**Resilient and Scalable Web Application Deployment in AWS**

**Project Description: -**

* This project involves designing and implementing a highly available and scalable web application infrastructure on AWS.
* The architecture leverage AWS services to ensure **Fault tolerance**, **Load balancing**, **Secure user access**.
* The core of the project is to deploy a web application that can handle varying loads efficiently and maintain high availability across multiple Availability Zone (AZs).

**Objectives: -**

1. **High Availability** – Achieve minimal downtime for the web application by utilizing multiple Availability.
2. **Scalability** – Use AWS Auto Scaling to adjust resources automatically in response to traffic changes, ensuring efficient performance.
3. **Security** – Implement security measures focusing on security groups and secure communication.
4. **Resilience** – Develop a resilient application setup that can withstand failures and traffic spikes without manual intervention.

**Core AWS Service Utilization: -**

* **Virtual Private Cloud (VPC)** – Set up a custom VPC to provide a isolated network environment. This VPC will have public and private subnets across different AZs for enhanced security and availability.
* **Elastic File System (EFS)** – Leverage EFS for scalable file storage, which can be concurrently accessed by instances for storing shared application data.
* **Elastic Compute Cloud (EC2)** – Utilize EC2 instances to host the web application.
* These instances will serve as the compute resources running the application, benefiting from AWS's secure, resizable compute capacity.
* **AWS Auto Scaling** – Configure Auto Scaling to dynamically adjust the number of EC2 instances, ensuring that the application scales efficiently with demand.
* **Application Load Balancer (ALB)** – Utilize an ALB to distribute incoming traffic across multiple EC2 instances in different AZs, enhancing the fault tolerance and availability of the application.
* **Route 53** – Employ Route 53 for domain management and to route end-user requests to the application in a reliable and cost-effective manner.

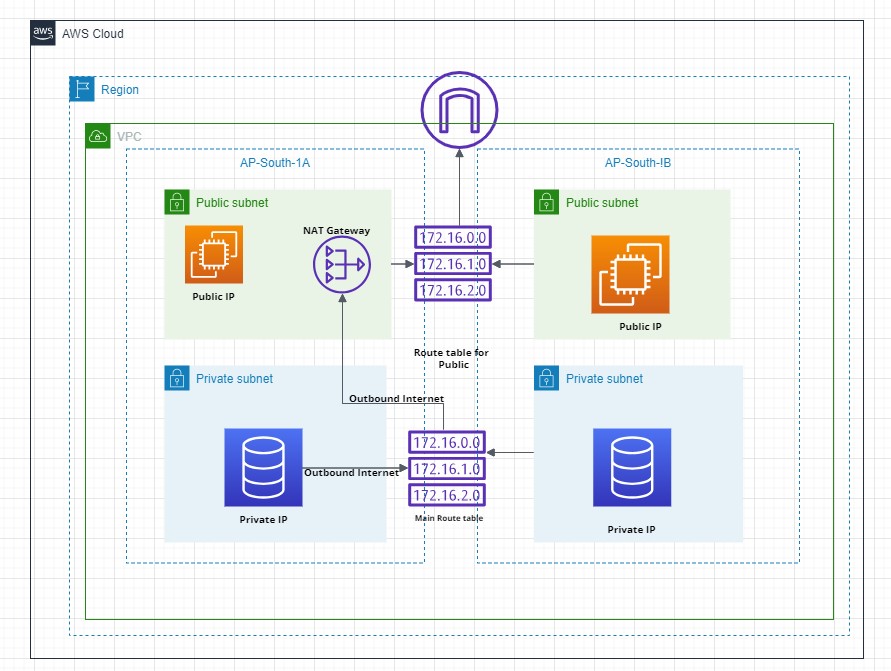
**Project phases**

1. **Design Phase** – Architect the solution, focusing on the application’s **Security, Scalability, Availability requirements.**
2. **Implementation Phase** –
   1. Create the VPC, subnets, and security groups.
   2. Configure EFS.
   3. Setup Custom AMI For Auto Scaling.
   4. Set up and test Auto Scaling.
   5. Deploy the ALB.
   6. Integrate Route 53 for domain management.
3. **Testing and Optimization Phase** – Conduct functional and load testing to ensure the application's performance and scalability meet requirements.
4. **Documentation** Phase – Produce detailed documentation covering the architecture, configuration, and deployment process.

**Deliverables –**

* Architectural diagrams and design documentation.
* Implementation and configuration guide.
* Performance and optimization report.
* A comprehensive project presentation detailing the deployment strategy, encountered challenges, and solutions.

**Phase 1: - Creation of VPC, Subnets and Security Groups**



This AWS architecture is designed for high availability and resilience by leveraging multiple Availability Zones (AZs) within a VPC. The VPC spans two AZs (AP-South-1A and AP-South-1B), each containing both public and private subnets. Public subnets in each AZ (192.168.0.0/26 and 192.168.0.64/26) host internet-facing resources, ensuring accessibility even if one AZ fails. Private subnets in each AZ (192.168.0.128/26 and 192.168.0.192/26) house backend resources, maintaining functionality during an AZ outage. A NAT Gateway in the AP-South-1A public subnet allows secure internet access for private subnet instances, with the potential to add another NAT Gateway in AP-South-1B for enhanced resilience. Route tables direct traffic efficiently, with the main route table sending private subnet traffic through the NAT Gateway and public subnet traffic directly to the internet gateway. This architecture ensures continuous operation, minimal downtime, and robust disaster recovery capabilities.

Let's break down each component in detail:

**Components:**

1. **VPC (Virtual Private Cloud):**
   * The entire architecture is encapsulated within a VPC, providing isolated networking within the AWS cloud.
   * The VPC has a CIDR block of 192.168.0.0/24.

Log into AWS Management Console and Navigate to VPC to Create the Virtual Private Cloud for our project.

