PROJECT ASSIGNMENT

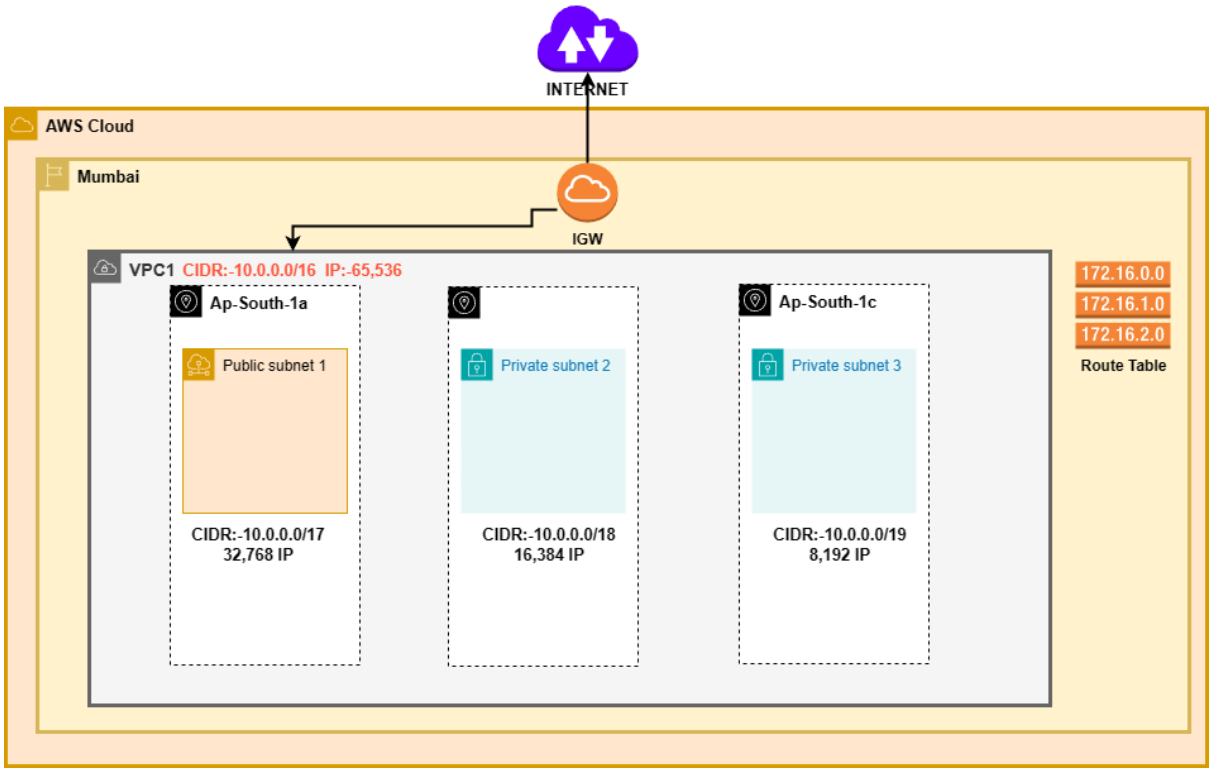
BATCH ID: UA003AWS

Name: -Deepak Patra

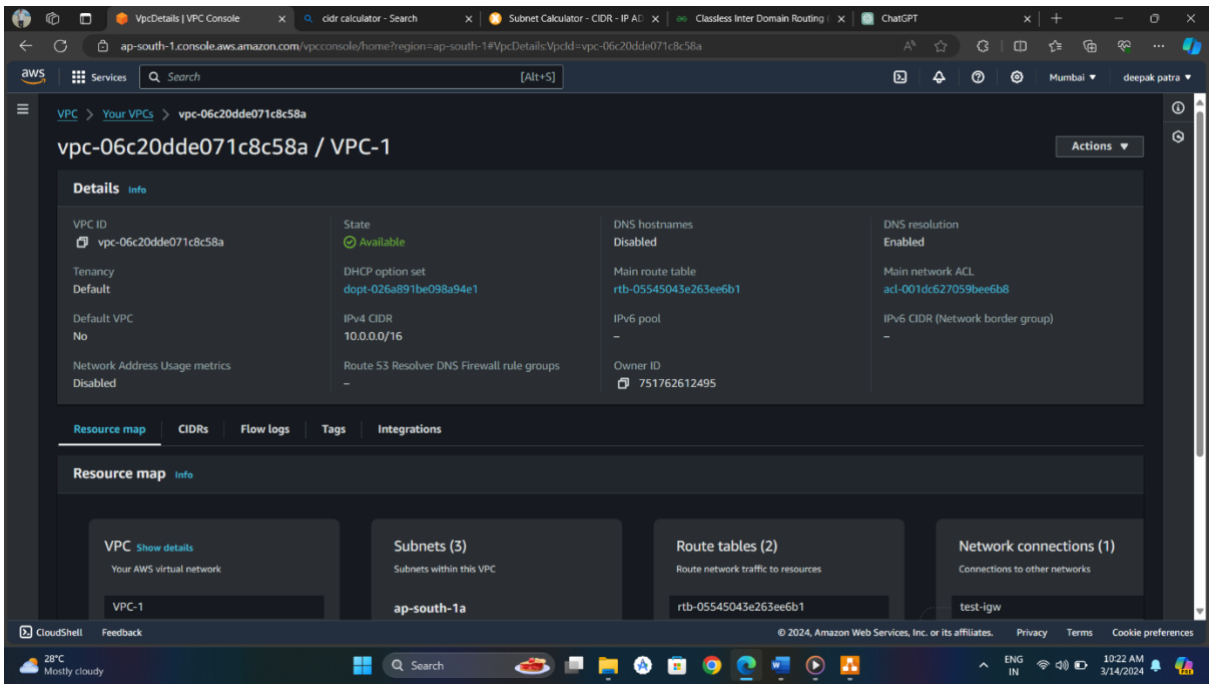
Ref No: - UA/BLR/000013

1. **Create a VPC, e.g., VPC1, and create three subnets within it: one that is half the size and two that are quarter-sized. The larger subnet should be public, while the other two should be private.**

