

Three Tier Website Deployment on Amazon EC2 Instance

Submitted by
Mahir Dasare (18)

Under the guidance of
Dr S. S. Borde

In partial fulfilment of the award of
Bachelor of Technology
(Electronics & Computer Engineering)



Department of Electronics & Computer Engineering
Marathwada Institute of Technology,
Chhatrapati Sambhaji Nagar
Maharashtra
[2024-2025]

Abstract

This project report details the development and deployment of a Swiggy-like website using Amazon EC2 (Elastic Compute Cloud), one of Amazon Web Services' (AWS) most powerful cloud computing platforms.

The project focuses on leveraging EC2's scalable virtual infrastructure to efficiently host a website that can handle dynamic content and large volumes of traffic, as required by a food delivery platform like Swiggy.

The report highlights the need for cloud-based solutions in modern web hosting, offering enhanced flexibility, scalability, and cost-effectiveness over traditional hosting methods.

In this project, an EC2 instance is configured with the required software stack, including a web server, database, and backend logic, enabling a live website deployment.

Furthermore, the project discusses the advantages of using Amazon EC2 for website hosting, including scalability to handle variable traffic, high availability, reliability, and a pay-as-you-go pricing model. These features make EC2 particularly suited for businesses like Swiggy, which rely on stable and fast performance to manage real-time transactions and user interactions.

Deploying a website on an Amazon EC2 (Elastic Compute Cloud) instance involves the strategic setup of a virtual server to host your online content. This process begins with selecting the appropriate EC2 instance type, akin to choosing a new home based on your needs and budget. After launching the instance, you remotely access it using SSH, much like unlocking the doors to your new house. Inside, you set up a web server (such as Apache or Nginx) that will act as the host for your website—essentially furnishing your new space.

The next crucial step is configuring your domain name, directing it to your EC2 instance, similar to assigning your home a recognizable address. With your domain set, you then upload your website files, decorating your virtual home with your digital content. To safeguard your new online presence, you secure it with firewalls and SSL certificates, ensuring a safe environment for visitors. Continuous maintenance is essential, just like regular home upkeep, to keep your website running smoothly and efficiently. The entire journey transforms a mere idea into a vibrant, accessible web presence, blending technical prowess with a touch of creative vision.

CONTENT

CHAPTERS	PAGE NO
1. INTRODUCTION	
1.1 Introduction	5
1.2 Need of project	6
1.3 Objective of project	7
1.4 Features	8
2. LITERATURE SURVEY	9
2.1 Paper-1	
2.2 Paper-2	
2.3 Paper – 3	
3. SYSTEM MODELING	11
3.1 System Architecture	12
3.2 Data Flow Diagram and Working	14
3.3 Data Modelling	16
3.4 Implementation process	18
4 RESULT ANALYSIS	20
4.1 Software Requirements	21
4.2 Software Description	22
4.3 Advantages	23
4.4 Disadvantages	24
5. CONCLUSIONS	26

1. Introduction

The rise of cloud computing has revolutionized the way websites and applications are hosted and managed.

Amazon EC2, one of the most popular cloud services, offers scalable computing resources, which makes it ideal for launching websites and web applications.

This project demonstrates the process of launching a website similar to Swiggy, a popular food delivery service, using Amazon EC2.

This will include setting up an instance, configuring the necessary software, and deploying the website for public access. Picture yourself as an entrepreneur with a vision to create an online store. To bring this vision to life, you turn to Amazon EC2, a service that provides the cloud-based infrastructure needed to host your website.

Your journey begins by selecting an EC2 instance, similar to choosing the perfect location for your shop. Once you've launched your instance, you use SSH to access it, much like obtaining the keys to your new store. Inside, you set up a web server like Apache or Nginx, transforming an empty space into a welcoming environment for your website.

Next, you register a domain name, which is akin to putting up a sign with your shop's name, making it easy for people to find you. You then upload your website files, arranging your virtual store with all the products and decorations that represent your brand.

To ensure your store is safe, you set up security measures such as firewalls and SSL certificates, providing a secure experience for your visitors. Just like a physical store, your online store requires regular maintenance to keep everything running smoothly.

By using Amazon EC2, you take a technical process and turn it into a creative project, building a space on the internet where your vision can thrive.

1.2 Need of project

Launching websites via traditional hosting platforms can be cumbersome, involving significant upfront costs, fixed hardware limitations, and potential maintenance challenges.

- **Growth Flexibility:** As your website attracts more visitors, Amazon EC2 can easily scale to handle increased traffic, ensuring your site runs smoothly without slowing down.
- **Cost Savings:** Instead of spending a lot on physical servers, Amazon EC2 offers a pay-as-you-go model, which is budget-friendly for businesses of all sizes.
- **Reliability:** With Amazon EC2, your website is hosted on a reliable infrastructure with minimal downtime, so your site is always available to visitors.
- **Security:** Amazon EC2 provides robust security features like firewalls and SSL certificates, keeping your site and data safe from threats.
- **Global Access:** Amazon's extensive network allows you to host your website closer to your audience, improving load times and user experience, especially for international visitors.
- **Seamless Integration:** EC2 works smoothly with other AWS services like Amazon RDS for databases and S3 for storage, offering a complete hosting solution.
- **Customization:** You can customize your EC2 instance to meet the specific needs of your website, giving you control over your hosting environment.

1.3 Objective of project

- To Establish a Robust Online Presence: Create a website that can handle varying levels of traffic efficiently, ensuring high availability and performance.
- To Leverage AWS Infrastructure: Utilize Amazon EC2 and other AWS services to build a flexible and scalable hosting environment that can grow alongside the website's demands.
- To Ensure Cost Efficiency: Implement a cost-effective solution by using AWS's pay-as-you-go model, reducing the need for significant upfront investments in hardware and maintenance.
- To Enhance Security: Implement robust security measures, including firewalls, SSL certificates, and security groups, to protect the website and user data.
- To Optimize User Experience: Use AWS's global network of data centers to improve website load times and accessibility for a global audience.
- To Facilitate Easy Management and Maintenance: Integrate with other AWS services for seamless management, monitoring, and updates, ensuring the website remains up-to-date and secure.
- To Provide a Foundation for Future Growth: Create a scalable solution that can be easily expanded or modified as the website's requirements evolve.

