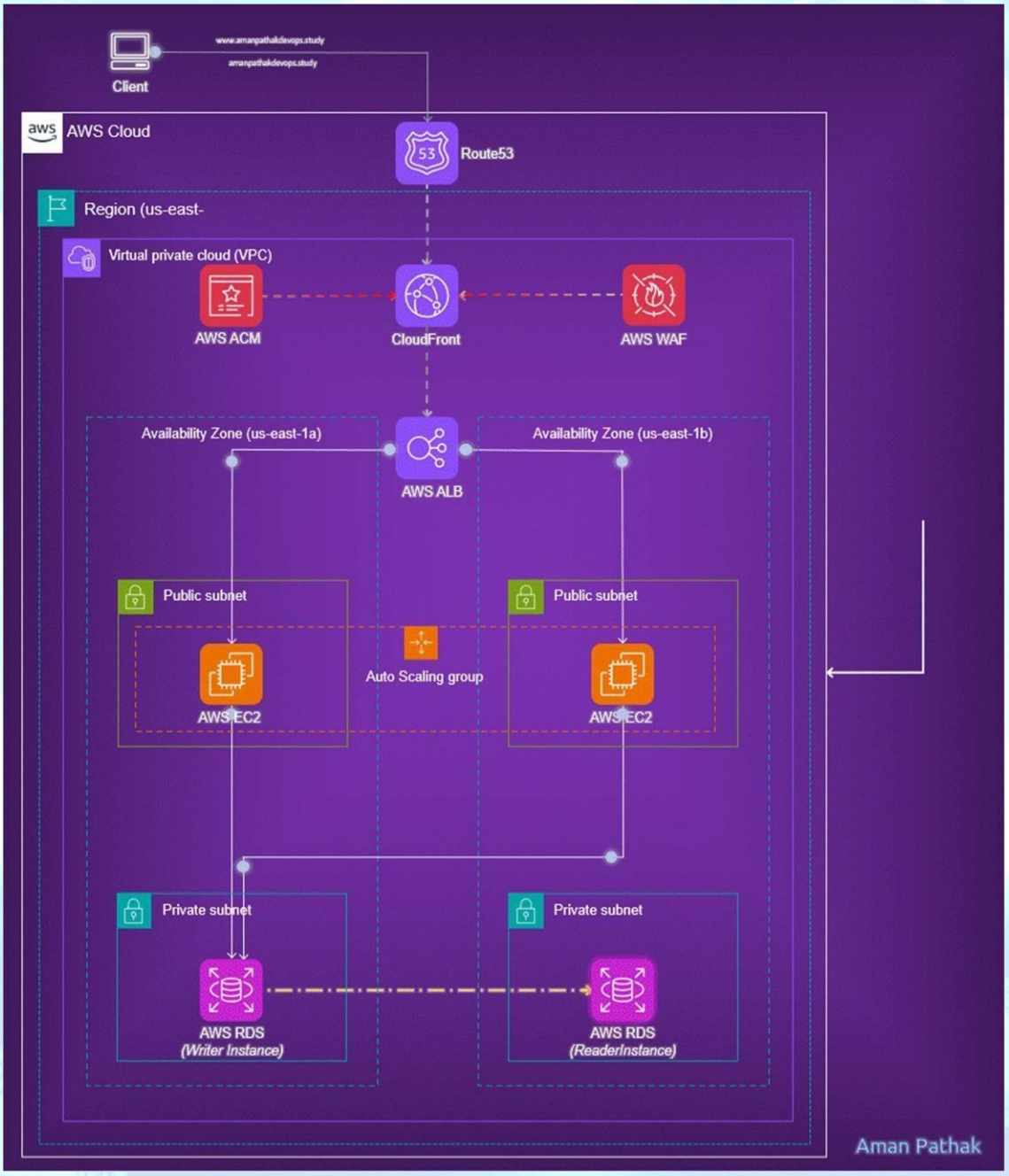
NAME: S. khalindar , **BATCH NO:** 140

Implementation of 2-Tier Architecture in AWS Using Route 53, EC2, Load Balancer, Auto Scaling, CloudFront, and WAF



AWS Architecture Overview:

This architecture represents a **highly available, scalable, and secure web application deployment** on AWS. It leverages multiple AWS services across two Availability Zones to ensure fault tolerance, performance optimization, and robust security.

Key Strengths of this Architecture:

- **High Availability**: Two Availability Zones (us-east-1a and us-east-1b) ensure redundancy. If one zone goes down, the other keeps your app running.
- **Scalability**: Auto Scaling Groups dynamically adjust the number of EC2 instances based on traffic, keeping performance smooth and cost optimized.
- **Security Layers**:
  - **AWS WAF** protects against common web exploits.
  - **AWS ACM** handles SSL/TLS certificates for secure communication.
  - **Private subnets** isolate sensitive components like databases from public access.
- **Global Performance**: CloudFront CDN caches content closer to users, reducing latency.
- **Traffic Management**: Route 53 handles DNS routing, while the ALB smartly distributes incoming traffic across healthy EC2 instances.

Uses cases for business perspective:

- Hosting highly available and scalable web applications or websites.
- Deploying multi-tier applications where separation of presentation, application, and data layers is crucial for security and manageability.
- Applications requiring low-latency content delivery and protection against web exploits.
- Businesses needing a flexible and cost-effective infrastructure that scales with demand.

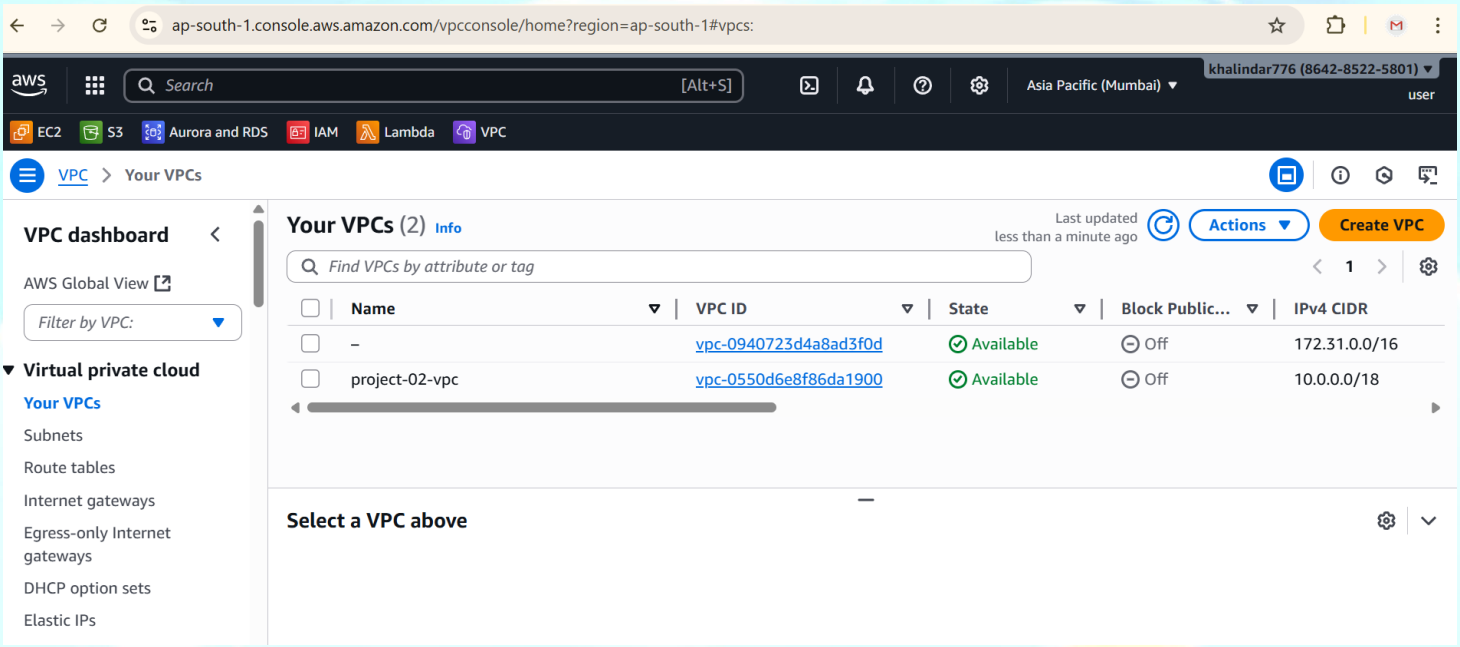
Steps to follow:

- VPC (Subnets, Route Tables, and IGW)
- EC2 (Security Group, AMI, Template, Target Groups, Load Balancer, and Auto Scaling Group)
- Route 53 (Hosted Zone, and health checks)
- Go Daddy (DNS)
- ACM (Public TLS/SSL Certificate)
- Cloud Front
- WAF (For security)

PROCESS :

