, and then run the installer command.  
   **Description**: For detailed instructions, see the CloudEndure documentation in the "References and Help" section.  
   **Skills required**: CloudEndure Admin, OS Admin

3. **When the agent has been installed**, the source machine will appear in the CloudEndure console, in the Machines tab. Wait as the machine undergoes initial data replication.  
   **Skills required**: CloudEndure Admin, OS Admin

#### Configure the target machine's blueprint

**Task**

1. **In the CloudEndure console**, choose a machine name from the Machines tab to access the Machine Details pane.  
   **Skills required**: CloudEndure Admin, OS Admin

2. **Choose the Blueprint tab and configure settings.**  
   **Description**: You can configure a variety of Blueprint settings. For detailed instructions, see the CloudEndure documentation in the "References and Help" section.  
   **Skills required**: CloudEndure Admin, OS Admin

#### Perform a test

**Task**

1. **In the CloudEndure console**, in the Machines tab, select the box to the left of each source machine.  
   **Skills required**: AWS SysAdmin, CloudEndure Admin

2. **In the "Launch x Target Machines" menu**, choose "Test Mode."  
   **"x" represents the number of target machines.**  
   **Skills required**: AWS SysAdmin, CloudEndure Admin
### Migrate an Azure VM to Amazon EC2 using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose &quot;Continue&quot; when prompted. The machines will be launched.</strong></td>
<td>You can review the launch process on the Job Progress tab.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td><strong>Verify that the test completed successfully.</strong></td>
<td>The Machine tab will indicate that testing is complete: The edge of each tested source machine will be marked in green. The orange flag icon under the Status column, which previously indicated that the machine had not been tested, will disappear (if no other issues are found). The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the Live Migration Lifecycle column, a message will appear stating that the machine has been tested recently.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td><strong>Test the target machines by choosing each machine's name, opening the Target tab, copying the public IP address, and navigating to that IP address.</strong></td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
</tbody>
</table>

### Schedule and perform a cutover

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schedule a cutover window.</strong></td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td><strong>In the Machines tab, select the box to the left of each source machine.</strong></td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td><strong>In the &quot;Launch x Target Machines&quot; menu, choose &quot;Cutover Mode.&quot;</strong></td>
<td>&quot;x&quot; represents the number of target machines.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td><strong>Choose &quot;Continue&quot; when prompted. The machines will be launched.</strong></td>
<td>You can review the launch process on the Job Progress tab.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td><strong>Verify that the cutover completed successfully.</strong></td>
<td>The Machines tab will indicate that the cutover is complete: The edge of each tested source machine will be marked in green. The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>appear. In the Live Migration Lifecycle column, a message will appear stating that the machine has cut over.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

**Product page on AWS**
- CloudEndure Migration overview

**AWS Marketplace**
- CloudEndure Migration

**CloudEndure documentation**
- Signing in to the console
- Creating a project
- Generating and using credentials
- Configuring replication settings
- Installing agents

**Tutorials and videos**
- CloudEndure Troubleshooting Playbook
- CloudEndure Video Guide

**Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud**

*Created by Will Bauer (AWS)*

<table>
<thead>
<tr>
<th>Source: F5 BIG-IP TMOS 13.1 and later</th>
<th>Target: F5 BIG-IP VE on AWS</th>
<th>R Type: Rehost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment: Production</td>
<td>Technologies: Migration; Security, identity, compliance; Networking</td>
<td>Workload: All other workloads</td>
</tr>
<tr>
<td>AWS services: Amazon EC2;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon VPC; AWS Transit Gateway;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon CloudFront; Amazon CloudWatch; AWS Global Accelerator; AWS CloudFormation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

Organizations are looking to migrate to the Amazon Web Services (AWS) Cloud to increase their agility and resilience. After you migrate your F5 BIG-IP security and traffic management solutions to the AWS Cloud, you can focus on agility and adoption of high-value operational models across your enterprise architecture.

This pattern describes how to migrate an F5 BIG-IP workload to an F5 BIG-IP Virtual Edition (VE) workload on the AWS Cloud. The workload will be migrated by rehosting the existing environment and deploying aspects of replatforming, such as service discovery and API integrations. AWS CloudFormation templates accelerate your workload's migration to the AWS Cloud.

This pattern is intended for technical engineering and architectural teams that are migrating F5 security and traffic management solutions, and accompanies the guide Migrating from F5 BIG-IP to F5 BIG-IP VE on the AWS Cloud on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- An existing on-premises F5 BIG-IP workload.
- Existing F5 licenses for BIG-IP VE versions.
- An active AWS account.
- An existing virtual private cloud (VPC) configured with an egress through a NAT gateway or Elastic IP address, and configured with access to the following endpoints: Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), AWS Security Token Service (AWS STS), and Amazon CloudWatch. You can also modify the Modular and scalable VPC architecture Quick Start as a building block for your deployments.
- One or two existing Availability Zones, depending on your requirements.
- Three existing private subnets in each Availability Zone.
- AWS CloudFormation templates, available in the F5 GitHub repository.

During the migration, you might also use the following, depending on your requirements:

- An F5 Cloud Failover Extension to manage Elastic IP address mapping, secondary IP mapping, and route table changes.
- If you use multiple Availability Zones, you will need to use the F5 Cloud Failover Extensions to handle the Elastic IP mapping to virtual servers.
- You should consider using F5 Application Services 3 (AS3), F5 Application Services Templates (FAST), or another infrastructure as code (IaC) model to manage the configurations. Preparing the configurations in an IaC model and using code repositories will help with the migration and your ongoing management efforts.

Expertise

- This pattern requires familiarity with how one or more VPCs can be connected to existing data centers. For more information about this, see Network-to-Amazon VPC connectivity options in the Amazon VPC documentation.
- Familiarity is also required with F5 products and modules, including Traffic Management Operating System (TMOS), Local Traffic Manager (LTM), Global Traffic Manager (GTM), Access Policy Manager (APM), Application Security Manager (ASM), Advanced Firewall Manager (AFM), and BIG-IQ.
Product versions

- We recommend that you use F5 BIG-IP version 13.1 or later, although the pattern supports F5 BIG-IP version 12.1 or later.

Architecture

Source technology stack

- F5 BIG-IP workload

Target technology stack

- Amazon CloudFront
- Amazon CloudWatch
- Amazon EC2
- Amazon S3
- Amazon VPC
- AWS Global Accelerator
- AWS STS
- AWS Transit Gateway
- F5 BIG-IP VE

Target architecture

Tools

- **AWS CloudFormation** – AWS CloudFormation helps you to create and provision AWS infrastructure deployments predictably and repeatedly.
- **Amazon CloudFront** - CloudFront speeds up distribution of your static and dynamic web content, such as .html, .css, .php, image, and media files.
- **Amazon CloudWatch** – CloudWatch provides a reliable, scalable, and flexible monitoring solution that you can start using within minutes.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity for you to build and host your software systems.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you’ve defined.
- **AWS Identity and Access Management (IAM)** – IAM is a web service for securely controlling access to AWS services.
- **AWS Landing Zone** – AWS Landing Zone is a solution that helps customers quickly set up a secure, multi-account AWS environment based on AWS best practices.
- **AWS STS** – AWS Security Token Service (AWS STS) is a web service that helps you request temporary, limited-privilege credentials for IAM users or for users that you authenticate (federated users).
- **AWS Transit Gateway** – AWS Transit Gateway is a highly available and scalable service to consolidate the Amazon VPC routing configuration for a Region with a hub-and-spoke architecture.

**Epics**

**Discovery and assessment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assess the performance of F5 BIG-IP.</td>
<td>Collect and record the performance metrics of the applications on the virtual server, and metrics of systems that will be migrated. This will help to correctly size the target AWS infrastructure for better cost optimization.</td>
<td>F5 Architect, Engineer and Network Architect, Engineer</td>
</tr>
<tr>
<td>Evaluate the F5 BIG-IP operating system and configuration.</td>
<td>Evaluate which objects will be migrated and if a network structure needs to be maintained, such as VLANs.</td>
<td>F5 Architect, Engineer</td>
</tr>
<tr>
<td>Evaluate F5 license options.</td>
<td>Evaluate which license and consumption model you will require. This assessment should be based on your evaluation of the F5 BIG-IP operating system and configuration.</td>
<td>F5 Architect, Engineer</td>
</tr>
<tr>
<td>Evaluate the public applications.</td>
<td>Determine which applications will require public IP addresses. Align those applications to the required instances and clusters to meet performance and service-level agreement (SLA) requirements.</td>
<td>F5 Architect, Cloud Architect, Network Architect, Engineer, App Teams</td>
</tr>
<tr>
<td>Evaluate internal applications.</td>
<td>Evaluate which applications will be used by internal users. Make sure you know where those internal users sit in the organization and how those</td>
<td>F5 Architect, Cloud Architect, Network Architect, Engineer, App Teams</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>environments connect to the AWS Cloud. You should also make sure those applications can use domain name system (DNS) as part of the default domain.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finalize the AMI.</td>
<td>Not all F5 BIG-IP versions are created as Amazon Machine Images (AMIs). You can use the F5 BIG-IP Image Generator Tool if you have specific required quick-fix engineering (QFE) versions. For more information about this tool, see the &quot;Related resources&quot; section.</td>
</tr>
<tr>
<td></td>
<td>Finalize the instance types and architecture.</td>
<td>Decide on the instance types, VPC architecture, and interconnected architecture.</td>
</tr>
</tbody>
</table>

**Complete security and compliance-related activities**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Document the existing F5 security policies.</td>
<td>Collect and document existing F5 security policies. Make sure you create a copy of them in a secure code repository.</td>
</tr>
<tr>
<td></td>
<td>Encrypt the AMI.</td>
<td>(Optional) Your organization might require encryption of data at rest. For more information about creating a custom Bring Your Own License (BYOL) image, see the &quot;Related resources&quot; section.</td>
</tr>
<tr>
<td></td>
<td>Harden the devices.</td>
<td>This will help protect against potential vulnerabilities.</td>
</tr>
</tbody>
</table>

**Configure your new AWS environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create edge and security accounts.</td>
<td>Sign in to the AWS Management Console and create the AWS accounts that will provide and operate the edge and security services. These accounts might be different from the accounts that operate VPCs for shared services and applications.</td>
</tr>
</tbody>
</table>
## Deploy edge and security VPCs.

**Description:** Set up and configure the VPCs required to deliver edge and security services.

**Skills required:** Cloud Architect, Engineer

## Connect to the source data center.

**Description:** Connect to the source data center that hosts your F5 BIG-IP workload.

**Skills required:** Cloud Architect, Network Architect, Engineer

## Deploy the VPC connections.

**Description:** Connect the edge and security service VPCs to the application VPCs.

**Skills required:** Network Architect, Engineer

## Deploy the instances.

**Description:** Deploy the instances by using the AWS CloudFormation templates from the "Related resources" section.

**Skills required:** F5 Architect, Engineer

## Test and configure instance failover.

**Description:** Make sure that the AWS Advanced HA IAPP template or F5 Cloud Failover Extension is configured and operating correctly.

**Skills required:** F5 Architect, Engineer

### Configure networking

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the VPC topology.</td>
<td>Open the Amazon VPC console and make sure that your VPC has all the required subnets and protections for the F5 BIG-IP VE deployment.</td>
<td>Network Architect, F5 Architect, Cloud Architect, Engineer</td>
</tr>
<tr>
<td>Prepare your VPC endpoints.</td>
<td>Prepare the VPC endpoints for Amazon EC2, Amazon S3, and AWS STS if an F5 BIG-IP workload does not have access to a NAT Gateway or Elastic IP address on a TMM interface.</td>
<td>Cloud Architect, Engineer</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the configuration.</td>
<td>Migrate the F5 BIG-IP configuration to F5 BIG-IP VE on the AWS Cloud.</td>
<td>F5 Architect, Engineer</td>
</tr>
<tr>
<td>Associate the secondary IPs.</td>
<td>Virtual server IP addresses have a relationship with</td>
<td>F5 Architect, Engineer</td>
</tr>
</tbody>
</table>
Task | Description | Skills required
--- | --- | ---
Assign secondary IP addresses and make sure “Allow remap/reassignment” is selected. | | 

Test configurations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the virtual server configurations.</td>
<td>Test the virtual servers.</td>
<td>F5 Architect, App Teams</td>
</tr>
</tbody>
</table>

Finalize operations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the backup strategy.</td>
<td>Systems must be shut down to create a full snapshot. For more information, see “Updating an F5 BIG-IP virtual machine” in the Related resources section.</td>
<td>F5 Architect, Cloud Architect, Engineer</td>
</tr>
<tr>
<td>Create the cluster failover runbook.</td>
<td>Make sure that the failover runbook process is complete.</td>
<td>F5 Architect, Engineer</td>
</tr>
<tr>
<td>Set up and validate logging.</td>
<td>Configure F5 Telemetry Streaming to send logs to the required destinations.</td>
<td>F5 Architect, Engineer</td>
</tr>
</tbody>
</table>

Complete the cutover

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut over to the new deployment.</td>
<td></td>
<td>F5 Architect, Cloud Architect, Network Architect, Engineer, AppTeams</td>
</tr>
</tbody>
</table>

Related resources

**Migration guide**
- Migrating from F5 BIG-IP to F5 BIG-IP VE on the AWS Cloud

**F5 resources**
- AWS CloudFormation templates in the F5 GitHub repository
- F5 in AWS Marketplace
Migrate an on-premises Go web application to AWS Elastic Beanstalk by using the binary method

*Created by Suhas Basavaraj (AWS)*

<table>
<thead>
<tr>
<th>R Type: Rehost</th>
<th>Source: Applications</th>
<th>Target: Elastic Beanstalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Websites &amp; web apps; Migration</td>
</tr>
</tbody>
</table>

**AWS services:** AWS Elastic Beanstalk

**Summary**

This pattern describes how to migrate an on-premises Go web application to AWS Elastic Beanstalk. After the application is migrated, Elastic Beanstalk builds the binary for the source bundle and deploys it to an Amazon Elastic Compute Cloud (Amazon EC2) instance.

As a rehost migration strategy, this pattern's approach is fast and requires no code changes, which means less testing and migration time.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- An on-premises Go web application.
- A GitHub repository that contains your Go application's source code. If you do not use GitHub, there are other ways to [create an application source bundle for Elastic Beanstalk](#).

**Product versions**
• The most recent Go 1.14.x version supported by Elastic Beanstalk. For more information, see the "Related resources" section.

Architecture

Source technology stack
• An on-premises Go web application

Target technology stack
• AWS Elastic Beanstalk
• Amazon CloudWatch

Target architecture

Tools
• AWS Elastic Beanstalk – AWS Elastic Beanstalk quickly deploys and manages applications in the AWS Cloud without users having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control.
• GitHub – An open-source distributed version control system.
## Epics

### Create the Go web application source bundle .zip file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the source bundle for the Go application.</td>
<td>Open the GitHub repository that contains your Go application's source code and prepare the source bundle. The source bundle contains an &quot;application.go&quot; source file in the root directory, which hosts the main package for your Go application. If you do not use GitHub, see the &quot;Prerequisites&quot; section for other ways to create your application source bundle.</td>
<td>System Admin, Application Developer</td>
</tr>
<tr>
<td>Create a configuration file.</td>
<td>Create an &quot;.ebextensions&quot; folder in your source bundle, and then create an &quot;options.config&quot; file inside this folder. For more information, see the &quot;Related resources&quot; section.</td>
<td>System Admin, Application Developer</td>
</tr>
<tr>
<td>Create the source bundle .zip file.</td>
<td>Run the &quot;git archive -o ../godemoapp.zip HEAD&quot; command. This creates the source bundle .zip file. Download and save the .zip file as a local file. Important: The .zip file cannot exceed 512 MB and cannot include a parent folder or top-level directory.</td>
<td>System Admin, Application Developer</td>
</tr>
</tbody>
</table>

### Migrate the Go web application to Elastic Beanstalk

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the Elastic Beanstalk application.</td>
<td>Sign in to the AWS Management Console and open the Elastic Beanstalk console. Choose your AWS Region from the &quot;Regions&quot; list. In the navigation pane, choose &quot;Applications,&quot; and then choose an existing Elastic Beanstalk application or create one. For instructions on how to create an Elastic Beanstalk application, see the &quot;Related resources&quot; section.</td>
<td>System Admin, Application Developer</td>
</tr>
<tr>
<td>Initiate the Elastic Beanstalk web server environment.</td>
<td>On the application overview page, choose &quot;Create a new&quot;</td>
<td>System Admin, Application Developer</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th><strong>Task</strong></th>
<th><strong>Description</strong></th>
<th><strong>Skills required</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>environment,” and then choose “Web server environment.” Fill out “Environment name” and “Domain name.” Choose “Platform version,” and select Go as your platform.</td>
<td></td>
<td>System Admin, Application Developer</td>
</tr>
<tr>
<td>Test the deployed Go web application.</td>
<td>You will be redirected to the Elastic Beanstalk application’s overview page. On the left side, next to “Environment ID,” choose the URL ending in “elasticbeanstalk.com” to navigate to your application. Your application must use this name in its configuration file as an environment variable and display it on the web page.</td>
<td>System Admin, Application Developer</td>
</tr>
</tbody>
</table>

### Related resources

- Go platform versions supported by Elastic Beanstalk
- Using configuration files with Elastic Beanstalk
- Creating an example application in Elastic Beanstalk

### Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP

*Created by Akash Kumar (AWS)*

<table>
<thead>
<tr>
<th><strong>Environment:</strong> Production</th>
<th><strong>Source:</strong> Storage</th>
<th><strong>Target:</strong> Amazon S3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R Type:</strong> Rehost</td>
<td><strong>Technologies:</strong> Migration; Storage &amp; backup; Websites &amp; web apps</td>
<td><strong>AWS services:</strong> Amazon S3; AWS Transfer Family; Amazon CloudWatch Logs</td>
</tr>
</tbody>
</table>

### Summary

This pattern describes how to migrate an on-premises file transfer solution that uses the Secure Shell (SSH) File Transfer Protocol (SFTP) to the Amazon Web Services (AWS) Cloud by using the AWS Transfer Family.
for SFTP service. Users generally connect to an SFTP server either through its domain name or by fixed IP. This pattern covers both cases.

AWS Transfer for SFTP is a member of the AWS Transfer Family. It is a secure transfer service that you can use to transfer files into and out of AWS storage services over SFTP. You can use AWS Transfer for SFTP with Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS). This pattern uses Amazon S3 for storage.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing SFTP domain name or fixed SFTP IP.

Limitations

- The data files to be transferred must be smaller than 5 TB in size. For files over 5 TB, you can perform a multipart upload to Amazon S3 or choose another data transfer method.

Architecture

Source technology stack

- On-premises flat files or database dump files

Target technology stack

- AWS Transfer for SFTP
- Amazon S3
- Amazon Virtual Private Cloud (Amazon VPC)
- AWS Identity and Access Management (IAM) roles and policies
- Elastic IP addresses
- Security groups
- Amazon CloudWatch Logs (optional)

Target architecture
Automation and scale

To automate the target architecture for this pattern, use the attached AWS CloudFormation templates:

- amazon-vpc-subnets.yml provisions a virtual private cloud (VPC) with two public and two private subnets.
- amazon-sftp-server.yml provisions the SFTP server.
- amazon-sftp-customer.yml adds users.

Tools

AWS services

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses Amazon S3 as the storage system for file transfers.
• **AWS Transfer for SFTP** – AWS Transfer for SFTP is a service in the AWS Transfer Family. It enables you to transfer files into and out of AWS storage services over SFTP.

• **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

• **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. This virtual network closely resembles a traditional network that you’d operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

• **Amazon CloudWatch Logs** – Amazon CloudWatch Logs centralizes the logs from all your systems, applications, and AWS services that you use. You can view and monitor the logs, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis.

### Epics

**Create a VPC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC with subnets.</td>
<td>Open the Amazon VPC console at <a href="https://console.aws.amazon.com/vpc/">https://console.aws.amazon.com/vpc/</a>. Create a virtual private cloud (VPC) with two public subnets. (The second subnet provides high availability.) —or— You can deploy the attached CloudFormation template, <code>amazon-vpc-subnets.yml</code>, in the CloudFormation console to automate the tasks in this epic.</td>
<td>Developer, Systems administrator</td>
</tr>
<tr>
<td>Add an internet gateway.</td>
<td>Provision an internet gateway and attach it to the VPC.</td>
<td>Developer, Systems administrator</td>
</tr>
<tr>
<td>Migrate an existing IP.</td>
<td>Attach an existing IP to the Elastic IP address. You can create an Elastic IP address from your address pool and use it.</td>
<td>Developer, Systems administrator</td>
</tr>
</tbody>
</table>

**Provision an SFTP server**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SFTP server.</td>
<td>Open the AWS Transfer Family console at <a href="https://console.aws.amazon.com/transfer/">https://console.aws.amazon.com/transfer/</a>. Follow the instructions in Create an internet-facing</td>
<td>Developer, Systems administrator</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Task Description: Migrate the domain name.
Attach the existing domain name to the custom hostname. If you're using a new domain name, use the Amazon Route 53 DNS alias. For an existing domain name, choose Other DNS. For more information, see Working with custom hostnames in the AWS Transfer Family documentation.

#### Skills required: Developer, Systems administrator

#### Task Description: Add a CloudWatch logging role.
(Optional) if you want to enable CloudWatch logging, create a Transfer role with the CloudWatch Logs API operations logs:CreateLogGroup, logs:CreateLogStream, logs:DescribeLogStreams, and logs:PutLogEvents. For more information, see Log activity with CloudWatch in the AWS Transfer Family documentation.

#### Skills required: Developer, system admin

#### Task Description: Save and submit.
Choose Save. For Actions, choose Start and wait for the SFTP server to be created with the status Online.

#### Skills required: Developer, Systems administrator

---

### Map Elastic IP addresses to the SFTP server

#### Task Description: Stop the server so you can modify settings.
On the AWS Transfer Family console, choose Servers, and then select the SFTP server you created. For Actions, choose Stop. When the server is offline,

#### Skills required: Developer, system admin
### Choose Availability Zones and subnets.

In the **Availability Zones** section, choose the Availability Zones and subnets for your VPC. 

**Skills required:** Developer, Systems administrator

### Add Elastic IP addresses.

For **IPv4 Addresses**, choose an Elastic IP address for each subnet, and then choose **Save**.

**Skills required:** Developer, Systems administrator

## Add users

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an IAM role for users to access the S3 bucket. | Create a IAM role for Transfer and add `s3:ListBucket`, `s3:GetBucketLocation`, `s3:GetObject` with the S3 bucket name as a resource. For more information, see Create an IAM role and policy in the AWS Transfer Family documentation.  
---or---  
You can deploy the attached CloudFormation template, `amazon-sftp-customer.yml`, in the CloudFormation console to automate the tasks in this epic. | Developer, Systems administrator     |
| Create an S3 bucket.                      | Create a S3 bucket for the application.                                                                                                                                                                    | Developer, Systems administrator          |
| Create optional folders.                 | (Optional) If you want to store files for users separately, in specific Amazon S3 folders, add folders as appropriate.                                                                                      | Developer, Systems administrator          |
| Create an SSH public key.                | To create an SSH key pair, see Generate SSH keys in the AWS Transfer Family documentation.                                                                                                                | Developer, Systems administrator          |
| Add users.                               | On the AWS Transfer Family console, choose **Servers**, select the SFTP server you created, and then choose **Add user**. For **Home directory**, choose the S3 bucket you created. For **SSH public key**, specify the public key portion of the SSH key pair. Add users for the SFTP server, and then choose **Add**. | Developer, Systems administrator          |
Test the SFTP server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the security group.</td>
<td>In the Security Groups section of your SFTP server, add your test machine's IP to gain SFTP access.</td>
<td>Developer</td>
</tr>
<tr>
<td>Use an SFTP client utility to test the server.</td>
<td>Test file transfers by using any SFTP client utility. For a list of clients and instructions, see Transferring files using a client in the AWS Transfer Family documentation.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

Related resources

- AWS Transfer Family User Guide
- Amazon S3 User Guide
- Elastic IP addresses in the Amazon EC2 documentation

Attachments

attachment.zip

Migrate small sets of data from on premises to Amazon S3 using AWS SFTP

Created by Charles Gibson (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Rehost</th>
<th>Source:</th>
<th>Storage</th>
<th>Target:</th>
<th>Amazon S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>Production</td>
<td>Technologies:</td>
<td>Storage &amp; backup; Migration</td>
</tr>
</tbody>
</table>

AWS services: Amazon S3

Summary

This pattern describes how to migrate small sets of data (5 TB or less) from on-premises data centers to Amazon Simple Storage Service (Amazon S3) by using AWS Transfer for SFTP (AWS SFTP). The data can be either database dumps or flat files.

Prerequisites and limitations

Prerequisites
- An active AWS account
- An AWS Direct Connect link established between your data center and AWS

Limitations
- The data files must be less than 5 TB. For files over 5 TB, you can perform a multipart upload to Amazon S3 or choose another data transfer method.

Architecture

Source technology stack
- On-premises flat files or database dumps

Target technology stack
- Amazon S3

Source and target architecture

Tools
- AWS SFTP – Enables the transfer of files directly into and out of Amazon S3 using Secure File Transfer Protocol (SFTP).
- AWS Direct Connect – Establishes a dedicated network connection from your on-premises data centers to AWS.
- VPC endpoints – Enable you to privately connect a VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without an internet gateway, network address translation (NAT) device, VPN connection, or AWS Direct Connect connection. Instances in a VPC don’t require public IP addresses to communicate with resources in the service.
## Epics

### Prepare for the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document the current SFTP requirements.</td>
<td></td>
<td>Application owner, SA</td>
</tr>
<tr>
<td>Identify the authentication requirements.</td>
<td>Requirements may include key-based authentication, user name or password, or identity provider (IdP).</td>
<td>Application owner, SA</td>
</tr>
<tr>
<td>Identify the application integration requirements.</td>
<td></td>
<td>Application owner</td>
</tr>
<tr>
<td>Identify the users who require the service.</td>
<td></td>
<td>Application owner</td>
</tr>
<tr>
<td>Determine the DNS name for the SFTP server endpoint.</td>
<td></td>
<td>Networking</td>
</tr>
<tr>
<td>Determine the backup strategy.</td>
<td></td>
<td>SA, DBA (if data is transferred)</td>
</tr>
<tr>
<td>Identify the application migration or cutover strategy.</td>
<td></td>
<td>Application owner, SA, DBA</td>
</tr>
</tbody>
</table>

## Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create one or more virtual private clouds (VPCs) and subnets in your AWS account.</td>
<td></td>
<td>Application owner, AMS</td>
</tr>
<tr>
<td>Create the security groups and network access control list (ACL).</td>
<td></td>
<td>Security, Networking, AMS</td>
</tr>
<tr>
<td>Create the S3 bucket.</td>
<td></td>
<td>Application owner, AMS</td>
</tr>
<tr>
<td>Create the identity and access management (IAM) role.</td>
<td>Create an IAM policy that includes the permissions to enable AWS SFTP to access your S3 bucket. This IAM policy determines what level of access you provide SFTP users. Create another IAM policy to establish a trust relationship with AWS SFTP.</td>
<td>Security, AMS</td>
</tr>
<tr>
<td>Associate a registered domain (optional).</td>
<td>If you have your own registered domain, you can associate it with the SFTP server. You can route SFTP traffic to your SFTP server endpoint from a domain or from a subdomain.</td>
<td>Networking, AMS</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Create an SFTP server.</td>
<td>Specify the identity provider type used by the service to authenticate your users.</td>
<td>Application owner, AMS</td>
</tr>
<tr>
<td>Open an SFTP client.</td>
<td>Open an SFTP client and configure the connection to use the SFTP endpoint host. AWS SFTP supports any standard SFTP client. Commonly used SFTP clients include OpenSSH, WinSCP, Cyberduck, and FileZilla. You can get the SFTP server host name from the AWS SFTP console.</td>
<td>Application owner, AMS</td>
</tr>
</tbody>
</table>

Plan and test

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan the application migration.</td>
<td>Plan for any application configuration changes required, set the migration date, and determine the test schedule.</td>
<td>Application owner, AMS</td>
</tr>
<tr>
<td>Test the infrastructure.</td>
<td>Test in a non-production environment.</td>
<td>Application owner, AMS</td>
</tr>
</tbody>
</table>

Related resources

References

- AWS Transfer for SFTP User Guide
- AWS Direct Connect resources
- VPC Endpoints

Tutorials and videos

- AWS Transfer for SFTP (video)
- AWS Transfer for SFTP user guide
- AWS SA Whiteboarding - Direct Connect (video)

Migrate from Oracle GlassFish to AWS Elastic Beanstalk

Created by Sandeep Bondugula (AWS)
Summary

This pattern describes how to migrate a Java application running on an on-premises Oracle GlassFish server to AWS Elastic Beanstalk in the AWS Cloud.

On AWS, the Java application is deployed on a Docker GlassFish server with AWS Elastic Beanstalk, which runs in an Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling group.

Additional features:

- Amazon Elastic Beanstalk acts as a wrapper for several underlying resources. It sets up Elastic Load Balancing (which handles incoming traffic from Amazon Route 53), disperses the traffic to one or more EC2 instances, and also serves as a deployment tool.
- To migrate an on-premises database to Amazon Relational Database Service (Amazon RDS), update the database connection details. In the backend database, you can configure Amazon RDS Multi-AZ deployments and choose the database engine type.
- You can use Multi-AZ deployment for high availability along with the Auto Scaling group and scaling policy to improve resiliency.
- You can set up a scaling policy based on Amazon CloudWatch metrics.
- In AWS Elastic Beanstalk, you can configure the underlying Elastic Load Balancing settings and Amazon EC2 Auto Scaling.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An on-premises Java application running on GlassFish
- A Java Web Application Resource (WAR) file

Product versions

- Oracle Glassfish 4.1.2 and 5.0
- Java 7 GlassFish 4.0
- Java 8 GlassFish 4.1 or later

Architecture

Source technology stack

- Applications developed in GlassFish
Target technology stack

- Elastic Beanstalk

Target architecture

Deployment workflow
Tools

- **Amazon Elastic Beanstalk** – A service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on servers including Apache, NGINX, Passenger, and IIS.
- **Amazon CloudWatch** – Provides data and actionable insights to monitor applications, responds to systemwide performance changes, optimizes resource utilization, and provides a unified view of operational health.
- **Docker** – A platform that packages software into standardized units to build, test, and deploy applications quickly.
- **Java** – A general-purpose programming language. Java is class-based, object-oriented, and designed to have fewer implementation dependencies.

Epics

Set up a VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create a virtual private cloud (VPC) instance with the required information.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Create at least two subnets within the VPC.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Create a route table per requirements.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

Set up Amazon S3

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create an Amazon Simple Storage Service (Amazon S3) bucket.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Copy the WAR file to the S3 bucket and upload the application code.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

Create an IAM role

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Create an AWS Identity and Access Management (IAM) role.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>You can use the default &quot;aws-elasticbeanstalk-ec2-role&quot; profile, or let Elastic Beanstalk create it automatically.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>
## Set up Elastic Beanstalk

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the Elastic Beanstalk dashboard.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create a new application and choose the</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>web server environment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose GlassFish Docker as the</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>preconfigured platform.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upload the code.</td>
<td>Provide the S3 bucket file URL or ZIP file from local system files.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Choose the environment type.</td>
<td>In Configuration Capacity settings, choose either Single Instance or Load</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Balancer.</td>
<td></td>
</tr>
<tr>
<td>Configure Load Balancer.</td>
<td>If you chose Load Balancer in the previous step, configure the Multi-AZ</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>deployment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Configuration Security settings, choose the previously created IAM role.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>In Configuration Security settings, if you have an existing key pair, use</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>it or create a new Amazon EC2 key pair.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In Configuration Monitoring settings, configure Amazon CloudWatch.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>In Configuration Security settings, choose the previously created VPC.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Choose Create Environment.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

## Test the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the application by using the URL</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>provided in the created environment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply the Domain Name Service (DNS)</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>changes in Amazon Route 53.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Related resources

- Oracle GlassFish documentation
- GlassFish Open Source Java EE Reference Implementation
- AWS Elastic Beanstalk documentation
- Using Elastic Beanstalk with Amazon CloudWatch
- AWS Elastic Beanstalk pricing
- EC2 Auto Scaling Group
- Scaling the Size of Your Auto Scaling Group
- Amazon RDS Multi-AZ Deployments

Migrate an on-premises Oracle database to Oracle on Amazon EC2

Created by Baji Shaik (AWS)

<table>
<thead>
<tr>
<th>R Type</th>
<th>Source: Databases: Relational</th>
<th>Target: Oracle on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
</tr>
<tr>
<td>Workload:</td>
<td>Oracle</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern walks you through the steps for migrating an on-premises Oracle database to Oracle on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It describes two options for migration: using AWS Data Migration Service (AWS DMS) or using native Oracle tools such as RMAN, Data Pump import/export, transportable tablespaces, and Oracle GoldenGate.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database in an on-premises data center

Limitations

- The target operating system (OS) must be supported by Amazon EC2. For a complete list of supported systems, see Amazon EC2 FAQs.

Product versions

- Oracle versions 10.2 and later (for versions 10.x), 11g and up to 12.2, and 18c for the Enterprise, Standard, Standard One, and Standard Two editions. For the latest list of versions supported by AWS DMS, see "On-premises and Amazon EC2 instance databases" in Sources for Data Migration in the AWS DMS documentation.
Architecture

Source technology stack

• An on-premises Oracle database

Target technology stack

• An Oracle database instance on Amazon EC2

Target architecture

Data migration architecture

Using AWS DMS:
Using native Oracle tools:
Tools

- **AWS DMS - AWS Database Migration Services** (AWS DMS) supports several types of source and target databases. For information about the database versions and editions that are supported, see Using an Oracle Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

- **Native Oracle tools** - RMAN, Data Pump import/export, transportable tablespaces, Oracle GoldenGate

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the versions of the source and target databases.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the version of the target OS.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify hardware requirements for the target server instance based on the Oracle</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
### Migrate from Oracle to Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network requirements (latency and bandwidth).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network/host access security requirements for source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify a list of OS users required for Oracle software installation.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Download AWS Schema Conversion Tool (AWS SCT) and drivers.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an AWS SCT project for the workload, and connect to the source database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Generate SQL files for the creation of objects (tables, indexes, sequences, etc.).</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Determine a backup strategy.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Determine availability requirements.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the application migration/switch-over strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

#### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets in your AWS account.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups and network access control lists (ACLs).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the EC2 instance.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

---

1035
Install the Oracle software

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the OS users and groups</td>
<td>required for the Oracle software.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Download the required version</td>
<td>of Oracle software.</td>
<td></td>
</tr>
<tr>
<td>Install the Oracle software on</td>
<td>the EC2 instance.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Create objects like tables,</td>
<td>primary keys, views, and sequences by using the scripts generated by AWS SCT.</td>
<td>DBA</td>
</tr>
<tr>
<td>Migrate data - option 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use native Oracle tools or third-</td>
<td>party tools to migrate database objects and data.</td>
<td>DBA</td>
</tr>
<tr>
<td>party tools to migrate database</td>
<td>Oracle tools include Data Pump import/export, RMAN, transportable</td>
<td></td>
</tr>
<tr>
<td>objects and data.</td>
<td>tablespaces, and GoldenGate.</td>
<td></td>
</tr>
<tr>
<td>Migrate data - option 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine the migration method.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a replication instance in</td>
<td>the AWS DMS console.</td>
<td>DBA</td>
</tr>
<tr>
<td>the AWS DMS console.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create source and target endpoints.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a replication task.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Enable change data capture (CDC)</td>
<td>to capture changes for a continuous replication.</td>
<td>DBA</td>
</tr>
<tr>
<td>Run the replication task and</td>
<td>monitor logs.</td>
<td>DBA</td>
</tr>
<tr>
<td>monitor logs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create secondary objects like</td>
<td>indexes and foreign keys when the full load is done.</td>
<td>DBA</td>
</tr>
<tr>
<td>indexes and foreign keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td>the full load is done.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application cutover/switch-over strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS Secrets Manager resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Related resources

References

- Strategies for Migrating Oracle Databases to AWS
- Amazon EC2 website
- AWS DMS website
- AWS DMS blog posts
- Amazon EC2 Pricing
- Licensing Oracle Software in the Cloud Computing Environment

Tutorials and videos

- Getting Started with Amazon EC2
- Getting Started with AWS DMS
- Introduction to Amazon EC2 - Elastic Cloud Server & Hosting with AWS (video)
Migrate an on-premises Oracle database to Amazon EC2 using CloudEndure

*Created by Pavel Rubin (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type: Rehost</td>
<td>Workload: Oracle</td>
<td>Technologies: Migration; Databases</td>
</tr>
<tr>
<td>AWS services: Amazon EC2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern provides guidance for migrating an Oracle source database from an on-premises data center to an Amazon Elastic Compute Cloud (Amazon EC2) instance using CloudEndure Migration.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the Amazon Web Services (AWS) Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- A source Oracle database in an on-premises data center

**Limitations**

- The target operating system must be supported by Amazon EC2 (for a complete list of supported operating systems, see the General section in Amazon EC2 FAQs)

**Architecture**

**Source technology stack**

- An on-premises Oracle database

**Target technology stack**

- An Oracle database instance on Amazon EC2

**Source and target architecture**

*CloudEndure Migration:*
AWS Prescriptive Guidance Patterns
Migrate from Oracle to Amazon EC2 using CloudEndure

Network architecture:

Tools

- **CloudEndure Migration** – CloudEndure Migration helps you simplify, expedite, and automate large-scale migration to AWS. Continuous Data Replication takes place in the background, without application disruption or performance impact, which ensures that data is synced in real time and minimizes cutover windows. When migration cutover is initiated, CloudEndure runs a highly automated...
machine conversion and orchestration process, allowing even the most complex applications and databases to run natively on AWS without compatibility issues and with minimal IT skills necessary.

Epics

Sign up for CloudEndure through AWS Marketplace

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate to AWS Marketplace, search for CloudEndure Migration, and subscribe.</td>
<td>For a link to AWS Marketplace and for more information about the steps described in subsequent stories, see the &quot;Related resources&quot; section at the end of this pattern.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Register the account and confirm the email.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Set the account password, and accept terms and conditions.</td>
<td>The password must be at least 8 characters long and must contain one uppercase letter, one lowercase letter, one digit, and one special character.</td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>

Sign in to CloudEndure and create a project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure user console.</td>
<td>On the CloudEndure console, sign in with the credentials you created in the previous step.</td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Create a new project.</td>
<td>Choose the plus (+) button in the upper-left corner of the console to create a project.</td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Enter a project name and select a project type and license package.</td>
<td>Select &quot;Live Migration&quot; as the project type. You can acquire a license through AWS Marketplace.</td>
<td>CloudEndure Admin</td>
</tr>
</tbody>
</table>

Generate and use AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the AWS Identity and Access Management (IAM) console, generate an access key ID and secret access key.</td>
<td>To generate the required AWS credentials to use with the CloudEndure user console, you need to create at least one IAM user and assign the proper permission policy to this user. The user console requires an AWS SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>access key ID and a secret access key.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose “Policies,” and then choose &quot;Create policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the JSON tab, copy the CloudEndure IAM policy into the field, and then choose &quot;Review policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Enter a name and description for the policy, and then choose &quot;Create policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create a new IAM user.</td>
<td>In the IAM console, navigate to Users, choose &quot;Add user,&quot; assign the user a name, and then set the &quot;Access type&quot; to Programmatic access.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose &quot;Next: Permissions&quot; and then under &quot;Set permissions for user,&quot; choose &quot;Attach existing policies.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Locate the policy you created, check the box to the left of the policy name, and then choose &quot;Next: Review.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Verify that the fields are correct, and then choose &quot;Create user.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy the access key ID and secret access key and store them in a safe location.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Sign in to the CloudEndure user console, and select your migration project.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the Setup &amp; Info tab from the left menu, and navigate to AWS credentials.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy and paste your AWS access key ID and secret access key into the corresponding fields, and then choose Save.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
### Configure replication settings

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure user console, navigate to Setup &amp; Info, and then choose “Replication settings.”</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>For &quot;Live Migration Source,&quot; choose &quot;Other Infrastructure.&quot;</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Define your replication servers.</td>
<td>See the CloudEndure documentation links in the &quot;Related resources&quot; section for detailed instructions.</td>
<td>CloudEndure Admin</td>
</tr>
</tbody>
</table>

### Install CloudEndure agents on Oracle source machines

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure user console, navigate to &quot;Machines,&quot; &quot;Machine Actions,&quot; &quot;Add Machines.&quot;</td>
<td></td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>Locate your agent installation token.</td>
<td>When you run the installer file on a source machine, you are first asked to enter your installation token. The token is a unique string of characters that is automatically generated for you when your CloudEndure account is activated. One installation token enables you to install the agent on as many source machines as your license package and project allow.</td>
<td>CloudEndure Admin</td>
</tr>
<tr>
<td>For Linux machines, copy the installer command, log in to your source machines, and run the installer.</td>
<td>For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure Admin, OS Admin</td>
</tr>
<tr>
<td>For Windows machines, download the installer file to each machine, and then run the installer command.</td>
<td>For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure Admin, OS Admin</td>
</tr>
<tr>
<td>Wait as the machine undergoes initial data replication.</td>
<td>After the agent is installed, the source machine will appear on the &quot;Machines&quot; tab in the CloudEndure console.</td>
<td>CloudEndure Admin, OS Admin</td>
</tr>
</tbody>
</table>
Configure the target machine's blueprint

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure console, choose a machine name from the Machines tab to access the Machine Details pane.</td>
<td></td>
<td>CloudEndure Admin, OS Admin</td>
</tr>
<tr>
<td>Select the Blueprint tab from the menu.</td>
<td>You can configure a variety of Blueprint settings. For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure Admin, OS Admin</td>
</tr>
</tbody>
</table>

Perform a test

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure console, in the Machines tab, select the box to the left of each source machine.</td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>In the &quot;Launch x Target Machines&quot; menu, choose &quot;Test Mode.&quot;</td>
<td>&quot;x&quot; represents the number of machines to launch.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>Choose &quot;Continue&quot; when prompted. The machines will be launched.</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>Verify that the test completed successfully.</td>
<td>The &quot;Machine&quot; tab will indicate that testing is complete: The edge of each tested &quot;Source&quot; machine will be marked in green. The orange flag icon under the &quot;Status&quot; column, which previously indicated that the machine had not been tested, will disappear (if no other issues are found). The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the &quot;Live Migration Lifecycle&quot; column, a message will appear stating that the machine has been tested recently.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>Test the target machines by choosing each machine's name, opening the Target tab, copying the public IP, and navigating to that IP.</td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
</tbody>
</table>
## Schedule and perform a cutover

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule a cutover window.</td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>In the Machines tab, select the box to the left of each source machine.</td>
<td></td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>In the &quot;Launch x Target Machines&quot; menu, choose &quot;Cutover Mode.&quot;</td>
<td>&quot;x&quot; represents the number of machines to launch.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>Choose Continue when prompted. The machines will be launched.</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
<tr>
<td>Verify that the cutover completed successfully.</td>
<td>The &quot;Machines&quot; tab will indicate that the cutover is complete: The edge of each tested source machine will be marked in green. The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the &quot;Live Migration Lifecycle&quot; column, a message will appear stating that the machine has cut over.</td>
<td>AWS SysAdmin, CloudEndure Admin</td>
</tr>
</tbody>
</table>

## Related resources

### Product page on AWS
- CloudEndure Migration

### AWS Marketplace
- CloudEndure Migration

### CloudEndure
- CloudEndure console
- CloudEndure IAM policy

### CloudEndure documentation
- Signing in to the console
- Creating a project
- Generating and using credentials
- Configuring replication settings
Migrate an on-premises Oracle database to Amazon EC2 using Oracle Data Pump

Created by Navakanth Talluri (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Source: Databases: Relational</th>
<th>Target: Oracle on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Oracle</td>
<td>AWS services: Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an on-premises Oracle database to an Amazon Elastic Compute Cloud (Amazon EC2) instance by using Oracle Data Pump.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An on-premises Oracle database, which must be non-RAC (Real Application Clusters) and non-ASM (Automatic Storage Management). The source database must be in open mode.
- AWS Direct Connect link between the on-premises data center and AWS.

Product versions

- Oracle Database 10g Release 1 (10.1) and later

Architecture

Source technology stack

- Non-RAC and Non-ASM on-premises Oracle database

Target technology stack

- Oracle database on an EC2 instance
Tools

- **Oracle Data Pump** - Oracle Data Pump provides high-speed movement of data and metadata from one database to another. Data Pump is available only on Oracle Database 10g Release 1 (10.1) and later.

- **AWS Direct Connect** - AWS Direct Connect is a cloud service designed to establish a dedicated network connection from an on-premises instance to AWS.
## Epics

**Set up an EC2 instance on AWS**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identify the source hardware configuration for the on-premises host and the kernel parameters.</strong></td>
<td>Validate the existing configuration, including storage size, input/output operations per second (IOPS), CPU, and so on, which is important for Oracle licensing, which is based on CPU cores.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Create the infrastructure on AWS.</strong></td>
<td>Create a virtual private cloud (VPC) and subnets, security groups, network access control lists (ACLs), route tables, and internet gateway. For instructions, see the links in the &quot;Related resources&quot; section.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Set up the host on an EC2 instance, based on the configuration from step 1, and create the new database.</strong></td>
<td>The version of the Oracle database on the EC2 instance can be different from the on-premises version, because we use logical backups. The target database must be in open mode, and the source database Transparent Network Substrate (TNS) detail must exist at the target database.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Create a link to the on-premises database.</strong></td>
<td>Create a link between the Oracle database on the EC2 instance and the on-premises Oracle database.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Verify the connectivity between the EC2 instance and the on-premises host by using a database link (dblink).</strong></td>
<td>For instructions, see the links in the &quot;Related resources&quot; section.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Once the database downtime is approved, stop the applications that connect to the source database.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Schedule the data migration job on the target host to connect to the target database and to extract the data from the on-premises host.</strong></td>
<td>Use the command &quot;impdb&quot; to schedule the Data Pump import. For instructions, see the links in the &quot;Related resources&quot; section.</td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Validate the data.</strong></td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make the source database read-only.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Point the application to connect to the Oracle database on the EC2 instance.</td>
<td></td>
<td>DBA, Developer</td>
</tr>
</tbody>
</table>

Related resources

AWS references
- VPCs and Subnets
- Tutorial: Create an Amazon VPC for Use with a DB Instance

Oracle references
- NETWORK_LINK
- Using Network Link Import to Move Data
- Starting Up a Database
- Create an Instance

Migrate an on-premises PostgreSQL database to Amazon EC2

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type: Rehost</th>
<th>Source: Databases: Relational</th>
<th>Target: PostgreSQL on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
</tbody>
</table>

| Workload: Open-source | AWS services: Amazon EC2 |

Summary

This pattern describes the migration steps from an on-premises PostgreSQL source database to a PostgreSQL database on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It covers two options for migration: using AWS Data Migration Service (AWS DMS) or using native PostgreSQL tools such as `pg_dump`, `pg_restore`, and `psql`.

Prerequisites and limitations

Prerequisites
• An active AWS account
• A hardware platform supported by Amazon EC2
• A compatible operating system (OS), which you can install by using an Amazon Machine Image (AMI) or install yourself
• A PostgreSQL source database in an on-premises data center

Product versions

• PostgreSQL 9.4 and later (for versions 9.x), 10.x, and 11.x, if you're using AWS DMS for migration. For the latest list of supported versions and features, see Using a PostgreSQL Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture

Source technology stack

• On-premises PostgreSQL database

Target technology stack

• PostgreSQL database on an EC2 instance

Target architecture
Data migration architecture

*Using AWS DMS:*

*Using native PostgreSQL tools:*
**Tools**

- **AWS DMS** - AWS Data Migration Service (AWS DMS) supports different sources and target databases. For information about the database versions and editions that are supported, see Using a PostgreSQL Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

- **Native PostgreSQL tools** - These include `pg_dump`, `pg_restore`, and `psql`.

**Epics**

Analyze the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the target OS version.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the hardware requirements for target server instance based on the PostgreSQL compatibility list and capacity requirements.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate from PostgreSQL to Amazon EC2**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identify the storage requirements for the storage type and capacity.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Identify the network requirements, latency, and bandwidth.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Choose the proper instance type, capacity, storage features, and network features.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Identify the network and host access security requirements for the source and target databases.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Identify a list of the OS users required for PostgreSQL software installation.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Determine the backup strategy.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Determine the availability requirements.</strong></td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td><strong>Identify the application migration and switchover strategy.</strong></td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

#### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a virtual private cloud (VPC) and subnets.</strong></td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td><strong>Create security groups and the network access control list (ACL).</strong></td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td><strong>Configure and start an EC2 instance.</strong></td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

#### Install the software

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create OS users and the groups required for PostgreSQL software to work.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td><strong>Download the required version of PostgreSQL software.</strong></td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
### Migrate from PostgreSQL to Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install PostgreSQL software on the EC2 instance and configure the server.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

#### Migrate the data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the database objects and data by using native PostgreSQL or third-party tools.</td>
<td>Native PostgreSQL tools include pg_dump.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Migrate the data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data by using AWS DMS.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

#### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application cutover or switchover strategy.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

#### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Validate and review the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
Migrate an on-premises SAP ASE database to Amazon EC2

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type: Rehost</th>
<th>Source: Databases: Relational</th>
<th>Target: SAP Adaptive Server Enterprise on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: SAP</td>
<td>AWS services: Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an SAP Adaptive Server Enterprise (ASE) database from an on-premises host to an Amazon Elastic Compute Cloud (Amazon EC2) instance. The pattern covers the use of AWS Database Migration Service (AWS DMS) or SAP ASE native tools such as ASE Cockpit, Sybase Central for ASE, and DBA Cockpit for migration.

Prerequisites and limitations

Prerequisites

• An active AWS account
• An SAP ASE source database in an on-premises data center

Limitations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>Migration Lead, App owner</td>
</tr>
</tbody>
</table>
• The source database must be less than 64 TB

Product versions
• SAP ASE version 15.x and 16.x or later

Architecture

Source technology stack
• On-premises SAP ASE database

Target technology stack
• SAP ASE database on an EC2 instance

Database migration architecture

*Using AWS DMS:*

*Using native SAP ASE tools:*
Tools

- **AWS DMS** - AWS Data Migration Service (AWS DMS) supports several different source and target databases. For more information, see Sources for Data Migration and Targets for Data Migration. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

- **SAP ASE** - Native tools include ASE Cockpit, Sybase Central for ASE, and DBA Cockpit.

Epics

Analyze the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the target OS version.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance based on the SAP ASE compatibility list and capacity requirements.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from SAP ASE to Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Identify the requirements for the storage type and capacity.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Identify the network requirements including latency and bandwidth.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Choose the proper instance type, capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Identify the network and host access security requirements for the source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Identify a list of operating system users required for the SAP ASE software installation.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Determine the backup strategy.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Determine the availability requirements.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Identify the application migration and switchover strategy.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Create security groups and the network access control list (ACL).</td>
<td>SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Configure and start the EC2 instance.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Install the software

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Create the OS users and groups required for SAP ASE software to work.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Download the required version of SAP ASE software.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Install the SAP ASE database, backup server software, and</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

1057
## Migrate from SAP ASE to Amazon EC2

### Task | Description | Skills required
--- | --- | ---
replication server software on the EC2 instance and then configure the server. |  |

### Migrate the data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the database objects and data by using native SAP ASE tools or third-party tools.</td>
<td>See the documentation for the SAP ASE or third-party tools. These include ASE Cockpit, Sybase Central for ASE, and DBA Cockpit.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data by using AWS DMS.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application cutover or switchover strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Validate and review the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, percent of manual</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
## Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>versus tool cost savings, and so on.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Close the project and provide any feedback.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

## Related resources

**References**

- Amazon EC2
- AWS DMS
- Amazon EC2 pricing

**Tutorials and videos**

- Getting Started with Amazon EC2
- Getting Started with AWS Database Migration Service
- AWS Data Migration Service (video)
- Introduction to Amazon EC2 - Elastic Cloud Server & Hosting with AWS (video)

## Migrate SAP workloads to an SAP database on Amazon EC2 using CloudEndure

*Created by Pavel Rubin (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>SAP Database on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Rehost</td>
<td>Workload:</td>
<td>SAP</td>
<td>Technologies:</td>
<td>Migration; Databases</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon EC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Summary

This pattern outlines the steps for migrating SAP workloads from any SAP source database to an SAP target database on Amazon Elastic Compute Cloud (Amazon EC2) by using CloudEndure Migration.

SAP workloads include SAP Customer Relationship Management (CRM), SAP Enterprise Resource Planning (ERP), and SAP S/4HANA.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the Amazon Web Services (AWS) Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.
Prerequisites and limitations

Prerequisites

- An active AWS account
- An SAP Adaptive Server Enterprise (ASE) source database for Linux or Windows, in an on-premises data center

Limitations

- The target operating system must be supported by Amazon EC2; for more information, see Amazon EC2 FAQs.

Product versions

- SAP ASE version 15.7 and later

Architecture

Source technology stack

- An SAP ASE database

Target technology stack

- Amazon EC2

Data migration architecture
Tools

- **CloudEndure Migration** – CloudEndure Migration simplifies, expedites, and reduces the cost of cloud migration by automating large-scale migration to AWS. Continuous data replication takes place in the background, without application disruption or performance impact, which ensures that data is synced in real time and minimizes cutover windows. When migration cutover is initiated, CloudEndure runs a highly automated machine conversion and orchestration process, allowing even the most complex applications and databases to run natively on AWS, without compatibility issues, and requires only minimal IT skills.
Epics

Sign up for CloudEndure through AWS Marketplace

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In AWS Marketplace, search for and subscribe to CloudEndure Migration.</td>
<td>For a link to AWS Marketplace and for more information about the steps described in subsequent stories, see the &quot;Related resources&quot; section at the end of this pattern.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Register an account and confirm your email address.</td>
<td></td>
<td>General AWS</td>
</tr>
<tr>
<td>Set the account password and accept the terms and conditions.</td>
<td>Password must be at least 8 characters long and must contain one uppercase letter, one lowercase letter, one digit, and one special character.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Sign in to CloudEndure and create a project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure user console.</td>
<td>On the CloudEndure console, sign in with the credentials you created in the previous step.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Create a new project.</td>
<td>Select the plus (+) button in the upper-left corner of the CloudEndure user console to create a new project.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Enter a project name and select a project type and license package.</td>
<td>Select &quot;Live Migration&quot; as the project type. You can acquire a license through AWS Marketplace.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

Generate and use AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the AWS Identity and Access Management (IAM) console, generate an access key ID and secret access key.</td>
<td>To generate the required AWS credentials to use with the CloudEndure user console, you need to create at least one IAM user and assign the proper permission policy to this user. The user console requires an access key ID and a secret access key.</td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
### Migrate from SAP ASE to Amazon EC2 using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose &quot;Policies,&quot; and then choose &quot;Create policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the &quot;JSON&quot; tab, copy the CloudEndure IAM policy into the field, and then choose &quot;Review policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Enter a name and description for the policy, and then choose &quot;Create policy.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create a new IAM user.</td>
<td>In the IAM console, navigate to Users, choose &quot;Add user,&quot; assign the user a name, and then set the &quot;Access type&quot; to Programmatic access.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose &quot;Next: Permissions&quot; and then under &quot;Set permissions for user,&quot; choose &quot;Attach existing policies.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Locate the policy you created, check the box to the left of the policy name, and then choose &quot;Next: Review.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Verify that the fields are correct, and then choose &quot;Create user.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy the access key ID and secret access key and store them in a safe location.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Sign in to the CloudEndure user console, and then select your SAP to AWS migration project.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the &quot;Setup &amp; Info&quot; tab from the left menu, and navigate to AWS credentials.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy and paste your AWS access key ID and secret access key into the corresponding fields, and then choose &quot;Save.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>

**Configure replication settings**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the CloudEndure user console, navigate to &quot;Setup</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

Migrate from SAP ASE to Amazon EC2 using CloudEndure

### Install the CloudEndure agents on the SAP source machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the CloudEndure user console, navigate to &quot;Machines,&quot; &quot;Machine Actions,&quot; &quot;Add Machines.&quot;</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Locate your agent installation token.</td>
<td>When you run the installer file on a source machine, you are first asked to enter your installation token. The token is a unique string of characters that is automatically generated for you when your CloudEndure account is activated. One installation token enables you to install the agent on as many source machines as your license package and project allow.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>For Linux machines, copy the installer command, log in to your source machine, and then run the installer.</td>
<td>For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>For Windows machines, download the installer file to each machine, and then run the installer command.</td>
<td>For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Wait as the machine undergoes initial data replication.</td>
<td>After the agent is installed, the source machine will appear on the &quot;Machines&quot; tab in the CloudEndure console.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

### Configure the target machine's blueprint

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the CloudEndure console, choose a machine name from</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns
### Migrate from SAP ASE to Amazon EC2 using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform a test</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
<tr>
<td>On the CloudEndure console, on the &quot;Machines&quot; tab, select the box to the left of each source machine.</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>In the &quot;Launch x Target Machines&quot; menu, choose &quot;Test Mode.&quot;</td>
<td>&quot;x&quot; represents the number of machines to launch.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Choose &quot;Continue&quot; when prompted. The machines will be launched.</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Verify that the test completed successfully.</td>
<td>The &quot;Machine&quot; tab will indicate that testing is complete: The edge of each tested &quot;Source&quot; machine will be marked in green. The orange flag icon under the &quot;Status&quot; column, which previously indicated that the machine had not been tested, will disappear (if no other issues are found). The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the &quot;Live Migration Lifecycle&quot; column, a message will appear stating that the machine has been tested recently.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Test the target machines by choosing each machine's name, opening the &quot;Target&quot; tab, copying the public IP address, and navigating to that IP address.</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>
Schedule and perform a cutover to the Amazon EC2 target

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule a cutover window.</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>On the &quot;Machines&quot; tab, check the box to the left of each source machine.</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>On the &quot;Launch x Target Machines&quot; menu, choose &quot;Cutover Mode.&quot;</td>
<td>&quot;x&quot; represents the number of machines to launch.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Choose &quot;Continue&quot; when prompted. The machines will be launched.</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Verify that the cutover completed successfully.</td>
<td>The &quot;Machines&quot; tab will indicate that the cutover is complete: The edge of each tested source machine will be marked in green. The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the &quot;Live Migration Lifecycle&quot; column, a message will appear stating that the machine has cut over.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

Related resources

Product page on AWS

- CloudEndure Migration

AWS Marketplace

- CloudEndure Migration

CloudEndure

- CloudEndure console
- CloudEndure IAM policy

CloudEndure documentation

- Signing in to the CloudEndure User Console
- Creating a CloudEndure Project
- Generating and Using AWS Credentials
- Configuring Replication Settings
- Installing CloudEndure Agents
Tutorials and videos

• CloudEndure Troubleshooting Playbook
• CloudEndure FAQs
• CloudEndure Video Guide

Migrate an on-premises Microsoft SQL Server database to Amazon EC2

Created by Mark Szalkiewicz (AWS)

<table>
<thead>
<tr>
<th>R Type: Rehost</th>
<th>Source: Databases: Relational</th>
<th>Target: Microsoft SQL Server on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Microsoft</td>
<td>AWS services: Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It covers two options for migration: using AWS Data Migration Service (AWS DMS) or using native Microsoft SQL Server tools such as backup and restore, Copy Database Wizard, or copy and attach database.

Prerequisites and limitations

Prerequisites

• An active AWS account
• An operating system supported by Amazon EC2 (for a complete list of supported operating system versions, see Amazon EC2 FAQs)
• A Microsoft SQL Server source database in an on-premises data center

Product versions

• Microsoft SQL Server versions 2005, 2008, 2008R2, 2012, 2014, 2016, and 2017 for the Enterprise, Standard, Workgroup, and Developer editions, if you're using AWS DMS. To migrate Microsoft SQL Server Web or Express edition, use native or third-party tools. For the latest list of supported versions, see Using a Microsoft SQL Server Database as a Target for AWS DMS.

Architecture

Source technology stack

• On-premises Microsoft SQL Server database
Target technology stack

- Microsoft SQL Server database on an EC2 instance

Target architecture

Data migration architecture

- Using AWS DMS
• Using native SQL Server tools
Tools

- **AWS DMS - AWS Data Migration Service** (AWS DMS) helps you migrate your data to and from widely used commercial and open-source databases, including Oracle, SQL Server, MySQL, and PostgreSQL. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.

- **Native Microsoft SQL Server tools** - These include backup and restore, Copy Database Wizard, and copy and attach database.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the target operating system version.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance based on the Microsoft SQL Server</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
### Migrate from SQL Server to Amazon EC2

#### Task
**Description**

<table>
<thead>
<tr>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>compatibility list and capacity requirements.</td>
</tr>
<tr>
<td>Identify the storage requirements for type and capacity.</td>
</tr>
<tr>
<td>Identify the network requirements, including latency and bandwidth.</td>
</tr>
<tr>
<td>Choose the EC2 instance type based on capacity, storage features, and network features.</td>
</tr>
<tr>
<td>Identify the network and host access security requirements for the source and target databases.</td>
</tr>
<tr>
<td>Identify a list of users required for the Microsoft SQL Server software installation.</td>
</tr>
<tr>
<td>Determine the backup strategy.</td>
</tr>
<tr>
<td>Determine the availability requirements.</td>
</tr>
<tr>
<td>Identify the application migration and cutover strategy.</td>
</tr>
</tbody>
</table>

#### Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Create security groups and network access control list (ACL).</td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Configure and start an EC2 instance.</td>
<td>SysAdmin</td>
<td></td>
</tr>
</tbody>
</table>

#### Install the software

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the users and groups required for Microsoft SQL Server software.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Download the Microsoft SQL Server software.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Install Microsoft SQL Server software on the EC2 instance and configure the server.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate the data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native Microsoft SQL Server tools or third-party tools to migrate the database objects and data.</td>
<td>Tools include backup and restore, Copy Database Wizard, and copy and attach database.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data by using AWS DMS.</td>
<td>For detailed information on using AWS DMS, see the links in the References and Help section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td>Use AWS Schema Conversion Tool (AWS SCT) to analyze and modify SQL code embedded in application source code.</td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application switch-over strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down all temporary AWS resources.</td>
<td>Temporary resources include the AWS DMS replication instance and the EC2 instance for AWS SCT.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Migrate from SQL Server to Amazon EC2 using CloudEndure

<table>
<thead>
<tr>
<th><strong>Task</strong></th>
<th><strong>Description</strong></th>
<th><strong>Skills required</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close the project and provide feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Related resources

**References**

- Deploying Microsoft SQL Server on Amazon Web Services
- Amazon EC2
- Amazon EC2 FAQs
- AWS Database Migration Service
- Amazon EC2 pricing
- Microsoft Products on AWS
- Microsoft Licensing on AWS
- Microsoft SQL Server on AWS

**Tutorials and videos**

- Getting Started with Amazon EC2
- Getting Started with AWS Database Migration Service
- Add an Amazon EC2 Instance to Your Directory (Simple AD and Microsoft AD)
- AWS Database Migration Service (video)
- Introduction to Amazon EC2 - Elastic Cloud Server & Hosting with AWS (video)

### Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using CloudEndure

*Created by Pavel Rubin (AWS)*

<table>
<thead>
<tr>
<th><strong>Environment:</strong></th>
<th>Production</th>
<th><strong>Source:</strong> Databases: Relational</th>
<th><strong>Target:</strong> Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R Type:</strong></td>
<td>Rehost</td>
<td><strong>Workload:</strong> Microsoft</td>
<td><strong>Technologies:</strong> Migration; Databases</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon EC2

### Summary

This pattern provides guidance for migrating a Microsoft SQL Server database from an on-premises data center host to an Amazon Elastic Compute Cloud (Amazon EC2) instance using CloudEndure Migration.
**Note**: AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the Amazon Web Services (AWS) Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account

**Product versions**


**Architecture**

**Source technology stack**

- Microsoft SQL Server database in an on-premises data center

**Target technology stack**

- Microsoft SQL Server database on an EC2 instance (for a complete list of supported operating system versions, see Amazon EC2 FAQs)

**Source and target architecture**

*CloudEndure migration:*

![CloudEndure Migration Diagram](image-url)
AWS Prescriptive Guidance Patterns
Migrate from SQL Server to
Amazon EC2 using CloudEndure

Network architecture:

Tools

- **CloudEndure Migration** - CloudEndure Migration helps you simplify, expedite, and automate large-scale migration to AWS. Continuous Data Replication takes place in the background, without
application disruption or performance impact, which ensures that data is synced in real-time and minimizes cutover windows. When migration cutover is initiated, CloudEndure runs a highly automated machine conversion and orchestration process, allowing even the most complex applications and databases to run natively on AWS, without compatibility issues and with minimal IT skills necessary.

Epics

Sign up for CloudEndure through AWS Marketplace

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate to AWS Marketplace, search for CloudEndure Migration, and subscribe.</td>
<td>For a link to AWS Marketplace and for more information about the steps described in subsequent stories, see the &quot;Related resources&quot; section at the end of this pattern.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Register the account and confirm the email.</td>
<td></td>
<td>General AWS</td>
</tr>
<tr>
<td>Set the account password and accept terms and conditions.</td>
<td>Passwords must be at least 8 characters long and must contain one uppercase letter, one lowercase letter, one digit, and one special character.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Sign in to CloudEndure and create a project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure user console.</td>
<td>On the CloudEndure console, sign in with the credentials you created in the previous step.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Create a new project.</td>
<td>Choose the plus (+) button in the upper-left corner of the console to create a project.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Enter a project name and select a project type and license package.</td>
<td>Select &quot;Live Migration&quot; as the project type. You can acquire a license through AWS Marketplace.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

Generate and use AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the AWS Identity and Access Management (IAM) console, generate an access key ID and secret access key.</td>
<td>To generate the required AWS credentials to use with the CloudEndure user console, you need to create at least one IAM user and assign the proper</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Create a new IAM user.</td>
<td>On the IAM console, navigate to Users, choose &quot;Add user,&quot; assign the user a name, and then set the &quot;Access type&quot; to Programmatic access.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose &quot;Next: Permissions&quot; and then under &quot;Set permissions for user,&quot; choose &quot;Attach existing policies.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Locate the policy you created, check the box to the left of the policy name, and then choose &quot;Next: Review.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Verify that the fields are correct, and then choose &quot;Create user.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy the access key ID and secret access key and store them in a safe location.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Sign in to the CloudEndure user console, and select your migration project.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Choose the &quot;Setup &amp; Info&quot; tab from the left menu, and navigate to AWS credentials.</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Copy and paste your AWS access key ID and secret access key into the corresponding fields, and then choose &quot;Save.&quot;</td>
<td></td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
## Configure replication settings

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure user console, navigate to “Setup &amp; Info,” and then choose “Replication settings.”</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>For &quot;Live Migration Source,&quot; choose &quot;Other Infrastructure.&quot;</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Define your replication servers.</td>
<td>See the CloudEndure documentation links in the &quot;Related resources&quot; section for detailed instructions.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

## Install CloudEndure agents on Microsoft SQL Server source machines

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the CloudEndure user console, navigate to “Machines,&quot; &quot;Machine Actions,&quot; “Add Machines.&quot;</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Locate your agent installation token.</td>
<td>When you run the installer file on a source machine, you are first asked to enter your installation token. The token is a unique string of characters that is automatically generated for you when your CloudEndure account is activated. One installation token enables you to install the agent on as many source machines as your license package and project allow.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>For Linux machines, copy the installer command, log in to your source machines, and then run the installer.</td>
<td>For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>For Windows machines, download the installer file to each machine, and then run the installer command.</td>
<td>For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Wait as the machine undergoes initial data replication.</td>
<td>After the agent is installed, the source machine will appear on the &quot;Machines&quot; tab on the CloudEndure console.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>
Configure the target machine's blueprint

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the CloudEndure console, choose a machine name from the &quot;Machines&quot; tab to access the &quot;Machine Details&quot; pane.</td>
<td>You can configure a variety of Blueprint settings. For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Select the &quot;Blueprint&quot; tab from the menu.</td>
<td>You can configure a variety of Blueprint settings. For detailed instructions, see the CloudEndure documentation in the &quot;Related resources&quot; section.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

Perform a test

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the CloudEndure console, on the &quot;Machines&quot; tab, select the box to the left of each source machine.</td>
<td>&quot;x&quot; represents the number of machines to launch.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>On the &quot;Launch x Target Machines&quot; menu, choose &quot;Test Mode.&quot;</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Choose &quot;Continue&quot; when prompted. The machines will be launched.</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Verify that the test completed successfully.</td>
<td>The &quot;Machine&quot; tab will indicate that testing is complete: The edge of each tested &quot;Source&quot; machine will be marked in green. The orange flag icon under the &quot;Status&quot; column, which previously indicated that the machine had not been tested, will disappear (if no other issues are found). The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the &quot;Live Migration Lifecycle&quot; column, a message will appear stating that the machine has been tested recently.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

Test the target machines by choosing each machine's name, opening the "Target" tab, copying the public IP address, | CloudEndure admin |
## AWS Prescriptive Guidance Patterns
### Migrate from SQL Server to Amazon EC2 using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>and navigating to that IP address.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Schedule and perform a cutover**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule a cutover window.</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>On the &quot;Machines&quot; tab, select the box to the left of each source machine.</td>
<td></td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>On the &quot;Launch x Target Machines&quot; menu, choose &quot;Cutover Mode.&quot;</td>
<td>&quot;x&quot; represents the number of machines to launch.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Choose &quot;Continue&quot; when prompted. The machines will be launched.</td>
<td>You can review the launch process on the &quot;Job Progress&quot; tab.</td>
<td>CloudEndure admin</td>
</tr>
<tr>
<td>Verify that the cutover completed successfully.</td>
<td>The &quot;Machines&quot; tab will indicate that the cutover is complete: The edge of each tested source machine will be marked in green. The purple icon, which indicates that a target machine has been launched for the selected source machine, will appear. In the &quot;Live Migration Lifecycle&quot; column, a message will appear stating that the machine has cut over.</td>
<td>CloudEndure admin</td>
</tr>
</tbody>
</table>

**Related resources**

**AWS Marketplace**
- CloudEndure Migration

**CloudEndure**
- CloudEndure console
- CloudEndure IAM policy

**CloudEndure documentation**
- Signing in to the console
- Creating a project
- Generating and using credentials
- Configuring replication settings
Migrate an on-premises MariaDB database to Amazon EC2 using MySQL tools

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Rehost</th>
<th>Source: Databases: Relational</th>
<th>Target: MariaDB on Amazon EC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate an on-premises MariaDB source database to a MariaDB database on an Amazon Elastic Compute Cloud (Amazon EC2) instance. This pattern includes the use of native MySQL tools such as mysqldump and mysql.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A MariaDB source database in on-premises data center

Limitations

- For a list of target operating systems supported by Amazon EC2, see Amazon EC2 FAQs.

Product versions

- MariaDB version 10.x
**Architecture**

**Source technology stack**
- On-premises MariaDB database

**Target technology stack**
- MariaDB database on an EC2 instance

**Target architecture**

AWS data migration architecture
Tools

- Native MySQL tools: `mysqldump` and `mysql`

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the target operating system version.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance based on the MariaDB compatibility list and capacity requirements.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Identify network requirements such as latency and bandwidth.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>General AWS</td>
</tr>
<tr>
<td>Identify the network or host access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
<tr>
<td>Identify a list of operating system users required for MariaDB software installation.</td>
<td></td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
<tr>
<td>Determine a backup strategy.</td>
<td></td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
<tr>
<td>Determine availability requirements.</td>
<td></td>
<td>DBA, General AWS</td>
</tr>
<tr>
<td>Identify the application migration or switchover strategy.</td>
<td></td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
</tbody>
</table>

**Configure the infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td></td>
<td>General AWS</td>
</tr>
<tr>
<td>Create security groups and network access control lists (ACLs).</td>
<td></td>
<td>General AWS</td>
</tr>
<tr>
<td>Configure and start an EC2 instance.</td>
<td></td>
<td>General AWS</td>
</tr>
</tbody>
</table>

**Install MariaDB software**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the operating system users and groups required for the MariaDB software to work.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Download the required version of the MariaDB software.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Install the MariaDB software on the EC2 instance and configure the server.</td>
<td></td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
</tbody>
</table>
Migrate data from the source database to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native MySQL tools or third-party tools to migrate database objects and data.</td>
<td>In the source database, use mysqldump to create an output file that contains database objects and data. In the target database, use mysql to restore the data.</td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
<tr>
<td>Validate the data.</td>
<td>Check the source and target databases to confirm that the data migration was successful.</td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
</tbody>
</table>

Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner, General AWS</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application cutover or switchover strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner, General AWS</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin, General AWS</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner, General AWS</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close the project and provide feedback.</td>
<td></td>
<td>SysAdmin, App owner, Customer, DBA</td>
</tr>
</tbody>
</table>

Related resources

- Amazon EC2
- Amazon EC2 Pricing
Migrate an on-premises MySQL database to Amazon EC2

Created by Sergey Dmitriev (AWS)

Summary

This pattern provides guidance for migrating an on-premises MySQL database to a MySQL database on an Amazon Elastic Compute Cloud (Amazon EC2) instance. The pattern discusses the use of AWS Database Migration Service (AWS DMS) or native MySQL tools such as `mysqldbcopy` and `mysqldump` for the migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A MySQL source database in an on-premises data center

Product versions

- MySQL versions 5.5, 5.6, and 5.7
- For a list of target operating systems supported by Amazon EC2, see Amazon EC2 FAQs

Architecture

Source technology stack

- An on-premises MySQL database

Target technology stack

- A MySQL database instance on Amazon EC2

AWS data migration methods

- AWS DMS
- Native MySQL tools (`mysqldbcopy`, `mysqldump`)
Target architecture

AWS data migration architecture

Using AWS DMS:
Using native MySQL tools:
Tools

- **AWS DMS - AWS Database Migration Service** (AWS DMS) supports several source and target databases. For information about MySQL source and target databases supported by AWS DMS, see [Migrating MySQL-Compatible Databases to AWS](#). If your source database isn't supported by AWS DMS, you must choose another method to migrate your data.

- **Native MySQL tools - mysqldbcopy and mysqldump**

### Epics

**Plan the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Identify the target operating system version.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance based on the MySQL compatibility list and capacity requirements.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network requirements such as latency and bandwidth.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network or host access security requirements for the source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify a list of operating system users required for MySQL software installation.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Determine a backup strategy.</td>
<td>DBA</td>
</tr>
<tr>
<td>Determine availability requirements.</td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the application migration or switchover strategy.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC) and subnets.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups and network access control lists (ACLs).</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an EC2 instance.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Install MySQL software

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the operating system users and groups required for the MySQL software to work.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Download the required version of the MySQL software.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Install the MySQL software on the EC2 instance and configure the server.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
Migrate data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native MySQL tools or third-party tools to migrate database objects and data.</td>
<td>These tools include mysqldbcopy and mysqldump.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate data with AWS DMS.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application cutover or switchover strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td>Shut down the AWS DMS replication instance.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

References
Rehost on-premises workloads in the AWS Cloud: migration checklist

Created by Srikanth Rangavajhala (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>On-premises workloads</th>
<th>Target:</th>
<th>AWS Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Rehost</td>
<td>Technologies:</td>
<td>Migration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

Rehosting on-premises workloads in the Amazon Web Services (AWS) Cloud involves the following migration phases: planning, pre-discovery, discovery, build, test, and cutover. This pattern outlines the phases and their related tasks. The tasks are described at a high level and support about 75% of all application workloads. You can implement these tasks over two to three weeks, in an agile sprint cycle.

You should review and vet these tasks with your migration team and consultants. After the review, you can gather the input, eliminate or re-evaluate tasks as necessary to meet your requirements, and modify other tasks to support at least 75% of the application workloads in your portfolio. You can then use an agile project management tool such as Rally or Atlassian Jira to import the tasks, assign them to resources, and track your migration activities.

The pattern assumes that you’re using CloudEndure Migration to rehost your workloads, but you can use your migration tool of choice.

Prerequisites and limitations

Prerequisites

- Project management tool for tracking migration tasks (for example, Rally or Atlassian Jira)
- Migration tool for rehosting your workloads on AWS (for example, CloudEndure Migration)

Architecture

Source platform

- On-premises source stack (including technologies, applications, databases, and infrastructure)
Target platform

- AWS Cloud target stack (including technologies, applications, databases, and infrastructure)

Architecture

The following diagram illustrates rehosting (discovering and migrating servers from an on-premises source environment to AWS) by using CloudEndure Migration.

Tools

- You can use a migration and project management tool of your choice.

Epics

Planning phase

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groom the pre-discovery backlog.</td>
<td>Conduct the pre-discovery backlog grooming working session with department leads and application owners.</td>
<td>Project manager, Agile scrum leader</td>
</tr>
<tr>
<td>Conduct the sprint planning working session.</td>
<td>In this scoping exercise, you distribute the applications that you want to migrate across sprints and waves.</td>
<td>Project manager, Agile scrum leader</td>
</tr>
</tbody>
</table>
## Pre-discovery phase

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm application knowledge.</td>
<td>Confirm and document the application owner and their knowledge of the application. Determine whether there's another point person for technical questions.</td>
<td>Migration specialist (interviewer)</td>
</tr>
<tr>
<td>Determine application compliance requirements.</td>
<td>Confirm with the application owner that the application doesn't have to comply with requirements for Payment Card Industry Data Security Standard (PCI DSS), Sarbanes-Oxley Act (SOX), personally identifiable information (PII), or other standards. If compliance requirements exist, teams must finish their compliance checks on the servers that will be migrated.</td>
<td>Migration specialist (interviewer)</td>
</tr>
<tr>
<td>Confirm production release requirements.</td>
<td>Confirm the requirements for releasing the migrated application to production (such as release date and downtime duration) with the application owner or technical contact.</td>
<td>Migration specialist (interviewer)</td>
</tr>
<tr>
<td>Get server list.</td>
<td>Get the list of servers that are associated with the targeted application.</td>
<td>Migration specialist (interviewer)</td>
</tr>
<tr>
<td>Get the logical diagram that shows the current state.</td>
<td>Obtain the current state diagram for the application from the enterprise architect or the application owner.</td>
<td>Migration specialist (interviewer)</td>
</tr>
<tr>
<td>Create a logical diagram that shows the target state.</td>
<td>Create a logical diagram of the application that shows the target architecture on AWS. This diagram should illustrate servers, connectivity, and mapping factors.</td>
<td>Enterprise architect, Business owner</td>
</tr>
<tr>
<td>Get server information.</td>
<td>Collect information about the servers that are associated with the application, including their configuration details.</td>
<td>Migration specialist (interviewer)</td>
</tr>
<tr>
<td>Add server information to the discovery template.</td>
<td>Add detailed server information to the application discovery template (mobilize-application-questionnaire.xlsx in the Attachments section). This</td>
<td>Migration specialist (interviewer)</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

Rehost on-premises workloads on AWS: migration checklist

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>template includes all the application-related security, infrastructure, operating system, and networking details.</td>
<td></td>
</tr>
<tr>
<td>Publish the application discovery template.</td>
<td>Share the application discovery template with the application owner and migration team for common access and use.</td>
<td>Migration specialist (interviewer)</td>
</tr>
</tbody>
</table>

### Discovery phase

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm server list.</td>
<td>Confirm the list of servers and the purpose of each server with the application owner or technical lead.</td>
<td>Migration specialist</td>
</tr>
<tr>
<td>Identify and add server groups.</td>
<td>Identify server groups such as web servers or application servers, and add this information to the application discovery template. Select the tier of the application (web, application, database) that each server should belong to.</td>
<td>Migration specialist</td>
</tr>
<tr>
<td>Fill in the application discovery template.</td>
<td>Complete the details of the application discovery template, with the help of migration team, application team, and AWS.</td>
<td>Migration specialist</td>
</tr>
<tr>
<td>Add missing server details (middleware and OS teams).</td>
<td>Ask middleware and operating system (OS) teams to review the application discovery template and add any missing server details, including database information.</td>
<td>Migration specialist</td>
</tr>
<tr>
<td>Get inbound/outbound traffic rules (network team).</td>
<td>Ask the network team to get the inbound/outbound traffic rules for the source and destination servers. The network team should also add existing firewall rules, export these to a security group format, and add existing load balancers to the application discovery template.</td>
<td>Migration specialist</td>
</tr>
<tr>
<td>Identify required tagging.</td>
<td>Determine the tagging requirements for the application.</td>
<td>Migration specialist</td>
</tr>
<tr>
<td>Create firewall request details.</td>
<td>Capture and filter the firewall rules that are required to</td>
<td>Migration specialist, Solutions architect, Network lead</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

Rehost on-premises workloads on AWS: migration checklist

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
<tr>
<td>Communicate with the application.</td>
<td></td>
<td>Migration specialist, Solutions architect, Network lead</td>
</tr>
<tr>
<td>Update the EC2 instance type.</td>
<td>Update the Amazon Elastic Compute Cloud (Amazon EC2) instance type to be used in the target environment, based on infrastructure and server requirements.</td>
<td>Migration specialist, Solutions architect, Network lead</td>
</tr>
<tr>
<td>Identify the current state diagram.</td>
<td>Identify or create the diagram that shows the current state of the application. This diagram will be used in the information security (InfoSec) request.</td>
<td>Migration specialist, Solutions architect</td>
</tr>
<tr>
<td>Finalize the future state diagram.</td>
<td>Finalize the diagram that shows the future (target) state for the application. This diagram will also be used in the InfoSec request.</td>
<td>Migration specialist, Solutions architect</td>
</tr>
<tr>
<td>Create firewall or security group service requests.</td>
<td>Create firewall or security group service requests (for development/QA, pre-production, and production). If you’re using CloudEndure Migration, include CloudEndure replication ports, if they’re not already open.</td>
<td>Migration specialist, Solutions architect, Network lead</td>
</tr>
<tr>
<td>Review firewall or security group requests (InfoSec team).</td>
<td>In this step, the InfoSec team reviews and approves the firewall or security group requests that were created in the previous step.</td>
<td>InfoSec engineer, Migration specialist</td>
</tr>
<tr>
<td>Implement firewall security group requests (network team).</td>
<td>After the InfoSec team approves the firewall requests, the network team implements the required inbound/outbound firewall rules.</td>
<td>Migration specialist, Solutions architect, Network lead</td>
</tr>
</tbody>
</table>

**Build phase (repeat for development/QA, pre-production, production environments)**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudEndure project.</td>
<td>Create a CloudEndure project for application migration. For more information, see the CloudEndure documentation.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Create a service request to install CloudEndure Agent.</strong></td>
<td>Create a service request to install CloudEndure Agent on development/QA, pre-production, or production machines.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td><strong>Install CloudEndure Agent.</strong></td>
<td>Install CloudEndure Agent on development/QA, pre-production, or production machines. To install CloudEndure on multiple machines, see the pattern <a href="https://aws.amazon.com/cloudendure/docs/installation-guide/system-manager.html">Install CloudEndure Agent through AWS Systems Manager on Windows and Linux</a>. For additional information, see the CloudEndure documentation.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td><strong>Validate continuous replication status.</strong></td>
<td>When you install CloudEndure Agent, the replication instance initiates continuous data replication. You can validate the replication status on development/QA, pre-production, or production machines.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td><strong>Validate and complete Blueprint settings.</strong></td>
<td>Validate and finish configuring Blueprint settings in CloudEndure for the application servers in the development/QA, pre-production, and production environments. The Blueprint provides the instructions for creating and launching a target machine for a specified source machine. For more information, see the CloudEndure documentation.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
</tbody>
</table>

**Test phase (repeat for development/QA, pre-production, and production environments)**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a service request.</strong></td>
<td>Create a service request for the infrastructure team and other teams to perform application cutover to development/QA, pre-production, or production instances.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
</tbody>
</table>
### Rehost on-premises workloads on AWS: migration checklist

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure a load balancer (optional).</td>
<td>Configure required load balancers, such as an Application Load Balancer or an F5 load balancer with iRules.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Create EC2 instances.</td>
<td>Create EC2 instances by using the CloudEndure Blueprint, with the appropriate tags from the application discovery template. For more information, see the CloudEndure documentation.</td>
<td>Migration Specialist, Cloud Administrator</td>
</tr>
<tr>
<td>Modify DNS entries.</td>
<td>Modify Domain Name System (DNS) entries. (Use resolv.conf or host.conf for a Microsoft Windows environment.) Configure each EC2 instance to point to the new IP address of this host.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Test connectivity to backend hosts from EC2 instances.</td>
<td>Check the logins by using the domain credentials for the migrated servers.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Update the DNS A record.</td>
<td>Update the DNS A record for each host to point to the new Amazon EC2 private IP address.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Update the DNS CNAME record.</td>
<td>Update the DNS CNAME record for virtual IPs (load balancer names) to point to the cluster for web and application servers.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Test the application in applicable environments.</td>
<td>Log in to the new EC2 instance and test the application in the development/QA, pre-production, and production environments.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
</tbody>
</table>

**Cutover phase**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create production deployment plan.</td>
<td>Create a production deployment plan (including a backout plan).</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Notify operations team of downtime.</td>
<td>Notify the operations team of the downtime schedule for the servers. Some teams might require a change request or service request (CR/SR) ticket for this notification.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns
#### Relocate

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicate production machines.</td>
<td>Replicate production machines by using CloudEndure Migration (or other migration tool).</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Cut over to the cloud.</td>
<td>Perform cutover activities for the production environment.</td>
<td>Migration specialist, Cloud administrator</td>
</tr>
<tr>
<td>Shut down on-premises servers.</td>
<td>Shut down the on-premises servers that were migrated to AWS.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

**Related resources**

**References**
- How to migrate
- Migrating workloads to the AWS Cloud by using CloudEndure Migration
- AWS CloudEndure Migration Factory Solution
- Automating large-scale server migrations with CloudEndure Migration Factory
- AWS Migration Acceleration Program

**Tools**
- Migration tool comparison
- CloudEndure Migration subscription

**Attachments**
- attachment.zip

---

**Relocate**

**Topics**
- Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication (p. 1100)
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX (p. 1105)
- Migrate an Amazon RDS DB instance to another VPC or account (p. 1108)
- Migrate an Amazon RDS for Oracle DB instance to another VPC (p. 1111)
- Migrate an Amazon Redshift cluster to an AWS Region in China (p. 1114)
- Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport (p. 1124)
Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication

Summary

This pattern walks you through the steps for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle source database to a different AWS account and AWS Region. The pattern uses a DB snapshot for a one-time full data load, and enables AWS Database Migration Service (AWS DMS) for ongoing replication.

Prerequisites and limitations

Prerequisites

• An active AWS account that contains the source Amazon RDS for Oracle database, which has been encrypted using a non-default AWS Key Management Service (AWS KMS) key
• An active AWS account in a different AWS Region from the source database, to use for the target Amazon RDS for Oracle database
• Virtual private cloud (VPC) peering between the source and target VPCs
• Familiarity with using an Oracle database as a source for AWS DMS
• Familiarity with using an Oracle database as a target for AWS DMS

Product versions

• Oracle versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions and editions, see Using an Oracle Database as a Source for AWS DMS and with Using an Oracle database as a target for AWS DMS in the AWS documentation. For Oracle versions supported by Amazon RDS, see Oracle on Amazon RDS.

Architecture

Source and target technology stacks

• Amazon RDS for Oracle DB instance
Ongoing replication architecture

Tools

Tools used for one-time full data load:

- **Amazon RDS DB snapshot** - Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. When you create a DB snapshot, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. The amount of time it takes to create a snapshot varies with the size of your databases. Because the snapshot includes the entire storage volume, the size of files, such as temporary files, also affects the amount of time it takes to create the snapshot. For more information about using DB snapshots, see Creating a DB Snapshot in the Amazon RDS documentation.

- **KMS key for Amazon RDS encryption** - When you create an encrypted DB instance, you can also supply the KMS key identifier for your encryption key. If you don't specify a KMS key identifier, Amazon RDS uses your default encryption key for your new DB instance. AWS KMS creates your default encryption key for your AWS account. Your AWS account has a different default encryption key for each AWS Region. For this pattern, the Amazon RDS DB instance should be encrypted using the
non-default KMS key. For more information about using KMS keys for Amazon RDS encryption, see Encrypting Amazon RDS Resources in the Amazon RDS documentation.

Tools used for ongoing replication:

- **AWS DMS** - This pattern uses AWS DMS to replicate ongoing changes and to keep the source and target databases in sync. For more information about using AWS DMS for ongoing replication, see Working with an AWS DMS Replication Instance in the AWS DMS documentation.

**Epics**

Configure your source AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare the source Oracle DB instance</td>
<td>Let the Amazon RDS for Oracle DB instance run in ARCHIVELOG mode, and set the retention period. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Amazon-Managed">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Amazon-Managed</a>.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set supplemental logging for the source Oracle DB instance</td>
<td>Set database-level and table-level supplemental logging for the Amazon RDS Oracle DB instance. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Amazon-Managed">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Amazon-Managed</a>.</td>
<td>DBA</td>
</tr>
<tr>
<td>Update the KMS key policy in the source account</td>
<td>Update the KMS key policy in the source AWS account to allow the target AWS account to use the encrypted Amazon RDS KMS key. For details, see <a href="https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html#key-policy-modifying-external-accounts">https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html#key-policy-modifying-external-accounts</a>.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create a manual Amazon RDS DB snapshot of the source DB instance</td>
<td></td>
<td>AWS IAM user</td>
</tr>
<tr>
<td>Share the manual, encrypted Amazon RDS snapshot with the target AWS account</td>
<td>For details, see <a href="https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ShareSnapshot.html">https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ShareSnapshot.html</a>.</td>
<td>AWS IAM user</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Migrate Amazon RDS for Oracle to another AWS Region and account

#### Configure your target AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach a policy.</td>
<td>In the target AWS account, attach an AWS Identity and Access Management (IAM) policy to the root IAM user, to allow the IAM user to copy an encrypted DB snapshot using the shared AWS KMS key.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Switch to the source AWS Region.</td>
<td></td>
<td>AWS IAM user</td>
</tr>
<tr>
<td>Copy the snapshot.</td>
<td>In the Amazon RDS console, in the Snapshots pane, choose Shared with Me, and select the shared snapshot. Copy the snapshot to the same AWS Region as the source database by using the Amazon Resource Name (ARN) for the KMS key used by the source database. For details, see <a href="https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html">https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html</a>.</td>
<td>AWS IAM user</td>
</tr>
<tr>
<td>Switch to the target AWS Region, and create a new KMS key.</td>
<td></td>
<td>AWS IAM user</td>
</tr>
<tr>
<td>Copy the snapshot.</td>
<td>Switch to the source AWS Region. In the Amazon RDS console, in the Snapshots pane, choose Owned by Me, and select the copied snapshot. Copy the snapshot to the target AWS Region by using the KMS key for the new target AWS Region.</td>
<td>IAM user</td>
</tr>
<tr>
<td>Restore the snapshot.</td>
<td>Switch to the target AWS Region. In the Amazon RDS console, in the Snapshots pane, choose Owned by Me. Select the copied snapshot and restore it to an Amazon RDS for Oracle DB instance. For details, see <a href="https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_RestoreFromSnapshot.html">https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_RestoreFromSnapshot.html</a>.</td>
<td>AWS IAM user</td>
</tr>
</tbody>
</table>
Prepare your source database for ongoing replication

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Oracle user with the appropriate permissions.</td>
<td>Create an Oracle user with the required privileges for Oracle as a source for AWS DMS. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html</a>.</td>
<td>DBA</td>
</tr>
<tr>
<td>Configure the source database for Oracle LogMiner or Oracle Binary Reader.</td>
<td>For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html</a>.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Prepare your target database for ongoing replication

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Oracle user with the appropriate permissions.</td>
<td>Create an Oracle user with the required privileges for Oracle as a target for AWS DMS. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.Oracle.html#CHAP_Target.Oracle.Privileges">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.Oracle.html#CHAP_Target.Oracle.Privileges</a>.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Create AWS DMS components

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a replication instance in the target AWS Region.</td>
<td>Create a replication instance in the VPC of the target AWS Region. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_GettingStarted.html#CHAP_GettingStarted.ReplicationInstance">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_GettingStarted.html#CHAP_GettingStarted.ReplicationInstance</a>.</td>
<td>IAM user</td>
</tr>
<tr>
<td>Create source and target endpoints with required encryption, and test connections.</td>
<td>For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_GettingStarted.html#CHAP_GettingStarted.Endpoints">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_GettingStarted.html#CHAP_GettingStarted.Endpoints</a>.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create replication tasks.</td>
<td>For the migration type, choose ongoing replication. For the change data capture (CDC) start point, use the Oracle system change number (SCN) when the Amazon RDS snapshot was taken for full load, or the timestamp when the full load was taken. For TargetTablePrepMode, choose</td>
<td>IAM user</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>DO NOTHING.</strong> If the task has large binary object (LOB) data tables, choose Limited LOB mode, and set the max LOB size to the maximum size of the LOB data in the table. Enable logging. Group tables that are related through keys into a single task. If there are tables with a large amount of LOB data and the table has no relationship with other tables, create a separate task for it with the LOB settings described previously. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_GettingStarted.html#CHAP_GettingStarted.Tasks">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_GettingStarted.html#CHAP_GettingStarted.Tasks</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Start the tasks and monitor them.</strong></td>
<td>For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Monitoring.html">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Monitoring.html</a>.</td>
<td>IAM user</td>
</tr>
<tr>
<td><strong>Enable validation on the task if needed.</strong></td>
<td>Note that enabling validation does have a performance impact on the replication. For details, see <a href="https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Validating.html">https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Validating.html</a>.</td>
<td>IAM user</td>
</tr>
</tbody>
</table>

**Related resources**

- Changing a KMS key policy
- Creating a manual Amazon RDS DB snapshot
- Sharing a manual Amazon RDS DB snapshot
- Copying a snapshot
- Restoring from an Amazon RDS DB snapshot
- Getting started with AWS DMS
- Using an Oracle database as a source for AWS DMS
- Using an Oracle database as a target for AWS DMS
- AWS DMS setup using VPC peering
- How do I share manual Amazon RDS DB snapshots or DB cluster snapshots with another AWS account? ([AWS Knowledge Center article](https://aws.amazon.com/documentation/dms/))

**Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX**
Summary

This pattern describes the use of VMware Hybrid Cloud Extension (HCX) to migrate your on-premises virtual machines (VMs) and applications to VMware Cloud on Amazon Web Services (AWS). The migration uses VMware enterprise-class software-defined data center (SDDC) software on the AWS Cloud to provide optimized access to AWS services.

VMware Cloud on AWS integrates compute, storage, and network virtualization products (vSphere, vSAN, and VMware NSX) with VMware vCenter server management, which is optimized to run on dedicated, elastic, bare-metal AWS infrastructure. The resulting infrastructure is low-maintenance, simplified, and hyper-converged.

With this service, IT teams can manage their cloud-based resources with familiar VMware tools. For more information, see VMware Cloud on AWS on the VMware website.

VMware HCX supports three types of cloud migrations:

- **Hybridity (data center extension):** Extending an existing, on-premises VMware SDDC to AWS to provide footprint expansion, on-demand capacity, a testing/development environment, and virtual desktops.
- **Cloud evacuation (data center-wide infrastructure refresh):** Consolidating data centers and moving completely to the AWS Cloud (including handling data center co-location or end of lease).
- **Application-specific:** Moving individual applications to the AWS Cloud to meet specific business needs.

Prerequisites and limitations

Prerequisites

- Sign up for an AWS account (required for VMware Cloud SDDC creation).
- Sign up for a My VMware account. Register at https://my.vmware.com/web/vmware/ and fill out all fields.
- Check the version of vCenter and hosts, and collect the number of VMs. If possible, ask for an RVTools export to display information about your virtual environments. We recommend vCenter version 6.0 or later.
- You must deploy distributed virtual switches if you want to extend data center networks (L2), test vMotion by using HCX, or analyze application dependency by using vRealize Network Insight.
- Pick a non-conflicting on-premises current management subnet network to create the SDDC on VMware Cloud on AWS.
- Validate HCX requirements by reviewing the prerequisites provided in the VMware HCX User Guide.
- Identify and group VMs for waves of migration. Check for VMs that you can use for testing.
- Collect any data about relative bandwidth consumption, WAN compression, and data transfer speed.

Notes
• No need for VMware NSX-V or NSX-T on premises.
• No additional costs for HCX (it’s included in VMware Cloud on AWS).

Architecture

The following diagram shows the HCX solution built on multiple component services. Each component supports a specific function in the HCX solution. For more information about each HCX component, see the blog post Migrating Workloads to VMware Cloud on AWS with Hybrid Cloud Extension (HCX).

Source technology stack
• On-premises VMs and applications managed by VMware vSphere

Target technology stack
• VMware Cloud on AWS

Tools
• VMware HCX – VMware HCX is a tool that you can use to migrate your applications and workloads across data centers and cloud environments. It is included with VMware Cloud on AWS.
Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose a migration strategy.</td>
<td>Decide whether you want to extend your data center (hybridity), move all your data centers (cloud evacuation), or move specific applications to AWS.</td>
<td>SysAdmin, App owner</td>
</tr>
<tr>
<td>Validate HCX requirements.</td>
<td>For migration information, review the VMware HCX User Guide.</td>
<td>SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Migrate to VMware Cloud on AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate your VMs or applications.</td>
<td>For more information, see Hybrid Migration with VMware HCX in the VMware documentation.</td>
<td>SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

- VMware Cloud on AWS: Getting Started
- Hybrid Migration with VMware HCX
- VMware HCX User Guide
- VMware Cloud on AWS Pricing
- VMware Cloud on AWS Roadmap

Migrate an Amazon RDS DB instance to another VPC or account

Created by Virender Singla (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>AWS services: Amazon RDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

This pattern provides guidance for migrating an Amazon Relational Database Service (Amazon RDS) DB instance from one virtual private cloud (VPC) to another in the same AWS account, or from one AWS account to another AWS account.

This pattern is useful if you want to migrate your Amazon RDS DB instances to another VPC or account for segregation or security reasons. (For example, you might want to place your application stack and database in different accounts.)

Because this migration causes database downtime, you need to plan accordingly. Migrating a DB instance to another AWS account causes more downtime than migrating it to another VPC, because taking a manual snapshot, sharing it, and restoring the snapshot to the target account takes time. Alternatively, you can use AWS Data Migration Service (AWS DMS) to minimize downtime for the change. However, this process can also be time-consuming, depending on the database changes and transaction rates.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) permissions required for the VPC, subnets, and Amazon RDS console.

Limitations

- Migrate during low peak times, because changes to a VPC cause a database reboot, resulting in application outages.
- Limitations when migrating Amazon RDS to another VPC:
  - Amazon RDS should not be in multiple Availability Zones.
  - Amazon RDS should not have any read replicas.
  - The subnet group created in the target VPC must have subnets from the Availability Zone where the source database runs.
- Limitations when migrating Amazon RDS to another AWS account:
  - Sharing snapshots encrypted with the default service key for Amazon RDS is currently not supported.

Architecture

Source technology stack

- Amazon RDS DB instances in a VPC / AWS account

Target technology stack

- Amazon RDS DB instances in another VPC / AWS account
Tools

- **AWS DMS** – If database downtime is a concern when migrating an Amazon RDS DB instance to another account, use AWS Data Migration Services (AWS DMS). This service provides data replication, which generally causes less than 5 minutes of outage time.

Epics

Migrate to a different VPC in the same AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new VPC.</td>
<td>In the Amazon VPC console, create a new VPC and subnets with the desired properties and IP address ranges.</td>
<td>Admin</td>
</tr>
<tr>
<td>Create a DB subnet group.</td>
<td>In the Amazon RDS console, choose &quot;Subnet groups,&quot; choose &quot;Create DB subnet group,&quot; and then enter the subnet name, description, and VPC ID.</td>
<td>Admin</td>
</tr>
<tr>
<td>Modify the Amazon RDS DB instance to choose a new subnet group and security group.</td>
<td>In the Amazon RDS console, under Network &amp; Security, choose the subnet group associated with the new VPC, and then choose the appropriate security group for that VPC. Choose &quot;Apply immediately.&quot;</td>
<td>Admin</td>
</tr>
</tbody>
</table>

Migrate to a different AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new VPC and subnet group in the target AWS account.</td>
<td>In the Amazon VPC console, create a new VPC with the desired properties and IP address ranges. Create subnets and then create DB subnet groups in the Amazon RDS console.</td>
<td>Admin</td>
</tr>
<tr>
<td>Take a manual snapshot of the database and share the snapshot with the target account.</td>
<td>Share the snapshot with the target AWS account by providing the target account ID.</td>
<td>Admin</td>
</tr>
<tr>
<td>Launch a new Amazon RDS DB instance from the shared snapshot in the new subnet group.</td>
<td></td>
<td>Admin</td>
</tr>
</tbody>
</table>
Related resources

- Amazon VPC documentation
- Amazon RDS documentation
- How do I change the VPC for an RDS DB instance? (AWS Knowledge Center article)
- How do I transfer ownership of Amazon RDS resources to a different AWS account? (AWS Knowledge Center article)
- How do I share manual Amazon RDS DB snapshots or DB cluster snapshots with another AWS account? (AWS Knowledge Center article)
- AWS DMS documentation

Migrate an Amazon RDS for Oracle DB instance to another VPC

Created by Pinesh Singal (AWS)

<table>
<thead>
<tr>
<th>Environment: PoC or pilot</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type: Relocate</td>
<td>Workload: Oracle</td>
<td>Technologies: Migration; Databases</td>
</tr>
<tr>
<td>AWS services: Amazon RDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This migration pattern provides step-by-step guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle database (DB) instance from one virtual private cloud (VPC) to another VPC in same Amazon Web Services (AWS) account. For example, you can use this pattern if your business requires that the database and the Amazon Elastic Compute Cloud (Amazon EC2) application server are in the same VPC.

The pattern describes an online migration strategy with almost no downtime for a multi-terabyte Oracle source database with a high number of transactions.

To move an Amazon RDS for Oracle DB instance to another VPC, you must change the Amazon RDS subnet group. This subnet group needs to be preconfigured with the new VPC and required subnets. During the VPC change from one network to another, the Amazon RDS instance reboots, so the database won't be accessible while the movement is in progress.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Two VPCs with private subnets
- An Amazon RDS for Oracle database instance (up and running), configured with inbound and outbound security groups

Limitations
A DB instance that spans multiple Availability Zones (Multi-AZ) is not supported. This pattern, however, provides a way to work around this limitation.

- The DB instance can’t be migrated while a read replica is turned on.
- The subnet group in the new VPC should be in the same Availability Zone as the database.
- Migration should occur during scheduled maintenance period or low-traffic times, because moving the DB to another VPC causes a database reboot, resulting in application outages for few minutes.

**Product versions**

- Amazon RDS for Oracle DB instance, 12.1.0.2 and later

**Architecture**

**Source technology stack**

- An Amazon RDS for Oracle 12.1.0.2.v22 DB instance in a VPC
- A VPC configured in a separate route table
- Amazon RDS subnet groups configured in a VPC
- Amazon RDS option groups (if needed)

**Target technology stack**

- Amazon RDS for Oracle database instance with version 12.1.0.2.v22 in another VPC
- Amazon VPC configured in separate route
- Amazon RDS Subnet Groups configured in new VPC
- Amazon RDS Option Groups (if needed)

**Source and target architecture**

The following diagram shows using the console to move the Amazon RDS for Oracle DB from a private subnet in one VPC to a private subnet in a different VPC.

1. Use the console to modify the source Amazon RDS for Oracle DB instance.
2. In the target VPC, modify the subnet group, and modify the option group if used.
## Tools

- **Amazon RDS** – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for a relational database and manages common database administration tasks. This pattern uses Amazon RDS for Oracle.

## Epics

**Change the configuration of the Amazon RDS for Oracle database in the existing VPC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a subnet group.</td>
<td>Configure a subnet group in Amazon RDS.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create an option group.</td>
<td>(Optional) Configure an option group in Amazon RDS.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Modify the Amazon RDS for Oracle DB instance.</td>
<td>Modify the database with the subnet group and option group.</td>
<td>General AWS, DBA</td>
</tr>
</tbody>
</table>
| Update the Oracle database, if necessary. | To migrate the source Amazon RDS for Oracle database, make the following changes:  
  - Remove read replicas, if they exist.  
  - Turn off the Multi-AZ feature, if it’s turned on. | General AWS     |

**Configure the Amazon RDS for Oracle database in the target VPC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a subnet group.</td>
<td>In Amazon RDS, configure a subnet group using the subnet of the new VPC and the Availability Zone of the database.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create an option group.</td>
<td>(Optional) Configure an option group in Amazon RDS.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
| Modify the Amazon RDS for Oracle database. | Modify the database with new subnet group and option group of the new VPC. You can apply these changes immediately or in a maintenance window. 
  The modification can take several minutes to complete. During the modification, you will see the following status changes: | General AWS, DBA |
### Task Description

- **moving-to-vpc**
- Configuring-enhanced-monitoring
- Modifying
- Available

The modification will attach the default security group of the new VPC. Attach a new security group as needed by Amazon RDS for Oracle.

- **Update the Amazon RDS for Oracle database, if necessary.**
  - After migrating to the target Amazon RDS for Oracle database in the new VPC, make the following modifications, if needed:
    - Turn on read replicas, if they existed in the source database.
    - Turn on the Multi-AZ feature, if it was turned on in the source database.
  - General AWS

- **Test application connectivity.**
  - Perform a database connectivity test from any application. Confirm that the modified Amazon RDS for Oracle DB in the new VPC is connected and is accessible from the application.
  - App owner

### Related resources

- Amazon VPC documentation
- VPCs and subnets
- Working with a DB instance in a VPC
- Amazon RDS documentation
- Oracle on Amazon RDS
- Amazon RDS console
- How do I change the VPC for an Amazon RDS DB instance?

### Migrate an Amazon Redshift cluster to an AWS Region in China

*Created by Jing Yan (AWS)*
**Summary**

This pattern provides a step-by-step approach to migrate an Amazon Redshift cluster to an AWS Region in China from another AWS Region.

This pattern uses SQL commands to recreate all the database objects, and uses the UNLOAD command to move this data from Amazon Redshift to an Amazon Simple Storage Service (Amazon S3) bucket in the source Region. The data is then migrated to an S3 bucket in the AWS Region in China. The COPY command is used to load data from the S3 bucket and transfer it to the target Amazon Redshift cluster.

Amazon Redshift doesn't currently support cross-Region features such as snapshot copying to AWS Regions in China. This pattern provides a way to work around that limitation. You can also reverse the steps in this pattern to migrate data from an AWS Region in China to another AWS Region.

**Prerequisites and limitations**

*Prerequisites*

- Active AWS accounts in both a China Region and an AWS Region outside China
- Existing Amazon Redshift clusters in both a China Region and an AWS Region outside China

*Limitations*

- This is an offline migration, which means the source Amazon Redshift cluster cannot perform write operations during the migration.

**Architecture**

*Source technology stack*

- Amazon Redshift cluster in an AWS Region outside China

*Target technology stack*

- Amazon Redshift cluster in an AWS Region in China
Tools

Tools

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance. You can use Amazon S3 to store data from Amazon Redshift, and you can copy data from an S3 bucket to Amazon Redshift.
- **Amazon Redshift** – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud.
- **psql** – psql is a terminal-based front-end to PostgreSQL.

Epics

Prepare for migration in the source Region

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch and configure an EC2 instance in the source Region.</td>
<td>Sign in to the AWS Management Console and open the Amazon Elastic Compute Cloud (Amazon EC2) console. Your current Region is displayed in the navigation bar at the top of the screen. This Region cannot be an AWS Region in China. From the Amazon EC2 console dashboard, choose “Launch instance,” and</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>create and configure an EC2 instance.</td>
<td>Important: Ensure your EC2 security groups for inbound rules allow unrestricted access to TCP port 22 from your source machine. For instructions on how to launch and configure an EC2 instance, see the “Related resources” section.</td>
<td></td>
</tr>
<tr>
<td>Install the psql tool.</td>
<td>Download and install PostgreSQL. Amazon Redshift does not provide the psql tool, it is installed with PostgreSQL. For more information about using psql and installing PostgreSQL tools, see the “Related resources” section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Record the Amazon Redshift cluster details.</td>
<td>Open the Amazon Redshift console, and choose “Clusters” in the navigation pane. Then choose the Amazon Redshift cluster name from the list. On the “Properties” tab, in the “Database configurations” section, record the “Database name” and “Port.” Open the “Connection details” section and record the “Endpoint,” which is in the “endpoint:&lt;port&gt;/&lt;databasename&gt;” format. Important: Ensure your Amazon Redshift security groups for inbound rules allow unrestricted access to TCP port 5439 from your EC2 instance.</td>
<td>DBA</td>
</tr>
<tr>
<td>Connect psql to the Amazon Redshift cluster.</td>
<td>At a command prompt, specify the connection information by running the “psql -h &lt;endpoint&gt; -U &lt;userid&gt; -d &lt;databasename&gt; -p &lt;port&gt;” command. At the psql password prompt, enter the password for the “&lt;userid&gt;” user. You are then connected to the Amazon Redshift cluster and can interactively enter commands.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create an S3 bucket.</td>
<td>Open the Amazon S3 console, and create an S3 bucket to hold the files exported from Amazon Redshift. For instructions on how to create an S3 bucket, see the “Related resources” section.</td>
<td>DBA, AWS General</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Create an IAM policy that supports unloading data.
Open the AWS Identity and Access Management (IAM) console and choose “Policies.” Choose “Create policy,” and choose the “JSON” tab. Copy and paste the IAM policy for unloading data from the “Additional information” section. Important: Replace “s3_bucket_name” with your S3 bucket’s name. Choose “Review policy,” and enter a name and description for the policy. Choose “Create policy.”

### Create an IAM role to allow UNLOAD operation for Amazon Redshift.
Open the IAM console and choose “Roles.” Choose “Create role,” and choose “AWS service” in “Select type of trusted entity.” Choose “Redshift” for the service, choose “Redshift – Customizable,” and then choose “Next.” Choose the “Unload” policy you created earlier, and choose “Next.” Enter a “Role name,” and choose “Create role.”

### Associate IAM role with the Amazon Redshift cluster.
Open the Amazon Redshift console, and choose “Manage IAM roles.” Choose “Available roles” from the dropdown menu and choose the role you created earlier. Choose “Apply changes.” When the “Status” for the IAM role on the “Manage IAM roles” shows as “In-sync”, you can run the UNLOAD command.

### Stop write operations to the Amazon Redshift cluster.
You must remember to stop all write operations to the source Amazon Redshift cluster until the migration is complete.

## Prepare for migration in the target Region

### Launch and configure an EC2 instance in the target Region.
Sign in to the AWS Management Console for a Region in China, either Beijing or Ningxia. From the Amazon EC2 console, choose “Launch instance,” and create and configure an EC2 instance. Important: Make sure your
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2 security groups for inbound rules allow unrestricted access to TCP port 22 from your source machine. For further instructions on how to launch and configure an EC2 instance, see the &quot;Related resources&quot; section.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Record the Amazon Redshift cluster details.</td>
<td>Open the Amazon Redshift console, and choose “Clusters” in the navigation pane. Then choose the Amazon Redshift cluster name from the list. On the “Properties” tab, in the “Database configurations” section, record the “Database name” and “Port.” Open the “Connection details” section and record the “Endpoint,” which is in the “endpoint:&lt;port&gt;/&lt;databasename&gt;” format. Important: Make sure your Amazon Redshift security groups for inbound rules allow unrestricted access to TCP port 5439 from your EC2 instance.</td>
<td>DBA</td>
</tr>
<tr>
<td>Connect psql to the Amazon Redshift cluster.</td>
<td>At a command prompt, specify the connection information by running the “psql -h &lt;endpoint&gt; -U &lt;userid&gt; -d &lt;databasename&gt; -p &lt;port&gt;” command. At the psql password prompt, enter the password for the “&lt;userid&gt;” user. You are then connected to the Amazon Redshift cluster and can interactively enter commands.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create an S3 bucket.</td>
<td>Open the Amazon S3 console, and create an S3 bucket to hold the exported files from Amazon Redshift. For help with this and other stories, see the “Related resources” section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Create an IAM policy that supports copying data.</td>
<td>Open the IAM console and choose “Policies.” Choose “Create policy,” and choose the “JSON” tab. Copy and paste the IAM policy for copying data from the “Additional information” section. Important: Replace “s3_bucket_name” with your S3 bucket’s name. Choose “Review policy,” enter a name and description for the policy. Choose “Create policy.”</td>
<td>DBA</td>
</tr>
<tr>
<td>Create an IAM role to allow COPY operation for Amazon Redshift.</td>
<td>Open the IAM console and choose “Roles.” Choose “Create role,” and choose “AWS service” in “Select type of trusted entity.” Choose “Redshift” for the service, choose “Redshift – Customizable,” and then choose “Next.” Choose the “Copy” policy you created earlier, and choose “Next.” Enter a “Role name,” and choose “Create role.”</td>
<td>DBA</td>
</tr>
<tr>
<td>Associate IAM role with the Amazon Redshift cluster.</td>
<td>Open the Amazon Redshift console, and choose “Manage IAM roles.” Choose “Available roles” from the dropdown menu and choose the role you created earlier. Choose “Apply changes.” When the “Status” for the IAM role on the “Manage IAM roles” shows as “In-sync”, you can run the “COPY” command.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Verify source data and object information before beginning the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the rows in the source Amazon Redshift tables.</td>
<td>Use the scripts in the “Additional information” section to verify and record the number of rows in the source Amazon Redshift tables. Remember to split the data evenly for the UNLOAD and COPY scripts. This will improve the data unloading and loading efficiency, because the data quantity covered by each script will be balanced.</td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Verify the number of database objects in the source Amazon Redshift cluster.</td>
<td>Use the scripts in the &quot;Additional information&quot; section to verify and record the number of databases, users, schemas, tables, views, and user-defined functions (UDFs) in your source Amazon Redshift cluster.</td>
<td>DBA</td>
</tr>
<tr>
<td>Verify SQL statement results before migration.</td>
<td>Some SQL statements for data validation should be sorted according to actual business and data situations. This is to verify the imported data to ensure it is consistent and displayed correctly.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Migrate data and objects to the target Region**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Amazon Redshift DDL scripts.</td>
<td>Generate Data Definition Language (DDL) scripts by using the links from the “SQL statements to query Amazon Redshift” section in the “Additional information” section. These DDL scripts should include the “create user,” “create schema,” “privileges on schema to user,” “create table/view,” “privileges on objects to user,” and “create function” queries.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create objects in the Amazon Redshift cluster for the target Region.</td>
<td>Run the DDL scripts by using the AWS Command Line Interface (AWS CLI) in the AWS Region in China. These scripts will create objects in the Amazon Redshift cluster for the target Region.</td>
<td>DBA</td>
</tr>
<tr>
<td>Unload source Amazon Redshift cluster data to the S3 bucket.</td>
<td>Run the UNLOAD command to unload data from the Amazon Redshift cluster in the source Region to the S3 bucket.</td>
<td>DBA, Developer</td>
</tr>
<tr>
<td>Transfer source Region S3 bucket data to target Region S3 bucket.</td>
<td>Transfer the data from your source Region S3 bucket to the target S3 bucket. Because the &quot;$ aws s3 sync&quot; command cannot be used, make sure you use the process outlined in the “Transferring Amazon S3 data from AWS Regions to AWS”</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### Task | Description | Skills required
--- | --- | ---
Load data into the target Amazon Redshift cluster. | In the psql tool for your target Region, run the COPY command to load data from the S3 bucket to the target Amazon Redshift cluster. | DBA

**Verify the data in the source and target Regions after the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify and compare the number of rows in the source and target tables.</td>
<td>Verify and compare the number of table rows in the source and target Regions to ensure all are migrated.</td>
<td>DBA</td>
</tr>
<tr>
<td>Verify and compare the number of source and target database objects.</td>
<td>Verify and compare all database objects in the source and target Regions to ensure all are migrated.</td>
<td>DBA</td>
</tr>
<tr>
<td>Verify and compare SQL script results in the source and target Regions.</td>
<td>Run the SQL scripts prepared before the migration. Verify and compare the data to ensure that the SQL results are correct.</td>
<td>DBA</td>
</tr>
<tr>
<td>Reset the passwords of all users in the target Amazon Redshift cluster.</td>
<td>After the migration is complete and all data is verified, you should reset all user passwords for the Amazon Redshift cluster in the AWS Region in China.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Related resources
- Transferring Amazon S3 data from AWS Regions to AWS Regions in China
- Creating an S3 bucket
- Launching an EC2 instance
- Resetting an Amazon Redshift user password
- psql documentation

### Additional information

*IAM policy for unloading data*

```json
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
```
"Resource": ["arn:aws:s3:::s3_bucket_name"],
},
{
"Effect": "Allow",
"Action": ["s3:GetObject", "s3:DeleteObject"],
"Resource": ["arn:aws:s3:::s3_bucket_name/*"]
}
}

**IAM policy for copying data**

{  
"Version": "2012-10-17",
"Statement": [
  {
"Effect": "Allow",
"Action": ["s3:ListBucket"],
"Resource": ["arn:aws:s3:::s3_bucket_name"]
  },
  {
"Effect": "Allow",
"Action": ["s3:GetObject"],
"Resource": ["arn:aws:s3:::s3_bucket_name/*"]
  }
]
}

**SQL statements to query Amazon Redshift**

###Database
select * from pg_database where datdba>1;

###User
select * from pg_user where usesysid>1;

###Schema
SELECT n.nspname AS "Name",
    pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
FROM pg_catalog.pg_namespace n
WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
ORDER BY 1;

###Table
select count(*) from pg_tables where schemaname not in ('pg_catalog','information_schema');

select schemaname,count(*) from pg_tables where schemaname not in ('pg_catalog','information_schema') group by schemaname order by 1;

###View
SELECT
    n.nspname AS schemaname,c.relname AS viewname,pg_catalog.pg_get_userbyid(c.relowner) as "Owner"
Transport PostgreSQL databases between Amazon RDS DB instances using pg_transport

Created by Raunak Rishabh (AWS)

### Table

<table>
<thead>
<tr>
<th>Environment</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoC or pilot</td>
<td>Databases: Relational</td>
<td>Amazon RDS for PostgreSQL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R Type</th>
<th>Workload</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relocate</td>
<td>Open-source</td>
<td>Migration; Databases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AWS services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon RDS</td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes the steps for migrating extremely large databases between two Amazon Relational Database Service (Amazon RDS) for PostgreSQL DB instances by using the **pg_transport** command.
extension. This extension provides a physical transport mechanism to move each database. By streaming the database files with minimal processing, it provides an extremely fast method for migrating large databases between DB instances with minimal downtime. This extension uses a pull model where the target DB instance imports the database from the source DB instance.

**Prerequisites and limitations**

**Prerequisites**

- Both DB instances must run the same major version of PostgreSQL.
- The database must not exist on the target. Otherwise, the transport fails.
- No extension other than `pg_transport` must be enabled in the source database.
- All source database objects must be in the default `pg_default` tablespace.
- The security group of the source DB instance should allow traffic from the target DB instance.
- Install a PostgreSQL client like psql or PgAdmin to work with the Amazon RDS PostgreSQL DB instance. You can install the client either in your local system or use an Amazon Elastic Compute Cloud (Amazon EC2) instance. In this pattern, we use psql on an EC2 instance.

**Limitations**

- You can't transport databases between different major versions of RDS PostgreSQL.
- The access privileges and ownership from the source database are not transferred to the target database.
- You can't transport databases on read replicas or on parent instances of read replicas.
- You can't use `reg` data types in any database tables that you plan to transport with this method.
- You can run up to 32 total transports (including both imports and exports) at the same time on a DB instance.
- You do not have an option to rename, include/exclude tables on the go. Everything is migrated as it is.

**Caution**

- Make backups before removing the extension, because removing the extension also removes dependent objects and some data that's critical to the operation of the database.
- Consider the instance class and processes running on other database on source instance while deciding on the number of workers and `work_mem` values for `pg_transport`.

*Note: While the transport is running on one database, there is no effect of the same on the other databases.*

**Product versions**

- Amazon RDS for PostgreSQL 10.10 and later, and 11.5 and later. For the latest version information, see [Transporting PostgreSQL Databases Between DB Instances](#) in the Amazon RDS documentation.

**Tools**

- `pg_transport` extension - This extension provides a physical transport mechanism to move each database. By streaming the database files with minimal processing, physical transport moves data
much faster than traditional dump and load processes and takes minimal downtime. PostgreSQL transportable databases use a pull model where the destination DB instance imports the database from the source DB instance. You install this extension on your DB instances when you prepare the source and target environments, as explained in this pattern.

- **psql** or **pgAdmin**- These utilities enable you to connect to, and work with, your PostgreSQL DB instances.

