

Deploy an Amazon Aurora PostgreSQL DB cluster with recommended best practices using AWS CloudFormation

In this document, I cover how to build a quick start reference deployment of [Amazon Aurora PostgreSQL](#) cluster. The cluster is based on AWS best practices for security and high availability and you can create it quickly by using [AWS CloudFormation](#). I walk through a set of sample CloudFormation templates, which you can customize to suit your needs.

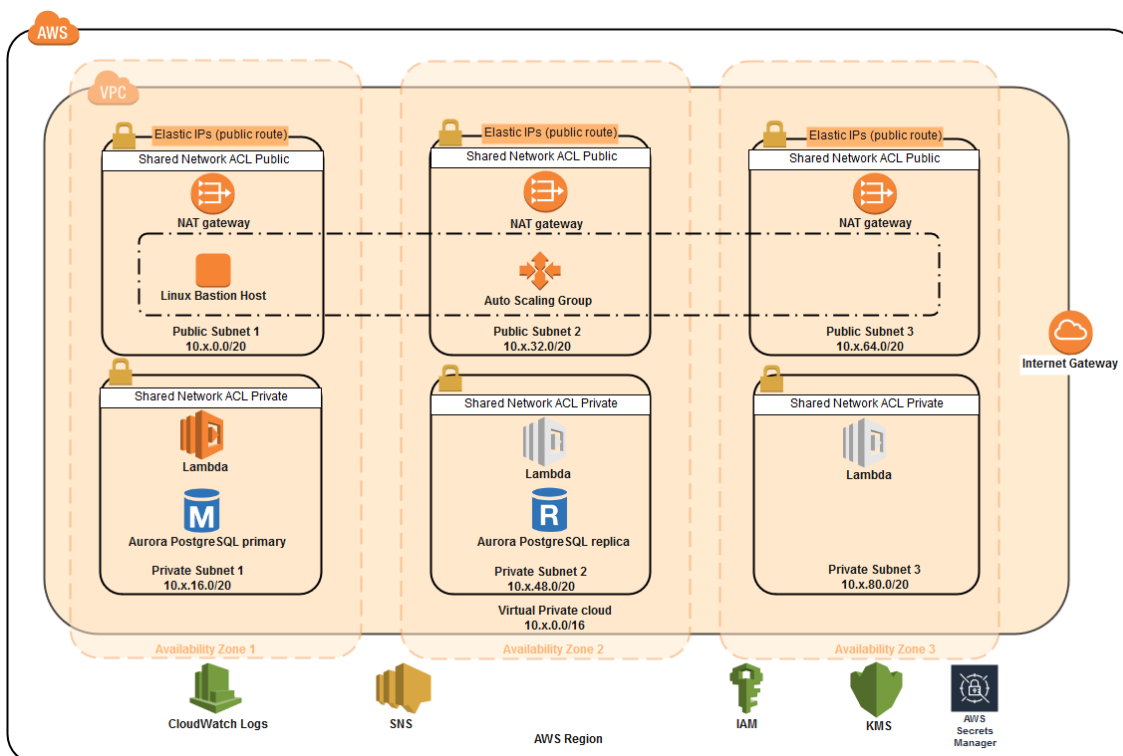
Amazon Aurora is a MySQL and PostgreSQL-compatible [relational database](#) built for the cloud. Aurora combines the performance and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases.

The PostgreSQL-compatible edition of Aurora delivers up to three times the throughput of standard PostgreSQL running on the same hardware. This enables existing PostgreSQL applications and tools to run without being modified. The combination of PostgreSQL compatibility with Aurora enterprise database capabilities provides an ideal target for commercial database migrations.

When you are starting your journey with Aurora PostgreSQL and want to set up AWS resources based on the recommended best practices of [AWS Well-Architected Framework](#), you can use the CloudFormation templates provided here.

Architecture overview

Here is a diagram of our architecture and a brief summary of what you are going to set up.



The sample CloudFormation templates provision the network infrastructure and all the components shown in the architecture diagram. I broke the CloudFormation templates into the following three stacks.

1. CloudFormation template to set up VPC, subnets, route tables, internet gateway, NAT gateway, S3 gateway endpoint, [AWS Secrets Manager](#) interface endpoint, and other networking components.
2. CloudFormation template to set up an Amazon Linux bastion host in an Auto Scaling group to connect to the Aurora PostgreSQL DB cluster.
3. CloudFormation template to set up Aurora PostgreSQL DB cluster with master user password stored in AWS Secrets Manager and bootstrap the database using [AWS Lambda](#).

