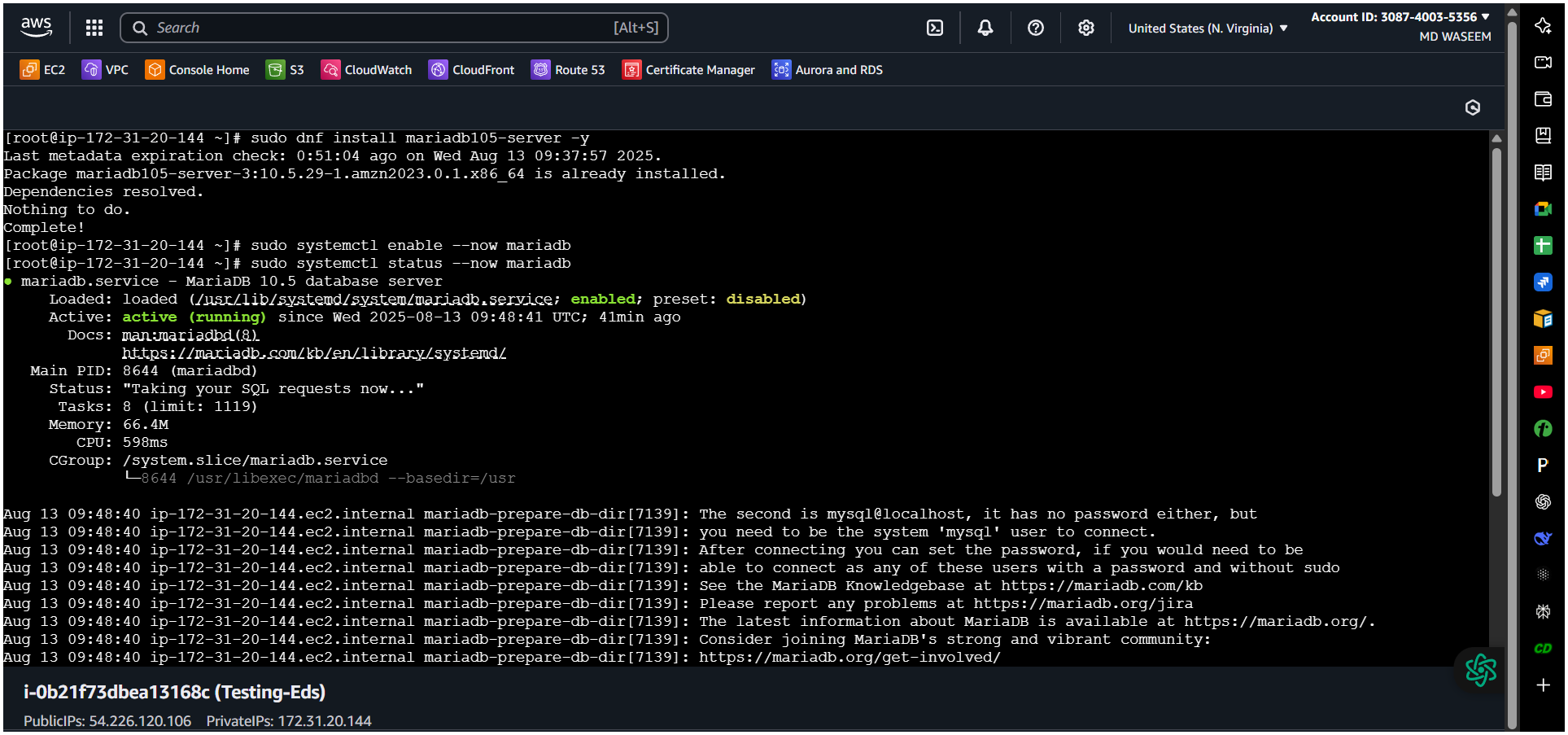
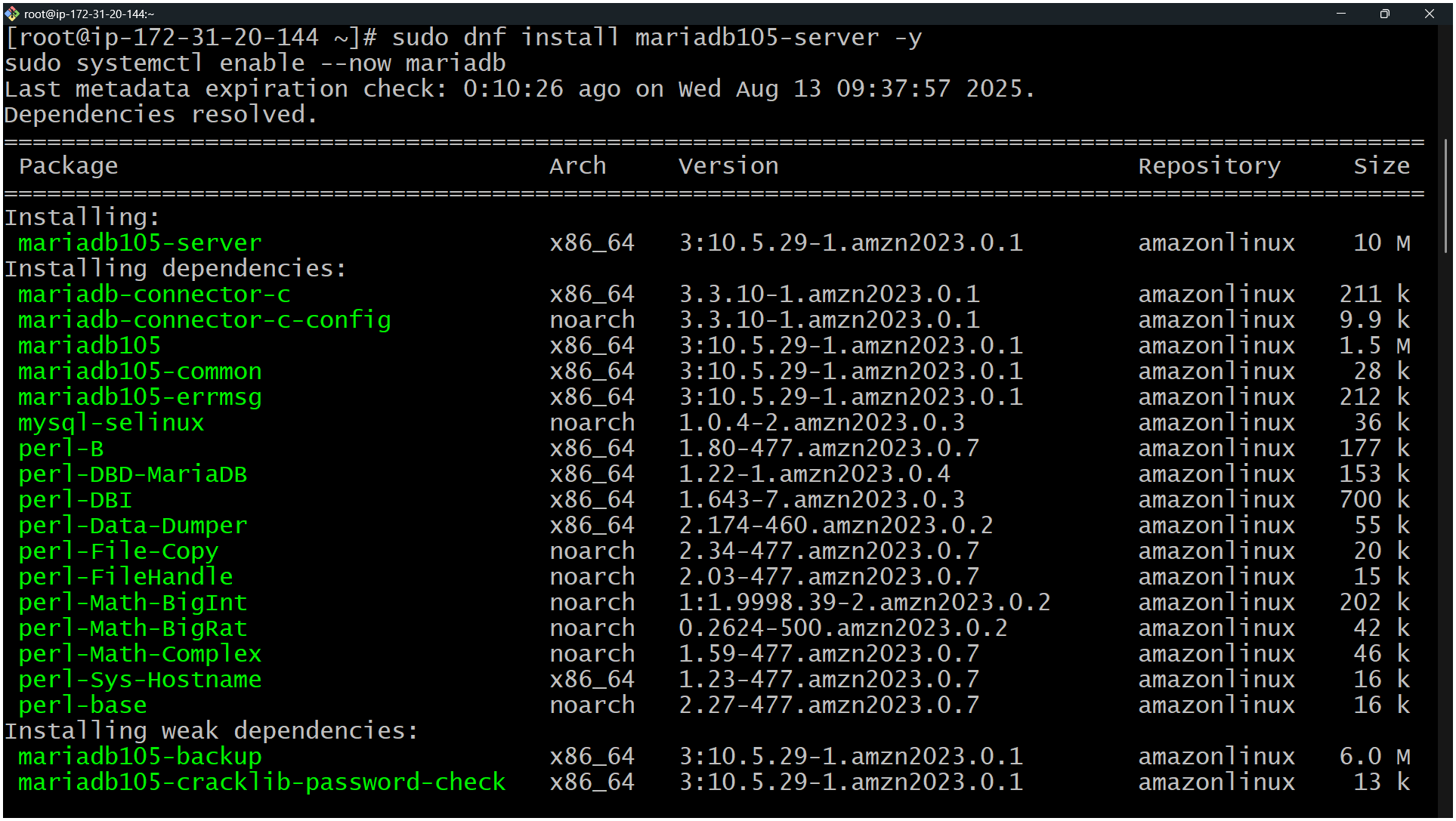
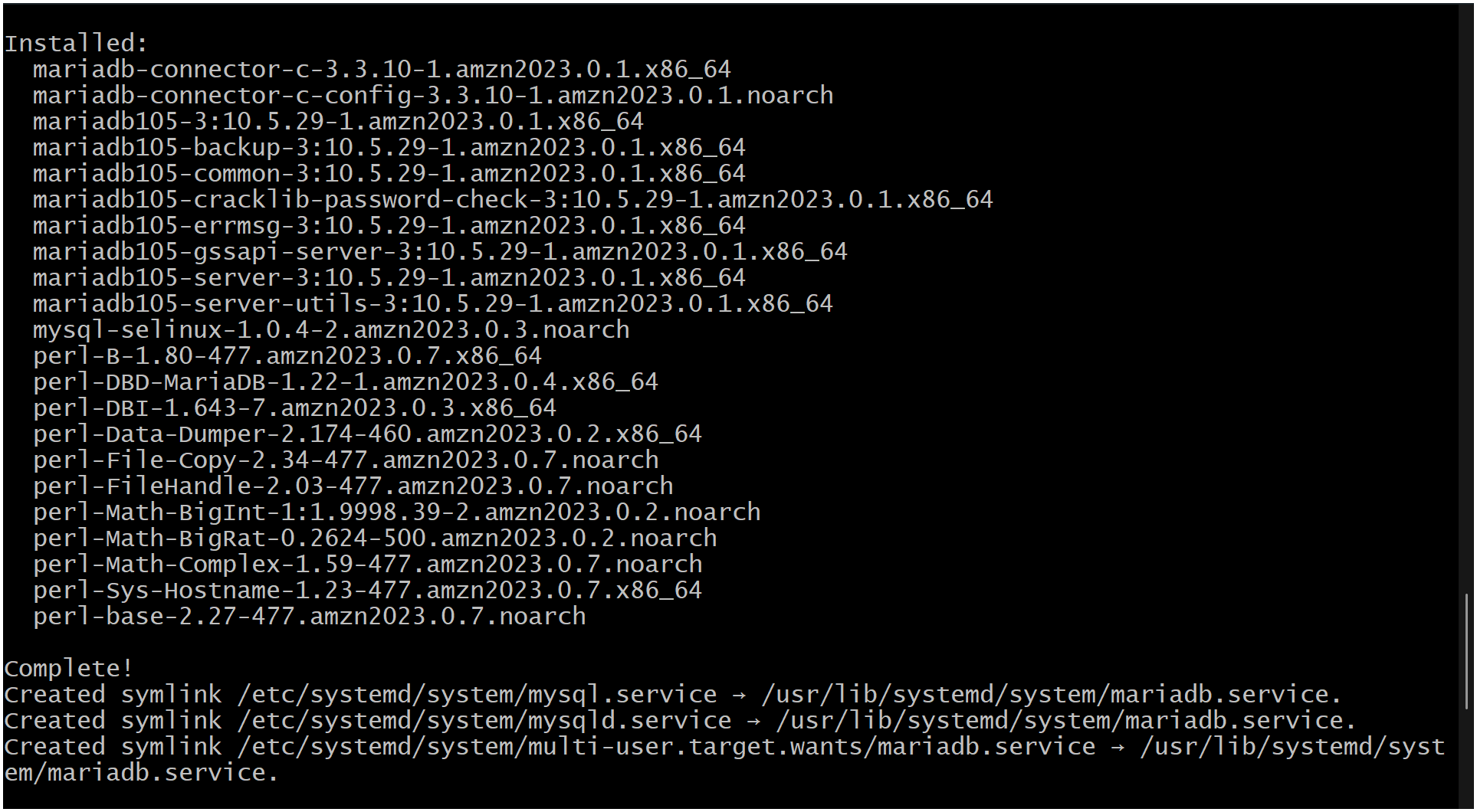
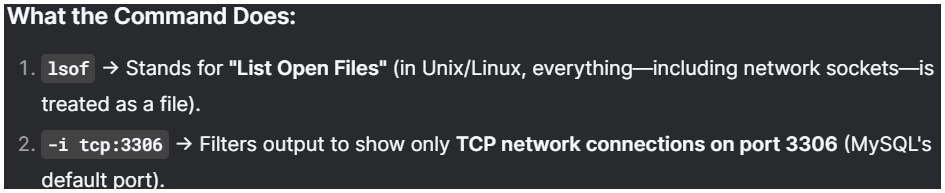
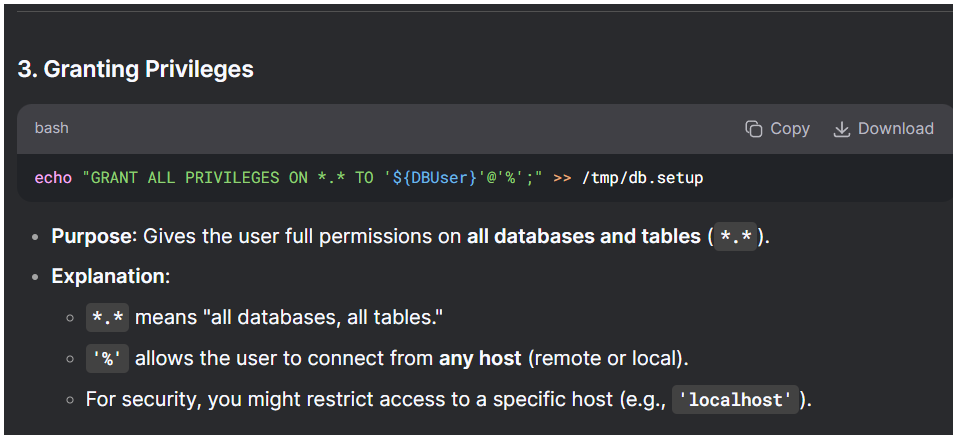
**Brief Explanation of RDS**

