1. INTRODUCTION

In today's digital era, calculator applications serve diverse purposes, and hosting them on Amazon Web Services (AWS) provides a scalable and reliable platform for delivery. In the digital era, where computation is an integral part of our daily lives, the role of calculator applications has expanded beyond simple arithmetic. They now power a diverse range of tasks, from financial calculations to scientific simulations. The ability to create, deploy, and scale these applications efficiently is paramount. It is a comprehensive exploration of this very endeavor, titled "Creating Calculator Websites and Hosting with Network Load Balancer, WinSCP, VPC, IGW, Security Groups, Route Tables, Ubuntu Servers, and Autoscaling on AWS."

Our journey into the world of calculator applications begins with a vision to harness the capabilities of Amazon Web Services (AWS) to create scalable and secure hosting environments. This project is designed for both newcomers looking to grasp the fundamentals of AWS infrastructure and seasoned developers seeking to refine their hosting skills. In this introduction, we will provide an overview of the project's objectives, the challenges it aims to address, and the significance of calculator applications in the digital landscape. We will also highlight the pivotal role that AWS services play in achieving our goals, emphasizing Network Load Balancers (NLBs) as a key component for efficient traffic management.

The results section will showcase the tangible outcomes of our endeavors, with performance metrics, scalability observations, and insights into securing file transfers and configuring DNS settings. We will explore each aspect of the project, from setting up the AWS infrastructure with Virtual Private Clouds (VPCs), Internet Gateways (IGWs), and Security Groups to deploying Ubuntu servers, configuring Load Balancers, implementing Autoscaling, and securing file transfers with WinSCP. By the end of this guide, you will have a robust understanding of how to create, host, and manage calculator websites on AWS.

The following AWS services are utilized in this project:

- **Network Load Balancer (NLB):** Network Load Balancer is a highly available and scalable load balancing service in AWS. It efficiently distributes incoming network traffic across multiple targets, such as Amazon EC2 instances, ensuring high availability and optimal resource utilization. In your project, NLB plays a pivotal role in managing traffic to your calculator websites, enhancing their reliability and performance.

- **Amazon VPC (Virtual Private Cloud):** Amazon VPC allows you to create isolated virtual networks within AWS. It provides complete control over your network settings, including IP address ranges, subnets, and route tables. VPC ensures the secure and scalable foundation for your calculator website hosting by defining network architecture and controlling traffic flow.

- **Internet Gateway (IGW):** The Internet Gateway is a VPC component that enables communication between your VPC and the internet. It allows your calculator websites to be accessible from the web, ensuring users can reach your hosted applications securely.

- **Security Groups:** Security Groups act as virtual firewalls for your Amazon EC2 instances within your VPC. They control inbound and outbound traffic, allowing you to specify which connections are allowed. In your project, they play a crucial role in enhancing the security of your calculator website infrastructure.

- **Route Tables:** Route Tables in Amazon VPC determine how network traffic is directed within the VPC. They specify the paths traffic should take, ensuring that it reaches its intended destinations. In your project, you'll configure custom route tables to manage traffic routing efficiently.

- **Amazon EC2 (Elastic Compute Cloud):** Amazon EC2 provides resizable compute capacity in the cloud. In your project, EC2 instances serve as the hosting environment for your calculator websites. They are configured with Ubuntu servers and web server software to run the applications.

- **Auto Scaling:** Auto Scaling is a service that automatically adjusts the number of EC2 instances in your application's Auto Scaling group to maintain performance and meet specified criteria. It ensures that your calculator websites can handle varying levels of traffic efficiently, maintaining high availability and cost-effectiveness.

- **WinSCP (Secure File Transfer Tool):** While not an AWS service, WinSCP is a third-party secure file transfer tool used to manage files and configurations on your EC2 instances. It plays a crucial role in securely transferring files to and from your hosting environment and is essential for maintaining and updating your calculator websites.

2. METHODOLOGY

The methodology encompassed several phases, including requirements gathering, design, implementation, testing, and deployment. The following sections outline each phase of the methodology in detail.

- **Requirements Gathering:** Gathering specific requirements for the calculator websites. This should include functional requirements, performance expectations, and any security or compliance considerations. Identify file transfer and management requirements, emphasizing the secure transfer of files using WinSCP.

- **Architecture and Infrastructure Planning:** Design the overall architecture of the hosting environment, emphasizing the use of AWS services such as Amazon VPC, EC2, Network Load Balancer, Auto Scaling, and other components. Determine the required AWS resources, including instance types, storage, and network configurations. Plan for the integration of WinSCP into the file transfer workflow.

- **Amazon VPC Setup:** Create a Virtual Private Cloud (VPC) to isolate and secure the hosting environment. Define subnets, route tables, and network ACLs to control traffic flow within the VPC.

- **Amazon EC2 Instances Configuration:** Launch and configure Ubuntu Server-based Amazon EC2 instances to serve as the hosting environment for the calculator websites. Set up the necessary software stack, including web servers, databases, and any additional dependencies.

- **Website Development and Deployment:** Develop the calculator websites, ensuring they are designed for scalability and optimal performance. Create deployment scripts or procedures to deploy the websites onto the EC2 instances

- **Network Load Balancer Configuration:** Configure the Network Load Balancer (NLB) to evenly distribute incoming traffic across the EC2 instances. Define NLB listeners, target groups, and health checks.

- **File Transfer and Configuration with WinSCP:** Install and configure WinSCP for secure file transfer between your local environment and the AWS EC2 instances. Use WinSCP to transfer website files, configurations, and updates to the hosted servers securely.

- **Auto Scaling Implementation:** Create an Auto Scaling group to automatically adjust the number of EC2 instances based on traffic patterns and demand. Set up scaling policies, alarms, and desired instance counts for Auto Scaling.

- **Security and Access Control:** Implement security best practices, including regular security updates and patch management for the EC2 instances. Configure AWS Security Groups and Network ACLs to control inbound and outbound traffic. Define access controls and permissions for team members involved in managing the hosting environment.

- **Testing and Quality Assurance:** Conduct comprehensive testing of the calculator websites to ensure all functionalities work as intended. Perform load testing to assess the effectiveness of Auto Scaling and NLB in handling increased traffic loads.

- **Documentation and Reporting:** Create comprehensive documentation covering all aspects of the project, including architecture, configurations, procedures, and the use of WinSCP. Generate reports summarizing the project's implementation, testing results, and any encountered challenges.

This methodology outlines a systematic approach to creating calculator websites and hosting them on AWS, emphasizing the integration of WinSCP for secure file transfers. It ensures security, scalability, and reliability in the hosting environment while addressing thunique file transfer needs of the project.

3. SYSTEM DESIGN / ARCHITECTURE

This system architecture diagram represents the infrastructure and components that power our calculator-based web application. At its core, our application relies on a set of web servers that handle user requests and process calculations. These web servers are deployed within an Amazon Virtual Private Cloud (VPC) for security and scalability. Additionally, we utilize a Network Load Balancer (NLB) to efficiently distribute incoming traffic across multiple server instances, ensuring high availability and optimal performance. The architecture also includes a database server, responsible for storing user data and application settings. As this diagram unfolds, you will gain insights into the interconnected components that enable our calculator application to deliver a seamless and reliable experience to users.