The stacks are integrated using exported output values. Using three different CloudFormation stacks instead of one nested stack gives you some flexibility. For example, you can choose to deploy the VPC and bastion host CloudFormation stacks once and Aurora PostgreSQL DB cluster CloudFormation stack multiple times in an AWS Region.

Best practices

The architecture built by these CloudFormation templates supports AWS best practices for high availability and security.

The VPC CloudFormation template takes care of the following:

1. Sets up three Availability Zones for high availability and disaster recovery. Availability Zones are geographically distributed within a Region and spaced for best insulation and stability in the event of a natural disaster.
2. Provisions one public subnet and one private subnet for each Availability Zone. We recommend using public subnets for external-facing resources and private subnets for internal resources to reduce the risk of exfiltration of data.
3. Creates and associates [network ACLs](#) with default rules to the private and public subnets. AWS recommends using network ACLs as firewalls to control inbound and outbound traffic at the subnet level. These network ACLs provide individual controls that you can customize as a second layer of defense.
4. Creates and associates independent routing tables for each of the private subnets, which you can configure as necessary to control the flow of traffic within and outside the Amazon VPC. The public subnets share a single routing table, because they all use the same internet gateway as the sole route to communicate with the internet.
5. Creates a [NAT gateway](#) in each of the three public subnets for high availability. NAT gateways offer major advantages over NAT instances in terms of deployment, availability, and maintenance. NAT gateways allow instances in a private subnet to connect to the internet or other AWS services while they prevent the internet from initiating a connection with those instances.
6. Creates an [S3 VPC endpoint](#), which provides resources in private subnets; e.g., AWS Lambda, to communicate with Amazon S3 in a secure and reliable way.
7. Creates AWS Secrets Manager [interface VPC endpoint](#), which provides Lambda resources in private subnets to communicate in a secure way with Secrets Manager service without requiring internet access.

The Amazon Linux bastion host CloudFormation template takes care of the following:

1. Creates an [auto scaling group](#) spread across the three public subnets set up by the VPC CloudFormation template. The Auto Scaling group ensures that the Amazon Linux bastion host is always available in one of the three Availability Zones.
2. Sets up [Elastic IP address](#) and associates with the Amazon Linux bastion host. Elastic IP address makes it easier to remember and allow these IP addresses from on-premises firewalls. If an instance is terminated and the Auto Scaling group launches a new instance in its place, the existing Elastic IP address is re-associated with the new instance. This ensures that the same trusted Elastic IP address is used at all times.

3. Sets up an [EC2 security group](#) and associates with the Amazon Linux bastion host. This allows locking down access to the bastion hosts to known CIDR scopes and port for ingress.
4. Creates an [Amazon CloudWatch Logs](#) log group to hold the Amazon Linux bastion host's shell history logs and sets up a CloudWatch metric to keep track of SSH command counts. This helps in security audits by allowing you to check when and by whom the bastion host is being accessed.
5. Creates a CloudWatch alarm to monitor the CPU on the bastion host and send SNS notification when the alarm is triggered.

The Aurora PostgreSQL DB cluster template takes care of the following:

1. Creates a Multi-AZ Aurora DB cluster with a primary instance and an Aurora replica in two separate Availability Zones for a production or pre-production type of environment. This is recommended by AWS for high availability. Aurora automatically fails over to an Aurora Replica in case the primary DB instance becomes unavailable.
2. Places the Aurora DB cluster in the private subnets according to AWS security best practice. To access the DB cluster use the Amazon Linux bastion host, which is set up by the Linux bastion host CloudFormation template.
3. Sets up an EC2 security group and associates with the Aurora DB cluster. This allows locking down access to the DB cluster to known CIDR scopes and port for ingress.
4. Generates a random master user password by using AWS Secrets Manager and associates this password with the Aurora DB cluster. A Python-based Lambda function backed by CloudFormation custom resource configures automatic password rotation every 30 days using AWS Secret Manager provided [Serverless Application Repository](#). AWS recommends rotating passwords regularly to prevent unauthorized access in case the password is compromised.
5. Creates a DB cluster parameter group with the following setting and associates with Aurora DB cluster.

