Get unlimited access to the best of Medium for less than $1/week. Become a member

Create Amazon Aurora DB Cluster for MySQL

ABHISHEK KUMAR · Follow
6 min read · Apr 15, 2021

2

Amazon Aurora is a fully managed relational database engine that is compatible with MySQL and PostgreSQL. **Aurora** has high-performance storage sub-systems, Hence MySQL and PostgreSQL databases engines can take advantage of fast distributed storage. The storage can automatically **go up to** 64 TB. Amazon Aurora also provides DB clustering and replication.

High Availability
When we set up the Aurora cluster, We will be having two endpoints.

Writer endpoint — which is used for reading and writing connections.
Reader endpoint — which is used for read-only connections.
We will also get the instance endpoints while we set up the aurora cluster.

If we are using instance endpoints in our applications, In case of failure we need to manually change the endpoints in the application code.

But with the Writer & reader endpoints, It is capable of managing DB instance failover automatically and also better than the instance endpoints.

If the primary DB instance fails, Aurora automatically failover to a new primary DB instance. It will either create a new primary DB instance or will promote the existing aurora replica to a new primary DB instance.

Creating amazon aurora DB cluster
Amazon Aurora DB cluster is compatible with both MySQL and PostgreSQL DB engines.

In this tutorial, We will set up the aurora DB cluster for the MySQL DB engine.

While creating a cluster, The DB cluster contains a primary writer DB instance and up to 15 reader DB instances (Read replica)

Read replicas can be either created in the same region as of master or we can create a read replica in different AWS regions with a latency of less than 1 second.

We can either use the existing VPC and the subnet group to create the Aurora DB cluster. Else we can,

1. **Create a VPC with Public and Private subnets.**

2. Create a Custom subnet group from the new VPC.

3. To create subnet groups, Login to the RDS console.

4. In the navigation pane, Choose Subnet groups

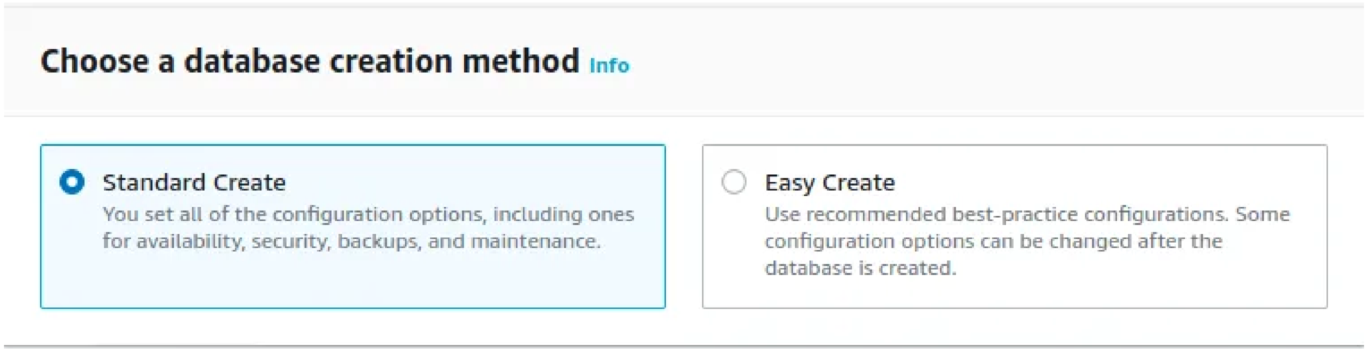
5. Click Create DB Subnet group provide a name for the subnet group and then choose the VPC that you have created.

For Add subnets, We must choose at least 1 subnet from different availability zones and then click Create

Now we start creating the aurora DB cluster for MySQL.

In the navigation panel, Choose Databases, Click Create database

In the Create database page, Choose a database creation method, For this guide, I am choosing Standard Create

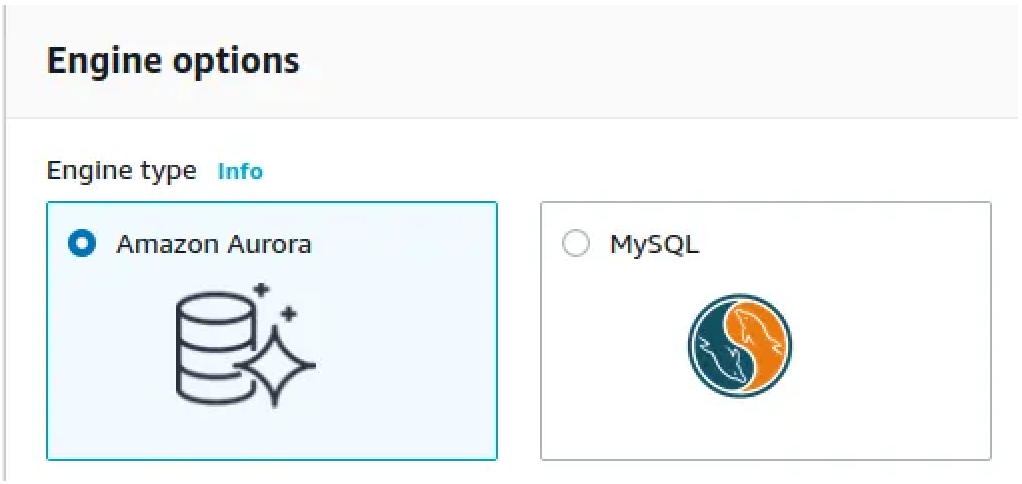


Choose a database creation method Info

[✓ Standard Create]
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy Create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

