Implementation Guide

Scalable Analytics Using Apache Druid on AWS

Scalable Analytics Using Apache Druid on AWS: Implementation Guide

Table of Contents

Solution to set up, operate, leverage scalable analytics capabilities, and manage a hosting environment for Apache Druid on AWS

Publication date: *January 2024 (last update: July 2024)*

The Scalable Analytics using Apache Druid on AWS solution allows you to efficiently deploy, operate, manage and customize a cost-effective, highly available, resilient, and fault tolerant hosting environment for Apache Druid analytics databases on AWS. We expect that customers will be familiar with Apache Druid before deploying and using this solution.

This implementation guide provides an overview of the Scalable Analytics using Apache Druid on AWS solution, its reference architecture and components, considerations for planning the deployment, configuration steps for deploying the solution to the Amazon Web Services (AWS) Cloud.

This guide is intended for solution architects, business decision makers, DevOps engineers, database services administrators, and cloud professionals who want to implement Apache Druid on AWS in their environment.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution across small, medium, or large usage profiles. The estimated cost for running this solution in the US East (N. Virginia) Region for a medium usage profile is USD $2205.47 per month for AWS resources.	Cost
Understand the security considerations for this solution, and recommended security best practices across the solution features.	Security and Security best practices
Know how to configure the solution.	Configure the solution

If you want to . . .	Read . . .
Describes the various options that you configure for your use case while deploying Apache Druid in your AWS account.	
Know which AWS Regions support this solution.	Supported AWS Regions
Find out how to use CloudWatch to monitor the solution. Provides information on all the Druid data logs in Amazon CloudWatch for monitorin g purposes, including alarms, logs, and a dashboard for reporting purposes.	Monitoring the solution
Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) to deploy the solution.	GitHub repository

Features and benefits

The solution provides the following features:

Easily deploy Druid clusters to AWS accounts

The solution offers customers flexibility to customize installations, including your choice of AWS compute engine and storage from a variety of instance and serverless options. You can choose different compute types, such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Kubernetes Service (Amazon EKS) , or AWS Fargate (Fargate), helping you to select the most suitable infrastructure for your specific needs.

High degree of customization

The solution supports various EC2 instance types, including Graviton instances, and offers flexibility in selecting database services, like Aurora PostgreSQL - Compatible Edition, Aurora PostgreSQL

Serverless, or bringing your own database. Customers have the freedom to fine-tune Druid configuration parameters to meet their requirements precisely.

High Availability and resiliency

The solution provides high availability and resiliency through features like automatic scaling with customizable policies, and distributing Druid nodes across multiple availability zones. It also supports recreating clusters from metadata store and deep storage backups, ensuring data is protected and available even in the face of unexpected failures.

Built-in logging and monitoring with Amazon CloudWatch

The solution outputs log entries, emitted by Druid, to a centralized Amazon CloudWatch log group to facilitate debugging and troubleshooting activities, sets up a monitoring dashboard to track the health of the Druid cluster, and configures alarms based on customer preferences.

Integration with Service Catalog AppRegistry and Application Manager, a capability of AWS Systems Manager

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both Service Catalog AppRegistry and Application Manager. With this integration, you can centrally manage the solution's resources and enable application search, reporting, and management actions.

Use cases

Real time ingestion and fast query performance

Apache Druid is a database that is most often used for powering use cases where Scalable ingest, fast query performance, and high uptime are important. Druid is commonly used as the database for GUIs of analytical applications, or as a backend for highly concurrent APIs that need fast aggregations.

Common application areas for Druid include:

- Clickstream analytics (web and mobile analytics) Risk/fraud analysis
- Network telemetry analytics (network performance monitoring)
- Server metrics storage
- Supply chain analytics (manufacturing metrics)

- Application performance metrics
- Business intelligence / OLAP

Concepts and definitions

For a general reference of AWS terms, see the AWS glossary in the AWS General Reference.

segment

Apache Druid stores its data and indexes in *segment files* partitioned by time. Druid creates a segment for each segment interval that contains data.

quorum

A replicated group of servers in the same application is called a *quorum*, and in replicated mode, all servers in the quorum have copies of the same configuration file.

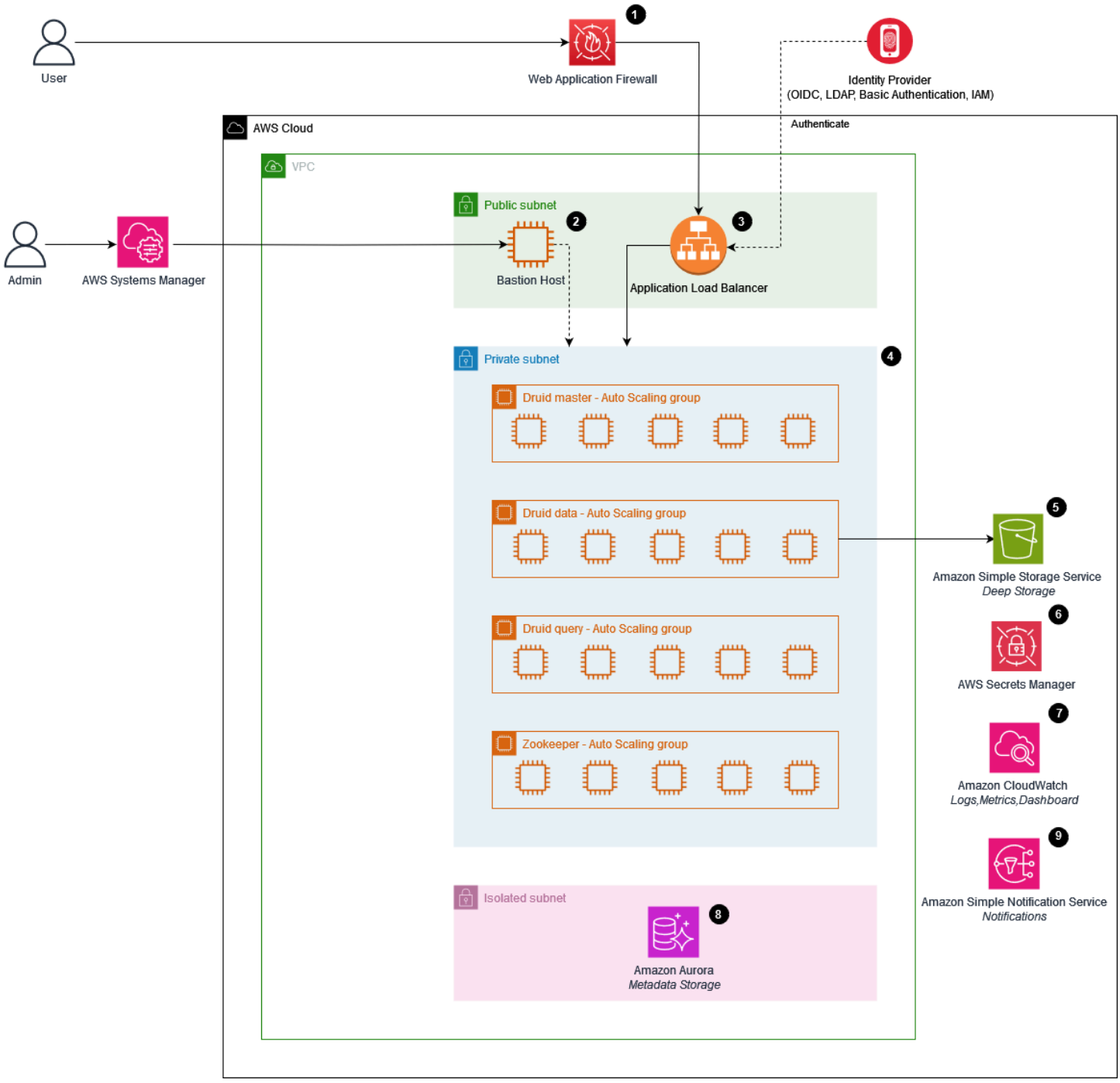
> **Note**
>
> For a general reference of standard Apache Druid concepts, refer to the Apache Druid documentation.

Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

Architecture diagram

Deploying this solution with the default parameters deploys the following components in your AWS account.



Scalable Analytics using Apache Druid on AWS - Architecture diagram

> **Note**
>
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) (AWS CDK) constructs.

The high-level process flow for the solution components deployed with the AWS CDK constructs is as follows. The numbers and description matches the number designated in the following architecture diagram.

The solution deploys the following components that work together to provide a production-ready Druid cluster:

1. AWS Web Application Firewall (AWS WAF) to protect the Druid web console and Druid API endpoints against common web exploits and bots that may affect availability, compromise security, or consume excessive resources. AWS WAF is only provisioned and deployed for internet facing clusters.
2. A security hardened Linux server (Bastion host) to manage access to the Druid servers running in a private network separate from an external network. It can also be used to access the Druid web console through SSH tunneling where a private Application Load Balancer (ALB) is deployed.
3. An ALB serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple query servers in multiple Availability Zones.
4. A private subnet consisting of:
   - **Druid master Auto scaling group**: An Auto scaling group contains a collection of Druid master servers. A Master server manages data ingestion and availability and is responsible for starting new ingestion jobs and coordinating availability of data on the data servers. Within a Master server, functionality is split between two processes: the Coordinator and Overlord.
   - **Druid data Auto scaling group**: An Auto scaling group contains a collection of Druid data servers. A data server runs ingestion jobs and stores queryable data. Within a data server, functionality is split between two processes: the Historical and MiddleManager.
   - **Druid query Auto scaling group**: An Auto scaling group contains a collection of Druid query servers. A query server provides the endpoints that users and client applications interact with, routing queries to data servers or other query servers. Within a Query server, functionality is split between two processes: the Broker and Router.
   - **ZooKeeper Auto scaling group**: An Auto scaling group contains a collection of ZooKeeper servers. Apache Druid uses Apache ZooKeeper (ZK) for management of current cluster state.

5. An Amazon Simple Storage Service (Amazon S3) bucket provides deep storage for the Apache Druid cluster. Deep storage is the location where the segments are stored.
6. AWS Secrets Manager stores the secrets used by Apache Druid including the RDS secret, and the administrator secret. It also stores the credentials for the system account the Druid components use to authenticate with each other.
7. Amazon CloudWatch support logs, metrics, and dashboards.
8. An Amazon Aurora PostgreSQL database provides the metadata storage for the Apache Druid cluster. Druid uses the metadata store to house only metadata about the system only, and does not store the actual data.
9. The notification system, powered by Amazon Simple Notification Service (Amazon SNS), delivers alerts or alarms promptly when system events occur. This ensures immediate awareness and action when needed.

AWS Well-Architected design considerations

This solution uses the best practices from the AWS Well-Architected Framework, which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework benefit this solution.

Operational excellence

This section describes how we architected this solution using the principles and best practices of the operational excellence pillar.

- Logs and metrics from all Druid components are gathered and stored in CloudWatch.
- A comprehensive CloudWatch dashboard is provided to monitor the operational status of underlying services.
- Alarms are set up within CloudWatch to provide timely notifications for issues or anomalies.
- Server access logging is enabled to provide detail records for the requests that are made to an Amazon S3 bucket.
- Amazon Virtual Private Cloud (Amazon VPC) flow logs are enabled to monitor IP traffic both incoming and outgoing through network interfaces in your VPC Security.

Security

This section describes how we architected this solution using the principles and best practices of the security pillar.

- Multiple authentication schemas are supported including basic authentication, OIDC authentication, and LDAP authentication.
- All inter service communications use AWS Identity and Access Management (IAM) roles. Communications between EC2 instances hosting the Druid process and Aurora Postgres uses basic authentication and does not use IAM.
- All IAM roles used by the solution follow the least privilege access principle. They only contain the minimum permissions required so that the service can function properly.
- AWS WAF is associated with AWS ALB to protect the Druid cluster from common application-layer exploits. AWS WAF is only provisioned and associated with the Application Load Balancer (ALB) when it is configured to be internet-facing and in the public mode.
- All data stored in Amazon Aurora, AWS Backup, and Amazon S3 buckets have encryption at REST with customer managed keys.
- All communication between Apache Druid and AWS service endpoints is covered by TLS.
- TLS connectivity is implemented within the Druid cluster, as well as from the Druid cluster to the rest of the supported AWS services.
- VPC endpoints are introduced to privately connect to supported AWS services.

Reliability

This section describes how we architected this solution using the principles and best practices of the reliability pillar.

- Amazon EC2 Auto Scaling is used to distribute instances across Availability Zones, and replace the failed instances automatically.
- The database-first migration strategy allows for cluster restoration using existing backups of the metadata store and deep storage.
- The solution stores data in Amazon S3 so it persists in multiple Availability Zones by default.
- AWS Backup is used to regularly backup the metadata store at defined intervals.

Performance efficiency

This section describes how we architected this solution using the principles and best practices of the performance efficiency pillar.

- The solution supports AWS Fargate for serverless compute and Aurora PostgreSQL Serverless.
- You can deploy the solution in any AWS Region that supports the required AWS services.
- The solution provides versatile Automatic scaling policies, including CPU utilization, request per second, and scheduled scaling.
- Developed using AWS CDK and managed through AWS CloudFormation stacks, it follows a complete Infrastructure-as-Code (IAC) approach, simplifying upgrades and resource management.
- The solution maximizes the utilization of AWS Managed Services. For more details, refer to the AWS services used in this solution section.

Cost optimization

This section describes how we architected this solution using the principles and best practices of the cost optimization pillar.

- The solution offers support for various EC2 instance types, including Graviton-based EC2 instances.
- It supports a full serverless architecture by leveraging AWS Fargate and Aurora PostgreSQL Serverless.

Sustainability

This section describes how we architected this solution using the principles and best practices of the sustainability pillar.

- Support for Graviton-based EC2 instances aids in minimizing your carbon footprint and aligning with sustainability objectives.
- Amazon EC2 Auto Scaling is used to scale your workloads dynamically. The predicative auto scaling is used to proactively scale as you anticipate predicted and planned changes in demand.

Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

- Network infrastructure
- Metadata storage
- Deep storage
- ZooKeeper quorum
- Installation files
- Default users
- Logs, metrics, and dashboard
- Notifications
- AWS CloudFormation custom resources

Network infrastructure

By default, the solution provisions a new Virtual Private Cloud (VPC) consisting of three types of subnets: public, private, and isolated. The subnet type is determined by how you configure routing for your subnets. To read more, refer to the Subnet types for Amazon VPC section.