These DB cluster parameters are provided as a general guide. You should review and customize them to suit your needs.

| Parameter | Value | Description |
|-------------------------------|-------|---|
| rds.force_ssl | 1 | This ensures that connections to the Aurora instance use SSL. |

6. Creates a DB parameter group with the following settings and associates with Aurora DB instances.

These DB instance parameters are provided as general guidance. You should review and customize them to suit your needs.

| Parameter | Value | Description |
|---|--|--|
| shared_preload_libraries | auto_explain, pg_stat_statements, pg_hint_plan, pgaudit | auto_explain module provides a means for logging execution plans of slow statements automatically, without having to run EXPLAIN by hand. pg_stat_statements module provides a means for tracking execution statistics of all SQL statements executed by a server. pg_hint_plan gives PostgreSQL ability to manually force some decisions in execution plans. pgaudit provides detailed session and/or object audit logging via the standard logging facility provided by PostgreSQL. |
| log_statement | "ddl" | Controls which SQL statements are logged |
| log_connections | 1 | Causes each attempted connection to the server to be logged, as well as successful completion of client authentication. |
| log_disconnections | 1 | Causes session terminations to be logged. |
| log_lock_waits | 1 | Controls whether a log message is produced when a session waits longer than deadlock_timeout to acquire a lock. |
| log_min_duration_statement | 5000 | Causes the duration of each completed statement to be logged if the statement ran for at least the specified number of milliseconds. |
| auto_explain.log_min_duration | 5000 | Minimum statement execution time, in milliseconds, that causes the statement's plan to be logged. |
| auto_explain.log_verbose | 1 | Controls whether verbose details are printed when an execution plan is logged |
| log_rotation_age | 1440 | When logging_collector is enabled, this parameter determines the maximum lifetime in minutes of an individual log file. After this many minutes have elapsed, a new log file is created. |
| log_rotation_size | 102400 | When logging_collector is enabled, this parameter determines the maximum size in kilobytes of an individual log file. After this many kilobytes have been emitted into a log file, a new log file is created. |
| rds.log_retention_period | 10080 | Amazon RDS deletes PostgreSQL logs that are older than the specified value in minutes. |
| random_page_cost | 1 | Makes index scan looks more attractive to the planner compared to table sequential scans |
| track_activity_query_size | 16384 | Specifies the number of bytes reserved to track the currently executing command for each active session, for the "pg_stat_activity.query" field. |
| idle_in_transaction_session_timeout | 7200000 | Terminate any session with an open transaction that has been idle for longer than the specified duration in milliseconds. |

| | | |
|-----------------------------------|------------------|--|
| statement_timeout | 7200000 | Abort any statement that takes more than the specified number of milliseconds, starting from the time the command arrives at the server from the client. |
| search_path | "\$user",public' | This variable specifies the order in which schemas are searched when an object (table, data type, function, etc.) is referenced by a simple name with no schema specified. |

- Creates a customer-managed encryption key using [AWS KMS](#) and enables Aurora DB cluster [encryption](#) at rest by using that key.
- Configures backup retention to 35 days and 7 days for production and non-production type of environment respectively. This is to ensure the production database can be recovered to any point in time in the last 35 days. Similarly, the non-production database can be recovered to any point in time in the last 7 days.
- Automatically enables [Enhanced Monitoring](#) for production type of environment with one-second granularity so that you can view real time OS metrics and troubleshoot database performance issues.
- Automatically enables [Performance Insight](#) and configures performance insight data encryption using a customer-managed AWS KMS encryption key. Sets performance insight data retention to 2 years and 7 days for production and non-production type of environments respectively. Performance insight helps to quickly detect database performance problems and determine when and where to take action.
- Enables [IAM database authentication](#) to use authentication token instead of database password for database login using AWS Identity and Access Management (IAM). When you use IAM database authentication, network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
- Disables automatic minor version upgrade option for production type of environment, so that you can perform the upgrade according to your schedule when all non-production environments are upgraded and you have successfully tested application functionality.
- Configures CloudWatch alarms for key [CloudWatch metrics](#) like CPUUtilization, MaximumUsedTransactionIDs and FreeLocalStorage for the Aurora database instances and sends SNS notification when the alarm is triggered.
- Configures RDS event notifications for db-cluster, db-instance and db-parameter-group source types to notify you when an important event is generated.
- Attaches mandatory [common tags](#) to the Aurora cluster and Database instances. AWS recommends assigning tags to your cloud infrastructure resources to manage resource access control, cost tracking, automation, and organization.
- Bootstraps the Aurora DB cluster upon creation by using a Lambda function. This creates pg_stat_statements extension for tracking execution statistics of all SQL statements executed by the server. The Lambda function also creates a pgaudit extension, which provides session and object-level auditing capabilities.

