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.

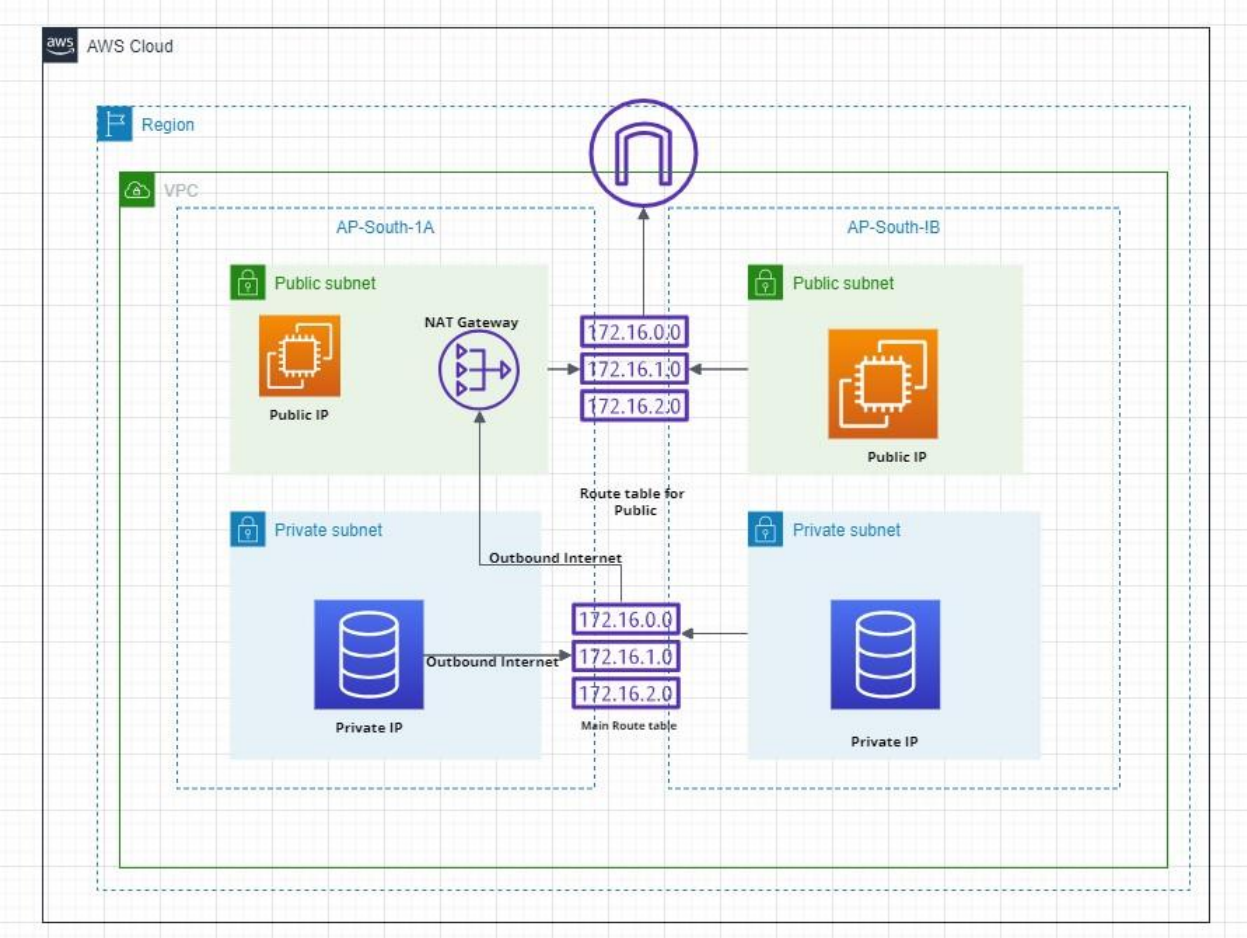
Project phases

1. **Design Phase** – Architect the solution, focusing on the application's **Security, Scalability, Availability requirements.**
2. **Implementation Phase** –
    a. Create the VPC, subnets, and security groups.
    b. Configure EFS.
    c. Setup Custom AMI For Auto Scaling.
    d. Set up and test Auto Scaling.
    e. Deploy the ALB.
    f. 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



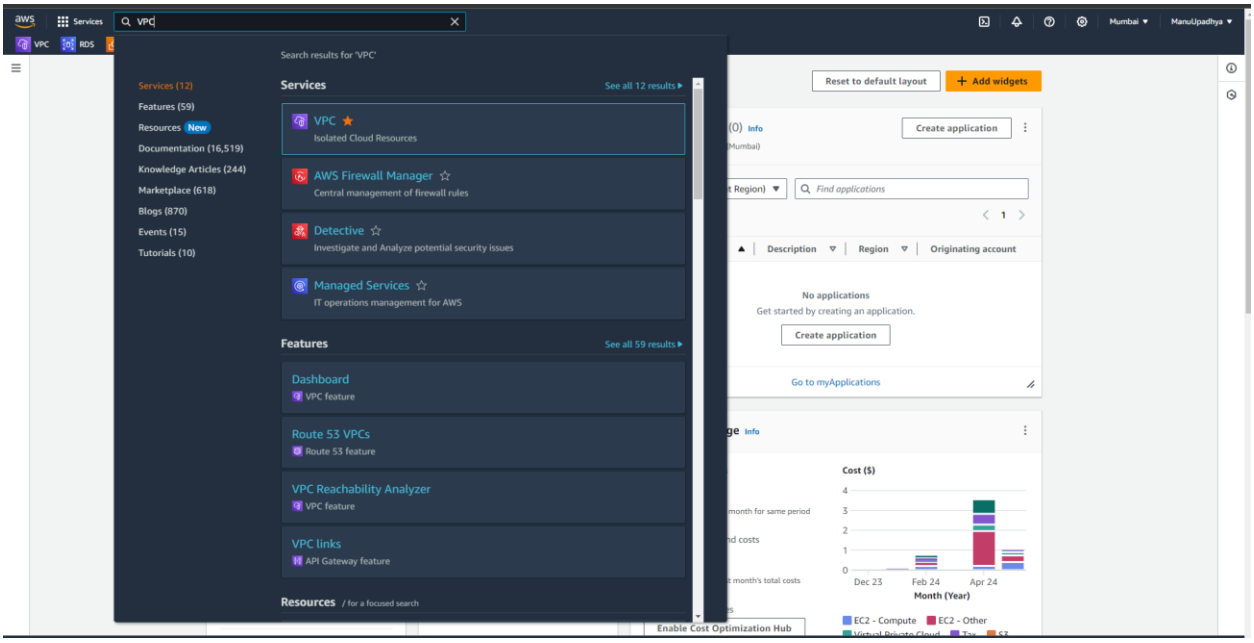
This AWS architecture is designed for high availability and resilience by leveraging multiple Availability Zones (AZs) within a VPC. The VPC spans two AZs (AP-South-1A and AP-South-1B), each containing both public and private subnets. Public subnets in each AZ (192.168.0.0/26 and 192.168.0.64/26) host 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:

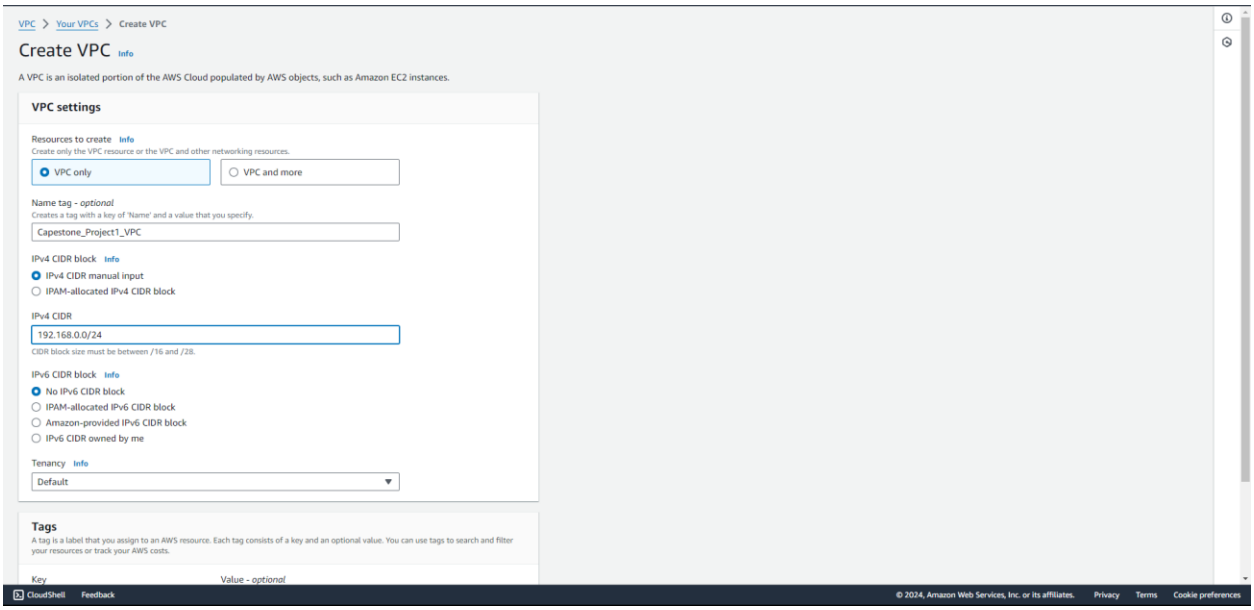
Components:

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.

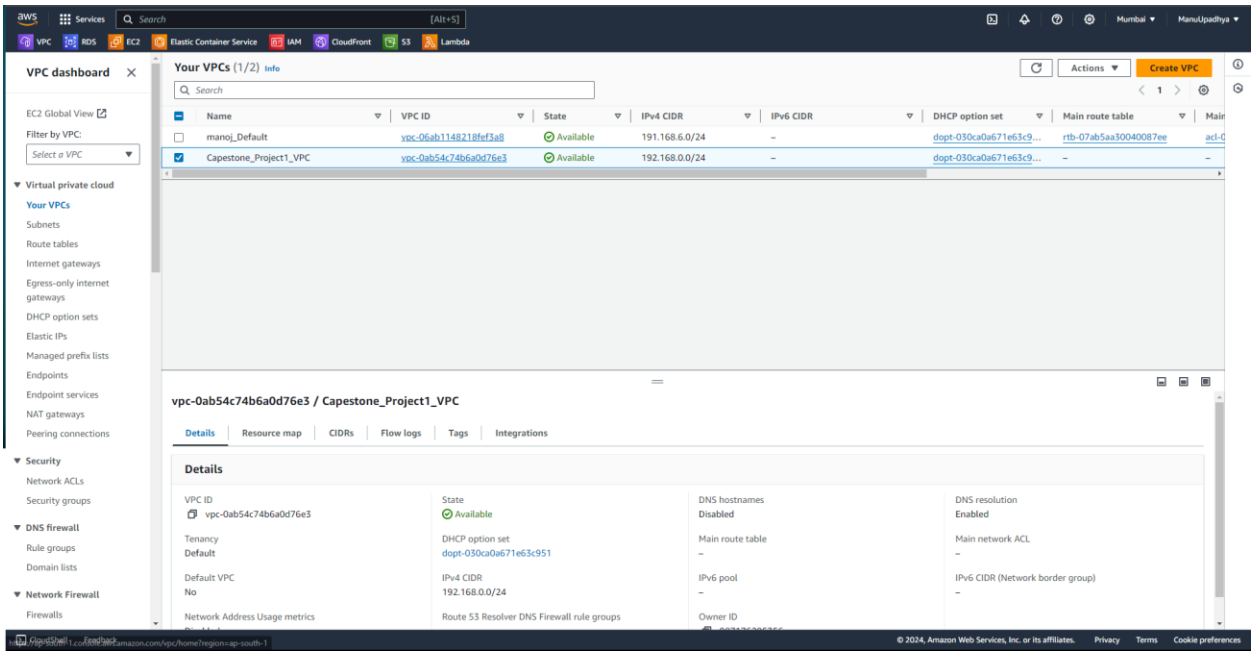


Click on Create VPC→ Provide name, IPV4 CIDR block and optional Tags.



Then click on create VPC.

Navigate back to VPC dashboard to check the created VPC.

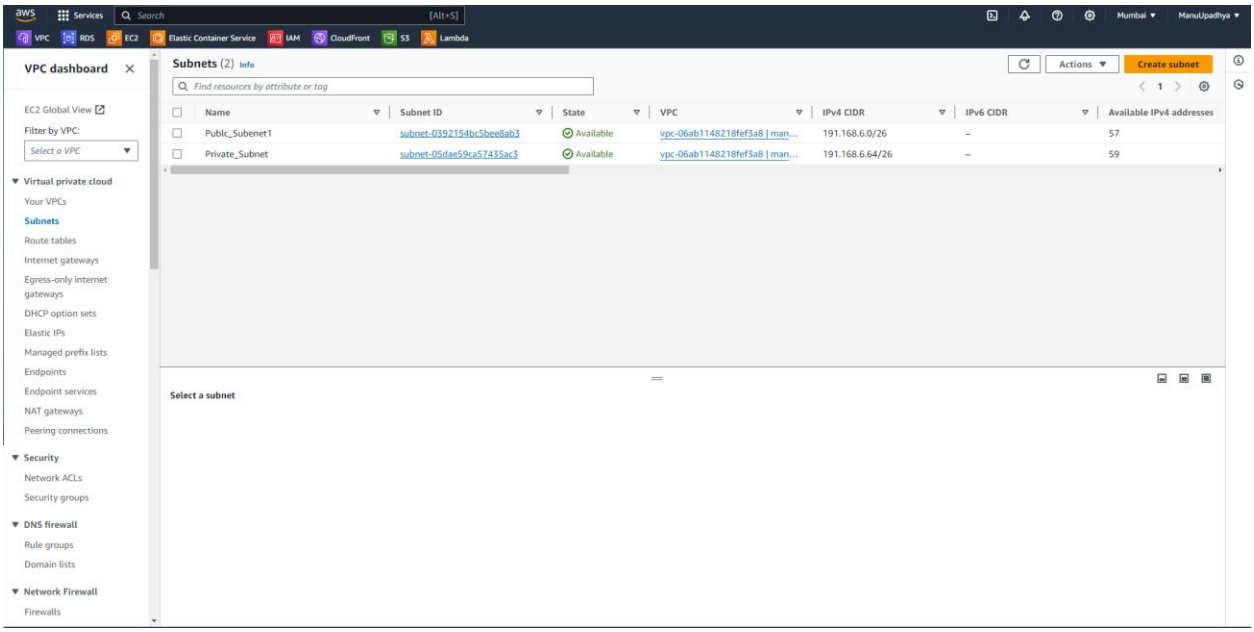


2. **Availability Zones:**
    - The VPC spans two Availability Zones: AP-South-1A and AP-South-1B. These zones ensure high availability and fault tolerance.
3. **Subnets:**
    - **Public Subnets**:

- 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.

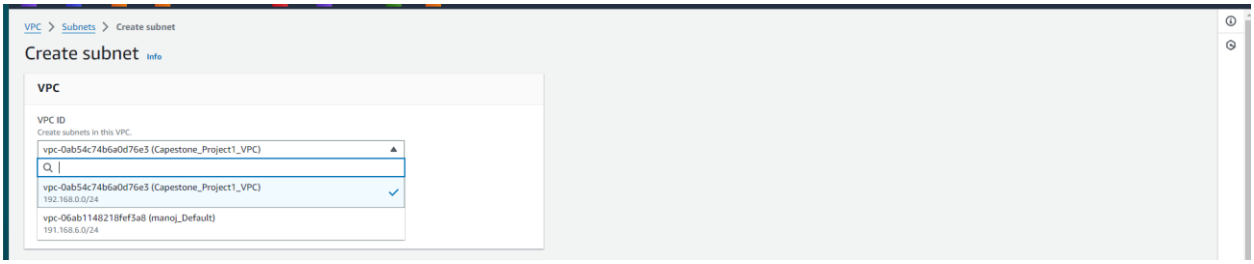
- **Private Subnets**:
  - AP-South-1A Private Subnet: CIDR block 192.168.0.128/26.
  - AP-South-1B Private Subnet: CIDR block 192.168.0.192/26.
  - These subnets host resources that should not be directly accessible from the internet, such as databases.

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



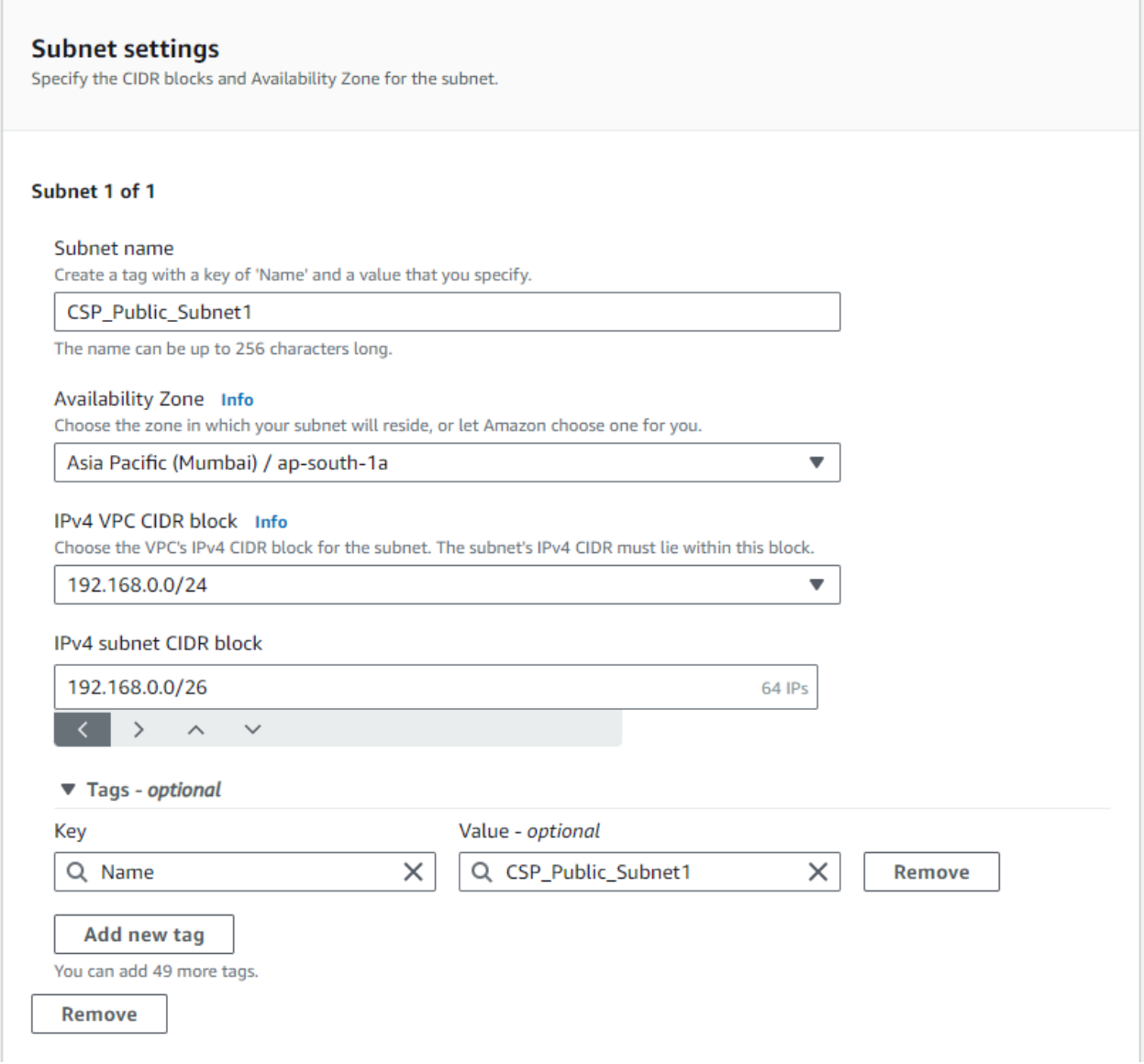
Click on Create Subnet.

Select the VPC ID from the dropdown.