For Engine options, Choose Amazon Aurora



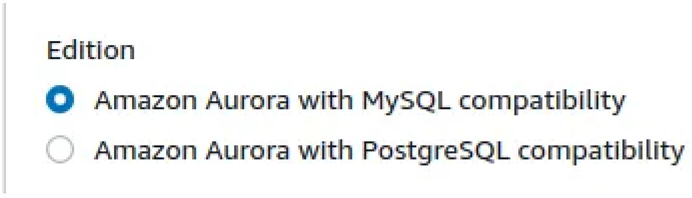
Engine options

Engine type Info

[✓ Amazon Aurora]

MySQL

For Edition, Choose Amazon Aurora with MySQL compatibility



Edition

[✓ Amazon Aurora with MySQL compatibility]

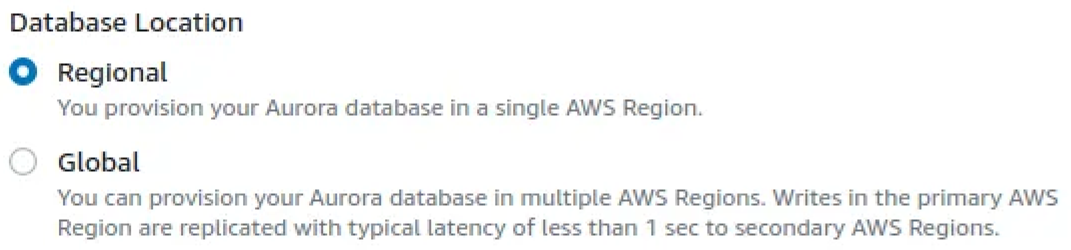
Amazon Aurora with PostgreSQL compatibility

For version, Choose the version of Aurora (MySQL) that is compatible with the application.

For Database Location,

If we choose Regional, Both the Writer and the reader endpoints will be created in the same AWS region.

If We choose Global, We have to Write the endpoint in one region and the replica (reader) in the different AWS regions.



Database Location

[✓ Regional]
You provision your Aurora database in a single AWS Region.

Global
You can provision your Aurora database in multiple AWS Regions. Writes in the primary AWS Region are replicated with typical latency of less than 1 sec to secondary AWS Regions.

For Database features,

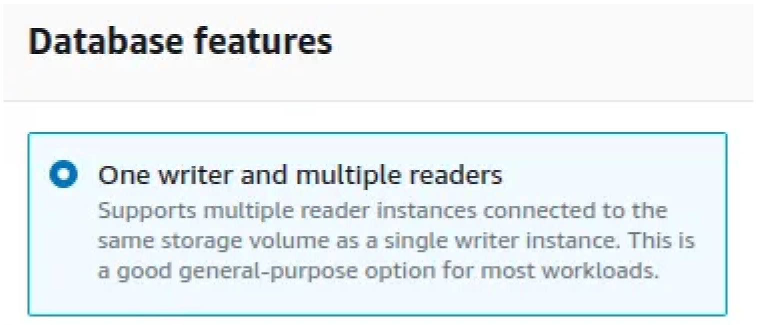
One writer and multiple readers, In this scenario, We can maintain high availability by locating aurora replicas in different availability zones. Hence aurora automatically failover to new primary DB instance by promoting the read replica to the primary DB instance, In case of the primary DB instance failure.

Also, all the reader instance connects to the same storage volume.

Multiple Writers, All the DB instances will have read and write capability and they will be connecting to the same storage volume. If continuous write availability is required, This option is preferred.

Serverless, It automatically scales up and down the DB instances as per the application needs. It will also help to stop and start the DB cluster automatically when not required.

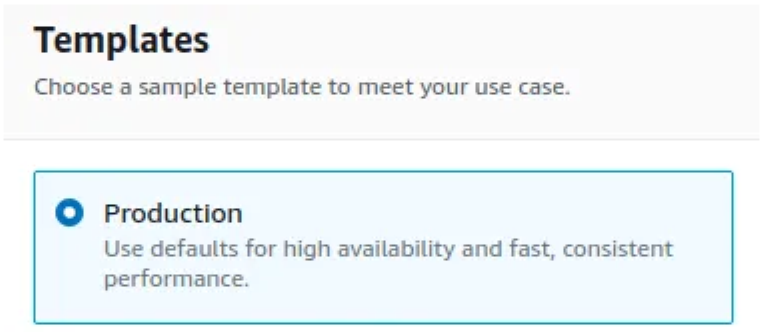
In this guide, I choose One writer and multiple readers



Database features

[✓ One writer and multiple readers]
Supports multiple reader instances connected to the same storage volume as a single writer instance. This is a good general-purpose option for most workloads.

