AWS LOAD BALANCING FOR WEBSITE OPTIMIZATION

Prepared in the partial fulfilment of the Summer Internship Program on AWS at

Under the guidance of

Mr. Anil Kumar Varanasi

Submitted by

Siva Nagendra Akurathi, 21471A0560

Narasaraopeta Engineering College, Narasaraopet

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of my summer internship project at **Andhra Pradesh Skill Development Corporation (APSSDC)**. This opportunity has been an enriching and transformative experience for me, and I am truly thankful for the support, guidance, and encouragement I have received along the way.

I would like to thank the entire team at **Andhra Pradesh Skill Development Corporation (APSSDC)** for fostering a collaborative and innovative environment. The camaraderie, knowledge sharing, and feedback I received from my colleagues significantly contributed to the development and success of this project.

In conclusion, I am honoured to have been a part of this internship program, and I look forward to leveraging the skills and knowledge gained to contribute positively to future endeavours. Thank you.

Sincerely,

Siva Nagendra Akurathi, 21471A0560

sivanagendra2004@gmail.com

ABSTRACT

The "AWS Load Balancing for Website Optimization" project represents a critical initiative aimed at enhancing the performance and availability of our website. In an increasingly digital world, ensuring a seamless online experience for users is paramount. This project leverages Amazon Web Services (AWS) to implement an efficient load balancing solution.

Load balancing plays a pivotal role in distributing incoming web traffic across multiple servers, preventing overload, and reducing the risk of downtime. The project's primary focus is to achieve optimal website performance, reduce latency, and improve response times, thus significantly enhancing the overall user experience.

Additionally, the implementation of redundant server configurations ensures high availability, bolstering our website's reliability and mitigating potential disruptions. The extensible design of our AWS infrastructure positions us for future scalability, accommodating anticipated growth in web traffic and potential expansion.

In summary, "AWS Load Balancing for Website Optimization" underscores our commitment to delivering a reliable, high-performing, and scalable online platform. This project not only elevates our website's performance but also contributes to cost-efficiency by optimizing resource utilization and minimizing hardware investments.

TABLE OF CONTENTS

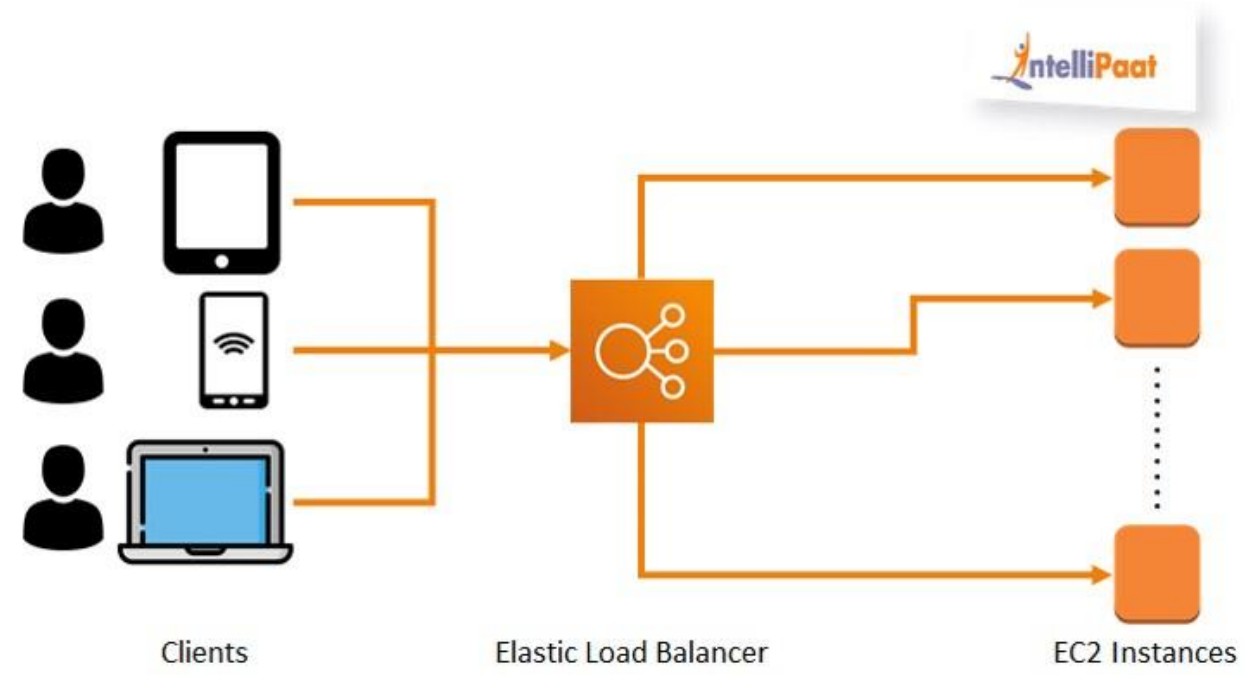
1. INTRODUCTION

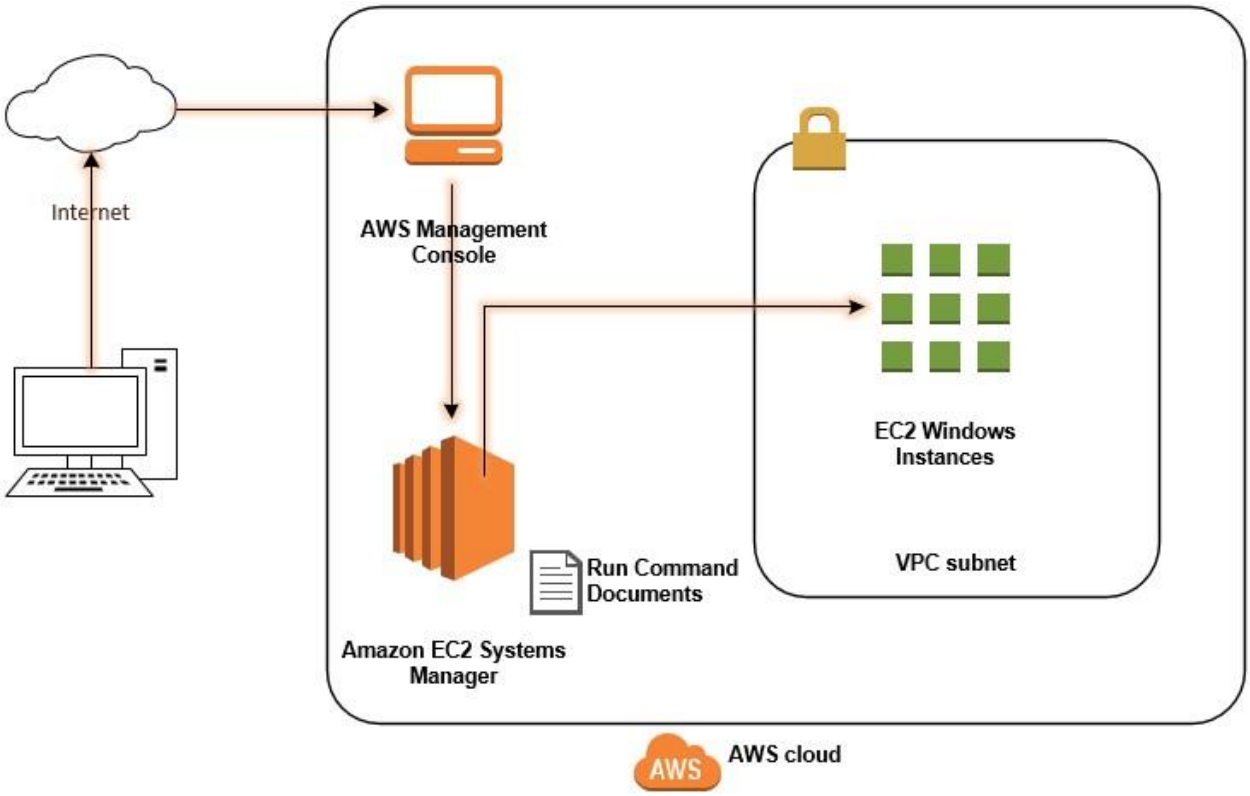
In the contemporary digital landscape, maintaining a strong online presence is fundamental for businesses, organizations, and individuals alike. However, as web traffic continues to surge, the challenge of ensuring consistent website accessibility and responsiveness under varying workloads becomes increasingly critical. The "Managing Website Using AWS Load Balancer" project is designed to tackle these challenges by harnessing the capabilities of Amazon Web Services (AWS) to construct a robust and scalable web infrastructure.

