Privitar AWS Deployment Guide

Publication date November 29, 2021

Table of Contents

1. Introduction

The Privitar Data Privacy platform is a data privacy solution that enables organizations to use sensitive datasets more safely.

NOTE
For ease of reference, the Privitar Data Privacy platform is referred to as the **Privitar platform**, (or just **Privitar**) in the rest of this manual.

The Privitar platform can be deployed in a variety of different configurations. This document describes how to deploy the native Amazon Web Services (AWS) instance of the Privitar platform.

NOTE
For ease of reference, the native AWS instance of the Privitar platform is referred to as **Privitar AWS**.

Privitar AWS is supplied as an AWS Cloud Development Kit (CDK) application that can be deployed into an AWS Cloud environment.

This document assumes that you are working in a Development Operations environment and have experience in deploying AWS resources into an AWS Cloud environment.

1.1. Compatibility with the Privitar platform

Privitar AWS contains most of the features of the Privitar platform application running in other types of environments. This includes:

- Importing schemas from AWS Glue Data Catalog
- Creating and running masking or unmasking jobs using AWS Glue ETL
- Embedding and extracting watermarks
- Masking rules and manual generalization (without k-anonymity)
- Running jobs on Privitar On Demand or AWS EMR (separate deployment of these 2 components is required)

The following features of the Privitar platform are **not** present in the current version of Privitar AWS:

- Usage of Encrypt rule in AWS Glue Jobs

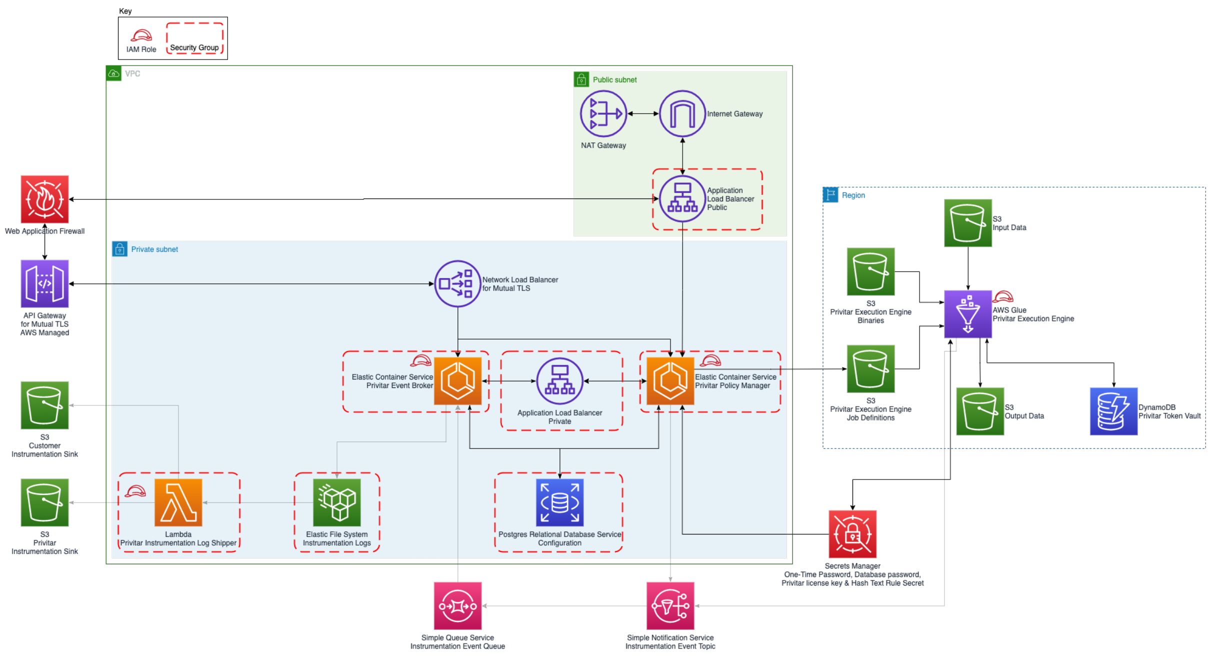
- Kerberos authentication for EMR
- Automatic generalization and k-anonymity in AWS Glue Jobs
- Supporting multiple tables masking/unmasking with a single AWS Glue Job

1.2. Architecture

This section describes the architecture of the Privitar AWS deployment together with information about the security implications of deploying AWS in your AWS Cloud environment.

1.2.1. Deployment architecture

The following diagram shows the overall architecture of the Privitar AWS Deployment and the associated AWS services that it uses:


Key
IAM Role
Security Group
VPC
Public subnet
NAT Gateway
Internet Gateway
Application Load Balancer Public
Web Application Firewall
API Gateway for Mutual TLS AWS Managed
Private subnet
Network Load Balancer for Mutual TLS
Elastic Container Service Privitar Event Broker
Application Load Balancer Private
Elastic Container Service Privitar Policy Manager
S3 Customer Instrumentation Sink
S3 Privitar Instrumentation Sink
Lambda Privitar Instrumentation Log Shipper
Elastic File System Instrumentation Logs
Postgres Relational Database Service Configuration
Region
S3 Input Data
S3 Privitar Execution Engine Binaries
AWS Glue Privitar Execution Engine
S3 Privitar Execution Engine Job Definitions
S3 Output Data
DynamoDB Privitar Token Vault
Secrets Manager One-Time Password, Database password, Privitar license key & Hash Text Rule Secret
Simple Queue Service Instrumentation Event Queue
Simple Notification Service Instrumentation Event Topic

The following table describes the AWS services that are used in the deployment:

Service	Description
Application Load Balancer and Network Load Balancer	Route traffic from the Internet to services, and between services.
Web Application Firewall	Blocks known malicious entities and provides rate limiting.
API Gateway	Terminate mutual TLS.
Elastic Container Service with Fargate	Host the applications that comprise Privitar AWS.
Relational Database Service	Store configuration and state for the applications that comprise Privitar AWS.
Secrets Manager	Store sensitive infrastructure information. For example, the configuration database (ConfigDB) password, the SAML secret, JSON web token (JWT) secret, Privitar NOVLT secret, watermarking secrets, and any others.

Service	Description
Simple Queue Service and Simple Notification Service	Route instrumentation telemetry data from Privitar Policy Manager and Glue execution environments to the Privitar Event Broker.
Elastic File System	Store event broker instrumentation logs, prior to shipping.
Lambda	Ship instrumentation logs and performs other small tasks. For example, key rolling.
Glue	Infer from Glue Schema. Execute Privitar jobs in Glue ETL.
DynamoDB	Store Privitar consistent tokenization mappings. This is referred to as the **Token Vault**.
S3	Store state, Glue ETL binaries, data, and logs.
VPC networking	Core infrastructure for the network layer, including NAT gateway, internet gateway, VPC, subnets, security groups.
Key Management System (KMS)	Encryption keys for resources created as part of this deployment.

