

# SYSTEM MODELING

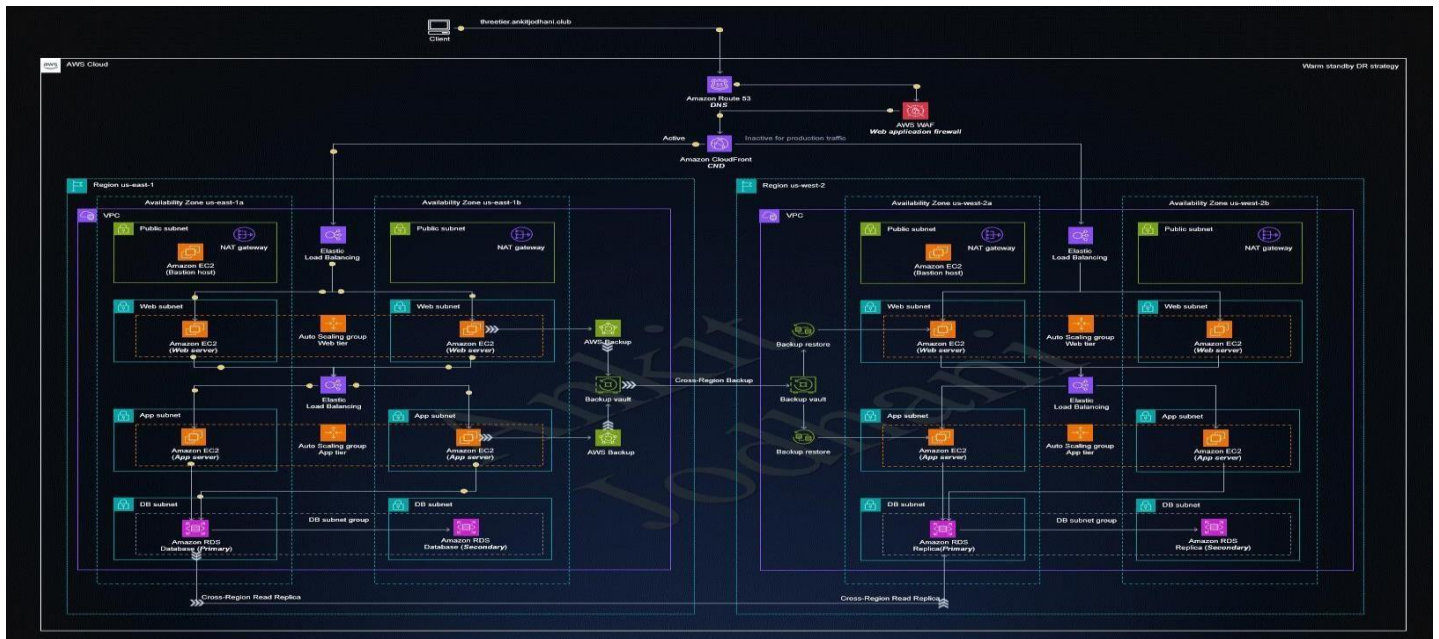


Figure 3.1 System Architecture

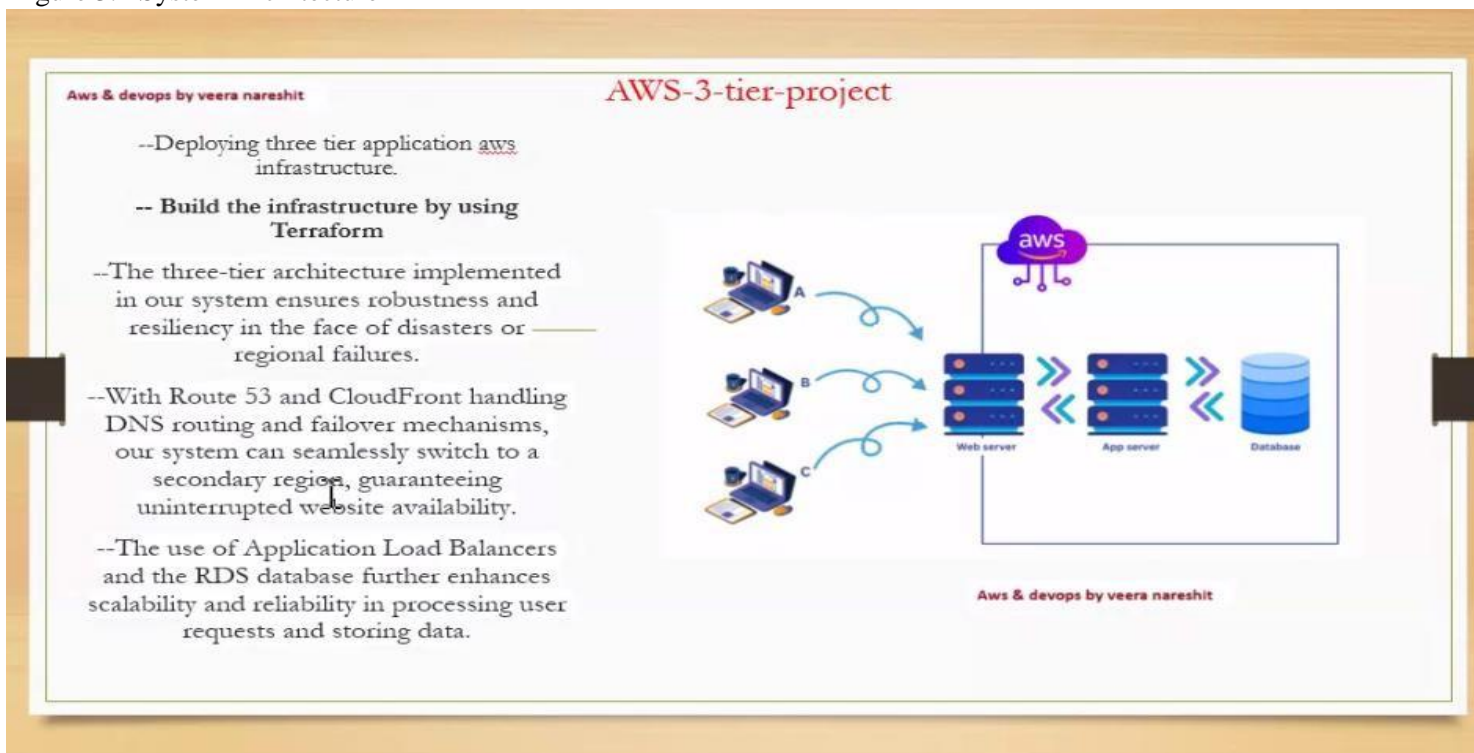


Figure 3.2

## 3.2 Software Requirements

- **Amazon Machine Image (AMI):** A pre-configured template that contains the operating system, application server, and applications required to start an instance. Popular choices include Amazon Linux 2 and Ubuntu2.
- **Web Server Software:** This is necessary to serve your application to visitors. Common options include Apache, Nginx, and Microsoft IIS3.
- **SSH Client:** To securely connect to your EC2 instance, you'll need an SSH client like PuTTY (for Windows) or the terminal (for macOS/Linux).
- **SSL/TLS Certificate:** To enable HTTPS and secure your application, you'll need an SSL certificate. Let's Encrypt is a popular choice for free SSL certificates4.
- **Domain Name:** A registered domain name that you can point to your EC2 instance. Services like GoDaddy or AWS Route 53 can help with this2.
- **Version Control System (Optional):** If you're using Git for version control, you'll need Git Bash or another Git client.
- **Database Software (Optional):** If your application requires a database, you might need software like MySQL, PostgreSQL, or MongoDB Steps:

### 3.3 Software Description

- **Amazon Machine Image (AMI):**

**Description:** An AMI is a template that contains the software configuration (operating system, application server, applications) required to launch an instance. Popular choices include Amazon Linux 2, Ubuntu, and other pre-configured images.