Key Components:

Amazon Web Services (AWS): AWS, a leading cloud computing platform, provides a comprehensive suite of services for creating, deploying, and managing web applications and websites. It offers unparalleled scalability, reliability, and cost-efficiency.

Load Balancer: At the heart of this project lies the load balancer, a pivotal component responsible for intelligent traffic management. It ensures the equitable distribution of incoming requests across multiple backend servers or instances, eliminating the risk of server overload and preventing a single server from becoming a bottleneck.


IntelliPaat
Clients
Elastic Load Balancer
EC2 Instances


Internet
AWS Management Console
EC2 Windows Instances
VPC subnet
Run Command Documents
Amazon EC2 Systems Manager
AWS
AWS cloud

Amazon EC2 Instances: Amazon Elastic Compute Cloud (EC2) instances, being virtual servers in the AWS cloud, play a pivotal role in hosting our website. They are meticulously managed and dynamically scaled to meet the demands of our workload.

Auto Scaling: AWS Auto Scaling is employed to empower our infrastructure with the capability to automatically adjust the number of EC2 instances in response to changing traffic patterns. This intelligent system ensures that our website consistently delivers optimal performance without necessitating manual intervention.

Amazon RDS: For websites reliant on databases, Amazon Relational Database Service (RDS) is employed to manage database instances efficiently. This service ensures high availability, scalability, and simplified database management.

Project Goals:

The "**Managing Website Using AWS Load Balancer**" project is driven by several key objectives:

High Availability: Ensuring our website remains accessible 24/7, thus minimizing downtime and providing a reliable user experience.

Scalability: Developing an infrastructure capable of seamlessly accommodating traffic surges by automatically scaling resources up or down as needed.

Fault Tolerance: Integrating redundancy mechanisms into our system to mitigate the impact of hardware or software failures, enhancing overall system resilience.

Cost Optimization: Utilizing AWS services efficiently to control costs while preserving peak performance.

Benefits:

The project yields numerous benefits that translate into a superior website experience:

Improved Website Performance: Users experience faster load times and heightened responsiveness.

Enhanced Reliability: Downtime stemming from server failures is drastically reduced, bolstering website dependability.

Cost Efficiency: The "pay as you go" model ensures that you are billed only for the resources you consume, coupled with the flexibility to scale as needed.

Scalability: Easily accommodate traffic growth without manual intervention, adapting effortlessly to varying demands.

Simplified Management: AWS provides a suite of tools for monitoring, scaling, and managing your infrastructure, streamlining maintenance efforts.

2. METHODOLOGY

The development of the Load Balancer project adheres to a structured and iterative methodology to ensure the successful implementation of efficient load balancing while aligning with the core project objectives. The methodology consists of distinct phases, including requirements analysis, design, implementation, testing, and deployment:

2.1. Requirements Analysis

The project initiation starts with a comprehensive analysis of the load balancing requirements and expectations.

Initial discussions with stakeholders, including network administrators and system architects, are conducted to define essential features and functionalities necessary for effective load balancing. Feedback sessions are organized to gather insights, identify potential challenges, and refine the project scope, considering factors such as traffic distribution, fault tolerance, and scalability.

2.2. Design

Building upon the gathered requirements, a systematic system design is formulated. This phase aims to create a clear blueprint for the load balancing architecture.

The design emphasizes the interaction between various components, including load balancer algorithms, backend servers, and network configurations.

Particular attention is given to ensuring efficient traffic distribution, health monitoring, and redundancy mechanisms.

2.3. Implementation

The implementation phase focuses on translating the design specifications into a fully functional load balancing system.

Leveraging modern networking technologies, load balancing algorithms, and scripting languages, the system is developed to provide efficient traffic distribution.

Load balancer software or hardware is integrated into the network infrastructure, and configurations are fine-tuned for optimal performance.

Monitoring tools and scripts are implemented to track server health and performance.

2.4. Testing

Rigorous testing is essential to ensure the reliability and performance of the load balancing system.

Load testing is conducted to simulate various traffic conditions and confirm that the load balancer effectively distributes incoming requests.

Failover and redundancy testing is performed to validate the system's ability to handle server failures without service interruption.

Network security testing is conducted to ensure that the load balancer does not introduce vulnerabilities into the system.

2.5. Deployment

The deployment phase involves integrating the Load Balancer into the production network.

Load balancing rules and policies are configured to match the specific needs of the organization. Monitoring and alerting systems are set up to track the performance of the load balancer and backend servers.

2.6. Continuous Iteration and Improvement

Post-deployment, an iterative approach is adopted to incorporate user feedback, address performance optimizations, and enhance load balancing features.

Ongoing interactions with network administrators and users provide valuable insights to guide refinements and future enhancements.

Regular maintenance and updates ensure that the load balancing system continues to meet the evolving needs of the organization.

This methodology ensures a systematic and user-centric approach to the development of the Load Balancer project, resulting in a reliable, efficient, and scalable solution for optimizing traffic distribution while aligning with the core project objectives.

3.SYSTEM DESIGN / ARCHITECTURE

The Load Balancer project is designed as a cloud-native, serverless application engineered to efficiently distribute incoming web traffic across multiple servers or instances. The architecture follows a modular design approach, making extensive use of Amazon Web Services (AWS) to ensure reliable and responsive traffic distribution.