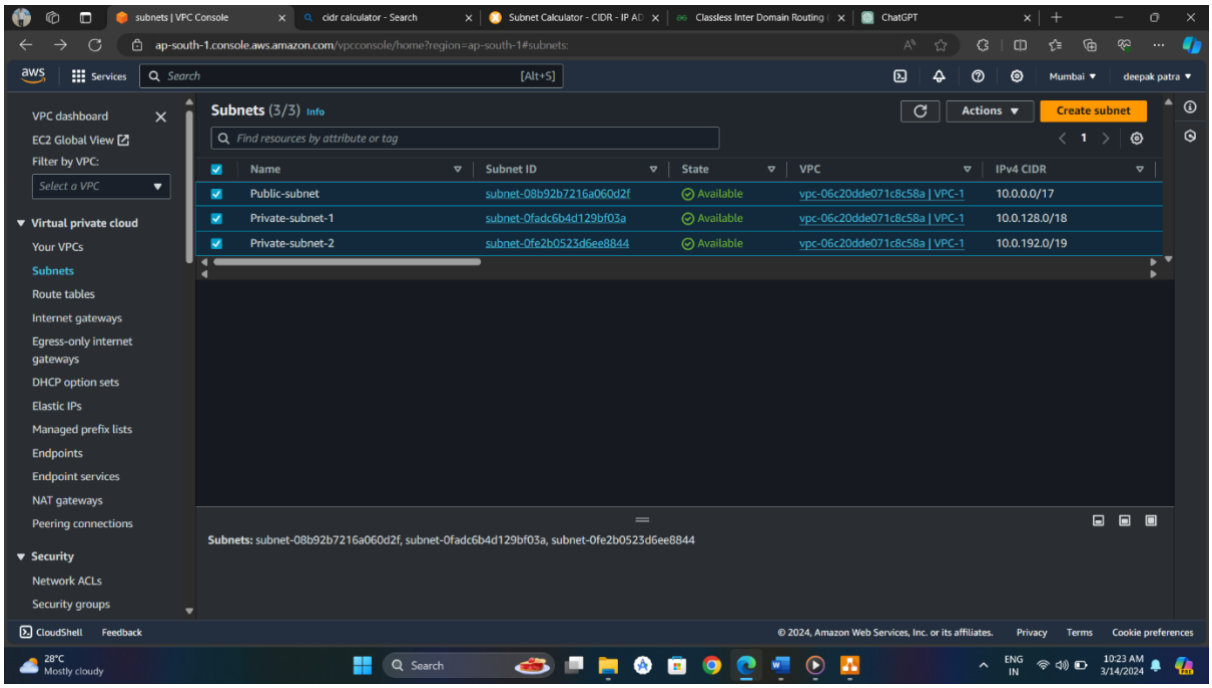
Architecture: -



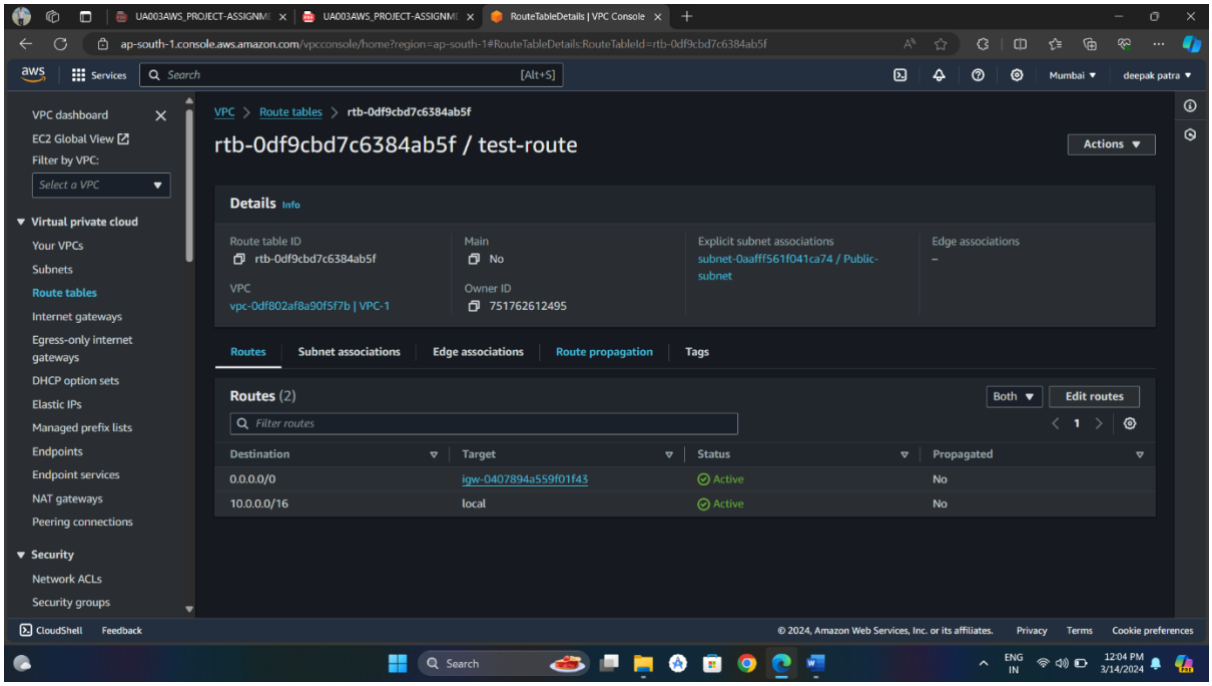
VPC Screen: -



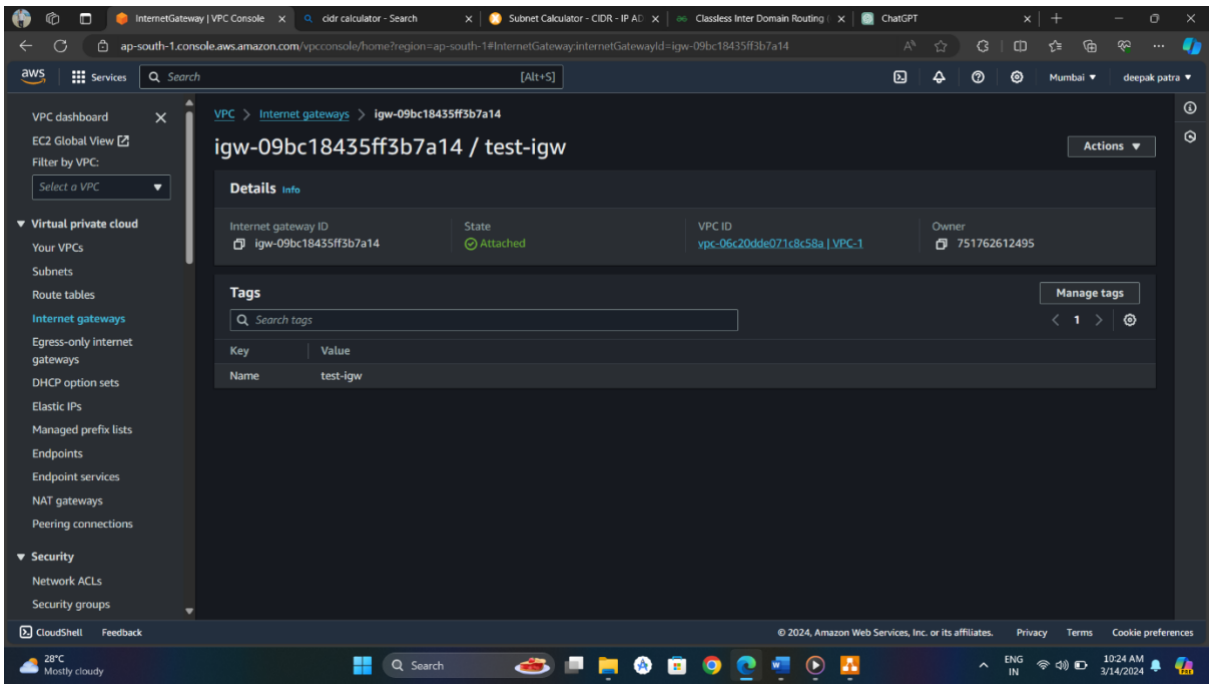
Subnet Screen: -



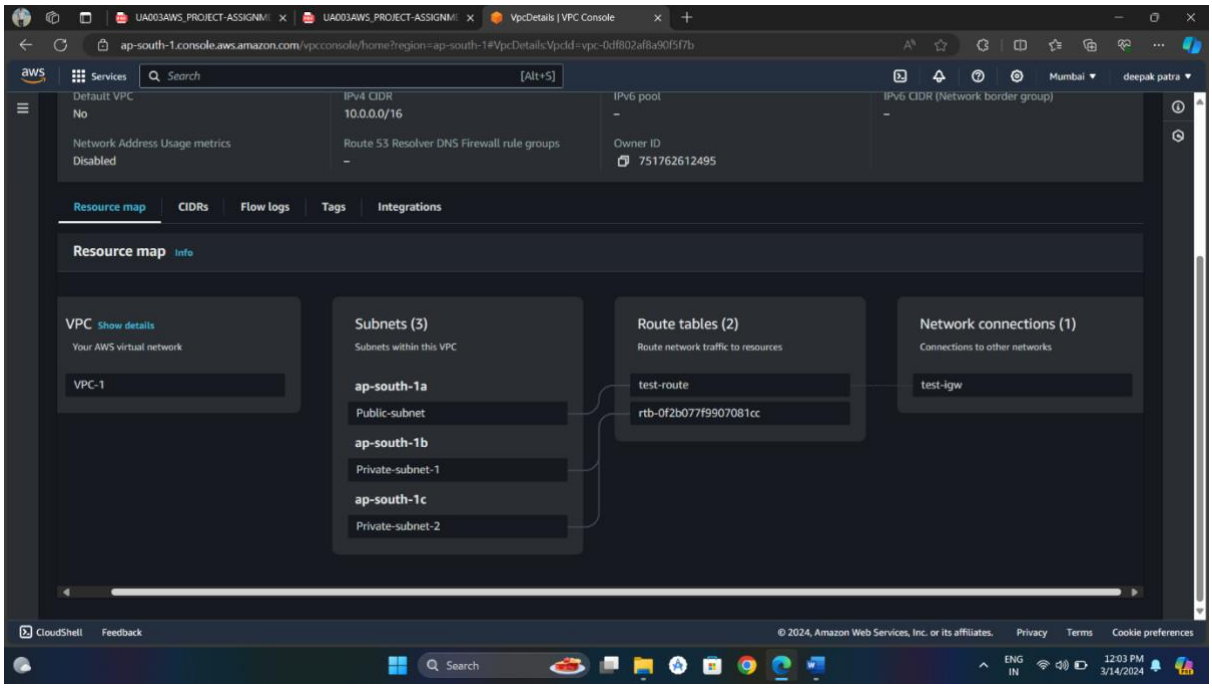
RTB Screen: -



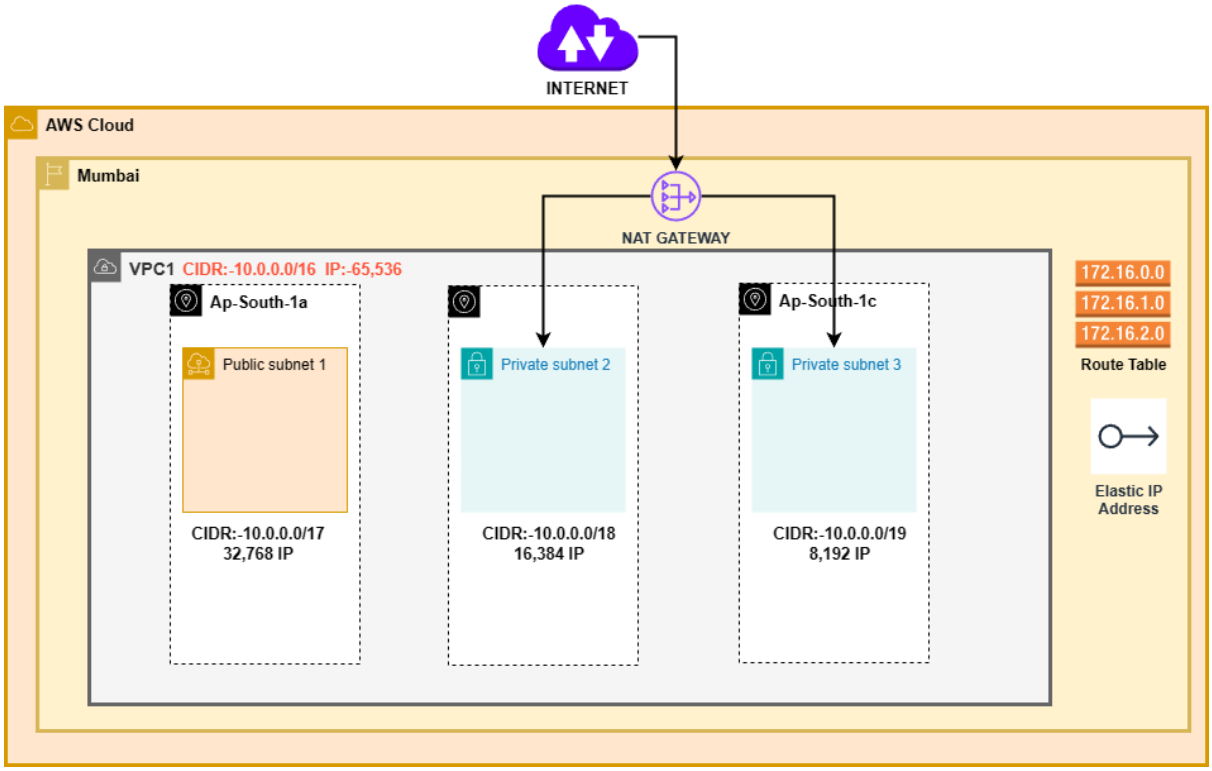
IGW Screen: -



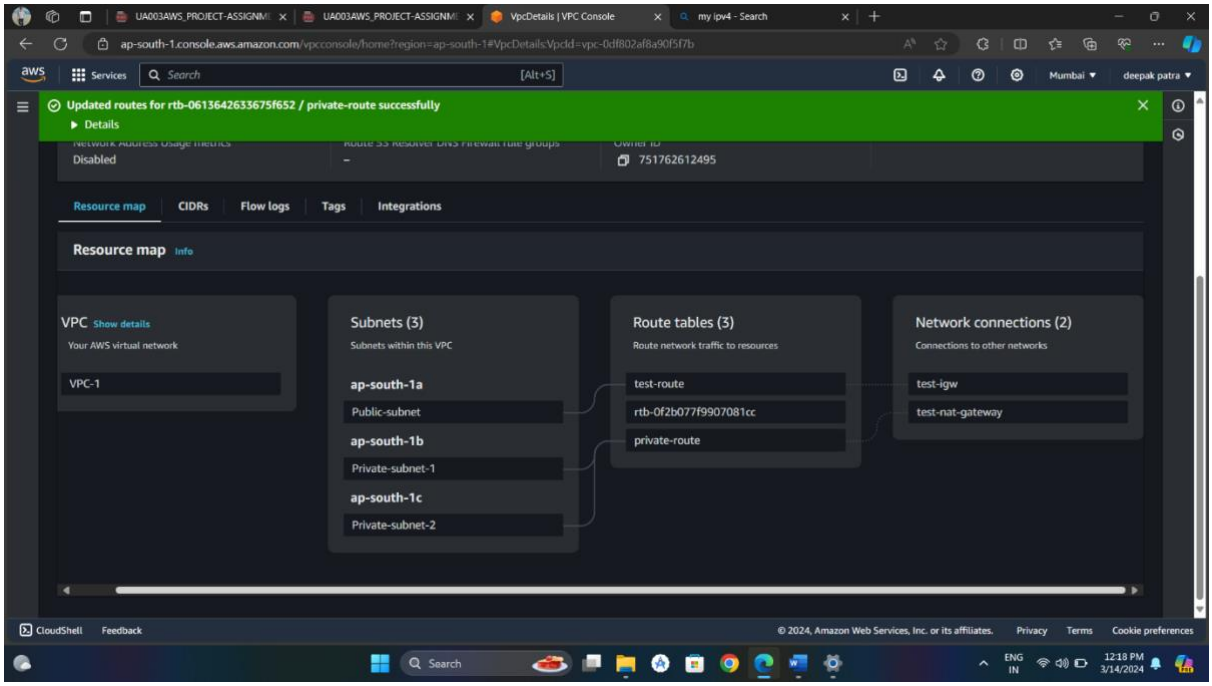
VPC RESOURCES MAP: -



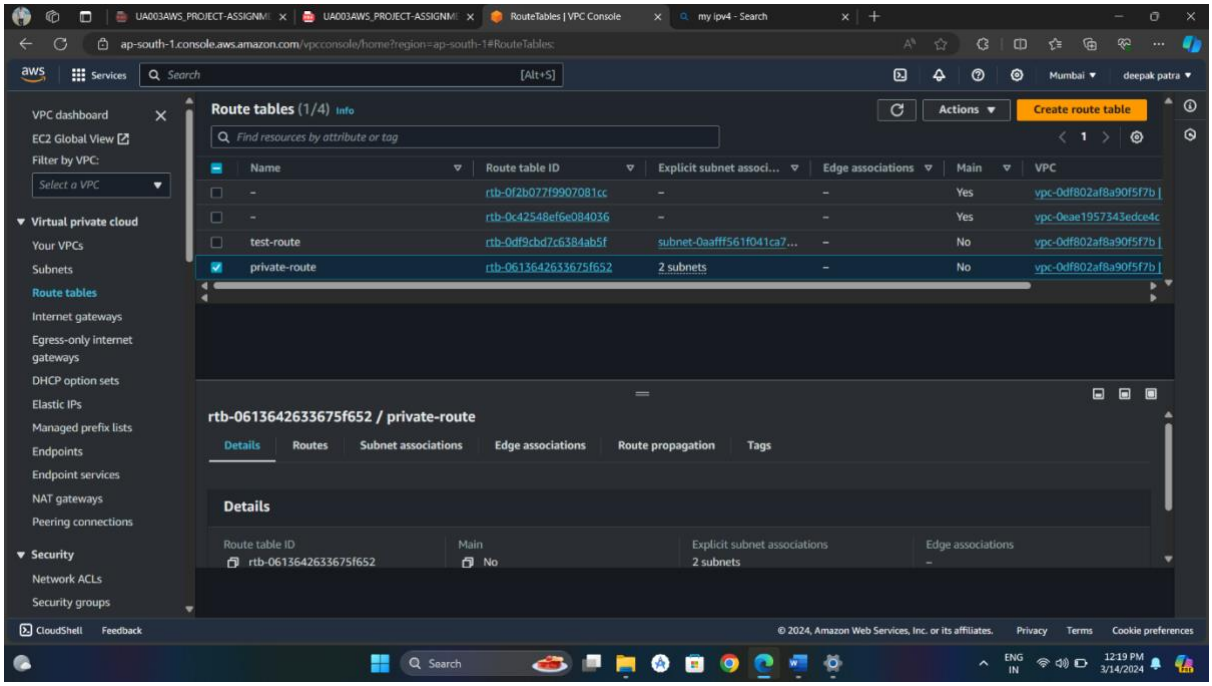
2. Establish a NAT gateway connection for subnet 2 and subnet 3 in VPC1.



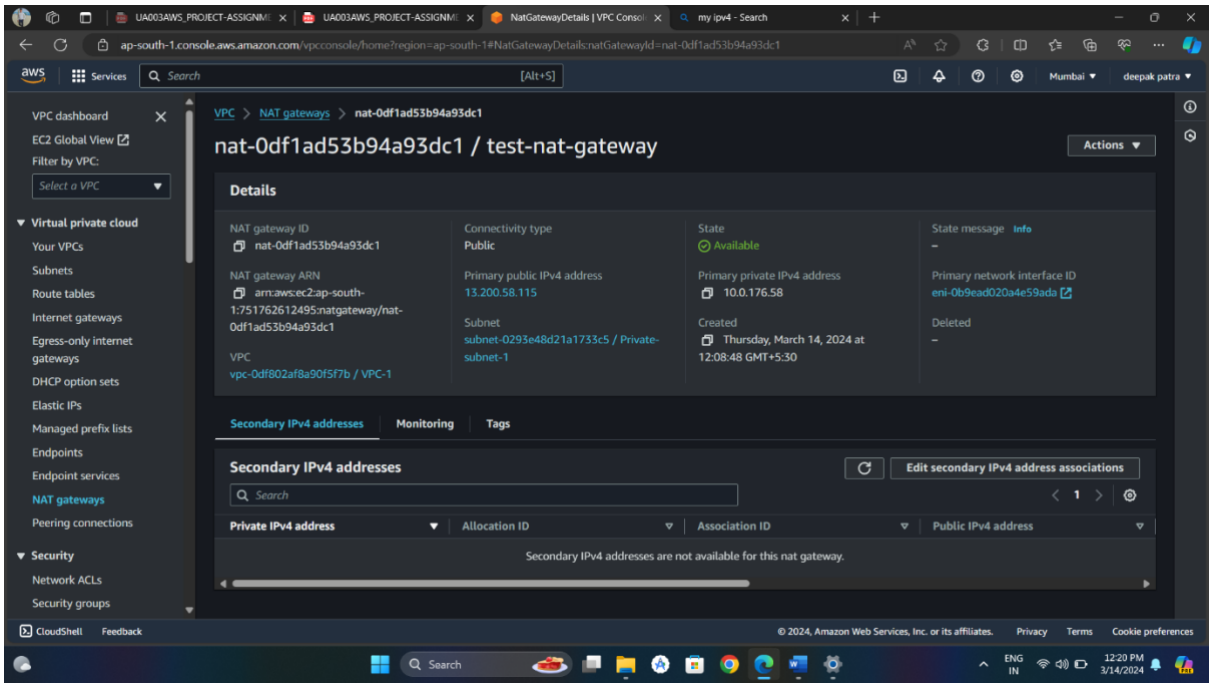
Resources map



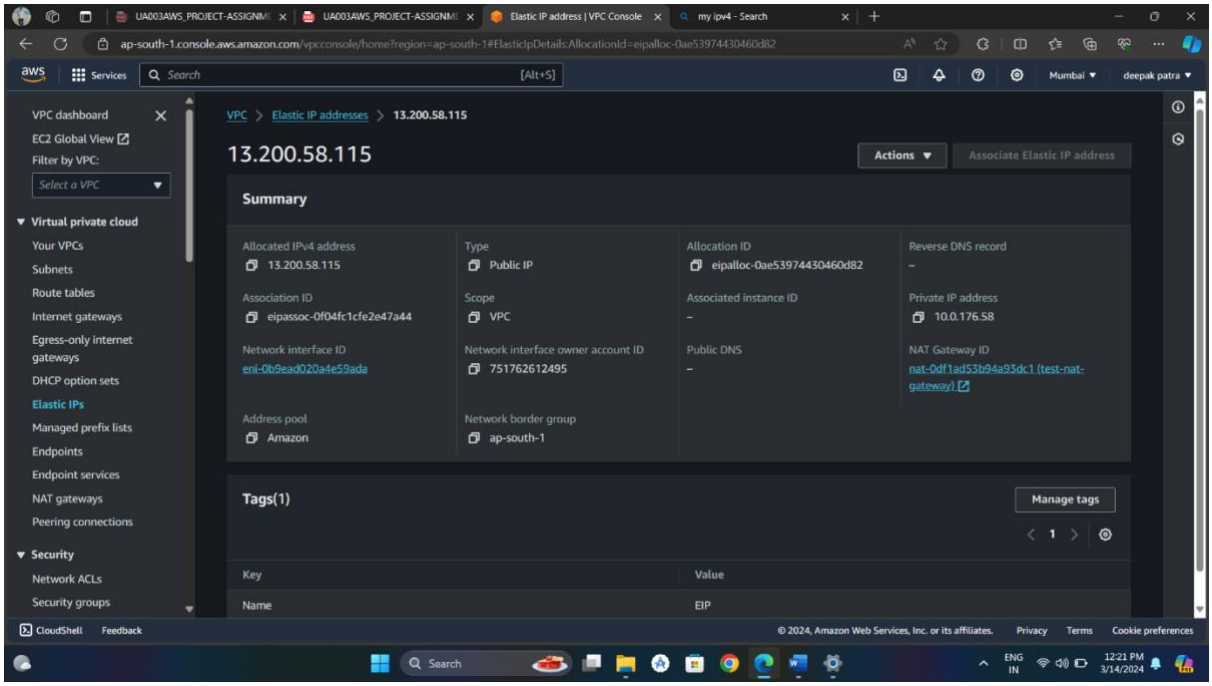
RTB SCREEN



NAT SCREEN

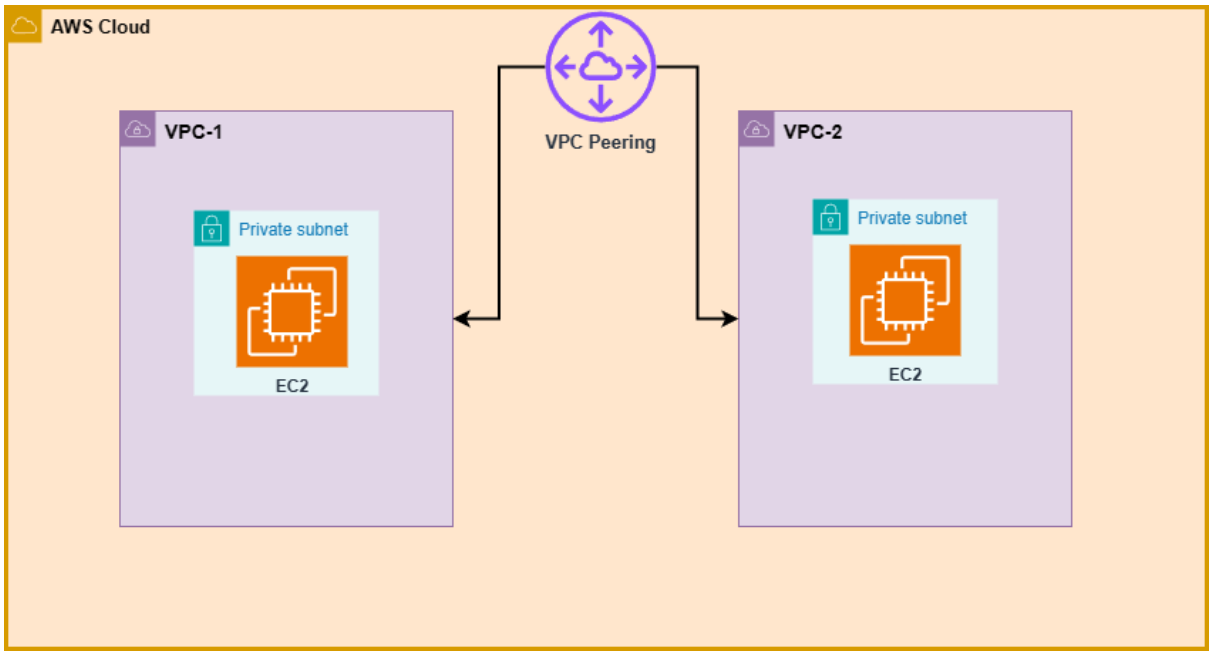


EIP SCREEN

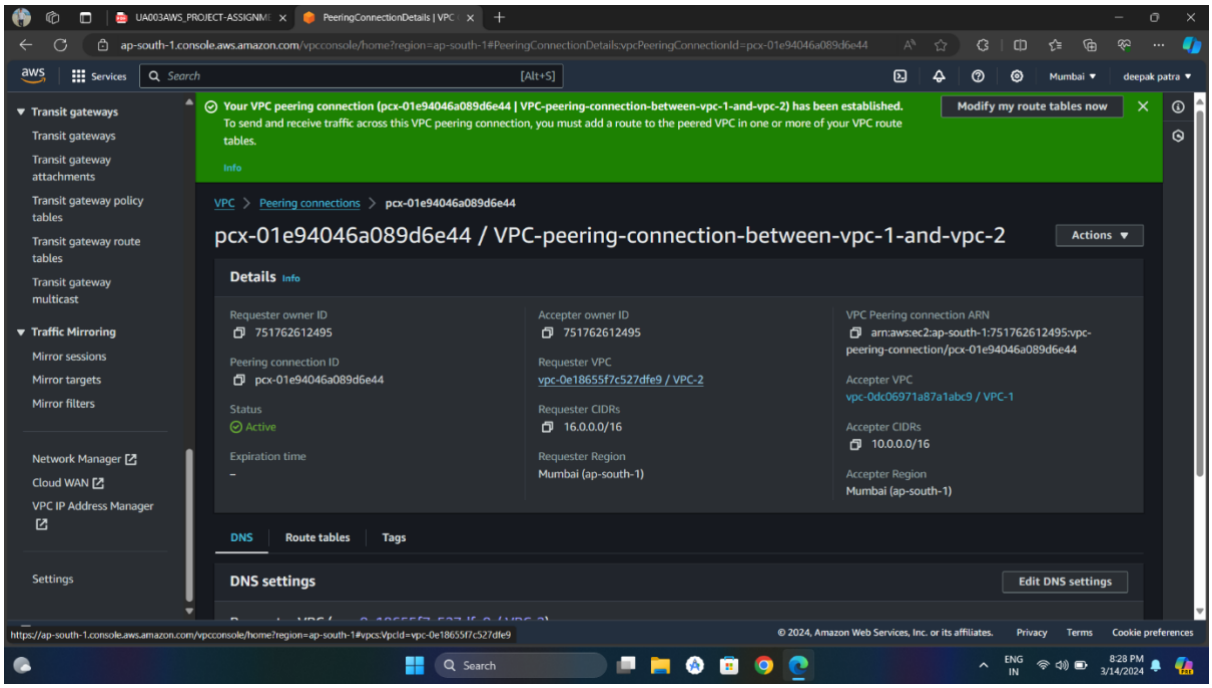


3. **Create another VPC, e.g., VPC2, and set up one subnet within it without internet connectivity. Connect to the Windows server running in the VPC2 subnet using a Peering connection from Subnet 2 of VPC1.**

