**Task on RDS**

1. Create MariaDB DB on EC2.

2. Insert some dummy data.

**Step 1: Launch an EC2 Instance**

1. Go to the **AWS Management Console → EC2 → Launch Instance**.
2. Choose **Amazon Linux 2023** (or Ubuntu if you prefer).
3. Select **t2.micro** (free tier eligible).
4. Create or select a **key pair**.
5. In **Security Group**, allow inbound rules:
   * 22 (SSH)
   * 3306 (MySQL/MariaDB)
6. Launch the instance.

**🔐 Step 2: Connect to EC2 via SSH**

ssh -i your-key.pem ec2-user@<EC2-Public-IP>

**⚙️ Step 3: Install MariaDB Server**

For **Amazon Linux 2023**:

sudo dnf install mariadb105-server -y

Start and enable the service:

sudo systemctl start mariadb

sudo systemctl enable mariadb

**🔧 Step 4: Secure MariaDB Installation**

Run:

sudo mysql\_secure\_installation

Recommended answers:

Switch to unix\_socket authentication [Y/n] n

Change the root password? [Y/n] y

Remove anonymous users? [Y/n] y

Disallow root login remotely? [Y/n] y

Remove test database and access to it? [Y/n] y

Reload privilege tables now? [Y/n] y

**🗄️ Step 5: Login to MariaDB**

sudo mysql -u root -p

Enter the **root password** you just set.

**📦 Step 6: Create Database and User**

Use your example variables:

CREATE DATABASE ec2db;

CREATE USER 'ec2dbuser'@'%' IDENTIFIED BY 'admin123456';

GRANT ALL PRIVILEGES ON ec2db.\* TO 'ec2dbuser'@'%';

FLUSH PRIVILEGES;

**🧱 Step 7: Insert Dummy Table and Data**

USE ec2db;

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

role VARCHAR(50),

salary INT

);

INSERT INTO employees (name, role, salary) VALUES

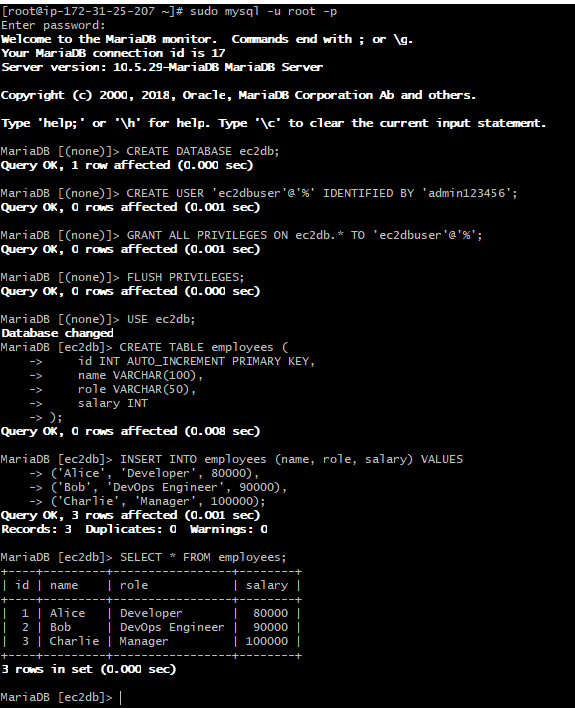
('Alice', 'Developer', 80000),

('Bob', 'DevOps Engineer', 90000),

('Charlie', 'Manager', 100000);

**🔍 Step 8: Verify Data**

SELECT \* FROM employees;



3.Take the backup of dummy data on EC2.

### **Command to Backup**

mysqldump -u root -p ec2db > /root/ec2db\_backup.sql

* -u root → MariaDB root user
* -p → prompts for password
* ec2db → database name
* > /root/ec2db\_backup.sql → stores backup file in /root

After running this, it will ask for your **MariaDB root password**.

### **Verify Backup**

ls -lh /root/ec2db\_backup.sql

You should see something like:

-rw-r--r-- 1 root root 1.2K Oct 8 19:50 /root/ec2db\_backup.sql

### **Optional: Compress Backup**

gzip /root/ec2db\_backup.sql

This will produce:

/root/ec2db\_backup.sql.gz

to save space.

4.Launch MariaDB RDS instance.

