**Aws Database Migration Service [DMS]**

**Project Overview**

This project aims to demonstrate the migration of a MySQL database from an on-premises EC2 instance to an AWS RDS MySQL instance using AWS Database Migration Service (DMS). The goal is to set up two AWS accounts—one for the source database and one for the destination database—and facilitate data transfer.

**Purpose**

The purpose of this project is to demonstrate the process of migrating a MySQL database from an on-premises environment to an Amazon Web Services (AWS) cloud environment using AWS Database Migration Service (DMS). This migration will enhance data accessibility, scalability, and reliability while providing a cost-effective solution for managing database infrastructure.

**Components**

**On-Premises Environment:**

**EC2 Instance:** A virtual server where the MySQL database is hosted. This instance will run the MySQL server and store the initial data set.

**MySQL Database:** The source database containing the data that will be migrated. This will include the schema, tables, and records.

**AWS Cloud Environment:**

**AWS Accounts:** Two separate AWS accounts are set up—one for the source database (where the EC2 instance is located) and one for the destination (where the RDS instance will reside).

**RDS (Relational Database Service):** The target environment for the MySQL database in AWS. This service provides a managed database solution, enabling easier maintenance and scalability.

**AWS DMS (Database Migration Service):** A cloud service that facilitates the migration of databases. DMS will be used to transfer data from the on-premises MySQL database to the RDS MySQL instance.

**DMS Replication Instance:** A dedicated instance that performs the actual data migration tasks. It manages the connection to both the source and target databases.

**Endpoints:**

**Source Endpoint:** Configuration that specifies the connection details for the on-premises MySQL database.

**Target Endpoint:** Configuration that specifies the connection details for the AWS RDS MySQL database.

**Networking Components:**

**VPC (Virtual Private Cloud):** The network environment in AWS where the RDS instance will be deployed.

**Security Groups:** Firewalls that control inbound and outbound traffic to the EC2 instance and RDS instance, ensuring secure connectivity.

**Data Transfer:**

**Migration Task:** The configuration within DMS that defines the specifics of what data will be transferred and how (e.g., full data load, ongoing replication).

**Architecture Diagram**

**ON-PREMISE**

**MYSQL**

SOURCE ENDPOINT

**RDS**

TARGET ENDPOINT

**DMS**

**Prerequisites**

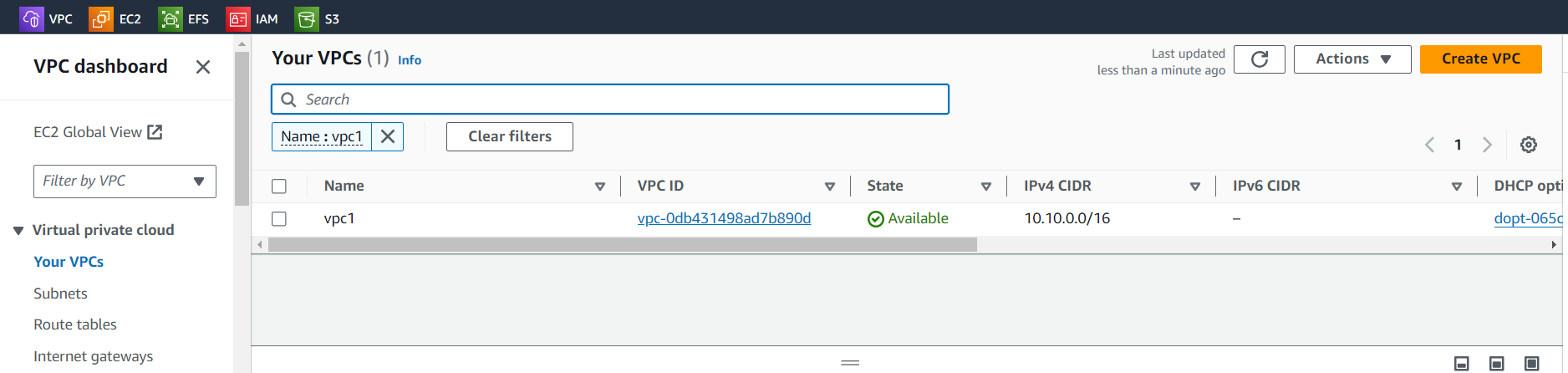
**AWS Account**: Ensure you have two AWS accounts set up: one for the source (on-premises) and one for the destination (RDS).

**IAM Roles/Permissions**: Create an IAM role in source (on-premises) with permissions to manage EC2 and RDS. Attach policies such as AmazonEC2FullAccess and AmazonRDSFullAccess.