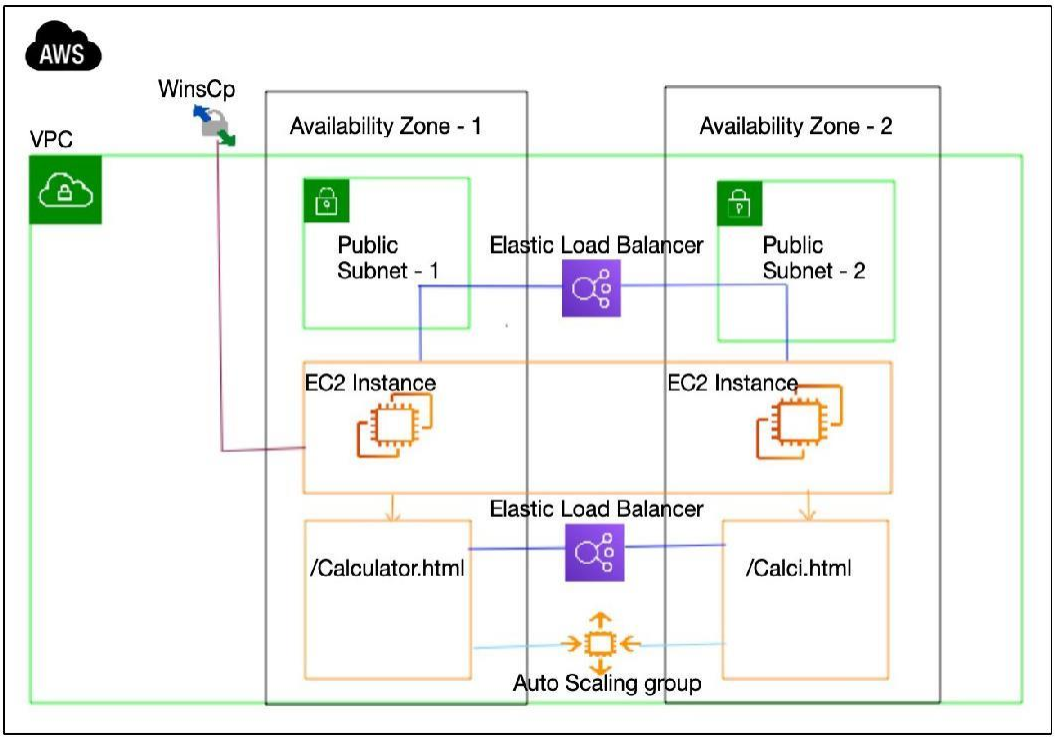

Fig.3.1 : Architecture of the Calculator Web-Application

3.1 BUILDING CALCULATOR WEBSITES:

Calculator website is build using HTM, JAVA SCRIPT, and CSS. Here's an overview of the key steps and information to create a Calculator website:

1. Website creation involves:

- HTML File: Create an HTML file (e.g., index.html) to define the structure of your web page.
- CSS File: Create a CSS file (e.g., style.css) to style your calculator interface.
- JavaScript File: Create a JavaScript file (e.g., script.js) to handle calculator functionality.

2. HTML Structure: Create the basic structure of your web page using HTML elements. Typically, this includes a header, calculator display area, and a set of buttons for numbers and operations. Use HTML form elements and input fields to display and process calculations.

3. Styling with CSS: Apply CSS styles to your HTML elements to create an attractive and user-friendly calculator interface. Use CSS classes and IDs to target specific elements for styling.

4. JavaScript Functionality: Implement JavaScript functions to handle user interactions and perform calculations. Capture user input from button clicks and update the display accordingly. Implement functions for arithmetic operations such as addition, subtraction, multiplication, and division. Ensure that your calculator handles edge cases and error conditions gracefully.

5. User Interaction: Add event listeners to buttons so that the calculator responds to user clicks. Implement keyboard input handling to allow users to type calculations.

6. Calculator Logic: Develop the logic for the calculator to perform arithmetic operations. Ensure that the calculator handles complex operations like parentheses and follows the order of operations (PEMDAS/BODMAS).

7. **Display and Output:** Create an area on the webpage where the calculator display is updated with user input and calculation results. Ensure that the display is responsive and adjusts to fit the content.

8. **Testing:** Test your calculator thoroughly to ensure accurate calculations and a smooth user experience. Check for edge cases and handle errors gracefully.

9. **Deployment:** Once your calculator website is complete and fully tested, you can deploy it to a web server or hosting platform of your choice. Consider using cloud hosting services or website builders to make your calculator website accessible online

Fig. 3.2: Scientific Calculator Using HTML , Java Script ,CSS

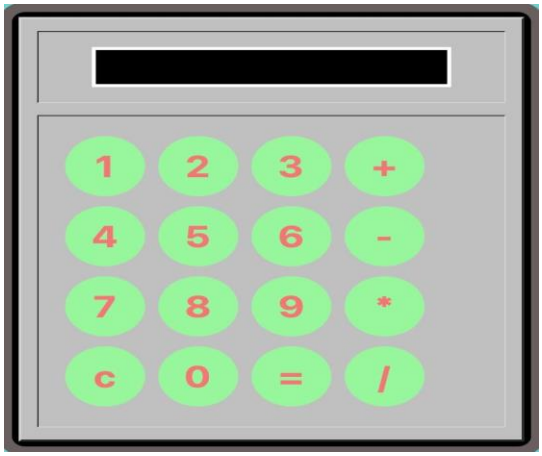

Fig. 3.3 : Calculator using HTML, CSS, Java Script

3.2 CREATING VPC (VIRTUAL PRIVATE CLOUD):

Creating a Virtual Private Cloud (VPC) in Amazon Web Services (AWS) is a fundamental step in setting up your network infrastructure for cloud-based applications and services. A VPC is a logically isolated section of the AWS cloud where you can launch AWS resources like EC2 instances, RDS databases, and more. It allows you to have control over your network environment, including IP address ranges, subnets, routing tables, and security settings. Here's how you can create a VPC in AWS:

1. **Sign in to AWS Console:**

- Go to the AWS Management Console (https://aws.amazon.com/).
- Sign in with your AWS account credentials.

2. **Access the VPC Dashboard:**

- In the AWS Management Console, search for and select "VPC" under the "Networking & Content Delivery" section, or you can also find it under "Services" > "Networking & Content Delivery" > "VPC."

3. **Create a VPC:**

- In the VPC Dashboard, click on the "Create VPC" button.

4. **Configure VPC Settings:**

- Fill out the necessary details:
- Name Tag: Give your VPC a descriptive name – PROJECT-VPC.
- IPv4 CIDR Block: Define the IP address range for your VPC in CIDR notation.
- IPv6 CIDR Block (Optional): You can also assign an IPv6 CIDR block if needed.
- Tenancy: Choose between "Default" (shared hardware) or "Dedicated tenancy.

5. **Review and Create:**

- Review your settings to ensure they are correct.
- Click the "Create VPC" button.

6. VPC Creation:

- AWS will create your VPC based on the settings you provided. This process usually takes a few moments. Select VPC, Click on Actions choose Edit VPC settings and enable DNS hostname option and then save.