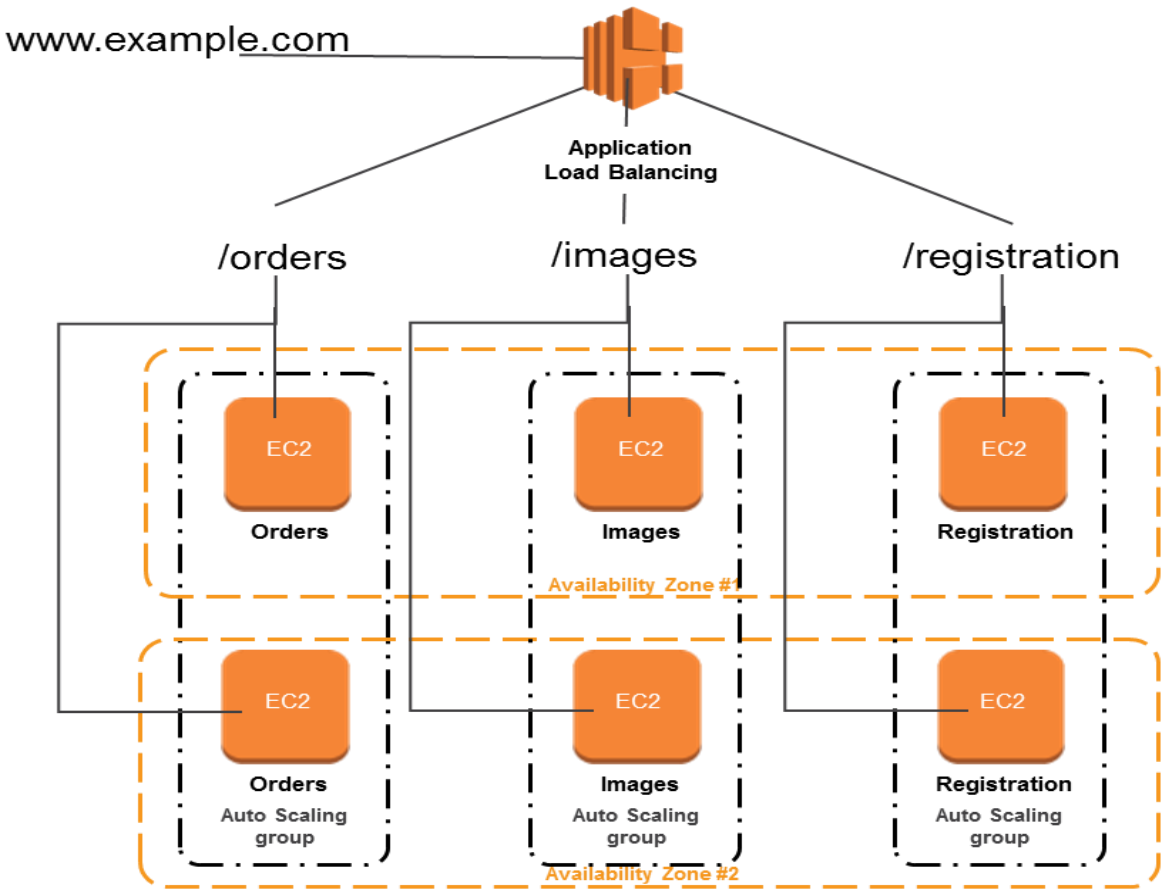
3.1. Load Balancing Layer

At the core of the system is the Load Balancing Layer, responsible for intelligently routing incoming traffic to backend servers.

Key components and aspects of this layer include:

AWS Elastic Load Balancer (ELB): ELB acts as the central load balancer component, distributing incoming HTTP/HTTPS requests across a pool of backend servers. It offers various load balancing algorithms, including round-robin and least connections, to ensure equitable distribution.

Auto Scaling: AWS Auto Scaling is employed to dynamically adjust the number of backend servers based on traffic load. This ensures that the system can handle varying levels of traffic without manual intervention.



3.2. Backend Server Layer

The Backend Server Layer comprises the servers or instances that host the website or web application. Key considerations for this layer include: **Amazon EC2 Instances**: Amazon Elastic Compute Cloud (EC2) instances are utilized as backend servers. They host the website/application and are grouped into an Auto Scaling Group. Health Checks: ELB periodically conducts health checks on backend servers to ensure they are responsive and healthy. Unhealthy instances are automatically replaced.

3.3. Scalability and Fault Tolerance

The architecture inherently supports scalability, with Auto Scaling Group instances automatically scaling up or down to accommodate traffic fluctuations.

Fault tolerance is assured through ELB's health checks and the automatic replacement of unhealthy instances. In case of instance failures, ELB routes traffic to healthy instances, minimizing disruptions.

3.4. Security

Security measures are implemented to ensure secure traffic distribution and protect against potential threats:

AWS Identity and Access Management (IAM): IAM roles are configured to grant only the necessary permissions to ELB for routing traffic to backend servers, enhancing security. SSL/TLS Certificates: SSL/TLS certificates are implemented to encrypt data in transit, securing user interactions with the website/application.

3.5. Monitoring and Logging

Monitoring and logging are essential for system visibility, performance analysis, and issue resolution:

Amazon CloudWatch: CloudWatch is used to monitor ELB's performance, track server health, and configure alarms for critical events.

Access Logs: Detailed access logs are maintained to track incoming requests and responses, aiding in debugging and security audits.

3.6. Deployment

The deployment phase involves configuring ELB, defining health checks, and setting up Auto Scaling policies. The architecture allows for seamless deployment of backend servers through Auto Scaling Groups.

3.7. Maintenance and Updates

The serverless nature of ELB simplifies maintenance and updates. Changes to routing configurations or backend servers can be made without impacting the entire system, ensuring minimal downtime and enhancing flexibility.

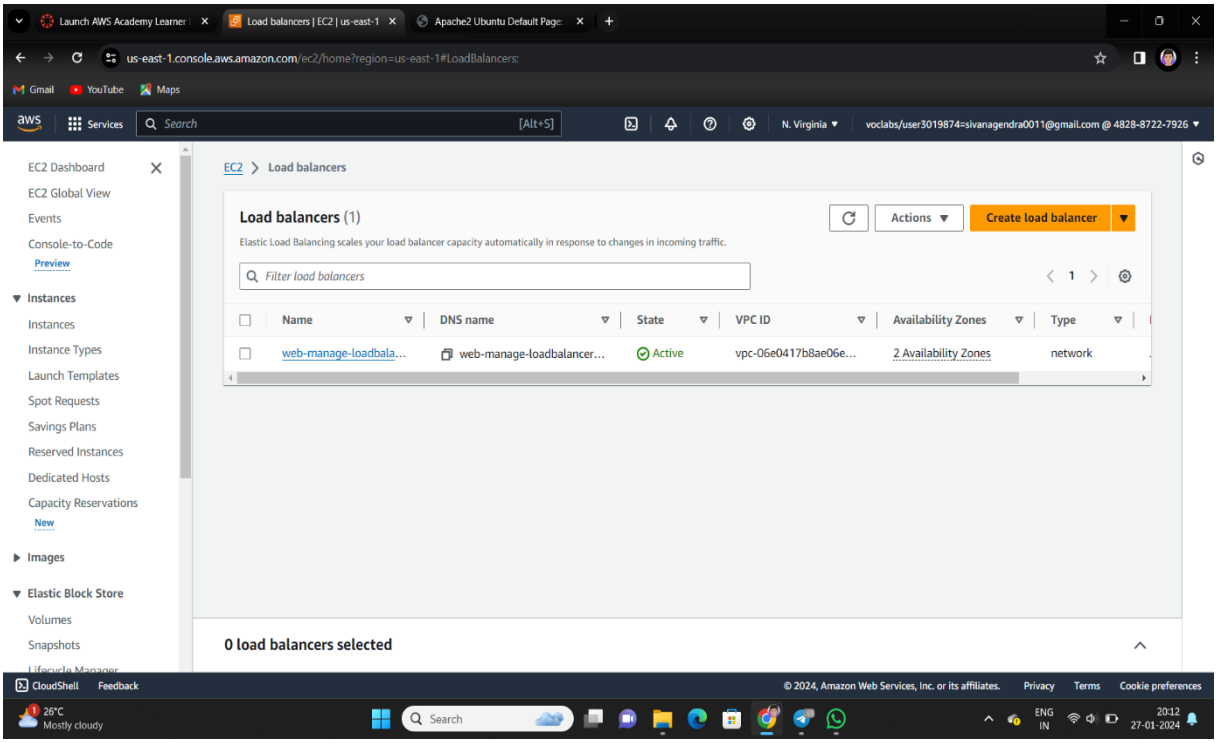
This architectural framework provides a scalable, efficient, and cost-effective solution for load balancing web traffic, aligning with the core objectives of the Load Balancer project.

