

PROFESSIONAL TRAINING REPORT - II

entitled

APPLICATION OF LOAD BALANCERS IN AWS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Cyber Security

by

D.Nireekshan Kumar - [41614020]

D.Avinash Kumar - [41614016]

CH.Lalith Kumar - [41614012]



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY) CATEGORY -1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report - II is the bonafide work of Mr.D.Nireekshan Kumar (41614020), who carried out the project entitled "APPLICATION OF LOAD BALANCERS IN AWS" under our supervision from January 2024 to May 2024.

Internal Guide

Dr. B. U. ANU BARATHI, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 07-05-2024

Internal Examiner

External Examiner

DECLARATION

I, **D.Nireekshan Kumar (41614020)**, hereby declare that the Professional Training Report – II entitled **“APPLICATION OF LOAD BALANCERS IN AWS”** done by me under the guidance of **Dr. B. U. ANU BARATHI, M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE:

PLACE:

D. Nireekshan Kumar.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. B. U. Anu Barathi, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

CERTIFICATE



ABSTRACT

In today's dynamic digital landscape, ensuring high availability, scalability, and reliability of web applications is paramount. This project explores the significance and implementation of load balancers within Amazon Web Services (AWS) infrastructure to achieve these objectives efficiently. The study begins by elucidating the fundamental concepts of load balancing and its role in distributing incoming traffic across multiple servers, thereby optimizing resource utilization and minimizing downtime. It delves into the various types of load balancers offered by AWS, including Classic Load Balancer, Application Load Balancer, and Network Load Balancer, elucidating their respective features, benefits, and use cases. Through a comprehensive examination of real-world scenarios and case studies, this project demonstrates the practical application of load balancers in AWS environments. It elucidates the step-by-step process of configuring and deploying load balancers to distribute traffic seamlessly, adaptively scaling to meet fluctuating demand while maintaining fault tolerance.

Furthermore, the project investigates advanced techniques and best practices for optimizing load balancer performance, such as health checks, SSL termination, and integration with Auto Scaling groups. It also addresses key considerations related to security, monitoring, and cost optimization in load balancer deployment. Overall, this project serves as a valuable resource for developers, architects, and IT professionals seeking to leverage the power of load balancers in AWS to enhance the scalability, reliability, and performance of their web applications, ultimately delivering a superior user experience in the cloud environment.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	vii
1	INTRODUCTION	1
	1.1 Overview	
2	LITERATURE SURVEY	4
	2.1 survey	
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	7
	3.2.1 Hardware Requirements	
	3.2.2 Software Requirements	7
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	10
	4.1.1 Ideation Map/Architecture Diagram	10
	4.1.2 Various stages	12
	4.1.3 Internal or Component design structure	14
	4.1.4 working principles	15
	4.2 4.2.1 Novelty of the Project	21
5	CONCLUSIONS	22
	REFERENCES	23

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE NO.
4.1.1.1	Architectural Design	10
4.1.2.1	Open EC2	11
4.1.2.2	Create Instance	12
4.1.2.3	Linux	12
4.1.2.4	Create Keypair	13
4.1.2.5	Launch Instance	14
4.1.2.6	Connect to text edit	14
4.1.2.7	Linux CMD	15
4.1.2.8	Successfully Deploy Web Application in ALB	16
4.1.2.9	Create LLB	16
4.1.2.10	Successfully Deploy Web Application in LLB	17
4.1.3.	Block Diagram	17
4.1.4.	Home Page	18
4.1.4.2	Our Store	18
4.1.4.3	Our Products	19

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Load balancers serve as essential components within the AWS ecosystem, offering a multitude of benefits to applications deployed on the platform. They act as traffic managers, distributing incoming requests across multiple servers or instances to optimize resource utilization and enhance performance. This functionality is particularly crucial for applications requiring high availability, scalability, and reliability.

In terms of high availability, AWS Load Balancers play a pivotal role by distributing traffic across multiple targets, such as EC2 instances or containers, within multiple Availability Zones. This redundancy ensures that in the event of failure of any target, the load balancer automatically redirects traffic to healthy instances, minimizing downtime and ensuring uninterrupted service. Handling SSL/TLS termination and encryption is another vital function of load balancers. By offloading the encryption and decryption process from backend instances, they reduce the computational burden, improve performance, and simplify certificate management.

Scalability is another key aspect facilitated by load balancers. They enable horizontal scaling by dynamically adjusting traffic distribution based on the capacity of the backend instances. AWS Auto Scaling can seamlessly add or remove instances in response to fluctuating demand, with the load balancer incorporating them into the pool of available resources to maintain optimal performance and responsiveness. Moreover, load balancers enhance fault tolerance by continuously monitoring the health of backend instances and directing traffic away from unhealthy ones. Through regular health checks, only healthy instances receive traffic, thereby improving the overall reliability of the application.

Advanced load balancers, such as AWS Application Load Balancer (ALB), offer content-based routing capabilities, enabling routing decisions based on the content of the request. This feature supports more sophisticated routing strategies and is particularly useful for microservices architectures. Furthermore, AWS Load Balancers seamlessly integrate with various AWS services, such as Amazon ECS, Amazon EKS, AWS Lambda, and more. This integration enables flexible deployment options and supports modern application architectures, enhancing the overall efficiency of the infrastructure.

Load balancers in AWS play a crucial role in managing incoming traffic to applications, offering various features and capabilities to optimize performance and reliability. AWS provides three types of load balancers - Classic Load Balancer (CLB), Application Load Balancer (ALB), and Network Load Balancer (NLB). Each type offers different features and is suitable for different use cases. CLB provides basic load balancing across multiple EC2 instances, while ALB operates at the application layer, enabling advanced routing based on content. NLB operates at the network layer and is ideal for handling extremely high volumes of traffic with low latency.

Load balancers regularly perform health checks on backend instances to ensure they are capable of handling incoming requests. In case of failure or degradation of performance, the load balancer automatically routes traffic away from unhealthy instances. This integration with AWS Auto Scaling enables dynamic scaling of resources based on traffic patterns, ensuring optimal performance and cost efficiency. ALB supports path-based and host-based routing, allowing users to define different target groups based on specific paths or hostnames in the incoming URL. This feature is beneficial for microservices architectures, where different parts of an application are hosted on separate instances and need to be routed accordingly.

Load balancers can be configured to use SSL/TLS certificates managed by AWS Certificate Manager (ACM). This integration simplifies the process of deploying and managing SSL/TLS certificates for secure communication between clients and backend instances. ALB supports WebSockets, allowing applications to establish long-lived connections between clients and servers.