**Purpose:** To provide a pre-packaged environment that you can quickly and easily deploy on EC2, saving time on initial setup.

- **Web Server Software:**

**Description:** Software like Apache, Nginx, or Microsoft IIS that handles the HTTP requests and serves your Application to visitors.

**Purpose:** To serve your Application's content to users. Apache and Nginx are particularly popular for their performance and flexibility.

- **SSH Client:**

**Description:** SSH (Secure Shell) clients like PuTTY (for Windows) or the terminal (for macOS/Linux) allow you to securely connect to your EC2 instance.

**Purpose:** To securely access your virtual server for configuration and management tasks.

- **SSL/TLS Certificate:**

**Description:** SSL certificates, such as those from Let's Encrypt, encrypt data between the user's browser and your server, enabling HTTPS.

**Purpose:** To provide secure, encrypted communication between your Application and its visitors, ensuring data privacy and integrity.

- **Domain Name:**

**Description:** A domain name is the address where users can access your Application (e.g., www.example.com). You can register a domain through services like GoDaddy or AWS Route 53.

**Purpose:** To give your Application a memorable and accessible address on the internet.

- **Version Control System (Optional):**

**Description:** Tools like Git help you manage changes to your Application's code, track revisions, and collaborate with others.

**Purpose:** To maintain version control over your Application's codebase, making it easier to track changes, collaborate, and roll back if necessary.

- **Database Software (Optional):**

**Description:** Database management systems like MySQL, PostgreSQL, or MongoDB store and manage your Application's data.

**Purpose:** To provide a reliable way to store and retrieve data for dynamic Applications that require backend data management

### **3.4 Implementation process**

### **1. Setting Up an AWS Account**

First, you need to create an account on Amazon Web Services (AWS). Once the account is ready, you can use the AWS Management Console to access various services. This is the starting point for launching your virtual server (EC2 instance).

### **2. Launching an EC2 Instance**

An EC2 instance is like a virtual computer where your Application will run. To launch one, you select an operating system and configuration using an Amazon Machine Image (AMI). You also choose the size of your instance based on the performance your Application needs—small instances for lightweight Applications and larger ones for hightraffic sites.

### **3. Configuring Security**

To protect your Application, you need to set up security rules, known as Security Groups. These rules allow specific types of traffic to reach your server, like HTTP or HTTPS for Application visitors and SSH for remote management.

### **4. Connecting to the Server**

Once your EC2 instance is running, you connect to it remotely using SSH. This allows you to control and configure the server from your local machine. You'll use a private key file for secure access.

### **5. Installing the Web Server**

After gaining access, you install a web server, such as Apache or Nginx, which is software that delivers your Application to users. You also install any other software your Application needs, like PHP, Python, or Node.js.

### **6. Uploading Your Application Files**

The next step is to upload your Application's files to the server. You can use tools like SCP (Secure Copy Protocol) or Git to transfer files securely. After uploading, you organize the files and ensure everything is set up correctly for your Application to work.

### **7. Setting Up a Database (If Needed)**

If your Application relies on a database, you configure it at this stage. You can use services like AWS RDS for a managed database solution, making it easier to handle large amounts of data.

## 8. Configuring the Domain Name

To make your Application accessible through a domain ,you map your domain name to the public IP address of your EC2 instance. This step is done using DNS services like AWS Route 53.

9. **Testing and Monitoring** Once everything is in place, you test your Application to ensure it's working as expected. AWS provides tools like CloudWatch to monitor your server's performance and alert you about any issues.

## 10. Scaling and Optimization

As your Application grows, you can scale your EC2 instance to handle more visitors. AWS makes it easy to add more computing power or distribute traffic across multiple servers using load balancers.

## 4. Result Analysis

The AWS Free Tier offers new users the opportunity to explore and experiment with Amazon Web Services at no cost, within specified usage limits. It comprises three types of offerings:

1. **12 Months Free:** Available to new AWS customers for 12 months from the sign-up date. Services under this category include:
  - **Amazon EC2:** 750 hours per month of Linux, RHEL, or SLES t2.micro or t3.micro instances, and 750 hours per month of Windows t2.micro or t3.micro instances.
  - **Amazon S3:** 5 GB of standard storage.
  - **Amazon RDS:** 750 hours per month of Single-AZ db.t2.micro or db.t3.micro instances, with options for MySQL, PostgreSQL, MariaDB, Oracle BYOL, or SQL Server.

2. **Always Free:** These offers are available to all AWS customers indefinitely. Examples include:
  - **AWS Lambda:** 1 million free requests and 400,000 GB-seconds of compute time per month.
  - **Amazon DynamoDB:** 25 GB of storage, 25 write capacity units, and 25 read capacity units.
3. **Free Trials:** Short-term trial offers that start from the first usage. For instance:
  - **Amazon CloudSearch:** Free for up to 30 days, including 750 hours of search instance usage and 10,000 document batch uploads.

## Amazon Web Services

It's crucial to monitor your usage to stay within the Free Tier limits. Exceeding these limits or using services not covered by the Free Tier will result in standard pay-as-you-go charges. To avoid unexpected costs, regularly check your AWS Billing and Cost Management Dashboard. For more detailed information.



Figure 4.1 Steps

:

Region Structure :1-ami-ec2, 2-us-west-2, 3-us-east-1

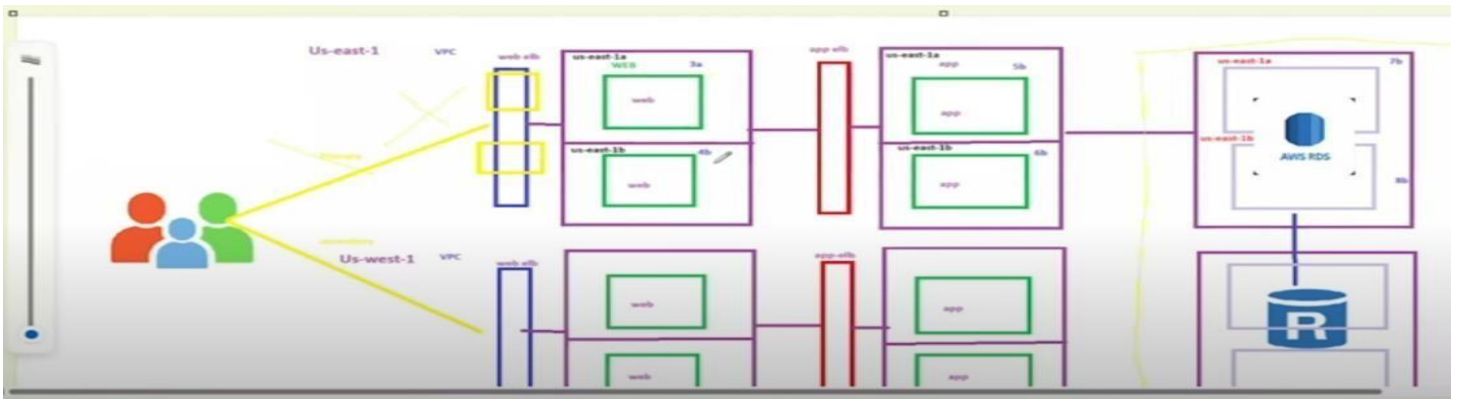


Figure 4.2

**SSH (Secure Shell)** is a cryptographic network protocol used to securely connect to a remote computer or server. It allows users to execute commands, manage systems, and transfer files over an encrypted connection.

### Key Features of SSH

1. **Secure Communication:** Encrypts all data transmitted between the client and server, protecting it from eavesdropping.
2. **Authentication:**
  - **Password-based:** Users log in using a username and password.
  - **Key-based:** Uses public-private key pairs for enhanced security.
3. **Port Forwarding:** Enables secure tunneling for accessing network services running on remote servers.
4. **File Transfer:**
  - **SCP (Secure Copy Protocol):** For transferring files securely.
  - **SFTP (SSH File Transfer Protocol):** For interactive file transfers.
5. **Remote Command Execution:** Allows users to run commands on a remote server as if they were logged in locally.

VPC



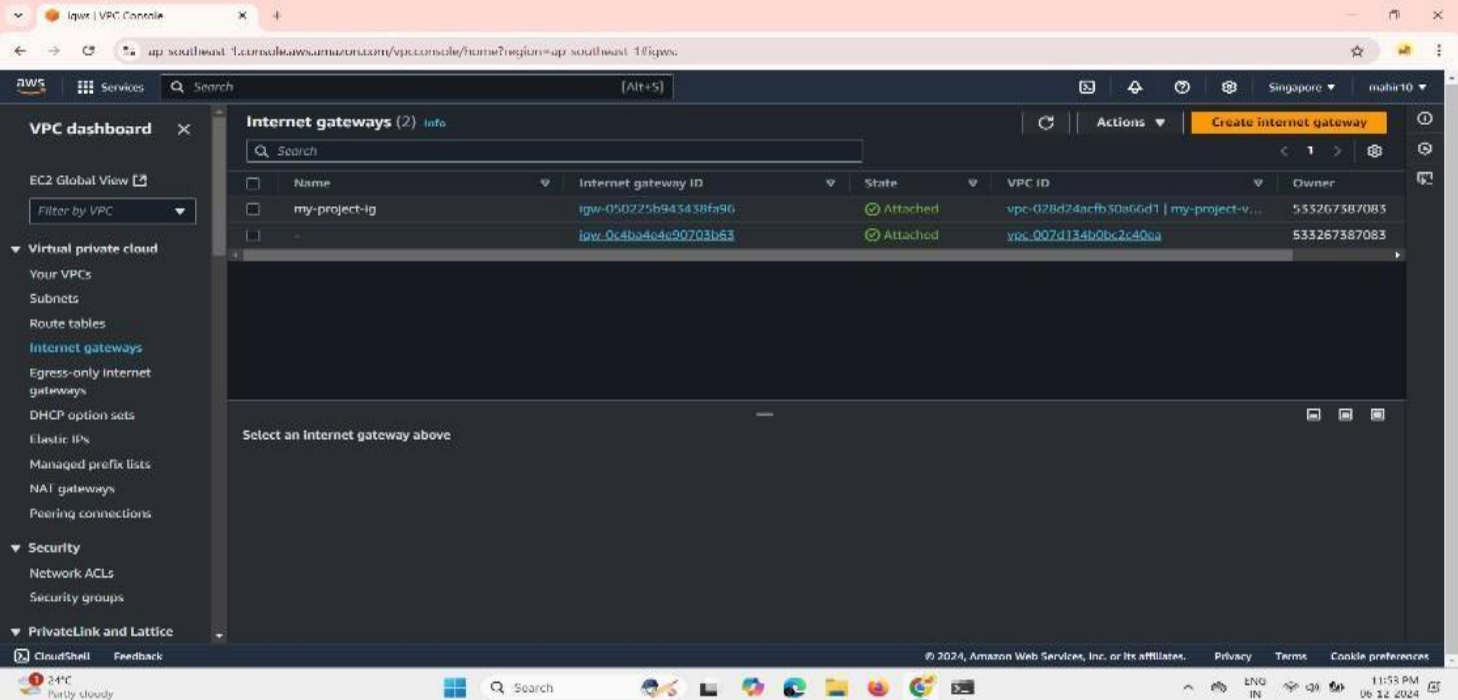
Figure 4.3  
Internet Gateway

Figure 4.4

**Public access** [Info](#)

☐ **Yes**  
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

☒ **No**  
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.



Name	Internet gateway ID	State	VPC ID	Owner
my-project-ig	igw-050225b945438fa96	Attached	vpc-028d24acfb30a66d1   my-project-v...	533267387083
-	igw-0c4ba4c4c90703b63	Attached	vpc-007d134b0bc2c406a	533267387083

Route table (my-rt-private-project), (my-pub-rt-project)

Figure 4.5

Create 16 Subnet: 2Public and other Private, created 2Database subnet and Application.