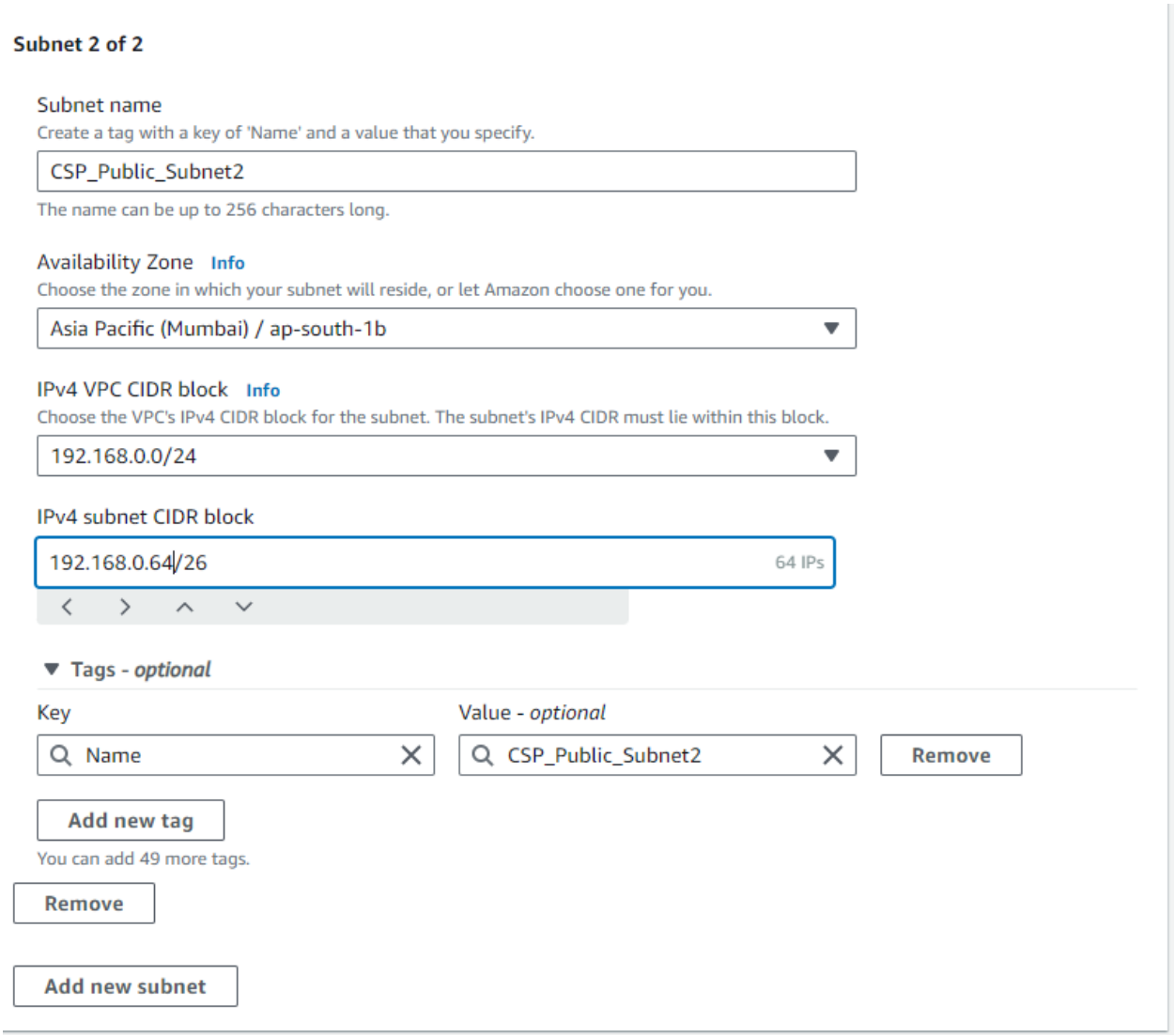
Creation of Public Subnet 1 → Enter Subnet name (CSP_Public_Subnet1) and select the Availability Zone (Ap-South-1a) in which the subnet should be created and enter the IPv4 subnet CIDR block.

We can add optional tags if needed.



Click on add new subnet.

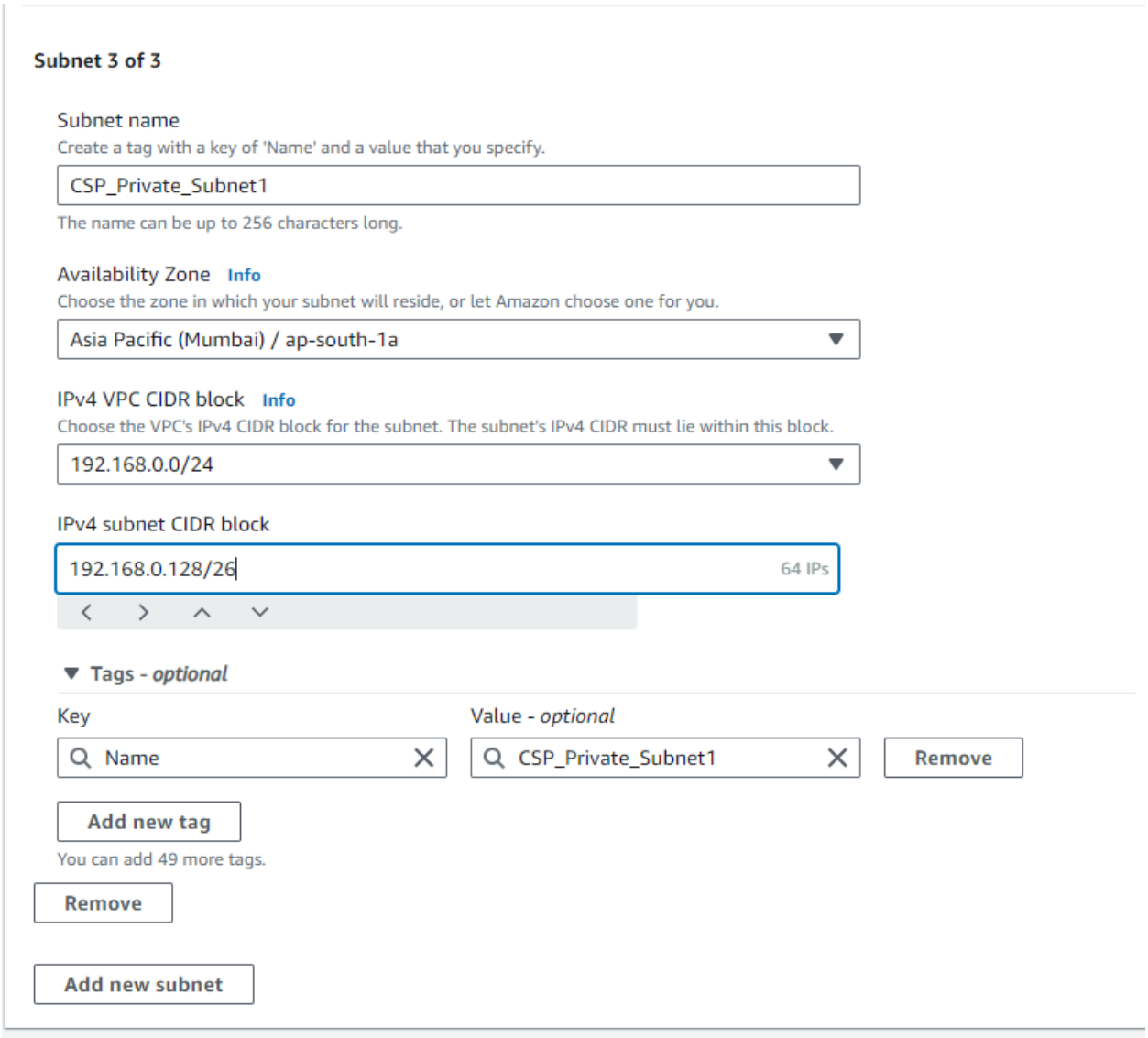
Creation of Public Subnet 2 → Enter Subnet name (CSP_Public_Subnet2) and select the Availability Zone (Ap-South-1b) in which the subnet should be created and enter the IPv4 subnet CIDR block.



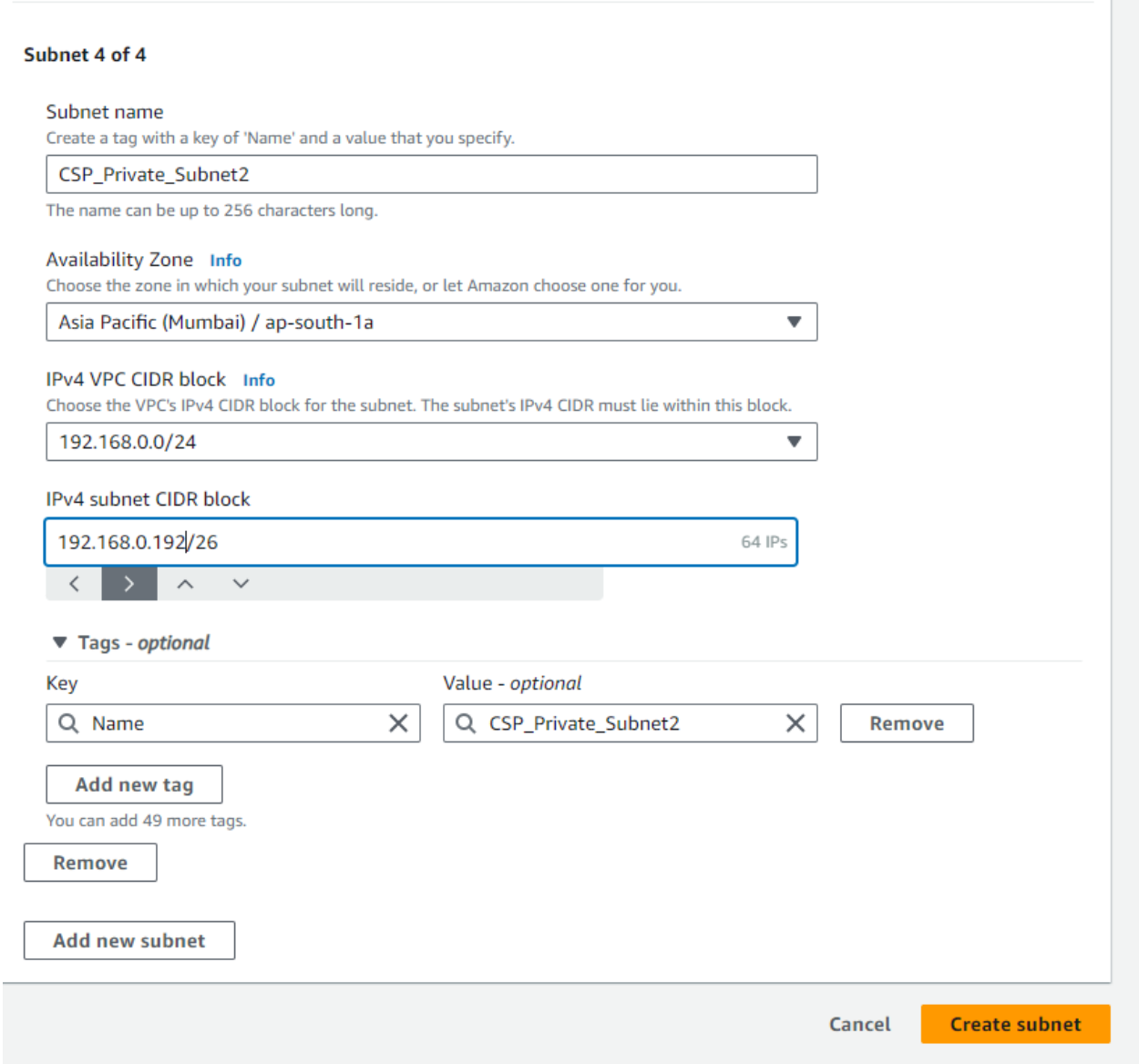
Click on Add new Subnet.

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.

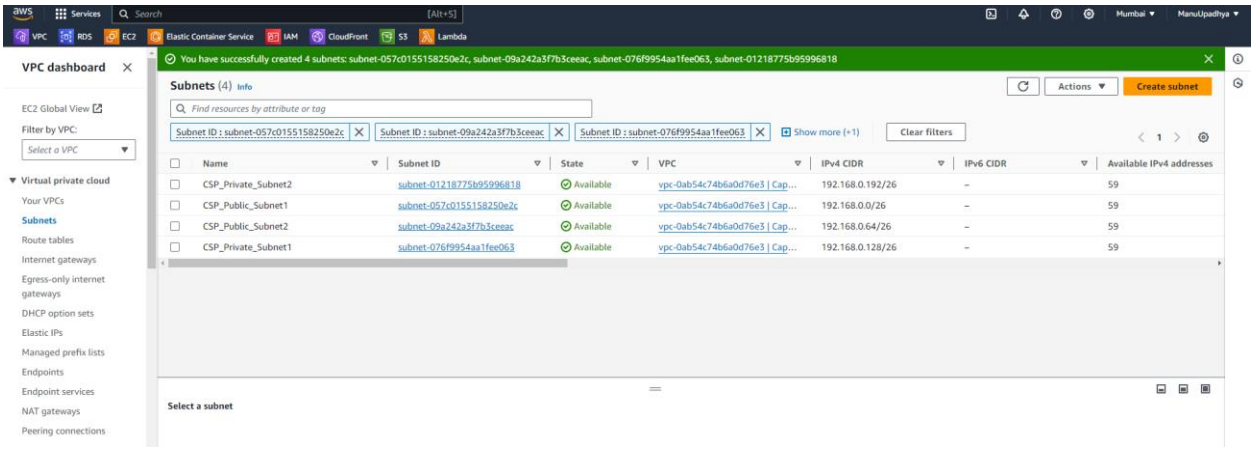


Creation of Private Subnet 2 → Enter Subnet name (CSP_Private_Subnet2) and select the Availability Zone (Ap-South-1b) in which the subnet should be created and enter the IPv4 subnet CIDR block.



Click on Create Subnet to Create all four subnets.

We can see all four subnets in Subnet Dashboard.

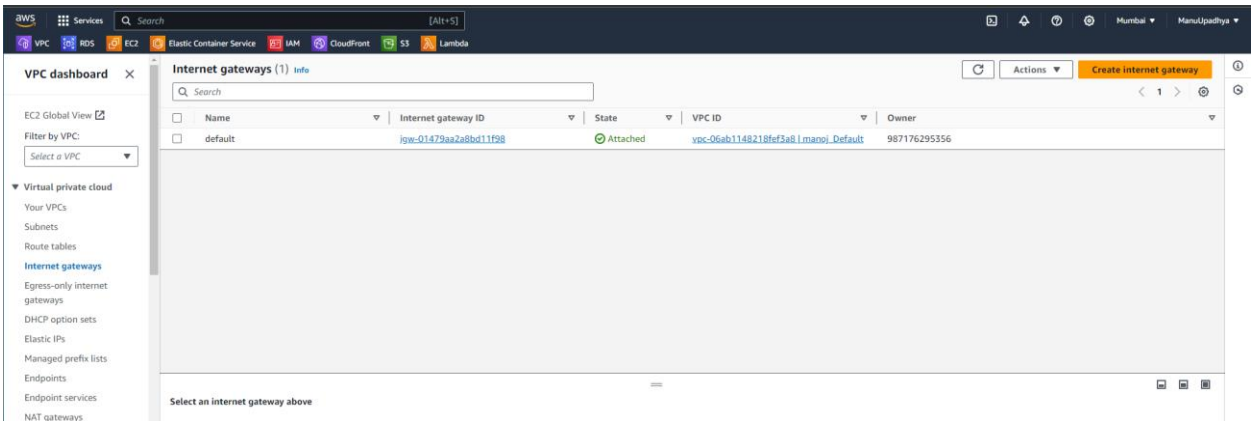


4. Internet Gateway:

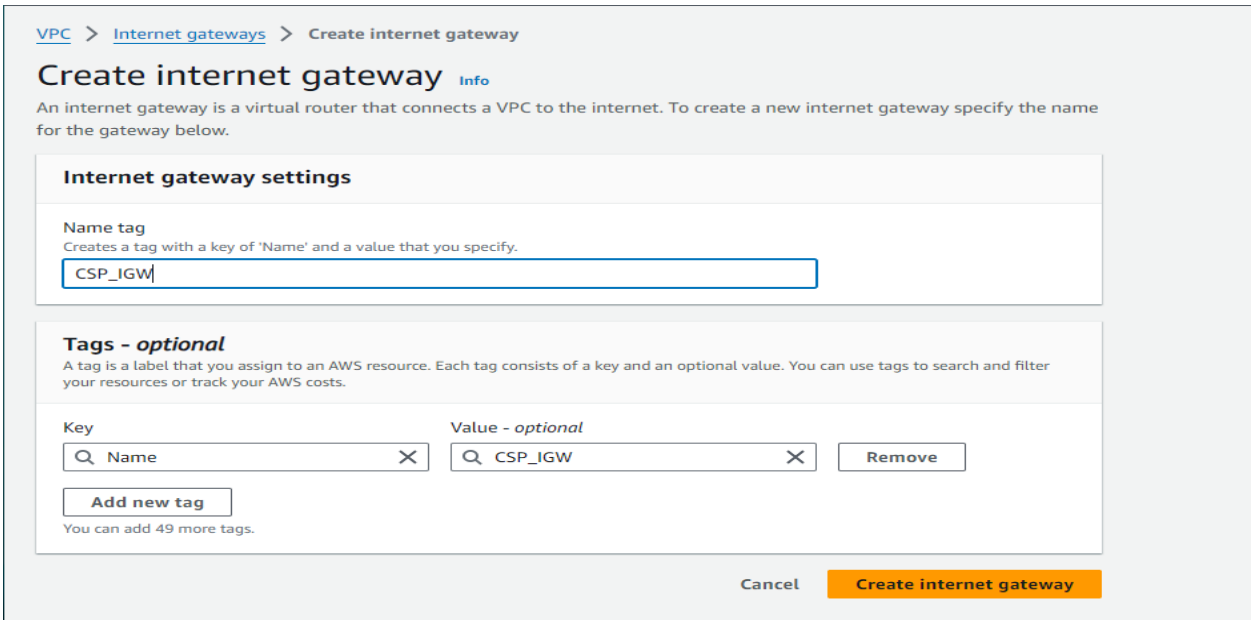
Allows communication between instances in the VPC and the internet.

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

Navigate to Internet gateway dashboard from VPC.

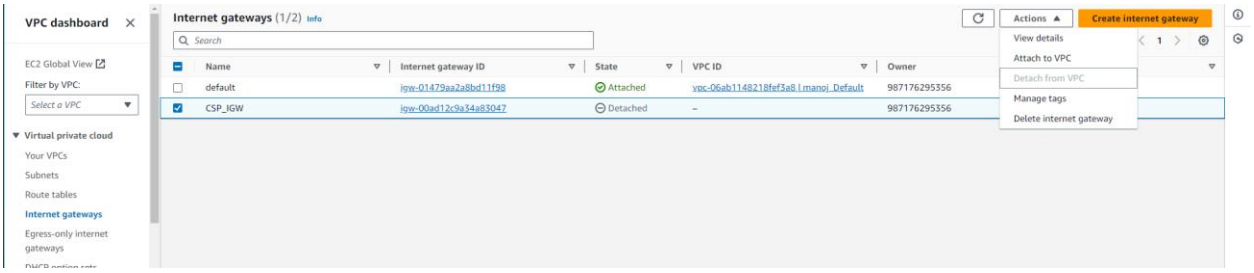


Click on create internet gateway → give name (CSP_IGW), add an optional tag if needed.

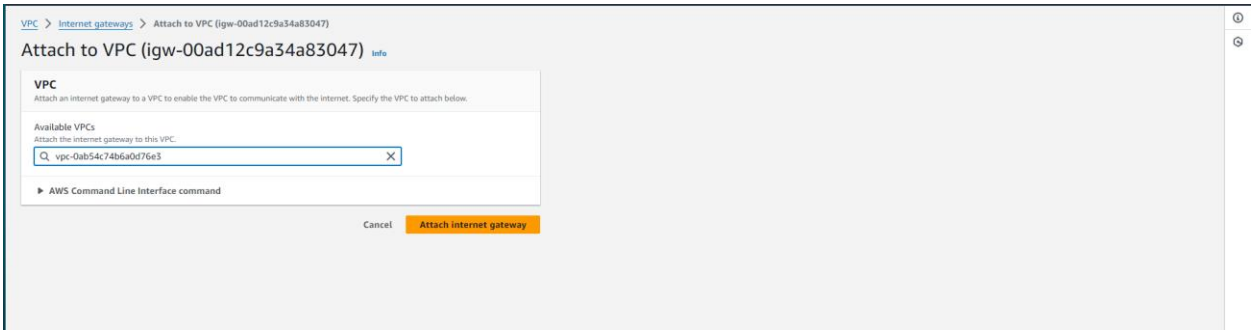


Click on create internet gateway.

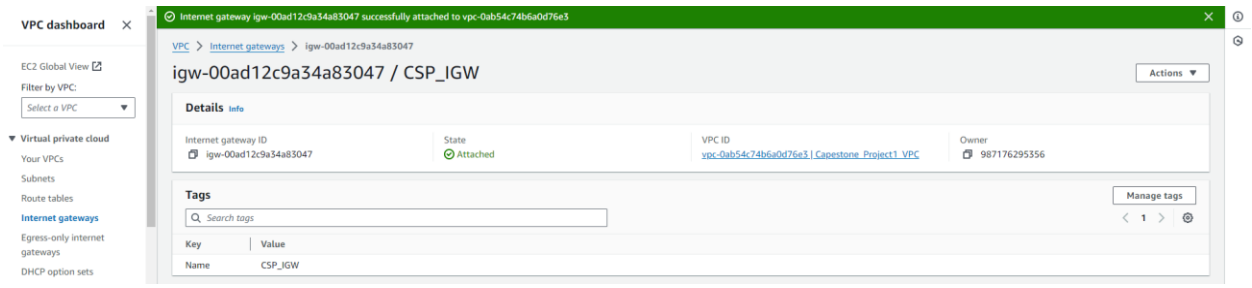
We need to attach the created Internet Gateway to VPC.