Prerequisites

Before setting up the CloudFormation stacks, note the following prerequisites.

- You must have an AWS account and an [AWS Identity and Access Management \(IAM\)](#) user with sufficient permissions to interact with the AWS Management Console and services listed in the Architecture overview section. Your IAM permissions must also include access to create IAM roles and policies created by the AWS CloudFormation template.
- The VPC CloudFormation stack requires three Availability Zones for setting up the public and private subnets. Make sure to select an [AWS Region](#) that has at least three Availability Zones.
- Create an [EC2 key pair](#) using Amazon EC2 console in the AWS Region where you are planning to set up the CloudFormation stacks. Make sure to save the private key, as this is the only time you can do this. You use this EC2 key pair as an input parameter while setting up the Amazon Linux bastion host CloudFormation stack.

4. Create an Amazon S3 bucket and upload the sample database bootstrap AWS Lambda zip file, as follows.
 - a. Create an S3 bucket in the same AWS Region, where you planning to set up the Aurora PostgreSQL DB cluster. No additional settings are required within the bucket configuration.
 - b. In the bucket, create a folder (prefix).
 - c. Download the sample AWS Lambda zip file **dbbootstrap.zip** from [GitHub](#).
 - d. Upload the zip file to the S3 bucket folder. No additional settings are required on the upload screen.

You use the S3 bucket name along with the S3 key prefix as input parameters while setting up the Aurora DB cluster CloudFormation Stack. This Zip file is used to create a Lambda function written in Python 2.7, which uses PyGreSQL module to connect to the Aurora PostgreSQL cluster and bootstrap it upon creation.

PyGreSQL is released under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT license. For details, refer [LICENSE.txt](#) in GitHub.

Set up the resources using AWS CloudFormation

These CloudFormation templates are provided as a general guide. You should review and customize them to suit your needs. Some of the resources deployed by these stacks incur costs as long as they are in use.

Setup VPC, subnets, and other networking components

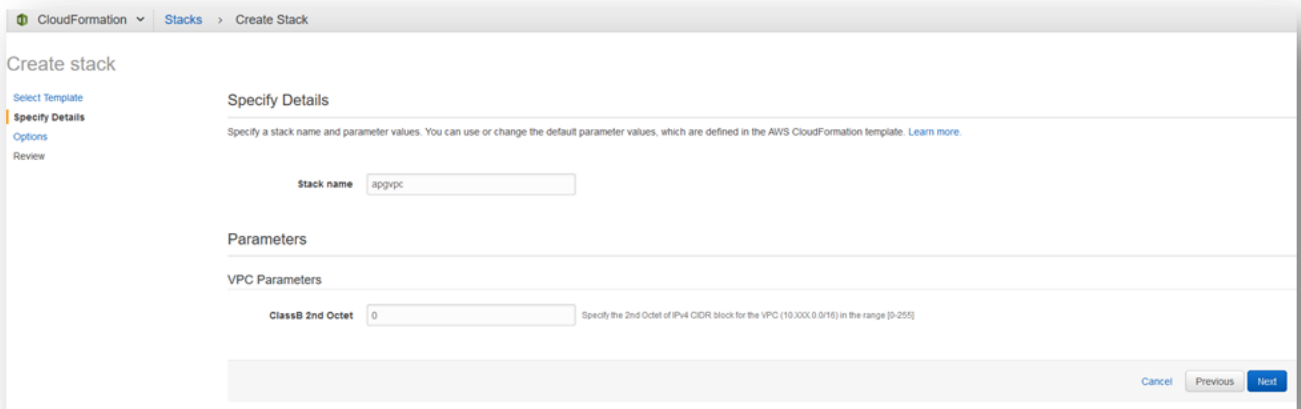
Launch Stack 

1. Choose **Launch Stack**. This button automatically launches the AWS CloudFormation service in your AWS account with a template. You are prompted to sign-in if needed. You can view the CloudFormation template from within the console if required.
2. Choose the AWS Region where you want to create the stack on top right of the screen and then choose **Next**.

This CloudFormation stack requires three Availability Zones for setting up the public and private subnets. Select an [AWS Region](#) that has at least three Availability Zones.