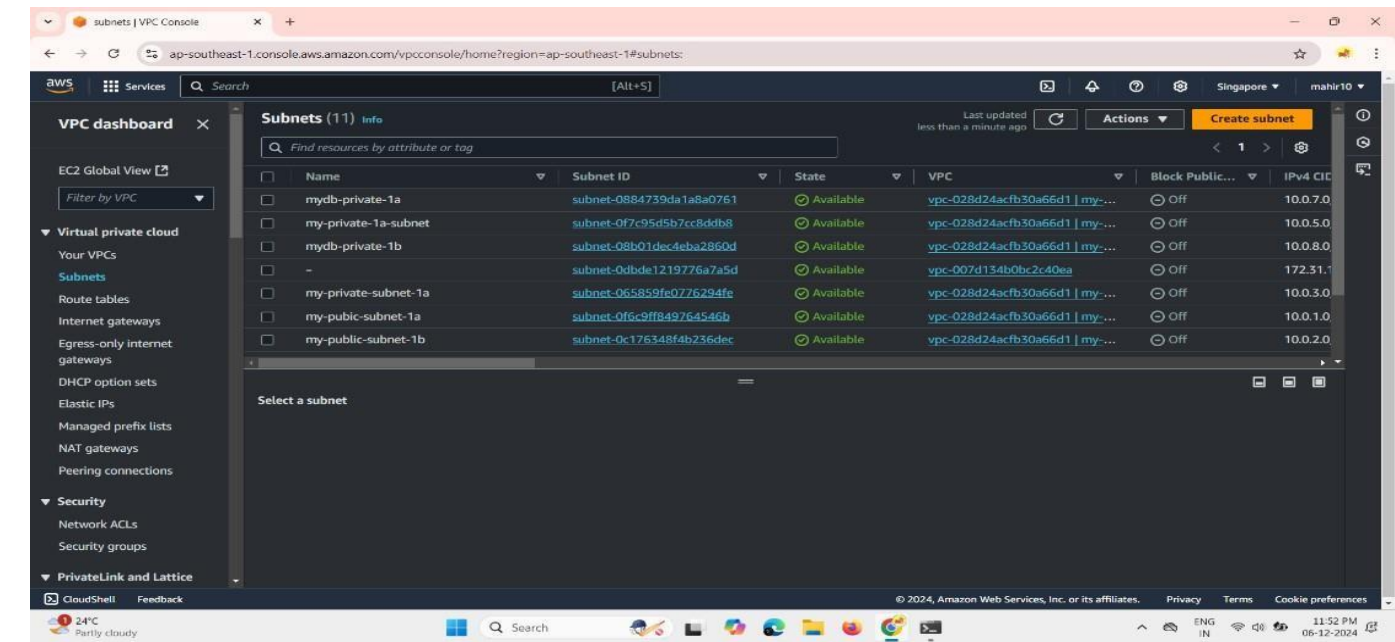


Figure 4.6

Create my-nat-project ( Nat gateway )