Within this configuration, the Druid EC2 instances operate within the private subnets. Security groups are employed to enhance the security of these instances by permitting traffic exclusively from the ALB or from other instances within the Druid cluster, thus restricting access to a select set of trusted sources.

The Druid query instances are accessible via an Application Load Balancer (ALB) that exposes only HTTP and HTTPS protocols. An additional bastion host can be deployed to facilitate access to the instances located within the private subnet or the database in the isolated subnet. Additionally, you can deploy the solution within an existing VPC if needed.

Metadata storage

By default, the solution deploys an Amazon Aurora PostgreSQL cluster in an isolated subnet to serve as metadata storage for the Druid cluster, ensuring TLS connectivity to secure

communication with the DB cluster. It also generates a secret to store the credentials of the DB cluster's master user. For example, for credentials for the master user of Amazon Aurora cluster.

```
{
        "dbClusterIdentifier": "druidec2stack-<cluster name>-
druidmetadataconstructaurorac-<random characters>",
        "password": "<password>",
        "dbname": "DruidMetadata",
        "engine": "postgres",
        "port": 5432,
        "host": "druidec2stack-<cluster name>-druidmetadataconstructaurorac-<random
 characters>.cluster-<random characters>.<region>.rds.amazonaws.com",
        "username": "druid"
}
```

Aurora automatically backs up your cluster volume and preserves restore data for the duration of the backup retention period. By default, the solution configures the backup retention period as 14 days.

Aurora automated backups are continuous and incremental, so you can quickly restore to any point within the backup retention period. Additionally, the solution offers support for using AWS Backup to create backups for the Aurora cluster.

Deep storage

As a default configuration, the solution creates a new S3 bucket designated as deep storage for the Druid cluster. Additionally, it generates a AWS Key Management Service (AWS KMS) key to provide server-side encryption with AWS KMS (SSE-KMS) for the deep storage.

ZooKeeper quorum

The solution sets up a ZooKeeper quorum, consisting of the specified number of instances as defined in the CDK configuration. Metrics from these ZooKeeper instances are gathered and sent to CloudWatch under the metric namespace `AWSSolutions/Druid`.

The solution additionally allocates an extra Elastic Network Interface (ENI) for each instance, enabling Druid to establish connections with ZooKeeper through static IP addresses.

Installation files

The solution sets up an S3 bucket for storing the installation files required by the solution.

During the build process, the solution downloads the necessary files, which include Apache Druid, Apache ZooKeeper, and Druid community extensions from the Apache Content Delivery Network (CDN) and then uploads these files into the installation bucket. The solution performs a signature/hash verification to ensure the binary's authenticity and integrity prior to deployment.

As part of the EC2 bootstrapping process, these files are retrieved from the installation bucket and installed onto the instances under the `/home/druid-cluster` directory.

> **Note**
>
> The installation files are kept in the installation bucket throughout the lifetime of the cluster. When a new EC2 instance starts, it retrieves the installation files from the installation bucket. The installation files may also get updated when configuration is changed, such as upgrading Apache Druid from v26 to v27.

Default users

The solution uses Druid's authentication and authorization module for user authentication, with basic authentication enabled by default.

During the initial deployment, it creates default users `admin` and `druid_system`, both with full permissions. The `admin` user is intended for your use, while the `druid_system` user is specifically reserved for internal system communication and operations.

Upon deployment, the solution generates the following secrets in AWS Secrets Manager to securely store the credentials for the `admin` and `druid_system` users.

Secret Description	Secret Example
Administrator user credentials for Druid cluster `<cluster name>`	`{ "password": "<password>", "username": "admin" }`

Secret Description	Secret Example
Internal system user credentials for Druid cluster `<cluster name>`	`{ "password": "<password>", "username": "druid_system" }`

Note

Modifying the password in AWS Secrets Manager will not automatically change the user's password. To update a user's password, you must change the password using the Druid API and then update the password manually in AWS Secrets Manager.

Logs, metrics, and dashboard

The solution establishes a log group named `/aws/solutions/druid/<cluster name>` in CloudWatch for storing various logs collected from EC2 instances. These logs include:

- Druid process logs from EC2 instances
- ZooKeeper process logs
- System initialization logs from EC2 instances

In addition, it also sets up a S3 bucket to store the logs gathered from AWS infrastructure and services, including:

- VPC Flow Logs
- ALB access logs
- S3 bucket access logs

The solution creates a metric namespace named `AWSSolutions/Druid` to store the metrics collected from EC2 instances. These metrics include:

- Druid application metrics (e.g. ingestion/query throughput)
- Operating system metrics (e.g. CPU/memory/disk utilization)

The solution includes an Amazon CloudWatch dashboard named `druid-<cluster name>-ops-dashboard` configured to provide an overview of the health and operational status of all components in the solution.

Notifications

When deployed, the solution sets up the following topics in the Amazon SNS:

- Alarm notification topic – This topic receives the notifications for the CloudWatch alarms.
- Auto scaling notification topic – This topic receives the scaling event notifications from auto scaling groups.

To receive messages published to the topics, you must subscribe an endpoint to the topic. When you subscribe an endpoint to a topic, the endpoint begins to receive messages published to the associated topic.

AWS CloudFormation custom resources

The solution incorporates AWS CloudFormation custom resources to handle the following parameters:

- Set up default roles, permissions, and group mappings for OIDC authentication
- Set up Druid retention configuration rules

AWS services in this solution

AWS service	Description
Amazon S3	**Core.** The solution provisions the following S3 buckets: • Deep storage bucket to store the segments. • Installation bucket to store the installation files as needed by the solution. • Access logging bucket to store the access logs from ALB, and S3 buckets.

AWS service	Description
Amazon Elastic Compute Cloud	**Core.** The solution provisions EC2 instances to run Apache Druid and Apache ZooKeeper.
Amazon Aurora	**Core.** The solution provisions an Aurora PostgreSQL cluster to serve as the metadata storage.
Amazon Elastic Load Balancer	**Core.** Application load balancer to distribute the incoming traffic among the Druid query nodes.
AWS Secrets Manager	**Core.** Secrets to store master user credentials for Aurora DB cluster, and credentials of the users “admin” and “druid_system”.
AWS Key Management System	**Core.** KMS keys used to encrypt the data in S3 buckets, Aurora cluster, SNS topic, and EFS.
Amazon Elastic Block Store	**Core.** EBS volumes to serve as segment cache for historical nodes.
Amazon CloudWatch	**Supporting.** The solution uses CloudWatch for logs, metrics, alarms, and dashboard.
Amazon Simple Notification Service	**Supporting.** Topics to receive CloudWatch alarm notifications and auto scaling group scaling event notifications.
AWS WAF	**Supporting.** Protect Druid web console and API endpoints from common application-layer exploits that can affect availability or consume excessive resources.
AWS Systems Manager	**Supporting.** Provides application-level resource monitoring and visualization of resource operations and cost data.

AWS service	Description
Amazon EventBridge	**Supporting.** The solution creates an EventBridge rule to receive the event from auto scaling group.
Amazon Elastic Kubernetes Service	**Optional.** When opting for EKS deployment, the solution initializes an EKS cluster to execute the Apache Druid workload.
Amazon Elastic File System	**Optional.** When opting for EKS Fargate deployment, the solution creates an EFS filesystem to provide storage to Fargate workloads.
Amazon Route 53	**Optional.** The solution provides the option for integration with Route 53 to manage the domain for accessing the Druid cluster.

Plan your deployment

This section describes the cost, security, and quotas considerations prior to deploying the solution.

Supported AWS Regions

Scalable Analytics using Apache Druid on AWS is available in the following AWS Regions:

Region name	Region name
us-east-1	US East (N. Virginia)
us-gov-east-1	AWS GovCloud (US-East)
us-east-2	US East (Ohio)
us-west-1	US West (Northern California)*
us-west-2	US West (Oregon)
af-south-1	Africa (Cape Town)
ap-east-1	Asia Pacific (Hong Kong)
ap-south-2	Asia Pacific (Hyderabad)
ap-southeast-3	Asia Pacific (Jakarta)
ap-southeast-4	Asia Pacific (Melbourne)
ap-south-1	Asia Pacific (Mumbai)
ap-northeast-3	Asia Pacific (Osaka)
ap-northeast-2	Asia Pacific (Seoul)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)

Region name	Region name
ap-northeast-1	Asia Pacific (Tokyo)
ca-central-1	Canada (Central)
eu-central-1	Europe (Frankfurt)
eu-west-1	Europe (Ireland)
eu-west-2	Europe (London)
eu-west-3	Europe (Paris)
eu-south-2	Europe (Spain)
eu-north-1	Europe (Stockholm)
eu-central-2	Europe (Zurich)
me-south-1	Middle East (Bahrain)
me-central-1	Middle East (UAE)
sa-east-1	South America (São Paulo)

Cost

You are responsible for the cost of the AWS services used while running this solution. As of the latest revision, the costs for running this solution with the default settings (small usage profile) in the US East (N. Virginia) Region is approximately **$714.46 per month**, for a medium usage profile in the US East (N. Virginia) Region is approximately **$2,202.47 per month**, and for a large usage profile in the US East (N. Virginia) Region is approximately **$13,645.27 per month**.

These costs are for the resources shown in the Cost table section. See the pricing webpage for each AWS service used in this solution.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, see the pricing webpage for each AWS service used in this solution.

Cost table

The following tables provide a sample cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region for one month, encompassing the small, medium, and large usage profiles.

Small usage profile

Profile assumptions: ingestion throughput at 30,000 records per second, query throughput at 25 queries per second.

AWS service	Dimensions	Cost [USD]
Amazon EC2	• Druid master: 3 x t4g.medium • Druid query: 3 x t4g.medium • Druid data: 3 x (t4g.medium + 100GB EBS GP2 volume) • ZooKeeper: 3 x t4g.small	$287.53
Amazon ELB	1 x ALB, 5 GB/h processed bytes (EC2 Instances and IP addresses as targets)	$45.63
Amazon Aurora	3 x db.t4g.medium	$247.81
Amazon S3	1 TB standard storage + 1,000,000 requests per month	$29.67
AWS Key Management Service	7 x customer managed keys	$7
AWS Secrets Manager	4 x secrets	$1.6
Amazon CloudWatch	50 GB standard logs ingested per month, 200 custom	$95.22

AWS service	Dimensions	Cost [USD]
	metrics + 1,000,000 metric requests per month	
	Total:	**$714.46 [USD] / month**

Medium usage profile

Profile assumptions: ingestion throughput at 120,000 records per second, query throughput at 100 queries per second.

AWS service	Dimensions	Cost [USD]
Amazon EC2	• Druid master: 3 x m6g.xlarge • Druid query: 3 x m6g.xlarge • Druid data: 3 x (m6g.2xlarge + 500GB EBS GP2 volume) • ZooKeeper: 3 x t4g.medium	$1,572.62
Amazon ELB	1 x ALB, 20 GB/h processed bytes (EC2 Instances and IP addresses as targets)	$133.23
Amazon Aurora	3 x db.t4g.medium	$247.81
Amazon S3	5 TB standard storage + 5,000,000 requests per month	$119.76
AWS Key Management Service	7 x customer managed keys	$7
AWS Secrets Manager	4 x secrets	$1.6

AWS service	Dimensions	Cost [USD]
Amazon CloudWatch	100 GB standard logs ingested per month, 200 custom metrics + 1,000,000 metric requests per month	$120.45
	Total:	**$2,202.47 [USD] / month**

Large usage profile

Profile assumptions: ingestion throughput at 1.4 million records per second, query throughput at 1,200 queries per second.

AWS service	Dimensions	Cost [USD]
Amazon EC2	• Druid master: 3 x m6g.4xlarge • Druid query: 3 x m6g.4xlarge • Druid data: 3 x (m6g.16xlarge + 5 TB EBS GP2 volume) • ZooKeeper: 3 x m6g.2xlarge	$10,268.76
Amazon ELB	1 x ALB, 200 GB/h processed bytes (EC2 Instances and IP addresses as targets)	$1,184.43
Amazon Aurora	3 x db.t3.large	$427.39
Amazon S3	50 TB standard storage + 10,000,000 requests per month	$1,181.60
AWS Key Management Service	7 x customer managed keys	$7

AWS service	Dimensions	Cost [USD]
AWS Secrets Manager	4 x secrets	$1.6
Amazon CloudWatch	1,000 GB standard logs ingested per month, 200 custom metrics + 1,000,000 metric requests per month	$574.50
	Total:	**$13,645.27 [USD] / month**

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared responsibility model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. The solution creates IAM roles that grant the solution's constructs to access Regional resources provisioned by the solution, such as:

- IAM role used by the EC2 instances that run Druid workloads to read and write data in S3 buckets.
- IAM roles used by AWS CloudFormation custom resources to retrieve the password from the Druid `system_user` secret within AWS Secrets Manager.

By default, all Amazon S3 buckets for the solution have the following configuration:

- Blocked all public access
- Versioning enabled
- Access log enabled

- Encryption at rest by an AWS KMS customer managed key

Additionally, the Amazon S3 buckets are also configured with a default buckets policy that deny all non-HTTPS requests to ensure data in transit encryption.

AWS WAF

This solution incorporates the deployment of AWS Web Application Firewall (WAF) when the Application Load Balancer (ALB) is configured to be internet-facing. AWS WAF is used to enhance the security of the web applications exposed through the ALB by providing protection against various web-based threats and attacks.

AWS Key Management Service keys

The solution allows you to provide your own AWS KMS keys to encrypt stored data in the S3 bucket and Aurora cluster. We recommend referring to the security best practices for AWS Key Management Service to enhance the protection of your encryption keys.

Data protection

All data committed to the solution is encrypted at rest using AWS Key Management Service (AWS KMS) customer managed keys. This includes data stored in the following services:

- Amazon S3
- Amazon Aurora
- Amazon SNS

Communication between the solution's different components is over HTTPS to ensure data encryption in transit.

Security best practices

This section provides several security features to consider, as you develop and implement your own security policies.