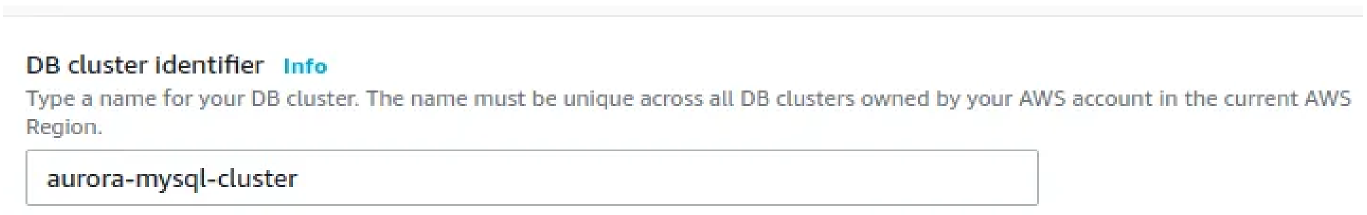
For Templates, Choose Production, For high availability, fast and consistent performance.



Templates
Choose a sample template to meet your use case.

[✓ Production]
Use defaults for high availability and fast, consistent performance.

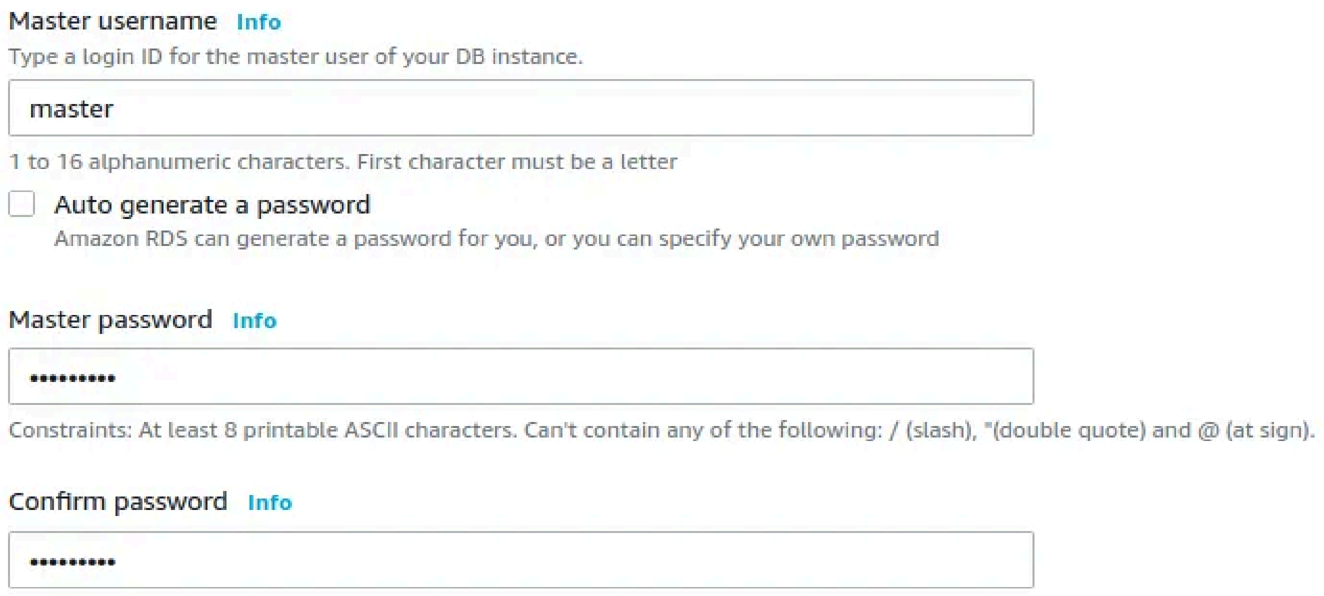
Under Settings, For DB cluster identifier, Provide a name for the DB cluster



DB cluster identifier Info
Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.

aurora-mysql-cluster

For Credentials settings, Set a Master username and password.