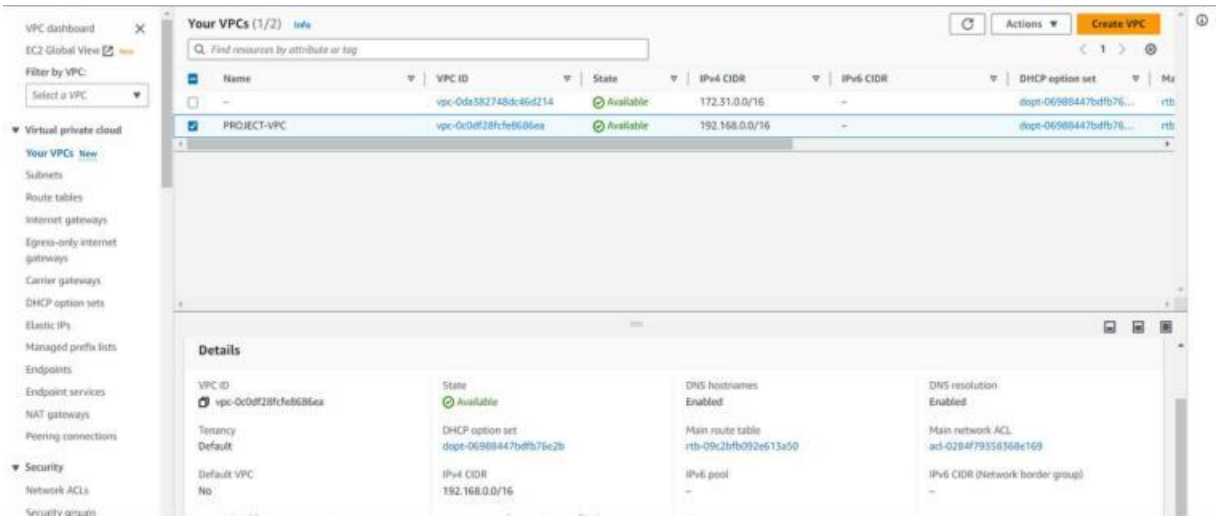

Fig. 3. 4 : VPC Details After Creation

3.3 CRETING PUBLIC SUBNETS

Creating public subnets in AWS is a crucial part of setting up a Virtual Private Cloud (VPC) to host resources that need to be accessible from the internet. Public subnets typically house resources like web servers, load balancers, and other services that need to accept incoming traffic from the internet. Here's how you can create subnets in AWS**:**

1. **Access the VPC Dashboard:**

- In the AWS Management Console, search for and select "VPC" under the "Networking & Content Delivery" section, or you can also find it under "Services" > "Networking & Content Delivery" > "VPC."

2. **Create the First Public Subnet (PROJECT SUBNET-1):**

- In the VPC Dashboard, click on "Subnets" in the left-hand navigation pane.
- Click the "Create Subnet" button to begin creating a new subnet.
- Configure the subnet as follows:
- Name Tag: PROJECT SUBNET-1
- VPC: Select your existing VPC: PROJECT-VPC.
- Availability Zone: Choose "us-east-1a" (or the appropriate zone for your region).
- IPv4 CIDR Block: 192.168.1.0/24
- Auto-assign IPv4 Public IP: Enable this option.
- Click the "Create subnet" button.

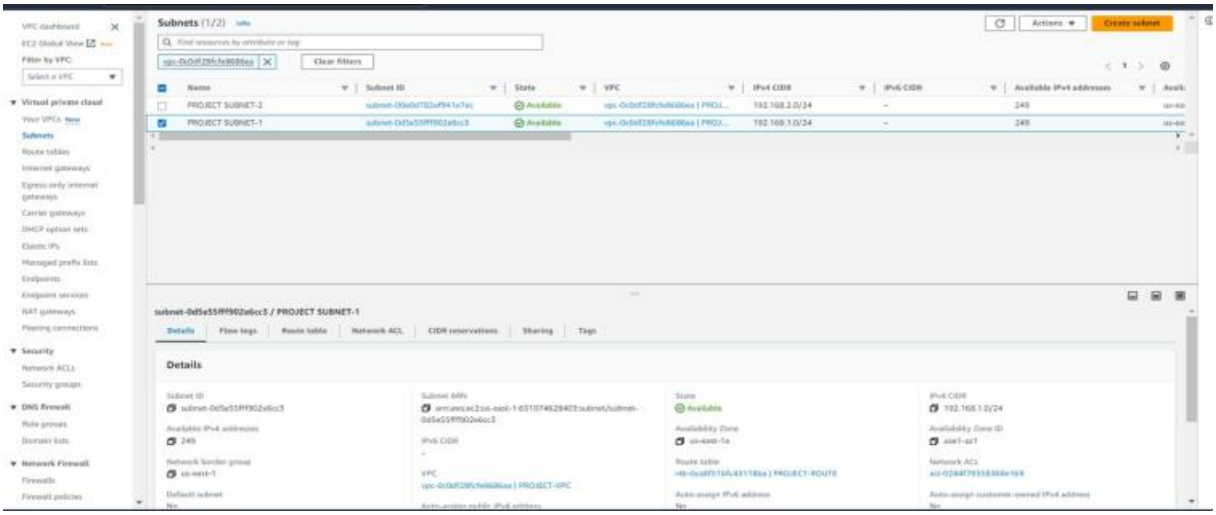

Fig. 3.5 : Public Subnet-1 After Creation

3. Create the Second Public Subnet (PROJECT SUBNET -2):

- Follow the same steps as above, but configure the subnet as follows:
    - Name Tag: PROJECT SUBNET-2
    - VPC: Select your existing VPC: PROJECT-VPC.
    - Availability Zone: Choose "us-east-1b" (or the appropriate zone for your region).
    - IPv4 CIDR Block: 192.168.2.0/24
    - Auto-assign IPv4 Public IP: Enable this option.
- Click the "Create subnet" button.

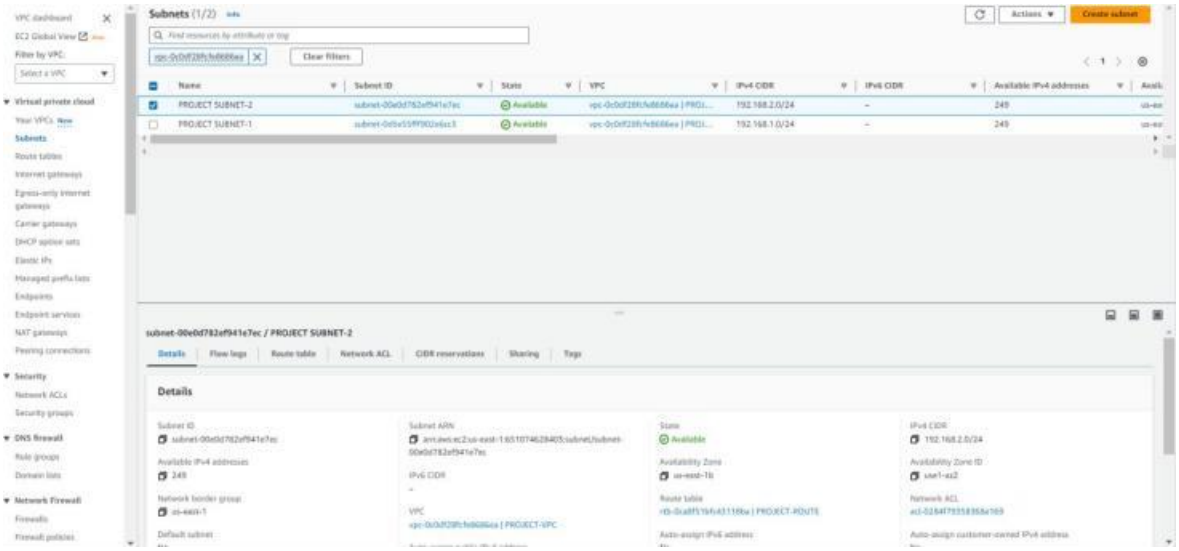

Fig. 3.6 : Public Subnet-2 After Creation

3.4 CREATING INTERNET GATEWAY AND ATTACH TO VPC

An internet gateway is a network device or software application that connects a local area network (LAN) or private network to the internet. It serves as the entry and exit point for data traffic between the internal network and the external internet. Internet gateways play a crucial role in routing data packets between devices within the local network and the global internet

1. **Access the VPC Dashboard:**

- In the AWS Management Console, search for and select "VPC" under the "Networking & Content Delivery" section, or you can also find it under "Services" > "Network Content Delivery" > "VPC."