> **Note**
>
> The following best practices are general guidelines and do not represent a complete security solution. These best practices might not be appropriate or sufficient for your specific environment, so treat them as helpful considerations, not prescriptive guidance.

User authentication and authorization

The solution is configured to utilize basic authentication by default, offering adaptability for additional support such as OIDC and LDAP authentication through configuration settings. However, we advise caution when customizing user authentication and authorization settings.

- In a production environment, we strongly recommend activating basic authentication at the very least to maintain a baseline security measure.
- For OIDC authentication, ensuring the accuracy of group and role mapping is essential.
- When creating Druid roles, adhere to the principle of least privilege to establish a minimum permission security stance.

Domain and TLS certificate

The solution integrates with Amazon Route 53 and AWS Certificate Manager, streamlining the provisioning of a domain and TLS certificate during deployment when a Route53 hosted zone configuration is specified. Alternatively, it provides the flexibility to deploy without Route 53 configuration, allowing for an external domain setup. In this scenario, a default HTTP listener is established using the application load balancer, and traffic will be exposed over HTTP without encryption in transit.

This scenario poses significant security risks, such as the potential for eavesdropping on sensitive information. To address this concern, we highly recommended establishing a custom domain and employ the TLS policy *ELBSecurityPolicy-TLS13-1-2-2021-06* to enhance security by encrypting the data in transit.

Upon completing the external domain setup, it is recommended to configure the TLS certificate ARN along with the domain, and trigger a redeployment of the solution to implement these changes. This verifies that the application load balancer exposes HTTPS, thereby fortifying communication security through TLS encryption.

Third-party extensions

The solution's default configuration includes a minimum set of essential extensions to enable core functionalities. Users have the flexibility to tailor the list of extensions that can be loaded into the cluster. It is important for users to assume responsibility for the security of the selected extensions.

To uphold a robust security posture, we strongly advise consistently monitoring for new releases and promptly applying updates, thereby proactively addressing and mitigating any potential vulnerabilities.

Deploy the solution into existing VPC

The solution offers the flexibility to deploy into an existing VPC. When opting for this configuration, it is advisable to ensure that the existing VPC is equipped with three distinct types of subnets: public, private, and isolated, spanning across at least two availability zones.

Whether a subnet is public or private refers to whether traffic within the subnet is routed through an internet gateway. Public subnets have a route table entry to the internet through the internet gateway, but private subnets do not have this entry. Isolated subnets have no routes to destinations outside its VPC. For more information about subnet types, refer to the definition of subnet types. This strategic subnet arrangement allows for optimal segregation of component; the RDS database clusters operate in the isolated subnets, Druid nodes in the private subnets, and the load balancer is positioned in the public subnets. This architecture enhances security and scalability by appropriately isolating different layers of the infrastructure.

AMI security

The solution automatically selects the latest AMI for Amazon Linux 2 in the deployment process. Opting for the most recent AMI ensures that the system benefits from the latest security patches, thereby maintaining an up-to-date and secure environment. This proactive approach aligns with best practices for security and helps safeguard the integrity of the deployed instances. Continuous use of the latest AMI version contributes to a more resilient and well-protected infrastructure.

The solution also supports the flexibility to bring your own AMI. It is important to ensure that your base AMI adheres to security best practices.

- Start with a secure and minimal base image, and then apply patches systematically to maintain a secure foundation for your instances.
- Establish a routine and predictable patching schedule for your AMIs.

- Regularly check for operating system and software updates, and apply patches in a timely manner to address known vulnerabilities.

Public and private mode for EKS cluster endpoint

Amazon EKS offers public-only, public-and-private, and private-only cluster endpoint modes. The default mode is configured to be private-only in the solution, however we recommend configuring cluster endpoint in public and private mode. This option allows Kubernetes API calls within your cluster's VPC (such as node-to-control-plane communication) to use the private VPC endpoint and traffic to remain within your cluster's VPC.

You can access your cluster API server from the internet. However, we strongly recommend limiting the CIDR blocks that can use the public endpoint. Learn how to configure public and private endpoint access, including limiting CIDR blocks. For more guidance on how to secure the EKS clusters, refer to the EKS best practices topic.

EKS master IAM role

For EKS deployment, the solution requires the user to supply an ARN for the EKS master role. This role, serving as the IAM principal responsible for creating the cluster, is automatically endowed with `system:masters` permissions within the cluster's Role-Based Access Control (RBAC) configuration in the Amazon EKS control panel. For more guidance on how to secure the access for this role, refer to the EKS best practices topic.

Quotas

Service Quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the services implemented in this solution. For more information, see AWS service quotas.

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the Service endpoints and quotas page in the PDF instead.

AWS CloudFormation quotas

Your AWS account has AWS CloudFormation quotas that you should be aware of when launching the stack in this solution. By understanding these quotas, you can avoid limitation errors that would prevent you from deploying this solution successfully. For more information, see AWS CloudFormation quotas in the *AWS CloudFormation User's Guide*.

Deploy the solution

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Before you launch the solution, review the cost, architecture, network security, and other considerations discussed earlier in this guide.

Prerequisites

Build environment specifications

To build and deploy this solution:

- We recommend using Ubuntu with minimum 4 cores CPU and16GB RAM. MacOS (Intel) or other Linux distributions are also supported.
- The computer used to build the solution must be able to access the internet.

AWS account

A CDK bootstrapped AWS account: You must Bootstrap your AWS CDK environment in the target Region you want to deploy, using the AWS CDK toolkit's `cdk bootstrap` command. From the command line, authenticate into your AWS account, and run `cdk bootstrap aws://<YOUR ACCOUNT NUMBER>/<REGION>`. For more information, refer to the AWS CDK's Bootstrapping page.

Tools

- The latest version of the AWS CLI, installed and configured.
- The latest version of the AWS CDK.
- Nodejs version 16 or newer.
- Git command line.
- Java Runtime

- The solution requires a Java 8 Runtime. We recommend using Amazon Corretto 8. Alternatively, you can also use other OpenJDKs such as Eclipse Temurin.

- Maven (>=3.5.2)

- https://maven.apache.org/install.html. We recommend configuring Maven to use an OpenJDK8 compatible JAVA version, such as Amazon Corretto 8.

- Docker Desktop (>= v20.10): https://www.docker.com/get-started/
- Curl: https://curl.se/

Deployment process overview

Follow the step-by-step instructions in this section to configure and deploy the solution into your account. Before you launch the solution, review the cost, architecture, network security, and other considerations discussed earlier in this guide.

Time to deploy: Approximately 40 minutes

Choose deployment option

You can deploy Druid with one of the following compute options:

- Amazon EC2 (default option)
- EKS with EC2 hosting
- EKS with Fargate hosting

You can have multiple deployments/clusters in the same Region/account, and choose different compute options across the deployments.

Choose Druid configuration

You can use one of the three pre-configured Druid settings: **small, medium**, or **large**. If your use case matches any of these settings, use the source/quickstart/ folders (small, medium, or large) to deploy Apache Druid in your AWS account with these settings pre-configured.

Small usage profile (profile assumptions: ingestion throughput at 30,000 records per second, query throughput at 25 queries per second)

AWS service	Dimensions
Amazon EC2	• Druid master: 3 x t4g.medium • Druid query: 3 x t4g.medium • Druid data: 3 x (t4g.medium + 100GB EBS GP2 volume) • ZooKeeper: 3 x t4g.small
Amazon ELB	1 x ALB, 5 GB/h processed bytes (EC2 Instances and IP addresses as targets)
Amazon Aurora	3 x db.t4g.medium
Amazon S3	1 TB standard storage + 1,000,000 requests per month
AWS Key Management Service	7 x customer managed key
AWS Secrets Manager	4 x secrets
Amazon CloudWatch	50 GB standard logs ingested per month, 200 custom metrics + 1,000,000 metric requests per month

Medium usage profile (profile assumptions: ingestion throughput at 120,000 records per second, query throughput at 100 queries per second)

AWS service	Dimensions
Amazon EC2	• Druid master: 3 x m6g.xlarge • Druid query: 3 x m6g.xlarge • Druid data: 3 x (m6g.2xlarge + 500 GB EBS GP2 volume) • ZooKeeper: 3 x t4g.medium

AWS service	Dimensions
Amazon ELB	1 x ALB, 20 GB/h processed bytes (EC2 Instances and IP addresses as targets)
Amazon Aurora	3 x db.t4g.medium
Amazon S3	5 TB standard storage + 5,000,000 requests per month
AWS Key Management Service	7 x customer managed key
AWS Secrets Manager	4 x secrets
Amazon CloudWatch	100 GB standard logs ingested per month, 200 custom metrics + 1,000,000 metric requests per month

Large usage profile (profile assumptions: ingestion throughput at 1.4 million records per second, query throughput at 1,200 queries per second)

AWS service	Dimensions
Amazon EC2	• Druid master: 3 x m6g.4xlarge • Druid query: 3 x m6g.4xlarge • Druid data: 3 x (m6g.16xlarge + 5 TB EBS GP2 volume) • ZooKeeper: 3 x m5.2xlarge
Amazon ELB	1 x ALB, 200 GB/h processed bytes (EC2 Instances and IP addresses as targets)
Amazon Aurora	3 x db.t3.large
Amazon S3	50 TB standard storage + 10,000,000 requests per month
AWS Key Management Service	7 x customer managed key

AWS service	Dimensions
AWS Secrets Manager	4 x secrets
Amazon CloudWatch	1,000 GB standard logs ingested per month, 200 custom metrics + 1,000,000 metric requests per month

Build and deploy

1. From the solution GitHub repository, download the source files for this solution The Scalable Analytics using Apache Druid on AWS templates are generated using the AWS Cloud Development Kit (AWS CDK) (AWS CDK).
2. Open the terminal and navigate to the source directory: `cd source/`
3. Using `cdk.json`, configure the solution for your requirements.

> **Note**
>
> We recommend using the `cdk.json` example in the `source/quickstart` folder, and making changes to suit your use cases accordingly. Refer to the Configure the solution section for more information.

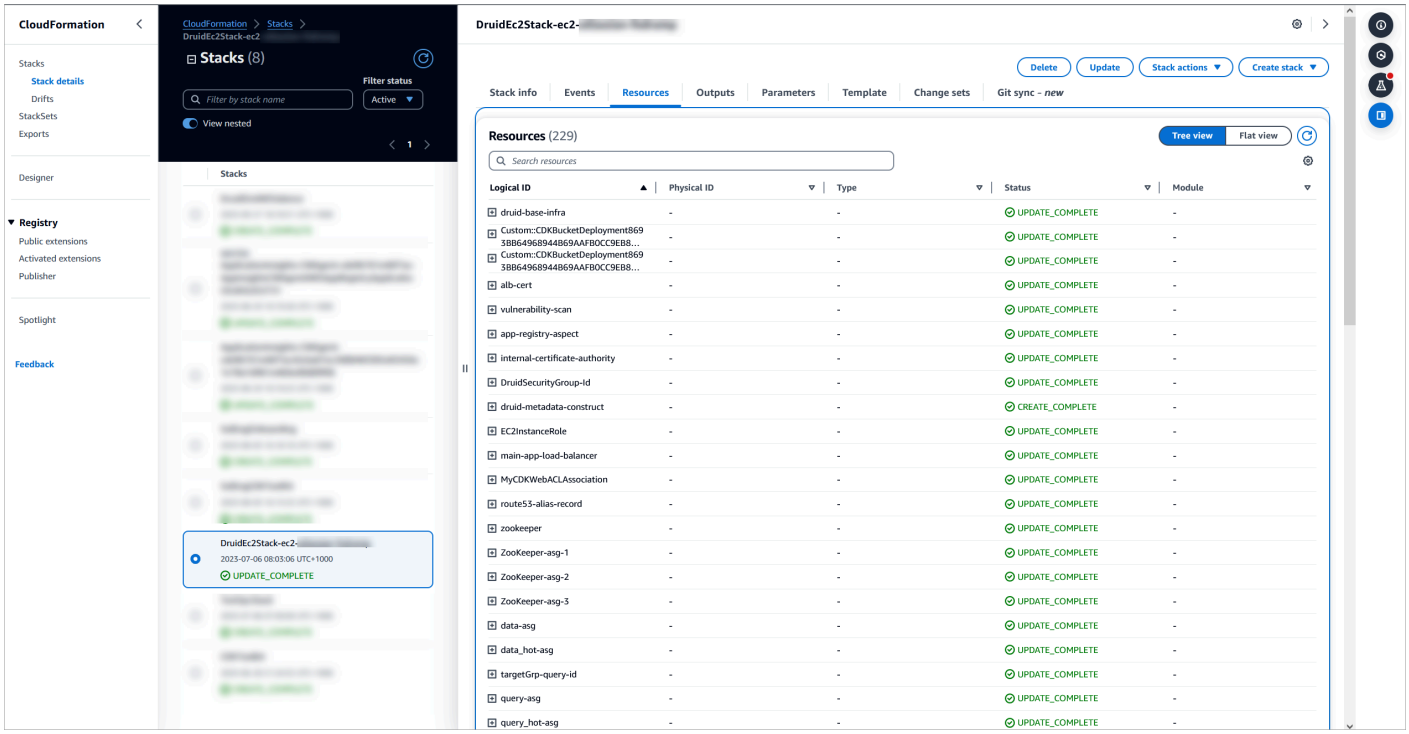
4. To install the solution dependencies, type `npm install`.
5. To build the code, type `npm run build`.
6. To deploy the solution, type `npm run cdk deploy`.

Post-deployment

Once you have configured/customized and deployed the solution, you can log into the AWS Management Console, and verify the stacks installed as part of the deployment.

AWS CDK will deploy a stack with the specified name and provisioned resources for the solution that will show up shortly after this stack is completed on deployment. The main stack of the solution is named DruidOptionStack-CustomName and contains the relevant solution resources.

1. Sign into your AWS Management console, and navigate to **CloudFormation > Stacks**. Make sure you select the Region where this solution has been deployed.
2. Select the stack name to view the provisioned resources.



Solution provisioned resources

Configure the solution

This section describes the various options that you configure for your use case while deploying Apache Druid in your AWS account.

The following table lists the configuration, setting, and if the configuration is mandatory before you deploy the solution in your AWS account.