## ****Step 1 — Go to RDS in AWS Console****

1. Sign in to the AWS Management Console.
2. Navigate to: **Services → RDS**.

## ****Step 2 — Create Database****

Click **Create database**.

### **Database creation method**

* Choose **Standard create** (more control).

### **Engine options**

* Engine: **MariaDB**
* Version: choose latest stable (e.g., 10.11.x).

### **Templates**

* Choose: **Free tier** (if eligible) for testing.

## ****Step 3 — Configure Database****

* **DB instance identifier**: mariadb-instance
* **Master username**: admin
* **Master password**: admin123456 (write it down)
* **Confirm password**: admin123456

## ****Step 4 — Instance size****

* Select db.t3.micro (free tier).

## ****Step 5 — Storage****

* Storage type: General Purpose SSD (gp3)
* Allocated storage: 20 GB (default)

## ****Step 6 — Connectivity****

* Virtual Private Cloud (VPC): choose your existing VPC or default.
* Subnet group: default.
* Public access: **Yes** (to connect from your local machine or EC2).
* VPC security group: **Create new** or choose existing (allow inbound port 3306).

## ****Step 7 — Additional Configuration****

* Database name: ec2db (or your choice)
* Enable automatic backups (optional but recommended)
* Backup retention period: 7 days (default)

## ****Step 8 — Create Database****

Click **Create database**.  
RDS will take several minutes to launch the instance.

## ****Step 9 — Connect to RDS****

Once the instance is available:

1. Go to **RDS → Databases**.
2. Select your MariaDB instance.
3. Copy the **Endpoint** and **Port** (e.g., mariadb-instance.xxxxxx.us-east-1.rds.amazonaws.com:3306).

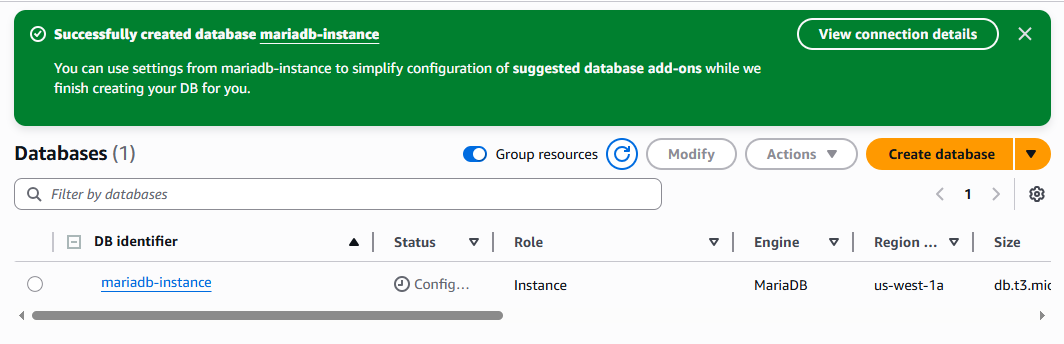
### **Connect via EC2 or Local Machine**

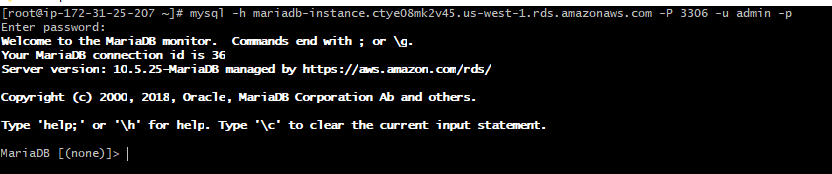
mysql -h <RDS-endpoint> -P 3306 -u admin -p

Example:

mysql -h mariadb-instance.xxxxxx.us-east-1.rds.amazonaws.com -P 3306 -u admin -p

Enter password: admin123456.





5. Migrate database from EC2 to RDS.

### **Step 1 — Test Connection to RDS**

On your EC2 instance:

mysql -h mariadb-instance.xxxxxx.us-east-1.rds.amazonaws.com -P 3306 -u admin -p

Enter your password (admin123456 or what you set).

If it connects successfully, proceed.

### **Step 2 — Restore Backup**

Still on your EC2 instance:

mysql -h mariadb-instance.xxxxxx.us-east-1.rds.amazonaws.com -P 3306 -u admin -p Rds\_maria < /root/ec2db\_backup.sql

* Rds\_maria → your target DB name on RDS.
* /root/ec2db\_backup.sql → your backup file location.