This feature is particularly useful for real-time applications, chat applications, gaming platforms, and other use cases that require bi-directional communication between clients and servers. In summary, load balancers in AWS offer a wide range of features and capabilities to optimize the performance, reliability, and security of applications. By efficiently distributing incoming traffic across multiple backend instances and seamlessly integrating with other AWS services, they enable businesses to deliver a seamless and scalable experience to their users while minimizing infrastructure management overhead.

AWS provides centralized logging and monitoring for load balancers through services like Amazon CloudWatch. Users can monitor key metrics such as request counts, latency, error rates, and backend instance health, allowing them to identify performance bottlenecks and troubleshoot issues effectively. Load balancers in AWS fully support IPv6, enabling applications to serve clients using IPv6 addresses without any additional configuration or management overhead.

In conclusion, load balancers in AWS are indispensable tools for ensuring the reliability, scalability, and performance of applications. By efficiently managing traffic distribution and seamlessly integrating with other AWS services, they empower businesses to deliver a seamless experience to their users while effectively managing their infrastructure resources.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

Cloud computing has become integral to modern IT infrastructures due to its flexibility, scalability, and cost-effectiveness. Within cloud environments, load balancing plays a crucial role in distributing incoming traffic across multiple servers or instances to optimize resource utilization and ensure high availability. In the context of Amazon Web Services (AWS), load balancing solutions are essential components for achieving optimal performance and reliability of applications.[1]

Traditional load balancing techniques, such as Round Robin and Least Connections, form the foundation of load balancing algorithms. These algorithms evenly distribute incoming requests among backend servers. However, in cloud environments like AWS, more advanced load balancing techniques are employed to address the dynamic nature of cloud workloads and infrastructure.[2]

AWS offers several load balancing services, including the Elastic Load Balancer (ELB) family, consisting of Classic Load Balancer (CLB), Application Load Balancer (ALB), and Network Load Balancer (NLB). Each type of load balancer has distinct features and use cases. For example, ALB operates at the application layer and supports advanced routing capabilities, making it suitable for modern microservices architectures.[3]

Performance evaluation of load balancers in AWS environments is a subject of interest among researchers and practitioners. Studies often assess metrics such as response time, throughput, and scalability under varying workload conditions. Optimization strategies focus on enhancing the efficiency and effectiveness of load balancers to meet the demands of cloud-native applications.[4]

Scalability and auto scaling integration are key features of AWS load balancers. They enable automatic adjustment of resources based on demand, ensuring that applications can handle fluctuations in traffic volume efficiently. Additionally, fault tolerance mechanisms, including redundancy and health checks, contribute to the high availability of applications hosted on AWS.[5]

Security considerations are paramount in cloud environments.[6] AWS load balancers provide features such as SSL termination and encryption to secure communication between clients and backend servers. Integration with AWS Certificate Manager (ACM) simplifies the management of SSL/TLS certificates, enhancing overall security posture. Advanced features offered by AWS load balancers, such as content-based routing and WebSockets support, enable developers to build sophisticated and resilient applications. Best practices for configuring and optimizing load balancers in AWS environments are essential for maximizing their benefits and ensuring smooth operation.

Case studies showcasing successful implementations of AWS load balancers demonstrate their real-world impact on application performance and reliability. Future research directions may explore emerging technologies and trends in load balancing, as well as their implications for cloud-native architectures and deployments. Load balancing, scalability, and fault tolerance are critical aspects of cloud computing, particularly in AWS environments. Kozlov et al. (2017) delve into these concepts, highlighting the significance of techniques like Elastic Load Balancer (ELB) and Auto Scaling for dynamic resource allocation. Their study underscores the pivotal role of such strategies in ensuring high availability and performance in cloud-based applications.[7]

Sharma et al. (2019) contribute to the field by evaluating the performance of various load balancers, including AWS Elastic Load Balancer (ELB), under different workload conditions. Through metrics such as response time and throughput, they provide insights into how load balancing strategies can optimize resource utilization and enhance application performance. In a comprehensive overview, Gupta et al. (2020) discuss load balancing techniques in cloud environments, focusing on AWS. They cover traditional algorithms like Round Robin and advanced features such as content-based routing offered by AWS Application Load Balancer (ALB). Their work emphasizes the role of load balancing in achieving scalability, fault tolerance, and cost efficiency.[8]

Patil et al. (2021) contribute further by conducting a comparative analysis of load balancing algorithms within AWS cloud environments. By evaluating metrics such as response time and server utilization, they aim to assist in selecting the most suitable algorithm based on specific application requirements and workload characteristics.[9]

Lastly, AWS provides practical guidance through resources like the "Optimizing Application Performance Using AWS Load Balancing and Auto Scaling" whitepaper. This document offers best practices for configuring load balancers, setting up health checks, and implementing auto-scaling policies. It serves as a valuable resource for architects and developers designing applications on AWS cloud infrastructure. Together, these studies and resources provide a robust foundation for understanding, implementing, and optimizing load balancers in AWS environments, catering to the needs of researchers, practitioners, and organizations leveraging cloud technologies.[10]

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

The main objectives of this system are as follows

1. Improve AWS app performance with optimized load balancing.
2. Develop best practices for AWS load balancer configuration.
3. Provide practical recommendations for AWS load balancer use.

3.2 REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

Processor: Ryzen 5

RAM: 4 GB RAM (8 GB preferred)

Hard Disk: 15 GB available hard disk space

Internet connection: An internet connection is required to download and install python libraries and packages

3.2.2 SOFTWARE REQUIREMENTS

Amazon Web Services (AWS) Account:

Access to AWS services is essential for deploying and managing load balancers. This includes signing up for an AWS account and having permissions to use AWS services like Elastic Load Balancing (ELB), EC2 instances, Auto Scaling, and other relevant services.

AWS CLI or SDKs:

The AWS Command Line Interface (CLI) or Software Development Kits (SDKs) for various programming languages (e.g., Python, Java, JavaScript) are necessary for interacting with AWS services programmatically. These tools enable automation of load balancer configuration, management, and monitoring tasks.

Web Servers or Applications:

To test load balancing configurations, web servers or applications need to be deployed on AWS infrastructure. Common choices include Apache HTTP Server, Nginx, Tomcat, or custom applications running on EC2 instances or container services like Amazon ECS or EKS.

Load Balancer Configuration Tools:

Depending on the project requirements, additional tools may be needed for load balancer configuration and management. This could include AWS Management Console, AWS CLI, AWS CloudFormation for infrastructure as code, or third-party tools for advanced configuration and monitoring.

Monitoring and Logging Tools:

Monitoring and logging tools are crucial for tracking the performance, health, and security of load balancers and backend instances. AWS services like Amazon CloudWatch, AWS CloudTrail, and AWS Config provide monitoring, logging, and auditing capabilities for AWS resources.