A screenshot of a computer

Description automatically generated

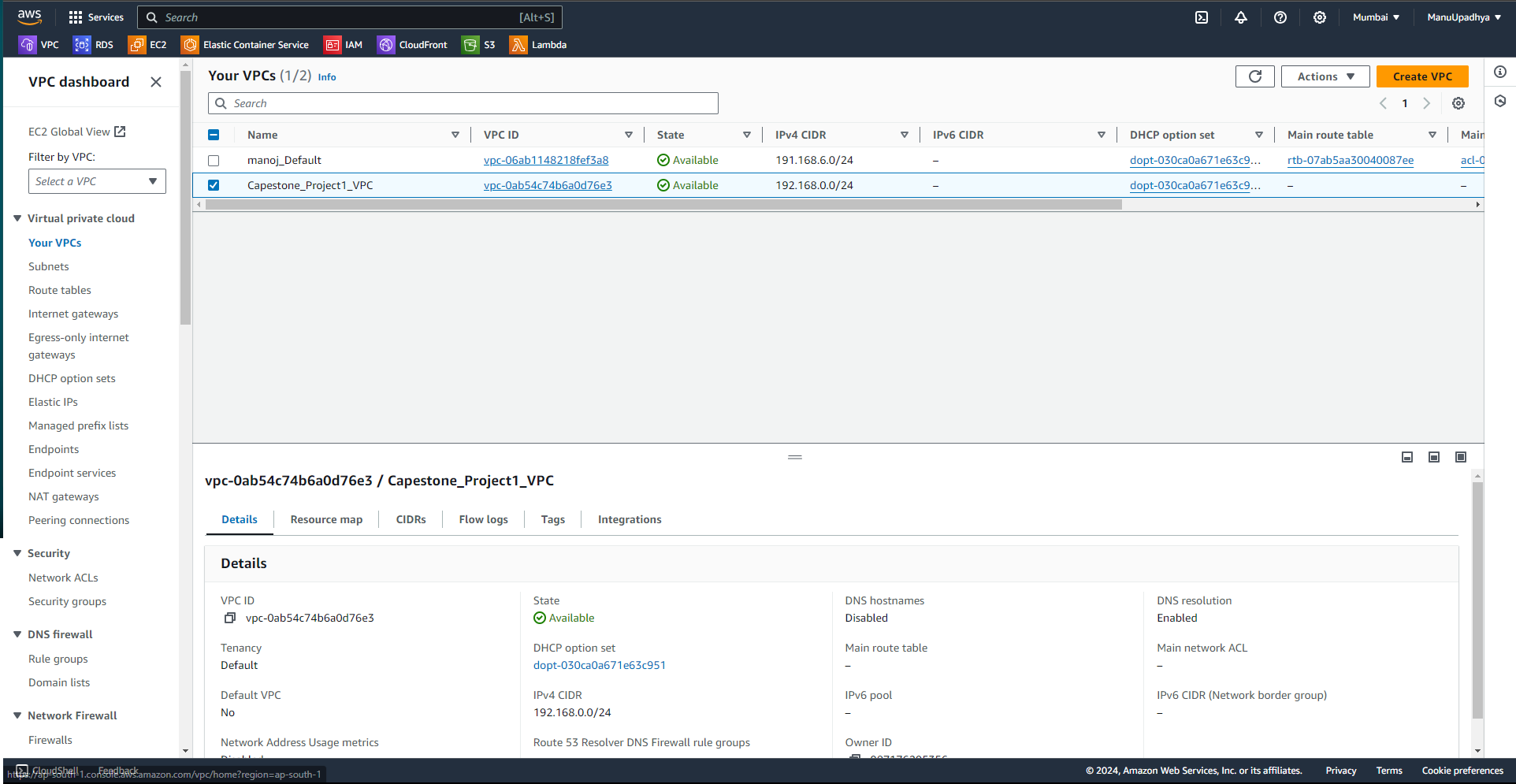
Click on Create VPC🡪 Provide name, IPV4 CIDR block and optional Tags.

A screenshot of a computer

Description automatically generated

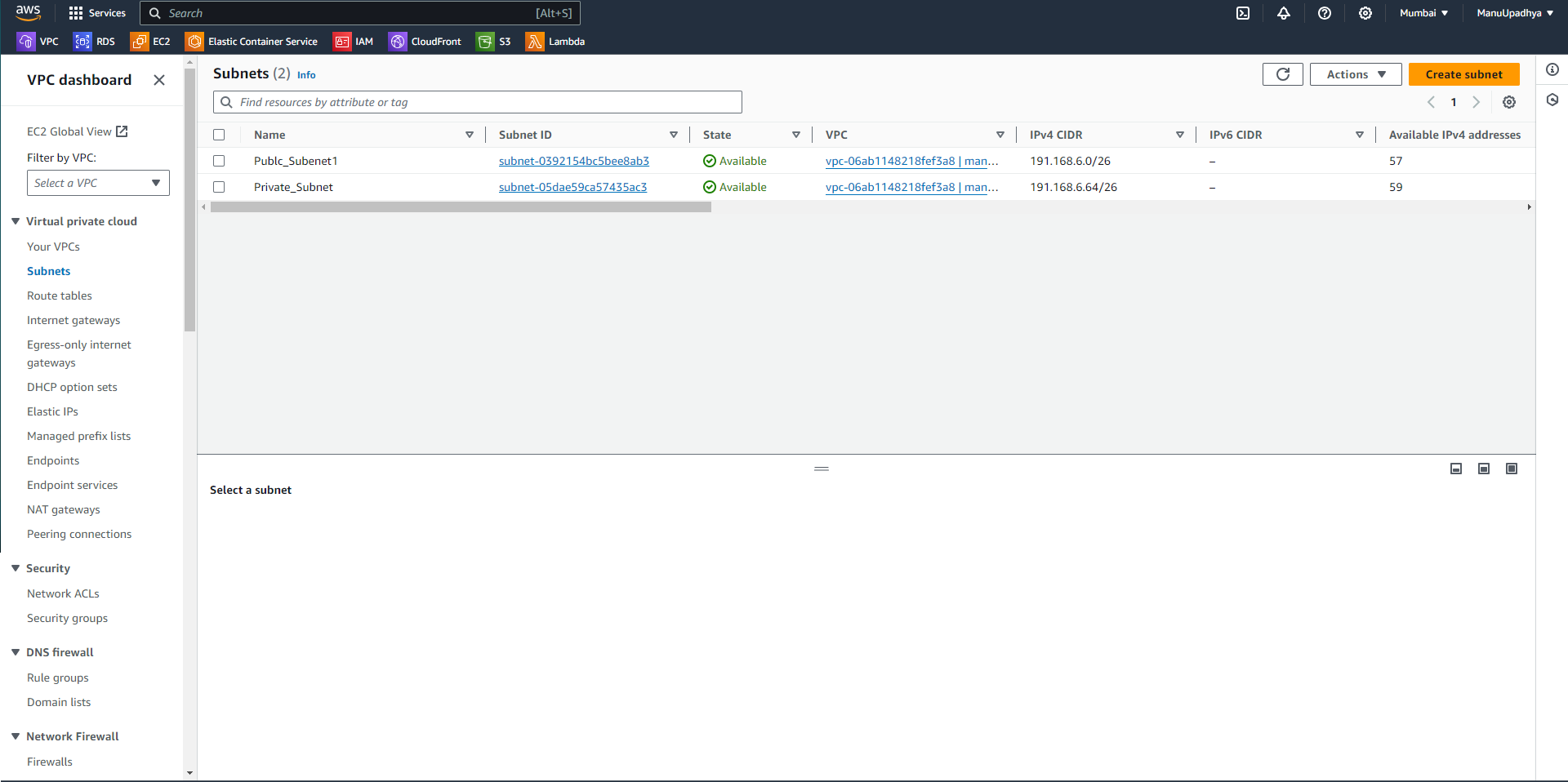
Then click on create VPC.