It will ask for your RDS password and then start restoring.

⏳ This may take a few seconds or minutes depending on backup size.

### **Step 3 — Verify Data**

Connect to RDS:

mysql -h mariadb-instance.xxxxxx.us-east-1.rds.amazonaws.com -P 3306 -u admin -p

Inside MariaDB:

USE Rds\_maria;

SHOW TABLES;

SELECT \* FROM employees;

You should see your dummy data:

+----+---------+-----------------+--------+

| id | name | role | salary |

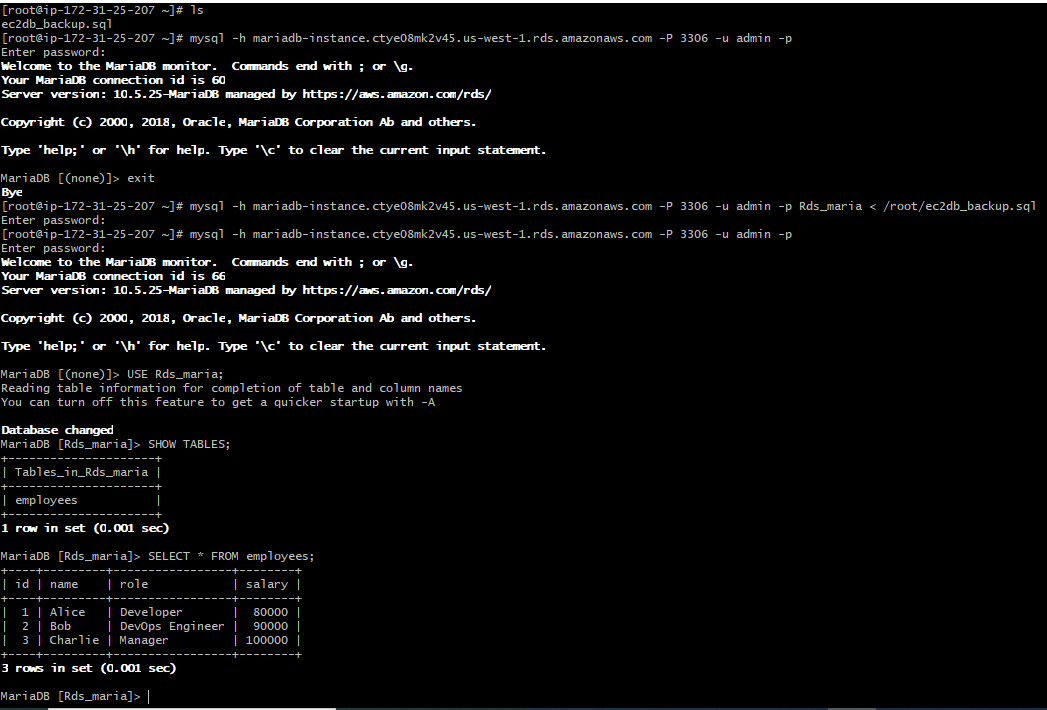
+----+---------+-----------------+--------+

| 1 | Alice | Developer | 80000 |

| 2 | Bob | DevOps Engineer | 90000 |

| 3 | Charlie | Manager | 100000 |

+----+---------+-----------------+--------+



6.Install MySQL DB on EC2.

7.Launch MySQL RDS image.

Perfect 👍 — let’s go step by step to **launch a MySQL RDS instance** in AWS.

You’ll be creating a **managed MySQL database** (RDS), separate from your EC2 MariaDB setup.

**🧩 Step 1: Go to RDS Console**

1. Open **AWS Management Console → RDS**  
   👉 https://console.aws.amazon.com/rds
2. Click **“Create database”**.

**🧩 Step 2: Choose Database Creation Method**

Select:

* **Standard create**

**🧩 Step 3: Choose Engine**

* **Engine type:** ✅ **MySQL**
* **Version:** Choose the latest **MySQL 8.x**

**🧩 Step 4: Choose Template**

If you’re just testing:

* Choose **Free tier**  
  (Eligible for db.t3.micro instance type and 20 GB storage)

**🧩 Step 5: Settings**

| **Field** | **Value** |
| --- | --- |
| **DB instance identifier** | mysql-db-instance |
| **Master username** | admin |
| **Master password** | admin123456 |
| **Confirm password** | admin123456 |

✅ You can use your earlier password for consistency.