**Setup Instructions**

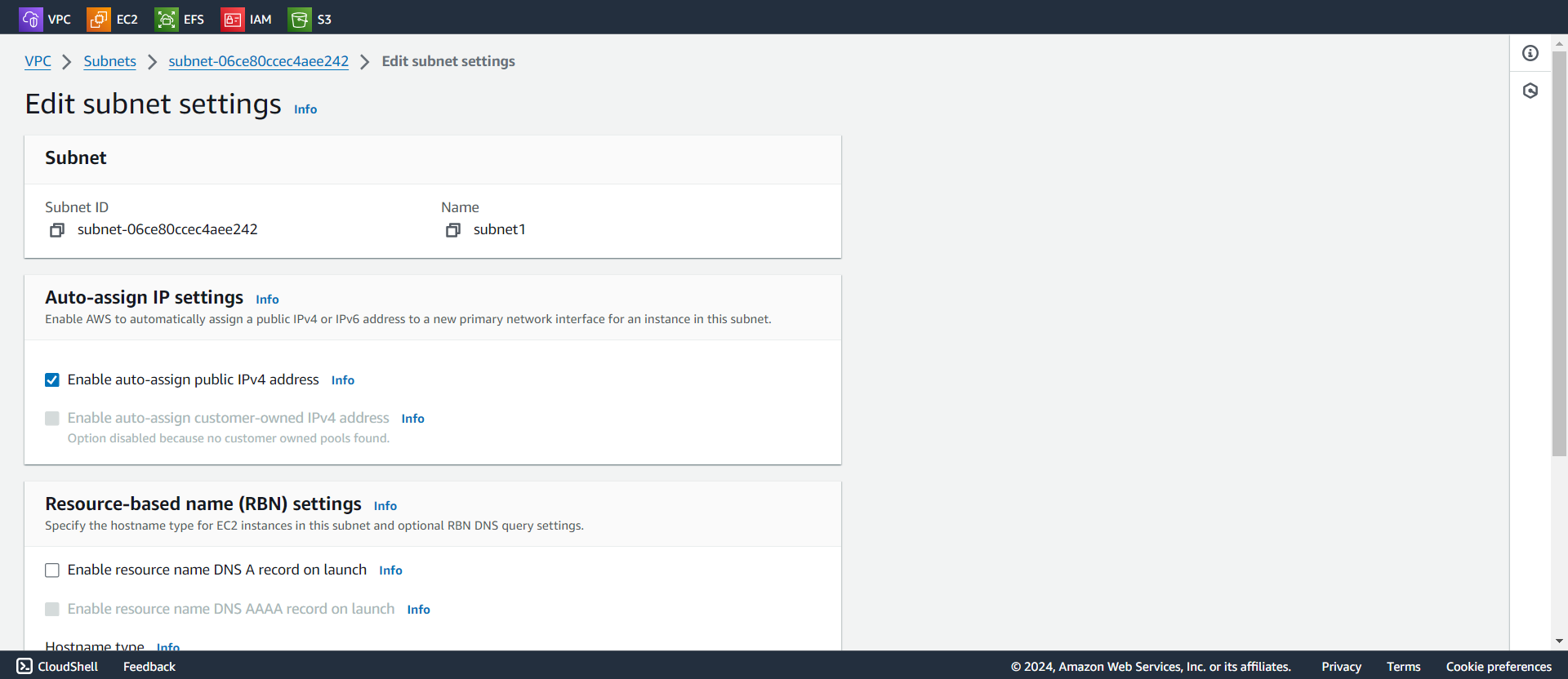
**Source Account Setup**

**VPC Creation**: After login in to the AWS management console click create VPC. Then provide a name and set IPv4 CIDR block: 10.10.0.0/16, set tendency as default and then click create.



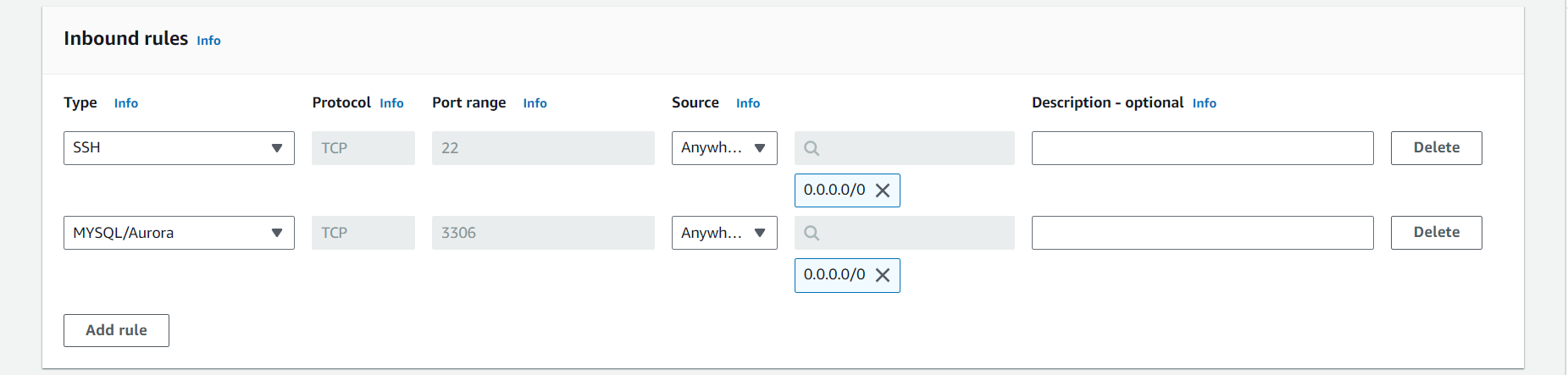
**Create Subnets:** Create a subnet by setting a name and choose the created VPC. Then set IPv4 CIDR block: 10.10.1.0/24 and availability zone