Select the project VPC.



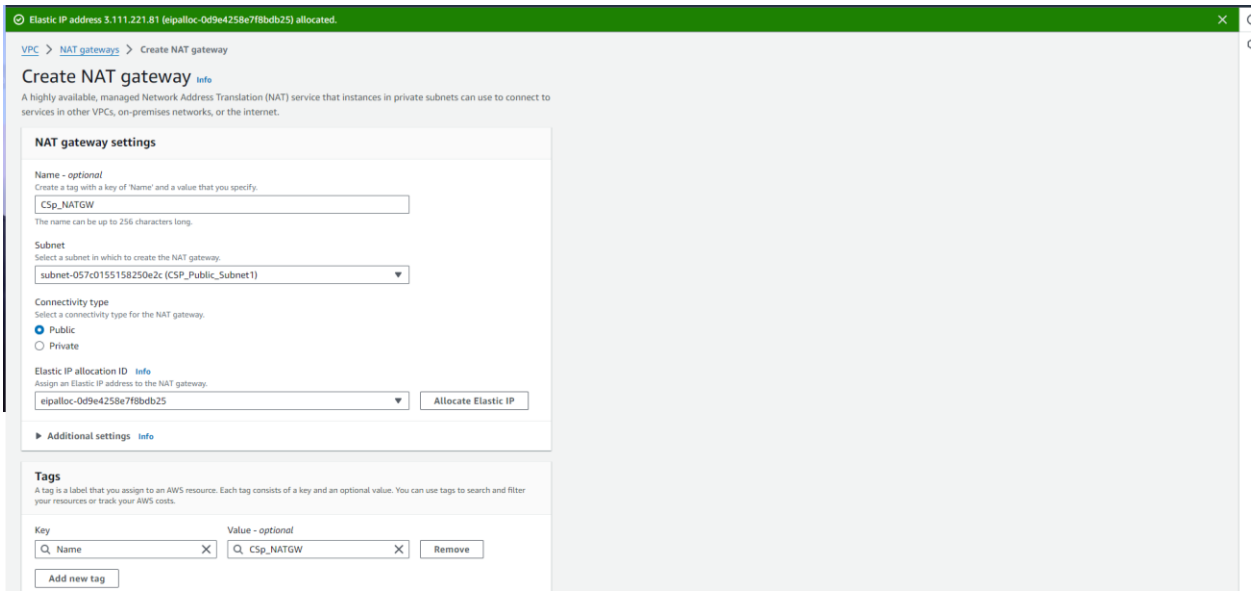
Click on the attach internet gateway.



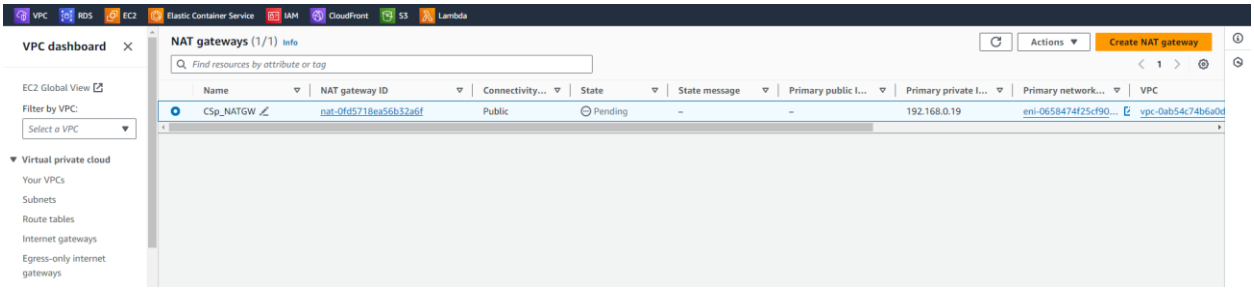
5. **NAT Gateway:**
    - Located in the AP-South-1A Public Subnet.
    - Provides outbound internet access to instances in the private subnets while keeping them secure from incoming internet traffic.
    - It has a public IP address assigned.

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.



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.

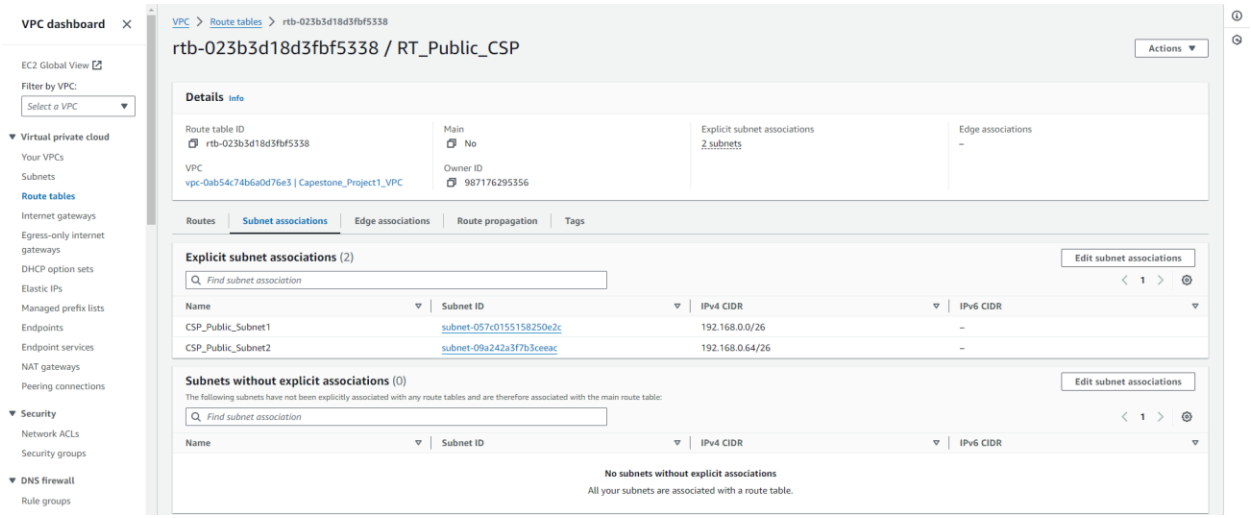
6. **Route Tables:**
    - **Main Route Table (for Private Subnets):**
        - Routes traffic destined for 0.0.0.0/0 (internet) to the NAT Gateway.
        - Ensures that instances in private subnets can access the internet indirectly.
    - **Route Table for Public Subnets:**

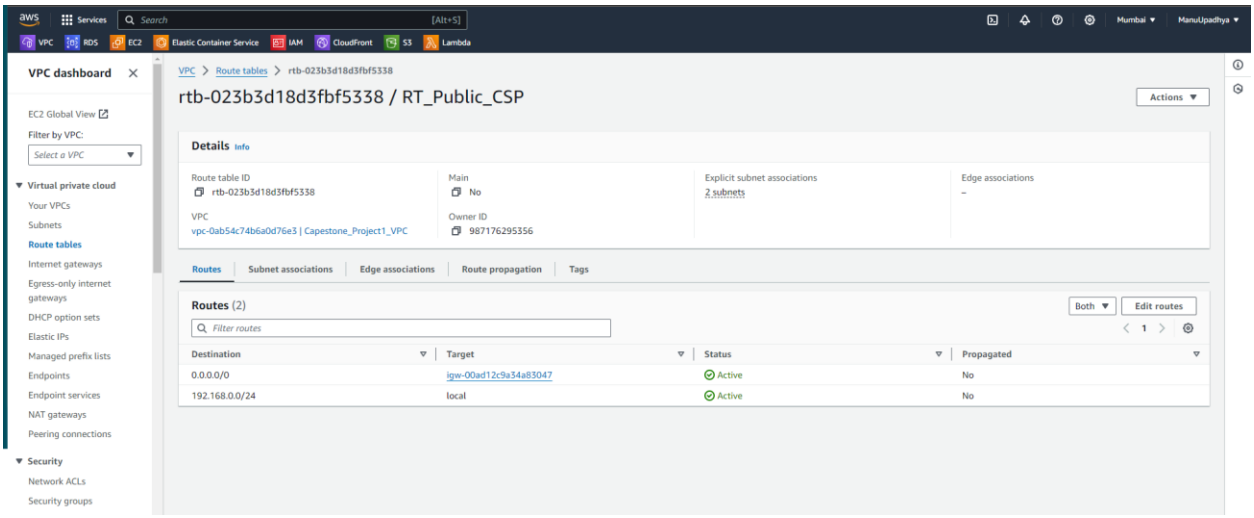
      Directs traffic destined for 0.0.0.0/0 to the internet gateway.

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

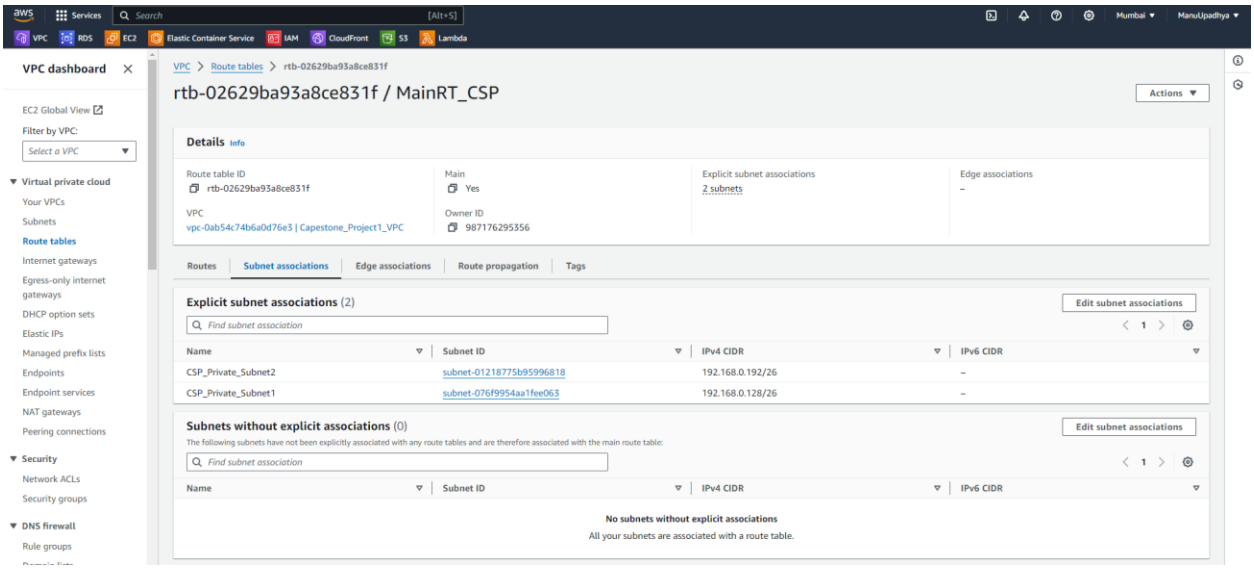
Create two route table MainRT_CSP for Private Subnets and RT_Public_CSP for Public Subnets.

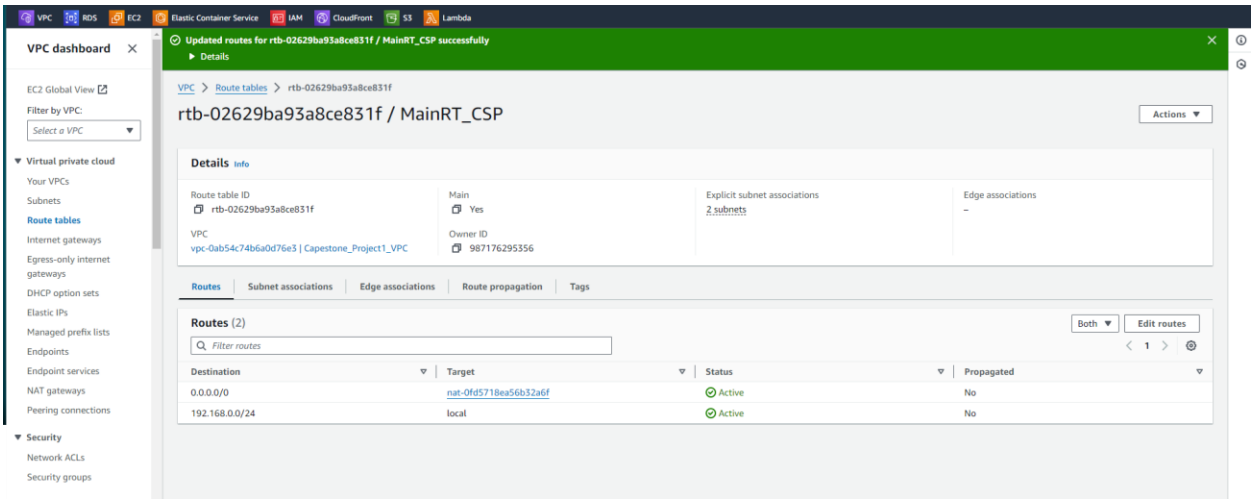
Associate Public Subnets to RT_Public_CSP Route table and add route (0.0.0.0/0) via Internet gateway. This means that all network traffic (to and from) internet to subnet route via Internet gateway.





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.





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.

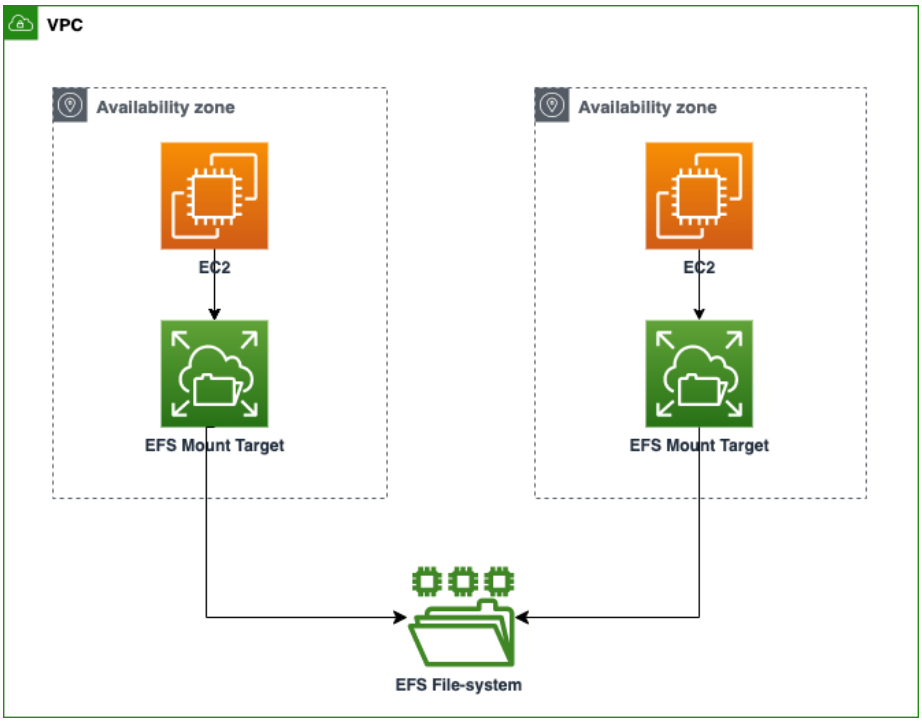
Return traffic from the internet is received by the NAT Gateway, which then translates the public IP address back to the private IP addresses of the instances and routes it accordingly.

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.

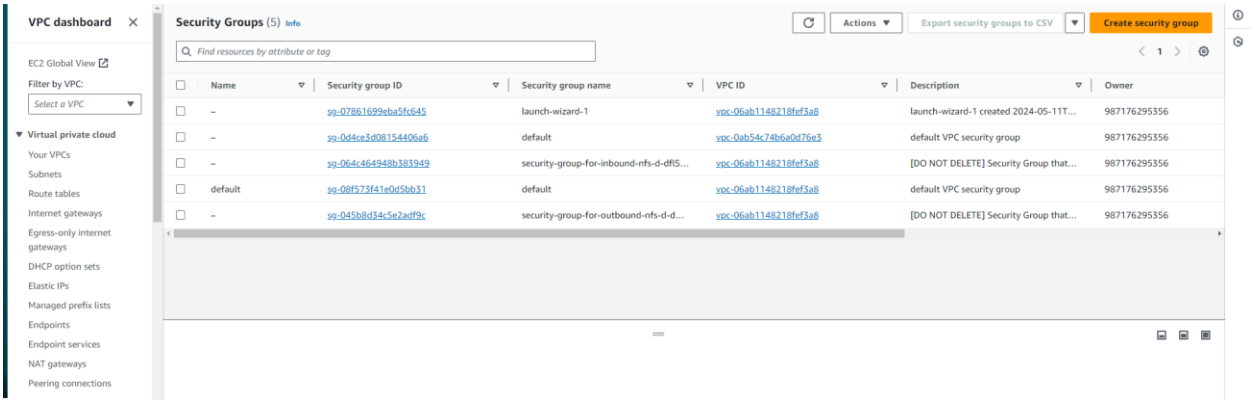
These instances can send and receive traffic directly from the internet as they have public IP addresses assigned.

Phase 2 Configure EFS



Creating Security Group for EFS

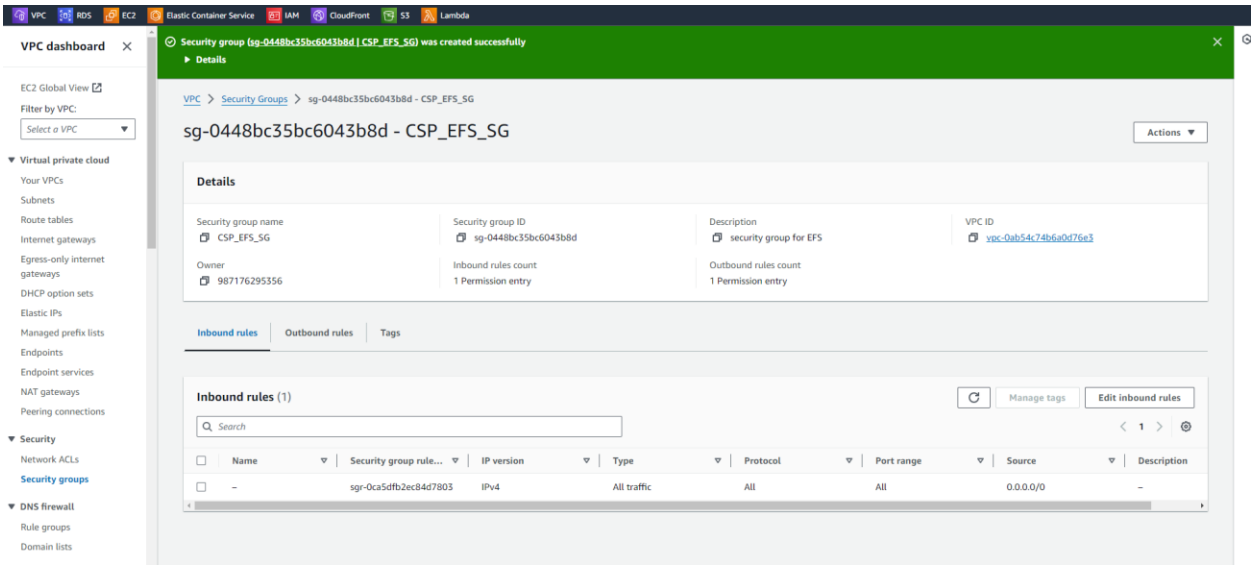
Navigate to Security group Dashboard from VPC Dashboard.



Click on Create Security group.

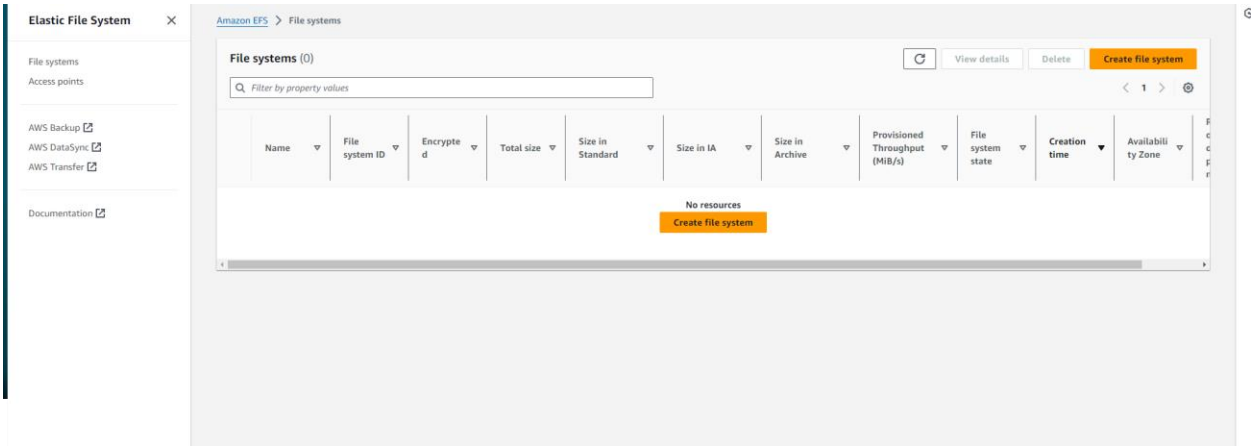
Give name, select VPC, select all traffic allowed inbound rule for as of now we will change this in future.