## Epics

**Create the target parameter group**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a parameter group for the target system.</td>
<td>Specify a group name that identifies it as a target parameter group; for example, pgtarget-param-group. For instructions, see the link provided in the References section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Modify the parameters listed in the following steps.</td>
<td>For details on modifying parameters, see the link provided in the References section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Add &quot;pg_transport&quot; to the &quot;shared_preload_libraries&quot; parameter.</td>
<td>shared_preload_libraries = pg_stat_statements, pg_transport</td>
<td>DBA</td>
</tr>
<tr>
<td>Set the &quot;pg_transport.num_workers&quot; parameter.</td>
<td>Choose the number of workers you want to run the transport with. Value set here will determine the number of 'transport.send_file' workers that will be created in the source.</td>
<td>DBA</td>
</tr>
<tr>
<td>Increase the &quot;max_worker_processes&quot; value.</td>
<td>Increase the the value of &quot;max_worker_processes&quot; to more than three times the value of &quot;pg_transport.num_workers&quot;. If this fails, then pg_transport recommends you a minimum value.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set &quot;pg_transport.timing&quot; to 1.</td>
<td>This setting enables the reporting of timing information during the transport.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set the &quot;pg_transport.work_mem&quot; parameter.</td>
<td>This parameter specifies the maximum memory to allocate to each worker. The default value is 128 MB.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
Create the source parameter group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a parameter group for the source system.</td>
<td>For instructions, see the link provided in the References section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Provide a group name and description.</td>
<td>Specify a group name that identifies it as a source parameter group; for example, pgsource-param-group.</td>
<td>DBA</td>
</tr>
<tr>
<td>Modify the parameters listed in the following steps.</td>
<td>For details on modifying parameters, see the link provided in the References section.</td>
<td>DBA</td>
</tr>
<tr>
<td>Add &quot;pg_transport&quot; to the &quot;shared_preload_libraries&quot; parameter.</td>
<td>shared_preload_libraries = pg_stat_statements, pg_transport</td>
<td>DBA</td>
</tr>
<tr>
<td>Set the &quot;pg_transport.num_workers&quot; parameter.</td>
<td>Value of this parameter defined in the target determines the number of 'transport.send_file' workers to be used here. If you have an import running on this instance, then increase this value keeping in mind the number of workers already running.</td>
<td>DBA</td>
</tr>
<tr>
<td>Increase the &quot;max_worker_processes&quot; value.</td>
<td>Increase the the value of &quot;max_worker_processes&quot; to more than three times the value of &quot;pg_transport.num_workers&quot; on the target instance. If this fails, then pg_transport recommends you a minimum value</td>
<td>DBA</td>
</tr>
<tr>
<td>Set the &quot;pg_transport.work_mem&quot; parameter.</td>
<td>This parameter specifies the maximum memory to allocate to each worker. The default value is 128 MB.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Prepare the target environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new Amazon RDS for PostgreSQL DB instance to transport your source database to.</td>
<td>Determine the instance class and PostgreSQL version based on your business requirements.</td>
<td>DBA/SysAdmin/DBArchitect</td>
</tr>
<tr>
<td>Modify the security group of the target to allow connections on the DB instance port from the EC2 instance.</td>
<td>By default, the port for the PostgreSQL instance is 5432. If you’re using another port,</td>
<td>DBA/SysAdmin</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Transport PostgreSQL databases between Amazon RDS DB instances

### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>connections to that port must be open for the EC2 instance.</td>
<td>DBA</td>
</tr>
<tr>
<td>Modify the instance, and assign the new target parameter group.</td>
<td>For example, pgtarget-param-group.</td>
</tr>
<tr>
<td>Restart the target RDS DB instance.</td>
<td>&quot;shared_preload_libraries&quot; and &quot;max_worker_processes&quot; are static parameters and require reboot of the instance</td>
</tr>
<tr>
<td>Connect to the database from the EC2 instance using psql.</td>
<td>Command: psql -h &lt;rds_end_point&gt; -p PORT -U username -d database -W</td>
</tr>
<tr>
<td>Create the pg_transport extension.</td>
<td>Run the &quot;create extension pg_transport;&quot; command as a user with the rds_superuser role.</td>
</tr>
</tbody>
</table>

### Prepare the source environment

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify the security group of the source to allow connections on the DB instance port from the Amazon EC2 instance and target DB instance</td>
<td>By default, the port for PostgreSQL instance is 5432. If you're using another port, connections to that port must be open for the EC2 instance.</td>
</tr>
<tr>
<td>Modify the instance and assign the new source parameter group.</td>
<td>For example, pgsoure-param-group.</td>
</tr>
<tr>
<td>Restart the source RDS DB instance.</td>
<td>&quot;shared_preload_libraries&quot; and &quot;max_worker_processes&quot; are static parameters and require reboot of the instance</td>
</tr>
<tr>
<td>Connect to the database from the EC2 instance using psql.</td>
<td>Command: psql -h &lt;rds_end_point&gt; -p PORT -U username -d database -W</td>
</tr>
<tr>
<td>Create the pg_transport extension and remove all other extensions from the databases to be transported.</td>
<td>The transport will fail if there are any extensions other than pg_transport installed on the source database. This command must be run by a user with the rds_superuser role.</td>
</tr>
</tbody>
</table>
Perform the transport

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use &quot;pg_transport.import_from_server&quot; function first to perform a dry run.</td>
<td>This function is used to perform dry run. It would display any error that you would see while running the main transport. Resolve the errors and run the main transport. Command: SELECT transport.import_from_server( 'source-db-instance-endpoint', source-db-instance-port, 'source-db-instance-user', 'source-user-password', 'source-database-name', 'destination-user-password', 'true');</td>
<td>DBA</td>
</tr>
<tr>
<td>If the dry run is successful, set &quot;dry-run=false&quot; and initiate the database transport.</td>
<td>This function is used to perform the transport. It connects to the source and imports the data. SELECT transport.import_from_server( 'source-db-instance-endpoint', source-db-instance-port, 'source-db-instance-user', 'source-user-password', 'source-database-name', 'destination-user-password', false);</td>
<td>DBA</td>
</tr>
<tr>
<td>Validate the data in the target environment.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Add all the roles and permissions to the target.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Enable all required extensions in the target and source, if necessary.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Revert the value of the &quot;max_worker_processes&quot; parameter.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Related resources

References:
- Amazon RDS documentation
- pg_transport documentation
- Migrating databases using RDS PostgreSQL Transportable Databases (blog post)
- PostgreSQL downloads
- psql utility
- PgAdmin installation
• Creating a DB Parameter Group
• Modify Parameters in a DB Parameter Group

Replatform

Topics
• Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker using AWS Developer Tools (p. 1131)
• Migrate OpenText TeamSite workloads to the AWS Cloud (p. 1137)
• Migrate Windows SSL certificates to an Application Load Balancer using ACM (p. 1150)
• Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS (p. 1155)
• Migrate data from Microsoft Azure Blob storage and Backblaze B2 Cloud Storage to Amazon S3 by using Rclone (p. 1159)
• Migrate from Couchbase Server to Couchbase Cloud on AWS (p. 1164)
• Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 (p. 1182)
• Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling (p. 1188)
• Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk (p. 1193)
• Migrate a self-hosted MongoDB environment to MongoDB Atlas on the AWS Cloud (p. 1198)
• Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS (p. 1204)
• Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS (p. 1210)
• Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash (p. 1214)
• Migrate an on-premises Oracle database to Amazon RDS for Oracle (p. 1219)
• Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump (p. 1224)
• Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical (p. 1228)
• Migrate an on-premises PostgreSQL database to Amazon RDS for PostgreSQL (p. 1231)
• Migrate an on-premises PostgreSQL database to Aurora PostgreSQL (p. 1237)
• Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux (p. 1242)
• Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server (p. 1245)
• Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using Amazon S3 and SSMS (p. 1249)
• Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers (p. 1253)
• Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT (p. 1257)
• Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools (p. 1264)
• Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3 (p. 1268)
• Migrate on-premises Java applications to AWS using AWS App2Container (p. 1278)
• Populate your CMDB after integrating AWS Config with ServiceNow (p. 1285)
• Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition (p. 1291)
Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker using AWS Developer Tools

Created by Scot Marvin (AWS)

<table>
<thead>
<tr>
<th>R Type: Replatform</th>
<th>Source: Machine Learning</th>
<th>Target: Amazon SageMaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Machine learning &amp; AI; DevOps; Migration</td>
</tr>
<tr>
<td>AWS services: Amazon SageMaker</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating an on-premises machine learning (ML) application running on Unix or Linux servers to be trained and deployed on AWS using Amazon SageMaker. This deployment uses a continuous integration and continuous deployment (CI/CD) pipeline. The migration pattern is deployed using an AWS CloudFormation stack.

Prerequisites and limitations

Prerequisites

- An active AWS account using AWS Landing Zone
- AWS Command Line Interface (AWS CLI) installed and configured on your Unix or Linux server
- An ML source code repository in either GitHub, AWS CodeCommit, or Amazon Simple Storage Service (Amazon S3)

Limitations

- Only 300 individual pipelines can be deployed in one AWS Region.
- This pattern is intended for supervised ML workloads with train-and-deploy code in Python.

Product versions

- Docker version 19.03.5, build 633a0ea, using Python 3.6x

Architecture

Source technology stack

- On-premises Linux compute instance with data on either the local file system or in a relational database

Source architecture
Target technology stack

- AWS CodePipeline deployed with Amazon S3 for data storage and Amazon DynamoDB as metadata store for tracking or logging pipeline runs

Target architecture
Application migration architecture

- Native Python package and AWS CodeCommit repository (and an SQL client, for on-premises datasets on database instance)
Tools

- Python
- Git
- AWS CLI – The AWS CLI deploys the AWS CloudFormation stack and moves data to the S3 bucket. The S3 bucket, in turn, leads to the target.

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate source code and datasets.</td>
<td></td>
<td>Data scientist</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify target build, train, and deployment instance types and sizes.</td>
<td></td>
<td>Data engineer, Data scientist</td>
</tr>
<tr>
<td>Create capability list and capacity requirements.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify network requirements.</td>
<td></td>
<td>DBA, Systems administrator</td>
</tr>
<tr>
<td>Identify the network or host access security requirements for the source and target applications.</td>
<td></td>
<td>Data engineer, ML engineer, Systems administrator</td>
</tr>
<tr>
<td>Determine backup strategy.</td>
<td></td>
<td>ML engineer, Systems administrator</td>
</tr>
<tr>
<td>Determine availability requirements.</td>
<td></td>
<td>ML engineer, Systems administrator</td>
</tr>
<tr>
<td>Identify the application migration or switchover strategy.</td>
<td></td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>

**Configure the infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td>ML engineer, Systems administrator</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>ML engineer, Systems administrator</td>
</tr>
<tr>
<td>Set up an Amazon S3 bucket and AWS CodeCommit repository branches for ML code.</td>
<td></td>
<td>ML engineer</td>
</tr>
</tbody>
</table>

**Upload the data and code**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native MySQL tools or third-party tools to migrate train, validate, and test datasets to provisioned S3 bucket.</td>
<td>This is required for AWS CloudFormation stack deployment.</td>
<td>Data engineer, ML engineer</td>
</tr>
<tr>
<td>Package the ML train and hosting code as Python packages and push to the provisioned repository in AWS CodeCommit or GitHub.</td>
<td>You need the repository’s branch name to deploy the AWS CloudFormation template for migration.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>
Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the ML workload migration strategy.</td>
<td></td>
<td>Application owner, ML engineer</td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation stack.</td>
<td>Use the AWS CLI to create the stack declared in the YAML template provided with this solution.</td>
<td>Data scientist, ML engineer</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients over to the new infrastructure.</td>
<td></td>
<td>Application owner, Data scientist, ML engineer</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td>Shut down any custom resources from the AWS CloudFormation template (for example, any AWS Lambda functions that aren't being used).</td>
<td>Data scientist, ML engineer</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>Application owner, Data scientist</td>
</tr>
<tr>
<td>Validate the results and the ML model evaluation metrics with operators.</td>
<td>Make sure that model performance matches the application users' expectations and is comparable to the on-premises state.</td>
<td>Application owner, Data scientist</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>Application owner, ML engineer</td>
</tr>
</tbody>
</table>

Related resources

- AWS CodePipeline
- AWS CodeBuild
- Amazon SageMaker
- Amazon S3
- Amazon DynamoDB
- AWS Lambda
**Attachments**

attachment.zip

**Migrate OpenText TeamSite workloads to the AWS Cloud**

*Created by Battulga Purevragchaa (AWS), Michael Stewart (TBSCG), and Carlos Marruenda Molina (TBSCG)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Source: On premises</th>
<th>Target: AWS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R Type:</strong> Replatform</td>
<td><strong>Workload:</strong> All other workloads</td>
<td><strong>Technologies:</strong> Migration; Websites &amp; web apps</td>
</tr>
<tr>
<td><strong>AWS services:</strong> Amazon EC2; Amazon RDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

Many OpenText Experience Platform instances are hosted on premises or on traditional hosting solutions with fixed capacity and legacy cost models. Migrating your OpenText Experience Platform workloads to the Amazon Web Services (AWS) Cloud provides additional capabilities and value by increasing your business agility and integration opportunities, in addition to reducing your overall ownership cost.

This pattern provides steps and a template to migrate OpenText TeamSite workloads to the AWS Cloud. The pattern helps you understand how to scope and budget your migration projects by providing a detailed *Epics* section that guides you through an OpenText TeamSite migration process.

This pattern was developed by AWS and TBSCG, an AWS Partner, and accompanies the guide *Migrating OpenText TeamSite and Media Management workloads to the AWS Cloud* on the AWS Prescriptive Guidance website.

**Prerequisites and limitations**

**Prerequisites**

- At least one active AWS account
- An OpenText workload hosted in an on-premises data center or on another cloud provider
- Active OpenText licenses

The migration process also requires the roles and responsibilities that are described in the following table.

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sponsor</td>
<td>Internal sponsorship</td>
</tr>
<tr>
<td>Delivery manager</td>
<td>Migration delivery</td>
</tr>
<tr>
<td>Solutions architect</td>
<td>Define the current and new architecture</td>
</tr>
<tr>
<td>DevOps engineer</td>
<td>DevOps activities</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate OpenText TeamSite workloads to AWS**

<table>
<thead>
<tr>
<th>Role</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA tester</td>
<td>System-level testing</td>
</tr>
<tr>
<td>Product owner</td>
<td>Task prioritization based on business requirements</td>
</tr>
<tr>
<td>TeamSite authors</td>
<td>Migration user acceptance testing (UAT)</td>
</tr>
<tr>
<td>TeamSite administrator</td>
<td>Migration UAT</td>
</tr>
<tr>
<td>OpenText lead</td>
<td>OpenText product specialist</td>
</tr>
<tr>
<td>OpenText developer</td>
<td>OpenText product specialist</td>
</tr>
<tr>
<td>Pricing specialist</td>
<td>AWS and OpenText licensing</td>
</tr>
<tr>
<td>IT security</td>
<td>IT security baseline</td>
</tr>
<tr>
<td>Third-party integration developer</td>
<td>Rework existing integrations</td>
</tr>
<tr>
<td>Front-end developer</td>
<td>Make changes to migrated front-end code</td>
</tr>
<tr>
<td>Database administrator</td>
<td>Database configuration</td>
</tr>
</tbody>
</table>

**Limitations**

- Ensure compatibility with your target operating systems (OSs). You can use the compatibility matrix from the product release notes of the OpenText product version that you are migrating.

**Architecture**

**Source technology stack**

- OpenText customer experience solutions hosted on premises or on another cloud provider:
  - OpenText TeamSite
  - OpenText LiveSite
  - OpenText Media Management
  - OpenText MediaBin

**Target technology stack**

- An OpenText Customer Experience platform hosted on the AWS Cloud and that uses the following AWS services:
  - Amazon Elastic Compute Cloud (Amazon EC2)
  - Amazon Elastic Container Service (Amazon ECS)
  - Amazon Elasticsearch Service (Amazon ES)
  - Elastic Load Balancing
  - AWS Lambda
  - Amazon API Gateway
  - Amazon Relational Database Service (Amazon RDS)
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon Simple Storage Service (Amazon S3)

**Target architecture**
**Tools**

- **AWS Database Migration Service (AWS DMS)** – AWS DMS is a cloud service that makes it easy to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores.
- **AWS Server Migration Service (AWS SMS)** – AWS SMS combines data collection tools with automated server replication to speed up the migration of on-premises servers to AWS.

**Epics**

**Discovery and assessment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hold workshops on discovery</td>
<td>Hold workshops with business and technical teams to discover the current</td>
<td>Sponsor (optional), Delivery</td>
</tr>
<tr>
<td>requirements.</td>
<td>landscape, gather requirements, and validate the migration strategy.</td>
<td>manager, Solutions architect,</td>
</tr>
<tr>
<td></td>
<td>Depending on your migration’s complexity and scope, your organization might</td>
<td>OpenText lead, Product owner</td>
</tr>
<tr>
<td></td>
<td>require several workshops.</td>
<td></td>
</tr>
<tr>
<td><strong>Duration</strong>: Two weeks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze solution and migration</td>
<td>Analyze and document the business, functional, and technical requirements</td>
<td>Solutions architect, OpenText</td>
</tr>
<tr>
<td>requirements.</td>
<td>that influence the design of the planned solution and migration process.</td>
<td>lead, Product owner</td>
</tr>
<tr>
<td><strong>Duration</strong>: One week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Document your existing OpenText architecture.</td>
<td>Document your existing OpenText architecture, including core components and all related applications and services.</td>
<td>Solutions architect, OpenText lead, Product owner</td>
</tr>
<tr>
<td><strong>Duration</strong>: One week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define the planned AWS architecture.</td>
<td>Define your planned AWS architecture based on the identified components, requirements, and using the OpenText compatibility matrix. You can find the OpenText compatibility matrix in the release notes of your OpenText TeamSite version.</td>
<td>Solutions architect, OpenText lead, Product owner, IT security</td>
</tr>
<tr>
<td><strong>Duration</strong>: One week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assess the size of your planned AWS architecture.</td>
<td>Size requirements vary for different architectural components depending on the workload and other non-functional requirements.</td>
<td>Solutions architect, OpenText lead</td>
</tr>
<tr>
<td><strong>Duration</strong>: Two days</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calculate the TCO.</td>
<td>Calculate the total cost of ownership (TCO) for your proposed solution.</td>
<td>Solutions architect, Pricing specialist</td>
</tr>
<tr>
<td><strong>Duration</strong>: Two days</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define the migration strategy for each component.</td>
<td>Define and document which of the seven common migration strategies (7 Rs) to use for each core or additional component that must be migrated to the AWS Cloud.</td>
<td>Solutions architect, OpenText lead, Product owner</td>
</tr>
<tr>
<td><strong>Duration</strong>: One week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define the migration process for the components.</td>
<td>Define the detailed migration process for each of your workload's components.</td>
<td>Solutions architect, OpenText lead, Product owner, IT security</td>
</tr>
<tr>
<td><strong>Duration</strong>: One week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define the global migration process and dependencies.</td>
<td>Create a global migration process and calendar that includes the migration details for components, dependencies, and business continuity.</td>
<td>Solutions architect, OpenText lead, Product owner, IT security</td>
</tr>
<tr>
<td><strong>Duration</strong>: Three days</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Security and compliance activities

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create security policies.</td>
<td>Configure the customer managed security policies in your AWS accounts. These should include password complexity and rotation, in addition to automatically turning off unused accounts. For more information about customer managed policies, see <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/index.html">Customer managed policies</a> in the AWS Identity and Access Management (IAM) documentation.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Create IAM users.</td>
<td>Create the IAM users that require access to the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK. For more information about creating IAM users, see <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id-users-create.html">Creating an IAM user in your AWS account</a> in the IAM documentation.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Create IAM groups.</td>
<td>Create the required IAM user groups (for example, administrator or developer groups) and add IAM users to those groups. For more information about IAM user groups, see <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id-users-groups.html">IAM user groups</a> in the IAM documentation.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Attach security policies.</td>
<td>Attach security policies to the IAM groups or roles. For more information about this, see <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id/users-groups-administrativeactions.html">Attaching a policy to an IAM user group</a> in the IAM documentation.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Turn on detailed billing.</td>
<td>For more information about billing, see <a href="https://docs.aws.amazon.com/billing/latest/utilguide/monitoring-billing.html">Monitoring your usage and costs</a> in the AWS Billing and Cost Management documentation.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Check the contact details for your accounts.</td>
<td>Make sure that the contact details for your accounts are up to date and map to more</td>
<td>Solutions architect, Product owner</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>than one individual in your organization.</td>
<td>For more information, see Managing an AWS account in the AWS Billing and Cost Management documentation.</td>
<td></td>
</tr>
<tr>
<td>Add security contact information.</td>
<td>Configure your contact information with your security contact information.</td>
<td>Solutions architect, IT security</td>
</tr>
<tr>
<td>Set up IAM roles for EC2 instances.</td>
<td>Configure the IAM roles for the EC2 instances.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Configure access to AWS Support.</td>
<td>Attach an IAM policy to IAM users that require access to AWS Support for Support Center and to create support cases.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Enable CloudTrail.</td>
<td>Automatically enable AWS CloudTrail in all your AWS Regions.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Enable CloudTrail log file validation.</td>
<td>Enable the validation of CloudTrail log files.</td>
<td>Solutions architect</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate OpenText TeamSite workloads to AWS**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Restrict access to any S3 buckets that contain CloudTrail logs.** | Apply a bucket policy restricting access to S3 buckets that contain CloudTrail log files.  
For more information about this, see [Amazon S3 bucket policy for CloudTrail](https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-viewer.html) in the AWS CloudTrail documentation. | Solutions architect |
| **Integrate CloudTrail with CloudWatch Logs** | Integrate trails generated by CloudTrail with Amazon CloudWatch Logs.  
For more information about this, see [Sending events to CloudWatch Logs](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Send-CloudTrail-Logs-to-CloudWatch-Logs.html) in the AWS CloudTrail documentation. | Solutions architect |
| **Enable AWS Config in all required Regions.** | Automatically enable AWS Config in all required Regions.  
You can set up AWS Config by using AWS CLI. For more information, see [Setting Up AWS Config with the AWS CLI](https://docs.aws.amazon.com/config/latest/usage/config-cli.html) in the AWS Config documentation. | Solutions architect |
| **Enable logging of S3 bucket access.** | Automate S3 bucket access logging with CloudTrail.  
For more information about this, see [Enabling CloudTrail event logging for S3 buckets and objects](https://docs.aws.amazon.com/AmazonS3/latest/dev/index.html) in the Amazon S3 documentation. | Solutions architect |
| **Configure AWS KMS key policies for CloudTrail.** | Automate the configuration of AWS Key Management Service (AWS KMS) key policies for CloudTrail.  
For more information about this, see [Configure AWS KMS key policies for CloudTrail](https://docs.aws.amazon.com/cloudtrail/latest/userguide/cloudtrail-config-kms.html) in the AWS CloudTrail documentation. | Solutions architect |
| **Encrypt CloudTrail logs at rest.** | Configure server-side encryption of CloudTrail logs using customer managed keys held in AWS KMS.  
For more information about this, see [Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS)](https://docs.aws.amazon.com/cloudtrail/latest/userguide/cloudtrail-configure-kms.html) in the AWS CloudTrail documentation. | Solutions architect |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatically rotate KMS keys.</td>
<td>Configure the rotation of AWS KMS keys.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see How to enable and disable automatic key rotation in the AWS KMS documentation.</td>
<td></td>
</tr>
<tr>
<td>Configure CloudWatch alarms.</td>
<td>Configure the Amazon CloudWatch alarms that are initiated by specific events. For example, unauthorized requests to APIs or use of the root account.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see How to receive notifications when your AWS account’s root access keys are used from the AWS Security Blog.</td>
<td></td>
</tr>
<tr>
<td>Configure security groups.</td>
<td>Configure security groups to ensure that unrestricted inbound traffic is not allowed on ports 22 and 3389.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td>Turn on VPC flow logging.</td>
<td>Capture rejected IP traffic to and from network interfaces in your virtual private cloud (VPC) and configure CloudWatch to capture it.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see Creating a flow log in the Amazon VPC documentation.</td>
<td></td>
</tr>
<tr>
<td>Modify the default security group to restrict all traffic.</td>
<td>Modify each VPC’s default security group so that traffic is denied by default and access is explicitly granted through your security groups.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see Security groups for your VPC in the Amazon VPC documentation.</td>
<td></td>
</tr>
<tr>
<td>Configure routing tables between the VPCs.</td>
<td>Configure the routing tables for VPC peering with the least access necessary.</td>
<td>Solutions architect</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see Updating your route tables for a VPC peering connection in the Amazon VPC documentation.</td>
<td></td>
</tr>
</tbody>
</table>
## Setup activities for the new AWS infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provision the AWS infrastructure.</td>
<td>Create the AWS accounts and resources.</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td></td>
<td><strong>Duration</strong>: Two weeks</td>
<td></td>
</tr>
<tr>
<td>Set up DevOps tools and processes.</td>
<td>Set up DevOps tools and procedures, such as continuous integration and</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td></td>
<td>continuous delivery (CI/CD) pipelines and automated testing frameworks.</td>
<td></td>
</tr>
<tr>
<td>Automate the migration of core components.</td>
<td>Use existing templates or scripts to automate the installation and</td>
<td>DevOps engineer, Solutions architect, OpenText lead</td>
</tr>
<tr>
<td></td>
<td>configuration of OpenText products including TeamSite, LiveSite, OpenDeploy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and MediaBin.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Duration</strong>: One week</td>
<td></td>
</tr>
<tr>
<td>Automate the migration of additional</td>
<td>Analyze and automate the migration of additional</td>
<td>DevOps engineer, Solutions architect, OpenText lead</td>
</tr>
<tr>
<td>components.</td>
<td>applications that are integrated with OpenText core components (for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>example, additional databases, communication, monitoring, or cache</td>
<td></td>
</tr>
<tr>
<td></td>
<td>components).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Duration</strong>: Two weeks</td>
<td></td>
</tr>
<tr>
<td>Adapt core components.</td>
<td>Make any required changes to customizations of OpenText core components</td>
<td>Solutions architect, OpenText lead, OpenText developer,</td>
</tr>
<tr>
<td></td>
<td>(for example, integrations).</td>
<td>Third-party integration developer, Front-end developer</td>
</tr>
<tr>
<td>Implement and configure additional</td>
<td>Provision, configure, and implement any new AWS services, such as AWS</td>
<td>DevOps engineer, Solutions architect, Third-party</td>
</tr>
<tr>
<td>services.</td>
<td>Lambda functions or Amazon API Gateway.</td>
<td>integration developer, Front-end developer</td>
</tr>
<tr>
<td>Migrate or refactor other components.</td>
<td>Migrate additional components, including any required refactoring. This</td>
<td>DevOps engineer, Solutions architect, Third-party</td>
</tr>
<tr>
<td></td>
<td>includes external applications such as custom-made reporting portals or</td>
<td>integration developer, Front-end developer</td>
</tr>
<tr>
<td></td>
<td>existing API integration layers.</td>
<td></td>
</tr>
<tr>
<td>Carry out migration in development</td>
<td>Automated migration activities for the development environment, including</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>environment.</td>
<td>system provisioning, data</td>
<td></td>
</tr>
</tbody>
</table>

1145
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carry out migration in production environment.</td>
<td>Automated migration activities for the production environment, including system provisioning, data migration, application migration, installation, and configuration.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Networking activities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>Define CIDR blocks for each VPC.</td>
<td>Define the Classless Inter-Domain Routing (CIDR) block (the IP range and mask) for each non-default VPC.</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td>Duration: Less than one week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define subnets and Availability Zones.</td>
<td>Define the subnets and Availability Zones that are used in each non-default VPC.</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td>Duration: Less than one week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define security groups.</td>
<td>Define security groups and security group rules for controlling security on AWS resources.</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td>Duration: Less than one week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define network ACLs.</td>
<td>Define the network access control lists (ACLs) to control security at subnet boundaries.</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td>Duration: Less than one week</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Migrate databases</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>Prepare the source databases.</td>
<td>Use AWS DMS to prepare each source database for ongoing replication to the AWS Cloud.</td>
<td>DevOps engineer, Solutions architect</td>
</tr>
<tr>
<td>Create the databases for the OpenText core components.</td>
<td>Create the databases required by the Opentext TeamSite, LiveSite, and MediaBin components. Make sure that users and access rights are correctly configured</td>
<td>Solutions architect, OpenText lead, OpenText developer</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Migrate OpenText TeamSite workloads to AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy data from source database servers.</td>
<td>Automate the process of copying data for OpenText core components from the source database server to the target database server.</td>
<td>Solutions architect, OpenText lead, OpenText developer</td>
</tr>
<tr>
<td>Synchronize data from the database servers.</td>
<td>Automate the process of performing regular data synchronization from the source databases to the target databases.</td>
<td>OpenText developer</td>
</tr>
</tbody>
</table>

### Content migration activities

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy the OpenText TeamSite content stores.</td>
<td>Automate the process of copying the content stores from the source OpenText TeamSite server to the target OpenText TeamSite server.</td>
<td>Solutions architect, OpenText lead, OpenText developer</td>
</tr>
<tr>
<td>Map users and groups.</td>
<td>Internal mapping of internal OpenText TeamSite user IDs to target system IDs.</td>
<td>OpenText lead</td>
</tr>
<tr>
<td>Synchronize the OpenText TeamSite content stores.</td>
<td>Automate the process of performing regular synchronizing of source and target content stores. This is implemented as part of the migration and QA process.</td>
<td>OpenText developer</td>
</tr>
<tr>
<td>Copy data from web servers.</td>
<td>Automate the process of copying data from the source web servers to the target web servers.</td>
<td>Solutions architect, OpenText lead, OpenText developer</td>
</tr>
<tr>
<td>Synchronize the web server data.</td>
<td>Automate the process of performing regular synchronizing of source and target web server data.</td>
<td>OpenText developer</td>
</tr>
<tr>
<td>Copy data from web server file system.</td>
<td>Automate the process of copying content and other web assets from the source web server file system to the target web servers.</td>
<td>Solutions architect, OpenText lead, OpenText developer</td>
</tr>
<tr>
<td>Synchronize the web server file systems.</td>
<td>Automate the process of performing regular synchronizing of content and other web assets from the</td>
<td>OpenText developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Generate feeds and indexes.</strong></td>
<td>Automate the process of running any processes that generate feeds or other indexes (for example, web search) that uses OpenText TeamSite or web server content as a data source.</td>
<td>Solutions architect, OpenText lead, OpenText developer</td>
</tr>
<tr>
<td><strong>Synchronize the generation of feeds and indexes.</strong></td>
<td>Automate the process of performing regular regeneration of feeds and indexes after data synchronizations.</td>
<td>OpenText developer</td>
</tr>
</tbody>
</table>

**Testing and QA activities**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Perform migration QA.</strong></td>
<td>Test the target AWS environment, applications, and services to ensure the automated migration processes are correctly built and configured.</td>
<td>DevOps engineer, OpenText lead, QA tester</td>
</tr>
<tr>
<td><strong>Carry out performance testing.</strong></td>
<td>Test the performance in terms of responsiveness and stability under a particular workload. Investigate, measure, validate, or verify other quality attributes of the destination system, such as scalability and reliability. For this test to be useful, you must have a testing environment that is the same size as your production environment. <strong>Duration:</strong> Between one and two weeks</td>
<td>DevOps engineer, OpenText lead</td>
</tr>
<tr>
<td><strong>Security testing.</strong></td>
<td>Vulnerability scanning and penetration testing to reveal potential flaws in the security mechanisms of an application that protect data and maintain functionality as required. For this test to be useful, you must have a testing environment that is equivalent to your production environment in</td>
<td>DevOps engineer, OpenText lead</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate OpenText TeamSite workloads to AWS**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>terms of networking and security.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Duration:</strong> Between one and two weeks</td>
<td></td>
</tr>
</tbody>
</table>

### Operational integration activities

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check operational readiness.</td>
<td>Understand how you currently perform IT operations and how you will operate in the AWS Cloud. You can achieve this business outcome by defining a cloud operating model.</td>
<td>DevOps engineer, OpenText lead, Service delivery manager</td>
</tr>
<tr>
<td></td>
<td><strong>Duration:</strong> One week</td>
<td></td>
</tr>
<tr>
<td>Invest in operations automation.</td>
<td>Invest in automation to deliver an AWS operating model.</td>
<td>DevOps engineer, OpenText lead, Service delivery manager</td>
</tr>
<tr>
<td>Integrate operations.</td>
<td>Continue using current IT tools and extend them through integration to the AWS Cloud.</td>
<td>DevOps engineer, OpenText lead, Service delivery manager</td>
</tr>
</tbody>
</table>

### Cutover activities

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch DNS.</td>
<td>Manually switch the domain name system (DNS) from existing hosts to hosts based in the AWS Cloud.</td>
<td>DevOps engineer, OpenText lead</td>
</tr>
<tr>
<td></td>
<td><strong>Duration:</strong> One hour</td>
<td></td>
</tr>
<tr>
<td>Test disaster recovery.</td>
<td>Test disaster recovery, backup restore, and run your automated tests.</td>
<td>DevOps engineer, OpenText lead, QA tester</td>
</tr>
<tr>
<td></td>
<td><strong>Duration:</strong> One day</td>
<td></td>
</tr>
<tr>
<td>Validate monitoring and analytics.</td>
<td>Validate that the monitoring and analytics are working.</td>
<td>DevOps engineer, OpenText lead</td>
</tr>
<tr>
<td></td>
<td><strong>Duration:</strong> Two hours</td>
<td></td>
</tr>
<tr>
<td>Turn off old environment and request the server’s shutdown.</td>
<td></td>
<td>DevOps engineer, OpenText lead</td>
</tr>
</tbody>
</table>
Related resources

- Customer managed policies
- Creating an IAM user in your AWS account
- IAM user groups
- Attaching a policy to an IAM user group
- Monitoring your usage and cost
- Managing an AWS account
- IAM roles for Amazon EC2
- Access permissions for AWS Support
- Using create-trail
- Enabling log file integrity validation for CloudTrail
- Amazon S3 bucket policy for CloudTrail
- Sending events to CloudWatch Logs
- Setting Up AWS Config with the AWS CLI
- Enabling CloudTrail event logging for S3 buckets and objects
- Configure AWS KMS key policies for CloudTrail
- Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS)
- How to enable and disable automatic key rotation
- How to receive notifications when your AWS account’s root access keys are used
- Creating a flow log
- Security groups for your VPC
- Updating your route tables for a VPC peering connection

Migrate Windows SSL certificates to an Application Load Balancer using ACM

Created by Chandra Sekhar Yaratha (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source: Windows web application</th>
<th>Target: Application Load Balancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: Production</td>
<td>Technologies: Management &amp; governance; Websites &amp; web apps; Migration</td>
</tr>
<tr>
<td>Workload: Microsoft</td>
<td>AWS services: Elastic Load Balancing; AWS Certificate Manager</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

The pattern provides guidance for using AWS Certificate Manager (ACM) to migrate existing Secure Sockets Layer (SSL) certificates from websites hosted on-premises on Microsoft Internet Information
Services (IIS) on Windows servers. The SSL certificates can then be used with Elastic Load Balancing on AWS. SSL protects your data, affirms your identity, provides better search engine rankings, helps to satisfy Payment Card Industry Data Security Standard (PCI DSS) requirements, and improves customer trust. Developers and IT teams that manage these workloads want their web applications and infrastructure, including the IIS server and Windows Server, to remain compliant with their baseline policies.

This pattern covers manually exporting existing SSL certificates from Windows IIS and then importing them into ACM in your AWS account, creating an Application Load Balancer for your application, and configuring the Application Load Balancer to use your imported certificates. HTTPS connections are then terminated on the Application Load Balancer, and you don't need further configuration overhead on the web server. For more details, see Create an HTTPS listener for your Application Load Balancer.

Windows servers use .pfx/.p12 files to contain the public key file (SSL certificate) and its unique private key file. The Certificate Authority (CA) provides you with your public key file. You use your server to generate the associated private key file where the certificate signing request (CSR) was created.

Prerequisites and limitations

Prerequisites

- An active AWS account. You must have a virtual private cloud (VPC) with at least one private subnet in each of the Availability Zones used by your targets. The public subnets must include Network Address Translation (NAT) gateways and Remote Desktop Gateway instances.
- A web application running on IIS.
- The Administrator role on the on-premises IIS Windows server.
- For the Windows server, full internet access to AWS.

Limitations

- There can be one HTTP protocol and one HTTPS protocol bound to the website. The protocols are bound to ports tcp/80 and tcp/443, respectively.

Product versions

- IIS version 8.0 or above running on Windows Server 2012 or above

Architecture

Source technology stack

- Any HTTP Server (IIS or Apache HTTP Server or NGINX) implementation with SSL to ensure that data is transmitted securely in an encrypted connection (HTTPS)

Source architecture
Target technology stack

- ACM certificates in your AWS account
- An Application Load Balancer configured to use your imported certificates
- Windows Server instances in the private subnets

Target architecture
AWS Prescriptive Guidance Patterns
Migrate Windows SSL certificates to an
Application Load Balancer using ACM

Tools

- **AWS Certificate Manager** – AWS Certificate Manager (ACM) is a service that you use to provision, manage, and deploy public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services. SSL/TLS certificates are used to secure network communications and establish the identity of websites over the internet as well as resources on private networks. ACM removes the time-consuming manual process of purchasing, uploading, and renewing SSL/TLS certificates.

- **Elastic Load Balancing** – Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, IP addresses, and AWS Lambda functions.

Epics

Export a .pfx file

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export the .pfx file from Windows Server.</td>
<td>To export the SSL certificate as a .pfx file from the on-premises IIS manager in Windows Server, do the following. (1) Choose &quot;Start,&quot; &quot;Administrative,&quot; &quot;Internet Information Services (IIS) Manager.&quot; (2) Select the server name, and under &quot;Security,&quot; double-click &quot;Server Certificates.&quot; (3) Choose the certificate that you want to export, and then choose &quot;Export.&quot; (4) In the &quot;Export Certificate&quot; pop-up box, choose a location, path, and name for your .pfx file. (5) Specify and confirm a password for your .pfx file. Note: You need this password when you install the .pfx file. (6) Choose &quot;OK.&quot; Your .pfx file should now be saved to the location and path you specified.</td>
<td>SysAdmin/Customer</td>
</tr>
</tbody>
</table>

Enter the password for the .pfx file. | While exporting the .pfx file, when you are prompted, enter the password that you specified for the .pfx file. | SysAdmin/Customer |

Import a certificate into ACM

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare to import the certificate.</td>
<td>On the ACM console, choose &quot;Import a certificate.&quot;</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Migrate Windows SSL certificates to an Application Load Balancer using ACM

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the certificate body.</td>
<td>For &quot;Certificate body,&quot; paste the PEM-encoded certificate that you want to import. For more information about the commands and steps described in this and other stories, see the links in the &quot;Related resources&quot; section at the end of this pattern.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Provide the certificate private key.</td>
<td>For &quot;Certificate private key,&quot; paste the PEM-encoded, unencrypted private key that matches the certificate's public key.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Provide the certificate chain.</td>
<td>For &quot;Certificate chain,&quot; paste the PEM-encoded certificate chain. The PEM-encoded certificate chain is stored in a file named &quot;CertificateChain.pem.&quot;</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Import the certificate.</td>
<td>Choose &quot;Review and import.&quot; Review the information about your certificate, then choose &quot;Import.&quot;</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Create an Application Load Balancer

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the load balancer and listeners.</td>
<td>Create a load balancer and a listener. Add a second listener (HTTPS) for port 443.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Configure security settings for the HTTPS listener.</td>
<td>Configure a certificate and security policy.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Configure a security group and a target group.</td>
<td>For &quot;Target type,&quot; choose &quot;Instance.&quot;</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Configure targets for the target group.</td>
<td>Choose &quot;Next: Register targets.&quot; Register the targets by instance ID.</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>Create the load balancer.</td>
<td>Choose &quot;Next: Review.&quot; On the &quot;Review&quot; page, choose &quot;Create.&quot; After the load balancer is created, choose &quot;Close.&quot;</td>
<td>Cloud Administrator</td>
</tr>
</tbody>
</table>

Related resources

Import a certificate into ACM
Summary

This pattern describes how to migrate a .NET Framework or .NET Core web or console application from using the Microsoft Azure Service Bus queue messaging platform to Amazon Simple Queue Service (Amazon SQS).

Applications use messaging services to send data to, and receive data from, other applications. These services help build decoupled, highly scalable microservices, distributed systems, and serverless applications in the cloud.

Azure Service Bus queues are part of a broader Azure messaging infrastructure that supports queuing and publish/subscribe messaging.

Amazon SQS is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS eliminates the complexity and overhead associated with managing and operating message-oriented middleware, and enables developers to focus on differentiating work. Using Amazon SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A .NET Framework or .NET Core web or console application that uses Azure Service Bus queues (sample code attached)
Product versions

• .NET Framework 3.5 or later, or .NET Core 1.0.1, 2.0.0, or later

Architecture

Source technology stack

• A .NET (Core or Framework) web or console application that uses an Azure Service Bus queue to send messages

Target technology stack

• Amazon SQS

Tools

Tools

• Microsoft Visual Studio

Code

To create an AWS Identity and Access management (IAM) policy for Amazon SQS:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose Policies, and then choose Create policy.

3. Choose the JSON tab, and paste the following code:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl",
                "sqs:ChangeMessageVisibility",
                "sqs:SendMessageBatch",
                "sqs:ReceiveMessage",
                "sqs:SendMessage",
                "sqs:GetQueueAttributes",
                "sqs:ListQueueTags",
                "sqs:ListDeadLetterSourceQueues",
                "sqs:DeleteMessageBatch",
                "sqs:PurgeQueue",
                "sqs:DeleteQueue",
                "sqs:CreateQueue",
                "sqs:ChangeMessageVisibilityBatch",
                "sqs:SetQueueAttributes"
            ],
            "Resource": "arn:aws:sqs:*:<AccountId>:*
        }
    ]
}
```
4. Choose **Review policy**, type a name, and then choose **Create policy**.

5. Attach the newly created policy to your existing IAM role or create a new role.

## Epics

### Set up Amazon SQS in AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM policy for Amazon SQS.</td>
<td>Create the IAM policy that will provide access to Amazon SQS. See the Code section for a sample policy.</td>
<td>System engineer</td>
</tr>
<tr>
<td>Create an AWS profile.</td>
<td>Create a new profile by running the AWS Tools for PowerShell command Set-AWSCredential. This command stores your access key and secret key in your default credentials file under the profile name you specify. Link the Amazon SQS policy you created earlier with this account. Keep the AWS access key ID and secret access key. These will be required in the next steps.</td>
<td>System engineer</td>
</tr>
<tr>
<td>Create an SQS queue.</td>
<td>You can create a standard queue or a first in, first out (FIFO) queue. For instructions, see the link in the References section.</td>
<td>System engineer</td>
</tr>
</tbody>
</table>

### Revise your .NET application code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install AWS Toolkit for Visual Studio.</td>
<td>This toolkit is an extension for Microsoft Visual Studio and makes it easier for you to build and deploy .NET applications in AWS. For installation and usage instructions, see the link in the References section.</td>
<td>Application developer</td>
</tr>
<tr>
<td>Install the AWSSDK.SQS NuGet package.</td>
<td>You can install AWSSDK.SQS by choosing &quot;Manage NuGet&quot;</td>
<td>Application developer</td>
</tr>
</tbody>
</table>
### Migrate a messaging queue from Microsoft Azure to Amazon SQS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an AWSCredentials object in your .NET application.</strong></td>
<td>The sample application in the attachment shows how to create a BasicAWSCredentials object, which inherits from AWSCredentials. You can use the access key ID and secret access key from earlier, or let the object pick these from the .aws folder as part of the user profile at run time.</td>
<td>Application developer</td>
</tr>
<tr>
<td><strong>Create an SQS client object.</strong></td>
<td>Create an SQS client object (AmazonSQSClient) for .NET Framework. This is part of the Amazon.SQS namespace. This object is required instead of IQueueClient, which is part of the Microsoft.Azure.ServiceBus namespace.</td>
<td>Application developer</td>
</tr>
<tr>
<td><strong>Call the SendMessageAsync method to send messages to the SQS queue.</strong></td>
<td>Change the code that sends the message to the queue to use the amazonSqsClient.SendMessageAsync method. For details, see the attached code sample.</td>
<td>Application developer</td>
</tr>
<tr>
<td><strong>Call the ReceiveMessageAsync method to receive messages from the SQS queue.</strong></td>
<td>Change the code that receives the message to use the amazonSqsClient.ReceiveMessageAsync method. For details, see the attached code sample.</td>
<td>Application developer</td>
</tr>
<tr>
<td><strong>Call the DeleteMessageAsync method to delete messages from the SQS queue.</strong></td>
<td>To delete messages, change the code from the queueClient.CompleteAsync method to the amazonSqsClient.DeleteMessageAsync method. For details, see the attached code sample.</td>
<td>Application developer</td>
</tr>
</tbody>
</table>

### Related resources
- AWS SDK for .NET Developer Guide
- Messaging Using Amazon SQS
- Creating and Using an Amazon SQS Queue with the AWS SDK for .NET
- Send an Amazon SQS Message
- Receive a Message from an Amazon SQS Queue
- Delete a Message from an Amazon SQS Queue
- AWS Toolkit for Visual Studio
Additional information

This pattern includes two sample applications (see the attachments section):

- **AzureSbTestApp** includes code that uses the Azure Service Bus queue.
- **AmazonSqsTestApp** uses Amazon SQS. This is a console application that uses .NET Core 2.2 and includes examples for sending and receiving messages.

Notes:

- queueClient is an object of IQueueClient, which is part of the Microsoft.Azure.ServiceBus namespace (included in the Microsoft.Azure.ServiceBus NuGet package).
- amazonSqsClient is an object of AmazonSQSClient, which is part of the Amazon.SQS namespace (included in the AWSSDK.SQS NuGet package).
- Depending upon where the code is running, say if its running on EC2, the role needs to have permission to write into the SQS Queue.

Attachments

attachment.zip

Migrate data from Microsoft Azure Blob storage and Backblaze B2 Cloud Storage to Amazon S3 by using Rclone

*Created by Suhas Basavaraj (AWS)*

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source:</th>
<th>Microsoft Azure storage container and Backblaze B2 bucket</th>
<th>Target:</th>
<th>Amazon S3 bucket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Storage &amp; backup; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Microsoft</td>
<td>AWS services:</td>
<td>Amazon S3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to use Rclone to migrate data from Microsoft Azure Blob storage or Backblaze B2 Cloud Storage containers to an Amazon Simple Storage Service (Amazon S3) bucket. Rclone is a command-line program written in Go and used to synchronize files and directories from cloud providers.

Prerequisites and limitations

**Prerequisites**

- An active AWS account
AWS Prescriptive Guidance Patterns
Migrate data from Microsoft Azure Blob and Backblaze B2 to Amazon S3

- An existing S3 bucket with AWS Identity and Access Management (IAM) user permissions
- Data stored in Azure Blob or Backblaze B2 containers

Product versions
- Rclone v1.52.3

Architecture

Source technology stack
- Azure Blob or Backblaze B2 containers

Target technology stack
- Amazon S3 bucket
Tools

- **Amazon S3** - A highly scalable object storage service. It can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- **Rclone** - A command-line program used to manage files on cloud storage.
# Epics

## Prepare destination S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select destination S3 bucket.</td>
<td>Sign in to the AWS Management Console and open the Amazon S3 console. Choose a destination S3 bucket for the data you will migrate. For more information, see the “Related resources” section.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Copy S3 bucket access information.</td>
<td>Open the IAM console. Copy and store the access key ID, secret access key, and AWS Region for your S3 bucket. For more information, see the “Related resources” section.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Add policy to S3 bucket.</td>
<td>Open the Amazon S3 console and choose your bucket. Choose “Permissions,” and then choose “Bucket policy.” Paste the policy JSON file and choose “Save.” See the “Attachments” section for the policy JSON file.</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Attach policy to IAM user.</td>
<td>Open the IAM console and assign this policy to a user; see the “Related resources” section for more information. The resource entry must include both resource Amazon Resource Names (ARNs).</td>
<td>AWS General, Developer</td>
</tr>
<tr>
<td>Prepare source storage containers.</td>
<td>Retrieve account names and access keys for the containers you want to migrate data from.</td>
<td>AWS General, Developer</td>
</tr>
</tbody>
</table>

## Install and configure Rclone with source containers

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download and install Rclone.</td>
<td>Download and install the Rclone command line program. For the installation commands, see the “Additional information” section.</td>
<td>SSH, Developer</td>
</tr>
<tr>
<td>Configure Rclone to connect to your source containers.</td>
<td>Run the “rclone config” command; it will open an interactive form. Press “Enter” to keep the default selections, and enter these commands with your source container information:</td>
<td>Rclone, Developer</td>
</tr>
</tbody>
</table>
# AWS Prescriptive Guidance Patterns

## Migrate data from Microsoft Azure Blob and Backblaze B2 to Amazon S3

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure Rclone to connect to Amazon S3.</strong></td>
<td>Run the &quot;rclone config&quot; command; it will open an interactive form. Press “Enter” to keep the default selections, and then enter your S3 bucket information, including the access key ID, secret access key, the AWS Region, and access control list (ACL). For further instructions, see the rclone.conf file in the &quot;Attachments&quot; section.</td>
<td>Rclone, Developer</td>
</tr>
<tr>
<td><strong>Verify Rclone configuration.</strong></td>
<td>To check that Rclone is configured correctly, verify the contents of &quot;.config/rclone/rclone.conf&quot; in your home directory. For further instructions, see the rclone.conf file in the &quot;Attachments&quot; section.</td>
<td>Rclone, Developer</td>
</tr>
<tr>
<td><strong>Migrate data using Rclone</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Verify source container connectivity.</strong></td>
<td>In Rclone, run &quot;rclone listremotes&quot; to check that the Azure Blob or Backblaze B2 containers are in the remotes list. Check that you can access objects inside your Azure Blob or Backblaze B2 containers by using the Rclone commands in the verification.txt file available in the “Attachments” section.</td>
<td>Rclone, Developer</td>
</tr>
<tr>
<td><strong>Migrate data from your containers.</strong></td>
<td>Run the Rclone &quot;Copy&quot; command. For instructions, see the migrate_data.txt file in the “Attachments” section. This command copies the data from the source container to the destination S3 bucket. Important: If you run “Sync” instead of “Copy,” data that isn’t present in the source container will be deleted from the destination S3 bucket.</td>
<td>Rclone, Developer</td>
</tr>
<tr>
<td><strong>Sync your containers.</strong></td>
<td>After the initial copy is completed, run the Rclone</td>
<td>Rclone, Developer</td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sync” command for ongoing migration, so that only new files missing from the destination S3 bucket will be copied. For further instructions, see the migrate_data.txt file in the “Attachments” section.</td>
<td></td>
<td>AWS General, Developer, Rclone</td>
</tr>
<tr>
<td>Verify data has successfully migrated.</td>
<td>To check that data was successfully copied to the destination S3 bucket, run the Rclone “lsd” and “ls” commands. For more information, see the verification.txt file in the “Attachments” section.</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

- Creating an S3 bucket
- Adding an S3 bucket policy
- Adding IAM policies and permissions
- Installing Rclone
- Configuring Rclone
- Creating a Backblaze B2 container
- Creating a Microsoft Azure Blob container

### Additional information

Use these scripts to install Rclone. For Linux, macOS, and BSD systems, run:

```bash
curl https://rclone.org/install.sh | sudo bash
```

For beta installation, run:

```bash
curl https://rclone.org/install.sh | sudo bash -s beta
```

### Attachments

attachment.zip

### Migrate from Couchbase Server to Couchbase Cloud on AWS

*Created by Battulga Purevragchaa (AWS) and Mark Gamble (Couchbase)*

| Environment: Production | Source: Couchbase Server | Target: Couchbase Cloud |
Summary

Couchbase Cloud is a fully managed, NoSQL database as a service (DBaaS) for mission-critical applications (for example, user profiles or online catalogs and inventory management). Couchbase Cloud helps manage workloads in your Amazon Web Services (AWS) account and uses transparent in-VPC deployments, multiple cluster management in a single interface, and hybrid cloud replication.

Couchbase Cloud helps you instantly scale your Couchbase Server applications, while removing the need to take care of installations, upgrades, backups, or general database maintenance. Couchbase Cloud provides scalability, helps you create multi-node clusters in minutes, and supports all Couchbase Server features (for example, N1QL, Full Text Search, Eventing Service, and Analytics Service).

This pattern describes the steps and best practices for migrating a self-managed Couchbase Server environment to Couchbase Cloud on the AWS Cloud. The pattern provides a repeatable process for migrating data and indexes from Couchbase Server clusters, running either on premises or in the cloud, to Couchbase Cloud.

Prerequisites and limitations

Prerequisites

- An active AWS account. For more information about the permissions and resource capacity required for your AWS account, see Cloud provider requirements in the Couchbase documentation.
- An existing Couchbase Cloud account. You can also create a Couchbase Cloud account on AWS and use the Couchbase Cloud free trial. Follow the instructions in the Getting Started with Couchbase Cloud tutorial and then upgrade to a paid Couchbase Cloud account to configure your cluster for the migration.
- An existing self-managed Couchbase Server environment either on premises or deployed on a cloud service provider.
- This pattern provides the following two migration options:
  - **Option 1** is appropriate if you have less than 50 indexes to migrate.
  - **Option 2** is appropriate if you have more than 50 indexes to migrate. If you use this migration option, you must install Couchbase Shell and create a configuration file by using the sample file from the Code section.

The migration process also requires the roles and expertise described in the following table.
AWS Prescriptive Guidance Patterns
Migrate from Couchbase Server to Couchbase Cloud

| Systems administrator, IT administrator | • Basic command line knowledge is helpful but not required | • Familiarity with self-managed Couchbase Server system environment and administration | • Open ports and determine fully qualified domain name (FQDN) and IP addresses on self-managed Couchbase Server cluster nodes |

Skills
• Familiarity with administering Couchbase Server and Couchbase Cloud
• Familiarity with opening TCP ports and running commands in a command line interface (CLI)

Limitations
• This pattern is used to migrate data and indexes from Couchbase Server to Couchbase Cloud on AWS. The pattern doesn’t apply to migrating Couchbase Eventing Service, Couchbase Analytics, or Couchbase Full Text Search.
• Couchbase Cloud is only available in the following AWS Regions:
  • US East (Ohio) Region (us-east-2)
  • US East (N. Virginia) Region (us-east-1)
  • US West (Oregon) Region (us-west-2)
  • Asia Pacific (Mumbai) Region (ap-south-1)
  • Asia Pacific (Seoul) Region (ap-northeast-2)
  • Asia Pacific (Singapore) Region (ap-southeast-1)
  • Asia Pacific (Sydney) Region (ap-southeast-2)
  • Asia Pacific (Tokyo) Region (ap-northeast-1)
  • Canada (Central) Region (ca-central-1)
  • Europe (Frankfurt) Region (eu-central-1)
  • Europe (Ireland) Region (eu-west-1)
  • Europe (London) Region (eu-west-2)
  • Europe (Paris) Region (eu-west-3)
  • Europe (Stockholm) Region (eu-north-1)

Product versions
• These migration steps apply to self-managed instances of Couchbase Server Enterprise Edition version 5.x or later.

Architecture
Source technology stack
• Couchbase Server Enterprise Edition version 5.x or later
  • Buckets
  • Indexes
Target technology stack

- Couchbase Cloud, with in-VPC deployment of clusters to your AWS account
  - Buckets
  - Indexes

Target architecture

1. You can access Couchbase Cloud by using the **Couchbase Cloud Control Plane**. You can use the Couchbase Cloud Control Plane control and monitor your account, cluster and data management, indexes, users and groups, access permissions, and event management.

2. The **Couchbase Cloud Data Plane** deploys in a VPC in your AWS account. After you create a new cluster, Couchbase Cloud deploys an Amazon Elastic Kubernetes Service (Amazon EKS) cluster in your VPC and the **Couchbase Autonomous Operator** provides management capabilities. Your cluster automatically deploys across multiple Availability Zones in the AWS account’s Region.

3. You can also develop and deploy Couchbase applications in another VPC in your AWS account. Typically, this VPC accesses the Couchbase Cloud Data Plane through VPC peering or AWS Transit Gateway.

Tools

- **Couchbase XDCR** – Couchbase Cross Data Center Replication (XDCR) allows data to be replicated across clusters that are located in different cloud providers and different data centers.
- **Couchbase Shell** – Couchbase Shell is a command-line shell for Couchbase Server and Couchbase Cloud to access local and remote Couchbase clusters. Couchbase Shell is used to migrate indexes in this pattern.

Important: The **Prepare your migration** epic provides general guidance for assessing the size of your self-managed Couchbase Server cluster and using it to identify the required configurations for your Couchbase Cloud target cluster. For help with a more detailed Couchbase Cloud sizing exercise, please contact Couchbase Cloud.

Code

The following code is a sample configuration file for Couchbase Shell. You must use this sample configuration file if you choose **option 2** for your migration:
Version = 1

[[clusters]]
identifier = "On-Prem-Cluster"
hostnames = ["<SELF_MANAGED_COUCHBASE_CLUSTER>"]
default-bucket = "travel-sample"
username = "<SELF_MANAGED_ADMIN>"
password = "<SELF_MANAGED_ADMIN_PWD>"
cert-path = "<ABSOLUTE_PATH_TO_SELF_MANAGED_ROOT_CERT>"
data-timeout = "2500ms"
connect-timeout = "7500ms"
query-timeout = "75s"

[[clusters]]
identifier = "Cloud-Cluster"
hostnames = ["<COUCHBASE_CLOUD_ENDPOINT>"]
default-bucket = "travel-sample"
username = "<CLOUD_DATABASE_USER>"
password = "<CLOUD_DATABASE_USER_PWD>"
cert-path = "<ABSOLUTE_PATH_TO_COUCHBASE_CLOUD_ROOT_CERT>"
data-timeout = "2500ms"
connect-timeout = "7500ms"
query-timeout = "75s"

Before saving the configuration file, use the following table to make sure that you added your own target and destination cluster information.

| `<SELF_MANAGED_COUCHBASE_CLUSTER>` | Use the IP address for your self-managed Couchbase Server cluster |
| `<SELF_MANAGED_ADMIN>` | Use the administrator user for your self-managed Couchbase Server cluster |
| `<ABSOLUTE_PATH_TO_SELF_MANAGED_ROOT_CERT>` | Use the absolute path to the saved root certificate file for your self-managed Couchbase Server cluster |
| `<COUCHBASE_CLOUD_ENDPOINT>` | Use the connection endpoint for your self-managed Couchbase Server cluster |
| `<CLOUD_DATABASE_USER>` | Use the database user for your Couchbase Cloud cluster |
| `<CLOUD_DATABASE_USER_PWD>` | Use the database user password for your Couchbase Cloud cluster |
| `<ABSOLUTE_PATH_TO_COUCHBASE_CLOUD_ROOT_CERT>` | Use the absolute path to the saved root certificate file for your Couchbase Cloud cluster |
## Epics

### Prepare the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluate the size of the self-managed Couchbase Server cluster.</td>
<td>Log in to the Couchbase Web Console for Couchbase Server and assess your self-managed cluster’s nodes and buckets. Choose the <strong>Servers</strong> tab in the navigation bar to show a list of cluster nodes. Record the number of nodes and then choose each node on the list to display its properties. Record the memory and storage for each individual node. Choose the <strong>Buckets</strong> tab in the navigation bar and then choose each bucket in the list to display its properties. Record the RAM quota and conflict resolution setting for each bucket. You must use your self-managed Couchbase Server cluster configurations as a general guide to correctly size and configure the destination cluster on Couchbase Cloud.</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Record Couchbase Service distribution on the self-managed Couchbase Server cluster.</td>
<td>In the Couchbase Web Console, choose the <strong>Servers</strong> tab to display the list of cluster nodes. Choose each node to display its properties and then record the Couchbase Service distribution for each node (<strong>Data Service</strong>, <strong>Query Service</strong>, <strong>Index Service</strong>, <strong>Search Service</strong>, <strong>Analytics Service</strong>, and <strong>Eventing Service</strong>).</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Record the connection properties of the self-managed Couchbase Server cluster nodes.</td>
<td>Record the FQDN of your self-managed Couchbase Server cluster and record the IP addresses of the cluster nodes. If you have a multi-node cluster, each node in the cluster must have a public FQDN entry that resolves to a public IP address and can be reached by the Couchbase Cloud Control Plane.</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
</tbody>
</table>
### For example, if you have a Couchbase Server cluster with three nodes that you want to connect to Couchbase Cloud, you can use the nslookup tool for the nodes' FQDN to determine each IP address.

**Important:** You must open TCP ports 8091, 8093, and 8095 for inbound traffic on each node system.

### Deploy and configure resources on Couchbase Cloud

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect your Couchbase Cloud account to your AWS account.</td>
<td>Connect your Couchbase Cloud account to your AWS account by following the instructions in Connect a Cloud from the Couchbase Cloud documentation. You must make sure that your AWS account has the required permissions to connect to your Couchbase Cloud account. For more information about this, see Cloud provider requirements in the Couchbase documentation. <strong>Important:</strong> If you are using the Couchbase Cloud free trial, you must convert it to a paid account before beginning your migration. To convert your account, open the Billing section of the Couchbase Cloud Control Plane and then choose Add Activation ID.</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
<tr>
<td>Create and configure a destination cluster on Couchbase Cloud.</td>
<td>Log in to your Couchbase Cloud Control Plane, choose the Dashboard or Clusters tab in the main navigation menu and then choose Create Cluster. Use the information that you recorded from the review of your self-managed Couchbase Server cluster and choose the cluster template that meets the configuration’s requirements.</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>you don't find an appropriate template, choose <strong>Custom Template</strong> in the <strong>Cluster Sizing</strong> editor. Choose and configure the nodes to match your self-managed Couchbase Server cluster environment, including number of nodes, services distribution, compute or RAM, and storage. Choose a support zone and support package, and then deploy the cluster. For detailed steps and instructions, see Create a cluster in the Couchbase documentation. <strong>Important:</strong> Couchbase Cloud uses multi-dimensional scaling best practices. Services and nodes can only be chosen according to deployment best practices. This could mean that you can't exactly match your self-managed Couchbase Server cluster's configurations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a database user. Create a database user for the new cluster by following the instructions in Configure database users from the Couchbase documentation. A database user is specific to a cluster and consists of a username, password, and a set of bucket privileges. A database user is required for creating buckets and accessing bucket data. An organization user needs a database user assigned to them if they want to access bucket data on a particular cluster, either remotely or through the Couchbase Cloud user interface (UI). Creating the database user allows you to create and manage the target buckets in your Couchbase Cloud cluster.</td>
<td>Couchbase administrator</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Create target buckets.</td>
<td>Create one target bucket in your Couchbase Cloud cluster for each source bucket by following the instructions from Create a bucket in the Couchbase documentation. Your target bucket configurations must match the bucket names, memory settings, and conflict resolution methods of the buckets in your self-managed Couchbase Server cluster. <strong>Important</strong>: Bucket names cannot contain an underscore. If your target bucket name doesn’t match the source bucket name, you must record the mapping of the source bucket name to target bucket name.</td>
<td>Couchbase administrator</td>
</tr>
</tbody>
</table>

**Migrate the data and indexes – option 1**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open TCP ports on the self-managed Couchbase Server cluster nodes.</td>
<td>Make sure that TCP ports 8091, 8093, and 8095 are open for inbound traffic on the self-managed Couchbase Server cluster’s nodes.</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
<tr>
<td>Connect self-managed Couchbase Server cluster to Couchbase Cloud cluster.</td>
<td>Follow the instructions from Connect a self-managed cluster and connect your self-managed Couchbase Server cluster to your Couchbase Cloud account. Make sure that you use the self-managed Couchbase Server cluster’s FQDN in the Host Name property and enter the IP addresses of the target cluster nodes in the Public/NAT gateway IPs section. The IP addresses must be separated by commas.</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Set up Couchbase XDCR.</td>
<td>Couchbase Cross Data Center Replication (XDCR) helps replicate data across clusters that are located in different</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>cloud providers and data centers. It is used to migrate data into Couchbase Cloud from self-managed Couchbase Server clusters. In the Couchbase Cloud Control Plane, set up an XDCR replication for your self-managed Couchbase Server cluster. For steps and instructions, see Manage replications in the Couchbase documentation. <strong>Important:</strong> Leave the Bidirectional box unchecked.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| Migrate self-managed cluster indexes to Couchbase Cloud. | **Important:** We recommend this process if you have less than 50 indexes to migrate. If you have more than 50 indexes to migrate, we recommend that you use migration option 2. In the Couchbase Web Console, choose **Indexes**. In the list of indexes, choose the first index that you want to migrate. The index definition is then displayed. Copy the index definition by using the `CREATE` statement but don’t copy `WITH` `{ "defer_build":true }`. For example, you would only copy `CREATE INDEX `cityindex` ON `travel-sample`(`city`) from the following sample index definition:  

```
CREATE INDEX `cityindex` ON `travel-sample`(`city`) WITH `{ "defer_build":true }`
```

In the Couchbase Cloud Control Plane, choose **Clusters** and then choose the destination cluster. Choose **Query Workbench** from the **Tools** dropdown list. Paste the `CREATE` statement that you copied earlier into the **Query Editor** and then choose **Execute**. This creates and builds the index. To confirm that the index is created, choose **Indexes** from the **Tools** dropdown list. The list shows that the index was created and built. Repeat this process for each index that must be migrated. | Couchbase administrator, Systems administrator |
## Migrate the data and indexes – option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open TCP ports on the self-managed Couchbase Server cluster nodes.</td>
<td>Make sure that TCP ports 8091, 8093, and 8095 are open for inbound traffic on the self-managed Couchbase Server cluster’s nodes.</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
<tr>
<td>Connect self-managed Couchbase Server cluster to Couchbase Cloud cluster.</td>
<td>Follow the instructions from Connect a self-managed cluster and connect your self-managed Couchbase Server cluster to your Couchbase Cloud account. Make sure that you use the self-managed Couchbase Server cluster’s FQDN in the Host Name property and enter the IP addresses of the target cluster nodes in the Public/NAT gateway IPs section. The IP addresses must be separated by commas.</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
<tr>
<td>Set up Couchbase XDCR.</td>
<td>Couchbase Cross Data Center Replication (XDCR) helps replicate data across clusters that are located in different cloud providers and different data centers. It is used to migrate data into Couchbase Cloud from self-managed Couchbase Server clusters. On the Couchbase Cloud Control Plane, set up an XDCR replication in your source Couchbase Server self-managed cluster. For steps and instructions, see Manage replications in the Couchbase documentation. Important: Leave the Bidirectional box unchecked.</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Prepare to migrate the self-managed cluster indexes.</td>
<td>Important: We recommend this process if you have more than 50 indexes to migrate. If you have less than 50 indexes to migrate, we recommend that you choose migration option 1. You must use Couchbase Shell to migrate the index definitions. You can install Couchbase</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
|      | Shell on any system that has internet access to both your self-managed Couchbase Server and Couchbase Cloud clusters. For more information, see [Install Couchbase Shell version 0.5.0 in the Couchbase Shell documentation](#). Make sure that you confirm that Couchbase Shell is installed by testing a connection to your self-managed cluster in a command line terminal. **Note:** This migration option requires that your target bucket names are identical to the source bucket names. | }
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Couchbase Cloud cluster's connection and add IP address.</td>
<td>In the Couchbase Cloud Control Plane, choose <strong>Clusters</strong> and then choose your destination cluster. Choose the <strong>Connect</strong> tab for the cluster and record the connection endpoint for your cluster under <strong>Wide Area Network</strong>. Add the IP address for the system where Couchbase Shell is installed as an <strong>Allowed IP</strong> for your Couchbase Cloud cluster. For more information about this, see <strong>Configure allowed IP addresses</strong> in the Couchbase documentation. In the Couchbase Web Console, copy the root certificate for your self-managed Couchbase Server cluster and save it as a .pem file in a folder on the system that runs Couchbase Shell. For more information about the root certificate, see <strong>Root certificate</strong> in the Couchbase documentation. In the Couchbase Cloud Control Plane, choose <strong>Clusters</strong> and then choose your destination cluster. Choose the <strong>Connect</strong> tab and then choose <strong>Download</strong> under <strong>Root Certificate</strong>. Save the root certificate as a .pem file to the same folder where you saved the root certificate file for your self-managed Couchbase Server cluster</td>
<td>Couchbase administrator, Systems administrator</td>
</tr>
</tbody>
</table>
## Task

Create the configuration file for Couchbase Shell.

## Description

Use the sample configuration file provided in the Code section to create a configuration file for Couchbase Shell. This allows Couchbase Shell to connect to your self-managed Couchbase Server and Couchbase Cloud clusters.

Create a configuration dotfile in the Couchbase Shell installation's home directory (for example, `/ <HOME_DIRECTORY>/.cbsh/config`). For more information, see Config dotfiles in the Couchbase documentation.

Add connection properties for the target and destination clusters to the configuration file. You can use the sample configuration file provided by this pattern and edit the settings for your clusters.

Save the configuration file with the updated settings to the .cbsh folder (for example, `/ <HOME_DIRECTORY>/.cbsh/config`).

## Skills required

Couchbase administrator, Systems administrator
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Migrate the index definitions. | Open the command line terminal on the system where you installed Couchbase Shell.  
Run the following command to start Couchbase Shell:  
./cbsh  
Run the following command to connect to the self-managed Couchbase Server cluster:  
use cluster On-Prem-Cluster  
For each bucket that you want to migrate, run the following command to migrate index definitions from the self-managed Couchbase Server cluster to the Couchbase Cloud cluster:  
query indexes --definitions | where bucket =~ "<BUCKET_NAME>"| get definition | each  
{ query $it --cluster Cloud-Cluster }  
**Note:** Make sure that you replace `<BUCKET_NAME>` with the bucket name that corresponds to the indexes that you want to migrate.  
Repeat for each bucket. | Couchbase administrator, Systems administrator |
### Build the index definitions.

Run the following command to switch context to the Couchbase Cloud cluster:

```
use cluster Cloud-Cluster
```

Run the following command to build the index definitions that were migrated to the Couchbase Cloud cluster:

```
query "BUILD INDEX ON <BUCKET_NAME> ((SELECT RAW name FROM system:indexes WHERE keyspace_id = '<BUCKET_NAME>' AND state = 'deferred'));"
```

**Note:** Replace `<BUCKET_NAME>` with the bucket name that corresponds to the indexes you want to build.

Repeat for each bucket.

### Test and verify the migration

#### Verify data migration.

In the Couchbase Cloud Control Plane, choose **Clusters** and then choose the target cluster in your cluster list. Choose the **Buckets** tab for your target cluster.

Verify that the number of **Items** (documents) in the target bucket match the number of items in the source bucket.

In the target cluster, choose **Documents** in the **Tools** dropdown list. Verify that all documents were migrated.

(Optional) After all data is migrated, you can shut down the replication by deleting it. For more information about this, see [Delete a replication](https://docs.aws.amazon.com/couchbase/latest/guide/delete-replication.html) in the Couchbase documentation.
### AWS Prescriptive Guidance Patterns

**Migrate from Couchbase Server to Couchbase Cloud**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify index migration.</td>
<td>In the Couchbase Cloud Control Plane, choose <strong>Indexes</strong> in the <strong>Tools</strong> dropdown list for your target cluster. Verify that the indexes are migrated and built.</td>
<td>Couchbase administrator</td>
</tr>
<tr>
<td>Verify query runtime.</td>
<td>In the Couchbase Cloud Control Plane, choose <strong>Query Workbench</strong> in the <strong>Tools</strong> dropdown list for your target cluster. Run a sample N1QL query or a query used in your application. Make sure that you receive the same results as the query in your self-managed Couchbase Server cluster.</td>
<td>Couchbase administrator</td>
</tr>
</tbody>
</table>

### Related resources

#### Prepare the migration

- Get started with the Couchbase Cloud free trial
- Cloud provider requirements for Couchbase Cloud
- Couchbase Cloud documentation
- Create a Couchbase Cloud account
- Connect to AWS from Couchbase Cloud
- Couchbase Cloud sizing guidelines
- Couchbase Cloud cluster scaling
- Couchbase Server documentation
- Couchbase Server cluster sizing guidelines

#### Deploy and configure resources on Couchbase Cloud

- Configure a database user on Couchbase Cloud
- Couchbase Cloud buckets overview
- Couchbase Cloud cluster overview
- Couchbase Cloud projects overview
- Create a cluster on Couchbase Cloud
- Create a new bucket on Couchbase Cloud

#### Migrate the data and indexes

- Connect a self-managed cluster on Couchbase Cloud
- Set up XDCR between a self-managed cluster and Couchbase Cloud cluster
- Couchbase Shell documentation

---

**Couchbase Cloud Support and SLAs**

---

1181
Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2

*Created by Neal Ardeljan (AWS)*

<table>
<thead>
<tr>
<th><strong>R Type:</strong> Replatform</th>
<th><strong>Source:</strong> Applications</th>
<th><strong>Target:</strong> Apache Tomcat on an Amazon EC2 instance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Created by:</strong> AWS</td>
<td><strong>Environment:</strong> Production</td>
<td><strong>Technologies:</strong> Websites &amp; web apps; Migration</td>
</tr>
<tr>
<td><strong>Workload:</strong> IBM; Open-source</td>
<td><strong>AWS services:</strong> Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern walks you through the steps for migrating from an on-premises Red Hat Enterprise Linux (RHEL) 6.9 system running IBM WebSphere to RHEL 7.4 running Apache Tomcat on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

The pattern can be applied to the following source and target versions:

- WebSphere 7 to Tomcat 6 (with Java 1.6)
- WebSphere 7 to Tomcat 7 (with Java 1.6)
- WebSphere 8.5.5 to Tomcat 7 (with Java 1.6 or higher)

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- Source Java code, with the following assumptions:
  - Uses the Java Development Kit (JDK) version of Java 1.6 or higher
  - Uses the Spring or Apache Struts framework
  - Doesn't use the Enterprise Java Beans (EJB) framework or any other WebSphere server functionality that's not readily available for Tomcat
  - Primarily uses servlets or Java Server Pages (JSPs)
  - Uses Java Database Connectivity (JDBC) connectors to connect to databases
- Source IBM WebSphere version 7 or higher
- Target Tomcat version 6 or higher

**Architecture**

*Source technology stack*
- A web application built using the Apache Struts Model-View-Controller (MVC) framework
- A web application running on IBM WebSphere version 7 or 8.5.5.
- A web application that uses a Lightweight Directory Access Protocol (LDAP) connector to connect to an LDAP directory (iPlanet/eTrust)
- An application that uses IBM Tivoli Access Manager (TAM) connectivity to update the TAM user password (in the present implementation, applications use PD.jar)

**On-premises databases**
- Oracle 11g Standard Edition (SE)
- Oracle 11g Standard Edition 1 (SE1)
- Oracle 11g Standard Edition 2 (SE2)
- Oracle 11g Enterprise Edition (EE)

**Target technology stack**
- Apache Tomcat version 7 running on RHEL on an EC2 instance
- Amazon Relational Database Service (Amazon RDS) for Oracle

For more information about the Oracle versions supported by Amazon RDS, see the Amazon RDS for Oracle website.

**Target architecture**
Tools

- Application tier: Rebuilding Java application into a WAR file
- Database tier: Oracle native backup and restore
Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete the application discovery, current state footprint, and performance baseline.</td>
<td></td>
<td>BA, Migration Lead</td>
</tr>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server EC2 instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper EC2 instance type based on capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy and tooling.</td>
<td></td>
<td>DBA, Migration Lead</td>
</tr>
<tr>
<td>Complete the migration design and migration guide for the application.</td>
<td></td>
<td>Build Lead, Migration Lead</td>
</tr>
<tr>
<td>Complete the application migration runbook.</td>
<td></td>
<td>Build Lead, Cutover Lead, Testing Lead, Migration Lead</td>
</tr>
</tbody>
</table>

Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create the security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start Amazon RDS for Oracle.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create or obtain access to the endpoints to fetch the database backup files.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Use the native database engine or a third-party tool to migrate database objects and data.</td>
<td>For details, see &quot;Migrating database objects and data&quot; in the Additional information section.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lodge the change request (CR) for migration.</td>
<td></td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Obtain the CR approval for migration.</td>
<td></td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Follow the application migration strategy per the application migration runbook.</td>
<td>For details, see &quot;Setting up the application tier&quot; in the Additional information section.</td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Upgrade the application (if necessary).</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Complete the functional, non-functional, data validation, SLA, and performance tests.</td>
<td></td>
<td>Testing Lead, App Owner, App Users</td>
</tr>
</tbody>
</table>

Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obtain signoff from the application owner or business owner.</td>
<td></td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Switch the application clients to the new infrastructure.</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, Migration Engineer, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents</td>
<td></td>
<td>Migration Lead</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Migrate from IBM WebSphere to Apache Tomcat on Amazon EC2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gather the metrics around time to migrate, percentage of manual vs. tool, cost savings, etc.</td>
<td>Migration Lead</td>
</tr>
<tr>
<td></td>
<td>Close out the project and provide feedback.</td>
<td>Migration Lead, App Owner</td>
</tr>
</tbody>
</table>

**Related resources**

**References**
- Apache Tomcat 7.0 documentation
- Apache Tomcat 7.0 installation guide
- Apache Tomcat JNDI documentation
- Amazon RDS for Oracle website
- Amazon RDS pricing
- Oracle and Amazon Web Services
- Oracle on Amazon RDS
- Amazon RDS Multi-AZ Deployments

**Tutorials and videos**
- Getting Started with Amazon RDS

**Additional information**

**Migrating database objects and data**
For example, if you’re using native Oracle backup/restore utilities:

1. Create the Amazon Simple Storage Service (Amazon S3) backup for database backup files (optional).
2. Back up the Oracle DB data to the network shared folder.
3. Log in to the migration staging server to map the network share folder.
4. Copy data from the network share folder to the S3 bucket.
5. Request an Amazon RDS Multi-AZ deployment for Oracle.
6. Restore the on-premises database backup to Amazon RDS for Oracle.

**Setting up the application tier**

1. Install Tomcat 7 from the Apache Tomcat website.
2. Package the application and shared libraries into a WAR file.
3. Deploy the WAR file in Tomcat.
4. Monitor the start log to Linux `cat` any missing shared libraries from WebSphere.
5. Watch the start record to Linux `cat` any WebSphere-specific deployment descriptor extensions.
6. Collect any missing dependent Java libraries from the WebSphere server.
7. Amend WebSphere-specific deployment descriptor elements with Tomcat-compatible equivalents.
8. Rebuild the WAR file with the dependent Java libraries and updated deployment descriptors.
9. Update the LDAP configuration, database configuration, and test connections (see Realm Configuration HOW-TO and JNDI Datasource HOW-TO in the Apache Tomcat documentation).
10. Test the installed application against the restored Amazon RDS for Oracle database.
11. Create an Amazon Machine Image (AMI) for Linux from the EC2 instance.
12. Launch the completed architecture with the Application Load Balancer and Auto Scaling group.
13. Update the URLs (by using the WebSEAL junction) to point to the Application Load Balancer.
14. Update the configuration management database (CMDB).

Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling

*Created by Kevin Yung (AWS)*

<table>
<thead>
<tr>
<th>R Type: Replatform</th>
<th>Source: Applications</th>
<th>Target: Apache Tomcat on an Amazon EC2 instance with Auto Scaling enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Websites &amp; web apps; Migration</td>
</tr>
<tr>
<td>Workload: Open-source; IBM</td>
<td>AWS services: Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern provides guidance for migrating a Java application from IBM WebSphere Application Server to Apache Tomcat on an Amazon Elastic Compute Cloud (Amazon EC2) instance with Amazon EC2 Auto Scaling enabled.

By using this pattern, you can achieve:

- A reduction in IBM licensing costs
- High availability using Multi-AZ deployment
- Improved application resiliency with Amazon EC2 Auto Scaling

**Prerequisites and limitations**

**Prerequisites**

- Java applications (version 7.x or 8.x) should be developed in LAMP stacks.
- The target state is to host Java applications on Linux hosts. This pattern has been successfully implemented in a Red Hat Enterprise Linux (RHEL) 7 environment. Other Linux distributions can follow this pattern, but the configuration of the Apache Tomcat distribution should be referenced.
- You should understand the Java application's dependencies.
- You must have access to the Java application source code to make changes.

**Limitations and replatforming changes**
• You should understand the enterprise archive (EAR) components and verify that all libraries are packaged in the web component WAR files. You need to configure the Apache Maven WAR Plugin and produce WAR file artifacts.

• When using Apache Tomcat 8, there is a known conflict between servlet-api.jar and the application package built-in jar files. To resolve this issue, delete servlet-api.jar from the application package.

• You must configure WEB-INF/resources located in the classpath of the Apache Tomcat configuration. By default, the JAR libraries are not loaded in the directory. Alternatively, you can deploy all the resources under src/main/resources.

• Check for any hard-coded context roots within the Java application, and update the new context root of Apache Tomcat.

• To set JVM runtime options, you can create the configuration file setenv.sh in the Apache Tomcat bin folder; for example, JAVA_OPTS, JAVA_HOME, etc.

• Authentication is configured at the container level and is set up as a realm in Apache Tomcat configurations. Authentication is established for any of the following three realms:
  • JDBC Database Realm looks up users in a relational database accessed by the JDBC driver.
  • DataSource Database Realm looks up users in a database that is accessed by JNDI.
  • JNDI Directory Realm looks up users in the Lightweight Directory Access Protocol (LDAP) directory that is accessed by the JNDI provider. The look-ups require:
    • LDAP connection details: user search base, search filter, role base, role filter
    • The key JNDI Directory Realm: Connects to LDAP, authenticates users, and retrieves all groups in which a user is a member

• Authorization: In the case of a container with a role-based authorization that checks the authorization constraints in web.xml, web resources must be defined and compared to the roles defined in the constraints. If LDAP doesn’t have group-role mapping, you must set the attribute <security-role-ref> in web.xml to achieve group-role mapping. To see an example of a configuration document, see the Oracle documentation.

• Database connection: Create a resource definition in Apache Tomcat with an Amazon Relational Database Service (Amazon RDS) endpoint URL and connection details. Update the application code to reference a DataSource by using JNDI lookup. An existing DB connection defined in WebSphere would not work, as it uses WebSphere’s JNDI names. You can add a <resource-ref> entry in web.xml with the JNDI name and DataSource type definition. To see a sample configuration document, see the Apache Tomcat documentation.

• Logging: By default, Apache Tomcat logs to the console or to a log file. You can enable realm-level tracing by updating logging.properties (see Logging in Tomcat). If you are using Apache Log4j to append logs to a file, you must download tomcat-juli and add it to the classpath.

• Session management: If you are retaining IBM WebSEAL for application load balancing and session management, no change is required. If you are using an Application Load Balancer or Network Load Balancer on AWS to replace the IBM WebSEAL component, you must set up session management by using an Amazon ElastiCache instance with a Memcached cluster and set up Apache Tomcat to use open-source session management.

• If you are using the IBM WebSEAL forward proxy, you must set up a new Network Load Balancer on AWS. Use the IPs provided by the Network Load Balancer for WebSEAL junction configurations.

• SSL configuration: We recommend that you use Secure Sockets Layer (SSL) for end-to-end communications. To set up an SSL server configuration in Apache Tomcat, follow the instructions in the Apache Tomcat documentation.

Architecture

Source technology stack

• IBM WebSphere Application Server
Target technology stack

- The architecture uses Elastic Load Balancing (version 2). If you are using IBM WebSEAL for Identify management and load balancing, you can select a Network Load Balancer on AWS to integrate with the IBM WebSEAL reverse proxy.
- Java applications are deployed to an Apache Tomcat application server, which runs on an EC2 instance in an Amazon EC2 Auto Scaling group. You can set up a scaling policy based on Amazon CloudWatch metrics such as CPU utilization.
- If you’re retiring the use of IBM WebSEAL for load balancing, you can use Amazon ElastiCache for Memcached for session management.
- For the back-end database, you can deploy High Availability (Multi-AZ) for Amazon RDS and select a database engine type.

Target architecture

Tools

- AWS CloudFormation
- AWS Command Line Interface (AWS CLI)
- Apache Tomcat (version 7.x or 8.x)
- RHEL 7 or Centos 7
- Amazon RDS Multi-AZ deployment
- Amazon ElastiCache for Memcached (optional)

**Epics**

**Set up the VPC**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create subnets.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create routing tables if necessary.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create network access control lists (ACLs)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Set up AWS Direct Connect or a corporate VPN connection. | | }

**Replatform the application**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refactor the application build Maven configuration to generate the WAR artifacts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refactor the application dependency data sources in Apache Tomcat.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refactor the application source codes to use JNDI names in Apache Tomcat.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy the WAR artifacts into Apache Tomcat.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete application validations and tests.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Configure the network

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the corporate firewall to allow the connection to dependency services.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure the corporate firewall to allow end-user access to Elastic Load Balancing on AWS.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Create the application infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and deploy the application on an EC2 instance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an Amazon ElastiCache for Memcached cluster for session management.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an Amazon RDS Multi-AZ instance for the backend database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create SSL certificates and import them into AWS Certificate Manager (ACM).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install SSL certificates on load balancers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install SSL certificates for Apache Tomcat servers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete application validations and tests.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the existing infrastructure.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restore the database from production to Amazon RDS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cut over the application by making DNS changes.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Related resources

References

- Apache Tomcat 7.0 documentation
- Apache Tomcat 7.0 installation guide
- Apache Tomcat JNDI documentation
- Amazon RDS Multi-AZ Deployments
- Amazon ElastiCache for Memcached

Tutorials and videos

- Getting Started with Amazon RDS

Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk

*Created by Raghavender Madamshitti (AWS)*

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source:</th>
<th>Applications</th>
<th>Target:</th>
<th>AWS Elastic Beanstalk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Websites &amp; web apps; Migration</td>
</tr>
</tbody>
</table>

| Workload: | Microsoft |

Summary

This pattern describes how to migrate a .NET web application hosted on Microsoft Azure App Service to AWS Elastic Beanstalk. There are two ways to migrate applications to Elastic Beanstalk:

- Use AWS Toolkit for Visual Studio - This plugin for the Microsoft Visual Studio IDE provides the easiest and most straightforward way to deploy custom .NET applications to AWS. You can use this approach to deploy .NET code directly to AWS and to create supporting resources, such as Amazon Relational Database Service (Amazon RDS) for SQL Server databases, directly from Visual Studio.

- Upload and deploy to Elastic Beanstalk - Each Azure App Service includes a background service called Kudu, which is useful for capturing memory dumps and deployment logs, viewing configuration parameters, and accessing deployment packages. You can use the Kudu console to access Azure App Service content, extract the deployment package, and then upload the package to Elastic Beanstalk by using the upload and deploy option in the Elastic Beanstalk console.

This pattern describes the second approach (uploading your application to Elastic Beanstalk through Kudu). The pattern also uses the following AWS services: AWS Elastic Beanstalk, Amazon Virtual Private Cloud (Amazon VPC), Amazon CloudWatch, Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling, Amazon Simple Storage Service (Amazon S3), and Amazon Route 53.

The .NET web application is deployed to AWS Elastic Beanstalk, which runs in an Amazon EC2 Auto Scaling Group. You can set up a scaling policy based on Amazon CloudWatch metrics such as CPU
utilization. For a database, you can use Amazon RDS in a Multi-AZ environment, or Amazon DynamoDB, depending on your application and business requirements.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A .NET web application running in Azure App Service
- Permission to use the Azure App Service Kudu console

Product versions

- .NET Core (x64) 1.0.1, 2.0.0, or later, or .NET Framework 4.x, 3.5 (see .NET on Windows Server platform history)
- Internet Information Services (IIS) version 8.0 or later, running on Windows Server 2012 or later
- .NET 2.0 or 4.0 Runtime.

Architecture

Source technology stack

- Application developed using .NET Framework 3.5 or later, or .NET Core 1.0.1, 2.0.0, or later, and hosted on Azure App Service (web app or API app)

Target technology stack

- AWS Elastic Beanstalk running in an Amazon EC2 Auto Scaling group

Migration architecture
Deployment workflow

Tools

- .NET Core or .NET Framework
- C#
- IIS
- Kudu console

AWS services and features
AWS Elastic Beanstalk – Elastic Beanstalk is an easy-to-use service for deploying and scaling .NET web applications. Elastic Beanstalk automatically manages capacity provisioning, load balancing, and auto scaling.

Amazon EC2 Auto Scaling group – Elastic Beanstalk includes an Auto Scaling group that manages the Amazon EC2 instances in the environment. In a single-instance environment, the Auto Scaling group ensures that there is always one instance running. In a load-balanced environment, you can configure the group with a range of instances to run, and Amazon EC2 Auto Scaling adds or removes instances as needed, based on load.

Elastic Load Balancing – When you enable load balancing in AWS Elastic Beanstalk, it creates a load balancer that distributes traffic among the EC2 instances in the environment.

Amazon CloudWatch – Elastic Beanstalk automatically uses Amazon CloudWatch to provide information about your application and environment resources. Amazon CloudWatch supports standard metrics, custom metrics, and alarms.

Amazon Route 53 – Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. You can use Route 53 alias records to map custom domain names to AWS Elastic Beanstalk environments.

Epics

Set up a VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up a virtual private cloud (VPC).</td>
<td>In your AWS account, create a VPC with the required information.</td>
<td>System Admin</td>
</tr>
<tr>
<td>Create subnets.</td>
<td>Create two or more subnets in your VPC.</td>
<td>System Admin</td>
</tr>
<tr>
<td>Create a route table.</td>
<td>Create a route table, based on your requirements.</td>
<td>System Admin</td>
</tr>
</tbody>
</table>

Set up Elastic Beanstalk

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access the Azure App Service Kudu console.</td>
<td>Access Kudu through the Azure portal by navigating to the App Service dashboard, and then choosing &quot;Advanced Tools,&quot; &quot;Go.&quot; Or, you can modify the Azure App Service URL as follows: https://&lt;appservicename&gt;.scm.azurewebsites.net.</td>
<td>System Admin/ Application Developer</td>
</tr>
<tr>
<td>Download the deployment package from Kudu.</td>
<td>Navigate to Windows PowerShell by choosing the DebugConsole option. This will open the Kudo console. Go to the wwwroot folder and download it. This will</td>
<td>System Admin/ Application Developer</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate from Microsoft Azure App Service to AWS Elastic Beanstalk**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Download the Azure App Service</strong></td>
<td>deployment package as a zip file. For an example, see the attachment.</td>
<td></td>
</tr>
<tr>
<td><strong>Create a package for Elastic Beanstalk.</strong></td>
<td>Unzip the deployment package that you downloaded from Azure App Service. Create a JSON file called <code>aws-windows-deployment-manifest.json</code> (this file is required only for .NET Core applications). Create a zip file that includes <code>aws-windows-deployment-manifest.json</code> and the Azure App Service deployment package file. For an example, see the attachment.</td>
<td>System Admin/ Application Developer</td>
</tr>
<tr>
<td><strong>Create a new Elastic Beanstalk application.</strong></td>
<td>Open the Elastic Beanstalk console. Choose an existing application or create a new application.</td>
<td>System Admin/ Application Developer</td>
</tr>
<tr>
<td><strong>Create the environment.</strong></td>
<td>In the Elastic Beanstalk console, Actions menu, choose &quot;Create environment.&quot; Select the web server environment and .NET/ IIS platform. For application code, choose &quot;Upload.&quot; Upload the zip file that you prepared for Elastic Beanstalk, and then choose &quot;Create Environment.&quot;</td>
<td>System Admin/ Application Developer</td>
</tr>
<tr>
<td><strong>Configure Amazon CloudWatch.</strong></td>
<td>By default, basic CloudWatch monitoring is enabled. If you want to change the configuration, in the Elastic Beanstalk wizard, choose the published application, and then choose &quot;Monitoring.&quot;</td>
<td>System Admin</td>
</tr>
<tr>
<td><strong>Verify that the deployment package is in Amazon S3.</strong></td>
<td>When the application environment has been created, you can find the deployment package in the S3 bucket.</td>
<td>System Admin/ Application Developer</td>
</tr>
<tr>
<td><strong>Test the application.</strong></td>
<td>When the environment has been created, use the URL provided in the Elastic Beanstalk console to test the application.</td>
<td>System Admin</td>
</tr>
</tbody>
</table>

### Related resources

- [AWS Elastic Beanstack concepts](Elastic Beanstalk documentation)
- [Getting Started with .NET on Elastic Beanstalk](Elastic Beanstalk documentation)
Additional information

Notes

- If you are migrating an on-premises or Azure SQL Server database to Amazon RDS, you must update the database connection details as well.
- For testing purposes, a sample demo application is attached.

Attachments

attachment.zip

Migrate a self-hosted MongoDB environment to MongoDB Atlas on the AWS Cloud

Created by Suresh Veeragoni (AWS)

<table>
<thead>
<tr>
<th>Source:</th>
<th>Target:</th>
<th>R Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB</td>
<td>MongoDB Atlas on AWS</td>
<td>Replatform</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Technologies:</th>
<th>Workload:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production</td>
<td>Migration; Analytics; Databases</td>
<td>All other workloads</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AWS services:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2; Amazon VPC</td>
</tr>
</tbody>
</table>

Summary

This pattern describes the steps for migrating from a self-managed MongoDB environment (including MongoDB Community Server, Enterprise Server, Enterprise Advanced, mLab, or any managed MongoDB cluster) to MongoDB Atlas on the Amazon Web Services (AWS) Cloud. It uses the Atlas Live Migration Service to help accelerate the data migration from MongoDB to MongoDB Atlas.

The pattern accompanies the guide Migrating from MongoDB to MongoDB Atlas on the AWS Cloud on the AWS Prescriptive Guidance website. It provides the implementation steps for the migration.

The pattern is intended for AWS Service Integrator Partners (SI Partners) and AWS users.

Prerequisites and limitations

Prerequisites
A source MongoDB environment to migrate to MongoDB Atlas

Expertise

This pattern requires familiarity with MongoDB, MongoDB Atlas, and AWS services. For more information, see Roles and responsibilities in the guide Migrating from MongoDB to MongoDB Atlas on the AWS Cloud on the AWS Prescriptive Guidance website.

Product versions

MongoDB version 2.6 or later

Architecture

For MongoDB Atlas reference architectures that support different usage scenarios, see MongoDB Atlas reference architectures on AWS in the guide Migrating from MongoDB to MongoDB Atlas on the AWS Cloud on the AWS Prescriptive Guidance website.

Tools

Atlas Live Migration Service – A free MongoDB utility that helps migrate databases to Atlas. This service keeps the source database in sync with the destination database until cutover. When you’re ready to cut over, you stop your application instances, point them to the destination Atlas cluster, and restart them.

Epics

Discovery and assessment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the cluster size.</td>
<td>Estimate the working set size by using the information from db.stats() for the total index space. Assume that a percentage of your data space will be accessed frequently. Or, you can estimate your memory requirements based on your own assumptions. This task should take approximately one week. For more information and examples for this and the other stories in this epic, see the links in the “Related resources” section.</td>
<td>MongoDB DBA, Application architect</td>
</tr>
<tr>
<td>Estimate network bandwidth</td>
<td>To estimate your network bandwidth requirements, multiply the average document size by the number of documents served per second. Consider the maximum traffic</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>requirements.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>that any node on your cluster will bear as the basis. To calculate downstream data transfer rates from your cluster to client applications, use the sum of the total documents returned over a period of time. If your applications read from secondary nodes, divide this number of total documents by the number of nodes that can serve read operations. To find the average document size for a database, use the db.stats().avgObjSize command. This task will typically take one day.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Select the Atlas tier.</td>
<td>Follow the instructions in the MongoDB documentation to select the correct Atlas cluster tier.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Plan for application cutover.</td>
<td></td>
<td>MongoDB DBA, Application architect</td>
</tr>
</tbody>
</table>

Set up a new MongoDB Atlas environment on AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new MongoDB Atlas cluster on AWS.</td>
<td>In MongoDB Atlas, choose “Build a Cluster” to display the “Create New Cluster” dialog box. Select AWS as the cloud provider.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Select Regions and global cluster configuration.</td>
<td>Select from the list of available AWS Regions for your Atlas cluster. Configure global clusters if required.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Select the cluster tier.</td>
<td>Select your preferred cluster tier. Your tier selection determines factors such as memory, storage, and IOPS specification.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Configure additional cluster settings.</td>
<td>Configure additional cluster settings such as MongoDB version, backup, and encryption options. For more information about these options, see the links in the “Related resources” section.</td>
<td>MongoDB DBA</td>
</tr>
</tbody>
</table>
## Configure security and compliance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the access list.</td>
<td>To connect to the Atlas cluster, you must add an entry to the project's access list. Atlas uses Transport Layer Security (TLS) / Secure Sockets Layer (SSL) to encrypt the connections to the virtual private cloud (VPC) for your database. To set up the access list for the project and for more information about the stories in this epic, see the links in the &quot;Related resources&quot; section.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Authenticate and authorize users.</td>
<td>You must create and authenticate the database users who will access the MongoDB Atlas clusters. To access clusters in a project, users must belong to that project, and they can belong to multiple projects.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Create custom roles.</td>
<td>(Optional) Atlas supports creating custom roles in cases where the built-in Atlas database user privileges don’t cover your desired set of privileges.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Set up VPC peering.</td>
<td>(Optional) Atlas supports VPC peering with other AWS, Azure, or Google Cloud Platform (GCP) VPCs.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Set up an AWS PrivateLink endpoint.</td>
<td>(Optional) You can set up private endpoints on AWS by using AWS PrivateLink.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Enable two-factor authentication.</td>
<td>(Optional) Atlas supports two-factor authentication (2FA) to help users control access to their Atlas accounts.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Set up user authentication and authorization with LDAP.</td>
<td>(Optional) Atlas supports performing user authentication and authorization with Lightweight Directory Access Protocol (LDAP).</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Set up unified AWS access.</td>
<td>(Optional) Some Atlas features, including Atlas Data Lake and encryption at rest using customer key management, use AWS Identity and Access Management (IAM), and AWS Single Sign-On (SSO).</td>
<td>MongoDB DBA</td>
</tr>
</tbody>
</table>
## Migrate from MongoDB to MongoDB Atlas on AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Management (AWS IAM) roles for authentication.</td>
<td></td>
</tr>
<tr>
<td>Set up encryption at rest using AWS KMS.</td>
<td>(Optional) Atlas supports using AWS Key Management System (AWS KMS) to encrypt storage engines and cloud provider backups.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Set up client-side field-level encryption.</td>
<td>(Optional) Atlas supports client-side field level encryption, including automatic encryption of fields.</td>
<td>MongoDB DBA</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add the Atlas Live Migration Service to the access list in your AWS source cluster.</td>
<td>This helps prepare the source environment to connect to the target Atlas cluster.</td>
<td>MongoDB DBA</td>
</tr>
<tr>
<td>Validate your AWS credentials with Atlas Live Migration Service.</td>
<td>Choose “Start migration.” When the “Prepare to Cutover” button turns green, perform the cutover. Review Atlas cluster performance metrics.</td>
<td>MongoDB DBA</td>
</tr>
</tbody>
</table>

### Configure operational integration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to the MongoDB Atlas cluster.</td>
<td>Application developer</td>
<td></td>
</tr>
<tr>
<td>Interact with cluster data.</td>
<td>Application developer</td>
<td></td>
</tr>
<tr>
<td>Monitor your clusters.</td>
<td>MongoDB DBA</td>
<td></td>
</tr>
<tr>
<td>Back up and restore cluster data.</td>
<td>MongoDB DBA</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

**Migration guide**
- Migrating from MongoDB to MongoDB Atlas on the AWS Cloud
Discovery and assessment

- Memory
- Sizing example with Atlas sample data sets
- Sizing example for mobile applications
- Network Traffic
- Cluster Auto-Scaling
- Atlas sizing template

Configuring security and compliance

- Configure IP Access List Entries
- Configure database users
- Atlas User Access
- Configure Custom Roles
- Database User Privileges
- Set up a Network Peering Connection
- Set up a Private Endpoint
- Two Factor Authentication
- Set up User Authentication and Authorization with LDAP
- Atlas Data Lake
- Encryption at Rest using Customer Key Management
- Using IAM roles
- Client-Side Field Level Encryption
- Automatic Client-Side Field Level Encryption
- MongoDB Atlas Security
- MongoDB Trust Center
- Security Features and Setup

Setting up a new MongoDB Atlas environment on AWS

- Cloud Providers and Regions
- Global Clusters
- Cluster Tier
- Additional Cluster Settings
- Get Started with Atlas
- Atlas User Access
- Clusters

Migrating data

- Monitor Your Cluster

Integrating operations

- Connect to a Cluster
- Perform CRUD Operations in Atlas
- Monitor Your Cluster
• Backup and Restore Cluster Data

**Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS**

*Created by Anya Epishcheva (AWS)*

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Source:</th>
<th>Target:</th>
<th>Created by:</th>
<th>Environment:</th>
<th>Technologies:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replatform</td>
<td>Containers</td>
<td>Apache Tomcat (TomEE) on Amazon ECS</td>
<td>AWS</td>
<td>PoC or pilot</td>
<td>Containers &amp; microservices; Migration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Workload:</th>
<th>AWS services:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>Amazon ECS</td>
</tr>
</tbody>
</table>

**Summary**

This pattern discusses the steps for migrating an on-premises Oracle Solaris SPARC system running Oracle WebLogic to a Docker container-based installation running Apache TomEE (Apache Tomcat with added container support) with Amazon Elastic Container Service (Amazon ECS).

For information about migrating databases that are associated with the applications you are migrating from Oracle WebLogic to Tomcat, see the database migration patterns in this catalog.

**Best practices**

Steps for migrating Java and Java Enterprise Edition (Java EE) web applications vary, depending on the number of container-specific resources used by the application. Spring-based applications are typically easier to migrate, because they have a small number of dependencies on the deployment container. In contrast, Java EE applications that use Enterprise JavaBeans (EJBs) and managed container resources such as thread pools, Java Authentication and Authorization Service (JAAS), and Container-Managed Persistence (CMP) require more effort.

Applications developed for Oracle Application Server frequently use the Oracle Identity Management suite. Customers migrating to open-source application servers frequently choose to re-implement identity and access management using SAML-based federation. Others use Oracle HTTP Server Webgate for cases when migrating from the Oracle Identity Management suite isn't an option.

Java and Java EE web applications are great candidates for deployment on AWS services that are Docker-based, such as AWS Fargate and Amazon ECS. Customers frequently choose a Docker image with the latest version of the target application server (such as TomEE) and the Java Development Kit (JDK) pre-installed. They install their applications on top of the base Docker image, publish it in their Amazon Elastic Container Registry (Amazon ECR) registry, and use it for scalable deployment of their applications on AWS Fargate or Amazon ECS.

Ideally, application deployment is elastic; that is, the number of application instances scales in or out, depending on traffic or workload. This means that application instances need to come online or be terminated to adjust capacity for demand.

When moving a Java application to AWS, consider making it stateless. This is a key architectural principle of the AWS Well-Architected Framework that will enable horizontal scaling using containerization. For example, most Java-based web applications store user-session information locally. To survive application
instance termination due to automatic scaling in Amazon Elastic Compute Cloud (Amazon EC2) or for other reasons, user-session information should be stored globally so that web application users can continue to work seamlessly and transparently without reconnecting or relogging into a web application. There are several architectural options for this approach, including Amazon ElastiCache for Redis, or storing session state in a global database. Application servers such as TomEE have plugins, which enable session storage and management via Redis, databases, and other global data stores.

Use a common, centralized logging and debugging tool that is easily integrated with Amazon CloudWatch and AWS X-Ray. Migration provides an opportunity to improve application lifecycle capabilities. For example, you might want to automate the build process so that changes are easily made using a continuous integration and continuous delivery (CI/CD) pipeline. This may require changes to the application so that it can be deployed without any downtime.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- Source Java code and JDK
- Source application built with Oracle WebLogic
- Defined solution for identity and access management (SAML or Oracle Webgate)
- Defined solution for application session management (moving like-for-like or with Amazon ElastiCache, or making the application stateless if needed)
- Understanding if the team needs to refactor J2EE-specific libraries for portability to Apache TomEE (see Java EE 7 Implementation Status on the Apache website)
- Hardened TomEE image based on your security requirements
- Container image with pre-installed target TomEE
- Application remediation agreed and implemented if needed (for example, logging debug build, authentication)

**Product versions**

- Oracle WebLogic OC4J, 9i, 10g
- Tomcat 7 (with Java 1.6 or later)

**Architecture**

**Source technology stack**

- Web application built using Oracle WebLogic
- Web application using Oracle Webgate or SAML authentication
- Web applications connected to Oracle Database version 10g and later

**Target technology stack**

- TomEE (Apache Tomcat with added container support) running on Amazon ECS (see also Deploying Java Web Applications and Java Microservices on Amazon ECS)
- Amazon Relational Database Service (Amazon RDS) for Oracle; for Oracle versions supported by Amazon RDS, see Amazon RDS for Oracle

**Target architecture**
Tools

To operate on TomEE, a Java application must be rebuilt into a .war file. In some cases, application changes may be required to operate the application on TomEE; you should check to make sure that the necessary configuration options and environment properties are defined correctly.

Also, Java Naming and Directory Interface (JNDI) lookups and JavaServer Pages (JSP) namespaces should be defined correctly. Consider checking file names used by the application to avoid naming collisions with built-in T libraries. For example, persistence.xml is a file name used by the Apache OpenJPA framework (which is bundled with OpenEJB in TomEE) for configuration purposes. The persistence.xml file in PUI contains Spring framework bean declarations.

TomEE version 7.0.3 and later (Tomcat 8.5.7 and later) returns an HTTP 400 response (bad request) for raw (unencoded) URLs with special characters. The server response appears as a blank page to the end-user. Earlier versions of TomEE and Tomcat allowed the use of certain unencoded special characters in URLs; however, it’s considered unsafe, as stated on the CVE-2016-6816 website. To resolve the URL encoding issue, the URLs passed to the browser directly via JavaScript must be encoded with the encodeURI() method instead of being used as raw strings.

After you deploy the .war file in TomEE, monitor start log to Linux cat for any missing shared libraries and Oracle-specific extensions to add missing components from Tomcat libraries.

General procedure

- Configure the application on TomEE.
• Identify and reconfigure application server-specific configuration files and resources from source to target format.
• Identify and reconfigure JNDI resources.
• Adjust EJB namespace and lookups to the format required by the target application server (if applicable).
• Reconfigure JAAS application container-specific security roles and principle mappings (if applicable).
• Package the application and shared libraries into a .war file.
• Deploy the .war file in TomEE by using the Docker container provided.
• Monitor start log to identify any missing shared library and deployment descriptor extensions. If any are found, go back to the first task.
• Test the installed application against the restored Amazon RDS database.
• Launch the complete architecture with a load balancer and Amazon ECS cluster by following the instructions in Deploy Docker Containers.
• Update the URLs to point to the load balancer.
• Update the configuration management database (CMDB).

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform application discovery</td>
<td>(current state footprint and performance baseline).</td>
<td>BA, Migration Lead</td>
</tr>
<tr>
<td>Validate source and target database</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Validate the source and target</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Identify hardware and storage</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify network access security</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify application migration</td>
<td></td>
<td>DBA, Migration Lead</td>
</tr>
<tr>
<td>Complete the migration design and</td>
<td></td>
<td>Build Lead, Migration Lead</td>
</tr>
<tr>
<td>Complete the application migration</td>
<td></td>
<td>Build Lead, Cutover Lead, Testing</td>
</tr>
<tr>
<td>Plan the migration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Configure the infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon RDS DB instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Configure Amazon ECS deployment.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Package your application as a Docker image.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Push the image to the Amazon ECR registry (or skip this step and push it to the Amazon ECS cluster).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure the task definition for the application and Amazon ECS service options.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure your cluster, review security settings, and set AWS Identity and Access Management (IAM) roles.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Launch your setup and run tests according to your application migration runbook.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

## Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get your security assurance team's permission to move production data to AWS.</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Create and obtain access to endpoints to fetch database backup files.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Use the native database engine or third-party tools to migrate database objects and data.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Run necessary tests from the application migration runbook to confirm successful data migration.</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
</tbody>
</table>
## Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a change request (CR) for migration.</td>
<td></td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Obtain CR approval for migration.</td>
<td></td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Follow the application migration strategy from the application migration runbook.</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Upgrade the application (if required).</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Complete functional, non-functional, data validation, SLA, and performance tests.</td>
<td></td>
<td>Testing Lead, App Owner, App Users</td>
</tr>
</tbody>
</table>

## Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obtain signoff from the application or business owner.</td>
<td></td>
<td>Cutover Lead</td>
</tr>
<tr>
<td>Run a table topic exercise to walk through all the steps of the cutover runbook.</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
<tr>
<td>Switch application clients to the new infrastructure.</td>
<td></td>
<td>DBA, Migration Engineer, App owner</td>
</tr>
</tbody>
</table>

## Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down temporary AWS resources.</td>
<td></td>
<td>DBA, Migration Engineer, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>Migration Lead</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>Migration Lead</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>Migration Lead, App Owner</td>
</tr>
</tbody>
</table>

## Related resources

References
Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS

Created by Chethan Gangadharaih (AWS)

<table>
<thead>
<tr>
<th>R Type: Replatform</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Oracle</td>
<td>AWS services: Amazon EC2; Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes the steps for migrating an Oracle database on Amazon Elastic Compute Cloud (Amazon EC2) to Amazon Relational Database Service (Amazon RDS) for Oracle by using AWS Database Migration Service (AWS DMS). The pattern also uses Oracle SQL Developer or SQL *Plus to connect to your Oracle DB instance, and includes an AWS CloudFormation template that automates some of the tasks.

Migrating to Amazon RDS for Oracle enables you to focus on your business and applications while Amazon RDS takes care of database administration tasks such as provisioning databases, backup and recovery, security patches, version upgrades, and storage management.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Machine Image (AMI) for Oracle Database on Amazon EC2
Product versions

- AWS DMS supports Oracle versions 11g (version 11.2.0.3.v1 and later), 12c, and 18c for Amazon RDS instance databases for the Enterprise, Standard, Standard One, and Standard Two editions. For the latest information about supported versions, see Using an Oracle Database as a Target for AWS DMS in the AWS documentation. (The attached AWS CloudFormation templates use Oracle version 12c as the source database.)
- Oracle SQL Developer 4.0.3

Architecture

Source architecture

- Oracle Database on Amazon EC2

Target architecture

- Amazon RDS for Oracle

Migration architecture

Tools

- AWS DMS – AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. It supports both homogeneous and heterogeneous migrations. For information about the Oracle database versions and editions that are supported, see Using an Oracle Database as a Source for AWS DMS and Using an Oracle Database as a Target for AWS DMS in the AWS documentation.
- Oracle SQL Developer or SQL *Plus – These tools enable you to connect to the Amazon RDS for Oracle DB instance.
### Epics

**Set up your target database**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon RDS for Oracle DB instance.</td>
<td>Sign in to the AWS Management Console and open the Amazon RDS console at <a href="https://console.aws.amazon.com/rds/">https://console.aws.amazon.com/rds/</a>. Create an Oracle DB instance by selecting the appropriate engine, template, database credentials setting, instance type, storage, Multi-AZ settings, virtual private cloud (VPC) and configuration, login credentials, and additional settings for the Oracle database. For instructions, view the links in the &quot;Related resources&quot; section. Or use the AWS CloudFormation template (Create_RDS.yaml) in the attachment to create the Amazon RDS for Oracle DB instance.</td>
<td>Developer</td>
</tr>
<tr>
<td>Connect to Amazon RDS and grant privileges to the Oracle user.</td>
<td>Modify the security group to open the appropriate ports to connect from the local machine and the AWS DMS replication instance. When you configure connectivity, make sure that the &quot;Publicly accessible&quot; option is selected so you can connect to the database from outside the VPC. Connect to Amazon RDS with Oracle SQL Developer or SQL *Plus by using the login credentials, create an AWS DMS user, and provide the required privileges to the AWS DMS user to modify the database.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

**Configure the security group of the source EC2 instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check if the Oracle database is up and running.</td>
<td>Use Secure Shell (SSH) to connect to the EC2 instance, and try connecting to the Oracle database by using SQL *Plus.</td>
<td>Developer</td>
</tr>
<tr>
<td>Modify the security group.</td>
<td>Modify the security group of the EC2 instance to open</td>
<td>Developer</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Migrate from Oracle on Amazon EC2 to Amazon RDS for Oracle**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>appropriate ports, so you can connect from your local machine and the AWS DMS replication instance.</td>
<td></td>
</tr>
</tbody>
</table>

### Set up AWS DMS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS DMS replication instance.</td>
<td>In AWS DMS, create a replication instance in the same VPC as your Amazon RDS for Oracle DB instance. Specify the name and description for the replication instance, choose the instance class and replication engine version (use the default), choose the VPC in which you created the Amazon RDS DB instance, set Multi-AZ settings if required, allocate storage, specify the Availability Zone, and configure additional settings. Alternatively, you can use the AWS CloudFormation template (DMS.yaml) in the attachment to implement this step.</td>
<td>DBA</td>
</tr>
<tr>
<td>Connect to the source and target database endpoints.</td>
<td>Create the source and target database endpoints by specifying the endpoint identifier, engine, server, port, login credentials, and extra connection attributes. For the source server, use the public DNS of the EC2 instance that's hosting the Oracle database. For the target server, use the endpoint of Amazon RDS for Oracle. Run a test to verify that the source and target connections are working. Alternatively, you can use the AWS CloudFormation template (DMS.yaml) in the attachment to implement this step.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create an AWS DMS task.</td>
<td>Create an AWS DMS task to migrate data from the source endpoint to the target endpoint, to set up replication between the source and destination endpoint, or both. When creating the AWS DMS</td>
<td>DBA</td>
</tr>
</tbody>
</table>
**Related resources**

- Creating an Amazon RDS DB instance
- Connecting to a DB Instance Running the Oracle Database Engine
- AWS DMS documentation
- AWS DMS Step-by-Step Walkthroughs
- Migrating Oracle Databases to the AWS Cloud

**Attachments**

attachment.zip

**Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash**

_Created by Aditya Goteti (AWS)_

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon ES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R Type:</strong></td>
<td>Replatform</td>
<td><strong>Workload:</strong></td>
<td>Oracle</td>
<td><strong>Technologies:</strong></td>
<td>Migration; Databases</td>
</tr>
<tr>
<td><strong>AWS services:</strong></td>
<td>Amazon OpenSearch Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes how to move data from an on-premises Oracle database to Amazon OpenSearch Service (successor to Amazon Elasticsearch Service) using Logstash. It includes architectural considerations and some required skill sets and recommendations. The data can be from a single table or from multiple tables in which a full-text search will need to be performed.

Amazon OpenSearch Service can be configured within a virtual private cloud (VPC), or it can be placed publicly with IP-based restrictions. This pattern describes a scenario where Amazon OpenSearch...
Service is configured within a VPC. Logstash is used to collect the data from the Oracle database, parse it to JSON format, and then feed the data into Amazon OpenSearch Service.

Prerequisites and limitations

**Prerequisites**

- An active AWS account
- Java 8 (required by Logstash 6.4.3)
- Connectivity between the on-premises database servers and Amazon Elastic Compute Cloud (Amazon EC2) instances in a VPC, established using AWS Virtual Private Network (AWS VPN)
- A query to retrieve the required data to be pushed to Amazon OpenSearch Service from the database
- Oracle Java Database Connectivity (JDBC) drivers

**Limitations**

- Logstash cannot identify records that are hard-deleted from the database

**Product versions**

- Oracle Database 12c
- Amazon OpenSearch Service 6.3
- Logstash 6.4.3

Architecture

**Source technology stack**

- On-premises Oracle database
- On-premises AWS VPN

**Target technology stack**

- VPC
- EC2 instance
- Amazon OpenSearch Service
- Logstash
- NAT Gateway (for operating system updates on EC2 instances and to install Java 8, Logstash, and plugins)

**Data migration architecture**
### Tools

- Logstash 6.4.3
- JDBC input plugin ([download and more information](#))
- Logstash output plugin ([logstash-output-amazon_es](#))
- Oracle JDBC drivers

### Epics

#### Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the size of the source data.</td>
<td>The size of the source data is one of the parameters that you use to determine the number of shards to be configured in an index.</td>
<td>DBA, Database Developer</td>
</tr>
<tr>
<td>Analyze the data types of each column and corresponding data.</td>
<td>Amazon OpenSearch Service dynamically maps the data type when a previously unseen field is found in the document. If there are any specific data types or formats (e.g., date fields) that need to be explicitly declared, identify the fields and define the mapping for those fields during index creation.</td>
<td>App owner, Developer, Database Developer</td>
</tr>
<tr>
<td>Determine if there are any columns with primary or unique keys.</td>
<td>To avoid duplication of records in Amazon OpenSearch Service during updates or updates.</td>
<td>App owner, Developer</td>
</tr>
</tbody>
</table>
### Migrate from Oracle to Amazon OpenSearch Service with Logstash

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
<tr>
<td></td>
<td>inserts, you need to configure the document_id setting in the output section of the amazon_es plugin (e.g., document_id =&gt; &quot;%{customer_id}&quot; where customer_id is a primary key).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analyze the number and frequency of new records added; check how frequently the records are deleted.</td>
<td>This task is required to understand the growth rate of source data. If data is intensively read and insertions are rare, you can have a single index. If new records are inserted frequently and there are no deletions, the shard size can easily exceed the recommended maximum size of 50 GB. In this case, you can dynamically create an index by configuring index patterns in Logstash and in the code where you can access it by using an alias.</td>
</tr>
<tr>
<td></td>
<td>Determine how many replicas are required.</td>
<td>App owner, Developer</td>
</tr>
<tr>
<td></td>
<td>Determine the number of shards to be configured on the index.</td>
<td>App owner, Developer</td>
</tr>
<tr>
<td></td>
<td>Identify the instance types for dedicated master nodes, data nodes, and the EC2 instance.</td>
<td>For more information, see the References and Help section.</td>
</tr>
<tr>
<td></td>
<td>Determine the number of dedicated master nodes and data nodes required.</td>
<td>For more information, see the References and Help section.</td>
</tr>
</tbody>
</table>

### Migrate data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Launch an EC2 instance within the VPC to which AWS VPN is connected.</td>
<td>Amazon VPC constructs AWS VPN</td>
</tr>
<tr>
<td></td>
<td>Install Logstash on the EC2 instance.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Install the required Logstash plugins jdbc-input and logstash-output-amazon_es.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Configure Logstash.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

Create the Logstash keystore to store sensitive information such as AWS Secrets Manager keys and database credentials, and
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the dead letter queue and persistent queue.</td>
<td>By default, when Logstash encounters an event that it cannot process because the data contains a mapping error or some other issue, the Logstash pipeline either hangs or drops the unsuccessful event. To protect against data loss in this situation, you can configure Logstash to write unsuccessful events to a dead letter queue instead of dropping them. To protect against data loss during abnormal termination, Logstash has a persistent queue feature that will store the message queue on disk. Persistent queues provide the data durability in Logstash.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the Amazon OpenSearch Service domain with an access policy that doesn't require signing requests with AWS Identity and Access Management (IAM) credentials.</td>
<td>The Amazon OpenSearch Service domain must be created within the same VPC. You should also select the instance types and set the number of dedicated and master nodes based on your analysis.                                                                 │ Developer</td>
<td></td>
</tr>
<tr>
<td>Configure the required Amazon OpenSearch Service logs.</td>
<td>For more information, see <a href="https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/cloudwatch-alarms.html">https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/cloudwatch-alarms.html</a>.</td>
<td></td>
</tr>
<tr>
<td>Create the index.</td>
<td></td>
<td>Developer</td>
</tr>
<tr>
<td>Start Logstash.</td>
<td>Run Logstash as a background service. Logstash runs the configured SQL query, pulls the data, converts it to JSON format, and feeds it to Amazon OpenSearch Service. For the initial load, do not configure the scheduler in the Logstash configuration file.</td>
<td>Developer</td>
</tr>
</tbody>
</table>
Migrate from Oracle to Amazon RDS for Oracle

**Task**

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check the number of documents on the index and whether all documents are present in the source database. During initial load, they are added to the index and used to stop Logstash. Change the Logstash configuration to add a scheduler that runs at a fixed interval depending on client requirements, and restart Logstash. Logstash will pick only the records that were updated or added after the last run, and the last run timestamp is stored in the file configured with the property <code>last_run_metadata_path =&gt; “/usr/share/logstash/.logstashjdbc_last_run”</code> in the Logstash configuration file.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

**Related resources**

- Recommended CloudWatch Alarms
- Dedicated Amazon OpenSearch Service Master Nodes
- Sizing Amazon OpenSearch Service Domains
- Logstash documentation
- JDBC input plugin
- Logstash output plugin
- Amazon OpenSearch Service website

**Migrate an on-premises Oracle database to Amazon RDS for Oracle**

*Created by Baji Shaik (AWS)*

<table>
<thead>
<tr>
<th>R Type: Replatform</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Oracle</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern describes the steps for migrating on-premises Oracle databases to Amazon Relational Database Service (Amazon RDS) for Oracle. It describes two options for migration: using AWS Database Migration Service (AWS DMS) or using native Oracle tools such as Oracle Data Pump import/export and Oracle GoldenGate.
Prerequisites and limitations

Prerequisites

- An active AWS account
- An Oracle database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- Oracle versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions and editions, see Oracle on Amazon RDS in the AWS documentation. If you’re using AWS DMS, see also Using an Oracle Database as a Source for AWS DMS for Oracle versions supported by AWS DMS.

Architecture

Source technology stack

- On-premises Oracle databases

Target technology stack

- Amazon RDS for Oracle

Source and target architecture

Using AWS DMS:
Using native Oracle tools:
Tools

- **AWS DMS** - *AWS Database Migration Service* (AWS DMS) helps you migrate relational databases, data warehouses, NoSQL databases, and other types of data stores. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups. For information about the Oracle database versions and editions that are supported, see Using an Oracle Database as a Source for AWS DMS.

- **Native Oracle tools** - Oracle Data Pump import/export, Oracle GoldenGate

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the versions of the source and target databases.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
## Migrate from Oracle to Amazon RDS for Oracle

### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

### Configure the infrastructure

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an Amazon RDS DB instance.</td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate data - option 1

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native Oracle tools or third-party tools to migrate database objects and data.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

### Migrate data - option 2

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data with AWS DMS.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics for time to migrate, percentage of manual vs. tool, cost savings, etc.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Related resources

References

- Strategies for Migrating Oracle Databases to AWS
- AWS DMS website
- Amazon RDS Pricing

Tutorials and videos

- Getting Started with AWS Database Migration Service
- Getting Started with Amazon RDS
- Technical introduction to AWS DMS (video)
- Technical introduction to Amazon RDS (video)

Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump

Created by Mohan Annam (AWS)
Summary

This pattern describes how to migrate an Oracle database from an on-premises data center to an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance by using Oracle Data Pump.

The pattern involves creating a data dump file from the source database, storing the file in an Amazon Simple Storage Service (Amazon S3) bucket, and then restoring the data to an Amazon RDS for Oracle DB instance. This pattern is useful when you encounter limitations using AWS Database Migration Service (AWS DMS) for the migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The required permissions to create roles in AWS Identity and Access Management (IAM) and for an Amazon S3 multipart upload
- The required permissions to export data from the source database

Product versions

- Oracle Data Pump is available only for Oracle Database 10g Release 1 (10.1) and later versions.

Architecture

Source technology stack

- On-premises Oracle databases

Target technology stack

- Amazon RDS for Oracle
- SQL client (Oracle SQL Developer)
- An S3 bucket

Source and target architecture
Tools

- **Oracle Data Pump** – Oracle Data Pump is used to export the data dump (.dmp) file to the Oracle server, and to import it into Amazon RDS for Oracle. For more information, see [Importing Data with Oracle Data Pump and an Amazon S3 Bucket](#) in the Amazon RDS documentation.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a durable storage service that provides faster import to Amazon RDS. This pattern uses an S3 bucket to store the data dump file.

- **AWS IAM** – AWS Identity and Access Management (IAM) is used to create the roles and policies necessary for migrating data from Amazon S3 to Amazon RDS for Oracle.

- **Oracle SQL Developer** – SQL Developer interacts with both the on-premises Oracle database and Amazon RDS for Oracle, to run SQL commands required for exporting and importing data.

Epics

**Set up the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the S3 bucket.</td>
<td>Store the database dump file and provide necessary access permissions for dump file storage.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Create the IAM role and assign policies.</td>
<td>Follow the instructions in the Prerequisites for Amazon RDS for Oracle integration with Amazon S3 section of the Amazon RDS documentation.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Create the data dump file from the source Oracle database.</td>
<td>See &quot;Creating a data dump file&quot; in the Additional information section.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Upload the dump file to the S3 bucket.</td>
<td>See &quot;Uploading the data dump file&quot; in the Additional information section.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Create the target Amazon RDS for Oracle DB instance and assign the Amazon S3 integration role.</td>
<td>Follow the instructions at Creating an Oracle DB instance and connecting to a database on an Oracle DB instance in the Amazon RDS documentation.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Import the dump file into the target database.</td>
<td>Follow the two steps described in &quot;Uploading the data dump file to Amazon RDS&quot; in the Additional information section.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Restore the imported dump file to the target database.</td>
<td>See &quot;Restoring the schema and data to Amazon RDS&quot; in the Additional information section.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

Related resources

- Prerequisites for Amazon RDS Oracle Integration with Amazon S3
- Creating an RDS DB Instance
- Importing Data with Oracle Data Pump and an Amazon S3 Bucket
- Amazon S3 documentation
- Multipart Upload Overview
- IAM documentation
- Amazon RDS documentation
- Oracle Data Pump documentation
- Getting Started with Oracle SQL Developer 4.0

Additional information

Creating a data dump file

To create a dump file named sample.dmp in the DATA_PUMP_DIR directory, use this script:

```sql
DECLARE
  hdnl NUMBER;
BEGIN
  hdnl := DBMS_DATAPUMP.OPEN( operation => 'EXPORT', job_mode => 'SCHEMA', job_name=>null);
  DBMS_DATAPUMP.ADD_FILE(handle => hdnl, filename => 'sample.dmp', directory => 'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
  DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'exp.log', directory => 'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(hdnl,'SCHEMA_EXPR','IN (''ADMIN'')');
  DBMS_DATAPUMP.METADATA_FILTER(hdnl,'EXCLUDE_NAME_EXPR',q'
          [IN (SELECT NAME FROM sys.OBJ$ WHERE TYPE# IN (66,67,74,79,59,62,46) and OWNER# IN (SELECT USER# from sys.user$ where NAME in ('RDSADMIN','SYS','SYSTEM','RDS_DATAGUARD','RDSSEC')))]
         ', 'PROCOBJ');
  DBMS_DATAPUMP.START_JOB(hdnl);
END;
```

Uploading the data dump file

Use the following command-line interface (CLI) command from the folder where the dump file was created. This will upload the dump file to Amazon S3 by using multipart upload:
Uploading the data dump file to Amazon RDS

To import the dump file into the target database:

1. Connect to the target Amazon RDS for Oracle database from Oracle SQL Developer, and then run the following SQL command to create the user to import the database:

```sql
create user <USER_NAME> identified by <PASSWORD>;
grant create session, resource to <USER_NAME>;
alter user <USER_NAME> quota 100M on users;
```

2. Run the following SQL command to import the dump file from the S3 bucket to the Amazon RDS for Oracle database:

```sql
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
    p_bucket_name => 'my-s3-integration1',
    p_directory_name => 'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

Restoring the schema and data to Amazon RDS

Run the following SQL command from Oracle SQL Developer to restore the schema and data from the imported dump file:

```sql
DECLARE
   hdnl NUMBER;
BEGIN
   hdnl := DBMS_DATAPUMP.OPEN( operation => 'IMPORT', job_mode => 'SCHEMA', job_name=>null);
   DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'sample.dmp', directory =>
                    'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
   DBMS_DATAPUMP.METADATA_FILTER(hdnl,'SCHEMA_EXPR','IN (''ADMIN'')');
   DBMS_DATAPUMP.START_JOB(hdnl);
END;
/
```

Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical

Created by Rajesh Madiwale (AWS)

<table>
<thead>
<tr>
<th>R Type: Replatform</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by: AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload: Open-source</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern outlines steps for migrating a PostgreSQL database (version 9.5 and later) from Amazon Elastic Compute Cloud (Amazon EC2) to Amazon Relational Database Service (Amazon RDS) for
PostgreSQL by using the PostgreSQL pglogical extension. Amazon RDS now supports the pglogical extension for PostgreSQL version 9.6.