4.IMPLEMENTATION

The implementation phase of the Load Balancer project involved translating the design specifications into a robust and functional system for distributing incoming web traffic effectively. This section provides an overview of the key components and technologies used during the implementation process.

4.1. Configuration of Load Balancer

The central component of the system is the configuration of the load balancer, typically Amazon Elastic Load Balancer (ELB) in an AWS environment. The following steps were executed to configure and manage the load balancer:



ELB Configuration: ELB was provisioned and configured with appropriate settings to distribute incoming HTTP/HTTPS requests across backend servers or instances.

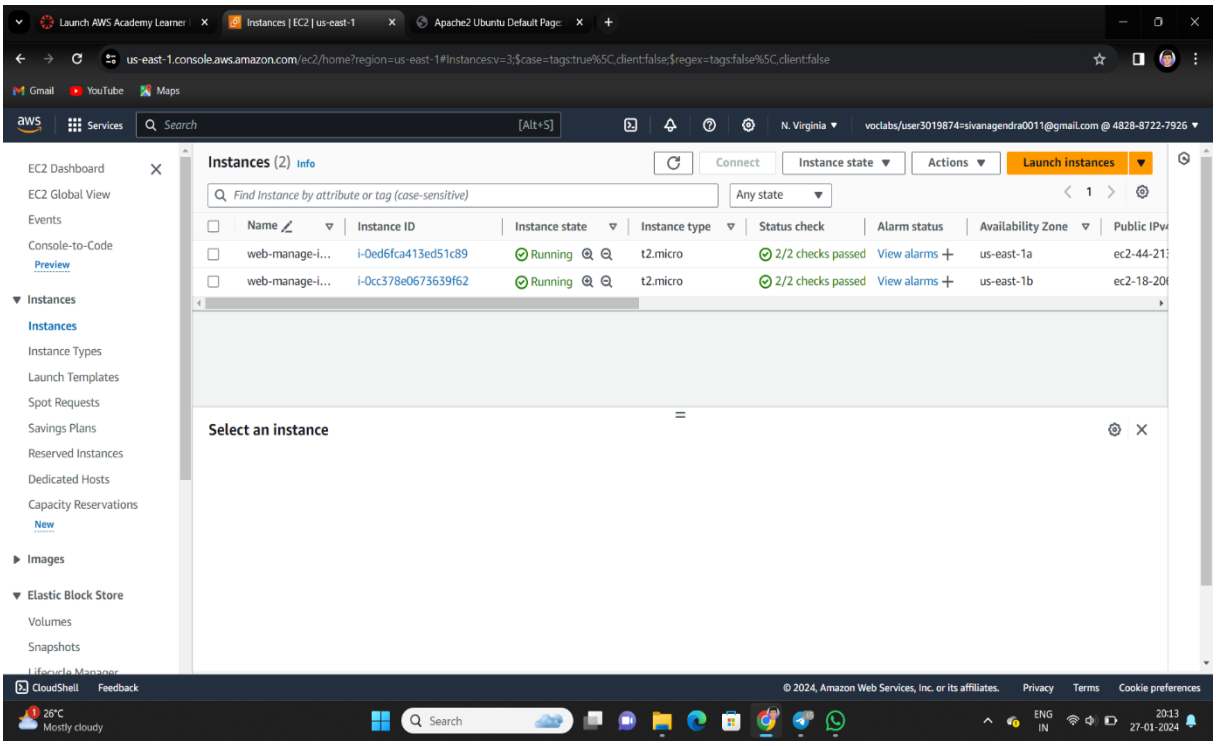
Listeners and Target Groups: ELB listeners and target groups were defined to specify the protocol and port for incoming traffic and to group backend servers.

Health Checks: Health checks were set up to monitor the status of backend servers, automatically routing traffic away from unhealthy instances.

4.2. Backend Server Setup

The Backend Server Setup consists of the servers or instances responsible for hosting the web application. Key considerations for this layer included:

Amazon EC2 Instances: Amazon Elastic Compute Cloud (EC2) instances were provisioned as backend servers, hosting the web application.

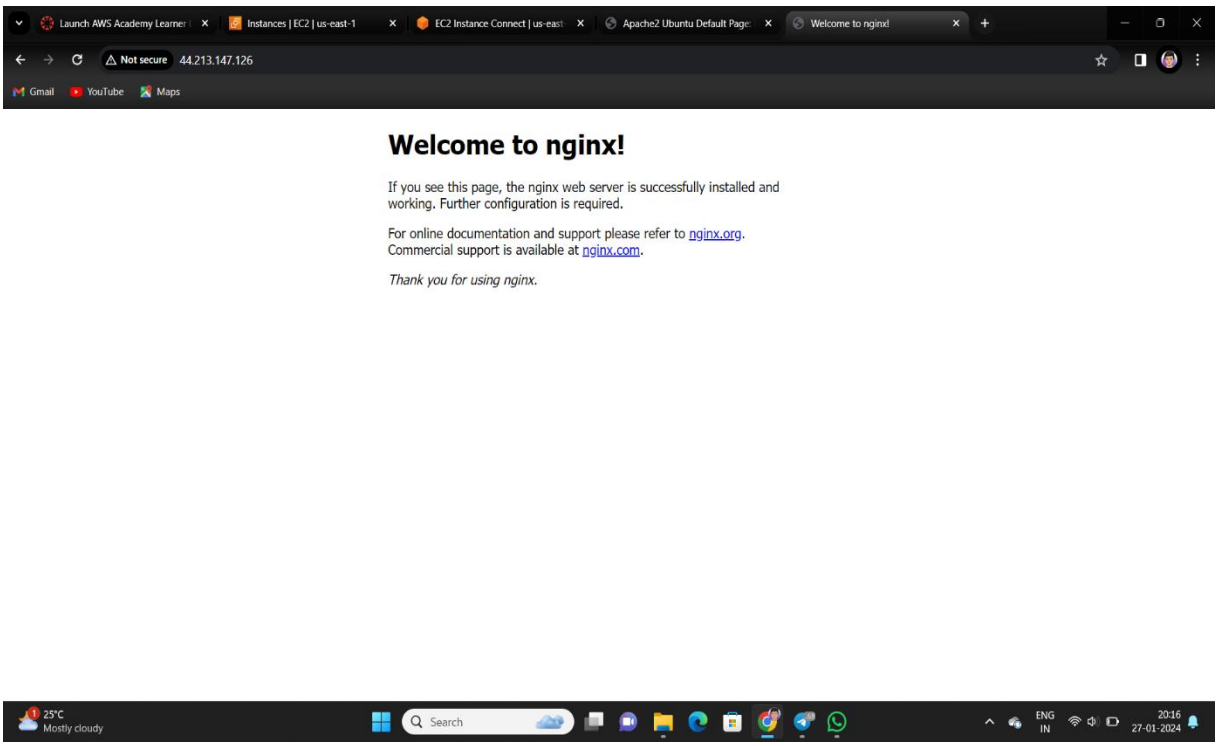


Server-1 (Linux Server):

Nginx installed into servre-1, by using following commands:

- ➔ **sudo su –** (to get into root user)
- ➔ **yum update -y** (for updating server)
- ➔ **sudo yum install nginx** (to install nginx to the server)

server output before applying load balancer:



A sample template of nginx that will be incorporated into the server when it was installed. It has its own IP address. It can be used to get the output.