3. The CloudFormation stack requires a few parameters, as shown in the following screenshot.
 - **Stack name** : Enter a meaningful name for the stack, e.g., *apgvpc*
 - **ClassB 2nd Octet** : Specify the 2nd Octet of IPv4 CIDR block for the VPC (10.XXX.0.0/16). You can specify any number between and including 0 to 255, e.g., specify 33 to create a VPC with IPv4 CIDR block 10.33.0.0/16.

To learn more about VPC and subnet sizing for IPv4, refer [AWS VPC Documentation](#).



4. After entering all the parameter values, choose **Next**.
5. On the next screen, enter any required tags, an [IAM](#) role, or any [advanced options](#), and then choose **Next**.
6. Review the details on the final screen, and then choose **Create** to start building the networking resources.

It takes about four minutes to create the stack. Check the AWS CloudFormation Resources section to see the Physical IDs of the various components set up by this stack.

Now let's set up the Amazon Linux bastion host, which you use to login to the Aurora PostgreSQL DB cluster.

Setup Amazon Linux bastion host



1. Choose **Launch Stack**. This button automatically launches the AWS CloudFormation service in your AWS account with a template to launch.
2. Choose the AWS Region where you want to create the stack on top right of the screen and then choose **Next**.
3. The CloudFormation stack requires a few parameters, as shown in the following screenshots.
 - **Stack name** : Enter a meaningful name for the stack, e.g., *apgbastion*
 - **ParentVPCStack** : Enter the CloudFormation stack name for the VPC stack that was set up in the previous step. Refer to the CloudFormation dashboard in AWS Console to get this, e.g., *apgvpc*
 - **Allowed Bastion External Access CIDR** : Enter allowed CIDR block in the x.x.x.x/x format for external SSH access to the bastion host
 - **Key Pair Name** : Select the key pair name that was set up in the Prerequisites section.
 - **Bastion Instance Type** : Select Amazon EC2 instance type for the bastion instance
 - **LogsRetentionInDays** : Specify the number of days you want to retain CloudWatch log events for the bastion host
 - **SNS Notification Email** : Enter the Email notification list used to configure a SNS topic for sending CloudWatch alarm notifications
 - **Bastion Tenancy** : Select the VPC Tenancy in which the bastion host is launched
 - **Enable Banner** : Select if you want a banner to be displayed when connecting via SSH to the bastion
 - **Bastion Banner** : Use Default or provide AWS S3 location for the file containing Banner text to be displayed upon login
 - **Enable TCP Forwarding** : Select if you want to Enable/Disable TCP Forwarding. Setting this value to **true** enables TCP forwarding (SSH tunneling). This can be very useful but it is also a security risk, so I recommend that you keep the default (disabled) setting unless required.
 - **Enable X11 Forwarding** : Select if you want to Enable/Disable X11 Forwarding. Setting this value to **true** enables X Windows over SSH. X11 forwarding can be very useful but it is also a security risk, so I recommend that you keep the default (disabled) setting unless required.
 - **Custom Bootstrap Script** : Optional. Specify custom bootstrap script AWS S3 location to run during bastion host setup
 - **AMI override** : Optional. Specify a AWS Region-specific image to use for the instance

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name

Parameters

Network Configuration

ParentVPCStack Stack name of parent VPC stack based on VPC-3AZs yaml template. Refer Cloudformation dashboard in AWS Console to get this.

Allowed Bastion External Access CIDR Allowed CIDR block in the x.x.x.x/x format for external SSH access to the bastion host

Amazon EC2 Configuration

Key Pair Name Enter a Public/private key pair. If you do not have one in this AWS Region, create it before continuing

Bastion Instance Type Amazon EC2 instance type for the bastion instance. t2 instance types are not supported for dedicated VPC tenancy (option below).

LogsRetentionInDays Specify the number of days you want to retain log events

SNS Notification Email The Email notification list is used to configure a SNS topic for sending cloudwatch alarm notifications

Linux Bastion Configuration

Bastion Tenancy VPC Tenancy in which bastion host will be launched. Options: 'dedicated' or 'default'

Enable Banner To include a banner to be displayed when connecting via SSH to the bastion, set this parameter to true

Bastion Banner Banner text to display upon login. Use default or provide AWS S3 location for the file containing Banner text.