Prerequisites and limitations

Prerequisites

- Choose the right type of Amazon RDS instance. For more information, see Amazon RDS Instance Types.
- Ensure that the source and target versions of PostgreSQL are the same.
- Install and integrate the pglogical extension with PostgreSQL on Amazon EC2.

Product versions

- PostgreSQL version 9.6 and later on Amazon RDS. For the latest list of PostgreSQL versions and features supported on Amazon RDS, see PostgreSQL on Amazon RDS in the AWS documentation. This pattern was tested by migrating PostgreSQL 9.5 to Amazon RDS version 10, but it also applies to PostgreSQL version 9.6 and later.

Architecture

Data migration architecture

Tools

- pglogical extension
Epics

Migrate using the pglogical extension

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install a PostgreSQL database on the Amazon RDS instance.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Obtain a schema dump from the source PostgreSQL database and restore it into the target PostgreSQL database.</td>
<td>The pg_dump utility creates a schema and data dump of a PostgreSQL database. The dump script generated by pg_dump loads data into a database with the same name and recreates the tables, indexes, and foreign keys. Use the pg_restore utility to restore the database in the target location.</td>
<td>DBA</td>
</tr>
<tr>
<td>Set the rds.logical_replication static parameter to 1 in the Amazon RDS DB parameter group.</td>
<td>Enabling the rds.logical_replication parameter affects the DB cluster's performance.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create the pglogical extension on the source and target PostgreSQL databases.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a publisher on the source PostgreSQL database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a replication set on the source PostgreSQL database and add tables and sequences to the replication set.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a subscriber on the target PostgreSQL database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create a subscription on the target PostgreSQL database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Validate your data.</td>
<td>Check the source and target databases to confirm that data is being replicated successfully.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Related resources

- Amazon RDS
- Logical Replication for PostgreSQL on Amazon RDS
- pglogical (GitHub repository)
- Limitations of pglogical
Migrate an on-premises PostgreSQL database to Amazon RDS for PostgreSQL

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon RDS for PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
<td>AWS services:</td>
<td>Amazon RDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern walks you through the steps for migrating an on-premises PostgreSQL source database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL. The pattern includes two migration options: using AWS Data Migration Service (AWS DMS) or using native PostgreSQL tools such as `pg_dump`, `pg_restore`, and `psql`, or third-party tools.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A PostgreSQL source database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- PostgreSQL versions 9.3.12–12. For the latest list of supported versions and features, see [PostgreSQL on Amazon RDS](https://aws.amazon.com/rds/postgresql) in the AWS documentation. If you’re using AWS DMS as your migration tool, see also [Using a PostgreSQL Database as a Source for AWS DMS](https://docs.aws.amazon.com/dms/latest/ug/rds-postgresql-created-migration-pattern.html) for supported versions.

Architecture

Source technology stack

- An on-premises PostgreSQL database

Target technology stack

- An Amazon RDS for PostgreSQL database (same PostgreSQL version as the source database)
Target architecture

Data migration architecture

Using AWS DMS:
Using native PostgreSQL or third-party tools:
Tools

- **AWS DMS** – AWS Data Migration Service (AWS DMS) supports several different source and target databases. See Using a PostgreSQL Database as a Source for AWS DMS in the AWS DMS documentation for a list of supported PostgreSQL source and target database versions and editions. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

- **Native PostgreSQL tools** – These include `pg_dump`, `pg_restore`, and `psql`.

**Epics**

Analyze the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target</td>
<td>Validate the source and target database version</td>
<td>DBA</td>
</tr>
<tr>
<td>database version and engine.</td>
<td>and engine.</td>
<td></td>
</tr>
<tr>
<td>Identify hardware requirements</td>
<td>Identify hardware requirements for the target</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>for the target server instance.</td>
<td>server instance.</td>
<td></td>
</tr>
<tr>
<td>Identify storage requirements:</td>
<td>Identify storage requirements: storage type and</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>storage type and capacity.</td>
<td>capacity.</td>
<td></td>
</tr>
</tbody>
</table>
## Migrate from PostgreSQL to Amazon RDS for PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose proper instance type capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Identify network access security requirements for source and target databases.</td>
<td>DBA, SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
<td></td>
</tr>
</tbody>
</table>

### Configure infrastructure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Create security groups.</td>
<td>SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Configure and start an Amazon RDS DB instance.</td>
<td>SysAdmin</td>
<td></td>
</tr>
</tbody>
</table>

### Migrate data – option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native PostgreSQL tools or third-party tools to migrate database objects and data.</td>
<td>We recommend that you use native PostgreSQL database migration tools under the following conditions: you have a homogeneous migration, where you are migrating from a database with the same database engine as the target database; you are migrating an entire database; and the native tools allow you to migrate your system with minimal downtime.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate data – option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate data with AWS DMS.</td>
<td>In most cases not listed under the first option, performing a database migration using AWS DMS is the best approach. AWS DMS can migrate databases without downtime and, for many database engines, continue</td>
<td>DBA</td>
</tr>
</tbody>
</table>
### Migrate from PostgreSQL to Amazon RDS for PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ongoing replication until you are ready to switch over to the target database. You can migrate to either the same database engine or a different database engine using AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.</td>
<td></td>
</tr>
</tbody>
</table>

#### Migrate applications

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Follow your chosen application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

#### Cut over to Amazon RDS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Switch over application clients to the new infrastructure.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

#### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shut down temporary AWS resources.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Review and validate project documents.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td></td>
<td>Gather metrics detailing time to migrate, percentage of manual versus tool operations, cost savings, etc.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td></td>
<td>Close out the project and provide any feedback.</td>
<td>Migration Lead, App owner</td>
</tr>
</tbody>
</table>

### Related resources

**References**

- AWS Data Migration Service
- Amazon RDS Pricing
Migrate an on-premises PostgreSQL database to Aurora PostgreSQL

Created by Sergey Dmitriev (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon Aurora (PostgreSQL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td>Technologies:</td>
<td>Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
<td>AWS services:</td>
<td>Amazon Aurora</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes the migration steps from an on-premises PostgreSQL source database to an Amazon Aurora PostgreSQL-Compatible Edition database. The pattern includes two migration options: using AWS Data Migration Service (AWS DMS) or using native PostgreSQL tools (such as `pg_dump`, `pg_restore`, and `psql`) or third-party tools.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A PostgreSQL source database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- PostgreSQL 9.4 and later (for versions 9.x), 10.x, and 11.x, if you're using AWS DMS. For the latest list of supported versions, Using a PostgreSQL Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture

Source technology stack
AWS Prescriptive Guidance Patterns
Migrate from PostgreSQL to Aurora PostgreSQL

- On-premises PostgreSQL database

**Target technology stack**
- Aurora PostgreSQL-Compatible DB instance

**Target architecture**

![Architecture Diagram]

**Data migration architecture**

*Using AWS DMS:*
AWS Prescriptive Guidance Patterns
Migrate from PostgreSQL to Aurora PostgreSQL

Using native PostgreSQL tools:
Tools

- **AWS DMS** - AWS Data Migration Service (AWS DMS) supports different sources and target databases. For information about how to validate the PostgreSQL source and target database versions and editions supported for use with AWS DMS, see [Using a PostgreSQL Database as a Source for AWS DMS](#). We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

- **Native PostgreSQL tools** - These include `pg_dump`, `pg_restore`, and `psql`.

Epics

**Analyze the migration**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database versions.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server database.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the requirements for the storage type and capacity.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Choose the proper instance type, capacity, storage features, and network features.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for the source and target databases.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Configure the infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start the Amazon Aurora DB cluster.</td>
<td></td>
<td>SysAdmin</td>
</tr>
</tbody>
</table>
Migrate data - option 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the database objects and data by using native PostgreSQL or third-party tools.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate data - option 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate the data by using AWS DMS.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Cut over to the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch over the application clients to the new infrastructure.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

Related resources

References
Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux

Created by Tirumala Rama Chandra Murty Dasari (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon EC2 Linux with Microsoft SQL Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Microsoft</td>
<td>AWS services: Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to migrate from an on-premises Microsoft SQL Server database running on Microsoft Windows, to Microsoft SQL Server on an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance by using backup and restore utilities.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon EC2 Linux AMI (Amazon Machine Image) with Microsoft SQL Server
- AWS Direct Connect between on-premises Windows and Microsoft SQL Server on the Linux EC2 instance

Architecture

Source technology stack

- On-premises Microsoft SQL Server database

Target technology stack

- AWS Data Migration Service
- Amazon Virtual Private Cloud VPCs and Amazon Aurora
- Amazon Aurora pricing
- Using a PostgreSQL Database as a Source for AWS DMS

Tutorials and videos

- Getting Started with AWS Database Migration Service
- AWS Data Migration Service Step-by-Step Walkthroughs
- Amazon Aurora resources

Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux

Created by Tirumala Rama Chandra Murty Dasari (AWS)
• Linux EC2 instance with a Microsoft SQL Server database

Database migration architecture

Tools

• **WinSCP** - This tool enables Windows users to easily share files with Linux users.
• **Sqlcmd** - This command-line utility lets you submit T-SQL statements or batches to local and remote instances of SQL Server. The utility is extremely useful for repetitive database tasks such as batch processing or unit testing.
Epics

Prepare the EC2 Linux instance with SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select an AMI that provides the Linux operating system and includes Microsoft SQL Server.</td>
<td>Sysadmin</td>
</tr>
<tr>
<td></td>
<td>Configure the AMI to create an EC2 instance.</td>
<td>Sysadmin</td>
</tr>
<tr>
<td></td>
<td>Create inbound and outbound rules for security groups.</td>
<td>Sysadmin</td>
</tr>
<tr>
<td></td>
<td>Configure the Linux EC2 instance for a Microsoft SQL Server database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Create users and provide permissions as in the source database.</td>
<td>Appowner, DBA</td>
</tr>
<tr>
<td></td>
<td>Install SQL Server tools and the sqlcmd utility on the Linux EC2 instance.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Back up the database and move backup file to Linux EC2 instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Back up the on-premises SQL Server database.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Install WinSCP on Microsoft SQL Server.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Move the backup file to the Linux EC2 instance running Microsoft SQL Server.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Restore the database on Linux EC2 instance running SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Restore the database from the database backup file by using the sqlcmd utility.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Validate database objects and data.</td>
<td>Developer, Test engineer</td>
</tr>
</tbody>
</table>
Cut over from Windows SQL Server to Windows SQL Server on Linux EC2 instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Validate database objects and data.</td>
<td>Developer, Test engineer</td>
</tr>
<tr>
<td></td>
<td>Cut over from the on-premises Microsoft SQL Server database to the Linux EC2 instance running Microsoft SQL Server.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

Related resources
- How to configure SQL Server 2017 on Amazon Linux 2 and Ubuntu AMIs
- Installation of SQL tools on a Linux instance
- Backup and restoration from an on-premises Microsoft SQL Server database to Microsoft SQL Server on a Linux EC2 instance

Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server

Created by Mark Szalkiewicz (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for Microsoft SQL Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: PoC or pilot</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Microsoft</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides guidance for migrating from an on-premises Microsoft SQL Server database to Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server. It describes two options for migration: using AWS Data Migration Service (AWS DMS) or using native Microsoft SQL Server tools such as Copy Database Wizard.

Prerequisites and limitations

Prerequisites
- An active AWS account
- A source Microsoft SQL Server database in an on-premises data center

Limitations
• Database size limit: 16 TB

Product versions

• SQL Server 2012-2017, Enterprise, Standard, Workgroup, and Developer editions. For the latest list of supported versions and features, see Microsoft SQL Server on Amazon RDS in the AWS documentation. If you’re using AWS DMS, see also Using a Microsoft SQL Server Database as a Target for AWS DMS for SQL Server versions supported by AWS DMS.

Architecture

Source technology stack

• An on-premises Microsoft SQL Server database

Target technology stack

• An Amazon RDS for Microsoft SQL Server DB instance

Source and target architecture

Using AWS DMS:

Using native Microsoft SQL Server tools:
Tools

- **AWS DMS** - AWS Database Migration Service (AWS DMS) supports several types of source and target databases. For details, see AWS DMS Step-by-Step Walkthroughs. If AWS DMS doesn't support the source database, select another method for migrating the data.

- **Native Microsoft SQL Server tools** - Backup and restore; Copy Database Wizard: copy and attach database

**Epics**

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify the hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the storage requirements (storage type and capacity).</td>
<td></td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>
### Migrate from SQL Server to Amazon RDS for SQL Server

#### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the network access security requirements for source and target databases.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

#### Configure the infrastructure

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create security groups.</td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Configure and start an Amazon RDS DB instance.</td>
<td>DBA, SysAdmin</td>
</tr>
</tbody>
</table>

#### Migrate data - option 1

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native Microsoft SQL Server tools or third-party tools to migrate database objects and data.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Migrate data - option 2

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrate data with AWS DMS.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

#### Migrate the application

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>
**Cut over**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Switch the application clients over to the new infrastructure.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Close the project**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shut down the temporary AWS resources.</td>
<td>DBA, SysAdmin</td>
</tr>
<tr>
<td></td>
<td>Review and validate the project documents.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td></td>
<td>Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
<tr>
<td></td>
<td>Close out the project and provide feedback.</td>
<td>DBA, SysAdmin, App owner</td>
</tr>
</tbody>
</table>

**Related resources**

**References**

- Deploying Microsoft SQL Server on Amazon Web Services
- AWS DMS website
- Amazon RDS Pricing
- Microsoft Products on AWS
- Microsoft Licensing on AWS
- Microsoft SQL Server on AWS
- Using Windows Authentication with a Microsoft SQL Server DB Instance
- Amazon RDS Multi-AZ Deployments

**Tutorials and videos**

- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS DMS (video)
- Amazon RDS (video)

**Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using Amazon S3 and SSMS**
Summary

This pattern describes how to migrate an on-premises Microsoft SQL Server database to an Amazon Relational Database Service (Amazon RDS) for SQL Server DB instance. The migration process includes making a backup and restoring the backup in an Amazon Simple Storage Service (Amazon S3) bucket, and using SQL Server Management Studio (SSMS).

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) user policies to access S3 buckets and the Amazon RDS for SQL Server DB instance

Product versions

- SQL Server 2012-2017 (for the latest list of supported versions and features, see Microsoft SQL Server on Amazon RDS in the AWS documentation)

Architecture

Source technology stack

- An on-premises Microsoft SQL Server database

Target technology stack

- An Amazon RDS for SQL Server DB instance

Data migration architecture
Tools

- **SSMS** - Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

Epics

Create an Amazon RDS for Microsoft SQL Server DB instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select SQL Server as the database engine in Amazon RDS for SQL Server.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Choose the SQL Server Express Edition.</td>
<td>DBA</td>
</tr>
<tr>
<td></td>
<td>Specify database details.</td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

Create a backup file from the on-premises Microsoft SQL Server database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Connect to the on-premises SQL Server database through SSMS.</td>
<td>DBA</td>
</tr>
</tbody>
</table>
Upload the backup file to Amazon S3

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a bucket in Amazon S3.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Upload the backup file to the S3 bucket using AWS Direct Connect.</td>
<td></td>
<td>SysopsAdmin</td>
</tr>
</tbody>
</table>

Restore the database in Amazon RDS for SQL Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the Amazon RDS console at <a href="https://console.aws.amazon.com/rds/">https://console.aws.amazon.com/rds/</a>.*</td>
<td></td>
<td>SysopsAdmin</td>
</tr>
<tr>
<td>Choose Option groups in the navigation pane.</td>
<td></td>
<td>SysopsAdmin</td>
</tr>
<tr>
<td>Choose the Create group button.</td>
<td></td>
<td>SysopsAdmin</td>
</tr>
<tr>
<td>Add the SQLSERVER_BACKUP_RESTORE option to the option group.</td>
<td></td>
<td>SysopsAdmin</td>
</tr>
<tr>
<td>Add the option group to Amazon RDS for SQL Server.</td>
<td></td>
<td>SysopsAdmin</td>
</tr>
<tr>
<td>Connect to Amazon RDS for SQL Server through SSMS.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Call the msdb.dbo.rds_restore_database stored procedure to restore the database.</td>
<td></td>
<td>DBA</td>
</tr>
</tbody>
</table>

Validate the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the objects and data between the source database and Amazon RDS for SQL Server.</td>
<td></td>
<td>App owner, DBA</td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>After validation, redirect application traffic to the Amazon RDS for SQL Server DB instance.</td>
<td></td>
<td>App owner, DBA</td>
</tr>
</tbody>
</table>

Related resources

- Amazon S3 documentation
- Amazon RDS for SQL Server documentation
- Options for the Microsoft SQL Server Database Engine

Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers

*Created by Kevin Yung (AWS)*

<table>
<thead>
<tr>
<th>R Type:</th>
<th>Replatform</th>
<th>Source: Databases: Relational</th>
<th>Target: Amazon RDS for Microsoft SQL Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment: Production</td>
<td>Technologies: Databases; Migration</td>
</tr>
<tr>
<td>Workload:</td>
<td>Microsoft</td>
<td>AWS services: Amazon RDS</td>
<td></td>
</tr>
</tbody>
</table>

Summary

Linked servers enable Microsoft SQL Server to run SQL statements on other instances of database servers. This pattern describes how you can migrate your on-premises Microsoft SQL Server database to Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server to achieve lower cost and higher availability. Currently, Amazon RDS for Microsoft SQL Server doesn't support connections outside an Amazon Virtual Private Cloud (Amazon VPC) network.

You can use this pattern to achieve the following objectives:

- To migrate Microsoft SQL Server to Amazon RDS for Microsoft SQL Server without breaking linked server capabilities.
- To prioritize and migrate linked Microsoft SQL Server in different waves.

Prerequisites and limitations

**Prerequisites**

- Check whether Microsoft SQL Server on Amazon RDS supports the features you require.
• Make sure that you can use either Amazon RDS for Microsoft SQL Server with default collations or collations set over database levels.

Architecture

Source technology stack
• On-premises databases (Microsoft SQL Server)

Target technology stack
• Amazon RDS for SQL Server

Source state architecture
In the target state, you migrate Microsoft SQL Server to Amazon RDS for Microsoft SQL Server by using linked servers. This architecture uses a Network Load Balancer to proxy the traffic from Amazon RDS for Microsoft SQL Server to on-premises servers running Microsoft SQL Server. The following diagram shows the reverse proxy capability for the Network Load Balancer.
Tools

- AWS CloudFormation
- Network Load Balancer
- Amazon RDS for SQL Server in multiple Availability Zones (Multi-AZs)
- AWS Database Migration Service (AWS DMS)

Epics

Create a landing zone VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the CIDR allocation.</td>
<td>Create a virtual private cloud (VPC).</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create the VPC subnets.</td>
<td>Create the subnet access control lists (ACLs).</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create the subnet route tables.</td>
<td>Create a connection with AWS Direct Connect or AWS Virtual Private Network (VPN).</td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>

Migrate the database to Amazon RDS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon RDS for Microsoft SQL Server DB instance.</td>
<td>Create an AWS DMS replication instance.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Create source and target database endpoints in AWS DMS.</td>
<td>Create the migration task and set continuous replication to ON after a full load.</td>
<td>AWS SysAdmin</td>
</tr>
</tbody>
</table>
## Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request a firewall change to allow Amazon RDS for Microsoft SQL Server to access the on-premises SQL Server databases.</td>
<td>AWS SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Create a Network Load Balancer.</td>
<td>AWS SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Create a target group that targets the database servers in your data center</td>
<td>We recommend that you use hostnames in the target setup to incorporate data center (DC) failover events.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Run the SQL statement for linked server setup.</td>
<td>Run the SQL statements for adding a linked server by using the Microsoft SQL management tool against the Amazon RDS for Microsoft SQL Server DB instance. In the SQL statement, set @datasrc to use the Network Load Balancer hostname. Add linked server login credentials by using the Microsoft SQL management tool against the Amazon RDS for Microsoft SQL Server DB instance.</td>
<td>AWS SysAdmin</td>
</tr>
<tr>
<td>Test and validate the SQL Server functions.</td>
<td>AWS SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Create a cutover.</td>
<td>AWS SysAdmin</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

- Common Management Tasks for Microsoft SQL Server on Amazon RDS
- Collations and Character Sets for Microsoft SQL Server
- Network Load Balancer documentation
- Implement Linked Servers with Amazon RDS for Microsoft SQL Server (blog post)

### Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT

*Created by Mark Szalkiewicz (AWS)*

| R Type: Replatform | Source: Databases: Relational | Target: Amazon Aurora MySQL |
| Created by: AWS | Environment: PoC or pilot | Technologies: Databases; Migration |
| Workload: Microsoft | AWS services: Amazon Aurora |
Summary

This pattern describes how to migrate a Microsoft SQL Server database that is either on premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance to Amazon Aurora MySQL. The pattern uses AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT) for data migration and schema conversion.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Microsoft SQL Server source database in an on-premises data center or on an EC2 instance
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed on either a local machine or an EC2 instance where AWS SCT is installed

Limitations

- Database size limit: 64 TB

Product versions

- Microsoft SQL Server 2008, 2008R2, 2012, 2014, 2016, and 2017 for the Enterprise, Standard, Workgroup, and Developer editions. The Web and Express editions aren’t supported by AWS DMS. For the latest list of supported versions, see Using a Microsoft SQL Server Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Microsoft SQL Server versions supported by AWS SCT, see the AWS SCT documentation.
- MySQL versions 5.5, 5.6, and 5.7. For the latest list of supported versions, see Using a MySQL-Compatible Database as a Target for AWS DMS.

Architecture

Source technology stack

One of the following:

- An on-premises Microsoft SQL Server database
- A Microsoft SQL Server database on an EC2 instance

Target technology stack

- Aurora MySQL

Data migration architecture

- From a Microsoft SQL Server database running in the AWS Cloud
- From a Microsoft SQL Server database running in an on-premises data center
Tools

- **AWS DMS** - AWS Data Migration Service (AWS DMS) helps you migrate your data to and from widely used commercial and open-source databases, including Oracle, SQL Server, MySQL, and PostgreSQL. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.

- **AWS SCT** - AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations easy by automatically converting the source database schema and a majority of the custom code to a format compatible with the target database.

Epics

Prepare for the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate the source and target database version and engine.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Create an outbound security group for the source and target databases.</td>
<td></td>
<td>SysAdmin</td>
</tr>
<tr>
<td>Create and configure an EC2 instance for AWS SCT, if required.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Download the latest version of AWS SCT and associated drivers.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Add and validate the prerequisite users and grants in the source database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Create an AWS SCT project for the workload and connect to the source database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Generate an assessment report and evaluate feasibility.</td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>

Prepare the target database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a target Amazon RDS DB instance, using Amazon Aurora as the database engine.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Extract the list of users, roles, and grants from the source.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Map the existing database users to the new database users.</td>
<td>App owner</td>
<td></td>
</tr>
<tr>
<td>Create users in the target database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Apply roles from the previous step to the target database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Review the database options, parameters, network files, and database links in the source database, and then evaluate their applicability to the target database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Apply any relevant settings to the target.</td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>

Transfer objects

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure AWS SCT connectivity to the target database.</td>
<td>DBA</td>
<td></td>
</tr>
<tr>
<td>Convert the schema using AWS SCT.</td>
<td>DBA</td>
<td></td>
</tr>
</tbody>
</table>
### Migrate from SQL Server to Aurora MySQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Convert automatically</strong></td>
<td>convert automatically is clearly marked so that you can convert it yourself.</td>
<td></td>
</tr>
<tr>
<td>Review the generated SQL</td>
<td>Review the generated SQL report and save any errors and warnings.</td>
<td>DBA</td>
</tr>
<tr>
<td>Apply automated schema</td>
<td>Apply automated schema changes to the target or save them as a .sql file.</td>
<td>DBA</td>
</tr>
<tr>
<td>Validate that AWS SCT</td>
<td>Validate that AWS SCT created the objects on the target.</td>
<td>DBA</td>
</tr>
<tr>
<td>Manually rewrite, reject,</td>
<td>Manually rewrite, reject, or redesign any items that failed to convert</td>
<td>DBA</td>
</tr>
<tr>
<td>Relocate or redesign</td>
<td>automatically.</td>
<td></td>
</tr>
<tr>
<td>Apply the generated role</td>
<td>Apply the generated role and user grants and review any exceptions.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Migrate the data

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the migration</td>
<td>Determine the migration method.</td>
<td>DBA</td>
</tr>
<tr>
<td>method.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a replication</td>
<td>Create a replication instance from the AWS DMS console.</td>
<td>DBA</td>
</tr>
<tr>
<td>instance</td>
<td>For detailed information on using AWS DMS, see the links in the &quot;Related</td>
<td></td>
</tr>
<tr>
<td>from the AWS DMS console.</td>
<td>resources&quot; section.</td>
<td></td>
</tr>
<tr>
<td>Create the source and</td>
<td>Create the source and target endpoints.</td>
<td>DBA</td>
</tr>
<tr>
<td>target endpoints.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a replication task.</td>
<td>Start the replication task and monitor the logs.</td>
<td>DBA</td>
</tr>
<tr>
<td>Start the replication task</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Migrate the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use AWS SCT to analyze</td>
<td>Use AWS SCT to analyze and convert the SQL items within the application</td>
<td>App owner</td>
</tr>
<tr>
<td>and convert the SQL items</td>
<td>code.</td>
<td></td>
</tr>
<tr>
<td>within the application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>code.</td>
<td>When you convert your database schema from one engine to another, you also</td>
<td></td>
</tr>
<tr>
<td>within the application</td>
<td>need to update the SQL code in your applications to interact with the new</td>
<td></td>
</tr>
<tr>
<td>code.</td>
<td>database engine instead of the old one. You can view, analyze, edit, and</td>
<td></td>
</tr>
<tr>
<td>code.</td>
<td>save.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>the converted SQL code. For detailed information on using AWS SCT, see the links in the &quot;Related resources&quot; section.</td>
<td></td>
</tr>
<tr>
<td>Create the new application servers on AWS.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Migrate the application code to the new servers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Configure the application server for the target database and drivers.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Fix any code that's specific to the source database engine in the application.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Optimize the application code for the target engine.</td>
<td></td>
<td>App owner</td>
</tr>
</tbody>
</table>

### Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply any new users, grants, and code changes to the target.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Lock the application for any changes.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Validate that all changes were propagated to the target database.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Point the new application server to the target database.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Recheck everything.</td>
<td></td>
<td>App owner</td>
</tr>
<tr>
<td>Go live.</td>
<td></td>
<td>App owner</td>
</tr>
</tbody>
</table>

### Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources (AWS DMS replication instance and EC2 instance used for AWS SCT).</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Update feedback on the AWS DMS process for internal teams.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>
**AWS Prescriptive Guidance Patterns**

**Migrate from on-premises MariaDB to Amazon RDS for MariaDB**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revise the AWS DMS process and improve the template if necessary.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
<tr>
<td>Close the project and provide any feedback.</td>
<td></td>
<td>DBA, App owner</td>
</tr>
</tbody>
</table>

**Related resources**

**References**
- AWS DMS User Guide
- AWS SCT User Guide
- Amazon Aurora Pricing

**Tutorials and videos**
- Getting Started with AWS Database Migration Service
- Getting Started with the AWS Schema Conversion Tool
- Amazon RDS resources
- AWS DMS Step-by-Step Walkthroughs

**Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools**

*Created by Sergey Dmitriev (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Databases: Relational</th>
<th>Target:</th>
<th>Amazon RDS for MariaDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Replatform</td>
<td>Workload:</td>
<td>Open-source</td>
<td>Technologies: Migration; Databases</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern provides guidance for migrating an on-premises MariaDB database to Amazon Relational Database Service (Amazon RDS) for MariaDB by using native tools. If you have MySQL tools installed, you can use `mysql` and `mysqldump`. If you have MariaDB tools installed, you can use `mariadb` and `mariadb-dump`. MySQL and MariaDB tools have the same origin, but there are minor differences in MariaDB version 10.6 and later.
Prerequisites and limitations

Prerequisites

- An active AWS account
- A MariaDB source database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- MariaDB versions 10.0-10.6 (for the latest list of supported versions, see MariaDB on Amazon RDS in the AWS documentation)

Architecture

Source technology stack

- MariaDB database in an on-premises data center

Target technology stack

- Amazon RDS for MariaDB DB instance

Target architecture
Data migration architecture

Tools
- Native MySQL tools: `mysql` and `mysqldump`
- Native MariaDB tools: `mariadb` and `mariadb-dump`

Epics

Plan the migration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate source and target database versions and engines.</td>
<td></td>
<td>DBA</td>
</tr>
<tr>
<td>Identify hardware requirements for the target server instance.</td>
<td></td>
<td>DBA, Systems administrator</td>
</tr>
</tbody>
</table>
**AWS Prescriptive Guidance Patterns**
*Migrate from on-premises MariaDB to Amazon RDS for MariaDB*

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify storage requirements (storage type and capacity).</td>
<td>DBA, Systems administrator</td>
<td></td>
</tr>
<tr>
<td>Choose the proper instance type based on capacity, storage features, and network features.</td>
<td>DBA, Systems administrator</td>
<td></td>
</tr>
<tr>
<td>Identify the network access security requirements for source and target databases.</td>
<td>DBA, Systems administrator</td>
<td></td>
</tr>
<tr>
<td>Identify the application migration strategy.</td>
<td>DBA, App owner, Systems administrator</td>
<td></td>
</tr>
</tbody>
</table>

**Configure the infrastructure**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a virtual private cloud (VPC).</td>
<td>Systems administrator</td>
<td></td>
</tr>
<tr>
<td>Create security groups.</td>
<td>Systems administrator</td>
<td></td>
</tr>
<tr>
<td>Configure and start an Amazon RDS DB instance running MariaDB.</td>
<td>Systems administrator</td>
<td></td>
</tr>
</tbody>
</table>

**Migrate data**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use native tools to migrate database objects and data.</td>
<td>In the source database, use <code>mysql dump</code> or <code>mariadb-dump</code> to create an output file that contains database objects and data. In the target database, use <code>mysql</code> or <code>mariadb</code> to restore the data.</td>
<td>DBA</td>
</tr>
<tr>
<td>Validate the data.</td>
<td>Check the source and target databases to confirm that the data migration was successful.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

**Migrate the application**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow the application migration strategy.</td>
<td></td>
<td>DBA, App owner, Systems administrator</td>
</tr>
</tbody>
</table>
Cut over

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch the application clients over to the new infrastructure.</td>
<td></td>
<td>DBA, App owner, Systems administrator</td>
</tr>
</tbody>
</table>

Close the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shut down the temporary AWS resources.</td>
<td></td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Review and validate the project documents.</td>
<td></td>
<td>DBA, App owner, Systems administrator</td>
</tr>
<tr>
<td>Gather metrics around time to migrate, cost savings provided by tools, and so on.</td>
<td></td>
<td>DBA, App owner, Systems administrator</td>
</tr>
<tr>
<td>Close out the project and provide feedback.</td>
<td></td>
<td>DBA, App owner, Systems administrator</td>
</tr>
</tbody>
</table>

Related resources

Amazon RDS references
- Amazon RDS for MariaDB
- Amazon Virtual Private Cloud VPCs and Amazon RDS
- Amazon RDS Multi-AZ Deployments
- Amazon RDS Pricing

MySQL and MariaDB references
- mariadb-dump/mysqldump
- mysql Command-line Client

Tutorials and videos
- Getting Started with Amazon RDS

Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3

Created by Rohan Jamadagni (AWS) and Udayasimha Theepireddy (AWS)
Summary

This pattern describes how to migrate large, on-premises MySQL databases efficiently to Amazon Aurora MySQL by using Percona XtraBackup. Percona XtraBackup is an open-source, non-blocking backup utility for MySQL-based servers. The pattern shows how to use Amazon Elastic File System (Amazon EFS) to reduce the time to upload the backup to Amazon Simple Storage Service (Amazon S3) and to restore the backup to Amazon Aurora MySQL. The pattern also provides details on how to make incremental Percona backups to minimize the number of binary logs to be applied to the target Aurora MySQL database.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS Identity and Access Management (IAM) user with permissions to create IAM roles and policies.
- Network connectivity between the on-premises MySQL database and the virtual private cloud (VPC) on AWS.

Limitations

- The source servers must be Linux-based systems that can install a Network File System (NFS) client (nfs-utils/nfs-common).
- The S3 bucket used for uploading backup files supports server-side encryption (SSE-S3/SSE-KMS) only.
- Amazon S3 limits the size of the backup files to 5 TB. If your backup file exceeds 5 TB, you can split it into multiple, smaller files.
- The number of source files uploaded to the S3 bucket cannot exceed one million files.
- The pattern supports Percona XtraBackup full backup and incremental backup only. It doesn't support partial backups that use --tables, --tables-exclude, --tables-file, --databases, --databases-exclude, or --databases-file.
- Aurora doesn't restore user accounts, functions, stored procedures, or time zone information from the source MySQL database.

Product versions

- The source database must be MySQL version 5.5, 5.6, or 5.7.
- For MySQL 5.7, you must use Percona XtraBackup 2.4.
- For MySQL 5.6 and 5.6, you must use Percona XtraBackup 2.3 or 2.4.
Architecture

Source technology stack

- Linux-based operating system
- MySQL server
- Percona XtraBackup

Target technology stack

- Amazon Aurora
- Amazon S3
- Amazon EFS

Target architecture

Tools

- Percona XtraBackup – An open-source utility that performs streaming, compressed, and incremental backups of MySQL databases without disrupting or blocking your databases.
- Amazon Aurora – A fully managed relational database engine that makes it simple and cost-effective to set up, operate, and scale MySQL deployments. Aurora MySQL is a drop-in replacement for MySQL.
- Amazon S3 – An object storage service that offers scalability, data availability, security, and performance. You can use Amazon S3 to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.
- **Amazon EFS** – A scalable, fully managed, elastic NFS file system that you can use with AWS Cloud services and on-premises resources. Amazon EFS is built to scale on demand to petabytes without disrupting applications, and automatically grows and shrinks as you add and remove files.

**Epics**

Create an Amazon EFS file system

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a security group to associate with Amazon EFS mount targets.</td>
<td>Create a security group in the VPC that is configured with a VPN attachment to the on-premises database over AWS Transit Gateway. For more information about the commands and steps described in this and other stories, see the links in the &quot;Related resources&quot; section at the end of this pattern.</td>
<td>AWS DevOps/database administrator</td>
</tr>
<tr>
<td>Edit the security group rules.</td>
<td>Add an inbound rule, using type NFS, port 2049, and the IP range of the on-premises database server as the source. By default, the outbound rule allows all the traffic to leave. If this is not the case, add an outbound rule to open a connection for the NFS port. Add two more inbound rules: port 2049 (source: security group ID of this same security group) and port 22 (source: IP range from where you will connect to an EC2 instance).</td>
<td>AWS DevOps/database administrator</td>
</tr>
<tr>
<td>Create a file system.</td>
<td>In the mount targets, use the VPC and security group you created in the previous story. Choose the throughput mode and performance based on the I/O requirements of the on-premises database. Optionally, enable encryption at rest.</td>
<td>AWS DevOps/database administrator</td>
</tr>
</tbody>
</table>

**Mount the file system**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM instance profile role to be associated with an EC2 instance.</td>
<td>Create an IAM role that has permissions to upload objects to, and access objects in, Amazon S3. Choose the S3</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
### Migrate from on-premises MySQL to Aurora MySQL using Percona XtraBackup

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an EC2 instance.</td>
<td>Launch an Linux-based EC2 instance and attach the IAM instance profile role that you created in the previous step, and the security group you created earlier.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Install the NFS client.</td>
<td>Install the NFS client on the on-premises database server and on the EC2 instance. For installation instructions, see the &quot;Additional information&quot; section.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Mount the Amazon EFS file system.</td>
<td>Mount the Amazon EFS file system on premises and on the EC2 instance. On each server, create a directory for storing the backup, and mount the file system by using the mount target endpoint. For an example, see the &quot;Additional information&quot; section.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Make a backup of the MySQL source database

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Percona XtraBackup.</td>
<td>Install Percona XtraBackup 2.3 or 2.4 (depending on the version of your MySQL database) on the on-premises database server. For installation links, see the &quot;Related resources&quot; section.</td>
<td>Database administrator</td>
</tr>
<tr>
<td>Count the schemas and tables in the source database.</td>
<td>Gather and note the number of schemas and objects in the source MySQL database. You will use these counts to validate the Aurora MySQL database after migration.</td>
<td>Database administrator</td>
</tr>
<tr>
<td>(Optional) Note the latest binary log sequence from the source database.</td>
<td>Perform this step if you want to establish binary log replication between the source database and Aurora MySQL to minimize downtime. log-bin must be enabled, and server_id must be unique. Note the current binary log sequence from the source database, just before initiating a backup. Perform this step just before full backup if</td>
<td>Database administrator</td>
</tr>
</tbody>
</table>
## Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start a full backup of the source MySQL database.</strong></td>
<td>Start a full backup of the source MySQL database. If you're planning to use only full backup, if you're planning to make incremental backups after a full backup, perform this step just before the final incremental backup that you will restore on the Aurora MySQL DB instance.</td>
<td><strong>Database administrator</strong></td>
</tr>
<tr>
<td><strong>(Optional) Make incremental backups using Percona XtraBackup.</strong></td>
<td>Make a full backup of the MySQL source database using Percona XtraBackup. For example commands for full and incremental backups, see the &quot;Additional information&quot; section. Incremental backups can be used to reduce the amount of binary logs you need to apply to sync the source database with Aurora MySQL. Large-size and transaction-heavy databases might generate a large number of binary logs during backups. By taking incremental backups and storing them on a shared Amazon EFS file system, you can significantly reduce the time for backing up and uploading your database. For details, see the &quot;Additional information&quot; section. Continue to make incremental backups until you're ready to begin the migration process to Aurora.</td>
<td><strong>Database administrator</strong></td>
</tr>
<tr>
<td><strong>Prepare backups.</strong></td>
<td>In this step, transactional logs are applied to the backup for transactions that were in flight during the backup. Continue to apply transactional logs (--apply-log-only) to each incremental backup to merge the backups, except for the last backup. For examples, see the &quot;Additional information&quot; section. After this step, the complete, merged backup will be in ~/&lt;efs_mount_name&gt;/fullbackup.</td>
<td><strong>Database administrator</strong></td>
</tr>
<tr>
<td><strong>Zip and split the final merged backup.</strong></td>
<td>After you prepare the final, merged backup, use tar, zip, and split commands to create smaller zipped files of the backup. For examples, see the &quot;Additional information&quot; section.</td>
<td><strong>Database administrator</strong></td>
</tr>
</tbody>
</table>
## Migrate from on-premises MySQL to Aurora MySQL using Percona XtraBackup

### Task: Restore the backup to an Aurora MySQL DB cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the backup to Amazon S3.</td>
<td>The Amazon EFS file system where the backup files are stored is mounted on both the on-premises database and an EC2 instance, so the backup files are readily available to the EC2 instance. Connect to the EC2 instance by using Secure Shell (SSH) and upload the zipped backup files to a new or existing S3 bucket; for example: <code>aws s3 sync ~/&lt;efs_mount_name&gt;/fullbackup s3://&lt;bucket_name&gt;/fullbackup</code>. For additional details, see the links in the &quot;Related resources&quot; section.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Create a service role for Aurora to access Amazon S3.</td>
<td>Create an IAM role with trust &quot;rds.amazonaws.com&quot; and a policy that will enable Aurora to access the S3 bucket where the backup files are stored. The required permissions are ListBucket, GetObject, and GetObjectVersion.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Create the networking configuration for Aurora.</td>
<td>Create a cluster DB subnet group with at least two Availability Zones and a subnet route table configuration that allows outbound connectivity to the source database. Create a security group that allows outbound connections to the on-premises database, and allows administrators to connect to the Aurora DB cluster. For more information, see the links in the &quot;Related resources&quot; section.</td>
<td>AWS DevOps/database administrator</td>
</tr>
<tr>
<td>Restore the backup to an Aurora MySQL DB cluster.</td>
<td>Restore your data from the backup that you uploaded to Amazon S3. Specify the MySQL version of your source database, provide the S3 bucket name and folder path prefix where you uploaded the backup file (for example, &quot;fullbackup&quot; for the examples in the &quot;Additional information&quot; section), and provide the IAM role you created.</td>
<td>AWS DevOps/database administrator</td>
</tr>
</tbody>
</table>
Migrate from on-premises MySQL to Aurora MySQL using Percona XtraBackup

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>to authorize Aurora to access Amazon S3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validate the Aurora MySQL database.</td>
<td>Validate the count of schema and objects in the restored Aurora DB cluster</td>
<td>Database administrator</td>
</tr>
<tr>
<td></td>
<td>against the count you obtained from the source database.</td>
<td></td>
</tr>
<tr>
<td>Set up binlog replication.</td>
<td>Use the binary log sequence that you noted earlier, before making the last</td>
<td>AWS DevOps/database administrator</td>
</tr>
<tr>
<td></td>
<td>backup that was restored to the Aurora DB cluster. Create a replication user</td>
<td></td>
</tr>
<tr>
<td></td>
<td>on the source database, and follow the instructions in the &quot;Additional</td>
<td></td>
</tr>
<tr>
<td></td>
<td>information&quot; section to provide the appropriate privileges, to enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>replication on Aurora, and to confirm that the replication is in sync.</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

Creating an Amazon EFS file system

- Creating a security group (Amazon VPC documentation)
- Transit gateway VPN attachments (Amazon VPC documentation)
- Scaling VPN throughput using AWS Transit Gateway (Networking & Content Delivery blog)
- Creating an Amazon EFS file system (Amazon EFS documentation)
- Creating mount targets (Amazon EFS documentation)
- Encrypting data at rest (Amazon EFS documentation)

Mounting the file system

- IAM roles for Amazon EC2 (Amazon EC2 documentation)
- Launching an Amazon EC2 Linux instance (Amazon EC2 documentation)
- Installing the NFS client (Amazon EFS documentation)
- Mounting the Amazon EFS file system on your on-premises client (Amazon EFS documentation)
- Mounting EFS File Systems (Amazon EFS documentation)

Making a backup of the MySQL source database

- Installing Percona XtraBackup 2.3 (Percona XtraBackup documentation)
- Installing Percona XtraBackup 2.4 (Percona XtraBackup documentation)
- Setting the replication master configuration (MySQL documentation)
- Migrating data from an external MySQL database to an Aurora MySQL DB cluster (Aurora documentation)
- Incremental backup (Percona XtraBackup documentation)
Restoring the backup to Amazon Aurora MySQL

- Creating a bucket (Amazon S3 documentation)
- Connecting to your Linux instance using SSH (Amazon Ec2 documentation)
- Configuring the AWS CLI (AWS CLI documentation)
- sync command (AWS CLI command reference)
- Creating an IAM policy to access Amazon S3 resources (Aurora documentation)
- DB cluster prerequisites (Aurora documentation)
- Working with DB subnet groups (Aurora documentation)
- Creating a VPC security group for a private DB instance (Aurora documentation)
- Restoring an Aurora MySQL DB cluster from an S3 bucket (Aurora documentation)
- Setting up replication with MySQL or another Aurora DB cluster (Aurora documentation)
- mysql.rds_set_external_master procedure (MySQL on Amazon RDS SQL reference)
- mysql.rds_start_replication procedure (MySQL on Amazon RDS SQL reference)

Additional references

- Migrating data from an external MySQL database to an Aurora MySQL DB cluster (Aurora documentation)
- MySQL server downloads (Oracle website)

Tutorials and videos

- Migrating MySQL data to an Aurora MySQL DB cluster using Amazon S3 (AWS Knowledge Center)
- Amazon EFS setup and mount (video)

Additional information

Installing an NFS client

- If you are using Red Hat or a similar Linux operating system, use the command:

```bash
$ sudo yum -y install nfs-utils
```

- If you are using Ubuntu or a similar Linux operating system, use the command:

```bash
$ sudo apt-get -y install nfs-common
```

For more information, see the walkthrough in the Amazon EFS documentation.

Mounting the Amazon EFS file system

Use the commands:

```bash
mkdir ~/<efs_mount_name>
$ sudo mount -t nfs -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-target-IP: ~/<efs_mount_name>
```
For more information, see the walkthrough and Mounting EFS File Systems in the Amazon EFS documentation.

**Making backups of the MySQL source database**

**Full backups**

Use a command like the following, which takes the backup, zips it, and splits it into smaller chunks of 1 GB each:

```bash
xtrabackup --backup --user=dbuser --password=<password> --binlog-info=AUTO --stream=tar --target-dir=~/<efs_mount_name>/fullbackup | gzip - | split -d --bytes=1024MB -/~/<efs_mount_name>/fullbackup/backup.tar.gz &
```

If you're planning to make subsequent incremental backups after the full backup, do not zip and split the backup. Instead, use a command similar to the following:

```bash
xtrabackup --backup --user=dbuser --password=<password> --target-dir=~/<efs_mount_name>/fullbackup/
```

**Incremental backups**

Use the full backup path for the --incremental-basedir parameter; for example:

```bash
xtrabackup --backup --user=dbuser --password=<password> --target-dir=~/<efs_mount_name>/incremental/backupdate --incremental-basedir=~/<efs_mount_name>/fullbackup
```

where `basedir` is the path to the full backup and the xtrabackup_checkpoints file.

For more information about making backups, see Migrating Data from an External MySQL Database to an Amazon Aurora MySQL DB Cluster in the Aurora documentation.

**Preparing backups**

To prepare a full backup:

```bash
xtrabackup --prepare --apply-log-only --target-dir=~/<efs_mount_name>/fullbackup
```

To prepare an incremental backup:

```bash
xtrabackup --prepare --apply-log-only --target-dir=~/<efs_mount_name>/fullbackup --incremental-dir=~/<efs_mount_name>/incremental/06062020
```

To prepare the final backup:

```bash
xtrabackup --prepare --target-dir=~/<efs_mount_name>/fullbackup --incremental-dir=~/<efs_mount_name>/incremental/06072020
```

For more information, see Incremental backups in the Percona XtraBackup documentation.

**Zipping and splitting the merged backup**

To zip the merged backup at ~/<efs_mount_name>/fullbackup:

```bash
tar -zcvf <backupfilename.tar.gz> ~/<efs_mount_name>/fullbackup
```

To split the backup:
Setting up binlog replication

To create a replication user on the source database and provide the appropriate privileges:

```
CREATE USER 'repl_user'@'' IDENTIFIED BY ''; GRANT REPLICATION CLIENT, REPLICATION SLAVE ON
*.* TO 'repl_user'@'';
```

To enable replication on Aurora by connecting to the Aurora DB cluster:

```
CALL mysql.rds_set_external_master ('sourcedbinstanceIP', sourcedbport, 'repl_user', '',
'binlog_file_name', binlog_file_position, 0); CALL mysql.rds_start_replication;
```

To confirm that the replication is in sync:

```
SHOW Slave Status \G;
```

The **Seconds behind master** field shows how far behind Aurora is from the on-premises database.

**Migrate on-premises Java applications to AWS using AWS App2Container**

*Created by Dhananjay Karanjkar (AWS)*

<table>
<thead>
<tr>
<th><strong>Source:</strong></th>
<th>Applications</th>
<th><strong>Target:</strong></th>
<th>Containerized application deployed on Amazon ECS</th>
<th><strong>R Type:</strong></th>
<th>Replatform</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environment:</strong></td>
<td>PoC or pilot</td>
<td><strong>Technologies:</strong></td>
<td>Migration; Websites &amp; web apps</td>
<td><strong>Workload:</strong></td>
<td>Open-source</td>
</tr>
<tr>
<td><strong>AWS services:</strong></td>
<td>AWS EC2</td>
<td>Container Registry; Amazon ECS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

AWS App2Container (A2C) is a command line tool that helps transform existing applications running in virtual machines into containers, without needing any code changes. A2C discovers applications running on a server, identifies dependencies, and generates relevant artifacts for seamless deployment to Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS).

This pattern provides the steps for remotely migrating on-premises Java applications deployed on an application server to AWS Fargate or Amazon EKS by using App2Container through the worker machine.

The worker machine can be used in the following use cases:

- Docker installation is not allowed or not available on the application servers where the Java applications are running.
- You must manage the migration of multiple applications deployed on different physical or virtual servers.
Prerequisites and limitations

Prerequisites

- An application server with a Java application running on a Linux server
- A worker machine with a Linux operating system
- A worker machine with at least 20 GB of available disk space

Limitations

- Not all applications are supported. For more information, see Supported applications for Linux.

Architecture

Source technology stack

- Java applications running on Linux server

Target technology stack

- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline
- Amazon Elastic Container Registry
- AWS Fargate

Target architecture
Tools

Tools

- **AWS App2Container** – AWS App2Container (A2C) is a command line tool to help you lift and shift applications that run in your on-premises data centers or on virtual machines, so that they run in containers that are managed by Amazon ECS or Amazon EKS.
- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodeCommit** – AWS CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.
- **AWS CodePipeline** – AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.
- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that for running, stopping, and managing containers on a cluster.
- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
- **Amazon EKS** – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- **AWS Fargate** – AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon Elastic Compute Cloud (Amazon EC2) instances. With Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers.

Epics

Set up credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a secret to access the application server.</td>
<td>To access the application server remotely from the worker machine, create a secret in AWS Secrets Manager. For your secret, you can use either the SSH private key or the Certificate and the SSH private key. For more information, see Manage secrets for AWS App2Container.</td>
<td>DevOps, Developer</td>
</tr>
</tbody>
</table>

Set up the worker machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the tar file.</td>
<td>Run <code>sudo yum install -y tar</code>.</td>
<td>DevOps, Developer</td>
</tr>
</tbody>
</table>

<p>| Install the AWS CLI. | To install the Amazon Command Line Interface (AWS CLI), run <code>curl &quot;https://</code> | DevOps, Developer     |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Install App2Container.</strong></td>
<td>Run the following commands:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* awscli.amazonaws.com/awscli-exe-linux-x86_64.zip&quot; -o &quot;awscliv2.zip&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Unzip awscliv2.zip.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Run sudo ./aws/install.</td>
</tr>
<tr>
<td></td>
<td><strong>Configure the profiles.</strong></td>
<td>To configure the AWS default profile, run <code>sudo aws configure</code>.</td>
</tr>
</tbody>
</table>
|      |             |   * To configure the named AWS default profile, run `sudo aws configure --profile <profile name>`.
|      | **Install Docker.** | Run the following commands.  |
|      |             |   * `sudo yum install -y docker`  |
|      |             |   * `sudo systemctl enable docker & sudo systemctl restart docker` |
### Initialize App2Container.

**Task**
- **Description**: To initialize App2Container, you need the following information:
  - **workspace**: To store application containerization artifacts. We recommend providing a directory path that has at least 20 GB of free disk space.
  - **awsProfile**: AWS profile configured on the server. This is required to upload artifacts to Amazon S3, run the `containerize` command, and generate AWS artifacts for deployment on Amazon ECS or Amazon EKS.
  - **s3Bucket**: To extract and store AWS artifacts.
  - **metricsReportPermission**: To collect and store metrics reported.
  - **dockerContentTrust**: To sign the Docker image.

**Run**
- `sudo app2container init`.

**Skills required**: DevOps, Developer

### Configure the worker machine

**Task**
- **Description**: To configure the worker machine to remotely connect and run App2Container commands on the application server, the following information is required:
  - **Server FQDN**: The fully qualified domain name of the application server.
  - **Server IP address**: The IP address of the application server. Either the FQDN or the IP address is sufficient.
  - **SecretARN**: The Amazon Resource Name (ARN) of the secret that is used to connect to the application server and is stored in Secrets Manager.
  - **AuthMethod**: The key or cert authentication method.

**Skills required**: DevOps, Developer
# Migrate on-premises applications using App2Container

## Task: Discover, analyze, and extract applications on the worker machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Discover the on-premises Java applications.** | To remotely discover all the applications running on the application server, run the following command.  

```bash
sudo app2container remote inventory --target <FQDN/IP of App server>
```

This command generates a list of deployed applications in `inventory.json`. | Developer, DevOps |
| **Analyze the discovered applications.**     | To remotely analyze each application by using the application-id obtained in the inventory stage, run the following command.  

```bash
sudo app2container remote analyze --application-id <java-app-id> --target <FQDN/IP of App Server>
```

This generates `analysis.json` file in the workspace location. After this file is generated, you can alter the containerization parameters based on your needs. | Developer, DevOps |
| **Extract the analyzed applications.**       | To generate an application archive for the analyzed application, remotely run the following command, which will generate the tar bundle in the workspace location.  

```bash
sudo app2container remote extract --application-id <application id> --target <FQDN/IP of App Server>
```

|  |

---

1283
# Migrate on-premises applications using App2Container

## Containerize the extracted artifacts on the worker machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containerize</td>
<td>Containerize the artifacts extracted in the previous step by running the following command.</td>
<td>Developer, DevOps</td>
</tr>
<tr>
<td></td>
<td><code>sudo app2container containerize --input-archive &lt;tar bundle location on worker machine&gt;</code></td>
<td></td>
</tr>
<tr>
<td>Finalize the target.</td>
<td>To finalize the target, open <code>deployment.json</code>, which is created when the containerize command runs. To specify AWS Fargate as the target, set <code>createEcsArtifacts</code> to <code>true</code>. To set Amazon EKS as the target, set <code>createEksArtifacts</code> to <code>true</code>.</td>
<td>Developer, DevOps</td>
</tr>
</tbody>
</table>

## Generate and provision AWS artifacts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate AWS</td>
<td>To generate deployment artifacts, run the following command.</td>
<td>DevOps</td>
</tr>
<tr>
<td>deployment artifacts</td>
<td><code>sudo app2container generate app-deployment --application-id &lt;application id&gt;</code></td>
<td></td>
</tr>
<tr>
<td>on the worker machine.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provision the artifacts.</td>
<td>To further provision the generated artifacts, deploy the AWS CloudFormation template by running the following command.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Populate your CMDB after integrating AWS Config with ServiceNow

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>aws cloudformation deploy --template-file &lt;path to ecs-master.yml&gt; --capabilities CAPABILITY_NAMED_IAM --stack-name &lt;application id&gt;=ECS</code></td>
<td>aws cloudformation deploy --template-file &lt;path to ecs-master.yml&gt; --capabilities CAPABILITY_NAMED_IAM --stack-name &lt;application id&gt;=ECS</td>
</tr>
<tr>
<td>Generate the pipeline.</td>
<td>Modify pipeline.json, which was created in the previous story, based on your needs. Then run the generate pipeline command to generate the pipeline deployment artifacts.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

Related resources
- What is App2Container?
- AWS App2Container blog post
- AWS CLI configuration basics
- Docker basics for Amazon ECS
- Docker commands

Populate your CMDB after integrating AWS Config with ServiceNow

Created by Rahul Goyal (AWS) and Andrew Wageh (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Data Management</th>
<th>Target:</th>
<th>AWS Config</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Replatform</td>
<td>Technologies:</td>
<td>Migration; Management &amp; governance</td>
<td>AWS services:</td>
<td>AWS CloudFormation; Amazon SNS; Amazon EC2</td>
</tr>
</tbody>
</table>

Summary
This pattern demonstrates how to integrate AWS Config with ServiceNow so that you can update your configuration management database (CMDB) after you spin up an Amazon Elastic Compute Cloud (Amazon EC2) instance in the Amazon Web Services (AWS) Cloud. The pattern works by using Amazon Simple Notification Service (Amazon SNS) to issue a notification after your EC2 instance spins up. The notification creates a server record in your CMDB and then tests the integration by creating an EC2 instance from an AWS CloudFormation stack.

Prerequisites and limitations

Prerequisites
• An active AWS account
• An active ServiceNow account (developer accounts are valid)
• A CMDB
• Basic knowledge of ServiceNow, AWS Config, and Amazon SNS

Architecture

The following diagram shows you how to update your CMDB data by integrating AWS Config with ServiceNow.

The diagram shows the following workflow:

1. The user provisions the EC2 instance by using an AWS CloudFormation stack.
2. AWS Config records the information about the EC2 instance from the AWS CloudFormation stack.
3. The information that AWS Config captures about the EC2 instance gets published to the Amazon SNS topic.
4. Amazon SNS pushes the SNS topic to HTTP/S webhooks.
5. The ServiceNow REST API receives information about the EC2 instance from the HTTP/S webhooks.
6. JavaScript processes the API data.
7. A JSON handler extracts the server information.
8. The CMDB gets updated with the server information.

Technology stack

• Amazon SNS
• AWS CloudFormation
• AWS Config
• ServiceNow
Tools

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **AWS Config** – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.

- **ServiceNow** – ServiceNow is a cloud management solution that integrates with the AWS Cloud to provide comprehensive service lifecycle management.

Epics

**Configure the ServiceNow instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Request a ServiceNow instance. | 1. Sign in to the ServiceNow Developer Site and request a developer instance (ServiceNow website).  
2. Log in to the developer instance as an administrator and note your login credentials for later use. | ServiceNow      |
| Import a custom application in ServiceNow. | 1. On the ServiceNow Developer Site (ServiceNow website), choose System Applications.  
2. Choose Studio and then choose Import From Source Control.  
4. Avoid entering a user name and password, and then choose Import.  
5. Close the tab for ServiceNow Studio. | ServiceNow      |
| Validate your new application in ServiceNow. | Refresh your ServiceNow browser tab and then choose AWS SNS. There are now three new navigation links in the left navigation pane. | ServiceNow      |
### Create an SNS topic and subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SNS topic.</td>
<td>Sign in to the AWS Management Console and use the instructions from Creating an Amazon SNS topic in the Amazon SNS documentation to create an SNS topic in the US East (N. Virginia) AWS Region.</td>
<td>Amazon SNS</td>
</tr>
</tbody>
</table>
| Subscribe the endpoint to the SNS topic. | Sign in to the AWS Management Console and use the instructions from Subscribing to an Amazon SNS topic in the Amazon SNS documentation to subscribe your ServiceNow instance to your new SNS topic.  
When you create the subscription, be sure to:  
  • Choose HTTP/HTTPS for Protocol.  
  • Enter the link to your ServiceNow developer instance for Endpoint (for example, https://admin:[your ServiceNow admin password]@[your developer instance].service-now.com/api/x_snc_aws_sns/aws_sns). Note: To get the link to your ServiceNow developer instance, use the administrator password that you received when you requested a free ServiceNow developer instance. | Amazon SNS          |

### Confirm the SNS subscription in ServiceNow

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the subscription.</td>
<td>1. Sign in to the ServiceNow Developer Site (ServiceNow website).</td>
<td>ServiceNow</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Populate your CMDB after integrating
AWS Config with ServiceNow

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. On your ServiceNow browser tab, choose <strong>SNS</strong> and then choose <strong>Subscriptions</strong>. You can see that AWS created a new record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Confirm the subscription.</strong></td>
<td><strong>ServiceNow</strong></td>
</tr>
</tbody>
</table>
|      | 1. To open the subscription, choose **ServiceNow**.  
2. Choose **Confirm Subscription**. **Note**: Stay on this page to create a handler in the next section. |                |

Create a handler for the subscription in ServiceNow

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Create a handler.</strong></td>
<td><strong>ServiceNow</strong></td>
</tr>
<tr>
<td></td>
<td>To allow ServiceNow to receive messages from AWS, create a handler in ServiceNow. The handler creates a new record in the CMDB server table (<strong>cmdb_ci_server</strong>) whenever a new EC2 instance is created from an AWS CloudFormation stack.</td>
<td></td>
</tr>
</tbody>
</table>
|      | 1. For **Handler Related** on the **Subscriptions** form, choose **New**.  
2. For **Name**, enter a name for the handler (for example, **Create CMDB Server from EC2**).  
3. For **Code**, paste the following code into the function that's inside the handler: |                |
|      | var webserver = new GlideRecord("cmdb_ci_server");  
webserver.initialize();  
webserver.name = "AWS WebServer +"+message.configurationItem.configuration.launchTime ;  
webserver.short_description = "Monitoring is "+message.configurationItem.configuration.monitoring.state 
"and Instance Type is" +"+message.configurationItem.configuration.instanceType ;  
webserver.asset_tag = message.configurationItem.configuration.instanceId ;  
webserver.insert();  
4. Choose **Submit**. |                |
Configure AWS Config

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Update your configuration settings. | 1. Sign in to the AWS Management console and choose the AWS Config console.  
2. On the navigation pane, choose **Settings**, and then choose **Edit**.  
3. In the **Recorder** section, confirm that the **Enable recording** check box is selected.  
4. In the **General Settings** section, for **Resource types to record**, choose **Record all resources supported in this region** and then select the **Include global resources (e.g., AWS IAM resources)** check box.  
5. In the **Delivery method** section, for **Amazon SNS topic**, select the **Stream configuration changes and notifications to an Amazon SNS topic** check box, and then choose a topic from your account.  
6. For **SNS topic name**, choose the Amazon Resource Name (ARN) link for the topic from step 5.  
7. Choose **Save**. | AWS Config |

Test the integration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a test AWS CloudFormation stack. | You can test the integration between AWS Config and ServiceNow by creating a test AWS CloudFormation stack that initiates a recording in AWS Config. This process then creates an SNS notification, which creates a configuration item in the ServiceNow CMDB.  
Sign in to the AWS Management Console and use the instructions | AWS CloudFormation |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>from Creating a stack on the AWS CloudFormation console to create a test stack.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confirm the creation of the server.</td>
<td>1. Sign in to the ServiceNow Developer Site (ServiceNow website). 2. Choose Server and then confirm that a new server was created.</td>
<td>AWS CloudFormation</td>
</tr>
</tbody>
</table>

### Related resources

- AWS Config ([AWS Config documentation](#))
- ServiceNow ([ServiceNow documentation](#))
- Amazon Simple Notification Service ([Amazon SNS documentation](#))
- AWS CloudFormation ([AWS CloudFormation documentation](#))

### Additional information

This pattern demonstrates a mechanism for integrating AWS Config with your ServiceNow CMDB, which captures instance details as configuration items in the CMDB server table whenever EC2 instances are created by using AWS CloudFormation.

You can use this mechanism to do the following:

- Use ServiceNow handlers to update the CMDB record with instance details.
- Customize your handlers to scale the integration of AWS Config with your ServiceNow CMDB.
- Get updated instance details and other information.
- Send notifications and perform actions, including discovery, workflow, and more. For example, if you make a small change across a list of resource types (such as adding a tag), then you can bypass discovery needs and discover existing resources. A change recording is initiated in AWS Config, which then creates those resources in the CMDB.

### Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition

*Created by Dwarika Patra (AWS) and Deepesh J (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>SQL Server</th>
<th>Target:</th>
<th>Aurora PostgreSQL-Compatible</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Replatform</td>
<td>Workload:</td>
<td>Microsoft; Open-source</td>
<td>Technologies:</td>
<td>Migration; Databases</td>
</tr>
</tbody>
</table>
AWS Prescriptive Guidance Patterns
Python and Perl application changes
to support database migrations

AWS services: Amazon Aurora

Summary

This pattern describes changes to application repositories that might be required when you migrate databases from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition. The pattern assumes that these applications are Python-based or Perl-based, and provides separate instructions for these scripting languages.

Migrating SQL Server databases to Aurora PostgreSQL-Compatible involves schema conversion, database object conversion, data migration, and data loading. Because of the differences between PostgreSQL and SQL Server (relating to data types, connection objects, syntax, and logic), the most difficult migration task involves making the necessary changes to the code base so that it works correctly with PostgreSQL.

For a Python-based application, connection objects and classes are scattered throughout the system. Also, the Python code base might use multiple libraries to connect to the database. If the database connection interface changes, the objects that run the application’s inline queries also require changes.

For a Perl-based application, changes involve connection objects, database connection drivers, static and dynamic inline SQL statements, and how the application handles complex dynamic DML queries and results sets.

When you migrate your application, you can also consider possible enhancements on AWS, such as replacing the FTP server with Amazon Simple Storage Service (Amazon S3) access.

The application migration process involves the following challenges:

• Connection objects. If connection objects are scattered in the code with multiple libraries and function calls, you might have to find a generalized way to change them to support PostgreSQL.

• Error or exception handling during record retrieval or updates. If you have conditional create, read, update, and delete (CRUD) operations on the database that return variables, results sets, or data frames, any errors or exceptions might result in application errors with cascading effects. These should be handled carefully with proper validations and save points. One such save point is to call large inline SQL queries or database objects inside `BEGIN...EXCEPTION...END` blocks.

• Controlling transactions and their validation. These includes manual and automatic commits and rollbacks. The PostgreSQL driver for PERL requires you to always explicitly set the auto-commit attribute.

• Handling dynamic SQL queries. This requires a strong understanding of the query logic and iterative testing to ensure that queries work as expected.

• Performance. You should ensure that code changes don't result in degraded application performance.

This pattern explains the conversion process in detail. You can automate some of these steps through small scripts.

Prerequisites and limitations

Prerequisites

• Working knowledge of Python and Perl syntax.
• Basic skills in SQL Server and PostgreSQL.
• Understanding of your existing application architecture.
• Access to your application code, SQL Server database, and PostgreSQL database.
• Access to Windows, Linux (or other Unix) development environment with credentials for developing, testing, and validating application changes.
• For a Python-based application, standard Python libraries that your application might require, such as Pandas to handle data frames, and psycopg2 or SQLAlchemy for database connections.
• For a Perl-based application, required Perl packages with dependent libraries or modules. The Comprehensive Perl Archive Network (CPAN) module can support most application requirements.
• All required dependent customized libraries or modules.
• Database credentials for read access to SQL Server and read/write access to Aurora.
• PostgreSQL to validate and debug application changes with service and user accounts.
• Access to development tools during application migration such as Visual Studio Code, Sublime Text, or pgAdmin.

Limitations
• Some Python or Perl versions, modules, libraries, and packages aren’t compatible with the cloud environment.
• Some third-party libraries and frameworks used for SQL Server cannot be replaced to support PostgreSQL migration.
• Performance variations might require changes to your application, to inline Transact-SQL (T-SQL) queries, database functions, and stored procedures.
• PostgreSQL supports lowercase names for table names, column names, and other database objects.
• Some data types, such as UUID columns, are stored in lowercase only. Python and Perl applications must handle such case differences.
• Character encoding differences must be handled with the correct data type for the corresponding text columns in the PostgreSQL database.

Product versions
• Python 3.6 or later (use the version that supports your operating system)
• Perl 5.8.3 or later (use the version that supports your operating system)
• Aurora PostgreSQL-Compatible Edition 4.2 or later (see details)

Architecture

Source technology stack
• Scripting (application programming) language: Python 2.7 or later, or Perl 5.8
• Database: Microsoft SQL Server version 13
• Operating system: Red Hat Enterprise Linux (RHEL) 7

Target technology stack
• Scripting (application programming) language: Python 3.6 or later, or Perl 5.8 or later
• Database: Aurora PostgreSQL-Compatible 4.2
• Operating system: RHEL 7

Migration architecture
Tools

AWS services and tools

- **Aurora PostgreSQL-Compatible** – Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine that combines the speed and reliability of high-end commercial databases with the cost-effectiveness of open-source databases. Aurora PostgreSQL is a drop-in replacement for PostgreSQL and makes it easier and more cost-effective to set up, operate, and scale your new and existing PostgreSQL deployments.

- **AWS CLI** – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services by using commands in your command-line shell.

Other tools

- Python and PostgresSQL database connection libraries such as psycopg2 and SQLAlchemy
- Perl and its DBI modules
- PostgreSQL interactive terminal (psql)

Epics

Migrate your application repository to PostgreSQL – high-level steps

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Follow these code conversion steps to migrate your application to PostgreSQL. | 1. Set database-specific ODBC drivers and libraries for PostgreSQL. For example, you can use one of the CPAN modules for Perl and pyodbc, psycopg2, or SQLAlchemy for Python.  
2. Convert database objects by using these libraries to connect to Aurora PostgreSQL-Compatible.  
3. Apply code changes in existing application modules | App developer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>to get compatible T-SQL statements.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Rewrite database-specific function calls and stored procedures in application code.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Handle changes to your application’s variables and their data types that are used for inline SQL queries.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Handle incompatible database-specific functions.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Complete end-to-end testing of converted application code for database migration.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Compare results from Microsoft SQL Server against the application you migrated to PostgreSQL.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Perform application performance benchmarking between Microsoft SQL Server and PostgreSQL.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Revise stored procedures or inline T-SQL statements called by the application to improve performance.</td>
<td></td>
</tr>
</tbody>
</table>

The following epics provide detailed instructions for some of these conversion tasks for Python and Perl applications.
### AWS Prescriptive Guidance Patterns

**Python and Perl application changes to support database migrations**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Use a checklist for each step of the migration. | Add the following to your checklist for each step of application migration, including the final step:  
  - Review PostgreSQL documentation to ensure that all your changes are compatible with the PostgreSQL standard.  
  - Check for integer and floating values for columns.  
  - Identify the number of rows inserted, updated, and extracted, along with the column names and date/time stamps. You can use a diff utility or write a script to automate these checks.  
  - Complete performance checks for large inline SQL statements, and check the overall performance of the application.  
  - Check for correct error handling for database operations and graceful program exit by using multiple try/catch blocks.  
  - Check to ensure that proper logging processes are in place. | App developer |

### Analyze and update your application – Python code base

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Analyze your existing Python code base. | Your analysis should include the following to facilitate the application migration process:  
  - Identify all connection objects in the code.  
  - Identify all incompatible inline SQL queries (such as T-SQL statements and stored procedures) and analyze required changes.  
  - Review the documentation for your code and track the control flow to understand code functionality. This will be helpful later when | App developer |
### AWS Prescriptive Guidance Patterns
### Python and Perl application changes to support database migrations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>you test the application for performance or load comparisons.</td>
<td>• Understand the purpose of the application so you can test it effectively after database conversion. Most Python applications that are candidates for conversion with database migrations are either feeds that load data from other sources into database tables, or extractors that retrieve data from the tables and transform them into different output formats (such as CSV, JSON, or flat files) that are suitable for creating reports or for making API calls to perform validations.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Convert your database connections to support PostgreSQL.</td>
<td>Most Python applications use the pyodbc library to connect with SQL Server databases as follows.</td>
<td>App developer</td>
</tr>
</tbody>
</table>

```python
import pyodbc
....
try:
    conn_string = "Driver=ODBC Driver 17 for SQL Server;UID={};PWD={};Server={};Database={}".format(conn_user, conn_password, conn_server, conn_database)
    conn = pyodbc.connect(conn_string)
    cur = conn.cursor()
    result = cur.execute(query_string)
    for row in result:
        print (row)
except Exception as e:
    print(str(e))
```

Convert the database connection to support PostgreSQL as follows.

```python
import pyodbc
import psycopg2
....
try:
    conn_string = 'postgresql+psycopg2://'+conn_user+':'+conn_password+'@'+conn_server+'/'+'conn_database
    conn = pyodbc.connect(conn_string, connect_args={'options':'-csearch_path=dbo'})
    cur = conn.cursor()
    result = cur.execute(query_string)
    for row in result:
        print (row)
except Exception as e:
    print(str(e))
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change inline SQL queries to PostgreSQL.</td>
<td>Convert your inline SQL queries to a PostgreSQL-compatible format. For example, the following SQL Server query retrieves a string from a table.</td>
<td>App developer</td>
</tr>
<tr>
<td></td>
<td>```python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dtype = &quot;type1&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stm = &quot;&quot;SELECT TOP 1 searchcode FROM TypesTable (NOLOCK) WHERE code=&quot;&quot; + str(dtype) + &quot;&quot;&quot;&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td># For Microsoft SQL Server Database Connection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>engine = create_engine('mssql+pyodbc:///?odbc_connect=%s' % urllib.parse.quote_plus(conn_string), connect_args={'connect_timeout':login_timeout})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>conn = engine.connect()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rs = conn.execute(stm)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for row in rs: print(row)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After conversion, the PostgreSQL-compatible inline SQL query looks like the following.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dtype = &quot;type1&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stm = &quot;&quot;SELECT searchcode FROM TypesTable WHERE code=&quot;&quot; + str(dtype) + &quot;&quot; LIMIT 1&quot;&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td># For PostgreSQL Database Connection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>engine = create_engine('postgresql+psycopg2://%s' % conn_string, connect_args={'connect_timeout':login_timeout})</td>
<td></td>
</tr>
<tr>
<td></td>
<td>conn = engine.connect()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rs = conn.execute(stm)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for row in rs: print(row)</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| Handle dynamic SQL queries. | Dynamic SQL can be present in one script or in multiple Python scripts. Earlier examples showed how to use Python's string replace function to insert variables for constructing dynamic SQL queries. An alternate approach is to append the query string with variables wherever applicable. In the following example, the query string is constructed on the fly based on the values returned by a function.  

```python
query = "SELECT id from equity e join issues i on e.permId=i.permId where e.id"
query += get_id_filter(ids) + " e.id is NOT NULL"
```
These types of dynamic queries are very common during application migration. Follow these steps to handle dynamic queries:  
- Check the overall syntax (for example, the syntax for the `SELECT` statement with a `JOIN` clause).  
- Verify all variables or column names used in the query, such as `i` and `id`.  
- Check the functions, arguments, and return values used in the query (for example, `get_id_filter` and its argument `ids`). | App developer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle results sets, variables, and data frames.</td>
<td>For Microsoft SQL Server, you use Python methods such as fetchone() or fetchall() to retrieve the results set from the database. You can also use fetchmany(size) and specify the number of records to return from the results set. To do this, you can use the <code>pyodbc</code> connection object as shown in the following example.</td>
<td>App developer</td>
</tr>
</tbody>
</table>

**pyodbc (Microsoft SQL Server)**

```python
import pyodbc
server = 'tcp:myserver.database.windows.net'
database = 'exampledb'
username = 'exampleusername'
password = 'examplepassword'
conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+password+)
cursor = conn.cursor()
cursor.execute("SELECT * FROM ITEMS")
row = cursor.fetchone()
while row:
    print(row[0])
    row = cursor.fetchone()
```

In Aurora, to perform similar tasks such as connecting to PostgreSQL and fetching results sets, you can use either `psycopg2` or SQLAlchemy. These Python libraries provide the connection module and cursor object to traverse through the PostgreSQL database records, as shown in the following example.

**psycopg2 (Aurora PostgreSQL-Compatible)**

```python
import psycopg2
query = "SELECT * FROM ITEMS;"
//Initialize variables
host=dbname=user=password=port=sslmode=connect_timeout=""
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>connstring = &quot;host='{host}' dbname='{dbname}' user='{user}' \</td>
<td>connstring = \</td>
</tr>
<tr>
<td></td>
<td>password='{password}' port='{port}'&quot;.format(host=host, dbname=dbname, \</td>
<td>(Aurora</td>
</tr>
<tr>
<td></td>
<td>\ user=user, password=password, port=port) conn =</td>
<td>PostgreSQL-</td>
</tr>
<tr>
<td></td>
<td>psycopg2.connect(connstring) cursor = conn.cursor() cursor.execute(query)</td>
<td>Compatible)</td>
</tr>
<tr>
<td></td>
<td>column_names = [column[0] for column in cursor.description] print(&quot;Column</td>
<td>SQLAlchemy:</td>
</tr>
<tr>
<td></td>
<td>Names: &quot;, column_names) print(&quot;Column values: &quot;) for row in cursor:</td>
<td>(Aurora</td>
</tr>
<tr>
<td></td>
<td>print(&quot;itemid &quot;:&quot;, row[0]) print(&quot;itemdescription &quot;:&quot;, row[1]) print(&quot;</td>
<td>PostgreSQL-</td>
</tr>
<tr>
<td></td>
<td>itemprice &quot;:&quot;, row[3]) print(&quot;df=&quot;, df)</td>
<td>Compatible)</td>
</tr>
</tbody>
</table>

**SQLAlchemy: (Aurora PostgreSQL-Compatible)**

```python
from sqlalchemy import create_engine
from pandas import DataFrame
conn_string = 'postgresql://core:database=localhost:5432/exampledatabase'
engine = create_engine(conn_string)
conn = engine.connect()
dataid = 1001
result = conn.execute("SELECT * FROM ITEMS")
df = DataFrame(result.fetchall())
df.columns = result.keys()
df = pd.DataFrame()
ext
```
### AWS Prescriptive Guidance Patterns
**Python and Perl application changes to support database migrations**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Test your application during and after migration.                    | Testing the migrated Python application is an ongoing process. Because the migration includes connection object changes (psycopg2 or SQLAlchemy), error handling, new features (data frames), inline SQL changes, bulk copy functionalities (bcp instead of COPY), and similar changes, it must be tested carefully during and after application migration. Check for:  
  - Error conditions and handling  
  - Any record mismatches after migration  
  - Record updates or deletions  
  - Time required to run the application                                                                                                                          | App developer      |

### Analyze and update your application – Perl code base

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Analyze your existing Perl code base.                               | Your analysis should include the following to facilitate the application migration process. You should identify:  
  - Any INI or configuration-based code  
  - Database-specific standard Open Database Connectivity (ODBC) Perl drivers or any customized drivers  
  - Code changes required for inline and T-SQL queries  
  - Interactions among various Perl modules (for example, a single Perl ODBC connection object that is called or used by multiple functional components)  
  - Dataset and results set handling  
  - External, dependent Perl libraries  
  - Any APIs that are used in the application                                                                                                                  | App developer      |
Perl version compatibility and driver compatibility with Aurora PostgreSQL-Compatible

Convert the connections from the Perl application and DBI module to support PostgreSQL.

Perl-based applications generally use the Perl DBI module, which is a standard database access module for the Perl programming language. You can use the same DBI module with different drivers for SQL Server and PostgreSQL.

For more information about required Perl modules, installations, and other instructions, see the DBD::Pg documentation. The following example connects to Aurora PostgreSQL-Compatible at exampletest-aurorapg-database-cluster-samplencluster.us-east-.rds.amazonaws.com.

```perl
#!/usr/bin/perl
use DBI;
use strict;
my $driver = "Pg";
my $hostname = "exampletest-aurorapg-database-cluster-samplencluster.us-east.rds.amazonaws.com"
my $dsn = "DBI:$driver:dbname = $hostname;host = 127.0.0.1;port = 5432";
my $username = "postgres";
my $password = "pass123";
$dbh = DBI->connect("dbi:Pg:dbname=$hostname;host=$host;port=$port;options=$options", $username, $password, {AutoCommit => 0, RaiseError => 1, PrintError => 0});
```

Skills required

- App developer
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Change Inline SQL queries to PostgreSQL. | Your application might have inline SQL queries with `SELECT`, `DELETE`, `UPDATE`, and similar statements that include query clauses that PostgreSQL doesn't support. For example, query keywords such as `TOP` and `NOLOCK` aren't supported in PostgreSQL. The following examples show how you can handle `TOP`, `NOLOCK`, and Boolean variables.  

In SQL Server:  

```sql
$sqlStr = $sqlStr . 
  "WHERE a.student_id in
  (SELECT TOP $numofRecords
   c_student_id  
   FROM active_student_record b
   WITH (NOLOCK) 
   INNER JOIN
   student_contributor c
   WITH (NOLOCK) on
   c.contributor_id = b.c_student_contr_id)"
```

For PostgreSQL, convert to:  

```sql
$sqlStr = $sqlStr . 
  "WHERE a.student_id in
  (SELECT TOP $numofRecords
   c_student_id  
   FROM active_student_record b
   INNER JOIN
   student_contributor c
   on c.contributor_id = b.c_student_contr_id WHERE
   b_current_1 is true
   LIMIT $numofRecords)"
```
AWS Prescriptive Guidance Patterns
Python and Perl application changes
to support database migrations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle dynamic SQL queries and Perl variables.</td>
<td>Dynamic SQL queries are SQL statements that are built at application runtime. These queries are constructed dynamically when the application is running, depending on certain conditions, so the full text of the query isn’t known until runtime. An example is a financial analytics application that analyzes the top 10 shares on a daily basis, and these shares change every day. The SQL tables are created based on top performers, and the values aren't known until runtime. Let's say that the inline SQL queries for this example are passed to a wrapper function to get the results set in a variable, and then a variable uses a condition to determine whether the table exists: • If the table exists, don't create it; do some processing. • If the table doesn't exist, create the table and also do some processing. Here's an example of variable handling, followed by the SQL Server and PostgreSQL queries for this use case.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>App developer</td>
</tr>
</tbody>
</table>

```perl
my $tableexists = db_read( arg 1, $sql_qry, undef, 'writer');
my $table_already_exists = $tableexists->[0]{table_exists};
if ($table_already_exists){
  # do some thing
} else {
  # do something else
}
```

**SQL Server:**

```sql
my $sql_qry = "SELECT OBJECT_ID('$backendTable',
```
The following example uses a Perl variable in inline SQL, which runs a SELECT statement with a JOIN to fetch the primary key of the table and position of the key column.

**PostgreSQL:**

```perl
my $sql_qry = "SELECT
    TO_REGCLASS('$backendTable', 'U') table_exists", undef,
    'writer')";
```

**SQL Server:**

```perl
my $sql_qry = "SELECT
    column_name',
    character_maxi mum_length \
FROM
    INFORMATION_SCHEMA.COLUMNS \
WHERE
    TABLE_SCHEMA='$example_schemaInfo'
    AND
    TABLE_NAME='$example_table'
    AND DATA_TYPE IN
    ('varchar','nvarchar');"
```

**PostgreSQL:**

```perl
my $sql_qry = "SELECT
    c1.column_name,
    c1.ordinal_position \
FROM
    information_schema.key_column_usage AS c LEFT \n    JOIN
    information_schema.table_constraints AS t1 \n    ON t1.constraint_name = c1.constraint_name \n    WHERE t1.table_name = '$example_schemaInfo'.'$example_table'
    AND t1.constraint_type = 'PRIMARY KEY' ;"
```

---

AWS Prescriptive Guidance Patterns
Python and Perl application changes to support database migrations

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>('U') table_exists&quot;, undef, 'writer')&quot;;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>my $sql_qry = &quot;SELECT TO_REGCLASS('$backendTable', 'U') table_exists&quot;, undef, 'writer')&quot;;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The following example uses a Perl variable in inline SQL, which runs a SELECT statement with a JOIN to fetch the primary key of the table and position of the key column.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL Server:</td>
<td></td>
</tr>
</tbody>
</table>
|     | my $sql_qry = "SELECT column_name', character_maxi mum_length \
FROM INFORMATION_SCHEMA.COLUMNS \
WHERE TABLE_SCHEMA='$example_schemaInfo' 
AND TABLE_NAME='$example_table' 
AND DATA_TYPE IN ('varchar','nvarchar');"; |                  |
|     | PostgreSQL: |                  |
|     | my $sql_qry = "SELECT c1.column_name, c1.ordinal_position \
FROM information_schema.key_column_usage AS c LEFT \nJOIN information_schema.table_constraints AS t1 \nON t1.constraint_name = c1.constraint_name \nWHERE t1.table_name = '$example_schemaInfo'.'$example_table' 
AND t1.constraint_type = 'PRIMARY KEY' ;"; |                  |
Make additional changes to your Perl-based or Python-based application to support PostgreSQL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Convert additional SQL Server constructs to PostgreSQL. | The following changes apply to all applications, regardless of programming language.  
  • Qualify database objects that your application uses with new and appropriate schema names.  
  • Handle LIKE operators for case-sensitive matching with the collation feature in PostgreSQL.  
  • Handle unsupported database specific functions such as DATEDIFF, DATEADD, GETDATE, CONVERT, and CAST operators. For equivalent PostgreSQL-compatible functions, see Native or built-in SQL functions in the Additional information (p. 1309) section.  
  • Handle Boolean values in comparison statements.  
  • Handle return values from functions. These could be record sets, data frames, variables, and Boolean values. Handle these according to the requirements of your application and to support PostgreSQL.  
  • Handle anonymous blocks (such as BEGIN TRAN) with new, user-defined PostgreSQL functions.  
  • Convert bulk inserts for rows. The PostgreSQL equivalent of the SQL Server bulk copy (bcp) utility, which is called from inside the application, is COPY.  
  • Convert column concatenation operators. SQL Server uses + for string concatenation, but PostgreSQL uses ||. | App developer |
Improve performance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take advantage of AWS services to make performance enhancements.</td>
<td>When you migrate to the AWS Cloud, you can refine your application and database design to take advantage of AWS services. For example, if the queries from your Python application, which is connected to an Aurora PostgreSQL-Compatible database server, is taking more time than your original Microsoft SQL Server queries, you could consider creating a feed of historical data directly to an Amazon Simple Storage Service (Amazon S3) bucket from the Aurora server, and use Amazon Athena-based SQL queries to generate reports and analytic data queries for your user dashboards.</td>
<td>App developer, Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Perl
- Perl DBI Module
- Python
- psycopg2
- SQLAlchemy
- Bulk Copy - PostgreSQL
- Bulk Copy - Microsoft SQL Server
- PostgreSQL
- Working with Amazon Aurora PostgreSQL

Additional information

Both Microsoft SQL Server and Aurora PostgreSQL-Compatible are ANSI SQL-compliant. However, you should still be aware of any incompatibilities in syntax, column data types, native database-specific functions, bulk inserts, and case sensitivity when you migrate your Python or Perl application from SQL Server to PostgreSQL.

The following sections provide more information about possible inconsistencies.

Data type comparison

Data type changes from SQL Server to PostgreSQL can lead to significant differences in the resulting data that applications operate on. For a comparison of data types, see the table on the Sqlines website.

Native or built-in SQL functions
The behavior of some functions differs between SQL Server and PostgreSQL databases. The following table provides a comparison.

<table>
<thead>
<tr>
<th>Microsoft SQL Server</th>
<th>Description</th>
<th>PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAST</td>
<td>Converts a value from one data type to another.</td>
<td>PostgreSQL type :: operator</td>
</tr>
<tr>
<td>GETDATE()</td>
<td>Returns the current database system date and time, in a YYYY-MM-DD hh:mm:ss.mmm format.</td>
<td>CLOCK_TIMESTAMP</td>
</tr>
<tr>
<td>DATEADD</td>
<td>Adds a time/date interval to a date.</td>
<td>INTERVAL expression</td>
</tr>
<tr>
<td>CONVERT</td>
<td>Converts a value to a specific data format.</td>
<td>TO_CHAR</td>
</tr>
<tr>
<td>DATEDIFF</td>
<td>Returns the difference between two dates.</td>
<td>DATE_PART</td>
</tr>
<tr>
<td>TOP</td>
<td>Limits the number of rows in a SELECT results set.</td>
<td>LIMIT/FETCH</td>
</tr>
</tbody>
</table>

Anonymous blocks

A structured SQL query is organized into sections such as declaration, executables, and exception handling. The following table compares the Microsoft SQL Server and PostgreSQL versions of a simple anonymous block. For complex anonymous blocks, we recommend that you call a custom database function within your application.

<table>
<thead>
<tr>
<th>Microsoft SQL Server</th>
<th>PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>my $sql_qry1=</td>
<td>my $sql_qry1=</td>
</tr>
<tr>
<td>my $sql_qry2 =</td>
<td>my $sql_qry2 =</td>
</tr>
</tbody>
</table>
| my $sqlqry = "BEGIN TRAN $sql_qry1 $sql_qry2 if @\@error !=0 ROLLBACK TRAN else COMIT TRAN"; | my $sql_qry = "DO $$
BEGIN $header_sql $content_sql END $$"; |

Other differences

- **Bulk inserts of rows**: The PostgreSQL equivalent of the Microsoft SQL Server bcp utility is COPY.
- **Case sensitivity**: Column names are case-sensitive in PostgreSQL, so you have to convert your SQL Server column names to lowercase or uppercase. This becomes a factor when you extract or compare data, or place column names in results sets or variables. The following example identifies columns where values might be stored in uppercase or lowercase.

```sql
my $sql_qry = "SELECT $record_id FROM $exampleTable WHERE LOWER($record_name) = 'failed transaction";```

1310
• **Concatenation:** SQL Server uses `+` as an operator for string concatenation, whereas PostgreSQL uses `||`.

• **Validation:** You should test and validate inline SQL queries and functions before you use them in application code for PostgreSQL.

---

### Migration patterns by workload

#### Topics

- IBM (p. 1311)
- Microsoft (p. 1311)
- N/A (p. 1312)
- Open-source (p. 1312)
- Oracle (p. 1312)
- SAP (p. 1313)

---

### IBM

- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 836)
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 (p. 1182)
- Move mainframe files directly to Amazon S3 using Transfer Family (p. 779)

---

### Microsoft

- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition (p. 1291)
- Configure SAML SSO from Microsoft Azure AD to CloudEndure Migration (p. 752)
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python (p. 760)
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account (p. 984)
- Install CloudEndure Agent through AWS Systems Manager on Windows and Linux (p. 768)
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS (p. 1155)
- Migrate a Microsoft Azure VM to Amazon EC2 using CloudEndure (p. 1000)
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS (p. 816)
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT (p. 1257)
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk (p. 1193)
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 (p. 1067)
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using CloudEndure (p. 1073)
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server (p. 1245)
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using Amazon S3 and SSMS (p. 1249)
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers (p. 1253)
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS (p. 940)
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents (p. 944)
• Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux (p. 1242)
• Migrate data from Microsoft Azure Blob storage and Backblaze B2 Cloud Storage to Amazon S3 by using Rclone (p. 1159)
• Migrate Windows SSL certificates to an Application Load Balancer using ACM (p. 1150)

N/A

• Create an approval process for firewall requests during a rehost migration to AWS (p. 981)

Open-source

• Migrate an on-premises Linux server to an Amazon EC2 Linux instance using AWS SMS (p. 991)
• Migrate an on-premises MariaDB database to Amazon EC2 using MySQL tools (p. 1081)
• Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools (p. 1264)
• Migrate an on-premises MySQL database to Amazon EC2 (p. 1086)
• Migrate an on-premises MySQL database to Amazon RDS for MySQL (p. 774)
• Migrate an on-premises MySQL database to Aurora MySQL (p. 953)
• Migrate an on-premises PostgreSQL database to Amazon EC2 (p. 1048)
• Migrate an on-premises PostgreSQL database to Amazon RDS for PostgreSQL (p. 1231)
• Migrate an on-premises PostgreSQL database to Aurora PostgreSQL (p. 1237)
• Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling (p. 1188)
• Migrate from Oracle GlassFish to AWS Elastic Beanstalk (p. 1026)
• Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical (p. 1228)
• Migrate on-premises Java applications to AWS using AWS App2Container (p. 1278)
• Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3 (p. 1268)
• Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible Edition (p. 806)
• Restart the CloudEndure Agent automatically without disabling SELinux after rebooting a RHEL source server (p. 787)
• Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport (p. 1124)

Oracle

• Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT (p. 792)
• Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation (p. 797)
• Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication (p. 1100)
• Migrate an Amazon RDS for Oracle DB instance to another VPC (p. 1111)
• Migrate an on-premises Oracle database to Amazon EC2 using CloudEndure (p. 1038)
• Migrate an on-premises Oracle database to Amazon EC2 using Oracle Data Pump (p. 1045)
• Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash (p. 1214)
• Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 876)
• Migrate an on-premises Oracle database to Amazon RDS for Oracle (p. 1219)
• Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump (p. 1224)
• Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL using an Oracle bystander and AWS DMS (p. 880)
• Migrate an on-premises Oracle database to Amazon Redshift using AWS SCT data extraction agents (p. 899)
• Migrate an on-premises Oracle database to Oracle on Amazon EC2 (p. 1031)
• Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT (p. 870)
• Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS (p. 1210)
• Migrate an Oracle Database to Amazon Aurora MySQL using AWS DMS and AWS SCT (p. 902)
• Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT (p. 891)
• Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT (p. 908)
• Migrate data from an on-premises Oracle database to Aurora PostgreSQL (p. 915)
• Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL (p. 830)
• Migrate from an on-premises Oracle database or Amazon RDS for Oracle to Amazon DynamoDB using AWS DMS and AWS SCT (p. 865)
• Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS (p. 847)
• Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS (p. 840)
• Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT (p. 854)
• Migrate from Oracle to Amazon DocumentDB using AWS DMS (p. 860)
• Migrate from Oracle to Amazon RDS for PostgreSQL using Oracle GoldenGate (p. 888)
• Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS (p. 1204)
• Migrate Oracle native functions to PostgreSQL using extensions (p. 811)
• Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible (p. 958)
• Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL (p. 969)

SAP

• Migrate an on-premises SAP ASE database to Amazon EC2 (p. 1054)
• Migrate an SAP ASE database to Aurora MySQL using AWS DMS (p. 926)
• Migrate an SAP ASE database to Aurora PostgreSQL using AWS DMS (p. 931)
• Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS (p. 922)
• Migrate from SAP ASE to PostgreSQL on Amazon EC2 using AWS DMS (p. 936)
• Migrate SAP workloads to an SAP database on Amazon EC2 using CloudEndure (p. 1059)

More patterns

• Assess application readiness for migration to the AWS Cloud by using CAST Highlight (p. 1321)
• Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode (p. 559)
• Connect to AWS MGN data and control planes over a private network (p. 1523)
• Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL (p. 19)
• Convert the Teradata RESET WHEN feature to Amazon Redshift SQL (p. 25)
• Copy Amazon DynamoDB tables across accounts using AWS Backup (p. 297)
• Copy data from an S3 bucket in one account and Region to another account and Region (p. 100)
• Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing (p. 587)
• Deploy multiple-stack applications using AWS CDK with TypeScript (p. 1408)
• Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator (p. 286)
• Migrate SAP HANA to AWS using SAP HSR with the same hostname (p. 336)
• Migrate SQL Server to AWS using distributed availability groups (p. 346)
• Migrate VMs to VMware Cloud on AWS by using HCX OS Assisted Migration (p. 575)
• Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS (p. 1538)
• Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud (p. 1551)
• Simplify private certificate management by using ACM Private CA and AWS RAM (p. 1721)
Modernization

Topics

- Analyze and visualize software architecture in CAST Imaging (p. 1315)
- Assess application readiness for migration to the AWS Cloud by using CAST Highlight (p. 1321)
- Automatically archive items to Amazon S3 using DynamoDB TTL (p. 1334)
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager (p. 1345)
- Build a multi-tenant serverless architecture in Amazon OpenSearch Service (p. 1359)
- Cache secrets using AWS Lambda extensions (p. 1391)
- Convert and unpack data from EBCDIC to ASCII (p. 1398)
- Deploy multiple-stack applications using AWS CDK with TypeScript (p. 1408)
- Automate deployment of nested applications using AWS SAM (p. 1416)
- Implement SaaS tenant isolation for Amazon S3 by using an AWS Lambda token vending machine (p. 1434)
- Implement the serverless saga pattern by using AWS Step Functions (p. 1448)
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK (p. 1459)
- Modernize ASP.NET Web Forms applications on AWS (p. 1466)
- Run event-driven and scheduled workloads at scale with AWS Fargate (p. 1475)
- Set up CI/CD for AWS AppSync GraphQL API updates (p. 1482)
- Decompose monoliths into microservices by using CQRS and event sourcing (p. 1491)
- More patterns (p. 1508)

Analyze and visualize software architecture in CAST Imaging

Created by Arpita Sinha (Cast Software)

| Environment: Production | Technologies: Modernization | Workload: All other workloads |

Summary

You can use CAST Imaging to navigate a complex software system visually, perform a precise analysis of the software structure, and make decisions about your application’s architecture, especially for modernization purposes. This pattern describes the steps for onboarding source code to CAST Application Intelligence Platform (CAST AIP) through CAST AIP Console to generate application data. The data is then published to CAST Imaging, where you can visualize and navigate your application architecture layer by layer.

Prerequisites and limitations

Prerequisites
• An active AWS account.
• An Amazon Machine Image (AMI) for Microsoft Windows Server Base or Core (see Windows AMIs) for CAST Imaging on Amazon Elastic Compute Cloud (Amazon EC2) with 32 GB RAM and 100 GB SSD.
• A CAST Extend account for accessing downloads.
• CAST AIP and CAST Imaging license keys. You can obtain these by contacting CAST at c.yarlagadda@castsoftware.com.
• The complete source code of the application you want to analyze, in extracted and compressed (.zip) format.
• Compliance with AIP Console deployment prerequisites.

Limitations
• The CAST Web installer is available only for the Microsoft Windows operating system, and installs CAST AIP version 8.3.25 or later.
• CAST Extend doesn’t support Internet Explorer.

Product versions
• The CAST Web Installer (available from CAST Extend) automatically downloads the latest version of CAST AIP Core, CAST Console, CAST Storage Service, and CAST Imaging.

Architecture
The following workflow illustrates the onboarding of application source code to CAST AIP through AIP Console at a high level. CAST AIP generates application source code metadata by reverse-engineering front-end, middleware, and back-end code. The application data that is generated by CAST AIP is automatically imported into CAST Imaging for investigation purposes.

Here’s a snapshot of how this process works in AIP Console and CAST Imaging.
Tools

- **CAST Web Installer** – Installs all required software in one step, on a single machine, with minimum configuration required.

Epics

**Set up the CAST AIP environment**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the CAST Web Installer.</td>
<td>Log in to CAST Extend and search for Webinstaller. (If you need to)</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
</tbody>
</table>
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the installer.</td>
<td>The CAST Web Installer (CASTWebInstaller.exe) installs the following components: CAST AIP Core, AIP Console and AIP Node package, CAST Storage Service for Windows (customized PostgreSQL RDBMS), and CAST Imaging for Windows. When the CAST Web Installer finishes the installation, a Readme window displays a summary of the installation.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Configure CAST AIP Console to publish results into CAST Imaging.</td>
<td><strong>Log in to AIP Console.</strong> Go to Administration Center, System Settings, Imaging Settings to fully automate data import into CAST Imaging as part of the standard onboarding and rescan processes. This will ensure that when analysis processes are complete, all data is ready to use in CAST Imaging without requiring any additional manual actions.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
</tbody>
</table>

### Add an application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the AIP Console home page.</td>
<td>Open the AIP Console home page at http://&lt;server&gt;:8081/ui/index.html, and enter your CAST credentials. For help, see the Login section of the AIP Console documentation.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Prepare your source code.</td>
<td>Before you begin the onboarding process, make sure that your source code is prepared in appropriate format in a single .zip file.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Add a new application in AIP Console.</td>
<td>Choose Add application, enter the application name, and then</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
</tbody>
</table>
## Add and analyze your source code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the Delivery wizard.</td>
<td>Find the application that you created, and then choose <strong>Add version</strong>.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Add the source code.</td>
<td>Drag and drop the source code (.zip file) into the wizard, or choose the upload cloud icon to add a file by using the &quot;open file&quot; method.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Start the analysis process.</td>
<td>In the Delivery wizard, provide the version details and specify the configuration options as explained in the <strong>Standard onboarding</strong> section of the CAST Imaging documentation. Make sure that the <strong>Publish to CAST Imaging</strong> option is selected, and then choose <strong>Proceed</strong>. This will start the analysis process for the source code. The progress window in AIP Console shows each step, and displays a notification when analysis is complete.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
</tbody>
</table>

## Verify results and data published to CAST Imaging

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check status and logs</td>
<td>When all actions have been completed, you should check the Progress window for a &quot;success&quot; message. You can check the individual logs for all the actions that have been performed immediately after they have completed. To check the logs, you can click the View log option in the Progress window.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Check application details</td>
<td>Use the <strong>Application details panel</strong> to view information about</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
</tbody>
</table>

choose **Add**. You can see the progress of each step in AIP Console. For details, see the **Add a new application** section of the AIP Console documentation.
### Verify and access CAST Imaging

- **Task**: Verify and access CAST Imaging
- **Description**: In the Application Management screen of AIP Console, verify that the version status of your application is **imaging processed**. This means that the source code has been analyzed and uploaded to your CAST Imaging instance and you will see an Imaging icon with your version. Use the CAST Imaging icon to take you directly to the specific application data in CAST Imaging.
- **Skills required**: Software architects, Developers, Technical leaders

### Get started with CAST Imaging

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log in to CAST Imaging.</td>
<td>Open <a href="http://localhost:8083">http://localhost:8083</a>, enter your CAST administrator credentials, and select your application. For help, see the Login section of the CAST Imaging documentation.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Set up the imaging environment.</td>
<td>Configure the CAST Imaging environment through Admin Center to manage application data, user permissions, and other preferences. For more information, see the Configuration options section of the CAST Imaging documentation.</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
<tr>
<td>Verify that the application is available in CAST Imaging.</td>
<td>Open the Admin Center - Application management panel in CAST Imaging Admin Center and verify that the analyzed application is available.</td>
<td></td>
</tr>
<tr>
<td>Explore CAST Imaging features.</td>
<td>Use the Help icon to display the CAST Imaging Helper. This option provides a tutorial that helps you get started viewing your software architecture in CAST Imaging. For more information about these</td>
<td>Software architects, Developers, Technical leaders</td>
</tr>
</tbody>
</table>
CAST Highlight is a software as a service (SaaS) solution for performing rapid application portfolio analysis. This pattern describes how to configure and use CAST Highlight to assess the cloud readiness of custom software applications across an organization's IT portfolio, and to plan for modernization or migration to the Amazon Web Services (AWS) Cloud.

CAST Highlight generates insights into an application's cloud readiness, identifies code blockers that need to be removed before a migration, estimates the effort to remove these blockers, and recommends AWS services that individual applications could use after the migration.
This pattern describes the procedure for setting up and using CAST Highlight, which consists of five steps: new user setup, application management, campaign management, source code analysis, and results analysis. You must complete all the steps in the Epics section of this pattern to ensure a successful application scan and analysis.

Prerequisites and limitations

Prerequisites

- An active CAST Highlight account with Portfolio Manager permissions.
- At least 300 MB free disk space and 4 GB memory on your local computer to install the CAST Highlight Local Agent.
- Microsoft Windows 8 or later.
- Your application source code must be stored in text files that are accessible from the machine where the Local Agent is installed. No source code leaves the premises and all code is locally scanned.

Architecture

The following diagram illustrates the workflow for using CAST Highlight.

The workflow consists of the following steps:

1. Log in to the CAST Highlight portal, download the Local Agent, and install it on your local computer. Amazon Simple Storage Service (Amazon S3) stores the Local Agent installation package.
2. Scan your source code files and produce a results file.
3. Upload the results file to the CAST Highlight portal. **Important:** No source code is included in the results file.
4. Answer survey questions for each application that you scanned.
5. View the dashboards and reports available in the CAST Highlight portal. Amazon Relational Database Service (Amazon RDS) stores the code scan, analysis results, and the CAST Highlight software data.

**Technology stack**

CAST Highlight supports the following technologies to analyze application cloud readiness:

- Java
- COBOL
- C#
- PHP
- JavaScript
- TypeScript
- Python
- Microsoft Transact-SQL
- VB.Net
- Kotlin
- Swift

**Automation and scale**

- A **CLI analyzer** can be used to automate the CAST Highlight analysis process.

**Tools**

No tools are required for this pattern if all the prerequisites are met. However, you can choose to use optional tools such as source code management (SCM) utilities, code extractors, or other tools to manage your source code files.

**Epics**

**New user setup**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate your CAST Highlight account and choose your password.</td>
<td>All first-time CAST Highlight users receive an account activation email. Follow the activation link to activate your CAST Highlight account and enter a password to complete the activation process.</td>
<td>N/A</td>
</tr>
<tr>
<td>Log in to the CAST Highlight portal.</td>
<td>The CAST Highlight homepage appears after you enter your new password. Log in to the</td>
<td>N/A</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>CAST Highlight portal with your user credentials.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Application management**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an application record.</td>
<td>In the CAST Highlight portal, navigate to the Manage Application tab in the Manage Portfolio section and choose + Create Application.</td>
<td>N/A</td>
</tr>
<tr>
<td>Choose an application name.</td>
<td>Enter a name for your application. This name is used for your application record in CAST Highlight.</td>
<td>N/A</td>
</tr>
<tr>
<td>Repeat the steps for all applications.</td>
<td>Repeat these steps for each application that you want to scan.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Campaign management**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a campaign.</td>
<td>CAST Highlight uses &quot;campaign&quot; to describe a set of applications that will be analyzed at a specific time. In the CAST Highlight portal, navigate to the Manage Campaigns tab in the Manage Portfolio section. Choose Create Campaign to launch the campaign creation screen.</td>
<td>N/A</td>
</tr>
<tr>
<td>Enter a name and choose a closing date for the campaign.</td>
<td>Enter a name for your campaign and choose a closing date for it. Important: Contributors cannot submit application analysis results after the campaign's closing date.</td>
<td>N/A</td>
</tr>
<tr>
<td>Decide to include source code scan, survey answers, and domain and application scope.</td>
<td>Choose one or more of the standard surveys that are used to enhance the source code analysis data with qualitative information. The survey categories are Business impact, Software maintenance effort, CloudReady, and Application</td>
<td>N/A</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>Choose the domain and applications that are analyzed during the campaign.</td>
<td></td>
</tr>
<tr>
<td><strong>Important</strong></td>
<td>Make sure that you add all applications you want to scan in the Manage Applications section before you begin the campaign.</td>
<td></td>
</tr>
<tr>
<td>Customize the launch message.</td>
<td>Customize the launch message that will be sent by email to all contributors associated with the applications in the campaign.</td>
<td>N/A</td>
</tr>
<tr>
<td>Launch the campaign.</td>
<td>Choose Complete to launch the campaign.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Source code analysis**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the CAST Highlight Local Agent.</td>
<td>In the CAST Highlight portal, choose Application Scans and download the Local Agent to your local computer.</td>
<td>N/A</td>
</tr>
<tr>
<td>Install the Local Agent.</td>
<td>Launch the CASTHighlightSetup.exe installation program and follow the setup instructions that appear. After the Local Agent is installed, you are ready to analyze your applications.</td>
<td>N/A</td>
</tr>
<tr>
<td>Define the scope of the Local Agent code scan.</td>
<td>The code analysis is performed at the file level and doesn't consider the logical links or dependencies among files. All files are considered equal and part of the application. To provide accurate and consistent results, prepare your code scan scope by using the file or folder exclusion features available in the Local Agent.</td>
<td>N/A</td>
</tr>
<tr>
<td>Include open-source or COTS packages.</td>
<td>(Optional) If you want to include open-source or commercial off-the-shelf (COTS) packages, make sure they're included in the folders that you're planning to scan. Typically, external libraries are grouped in a subfolder called</td>
<td>N/A</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>&quot;third-party&quot; or something similar, and the main code is often located in the &quot;src/main&quot; file folder.</td>
<td>N/A</td>
</tr>
<tr>
<td>Exclude test classes.</td>
<td>Test classes are typically excluded from the source code analysis because they are generally not part of the compiled application. However, you can choose to include them in the scan if required.</td>
<td>N/A</td>
</tr>
<tr>
<td>Exclude SCM, build, and deployment folders.</td>
<td>For more consistent results, you should avoid including SCM, build, or deployment folders (for example, .git or .svn files) in your scan.</td>
<td>N/A</td>
</tr>
<tr>
<td>Include dependency files.</td>
<td>If you want insights into frameworks and dependencies whose physical files are not part of the folder you're scanning, then make sure you include the dependency files (such as pom.xml, build.gradle, package.json, or .vcsproj files).</td>
<td>N/A</td>
</tr>
<tr>
<td>Invoke the Local Agent.</td>
<td>Run the Local Agent on your local Windows machine.</td>
<td>N/A</td>
</tr>
<tr>
<td>Choose the folder that contains your source code.</td>
<td>Choose the folder that contains your source code. You can add multiple folders to be discovered by the Local Agent. Although the Local Agent does support source discovery through network paths, you should make sure source folders are located on your local machine. <strong>Important:</strong> We recommend running multiple scans if there are more than 10,000 files in your source folders.</td>
<td>N/A</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Start file discovery.         | On the Local Agent dashboard, choose Discover Files. The Local Agent discovers files in your folders and subfolders, and detects their technologies. You can choose the Cancel button to cancel the discovery at any time.  
   
   After the file discovery finishes, the Local Agent lists the folders and files that were found. The Technologies column shows associated technologies and file count. The Path column shows the location of the folders and files. | N/A             |
| Refine the source code scan configuration. | (Optional) To refine the Local Agent scan, you can deactivate one or more technologies for a specific folder or file. If all technologies are deactivated, your folder or file will be excluded from the scan's scope.  
   
   To deactivate technologies, choose the yellow label of the technology you want to deactivate. You can also choose the filter icon when hovering over a file or a folder to associate a technology with a specific file or folder. These settings are saved and speed up the discovery process for the folder or file. | N/A             |
<p>| Start source code scan.       | After you configure your scan, choose &quot;Scan Files&quot; to begin the scanning process.                                                                                                                         | N/A             |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Check for green or grey labels. | After the source code scan is complete, a status label is shown at folder and file levels.  
A green label means that files were correctly scanned with the associated technology.  
A grey label means that files were not scanned and are excluded. The reason for their exclusion is shown when you hover over the label of each file. Possible reasons for file exclusion include binary files, unreadable files, missing files, external library, encoded files, generated files, syntax errors, content that isn’t in the expected language, code that isn’t compliant with enough analysis criteria, files that exceed the size limit (10 MB), timeout issues, or unavailability of the analyzer. | N/A             |
<p>| Modify the scan configuration and scan code again. | (Optional) You can modify your scan configuration settings and choose <strong>Scan Files</strong> to scan the files again.                                                                                               | N/A             |
| Confirm scan results.           | Choose <strong>Confirm Results</strong> if the scan results meet your requirements.                                                                                                                                         | N/A             |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>View frameworks and software libraries found by the Local Agent.</td>
<td>View the frameworks and software libraries used or referenced by your applications, and discovered by the Local Agent during the code scan. The first group lists frameworks that are officially referenced and discovered by the Local Agent. The second group lists potential relevant frameworks and libraries based on exploiting configuration files in your source code folders (for example, Maven pom.xml, build.gradle, .vcproj, or .json dependency files). You can keep or ignore elements from these lists by choosing their individual switch button. <strong>Important</strong>: If a framework is turned off, it is not listed in the CAST Highlight portal or attached to your application.</td>
<td>N/A</td>
</tr>
<tr>
<td>Manually add a framework or library.</td>
<td>(Optional) You can manually add a framework or a library that is not referenced by CAST Highlight. In the Local Agent, choose the + icon and provide the technology, name, version number, functional type, and license type (for example, MIT, Apache, or LGPL). Choose <strong>Confirm Frameworks</strong> to add them to your results.</td>
<td>N/A</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Save code scan results.</td>
<td>The Local Agent displays a summary of your code scan results grouped by technology. Choose Save Results and specify the folder you want the results to be saved to. The Local Agent generates one .zip file per scan, which contains all the analysis results. Depending on the number of distinct technologies and root source folders, the Local Agent automatically generates one or several .csv files with the <code>FolderName.Technology.date.csv</code> naming structure.</td>
<td>N/A</td>
</tr>
<tr>
<td>Upload the code scan results to the CAST Highlight portal.</td>
<td>In the CAST Highlight portal, choose the applications you analyzed in the Application Scans section. Choose Upload Results and choose the .csv files. You can also upload the .csv files individually. After each file is uploaded, a record of the upload appears on your screen.</td>
<td>N/A</td>
</tr>
<tr>
<td>Delete analysis result files, if required.</td>
<td>(Optional) An analysis results file can be deleted at any time during the upload process by choosing the trash can icon. Important: Only users with Portfolio Manager privileges or the contributor who uploaded the results can delete the results.</td>
<td>N/A</td>
</tr>
<tr>
<td>Answer application survey.</td>
<td>A Survey button appears on applications that require a survey. Choose Survey, answer the questions for each section of the survey, and choose Submit after you finish. The progress of your survey is displayed at the top of your screen. You are able to submit your results after all mandatory information is submitted. However, you can enrich the data in your organization's CAST Highlight instance by answering all the questions.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Submit code scan results.

**Task**: Submit code scan results.

**Description**: After you upload all the .csv result files for the application and complete the survey questions, choose **Submit** in the **Application Scans** section. This step is required to complete the process and ensure that the results are available in the CAST Highlight portal.

**Skills required**: N/A

### Results analysis

#### View CAST Highlight portal homepage.

**Task**: View CAST Highlight portal homepage.

**Description**: The CAST Highlight portal homepage includes tiles that have high-level information about your application portfolio, such as software health, CloudReady, and open-source safety scores for your entire portfolio. The homepage also includes the number of onboarded applications. For more information about the CAST Highlight metrics definitions and measurement methodology, see [CAST Highlight – Metrics and methodology (Microsoft PowerPoint presentation)].

**Skills required**: N/A

#### View the CloudReady dashboard.

**Task**: View the CloudReady dashboard.

**Description**: Choose the **CloudReady** tile to open the CloudReady dashboard. This is the primary portfolio-level dashboard for assessing the cloud readiness of your applications. It helps you plan and develop a portfolio roadmap for your cloud migration.

#### Identify cloud migration roadblocks.

**Task**: Identify cloud migration roadblocks.

**Description**: The CloudReady dashboard shows all onboarded applications as bubbles on a chart. The chart has **Business impact** on the vertical axis, which is based on the answers to the business impact survey about each application.

On the horizontal axis is **Cloud readiness**, which is based on a combination of code analysis.

**Skills required**: N/A
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>data and the CloudReady survey answers for each application. The bubble size represents the number of cloud migration roadblocks found in the source code analysis. This can be changed if you choose other options from the dropdown menu, such as lines of code and full-time equivalent employees (FTEs).</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| Perform portfolio segmentation. | You can perform a portfolio segmentation with the following segment filters: Core Cloud, Quick Wins, Long-Term Bets and Pursue Later.  
- The Core Cloud segment shows applications that could be modernized and refactored for platform as a service (PaaS).  
- The Quick Wins segment shows applications that might be good places to start your migration.  
- The Long-Term Bets segment shows applications that could initially be rehosted to infrastructure as a service (IaaS).  
- The Pursue Later segment shows applications that could be replaced or retired. | N/A |
| Choose an application to analyze. | On the CloudReady dashboard, choose an application bubble to analyze that application. Choose the name of the application in the table after the bubble chart to begin the deeper analysis. Different dashboards are available to analyze individual applications, such as Code Insights (software health patterns), Trends, and Software Composition (open-source risks). | N/A |
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analyze the CloudReady results of an individual application.</strong></td>
<td>Choose the <strong>CloudReady</strong> tab, which shows the application's overall CloudReady score. This score is a weighted average based on a combination of the CloudReady survey answers and the CloudReady code scan. The answers to the survey questions appear in the table below the tiles. Choose <strong>CloudReady Code Scan</strong> to view the code scan results. There is a list of CloudReady patterns that the application code was scanned for. This list includes the following columns:</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• <strong>Cloud Requirement</strong> is the specific code pattern.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Technology</strong> is the pattern’s programming language. “Impact” is the pattern’s impact on the application (C = code, F = framework, A = architecture).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Criticality</strong> is the level of importance of addressing this pattern before migrating.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Contribution</strong> is how this pattern contributes to the overall CloudReady score. If the pattern is green, it is a booster and increases the CloudReady score. If the pattern is red, it is a blocker and decreases the CloudReady score. If the pattern has no color, it is a blocker that was not detected and increases the CloudReady score.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Roadblocks</strong> are the number of individual occurrences of a blocker pattern. Choose the roadblock number to show a list of the source code files where the pattern was detected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Est. Effort</strong> is an estimate of the number of days it will take to remediate the roadblocks in each row.</td>
<td></td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export data to Microsoft Excel.</td>
<td>(Optional) Choose <a href="#">Export to Excel</a> to export the data for further analysis. The application analysis results data can be used to further analyze the cloud readiness of an application and determine what code must to be updated before a migration.</td>
<td>N/A</td>
</tr>
<tr>
<td>View recommendations.</td>
<td>Choose <a href="#">Recommendations</a> next to <a href="#">CloudReady Code Scan</a> to view the <a href="#">Cloud Service Recommendations</a> screen. This identifies AWS services that the application could adopt based on its characteristics. Repeat this step to view recommendations for all the applications that you analyzed.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Related resources

**Campaign management**

- [CAST Highlight – Configure your application portfolio (video)](#)

**Source code analysis**

- [CAST Highlight – Assess your applications (video)](#)

**Other resources**

- [CAST Highlight in AWS Marketplace](#)
- [CAST Highlight – Documentation, product tutorials, and third-party tools](#)
- [CAST Highlight – Cloud readiness demo (video)](#)

### Automatically archive items to Amazon S3 using DynamoDB TTL

*Created by Tabby Ward (AWS), HARI OHM PRASATH RAJAGOPAL (AWS), and Sunny Goel*
Summary

This pattern provides steps to remove older data from an Amazon DynamoDB table and archive it to an Amazon Simple Storage Service (Amazon S3) bucket on Amazon Web Services (AWS) without having to manage a fleet of servers.

This pattern uses Amazon DynamoDB Time to Live (TTL) to automatically delete old items and Amazon DynamoDB Streams to capture the TTL-expired items. It then connects DynamoDB Streams to AWS Lambda, which runs the code without provisioning or managing any servers.

When new items are added to the DynamoDB stream, the Lambda function is initiated and writes the data to an Amazon Kinesis Data Firehose delivery stream. Kinesis Data Firehose provides a simple, fully managed solution to load the data as an archive into Amazon S3.

DynamoDB is often used to store time series data, such as webpage click-stream data or Internet of Things (IoT) data from sensors and connected devices. Rather than deleting less frequently accessed items, many customers want to archive them for auditing purposes. TTL simplifies this archiving by automatically deleting items based on the timestamp attribute.

Items deleted by TTL can be identified in DynamoDB Streams, which captures a time-ordered sequence of item-level modifications and stores the sequence in a log for up to 24 hours. This data can be consumed by a Lambda function and archived in an Amazon S3 bucket to reduce the storage cost. To further reduce the costs, Amazon S3 lifecycle rules can be created to automatically transition the data (as soon as it gets created) to lowest-cost storage classes, such as S3 Glacier Instant Retrieval or S3 Glacier Flexible Retrieval, or Amazon S3 Glacier Deep Archive for long-term storage.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) 1.7 or later, installed and configured on macOS, Linux, or Windows.
- Python 3.7 or later.
- Boto3, installed and configured. If Boto3 is not already installed, run the `python -m pip install boto3` command to install it.

Architecture

Technology stack

- Amazon DynamoDB
- Amazon DynamoDB Streams
AWS Prescriptive Guidance Patterns

Tools

- **Amazon Kinesis Data Firehose**
- **AWS Lambda**
- **Amazon S3**

1. Items are deleted by TTL.
2. The DynamoDB stream trigger invokes the Lambda stream processor function.
3. The Lambda function puts records in the Kinesis Data Firehose delivery stream in batch format.
4. Data records are archived in the S3 bucket.

**Tools**

- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services.
- **Amazon DynamoDB** – Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale.
- **Amazon DynamoDB Time to Live (TTL)** – Amazon DynamoDB TTL helps you define a per-item timestamp to determine when an item is no longer required.
- **Amazon DynamoDB Streams** – Amazon DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table and stores this information in a log for up to 24 hours.
- **Amazon Kinesis Data Firehose** – Amazon Kinesis Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores, and analytics services.
- **AWS Lambda** – AWS Lambda runs code without the need to provision or manage servers. You pay only for the compute time you consume.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.

**Code**

The code for this pattern is available in the GitHub Archive items to S3 using DynamoDB TTL repository.

**Epics**

Set up a DynamoDB table, TTL, and a DynamoDB stream

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a DynamoDB table.</td>
<td>Use the AWS CLI to create a table in DynamoDB called Reservation. Choose random read capacity unit (RCU) and write capacity unit (WCU), and give your table two</td>
<td>Cloud architect, App developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>attributes:</strong> ReservationID and ReservationDate.</td>
<td>aws dynamodb create-table \ --table-name Reservation \ --attribute-definitions AttributeName=ReservationID,AttributeType=S AttributeName=ReservationDate,AttributeType=N \ --key-schema AttributeName=ReservationID,KeyType=HASH AttributeName=ReservationDate,KeyType=RANGE \ --provisioned-throughput ReadCapacityUnits=100,WriteCapacityUnits=100</td>
<td>Cloud architect, App developer</td>
</tr>
<tr>
<td>ReservationDate is an epoch timestamp that will be used to turn on TTL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turn on DynamoDB TTL.</td>
<td>Use the AWS CLI to turn on DynamoDB TTL for the ReservationDate attribute.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws dynamodb update-time-to-live \ --table-name Reservation \ --time-to-live-specification Enabled=true,AttributeName=ReservationDate</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Turn on a DynamoDB stream.</strong></td>
<td>Use the AWS CLI to turn on a DynamoDB stream for the Reservation table by using the NEW_AND_OLD_IMAGES stream type.</td>
<td>Cloud architect, App developer</td>
</tr>
</tbody>
</table>

```bash
aws dynamodb update-table \
  --table-name Reservation \
  --stream-specification
  StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

This stream will contain records for new items, updated items, deleted items, and items that are deleted by TTL. The records for items that are deleted by TTL contain an additional metadata attribute to distinguish them from items that were deleted manually. The `userIdentity` field for TTL deletions indicates that the DynamoDB service performed the delete action.