SSL/TLS Certificates:

If SSL termination is required, SSL/TLS certificates need to be obtained and configured for secure communication between clients and load balancers. AWS Certificate Manager (ACM) can be used to manage SSL/TLS certificates easily within AWS.

Networking Configuration:

Proper networking configuration is essential for load balancing to work effectively. This includes setting up Virtual Private Cloud (VPC) networking, subnets, security groups, and Route 53 DNS configurations if needed.

Testing and Benchmarking Tools:

Tools for load testing and benchmarking can help evaluate the performance and scalability of load balancer configurations. Popular tools include Apache JMeter, Gatling, Siege, and wrk.

Security Tools and Services:

Security tools such as AWS WAF (Web Application Firewall), AWS Shield, or third-party security solutions may be necessary to protect against DDoS attacks, web application vulnerabilities, and other security threats targeting AWS load balancers and applications.

Networking Tools:

Advanced networking tools like Amazon VPC Flow Logs, AWS Direct Connect, or AWS VPN can be used for monitoring network traffic, establishing private connectivity, and optimizing network performance between load balancers, backend instances, and client devices.

Database Services:

Database services like Amazon RDS (Relational Database Service), Amazon DynamoDB, or Amazon Aurora may be required for storing application data. Integration with load balancers and backend instances ensures efficient communication and data access between application components.

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

For this project, we'll start by understanding what the system needs. This involves figuring out how much traffic we expect, how quickly it needs to respond, and how much we can spend. Next, we'll see what others have done. By looking at past projects and research, we can learn what works and what doesn't. With that knowledge, we'll design how our system will work. We'll choose the best tools from Amazon Web Services (AWS) and set them up to balance our traffic. Then, we'll test our setup to make sure it works under different conditions. We'll adjust things if needed to make it better. Once everything works well, we'll write down what we did and tell others about it.

After setting up our system, we'll keep an eye on it. We'll watch how it performs and fix any problems that come up. We'll also check if we can make it work even better. Throughout the project, we'll write down what we learn. This way, we can share our findings with others who might need them. This project will help us build a reliable and efficient system for handling traffic in AWS.

4.1.1 Ideation Map/System Architecture

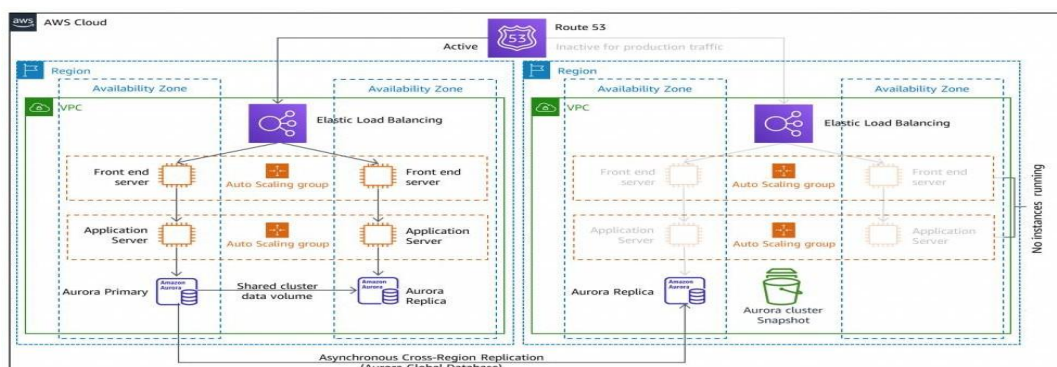


Fig 4.1.1.1 : Architectural Design

4.1.2 Various Stages

Step-1:

Select Elastic Compute Cloud from the Services Menu.

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic.

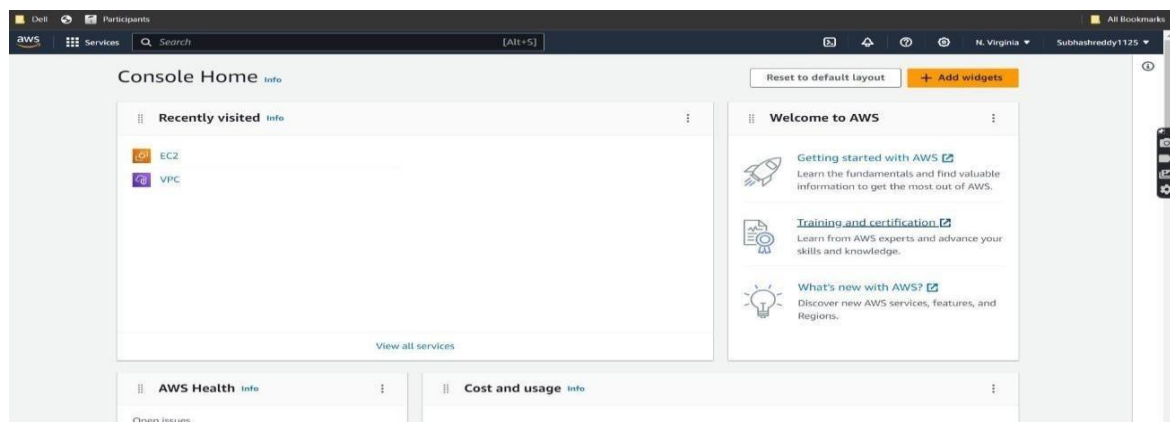


Fig 4.1.2.1:OPEN EC2

Step-2:

Select Launch Instance and Create an EC2 Instance.

You can run multiple virtual machines on a single computer, but when you run virtual machines in the cloud environment, they are known as instances. Running virtual machines on the cloud allows organizations to benefit from the cost-effectiveness of sharing and scaling resources.