2. **Create an Internet Gateway:**

- In the VPC Dashboard, click on "Internet Gateways" in the left-hand navigation pane.
- Click the "Create Internet Gateway" button.

3. **Configure Internet Gateway:**

- In the "Create Internet Gateway" wizard, fill out the necessary details:
- **Name Tag:** Project-IGW
- Click the "Create Internet Gateway" button.

4. **Attach Internet Gateway to VPC:**

- After creating the Internet Gateway, select it in the list of Internet Gateways.
- From the "Actions" menu, choose "Attach to VPC."
- Select the VPC you want to attach the Internet Gateway.
- Click the "Attach Internet Gateway" button.

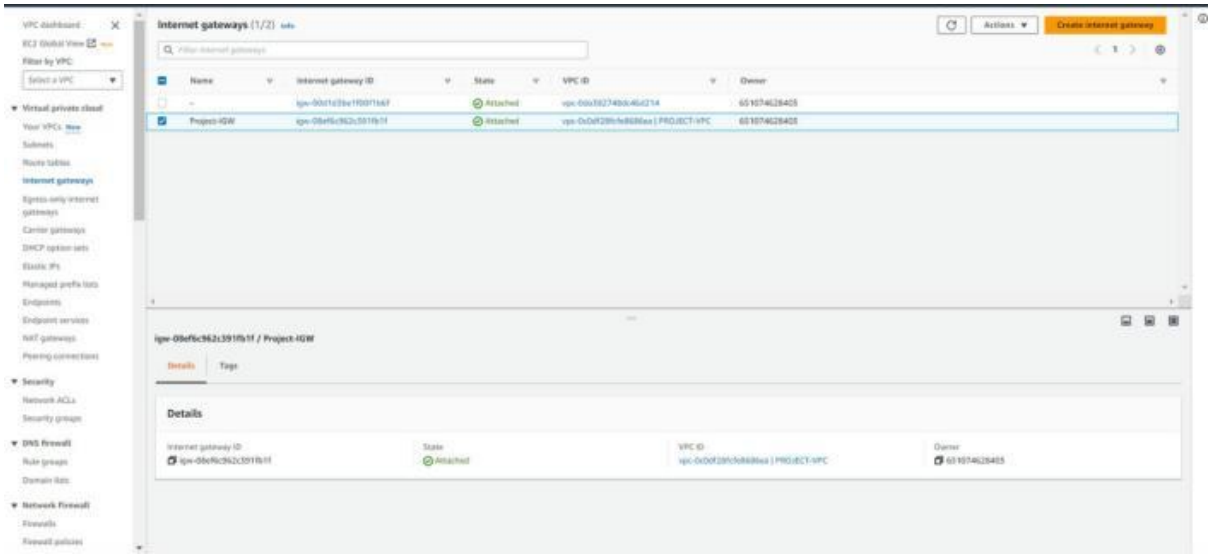

Fig.3.7 : Internet Gateway After Attaching to VPC

3.5 CREATING ROUTE TABLES

A routing table is a data structure used in computer networking to store information about how to forward data packets from one network or host to another. Routing tables are crucial for routers, switches, and other networking devices to make decisions about the best path for data to take as it travels through a network.

1. **Access the VPC Dashboard:**

- In the AWS Management Console, search for and select "VPC" under the "Networking & Content Delivery" section, or you can also find it under "Services" > "Networking & Content Delivery" > "VPC."

2. **Create a Custom Route Table:**

- In the VPC Dashboard, click on "Route Tables" in the left-hand navigation pane.
- Click the "Create Route Table" button.

3. **Configure Route Table:**

- Fill out the necessary details for your custom route table:
- **Name Tag:** PROJECT-ROUTE
- **VPC:** Select the VPC you want to associate this route table with PROJECT-VPC.
- Click the "Create Route Table" button.

4. **Edit Routes:**

- After creating the route table, select it from the list of route tables.
- In the "Routes" tab, click the "Edit routes" button.

5. **Add Default Route to Internet Gateway:**

- In the "Edit routes" window, click the "Add route" button.
- In the "Destination" field, enter **0.0.0.0/0** to represent all internet-bound traffic.
- In the "Target" field, select the Internet Gateway that previously created (e.g., "Project-IGW").
- Click the "Save routes" button to add the default route.

6. **Associating Subnets with the Route Table:**

- To make use of this custom route table, you need to associate it with the subnets that you want to have the default route.

- In the "Subnet Associations" tab, click the "Edit subnet associations" button.
- Select the subnets you want to associate with this route table (PROJECT SUBNET-1, PROJECT SUBNET-2).
- Click the "Save associations" button.