Navigate back to VPC dashboard to check the created VPC.



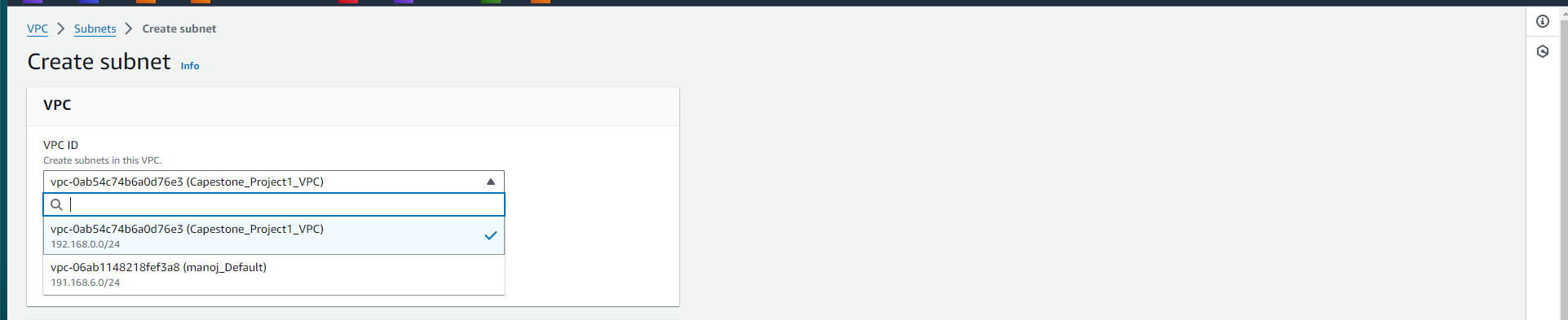
1. **Availability Zones:**
   * The VPC spans two Availability Zones: AP-South-1A and AP-South-1B. These zones ensure high availability and fault tolerance.
2. **Subnets:**
   * **Public Subnets**:
     + AP-South-1A Public Subnet: CIDR block 192.168.0.0/26
     + AP-South-1B Public Subnet: CIDR block 192.168.0.64/26.
     + These subnets contain resources that need to be accessible from the internet, such as EC2 instances with public IP addresses.
   * **Private Subnets**:
     + AP-South-1A Private Subnet: CIDR block 192.168.0.128/26.
     + AP-South-1B Private Subnet: CIDR block 192.168.0.192/26.
     + These subnets host resources that should not be directly accessible from the internet, such as databases.

Now Navigate to VPC🡪Subnets to create public and private subnets.



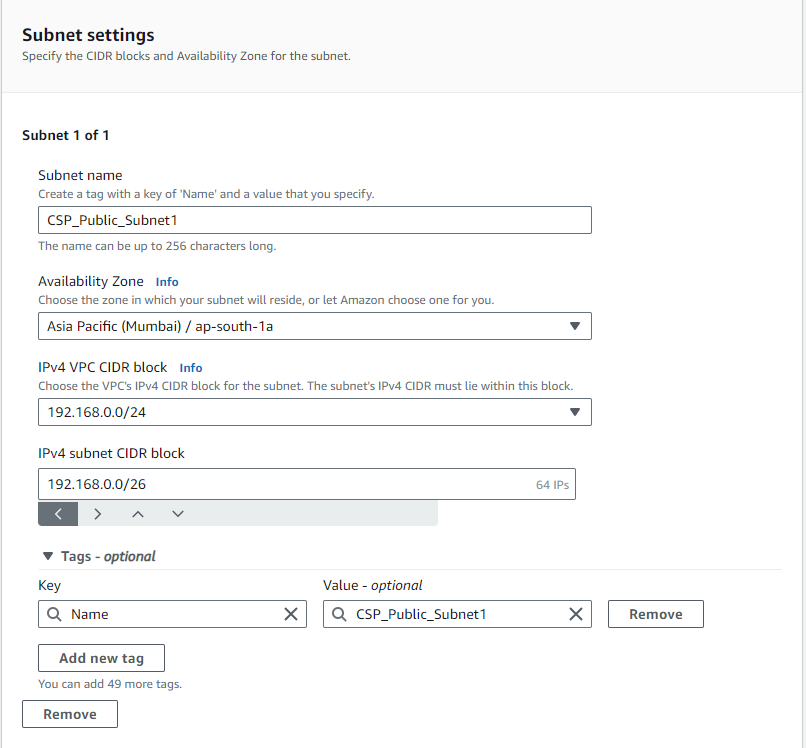
Click on Create Subnet.