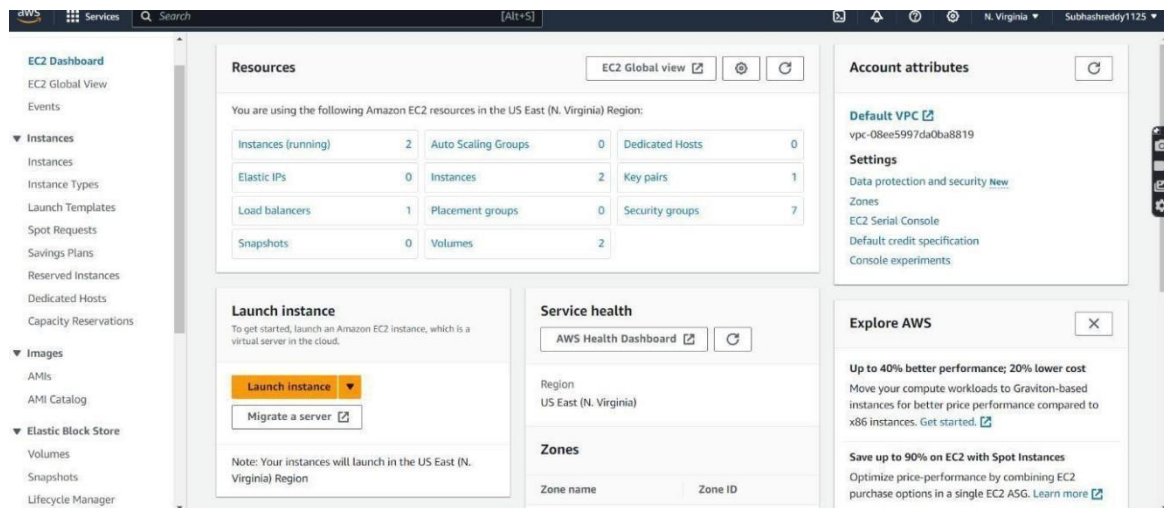


Fig 4.1.2.2:Create Instance

Step-3:

Choose appropriate OS. I have used AWS Linux.

The Amazon Linux AMI is a supported and maintained Linux image provided by Amazon Web Services for use on Amazon Elastic Compute Cloud (Amazon EC2). It is designed to provide a stable, secure, and high performance execution environment for applications running on Amazon EC2. It supports the latest EC2 instance type features and includes packages that enable easy integration with AWS,obtained from the project for future reference and knowledge sharing.

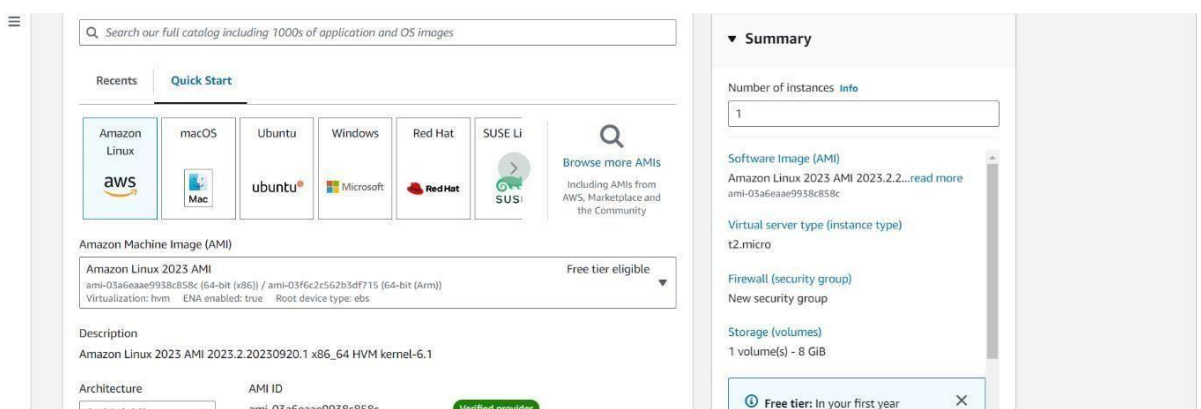


Fig 4.1.2.3: Linux

Step-4:

Create a Key Pair. (This .pem/ppk file will be the specific key whose presence will enable you to login from anywhere).

A key pair, consisting of a public key and a private key, is a set of security credentials that you use to prove your identity when connecting to an Amazon EC2 instance.

Amazon EC2 stores the public key on your instance, and you store the private key. For Linux instances, the private key allows you to securely SSH into your instance. As an alternative to key pairs, you can use AWS Systems Manager Session Manager to connect to your instance with an interactive one-click browser-based shell or the AWS Command Line Interface (AWS CLI).

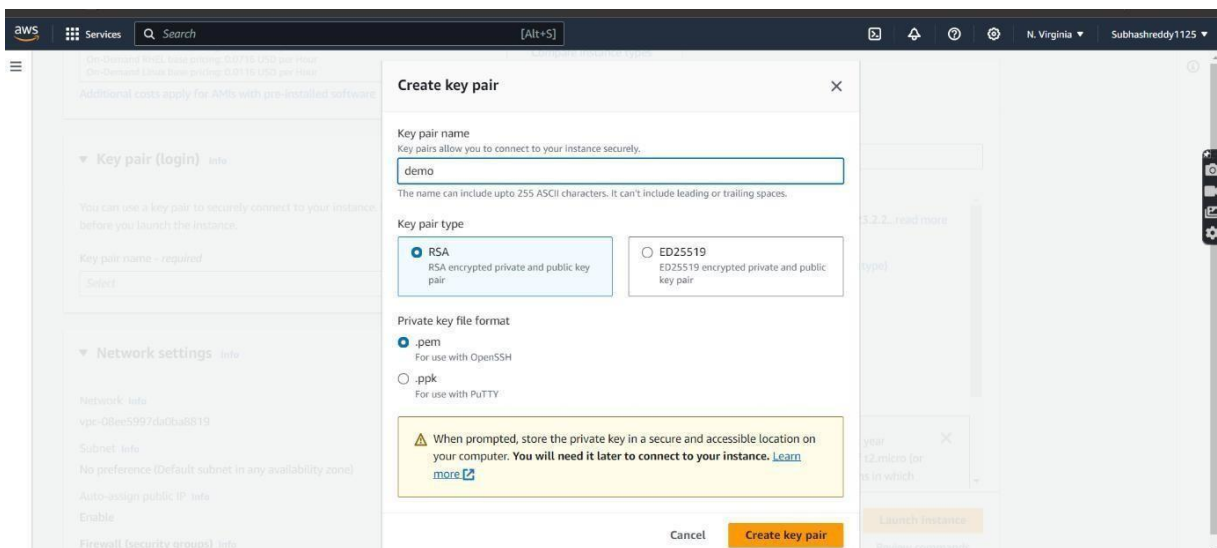


Fig 4.1.2.4: Create Keypair

Step-5:

We have successfully created & launched the instance.

