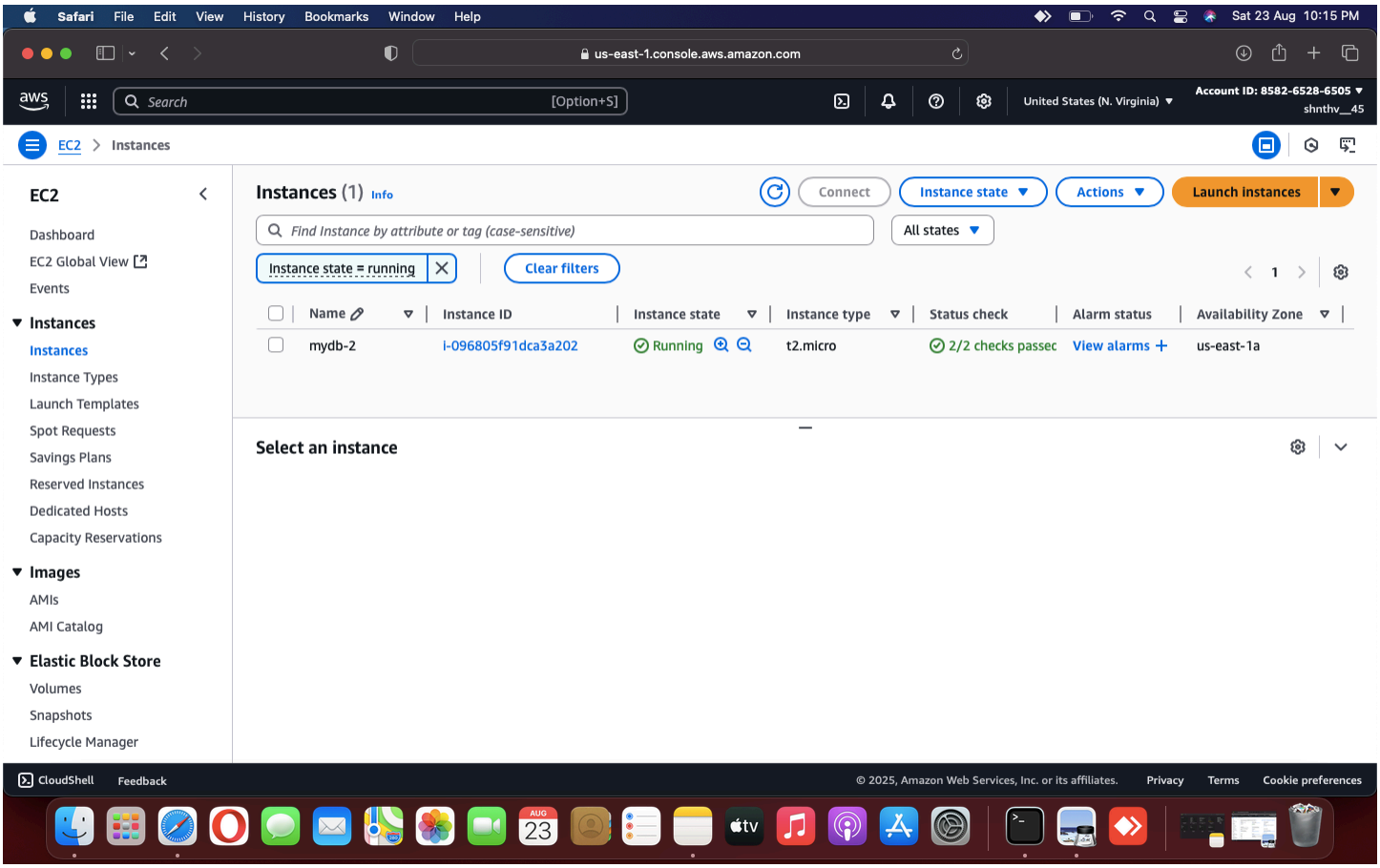
NullClass Cloud Technology (AWS)

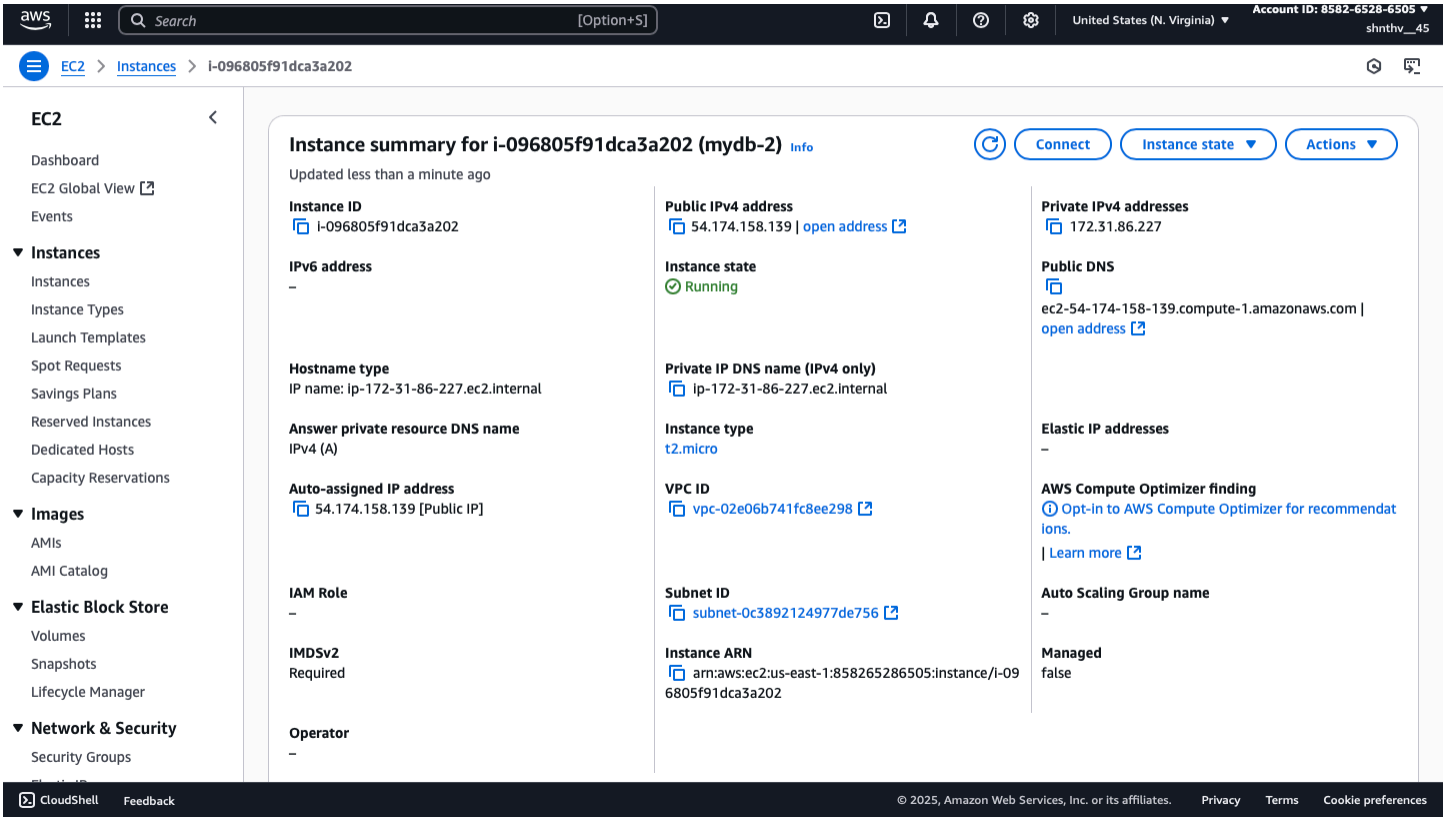
1.Create an RDS MySQL Database and Connect from EC2 Launch an RDS MySQL instance inside your VPC. Modify the security group to allow connections from your EC2 instance. Connect to the database from your EC2 instance using MySQL CLI.

This guide explains how to set up an **Amazon RDS MySQL database** and connect to it from an **EC2 instance**. We will launch an EC2 server, create an RDS MySQL database in the same VPC, configure security groups to allow communication, install the MySQL client on EC2, and then connect to the database. Finally, we will create a table, insert sample records, and verify the connection.

1. Launch an EC2 Instance

1. Log in to the **AWS Management Console**.
2. Navigate to **EC2 > Instances > Launch Instance**.
3. Choose an Amazon Machine Image (AMI):
   - Select **Amazon Linux 2023** (or Amazon Linux 2).
4. Choose instance type:
   - Example: **t2.micro** (Free Tier eligible).
5. **Key Pair**: Create or select an existing key pair (e.g., sonu-3.pem).
6. Network settings:
   - Attach to your default VPC.
   - Enable a public IP.
7. **Launch** the instance.