Select the VPC ID from the dropdown.



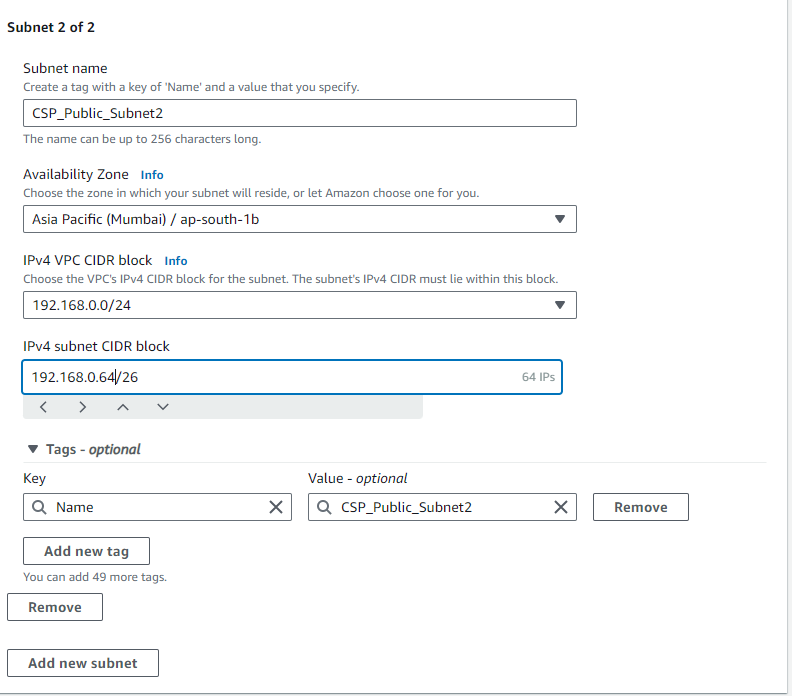
Creation of Public Subnet 1 🡪 Enter Subnet name (CSP\_Public\_Subnet1) and select the Availability Zone (Ap-South-1a) in which the subnet should be created and enter the IPv4 subnet CIDR block.

We can add optional tags if needed.



Click on add new subnet.

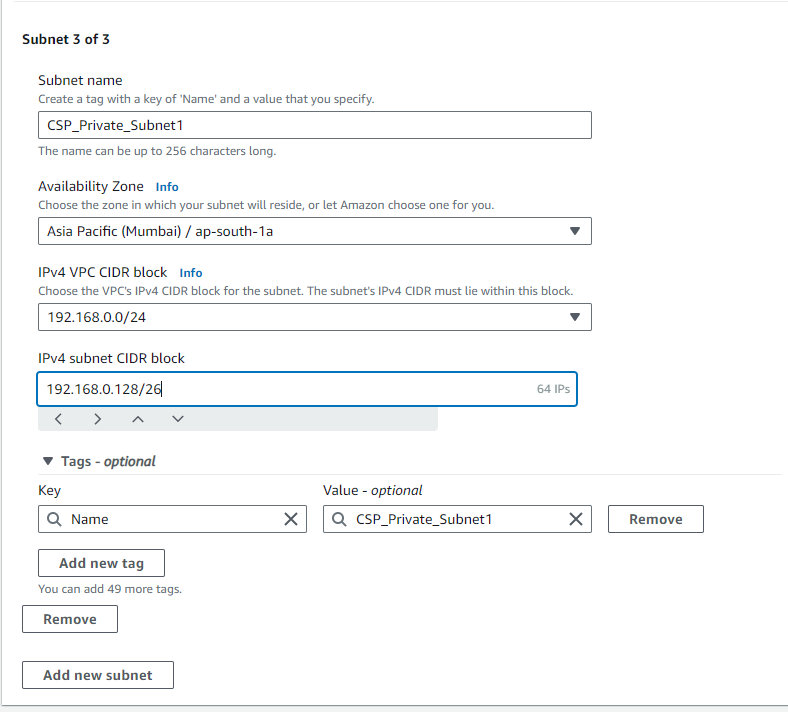
Creation of Public Subnet 2 🡪 Enter Subnet name (CSP\_Public\_Subnet2) and select the Availability Zone (Ap-South-1b) in which the subnet should be created and enter the IPv4 subnet CIDR block.



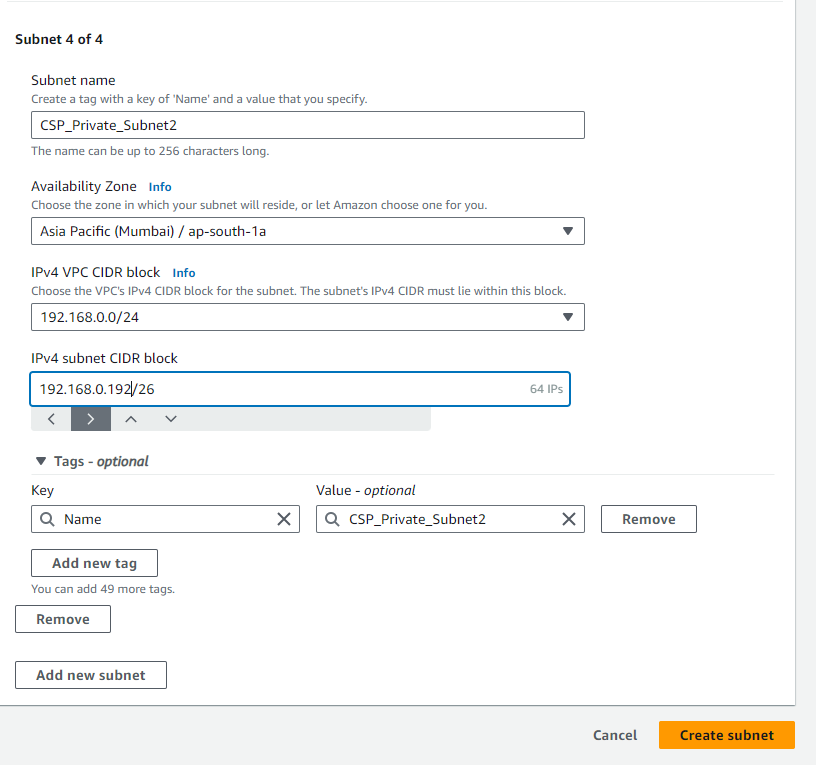
Click on Add new Subnet.

