# MYSQL database backup and migration from EC2 instance to amazon RDS

## 1.Introduction

**This project demonstrates a complete end-to-end workflow for managing MySQL database backups and migrating data from an Amazon EC2 instance to Amazon RDS. The goal is to build a reliable, scalable, and secure database environment using AWS cloud infrastructure.**

### 1.1 The implementation includes:

- **Deploying MySQL on an EC2 instance**
- **Setting up an Amazon RDS MariaDB instance**
- **Creating databases and sample data**
- **Taking backups using mysqldump**
- **Restoring the backup into the RDS instance**
- **This process ensures high availability, automated backups, and easier database maintenance.**

**This process ensures high availability, automated backups, and easier database maintenance.**

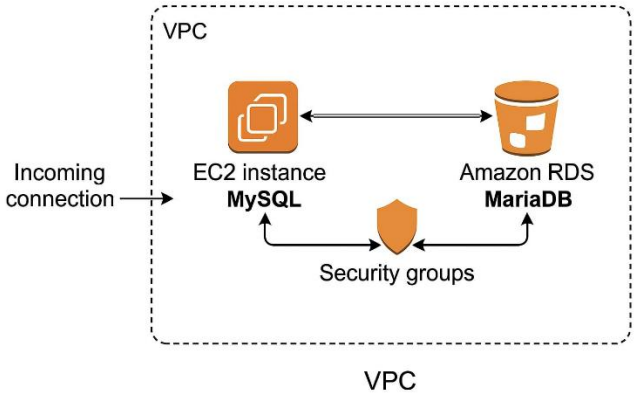
## 2.Objective

- **To set up and configure a MySQL database on an Amazon EC2 instance.**
- **To create a secure and scalable Amazon RDS MySQL instance.**
- **To migrate database data from EC2 to RDS using mysqldump.**
- **To ensure reliable database backup and restoration for data safety.**
- **To improve database availability, performance, and ease of management using AWS services.**

## 3. Benefits

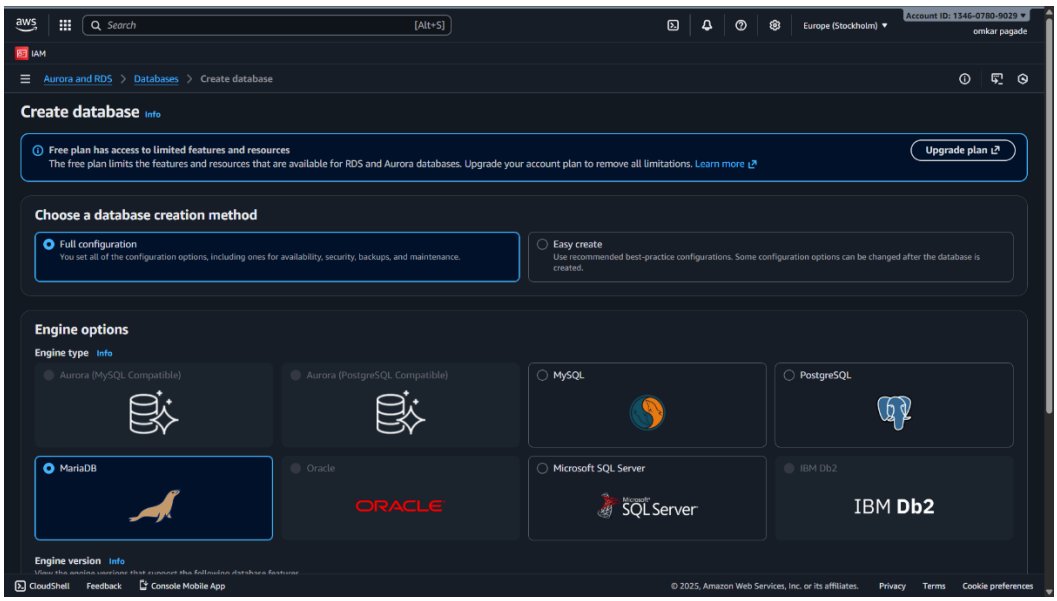
- **Automated backups**
- **Easy scaling**
- **High availability**
- **Reduced operational overhead**