In this pattern, only the items deleted by TTL are archived, but you could archive only the records where `eventName` is REMOVE and `userIdentity` contains `principalId` equal to `dynamodb.amazonaws.com`.

---

**Create and configure an S3 bucket**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create an S3 bucket.</strong></td>
<td>Use the AWS CLI to create a destination S3 bucket in your AWS Region, replacing <code>us-east-1</code> with your Region.</td>
<td>Cloud architect, App developer</td>
</tr>
</tbody>
</table>

```bash
aws s3api create-bucket \
  --bucket reservationfirehosedestinationbucket \
  --region us-east-1
```

Make sure that the S3 bucket's name is globally unique, because the namespace is shared by all AWS accounts.
### Create a 30-day lifecycle policy for the S3 bucket.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a 30-day lifecycle policy for the S3 bucket. | 1. Sign in to the AWS Management Console and open the Amazon S3 console.  
2. Choose the S3 bucket that contains the data from Kinesis Data Firehose.  
3. In the S3 bucket, choose the Management tab, and choose Add lifecycle rule.  
4. Enter a name for your rule in the Lifecycle rule dialog box, and configure a 30-day lifecycle rule for your bucket. | Cloud architect, App developer |

### Create a Kinesis Data Firehose delivery stream

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create and configure a Kinesis Data Firehose delivery stream. | Download and edit the CreateFireHoseToS3.py code example from the GitHub repository.  
This code is written in Python and shows you how to create a Kinesis Data Firehose delivery stream and an AWS Identity and Access Management (IAM) role. The IAM role will have a policy that can be used by Kinesis Data Firehose to write to the destination S3 bucket.  
To run the script, use the following command and command line arguments.  
Argument 1= <Your_S3_bucket_ARN>, which is the Amazon Resource Name (ARN) for the bucket that you created earlier  
Argument 2= Your Kinesis Data Firehose name (This pilot is using firehose_to_s3_stream.)  
Argument 3= Your IAM role name (This pilot is using firehose_to_s3.) | Cloud architect, App developer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>python CreateFireHoseToS3.py &lt;Your_S3_Bucket_ARN&gt; firehose_to_s3_stream firehose_to_s3</td>
<td>If the specified IAM role does not exist, the script will create an assume role with a trusted relationship policy, as well as a policy that grants sufficient Amazon S3 permission. For examples of these policies, see the Additional information section.</td>
<td>Cloud architect, App developer</td>
</tr>
<tr>
<td>Verify the Kinesis Data Firehose delivery stream.</td>
<td>Describe the Kinesis Data Firehose delivery stream by using the AWS CLI to verify that the delivery stream was successfully created.</td>
<td></td>
</tr>
<tr>
<td>Create a Lambda function to process the Kinesis Data Firehose delivery stream</td>
<td>Create a trust policy for the Lambda function.</td>
<td>Cloud architect, App developer</td>
</tr>
<tr>
<td>Create a trust policy file with the following information.</td>
<td>This gives your function permission to access AWS resources.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Create an execution role for the Lambda function. | To create the execution role, run the following code.  
```bash
aws iam create-role --role-name lambda-ex --assume-role-policy-document file://TrustPolicy.json
``` | Cloud architect, App developer |
| Add permission to the role. | To add permission to the role, use the `attach-policy-to-role` command.  
```bash
aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole  
aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole  
aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/AmazonKinesisFirehoseFullAccess  
aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/IAMFullAccess
``` | Cloud architect, App developer |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Lambda function</td>
<td>Compress the LambdaStreamProcessor.py file from the code repository by running the following command.</td>
<td>Cloud architect, App developer</td>
</tr>
<tr>
<td></td>
<td><code>zip function.zip</code>&lt;br&gt;<code>LambdaStreamProcessor.py</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When you create the Lambda function, you will need the Lambda execution role ARN. To get the ARN, run the following code.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>aws iam get-role --role-name lambda-ex</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To create the Lambda function, run the following code.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>aws lambda create-function --function-name LambdaStreamProcessor \</code>&lt;br&gt;<code>--zip-file fileb://function.zip</code>&lt;br&gt;<code>--handler LambdaStreamProcessor.handler</code>&lt;br&gt;<code>--runtime python3.8</code>&lt;br&gt;<code>--role {Your Lambda Execution Role ARN}</code>&lt;br&gt;<code>--environment Variables=&quot;{firehose_name=firehose_to_s3_stream,bucket_arn = arn:aws:s3:::reservationfirehosedestinationbucket,iam_role_name = firehose_to_s3, batch_size=400}&quot;</code></td>
<td></td>
</tr>
<tr>
<td>Configure the Lambda function trigger.</td>
<td>Use the AWS CLI to configure the trigger (DynamoDB Streams), which invokes the Lambda function. The batch size of 400 is to avoid running into Lambda concurrency issues.</td>
<td>Cloud architect, App developer</td>
</tr>
<tr>
<td></td>
<td><code>aws lambda create-event-source-mapping --function-name LambdaStreamProcessor \</code>&lt;br&gt;<code>--batch-size 400 --starting-position LATEST</code>&lt;br&gt;<code>--event-source-arn &lt;Your Latest Stream ARN From DynamoDB Console&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
Test the functionality

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add items with expired timestamps to the Reservation table.</td>
<td>To test the functionality, add items with expired epoch timestamps to the Reservation table. TTL will automatically delete items based on the timestamp. The Lambda function is initiated upon DynamoDB Stream activities, and it filters the event to identify REMOVE activity or deleted items. It then puts records in the Kinesis Data Firehose delivery stream in batch format. The Kinesis Data Firehose delivery stream transfers items to a destination S3 bucket with the <code>firehosetos3example/year=current year/month=current month/day=current day/hour=current hour/prefix</code>. <strong>Important:</strong> To optimize data retrieval, configure Amazon S3 with the Prefix and ErrorOutputPrefix that are detailed in the Additional information section.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Clean up the resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete all resources.</td>
<td>Delete all the resources to ensure that you aren’t charged for any services that you aren’t using.</td>
<td>Cloud architect, App developer</td>
</tr>
</tbody>
</table>

Related resources

- Managing your storage lifecycle
- Amazon S3 Storage Classes
- AWS SDK for Python (Boto3) documentation
Create and configure a Kinesis Data Firehose delivery stream – Policy examples

**Kinesis Data Firehose trusted relationship policy example document**

```json
firehose_assume_role = {
    'Version': '2012-10-17',
    'Statement': [
        {
            'Sid': '',
            'Effect': 'Allow',
            'Principal': {
                'Service': 'firehose.amazonaws.com'
            },
            'Action': 'sts:AssumeRole'
        }
    ]
}
```

**S3 permissions policy example**

```json
s3_access = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "{your s3_bucket ARN}/**",
                "{Your s3 bucket ARN}"
            ]
        }
    ]
}
```

**Test the functionality – Amazon S3 configuration**

The Amazon S3 configuration with the following Prefix and ErrorOutputPrefix is chosen to optimize data retrieval.

**Prefix**

```bash
firehosetos3example/year=! {timestamp: yyyy}/month=! {timestamp:MM}/day=! {timestamp:dd}/hour=!{timestamp:HH}/
```

Kinesis Data Firehose first creates a base folder called `firehosetos3example` directly under the S3 bucket. It then evaluates the expressions `{timestamp:yyyy}`, `{timestamp:MM}`, `{timestamp:dd}`, and `{timestamp:HH}` to year, month, day, and hour using the Java `DateTimeFormatter` format.

For example, an approximate arrival timestamp of 1604683577 in Unix epoch time evaluates to year=2020, month=11, day=06, and hour=05. Therefore, the location in Amazon S3, where data
records are delivered, evaluates to firehosetos3example/year=2020/month=11/day=06/hour=05/.

ErrorOutputPrefix

```
firehosetos3erroroutputbase/{firehose:random-string}/{firehose:error-output-type}/{timestamp:yyyy/MM/dd}/
```

The ErrorOutputPrefix results in a base folder called firehosetos3erroroutputbase directly under the S3 bucket. The expression `{firehose:random-string}` evaluates to an 11-character random string such as ztWxkdg3Thg. The location for an Amazon S3 object where failed records are delivered could evaluate to firehosetos3erroroutputbase/ztWxkdg3Thg/processing-failed/2020/11/06/.

### Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager

*Created by Kevin Yung (AWS), Peter Woods (Micro Focus), Abraham Rondon (Micro Focus), and Krithika Palani Selvam (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Modernization; Cloud-native; DevOps; Infrastructure</th>
</tr>
</thead>
</table>

**Summary**

This pattern introduces a scalable architecture for mainframe applications using Micro Focus Enterprise Server in Scale-Out Performance and Availability Cluster (PAC) and an Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling group on Amazon Web Services (AWS). The solution is fully automated with AWS Systems Manager and Amazon EC2 Auto Scaling lifecycle hooks. By using this pattern, you can set up your mainframe online and batch applications to achieve high resiliency by automatically scaling in and out based on your capacity demands.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account.
- Micro Focus Enterprise Server software and license. For details, contact Micro Focus sales.
- An understanding of the concept of rebuilding and delivering a mainframe application to run in Micro Focus Enterprise Server. For a high-level overview, see Micro Focus Enterprise Server Data Sheet.
- An understanding of the concepts in Micro Focus Enterprise Server scale-out Performance and Availability Cluster. For more information, see the Micro Focus Enterprise Server documentation.
- An understanding of the overall concept of mainframe application DevOps with continuous integration (CI). For an AWS Prescriptive Guidance pattern that was developed by AWS and Micro Focus, see Mainframe modernization: DevOps on AWS with Micro Focus.

**Limitations**

- For a list of platforms that are supported by Micro Focus Enterprise Server, see the Micro Focus Enterprise Server Data Sheet.
• The scripts and tests used in this pattern are based on Amazon EC2 Windows Server 2019; other Windows Server versions and operating systems were not tested for this pattern.
• The pattern is based on Micro Focus Enterprise Server 6.0 for Windows; earlier or later releases were not tested in the development of this pattern.

Product versions

• Micro Focus Enterprise Server 6.0
• Windows Server 2019

Architecture

In the conventional mainframe environment, you must provision hardware to host your applications and corporate data. To cater for and meet the spikes in seasonal, monthly, quarterly, or even unprecedented or unexpected demands, mainframe users must scale out by purchasing additional storage and compute capacity. Increasing the number of storage and compute capacity resources improves overall performance, but the scaling is not linear.

This is not the case when you start adopting an on-demand consumption model on AWS by using Amazon EC2 Auto Scaling and Micro Focus Enterprise Servers. The following sections explain detail how to build a fully automated, scalable mainframe application architecture using Micro Focus Enterprise Server Scale-Out Performance and Availability Cluster (PAC) with an Amazon EC2 Auto Scaling group.

Micro Focus Enterprise Server automatic scaling architecture

First, it is important to understand the basic concepts of Micro Focus Enterprise Server. This environment provides a mainframe-compatible, x86 deployment environment for applications that have traditionally run on the IBM mainframe. It delivers both online and batch runs and a transaction environment that supports the following:

• IBM COBOL
• IBM PL/I
• IBM JCL batch jobs
• IBM CICS and IMS TM transactions
• Web services
• Common batch utilities, including SORT

Micro Focus Enterprise Server enables mainframe applications to run with minimal changes. Existing mainframe workloads can be moved to x86 platforms and modernized to take advantage of AWS Cloud native extensions for rapid expansion to new markets or geographies.

The AWS Prescriptive Guidance pattern Mainframe modernization: DevOps on AWS with Micro Focus introduced the architecture to accelerate the development and testing of mainframe applications on AWS using Micro Focus Enterprise Developer and Enterprise Test Server with AWS CodePipeline and AWS CodeBuild. This pattern focuses on the deployment of mainframe applications to the AWS production environment to achieve high availability and resiliency.

In a mainframe production environment, you might have set up IBM Parallel Sysplex in the mainframe to achieve high performance and high availability. To create a scale-out architecture similar to Sysplex, Micro Focus introduced the Performance and Availability Cluster (PAC) to Enterprise Server. PACs support mainframe application deployment onto multiple Enterprise Server regions managed as a single image and scaled out in Amazon EC2 instances. PACs also support predictable application performance and system throughput on demand.
In a PAC, multiple Enterprise Server instances work together as a single logical entity. Failure of one Enterprise Server instance, therefore, will not interrupt business continuity because capacity is shared with other regions while new instances are automatically started using industry standard functionality such as an Amazon EC2 Auto Scaling group. This removes single points of failure, improving resilience to hardware, network, and application issues. Scaled-out Enterprise Server instances can be operated and managed by using the Enterprise Server Common Web Administration (ESCWA) APIs, simplifying the operational maintenance and serviceability of Enterprise Servers.

**Note:** Micro Focus recommends that the Performance and Availability Cluster (PAC) should consist of at least three Enterprise Server regions so that availability is not compromised in the event an Enterprise Server region fails or requires maintenance.

PAC configuration requires a supported relational database management service (RDBMS) to manage the region database, a cross-region database, and optional data store databases. A data store database should be used to managed Virtual Storage Access Method (VSAM) files using the Micro Focus Database File Handler support to improve availability and scalability. Supported RDBMSs include the following:

- Microsoft SQL Server 2009 R2 and later
- PostgreSQL 10.x, including Amazon Aurora PostgreSQL-Compatible Edition
- DB2 10.4 and later

For details of supported RDBMS and PAC requirements, see Micro Focus Enterprise Server - Prerequisites and Micro Focus Enterprise Server - Recommended PAC Configuration.

The following diagram shows a typical AWS architecture setup for a Micro Focus PAC.
1. **Enterprise Server instances automatic scaling group**
   - **Component**: Amazon EC2
   - **Description**: Set up an automatic scaling group deployed with Enterprise Server instances in a PAC. The number of instances can be scaled out or in initiated by Amazon CloudWatch alarms using CloudWatch metrics.

2. **Enterprise Server ESCWA instances automatic scaling group**
   - **Component**: Amazon EC2
   - **Description**: Set up an automatic scaling group deployed with Enterprise Server Common Web Administration (ESCWA). ESCWA provides cluster management APIs. The ESCWA servers act as a control plane to add or remove Enterprise Servers and start or stop Enterprise Server regions.
### Component

<table>
<thead>
<tr>
<th>Number</th>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Amazon Aurora instance in a Multi-AZ setup</td>
<td>Set up a relational database management system (RDBMS) to host both user and system data files to be shared across the Enterprise Server instances.</td>
</tr>
<tr>
<td>4</td>
<td>Amazon ElastiCache for Redis instance and replica</td>
<td>Set up an ElastiCache Redis primary instance and at least one replica to host user data and act as a scale-out repository (SOR) for the Enterprise Server instances. You can configure one or more scale-out repository to store specific types of user data. Enterprise Server uses a Redis NoSQL database as an SOR, a requirement to maintain PAC integrity.</td>
</tr>
<tr>
<td>5</td>
<td>Network Load Balancer</td>
<td>Set up a load balancer, providing a hostname for applications to connect to the services provided by Enterprise Server instances (for example, accessing the application through a 3270 emulator).</td>
</tr>
</tbody>
</table>

These components form the minimum requirement for a Micro Focus Enterprise Server PAC cluster. The next section covers cluster management automation.

**Using AWS Systems Manager Automation for scaling**

After the PAC cluster is deployed on AWS, the PAC is managed through the Enterprise Server Common Web Administration (ESCWA) APIs.

To automate the cluster management tasks during automatic scaling events, you can use Systems Manager Automation runbooks and Amazon EC2 Auto Scaling with Amazon EventBridge. The architecture of these automations is shown in the following diagram.
### Component | Description
--- | ---
1 | Automatic scaling lifecycle hook
Set up automatic scaling lifecycle hooks and send notifications to Amazon EventBridge when new instances are launched and existing instances are terminated in the automatic scaling group.
2 | Amazon EventBridge
Set up an Amazon EventBridge rule to route automatic scaling events to Systems Manager Automation runbook targets.
3 | Automation runbooks
Set up Systems Manager Automation runbooks to run Windows PowerShell scripts and invoke ESCWA APIs to manage the PAC. For examples, see the Additional information section.
4 | Enterprise Server ESCWA instance in an automatic scaling group
Set up an Enterprise Server ESCWA instance in an automatic scaling group. The ESCWA instance provides APIs to manage the PAC.
Tools

- **Micro Focus Enterprise Server** – Micro Focus Enterprise Server provides the run environment for applications created with any integrated development environment (IDE) variant of Enterprise Developer.

- **Amazon EC2 Auto Scaling** – Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups, and specify minimum and maximum numbers of instances.

- **Amazon ElastiCache for Redis** – Amazon ElastiCache is a web service for setting up, managing, and scaling a distributed in-memory data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution.

- **Amazon RDS** – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for a relational database and manages common database administration tasks.

- **AWS Systems Manager** – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

Epics

Create an Amazon Aurora instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS CloudFormation template for an Amazon Aurora instance.</td>
<td>Use the <a href="https://aws.amazon.com">AWS example code snippet</a> to make a CloudFormation template that will create an Amazon Aurora PostgreSQL-Compatible Edition instance.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy a CloudFormation stack to create the Amazon Aurora instance.</td>
<td>Use the CloudFormation template to create an Aurora PostgreSQL-Compatible instance that has Multi-AZ replication enabled for production workloads.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Configure database connection settings for Enterprise Server.</td>
<td>Follow the instructions in the <a href="https://microfocus.com">Micro Focus documentation</a> to prepare the connection strings and database configuration for Micro Focus Enterprise Server.</td>
<td>Data engineer, DevOps engineer</td>
</tr>
</tbody>
</table>
### Create an Amazon ElastiCache cluster for the Redis instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudFormation template for the Amazon ElastiCache cluster</td>
<td>Use the AWS example code snippet to make a CloudFormation template that will create an Amazon ElastiCache cluster for the Redis instance.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>for the Redis instance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy the CloudFormation stack to create an Amazon ElastiCache</td>
<td>Create the Amazon ElastiCache cluster for the Redis instance that has Multi-AZ replication enabled for production workloads.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>stack to create an Amazon ElastiCache cluster for the Redis instance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure Enterprise Server PSOR connection settings.</td>
<td>Follow the instructions in the Micro Focus documentation to prepare the PAC Scale-Out Repository (PSOR) connection configuration for Micro Focus Enterprise Server PAC.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

### Create a Micro Focus Enterprise Server ESCWA automatic scaling group

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Micro Focus Enterprise Server AMI.</td>
<td>Create an Amazon EC2 Windows Server instance and install the Micro Focus Enterprise Server binary in the EC2 instance. Create an Amazon Machine Image (AMI) of the EC2 instance. For more information, see the Enterprise Server installation documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Create a CloudFormation template for Enterprise Server ESCWA.</td>
<td>Use the AWS example code snippet to make a template for creating a custom stack of Enterprise Server ESCWA in an automatic scaling group.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation stack to create an Amazon EC2 scaling group</td>
<td>Use the CloudFormation template to deploy the automatic scaling group with the Micro Focus Enterprise Server ESCWA AMI created in the previous story.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>for Enterprise Server ESCWA.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Create an AWS Systems Manager Automation runbook

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudFormation template for a Systems Manager Automation runbook.</td>
<td>Use the example code snippets in the Additional information section to make a CloudFormation template that will create a Systems Manager Automation runbook for automating PAC creation, Enterprise Server scale in, and Enterprise Server scale out.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation stack that contains the Systems Manager Automation runbook.</td>
<td>Use the CloudFormation template to deploy a stack that contains the Automation runbook for PAC creation, Enterprise Server scale in, and Enterprise Server scale out.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Create an automatic scaling group for Micro Focus Enterprise Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudFormation template for setting up an automatic scaling group for Micro Focus Enterprise Server.</td>
<td>Use the AWS example code snippet to make a CloudFormation template that will create an automatic scaling group. This template will reuse the same AMI that was created for the Micro Focus Enterprise Server ESCWA instance. Then use an AWS example code snippet to create the automatic scaling lifecycle event and set up Amazon EventBridge to filter for scale-out and scale-in events in the same CloudFormation template.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation stack for the automatic scaling group for Micro Focus Enterprise Servers.</td>
<td>Deploy the CloudFormation stack that contains the automatic scaling group for Micro Focus Enterprise Servers.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Micro Focus Enterprise Server Performance and Availability Cluster (PAC)
- Amazon EC2 Auto Scaling lifecycle hooks
- Running automations with triggers using EventBridge
Additional information

The following scenarios must be automated for scaling in or scaling out the PAC clusters.

Automation for starting or recreating a PAC

At the start of a PAC cluster, Enterprise Server requires ESCWA to invoke APIs to create a PAC configuration. This starts and adds Enterprise Server regions into the PAC. To create or recreate a PAC, use the following steps:

1. Configure a PAC Scale-Out Repository (PSOR) in ESCWA with a given name.

   ```
   POST /server/v1/config/groups/sors
   ```

2. Create a PAC with a given name and attach the PSOR to it.

   ```
   POST /server/v1/config/groups/pacs
   ```

3. Configure the region database and cross-region database if this is the first time you are setting up a PAC. **Note:** This step uses SQL queries and the Micro Focus Enterprise Suite command line `dbhfplan` tool to create the database and import initial data.

4. Install the PAC definition into the Enterprise Server regions.

   ```
   POST /server/v1/config/mfds
   POST /native/v1/config/groups/pacs/${pac_uid}/install
   ```

5. Start Enterprise Server regions in the PAC.

   ```
   POST /native/v1/regions/${host_ip}/${port}/${region_name}/start
   ```

The previous steps can be implemented by using a Windows PowerShell script.

The following steps explain how to build an automation for creating a PAC by reusing the Windows PowerShell script.

1. Create an Amazon EC2 launch template that downloads or creates the Windows PowerShell script as part of the bootstrap process. For example, you can use EC2 user data to download the script from an Amazon Simple Storage Service (Amazon S3) bucket.

2. Create an AWS Systems Manager Automation runbook to invoke the Windows PowerShell script.

3. Associate the runbook to the ESCWA instance by using the instance tag.

4. Create an ESCWA automatic scaling group by using the launch template.

You can use the following example AWS CloudFormation snippet to create the Automation runbook.

*Example CloudFormation snippet for a Systems Manager Automation runbook used for PAC creation*

```
PACInitDocument:
  Type: AWS::SSM::Document
  Properties:
    DocumentType: Command
    Content:
      schemaVersion: '2.2'
      description: Operation Runbook to create Enterprise Server PAC
      mainSteps:
```
For more information, see Micro Focus Enterprise Server - Configuring a PAC.

Automation for scaling out with a new Enterprise Server instance

When an Enterprise Server instance is scaled out, its Enterprise Server region must be added to the PAC. The following steps explain how to invoke ESCWA APIs and add the Enterprise Server region into the PAC.

1. Install the PAC definition into the Enterprise Server regions.

   ```
   POST '/server/v1/config/mfds'
   POST /native/v1/config/groups/pacs/{pac_uid}/install
   ```

2. Warm Start the region in the PAC.

   ```
   POST /native/v1/regions/{host_ip}/{port}/{region_name}/start
   ```

3. Add the Enterprise Server instance to the load balancer by associating the automatic scaling group to the load balancer.

   The previous steps can be implemented by using a Windows PowerShell script. For more information, see Micro Focus Enterprise Server - Configuring a PAC.

   The following steps can be used to build an event driven automation to add a newly launched Enterprise Server instance into a PAC by reusing the Windows PowerShell script.
1. Create an Amazon EC2 launch template for Enterprise Server instance that provisions an Enterprise Server Region during its bootstrap. For example, you can use the Micro Focus Enterprise Server command mfds to import a region configuration. For further details and options available for this command, see the Enterprise Server Reference.

2. Create an Enterprise Server automatic scaling group that uses the launch template created in the previous step.

3. Create a Systems Manager Automation runbook to invoke the Windows PowerShell script.

4. Associate the runbook to the ESCWA instance by using the instance tag.

5. Create an Amazon EventBridge rule to filter for the EC2 Instance Launch Successful event for the Enterprise Server automatic scaling group, and create the target to use the Automation runbook.

You can use the following example CloudFormation snippet to create the Automation runbook and the EventBridge rule.

Example CloudFormation snippet for Systems Manager used for scaling out Enterprise Server instances

```
ScaleOutDocument:
  Type: AWS::SSM::Document
  Properties:
    DocumentType: Command
    Content:
      schemaVersion: '2.2'
      description: Operation Runbook to Adding MFDS Server into an existing PAC
      parameters:
        MfdsPort:
          type: String
        InstanceIpAddress:
          type: String
          default: "Not-Available"
        InstanceId:
          type: String
          default: "Not-Available"
      mainSteps:
        - action: aws:runPowerShellScript
          name: Add_MFDS
          inputs:
            onFailure: Abort
            timeoutSeconds: "300"
            runCommand:
              - $ip = "{{InstanceIpAddress}}"
              if ( $ip -eq "Not-Available" ) {
                $ip = aws ec2 describe-instances --instance-id {{InstanceId}} --output text --query "Reservations[0].Instances[0].PrivateIpAddress"
              }
              C:\Scripts\Scale-Out.ps1 -host_ip $ip -port {{MfdsPort}}

PacScaleOutAutomation:
  Type: AWS::SSM::Document
  Properties:
    DocumentType: Automation
    Content:
      parameters:
        MfdsPort:
          type: String
        InstanceIpAddress:
          type: String
          default: "Not-Available"
        InstanceId:
          type: String
          default: "Not-Available"
```
AWS Prescriptive Guidance Patterns
Additional information

```
description: Scale Out 1 New Server in Micro Focus PAC Cluster via ESCWA Server
schemaVersion: '0.3'
assumeRole: !GetAtt SsmAssumeRole.Arn
mainSteps:
  - name: RunScaleOutCommand
    action: aws:runCommand
    timeoutSeconds: 300
    onFailure: Abort
    inputs:
      DocumentName: !Ref ScaleOutDocument
      Parameters:
        InstanceIpAddress: "{InstanceId}"
        InstanceId: "{InstanceId}"
        MfdsPort: "{MfdsPort}"
      Targets:
        - Key: tag:Enterprise Server - ESCWA
          Values:
            - "true"
```

**Automation for scaling in an Enterprise Server instance**

Similar to scaling out, when an Enterprise Server instance is scaled in, the event EC2 Instance-terminate Lifecycle Action is initiated, and the following process and API calls are needed to remove a Micro Focus Enterprise Server instance from the PAC.

1. Stop the region in the terminating Enterprise Server instance.

   ```
   POST "/native/v1/regions/${host_ip}/${port}/${region_name}/stop"
   ```

2. Remove the Enterprise Server Instance from the PAC.

   ```
   DELETE "/server/v1/config/mfds/${uid}"
   ```

3. Send signal to continue terminating the Enterprise Server instance.

   The previous steps can be implemented in a Windows PowerShell script. For additional details of this process, see Micro Focus Enterprise Server document - Administering a PAC.

   The following steps explain how to build an event-driven automation to terminate an Enterprise Server instance from a PAC by reusing the Windows PowerShell script.

   1. Create a Systems Manager Automation runbook to invoke the Windows PowerShell script.
   2. Associate the runbook to the ESCWA instance by using the instance tag.
   3. Create an automatic scaling group lifecycle hook for EC2 instance termination.
   4. Create an Amazon EventBridge rule to filter EC2 Instance-terminate Lifecycle Action event for the Enterprise Server automatic scaling group, and create the target to use the Automation runbook.

   You can use the following example CloudFormation template for creating a Systems Manager Automation runbook, lifecycle hook, and EventBridge rule.

   **Example CloudFormation snippet for a Systems Manager Automation runbook used for scaling in an Enterprise Server instance**

   ```
   ScaleInDocument:
     Type: AWS::SSM::Document
     Properties:
       DocumentType: Command
   ```
AWS Prescriptive Guidance Patterns
Additional information

Content:

```
schemaVersion: '2.2'
description: Operation Runbook to Remove MFDS Server from PAC
parameters:
  MfdsPort:
    type: String
  InstanceIpAddress:
    type: String
default: "Not-Available"
  InstanceId:
    type: String
default: "Not-Available"
mainSteps:
- action: aws:runPowerShellScript
  name: Remove_MFDS
  inputs:
    onFailure: Abort
    runCommand:
    - $ip = "{{InstanceIpAddress}}"
    if ( $ip -eq "Not-Available" ) {
      $ip = aws ec2 describe-instances --instance-id {{InstanceId}} --output text --query "Reservations[0].Instances[0].PrivateIpAddress"
    }
    C:\Scripts\Scale-In.ps1 -host_ip $ip -port {{MfdsPort}}

PacScaleInAutomation:
Type: AWS::SSM::Document
Properties:
  DocumentType: Automation
  Content:
    parameters:
      MfdsPort:
        type: String
      InstanceIpAddress:
        type: String
default: "Not-Available"
      InstanceId:
        type: String
default: "Not-Available"
    description: Scale In 1 New Server in Micro Focus PAC Cluster via ESCWA Server
    schemaVersion: '0.3'
    assumeRole: !GetAtt SsmAssumeRole.Arn
    mainSteps:
    - name: RunScaleInCommand
      action: aws:runCommand
      timeoutSeconds: "600"
      onFailure: Abort
      inputs:
        DocumentName: !Ref ScaleInDocument
        Parameters:
          InstanceIpAddress: "{{InstanceIpAddress}}"
          MfdsPort: "{{MfdsPort}}"
          InstanceId: "{{InstanceId}}"
        Targets:
          - Key: tag:Enterprise Server - ESCWA
            Values:
              - "true"
        - name: TerminateTheInstance
          action: aws:executeAwsApi
          inputs:
            Service: autoscaling
            Api: CompleteLifecycleAction
            AutoScalingGroupName: !Ref AutoScalingGroup
            InstanceId: "{{ InstanceId }}"
            LifecycleActionResult: CONTINUE
```

1358
Build a multi-tenant serverless architecture in Amazon OpenSearch Service

Created by Tabby Ward (AWS) and Nisha Gambhir (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Modernization; SaaS; Serverless</th>
<th>Workload:</th>
<th>Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services:</strong></td>
<td>Amazon OpenSearch Service; AWS Lambda; Amazon S3; Amazon API Gateway</td>
<td><strong>AWS services:</strong> Amazon OpenSearch Service; AWS Lambda; Amazon S3; Amazon API Gateway</td>
<td><strong>Workload:</strong> Open-source</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

Amazon OpenSearch Service (successor to Amazon Elasticsearch Service) is a managed service that makes it easy to deploy, operate, and scale Elasticsearch, which is a popular open-source search and analytics engine. Amazon OpenSearch Service provides free-text search as well as near real-time ingestion and dashboarding for streaming data such as logs and metrics.

Software as a service (SaaS) providers frequently use Amazon OpenSearch Service to address a broad range of use cases, such as gaining customer insights in a scalable and secure way while reducing complexity and downtime.

Using Amazon OpenSearch Service in a multi-tenant environment introduces a series of considerations that affect partitioning, isolation, deployment, and management of your SaaS solution. SaaS providers have to consider how to effectively scale their Elasticsearch clusters with continually shifting workloads. They also need to consider how tiering and noisy neighbor conditions could impact their partitioning model.

This pattern reviews the models that are used to represent and isolate tenant data with Elasticsearch constructs. In addition, the pattern focuses on a simple serverless reference architecture as an example to demonstrate indexing and searching using Amazon OpenSearch Service in a multi-tenant environment. It implements the pool data partitioning model, which shares the same index among all tenants while maintaining a tenant's data isolation. This pattern uses the following Amazon Web Services (AWS) services: Amazon API Gateway, AWS Lambda, Amazon Simple Storage Service (Amazon S3), and Amazon OpenSearch Service.

For more information about the pool model and other data partitioning models, see the Additional information (p. 1385) section.
Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2.x, installed and configured on macOS, Linux, or Windows
- Python version 3.7
- pip3 – The Python source code is provided as a .zip file to be deployed in a Lambda function. If you want to use the code locally or customize it, follow these steps to develop and recompile the source code:
  1. Generate the requirements.txt file by running the following command in the same directory as the Python scripts: `pip3 freeze > requirements.txt`
  2. Install the dependencies: `pip3 install -r requirements.txt`

Limitations

- This code runs in Python, and doesn’t currently support other programming languages.
- The sample application doesn’t include AWS cross-Region or disaster recovery (DR) support.
- This pattern is intended for demonstration purposes only. It is not intended to be used in a production environment.

Architecture

The following diagram illustrates the high-level architecture of this pattern. The architecture includes the following:

- AWS Lambda to index and query the content
- Amazon OpenSearch Service to perform search
- Amazon API Gateway to provide an API interaction with the user
- Amazon S3 to store raw (non-indexed) data
- Amazon CloudWatch to monitor logs
- AWS Identity and Access Management (IAM) to create tenant roles and policies
Automation and scale

For simplicity, the pattern uses AWS CLI to provision the infrastructure and to deploy the sample code. You can create an AWS CloudFormation template or AWS Cloud Development Kit (AWS CDK) scripts to automate the pattern.

Tools

AWS services

- **AWS CLI** - AWS Command Line Interface (AWS CLI) is a unified tool for managing AWS services and resources by using commands in your command-line shell.
- **AWS Lambda** – AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon API Gateway** – Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- **Amazon S3** - Amazon Simple Storage Service (Amazon S3) is an object storage service that lets you store and retrieve any amount of information at any time, from anywhere on the web.
- **Amazon OpenSearch Service** – Amazon OpenSearch Service (successor to Amazon Elasticsearch) is a fully managed service that makes it easy for you to deploy, secure, and run Elasticsearch cost-effectively at scale.

Code

The attachment provides sample files for this pattern. These include:

- **index_lambda_package.zip** – The Lambda function for indexing data in Amazon OpenSearch Service by using the pool model.
- **search_lambda_package.zip** – The Lambda function for searching for data in Amazon OpenSearch Service.
• Tenant-1-data – Sample raw (non-indexed) data for Tenant-1.
• Tenant-2-data – Sample raw (non-indexed) data for Tenant-2.

Important: The stories in this pattern include CLI command examples that are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

Epics

Create and configure an S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket.</td>
<td>Create an S3 bucket in your AWS Region. This bucket will hold the non-indexed tenant data for the sample application. Make sure that the S3 bucket’s name is globally unique, because the namespace is shared by all AWS accounts.</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>To create an S3 bucket, you can use the AWS CLI <code>create-bucket</code> command as follows:</td>
<td></td>
</tr>
</tbody>
</table>
|                           | ```aws s3api create-bucket \  
|                           |   --bucket tenantrawdata \  
|                           |   --region <your-AWS-Region>```                                                                                                                                         |                                  |
|                           | where `tenantrawdata` is the S3 bucket name. (You can use any unique name that follows the bucket naming guidelines.)                                                                                     |                                  |

Create and configure an Elasticsearch cluster

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon OpenSearch Service domain.</td>
<td>Run the AWS CLI <code>create-elasticsearch-domain</code> command to create an Amazon OpenSearch Service domain:</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>
|                                 | ```aws es create-elasticsearch-domain \  
|                                 |   --domain-name vpc-cli-example \  
|                                 |   --elasticsearch-version 7.10 \  
<p>|                                 |   --elasticsearch-cluster-config InstanceType=t3.medium.elasticsearch,InstanceCount=1```                                                                                                               |                                  |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--ebs-options</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBSEnabled=true,VolumeType(gp2),VolumeSize=10</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\--domain-endpoint-options</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>{&quot;EnforceHTTPS&quot;: true}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\--encryption-at-rest-options</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>{&quot;Enabled&quot;: true}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\--node-to-node-encryption-options</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>{&quot;Enabled&quot;: true}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\--advanced-security-options</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>{&quot;Enabled&quot;: true,&quot;InternalUserDatabaseEnabled&quot;: true,&quot;MasterUserOptions&quot;: {&quot;MasterUserName&quot;: &quot;KibanaUser&quot;,&quot;MasterPassword&quot;: &quot;NewKibanaPassword@123&quot;}}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\--vpc-options</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>{&quot;SubnetIds&quot;: [&quot;&lt;subnet-id&gt;&quot;], &quot;SecurityGroupIds&quot;: [&quot;&lt;sg-id&gt;&quot;]}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>\--access-policies</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The instance count is set to 1 because the domain is for testing purposes. You need to enable fine-grained access control by using the advanced-security-options parameter, because the details cannot be changed after the domain has been created.

This command creates a master user name (KibanaUser) and a password that you can use to log in to the Kibana console.

Because the domain is part of a virtual private cloud (VPC), you have to make sure that you can reach the Elasticsearch instance.
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>by specifying the access policy to use.</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>For more information, see <a href="#">Launching your Amazon OpenSearch Service domains using a VPC in the Amazon ES documentation</a>.</td>
<td></td>
</tr>
</tbody>
</table>

Set up a bastion host.

Set up a Amazon Elastic Compute Cloud (Amazon EC2) Windows instance as a bastion host to access the Kibana console. The Elasticsearch security group must allow traffic from the Amazon EC2 security group. For instructions, see the blog post [Controlling Network Access to EC2 Instances Using a Bastion Server](#).

When the bastion host has been set up, and you have the security group that is associated with the instance available, use the AWS CLI `authorize-security-group-ingress` command to add permission to the Elasticsearch security group to allow port 443 from the Amazon EC2 (bastion host) security group.

```bash
aws ec2 authorize-security-group-ingress \
  --group-id <SecurityGroupIdElasticSearch> \
  --protocol tcp \
  --port 443 \
  --source-group <SecurityGroupIdBastionHostEC2>
```

### Create and configure the Lambda index function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run the AWS CLI <code>create-role</code> command to grant the Lambda index function access to AWS services and resources:</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>
|      | `aws iam create-role \
  --role-name index-lambda-role` | |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>--assume-role-</td>
<td>policy-document file://lambda_assume_role.json</td>
<td>Cloud architect, Cloud</td>
</tr>
<tr>
<td></td>
<td>where</td>
<td>administrator</td>
</tr>
<tr>
<td></td>
<td>lambda_assume_role.json is a JSON document in the current folder that grants</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AssumeRole permissions to the Lambda function, as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Version&quot;: &quot;2012-10-17&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Statement&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Effect&quot;: &quot;Allow&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Principal&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Service&quot;: &quot;lambda.amazonaws.com&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;: &quot;sts:AssumeRole&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Attach managed policies to the</td>
<td>Run the AWS CLI attach-role-policy command to attach managed policies to the</td>
<td>Cloud architect, Cloud</td>
</tr>
<tr>
<td>Lambda role.</td>
<td>role created in the previous step. These two policies give the role</td>
<td>administrator</td>
</tr>
<tr>
<td></td>
<td>permissions to create an elastic network interface and to write logs to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CloudWatch Logs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws iam attach-role-policy \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--role-name index-lambda-role \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aws iam attach-role-policy \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--role-name index-lambda-role \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Create a policy to give the Lambda index function permission to read the S3 objects.</td>
<td>Run the AWS CLI <code>create-policy</code> command to give the Lambda index function <code>s3:GetObject</code> permission to read the objects in the S3 bucket:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>aws iam create-policy --policy-name s3-permission-policy --policy-document file://s3-policy.json</code></td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>The file <code>s3-policy.json</code> is a JSON document in the current folder that grants <code>s3:GetObject</code> permissions to allow read access to S3 objects. If you used a different name when you created the S3 bucket, provide the correct bucket name in the <code>Resource</code> section in the following:</td>
<td></td>
</tr>
<tr>
<td>Attach the Amazon S3 permission policy to the Lambda execution role.</td>
<td>Run the AWS CLI <code>attach-role-policy</code> command to attach the Amazon S3 permission policy you created in the previous step to the Lambda execution role:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>aws iam attach-role-policy --role-name index-lambda-role --policy-arn &lt;PolicyARN&gt;</code></td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>where <code>PolicyARN</code> is the Amazon ResourceARN (ARN) of the Amazon S3 permission policy. You can get this value from the output of the previous command.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Create the Lambda index function.</td>
<td>Run the AWS CLI <code>create-function</code> command to create the Lambda index function, which will access Amazon OpenSearch Service:</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>
| | ```bash
aws lambda create-function
   --function-name index-lambda-function
   --zip-file fileb://index_lambda_package.zip
   --handler lambda_index.lambda_handler
   --runtime python3.7
   --role "arn:aws:iam::account-id:role/index-lambda-role"
   --timeout 30
   --vpc-config "{"SubnetIds": ["<subnet-id1>", "<subnet-id2>"], "SecurityGroupIds": ["<sg-1>"]]"
``` | |
| Allow Amazon S3 to call the Lambda index function. | Run the AWS CLI `add-permission` command to give Amazon S3 the permission to call the Lambda index function: | Cloud architect, Cloud administrator |
| | ```bash
aws lambda add-permission
   --function-name index-lambda-function
   --statement-id s3-permissions
   --action lambda:InvokeFunction
   --principal s3.amazonaws.com
   --source-arn "arn:aws:s3:::tenantrawdata"
   --source-account "<account-id>"
``` | |
### Add a Lambda trigger for the Amazon S3 event.

**Task**
Add a Lambda trigger for the Amazon S3 event.

**Description**
Run the AWS CLI `put-bucket-notification-configuration` command to send notifications to the Lambda index function when the Amazon S3 `ObjectCreated` event is detected. The index function runs whenever an object is uploaded to the S3 bucket.

```bash
aws s3api put-bucket-notification-configuration
   --bucket tenantrawdata
   --notification-configuration file://s3-trigger.json
```

The file `s3-trigger.json` is a JSON document in the current folder that adds the resource policy to the Lambda function when the Amazon S3 `ObjectCreated` event occurs.

**Skills required**
Cloud architect, Cloud administrator

---

### Create and configure the Lambda search function

**Task**
Create the Lambda execution role.

**Description**
Run the AWS CLI `create-role` command to grant the Lambda search function access to AWS services and resources:

```bash
aws iam create-role
   --role-name search-lambda-role
   --assume-role-policy-document file://lambda_assume_role.json
```

Where `lambda_assume_role.json` is a JSON document in the current folder that grants `AssumeRole` permissions to the Lambda function, as follows:

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

**Skills required**
Cloud architect, Cloud administrator
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach managed policies to the Lambda role.</td>
<td>Run the AWS CLI <code>attach-role-policy</code> command to attach managed policies to the role created in the previous step. These two policies give the role permissions to create an elastic network interface and to write logs to CloudWatch Logs.</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>

```
aws iam attach-role-policy \
--role-name search-lambda-role \
--policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 
aws iam attach-role-policy \
--role-name search-lambda-role \
--policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole
```
### Create the Lambda search function

**Task**
Create the Lambda search function.

**Description**
Run the AWS CLI `create-function` command to create the Lambda search function, which will access Amazon OpenSearch Service:

```bash
aws lambda create-function
  --function-name search-lambda-function
  --zip-file fileb://search_lambda_package.zip
  --handler lambda_search.lambda_handler
  --runtime python3.7
  --role "arn:aws:iam::account-id:role/search-lambda-role"
  --timeout 30
  --vpc-config "{"SubnetIds": ["<subnet-id1>", "<subnet-id2>"], "SecurityGroupIds": ["<sg-1>"]}"
```

**Skills required**
Cloud architect, Cloud administrator

### Create and configure tenant roles

**Task**
Create tenant IAM roles.

**Description**
Run the AWS CLI `create-role` command to create two tenant roles that will be used to test the search functionality:

```bash
aws iam create-role
  --role-name Tenant-1-role
  --assume-role-policy-document file://assume-role-policy.json

aws iam create-role
  --role-name Tenant-2-role
  --assume-role-policy-document file://assume-role-policy.json
```

**Skills required**
Cloud architect, Cloud administrator
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{   &quot;Version&quot;: &quot;2012-10-17&quot;,   &quot;Statement&quot;: [   {   &quot;Effect&quot;: &quot;Allow&quot;,   &quot;Principal&quot;: {   &quot;AWS&quot;: &quot;&lt;Lambda execution role for index function&gt;&quot;,   &quot;AWS&quot;: &quot;&lt;Lambda execution role for search function&gt;&quot;   },   &quot;Action&quot;: &quot;sts:AssumeRole&quot;   } ] }</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AWS Prescriptive Guidance Patterns
Epics
Create a tenant IAM policy.

Run the AWS CLI `create-policy` command to create a tenant policy that grants access to Elasticsearch operations:

```
aws iam create-policy
  --policy-name tenant-policy
  --policy-document file://policy.json
```

The file `policy.json` is a JSON document in the current folder that grants permissions on Elasticsearch:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "es:ESHttpDelete",
        "es:ESHttpGet",
        "es:ESHttpHead",
        "es:ESHttpPost",
        "es:ESHttpPut",
        "es:ESHttpPatch"
      ],
      "Resource": [<ARN of Elasticsearch domain created earlier>]
    }
  ]
}
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Attach the tenant IAM policy to the tenant roles. | Run the AWS CLI `attach-role-policy` command to attach the tenant IAM policy to the two tenant roles you created in the earlier step:  
```
aws iam attach-role-policy \
  --policy-arn 
  arn:aws:iam::account-id:policy/tenant-policy \
  --role-name Tenant-1-role
aws iam attach-role-policy \
  --policy-arn 
  arn:aws:iam::account-id:policy/tenant-policy \
  --role-name Tenant-2-role
```
The policy ARN is from the output of the previous step. | Cloud architect, Cloud administrator |
| Create an IAM policy to give Lambda permissions to assume role. | Run the AWS CLI `create-policy` command to create a policy for Lambda to assume the tenant role:  
```
aws iam create-policy \
  --policy-name assume-tenant-role-policy \
  --policy-document file://lambda_policy.json
```
The file `lambda_policy.json` is a JSON document in the current folder that grants permissions to `AssumeRole`:  
```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "<ARN of tenant role created earlier>"
    }
  ]
}
```
For Resource, you can use a wildcard character to avoid creating a new policy for each tenant. | Cloud architect, Cloud administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an IAM policy to give the Lambda index role permission to access Amazon S3. | Run the AWS CLI `create-policy` command to give the Lambda index role permission to access the objects in the S3 bucket:  
```bash
aws iam create-policy \
  --policy-name s3-permission-policy \
  --policy-document file://s3_lambda_policy.json
```
|  | The file `s3_lambda_policy.json` is the following JSON policy document in the current folder:  
```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::tenantrawdata/*"
    }
  ]
}
```
<p>|  | Cloud architect, Cloud administrator |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Attach the policy to the Lambda execution role. | Run the AWS CLI `attach-role-policy` command to attach the policy created in the previous step to the Lambda index and search execution roles you created earlier:  
```bash  
aws iam attach-role-policy 
   --policy-arn  
   arn:aws:iam::account-id:policy/assume-tenant-role-policy 
   --role-name index-lambda-role 
aws iam attach-role-policy 
   --policy-arn  
   arn:aws:iam::account-id:policy/assume-tenant-role-policy 
   --role-name search-lambda-role 
aws iam attach-role-policy 
   --policy-arn  
   arn:aws:iam::account-id:policy/s3-permission-policy 
   --role-name index-lambda-role 
```
| Cloud architect, Cloud administrator |
| **Create and configure a search API** | The policy ARN is from the output of the previous step. |
| Create a REST API in API Gateway. | Run the CLI `create-rest-api` command to create a REST API resource:  
```bash  
aws apigateway create-rest-api 
   --name Test-Api 
   --endpoint-configuration  
   "{ "types": ["REGIONAL \"] }" 
```
<p>| Cloud architect, Cloud administrator |
| For the endpoint configuration type, you can specify EDGE instead of REGIONAL to use edge locations instead of a particular AWS Region. |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a resource for the search API.</td>
<td>The search API resource starts the Lambda search function with the resource name search. (You don't have to create an API for the Lambda index function, because it runs automatically when objects are uploaded to the S3 bucket.)</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>

1. Run the AWS CLI `get-resources` command to get the parent ID for the root path:

   ```bash
   aws apigateway get-resources \
   --rest-api-id <API-ID>
   ```

   Note the value of the ID field. You will use this parent ID in the next command.

   ```json
   {
   "items": [
   {
   "id": "zpsri964ck",
   "path": "/
   
   
   }
   ]
   }
   ```

2. Run the AWS CLI `create-resource` command to create a resource for the search API. For `parent-id`, specify the ID from the previous command.

   ```bash
   aws apigateway create-resource \
   --rest-api-id <API-ID> \
   --parent-id <Parent-ID> \
   --path-part search
   ```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a GET method for the search API. | Run the AWS CLI `put-method` command to create a GET method for the search API:  
```bash
aws apigateway put-method 
  --rest-api-id <API-ID> 
  --resource-id <ID from the previous command output> 
  --http-method GET 
  --authorization-type "NONE" 
  --no-api-key-required
```
For resource-id, specify the ID from the output of the `create-resource` command. | Cloud architect, Cloud administrator |
| Create a method response for the search API. | Run the AWS CLI `put-method-response` command to add a method response for the search API:  
```bash
aws apigateway put-method-response 
  --rest-api-id <API-ID> 
  --resource-id <ID from the create-resource command output> 
  --http-method GET 
  --status-code 200 
  --response-models "{"application/json": 
  \"Empty\"}"
```
For resource-id, specify the ID from the output of the earlier `create-resource` command. | Cloud architect, Cloud administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up a proxy Lambda integration for the search API.</td>
<td>Run the AWS CLI command <code>put-integration</code> command to set up an integration with the Lambda search function:</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|  | `aws apigateway put-integration \ 
|--rest-api-id <API-ID> \ 
|--resource-id <ID from the create-resource command output> \ 
|--http-method GET \ 
|--type AWS_PROXY \ 
|--integration-http-method GET \ 
|  | For resource-id, specify the ID from the earlier create-resource command. | |
| Grant API Gateway permission to call the Lambda search function. | Run the AWS CLI add-permission command to give API Gateway permission to use the search function: | Cloud architect, Cloud administrator |
|  | | |
|  | `aws lambda add-permission \ 
|--function-name <function-name> \ 
|--statement-id apigateway-get \ 
|--action lambda:InvokeFunction \ 
|--principal apigateway.amazonaws.com \ 
|  | Change the source-arn path if you used a different API resource name instead of search. | |
### Deploy the search API.

**Description**
Run the AWS CLI `create-deployment` command to create a stage resource named `dev`:

```bash
aws apigateway create-deployment \
  --rest-api-id <API-ID> \
  --stage-name dev
```

If you update the API, you can use the same CLI command to redeploy it to the same stage.

**Skills required**
Cloud architect, Cloud administrator

---

### Create and configure Kibana roles

**Task**
Log in to the Kibana console.

1. Find the link to Kibana on your domain dashboard on the Amazon OpenSearch Service console. The URL is in the form: `<domain-endpoint>/_plugin/kibana/`.
2. Use the bastion host you configured in the first epic to access the Kibana console.
3. Log in to the Kibana console by using the master user name and password from the earlier step, when you created the Amazon OpenSearch Service domain.
4. When prompted to select a tenant, choose `Private`.

**Skills required**
Cloud architect, Cloud administrator

---

**Task**
Create and configure Kibana roles.

To provide data isolation and to make sure that one tenant cannot retrieve the data of another tenant, you need to use document security, which allows tenants to access only documents that contain their tenant ID.

1. On the Kibana console, in the navigation pane, choose `Security, Role`.
2. Create a new tenant role.
3. Set cluster permissions to `indices_all`, which gives create, read, update, and
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>delete (CRUD) permissions on the Amazon OpenSearch Service index.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Restrict index permissions to the <strong>tenant-data</strong> index. (The index name should match the name in the Lambda search and index functions.)</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Set index permissions to <strong>indices_all</strong>, to enable users to perform all index-related operations. (You can restrict operations for more granular access, depending on your requirements.)</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>For document-level security, use the following policy to filter documents by tenant ID, to provide data isolation for tenants in a shared index:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;bool&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;must&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;match&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;TenantId&quot;: &quot;Tenant-1&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The index name, properties, and values are case-sensitive.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Map users to roles.</td>
<td>1. Choose the <strong>Mapped users</strong> tab for the role, and then choose <strong>Map users</strong>. 2. In the <strong>Backend roles</strong> section, specify the ARN of the IAM tenant role that you created earlier, and then choose <strong>Map</strong>. This maps the IAM tenant role to the Kibana role so that tenant-specific search returns data for that tenant only. For example, if the IAM role name for Tenant-1 is <strong>Tenant-1-Role</strong>, specify the ARN for <strong>Tenant-1-Role</strong> (from the <strong>Create and configure tenant roles</strong> epic) in the <strong>Backend roles</strong> box for the Tenant-1 Kibana role. 3. Repeat steps 1 and 2 for Tenant-2. We recommend that you automate the creation of the tenant and Kibana roles at the time of tenant onboarding.</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>
| Create the tenant-data index. | In the navigation pane, under **Management**, choose **Dev Tools**, and then run the following command. This command creates the **tenant-data** index to define the mapping for the **TenantId** property.  

```json
PUT /tenant-data
{
    "mappings": {
        "properties": {
            "TenantId": { "type": "keyword"}
        }
    }
}
``` | Cloud architect, Cloud administrator |

### Create VPC endpoints for Amazon S3 and AWS STS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC endpoint for Amazon S3.</td>
<td>Run the AWS CLI <code>create-vpc-endpoint</code> command to create a VPC endpoint for Amazon</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>S3. The endpoint enables the Lambda index function in the VPC to access the Amazon S3 service.</td>
<td>aws ec2 create-vpc-endpoint --vpc-id &lt;VPC-ID&gt; --service-name com.amazonaws.us-east-1.s3 --route-table-ids &lt;route-table-ID&gt;</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td>Create a VPC endpoint for AWS STS.</td>
<td>Run the AWS CLI create-vpc-endpoint command to create a VPC endpoint for AWS Security Token Service (AWS STS). The endpoint enables the Lambda index and search functions in the VPC to access the AWS STS service. The functions use AWS STS when they assume the IAM role.</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</tbody>
</table>

aws ec2 create-vpc-endpoint

-vpc-id <VPC-ID>

--vpc-endpoint-type Interface

--service-name com.amazonaws.us-east-1.sts

--subnet-id <subnet-ID>

--security-group-id <security-group-ID>

For vpc-id, specify the VPC that you're using for the Lambda index function. For service-name, use the correct URL for the Amazon S3 endpoint. For route-table-ids, specify the route table that's associated with the VPC endpoint.

For vpc-id, specify the VPC that you're using for the Lambda index and search functions. For subnet-id, provide the subnet in which this endpoint should be created. For security-group-id, specify the security group to associate this endpoint with. (It could be the same as the security group Lambda uses.)
## Test multi-tenancy and data isolation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Update the Python files for the index and search functions. | 1. In the `index_lambda_package.zip` file, edit the `lambda_index.py` file to update the AWS account ID, AWS Region, and Elasticsearch endpoint information.  
2. In the `search_lambda_package.zip` file, edit the `lambda_search.py` file to update the AWS account ID, AWS Region, and Elasticsearch endpoint information.  
You can get the Elasticsearch endpoint from the **Overview** tab of the Amazon OpenSearch Service console. It has the format `<AWS-Region>.es.amazonaws.com`. | Cloud architect, App developer            |
| Update the Lambda code.           | Use the AWS CLI `update-function-code` command to update the Lambda code with the changes you made to the Python files:  
```bash  
aws lambda update-function-code  
--function-name index-lambda-function  
--zip-file fileb://index_lambda_package.zip  
aws lambda update-function-code  
--function-name search-lambda-function  
--zip-file fileb://search_lambda_package.zip  
``` | Cloud architect, App developer            |
| Upload raw data to the S3 bucket. | Use the AWS CLI `cp` command to upload data for the Tenant-1 and Tenant-2 objects to the `tenantrawdata` bucket (specify the name of the S3 bucket you created for this purpose):  
```bash  
aws s3 cp tenant-1-data s3://tenantrawdata  
``` | Cloud architect, Cloud administrator     |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>aws s3 cp tenant-2-data s3://tenantrawdata</strong></td>
<td>The S3 bucket is set up to run the Lambda index function whenever data is uploaded so that the document is indexed in Elasticsearch.</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
<tr>
<td><strong>Search data from the Kibana console.</strong></td>
<td>On the Kibana console, run the following query:</td>
<td></td>
</tr>
<tr>
<td><strong>GET tenant-data/_search</strong></td>
<td>This query displays all the documents indexed in Elasticsearch. In this case, you should see two, separate documents for Tenant-1 and Tenant-2.</td>
<td></td>
</tr>
</tbody>
</table>
| **Test the search API from API Gateway.** | 1. In the API Gateway console, open the search API, choose the GET method inside the search resource, and then choose Test.  
2. In the test window, provide the following query string (case-sensitive) for the tenant ID, and then choose Test. | Cloud architect, App developer |
| | **TenantId=Tenant-1** | The Lambda function sends a query to Amazon OpenSearch Service that filters the tenant document based on the document-level security. The method returns the document that belongs to Tenant-1.  
3. Change the query string to: | |
| | **TenantId=Tenant-2** | This query returns the document that belongs to Tenant-2. | |

For screen illustrations, see the Additional information (p. 1385) section.
Related resources

- AWS SDK for Python (Boto3)
- AWS Lambda documentation
- Amazon API Gateway documentation
- Amazon S3 documentation
- Amazon OpenSearch Service documentation
  - Fine-grained access control in Amazon OpenSearch Service
  - Creating a search application with Amazon OpenSearch Service
  - Launching your Amazon OpenSearch Service domains within a VPC

Additional information

Data partitioning models

There are three common data partitioning models used in multi-tenant systems: silo, pool, and hybrid. The model you choose depends on the compliance, noisy neighbor, operations, and isolation needs of your environment.

Silo model

In the silo model, each tenant's data is stored in a distinct storage area where there is no commingling of tenant data. You can use two approaches to implement the silo model with Amazon OpenSearch Service: domain per tenant and index per tenant.

- **Domain per tenant** – You can use a separate Amazon OpenSearch Service domain (synonymous with an Elasticsearch cluster) per tenant. Placing each tenant in its own domain provides all the benefits associated with having data in a standalone construct. However, this approach introduces management and agility challenges. Its distributed nature makes it harder to aggregate and assess the operational health and activity of tenants. This is a costly option that requires each Amazon OpenSearch Service domain to have three master nodes and two data nodes for production workloads at the minimum.
**Index per tenant** – You can place tenant data in separate indexes within an Amazon OpenSearch Service cluster. With this approach, you use a tenant identifier when you create and name the index, by pre-pending the tenant identifier to the index name. The index per tenant approach helps you achieve your silo goals without introducing a completely separate cluster for each tenant. However, you might encounter memory pressure if the number of indexes grows, because this approach requires more shards, and the master node has to handle more allocation and rebalancing.
Isolation in the silo model – In the silo model, you use IAM policies to isolate the domains or indexes that hold each tenant's data. These policies prevent one tenant from accessing another tenant's data. To implement your silo isolation model, you can create a resource-based policy that controls access to your tenant resource. This is often a domain access policy that specifies which actions a principal can perform on the domain's sub-resources, including Elasticsearch indexes and APIs. With IAM identity-based policies, you can specify allowed or denied actions on the domain, indexes, or APIs within Amazon OpenSearch Service. The Action element of an IAM policy describes the specific action or actions that are allowed or denied by the policy, and the Principal element specifies the affected accounts, users, or roles.

The following sample policy grants Tenant-1 full access (as specified by es:* ) to the sub-resources on the tenant-1 domain only. The trailing /* in the Resource element indicates that this policy applies to the domain's sub-resources, not to the domain itself. When this policy is in effect, tenants are not allowed to create a new domain or modify settings on an existing domain.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::aws-account-id:user/Tenant-1"
            },
            "Action": "es:*",
        }
    ]
}
```

To implement the tenant per Index silo model, you would need to modify this sample policy to further restrict Tenant-1 to the specified index or indexes, by specifying the index name. The following sample policy restricts Tenant-1 to the tenant-index-1 index.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::aws-account-id:user/Tenant-1"
            },
            "Action": "es:*",
        }
    ]
}
```
Pool model

In the pool model, all tenant data is stored in an index within the same domain. The tenant identifier is included in the data (document) and used as the partition key, so you can determine which data belongs to which tenant. This model reduces the management overhead. Operating and managing the pooled index is easier and more efficient than managing multiple indexes. However, because tenant data is commingled within the same index, you lose the natural tenant isolation that the silo model provides. This approach might also degrade performance because of the noisy neighbor effect.

Tenant isolation in the pool model – In general, tenant isolation is challenging to implement in the pool model. The IAM mechanism used with the silo model doesn’t allow you to describe isolation based on the tenant ID stored in your document.

An alternative approach is to use the fine-grained access control (FGAC) support provided by the Open Distro for Elasticsearch. FGAC allows you to control permissions at an index, document, or field level. With each request, FGAC evaluates the user credentials and either authenticates the user or denies access. If FGAC authenticates the user, it fetches all roles mapped to that user and uses the complete set of permissions to determine how to handle the request.

To achieve the required isolation in the pooled model, you can use document-level security, which lets you restrict a role to a subset of documents in an index. The following sample role restricts queries to Tenant-1. By applying this role to Tenant-1, you can achieve the necessary isolation.
Hybrid model

The **hybrid model** uses a combination of the silo and pool models in the same environment to offer unique experiences to each tenant tier (such as free, standard, and premium tiers). Each tier follows the same security profile that was used in the pool model.

Tenant isolation in the hybrid model – In the hybrid model, you follow the same security profile as in the pool model, where using the FGAC security model at the document level provided tenant isolation. Although this strategy simplifies cluster management and offers agility, it complicates other aspects of the architecture. For example, your code requires additional complexity to determine which model is associated with each tenant. You also have to ensure that single-tenant queries don’t saturate the entire domain and degrade the experience for other tenants.

**Testing in API Gateway**

*Test window for Tenant-1 query*
Test window for Tenant-2 query
Attachments

attachment.zip

Cache secrets using AWS Lambda extensions

Created by HARI OHM PRASATH RAJAGOPAL (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Modernization; Cloud-native; Serverless; Software development &amp; testing</th>
<th>Workload:</th>
<th>Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS Lambda; AWS Secrets Manager</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern demonstrates how to host an HTTP server that caches secrets required for an AWS Lambda function. The Lambda extension does the following:
• Hosts a local HTTP server, which is invoked by a Lambda function to retrieve the secret instead of retrieving it directly from AWS Secrets Manager.

• Uses a config.yaml file inside the Lambda function to get the list of secrets that must be cached in memory.

• Refreshes the secrets cache based on the value (in minutes) set in the environment variable CACHE_TIMEOUT. If no value is specified, it defaults to 10.

This extension is helpful in the following use cases:

• If the Lambda function needs access to Secrets Manager for reading a secret, the secret can get cached as part of the extension startup instead of being fetched as part of Lambda's first request. This helps to avoid a cold start.

• The Lambda function specifies which secrets are cached by using the configuration file (config.yaml), which is part of the deployment package.

• The Lambda function allows the expiration of in-memory cache, which can be configured as a Lambda environment variable.

• The business can maintain the cache layer in-memory instead of using a /tmp storage for reading and writing values in Secrets Manager.

Prerequisites and limitations

Prerequisites

• An active AWS account

• AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows

• Zip utility installed in the local system

• The code that is in the Attachments section

Architecture

The following diagram provides a high-level view of all the components.
1. Secrets are cached based on the `config.yaml` file defined inside the deployment package of the Lambda function, and the cache is refreshed based on environment variables.

2. Based on the keys specified inside `config.yaml`, the secrets are read and cached in-memory from Secrets Manager.

3. The Lambda function reads the secret by making an HTTP GET call to the extension, and the extension returns the value from the in-memory cache.

**Initializing the extension and reading secrets from the cache**

The following sequence diagram shows the initialization of the Lambda extension and how the Lambda function reads cached secrets by using the HTTP server that is hosted inside the extension.
The sequence of events inside the Lambda function and the extension includes the following.

1. Part of the initialization extension reads the `config.yaml` file from the Lambda function under the default path (`/var/task`) and gets the list of secrets that need to be cached.
2. When the extension has a list of keys that need to be fetched, it invokes Secrets Manager to get all the secrets (specified in `config.yaml`) and stores them in memory.
3. The extension starts a local HTTP server.
4. To get the secret value, the Lambda function makes an HTTP GET `/cache/<secretname>` call to the extension.
5. The extension returns the cached value to the Lambda function based on the `secretname` path parameter.

Technology

- AWS Lambda
- AWS Secrets Manager

Tools

- AWS CLI – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services.
- AWS Lambda – AWS Lambda supports running code without provisioning or managing servers.
- AWS Secrets Manager – AWS Secrets Manager enables the replacement of hardcoded credentials, including passwords, with an API call to Secrets Manager to retrieve the secret. Because Secrets Manager can automatically rotate the secret according to a schedule, you can replace long-term secrets with short-term ones, reducing the risk of compromise.
• **Git client** – Git provides GUI tools, or you can use the command line or a desktop tool to check out the required artifacts from GitHub.

• **Node.js** – Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications. The **runtime for Node.js** should be installed on the local system.

### Epics

#### Create a secret in Secrets Manager

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a secret in Secrets Manager.</td>
<td>Run the command that is in the <strong>Additional information</strong> section. The command assumes that you placed your secret, such as this example JavaScript Object Notation (JSON) text structure <code>{&quot;username&quot;:&quot;janedoe&quot;,&quot;password&quot;:&quot;aBC1D2!*3EE&quot;}</code>, in a file named <code>mycreds.json</code>.</td>
<td>Developer, AWS systems administrator</td>
</tr>
</tbody>
</table>

#### Create the Lambda function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new Lambda function.</td>
<td>On the AWS Management Console, choose <strong>Lambda</strong> and then choose <strong>Create function</strong>. Create a new Node.js runtime Lambda function with the name <code>Secrets-Extension-Lambda-Test</code>. Increase the memory of Lambda to 1200 MB and <strong>Timeout</strong> to 30 seconds.</td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Assign a policy to the Lambda IAM role.</td>
<td>Choose the <strong>Permission</strong> tab. Under <strong>Execution role</strong>, choose the <strong>Role name</strong> link and attach the <strong>SecretsManagerReadWrite</strong> policy to the AWS Identity and Access Management (IAM) role. <strong>Note:</strong> As a best practice, create an IAM policy with read-only permission on the secret that needs to be cached, and assign it to the IAM role.</td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Update the handler code.</td>
<td>Choose the <strong>Configuration</strong> tab and update the function with the code in the <strong>Additional information</strong> section. The code invokes the local server hosted</td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>inside the Lambda extension to read the value of the secret instead of directly going to Secrets Manager.</td>
<td></td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Create a secrets cache configuration file.</td>
<td>Create a new file named config.yaml under the root directory with the contents specified in the Additional information section. The value secret_now is the name of the secret that you created in the previous step and that will be cached by the extension. If you want more secrets to be cached, you can keep adding them here. Choose the Deploy button.</td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Create the cache, rebuild timeout, and update the Lambda runtime.</td>
<td>Create a new environment variable CACHE_TIMEOUT, and set the value in minutes. The cache will be refreshed based on this value. Note: If the environment value is not found, the cache gets refreshed every 10 minutes.</td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Deploy the Lambda extension</td>
<td>To deploy the extension and associate the layer to the Lambda function, run the command that is in the Additional information section.</td>
<td>Developer, AWS systems administrator</td>
</tr>
<tr>
<td>Invoke the Lambda function</td>
<td>To invoke the Lambda function, use the command that is in the Additional information section. The function should return StatusCode:200.</td>
<td>Developer, AWS systems administrator</td>
</tr>
</tbody>
</table>
Verify Lambda extension invocation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Lambda extension invocation.</td>
<td>On the Amazon CloudWatch console, navigate to Logs, Log Groups. Select the log group /aws/lambda/Secrets-Extension-Lambda-Test. View the log stream to see the runtime log with Response from cache followed by the user name and password stored in mycred.json.</td>
<td>Developer, AWS systems administrator</td>
</tr>
</tbody>
</table>

Related resources

- Building Extensions for AWS Lambda – In preview
- AWS Lambda extension samples
- AWS CLI command reference for Secrets Manager
- AWS Lambda Getting Started

Additional information

Create secret

The following command assumes that you placed your secret, such as this example JSON text structure \{"username": "janedoe", "password": "aBC1D2*!3EE"\}, in a file named mycreds.json.

```bash
aws secretsmanager create-secret --name secret_now --secret-string file://mycreds.json
```

Update the handler code

A sample with the following code is in example-function/index.js in the zipped files that are provided in the Attachments section.

```javascript
exports.handler = function(event, context, callback) {
  const https = require('http')
  const options = {
    hostname: 'localhost',
    port: 8080,
    path: '/cache/secret_now',
    method: 'GET'
  }
  const req = https.request(options, res => {
    res.on('data', d => {
      console.log("Response from cache: "+d);
      return d;
    })
  })
}
```
Create a secrets cache configuration file

A sample of the following code is in the example-function/config.yaml file in the Attachments section. In the example, secret_now is the name of the secret that was created in the previous step and will be cached by the extension. To cache additional secrets, add them to this file.

```
SecretManagers:
  - secrets:
    - secret_now
```

Deploy the Lambda extension

```
> chmod +x deploy.sh
> ./deploy.sh
```

Invoke the Lambda function

```
aws lambda invoke \\
  --function-name "Secrets-Extension-Lambda-Test" \\
  --payload '{"payload": "hello"}' /tmp/invoke-result \\
  --cli-binary-format raw-in-base64-out \\
  --log-type Tail
```

The function should return "StatusCode": 200.

Attachments

attachment.zip

Convert and unpack data from EBCDIC to ASCII

Created by Luis Gustavo Dantas (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: Production</th>
<th>Source: Mainframe EBCDIC data</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Mainframe Data Utilities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target: Distributed/cloud modernized ASCII data</th>
<th>R Type: N/A</th>
<th>Workload: IBM</th>
</tr>
</thead>
</table>
Summary

Because mainframes typically host critical business data, modernizing data is one of the most important tasks when migrating data to the Amazon Web Services (AWS) Cloud or other American Standard Code for Information Interchange (ASCII) environment. On mainframes, data is typically encoded in extended binary-coded decimal interchange code (EBCDIC) format. Exporting database, Virtual Storage Access Method (VSAM), or flat files generally produces packed, binary EBCDIC files, which are more complex to migrate. The most commonly used database migration solution is change data capture (CDC), which, in most cases, automatically converts data encoding. However, CDC mechanisms may not be available for these database, VSAM, or flat files. For these files, an alternative approach is required to modernize the data.

This pattern describes how to modernize EBCDIC data by converting it to ASCII format. After conversion, you can load the data into modern databases or have applications in the cloud process the data directly. The pattern uses the conversion script and sample files in the mainframe-data-utilities GitHub repository.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An EBCDIC input file and its corresponding common business-oriented language (COBOL) copybook. A sample EBCDIC file and COBOL copybook are included in the mainframe-data-utilities GitHub repository. For more information about COBOL copybooks, see Enterprise COBOL for z/OS 6.3 Programming Guide on the IBM website.

Limitations

- The current Python solution doesn’t support multi-layout files.
- File layouts defined inside COBOL programs are not supported. They must be made available separately.

Architecture

Source technology stack

- EBCDIC data on a mainframe
- COBOL copybook

Target technology stack

- Amazon Elastic Compute Cloud (Amazon EC2) instance in a virtual private cloud (VPC)
- Amazon Elastic Block Store (Amazon EBS)
- Python and its required packages, JavaScript Object Notation (JSON), sys, and datetime
- ASCII flat file ready to be read by a modern application or loaded in a relational database table

Target architecture
The architecture diagram shows the process of converting an EBCDIC file to an ASCII file on an EC2 instance:

1. Using the `parse_copybook_to_json.py` script, you convert the COBOL copybook to a JSON file.
2. Using the JSON file and the `extract_ebcdic_to_ascii.py` script, you convert the EBCDIC data to an ASCII file.

**Automation and scale**

After the resources needed for the first manual file conversions are in place, you can automate file conversion. This pattern doesn’t include instructions for automation. There are multiple ways to automate the conversion. The following is an overview of one possible approach:

1. Encapsulate the AWS Command Line Interface (AWS CLI) and Python script commands into a shell script.
2. Create an AWS Lambda function that asynchronously submits the shell script job into an EC2 instance. For more information, see Scheduling SSH jobs using AWS Lambda.

3. Create an Amazon Simple Storage Service (Amazon S3) trigger that invokes the Lambda function every time a legacy file is uploaded. For more information, see Using an Amazon S3 trigger to invoke a Lambda function.

**Tools**

**AWS services**

- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.
- **Amazon EBS** – Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with EC2 instances.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent interface for interacting with all parts of AWS.
- **AWS Identity and Access Management** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services.

**Other tools**

- **GitHub** – GitHub is a code-hosting service that provides collaboration tools and version control.
- **Python** – Python is a high-level programming language.

**Code**

The code for this pattern is available in the mainframe-data-utilities GitHub repository.

**Epics**

**Prepare the EC2 instance**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch an EC2 instance.</td>
<td>The EC2 instance must be running Linux and have outbound internet access. This allows the instance to access the Python source code available on GitHub. To create the instance:</td>
<td>General AWS</td>
</tr>
<tr>
<td></td>
<td>2. Launch an EC2 Linux instance. Use a public IP address and allow inbound access through port 22. Ensure that the storage size of the instance is at least twice the size of the EBCDIC data file. For</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| **Install Python and Git.** | 1. Using a secure shell (SSH) client, connect to the EC2 instance you just launched. For more information, see [Connect to your Linux instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-ssh.html).  
2. Install Python 3 or later from the Python website.  
3. In the Amazon EC2 console, run the following command. This installs Git on the EC2 instance.  
   
   ```bash
   sudo yum install git
   ```  
4. Run the following command and confirm that Git has been successfully installed.  
   
   ```bash
   git --version
   ``` | General AWS, Linux |
| **Clone the GitHub repository.** | 1. In the Amazon EC2 console, run the following command. This clones the `mainframe-data-utilities` repository from GitHub and opens the default copy location, the home folder.  
   
   ```bash
   git clone git://github.com/aws-samples/mainframe-data-utilities.git
   ```  
2. In the home folder, confirm that the `mainframe-data-utilities` folder is present. | General AWS, GitHub |

---

**Create the ASCII file from the EBCDIC data**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parse the COBOL copybook into the JSON layout file.</strong></td>
<td>Inside the <code>mainframe-data-utilities</code> folder, run the <code>parse_copybook_to_json.py</code> script. This automation module reads the file layout from a COBOL copybook and creates a JSON file. The JSON file contains the information required to</td>
<td>General AWS, Linux</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>interpret and extract the data from the source file. This creates the JSON metadata from the COBOL copybook. Use the command from <strong>Parse COBOL copybook to JSON</strong> in the Additional information section.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The script prints the received arguments.</td>
<td></td>
</tr>
</tbody>
</table>

---

Copybook file............| LegacyReference/ COBPACK2.cpy
Parsed copybook (JSON List).| sample-data/ cobpack2-list.json
JSON Dict (documentation)...| sample-data/cobpack2-dict.json
ASCII file...............| sample-data/ COBPACK.ASCII.txt
EBCDIC file...............| sample-data/ COBPACK.OUTFILE.txt
Print each.................| 10000

For more information about the arguments, see the README file in the GitHub repository.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Inspect the JSON layout file. | 1. Navigate to the output path defined in the `parse_copybook_to_json.py` script.  
2. Check the creation time of the `sample-data/cobpack2-list.json` file to confirm that you have selected the appropriate JSON layout file.  
3. Examine the JSON file and confirm that the contents are similar to the following:  

```
"input": "extract-ebcdic-to-ascii/COBPACK.OUTFILE.txt",
"output": "extract-ebcdic-to-ascii/COBPACK.ASCII.txt",
"max": 0,
"skip": 0,
"print": 10000,
"lrecl": 150,
"rem-low-values": true,
"separator": "|",
"transf": [  
  {  
    "type": "ch",
    "bytes": 19,
    "name": "OUTFILE-TEXT"
  }
```

The most important attributes of the JSON layout file are:  
- **input** – Contains the path of the EBCDIC file to be converted  
- **output** – Defines the path where the ASCII file will be generated  
- **lrecl** – Specifies the size in bytes of the logical record length  
- **transf** – Lists all fields and their size in bytes

For more information about the JSON layout file, see the [README file](#) in the GitHub repository. | General AWS, JSON |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the ASCII file.</td>
<td>Run the <code>extract_ebcdic_to_ascii.py</code> script, which is included in the cloned GitHub repository. This script reads the EBCDIC file and writes a converted and readable ASCII file.</td>
<td>General AWS</td>
</tr>
<tr>
<td></td>
<td>python3 <code>extract_ebcdic_to_ascii.py -local-json sample-data/cobpack2-list.json</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>As the script processes the EBCDIC data, it prints a message for every batch of 10,000 records. See the following example.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parameter file: sample-data/cobpack2-list.json</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:28.551353</td>
<td>Records processed: 10000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:29.045221</td>
<td>Records processed: 20000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:29.539174</td>
<td>Records processed: 30000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:30.030418</td>
<td>Records processed: 40000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:30.521879</td>
<td>Records processed: 50000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:30.994222</td>
<td>Records processed: 60000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:31.464385</td>
<td>Records processed: 70000</td>
</tr>
<tr>
<td></td>
<td>2021-09-06 16:22:31.934501</td>
<td>Records processed: 80000</td>
</tr>
<tr>
<td></td>
<td>For information about how to change the print frequency, see the README file in the GitHub repository.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Examine the ASCII file.</td>
<td>1. Check the creation time of the <code>extract-ebcdic-to-ascii/COBPACK.ASCII.txt</code> file to verify that it was recently created.</td>
<td>General AWS, Linux</td>
</tr>
<tr>
<td></td>
<td>2. In the Amazon EC2 console, enter the following command. This opens the first record of the ASCII file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>head sample-data/COBPACK.ASCII.txt -n 1</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Examine the contents of the first record. Because EBCDIC files are usually binary, they don't have carriage return and line feed (CRLF) special characters. The <code>extract_ebcdic_to_ascii.py</code> script adds a pipe character as a column separator, which is defined in the script parameters. If you used the sample EBCDIC data provided, for the result, see Sample ASCII data in the Additional information section.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Evaluate the EBCDIC file.</td>
<td>In the Amazon EC2 console, enter the following command. This opens the first record of the EBCDIC file.</td>
<td>General AWS, Linux, EBCDIC</td>
</tr>
<tr>
<td></td>
<td>head sample-data/COBPACK.OUTFILE.txt -c 150</td>
<td>xxd</td>
</tr>
<tr>
<td></td>
<td>If you used the sample EBCDIC file, the following is the result.</td>
<td></td>
</tr>
</tbody>
</table>

To evaluate the equivalence between the source and target files, comprehensive knowledge of EBCDIC is required. For example, the first character of the sample EBCDIC file is a hyphen (−). In hexadecimal notation of the EBCDIC file, this character is represented by 60, and in hexadecimal notation of the ASCII file, this character is represented by 2D. For an EBCDIC-to-ASCII conversion table, see [EBCDIC to ASCII](https://www.ibm.com) on the IBM website.
Related resources

References
- The EBCDIC character set (IBM documentation)
- EBCDIC to ASCII (IBM documentation)
- COBOL (IBM documentation)
- Basic JCL concepts (IBM documentation)
- Connect to your Linux instance (Amazon EC2 documentation)

Tutorials
- Scheduling SSH jobs using AWS Lambda (AWS blog post)
- Using an Amazon S3 trigger to invoke a Lambda function (AWS Lambda documentation)

Additional information

Parse COBOL copybook to JSON
The following command converts the COBOL copybook to a JSON file.

```
```

Sample ASCII data
If you used the sample EBCDIC file provided, the following is the first record in the ASCII file.

```
-000000000100000000|000000000100000000|-000000000100000000|0|0|
100000000|-100000000|100000000|-100000000|00000|00000|100000000|-100000000|
0000000000100000000|-0000000000100000000|A|A||
```

Deploy multiple-stack applications using AWS CDK with TypeScript

*Created by Rahul Gaikwad (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Modernization; Migration; DevOps</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon API Gateway; AWS Lambda; Amazon Kinesis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

This pattern provides a step-by-step approach for application deployment on Amazon Web Services (AWS) using AWS Cloud Development Kit (AWS CDK) with TypeScript. As an example, the pattern deploys a serverless real-time analytics application.

The pattern builds and deploys nested stack applications. The parent AWS CloudFormation stack calls the child, or nested, stacks. Each child stack builds and deploys the AWS resources that are defined in the CloudFormation stack. AWS CDK Toolkit, the command line interface (CLI) command `cdk`, is the primary interface for the CloudFormation stacks.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Existing virtual private cloud (VPC) and subnets
- AWS CDK Toolkit installed and configured
- A user with administrator permissions and a set of access keys.
- Node.js
- AWS Command Line Interface (AWS CLI)

Limitations

- Because AWS CDK uses AWS CloudFormation, AWS CDK applications are subject to CloudFormation service quotas. For more information, see AWS CloudFormation quotas.

Product versions

This pattern has been built and tested using the following tools and versions.

- AWS CDK Toolkit 1.83.0
- Node.js 14.13.0
- npm 7.0.14

The pattern should work with any version of AWS CDK or npm. Note that Node.js versions 13.0.0 through 13.6.0 are not compatible with the AWS CDK.

Architecture

Target technology stack

- AWS Amplify Console
- Amazon API Gateway
- AWS CDK
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- AWS Lambda
• Amazon Simple Storage Service (Amazon S3)

**Target architecture**

The following diagram shows multiple-stack application deployment using AWS CDK with TypeScript.

The following diagram shows the architecture of the example serverless real-time application.

**Tools**
- **AWS Amplify Console** – The AWS Amplify Console is the control center for fullstack web and mobile application deployments in AWS. Amplify Console hosting provides a git-based workflow for hosting fullstack serverless web apps with continuous deployment. The Admin UI is a visual interface for frontend web and mobile developers to create and manage app backends outside the AWS console.
- **Amazon API Gateway** – Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- **AWS CDK** – AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- **AWS CDK Toolkit** – The AWS CDK Toolkit, the CLI command `cdk`, is the primary tool for interacting with your AWS CDK app. It runs your app, interrogates the application model you defined, and produces and deploys the AWS CloudFormation templates generated by the AWS CDK.
- **Amazon CloudFront** – Amazon CloudFront is a web service that speeds up distribution of static and dynamic web content, such as .html, .css, .js, and image files. CloudFront delivers your content through a worldwide network of data centers called edge locations for lower latency and improved performance.
- **Amazon Cognito** – Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party.
- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- **Amazon Kinesis Data Firehose** – Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service (successor to Amazon Elasticsearch Service), Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers.
- **Amazon Kinesis Data Streams** – Amazon Kinesis Data Streams is a service for collecting and processing large streams of data records in real time.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.

**Code**

The code for this pattern is attached.

**Epics**

**Install AWS CDK Toolkit**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install AWS CDK Toolkit.</td>
<td>To install AWS CDK Toolkit globally, run the following command.</td>
<td>DevOps</td>
</tr>
<tr>
<td></td>
<td><code>npm install -g aws-cdk</code></td>
<td></td>
</tr>
<tr>
<td>Verify the version.</td>
<td>To verify the AWS CDK Toolkit version, run the following command.</td>
<td>DevOps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Epics

### Task Description Skills required

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdk --version</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Set up AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up credentials.</td>
<td>To set up credentials, run the aws configure command and follow the prompts.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
| | $aws configure  
AWS Access Key ID [None]:  
AWS Secret Access Key [None]:  
your_secret_access_key  
Default region name [None]:  
Default output format [None]: | |

### Download the project code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the attached project code.</td>
<td>For more information about the directory and file structure, see the Additional information section.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Bootstrap the AWS CDK environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Bootstrap the environment. | To deploy the AWS CloudFormation template to the account and AWS Region that you want to use, run the following command.  
  cdk bootstrap <account>/<Region>  
  For more information, see the AWS documentation. | DevOps |
## Build and deploy the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the project.</td>
<td>To build the project code, run the <code>npm run build</code> command.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Deploy the project.</td>
<td>To deploy the project code, run the <code>cdk deploy</code> command.</td>
<td></td>
</tr>
</tbody>
</table>

## Verify outputs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify stack creation.</td>
<td>On the AWS Management Console, choose <strong>CloudFormation</strong>. In the stacks for the project, verify that a parent stack and two child stacks have been created.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

## Test the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send data to Kinesis Data Streams.</td>
<td>Configure your AWS Account to send data to Kinesis Data Streams using Amazon Kinesis Data Generator (KDG). For more information, see Amazon Kinesis Data Generator.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Create an Amazon Cognito user.</td>
<td>To create an Amazon Cognito user, download the cognito-setup.json CloudFormation template from the Create an Amazon Cognito User section on the Kinesis Data Generator help page. Initiate the template, and then enter your Amazon Cognito Username and Password. The Outputs tab lists the Kinesis Data Generator URL.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Log in to Kinesis Data Generator</td>
<td>To log in to KDG, use the Amazon Cognito credentials that you provided and the Kinesis Data Generator URL.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Test the application.</td>
<td>In KDG, in Record template, <strong>Template 1</strong>, paste the test code from the Additional information section, and choose <strong>Send data</strong>.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test API Gateway.</td>
<td>After the data has been ingested, test API Gateway by using the GET method to retrieve data.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Related resources

#### References
- AWS Cloud Development Kit
- AWS CDK on GitHub
- Working with nested stacks
- AWS sample example - Serverless real-time analytics

### Additional information

#### Directory and file details
This pattern sets up the following three stacks.

- **parent-cdk-stack.ts** – This stack acts as the parent stack and calls the two child applications as nested stacks.
- **real-time-analytics-poc-stack.ts** – This nested stack contains the infrastructure and application code.
- **real-time-analytics-web-stack.ts** – This nested stack contains only the static web application code.

**Important files and their functionality**

- **bin/real-time-analytics-poc.ts** – Entry point of the AWS CDK application. It loads all stacks defined under lib/.
- **lib/real-time-analytics-poc-stack.ts** – Definition of the AWS CDK application's stack (real-time-analytics-poc).
- **lib/real-time-analytics-web-stack.ts** – Definition of the AWS CDK application's stack (real-time-analytics-web-stack).
- **lib/parent-cdk-stack.ts** – Definition of the AWS CDK application's stack (parent-cdk).
- **package.json** – npm module manifest, which includes the application name, version, and dependencies.
- **package-lock.json** – Maintained by npm.
- **cdk.json** – Toolkit for running the application.
- **tsconfig.json** – The project’s TypeScript configuration.
- **.gitignore** – List of files that Git should exclude from source control.
- **node_modules** – Maintained by npm; includes the project’s dependencies.

The following section of code in the parent stack calls child applications as a nested AWS CDK stacks.

```javascript
import * as cdk from '@aws-cdk/core';
import { Construct, Stack, StackProps } from '@aws-cdk/core';
```
import { RealTimeAnalyticsPocStack } from './real-time-analytics-poc-stack';
import { RealTimeAnalyticsWebStack } from './real-time-analytics-web-stack';

export class CdkParentStack extends Stack {
    constructor(scope: Construct, id: string, props?: StackProps) {
        super(scope, id, props);

        new RealTimeAnalyticsPocStack(this, 'RealTimeAnalyticsPocStack');
        new RealTimeAnalyticsWebStack(this, 'RealTimeAnalyticsWebStack');
    }
}

Code for testing

session={{date.now('YYYYMMDD')}}|sequence={{date.now('x')}}|
reception={{date.now('x')}}|instrument={{random.number(9)}}|l1={{random.number(20)}}|
price_0={{random.number({'min':10000, 'max':30000})}}|price_1={{random.number({'min':10000, 'max':30000})}}|
price_2={{random.number({'min':10000, 'max':30000})}}|price_3={{random.number({'min':10000, 'max':30000})}}|
price_4={{random.number({'min':10000, 'max':30000})}}|price_5={{random.number({'min':10000, 'max':30000})}}|
price_6={{random.number({'min':10000, 'max':30000})}}|price_7={{random.number({'min':10000, 'max':30000})}}|
price_8={{random.number({'min':10000, 'max':30000})}}|

Testing API Gateway

On the API Gateway console, test API Gateway by using the GET method.

APIs > analytics-api (path: /data) > Resources > (data [77]) > GET

Path
No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.

Query Strings

- date

Headers

- date

Stage Variables

No Stage variables exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request body is not supported for GET methods.

Attachments

attachment.zip
Automate deployment of nested applications using AWS SAM

Summary

You can build serverless applications on the Amazon Web Services (AWS) Cloud by using the AWS Serverless Application Model (AWS SAM). An open-source framework, AWS SAM provides shorthand syntax to express functions, APIs, databases, and event source mappings. With a few lines of configuration, you can define the application you want and model it.

This pattern uses AWS SAM templates to automate the deployment of nested applications. A nested application is an application within another application. Parent applications call their child applications. These are loosely coupled components of a serverless architecture.

Using nested applications, you can rapidly build highly sophisticated serverless architectures by reusing services or components that are independently authored and maintained but are composed using AWS SAM and the Serverless Application Repository. Nested applications help you to build applications that are more powerful, avoid duplicated work, and ensure consistency and best practices across your teams and organizations. To demonstrate nested applications, the pattern deploys a sample AWS serverless shopping cart application.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An existing virtual private cloud (VPC) and subnets
- An integrated development environment, such as AWS Cloud9 or Visual Studio Code (for more information, see Start building with SDKs and tools)
- Python wheel library installed using `pip install wheel`, if it's not already installed

Limitations

- The maximum number of applications that can be nested in a serverless application is 200.
- The maximum number of parameters for a nested application can have 60.

Product versions

- This solution is built on AWS SAM command line interface (AWS SAM CLI) version 1.21.1, but this architecture should work with later AWS SAM CLI versions.
Architecture

Target technology stack

- Amazon API Gateway
- AWS SAM
- Amazon Cognito
- Amazon DynamoDB
- AWS Lambda
- Amazon Simple Queue Service (Amazon SQS) queue

Target architecture

The following diagram shows the architecture of sample AWS serverless shopping cart application.

In this solution setup, AWS SAM CLI serves as the interface for AWS CloudFormation stacks. AWS SAM templates automatically deploy nested applications. The parent SAM template calls the child templates, and the parent CloudFormation stack deploys the child stacks. Each child stack builds the AWS resources that are defined in the AWS SAM CloudFormation templates.
• **Amazon API Gateway** – Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.

• **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run on AWS. You create a template that describes all the AWS resources that you want, and CloudFormation provisions and configures those resources for you.

• **Amazon Cognito** – Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps.

• **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic.

• **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

• **AWS SAM** – AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS. A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that a serverless application is more than a Lambda function—it can include additional resources such as APIs, databases, and event source mappings.

• **Amazon SQS** – Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue. You can use Amazon SQS to integrate and decouple distributed software systems and components.
## Epics

### Install AWS SAM CLI

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install AWS SAM CLI.</td>
<td>To install AWS SAM CLI, see the instructions in the AWS SAM documentation.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Set up AWS credentials.</td>
<td>To set AWS credentials so that the AWS SAM CLI can make calls to AWS services on your behalf, run the <code>aws configure</code> command and follow the prompts:</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

```bash
#aws configure
AWS Access Key ID [None]: <your_access_key_id>
AWS Secret Access Key [None]: your_secret_access_key
Default region name [None]:
Default output format [None]:
```

### Initialize the AWS SAM project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy templates to initialize the project.</td>
<td>To initialize the project, run the <code>SAM init</code> command. When prompted to choose a template source, choose Custom Template Location.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Create, compile, and build the SAM template code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the AWS SAM application templates.</td>
<td>Create the templates for the nested applications. In this example, the nested application templates are <code>auth.yaml</code>,</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>product-mock.yaml, and shoppingcart-service.yaml. For more information, see the Additional information section.</td>
<td></td>
</tr>
<tr>
<td>Create the parent template.</td>
<td>Create the template that will invoke the nested application templates. In this example, the parent template is template.yml.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Compile and build the AWS SAM template code.</td>
<td>Using the AWS SAM CLI, run the following command.</td>
<td>DevOps</td>
</tr>
<tr>
<td></td>
<td>sam build</td>
<td></td>
</tr>
</tbody>
</table>

### Deploy the AWS SAM template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the applications.</td>
<td>To launch the SAM template code that creates the nested application CloudFormation stacks and deploys code in the AWS environment, run the following command.</td>
<td>DevOps</td>
</tr>
<tr>
<td></td>
<td>sam deploy --guided --stack-name shopping-cart-nested-stack --capabilities CAPABILITY_IAM CAPABILITY_AUTO_EXPAND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see the Additional information section.</td>
<td></td>
</tr>
</tbody>
</table>

### Verify the deployment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the stacks.</td>
<td>To review and verify the AWS CloudFormation stacks and all AWS resources that were defined in the AWS SAM templates, log in to the AWS Management Console. For more information, see the Additional information section.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
Related resources

References

- AWS Serverless Application Model (AWS SAM)
- AWS SAM on GitHub
- Serverless Shopping Cart Microservice (AWS sample application) https://github.com/aws/serverless-application-model

Tutorials and videos

- https://youtu.be/1NU7vyJw9LU Build a Serverless App https://youtu.be/1NU7vyJw9LU
- AWS Online Tech Talks: Serverless Application Building and Deployments with AWS SAM

Additional information

Directory structure of the applications

After all the code is in place, the example has the following directory structure:

- `layers` – A layer is a file archive that contains libraries, a custom runtime, or other dependencies. With layers, you can use libraries in your function without needing to include them in a deployment package.
- `product-mock-service` – This folder contains all product-related Lambda functions and files.
- `shopping-cart-service` – This folder contains all shopping-related Lambda functions and files.
**Nested application templates**

For the independent applications, the example includes the following three .yaml files.

*auth.yaml*

This template sets up authentication-related resources, such as Amazon Cognito and AWS Systems Manager Parameter Store.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  auth-resources

SAM Template for auth resources

Globals:
  Function:
    Timeout: 3

Resources:
  CognitoUserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: !Sub ${AWS::StackName}-UserPool
      AutoVerifiedAttributes:
        - email
  UserPoolClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      ClientName: my-app
      GenerateSecret: false
      UserPoolId: !Ref CognitoUserPool
      ExplicitAuthFlows:
        - ADMIN_NO_SRP_AUTH

  UserPoolSSM:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Name: /serverless-shopping-cart-demo/auth/user-pool-id
      Value: !Ref CognitoUserPool

  UserPoolARNSSM:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Name: /serverless-shopping-cart-demo/auth/user-pool-arn
      Value: !GetAtt CognitoUserPool.Arn

  UserPoolAppClientSSM:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Name: /serverless-shopping-cart-demo/auth/user-pool-client-id
      Value: !Ref UserPoolClient

Outputs:
  CognitoUserPoolId:
    Description: "Cognito User Pool ID"
    Value: !Ref CognitoUserPool
```
CognitoAppClientId:
  Description: "Cognito App Client ID"
  Value: !Ref UserPoolClient

UserPoolARNSSM:
  Description: "UserPool ID"
  Value: !Ref UserPoolARNSSM

**product-mock.yaml**

This template deploys product-related resources, such as Lambda functions and Amazon API Gateway.

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  product-service

SAM Template for mock product-service

Parameters:
  AllowedOrigin:
    Type: 'String'

Globals:
  Function:
    Timeout: 5
    Tracing: Active
    AutoPublishAlias: live
    Runtime: python3.8
    MemorySize: 256
    Environment:
      Variables:
        LOG_LEVEL: "DEBUG"
        ALLOWED_ORIGIN: !Ref AllowedOrigin
        POWERTOOLS_SERVICE_NAME: product-mock
        POWERTOOLS_METRICS_NAMESPACE: ecommerce-app

Api:
  EndpointConfiguration: REGIONAL
  TracingEnabled: true
  OpenApiVersion: '2.0'
  Cors:
    AllowMethods: "'OPTIONS,POST,GET'"
    AllowHeaders: "'Content-Type'"
    AllowOrigin: !Sub "'${AllowedOrigin}'"

Resources:
  GetProductFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: product-mock-service/
      Handler: get_product.lambda_handler
      Events:
        ListCart:
          Type: Api
          Properties:
            Path: /product/{product_id}
            Method: get

  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: product-mock-service/
```

Handler: get_products.lambda_handler
Events:
  ListCart:
    Type: Api
    Properties:
      Path: /product
      Method: get

GetProductApiUrl:
  Type: AWS::SSM::Parameter
  Properties:
    Type: String
    Name: /serverless-shopping-cart-demo/products/products-api-url
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/
Prod"

Outputs:
  ProductApi:
    Description: "API Gateway endpoint URL for Prod stage for Product Mock Service"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/
Prod"

shoppingcart-service.yaml

This template sets up shopping cart–related resources, such as AWS Identity and Access Management (IAM), DynamoDB tables, and Lambda functions.

AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  shoppingcart-service

SAM Template for shoppingcart-service

Parameters:
  UserPoolArn:
    Type: 'AWS::SSM::Parameter::Value<String>'
    Default: '/serverless-shopping-cart-demo/auth/user-pool-arn'
  UserPoolId:
    Type: 'AWS::SSM::Parameter::Value<String>'
    Default: '/serverless-shopping-cart-demo/auth/user-pool-id'
  ProductServiceUrl:
    Type: 'AWS::SSM::Parameter::Value<String>'
    Default: '/serverless-shopping-cart-demo/products/products-api-url'
  AllowedOrigin:
    Type: 'String'

Globals:
  Function:
    Timeout: 5
    MemorySize: 512
    Tracing: Active
    AutoPublishAlias: live
    Runtime: python3.8
    Environment:
      Variables:
        TABLE_NAME: !Ref DynamoDBShoppingCartTable
        LOG_LEVEL: "INFO"
        ALLOWED_ORIGIN: !Ref AllowedOrigin
        POWERTOOLS_SERVICE_NAME: shopping-cart
        POWERTOOLS_METRICS_NAMESPACE: ecommerce-app

1425
EndpointConfiguration: REGIONAL
TracingEnabled: true
OpenApiVersion: '2.0'
Cors:
  AllowMethods: "'OPTIONS,POST,GET,PUT'"
  AllowHeaders: "'Content-Type,Authorization'"
  AllowCredentials: true
  AllowOrigin: !Sub "'${AllowedOrigin}'"

Resources:
  UtilsLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: ./layers/
      CompatibleRuntimes:
        - python3.8
    Metadata:
      BuildMethod: python3.8

  CartApi:
    Type: AWS::Serverless::Api
    DependsOn:
      - ApiGWAccount
    Properties:
      StageName: Prod
      MethodSettings:
        - DataTraceEnabled: True
        - MetricsEnabled: True
        - ResourcePath: "/*
        - HttpMethod: "*"
        - LoggingLevel: INFO
      Auth:
        Authorizers:
          CognitoAuthorizer:
            UserPoolArn: !Ref UserPoolArn
            Identity: # OPTIONAL
            Header: Authorization # OPTIONAL; Default: 'Authorization'

  ListCartRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Action: "sts:AssumeRole"
          - Effect: "Allow"
          - Principal:
              Service:
                - "lambda.amazonaws.com"

  AddToCartRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Action:
          - Effect: "Allow"
            Principal:
            Service:
              - "lambda.amazonaws.com"
LambdaLoggingPolicy:
  Type: "AWS::IAM::Policy"
  Properties:
    PolicyName: LambdaXRayPolicy
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Action: ["xray:PutTraceSegments",
                    "xray:PutTelemetryRecords",
                    "logs:CreateLogGroup",
                    "logs:CreateLogStream",
                    "logs:PutLogEvents"
                 ]
          Resource: "*"
    Roles:
      - !Ref ListCartRole
      - !Ref AddToCartRole

DynamoDBReadPolicy:
  Type: "AWS::IAM::Policy"
  Properties:
    PolicyName: DynamoDBReadPolicy
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Action: ["dynamodb:GetItem",
                    "dynamodb:Scan",
                    "dynamodb:Query",
                    "dynamodb:BatchGetItem",
                    "dynamodb:DescribeTable"
                 ]
          Resource: !GetAtt DynamoDBShoppingCartTable.Arn
    Roles:
      - !Ref ListCartRole
      - !Ref AddToCartRole

DynamoDBWritePolicy:
  Type: "AWS::IAM::Policy"
  Properties:
    PolicyName: DynamoDBWritePolicy
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Action: ["dynamodb:PutItem",
                    "dynamodb:UpdateItem",
                    "dynamodb:ConditionCheckItem",
                    "dynamodb:DeleteItem",
                    "dynamodb:BatchWriteItem"
                 ]
          Resource: !GetAtt DynamoDBShoppingCartTable.Arn
    Roles:
- !Ref AddToCartRole

SQSSendMessagePolicy:
  Type: "AWS::IAM::Policy"
  Properties:
    PolicyName: SQSSendMessagePolicy
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Action: [
            "sqs:SendMessage"
          ]
          Resource: !GetAtt CartDeleteSQSQueue.Arn
          Roles: - !Ref AddToCartRole

ListCartFunction:
  Type: AWS::Serverless::Function
  DependsOn: - LambdaLoggingPolicy
  Properties:
    CodeUri: shopping-cart-service/
    Handler: list_cart.lambda_handler
    Role: !GetAtt ListCartRole.Arn
    Layers: - !Ref UtilsLayer
    Environment:
      Variables:
        USERPOOL_ID: !Ref UserPoolId
    Events:
      ListCart:
        Type: Api
        Properties:
          RestApiId: !Ref CartApi
          Path: /cart
          Method: get

AddToCartFunction:
  Type: AWS::Serverless::Function
  DependsOn: - LambdaLoggingPolicy
  Properties:
    CodeUri: shopping-cart-service/
    Handler: add_to_cart.lambda_handler
    Role: !GetAtt AddToCartRole.Arn
    Layers: - !Ref UtilsLayer
    Environment:
      Variables:
        PRODUCT_SERVICE_URL: !Ref ProductServiceUrl
        USERPOOL_ID: !Ref UserPoolId
    Events:
      AddToCart:
        Type: Api
        Properties:
          RestApiId: !Ref CartApi
          Path: /cart
          Method: post

UpdateCartFunction:
Type: AWS::Serverless::Function
DependsOn:
  - LambdaLoggingPolicy
Properties:
  CodeUri: shopping-cart-service/
  Handler: update_cart.lambda_handler
  Role: !GetAtt AddToCartRole.Arn
  Layers:
    - !Ref UtilsLayer
Environment:
  Variables:
    PRODUCT_SERVICE_URL: !Ref ProductServiceUrl
    USERPOOL_ID: !Ref UserPoolId
Events:
  AddToCart:
    Type: Api
    Properties:
      RestApiId: !Ref CartApi
      Path: /cart/{product_id}
      Method: put

MigrateCartFunction:
Type: AWS::Serverless::Function
DependsOn:
  - LambdaLoggingPolicy
Properties:
  CodeUri: shopping-cart-service/
  Handler: migrate_cart.lambda_handler
  Timeout: 30
  Layers:
    - !Ref UtilsLayer
Environment:
  Variables:
    PRODUCT_SERVICE_URL: !Ref ProductServiceUrl
    USERPOOL_ID: !Ref UserPoolId
    DELETE_FROM_CART_SQS_QUEUE: !Ref CartDeleteSQSQueue
  Role: !GetAtt AddToCartRole.Arn
Events:
  AddToCart:
    Type: Api
    Properties:
      RestApiId: !Ref CartApi
      Path: /cart/migrate
      Method: post
      Auth:
        Authorizer: CognitoAuthorizer

CheckoutCartFunction:
Type: AWS::Serverless::Function
DependsOn:
  - LambdaLoggingPolicy
Properties:
  CodeUri: shopping-cart-service/
  Handler: checkout_cart.lambda_handler
  Timeout: 10
  Layers:
    - !Ref UtilsLayer
Environment:
  Variables:
    PRODUCT_SERVICE_URL: !Ref ProductServiceUrl
    USERPOOL_ID: !Ref UserPoolId
  Role: !GetAtt AddToCartRole.Arn
Events:
  AddToCart:
Type: Api
Properties:
  RestApiId: !Ref CartApi
  Path: /cart/checkout
  Method: post
  Auth:
    Authorizer: CognitoAuthorizer

GetCartTotalFunction:
  Type: AWS::Serverless::Function
  DependsOn:
    - LambdaLoggingPolicy
  Properties:
    CodeUri: shopping-cart-service/
    Handler: get_cart_total.lambda_handler
    Timeout: 10
    Layers:
      - !Ref UtilsLayer
    Role: !GetAtt ListCartRole.Arn
  Events:
    GetCartTotal:
      Type: Api
      Properties:
        RestApiId: !Ref CartApi
        Path: /cart/{product_id}/total
        Method: get

DeleteFromCartFunction:
  Type: AWS::Serverless::Function
  DependsOn:
    - LambdaLoggingPolicy
  Properties:
    CodeUri: shopping-cart-service/
    Handler: delete_from_cart.lambda_handler
    ReservedConcurrentExecutions: 25  # Keep the ddb spikes down in case of many deletes at once
    Policies:
      - SQSPollerPolicy:
          QueueName: !GetAtt CartDeleteSQSQueue.QueueName
        Statement:
          - Effect: Allow
            Action:
              - "dynamodb:DeleteItem"
              - "dynamodb:BatchWriteItem"
            Resource:
              - !GetAtt DynamoDBShoppingCartTable.Arn
    Layers:
      - !Ref UtilsLayer
  Environment:
    Variables:
      USERPOOL_ID: !Ref UserPoolId
  Events:
    RetrieveFromSQS:
      Type: SQS
      Properties:
        Queue: !GetAtt CartDeleteSQSQueue.Arn
        BatchSize: 5

CartDBStreamHandler:
  Type: AWS::Serverless::Function
  DependsOn:
    - LambdaLoggingPolicy
Properties:
- CodeUri: shopping-cart-service/
- Handler: db_stream_handler.lambda_handler
- Layers:
  - !Ref UtilsLayer
- Policies:
  - AWSLambdaDynamoDBExecutionRole
  - Statement:
    - Effect: Allow
      Action:
        - "dynamodb:UpdateItem"
      Resource:
        - !GetAtt DynamoDBShoppingCartTable.Arn
- Events:
  - Stream:
    - Type: DynamoDB
    - Properties:
      - Stream: !GetAtt DynamoDBShoppingCartTable.StreamArn
      - BatchSize: 100
      - MaximumBatchingWindowInSeconds: 60
      - StartingPosition: LATEST

DynamoDBShoppingCartTable:
- Type: AWS::DynamoDB::Table
- Properties:
  - AttributeDefinitions:
    - AttributeName: pk
      AttributeType: S
    - AttributeName: sk
      AttributeType: S
  - KeySchema:
    - AttributeName: pk
      KeyType: HASH
    - AttributeName: sk
      KeyType: RANGE
  - BillingMode: PAY_PER_REQUEST
  - StreamSpecification:
    - StreamViewType: 'NEW_AND_OLD_IMAGES'
  - TimeToLiveSpecification:
    - AttributeName: expirationTime
      Enabled: True

APIGWCloudWatchRole:
- Type: 'AWS::IAM::Role'
- Properties:
  - AssumeRolePolicyDocument:
    - Version: 2012-10-17
    - Statement:
      - Effect: Allow
      - Principal:
        - Service:
          - apigateway.amazonaws.com
      - Action: 'sts:AssumeRole'
      - Path: /
      - ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs

ApiGWAccount:
- Type: 'AWS::ApiGateway::Account'
- Properties:
  - CloudWatchRoleArn: !GetAtt APIGWCloudWatchRole.Arn

CartDeleteSQSQueue:
Additional information

Type: AWS::SQS::Queue
Properties:
  VisibilityTimeout: 20
  RedrivePolicy:
    deadLetterTargetArn: !GetAtt CartDeleteSQSDLQ.Arn
    maxReceiveCount: 5
CartDeleteSQSDLQ:
  Type: AWS::SQS::Queue

CartApiUrl:
  Type: AWS::SSM::Parameter
Properties:
  Type: String
  Name: /serverless-shopping-cart-demo/shopping-cart/cart-api-url
  Value: !Sub "https://${CartApi}.execute-api.${AWS::Region}.amazonaws.com/Prod"

Outputs:
  CartApi:
    Description: "API Gateway endpoint URL for Prod stage for Cart Service"
    Value: !Sub "https://${CartApi}.execute-api.${AWS::Region}.amazonaws.com/Prod"

The parent template

The `template.yaml` file calls all the independent child stacks. All separate applications would work as nested application in single parent SAM template.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  SAM Template for Nested application resources
Resources:
  Auth:
    Type: AWS::Serverless::Application
    Properties:
      Location: auth.yaml
  Product:
    Type: AWS::Serverless::Application
    Properties:
      Location: product-mock.yaml
      Parameters:
        AllowedOrigin: 'http://localhost:8080'
    DependsOn: Auth
  Shopping:
    Type: AWS::Serverless::Application
    Properties:
      Location: shoppingcart-service.yaml
      Parameters:
        AllowedOrigin: 'http://localhost:8080'
    DependsOn: Product
```

Running the sam deploy command

When you run the command `sam deploy --guided --stack-name shopping-cart-nested-stack --capabilities CAPABILITY_IAM CAPABILITY_AUTO_EXPAND`, it will prompt with a few questions. Answer all questions with `y`, and then you will see the following screen outputs.
Stacks on the console

Log in to the console and choose to the CloudFormation service to see the parent and child stacks. In this example, sam-shopping-cart is the parent stack that calls the nested Auth, Product and Shopping stacks.
The product stack gives the Product API Gateway URL link as an output.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductApi</td>
<td><a href="https://ame02polyic.execute-api.us-east-1.amazonaws.com/Prod">https://ame02polyic.execute-api.us-east-1.amazonaws.com/Prod</a></td>
<td>API Gateway endpoint URL for Prod stage for Product Mock Service</td>
</tr>
</tbody>
</table>

The following figure shows the output of the Product API Gateway /GET method.

Attachments

attachment.zip

Implement SaaS tenant isolation for Amazon S3 by using an AWS Lambda token vending machine

Created by Tabby Ward (AWS), sravan periyathambi (AWS), and Thomas Davis (AWS)
Summary

Multitenant SaaS applications must implement systems to ensure that tenant isolation is maintained. When you store tenant data on the same Amazon Web Services (AWS) resource—such as multiple tenants storing data in the same Amazon Simple Storage Service (Amazon S3) bucket—you must ensure that cross-tenant access cannot occur. Token vending machines (TVMs) are one way to guarantee tenant data isolation. These machines provide a mechanism for obtaining tokens while abstracting the complexity of how these tokens are generated. Developers can use a TVM without having detailed knowledge of how it produces tokens.

This pattern implements a TVM by using AWS Lambda. The TVM generates a token that consists of temporary security token service (STS) credentials that limit access to a single SaaS tenant’s data in an S3 bucket.

TVMs, and the code that’s provided with this pattern, are typically used with claims that are derived from JSON Web Tokens (JWTs) to associate requests for AWS resources with a tenant-scoped AWS Identity and Access Management (IAM) policy. You can use the code in this pattern as a basis to implement a SaaS application that generates scoped, temporary STS credentials based on the claims provided in a JWT token.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 1.19.0 or later, installed and configured on macOS, Linux, or Windows. Alternatively, you can use AWS CLI version 2.1 or later.

Limitations

- This code runs in Java and doesn’t currently support other programming languages.
- The sample application does not include AWS cross-Region or disaster recovery (DR) support.
- This pattern demonstrates how a Lambda TVM for a SaaS application can provide scoped tenant access. It is not intended to be used in production environments.

Architecture

Target technology stack

- AWS Lambda
- Amazon S3
- IAM
- AWS Security Token Service (AWS STS)

Target architecture
Tools

AWS services

- **AWS CLI** – AWS Command Line Interface (AWS CLI) is a unified tool for interacting with AWS services by using commands in your command-line shell.
- **AWS Lambda** – AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance.
- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services.
- **AWS STS** - AWS Security Token Service (AWS STS) enables you to request temporary, limited-privilege credentials for IAM users or for users whom you authenticate (federated users).

Code

The source code for this pattern is available as an attachment and includes the following files:

- **s3UploadSample.jar** provides the source code for a Lambda function that uploads a JSON document to an S3 bucket.
- **tvmlayer.zip** provides a reusable Java library that supplies a token (STS temporary credentials) for the Lambda function to access the S3 bucket and upload the JSON document.
- **token-vending-machine-sample-app.zip** provides the source code used to create these artifacts and compilation instructions.

To use these files, follow the instructions in the next section.
## Epics

### Determine variable values

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine variable values.</td>
<td>The implementation of this pattern requires includes several variable names that must be used consistently. Determine the values that should be used for each variable, and provide that value when requested in subsequent steps.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

**<AWS Account ID>** — The 12-digit account ID that is associated with the AWS account you are implementing this pattern in. For information about how to find your AWS account ID, see Your AWS account ID and its alias in the IAM documentation.

**<AWS Region>** — The AWS Region that you are implementing this pattern in. For more information on AWS Regions, see Regions and Availability Zones on the AWS website.

**<sample-tenant-name>** — The name of a tenant to use in the application. We recommend that you use only alphanumeric characters in this value for simplicity, but you can use any valid name for an S3 object key.

**<sample-tvm-role-name>** — The name of the IAM role attached to the Lambda function that runs the TVM and sample application. The role name is a string that consists of uppercase and lowercase alphanumeric characters with no spaces. You can also include any of the following characters: underscore (_), plus sign (+), equal sign (=), comma (,), period (.), at sign (@), and hyphen (-). The role...
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name must be unique within the account</strong></td>
<td></td>
</tr>
<tr>
<td><strong>&lt;sample-app-role-name&gt;</strong></td>
<td>The name of the IAM role that is assumed by the Lambda function when it generates scoped, temporary STS credentials. The role name is a string that consists of uppercase and lowercase alphanumeric characters with no spaces. You can also include any of the following characters: underscore (_), plus sign (+), equal sign (=), comma (,), period (.), at sign (@), and hyphen (-). The role name must be unique within the account.</td>
</tr>
<tr>
<td><strong>&lt;sample-app-function-name&gt;</strong></td>
<td>The name of the Lambda function. This is a string that's up to 64 characters in length.</td>
</tr>
</tbody>
</table>
| **<sample-app-bucket-name>**              | The name of an S3 bucket that must be accessed with permissions that are scoped to a specific tenant. S3 bucket names:  
  - Must be between 3 and 63 characters long.  
  - Must consist of only lowercase letters, numbers, periods (.), and hyphens (-).  
  - Must begin and end with a letter or number.  
  - Must not be formatted as an IP address (for example, 192.168.5.4).  
  - Must be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: aws (Standard Regions), aws-cn (China Regions), and aws-us-gov (AWS GovCloud [US] Regions). |
### Create an S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an S3 bucket for the sample application. | Use the following AWS CLI command to create an S3 bucket. Provide the `<sample-app-bucket-name>` value in the code snippet:  

```bash
aws s3api create-bucket --bucket <sample-app-bucket-name>
```

The Lambda sample application uploads JSON files to this bucket. | Cloud administrator |

### Create the IAM TVM role and policy

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a TVM role. | Use one of the following AWS CLI commands to create an IAM role. Provide the `<sample-tvm-role-name>` value in the command.  

For macOS or Linux shells:  

```bash
aws iam create-role --role-name <sample-tvm-role-name> --assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]}
```

For the Windows command line:  

```bash
aws iam create-role --role-name <sample-tvm-role-name> --assume-role-policy-document "{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]}
```

The Lambda sample application assumes this role when the | Cloud administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>application is invoked. The capability to assume the application role with a scoped policy gives the code broader permissions to access the S3 bucket.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create an inline TVM role policy.</td>
<td>Use one of the following AWS CLI commands to create an IAM policy. Provide the <code>&lt;sample-tvm-role-name&gt;</code>, <code>&lt;AWS Account ID&gt;</code>, and <code>&lt;sample-app-role-name&gt;</code> values in the command.</td>
<td></td>
</tr>
</tbody>
</table>

For macOS or Linux shells:
```
aws iam put-role-policy
--role-name <sample-tvm-role-name> --policy-name assume-app-role
--policy-document '{"Version": "2012-10-17", "Statement": [
{"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::<AWS Account ID>:role/<sample-app-role-name>"}],"}
}'
```

For the Windows command line:
```
aws iam put-role-policy
--role-name <sample-tvm-role-name> --policy-name assume-app-role
''
```

This policy is attached to the TVM role. It gives the code the capability to assume the application role, which has broader permissions to access the S3 bucket. |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach the managed Lambda policy.</td>
<td>Use the following AWS CLI command to attach the AWSLambdaBasicExecutionRole IAM policy. Provide the <code>&lt;sample-tvm-role-name&gt;</code> value in the command:</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td></td>
<td><strong>aws iam attach-role-policy</strong>&lt;br&gt;<strong>--role-name</strong> <code>&lt;sample-tvm-role-name&gt;</code>&lt;br&gt;<strong>--policy-arn</strong> <code>arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This managed policy is attached to the TVM role to permit Lambda to send logs to Amazon CloudWatch.</td>
<td></td>
</tr>
</tbody>
</table>

Create the IAM application role and policy

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the application role.</td>
<td>Use one of the following AWS CLI commands to create an IAM role. Provide the <code>&lt;sample-app-role-name&gt;</code>, <code>&lt;AWS Account ID&gt;</code>, and <code>&lt;sample-tvm-role-name&gt;</code> values in the command.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td></td>
<td>For macOS or Linux shells:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>aws iam create-role</strong>&lt;br&gt;<strong>--role-name</strong> <code>&lt;sample-app-role-name&gt;</code>&lt;br&gt;<strong>--assume-role-policy-document</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For the Windows command line:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>aws iam create-role</strong>&lt;br&gt;<strong>--role-name</strong> <code>&lt;sample-app-role-name&gt;</code>&lt;br&gt;<strong>--assume-role-policy-document</strong></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
|     | `{tvms-role-name}>",
"Action
"
: 
"sts:AssumeRole")
" | The Lambda sample application assumes this role with a scoped policy to get tenant-based access to an S3 bucket. |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an inline application role policy.</td>
<td>Use one of the following AWS CLI commands to create an IAM policy. Provide the <code>&lt;sample-app-role-name&gt;</code> and <code>&lt;sample-app-bucket-name&gt;</code> values in the command.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

For macOS or Linux shells:

```bash
aws iam put-role-policy
--role-name <sample-app-role-name>
--policy-name s3-bucket-access
```

For the Windows command line:

```bash
aws iam put-role-policy
--role-name <sample-app-role-name>
--policy-name s3-bucket-access
```

This policy is attached to the application role. It provides broad access to objects in the S3 bucket. When the sample application assumes the role, these permissions are scoped to a specific tenant with the TVM's dynamically generated policy.
## Create the Lambda sample application with TVM

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the compiled source files.</td>
<td>Download the \texttt{s3UploadSample.jar} and \texttt{tvm-layer.zip} files, which are included as attachments. The source code used to create these artifacts and compilation instructions are provided in \texttt{token-vending-machine-sample-app.zip}.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Create the Lambda layer.</td>
<td>Use the following AWS CLI command to create a Lambda layer, which makes the TVM accessible to Lambda.</td>
<td>Cloud administrator, App developer</td>
</tr>
<tr>
<td></td>
<td>\texttt{Note:} If you aren't running this command from the location where you downloaded \texttt{tvm-layer.zip}, provide the correct path to \texttt{tvm-layer.zip} in the \texttt{--zip-file} parameter.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\texttt{aws lambda publish-layer-version --layer-name sample-token-vending-machine --compatible-runtimes java11 --zip-file fileb://tvm-layer.zip}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This command creates a Lambda layer that contains the reusable TVM library.</td>
<td></td>
</tr>
<tr>
<td>Create the Lambda function.</td>
<td>Use the following AWS CLI command to create a Lambda function. Provide the \texttt{&lt;sample-app-function-name&gt;}, \texttt{&lt;AWS Account ID&gt;}, \texttt{&lt;AWS Region&gt;}, \texttt{&lt;sample-tvm-role-name&gt;}, \texttt{&lt;sample-app-bucket-name&gt;}, and \texttt{&lt;sample-app-role-name&gt;} values in the command.</td>
<td>Cloud administrator, App developer</td>
</tr>
<tr>
<td></td>
<td>\texttt{Note:} If you aren't running this command from the location where you downloaded \texttt{s3UploadSample.jar}, provide the correct path to \texttt{s3UploadSample.jar} in the \texttt{--zip-file} parameter.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\texttt{aws lambda create-function --function-name &lt;sample-}</td>
<td></td>
</tr>
</tbody>
</table>
Invoke the Lambda sample application.

Use one of the following AWS CLI commands to start the Lambda sample application with its expected payload. Provide the `<sample-app-function-name>` and `<sample-tenant-name>` values in the command.

For macOS and Linux shells:

```bash
aws lambda invoke --function <sample-app-function-name> --invocation-type RequestResponse --payload '"
  "tenant": 
  "<sample-tenant-name>"
"' --cli-binary-format raw-in-base64-out response.json
```

For the Windows command line:

This command creates a Lambda function with the sample application code and the TVM layer attached. It also sets two environment variables: S3_BUCKET and ROLE. The sample application uses these variables to determine the role to assume and the S3 bucket to upload JSON documents to.

Test the sample application and TVM

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoke the Lambda sample application.</td>
<td>Use one of the following AWS CLI commands to start the Lambda sample application with its expected payload. Provide the <code>&lt;sample-app-function-name&gt;</code> and <code>&lt;sample-tenant-name&gt;</code> values in the command.</td>
<td>Cloud administrator, App developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>aws lambda invoke</td>
<td>This command calls the Lambda function and returns the result in a <code>response.json</code> document. On many Unix-based systems, you can change <code>response.json</code> to <code>/dev/stdout</code> to output the results directly to your shell without creating another file. Note: Changing the <code>&lt;sample-tenant-name&gt;</code> value in subsequent invocations of this Lambda function alters the location of the JSON document and the permissions the token provides.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>View the S3 bucket to see created objects</td>
<td>Browse to the S3 bucket (<code>&lt;sample-app-bucket-name&gt;</code>) that you created earlier. This bucket contains an S3 object prefix with the value of <code>&lt;sample-tenant-name&gt;</code>. Under that prefix, you will find a JSON document named with a UUID. Invoking the sample application multiple times adds more JSON documents.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>View Cloudwatch logs for the sample</td>
<td>View the Cloudwatch logs associated with the Lambda function named <code>&lt;sample-app-function-name&gt;</code>. For instructions, see Accessing Amazon CloudWatch logs for AWS Lambda in the AWS Lambda documentation. You can view the tenant-scoped policy generated by the TVM in these logs. This tenant-scoped policy gives permissions for the sample application to the Amazon S3 PutObject, GetObject, DeleteObject, and ListBucket APIs, but only for the object prefix associated with <code>&lt;sample-tenant-name&gt;</code>. In subsequent invocations of the sample application, if you change the <code>&lt;sample-tenant-name&gt;</code>, the TVM updates the scoped policy to correspond to the tenant provided in the invocation payload. This dynamically generated policy shows how tenant-scoped access can be maintained with a TVM in SaaS applications. The TVM functionality is provided in a Lambda layer so that it can be attached to other Lambda functions used by an application without having to replicate the code. For an illustration of the dynamically generated policy, see the Additional information (p. 1448) section.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

**Related resources**

- Isolating Tenants with Dynamically Generated IAM Policies (blog post)
- Applying Dynamically Generated Isolation Policies in SaaS Environment (blog post)
- AWS SaaS Boost (an open-source reference environment that helps you move your SaaS offering to AWS)
Additional information

The following Amazon Cloudwatch log shows the dynamically generated policy produced by the TVM code in this pattern. In this screenshot, the `<sample-app-bucket-name>` is DOC-EXAMPLE-BUCKET and the `<sample-tenant-name>` is test-tenant-1. The STS credentials returned by this scoped policy are unable to perform any actions on objects in the S3 bucket except for objects that are associated with the object key prefix test-tenant-1.