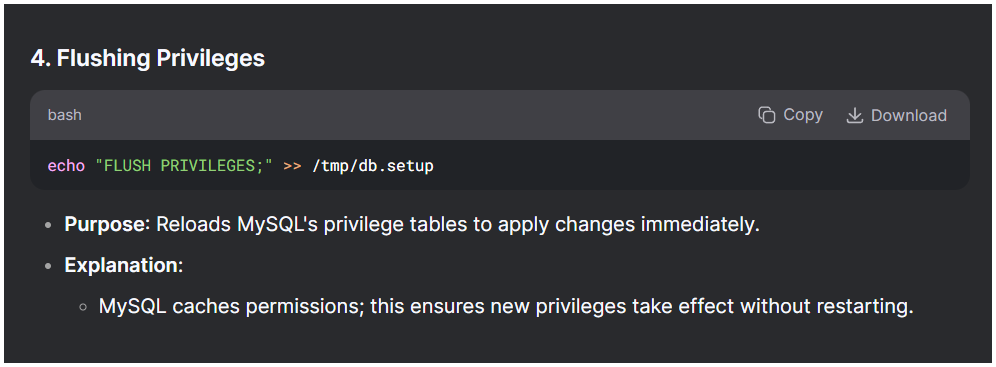
**Amazon RDS (Relational Database Service) is a managed database service that simplifies setup, operation, and scaling of relational databases in the cloud. It automates administrative tasks like hardware provisioning, database setup, patching, and backups while providing cost-efficient and resizable capacity.**

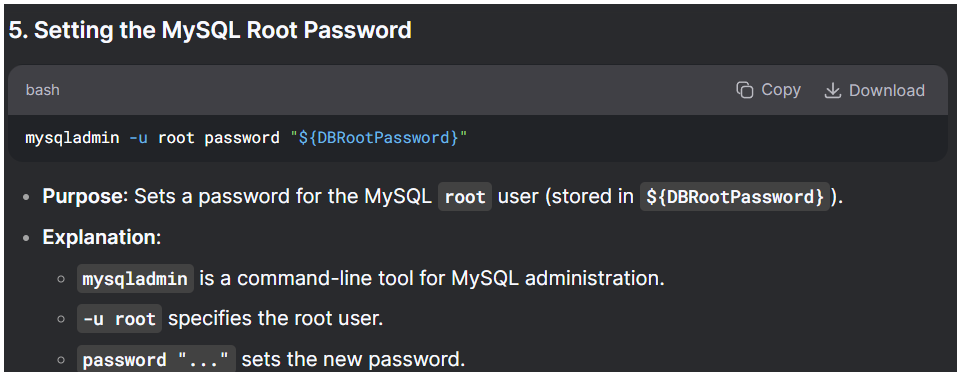
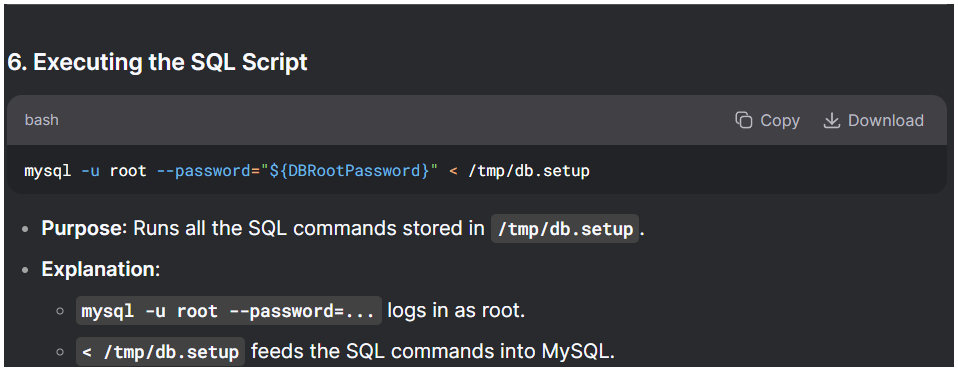
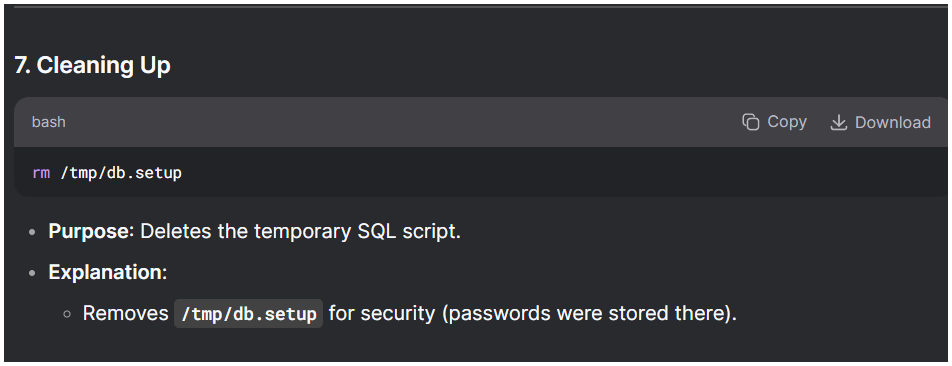
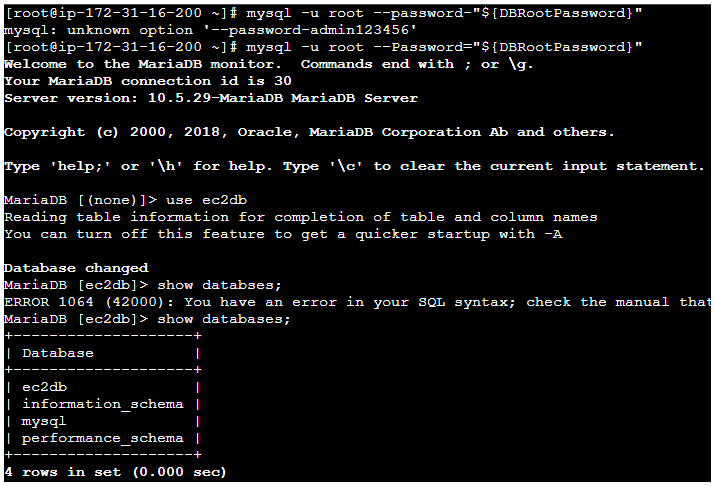
**1) Create Mariadb Db On Ec2.**

MARIA DB INSTALLATION  
# Update system  
sudo dnf update -y  
# Search available MariaDB versions  
sudo dnf search mariadb  
# Install MariaDB server (default version in AL2023 is 10.5+)  
sudo dnf install -y mariadb105-server  
# Enable and start MariaDB  
sudo systemctl enable --now mariadb  
# Check status  
systemctl status mariadb**  
  
  
  
🡪 Here I have installed in bash  
  
**Lsof -i tcp:3306

Set Environmental Variables  
===========================  
DBName=ec2db  
DBPassword=admin123456  
DBRootPassword=admin123456  
DBUser=ec2dbuser  
Database Setup on EC2 Instance:  
===============================  
  
  
echo "CREATE DATABASE ${DBName};" >> /tmp/db.setup

  
echo "CREATE USER '${DBUser}' IDENTIFIED BY '${DBPassword}';" >> /tmp/db.setup  
  
  
echo "GRANT ALL PRIVILEGES ON \*.\* TO '${DBUser}'@'%';" >> /tmp/db.setup

  
echo "FLUSH PRIVILEGES;" >> /tmp/db.setup

  
mysqladmin -u root password "${DBRootPassword}"  
  
  
  
mysql -u root --password="${DBRootPassword}" < /tmp/db.setup  
  
  
rm /tmp/db.setup  
 **  
Show databases;  
  
  
Why these settings?**

* **EBS gp3** gives predictable SSD performance and lets you raise IOPS/throughput separately from size (good for DBs.
* Locking **3306** to known sources reduces exposure.

**1. Deployment & High Availability (HA)**

| **Setting** | **Aurora** | **RDS for MariaDB** |
| --- | --- | --- |
| **Multi-AZ Deployment** | **✅ Yes (Aurora Replicas)** | **✅ Yes (Standby instance)** |
| **Read Replicas** | **✅ Up to 15 (low-latency, auto-scaling)** | **✅ Up to 5 (async replication)** |
| **Failover Time** | **~30 sec (faster)** | **~60-120 sec (slower)** |
| **Storage Auto-Scaling** | **✅ Yes (10GB → 128TB)** | **✅ Yes (up to 64TB)** |

**Key Differences:**

* **Aurora uses a shared storage cluster, while RDS MariaDB uses traditional EBS volumes.**
* **Aurora Replicas share storage, reducing replication lag compared to RDS async replicas.**

**2. Performance & Scaling**

| **Setting** | **Aurora** | **RDS for MariaDB** |
| --- | --- | --- |
| **Storage Type** | **SSD-backed, auto-scaling** | **GP2/GP3/IO1 (EBS)** |
| **Max Connections** | **Higher (depends on DB size)** | **Limited by instance type** |
| **Parallel Queries** | **✅ Yes (Aurora MySQL)** | **❌ No** |
| **Scaling (Reads)** | **✅ Fast (Aurora Replicas)** | **✅ Manual (Read Replicas)** |
| **Scaling (Writes)** | **✅ Fast (Aurora Serverless v2)** | **❌ Requires manual upgrade** |

**Key Differences:**

* **Aurora has 5x performance boost over standard MySQL/MariaDB due to optimized storage and caching.**
* **RDS MariaDB is more traditional, with predictable EBS-based performance.**