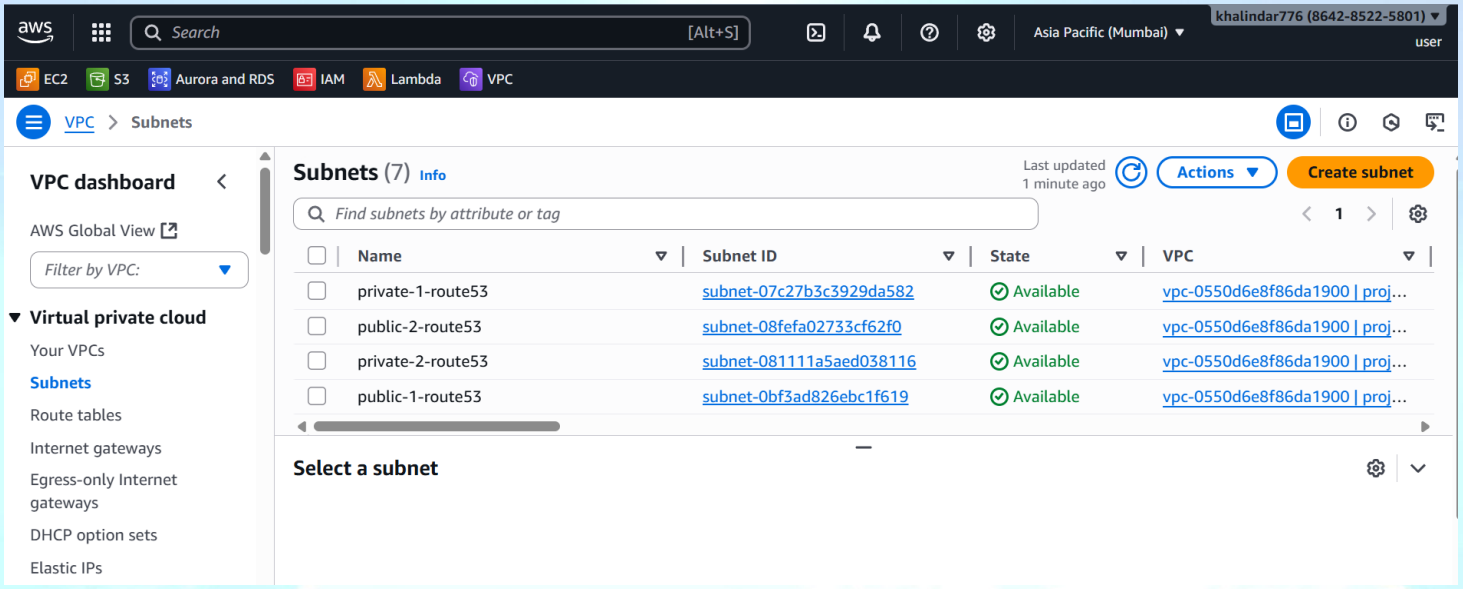
Step 1: Create VPC (Virtual Private Cloud).

- Cerate VPC
- VPC name is **project-02-vpc**
- IPV4 address 120.0.0.0/16
- VPC created



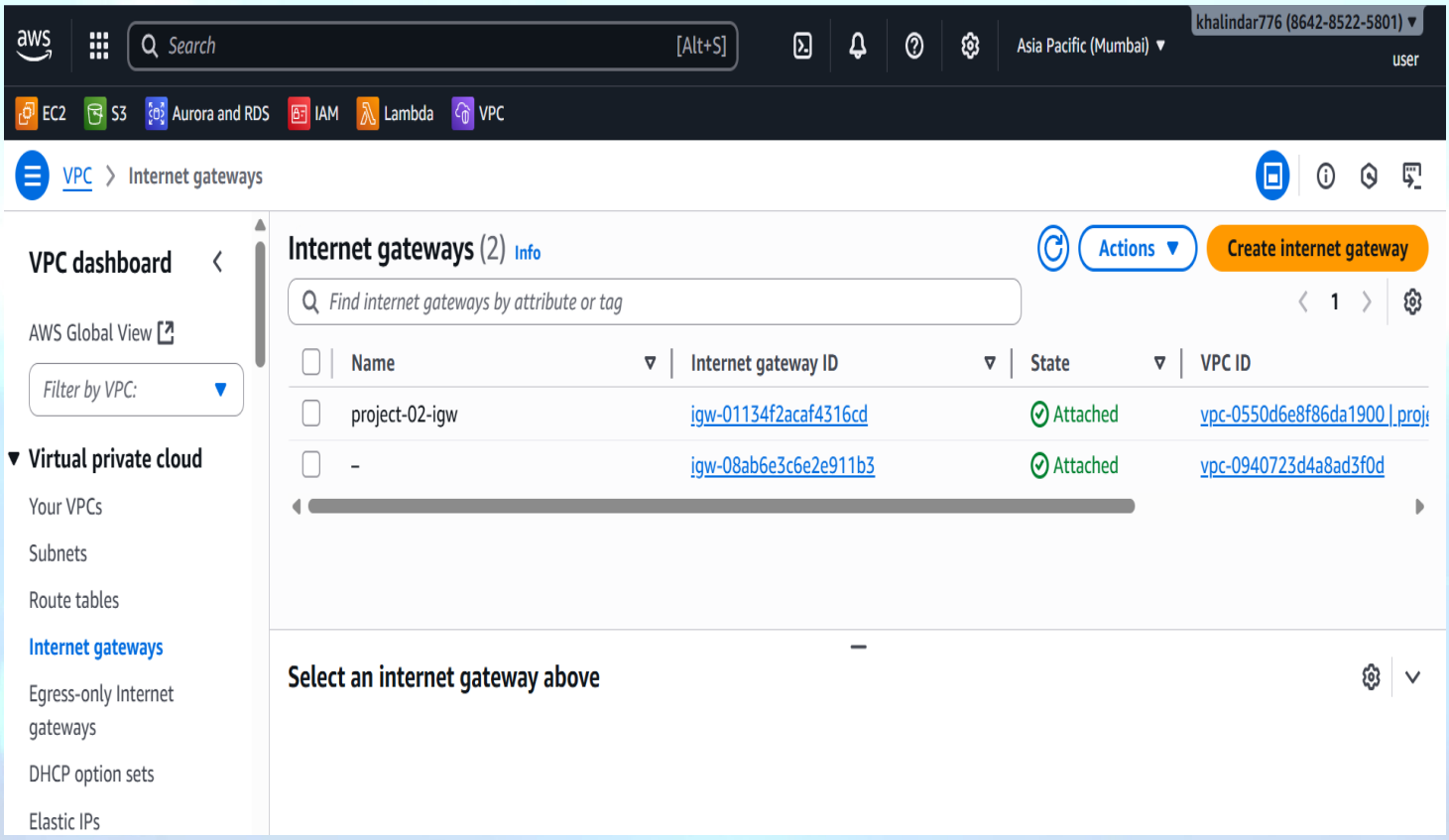
Step 2: Create Subnets (4).

- Create 2 public subnets & 2 private subnets
- Given names for 4 Subnets (**pub-1-route53, pub-2-route53, private-1-route53, and private2route53**).
- Select the given VPC id : project-vpc
- Select Availability zones for all subnets
- Customize the IP address with the proper ranges for all subnets Create the Subnet



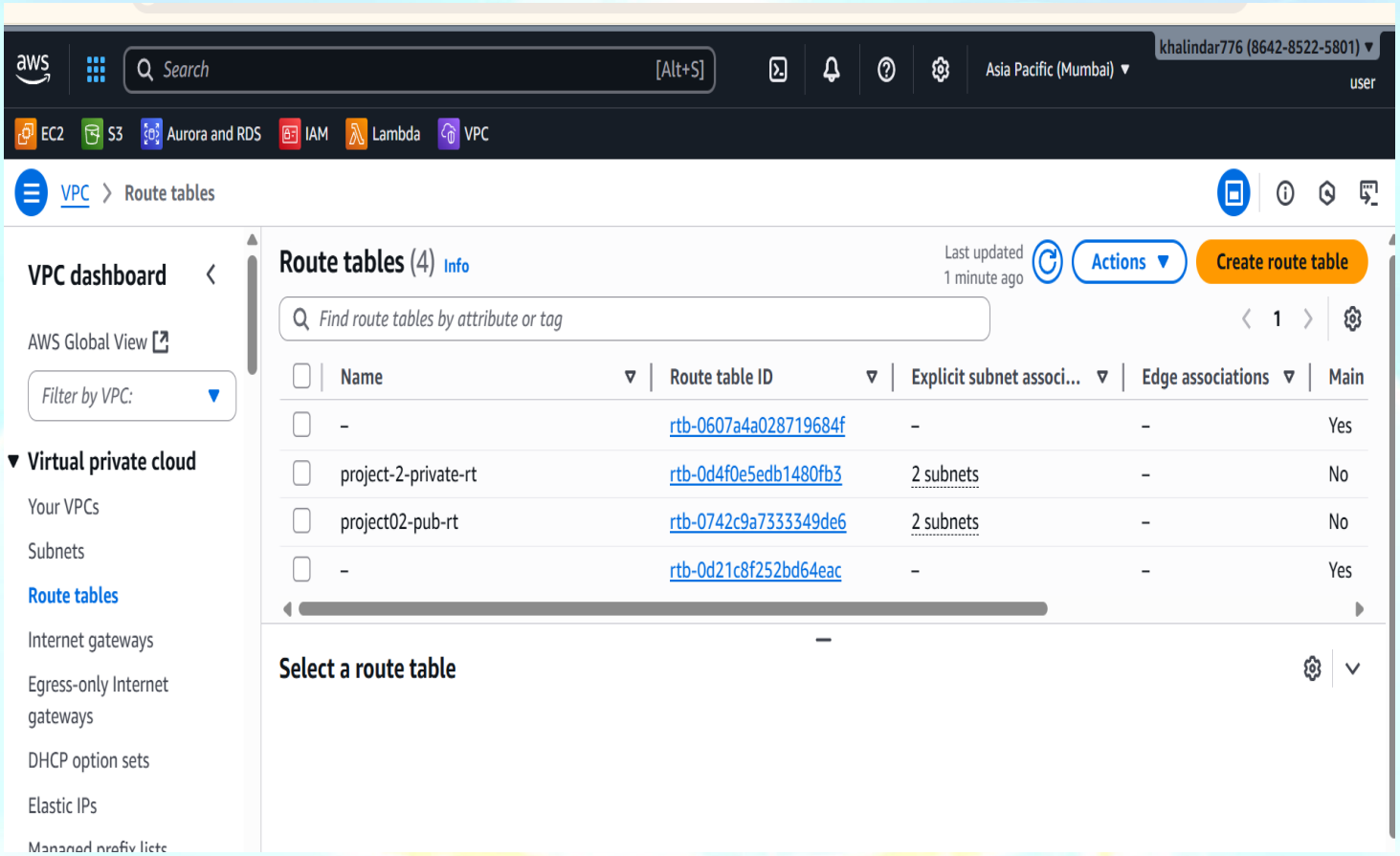
Step 3: Create an Internet Gateway (IGW).