1.2.2. Deployment Security

There are certain security implications that you need to be aware of when deploying Privitar AWS.

Internet-facing components

The deployment contains internet facing components. This means that:

- A Route53 registered domain and public hosted zone is required. This means that the domain name and certificates are public.
- The AWS Application Load Balancer that the Route53 DNS A record points to is internet-facing.
- The optional mutual TLS enabled AWS API Gateway is internet-facing.

AWS Glue

Glue jobs created by Privitar AWS currently run inside the AWS managed networks, and not inside customer owned VPCs. It should be noted that AWS managed networks have outbound internet access.

2. Deployment Overview

This section presents an overview of the steps to follow to deploy Privitar AWS in an AWS Cloud environment. For each step described a link is provided to the appropriate section in the document that describes the procedure to follow to complete that step.

⚠ **CAUTION**

You should not complete any actions required for this deployment as an AWS Account 'root' user. Instead, choose a lower privileged IAM user or role when following these steps.

Step	Summary	Description
1	Read Pre-requisites	Ensure that you have the resources to deploy Privitar AWS. This includes: • Privitar AWS resources • AWS Cloud resources See, Pre-requisites for deploying Privitar AWS [6]
2	Bootstrap AWS CDK	Perform the initial setup and configuration of the AWS CDK resources that are required to deploy Privitar AWS in your AWS Cloud. See, Bootstrap AWS Cloud Development Kit [8]
3	Upload the Privitar AWS Binaries	Upload the Privitar AWS binaries to an AWS S3 location. See, Copying the Privitar AWS Binaries [8]
4	Deploy Privitar AWS	Configure the Privitar AWS CloudFormation template to create a Stack containing the resources that are required by Privitar AWS. If the Stack is successfully created, CodePipeline will be automatically triggered to deploy Privitar AWS. See, Deploy Privitar AWS [9]

2.1. Pre-requisites for deploying Privitar AWS

This section details the pre-requisites to successfully deploy Privitar AWS in an AWS Cloud environment. There are two main areas:

- Privitar resources
- AWS Cloud resources

2.1.1. Privitar Resources

The following are needed for the deployment. Some of the items are provided as links to locations in the the private AWS Cloud environment that is hosted by Privitar. Other resources will be sent to you directly by Privitar:

- A launch-stack URL. This is a link to an AWS CloudFormation template that defines the resources used by Privitar AWS. Privitar will have granted you access to this location:

  ```
  https://eu-central-1.console.aws.amazon.com/cloudformation/home?
  region=eu-west-1#/stacks/create/review?templateURL=https://
  privitar-latest-release.s3-eu-west-1.amazonaws.com/cloudformation/
  cf_template.yaml
  ```
- The Privitar AWS infrastructure binaries (`infra.zip`) URL Privitar will have granted you access to this location:

  ```
  s3://privitar-latest-release/infra.zip
  ```
- A license key file. This key must be stored in AWS Secrets Manager. It is used to access Privitar AWS. Privitar will send you this information.

2.1.2. AWS Cloud Resources

The following items must be present in your AWS Cloud environment:

- Amazon S3 bucket location to store the Privitar binaries.

  The Privitar AWS deployment uses AWS CodePipeline. This pipeline consumes a zip file provided by Privitar which contains Privitar AWS artifacts and an AWS Cloud Development Kit application. Subsequent installations and updates of Privitar AWS involve uploading a zip file (provided by Privitar) to a configured location in S3 that is consumed by the pipeline.

  To create an Amazon S3 bucket to store the Privitar AWS artifacts and AWS Cloud Development Kit application, see Amazon S3 User Guide - Creating a Bucket.

  > **NOTE**
  >
  > The bucket that is created *must* be versioned. If it is not versioned, CodePipeline will raise an error. To remedy this, enable versioning for the S3 bucket and run CodePipeline again. See, Amazon S3 User Guide - Using versioning in S3 buckets.

- Amazon S3 bucket location to store Privitar AWS Instrumentation Events.

  For more information about the Instrumentation system on the Privitar Platform, refer to the *Privitar User Guide*.
- AWS Secrets Manager to store the Privitar license key. The key should be stored in plaintext.

  To create a secret in AWS Secrets Manager to store the Privitar license key, see AWS Secrets Manager User Guide - Creating a Secret.
- An Amazon Route53 registered domain and an Amazon Route53 public hosted zone.

  These resources are used to automatically create DNS A records to route traffic from the Route53 domain to Privitar services. This hosted zone also enables HTTPS by facilitating the automatic issuing of publicly signed certificates in AWS Certificate Manager. As part of this process, ownership of the domain is automatically confirmed by writing CNAME records which AWS Certificate Manager verifies.

To setup Amazon Route53, see Amazon Route53 Developer Guide.

- A mutual TLS certificate authority file stored in S3. (Optional)

  Note that this is only necessary if mutual TLS is required. Mutual TLS enables the use of Privitar On Demand, the Privitar SDK and Privitar Policy Manager REST APIs. If this option is specified, an AWS API Gateway will be created which provides mutual TLS connectivity using the configured certificate authority.

  To setup Mutual TLS to enable the use of Privitar On Demand, the Privitar SDK and Privitar Policy Manager REST APIs, see Amazon API Gateway Developer Guide.

2.2. Bootstrap AWS Cloud Development Kit

Privitar AWS is an AWS Cloud Development Kit (CDK) application. In order to begin to deploy Privitar AWS into an AWS Cloud environment, you need to provision resources that the AWS CDK needs to perform the deployment. These resources will include, an Amazon S3 bucket for storing data for example. The process of provisioning these initial resources is called *bootstrapping*.

You can learn about bootstrapping within the Amazon Guide to Bootstrapping. However, you must bootstrap with a version of the AWS CDK that is compatible with the Privitar deployment.

A compatible AWS CDK Bootstrap Cloudformation template (v1.107.0) can be downloaded directly from the AWS CDK project at: https://raw.githubusercontent.com/aws/aws-cdk/v1.107.0/packages/aws-cdk/lib/api/bootstrap/bootstrap-template.yaml

NOTE

If you are not familiar with how to bootstrap the AWS CDK, Prvitar can provide guidance and support. Contact `support@privitar.com`.

2.3. Copying the Privitar AWS Binaries

The Privitar AWS binaries consist of all the required software to run Privitar in an AWS Cloud environment. The binaries are available from the following URL which points to a shared AWS Cloud location that is hosted by Privitar. The location is private, but Privitar will have given you access. (See Pre-requisites for deploying Privitar AWS [6].):

```
s3://privitar-latest-release/infra.zip
```

The easiest way to copy the zip file from the Privitar AWS Cloud Environment to your AWS Cloud environment is to use the AWS CloudShell service. This service has the AWS Command Line Interface (CLI) (v2) installed and configured so you can run `aws s3` commands to manage the contents of your s3 buckets:

1. Sign in to the AWS Management Console and open the Amazon S3 console at:

   `https://console.aws.amazon.com/s3/`
2. Choose the **CloudShell** AWS service. This is a browser-based *shell* that makes it easy to securely manage, explore, and interact with your AWS resources using a command-line.
3. From CloudShell, enter the following command to copy the zip file from Privitar to your location, where `<my-customer-bucket>` is an S3 location in your area where you wish to copy the file:

   ```
   aws s3 cp s3://privitar-latest-release/infra.zip s3://my-customer-
   bucket/infra.zip
   ```

For a comprehensive description of using CloudShell, refer to AWS CLoudShell User Guide.