Configuration	Setting	Default value	Mandatory/Optional
Amazon Machine Images (AMI)	`customAmi`	Latest Amazon Linux 2 AMI	Optional
Data retention policy	`retainData`	true	Optional

Configuration	Setting	Default value	Mandatory/Optional
Network			Optional
• VPC CIDR Range	`vpcCidr`	10.0.0.0/16	Optional
• VPC ID	`vpcId`	N/A	Optional
• Application Load Balancer	`internetFacing`	true	Optional
• Bastion host	`bastionHost`	false	Optional
Druid domain			Optional
• Route53 hosted zone	`route53HostedZoneName` , and `route53HostedZoneId`	N/A	Optional
• Druid domain	`druidDomain`	N/A	Optional
• TLS certificate	`tlsCertificateArn`	N/A	Optional
FIPS 140-2	`useFipsEndpoint`	false	Optional
Identity provider	`oidcIdpConfig`	N/A	Optional
Druid configuration			Mandatory
• Version	`druidVersion`		Mandatory
• Cluster name	`druidClusterName`		Mandatory
• Operation platform	`druidOperationPlatform`		Mandatory

Configuration	Setting	Default value	Mandatory/Optional
• Custom IAM policy list	`druidInstanceIamPolicyArns`	N/A	Optional
• Query concurrency rate limit	`druidConcurrentQueryLimit`	100	Optional
• Extensions	`druidExtensions`		Mandatory
• Common runtime properties	`druidCommonRuntimeConfig`	N/A	Optional
• Retention rules	`druidRetentionRules`		Optional
Druid metadata store	`druidMetadataStoreConfig`	Amazon Aurora PostgreSQL Serverless	Mandatory
Druid deep storage	`druidDeepStorageConfig`		Mandatory
Druid EC2 configuration			**Mandatory** (if the druidOperationPlatform is EC2)
• Auto scaling group	`<Druid node type>`		Mandatory
• Rolling update policy	`rollingUpdatePolicy`	N/A	Optional
• Auto scaling policy	`autoScalingPolicy`	N/A	Optional

Configuration	Setting	Default value	Mandatory/Optional
• Runtime configuration	`runtimeConfig`	N/A	Optional
Druid EKS configuration			**Mandatory** (if the druidOperationPlatform is EKS)
• EKS cluster configuration	`druidEksConfig`		Mandatory
• EC2 capacity provider configuration	`<nodeGroupName>`		**Mandatory** (if capacityProviderType is EC2)
• Fargate capacity provider configuration	`<druidProcessName>`		**Mandatory** (if capacityProviderType is Fargate)

Amazon Machine Images (AMI)

By default, the EC2 hosting option provisions EC2 instances with Amazon Linux 2.

To override this, specify the `customAmi` object in the `cdk.json` file. This object should provide the AMI name and owners' account IDs or the alias that cdk would use to perform an AMI lookup. Depending on the instance types utilized in the cluster, provide the corresponding AMI for **arm64** (Graviton instances) or **amd64** (x86-based instance types).

> **Note**
>
> The solution has been tested with Amazon Linux, Ubuntu 20.04 LTS and Ubuntu 22.04 LTS.

```
"customAmi": {
    "arm64": {
        "name": "ubuntu/images/hvm-ssd/ubuntu-focal-20.04-arm64-server*",
        "owners": ["amazon"]
```

```
    },
    "amd64": {
        "name": "ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server*",
        "owners": ["amazon"]
    }
},
```

Data retention policy

By default, all the solution data (S3 buckets, Aurora DB instances, Aurora DB snapshots etc.) is retained when you uninstall the solution.

To remove this data, in the configuration file, set the retainData flag to false. You are liable for the service charges when solution data is retained in the default configuration.

```
"retainData": false,
```

Network

Virtual Private Cloud (VPC)

- VPC CIDR Range: By default, the solution creates a new VPC for deployment with 10.120.0.0/16 as the default CIDR range. You can override this using this configuration:

```
"vpcCidr": "x.x.x.x/x",
```

- VPC ID: You can deploy to an existing VPC. To do this, use the following configuration option:

```
"vpcId": "vpc-xxxxxxxxxxxxxxxx"
```

Application Load Balancer setting

The solution will provision an ALB to route requests to Druid query instances. It supports both internet facing and private options as follows.

```
"internetFacing": true,
```

The default setting creates a private Application Load Balancer (ALB). To access this private ALB, you can either use an AWS VPN or SSH tunneling via a bastion host. However, if the internetFacing parameter is set to true, the ALB will be publicly available. In such an instance, use the AWS WAF (Web Application Firewall) with your ALB to allow or block requests based on rules in a web access control list (web ACL).

Bastion Host

You can launch a bastion host in the public subnet using this configuration, which facilitates the access to the hosts and services running in private and isolated subnets.

```
"bastionHost": true,
```

Druid domain

This solution can integrate with Amazon Route 53 to automate the creation of domains, streamlining access to the Druid web console and API endpoints.

With the following Route 53 configuration, this solution automatically creates a Route53 domain using the specified druidDomain in the supplied hosted zone. Additionally, this solution creates a certificate in the AWS Certificate Manager (ACM) and associates it with the Application Load Balancer (ALB) to enable secure HTTPS communication.

- Route 53 hosted zone configuration

```
"route53HostedZoneName": "<The hosted zone name in Route 53. eg. example.com>",

"route53HostedZoneId": "<The hosted zone ID in Route 53. eg. Z03935053V8YUTYYYEXXX>",
```

- Druid domain configuration

```
"druidDomain": "<A full domain address. eg. druid.example.com>"
```

> **Note**
>
> If none of these configurations are set up, the solution assigns a generic Application Load Balancer (ALB) domain name for your Druid cluster which will expose HTTP protocol only.

You have the flexibility to set up the domain outside of the solution. In such a case, you must configure the TLS certificate and the druidDomain to facilitate secure HTTPS access to the Druid cluster after the domain has been successfully set up.

- TLS certificate configuration

```
"tlsCertificateArn": "<The ARN of ACM certificate>"
```

FIPS 140-2

The Federal Information Processing Standard (FIPS) Publication 140-2 is a US and Canadian government standard that specifies the security requirements for cryptographic modules that protect sensitive information.

Note

FIPSEndpoints are only available in North American (NA) Regions.

If you want to use FIPS 140-2 validated cryptographic modules, you can set this using this configuration:

```
"useFipsEndpoint": true,
```

Identity provider

By default, this solution activates basic authentication, enabling access to the Druid web console and API through a username and password. Additionally, you can use user access federation through a third-party identity provider (IdP) that supports OpenID Connect (OIDC) authentication. If you have an existing enterprise Identity Provider, you can integrate it using this CDK configuration.

```
"oidcIdpConfig": {
    "clientId": "<OIDC IDP client ID>",
    "clientSecretArn": "<The ARN of the secret in AWS secret manager which stores the
 client secret and cookiePassphrase.>",
```

```
    "discoveryURI": "<OIDC IDP client discovery URI. e.g. https://dev-
pkaccwqn.us.auth0.com/.well-known/openid-configuration>",
    "groupClaimName": "<OIDC claim name that contains the group name list that the user
 belongs to>",
    "groupRoleMappings": {
        "<groupName>": ["<roleName>", ...],
    }
}
```

The OIDC group mapping facilitates the integration of Role-Based Access Control (RBAC). Users authenticate through an OIDC identity provider, and their group information is extracted from the ID token. This information is then mapped to the corresponding RBAC roles within Druid, governing users' access to resources based on their external OIDC group memberships. This integration ensures that access privileges in Druid align with the user roles defined in the OIDC identity provider. It consists of the following two fields:

- `groupClaimName`: <the OIDC claim name in the ID token that contains the group membership list that the user belongs to.>
- `groupRoleMappings`: <it defines a mapping mechanism that associates OIDC groups with specific RBAC roles in Druid>

The client secret consists of the following two fields:

- `clientSecret`: <the secret code for the OIDC IdP client, which is typically generated by the OIDC IdP.>
- `cookiePassphrase`: <a password that uses a mix of letters and numbers in plain text form.>

Example: Identity provider configuration to federate through Amazon Cognito.

```
"oidcIdpConfig": {
     "clientId": "<OIDC IDP client ID>",
     "clientSecretArn": "arn:aws:secretsmanager:<region>:<account>:secret:<secret-
name>-<random-6-characters>",
     "discoveryURI": "https://cognito-idp.<region>.amazonaws.com/<user pool id>/.well-
known/openid-configuration",
     "groupClaimName": "cognito:groups",
     "groupRoleMappings": {
          "developers": ["administrator"],
          "scrum-masters": ["read"]
```

```
        }
    }
```

Note

You must configure the redirect URI on the IDP side as `https://<druid_domain>/druid-ext/druid-oidc/callback`.

Druid basic configuration

Version

Apache Druid release version (eg. 27.0.0) that you want to run. We recommend using the latest stable Druid version.

```
"druidVersion": "27.0.0",
```

Cluster name

A sequence of ASCII characters that uniquely identifies each Druid cluster. If there are multiple deployments, make sure that you have an unique cluster name for each cluster. The cluster name is appended to the CloudFormation stack name.

```
"druidClusterName": "dev",
```

Operation platform

The intended computing service for hosting a Druid cluster currently has two options available:

- EC2: The solution will be deployed on Amazon EC2 hosts. For more information on how to configure Druid EC2, refer to the Druid EC2 configuration section.

```
"druidOperationPlatform": "ec2",
```

- EKS: The solution will be deployed on Amazon EKS.

Custom IAM policy list (optional)

By default, this solution creates an IAM role that gives Druid components the minimum permissions to function. You can extend this role with additional permissions by using the following configuration:

```
"druidInstanceIamPolicyArns": ["The ARN of custom IAM policy"...]
```

Query concurrency rate limit (optional)

The maximum number of concurrent queries that the Druid cluster can handle. The default value is 100.

- This parameter sets `druid.broker.http.numConnections` on the broker nodes and `druid.server.http.numThreads` on the historical nodes.
- In a cluster with 2 broker nodes and the default value of 100, the `druid.server.http.numThreads` parameter on the historical nodes will be set to 100, while the `druid.broker.http.numConnections` parameter on the broker nodes will be set to 50. This configuration ensures that the `druid.server.http.numThreads` value must be equal to or higher than the sum of `druid.broker.http.numConnections` across all the Brokers in the cluster.

To customize this default configuration, you can override the values using the following configuration:

```
"druidConcurrentQueryLimit": number
```

Extensions

A list of Druid extensions to load into the cluster. To load the core extensions for Druid, you can modify the list by adding or removing extensions from it.

> **Note**
>
> It is the customer's responsibility to ensure the extension's security, performance, and compatibility. To uphold a robust security posture, we strongly advise consistently

monitoring for new releases and promptly applying updates, thereby proactively addressing and mitigating any potential vulnerabilities.

To load any custom extensions that you have developed in-house, make sure that the artifacts for those extensions (such as Java JAR files) are also copied into the `source/lib/docker/extensions` folder. To reduce the configuration overhead, the solution will automatically incorporate the extensions to the user provided extension list:

- druid-oidc
- druid-cloudwatch
- druid-basic-security
- druid-s3-extensions
- postgresql-metadata-storage, and
- simple-client-sslcontext.

Druid extensions example

```
"druidExtensions": [
    "druid-hdfs-storage",
    "druid-kafka-indexing-service",
    "druid-kinesis-indexing-service",
    "druid-avro-extensions",
    "druid-parquet-extensions",
    "druid-protobuf-extensions",
    "druid-orc-extensions"
],
```

Common runtime properties(optional)

By default, the solution features a common.runtime.properties file located within the `source/lib/uploads/config/_common` directory, to cater to the majority of the use cases. However, you can customize and override the settings in this file using the CDK configuration as needed.

Druid common runtime customization configuration example:

```
"druidCommonRuntimeConfig": {
     "druid.startup.logging.logProperties": false
```

```
    }
```

Retention rules (optional)

The data retention policies specify which data to retain and which data to drop from the cluster. The rule chain is evaluated from top to bottom, with the default rule chain always added at the bottom. For more information, refer to the Using rules to drop and retain data guide.

Alternatively, you can use the Druid API or the Druid Web Console to configure the retention rules for Druid.

Druid retention rules example

```
"druidRetentionRules": [
          {
          "type": "loadByPeriod",
          "period": "P1M",
          "includeFuture": true,
          "tieredReplicants": {
              "hot": 3,
              "_default_tier": 1
              }
          },
          {
          "type": "loadForever",
          "tieredReplicants": {
             "hot": 1,
             "_default_tier": 1
              }
          }
]
```

Druid metadata store

In the default configuration, this solution sets up an Amazon Aurora Serverless v1 cluster, using the PostgreSQL-compatible engine as the metadata store for Druid. If you prefer to use an Amazon Aurora cluster as the metadata store, connect to an existing PostgreSQL database, or create the metadata store from a snapshot, you can customize the configuration by applying the following settings:

```
"druidMetadataStoreConfig": {
```

```
        // Metadata store type. Acceptable values include aurora, aurora_serverless, or
 custom.
        "metadataStoreType": "<aurora | aurora_serverless | custom>",

        // The metadata store configuration differs based on the chosen type of
 metadata store.
        "metadataStoreConfig": {
        },

        // Optional. The configuration for the backup plan determines the frequency and
 retention policy for the database snapshots.
        "backupPlanConfig" {

        // The cron expression which defines the snapshot frequency. E.g. cron(0 10 *
 * ? *) which runs at 10:00 am (UTC) every day.
        "scheduleExpression": "<cron expression>",

        // Number of days that the snapshot will be kept
        "deleteAfterDays": number
        }
}
```

Create a new Aurora metadata store (recommended)

This section provides instructions for creating a new Aurora metadata store.

```
"druidMetadataStoreConfig": {
      "metadataStoreType": "aurora",
      "metadataStoreConfig": {
          // Optional. Aurora DB instance class type. eg. t3.large
         "rdsInstanceType": string,
          // Optional. Number of DB instances. Default is 2.
         "rdsInstanceCount": number,
         // Optional. The parameter group name for DB instance.
         "rdsParameterGroupName": string,
         // Optional. Aurora PostgreSQL engine version. The default version is 13.6.
 Please be aware that updating the version of your existing RDS instance will not
 result in data migration.
         "rdsEngineVersion": string
         }
      }
```

Example configuration to create a new Aurora metadata store:

```
"druidMetadataStoreConfig": {
     "metadataStoreType": "aurora",
     "metadataStoreConfig": {
          "rdsInstanceType": "t3.large",
          "rdsInstanceCount": 2,
          "rdsParameterGroupName":
          "aurora-postgresql13",
          "rdsEngineVersion": "13.6"
          }
     }
```

Create a new Aurora metadata store from a snapshot

This section provides instructions on how to create a new Aurora metadata store from a snapshot.