## 4.Architecture Diagram

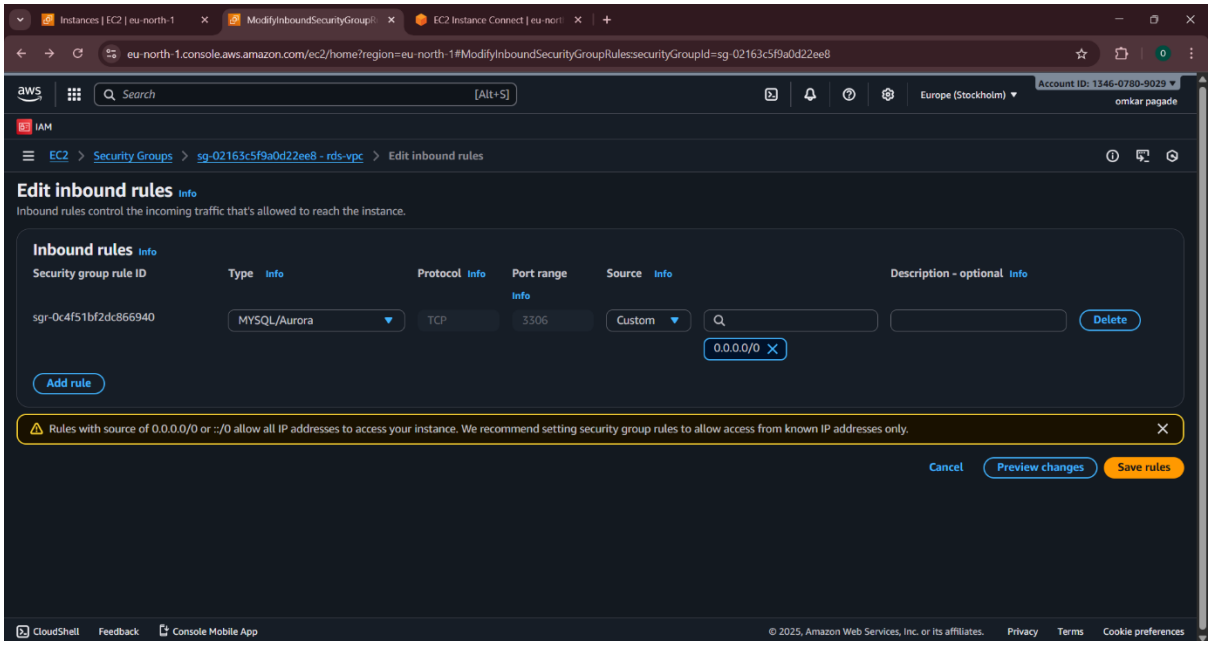



## 5.Scrrenshots

## 5.1 Create RDS MariaDB Instance

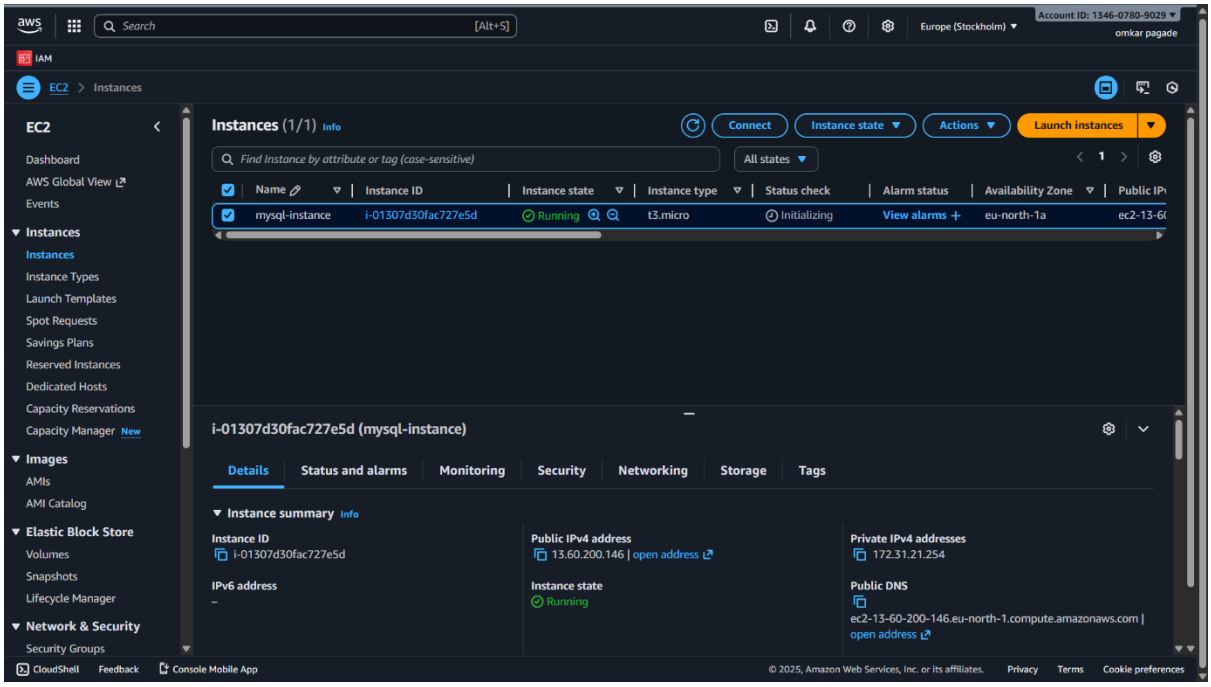


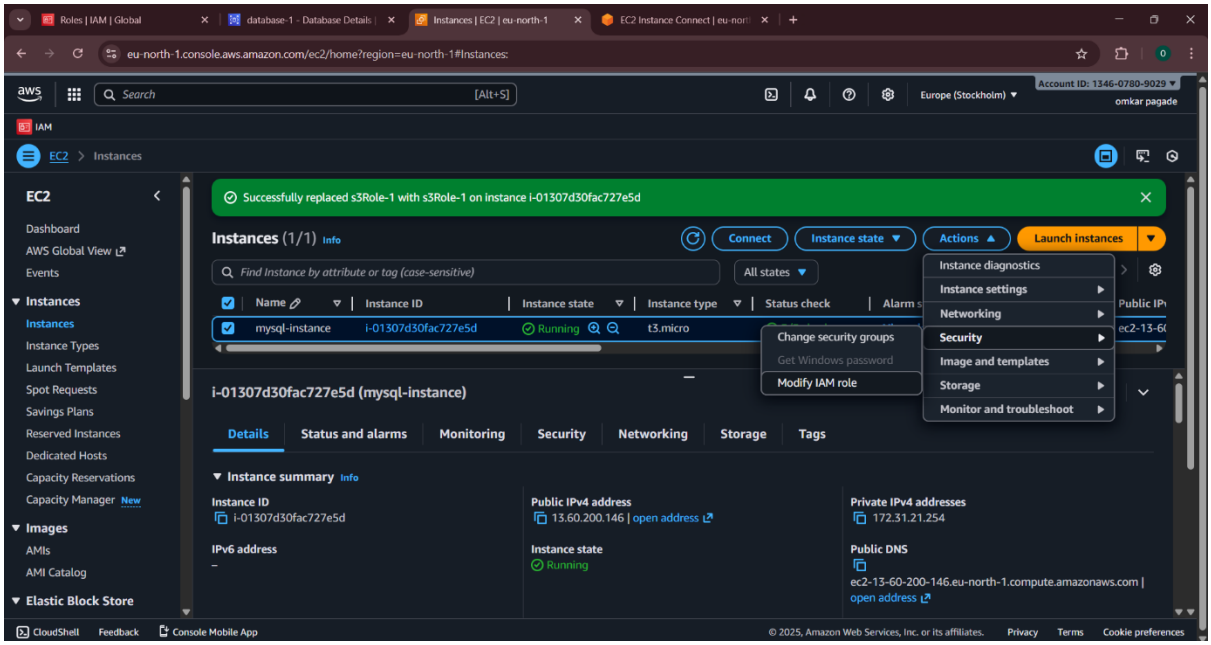

- **Create RDS instance select full config & MariaDB**
- **Burstable classes (T series) are low-cost database servers that run at normal/low performance