1.4 Key Features of Amazon EC2

- **Scalability:** Dynamically scale instances up or down based on demand using Auto Scaling.
- **Flexibility:** Choose from a wide range of instance types, operating systems, and configurations to meet application requirements.
- **Pay-as-You-Go Pricing:** Pay only for the compute capacity you use, with options for on-demand, reserved, or spot instances.
- **High Availability:** Deploy applications across multiple availability zones to ensure fault tolerance and reliability.
- **Customizable Storage:** Attach Elastic Block Store (EBS) for scalable storage, or use instance storage for temporary data.
- **Enhanced Security:** Secure data using Security Groups, IAM roles, and encryption options.
- **Global Infrastructure:** Access EC2 instances in regions and availability zones worldwide for low-latency performance.
- **Integrated Monitoring:** Use AWS CloudWatch to track metrics, logs, and performance of instances.
- **Elastic IPs:** Assign static IP addresses for reliable server connections.
- **Support for Containers:** Easily run Docker containers on EC2 with Amazon Elastic Container Service (ECS) or Kubernetes.
- **Networking Features:** Configure Virtual Private Cloud (VPC) for advanced network settings like subnets and private IPs.
- **Spot Instances:** Leverage unused AWS capacity for cost savings in non-critical workloads.

2. LITERATURE SURVEY

The literature survey reviews cloud computing solutions, particularly focusing on Amazon Web Services (AWS), and explores different methods of deploying websites and web applications in cloud environments.

- **Survey on Serverless Computing** by Hassan B. Hassan, Saman A. Barakat, and Qusay I. Sarhan (2021)

This paper provides a comprehensive review of serverless computing, discussing its benefits, challenges, and applications. It highlights how serverless computing can reduce costs, improve scalability, and eliminate server-side management¹.

- **Creating Serverless Web Applications with AWS"** by Varun Singh, Chirag Acharya, Anisha Jain, Karishma Gupta, and Dr. Khushbu Wajari (2023)

This research paper categorizes serverless patterns and presents a case study using AWS Lambda, Amazon DynamoDB, and other services to process real-time data streams. It emphasizes the advantages of serverless computing in building scalable and reliable applications².

- **A Comparative Analysis of AWS Cloud-Native Application Deployment Model"** by Khandakar Razoan Ahmed and Motaharul Islam (2022)

This conference paper compares different cloud-native application deployment models, focusing on virtual machines and Docker containers. It evaluates their performance in terms of HTTP response, error percentage, and throughput³.

- **Deploy Website with AWS EC2 and Let's Encrypt [Step-by-Step]"** by Mahnoor Malik (2024)

This step-by-step guide provides practical instructions for deploying a website using AWS EC2 and Let's Encrypt. It covers creating an EC2 instance, configuring security groups, and setting up SSL certificates⁴

□ **Step-by-Step Guide: Creating an EC2 Instance to Host Your Website on AWS"** by MikaZuki (2023)

This tutorial walks through the process of setting up an Amazon EC2 instance to host a website. It includes launching an instance, configuring security groups, and installing a web serve

- **Serverless Computing:** Imagine a world where you don't have to worry about managing servers. Serverless computing, as discussed by Hassan B. Hassan and colleagues, offers this freedom by reducing costs, improving scalability, and eliminating server-side management¹.
- **Serverless Web Applications:** Varun Singh and his team explore how serverless computing can simplify web application development. They present a case study using AWS Lambda and other services to process real-time data, showcasing the benefits of this approach².
- **Cloud-Native Deployment Models:** Khandakar Razoan Ahmed and Motaharul Islam compare different deployment models, such as virtual machines and Docker containers. They evaluate their performance and help you choose the best option for your needs³.
- **Practical Guide to EC2 Deployment:** Mahnoor Malik provides a step-by-step guide to deploying a website using EC2 and Let's Encrypt. This guide is perfect for those who want a hands-on approach to setting up their website