Master username Info

Type a login ID for the master user of your DB instance.

master

1 to 16 alphanumeric characters. First character must be a letter

☐ Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password

Master password Info

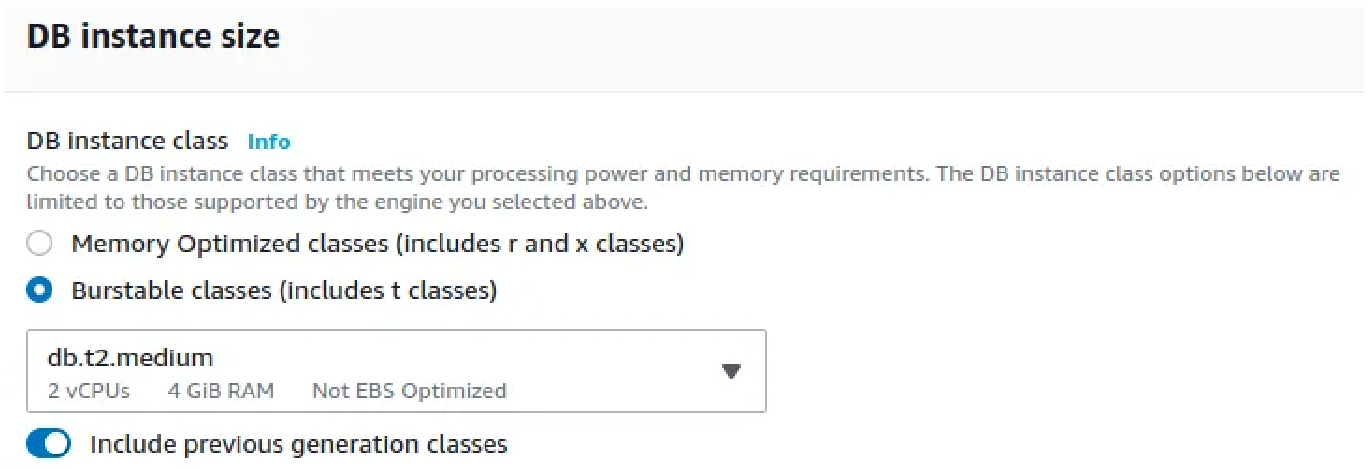
••••••••••

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote) and @ (at sign).

Confirm password Info

••••••••••

For DB instance size, Choose the DB instance class as per your requirement.



DB instance size

DB instance class Info

Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.

○ Memory Optimized classes (includes r and x classes)

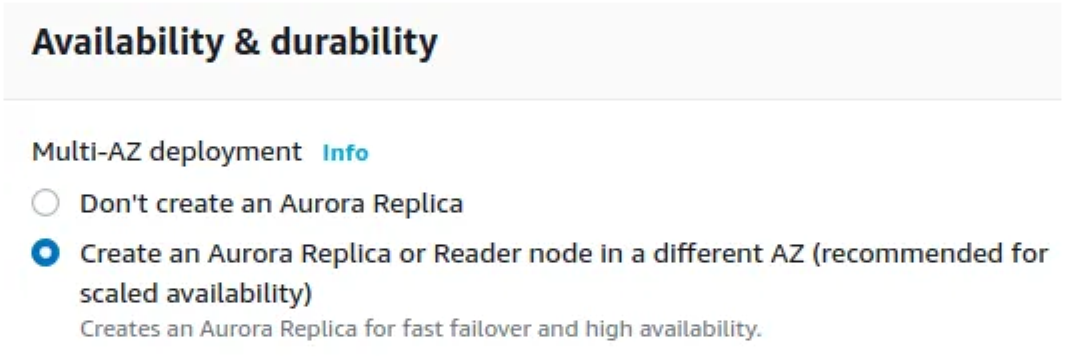
● Burstable classes (includes t classes)

db.t2.medium
2 vCPUs 4 GiB RAM Not EBS Optimized

Include previous generation classes: on

For Availability & durability, As a DB cluster, We have to create a replica in a different availability zone, which will be a standby replica.

It will automatically failover in case of the primary failure.



Availability & durability

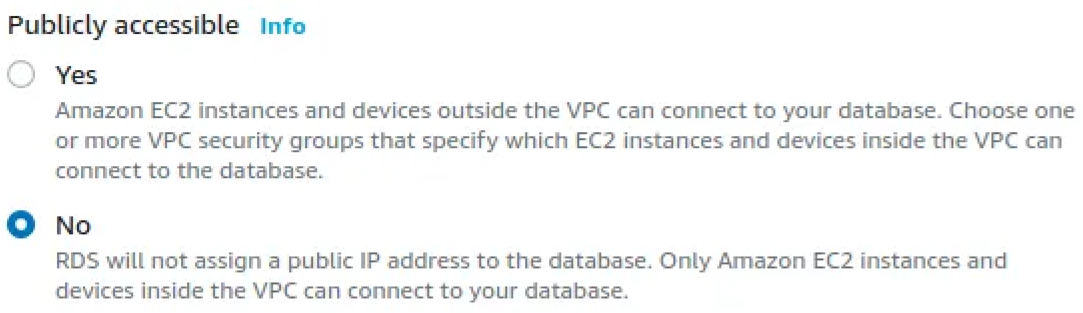
Multi-AZ deployment Info

- Don't create an Aurora Replica [✓ Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)]
  Creates an Aurora Replica for fast failover and high availability.

For Connectivity, Choose the VPC that you have created.

Under Additional connectivity configuration, Choose the Subnet group that you have created.

For Publicly accessible, Choose No, It is always recommended to host the databases private. As we don't want to expose the databases to the public network.