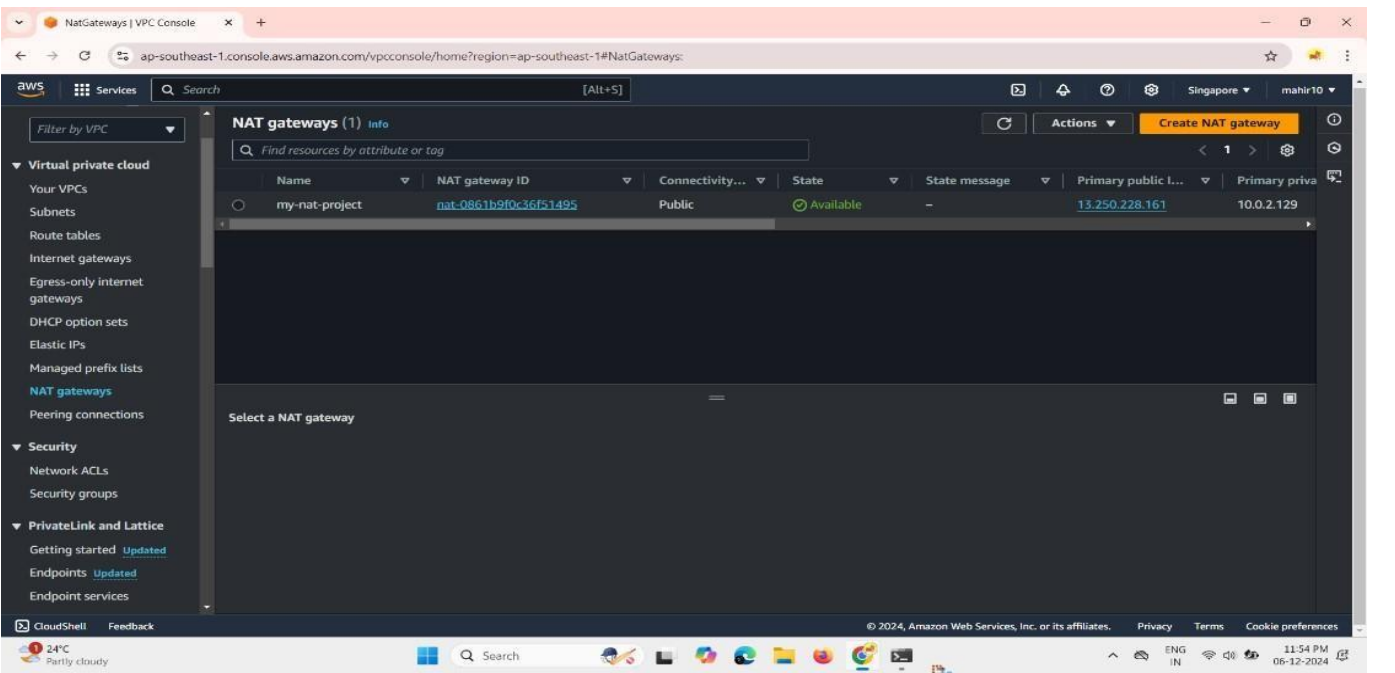
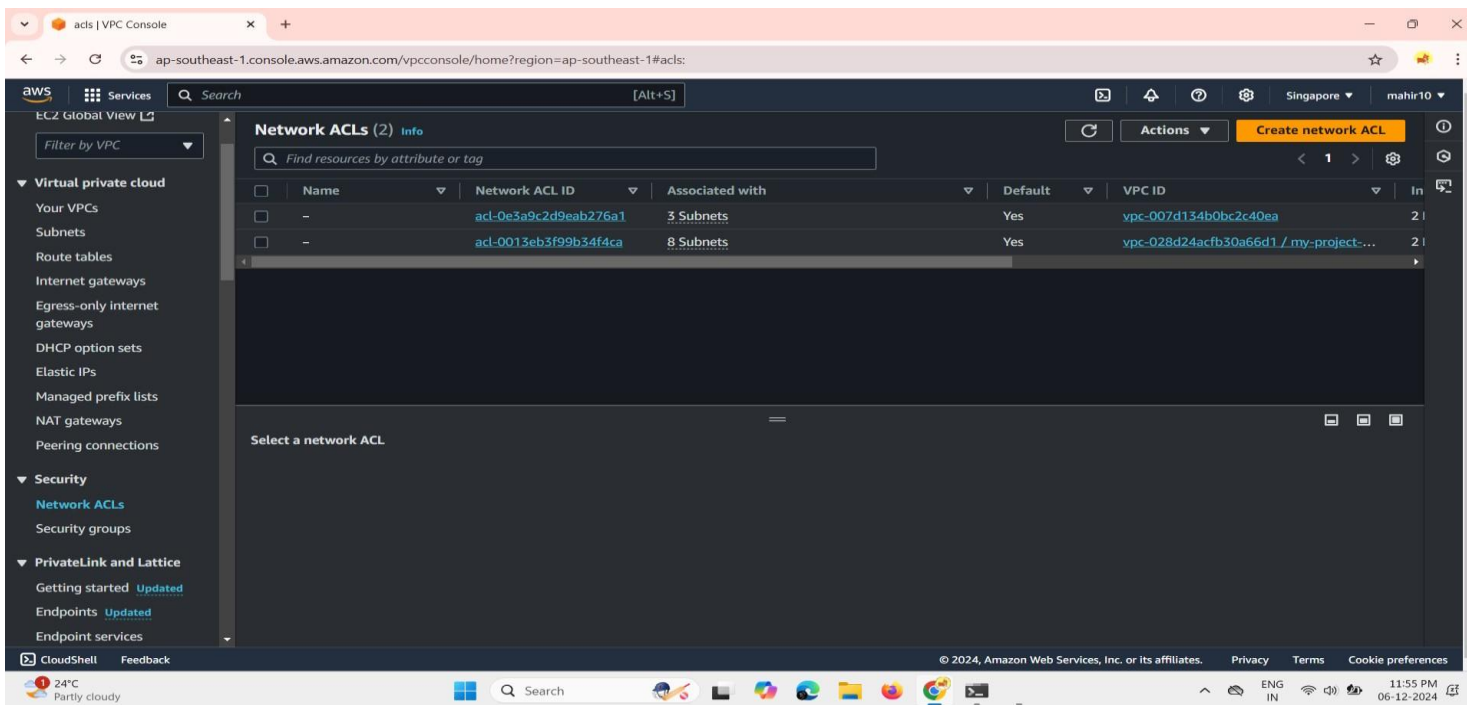
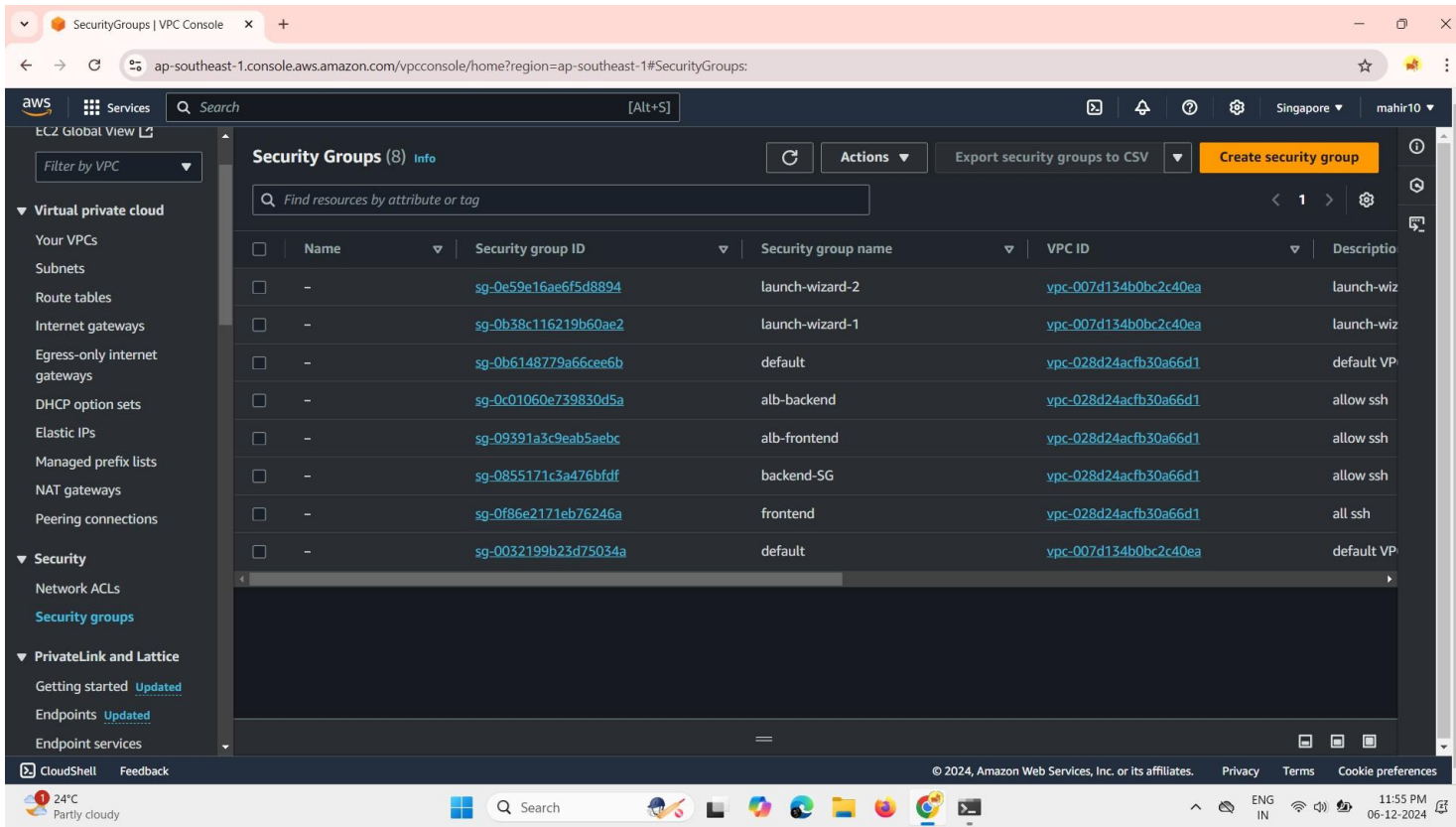