- Create Internet Gateway for public subnets access the internet connection Name : **project-02-ijw** Create IG
- In Actions select and Attach VPC (in Notifications)
- Attach given VPC (Project-02-vpc)



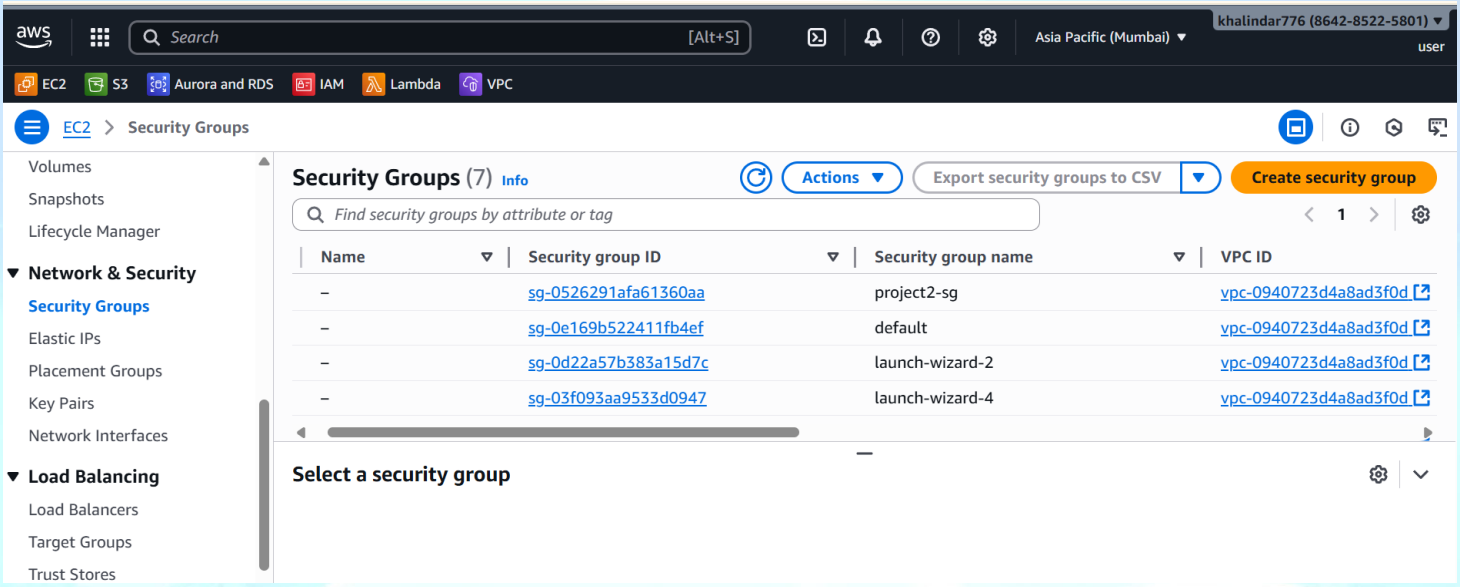
Step 4: Create Route tables (2)

- Create 1 public Route table & 1 private Route table
- Name : **project02-pub-rt**
- Select the existing VPC (project02-vpc)
- Created Root table



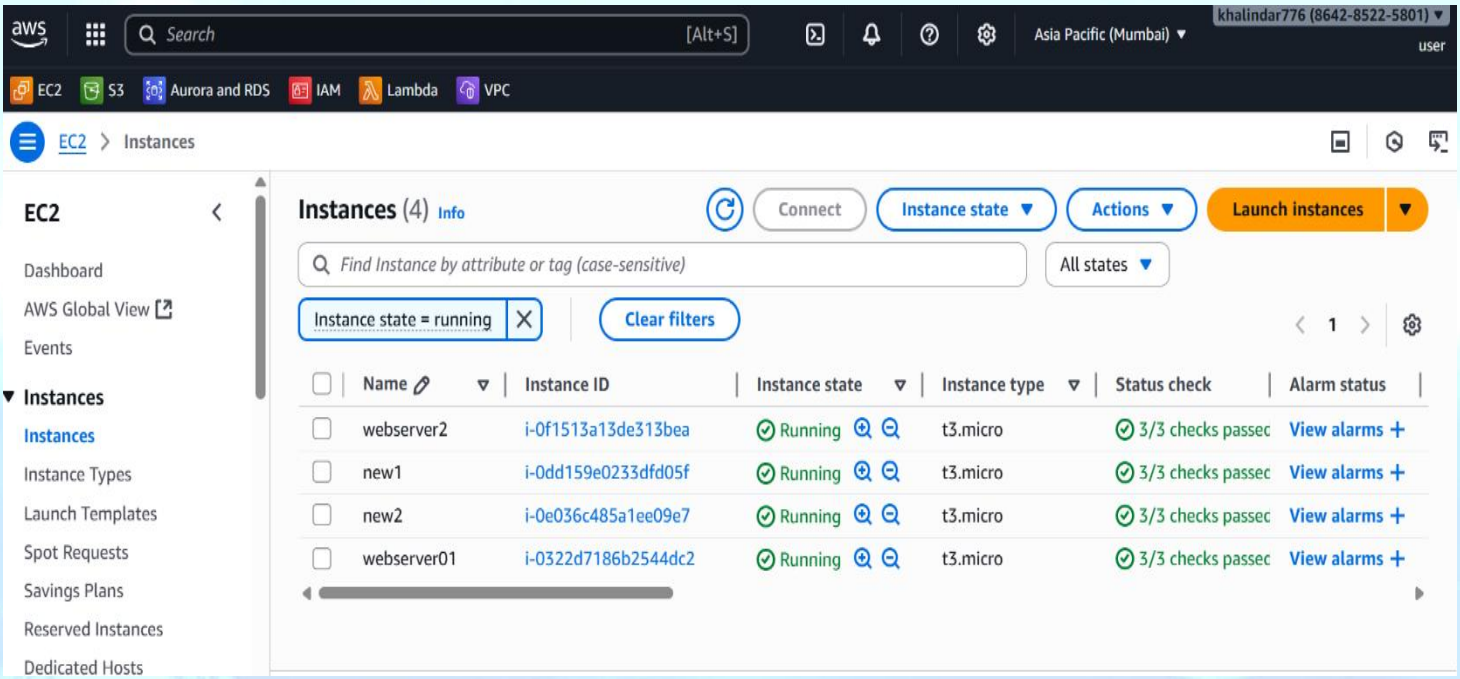
Step5:Create Security Group

- It maintains inbound and outbound traffic rules
- Name : **project02-sg**
- Add inbound rules for SSH (22), HTTP (80), and HTTPS (443) to allow secure access and web traffic. These rules permit incoming connections from all IP addresses, enabling public access to the EC2 instances.
- Create SG



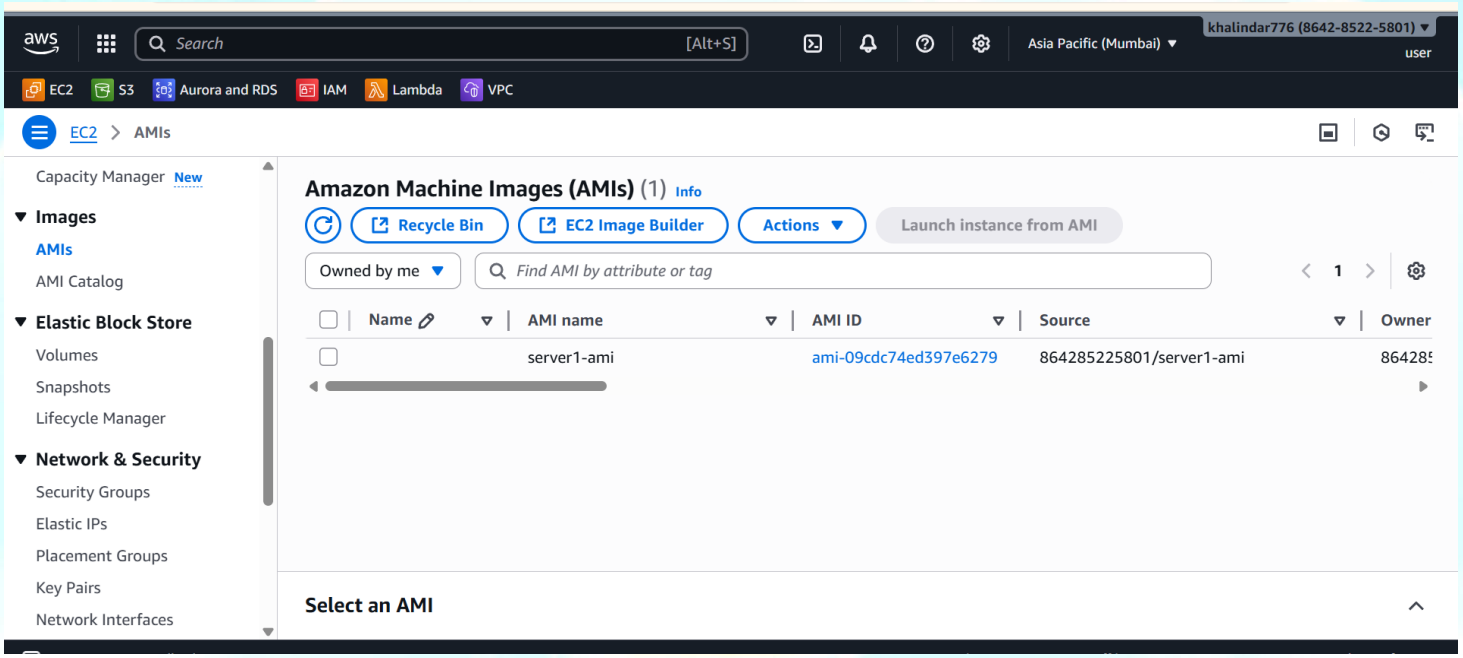
Step 6: Create and launch EC2 instances (2).