> **Note**
>
> The solution will try to create two magic users, specifically `admin` and `druid_system`, to run internal operations if these users don't exist in the snapshot. If these users already exist in the snapshot, generate a secret in AWS Secrets Manager that includes both the username and password, and then proceed to integrate it into the metadata store configuration. If the secrets are encrypted by a KMS key, ensure that the KMS key permits access from AWS Secrets Manager to enable decryption of the secret.

```
"druidMetadataStoreConfig": {
    "metadataStoreType": "aurora",
    "metadataStoreConfig": {
        // Optional. Aurora DB instance class type. eg. t3.large
        "rdsInstanceType": string,

        // Optional. Number of DB instances. Default is 2.
        "rdsInstanceCount": number,

        // The ARN of RDS snapshot
        "rdsSnapshotArn": string,

        // Optional. The ARN of KMS key which is used to encrypt the database. Required
 if the storage is encrypted with a KMS key.
        "rdsSnapshotEncryptionKeyArn": string,
```

```
        // Optional. Required if the snapshot contains 'druid_system' user.
        "druidInternalUserSecretArn": "<The ARN of secret which stores username and
 password of the druid_system user >",

        // Optional. Required if the snapshot contains 'admin' user.
        "druidAdminUserSecretArn": "<The ARN of secret which stores username and
 password of the admin user >",

        // Optional. The parameter group name for DB instance.
        "rdsParameterGroupName": string,

        // Optional. Aurora PostgreSQL engine version. The default version is 13.6.
 Please be aware that updating the version of your existing RDS instance will not
 result in data migration.
        "rdsEngineVersion": string
    }
}
```

Example configuration to create a metadata store from snapshot:

```
"druidMetadataStoreConfig": {
    "metadataStoreType": "aurora",
    "metadataStoreConfig": {
        "rdsInstanceType": "t3.large",
        "rdsInstanceCount": 2,
        "rdsSnapshotArn": "arn:aws:rds:<region>:<account-id>:cluster-
snapshot:<snapshort-name>",
        "rdsSnapshotEncryptionKeyArn": "arn:aws:kms:<region>:<account-id>:key/<key-
id>",
        "druidInternalUserSecretArn": "arn:aws:secretsmanager:<region>:<account-
id>:secret:<secret-id>",
        "druidAdminUserSecretArn": "arn:aws:secretsmanager:<region>:<account-
id>:secret:<secret-id>",
        "rdsParameterGroupName": "aurora-postgresql13",
        "rdsEngineVersion": "13.6"
    }
}
```

Use your own PostgreSQL database for metadata store

This section explains how to use your own PostgreSQL database for the metadata store.

Note

By default, this solution will enforce TLS for the database connectivity. If you're using non-public certificates, you must ensure the non-public certificates can be validated by the Druid system.

```
"druidMetadataStoreConfig": {
    "metadataStoreType": "custom",
    "metadataStoreConfig": {
        // The endpoint address of database
        "databaseUri": string,

        // Optional. The listening port of database. Default to be 5432.
        "databasePort": number,

        // Optional. Database name. Default to be DruidMetadata.
        "databaseName": string,

        // The ARN of secret in secrets manager which stores the administrative
 username and password in JSON format
        // The secret must contain fields namely "username" and "password"
        "databaseSecretArn": string
    }
}
```

Example configuration to connect to an external metadata store:

```
"druidMetadataStoreConfig": {
    "metadataStoreType": "custom",
    "metadataStoreConfig": {
        "databaseUri": "druidec2stack-dev296-druidmetadataconstructaurorac-xxx.cluster-
xxx.us-east-1.rds.amazonaws.com",
        "databasePort": 5432,
        "databaseName": "DruidMetadata",
        "databaseSecretArn": "arn:aws:secretsmanager:<region>:<account>:secret:<secret-
id>"
    }
}
```

Druid deep storage

By default, this solution will set up a bucket in Amazon S3 for deep storage. To use your own bucket for deep storage, you can override this default configuration using the following configuration:

```
"druidDeepStorageConfig": {
    // The ARN of S3 bucket to be used as deep storage
    "bucketArn": string,
    // The bucket prefix to be used to store segments
    "bucketPrefix": string,
    // Optional when using SSE-S3. The KMS key to be used to encrypt the data.
    "bucketEncryptionKeyArn": string
}
```

Example deep storage configuration:

```
"druidDeepStorageConfig": {
    "bucketArn": "arn:aws:s3:::<bucket-id>",
    "bucketPrefix": "druid/segments",
    "bucketEncryptionKeyArn": "arn:aws:kms:<region>:<account-id>:key/<key-id>"
}
```

Druid EC2 configuration

Auto scaling group configuration

This section provides instructions and details about the Auto scaling group configuration for a variety of Druid node types, including master, query, data, historical, middleManager, and ZooKeeper.

This solution also fully supports the service tiering functionalities provided by Druid, offering enhanced resource management and optimization. The service tiering feature allows you to create distinct groups of Historicals and Brokers, each responsible for managing queries based on the segments and resource requirements of the query.

It covers the following fields for each node type:

```
// Druid node type. Valid values include data, master, query, and zookeeper. In case of
 service tiering, it also supports data_<tier>, historical_<tier>, middleManager_<tier>
```

```
"<Druid node type>": {
      // Minimum number of EC2 instances in the auto scaling group
      "minNodes": number,

      // Optional, maximum number of EC2 instances in the auto scaling group
      "maxNodes": number,

      // Optional. EBS volume size in GB for the root file system of EC2 instance.
      "rootVolumeSize": number,

      // Optional. The size of EBS volume (GB) utilized for segment caching on
 historical node. Required if not using storage optimized instance types.
      "segmentCacheVolumeSize": number,

      // EC2 instance type. eg. m5.2xlarge.
      "instanceType": string

      }
```

Rolling update policy (optional)

A rolling update is a deployment strategy that incrementally replaces previous versions of an application with new versions of an application by completely replacing the infrastructure on which the application is running.

It minimizes the impact on application availability and ensures a smooth transition to the new version.

The solution operates on a default process of sequentially replacing the Druid EC2 instances in a specific order: data, query, and master. This replacement process involves stopping the previous instance initially, followed by the creation of a new instance.

Once the new instance is confirmed to be in a healthy state, the process proceeds to replace the next instance in the specified order. In the event that an instance is deemed unhealthy during the update process, the system will initiate a rollback to the previous version.

You can customize and override the batch size using the following configuration:

```
// Optional. The rollingUpdatePolicy defines how an Auto Scaling group resource is
 updated.
"rollingUpdatePolicy": {
    // Optional. Number of instances that is going to be replaced in each batch.
 Default 1.
```

```
    "maxBatchSize": number
},
```

Auto scaling policy (optional)

An Auto scaling policy consists of predefined rules and parameters that dictate how the scaling process should occur. These rules are defined by the user based on specific metrics, such as CPU utilization, network traffic, or application-specific performance metrics. In addition to dynamic auto scaling based on Scalable metrics, it also offers schedule-based auto scaling.

```
"autoScalingPolicy": {
    "cpuUtilisationPercent": number between [1, 100],
    "schedulePolicies": [
        {
            // A UTC timestamp represented by a cron expression.
            // eg. "0 5 * * *" means 5:00 AM in UTC time
            "scheduleExpression": "<cron expression>",
            "minNodes": number,
            "maxNodes": number
        }
    ]
}
```

Runtime configuration (optional)

The solution comes pre-configured with Druid, which suits most use cases. You can customize the Druid configuration (runtime.properties) using the following settings:

```
"runtimeConfig": {
    // Druid process type depending on the node type. Valid values include coordinator,
 overlord, middleManager, historical, router, and broker.
    "<Druid process type>": {
        // The key name of Druid configuration item. eg. druid.server.http.numThreads
        "<druid configuration key>": "<druid configuration value>"
        ...
    },
}
```

Runtime configuration customization example:

```
"runtimeConfig": {
    "historical": {
```

```
        "druid.server.http.numThreads": 300
    },
    "middleManager": {
        "druid.server.http.numThreads": 200
    }
}
```

Druid EC2 configuration example

```
"druidEc2Config": {
    "master": {
        "minNodes": 2,
        "maxNodes": 2,
        "rootVolumeSize": 50,
        "instanceType": "m5.2xlarge"
    },
    "query": {
        "minNodes": 2,
        "maxNodes": 3,
        "rootVolumeSize": 50,
        "instanceType": "m5.2xlarge",
        "autoScalingPolicy": {
            "cpuUtilisationPercent": 60
        }
    },
    "data": {
        "minNodes": 3,
        "maxNodes": 3,
        "rootVolumeSize": 200,
        "segmentCacheVolumeSize": 500,
        "instanceType": "m5d.2xlarge"
    },
    "zookeeper": {
        "minNodes": 3,
        "maxNodes": 3,
        "rootVolumeSize": 50,
        "instanceType": "m6g.xlarge"
    }
}
```

Example: Druid deployment using EC2 for a medium profile

```
{
```

```
    "vpcCidr": "xx.xxx.0.0/16",
    "route53HostedZoneName": "<route53 hosted zone name>",
    "route53HostedZoneId": "<route 53 hosted zone ID>",
    "druidDomain": "<Domain where Druid is hosted>",
    "internetFacing": true,
    "druidClusterName": "ec2",
    "druidVersion": "27.0.0",
    "druidOperationPlatform": "ec2",
    "retainData": true,
    "druidExtensions": [
        "druid-hdfs-storage",
        "druid-kafka-indexing-service",
        "druid-datasketches",
        "druid-s3-extensions",
        "postgresql-metadata-storage",
        "druid-kinesis-indexing-service",
        "druid-avro-extensions",
        "druid-parquet-extensions",
        "druid-protobuf-extensions",
        "druid-cloudwatch",
        "druid-orc-extensions",
        "druid-basic-security",
        "druid-pac4j",
        "simple-client-sslcontext"
    ],
        "druidEc2Config": {
            "master": {
                "minNodes": 3,
                "maxNodes": 3,
                "rootVolumeSize": 50,
                "instanceType": "m6g.xlarge"
            },
            "query": {
                "minNodes": 3,
                "maxNodes": 3,
                "rootVolumeSize": 50,
                "instanceType": "m6g.xlarge"
            },
            "data": {
                "minNodes": 3,
                "maxNodes": 3,
                "rootVolumeSize": 200,
                "segmentCacheVolumeSize": 500,
                "instanceType": "m6g.2xlarge"
```

```
            },
            "zookeeper": {
                "minNodes": 3,
                "maxNodes": 3,
                "instanceType": "t4g.medium"
            }
        },
        "druidMetadataStoreConfig": {
            "metadataStoreType": "aurora",
            "metadataStoreConfig": {
                "rdsInstanceType": "t3.medium",
                "rdsInstanceCount": 3
            }
        }
}
```

You can find more configuration examples about EC2 configuration in the `source/quickstart` directory.

Druid EKS configuration

The solution will set up a new Amazon EKS cluster to serve as the hosting environment for Druid.

This section provides instructions on the configuration for the EKS cluster, including node group settings, involves the provisioning of four distinct node groups during deployment: master, query, data, and zookeeper.

EKS cluster configuration

```
"druidEksConfig": {
    // Kubernetes API server endpoint access. Valid values include PUBLIC, PRIVATE, and
 PUBLIC_AND_PRIVATE
    "endpointAccess": "PUBLIC",

    // The ARN of IAM role for the EKS cluster master
    "clusterMasterPrincipalArn": "<the ARN of IAM role for the cluster master>",

    // The underlying capacity provider for the EKS cluster. Valid values include ec2
 and fargate.
    "capacityProviderType": "<ec2 or fargate>",

    // The capacity provider configuration. It differs based on the chosen capacity
 provider.
```

```
    "capacityProviderConfig": {
    }
}
```

EC2 capacity provider configuration

This is required if `capacityProviderType` is ec2.

```
// Node group name, valid values include master, query, data, and zookeeper. In case of
 service tiering, data_<tier> is also supported.

"<nodeGroupName>": {
    // Minimum number of EC2 instances in the node group
    "minNodes": number,

    // Optional. Maximum number of EC2 instances in the node group
    "maxNodes": number,

    // Optional. EBS volume size in GB for the root file system of EC2 instance.
    "rootVolumeSize": number,

    // EBS volume size in GB. Persistent volume for middleManager pods. Only required
 for middleManager pods.
    "taskCacheVolumeSize": number,

    // EBS volume size in GB. Persistent volume for historical pods. Only required for
 historical pods.
    "segmentCacheVolumeSize": number,

    // EC2 instance type for the node group.
    "instanceType": string
},
```

Fargate capacity provider configuration

This is required if `capacityProviderType` is Fargate.

```
// Druid process name, valid values include coordinator, overlord, middleManager,
 historical, router, broker, and zookeeper.

"<druidProcessName>": {
    // Minimum number of kubernetes pods
```

```
    "minNodes": number,

    // Optional. Maximum number of kubernetes pods
    "maxNodes": number,

    // Number of vCPUs to assign to each pod. eg. 2 means 2 vCPUs.
    "cpu": number,

    // The amount of memory to assign to each pod. eg. 4Gi means 4GB memory.
    "memory": string
}
```

Druid EKS configuration example (EC2 capacity provider)

```
"druidEksConfig": {
    "endpointAccess": "PUBLIC",
    "clusterMasterPrincipalArn": "arn:aws:iam::<account id>:role/<role name>",
    "capacityProviderType": "ec2",
    "capacityProviderConfig": {
        "master": {
            "minNodes": 2,
            "maxNodes": 2,
            "rootVolumeSize": 50,
            "instanceType": "m5.2xlarge"
        },
        "query": {
            "minNodes": 2,
            "maxNodes": 2,
            "rootVolumeSize": 50,
            "instanceType": "m5.2xlarge"
        },
        "data": {
            "minNodes": 3,
            "maxNodes": 3,
            "rootVolumeSize": 50,
            "taskCacheVolumeSize": 300,
            "segmentCacheVolumeSize": 500,
            "instanceType": "m5.2xlarge"
        },
        "zookeeper": {
            "minNodes": 3,
            "maxNodes": 3,
            "rootVolumeSize": 50,
```

```
            "instanceType": "m5.large"
        }
    }
}
```