3.SYSTEM MODELING

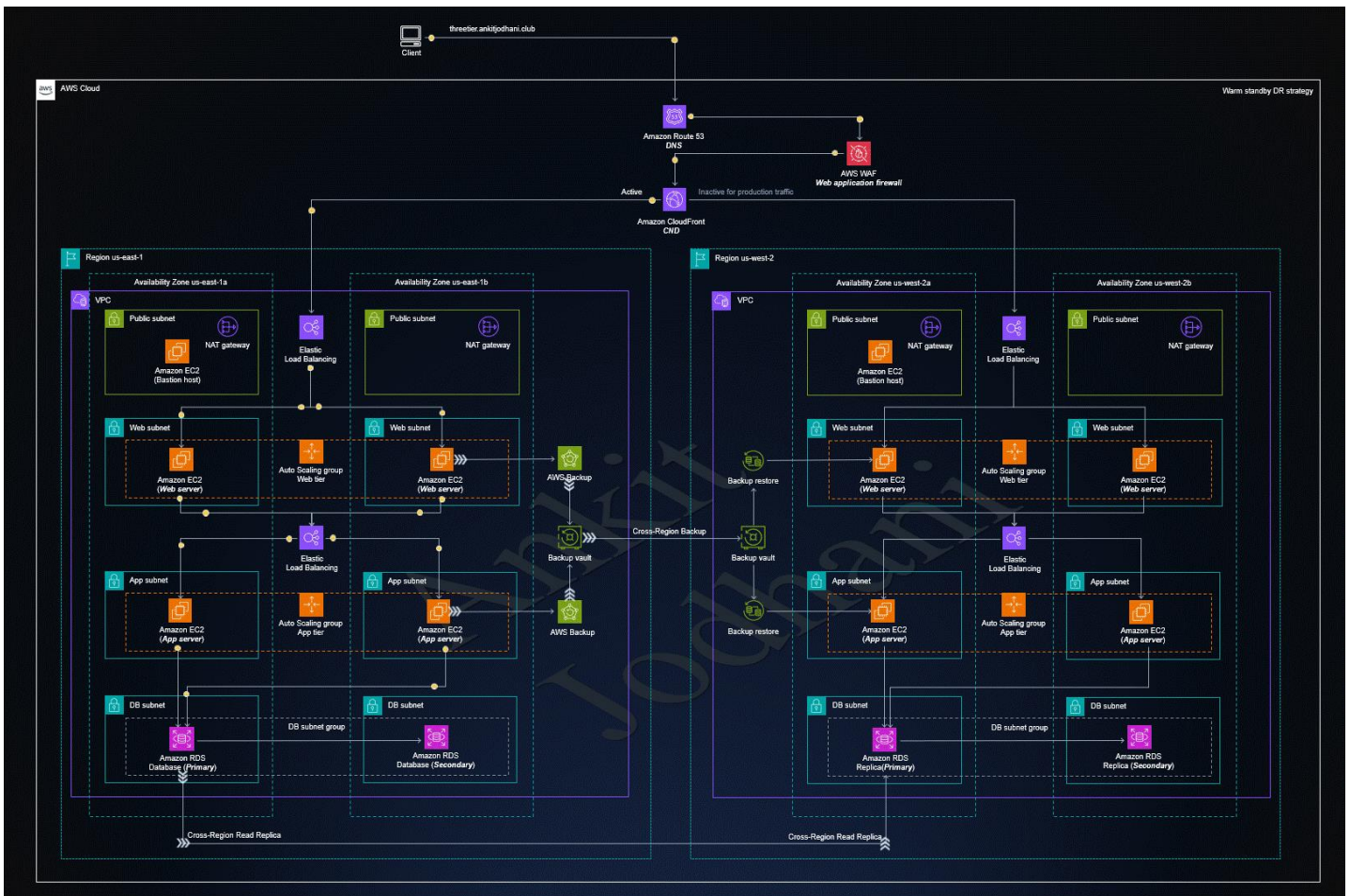


Figure 3.1 System Architecture

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 website 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 website, 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 website 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 website to visitors.

Purpose: To serve your website'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 website and its visitors, ensuring data privacy and integrity.

- **Domain Name:**

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

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

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

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

Purpose: To maintain version control over your website'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 website's data.

Purpose: To provide a reliable way to store and retrieve data for dynamic websites 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 website 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 website needs—small instances for lightweight websites and larger ones for hightraffic sites.

3. Configuring Security

To protect your website, 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 website 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 website to users. You also install any other software your website needs, like PHP, Python, or Node.js.

6. Uploading Your Website Files

The next step is to upload your website'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 website to work.

7. Setting Up a Database (If Needed)

If your website 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 website 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 website 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 website 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.

Three Types of Offers

More than 100 AWS products are available on AWS Free Tier today. Three different types of free offers are available depending on the product used. Click an icon below to explore our offers.



Figure 4.1

Steps :

Mumbai and Region Structure



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.

Public access [Info](#)

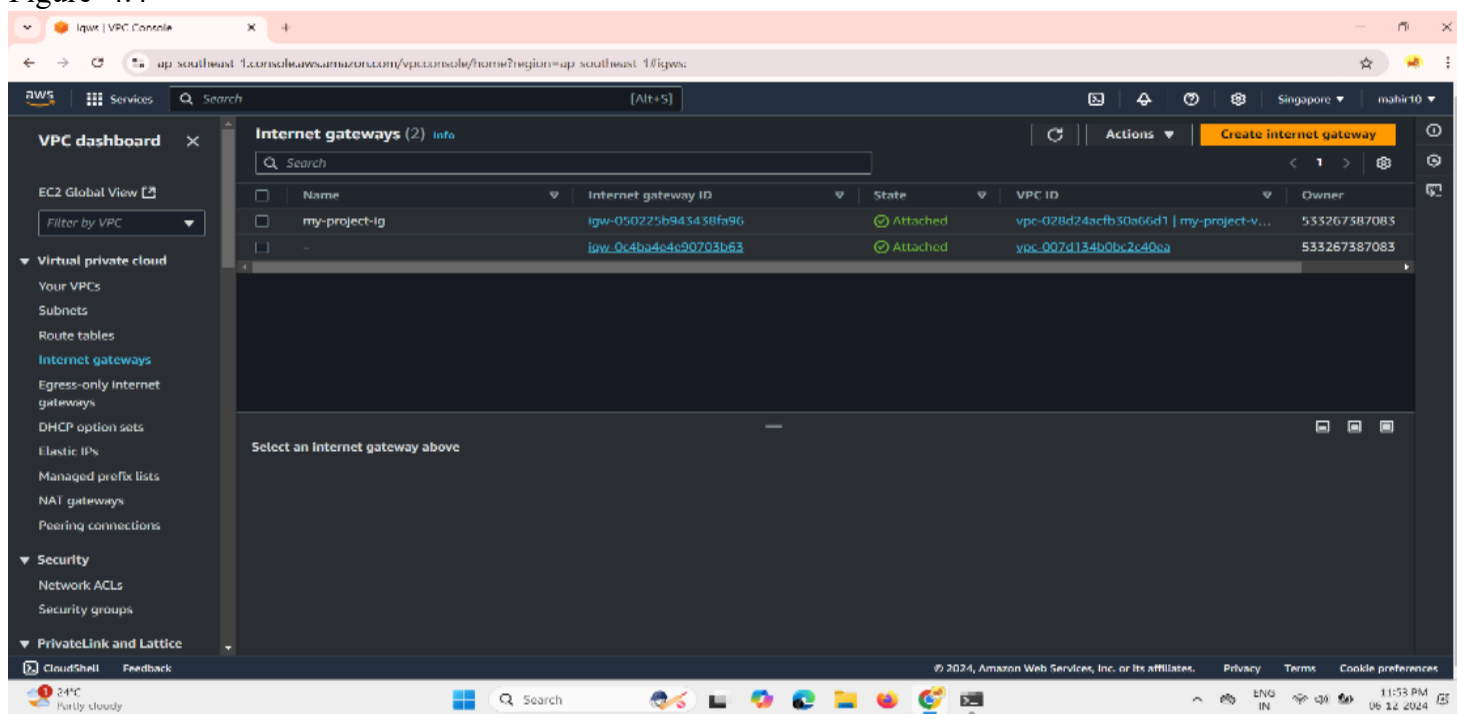
☐ **Yes**
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

☒ **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.