Created Security Group.



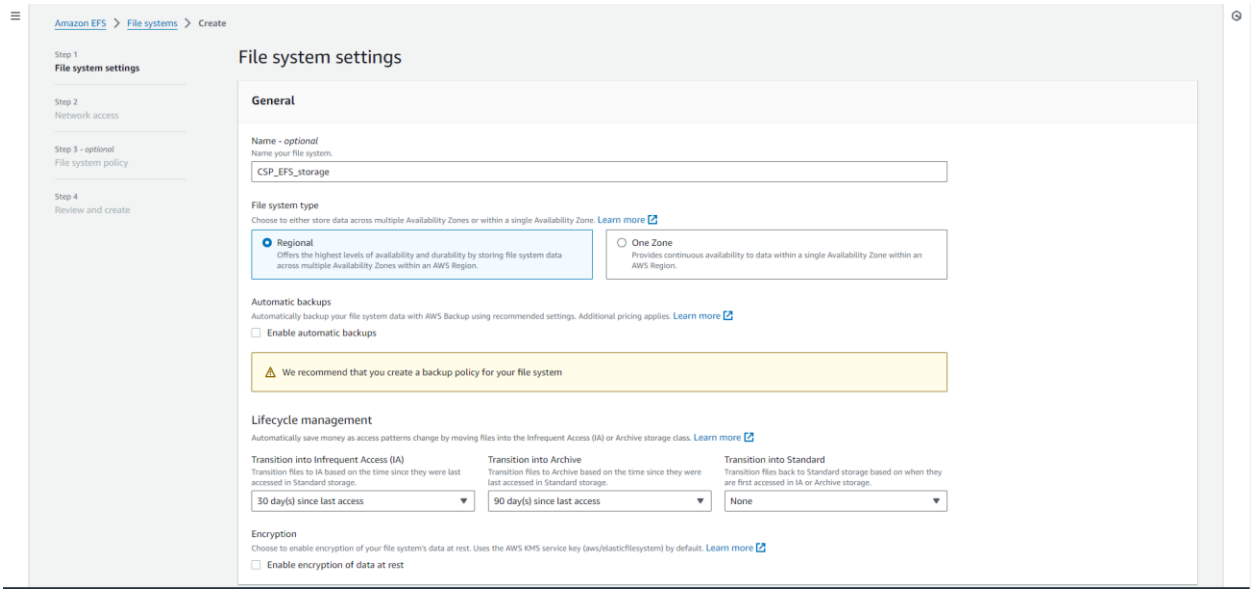
Creating EFS

Navigate to EFS from Amazon Management Console.

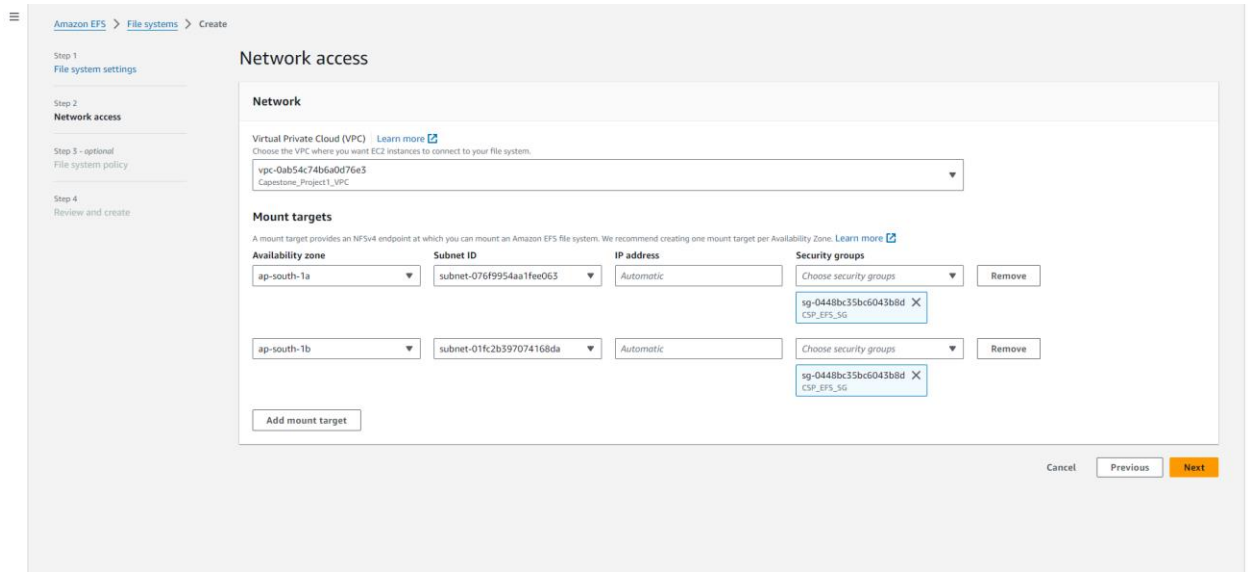


Click on Create File System.

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



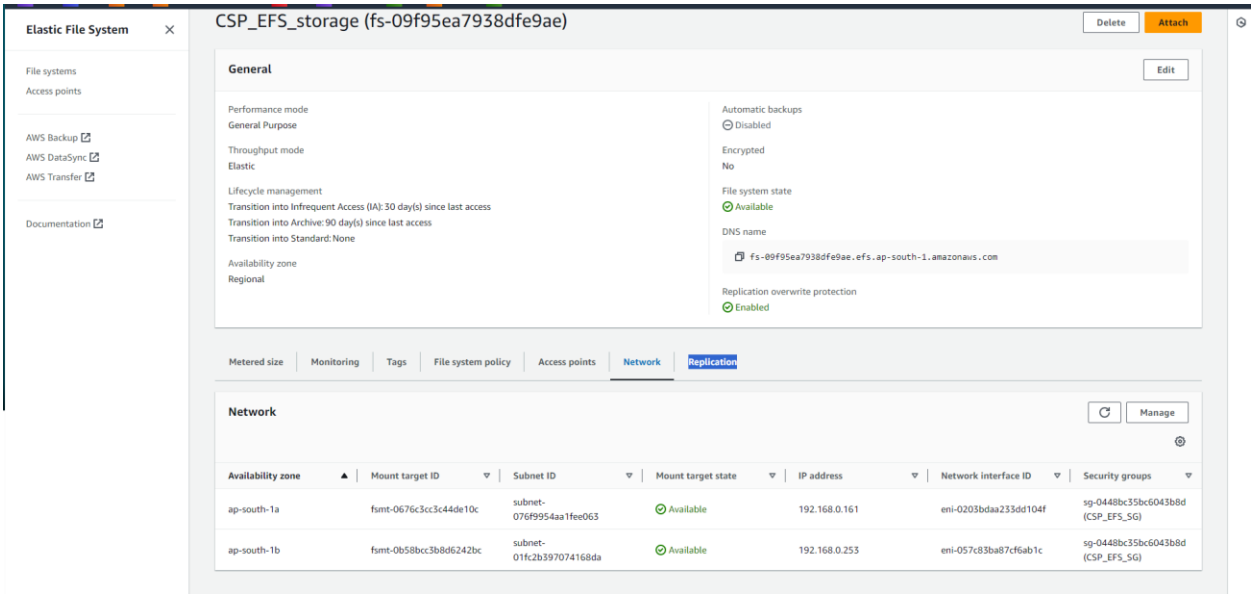
Next step is to Mount Targets. We will select the subnet where we want to place our file system, and select the created EFS security group.



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.



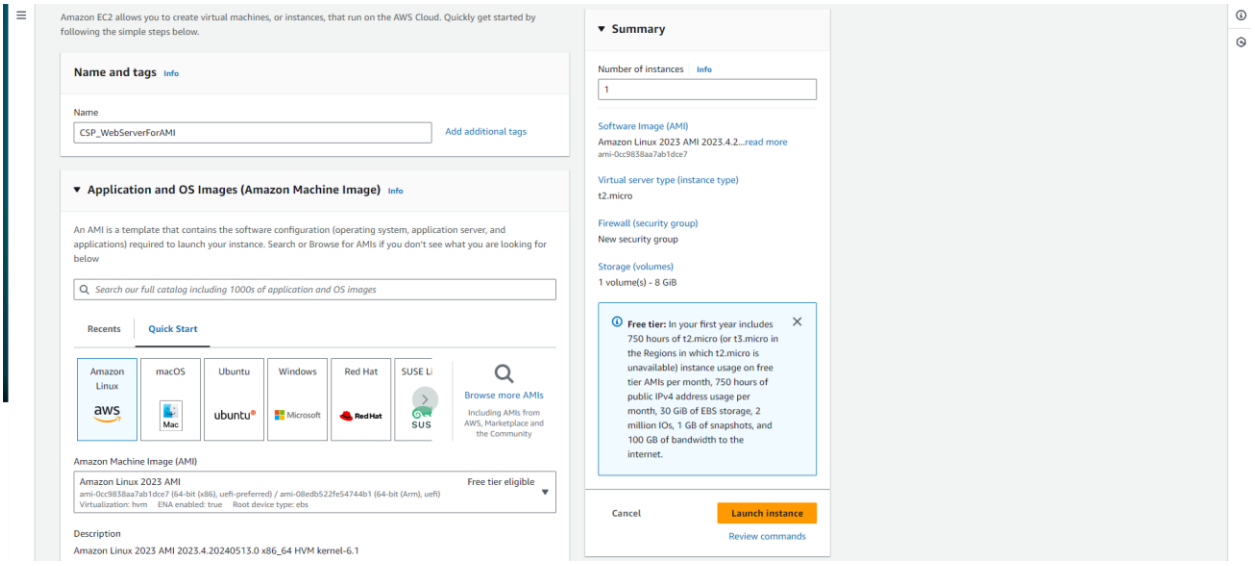
This completes our Phase 2 setup.

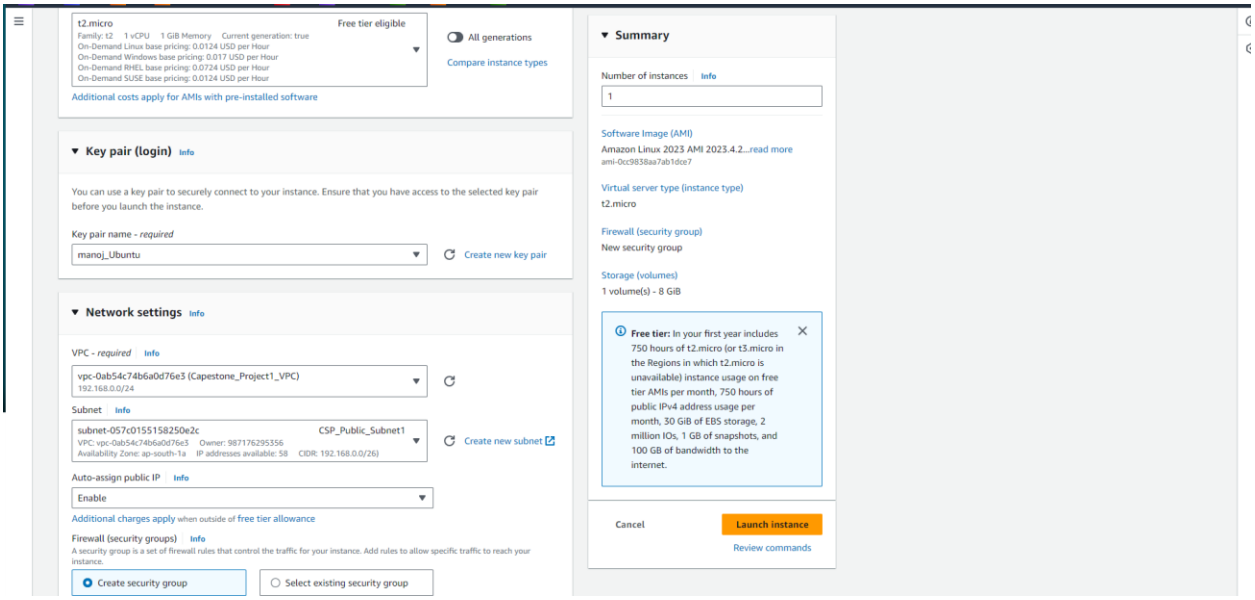
Phase 3 Setup Custom AMI for Auto Scaling

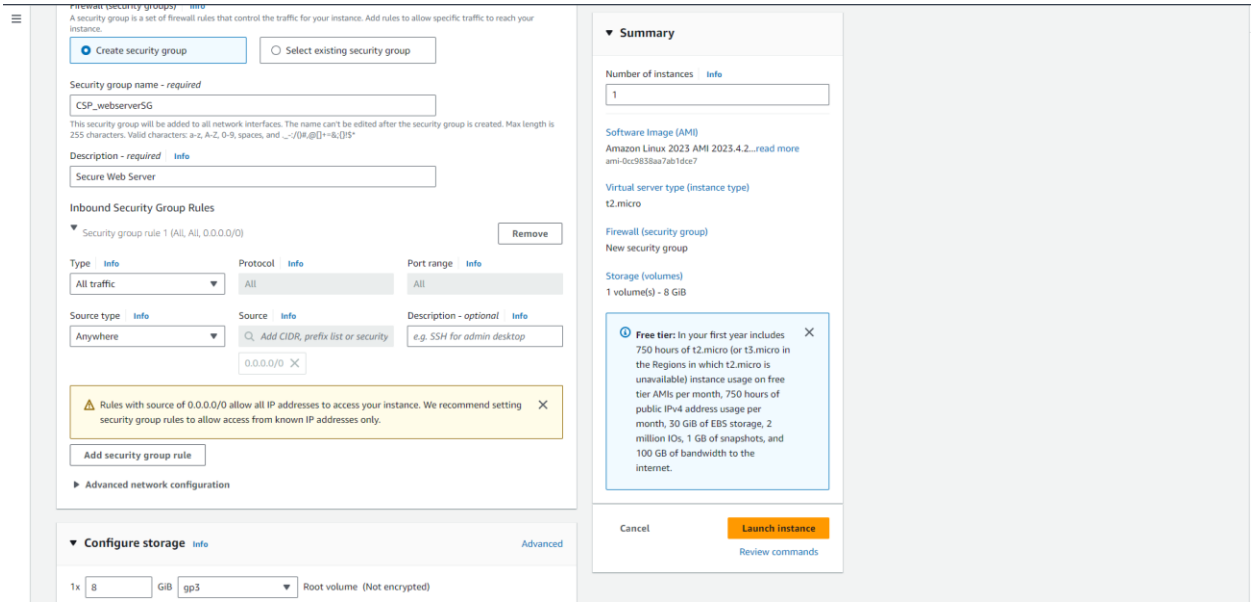
AWS Auto Scaling is a service that helps you automatically adjust the capacity of your applications running on AWS to maintain steady, predictable performance at the lowest possible cost. It involves scaling in and out (adjusting the number of instances) based on the real-time demand and predefined policies.

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.



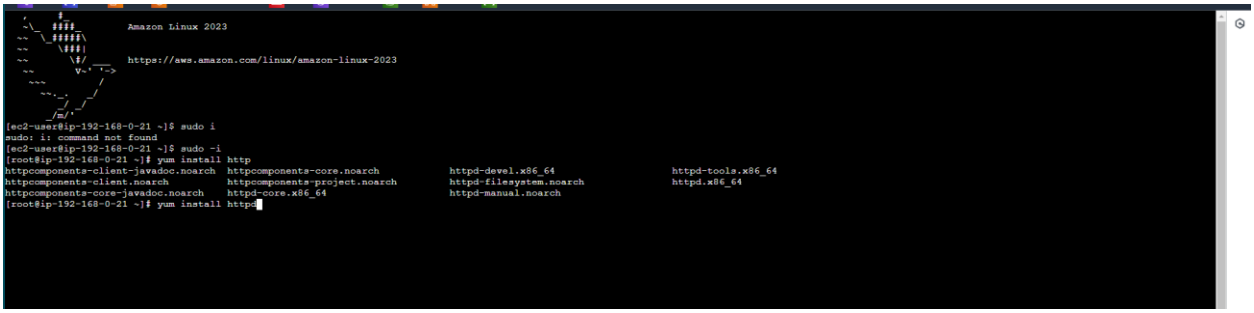




Click on Launch instance.

Log into created instance.

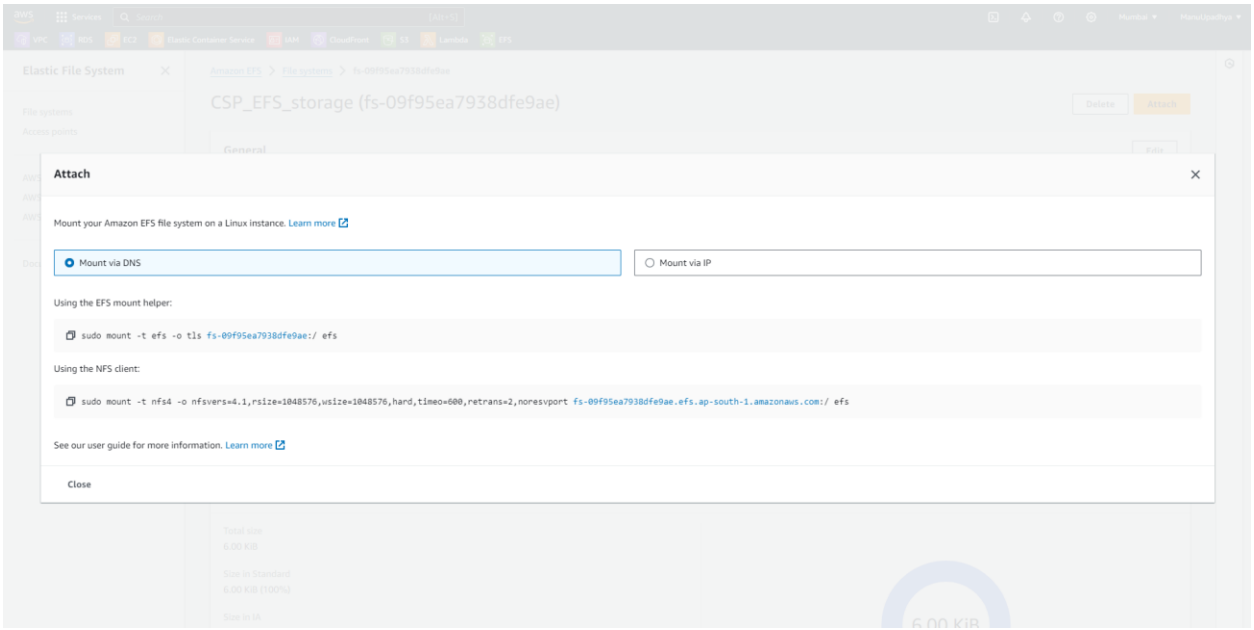
Installing Apache web server



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

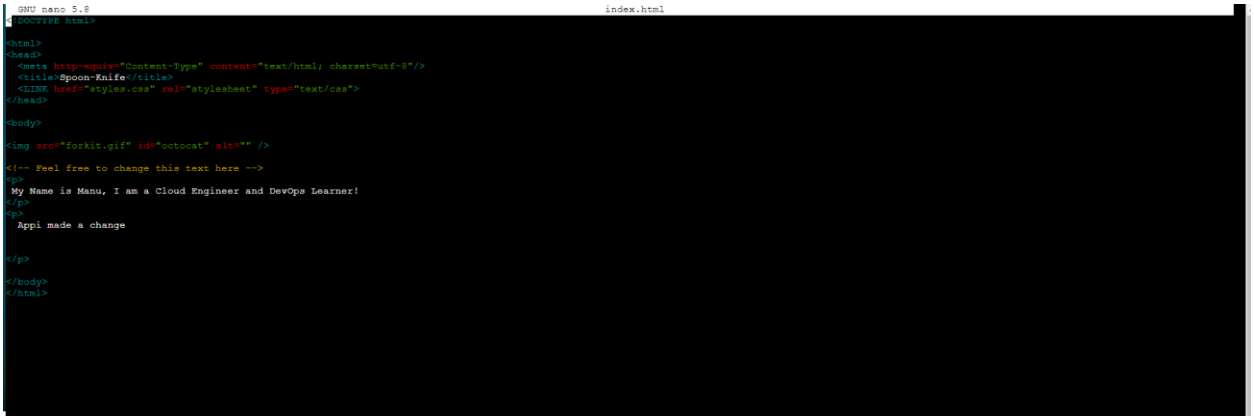
Now we will connect the created instance with created EFS file system.

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



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

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



Sample Web application created.