Now we are going to create Private Subnets

Creation of Private Subnet 1 🡪 Enter Subnet name (CSP\_Private\_Subnet1) and select the Availability Zone (Ap-South-1a) in which the subnet should be created and enter the IPv4 subnet CIDR block.

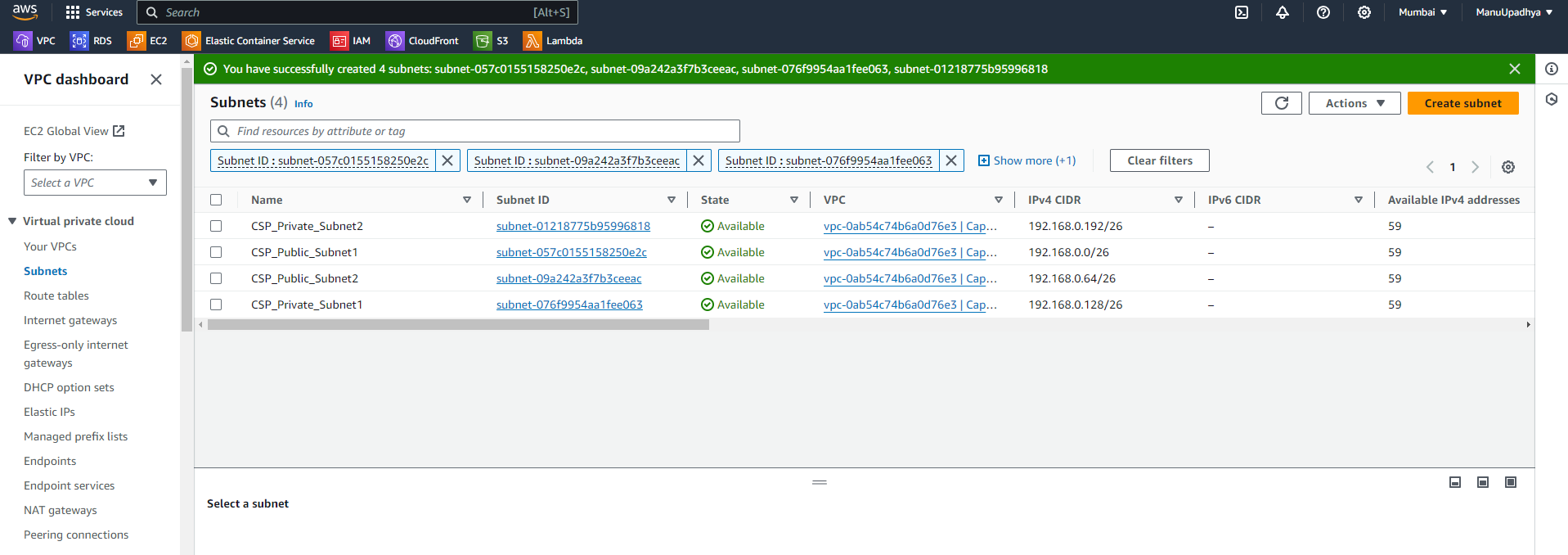


Creation of Private Subnet 2 🡪 Enter Subnet name (CSP\_Private\_Subnet2) and select the Availability Zone (Ap-South-1b) in which the subnet should be created and enter the IPv4 subnet CIDR block.



Click on Create Subnet to Create all four subnets.

We can see all four subnets in Subnet Dashboard.



1. **Internet Gateway:**

Allows communication between instances in the VPC and the internet.

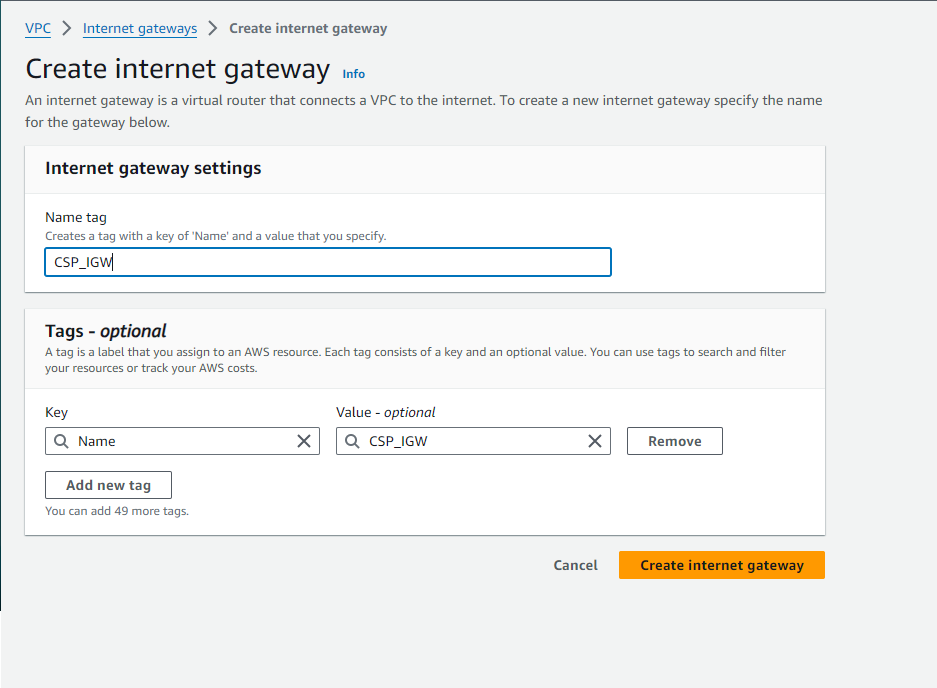
Associated with the VPC to enable direct internet access for resources in public subnets.