**3. Backup & Recovery**

| **Setting** | **Aurora** | **RDS for MariaDB** |
| --- | --- | --- |
| **Automated Backups** | **✅ Yes** | **✅ Yes** |
| **Point-in-Time Recovery (PITR)** | **✅ Yes** | **✅ Yes** |
| **Backup Retention** | **1-35 days** | **1-35 days** |
| **Snapshot Export to S3** | **✅ Yes** | **✅ Yes** |
| **Fast Cloning** | **✅ Yes (copy-on-write)** | **❌ No** |

**Key Differences:**

* **Aurora allows instantaneous snapshots (no performance impact).**
* **RDS MariaDB backups may slow down I/O during snapshots.**

**4. Security & Compliance**

| **Setting** | **Aurora** | **RDS for MariaDB** |
| --- | --- | --- |
| **Encryption at Rest** | **✅ Yes (KMS)** | **✅ Yes (KMS)** |
| **Encryption in Transit** | **✅ Yes (SSL/TLS)** | **✅ Yes (SSL/TLS)** |
| **IAM Authentication** | **✅ Yes** | **✅ Yes** |
| **Network Isolation (VPC)** | **✅ Yes** | **✅ Yes** |
| **Automated Patching** | **✅ Yes** | **✅ Yes** |

**Key Differences:**

* **Both support AWS KMS encryption, but Aurora has tighter integration with IAM roles.**

**5. Cost Comparison**

| **Setting** | **Aurora** | **RDS for MariaDB** |
| --- | --- | --- |
| **Pricing Model** | **Per vCPU + storage** | **Per instance + storage** |
| **Storage Cost** | **Higher (but more efficient)** | **Lower (EBS-based)** |
| **Serverless Option** | **✅ Aurora Serverless v2** | **❌ No** |

**Key Differences:**

* **Aurora is ~20% more expensive but offers better performance.**
* **RDS MariaDB is cheaper for small workloads.**

**6. When to Use Which?**

**Choose Aurora If:**

**✔ Need high performance (5x faster than MySQL/MariaDB).  
✔ Want automatic scaling (Aurora Serverless v2).  
✔ Need low-latency read replicas (Aurora Replicas).**

**Choose RDS for MariaDB If:**

**✔ Prefer traditional MariaDB compatibility.  
✔ Have budget constraints (cheaper for small workloads).  
✔ Don’t need Aurora’s advanced features.**

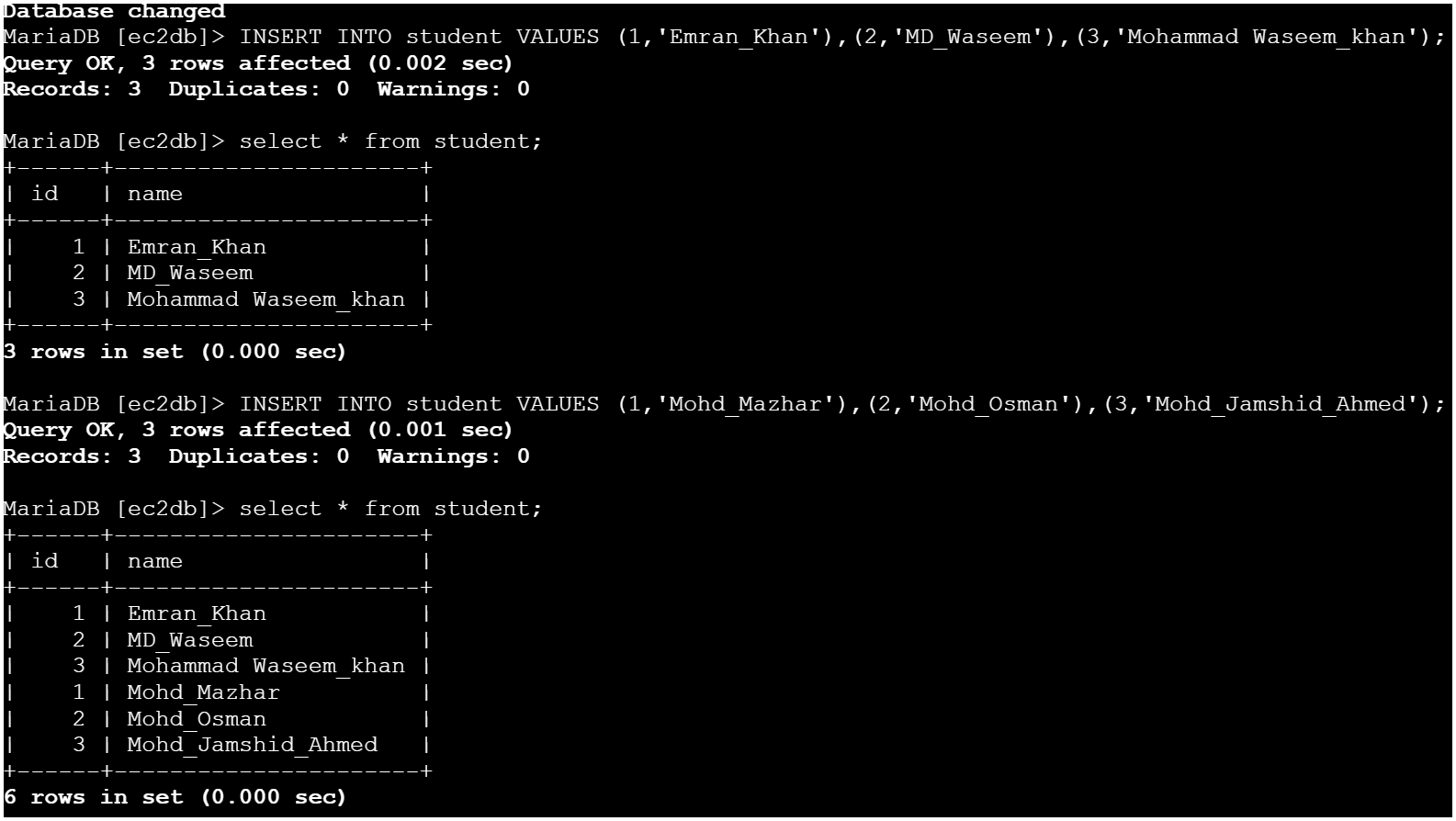
**Final Recommendation**

* **For mission-critical, high-performance apps → Aurora.**
* **For cost-sensitive, traditional MariaDB workloads → RDS MariaDB.**

**2) Insert Some Dummy Data**

**IN mariadb type the below**

* **Create table student (id INT, name VARCHAR (45));**
* **INSERT INTO student VALUES (1,’Emran’);**

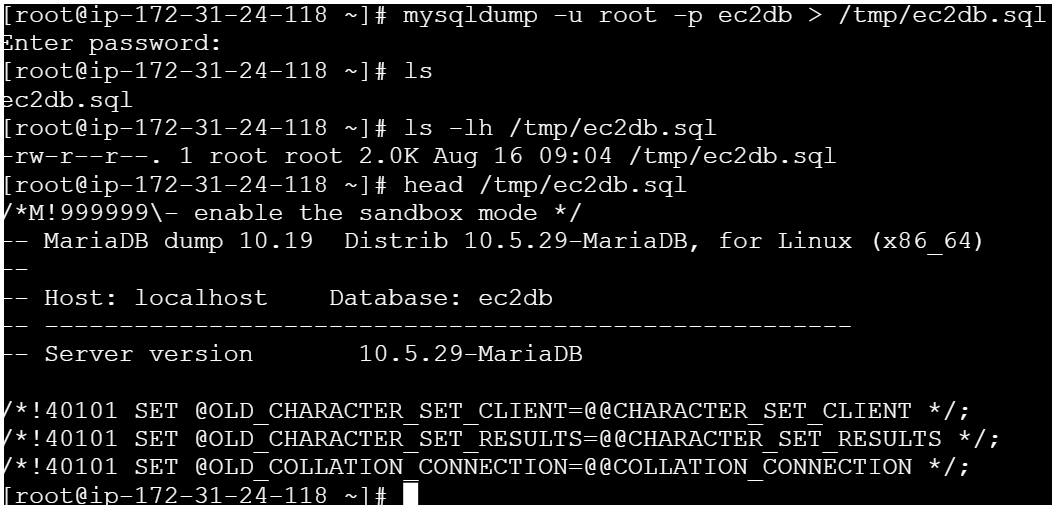
****

**3) Take The Backup Of Dummy Data On Ec2**

**1. Backup of ec2db →(stay in root user and type this)**

**mysqldump -u root -p ec2db > /tmp/ec2db.sql  
Enter password:  
  
2. Dump file saved at /tmp/ec2db.sql  
  
3. Checking file exist or not. ls -lh /tmp/ec2db.sql**

**4. View content head /tmp/ec2db.sql  
  
5) launch MariaDB RDS instance.  
Open AWS Console → RDS → Databases → Create   
database  
Engine options → Choose MariaDB**

**  
  
  
  
  
  
  
  
  
  
  
  
  
  
4) Launch Mariadb RDS Instance.**

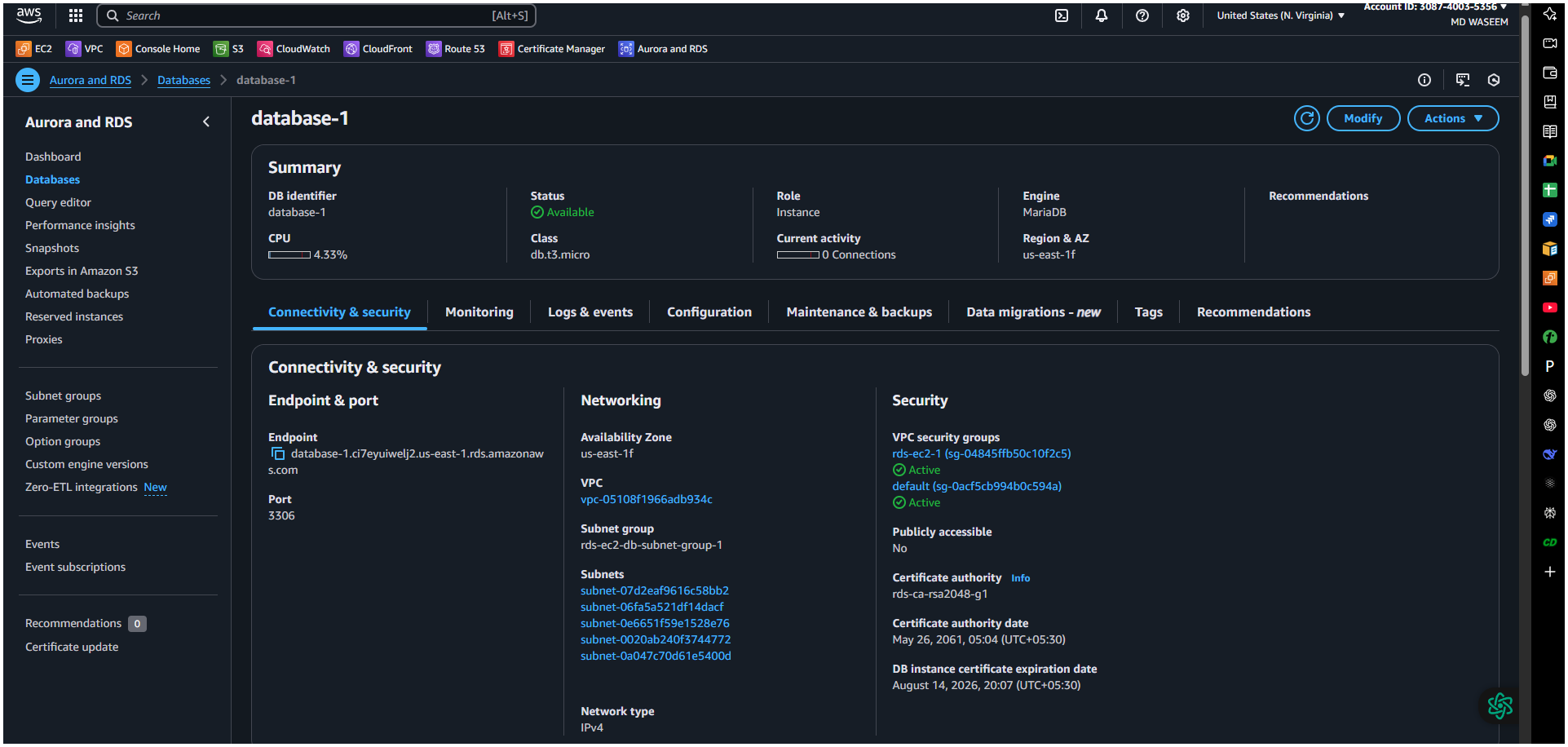
* **Open AWS Console → RDS → Databases → Create**