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

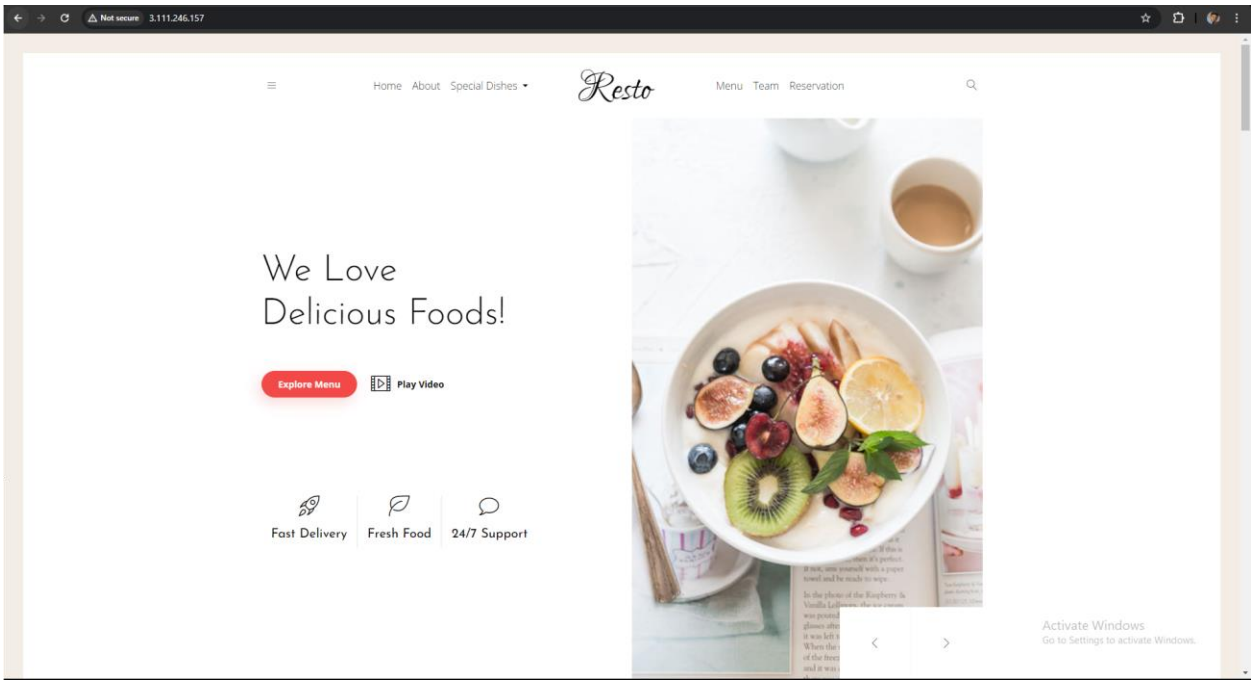
Navigate to nano /etc/fstab and add the text below into the document.

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

```
  GNU nano 5.8                                                                /etc/fstab
fs-09f95ea7938dfe9ae.efs.ap-south-1.amazonaws.com:/ /var/ww/html/ efs defaults,_netdev 0 0
#
UUID=1600e074-aa16-449f-b780-97f8a31928c1     /           xfs     defaults,noatime  1   1
UUID=5F21-CDC5        /boot/efi       vfat    defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2
```

Save and exit the file.

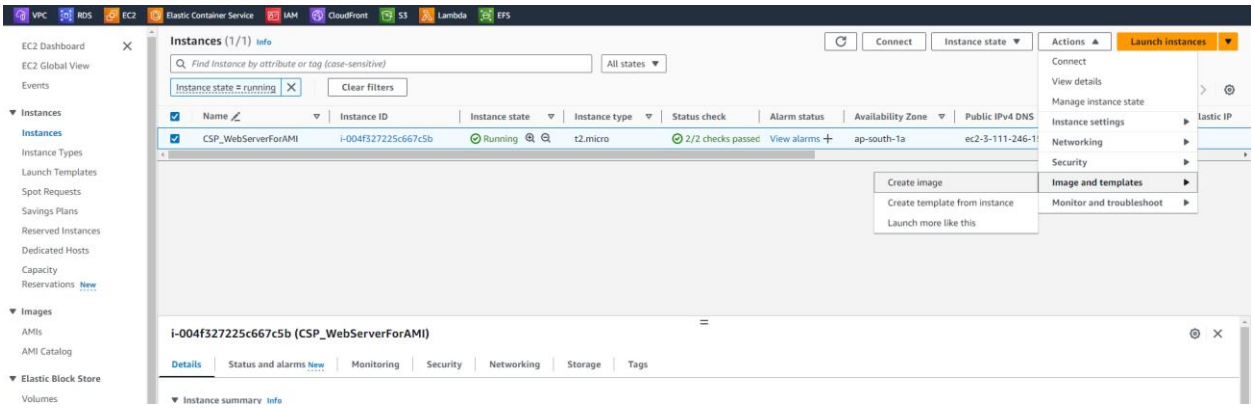
Now we need to verify our hosted web site running correctly or not.

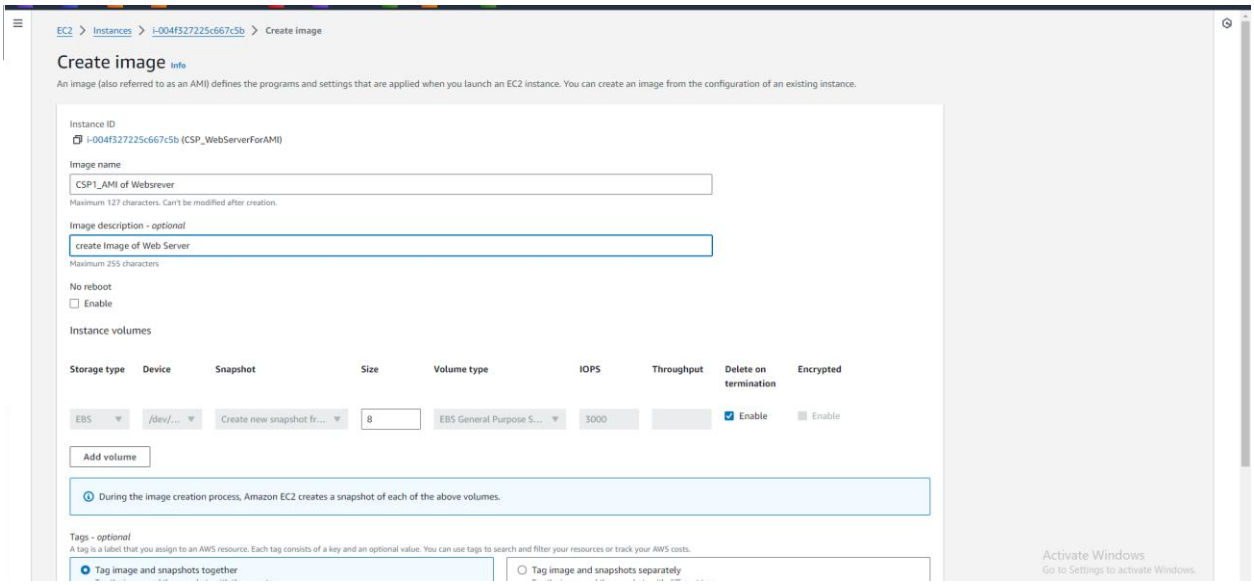


I can open my static website using ip address. If stop and start instance, then also I am getting the website. Indicating that EFS is working correctly.

Creation of AMI from this web server: -

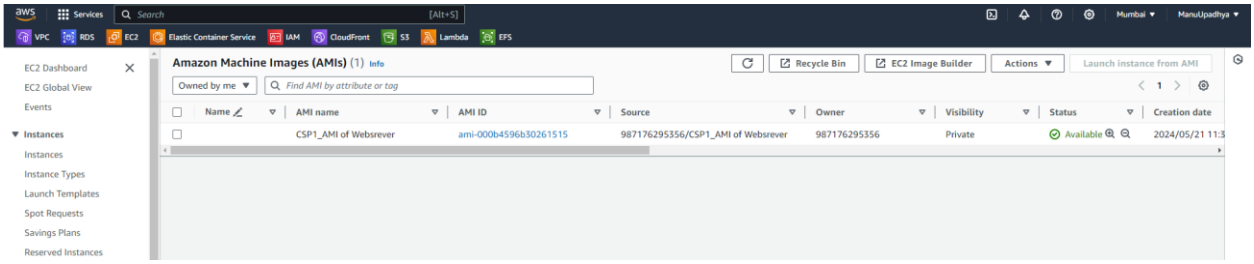
Now select the instance→actions→Image and templates →Create Image.





Click on create Image.

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



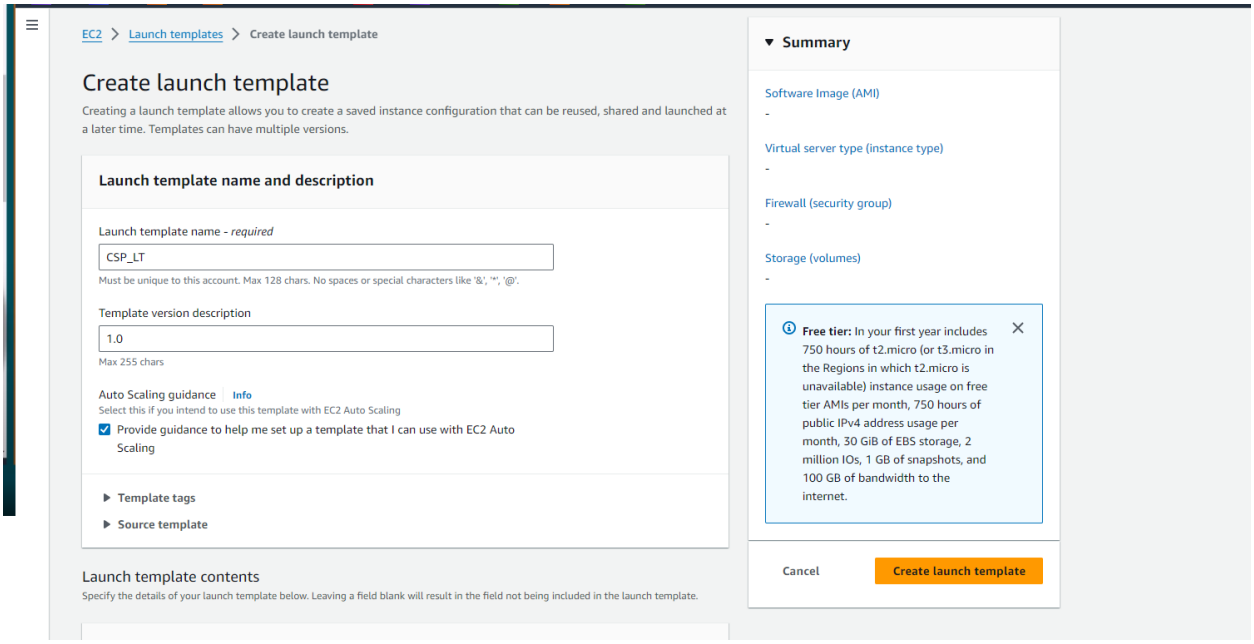
Now delete the instance since it is not required for future, we create instances using created AMI.

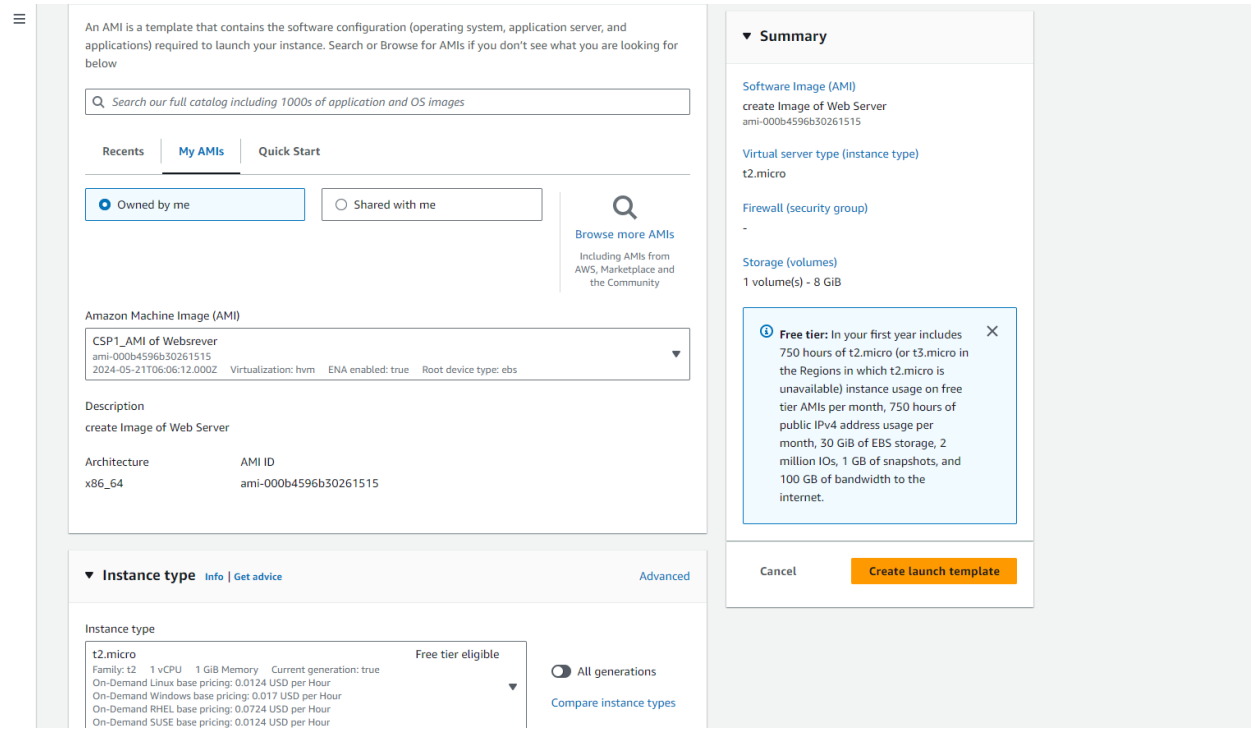
Phase 3 is completed here.

Phase 4 Setup and test Auto Scaling

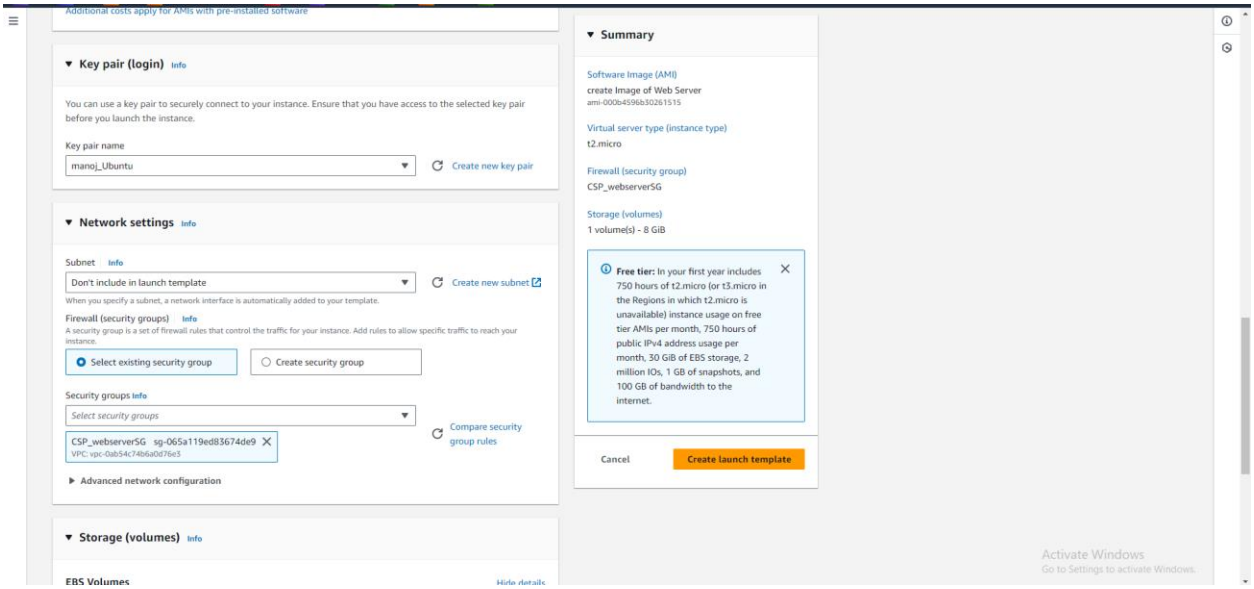
The initial step while setting up Auto Scaling is to configure the auto scaling group.

We need to create a **Launch Template** for Auto Scaling group. Click on launch template, then give name, version and select the created custom AMI.

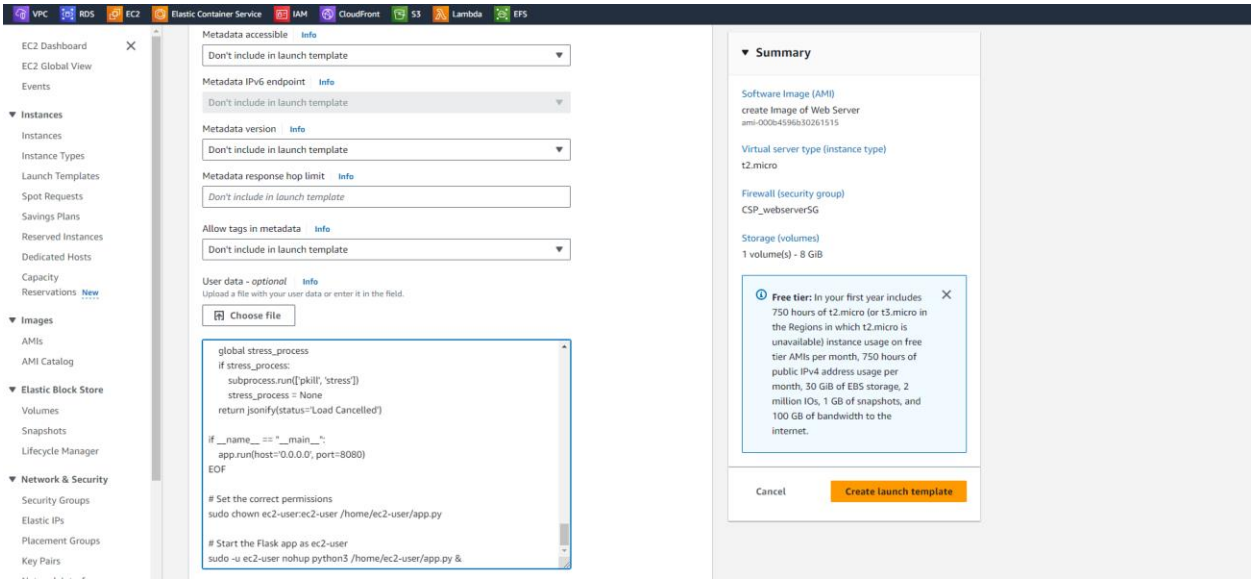




Select instance type, key-pair, do not choose subnet while creating Launch Template, choose the created security group

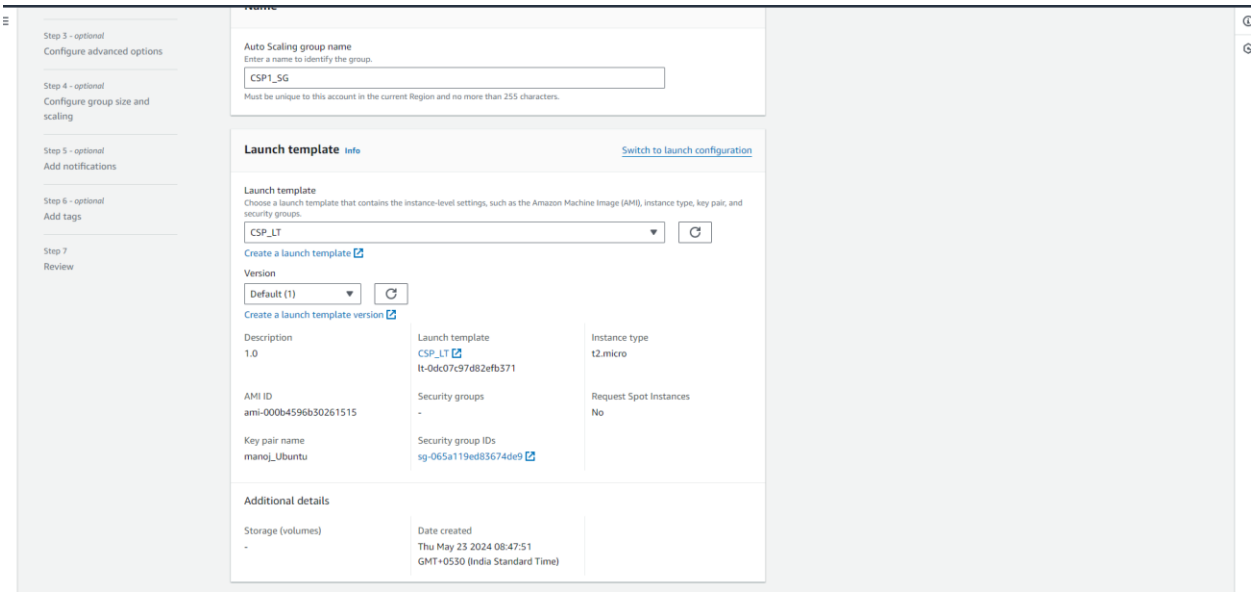


Navigate to advanced details to add user data script (it is for test application used to test auto scaling).