VPC
Figure 4.3

Internet Gateway

Figure 4.4



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

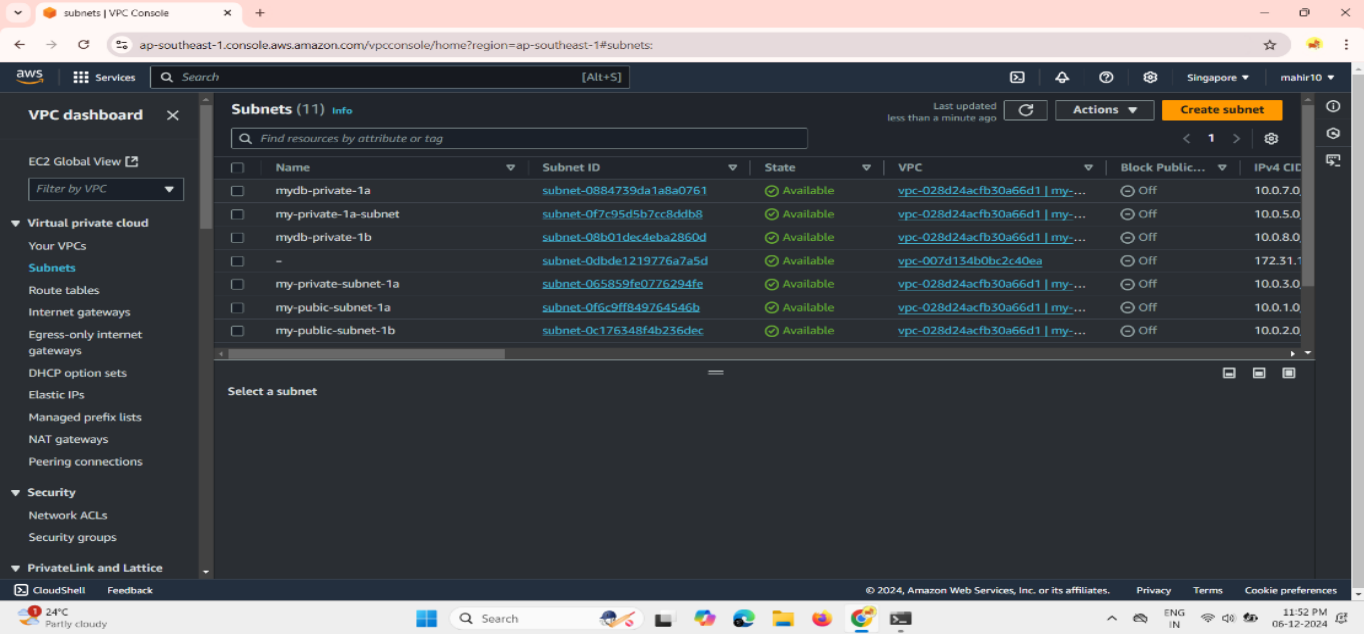


Figure 4.5

Create my-nat-project
Nat gateway

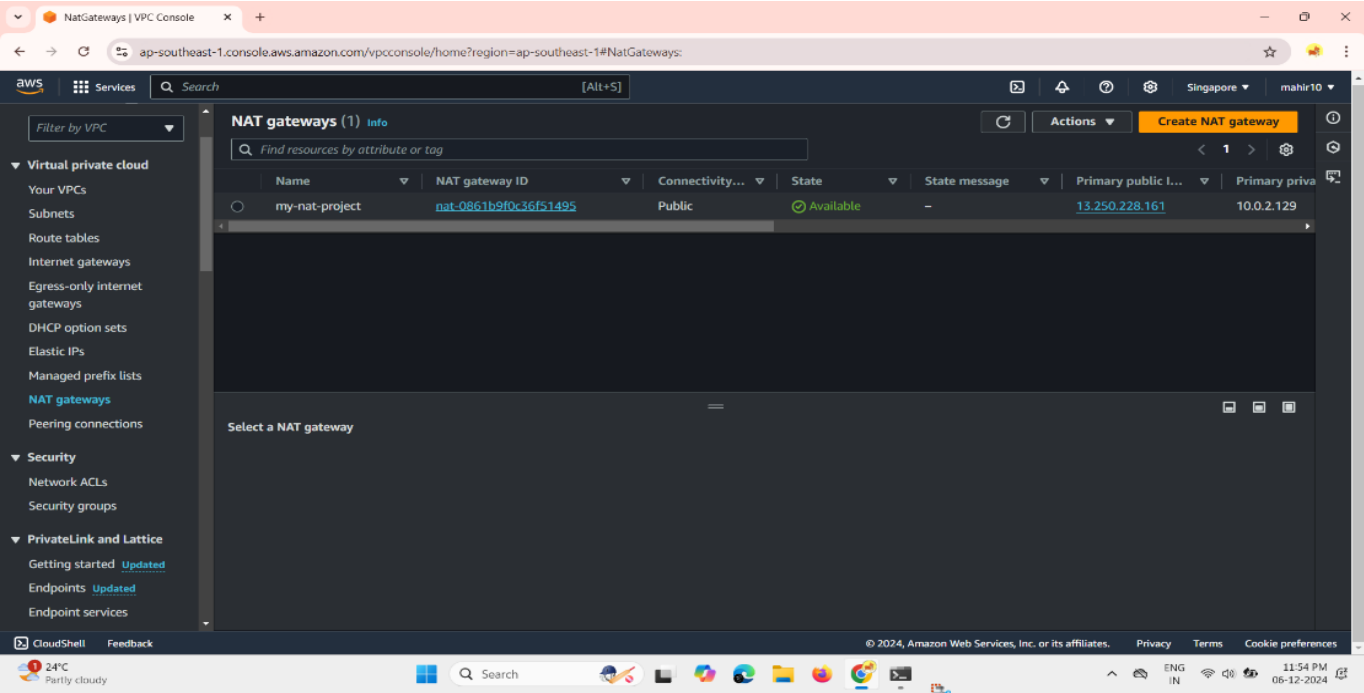


Figure 4.6

Created 2Load Balancer, frontend and backend ALB