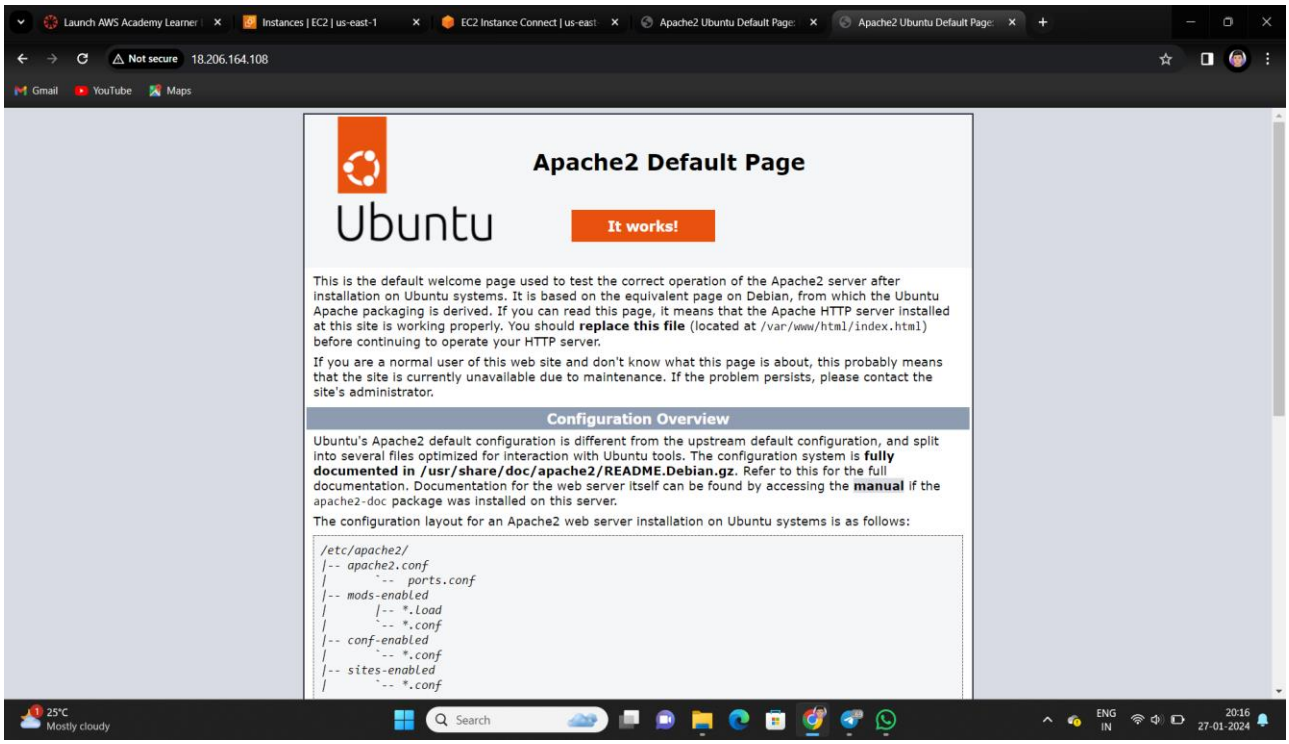
This server created in Subnet-1 of our own vpc.

Server-2 (Ubuntu Server):

Nginx installed into servre-1, by using following commands:

- ➔ **sudo su –** (to get into root user)
- ➔ **apt-get update -y** (for updating server)
- ➔ **apt-get install apache2 -y** (to install apache to the server)

server output before applying load balancer:

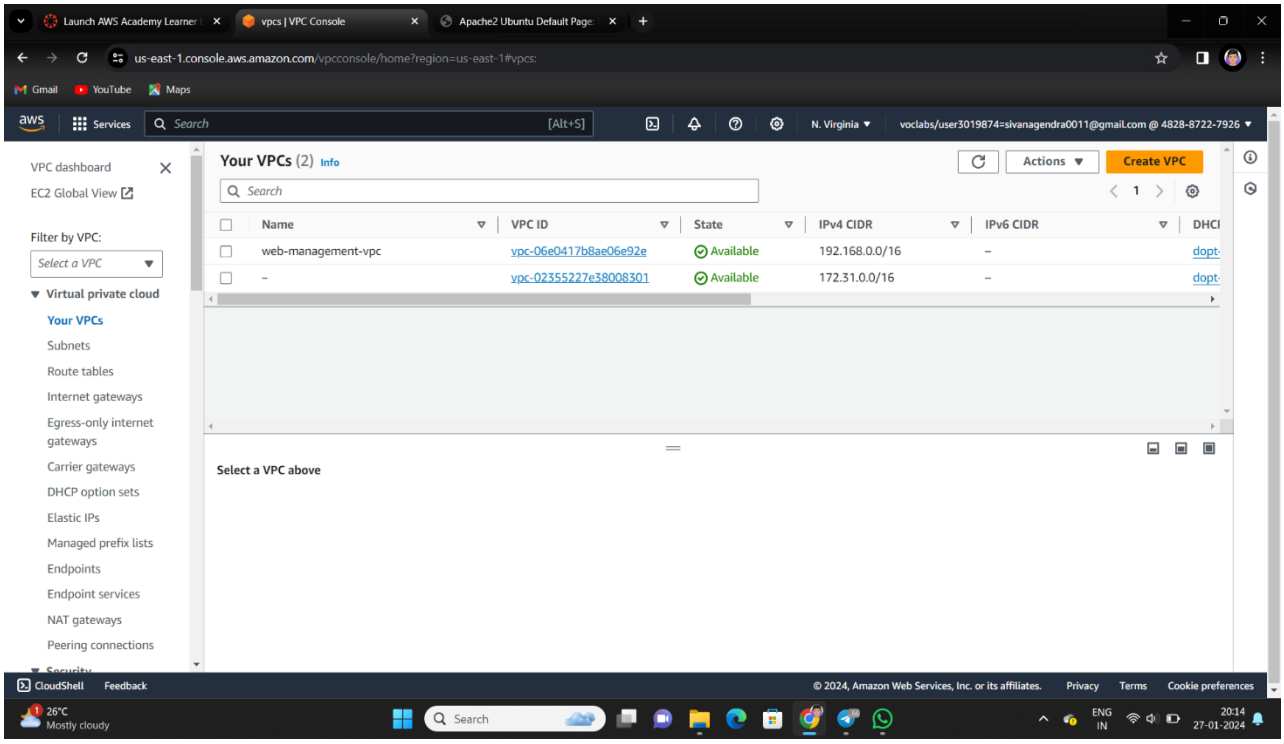


A sample template of apache that will be incorporated into the server when it was installed. It has its own IP address. It can be used to get the output.