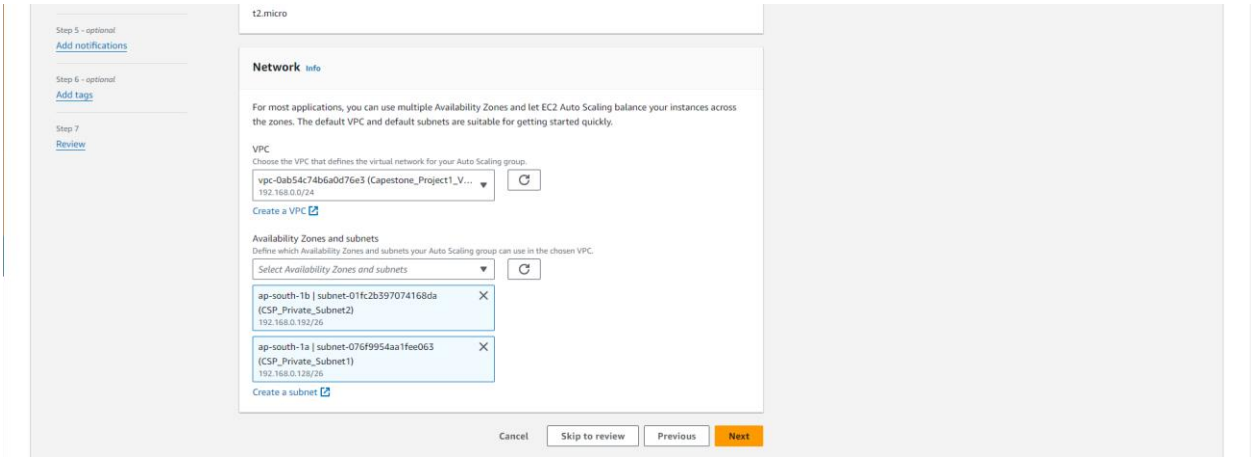


Click on Launch template.

Now navigate back to Auto Scaling page to create Auto Scaling group.

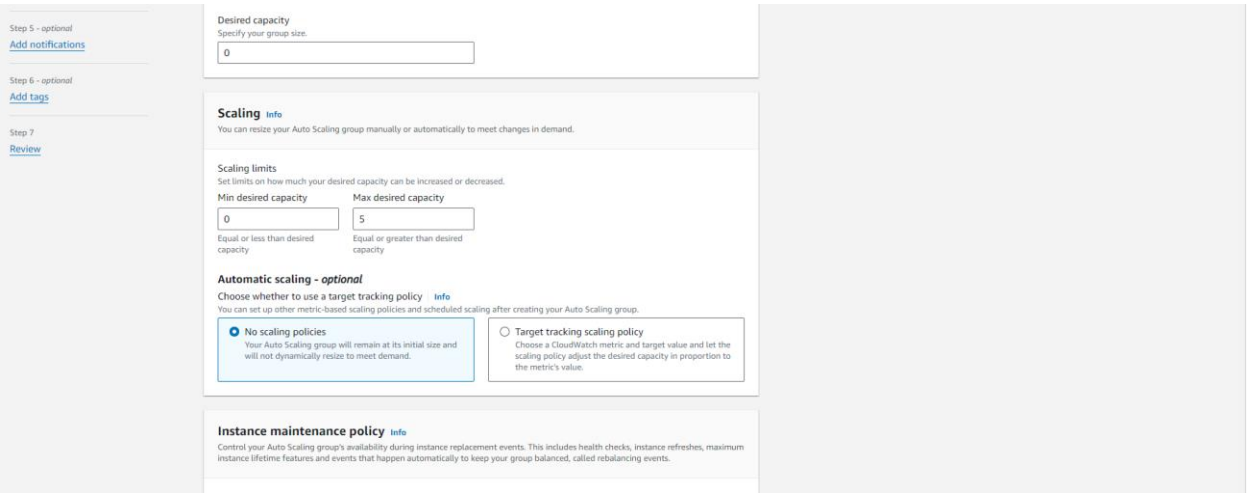


Give name and select created launch template, click on next.

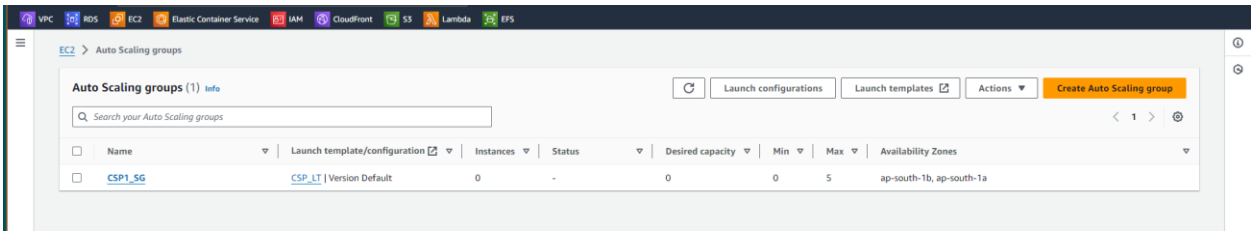


Configure network settings as required. Choose created VPC and private subnet.

Enter the Desire capacity and scaling max desired capacity.



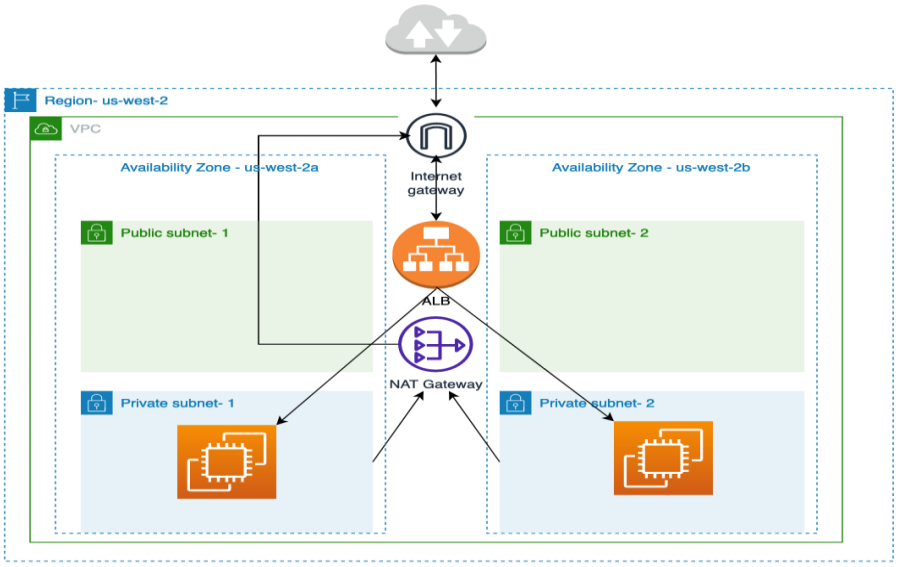
click on next→next→ next→ review and create Auto Scaling group.



Created Auto Scaling group.

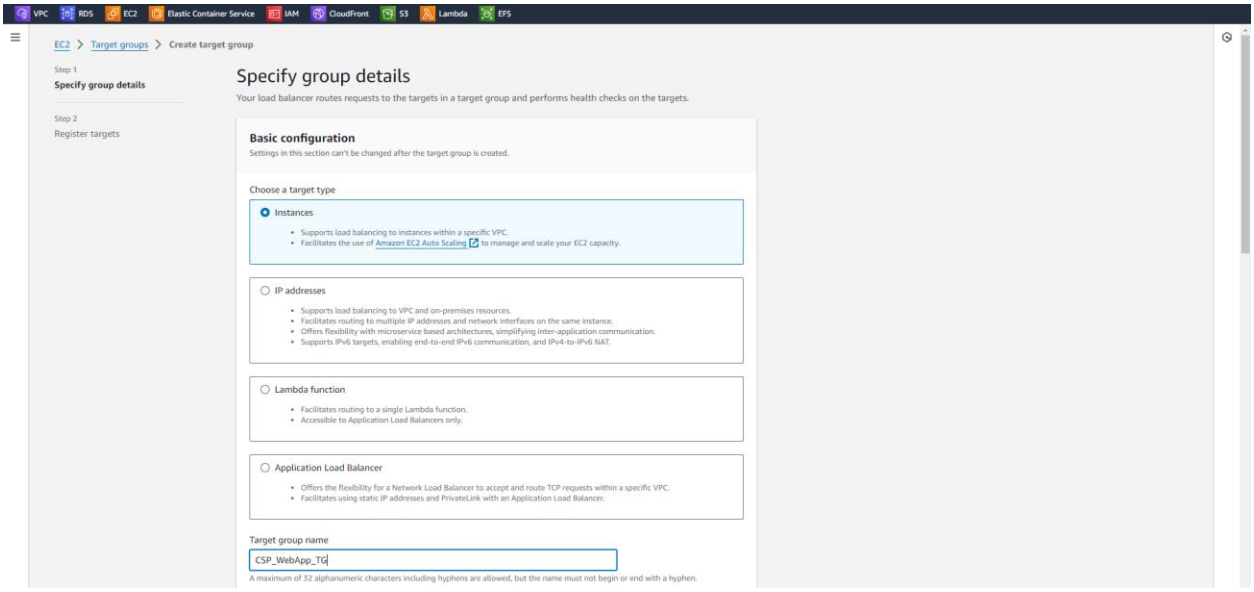
This completes the phase 4.

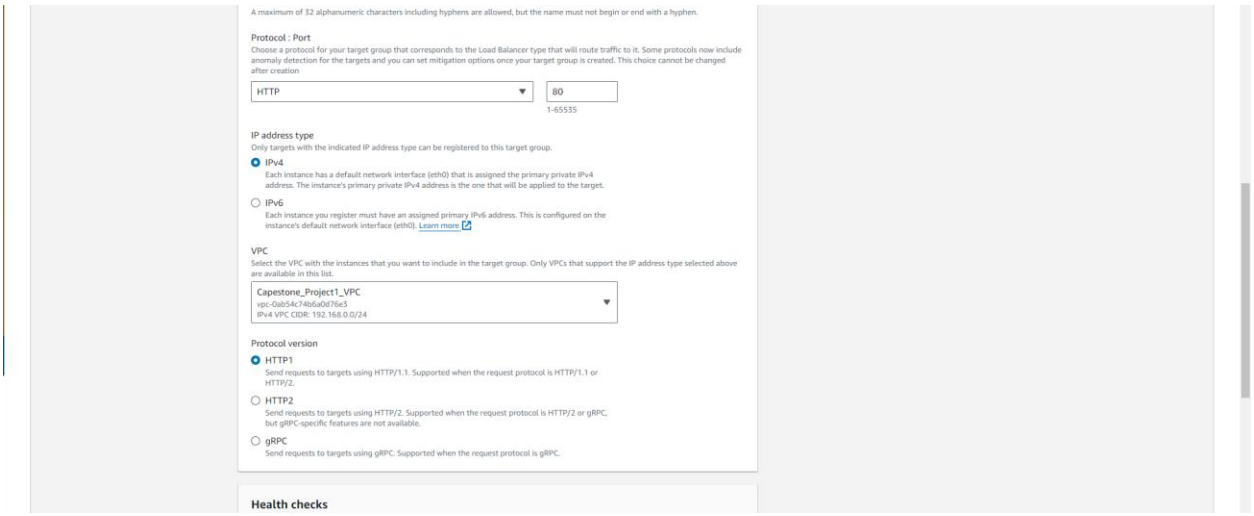
Phase 5 Setting Up Application Load balancer for the application.



Creation of target group for load balancer.

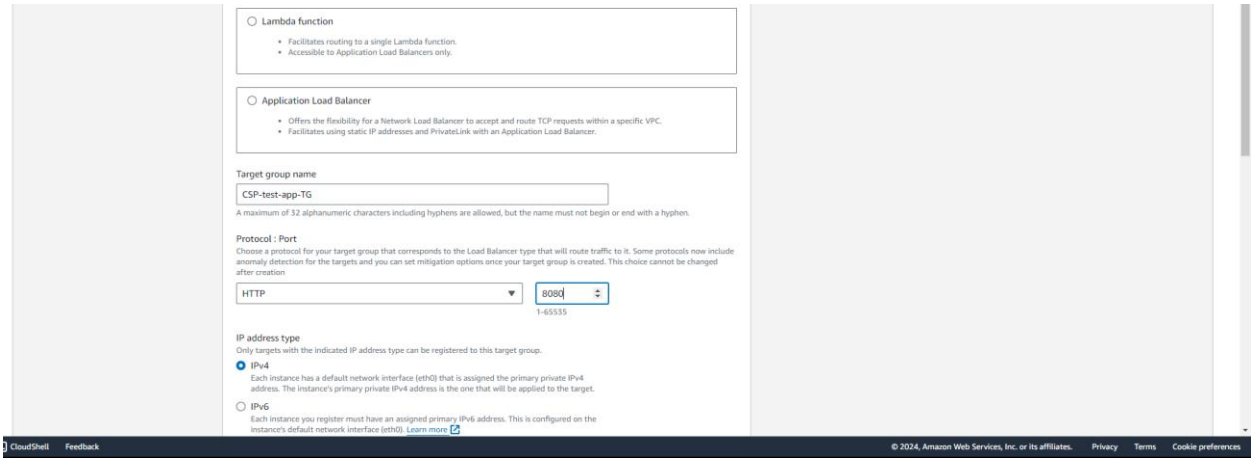
Navigate to AWS Management console→ EC2 Dashboard→target group click on create target group.



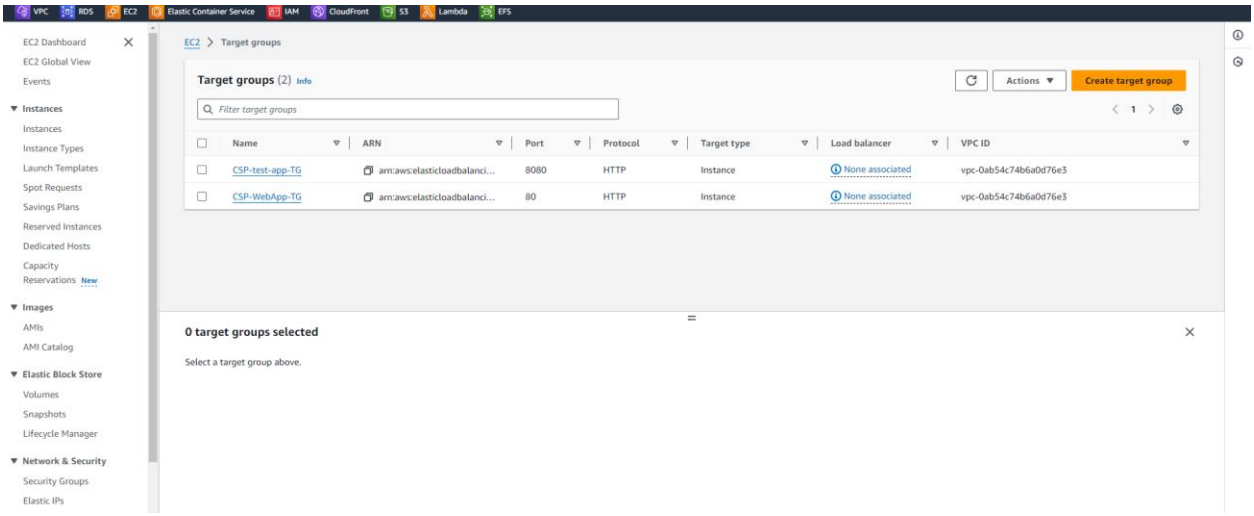


Select VPC and click on create target group. Port number 80 for our application.

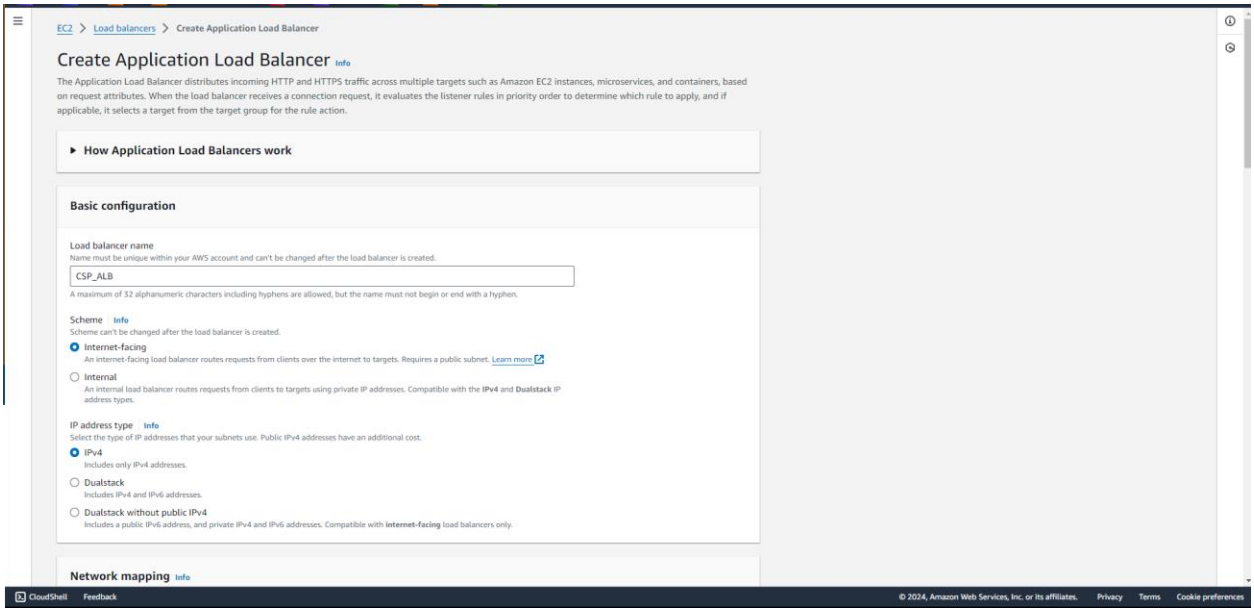
Now we need to create another target group for our test application.

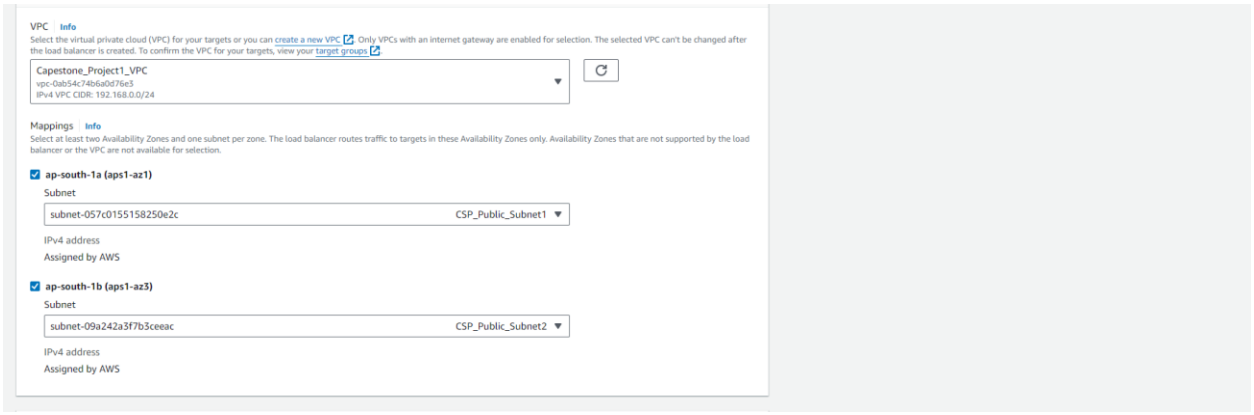


test app will be having port 8080. Remaining steps are same as we created previous TG. Created Target groups.