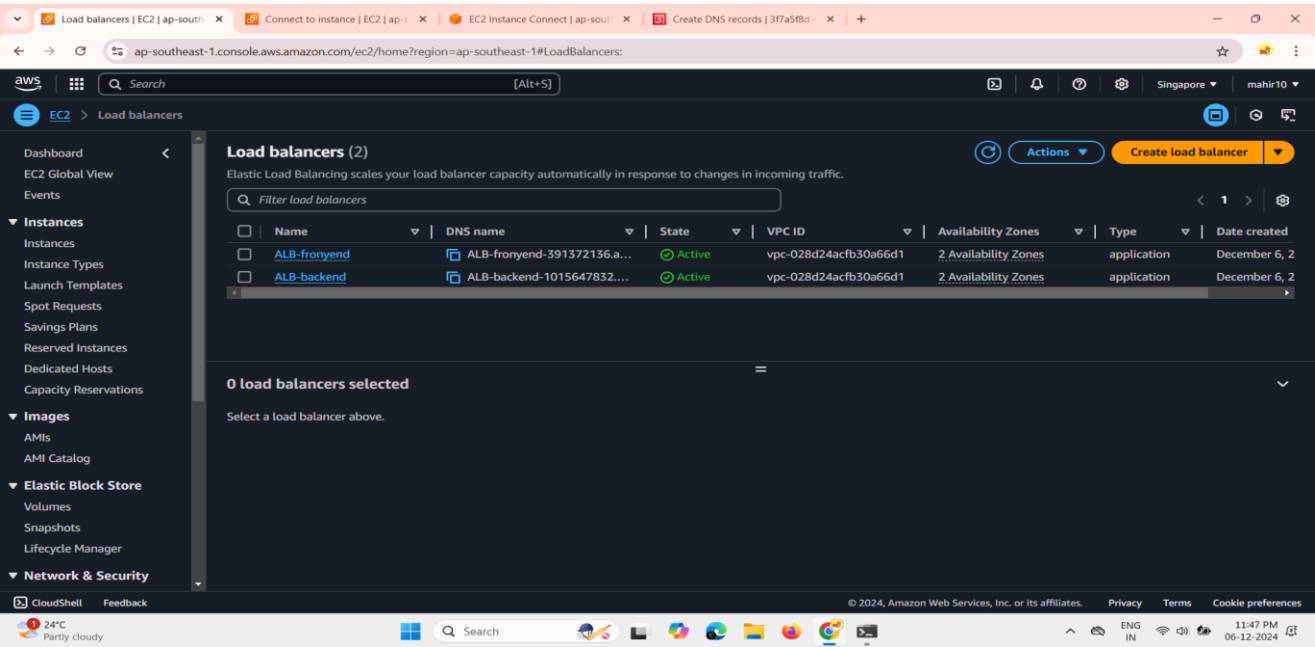


Figure 4.7

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

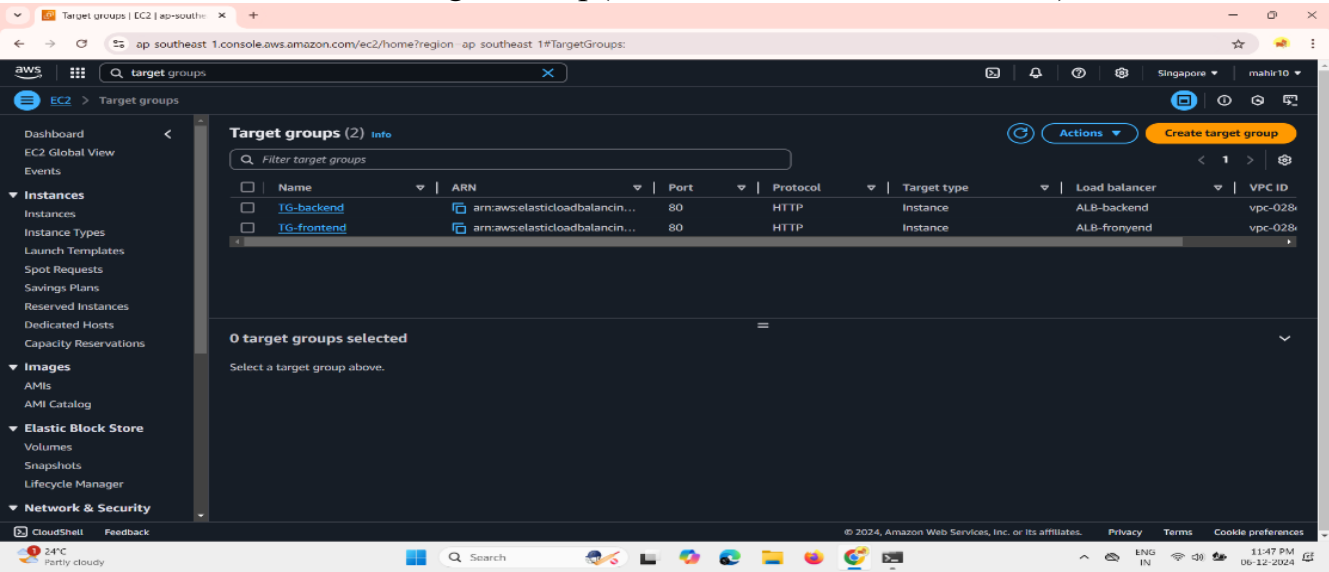


Figure 4.8

Create Auto Scaling(ASG frontend and backend)

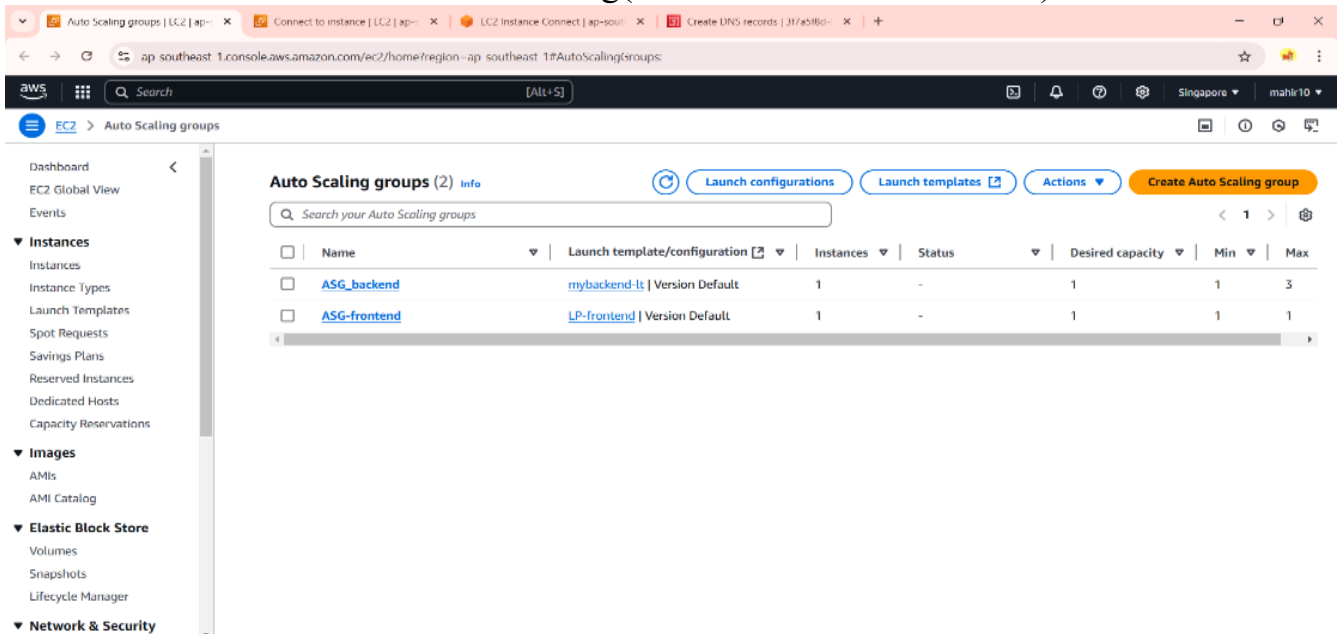


Figure 4.9

Create Launch Templet (ID)