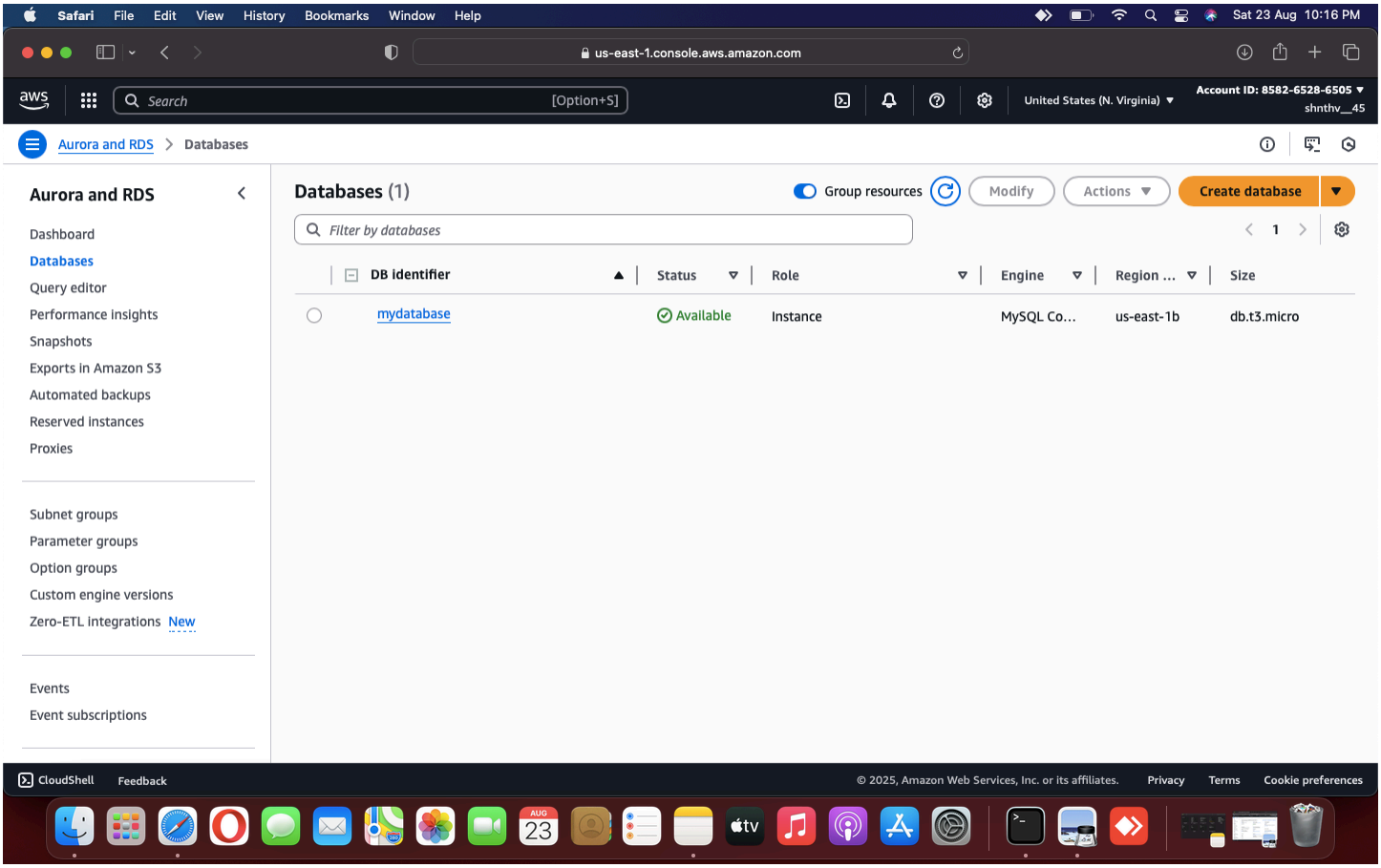


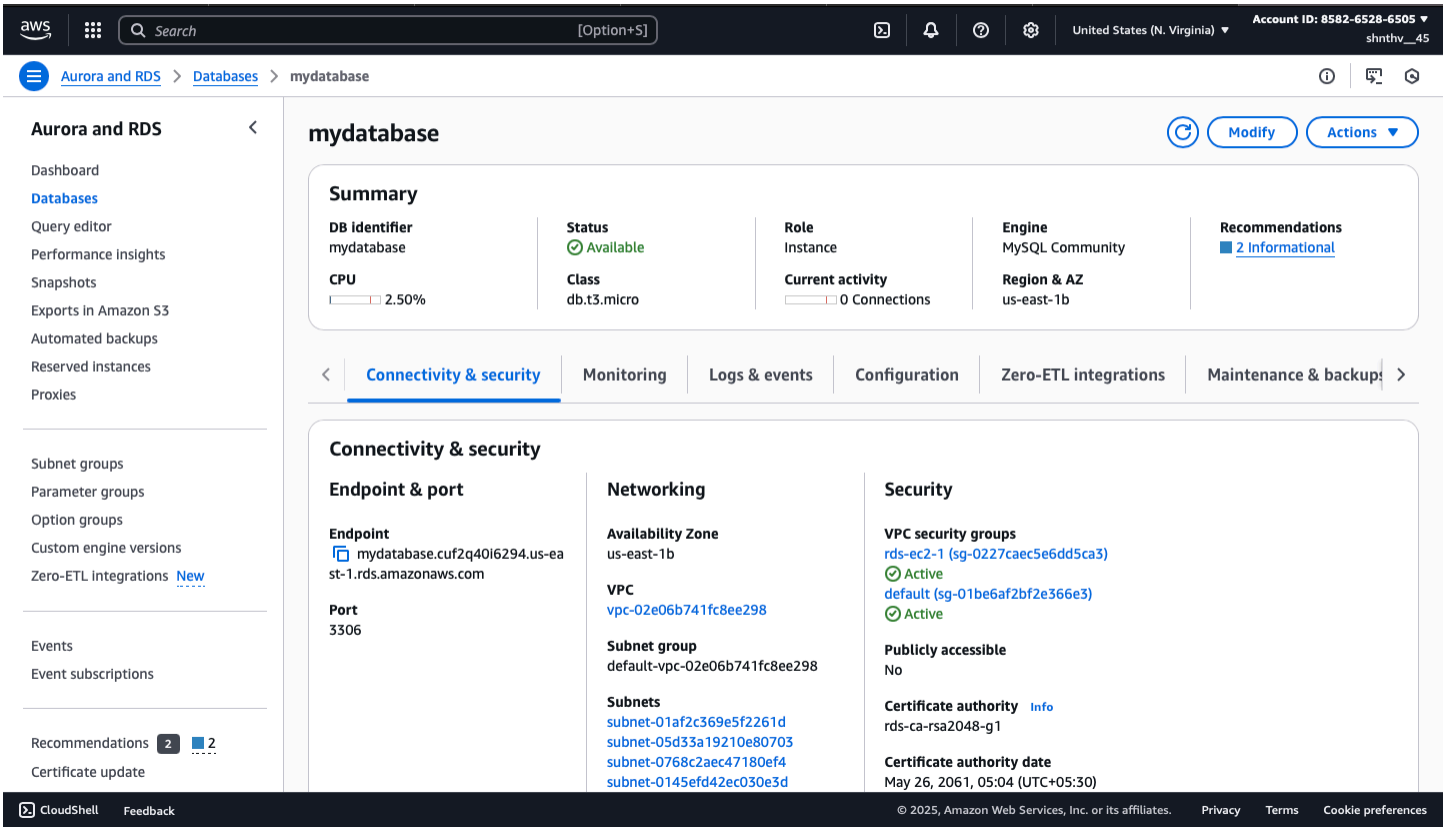


2. Launch an RDS MySQL Database

1. Go to **RDS > Databases > Create Database**.
2. Choose **Standard Create** → **MySQL**.
3. Select version: Example **MySQL 8.0**.

4. Choose **Free Tier** template.
5. DB instance identifier:
    - Example: mydatabase.
6. Set username & password:
    - Username: databasemysql.
    - Enter a secure password.
7. Connectivity:
    - VPC: Select the same VPC as your EC2 instance.
    - Subnet group: Default.
    - Public access: **No** (private DB).
    - Security group: Select or create one (we'll configure it next).
8. Create the database.



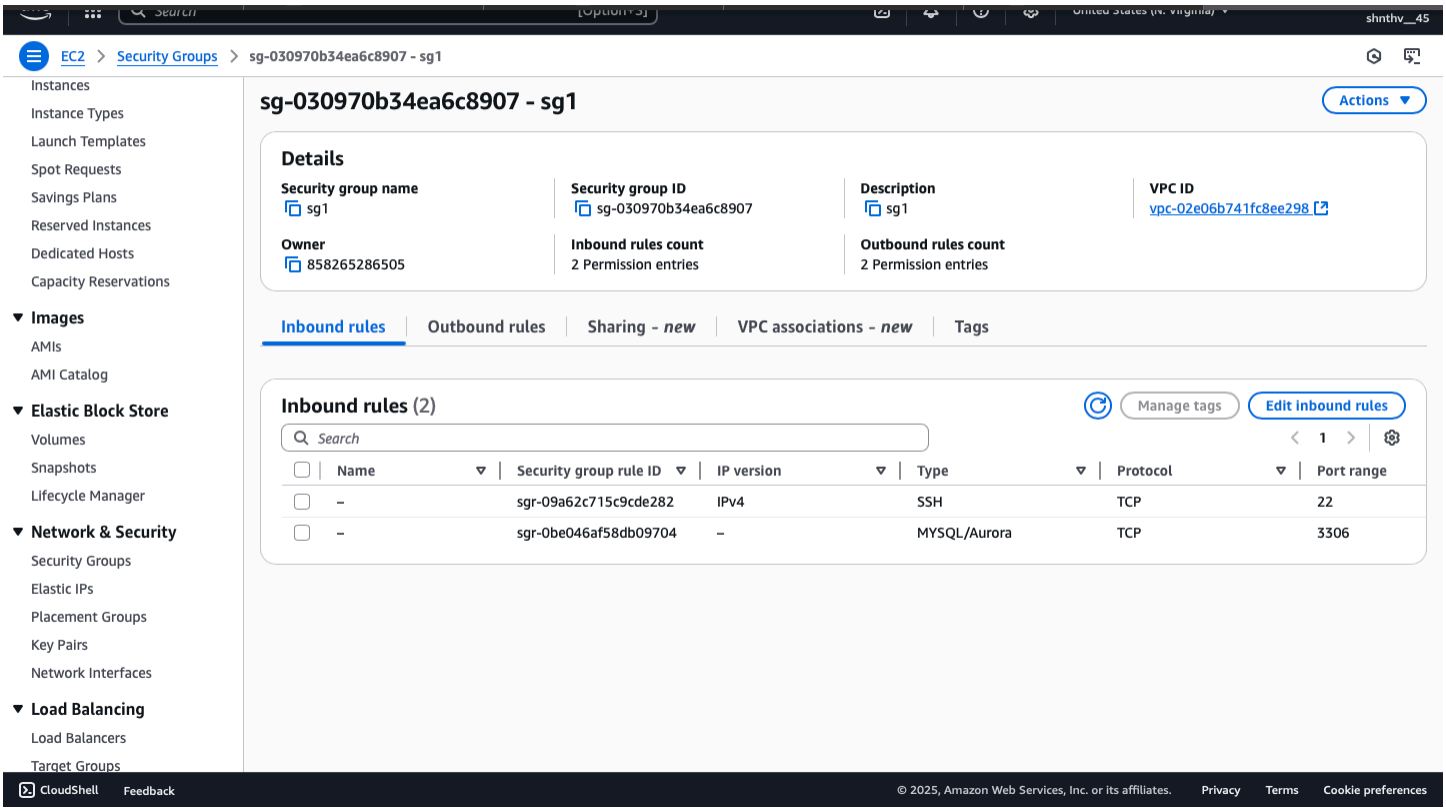


3. Configure Security Groups

For EC2 to connect to RDS, security rules must allow traffic:

- **On RDS Security Group**:
  1. Go to **EC2 > Security Groups**.
  2. Find the RDS security group (e.g., rds-ec2-1).
  3. Edit inbound rules → Add rule:
     - **Type**: MySQL/Aurora
     - **Protocol**: TCP
     - **Port**: 3306
     - **Source**: Custom → Select your EC2's Security Group
- **On EC2 Security Group**:
  1. Add inbound rule:
     - **Type**: SSH
     - **Protocol**: TCP
     - **Port**: 22
     - **Source**: 0.0.0.0/0 (for testing; later restrict to your IP).

Now EC2 can reach RDS on port **3306**.



4. Connect to Your EC2 Instance via SSH

1. Open a terminal (Linux/Mac) or PowerShell (Windows).
2. Locate your key file: sonu-3.pem.
3. Set permissions so it's not publicly visible:
4. chmod 400 sonu-3.pem
5. Connect using EC2's **Public DNS**:
6. ssh -i "sonu-3.pem" ec2-user@ec2-54-174-158-139.compute-1.amazonaws.com
7. Once connected, you'll be inside your EC2 shell.