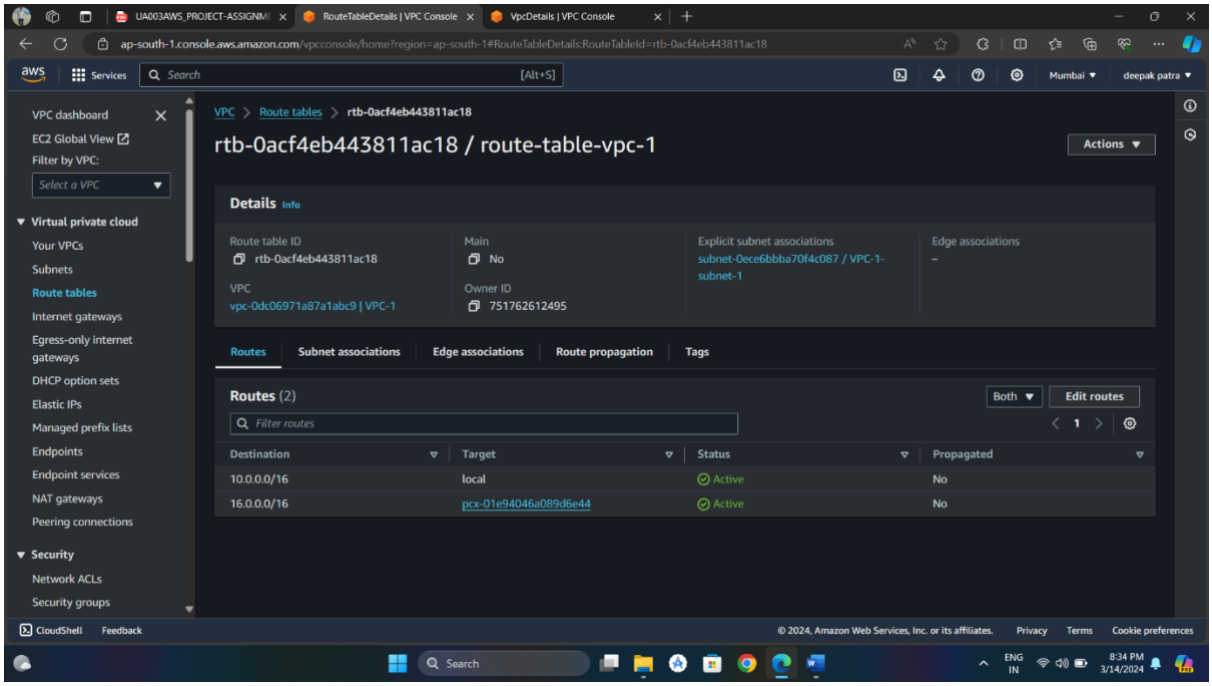
Architecture: -



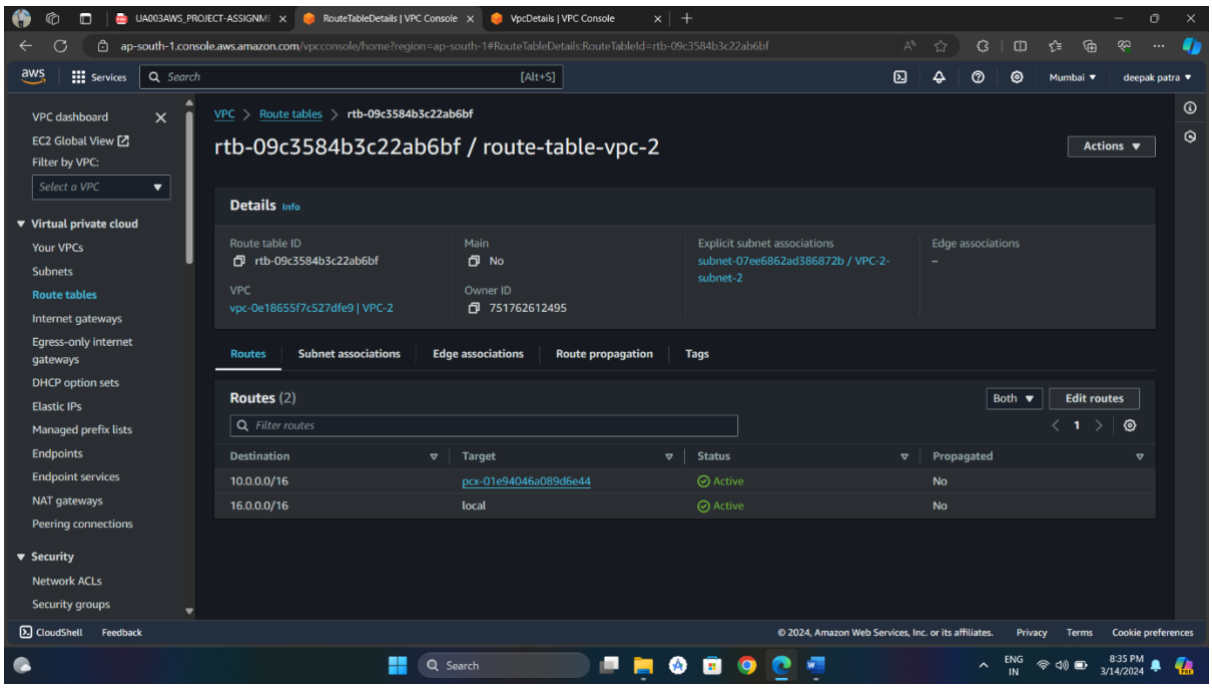
PCX Screen



RTB-1 Screen



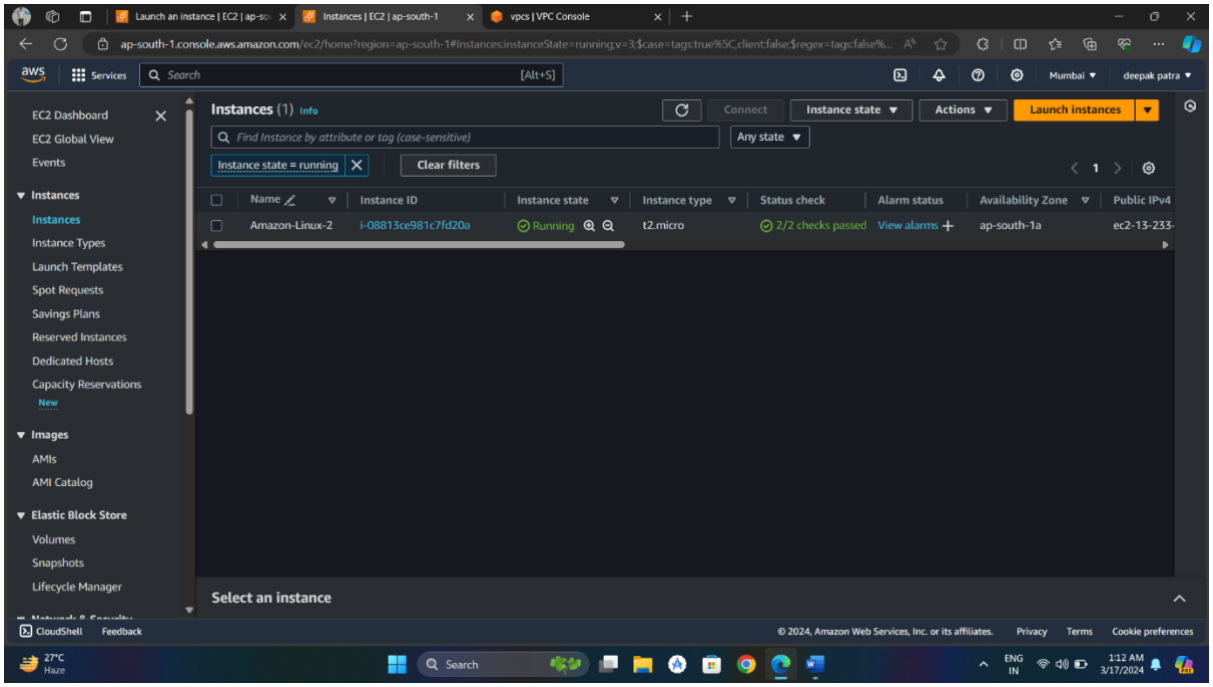
RTB-2-Screen

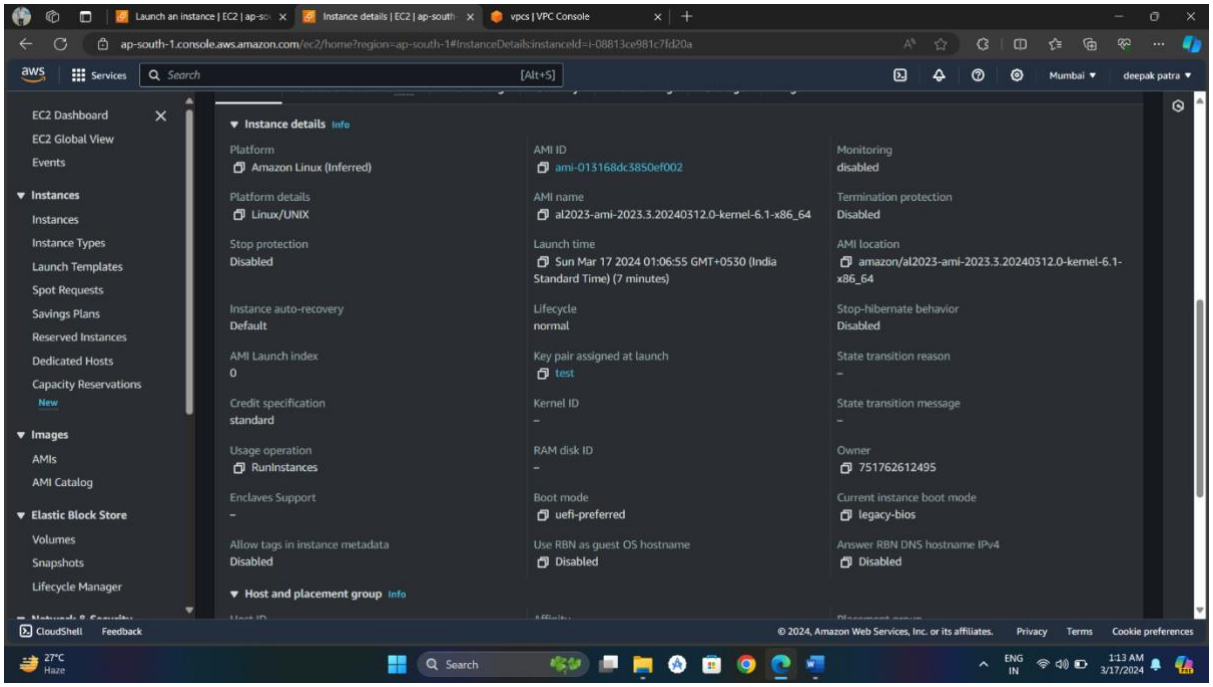


Task: Network Security

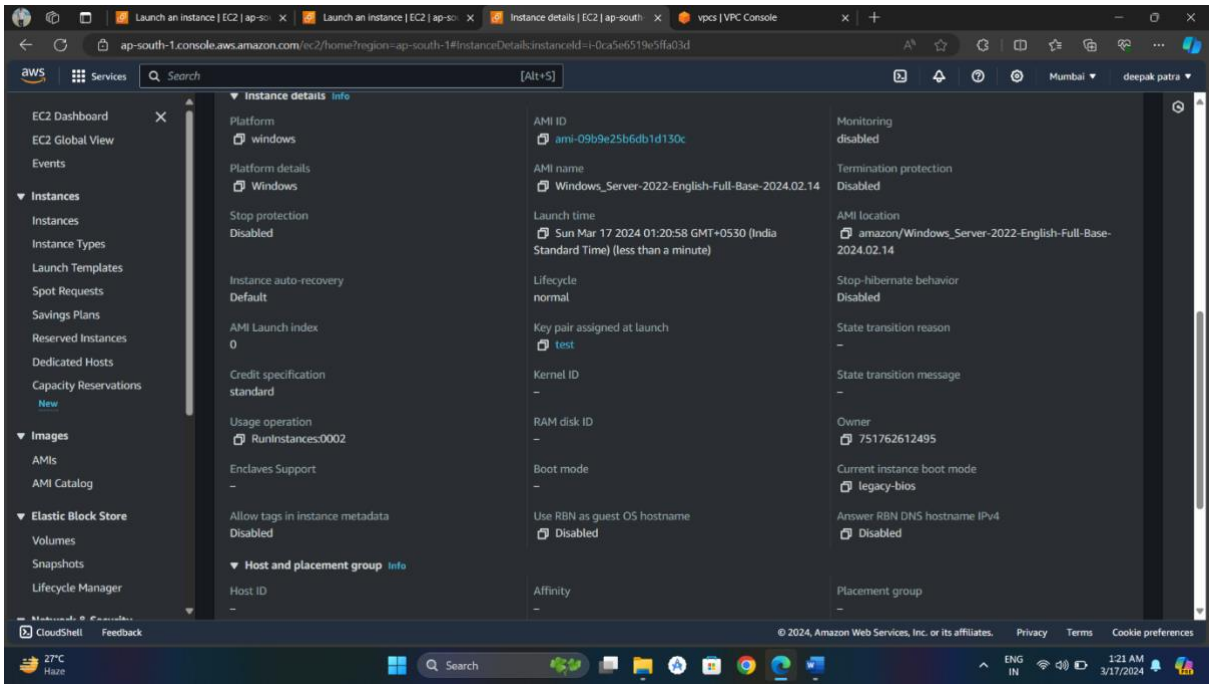
1. **Launch two instances: one with Windows and the other with Amazon Linux 2.**

Linux: -

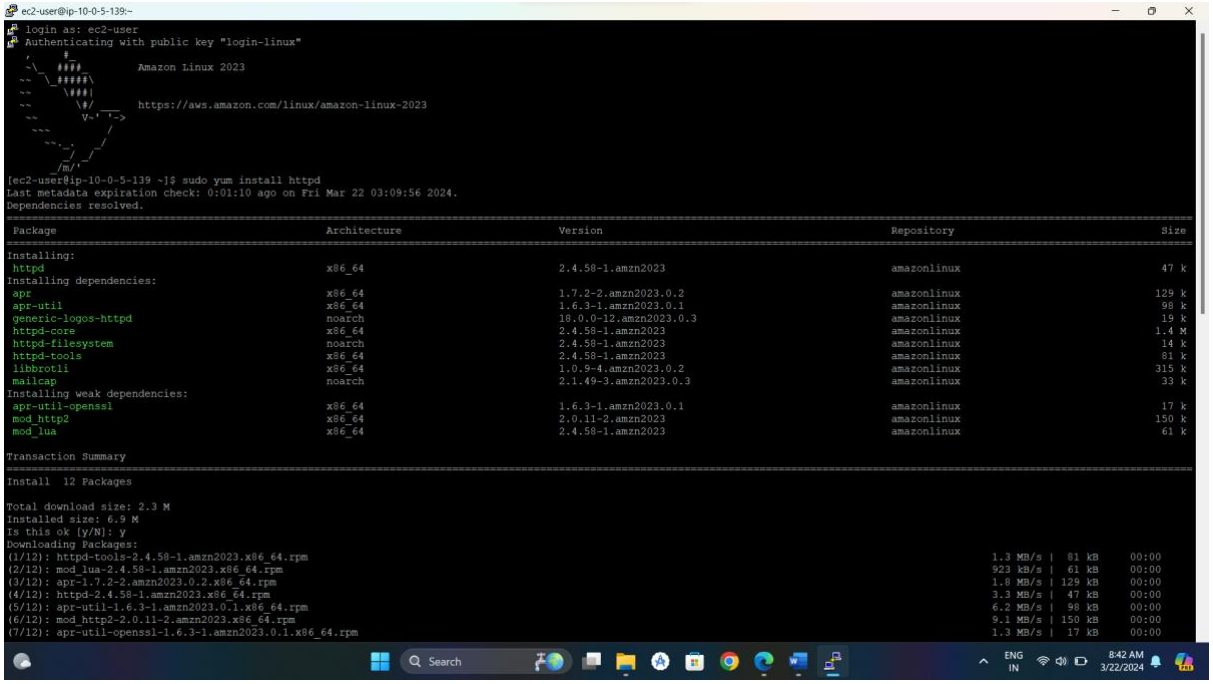




Windows: -



2. Install an httpd web server on the Amazon Linux 2 instance.



3. Log in to the Windows machine and check if the httpd server is running by sending requests to the IP address of the Linux server through a browser and ensure that the proper ports are opened in the security groups.

Way to follow: -

1. Log in to the Windows Machine:

- Access the AWS Management Console.
- Navigate to the EC2 service.
- Locate and select the Windows EC2 instance you want to log in to.
- Click on the "Connect" button.
- Follow the instructions provided to connect to the Windows instance using Remote Desktop Protocol (RDP).