**🧩 Step 6: DB Instance Configuration**

| **Setting** | **Recommended** |
| --- | --- |
| **DB instance class** | db.t3.micro (Free tier) |
| **Storage type** | General Purpose (SSD) |
| **Allocated storage** | 20 GB |
| **Enable storage autoscaling** | Optional |

**🧩 Step 7: Connectivity**

1. **VPC:** Choose your existing VPC (or default VPC).
2. **Public access:**
   * ✅ Select **Yes** if you want to connect from outside (e.g., from your EC2 or Workbench).
3. **VPC security group:**
   * Choose an existing group that allows inbound **port 3306**
   * Or create a new one with:
     + **Type:** MySQL/Aurora
     + **Port:** 3306
     + **Source:** Your IP or EC2’s private IP CIDR
4. **Availability Zone:** No preference (default).

**🧩 Step 8: Database Authentication**

* Choose **Password authentication** (default).

**🧩 Step 9: Additional Configuration**

(Optional but useful)

* **Initial database name:** ec2db
* **Backup retention:** Keep default (7 days)
* **Encryption:** Optional
* **Monitoring & Logs:** Default (you can enable later)

**🧩 Step 10: Create Database**

Click **Create database**.

Wait 5–10 minutes ⏳  
→ RDS status will change from **Creating → Available**

**🧩 Step 11: Connect to RDS**

Once available:

1. Go to **RDS → Databases → mysql-db-instance**
2. Copy the **Endpoint** (something like mysql-db-instance.xxxxxx.us-east-1.rds.amazonaws.com)

From your EC2 or local system:

mysql -h <RDS-endpoint> -u admin -p

Enter:

admin123456

✅ You’re now connected to your MySQL RDS!

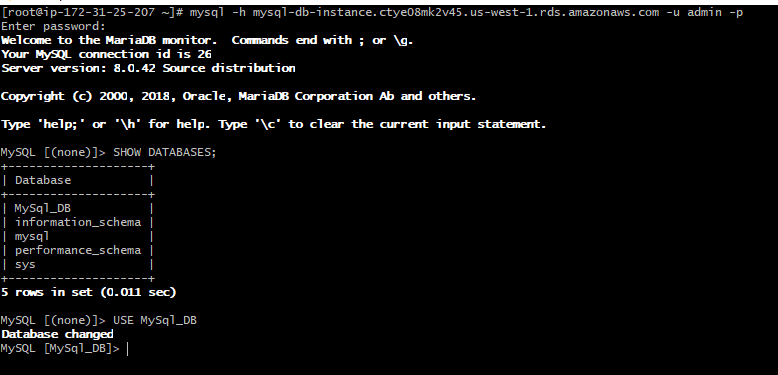
**🧠 Optional: Verify Database**

Inside MySQL:

SHOW DATABASES;

USE ec2db;

✅ **You now have a running MySQL RDS instance!**



9.Take backup of DB and restore the DB.

### **Step 1 — Go to the RDS Console**

1. Log in to AWS Console.
2. Navigate to **RDS** service:  
   AWS RDS Console

### **📌 Step 2 — Select Your Database**

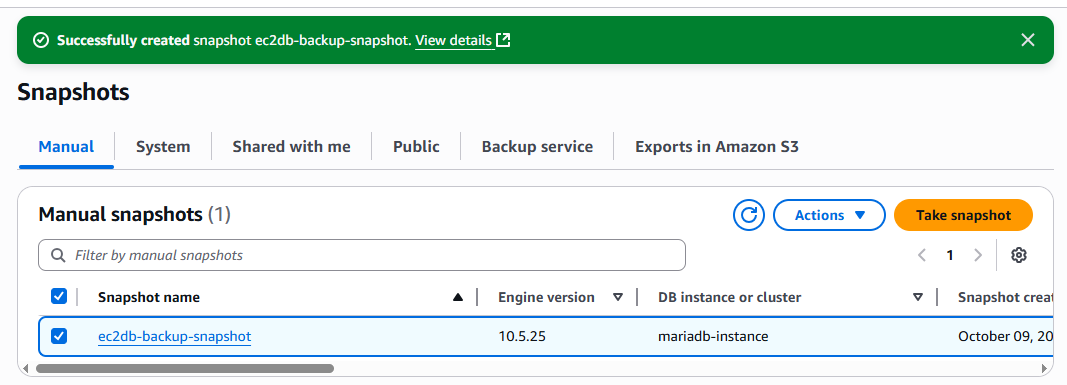
1. In the left menu, click **Databases**.
2. Click your **MariaDB instance** (e.g., mariadb-instance).