Navigate to Internet gateway dashboard from VPC.

A screenshot of a computer

Description automatically generated

Click on create internet gateway 🡪 give name (CSP\_IGW), add an optional tag if needed.



Click on create internet gateway.

We need to attach the created Internet Gateway to VPC.

A screenshot of a computer

Description automatically generated

Select the project VPC.

A screenshot of a computer

Description automatically generated

Click on the attach internet gateway.

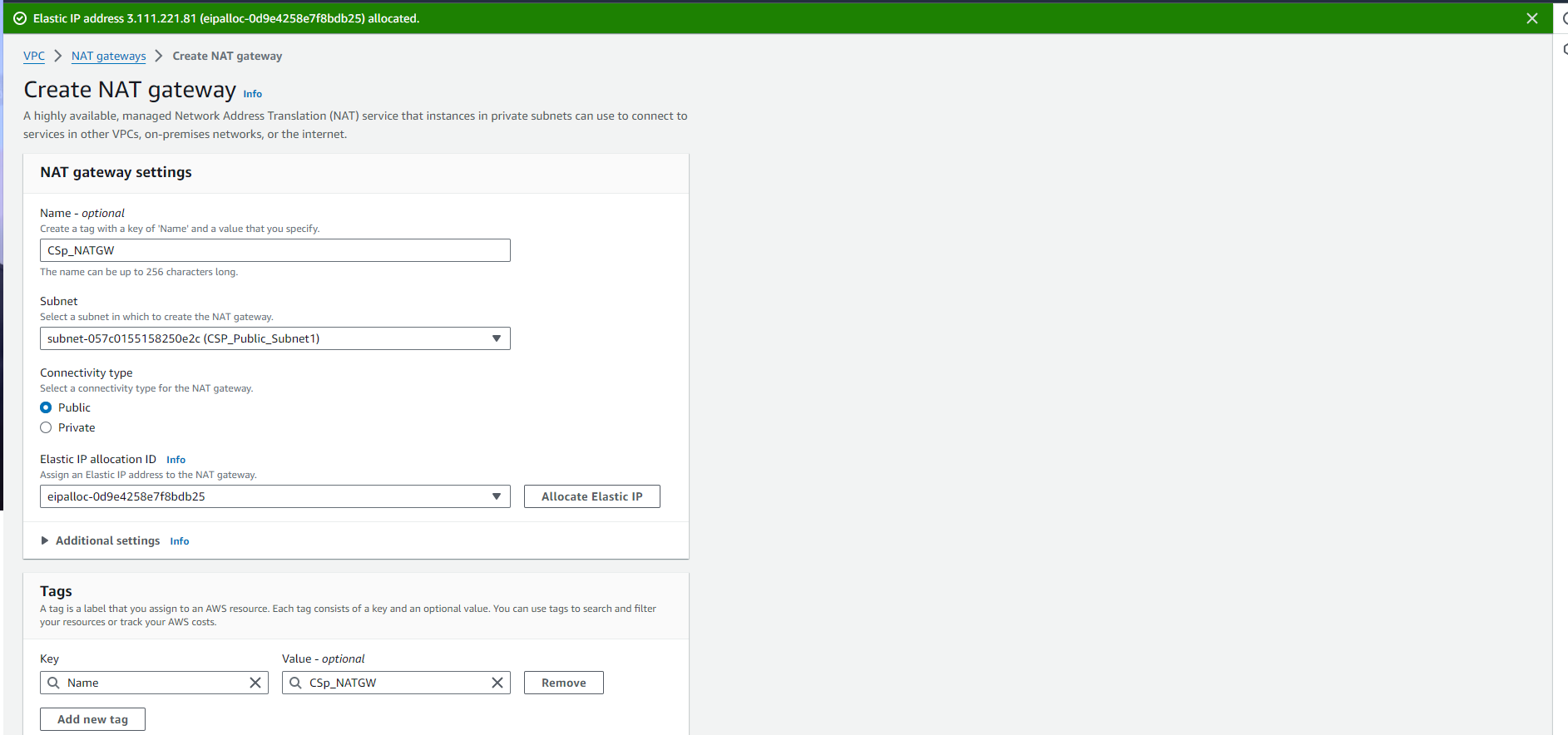
A screenshot of a computer

Description automatically generated

1. **NAT Gateway:**
   * Located in the AP-South-1A Public Subnet.
   * Provides outbound internet access to instances in the private subnets while keeping them secure from incoming internet traffic.
   * It has a public IP address assigned.

Navigate to NAT gateway from VPC Dashboard. Click on create NAT Gateway.

Enter name (CSP\_NATGW) and select the Public Subnet, and click on “allocate Elastic IP” and add optional tag if needed.



Click on Create NAT Gateway.

A screenshot of a computer

Description automatically generated

We can see created NAT Gateway in the NAT Gateway Dashboard. The status of gateway will change from pending to available in 4-5 minutes.