- Create two EC2 instances and names (**server1**) & (**server2**).
- Select the existing VPC (project02-vpc).
- Select the Security Group (sgroute53) for 2 instances Launch instance with ubuntu OS, Instance type (t2micro) Select the keypair which we created (Mumbai.pem).
- Configure the Network settings (project02-vpc, Subnets (1a, 1b), SG, And Public IP enables). Click on launch Instance.



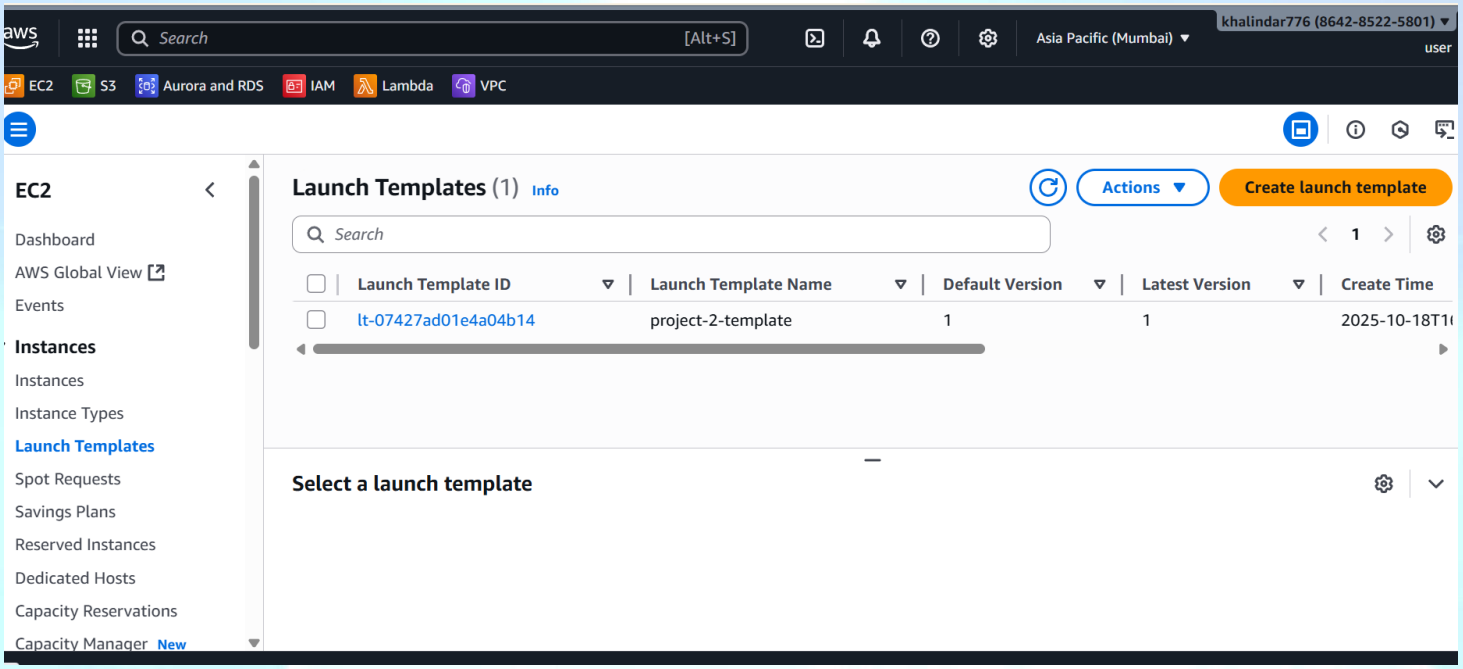
Step 7: Creating AMI's (image)with Public Server

- We need to setup auto scaling on that create AMI and launch template.
- First create a AMI and name is **server1-ami**
- To create an AMI first select the instance and in actions click on image and template option then click on image. Provide the details for AMI
- Create AMI.



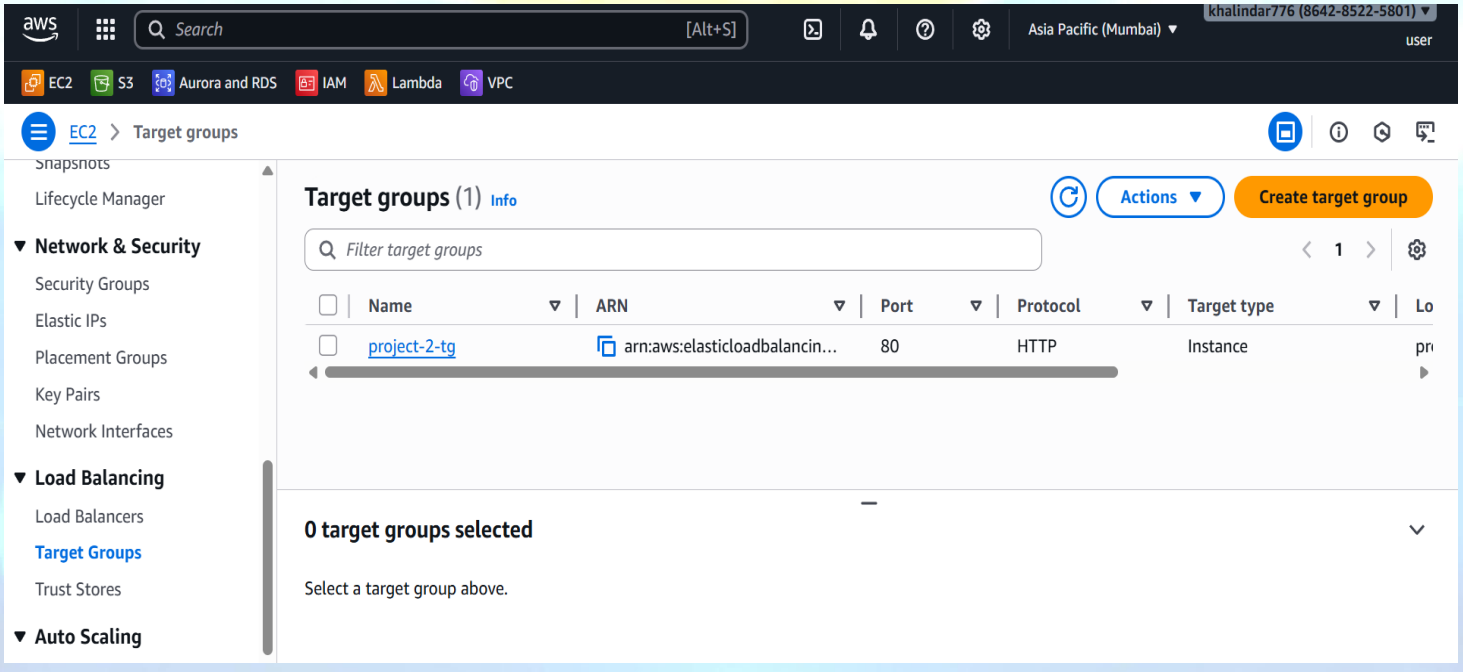
Step 8 : Creating Launch Template

- Click on "Launch Templates" in the left sidebar, then "Create launch template". Configure Template Details.
- Name and description (**project02-template**)
- AMI ID (select existing AMI)
- Instance type (t2.micro)
- Key pair (select existing key pair)
- Security groups (select existing SG)
- Select the security group and enable the auto assign public-ip in advanced network configuration.
- Launch Template.



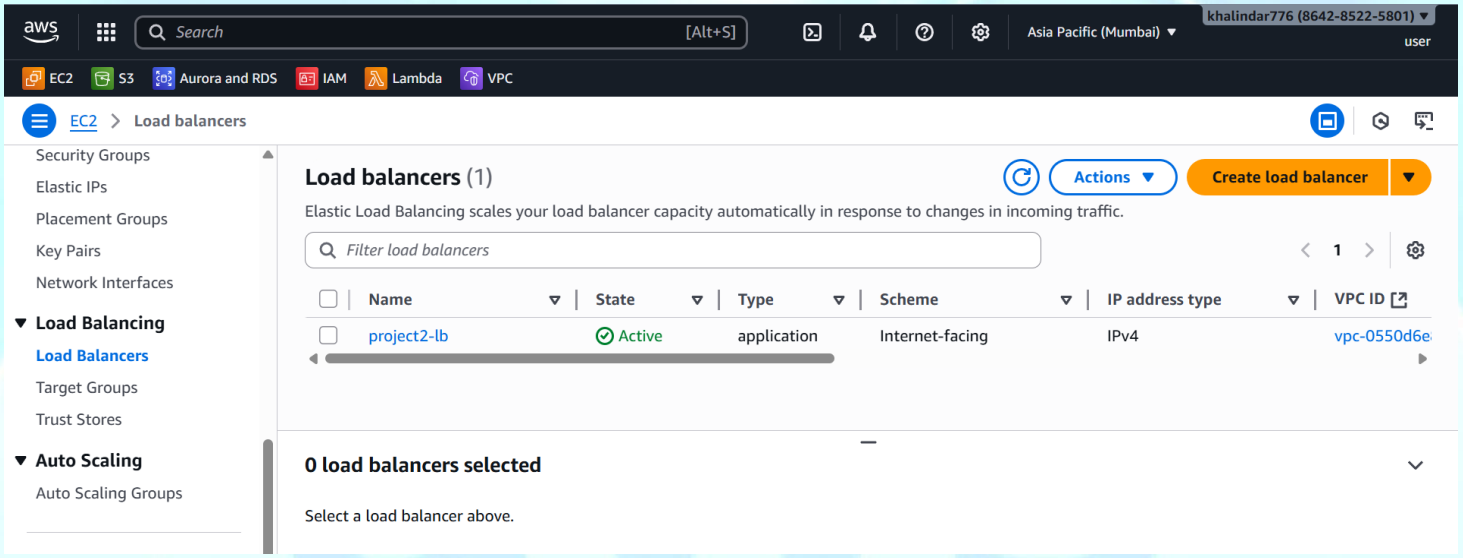
Step 9: Creating Target Group (TG)