most of the time, but can become fast for a short time when needed.**
- **Go to security group of RDS instances and edit inbound rule and allow IPv4 anywhere and save**

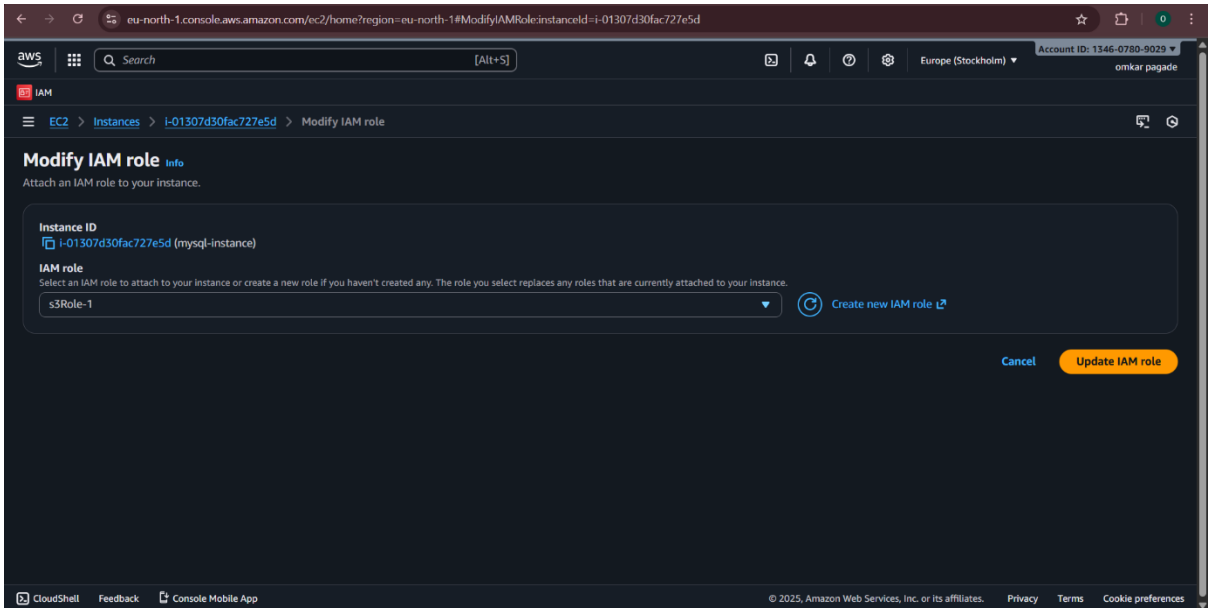



**We edit inbound rule because Amazon RDS is protected by a Security Group, and if the inbound rule does NOT allow your connecting IP, the connection will FAIL.**

## 5.2 Create EC2 Instance






- **Create EC2 instance and Allow MYSQL in security group**
- **Modify the IAM Role give a S3 full access**

## 5.2 Connect To EC2 Instance and Run the commands

```
[ec2-user@ip-172-31-21-254 ~]$ sudo -i
[root@ip-172-31-21-254 ~]# yum install mariadb105* -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
```