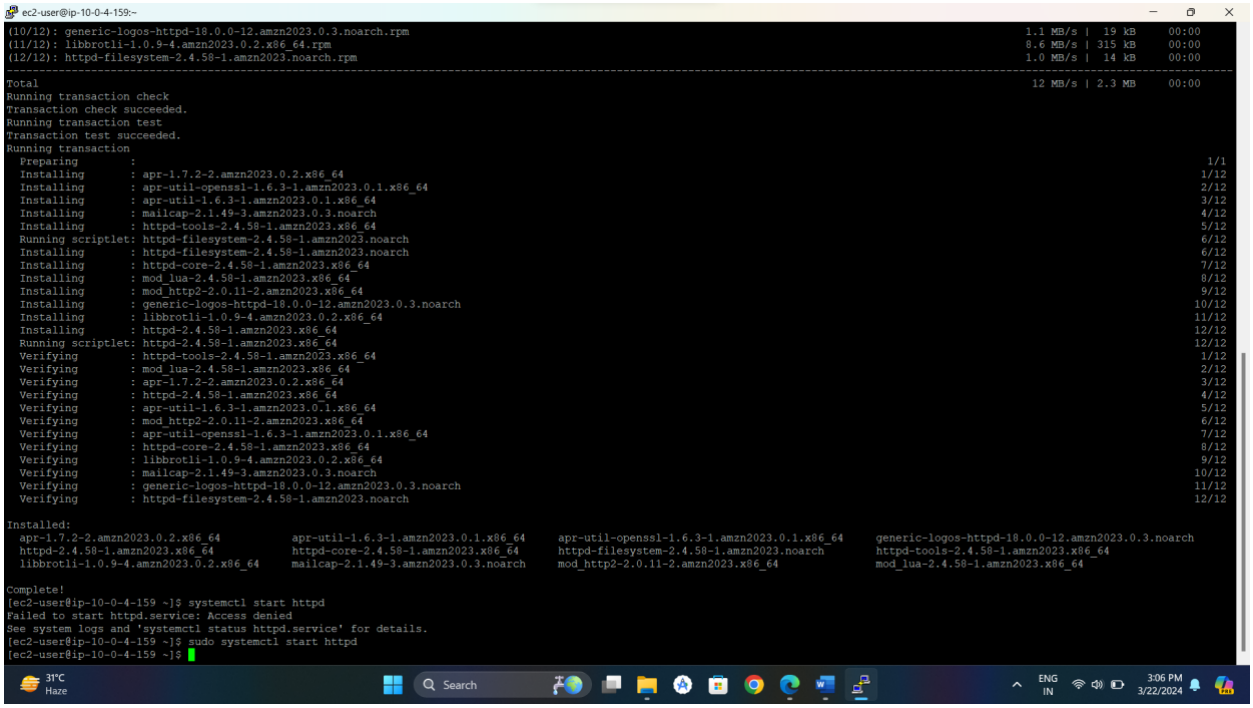
2. Check if the HTTPD Server is Running on the Linux Server request to windows:

- Once logged in to the Windows instance, open a web browser (e.g., Internet Explorer, Google Chrome).
- Enter the IP address of the Linux server in the address bar of the browser.
- If the HTTPD server is running on the Linux server and accessible from the Windows machine, you should see the default Apache web page or any other content hosted by the server.

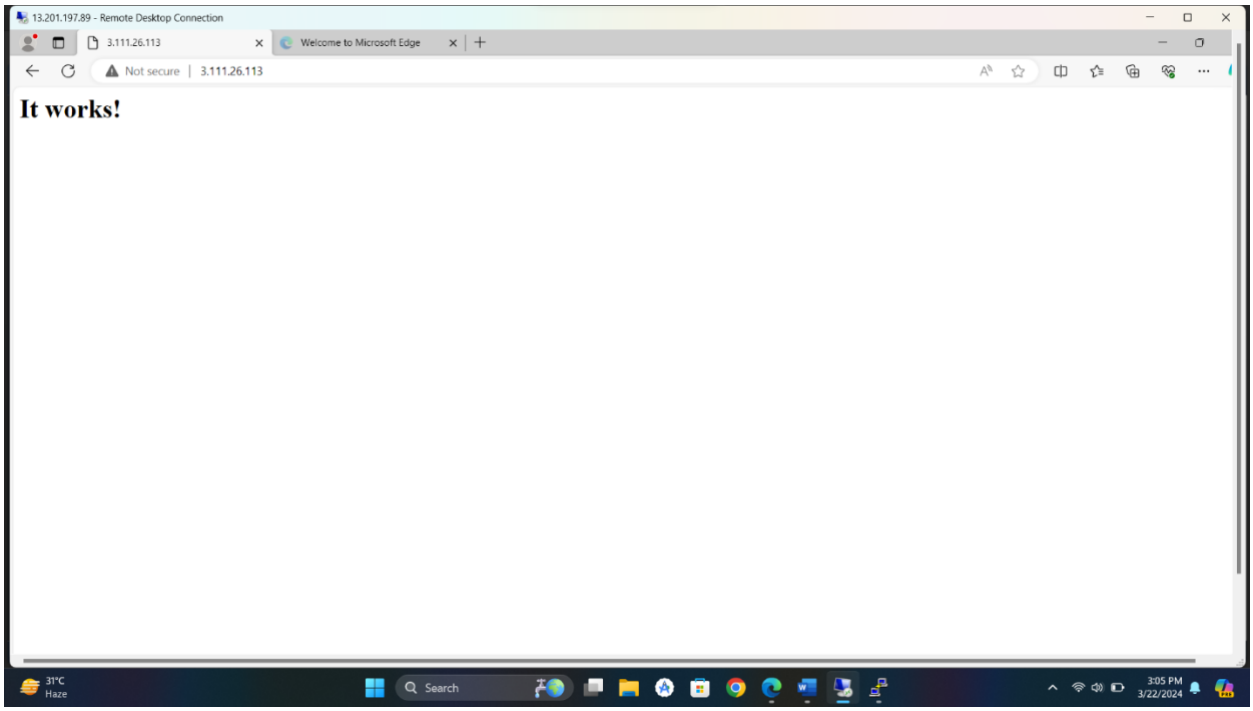
3. Ensure Proper Ports are Opened in Security Groups:

- Go to the AWS Management Console.
- Navigate to the EC2 service.
- Click on "Security Groups" in the navigation pane.
- Find and select the security group associated with your Linux EC2 instance.
- Click on the "Inbound Rules" tab.
- Ensure that port 80 (HTTP) is open for incoming traffic from the Windows instance's IP address or from any source (0.0.0.0/0) if you want to allow access from any location.
- If port 80 is not open, click on "Edit inbound rules" and add a new rule allowing HTTP traffic (port 80) from the desired source.

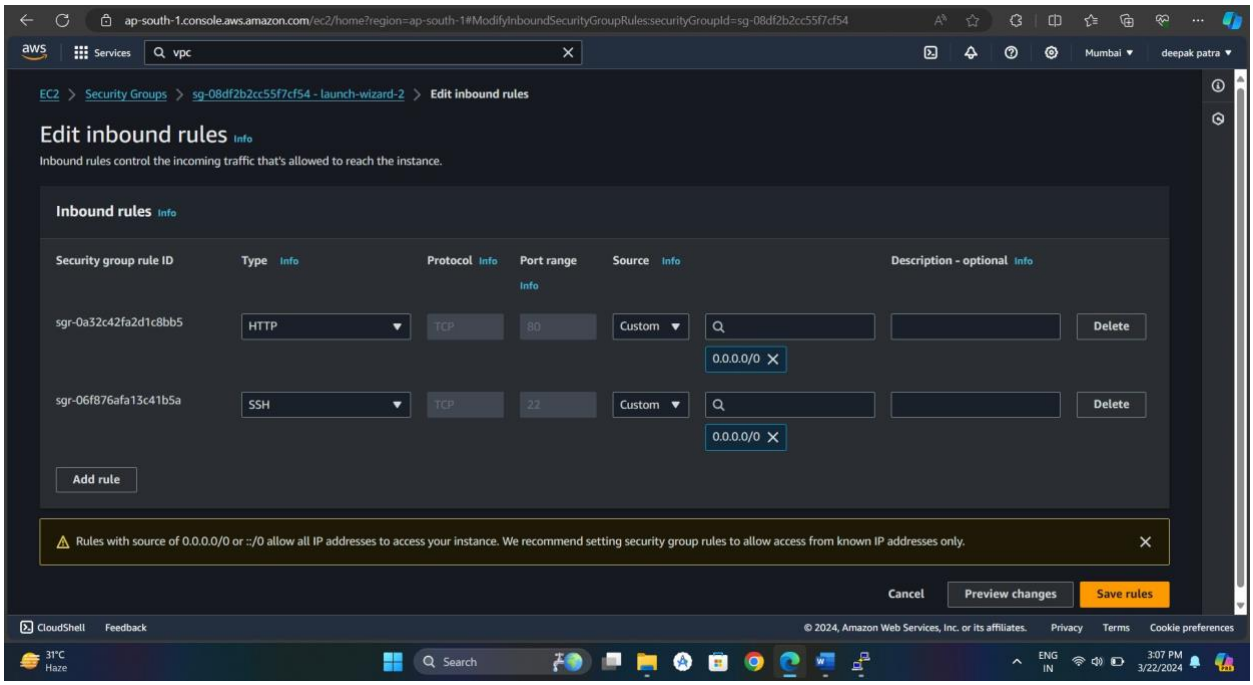
Httpd server installed on Linux



Httpd server successfully sending requests to Windows Browser.



Proper port 80 are opened on Amazon Linux Instances



Task: EC2, Snapshot

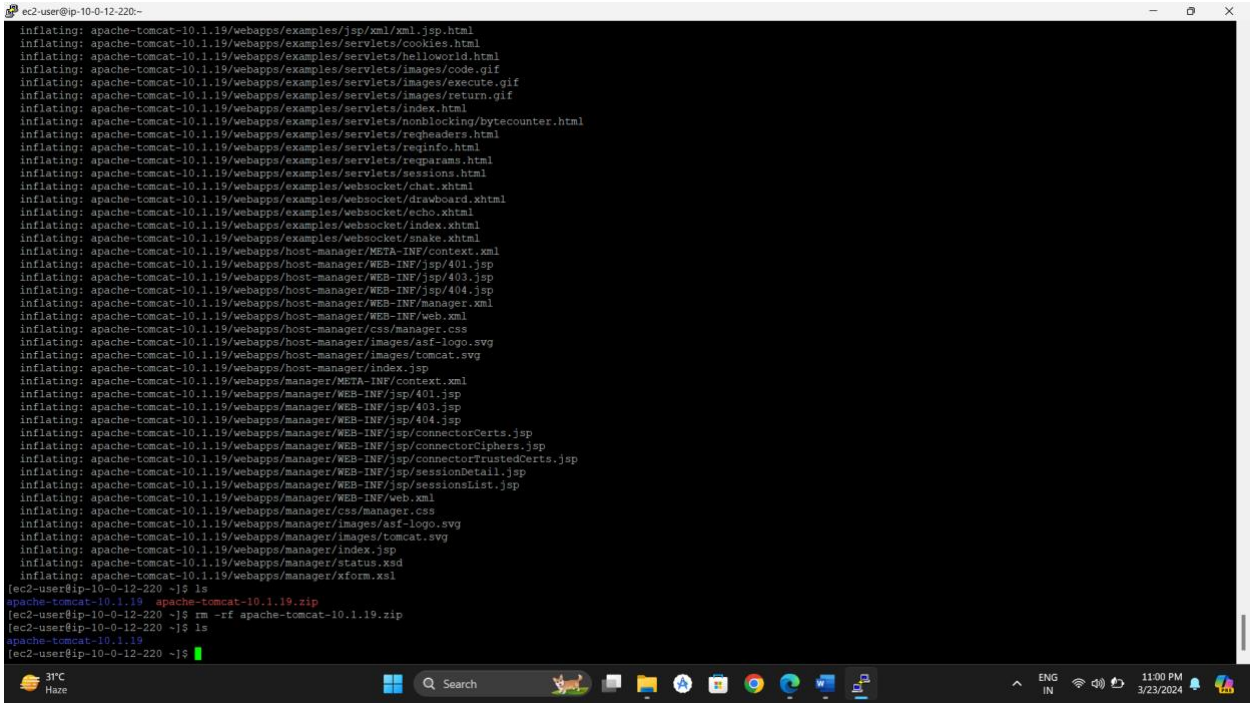
1. Launch an EC2 instance and install the Tomcat server on it.

Path to be followed:

- Log in to the AWS Management Console.
- Navigate to the EC2 dashboard.
- Click on "Launch Instance".
- Choose an Amazon Machine Image (AMI) based on your requirements (e.g., Amazon Linux, Ubuntu, etc.).
- Select an instance type.
- Configure instance details (e.g., network, subnet, security group).
- Add storage as needed.
- Add tags if required.

- Configure security groups to allow inbound traffic on port 8080 (default Tomcat port).
- Review and launch the instance.
- Connect to your instance using SSH.
- Install Tomcat using appropriate package management commands based on the chosen Linux distribution.
- Use wget command with Apache Tomcats Zip link for downloading Tomcat.
- Yum install java -y (install java because building Tomcat it requires JDK).

Apache Tomcat

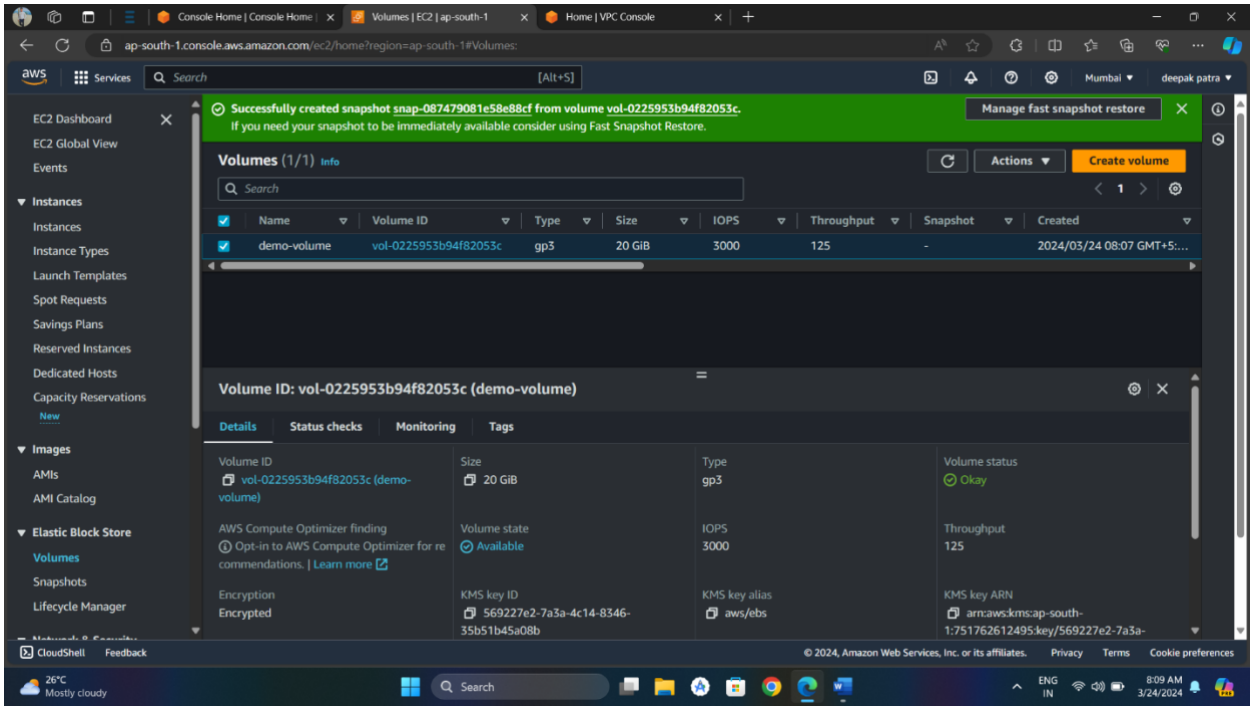


2. Create a volume for the instance and take a snapshot.

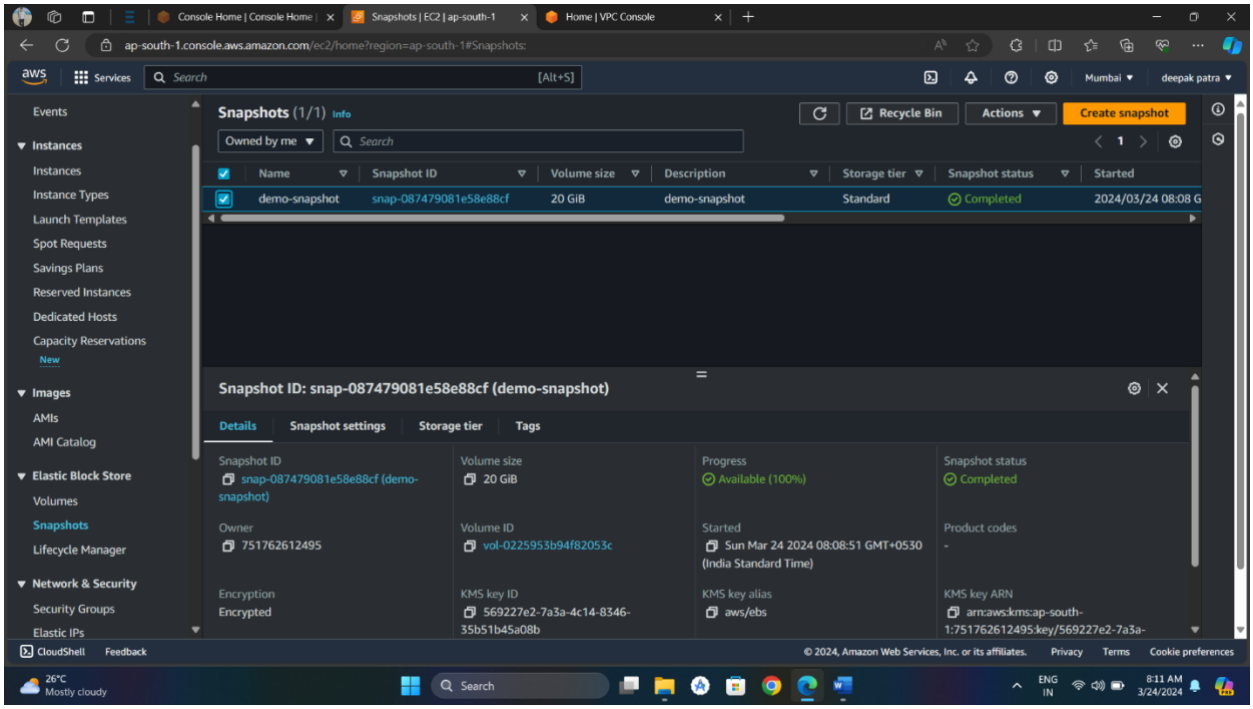
Path to be followed:

- Navigate to the EC2 dashboard.
- Under "Elastic Block Store", click on "Volumes".
- Click on "Create Volume".
- Configure the volume settings (size, availability zone).
- Once created, select the volume, and click on "Actions" -> "Create Snapshot".
- Give the snapshot a name and description.
- Confirm and create the snapshot.