- Before creating a load balancer we need to create target group.
- Click on create target group and name is **project02-TG**
- Select the existing VPC (project02-vpc)
- Register the Targets (server1& server2) with include as pending
- Create TG



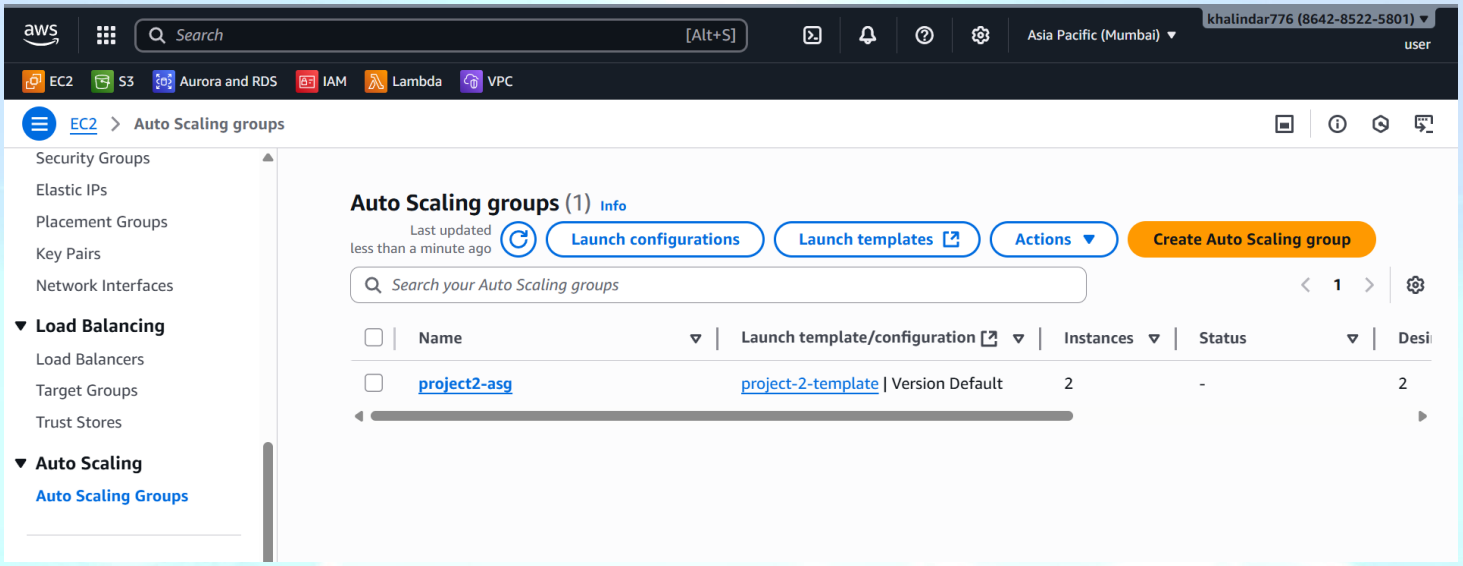
Step 10: Creating Load Balancer(LB)

- Now Go to load balancer page and click on create. Select Application Load Balancer.
- Then provide the LB name is **project2-lB**
- Select the VPC and select the public Availability Zones(2) Select the existing target group (TGroute53) in default action.
- Create LB



Step 11 : Creating Auto Scaling

- As per the requirement we need to create auto scaling group Create an Auto scaling Group and name is **project2-ASG**.
- select the existing template (template-route53) and image (image-Route53).
- And select the existing VPC (vpc-Route53) and availability zones (Pub-1-route53, Pub-2-route53).
- Then select on attach to an existing load balancer(LB-route53) and target group (TG-route53)
- Now provide the details for **desired capacity** (2) in group size.And provide details for **min desired** (2) and **max desired capacity** (4) in scaling section.
- Select the target tracking policies. In Desired policies and **target value** (ex:60%), and **instance warmup** (ex:100 secs).
- Create Auto scaling Group.
- Once created the ASG and then automatically created the desired capacity servers (2).



Step 12 : **Connect Server-1 to Terminal with SSH String (Using below commands)**

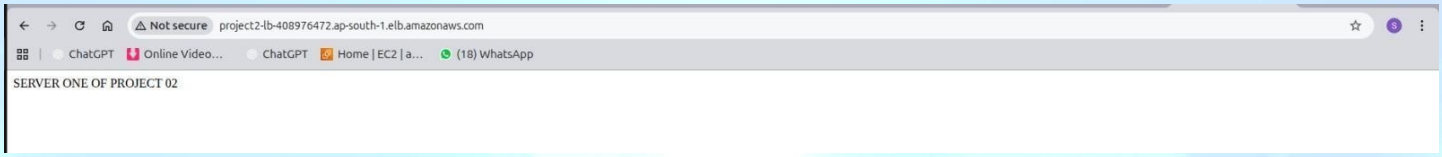
- Go to server-1 and connect the terminal with SSH string
- Change the ubuntu to root user : **sudo -i**
- Update the server : **sudo apt update –y**
- Install the apache2 software in server **: sudo apt install nginx**
- Then check the index.html file : **cd /usr/share/nginx/html**
- Remove the existing index.html file : **rm index.html** Create the file : **vi index.html**

  Insert the data : Ex- **SERVER ONE OF PROJECT 02**

- Restart the server : **systemctl restart nginx**
- Check the output with Public ip/LB-DNS name : **54.209155.54:80 /LB-DNS name in web browser**
- Even you can check in terminal with private ip : **curl 120.0.0.178:80**

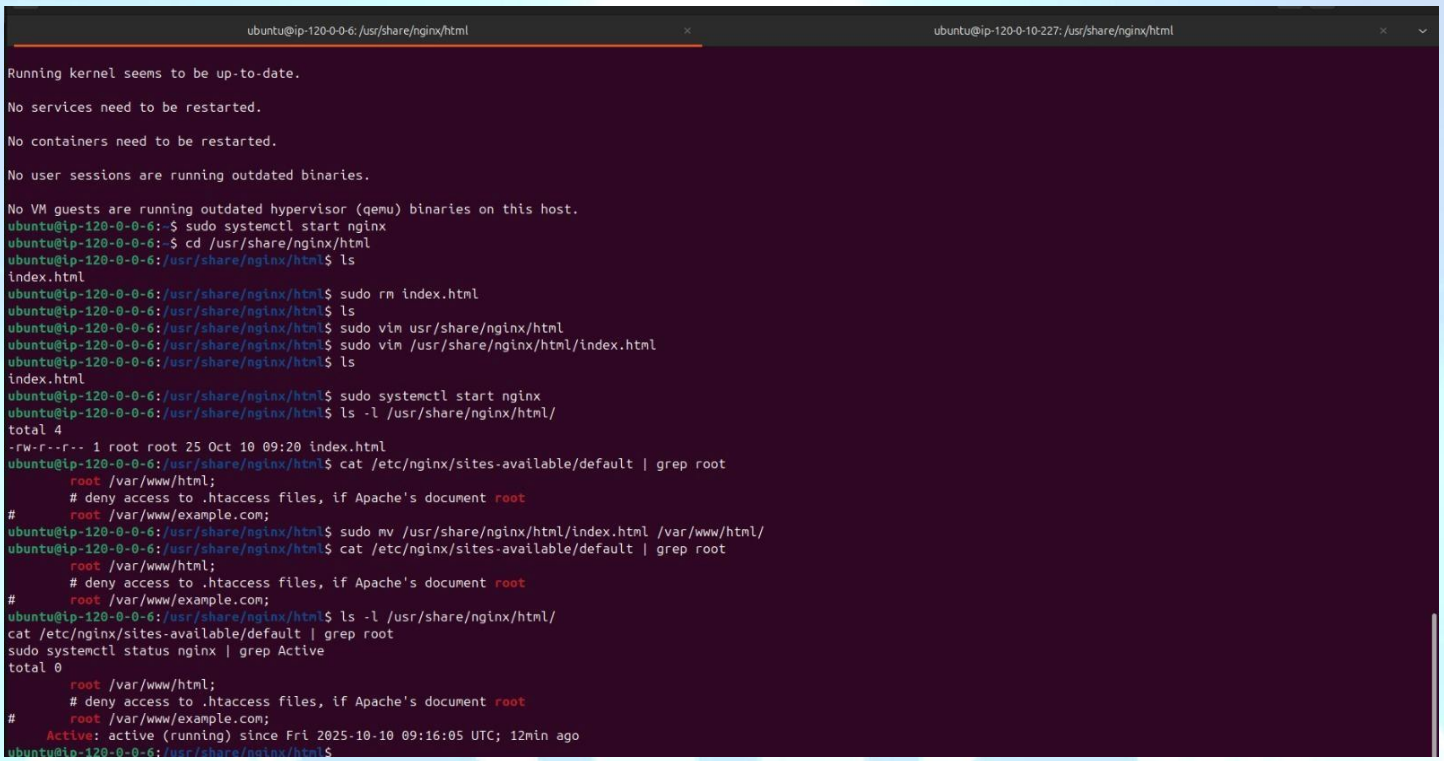
Output:1

- **The output for server-1 below, but it's not secure.**



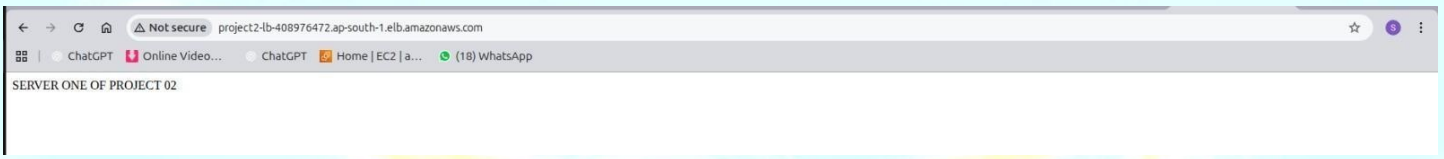
Step 13 : **Connect Server-2 to Terminal with SSH String (Using below commands)**