- **Install mariadb105* (105*) is for download all dependency of MariaDB**
- **Copy the endpoint of RDS instance for giving the access the RDS instance**
- **Run command :- mysql -h <EndPoint> -u username -p(password)**
- **Do not give space in -p and password**

```
[root@ip-172-31-21-254 ~]# mysql -h database-1.cpaw0eia8hjs.eu-north-1.rds.amazonaws.com -u admin -pomkar123
```

- **create database using (create database nameofdatabase)**
- **and type EXIT to exit to RDS instance**
- **take backup of database in .sql file**
- **start the MariaDB service using systemctl after that**
- **wirte mysql_secure_installation**
- **creates database on your localhost and insert values iin that**

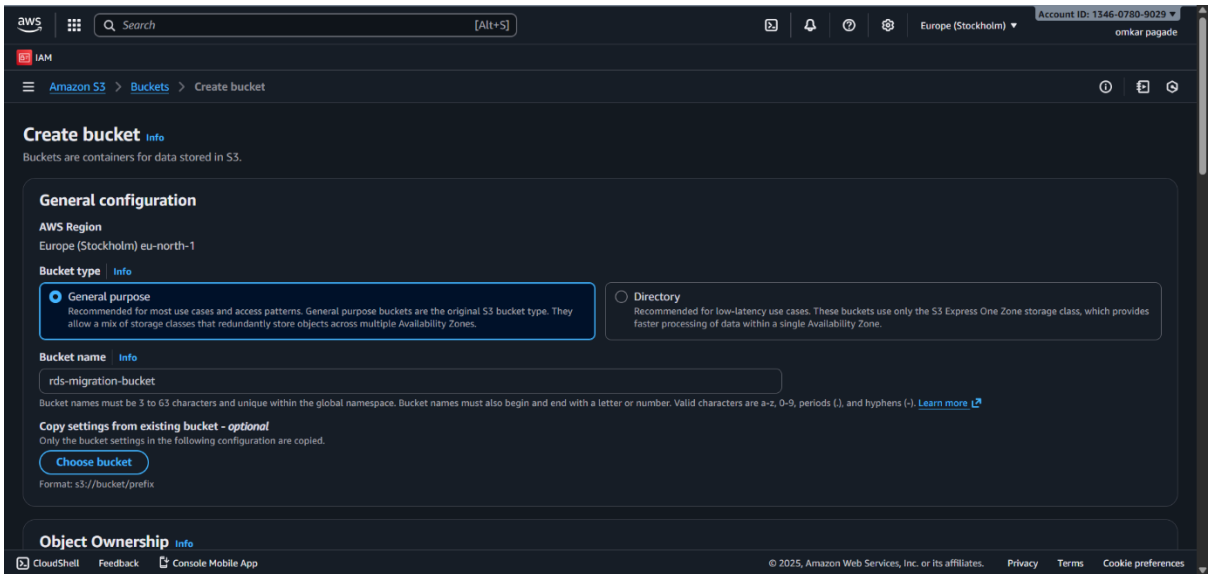
```
[root@ip-172-31-21-254 ~]# mysql -h localhost -u root -p123
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 15
Server version: 10.5.29-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| project_pune       |
+--------------------+
4 rows in set (0.000 sec)
```

```
MariaDB [(none)]> use project_pune
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [project_pune]> show table;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' at line
1
MariaDB [project_pune]> show tables;
+------------------------+
| Tables_in_project_pune |
+------------------------+
| emp                    |
+------------------------+
1 row in set (0.000 sec)

MariaDB [project_pune]> select * from emp;
+--------------+---------+-------+
| student_name | roll_no | score |
+--------------+---------+-------+
| omkar        |       1 |     1 |
+--------------+---------+-------+
1 row in set (0.000 sec)

MariaDB [project_pune]> exit
Bye
[root@ip-172-31-21-254 ~]# mysqldump -u root -p123 project_pune > emp.sql
[root@ip-172-31-21-254 ~]# ls
emp.sql
[root@ip-172-31-21-254 ~]# mysql -h database-1.cpaw0eia8hjs.eu-north-1.rds.amazonaws.com -u admin -pomkar123 pune < emp.sql
[root@ip-172-31-21-254 ~]#
```