Volume



Snapshot



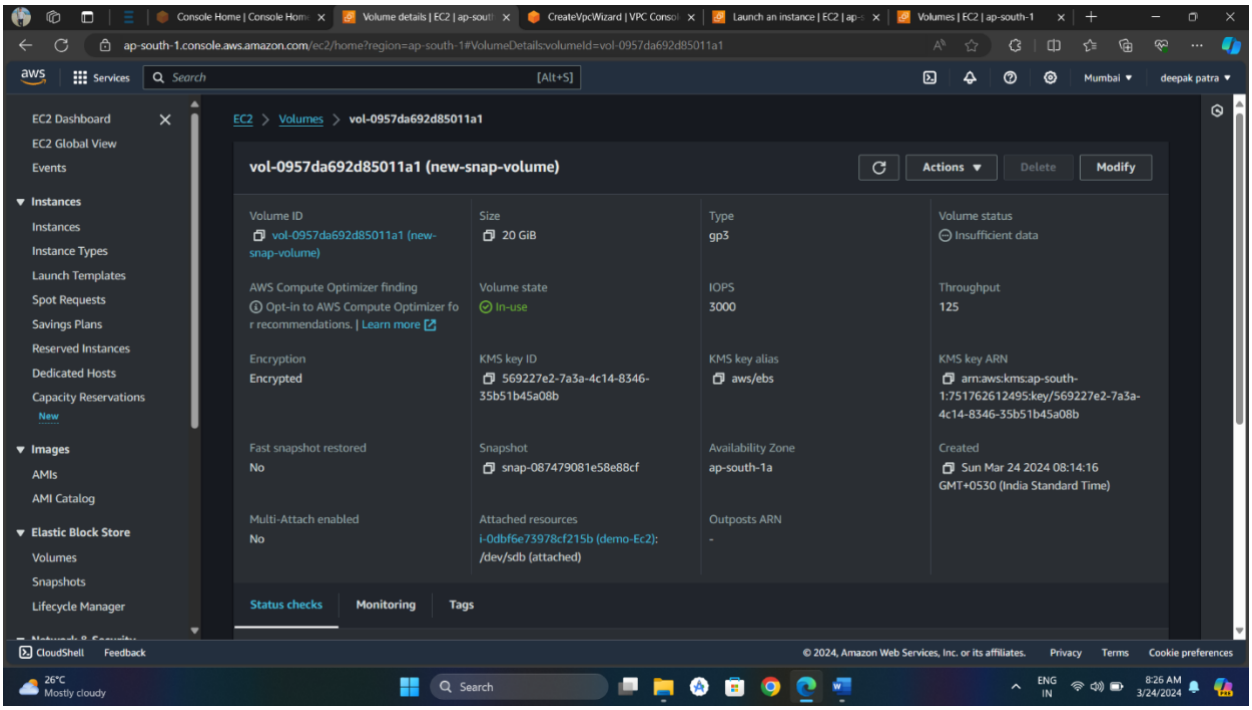
3. Create and attach the volume through the snapshot to another instance launched in a different availability zone but within the same security group.

Path to be followed:

- Navigate to the EC2 dashboard.
- Under "Elastic Block Store", click on "Snapshots".
- Select the snapshot you created in the previous step.
- Click on "Actions" -> "Create Volume".
- Configure the volume settings (e.g., size, availability zone).
- Once created, select the new volume and click on "Actions" -> "Attach Volume".
- Choose the instance to attach the volume to.

- SSH into the instance and mount the attached volume.(doubt)

Volume Attached

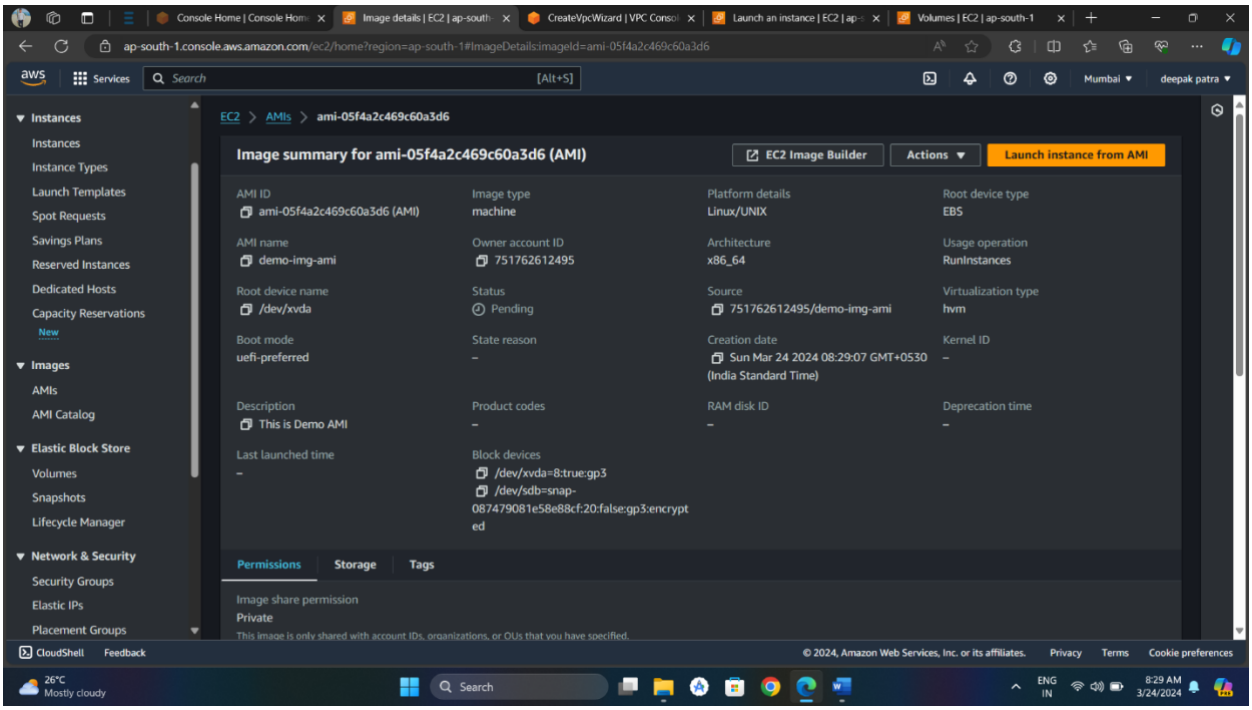


4. Create an AMI backup of that instance.

Path to be followed:

- Navigate to the EC2 dashboard.
- Select the instance you want to create an AMI backup of.
- Click on "Actions" -> "Image and templates" -> "Create image".
- Provide a name and description for the AMI.
- Configure any additional settings if needed.
- Confirm and create the AMI backup.

AMI



EBS Storage Problem Statement:

Scenario: You work for XYZ Corporation, and your corporation wants to launch a new web-based application using AWS Virtual Machines. Configure the resources accordingly with appropriate storage for the tasks.

Answer:

To configure AWS resources for launching a new web-based application for XYZ Corporation using Virtual Machines, we'll need to consider various components such as Virtual Machines, storage, networking, and security. Below is a basic setup to get started:

1. EC2 Instances (Virtual Machines):

- First, I Determine the appropriate EC2 instance types based on the application's requirements (e.g., CPU, memory, network performance).
- Then I Choose the suitable operating system for the instances (e.g., Amazon Linux, Ubuntu, Windows).
- Then I Configure the number of instances required for scalability and redundancy.
- I will definitely Ensure that the instances are launched in the customers desired AWS region for optimal performance and compliance.

Storage:

- I Utilize Amazon Elastic Block Store (EBS) for block storage that can be attached to EC2 instances.
- Then I will Determine the size and type (e.g., General Purpose SSD, Provisioned IOPS SSD) of EBS volumes based on the application's storage requirements and performance needs.
- I will Consider utilizing Amazon Elastic File System (EFS) for scalable and elastic file storage if the application requires shared file storage across multiple instances.
- I need to Set up snapshots for EBS volumes to enable backup and disaster recovery capabilities.

Networking:

- I need to Configure Virtual Private Cloud (VPC) to launch EC2 instances in an isolated network environment.
- I Define subnets within the VPC to logically segment resources.
- -Then I Set up security groups to control inbound and outbound traffic to EC2 instances.
- Then I Configure Network Access Control Lists (NACLs) for additional network-level security.
- Then I Assign Elastic IP addresses (EIPs) to EC2 instances for static public IP addresses if needed.
- Then I Implement AWS CloudFront for content delivery and caching to improve latency and global reach.

Security:

- I Utilize AWS Identity and Access Management (IAM) to manage user access and permissions.

Monitoring and Logging:

- I need to Set up Amazon CloudWatch to monitor EC2 instances, EBS volumes, and other AWS services for metrics and logs.
- Then I Configure alarms and notifications for monitoring key performance indicators and thresholds.
- Then I Utilize AWS CloudTrail for auditing the AWS resources.

High Availability and Scalability:

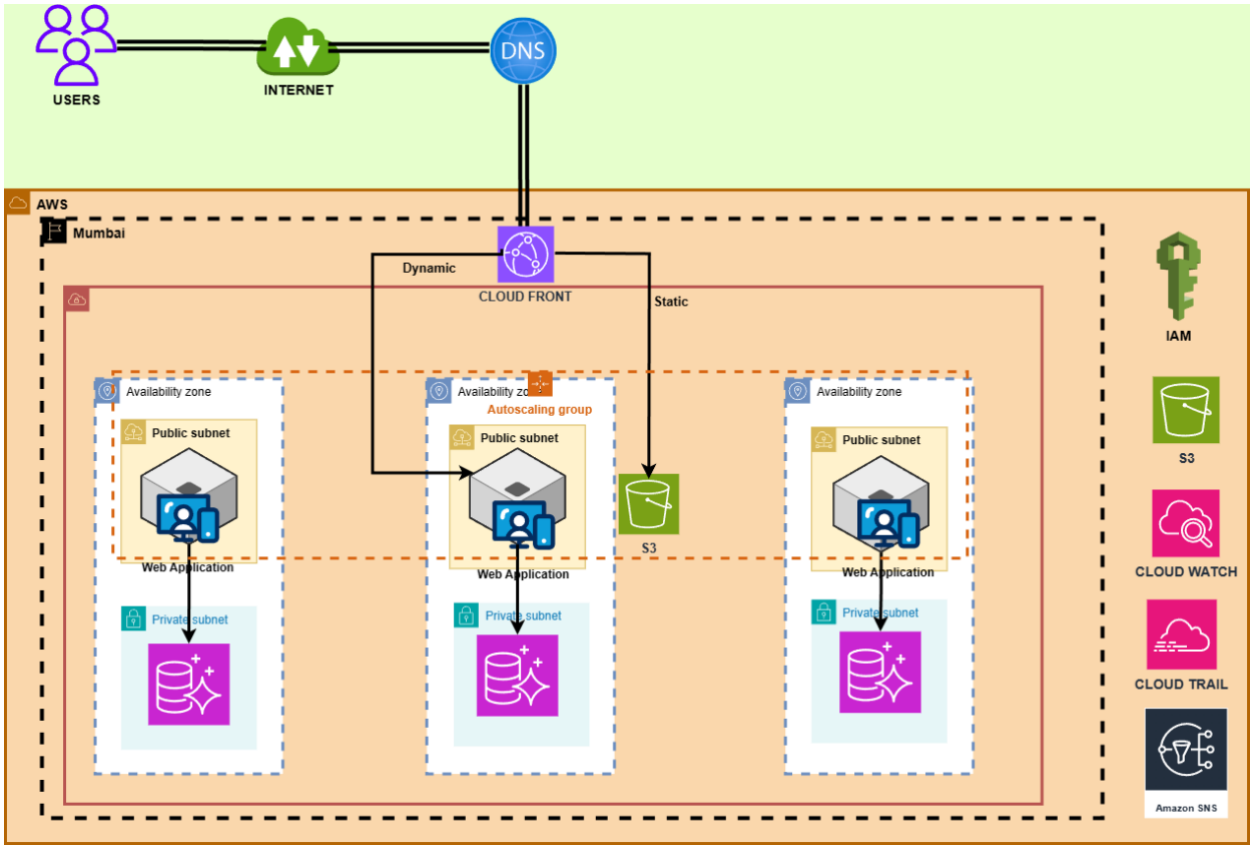
- I Utilize Auto Scaling groups to automatically adjust the number of EC2 instances based on demand.
- Then I Configure Elastic Load Balancing (ELB) to distribute incoming traffic across multiple EC2 instances for high availability and fault tolerance.
- I need to Utilize AWS Route 53 for DNS management and to route traffic to the application's endpoints.

Additionally, thing I will do:

- I Enable WAF (web Application Firewall). Why Because I will disallow some specific ip address as per my requirement.
- I Use ACM (Amazon Certificate Manager). Why Because when request comes from HTTP, I will redirect it to HTTPS port 443.
- I Use Route 53 for define multiple Route Policy's like: - simple Route, Weighted Route, Failover Route, Geolocation Route.

By configuring these AWS resources appropriately, XYZ Corporation can launch their new web-based application with the necessary scalability, reliability, and security on AWS Virtual Machines.

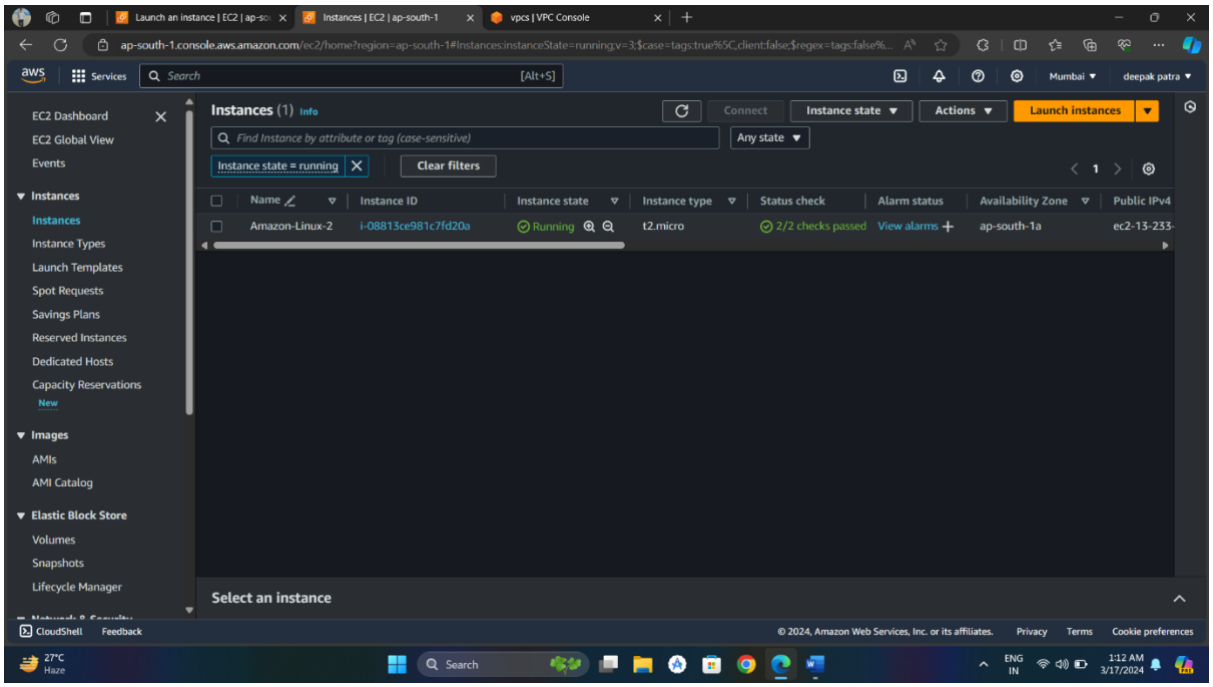
Architecture: -



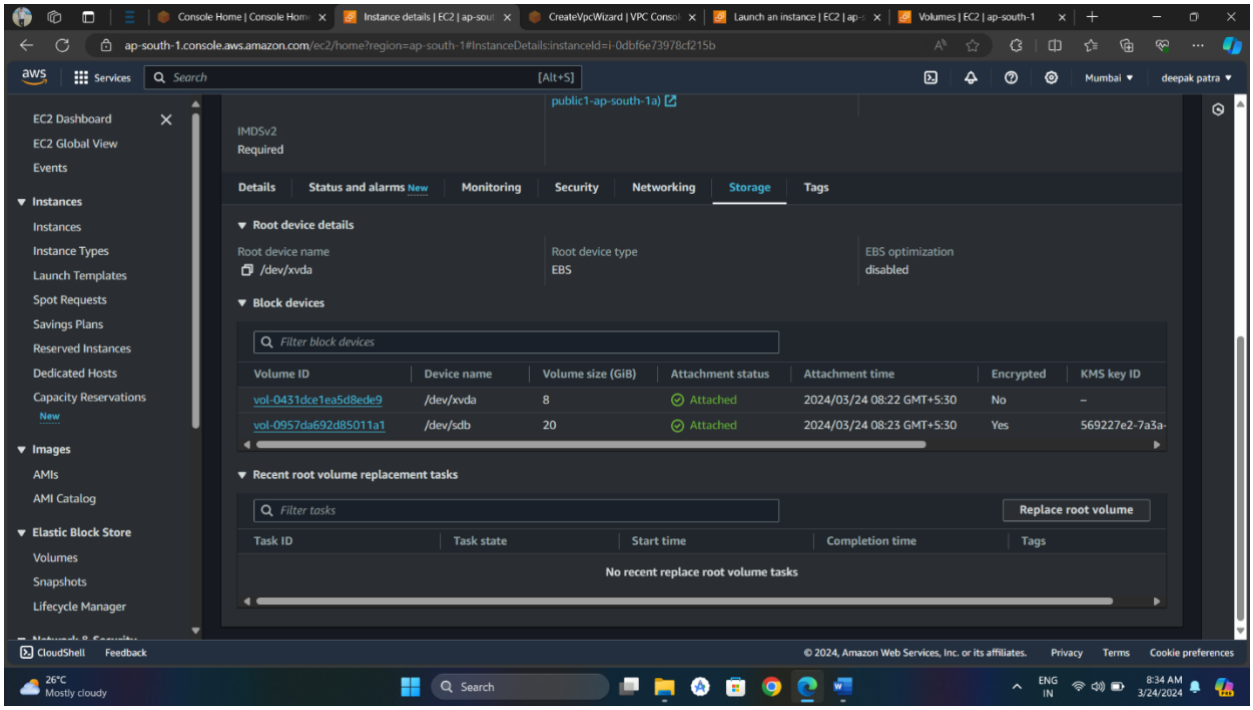
Demo-Screenshots

Tasks To Be Performed:

1. Launch a Linux EC2 instance.



2. Create an EBS volume with 20 GB of storage and attach it to the created EC2 instance.

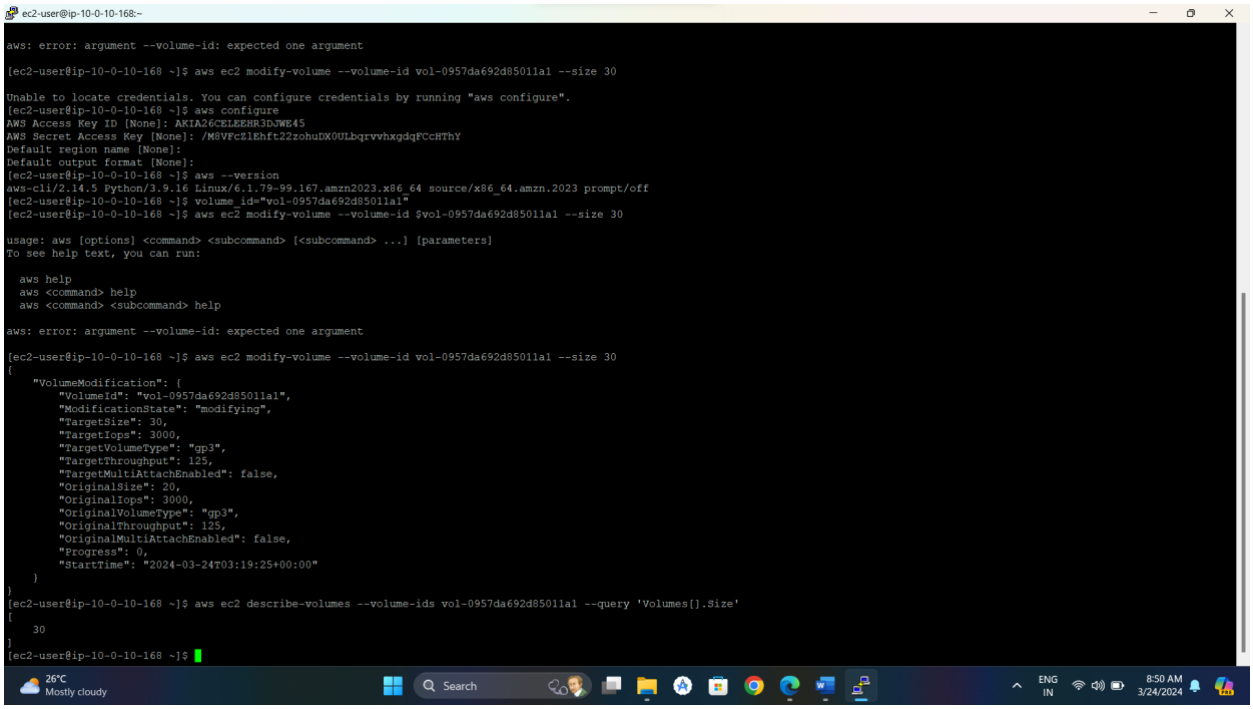


3. Resize the attached volume from 20 GB to 30 GB and ensure it reflects in the connected instance.

Path to be followed

- Configure AWS CLI using Access key. (AWS Configure)
- aws ec2 modify-volume --volume-id $volume_id --size 30 (fire this command).
- aws ec2 describe-volumes --volume-ids $volume_id --query 'Volumes[].Size' (fire this for checking size).

EBS volume Change by CLI using Access key

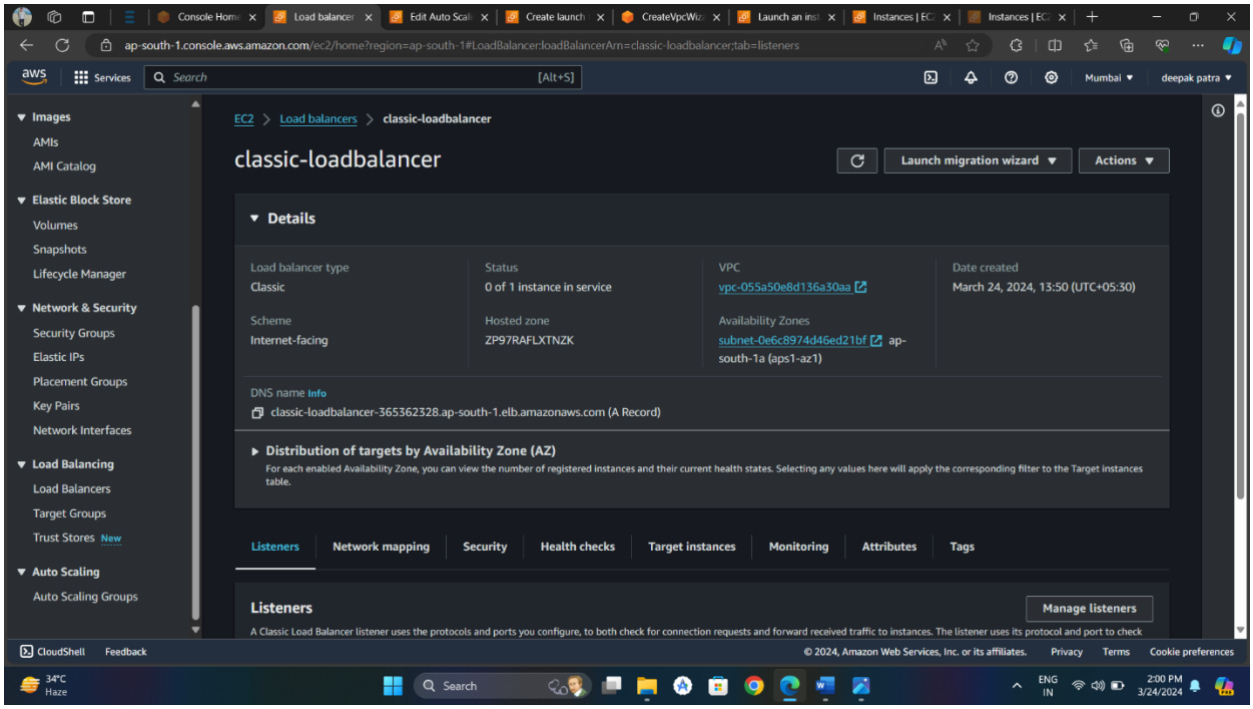


Problem Statements:

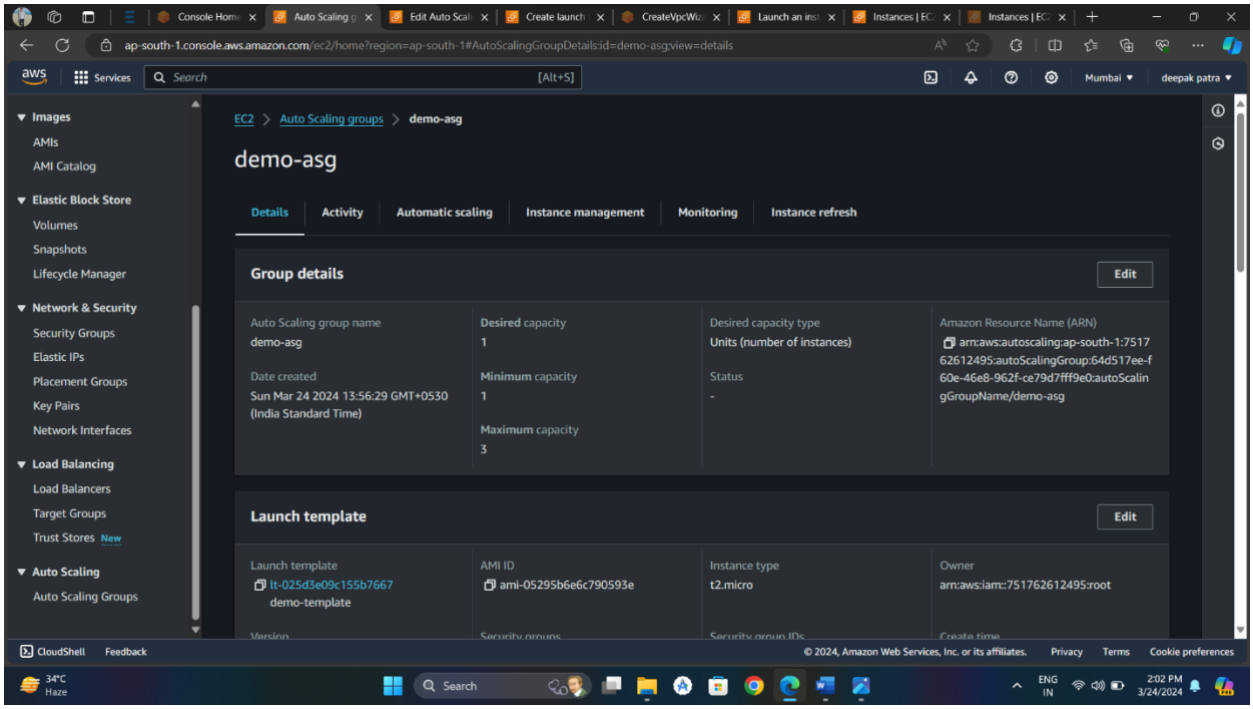
Task 1:

1. Create a Classic Load Balancer (CLB) and attach it to an Auto Scaling Group (ASG).

Load-Balancer

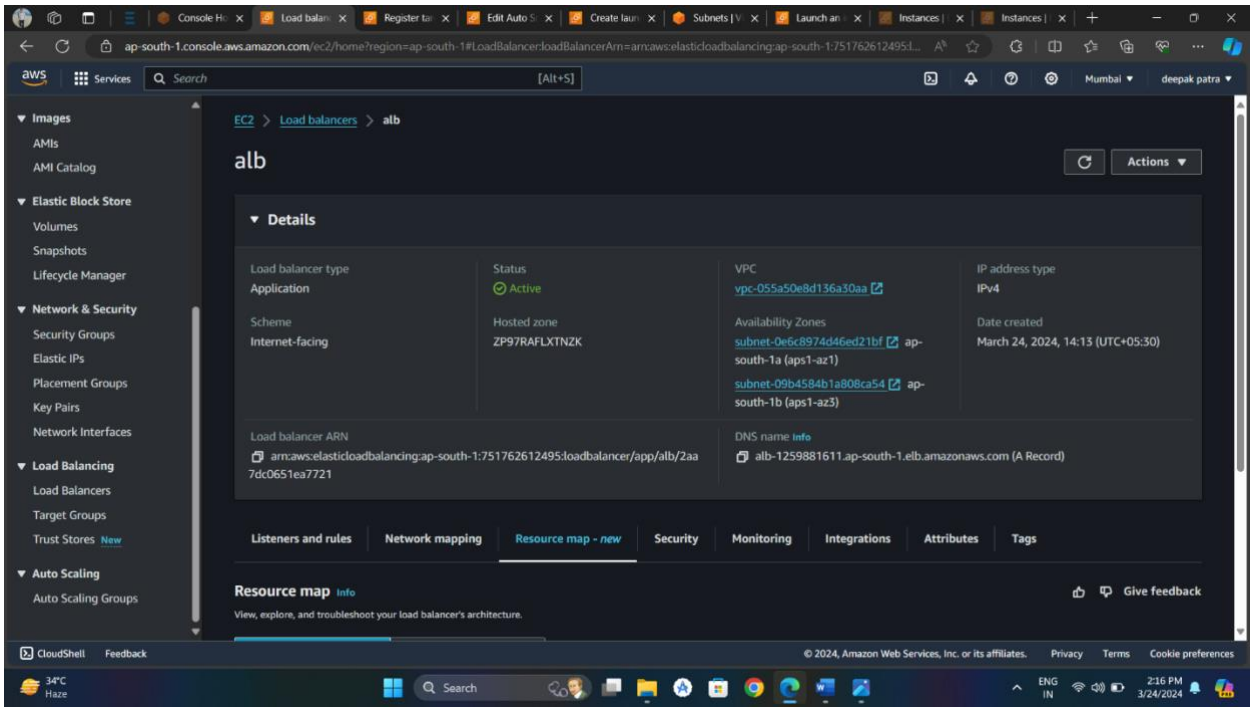


Auto-Scaling-Group

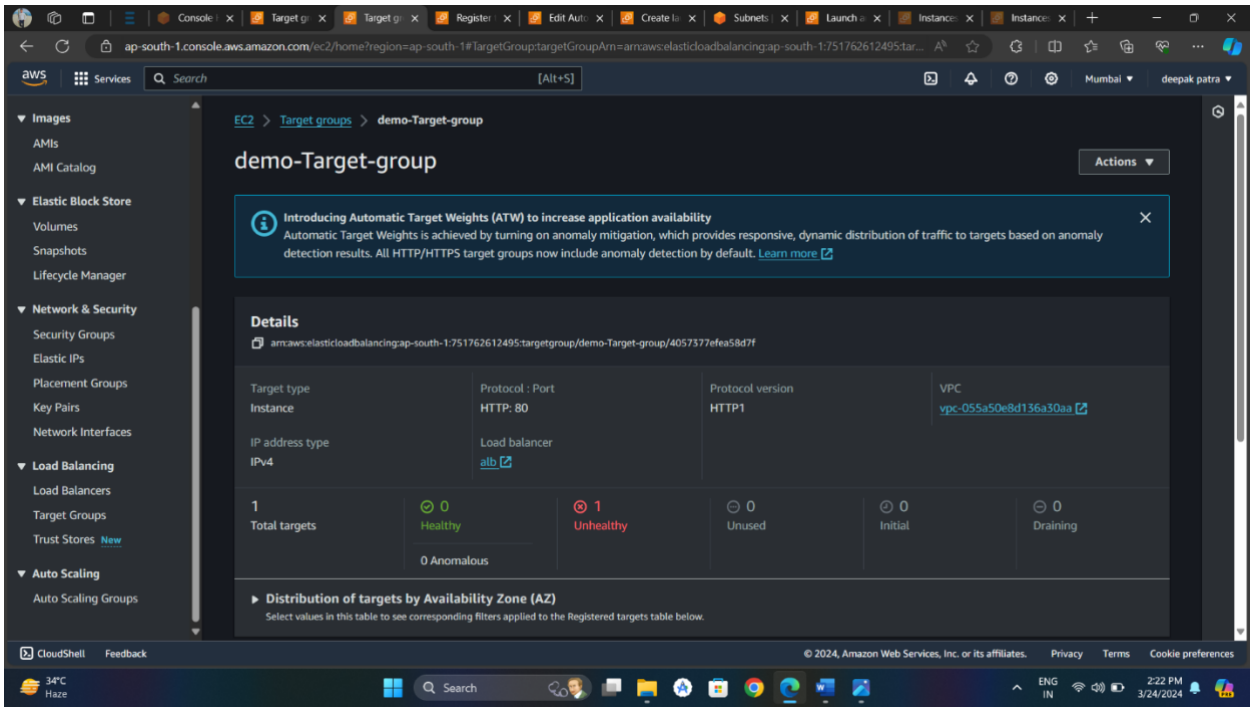


Task 2:

1. Create an Application Load Balancer (ALB).

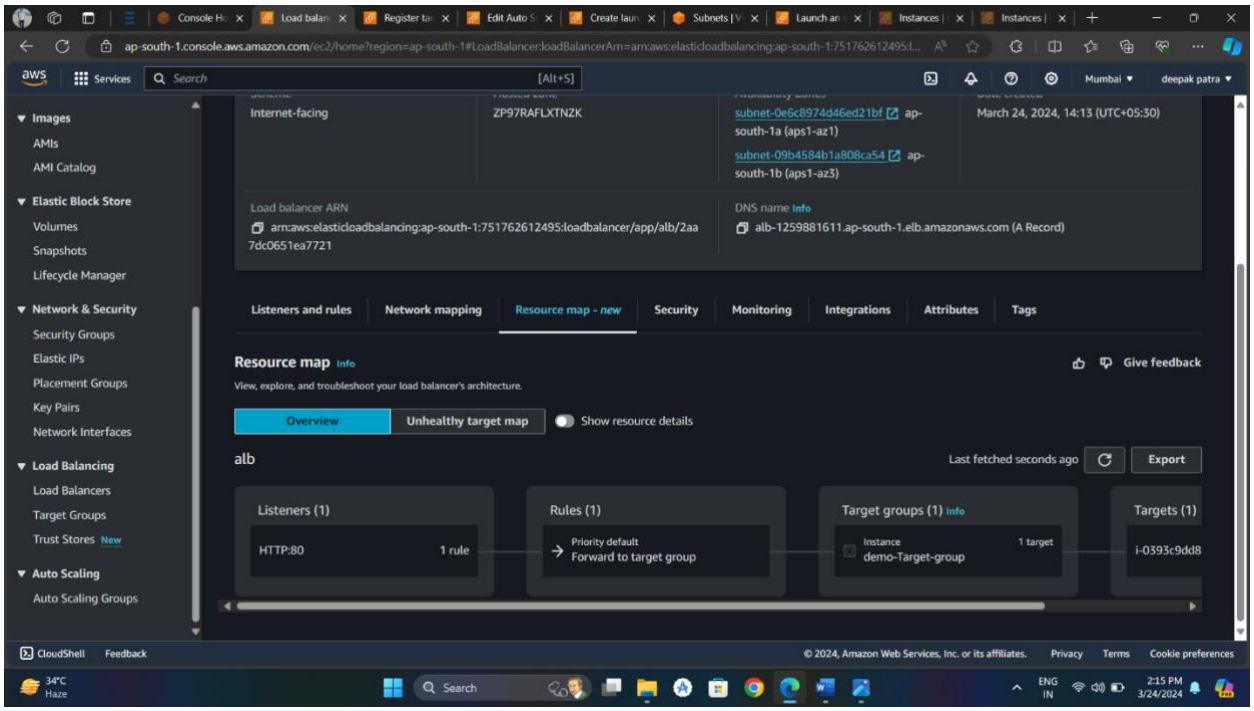


2. Attach Target Groups to the ALB.



3.Traffic should be served through the ALB and if you are purchasing any domain then point it to ALB DNS.