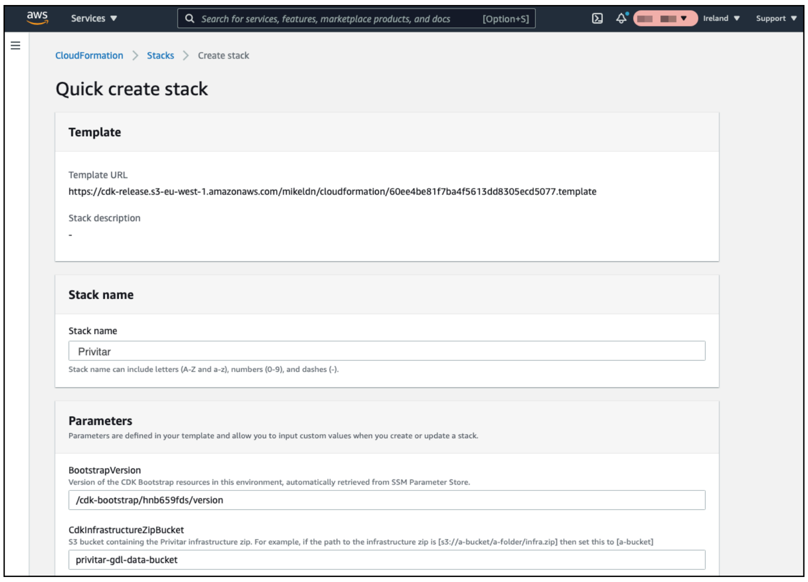
2.4. Deploy Privitar AWS

The Privitar AWS Stack contains all the resources that are required to deploy Privitar AWS in your AWS Cloud environment. Once the stack has been configured and successfully created, CodePipeline will then automatically deploy Privitar AWS.

To create the stack and configure the resources that are included in the stack, Privitar provides a **launch stack URL** that links to an AWS CloudFormation template that defines the resources used by Privitar AWS.

To create the stack, click on the Privitar launch stack URL. (This URL will have been sent to you from Privitar. See, Pre-requisites for deploying Privitar AWS [6]).

The AWS **Quick create stack** page is displayed:


aws
Services
Search for services, features, marketplace products, and docs
[Option+S]
[redacted]
Ireland
Support
CloudFormation > Stacks > Create stack

Quick create stack

Template

Template URL
https://cdk-release.s3-eu-west-1.amazonaws.com/mikeldn/cloudformation/60ee4be81f7ba4f5613dd8305ecd5077.template

Stack description
-

Stack name

Stack name
Privitar
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

BootstrapVersion
Version of the CDK Bootstrap resources in this environment, automatically retrieved from SSM Parameter Store.
/cdk-bootstrap/hnb659fds/version

CdkInfrastructureZipBucket
S3 bucket containing the Privitar infrastructure zip. For example, if the path to the infrastructure zip is [s3://a-bucket/a-folder/infra.zip] then set this to [a-bucket]
privitar-gdl-data-bucket

This page lists the configuration parameters that are available for the Privitar AWS stack. Many of the parameters contain sensible default values, but others must be completed to set up the stack for your particular AWS Cloud environment.

To create and configure the Privitar AWS stack:

1. Enter a name for the stack in the **Stack name** field.

   The default name is **Privitar**. You can change this name to one that is more suitable for your AWS Cloud environment. A stack name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 128 characters.
2. Enter new values to configure the Privitar AWS stack in the **Parameters** section.

   The following table describes the parameters that must be completed to create a Privitar AWS stack. (For more information about the optional parameters available in this section, see Stack Parameters [24].)

Parameter	Description

CdkInfrastructureZipBucket	The Amazon S3 bucket name of the S3 bucket containing the Privitar AWS infrastructure zip file. The AWS CodePipeline deployed by the CloudFormation template will look for the Privitar infrastructure zip file in this bucket.
	
	For example, if the full path to the infrastructure zip file is:
	
	`s3://a-bucket/a-folder/infra.zip`
	
	then set this parameter to:
	
	`a-bucket`

CdkInfrastructureZipKey	The Amazon S3 key for the S3 object that is the Privitar AWS infrastructure zip file.
	
	For example, if the full path to the infrastructure zip file is:
	
	`s3://a-bucket/a-folder/infra.zip`
	
	then set this parameter to:
	
	`a-folder/infra.zip`

CreateTenantInstrumentationBucket	Whether or not to create a separate S3 bucket for storing the Privitar AWS instrumentation events. This can be set to `True` or `False`. The default setting is: `False`. If set to `True`, you must provide an S3 bucket location to store the events. The location is provided by Privitar and can be set using the PrivitarInstrumentationBucket parameter. The name of the bucket created is of the format: privitar-${deploymentId}-instrumentation-bucket where ${deploymentId} is substituted with the setting of the DeploymentID parameter of this template.

Parameter	Description
DeploymentID	A unique identifier for the Privitar AWS deployment. Resources created by Privitar AWS CloudFormation will include the value of this parameter in their identifier and name. This helps AWS users to understand the resource, and enables running multiple Privitar deployments in the same AWS account. The deployment ID can contain only alphanumeric characters (lower-case alphabetic characters only) and hyphens. It must start with an alphabetic character and can't be longer than 128 characters. For example: `data1234`
HostedZoneID	The AWS Route 53 public hosted zone ID to use to generate a domain name and certificate that can be used by Privitar AWS. AWS Route 53 is used to automatically create DNS A records to route traffic from the Route53 domain to Privitar services. This hosted zone also enables HTTPS by facilitating the automatic issuing of publicly signed certificates in AWS Certificate Manager. As part of this process, ownership of the domain is automatically confirmed by writing CNAME records which AWS Certificate Manager verifies. For example: `Z053459332BOI952MZ6ZK`
HostedZoneName	The AWS Route 53 hosted zone domain name of the public hosted zone specified in the `HostedZoneID` parameter. For example: `privitar-aws.com`
LicenceKeySecret (Provided by Privitar)	The ARN of a plaintext AWS Secrets Manager secret that contains the Privitar license key file contents. This license key may need to be periodically updated. For example: `arn:aws:secretsmanager:eu-west-1:423009687000:secret:license-key`

Parameter	Description
PrivitarInstrumentationBucket (Provided by Privitar)	The name of the S3 bucket location to store Privitar AWS instrumentation events. The events are non-sensitive data events that describe how Privitar AWS is being used. Typically, the S3 bucket location will reside in Privitar's AWS account.