1. **Route Tables:**

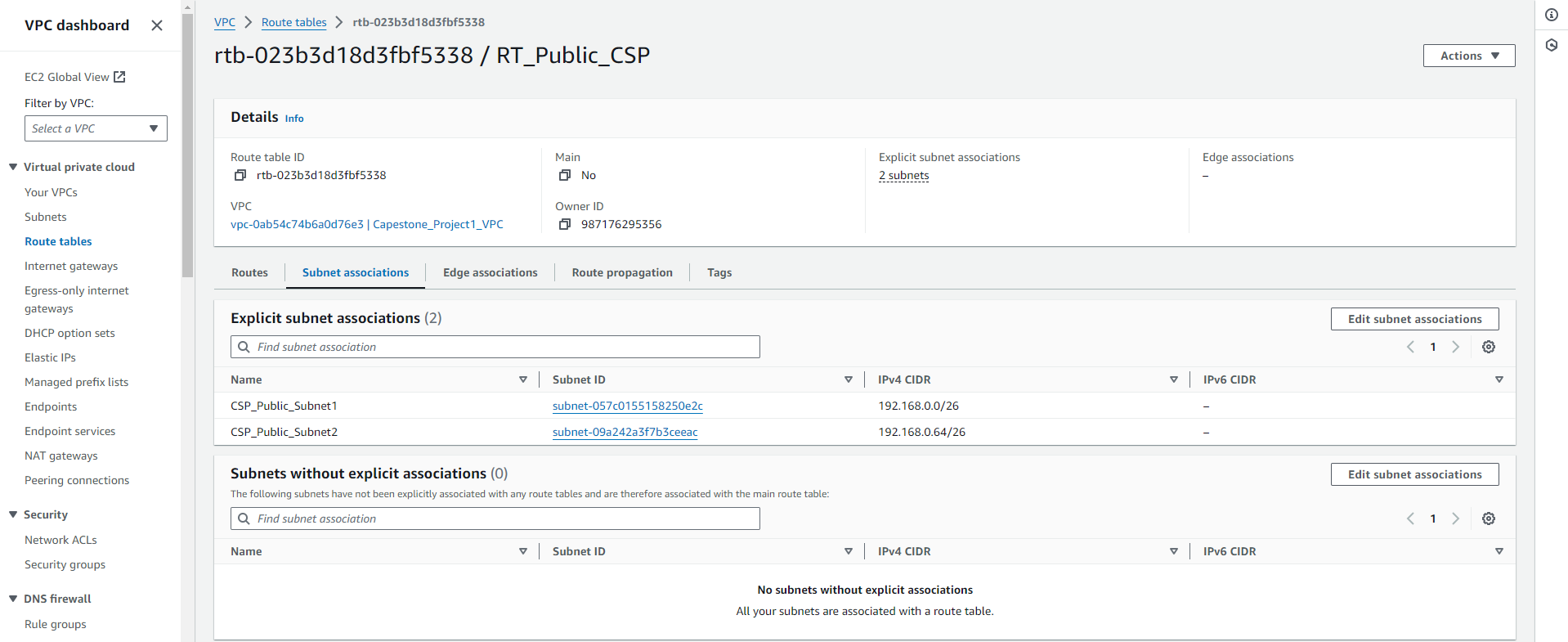
* **Main Route Table (for Private Subnets):**
  + Routes traffic destined for 0.0.0.0/0 (internet) to the NAT Gateway.
  + Ensures that instances in private subnets can access the internet indirectly.
* **Route Table for Public Subnets:**

Directs traffic destined for 0.0.0.0/0 to the internet gateway.

Allows instances in the public subnets to have direct access to and from the internet.

Create two route table MainRT\_CSP for Private Subnets and RT\_Public\_CSP for Public Subnets.

Associate Public Subnets to RT\_Public\_CSP Route table and add route (0.0.0.0/0) via Internet gateway. This means that all network traffic (to and from) internet to subnet route via Internet gateway.



A screenshot of a computer

Description automatically generated

Associate Private Subnets to MainRT\_CSP Route table and add route (0.0.0.0/0) via Nat Gateway. This means that subnet can access the internet, but internet cannot access the subnet. This is the main purpose of using NAT gateway for private subnets.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Phase 1 is almost completed we are left with security group creation we will create security group in future.

**Detailed Flow of Traffic:**

**Outbound Internet Access for Private Subnets**:

Instances in private subnets (e.g., 192.168.0.128/26 and 192.168.0.128/26) send traffic destined for the internet to the NAT Gateway via the main route table.

The NAT Gateway, located in the public subnet of AP-South-1A, translates the private IP addresses to its own public IP address and sends the traffic to the internet.

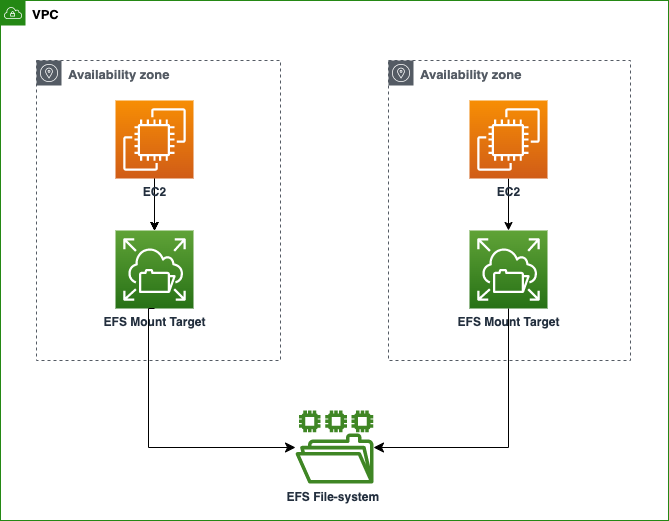
Return traffic from the internet is received by the NAT Gateway, which then translates the public IP address back to the private IP addresses of the instances and routes it accordingly.

**Internet Access for Public Subnets:**

Instances in public subnets (e.g.,192.168.0.0/26 and 192.168.0.64/26) route traffic destined for the internet directly through the internet gateway.

These instances can send and receive traffic directly from the internet as they have public IP addresses assigned.

**Phase 2 Configure EFS**

****

**Creating Security Group for EFS**

Navigate to Security group Dashboard from VPC Dashboard.

A screenshot of a computer

Description automatically generated

Click on Create Security group.

Give name, select VPC, select all traffic allowed inbound rule for as of now we will change this in future.

Created Security Group.

A screenshot of a computer

Description automatically generated

**Creating EFS**

Navigate to EFS from Amazon Management Console.

A screenshot of a computer

Description automatically generated

Click on Create File System.

Give the name for the file system and select the VPC. Then click on Customize.

A screenshot of a computer

Description automatically generated

Next step is to Mount Targets. We will select the subnet where we want to place our file system, and select the created EFS security group.

A screenshot of a computer

Description automatically generated

Click on next, next then create.