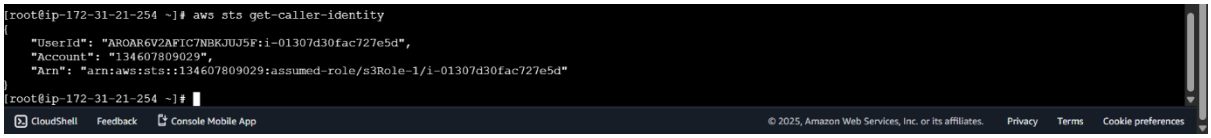
```
Bye
[root@ip-172-31-21-254 ~]# mysqldump -u root -p123 project_pune > emp.sql
[root@ip-172-31-21-254 ~]# ls
emp.sql
[root@ip-172-31-21-254 ~]# mysql -h database-1.cpaw0eia8hjs.eu-north-1.rds.amazonaws.com -u admin -pomkar123 pune < emp.sql
[root@ip-172-31-21-254 ~]#
```

- **Take backup of your .sql file (mysqldump -u root -p123 > emp.sql)**
- **And transfer that into RDS instance in pune database**
- **(mysql -h <endpoint> -u admin -p(password) < your backup .sql filename)**

## 5.3 Create S3 bucket for backup



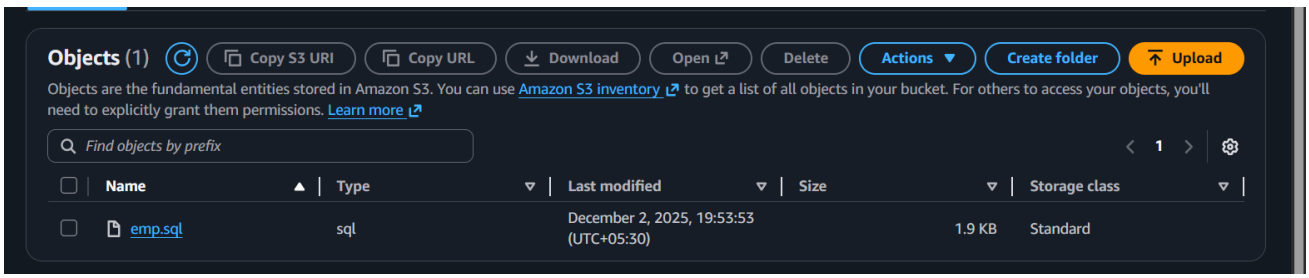

- **Create S3 Buckets**




- **Check IAM role(S3 full accessed) is there to take backup in instance**

```
[root@ip-172-31-21-254 ~]# aws s3 cp emp.sql s3://rds-migration-backup-bucket-2027/
upload: ./emp.sql to s3://rds-migration-backup-bucket-2027/emp.sql
[root@ip-172-31-21-254 ~]#
```

- **To copy of your .sql file in EC2 instance to S3 Bucket**
- **(aws s3 cp <you .sql file name> s3://<name of you bucket>/)**



## 6.Conclusion

**This project successfully demonstrates the complete process of setting up, securing, and migrating a MySQL database environment using Amazon Web Services. A MySQL database was hosted on an EC2 instance, backed up using the mysqldump utility, and restored into a fully managed Amazon RDS MariaDB instance. The implementation highlights essential cloud concepts such as instance provisioning, security group configuration, data migration, and backup management.**

**Through this project, we achieved improved database availability, reliability, and performance by shifting from a manually managed EC2-based database to a scalable and automated RDS environment. The workflow also ensures data safety through regular backups and provides a practical understanding of deploying production-ready database systems on AWS.**

**Overall, the project enhances cloud proficiency and demonstrates a real-world DevOps practice of securely managing and migrating databases in the AWS ecosystem.**