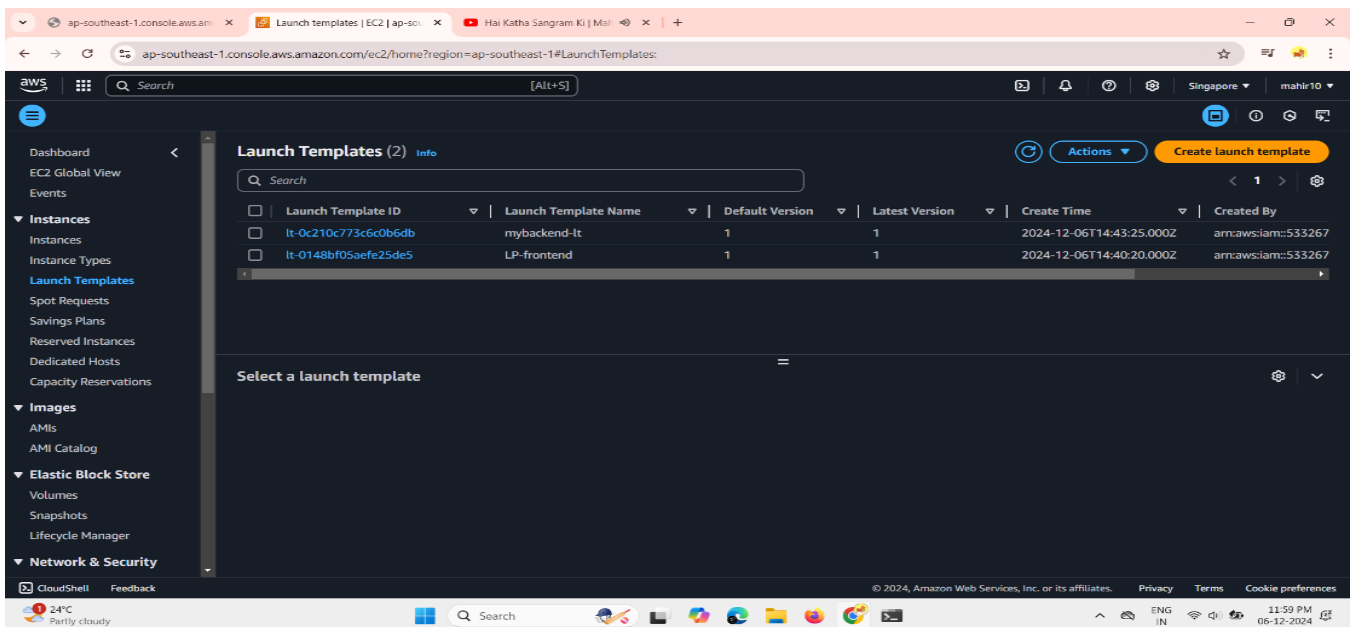


Figure 4.10

Create AMI Frontend and Backend

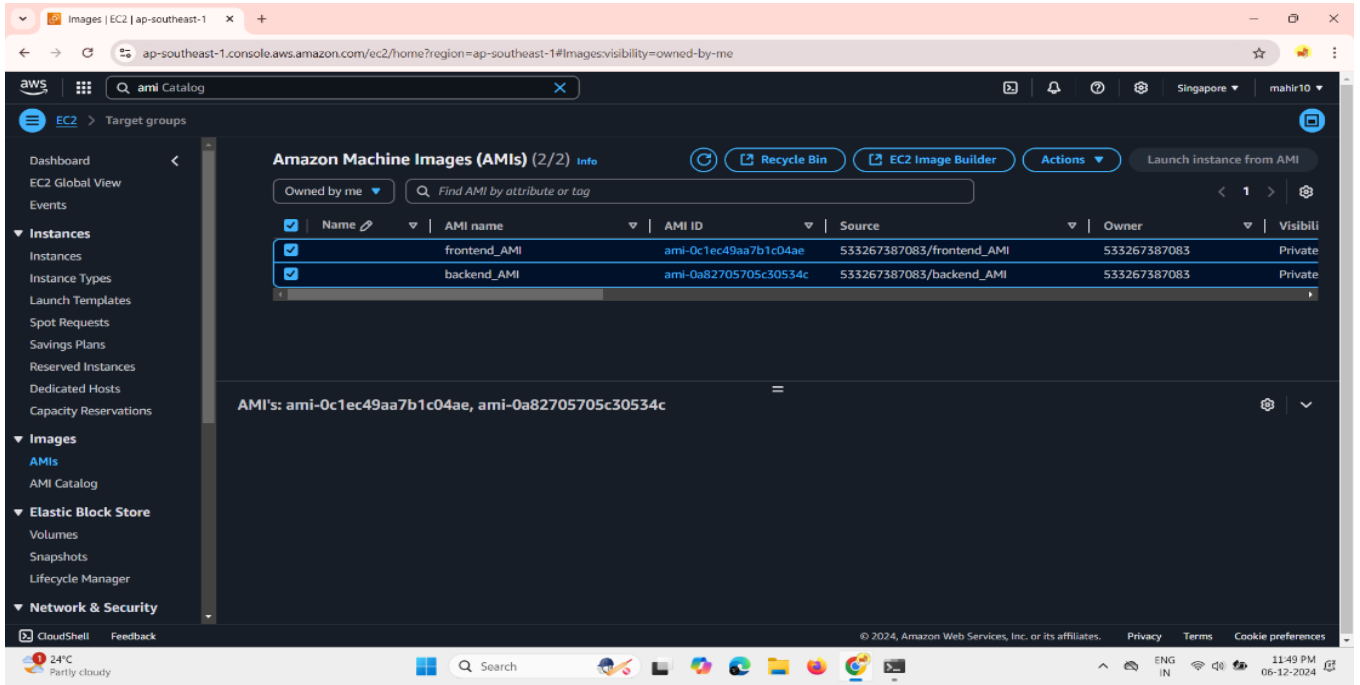


Figure 4.11