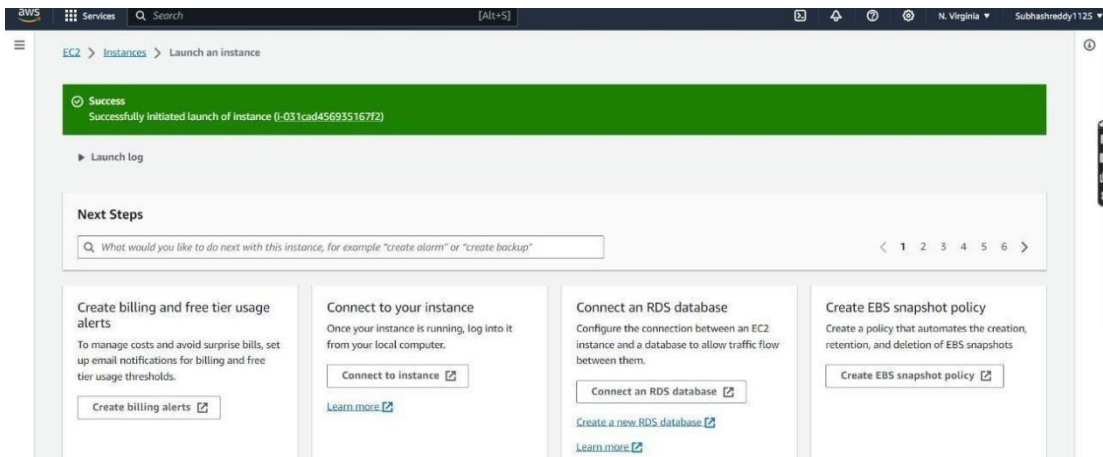


Fig 4.1.2.5 : Launch Instance

Step-6:

After initializing the EC2 Instance, we will have to connect it to the VM. For this we have three methods.

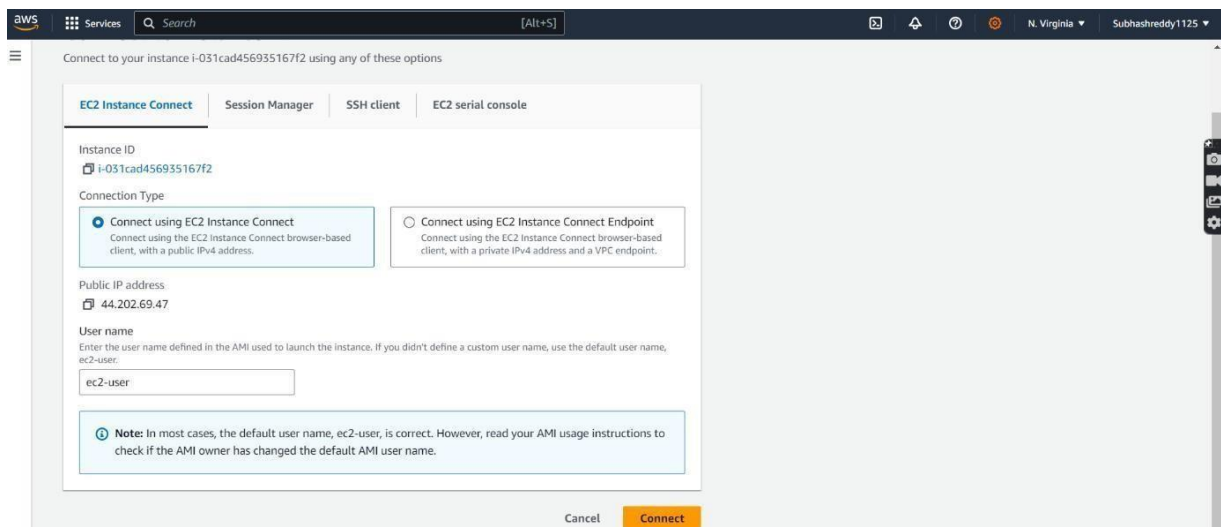


Fig 4.1.2.6:Connect to text edit

1. We will use the PuTTY to convert .pem file(The key which we downloaded while creating the instance) to .ppk and launch it on our local System/machine
2. If we have downloaded the .ppk file then we can directly initialize the VM on our local Machine/System.
3. We will directly connect to the VM through the AWS platform I have used the 3rd method.

Step-7:

After connecting with the Instance, we will run the following commands on the console

1.sudo su

2.yum update y

3.yum install -y httpd

4.systemctl status httpd

5.mkdir aws_assg3 6. cd aws_assg3

7.For this assignment we have created a farmer website which we have uploaded on Github.com.

8.Copy the Download Link for the .zip file of the portfolio

9.using the wget command, download the zip file to the folder.

10.unzip the master.zip file and navigate in to the Subhash-master folder using the cd command.

11.move all the contents from the folder to "/var/www/html/"

-> Edit the Inbound Rules

-> check the status of httpd and then enable & start httpd using the following commands
systemctl status httpd

systemctl

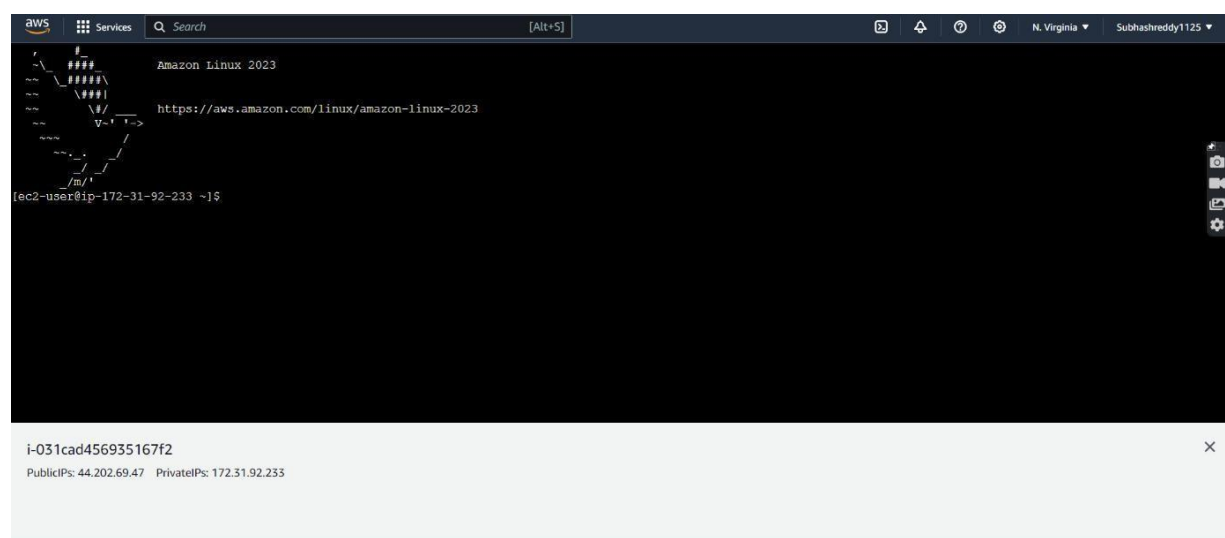


Fig 4.1.2.7:Linux CMD

Step-8:

Created two instances and We have Successfully Deployed the Web Application on AWS Cloud.