5. Install MySQL Client on EC2

On Amazon Linux 2023, run:

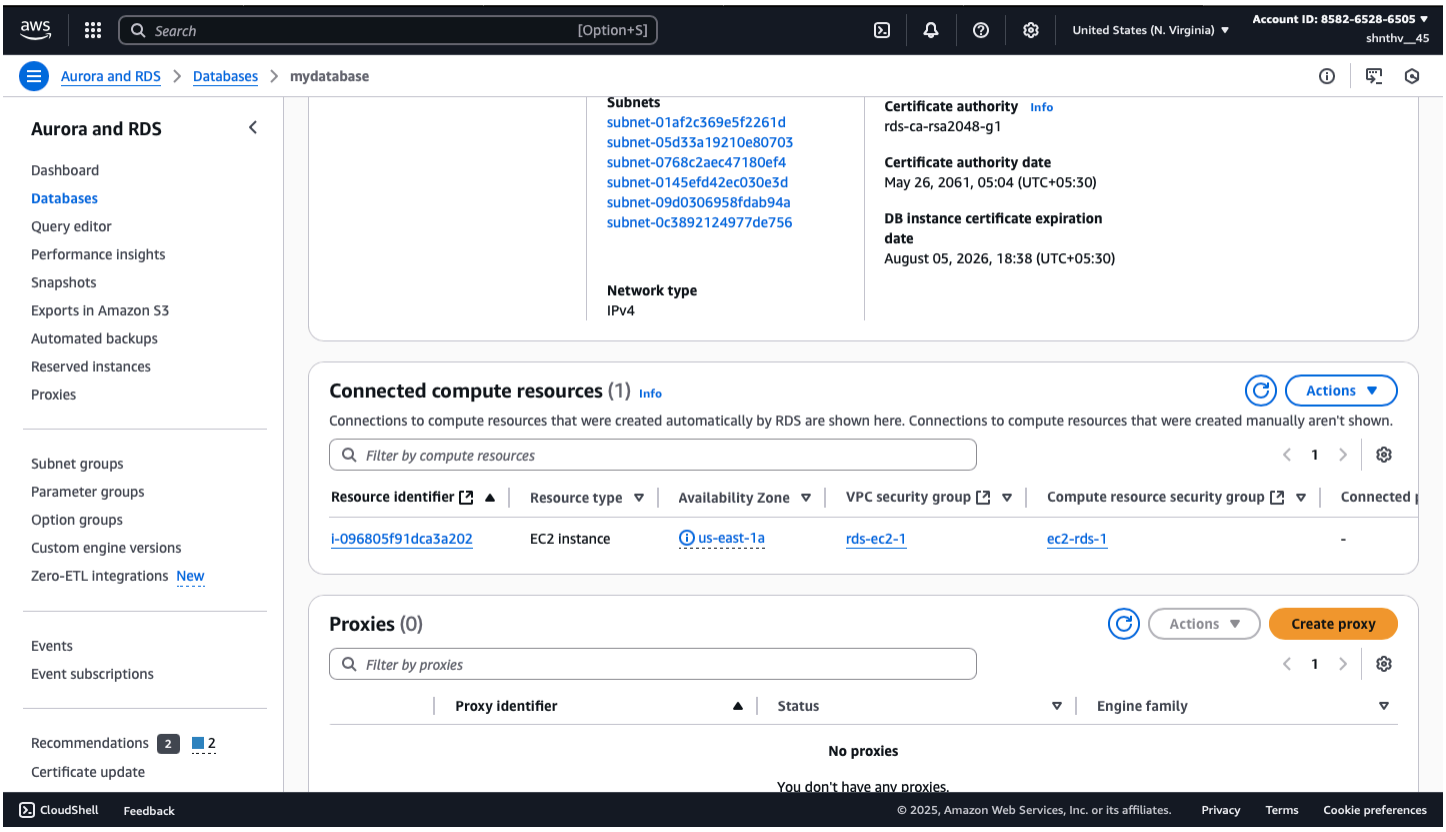
sudo dnf update -y

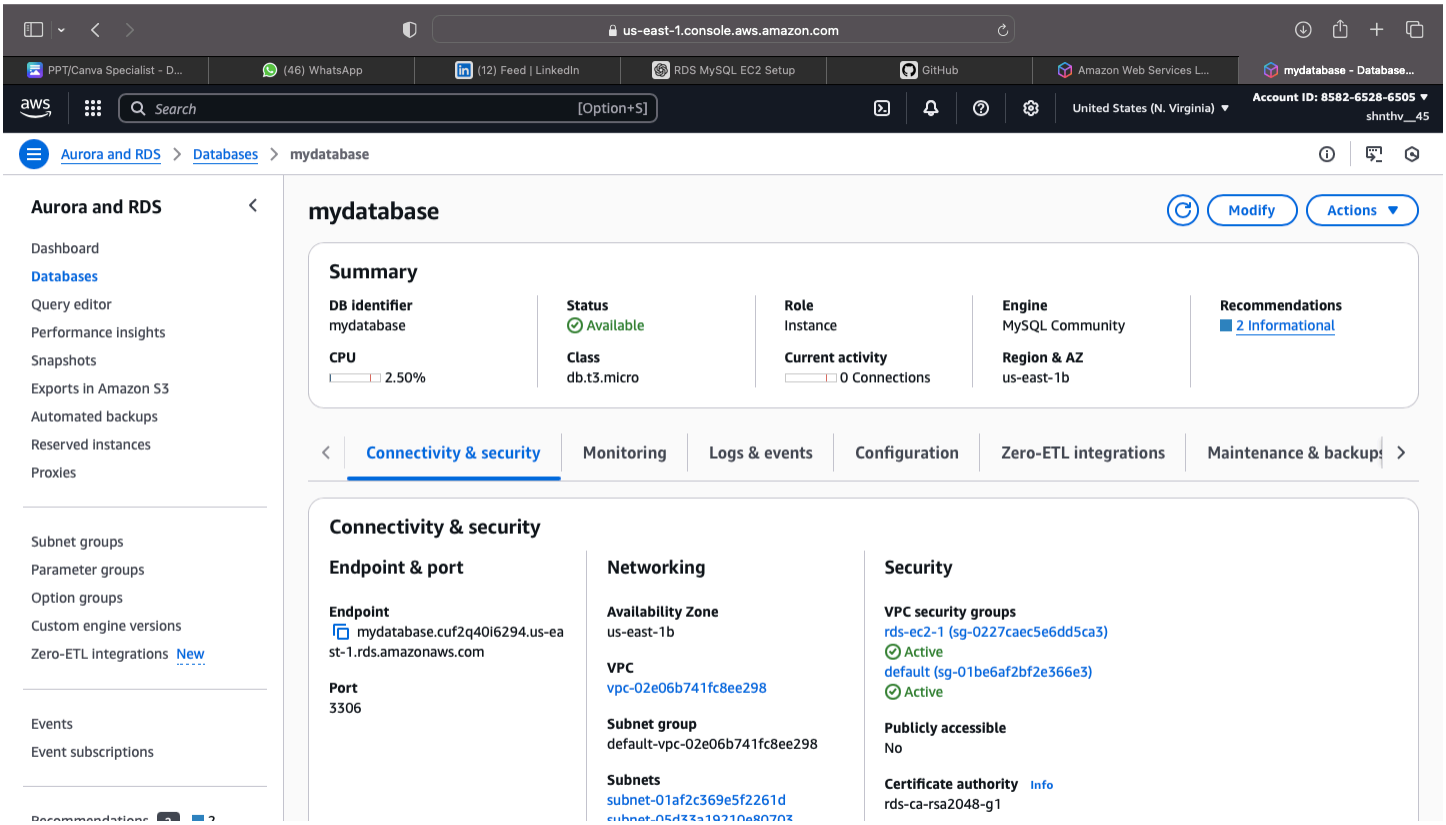
sudo dnf install -y mariadb105

This installs the MySQL/MariaDB client, which lets you connect to RDS.

6. Connect from EC2 to RDS

1. Go to **RDS > Databases > your database (mydatabase)**.
2. Copy the **endpoint** (e.g.,mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com
3. From your EC2 terminal, connect:
4. mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u databasemysql -p
5. Enter your DB password when prompted.
6. You are now connected to your MySQL RDS database





7. Create a Database and Table

Inside the MySQL shell, run:

```
CREATE DATABASE company;
```

USE company;

CREATE TABLE users (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(50),

email VARCHAR(100)

);

8. Insert and Verify Data

Insert some sample records:

INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');

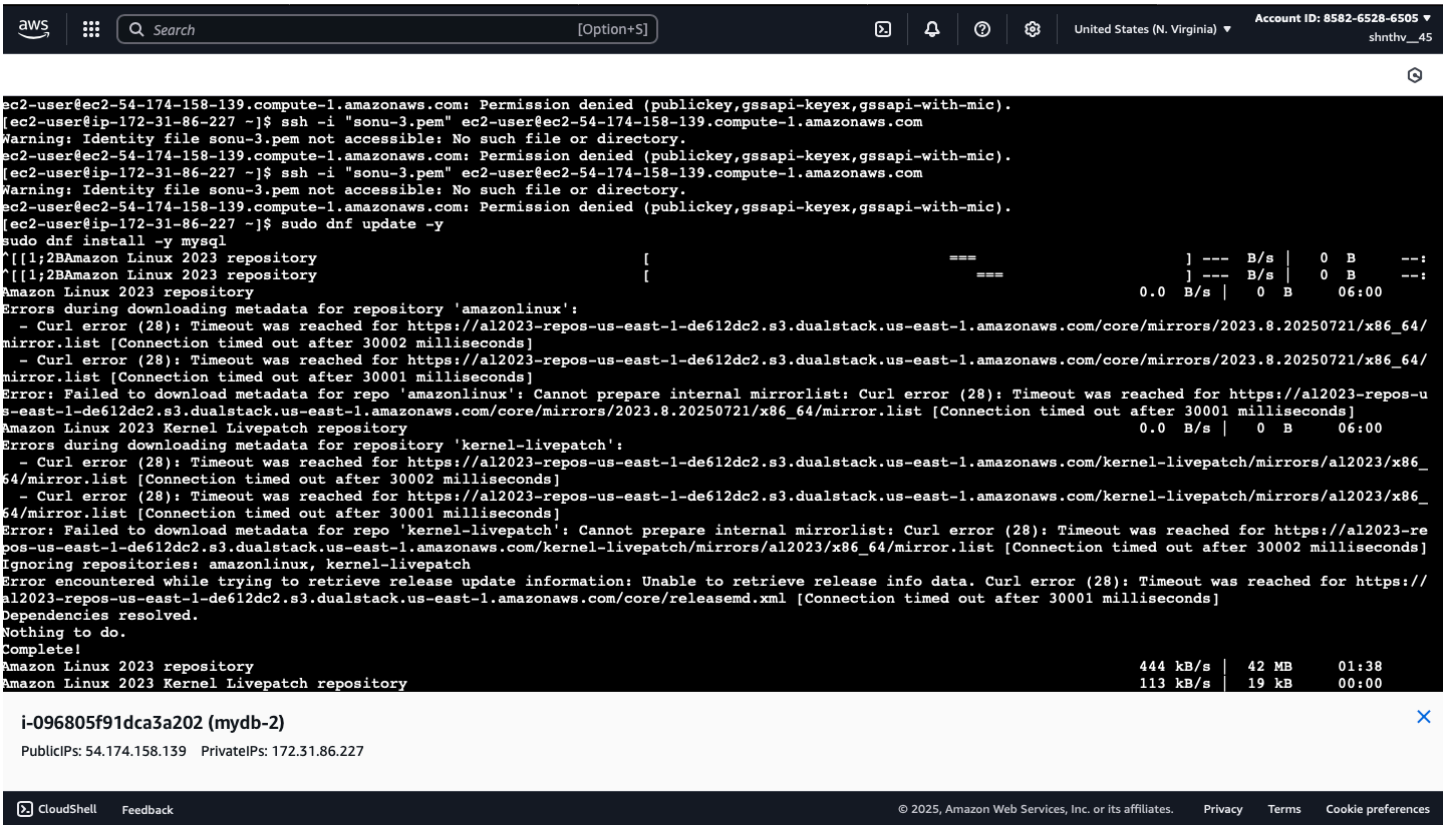
INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');