While creating subnets enable the auto-assign public IPv4 address

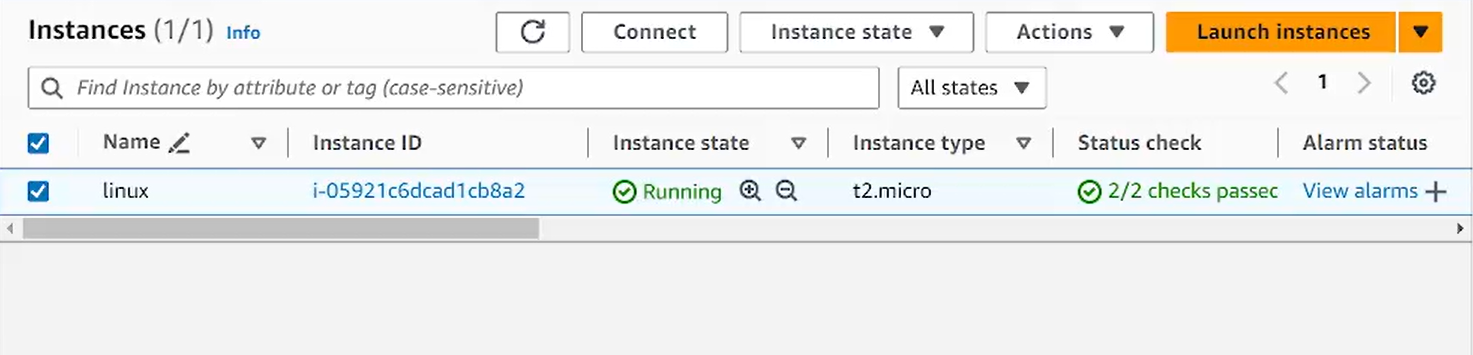


**Create an Internet Gateway: create an internet gateway then Attach it to VPC and enter the route table entries and associate it with the created subnet.**

**Create Security Groups: create a security group** choose the created VPC and add two inbound rules SSH and MYSQL/Aurora then set source as anywhere from IPv4



**Launch EC2 Instance:** Navigate to EC2 Dashboard and click Launch Instance. Then provide a name, choose created VPC, instance type, security group and IAM role.



**MySQL Installation:**

**Install MySQL:**

**sudo apt update**

**sudo apt install mysql-server**

**Secure MySQL Installation:**

**sudo mysql\_secure\_installation**

**sudo systemctl start mysql**

**sudo systemctl enable mysql**

**Create Database and Insert Records:**

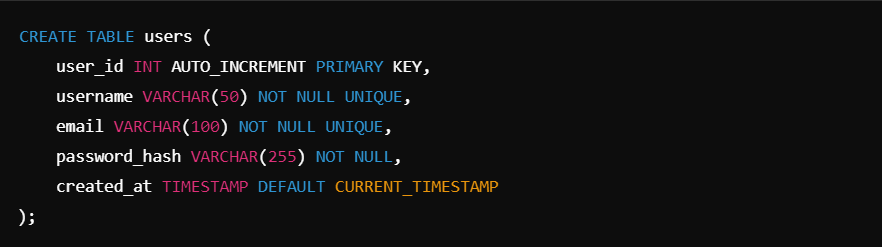
**sudo mysql**

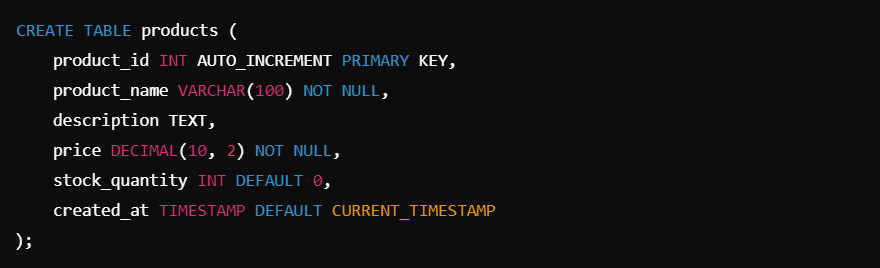
**CREATE USER 'virtualadmin'@'%' IDENTIFIED BY 'Admin@123456789!';**