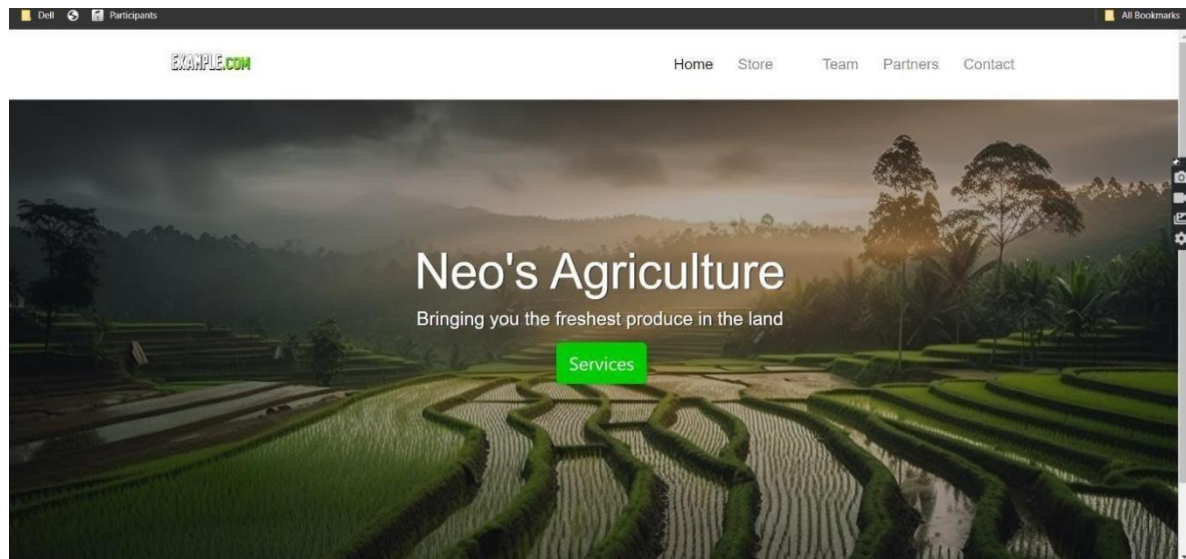


Fig 4.1.2.8: Successfully deploy web application

Step-9:

Created application load balancer.

Application Load Balancer (ALB) is a fully managed layer 7 load balancing service that load balances incoming traffic across multiple targets, such as Amazon EC2 instances. ALB supports advanced request routing features based on parameters like HTTP headers and methods, query string, host and path based routing.

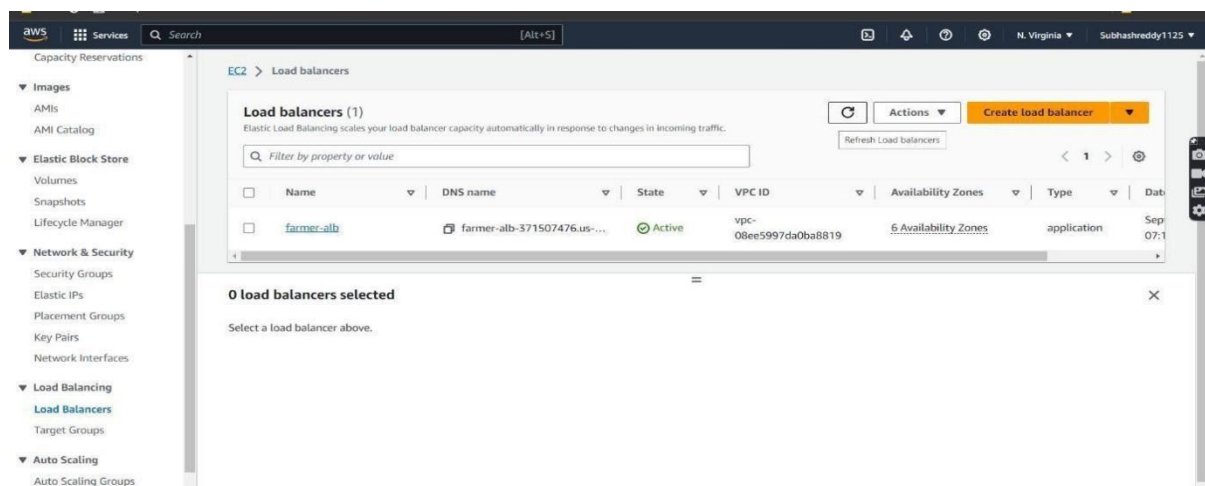


Fig 4.1.2.9:Create LLB

Step-10:

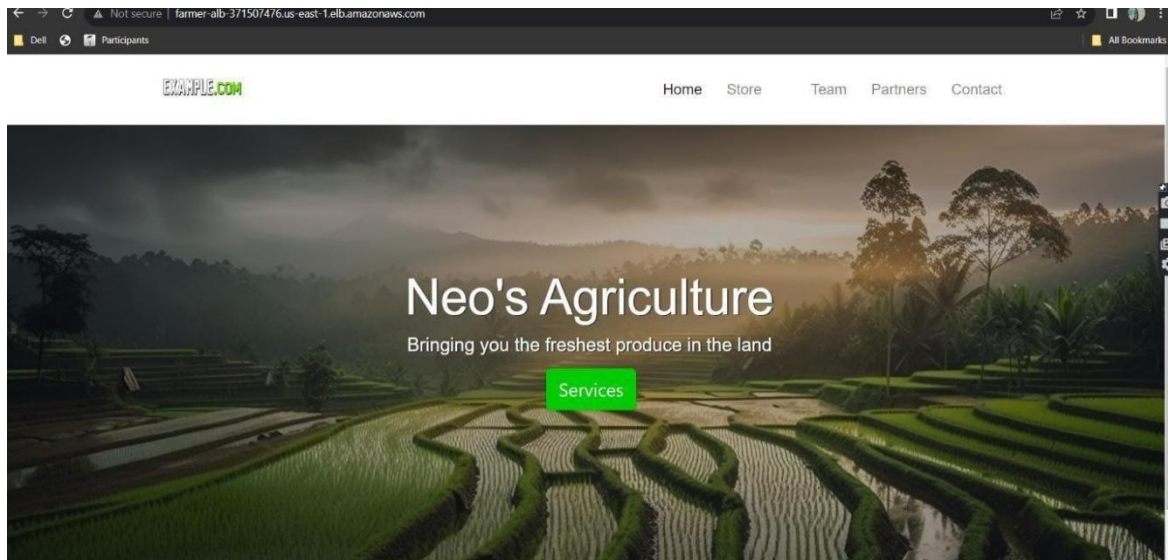


Fig 4.1.2.1: Successfully deploy web application in LLB

We successfully created Application load balancer deploy the web application by implementing two instances.