INSERT INTO users (name, email) VALUES ('Charlie', 'charlie@example.com');

INSERT INTO users (name, email) VALUES ('Diana', 'diana@example.com');

Check the data:

SELECT * FROM users;



```
Complete!
[ec2-user@ip-172-31-86-227 ~]$ sudo yum install -y mysql
Last metadata expiration check: 0:00:56 ago on Mon Aug 18 20:49:20 2025.
No match for argument: mysql
Error: Unable to find a match: mysql
[ec2-user@ip-172-31-86-227 ~]$ sudo dnf install -y mariadb105
Last metadata expiration check: 0:01:34 ago on Mon Aug 18 20:49:20 2025.
Dependencies resolved.
================================================================================
 Package                    Architecture  Version                    Repository  Size
================================================================================
Installing:
 mariadb105                 x86_64        3:10.5.29-1.amzn2023.0.1   amazonlinux 1.5 M
Installing dependencies:
 mariadb-connector-c        x86_64        3.3.10-1.amzn2023.0.1      amazonlinux 211 k
 mariadb-connector-c-config noarch        3.3.10-1.amzn2023.0.1      amazonlinux 9.9 k
 mariadb105-common          x86_64        3:10.5.29-1.amzn2023.0.1   amazonlinux 28 k
 perl-Sys-Hostname          x86_64        1.23-477.amzn2023.0.7      amazonlinux 16 k

Transaction Summary
================================================================================
Install  5 Packages

Total download size: 1.8 M
Installed size: 19 M
Downloading Packages:
(1/5): mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch.rpm  204 kB/s | 9.9 kB  00:00
(2/5): mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64.rpm         3.5 MB/s | 211 kB  00:00
(3/5): mariadb105-10.5.29-1.amzn2023.0.1.x86_64.rpm                  18 MB/s | 1.5 MB  00:00
(4/5): mariadb105-common-10.5.29-1.amzn2023.0.1.x86_64.rpm          724 kB/s |  28 kB  00:00
(5/5): perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64.rpm           347 kB/s |  16 kB  00:00
--------------------------------------------------------------------------------
Total                                                                13 MB/s | 1.8 MB  00:00
Running transaction check
Transaction check succeeded.
```

i-096805f91dca3a202 (mydb-2)

PublicIPs: 54.174.158.139 PrivateIPs: 172.31.86.227

aws Search [Option+S] United States (N. Virginia) Account ID: 8582-6528-6505 shnthv__45

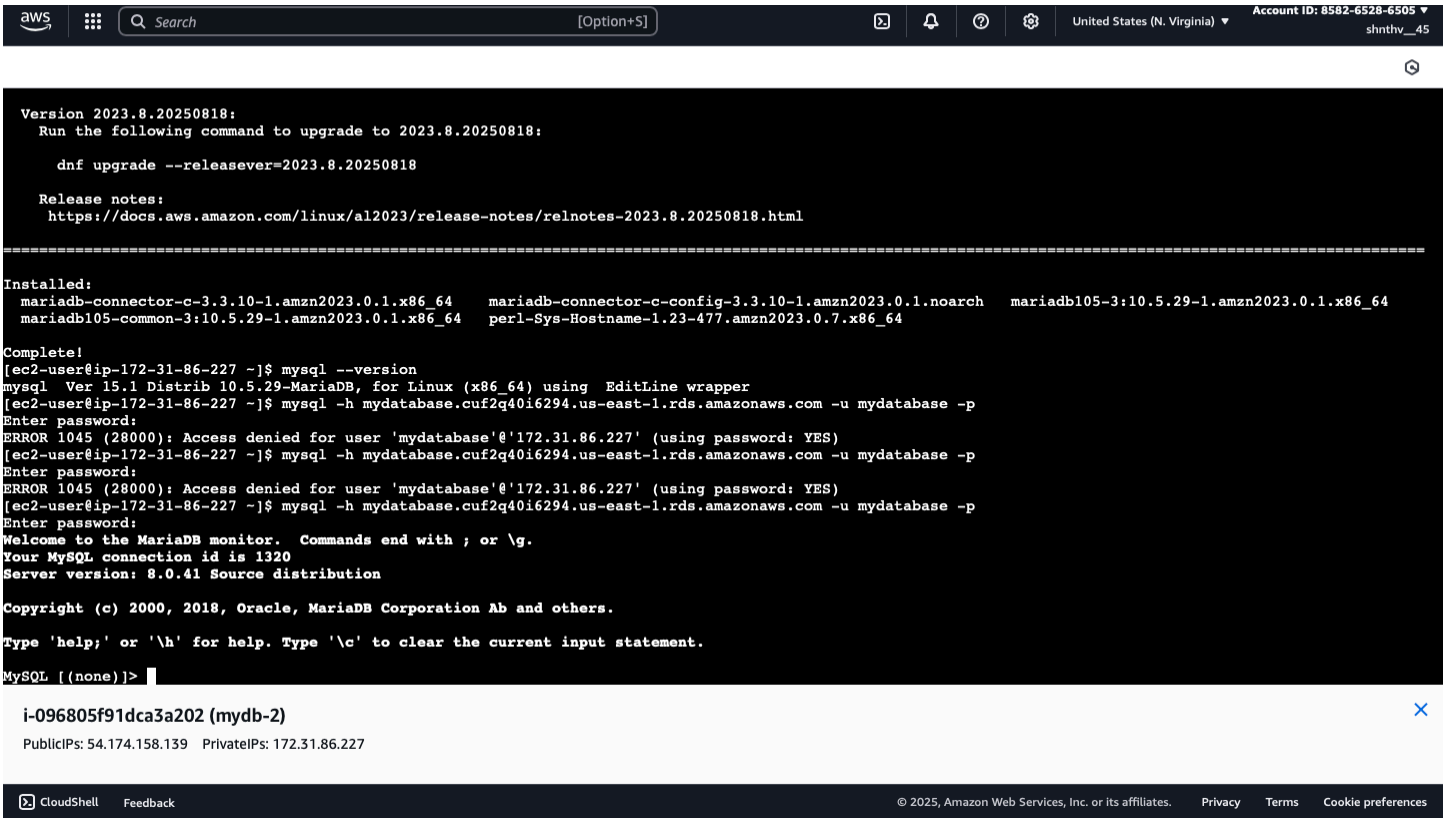
```
  Verifying        : mariadb105-common-3:10.5.29-1.amzn2023.0.1.x86_64          4/5
  Verifying        : perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64             5/5
================================================================================
WARNING:
  A newer release of "Amazon Linux" is available.

  Available Versions:

  Version 2.023.8.20250808:
    Run the following command to upgrade to 2023.8.20250808:

      dnf upgrade --releasever=2023.8.20250808

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250808.html

  Version 2.023.8.20250818:
    Run the following command to upgrade to 2023.8.20250818:

      dnf upgrade --releasever=2023.8.20250818

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250818.html

================================================================================

Installed:
  mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64    mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch   mariadb105-3:10.5.29-1.amzn2023.0.1.x86_64
  mariadb105-common-3:10.5.29-1.amzn2023.0.1.x86_64   perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64

Complete!
[ec2-user@ip-172-31-86-227 ~]$ mysql --version
mysql  Ver 15.1 Distrib 10.5.29-MariaDB, for Linux (x86_64) using  EditLine wrapper
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p
Enter password:
```

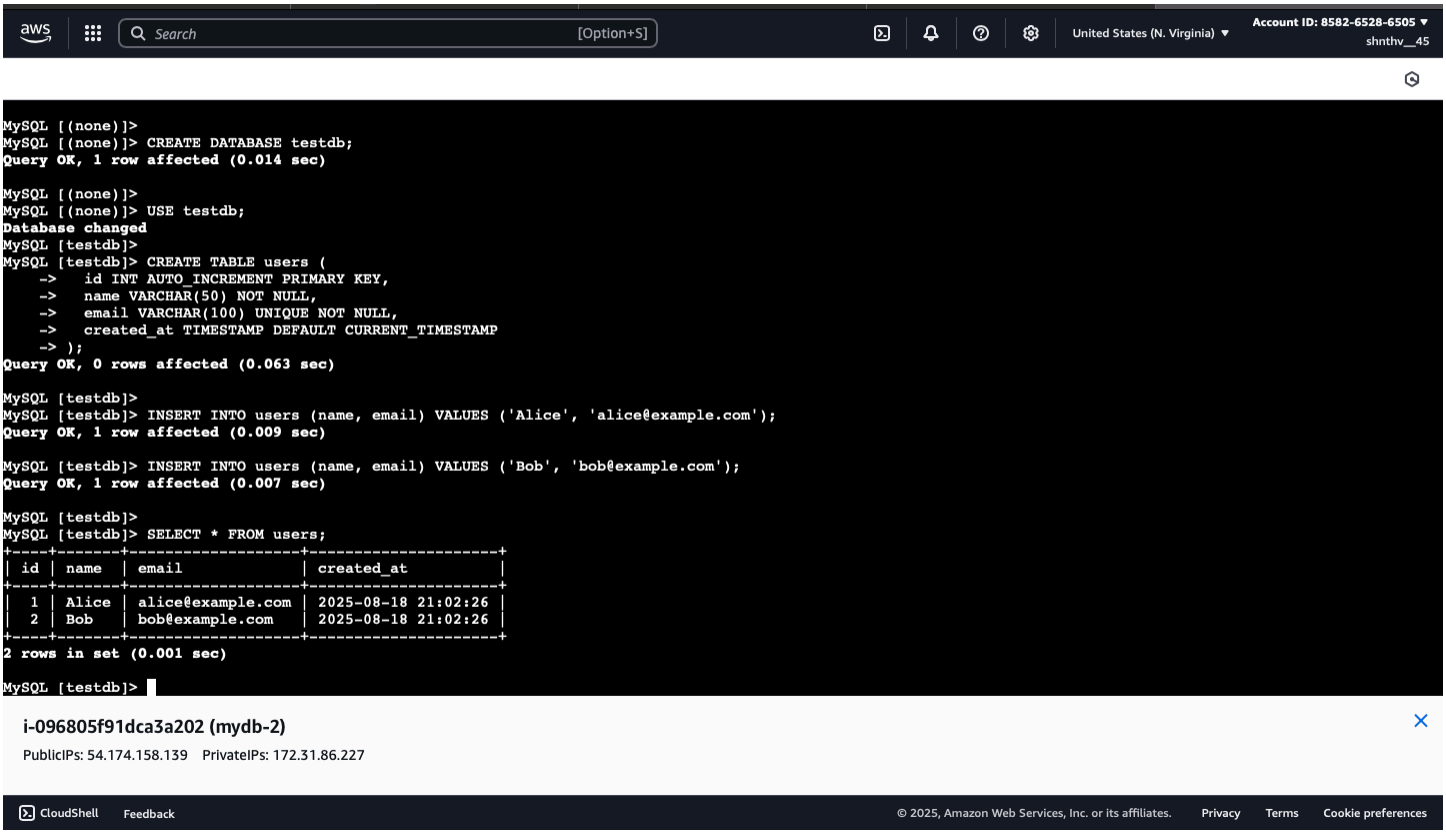
i-096805f91dca3a202 (mydb-2)