Figure 4.7

## Network ACLs





Security group (8)

Figure 4.9

Created 2Load Balancer, frontend and backend ALB

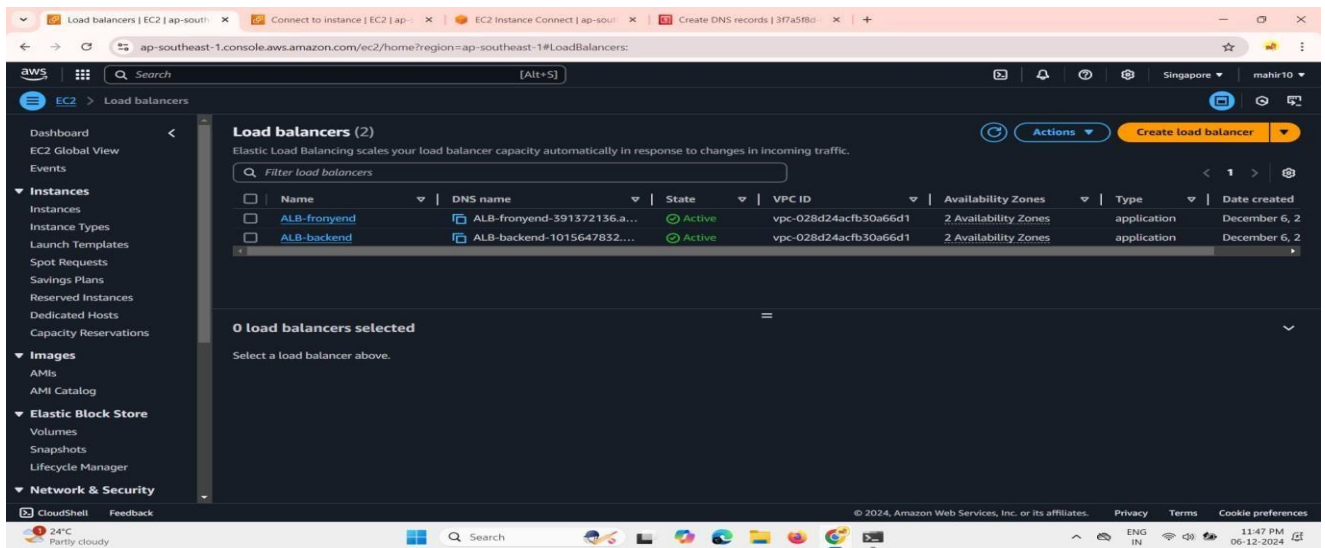


Figure 4.10

## Create Target Group (TG-frontend and TG-backend)

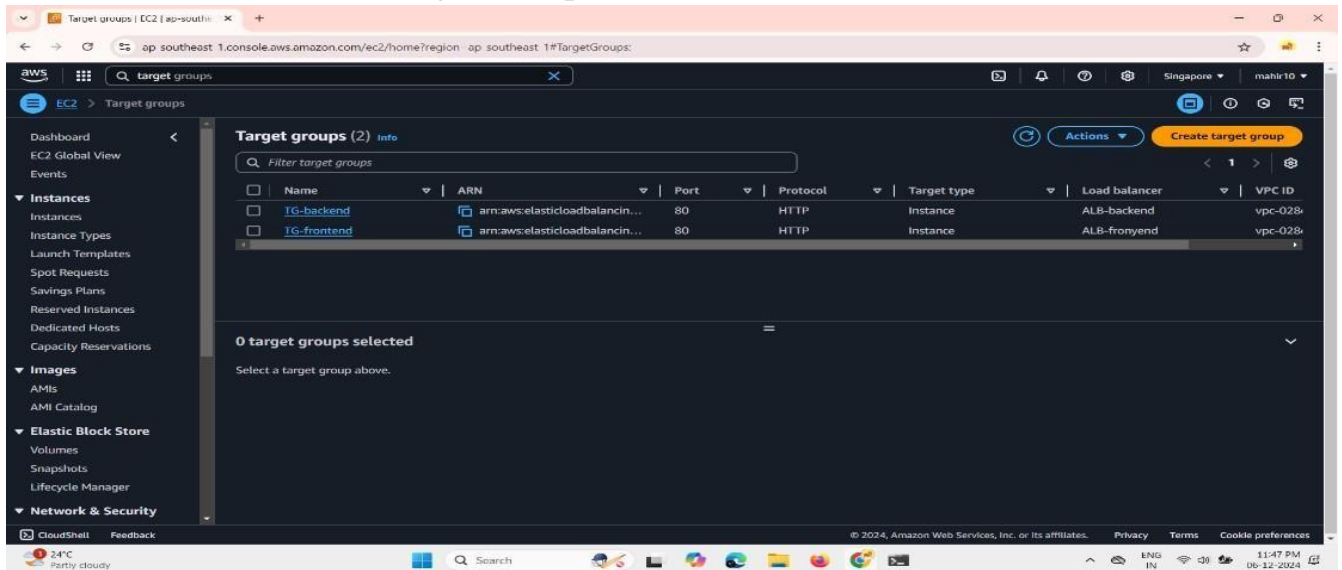


Figure 4.11