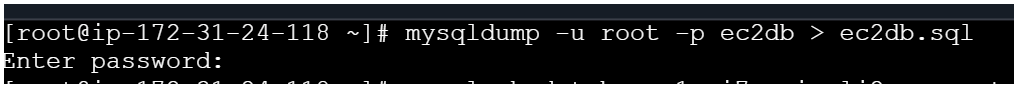
**Database**

* **Engine options → Choose MariaDB**
* **Version → Choose e.g. 10.6**
* **Templates → “Free tier” or “Production”**
* **DB instance identifier → e.g., mariadb-lab**
* **Master username → admin**
* **Master password → set strong password**
* **Instance configuration → Choose db.t3.micro for free tier**
* **Connectivity →**
* **● VPC: Choose your VPC**
* **● Public access: No (recommended)**
* **● Security group: Choose SG that allows port 3306 from EC2**
* **Database authentication → Password authentication**

**Click Create database**

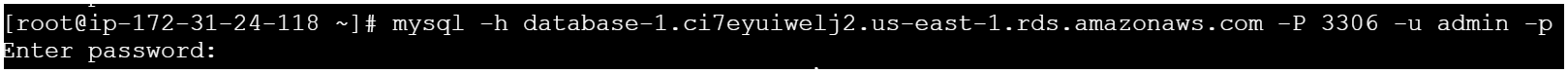
**Wait for Status → Available**

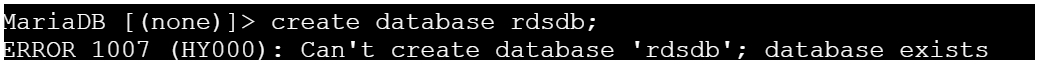
 **5) Migrate Database From Ec2 To RDS.  
  
. Get the dump of your existing DB on EC2  
Taking backup →(stay in root use and type this)**

**mysqldump -u root -p ec2db > ec2db.sql**

**2. Login to maridb  
mysql -h database-1.c030o8c66sum.us-  
east-1.rds.amazonaws.com -P 3306 -u admin -p**

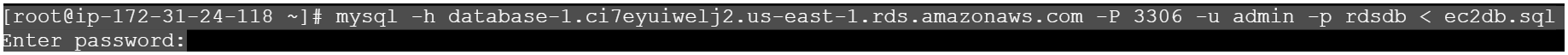
**Enter Password**

****

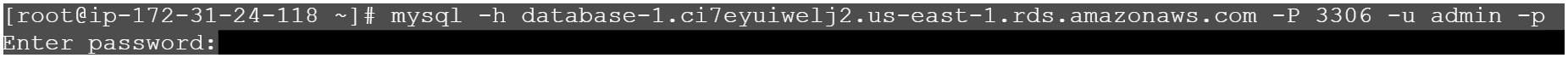
**3. Create Database Rdsdb; And Exit**

**I got this error cause I have already created before ( for testing )**

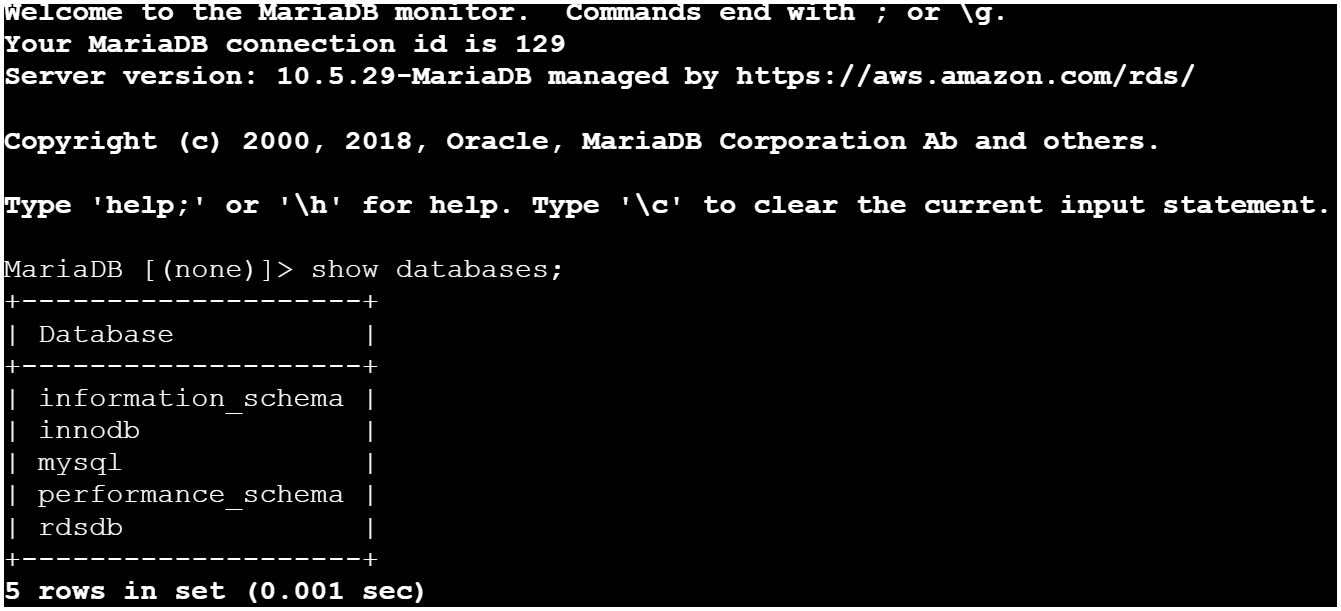
**4. Migrate the DB dump that you have taken in step 1 to RDS.  
mysql -h database-1.c030o8c66sum.us-  
east-1.rds.amazonaws.com -P 3306 -u admin -p rdsdb < ec2db.sql**

**Enter Password Of Your Database Here**

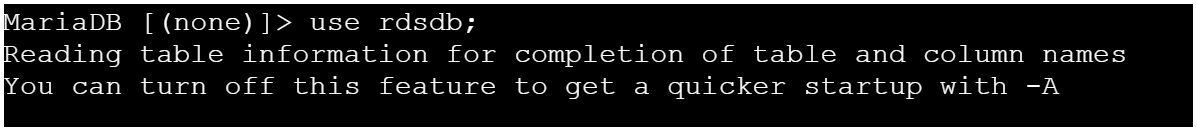
**5. Connect to your RDS DB instance  
 mysql -h database-1.c030o8c66sum.us-  
east-1.rds.amazonaws.com -P 3306 -u admin -p**

**Enter the password of your database here**

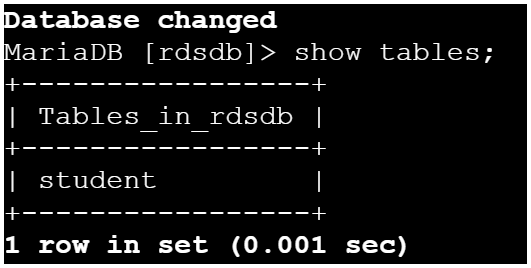
**6.Show databases;**

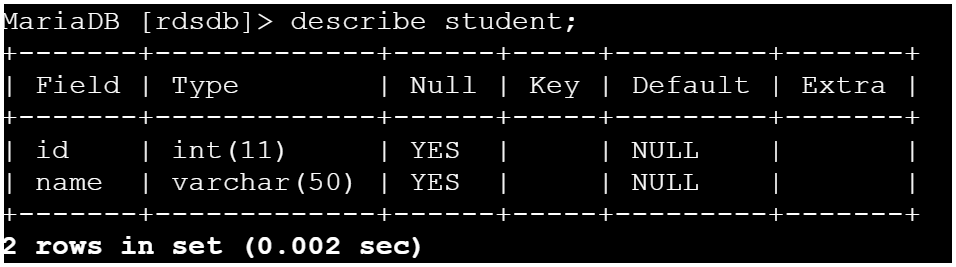
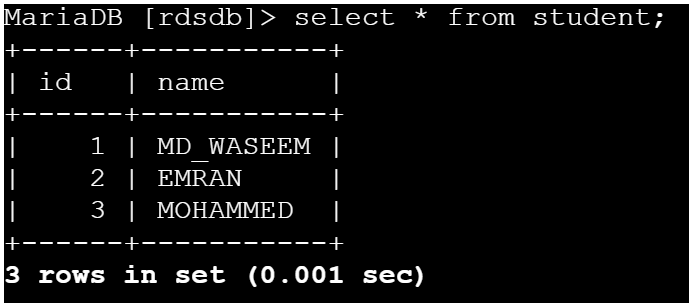
****