```
[2021-07-29 16:57:07.376] f5eb7024-ec14-4d2c-9c3c-d69687d61f24 INFO c.a.a.t.TokenVendingMachine -
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        ],
        "Condition": {
            "StringLike": {
                "s3:prefix": [
                    "test-tenant-1",
                    "test-tenant-1/",
                    "test-tenant-1/*"
                ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/test-tenant-1/*"
        ]
    }
    ]
}
```

Attachments

attachment.zip

Implement the serverless saga pattern by using AWS Step Functions

*Created by Tabby Ward (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Modernization; Serverless; Cloud-native</th>
<th>Workload:</th>
<th>Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS CloudFormation; Amazon DynamoDB; Amazon API</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1448
Summary

In a microservices architecture, the main goal is to build decoupled and independent components to promote agility, flexibility, and faster time to market for your applications. As a result of decoupling, each microservice component has its own data persistence layer. In a distributed architecture, business transactions can span multiple microservices. Because these microservices cannot use a single atomicity, consistency, isolation, durability (ACID) transaction, you might end up with partial transactions. In this case, some control logic is needed to undo the transactions that have already been processed. The distributed saga pattern is typically used for this purpose.

The saga pattern is a failure management pattern that helps establish consistency in distributed applications and coordinates transactions between multiple microservices to maintain data consistency. When you use the saga pattern, every service that performs a transaction publishes an event that triggers subsequent services to perform the next transaction in the chain. This continues until the last transaction in the chain is complete. If a business transaction fails, saga orchestrates a series of compensating transactions that undo the changes that were made by the preceding transactions.

The following sections demonstrate how to automate the setup and deployment of a sample application (which handles travel reservations) by using the AWS Cloud Development Kit (AWS CDK) along with serverless technologies such as AWS Step Functions, AWS Lambda, and Amazon DynamoDB. The sample application also uses Amazon API Gateway and Amazon Simple Notification Service (Amazon SNS) to implement a saga execution coordinator.

For more information about the saga pattern and other data persistence patterns, see the guide Enabling data persistence in microservices on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- An active AWS account with programmatic access
- AWS CDK Toolkit stack, which is used to provision resources from this pattern, deployed to your target AWS Region
- NodeJS, which is required for the AWS CDK
- AWS CDK CLI with AWS account configuration, so that you can create AWS resources by using the cdk deploy command
- A code editor of your choice (such as Visual Studio Code, Sublime, or Atom)

Product versions

- NodeJS version 14
- AWS CDK version 1.89.0

Limitations

Event sourcing is a natural way to implement the saga orchestration pattern in a microservices architecture where all components are loosely coupled and don’t have direct knowledge of one another. If your transaction involves a small number of steps (three to five), the saga pattern might be a great fit. However, complexity increases with the number of microservices and the number of steps.
Testing and debugging can become difficult when you're using this design, because you have to have all services running in order to simulate the transaction pattern.

**Architecture**

**Target architecture**

The proposed architecture uses AWS Step Functions to build a saga pattern to book flights, book car rentals, and process payments for a vacation.

The following workflow diagram illustrates the typical flow of the travel reservation system. The workflow consists of reserving air travel ("ReserveFlight"), reserving a car ("ReserveCarRental"), processing payments ("ProcessPayment"), confirming flight reservations ("ConfirmFlight"), and confirming car rentals ("ConfirmCarRental") followed by a success notification when these steps are complete. However, if the system encounters any errors in running any of these transactions, it starts to fail backward. For example, an error with payment processing ("ProcessPayment") triggers a refund ("RefundPayment"), which then triggers a cancellation of the rental car and flight ("CancelRentalReservation" and "CancelFlightReservation"), which ends the entire transaction with a failure message.

This pattern deploys separate Lambda functions for each task that is highlighted in the diagram as well as three DynamoDB tables for flights, car rentals, and payments. Each Lambda function creates, updates, or deletes the rows in the respective DynamoDB tables, depending on whether a transaction is confirmed or rolled back. The pattern uses Amazon SNS to send text (SMS) messages to subscribers, notifying them of failed or successful transactions.
Automation and scale

You can create the configuration for this architecture by using an AWS CloudFormation template or AWS CDK scripts. See the attachment for the AWS CDK scripts, which are written in TypeScript.

Tools

AWS services

- **AWS Step Functions** – AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through the Step Functions graphical console, you see your application’s workflow as a series of event-driven steps.
- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use DynamoDB to create a database table that can store and retrieve any amount of data, and serve any level of request traffic.
- **AWS Lambda** – AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon API Gateway** – Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.
• **AWS CDK** – The AWS Cloud Development Kit (AWS CDK) is a software development framework for defining your cloud application resources by using familiar programming languages such as TypeScript, JavaScript, Python, Java, and C#/.Net.

**Code**

The code for a sample application that demonstrates the saga pattern, including the AWS CDK scripts, the Lambda functions, and the DynamoDB tables, is attached as an archive. Follow the instructions in the first epic to install these.

**Epics**

**Install packages, compile, and build**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the NPM packages.</td>
<td>After you download and extract the contents of the attached archive, run the following command in the /cdk-serverless-saga folder to download and install all Node Package Manager (NPM) packages: npm install</td>
<td>Developer, Cloud architect</td>
</tr>
<tr>
<td>Compile the AWS CDK scripts.</td>
<td>In the /cdk-serverless-saga folder, run the following command to instruct the TypeScript transpiler to create all necessary JavaScript files: npm run build</td>
<td>Developer, Cloud architect</td>
</tr>
<tr>
<td>Watch for changes and recompile.</td>
<td>In the /cdk-serverless-saga folder, run the following command in a separate terminal window, to watch for code changes and compile the code when it detects a change: npm run watch</td>
<td>Developer, Cloud architect</td>
</tr>
<tr>
<td>Run unit tests.</td>
<td>In the /cdk-serverless-saga folder, run the following command to perform the Jest unit tests: npm run test</td>
<td>Developer, Cloud architect</td>
</tr>
</tbody>
</table>
## Deploy resources to the target AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the demo stack to AWS. | In the /cdk-serverless-saga folder, run the following command to create a deployment assembly and to deploy it to the default AWS account and Region:  
```
cdk deploy
```
This step might take several minutes to complete. This command uses the default credentials that were configured for the AWS Command Line Interface (AWS CLI). To specify named profile credentials, use the command:  
```
cdk deploy --profile <aws_cli_profile_name>
```
Note the API Gateway URL that is displayed on the console after deployment is complete. You will need this information to test the saga execution flow. | Developer, Cloud architect |
| Compare the deployed stack with the current state. | In the /cdk-serverless-saga folder, run the following command to compare the deployed stack with the current state after making changes to the source code:  
```
cdk diff
```
| Developer, Cloud architect |
| Create an AWS CloudFormation template. | In the /cdk-serverless-saga folder, run the following command to synthesize the stack into a CloudFormation template:  
```
cdk synth
```
| Developer, Cloud architect |
Test the execution flow

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the saga execution flow</td>
<td>Navigate to the API Gateway URL that you noted in the earlier step, when you deployed the stack. This URL triggers the state machine to start. For more information about how to manipulate the flow of the state machine by passing different URL parameters, see the Additional information section. To view the results, sign in to the AWS Management Console, and navigate to the Step Functions console. Here, you can see every step of the saga state machine. You can also view the DynamoDB table to see the records inserted, updated, or deleted. If you refresh the screen frequently, you can watch the transaction status change from pending to confirmed. You can subscribe to the SNS topic by updating the code in the stateMachine.ts file with your cell phone number to receive SMS messages upon successful or failed reservations. For more information, see Amazon SNS in the Additional information section.</td>
<td>Developer, Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

Technical papers

- Implementing Microservices on AWS
- Serverless Application Lens
- Enabling Data Persistence in Microservices

AWS service documentation

- Getting started with the AWS CDK
- AWS Step Functions
- Amazon DynamoDB
- AWS Lambda
Additional information

Code

For testing purposes, this pattern deploys API Gateway and a test Lambda function that triggers the Step Functions state machine. With Step Functions, you can control the functionality of the travel reservation system by passing a `run_type` parameter to mimic failures in “ReserveFlight,” “ReserveCarRental,” “ProcessPayment,” “ConfirmFlight,” and “ConfirmCarRental.”

The saga Lambda function (sagaLambda.ts) takes input from the query parameters in the API Gateway URL, creates the following JSON object, and passes it to Step Functions for execution:

```javascript
let input = {
  "trip_id": tripID, // value taken from query parameter, default is AWS request ID
  "depart_city": "Detroit",
  "depart_time": "2021-07-07T06:00:00.000Z",
  "arrive_city": "Frankfurt",
  "arrive_time": "2021-07-09T08:00:00.000Z",
  "rental": "BMW",
  "rental_from": "2021-07-09T00:00:00.000Z",
  "rental_to": "2021-07-17T00:00:00.000Z",
  "run_type": runType // value taken from query parameter, default is "success"
};
```

You can experiment with different flows of the Step Functions state machine by passing the following URL parameters:

- **Successful Execution** – https://[api gateway url]
- **Reserve Flight Fail** – https://[api gateway url]?runType=failFlightsReservation
- **Confirm Flight Fail** – https://[api gateway url]?runType=failFlightsConfirmation
- **Reserve Car Rental Fail** – https://[api gateway url]?runType=failCarRentalReservation
- **Confirm Car Rental Fail** – https://[api gateway url]?runType=failCarRentalConfirmation
- **Process Payment Fail** – https://[api gateway url]?runType=failPayment
- **Pass a Trip ID** – https://[api gateway url]?tripID={by default, trip ID will be the AWS request ID}

**AWS CDK script**

The attached archive includes an AWS CDK script that you can use to create the entire sample travel reservation application.

**DynamoDB tables**

The attached code creates the following DynamoDB tables. These tables hold flight reservations, car rental reservations, and payment information.

```javascript
const flightTable = new dynamodb.Table(this,'Flights',{partitionKey:{name:'pk', type:dynamodb.AttributeType.STRING},sortKey:{name:'sk', type: dynamodb.AttributeType.STRING}})
```
const rentalTable = new dynamodb.Table(this,'Rentals',{partitionKey:{name:'pk', type:dynamodb.AttributeType.STRING},sortKey:{name:'sk', type: dynamodb.AttributeType.STRING}})

const paymentTable = new dynamodb.Table(this,'Payments',{partitionKey:{name:'pk', type:dynamodb.AttributeType.STRING},sortKey:{name:'sk', type: dynamodb.AttributeType.STRING}})

Flight Data Model:
var params = {
    TableName: process.env.TABLE_NAME,
    Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: flightReservationID},
        'trip_id' : {S: event.trip_id},
        'id': {S: flightReservationID},
        'depart_city': {S: event.depart_city},
        'depart_time': {S: event.depart_time},
        'arrive_city': {S: event.arrive_city},
        'arrive_time': {S: event.arrive_time},
        'transaction_status': {S: 'pending'}
    }
};

Car Rental Data Model:
var params = {
    TableName: process.env.TABLE_NAME,
    Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: carRentalReservationID},
        'trip_id' : {S: event.trip_id},
        'id': {S: carRentalReservationID},
        'rental': {S: event.rental},
        'rental_from': {S: event.rental_from},
        'rental_to': {S: event.rental_to},
        'transaction_status': {S: 'pending'}
    }
};

Payment Data Model:
var params = {
    TableName: process.env.TABLE_NAME,
    Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: paymentID},
        'trip_id' : {S: event.trip_id},
        'id': {S: paymentID},
        'amount': {S: "750.00"}, // hard coded for simplicity as implementing any monetary
        transaction functionality is beyond the scope of this pattern
        'currency': {S: "USD"},
        'transaction_status': {S: "confirmed"}
    }
};

Lambda functions

The following functions will be created to support the state machine flow and execution in Step Functions:

- **Reserve Flights**: Inserts a record into the DynamoDB Flights table with a transaction_status of pending, to book a flight.
• **Confirm Flight**: Updates the record in the DynamoDB Flights table, to set transaction_status to confirmed, to confirm the flight.

• **Cancel Flights Reservation**: Deletes the record from the DynamoDB Flights table, to cancel the pending flight.

• **Reserve Car Rentals**: Inserts a record into the DynamoDB CarRentals table with a transaction_status of pending, to book a car rental.

• **Confirm Car Rentals**: Updates the record in the DynamoDB CarRentals table, to set transaction_status to confirmed, to confirm the car rental.

• **Cancel Car Rentals Reservation**: Deletes the record from the DynamoDB CarRentals table, to cancel the pending car rental.

• **Process Payment**: Inserts a record into the DynamoDB Payment table for the payment.

• **Cancel Payment**: Deletes the record from the DynamoDB Payments table for the payment.

```javascript
let fn = new lambda.Function(scope, id, {
  runtime: lambda.Runtime.NODEJS_12_X,
  code: lambda.Code.fromAsset('lambdas'),
  handler:handler,
  environment: {
    TABLE_NAME: table.tableName
  }
});
```

**Amazon SNS**

The sample application creates the following topic and subscription for sending SMS messages and notifying the customer about successful or failed reservations. If you want to receive text messages while testing the sample application, update the SMS subscription with your valid phone number in the stateMachine.ts file (in the second line of the following code):

```javascript
const topic = new sns.Topic(this, 'Topic');
topic.addSubscription(new subscriptions.SmsSubscription('+11111111111'));
const snsNotificationFailure = new tasks.SnsPublish(this, 'SendingSMSFailure', {
  topic:topic,
  integrationPattern: sfn.IntegrationPattern.REQUEST_RESPONSE,
  message: sfn.TaskInput.fromText('Your Travel Reservation Failed'),
});

const snsNotificationSuccess = new tasks.SnsPublish(this, 'SendingSMSSuccess', {
  topic:topic,
  integrationPattern: sfn.IntegrationPattern.REQUEST_RESPONSE,
  message: sfn.TaskInput.fromText('Your Travel Reservation is Successful'),
});
```

**Successful reservations**

The following flow illustrates a successful reservation with “ReserveFlight,” “ReserveCarRental,” and “ProcessPayment” followed by “ConfirmFlight” and “ConfirmCarRental.” The customer is notified about the successful booking through SMS messages that are sent to the subscriber of the SNS topic.
Failed reservations

This flow is an example of failure in the saga pattern. If, after booking flights and car rentals, "ProcessPayment" fails, steps are canceled in reverse order. The reservations are released, and the customer is notified of the failure through SMS messages that are sent to the subscriber of the SNS topic.
Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK

Created by Rahul Sharad Gaikwad (AWS)

**Code repository:**
- amazon-ecs-anywhere-cdk-samples

**Environment:** PoC or pilot

**Technologies:** Modernization; Containers & microservices; DevOps; Hybrid cloud; Infrastructure
Summary

Amazon ECS Anywhere is an extension of the Amazon Elastic Container Service (Amazon ECS). You can use ECS Anywhere to deploy native Amazon ECS tasks in an on-premises or customer-managed environment. This feature helps reduce costs and mitigate complex local container orchestration and operations. You can use ECS Anywhere to deploy and run container applications in both on-premises and cloud environments. It removes the need for your team to learn multiple domains and skill sets, or to manage complex software on their own.

This pattern demonstrates the steps to set up ECS Anywhere by using AWS Cloud Development Kit (AWS CDK) stacks.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. (See Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.)
- AWS CDK Toolkit, installed and configured. (See AWS CDK Toolkit in the AWS CDK documentation, and follow the instructions to install the latest version globally.)
- Node package manager (npm), installed and configured for the AWS CDK in TypeScript. (See Downloading and installing Node.js and npm in the npm documentation.)

Limitations

- For limitations and considerations, see External instances (Amazon ECS Anywhere) in the Amazon ECS documentation.

Product versions

- AWS CDK Toolkit version 1.116.0 or later
- npm version 7.20.3 or later
- Node.js version 16.6.1 or later

Architecture

Target technology stack

- AWS CloudFormation
- AWS CDK
- Amazon ECS Anywhere
- AWS Identity and Access Management (IAM)

Target architecture
1. When you deploy the AWS CDK stack, it creates a CloudFormation stack on AWS.
2. The CloudFormation stack provisions an Amazon ECS cluster and related AWS resources.
3. To register an external instance with an Amazon ECS cluster, you must install AWS Systems Manager Agent (SSM Agent) on your virtual machine (VM) and register the VM as an AWS Systems Manager managed instance.
4. You must also install the Amazon ECS container agent and Docker on your VM to register it as an external instance with the Amazon ECS cluster.
5. When the external instance is registered and configured with the Amazon ECS cluster, it can run multiple containers on your VM, which is registered as an external instance.

Automation and scale

The GitHub repository that is provided with this pattern uses the AWS CDK as an infrastructure as code (IaC) tool to create the configuration for this architecture. AWS CDK helps you orchestrate resources and set up ECS Anywhere.
Tools

- **AWS CDK** – The AWS Cloud Development Kit (AWS CDK) helps you define your cloud infrastructure as code in one of five supported programming languages: TypeScript, JavaScript, Python, Java, and C#.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent command-line interface for interacting with AWS services and resources.

Code

The source code for this pattern is available on GitHub, in the [ECS Anywhere CDK Samples](https://github.com/AWS-ECS-Anywhere/CDK-Samples) repository. To clone and use the repository, follow the instructions in the next section.

Epics

Verify AWS CDK configuration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the AWS CDK version.</td>
<td>Verify the version of the AWS CDK Toolkit by running the following command:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td><code>cdk --version</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This pattern requires version 1.116.0 or later. If you have an earlier version of the AWS CDK, follow the instructions in the AWS CDK documentation to update it.</td>
<td></td>
</tr>
<tr>
<td>Set up AWS credentials.</td>
<td>To set up credentials, run the <code>aws configure</code> command and follow the prompts:</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td><code>$aws configure</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Access Key ID [None]: <code>&lt;your-access-key-ID&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWS Secret Access Key [None]: <code>&lt;your-secret-access-key&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default region name [None]: <code>&lt;your-Region-name&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default output format [None]:</td>
<td></td>
</tr>
</tbody>
</table>

Bootstrap the AWS CDK environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the AWS CDK code repository.</td>
<td>Clone the GitHub code repository for this pattern by using the command:</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
### Bootstrap the environment.

To deploy the AWS CloudFormation template to the account and AWS Region that you want to use, run the following command:

```
cdk bootstrap <account-number>/<Region>
```

For more information, see [Bootstrapping](#) in the AWS CDK documentation.

**Skills required:** DevOps engineer

### Build and deploy the project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Install package dependencies and compile TypeScript files. | Install the package dependencies and compile the TypeScript files by running the following commands:  
*$cd$ amazon-ecs-anywhere-cdk-samples
*$npm install
*$npm fund
These commands install all the packages from the sample repository. **Important:** If you get any errors about missing packages, use one of the following commands:  
*$npm ci
---or---
*$npm install -g @aws-cdk/<package_name>
| DevOps engineer |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the project.</td>
<td>To build the project code, run the command: <strong>npm run build</strong>&lt;br&gt;For more information about building and deploying the project, see Your first AWS CDK app in the AWS CDK documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Deploy the project.</td>
<td>To deploy the project code, run the command: <strong>cdk deploy</strong></td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Verify stack creation and output.</td>
<td>Open the AWS CloudFormation console at <a href="https://console.aws.amazon.com/cloudformation">https://console.aws.amazon.com/cloudformation</a>, and choose the EcsAnywhereStack stack. The Outputs tab shows the commands to run on your external VM.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

### Set up an on-premises machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up your VM by using Vagrant.</td>
<td>For demonstration purposes, you can use HashiCorp Vagrant to create a VM. Vagrant is an open-source utility for building and maintaining portable virtual software development environments. Create a Vagrant VM by running the <em>vagrant up</em> command from the root directory where Vagrantfile is placed. For more information, see the Vagrant documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Register your VM as an external instance.</td>
<td>1. Log in to the Vagrant VM by using the <em>vagrant ssh</em> command. For more information, see the Vagrant documentation.&lt;br&gt;2. Create an activation code and ID that you can use to register your VM with AWS Systems Manager and to activate your external instance. The output from this command</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1.</td>
<td>includes ActivationId and ActivationCode values:</td>
<td>aws ssm create-activation --iam-role EcsAnywhereInstanceRole</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tee ssm-activation.json</td>
</tr>
<tr>
<td>3.</td>
<td>Export the activation ID and code values:</td>
<td>export ACTIVATION_ID=&lt;activation-ID&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>export ACTIVATION_CODE=&lt;activation-code&gt;</td>
</tr>
<tr>
<td>5.</td>
<td>Run the installation script on your on-premises server or VM:</td>
<td>sudo ./ecs-anywhere-install.sh \ --cluster test-ecs-anywhere \ --activation-id $ACTIVATION_ID \ --activation-code $ACTIVATION_CODE \ --region &lt;Region&gt;</td>
</tr>
</tbody>
</table>

For more information about setting up and registering your VM, see [Registering an external instance to a cluster](#) in the Amazon ECS documentation.
### Verify the status of ECS Anywhere and the external VM.

To verify whether your virtual box is connected to the Amazon ECS control plane and running, use the following commands:

```
aws ssm describe-instance-information
aws ecs list-container-instances --cluster $CLUSTER_NAME
```

**Skills required:** DevOps engineer

### Clean up

**Task**

Clean up and delete resources.

**Description**

After you walk through this pattern, you should remove the resources you created to avoid incurring any further charges. To clean up, run the command:

```
ck destroy
```

**Skills required:** DevOps engineer

### Related resources

- Amazon ECS Anywhere Documentation
- Amazon ECS Anywhere Demo
- Amazon ECS Anywhere Workshop Samples

### Modernize ASP.NET Web Forms applications on AWS

*Created by Vijai Anand Ramalingam (AWS) and Sreelaxmi Pai (AWS)*

**Environment:** PoC or pilot

**Technologies:** Modernization; Containers & microservices; Web hosting; Websites & web apps; Software development & testing

**Workload:** Microsoft

**AWS services:** Amazon CloudWatch; Amazon ECS; AWS Systems Manager
Summary

This pattern describes the steps for modernizing a legacy, monolith ASP.NET Web Forms application by porting it to ASP.NET Core on Amazon Web Services (AWS).

Porting ASP.NET Web Forms applications to ASP.NET Core helps you take advantage of the performance, cost savings, and robust ecosystem of Linux. However, it can be a significant manual effort. In this pattern, the legacy application is modernized incrementally by using a phased approach, and then containerized in the AWS Cloud.

Consider a legacy, monolith application for a shopping cart. Let’s assume that it was created as an ASP.NET Web Forms application and consists of .aspx pages with a code-behind (aspx.cs) file. The modernization process consists of these steps:

1. Break the monolith into microservices by using the appropriate decomposition patterns. For more information, see the guide Decomposing monoliths into microservices on the AWS Prescriptive Guidance website.
2. Port your legacy ASP.NET Web Forms (.NET Framework) application to ASP.NET Core in .NET 5 or later. In this pattern, you use Porting Assistant for .NET to scan your ASP.NET Web Forms application and identify incompatibilities with ASP.NET Core. This reduces the manual porting effort.
3. Redevelop the Web Forms UI layer by using React. This pattern doesn’t cover UI redevelopment. For instructions, see Create a New React App in the React documentation.
4. Redevelop the Web Forms code-behind file (business interface) as an ASP.NET Core web API. This pattern uses NDepend reports to help identify required files and dependencies.
5. Upgrade shared/common projects, such as Business Logic and Data Access, in your legacy application to .NET 5 or later by using Porting Assistant for .NET.
6. Add AWS services to complement your application. For example, you can use Amazon CloudWatch Logs to monitor, store, and access your application’s logs, and AWS Systems Manager to store your application settings.
7. Containerize the modernized ASP.NET Core application. This pattern creates a Docker file that targets Linux in Visual Studio and uses Docker Desktop to test it locally. This step assumes that your legacy application is already running on an on-premises or Amazon Elastic Compute Cloud (Amazon EC2) Windows instance. For more information, see the pattern Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance.
8. Deploy the modernized ASP.NET core application to Amazon Elastic Container Service (Amazon ECS). This pattern doesn’t cover the deployment step. For instructions, see the Amazon ECS Workshop.

Note This pattern doesn’t cover UI development, database modernization, or container deployment steps.

Prerequisites and limitations

Prerequisites

- Visual Studio or Visual Studio Code, downloaded and installed.
- Access to an AWS account using the AWS Management Console and the AWS Command Line Interface (AWS CLI) version 2. (See the instructions for configuring the AWS CLI.)
- The AWS Toolkit for Visual Studio (see setup instructions).
- Docker Desktop, downloaded and installed.
- .NET SDK, downloaded and installed.
- NDepend tool, downloaded and installed. To install the NDepend extension for Visual Studio, run NDepend.VisualStudioExtension.Installer (see instructions). You can select Visual Studio 2019 or 2022, depending on your requirements.
• Porting Assistant for .NET, downloaded and installed.

Architecture

Modernizing the shopping cart application

The following diagram illustrates the modernization process for a legacy ASP.NET shopping cart application.

Target architecture

The following diagram illustrates the architecture of the modernized shopping cart application on AWS. ASP.NET Core web APIs are deployed to an Amazon ECS cluster. Logging and configuration services are provided by Amazon CloudWatch Logs and AWS Systems Manager.
Tools

AWS services

- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of EC2 instances that you manage.

- **Amazon CloudWatch Logs** – Amazon CloudWatch Logs centralizes the logs from all your systems, applications, and AWS services that you use. You can view and monitor the logs, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis.

- **AWS Systems Manager** – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting (or taking corrective action) on any policy violations it detects.

Tools

- **Visual Studio** or **Visual Studio Code** – Tools for building .NET applications, web APIs, and other programs.

- **AWS Toolkit for Visual Studio** – An extension for Visual Studio that helps develop, debug, and deploy .NET applications that use AWS services.

- **Docker Desktop** – A tool that simplifies building and deploying containerized applications.

- **NDepend** – An analyzer that monitors .NET code for dependencies, quality issues, and code changes.
- **Porting Assistant for .NET** – An analysis tool that scans .NET code to identify incompatibilities with .NET Core and to estimate the migration effort.

## Epics

### Port your legacy application to .NET 5 or later version

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrade your .NET Framework legacy application to .NET 5.</td>
<td>You can use Porting Assistant for .NET to convert your legacy ASP.NET Web Forms application to .NET 5 or later. Follow the instructions in the Porting Assistant for .NET documentation.</td>
<td>App developer</td>
</tr>
</tbody>
</table>
| Generate NDepend reports.                 | When you modernize your ASP.NET Web Forms application by decomposing it into microservices, you might not need all the .cs files from the legacy application. You can use NDepend to generate a report for any code-behind (.cs) file, to get all the callers and callees. This report helps you identify and use only the required files in your microservices. After you install NDepend (see the Prerequisites (p. 1467) section), open the solution (.sln file) for your legacy application in Visual Studio and follow these steps:  

1. Build the legacy application in Visual Studio.  
2. On the Visual Studio menu bar, choose NDepend, Attach new NDepend project to current VS solution.  
3. Choose Analyze .NET assemblies.  
4. When analysis is complete, navigate to the project in Solution Explorer. Right-click any code-behind file (for example, listproducts.aspx.cs) that you want to generate the report for, and then choose Show on Dependency Graph. | App developer     |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>In the navigation bar, choose <strong>Callers and callees</strong>, and then choose <strong>Edit code query</strong>.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>In the <strong>Queries and Rules Edit pane</strong>, choose the download arrow, and then choose <strong>Export to Excel</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This process generates a report for the code-behind file that lists all callers and callees. For more information about the dependency graph, see the <strong>NDepend documentation</strong>.</td>
<td></td>
</tr>
</tbody>
</table>
| Create a new .NET 5 solution. | To create a new .NET 5 (or later) structure for your modernized ASP.NET Core web APIs:  
2. Create a new, blank solution.  
3. Create new projects that target .NET 5 (or later), based on your legacy application. For examples of legacy and new projects for a shopping cart application, see the **Additional information (p. 1475)** section.  
4. Use the NDepend report from the previous step to identify all required files. Copy these files from the application you upgraded earlier and add them to the new solution.  
5. Build the solution and fix all issues. | **App developer**  
|      | For more information about creating projects and solutions, see the **Visual Studio documentation**. |  
|      | **Note** As you build the solution and verify functionality, you might identify several additional files to be added to the solution, in addition to the files that NDepend identified. |
Update your application code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Implement web APIs with ASP.NET Core. | Let’s assume that one of the microservices that you identified in your legacy monolith shopping cart application is *Products*. You created a new ASP.NET Core web API project for *Products* in the previous epic. In this step, you identify and modernize all the web forms (.aspx pages) that are related to *Products*. Let’s assume that *Products* consists of four web forms, as illustrated earlier in the Architecture (p. 1468) section:  

- List Products  
- View Product  
- Add/Edit Product  
- Delete Product  

You should analyze each web form, identify all the requests that are sent to the database to perform some logic, and get responses. You can implement each request as a web API endpoint. Given its web forms, *Products* can have the following possible endpoints:  

- /api/products  
- /api/products/{id}  
- /api/products/add  
- /api/products/update/{id}  
- /api/products/delete/{id}  

As mentioned previously, you can also reuse all the other projects that you upgraded to .NET 5, including Business Logic, Data Access, and shared/common projects. | App developer |
| Configure Amazon CloudWatch Logs. | You can use Amazon CloudWatch Logs to monitor, store, and access your application’s logs. You can log | App developer |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task</strong></td>
<td><strong>Description</strong></td>
<td><strong>Skills required</strong></td>
</tr>
<tr>
<td>data into Amazon CloudWatch Logs by using an AWS SDK. You can also integrate .NET applications with CloudWatch Logs by using popular .NET logging frameworks such as NLog, Log4Net, and ASP.NET Core logging framework.</td>
<td><strong>App developer</strong></td>
<td></td>
</tr>
<tr>
<td>Configure AWS Systems Manager Parameter Store.</td>
<td>You can use AWS Systems Manager Parameter Store to store application settings such as connection strings separately from your application’s code. The NuGet package Amazon.Extensions.Configuration.SystemsManager simplifies how your application loads these settings from the AWS Systems Manager Parameter Store into the .NET Core configuration system.</td>
<td><strong>App developer</strong></td>
</tr>
</tbody>
</table>

**Add authentication and authorization**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use a shared cookie for authentication.</td>
<td>Modernizing a legacy monolith application is an iterative process and requires the monolith and its modernized version to co-exist. You can use a shared cookie to achieve seamless authentication between the two versions. The legacy ASP.NET application continues to validate user credentials and issues the cookie while the modernized ASP.NET Core application validates the cookie.</td>
<td><strong>App developer</strong></td>
</tr>
</tbody>
</table>

For instructions and sample code, see the [sample GitHub project](https://github.com/).
## Build and run the container locally

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a Docker image by using Visual Studio. | In this step, you create a Docker file by using the Visual Studio for .NET Core web API.  
2. In Solution Explorer, from the context (right-click) menu of your project, choose Add, **Docker Support**.  
3. Select **Linux** as the target operating system.  
Visual Studio creates a Docker file for your project. For a sample Docker file, see **Visual Studio Container Tools for Docker** on the Microsoft website. | App developer |
| Build and run the container by using Docker Desktop. | Now you can build, create and run the container in Docker Desktop.  
1. Open a Command Prompt window. Navigate to the solution folder where the Docker file is located. Run the following command to create the Docker image:  
```
docker build -t aspnetcorewebapiimage -f Dockerfile .
```
2. Run the following command to view all Docker images:  
```
docker images
```
3. Run the following command to create and run a container:  
```
docker run -d -p 8080:80 --name aspnetcorewebapicontainer aspnetcorewebapiimage
```
4. Open Docker Desktop, and then choose **Containers/Apps**. You can see a new container named `aspnetcorewebapicontainer` running. | App developer |
Related resources

- Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance (AWS Prescriptive Guidance)
- Amazon ECS Workshop
- Perform ECS blue/green deployments through CodeDeploy using AWS CloudFormation (AWS CloudFormation documentation)
- Getting started with NDepend (NDepend documentation)
- Porting Assistant for .NET

Additional information

The following tables provide examples of sample projects for a legacy shopping cart application and the equivalent projects in your modernized ASP.NET Core application.

### Legacy solution:

<table>
<thead>
<tr>
<th>Project name</th>
<th>Project template</th>
<th>Target framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Interface</td>
<td>Class Library</td>
<td>.NET Framework</td>
</tr>
<tr>
<td>BusinessLogic</td>
<td>Class Library</td>
<td>.NET Framework</td>
</tr>
<tr>
<td>WebApplication</td>
<td>ASP.NET Framework Web Application</td>
<td>.NET Framework</td>
</tr>
<tr>
<td>UnitTests</td>
<td>NUnit Test Project</td>
<td>.NET Framework</td>
</tr>
<tr>
<td>Shared -&gt;Common</td>
<td>Class Library</td>
<td>.NET Framework</td>
</tr>
<tr>
<td>Shared -&gt;Framework</td>
<td>Class Library</td>
<td>.NET Framework</td>
</tr>
</tbody>
</table>

### New solution:

<table>
<thead>
<tr>
<th>Project name</th>
<th>Project template</th>
<th>Target framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessLogic</td>
<td>Class Library</td>
<td>.NET 5.0</td>
</tr>
<tr>
<td>&lt;WebAPI&gt;</td>
<td>ASP.NET Core Web API</td>
<td>.NET 5.0</td>
</tr>
<tr>
<td>&lt;WebAPI&gt;.UnitTests</td>
<td>NUnit 3 Test Project</td>
<td>.NET 5.0</td>
</tr>
<tr>
<td>Shared -&gt;Common</td>
<td>Class Library</td>
<td>.NET 5.0</td>
</tr>
<tr>
<td>Shared -&gt;Framework</td>
<td>Class Library</td>
<td>.NET 5.0</td>
</tr>
</tbody>
</table>

Run event-driven and scheduled workloads at scale with AWS Fargate

*Created by HARI OHM PRASATH RAJAGOPAL (AWS)*
Summary

This pattern describes how to run scheduled and event-driven workloads at scale on the Amazon Web Services (AWS) Cloud by using AWS Fargate.

In the use case that this pattern sets up, code is scanned for AWS sensitive information, such as the AWS account number and credentials, whenever a pull request is submitted. The pull request initiates a Lambda function. The Lambda function invokes a Fargate task that takes care of the code scan. Lambda is initiated whenever a new pull request is raised. If the scan finds any sensitive information, Amazon Simple Notification Service (Amazon SNS) sends the scan results are sent in an email message.

This pattern is helpful in the following business use cases:

- If your business must run many scheduled and event-driven workloads that cannot be run by AWS Lambda because of limitations around runtime (a 15-minute limit) or memory
- If you want AWS to manage the instances provisioned for these workloads

When you use this pattern, you have the option of creating a new virtual private cloud (VPC).

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS CodeCommit for hosting the code base and creating pull requests
- AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows
- Workloads running in containers
- Apache Maven executable set up in classpath

Architecture
The overall flow includes the following steps.

1. Whenever a new pull request is submitted in CodeCommit, a Lambda function is initiated. The Lambda function listens through the CodeCommit Pull Request State Change event via Amazon EventBridge.

2. The Lambda function submits a new Fargate task with the following environment parameters for checking out the code and scanning it.

   ```
   RUNNER # <<TaskARN>>
   SNS_TOPIC # <<SNSTopicARN>>
   SUBNET # <<Subnet in which Fargate task gets launched>>
   ```

   If the scan finds sensitive information in the code, Fargate pushes a new message to the Amazon SNS topic.

3. An SNS subscriber reads the message from the topic and sends an email message.

**Technology**

- AWS CodeCommit
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon SNS
- Docker

**Tools**

- **AWS CLI** – AWS Command Line Interface (CLI) is a unified tool to manage your AWS services.
- **AWS CodeCommit** – AWS CodeCommit is a fully managed source control service that hosts secure Git-based repositories. Using CodeCommit, teams can collaborate on code in a secure and highly scalable environment.
- **Amazon ECR** – Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that developers can use to store, manage, and deploy Docker container images.
- **Amazon ECS** – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service. You can use Amazon ECS to run, stop, and manage containers on a cluster.
- **AWS Fargate** – AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel. Clients that subscribe to the SNS topic receive published messages using a supported protocol, such as Lambda, email, mobile push notifications, and mobile text messages (SMS).
- **Docker** – Docker helps you build, test, and deliver applications in packages called containers.
- **Git client** – Command line or desktop tool to check out the required artifacts
- **Maven** – Apache Maven is a project management tool for centrally managing a project’s build, reporting, and documentation.

### Epics

**Set up the local repository**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the code.</td>
<td>In the Attachments section, download the .zip file and extract the files.</td>
<td>Developer, AWS system administrator</td>
</tr>
<tr>
<td>Set up the repo.</td>
<td>Run <code>mvn clean install</code> on the root folder.</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>

### Create an Amazon ECR image and push the image

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon ECR repository and log in.</td>
<td>Open the Amazon ECR console. In the navigation pane, choose Repositories, and then choose Create repository. For help with this and other stories, see the Related resources section.</td>
<td>Developer, AWS system administrator</td>
</tr>
<tr>
<td>Push your container image.</td>
<td>Open the repository, choose View push commands, and log in to Docker. After you are logged in, run the commands, with the required substitutions, that are under Push the container image in the Additional information</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>
## Create the CodeCommit repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the CodeCommit</td>
<td>To create a new AWS CodeCommit repository, run the command under Create the CodeCommit repository in the Additional information section.</td>
<td>Developer, AWS system administrator</td>
</tr>
<tr>
<td>repository.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Create the VPC (optional)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC.</td>
<td>If you want to use a new VPC rather than an existing one, run the commands under Create a VPC in the Additional information section. The AWS Cloud Development Kit (AWS CDK) script will output the IDs of the VPC and subnet that were created.</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>

## Create the Amazon ECS cluster and Fargate task

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the cluster and the</td>
<td>To create an Amazon ECS cluster and Fargate task definition, run the commands under Create the cluster and task in the Additional information section. Make sure that the correct VPC ID and Amazon ECR repo URI are passed in as a parameter while running the shell script. The script creates a Fargate task definition that points to the Docker image (responsible for scanning). The script then creates a job and an associated execution role.</td>
<td>Developer, AWS system administrator</td>
</tr>
<tr>
<td>task.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify the Amazon ECS cluster.</td>
<td>Open the Amazon ECS console. In the navigation pane, choose <strong>Clusters</strong>, and choose the newly created Amazon ECS cluster named <strong>Fargate-Job-Cluster</strong>. After this, choose <strong>Task definition</strong> in the navigation pane, and confirm that there is a new task definition with the prefix awscdkfargateecsTaskDef.</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>

**Create the SNS topic and subscriber**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SNS topic.</td>
<td>To create an SNS topic, run the command under <strong>Create the SNS topic</strong> in the <strong>Additional information</strong> section. After creation is successful, note the SNS ARN, which is used in the next step.</td>
<td>Developer, AWS system administrator</td>
</tr>
<tr>
<td>Create the SNS subscriber.</td>
<td>To create an email subscriber for the SNS topic, run the command under <strong>Create the SNS subscriber</strong> in the <strong>Additional information</strong> section. Make sure to replace <strong>TopicARN</strong> and <strong>Email address</strong> used in the CLI command. To receive email notifications, make sure to confirm the email address that is used as a subscriber.</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>

**Create the Lambda function and CodeCommit trigger**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the function and the trigger.</td>
<td>To create a Lambda function with a CodeCommit trigger, run the command under <strong>Lambda function and CodeCommit trigger</strong> in the <strong>Additional information</strong> section. Make sure to replace the parameters with the corresponding values before running the command. The script creates the Lambda function and configures it to be</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>
### Test the application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the application.</td>
<td>If you check in any AWS sensitive information to CodeCommit repo, the Lambda function should be initiated. The Lambda function initiates the Fargate task, which scans the code and sends the scan results in an email notification.</td>
<td>Developer, AWS system administrator</td>
</tr>
</tbody>
</table>

### Related resources

- Creating an Amazon ECR repository
- Pushing Docker images to Amazon ECR

### Additional information

#### Push the container image

```bash
> cd 1-ecr-image-push
> ./run.sh <<ecr-repository>>
```

#### Create the CodeCommit repository

```bash
aws codecommit create-repository --repository-name test-repo --repository-description "My Test repository"
```

#### Create a VPC

```bash
> cd 2-create-vpc
> ./run.sh
```

#### Output

```bash
aws-batch-cdk-vpc-efs-launch-template.privatesubnet = subnet-<<id>>
aws-batch-cdk-vpc-efs-launch-template.publicsubnet = subnet-<<id>>
aws-batch-cdk-vpc-efs-launch-template.vpcid = vpc-<<id>>
```
Create the cluster and task

```bash
> export CDK_DEFAULT_ACCOUNT = <<aws_account_id>>
> export CDK_DEFAULT_REGION = <<aws_region>>
> cd 3-create-ecs-task
> ./run.sh <<vpc-id>> <<ecr-repo-uri>>
```

Output

```bash
aws-cdk-fargate-ecs.CLUSTERNAME = Fargate-Job-Cluster
aws-cdk-fargate-ecs.ClusterARN = <<cluster_arn>>
aws-cdk-fargate-ecs.ContainerARN = Fargate-Container
aws-cdk-fargate-ecs.TaskARN = <<task_arn>>
aws-cdk-fargate-ecs.TaskExecutionRole = <<execution_role_arn>>
aws-cdk-fargate-ecs.TaskRole = <<task_role_arn>>
```

Create the SNS topic

```bash
aws sns create-topic --name code-commit-topic
```

Create the SNS subscriber

```bash
aws sns subscribe \ 
   --topic-arn <<topic_arn>> \ 
   --protocol email \ 
   --notification-endpoint <<email_address>>
```

Lambda function and CodeCommit trigger

```bash
> export CDK_DEFAULT_ACCOUNT = <<aws_account_id>>
> export CDK_DEFAULT_REGION = <<aws_region>>
> cd 5-Lambda-CodeCommit-Trigger
> ./run.sh <<taskarn>> <<snstopicarn>> subnet-<<id>> <<codecommitarn>>
```

Output

```bash
aws-cdk-fargate-lambda-event.Cloudwatchrule = <<cloudwatchrule>>
aws-cdk-fargate-lambda-event.CodeCommitLambda = AWS-Code-Scanner-Function
aws-cdk-fargate-lambda-event.LambdaRole = <<lambdaiamrole>>
```

Attachments

attachment.zip

Set up CI/CD for AWS AppSync GraphQL API updates

1482
## Summary

Continuous integration and continuous delivery (CI/CD) are critical to recognizing the value of modern application development. Amazon Web Services (AWS) provides multiple services and open-source frameworks that help developers set up a CI/CD pipeline quickly. By automating build, testing and deployment, developers can focus on rapidly delivering new functionality instead of on maintaining and securing CI servers, manually testing their applications, and performing any error-prone deployments of the application stack due to a missed step.

This pattern demonstrates how to apply CI/CD best practices to the development, testing, and deployment of the AWS AppSync (GraphQL) backend API. The pattern builds on the AWS Blog post Building Scalable GraphQL APIs on AWS with CDK, TypeScript, AWS AppSync, Amazon DynamoDB, and AWS Lambda. The pattern sets up a CI/CD pipeline for deploying GraphQL API updates to AWS AppSync, a fully managed GraphQL API service. The pattern uses the AWS Cloud Development Kit (AWS CDK) pipelines module to provision a CI/CD pipeline. AWS CodePipeline orchestrates the building and testing of source code using AWS CodeBuild and the deployment of AWS resources using AWS CloudFormation stacks that are synthesized as a build artifact of the AWS CDK code.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account

**Package versions**

The source code (attached) has been tested with the following package versions in the development environment:

- AWS CDK 1.82.0
- Node.js 15.5.0
- npm 6.14.10
- TypeScript 3.9.7

For additional dependencies and libraries used, see `package.json` in the attached source code. Those dependencies and libraries are installed as a part of setting up your development environment as described in the Epics section.

### Architecture

**Target technology stack**

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Modernization; Cloud-native; Software development &amp; testing; Serverless; DevOps</th>
<th>Workload: Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services:</strong></td>
<td>AWS AppSync; AWS CodeBuild; AWS CodePipeline; AWS CloudFormation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Created by Apoorva Kulkarni (AWS)*
Application resources

- AWS AppSync GraphQL API, GraphQL schema, resolvers, API key, data source
- AWS CloudFormation stack
- Amazon DynamoDB table
- AWS Identity and Access Management (IAM) roles and policies
- AWS Lambda function

Pipeline resources

- AWS CloudFormation stack
- AWS CodeBuild projects
- AWS CodePipeline pipeline, webhook
- IAM roles and policies
- AWS Key Management Service (AWS KMS) key and alias
- AWS Secrets Manager
- Amazon Simple Storage Service (Amazon S3) bucket

Target architecture

The following diagram shows details of the stack update architecture.
To add target environments for deployment of the application stack in different accounts and Regions, use the `pipeline.addApplicationStage()` API of the AWS CDK pipeline module. For more information about cross-account deployments, see the documentation.

**Tools**

- **AWS AppSync** – AWS AppSync provides a robust, scalable GraphQL interface for application developers to combine data from multiple sources, including Amazon DynamoDB and AWS Lambda.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services.
- **AWS Cloud Development Kit** – The AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources.
- **AWS CodeBuild** – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- **AWS CodePipeline** – AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.
- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- **AWS Lambda** – AWS Lambda supports running code without provisioning or managing servers.
- **Node.js runtime** – Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- **Git client** – A Git client is a command line or desktop tool for checking out the required artifacts from a Git repository.
## Epics

### Set up the development environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up the workstation.</td>
<td>Install the Node.js runtime, AWS CDK, Git, and TypeScript.</td>
<td>Developer</td>
</tr>
<tr>
<td>Install the AWS CDK.</td>
<td>To set up a development environment for the AWS CDK, follow the instructions in the AWS CDK documentation.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create a GitHub repository for the source code.</td>
<td>Create a new GitHub repo called cdk-graphql-backend.</td>
<td>Developer</td>
</tr>
<tr>
<td>Push the code.</td>
<td>Download the attached .zip file. In the project directory, set up the GitHub repository as origin, commit the changes, and push them to the origin.</td>
<td>Developer</td>
</tr>
<tr>
<td>Install dependencies.</td>
<td>To install all the dependencies of the reference example as defined in package.json, run the npm install command.</td>
<td>Developer</td>
</tr>
<tr>
<td>Set up the AWS CLI.</td>
<td>Set up the AWS CLI. Use profiles or environment variables so that you are able to issue AWS CLI commands against your AWS account. For more information, see the AWS documentation.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

```bash
cd cdk-graphql-backend
git init --initial-branch=main
git add .
git commit -m 'initial commit'
git remote add origin https://github.com/<GitHub Org>/cdk-graphql-backend.git
git push origin main
```

```bash
npm install
npm run build
```
## Create credentials.

Obtain GitHub token, and create an AWS Secrets Manager secret to use for CodePipeline integration.

```bash
aws secretsmanager create-secret
   --name GITHUB_TOKEN
   --description "Github Token"
   --secret-string "<Insert Github Oauth Token>"
```

Skills required: Developer

## Bootstrap the AWS CDK.

To bootstrap the AWS CDK, use the `CDK_NEW_BOOTSTRAP` flag.

```bash
env CDK_NEW_BOOTSTRAP=1 cdk bootstrap
   --cloudformation-execution-policies
   arn:aws:iam::aws:policy/AdministratorAccess
   aws://<AWS ACCOUNT ID>/us-west-2
```

Skills required: Developer

## Develop the GraphQL API backend

### Task | Description | Skills required
--- | --- | ---
Create the GraphQL schema. | For information on creating the GraphQL schema, see the AWS documentation. | Developer
Create AWS AppSync GraphQL API. | Create an AWS AppSync API backend by using AWS CDK code, as shown in the source code. | Developer
Create the Lambda data source. | Develop a Lambda function to serve as the data source for the AWS AppSync resolvers for the queries and mutations. | Developer
Create the AWS Appsync resolvers. | To route the queries and mutations to the appropriate data source, create resolvers in AWS AppSync. | Developer
Create the data stores. | Create data stores, such as DynamoDB tables, for your GraphQL API. | Developer
Set up the CI/CD pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Source stage.</td>
<td>Set up the source stage so that commits can activate the pipeline.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the Build stage.</td>
<td>Set up the Build stage to build the AWS CDK source code. In addition, build any Lambda code that is provided as the data source for the AWS AppSync API.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the Unit Tests stage.</td>
<td>Create unit tests to test the synthesized CloudFormation stacks. Include application unit tests for Lambda functions, GraphQL schema validation, and so forth.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the Alpha Deployment stage.</td>
<td>Create a stage to deploy the application stack to an Alpha environment for further testing and validations.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the End-to-End Test stage.</td>
<td>Create end-to-end tests to validate the AWS AppSync GraphQL API endpoint returns appropriate results for queries and mutations.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the Production Deployment stage.</td>
<td>Create a stage to deploy the application stack to a production environment for end users. To deploy the pipeline stack, run the following command.</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>cdk deploy --parameters GitHubOrg=&lt;GitHub Org&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Related resources

- AWS AppSync
- AWS CodePipeline
- AWS CodeBuild
- AWS CloudFormation
- AWS CDK
- Getting started with the AWS CDK
- AWS CDK pipelines module
- Amplify JavaScript libraries
- GraphQL
- graphql-schema-utilities
• Building Scalable GraphQL APIs on AWS with CDK, TypeScript, AWS AppSync, Amazon DynamoDB, and AWS Lambda (blog post)
• Merging GraphQL schema files and more from the CLI (blog post)
• jest-runner-groups
• Snapshot testing with Jest

Additional testing with Jest

Testing

Unit testing

The example shows how to perform unit testing of the application's synthesized CloudFormation stack by using the Jest library's snapshot testing capabilities. A snapshot test case renders a stack, takes a snapshot, then compares it to a reference snapshot file stored alongside the test. If the two snapshots do not match, the test will fail. Snapshot mismatches can occur either because the change is unexpected or because the reference snapshot must be updated to the new version of the stack.

```javascript
test('Snapshot test - AppsyncCdkApp', () => {
    const stack = new Stack();
    // WHEN
    new AppSyncCdkApp.AppsyncCdkAppStack(stack, 'MyTestConstruct');
    // THEN
    expect(SynthUtils.toCloudFormation(stack)).toMatchSnapshot();
});
```

Updating tests

To update the unit tests when adding test cases or making changes to the AWS CDK code of the application stack, first update snapshot files. To update the snapshot files, run the following command.

```
$ npm test -- -u --group=unit
```

End-to-end testing

Testing your AWS AppSync GraphQL API backend through its API interface gives you the most confidence to determine whether your API is working as expected for your customers. The following example shows how to use the Amplify JavaScript library combined with the Jest snapshot testing described in the Unit testing section. You can use this approach to perform end-to-end tests that mimic an API client performing operations against the Alpha environment.

```javascript
import Amplify, { API } from 'aws-amplify';
Amplify.configure({
    aws_appsync_region: process.env.AWS_DEFAULT_REGION, // Stack region
    aws_appsync_graphqlEndpoint: process.env.API_URL, // AWS AppSync endpoint
    aws_appsync_authenticationType: "API_KEY", //Primary AWS AppSync authentication type
    aws_appsync_apiKey: process.env.API_KEY // AppSync API Key
});

describe('List Empty Notes', () => {
    it('List Notes, should be empty', async () => {
        const query = `query listNotes {
            listNotes {
                id name completed
            }
        }`
        const response = await API.graphql({
            query
```
Updating Tests

To update either the end-to-end tests when updating the graphql schema or making other changes, run the following command.

```
$ npm test --u --group=e2e
```

Pipeline stages

After the pipeline has been deployed successfully, further changes to the application code or the pipeline can be deployed by pushing commits to GitHub. The following sections describe what the different stages in the pipeline do.

Source

Creates a webhook for integrating with GitHub so that any commit to the main branch initiates the pipeline process.

Build

Creates a CodeBuild project to build the source code. The AWS CDK pipelines module uses this stage to synthesize CloudFormation stacks. Additionally, in this pattern, a TypeScript Lambda function, which is included in the application source code, is also compiled.

UpdatePipeline (self-mutate)

Applies any changes to the pipeline itself through self-mutation using a CloudFormation change set. For more information, see the AWS CDK pipeline module documentation.

Assets

Creates a CodeBuild project to prepare and publish AWS CDK assets. For more information, see the AWS CDK pipeline module documentation.

UnitTests

Creates a CodeBuild project to run any unit tests that exist in the source code. In addition, you can use this stage to validate the graphql schema using graphql-schema-utilities, an open-source package for validating and merging graphql schemas.

DeployAlpha

Uses a CloudFormation change set to deploy the application stack that provisions application resources, such as AWS AppSync, DynamoDB tables, and Lambda functions, to an Alpha environment where you can run integration tests, end-to-end tests, and so forth.

E2ETesting

Creates a CodeBuild project to perform end-to-end testing against the Alpha environment.

DeployProd

Uses a CloudFormation change set to deploy the application stack that provisions application resources to the production environment.
Decompose monoliths into microservices by using CQRS and event sourcing

*Created by Rodolfo Jr. Cerrada (AWS), Dmitry Gulin (AWS), and Tabby Ward (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>Monolith CRUD model</th>
<th>Target:</th>
<th>Microservices</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Re-architect</td>
<td>Workload:</td>
<td>Open-source</td>
<td>Technologies:</td>
<td>Modernization; Messaging &amp; communications; Serverless</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon DynamoDB; AWS Lambda; Amazon SNS

### Summary

This pattern combines two patterns, using both the command query responsibility separation (CQRS) pattern and the event sourcing pattern. The CQRS pattern separates responsibilities of the command and query models. The event sourcing pattern takes advantage of asynchronous event-driven communication to improve the overall user experience.

You can use CQRS and Amazon Web Services (AWS) services to maintain and scale each data model independently while refactoring your monolith application into microservices architecture. Then you can use the event sourcing pattern to synchronize data from the command database to the query database.

This pattern uses example code that includes a solution (*.sln) file that you can open using the latest version of Visual Studio. The example contains Reward API code to showcase how CQRS and event sourcing work in AWS serverless and traditional or on-premises applications.

To learn more about CQRS and event sourcing, see the *Additional information* section.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account
- Amazon CloudWatch
- Amazon DynamoDB tables
- Amazon DynamoDB Streams
- AWS Identity and Access Management (IAM) access key and secret key; for more information, see the video in the *Related resources* section
- AWS Lambda
• Familiarity with Visual Studio
• Familiarity with AWS Toolkit for Visual Studio; for more information, see the AWS Toolkit for Visual Studio demo video in the Related resources section

Product versions
• .NET Core 3.1. This component is an option in the Visual Studio installation. To include .NET Core during installation, select NET Core cross-platform development.

Limitations
• The example code for a traditional on-premises application (ASP.NET Core Web API and data access objects) does not come with a database. However, it comes with the CustomerData in-memory object, which acts as a mock database. The code provided is enough for you to test the pattern.

Architecture

Source technology stack
• ASP.NET Core Web API project
• IIS Web Server
• Data access object
• CRUD model

Source architecture
In the source architecture, the CRUD model contains both command and query interfaces in one application. For example code, see CustomerDAO.cs (attached).

Target technology stack
• Amazon DynamoDB
• Amazon DynamoDB Streams
• AWS Lambda
• (Optional) Amazon API Gateway
• (Optional) Amazon Simple Notification Service (Amazon SNS)

Target architecture

In the target architecture, the command and query interfaces are separated. The architecture shown in the following diagram can be extended with API Gateway and Amazon SNS. For more information, see the Additional information section.

1. Command Lambda functions perform write operations, such as create, update, or delete, on the database.
2. Query Lambda functions perform read operations, such as get or select, on the database.
3. This Lambda function processes the DynamoDB streams from the Command database and updates the Query database for the changes.

Tools

Tools

• Amazon DynamoDB – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
• Amazon DynamoDB Streams – DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table. It then stores this information in a log for up to 24 hours. Encryption at rest encrypts the data in DynamoDB streams.
• AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
• **AWS Management Console** – The AWS Management Console is a web application that comprises a broad collection of service consoles for managing AWS services.


• **AWS Toolkit for Visual Studio** – The AWS Toolkit for Visual Studio is a plugin for the Visual Studio IDE. The AWS Toolkit for Visual Studio makes it easier for you to develop, debug, and deploy .NET applications that use AWS services.

**Code**

The example code is attached. For instructions on deploying the example code, see the *Epics* section.

**Epics**

**Open and build the solution**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Open the solution. | 1. Download the example source code ([CQRS-ES Code.zip](#)) from the Attachments section, and extract the files.  
  2. In the Visual Studio IDE, choose File, Open, Project Solution, and navigate to the folder where you extracted the source code.  
  3. Choose [AWS.APG.CQRSES.sln](#), and then choose Open. The entire solution is loaded into Visual Studio. | App developer            |
| Build the solution.| Open the context (right-click) menu for the solution, and then choose Build Solution. This will build and compile all the projects in the solution. It should compile successfully.  
  Visual Studio Solution Explorer should show the directory structure.  
  - CQRS On-Premises Code Sample contains an example of using CQRS on premises.  
  - CQRS AWS Serverless contains all the CQRS and event-sourcing example code using AWS serverless services. | App developer            |
## Build the DynamoDB tables

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide credentials.</td>
<td>If you don't have an access key yet, see the video in the Related resources section.</td>
<td>App developer, Data engineer, DBA</td>
</tr>
<tr>
<td></td>
<td>1. In Solution Explorer, expand <strong>CQRS AWS Serverless</strong>, and then expand the <strong>Build</strong> solution folder.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Expand the <strong>AwS.APG.CQRSES.Build</strong> project and view the <strong>Program.cs</strong> file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Scroll to the top of <strong>Program.cs</strong> and look for <strong>Program()</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Replace <strong>YOUR ACCESS KEY</strong> with your account access key, and replace <strong>YOUR SECRET KEY</strong> with your account secret key. Note that in a production environment, you would not hardcode your keys. Instead, you could use <strong>AWS Secrets Manager</strong> to store and retrieve the credentials.</td>
<td></td>
</tr>
<tr>
<td>Build the project.</td>
<td>To build the project, open the context (right-click) menu for the <strong>AwS.APG.CQRSES.Build</strong> project, and then choose <strong>Build</strong>.</td>
<td>App developer, Data engineer, DBA</td>
</tr>
<tr>
<td>Build and populate the tables.</td>
<td>To build the tables and populate them with seed data, open the context (right-click) menu for the <strong>AwS.APG.CQRSES.Build</strong> project, and then choose <strong>Debug, Start New Instance</strong>.</td>
<td>App developer, Data engineer, DBA</td>
</tr>
<tr>
<td>Verify the table construction and the data.</td>
<td>To verify, navigate to <strong>AWS Explorer</strong>, and expand <strong>Amazon DynamoDB</strong>. It should display the tables. Open each table to display the example data.</td>
<td>App developer, Data engineer, DBA</td>
</tr>
</tbody>
</table>

### Run local tests

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the CQRS project.</td>
<td>1. Open the solution, and navigate to the <strong>CQRS AWS</strong></td>
<td>App developer, Test engineer</td>
</tr>
</tbody>
</table>
### Services/CQRS/Tests

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Navigate to the <strong>CQRS AWS Services/Event Source/Tests</strong> solution folder.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>In the <strong>AWS.APG.CQRSES.CQRSLambda.Tests</strong> project, open <strong>BaseFunctionTest.cs</strong>, and replace <strong>AccessKey</strong> and <strong>SecretKey</strong> with the IAM keys that you created.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Save the changes.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>To compile and build the test project, open the context (right-click) menu for the project, and then choose <strong>Build</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

### Build the event-sourcing project.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Navigate to the <strong>CQRS AWS Services/Event Source/Tests</strong> solution folder.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>In the <strong>AWS.APG.CQRSES.EventSourceLambda.Tests</strong> project, open <strong>BaseFunctionTest.cs</strong>, and replace <strong>AccessKey</strong> and <strong>SecretKey</strong> with the IAM keys that you created.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Save the changes.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>To compile and build the test project, open the context (right-click) menu for the project, and then choose <strong>Build</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

### Run the tests.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>To run all tests, choose <strong>View</strong>, <strong>Test Explorer</strong>, and then choose <strong>Run All Tests In View</strong>. All tests should pass, which is indicated by a green check mark icon.</td>
<td></td>
</tr>
</tbody>
</table>

### Publish the CQRS Lambda functions to AWS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Publish the first Lambda function. | 1. In Solution Explorer, open the context (right-click) menu for the **AWS.APG.CQRSES.CommandCreateLambda** project, and then choose **Publish to AWS Lambda**.  
2. Select the profile that you want to use and the AWS Region where you want to |  |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>deploy the Lambda function, and the function name.</td>
<td>App developer, DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>3. For the remaining fields, keep the default values, and choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. In <strong>Role Name</strong> dropdown list, select <strong>AWSLambdaFullAccess</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. To provide your account keys, choose <strong>Add</strong>, and enter <strong>AccessKey</strong> as the variable and your access key as the value. Then choose <strong>Add</strong> again, enter <strong>SecretKey</strong> as the variable and your secret key as the value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. For the remaining fields, keep the default values, and choose <strong>Upload</strong>. After the Lambda test function uploads, it appears in Visual Studio automatically.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7. Repeat steps 1-6 for the following projects:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS.APG.CQRSES.CommandDeleteLambda</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS.APG.CQRSES.CommandUpdateLambda</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS.APG.CQRSES.CommandAddRewardLambda</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS.APG.CQRSES.CommandRedeemRewardLambda</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS.APG.CQRSES.QueryCustomerListLambda</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS.APG.CQRSES.QueryRewardLambda</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Verify the function upload.</strong> (Optional) You can verify that the function was successfully loaded by navigating to AWS Explorer and expanding <strong>AWS Lambda</strong>. To open the test window, choose the Lambda function (double-click).</td>
<td></td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Test the Lambda function. | 1. Enter the request data, or copy an example request data from **Test data** in the **Additional information** section. Make sure that you select data that is for the function you are testing.  
2. To run the test, choose **Invoke**. The response and any errors are displayed in the **Response** text box, and logs are shown in the **Logs** text box or in CloudWatch Logs.  
3. To verify the data, in AWS Explorer, choose the DynamoDB table (double-click). | App developer, DevOps engineer |

All CQRS Lambda projects are found under the **CQRS AWS Serverless\CQRS\Command Microservice** and **CQRS AWS Serverless\CQRS\Command Microservice** solution folders. For the solution directory and projects, see **Source code directory** in the **Additional information** section.

<table>
<thead>
<tr>
<th>Publish the remaining functions.</th>
<th>Repeat the previous steps for the following projects:</th>
<th>App developer, DevOps engineer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• AWS.APG.CQRSES.CommandDeleteLambda</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AWS.APG.CQRSES.CommandUpdateLambda</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AWS.APG.CQRSES.CommandAddRewardLambda</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AWS.APG.CQRSES.CommandRedeemRewardLambda</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AWS.APG.CQRSES.QueryCustomerListLambda</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• AWS.APG.CQRSES.QueryRewardLambda</td>
<td></td>
</tr>
</tbody>
</table>

### Set up the Lambda function as an event listener

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Publish the Customer and Reward Lambda event handlers. | To publish each event handler, follow the steps in the preceding epic.  
The projects are under the **CQRS AWS Serverless\Event Source\Customer Event** and **CQRS AWS Serverless** | App developer |
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Attach the event-sourcing Lambda event listener. | 1. Log in to the AWS Management Console using the same account you use when you publish the Lambda projects.  
2. For the **Region**, select **US East 1** or the Region where you deployed the Lambda functions in the previous epic.  
3. Navigate to the **Lambda** service.  
4. Select the **EventSourceCustomer** Lambda function.  
5. Choose **Add Trigger**.  
6. In the **Trigger configuration** dropdown list, select **DynamoDB**.  
7. In the **DynamoDB table** dropdown list, select **cqrses-customer-cmd**.  
8. In the **Starting position** dropdown list, select **Trim horizon** from . **Trim horizon** means that the DynamoDB trigger will start reading at the last (untrimmed) stream record, which is the oldest record in the shard.  
9. Select the **Enable trigger** check box.  
10. For the remaining fields, keep the default values, and choose **Add**.  

After the listener is successfully attached to the DynamoDB table, it will be displayed on the Lambda designer page. | **App developer** |
| Publish and attach the EventSourceReward Lambda function. | To publish and attach the EventSourceReward Lambda function, repeat the steps in the previous two stories, selecting **cqrses-reward-cmd** from the **DynamoDB table** dropdown list. | **App developer** |
## Test and validate the DynamoDB streams and Lambda trigger

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Test the stream and the Lambda trigger. | 1. In Visual Studio, navigate to AWS Explorer.  
2. Expand AWS Lambda, and choose the `CommandRedeemReward` function (double-click). In the function window that opens, you can test the function.  
3. In the Request text box, enter the request data in JavaScript Object Notation (JSON) format. For an example request, see Test data in the Additional information section.  
4. Choose Invoke. | App developer |
| Validate, using the DynamoDB reward query table. | 1. Open the `cqrses-reward-query` table.  
2. Check the points of the customer that redeemed the reward. The redeemed points should be subtracted from the customer's total aggregated points. | App developer |
| Validate, using CloudWatch Logs. | 1. Navigate to CloudWatch and choose Log groups.  
2. The `/aws/lambda/EventSourceReward` log group contains the logs for the EventSourceReward trigger. All Lambda calls are logged, including the messages you placed in `context.Logger.LogLine` and `Console.WriteLine` in the Lambda code. | App developer |
| Validate the EventSourceCustomer trigger. | To validate the EventSourceCustomer trigger, repeat the steps in this epic, using the EventSourceCustomer trigger's respective customer table and CloudWatch logs. | App developer |

### Related resources

**References**
Visual Studio 2019 Community Edition downloads
AWS Toolkit for Visual Studio download
AWS Toolkit for Visual Studio User Guide
Serverless on AWS
DynamoDB Use Cases and Design Patterns
Martin Fowler CQRS
Martin Fowler Event Sourcing

Videos
AWS Toolkit for Visual Studio demo
How do I create an access key ID for a new IAM user?

Additional information

CQRS and event sourcing

CQRS

The CQRS pattern separates a single conceptual operations model, such as a data access object single CRUD (create, read, update, delete) model, into command and query operations models. The command model refers to any operation, such as create, update, or delete, that changes the state. The query model refers to any operation that returns a value.

1. The Customer CRUD model includes the following interfaces:
   - Create Customer()
   - UpdateCustomer()
   - DeleteCustomer()
   - AddPoints()
   - RedeemPoints()
   - GetVIPCustomers()
   - GetCustomerList()
   - GetCustomerPoints()

As your requirements become more complex, you can move from this single-model approach. CQRS uses a command model and a query model to separate the responsibility for writing and reading data. That way, the data can be independently maintained and managed. With a clear separation of responsibilities, enhancements to each model do not impact the other. This separation improves maintenance and performance, and it reduces the complexity of the application as it grows.
1. Interfaces in the Customer Command model:
   - CreateCustomer()
   - UpdateCustomer()
   - DeleteCustomer()
   - AddPoints()
   - RedeemPoints()
2. Interfaces in the Customer Query model:
   - GetVIPCustomers()
   - GetCustomerList()
   - GetCustomerPoints()
   - GetMonthlyStatement()

For example code, see Source code directory.

The CQRS pattern then decouples the database. This decoupling leads to the total independence of each service, which is the main ingredient of microservice architecture.

Using CQRS in the AWS Cloud, you can further optimize each service. For example, you can set different compute settings or choose between a serverless or a container-based microservice. You can replace your
on-premises caching with Amazon ElastiCache. If you have an on-premises publish/subscribe messaging, you can replace it with Amazon Simple Notification Service (Amazon SNS). Additionally, you can take advantage of pay-as-you-go pricing and the wide array of AWS services that you pay only for what you use.

CQRS includes the following benefits:

- **Independent scaling** – Each model can have its scaling strategy adjusted to meet the requirements and demand of the service. Similar to high-performance applications, separating read and write enables the model to scale independently to address each demand. You can also add or reduce compute resources to address the scalability demand of one model without affecting the other.
- **Independent maintenance** – Separation of query and command models improves the maintainability of the models. You can make code changes and enhancements to one model without affecting the other.
- **Security** – It's easier to apply the permissions and policies to separate models for read and write.
- **Optimized reads** – You can define a schema that is optimized for queries. For example, you can define a schema for the aggregated data and a separate schema for the fact tables.
- **Integration** – CQRS fits well with event-based programming models.
- **Managed complexity** – The separation into query and command models is suited to complex domains.

When using CQRS, keep in mind the following caveats:

- The CQRS pattern applies only to a specific portion of an application and not the whole application. If implemented on a domain that does not fit the pattern, it can reduce productivity, increase risk, and introduce complexity.
- The pattern works best for frequently used models that have an imbalance read and write operations.
- For read-heavy applications, such as large reports that take time to process, CQRS gives you the option to select the right database and create a schema to store your aggregated data. This improves the response time of reading and viewing the report by processing the report data only one time and dumping it in the aggregated table.
- For the write-heavy applications, you can configure the database for write operations and allow the command microservice to scale independently when the demand for write increases. For examples, see the AWS.APG.CQRSES.CommandRedeemRewardLambda and AWS.APG.CQRSES.CommandAddRewardLambda microservices.

**Event sourcing**

The next step is to use event sourcing to synchronize the query database when a command is run. For example, consider the following events:

- A customer reward point is added that requires the customer total or aggregated reward points in the query database to be updated.
- A customer’s last name is updated in the command database, which requires the surrogate customer information in the query database to be updated.

In the traditional CRUD model, you ensure consistency of data by locking the data until it finishes a transaction. In event sourcing, the data are synchronized through publishing a series of events that will be consumed by a subscriber to update its respective data.

The event-sourcing pattern ensures and records a full series of actions taken on the data and publishes it through a sequence of events. These events represent a set of changes to the data that subscribers of that event must process to keep their record updated. These events are consumed by the subscriber, synchronizing the data on the subscriber’s database. In this case, that’s the query database.
The following diagram shows event sourcing used with CQRS on AWS.

1. Command Lambda functions perform write operations, such as create, update, or delete, on the database.
2. Query Lambda functions perform read operations, such as get or select, on the database.
3. This Lambda function processes the DynamoDB streams from the Command database and updates the Query database for the changes. You can also use this function also to publish a message to Amazon SNS so that its subscribers can process the data.
4. (Optional) The Lambda event subscriber processes the message published by Amazon SNS and updates the Query database.
5. (Optional) Amazon SNS sends email notification of the write operation.

On AWS, the query database can be synchronized by DynamoDB Streams. DynamoDB captures a time-ordered sequence of item-level modifications in a DynamoDB table in near-real time and durably stores the information within 24 hours.

Activating DynamoDB Streams enables the database to publish a sequence of events that makes the event sourcing pattern possible. The event sourcing pattern adds the event subscriber. The event subscriber application consumes the event and processes it depending on the subscriber's responsibility. In the previous diagram, the event subscriber pushes the changes to the Query DynamoDB database to keep the data synchronized. The use of Amazon SNS, the message broker, and the event subscriber application keeps the architecture decoupled.

Event sourcing includes the following benefits:

- Consistency for transactional data
- A reliable audit trail and history of the actions, which can be used to monitor actions taken in the data
- Allows distributed applications such as microservices to synchronize their data across the environment
- Reliable publication of events whenever the state changes
- Reconstructing or replaying of past states
- Loosely coupled entities that exchange events for migration from a monolithic application to microservices
- Reduction of conflicts caused by concurrent updates; event sourcing avoids the requirement to update objects directly in the data store
- Flexibility and extensibility from decoupling the task and the event
- External system updates
• Management of multiple tasks in a single event

When using event sourcing, keep in mind the following caveats:

• Because there is some delay in updating data between the source subscriber databases, the only way to undo a change is to add a compensating event to the event store.
• Implementing event sourcing has a learning curve since its different style of programming.

Test data

Use the following test data to test the Lambda function after successful deployment.

**CommandCreate Customer**

```
{  "Id":1501,  "Firstname":"John",  "Lastname":"Done",  "CompanyName":"AnyCompany",  "Address": "USA",  "VIP":true }
```

**CommandUpdate Customer**

```
{  "Id":1501,  "Firstname":"John",  "Lastname":"Doe",  "CompanyName":"Example Corp.",  "Address": "Seattle, USA",  "VIP":true }
```

**CommandDelete Customer**

Enter the customer ID as request data. For example, if the customer ID is 151, enter 151 as request data.

```
151
```

**QueryCustomerList**

This is blank. When it is invoked, it will return all customers.

**CommandAddReward**

This will add 40 points to customer with ID 1 (Richard).

```
{
  "Id":10101,
  "CustomerId":1,
  "Points":40
}
```

**CommandRedeemReward**

This will deduct 15 points to customer with ID 1 (Richard).

```
{
  "Id":10110,
  "CustomerId":1,
  "Points":15
}
```

**QueryReward**

Enter the ID of the customer. For example, enter 1 for Richard, 2 for Arnav, and 3 for Shirley.

```
2
```
Source code directory

Use the following table as a guide to the directory structure of the Visual Studio solution.

CQRS On-Premises Code Sample solution directory

Customer CRUD model

CQRS On-Premises Code Sample\CRUD Model\AWS.APG.CQRSES.DAL project

CQRS version of the Customer CRUD model

- Customer command: CQRS On-Premises Code Sample\CQRS Model\Command Microservice \AWS.APG.CQRSES.Commandproject
• **Customer query:** CQRS On-Premises Code Sample\CQRS Model\Query Microservice
  \AWS.APG.CQRSES.Query project

**Command and Query microservices**

The Command microservice is under the solution folder CQRS On-Premises Code Sample\CQRS Model\Command Microservice:

• **AWS.APG.CQRSES.CommandMicroservice** ASP.NET Core API project acts as the entry point where consumers interact with the service.
• **AWS.APG.CQRSES.Command** .NET Core project is an object that hosts command-related objects and interfaces.

The query microservice is under the solution folder CQRS On-Premises Code Sample\CQRS Model\Query Microservice:

• **AWS.APG.CQRSES.QueryMicroservice** ASP.NET Core API project acts as the entry point where consumers interact with the service.
• **AWS.APG.CQRSES.Query** .NET Core project is an object that hosts query-related objects and interfaces.

**CQRS AWS Serverless code solution directory**

```
+ Solution 'AWS.APG.CQRSES' (18 of 18 projects)
  + CQRS AWS Serverless
    + Build
      + AWS.APG.CQRSES.Build
    + CQRS
      + Command Microservice
        + Customer
          + AWS.APG.CQRSES.CommandCreateLambda
          + AWS.APG.CQRSES.CommandDeleteLambda
          + AWS.APG.CQRSES.CommandUpdateLambda
        + Reward
          + AWS.APG.CQRSES.CommandAddRewardLambda
          + AWS.APG.CQRSES.CommandRedeemRewardLambda
      + Query Microservice
        + Customer
          + AWS.APG.CQRSES.QueryCustomerListLambda
        + Reward
          + AWS.APG.CQRSES.QueryRewardLambda
      + Tests
    + Event Source
      + Customer Event
        + AWS.APG.CQRSES.EventSourceCustomer
      + Reward Event
        + AWS.APG.CQRSES.EventSourceReward
      + Tests
    + CQRS On-Premises Code Sample
```
This code is the AWS version of the on-premises code using AWS serverless services.

In C# .NET Core, each Lambda function is represented by one .NET Core project. In this pattern's example code, there is a separate project for each interface in the command and query models.

**CQRS using AWS services**

You can find the root solution directory for CQRS using AWS serverless services is in the CQRS AWS Serverless\CQRS folder. The example includes two models: Customer and Reward.

The command Lambda functions for Customer and Reward are under CQRS\Command Microservice \Customer and CQRS\Command Microservice\Reward folders. They contain the following Lambda projects:

- Customer command: CommandCreateLambda, CommandDeleteLambda, and CommandUpdateLambda
- Reward command: CommandAddRewardLambda and CommandRedeemRewardLambda

The query Lambda functions for Customer and Reward are found under the CQRS\Query Microservice\Customer and CQRS\QueryMicroservice\Reward folders. They contain the QueryCustomerListLambda and QueryRewardLambda Lambda projects.

**CQRS test project**

The test project is under the CQRS\Tests folder. This project contains a test script to automate testing the CQRS Lambda functions.

**Event sourcing using AWS services**

The following Lambda event handlers are initiated by the Customer and Reward DynamoDB streams to process and synchronize the data in query tables.

- The EventSourceCustomer Lambda function is mapped to the Customer table (cqrses-customer-cmd) DynamoDB stream.
- The EventSourceReward Lambda function is mapped to the Reward table (cqrses-reward-cmd) DynamoDB stream.

**Attachments**

attachment.zip

**More patterns**

- Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer (p. 138)
- Automate adding or updating Windows registry entries using AWS Systems Manager (p. 687)
- Automate migration strategy identification and planning using AppScore (p. 747)
- Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline (p. 156)
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK (p. 395)
- Create a pipeline and AMI using CodePipeline and HashiCorp Packer (p. 447)
- Create a pipeline and deploy artifact updates to on-premises EC2 instances using CodePipeline (p. 452)
- Deploy Lambda functions with container images (p. 191)
AWS Prescriptive Guidance Patterns
More patterns

- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT (p. 792)
- Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions (p. 728)
- Migrate an AWS member account from AWS Organizations to AWS Control Tower (p. 733)
- Optimize AWS App2Container generated Docker images (p. 229)
- Perform cross-Region failover and failback by using CloudEndure Disaster Recovery (p. 1826)
- Run Amazon ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere (p. 251)
- Set up a Helm v3 chart repository in Amazon S3 (p. 524)
- Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization (p. 709)
- Use CloudEndure for disaster recovery of an on-premises database (p. 1834)
Verify that ELB load balancers require TLS termination

Created by Priyanka Chaudhary (AWS)

| Environment: Production | Technologies: Networking; Security, identity, compliance | AWS services: Amazon CloudWatch Events; Elastic Load Balancing; AWS Lambda |

Summary

On the Amazon Web Services (AWS) Cloud, Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, IP addresses, and AWS Lambda functions. The load balancers use listeners to define the ports and protocols that the load balancer uses to accept traffic from users. Application Load Balancers make routing decisions at the application layer and use the HTTP/HTTPS protocols. Classic Load Balancers make routing decisions at either the transport layer, by using TCP or Secure Sockets Layer (SSL) protocols, or at the application layer, by using HTTP/HTTPS.

This pattern provides a security control that examines multiple event types for Application Load Balancers and Classic Load Balancers. When the function is invoked, AWS Lambda inspects the event and ensures that the load balancer is compliant.

The function initiates an Amazon CloudWatch Events event on the following API calls: `CreateLoadBalancer`, `CreateLoadBalancerListeners`, `DeleteLoadBalancerListeners`, `CreateLoadBalancerPolicy`, `SetLoadBalancerPoliciesOfListener`, `CreateListener`, `DeleteListener`, and `ModifyListener`. When the event detects one of these APIs, it calls AWS Lambda, which runs a Python script. The Python script evaluates to see if the listener contains an SSL certificate, and if the policy that is applied is using Transport Layer Security (TLS). If the SSL policy is determined to be anything other than TLS, the function sends an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information.

Prerequisites and limitations

Prerequisites
• An active AWS account

Limitations
• This security control does not check for existing load balancers, unless an update is made to the load balancer listeners.
• This security control is regional. You must deploy it in each AWS Region you want to monitor.

Architecture

Target architecture

Automation and scale
• If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services
• AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually.
• Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
• AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers.
• Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
• Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code
This pattern includes the following attachments:

- ELBRequirestlstermination.zip – The Lambda code for the security control.
- ELBRequirestlstermination.yml – The CloudFormation template that sets up the event and Lambda function.

Epics

Set up the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be in the same AWS Region as the load balancer that you want to evaluate. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Upload the Lambda code.</td>
<td>Upload the Lambda code (ELBRequirestlstermination.zip file) that's provided in the Attachments section to the S3 bucket.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Deploy the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the AWS CloudFormation template.</td>
<td>Open the AWS CloudFormation console in the same AWS Region as your S3 bucket and deploy the attached template ELBRequirestlstermination.yml. For more information about deploying AWS CloudFormation templates, see Creating a stack on the AWS CloudFormation console in the CloudFormation documentation.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
| Complete the parameters in the template. | When you launch the template, you'll be prompted for the following information:  
  - **S3 bucket:** Specify the bucket that you created or selected in the first epic. This is where you uploaded the Lambda code. | Cloud architect |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the attached Lambda code (ELBRequireInstTermination.zip file).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>S3 key:</strong> Specify the location of the Lambda .zip file in your S3 bucket (for example, ELBRequireInstTermination.zip or controls/ELBRequireInstTermination.zip). Do not include leading slashes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Notification email:</strong> Provide an active email address where you want to receive Amazon SNS notifications.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Lambda logging level:</strong> Specify the logging level and frequency for the Lambda function. Use <strong>Info</strong> to log detailed informational messages on progress, <strong>Error</strong> for error events that would still allow the deployment to continue, and <strong>Warning</strong> for potentially harmful situations.</td>
<td></td>
</tr>
</tbody>
</table>

**Confirm the subscription**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Confirm the subscription.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email to the email address you provided. You must confirm this email subscription to start receiving violation notifications.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- What is AWS Lambda? (AWS Lambda documentation)
- What is a Classic Load Balancer? (ELB documentation)
- What is an Application Load Balancer? (ELB documentation)

**Attachments**

attachment.zip
Automate the setup of inter-Region peering with AWS Transit Gateway

*Created by Ram Kandaswamy (AWS)*

| Environment: Production | Technologies: Networking; Hybrid cloud | AWS services: AWS Transit Gateway; AWS Step Functions; AWS Lambda |

**Summary**

AWS Transit Gateway connects virtual private clouds (VPCs) and on-premises networks through a central hub. Transit Gateway traffic always stays on the global Amazon Web Services (AWS) backbone and doesn’t traverse the public internet, which reduces threat vectors, such as common exploits and distributed denial of service (DDoS) attacks.

If you need to communicate between two or more AWS Regions, you can use inter-Region Transit Gateway peering to establish peering connections between transit gateways in different Regions. However, manually configuring inter-Region peering with Transit Gateway can be a time-consuming process that has multiple steps. This pattern provides an automated process to remove these manual steps by using code to perform the peering. You can use this approach if you have to repeatedly configure several Regions and AWS accounts during a multi-Region organization setup.

This pattern uses an AWS CloudFormation stack that includes the AWS Step Functions workflow, AWS Lambda functions, AWS Identity and Access Management (IAM) roles, and log groups in Amazon CloudWatch Logs. You can then start a Step Functions execution and create the inter-Region peering connection for your transit gateways.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- An existing Amazon Simple Storage Service (Amazon S3) bucket.
- Transit gateways, created and configured in the requestor Region and the acceptor Regions. The *requester* Region is where a peering request is originated and the *acceptor* Regions accept the peering request. For more information about this, see [Creating and accepting a VPC peering connection](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-peering-req-accept.html) in the Amazon VPC documentation.
- VPCs, installed and configured in the acceptor and requester Regions. For steps to create a VPC, see [Create the VPC](https://docs.aws.amazon.com/vpc/latest/userguide/get-started-with-amazon-vpc.html) from Get Started with Amazon VPC in the Amazon VPC documentation.
- The VPCs must use the `addToTransitGateway` tag and `true` value.
- Security groups and network access control lists (ACLs) for your VPCs, configured according to your requirements. For more information about this, see [Security groups for your VPC](https://docs.aws.amazon.com/vpc/latest/userguide/security-groups-vpc.html) and [Network ACLs](https://docs.aws.amazon.com/vpc/latest/userguide/what-is-network-acl.html) in the Amazon VPC documentation.

**AWS Regions and limitations**

- Only certain AWS Regions support inter-Region peering. For a full list of Regions that support inter-Region peering, see the [AWS Transit Gateway FAQs](https://docs.aws.amazon.com/itmgateway/latest/itmgateway_concepts/faq.html).
In the attached sample code, the requestor Region is assumed to be `us-east-2`, and the acceptor Region is assumed to be `us-west-2`. If you want to configure different Regions, you must edit these values in all Python files. To implement a more complex setup that involves more than two Regions, you can change the Step Function to pass the Regions as a parameter to the Lambda function and run the function for each combination.

**Architecture**

The diagram shows a workflow with the following steps:

1. The user creates an AWS CloudFormation stack.
2. AWS CloudFormation creates a Step Functions state machine that uses a Lambda function. For more information about this, see Creating a Step Functions state machine that uses Lambda in the AWS Step Functions documentation.
3. Step Functions calls a Lambda function for peering.
4. The Lambda function creates a peering connection between transit gateways.
5. Step Functions calls a Lambda function for route table modifications.
6. The Lambda function modifies the route tables by adding the Classless Inter-Domain Routing (CIDR) block of the VPCs.

**Step Functions workflow**
The diagram shows the following Step Functions workflow:

1. The Step Functions workflow calls the Lambda function for the transit gateway peering.
2. There is a timer call to wait for one minute.
3. The peering status is retrieved and sent to the condition block. The block is responsible for the looping.
4. If the success condition is not met, the workflow is coded to enter the timer stage.
5. If the success condition is met, a Lambda function is called to modify the route tables. After this call, the Step Functions workflow ends.

**Tools**

- **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your AWS resources.
- **Amazon CloudWatch Logs** – CloudWatch Logs helps you centralize the logs from all of your systems, applications, and AWS services that you use.
- **AWS Identity and Access Management (IAM)** – IAM is a web service for securely controlling access to AWS services.
- **AWS Lambda** – Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources.
- **AWS Step Functions** – Step Functions makes it easy to coordinate the components of distributed applications as a series of steps in a visual workflow.
# Epics

## Automate peering

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the attached files to your S3 bucket.</td>
<td>Sign in to the AWS Management Console, open the Amazon S3 console, and then upload the modify-transit-gateway-routes.zip, peer-transit-gateway.zip, and get-transit-gateway-peering-status.zip files (attached) to your S3 bucket.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
| Create the AWS CloudFormation stack. | Run the following command to create an AWS CloudFormation stack using the transit-gateway-peering.json file (attached):  
```bash  
aws cloudformation create-stack --stack-name myteststack --template-body file://sampletemplate.json  
```  
The AWS CloudFormation stack creates the Step Functions workflow, the Lambda functions, IAM roles, and CloudWatch log groups.  
Make sure that the AWS CloudFormation template refers to the S3 bucket that contains the files that you uploaded earlier.  
**Note:** You can also create a stack by using the AWS CloudFormation console. For more information about this, see [Creating a stack on the AWS CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-create-stack-quickstart.html) in the AWS CloudFormation documentation. | DevOps engineer |
| Start a new execution in Step Functions. | Open the Step Functions console and start a new execution. Step Functions calls the Lambda function and creates the peering connection for the transit gateways. You don't need an input JSON file. Verify that an | DevOps engineer, General AWS |
attachment is available and that the connection type is **Peering**.

For more information about this, see [Start a new execution from Getting started with AWS Step Functions in the AWS Steps Functions documentation](#).

**Related resources**

- Executions in Step Functions
- Transit gateway peering attachments
- Interconnecting VPCs across AWS Regions using AWS Transit Gateway - Demo (video)

**Attachments**

attachment.zip

---

**Centralize network connectivity using AWS Transit Gateway**

*Created by Mydhili Palagummi (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Networking</th>
<th>AWS services: AWS Transit Gateway; Amazon VPC</th>
</tr>
</thead>
</table>

1518
Summary

This pattern describes the simplest configuration in which AWS Transit Gateway can be used to connect an on-premises network to virtual private clouds (VPCs) in multiple AWS accounts within an AWS Region. Using this setup, you can establish a hybrid network that connects multiple VPC networks in a Region and an on-premises network. This is accomplished by using a transit gateway and a virtual private network (VPN) connection to the on-premises network.

Prerequisites and limitations

Prerequisites

- AWS accounts and AWS Organizations set up
- VPCs in multiple AWS accounts, without overlapping Classless Inter-Domain Routing (CIDR) blocks

Limitations

This pattern does not support the isolation of traffic between certain VPCs or the on-premises network. All the networks attached to the transit gateway will be able to reach each other. To isolate traffic, you need to use custom route tables on the transit gateway. This pattern only connects the VPCs and on-premises network by using a single default transit gateway route table, which is the simplest configuration.

Architecture

Target technology stack

- AWS Transit Gateway
- AWS Site-to-Site VPN
- VPC
- AWS Resource Access Manager (AWS RAM)

Target architecture

![Target architecture diagram](image)

Tools

Tools
• **AWS Transit Gateway** – AWS Transit Gateway acts as a cloud router and simplifies network architecture by allowing you to connect Amazon VPCs in multiple AWS accounts and on-premises network to a single gateway.

• **AWS Resource Access Manager** – AWS Resource Access Manager (AWS RAM) supports the secure sharing of AWS resources with your AWS accounts, organizational units, or your entire organization from AWS Organizations.

## Epics

### Create a transit gateway in a network account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS account specifically for network services in your organization.</td>
<td>This account in AWS Organizations serves as the account for hosting network services. A central transit gateway that connects all accounts will be deployed in this account.</td>
<td>System Admin</td>
</tr>
<tr>
<td>Create a transit gateway in the network service account.</td>
<td>A transit gateway is a regional resource and can connect VPCs that are in different AWS accounts within the same AWS Region. Launch a transit gateway resource in the desired Region. Turn on default route table association and propagation to achieve the configuration described in this pattern. For help with this and with procedures in other epics, see the &quot;Related resources&quot; section.</td>
<td>Network Admin</td>
</tr>
</tbody>
</table>

### Connect the transit gateway to your on-premises network

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up a customer gateway device for the VPN connection.</td>
<td>The customer gateway device is attached on the on-premises side of the Site-to-Site VPN connection between the transit gateway and your on-premises network. In this task, identify or launch a supported on-premises customer device and note its public IP address for step 2. VPN configuration on this device is done in step 3.</td>
<td>Network Admin</td>
</tr>
</tbody>
</table>
## Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the network services account, create a VPN attachment to the transit gateway.</td>
<td>To set up a connection, create a VPN attachment for the transit gateway.</td>
<td>Network Admin</td>
</tr>
<tr>
<td>Configure the VPN on the customer gateway device in your on-premises network.</td>
<td>Download the configuration file for the Site-to-Site VPN connection associated with the transit gateway and configure VPN settings on the customer gateway device.</td>
<td>Network Admin</td>
</tr>
</tbody>
</table>

### Share the transit gateway in the network services account to other AWS accounts or your organization

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the AWS Organizations management account, turn on sharing.</td>
<td>To share the transit gateway with your organization or with certain organizational units, turn on sharing in AWS Organizations. Otherwise, you would need to share the transit gateway for each account individually.</td>
<td>System Admin</td>
</tr>
<tr>
<td>Create the transit gateway resource share in the network services account.</td>
<td>To allow VPCs in other AWS accounts within your organization to connect to the transit gateway, in the network services account, use the AWS RAM console to share the transit gateway resource.</td>
<td>System Admin</td>
</tr>
</tbody>
</table>

### Connect VPCs to the transit gateway

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create VPC attachments in individual accounts.</td>
<td>In the accounts to which the transit gateway has been shared, create transit gateway VPC attachments.</td>
<td>Network Admin</td>
</tr>
<tr>
<td>Accept the VPC attachment requests.</td>
<td>In the network services account, accept the transit gateway VPC attachment requests.</td>
<td>Network Admin</td>
</tr>
</tbody>
</table>
## Configure routing

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure routes in individual VPCs.</td>
<td>In each individual account VPC, add routes to the on-premises network and to other VPC networks, using the transit gateway as the target.</td>
<td>Network Admin</td>
</tr>
<tr>
<td>Configure routes in the transit gateway route table.</td>
<td>Routes from VPCs and the VPN connection should be propagated and should appear in the transit gateway default route table. If needed, create any static routes (one example is static routes for the static VPN connection) in the transit gateway default route table.</td>
<td>Network Admin</td>
</tr>
<tr>
<td>Add security group and network access control list (ACL) rules.</td>
<td>For the EC2 instances and other resources in the VPC, ensure that the security group rules and the network ACL rules allow traffic between VPCs as well as the on-premises network.</td>
<td>Network Admin</td>
</tr>
</tbody>
</table>

## Test connectivity

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test connectivity between VPCs.</td>
<td>Ensure that network ACL and security groups allow Internet Control Message Protocol (ICMP) traffic, and then ping from instances in a VPC to another VPC that is also connected to the transit gateway.</td>
<td>Network Admin</td>
</tr>
<tr>
<td>Test connectivity between VPCs and the on-premises network.</td>
<td>Ensure that network ACL rules, security group rules, and any firewalls allow ICMP traffic, and then ping between the on-premises network and the EC2 instances in the VPCs. Network communication must be initiated from the on-premises network first to bring the VPN connection to UP status.</td>
<td>Network Admin</td>
</tr>
</tbody>
</table>

## Related resources

*Create a transit gateway*
Connect to AWS MGN data and control planes over a private network

- Create a transit gateway

*Connect the transit gateway to your on-premises network*
- Your customer gateway device
- Transit gateway VPN attachments
- AWS Site-to-Site VPN: Download the configuration file

*Share the transit gateway*
- Enable sharing with AWS Organizations
- AWS RAM: Create a resource share

*Connect VPCs*
- Create a transit gateway attachment to a VPC
- Accept a shared attachment

*Configure routes*
- Adding and removing routes from a route table
- Create a static route
- Security groups: Adding, removing, and updating rules
- Network ACLs: Adding and deleting rules

*Other resources*
- Building a scalable and secure multi VPC AWS Network Infrastructure (AWS whitepaper)
- Working with shared resources (AWS RAM documentation)
- Working with transit gateways (AWS Transit Gateway documentation)

**Summary**
This pattern explains how you can connect to an AWS Application Migration Service (AWS MGN) data plane and control plane on a private, secured network by using interface VPC endpoints.

AWS MGN is a highly automated lift-and-shift (rehost) solution that simplifies, expedites, and reduces the cost of migrating applications to AWS. It enables companies to rehost a large number of physical, virtual, or cloud servers without compatibility issues, performance disruption, or long cutover windows.
AWS MGN is available from the AWS Management Console. This enables seamless integration with other AWS services, such as AWS CloudTrail, Amazon CloudWatch, and AWS Identity and Access Management (IAM).

You can connect from a source data center to a data plane—that is, to a subnet that serves as a staging area for data replication in the destination VPC—over a private connection by using AWS VPN services, AWS Direct Connect, or VPC peering in AWS MGN. You can also use interface VPC endpoints powered by AWS PrivateLink to connect to an AWS MGN control plane over a private network.

**Prerequisites and limitations**

**Prerequisites**

- **Staging area subnet** – Before you set up AWS MGN, create a subnet to be used as a staging area for data replicated from your source servers to AWS (that is, a data plane). You must specify this subnet in the Replication Settings template when you first access the AWS MGN console. You can override this subnet for specific source servers in the Replication Settings template. Although you can use an existing subnet in your AWS account, we recommend that you create a new, dedicated subnet for this purpose.

- **Network requirements** – The replication servers that are launched by AWS MGN in your staging area subnet have to be able to send data to the AWS MGN API endpoint at https://mgn.<region>.amazonaws.com/, where <region> is the code for the AWS Region you are replicating to (for example, https://mgn.us-east-1.amazonaws.com). Amazon Simple Storage Service (Amazon S3) service URLs are required for downloading AWS MGN software.
  - The AWS Replication Agent installer should have access to the S3 bucket URL of the AWS Region you are using with AWS MGN.
  - The staging area subnet should have access to Amazon S3.
  - The source servers on which the AWS Replication Agent is installed must be able to send data to the replication servers in the staging area subnet and to the AWS MGN API endpoint at https://mgn.<region>.amazonaws.com/.

  
  
  
  The following table lists the required ports.

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Port</th>
<th>For more information, see</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source data center</td>
<td>Amazon S3 service URLs</td>
<td>443 (TCP)</td>
<td>Communication over TCP port 443</td>
</tr>
<tr>
<td>Source data center</td>
<td>AWS Region-specific console address for AWS MGN</td>
<td>443 (TCP)</td>
<td>Communication between the source servers and AWS MGN over TCP port 443</td>
</tr>
<tr>
<td>Source data center</td>
<td>Staging area subnet</td>
<td>1500 (TCP)</td>
<td>Communication between the source servers and the staging area subnet over TCP port 1500</td>
</tr>
<tr>
<td>Staging area subnet</td>
<td>AWS Region-specific console address for AWS MGN</td>
<td>443 (TCP)</td>
<td>Communication between the staging area subnet and the AWS MGN API endpoint over TCP port 1500</td>
</tr>
</tbody>
</table>
### Architecture

The following diagram illustrates the network architecture for a typical migration. For more information about this architecture, see the AWS MGN documentation and the AWS MGN service architecture and network architecture video.

The following detailed view shows the configuration of interface VPC endpoints in the staging area VPC to connect Amazon S3 and AWS MGN.

### Limitations

AWS MGN isn’t currently available in all AWS Regions and operating systems.

- **Supported AWS Regions**
- **Supported operating systems**
Tools

- **AWS MGN** – An AWS service that simplifies, expedites, and reduces the cost of rehosting applications on AWS.
- **Interface VPC endpoints** – Endpoints that enable you to connect to services that are powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect.
connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Epics

Create endpoints for AWS MGN, Amazon EC2, and Amazon S3

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure the interface endpoint for AWS MGN. | The source data center and staging area VPC connect privately to the AWS MGN control plane through the interface endpoint that you create in the target staging area VPC. To create the endpoint:  
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.  
2. In the navigation pane, choose Endpoints, Create Endpoint.  
3. For Service Name, enter com.amazonaws.<region>.mgn. For Type, choose Interface.  
4. For VPC, select a target staging area VPC to create the endpoint. For Subnets, select the subnets (Availability Zones) in which to create the endpoint network interfaces.  
5. To turn on private DNS for the interface endpoint, select Enable DNS Name.  
6. Select a security group that allows ingress from the staging area VPC subnet over TCP 443.  
7. Choose Create endpoint.  
For more information, see Interface VPC endpoints in the Amazon VPC documentation. | Migration lead |
| Configure the interface endpoint for Amazon EC2. | The staging area VPC connects privately to the Amazon EC2 API through the interface endpoint that you create in the target staging area VPC. To create the endpoint, follow the instructions provided in the previous story. | Migration lead |
### Configure the gateway endpoint for Amazon S3.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
</table>
| | • For service name, enter com.amazonaws.<region>.ec2. 
| | • The security group must allow inbound HTTPS traffic from the staging area VPC subnet over port 443. 
| | • Enable private DNS names. |

Configure the gateway endpoint for Amazon S3. The source data center and staging area VPC connect privately to the Amazon S3 API through the Interface endpoint that you create in the target staging area VPC. To create the endpoint:

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.  
2. In the navigation pane, choose **Endpoints**, **Create Endpoint**.  
3. For **Service Name**, enter com.amazonaws.<region>.s3. For **Type**, choose **Interface**.  

The VPC security group must allow inbound HTTPS traffic from the staging area VPC subnet over port 443.

**Notes:**

- Gateway endpoint connections cannot be extended out of a VPC. (For details, see the [Amazon VPC documentation](https://aws.amazon.com/documentation/vpc).)  
- Amazon S3 interface endpoints do not support private DNS names. (For details, see the [Amazon VPC documentation](https://aws.amazon.com/documentation/vpc).)

---

### Connect to the AWS MGN control plane over a private link

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install AWS Replication Agent by using AWS PrivateLink.</td>
<td>1. Download the AWS Replication Agent to a private</td>
<td>Migration engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1.</td>
<td>S3 bucket in the destination Region.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Log in to the source servers to be migrated. The AWS Replication Agent installer needs network access to the AWS MGN and Amazon S3 endpoints. Because your on-premises network isn’t open to AWS MGN and Amazon S3 public endpoints, you must install the Agent with the help of the interface endpoints you created in the previous steps by using AWS PrivateLink. For example, for Linux:</td>
<td></td>
</tr>
</tbody>
</table>

```bash
aws s3 --region <aws_region> --endpoint-url https://bucket.<s3-endpoint-DNS-name> cp s3://<bucket-name>/<source-path> <destination-path>
```

```bash
sudo python3 aws-replication-installer-init.py --region <aws_region> --aws-access-key-id <access-key> --aws-secret-access-key <secret-key> \ --no-prompt --s3-endpoint <s3-endpoint-DNS-name>
```

For more information, see [AWS Replication Agent installation instructions](#) in the AWS MGN documentation.

After you have established your connection with AWS MGN and installed the AWS Replication Agent, follow the instructions in the [AWS MGN documentation](#) to migrate your source servers to your target VPC and subnet.

### Related resources

AWS MGN documentation
Create Infoblox objects using AWS CloudFormation custom resources and Amazon SNS

Created by Tim Sutton (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Networking</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services:</strong></td>
<td>Amazon SNS; AWS CloudFormation; AWS KMS; AWS Lambda; AWS Organizations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

Infoblox Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and IP address management (Infoblox DDI) enables you to centralize and efficiently control a complex hybrid environment. With Infoblox DDI, you can discover and record all network assets in one authoritative IP address management (IPAM) database, in addition to managing DNS on premises and on the Amazon Web Services (AWS) Cloud by using the same appliances.

This pattern describes how to use an AWS CloudFormation custom resource to create Infoblox objects (for example, DNS records or IPAM objects) by calling the Infoblox WAPI API. For more information about the Infoblox WAPI, see the WAPI documentation in the Infoblox documentation.

By using this pattern's approach, you can obtain a unified view of DNS records and IPAM configurations for your AWS and on-premises environments, in addition to removing manual processes that create records and provision your networks. You can use this pattern's approach for the following use cases:

- Adding an A record after creating an Amazon Elastic Compute Cloud (Amazon EC2) instance
- Adding a CNAME record after creating an Application Load Balancer
- Adding a network object after creating a virtual private cloud (VPC)
- Providing the next network range and using that range to create subnets

You can also extend this pattern and use other Infoblox device features such as adding different DNS record types or configuring Infoblox vDiscovery.
The pattern uses a hub-and-spoke design in which the hub requires connectivity to the Infoblox appliance on the AWS Cloud or on premises and uses AWS Lambda to call the Infoblox API. The spoke is in the same or a different account in the same organization in AWS Organizations, and calls the Lambda function by using an AWS CloudFormation custom resource.

Prerequisites and limitations

Prerequisites

- An existing Infoblox appliance or grid, installed on the AWS Cloud, on premises, or both, and configured with an admin user that can administer IPAM and DNS actions. For more information about this, see About admin accounts in the Infoblox documentation.
- An existing DNS authoritative zone that you want to add records on the Infoblox appliance. For more information about this, see Configuring authoritative zones in the Infoblox documentation.
- Two active AWS accounts in AWS Organizations. One account is the hub account and the other account is the spoke account.
- The hub and spoke accounts must be in the same AWS Region.
- The hub account’s VPC must connect to the Infoblox appliance; for example, by using AWS Transit Gateway or VPC peering.
- AWS Serverless Application Model (AWS SAM), locally installed and configured with AWS Cloud9 or AWS CloudShell.
- The Infoblox-Hub.zip and ClientTest.yaml files (attached), downloaded to the local environment that contains AWS SAM.

Limitations

- The AWS CloudFormation custom resource's service token must be from the same Region where the stack is created. We recommend that you use a hub account in each Region, instead of creating an Amazon Simple Notification Service (Amazon SNS) topic in one Region and calling the Lambda function in another Region.

Product versions

- Infoblox WAPI version 2.7

Architecture

The following diagrams shows this pattern's workflow.
The diagram shows the following components for this pattern’s solution:

1. AWS CloudFormation custom resources enable you to write custom provisioning logic in templates that AWS CloudFormation runs when you create, update, or delete stacks. When you create a stack, AWS CloudFormation sends a create request to an SNS topic that’s monitored by an application running on an EC2 instance.

2. The Amazon SNS notification from the AWS CloudFormation custom resource is encrypted through a specific AWS Key Management Service (AWS KMS) key and access is restricted to accounts in your organization in Organizations. The SNS topic initiates the Lambda resource that calls the Infoblox WAPI API.

3. Amazon SNS invokes the following Lambda functions that take the Infoblox WAPI URL, the username, and password AWS Secrets Manager Amazon Resource Names (ARNs) as environment variables:
   - `dnsapi.lambda_handler` – Receives the `DNSName`, `DNSType`, and `DNSValue` values from the AWS CloudFormation custom resource and uses these to create DNS A records and CNAMEs.
   - `ipaddr.lambda_handler` – Receives the `VPCCIDR`, `Type`, `SubnetPrefix`, and `Network Name` values from the AWS CloudFormation custom resource and uses these to add the network data into the Infoblox IPAM database or provide the custom resource with the next available network that can be used to create new subnets.
   - `describeprefixes.lambda_handler` – Calls the `describe_managed_prefix_lists` AWS API by using the "com.amazonaws."+Region+".s3" filter to retrieve the required prefix ID.
   - **Important**: These Lambda functions are written in Python and are similar to each other but call different APIs.

4. You can deploy the Infoblox grid as physical, virtual, or cloud-based network appliances. It can be deployed on-premises or as a virtual appliance using a range of hypervisors, including VMware ESXi, Microsoft Hyper-V, Linux KVM, and Xen. You can also deploy the Infoblox grid on the AWS Cloud with an Amazon Machine Image (AMI).

5. The diagram shows a hybrid solution for the Infoblox grid that provides DNS and IPAM to resources on the AWS Cloud and on premises.

**Technology stack**
Tools

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS KMS** – AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud. AWS KMS keys and functionality are used by other AWS services, and you can use them to protect data in your own applications that use AWS.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

- **AWS SAM** – AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS. AWS SAM provides you with a template specification to define your serverless application, and a command line interface (CLI) tool.

- **AWS Secrets Manager** – AWS Secrets Manager is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Code

You can use the ClientTest.yaml sample AWS CloudFormation template (attached) to test the Infoblox hub. You can customize the AWS CloudFormation template to include the custom resources from the following table:
| Create an A record using the Infoblox spoke custom resource | Return values:  
infobloxref – Infoblox references  
Example resource:  
ARECORDCustomResource:  
  Type: "Custom::InfobloxAPI"  
  Properties:  
    ServiceToken: !Sub arn:aws:sns:${AWS::Region}:${HubAccountID}:RunInfobloxDNSFunction  
    DNSName: 'arecordtest.company.com'  
    DNSType: 'ARecord'  
    DNSValue: '10.0.0.1' |
| Create a CNAME record using the Infoblox spoke custom resource | Return values:  
infobloxref – Infoblox references  
Example resource:  
CNAMECustomResource:  
  Type: "Custom::InfobloxAPI"  
  Properties:  
    ServiceToken: !Sub arn:aws:sns:${AWS::Region}:${HubAccountID}:RunInfobloxDNSFunction  
    DNSName: 'cnametest.company.com'  
    DNSType: 'cname'  
    DNSValue: 'aws.amazon.com' |
| Create a network object using the Infoblox spoke custom resource | Return values:  
infobloxref – Infoblox references  
network – Network range (the same as VPCCIDR)  
Example resource:  
VPCCustomResource:  
  Type: 'Custom::InfobloxAPI'  
  Properties: |
Retrieve the next available subnet using the Infoblox spoke custom resource

Return values:
infobloxref – Infoblox references
network – The subnet's network range

Example resource:

<table>
<thead>
<tr>
<th>Subnet1CustomResource:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: 'Custom::InfobloxAPI'</td>
</tr>
<tr>
<td>DependsOn: VPCCustomResource</td>
</tr>
<tr>
<td>Properties:</td>
</tr>
<tr>
<td>ServiceToken: !Sub arn:aws:sns:${AWS::Region}: ${HubAccountID}:RunInfobloxNextSubnetFunction</td>
</tr>
<tr>
<td>VPCCIDR: !Ref VpcCIDR</td>
</tr>
<tr>
<td>Type: Subnet</td>
</tr>
<tr>
<td>SubnetPrefix: !Ref SubnetPrefix</td>
</tr>
<tr>
<td>NetworkName: My-Subnet</td>
</tr>
</tbody>
</table>

Epics

Create and configure the hub account's VPC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a VPC with a connection to the Infoblox appliance.</td>
<td>Sign in to the AWS Management Console for your hub account and create a VPC by following the steps in the Amazon VPC on the AWS Cloud Quick Start reference deployment from AWS Quick Starts. <strong>Important:</strong> The VPC must have HTTPS connectivity to the Infoblox appliance and we recommend that you</td>
<td>Network administrator, System administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| (Optional) Create the VPC endpoints for private subnets. | VPC endpoints provide connectivity to public services for your private subnets. The following endpoints are required:  
   - A gateway endpoint for Amazon Simple Storage Service (Amazon S3) to allow Lambda to communicate with AWS CloudFormation  
   - An interface endpoint for Secrets Manager to enable connectivity with Secrets Manager  
   - An interface endpoint for AWS KMS to allow the encryption of the SNS topic and Secrets Manager secret  
   For more information about creating endpoints for private subnets, see VPC endpoints in the Amazon VPC documentation. | Network administrator, Systems administrator |

**Deploy the Infoblox hub**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Build the AWS SAM template. | 1. Run the **unzip Infoblox-Hub.zip** command in the environment that contains AWS SAM.  
2. Run the **cd Hub/** command to change your directory to the **Hub** directory.  
3. Run the **sam build** command to process the AWS SAM template file, application code, and any language-specific files and dependencies. The **sam build** command also copies build artifacts in the format and location expected for the following story. | Developer, System administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS SAM template.</td>
<td>The <code>sam deploy</code> command takes the required parameters and saves them into the <code>samconfig.toml</code> file, stores the AWS CloudFormation template and Lambda functions in an S3 bucket, and then deploys the AWS CloudFormation template into your hub account.</td>
<td>Developer, System administrator</td>
</tr>
</tbody>
</table>

The following sample code shows how to deploy the AWS SAM template:

```
$ sam deploy --guided

Configuring SAM deploy
=======================
Looking for config file [samconfig.toml] :
Found
Reading default arguments : Success
Setting default arguments for 'sam deploy'

Stack Name [Infoblox-Hub]:
AWS Region [eu-west-1]:
Parameter InfobloxUsername:
Parameter InfobloxPassword:
Parameter InfobloxIPAddress [xxx.xxx.xx.xxx]:
Parameter AWSOrganisationID [o-xxxxxxxxxx]:
Parameter VPCID [vpc-xxxxxxxxxx]:
Parameter VPCCIDR [xxx.xxx.xxx.xxx/16]:
Parameter VPCSubnetID1 [subnet-xxx]:
Parameter VPCSubnetID2 [subnet-xxx]:
Parameter VPCSubnetID3 [subnet-xxx]:
Parameter VPCSubnetID4 []:
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy

Confirm changes before deploy [Y/n]: y
```
#SAM needs permission to be able to create roles to connect to the resources in your template
Allow SAM CLI IAM role creation [Y/n]: n
Capabilities [['CAPABILITY_NAMED_IAM']]:
Save arguments to configuration file [Y/n]: y
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]:

Important: You must use the --guided option each time because the Infoblox username and password are not stored in the samconfig.toml file

### Related resources
- Getting started with WAPIs using Postman (Infoblox Blog)
- Provisioning vNIOS for AWS Using the BYOL Model (Infoblox documentation)
- `quickstart-aws-vpc` (GitHub repo)
- `describe_managed_prefix_lists` (AWS SDK for Python documentation)

### Attachments
- attachment.zip

## Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS

*Created by Sachin Trivedi (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Source:</th>
<th>On-Premise</th>
<th>Target:</th>
<th>AWS Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Type:</td>
<td>Replatform</td>
<td>Workload:</td>
<td>All other workloads</td>
<td>Technologies:</td>
<td>Networking; Hybrid cloud; Migration</td>
</tr>
<tr>
<td>AWS services:</td>
<td>Amazon CloudFront; Elastic Load Balancing; AWS Lambda</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

When you migrate an application that uses an F5 Load balancer to Amazon Web Services (AWS) and want to use an Application Load Balancer on AWS, migrating F5 rules for header modifications is a common problem. An Application Load Balancer doesn’t support header modifications, but you can use Amazon CloudFront as a content delivery network (CDN) and Lambda@Edge to modify headers.

This pattern describes the required integrations and provides sample code for header modification by using AWS CloudFront and Lambda@Edge.

Prerequisites and limitations

Prerequisites

- An on-premises application that uses an F5 load balancer with a configuration that replaces the HTTP header value by using if, else. For more information about this configuration, see HTTP::header in the F5 product documentation.

Limitations

- This pattern applies to F5 load balancer header customization. For other third-party load balancers, please check the load balancer documentation for support information.
- The Lambda functions that you use for Lambda@Edge must be in the US East (N. Virginia) Region.

Architecture

The following diagram shows the architecture on AWS, including the integration flow between the CDN and other AWS components.

Tools

AWS services

- Application Load Balancer — An Application Load Balancer is an AWS fully managed load balancing service that functions at the seventh layer of the Open Systems Interconnection (OSI) model. It balances traffic across multiple targets and supports advanced routing requests based on HTTP headers and methods, query strings, and host-based or path-based routing.
• **Amazon CloudFront** – Amazon CloudFront is a web service that speeds up the distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations for lower latency and improved performance.

• **Lambda@Edge** – Lambda@Edge is an extension of AWS Lambda that lets you run functions to customize the content that CloudFront delivers. You can author functions in the US East (N. Virginia) Region, and then associate the function with a CloudFront distribution to automatically replicate your code around the world, without provisioning or managing servers. This reduces latency and improves the user experience.

**Code**

The following sample code provides a blueprint for modifying CloudFront response headers. Follow the instructions in the *Epics* section to deploy the code.