## Create Auto Scaling (ASG frontend and backend)

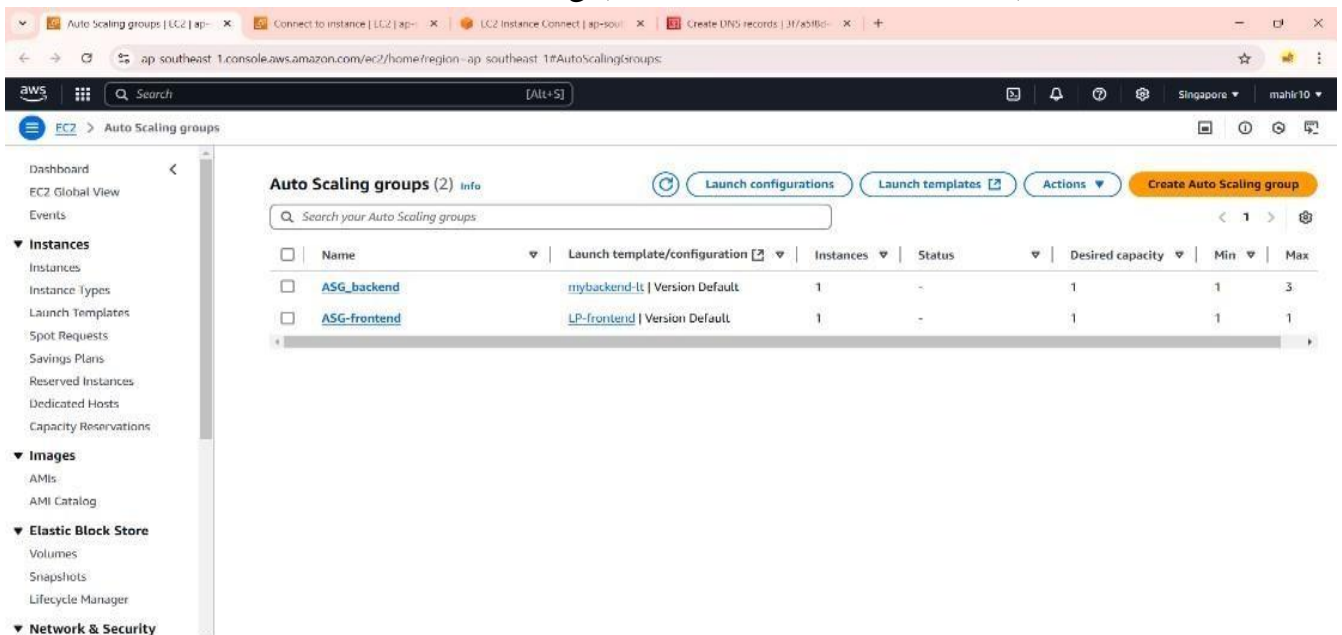


Figure 4.12



## Create Launch Templet (ID)

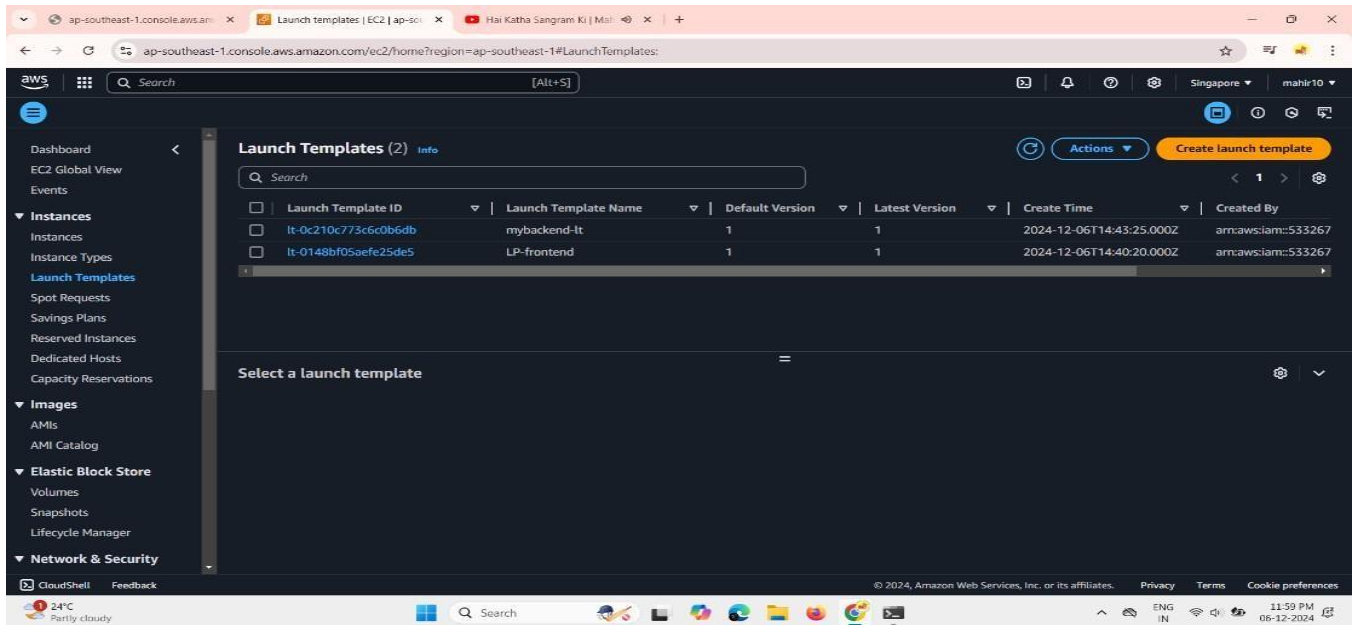


Figure 4.13

## Create AMI Frontend and Backend

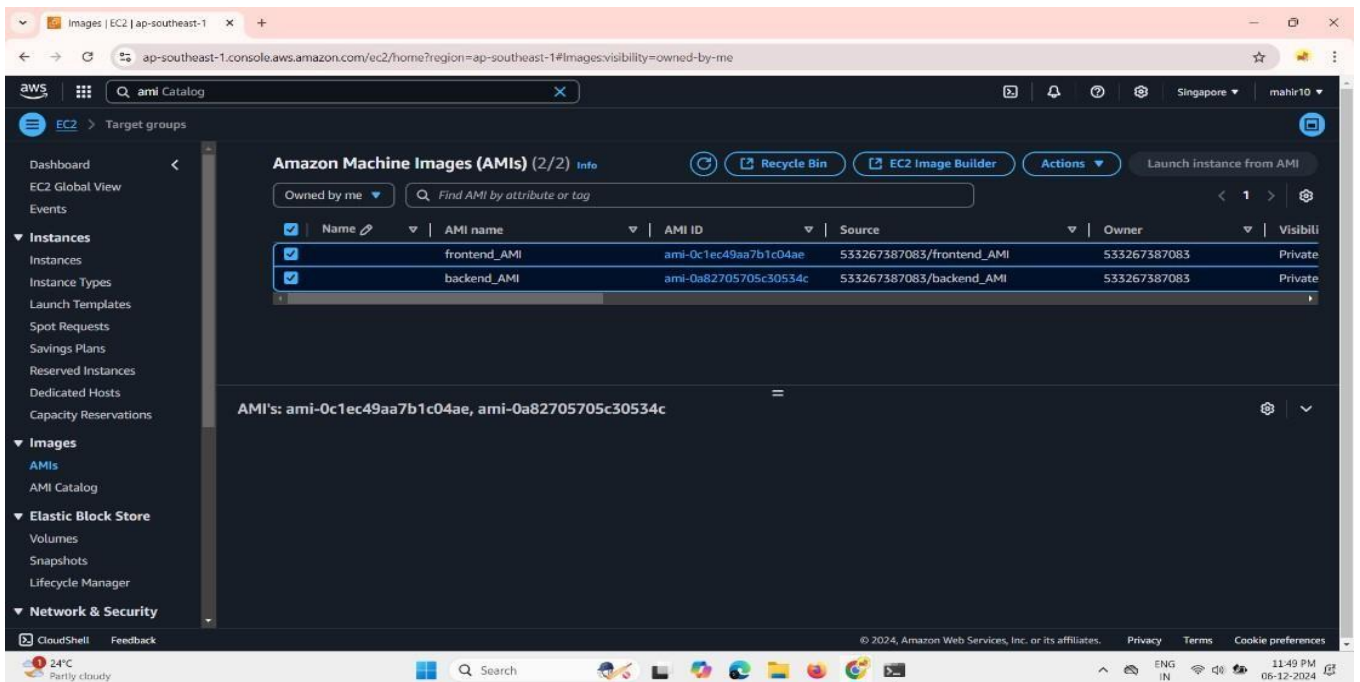
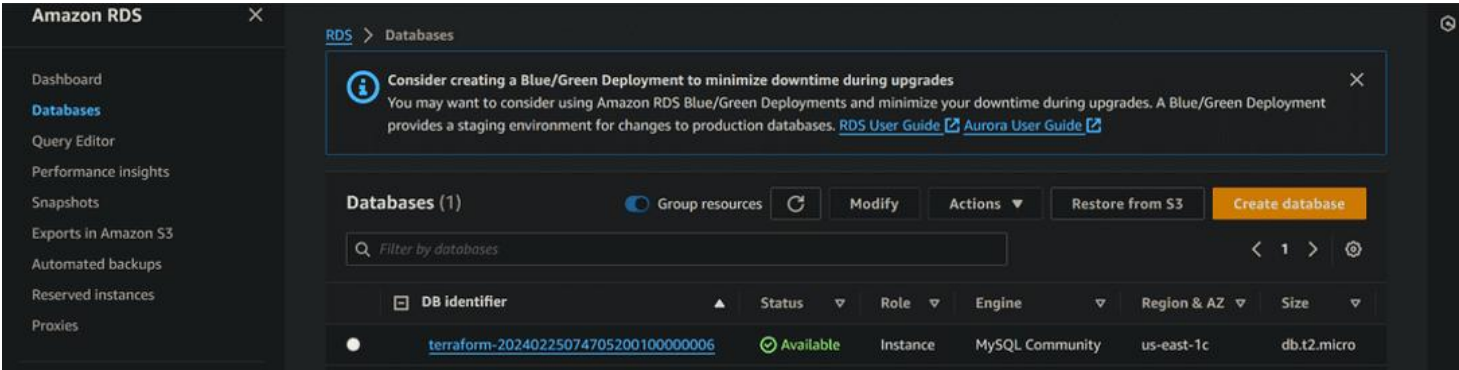
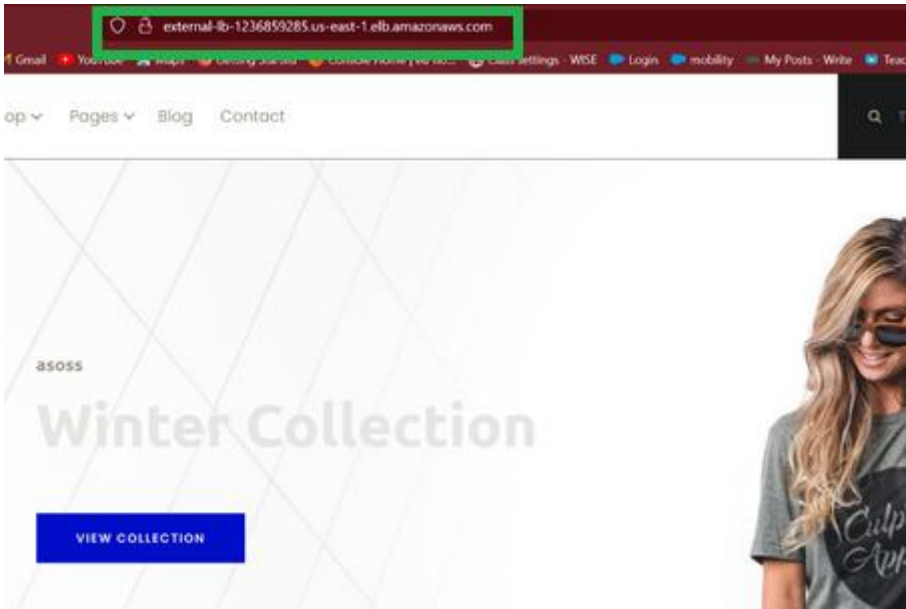