Example: Druid deployment using EKS for a medium profile

```
{
    "vpcCidr": "xx.xxx.0.0/16",
    "route53HostedZoneName": "<route53 hosted zone name>",
    "route53HostedZoneId": "<route 53 hosted zone ID>",
    "druidDomain": "<Domain where Druid is hosted>",
    "internetFacing": true,
    "druidClusterName": "eks",
    "retainData": false,
    "druidVersion": "27.0.0",
    "druidOperationPlatform": "eks",
    "druidExtensions": [
        "druid-hdfs-storage",
        "druid-kafka-indexing-service",
        "druid-datasketches",
        "druid-s3-extensions",
        "postgresql-metadata-storage",
        "druid-kinesis-indexing-service",
        "druid-avro-extensions",
        "druid-parquet-extensions",
        "druid-protobuf-extensions",
        "druid-cloudwatch",
        "druid-orc-extensions",
        "druid-basic-security",
        "druid-pac4j"
    ],
    "druidEksConfig": {
        "endpointAccess": "PUBLIC",
        "clusterMasterPrincipalArn": "arn:aws:iam::<account id>:role/<role name>",
        "capacityProviderType": "ec2",
        "capacityProviderConfig": {

"master": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "instanceType": "m5.xlarge"
                },
```

```
                "query": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "instanceType": "m5.xlarge"
                },
                "data": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "taskCacheVolumeSize": 300,
                    "segmentCacheVolumeSize": 500,
                    "instanceType": "m5.large"
                },
                "zookeeper": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "rootVolumeSize": 50,
                    "instanceType": "t3.medium"
                }
            }
        },
        "druidMetadataStoreConfig": {
            "metadataStoreType": "aurora",
            "metadataStoreConfig": {
                "rdsInstanceType": "t3.medium",
                "rdsInstanceCount": 3
            }
    }
}
```

Example: Druid configuration using EKS and Fargate, for a medium profile

```
{
    "vpcCidr": "xx.xxx.0.0/16",
    "route53HostedZoneName": "<route53 hosted zone name>",
    "route53HostedZoneId": "<route 53 hosted zone ID>",
    "druidDomain": "<Domain where Druid is hosted>",
    "internetFacing": true,
    "druidVersion": "27.0.0",
    "druidClusterName": "fargate",
    "druidOperationPlatform": "eks",
    "retainData": false,
    "druidExtensions": [
        "druid-hdfs-storage",
```

```
        "druid-kafka-indexing-service",
        "druid-datasketches",
        "druid-s3-extensions",
        "postgresql-metadata-storage",
        "druid-kinesis-indexing-service",
        "druid-avro-extensions",
        "druid-parquet-extensions",
        "druid-protobuf-extensions",
        "druid-cloudwatch",
        "druid-orc-extensions",
        "druid-basic-security",
        "druid-pac4j"
    ],

"druidEksConfig": {
            "endpointAccess": "PUBLIC",
            "clusterMasterPrincipalArn": "arn:aws:iam::<account id>:role/<role name>",
            "capacityProviderType": "fargate",
            "capacityProviderConfig": {
                "historical": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "cpu": 4,
                    "memory": "16Gi"
                },
                "middleManager": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "cpu": 4,
                    "memory": "16Gi"
                },
                "broker": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "cpu": 2,
                    "memory": "8Gi"
                },
                "router": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "cpu": 2,
                    "memory": "8Gi"
                },
                "coordinator": {
```

```
                    "minNodes": 1,
                    "maxNodes": 1,
                    "cpu": 2,
                    "memory": "8Gi"
                },
                "overlord": {
                    "minNodes": 1,
                    "maxNodes": 1,
                    "cpu": 2,
                    "memory": "8Gi"
                },
                "zookeeper": {
                    "minNodes": 3,
                    "maxNodes": 3,
                    "cpu": 2,
                    "memory": "4Gi"
                }
            }
        },
        "druidMetadataStoreConfig": {
            "metadataStoreType": "aurora",
            "metadataStoreConfig": {
                "rdsInstanceType": "t3.medium",
                "rdsInstanceCount": 3
            }
    }
}
```

You can find more examples about EKS configuration in the `source/quickstart` directory.

Monitor the solution

Follow the step-by-step instructions in this section for:

- Monitoring cost and manage portfolio using Service Catalog AppRegistry, and
- Monitoring solution performance and operational data using Amazon CloudWatch.

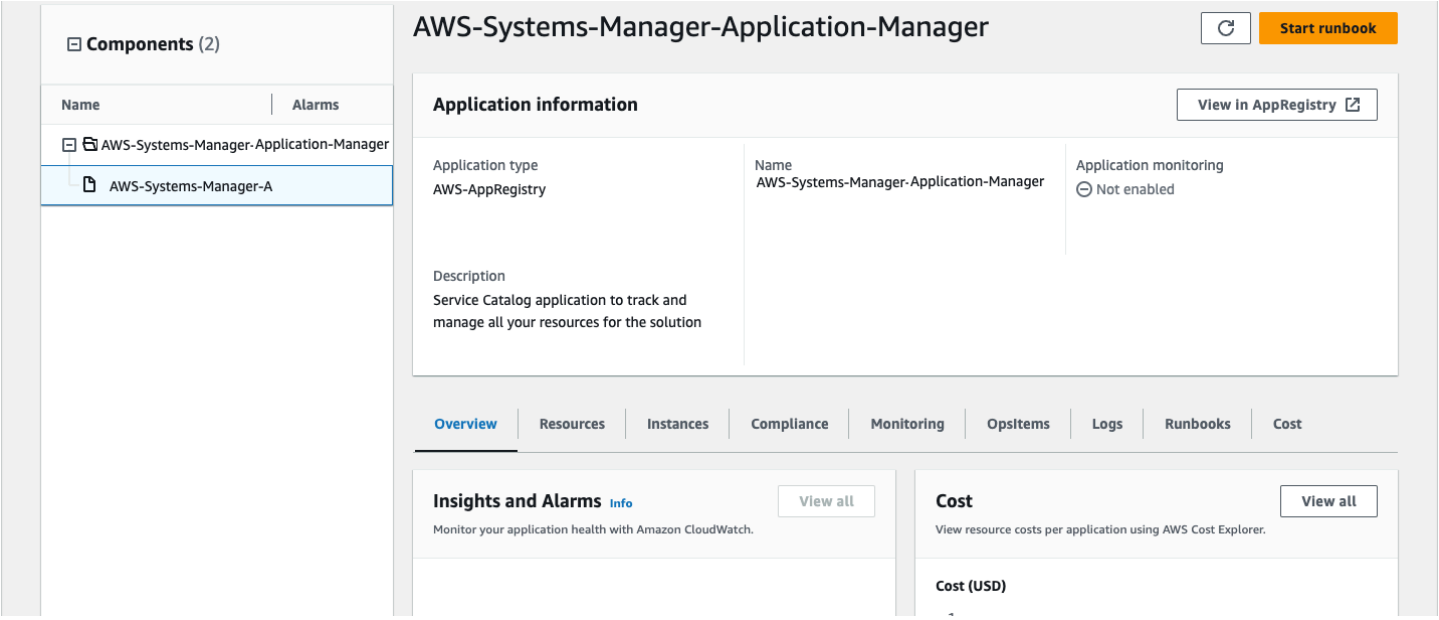
Monitoring cost and portfolio with Service Catalog AppRegistry

This solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both Service Catalog AppRegistry and AWS Systems Manager Application Manager.

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.
- View operations data for the resources of this solution (such as deployment status, CloudWatch alarms, resource configurations, and operational issues) in the context of an application.

  The following figure depicts an example of the application view for the solution stack in Application Manager.

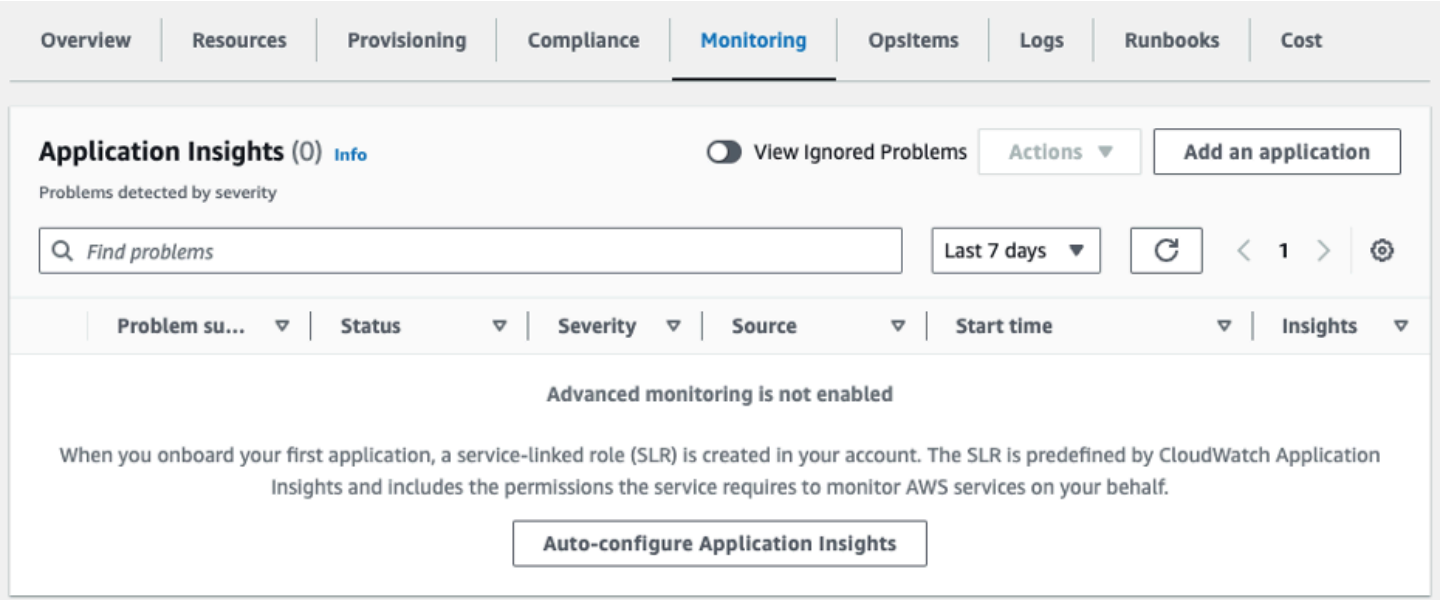


Solution stack in Application Manager

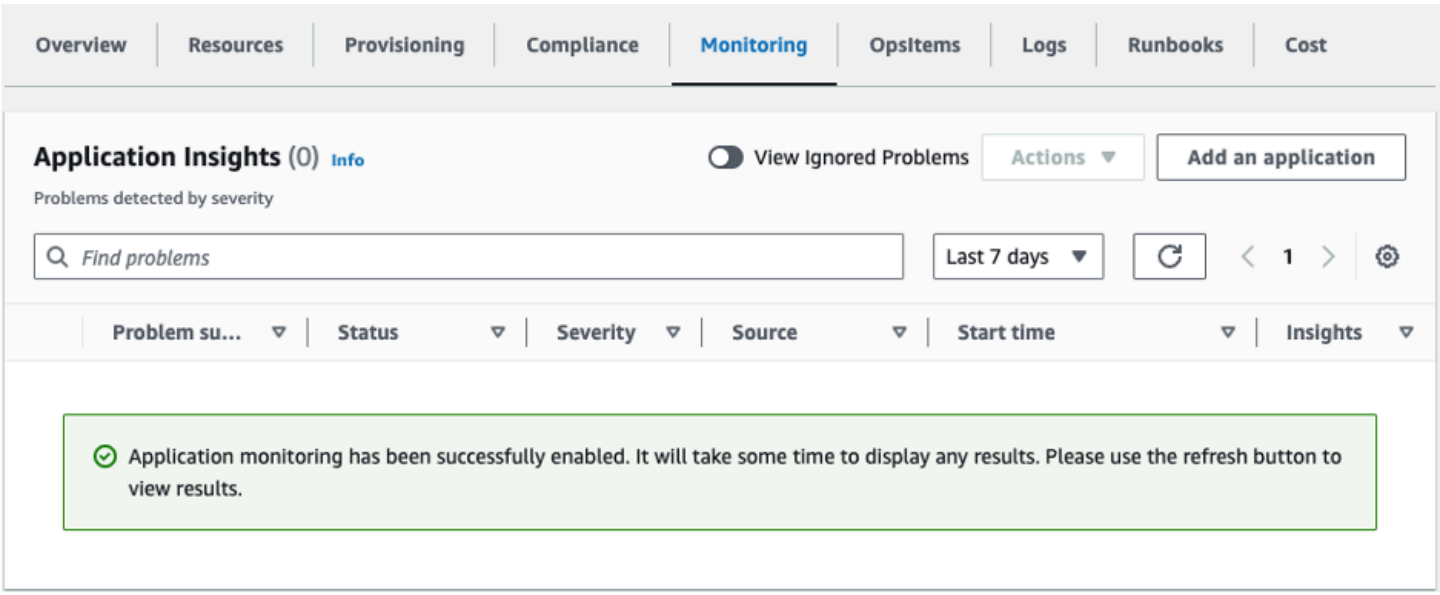
Activate CloudWatch Application Insights

1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, search for the application name for this solution and select it.

   The application name will have App Registry in the **Application Source** column, and will have a combination of the solution name, Region, account ID, or stack name.
4. In the **Components** tree, choose the application stack you want to activate.
5. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Insights**.