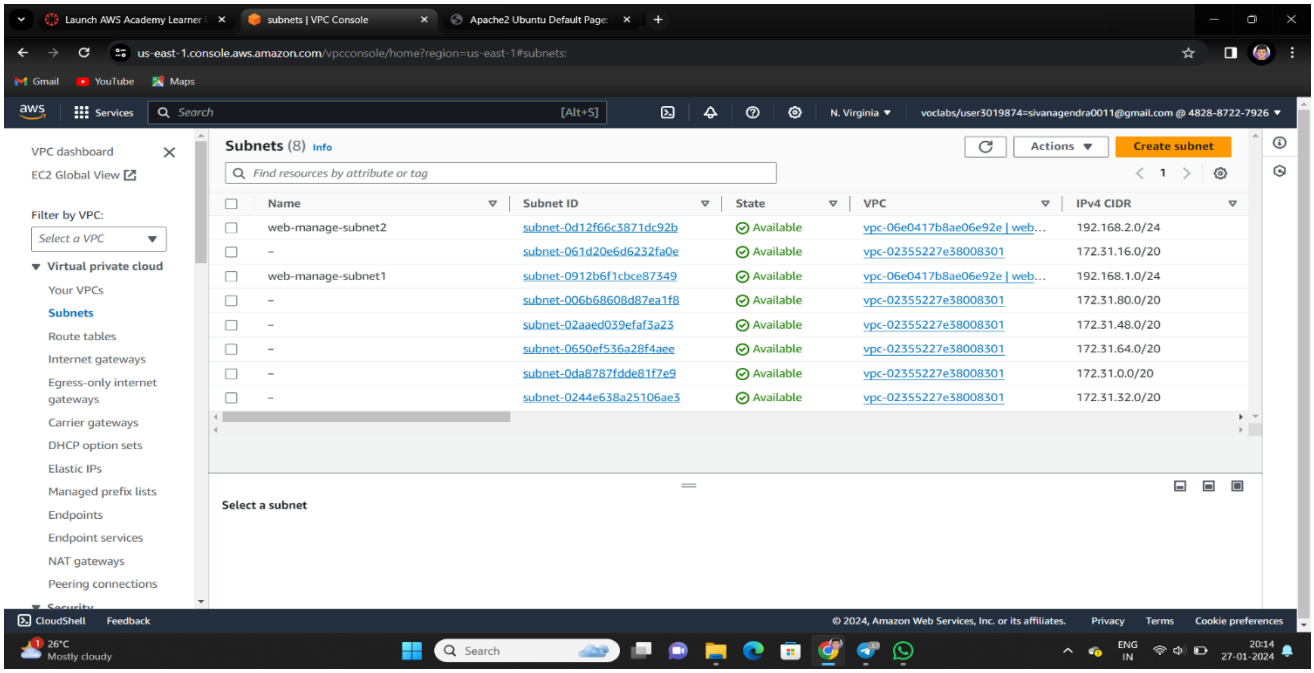
This server created in Subnet-2 of our own vpc.

Amazon VPC Service:

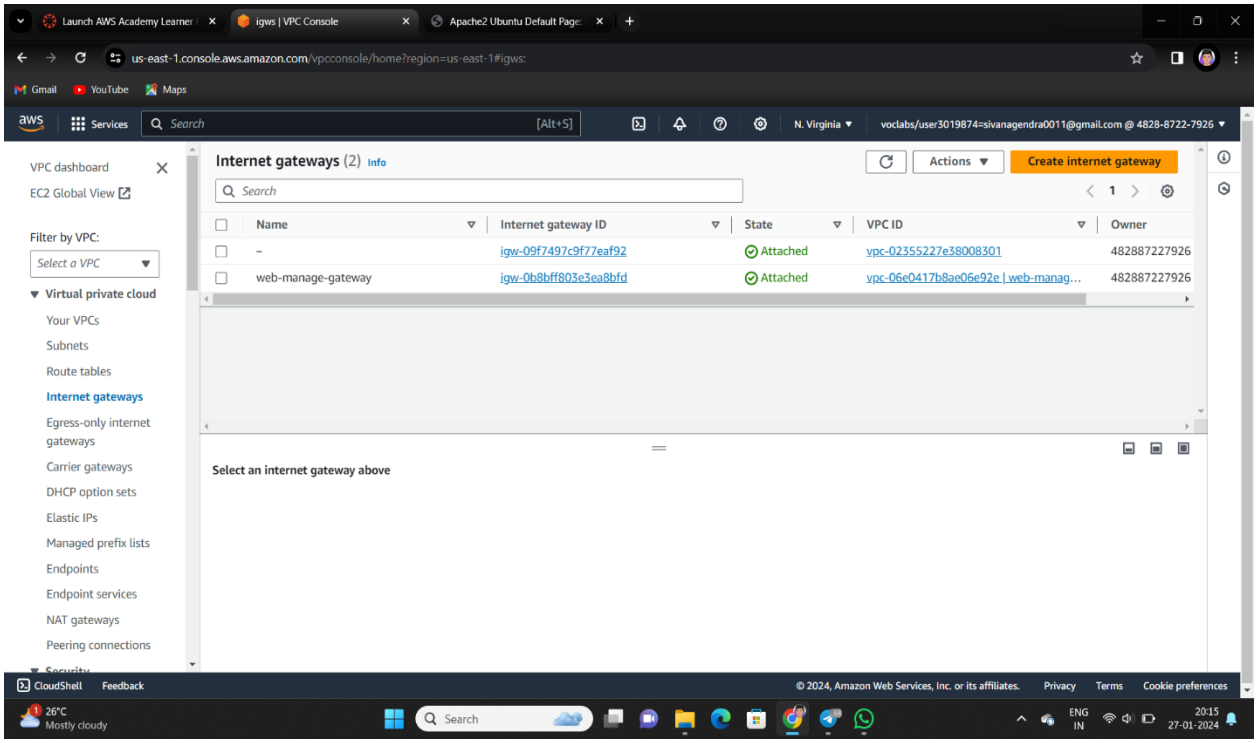
A VPC enables inbound and outbound filtering by providing security groups and network access control lists.



Subnets: Two subnets in two different regions in our vpc.