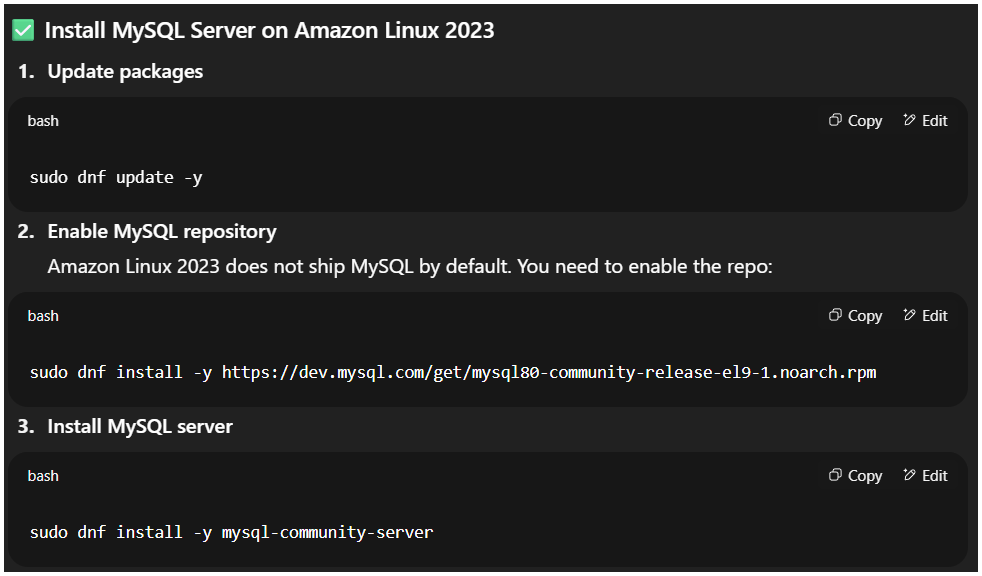
**7. Use rdsdb;**

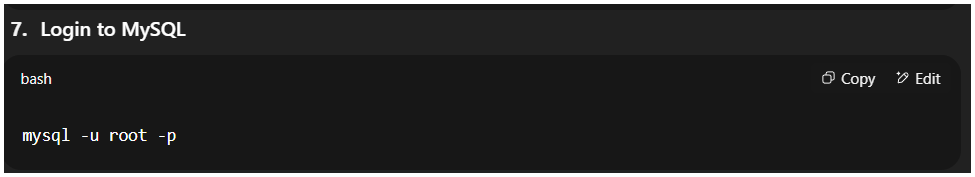
****

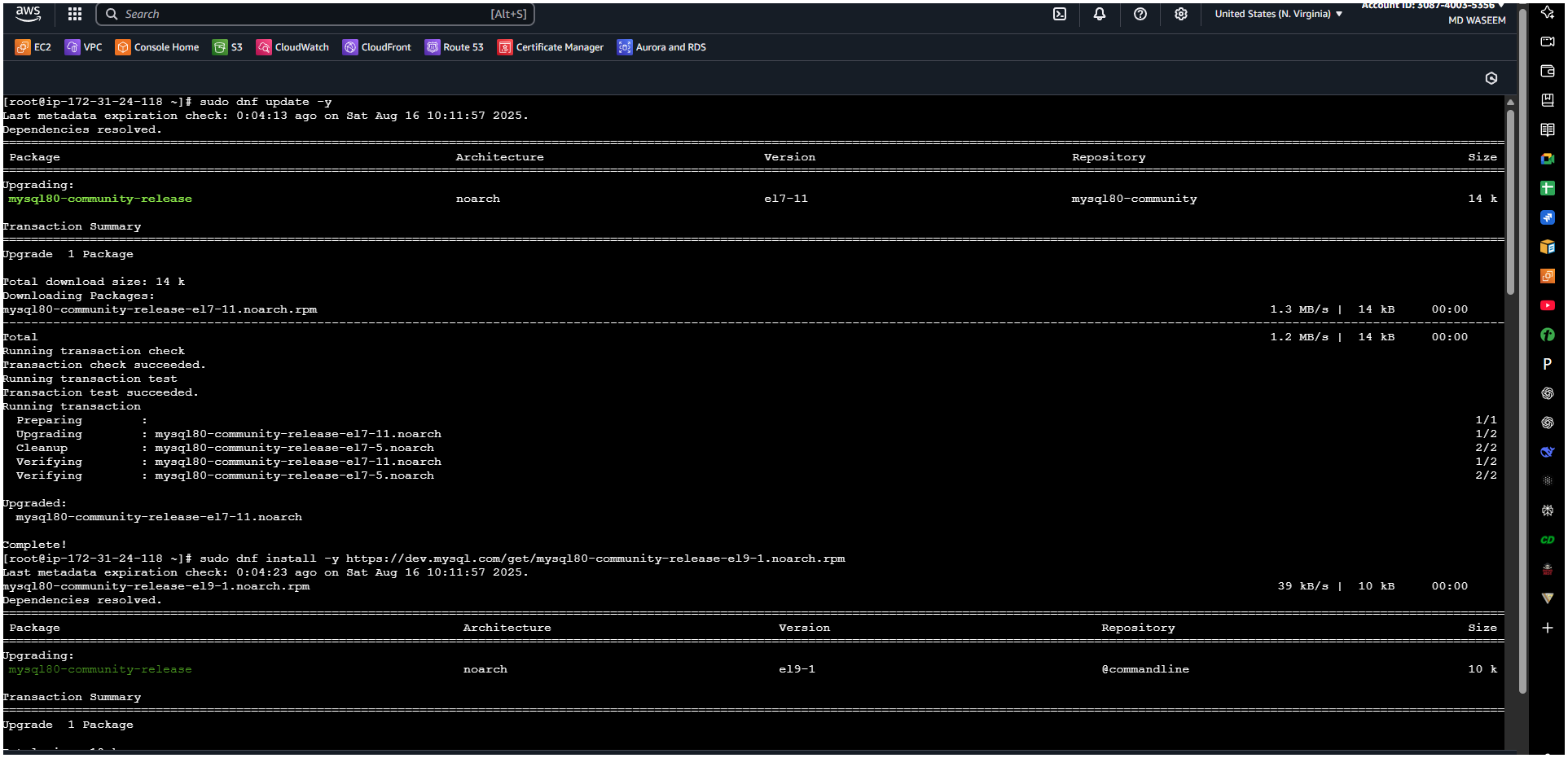
**8. Show tables;**

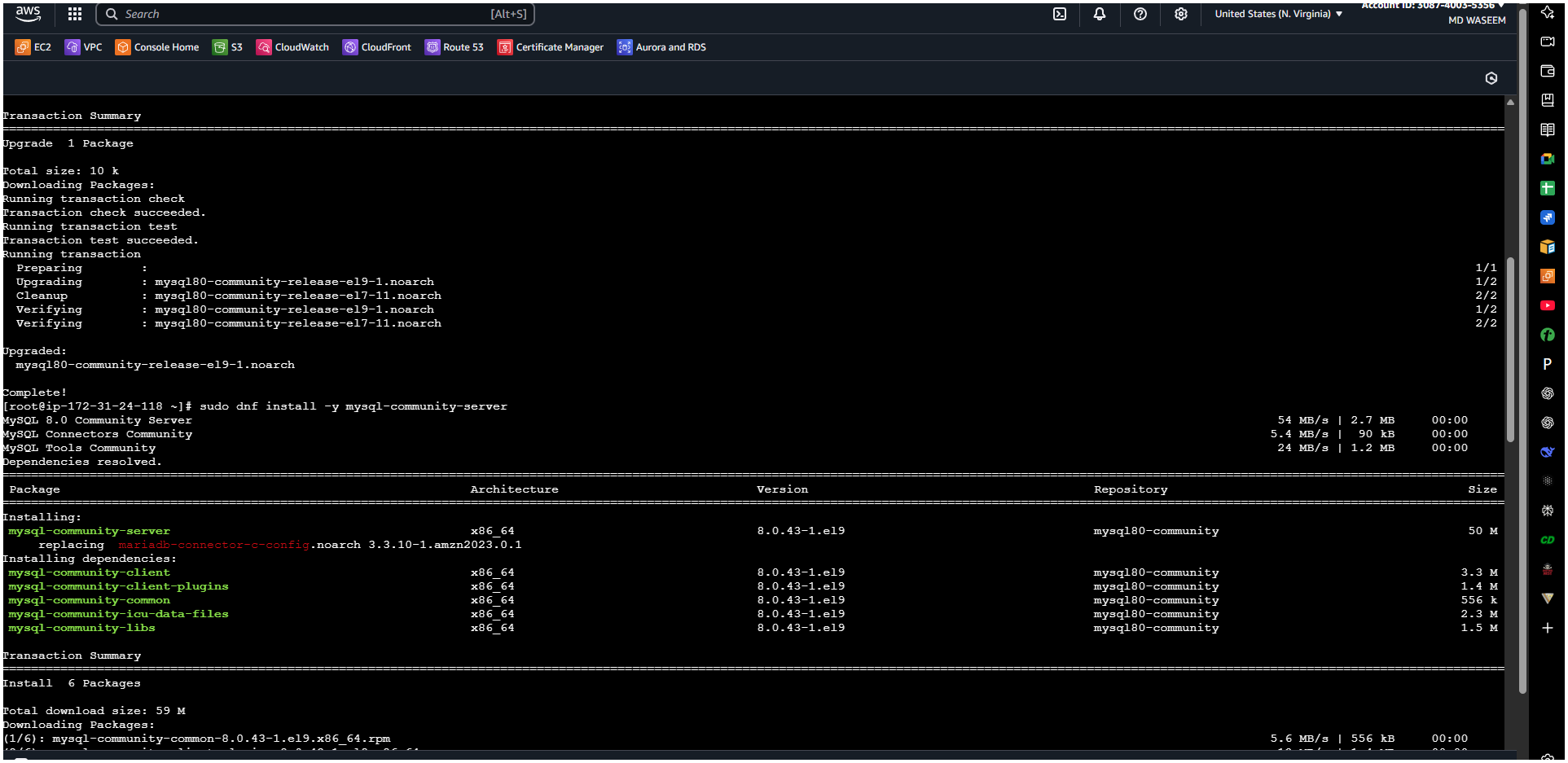
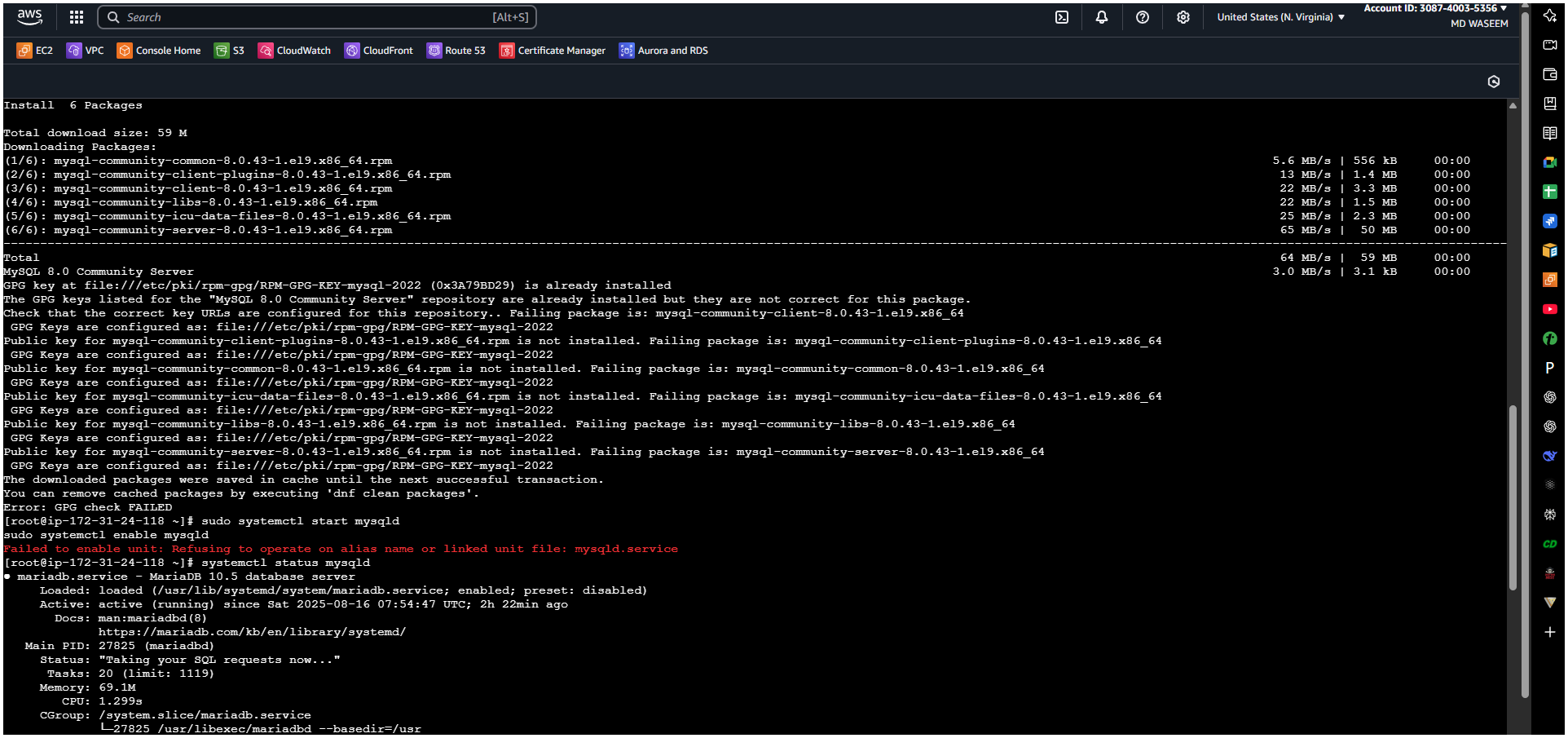
**  
  