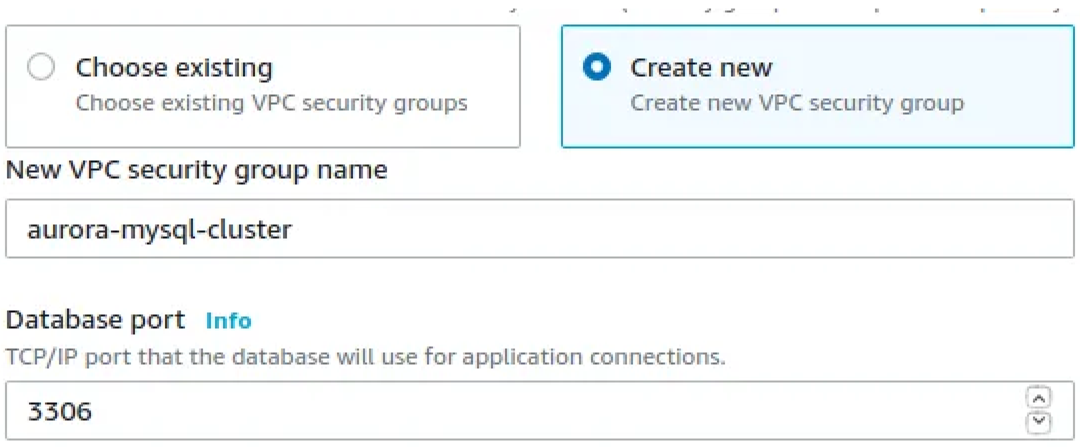


Publicly accessible Info

Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

[✓ No]
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

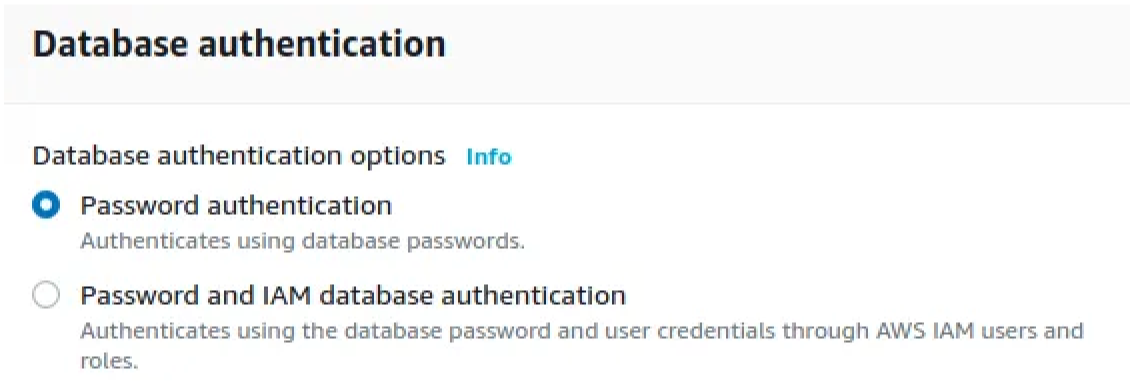
For the VPC security group, Create a new security group, Allowing port 3306 only to certain Instances or subnets.


Choose existing
Choose existing VPC security groups
[✓ Create new]
Create new VPC security group

New VPC security group name
aurora-mysql-cluster

Database port Info
TCP/IP port that the database will use for application connections.
3306

For Database authentication, Choose Password authentication


Database authentication

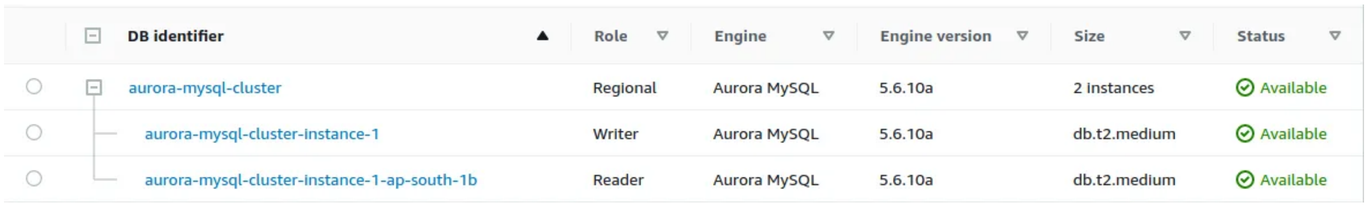
Database authentication options Info
[✓ Password authentication]
Authenticates using database passwords.
Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.

Under Additional configuration

If required, We can configure the Backup strategy, Encryption, Monitoring, and Logging.