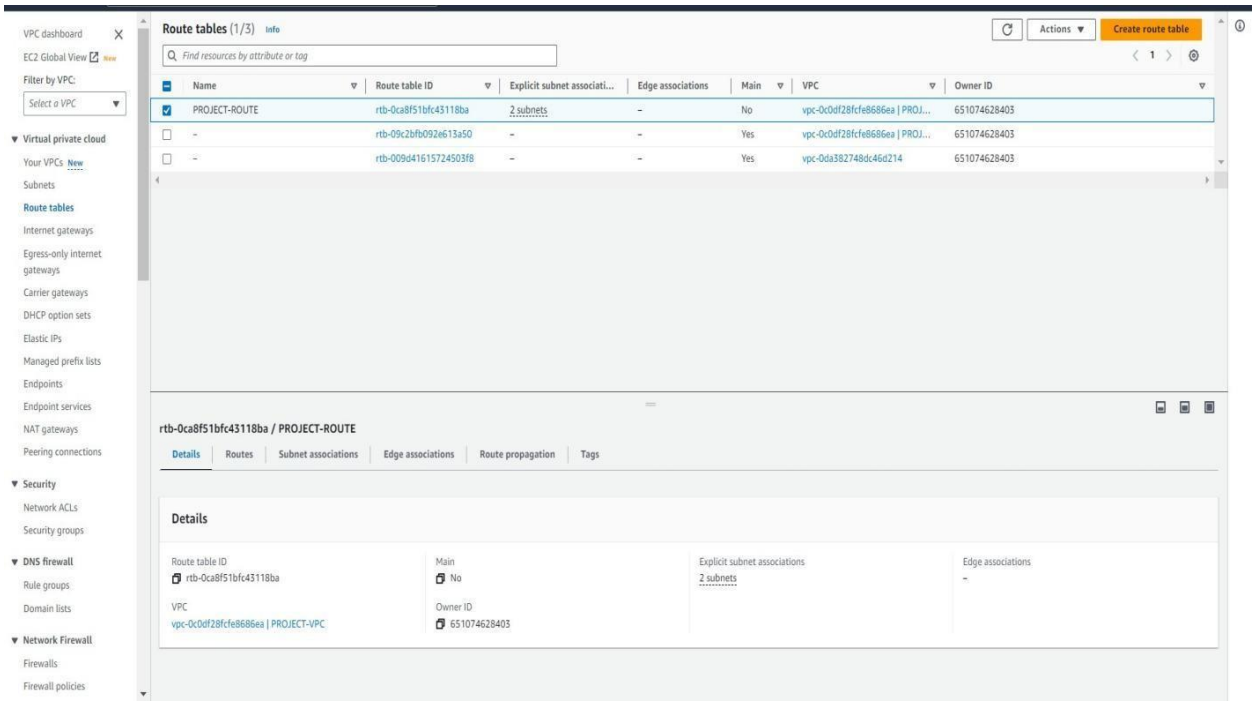

Fig.3.8 : Details of Route table which is associated by two subnets

4. IMPLEMENTATION

The implementation phase, the frontend of the Calculator Hosting System was constructed using standard web technologies such as HTML, CSS, and JavaScript. These technologies were essential for creating a visually appealing and responsive user interface. The primary goal was to transform the design specifications into a fully operational web application.

4.1 Frontend Development

The frontend of the Scientific Calculator was developed using HTML, CSS, and JavaScript to create an intuitive and visually appealing user interface. Here's a breakdown of the key components and features:

- **HTML Structure**: The HTML structure defines the layout of the calculator, including a title, display area, and rows of buttons.
- **CSS Styling**: Cascading Style Sheets (CSS) were used to style the calculator, including defining colours, button styles, and overall layout. The design focused on making the calculator visually appealing.
- **Display Area**: A dynamic display area with an input field (id="screen") was created to show the current input and calculation results.
- **Buttons**: Numeric buttons (0-9), basic operators (+, -, *, /), and advanced function buttons (e.g., sin, cos, log) were implemented. Each button has a specific function associated with it and is styled for interactivity.
- **Button Click Events**: JavaScript was used to handle button click events. When a button is clicked, the associated function is executed, updating the display accordingly.

4.2 Creating Ec2 Instances

Amazon Elastic Compute Cloud (Amazon EC2) is a fundamental web service provided by Amazon Web Services (AWS) that allows users to rent virtual servers in the cloud. EC2 instances provide scalable compute capacity, making it easy to deploy and manage applications, run workloads, and host websites.

Launching 1st Instance:

1. **Launch EC2 Instances:**
   - In the AWS Management Console, navigate to the EC2 service.
2. **Launch the First EC2 Instance (PROJECT-UB1):**
   - Click the "Launch Instances" button.
   - Choose an Amazon Machine Image (AMI) for Ubuntu.
   - Select an instance type based on your requirements.
   - Configure the instance details:
     - **Network:** Choose "PROJECT-VPC."
     - **Subnet:** Choose "PROJECT SUBNET-1."
     - **Auto-assign Public IP:** Enable this option to assign a public IP address to the instance.
   - Click "Next: Add Storage" and configure storage as needed.
   - Click "Next: Add Tags" and add a name tag, e.g., "Name: PROJECT-UB1."
   - Click "Next: Configure Security Group."
3. **Create a Security Group (PROJECT-SECURITY):**
   - If you haven't created a security group yet, click the "Create a new security group" link.
   - Configure the security group as follows:
     - **Security group name:** PROJECT-SECURITY
     - **Description:** A description for the security group
     - **Add rules:**
       - Type: SSH (for allowing SSH from anywhere).

- Type: HTTP (for allowing HTTP from anywhere).
- Type: HTTPS (for allowing HTTPS from anywhere).

- Click "Review and Launch."

4. Review and Launch (PROJECT-UB1):

- Review your instance details, storage, tags, and security group.
- Click "Launch Instance."

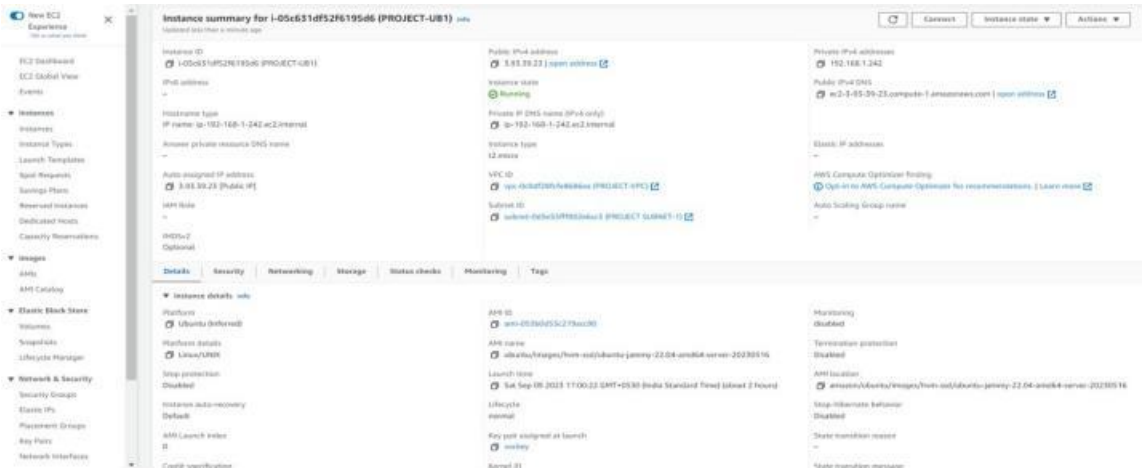

Fig. 4.1 : Details of Instance After Launching First Instance

Launching 2nd Instance:

1. Launch EC2 Instances:

- In the AWS Management Console, navigate to the EC2 service.

2. Launch the First EC2 Instance (PROJECT-UB1)

- Click the "Launch Instances" button.
- Choose an Amazon Machine Image (AMI) for Ubuntu.
- Select an instance type based on your requirements.
- Configure the instance details:
  - **Network:** Choose "PROJECT-VPC."
  - **Subnet:** Choose "PROJECT SUBNET-1."

- **Auto-assign Public IP:** Enable this option to assign a public IP address to the instance.

- Click "Next: Add Storage" and configure storage as needed.
- Click "Next: Add Tags" and add a name tag, e.g., "Name: PROJECT-UB1."
- Click "Next: Configure Security Group."

3. Select an Existing Security Group (PROJECT-SG-1):

- Choose an existing security group that you previously created (PROJECT-SECURITY) or one that allows SSH, HTTP, and HTTPS traffic from anywhere.
- Click "Next: Instance Launch."