It will take some time to create a file system and get status available.

To check whether the EFS was created correctly and mounted properly we will navigate to EFS, by clicking on file system name in the dashboard, the go to network tab we need to have mount target state as **available** which indicates target mounted correctly.

**A screenshot of a computer

Description automatically generated**

This completes our Phase 2 setup.

**Phase 3 Setup Custom AMI for Auto Scaling**

AWS Auto Scaling is a service that helps you automatically adjust the capacity of your applications running on AWS to maintain steady, predictable performance at the lowest possible cost. It involves scaling in and out (adjusting the number of instances) based on the real-time demand and predefined policies.

Before creating an AMI, we will first create an EC2 instance then from the created EC2 instance we will create an AMI.

Navigate to EC2 Dashboard, click on Launch instance.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Click on Launch instance.

Log into created instance.

**Installing Apache web server**

**A screenshot of a computer

Description automatically generated**

Check whether server installed correctly or not using **systemctl start httpd** command.

Now we will connect the created instance with created EFS file system.

Navigate to EFS🡪select the created file system and click on attach

A screenshot of a computer

Description automatically generated

Now navigate to ec2 instance command line and run the command.

A black screen with white text

Description automatically generated

Now Navigate inside the /var/www/html/ and copy the web application.

A black screen with a black background

Description automatically generated

Sample Web application created.

The last step remaining is we need to mount the EFS permanently.

Navigate to nano /etc/fstab and add the text below into the document.

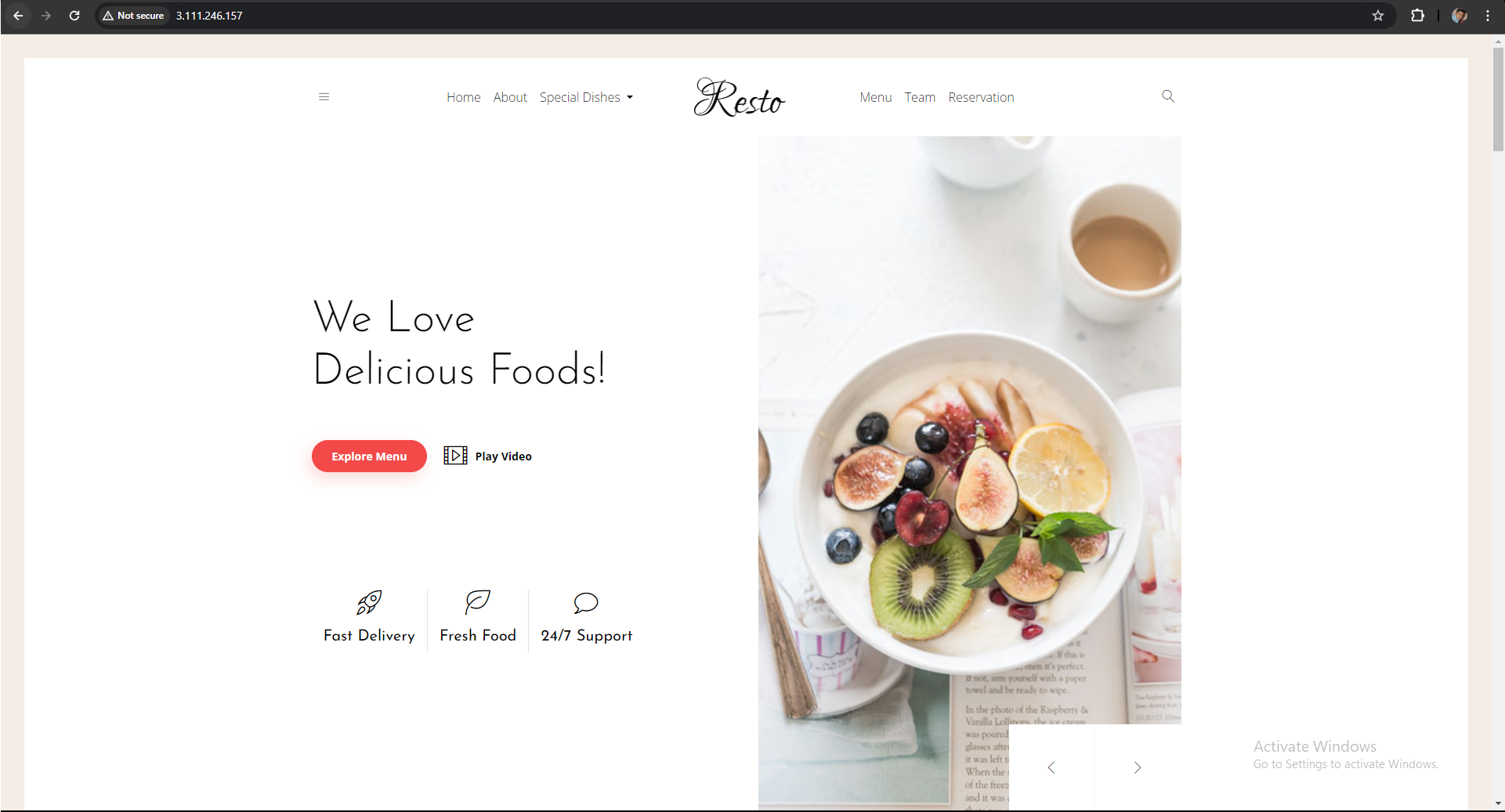
**file-system-id.efs.aws-region.amazonaws.com:/ /mount/point efs defaults,\_netdev 0 0**

A black background with white text

Description automatically generated

Save and exit the file.

Now we need to verify our hosted web site running correctly or not.



I can open my static website using ip address. If stop and start instance, then also I am getting the website. Indicating that EFS is working correctly.

Creation of AMI from this web server: -

Now select the instance🡪actions🡪Image and templates 🡪Create Image.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Click on create Image.

We can see the created image in AMI Section of EC2 Dashboard.

A screenshot of a computer

Description automatically generated

Now delete the instance since it is not required for future, we create instances using created AMI.

Phase 3 is completed here.