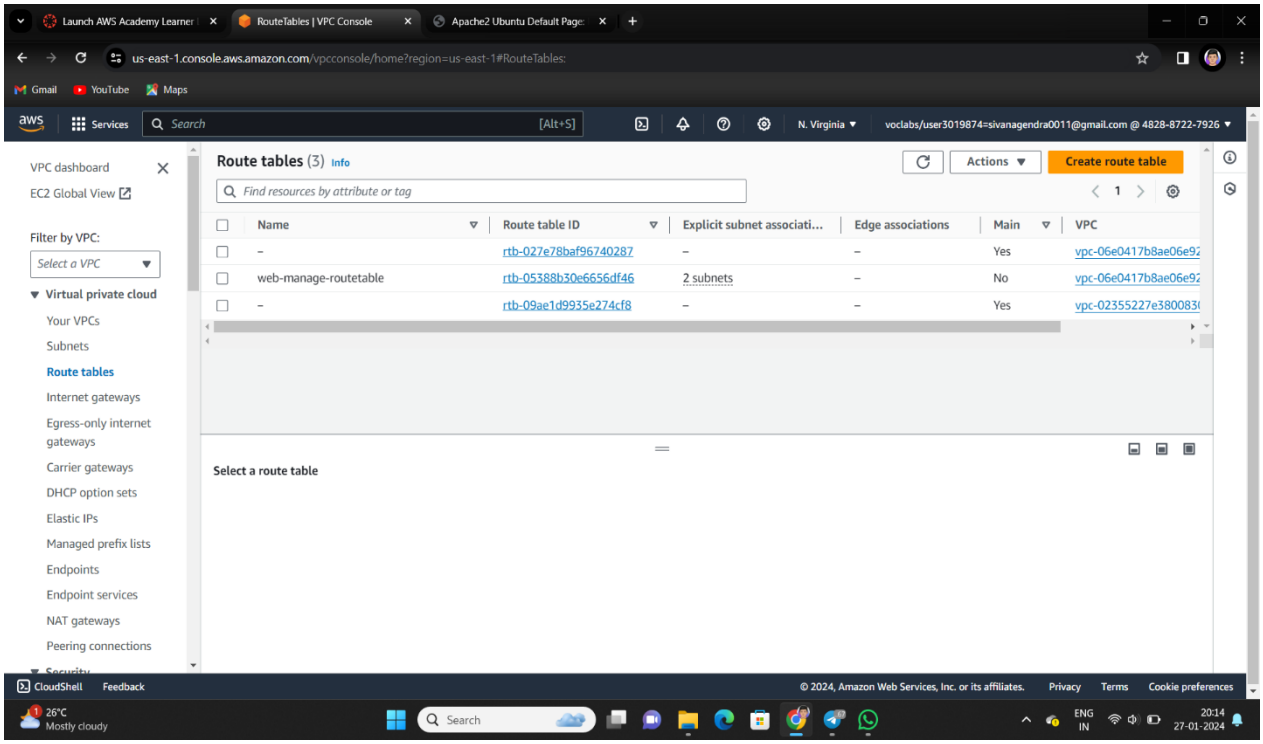


Internet Gateway:



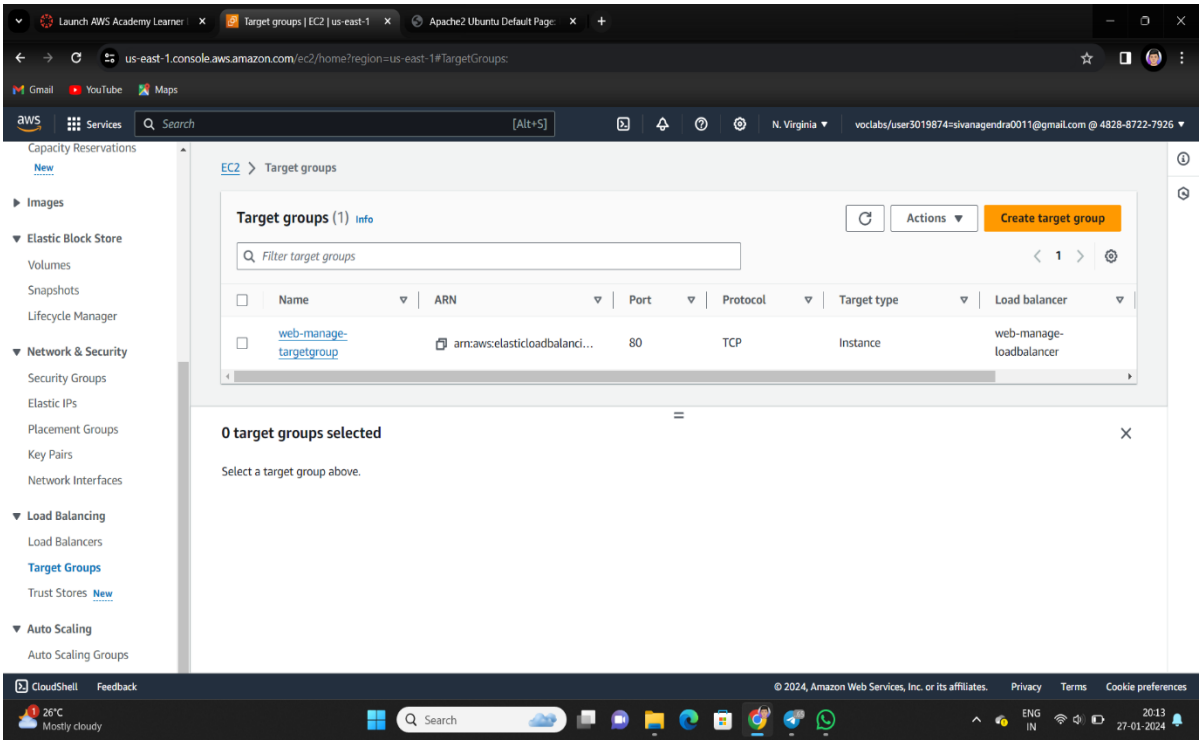
Rout Table:

Route table is used to load the traffic to the subnets.



Target-Group:

A load balancer routes requests to its targets using the protocol and port number that you specified when you created the target group.



These are all the requirements for managing load.

5.RESULTS

The implementation of the Load Balancer project has led to several noteworthy outcomes and achievements, enhancing the web infrastructure's performance and reliability. Here are the key results:

5.1. Efficient Traffic Distribution

The primary goal of the project was to ensure efficient traffic distribution across multiple backend servers. The implementation of the load balancer achieved this objective, allowing incoming web traffic to be evenly distributed, preventing overloading of individual servers, and optimizing response times.

Consider DNS name from the Load balancer that we created and paste in the new tab.