Ans: Currently I have not purchased any DNS.



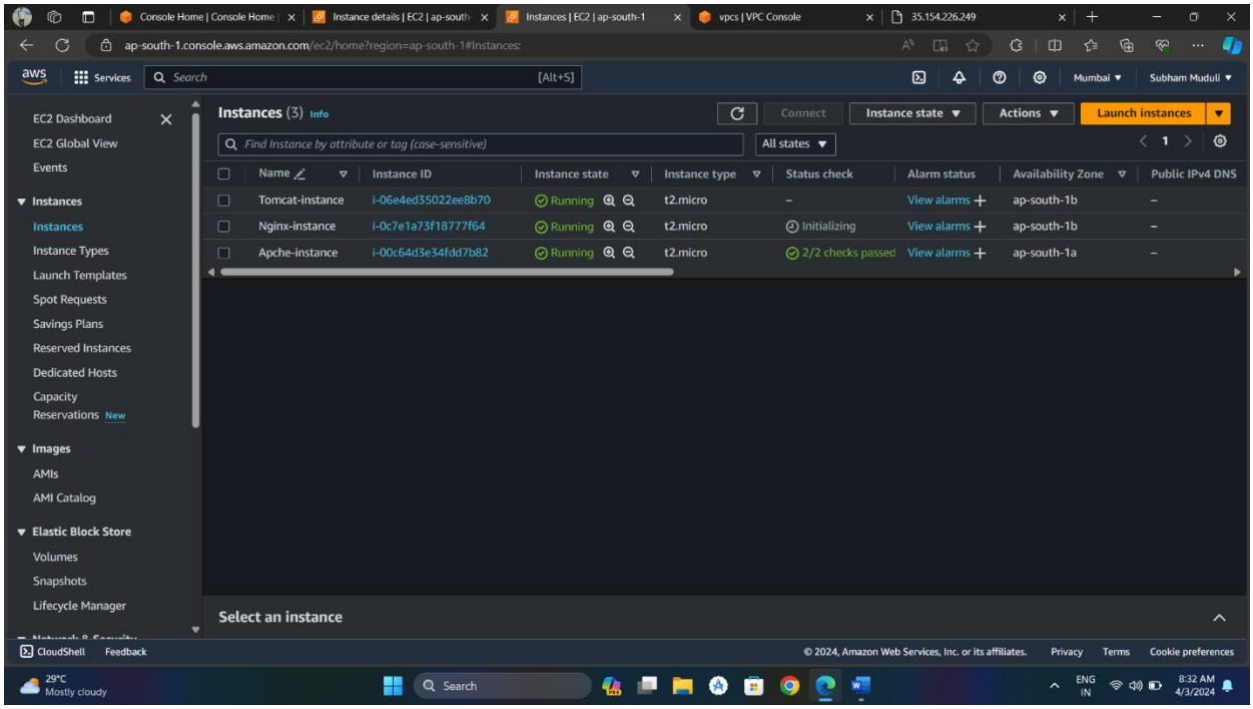
Problem Statement:

Task 1:

Set up a Network Load Balancer for an Apache web server where the load balancer port should be 25. And configure Nginx Web Server on load balancer port 58. Additionally, set up a Tomcat App server on load balancer port 90.

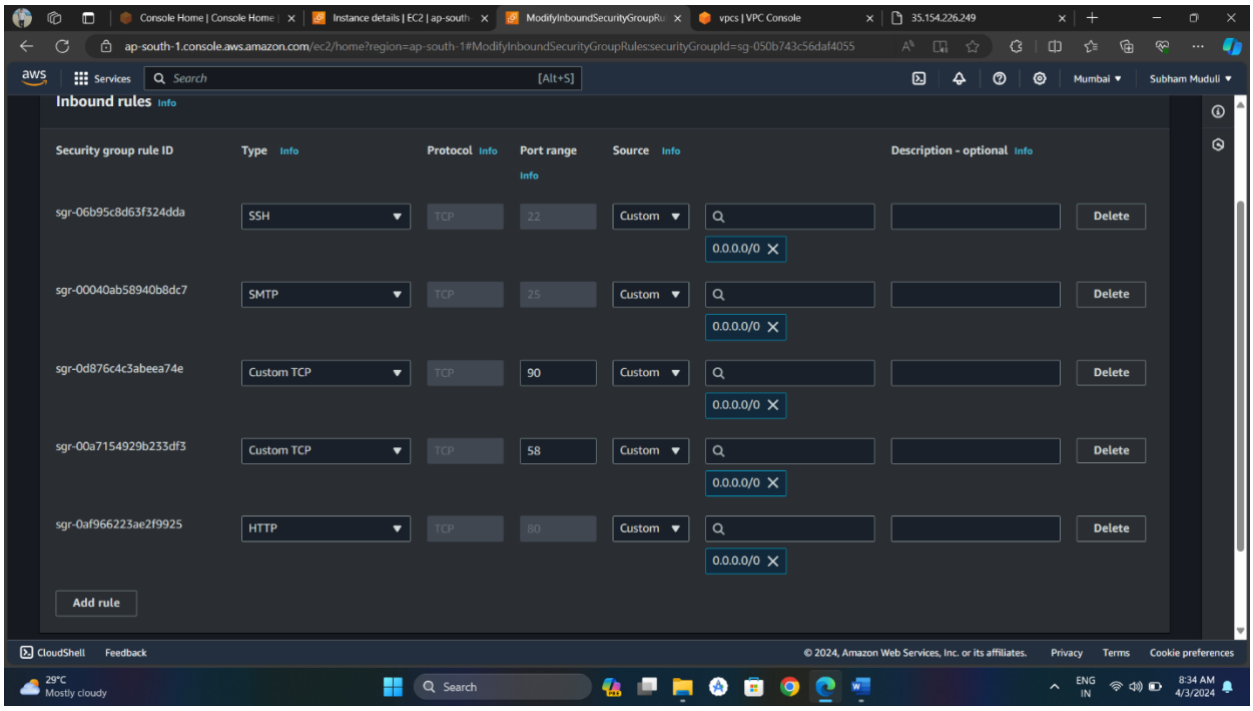
1. Create EC2 Instances:

- **Launch EC2 instances for Apache, Nginx, and Tomcat servers, ensuring they are configured properly with the required software.**



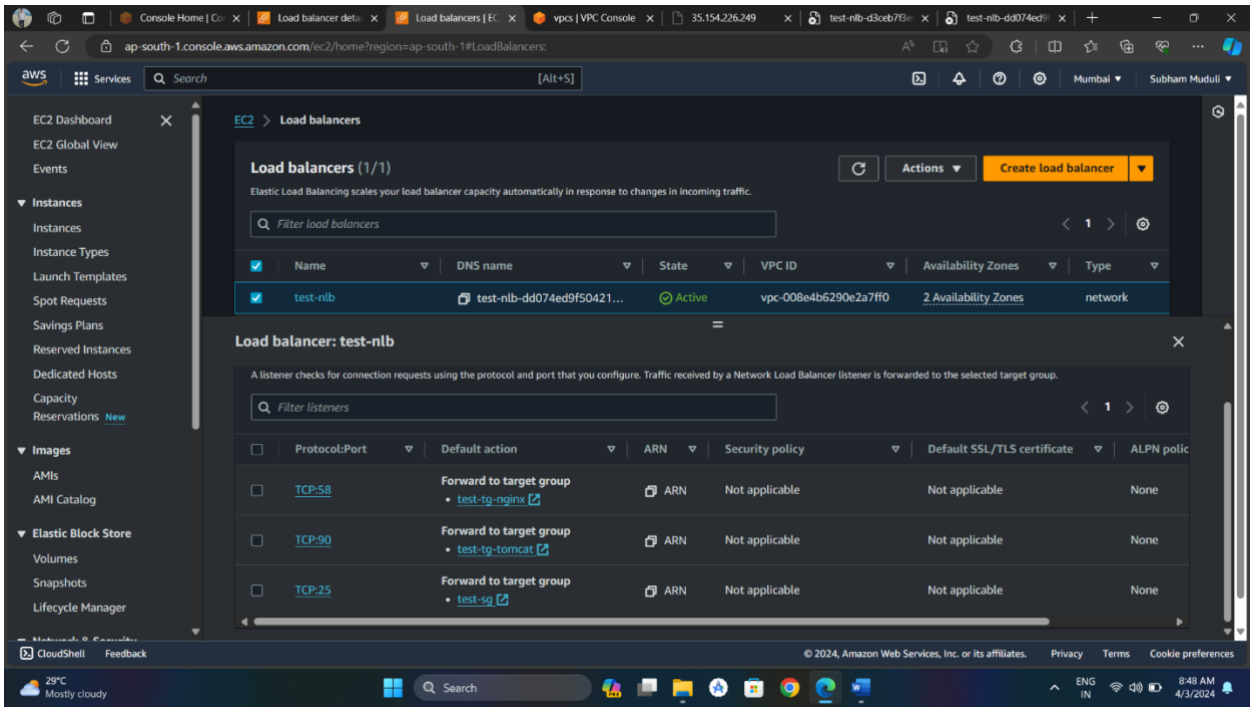
2. Configure Security Groups:

- Create a security group that allows inbound traffic on ports 25, 58, and 90 from the load balancer to the respective instances.
- Attach this security group to all the EC2 instances.



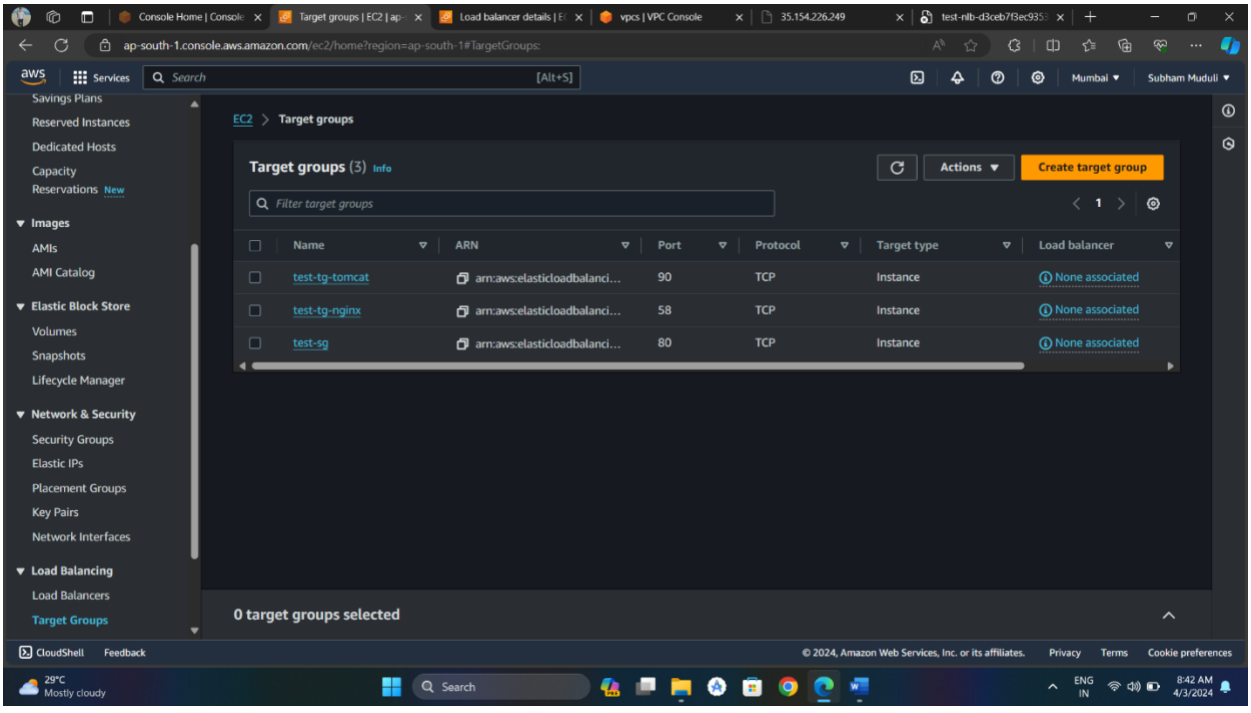
3. Set Up the Network Load Balancer:

- Navigate to the EC2 dashboard and select "Load Balancers" from the sidebar.
- Click on "Create Load Balancer" and choose "Network Load Balancer".
- Configure the NLB settings, such as name, scheme, IP address type, and listeners:
- For port 25, set the listener protocol to TCP and port to 25, then configure the target group for Apache.
- For port 58, set the listener protocol to TCP and port to 58, then configure the target group for Nginx.
- For port 90, set the listener protocol to TCP and port to 90, then configure the target group for Tomcat.
- Configure the availability zones where your EC2 instances are located.
- Configure security settings and tags as needed.
- Create the NLB.



4. Create Target Groups:

- Navigate to the EC2 dashboard and select "Target Groups" from the sidebar.
- Click on "Create target group" and configure the target group settings:
- For Apache, set the target type to instance and choose the instances running Apache.
- For Nginx, set the target type to instance and choose the instances running Nginx.
- For Tomcat, set the target type to instance and choose the instances running Tomcat.
- Create the target groups.

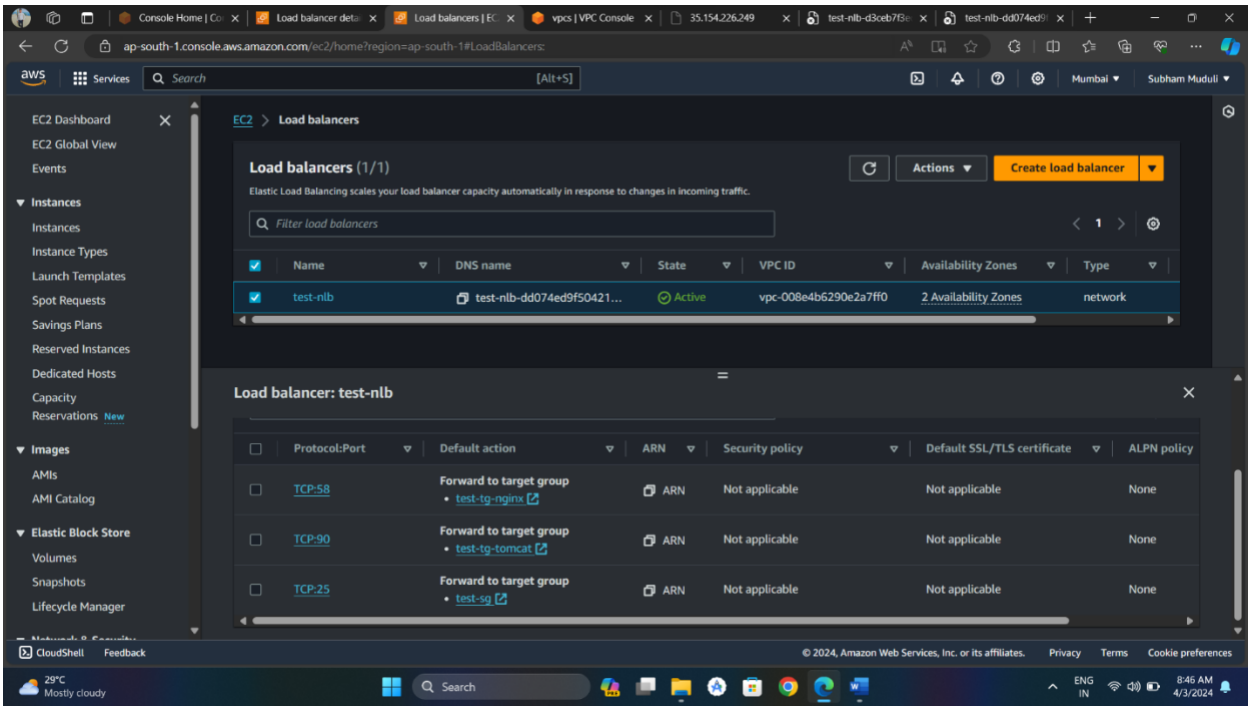


5. Register Targets:

- After creating the target groups, register the corresponding EC2 instances (Apache, Nginx, and Tomcat servers) with their respective target groups.