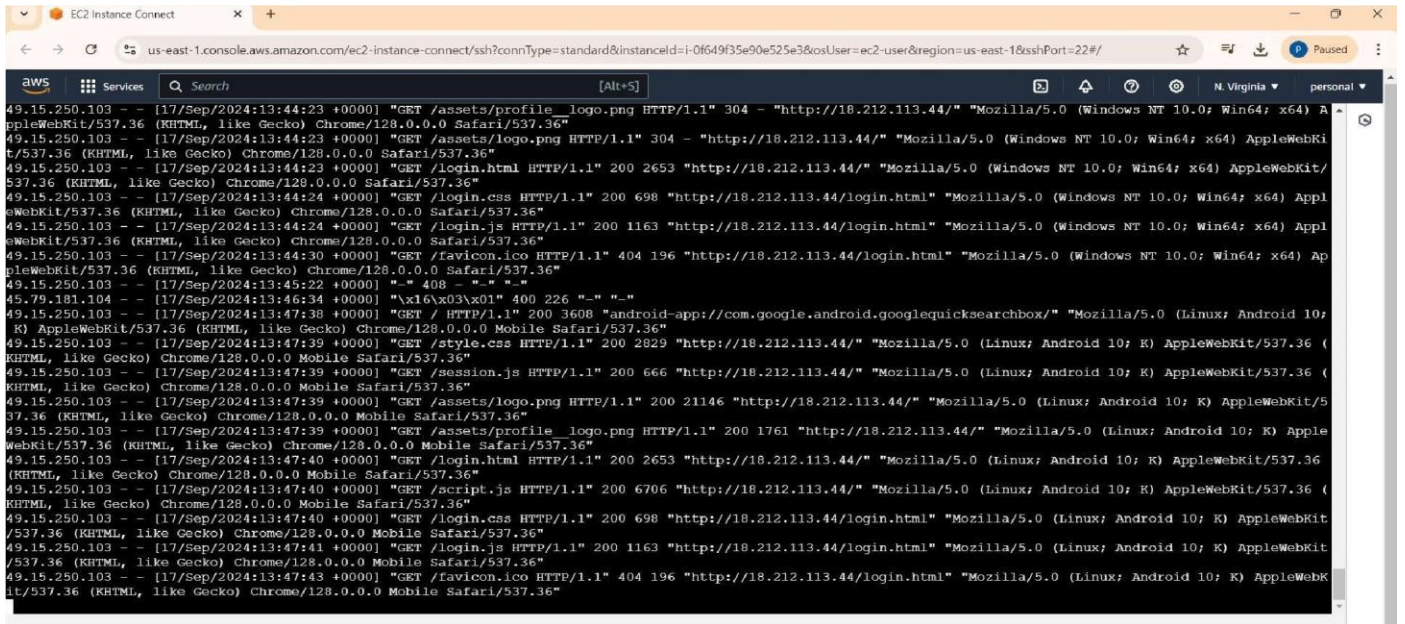
Figure 4.14  
RDS



Application



## Monitoring Mobaxtrm / Powershell



The screenshot shows a terminal window titled "EC2 Instance Connect" with a URL bar indicating an AWS console connection. The terminal displays a log of HTTP requests, each with a timestamp, IP address, method, path, status code, and user agent. The requests are from various user agents, including Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36, Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36, and Chrome/128.0.0.0 Mobile Safari/537.36. The requests are for various resources, including /assets/profile\_logo.png, /assets/logo.png, /login.html, /login.css, /login.js, /favicon.ico, /style.css, /session.js, and /script.js. The status codes are mostly 200, with one 408 and one 404.

```
49.15.250.103 - - [17/Sep/2024:13:44:23 +0000] "GET /assets/profile_logo.png HTTP/1.1" 304 - "http://18.212.113.44/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:44:23 +0000] "GET /assets/logo.png HTTP/1.1" 304 - "http://18.212.113.44/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:44:23 +0000] "GET /login.html HTTP/1.1" 200 2653 "http://18.212.113.44/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:44:24 +0000] "GET /login.css HTTP/1.1" 200 698 "http://18.212.113.44/login.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:44:24 +0000] "GET /login.js HTTP/1.1" 200 1163 "http://18.212.113.44/login.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:44:30 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://18.212.113.44/login.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:45:22 +0000] "-" 408 - "-" "-"
45.79.181.104 - - [17/Sep/2024:13:46:34 +0000] "\x16\x03\x01" 400 226 "-" "-"
49.15.250.103 - - [17/Sep/2024:13:47:38 +0000] "GET / HTTP/1.1" 200 3608 "android-app://com.google.android.googlequicksearchbox/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:39 +0000] "GET /style.css HTTP/1.1" 200 2829 "http://18.212.113.44/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:39 +0000] "GET /session.js HTTP/1.1" 200 666 "http://18.212.113.44/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:39 +0000] "GET /assets/logo.png HTTP/1.1" 200 21146 "http://18.212.113.44/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:39 +0000] "GET /assets/profile_logo.png HTTP/1.1" 200 1761 "http://18.212.113.44/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:40 +0000] "GET /login.html HTTP/1.1" 200 2653 "http://18.212.113.44/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:40 +0000] "GET /script.js HTTP/1.1" 200 6706 "http://18.212.113.44/" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:40 +0000] "GET /login.css HTTP/1.1" 200 698 "http://18.212.113.44/login.html" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:41 +0000] "GET /login.js HTTP/1.1" 200 1163 "http://18.212.113.44/login.html" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
49.15.250.103 - - [17/Sep/2024:13:47:43 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://18.212.113.44/login.html" "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Mobile Safari/537.36"
```

Figure 4.16