Create Certificate Manager frontend and backend

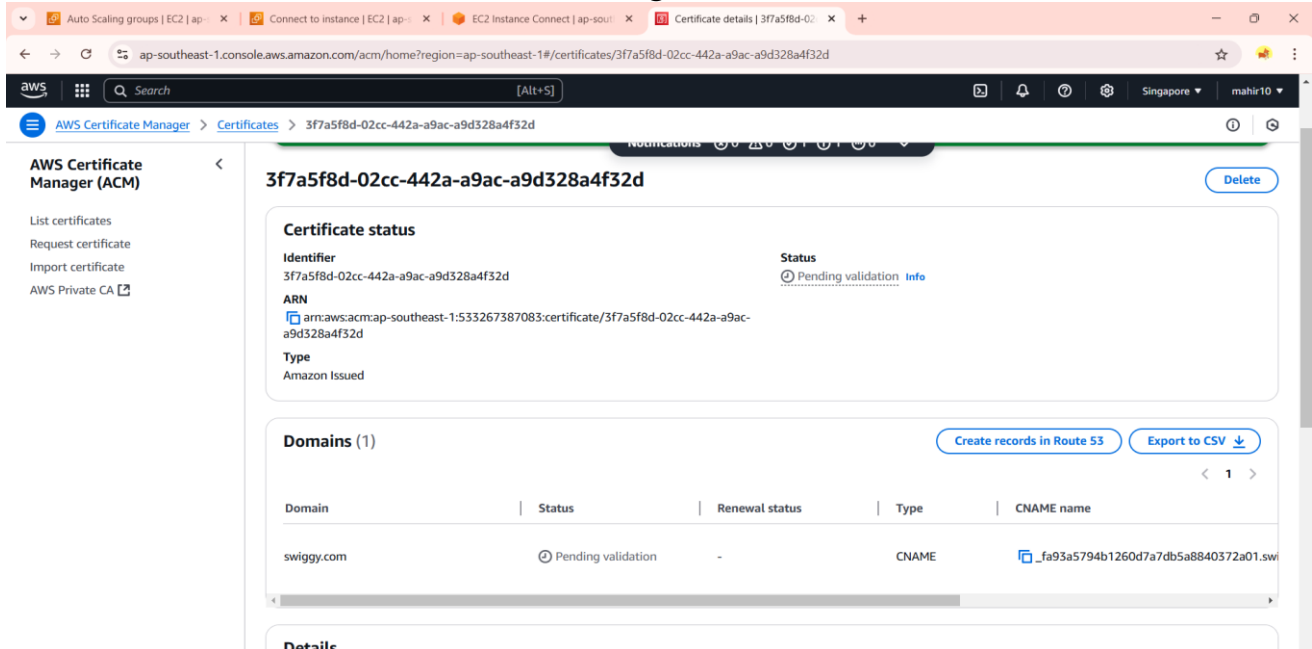
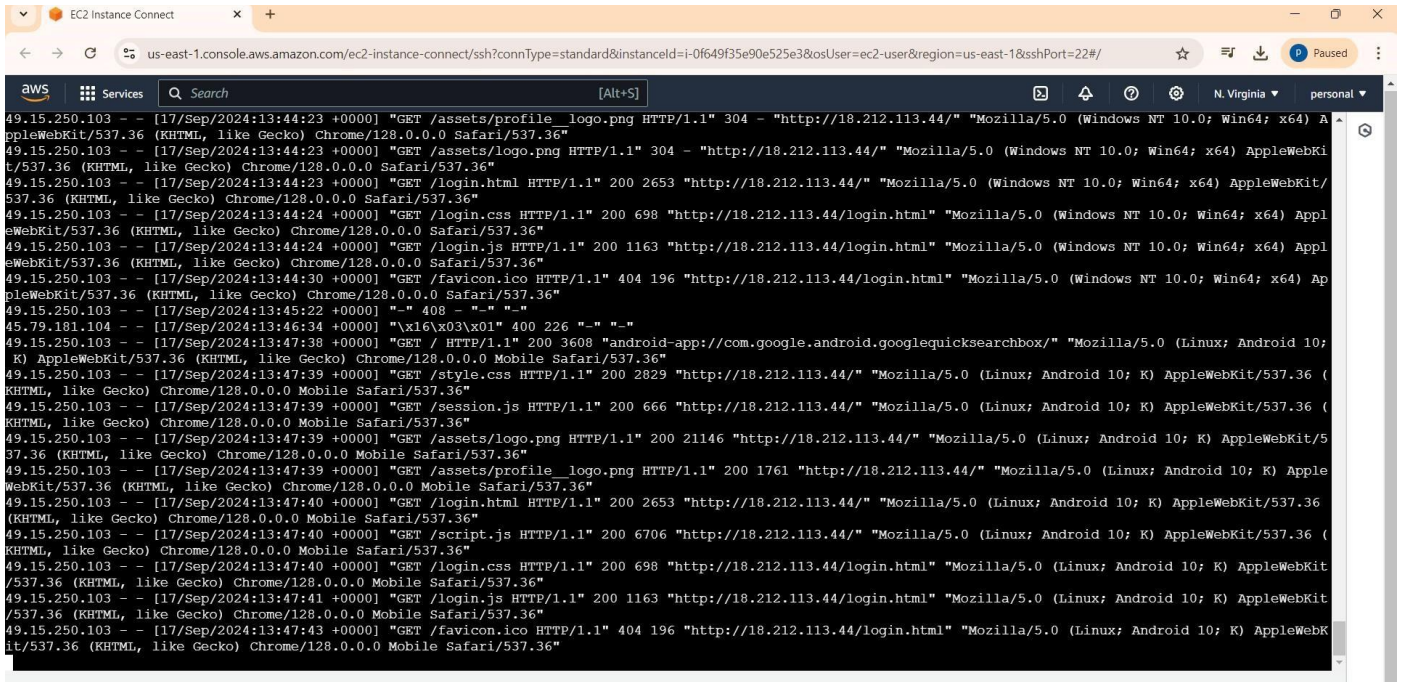