3. Click the check box in the **Capabilities** section to agree to the changes that will be made to your AWS account by creating the Privitar AWS stack.
4. Select **Create stack** to create the Privitar AWS stack.

   If the stack is created successfully, it will appear in the AWS CloudFormation stack list:

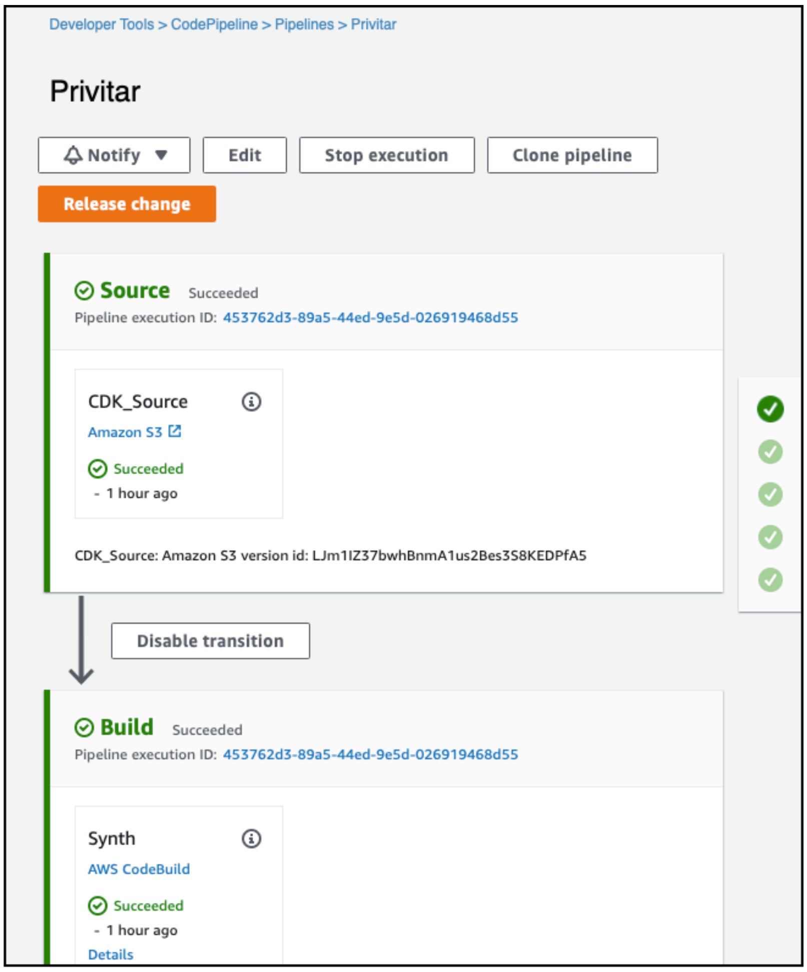
   ```
   CloudFormation > Stacks > <stack-name>
   ```

   where `<stack-name>` is the name of the stack that you defined in the **Stack name** field. By default, the name will be **Privitar**.

   CloudFormation will report **CREATE COMPLETE** in the **Status** column to indicate that the stack has been successfully created.

   The successful completion of the stack by CloudFormation will automatically trigger CodePipeline to deploy Privitar AWS.
5. Navigate to **Developer Tools > CodePipeline > Pipelines > <stack-name>**

   From this location you can observe each stage of the deployment taking place. For example:



Source Succeeded
Pipeline execution ID: 453762d3-89a5-44ed-9e5d-026919468d55

CDK_Source
Amazon S3
Succeeded
- 1 hour ago

CDK_Source: Amazon S3 version id: LJm1IZ37bwhBnmA1us2Bes3S8KEDPfA5

Build Succeeded
Pipeline execution ID: 453762d3-89a5-44ed-9e5d-026919468d55

Synth
AWS CodeBuild
Succeeded
- 1 hour ago

6. In the CodePipeline interface, look for the **Review** stage to manually approve the changes before they are deployed. When the change set is ready to be reviewed, click the **Review** button in the **ManualApproval** action, then click **Accept** to accept the changes and continue the deployment.
7. If the deployment is successful, a new deployment will be created in **CloudFormation > Stacks**. The new stack will have **-deployed** appended to the stack name. So, in the above example, the Privitar AWS deployment created would be:

   Privitar-deployed
8. To confirm that Privitar AWS has been deployed, enter the URL of the deployment.

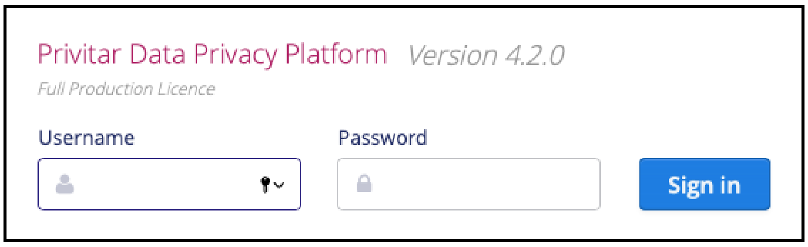
   The URL is derived from the `DeploymentID` parameter together with the `HostedZoneName` that were defined when the Privitar AWS stack was created. For example, if these two parameters were defined as:

   - DeploymentID - `data1234`
   - HostedZoneName - `privitar-aws.com`

   The URL for Privitar AWS would be:

   `https://data1234.privitar-aws.com`

   Enter the URL in a browser, the Privitar AWS login screen is displayed:



9. A one-time-use password has been randomly generated and stored as a secret in AWS Secrets Manager, under the name of `privitar/DeploymentID/one-time-password`, where `DeploymentID` is the value of the parameter defined in the Privitar AWS Stack. To retrieve the one-time password in the AWS Secret Manager service, go to the AWS Console and navigate to **AWS Secrets Manager > Secrets**, click on the secret with the name defined above and click **Retrieve Secret Value** to show the password value.
10. Go back to the URL for Privitar AWS, use the username `admin` and the one-time password retrieved in the step above to log-in. The Privitar AWS dashboard is displayed.

    Refer to Using Privitar AWS [23] for more information on how to setup and begin using Privitar AWS.

3. Optional - Configure Single Sign-On with SAML

The Privitar Platforms supports two different User Management modes:

1. Internal (default). The Privitar Platform manages the users credentials and their groups membership in its local database.
2. Single Sign-On (SSO). The Privitar Platform uses a 3rd party Identity Provider via the SAML2.0 protocol.

When using single sign-on, there are two parties, the Privitar Platform as Service Provider (SP), and a third-party identity service as the Identity Provider (IDP).