PublicIPs: 54.174.158.139 PrivateIPs: 172.31.86.227

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Search [Option+S]
United States (N. Virginia)
Account ID: 8582-6528-6505
shnthv__45
Version 2023.8.20250818:
Run the following command to upgrade to 2023.8.20250818:
dnf upgrade --releasever=2023.8.20250818
Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.8.20250818.html
Installed:
mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64 mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch mariadb105-3:10.5.29-1.amzn2023.0.1.x86_64
mariadb105-common-3:10.5.29-1.amzn2023.0.1.x86_64 perl-Sys-Hostname-1.23-477.amzn2023.0.7.x86_64
Complete!
[ec2-user@ip-172-31-86-227 ~]$ mysql --version
mysql Ver 15.1 Distrib 10.5.29-MariaDB, for Linux (x86_64) using EditLine wrapper
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p
Enter password:
ERROR 1045 (28000): Access denied for user 'mydatabase'@'172.31.86.227' (using password: YES)
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p
Enter password:
ERROR 1045 (28000): Access denied for user 'mydatabase'@'172.31.86.227' (using password: YES)
[ec2-user@ip-172-31-86-227 ~]$ mysql -h mydatabase.cuf2q40i6294.us-east-1.rds.amazonaws.com -u mydatabase -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 1320
Server version: 8.0.41 Source distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MySQL [(none)]>
i-096805f91dca3a202 (mydb-2)
PublicIPs: 54.174.158.139 PrivateIPs: 172.31.86.227
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Search [Option+S]
United States (N. Virginia)
Account ID: 8582-6528-6505
shnthv__45
MySQL [(none)]>
MySQL [(none)]> CREATE DATABASE testdb;
Query OK, 1 row affected (0.014 sec)
MySQL [(none)]>
MySQL [(none)]> USE testdb;
Database changed
MySQL [testdb]>
MySQL [testdb]> CREATE TABLE users (
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> name VARCHAR(50) NOT NULL,
-> email VARCHAR(100) UNIQUE NOT NULL,
-> created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
->);
Query OK, 0 rows affected (0.063 sec)
MySQL [testdb]>
MySQL [testdb]> INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
Query OK, 1 row affected (0.009 sec)
MySQL [testdb]> INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
Query OK, 1 row affected (0.007 sec)
MySQL [testdb]>
MySQL [testdb]> SELECT * FROM users;
+----+-------+-----------------+---------------------+
| id | name | email | created_at |
+----+-------+-----------------+---------------------+
| 1 | Alice | alice@example.com | 2025-08-18 21:02:26 |
| 2 | Bob | bob@example.com | 2025-08-18 21:02:26 |
+----+-------+-----------------+---------------------+
2 rows in set (0.001 sec)
MySQL [testdb]>
i-096805f91dca3a202 (mydb-2)
PublicIPs: 54.174.158.139 PrivateIPs: 172.31.86.227
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```
MySQL [testdb]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testdb             |
+--------------------+
5 rows in set (0.001 sec)

MySQL [testdb]> CREATE DATABASE projectdb;
Query OK, 1 row affected (0.006 sec)

MySQL [testdb]> USE projectdb;
Database changed
MySQL [projectdb]> CREATE TABLE users (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   name VARCHAR(50) NOT NULL,
    ->   email VARCHAR(100) UNIQUE NOT NULL,
    ->   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.038 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Alice', 'alice@example.com');
Query OK, 1 row affected (0.005 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Bob', 'bob@example.com');
Query OK, 1 row affected (0.004 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Charlie', 'charlie@example.com');
Query OK, 1 row affected (0.005 sec)