  
  
  
  
  
  
  
  
  
9. Describe student;**

**  
  
  
  
10. Select \* from student;  
  
  
  
  
  
  
  
  
  
  
  
  
  
6) Install MySQL DB on ec2**

**  
**

****

****

**  
**

**7) Launch MySQL RDS image.**

**🔹 Step 1: Open RDS Console**

1. **Go to AWS Console → RDS → Databases → Create database**

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

* **Select Standard create (for more control).**
* **Or choose Easy create (AWS picks defaults for you).**

**🔹 Step 3: Choose Engine**

* **Select MySQL**
* **Choose the version (e.g., MySQL 8.0).**

**🔹 Step 4: Templates**

* **Production (high availability, Multi-AZ, backups, monitoring).**
* **Dev/Test (cheaper, for learning/testing).**

**🔹 Step 5: Settings**

* **DB instance identifier → e.g., my-mysql-db**
* **Master username → e.g., admin**
* **Master password → set a strong password**

**🔹 Step 6: Instance Configuration**

* **Choose instance type → e.g., db.t3.micro (free tier eligible).**

**🔹 Step 7: Storage**

* **Select storage type → e.g., gp3**
* **Set allocated storage (default: 20 GB).**
* **Enable storage autoscaling if needed.**

**🔹 Step 8: Connectivity**

* **VPC → choose default VPC (or custom if you have one).**
* **Subnet group → select default.**
* **Public access → Choose Yes if you want to connect from outside AWS.**
* **VPC security group → create new or choose existing (make sure port 3306 is allowed).**

**🔹 Step 9: Additional Configuration**

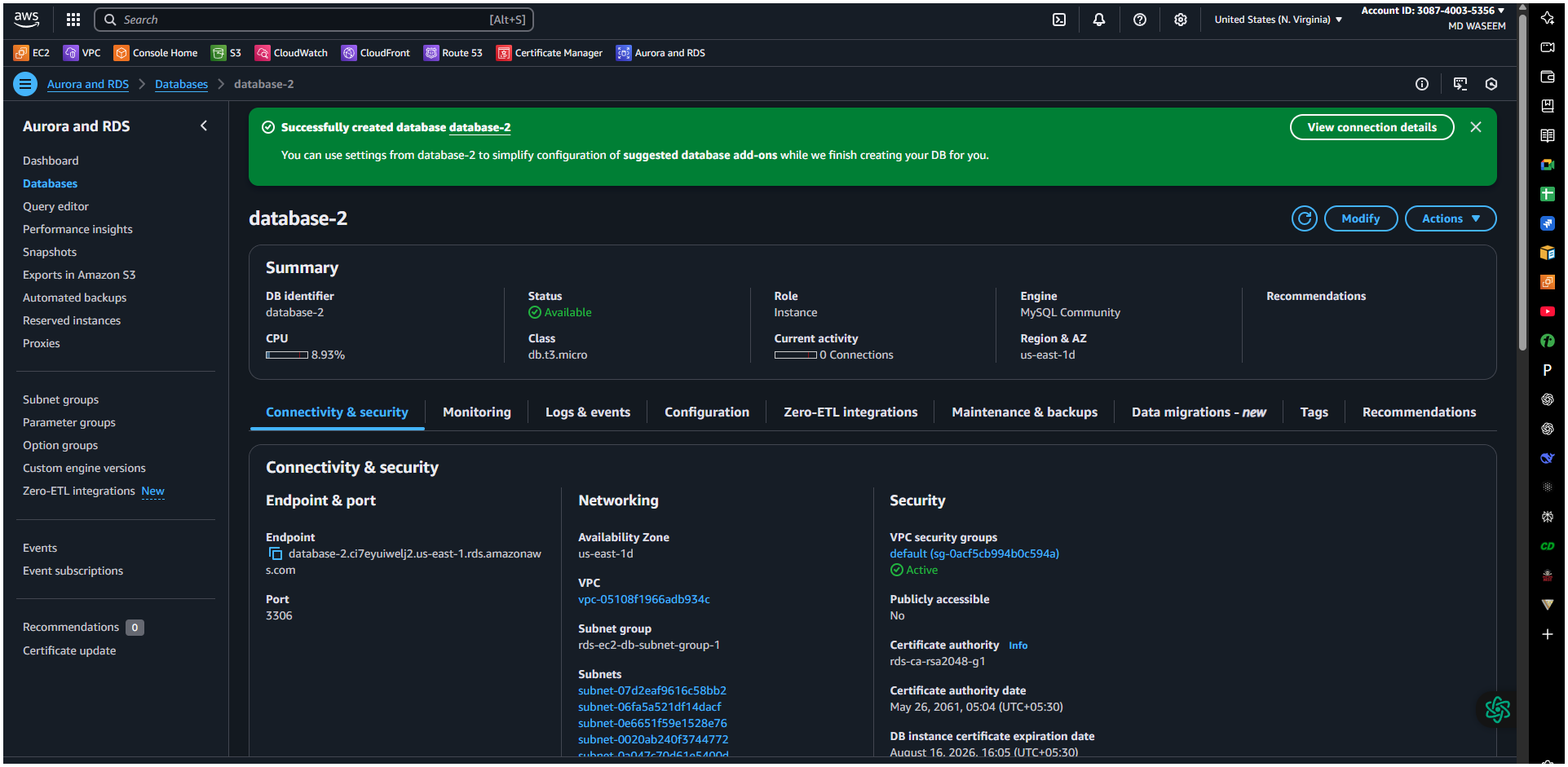
* **Database name → e.g., testdb**
* **DB parameter groups, option groups → leave default.**
* **Backup, Monitoring, Maintenance → leave defaults (or customize).**

**🔹 Step 10: Launch**

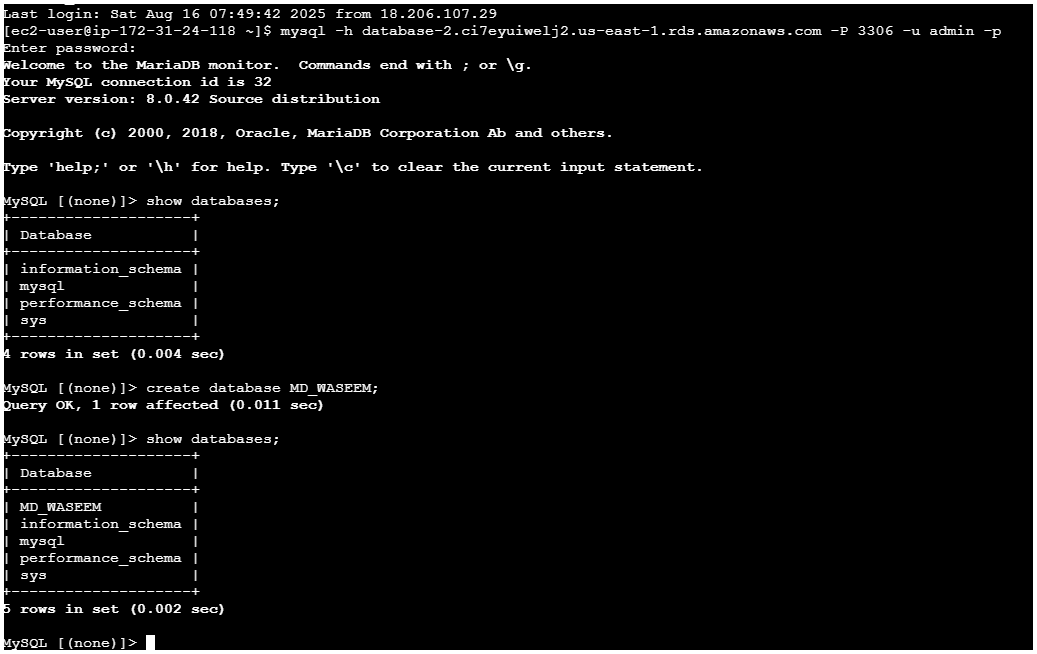
* **Click Create database**
* **Wait a few minutes → Status will become Available**

**🔹 Step 11: Connect to RDS**

1. **Go to RDS → Databases → Click your DB.**
2. **Copy the endpoint (something like mydb.abc123xyz.us-east-1.rds.amazonaws.com).**
3. **On your EC2 or local machine, run:**