Follow the following steps to configure SSO:

1. Obtain the metadata file from the IDP service and upload it to an S3 bucket which can be read by the Privitar service. For example, a possible location could be the bucket where infra.zip is to be uploaded (as specified by the CF parameter `CdkInfrastructureZipBucket`).
2. Provide the following settings via CloudFormation parameters:

Parameter	Value
UserManagementProvider	Set this parameter to `SSO_SAML` to enable Single Sign On.
SsoSamlIdpMetadataS3Uri	This parameter should be set to the S3 location where the metadata file was uploaded in step 1.
SsoSamlSuperuserGroup	This parameter should contain the name of the group which maps on to superuser privilege.
SsoSamlUserAttribute	The name of the saml attribute representing a unique username that can be used by publisher to manage the user. The identity provider will give a saml assertion containing this attribute
SsoSamlUserDisplayNameAttribute	The name of the SAML attribute representing a display name for the user which is used as the user's name in the policy manager UI.
SsoSamlUserGroupsAttribute	A list of permissions groups to which the user belongs. These can be mapped on the Teams and Roles in the Privitar UI. Note: Privitar is case insensitive with regards to group names.

3. Apply the changes to the Stack (see the Modify Stack Parameter [20] section of this guide) and wait for the changes to be applied.
4. Once the Privitar Policy Manager has started, download the Service Provider metadata file from the following address: `https://<DeploymentID>.<HostedZoneName>/saml/metadata`. Use this file to configure your IDP service.

NOTE

SAML RSA Key Pair

The deployment automatically creates an RSA key pair required for signing and verifying SAML messages. This is stored in **AWS Secrets Manager** under the identifier `privitar-sso-saml-default-key-pair-{deployment_id}`.

If you wish to rotate the key pair, an administrator can delete the secret and re-run the deployment through the AWS CodePipeline screen by selecting **Release Change**. The deployment will generate a new RSA key pair. When the deployment is complete, you will need to reconfigure your IDP service with the new Service Provider metadata file. Be prepared for a period of downtime during these operations.

4. Manage the Deployment

To facilitate the installation and update process using Cloud Development Kit, Privitar uses the AWS CodePipeline continuous delivery service. CodePipeline makes deployment and future updates easier, by automating many of the deployment steps. For example, a Privitar AWS update involves simply uploading the zip file to a defined S3 location, which will then trigger CodePipeline to redeploy the application.

It is also possible to use CodePipeline to roll back a deployment if an error occurs during the deployment. This section describes how to manage the deployment.

4.1. CodePipeline Deployment Stages

The following table describes the sequential deployment steps that are executed by CodePipeline. If any of the steps fail to complete, then the update halts and rolls back any infrastructure changes. Although, any database changes made by applications as part of the update are not rolled back automatically.

NOTE
When deploying for the first time, a deployment should typically take between 60 to 90 minutes.

Stage	Description
1. Source	Triggers the Pipeline when a new Privitar infrastructure zip file is uploaded to the appropriate S3 bucket location.
2. Build	Synthesizes the Cloud Development Kit (CDK) application to CloudFormation. This generates several CloudFormation templates, one for each stack that makes up the deployment, including several nested stacks.
3. UpdatePipeline	If the CodePipeline itself has changed between releases, then the template and its resources (the pipeline) are updated. The primary purpose of this is to add or change CloudFormation parameters or to update the CDK version being used.
4. Assets	Uploads the CloudFormation templates generated during the Build stage and uploads any binaries (such as the Privitar Glue integration).

Stage	Description
5. Pre-deployment	The images that are required for deployment are copied from the Privitar repository to your repository (CopyEcrimage). If the deployment is an update for Privitar AWS, then a snapshot (CreateDBSnapshot) of the Privitar Policy Manager configuration database is taken. A snapshot will be required if the deployment fails and a roll back is required. An SSO SAML key-pair (CreateSSoSamlDefaultKeyPair) is created, if you enable it as a stack parameter.
6. Review	The CodePipeline will pause at the Review section and wait for your input to proceed. Click **Review** under ManualApproval. Should you wish to inspect the changes that are about to be applied, click the link to the AWS CloudFormation change set. Do not apply the change set or execute the AWS CloudFormation outside of the CodePipeline view. Instead, return to the CodePipeline view. Click **Approve** when you wish to proceed with the deployment.
7. Deployment-stack	The CodePipeline executes the CloudFormation change set. This stage rolls out the update to the `-deployment` CloudFormation stack and its resources. Examples of changes include: modify existing IAM policies, roll out updated ECR images of Privitar applications to ECS.

4.2. Rolling Back a Deployment

A manual rollback is required if the deployment has failed to complete during any of the stages outlined in the previous section.

To perform a manual roll-back

1. If CodePipeline reports that the Privitar AWS CloudFormation stack is in one of the following states:
   - ROLLBACK_FAILED
   - UPDATE_ROLLBACK_FAILED

   To recover from either of these states, upload the Privitar AWS zip file for the version that you want to roll-back to and use CodePipeline to roll back to the different version. If this also fails, contact Privitar as manual changes may be needed.

   If the CodePipeline reports that the stack is in one of these following states:
   - ROLLBACK_COMPLETE
   - UPDATE_COMPLETE

   This means that the infrastructure and ECS image versions will already have been automatically rolled back, so there is no further action required.
2. To roll back the configuration database to the snapshot created during the *Pre-deployment, CreateDBSnapshot* stage of the CodePipeline run that is being rolled back, see Restoring from a DB snapshot.

   Note that when working through this information:

- The RDS will be called:

  `${DeploymentID}-db`
- The snapshot will be called:

  `${DeploymentID}-db-pre-update-${date-time}`

  For example:

  `privitar-db-pre-update-2021-02-11-11-34`

3. Restart the Privitar Policy Manager and Event Broker ECS containers so that they are automatically restarted and recover.

   This can be done from ECS, by selecting **Stop All** from the ECS **Task** tab.

5. Modify Stack Parameters

It is possible to make changes to configuration of the existing Privitar AWS deployment by modifying the Privitar AWS stack parameters and updating the stack.

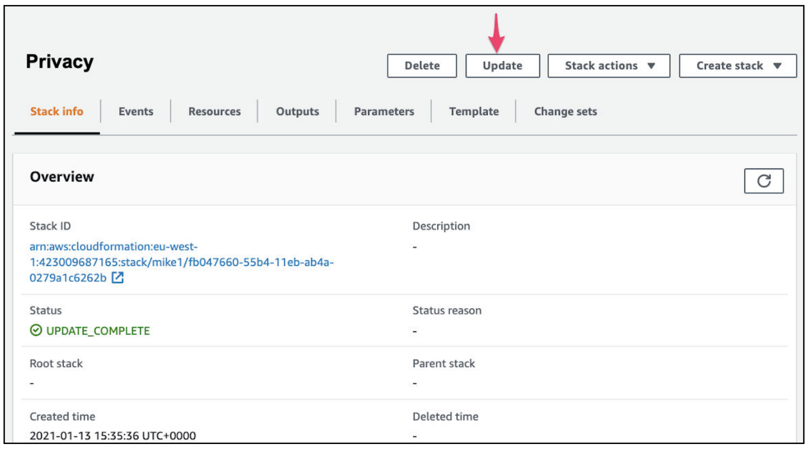
Some of the changes that can be made, include:

- Adding functionality that you did not include in a previous deployment. For example, mutual TLS.
- Taking advantage of a new configuration parameter that has been added to the Privitar AWS stack.