```javascript
exports.handler = async (event, context) => {
  const response = event.Records[0].cf.response;
  const headers = response.headers;

  const headerNameSrc = 'content-security-policy';
  const headerNameValue = '*.xyz.com';

  if (headers[headerNameSrc.toLowerCase()]) {
    headers[headerNameSrc.toLowerCase()] = [{
      key: headerNameSrc,
      value: headerNameValue,
    }];
    console.log(`Response header "${headerNameSrc}" was set to ` + `"${headers[headerNameSrc.toLowerCase()][0].value}"
    `);
  } else {
    headers[headerNameSrc.toLowerCase()] = [{
      key: headerNameSrc,
      value: headerNameValue,
    }];
  }
  return response;
};
```

**Epics**

Create a CDN distribution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudFront web distribution.</td>
<td>In this step, you create a CloudFront distribution to tell CloudFront where you want content to be delivered from, and the details about how to track and manage content delivery. To create a distribution by using the console, sign in to</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td></td>
<td>the AWS Management Console, open the CloudFront console, and then follow the steps in the CloudFront documentation.</td>
<td></td>
</tr>
</tbody>
</table>

### Create and deploy the Lambda@Edge function

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>You can create a Lambda@Edge function by using a blueprint for modifying CloudFront response headers. (Other blueprints are available for different use cases; for more information, see Lambda@Edge example functions in the CloudFront documentation.)</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>To create a Lambda@Edge function:</td>
<td></td>
</tr>
</tbody>
</table>
|      | 1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.  
2. Make sure that you're in the US East (N. Virginia) Region. CloudFront blueprints are available only in this Region.  
3. Choose Create function.  
4. Choose Use a blueprint, and then enter cloudfront in the Blueprints search field.  
5. Choose the cloudfront-modify-response-header blueprint, and then choose Configure.  
6. On the Basic information page, enter the following information:  
   a. Enter a function name.  
   b. For Execution role, choose Create a new role from AWS policy templates.  
   c. Associate the required AWS Identity and Access Management (IAM) role name.  
7. Choose Create function. |                              |
### AWS Prescriptive Guidance Patterns

#### Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>8. In the <strong>Designer</strong> section of the page, choose your function name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. In the <strong>Function code</strong> section, replace the template code with the sample code provided previously in this pattern, in the <strong>Code</strong> section.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. In the sample code, replace <code>xyz.com</code> with your domain name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Choose <strong>Save</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy the Lambda@Edge function.</td>
<td>Follow the instructions in <a href="https://aws.amazon.com/getting-started/hands-on/tutorial-creating-a-simple-lambdaedge-function/">step 4 of the Tutorial: Creating a simple Lambda@Edge function</a> in the Amazon CloudFront documentation to configure the CloudFront trigger and deploy the function.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>

### Related resources

**CloudFront documentation**

- Request and response behavior for custom origins
- Working with distributions
- Lambda@Edge example functions
- Customizing at the edge with Lambda@Edge
- Tutorial: Creating a simple Lambda@Edge function

### Tag Transit Gateway attachments automatically using AWS Organizations

*Created by Richard Milner-Watts (AWS), Haris Bin Ayub (AWS), and John Capps (AWS)*

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: Production</th>
<th>Technologies: Networking; Infrastructure; Management &amp; governance; Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transit Gateway Attachment Tagger</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AWS services:** AWS Step Functions; AWS Transit Gateway; Amazon VPC; AWS Lambda

**Environment:** Production

**Technologies:** Networking; Infrastructure; Management & governance; Operations

1542
Summary

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

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

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

Prerequisites and limitations

Prerequisites

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

Architecture

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

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

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

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

**Target technology stack**

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

**Target architecture**

The solution architecture and workflow are shown in the following diagram.
1. The scheduled event initiates the rule.
2. The EventBridge rule starts the Step Functions state machine.
3. The state machine invokes the LambdaOrganizationsAccountQuery Lambda function.
4. The LambdaOrganizationsAccountQuery Lambda function assumes the role in the organization management account.
5. The LambdaOrganizationsAccountQuery Lambda function calls the Organizations API to return AWS account metadata.
6. The state machine invokes the LambdaTGWAttachmentQuery Lambda function.
7. For each Region, in parallel, the state machine invokes LambdaTGWRTBQuery Lambda function.
8. For each Region, in parallel, the state machine invokes LambdaTGWTag Lambda function.
9. Name tags are created for Transit Gateway attachments in the Shared Networking account.
Automation and scale

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

Tools

AWS services

- **AWS CloudFormation** – AWS CloudFormation provides a way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code.

- **Amazon EventBridge** – Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge receives an event, an indicator of a change in environment, and applies a rule to route the event to a target. Rules match events to targets based on either the structure of the event, called an event pattern, or on a schedule.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests each day to thousands each second. You pay only for the compute time that you consume. There is no charge when your code is not running.

- **AWS Organizations** – AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. Using AWS Organizations, you can programatically create new AWS accounts and allocate resources, group accounts to organize your workflows, apply policies to accounts or groups for governance, and simplify billing by using a single payment method for all of your accounts.

- **AWS Step Functions** – AWS Step Functions is a low-code visual workflow service used to orchestrate AWS services, automate business processes, and build serverless applications. Workflows manage failures, retries, parallelization, service integrations, and observability so developers can focus on higher-value business logic.

- **AWS Transit Gateway** – AWS Transit Gateway connects VPCs and on-premises networks through a central hub. This simplifies your network and puts an end to complex peering relationships. It acts as a cloud router, so that each new connection is made only one time.

- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) is a service for launching AWS resources in a logically isolated virtual network that you define.

Code

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

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

Epics

Deploy the main solution stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather required prerequisite</td>
<td>To configure cross-account access from the Lambda function to the AWS</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Organizations API, you need the account ID for the organization's management account. <strong>Note:</strong> The order in which the two CloudFormation stacks are created matters. You must deploy resources into the Shared Networking account first. The role in the Shared Networking account must already exist before deploying resources into the organization's management account. For more information, see the AWS documentation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Launch the CloudFormation template for the main solution stack.** | The template for the main solution stack will deploy the IAM roles, Step Functions workflow, Lambda functions, and the CloudWatch event. Open the AWS Management console for the Shared Networking account, and then open the CloudFormation console. Create the stack by using the `tgw-attachment-tagger-main-stack.yaml` template and the following values:  
  * **Stack name** – `tgw-attachment-tagger-main-stack`  
  * **awsOrganizationsRootAccountId** – Account ID for the organization's management account  
  * **TGWRegions** parameter – AWS Regions for the solution, entered as a comma-delimited string  
  * **TGWLList** parameter – Transit gateway IDs to be excluded from the solution, entered in a comma-delimited string | DevOps engineer |
|  | For more information about launching a CloudFormation stack, see the AWS documentation. |  |
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Verify that the solution has launched successfully.</strong></td>
<td>Wait for the CloudFormation stack to reach a status of <strong>CREATE_COMPLETE</strong>. This should take less than one minute. Open the Step Functions console, and verify that a new state machine has been created with the name <code>tgw-attachment-tagger-state-machine</code>.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

### Deploy the AWS Organizations stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gather required prerequisite information.</strong></td>
<td>To configure cross-account access from the Lambda function to the AWS Organizations API, you need the account ID for the Shared Networking account.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
| **Launch the CloudFormation template for the Organizations stack** | The template for the AWS Organizations stack will deploy the IAM role in the organization's management account. Access the AWS console for the organization's management account and then open the CloudFormation console. Create the stack by using the `tgw-attachment-tagger-organizations-stack.yaml` template and the following values:  
  - **Stack name** – `tgw-attachment-tagger-organizations-stack`  
  - **NetworkingAccountId** parameter – Account ID for the Shared Networking account  
  
  For the other stack creation options, use the defaults. | DevOps engineer |
| **Verify that the solution has launched successfully.** | Wait for the CloudFormation stack to reach a status of **CREATE_COMPLETE**. This should take less than one minute. | DevOps engineer |
### Task 1: Open the Identity and Access Management (IAM) console, and verify that a new role has been created with the name `tgw-attachment-tagger-organization-query-role`.

#### Skills required:
- DevOps engineer

### Verify the solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the state machine.</td>
<td>Open the Step Functions console for the Shared Networking account, and choose <strong>State machines</strong> in the navigation pane. Select the state machine <code>tgw-attachment-tagger-state-machine</code>, and choose <strong>Start Execution</strong>. Because the input to this state machine is not used by the solution, you can use the default value. {&quot;Comment&quot;: &quot;Insert your JSON here&quot;} Choose <strong>Start Execution</strong>.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Watch the state machine until completion.</td>
<td>On the new page that opens, you can watch the state machine run. The duration will depend on the number of Transit Gateway attachments to process. On this page, you can examine each step of the state machine. You can view the various tasks within the state machine and follow links to the CloudWatch logs for the Lambda functions. For the tasks that run in parallel within the map, you can use the <strong>Index</strong> dropdown list to view the specific implementations for each Region.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Verify the Transit Gateway attachment tags.</td>
<td>Open the VPC console for the Shared Networking account,</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>and choose Transit Gateway Attachments. On the console, a Name tag is provided for attachments that met the criteria (the attachment is propagated to a Transit Gateway route table, and the resource owner is a member of the organization).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify the CloudWatch event initiation.</td>
<td>Wait for the CloudWatch event to initiate. This is scheduled for 06:00 UTC. Then open the Step Functions console for the Shared Networking account, and choose State machines in the navigation pane. Select the state machine tgw-attachment-tagger-state-machine. Verify that the solution ran at 06:00 UTC.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

### Related resources
- AWS Organizations
- AWS Resource Access Manager
- Serverless Transit Network Orchestrator
- Creating a role to delegate permissions to an IAM user
- Creating a stack on the AWS CloudFormation console

### More patterns
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer (p. 128)
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer (p. 117)
- Check for single-host network entries in security group ingress rules for IPv4 and IPv6 (p. 1651)
- Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud (p. 1007)
- Send alerts from AWS Network Firewall to a Slack channel (p. 1712)
- Use Network Firewall to capture the DNS domain names from the Server Name Indication (SNI) for outbound traffic (p. 1734)
Operating systems

Topics
- Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud (p. 1551)
- More patterns (p. 1553)

Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud

Created by Premkumar Chelladurai (AWS)

Environment: Production
Technologies: Operating systems; Migration
Workload: Microsoft

AWS services: Amazon EC2

Summary

After you migrate Microsoft SQL Server running on Windows Server 2008 R2, 2012, or 2012 R2 to Amazon Elastic Compute Cloud (Amazon EC2) instances on the Amazon Web Services (AWS) Cloud, the connection to SQL Server fails and the following errors appear:

- [Microsoft][ODBC SQL Server Driver][DBNETLIB] General Network error
- ERROR [08S01] [Microsoft][SQL Native Client]Communication link failure. System.Data.SqlClient.SqlException: A transport-level error has occurred when sending the request to the server. (provider: TCP Provider, error: 0 - An existing connection was forcibly closed by the remote host.)
- TCP Provider: The semaphore timeout period has expired

This pattern describes how you can resolve these errors by turning off the Windows Scalable Networking Pack (SNP) features at the operating system (OS) and network interface level for SQL Server running on Windows Server 2008 R2, 2012, or 2012 R2.

Prerequisites and limitations

Prerequisites
- Administrator privileges for Windows Server.
- If you used AWS Application Migration Service as your migration tool, you require one of the following Windows Server versions:
  - Windows Server 2008 R2 Service Pack 1, 2012, or 2012 R2
If you used CloudEndure as your migration tool, you require one of the following Windows Server versions:

Tools

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- Windows Server – Windows Server is a platform for building an infrastructure of connected applications, networks, and web services.

Epics

Turn off SNP features at the OS and elastic network interface levels

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Turn off SNP features at the OS level. | 1. Sign in to Windows Server and open a command prompt as an administrator.  
2. Run the `netsh int tcp show global command`.  
3. In the output, check if either Receive-Side Scaling or Chimney Offload is in enabled mode. If either of them is enabled, run the following commands:  
   - `netsh int tcp set global chimney=disabled`  
   - `netsh int tcp set global rss=disabled` | AWS administrator, AWS systems administrator, Migration engineer, Cloud administrator |
| Turn off SNP features at the elastic network interface level. | 1. Choose Start, enter `ncpa.cpl`, and then press Enter.  
2. Right-click Elastic Network Adapter.  
3. In the popup menu, choose Properties.  
4. In the Ethernet Adapter Properties window, choose Configure.  
5. In the Amazon Elastic Network Adapter Properties popup window, choose the Advanced tab.  
6. In the Property section, turn off all offloads and RSS. | AWS administrator, Cloud administrator, AWS systems administrator |
Related resources

- How to troubleshoot advanced network performance features such as RSS and NetDMA

More patterns

- Back up Sun SPARC servers in the Stromasys Charon-SSP emulator on the AWS Cloud (p. 1785)
- Migrate an on-premises Linux server to an Amazon EC2 Linux instance using AWS SMS (p. 991)
**Operations**

**Topics**
- Automatically create an RFC in AMS using Python (p. 1554)
- Create an AWS Cloud9 IDE that uses Amazon EBS volumes with default encryption (p. 1558)
- More patterns (p. 1561)

Automatically create an RFC in AMS using Python

*Created by Gnanasekaran Kailasam (AWS)*

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Operations; Cloud-native</td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS Managed Services</td>
</tr>
</tbody>
</table>

**Summary**

AWS Managed Services (AMS) helps you to operate your cloud-based infrastructure more efficiently and securely by providing ongoing management of your Amazon Web Services (AWS) infrastructure. To make a change to your managed environment, you need to create and submit a new request for change (RFC) that includes a change type (CT) ID for a particular operation or action.

However, manually creating an RFC can take around five minutes and teams in your organization might need to submit multiple RFCs every day. This pattern helps you to automate the RFC creation process, reduce the creation time for each RFC, and eliminate manual errors.

This pattern describes how to use Python code to automatically create the Stop EC2 instance RFC that stops Amazon Elastic Compute Cloud (Amazon EC2) instances in your AMS account. You can then apply this pattern’s approach and the Python automation to other RFC types.

**Prerequisites and limitations**

**Prerequisites**
- An AMS Advanced account. For more information about this, see AMS operations plans in the AWS Managed Services documentation.
- At least one existing EC2 instance in your AMS account.
- An understanding of how to create and submit RFCs in AMS.
- Familiarity with Python.

**Limitations**
- You can only use RFCs for changes in your AMS account. Your AWS account uses different processes for similar changes.

**Architecture**

**Technology stack**
Automation and scale

This pattern provides sample code to automate the Stop EC2 instance RFC, but you can use this pattern's sample code and approach for other RFCs.

Tools

- **AWS Managed Services** – AMS helps you to operate your AWS infrastructure more efficiently and securely.
- **AWS CLI** – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. In AMS, the change management API provides operations to create and manage RFCs.
- **AWS SDK for Python (Boto3)** – SDK for Python makes it easy to integrate your Python application, library, or script with AWS services.

Code

The AMS Stop EC2 Instance.zip file (attached) contains the Python code for creating a Stop EC2 instance RFC. You can also configure this code to submit a single RFC for multiple EC2 instances.

Epics

Option 1 – Set up environment for macOS or Linux

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Install and validate Python. | 1. Open a terminal window and run the brew install python3 command.  
2. Validate that Python is correctly installed by running the python --version command.  
3. Validate that pip is correctly installed by running the pip --version command. | AWS systems administrator |
| Install AWS CLI.       | Run the pip install awscli --upgrade --user command to install AWS CLI. | AWS systems administrator |
| Install Boto3.         | Run the pip install boto3 command to install Boto3.                           | AWS systems administrator |
| Install JSON.          | Run the pip install json command to install JSON.                             | AWS systems administrator |
## AWS Prescriptive Guidance Patterns

### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up AMS CLI.</td>
<td>Sign in to the AWS Management Console, open the AMS console, and then choose Documentation. Download the .zip file that contains the AMS CLI, unzip it, and then install it on your local machine. After you install AMS CLI, run the <code>aws amscm help</code> command. The output provides information about the AMS change management process.</td>
<td>AWS systems administrator</td>
</tr>
</tbody>
</table>

### Option 2 – Set up environment for Windows

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Install and validate Python.| 1. Open the [Python releases for Windows](https://www.python.org/downloads/windows/) page, download the latest version, and then install Python.  
2. Validate that Python is correctly installed by running the `python --version` command.  
3. Validate that `pip` is correctly installed by running the `pip --version` command. | AWS systems administrator       |
| Install AWS CLI.            | Run the `pip install awscli --upgrade --user` command to install AWS CLI.                                                                                                                                 | AWS systems administrator       |
| Install Boto3.              | Run the `pip install boto3` command to install Boto3.                                                                                                                                                        | AWS systems administrator       |
| Install JSON.               | Run the `pip install json` command to install JSON.                                                                                                                                                         | AWS systems administrator       |
| Set up AMS CLI.             | Sign in to the AWS Management Console, open the AMS console, and then choose Documentation. Download the .zip file that contains the AMS CLI, unzip it, and then install it on your local machine. After you install AMS CLI, run the `aws amscm help` command. The output provides information about the AMS change management process. | AWS systems administrator       |
### Extract the CT ID and execution parameters for the RFC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract the CT ID, version, and execution parameters for the RFC.</td>
<td>Each RFC has a different CT ID, version, and execution parameters. You can extract this information by using one of the following options: &lt;br&gt;1. Follow the instructions from the <em>Finding a request for change (RFC) with the CLI</em> section in the <em>RFC use examples</em> from the AWS Managed Services documentation. &lt;br&gt;2. Open an existing RFC of a similar type or create new RFC as a test through the AMS console. Use the RFC's CT ID and execution parameters. For more information about this, see <em>Finding an RFC with the console</em> in the AWS Managed Services documentation.</td>
<td>AWS systems administrator</td>
</tr>
</tbody>
</table>

**Note:** To adapt this pattern's Python automation for other RFCs, replace the CT type and parameter values in the *ams_stop_ec2_instance* Python code file from the *AMS Stop EC2 Instance.zip* file (attached) with those that you extracted.

### Run the Python automation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the Python automation.</td>
<td>1. Download the <em>AMS Stop EC2 Instance.zip</em> file (attached) to your local machine and extract the file. &lt;br&gt;2. Update <code>input_instances</code> with your EC2 instance information. &lt;br&gt;3. Open a terminal and navigate to the path for your extracted code &lt;br&gt;4. Run the <code>pythonams_stop_ec2_instance.py</code> command.</td>
<td>AWS systems administrator</td>
</tr>
</tbody>
</table>
Create an AWS Cloud9 IDE that uses Amazon EBS volumes with default encryption

Created by Janardhan Malyala (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Operations</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>AWS Cloud9; AWS KMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

You can use encryption by default to enforce the encryption of your Amazon Elastic Block Store (Amazon EBS) volumes and snapshot copies on the Amazon Web Services (AWS) Cloud.

You can create an AWS Cloud9 integrated development environment (IDE) that uses EBS volumes encrypted by default. However, the AWS Identity and Access Management (IAM) service-linked role for AWS Cloud9 requires access to the AWS Key Management Service (AWS KMS) key for these EBS volumes. If access is not provided, the AWS Cloud9 IDE might fail to launch and debugging might be difficult.

This pattern provides the steps to add the service-linked role for AWS Cloud9 to the AWS KMS key that is used by your EBS volumes. The setup described by this pattern helps you successfully create and launch an IDE that uses EBS volumes with encryption by default.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Default encryption turned on for EBS volumes. For more information about encryption by default, see Amazon EBS encryption in the Amazon Elastic Compute Cloud (Amazon EC2) documentation.
- An existing customer managed KMS key for encrypting your EBS volumes.
Architecture

Technology stack

- AWS Cloud9
- IAM
- AWS KMS

Tools

- AWS Cloud9 – AWS Cloud9 is a cloud-based IDE that you use to write, run, and debug code.
- Amazon EBS – Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances.
- AWS KMS – AWS Key Management Service (AWS KMS) is an encryption and key management service.
- AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources.
Epics

Find the default encryption key value

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record the default encryption key value for the EBS volumes.</td>
<td>Sign in to the AWS Management Console and open the Amazon EC2 console. Choose EC2 dashboard, and then choose EBS encryption in Account attributes. Copy and record the value in Default encryption key.</td>
<td>Cloud architect, DevOps engineer</td>
</tr>
</tbody>
</table>

Provide access to the AWS KMS key

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide AWS Cloud9 with access to the KMS key for EBS volumes.</td>
<td>Open the AWS KMS console, and then choose Customer managed keys. Choose the KMS key used for Amazon EBS encryption, and then choose the Key policy tab. Navigate to the Key users section, and then choose Add. Search for AWSServiceRoleForAWSCloud9. This provides access for the AWS Cloud9 service-linked role. Choose Add. <strong>Important:</strong> The service-linked role for AWS Cloud9 is automatically created when you launch your first IDE. For more information, see Creating a service-linked role in the AWS Cloud9 documentation.</td>
<td>Cloud architect, DevOps engineer</td>
</tr>
</tbody>
</table>

Create and launch the IDE

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and launch the AWS Cloud9 IDE.</td>
<td>Open the AWS Cloud9 console and choose Create environment. Configure your IDE according to your requirements by following the steps from Creating an EC2 environment in the AWS Cloud9 documentation.</td>
<td>Cloud architect, DevOps engineer</td>
</tr>
</tbody>
</table>
Related resources

- Encrypt EBS volumes used by AWS Cloud9
- Create a service-linked role for AWS Cloud9
- Create an EC2 environment in AWS Cloud9

More patterns

- Allow EC2 instances write access to S3 buckets in AMS accounts (p. 1768)
- Automate security scans for cross-account workloads using Amazon Inspector and AWS Security Hub (p. 1596)
- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config (p. 1601)
- Configure logging and monitoring for security events in your AWS IoT environment (p. 624)
- Create alarms for custom metrics using Amazon CloudWatch anomaly detection (p. 705)
- Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates (p. 1655)
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK (p. 713)
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account (p. 984)
- Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands (p. 223)
- Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function (p. 500)
- Monitor and remediate scheduled deletion of AWS KMS keys (p. 1682)
- Run AWS Systems Manager automation tasks synchronously from AWS Step Functions (p. 1751)
- Run event-driven and scheduled workloads at scale with AWS Fargate (p. 1475)
- Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization (p. 709)
- Tag Transit Gateway attachments automatically using AWS Organizations (p. 1542)
Security, identity, compliance

Topics

- Verify that new Amazon Redshift clusters have required SSL endpoints (p. 1563)
- Monitor ElastiCache clusters for security groups (p. 1566)
- Verify that new Amazon Redshift clusters launch in a VPC (p. 1570)
- Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools (p. 1574)
-Authenticate Microsoft SQL Server on Amazon EC2 using AWS Directory Service (p. 1579)
- Automate SAML 2.0 federation for AWS multi-account environments that use Azure AD (p. 1583)
- Automate remediation for AWS Security Hub standard findings (p. 1592)
- Automate security scans for cross-account workloads using Amazon Inspector and AWS Security Hub (p. 1596)
- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config (p. 1601)
- Automatically remediate unencrypted Amazon RDS DB instances and clusters (p. 1608)
- Automatically rotate IAM user access keys at scale with AWS Organizations and AWS Secrets Manager (p. 1616)
- Automatically validate and deploy IAM policies and roles in an AWS account by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros (p. 1621)
- Bidirectionally integrate AWS Security Hub with Jira software (p. 1627)
- Centralized logging and multiple-account security guardrails (p. 1638)
- Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version (p. 1647)
- Check for single-host network entries in security group ingress rules for IPv4 and IPv6 (p. 1651)
- Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates (p. 1655)
- Enable transparent data encryption in Amazon RDS for SQL Server (p. 1660)
- Ensure that AWS CloudFormation stacks are launched from authorized S3 buckets (p. 1664)
- Ensure AWS load balancers use secure listener protocols (HTTPS, SSL/TLS) (p. 1667)
- Ensure encryption for Amazon EMR data at rest is enabled at launch (p. 1671)
- Ensure that an IAM profile is associated with an EC2 instance (p. 1675)
- Ensure an Amazon Redshift cluster is encrypted upon creation (p. 1679)
- Monitor and remediate scheduled deletion of AWS KMS keys (p. 1682)
- Manage credentials using AWS Secrets Manager (p. 1687)
- Monitor Amazon EMR clusters for in-transit encryption at launch (p. 1692)
- Monitor Amazon ElastiCache clusters for at-rest encryption (p. 1696)
- Monitor EC2 instance key pairs using AWS Config (p. 1700)
- Send a notification when an IAM user is created (p. 1704)
- Restrict access to AWS APIs for AWS SSO and IAM users through trusted source IP ranges (p. 1708)
- Scan Git repositories for sensitive information and security issues by using git-secrets (p. 1710)
- Send alerts from AWS Network Firewall to a Slack channel (p. 1712)
- Simplify private certificate management by using ACM Private CA and AWS RAM (p. 1721)
Verify that new Amazon Redshift clusters have required SSL endpoints

Summary

This pattern provides an Amazon Web Services (AWS) CloudFormation template that automatically notifies you when a new Amazon Redshift cluster is launched without Secure Sockets Layer (SSL) endpoints.

Amazon Redshift is a fully managed, petabyte-scale, cloud-based data warehouse service. It is designed for large-scale dataset storage and analysis. It is also used to perform large-scale database migrations. For security, Amazon Redshift supports SSL to encrypt the connection between the user's SQL Server client application and the Amazon Redshift cluster. To configure your cluster to require an SSL connection, you set the `require_SSL` parameter to `true` in the parameter group that is associated with the cluster during launch.

The security control provided with this pattern monitors Amazon Redshift API calls in AWS CloudTrail logs and initiates an Amazon CloudWatch Events event for the CreateCluster, ModifyCluster, RestoreFromClusterSnapshot, CreateClusterParameterGroup, and ModifyClusterParameterGroup APIs. When the event detects one of these APIs, it calls AWS Lambda, which runs a Python script. The Python function analyzes the CloudWatch event for the listed CloudTrail events. When an Amazon Redshift cluster is created, modified, or restored from an existing snapshot, a new parameter group is created for the cluster, or an existed parameter group is modified, the function checks the `require_SSL` parameter for the cluster. If the parameter value is `false`, the function sends an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information: the Amazon Redshift cluster name, AWS Region, AWS account, and Amazon Resource Name (ARN) for Lambda that this notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A virtual private cloud (VPC) with a cluster subnet group, and an associated security group.

Created by Priyanka Chaudhary (AWS)

| Environment: Production | Technologies: Security, identity, compliance; Analytics; Data lakes | AWS services: AWS CloudTrail; Amazon CloudWatch Events; Amazon Redshift; Amazon SNS; AWS Lambda |

1563
Limitations

- This security control is regional. You must deploy it in each AWS Region you want to monitor.

Architecture

Target architecture

Automation and scale

- If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers.
- Amazon Redshift – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes the following attachments:

- RedshiftSSLEndpointsRequired.zip – The Lambda code for the security control.
Epics

Set up the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be in the same AWS Region as the Amazon Redshift cluster you want to monitor. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Upload the Lambda code.</td>
<td>Upload the Lambda code .zip file provided in the Attachments section to the S3 bucket.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Deploy the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the AWS</td>
<td>Open the AWS CloudFormation console in the same AWS Region as your S3 bucket and deploy the attached template RedshiftSSLEndpointsRequired.yml. For more information about deploying AWS CloudFormation templates, see Creating a stack on the AWS CloudFormation console in the CloudFormation documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>CloudFormation template.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete the parameters in the</td>
<td>When you launch the template, you'll be prompted for the following information:</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>template.</td>
<td>• <strong>S3 bucket</strong>: Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (.zip file).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>S3 key</strong>: Specify the location of the Lambda .zip file</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>• Notification email:</td>
<td>Provide an active email address where you want to receive Amazon SNS notifications.</td>
<td></td>
</tr>
<tr>
<td>• Lambda logging level:</td>
<td>Specify the logging level and frequency for the Lambda function. Use <strong>Info</strong> to log detailed informational messages on progress, <strong>Error</strong> for error events that would still allow the deployment to continue, and <strong>Warning</strong> for potentially harmful situations.</td>
<td></td>
</tr>
</tbody>
</table>

**Confirm the subscription**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email to the email address you provided. You must confirm this email subscription to start receiving violation notifications.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Related resources**

- Creating an S3 bucket [Amazon S3 documentation](#)
- Uploading files to an S3 bucket [Amazon S3 documentation](#)
- Creating a stack on the AWS CloudFormation console [AWS CloudFormation documentation](#)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail [AWS CloudTrail documentation](#)
- Creating an Amazon Redshift cluster [Amazon Redshift documentation](#)
- Configuring security options for connections [Amazon Redshift documentation](#)

**Attachments**

attachment.zip

**Monitor ElastiCache clusters for security groups**
Summary

Amazon ElastiCache is an Amazon Web Services (AWS) service that provides a high-performance, scalable, and cost-effective caching solution for distributing an in-memory data store or cache environment in the cloud. It retrieves data from high-throughput and low-latency, in-memory data stores. This functionality makes it a popular choice for real-time use cases such as caching, session stores, gaming, geo-spatial services, real-time analytics, and queuing. ElastiCache offers Redis and Memcached data stores, both of which provide sub-millisecond response times.

A security group acts as a virtual firewall for your ElastiCache instances by controlling inbound and outbound traffic. Security groups act at the instance level, not at the subnet level. For each security group, you add one set of rules that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. You can specify allow rules but not deny rules.

This pattern provides a security control that monitors for API calls and generates an Amazon CloudWatch Events event on the CreateReplicationGroup, CreateCacheCluster, ModifyCacheCluster, and ModifyReplicationGroup operations. This event calls an AWS Lambda function, which runs a Python script. The function gets the replication group ID from the event JSON input, and performs the following checks to determine whether there's a security violation:

- Checks if the security group of the cluster matches the security group that's configured in the Lambda function.
- If the security group of the cluster doesn't match, the function sends a violation message to an email address you provide, by using an Amazon Simple Notification Service (Amazon SNS) notification.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An S3 bucket to upload the provided Lambda code.
- An email address where you would like to receive violation notifications.
- ElastiCache logging enabled, for access to all the API logs.

Limitations

- This detective control is regional and must be deployed in each AWS Region that you want to monitor.
- The control supports replication groups that are running in a virtual private cloud (VPC).

Architecture

Workflow architecture
Automation and scale

- If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- **Amazon ElastiCache** – Amazon ElastiCache makes it easy to set up, manage, and scale distributed in-memory cache environments in the AWS Cloud. It provides a high performance, resizable, and cost-effective in-memory cache, while removing complexity associated with deploying and managing a distributed cache environment. ElastiCache works with both the Redis and Memcached engines.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **AWS CloudWatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes an attachment with two files:

- **ElastiCacheAllowedSecurityGroup.zip** is a compressed file that includes the security control (Lambda code).

- **ElastiCacheAllowedSecurityGroup.yml** is a CloudFormation template that deploys the security control.

See the *Epics* section for information about how to use these files.
**Epics**

**Deploy the security control**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the code to an S3 bucket.</td>
<td>Create a new S3 bucket or use an existing S3 bucket to upload the attached ElastiCacheAllowedSecurityGroup.zip file (Lambda code). This bucket must be in the same AWS Region as the resources that you want to evaluate.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the ElastiCacheAllowedSecurityControl.yml file that's provided in the attachment. In the next epic, provide values for the template parameters.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Complete the parameters in the CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the resource that will be evaluated.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, ElastiCacheAllowedSecurityGroup.zip or controls/ ElasticCacheAllowedSecurityGroup.zip).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address where you want to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Specify a logging level.</td>
<td>Specify the logging level and verbosity. Info designates detailed informational messages on the application's progress and should be used only for</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the email subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Amazon VPCs and ElastiCache security (Amazon ElastiCache for Redis documentation)

Attachments

attachment.zip

Verify that new Amazon Redshift clusters launch in a VPC

Created by Priyanka Chaudhary (AWS)

| Environment: Production | Technologies: Security, identity, compliance; Analytics; Databases | AWS services: Amazon CloudWatch; AWS Lambda; Amazon Redshift |

Summary

This pattern provides an Amazon Web Services (AWS) CloudFormation template that automatically notifies you when an Amazon Redshift cluster is launched outside a virtual private cloud (VPC).

Amazon Redshift is a fully managed, petabyte-scale, cloud-based data warehouse product. It is designed for large-scale dataset storage and analysis. It is also used to perform large-scale database migrations. Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS
Cloud where you can launch AWS resources such as Amazon Redshift clusters in a virtual network that you define.

The security control provided with this pattern monitors Amazon Redshift API calls in AWS CloudTrail logs, and initiates an Amazon CloudWatch Events event for the CreateCluster and RestoreFromClusterSnapshot APIs. When the event detects one of these APIs, it calls AWS Lambda, which runs a Python script. The Python function analyzes the CloudWatch event. If an Amazon Redshift cluster is created or restored from a snapshot and appears outside the Amazon VPC network, the function sends an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information: the Amazon Redshift cluster name, AWS Region, AWS account, and Amazon Resource Name (ARN) for Lambda that this notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A VPC with a cluster subnet group, and an associated security group.

Limitations

- The AWS CloudFormation template supports the CreateCluster and RestoreFromClusterSnapshot actions (new clusters) only. It does not detect existing Amazon Redshift clusters that were created outside a VPC.
- This security control is regional. You must deploy it in each AWS Region you want to monitor.

Architecture

Target architecture

Automation and scale

If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually.
Epics

Set up the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket</td>
<td>On the <a href="https://console.aws.amazon.com/s3/home">Amazon S3 console</a>, choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be in the same AWS Region as the Amazon Redshift cluster that you want to monitor. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Upload the Lambda code</td>
<td>Upload the Lambda code (RedshiftMustBeInVPC.zip file) provided in the Attachments section to the S3 bucket.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
## Deploy the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the CloudFormation</td>
<td>Open the <a href="https://aws.amazon.com/cloudformation/">AWS CloudFormation console</a> in the same AWS Region as your S3 bucket and deploy the attached template (RedshiftMustBeInVPC.yml). For more information about deploying AWS CloudFormation templates, see <a href="https://docs.aws.amazon.com/cloudformation/latest/userguide/guide-deploy-aws.html">Creating a stack on the AWS CloudFormation console</a> in the CloudFormation documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>template.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete the parameters in the</td>
<td>When you launch the template, you'll be prompted for the following information:</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>template.</td>
<td>• <strong>S3 bucket</strong>: Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (.zip file). • <strong>S3 key</strong>: Specify the location of the Lambda .zip file in your S3 bucket (for example, <code>filename.zip</code> or <code>controls/filename.zip</code>). Do not include leading slashes. • <strong>Notification email</strong>: Provide an active email address where you want to receive Amazon SNS notifications. • <strong>Lambda logging level</strong>: Specify the logging level and frequency for the Lambda function. Use <code>Info</code> to log detailed informational messages on progress, <code>Error</code> for error events that would still allow the deployment to continue, and <code>Warning</code> for potentially harmful situations.</td>
<td></td>
</tr>
</tbody>
</table>

## Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email to the email address you provided.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
**Task**

You must confirm this email subscription to start receiving violation notifications.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>You must confirm this email subscription to start receiving violation notifications.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- Creating an S3 bucket ([Amazon S3 documentation](#))
- Uploading files to an S3 bucket ([Amazon S3 documentation](#))
- Creating a stack on the AWS CloudFormation console ([AWS CloudFormation documentation](#))
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail ([AWS CloudTrail documentation](#))
- Creating an Amazon Redshift cluster ([Amazon Redshift documentation](#))

**Attachments**

attachment.zip

---

**Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools**

*Created by Bibhuti Sahu (AWS)*

<table>
<thead>
<tr>
<th>Created by:</th>
<th>AWS</th>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Websites &amp; web apps; Security, identity, compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services</strong>:</td>
<td>Amazon Cognito</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Summary**

This pattern discusses how you can configure Amazon Cognito user pools and identity pools, and then enable an ASP.NET Core app to access AWS resources after successful authentication.

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. The two main components of Amazon Cognito are user pools and identity pools.

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Your users can also sign in through social identity providers like Google, Facebook, Amazon, or Apple, and through SAML identity providers.

Amazon Cognito identity pools (federated identities) enable you to create unique identities for your users and federate them with identity providers. With an identity pool, you can obtain temporary, limited-privilege AWS credentials to access other AWS services. Before you can begin using your new Amazon Cognito identity pool, you must assign one or more AWS Identity and Access Management (IAM) roles to determine the level of access you want your application users to have to your AWS resources. Identity pools define two types of identities: authenticated and unauthenticated. Each identity type
can be assigned its own role in IAM. Authenticated identities belong to users who are authenticated by a public login provider (Amazon Cognito user pools, Facebook, Google, SAML, or any OpenID Connect providers) or a developer provider (your own backend authentication process), whereas unauthenticated identities typically belong to guest users. When Amazon Cognito receives a user request, the service determines whether the request is authenticated or unauthenticated, determines which role is associated with that authentication type, and then uses the policy attached to that role to respond to the request.

Prerequisites and limitations

Prerequisites

- An AWS account with Amazon Cognito and AWS Identity and Access Management (IAM) permissions
- Access to the AWS resources you want to use
- ASP.NET Core 2.0.0 or later

Architecture

Technology stack

- Amazon Cognito
- ASP.NET Core

Target architecture

Tools

Tools, SDKs, and AWS services
• Visual Studio or Visual Studio Code
• Amazon.AspNetCore.Identity.Cognito (1.0.4) - NuGet package
• AWSSDK.S3 (3.3.110.32) - NuGet package
• Amazon Cognito

Code

The attached .zip file includes sample files that illustrate the following:

• How to retrieve an access token for the logged in user
• How to exchange an access token for AWS credentials
• How to access the Amazon Simple Storage Service (Amazon S3) service with AWS credentials

IAM role for authenticated identities

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "mobileanalytics:PutEvents",
        "cognito-sync:*",
        "cognito-identity:*",
        "s3:ListAllMyBuckets"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Epics

Create an Amazon Cognito user pool

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a user pool.</td>
<td>Sign in to the AWS Management Console and open the Amazon Cognito console at <a href="https://console.aws.amazon.com/cognito/home">https://console.aws.amazon.com/cognito/home</a>. Choose &quot;Manage User Pools&quot;. In the top-right corner of the page, choose &quot;Create a user pool&quot;. Provide a name for your user pool, choose &quot;Review defaults&quot;, and then choose &quot;Create pool&quot;. Note the pool ID.</td>
<td>Developer</td>
</tr>
<tr>
<td>Add an app client.</td>
<td>You can create an app to use the built-in webpages for signing up</td>
<td>Developer</td>
</tr>
</tbody>
</table>
and signing in your users. On the navigation bar on the left side of the user pool page, choose "App clients" under "General settings", and then choose "Add an app client". Give your app a name, and choose "Create app client". Note the app client ID and the client secret (choose "Show Details" to see the client secret).

Create an Amazon Cognito identity pool

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an identity pool.</td>
<td>In the Amazon Cognito console, choose &quot;Manage Identity Pools&quot;, and then choose &quot;Create new identity pool.&quot; Type a name for the identity pool. If you want to enable unauthenticated identities, select that option from the &quot;Unauthenticated identities&quot; section. In the &quot;Authentication providers&quot; section, configure the Cognito identity pool by setting the user pool ID and the app client ID, and then choose &quot;Create Pool&quot;.</td>
<td>Developer</td>
</tr>
<tr>
<td>Assign IAM roles for the identity pool.</td>
<td>You can edit the IAM roles for authenticated and unauthenticated users, or keep the defaults, and then choose &quot;Allow&quot;. For this pattern, we will edit the authenticated IAM role and provide access for s3:ListAllMyBuckets. For sample code, see the IAM role provided in the &quot;Tools used&quot; section.</td>
<td>Developer</td>
</tr>
<tr>
<td>Copy the identity pool ID.</td>
<td>When you choose &quot;Allow&quot; in the previous step, the &quot;Getting started with Amazon Cognito&quot; page is displayed. On this page, you can either copy the identity pool ID from the &quot;Get AWS Credentials&quot; section or choose &quot;Edit identity pool&quot; in the upper right and copy the identity pool ID from the screen that's displayed.</td>
<td>Developer</td>
</tr>
</tbody>
</table>
Configure your sample app

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the sample ASP.NET Core web app.</td>
<td>Clone the sample .NET core web app from <a href="https://github.com/aws/aws-aspnnet-cognito-identity-provider.git">https://github.com/aws/aws-aspnnet-cognito-identity-provider.git</a>. Navigate to the samples folder and open the solution. In this project, you'll configure the appsettings.json file and add a new page that will render all S3 buckets after successful sign in.</td>
<td>Developer</td>
</tr>
<tr>
<td>Add dependencies.</td>
<td>Add a NuGet dependency for Amazon.AspNetCore.Identity.Cognito to your ASP.NET Core application.</td>
<td>Developer</td>
</tr>
<tr>
<td>Add the configuration keys and values to appsettings.json.</td>
<td>Include the code from the attached appsettings.json file in your appsettings.json file, and then replace the placeholders with the values from the previous steps.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create a new user and sign in.</td>
<td>Create a new user in the Amazon Cognito user pool, and verify that the user exists under &quot;Users and Groups&quot; in the user pool.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create a new Razor Page called MyS3Buckets.</td>
<td>Add a new ASP.NET Core Razor Page to your sample app, and replace the content for MyS3Bucket.cshtml and MyS3Bucket.cshtml.cs from the attached sample. Add the new MyS3Bucket page under navigation in the _Layout.cshtml page.</td>
<td>Developer</td>
</tr>
</tbody>
</table>

Related resources

- Amazon Cognito
- Amazon Cognito user pools
- Amazon Cognito identity pools
- Access policy samples
- GitHub - AWS ASP.NET Cognito Identity Provider

Attachments

attachment.zip
Authenticate Microsoft SQL Server on Amazon EC2 using AWS Directory Service

Created by Jagdish Kantubugata (AWS)

<table>
<thead>
<tr>
<th>R Type:</th>
<th>N/A</th>
<th>Source:</th>
<th>Active Directory</th>
<th>Target:</th>
<th>AWS Directory Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created by:</td>
<td>AWS</td>
<td>Environment:</td>
<td>PoC or pilot</td>
<td></td>
<td>Technologies:</td>
</tr>
<tr>
<td>Workload:</td>
<td>Microsoft</td>
<td>AWS services:</td>
<td>AWS Directory Service</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to create an AWS Directory Service directory and use it to authenticate Microsoft SQL Server on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

AWS Directory Service provides multiple ways to use Amazon Cloud Directory and Microsoft Active Directory (AD) with other AWS services. Directories store information about users, groups, and devices, and administrators use them to manage access to information and resources. AWS Directory Service provides multiple directory choices for customers who want to use existing Microsoft AD or Lightweight Directory Access Protocol (LDAP)–aware applications in the cloud. It also offers those same choices to developers who need a directory to manage users, groups, devices, and access.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) with a minimum of two private subnets and two public subnets
- An AWS Identity and Access Management (IAM) role to join the server into the domain

Architecture

Source technology stack

- The source can be an on-premises Active Directory

Target technology stack

- AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD)

Target architecture
Tools

- SQL Server Management Studio (SSMS) - SSMS is a tool for managing Microsoft SQL Server, including accessing, configuring, and administering SQL Server components.

Epics

Set up a directory

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select AWS Managed Microsoft AD as the directory type.</td>
<td>In the AWS Directory Service console, choose Directories, Set up directory, AWS Managed Microsoft AD, Next.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Select edition.</td>
<td>Choose Standard Edition from the available editions for AWS Managed Microsoft AD.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Specify the directory DNS name.</td>
<td>Use a fully qualified domain name. This name will resolve inside your VPC only. It does not need to be publicly resolvable.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Set the administrator password.</td>
<td>Set the password for the default administrative user, which is named Admin.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Choose the VPC and subnets.</td>
<td>Choose the VPC that will contain your directory, and the subnets for the domain controllers. If you do not have a VPC with at least two subnets, you must create one.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Review and launch the directory.</td>
<td>Review the edition and price information for the directory, and then choose Create directory.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Launch an EC2 instance for SQL Server in the domain

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select an AMI for SQL Server.</td>
<td>In the Amazon EC2 console at <a href="https://console.aws.amazon.com/ec2/">https://console.aws.amazon.com/ec2/</a>, choose Launch instance, and then select the appropriate Amazon Machine Image (AMI) for SQL Server.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Configure instance details.</td>
<td>Configure the Windows instance to meet your requirements for SQL Server.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Add a network.</td>
<td>You can choose the VPC that your directory was created in.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Add a subnet.</td>
<td>Choose one of the public subnets in your VPC. The subnet that you choose must have all external traffic routed to an internet gateway. If this is not the case, you won't be able to connect to the instance remotely.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Choose your domain.</td>
<td>Choose the domain that you created from the Domain join directory list.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Select an IAM role.</td>
<td>Select an IAM role that has the AWS managed policies AmazonSSMMManagedInstanceCore and AmazonSSMDirectoryServiceAccess attached to it.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the instance.</td>
<td>Select a key pair, and then launch the instance.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>

### Authenticate SQL Server using Directory Service

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log in as a Windows administrator.</td>
<td>Log in to the Windows EC2 instance by using Windows administrator credentials.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Log in to SQL Server.</td>
<td>Launch SQL Server Management Studio (SSMS) and log in to SQL Server by using the Windows authentication method.</td>
<td>DBA</td>
</tr>
<tr>
<td>Create a login for the directory user.</td>
<td>In SSMS, choose Security, and then choose New Login.</td>
<td>DBA</td>
</tr>
<tr>
<td>Search for a login name.</td>
<td>Choose the search button next to the login text box.</td>
<td>DBA</td>
</tr>
<tr>
<td>Select a location.</td>
<td>In the Select User or Group dialog box, choose Locations.</td>
<td>DBA</td>
</tr>
<tr>
<td>Enter network credentials.</td>
<td>Enter the fully qualified network credentials you used when you created the directory service; for example: test.com\admin.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Select the directory.</td>
<td>Choose the AWS directory name, and then choose OK.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Select an object name.</td>
<td>Select the user for which you want to create the login. Select the location, choose the entire directory, search for the user, and add the login.</td>
<td>DBA</td>
</tr>
<tr>
<td>Log in to SQL Server instance.</td>
<td>Log in to the Windows EC2 instance for SQL Server by using your domain credentials.</td>
<td>DevOps</td>
</tr>
<tr>
<td>Log in to SQL Server as a domain user.</td>
<td>Launch SSMS and connect to the database engine by using the Windows authentication method.</td>
<td>DBA</td>
</tr>
</tbody>
</table>

### Related resources

- AWS Directory Service documentation (AWS website)
- Create your AWS Managed Microsoft AD directory (AWS Directory Service documentation)
- Seamlessly join a Windows EC2 instance (AWS Directory Service documentation)
Automate SAML 2.0 federation for AWS multi-account environments that use Azure AD

Created by Adam Spicer (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: Production</th>
<th>Technologies:</th>
<th>Security, identity, compliance; Infrastructure; Management &amp; governance; Hybrid cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Control Tower Customizations to automate SAML 2.0 Federation with Azure AD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Workload:</th>
<th>AWS services:</th>
<th>Security, identity, compliance; Infrastructure; Management &amp; governance; Hybrid cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production</td>
<td>Microsoft</td>
<td>AWS CloudFormation; AWS Identity and Access Management; AWS Organizations; AWS Secrets Manager; AWS Control Tower</td>
<td></td>
</tr>
</tbody>
</table>

Summary

Organizations that operate a multi-account strategy in Amazon Web Services (AWS) with AWS Control Tower or AWS Organizations might have a requirement to use Microsoft Azure Active Directory (Azure AD) single sign-on (SSO) for federation into the AWS Management Console. In situations where AWS Single Sign-On might not fit into an organization’s requirements, federation is achieved by using an enterprise application within Azure AD named AWS Single-Account Access, which is deployed within Azure AD for each AWS account.

As described in a Microsoft tutorial on integrating Azure AD SSO with AWS, programmatic access to each integrated account must be configured by using an AWS Identity and Access Management (IAM) user in each account. Azure AD uses the IAM user to retrieve IAM roles to synchronize with Azure AD. Domain administrators map those synchronized roles to groups of users within Azure AD to enable those users to federate into AWS with the appropriate IAM role.

If you have a multi-account strategy, the process of manually deploying the integration steps can be cumbersome and can slow down the process of enabling federation into a new AWS account. As each new AWS account gets created, domain administrators often rely on their cloud administrator to create the IAM user, generate the programmatic access keys, and provide the keys to domain administrators in a secure manner. This is often a manual process that prolongs the time from account creation until users can federate into the account. The solution provided in this pattern accelerates this process by providing prescriptive guidance and automation to enable domain administrators to directly obtain the IAM user programmatic access keys in a least privileged manner, without having to involve a third party. As a result, configuring federation for a new AWS account is streamlined through an improved process and automation.

Prerequisites and limitations

Prerequisites
• An AWS multi-account configuration that uses AWS Control Tower or AWS Organizations
• Azure AD Enterprise Applications (EA) created for AWS management and member accounts, with the identity provider (IdP) configured by using the EA metadata

Limitations

• The accounts in scope must be a member of the management account's AWS organization.
• All accounts in scope must be within an AWS organizational unit (OU).

Architecture

Target technology stack

After you implement this pattern, resources will be deployed into the management account and each member account. The following tables list these resources.

Management account

<table>
<thead>
<tr>
<th>AWS resource</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM role</td>
<td>AzureAdFederationAdminRole</td>
<td>An IAM role with a trust policy configured for the SAML identity provider (IdP) that allows the role to be assumed only by a companion role for member accounts in the organization. This IAM role acts as an intermediate role so that users can be configured to federate into the management account with this role and then assume the appropriate role in member accounts.</td>
</tr>
<tr>
<td>AWS CloudFormation stack set</td>
<td>AzureAdFederationStackSet</td>
<td>A stack set that enables you to deploy a stack to all accounts within the organization.</td>
</tr>
<tr>
<td>IAM user</td>
<td>AzureADAutomationUser</td>
<td>The IAM user to be used by Azure AD SSO to synchronize the roles for federation.</td>
</tr>
<tr>
<td>IAM group</td>
<td>AzureADAutomationGroup</td>
<td>The IAM group that contains the IAM user AzureADAutomationUser.</td>
</tr>
<tr>
<td>IAM managed policy</td>
<td>AzureADAllowIAMListRoles</td>
<td>A policy that allows the iam:ListRoles action on all resources. The policy is attached to the IAM group AzureADAutomationGroup to enable Azure AD to synchronize IAM roles.</td>
</tr>
<tr>
<td>AWS Secrets Manager secret access key</td>
<td>AzureAdFederation / CFNUserSecretAccessKey</td>
<td>The programmatic access keys for the IAM user</td>
</tr>
</tbody>
</table>
Member accounts

The resources deployed into the member accounts are created from the stack set (AzureAdFederationStackSet) that is configured in the management account.

<table>
<thead>
<tr>
<th>AWS resource</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM role</td>
<td>AzureAdFederationAssumeRole</td>
<td>An IAM role with a trust policy that is configured to allow to be assumed by a companion role created in the management account. The role allows access only to get the secret value that is stored in Secrets Manager.</td>
</tr>
<tr>
<td>IAM user</td>
<td>AzureADAutomationUser</td>
<td>The IAM user to be used by Azure AD SSO to synchronize the roles for federation.</td>
</tr>
<tr>
<td>IAM group</td>
<td>AzureADAutomationGroup</td>
<td>The IAM group that contains the IAM user AzureADAutomationUser.</td>
</tr>
<tr>
<td>IAM managed policy</td>
<td>AzureADAllowIAMLListRoles</td>
<td>A policy that allows the iam:ListRoles action on all resources. The policy is attached to the IAM group AzureADAutomationGroup to enable Azure AD to synchronize IAM roles.</td>
</tr>
<tr>
<td>Secrets Manager secret access key</td>
<td>AzureAdFederation / CFNUserSecretAccessKey</td>
<td>The programmatic access keys for the IAM user AzureAdAutomationUser, which are generated and stored as a secret within Secrets Manager.</td>
</tr>
</tbody>
</table>

Target architecture

The solution deploys a target architecture that uses AWS CloudFormation stacks and stack sets to create the federation components in all accounts. The following diagram illustrates the target architecture.
The following diagram shows the workflow for enabling and configuring the synchronization of member account IAM roles.
After member accounts have been created (step 1 in the diagram), a domain administrator can obtain the IAM user programmatic access keys to all member accounts by following these remaining steps:

- Step 2. Federate into the management account by using the IAM role `AzureAdFederationAdminRole`.
- Step 3. In the AWS Management Console, switch role to a new member account by using the IAM role `AzureAdFederationAssumeRole`.
- Step 4. Navigate to Secrets Manager and retrieve the keys in the `AzureADFederation / CFNUserSecretAccessKey` secret.
- Step 5. In the appropriate AWS enterprise application within Azure, configure the provisioning administrator credentials by using the retrieved secrets.

## Tools

### AWS services

- **AWS CloudFormation** – AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly.
- **IAM** – AWS Identity and Access Management (IAM) helps you secure access to AWS services and resources. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- **AWS Organizations** – AWS Organizations is an account management service that lets you consolidate multiple AWS accounts into an organization that you create and centrally manage.
• **AWS Control Tower** – AWS Control Tower provides the easiest way to set up and govern a secure, compliant, multi-account AWS environment based on best practices established by working with thousands of enterprises.

• **AWS Secrets Manager** – AWS Secrets Manager enables you to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

**Code**

The source code for this pattern is available on GitHub, in the [AWS Control Tower Customizations repository](https://github.com/aws-solutions/aws-control-tower-customizations). The source code includes two CloudFormation templates (YAML files) that you can use to deploy the target architecture. The epics and stories in the next section provide detailed instructions for using these templates.

**Epics**

**Create the federation administrator role and management account secrets**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the CloudFormation template to create the administrator role and IAM user in the management account.</td>
<td>From the management account, run the CloudFormation script named <code>azuread-fed-management-account.yaml</code> (see the GitHub repository). You will need to specify your AWS organization ID and the name of the SAML provider that exists in the management account.</td>
<td>CloudFormation</td>
</tr>
</tbody>
</table>

**Create federation assume-role resources in all member accounts**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create a CloudFormation stack set for all member accounts. | From the management account, use the `azuread-fed-member-account.yaml` template (see the GitHub repository) to create a new stack set with the following stack set deployment options. Deploy stacks in organization units:  
  - Specify all top-level parent OUs for member accounts that will receive this stack.  
  - Be sure that the management account and all other core accounts are within one of these top-level OUs. | CloudFormation |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regions: Choose one AWS Region where the IAM secrets will be stored within Secrets Manager.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Configure Azure AD federation synchronization for all member accounts**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Configure Azure AD Federation synchronization for the management account | The new role named AzureAdFederationAdminRole in the management account needs to be mapped within the Azure AD AWS enterprise application to enable an Azure AD administrator to configure the role synchronization process for all member accounts. Follow these steps:  
1. Obtain the IAM user programmatic access keys for AzureADAutomationUser in the management account by navigating to Secrets Manager and accessing the secret named AzureADFederation / CFNUserSecretAccessKey.  
2. Within the AWS enterprise application for the management account, configure the administrator credentials in the provisioning area by using the IAM user programmatic access keys you obtained in the previous step, test the connection, and enable provisioning.  
3. Wait for the synchronization to complete. Then map the AzureAdFederationAdminRole role to the appropriate Azure AD administrator users within the AWS enterprise application for the management account, to give them access to the role for the management account.  
4. Test the AWS enterprise application by federating into the management account using the | Azure AD administrator, Secrets Manager |

1589
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>AzureAdFederationAdminRole</strong> role. Test your access to one of the member accounts by using the following process:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a. Choose <strong>Switch Roles</strong> from the account menu, and provide one of the member account IDs and the <strong>AzureAdFederationAssumeRole</strong> role name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. After you switch roles, make sure that you can access the secret named <strong>AzureADFederation / CFNUserSecretAccessKey</strong> in Secrets Manager.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| Configure Azure AD federation synchronization for all member accounts. | As an Azure AD administrator, first federate into the management account by using the AzureAdFederationAdminRole role. For each member account, follow these steps:  
   1. From the management account, choose **Switch Roles** from the account menu, and provide one of the member account IDs and the AzureAdFederationAssumeRole role name. Then choose **Switch Role**.  
   2. In the member account, access the IAM user programmatic access key secret named AzureADFederation / CFNUserSecretAccessKey in Secrets Manager.  
   3. Within the given member account AWS enterprise application, configure the IAM user credentials in the provisioning area by using the IAM user programmatic access keys obtained earlier, test the connection, and enable provisioning.  
   4. Verify that any federation IAM roles are synchronized to the given AWS enterprise application. Map the roles appropriately to Azure AD users.  
   Repeat these steps for each member account. | Azure AD administrator, Secrets Manager |

**Related resources**

- AWS Identity and Access Management using SAML (AWS blog post)
- Configure single sign-on using Microsoft Azure Active Directory for Amazon Connect (AWS blog post)
- Tutorial: Azure Active Directory single sign-on (SSO) integration with AWS (Microsoft documentation)
Automate remediation for AWS Security Hub standard findings

Created by Chandini Penmetsa (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Security, identity, compliance</td>
</tr>
<tr>
<td>AWS services:</td>
<td>AWS CloudFormation; Amazon CloudWatch; AWS Lambda; AWS Security Hub; Amazon SNS</td>
</tr>
<tr>
<td>Workload:</td>
<td>All other workloads</td>
</tr>
</tbody>
</table>

Summary

With AWS Security Hub, you can enable checks for standard best practices such as the following:

- AWS Foundational Security Best Practices
- CIS AWS Foundations Benchmark
- PCI DSS

Each of these standards has predefined controls. Security Hub checks for the control in a given AWS account and reports the findings.

AWS Security Hub sends all findings to Amazon EventBridge by default. This pattern provides a security control that deploys an EventBridge rule to identify AWS Foundational Security Best Practices standard findings. The rule identifies the following findings for automatic scaling, virtual private clouds (VPCs), Amazon Elastic Block Store (Amazon EBS), and Amazon Relational Database Service (Amazon RDS) from the AWS Foundational Security Best Practices standard:

- [AutoScaling.1] Auto Scaling groups associated with a load balancer should use load balancer health checks
- [EC2.2] The VPC default security group should not allow inbound and outbound traffic
- [EC2.6] VPC flow logging should be enabled in all VPCs
- [EC2.7] EBS default encryption should be enabled
- [RDS.1] RDS snapshots should be private
- [RDS.6] Enhanced monitoring should be configured for RDS DB instances and clusters
- [RDS.7] RDS clusters should have deletion protection enabled

The EventBridge rule forwards these findings to an AWS Lambda function, which remediates the finding. The Lambda function then sends a notification with remediation information to an Amazon Simple Notification Service (Amazon SNS) topic.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the remediation notification
• Security Hub and AWS Config enabled in the AWS Region where you intend to deploy the control
• An Amazon Simple Storage Service (Amazon S3) bucket in same Region as the control to upload the AWS Lambda code

Limitations
• This security control automatically remediates new findings reported after the security control deployment. To remediate existing findings, select the findings manually on the Security Hub console. Then, under Actions, select the AFSBPRemediation custom action that was created as part of the deployment by AWS CloudFormation.
• This security control is regional and must be deployed in the AWS Regions that you intend to monitor.
• For the EC2.6 remedy, to enable VPC Flow Logs, an Amazon CloudWatch Logs log group will be created with /VpcFlowLogs/vpc_id format. If a log group exists with same name, the existing log group will be used.
• For the EC2.7 remedy, to enable Amazon EBS default encryption, the default AWS Key Management Service (AWS KMS) key is used. This change prevents the use of certain instances that do not support encryption.

Architecture

Target technology stack
• Lambda function
• Amazon SNS topic
• EventBridge rule
• AWS Identity and Access Management (IAM) roles for Lambda function, VPC Flow Logs, and Amazon Relational Database Service (Amazon RDS) Enhanced Monitoring

Target architecture

Automation and scale
If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want this to monitor.

Tools

Tools
• AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.
• EventBridge – Amazon EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, routing that data to targets such as Lambda functions.
• Lambda – AWS Lambda supports running code without provisioning or managing servers.
Epics

Deploy the security control

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket must be in the same Region as the Security Hub findings that are being evaluated.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Upload the Lambda code to the S3 bucket.</td>
<td>Upload the Lambda code .zip file that's provided in the “Attachments” section to the defined S3 bucket.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>Deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the Amazon S3 prefix.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, &lt;directory&gt;/&lt;filename&gt;.zip).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the SNS topic ARN.</td>
<td>Provide the SNS topic Amazon Resource Name (ARN) if you want to use an existing SNS topic for remediation</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an email address where you want to receive the remediation notifications (needed only when you want AWS CloudFormation to create the SNS topic).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. “Info” designates detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the VPC Flow Logs IAM role ARN.</td>
<td>Provide the IAM role ARN to be used for VPC Flow Logs. (If &quot;None&quot; is entered as input, AWS CloudFormation creates an IAM role and uses it.)</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the RDS Enhanced Monitoring IAM role ARN.</td>
<td>Provide the IAM role ARN to be used for RDS Enhanced Monitoring. (If &quot;None&quot; is entered, AWS CloudFormation creates an IAM role and uses it.)</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the Amazon SNS subscription.</td>
<td>When the template successfully deploys, if a new SNS topic was created, a subscription message is sent to the email address that you provided. To receive remediation notifications, you must confirm this subscription email message.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating a stack on the AWS CloudFormation console
- AWS Lambda
- AWS Security Hub
Automate security scans for cross-account workloads using Amazon Inspector and AWS Security Hub

Created by Ramya Pulipaka (AWS) and Mikesh Khanal (AWS)

| Environment: Production | Technologies: Security, identity, compliance; Operations | AWS services: Amazon Inspector; Amazon SNS; AWS Lambda; AWS Security Hub; Amazon CloudWatch |

Summary

This pattern describes how to automatically scan for vulnerabilities in cross-account workloads on the Amazon Web Services (AWS) Cloud.

The pattern helps create a schedule for host-based scans of Amazon Elastic Compute Cloud (Amazon EC2) instances that are grouped by tags or for network-based Amazon Inspector scans. An AWS CloudFormation stack deploys all the required AWS resources and services to your AWS accounts.

The Amazon Inspector findings are exported to AWS Security Hub and provide insights into vulnerabilities across your accounts, AWS Regions, virtual private clouds (VPCs), and EC2 instances. You can receive these findings by email or you can create an Amazon Simple Notification Service (Amazon SNS) topic that uses an HTTP endpoint to send the findings to ticketing tools, security information and event management (SIEM) software, or other third-party security solutions.

Prerequisites and limitations

Prerequisites

- An existing email address to receive email notifications from Amazon SNS.
- An existing HTTP endpoint used by ticketing tools, SIEM software, or other third-party security solutions.
- Active AWS accounts that host cross-account workloads, including a central audit account.
- Security Hub, enabled and configured. You can use this pattern without Security Hub, but we recommend using Security Hub because of the insights it generates. For more information, see Setting up Security Hub in the AWS Security Hub documentation.
- An Amazon Inspector agent must be installed on each EC2 instance that you want to scan. You can install the Amazon Inspector agent on multiple EC2 instances by using AWS Systems Manager Run Command.
Skills

- Experience using self-managed and service-managed permissions for stack sets in AWS CloudFormation. If you want to use self-managed permissions to deploy stack instances to specific accounts in specific Regions, you must create the required AWS Identity and Access Management (IAM) roles. If you want to use service-managed permissions to deploy stack instances to accounts managed by AWS Organizations in specific Regions, you don’t need to create the required IAM roles. For more information, see Create a stack set in the AWS CloudFormation documentation.

Limitations

- If no tags are applied to EC2 instances in an account, then Amazon Inspector scans all the EC2 instances in that account.
- The AWS CloudFormation stack sets and the onboard-audit-account.yaml file (attached) must be deployed in the same Region.
- By default, Amazon Inspector Classic doesn’t support aggregated findings. Security Hub is the recommended solution to viewing assessments for multiple accounts or AWS Regions.
- This pattern’s approach can scale under the publish quota of 30,000 transactions per second (TPS) for an SNS topic in the US East (N. Virginia) Region (us-east-1), although limits vary by Region. To scale more effectively and avoid data loss, we recommend using Amazon Simple Queue Service (Amazon SQS) in front of the SNS topic.

Architecture

The following diagram illustrates the workflow for automatically scanning EC2 instances.

The workflow consists of the following steps:
1. An Amazon EventBridge rule uses a cron expression to self-initiate on a specific schedule and initiates Amazon Inspector.

2. Amazon Inspector scans the tagged EC2 instances in the account.

3. Amazon Inspector sends the findings to Security Hub, which generates insights for workflow, prioritization, and remediation.

4. Amazon Inspector also sends the assessment's status to an SNS topic in the audit account. An AWS Lambda function is invoked if a findings reported event is published to the SNS topic.

5. The Lambda function fetches, formats, and sends the findings to another SNS topic in the audit account.

6. Findings are sent to the email addresses that are subscribed to the SNS topic. The full details and recommendations are sent in JSON format to the subscribed HTTP endpoint.

**Technology stack**

- AWS Control Tower
- EventBridge
- IAM
- Amazon Inspector
- Lambda
- Security Hub
- Amazon SNS

**Tools**

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications.

- **AWS CloudFormation StackSets** – AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and Regions with a single operation.

- **AWS Control Tower** – AWS Control Tower creates an abstraction or orchestration layer that combines and integrates the capabilities of several other AWS services, including AWS Organizations.

- **Amazon EventBridge** – EventBridge is a serverless event bus service that makes it easy to connect your applications with data from a variety of sources.

- **AWS Lambda** – Lambda is a compute service that helps you run code without provisioning or managing servers.

- **AWS Security Hub** – Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.

**Epics**

**Deploy the AWS CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template in the audit account.</td>
<td>Download and save the onboard-audit-</td>
<td>Developer, Security engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>account.yaml file (attached) to a local path on your computer.</td>
<td>Developer, Security engineer</td>
</tr>
<tr>
<td></td>
<td>Sign in to the AWS Management Console for your audit account, open the AWS CloudFormation console, and then choose <strong>Create stack</strong>.</td>
<td></td>
</tr>
</tbody>
</table>
|      | Choose **Prepare template** in the **Prerequisites** section, and then choose **Template is ready**.  
      | Choose **Template source** in the **Specify template** section, and then choose **Template is ready**.  
      | Upload the onboard-audit-account.yaml file and then configure the remaining options according to your requirements. |  |
|      | **Important**: Make sure that you configure the following input parameters: |  |
|      | • **DestinationEmailAddress** – Enter an email address to receive findings. |  |
|      | • **HTTPEndpoint** – Provide an HTTP endpoint for your ticketing or SIEM tools. |  |
|      | You can also deploy the AWS CloudFormation template by using AWS Command Line Interface (AWS CLI). For more information about this, see [Creating a stack](#) in the AWS CloudFormation documentation. |  |
|      | **Confirm the Amazon SNS subscription.** |  |
|      | Open your email inbox and choose **Confirm subscription** in the email that you receive from Amazon SNS. This opens a web browser window and displays the subscription confirmation. |  |
# Create AWS CloudFormation stack sets to automate the Amazon Inspector scan schedule

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create stack sets in the audit account.</td>
<td>Download the vulnerability-management-program.yaml file (attached) to a local path on your computer. On the AWS CloudFormation console, choose View stacksets and then choose Create StackSet. Choose Template is ready, choose Upload a template file, and then upload the vulnerability-management-program.yaml file. If you want to use self-managed permissions, follow the instructions from Create a stack set with self-managed permissions in the AWS CloudFormation documentation. This creates stack sets in individual accounts. If you want to use service-managed permissions, follow the instructions from Create a stack set with service-managed permissions in the AWS CloudFormation documentation. This creates stack sets in your entire organization or specified organizational units (OUs). <strong>Important:</strong> Make sure that the following input parameters are configured for your stack sets:</td>
<td>Developer, Security engineer</td>
</tr>
</tbody>
</table>

- **AssessmentSchedule** – The schedule for EventBridge using cron expressions.
- **Duration** – The duration of the Amazon Inspector assessment run in seconds.
- **CentralSNSTopicArn** – The Amazon Resource Name (ARN) for the central SNS topic.
- **Tagkey** – The tag key that is associated with the resource group.
AWS Prescriptive Guidance Patterns

Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Tagvalue – The tag value that is associated with the resource group.</td>
<td>If you want to scan EC2 instances in the audit account, you must run the <code>vulnerability-management-program.yaml</code> file as an AWS CloudFormation stack in the audit account.</td>
<td></td>
</tr>
<tr>
<td>Validate the solution.</td>
<td>Check that you receive findings by email or HTTP endpoint on the schedule that you specified for Amazon Inspector.</td>
<td>Developer, Security engineer</td>
</tr>
</tbody>
</table>

Related resources

• Scale your security vulnerability testing with Amazon Inspector
• Automatically remediate Amazon Inspector security findings
• How to simplify security assessment setup by using Amazon EC2, AWS Systems Manager, and Amazon Inspector

Attachments

attachment.zip

Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config

*Created by Manigandan Shri (AWS)*

Environment: Production

Technologies: Infrastructure; Operations; Security, identity, compliance

AWS services: Amazon S3; AWS Config; AWS KMS; AWS Identity and Access Management; AWS Systems Manager; AWS CloudTrail

Summary

Visibility over activity in your Amazon Web Services (AWS) account is an important security and operational best practice. AWS CloudTrail helps you with the governance, compliance, and operational and risk auditing of your account.
To ensure that CloudTrail remains enabled in your account, AWS Config provides the `cloudtrail-enabled` managed rule. If CloudTrail is turned off, the `cloudtrail-enabled` rule automatically re-enables it by using automatic remediation.

However, you must make sure that you follow security best practices for CloudTrail if you use automatic remediation. These best practices include enabling CloudTrail in all AWS Regions, logging read and write workloads, enabling insights, and encrypting log files with server-side encryption using AWS Key Management Service (AWS KMS) managed keys (SSE-KMS).

This pattern helps you follow these security best practices by providing a custom remediation action to automatically re-enable CloudTrail in your account.

**Important:** We recommend using service control policies (SCPs) to prevent any tampering with CloudTrail. For more information about this, see the Prevent tampering with AWS CloudTrail section of How to use AWS Organizations to simplify security at enormous scale on the AWS Security Blog.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account
- AWS Identity and Access Management (IAM) user and role permissions to create an AWS Systems Manager Automation runbook
- An existing trail for your account

**Limitations**

This pattern doesn't support the following actions:

- Setting an Amazon Simple Storage Service (Amazon S3) prefix key for the storage location
- Publishing to an Amazon Simple Notification Service (Amazon SNS) topic
- Configuring Amazon CloudWatch Logs to monitor your CloudTrail logs
Architecture

Technology stack

- AWS Config
- CloudTrail
- Systems Manager
- Systems Manager Automation

Tools

**AWS Config** – AWS Config provides a detailed view of the configuration of AWS resources in your account.

**AWS CloudTrail** – CloudTrail helps you enable governance, compliance, and operational and risk auditing of your account.

**AWS KMS** – AWS Key Management Service (AWS KMS) is an encryption and key management service.

**AWS Systems Manager** – Systems Manager helps you view and control your infrastructure on AWS.

**AWS Systems Manager Automation** – Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon Elastic Compute Cloud (Amazon EC2) instances and other AWS resources.

**Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
**Epics**

**Configure CloudTrail**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket.</td>
<td>Sign in to the AWS Management Console, open the Amazon S3 console, and then create an S3 bucket to store the CloudTrail logs. For more information, see <a href="https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket.html">Create an S3 bucket</a> in the Amazon S3 documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Add a bucket policy to allow CloudTrail to deliver log files to the S3 bucket.</td>
<td>CloudTrail must have the required permissions to deliver log files to your S3 bucket. On the Amazon S3 console, choose the S3 bucket that you created earlier and then choose Permissions. Create an S3 bucket policy by using the <a href="https://docs.aws.amazon.com/cloudtrail/latest/userguide/permission-cloudtrail.html">Amazon S3 bucket policy for CloudTrail</a> from the CloudTrail documentation. For steps on how to add a policy to an S3 bucket, see <a href="https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket-policies.html">Adding a bucket policy using the Amazon S3 console</a> in the Amazon S3 documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Create a KMS key.</td>
<td>Create an AWS KMS key for CloudTrail to encrypt objects before adding them to the S3 bucket. For help with this story, see <a href="https://docs.aws.amazon.com/cloudtrail/latest/userguide/secure-cloudtrail.html">Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS)</a> in the CloudTrail documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Add a key policy to the KMS key.</td>
<td>Attach a KMS key policy to allow CloudTrail to use the KMS key. For help with this story, see Encrypting CloudTrail log files with AWS KMS–managed keys (SSE-KMS) in the CloudTrail documentation. Important: CloudTrail doesn’t require Decrypt permissions.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Create AssumeRole for Systems Manager runbook</td>
<td>Create an AssumeRole for Systems Manager Automation to run the runbook. For instructions and more information about this, see Setting up automation in the Systems Manager documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Create and test the Systems Manager Automation runbook</td>
<td></td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Test the runbook.</td>
<td>On the System Manager console, test the Systems Manager Automation runbook that you created earlier. For more information about this, see Running a simple automation in the Systems Manager documentation.</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Set up the automatic remediation rule in AWS Config</td>
<td></td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Add the cloudtrail-enabled rule.</td>
<td>On the AWS Config console, choose Rules and then choose Add rule. On the Add rule page, choose Add custom rule. On the Configure rule page, enter a name and description, and add the cloudtrail-enabled</td>
<td>Systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>rule. For more information, see <a href="#">Managing your AWS Config rules</a> in the AWS Config documentation.</td>
<td></td>
</tr>
</tbody>
</table>
| Add the automatic remediation action. | From the **Actions** dropdown list, choose **Manage remediation**. Choose **Auto remediation** and then choose the Systems Manager runbook that you created earlier. The following are the required input parameters for CloudTrail:  
  - CloudTrailName  
  - CloudTrailS3BucketName  
  - CloudTrailKmsKeyId  
  - AssumeRole (optional)  
  The following input parameters are set to true by default:  
  - IsMultiRegionTrail  
  - IsOrganizationTrail  
  - IncludeGlobalServiceEvents  
  - EnableLogFileValidation  
  Retain the default values for the **Rate Limits parameter** and **Resource ID parameter**. Choose **Save**. For more information about this, see [Remediating noncompliant AWS resources with AWS Config rules](#) in the AWS Config documentation. | Systems administrator |
**Task** | **Description** | **Skills required**
--- | --- | ---
Test the automatic remediation rule. | To test the automatic remediation rule, open the CloudTrail console, choose Trails, and then choose the trail. Choose Stop logging to turn off logging for the trail. When you are prompted to confirm, choose Stop logging. CloudTrail stops logging activity for that trail. Follow the instructions from Evaluating your resources in the AWS Config documentation to make sure that CloudTrail was automatically re-enabled. | Systems administrator

---

**Related resources**

**Configure CloudTrail**

- Create an S3 bucket
- Amazon S3 bucket policy for CloudTrail
- Adding a bucket policy using the Amazon S3 console
- Creating a trail
- Setting up automation
- Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS)

**Create and test the Systems Manager Automation runbook**

- Creating Systems Manager documents
- Running a simple automation

**Set up the automatic remediation rule in AWS Config**

- Managing your AWS Config rules
- Remediating noncompliant AWS resources with AWS Config rules

**Additional resources**

- AWS CloudTrail - Security best practices
- Getting started with AWS Systems Manager
- Getting started with AWS Config
- Getting started with AWS CloudTrail
Automatically remediate unencrypted Amazon RDS DB instances and clusters

Created by Ajay Rawat (AWS) and Josh Joy (AWS)

| Environment: PoC or pilot | Technologies: Security, identity, compliance; Databases | AWS services: AWS Config; AWS KMS; AWS Identity and Access Management; AWS Systems Manager; Amazon RDS |

Summary

This pattern describes how to automatically remediate unencrypted Amazon Relational Database Service (Amazon RDS) DB instances and clusters on Amazon Web Services (AWS) by using AWS Config, AWS Systems Manager runbooks, and AWS Key Management Service (AWS KMS) keys.

Encrypted RDS DB instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications deployed in the AWS Cloud, and to fulfill compliance requirements for encryption at rest. You can enable encryption for an RDS DB instance when you create it, but not after it’s created. However, you can add encryption to an unencrypted RDS DB instance by creating a snapshot of your DB instance, and then creating an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot to get an encrypted copy of your original DB instance.

This pattern uses AWS Config rules to evaluate RDS DB instances and clusters. It applies remediation by using AWS Systems Manager runbooks, which define the actions to be performed on noncompliant Amazon RDS resources, and AWS KMS keys to encrypt the DB snapshots. It then enforces service control policies (SCPs) to prevent the creation of new DB instances and clusters without encryption.

The code for this pattern is provided in GitHub.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Files from the GitHub source code repository for this pattern downloaded to your computer
- An unencrypted RDS DB instance or cluster
- An existing AWS KMS key for encrypting RDS DB instances and clusters
- Access to update the KMS key resource policy
- AWS Config enabled in your AWS account (see Getting Started with AWS Config in the AWS documentation)
Limitations

- You can enable encryption for an RDS DB instance only when you create it, not after it has been created.
- You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance.
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
- Amazon RDS encryption is available for most DB instance classes. For a list of exceptions, see Encrypting Amazon RDS resources in the Amazon RDS documentation.
- To copy an encrypted snapshot from one AWS Region to another, you must specify the KMS key in the destination AWS Region. This is because KMS keys are specific to the AWS Region that they are created in.
- The source snapshot remains encrypted throughout the copy process. Amazon RDS uses envelope encryption to protect data during the copy process. For more information, see Envelope encryption in the AWS KMS documentation.
- You can't unencrypt an encrypted DB instance. However, you can export data from an encrypted DB instance and import the data into an unencrypted DB instance.
- You should delete a KMS key only when you are sure that you don't need to use it any longer. If you aren't sure, consider disabling the KMS key instead of deleting it. You can reenable a disabled KMS key if you need to use it again later, but you cannot recover a deleted KMS key.
- If you don't choose to retain automated backups, your automated backups that are in the same AWS Region as the DB instance are deleted. They can't be recovered after you delete the DB instance.
- Your automated backups are retained for the retention period that is set on the DB instance at the time you delete it. This set retention period occurs whether or not you choose to create a final DB snapshot.
- If automatic remediation is enabled, this solution encrypts all databases that have the same KMS key.

Architecture

The following diagram illustrates the architecture for the AWS CloudFormation implementation. Note that you can also implement this pattern by using the AWS Cloud Development Kit (AWS CDK).
Tools

**Tools**

- **AWS CloudFormation** – AWS CloudFormation helps you automatically set up your AWS resources. It enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).
- **AWS CDK** – The AWS Cloud Development Kit (AWS CDK) is a software development framework for defining your cloud infrastructure in code and provisioning it by using familiar programming languages.

**AWS services and features**

- **AWS Config** – AWS Config keeps track of the configuration of your AWS resources and their relationships to your other resources. It can also evaluate those AWS resources for compliance. This service uses rules that can be configured to evaluate AWS resources against desired configurations. You can use a set of AWS Config managed rules for common compliance scenarios, or you can create your own rules for custom scenarios. When an AWS resource is found to be noncompliant, you can specify a remediation action through an AWS Systems Manager runbook and optionally send an alert through an Amazon Simple Notification Service (Amazon SNS) topic. In other words, you can associate remediation actions with AWS Config rules and choose to run them automatically to address noncompliant resources without manual intervention. If a resource is still noncompliant after automatic remediation, you can set the rule to try automatic remediation again.
- **Amazon RDS** – Amazon Relational Database Service (Amazon RDS) makes it easier to set up, operate, and scale a relational database in the cloud. The basic building block of Amazon RDS is the DB
instance, which is an isolated database environment in the AWS Cloud. Amazon RDS provides a **selection of instance types** that are optimized to fit different relational database use cases. Instance types comprise various combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your database. Each instance type includes several instance sizes, allowing you to scale your database to the requirements of your target workload.

- **AWS KMS** – AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control AWS KMS keys, which encrypt your data. A KMS key is a logical representation of a root key. The KMS key includes metadata, such as the key ID, creation date, description, and key state.

- **IAM** – AWS Identity and Access Management (IAM) controls access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **SCPs** – Service control policies (SCPs) offer central control over the maximum available permissions for all accounts in your organization. SCPs help you ensure that your accounts stay within your organization's access control guidelines. SCPs don't affect users or roles in the management account. They affect only the member accounts in your organization. We strongly recommend that you don't attach SCPs to the root of your organization without thoroughly testing the impact that the policy has on accounts. Instead, create an organizational unit (OU) that you can move your accounts into one at a time, or at least in small numbers, to ensure that you don't inadvertently lock users out of key services.

**Code**

The source code and templates for this pattern are available in a [GitHub repository](https://github.com/). The pattern provides two implementation options: You can deploy an AWS CloudFormation template to create the remediation role that encrypts RDS DB instances and clusters, or use the AWS CDK. The repository has separate folders for these two options.

The **Epics** section provides step-by-step instructions for deploying the CloudFormation template. If you want to use the AWS CDK, follow the instructions in the README.md file in the GitHub repository.

**Epics**

**Create the IAM remediation role and AWS Systems Manager runbook**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the CloudFormation template</td>
<td>Download the unencrypted-to-encrypted-rds.template.json file from the GitHub repository.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
| Create the CloudFormation stack     | 1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation/.  
2. Launch the unencrypted-to-encrypted-rds.template.json template to create a new stack. | DevOps engineer       |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Review CloudFormation parameters and values. | 1. Review stack details and update values based on your environment requirements.  
                                           | 2. Choose Create stack to deploy the template.                               | DevOps engineer |
| Review the resources.                   | When the stack has been created, its status changes to CREATE_COMPLETE. Review the created resources (IAM role, AWS Systems Manager runbook) in the CloudFormation console. | DevOps engineer |

#### Update the AWS KMS key policy

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update your KMS key policy.</td>
<td>1. Make sure that the key alias alias/RDSEncryptionAtRestKMSAlias exists.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>2. The key policy statement should include the IAM remediation role. (Check the resources created by the CloudFormation template you deployed in the previous epic.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. In the following key policy, update the portions that are in bold to match your account and the IAM role that was created.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Sid&quot;: &quot;Allow access through RDS for all principals in the account that are authorized to use RDS&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Effect&quot;: &quot;Allow&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Principal&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;AWS&quot;: &quot;arn:aws:iam: &lt;your-AWS-account-ID&gt;:role/&lt;your-IAM-remediation-role&gt;&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;kms:Encrypt&quot;,</td>
<td></td>
</tr>
</tbody>
</table>

Find and remediate noncompliant resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| View noncompliant resources. | 1. To view a list of noncompliant resources, open the AWS Config console at https://console.aws.amazon.com/config/.  
2. In the navigation pane, choose Rules, and then choose the rds-storage-encrypted rule. | DevOps engineer |

The noncompliant resources listed in the AWS Config console will be instances, not clusters. The remediation automation encrypts instances and clusters, and creates either a newly encrypted instance or a newly created cluster. However, be sure not to simultaneously remediate multiple instances that belong to the same cluster.

Before you remediate any RDS DB instances or volumes, make sure that the RDS DB instance is not in use. Confirm that there are no write operations occurring while the snapshot is being created, to ensure that the snapshot contains the data.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remediate noncompliant resources.</td>
<td>1. When you are ready and the maintenance window is in effect, choose the resource to remediate, and then choose <strong>Remediate</strong>. The <strong>Action status</strong> column should now display <strong>Action execution queued</strong>.&lt;br&gt;2. View the progress and status of the remediation in Systems Manager. Open the AWS Systems Manager console at <a href="https://console.aws.amazon.com/systems-manager/">https://console.aws.amazon.com/systems-manager/</a>. In the navigation pane, choose <strong>Automation</strong>, and then select the execution ID of the corresponding automation to view further details.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Verify that the RDS DB instance is available.</td>
<td>After the automation completes, the newly encrypted RDS DB instance will become available. The encrypted RDS DB instance will have the prefix <code>encrypted</code> followed by the original name. For example, if the unencrypted RDS DB instance name was <code>database-1</code>, the newly encrypted RDS DB instance would be <code>encrypted-database-1</code>.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Terminate the unencrypted instance.</td>
<td>After remediation is complete and the newly encrypted resource has been validated, you can terminate the unencrypted instance. Make sure to confirm that the newly encrypted resource matches the unencrypted resource before you terminate any resources.</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>
Enforce SCPs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enforce SCPs</td>
<td>Enforce SCPs to prevent DB instances and clusters from being created without encryption in the future. Use the rds_encrypted.json file that's provided in the GitHub repository for this purpose, and follow the instructions in the AWS documentation.</td>
<td>Security engineer</td>
</tr>
</tbody>
</table>

Related resources

References

- Setting up AWS Config
- AWS Config custom rules
- AWS KMS concepts
- AWS Systems Manager documents
- Service control policies

Tools

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)

Guides and patterns

- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config

Additional information

Best practices

- Enable data encryption both at rest and in transit.
- Enable AWS Config in all accounts and AWS Regions.
- Record configuration changes to all resource types.
- Rotate your IAM credentials regularly.
- Leverage tagging for AWS Config, which makes is easier to manage, search for, and filter resources.

FAQ

Q. How does AWS Config work?

A. When you turn on AWS Config, it first discovers the supported AWS resources that exist in your account and generates a configuration item for each resource. AWS Config also generates configuration items when the configuration of a resource changes, and it maintains historical records of the
configuration items of your resources from the time you start the configuration recorder. By default, AWS Config creates configuration items for every supported resource in the AWS Region. If you don’t want AWS Config to create configuration items for all supported resources, you can specify the resource types that you want it to track.

**Q. How are AWS Config and AWS Config rules related to AWS Security Hub?**

**A.** AWS Security Hub is a security and compliance service that provides security and compliance posture management as a service. It uses AWS Config and AWS Config rules as its primary mechanism to evaluate the configuration of AWS resources. AWS Config rules can also be used to evaluate resource configuration directly. Config rules are also used by other AWS services, such as AWS Control Tower and AWS Firewall Manager.

### Automatically rotate IAM user access keys at scale with AWS Organizations and AWS Secrets Manager

*Created by Tracy Pierce (AWS) and Laura Seletos (AWS)*

| Environment: | PoC or pilot | Technologies: | Security, identity, compliance | AWS services: | AWS CloudFormation; Amazon CloudWatch Events; AWS Identity and Access Management; AWS Lambda; AWS Organizations; Amazon S3; Amazon SES; AWS Secrets Manager |

### Summary

**Access keys** are long-term credentials for an AWS Identity and Access Management (IAM) user or the Amazon Web Services (AWS) account root user. Regularly rotating your IAM credentials helps prevent a compromised set of IAM access keys from accessing components in your AWS account. Rotating IAM credentials is also an important part of security best practices in IAM.

This pattern helps you to automatically rotate IAM access keys by using two AWS CloudFormation templates. The main AWS CloudFormation template is deployed to a central AWS account and the assumed role AWS CloudFormation template is deployed to member accounts that you identify.

This centralized solution dynamically identifies all AWS account IDs in your organization in AWS Organizations and uses an assumed IAM role to locally run an AWS Lambda function across multiple accounts in your organization. By using this pattern, you can ensure that the following actions are automatically performed:

- New IAM access keys are generated when existing access keys are 90 days old.
- The new access keys are stored as a secret in AWS Secrets Manager. A resource-based policy allows only the specified IAM principal to access and retrieve the secret.
- The account owner of the new access keys receives a notification email.
- The previous access keys are deactivated at 100 days old.
- The previous access keys are then deleted at 110 days old.
- A centralized email notification is sent to the AWS account owner.
Lambda functions and Amazon CloudWatch automatically perform these actions. You can then retrieve the new access key pair and replace them in your code or applications.

**Prerequisites and limitations**

**Prerequisites**

- At least one active AWS account.
  - If you only have one account, then you can deploy both AWS CloudFormation templates as stacks.
  - If you have more than one account, then you can deploy the main AWS CloudFormation template in your central account as a stack and deploy the assume role AWS CloudFormation template with AWS CloudFormation StackSets to all required organizational units (OUs).

- AWS Organizations, configured and set up.
  - Your central management account must have permission to query AWS Organizations. For more information about this, see AWS Organizations and service-linked roles in the AWS Organizations documentation.
  - An IAM principal with permissions to launch the AWS CloudFormation template and associated resources. For more information about this, see Grant self-managed permissions in the AWS CloudFormation documentation.
  - An existing Amazon Simple Storage Service (Amazon S3) bucket to deploy the resources.
  - Amazon Simple Email Service (Amazon SES) must be out of the sandbox. For more information, see Moving out of the Amazon SES sandbox in the Amazon SES documentation.

**Architecture**

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

The diagram shows the following workflow:
1. A CloudWatch event initiates a Lambda function every 24 hours.
2. The Lambda function queries Organizations for a list of all AWS account IDs, account names, and account emails.
3. The Lambda function initiates a Lambda function for each AWS account ID and passes it the metadata for additional processing.
4. This second Lambda function then uses an assumed IAM role to access the AWS account ID. The Lambda script runs an audit against all users and their IAM access keys in the account.
5. If the IAM access key's age has not exceeded the best practice threshold, then the Lambda function takes no further action.
6. If the IAM access key's age has exceeded the best practice threshold, then the Lambda function determines which rotation action to perform.
7. When action is required, the Lambda function creates and updates a secret in AWS Secrets Manager if a new key is generated. A resource-based policy is also created that only allows the specified IAM principal to access and retrieve the secret.
8. A third Lambda function is initiated to notify the account's owner of the rotation activity. This Lambda function receives the AWS account ID, account name, account email, and the rotation actions that were performed.
9. The function queries the deployment S3 bucket for an email template and dynamically updates it with the relevant activity metadata. The email is then sent to the account owner's email address.

**Note:** This solution supports can be run in Audit Mode. In Audit Mode, no IAM access keys are modified but an email is still sent to notify end users.

**Technology stack**

- Amazon CloudWatch
- IAM
- AWS Lambda
- Organizations
- Amazon S3
- Secrets Manager
- Amazon SES
- Amazon Simple Notification Service (Amazon SNS)

**Automation and scale**

Using Organizations, you can leverage AWS CloudFormation StackSets to deploy the ASA-iam-key-auto-rotation-iam-assumed-roles.yaml AWS CloudFormation template in multiple accounts. The ASA-iam-key-auto-rotation-and-notifier-solution.yaml AWS CloudFormation template can only be deployed to the central management account and by using an AWS CloudFormation stack.

**Tools**

- **Amazon CloudWatch** – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.
- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few
requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage. Organizations includes account management and consolidated billing capabilities that enable you to better meet the budgetary, security, and compliance needs of your business.

- **AWS Secrets Manager** – AWS Secrets Manager is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

- **Amazon SES** – Amazon Simple Email Service (Amazon SES) is an email service that provides an easy, cost-effective way for you to send and receive email using your own email addresses and domains. For example, you can use Amazon SES to send marketing emails such as special offers, transactional emails such as order confirmations, and other types of correspondence.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

**Code**

The required AWS CloudFormation templates, Python scripts, and runbook documentation are available in the GitHub IAM key rotation repository.

**Epics**

**Deploy the solution**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose your deployment S3</td>
<td>Sign in to the AWS Management Console for your central account, open the</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>bucket.</td>
<td>Amazon S3 console, and then choose the S3 bucket for your deployment.</td>
<td></td>
</tr>
<tr>
<td>Clone the repository.</td>
<td>Clone the GitHub IAM key rotation repository to your local desktop.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Upload the files to the S3</td>
<td>Upload the cloned files to your S3 bucket. Use the following default folder</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>bucket.</td>
<td>structure to copy and paste all cloned files and directories: asa/asa-iam-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rotation</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can customize this folder structure in the AWS CloudFormation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>templates.</td>
<td></td>
</tr>
<tr>
<td>Modify the email template.</td>
<td>Modify the iam-auto-key-rotation-enforcement.html email</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>template according to your requirements. <strong>Note:</strong> Replace [Department Name Here] with your department’s name.</td>
<td>Cloud architect</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Launch the first AWS CloudFormation template.</td>
<td>Launch the ASA-iam-key-auto-rotation-iam-assumed-roles.yaml template and follow the steps on the AWS CloudFormation console. For more information about this, see Selecting a stack template in the AWS CloudFormation documentation. <strong>Important:</strong> Make sure that you provided the following parameters for the AWS CloudFormation template:</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>• The name of the deployment S3 bucket that contains your Lambda code</td>
<td>• The S3 bucket’s prefix</td>
<td></td>
</tr>
<tr>
<td>• Your organization’s unique identifier (ID)</td>
<td>• The AWS account ID where the main template will be deployed</td>
<td></td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Launch the second AWS CloudFormation template. | Launch the ASA-iam-key-auto-rotation-and-notifier-solution.yaml template and follow the steps on the AWS CloudFormation console. For more information about this, see Selecting a stack template in the AWS CloudFormation documentation. **Important:** Make sure that you provide the following parameters for the AWS CloudFormation template:  
- The name of the deployment S3 bucket that contains your Lambda code.  
- The S3 bucket's prefix.  
- Dry Run Flag to turn on Audit or Enforcement mode  
- Your AWS Organization ID. This is the unique ID of your organization, which begins with o- and is followed by 10 to 32 lowercase letters or digits  
- A valid email address that can be used in the sent from section of email notifications | Cloud architect |

Automatically validate and deploy IAM policies and roles in an AWS account by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros

*Created by Helton Henrique Ribeiro (AWS) and Guilherme Simoes (AWS)*

**Code repository:**  
- IAM roles pipeline

**AWS services:**  
- AWS CloudFormation; AWS CodeBuild; AWS CodeCommit; AWS CodePipeline; AWS Lambda; AWS SAM

**Environment:**  
- PoC or pilot

**Technologies:**  
- Security, identity, compliance; DevOps
Summary

This pattern describes the steps and provides code to create a deployment pipeline that allows your development teams to create AWS Identity and Access Management (IAM) policies and roles in your Amazon Web Services (AWS) accounts. This approach helps your organization reduce overhead for your operational teams and speed up the deployment process. It also helps your developers to create IAM roles and policies that are compatible with your existing governance and security controls.

This pattern's approach uses AWS Identity and Access Management Access Analyzer to validate the IAM policies that you want to attach to IAM roles and uses AWS CloudFormation to deploy the IAM roles. However, instead of directly editing the AWS CloudFormation template file, your development team creates JSON-formatted IAM policies and roles. An AWS CloudFormation macro transforms these JSON-formatted policy files into AWS CloudFormation IAM resource types before beginning the deployment.

The deployment pipeline (RolesPipeline) has source, validation, and deployment stages. During the source stage, your development team pushes the JSON files that contain the definition of the IAM roles and policies to an AWS CodeCommit repository. AWS CodeBuild then runs a script to validate those files and copies them to an Amazon Simple Storage Service (Amazon S3) bucket. Because your development teams don't have direct access to the AWS CloudFormation template file stored in a separate S3 bucket, they must follow the JSON file creation and validation process.

Finally, during the deployment phase, AWS CodeDeploy uses an AWS CloudFormation stack to update or delete the IAM policies and roles in an account.

**Important**: This pattern’s workflow is a proof of concept (POC) and we recommend that you only use it in a test environment. If you want to use this pattern’s approach in a production environment, see Security best practices in IAM in the IAM documentation and make the required changes to your IAM roles and AWS services.

Prerequisites and limitations

**Prerequisites**

- An active AWS account.
- A new or existing S3 bucket for the RolesPipeline pipeline. Make sure that your IAM user has the required permissions to upload objects to this bucket.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
- AWS Serverless Application Model (AWS SAM) CLI, installed and configured. For more information about this, see Installing the AWS SAM CLI in the AWS SAM documentation.
- Python 3, installed on your local machine. For more information about this, see the Python documentation.
- A Git client, installed and configured.
- The GitHub IAM roles pipeline repository, cloned to your local machine.
- Existing JSON-formatted IAM policies and roles. For more information about this, see the ReadMe file in the Github IAM roles pipeline repository.
- Your developer team must not have permissions to edit this solution’s AWS CodePipeline, CodeBuild, and CodeDeploy resources.

**Limitations**
• This pattern’s workflow is a proof of concept (POC) and we recommend that you only use it in a test environment. If you want to use this pattern’s approach in a production environment, see Security best practices in IAM in the IAM documentation and make the required changes to your IAM roles and AWS services.

Architecture

The following diagram shows you how to automatically validate and deploy IAM roles and policies to an account by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros.

The diagram shows the following workflow:

1. A developer writes JSON files that contain the definitions for the IAM policies and roles. The developer pushes the code to a CodeCommit repository and CodePipeline then initiates the RolesPipeline pipeline.
2. CodeBuild validates the JSON files by using IAM Access Analyzer. If there are any security or error-related findings, the deployment process is stopped.
3. If there are no security or error-related findings, the JSON files are sent to the RolesBucket S3 bucket.
4. An AWS CloudFormation macro implemented as an AWS Lambda function then reads the JSON files from the RolesBucket bucket and transforms them into AWS CloudFormation IAM resources types.
5. A predefined AWS CloudFormation stack installs, updates, or deletes the IAM policies and roles in the account.

Automation and scale

AWS CloudFormation templates that automatically deploy this pattern are provided in the GitHub IAM roles pipeline repository.
Tools

- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS Identity and Access Management Access Analyzer** – IAM Access Analyzer helps you identify the resources in your organization and accounts, such as S3 buckets or IAM roles, that are shared with an external entity. This helps you to identify unintended access to your resources and data.

- **AWS SAM** – AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS. AWS SAM provides you with a template specification to define your serverless application, and a CLI tool.

Code

The source code and templates for this pattern are available in the GitHub IAM roles pipeline repository.

Epics

Clone the repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the sample repository</td>
<td>Clone the GitHub IAM roles pipeline repository to your local machine.</td>
<td>App developer, General AWS</td>
</tr>
</tbody>
</table>

Deploy the RolesPipeline pipeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the pipeline.</td>
<td>1. Navigate to the directory that contains the cloned repository.</td>
<td>App developer, General AWS</td>
</tr>
<tr>
<td></td>
<td>2. Run the make deploy bucket=&lt;bucket_name&gt; command. <strong>Important:</strong> you must replace &lt;bucket_name&gt; with the bucket name for your existing S3 bucket.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Run the <code>aws codepipeline get-pipeline -name RolesPipeline</code> command to check if your deployment is successful.</td>
<td></td>
</tr>
</tbody>
</table>
### Clone the pipeline's repository.

1. The RolesPipeline AWS CloudFormation stack creates the `roles-pipeline-repo` CodeCommit repository.
2. Sign in to the AWS Management Console, open the AWS CodeCommit console, and then copy the CodeCommit repository’s URL to clone it to your local machine. For more information about this, see [Connect to an AWS CodeCommit repository](https://docs.aws.amazon.com/codecommit/latest/userguide/reference-Connecting-to-an-AWS-CodeCommit-repository.html) in the AWS CodeCommit documentation.

**Skills required:** App developer, General AWS

### Test the RolesPipeline pipeline

1. Create JSON files for your IAM policies and roles. You can use the samples in the `role-example` directory from the GitHub IAM roles pipeline repository.
2. Define your IAM policies and roles with the required configurations. **Important:** Make sure that you follow the format described in the `ReadMe` file from the GitHub IAM roles pipeline repository.
3. Push the modifications into the `roles-pipeline-repo` CodeCommit repository.
4. Verify the implementation of the RolesPipeline pipeline.
5. Make sure that the IAM policies and roles are correctly deployed in the account.
6. Validate if there is a permissions boundary associated to the IAM policies or roles. For more information about this, see [Permissions boundaries for IAM entities](https://docs.aws.amazon.com/IAM/latest/UserGuide/permissions-boundaries.html) in the IAM documentation.

**Skills required:** App developer, General AWS
## Test the RolesPipeline pipeline with invalid IAM policies and roles.

1. Modify the `roles-pipeline-repo` CodeCommit repository and include invalid IAM roles or policies. For example, you can use an action that doesn't exist or an invalid IAM policy version.
2. Verify the pipeline implementation. IAM Access Analyzer stops the pipeline during the validation stage if it detects invalid IAM policies or roles.

### Skills required

App developer, General AWS

## Clean up your resources

### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare for cleanup. Empty the S3 buckets and then run the destroy command.</td>
<td>App developer, General AWS</td>
</tr>
<tr>
<td>Delete the RolesStack stack.</td>
<td>App developer, General AWS</td>
</tr>
<tr>
<td>Delete the RolesPipeline stack. To delete the RolesPipeline AWS CloudFormation stack, follow the instructions from the ReadMe file in the Github IAM roles pipeline repository.</td>
<td>App developer, General AWS</td>
</tr>
</tbody>
</table>

## Related resources

- [IAM Access Analyzer - Policy validation](https://aws.amazon.com/blogs/aws/) (AWS News Blog)
- [Using AWS CloudFormation macros to perform custom processing on templates](https://docs.aws.amazon.com/cdk/latest/guide/macros.html) (AWS CloudFormation documentation)
- [Building Lambda functions with Python](https://docs.aws.amazon.com/lambda/latest/dg/python-howto.html) (AWS Lambda documentation)
Bidirectionally integrate AWS Security Hub with Jira software

Created by Joaquin Manuel Rinaudo (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: PoC or pilot</th>
<th>Technologies: Security, identity, compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Hub to JIRA Integration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Workload: All other workloads</th>
<th>AWS services: AWS Lambda; AWS Security Hub; Amazon CloudWatch</th>
</tr>
</thead>
</table>

Summary

This solution supports a bidirectional integration between AWS Security Hub and Jira. Using this solution, you can automatically and manually create and update JIRA tickets from Security Hub findings. Security teams can use this integration to notify developer teams of severe security findings that require action.

The solution allows you to:

- Select which Security Hub controls automatically create or update tickets in Jira.
- In the Security Hub console, use Security Hub custom actions to manually escalate tickets in Jira.
- Automatically assign tickets in Jira based on the AWS account tags defined in AWS Organizations. If this tag is not defined, a default assignee is used.
- Automatically suppress Security Hub findings that are marked as false positive or accepted risk in Jira.
- Automatically close a Jira ticket when its related finding is archived in Security Hub.
- Reopen Jira tickets when Security Hub findings reoccur.

Jira workflow

The solution uses a custom Jira workflow that allows developers to manage and document risks. As the issue moves through the workflow, bidirectional integration ensures that the status of the Jira ticket and Security Hub finding is synchronized across the workflows in both services. This workflow is a derivative of SecDevOps Risk Workflow by Dinis Cruz, licensed under CC BY 4.0. We recommend adding a Jira workflow condition so that only members of your security team can change the ticket status.
For an example of a Jira ticket automatically generated by this solution, see the Additional information (p. 1637) section of this pattern.

Prerequisites and limitations

**Prerequisites**

- If you want to deploy this solution across a multi-account AWS environment:
  - Your multi-account environment is active and managed by AWS Organizations.
  - Security Hub is enabled on your AWS accounts.
  - In AWS Organizations, you have designated a Security Hub administrator account.
  - You have a cross-account IAM role with `AWSOrganizationsReadOnlyAccess` permissions to the AWS Organizations management account.
  - (Optional) You have tagged your AWS accounts with `SecurityContactID`. This tag is used to assign Jira tickets to the defined security contacts.
- If you want to deploy this solution within a single AWS account:
  - You have an active AWS account.
  - Security Hub is enabled on your AWS account.
  - A Jira Server instance **Important**: This solution supports use of Jira Cloud. However, Jira Cloud does not support importing XML workflows, so you need to manually re-create the workflow in Jira.
  - Administrator permissions in Jira
  - One of the following Jira tokens:
    - For Jira Enterprise, a personal access token (PAT). For more information, see Using Personal Access Tokens (Atlassian support).
    - For Jira Cloud, a Jira API token. For more information, see Manage API tokens (Atlassian support).

**Architecture**

This section illustrates the architecture of the solution in various scenarios, such as when the developer and security engineer decide to accept the risk or decide to fix the issue.

*Scenario 1: Developer addresses the issue*
1. Security Hub generates a finding against a specified security control, such as those in the AWS Foundational Security Best Practices standard.

2. An Amazon CloudWatch event associated with the finding and the `CreateJIRA` action initiates an AWS Lambda function.

3. The Lambda function uses its configuration file and the finding's `GeneratorId` field to evaluate whether it should escalate the finding.

4. The Lambda function determines the finding should be escalated, it obtains the `SecurityContactID` account tag from AWS Organizations in the AWS management account. This ID is associated with the developer and is used as the assignee ID for the Jira ticket.

5. The Lambda function uses the credentials stored in AWS Secrets Manager to create a ticket in Jira. Jira notifies the developer.

6. The developer addresses the underlying security finding and, in Jira, changes the status of the ticket to TEST FIX.

7. Security Hub updates the finding as ARCHIVED, and a new event is generated. This event causes the Lambda function to automatically close the Jira ticket.

Scenario 2: Developer decides to accept the risk

1. Security Hub generates a finding against a specified security control, such as those in the AWS Foundational Security Best Practices standard.

2. A CloudWatch event associated with the finding and the `CreateJIRA` action initiates a Lambda function.

3. The Lambda function uses its configuration file and the finding's `GeneratorId` field to evaluate whether it should escalate the finding.

4. The Lambda function determines the finding should be escalated, it obtains the `SecurityContactID` account tag from AWS Organizations in the AWS management account. This ID is associated with the developer and is used as the assignee ID for the Jira ticket.
5. The Lambda function uses the credentials stored in Secrets Manager to create a ticket in Jira. Jira notifies the developer.

6. The developer decides to accept the risk and, in Jira, changes the status of the ticket to **AWAITING RISK ACCEPTANCE**.

7. The security engineer reviews the request and finds the business justification appropriate. The security engineer changes the status of the Jira ticket to **ACCEPTED RISK**. This closes the Jira ticket.

8. A CloudWatch daily event initiates the refresh Lambda function, which identifies closed JIRA tickets and updates their related Security Hub findings as **SUPPRESSED**.

---

**Tools**

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **Amazon CloudWatch Events** – Amazon CloudWatch Events monitors system events for your AWS resources. Using rules, you can match events and route them to target functions or streams. You can also schedule automated actions by using expressions.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

- **AWS Security Hub** – AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.
AWS Prescriptive Guidance Patterns

Epics

- **AWS Secrets Manager** – AWS Secrets Manager is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.

**Code repository**

The code for this pattern is available on GitHub, in the `aws-securityhub-jira-software-integration` repository. It includes the sample code and Jira workflow for this solution.

**Epics**

**Configure Jira**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import the workflow.</td>
<td>As an administrator in Jira, import the <code>issue-workflow.xml</code> file to your Jira Server instance. This file can be found in the <code>aws-securityhub-jira-software-integration</code> repository in GitHub. For instructions, see Using XML to create a workflow (Jira documentation).</td>
<td>Jira administrator</td>
</tr>
</tbody>
</table>
| Activate and assign the workflow. | Workflows are inactive until you assign them to a workflow scheme. You then assign the workflow scheme to a project.  
1. For your project, make sure you have identified an issue type scheme for the project. You can create a new issue type or select from an existing one, such as Bug.  
2. Assign the imported workflow to a workflow scheme according to the instructions in Activate a workflow (Jira documentation).  
3. Assign the workflow scheme to a project according to the instructions in Associate a workflow scheme with a project (Jira documentation). | Jira administrator      |

**Set up the solution parameters**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the solution parameters.</td>
<td>1. In the <code>conf</code> folder, open <code>params_prod.shfile</code>.</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>2.</td>
<td>Provide values for the following parameters:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>ORG_ACCOUNT_ID</strong> – The account ID for your AWS Organizations management account. The solution reads account tags and assigns tickets to the specific security contacts defined in those AWS account tags.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>ORG_ROLE</strong> – The name of the IAM role used to access the AWS Organization management account. This role must have OrganizationsReadOnlyAccess permissions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>EXTERNAL_ID</strong> – An optional parameter if you are using an external ID to assume the IAM role defined in <strong>ORG_ROLE</strong>. For more information, see How to use an external ID (IAM documentation).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>JIRA_DEFAULT_ASSIGNEE</strong> – This is the Jira ID for default assignee for all Security Issues. This default assigned is used in case account is not tagged properly or role cannot be assumed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>JIRA_INSTANCE</strong> – The HTTPS address for your Jira server in the following format: team-&lt;team-id&gt;.atlassian.net/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>JIRA_PROJECT_KEY</strong> – The name of the Jira project key used to create tickets, such as SEC or TEST. This project must already exist in Jira.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>ISSUE_TYPE</strong> – The name of the issue type scheme assigned to the project in Jira, such as Bug or Security Issue.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>REGIONS</strong> – List of AWS Region codes where you want to deploy this solution, such as eu-west-1.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>2.</td>
<td>In the Security Hub navigation pane, choose Findings.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Choose the finding title.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Choose the finding ID. This displays the complete JSON for the finding.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>In the JSON, copy the string in the GeneratorId field. This value is in AWS Security Finding Format (ASFF). For example, aws-foundational-security-best-practices/v/1.0.0/S3.1 corresponds to findings from the security control S3.1 S3 Block Public Access setting should be enabled.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Repeat these steps until you have copied all of the GeneratorID values for any findings you want to automate.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Save and close the solution parameter file.</td>
<td></td>
</tr>
</tbody>
</table>
### Add the findings to the configuration file.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>In <strong>src/code</strong>, open the config.jsonconfig file.</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>2.</td>
<td>Paste the GeneratorID values you retrieved in the previous story into the default parameter, and use commas to separate each ID.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Save and close the configuration file.</td>
<td></td>
</tr>
</tbody>
</table>

The following code example shows automating the aws-foundational-security-best-practices/v/1.0.0/SNS.1 and aws-foundational-security-best-practices/v/1.0.0/S3.1 findings.

```json
{
  "Controls": {
    "eu-west-1": [
      "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.22"
    ],
    "default": [
      "aws-foundational-security-best-practices/v/1.0.0/SNS.1,
      "aws-foundational-security-best-practices/v/1.0.0/S3.1"
    ]
  }
}
```

**Note:** You can choose to automate different findings for each AWS Region. A good practice to help prevent duplicated findings is to select a single Region to automate creation of IAM-related controls.

### Deploy the integration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>In a command line terminal, enter the following command:</td>
<td>AWS systems administrator</td>
</tr>
</tbody>
</table>

1634
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>./deploy.sh prod</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Upload Jira credentials to AWS Secrets Manager.</td>
<td>1. Open the Secrets Manager console at <a href="https://console.aws.amazon.com/secretsmanager/">https://console.aws.amazon.com/secretsmanager/</a>. 2. Under Secrets, choose <strong>Store a new secret</strong>. 3. For <strong>Secret type</strong>, choose <strong>Other type of secret</strong>. 4. If you are using Jira Enterprise, for <strong>Key/value pairs</strong>, do the following:  - In the first row, enter <code>auth</code> in the key box, and then enter <code>token_auth</code> in the value box.  - Add a second row, enter <code>token</code> in the key box, and then enter your personal access token in the value box.  If you are using Jira Cloud, for <strong>Key/value pairs</strong>, do the following:  - <strong>Key/value pairs</strong>, do the following:  - In the first row, enter <code>auth</code> in the key box, and then enter <code>basic_auth</code> in the value box.  - Add a second row, enter <code>token</code> in the key box, and then enter your API token in the value box.  - Add a third row, enter <code>email</code> in the key box, and then enter your email address in the value box. 5. Choose <strong>Next</strong>. 6. For <strong>Secret name</strong>, enter <code>Jira-Token</code>, and then at the bottom of the page, choose <strong>Next</strong>. 7. On the Secret rotation page, keep <strong>Disable automatic rotation</strong>, and then at the bottom of the page, choose <strong>Next</strong>.</td>
<td><strong>AWS systems administrator</strong></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>8. On the Review page, review the secret details, and then choose Store.</td>
<td>Create the Security Hub custom action.</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>1. For each AWS Region, in the AWS Command Line Interface (AWS CLI), use the <code>create-action-target</code> command to create a Security Hub custom action named <code>CreateJiraIssue</code>.</td>
<td>AWS systems administrator</td>
<td></td>
</tr>
<tr>
<td>4. In the list of findings, select the findings you want to escalate.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. In the Actions menu, choose <code>CreateJiraIssue</code>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- AWS Service Management Connector for Jira Service Management
- AWS Foundational Security Best Practices standard

**Additional information**

**Example of a Jira ticket**

When a specified Security Hub finding occurs, this solution automatically creates a Jira ticket. The ticket includes the following information:

- **Title** – The title identifies the security issue in the following format:

  AWS Security Issue :: <AWS account ID> :: <Security Hub finding title>

- **Description** – The description section of the ticket describes the security control associated with the finding, includes a link to the finding in the Security Hub console, and provides a short description of how to handle the security issue in the Jira workflow.
The following is an example of an automatically generated Jira ticket.

<table>
<thead>
<tr>
<th>Title</th>
<th>AWS Security Issue :: 012345678912 :: Lambda.1 Lambda function policies should prohibit public access.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>We detected a security finding within the AWS account 012345678912 you are responsible for.</td>
</tr>
<tr>
<td></td>
<td>This control checks whether the AWS Lambda function policy attached to the Lambda resource prohibits public access. If the Lambda function policy allows public access, the control fails.</td>
</tr>
<tr>
<td></td>
<td>&lt;Link to Security Hub finding&gt;</td>
</tr>
<tr>
<td>What do I need to do with the ticket?</td>
<td>• Access the account and verify the configuration. Acknowledge working on ticket by moving it to &quot;Allocated for Fix&quot;. Once fixed, moved to test fix so Security validates the issue is addressed.</td>
</tr>
<tr>
<td></td>
<td>• If you think risk should be accepted, move it to &quot;Awaiting Risk acceptance&quot;. This will require review by a Security engineer.</td>
</tr>
<tr>
<td></td>
<td>• If you think is a false positive, transition it to &quot;Mark as False Positive&quot;. This will get reviewed by a Security engineer and reopened/closed accordingly.</td>
</tr>
</tbody>
</table>

Attachments

attachment.zip

Centralized logging and multiple-account security guardrails

*Created by Ankush Verma (AWS) and Tracy (Pierce) Hickey (AWS)*

| Environment: Production | Technologies: Security, identity, compliance; Management & governance | AWS services: AWS CloudFormation; AWS Config; Amazon CloudWatch; AWS CodePipeline; Amazon GuardDuty; AWS Lambda; Amazon Macie; AWS Security Hub; Amazon S3 |
Summary

The approach covered in this pattern is suitable for customers who have multiple Amazon Web Services (AWS) accounts with AWS Organizations and are now encountering challenges when using AWS Control Tower, a landing zone, or account vending machine services to set up baseline guardrails in their accounts.

This pattern demonstrates the use of a streamlined multiple-account architecture to set up centralized logging and standardized security controls in a well-structured manner. With the help of AWS CloudFormation templates, AWS CodePipeline, and automation scripts, this setup is deployed in all accounts that belong to an organization.

The multiple-account architecture includes the following accounts:

- **Centralized logging account** – The account where all the virtual private cloud (VPC) flows logs, AWS CloudTrail logs, the AWS Config log, and all the logs of Amazon CloudWatch Logs (using subscriptions) from all the other accounts are stored.

- **Parent security account** – The account to serve as the parent account for the following security services that manage across multiple accounts.
  - Amazon GuardDuty
  - AWS Security Hub
  - Amazon Macie
  - Amazon Detective

- **Child accounts** – The other accounts in the organization. These accounts store all the useful logs in the centralized logging account. The child accounts join the parent security account as members for the security services.

After you launch the CloudFormation template (attached), it provisions three Amazon Simple Storage Service (Amazon S3) buckets in the centralized logging account. One bucket is used to store all AWS related logs (such as logs from VPC Flow Logs, CloudTrail, and AWS Config) from all the accounts. The second bucket is for storing the CloudFormation templates from all the accounts. The third bucket is for storing Amazon S3 access logs.

A separate CloudFormation template creates the pipeline that uses AWS CodeCommit. After the updated code is pushed to the CodeCommit repository, it takes care of launching resources and setting up security services in all the accounts. For more information about the file structure of the files that will be uploaded to the CodeCommit repository, see the README.md file (attached).

Prerequisites and limitations

**Prerequisites**

- An AWS Organizations organization ID, with all accounts joined to the same organization.
- An active email address to receive Amazon Simple Notification Service (Amazon SNS) notifications.
- Confirmed quotas for Amazon Simple Storage Service (Amazon S3) buckets in each of your accounts. By default, each account has 100 S3 buckets. If you require additional buckets, request a quota increase before you deploy this solution.

**Limitations**

All the accounts should be the part of the same organization. If you are not using AWS Organizations, you must modify certain policies, such as the S3 bucket policy, to allow access from the AWS Identity and Access Management (IAM) roles for each account.
Note: While the solution is being deployed, you must confirm the Amazon SNS subscription. The confirmation message is sent to the email address that you provide during the deployment process. This will initiate a few email alert messages to this email address, because these alarms are initiated whenever IAM role policies are created or modified in the account. During the deployment process, you can ignore these alert messages.

Architecture

Target technology stack

- Amazon CloudWatch alarms and logs
- AWS CodeCommit repository
- AWS CodePipeline
- AWS Config
- Amazon Detective
- Amazon GuardDuty
- IAM roles and permissions
- Amazon Macie
- S3 buckets
- AWS Security Hub
- Amazon SNS

Target architecture
1. Other accounts registered as child accounts of the parent security account for the security services
2. Security findings from all the child accounts, including the parent account

Resources

The following resources are provisioned automatically when the updated code is pushed to the CodeCommit repository in each account and AWS Region.

*CloudFormation stack 1 – Logging parent stack*
- Nested stack 1 – Standard IAM roles and policies
- Nested stack 2 – AWS Config setup in the account
- Nested stack 3 – CloudWatch alarms
  - SecurityGroupChangesAlarm
  - UnauthorizedAttemptAlarm
  - RootActivityAlarm
  - NetworkAclChangesAlarm
  - IAMUserManagementAlarm
  - IAMPolicyChangesAlarm
  - CloudTrailChangeAlarm
  - IAMCreateAccessKeyAlarm
  - Metric filters for creating metrics from CloudTrail logs and using them for alarms
- SNS topic

*CloudFormation stack 2 – Parent guardrail stack*
- Nested stack 1 – AWS Lambda function for setting up the account password policy
- Nested stack 2 – Basic AWS Config rules
  - CIS-SecurityGroupsMustRestrictSshTraffic
  - OpenSecurityGroupRuleCheck along with the Lambda function for security group rule evaluation
  - check-ec2-for-required-tag
  - check-for-unrestricted-ports

CloudFormation stack 3 – CloudWatch logs export

- Exporting CloudWatch logs from log groups to Amazon S3 using an Amazon Kinesis subscription

Tools

- **AWS CloudFormation** – AWS CloudFormation uses templates to model and provision, in an automated and secure manner, all the resources needed for your applications across all AWS Regions and accounts.

- **Amazon CloudWatch** – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

- **AWS CodeCommit** – AWS CodeCommit is a version-control service hosted by AWS. You can use CodeCommit to privately store and manage assets (such as documents, source code, and binary files) in the cloud.

- **AWS CodePipeline** – AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software.

- **AWS Config** – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.

- **Amazon Detective** – Amazon Detective is used to analyze, investigate, and quickly identify the root cause of security findings or suspicious activities. Detective automatically collects log data from your AWS resources. It then uses machine learning, statistical analysis, and graph theory to help you visualize and conduct faster and more efficient security investigations.

- **Amazon GuardDuty** – Amazon GuardDuty is a continuous security monitoring service that analyzes and processes the flow logs, CloudTrail management event logs, CloudTrail data event logs, and Domain Name System (DNS) logs. It uses threat intelligence feeds, such as lists of malicious IP addresses and domains, and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment.

- **AWS Identity and Access Management** – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

- **Amazon Macie** – Amazon Macie automates the discovery of sensitive data, such as personally identifiable information (PII) and financial data, to provide you with a better understanding of the data that your organization stores in Amazon S3.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

- **AWS Security Hub** – AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security standards and best practices.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).
## Epics

### Step 1: Set up the IAM roles in all the accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the Childaccount IAM role All Accounts CloudFormation template to create the IAM role in the us-east-1 Region.</td>
<td>To create the required IAM role, permissions, you must manually launch this template in each account, one by one (centralized logging account, parent security account, and all other AWS accounts in the organization) in the us-east-1 Region. The Childaccount IAM role All Accounts.yaml template is in the /templates/initial_deployment_templates directory of the package. The IAM role is used when making API calls for provisioning and setting up the rest of the architecture. Make sure that the name of the IAM role that is passed as a parameter is consistent across all the accounts.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>In the template parameters, provide the name of the IAM role.</td>
<td>Provide the IAM role that CodeBuild, in the parent security account, can assume in all other child accounts. The default role name is security_execute_child_stack_role.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>In the parameters, provide the account ID for the parent security account.</td>
<td>The parent security account is the account where CodeBuild runs.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

### Step 2: Set up S3 buckets in the centralized logging account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the centralized logging account, in us-east-1, launch the S3Buckets-Centralized-LoggingAccount.yaml CloudFormation template.</td>
<td>To create the S3 buckets in the centralized logging account, launch the S3Buckets-Centralized-LoggingAccount.yaml. The template is in the /templates/initial_deployment_templates directory of the package. The S3 buckets will store all the logs, templates, and Amazon S3 access logs. Make a note of</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>the all the S3 bucket names, which you will use to modify the parameter files in the following steps.</td>
<td></td>
</tr>
<tr>
<td>In the template parameters, provide the name of the S3 bucket for AWS logs storage.</td>
<td>Enter a name for the S3 Bucket Name for Centralized Logging in Logging Account parameter. This bucket acts as centralized location to store AWS logs, such as flow logs and CloudTrail logs, from all the accounts. Make a note of both the bucket name and the Amazon Resource Name (ARN).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the name of the S3 bucket for storing access logs.</td>
<td>Enter an S3 bucket name for the S3 Bucket Name for Access Logs in Logging Account parameter. This S3 bucket stores access logs for Amazon S3.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the name of the S3 bucket for storing templates.</td>
<td>Enter an S3 bucket name in the S3 Bucket Name for CloudFormation Template storage in Logging Account parameter.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the organization ID.</td>
<td>To provide access to S3 buckets within the organization, enter the ID for the organization in the Organization Id for Non-AMS accounts parameter.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Step 3: Deploy the CI/CD infrastructure in the parent security account**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the security-guard-rails-codepipeline-Centralized-SecurityAccount.yml CloudFormation template.</td>
<td>To deploy the CI/CD pipeline, manually launch the security-guard-rails-codepipeline-Centralized-SecurityAccount.yml template in the parent security account in us-east-1. The template is in the templates/initial_deployment_templates directory of the package. This pipeline will deploy all the infrastructure in all the child accounts.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns

**Epics**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Provide a name for the S3 bucket that will store templates in the centralized logging account.</strong></td>
<td>Enter the name of the S3 bucket that you provided for the S3 Bucket Name for the CloudFormation Template storage in Logging Account parameter in Step 2.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td><strong>Provide the name of the IAM role to be used in the child accounts.</strong></td>
<td>Enter the name that you provided for the Name of the IAM role parameter in Step 1.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td><strong>Provide an active email address for receiving CodePipeline failure notifications.</strong></td>
<td>Enter the email address that you want to use for receiving CodePipeline failure notifications and other CloudWatch alarm-related notifications.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Step 4: Update files to include account information**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modify Accountlist.json.</strong></td>
<td>In the <code>Accountlist.json</code> file, which is at the top level in the package, add the parent security account number and the child account numbers. Note that the ChildAccountList field also includes the parent security account number. See the example in the <code>deployment-instructions.md</code> file in the package.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td><strong>Modify accounts.csv</strong></td>
<td>In the <code>accounts.csv</code> file, which is at the top level in the package, add all the child accounts along with the email registered with the accounts. See the example in the <code>deployment-instructions.md</code> file.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
| **Modify parameters.config.** | In the `parameters.config` file, which is in the `/templates` folder, update the following six parameters:  
  - pNotifyEmail: The email address that you provided when you set up the pipeline (see Step 3)  
  - pstackNameLogging: The name of the CloudFormation stack for centralized logging | Cloud architect |
### Task Description Skills required

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| • pS3LogsBucket: The name of the S3 bucket where logs from all accounts will be stored (see Step 2)  
• pBucketName: The ARN for the S3 bucket used to store the logs  
• pTemplateBucketName: The name of the S3 buckets where templates will be stored (see Step 2)  
• pAllowedAccounts: Account IDs for the parent and child accounts | For the other parameters, you can keep the default values. For an example, see the deployment-instructions.md file in the package. | |

### Step 5: Access the CodeCommit repository and push the updated files

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access the CodeCommit repo that you created in Step 3.</td>
<td>From the Outputs section of the CI/CD infrastructure CloudFormation stack (launched in Step 3), note the name of the CodeCommit repository URL. Create access to the repository so that the files can be pushed to it for the infrastructure to be deployed in all the target accounts. For more information, see Setting up for AWS CodeCommit.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Push the files to the CodeCommit repository.</td>
<td>Install Git on your machine. Then run the Git commands to clone the empty repository, copy the files from your laptop to the repository folder, and push the artifacts to the repository. Check for the sample Git commands in the deployment-instructions.md file in the package. For basic Git commands, see the Related resources section.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Step 6: Confirm CodePipeline and CodeBuild status

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the status of CodePipeline and CodeBuild.</td>
<td>After you push the artifacts to the CodeCommit repo, confirm that the CodePipeline pipeline that you created in Step 3 has been initiated. Then check the CodeBuild logs to confirm the status or errors.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Deploying AWS CloudFormation templates
- Setting up for AWS CodeCommit
- Uploading files to S3 bucket
- Basic Git commands

Attachments

attachment.zip

Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version

Created by SaiJeevan Devireddy (AWS)

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Content delivery; Security, identity, compliance</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services: Amazon SNS; Amazon CloudFormation; Amazon CloudWatch; AWS Lambda</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern checks an Amazon CloudFront distribution to make sure that it uses HTTPS, uses Transport Layer Security (TLS) version 1.2 or later, and has access logging enabled. CloudFront is a service provided by Amazon Web Services (AWS) that speeds up the distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.
This pattern provides an AWS Lambda function that is initiated when Amazon CloudWatch Events detects the CloudFront API call `CreateDistribution`, `CreateDistributionWithTags`, or `UpdateDistribution`. The custom logic in the Lambda function evaluates all CloudFront distributions that were created or updated in the AWS account. It sends a violation notification by using Amazon Simple Notification Service (Amazon SNS) if it detects the following violations:

- **Global checks:**
  - Custom certificate doesn't use TLS version 1.2
  - Logging is disabled for distribution
- **Origin checks:**
  - Origin isn't configured with TLS version 1.2
  - Communication with origin is allowed on a protocol other than HTTPS
- **Behavior checks:**
  - Default behavior communication is allowed on a protocol other than HTTPS
  - Custom behavior communication is allowed on a protocol other than HTTPS

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- An email address where you want to receive the violation notifications

**Limitations**

- This security control doesn't check for existing Cloudfront distributions unless an update has been made to the distribution.
- CloudFront is considered a global service and isn't tied to a specific AWS Region. However, Amazon CloudWatch Logs and AWS Cloudtrail API logging for global services occur in the US East (N. Virginia) Region (`us-east-1`). Therefore, this security control for CloudFront must be deployed and maintained in `us-east-1`. This single deployment monitors all distributions for CloudFront. Do not deploy the security control in any other AWS Regions. (Deployment in other Regions will result in a failure to initiate CloudWatch Events and the Lambda function, and no SNS notifications.)
- This solution has gone through extensive testing with CloudFront web content distributions. It does not cover real-time messaging protocol (RTMP) streaming distributions.

**Architecture**

**Target technology stack**

- Lambda function
- SNS topic
- Amazon EventBridge rule

**Target architecture**
Automation and scale

- If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy the attached template across multiple accounts that you want to monitor.

Tools

AWS services

- **AWS CloudFormation** – CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.
- **Amazon EventBridge** – EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, routing that data to targets such as Lambda functions.
- **AWS Lambda** – Lambda supports running code without provisioning or managing servers.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- **Amazon SNS** – Amazon SNS coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

The attached code includes:

- A .zip file that contains the Lambda code (index.py)
- A CloudFormation template (.yml file) that you run to deploy the Lambda code

Epics

Upload the security control

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Create the S3 bucket for the Lambda code.</strong> On the Amazon S3 console,</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>create a S3 bucket with an unique name that does not contain leading slashes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An S3 bucket name is globally unique,</td>
<td></td>
</tr>
</tbody>
</table>
and the namespace is shared by all AWS accounts. Your S3 bucket must be in the Region where you are planning to deploy the Lambda code.

Upload the Lambda code to the S3 bucket. Upload the Lambda code (cloudfront_ssl_log_lambda.zip file) that's provided in the Attachments section to the S3 bucket you created in the previous step.

Cloud architect

Deploy the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>On the AWS CloudFormation console, in the same AWS Region as the S3 bucket, deploy the CloudFormation template (cloudfront-ssl-logging.yml) that's provided in the Attachments section.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Specify the S3 bucket name.</td>
<td>For the S3 Bucket parameter, specify the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Specify the Amazon S3 key name for the Lambda file.</td>
<td>For the S3 Key parameter, specify the Amazon S3 location of the Lambda code .zip file in your S3 bucket. Do not include leading slashes (for example, you can enter lambda.zip or controls/lambda.zip).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide a notification email address.</td>
<td>For the Notification email parameter, provide an email address where you would like to receive the violation notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>For the Lambda Logging level parameter, define the logging level for your Lambda function. Choose one of the following values:</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>• INFO to get detailed informational messages on the application's progress.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ERROR to get information about error events that could</td>
<td></td>
</tr>
</tbody>
</table>
### Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the CloudFormation template has been deployed successfully, a new SNS topic is created and a subscription message is sent to the email address you provided. You must confirm this email subscription to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

### Related resources

- AWS CloudFormation information
- Creating a stack on the AWS CloudFormation console ([CloudFormation documentation](https://docs.aws.amazon.com/cloudformation/userguide/creating-a-stack.html))
- CloudFront logging ([CloudFront documentation](https://docs.aws.amazon.com/cloudfront/latest/userguide/monitor-front-logs.html))
- Amazon S3 information
- AWS Lambda information

### Attachments

attachment.zip

Check for single-host network entries in security group ingress rules for IPv4 and IPv6

*Created by SaiJeevan Devireddy (AWS) and John Reynolds (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Networking; Security, identity, compliance</th>
<th>AWS services: Amazon SNS; AWS CloudFormation; Amazon CloudWatch; AWS Lambda; Amazon VPC</th>
</tr>
</thead>
</table>
Summary

This pattern provides a security control that notifies you when Amazon Web Services (AWS) resources do not meet your specifications. It provides an AWS Lambda function that looks for single-host network entries in both Internet Protocol version 4 (IPv4) and IPv6 security group source address fields. The Lambda function is initiated when Amazon CloudWatch Events detects the Amazon Elastic Compute Cloud (Amazon EC2) `AuthorizeSecurityGroupIngress` API call. The custom logic in the Lambda function evaluates the subnet mask of the CIDR block of the security group ingress rule. If the subnet mask is determined to be anything other than /32 (IPv4) or /128 (IPv6), the Lambda function sends a violation notification by using Amazon Simple Notification Service (Amazon SNS).

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the violation notifications

Limitations

- This security monitoring solution is regional and must be deployed in each AWS Region that you want to monitor.

Architecture

Target technology stack

- Lambda function
- SNS topic
- Amazon EventBridge rule

Target architecture

Automation and scale

- If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template across multiple accounts that you want to monitor.

Tools

AWS services

- AWS CloudFormation – CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.
• **Amazon EventBridge** – EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, and routes that data to targets such as Lambda functions.

• **AWS Lambda** – Lambda supports running code without provisioning or managing servers.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

• **Amazon SNS** – Amazon SNS coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

**Code**

The attached code includes:

- A .zip file that contains the Lambda security control code (index.py)
- A CloudFormation template (.yml file) that you run to deploy the Lambda code.

**Epics**

**Upload the security control**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the S3 bucket for the Lambda code.</td>
<td>On the Amazon S3 console, create a S3 bucket with an unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket must be in the AWS Region where you are intending to deploy the security group ingress check.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

| Upload the Lambda code to the S3 bucket. | Upload the Lambda code (security-control-lambda.zip file) that's provided in the Attachments section to the S3 bucket you created in the previous step. | Cloud architect       |

**Deploy the CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>On the AWS CloudFormation console, in the same AWS</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Region as the S3 bucket, deploy the CloudFormation template (security-control.yml) that's provided in the Attachments section.</td>
<td>Cloud architect</td>
<td></td>
</tr>
</tbody>
</table>

**Specify the S3 bucket name.**  
For the **S3 Bucket** parameter, specify the name of the S3 bucket that you created in the first epic.

**Specify the Amazon S3 key name for the Lambda file.**  
For the **S3 Key** parameter, specify the Amazon S3 location of the Lambda code .zip file in your S3 bucket. Do not include leading slashes (for example, you can enter lambda.zip or controls/lambda.zip).

**Provide a notification email address.**  
For the **Notification email** parameter, provide an email address where you would like to receive the violation notifications.

**Define the logging level.**  
For the **Lambda Logging level** parameter, define the logging level for your Lambda function. Choose one of the following values:  
- **INFO** to get detailed informational messages on the application's progress.  
- **ERROR** to get information about error events that could still allow the application to continue running.  
- **WARNING** to get information about potentially harmful situations.

**Confirm the subscription**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the CloudFormation template has been deployed successfully, a new SNS topic is created and a subscription message is sent to the email address you provided. You must confirm this email subscription to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates

*Created by Ram Kandaswamy (AWS)*

**Environment:** Production

**Technologies:** Security, identity, compliance; DevOps; Operations

**AWS services:** AWS CloudFormation; Amazon GuardDuty; AWS Lambda; AWS Identity and Access Management

**Summary**

You can enable Amazon GuardDuty on an Amazon Web Services (AWS) account by using an AWS CloudFormation template. By default, if GuardDuty is already enabled when you try to use CloudFormation to turn it on, the stack deployment fails. However, you can use conditions in your CloudFormation template to check whether GuardDuty is already enabled. CloudFormation supports the use of conditions that compare static values; it does not support using the output of another resource property within the same template. For more information, see *Conditions* in the CloudFormation user guide.

In this pattern, you use a CloudFormation custom resource backed by an AWS Lambda function to conditionally enable GuardDuty if it is not already enabled. If GuardDuty is enabled, the stack captures the status and records it in the output section of the stack. If GuardDuty is not enabled, the stack enables it.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- An AWS Identity and Access Management (IAM) role that has permissions to create, update, and delete CloudFormation stacks

**Limitations**
• If GuardDuty has been manually disabled for an AWS account or Region, this pattern does not enable GuardDuty for that target account or Region.

Architecture

Target technology stack

The pattern uses CloudFormation for Infrastructure as Code (IaC). You use a CloudFormation custom resource backed by a Lambda function to achieve the dynamic service-enablement capability.

Target architecture

The following high-level architecture diagram shows the process of enabling GuardDuty by deploying a CloudFormation template:

1. You deploy a CloudFormation template to create a CloudFormation stack.
2. The stack creates an IAM role and a Lambda function.
3. The Lambda function assumes the IAM role.
4. If GuardDuty is not already enabled on the target AWS account, the Lambda function enables it.

Automation and scale

You can use the AWS CloudFormation StackSet feature to extend this solution to multiple AWS accounts and AWS Regions. For more information, see Working with AWS CloudFormation StackSets in the CloudFormation user guide.

Tools

• AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.
• AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together
as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **Amazon GuardDuty** – Amazon GuardDuty is a continuous security monitoring service that analyzes and processes logs. It uses threat intelligence feeds, such as lists of malicious IP addresses and domains, and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment.

- **AWS IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

## Epics

Create the CloudFormation template and deploy the stack

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the CloudFormation template.</td>
<td>1. Copy the code in CloudFormation template in the Additional information (p. 1658) section. 2. Paste the code in a text editor. 3. Save the file as sample.yaml on your workstation.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
| Create the CloudFormation stack. | 1. In AWS CLI, enter the following command. This creates a new CloudFormation stack using the sample.yaml file. For more information, see Creating a stack in the CloudFormation user guide.  
   ```bash
   aws cloudformation create-stack
   --stack-name guardduty-cf-stack
   --template-body file://sample.yaml
   ```  
   2. Confirm the following value appears in the AWS CLI, indicating that the stack has been successfully created. The amount of time required to create the stack can vary.  
   ```json
   "StackStatus": "CREATE_COMPLETE",
   ``` | AWS DevOps |
## Related resources

### References
- AWS CloudFormation documentation
- AWS Lambda resource type reference
- CloudFormation resource type: AWS::IAM::Role
- CloudFormation resource type: AWS::GuardDuty::Detector
- Four ways to retrieve any AWS service property using AWS CloudFormation (blog)

### Tutorials and videos
- Simplify Your Infrastructure Management Using AWS CloudFormation (Tutorial)
- Use Amazon GuardDuty and AWS Security Hub to secure multiple accounts (AWS re:Invent 2020)
- Best practices for authoring AWS CloudFormation (AWS re:Invent 2019)
- Threat Detection on AWS: An Introduction to Amazon GuardDuty (AWS re:Inforce 2019)

### Additional information

**CloudFormation template**