****

**TESTING@mysql RDS IMG**

****

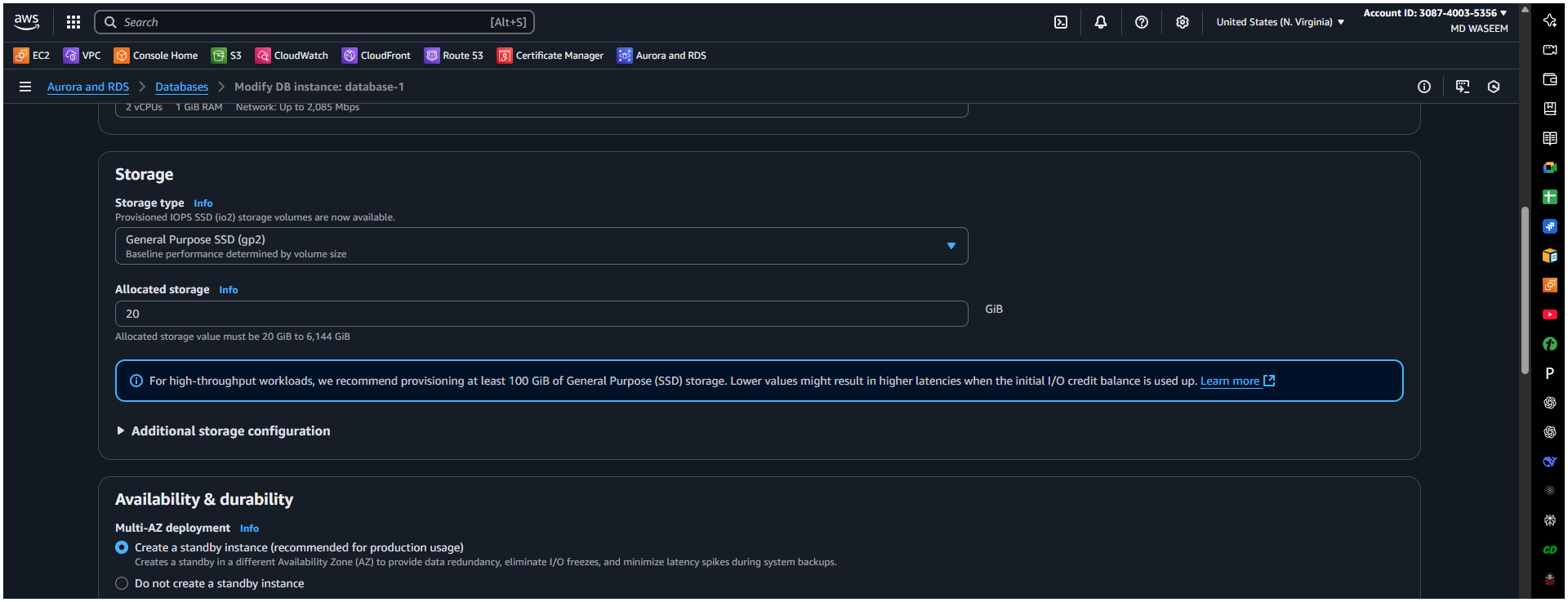
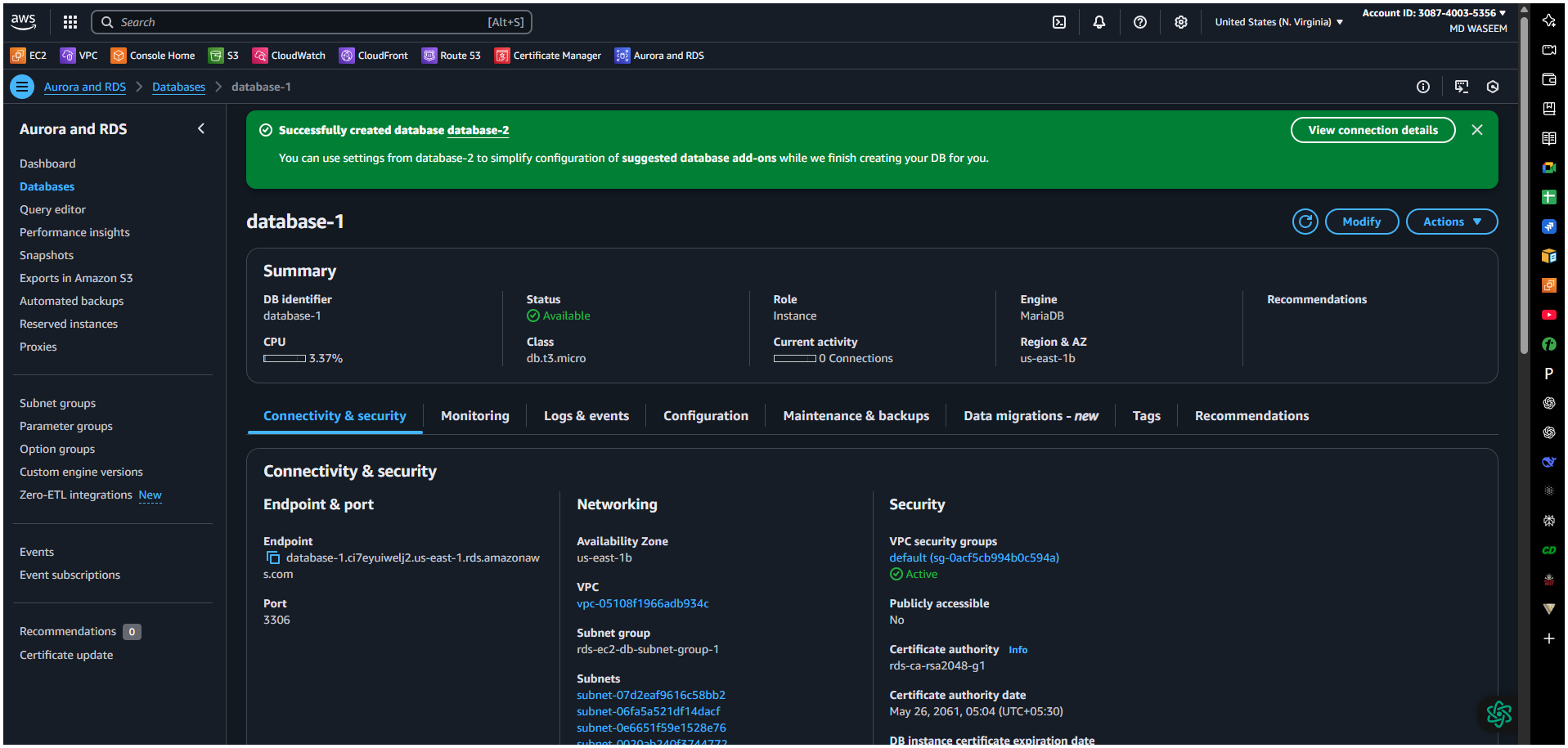
**8) Configure multi AZ**

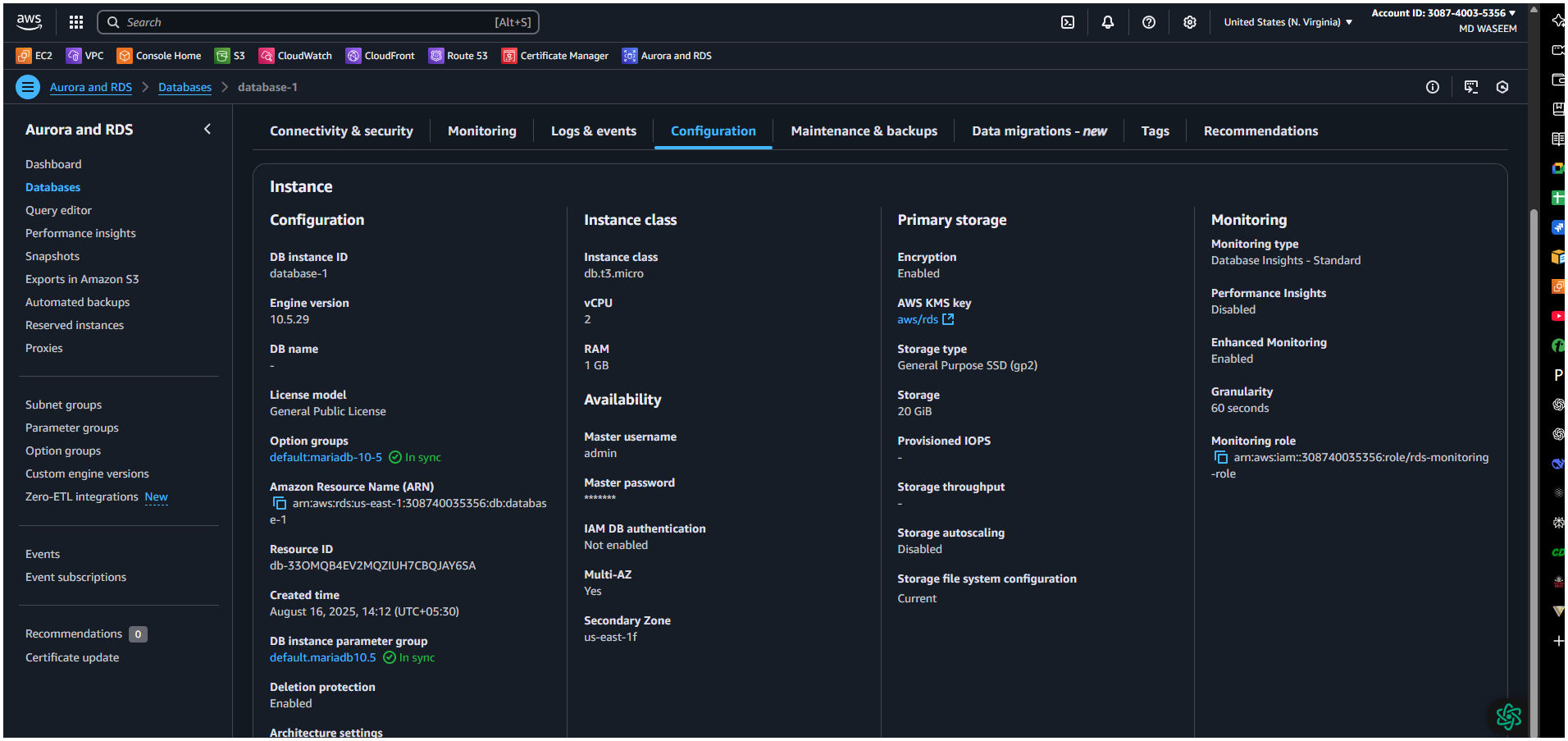
**In AWS Console → RDS → Select your DB → Modify or (we   
can go from action → convert to Multi-AZ)  
Under Availability &** **In AWS Console → RDS → Select your DB → Modify or (we**

**can go from action → convert to Multi-AZ)**

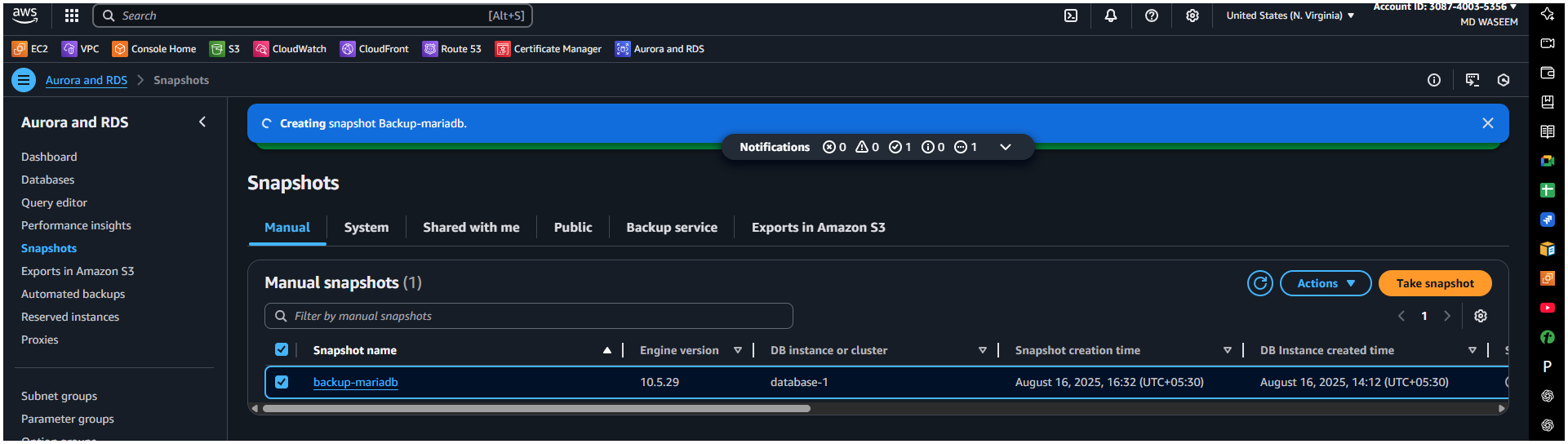
**Under Availability & durability, enable Multi-AZ deployment**