To modify the stack parameters:

1. Locate the Privitar AWS stack that you created in AWS CloudFormation. This is the original stack that you created, not the `-deployed` version. For example:

   `CloudFormation > Stacks > <stack-name>`
2. Click on **Update**:



Stack ID
arn:aws:cloudformation:eu-west-1:423009687165:stack/mike1/fb047660-55b4-11eb-ab4a-0279a1c6262b

Description
-

Status
UPDATE_COMPLETE

Status reason
-

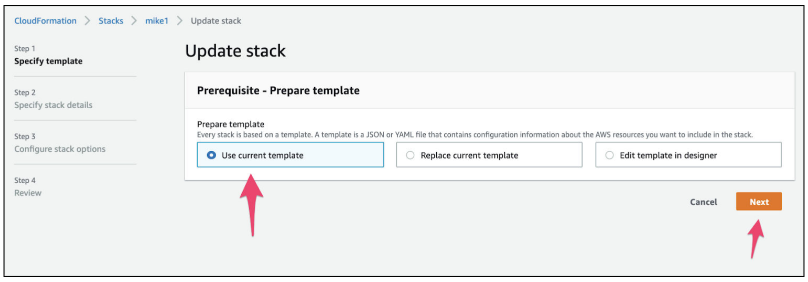
Root stack
-

Parent stack
-

Created time
2021-01-13 15:35:36 UTC+0000

Deleted time
-

The **Update Stack** window is displayed:



Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

[✓ Use current template] Replace current template Edit template in designer

3. Select **Use Current Template** and click **Next**.

   The **Specify stack details** window is displayed.
4. Modify the stack parameters as required.

   Refer to the descriptive text provided above each parameter for more information about the use of each parameter. For a summary of all the parameters, see Stack Parameters [24].

5. Click **Next**.

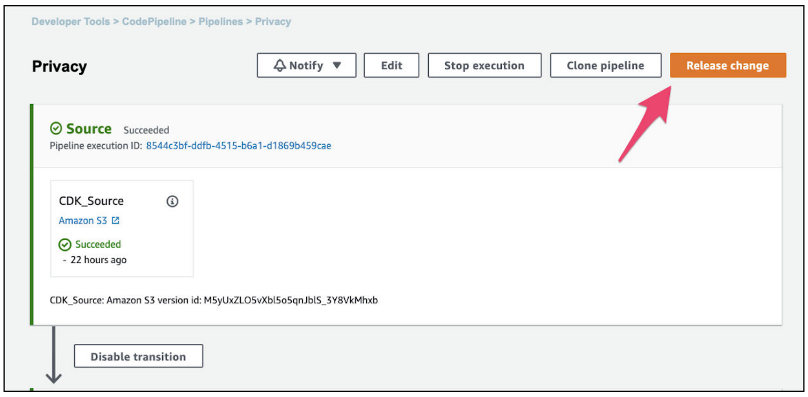
   The **Configure stack options** window is displayed. Modify any options in this window.
6. Click **Next**.

   The **Review** window is displayed summarizing all the values that have been specified. It is possible to make other changes to the stack configuration from here.
7. Click **Update stack** to confirm the changes and update the stack.

   The stack is updated and the main stack window is displayed.

   CloudFormation will report **CREATE COMPLETE** in the **Status** column to indicate that the stack has been successfully created.
8. To redeploy Privitar AWS with the new stack, for the new changes to take effect, you need to force an update on CodePipeline for the stack.

   Navigate to **Developer Tools > CodePipeline > Pipelines > <stack-name>** and click on **Release change** to force the update:



Pipeline execution ID: 8544c3bf-ddfb-4515-b6a1-d1869b459cae

- 22 hours ago

CDK_Source: Amazon S3 version id: M5yUxZLO5vXbl5o5qnJblS_3Y8VkMhxb

   CodePipeline will report **Succeeded** in the **Status** column.

   This means that all the stages in CodePipeline have been successful and the update to Privitar AWS has been successfully deployed. (If the deployment has failed for any reason, see AWS CodePipeline User Guide - Troubleshooting.)

6. Updating Privitar AWS

The Privitar AWS binaries consist of all the required software to run Privitar in an AWS Cloud environment. Any update for Privitar AWS will be supplied to you as a zip file from Privitar.

To update Privitar AWS:

1. Copy the new zip file from the Privitar AWS S3 bucket to the appropriate S3 bucket in your AWS Cloud environment.

   The S3 bucket location for your AWS Cloud environment is defined by the following parameters that were defined when the Privitar AWS stack was created:

   - `CdkInfrastructureZipBucket`
   - `CdkInfrastructureZipKey`

   (For more information about copying files to an S3 bucket, see Copying the Privitar AWS Binaries [8].)
2. Copying a new zip file to the S3 bucket, will automatically trigger AWS CodePipeline to begin to re-deploy the new version of Privitar AWS.
3. If the re-deployment is successful:
   - CloudFormation will report **UPDATE_COMPLETE** in the **Status** column.

     This means that the CloudFormation template has been successfully updated. (If the template has failed to update for any reason, see AWS CloudFormation User Guide - Troubleshooting.)
   - CodePipeline will report **Succeeded** in the **Status** column.

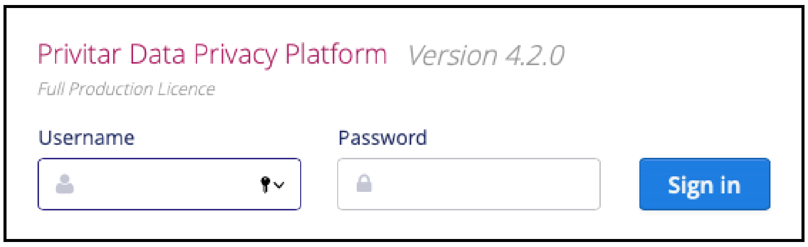
     This means that all the stages in CodePipeline have been successful and the update to Privitar AWS has been successfully deployed. (If the deployment has failed for any reason, see AWS CodePipeline User Guide - Troubleshooting.)
4. To confirm that Privitar AWS has been re-deployed, enter the URL of the deployment.

   For example:

   ```
   https://data1234.privitar-aws.com
   ```

   Enter the URL in a browser, the Privitar AWS login screen is displayed.

   If the deployment has been successful, the displayed **Version** of the platform will display the updated version number of the software:



7. Using Privitar AWS

This section describes some initial setup tasks for using Privitar AWS, following a new installation.

This section assumes that you have just deployed a new version of Privitar, as described in Deploy Privitar AWS [9].