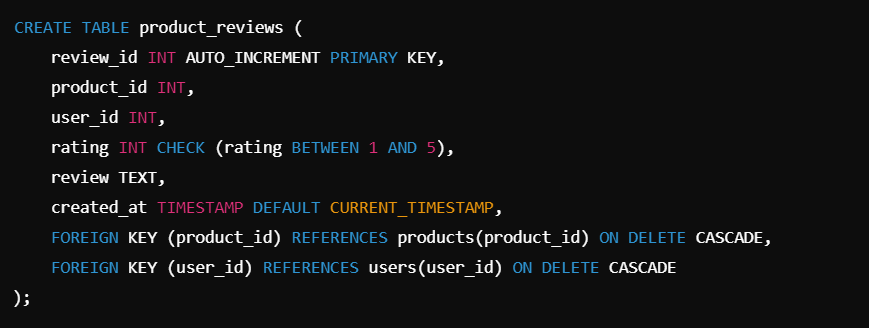
**GRANT ALL PRIVILEGES ON \*.\* TO 'virtualadmin'@'%'; FLUSH PRIVILEGES;**

**CREATE DATABASE migration\_project;**

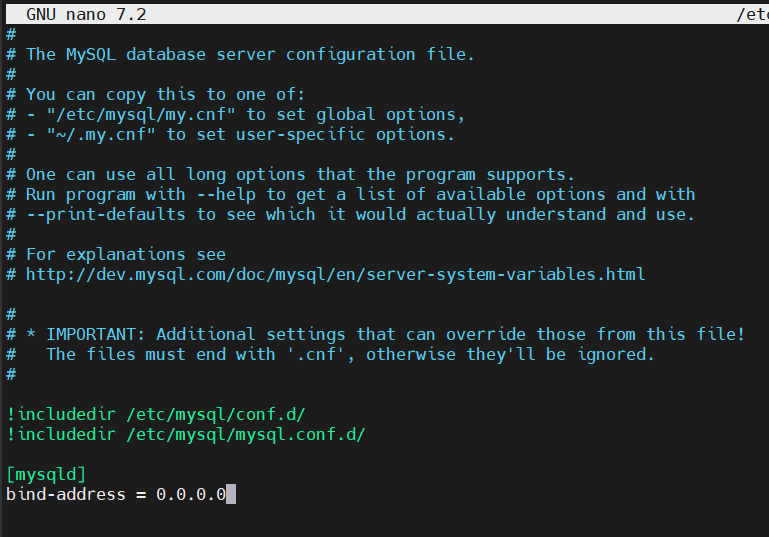
**USE migration\_project;**







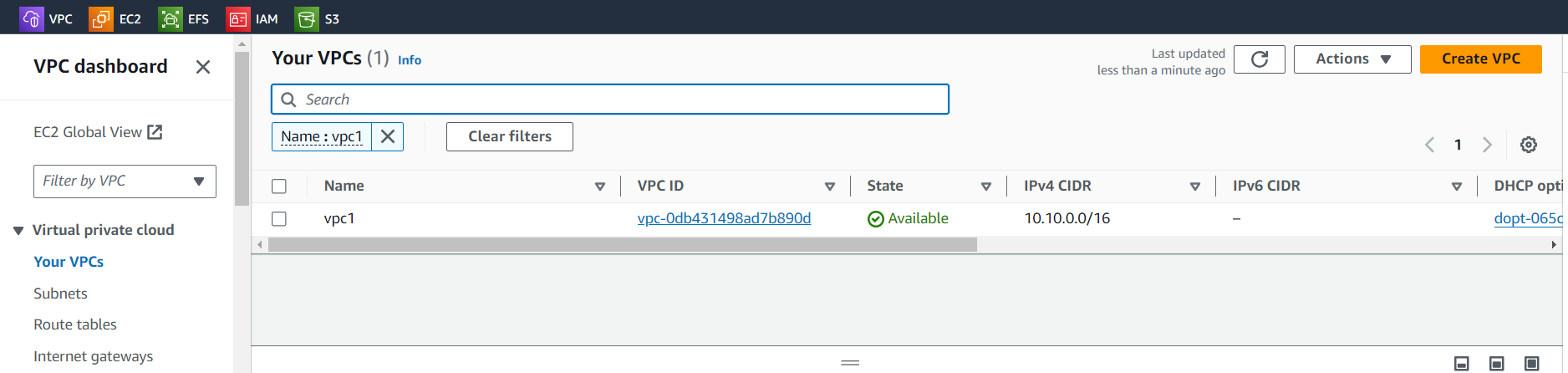
**Edit the MySQL Configuration File:** Use a text editor **sudo nano /etc/mysql/my.cnf** to open the file and add **Bind Address. It** allows remote connections, set the bind address to 0.0.0.0

****

Restart the mysql

**Destination Account Setup**

**VPC Creation**: After login in to the AWS management console click create VPC. Then provide a name and set IPv4 CIDR block: 10.10.0.0/16, set tendency as default and then click create.