```yaml
AWSTemplateFormatVersion: 2010-09-09
Resources:
  rLambdaLogGroup:
    Type: 'AWS::Logs::LogGroup'
    Properties:
      RetentionInDays: 7
      LogGroupName: /aws/lambda/resource-checker
  rLambdaCheckerLambdaRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Sub 'resource-checker-lambda-role-${AWS::Region}'
```
AssumeRolePolicyDocument:
Version: 2012-10-17
Statement:
- Effect: Allow
Principal:
  Service:
  - lambda.amazonaws.com
Action: 'sts:AssumeRole'
Path: /
Policies:
- PolicyName: !Sub 'resource-checker-lambda-policy-${AWS::Region}'
  PolicyDocument:
    Version: 2012-10-17
    Statement:
    - Sid: CreateLogGroup
      Effect: Allow
      Action:
      - 'logs:CreateLogGroup'
      - 'logs:CreateLogStream'
      - 'logs:PutLogEvents'
      - 'cloudformation:CreateStack'
      - 'cloudformation:DeleteStack'
      - 'cloudformation:DescribeStacks'
      - 'guardduty:CreateDetector'
      - 'guardduty:DeleteDetector'
      Resource: '*'
resourceCheckerLambda:
  Type: 'AWS::Lambda::Function'
  Properties:
    Description: Checks for resource type enabled and possibly name to exist
    FunctionName: resource-checker
    Handler: index.lambda_handler
    Role: !GetAtt 
      - rLambdaCheckerLambdaRole
      - Arn
    Runtime: python3.8
    MemorySize: 128
    Timeout: 180
    Code:
      ZipFile: |
      import boto3
      import os
      import json
      from botocore.exceptions import ClientError
      import cfnresponse
      guardduty=boto3.client('guardduty')
      cfn=boto3.client('cloudformation')
      def lambda_handler(event, context):
        print('Event: ', event)
        if 'RequestType' in event:
          if event['RequestType'] in ['Create','Update']:
            enabled=False
            try:
              response=guardduty.list_detectors()
              if "DetectorIds" in response and len(response["DetectorIds"])>0:
                enabled="AlreadyEnabled"
              elif "DetectorIds" in response and len(response["DetectorIds"])==0:
                cfn_response=cfn.create_stack(
                  StackName='guardduty-cfn-stack',
                  TemplateBody='{ "AWSTemplateFormatVersion": "2010-09-09",
                  "Description": "A sample template", "Resources": { "IRWorkshopGuardDutyDetector": {
                    "Type": "AWS::GuardDuty::Detector", "Properties": { "Enable": true } } } }')
Enable transparent data encryption in Amazon RDS for SQL Server

Alternative code option for the Lambda resource

The provided CloudFormation template uses inline code to reference the Lambda resource, for easier reference and guidance. Alternatively, you can place the Lambda code in an Amazon Simple Storage Service (Amazon S3) bucket and reference it in the CloudFormation template. Inline code doesn’t support package dependencies or libraries. You can support these by placing the Lambda code in an S3 bucket and referencing it in the CloudFormation template.

Replace the following lines of code:

```python
Code:
   ZipFile: |
```

with the following lines of code:

```python
Code:
   S3Bucket: <bucket name>
   S3Key: <python file name>
   S3ObjectVersion: <version>
```

The S3ObjectVersion property can be omitted if you are not using versioning in your S3 bucket. For more information, see Using versioning in S3 buckets in the Amazon S3 user guide.

Enable transparent data encryption in Amazon RDS for SQL Server

Created by Ranga Cherukuri (AWS)
Summary

This pattern describes how to implement Transparent Data Encryption (TDE) in Amazon Relational Database Service (Amazon RDS) for SQL Server to encrypt data at rest.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS for SQL Server DB instance

Product versions

Amazon RDS currently supports TDE for the following SQL Server versions and editions:

- SQL Server 2012 Enterprise Edition
- SQL Server 2014 Enterprise Edition
- SQL Server 2016 Enterprise Edition
- SQL Server 2017 Enterprise Edition

For the latest information about supported versions and editions, see Support for Transparent Data Encryption in SQL Server in the Amazon RDS documentation.

Architecture

Technology stack

- Amazon RDS for SQL Server
Tools

**Tools**

- SSMS - Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing a SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

**Code**

*Query to get the default certificate name:*

```sql
USE [master]
GO
SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'
GO
```

*Query to create a database encryption key (you can specify the AES_256 algorithm instead of AES_128):*

```sql
USE [Databasename]
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE [certificatename]
```
Query to enable database encryption:

```sql
ALTER DATABASE [Database Name] 
SET ENCRYPTION ON 
GO
```

Query to check the status of encryption:

```sql
SELECT DB_NAME(database_id) AS DatabaseName, encryption_state, percent_complete FROM 
sys.dm_database_encryption_keys
```

**Epics**

Create an option group in the Amazon RDS console

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the Amazon RDS console.</td>
<td>Sign in to the AWS Management Console and open the Amazon RDS console at <a href="https://console.aws.amazon.com/rds/">https://console.aws.amazon.com/rds/</a>.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Create an option group.</td>
<td>In the navigation pane, choose Option groups, Create group. Select sqlserver-ee as the DB engine and select the engine version of your choice.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Add the TRANSPARENT_DATA_ENCRYPTION option.</td>
<td>Edit the option group you created and add the option called TRANSPARENT_DATA_ENCRYPTION.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>

Associate the option group with the DB instance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the DB instance.</td>
<td>In the Amazon RDS console, in the navigation pane, choose Databases, and then choose the DB instance you want to associate with the option group.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Associate the DB instance with the option group.</td>
<td>Choose Modify, and then use the Option group setting to associate the SQL Server DB instance with the option group you created earlier.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Apply the changes.</td>
<td>Apply the changes immediately or during the next maintenance window, as desired.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>
Create the database encryption key

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the certificate name.</td>
<td>Get the default certificate name by using the query shown in the Tools section.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to the Amazon RDS for SQL Server DB instance using SSMS.</td>
<td>For instructions, see the SSMS documentation link in the References section.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Create the database encryption key by using the default certificate.</td>
<td>Create a database encryption key by using the default certificate name you got earlier. Use the T-SQL query provided in the Tools section to create a database encryption key. You can specify the AES_256 algorithm instead of AES_128.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Enable the encryption on the database.</td>
<td>Use the T-SQL query provided in the Tools section to enable database encryption.</td>
<td>Developer, DBA</td>
</tr>
<tr>
<td>Check the status of encryption.</td>
<td>Use the T-SQL query provided in the Tools section to check the status of encryption.</td>
<td>Developer, DBA</td>
</tr>
</tbody>
</table>

Related resources

- Support for Transparent Data Encryption in SQL Server (Amazon RDS documentation)
- Working with Option Groups (Amazon RDS documentation)
- Modifying an Amazon RDS DB Instance (Amazon RDS documentation)
- Transparent Data Encryption for SQL Server (Microsoft documentation)
- Using SSMS (Microsoft documentation)

Ensure that AWS CloudFormation stacks are launched from authorized S3 buckets

*Created by Chandini Penmetsa (AWS)*

**Environment:** Production  
**Technologies:** Security, identity, compliance  
**Workload:** All other workloads  
**AWS services:** Amazon SNS; AWS CloudFormation; Amazon
Summary

You can use AWS CloudFormation templates to set up Amazon Web Services (AWS) resources programmatically, so you spend less time managing those resources and more time focusing on your applications that run in AWS. This pattern provides a way to check that AWS CloudFormation stacks are created only from templates that are stored in specific Amazon Simple Storage Service (Amazon S3) buckets. This check is useful if you have a security or compliance requirement that dictates using templates stored in S3 buckets that are in an allow list.

This security control monitors the AWS CloudFormation CreateStack and UpdateStack API calls, and invokes an AWS Lambda function that checks if the template used in the call is from an authorized S3 bucket. If the template is from a non-authorized bucket, the Lambda function triggers an Amazon Simple Notification Service (Amazon SNS) email notification to the user with the relevant information.

Prerequisites and limitations

Prerequisites

- An active email address where you would like to receive violation notifications
- An S3 bucket to upload the provided Lambda code
- A list of authorized S3 bucket names

Limitations

- UpdateStack API calls that use an existing template in an unauthorized S3 bucket don't generate additional violations, because the URL for the S3 bucket isn't available in the Amazon EventBridge event. We recommend that you delete existing templates from unauthorized S3 buckets after you receive the original CreateStack violation notification.
- This security control doesn't monitor the following AWS CloudFormation events, because they handle updates after the template is initially deployed: CreateChangeSet, CreateStackSet, UpdateStackSet.
- You must deploy this security control in every AWS Region you want to monitor.

Architecture

Target technology stack

- AWS Lambda
- Amazon SNS
- Amazon EventBridge rule

Target architecture
Automation and scale

If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

- **AWS Cloudformation** – Helps you model and set up AWS resources using an infrastructure-as-code model.
- **Amazon EventBridge** – Delivers a stream of real-time data from your own applications, software-as-a-service (SaaS) applications, and AWS services, and routes that data to targets such as AWS Lambda.
- **AWS Lambda** – Lets you run code without provisioning or managing servers.
- **Amazon SNS** – Provides message delivery from publishers to subscribers. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.
- **Amazon S3** – Lets you store and retrieve any amount of data, at any time, from anywhere on the web.

Epics

Deploy the security control

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code to Amazon S3.</td>
<td>Upload the .zip file that contains the Lambda code provided in the &quot;Attachments&quot; section to a new or existing S3 bucket. This bucket should be in the same AWS Region as the resources you want to evaluate.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>Open the AWS CloudFormation console in the same Region as your S3 bucket and deploy the template provided in the &quot;Attachments&quot; section. Provide values for the parameters; these are described in the &quot;Additional information&quot; section.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription to the Amazon SNS topic.</td>
<td>When the AWS CloudFormation template deploys successfully, it sends a subscription email to the email address you provided. You must confirm this email subscription to start receiving notifications.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Related resources

- Deploying AWS CloudFormation templates
- Amazon EventBridge
- AWS Lambda
- Amazon S3

Additional information

When you deploy the AWS CloudFormation template provided with this pattern, you'll be prompted for the following information:

- **S3 bucket**: Specify the bucket where you uploaded the attached Lambda code (.zip file). You can create a new bucket or specify an existing bucket.
- **S3 key**: Specify the location of the Lambda .zip file in your S3 bucket (for example: `filename.zip` or `controls/filename.zip`). Do not use leading slash marks.
- **Notification email**: Provide an active email address where violation notifications should be sent.
- **Lambda logging level**: Specify the logging level for the Lambda function. Use **Info** to log detailed informational messages on progress, **Error** for error events that would still allow the deployment to continue, and **Warning** for potentially harmful situations.
- **Authorized buckets**: Provide a comma-delimited list of authorized S3 buckets.

Attachments

attachment.zip

Ensure AWS load balancers use secure listener protocols (HTTPS, SSL/TLS)

*Created by Chandini Penmetsa (AWS)*

<table>
<thead>
<tr>
<th>Environment: Production</th>
<th>Technologies: Security, identity, compliance</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS services</strong>: Amazon SNS; AWS CloudFormation; Amazon CloudWatch; AWS Lambda; Elastic Load Balancing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

On the Amazon Web Services (AWS) Cloud, Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, IP addresses, and AWS Lambda functions. The load balancers use listeners to
define the ports and protocols that the load balancer uses to accept traffic from users. Application Load Balancers make routing decisions at the application layer and use the HTTP/HTTPS protocols. Network Load Balancers make routing decisions at the transport layer and use the Transmission Control Protocol (TCP), Transport Layer Security (TLS), User Datagram Protocol (UDP), or TCP_UDP protocols. Classic Load Balancers make routing decisions at either the transport layer, using TCP or Secure Sockets Layer (SSL) protocols, or at the application layer, using HTTP/HTTPS.

Your organization might have a security or compliance requirement that load balancers accept traffic from users only on secure protocols, such as HTTPS or SSL/TLS.

This pattern provides a security control that uses an Amazon EventBridge rule to monitor the CreateListener and ModifyListener API calls for Application Load Balancers and Network Load Balancers, and the CreateLoadBalancerListeners and CreateLoadBalancer API calls for Classic Load Balancers. If HTTP, TCP/UDP, or TCP_UDP is used for the load balancer's listener protocol, the control invokes a Lambda function. The Lambda function publishes a message to an Amazon Simple Notification Service (Amazon SNS) topic to send a notification that contains the load balancer details.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the violation notification
- An Amazon Simple Storage Service (Amazon S3) bucket to store the Lambda code .zip file

Limitations

- This security control does not check for existing load balancers unless an update is made to the load balancer listeners.
- This security control is regional and must be deployed in the AWS Regions that you intend to monitor.

Architecture

Target technology stack

- Lambda function
- Amazon SNS topic
- EventBridge rule

Target architecture

![Diagram of AWS architecture](image-url)
Automation and scale

- If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want this to monitor.

Tools

AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.

Amazon EventBridge – Amazon EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, routing that data to targets such as Lambda functions.

AWS Lambda – Lambda supports running code without provisioning or managing servers.

Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Epics

Deploy the security control

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same Region as the load balancer that is being evaluated.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Upload the Lambda code to the S3</td>
<td>Upload the Lambda code .zip file that's provided in the &quot;Attachments&quot; section to the defined S3 bucket.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>bucket.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy the AWS CloudFormation</td>
<td>On the AWS CloudFormation console, in the same AWS Region as the S3 bucket, deploy the template that is provided in the &quot;Attachments&quot; section. In the next epic, provide the values for the parameters.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>template.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Enter the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the Amazon S3 prefix.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, &lt;directory&gt;/&lt;filename&gt;.zip).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the SNS topic ARN.</td>
<td>Provide the SNS topic Amazon Resource Name (ARN) if you want to use an existing SNS topic for violation notifications. To create new SNS topic, keep the value as &quot;None&quot; (the default value).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. “Info” designates detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, if a new SNS topic was created, a subscription email message is sent to the email address provided in the parameters. You must confirm this email subscription to receive violation notifications.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating a stack on the AWS CloudFormation console
- AWS Lambda
Ensure encryption for Amazon EMR data at rest is enabled at launch

Created by Priyanka Chaudhary (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Security, identity, compliance; Analytics</th>
<th>Workload:</th>
<th>Open-source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon EMR; Amazon SNS; AWS KMS; AWS CloudFormation; AWS Lambda; Amazon S3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

This pattern provides a security control for monitoring the encryption of Amazon EMR clusters on Amazon Web Services (AWS).

Data encryption helps prevent unauthorized users from reading data on a cluster and associated data storage systems. This includes data that may be intercepted as it travels the network, known as data in transit, and data that is saved to persistent media, known as data at-rest. Data at rest in Amazon Simple Storage Service (Amazon S3) can be encrypted in two ways.

- Server-side encryption with Amazon S3–managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), set up with policies that are suitable for Amazon EMR.

This security control monitors for API calls and initiates an Amazon CloudWatch Events event on RunJobFlow. The trigger invokes AWS Lambda, which runs a Python script. The function retrieves the EMR cluster ID from the event JSON input and determines whether there is a security violation by performing the following checks.

1. Check if an EMR cluster is associated with an Amazon EMR specific security configuration.
2. If an Amazon EMR specific security configuration is associated with the EMR cluster, check if Encryption-at-Rest is turned on.
3. If Encryption-at-Rest is not turned on, send an Amazon Simple Notification Service (Amazon SNS) notification that includes the EMR cluster name, violation details, AWS Region, AWS account, and the Lambda Amazon Resource Name (ARN) that this notification is sourced from.
Prerequisites and limitations

Prerequisites

- An active AWS account
- An S3 bucket for the Lambda code .zip file
- An email address where you want to receive the violation notification
- Amazon EMR logging turned off so that all the API logs can be retrieved

Limitations

- This detective control is regional and must be deployed in the AWS Regions you intend to monitor.

Product versions

- Amazon EMR release 4.8.0 and above

Architecture

Target technology stack

- Amazon EMR
- Amazon CloudWatch Events event
- Lambda function
- Amazon SNS

Automation and scale

- If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools
Epics

Define the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same Region as the Amazon EMR cluster that is being evaluated.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Upload the Lambda code to the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code to the S3 bucket.</td>
<td>Upload the Lambda code .zip file that's provided in the &quot;Attachments&quot; section to the defined S3 bucket.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>On the AWS CloudFormation console, in the same Region as your S3 bucket, deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters. For more information about deploying AWS CloudFormation templates, see the &quot;Related resources&quot; section.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Enter the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the Amazon S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, &lt;directory&gt;/&lt;filename&gt;.zip).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. &quot;Info&quot; designates detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email message to the email address provided. You must</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
### Related resources

- Creating a stack on the AWS CloudFormation console
- AWS Lambda
- Amazon EMR encryption options

### Attachments

attachment.zip

### Ensure that an IAM profile is associated with an EC2 instance

Created by Mansi Suratwala (AWS)

| Environment: Production | Technologies: Infrastructure; Security, identity, compliance | AWS services: Amazon EC2; AWS Identity and Access Management; Amazon CloudWatch; AWS Lambda; Amazon SNS |

### Summary

This pattern provides an AWS CloudFormation security control template that sets up automatic notification when an AWS Identity and Access Management (IAM) profile violation occurs for an Amazon Elastic Compute Cloud (Amazon EC2) instance.

An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.

Amazon CloudWatch Events initiates this check when AWS CloudTrail logs Amazon EC2 API calls based on the RunInstances, AssociateIamInstanceProfile, and ReplaceIamInstanceProfileAssociation actions. The trigger calls an AWS Lambda function, which uses an Amazon CloudWatch Events event to check for an IAM profile.

If an IAM profile does not exist, the Lambda function initiates an Amazon Simple Notification Service (Amazon SNS) email notification that includes the Amazon Web Services (AWS) account ID and the AWS Region.

If an IAM profile does exist, the Lambda function checks for any wildcard entries in the policy documents. If the wildcards entries exist, initiates an Amazon SNS violation notification, which helps you to...
implement enhanced security. The notification contains the name of the IAM profile, the event, the EC2 instance ID, the name of the managed policy, the violation, the account ID, and the Region.

Prerequisites and limitations

Prerequisites

• An active account
• An Amazon Simple Storage Service (Amazon S3) bucket for the Lambda code .zip file

Limitations

• The AWS CloudFormation template must be deployed for the RunInstances, AssociateIamInstanceProfile, and ReplaceIamInstanceProfileAssociation actions only.
• The security control does not monitor the detachment of IAM profiles.
• The security control does not check for modification of IAM policies that are attached to the EC2 instance IAM profile.
• The security control does not account for unsupported resource-level permissions that require the use of "Resource:*".

Architecture

Target technology stack

• Amazon EC2
• AWS CloudTrail
• Amazon CloudWatch
• AWS Lambda
• Amazon S3
• Amazon SNS

Target architecture

Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to launch the template only one time for each account or Region.
Tools

Tools

- **Amazon EC2** – Amazon EC2 provides scalable computing capacity (virtual servers) in the AWS Cloud.
- **AWS CloudTrail** – AWS CloudTrail helps you enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, a role, or an AWS service are recorded as events in CloudTrail.
- **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- **AWS Lambda** – AWS Lambda is a compute service that you can use to run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- **Amazon S3** – Amazon S3 provides highly scalable object storage that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- **Amazon SNS** – Amazon SNS enables applications and devices to send and receive notifications from the cloud.

Code

- A `.zip` file of the project is available as an attachment.

Epics

Define the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>To host the Lambda code <code>.zip</code> file, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same Region as the EC2 instance that is being evaluated.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Upload the Lambda code to the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code to the S3 bucket.</td>
<td>Upload the Lambda code that's provided in the &quot;Attachments&quot; section to the S3 bucket. The S3 bucket must be in the same Region as the EC2 instance being evaluated.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>
Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>Deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Enter the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, /.zip).</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. “Info” designates detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email message to the email address provided. You must confirm this email subscription to receive violation notifications.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating an S3 bucket
Ensure an Amazon Redshift cluster is encrypted upon creation

Created by Mansi Suratwala (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Technologies:</th>
<th>Workload:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production</td>
<td>Analytics; Data lakes; Security, identity, compliance</td>
<td>All other workloads</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon Redshift; Amazon SNS; AWS CloudTrail; Amazon CloudWatch; AWS Lambda; Amazon S3

**Summary**

This pattern provides an AWS CloudFormation template that provides you with automatic notification when a new Amazon Redshift cluster is created without encryption.

The AWS CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event watches for any Amazon Redshift cluster being created or being restored from a snapshot through AWS CloudTrail. If the cluster is created without AWS Key Management Service (AWS KMS) or cloud hardware security model (HSM) encryption in the AWS account, CloudWatch initiates a Lambda function that sends you an Amazon Simple Notification Service (Amazon SNS) notification informing you of the violation.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account.
- A virtual private cloud (VPC) with a cluster subnet group, and an associated security group.

**Limitations**

- The AWS CloudFormation template can be deployed for the `CreateCluster` and `RestoreFromClusterSnapshot` actions only.
**Architecture**

**Target technology stack**

- Amazon Redshift
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

**Target architecture**

Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run it only one time in each Region or account.

**Tools**

**Tools**

- **Amazon Redshift** – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Amazon Redshift is integrated with your data lake, which enables you to use your data to acquire new insights for your business and customers.

- **AWS CloudTrail** – AWS CloudTrail is an AWS service that helps you implement governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.

- **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.

- **AWS Lambda** – AWS Lambda supports running code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

- **Amazon S3** – Amazon S3 is a highly scalable object storage service that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

- **Amazon SNS** – Amazon SNS is a web service that coordinates and manages the delivery or sending of messages to between publishers and clients, including web servers and email addresses.
Code

- A.zip file of the project is available as an attachment.

Epics

Define the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>On the Amazon S3 console, choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. Your S3 bucket needs to be in the same Region as the Amazon Redshift cluster being evaluated. The S3 bucket's name cannot contain leading slashes.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Upload the Lambda code to the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code to the S3 bucket.</td>
<td>Upload the Lambda code provided in the &quot;Attachments&quot; section to the S3 bucket. The S3 bucket must be in the same Region as the Amazon Redshift cluster being evaluated.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the AWS CloudFormation template.</td>
<td>Deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

Complete the parameters in the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name the S3 bucket.</td>
<td>Enter the name of the S3 bucket that you created in the first epic.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud Architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. “Info” designates detailed informational messages on the application's progress. “Error” designates error events that could still allow the application to continue running. “Warning” designates potentially harmful situations.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

**Confirm the subscription**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email to the email address provided. You must confirm this email subscription to receive violation notifications.</td>
<td>Cloud Architect</td>
</tr>
</tbody>
</table>

**Related resources**

- Creating an S3 bucket
- Uploading files to an S3 bucket
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail
- Creating an Amazon Redshift cluster

**Attachments**

attachment.zip

**Monitor and remediate scheduled deletion of AWS KMS keys**

*Created by Mikesh Khanal (AWS) and Ramya Pulipaka (AWS)*
Summary

On the Amazon Web Services (AWS) Cloud, deleting an AWS Key Management Services (AWS KMS) key can result in data loss. Deletion removes the key material and all metadata associated with the AWS KMS key, and it is irreversible. After an AWS KMS key is deleted, you can no longer decrypt the data that were encrypted under that AWS KMS key, so that data cannot be recovered.

This pattern sets up monitoring, with notifications when an application or a user schedules an AWS KMS key for deletion. If you receive a notification, you might want to cancel deletion of the AWS KMS key and reconsider your decision to delete it. The pattern uses the AWS Systems Manager automation runbook `AWSConfigRemediation-CancelKeyDeletion` to facilitate canceling the deletion of an AWS KMS key.

Note: The pattern’s CloudFormation template must be deployed in all AWS Regions where you want to monitor deletion of AWS KMS keys.

Prerequisites and limitations

Prerequisites

- An active AWS account with an AWS Identity and Access Management (IAM) user
- Understanding of the following AWS services:
  - Amazon EventBridge
  - AWS KMS
  - Amazon Simple Notification Service (Amazon SNS)
  - AWS Systems Manager.

Limitations

- Any customization of the solution requires knowledge of AWS CloudFormation templates and the AWS services used in this pattern.
- Currently, this solution uses the default event bus, and it can be customized according to the requirements. For more information about the custom event bus, see the AWS documentation.

Architecture

Target technology stack

- Amazon EventBridge
- AWS KMS
- Amazon SNS
- AWS Systems Manager
- Automation using the following:
  - AWS Command Line Interface (AWS CLI) or AWS SDK
  - AWS CloudFormation stack

Target architecture
1. Deletion of an AWS KMS key is scheduled.
2. The scheduled-deletion event is evaluated by an EventBridge rule.
3. The EventBridge rule engages the Amazon SNS topic.
4. The EventBridge rule initiates the Systems Manager automation and runbooks.
5. The runbooks cancel the deletion.

Automation and scale

The CloudFormation stack deploys all the necessary resources and services for this solution to work. The pattern can be run independently in a single account or run using AWS CloudFormation StackSets for multiple independent accounts or an organization.

```
aws cloudformation create-stack  --stack-name <stack-name>  
   --template-body file://<Full-Path-of-file>  
   --parameters ParameterKey=,ParameterValue=  
   --capabilities CAPABILITY_NAMED_IAM
```

Tools

• **AWS CloudFormation** – AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run on AWS. You can use a CloudFormation template to create stacks in an AWS account in an AWS Region. The template describes all the AWS resources that you want, and CloudFormation provisions and configures those resources for you.

• **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open source tool that you can use to interact with AWS services using commands in your command line shell.

• **Amazon EventBridge** – Amazon EventBridge is a serverless event bus service connecting your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications and AWS services, and it routes that data to targets such as AWS Lambda. EventBridge simplifies the process of building event-driven architectures.

• **AWS KMS** – AWS Key Management Service (AWS KMS) is a managed service for creating and controlling AWS KMS keys, the encryption keys used to encrypt your data.
• **AWS SDKs** – AWS tools include SDKs so that you can develop and manage applications on AWS in the programming language of your choice.

• **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel.

• **AWS Systems Manager** – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

**Code**

• The `alerting_ct_logs.yaml` CloudFormation template for the project is attached.

## Epics

### Prepare the AWS account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install and configure AWS CLI.</td>
<td>Install AWS CLI version 2. Then configure the security credentials settings for an identity, the default output format, and the default AWS Region that AWS CLI uses to interact with AWS. The identity must have the required permissions to perform the tasks.</td>
<td>Developer, Security engineer</td>
</tr>
</tbody>
</table>

### Deploy the AWS CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the CloudFormation template.</td>
<td>Download the attachment to a local path on your computer and extract the <code>alerting_ct_logs.yaml</code> template file.</td>
<td>Developer, Security engineer</td>
</tr>
<tr>
<td>Deploy the template.</td>
<td>In the terminal window where the AWS account profile has been configured, run the following command.</td>
<td>Developer, Security engineer</td>
</tr>
</tbody>
</table>
## Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>Check your email inbox and choose <strong>Confirm subscription</strong> in the email message that you receive from Amazon</td>
<td>Developer, Security engineer</td>
</tr>
</tbody>
</table>

In the next step, enter values for the template parameters.

**Complete the template parameters.**

Enter the required values for the parameters.

- **DestinationEmailAddress** – The email address to receive an alert when an AWS KMS key is scheduled for deletion.
- **SNSTopicName** – The name of the Amazon SNS topic.
- **EnableRemediation** – Cancellation of the scheduled key deletion using a Systems Manager runbook. Allowed values are `true` and `false`.
- **AutomationAssumeRole** – The Amazon Resource Name (ARN) of the role that allows Systems Manager automation to perform the actions on your behalf. For more information, see the **Required IAM Permissions** section in the [AWSConfigRemediation-CancelKeyDeletion](#) documentation.
- **Capabilities** – For AWS CloudFormation to create the stack, you must explicitly acknowledge that your stack template contains certain capabilities.
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNS. A web browser window will open and display a subscription confirmation and your subscription ID.</td>
<td></td>
</tr>
</tbody>
</table>

### Related resources

**References**

- Creating a rule for an AWS service
- Creating an Amazon CloudWatch alarm to detect usage of an AWS KMS key that is pending deletion

**Tutorials and videos**

- How to get started with Amazon EventBridge
- Deep dive on Amazon EventBridge (AWS Online Tech Talks)

**AWS workshop**

- Working with EventBridge rules

### Additional information

The following code provides examples for extending the solution to monitor and notify you of any changes in any AWS Service. The examples include predefined patterns and custom patterns. For more information, see Events and event patterns in EventBridge.

```json
EventPattern:
  source:
    - aws.kms
  detail-type:
    - AWS API Call via CloudTrail
  detail:
    eventSource:
      - kms.amazonaws.com
    eventName:
      - ScheduleKeyDeletion
```

### Attachments

attachment.zip

### Manage credentials using AWS Secrets Manager

*Created by Durga Prasad Cheepuri (AWS)*

**Created by:** AWS  
**Environment:** PoC or pilot  
**Technologies:** Databases; Security, identity, compliance
Summary

This pattern walks you through using AWS Secrets Manager to dynamically fetch database credentials for a Java Spring application.

In the past, when you created a custom application that retrieves information from a database, you typically had to embed the credentials (the secret) for accessing the database directly in the application. When it was time to rotate the credentials, you had to invest time to update the application to use the new credentials, and then distribute the updated application. If you had multiple applications that shared credentials and you missed updating one of them, the application would fail. Because of this risk, many users chose not to regularly rotate their credentials, which effectively substituted one risk for another.

Secrets Manager enables you to replace hard-coded credentials in your code (including passwords) with an API call to retrieve the secret programmatically. This helps ensure that the secret can't be compromised by someone who is examining your code, because the secret simply isn't there. You can also configure Secrets Manager to automatically rotate the secret according to a schedule that you specify. This enables you to replace long-term secrets with short-term ones, which helps significantly reduce the risk of compromise. For more information, see the AWS Secrets Manager documentation.

Prerequisites and limitations

Prerequisites

- An AWS account with access to Secrets Manager
- A Java Spring application

Architecture

Source technology stack

- A Java Spring application with code that accesses a database, with DB credentials managed from the application.properties file.

Target technology stack

- A Java Spring application with code that accesses a database, with DB credentials managed in Secrets Manager. The application.properties file holds the secrets to Secrets Manager.

Secrets Manager integration with an application
**Tools**

- **Secrets Manager** – AWS Secrets Manager is an AWS service that makes it easier for you to manage secrets. Secrets can be database credentials, passwords, third-party API keys, and even arbitrary text. You can store and control access to these secrets centrally by using the Secrets Manager console, the Secrets Manager command-line interface (CLI), or the Secrets Manager API and SDKs.

**Epics**

**Store secret in Secrets Manager**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store the DB credentials as a secret in Secrets Manager.</td>
<td>Store Amazon Relational Database Service (Amazon RDS) or other DB credentials as a secret in Secrets Manager by following the steps in Creating a secret in the Secrets Manager documentation.</td>
<td>Sys Admin</td>
</tr>
<tr>
<td>Set permissions for the Spring application to access Secrets Manager.</td>
<td>Set the appropriate permissions based on how the Java Spring application uses Secrets Manager. To control access to the secret, create a policy based on the information provided in the Secrets Manager documentation, in the sections Using identity-based policies (IAM Policies) and ABAC for Secrets Manager and Using resource-based policies for Secrets Manager. Follow the steps in the section Retrieving the secret value in the Secrets Manager documentation.</td>
<td>Sys Admin</td>
</tr>
</tbody>
</table>
Update the Spring application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add JAR dependencies to use Secrets Manager.</td>
<td>See the Additional information section for details.</td>
<td>Java developer</td>
</tr>
<tr>
<td>Add the details of the secret to the Spring application.</td>
<td>Update the application.properties file with the secret name, endpoints, and AWS Region. For an example, see the Additional information section.</td>
<td>Java developer</td>
</tr>
<tr>
<td>Update the DB credentials retrieval code in Java.</td>
<td>In the application, update the Java code that fetches the DB credentials to fetch those details from Secrets Manager. For example code, see the Additional information section.</td>
<td>Java developer</td>
</tr>
</tbody>
</table>

Related resources

- AWS Secrets Manager documentation
- Using identity-based policies (IAM Policies) and ABAC for Secrets Manager
- Using resource-based policies for Secrets Manager
- Sample code

Additional information

Adding JAR dependencies for using Secrets Manager

**Maven:**

```xml
<groupId>com.amazonaws</groupId>
<artifactId>aws-java-sdk-secretsmanager</artifactId>
.getVersion>1.11.355</version>
```

**Gradle:**

```gradle
compile group: 'com.amazonaws', name: 'aws-java-sdk-secretsmanager', version: '1.11.355'
```

Updating the application.properties file with the details of the secret

```
spring.aws.secretsmanager.secretName=postgres-local
spring.aws.secretsmanager.endpoint=secretsmanager.us-east-1.amazonaws.com
spring.aws.secretsmanager.region=us-east-1
```

Updating the DB credentials retrieval code in Java

```java
String secretName = env.getProperty("spring.aws.secretsmanager.secretName");
String endpoints = env.getProperty("spring.aws.secretsmanager.endpoint");
```
String AWS Region = env.getProperty("spring.aws.secretsmanager.region");
AwsClientBuilder.EndpointConfiguration config = new
   AwsClientBuilder.EndpointConfiguration(endpoints, AWS Region);
AWSSecretsManagerClientBuilder clientBuilder =
   AWSSecretsManagerClientBuilder.standard();
clientBuilder.setEndpointConfiguration(config);
AWSSecretsManager client = clientBuilder.build();

ObjectMapper objectMapper = new ObjectMapper();
JsonNode secretsJson = null;
ByteBuffer binarySecretData;

GetSecretValueRequest getSecretValueRequest = new
   GetSecretValueRequest().withSecretId(secretName);
GetSecretValueResult getSecretValueResponse = null;
try {
   getSecretValueResponse = client.getSecretValue(getSecretValueRequest);
}

catch (ResourceNotFoundException e) {
   log.error("The requested secret " + secretName + " was not found");
}

catch (InvalidRequestException e) {
   log.error("The request was invalid due to: " + e.getMessage());
}

catch (InvalidParameterException e) {
   log.error("The request had invalid params: " + e.getMessage());
}
if (getSecretValueResponse == null) {
   return null;
} // Decrypted secret using the associated KMS key // Depending on whether the secret
   was a string or binary, one of these fields will be populated

String secret = getSecretValueResponse.getSecretString();

if (secret != null) {
   try {
      secretsJson = objectMapper.readTree(secret);
   }
   catch (IOException e) {
      log.error("Exception while retrieving secret values: " + e.getMessage());
   }
} else {
   log.error("The Secret String returned is null");
   return null;
}

String host = secretsJson.get("host").textValue();
String port = secretsJson.get("port").textValue();
String dbname = secretsJson.get("dbname").textValue();
String username = secretsJson.get("username").textValue();
String password = secretsJson.get("password").textValue();
Monitor Amazon EMR clusters for in-transit encryption at launch

Created by Susanne Kangnoh (AWS)

**Environment:** Production  
**Technologies:** Analytics; Big data; Cloud-native; Security, identity, compliance  
**Workload:** Open-source

**AWS services:** Amazon EMR; Amazon SNS; AWS CloudTrail; Amazon CloudWatch

### Summary

This pattern provides a security control that monitors Amazon EMR clusters at launch and sends an alert if in-transit encryption hasn't been enabled.

Amazon EMR is a web service that makes it easy for you to run big data frameworks, such as Apache Hadoop, to process and analyze data. Amazon EMR enables you to process vast amounts of data in a cost-effective way by running mapping and reducing steps in parallel.

Data encryption prevents unauthorized users from accessing or reading data at rest or data in transit. **Data at rest** refers to data that is stored in media such as a local file system on each node, Hadoop Distributed File System (HDFS), or the EMR File System (EMRFS) through Amazon Simple Storage Service (Amazon S3). **Data in transit** refers to data that travels the network and is in flight between jobs. In-transit encryption supports open-source encryption features for Apache Spark, Apache TEZ, Apache Hadoop, Apache HBase, and Presto. You enable encryption by creating a security configuration from the AWS Command Line Interface (AWS CLI), the console, or AWS SDKs, and specifying the data encryption settings. You can provide the encryption artifacts for in-transit encryption in these two ways:

- By uploading a compressed file of certificates to Amazon S3.
- By referencing a custom Java class that provides encryption artifacts.

The security control that's included with this pattern monitors API calls and generates an Amazon CloudWatch Events event on the **RunJobFlow** action. The event calls an AWS Lambda function, which runs a Python script. The function gets the EMR cluster ID from the event JSON input, and performs the following checks to determine whether there's a security violation:

- Checks if the EMR cluster has an Amazon EMR-specific security configuration.
- If the cluster does have a security configuration, checks to see if encryption in transit is enabled.
- If the cluster doesn't have a security configuration, sends an alert to an email address that you provide, by using Amazon Simple Notification Service (Amazon SNS). The notification specifies the EMR cluster name, violation details, AWS Region and account information, and the AWS Lambda ARN (Amazon Resource Name) that the notification is sourced from.

### Prerequisites and limitations

**Prerequisites**

- An active AWS account.
• An S3 bucket to upload the Lambda code that’s provided with this pattern.
• An email address where you would like to receive violation notifications.
• Amazon EMR logging enabled, for access to all the API logs.

Limitations
• This detective control is regional and must be deployed in each AWS Region that you want to monitor.

Product versions
• Amazon EMR release 4.8.0 or later.

Architecture

Workflow architecture

Automation and scale
• If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy the template in multiple accounts that you want to monitor.

Tools

AWS services

• Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Additionally, you can use Amazon EMR to transform and move large amounts of data into and out of other AWS data stores and databases, such as Amazon S3 and Amazon DynamoDB.

• AWS Cloudformation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

• AWS Cloudwatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

• AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically from a few
requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

**Code**

This pattern includes an attachment with two files:

- **EMRInTransitEncryption.zip** is a compressed file that includes the security control (Lambda code).
- **EMRInTransitEncryption.yml** is a CloudFormation template that deploys the security control.

See the *Epics* section for information about how to use these files.

**Epics**

**Deploy the security control**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the code to an S3 bucket.</td>
<td>Create a new S3 bucket or use an existing S3 bucket to upload the attached EMRInTransitEncryption.zip file (Lambda code). This bucket must be in the same AWS Region as the CloudFormation template and the resources that you want to evaluate.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the EMRInTransitEncryption.yml file that's provided in the attachment. In the next epic, provide values for the template parameters.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Complete the parameters in the CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>the CloudFormation template and the resource that will be evaluated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Specify the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, EMRInTransitEncryption.zip or controls/EMRInTransitEncryption.zip).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Specify an active email address where you want to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Specify a logging level.</td>
<td>Specify the logging level and verbosity for the Lambda logs. Info designates detailed informational messages on the application’s progress and should be used only for debugging. Error designates error events that could still allow the application to continue running. Warning designates potentially harmful situations.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Confirm the subscription**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the email subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Related resources**

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Encryption options (Amazon EMR documentation)

**Attachments**

attachment.zip
Monitor Amazon ElastiCache clusters for at-rest encryption

Created by Susanne Kangnoh (AWS)

**Environment:** Production  
**Technologies:** Security, identity, compliance; Databases; Infrastructure; Cloud-native  
**Workload:** Open-source

**AWS services:** Amazon SNS; Amazon CloudWatch; Amazon ElastiCache

**Summary**

Amazon ElastiCache is an Amazon Web Services (AWS) service that provides a high-performance, scalable, and cost-effective caching solution for distributing an in-memory data store or cache environment in the cloud. It retrieves data from high-throughput and low-latency, in-memory data stores. This functionality makes it a popular choice for real-time use cases such as caching, session stores, gaming, geo-spatial services, real-time analytics, and queuing. ElastiCache offers Redis and Memcached data stores, both of which provide sub-millisecond response times.

Data encryption helps prevent unauthorized users from reading sensitive data available on your Redis clusters and their associated cache storage systems. This includes data saved to persistent media, known as *data at rest*, and data that can be intercepted as it travels through the network between clients and cache servers, known as *data in transit*.

You can enable at-rest encryption for ElastiCache for Redis when you create a replication group, by setting the `AtRestEncryptionEnabled` parameter to `true`. When this parameter is enabled, it encrypts the disk during sync, backup, and swap operations, and encrypts backups stored in Amazon Simple Storage Service (Amazon S3). You cannot enable at-rest encryption on an existing replication group.

When you create a replication group, you can enable encryption at rest in these two ways:

- By choosing the `Default` option, which uses service-managed encryption at rest.
- By using a customer managed key and providing the key ID or Amazon Resource Name (ARN) from AWS Key Management Service (AWS KMS).

This pattern provides a security control that monitors for API calls and generates an Amazon CloudWatch Events event on the `CreateReplicationGroup` operation. This event calls an AWS Lambda function, which runs a Python script. The function gets the replication group ID from the event JSON input, and performs the following checks to determine whether there's a security violation:

- Checks if the `AtRestEncryptionEnabled` key exists.
- If `AtRestEncryptionEnabled` exists, checks the value to see if it is `true`.
- If the `AtRestEncryptionEnabled` value is set to `false`, sets a variable that tracks violations and sends a violation message to an email address you provide, by using an Amazon Simple Notification Service (Amazon SNS) notification.
Prerequisites and limitations

Prerequisites

• An active AWS account.
• An S3 bucket to upload the provided Lambda code.
• An email address where you would like to receive violation notifications.
• ElastiCache logging enabled, for access to all the API logs.

Limitations

• This detective control is regional and must be deployed in each AWS Region that you want to monitor.
• The control supports replication groups that are running in a virtual private cloud (VPC).
• The control supports replication groups that are running the following node types:
  • R5, R4, R3
  • M5, M4, M3
  • T3, T2

Product versions

• ElastiCache for Redis version 3.2.6 or later

Architecture

Workflow architecture

Automation and scale

• If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

• Amazon ElastiCache – Amazon ElastiCache makes it easy to set up, manage, and scale distributed in-memory cache environments in the AWS Cloud. It provides a high performance, resizable, and cost-effective in-memory cache, while removing complexity associated with deploying and managing a distributed cache environment. ElastiCache works with both the Redis and Memcached engines.
• AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together.
as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **AWS Cloudwatch Events** – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

## Code

This pattern includes an attachment with two files:

- **ElasticCache-EncryptionAtRest.zip** is a compressed file that includes the security control (Lambda code).

- **elasticache_encryption_at_rest.yml** is a CloudFormation template that deploys the security control.

See the *Epics* section for information about how to use these files.

## Epics

### Deploy the security control

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the code to an S3 bucket.</td>
<td>Create a new S3 bucket or use an existing S3 bucket to upload the attached ElasticCache-EncryptionAtRest.zip file (Lambda code). This bucket must be in the same AWS Region as the resources that you want to evaluate.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the CloudFormation template.</td>
<td>Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the elasticache_encryption_at_rest.yml file that's provided in the attachment. In the next epic, provide values for the template parameters.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
Complete the parameters in the CloudFormation template

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the resource that will be evaluated.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, ElasticCache-EncryptionAtRest.zip or controls/ElasticCache-EncryptionAtRest.zip).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address where you want to receive violation notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Specify a logging level.</td>
<td>Specify the logging level and verbosity. Info designates detailed informational messages on the application's progress and should be used only for debugging. Error designates error events that could still allow the application to continue running. Warning designates potentially harmful situations.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the email subscription.</td>
<td>When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- At-Rest Encryption in ElastiCache for Redis (Amazon ElastiCache documentation)
Monitor EC2 instance key pairs using AWS Config

**Summary**

When launching an Amazon Elastic Compute Cloud (Amazon EC2) instance on the Amazon Web Services (AWS) Cloud, a best practice is to create or use an existing key pair to connect to the instance. The key pair, which consists of a public key stored in the instance and a private key provided to the user, allows secure access through Secure Shell (SSH) to the instance and avoids the use of passwords. However, users can sometimes inadvertently launch instances without attaching a key pair. Because key pairs can be assigned only during the launch of an instance, it’s important to quickly identify and flag as noncompliant any instances launched without key pairs. This is particularly useful when working in accounts or environments that mandate the use of key pairs for instance access.

This pattern describes how to create a custom rule in AWS Config to monitor EC2 instance key pairs. When instances are identified as noncompliant, an alert is sent using Amazon Simple Notification Service (Amazon SNS) notifications initiated through an Amazon EventBridge event.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- AWS Config enabled for the AWS Region you want to monitor and configured to record all AWS resources

**Limitations**

- This solution is Region-specific. All resources should be created in the same AWS Region.

**Architecture**

**Target technology stack**

- AWS Config
- Amazon EventBridge
- AWS Lambda
Target architecture

1. AWS Config initiates the rule.
2. The rule invokes the Lambda function to evaluate compliance of EC2 instances.
3. The Lambda function sends the updated compliance state to AWS Config.
4. AWS Config sends an event to EventBridge.
5. EventBridge publishes compliance change notifications to an SNS topic.
6. Amazon SNS sends an alert in email.

Automation and scale

The solution can monitor any number of EC2 instances within a Region.

Tools

Tools

- **AWS Config** – AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.
- **Amazon EventBridge** – Amazon EventBridge is a serverless event bus service for connecting your applications with data from a variety of sources.
- **AWS Lambda** – AWS Lambda is a serverless compute service that supports running code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes.
- **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication.
The code for the Lambda function is attached.

**Epics**

Create a Lambda function to evaluate Amazon EC2 compliance

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an AWS Identity and Access Management (IAM) role for Lambda.</td>
<td>On the AWS Management Console, choose <strong>IAM</strong>, and then create the role, using Lambda as the trusted entity and adding the AmazonEventBridgeFullAccess and AWSConfigRulesExecutionRole permissions. For more information, see the AWS documentation.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
| Create and deploy the Lambda function.           | 1. On the Lambda console, create a function, using **Author from scratch**, with **Python 3.6** as the runtime and the previously created IAM role. Note the Amazon Resource Name (ARN).  
2. On the **Code** tab, choose `lambda_function.py`, and paste the code that is attached to this pattern.  
3. To save your changes, choose **Deploy**. | DevOps          |

Create a custom AWS Config rule

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add a custom AWS Config rule.                    | On the AWS Config console, add a custom rule, using the following settings:  
  - ARN – The ARN of the previously created Lambda function  
  - Trigger type – Configuration changes  
  - Scope of changes – Resources  
  - Resource type – Amazon EC2 instance  
  For more information, see the AWS documentation. | DevOps          |
### Configure email notifications when a compliance change event is detected

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the SNS topic and subscription.</td>
<td>On the Amazon SNS console, create a topic using <strong>Standard</strong> as the type, and then create a subscription using <strong>Email</strong> as the protocol. When you receive the confirmation email message, choose the link to confirm the subscription. For more information, see the AWS documentation.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
| Create an EventBridge rule to initiate Amazon SNS notifications. | On the EventBridge console, create a rule, using the following settings:  
  - Service name – AWS Config  
  - Event type – Config Rules Compliance Change  
  - Message type – Specific message types, ComplianceChangeNotification  
  - Specific rule name – The name of your previously created AWS Config rule  
  - Target – SNS topic, your previously created topic  
  For more information, see the AWS documentation. | DevOps          |

### Verify the rule and notifications

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create EC2 instances.</td>
<td>Create two EC2 instances of any type and attach a key pair, and create one EC2 instance without a key pair.</td>
<td>DevOps</td>
</tr>
</tbody>
</table>
| Verify the rule. | 1. On the AWS Config console, on the **Rules** page, select your rule.  
  2. To see compliant and noncompliant EC2 instances, change **Resources in scope** to **All**. Verify that two instances are listed as compliant and noncompliant. | DevOps          |
### Related resources

- Creating a role to delegate permissions to an AWS service
- Creating a custom rule in AWS Config
- Creating an Amazon SNS topic
- Subscribing to an Amazon SNS topic
- Create a rule in Amazon EventBridge

### Attachments

attachment.zip

### Send a notification when an IAM user is created

*Created by Mansi Suratwala (AWS)*

**Environment**: Production  
**Technologies**: Infrastructure; Security, identity, compliance  
**Workload**: All other workloads

**AWS services**: Amazon SNS; AWS Identity and Access Management; AWS Lambda; Amazon CloudWatch

### Summary

This pattern provides an Amazon Web Services (AWS) CloudFormation template that you can deploy to receive notifications automatically when AWS Identity and Access Management (IAM) users are created.

Using IAM, you can manage access to AWS services and resources securely. You can create and manage AWS users and groups, and use permissions to allow and deny those users and groups access to AWS resources.

The CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event uses AWS CloudTrail to monitor for any IAM user being created in the AWS account. If a user is created, the CloudWatch Events event initiates a Lambda function, which sends you an Amazon Simple Notification Service (Amazon SNS) notification informing you of the new user creation event.
Prerequisites and limitations

Prerequisites

• An active AWS account

Limitations

• The CloudFormation template must be deployed for **CreateUser** only.

Architecture

Target technology stack

• IAM
• AWS CloudTrail
• Amazon CloudWatch Events
• AWS Lambda
• Amazon Simple Storage Service (Amazon S3)
• Amazon SNS

Target architecture

Automation and scale

You can use the CloudFormation template multiple times for different AWS Regions and accounts. You need to run it only once in each Region or account.

Tools

Tools

• **IAM** – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
• **AWS CloudFormation** – AWS CloudFormation helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want, and CloudFormation takes care of provisioning and configuring those resources for you.
• **AWS CloudTrail** – AWS CloudTrail helps you manage governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, a role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.

• **Amazon CloudWatch Events** – Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources.

• **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

• **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery using Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

**Code**

A .zip file of the project is available as an attachment.

**Epics**

**Create the S3 bucket for the Lambda script**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the S3 bucket.</td>
<td>Open the Amazon S3 console, and choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. Your S3 bucket needs to be in the same Region as the Amazon Elastic Compute Cloud (Amazon EC2) instance that is being evaluated. The S3 bucket name cannot contain leading slashes.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Upload the Lambda code to the S3 bucket**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload the Lambda code.</td>
<td>Upload the Lambda code .zip file provided in the Attachments section to the S3 bucket that you defined.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Deploy the CloudFormation template**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy the CloudFormation</td>
<td>On the CloudFormation console, deploy the CloudFormation</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>template.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New_Create_IAM_User.yml template that's provided as an attachment to this pattern. In the next epic, provide values for the template parameters.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the S3 bucket name.</td>
<td>Enter the name of the S3 bucket that you created or chose in the first epic.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide the S3 key.</td>
<td>Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, &lt;directory&gt;/&lt;filename&gt;.zip).</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Provide an email address.</td>
<td>Provide an active email address to receive Amazon SNS notifications.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Define the logging level.</td>
<td>Define the logging level and frequency for your Lambda function. <strong>Info</strong> designates detailed informational messages on the application's progress. <strong>Error</strong> designates error events that could still allow the application to continue running. <strong>Warning</strong> designates potentially harmful situations.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Confirm the subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the subscription.</td>
<td>When the template successfully deploys, it sends a subscription email message to the email address provided. To receive notifications, you must confirm this email subscription.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Related resources

- Creating an S3 bucket
- Uploading files to an S3 bucket
Restrict access to AWS APIs for AWS SSO and IAM users through trusted source IP ranges

 Created by Mostefa Brougui (AWS)

**Environment:** PoC or pilot  **Technologies:** Security, identity, compliance  **AWS services:** AWS Identity and Access Management; AWS Organizations; AWS SSO

**Summary**

This pattern provides a service control policy (SCP) that you can attach to the organization root, an organizational unit (OU), or an account in Amazon Web Services (AWS) Organizations. The SCP restricts access to AWS APIs for AWS Identity and Access Management (IAM) users and AWS Single Sign-On (AWS SSO) users—either in an AWS SSO directory or synchronized from a third-party identity source. Access is restricted to a trusted set of IP ranges that you specify in the SCP. This SCP also prevents an administrator of a member account in the AWS Organizations organization from bypassing the policy by creating a rogue identity provider in IAM.

See the Attachments (p. 6) section for a sample JSON file that you can customize to create the SCP.

**Prerequisites and limitations**

**Prerequisites**

- An organization set up with all features enabled in AWS Organizations
- Access to your organization's master account with permissions to run the following actions to deploy the SCP:
  - organizations:CreatePolicy
  - organizations:AttachPolicy

**Limitations**

- The SCP applies only to IAM users and AWS SSO users, and doesn't restrict cross-account access.
- You can also use the SCP as a reference to restrict federated access through IAM through a restricted IAM role path. To do this, replace the last condition block in the attached file with the appropriate value (for example, "arn:aws:iam::*:role/human-access/federated-access/*").
**Tools**

**AWS services**

- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

- **AWS SSO** – AWS Single Sign-On (AWS SSO) is a cloud-based service that simplifies managing SSO access to AWS accounts and business applications. You can control SSO access and user permissions across all your AWS accounts in AWS Organizations. You can also manage access to applications that support Security Assertion Markup Language (SAML) 2.0.

**Code**

The attached file provides the JSON contents of the SCP for this pattern. To restrict access to specific IP ranges, replace the variable `${sourceip}` in the JSON file.

**Epics**

Restrict access to AWS based on trusted source IP ranges

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get trusted source IP ranges.</td>
<td>Obtain the trusted source IP ranges that you want to use for the SCP. For organizations, use web gateway public IPs as the trusted source IP ranges.</td>
<td>Cloud engineer</td>
</tr>
<tr>
<td>Create an SCP.</td>
<td>Use the attached JSON file to create the SCP. Replace <code>${sourceip}</code> with the source IP ranges you want to restrict access to. For instructions, see Creating, updating, and deleting service control policies in the AWS Organizations documentation.</td>
<td>Cloud engineer</td>
</tr>
<tr>
<td>Attach the SCP to an account to test it.</td>
<td>Use the AWS Management Console or the AWS Organizations API to attach the SCP you created to an account. For instructions, see Attaching and detaching service control policies in the AWS Organizations documentation. Perform tests to ensure that the SCP is working as expected.</td>
<td>Cloud engineer</td>
</tr>
</tbody>
</table>
Related resources

- AWS JSON policy elements: Principal (IAM documentation)
- Actions, resources, and condition keys for AWS Organizations (IAM service authorization reference)
- Service control policies (SCPs) (AWS Organizations documentation)
- Manage SSO access to your AWS accounts (AWS SSO documentation)

Attachments

attachment.zip

Scan Git repositories for sensitive information and security issues by using git-secrets

Created by Saurabh Singh (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technologies:</td>
<td>Security, identity, compliance</td>
</tr>
<tr>
<td>Workload:</td>
<td>Open-source</td>
</tr>
</tbody>
</table>

Summary

This pattern describes how to use the open-source git-secrets tool from AWS Labs. Git-secrets scans Git source repositories and finds code that may potentially include sensitive information, such as user passwords, or that has other security issues.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Git repository that requires a security scan

Architecture

Target architecture

- Git
- Git-secrets
## Tools

- **git-secrets** - Prevents you from committing sensitive information into Git repositories.
- **Git** - An open source distributed version control system.

## Epics

*Connect to an Amazon Elastic Compute Cloud (Amazon EC2) instance*

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to an Amazon EC2 instance by using SSH.</td>
<td>Connect to an EC2 instance by using SSH and a key pair file.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

### Install Git

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Git.</td>
<td>Install Git by using the &quot;yum install git -y&quot; command.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>
Clone Git repository and install git-secrets

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone Git source repository.</td>
<td>Clone the Git repository you want to scan. Choose the “Git clone” command from your home directory.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Install git-secrets.</td>
<td>Install git-secrets by cloning the source Git repository. For more information, see the &quot;Related resources&quot; section.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Scan git code repository

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to the source repository.</td>
<td>Switch to the folder for the Git repository you want to scan: “cd &lt;code_repository&gt;”.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Register the AWS rule set (Git hooks).</td>
<td>To configure git-secrets to scan your Git repository on each commit, run the command “git secrets --register-aws”.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Scan the repository.</td>
<td>Run the command “git secrets – scan”.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Review output file.</td>
<td>The tool generates an output file if it finds a vulnerability in your Git repository. For an example, see the attached file, output.txt.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Related resources

- View the Source Code for this Quick Start
- Install git-secrets

Attachments

attachment.zip

Send alerts from AWS Network Firewall to a Slack channel

Created by Venki Srivatsav (AWS)
Code repository: NfwSlackIntegration

Environment: PoC or pilot

Technologies: Security, identity, compliance; Networking

AWS services: AWS Lambda; AWS Network Firewall; Amazon S3

Summary

This pattern describes how to deploy a firewall by using Amazon Web Services (AWS) Network Firewall with the distributed deployment model and how to propagate the alerts generated by AWS Network Firewall to a configurable Slack channel.

Compliance standards such as Payment Card Industry Data Security Standard (PCI DSS) require that you install and maintain a firewall to protect customer data. In the AWS Cloud, a virtual private cloud (VPC) is considered the same as a physical network in the context of these compliance requirements. You can use Network Firewall to monitor network traffic between VPCs and to protect your workloads that run in VPCs governed by a compliance standard. Network Firewall blocks access or generates alerts when it detects unauthorized access from other VPCs in the same account. However, Network Firewall supports a limited number of destinations for delivering the alerts. These destinations include Amazon Simple Storage Service (Amazon S3) buckets, Amazon CloudWatch log groups, and Amazon Kinesis Data Firehose delivery streams. Any further action on these notifications requires offline analysis by using either Amazon Athena or Amazon Kinesis.

This pattern provides a method for propagating alerts that are generated by Network Firewall to a configurable Slack channel for further action in near real time. You can also extend the functionality to other alerting mechanisms such as PagerDuty, Jira, and email. (Those customizations are outside the scope of this pattern.)

Prerequisites and limitations

Prerequisites

- Slack channel (see Getting started in the Slack help center)
- Required privileges to send a message to the channel
- The Slack endpoint URL with an API token (select your app and choose an incoming webhook to see its URL; for more information, see Creating an Incoming Webhook in the Slack API documentation)
- An Amazon Elastic Compute Cloud (Amazon EC2) test instance in the workload subnets
- Test rules in Network Firewall
- Actual or simulated traffic to trigger the test rules
- An S3 bucket to hold the source files to be deployed

Limitations

- Currently this solution supports only a single Classless Inter-Domain Routing (CIDR) range as a filter for source and destination IPs.

Architecture

Target technology stack
- One VPC
- Four subnets (two for the firewall and two for workloads)
- Internet gateway
- Four route tables with rules
- S3 bucket used as an alert destination, configured with a bucket policy and event settings to run a Lambda function
- Lambda function with an execution role, to send Slack notifications
- AWS Secrets Manager secret for storing the Slack URL
- Network firewall with alert configuration
- Slack channel

All components except for the Slack channel are provisioned by the CloudFormation templates and the Lambda function that are provided with this pattern (see the Code (p. 1715)section).

Target architecture

This pattern sets up a decentralized network firewall with Slack integration. This architecture consists of a VPC with two Availability Zones. The VPC includes two protected subnets and two firewall subnets with network firewall endpoints. All traffic going into and out of the protected subnets can be monitored by creating firewall policies and rules. The network firewall is configured to place all alerts in an S3 bucket. This S3 bucket is configured to call a Lambda function when it receives a put event. The Lambda function fetches the configured Slack URL from Secrets Manager and sends the notification message to the Slack workspace.
For more information about this architecture, see the AWS blog post Deployment models for AWS Network Firewall.

**Tools**

**AWS services**

- **AWS Network Firewall** – AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for VPCs in the AWS Cloud. You can use Network Firewall to filter traffic at the perimeter of your VPC and protect your workloads on AWS.

- **AWS Secrets Manager** – AWS Secrets Manager is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. This pattern uses Secrets Manager to store the Slack URL.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses Amazon S3 to store the CloudFormation templates and Python script for the Lambda function. It also uses an S3 bucket as the network firewall alert destination.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. This pattern uses AWS CloudFormation to automatically deploy a distributed architecture for Firewall Manager.

**Code**

The code for this pattern is available on GitHub, in the Network Firewall Slack Integration repository. In the src folder of the repository, you’ll find:

- A set of CloudFormation files in YAML format. You use these templates to provision the components for this pattern.
- A Python source file (slack-lambda.py) to create the Lambda function.
- A .zip archive deployment package (slack-lambda.py.zip) to upload your Lambda function code.

To use these files, follow the instructions in the next section.

**Epics**

**Set up the S3 bucket**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an S3 bucket. | 1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.  
2. Choose or create an S3 bucket to host the code. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. | App developer, App owner, Cloud administrator |
### Task: Upload the CloudFormation templates and Lambda code.

1. Download the following files from the GitHub repository for this pattern:
   - base.yml
   - igw-ingress-route.yml
   - slack-lambda.py
   - slackLambda.yml
   - decentralized-deployment.yml
   - protected-subnet-route.yml
   - slack-lambda.py.zip
2. Upload the files to the S3 bucket you created.

### Skills required:
- App developer
- App owner
- Cloud administrator

### Task: Deploy the CloudFormation template

1. Open the AWS CloudFormation console in the same AWS Region as your S3 bucket and deploy the template base.yml. This template creates the required AWS resources and Lambda functions for the alerts to be transmitted to the Slack channel.

For more information about deploying CloudFormation templates, see Creating a stack on the AWS CloudFormation console in the CloudFormation documentation.

### Skills required:
- App developer
- App owner
- Cloud administrator

### Task: Complete the parameters in the template.

Specify the stack name and configure the parameter values. For a list of parameters, their descriptions, and default values, see CloudFormation.

### Skills required:
- App developer
- App owner
- Cloud administrator
## Create the stack.

1. Review stack details and update values based on your environment requirements.
2. Choose **Create stack** to deploy the template.

### Skills required
App developer, App owner, Cloud administrator

### Additional information
- **parameters** in the Additional information (p. 1719) section.
- **Target technology stack** (p. 1713)
- **Launch an instance** in the Amazon EC2 documentation
- **Regions and Zones** in the Amazon EC2 documentation

## Verify the solution

### Test the deployment.

1. Use the AWS CloudFormation console or the AWS Command Line Interface (AWS CLI) to verify that the resources listed in the **Target technology stack** (p. 1713) section have been created.

   If the CloudFormation template fails to deploy successfully, check the values you provided for the **pAvailabilityZone1** and **pAvailabilityZone2** parameters. These should be appropriate for the AWS Region you’re deploying the solution in. For a list of Availability Zones for each Region, see **Regions and Zones** in the Amazon EC2 documentation.

### Skills required
App developer, App owner, Cloud administrator

### Test functionality.

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).

2. Create an EC2 instance in one of the protected subnets. Choose an Amazon Linux 2 AMI (HVM) to use as an HTTPS server. For instructions, see **Launch an instance** in the Amazon EC2 documentation.

3. Use the following user data to install a web server on the EC2 instance:

```bash
#!/bin/bash
yum install httpd -y
systemctl start httpd
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| systemctl stop firewalld  
cd /var/www/html  
echo "Hello!! this is a NFW alert test page, 200 OK" > index.html | | |

4. Create the following network firewall rules:

**Stateless rule:**

- Source: 0.0.0.0/0
- Destination 10.0.3.65/32 (private IP of the EC2 instance)
- Action: Forward

**Stateful rule:**

- Protocol: HTTP
- Source ip/port: Any / Any
- Destination ip/port: Any / Any

5. Get the public IP of the web server you created in step 3.

6. Access the public IP in a browser. You should see the following message in the browser:

Hello!! this is a NFW alert test page, 200 OK

You will also get a notification in the Slack channel. The notification might be delayed, depending on the size of the message. For testing purposes, consider providing a CIDR filter that is not too narrow (for example, a CIDR value with /32 would be considered too narrow, and /8 would be too broad). For more information, see the *Filter behavior* section in Additional information (p. 1719).

### Related resources

- Deployment models for AWS Network Firewall ([AWS blog post](https://aws.amazon.com/blogs/security/deployment-models-for-aws-network-firewall/))
- AWS Network Firewall policies ([AWS documentation](https://docs.aws.amazon.com/wireguard/latest/userguide/))
Additional information

CloudFormation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default or sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pVpcName</td>
<td>The name of the VPC to create.</td>
<td>Inspection</td>
</tr>
<tr>
<td>pVpcCidr</td>
<td>The CIDR range for the VPC to create.</td>
<td>10.0.0.0/16</td>
</tr>
<tr>
<td>pVpcInstanceTenancy</td>
<td>How EC2 instances are distributed across physical hardware. Options are <strong>default</strong> (shared tenancy) or <strong>dedicated</strong> (single tenancy).</td>
<td>default</td>
</tr>
<tr>
<td>pAvailabilityZone1</td>
<td>The first Availability Zone for the infrastructure.</td>
<td>us-east-2a</td>
</tr>
<tr>
<td>pAvailabilityZone2</td>
<td>The second Availability Zone for the infrastructure.</td>
<td>us-east-2b</td>
</tr>
<tr>
<td>pNetworkFirewallSubnet1Cidr</td>
<td>The CIDR range for the first firewall subnet (minimum /28).</td>
<td>10.0.1.0/24</td>
</tr>
<tr>
<td>pNetworkFirewallSubnet2Cidr</td>
<td>The CIDR range for the second firewall subnet (minimum /28).</td>
<td>10.0.2.0/24</td>
</tr>
<tr>
<td>pProtectedSubnet1Cidr</td>
<td>The CIDR range for the first protected (workload) subnet.</td>
<td>10.0.3.0/24</td>
</tr>
<tr>
<td>pProtectedSubnet2Cidr</td>
<td>The CIDR range for the second protected (workload) subnet.</td>
<td>10.0.4.0/24</td>
</tr>
<tr>
<td>pS3BucketName</td>
<td>The name of the existing S3 bucket where you uploaded the Lambda source code.</td>
<td>us-w2-yourname-lambda-functions</td>
</tr>
<tr>
<td>pS3KeyPrefix</td>
<td>The prefix of the S3 bucket where you uploaded the Lambda source code.</td>
<td>aod-test</td>
</tr>
<tr>
<td>pAWSSecretName4Slack</td>
<td>The name of the secret that holds the Slack URL.</td>
<td>SlackEnpoint-Cfn</td>
</tr>
<tr>
<td>pSlackChannelName</td>
<td>The name of the Slack channel you created.</td>
<td>somename-notifications</td>
</tr>
<tr>
<td>pSlackUserName</td>
<td>Slack user name.</td>
<td>Slack User</td>
</tr>
<tr>
<td>pSecretKey</td>
<td>This can be any key. We recommend that you use the default.</td>
<td>webhookUrl</td>
</tr>
</tbody>
</table>
### Parameter | Description | Default or sample value
--- | --- | ---
pWebHookUrl | The value of the Slack URL. | https://hooks.slack.com/services/T???9T??/A031885JRM7/9D4Y?????
pAlertS3Bucket | The name of the S3 bucket to be used as the network firewall alert destination. This bucket will be created for you. | us-w2-yourname-security-aod-alerts
pSecretTagName | The tag name for the secret. | AppName
pSecretTagValue | The tag value for the specified tag name. | LambdaSlackIntegration
pdestCidr | The filter for the destination CIDR range. For more information, see the next section, Filter behavior. | 10.0.0.0/16
pdestCondition | A flag to indicate whether to exclude or include the destination match. For more information, see the next section. Valid values are include and exclude. | include
psrcCidr | The filter for the source CIDR range to alert. For more information, see the next section. | 118.2.0.0/16
psrcCondition | The flag to exclude or include the source match. For more information, see the next section. | include

### Filter behavior
If you haven’t configured any filters in AWS Lambda, all generated alerts are sent to your Slack channel. The source and destination IPs of the generated alerts are matched against the CIDR ranges you configured when you deployed the CloudFormation template. If a match is found, the condition is applied. If either the source or the destination falls within the configured CIDR range and at least one of them is configured with the condition include, an alert is generated. The following tables provide examples of CIDR values, conditions, and results.

<table>
<thead>
<tr>
<th>Source</th>
<th>Configured CIDR</th>
<th>Alert IP</th>
<th>Configured</th>
<th>Alert</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>10.0.0.25</td>
<td>include</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Configured CIDR</th>
<th>Alert IP</th>
<th>Configured</th>
<th>Alert</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>10.0.0.25</td>
<td>exclude</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Simplify private certificate management by using ACM Private CA and AWS RAM

Created by Everett Hinckley (AWS) and Vivek Goyal (AWS)

Code repository: ACMPCA Hierarchy

Environment: Production

Technologies: Security, identity, compliance; Infrastructure; Migration

AWS services: AWS Certificate Manager; AWS Organizations; AWS RAM

Summary

You can use Amazon Web Services (AWS) Certificate Manager Private Certificate Authority (ACM Private CA) to issue private certificates for authenticating internal resources and signing computer code. This pattern provides an AWS CloudFormation template for the rapid deployment of a multi-level CA hierarchy and consistent provisioning experience. Optionally, you can use AWS Resource Access Manager (AWS RAM) to securely share the CA within your organizations or organizational units (OUs) in AWS Organizations, and centralize the CA while using AWS RAM to manage permissions. There is no need for a private CA in every account, so this approach saves you money. Additionally, you can use Amazon Simple Storage Service (Amazon S3) to store the certificate revocation list (CRL) and access logs.
This implementation provides the following features and benefits:

- Centralizes and simplifies the management of the private CA hierarchy by using ACM Private CA.
- Exports certificates and keys to customer-managed devices on AWS and on premises.
- Uses an AWS CloudFormation template for a rapid deployment and consistent provisioning experience.
- Creates a private root CA along with 1, 2, 3, or 4 subordinate CA hierarchy.
- Optionally, uses AWS RAM to share the end-entity subordinate CA with other accounts at the organization or OU level.
- Saves money by removing the need for a private CA in every account by using AWS RAM.
- Creates an optional S3 bucket for the CRL.
- Creates an optional S3 bucket for CRL access logs.

**Prerequisites and limitations**

**Prerequisites**

If you want to share the CA within an AWS Organizations structure, identify or set up the following:

- A security account for creating the CA hierarchy and share.
- A separate OU or account for testing.
- Sharing enabled within the AWS Organizations management account. For more information, see Enable resource sharing within AWS Organizations in the AWS RAM documentation.

**Limitations**

- CAs are regional resources. All CAs reside in a single AWS account and in a single AWS Region.
- User-generated certificates and keys are not supported. For this use case, we recommend that you customize this solution to use an external root CA.
- A public CRL bucket is not supported. We recommend that you keep the CRL private. If internet access to the CRL is required, see the section on using Amazon CloudFront to serve CRLs in Enabling the S3 Block Public Access (BPA) feature in the ACM Private CA documentation.

**Architecture**

**Target technology stack**

- ACM Private CA
- AWS RAM
- Amazon S3
- AWS Organizations
- AWS CloudFormation

**Target architecture**

This pattern provides two options for sharing to AWS Organizations:

**Option 1** — Create the share at the organization level. All accounts in the organization can issue the private certificates by using the shared CA, as shown in the following diagram.
**Option 2** — Create the share at the organizational unit (OU) level. Only the accounts in the specified OU can issue the private certificates by using the shared CA. For example, in the following diagram, if the share is created at the Sandbox OU level, both Developer 1 and Developer 2 can issue private certificates by using the shared CA.
Tools

AWS services

- **ACM Private CA** – AWS Certificate Manager Private Certificate Authority (ACM Private CA) is a hosted private CA service for issuing and revoking private digital certificates. It helps you create private CA hierarchies, including root and subordinate CAs, without the investment and maintenance costs of operating an on-premises CA.

- **AWS RAM** – AWS Resource Access Manager (AWS RAM) helps you securely share your resources across AWS accounts and within your organization or OUs in AWS Organizations. To reduce operational overhead in a multi-account environment, you can create a resource and use AWS RAM to share that resource across accounts.

- **AWS Organizations** – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.
AWS Prescriptive Guidance Patterns

Epics

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses Amazon S3 to store the certificate revocation list (CRL) and access logs.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. This pattern uses AWS CloudFormation to automatically deploy a multi-level CA hierarchy.

**Code**

The source code for this pattern is available on GitHub, in the ACM Private CA hierarchy repository. The repository includes:

- The AWS CloudFormation template ACMPCA-RootCASubCA.yaml. You can use this template to deploy the CA hierarchy for this implementation.

- Test files for use cases such as requesting, exporting, describing, and deleting a certificate.

To use these files, follow the instructions in the *Epics* section.

**Epics**

**Architect the CA hierarchy**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect certificate subject information.</td>
<td>Gather certificate subject information about the certificate owner: organization name, organization unit, country, state, locality, and common name.</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
<tr>
<td>Collect optional information about AWS Organizations.</td>
<td>If the CA will be part of an AWS Organizations structure and you want to share the CA hierarchy inside that structure, collect the management account number, the organization ID, and optionally the OU ID (if you want to share the CA hierarchy only with a specific OU). Also, determine the AWS Organizations accounts or OUs, if any, that you want to share the CA with.</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
<tr>
<td>Design the CA hierarchy.</td>
<td>Determine which account will house the root and subordinate CAs. Determine how many subordinate levels the hierarchy requires between the root and the end-entity certificates. For more information, see Designing a CA hierarchy in the ACM Private CA documentation.</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
</tbody>
</table>
### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine naming and tagging conventions for the CA hierarchy.</td>
<td>Determine the names for the AWS resources: the root CA and each subordinate CA. Determine which tags should be assigned to each CA.</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
<tr>
<td>Determine required encryption and signing algorithms.</td>
<td>Determine the following:</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
<tr>
<td></td>
<td>• Your organization's encryption algorithm requirements for the public keys that your CA uses when it issues a certificate. The default is RSA_2048.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The key algorithm that your CA uses for certificate signing. The default is SHA256WITHRSA.</td>
<td></td>
</tr>
<tr>
<td>Determine certificate revocation requirements for the CA hierarchy.</td>
<td>If certificate revocation capabilities are required, establish a naming convention for the S3 bucket that contains the certificate revocation list (CRL).</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
<tr>
<td>Determine the logging requirements for the CA hierarchy.</td>
<td>If access logging capabilities are required, establish a naming convention for the S3 bucket that contains the access logs.</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
<tr>
<td>Determine certificate expiration periods.</td>
<td>Determine the expiration date for the root certificate (the default is 10 years), end-entity certificates (the default is 13 months), and subordinate CA certificates (the default is 3 years). Subordinate CA certificates should expire earlier than the CA certificates at higher levels in the hierarchy. For more information, see Managing the private CA lifecycle in the ACM Private CA documentation.</td>
<td>Cloud architect, Security architect, PKI engineer</td>
</tr>
</tbody>
</table>

#### Deploy the CA hierarchy

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete prerequisites.</td>
<td>Complete the steps in the Prerequisites (p. 1722) section of this pattern.</td>
<td>Cloud administrator, Security engineers, PKI engineers</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>
| Create CA roles for various personas. | 1. Determine the types of AWS Identity and Access Management (IAM) roles or AWS Single Sign-On (AWS SSO) users required to administer the various levels of the CA hierarchy, such as RootCAAdmin, SubordinateCAAdmin, and CertificateConsumer.  
2. Determine the granularity of policies needed to separate duties.  
3. Create the required IAM roles or AWS SSO users in the account that the CA hierarchy resides in. | Cloud administrator, Security engineers, PKI engineers |
| Deploy the CloudFormation stack. | 1. From the GitHub repository for this pattern, download the ACMPCA-RootCASubCA.yaml template.  
2. Deploy the template from the AWS CloudFormation console or from the AWS Command Line Interface (AWS CLI). For more information, see Working with stacks in the CloudFormation documentation.  
3. Complete the parameters in the template, including the organization name, the OU name, the key algorithm, the signing algorithm, and other options. | Cloud administrator, Security engineers, PKI engineers |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architect a solution for updating certificates used by user-managed resources.</strong></td>
<td>Resources of integrated AWS services, such as Elastic Load Balancing, update certificates automatically before expiration. However, user-managed resources, such as web servers that are running on Amazon Elastic Compute Cloud (Amazon EC2) instances, require another mechanism. 1. Determine which user-managed resources require end-entity certificates from the private CA. 2. Plan a process to be notified about the expiration of user-managed resources and certificates. For examples, see the following: • Using an AWS Config managed rule • Using Amazon CloudWatch and Amazon EventBridge 3. Write custom scripts to update certificates on user-managed resources and integrate them with AWS services to automate the updates. For more information about integrated AWS Services, see Services integrated with AWS Certificate Manager in the ACM documentation.</td>
<td>Cloud administrator, Security engineers, PKI engineers</td>
</tr>
</tbody>
</table>

**Validate and document the CA hierarchy**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Validate optional AWS RAM sharing.</strong></td>
<td>If the CA hierarchy is shared with other accounts in AWS Organizations, log in to one of those accounts from the AWS Management Console, navigate to the <strong>ACM Private CA console</strong>, and confirm that the newly created CA is shared to this account. Only the lowest-level CA in the hierarchy will be visible, because that is the CA that generates the end-entity certificates.</td>
<td>Cloud administrator, Security engineers, PKI engineers</td>
</tr>
</tbody>
</table>
### Task | Description | Skills required
--- | --- | ---
| Validate the CA hierarchy with certificate lifecycle tests. | In the GitHub repository for this pattern, locate the lifecycle tests. Run the tests from the AWS CLI to request a certificate, export a certificate, describe a certificate, and delete a certificate. | Cloud administrator, Security engineers, PKI engineers
| Import the certificate chain into trust stores. | For browsers and other applications to trust a certificate, the certificate’s issuer must be included in the browser’s trust store, which is a list of trusted CAs. Add the certificate chain for the new CA hierarchy to your browser’s and application’s trust store. Confirm that the end-entity certificates are trusted. | Cloud administrator, Security engineers, PKI engineers
| Create a runbook to document the CA hierarchy. | Create a runbook document to describe the architecture of the CA hierarchy, the account structure that can request end-entity certificates, the build process, and basic management tasks such as issuing end-entity certificates (unless you want to allow self-service by child accounts), usage, and tracking. | Cloud administrator, Security engineers, PKI engineers

### Related resources
- Designing a CA hierarchy ([ACM Private CA documentation](#))
- Creating a private CA ([ACM Private CA documentation](#))
- How to use AWS RAM to share your ACM Private CA cross-account ([AWS blog post](#))
- ACM Private CA best practices ([AWS blog post](#))
- Enable resource sharing within AWS Organizations ([AWS RAM documentation](#))
- Managing the private CA lifecycle ([ACM Private CA documentation](#))
- `acm-certificate-expiration-check` for AWS Config ([AWS Config documentation](#))
- AWS Certificate Manager now provides certificate expiry monitoring through Amazon CloudWatch ([AWS announcement](#))
- Services integrated with AWS Certificate Manager ([ACM documentation](#))

### Additional information
When you export certificates, use a passphrase that is cryptographically strong and aligns with your organization’s data loss prevention strategy.
Turn off security standard controls across all Security Hub member accounts in a multi-account environment

Created by Michael Fuellbier (AWS) and Ahmed Bakry (AWS)

**Environment:** Production  
**Technologies:** Security, identity, compliance; Serverless  
**AWS services:** Amazon DynamoDB; Amazon EventBridge; AWS Lambda; AWS Security Hub; AWS Step Functions

### Summary

In the Amazon Web Services (AWS) Cloud, AWS Security Hub standard controls, such as [CIS AWS Foundations Benchmark](https://aws.amazon.com/cis/) or [AWS Foundational Security Best Practices](https://aws.amazon.com/foundation/), can only be turned off (disabled) manually from within a single AWS account. In a multi-account environment, you can't turn off the controls across multiple Security Hub member accounts with "one click" (that is, one API call). This pattern demonstrates how to turn off the Security Hub standard controls across all the Security Hub member accounts managed by your Security Hub administrator account with one click.

### Prerequisites and limitations

**Prerequisites**

- A multi-account environment consisting of a Security Hub administrator account that manages multiple member accounts
- AWS Command Line Interface (AWS CLI) version 2, installed
- AWS SAM CLI, installed

**Limitations**

- This pattern works only in a multi-account environment where a single Security Hub administrator account manages multiple member accounts.
- If you change a lot of controls in a very short timeframe, the event initiation causes multiple parallel invocations, which can lead to API throttling and cause the invocations to fail. For example, this can happen if you programmatically change a lot of controls by using the [Security Hub Controls CLI](https://aws.amazon.com/cli/).

### Architecture

The following diagram shows an example of an AWS Step Functions workflow that turns off Security Hub standard controls across multiple Security Hub member accounts (as viewed from the Security Hub administrator account).
The diagram includes the following workflow:

1. An Amazon EventBridge rule is initiated on a daily schedule and invokes the state machine. **Note:** You can modify the timing of the rule by updating the **Schedule** parameter in your AWS CloudFormation template.

2. An EventBridge rule is initiated whenever a control is turned on or off in the Security Hub administrator account.

3. A Step Functions state machine propagates the status of the security standard controls (that is, controls that are turned on or off) from the Security Hub administrator account to the member accounts.

4. A cross-account AWS Identity and Access Management (IAM) role is deployed in each member account and assumed by the state machine. The state machine turns the controls on or off in each member account.

5. An Amazon DynamoDB table contains exceptions and information about which controls to turn on or off in a particular account. This information overrides the configurations fetched from the Security Hub administrator account for the specified member account.

**Note:** The purpose of the scheduled EventBridge rule is to ensure that newly added Security Hub member accounts have the same control status as existing accounts.

**Technology stack**

- Amazon DynamoDB
- Amazon EventBridge
- AWS CLI
- AWS Lambda
- AWS SAM CLI
- AWS Security Hub
- AWS Step Functions
Tools

- **Amazon DynamoDB** – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

- **Amazon EventBridge** – Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your applications, software as a service (SaaS) applications, and AWS services to targets such as Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other accounts.

- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- **AWS SAM** – AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS. AWS SAM provides you with a template specification to define your serverless application, and a command line interface (CLI) tool.

- **AWS Security Hub** – AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.

- **AWS Step Functions** – AWS Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications. Through the Step Functions graphical console, you see your application's workflow as a series of event-driven steps.

**Code**

The code for this pattern is available on the GitHub [AWS Security Hub Cross-Account Controls Disabler](https://github.com/aws-samples/aws-securityhub-cross-account-controls-disabler) repository. The code repository contains the following files and folders, which are applicable to this pattern:

- **UpdateMembers/template.yaml** – This file contains components deployed in the Security Hub administrator account, including the Step Functions state machine and the EventBridge rules.
- **member-iam-role/template.yaml** – This file contains the code to deploy the cross-account IAM role in a member account.
- **stateMachine.json** – This file defines the state machine's workflow.
- **GetMembers/index.py** – This file contains the code for the **GetMembers** state machine. A script retrieves the status of the security standard controls in all existing Security Hub member accounts.
- **UpdateMember/index.py** – This file contains a script that updates the control status in each member account.
- **CheckResult/index.py** – This file contains a script that checks the status of the workflow invocation (accepted or failed).
# Epics

## Deploy a cross-account IAM role in the Security Hub member accounts

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the account ID of the Security Hub administrator account.</td>
<td>Set up a Security Hub administrator account and then note the account ID of the administrator account.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

Deploy the AWS CloudFormation template that includes the cross-account IAM role in the member accounts.

To deploy the `member-iam-role/template.yaml` AWS CloudFormation template in all the member accounts managed by the Security Hub administrator account, run the following command:

```shell
aws cloudformation deploy --template-file member-iam-role/template.yaml --capabilities CAPABILITY_NAMED_IAM --stack-name <your-stack-name> --parameter-overrides SecurityHubAdminAccountId=<your-account-ID>
```

The `SecurityHubAdminAccountId` parameter must match the Security Hub administrator account ID that you noted earlier.

**Note:** Because Security Hub is a Regional service, you must deploy the template individually in each AWS Region.

## Deploy a state machine in the Security Hub administrator account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package the AWS CloudFormation template that includes the state machine with AWS SAM.</td>
<td>To package the <code>UpdateMembers/template.yaml</code> AWS CloudFormation template in the Security Hub administrator account, run the following command:</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>

```shell
sam package --template-file UpdateMembers/template.yaml
```
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Task** Deploy the packaged AWS CloudFormation template in the Security Hub administrator account. | **AWS DevOps**

To deploy the AWS CloudFormation template in the Security Hub administrator account, run the following command:

```bash
aws cloudformation deploy --template-file UpdateMembers/template-out.yaml --s3-bucket <your-s3-bucket-name>
```

**Note:** Your Amazon Simple Storage Service (Amazon S3) bucket must be in the same AWS Region where you deploy the AWS CloudFormation template.

In the member-iam-role/template.yaml template, the `MemberIAMRolePath` parameter must match the `IAMRolePath` parameter. The `MemberIAMRoleName` must match the `IAMRoleName` parameter.

### Related resources

- Designating a Security Hub administrator account ([AWS Security Hub documentation](#))
- Handling Errors, Retries, and adding Alerting to Step Function State Machine Executions ([AWS blog post](#))

### Use Network Firewall to capture the DNS domain names from the Server Name Indication (SNI) for outbound traffic

*Created by Kirankumar Chandrashekar (AWS)*

<table>
<thead>
<tr>
<th>Environment: PoC or pilot</th>
<th>Technologies: Security, identity, compliance; Networking; Websites &amp; web apps</th>
<th>Workload: All other workloads</th>
</tr>
</thead>
</table>

1734
Summary

This pattern shows you how to use Amazon Web Services (AWS) Network Firewall to collect the DNS domain names that are provided by the Server Name Indication (SNI) in the HTTPS header of your outbound network traffic. Network Firewall is a managed service that makes it easy to deploy critical network protections for Amazon Virtual Private Cloud (Amazon VPC), including the ability to secure outbound traffic with a firewall that blocks packets that fail to meet certain security requirements. Securing outbound traffic to specific DNS domain names is called egress filtering, which is the practice of monitoring and potentially restricting the flow of outbound information from one network to another.

After you capture the SNI data that passes through Network Firewall, you can use Amazon CloudWatch Logs and AWS Lambda to publish the data to an Amazon Simple Notification Service (Amazon SNS) topic that generates email notifications. The email notifications include the server name and other relevant SNI information. Additionally, you can use the output of this pattern to allow or restrict outbound traffic by domain name in the SNI by using firewall rules. For more information, see Working with stateful rule groups in AWS Network Firewall in the Network Firewall documentation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows
- Network Firewall, set up and configured in Amazon VPC and in use for inspecting outbound traffic

Note: Network Firewall can use any of the following VPC configurations:

- Simple single zone architecture with an internet gateway
- Multi zone architecture with an internet gateway
- Architecture with an internet gateway and a NAT gateway

Architecture

The following diagram shows how to use Network Firewall to collect SNI data from outbound network traffic, and then publish that data to an SNS topic by using CloudWatch Logs and Lambda.

![Architecture Diagram]
The diagram shows the following workflow:

1. Network Firewall collects domain names from the SNI data in the HTTPS header of your outbound network traffic.
2. CloudWatch Logs monitors the SNI data and invokes a Lambda function whenever the outbound network traffic passes through Network Firewall.
3. The Lambda function reads the SNI data captured by CloudWatch Logs and then publishes that data to an SNS topic.
4. The SNS topic sends you an email notification that includes the SNI data.

Automation and scale

- You can use AWS CloudFormation to create this pattern by using infrastructure as code.

Technology stack

- Amazon CloudWatch Logs
- Amazon SNS
- Amazon VPC
- AWS Lambda
- AWS Network Firewall

Tools

AWS services

- Amazon CloudWatch Logs – You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS CloudTrail, Amazon Route 53, and other sources.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).
- Amazon VPC – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you’ve defined. This virtual network closely resembles a traditional network that you’d operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
- AWS Lambda – AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- AWS Network Firewall – AWS Network Firewall is a managed service that makes it easy to deploy essential network protections for all of your Amazon VPCs.

Epics

Create a CloudWatch log group for Network Firewall

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CloudWatch log group.</td>
<td>1. Sign in to the AWS Management Console and open the CloudWatch console.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>
### Create an SNS topic and subscription

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an SNS topic.</td>
<td>To create an SNS topic, follow the instructions in the Amazon SNS documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Subscribe an endpoint to the SNS topic.</td>
<td>To subscribe an email address as an endpoint to the SNS topic that you created, follow the instructions in the Amazon SNS documentation. For Protocol, choose Email/Email-JSON. Note: You can also choose a different endpoint based on your requirements.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

### Set up logging in Network Firewall

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Enable firewall logging.                       | 1. Sign in to the AWS Management Console and open the Amazon VPC console.  
2. In the navigation pane, under NETWORK FIREWALL, choose Firewalls.  
3. In the Firewalls section, choose the firewall where you want to capture the server name from the SNI for outbound traffic.  
4. Choose the Firewall details tab, and then choose Edit in the Logging section. | Cloud administrator           |
### Task | Description | Skills required
--- | --- | ---
5. For **Log type**, select **Alert**. For **Log destination for alerts**, select **CloudWatch log group**.  
6. For **CloudWatch log group**, search for and choose the log group that you created earlier, and then choose **Save**.  

For more information about using CloudWatch Logs as a log destination for Network Firewall, see [Amazon CloudWatch Logs in the Network Firewall documentation](https://docs.aws.amazon.com/)

---

**Set up a stateful rule in Network Firewall**

### Task | Description | Skills required
--- | --- | ---
Create a stateful rule.  
1. Sign in to the AWS Management Console and open the [Amazon VPC console](https://docs.aws.amazon.com/vpc/).  
2. In the navigation pane, under **NETWORK FIREWALL**, choose **Network Firewall Rule Groups**.  
3. Choose **Create Network Firewall rule group**.  
4. On the **Create Network Firewall rule group** page, for the **Rule group type**, choose **Stateful rule group**.  
**Note:** For more information, see [Working with stateful rule groups in AWS Network Firewall](https://docs.aws.amazon.com/).  
5. In the **Stateful rule group** section, enter a name and description for the rule group.  
6. For **Capacity**, set the maximum capacity that you want to allow for the stateful rule group (up to the maximum of 30,000).  
**Note:** You can't change this setting after you create the rule group. For information about how to calculate capacity, see [Setting rule group capacity](https://docs.aws.amazon.com/).  

Cloud administrator
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Sign in to the AWS Management Console and open the Amazon VPC console.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>2.</td>
<td>In the navigation pane, under NETWORK FIREWALL, choose Firewalls.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>3.</td>
<td>Choose the firewall where you want to capture the server name from the SNI for outbound traffic.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>4.</td>
<td>In the Stateful rule groups section, choose Actions, and then choose Add unmanaged stateful rule groups.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>5.</td>
<td>On the Add unmanaged stateful rule groups page, select the stateful rule group that you created earlier, and then choose Add stateful rule group.</td>
<td>Cloud administrator</td>
</tr>
</tbody>
</table>

### Create a Lambda function to read the logs

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the code for the Lambda function.</td>
<td>In an integrated development environment (IDE) that can read logs.</td>
<td>App developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>the CloudWatch Logs event from Network Firewall for outbound traffic, paste in the following Python 3 code and replace <code>&lt;SNS-topic-ARN&gt;</code> with your value:</td>
<td></td>
</tr>
</tbody>
</table>

```python
import json
import gzip
import base64
import boto3
sns_client = boto3.client('sns')
def lambda_handler(event, context):
    decoded_event =
    json.loads(gzip.decompress(base64.b64decode(event['awslogs']['data']))
    body = '',
    (filtermatch) '''.format(loggroup=decoded_event['logGroup'], logstream=decoded_event['logStream'], filtermatch=decoded_event['logEvents'][0]['message'],
    )
    print(body)
    filterMatch =
    json.loads(body)
    data = []
    if 'http' in filterMatch['event']:
        data.append(filterMatch['event']['http']['hostname'])
    elif 'tls' in filterMatch['event']:
        data.append(filterMatch['event']['tls']['sni'])
    result = 'Domain accessed ' + 1*' ' + (data[0]) + 1*' ' + 'via AWS Network Firewall ' + 1*' ' + (filterMatch['firewall_name'])
    print(result)
    message = {'ServerName': result}
    send_to_sns = sns_client.publish(
        TargetArn=<SNS-topic-ARN>,
        #Replace with the SNS topic ARN
        Message=json.dumps({'default':
        json.dumps(message),
```
This code sample parses the CloudWatch Logs content and captures the server name provided by the SNI in the HTTPS header.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the Lambda function.</td>
<td>To create the Lambda function, follow the instructions in the Lambda documentation and choose Python 3.9 for Runtime.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Add the code to the Lambda function.</td>
<td>To add your Python code to the Lambda function that you created earlier, follow the instructions in the Lambda documentation.</td>
<td>Cloud administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| Add CloudWatch Logs as a trigger to the Lambda function. | 1. Sign in to the AWS Management Console and open the Lambda console.  
2. In the navigation pane, choose Functions, and then choose the function that you created earlier.  
3. In the Function overview section, choose Add trigger.  
4. On the Add trigger page, in the Trigger configuration section, choose CloudWatch Logs, and then choose Add.  
5. For Log group, choose the CloudWatch log group that you created earlier.  
6. For Filter name, enter a name for your filter.  
7. Choose Add.  
8. On the Configuration tab of your function’s page, in the Triggers section, select the trigger that you just added, and then choose Enable.  

For more information, see Using Lambda with CloudWatch Logs in the Lambda documentation. | Cloud administrator |
### AWS Prescriptive Guidance Patterns

#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Add SNS publish permissions.  | Add the `sns:Publish` permission to the Lambda execution role, so that Lambda can make API calls to publish messages to SNS.  
1. Find the execution role of the Lambda function that you created earlier.  
2. Add the following policy to your AWS Identity and Access Management (IAM) role:  
   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "AllowSNSPublish",
         "Effect": "Allow",
         "Action": [
           "sns:GetTopicAttributes",
           "sns:Subscribe",
           "sns:Unsubscribe",
           "sns:Publish"
         ],
         "Resource": "*
       }
     ]
   }
   ```  | Cloud administrator |

#### Test the functionality of your SNS notification

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Send traffic through Network Firewall.  | 1. Send or wait for HTTPS traffic to pass through Network Firewall.  
2. Check the SNS notification email that you receive from AWS when traffic passes through Network Firewall. The email includes the SNI details for outbound traffic. For example, the email generated from the Lambda code above will have the following content if the accessed domain name is https://aws.amazon.com  | Test engineer |

1743
and the subscription protocol is **EMAIL-JSON**:

```
{
    "Type": "Notification",
    "MessageId": "<messageID>",
    "Subject": "Server Name passed through the Network Firewall",
    "Message": "{}
        "ServerName": "Domain 'aws.amazon.com' accessed via AWS Network Firewall 'AWS-Network-Firewall-Multi-AZ-firewall"",
    "Timestamp": "2022-03-22T04:10:04.217Z",
    "SignatureVersion": "1",
    "Signature": "<Signature>",
    "SigningCertURL": "<SigningCertUrl>",
    "UnsubscribeURL": "<UnsubscribeURL>"
}
```

Then, check the Network Firewall alert log in Amazon CloudWatch by following the instructions in the Amazon CloudWatch documentation. The alert log shows the following output:

```
{
    "firewall_name": "AWS-Network-Firewall-Multi-AZ-firewall",
    "availability_zone": "us-east-2b",
    "event_timestamp": "<event timestamp>",
    "event": {
        "timestamp": "2021-03-22T04:10:04.214222+0000",
        "flow_id": "<flow ID>",
        "event_type": "alert",
        "src_ip": "10.1.3.76",
        "src_port": 22761,
        "dest_ip": "99.86.59.73",
        "dest_port": 443,
```
More patterns

- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer (p. 128)
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer (p. 117)
- Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer (p. 138)
- Allow EC2 instances write access to S3 buckets in AMS accounts (p. 1768)
- Analyze speech in real time using Amazon Transcribe and Amazon Comprehend (p. 636)
- Automate adding or updating Windows registry entries using AWS Systems Manager (p. 687)
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK (p. 384)
- Automatically encrypt existing and new Amazon EBS volumes (p. 1779)
- Check AWS CDK applications or CloudFormation templates for best practices by using cdk-nag rule packs (p. 414)
- Check EC2 instances for mandatory tags at launch (p. 579)
- Configure cross-account access to Amazon DynamoDB (p. 416)
- Configure logging and monitoring for security events in your AWS IoT environment (p. 624)
- Configure mutual TLS authentication for applications running on Amazon EKS (p. 428)
- Configure SAML SSO from Microsoft Azure AD to CloudEndure Migration (p. 752)
- Copy data from an S3 bucket in one account and Region to another account and Region (p. 100)
- Deploy Sophos web proxy UTM and Outbound Gateway on AWS (p. 583)
- Encrypt an existing Amazon RDS for PostgreSQL DB instance (p. 313)
- Enforce automatic tagging of Amazon RDS databases at launch (p. 319)
- Enforce tagging of Amazon EMR clusters at launch (p. 2)
- Ensure Amazon EMR logging to Amazon S3 is enabled at launch (p. 29)
- Ensure that Amazon EC2 instances launch only in approved AWS Regions (p. 593)
- Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere (p. 483)
- Help enforce DynamoDB tagging (p. 322)
- Identify and alert when Amazon Kinesis Data Firehose resources are not encrypted with an AWS KMS key (p. 684)
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK (p. 713)
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account (p. 984)
- Migrate an ELK Stack to Elastic Cloud on AWS (p. 43)
- Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud (p. 1007)
- Monitor Amazon Aurora for instances without encryption (p. 353)
- Rotate credentials without restarting containers (p. 246)
- Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let’s Encrypt (p. 530)
- Set up SSO for QuickSight by using AWS SSO and identity federation (p. 72)
- Verify that ELB load balancers require TLS termination (p. 1510)
Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Kinesis Data Firehose with AWS CDK

Created by Shashank Shrivastava (AWS) and Daniel Matuki da Cunha (AWS)

### Code repository:
- Amazon DynamoDB ingestion into Amazon S3

### Environment:
- PoC or pilot

### AWS services:
- AWS CDK;
- Amazon DynamoDB;
- Amazon Kinesis Data Firehose;
- Amazon Kinesis Data Streams;
- AWS Lambda;
- Amazon S3

### Technologies:
- Serverless; Data lakes; Databases; Storage & backup

### Summary

This pattern provides sample code and an application for delivering records from Amazon DynamoDB to Amazon Simple Storage Service (Amazon S3) by using Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose. The pattern’s approach uses AWS Cloud Development Kit (AWS CDK) L3 constructs and includes an example of how to perform data transformation with AWS Lambda before data is delivered to the target S3 bucket on the Amazon Web Services (AWS) Cloud.

Kinesis Data Streams records item-level modifications in DynamoDB tables and replicates them to the required Kinesis data stream. Your applications can access the Kinesis data stream and view the item-level changes in near-real time. Kinesis Data Streams also provides access to other Amazon Kinesis services, such as Kinesis Data Firehose and Amazon Kinesis Data Analytics. This means that you can build applications that provide real-time dashboards, generate alerts, implement dynamic pricing and advertising, and perform sophisticated data analysis.

You can use this pattern for your data integration use cases. For example, transportation vehicles or industrial equipment can send high volumes of data to a DynamoDB table. This data can then be transformed and stored in a data lake hosted in Amazon S3. You can then query and process the data.
and predict any potential defects by using serverless services such as Amazon Athena, Amazon Redshift Spectrum, Amazon Rekognition, and AWS Glue.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Getting started with the AWS CLI in the AWS CLI documentation.
- Node.js and npm, installed and configured. For more information about this, see Downloading and installing Node.js and npm in the npm documentation.
- The GitHub Amazon DynamoDB ingestion into Amazon S3 repository, cloned and configured on your local machine.
- Existing sample data for the DynamoDB table. The data must use the `{ "Id": { "S": "123" }, "MessageData": { "S": "Hello World" } }` format.

Architecture

The diagram shows the following workflow:

1. Data is ingested using Amazon API Gateway as a proxy for DynamoDB. However, you can use any other source to ingest data into DynamoDB.
2. Item-level changes are generated in near-real time in Kinesis Data Streams for delivery to Amazon S3.
3. Kinesis Data Streams sends the records to Kinesis Data Firehose for transformation and delivery.
4. A Lambda function converts the records from a DynamoDB record format to JSON-format, which contains only the record item attribute names and values.

Tools

- AWS CDK – AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- AWS CDK Toolkit – The AWS CDK Toolkit, the CLI command cdk, is the primary tool for interacting with your AWS CDK application. It runs your application, interrogates the application model you defined, and produces and deploys the AWS CloudFormation templates generated by the AWS CDK.
AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

Code

The code for this pattern is available in the GitHub Amazon DynamoDB ingestion into Amazon S3 repository.

Epics

Set up and configure the sample code

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the dependencies.</td>
<td>On your local machine, install the dependencies from the package.json files in the pattern/aws-dynamodb-kinesisstreams-s3 and sample-application directories by running the following commands:</td>
<td>App developer, General AWS</td>
</tr>
</tbody>
</table>
|                                                                      | 1. cd <project_root>/pattern/aws-dynamodb-kinesisstreams-s3  
2. npm install  
3. cd <project_root>/sample-application/  
4. npm install                                                                                   |                                                       |
| Generate the AWS CloudFormation template.                           | 1. Run the cd <project_root>/sample-application/ command.  
2. Run the cdk synth command to generate the AWS CloudFormation template.  
3. The AwsDynamodbKinesisfirehoseS3IngestionStack.template.json output is stored in the cdk.out directory. | App developer, General AWS, AWS DevOps              |
## AWS Prescriptive Guidance Patterns

### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Use AWS CDK or the AWS Management Console to process the template in AWS CloudFormation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Deploy the resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Check and deploy the resources. | 1. Run the `cdk diff` command to identify the resource types that are created by the AWS CDK construct.  
2. Run the `cdk deploy` command to deploy the resources. | App developer, General AWS, AWS DevOps |

## Ingest data into the DynamoDB table to test the solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Ingest your sample data into the DynamoDB table. | 1. Send a request to your DynamoDB table by running the following command in AWS CLI:  
```
aws dynamodb put-item 
--table-name <your_table_name> 
--item 
{"Id": "<primary_key_ID>"} 
"MessageData": 
{"S": "<data>"}
```  
By default, the `put-item` doesn't return any value as output if the operation succeeds. If the operation fails, it returns an error. The data is stored in DynamoDB and then sent to Kinesis Data Streams and Kinesis Data Firehose.  
**Note:** You use different approaches to add data into a DynamoDB table. For more information about this, see Load data into tables in the Amazon DynamoDB documentation. | App developer |
| Verify that a new object is created in the S3 bucket. | Sign in to the AWS Management Console and monitor the S3 bucket to verify that a new | App developer, General AWS |
Create an S3 object

Task | Description | Skills required
--- | --- | ---
object was created with the data that you sent. For more information about this, see `get-object` in the Amazon S3 API reference documentation.

Clean up resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean up resources.</td>
<td>Run the <code>cdk destroy</code> command to delete all the resources used by this pattern.</td>
<td>App developer, General AWS</td>
</tr>
</tbody>
</table>

Related resources

- [s3-static-site-stack.ts](https://github.com) (GitHub repository)
- Amazon API Gateway and Amazon DynamoDB module (GitHub repository)
- Amazon Kinesis Data Streams, Kinesis Data Firehose, and Amazon S3 module (GitHub repository)
- Change data capture for DynamoDB streams (Amazon DynamoDB documentation)
- Using Kinesis Data Streams to capture changes to DynamoDB (Amazon DynamoDB documentation)

Run AWS Systems Manager automation tasks synchronously from AWS Step Functions

Created by Elie El khoury

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment:</th>
<th>Technologies:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <a href="https://github.com">amazon-stepfunctions-ssm-waitfortasktoken</a></td>
<td>Production</td>
<td>Serverless; DevOps; Operations; End-user computing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Workload:</th>
<th>AWS services:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All other workloads</td>
<td>AWS Systems Manager; AWS Step Functions</td>
</tr>
</tbody>
</table>

Summary

This pattern explains how to integrate Amazon Web Services (AWS) Step Functions with AWS Systems Manager. It uses AWS SDK service integrations to call the AWS Systems Manager `startAutomationExecution` API with a task token from a state machine workflow, and pauses until the token returns with a success or failure call. To demonstrate the integration, this pattern implements an automation document (runbook) wrapper around the `AWS-RunShellScript` document and uses...
AWS Step Functions is a low-code, visual workflow service that you can use to build distributed applications, automate IT and business processes, and build data and machine learning pipelines by using AWS services. Workflows manage failures, retries, parallelization, service integrations, and observability so you can focus on higher-value business logic.

Automation, a capability of AWS Systems Manager, simplifies common maintenance, deployment, and remediation tasks for AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Relational Database Service (Amazon RDS), Amazon Redshift, and Amazon Simple Storage Service (Amazon S3). Automation gives you granular control over the concurrency of your automations. For example, you can specify how many resources to target concurrently, and how many errors can occur before an automation is stopped.

For implementation details, including runbook steps, parameters, and examples, see the Additional information (p. 1757) section.

**Prerequisites and limitations**

**Prerequisites**

- An active AWS account
- AWS Identity and Access Management (IAM) permissions to access AWS Step Functions and AWS Systems Manager
- An EC2 instance with AWS Systems Manager Agent (SSM Agent) installed on the instance
- An IAM instance profile for Systems Manager attached to the instance where you plan to run the runbook
- A Step Functions role that has the following IAM permissions (follow the principle of least privilege):

```json
{
  "Effect": "Allow",
  "Action": "ssm:StartAutomationExecution",
  "Resource": "*"
}
```

**Product versions**

- SSM document schema version 0.3 or later
- SSM Agent version 2.3.672.0 or later

**Architecture**

**Target technology stack**

- AWS Step Functions
- AWS Systems Manager Automation

**Target architecture**
Tools

**AWS services**

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

- **AWS Step Functions** – AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through the Step Functions graphical console, you see your application’s workflow as a series of event-driven steps.

- **AWS Systems Manager** – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

**Automation and scale**

- This pattern provides a CloudFormation template that you can use to deploy the runbooks on multiple instances. (See the GitHub [Step Functions and Systems Manager implementation](#) repository.)

---

1753
The code for this pattern is available in the GitHub Step Functions and Systems Manager implementation repository.

Epics

Create runbooks

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download the CloudFormation template.</td>
<td>Download the ssm-automation-documents.cfn.json template from the cloudformation folder of the GitHub repository.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
| Create runbooks.                          | Sign in to the AWS Management Console, open the AWS CloudFormation console, and deploy the template. For more information about deploying AWS CloudFormation templates, see Creating a stack on the AWS CloudFormation console in the CloudFormation documentation. The CloudFormation template deploys three resources:  
  - SfnRunCommandByInstanceIds – Runbook that lets you run AWS-RunShellScript by using instance IDs  
  - SfnRunCommandByTargets – Runbook that lets you run AWS-RunShellScript by using targets  
  - SSMSyncRole – The IAM role assumed by the runbooks | AWS DevOps      |

Create a sample state machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a test state machine.</td>
<td>Follow the instructions in the AWS Step Functions Developer Guide to create and run a state machine. For definition, use the following code. Make sure to update the InstanceIds value with the ID of a valid Systems Manager-enabled instance in your account.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
$": "States.Array($ $.Task.Token)", "workingDirectory": [ "/home/ssm-user/" ], "Commands": [ "echo "This is a test running automation waitForTaskToken" >> automation.log", "sleep 100" ] }, }, "End": true } } |                |

This code calls the runbook to run two commands that demonstrate the `waitForTaskToken` call to Systems Manager Automation. The task writes "This is a test running automation waitForTaskToken" into the `/home/ssm-user/automation.log` file, and then sleeps for 100 seconds before it responds back with the task token and releases the next task in the workflow.

If you want to call the `SfnRunCommandByTargets` runbook
### AWS Prescriptive Guidance Patterns

#### Related resources

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>instead, replace the Parameters section of the previous code with the following:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Parameters&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Targets&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Key&quot;:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;InstanceId&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Values&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;i-02573cafcfEXAMPLE&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;i-0471e04240EXAMPLE&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Update the IAM role for the state machine.</strong></td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>The previous step automatically creates a dedicated IAM role for the state machine. However, it doesn’t grant permissions to call the runbook. Update the role by adding the following permissions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Effect&quot;: &quot;Allow&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;ssm:StartAutomationExecution&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Resource&quot;: &quot;*&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Validate the synchronous calls.</strong></td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>Run the state machine to validate the synchronous call between Step Functions and Systems Manager Automation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For sample output, see the Additional information (p. 1757) section.</td>
<td></td>
</tr>
</tbody>
</table>

#### Related resources

- **Getting started with AWS Step Functions** ([AWS Step Functions Developer Guide](https://docs.aws.amazon.com/stepfunctions/latest/userguide/what-is-stepfunctions.html))
- **Wait for a callback with the task token** ([AWS Step functions Developer Guide, service integration patterns](https://docs.aws.amazon.com/stepfunctions/latest/userguide/what-is-stepfunctions.html))
- `send_task_success` and `send_task_failure` API calls ([Boto3 documentation](https://boto3.amazonaws.com/v1/documentation/api/latest/index.html))
- **AWS Systems Manager Automation** ([AWS Systems Manager User Guide](https://docs.aws.amazon.com/systemsmanager/latest/userguide/))
Additional information

Implementation details

This pattern provides an AWS CloudFormation template that deploys two Systems Manager runbooks:

- `SfnRunCommandByInstanceIds` runs the AWS-RunShellScript command by using instance IDs.
- `SfnRunCommandByTargets` runs the AWS-RunShellScript command by using targets.

Each runbook implements three steps to achieve a synchronous call when using the `.waitTaskToken` option in Step Functions.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RunCommand</td>
<td>Runs the RunShellScript command.</td>
</tr>
<tr>
<td>2</td>
<td>SendTaskFailure</td>
<td>Runs when step 1 is aborted or canceled. It calls the Step Functions <code>send_task_failure</code> API, which accepts three parameters as input: the token passed by the state machine, the failure error, and a description of the cause of the failure.</td>
</tr>
<tr>
<td>3</td>
<td>SendTaskSuccess</td>
<td>Runs when step 1 is successful. It calls the Step Functions <code>send_task_success</code> API, which accepts the token passed by the state machine as input.</td>
</tr>
</tbody>
</table>

Runbook parameters

`SfnRunCommandByInstanceIds` runbook

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Type</th>
<th>Optional or required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>executionTimeout</td>
<td>Integer</td>
<td>Optional</td>
<td>The time, in seconds, for a command to complete before it is considered to have failed. The default is 3600 (1 hour). The maximum value is 172800 (48 hours).</td>
</tr>
<tr>
<td>workingDirectory</td>
<td>String</td>
<td>Optional</td>
<td>The path to the working directory on your instance.</td>
</tr>
<tr>
<td>Commands</td>
<td>StringList</td>
<td>Required</td>
<td>The shell script or command to run.</td>
</tr>
</tbody>
</table>
### InstanceIds

<table>
<thead>
<tr>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringList</td>
<td></td>
<td>The IDs of the instances where you want to run the command.</td>
</tr>
</tbody>
</table>

### taskToken

<table>
<thead>
<tr>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
<td>The task token to use for callback responses.</td>
</tr>
</tbody>
</table>

### SfnRunCommandByTargets runbook

#### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Optional or required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>executionTimeout</td>
<td>Integer</td>
<td>Optional</td>
<td>The time, in seconds, for a command to complete before it is considered to have failed. The default is 3600 (1 hour). The maximum value is 172800 (48 hours).</td>
</tr>
<tr>
<td>workingDirectory</td>
<td>String</td>
<td>Optional</td>
<td>The path to the working directory on your instance.</td>
</tr>
<tr>
<td>Commands</td>
<td>StringList</td>
<td>Required</td>
<td>The shell script or command to run.</td>
</tr>
</tbody>
</table>
| Targets      | MapList    | Required             | An array of search criteria that identifies instances by using key-value pairs that you specify. For example: 
```json
  
  
  [{"Key": "InstanceIds","Values": ["i-02573cafcfEXAMPLE","i-0471e04240EXAMPLE"],}

  taskToken | String     | Required             | The task token to use for callback responses.                               |

### Sample output

The following table provides sample output from the step function. It shows that the total run time is over 100 seconds between step 5 (TaskSubmitted) and step 6 (TaskSucceeded). This demonstrates that the step function waited for the "sleep 100" command to finish before moving to the next task in the workflow.

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Step</th>
<th>Resource</th>
<th>Elapsed Time (ms)</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ExecutionStarted</td>
<td>0</td>
<td>Mar 11, 1758</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### More patterns

- Automate deployment of nested applications using AWS SAM (p. 1416)
- Automatically archive items to Amazon S3 using DynamoDB TTL (p. 1334)
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9 (p. 403)
- Build a multi-tenant serverless architecture in Amazon OpenSearch Service (p. 1359)
- Cache secrets using AWS Lambda extensions (p. 1391)
• Copy AWS Service Catalog products across different AWS accounts and AWS Regions (p. 699)
• Decompose monoliths into microservices by using CQRS and event sourcing (p. 1491)
• Deploy a React-based single-page application to Amazon S3 and CloudFront (p. 1844)
• Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code (p. 281)
• Deploy Lambda functions with container images (p. 191)
• Ensure Amazon EMR logging to Amazon S3 is enabled at launch (p. 29)
• Implement the serverless saga pattern by using AWS Step Functions (p. 1448)
• Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK (p. 713)
• Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function (p. 500)
• Orchestrates an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions (p. 58)
• Run event-driven and scheduled workloads at scale with AWS Fargate (p. 1475)
• Set up CI/CD for AWS AppSync GraphQL API updates (p. 1482)
• Turn off security standard controls across all Security Hub member accounts in a multi-account environment (p. 1730)
Run unit tests for a Node.js application from GitHub by using AWS CodeBuild

Created by Thomas Scott (AWS) and Jean-Baptiste Guillois (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment:</th>
<th>Technologies:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node JS Tests Sample</td>
<td>Production</td>
<td>Software development &amp; testing</td>
</tr>
</tbody>
</table>

AWS services: AWS CodeBuild

Summary

This pattern provides sample source code and key unit test components for a Node.js game API. It also includes instructions for running these unit tests from a GitHub repository by using AWS CodeBuild, as part of your continuous integration and continuous delivery (CI/CD) workflow.

Unit testing is a software development process in which different parts of an application, called units, are individually and independently tested for correct operation. Tests validate the quality of the code and confirm that it functions as expected. Other developers can also easily gain familiarity with your code base by consulting the tests. Unit tests reduce future refactoring time, help engineers get up to speed on your code base more quickly, and provide confidence in the expected behavior.