Enable TCP Forwarding Enable/Disable TCP Forwarding

Enable X11 Forwarding Enable/Disable X11 Forwarding

Other parameters

Custom Bootstrap Script Optional. Specify custom bootstrap script AWS S3 location to run during bastion host setup

AMI override Optional. Specify a region specific image to use for the instance

4. After entering all the parameter values, choose **Next**.
5. On the next screen, enter any required tags, an [IAM](#) role, or any [advanced options](#), and then choose **Next**.
6. Review the details on the final screen, **Check** the box for **"I acknowledge that AWS CloudFormation might create IAM resources"** and then choose **Create** to start building the networking resources.

It takes about five minutes to create the stack. Check the AWS CloudFormation Resources section to see the Physical IDs of the various components set up by this stack.

You are now ready to set up the Aurora PostgreSQL DB cluster.

Setup Aurora PostgreSQL DB cluster

Launch Stack 

1. Choose **Launch Stack**. This button automatically launches the AWS CloudFormation service in your AWS account with a template to launch.
2. Choose the AWS Region where you want to create the stack on top right of the screen and then choose **Next**.
3. The CloudFormation stack requires a few parameters, as shown in the following screenshots.
 - **Stack name** : Enter a meaningful name for the stack, e.g., *apgprod-sampleapp*
 - **EnvironmentStage** : Select the environment stage (dev, test, pre-prod, prod) of the Aurora PostgreSQL DB cluster. If you specify “prod” option for this parameter, DB backup retention is set to 35 days, enhanced monitoring is turned on with one-second granularity and automatic minor version upgrade is disabled. If you specify either “prod” or “pre-prod” option for this parameter, a Multi-AZ Aurora DB cluster with a primary instance and an Aurora replica in two separate Availability Zones is created.
 - **DBName** : Enter database name, e.g., *sampledb*
 - **DBPort**: Enter TCP/IP Port for the Database Instance
 - **DBUsername** : Enter Database master username, e.g., *master*
 - **DBInstanceClass** : Select Database Instance Class
 - **DBEngineVersion** : Select Database Engine Version
 - **DBSnapshotName** : Optional. Enter the DB Snapshot ID to restore. Leave this blank if you are not restoring from a snapshot.
 - **NotificationList** : Enter the Email notification list that is used to configure a SNS topic for sending CloudWatch alarm and RDS Event notifications
 - **LambdaBootStrapS3Bucket** : Specify the S3 bucket name that was created in the Prerequisites section, e.g., *apgbootstrapscrip-us-east-1*
 - **LambdaBootStrapS3Key** : Specify the S3 Key (containing the folder that was created in the Prerequisites section and the zip folder name) where the Database bootstrap script is stored, e.g., *lambda/dbbootstrap.zip*
 - **ParentVPCStack** : Provide Stack name of parent VPC stack. Refer CloudFormation dashboard in AWS Console to get this.
 - **ParentSSHBastionStack** : Provide Stack name of parent Amazon Linux bastion host stack. Refer CloudFormation dashboard in AWS Console to get this.
 - **Application** : The Application tag is used to designate the application of the associated AWS resource. In this capacity, application does not refer to an installed software component, but rather the overall business application that the resource supports.
 - **ApplicationVersion** : The ApplicationVersion tag is used to designate the specific version of the application.
 - **ProjectCostCenter** : The ProjectCostCenter tag is used to designate the cost center associated with the project of the given AWS resource.
 - **ServiceOwnersEmailContact** : The ServiceOwnersEmailContact tag is used to designate business owner(s) email address associated with the given AWS resource for sending outage or maintenance notifications.
 - **Confidentiality** : The Confidentiality tag is used to designate the confidentiality classification of the data that is associated with the resource.
 - **Compliance** : The Compliance tag is used to specify the Compliance level for the AWS resource.

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name

Parameters

Environment

EnvironmentStage The environment tag is used to designate the Environment Stage of the associated AWS resource.

DB Parameters

DBName Database Name

DBPort TCP/IP Port for the Database Instance

DBUsername Database master username

DBInstanceClass Database Instance Class

DBEngineVersion Select Database Engine Version

DBSnapshotName Optional. DB Snapshot ID to restore database. Leave this blank if you are not restoring from a snapshot.

NotificationList The Email notification list is used to configure a SNS topic for sending cloudwatch alarm and RDS Event notifications

LambdaBootstrapS3Bucket Optional. Specify S3 bucket name for e.g. appbootstrapscripts where Lambda DB Bootstrap Python 3.6 script is stored.

LambdaBootstrapS3Key Optional. Specify S3 key for e.g. lambda/dbbootstrap.zip where Lambda DB Bootstrap Python 3.6 script is stored.