- Go to server-2 and connect the terminal with SSH string
- Change the ubuntu to root user : **sudo -i**
- Update the server : **sudo apt update –y**
- Install the apache2 software in server **: sudo apt install nginx**
- Then check the index.html file : **cd /usr/share/nginx/html**
- Remove the existing index.html file : **rm index.html**
- Create the file : **vi index.html**
- Insert the data : Ex- **SERER ONE OF PROJECT 02**
- Restart the server : **systemctl restart nginx**
- Check the output with Public ip/LB-DNS name : **3.85.243.100:80 /LB-DNS name in web browser**
- Even you can check in terminal with private ip : **curl 120.0.10.247:80**



Output: 2

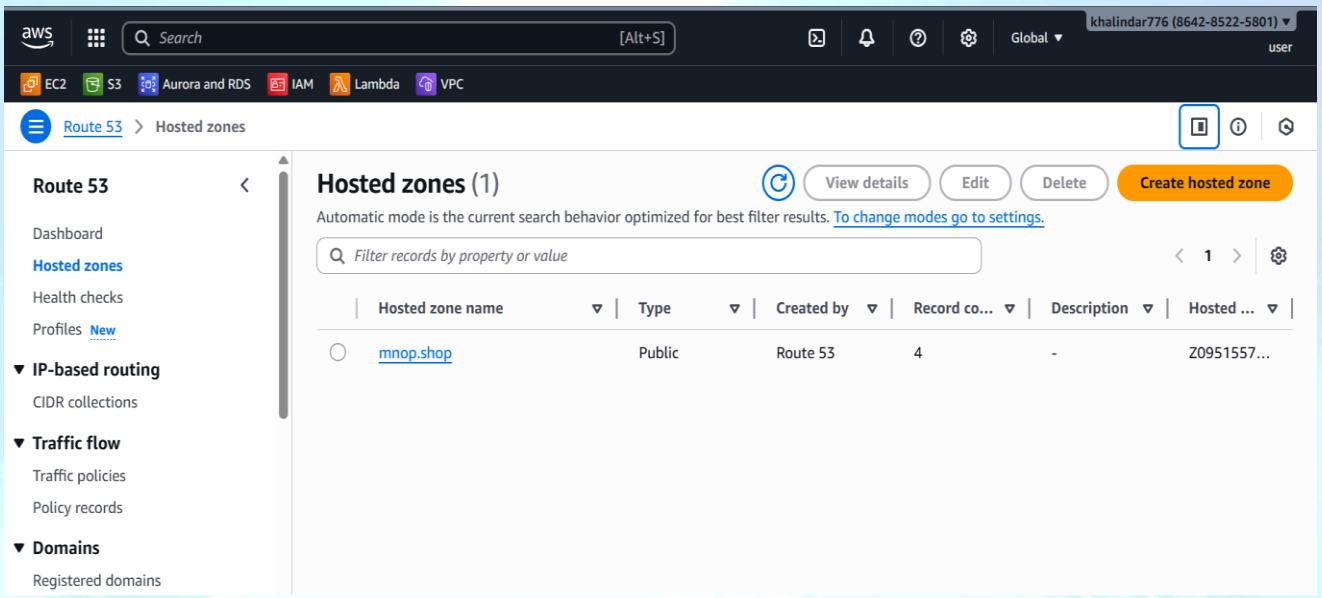
- **The output for server-2 below, but it's not secure.**



Note : For Secure the severs , we need to add the HTTPS (443) in LB (Add Listener)

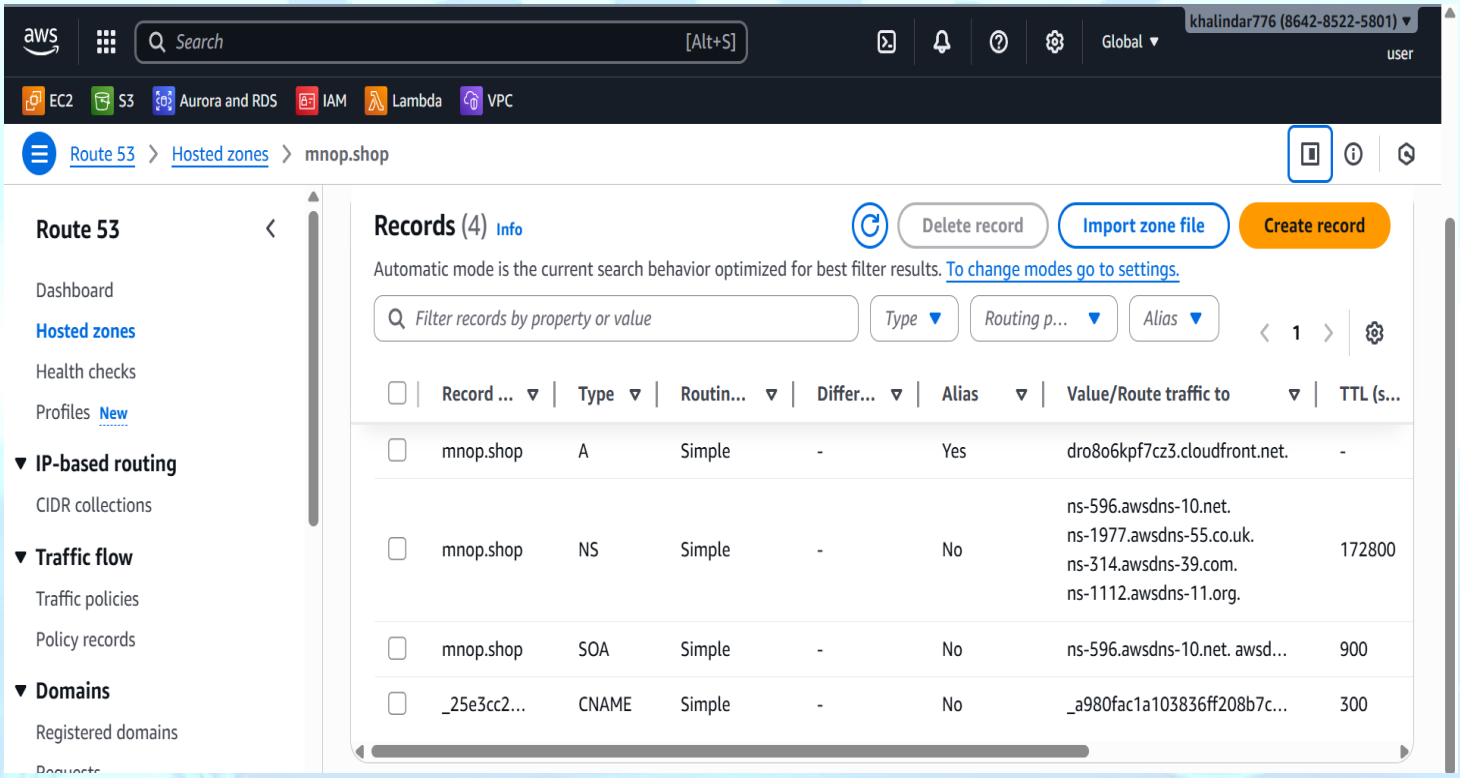
Step 14: Configure DNS with Route-53

- Create the Hosted Zone in Route 53 for the registered domain.
- Domain name is **mnop.shop.**



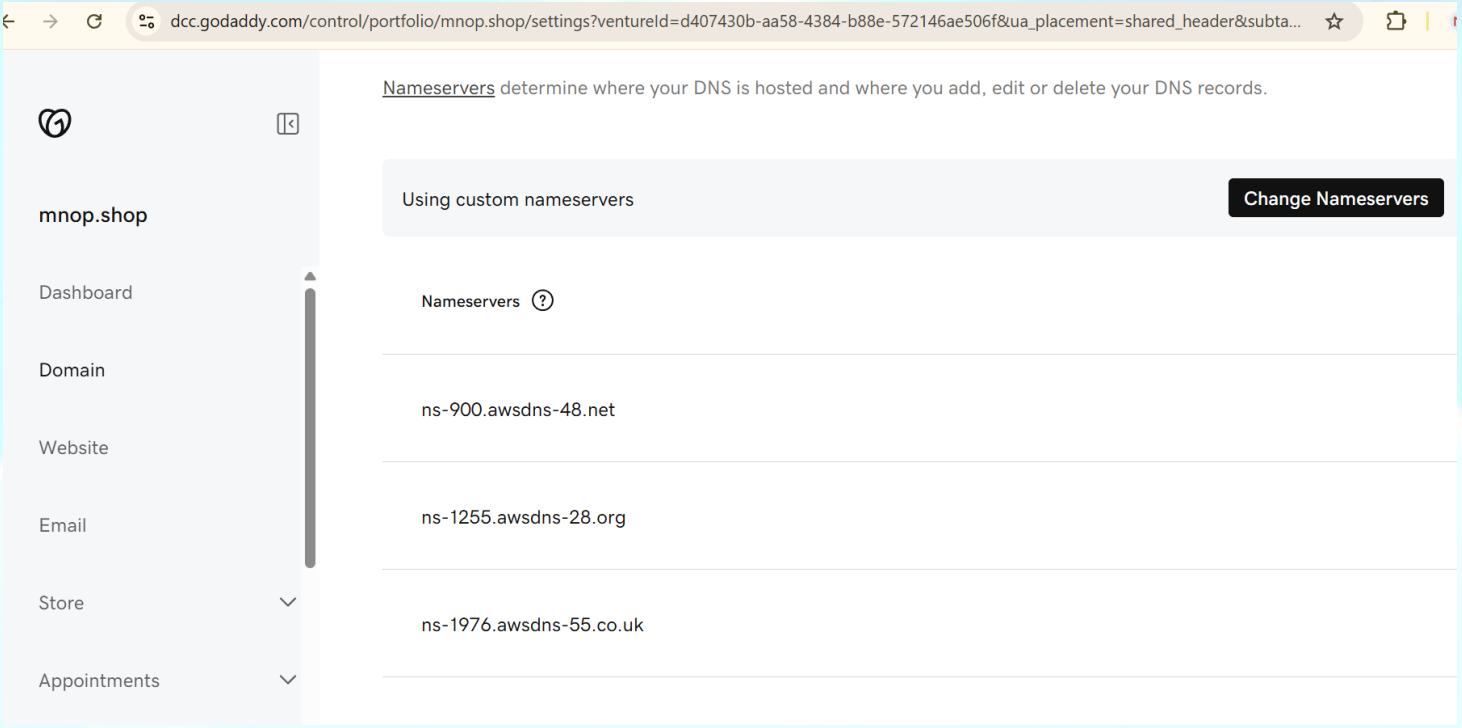
Hosted zone :