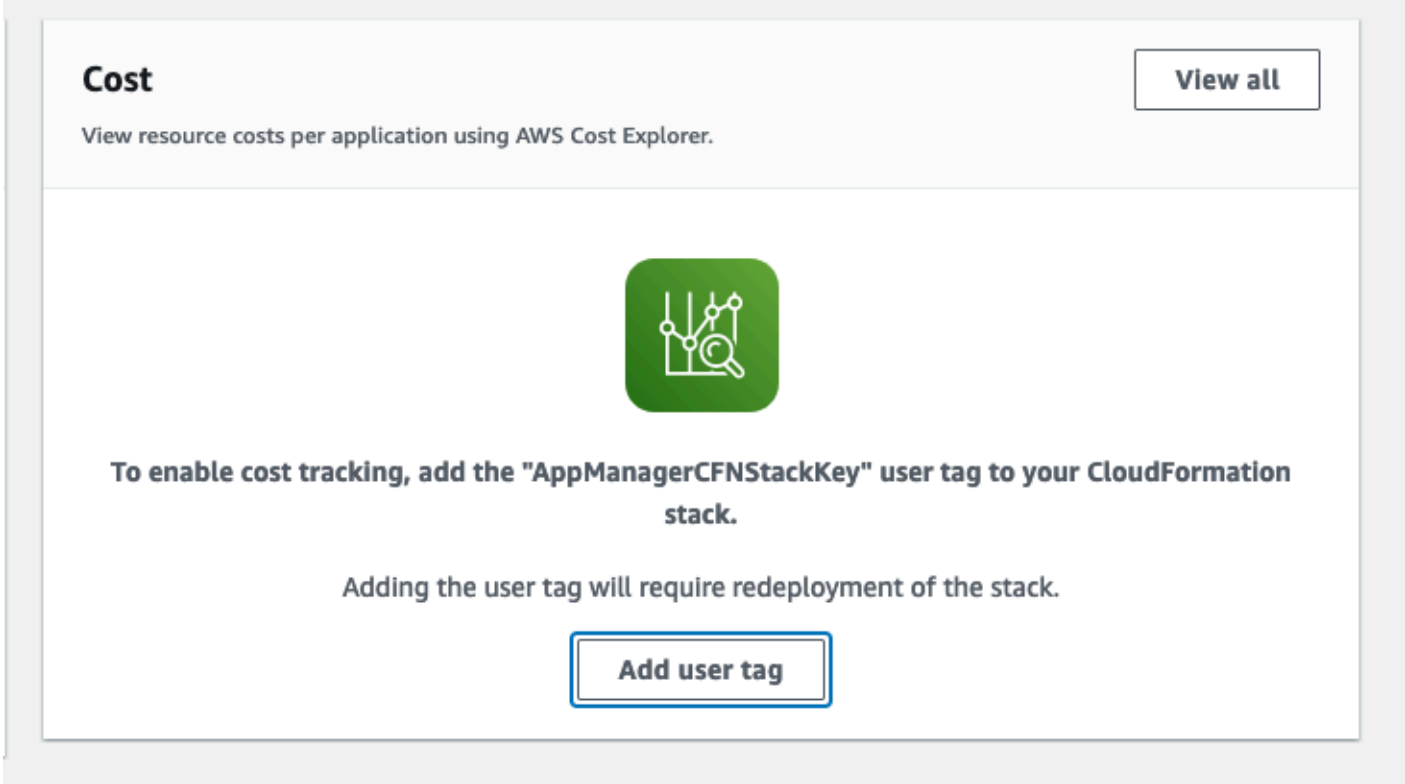
Monitoring for your applications is now activated and the following status box appears:



Confirm cost tags associated with the solution

After you activate cost allocation tags associated with the solution, you must confirm the cost allocation tags to see the costs for this solution. To confirm cost allocation tags:

1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose the application name for this solution and select it.
4. In the **Overview** tab, in **Cost**, select **Add user tag**.



5. On the **Add user tag** page, enter `confirm`, then select **Add user tag**.

The activation process can take up to 24 hours to complete and the tag data to appear.

Activate cost allocation tags associated with the solution

After you confirm the cost tags associated with this solution, you must activate the cost allocation tags to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization.

To activate cost allocation tags:

1. Sign in to the AWS Billing and Cost Management and Cost Management console.
2. In the navigation pane, select **Cost Allocation Tags**.
3. On the **Cost allocation tags** page, filter for the `AppManagerCFNStackKey` tag, then select the tag from the results shown.
4. Choose **Activate**.

AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time.

1. Sign in to the AWS Cost Management console.
2. In the navigation menu, select **Cost Explorer** to view the solution's costs and usage over time.

Monitoring performance and operations with Amazon CloudWatch

The solution captures all the Druid data logs in Amazon CloudWatch for monitoring purposes. This includes alarms, logs and a dashboard for reporting purposes.

Amazon CloudWatch gives you an application-level view into this solution and its resources so that you can:

- Monitor alarms, logs for your deployed clusters, and metrics associated with this solution from a central location.
- View operations data for the solution's AWS resources (such as deployment status, Amazon CloudWatch alarms, resource configurations, and operational issues) in the context of an application.

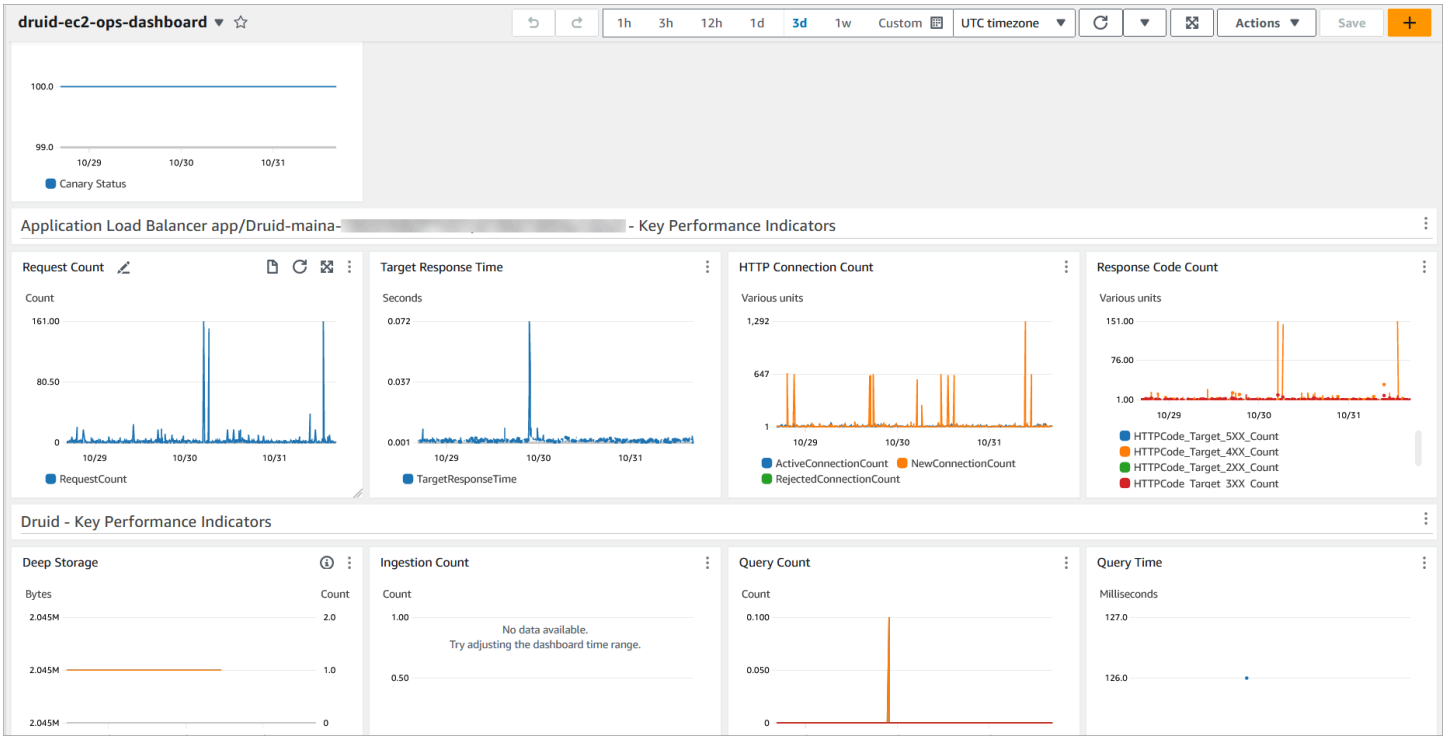
Sign in to the AWS Management Console, and navigate to **CloudWatch**. Use the left hand navigation to view this data for your Druid deployment.

> **Note**
>
> You must activate the CloudWatch Application Insights before you can use CloudWatch to monitor any alarms, logs, or dashboards for the solution. For more information, refer to the Activate CloudWatch Application Insights section.

Dashboard

1. From the left, select **CloudWatch > Dashboards**.

2. On the **Custom Dashboards** tab, click to select the dashboard you want to view. For example, `druid-ec2-`*`custom-name`*`-ops-dashboard`.



CloudWatch dashboard for Scalable Analytics using Apache Druid on AWS

The dashboard provides the following information for your Druid deployment.

Item	Description
Canary status	Displays the availability and latency of your web services and allows you to discover and troubleshoot any issues.
Application Load Balancer (ALB) – Key Performance Indicators	Provides the following data in relation to your ALB application: • Request Count • Target Response Time • HTTP Connection Count • Response Code Count

Item	Description
Druid – Key Performance Indicators	Provides the following data in relation to Druid core parameters: • Deep Storage • Ingestion Count • Query Count • Query Time
Druid ZooKeeper – Key Performance Indicators	Displays the following data in relation to the Zookeeper cluster state management: • CPU Utilization (%) • Network In/Out (bytes) • Memory Utilization (%) • Disk Utilization (%)
Druid data	Displays the following data in relation to the Druid ingestion jobs and queryable data: • CPU Utilization (%) • Network In/Out (bytes) • Memory Utilization (%) • Disk Utilization (%)
Druid query	Displays the following data in relation to the Druid queries: • CPU Utilization (%) • Network In/Out (bytes) • Memory Utilization (%) • Disk Utilization (%)

Item	Description
Druid master	Displays the following data in relation to the Druid data ingestion and availability: • CPU Utilization (%) • Network In/Out (bytes) • Memory Utilization (%) • Disk Utilization (%)
Aurora Cluster	Provides the following data in relation to the Druid databases: • CPU Utilization (%) • Database connections • Throughput • Free Memory

Alarms

Amazon CloudWatch provides detailed alarms for your Druid deployment, and displays the different states, conditions, and relevant actions associated with these alarms. To view the **Alarms** page, from the left, select **CloudWatch > Alarms.**

For more information about alarms, alarm states, actions, and configuring alarms, refer to the Amazon CloudWatch alarms page.

Logs

You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, and other sources. CloudWatch Logs allow you to centralize the logs from all of your Druid solution components, applications, and AWS services, in a single, highly scalable service.

To view logs for the Druid deployment, from the left, select **CloudWatch > logs > Log groups.**

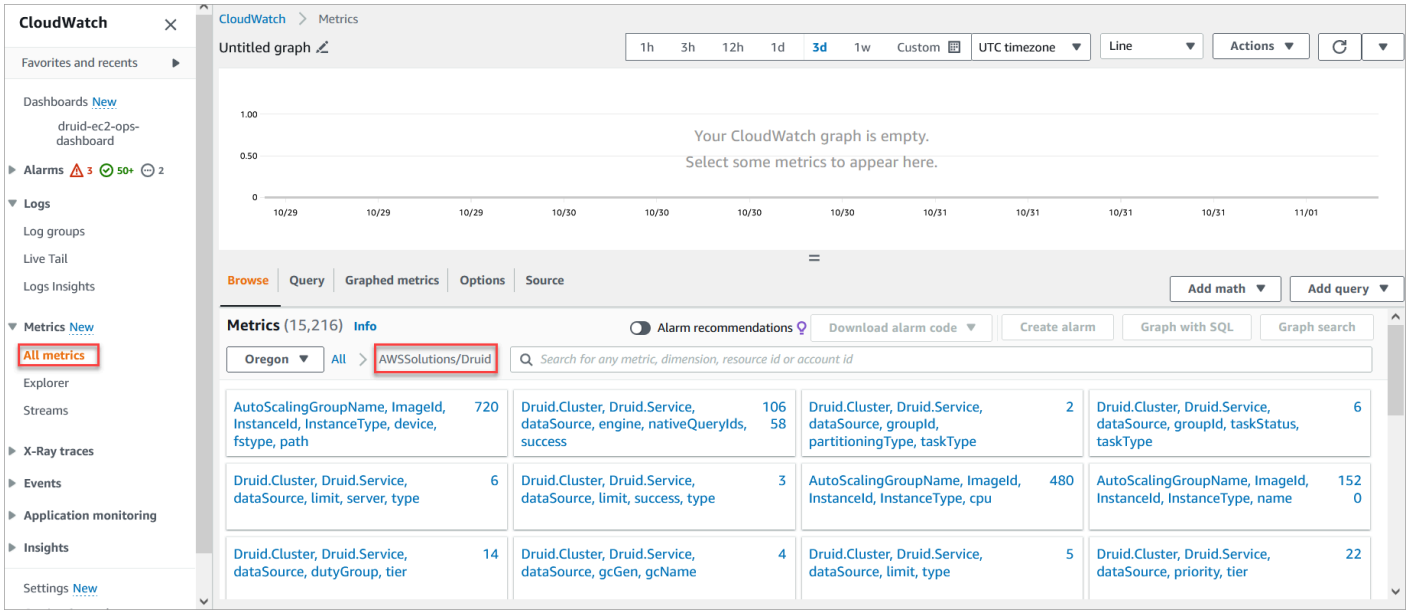
For more information about working with log groups and log streams, refer to the AWS CloudWatch logs page.

Metrics

In addition to logs and alarms, CloudWatch allows to view, and analyze data about the performance of your Druid deployment via custom namespaces.

To view metrics for the Druid deployment:

1. From the left, select **CloudWatch > Metrics > All metrics**.
2. On the Metrics page, under Custom namespaces, select `AWSSolutions/Druid`. This will display all metrics for the Druid deployment.



Metrics for Scalable Analytics using Apache Druid on AWS

3. Choose to select a relevant dimension to view additional information. The dimensions page provides a breakdown of individual Druid services, source, query, and other metrics.