Networking

ParentVPCStack Provide Stack name of parent VPC stack based on VPC-3AZs yaml template. Refer Cloudformation dashboard in AWS Console to get this.

ParentSSHBastionStack Provide Stack name of parent Amazon Linux bastion host stack based on VPC-SSH-Bastion yaml template. Refer Cloudformation dashboard in AWS Console to get this.

Mandatory Tags

Application The Application tag is used to designate the application of the associated AWS resource. In this capacity application does not refer to an installed software component, but rather the overall business application that the resource supports.

ApplicationVersion The ApplicationVersion tag is used to designate the specific version of the application. Format should be in the Pattern - "###"

ProjectCostCenter The ProjectCostCenter tag is used to designate the cost center associated with the project of the given AWS resource.

ServiceOwnersEmailContact The ServiceOwnersEmailContact tag is used to designate business owner(s) email address associated with the given AWS resource for sending outage or maintenance notifications

Confidentiality The Confidentiality tag is used to designate the confidentiality classification of the data that is associated with the resource.

Compliance The Compliance tag is used to specify the Compliance level for the AWS resource.

[Cancel](#) [Previous](#) [Next](#)

- After entering the all the parameter values, choose **Next**.
- On the next screen, enter any required tags, an [IAM](#) role, or any [advanced options](#), and then choose **Next**.

- Review the details on the final screen, **Check** the box for **I acknowledge that AWS CloudFormation might create IAM resources** and then choose **Create** to start building the networking resources.

It takes about 25 minutes to create the stack. This stack internally launches another CloudFormation stack to set up the Secrets Manager provided Lambda function, which is used to rotate the Aurora PostgreSQL DB cluster master user password. Check the AWS CloudFormation Resources section to see the Physical IDs of the various components set up by these stacks.

Now, let's login to the Aurora PostgreSQL DB cluster and run some basic commands.

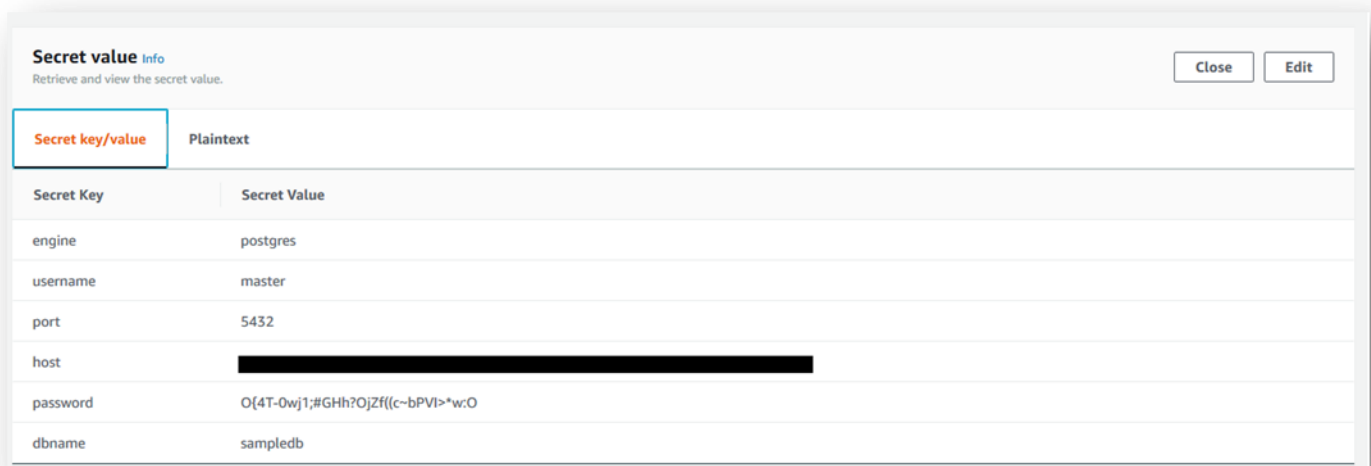
Login to Aurora DB cluster using the Amazon Linux bastion host

Note: The following instructions assume that you have a Linux computer and use SSH client to connect to the bastion host. For details on how to connect using various clients, refer [AWS Documentation](#).