### **📌 Step 3 — Create a Snapshot (Backup)**

1. In the DB details page, click **Actions** → **Take snapshot**.
2. Give your snapshot a name (example: ec2db-backup-snapshot).
3. Click **Take snapshot**.

✅ This creates a **manual snapshot** of your RDS MariaDB database.

You can see progress in **Snapshots → DB Snapshots**.

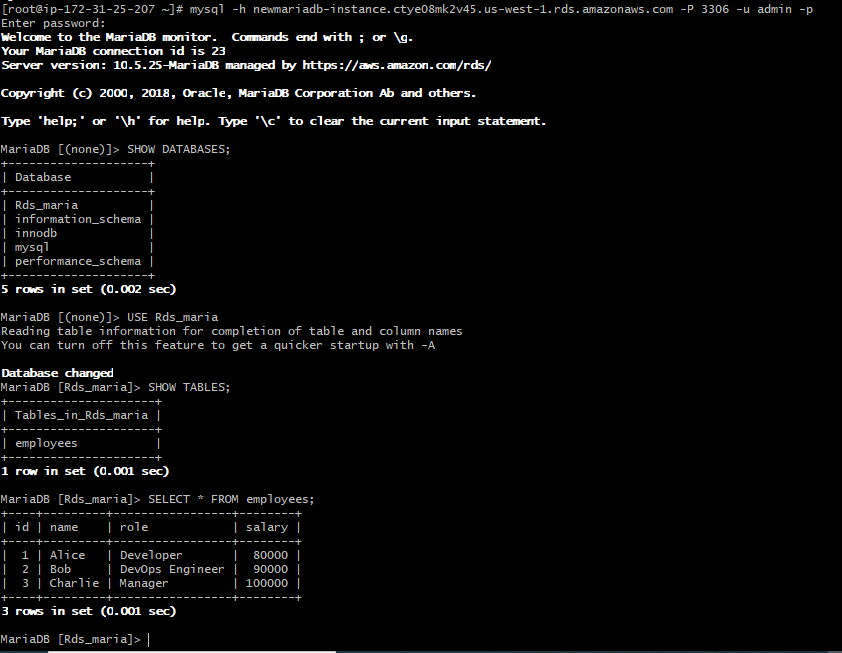


### **Step 4 — Restore From Snapshot**

When you want to restore:

1. Go to **Snapshots** in RDS Console.
2. Select your snapshot.
3. Click **Actions** → **Restore snapshot**.
4. Fill in:
   * **DB instance identifier** (new instance name).
   * DB instance settings (size, VPC, etc.).
5. Click **Restore DB Instance**.

✅ AWS will create a new MariaDB RDS instance with your snapshot data.



10.Create read replica.

### **Step 1 — Go to AWS RDS Console**

1. Log in to AWS Console.
2. Go to **RDS** service:  
   AWS RDS Console

### **📌 Step 2 — Select Your MariaDB Instance**

1. Click **Databases** from the left menu.
2. Select your MariaDB DB instance (e.g., mariadb-instance).

### **📌 Step 3 — Create Read Replica**

1. Click **Actions** → **Create read replica**.
2. Fill in the replica details:
   * **DB instance identifier** → Give a name (e.g., mariadb-replica).
   * **DB instance class** → Choose size depending on load (e.g., db.t3.medium).
   * **Multi-AZ deployment** → Optional, but increases availability.
   * **Public access** → Yes or No depending on your requirement.
   * **VPC, subnet group, security group** → Usually same as the source DB.
3. Click **Create read replica**.

### **📌 Step 4 — Wait for Creation**

* You’ll see the read replica appear in **Databases** list.
* Status will be **creating** until complete.

### **💡 Notes**

* A read replica is **read-only** — you can’t write data to it directly.
* It automatically replicates data from the primary DB.
* You can promote the replica to a standalone DB if needed later.