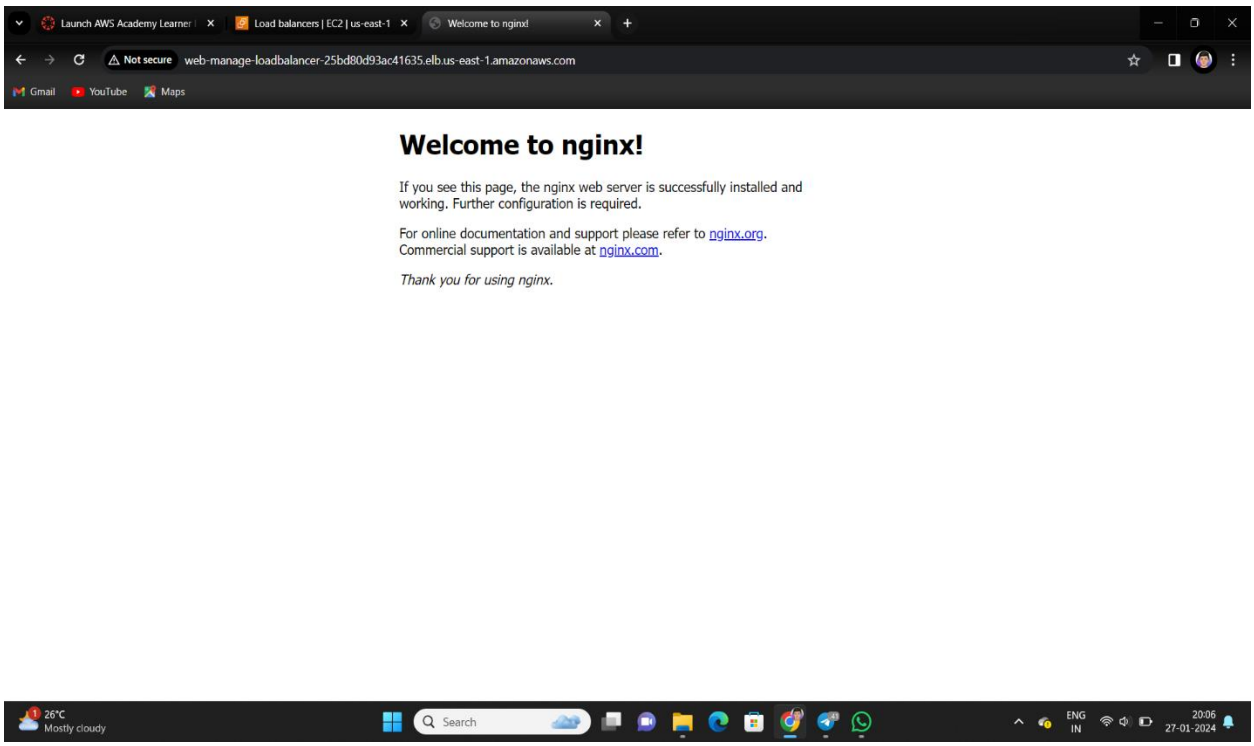
Then we will get output template. By applying load on the server continuously we will get another template. This is because of, heavy on load on single server leads to crashes, or it leads to delay of service. By applying load balancer on the server, we will get output template from any one of the server based on the load of the server.

In this way load balancer will perform traffic management in the server.

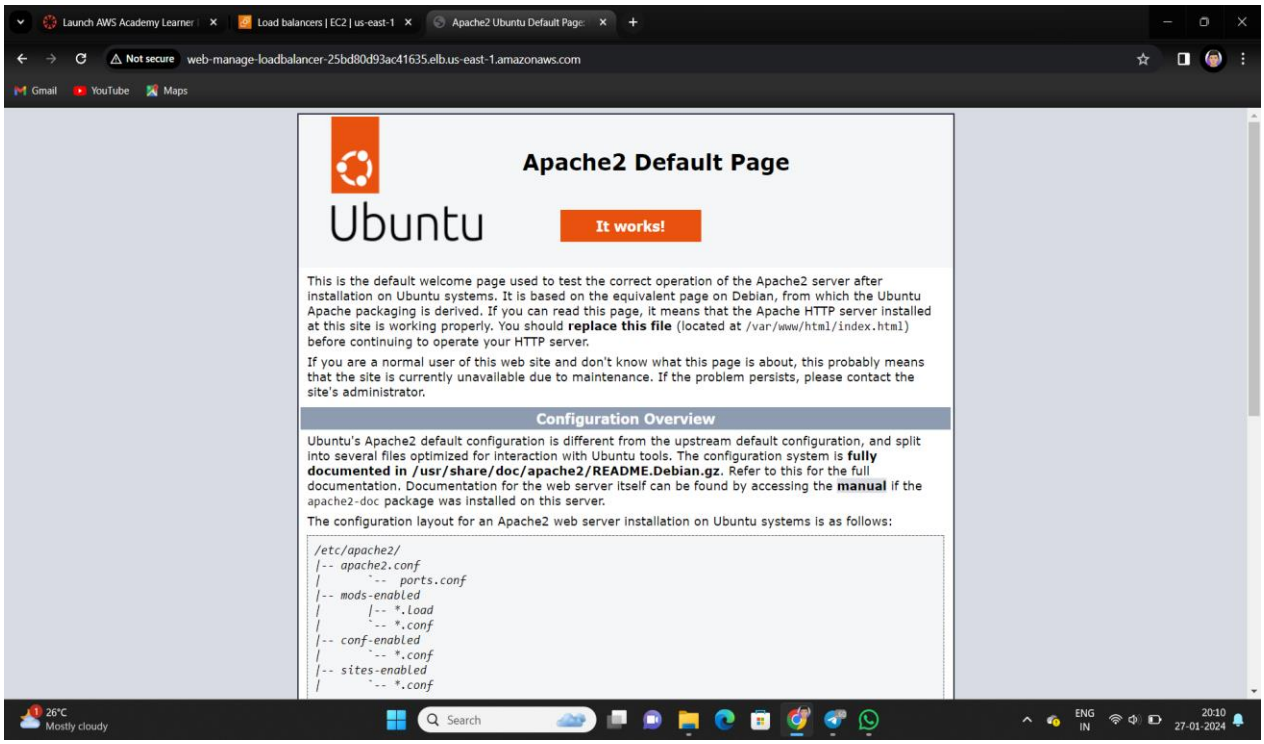
Before applying load:

Copy the DNS name from the load balancer that we have created and paste in the web browser.

I got the template from the server1 at which I installed nginx.



By applying high load we will get the response from the other server.



5.2. High Availability and Fault Tolerance

The Load Balancer project successfully established high availability and fault tolerance in the web infrastructure. With the auto-scaling capabilities of Amazon EC2 instances and health checks configured in the load balancer, the system can seamlessly handle increased loads and recover from server failures, minimizing downtime.

5.3. Enhanced Security

Security measures, including AWS IAM roles and SSL/TLS certificates, were implemented to enhance the security of the load balancer setup. This ensures secure communication between clients and the web application, safeguarding user data and interactions.

5.4. Scalability and Cost Optimization

The system's ability to dynamically adjust the number of backend servers based on traffic load ensures scalability without manual intervention. Additionally, cost optimization is achieved through efficient resource utilization, with resources scaling up or down as needed.

5.5. Simplified Management

The Load Balancer project simplifies the management of web infrastructure. Infrastructure as Code (IAC) tools and thorough documentation streamline deployment and maintenance processes, allowing for consistent and repeatable configurations.

5.6. Monitoring and Logging

Monitoring and logging using Amazon CloudWatch and access logs provide insights into system performance and access patterns. These tools aid in identifying issues, optimizing resource allocation, and ensuring the system's health.

5.7. Documentation and Automation

Comprehensive documentation of the load balancer configuration and infrastructure, combined with Infrastructure as Code (IAC), ensures that the system is well-documented and easy to manage. This enables future updates and enhancements to be executed smoothly.

6.CONCLUSION

In conclusion, the successful implementation of our website hosting and load balancing project is a pivotal achievement in bolstering the reliability and scalability of our online presence. This endeavor was driven by the essential need for a robust infrastructure capable of seamlessly handling fluctuating levels of web traffic while upholding consistent availability. The integration of a proficient load balancer stands as a cornerstone of this accomplishment, effectively distributing incoming traffic across multiple servers.

As a result, our website exhibits improved performance with reduced latency and faster response times, significantly enhancing the overall user experience. Moreover, the meticulous configuration of redundant server setups ensures high availability, mitigating the risk of potential disruptions. Looking forward, the extensible design of our infrastructure positions us well for future scalability, allowing for the seamless addition of servers to accommodate growing user demands and potential expansion.

Ultimately, this project not only enhances the reliability of our online platform but also contributes to cost-efficiency by optimizing resource allocation and reducing the need for substantial hardware investments.