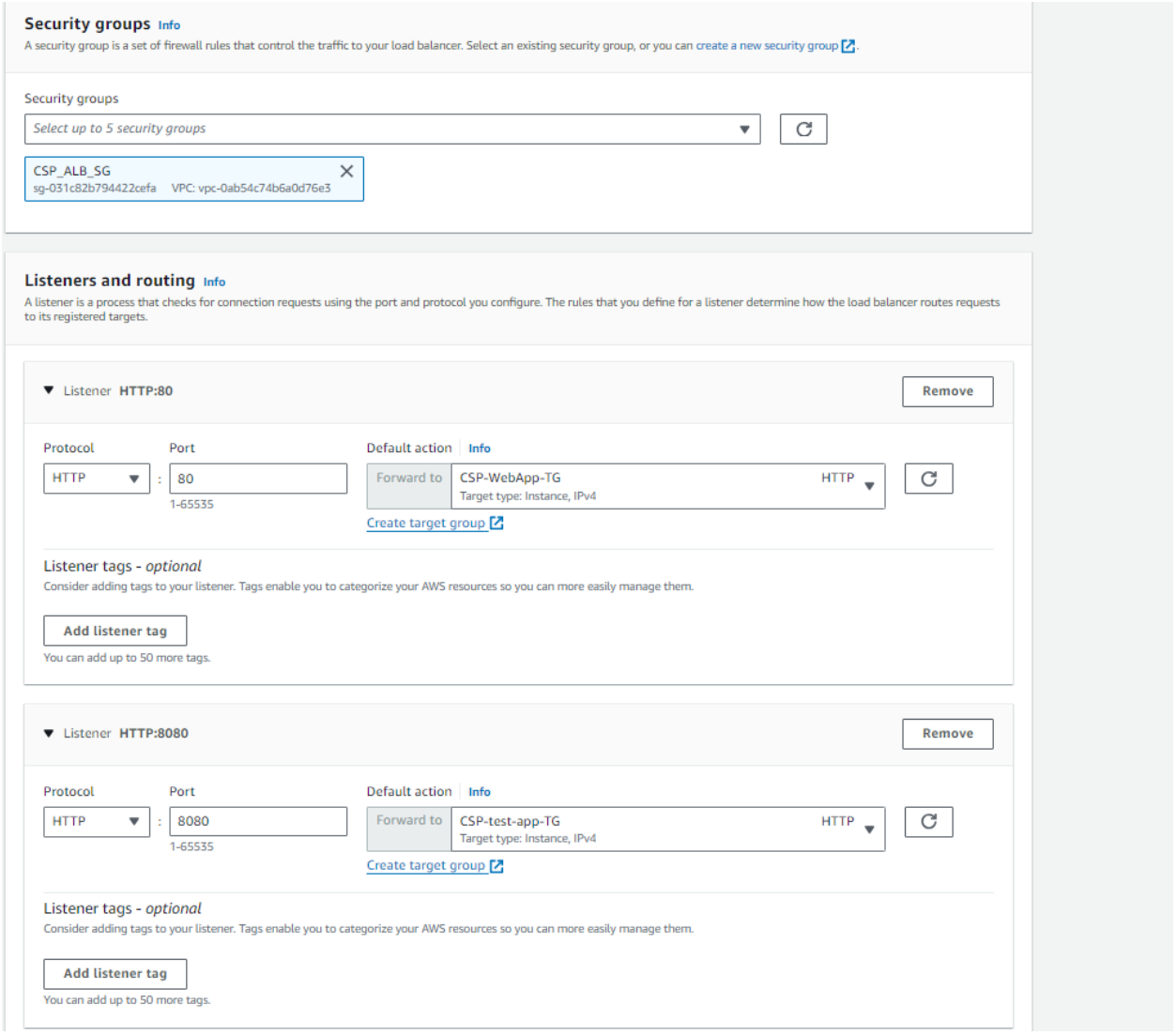
Now we will Create the **Application Load Balancer.**





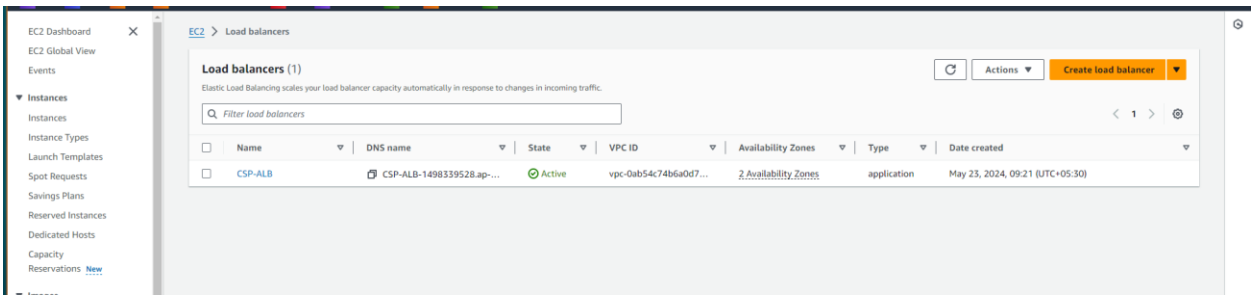
We have not created a Security Group for Load Balancer so we will create security group now.

Choose created security group and add listener (created target groups)

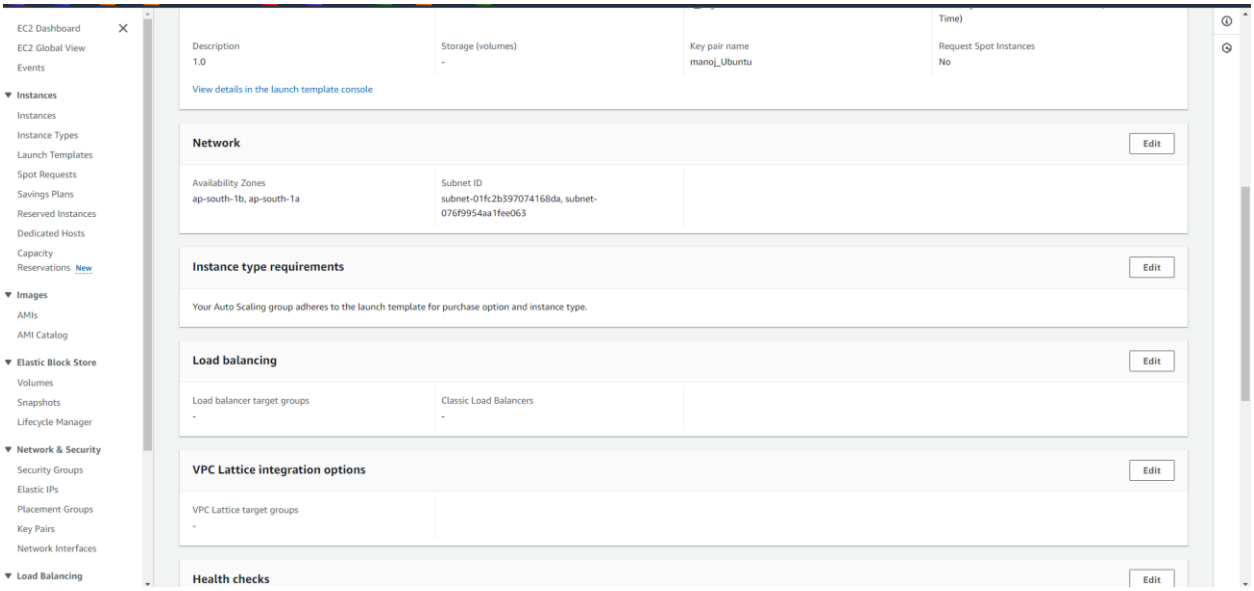


Click on create load balancer.

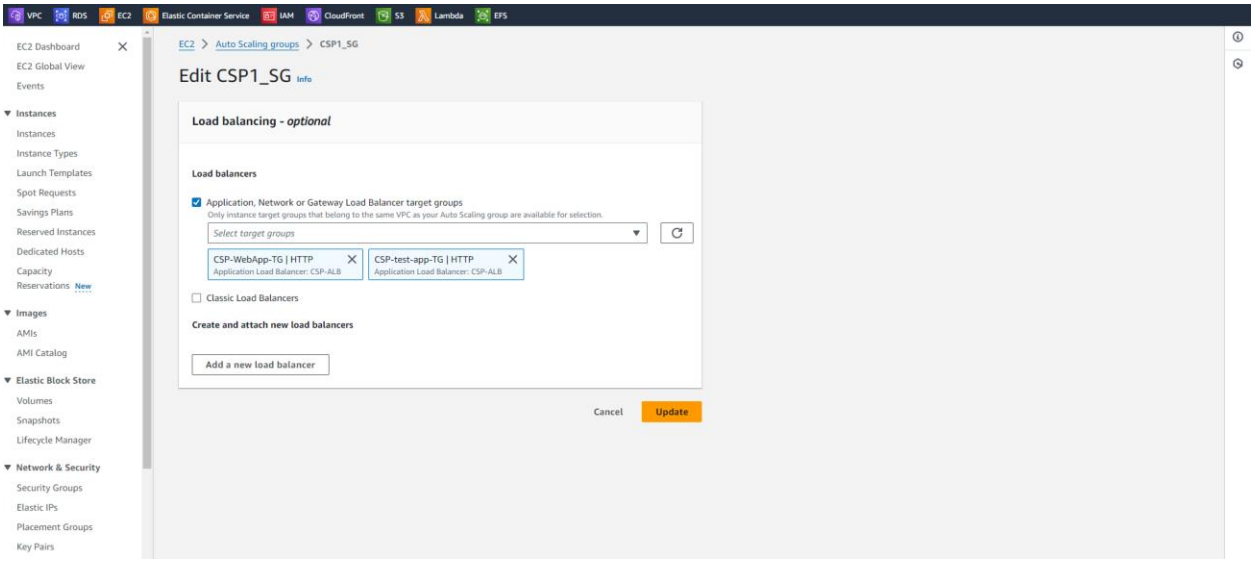
It will take few minutes to become active.



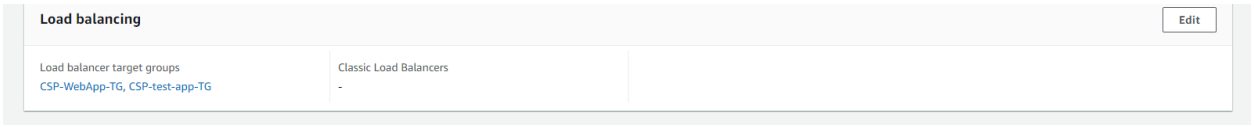
Now navigate to Auto Scaling group to link load balancer with Auto scaling group.



click on edit of load balancing.

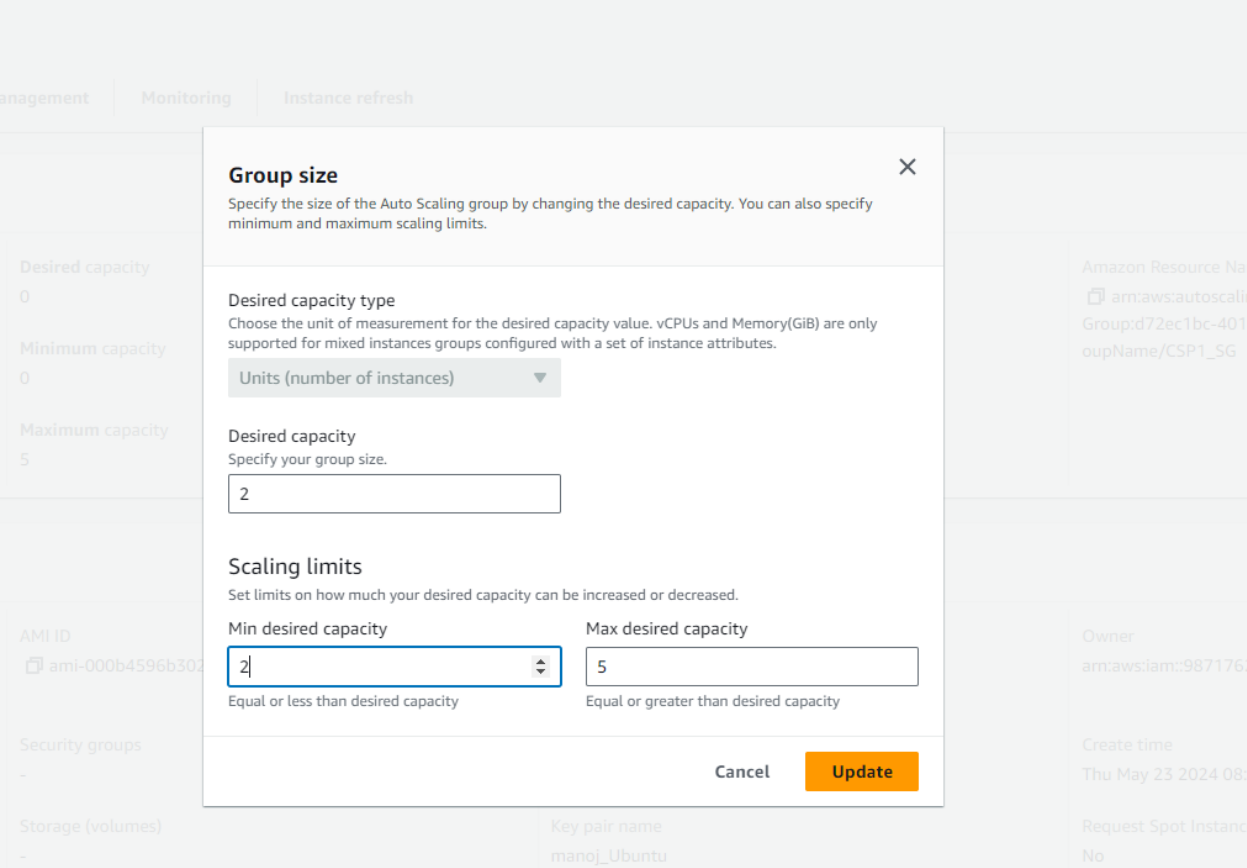


Select both target groups and click on update.

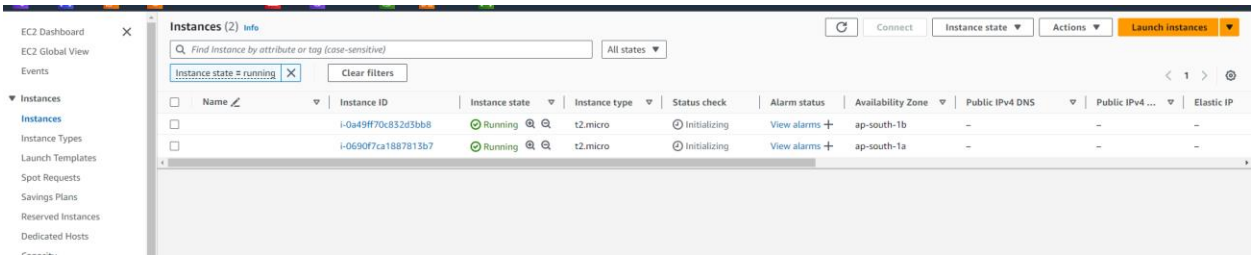


Now we will manually test.

we will manually increase the auto scaling desired capacity. Navigate to auto scaling group→Edit Group Details.

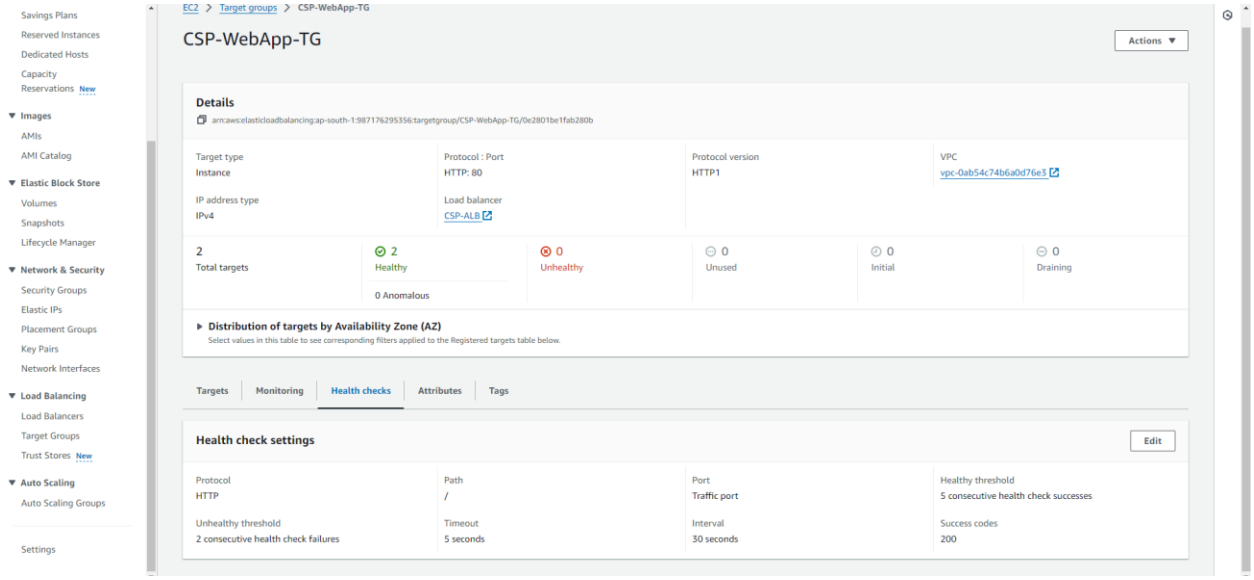


Click on Update it will create two instances. Post creation we will test the deployed application note that the Application will be deployed from AMI.



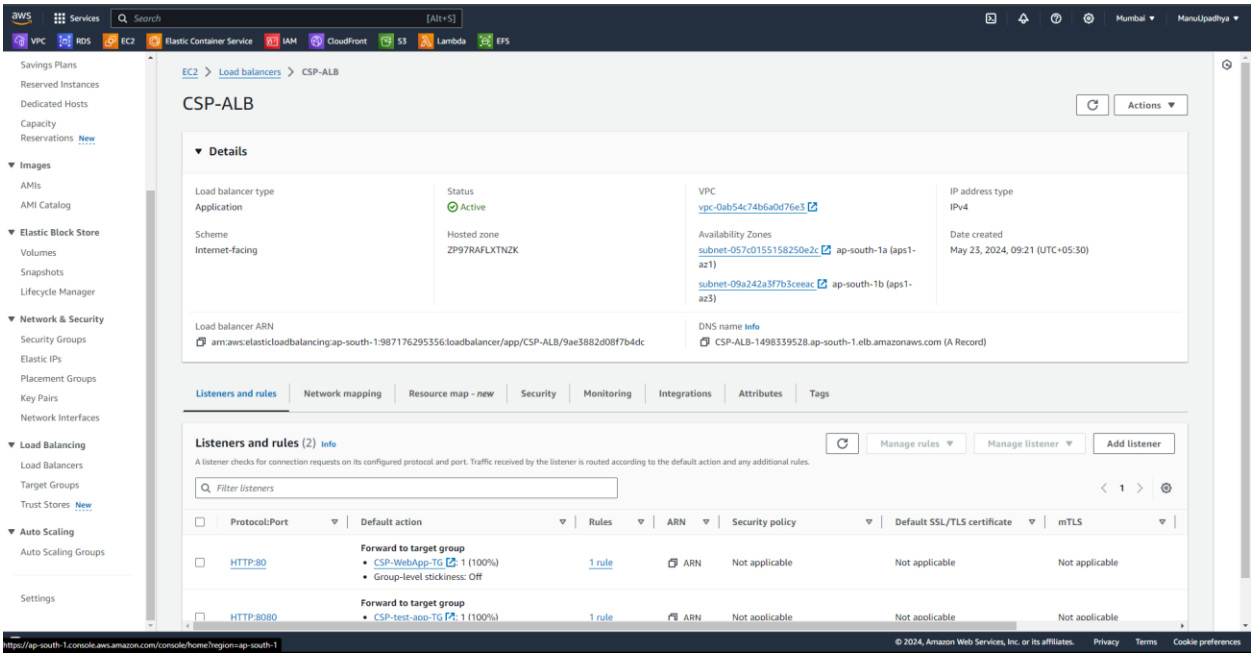
Instances are created by auto scaling group.

we will navigate to load balancing and check health of targets(launched instances/ web server)

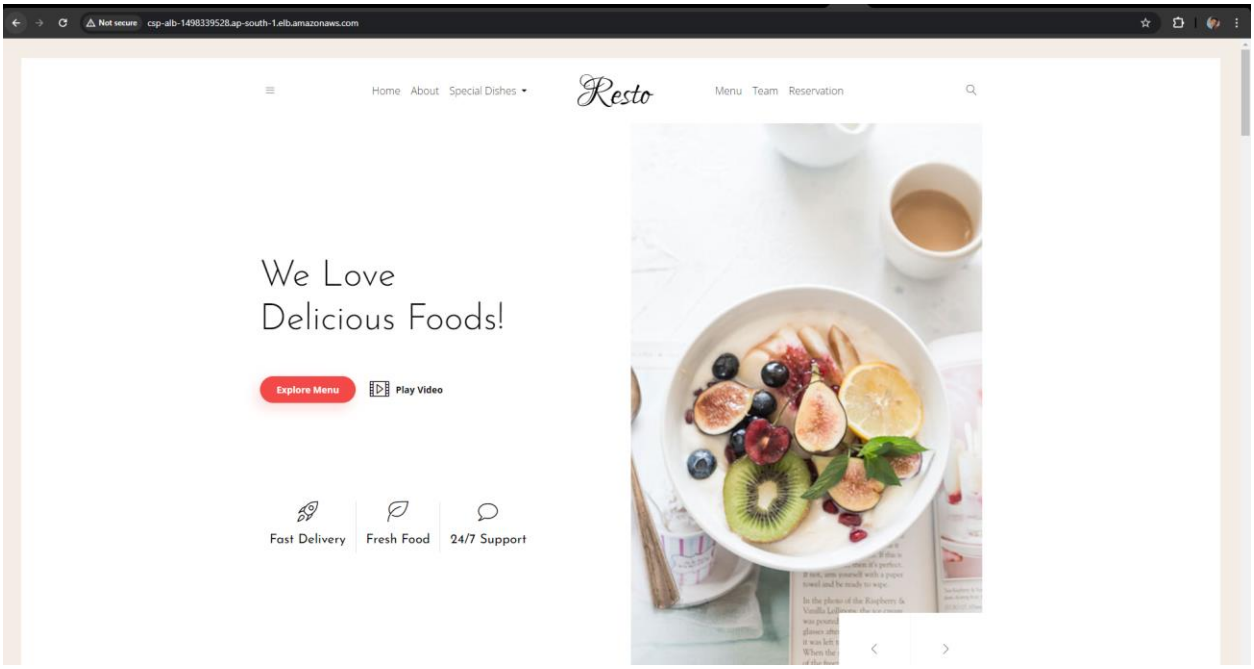


instances are healthy

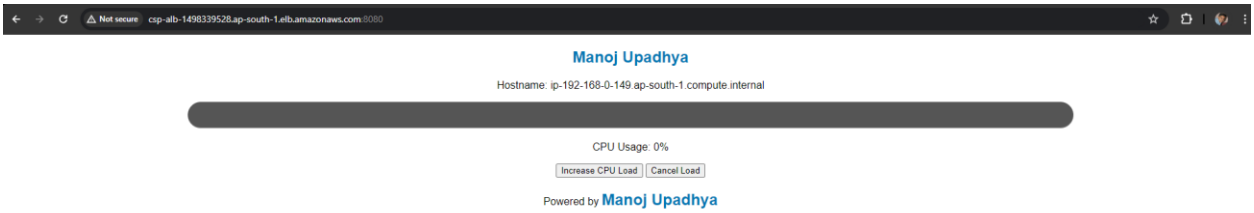
Now we will copy and paste the DNs end point of load balancer in browser to get application.