- While we create the Hosted zone automatically generate the default records (i.e NS and SOA)
- Name servers and SOA created under the DNS name mnop.shop



Go Daddy : **Registered a custom domain name using a third-party (e.g., GoDaddy)**

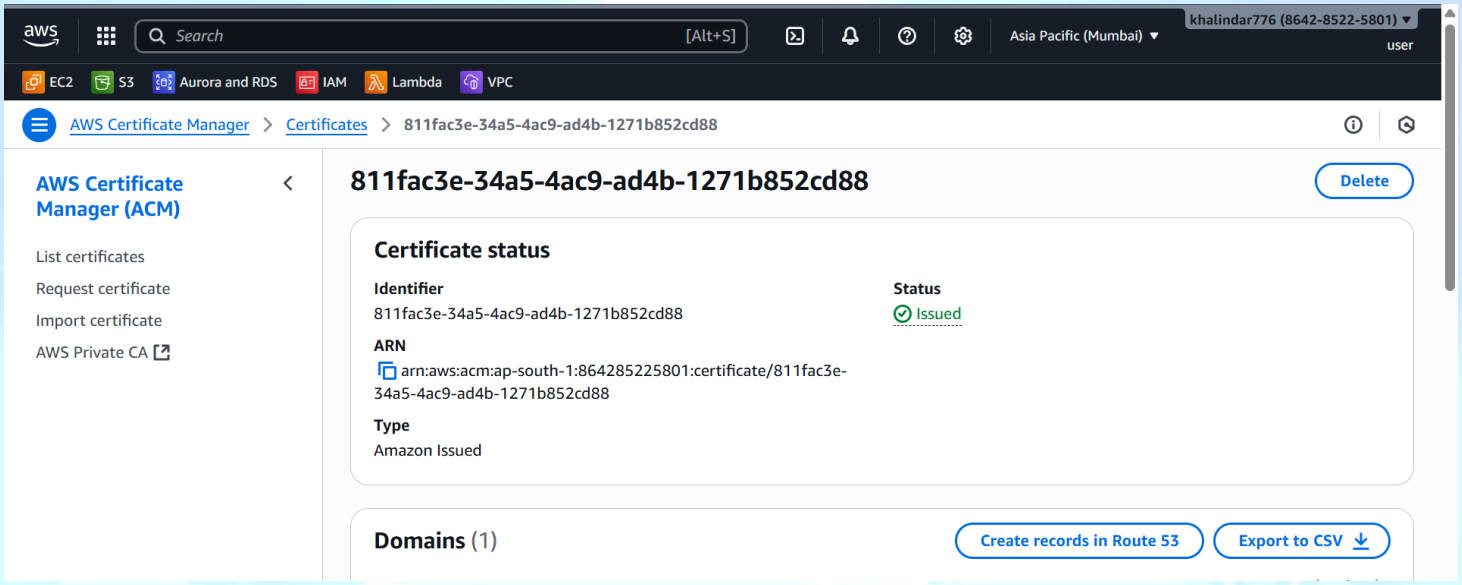
- Go to Go Daddy and login by using the existing credentials.
- Then check with domain and Name servers page with default servers are available.
- whatever the name servers generated in Hosted zone .Add/modify those server details to Name servers in DNS(Go Daddy-reference below attachment)



Note:As of now created the Hosted zone and Records. But when we created the Amazon Certificate Manager(ACM) the same time created the CNAME in Hosted zone (reference below attachment-Step 15)

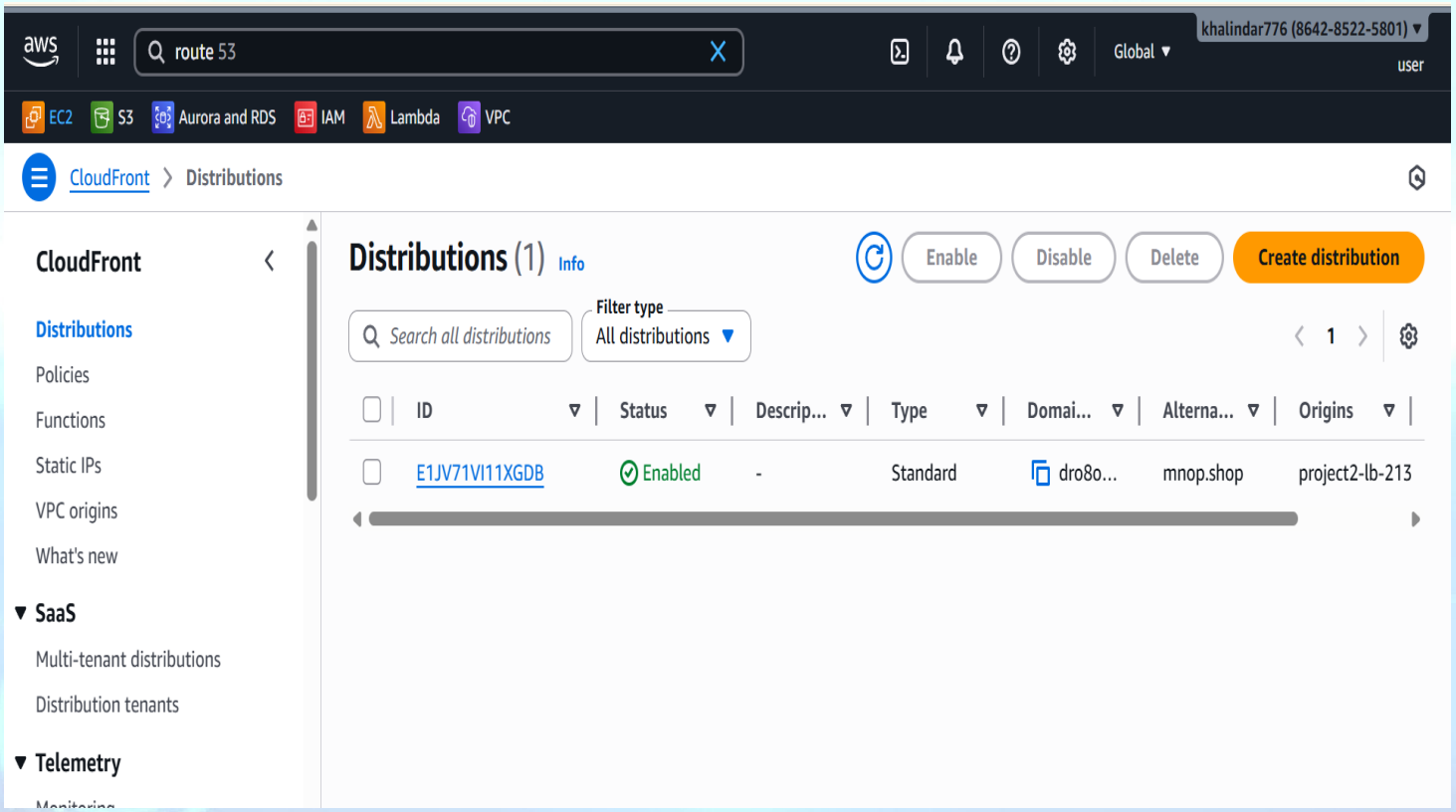
Step 15: ACM (Amazon Certificate Manager)

- Go to **ACM** and Request public certificate
- Domain: mnop.shop
- Validate via DNS
- Ensured secure routing by integrating with ACM (AWS Certificate Manager) for SSL/TLS Actually it takes time to issue the certificate status (have to wait some time).
- Once issued the certificate, and then automatically create the CNAME in Hosted zone (Route-53).



Step 16: Cloud Front (Content Delivery Network) with securely

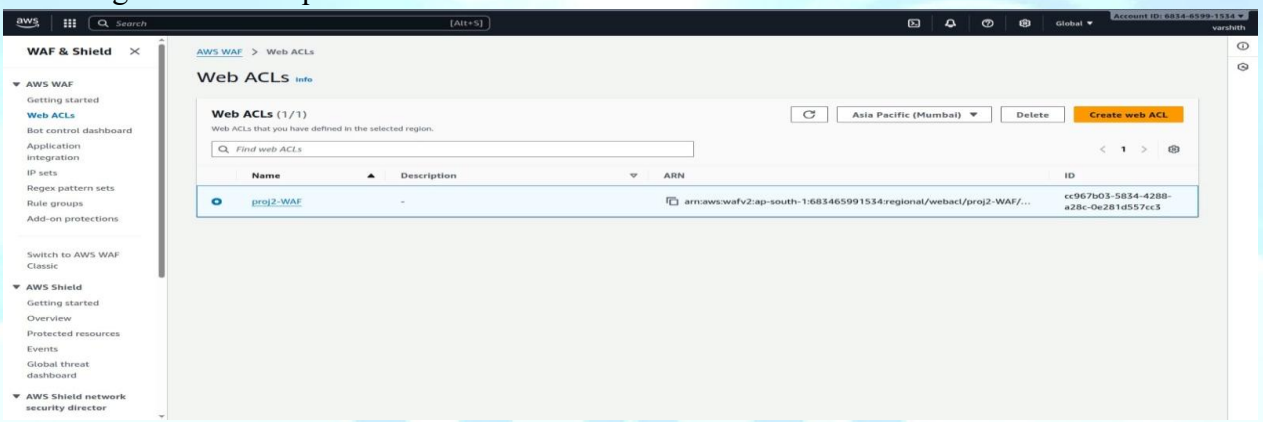
- Create Cloudfront Distribution and the name is **cloudfront-route53.**
- Created a **CloudFront Distribution** to serve static and dynamic content globally Linked the distribution to the ALB/ S3 bucket for origin content.
- Configured **Caching Behavior** to optimize performance and reduce latency