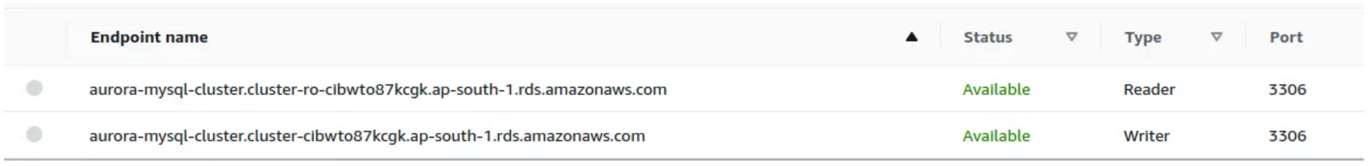
And finally, click Create database

The Aurora MySQL DB cluster is created.



	DB identifier	Role	Engine	Engine version	Size	Status
○	aurora-mysql-cluster	Regional	Aurora MySQL	5.6.10a	2 instances	✓ Available
○	aurora-mysql-cluster-instance-1	Writer	Aurora MySQL	5.6.10a	db.t2.medium	✓ Available
○	aurora-mysql-cluster-instance-1-ap-south-1b	Reader	Aurora MySQL	5.6.10a	db.t2.medium	✓ Available

If you click the identifier of the regional Role, Under Connectivity & Security, We can find Writer and reader endpoints.



	Endpoint name	Status	Type	Port
●	aurora-mysql-cluster.cluster-ro-cibwto87kcgk.ap-south-1.rds.amazonaws.com	Available	Reader	3306
●	aurora-mysql-cluster.cluster-cibwto87kcgk.ap-south-1.rds.amazonaws.com	Available	Writer	3306

We can use the reader and the writer endpoints with the applications.

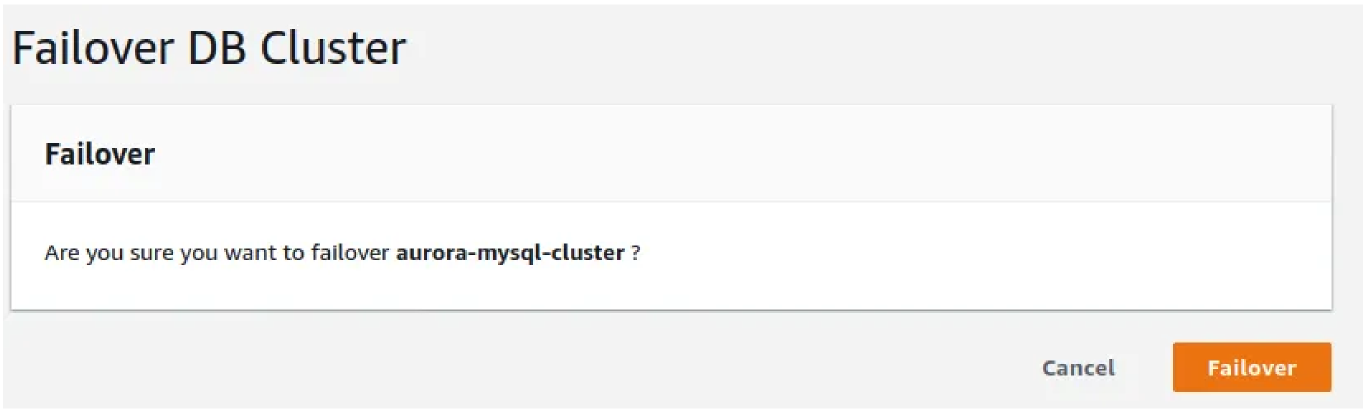
Again from the main console, If you click the DB identifier of the Writer, We can find the Writer Instance endpoint.

If you click the DB identifier of the Reader (which will be in a different availability zone), We can find the Reader Instance endpoint.

To manually test the failover for the DB cluster.

Choose the identifier of Writer Role and Under Actions, Choose Failover

It will ask for confirmation, Choose Failover

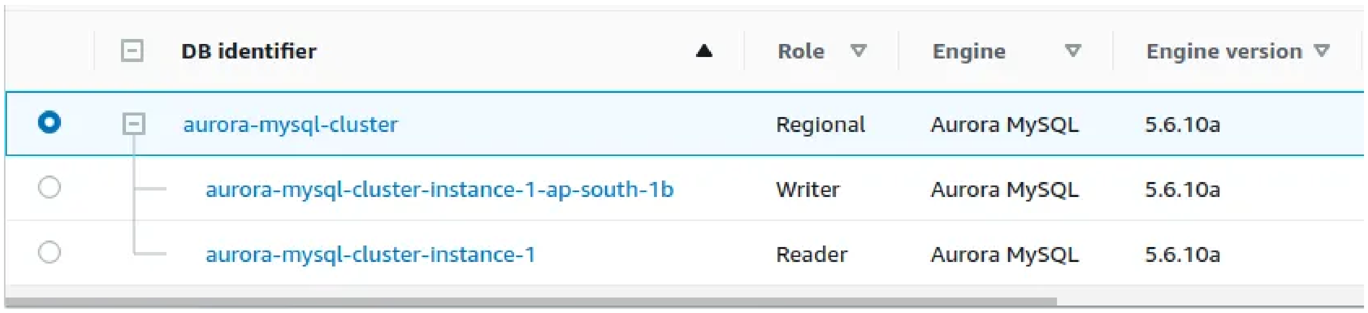


Failover DB Cluster

Failover

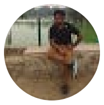
Are you sure you want to failover **aurora-mysql-cluster** ?