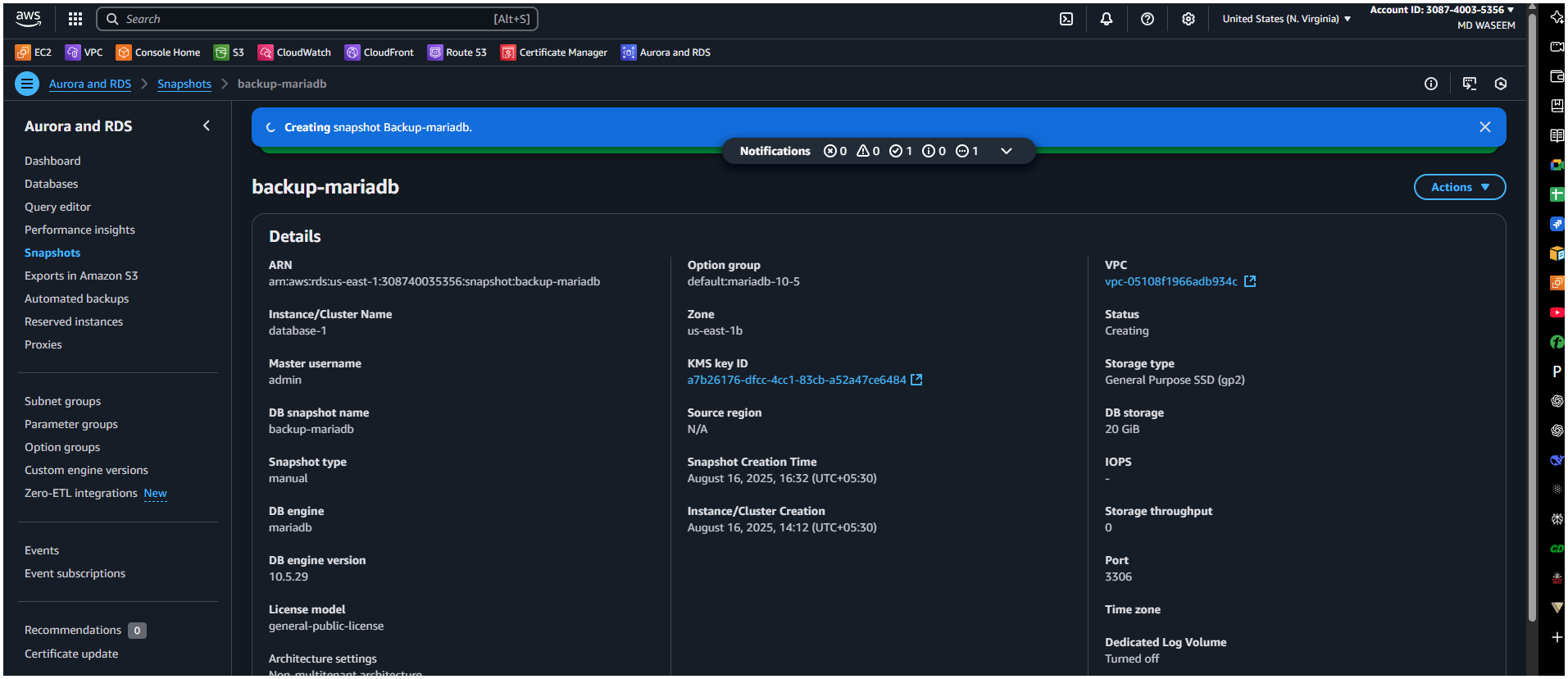
**Apply immediately durability, enable Multi-AZ deployment  
Apply immediately**

**  
**

In configuration tab we can check multi-AZ us-east-1d ****

**9) Take Backup of DB and restore the DB**

Go to actions → snapshot and name it and create ****

**  
  
  
  
  
  
  
  
  
  
  
  
  
10) Create Read Replica**

**What is a Read Replica?**

A **Read Replica** is a copy of your primary (source) database that **only handles read operations**. It helps:

* **Offload read traffic** from the primary database (improving performance).
* **Scale read-heavy workloads** (e.g., analytics, reporting, dashboards).
* **Reduce latency** by placing replicas in different regions.
* **Serve as a backup** (though not a replacement for backups).

**Key Differences: Read Replica vs. Multi-AZ**

| **Feature** | **Read Replica** | **Multi-AZ** |
| --- | --- | --- |
| **Purpose** | Read scaling | High availability (HA) |
| **Writes allowed?** | No (read-only) | Yes (primary instance) |
| **Automatic failover?** | No (manual promotion) | Yes (auto-failover) |
| **Replication type** | Asynchronous | Synchronous (for standby) |
| **Use case** | Reporting, analytics | Mission-critical workloads |

**How to Create a Read Replica in AWS RDS (MySQL) - Step by Step**

**Prerequisites**

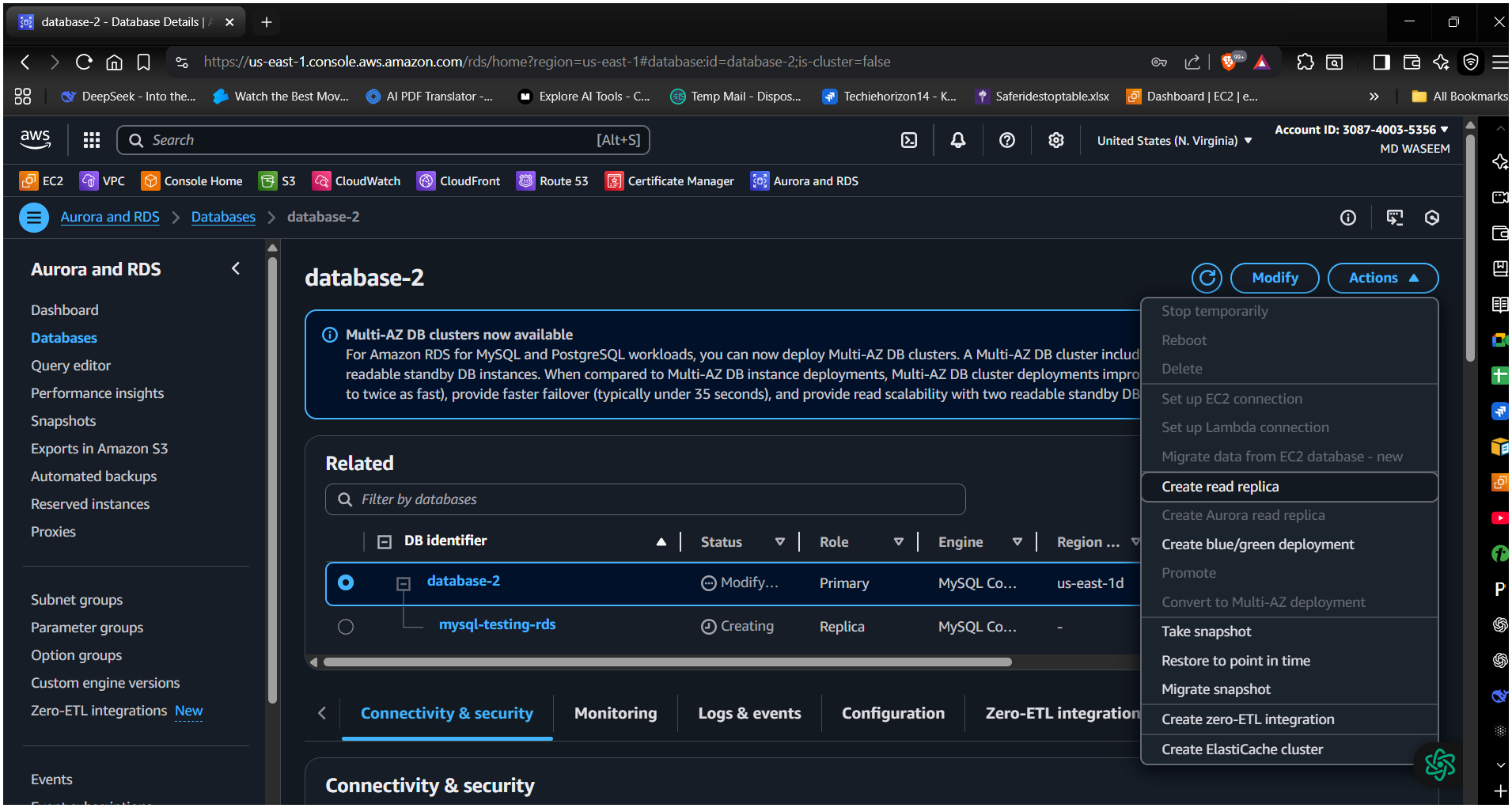
✅ A running **RDS MySQL** instance (primary DB).  
✅ **Automatic backups** must be enabled (required for replicas).

**Step 1: Go to AWS RDS Console**

1. Log in to the [**AWS Management Console**](https://aws.amazon.com/console/).
2. Navigate to **RDS > Databases**.

**Step 2: Select the Source DB Instance**

1. Click on the **primary MySQL DB** you want to replicate.
2. Click **Actions > Create read replica**.



**Step 3: Configure the Read Replica**

**1. Replica Settings**

* **DB instance identifier**: Enter a unique name (e.g., mysql-replica-1).
* **Instance class**: Choose the same or smaller size than the primary.
* **Multi-AZ deployment**: (Optional) Enable for HA on the replica itself.

**2. Connectivity & Storage**

* **Virtual Private Cloud (VPC)**: Same as primary (or cross-region if needed).
* **Public access**: Disable unless required.
* **Storage type**: SSD (gp2/io1) for better performance.

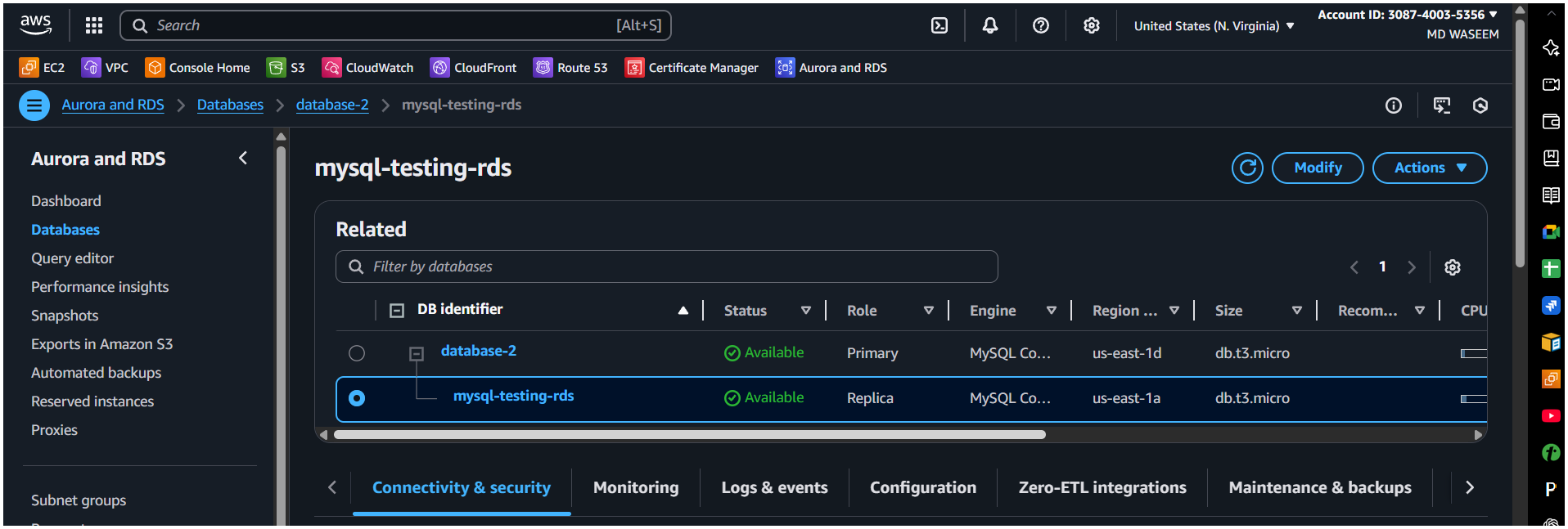
**3. Additional Settings**

* **Enable automatic backups**: Recommended (required for future replicas).
* **Maintenance window**: Set a low-traffic time.

**Step 4: Launch the Read Replica**

1. Click **Create read replica**.
2. Wait ~5-15 mins for AWS to provision and sync the replica.

**@@@@it took me 20 min@@@@**

****

**Step 5: Verify & Use the Read Replica**

✅ **Check replication status**:

* Go to **RDS > Databases** and check the replica’s status.
* Once **"Available"**, it’s ready for read traffic.

✅ **Test read operations**:

* Connect to the **replica’s endpoint** (different from the primary).
* Run SELECT queries to confirm it works.

✅ **Monitor replication lag**:

* Use Amazon CloudWatch metrics (ReplicaLag).
* High lag? Check workload or resize the replica.