MySQL [projectdb]> INSERT INTO users (name, email) VALUES ('Diana', 'diana@example.com');
```

i-096805f91dca3a202 (mydb-2)

PublicIPs: 54.174.158.139 PrivateIPs: 172.31.86.227

2.Configure Auto Scaling and Load Balancing Create a launch template for EC2 instances. Set up an auto-scaling group with minimum and maximum instance limits. Attach an application load balancer to distribute traffic.

Auto Scaling Group + Load Balancer on AWS

This guide explains **from scratch** how to set up an Auto Scaling Group (ASG) with a Load Balancer (ALB) to run a simple website on EC2 instances.

We'll do this in 5 main parts:

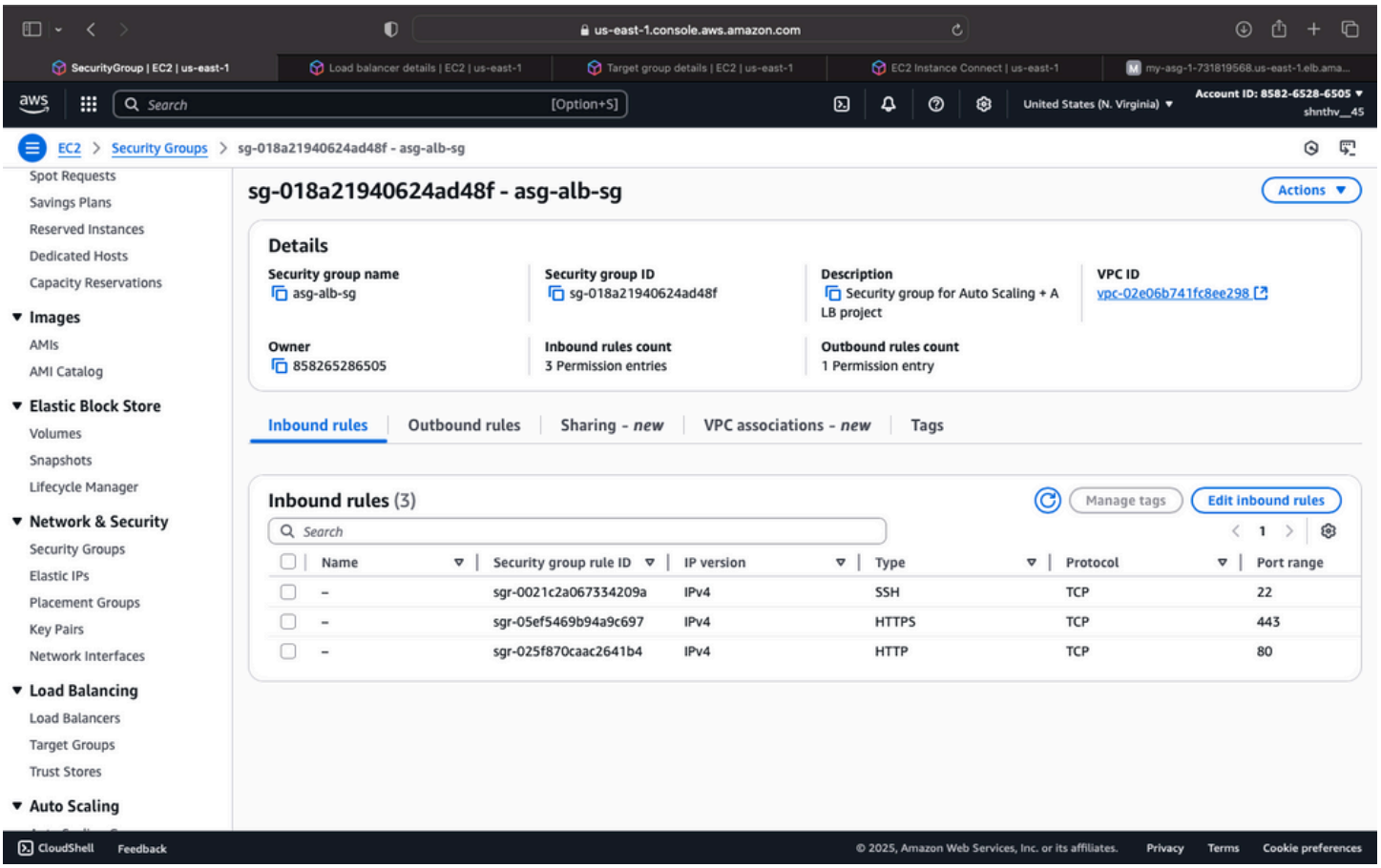
1. **Launch Template** → Blueprint for EC2.
2. **Security Group** → Firewall for EC2.
3. **Auto Scaling Group (ASG)** → Manages EC2 automatically.
4. **Application Load Balancer (ALB)** → Distributes traffic.
5. **Install Apache Web Server** → Host a webpage.

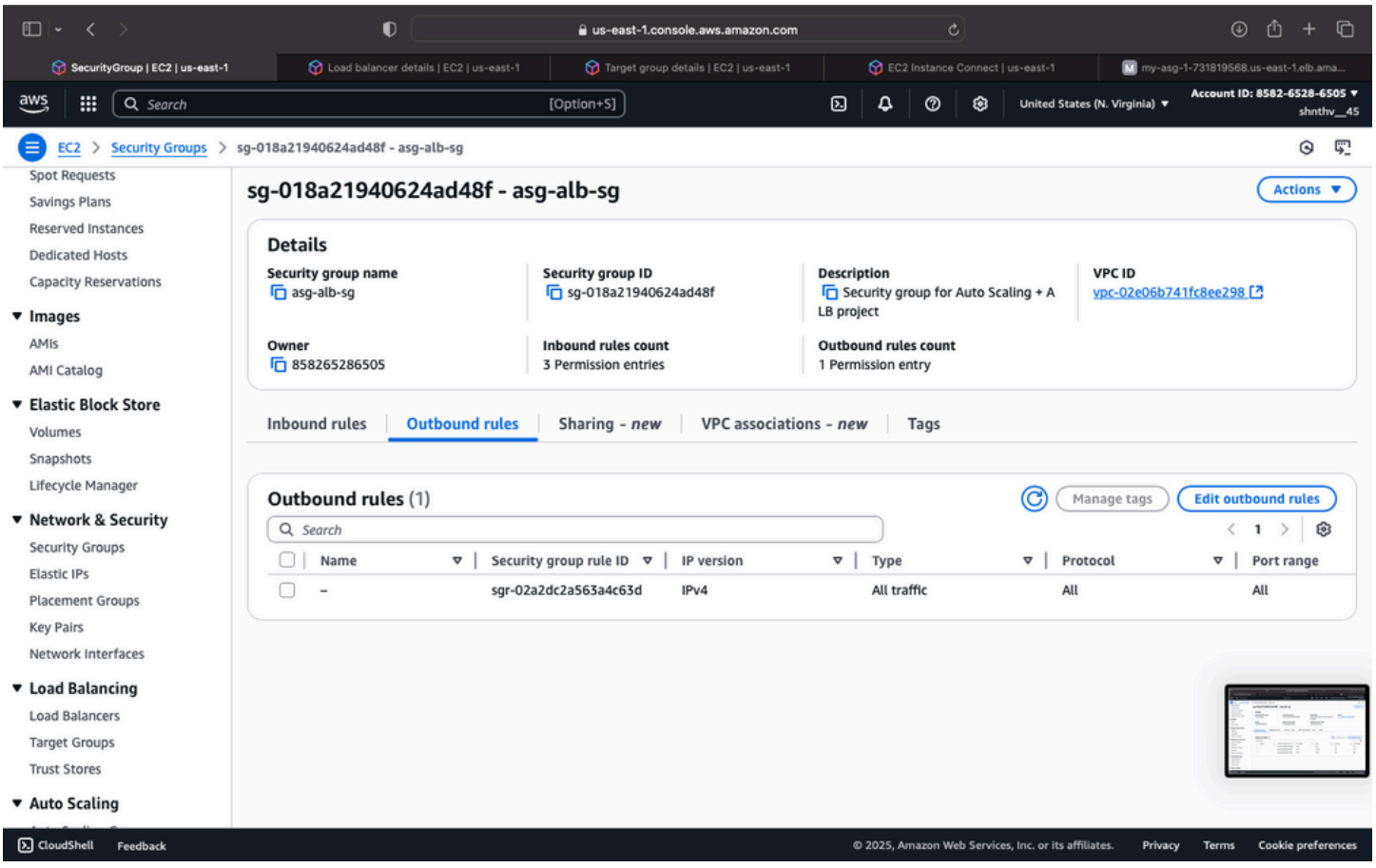
Step 1: Create a Security Group

A **Security Group** acts like a firewall. It controls what traffic can enter and leave your EC2 instance.

1. Go to **EC2 → Security Groups → Create Security Group**.
2. Fill details:
   - **Name:** asg-alb-sg
   - **Description:** Security Group for ASG + ALB
   - **VPC:** Default VPC
3. Under **Inbound Rules**, add:
   - **SSH (22)** → Source: My IP (to connect to instance)
   - **HTTP (80)** → Source: Anywhere (0.0.0.0/0)
4. Leave **Outbound Rules** as default (all traffic allowed).
5. Click **Create Security Group**

This allows us to connect via SSH and serve a website.

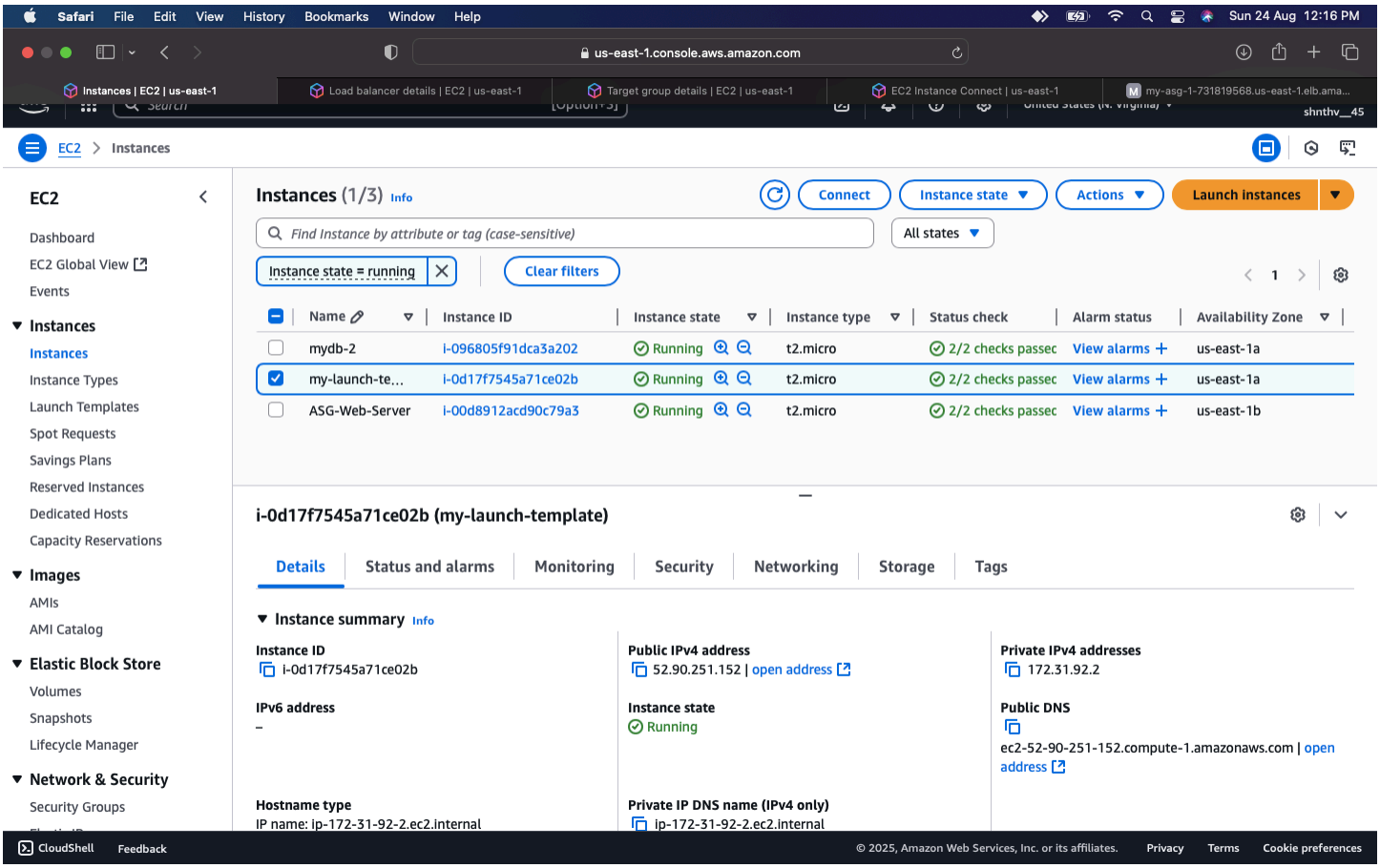




Step 2: Create a Launch Template

A **Launch Template** is a blueprint. It tells AWS how new EC2 instances should be created (OS, size, key, firewall).

1. Go to **EC2 → Launch Templates → Create Launch Template**.
2. Fill details:
   - **Name:** my-launch-template