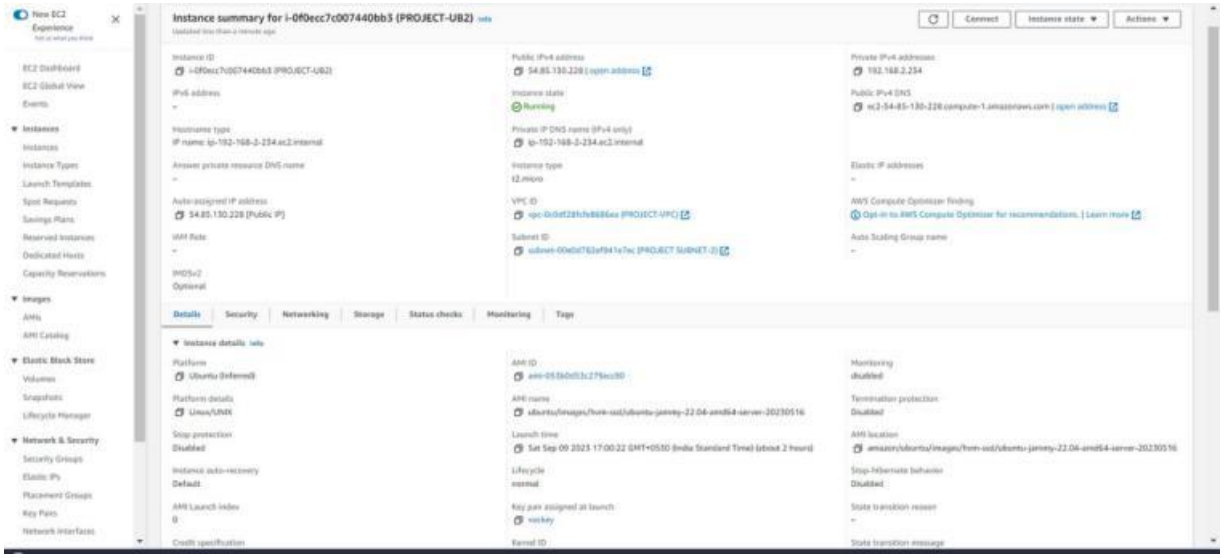

Fig.4.2 : Details of instance after launching second instance

4.3 CREATING NETWORK LOAD BALANCER

Steps involved in creating a load balancer :

- **AWS Account Setup and Region Selection:** Sign in to your AWS Management Console or create an AWS account if you don't have one. Choose an AWS region where you want to create your NLB and servers. Ensure that your servers (EC2 instances) and NLB are in the same region.
- **Create Ubuntu Servers (EC2 Instances):** Navigate to the EC2 dashboard in the selected region. Launch two EC2 instances using the Ubuntu Server AMI. Configure the instances with the necessary settings, including instance type, storage, security groups, and key pairs. Assign Elastic IP addresses to the instances for static public IPs if needed.
- **Configure Security Groups:** Create or configure security groups for your EC2 instances Ensure that the security groups allow traffic on the necessary ports (e.g., 80 for HTTP, 443 for HTTPS) and that they allow incoming traffic from the NLB.
- **Set Up Application or Services on EC2 Instances:** SSH into the Ubuntu servers. Install and configure your web server software (e.g., Apache) and any application or service you want to load balance. Ensure that your web servers respond to HTTP or HTTPS requests on the configured ports (e.g., 80 or 443).
- **Create a Target Group:** Go to the EC2 dashboard and navigate to the "Target Groups" section. Create a new target group and specify the protocol (HTTP or HTTPS) and port (e.g., 80 or 443) your servers will use. Register your EC2 instances with the target group.
- **Create a Network Load Balancer (NLB):** In the EC2 dashboard, go to the "Load Balancers" section. Create a new Network Load Balancer. Configure the NLB settings, including listeners (protocol and port), VPC, and subnets. Choose the target group created in the previous step as the default target group for the NLB
- **Update DNS or Domain Records (Optional):** If you want to use a custom domain, update your DNS records (e.g., A or CNAME records) to point to the NLB's DNS name. This step allows users to access your NLB using a custom domain name.
- **Test the NLB:** Access your NLB's DNS name or IP address in a web browser to test the load balancing. Ensure that traffic is evenly distributed between the two Ubuntu servers.

Your NLB is now set up to distribute incoming traffic across your two Ubuntu servers. This setup provides load balancing and high availability for your web application or services. Be sure to monitor your NLB and servers for performance and scaling as needed.

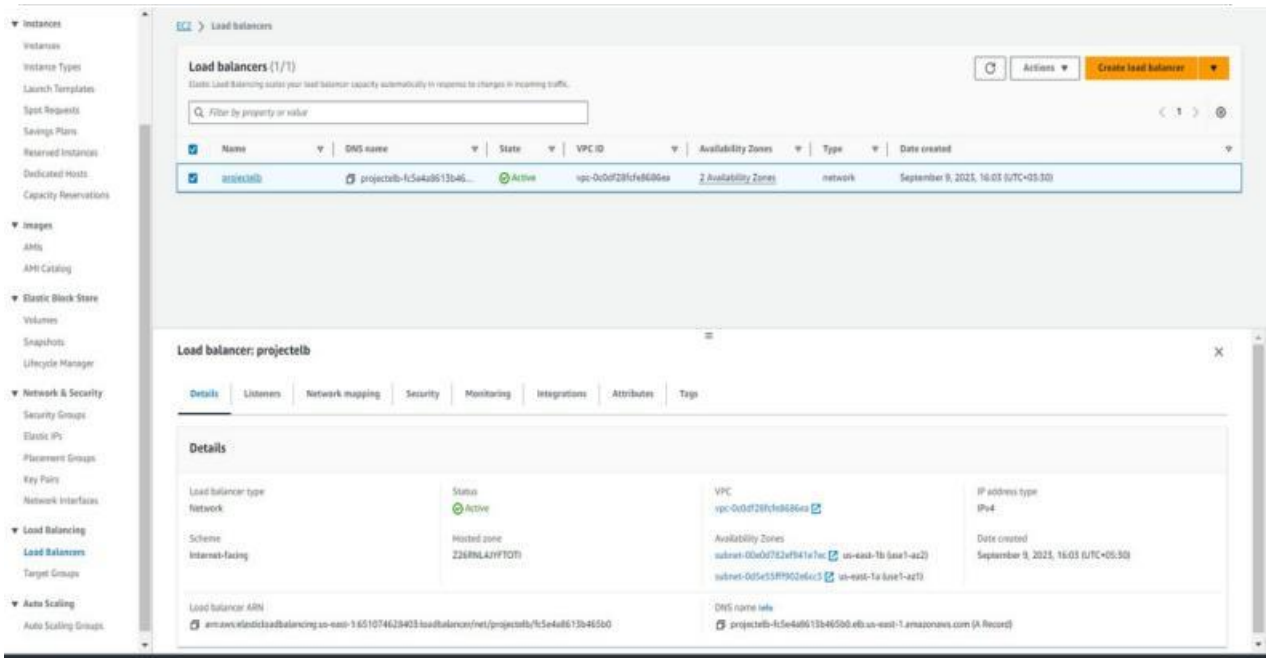

Fig.4.3 : Details of Network Load Balancer

4.4 HOSTING WEBSITES USING WINSCP

Steps involved in website hosting:

- **Connect to the Instances:** Ensures that you have already configured the load balancer to distribute traffic to your two Ubuntu instances. Launch WinSCP on your local computer. Create two separate sessions in WinSCP for each of your Ubuntu instances using their respective IP addresses or hostnames.