You can find the Role of the instances is changed, But the endpoints remain the same.



	DB identifier	Role	Engine	Engine version
●	aurora-mysql-cluster	Regional	Aurora MySQL	5.6.10a
○	aurora-mysql-cluster-instance-1-ap-south-1b	Writer	Aurora MySQL	5.6.10a
○	aurora-mysql-cluster-instance-1	Reader	Aurora MySQL	5.6.10a

AWS Aurora Create Aws Aurora



Written by ABHISHEK KUMAR

83 Followers · 31 Following

Follow

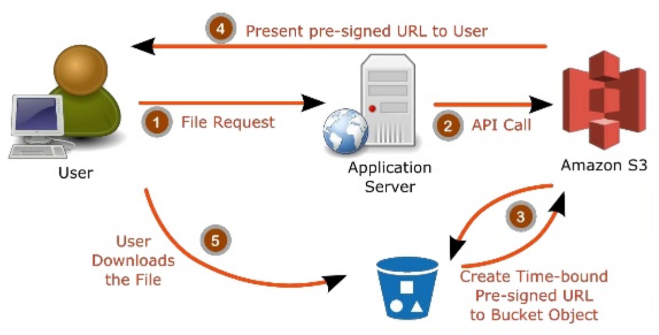
DevOps/Cloud | 2x AWS Certified | 1x Terraform Certified | 1x CKAD Certified | Gitlab

No responses yet

What are your thoughts?

Respond

More from ABHISHEK KUMAR


4 Present pre-signed URL to User
User
1 File Request
Application Server
2 API Call
Amazon S3
User Downloads the File
5
3
Create Time-bound Pre-signed URL to Bucket Object

ABHISHEK KUMAR

AWS S3 uploads using pre-signed URLs

How can I allow users to access objects in S3?

Apr 2, 2020 7

ABHISHEK KUMAR

MOUNT AWS SECRETS in EKS

Install CSI drivers

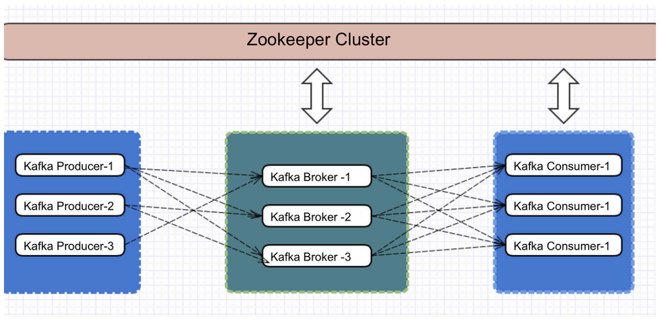
Sep 29, 2023 3 1

ABHISHEK KUMAR

Smart Contracts On The Blockchain: A deep dive in to Sma...

It is becoming difficult for the ordinary person to avoid the Blockchain buzz, and Businesse...

Apr 30, 2018 15 1



ABHISHEK KUMAR

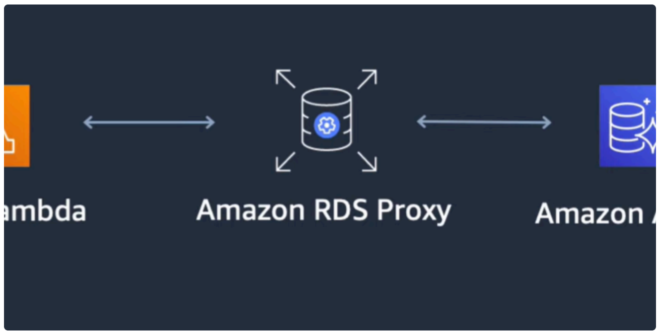
Kafka cluster with 3 nodes on a single machine.

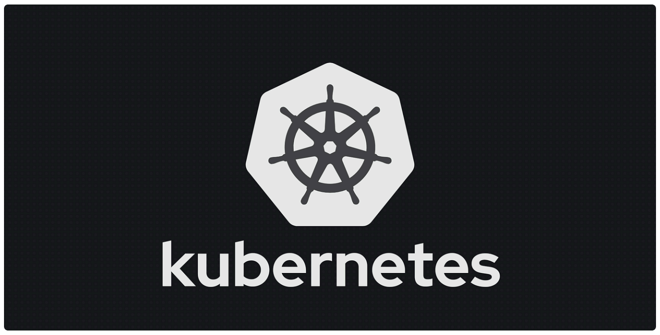
Kafka is a streaming platform has three key capabilities:

Jun 26, 2020 3

See all from ABHISHEK KUMAR

Recommended from Medium





In AWS in Plain English by Alice the Architect

Database Efficiency with Amazon RDS Proxy

In the dynamic world of cloud computing, where scalability and high availability are…