   - **AMI (OS):** Amazon Linux 2023 (free tier eligible)
   - **Instance type:** t2.micro
   - **Key pair:** Choose your .pem key (example: sonu-3.pem)
   - **Security Group:** Select asg-alb-sg (created above)
   - **Storage:** 8GB (default EBS volume)
3. Click **Create Launch Template**



Now AWS knows how to launch new EC2 instances automatically.

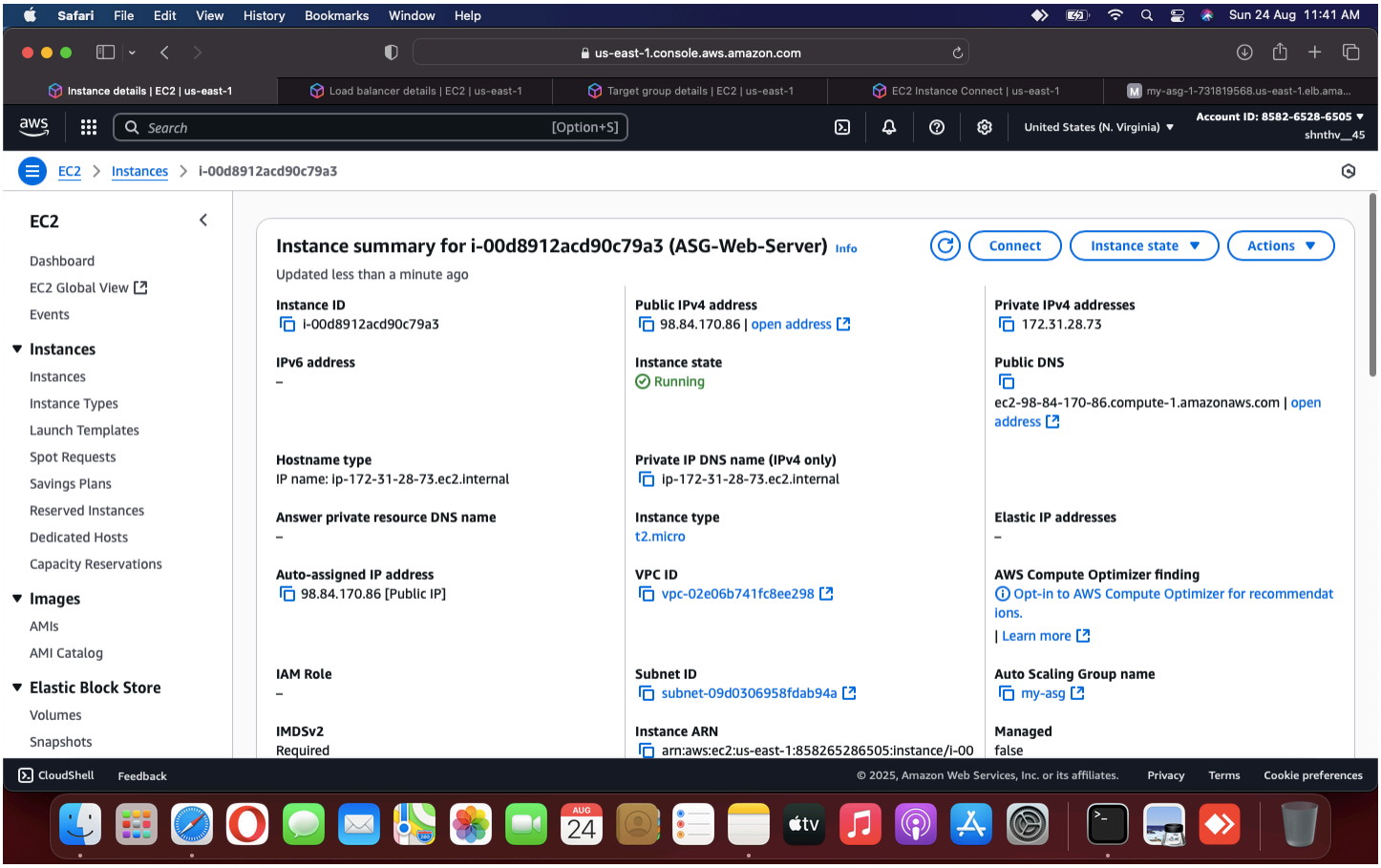
Step 3: Create an Auto Scaling Group (ASG)

An **Auto Scaling Group** ensures the right number of EC2 instances are always running. If one crashes, it creates a new one. If load increases, it adds more.

1. Go to **EC2 → Auto Scaling Groups → Create Auto Scaling Group**.
2. Select:
   - **Launch Template:** my-launch-template
   - **Name:** my-asg
3. Select **VPC:** Default VPC
   - Choose **at least 2 subnets** in different Availability Zones (e.g. us-east-1a and us-east-1b) → This ensures high availability.
4. Under **Load Balancing**, select:
   - **Attach to an existing load balancer** → (We'll create ALB next)

- Or choose **Create new ALB** directly here.

5. Configure **Group Size**:
   - **Desired capacity:** 1 (start with one instance)
   - **Min:** 1
   - **Max:** 3 (allows scaling up to 3 instances)
6. Add **Scaling Policy**:
   - **Target tracking policy** → Keep average CPU utilization at **50%**
7. Notifications (Optional):
   - Choose or create **SNS topic** → Get emails when instances launch/terminate.
8. Add **Tags**:
   - Example → Key: Name | Value: ASG-Web-Server



Click **Create Auto Scaling Group**

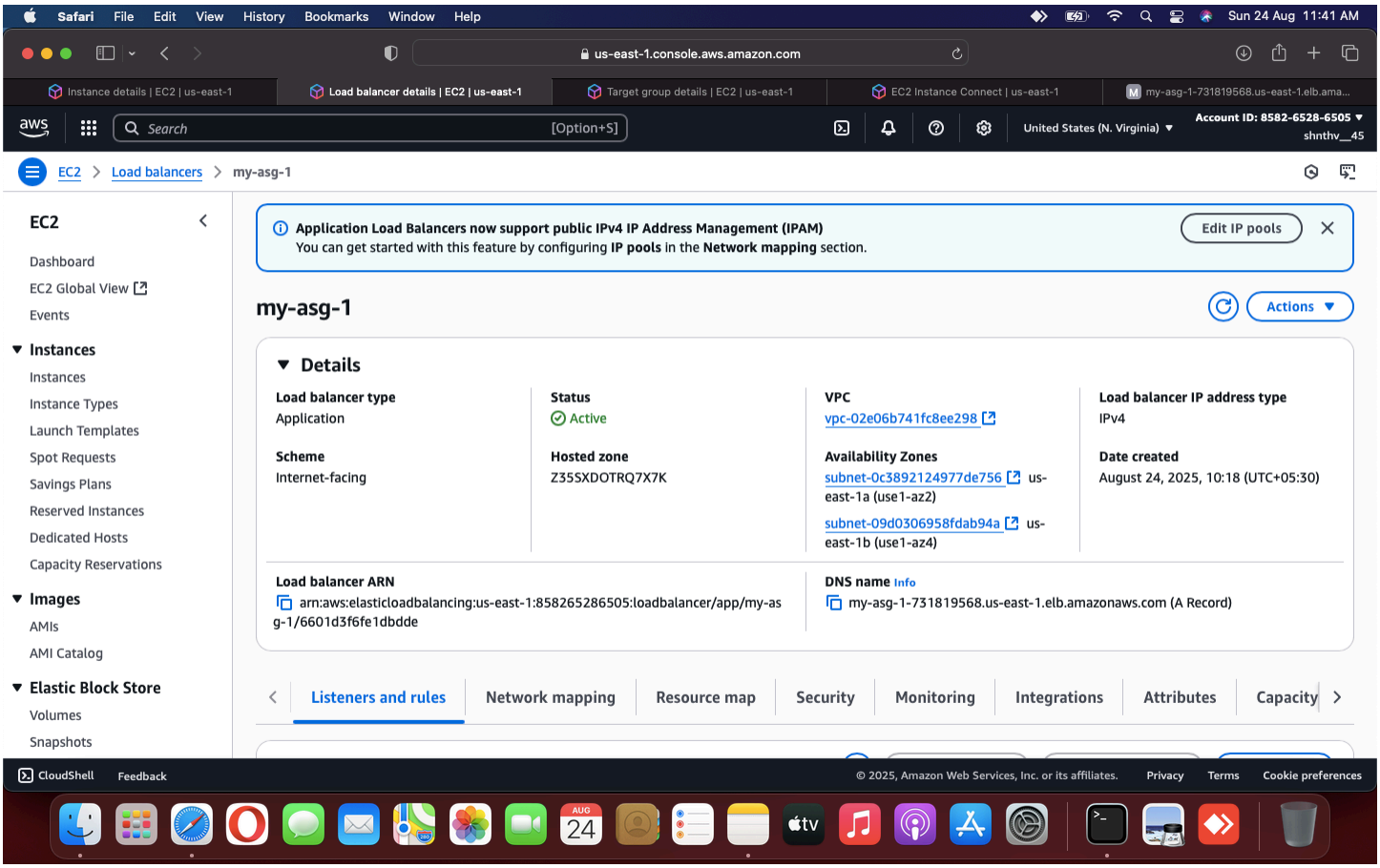
Now AWS will launch your first EC2 instance automatically.

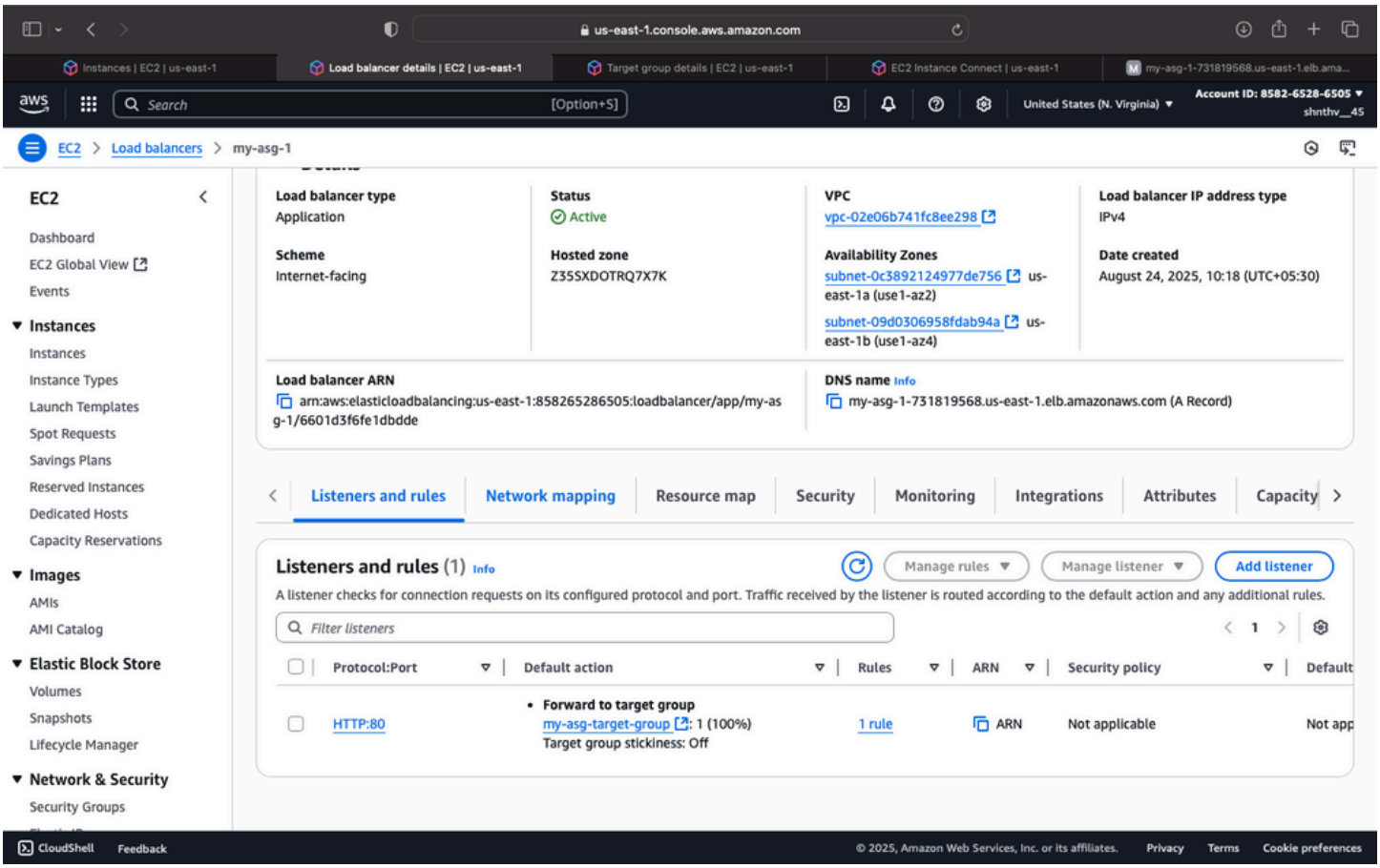
Step 4: Create an Application Load Balancer (ALB)

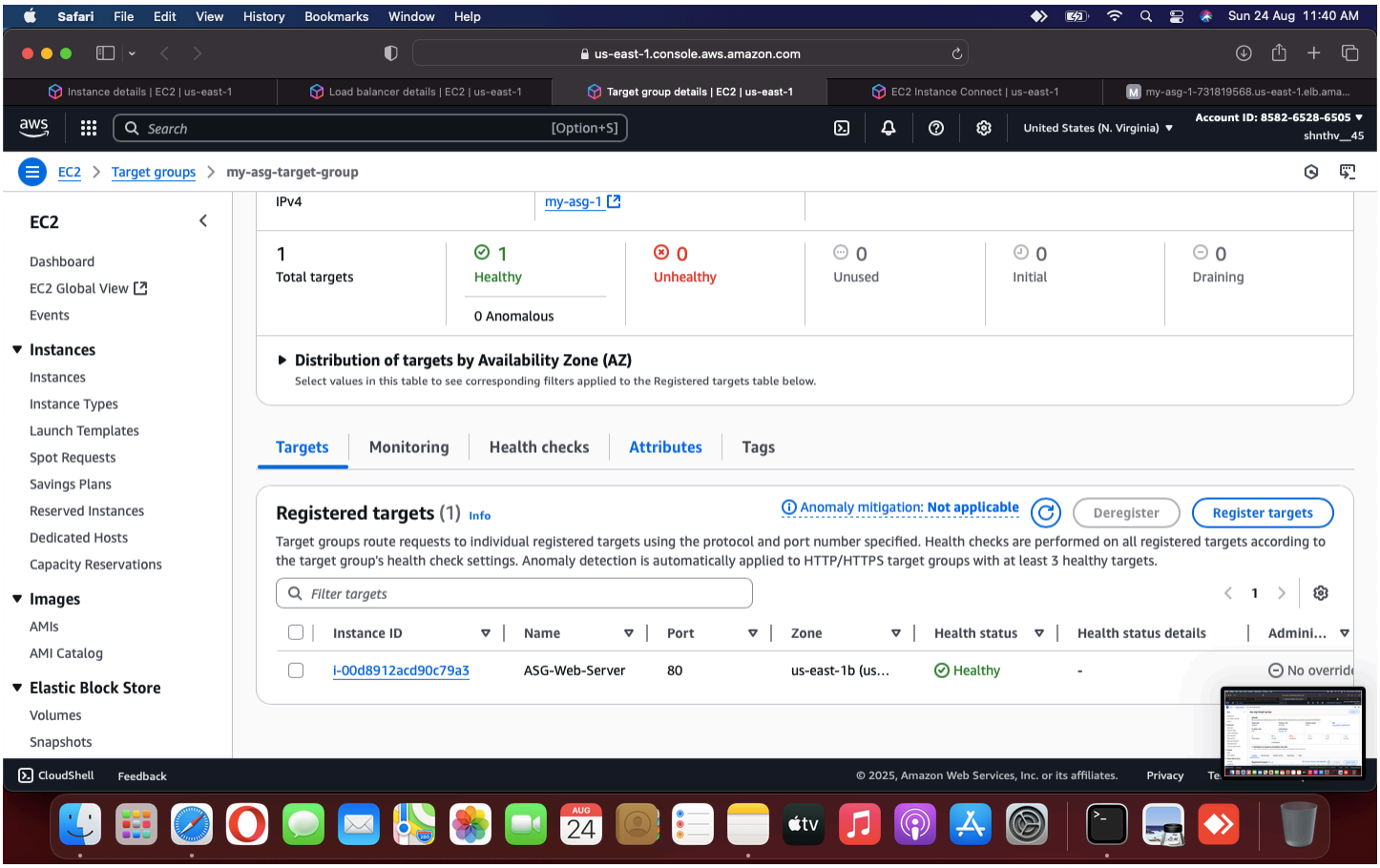
A **Load Balancer** makes sure user traffic is distributed across healthy instances. If one fails, traffic goes to another.

1. Go to **EC2 → Load Balancers → Create Load Balancer → Application Load Balancer**.