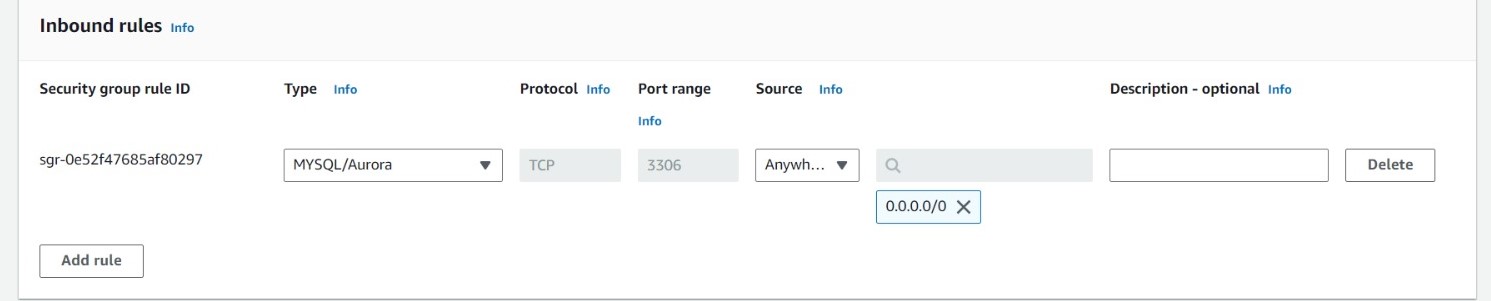
**Create Subnets:** Create a subnet by setting a name and choose the created VPC. Then set IPv4 CIDR block: 10.10.1.0/24 and availability zone

While creating subnets enable the auto-assign public IPv4 address

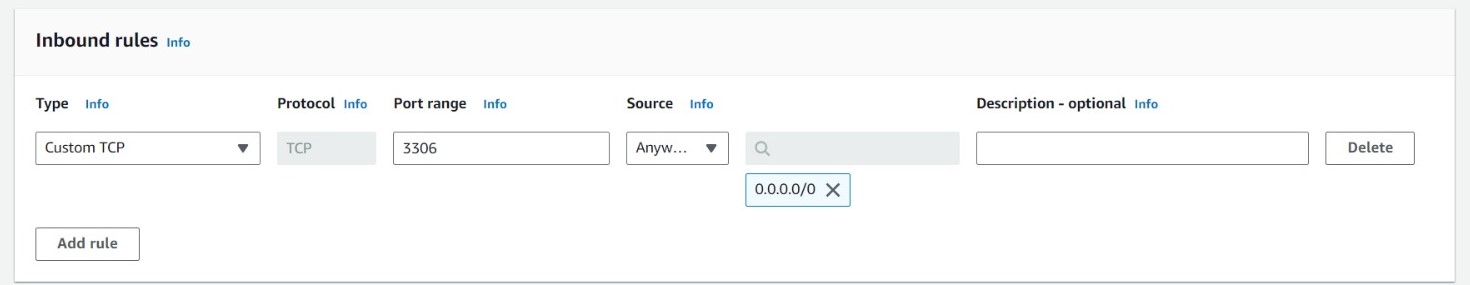
**Create an Internet Gateway: create an internet gateway then Attach it to VPC and enter the route table entries and associate it with the created subnet.**

**Create Security Groups: create a two security groups one for database and one for replication instance**

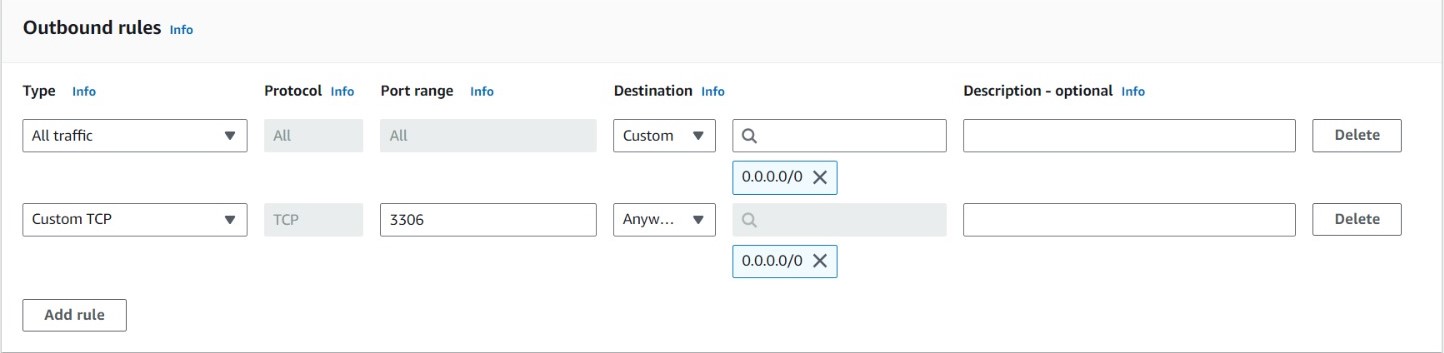
**Database:** In database choose the created VPC and add one inbound rules MYSQL/Aurora then set source as anywhere from IPv4.