- Move the private key of the EC2 key pair (that you had saved in the Prerequisites section) to a location on your SSH Client, where you are connecting to the Amazon Linux bastion host.
- Change the permission of the private key using the following command, so that it's not publicly viewable.

```
chmod 400 <private key file name, e.g., apgbastion.pem>
```

- Go to CloudFormation Dashboard in AWS Console and select the Amazon Linux bastion host stack. Select the **Outputs** tab and note down the **SSHCommand** parameter value, which you use to SSH to the Amazon Linux bastion host.
- On the SSH client, change directory to the location where you have saved the EC2 private key and then copy/paste the **SSHCommand** Value to SSH to the Amazon Linux bastion host.
- Go to AWS Secrets Manager dashboard in AWS Console. Select the secret name from the list of secrets that matches your Aurora PostgreSQL DB cluster CloudFormation stack name. Scroll down the page and select **Retrieve secret value**. Note down the password under **Secret key/value**, which you use to login to the Aurora PostgreSQL database.



- On the CloudFormation Dashboard, select the Aurora DB cluster Stack. Select **Outputs** tab and note the **PSQLCommandLine** parameter value, which you use to login to the Aurora PostgreSQL database using [psql client](#).
- PostgreSQL binaries were already set up on the Amazon Linux bastion host by EC2 Auto Scaling Launch Configuration. Copy/paste the **PSQLCommandLine** value at the command prompt of the bastion host.

psql --host=ClusterEndpoint --port=Port --username=DBUsername --dbname=DBName

When prompted, enter the master user password that you copied in step 5 above.

- Run some basic commands as shown below to list installed extensions and databases. The database bootstrap script has added `pg_stat_statement` and `pgaudit` extension as you can see by running `\dx` command.

```
% ssh -i "bastion.pem" ec2-user@
#####
#                               #
#   AWS Lambda@Edge             #
#                               #
#-----#
#   Authorized access only!     #
#   Disconnect IMMEDIATELY if you are not an authorized user!!!         #
#   All actions will be monitored and recorded.                         #
#-----#
#####
[ec2-user@ ~]$ psql --host= --port=5432 --username=master --dbname=sampled
Password for user master:
psql (9.6.10, server 10.4)
WARNING: psql major version 9.6, server major version 10.
Some psql features might not work.
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

5432 master@sampledb-> \dx
                                List of installed extensions
  Name      | Version | Schema | Description
-----
 pg_stat_statements | 1.5     | public | track execution statistics of all SQL statements executed
 pgaudit      | 1.2     | public | provides auditing functionality
 plpgsql      | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)

5432 master@sampledb-> \l
                                List of databases
  Name      | Owner   | Encoding | Collate | Ctype   | Access privileges
-----
 postgres   | master  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 rdsadmin   | rdsadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | rdsadmin=CTc/rdsadmin
 sampledb   | master  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | rdsadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/rdsadmin +
             |          |          |             |             | rdsadmin=CTc/rdsadmin
 template1  | master  | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/master +
             |          |          |             |             | master=CTc/master
(5 rows)

5432 master@sampledb-> select count(*) from pg_stat_statements;
 count
-----
    59
(1 row)
```

Summary

In this document, I showed you how to deploy an [Amazon Aurora PostgreSQL](#) DB cluster based on AWS security and high availability best practices using AWS CloudFormation. I hope you find the sample CloudFormation templates helpful and encourage you to modify them to support your business' needs. You might also consider modifying the Python based database bootstrap script used to create the AWS Lambda function. In it you could include all the standard database commands that you typically run after setting up an Aurora PostgreSQL DB cluster.

Before you use the Aurora DB cluster to set up your application-related database objects, consider creating the following:

- An application schema.
- A user with full access to create and modify objects in the application schema.
- A user with read-write access to the application schema.
- A user with read-only access to the application schema.

Use the master user that was set up with the Aurora DB cluster to administer the DB cluster only. The user with full access to the application schema should be used to create and modify application related database objects. Your application should use the read-write user for storing, updating, deleting and retrieving data. Any reporting kind of application should use the read-only user. It's a database security best practice to grant the minimum privileges required to perform required database operations.

Review [AWS CloudTrail](#), [AWS Config](#) and [Amazon GuardDuty](#) services and configure them for your AWS account, according to AWS security best practices. Together these services help you monitor activity in your AWS account; assess, audit, and evaluate the configurations of your AWS resources; monitor malicious or unauthorized behavior and detect security threats against your AWS resources.

Some of the AWS resources deployed by the CloudFormation stacks in this document incur cost as long as they are in use. Delete the stacks if you no longer need them. To clean up your stacks, use the CloudFormation console to remove the three stacks you created in reverse order.