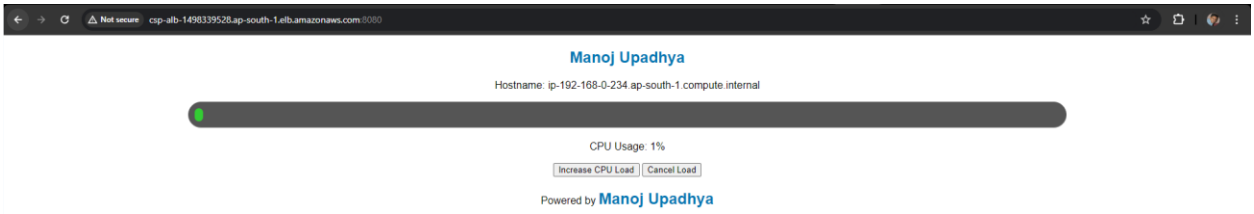
copy the DNS Name



Now how we will get to know the load balancer routing network traffic to which instance, here we use Test Application to check the host name ip of created EC2 instances using which we can understand to which EC2 instances the network traffic is routed.



If I refresh the page



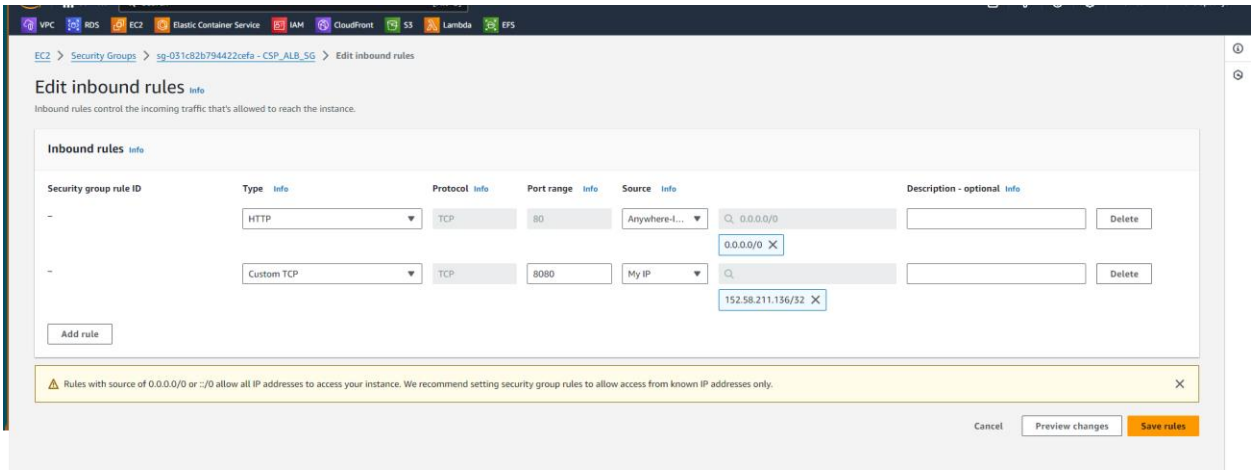
Notice the changed host name.

Phase 5 completed here.

Phase 6 Setting up all Security Group according to AWS Best Practices

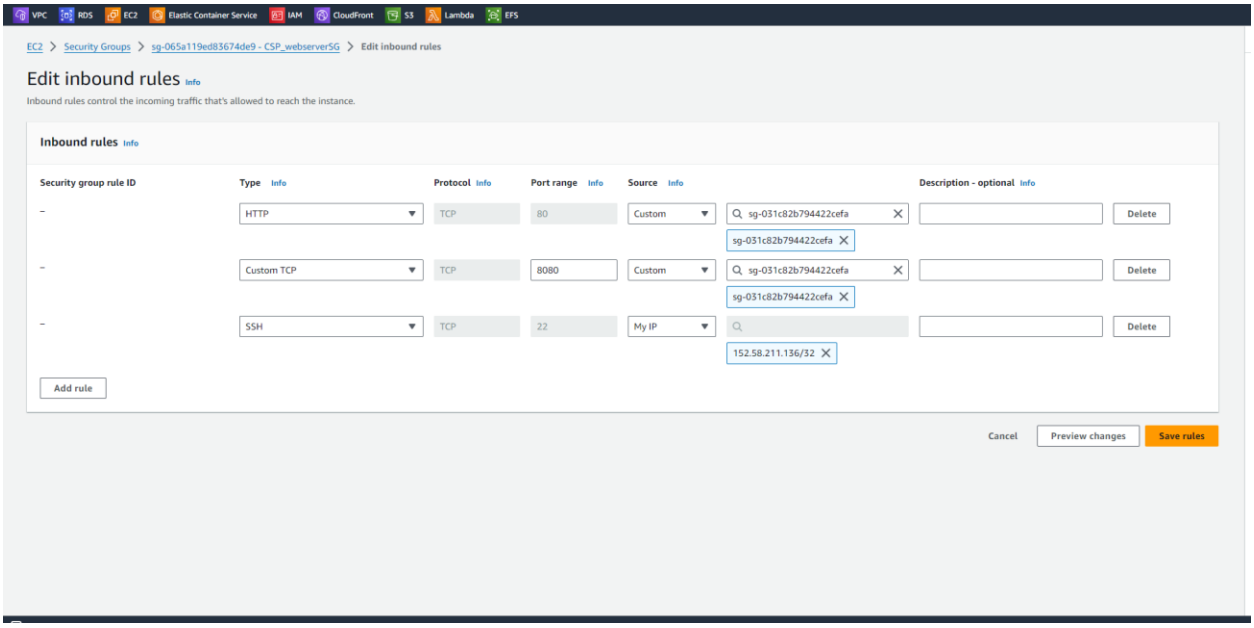
As you we have created 3 Security group (ALB SG, WEB SG, EFS, SG) all of them having all traffic allowed (inbound network traffics are allowed) Which is not a good practice, we will setup security group in this phase according to best practice.

ALB SG inbound rule : -
HTTP 80 from anywhere → web application ALB
HTTP 8080 only my ip → Test app Alb

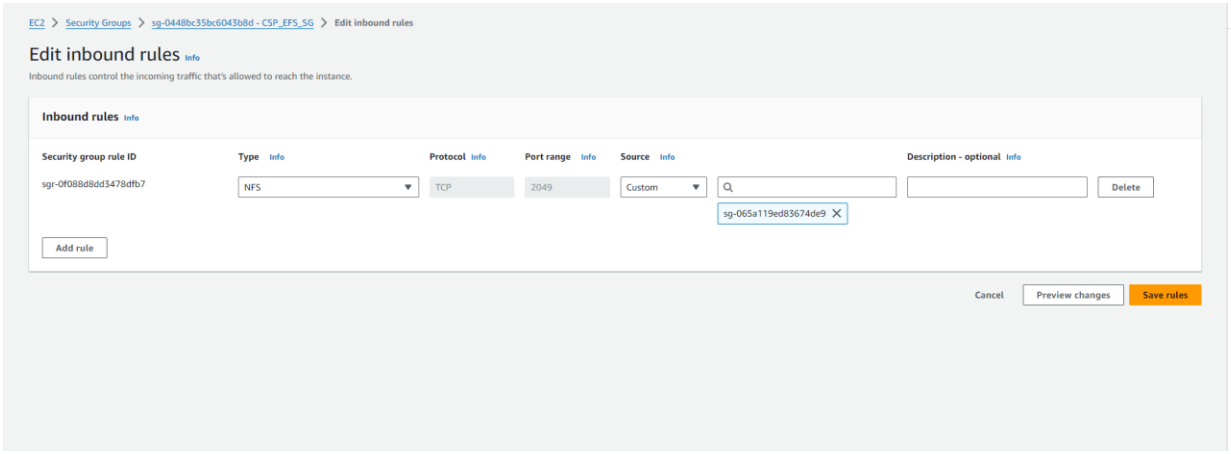


click on Save.

Web SG inbound rule: -
Here we will make sure that network traffic to Web Server only comes from Application Load Balancer, there in Source section we have selected ALB SG. Added ssh rule for debugging purposes.



EFS SG inbound rule :-
we allow NFS traffic from Web server only.



No we have configured all the security group in AWS best Practices.

Now test the application if its working fine or not, if any error made application will not behave properly.
How to test?
ALB DNS name for both web application and test application.

Check the target group health.

Check accessing test app using some other device it should not open in the other device.

This completed Phase 6.

Phase 7 Integration of Route 53 for Domain Name Management.

Prerequisite for this phase is you should have your own domain name.

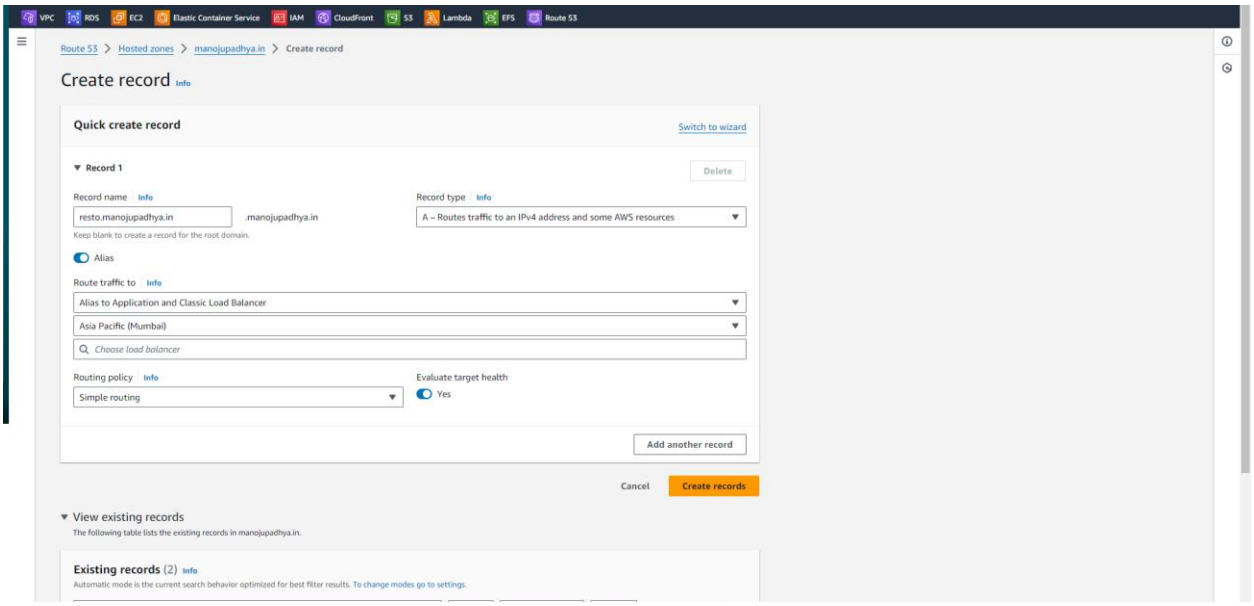
Navigate to → Route 53 in AWS Management Console.

Register a domain name, if you have purchased domain name from any other website add it to Route 53.

Create a Hosted Zone. After creating the Hosted Zone, we need to add Nameserver record present in hosted zone to our Domain name.

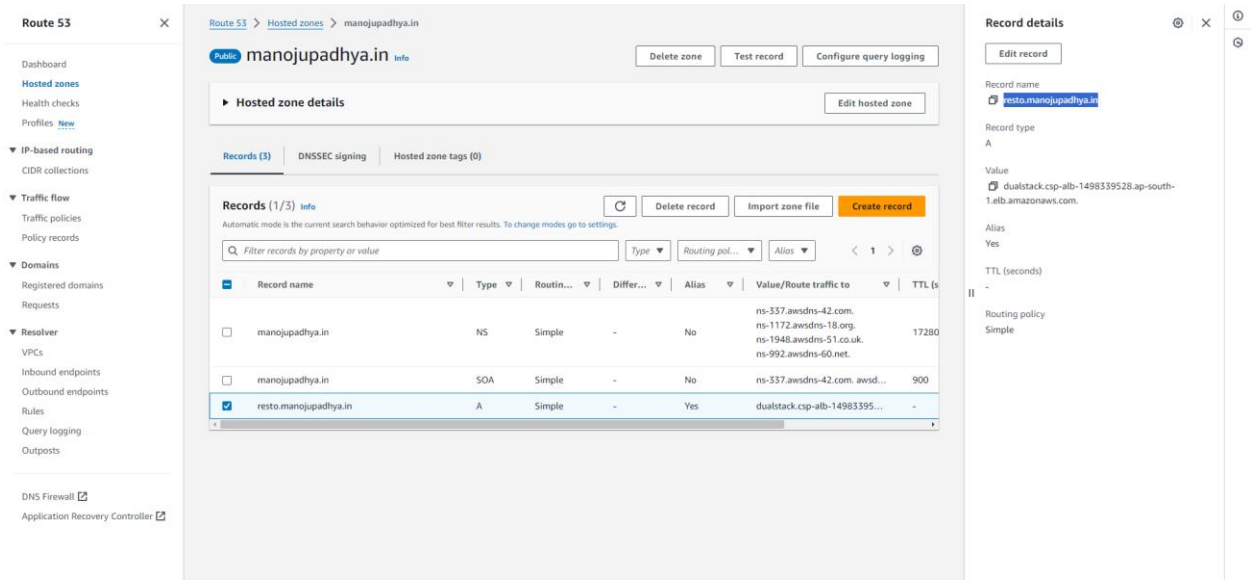
Now we need to create a Record for our web server purposes.

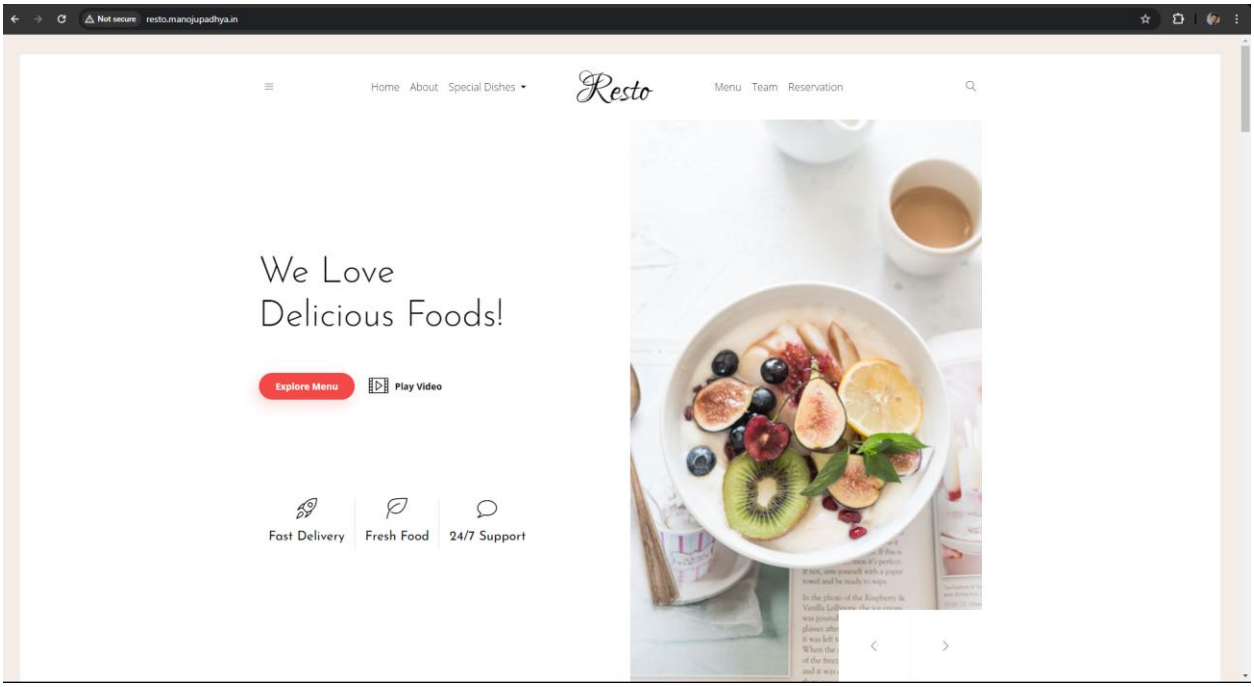
Click on create record.

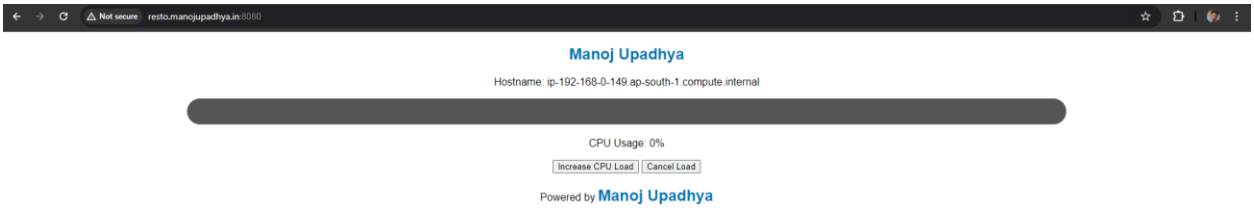


In the alias section choose alias as the Load balancer so that Domain name forwards the traffic to load balancer.
It will take some time to get the record created.

Now we can access the website using DNS name directly.


Route 53
Dashboard
Hosted zones
Health checks
Profiles New
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Outposts
DNS Firewall
Application Recovery Controller
Route 53 > Hosted zones > manojupadhya.in
Public manojupadhya.in Info
Delete zone
Test record
Configure query logging
Hosted zone details
Edit hosted zone
Records (3)
DNSSEC signing
Hosted zone tags (0)
Records (1/3) Info
Delete record
Import zone file
Create record
Filter records by property or value
Record name
Type
Routin...
Differ...
Alias
Value/Route traffic to
TTL (s
manojupadhya.in
NS
Simple
No
ns-337.awsdns-42.com.
ns-1172.awsdns-18.org.
ns-1948.awsdns-51.co.uk.
ns-992.awsdns-60.net.
17280
manojupadhya.in
SOA
Simple
No
ns-337.awsdns-42.com. awsd...
900
resto.manojupadhya.in
A
Simple
Yes
dualstack.csp-alb-14983395...
Record details
Edit record
Record name
resto.manojupadhya.in
Record type
A
Value
dualstack.csp-alb-1498339528.ap-south-1.elb.amazonaws.com.
Alias
Yes
TTL (seconds)
Routing policy
Simple


Not secure
resto.manojupadhya.in
Home
About
Special Dishes
Resto
Menu
Team
Reservation
We Love
Delicious Foods!
Explore Menu
Play Video
Fast Delivery
Fresh Food
24/7 Support



We can see the test and web application using domain name.

Now we can say the Project is 100% completed. All the phases are completed.

Now we are left with Testing and Optimization Phase.

Project Functionality Testing and load Testing.

Here we use a test application for testing purposes.

- High Availability.
  Navigate to EC2 Dashboard and terminate an instance. Try to access website, mean time autoscaling group should create one more instance and check these changes IP address/host name in test application as well.
- Setting up Dynamic Scaling Policy (If CPU Utilization > 60 Launch a new instance) and test using test application by increasing load.
-