Step 17: WAF (Web Application Firewall) and shield {Secure Your Application from Threats}

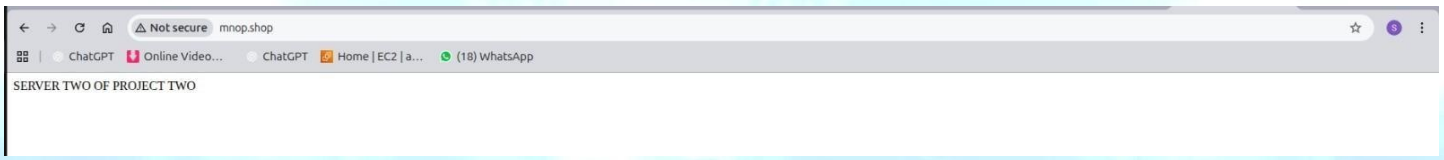
Web ACL:

- Created a **Web ACL** to define security rules for the application
- Attached the Web ACL to the Application Load Balancer (ALB)
- Monitored traffic using **WAF Logs** via CloudWatch for insights Validated rule effectiveness by simulating malicious request.



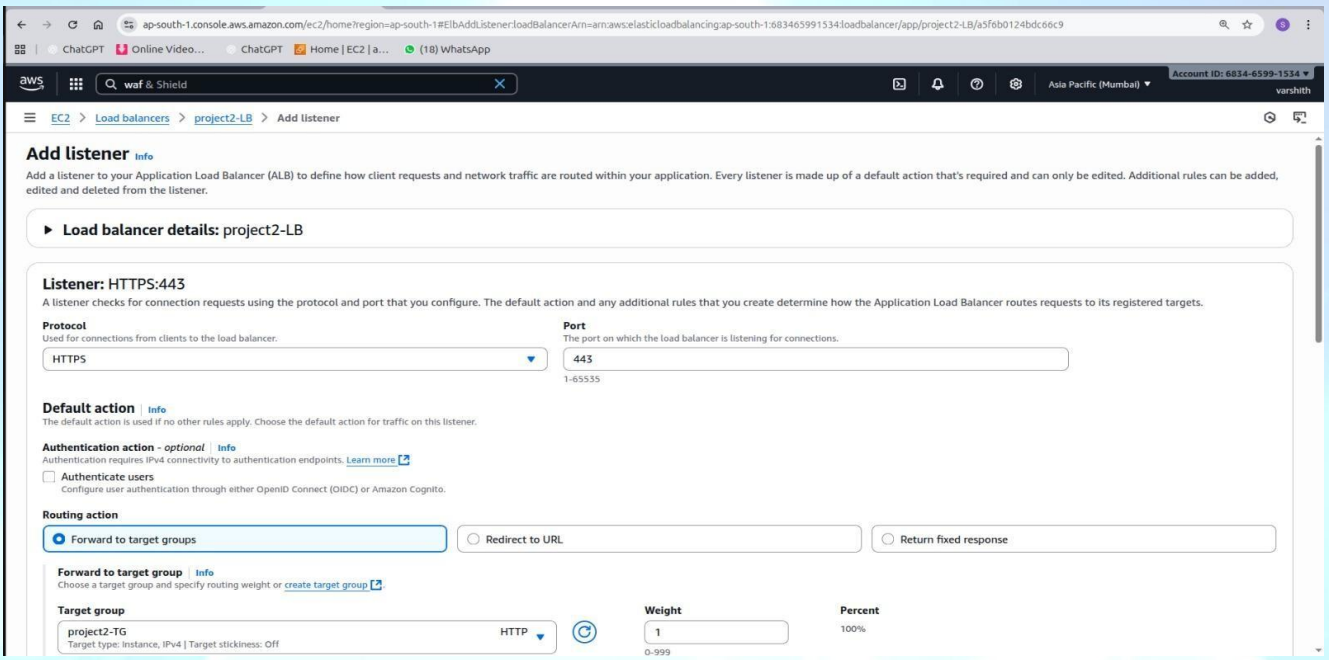
Step 18: Output

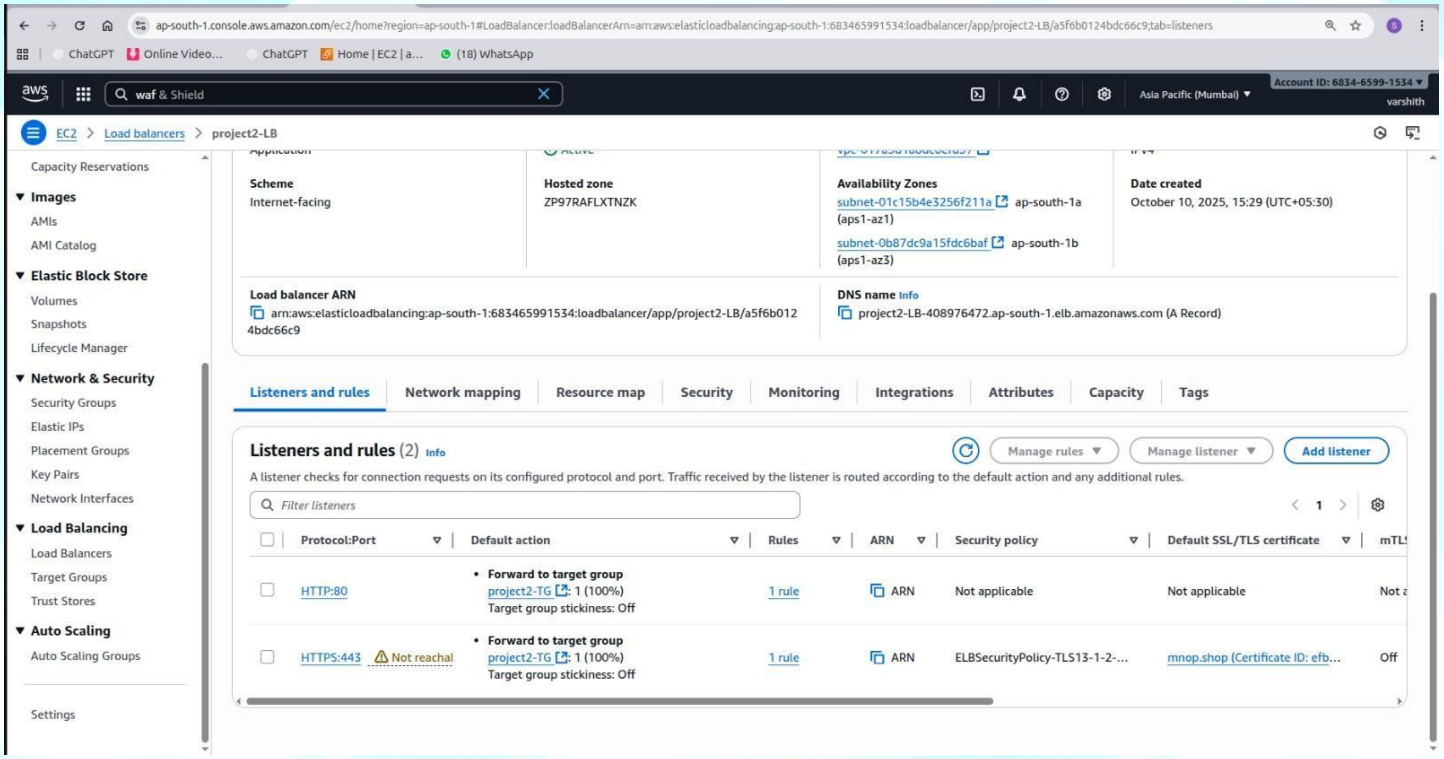
- Here we have to check the servers with DNS name (mnop.shop) still it is not secure.So,we need to add the Listeners for security purpose.



Step 19 : Add Listener for secure the connection – HTTPS (443)

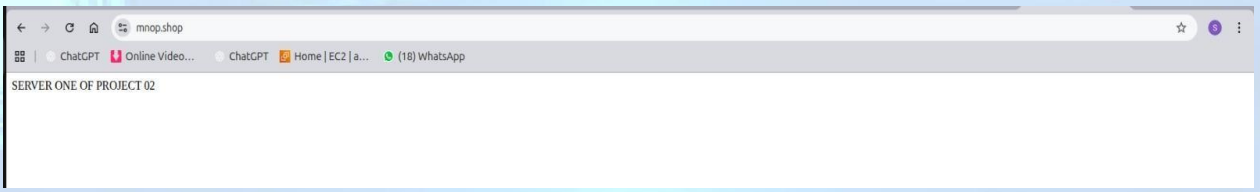
- Go to Load Balancer and add Listner.
- Select the protocol (HTTPS - 443).
- Select existing Target Group (TG-route53) and ACM (DNS Name) Finally added the Listener ,it shows HTTP-80 and HTTPS-443.





Step 20: Final output

- Check with DNS like mnop.shop in web server then the output shows the connection is Secure for both server-1 and server -2.



Conclusion

The depicted AWS Cloud architecture offers a robust and well-structured foundation for deploying modern web applications. By strategically integrating services such as Route 53, CloudFront, WAF, Application Load Balancer (ALB), EC2 with Auto Scaling, and RDS within a Virtual Private Cloud (VPC), it ensures a solution that is inherently scalable, highly available, secure, and performance-optimized. This allows businesses to concentrate on innovation and application development, while AWS handles the complexities of infrastructure management.

THANK YOU