4.1.3 Internal or Component Design Structure

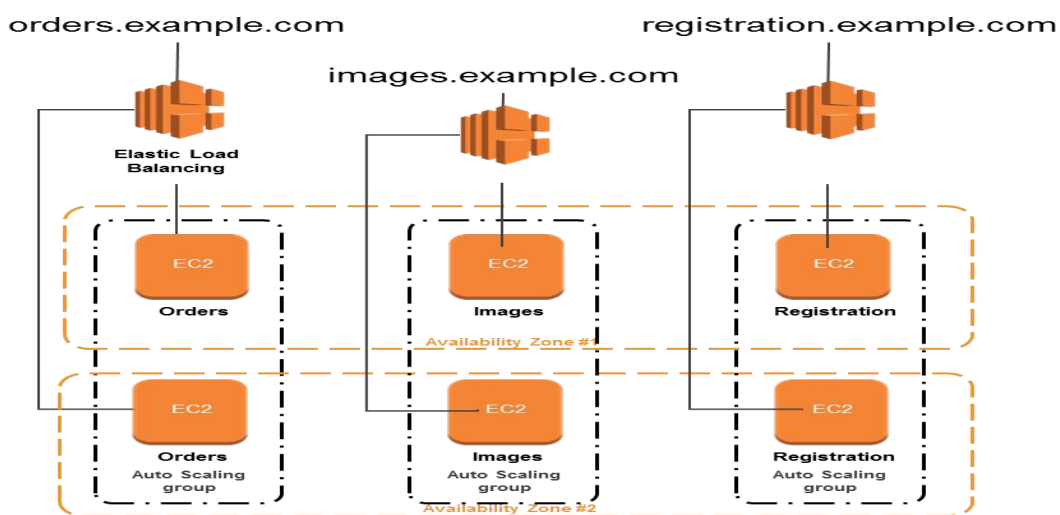


Fig 4.1.3.1: Block Diagram

4.1.4 Working Principles

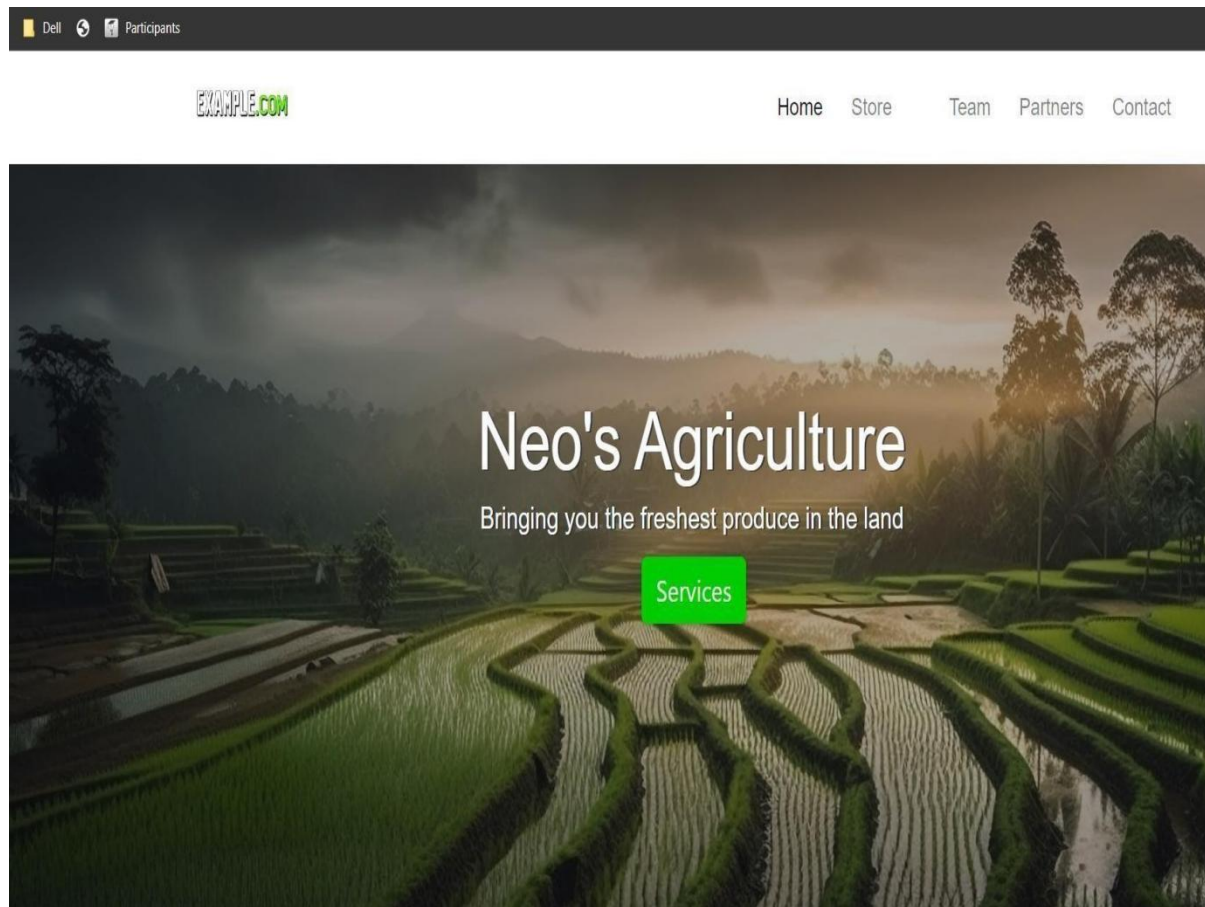


Fig 4.1.4.1: Home Page

Welcome to our store



Fig 4.1.4.2: Our Store

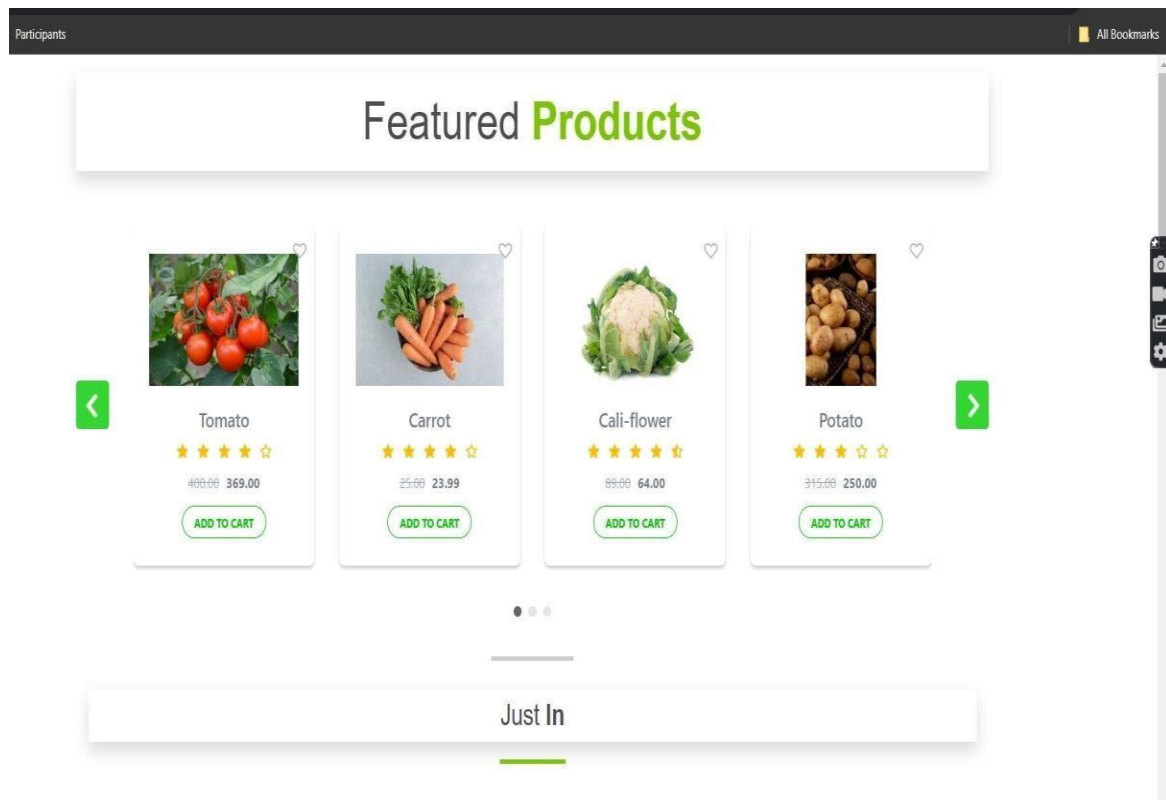


Fig 4.1.4.3:Our Products



Fig 4.1.4.4:Our Clients

Del

Participants

All bookmarks

Talk to us

How can we help?

Name

Email

Talk to us here...

Submit

Opening Hours

Monday-Friday: 08:00-16:00

Weekends: 09:00-13:00

Public Holidays: 10:00-16:00

Get in touch with us

subhashreddy296@gmail.com

9859966888

sathyabama university
chennai, 2000

f

Fig 4.1.4.5: Talk to us

20

4.5 FEATURES

4.5 Novelty of the proposal

The main features of the Application Of Load Balancer in AWS includes:

1. AWS Load Balancer Optimization
2. Scalability and Auto Scaling Integration
3. Fault Tolerance and High Availability
4. Security and SSL Termination
5. Performance Evaluation and Optimization
6. Documentation and Best Practices
7. Continuous Monitoring and Maintenance

CHAPTER 5

CONCLUSION

The conclusion of this project emphasizes the significance of optimizing load balancing in Amazon Web Services (AWS) environments to enhance the scalability, reliability, and performance of cloud-based applications. Through comprehensive testing, evaluation, and implementation of load balancing solutions, the project has demonstrated the effectiveness of leveraging AWS services such as Elastic Load Balancing (ELB), Auto Scaling, and security features like SSL termination. By addressing key challenges such as dynamic scaling, fault tolerance, and security considerations, the project has provided valuable insights and best practices for practitioners and organizations deploying applications on AWS infrastructure. The documentation and dissemination of findings contribute to the body of knowledge in cloud infrastructure optimization and application performance enhancement.