- **Transfer Website Files:** In each WinSCP session, navigate to the web server's document root directory where you want to host your websites (e.g., /var/www/html/). Open your local directory where your calculator and scientific calculator website files are stored. Select the website files for the calculator website in one session and the scientific calculator website in the other. Drag and drop the selected files from your local directory to the remote directory on each of the Ubuntu instances.

- **Verify Website Files:** After the transfer is complete for both sessions, verify that your website files are located in the correct directories on each Ubuntu instance. Ensure that file permissions are set correctly to allow the web server to access and serve the files.

- **Access Your Websites:** Use a web browser to access your calculator website by entering the public IP address or domain name associated with your load balancer. Similarly, access your scientific calculator website using the same IP address or domain name. You should now be able to view and use both hosted websites online.

Since the load balancer distributes traffic to your Ubuntu instances, the websites will be accessible through the load balancer's endpoint, ensuring that requests are balanced across both servers for high availability and scalability.

4.5 CONNECTING LOAD BALANCER TO AUTOSCALING

connecting load balancer to Auto Scaling ensures high availability and scalability web application. Steps involved in this process

- **Create an Auto Scaling Group for Ubuntu Instances:** Go to the AWS Management Console and navigate to the Auto Scaling service. Create a new Auto Scaling group (or use an existing one if you have it) for your Ubuntu instances. Configure the Auto Scaling group with the following details: Choose your desired instance type for Ubuntu (e.g., t2.micro).Set the minimum and maximum number of instances based on your scaling requirements. Specify the launch configuration for the Ubuntu instances (AMI, security groups, key pair, etc.)

- **Specify the Load Balancer:** During the Auto Scaling group setup, specify the load balancer to which the instances should be registered. Select the target group associated with your load balancer. This ensures that instances launched by the Auto Scaling group will automatically register with the load balancer.

- **Configure Health Checks:** Define health checks for your Auto Scaling group to determine the health of instances. Specify the target path and response codes that indicate a healthy instance. Instances failing health checks will be terminated and replaced.

- **Define Scaling Policies:** Create scaling policies for your Auto Scaling group. These policies determine when and how instances should be added or removed based on specific metrics. You can set up policies for scaling out (adding instances) and scaling in (removing instances) based on CPU utilization, network traffic, or other custom metrics.

- **Set Up CloudWatch Alarms (Optional):** Create CloudWatch alarms to monitor the performance and health of your instances and application. These alarms can trigger Auto Scaling actions when specific thresholds are breached.

- **Monitoring and Scaling:** Monitor the performance and traffic patterns of your application. Auto Scaling will automatically adjust the number of Ubuntu instances behind the load balancer based on your scaling policies and health checks. Instances that become unhealthy will be replaced, ensuring that only healthy instances serve traffic.

5. RESULT

In Building and hosting calculator websites using AWS services, these are the results:

- **Performance Metrics:** Provide performance metrics and benchmarks for your web application before and after implementing Auto Scaling and load balancing. Metrics could include response times, server load, and resource utilization.

- **Scalability Testing:** Detail the results of scalability testing. Show how your application's performance scales when subjected to varying levels of traffic or load. Include data on how Auto Scaling effectively added or removed instances as needed.

- **High Availability Testing:** Discuss the high availability achieved through the use of Auto Scaling and load balancing. Include any test scenarios or real-world incidents where the system demonstrated its ability to maintain availability.

- **Fault Tolerance and Redundancy:** Describe the system's fault tolerance capabilities and how it handles instances that fail health checks or become unavailable. Include data on how Auto Scaling replaced unhealthy instances.

- **Load Balancer Efficiency:** Present data on how the load balancer effectively distributed incoming traffic among instances. Discuss any load balancing algorithms used and their impact on performance.

- **Cost Efficiency:** Analyse the cost savings achieved by implementing Auto Scaling. Provide data on how the system optimized resource usage and reduced costs during low-traffic periods.

- **Security and Compliance:** Discuss how security measures were maintained or enhanced during the project. Include any compliance requirements met by the architecture.

- **Scaling Events:** Provide examples of real-world scaling events, such as traffic spikes or increased load, and describe how the system responded by automatically removing instances

- **Future Scalability and Optimization Opportunities:** These include additional metrics to monitor and advanced Auto Scaling policies that can be implemented to enhance the resilience and efficiency of our application.

The impact of implementing Auto Scaling and load balancing on the performance, availability, and efficiency of your calculator websites. It helps showcase the benefits and effectiveness of your project's architecture and configuration choices.