**Replication Instance:** For replication instance also choose the created VPC and add only one inbound rule custom TCP with port range 3306 then set the source as anywhere from IPv4.

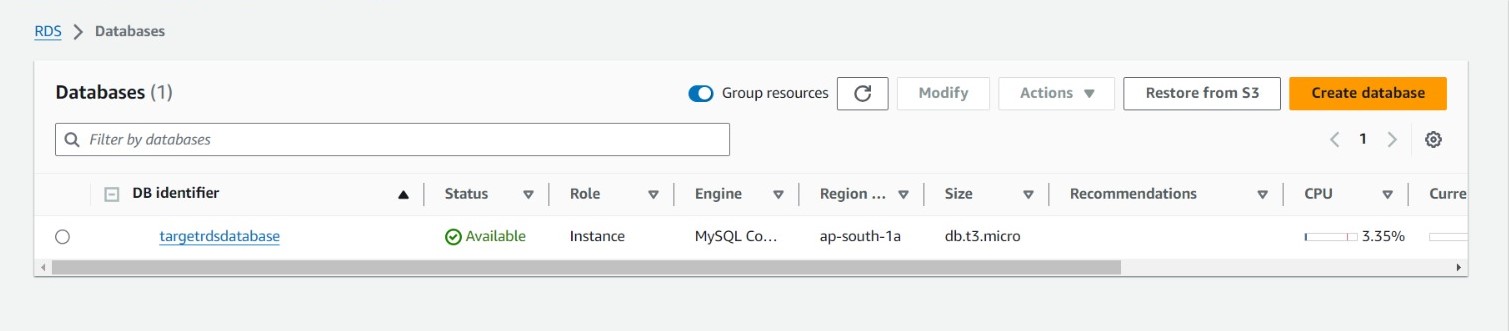


Add one outbound rule custom TCP with port range 3306 then set the source as anywhere from IPv4.

****

**Create an RDS Instance**

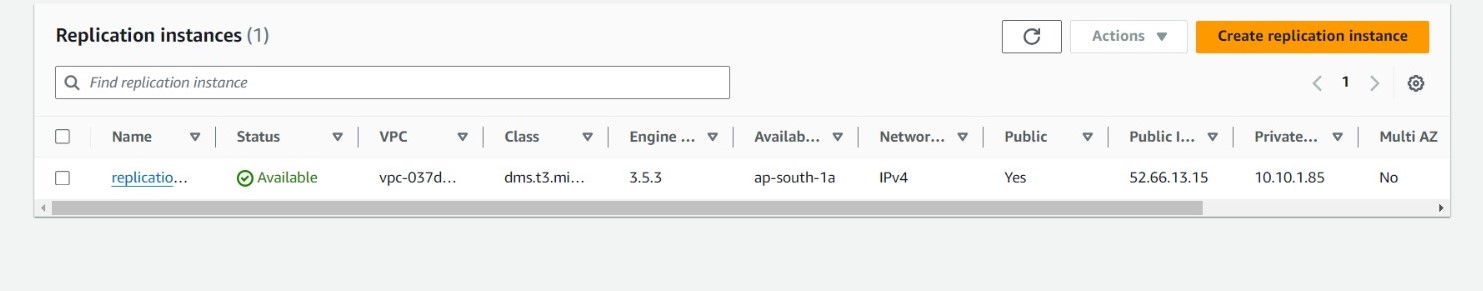
Navigate to RDS Dashboard Click on "Create database." Choose "Standard Create" for detailed configuration options. Select the "MySQL" engine. Choose a unique name for your RDS instance. Set up credentials for accessing the database. Choose an appropriate instance type. Select storage type and size as per your requirements. Choose the created **VPC**. Select the security group created for data base. Ensure the option "Publicly Accessible" is set to "Yes" if external access is needed. Select the security group created for data base.

****

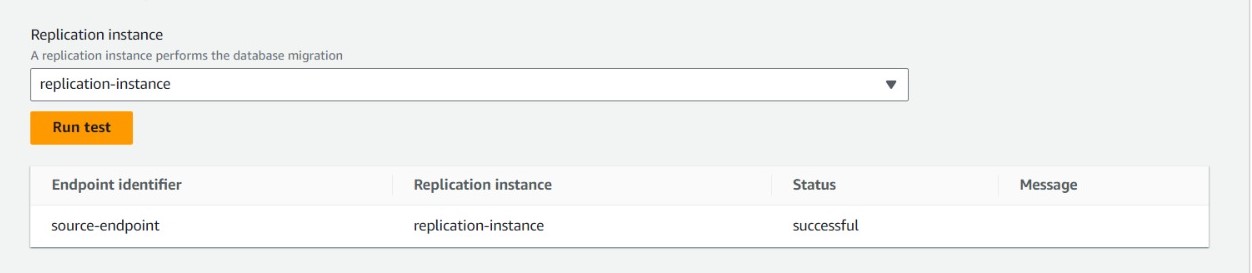
**Set Up AWS DMS**

In the AWS Management Console, search for and select "Database Migration Service."