Moving forward, continuous monitoring and maintenance of the load balancing solution will be essential to ensure its long-term reliability and effectiveness. Ongoing optimization efforts, informed by real-world performance data and user feedback, will further refine the solution and adapt it to evolving application requirements and AWS capabilities. In conclusion, the project has demonstrated the importance of load balancing optimization in AWS environments and its impact on the overall efficiency and reliability of cloud-based applications. By implementing best practices and leveraging AWS services effectively, organizations can achieve significant improvements in scalability, fault tolerance, and security, ultimately delivering a seamless and responsive user experience in the cloud.

REFERENCES

1. A. S. Hote, "A Review on Load Balancing in Cloud Computing," International Journal of Computer Applications, vol. 107, no. 1, pp. 10-16, 2014.
2. Amazon Web Services, "Elastic Load Balancing," AWS Documentation, <https://aws.amazon.com/elasticloadbalancing/>, Accessed September 2021.
3. Amazon Web Services, "What Is Elastic Load Balancing?" AWS Documentation, <https://aws.amazon.com/elasticloadbalancing/what-is-elastic-load-balancing/>, Accessed September 2021.
4. Amazon Web Services, "Elastic Load Balancing Integration with Other AWS Services," AWS Documentation, <https://aws.amazon.com/elasticloadbalancing/features/integration/>, Accessed September 2021.
5. Amazon Web Services, "What Is AWS Web Application Firewall?" AWS Documentation, <https://aws.amazon.com/waf/>, Accessed September 2021.
6. L. Xue and L. Jie, "An Effective Load Balancing Algorithm for Cloud Computing," 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, 2010, pp. 1-3.
7. M. Pathan and I. Yaqoob, "Load Balancing in Cloud Computing: A Review," in Advances in Computing and Communications, A. Abraham, V. Snášel, and A. Krömer, Eds. Berlin, Heidelberg: Springer, 2012, pp. 1-10.
8. V. Zhang, H. Zhang, and L. Sun, "Research on Load Balancing Algorithms in Cloud Computing," 2012 Second International Conference on Cloud and Green Computing, Xiangtan, China, 2012, pp. 42-45.
9. Xi. Liu, Lei. Pan, Chong-Jun. Wang, and Jun-Yuan. Xie, "A Lock- Free Solution for Load Balancing in Multi-Core Environment", 3rd IEEE International Workshop on Intelligent Systems and Applications (ISA), 2011, pages 1-4.
10. Yao, J. and He, J., "Load Balancing Strategy of Cloud Computing based on Artificial Bee Algorithm", Proc. of the 2012 8th Intl Conf. of Computing Technology and Information Management (ICCM), 24- 26 Apr. 2012, Seoul, Korea, pages 185 -189.