5.1 SCREENSHOTS OF THE OUTPUTS

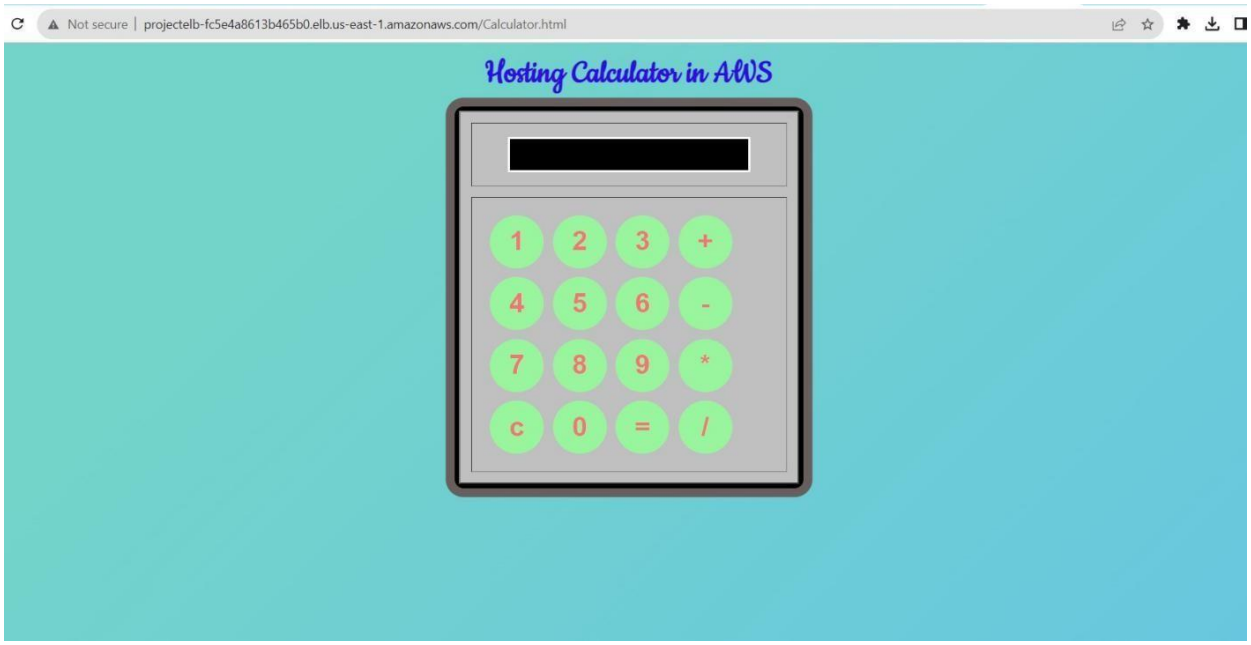

Fig.5.1 : Calculator Website Displayed After Copying Load Balancer DNS

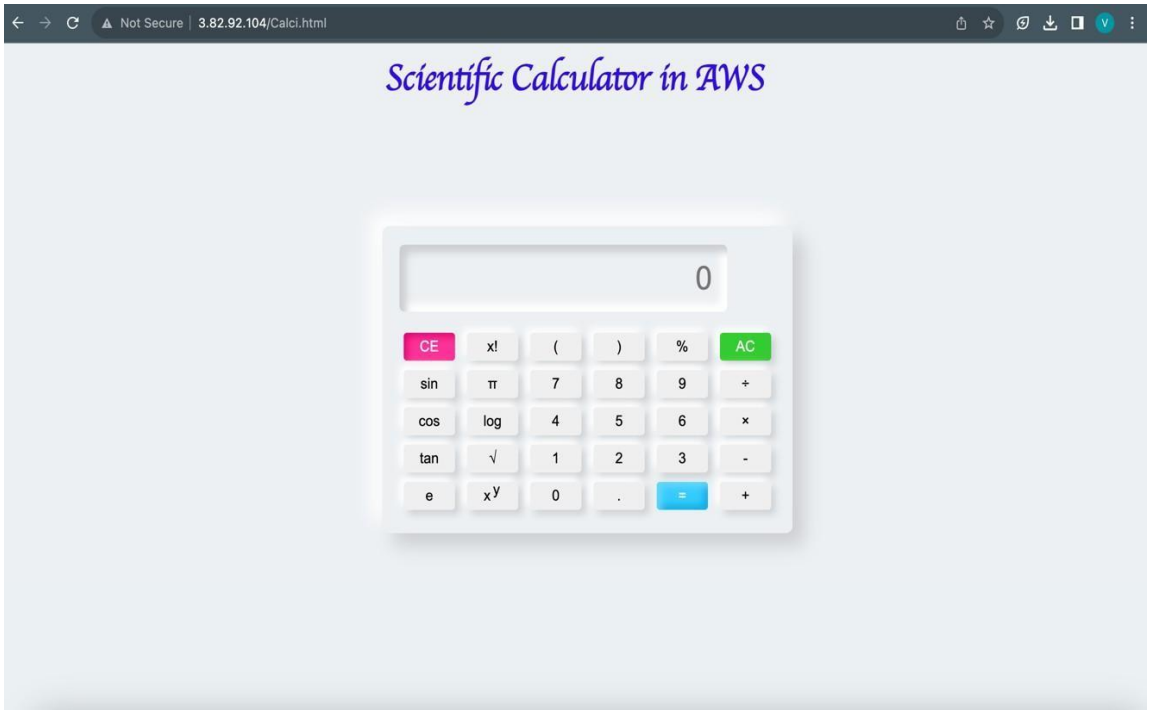

Fig.5.2 : Calculator Website Displayed After Copying Load Balancer DNS

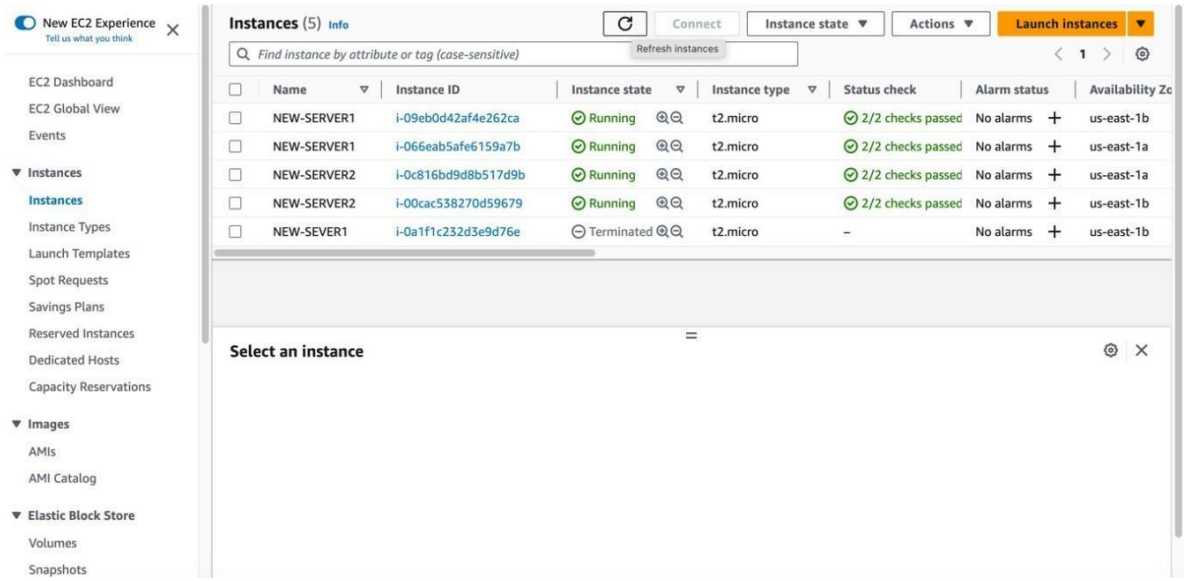

Fig.5.3 : Auto Scaling With Load Balancer

6. CONCLUSION

In building and hosting websites using WinSCP, Network Load Balancer, VPC, Ubuntu servers, and Auto Scaling on Amazon Web Services (AWS) has been an exploration of the synergy between cutting-edge technologies and practical web development strategies. With a focus on scalability, security, and efficient traffic management, this project has equipped developers and administrators with valuable insights into creating resilient web hosting environments. The inclusion of WinSCP has streamlined file transfers, while Network Load Balancers have optimized traffic distribution for enhanced performance and reliability.

The secure and isolated environment provided by the Virtual Private Cloud (VPC) has reinforced the importance of robust infrastructure design. Ubuntu servers have showcased the flexibility of open-source solutions, while Auto Scaling has ensured resource efficiency. We've emphasized the need for continuous improvement, monitoring, and adaptation in the ever-evolving digital landscape. As we conclude, we invite you to apply this knowledge to your own projects, recognizing the potential for innovation and success in the dynamic realm of web development and hosting.

7. BIBLIOGRAPHY & REFERENCES

During this online internship , I made use of every online resource possible . But mostly I used YouTube , Google and Simple learn platform courses in order to complete this project properly The reference links are given below!

Reference Links and documents:

[1] Virtual Private Cloud:

https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

[2] Amazon autoscaling:

https://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-auto-scaling-vpc-endpoints.html

[3] Load balancing:

https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

[4] My Github Repository Source Code Link:

https://github.com/Charishma2004/internship.git

8. PROPOSED ABSTRACT

Recently, food security has become a significant concern worldwide as a serious risk to sustainable societies. One of the key challenges facing food security is ensuring that plants are grown in optimal conditions. Therefore, it is important to implement effective monitoring systems in plant agriculture. In this abstract, we present an Internet of Things (IoT) application for plant agriculture monitoring systems to help farmers monitor and track plant environmental conditions. The system consists of a remote camera, power source, a sensor module, and an AWS cloud platform. To reduce power consumption, it periodically measures various environmental parameters such as temperature, humidity, and soil moisture with power management configured. Then , it sends them to AWS securely with token-based authentication. It also captures the plant using a remote camera and sends it to a cloud. Finally, users can monitor and track the collected plant information through our system's user interface from Android application on their mobile phones