In summary, the main setup tasks that you need to perform before you can use Privitar AWS to de-identify your data are listed below. For each task, there is a reference to to the specific section of this guide that provides more information:

- Setup Users and Teams. See, *Managing Roles* in the *Privitar User Guide*.
- Setup your AWS Glue Environment. See, *AWS Glue Environment Configuration* in the *Privitar User Guide*.

To begin using Privitar AWS, you need to:

1. Define a schema for the new input data. See, *Creating a Schema from an AWS Glue Data Catalog* in the *Privitar User Guide*.
2. Create a policy representing the de-identification operations required. See, *Creating a Policy* in the *Privitar User Guide*.
3. Define the Protected Data Domain (PDD) to be used as the destination for all data related to the use case in question. See, *Creating a PDD* in the *Privitar User Guide*.
4. Based on the policy, create and execute a job. This is done by selecting a policy and a destination Protected Data Domain. See, *Creating a Batch Job* and *Running a Batch Job* in the *Privitar User Guide*.

8. Stack Parameters

The following table describes the stack parameters that are used to configure the Privitar AWS stack. The bold parameters are mandatory.

Parameter	Description
BootstrapVersion	The version of the CDK Bootstrap resources in this environment, automatically retrieved from SSM Parameter Store.
CdkInfrastructureZipBucket	The Amazon S3 bucket name of the S3 bucket containing the Privitar AWS infrastructure zip file. The AWS CodePipeline deployed by the CloudFormation template will look for the Privitar infrastructure zip file in this bucket.
CdkInfrastructureZipKey	The Amazon S3 key for the S3 object that is the Privitar AWS infrastructure zip file.
CreateTenantInstrumentationBucket	Whether or not to create a separate S3 bucket for storing the Privitar AWS instrumentation events.
DeploymentID	A unique identifier for the Privitar AWS deployment.
DeploymentGlobalID	Across all of AWS, uniquely identify global entities; for example S3 buckets must have a unique name across all accounts. Global resources created by Privitar CloudFormation will include the value of this parameter in their name.
GlueDataBucketKeys	The IDs of AWS KMS customer managed keys (CMK) that Privitar AWS Glue ETL jobs are allowed to use.
GlueDataDestinationBuckets	Specifies what S3 buckets Privitar AWS Glue ETL jobs are allowed to write data to, as a comma delimited list.
GlueDataSourceBuckets	Specifies what S3 buckets Privitar AWS Glue ETL jobs are allowed to read data from, as a comma delimited list.
HadoopUserName	The username to use to read or write HDFS files, if running jobs against an AWS EMR cluster.
HostedZoneID	The AWS Route 53 public hosted zone ID to use to generate a domain name and certificate that can be used by Privitar AWS.
HostedZoneName	The AWS Route 53 hosted zone domain name of the public hosted zone specified in the `HostedZoneID` parameter.
LicenceKeySecret (Provided by Privitar)	The ARN of a plaintext AWS Secrets Manager secret that contains the Privitar license key file contents.
MutualTlsCertificateAuthorityPath	Full path to Mutual Transport Layer Security (MTLS) certificate authority.

Parameter	Description
PrivitarInstrumentationBucket (Provided by Privitar)	The name of the S3 bucket location to store Privitar AWS instrumentation events.
RdsSnapshotID	If creating a new deployment from an existing Privitar Policy Manager configuration database, specify an AWS RDS snapshot to restore from.
RdsSnapshotSecret	If RdsSnapshotID is specified then this option is required. This is the ARN of a plaintext AWS Secrets Manager secret, that contains the AWS RDS snapshot database password.
SsoSamlIdpMetadataS3Uri	The S3 location of the SAML Identity Provider metadata file. Required when using SSO_SAML as User Management Provider.
SsoSamlSuperuserGroup	This parameter should contain the name of the group which maps on to superuser privilege. Required when using SSO_SAML as User Management Provider.
SsoSamlUserAttribute	The name of the SAML attribute representing a unique username that can be used by Privitar to manage the user. The identity provider will give a SAML assertion containing this attribute. Required when using SSO_SAML as User Management Provider.
SsoSamlUserDisplayNameAttribute	The name of the saml attribute representing a display name for the user which is used as the user's name in the policy manager UI. Required when using SSO_SAML as User Management Provider.
SsoSamlUserGroupsAttribute	A list of permissions groups to which the user belongs. These can be mapped on the teams and roles in the Privitar UI. Note: Privitar is case insensitive with regards to group names. Required when using SSO_SAML as User Management Provider.
Tags	This parameter controls what AWS tags will be applied to resources created by Privitar CloudFormation.
TrustedAdminRoles	A comma delimited list of trusted AWS IAM roles. Privitar infrastructure includes strict security controls that may prevent AWS users from reading or writing to certain resources.
UserManagementProvider	This parameter defines the Identity Provider used by the Privitar Platform. Possible values are `INTERNAL` (default) or `SSO_SAML`.
VpcPeerings	This parameter allows the Privitar control plane VPC to be automatically peered with other VPCs.

Parameter	Description
WhatfixMode	Enable or disable the Help and Training Center (in-app user guidance) on the platform. FULL (the default) enables Help and Training Center content and **analytics collection**. CONTENT_ONLY enables Help and Training Center content only, with no analytics collection.
WhatfixUrl	This is a Privitar-supplied URL that is required to enable Help and Training Center content.

9. Deleting a Deployment

Should you wish to delete a deployment, you can delete the AWS CloudFormation stacks that created the deployment as outlined in Deploy Privitar AWS [9]. This will clear up most resources associated with the deployment.

This process does not automatically remove some resources, such as those that the application created or are sensitive, and these may have ongoing costs attached to them within AWS.

This following list details resources that an administrator may wish to remove manually within the AWS console. You can also remove some items through the Policy Manager prior to deleting a deployment, but this is not essential.

CAUTION

Please contact support@privitar.com if you are unsure about the impact of deleting any of the following resources.

- Secrets stored in **AWS Secrets Manager**, such as the secret enabling the use of the Hash Text rule or a secret enabling SAML authentication (if enabled).

  WARNING

  Deleting a secret will irreversibly prevent ongoing use of that secret in operations that require it.

- **Amazon DynamoDB tables** storing tokenized values. These can be deleted through the AWS console or deleted within the Policy Manager prior to deleting a deployment.

  WARNING

  Deletion of these tables deletes tokenized values and will irreversibly prevent any subsequent unmasking operations. This action deletes data.

- **AWS Glue jobs**, including historical metadata. These can be deleted through the AWS console or within the Policy Manager prior to deleting a deployment.
- **Amazon S3 buckets** used for instrumentation data (if enabled) and logs, for example. These can be deleted through the AWS console.
- **Amazon RDS snapshots** containing database backups of the Config DB component. These can be deleted through the AWS console.

10. Troubleshooting

This section aims to help resolve issues that you might encounter with the deployed solution.

10.1. Amazon CloudWatch Logs

To identify problems when performing the deployment, consult the AWS CloudFormation and AWS CodePipeline message output.