Figure 4.12

Monitoring Mobaxtrm / Powershell



The screenshot displays the AWS Management Console for an EC2 Instance Connect session. The browser address bar shows the URL: `us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0f649f35e90e525e3&osUser=ec2-user®ion=us-east-1&sshPort=22#`. The console interface includes the AWS logo, a search bar, and a navigation menu. The main content area shows a list of network connections, each with a timestamp, IP address, protocol, and user agent string. The connections are from various mobile devices, including iPhones and Android phones, all using Safari or Chrome browsers. The user agent strings are truncated in the screenshot.

```
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 - "-" "-"
49.79.181.104 - [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 4.13

4.3 Advantages

Launching a website using Amazon EC2 offers several distinct advantages:

1. **Scalability:** EC2 allows for both vertical and horizontal scaling of resources. The website can automatically adjust to handle higher traffic during peak times, making it ideal for platforms like Swiggy.
2. **Cost Efficiency:** With EC2's pay-as-you-go pricing model, you only pay for the computing resources you actually use, which can lead to significant savings compared to traditional hosting.
3. **Flexibility:** EC2 supports multiple operating systems and software platforms, enabling developers to choose the best stack for their needs. It also allows easy switching between different configurations.
4. **Reliability:** Amazon EC2 offers highly reliable services with data redundancy, regular backups, and load balancing, ensuring high uptime for your website.
5. **Security:** AWS provides multiple layers of security, including firewalls, encryption, and secure access controls, ensuring that sensitive data is well protected.
6. **Global Reach:** EC2 provides hosting in various geographic regions, allowing for faster load times for users in different parts of the world.
7. **Automation:** Features like Auto Scaling and Elastic Load Balancing enable the website to maintain optimal performance without manual intervention.

4.4 Disadvantages

While Amazon EC2 has many benefits, it's also important to be aware of some potential drawbacks:

1. **Complexity:** Setting up and managing EC2 instances can be pretty complicated, especially if you're new to cloud services. It requires a fair bit of technical knowledge and understanding of cloud infrastructure.
2. **Cost Management:** EC2 can get pricey if you're not careful. It's a pay-as-you-go service, so if you over-provision resources or forget to shut down instances you no longer need, the costs can add up quickly.
3. **Manual Maintenance:** Unlike some managed hosting services that handle updates and security for you, with EC2, you'll need to take care of these tasks yourself. This includes installing software updates, patching security vulnerabilities, and configuring your server.
4. **Learning Curve:** There's definitely a learning curve when it comes to using AWS services effectively. It takes time and effort to get comfortable with all the tools and settings.
5. **Scalability Challenges:** While EC2 is designed to be scalable, setting up autoscaling and managing multiple instances requires careful planning and can be challenging.
6. **Security Risks:** AWS provides robust security features, but you're responsible for configuring them correctly. Mistakes can leave your website vulnerable to attacks.
7. **Internet Dependency:** Since EC2 is cloud-based, you'll need a stable internet connection to manage and access your instances. Any issues with your internet can make it difficult to manage your website.

5.Conclusions

- **Scalable Hosting Solution:** Amazon EC2 provides a reliable and scalable platform for hosting websites, making it suitable for both small projects and large-scale applications..
- **Enhanced Security:** AWS offers robust security features, like Security Groups and Identity Access Management (IAM), which help safeguard websites from potential threats.
- **Learning Curve:** While EC2 is powerful, it has a steep learning curve. Users need to invest time in understanding AWS tools and services to fully utilize its potential.
- **Performance Monitoring:** AWS tools like CloudWatch provide excellent monitoring capabilities, helping users optimize their website performance and troubleshoot issues effectively. Deploying a website using Amazon EC2 is a strategic choice for businesses and individuals looking to establish a reliable and scalable online presence.
- Amazon EC2 offers a robust infrastructure that can handle varying levels of traffic, making it suitable for websites of all sizes.
The flexibility and customization options provided by EC2 allow users to tailor their server environment to meet specific requirements, ensuring optimal performance.
- One of the main advantages of using Amazon EC2 is its scalability. As your website grows and attracts more visitors, EC2 allows you to easily scale resources up or down to match demand.
- This flexibility is crucial for maintaining website performance and avoiding downtime during traffic spikes. Moreover, the pay-as-you-go pricing model helps manage costs effectively, ensuring that you only pay for the resources you actually use.
This global infrastructure also enables you to deploy your website closer to your users, reducing latency and improving load times for a better user experience.
- Security is another critical aspect of website deployment on Amazon EC2. AWS offers a range of security features, including firewalls, security groups, and SSL certificates, to protect your website and user data. By implementing these security measures, you can safeguard your online presence against threats and ensure compliance with industry standards and regulations.

- Maintenance is another consideration when using EC2. Unlike fully managed hosting services, EC2 requires you to handle maintenance tasks such as software updates, patching, and security configurations. This ongoing maintenance can be time-consuming and requires continuous monitoring to ensure that your website remains secure and performs optimally.
- Despite these challenges, the benefits of deploying a website on Amazon EC2 outweigh the drawbacks. The ability to scale resources, the reliability of the AWS infrastructure, and the robust security features make EC2 an excellent choice for hosting websites. Furthermore, the integration with other AWS services, such as Amazon RDS for databases and S3 for storage, provides a comprehensive and efficient hosting solution.
- In conclusion, deploying a website using Amazon EC2 empowers you to create a scalable, reliable, and secure online presence. While there are complexities and challenges involved, the flexibility and control offered by EC2, along with the extensive AWS ecosystem, make it a compelling option for businesses and individuals seeking to build a strong digital footprint.

References

Amazon Web Services (AWS) Documentation – EC2 Overview and Deployment:
<https://docs.aws.amazon.com/ec2/>

Swiggy's Official Website and Tech Stack Reviews – Insights into cloud- based deployment in the food delivery industry.

Apache/Nginx Documentation – Guide to setting up a web server:
<https://httpd.apache.org/docs/> and <https://nginx.org/en/docs/>

MySQL/MariaDB Documentation – Database setup and configuration:
<https://dev.mysql.com/doc/> and <https://mariadb.org/documentation/>

Literature on Cloud Computing in Web Development – Scholarly articles and case studies on cloud-based hosting and its benefits.

Guidance for Building a Containerized and Scalable Web Application on AWS:
<https://aws.amazon.com/solutions/guidance/building-a-containerized-and-scalable-webapplication-on-aws>

Step-by-Step Guide: Deploying a Full-Stack App on AWS EC2:
<https://dev.to/backendbro/step-by-step-guide-deploying-a-full-stack-app-onaws-ec2-21e1>

Step-by-step guide: Creating an EC2 Instance to Host your website on AWS:
<https://dev.to/mikacodez/step-by-step-guide-creating-an-ec2-instance-to-host-yourwebsite-on-aws-21ci>