Oct 12, 2024 52

In Stackademic by Crafting-Code

I Stopped Using Kubernetes. Our DevOps Team Is Happier Than Ever

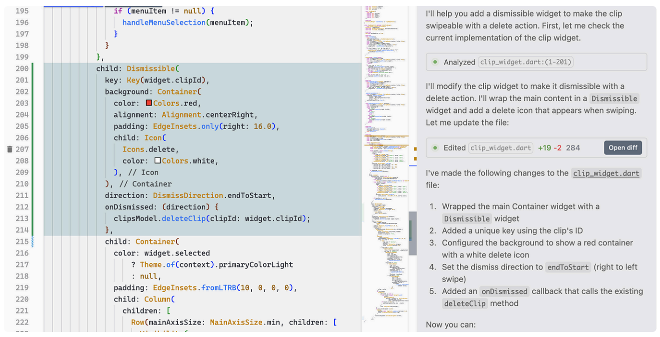
Why Letting Go of Kubernetes Worked for Us

Nov 19, 2024 5.2K 150

Lists

Natural Language Processing

1918 stories · 1573 saves



```
            if (menuItem != null) {
              handleMenuSelection(menuItem);
            }
          }
        },
        child: Dismissible(
          key: Key(widget.clipId),
          background: Container(
            color: Colors.red,
            alignment: Alignment.centerRight,
            padding: EdgeInsets.only(right: 16.0),
            child: Icon(
              Icons.delete,
              color: Colors.white,
            ), // Icon
          ), // Container
          direction: DismissDirection.endToStart,
          onDismissed: (direction) {
            clipsModel.deleteClip(clipId: widget.clipId);
          },
          child: Container(
            color: widget.selected
                ? Theme.of(context).primaryColorLight
                : null,
            padding: EdgeInsets.fromLTRB(10, 0, 0, 0),
            child: Column(
              children: [
                Row(mainAxisSize: MainAxisSize.min, children: [
```

I'll help you add a dismissible widget to make the clip swipeable with a delete action. First, let me check the current implementation of the clip widget.

Analyzed `clip_widget.dart:(1-201)`

I'll modify the clip widget to make it dismissible with a delete action. I'll wrap the main content in a `Dismissible` widget and add a delete icon that appears when swiping. Let me update the file:

Edited `clip_widget.dart` +19 -2 284 Open diff

I've made the following changes to the `clip_widget.dart` file:

1. Wrapped the main Container widget with a `Dismissible` widget
2. Added a unique key using the clip's ID
3. Configured the background to show a red container with a white delete icon
4. Set the dismiss direction to `endToStart` (right to left swipe)
5. Added an `onDismissed` callback that calls the existing `deleteClip` method

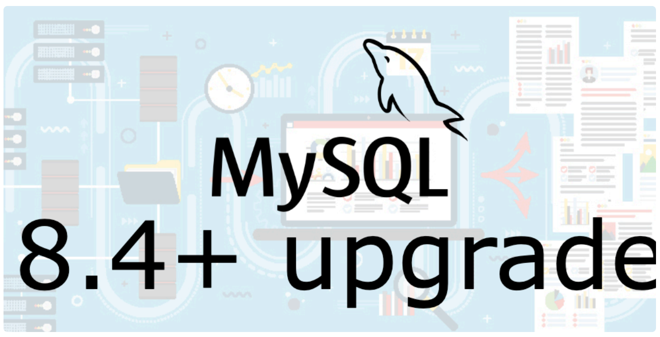
Now you can:

In Coding Beauty by Tari Ibaba

This new IDE just destroyed VS Code and Copilot without even…

Wow I never thought the day I stop using VS Code would come so soon…

Jan 17 2.2K 93



Alexander Sharov

Fix mysql_native_password not loaded errors in MySQL 8.4+

After upgrading to MySQL 8.4 or Bitnami's MySQL Helm chart version 11.0.0 and above,…

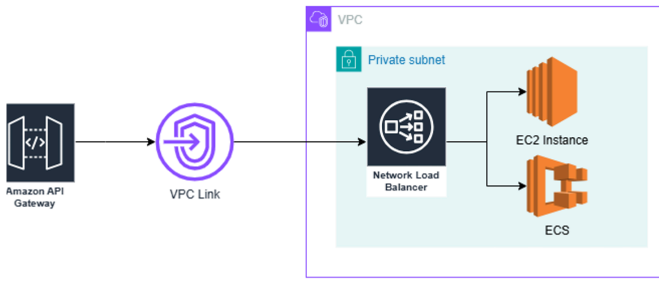
Oct 25, 2024 2 1

howtouselinux

The PostgreSQL Performance Playbook: Solving Slow Queries...

It was a typical Tuesday morning when the first sign of trouble came through.

Jan 18 33



Shiv Pal Singh Kaundal

How to Use AWS API Gateway VPC Link for Secure Access to Private...

In this guide, we'll explore how to use AWS API Gateway's VPC Link feature to securely...

Nov 16, 2024 12

See more recommendations