2. Fill details:
   - **Name:** my-asg-alb
   - **Scheme:** Internet-facing
   - **IP type:** IPv4

3. Network:
    - Choose **VPC:** Default
    - Select **2 subnets** in different Availability Zones (same as ASG).
4. Security Group:
    - Select **asg-alb-sg** (so it allows HTTP).
5. Listener:
    - Add **HTTP on port 80**.
6. Create **Target Group**:
    - **Name:** my-asg-target-group
    - **Target type:** Instances
    - **Protocol:** HTTP:80
    - **Health Check Path:** / (root page)
7. Register instances → Select EC2 instances from your ASG.
8. Click **Create Load Balancer**







Now ALB sits in front of your ASG and distributes traffic.

Step 5: Install Apache on EC2 Instance

Now we'll install a **web server** (Apache) to serve a webpage.

1. Go to **EC2 → Instances → Select instance → Connect**
   (or SSH from terminal: ssh -i sonu-3.pem ec2-user@<Public-IP>).
2. Run these commands one by one:

Update all packages

sudo yum update -y

Install Apache (httpd)

sudo yum install -y httpd

Start Apache

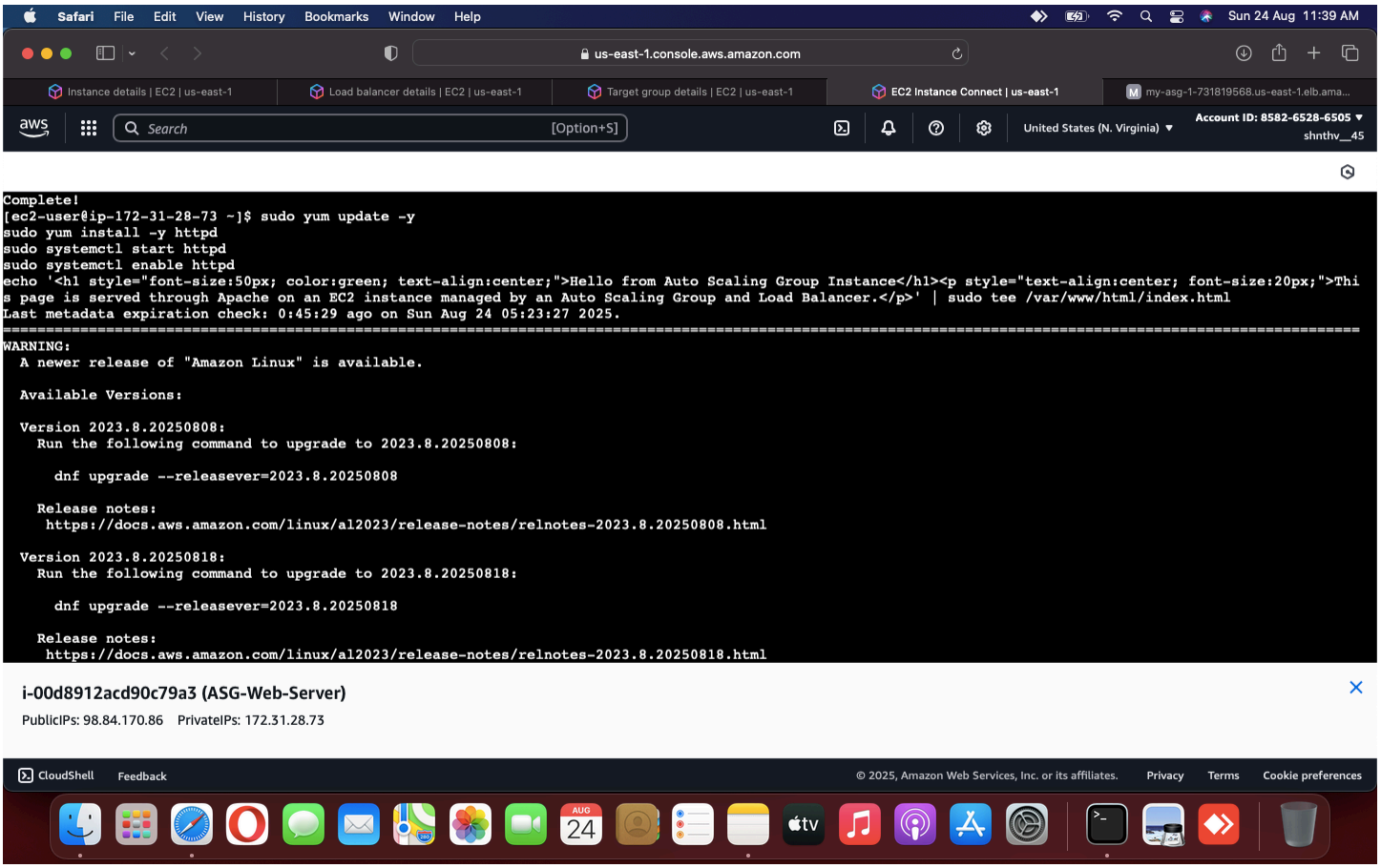
sudo systemctl start httpd

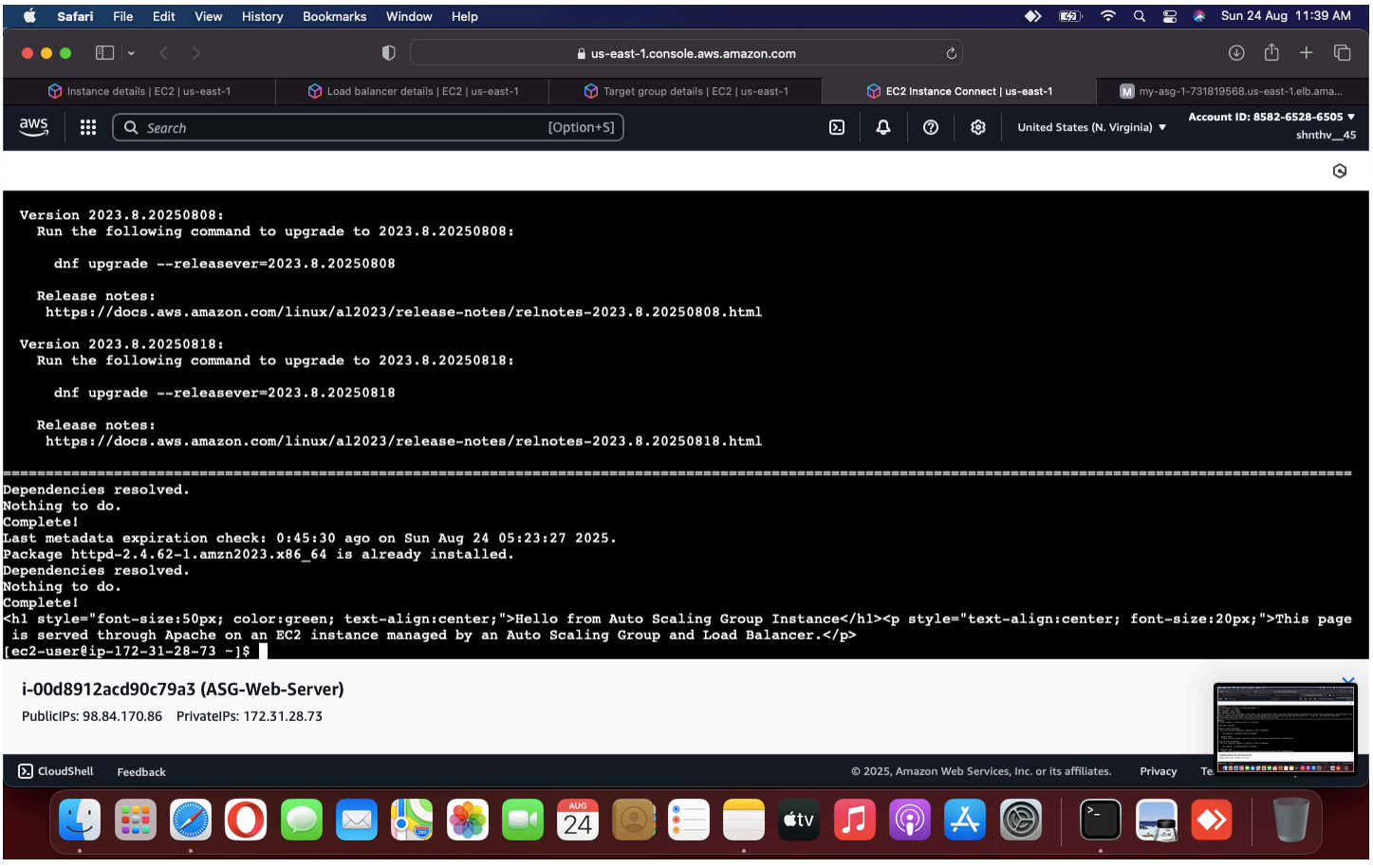
sudo systemctl enable httpd

Create a custom HTML page

echo '<h1 style="font-size:50px; color:green; text-align:center;">Hello from Auto Scaling Group Instance</h1>

This page is served through Apache on an EC2 instance managed by an Auto Scaling Group and Load Balancer. ' | sudo tee /var/www/html/index.html





Now your EC2 is serving a webpage.

Step 6: Test Your Setup

1. Copy the **DNS of your Load Balancer** (looks like my-asg-1-731819568.us-east-1.elb.amazonaws.com).
2. Open it in your browser.
3. You should see:

"**Hello from Auto Scaling Group Instance**" in **big green letters**.
If you refresh multiple times, ALB may send requests to different EC2 instances.
If one instance fails, ASG will replace it automatically.