Unit testing involves testing individual functions, including AWS Lambda functions. To create unit tests, you need a testing framework and a way of validating tests (assertions). The code examples in this pattern use the Mocha testing framework and the Chai assertion library.

For more information about unit testing and examples of test components, see the Additional information (p. 1765) section.

Prerequisites and limitations

- An active AWS account with correct CodeBuild permissions
- A GitHub account (see instructions for signing up)
- Git (see installation instructions)
- A code editor to make changes and push your code to GitHub (for example, you can use AWS Cloud9)

Architecture

This pattern implements the architecture that is shown in the following diagram.
Tools

Tools

- **Git** — Git is a version control system that you can use for code development.
- **AWS Cloud9** — AWS Cloud9 is an integrated development environment (IDE) that offers a rich code-editing experience with support for several programming languages and runtime debuggers, and a built-in terminal. It contains a collection of tools that you use to code, build, run, test, and debug software, and helps you release software to the cloud. You access the AWS Cloud9 IDE through a web browser.
- **AWS CodeBuild** — AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

Code

The source code for this pattern is available on GitHub, in the Sample game unit test application repository. You can create your own GitHub repository from this sample (option 1) or use the sample repository directly (option 2) for this pattern. Follow the instructions for each option in the next section. The option you follow will depend on your use case.

Epics

Option 1 - Run unit tests on your personal GitHub repository with CodeBuild

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create your own GitHub repository based on the sample project.</td>
<td>1. Log in to GitHub. 2. Create a new repository. For instructions, see the GitHub documentation.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>3.</td>
<td>Clone and push the sample repository into the new repository in your account.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
<tr>
<td>Create a new CodeBuild project.</td>
<td>1. Sign in to the AWS Management Console and open the CodeBuild console at <a href="https://console.aws.amazon.com/codesuite/codebuild/home">https://console.aws.amazon.com/codesuite/codebuild/home</a>. 2. Choose Create build project. 3. In the Project configuration section, for Project name, type aws-tests-sample-node-js. 4. In the Source section, for Source provider, choose GitHub. 5. For Repository, choose Repository in my GitHub account, and then paste the URL to your newly created GitHub repository. 6. In the Primary source webhook events section, select Rebuild every time a code change is pushed to this repository. 7. For event type, choose PUSH. 8. In the Environment section, choose Managed image, Amazon Linux 2, and the latest image. 9. Leave the default settings for all other options, and then choose Create build project.</td>
<td></td>
</tr>
<tr>
<td>Start the build.</td>
<td>On the Review page, choose Start build to run the build.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
</tbody>
</table>

**Option 2 - Run unit tests on a public repository with CodeBuild**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new CodeBuild build project.</td>
<td>1. Sign in to the AWS Management Console and open the CodeBuild console at <a href="https://console.aws.amazon.com/codesuite/codebuild/home">https://console.aws.amazon.com/codesuite/codebuild/home</a>. 2. Choose Create build project.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
</tbody>
</table>
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. In the <strong>Project configuration</strong> section, for <strong>Project name</strong>, type <code>aws-tests-sample-node-js</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. In the <strong>Source</strong> section, for <strong>Source provider</strong>, choose GitHub.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. For <strong>Repository</strong>, choose <strong>Public repository</strong>, and then paste the URL: <a href="https://github.com/aws-samples/node-js-tests-sample">https://github.com/aws-samples/node-js-tests-sample</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. In the <strong>Environment</strong> section, choose <strong>Managed image</strong>, Amazon Linux 2, and the latest image.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Leave the default settings for all other options, and then choose <strong>Create build project</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Start the build.</strong></td>
<td><strong>On the Review page, choose Start build to run the build.</strong></td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
</tbody>
</table>

### Analyze the unit tests

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>View test results.</td>
<td>In the CodeBuild console, review the unit test results from the CodeBuild job. They should match the results shown in the Additional information (p. 1765) section. These results validate the GitHub repository integration with CodeBuild.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
<tr>
<td>Apply a webhook.</td>
<td>You can now apply a webhook, so you can automatically start a build whenever you push code changes to the main branch of your repository. For instructions, see the CodeBuild documentation.</td>
<td>App developer, AWS administrator, AWS DevOps</td>
</tr>
</tbody>
</table>

### Related resources

- Sample game unit test application (GitHub repository with sample code)
- AWS CodeBuild documentation
Additional information

Unit test results

In the CodeBuild console, you should see the following test results after the project builds successfully.

```
> node-game-unit-tests@1.0.0 test /codebuild/output/src8139917245/src/github.com/aws-samples/node-js-tests-sample
> mocha

Mocha 1.7.0
Game Function Group
   Initialize Game
     ✓ Check that the game ID is between 0 and 1000
     ✓ Check that the play date is today
   Start Game
     ✓ Check that the game has two competitors
     ✓ Check that the two competitors have different names
   End Game
     ✓ Check that there are no more competitors after the game ends
   Game ID Stubs and Mocks
     ✓ Check that an out of range game ID is returned with Stub
     ✓ Using a mock check that the function was called and returns 9

7 passing (9ms)
```

Example unit test components

This section describes the four types of test components that are used in unit testing: assertions, spies, stubs, and mocks. It includes a brief explanation and code example of each component.

**Assertions**

An assertion is used to verify an expected result. This is an important test component because it validates the expected response from a given function. The following sample assertion validates that the returned ID is between 0 and 1000 when initializing a new game.

```javascript
const { expect } = require('chai');
const { Game } = require('../src/index');

describe('Game Function Group', () => {
    it('Check that the Game ID is between 0 and 1000', () => { 
        const game = new Game();
        expect(game.id).is.above(0).but.below(1000);
    });
});
```

**Spies**

A spy is used to observe what is happening when a function is running. For example, you might want to validate that the function has been called correctly. The following example shows that start and stop methods are called on a `Game` class object.

```javascript
const { expect } = require('chai');
const { spy } = require('sinon');
const { Game } = require('../src/index');

describe('Game Function Group', () => {
    it('should verify that the correct function is called', () => {
        const spyStart = spy(Game.prototype, 'start');
        const spyStop = spy(Game.prototype, 'stop');
        const game = new Game();
        game.start();
        game.stop();
        expect(spyStart.callCount).to.equal(1);
        expect(spyStop.callCount).to.equal(1);
    });
});
```
const spyStart = spy(Game.prototype, "start");
const spyStop = spy(Game.prototype, "stop");

const game = new Game();
game.start();
game.stop();

expect(spyStart.called).to.be.true
expect(spyStop.called).to.be.true
});
});

**Stubs**

A stub is used to override a function’s default response. This is especially useful when the function makes an external request, because you want to avoid making external requests from unit tests. (External requests are better suited for integration tests, which can physically test requests between different components.) In the following example, a stub forces a return ID from the `getId` function.

```javascript
const { expect } = require('chai');
const {stub} = require('sinon');
const {Game} = require('../src/index');

describe('Game Function Group', () => {
  it('Check that the Game ID is between 0 and 1000', function() {
    let generateIdStub = stub(Game.prototype, 'getId').returns(999999);
    const game = new Game();
    expect(game.getId).is.equal(999999);
    generateIdStub.restore();
  });
});

**Mocks**

A mock is a fake method that has a pre-programmed behavior for testing different scenarios. A mock can be considered an extended form of a stub and can carry out multiple tasks simultaneously. In the following example, a mock is used to validate three scenarios:

- Function is called
- Function is called with arguments
- Function returns the integer 9

```javascript
const { expect } = require('chai');
const {mock} = require('sinon');
const {Game} = require('../src/index');

describe('Game Function Group', () => {
  it('Check that the Game ID is between 0 and 1000', function() {
    let mock = mock(Game.prototype).expects('getId').withArgs().returns(9);
    const game = new Game();
    const id = get.getId();
    mock.verify();
    expect(id).is.equal(9);
  });
});
```
More patterns

- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK (p. 384)
- Build a video processing pipeline by using Amazon Kinesis Video Streams and AWS Fargate (p. 93)
- Cache secrets using AWS Lambda extensions (p. 1391)
- Deploy a clustered application to Amazon ECS by using AWS Copilot (p. 196)
- Deploy Lambda functions with container images (p. 191)
- Generate a static outbound IP address using a Lambda function, Amazon VPC, and a serverless architecture (p. 211)
- Modernize ASP.NET Web Forms applications on AWS (p. 1466)
- Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance (p. 258)
- Set up CI/CD for AWS AppSync GraphQL API updates (p. 1482)
Storage & backup

Topics

- Allow EC2 instances write access to S3 buckets in AMS accounts (p. 1768)
- Automate data stream ingestion into a Snowflake database by using Snowflake Snowpipe, Amazon S3, Amazon SNS, and Amazon Kinesis Data Firehose (p. 1771)
- Automatically encrypt existing and new Amazon EBS volumes (p. 1779)
- Back up Sun SPARC servers in the Stromaysys Charon-SSP emulator on the AWS Cloud (p. 1785)
- Back up and archive data to Amazon S3 with Veeam Backup & Replication (p. 1802)
- Migrate data from an on-premises Hadoop environment to Amazon S3 using DistCp with AWS PrivateLink for Amazon S3 (p. 1816)
- Perform cross-Region failover and failback by using CloudEndure Disaster Recovery (p. 1826)
- Use CloudEndure for disaster recovery of an on-premises database (p. 1834)
- More patterns (p. 1843)

Allow EC2 instances write access to S3 buckets in AMS accounts

Created by Mansi Suratwala (AWS)

<table>
<thead>
<tr>
<th>Environment:</th>
<th>Production</th>
<th>Technologies:</th>
<th>Storage &amp; backup; Databases; Security, identity, compliance; Operations</th>
<th>Workload:</th>
<th>All other workloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS services:</td>
<td>Amazon S3; AWS Managed Services</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary

AWS Managed Services (AMS) helps you to operate your Amazon Web Services (AWS) infrastructure more efficiently and securely. AMS accounts have security guardrails for standardized administration of your AWS resources. One guardrail is that default Amazon Elastic Compute Cloud (Amazon EC2) instance profiles don't allow write access to Amazon Simple Storage Service (Amazon S3) buckets. However, your organization might have multiple S3 buckets and require more control over access by EC2 instances. For example, you might want to store database backups from EC2 instances in an S3 bucket.

This pattern explains how to use Requests for Change (RFCs) to allow your EC2 instances write access to S3 buckets in your AMS account. An RFC is a request created by you or AMS to make a change in your managed environment and that includes a change type (CT) ID for a particular operation.

Prerequisites and limitations

Prerequisites

- An AMS Advanced account. For more information about this, see AMS operations plans in the AWS Managed Services documentation.
• Access to the customer-mc-user-role AWS Identity and Access Management (IAM) role to submit RFCs.
• AWS Command Line Interface (AWS CLI), installed and configured with the EC2 instances in your AMS account.
• An understanding of how to create and submit RFCs in AMS. For more information about this, see Change Management, RFCs, change types, and examples in the AWS Managed Services documentation.
• An understanding of manual and automated change types (CTs). For more information about this, see Automated and manual CTs in the AWS Managed Services documentation.

Architecture

Technology stack

• AMS
• AWS CLI
• Amazon EC2
• Amazon S3
• IAM

Tools

• AWS CLI – AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent interface for interacting with all parts of AWS.
• AWS Identity and Access Management (IAM) – IAM is a web service that helps you to securely control access to AWS resources.
• AWS Managed Services (AMS) – AMS helps you to operate your AWS infrastructure more efficiently and securely.
• Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
• Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.

Epics

Create an S3 bucket with an RFC

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an S3 bucket using an automated RFC.</td>
<td>1. Sign in to your AMS account, choose the Choose change type page, choose RFCs, and then choose Create RFC. 2. Submit the Create S3 Bucket automated RFC.</td>
<td>AWS systems administrator, AWS developer</td>
</tr>
</tbody>
</table>

Note: Make sure that you record the S3 bucket’s name.
Create an IAM instance profile and associate it with the EC2 instances

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit a manual RFC to create an IAM role.</td>
<td>When an AMS account is onboarded, a default customer-mc-ec2-instance-profile IAM instance profile is created and associated to each EC2 instance in your AMS account. However, the instance profile doesn't have Write permissions to your S3 buckets. To add the Write permissions, submit the Create IAM Resource manual RFC to create an IAM role that has the following three policies: customer_ec2_instance_, customer_deny_policy, and customer_ec2_s3_integration_policy. <strong>Important:</strong> The customer_ec2_instance_ and customer_deny_policy policies already exist in your AMS account. However, you need to create the customer_ec2_s3_integration_policy policy by using the following sample policy:</td>
<td>AWS systems administrator, AWS developer</td>
</tr>
</tbody>
</table>

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

Role Permissions:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            }
        }
    ]
}
```
### Task Description

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit a manual RFC to replace the IAM instance profile.</td>
<td>Submit a manual RFC to associate the target EC2 instances with the new IAM instance profile.</td>
<td>AWS systems administrator, AWS developer</td>
</tr>
<tr>
<td>Test a copy operation to the S3 bucket.</td>
<td>Test a copy operation to the S3 bucket by running the following command in AWS CLI: <code>aws s3 cp test.txt s3://&lt;S3 Bucket&gt;/test2.txt</code></td>
<td>AWS systems administrator, AWS developer</td>
</tr>
</tbody>
</table>

### Related resources

- Create an IAM instance profile for your Amazon EC2 instances
- Creating an S3 bucket (using the Amazon S3 console, AWS SDKs, or AWS CLI)

Automate data stream ingestion into a Snowflake database by using Snowflake Snowpipe, Amazon S3, Amazon SNS, and Amazon Kinesis Data Firehose

*Created by Bikash Chandra Rout (AWS)*
Summary

This pattern describes how you can use services on the Amazon Web Services (AWS) Cloud to process a continuous stream of data and load it into a Snowflake database. The pattern uses Amazon Kinesis Data Firehose to deliver the data to Amazon Simple Storage Service (Amazon S3), Amazon Simple Notification Service (Amazon SNS) to send notifications when new data is received, and Snowflake Snowpipe to load the data into a Snowflake database.

By following this pattern, you can have continuously generated data available for analysis in seconds, avoid multiple manual COPY commands, and have full support for semi-structured data on load.

Prerequisites and limitations

Prerequisites

• An active AWS account.
• A data source that is continuously sending data to a Kinesis Data Firehose delivery stream.
• An existing S3 bucket that is receiving the data from the Kinesis Data Firehose delivery stream.
• An active Snowflake account.

Limitations

• Snowflake Snowpipe doesn't connect directly to Kinesis Data Firehose.

Architecture

Technology stack

• Amazon Kinesis Data Firehose
• Amazon SNS
• Amazon S3
• Snowflake Snowpipe
• Snowflake database

**Tools**

• **Kinesis Data Firehose** – Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers.

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

• **Amazon SNS** – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.

• **Snowflake** – Snowflake is an analytic data warehouse provided as Software-as-a-Service (SaaS).

• **Snowflake Snowpipe** – Snowpipe loads data from files as soon as they’re available in a Snowflake stage.

**Epics**

**Set up a Snowflake Snowpipe**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a CSV file in Snowflake.</td>
<td>Sign in to Snowflake and run the “CREATE FILE FORMAT” command to create a CSV file with a specified field delimiter. For more information about this and other Snowflake commands, see the “Additional information” section.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create an external Snowflake stage.</td>
<td>Run the “CREATE STAGE” command to create an external Snowflake stage that references the CSV file you created earlier. Important: You will need the URL for the S3 bucket, your AWS access key, and your AWS secret access key. Run the “SHOW STAGES” command to verify that the Snowflake stage is created.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create the Snowflake target table.</td>
<td>Run the “CREATE TABLE” command to create the Snowflake table.</td>
<td>Developer</td>
</tr>
<tr>
<td>Create a pipe.</td>
<td>Run the “CREATE PIPE” command; make sure that “auto_ingest=true” in the command. Run the “SHOW PIPES” command to verify that the pipe is created. Copy and save the “notification_channel” column value. This value will be</td>
<td>Developer</td>
</tr>
</tbody>
</table>
## Configure the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the S3 bucket</td>
<td>used to configure Amazon S3 event notifications.</td>
<td></td>
</tr>
<tr>
<td>Configure the S3 bucket</td>
<td>Sign in to the AWS Management Console and open the Amazon S3 console. Choose the S3 bucket that contains the data from Kinesis Data Firehose. Then choose the “Management” tab in the S3 bucket and choose “Add lifecycle rule.” Enter a name for your rule in the “Lifecycle rule” dialog box and configure a 30-day lifecycle rule for your bucket. For help with this and other stories, see the “Related resources” section.</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Create an IAM policy for the S3 bucket</td>
<td>Open the AWS Identity and Access Management (IAM) console and choose “Policies.” Choose “Create policy,” and choose the “JSON” tab. Copy and paste the policy from the “Additional information” section into the JSON field. This policy will grant “PutObject” and “DeleteObject” permissions, as well as “GetObject,” “GetObjectVersion,” and “ListBucket” permissions. Choose “Review policy,” enter a policy name, and then choose “Create policy.”</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Assign the policy to an IAM role</td>
<td>Open the IAM console choose “Roles,” and then choose “Create role.” Choose “Another AWS account” as the trusted entity. Enter your AWS account ID, and choose “Require external ID.” Enter a placeholder ID that you will change it later. Choose “Next” and assign the IAM policy you created earlier. Then create the IAM role.</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Copy the Amazon Resource Name (ARN) for the IAM role</td>
<td>Open the IAM console and choose “Roles.” Choose the IAM role you created earlier, and</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>then copy and store the “Role ARN.”</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Set up a storage integration in Snowflake**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a storage integration in Snowflake.</td>
<td>Sign in to Snowflake and run the “CREATE STORAGE INTEGRATION” command. This will modify the trusted relationship, grant access to Snowflake, and provide the external ID for your Snowflake stage.</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Retrieve the IAM role for your Snowflake account.</td>
<td>Run the “DESC INTEGRATION” command to retrieve the ARN for the IAM role. Important: <code>&lt;integration_name&gt;</code> is the name of the Snowflake storage integration you created earlier.</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Record two column values.</td>
<td>Copy and save the values for the “storage_aws_iam_user_arn” and “storage_aws_external_id” columns.</td>
<td>System Administrator, Developer</td>
</tr>
</tbody>
</table>

**Allow Snowflake Snowpipe to access the S3 bucket**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify the IAM role policy.</td>
<td>Open the IAM console and choose &quot;Roles.&quot; Choose the IAM role you created earlier and choose the &quot;Trust relationships&quot; tab. Choose &quot;Edit trust relationship.&quot; Replace &quot;snowflake_external_id&quot; with the &quot;storage_aws_external_id&quot; value you copied earlier. Replace &quot;snowflake_user_arn&quot; with the storage_aws_iam_user_arn&quot; value you copied earlier. Then choose &quot;Update trust policy.&quot;</td>
<td>System Administrator, Developer</td>
</tr>
</tbody>
</table>
### Turn on and configure SNS notifications for the S3 bucket

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn on event notifications for the S3 bucket.</td>
<td>Open the Amazon S3 console and choose your bucket. Choose “Properties,” and under “Advanced settings” choose “Events.” Choose “Add notification,” and enter a name for this event. If you don’t enter a name, a globally unique identifier (GUID) will be used.</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Configure Amazon SNS notifications for the S3 bucket.</td>
<td>Under “Events,” choose “ObjectCreate (All)” and then choose “SQS Queue” in the “Send to” dropdown list. In the “SNS” list choose “Add SQS queue ARN,” and paste the “notification_channel” value you copied earlier. Then choose “Save.”</td>
<td>System Administrator, Developer</td>
</tr>
<tr>
<td>Subscribe the Snowflake SQS queue to the SNS topic.</td>
<td>Subscribe the Snowflake SQS queue to the SNS topic you created. For help with this step, see the “Related resources” section.</td>
<td>System Administrator, Developer</td>
</tr>
</tbody>
</table>

### Check the Snowflake stage integration

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check and test Snowpipe.</td>
<td>Sign in to Snowflake and open the Snowflake stage. Drop files into your S3 bucket and check if the Snowflake table loads them. Amazon S3 will send SNS notifications to Snowpipe when new objects appear in the S3 bucket.</td>
<td>System Administrator, Developer</td>
</tr>
</tbody>
</table>

### Related resources

- Create lifecycle policy for an S3 bucket
- Subscribe the Snowflake SQS Queue to the Amazon SNS Topic

### Additional information

Create a file format:
CREATE FILE FORMAT <name>
TYPE = 'CSV'
FIELD_DELIMITER = '|
SKIP_HEADER = 1;

Create an external stage:

externalStageParams (for Amazon S3) ::= 
URL = 's3://[//]
[ { STORAGE_INTEGRATION = } | { CREDENTIALS = ( { { AWS_KEY_ID = ` AWS_SECRET_KEY = ` 
[ AWS_TOKEN = `` ] } | AWS_ROLE = `` } ) } ] ]
[ ENCRYPTION = ( [ TYPE = 'AWS_CSE' ] [ MASTER_KEY = '' ] | 
[ TYPE = 'AWS_SSE_S3' ] | 
[ TYPE = 'AWS_SSE_KMS' [ KMS_KEY_ID = '' ] | 
[ TYPE = NONE ] ) ]

Create a table:

CREATE [ OR REPLACE ] [ ( [ LOCAL | GLOBAL ] TEMP[ORARY] | VOLATILE } | TRANSIENT ] TABLE 
[ IF NOT EXISTS ]
<table_name>
( <col_name> <col_type> [ { DEFAULT <expr>
| { AUTOINCREMENT | IDENTITY } [ ( <start_num>,
<step_num> ) | START <num> INCREMENT <num> ] ]
/* AUTOINCREMENT / IDENTITY supported only for numeric data
types (NUMBER, INT, etc.) */
| inlineConstraint ]
[ , <col_name> <col_type> ... ]
[ , outoflineConstraint ]
[ , ... ]
[ CLUSTER BY ( <expr> [ , <expr>, ... ] ) ]
[ STAGE_FILE_FORMAT = ( { FORMAT_NAME = '<file_format_name>'
| TYPE = { CSV | JSON | AVRO | ORC | PARQUET | XML } ]
[ formatTypeOptions ] ) ]
[ STAGE_COPY_OPTIONS = ( copyOptions ) ]
[ DATA_RETENTION_TIME_IN_DAYS = <num> ]
[ COPY GRANTS ]
[ COMMENT = '<string_literal>' ]

Show stages:

SHOW STAGES;

Create a pipe:

CREATE [ OR REPLACE ] PIPE [ IF NOT EXISTS ]
[ AUTO_INGEST = [ TRUE | FALSE ] ]
[ AWS_SNS_TOPIC = ]
[ INTEGRATION = '' ]
[ Comment = '' ]
AS

Show pipes:

SHOW PIPES [ LIKE '<pattern>' ]
[ IN { ACCOUNT | [ DATABASE ] <db_name> | [ SCHEMA ] <schema_name> } ]
Create a storage integration:

```sql
CREATE STORAGE INTEGRATION <integration_name>
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN = '<iam_role>'
STORAGE_ALLOWED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/')
[ STORAGE_BLOCKED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/') ]
```

Example:

```sql
create storage integration s3_int
type = external_stage
storage_provider = s3
enabled = true
storage_aws_role_arn = 'arn:aws:iam::001234567890:role/myrole'
storage_allowed_locations = ('s3://mybucket1/mypath1/','s3://mybucket2/mypath2/')
storage_blocked_locations = ('s3://mybucket1/mypath1/sensitivedata/','s3://mybucket2/mypath2/sensitivedata/');
```

For more information about this step, see Configuring a Snowflake storage integration to access Amazon S3 from the Snowflake documentation.

Describe an integration:

```sql
DESC INTEGRATION <integration_name>;
```

S3 bucket policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource": "arn:aws:s3:::*"
      },
      {
         "Effect": "Allow",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::",
         "Condition": {
            "StringLike": {
               "s3:prefix": [
                  "/*
               ]
            }
         }
      }
   ]
}
```
Automatically encrypt existing and new Amazon EBS volumes

Summary

Encryption of Amazon Elastic Block Store (Amazon EBS) volumes is important to an organization's data protection strategy. It is an important step in establishing a well-architected environment. Although there is no direct way to encrypt existing unencrypted EBS volumes or snapshots, you can encrypt them by creating a new volume or snapshot. For more information, see Encrypt EBS resources in the Amazon EC2 documentation. This pattern provides preventative and detective controls for encrypting your EBS volumes, both new and existing. In this pattern, you configure account settings, create automated remediation processes, and implement access controls.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- AWS Command Line Interface (AWS CLI), installed and configured on macOS, Linux, or Windows
- jq, installed and configured on macOS, Linux, or Windows
- AWS Identity and Access Management (IAM) permissions are provisioned to have read and write access to AWS CloudFormation, Amazon Elastic Compute Cloud (Amazon EC2), AWS Systems Manager, AWS Config, and AWS Key Management Service (AWS KMS)
- AWS Organizations is configured with all features enabled, a requirement for service control policies
- AWS Config is enabled in the target accounts

Limitations

- In your target AWS account, there must be no AWS Config rules named encrypted-volumes. This solution deploys a rule with this name. Preexisting rules with this name can cause the deployment to fail and result in unnecessary charges related to processing the same rule more than once.
- This solution encrypts all EBS volumes with the same AWS KMS key.
- If you enable encryption of EBS volumes for the account, this setting is Region-specific. If you enable it for an AWS Region, you cannot disable it for individual volumes or snapshots in that Region. For more information, see Encryption by default in the Amazon EC2 documentation.
• When you remediate existing, unencrypted EBS volumes, ensure that the EC2 instance is not in use. This automation shuts down the instance in order to detach the unencrypted volume and attach the encrypted one. There is downtime while the remediation is in progress. If this is a critical piece of infrastructure for your organization, make sure that manual or automatic high-availability configurations are in place so as to not impact the availability of any applications running on the instance. We recommend that you remediate critical resources only during standard maintenance windows.

Architecture

Automation workflow

1. AWS Config detects an unencrypted EBS volume.
2. An administrator uses AWS Config to send a remediation command to Systems Manager.
3. The Systems Manager automation takes a snapshot of the unencrypted EBS volume.
4. The Systems Manager automation uses AWS KMS to create an encrypted copy of the snapshot.
5. The Systems Manager automation does the following:
   a. Stops the affected EC2 instance if it is running
   b. Attaches the new, encrypted copy of the volume to the EC2 instance
   c. Returns the EC2 instance to its original state

Tools

AWS services

• AWS CLI – The AWS Command Line Interface (AWS CLI) provides direct access to the public application programming interfaces (APIs) of AWS services. You can explore a service's capabilities with the AWS CLI and develop shell scripts to manage your resources. In addition to the low-level API-equivalent commands, several AWS services provide customizations for the AWS CLI. Customizations can include higher-level commands that simplify using a service with a complex API.
• AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your AWS resources. You create a template that describes all the AWS resources that you want (such as Amazon EC2 instances), and CloudFormation provisions and configures those resources for you.
• AWS Config – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.

• Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.

• AWS KMS – AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud. AWS KMS keys and functionality are used by other AWS services, and you can use them to protect data in your AWS environment.

• AWS Organizations – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

• AWS Systems Manager Automation – Systems Manager Automation simplifies common maintenance and deployment tasks for Amazon EC2 instances and other AWS resources.

Other services

• jq – jq is a lightweight and flexible command-line JSON processor. You use this tool to extract key information from the AWS CLI output.

Code

• The code for this pattern is available in the GitHub Automatically remediate unencrypted EBS Volumes using customer KMS keys repository.

Epics

Automate remediation of unencrypted volumes

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download scripts and CloudFormation templates.</td>
<td>Download the shell script, JSON file, and CloudFormation templates from the GitHub Automatically remediate unencrypted EBS Volumes using customer KMS keys repository.</td>
<td>AWS administrator, General AWS</td>
</tr>
<tr>
<td>Identify the administrator for the AWS KMS key.</td>
<td>1. Sign in to the AWS Management Console and open the IAM console at <a href="https://console.aws.amazon.com/iam/">https://console.aws.amazon.com/iam/</a>. 2. Identify an IAM user or role who will be the AWS KMS key administrator. If a new user or role needs to be created for this purpose, create it now. For more information, see IAM Identities in the IAM documentation. This automation creates a new AWS KMS key.</td>
<td>AWS administrator, General AWS</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| Deploy the Stack1 CloudFormation template. | 1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.  
2. In CloudFormation, deploy the `Stack1.yaml` template. Note the following deployment details:  
   - Give the stack a clear and descriptive name. Note the stack name because you need this value in the next step.  
   - Paste the ARN of the key administrator into the only parameter field in Stack1. This IAM user or role becomes the administrator for the AWS KMS key created by the stack.  
For more information about deploying a CloudFormation template, see Working with AWS CloudFormation templates in the CloudFormation documentation. | AWS administrator, General AWS |
| Deploy the Stack2 CloudFormation template. | In CloudFormation, deploy the `Stack2.yaml` template. Note the following deployment details:  
   - Give the stack a clear and descriptive name.  
   - For the only parameter of Stack2, enter the name of the stack you created in the previous step. This allows Stack2 to reference the new AWS KMS key and role deployed by the stack in the previous step. | AWS administrator, General AWS |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an unencrypted volume for testing.</td>
<td>Create an EC2 instance with an unencrypted EBS volume. For instructions, see Create an Amazon EBS volume in the Amazon EC2 documentation. The instance type does not matter, and access to the instance is not needed. You can create a t2.micro instance to stay in the free tier, and you don’t need to create a key pair.</td>
<td>AWS administrator, General AWS</td>
</tr>
</tbody>
</table>
2. Confirm that your new, unencrypted test instance appears in the list of non-compliant resources. If the volume does not appear immediately, wait a few minutes and refresh the results. The AWS Config rule detects the resource changes shortly after the instance and volume are created.  
3. Select the resource, and then choose Remediate.  
You can view the remediation progress and status in Systems Manager as follows:  
2. In the navigation pane, choose Automation.  
3. Choose the Execution ID link to view the steps and the status. | AWS administrator, General AWS |
| Configure additional accounts or AWS Regions. | As needed for your use case, repeat this epic for any additional accounts or AWS Regions. | AWS administrator, General AWS |
Enable account-level encryption of EBS volumes

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the enable script.</td>
<td>In AWS CLI, run the enable-ebs-encryption-for-account.sh script. You downloaded this script from the GitHub repository in the previous epic.</td>
<td>AWS administrator, General AWS</td>
</tr>
<tr>
<td>Confirm the settings are updated.</td>
<td>1. Open the Amazon EC2 console at <a href="https://console.aws.amazon.com/ec2/">https://console.aws.amazon.com/ec2/</a>. 2. On the right side of the screen, choose EBS Encryption. 3. Confirm that Always encrypt new EBS volumes is turned on and the Default encryption key is set to the ARN that you specified in the previous epic. If the Always encrypt new EBS volumes setting is turned off or the key is still set to alias/aws/ebs, confirm that you are logged into the same account and AWS Region where you ran the shell script, and check your shell for error messages.</td>
<td>AWS administrator, General AWS</td>
</tr>
<tr>
<td>Configure additional accounts or AWS Regions.</td>
<td>As needed for your use case, repeat this epic for any additional accounts or AWS Regions.</td>
<td>AWS administrator, General AWS</td>
</tr>
</tbody>
</table>

Prevent creation of unencrypted instances

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a service control policy.</td>
<td>1. Open the AWS Organizations console at <a href="https://console.aws.amazon.com/organizations/v2/">https://console.aws.amazon.com/organizations/v2/</a>. 2. Create a new service control policy. For more information, see Creating a service control policy in the AWS Organizations documentation. 3. Add the contents of DenyUnencryptedEC2.json</td>
<td>AWS administrator, General AWS</td>
</tr>
</tbody>
</table>
1. You already downloaded this JSON file from the GitHub repository in the first epic.

2. Open the AWS Management Console, sign in to your AWS account, and navigate to the Organizations service.

3. In the left navigation pane, select ‘Create custom service control policy’ and then click ‘Create policy’.

4. Attach this policy to the organization root or any necessary organizational units (OUs). For more information, see [Attaching and detaching service control policies](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_policies_controlling.html) in the AWS Organizations documentation.

---

**Related resources**

**AWS service documentation**

- AWS CLI
- AWS Config
- AWS CloudFormation
- Amazon EC2
- AWS KMS
- AWS Organizations
- AWS Systems Manager Automation

**Other resources**

- [jq manual](https://jq.sh)
- [jq download](https://github.com/j�q/jq)

---

**Back up Sun SPARC servers in the Stromasys Charon-SSP emulator on the AWS Cloud**

*Created by Kevin Yung (AWS), Luis Ramos (Stromasys, Inc.), and Rohit Darji (AWS)*

<table>
<thead>
<tr>
<th>Environment</th>
<th>Technologies</th>
<th>Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production</td>
<td>Storage &amp; backup; Operating systems; DevOps</td>
<td>Oracle</td>
</tr>
</tbody>
</table>

**AWS services:** Amazon EFS; Amazon S3; AWS Storage Gateway; AWS Systems Manager; Amazon EC2
Summary

This pattern provides four options for backing up your Sun Microsystems SPARC servers after a migration from an on-premises environment to the Amazon Web Services (AWS) Cloud. These backup options help you to implement a backup plan that meets your organization's recovery point objective (RPO) and recovery time objective (RTO), uses automated approaches, and lowers your overall operational costs. The pattern provides an overview of the four backup options and steps to implement them.

If you use a Sun SPARC server hosted as a guest on a Stromasys Charon-SSP emulator, you can use the following three backup options:

- **Backup option 1: Stromasys virtual tape** – Use the Charon-SSP virtual tape feature to set up a backup facility in the Sun SPARC server and archive your backup files to Amazon Simple Storage Service (Amazon S3) and Amazon Simple Storage Service Glacier using AWS Systems Manager Automation.

- **Backup option 2: Stromasys snapshot** – Use the Charon-SSP snapshot feature to set up a backup facility for the Sun SPARC guest servers in Charon-SSP.

- **Backup option 3: Amazon Elastic Block Store (Amazon EBS) volume snapshot** – If you host the Charon-SSP emulator on Amazon Elastic Compute Cloud (Amazon EC2), you can use an Amazon EBS volume snapshot to create backups for a Sun SPARC file system.

If you use a Sun SPARC server hosted as a guest on hardware and Charon-SSP on Amazon EC2, you can use the following backup option:

- **Backup option 4: AWS Storage Gateway virtual tape library (VTL)** – Use a backup application with a Storage Gateway VTL Tape Gateway to back up the Sun SPARC servers.

If you use a Sun SPARC server hosted as a branded zone in a Sun SPARC server, you can use backup options 1, 2, and 4.

Stromasys provides software and services to emulate legacy SPARC, Alpha, VAX, and PA-RISC critical systems. For more information about migrating to the AWS Cloud using Stromasys emulation, see Rehosting SPARC, Alpha, or other legacy systems to AWS with Stromasys on the AWS Blog.

Prerequisites and limitations

**Prerequisites**

- An active AWS account.
- Existing Sun SPARC servers.
- Existing licenses for Charon-SSP. Licenses for Charon-SSP are available from AWS Marketplace and licenses for Stromasys Virtual Environment (VE) are available from Stromasys. For more information, contact Stromasys sales.
- Familiarity with Sun SPARC servers and Linux backups.
- Familiarity with Charon-SSP emulation technology. For more information about this, see Stromasys legacy server emulation in the Stromasys documentation.
- If you want to use the virtual tape facility or backup applications for your Sun SPARC servers file systems, you must create and configure the backup facilities for the Sun SPARC server file system.
- An understanding of RPO and RTO. For more information about this, see Disaster recovery objectives from the Reliability Pillar whitepaper in the AWS Well-Architected Framework documentation.
- To use Backup option 4, you must have the following:
  - A software-based backup application that supports a Storage Gateway VTL Tape Gateway. For more information about this, see Working with VTL devices in the AWS Storage Gateway documentation.
- Bacula Director or a similar backup application, installed and configured. For more information about this, see the Bacula Director documentation.

The following table provides information about the four backup options in this pattern.

<table>
<thead>
<tr>
<th>Backup options</th>
<th>Achieves crash consistency?</th>
<th>Achieves application consistency?</th>
<th>Virtual backup appliance solution?</th>
<th>Typical use cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1 – Stromasys virtual tape</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Sun SPARC server file systems backup with .tar or .zip files</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Application data backup</td>
</tr>
<tr>
<td>Option 2 – Stromasys snapshot</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Sun SPARC server snapshot</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Application data backup</td>
</tr>
<tr>
<td>Option 3 – Amazon EBS volume snapshot</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Sun SPARC server file systems snapshot</td>
</tr>
</tbody>
</table>
You can use AWS Backup to automate the Amazon EBS snapshot.

This backup option requires an automated script to flush in-flight transactions and configure a read-only or temporary stop of the EC2 instance during the Amazon EBS volume snapshot.

**Important:** This backup option might require application downtime or read-only mode to achieve application consistency.

<table>
<thead>
<tr>
<th>Option 4 – AWS Storage Gateway VTL</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can automatically back up Sun SPARC file system backup data to the VTL by using a backup agent.</td>
<td>Yes</td>
<td>This backup option requires an automated script to flush in-flight transactions and configure a read-only or temporary offline mode during the file system snapshot or application data dump.</td>
<td>A large fleet of Sun SPARC server file system backups</td>
</tr>
</tbody>
</table>

**Application data backup**

**Limitations**

- You can use this pattern’s approaches to back up individual Sun SPARC servers, but you can also use these backup options for shared data if you have applications that run in a cluster.

**Architecture**

**Backup option 1: Stromasys virtual tape**
Typically, if you run fewer than ten Sun SPARC servers in your data center, the servers are configured to back up file systems and application data by connecting to tape devices. To reduce change risks during a migration, you should use the same backup mechanism and reuse existing backup scripts. You can use the Charon-SSP emulator’s virtual tape feature to set up a virtual tape device in Charon-SSP Manager and make it available to the Sun SPARC guest server. The following is a typical Sun SPARC server backup process using tape devices:

1. The Sun SPARC server mounts a backup tape device as a small computer system interface (SCSI) device. The Sun SPARC server then uses a backup script that follows a customized runbook to back up the configuration and application data.
2. A schedule is set up to initiate the backup script and save the backup in a tape device.
3. After the backup is complete, the tape is ejected from the tape device and scheduled for offsite storage. Authorized personnel pick up the ejected disk and replace it with a rotation disk.

On the AWS Cloud, operations are automated to simplify this process and meet the shared responsibility model. The following diagram shows the setup for Sun SPARC servers and backups using backup option 1.

The diagram shows the following workflow:

1. Migrate the Sun SPARC servers to Amazon EC2 using the Charon-SSP Amazon Machine Image (AMI).
2. Create and attach the AWS Key Management Service (AWS KMS) encrypted EBS volumes as data volumes to Amazon EC2.
3. Configure SCSI devices using the Physical disk type in Charon-SSP Manager.
4. Create SCSI devices using the Virtual tape type in Charon-SSP Manager and map the SCSI device to the virtual tape file assigned to the Sun SPARC server. **Note:** Use Charon-SSP Manager to input the virtual tape name, location path for the virtual tape container file, and tape size in megabytes.
5. Map the SCSI devices to the data volumes available in Amazon EC2. **Note:** Use Charon-SSP Manager to create the SCSI mapping in the Virtual Machine Settings window. Enter the virtual tape device into the SCSI settings. The list of SCSI devices shows the SCSI ID, Device type, LUN ID, and the path to the devices.
6. Provision an AWS KMS encrypted Amazon Elastic File System (Amazon EFS) file share and mount the Amazon EFS file share to Amazon EC2 with Transport Layer Security (TLS) enabled.

7. Create virtual tape container files in the Amazon EFS file share for each guest Sun SPARC guest server.

8. Verify that the virtual tape device is configured in the Sun SPARC guest server and configure the backup script to use the virtual tape device. **Note** You can run the following command to validate that the virtual tape device is recognized in the Solaris operating system (OS):

```bash
mt -f /dev/rmt/1
config "Charon Virtual Tape", "Charon Virtual Tape", "CFGCHARONVIRTUALTAPE";
CFGCHARONVIRTUALTAPE =
2,0x2D,0x10659,4,0x00,0x00,0x00,0x120,3600,3600,3600,3600
```

9. Create a Systems Manager Automation runbook to orchestrate the backup process and rotate the tape container file. To initiate the backup, the automation starts by rewinding the tape and then runs the backup command. You can use the following sample backup commands:

- `mt -f <tape-device-name> rewind`
- `tar -cvf <tape-device-name> <files-to-backup>`
- `mt -f <tape-device-name> offline`

**Backup option 2 - Stromasys snapshot**

The Charon-SSP emulator provides guest snapshot capability and the emulated SPARC server can be suspended by Charon-SSP Manager. When the guest server is suspended, the system’s memory content is saved in a directory. A backup script or application can back up the snapshot file and the guest virtual disk files. Because of a SPARC pause, the Sun SPARC guest server requires downtime when the backup starts. We recommend this backup approach for applications that are deployed in a cluster where temporary reduced capacity is acceptable during the maintenance window.

The following diagram shows the setup of the Sun SPARC servers and backups using backup option 2.

The diagram shows the following workflow:

1. Migrate the Sun SPARC servers to Amazon EC2 using the Charon-SSP AMI.
2. Create and attach the AWS KMS encrypted EBS volumes as data volumes to Amazon EC2.

3. Provision an AWS KMS encrypted Amazon EFS file share and mount it to Amazon EC2 with TLS enabled.

4. Create virtual disks in Charon-SSP Manager and save the virtual disk file in the data volumes. You can use the mkdskcmd command line in the Charon-SSP package to create the virtual disk. **Note:** You can use the mkdskcml -o /usr/local/vm/leela/disk0.vdisk -d ST446452W and mkdskcml - o /usr/local/vm/bender/disk0.vdisk -z 4096 -c 16384 commands to create the virtual disk files.

5. Create a snapshot folder in the Amazon EFS file share for each Sun SPARC guest server.

6. Set up the folder path in the **Location snapshot** of the Charon-SSP Manager's **Preferences** window.

7. Create a Systems Manager Automation runbook to orchestrate the backup process and rotate the tape container file.

**Important:** For Step 6, you must configure Charon-SSP Manager to use a snapshot folder. In the **Preferences** window for Charon SSP Manager's SPARC guest server, enter the full path of the snapshot folder in the Linux system into the **Location** text field in the snapshot configuration group. For example, if Amazon EFS is mounted to `/home/` and a directory called `ssp-snapshot` is created to save snapshots, then you should enter `/home/ssp-snapshot`. If the Sun SPARC guest server is started through the Charon-SSP command line, then you must use `-snapshot=path` to specify the path to store snapshot files.

The following are sample backup Linux commands that you can use to back up a Sun SPARC guest server:

1. In the Linux host, run the following command to suspend the SPARC guest server: `kill -SIGTSTP <SSP-PID>`

2. Upload the Sun SPARC guest server snapshot file and virtual disks files from the Linux host to Amazon S3 with AWS Command Line Interface (AWS CLI): `aws s3 cp <snapshot_file> s3://<bucket-name>/<timestamp>/`

3. Restart the Sun SPARC guest server: `ssp4u -f <sparc-guest-config-file> -a <alias> -S <snapshot-location>`

**Backup option 3: Amazon EBS volume snapshot**

You can also use backup solutions from storage hardware in the data center (for example, taking snapshots of data volumes). If you migrated Solaris instances, you can use a similar approach with EBS volume snapshots.

The following diagram shows the setup of Sun SPARC servers and backups using backup option 3.
The diagram shows the following workflow:

1. Migrate the Sun SPARC servers to Amazon EC2 using the Charon-SSP AMI.
2. Create and attach the AWS KMS encrypted EBS volumes as data volumes to Amazon EC2.
3. Set up the Charon-SSP emulation configuration to use physical disks as virtual SCSI devices.
4. Mount the SCSI storage device in the Solaris guest.
5. Set up AWS Backup to create Amazon EBS multi-volume, crash-consistent backups. For more information about this, see Creating Amazon EBS multi-volume, crash-consistent backups in the AWS Backup documentation.

Note: This backup option achieves crash-consistent snapshots but most modern file systems can recover open files without corrupting them. This backup option doesn't always provide the consistency required for mission-critical applications that require in-flight transactions or processing to be fully committed before backing up. To achieve application consistency, we recommend that you use backup scripts with backup option 1 or 4.

Backup Option 4 – AWS Storage Gateway VTL

If you need backup continuity when migrating your Sun SPARC servers, then your backup solution cannot have an impact during the migration and you must have options to recover or retrieve backup data when required. This backup approach uses a Storage Gateway Tape Gateway and VTL stored in Amazon S3 to replace your on-premises tape library facilities. By using this approach, you can keep almost all of your existing configuration for some of your enterprise backup solutions.

We tested a SPARC server using Solaris 11 on an EC2 instance running the Charon-SSP emulator. In a typical setup, enterprise backup software must be installed and configured on the Solaris 11 server.

The following diagram shows the setup of Sun SPARC servers and backups using backup option 4.
The diagram shows the following workflow:

1. Set up an Amazon EC2 Linux instance and install Bacula Director.
2. Create a multi-AZ Amazon Relational Database Service (Amazon RDS) for MySQL DB instance for the Bacula Catalog database.
3. Set up an Amazon EC2 Linux instance and install the Charon-SSP emulator.
4. In Charon-SSP Manager, create a Sun SPARC virtual instance by following the Charon-SSP user guide in the Stromasys documentation. Install or restore the SPARC image on the virtual instance.
5. Configure Storage Gateway in a virtual private cloud (VPC) and set it up as a VTL Tape Gateway. For more information about this, see Creating a Tape Gateway in the AWS Storage Gateway documentation.
6. In Bacula Director, configure an Amazon RDS endpoint for the Bacula Catalog database.
7. In Bacula Director, configure the storage pool to use the Storage Gateway VTL.
8. In the Solaris 11 server, configure the Bacula file daemon to use the Bacula Director setup.

**Note:** For a lower risk and faster SPARC migration, you can create a backup pool through the Storage Gateway VTL Tape Gateway, create a backup policy for the backup pool, incrementally add backup targets to use the newly created backup policy, and validate backup results. When you migrate the server, it can be recovered from its backup on Amazon S3 to the emulated Solaris OS. A final delta synchronization is performed during a maintenance window before the cutover.

**Tools**

**Backup option 1: Stromasys virtual tape**

- **Stromasys Charon-SSP emulator** – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including operating systems (OSs) such as SunOS or Solaris, their layered products, and applications.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.
AWS Prescriptive Guidance Patterns
Tools

- **Amazon EFS** – Amazon Elastic File System (Amazon EFS) provides a simple, serverless, set-and-forget elastic file system for use with AWS Cloud services and on-premises resources.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- **Amazon S3 Glacier** – Amazon Simple Storage Service Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.
- **AWS Systems Manager Automation** – Automation, a capability of AWS Systems Manager, simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources.

**Backup option 2: Stromasys snapshot**

- **Stromasys Charon-SSP emulator** – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including OSs such as SunOS or Solaris, their layered products, and applications.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.
- **Amazon EFS** – Amazon Elastic File System (Amazon EFS) provides a simple, serverless, set-and-forget elastic file system for use with AWS Cloud services and on-premises resources.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- **Amazon S3 Glacier** – Amazon Simple Storage Service Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.
- **AWS Systems Manager Automation** – Automation, a capability of AWS Systems Manager, simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources.

**Backup option 3: Amazon EBS volume snapshot**

- **Stromasys Charon-SSP emulator** – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including OSs such as SunOS or Solaris, their layered products, and applications.
- **AWS Backup** – AWS Backup is a fully-managed data protection service that makes it easy to centralize and automate across AWS services, in the cloud, and on premises.
- **Amazon EBS** – Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.

**Backup option 4: AWS Storage Gateway VTL**

- **Stromasys Charon-SSP emulator** – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including OSs such as SunOS or Solaris, their layered products, and applications.
- **Bacula** – Bacula is an open-source, enterprise-level computer backup system. For more information about whether your existing backup application supports Tape Gateway, see Supported third-party backup applications for a Tape Gateway in the AWS Storage Gateway documentation.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.
- **Amazon RDS for MySQL** – Amazon Relational Database Service (Amazon RDS) supports DB instances running several versions of MySQL.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- **Amazon S3 Glacier** – Amazon Simple Storage Service Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.
- **AWS Storage Gateway** – Storage Gateway connects an on-premises software appliance with cloud-based storage to provide seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure.

## Epics

### Backup option 1 – Create a Stromasys virtual tape backup

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon EFS shared file system for virtual tape file storage.</td>
<td>Sign in to the AWS Management Console or use AWS CLI to create an Amazon EFS file system. For more information about this, see <a href="#">Create an Amazon EFS file system</a> in the Amazon EFS documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Configure the Linux host to mount the shared file system.</td>
<td>Install the Amazon EFS driver on the Amazon EC2 Linux instance and configure the Linux OS to mount the Amazon EFS shared file system during startup. For more information about this, see <a href="#">Mounting file systems using the EFS mount helper</a> in the Amazon EFS documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Install the Charon-SSP emulator.</td>
<td>Install the Charon-SSP emulator on the Amazon EC2 Linux instance. For more information about this, see <a href="#">Setting up an AWS Cloud instance for Charon-SSP</a> in the Stromasys documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Create a virtual tape file container in the shared file system for each Sun SPARC guest server.</td>
<td>Run the <code>touch &lt;vtape-container-name&gt;</code> command to create a virtual tape file container in the shared file system for each Sun SPARC guest server deployed in the Charon-SSP emulator.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Configure Charon-SSP Manager to create virtual tape devices for the Sun SPARC guest servers.</td>
<td>Log in to Charon-SSP Manager, create virtual tape devices, and configure them to use the virtual</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Tape container files for each Sun SPARC guest server.</td>
<td>For more information about this, see the Charon-SSP 5.2 for Linux user guide in the Stromasys documentation.</td>
<td></td>
</tr>
<tr>
<td>Validate that the virtual tape device is available in the Sun SPARC guest servers.</td>
<td>Log in to each Sun SPARC guest server and run the <code>mt -f /dev/rmt/1</code> command to validate that the virtual tape device is configured in the OS.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Develop the Systems Manager Automation runbook and automation.</td>
<td>Develop the Systems Manager Automation runbook and set up maintenance windows and associations in Systems Manager for scheduling the backup process. For more information about this, see Automation walkthroughs and Setting up maintenance windows in the AWS Systems Manager documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Configure Systems Manager Automation to archive rotated virtual tape container files.</td>
<td>Use the code sample from Back option 1 in the Additional information section to develop a Systems Manager Automation runbook to archive rotated virtual tape container files to Amazon S3 and Amazon S3 Glacier.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the Systems Manager Automation runbook for archiving and scheduling.</td>
<td>Deploy the Systems Manager Automation runbook and schedule it to automatically run in Systems Manager. For more information about this, see Automation walkthroughs in the Systems Manager documentation.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

**Backup option 2 – Create a Stromasys snapshot**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an Amazon EFS shared file system for virtual tape file storage.</td>
<td>Sign in to the AWS Management Console or use AWS CLI to create an Amazon EFS file system.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see <a href="#">Create your Amazon EFS file system</a> in the Amazon EFS documentation.</td>
<td></td>
</tr>
<tr>
<td>Configure the Linux host to mount the shared file system.</td>
<td>Install the Amazon EFS driver in the Amazon EC2 Linux instance and configure the Linux OS to mount the Amazon EFS shared file system during startup. For more information about this, see <a href="#">Mounting file systems using the EFS mount helper</a> in the Amazon EFS documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Install the Charon-SSP emulator.</td>
<td>Install the Charon-SSP emulator on the Amazon EC2 Linux instance. For more information about this, see <a href="#">Setting up an AWS Cloud instance for Charon-SSP</a> in the Stromasys documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Configure the Sun SPARC guest servers to start up with the snapshot option.</td>
<td>Use Charon-SSP Manager to set up the snapshot option for each Sun SPARC guest servers. For more information about this, see the <a href="#">Charon-SSP 5.2 for Linux user guide</a> in the Stromasys documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Develop the Systems Manager Automation runbook.</td>
<td>Use the code sample from <a href="#">Backup option 2</a> in the Additional information section to develop a Systems Manager Automation runbook to remotely run the snapshot command on a Sun SPARC guest server during a maintenance window.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td>Deploy the Systems Manager Automation runbook and set up the association to the Amazon EC2 Linux hosts.</td>
<td>Deploy the Systems Manager Automation runbook and set up maintenance windows and associations in Systems Manager for scheduling the backup process. For more information about this, see <a href="#">Automation walkthroughs</a> and <a href="#">Setting up Maintenance Windows</a> in the AWS Systems Manager documentation.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>
## AWS Prescriptive Guidance Patterns

### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive snapshots into long-term storage.</td>
<td>Use the runbook sample code from the <em>Additional information</em> section to develop a Systems Manager Automation runbook to archive snapshot files to Amazon S3 and Amazon S3 Glacier.</td>
<td>Cloud architect</td>
</tr>
</tbody>
</table>

### Backup option 3 – Create an Amazon EBS volume snapshot

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the Charon-SSP emulator.</td>
<td>Install the Charon-SSP emulator on the Amazon EC2 Linux instance.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see <a href="#">Setting up an AWS Cloud instance for Charon-SSP in the Stromasys documentation</a>.</td>
<td></td>
</tr>
<tr>
<td>Create EBS volumes for the Sun SPRAC guest servers.</td>
<td>Sign in to the AWS Management Console, open the Amazon EBS console, and then create EBS volumes for the Sun SPRAC guest servers.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see <a href="#">Setting up an AWS Cloud instance for Charon-SSP in the Stromasys documentation</a>.</td>
<td></td>
</tr>
<tr>
<td>Attach the EBS volumes to the Amazon EC2 Linux instance.</td>
<td>On the Amazon EC2 console, attach the EBS volumes to the Amazon EC2 Linux instance.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see <a href="#">Attach an Amazon EBS volume to an instance in the Amazon EC2 documentation</a>.</td>
<td></td>
</tr>
<tr>
<td>Map EBS volumes as SCSI drives in the Charon-SSP emulator.</td>
<td>Configure Charon-SSP Manager to map the EBS volumes as SCSI drives in the Sun SPARC guest servers.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td></td>
<td>For more information about this, see the <a href="#">SCSI storage configuration section of the Charon-SSP V5.2 for Linux guide</a> in the Stromasys documentation.</td>
<td></td>
</tr>
<tr>
<td>Configure the AWS Backup schedule for snapshotting the EBS volumes.</td>
<td>Set up AWS Backup policy and schedules to snapshot the EBS volumes.</td>
<td>AWS DevOps</td>
</tr>
</tbody>
</table>
### Backup option 4 – Create an AWS Storage Gateway VTL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create a Tape Gateway device.</strong></td>
<td>Sign in to the AWS Management Console, open the AWS Storage Gateway console, and then create a Tape Gateway device in a VPC. For more information about this, see Creating a gateway in the AWS Storage Gateway documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td><strong>Create an Amazon RDS DB instance for the Bacula Catalog.</strong></td>
<td>Open the Amazon RDS console and create an Amazon RDS for MySQL DB instance. For more information about this, see Creating a MySQL DB instance and connecting to a database on a MySQL DB instance in the Amazon RDS documentation.</td>
<td>Cloud architect</td>
</tr>
<tr>
<td><strong>Deploy the backup application controller in the VPC.</strong></td>
<td>Install Bacula on the EC2 instance, deploy the backup application controller, and then configure the backup storage to connect with the Tape Gateway device. You can use the sample Bacula Director storage daemon configuration in the Bacula-storage-daemon-config.txt file (attached). For more information about this, see the Bacula documentation.</td>
<td>AWS DevOps</td>
</tr>
<tr>
<td><strong>Set up backup application on the Sun SPARC guest servers.</strong></td>
<td>Set up a second client to install and set up the backup application on the Sun SPARC guest servers by using the sample Bacula configuration in the SUN-SPARC-Guest-Bacula-Config.txt file (attached).</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Set up the backup configuration and schedule.</td>
<td>Set up backup configuration and schedules in the backup application controller by using the sample Bacula Director configuration in the Bacula-Directory-Config.txt file (attached). For more information about this, see the Bacula documentation.</td>
<td>DevOps engineer</td>
</tr>
<tr>
<td>Validate that the backup configuration and schedules are correct.</td>
<td>Follow the instruction from the Bacula documentation to perform the validation and backup testing for your setup in the Sun SPARC guest servers. For example, you can use the following commands to validate the configuration files: • bacula-dir -t -c bacula-dir.conf • bacula-fd -t -c bacula-fd.conf • bacula-sd -t -c bacula-sd.conf</td>
<td>DevOps engineer</td>
</tr>
</tbody>
</table>

Related resources

- Charon virtual SPARC with VE licensing
- Charon virtual SPARC
- Using cloud services and object storage with Bacula Enterprise Edition
- Disaster recovery (DR) objectives
- AWS Storage Gateway - Working with VTL devices
- Charon legacy system emulation solutions

Additional information

Backup option 1 – Create a Stromasys virtual tape

You can use the following sample Systems Manager Automation runbook code to automatically start the backup and then swap the tapes:

```bash
... # example backup script saved in SUN SPARC Server 
#!/usr/bin/bash 
mt -f rewind 
tar -cvf 
mt -f offline 
... 
```
mainSteps:
- action: aws:runShellScript
  name: startBackup
  inputs:
  onFailure: Abort
  timeoutSeconds: "1200"
  runCommand:
  - |
    user={{BACKUP_USER}}
    keypair={{KEYPAIR_PATH}}
    server={{SUN_SPARC_IP}}
    backup_script={{BACKUP_SCRIPT}}
    ssh -i $keypair $user@$server -c "/usr/bin/bash $backup_script"
- action: aws:runShellScript
  name: swapVirtualDiskContainer
  inputs:
  onFailure: Abort
  timeoutSeconds: "1200"
  runCommand:
  - |
    mv {{TapeBackupContainerFile}} {{TapeBackupContainerFile}}.$(date +%s)
    touch {{TapeBackupContainerFile}}
  - action: aws:runShellScript
    name: uploadBackupArchiveToS3
    inputs:
    onFailure: Abort
    timeoutSeconds: "1200"
    runCommand:
    - |
      aws s3 cp {{TapeBackupContainerFile}} s3://{{BACKUP_BUCKET}}/
      {{SUN_SPARC_IP}}/$(date '+%Y-%m-%d')/
      ...

Backup option 2 – Stromasys snapshot

You can use the following sample Systems Manager Automation runbook code to automate the backup process:

...
Backup option 4 – AWS Storage Gateway VTL

If you use Solaris non-global zones to run virtualized legacy Sun SPARC servers, the backup application approach can be applied to non-global zones running in the Sun SPARC servers (for example, the backup client can run inside the non-global zones). However, the backup client can also run in the Solaris host and take snapshots of the non-global zones. The snapshots can then be backed up on a tape.

The following sample configuration adds the file system that hosts the Solaris non-global zones into the backup configuration for the Solaris host:

```yaml
FileSet {
    Name = "Branded Zones"
    Include {
        Options {
            signature = MD5
        }
        File = /zones
    }
}
```

Attachments

attachment.zip

Back up and archive data to Amazon S3 with Veeam Backup & Replication
Summary

This pattern details the process for sending backups created by Veeam Backup & Replication to supported Amazon Simple Storage Service (Amazon S3) object storage classes by using the Veeam scale-out backup repository capability.

Veeam supports multiple Amazon S3 storage classes to best fit your specific needs. You can choose the type of storage based on the data access, resiliency, and cost requirements of your backup or archive data. For example, you can store data that you don’t plan to use for 30 days or longer in Amazon S3 infrequent access (IA) for lower cost. If you’re planning to archive data for 90 days or longer, you can use Amazon Simple Storage Service Glacier with Veeam’s archive tier. You can also use S3 Object Lock to make backups immutable within Amazon S3.

This pattern doesn’t cover how to set up Veeam Backup & Replication with a tape gateway in Amazon Web Services (AWS) Storage Gateway. For information about that topic, see Veeam Backup & Replication using AWS VTL Gateway - Deployment Guide on the Veeam website.

Prerequisites and limitations

Prerequisites

- Veeam Backup & Replication, including Veeam Availability Suite or Veeam Backup Essentials, installed (you can register for a free trial)
- Veeam Backup & Replication license with Enterprise or Enterprise Plus functionality, which includes Veeam Universal License (VUL)
- An active AWS Identity and Access Management (IAM) user account with access to an Amazon S3 bucket
- An active IAM user account with access to Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Virtual Private Cloud (Amazon VPC) (if utilizing archive tier)
- Network connectivity from on premises to AWS services with available bandwidth for backup and restore traffic through a public internet connection or an AWS Direct Connect public virtual interface (VIF)

Limitations

- Veeam doesn't support S3 Lifecycle policies on any S3 buckets that are used as Veeam object storage repositories. These include policies with Amazon S3 storage class transitions and S3 Lifecycle expiration rules. Veeam must be the sole entity that manages these objects. Enabling S3 Lifecycle policies might have unexpected results, including data loss.

Product versions

- Veeam Backup & Replication v9.5 Update 4 or later (backup only or capacity tier)
- Veeam Backup & Replication v10 or later (backup or capacity tier and S3 Object Lock)
- Veeam Backup & Replication v11 or later (backup or capacity tier, archive or archive tier, and S3 Object Lock)
- S3 Standard
• S3 Standard-IA
• S3 One Zone-IA
• S3 Glacier (v11 only)
• S3 Glacier Deep Archive (v11 only)

Architecture

Source technology stack

• On-premises Veeam Backup & Replication installation with connectivity from a Veeam backup server or a Veeam gateway server to Amazon S3

Target technology stack

• Amazon S3
• Amazon VPC and Amazon EC2 (if using archive tier)

Target architecture

The following diagram shows the scale-out backup repository architecture.

Veeam Backup and Replication software protects data from logical errors such as system failures, application errors, or accidental deletion. In this diagram, backups are run on premises first, and a secondary copy is sent directly to Amazon S3. A backup represents a point-in-time copy of the data.

The workflow consists of three primary components that are required for tiering or copying backups to Amazon S3, and one optional component:

• Veeam Backup & Replication (1) – The backup server that is responsible for coordinating, controlling, and managing backup infrastructure, settings, jobs, recovery tasks, and other processes.
• Veeam gateway server (not shown in the diagram) – An optional on-premises gateway server that is required if the Veeam backup server doesn’t have outbound connectivity to Amazon S3.
• Scale-out backup repository (2) – Repository system with horizontal scaling support for multi-tier storage of data. The scale-out backup repository consists of one or more backup repositories that provide fast access to data and can be expanded with Amazon S3 object storage repositories for long-
term storage (capacity tier) and archiving (archive tier). Veeam uses the scale-out backup repository to tier data automatically between local (performance tier) and Amazon S3 object storage (capacity and archive tiers).

- **Amazon S3 (3)** – AWS object storage service that offers scalability, data availability, security, and performance.

**Automation and scale**

You can automate the creation of IAM resources and S3 buckets by using the AWS CloudFormation templates provided in the VeeamHub GitHub repository. The templates include both standard and immutable options.

**Tools**

**Tools and AWS services**

- **Veeam Backup & Replication** — Veeam Backup & Replication is a solution from Veeam for protecting, backing up, replicating, and restoring your virtual and physical workloads.
- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

**Code**

Use the CloudFormation templates provided in the VeeamHub GitHub repository to automatically create the IAM resources and S3 buckets for this pattern. If you prefer to create these resources manually, follow the steps in the Epics section.

**Epics**

**Configure Amazon S3 storage in your account**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM user.</td>
<td>Follow the instructions in the IAM documentation to create an IAM user with programmatic access. Veeam uses this entity to authenticate with AWS</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>1. Sign in to the AWS Management Console and open the Amazon S3 console at <a href="https://console.aws.amazon.com/s3/">https://console.aws.amazon.com/s3/</a>.</td>
<td><strong>Note</strong> Alternatively, you can use the CloudFormation templates provided in the VeeamHub GitHub repository to create an IAM user and S3 bucket for this pattern.</td>
<td>AWS administrator</td>
</tr>
</tbody>
</table>
| 2. If you don't already have an existing S3 bucket to use as the target storage, choose **Create bucket**, and specify a bucket name, AWS Region, and bucket settings. | • We recommend that you enable the **Block Public Access option** for the S3 bucket and set up the access and user permission policies to meet your organization's requirements. For an example, see the Amazon S3 documentation.  
• We recommend that you enable **S3 Object Lock**, even if you don't intend to use it right away. This setting can be enabled only at the time of S3 bucket creation. | For more information, see Creating a bucket in the Amazon S3 documentation. |
Add Amazon S3 and Amazon S3 Glacier (or S3 Glacier Deep Archive) to Veeam Backup & Replication

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the New Object Repository wizard.</td>
<td>Before you set up the object storage and scale-out backup repositories in Veeam, you must add the Amazon S3 and Amazon S3 Glacier storage repositories that you want to use for the capacity and archive tiers. In the next epic, you’ll connect these storage repositories to your scale-out backup repository.</td>
<td>AWS administrator, App owner</td>
</tr>
<tr>
<td>1. On the Veeam console, open the <strong>Backup Infrastructure</strong> view.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. In the inventory pane, choose the <strong>Backup Repositories</strong> node, and then choose <strong>Add Repository</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. In the <strong>Add Backup Repository</strong> dialog box, choose <strong>Object Storage</strong>, <strong>Amazon S3</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Amazon S3 storage for the capacity tier.</td>
<td>1. In the <strong>Amazon Cloud Storage Services</strong> dialog box, choose <strong>Amazon S3</strong>.</td>
<td>AWS administrator, App owner</td>
</tr>
<tr>
<td>2. At the <strong>Name</strong> step of the wizard, specify the object storage name and a brief description, such as the creator and creation date.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. At the <strong>Account</strong> step of the wizard, specify the object storage account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. For <strong>Credentials</strong>, choose the user credentials that you created in the first epic to access your Amazon S3 object storage.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. For <strong>AWS region</strong>, choose the AWS Region where the Amazon S3 bucket is located.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. At the <strong>Bucket</strong> step of the wizard, specify object storage settings.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. For <strong>Data center region</strong>, choose the AWS Region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>where the Amazon S3 bucket is located.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. For <strong>Bucket</strong>, choose the S3 bucket that you created in the first epic.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. For <strong>Folder</strong>, create or select a cloud folder to map your object storage repository to.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d. If you want to enable immutability, choose <strong>Make recent backups immutable for X days</strong> and set the period of time during which your backups should be locked. Note that enabling immutability results in increased costs because of the increased number of API calls to Amazon S3 from Veeam.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td><strong>At the Summary step</strong> of the wizard, review the configuration information, and then choose <strong>Finish</strong>.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Add Amazon S3 Glacier storage for the archive tier.</td>
<td>If you want to create an archive tier, use the IAM permissions detailed in the Additional information (p. 1814) section. 1. Launch the New Object Repository wizard as described previously. 2. In the Amazon Cloud Storage Services dialog box, choose Amazon S3 Glacier. 3. At the Name step of the wizard, specify the object storage name and a brief description, such as the creator and creation date. 4. At the Account step of the wizard, specify the object storage account. a. For Credentials, choose the user credentials you created in the first epic to access your Amazon S3 Glacier object storage. b. For AWS region, choose the AWS Region where the Amazon S3 bucket is located. 5. At the Bucket step of the wizard, specify object storage settings. a. For Data center region, choose the AWS Region. b. For Bucket, choose an S3 bucket to store your backup data. This can be the same bucket you used for the capacity tier. c. For Folder, create or select a cloud folder to map your object storage repository to. d. If you want to enable immutability, choose Make recent backups immutable for the entire duration of their retention policy. Note that enabling immutability results in increased costs because of the increased number</td>
<td>AWS administrator, App owner</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>e. If you want to use S3 Glacier Deep Archive as your archival storage class, choose <strong>Use the Deep Archive Storage Class</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. At the <strong>Proxy Appliance</strong> step of the wizard, configure the auxiliary instance that is used to transfer the data from Amazon S3 to Amazon S3 Glacier. You can use the default settings or configure each setting manually. To configure the settings manually:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a. Choose <strong>Customize</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. For <strong>EC2 instance type</strong>, choose the instance type for the proxy appliance, based on your speed and cost requirements for transferring the backup files to the archive tier of your scale-out backup repository.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c. For <strong>Amazon VPC</strong>, choose the VPC for the target instance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d. For <strong>Subnet</strong>, choose the subnet for the proxy appliance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e. For <strong>Security group</strong>, choose the security group to associate with the proxy appliance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f. For <strong>Redirector port</strong>, specify the TCP port for routing requests between the proxy appliance and backup infrastructure components.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g. Choose <strong>OK</strong> to confirm your settings.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. At the <strong>Summary</strong> step of the wizard, review the configuration information, and then choose <strong>Finish</strong>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Add scale-out backup repositories

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch the New Scale-Out Backup Repository wizard.</td>
<td>1. On the Veeam console, open the <strong>Backup Infrastructure</strong> view.&lt;br&gt;2. In the inventory pane, choose <strong>Scale-out Repositories</strong>, and then choose <strong>Add Scale-out Repository</strong>.</td>
<td>App owner, AWS systems administrator</td>
</tr>
<tr>
<td>Add a scale-out backup repository and configure capacity and archive tiers.</td>
<td>1. At the <strong>Name</strong> step of the wizard, specify the name and a brief description of the scale-out backup repository.&lt;br&gt;2. If needed, add performance extents. You can also use your existing Veeam local backup repository as your performance tier.&lt;br&gt;3. Choose <strong>Advanced</strong>, and specify additional options for the scale-out backup repository.&lt;br&gt;   • Choose <strong>Use per-machine backup files</strong> to create a separate backup file for each machine and write these files to the backup repository in multiple streams simultaneously. This option is recommended for better storage and compute resource utilization.&lt;br&gt;   • Choose <strong>Perform full backup when required extent is offline</strong> to create a full backup file in case an extent that contains restore points for an incremental backup goes offline. This option requires free space in the scale-out backup repository to host a full backup file.&lt;br&gt;4. At the <strong>Policy</strong> step of the wizard, specify the backup placement policy for the repository.&lt;br&gt;   • Choose <strong>Data locality</strong> to store full and incremental backup files that belong to the same chain together,</td>
<td>App owner, AWS systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>to the same performance extent. You can store files that belong to a new backup chain to the same performance extent or to another one (unless you use a deduplicating storage appliance as a performance extent).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Choose <strong>Performance</strong> to store full and incremental backup files to different performance extents. This option requires a fast and reliable network connection. If you choose <strong>Performance</strong>, you can restrict the types of backup files to store on each performance extent. For example, you can store full backup files on one extent and incremental backup files on other extents. To choose file types:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a. Choose <strong>Customize</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. In the <strong>Backup Placement Settings</strong> dialog box, choose a performance extent, and then choose <strong>Edit</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Choose the type of backup files you want to store on the extent.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>At the <strong>Capacity Tier</strong> step of the wizard, configure the long-term storage tier that you want to attach to the scale-out backup repository.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a. Choose <strong>Extend scale-out backup repository capacity with object storage</strong>. For the object storage repository, choose the Amazon S3 storage for the capacity tier that you added in the previous epic.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Choose <strong>Window</strong> to select a time window for moving or copying data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Choose <strong>Copy backups to object storage as soon as they are created</strong> to copy</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>all or only recently created backup files to the capacity extent.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d. Choose <strong>Move backups to object storage as they age out of the operational restores window</strong> to transfer inactive backup chains to the capacity extent. In the <strong>Move backup files older than X days</strong> field, specify a duration after which backup files should be offloaded. (To offload inactive backup chains on the day they were created, specify 0 days.) You can also choose <strong>Override</strong> to move backup files sooner if the scale-out backup repository has reached a threshold that you specify.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e. Choose <strong>Encrypt data uploaded to object storage</strong> and specify a password to encrypt all data and their metadata for offloading. Choose <strong>Add or Manage passwords</strong> to specify a new password.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>At the <strong>Archive Tier</strong> step of the wizard, configure the archive storage tier that you want to attach to the scale-out backup repository. (This step doesn't appear if you skipped adding Amazon S3 Glacier storage.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a. Choose <strong>Archive GFS full backups to object storage</strong>. For the object storage repository, choose the Amazon S3 Glacier storage you added in the previous epic.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. For <strong>Archive GFS backups older than N days</strong>, choose a time window for moving files to the archive extent. (To archive inactive backup chains on the day they were created, specify 0 days.)</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>7. At the <strong>Summary</strong> step of the wizard, review the configuration of the scale-out backup repository, and then choose <strong>Finish</strong>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

- Creating an IAM user in your AWS account (IAM documentation)
- Creating a bucket (Amazon S3 documentation)
- Blocking public access to your Amazon S3 storage (Amazon S3 documentation)
- Using S3 Object Lock (Amazon S3 documentation)
- Veeam technical documentation
- How to Create Secure IAM Policy for Connection to S3 Object Storage (Veeam documentation)

**Additional information**

The following sections provide sample IAM policies you can use when you create an IAM user in the Epics (p. 1805) section of this pattern.

**IAM policy for capacity tier**

**Note** Change the name of the S3 buckets in the example policy from `<yourbucketname>` to the name of the S3 bucket that you want to use for Veeam capacity tier backups.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "s3:GetObjectVersion",
            "s3:ListBucketVersions",
            "s3:ListBucket",
            "s3:PutObjectLegalHold",
            "s3:GetBucketVersioning",
            "s3:GetObjectLegalHold",
            "s3:GetBucketObjectLockConfiguration",
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetEncryptionConfiguration",
            "s3:PutObjectRetention",
            "s3:GetBucketObjectLockConfiguration",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:GetBucketLocation"
         ],
         "Resource": [
            "arn:aws:s3:::<yourbucketname>/*",
            "arn:aws:s3:::<yourbucketname>"
         ]
      }
   ]
}
IAM policy for archive tier

**Note** Change the name of the S3 buckets in the example policy from `<yourbucketname>` to the name of the S3 bucket that you want to use for Veeam archive tier backups.

To use your existing VPC, subnet, and security groups:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject",
        "s3:PutObject",
        "s3:GetObject",
        "s3:RestoreObject",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:GetBucketVersioning",
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation",
        "s3:GetBucketObjectLockConfiguration",
        "s3:GetObjectRetention",
        "s3:GetObjectVersion",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:DeleteObjectVersion",
        "s3:ListBucketVersions",
        "ec2:DescribeInstances",
        "ec2:CreateKeyPair",
        "ec2:DescribeKeyPairs",
        "ec2:RunInstances",
        "ec2:DeleteKeyPair",
        "ec2:DescribeVpcAttribute",
        "ec2:CreateTags",
        "ec2:DescribeSubnets",
        "ec2:TerminateInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeImages",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    }
  ]
}
```

To create new VPC, subnet, and security groups:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:ListBucket"
      ],
      "Resource": "*"
    }
  ]
}
```
Migrate data from an on-premises Hadoop environment to Amazon S3 using DistCp with AWS PrivateLink for Amazon S3

Created by Jason Owens (AWS), Andres Cantor (AWS), Jeff Klopfenstein (AWS), Bruno Rocha Oliveira (AWS), and Samuel Schmidt (AWS)
Summary

This pattern demonstrates how to migrate nearly any amount of data from an on-premises Apache Hadoop environment to the Amazon Web Services (AWS) Cloud by using the Apache open-source tool DistCp with AWS PrivateLink for Amazon Simple Storage Service (Amazon S3). Instead of using the public internet or a proxy solution to migrate data, you can use AWS PrivateLink for Amazon S3 to migrate data to Amazon S3 over a private network connection between your on-premises data center and an Amazon Virtual Private Cloud (Amazon VPC). If you use DNS entries in Amazon Route 53 or add entries in the /etc/hosts file in all nodes of your on-premises Hadoop cluster, then you are automatically directed to the correct interface endpoint.

This guide provides instructions for using DistCp for migrating data to the AWS Cloud. DistCp is the most commonly used tool, but other migration tools are available. For example, you can use offline AWS tools like AWS Snowball or AWS Snowmobile, or online AWS tools like AWS Storage Gateway or AWS DataSync. Additionally, you can use other open-source tools like Apache NiFi.

Prerequisites and limitations

Prerequisites

- An active AWS account with a private network connection between your on-premises data center and the AWS Cloud
- Hadoop, installed on premises with DistCp
- A Hadoop user with access to the migration data in the Hadoop Distributed File System (HDFS)
- AWS Command Line Interface (AWS CLI), installed and configured
- An AWS Identity and Access Management (IAM) user or role with permissions to put objects into an S3 bucket

Limitations

Virtual private cloud (VPC) limitations apply to AWS PrivateLink for Amazon S3. For more information, see Interface endpoint properties and limitations and AWS PrivateLink quotas (AWS PrivateLink documentation).

AWS PrivateLink for Amazon S3 doesn't support the following:

- Federal Information Processing Standard (FIPS) endpoints
- Website endpoints
- Legacy global endpoints

Architecture

Source technology stack
• Hadoop cluster with DistCp installed

Target technology stack

• Amazon S3
• Amazon VPC

Target architecture

The diagram shows how the Hadoop administrator uses DistCp to copy data from an on-premises environment through a private network connection, such as AWS Direct Connect, to Amazon S3 through an Amazon S3 interface endpoint.

Tools

• **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

• **Amazon VPC** – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you’ve defined. This virtual network closely resembles a traditional network that you’d operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

• **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

• **Apache Hadoop DistCp** – DistCp (distributed copy) is a tool used for copying large inter-clusters and intra-clusters. DistCp uses Apache MapReduce for distribution, error handling and recovery, and reporting.
# Epics

## Migrate data to the AWS Cloud

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| **Create an endpoint for AWS PrivateLink for Amazon S3.** | 1. Sign in to the AWS Management Console and open the Amazon VPC console.  
2. On the navigation pane, choose **Endpoints**, and then choose **Create Endpoint**.  
3. For **Service category**, choose **AWS services**.  
4. In the search box, enter `s3`, and then press `Enter`.  
5. In the search results, choose the `com.amazonaws.<your-aws-region>.s3` service name where the value in the **Type** column is **Interface**.  
6. For **VPC**, choose your VPC. For **Subnets**, choose your subnets.  
7. For **Security group**, choose or create a security group that allows TCP 443.  
8. Add tags based on your requirements and then choose **Create endpoint**. | AWS administrator |
| **Verify the endpoints and find the DNS entries.** | 1. Open the Amazon VPC console, choose **Endpoints**, and then select the endpoint that you created earlier.  
2. On the **Details** tab, find the first DNS entry for **DNS names**. This is the Regional DNS entry. When you use this DNS name, requests alternate between DNS entries specific to Availability Zones.  
3. Choose the **Subnets** tab. You can find the address of the endpoint’s elastic network interface in each Availability Zone. | AWS administrator |
<p>| <strong>Check the firewall rules and routing configurations.</strong> | To confirm that your firewall rules are open and that your networking configuration is correctly set up, use Telnet to | Network administrator, AWS administrator |</p>
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>test the endpoint on port 443. For example:</td>
<td><img src="#" alt="Example telnet command" /></td>
<td><img src="#" alt="Example telnet command" /></td>
</tr>
</tbody>
</table>

Note: If you use the Regional entry, a successful test shows that the DNS is alternating between the two IP addresses that you can see on the Subnets tab for your selected endpoint in the Amazon VPC console.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the name resolution.</td>
<td>You must configure the name resolution to allow Hadoop to access the Amazon S3 interface endpoint. You can’t use the endpoint name itself. Instead, you must resolve &lt;your-bucket-name&gt;.s3.&lt;your-aws-region&gt;.amazonaws.com or *.s3.&lt;your-aws-region&gt;.amazonaws.com.  For more information on this naming limitation, see Introducing the Hadoop S3A client (Hadoop website). Choose one of the following configuration options:</td>
<td>AWS administrator</td>
</tr>
<tr>
<td></td>
<td>• Use on-premises DNS to resolve the private IP address of the endpoint. You can override behavior for all buckets or selected buckets. For more information, see “Option 2: Access Amazon S3 using Domain Name System Response Policy Zones (DNS RPZ)” in Secure hybrid access to Amazon S3 using AWS PrivateLink (AWS blog post).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Configure on-premises DNS to conditionally forward traffic to the resolver inbound endpoints in the VPC. Traffic is forwarded to Route 53. For more information, see “Option 3: Forwarding DNS requests from on premises using Amazon Route 53 Resolver Inbound Endpoints” in Secure hybrid access to Amazon S3 using AWS PrivateLink (AWS blog post).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Edit the /etc/hosts file on all the nodes in your Hadoop cluster. This is a temporary solution for testing and isn’t recommended for production. To edit the /etc/hosts file, add an entry for either &lt;your-bucket-name&gt;.s3.&lt;your-aws-region&gt;.amazonaws.com or s3.&lt;your-aws-region&gt;.amazonaws.com.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>region&gt;.amazonaws.com. The <code>/etc/hosts</code> file can't have multiple IP addresses for an entry. You must choose a single IP address from one of the Availability Zones, which then becomes a single point of failure.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| Configure authentication for Amazon S3. | To authenticate to Amazon S3 through Hadoop, we recommend that you export temporary role credentials to the Hadoop environment. For more information, see [Authenticating with S3](Hadoop website). For long-running jobs, you can create a user and assign a policy that has permissions to put data into an S3 bucket only. The access key and secret key can be stored on Hadoop, accessible only to the DistCp job itself and to the Hadoop administrator. For more information on storing secrets, see [Storing secrets with Hadoop Credential Providers](Hadoop website). For more information on other authentication methods, see [How to get credentials of an IAM role for use with CLI access to an AWS account](AWS Single Sign-On documentation). To use temporary credentials, add the temporary credentials to your credentials file, or run the following commands to export the credentials to your environment:  
```bash
export AWS_SESSION_TOKEN=SECRET-SESSION-TOKEN
export AWS_ACCESS_KEY_ID=SESSION-ACCESS-KEY
export AWS_SECRET_ACCESS_KEY=SESSION-SECRET-KEY
```
If you have a traditional access key and secret key combination, run the following commands:  
```bash
export AWS_ACCESS_KEY_ID=my.aws.key
export AWS_SECRET_ACCESS_KEY=my.secret.key
```
**Note:** If you use an access key and secret key | AWS administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>combination, then change the credentials provider in the DistCp commands from &quot;org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider&quot; to &quot;org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider&quot;.</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Transfer data by using DistCp.</td>
<td>To use DistCp to transfer data, run the following commands:</td>
<td>Migration engineer, AWS administrator</td>
</tr>
<tr>
<td></td>
<td>```</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hadoop distcp -</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dfs.s3a.aws.credentials.provider=</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dfs.s3a.access.key=&quot;${AWS_ACCESS_KEY_ID}&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dfs.s3a.secret.key=&quot;${AWS_SECRET_ACCESS_KEY}&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dfs.s3a.session.token=&quot;${AWS_SESSION_TOKEN}&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dfs.s3a.path.style.access=true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dfs.s3a.connection.ssl.enabled=true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Dfs.s3a.endpoint=s3.&lt;your-aws-region&gt;.amazonaws.com</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hdfs:///user/root/ s3a://&lt;your-bucket-name&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The AWS Region of the endpoint isn't automatically discovered when you use the DistCp command with AWS PrivateLink for Amazon S3. Hadoop 3.3.2 and later versions resolve this issue by enabling the option to explicitly set the AWS Region of the S3 bucket. For more information, see <a href="https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-web.StartupGuide.html">S3A to add option fs.s3a.endpoint.region to set AWS region</a> (Hadoop website). For more information on additional S3A providers, see <a href="https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-web.StartupGuide.html">General S3A Client configuration</a> (Hadoop website). For example, if you use encryption, you can add the following option to the series of commands above depending on your type of encryption:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Dfs.s3a.server-side-encryption-algorithm=AES-256 [or SSE-C or SSE-KMS]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>```</td>
<td></td>
</tr>
</tbody>
</table>
Perform cross-Region failover and failback by using CloudEndure Disaster Recovery

*Created by Anuraag Deekonda (AWS)*

### Summary

This pattern provides guidance for setting up disaster recovery (DR) from one Amazon Web Services (AWS) Region to another by using CloudEndure Disaster Recovery.

A cross-Region DR strategy helps ensure that your business-critical data remains online and available during a natural disaster or technical failure in a Region. It also helps prevent latency issues that might arise if your users access data from different, geographically distant Regions.

CloudEndure Disaster Recovery helps protect critical databases, including Oracle, MySQL, and Microsoft SQL Server, as well as enterprise applications such as SAP. It supports automated failover to spin up target machines, continuous replication to a staging area, and automated failback to restore your source environment with minimal disruption.

This pattern covers an AWS-to-AWS DR strategy. If you have an on-premises database or application, you must use the CloudEndure Failback Client. For more information, see the pattern Use CloudEndure for disaster recovery of an on-premises database.
AWS Prescriptive Guidance Patterns
Prerequisites and limitations

Note: AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An active CloudEndure Disaster Recovery account
- One or more databases hosted on Amazon Elastic Compute Cloud (Amazon EC2) instances in the AWS Cloud

Limitations

- The workloads (databases and applications) must be hosted on EC2 instances. This pattern doesn’t support databases that are hosted on Amazon Relational Database Service (Amazon RDS) DB instances.

Architecture

AWS-to-AWS DR architecture

In the following DR architecture, CloudEndure Disaster Recovery continuously replicates data from source EC2 instances (source machines) to a staging area in another AWS Region. The staging area hosts the data on lightweight EC2 Linux instances. When an IT disruption causes a failover, these staged resources are deployed to EC2 instances in the target Region. When you’re ready to restore data and resume standard operations, you initiate a fallback to the source machines in the first Region.

Tools

- CloudEndure Disaster Recovery – CloudEndure Disaster Recovery continuously replicates your machines (including operating system, system state configuration, databases, applications, and files) into a low-cost staging area in your target AWS account and preferred AWS Region. In the case of a
disaster, you can instruct CloudEndure Disaster Recovery to automatically launch thousands of your machines in their fully provisioned state in minutes.

Epics

Set up a CloudEndure Disaster Recovery account

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to CloudEndure Disaster Recovery</td>
<td>In AWS Marketplace, subscribe to CloudEndure Disaster Recovery to AWS.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Set up an account.</td>
<td>Register and create a CloudEndure Disaster Recovery account. When you receive an email message, confirm the account request to complete registration. For more information, see Registering to CloudEndure Disaster Recovery in the CloudEndure documentation.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Create a CloudEndure project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure user console.</td>
<td>Navigate to <a href="https://console.cloudendure.com">https://console.cloudendure.com</a> and sign in with the credentials you created in the previous step.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Create a new project.</td>
<td>Choose the plus (+) button in the upper-left corner of the console to create a project.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Enter a project name, and select a project type and license package.</td>
<td>For project type, select Disaster Recovery. You can acquire a license through AWS Marketplace. For more information about creating a project, see Working with Projects in the CloudEndure documentation.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

Prepare your CloudEndure environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate and use AWS credentials.</td>
<td>AWS Identity and Access Management (IAM) credentials</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>are required for configuring the CloudEndure User Console. For instructions, see the CloudEndure migration guide.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Configure replication settings.</td>
<td>CloudEndure uses replication servers to provide a lightweight replication staging area on AWS. For instructions, see the Defining Replication Settings for AWS in the CloudEndure documentation. For Disaster Recovery Source and Disaster Recovery Target, choose your source and target Regions.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Install CloudEndure Agent on your source machine.</td>
<td>The CloudEndure Agent replicates your applications and data in the staging area on AWS. For installation instructions, see the CloudEndure migration guide.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Configure the target system Blueprint.</td>
<td>A Blueprint provides instructions for creating and launching a target machine for a specified source machine. For instructions, see the Configuring the Target Machine Blueprint in the CloudEndure documentation.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

Test your disaster recovery solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test the CloudEndure configuration.</td>
<td>Use the Test Mode workflow to launch and run a target machine in the target infrastructure. For instructions, see the CloudEndure migration guide.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Test your target instance.</td>
<td>To test your target machines: 1. Choose each machine's name 2. Open the Target tab, and copy the public IP. 3. Log in to the newly launched Amazon EC2 server.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>
## Perform a failover by using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Check failover settings.          | The failover process switches operations (the direction of data replication) from your source machines to target machines. For detailed information about this process, see the CloudEndure documentation. Before you start the failover, open the Machines pane from the CloudEndure User Console, and verify that the source machine you want to test for failover has the following status indications under each column:  
  - **Data Replication Progress** – Continuous Data Protection  
  - **Status** – Rocket icon, which indicates that the target machine can be launched  
  - **Disaster Recovery Lifecycle** – Tested Recently | CloudEndure administrator               |
| Start the failover.               | 1. Choose your source machine.  
2. On the Launch Target Instance menu, choose Recovery Mode.  
3. When prompted, choose Next. | CloudEndure administrator               |
| Choose the recovery point for the target machine. | Choose the recovery point to use when launching new target machines for the failover, and then choose Continue with Launch. You can choose the latest or a previous recovery point from the list. | CloudEndure administrator               |
| Monitor job progress.             | The Job Progress window provides details about the target machine launch process. When failover is complete, the status in the Disaster Recovery Lifecycle column will change to Failed over. | CloudEndure administrator               |
### Perform a failback by using CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare for failback.</td>
<td>After you perform a successful failover, you will want to fail your machines back to restore operations to their original state. CloudEndure lets you prepare for failback by reversing the direction of data replication from the target machine back to the source machine. CloudEndure then treats the currently launched target machines as source machines, and data flows from your selected target infrastructure back to your original source infrastructure. For detailed information about this process, see the CloudEndure documentation.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td></td>
<td>1. On the Project Actions menu, choose Prepare for Failback.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. At the prompt, choose Continue.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This step sets up the failback operation on all machines that have been added to the project.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The machines will undergo the initiation process, and the status in the Data Replication Progress column will change to Continuous Data Protection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information about the steps in this epic, see Performing a Failback with CloudEndure in the CloudEndure documentation.</td>
<td></td>
</tr>
<tr>
<td>Configure failback settings.</td>
<td>1. Choose a machine, and then choose the Failback Settings tab.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td></td>
<td>2. Edit machine replication settings.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Choose Save Failback Settings.</td>
<td></td>
</tr>
<tr>
<td>Launch target machines for your failed back machines.</td>
<td>1. Choose a machine.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>
### AWS Prescriptive Guidance Patterns
#### Epics

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| 2. On the Launch Target Instance menu, choose Recovery Mode, and then choose Next.  
3. Choose the recovery point to use when launching new target machines for the failback, and then choose Continue with Launch. You can choose the latest or a previous recovery point from the list. | | |
| Return to normal operations.  
1. When the target machines have been launched, choose Return to Normal Operation from the Project Actions menu. This reverses the direction of data replication back to its normal state (from the original source machine to the original target machine).  
2. At the prompt, choose Continue. Your machines will undergo the initiation sequence again.  
When the status in the Data Replication Progress column changes to Continuous Data Protection, failback is complete. | CloudEndure administrator | |

(Optional) Perform a failover or failback on specific machines

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the source machines.</td>
<td>In certain cases, you might want to perform a failback for a single machine (or specific machines) in a project. This can be achieved by moving the selected machines into a separate project and performing the failover or failback as discussed in the previous epics. When a source machine is moved to another project, the replication of its data continues without major interruptions (without rescanning or synchronization, except for a small backlog). The Blueprint of the machine is maintained without change. For</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>
### Task

<table>
<thead>
<tr>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>detailed information about this process, see the <a href="#">CloudEndure documentation</a>.</td>
<td></td>
</tr>
</tbody>
</table>
| 1. From the project list, choose the project that includes the source machine you want to move.  
2. On the **Machines** page, select the source machines.  
3. From the **Machine Actions** menu, choose **Move x Machines to Another Project**. |  |

| Move the machines to another project. | In the **Move machine to another Project** dialog box, from the **Move** list, choose the destination project, and then choose **Move Machines**.  
The destination project configuration must match the source project configuration. The projects must have the same:  
• CloudEndure account  
• Cloud account (both source and target infrastructure)  
• Solution type (Migration or Disaster Recovery)  
• Source and target infrastructure and Regions  
• License package | CloudEndure administrator |

| Fail over or fail back the moved machines. | When the machines have been moved to the new project, you can confirm that they appear on the **Machines** page of the destination project and perform the failover or failback action on the machines. | CloudEndure administrator |

## Related resources

**AWS Marketplace**
- [CloudEndure Migration](#)  
- [CloudEndure Disaster Recovery to AWS](#)

**CloudEndure documentation**
- [Signing in to the console](#)
• Creating a project
• Generating and using credentials
• Configuring replication settings
• Installing Agents
• Performing disaster recovery failover and fallback

**AWS Prescriptive Guidance publications**

• Migrating workloads to the AWS Cloud by using CloudEndure Migration
• Automating large-scale server migrations with CloudEndure Migration Factory
• Use CloudEndure for disaster recovery of an on-premises database

**Tutorials and videos**

• CloudEndure Troubleshooting Playbook
• CloudEndure Video Guide
• CloudEndure Training
• Disaster Recovery Demo (video)
• AWS to AWS Failover and Failback (video)

---

**Use CloudEndure for disaster recovery of an on-premises database**

*Created by Anuraag Deekonda (AWS)*

<table>
<thead>
<tr>
<th>Created by:</th>
<th>AWS</th>
<th>Environment:</th>
<th>PoC or pilot</th>
<th>Technologies:</th>
<th>Modernization; Databases; Storage &amp; backup</th>
</tr>
</thead>
</table>

**Summary**

This pattern uses CloudEndure Disaster Recovery and the CloudEndure Failback Client for disaster recovery (DR). It sets up DR for an on-premises data center host, using an Amazon Elastic Compute Cloud (Amazon EC2) instance.

You must use the CloudEndure Failback Client for replicating from a non-cloud or other cloud infrastructure to the Amazon Web Services (AWS) Cloud. After your disaster event is over, you will want to fail back your machines. CloudEndure prepares you for failback by reversing the direction of data replication from the target machine back to the source machine. The CloudEndure User Console treats the currently launched target machines as source machines. Replication is reversed from your selected target machines back to your original source infrastructure.

**Note:** AWS Application Migration Service (MGN) is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud. Customers who currently use CloudEndure Migration or AWS Server Migration Service (AWS SMS) are encouraged to switch to MGN for future migrations.

**Prerequisites and limitations**

**Prerequisites**

---

1834
• An active AWS account
• An on-premises database

Architecture

Source technology stack
• A database in an on-premises data center

Target technology stack
• A database on an EC2 instance (for a complete list of supported operating system versions, see Amazon EC2 FAQs)

Source and target network architecture

Tools
• CloudEndure Disaster Recovery – CloudEndure Disaster Recovery reduces downtime and data loss by providing fast, reliable recovery of physical, virtual, and cloud-based servers into AWS. CloudEndure Disaster Recovery continuously replicates your machines (including operating system, system state configuration, databases, applications, and files) into a low-cost staging area in your target AWS account and preferred Region. If there is a disaster, you can instruct CloudEndure Disaster Recovery to automatically launch thousands of machines in their fully provisioned state in minutes.
Epics

Subscribe to CloudEndure Disaster Recovery

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribe to CloudEndure Disaster Recovery.</td>
<td>CloudEndure Disaster Recovery is available in the AWS Marketplace.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Create a CloudEndure account.</td>
<td>Register for CloudEndure and create an account. Then, in email, confirm the subscription.</td>
<td>General AWS</td>
</tr>
<tr>
<td>Set the account password and accept terms and conditions.</td>
<td>Passwords must be at least eight characters long and must contain at least one uppercase letter, one lowercase letter, one digit, and one special character.</td>
<td>General AWS</td>
</tr>
</tbody>
</table>

Create a CloudEndure project

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign in to the CloudEndure User Console.</td>
<td>On the CloudEndure User Console, sign in with the credentials you created in the previous step.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Create a new project.</td>
<td>In the upper-left corner of the console, choose the plus (+) button to create a project. Select Disaster Recovery as the project type. You can acquire a license through AWS Marketplace.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

Generate and use AWS credentials

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an IAM policy for the CloudEndure solution.</td>
<td>The AWS Identity and Access Management (IAM) policy that you must create for running CloudEndure solution is based on a predefined CloudEndure policy. This CloudEndure policy contains the necessary permissions for using AWS as your target infrastructure.</td>
<td>AWS systems administrator</td>
</tr>
<tr>
<td>Create a new IAM user and generate AWS credentials.</td>
<td>To generate the required AWS credentials for the CloudEndure User Console, create at least</td>
<td>AWS systems administrator</td>
</tr>
</tbody>
</table>
### Configure replication settings

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the replication servers.</td>
<td>For more information, see the CloudEndure documentation.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

### Installing CloudEndure Agents on your source machine

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locate your Agent installation token.</td>
<td>On the CloudEndure User Console, navigate to Machines, Machine Actions, Add Machines. When you run the installer file on a source machine, you are first asked to enter your installation token. The token is a unique string of characters that is automatically generated for you when your CloudEndure account is activated. You can use one installation token to install the Agent on as many source machines as your project allows.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

On Linux machines, run the installer.

For Linux machines, copy the installer command, log in to your source machines, and run the installer.

For detailed instructions, see the CloudEndure documentation.

On Windows machines, run the installer.

For Windows machines, download the installer file to
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>each machine, and then run the installer command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For detailed instructions, see the CloudEndure documentation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replicate the data.</td>
<td>After the Agent is installed, CloudEndure starts to replicate the source machine starts to the staging area. When the initial sync is completed, the machine appears on the Machines tab on the CloudEndure User Console.</td>
<td></td>
</tr>
<tr>
<td>CloudEndure administrator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Configure the target machine's Blueprint

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose the source machine for the Blueprint.</td>
<td>On the CloudEndure User Console, on the Machines tab, choose the source machine to access the Machine Details pane.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Configure the Blueprint for the target machine.</td>
<td>On the Blueprint tab, configure the settings for your target machine based on your requirements. For detailed instructions, see the CloudEndure documentation.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>

### Test your DR solution

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Test Mode to test the solution.</td>
<td>For detailed instructions on Test Mode and test cutover verification, see the CloudEndure documentation.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Test your target instance launched on the Amazon EC2 server.</td>
<td>To test each of your target machines, choose the machine's name. Then open the Target tab, copy the new IP address, and log in to the newly launched server on the Amazon EC2 instance.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>
## Perform a failover with CloudEndure

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Verify source machine status.     | On the CloudEndure User Console **Machines** page, verify that the source machine you want to fail over has the following status indications:  
  - **Data Replication Progress** – Continuous Data Protection  
  - **Status** – Rocket icon, which indicates that the target machine can be launched  
  - **Disaster Recovery Lifecycle** – Tested Recently                                                                                           | CloudEndure administrator             |
| Start the cutover.                | 1. On the **Machines** page, choose your source machine.  
  2. On the **Launch Target Machines** tab, choose **Recovery Mode**.  
  3. Choose the recovery point for the target machine. The system will use the recovery point when launching the new target machines for the failover. You can use the latest recovery point or choose a previous recovery point from the list.  
  4. Choose **Continue with Launch**.                                                                                                         | CloudEndure administrator             |
| Check the job progress and completion status. | The **Job Progress** window displays details for the target machine launch process.  
  After the failover is complete, the **Disaster Recovery Lifecycle** status on the CloudEndure User Console changes to **Failed over** to indicate successful completion. | CloudEndure administrator             |

## Perform a failback with the CloudEndure Failback Client

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review the CloudEndure Failback Client requirements.</td>
<td>Use the CloudEndure Failback Client for replicating from an on-premises or other cloud infrastructure to AWS. The</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| CloudEndure Failback Client has the following requirements:  
- Machines must be configured to boot in BIOS mode, supporting MBR boot.  
- Machines configured to boot in UEFI mode, supporting GPT boot only, are not supported.  
- The CloudEndure Failback Client requires at least 4 GB of dedicated RAM. | CloudEndure administrator |
| Prepare for failback. | Before you can initiate the Prepare for Failback action, all source machines must have launched target machines in either Test Mode or Recovery Mode.  
On the Project Actions menu, choose Prepare for Failback, and then choose Continue.  
When Pair the CloudEndure Agent with the Failback Client is displayed, the machines are ready for failback. | CloudEndure administrator |
| Download the CloudEndure Failback Client in your on-premises environment. | To download the CloudEndure Failback Client into your source environment, do the following:  
1. In your DR project, choose Setup & Info.  
2. On the Replication Settings page, choose the Learn about failing back to “Other Infrastructure” link.  
3. In the Failing Back to an Unidentified Cloud/Other Infrastructure dialog box, choose download from here.  
The file will automatically be downloaded. | CloudEndure administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate replication of the on-premises machine.</td>
<td>To initiate replication of source machine, the target machine must be booted into the CloudEndure Failback Client Image (failback_client.iso). If the client can't fetch the networking settings using the Dynamic Host Configuration Protocol (DHCP), enter the settings manually. The CloudEndure Failback Client connects to console.clouendure.com over TCP port 443, and authenticates using the CloudEndure credentials that you are prompted to enter.</td>
<td>CloudEndure administrator</td>
</tr>
</tbody>
</table>
| Follow the instructions to provide the necessary details. | Provide the following details:  
- Installation token  
- Machine ID of the source machine  
- Disk mapping between source and target  

Make sure that the CloudEndure Failback Client has connectivity to the CloudEndure User Console and the target machine through public or private IP addresses. | CloudEndure administrator |
| Locate the source machine ID. | To locate the source machine ID, choose the machine name on the Machines tab, and copy the ID from the Source tab. | CloudEndure administrator |
| Connect the source machine to the target machine. | Provide the source machine ID (the server on AWS is now the source for the failback) in the on-premises server (target machine). The AWS machine (source) connects to the on-premises server (target) on TCP port 1500 to start the replication. 

After the initial replication is complete, the CloudEndure User Console indicates that replication is in **Continuous Data Protection** mode. | CloudEndure administrator |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit the failback settings, if necessary.</td>
<td>To edit the failback settings, choose the machine name, and then choose the <strong>Failback Settings</strong> tab.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td>Launch the target machine.</td>
<td>To launch the target machine, do the following:</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td></td>
<td>Select the check box to the left of each machine name, and choose <strong>Launch x Target Machine</strong>, and then choose <strong>Recovery Mode</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>In the dialog box, choose <strong>Next</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Choose the <strong>Latest</strong> recovery point, and then choose <strong>Continue with Launch</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After the launch process is complete, the CloudEndure User Console displays the status <strong>Pair the CloudEndure Agent with the Replication Server</strong> under <strong>Data Replication Progress</strong>.</td>
<td></td>
</tr>
<tr>
<td>Return the machines to normal operation.</td>
<td>Now change the direction of data replication so that the on-premises machine is the source and the AWS machine is the target. Choose Project Actions, and then choose <strong>Return to Normal</strong> and <strong>Continue</strong>.</td>
<td>CloudEndure administrator</td>
</tr>
<tr>
<td></td>
<td>The direction of data replication is reversed, and the machines undergo the initial sync process. The failback process is complete when the <strong>Data Replication Progress</strong> column displays the <strong>Continuous Data Protection</strong> status for all machines.</td>
<td></td>
</tr>
</tbody>
</table>

**Related resources**

**AWS Marketplace**
- CloudEndure Disaster Recovery

**CloudEndure documentation**
- Signing in to the console
- Creating a project
More patterns

- Generating and using credentials
- Configuring replication settings
- Installing CloudEndure Agents
- Performing Disaster Recovery failover

Tutorials and videos

- CloudEndure troubleshooting playbook
- CloudEndure videos
- Disaster Recovery to AWS demo

More patterns

- Automate data loading from Amazon S3 to Amazon Redshift using AWS Data Pipeline (p. 9)
- Automate event-driven backups from CodeCommit to Amazon S3 using CodeBuild and CloudWatch Events (p. 374)
- Automatically archive items to Amazon S3 using DynamoDB TTL (p. 1334)
- Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue (p. 12)
- Convert and unpack data from EBCDIC to ASCII (p. 1398)
- Copy data from an S3 bucket in one account and Region to another account and Region (p. 100)
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS (p. 166)
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Kinesis Data Firehose with AWS CDK (p. 1747)
- Load data from Amazon S3 to Amazon Redshift using AWS Glue (p. 39)
- Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP (p. 1017)
- Migrate data from Microsoft Azure Blob storage and Backblaze B2 Cloud Storage to Amazon S3 by using Rclone (p. 1159)
- Migrate small sets of data from on premises to Amazon S3 using AWS SFTP (p. 1023)
- Monitor Amazon Aurora for instances without encryption (p. 353)
- Move mainframe files directly to Amazon S3 using Transfer Family (p. 779)
- Run stateful workloads with persistent data storage by using Amazon EFS on Amazon EKS with AWS Fargate (p. 265)
- Successfully import an S3 bucket as an AWS CloudFormation stack (p. 103)
Deploy a React-based single-page application to Amazon S3 and CloudFront

Created by Jean-Baptiste Guillois (AWS)

<table>
<thead>
<tr>
<th>Code repository:</th>
<th>Environment: Production</th>
<th>Technologies: Websites &amp; web apps; Cloud-native; Serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td>• React-based CORS single-page application</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Workload: All other workloads | AWS services: Amazon CloudFront; Amazon S3; Amazon API Gateway |

Summary

A single-page application (SPA) is a web application implementation that loads a web document and updates it by using JavaScript APIs. Your customers can then use your website without loading entire pages from the server, which helps improve your website's performance and provides a more dynamic user experience.

This pattern provides a step-by-step approach and code to host a single-page application (SPA) written in React on Amazon Simple Storage Service (Amazon S3) and Amazon CloudFront. The pattern's sample SPA uses a REST API exposed by Amazon API Gateway to demonstrate cross-origin resource sharing (CORS) best practices.

Prerequisites and limitations

Prerequisites

• An active AWS account.
• An integrated development environment (IDE), such as AWS Cloud9.
• Node.js and npm, installed and configured. For more information, see the Downloads section of the Node.js documentation.
• Yarn, installed and configured. For more information, see the Yarn documentation.
Architecture

Technology stack

- Amazon API Gateway
- Amazon CloudFront
- Amazon Route 53
- Amazon Simple Storage Service (Amazon S3)
- Amazon Identity and Access Management (IAM)
- Amazon CloudWatch
- AWS CloudTrail
- AWS CloudFormation

Tools

- **AWS Cloud9** – AWS Cloud9 is an IDE that offers a rich code-editing experience with support for several programming languages and runtime debuggers, and a built-in terminal. AWS Cloud9 contains a collection of tools that you use to code, build, run, test, and debug software, and helps you release software to the cloud.

- **AWS CloudFormation** – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- **Amazon CloudFront** – Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations for lower latency and improved performance.

- **Amazon S3** – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
• **Amazon Route 53** – Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service that you can use to perform three main functions in any combination: domain registration, DNS routing, and health checking.

• **IAM** – AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.

• **AWS CloudTrail** – AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account.

• **Amazon CloudWatch** – Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables that you can measure for your resources and applications.

• **Amazon API Gateway** – Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.

**Code**

This pattern's sample application code is available in the GitHub [React-based CORS single-page application repository](https://github.com/aws-samples/react-cors-spa).

## Epics

**Locally build and deploy your application**

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone the repository.</td>
<td>We recommend using AWS Cloud9 as the IDE for this pattern, but you can also use another IDE (for example, Visual Studio Code or IntelliJ IDEA). Run the following command to clone the sample application's repository into your IDE:</td>
<td>App developer, AWS DevOps</td>
</tr>
</tbody>
</table>
| Locally deploy the application. | 1. In the project directory, run the `npm install` command to initiate the application dependencies.  
2. Run the `yarn start` command to start the application locally. | App developer, AWS DevOps            |
| Locally access the application. | Open a browser window and enter the `http://localhost:3000` URL to access the application. | App developer, AWS DevOps            |
## Deploy the SPA

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Deploy the AWS CloudFormation template. | 1. Sign in to the AWS Management Console, and then open the AWS CloudFormation console.  
2. Choose Create Stack, and then choose With new resources (standard).  
3. Choose Upload a template file.  
4. Choose Choose file, choose the react-cors-spa-stack.yaml file from the cloned repository, and then choose Next.  
5. Enter a name for your stack, and then choose Next.  
6. Keep all default options, and then choose Next.  
7. Review the final settings for your stack, and then choose Create stack. | App developer, AWS DevOps |
| Customize your application source files. | 1. After your stack is deployed, open the Output tab and identify the APIEndpoint URL and Bucket name.  
2. Copy the API endpoint URL.  
3. Navigate to <project_root>/src/App.js, and then paste the URL into the APIEndPoint variable value on line 26 of the App.js file. | App developer |
| Build the application package. | In your project directory, run the yarn build command to build the application package. | App developer |
| Deploy the application package. | 1. Open the Amazon S3 console.  
2. Identify and choose the S3 bucket that you created earlier.  
3. Choose Upload, and then choose Add files.  
4. Choose the content of your build folder.  
5. Choose Add folder, and then choose the static directory. **Important:** Don't choose | App developer, AWS DevOps |
## Embed an Amazon QuickSight dashboard in a local Angular application

**Task** | **Description** | **Skills required**
--- | --- | ---
| | the contents; choose the directory. 6. Choose *Upload* to upload the files and directory to your S3 bucket. | |

### Test the application

**Task** | **Description** | **Skills required**
--- | --- | ---
| Access and test the application. | 1. Open the Amazon CloudFront console, and then choose *Distributions*. 2. Choose the relevant distribution ID, choose the *Distribution domain name*, and then copy the URL. 3. Open a browser window, and then paste the URL to access the application. | App developer, AWS DevOps |

### Clean up the resources

**Task** | **Description** | **Skills required**
--- | --- | ---
| Delete the S3 bucket contents. | 1. Open the Amazon S3 console and choose the bucket that you created earlier. 2. Choose *Empty* to delete the bucket’s contents. | AWS DevOps, App developer |
| Delete the AWS CloudFormation stack. | 1. Open the AWS CloudFormation console and choose the stack that you created earlier. 2. Choose *Delete* to delete the stack and all related resources. | AWS DevOps, App developer |

*Created by Sean Griffin (AWS)*
Summary

This pattern provides guidance for embedding an Amazon QuickSight dashboard into a locally hosted Angular application for development or testing. The embedded analytics feature in Amazon QuickSight doesn't support this functionality natively. It requires an Amazon QuickSight account with an existing dashboard and knowledge of Angular.

When you work with embedded Amazon QuickSight dashboards, you would typically have to host your application on a web server to view the dashboard. This makes development more difficult, because you have to continuously push your changes to the web server to make sure everything is behaving correctly. This pattern shows how to run a locally hosted server and use Amazon QuickSight embedded analytics to make the development process easier and more streamlined.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- An active QuickSight account with session capacity pricing
- QuickSight Embedding SDK installed
- Angular CLI installed
- Familiarity with Angular
- mkcert installed

Limitations

- This pattern provides guidance on embedding a QuickSight dashboard by using the ANONYMOUS (publicly accessible) authentication type. If you are using AWS Identity and Access Management (IAM) or QuickSight authentication with your embedded dashboards, the provided code won't apply. However, the steps for hosting the Angular application in the Epics section are still valid.

Architecture

Technology stack

- Angular frontend
- AWS Lambda and Amazon API Gateway backend

Architecture

In this architecture, the HTTP APIs in API Gateway enable the local Angular application to call the Lambda function. The Lambda function returns the URL for embedding the Quicksight dashboard.
Automation and scale
You can automate the backend deployment by using AWS CloudFormation or AWS Serverless Application Model (AWS SAM).

Tools

- **Angular CLI** – The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.
- **QuickSight Embedding SDK** – You use this SDK to embed QuickSight dashboards into your HTML.
- **mkcert** – mkcert is a simple tool for creating locally trusted development certificates. It requires no configuration. mkcert is required because QuickSight allows only HTTPS requests for embedding dashboards.

AWS services

- **Amazon API Gateway** – Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- **AWS Lambda** – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- **Amazon QuickSight** – Amazon QuickSight is a business analytics service for building visualizations, performing ad hoc analyses, and getting business insights from your data.

Code
See the *Attachments* section to get the Lambda code for this pattern.

Epics

Generate EmbedURL

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an EmbedUrl policy.</td>
<td>Create an IAM policy named QuicksightGetDashboardEmbedUrl that has the following properties:</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Version&quot;: &quot;2012-10-17&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Statement&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Effect&quot;: &quot;Allow&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Action&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;quicksight:GetDashboardEmbedUrl&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;quicksight:GetAnonymousUserEmbedUrl&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Resource&quot;: &quot;*&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
</tbody>
</table>
| Create the Lambda function. | 1. On the Lambda console, open the Functions page.  
2. Choose Create Function.  
3. Under Basic information, do the following:  
   a. Choose Author from Scratch.  
   b. For Function name, enter get-qs-embed-url.  
   c. For Runtime, choose Python 3.9.  
4. Choose Create Function.  
5. Copy the code from the attached get-qs-embed-url.py file into the Lambda function Code source tab.  
6. Deploy your function.  
7. Add DASHBOARD_ID as an environment variable to your Lambda function:  
   a. On the Configuration tab, choose Environment variables, Edit, Add environment variable.  
   b. Add an environment variable with the key DASHBOARD_ID.  
   c. To get the value of your DASHBOARD_ID, navigate to your dashboard in QuickSight and copy the UUID at the end of the URL in your browser. For example, if the URL is https://us-east-1.quicksight.aws.amazon.com/snip/dashboards/<dashboard-id>, specify the <dashboard-id> part of the URL as the key value.  
   d. Choose Save.  
8. Modify the execution role of the Lambda function and add the QuicksightGetDashboardEmbedUrl policy to it.  
   a. On the Configuration tab, choose Permissions, and then choose your role name.                                                                 | App developer             |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>b.</td>
<td>Choose Attach policies, search for QuicksightGetDashboardEmbedUrl, select its checkbox, and then choose Attach policy.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Create and run a test event. You can use the &quot;Hello World&quot; template, because the function won't use any of the data in the test event.</td>
<td></td>
</tr>
<tr>
<td>a.</td>
<td>Choose the Test tab.</td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>Give your test event a name, and then choose Save changes.</td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>To test your Lambda function, choose Test. The response should look similar to this:</td>
<td></td>
</tr>
</tbody>
</table>

```json
{
  "statusCode": 200,
  "body": "https://us-east-1.quicksight.aws.amazon.com/embed/f1acc078687783b9a4543a05ba929b3a/dashboards/8a123480-8d77-4f71-92a0-0e736968b99b1?code=AYABeNod..."
}
```
### Create an API in API Gateway.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
</table>
| Create an API in API Gateway. | 1. On the API Gateway console, choose **Create API**, and then choose **HTTP API, Build**.  
   a. Choose **Add integration**, and then choose the Lambda function you just created.  
   b. For API name, enter **qs-embed-api**.  
   c. Choose **Next**, and then change **ANY** to **GET** for the `/get-qs-embed-url` route.  
   d. Choose **Next**, **Next**, and then **Create**.  
2. Navigate to the **CORS** section of your API and choose **Configure**. Configure the following:  
   • **Access Control Allow Origin**: https://my-qs-app.net:4200  
   • **Access Control Allow Headers**: *  
   • **Access Control Allow Methods**: *  
   • **Access Control Allow Credentials**: YES  
3. From the upper-left corner of the screen, navigate to the API **qs-embed-api**. Note the **Invoke URL** of the **$default** stage under **Stages for qs-embed-api**.  

**Note:** my-qs-app.net can be any domain. If you want to use a different domain name, make sure to update the **Allow Origin** information in the second step, and change my-qs-app.net in the following steps. | App developer |

### Create the Angular application

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the application with the Angular CLI.</td>
<td><strong>Note:</strong> The following commands are for Unix or MacOS machines.</td>
<td>App developer</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>Skills required</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1.</td>
<td>To create the application, use the following commands:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ng new quicksight-app --defaults</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cd quicksight-app/src/app</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ng g c dashboard</td>
<td></td>
</tr>
<tr>
<td></td>
<td>npm i amazon-quicksight-embedding-sdk</td>
<td></td>
</tr>
<tr>
<td></td>
<td>echo &quot;declare module 'amazon-quicksight-embedding-sdk';&quot; &gt;&gt; types.d.ts</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Navigate to your src/environments/environment.ts file, and add apiUrl and the URL of your API Gateway to it:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>export environment =</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{ production: false,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>apiUrl: &quot;&lt;Invoke URL from previous steps&gt;&quot; }</td>
<td></td>
</tr>
</tbody>
</table>
### Task
Add code to your dashboard.component.ts file.

### Description
Add the following code to the src/app/dashboard/dashboard.component.ts file:

```typescript
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import * as QuicksightEmbedding from 'amazon-quicksight-embedding-sdk';
import { environment } from '../../environments/environment';

@Component({
  selector: 'app-dashboard',
  templateUrl: './dashboard.component.html',
  styleUrls: ['./dashboard.component.css']
})
export class DashboardComponent implements OnInit {
  constructor(private http: HttpClient) {
  }
  dashboard: any;
  ngOnInit() {
    this.GetDashboardURL();
  }
  GetDashboardURL() {
    const url = await this.http.get(environment.apiUrl + 'get-qs-embed-url').toPromise()
    this.Dashboard(url);
  }
  Dashboard(embeddedURL: any) {
    var containerDiv = document.getElementById("dashboardContainer");
    var options = {
      url: embeddedURL,
      container: containerDiv,
      scrolling: "yes",
      height: "850px",
      width: "100%"
    };
    this.dashboard = QuicksightEmbedding.embedDashboard(options);
  }
}
```

### Skills required
App developer
### Task: Add code to your dashboard.component.html file.

**Description:** In the `src/app/dashboard/dashboard.component.html` file, insert the following `<div>` tag where you want the dashboard to be embedded:

```html
<div id="dashboardContainer"></div>
```

This code uses the ID to determine which HTML element the dashboard should be associated with.

**Skills required:** App developer

### Task: Modify your app.component.html file to load your dashboard component.

**Description:**
1. Delete all the existing code in the `src/app/app.component.html` file.
2. Add the following:

```html
<app-dashboard></app-dashboard>
```

**Skills required:** App developer

### Task: Import HttpClientModule into your app.module.ts file.

**Description:**
1. At the top of the `src/app/app.module.ts` file, add:

```typescript
import { HttpClientModule } from '@angular/common/http';
```
2. Add `HttpClientModule` as an import after `BrowserModule`.

**Skills required:** App developer

---

## Host the Angular application

### Task: Configure mkcert.

**Description:**

**Note:** The following commands are for Unix or MacOS machines. If you're using Windows, see the Additional information (p. 1858) section for the equivalent `echo` command.

1. To create a local certificate authority (CA) on your machine:

   ```bash
   mkcert -install
   ```

**Skills required:** App developer
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Configure my-qs-app.net to always redirect to your local PC:</td>
<td>echo &quot;127.0.0.1 my-qs-app.net&quot;</td>
<td>AWS administrator</td>
</tr>
<tr>
<td>3. Make sure you're in the src/ directory of the Angular project:</td>
<td>mkcert my-qs-app.net 127.0.0.1</td>
<td></td>
</tr>
</tbody>
</table>
| Configure QuickSight to allow your domain. | 1. In QuickSight, choose your name in the upper-right corner, and then choose Manage Quicksight.  
2. Navigate to Domains and Embedding.  
3. Add https://my-qs-app.net:4200 as an allowed domain. | |
| Start a local Angular development server. | Run the following command: | App developer |
| | ng serve --host my-qs-app.net --port 4200 --ssl --ssl-key ./src/my-qs-app.net-key.pem --ssl-cert ./src/my-qs-app.net.pem -o | |
| | This enables Secure Sockets Layer (SSL) with the custom certificate you created earlier.  
When the build is complete, it opens a browser window and you can view your embedded Amazon QuickSight dashboard hosted locally in Angular. | |

### Additional information

If you're using Windows, run the Command Prompt window as an administrator, and configure my-qs-app.net to always redirect to your local PC by using the following command:

```
echo 127.0.0.1 my-qs-app.net >> %WINDIR%\System32\Drivers\Etc\Hosts
```
More patterns

- Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools (p. 1574)
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer (p. 128)
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer (p. 117)
- Analyze speech in real time using Amazon Transcribe and Amazon Comprehend (p. 636)
- Automate migration strategy identification and planning using AppScore (p. 747)
- Automate static website deployment to Amazon S3 (p. 379)
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9 (p. 403)
- Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm (p. 409)
- Create a pipeline and AMI using CodePipeline and HashiCorp Packer (p. 447)
- Create a pipeline and deploy artifact updates to on-premises EC2 instances using CodePipeline (p. 452)
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS (p. 166)
- Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer (p. 203)
- Deploy Java microservices on Amazon ECS using Amazon ECR and AWS Fargate (p. 174)
- Deploy Java microservices on Amazon ECS using Amazon ECR and load balancing (p. 179)
- Deploy Java microservices on Amazon ECS using AWS Fargate (p. 169)
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS (p. 1155)
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk (p. 1193)
- Migrate an on-premises Go web application to AWS Elastic Beanstalk by using the binary method (p. 1014)
- Migrate an on-premises Linux server to an Amazon EC2 Linux instance using AWS SMS (p. 991)
- Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP (p. 1017)
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 (p. 1182)
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling (p. 1188)
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk (p. 1026)
- Migrate on-premises Java applications to AWS using AWS App2Container (p. 1278)
- Migrate OpenText TeamSite workloads to the AWS Cloud (p. 1137)
- Migrate Windows SSL certificates to an Application Load Balancer using ACM (p. 1150)
- Modernize ASP.NET Web Forms applications on AWS (p. 1466)
- Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance (p. 258)
- Use Network Firewall to capture the DNS domain names from the Server Name Indication (SNI) for outbound traffic (p. 1734)
- Visualize AI/ML model results using Flask and AWS Elastic Beanstalk (p. 672)
## Document History

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed from the top navigation bar on this page.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added new database migration patterns for Oracle and Microsoft SQL Server</td>
<td>See the patterns designated as new on the AWS Prescriptive Guidance home page.</td>
<td>January 22, 2020</td>
</tr>
<tr>
<td>Added new database migration patterns</td>
<td>See the patterns designated as new on the AWS Prescriptive Guidance home page.</td>
<td>December 23, 2019</td>
</tr>
<tr>
<td>Added new patterns for SAP ASE, VMware SDDC, MySQL, and MariaDB</td>
<td>See the matrix to view the latest set of patterns.</td>
<td>October 15, 2019</td>
</tr>
<tr>
<td>Added new patterns for Oracle databases and WANdisco Fusion</td>
<td>See the matrix to view new patterns for migrating Oracle databases and using WANdisco Fusion for migration, data replication, and deploying hybrid data lakes.</td>
<td>October 4, 2019</td>
</tr>
<tr>
<td>Added new patterns</td>
<td>See the matrix to view new patterns for migrating databases.</td>
<td>September 6, 2019</td>
</tr>
<tr>
<td>Initial publication (p. 1860)</td>
<td>—</td>
<td>August 5, 2019</td>
</tr>
</tbody>
</table>