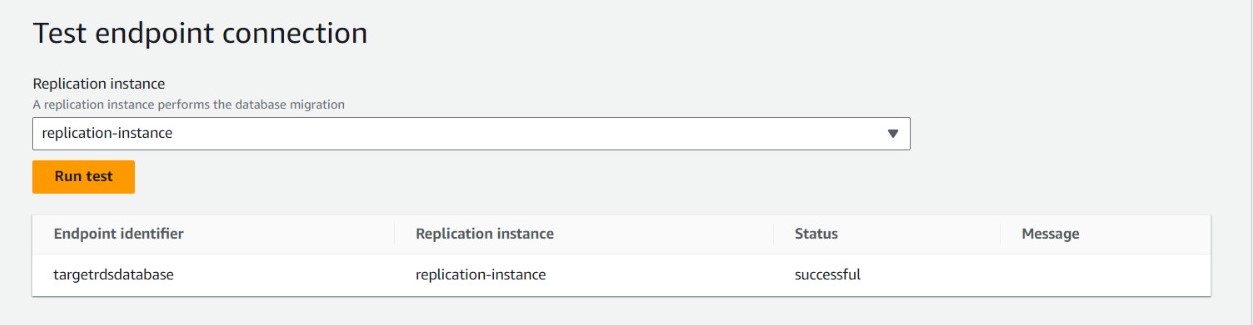
**Create a Replication Instance:** provide a unique name for your replication instance. Choose an instance size and Set storage size as needed. Select the security group created for replication instance.



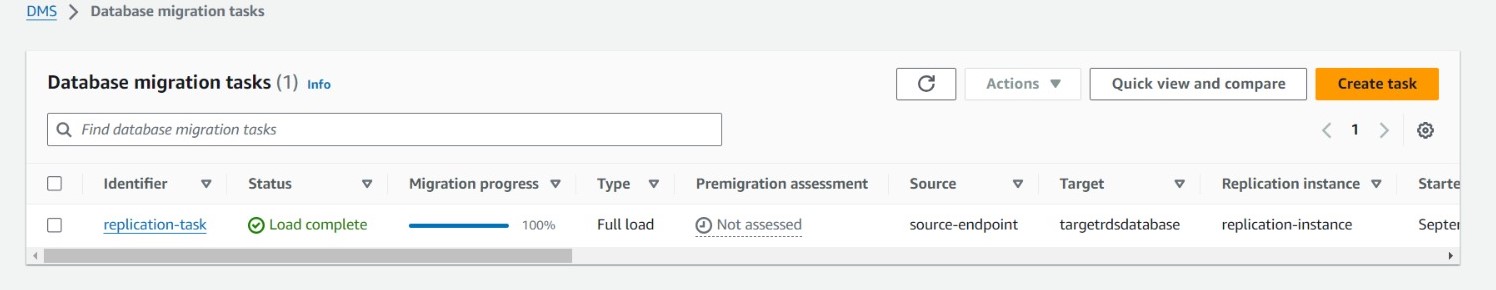
**Create Source Endpoint:** In the DMS dashboard, select "Endpoints" and click "Create endpoint." Choose "Source" as the endpoint type. provide a unique name for the source endpoint. Select engine as MYSQL. Provide the public IP or DNS of your on-premises MySQL server. Enter the appropriate credentials for the MySQL instance. Test the connection to ensure it's successful, then save the endpoint.