To assist with identifying problems post deployment, the following table details which Amazon CloudWatch logs can also be useful for troubleshooting particular components.

TIP

You can also contact Privitar support at support@privitar.com for assistance diagnosing an issue.

AWS Resource [Privitar Architectural Component]	**Log Group Format Name**	**Log Stream Format Name**	**How to Find Them**
Amazon Elastic Container Service (Event Broker) [Policy Manager]	{deployment-id}-event-broker	{log-group-name}/EventBrokerContainer/{stream-unique-id}	ECS > Cluster name: {deployment-id}-cluster > Tasks tab > Event broker task > Container > logs
	{deployment-id}-policy-manager	{log-group-name}/PolicyManagerContainer/{stream-unique-id}	ECS > Cluster name: {deployment-id}-cluster > Tasks > Policy Manager task > Container > logs
Amazon Relational Database Service (Amazon RDS) [PostgreSQL Configuration Database]	/aws/rds/instance/{deployment-id}-db/postgresql	{deployment-id}-db.{unique-id}	RDS > Databases > Database name: {deployment-id}-db > Configuration tab > Published logs > postgresql
	/aws/rds/instance/{deployment-id}-db/upgrade		RDS > Databases > Database name: {deployment-id}-db > Configuration tab > Published logs > upgrade

AWS Lambda [Instrumentation JWT key rotation lambda] [Instrumentation event uploader Lambda]	/aws/lambda/{deployment-id}-instrumentation-jwt-key-rotation	{date}/[$LATEST]{stream-unique-id}	Lambda > functions > function name: {deployment-id}-instrumentation-jwt-key-rotation > Monitor tab > View Logs in CloudWatch
	/aws/lambda/{deployment-id}-instrumentation-event-uploader		Lambda > functions > function name: {deployment-id}-instrumentation-event-uploader > Monitor tab > View Logs in CloudWatch
AWS Glue [Privitar Batch Jobs using AWS Glue ETL]	/aws-glue/jobs/error	{job-unique-id}	Glue > jobs > job name: privitar_{job-type}_{deployment-id}_{privitar-job-id} > error logs
	/aws-glue/jobs/logs-v2		Glue > jobs > job name: privitar_{job-type}_{deployment-id}_{privitar-job-id} > logs

10.2. Unable to Execute Jobs

If you are unable to execute as many Privitar jobs as you expect when running a Privitar Batch Job with AWS Glue, it's possible that you are hitting an AWS Glue Service Limit (listed on https://docs.aws.amazon.com/general/latest/gr/glue.html).

This could be because there are many concurrent Privitar Batch Jobs running (for example, you are attempting to exceed the "Max concurrent job runs per account") or that no more jobs can be created within the account (for example, "Max jobs per account"). You may need to raise an AWS Support Case to raise an AWS Service Limit. If you suspect this issue, please reach out to support@privitar.com for assistance.

10.3. Amazon CloudWatch Metrics

This table presents key Amazon CloudWatch metrics that can provide insight and assistance when troubleshooting a deployment.

TIP

You can also contact Privitar support at support@privitar.com for assistance diagnosing an issue.

AWS Resource	Insight	How to find them	Relevant Metrics
ALB	Connection and request behavior between components within the deployment (see architecture diagram [4])	EC2 > Load Balancers > name: {deployment-id}-{type}-alb-{unique-id} > Monitoring tab	ActiveConnectionCount, DroppedInvalidHeaderRequestCount, ForwardedInvalidHeaderRequestCount, HTTP_Fixed_Response_Count, HTTP_Redirect_Count, HTTP_Redirect_Url_Limit_Exceeded_Count, HTTPCode_ELB_3XX_Count, HTTPCode_ELB_4XX_Count, HTTPCode_ELB_5XX_Count, HTTPCode_ELB_500_Count, HTTPCode_ELB_502_Count, HTTPCode_ELB_503_Count, HTTPCode_ELB_504_Count, ProcessedBytes, RejectedConnectionCount, RequestCount, HealthyHostCount, UnHealthyHostCount, TargetResponseTime, TargetConnectionErrorCount, RequestCountPerTarget, HTTPCode_Target_2XX_Count, HTTPCode_Target_3XX_Count, HTTPCode_Target_4XX_Count, HTTPCode_Target_5XX_Count
DynamoDB	Token Vault behavior and resource utilization	DynamoDB > tables > table name: {deployment-id}_{pdd-id}_{type}	ConditionalCheckFailedRequests, ConsumedReadCapacityUnits, ConsumedWriteCapacityUnits, ReadThrottleEvents, ReturnedBytes, ReturnedItemCount, ReturnedRecordsCount, SuccessfulRequestLatency, SystemErrors, ThrottledRequests, UserErrors, WriteThrottleEvents
ECS	Resource utilization for the Policy Manager and the Event Broker	ECS > Cluster name: {deployment-id}-cluster > Metrics tab	CPUUtilization, MemoryUtilization

AWS Resource	Insight	How to find them	Relevant Metrics
NAT Gateway	Connections to external services outside the VPC	VPC > Nat Gateways > name: {deployment-id}/ deployment-stack/ deployment-root/network-stack/ deployment-vpc/{public-subnet-identifier} > Monitoring tab	ActiveConnectionCount, ConnectionAttemptCount, ConnectionEstablishedCount, ErrorPortAllocation, PacketsDropCount
NLB	Inbound mTLS requests to the deployment	EC2 > Loadbalancers > name: {deployment-id}-mtls-nlb > Monitoring tab	ActiveFlowCount, HealthyHostCount, ProcessedBytes, ProcessedPackets, UnHealthyHostCount
RDS	Config DB behavior and resource utilization	RDS > Databases > Database name: {deployment-id}-db >Monitoring tab	DatabaseConnections, FreeStorageSpace, ReadLatency, WriteLatency,
SNS	Instrumentation events	SNS > Topic > name: {deployment-id}-privitar-events(.fifo) > Monitoring tab	NumberOfMessagesPublished, NumberOfNotificationsDelivered, NumberOfNotificationsFailed, NumberOfNotificationsFilteredOut,
WAF	Inbound requests from the web application firewall (WAF)	WAF > Web Acls > name: {deployment-id}-waf > Logging and metrics tab	AllowedRequests, BlockedRequests, CountedRequests, PassedRequests

11. AWS Costs

You will pay Amazon for the AWS resources that you use as part of the deployment. This will include a standing cost to run the platform, for example the services listed in the Architecture [4] section of this guide, such as Amazon Elastic Container Service (Amazon ECS), as well as additional costs resulting from execution of a Privitar Batch Job with AWS Glue. The AWS Glue cost, for instance, will vary with the number of Privitar Batch Jobs that you run, and it will depend on the volumes of data processed. The Amazon DynamoDB cost, for instance, will vary with the volume of tokenized values stored and the capacity required to fulfill job tokenizations.