Troubleshooting

This section provides known issue resolution when deploying the solution.

If these instructions don't address your issue, see the Contact AWS Support section for instructions on opening an AWS Support case for this solution.

Problem: Deletion of the solution stack fails

When deleting the solution stack, you may see the following error from AWS CloudFormation console.

Status	Status reason
DELETE_FAILED	The following resource(s) failed to delete: [druidbaseinfradruidvpcinnervpcPrivateSubnetSubnet2Subnet293830FE, druidbaseinfradeepstoragebucket547BF480, druidbaseinfrabootstraps3bucketinstallationCFF3AF5D, druidbaseinfradruidvpcinnervpcPrivateSubnetSubnet1Subnet828E519C, druidbaseinfradruidvpcinnervpcPrivateSubnetSubnet3Subnet70682654, canarySecurityGroupBAAFA6CF].

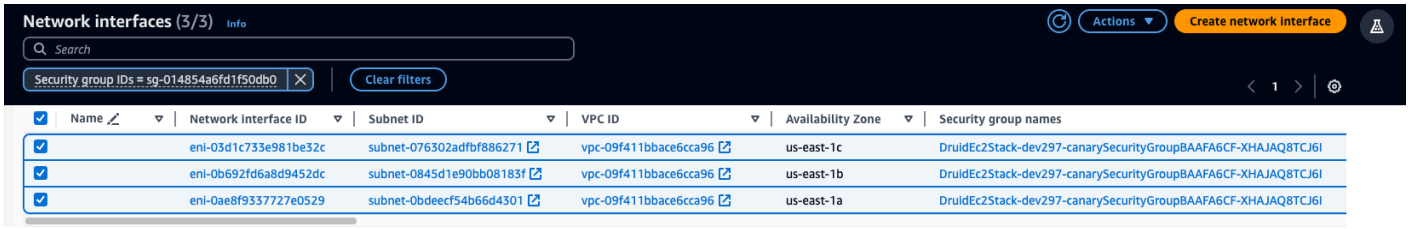
Delete solution stack - error message

Resolution

This issue occurs because the resources created by Amazon CloudWatch Synthetics are not released correctly, which is tracked in this GitHub issue.

To delete the stack completely:

1. In the primary account, navigate to the Amazon EC2 console.
2. Select the Network Interfaces that are linked to `canarySecurityGroup`.



List of network interfaces

3. Select **Actions**, and **Delete**.
4. Navigate to the CloudFormation console, and delete the stack again.

Problem: Deployment fails due to CloudFormation FAILURE signals

When deploying the solution, the deployment may fail due to CloudFormation FAILURE signals.

2023-11-08 16:04:36 UTC+1100	DruidEc2Stack-dev298	UPDATE_ROLLBACK_IN_PROGRESS	The following resource(s) failed to create: [historicalasgASG3DE22B4E].
2023-11-08 16:04:33 UTC+1100	historicalasgASG3DE22B4E	CREATE_FAILED	Received 2 FAILURE signal(s) out of 3. Unable to satisfy 100% MinSuccessfulInstancesPercent requirement
2023-11-08 16:04:32 UTC+1100	historicalasgASG3DE22B4E	CREATE_IN_PROGRESS	Received FAILURE signal with UniqueId i-063fae5ecc7ea2aff
2023-11-08 16:04:32 UTC+1100	historicalasgASG3DE22B4E	CREATE_IN_PROGRESS	Received FAILURE signal with UniqueId i-00894d2a7dfa24319
2023-11-08 15:47:59 UTC+1100	historicalasgASG3DE22B4E	CREATE_IN_PROGRESS	Resource creation Initiated
2023-11-08 15:47:58 UTC+1100	historicalasgASG3DE22B4E	CREATE_IN_PROGRESS	-

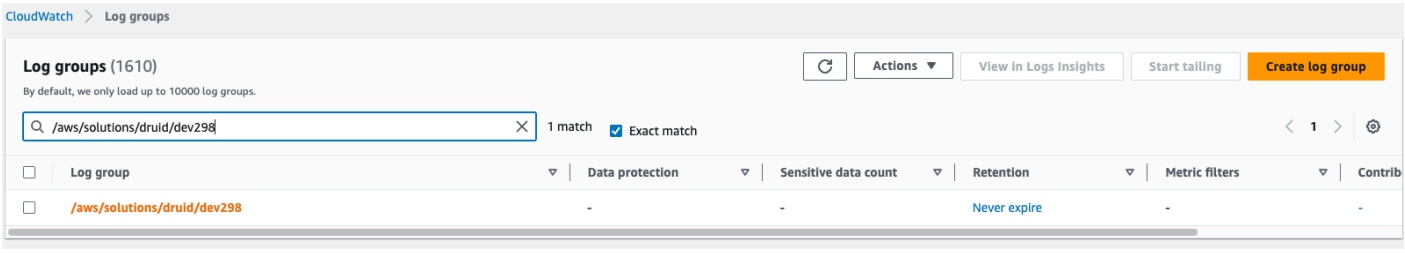
Solution deployment - error message

Resolution

The issue arises from the fact that the corresponding Druid process is unable to transition into service.

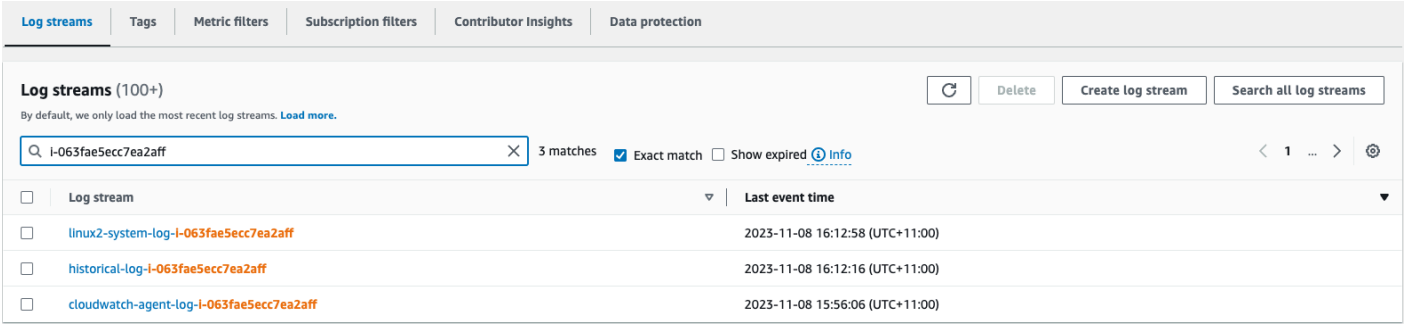
To identify the underlying cause:

1. In the primary account, navigate to the CloudWatch console.
2. Choose the log group named `/aws/solutions/druid/<cluster name>`. The provided screenshot displays the log group for the example cluster **dev298**.



CloudWatch - Log groups

3. Select the log stream by searching for the instance ID found in the CloudFormation console.



CloudWatch - Log streams

4. Inspect the relevant logs to determine the reason the instance does not start correctly.

Problem: Deployment fails due to unsupported RDS engine version

You might encounter the error `Cannot find version for aurora-postgresql` during deployment of the solution in certain regions. This occurs because the solution deploys RDS engine version 13.9 by default, and this version may not be available in the specified region.

Resolution

1. Search the RDS engine versions with the following AWS CLI.

```
aws rds describe-db-engine-versions --engine aurora-postgresql --query '*[].
[EngineVersion]' --output text --region <region>
```

2. Select an RDS engine version (it is advisable to choose the closest version to 13.9) from the provided list.
3. Configure the `druidMetadataStoreConfig` by using the selected RDS engine version.
4. Deploy the solution again.

Contact AWS Support

If you have AWS Developer Support, AWS Business Support, or AWS Enterprise Support, you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

Create case

1. Sign in to Support Center.
2. Choose **Create case**.

How can we help?

1. Choose **Technical**.
2. For **Service**, select **Solutions**.
3. For **Category**, select **Other Solutions**.
4. For **Severity**, select the option that best matches your use case.
5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

Additional information

1. For **Subject**, enter text summarizing your question or issue.
2. For **Description**, describe the issue in detail.
3. Choose **Attach files**.
4. Attach the information that AWS Support needs to process the request.

Help us resolve your case faster

1. Enter the requested information.
2. Choose **Next step: Solve now or contact us**.

Solve now or contact us

1. Review the **Solve now** solutions.
2. If you can't resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

Uninstall the solution

You can uninstall the solution by directly deleting the stacks from the AWS CloudFormation console.

To uninstall the solution, delete the stacks from the **AWSCloudFormation** console,

1. Go to the AWS CloudFormation console, and
2. Find and delete all the stacks with the prefix `DruidEc2Stack` or `DruidEksStack`.

Alternatively, to uninstall the solution, navigate to the `source` folder, and run `npm run cdk destroy`.

Use the solution

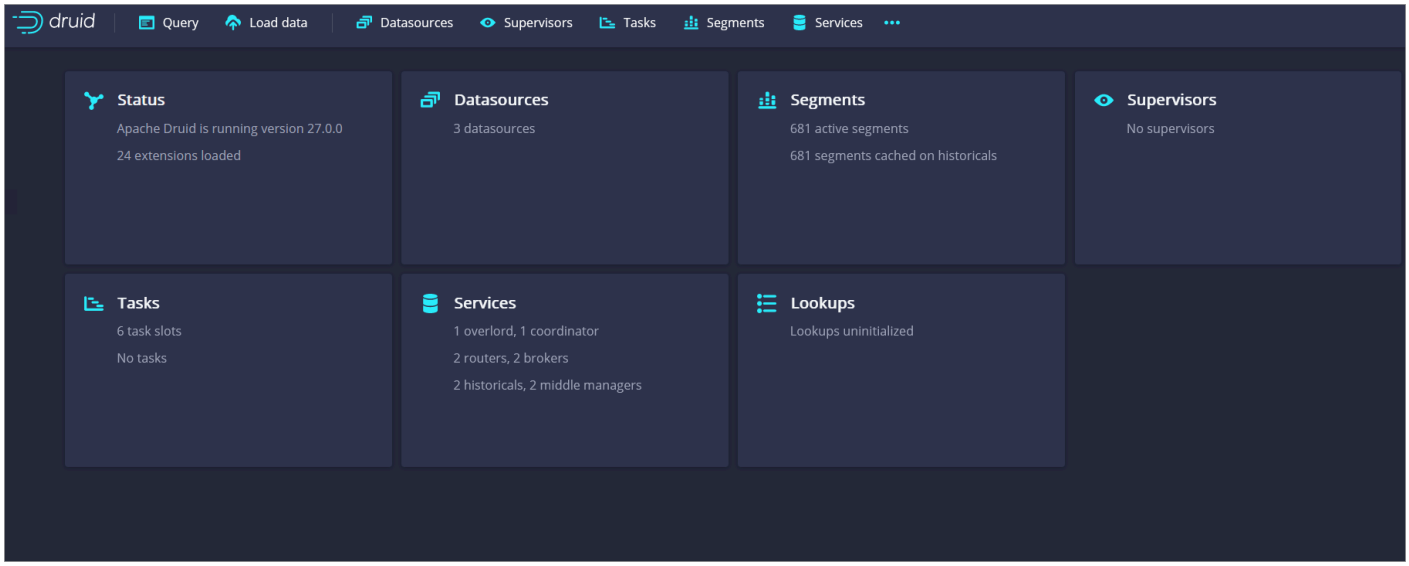
This section provides a user guide for using the Scalable Analytics using Apache Druid on AWS solution.

Access the Druid web console

1. Using the deployment output, get the website URL starting with `druid-base-url`.
2. Open the URL in your browser (we recommend using Chrome). You will be redirected to the sign-in page for the username and password.

> **Note**
>
> During the deployment process, an administrative user account is created with the username admin. To retrieve the password for this account from AWS Secrets Manager, search for the entry with a description *Administrator user credentials* for Druid cluster. You have the option to use the administrative user account to sign in, or create a new user account with reduced access permissions.

3. After signing in, the Apache Druid web console is displayed. The web console displays the Druid components deployed in your AWS account using the configuration that you used during the deployment process.



Apache Druid web console

For more information on how to ingest external data, refer to the Apache Druid tutorial documentation.

Developer guide

This section provides the source code for the solution.

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others.

The Scalable Analytics using Apache Druid on AWS templates are generated using the AWS Cloud Development Kit (AWS CDK). Refer to the README.md file for additional information.

CDK deployment

For configuring advanced settings or customizing the solution, it is recommended to download the source code from the GitHub source repository and build and deploy with AWS CDK. For more information, refer to the README.md file.

Reference

This section includes information about an optional feature for collecting unique metrics for this solution and a list of builders who contributed to this solution.

Anonymized data collection

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID** - The AWS solution identifier
- **Unique ID (UUID)** - Randomly generated, unique identifier for each Scalable Analytics using Apache Druid on AWS deployment
- **Timestamp** - Data-collection timestamp

AWS owns the data gathered though this survey. Data collection is subject to the Privacy Notice.

Opt out of operational metrics collection

To opt out of this feature, for the solution parameters, in cdk.json, set the **sendAnonymousData** parameter to No.

Contributors

- Frank Cao
- Van Vo Thanh
- Hafiz Saadullah
- Marc Teichtahl
- Swapnil Ogale
- APJ Solutions Engineering team
- Jason Wreath
- James Ousby

- Matt Jobson
- Chris Merrigan
- Gaurav Bhatnagar
- Verinder Singh
- Steven Hogarth

Revisions

For more information, see the CHANGELOG.md file in the GitHub repository.

Date	Change
January 2024	v1.0.0: Initial release
July 2024	Release version 1.0.1: Patch release • Fixed endpoints for CloudWatch agent to send logs/metrics when FIPS endpoints are enabled. • Fixed issue with hardcoded folders for middle manager task and processing. • Updated some package versions to resolve security vulnerabilities. • Updated pac4j code involved in OIDC/Cognito authentication. • Druid version updated to 29.0.1. • CDK version updated to 2.146.0. For more information about the changes, refer to the CHANGELOG.md file in the GitHub repository.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The Scalable Analytics using Apache Druid on AWS solution is licensed under the terms of the Apache License Version 2.0.