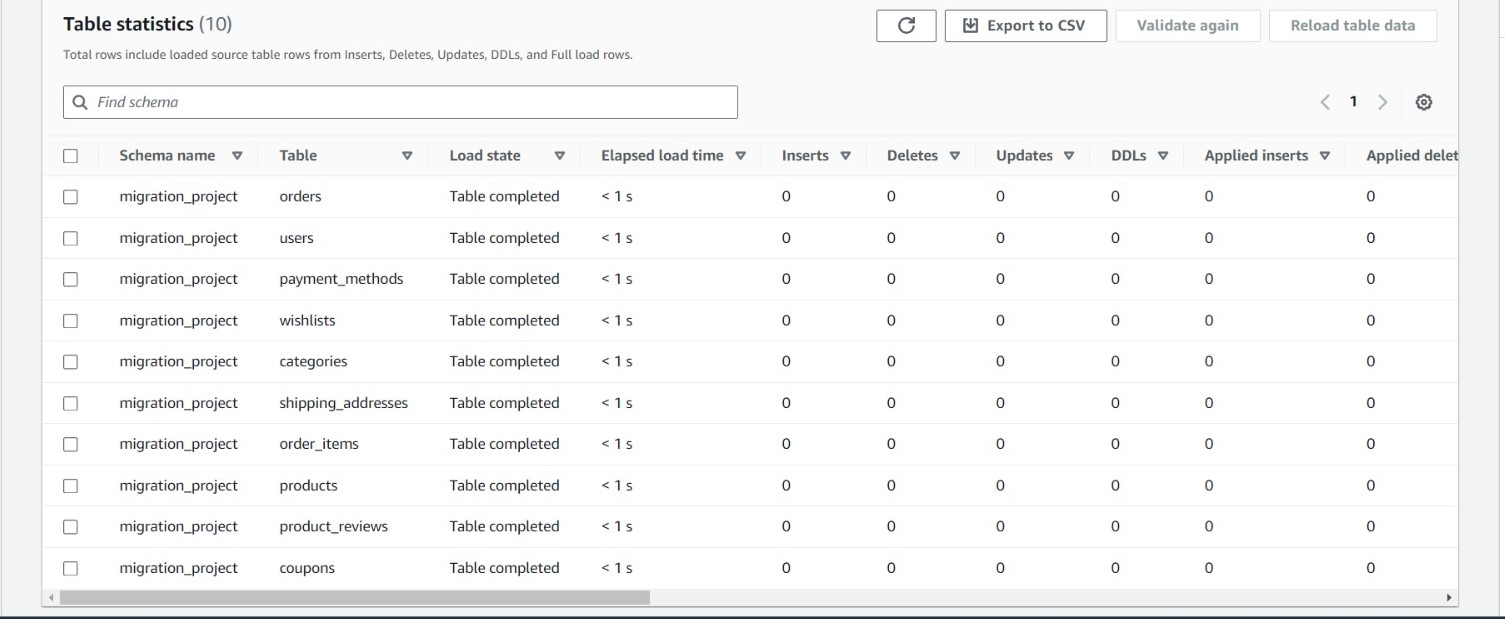
**Create Target Endpoint:** In the DMS dashboard, select "Endpoints" and click "Create endpoint." Choose "Target" as the endpoint type. Select "RDS" as the database engine and provide the connection details for your newly created RDS instance. Test the connection and save the endpoint.

****

**Create a Migration Task:** In the DMS dashboard, go to "Database migration tasks" and click "Create task." Enter a unique name for the migration task. Select the endpoints created earlier. Review the settings and click "Create task." Once created, start the migration task and monitor its progress.

****

Once the migration task is completed verify that the data has been successfully transferred to the RDS instance.



**Troubleshooting**

While testing the source endpoint in AWS Database Migration Service (DMS) for the MySQL database, the connection failed due to insufficient user privileges.

**Initial User Configuration**

Initially, a user was created with the following commands:

CREATE USER 'virtualadmin'@'localhost' IDENTIFIED BY 'Admin@123456789!';

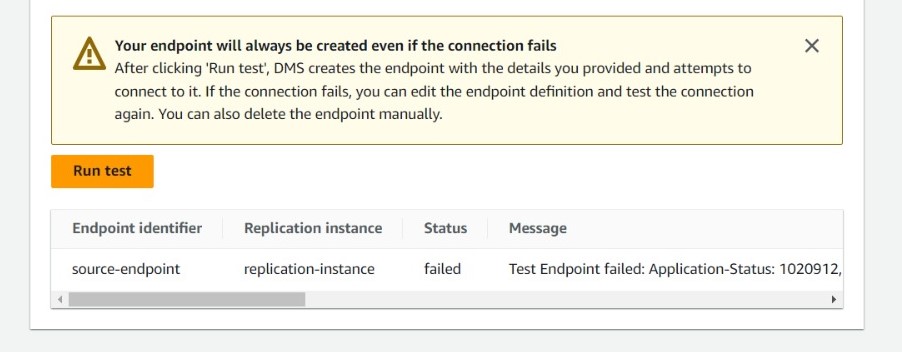
GRANT ALL PRIVILEGES ON \*.\* TO 'virtualadmin'@'localhost' WITH GRANT OPTION;

#### Issue Encountered

The initial user configuration was unsuccessful for the following reasons:

**Limited Host Access:** The user virtualadmin was restricted to connections only from localhost. Since the DMS replication instance attempts to connect from an external source, this configuration prevented a successful connection.

**Testing Failure:** When the source endpoint was tested, it failed because the DMS service was unable to authenticate using the virtualadmin user due to its restrictive host settings.



The new user was created with @'%', allowing connections from any host. All privileges were granted, ensuring that the user could access the database without restriction. After executing the new user commands, the source endpoint in DMS was tested again. The test was successful, confirming that the user virtualadmin now had the necessary permissions to connect to the MySQL database from any host.

**Conclusion**

In this project, we successfully demonstrated the migration of data from an on-premises MySQL database to an Amazon RDS MySQL instance using AWS Database Migration Service (DMS). This exercise not only showcased the technical capabilities of AWS but also underscored the significance of cloud solutions in modern data management.

The project fulfilled its primary objective of cloud migration while providing invaluable insights into best practices for managing cloud databases. By leveraging AWS services, we have positioned ourselves to adopt modern data management techniques, ultimately leading to enhanced efficiency and scalability in future operations.