6. Update Load Balancer Listener Rules:

- Once the target groups are set up, update the load balancer listener rules to forward traffic on ports 25, 58, and 90 to their respective target groups.



7. Test the Configuration:

- Verify that the NLB distributes traffic correctly to the Apache, Nginx, and Tomcat servers by accessing the NLB endpoint with the specified ports.
- for testing that NLB route traffic perfectly to each instance I go to config files of all web-server's and updated Listening port of all web-servers.

Assignment:

1. Create a private Route 53 domain and point it to an Application Load Balancer.

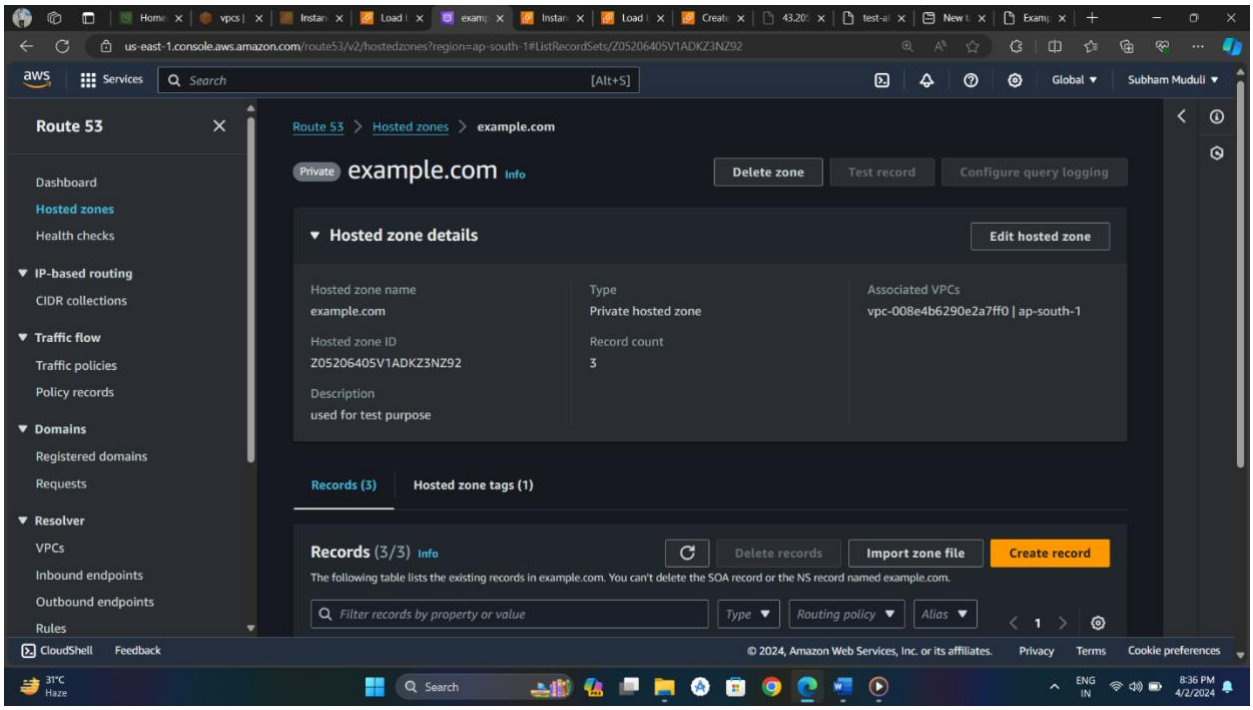
Complete path:

- Go to the Route 53 console.
- Choose "Create Hosted Zone."
- Enter the domain name for your private zone (e.g., example.com).
- Choose the type "Private hosted zone for Amazon VPC."
- Select the VPC(s) where you want to associate this private hosted zone.

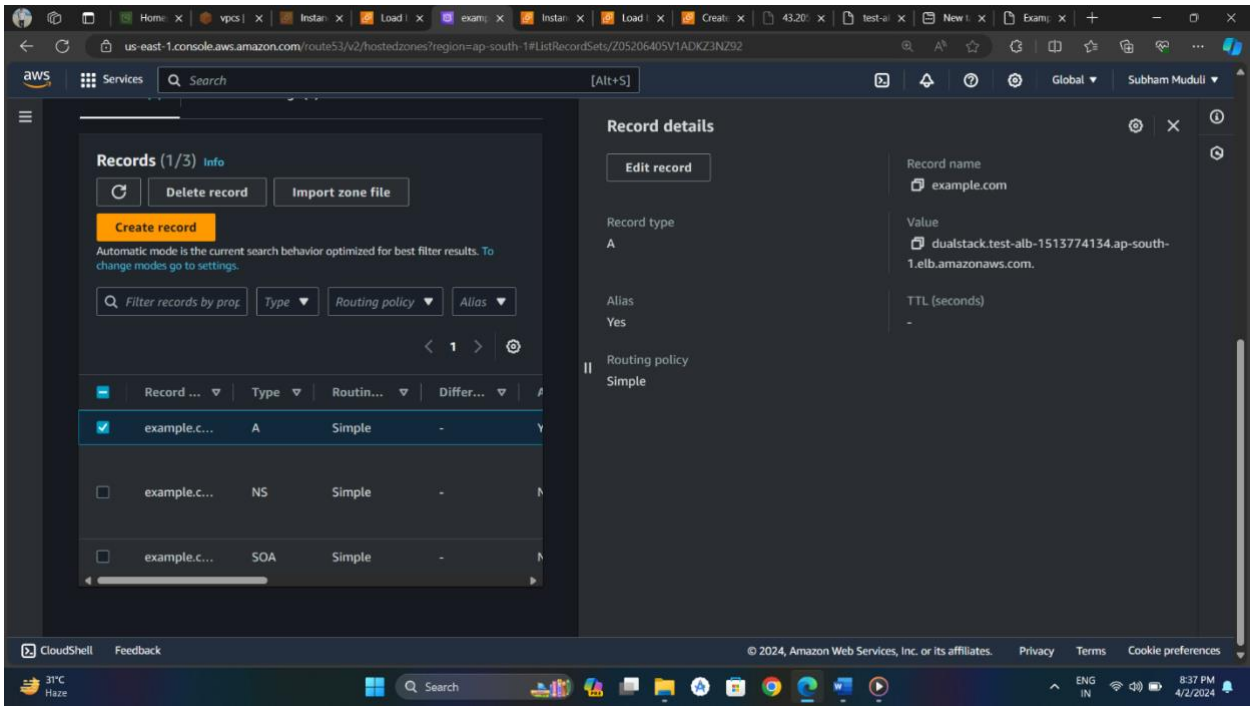
Create a Record Set:

- Inside your private hosted zone, create a new record set.
- Choose the record type appropriate for your use case (e.g., A or CNAME).
- Configure the record to point to the DNS name of your ALB.
- Configure the Application Load Balancer:
- Go to the EC2 console and navigate to the "Load Balancers" section.
- Select your ALB.

private Route 53 domain



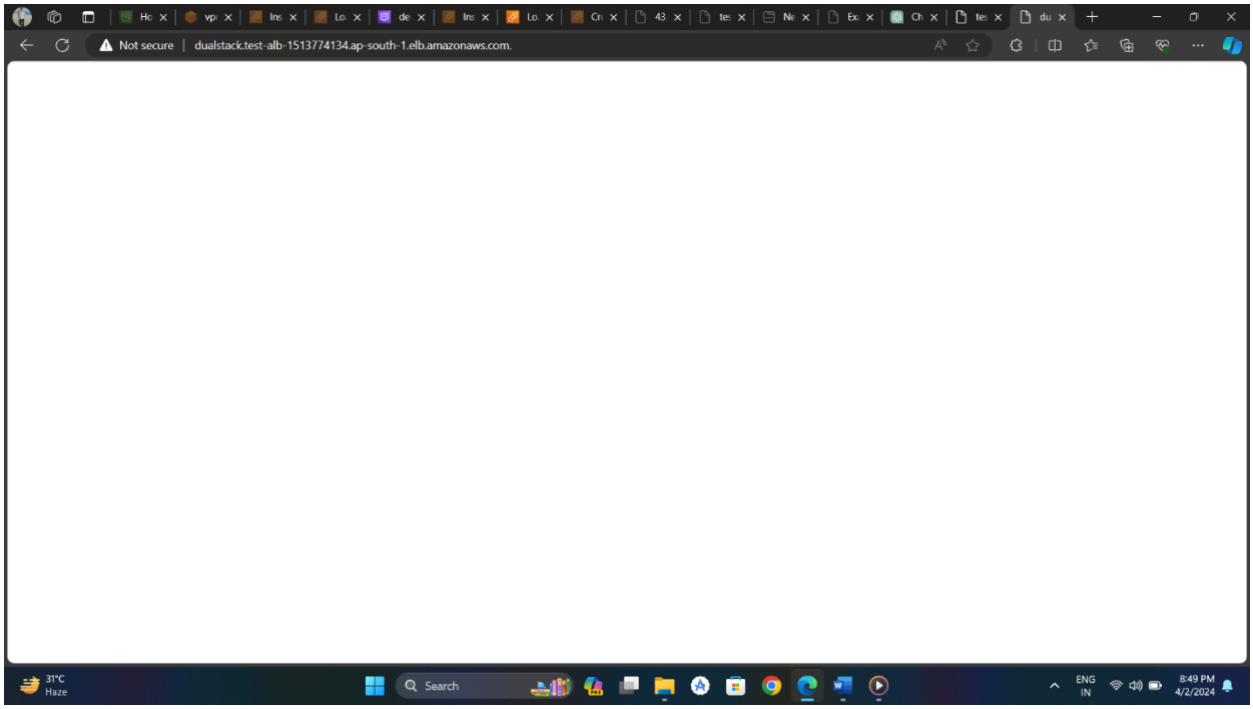
point it to an Application Load Balancer



2. Verify the domain through another machine in the same network.

Create a Private Hosted Zone in Route 53:

NB:- I didn't install any web server inside this. so for that it shows like this Empty White Screen.



S3:

1. Create an S3 bucket and restrict access to specific users.

2. Create a bucket accessible through an EC2 instance.

3. Create a bucket, copy, and sync data from a local source to the S3 bucket.

4. Hosting a static website in S3.

1. Create an S3 Bucket:

- Sign in to the AWS Management Console.
- Go to the S3 service.
- Click on "Create Bucket".
- Choose a unique bucket name and select the region.
- Do ACLs enabled.
- Disable Block *all* public access
- Click "Create".

2. Upload Website Files:

- In the bucket you created, click on the "Upload" button.
- Upload your website files (HTML, CSS, JavaScript, images, etc.).
- Ensure that your main page is named according to the index document specified earlier (e.g., index.html).

3. Configure Bucket for Website Hosting:

- Select the bucket you just created.
- Go to the "Properties" tab.
- Click on "Static website hosting enable it".
- Choose "Host a static website".

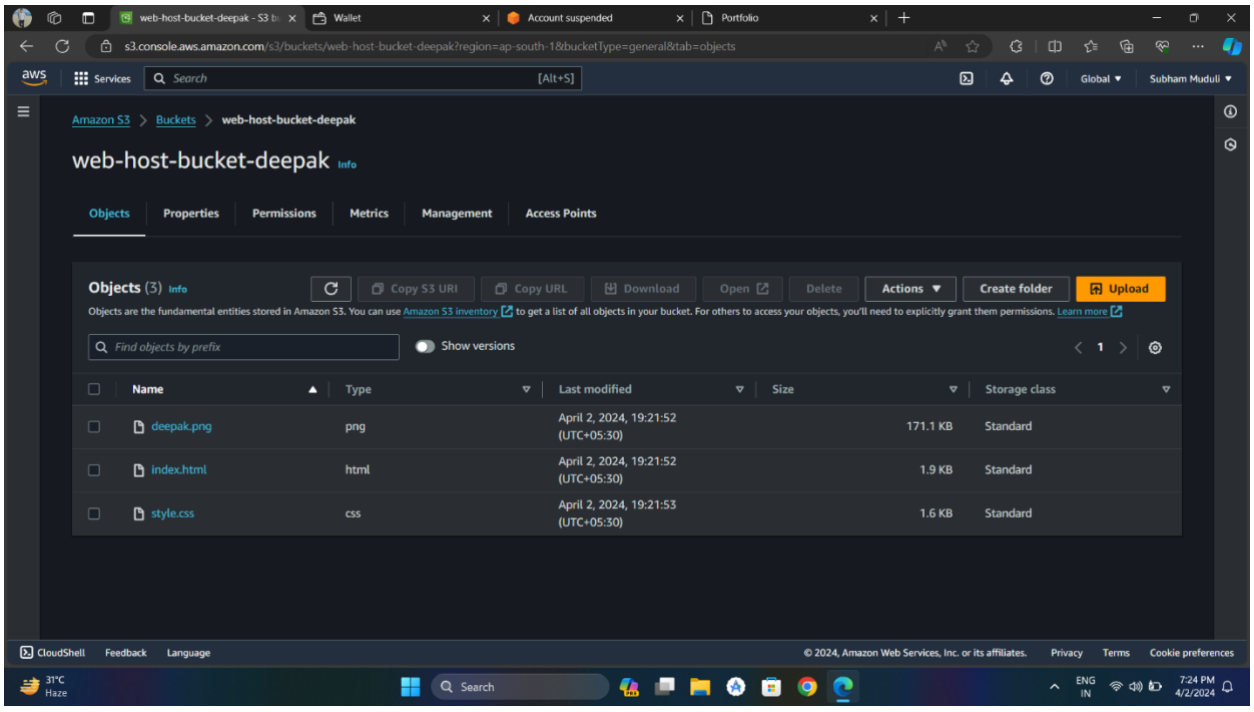
- Enter the index document (e.g., index.html).
- Optionally, enter an error document (e.g., error.html) for custom error pages.
- Click "Save".

Test Your Website:

- Once you've uploaded your files and set permissions, go back to the bucket's static website hosting configuration.
- You'll find the endpoint URL where your website is hosted. It will be something like "http://YOUR-BUCKET-NAME.s3-website-REGION.amazonaws.com."
- Open this URL in a web browser to verify that your website is working correctly.

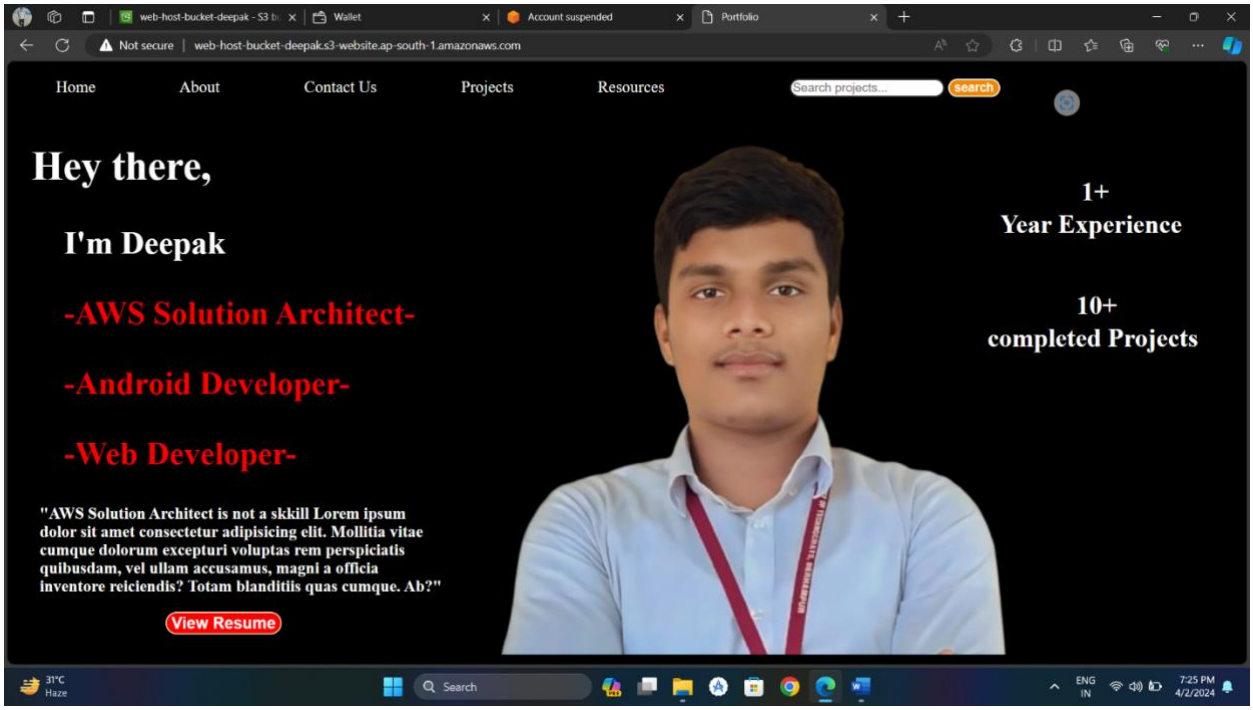
Demo

Bucket Screen



Website

URL-:http://web-host-bucket-deepak.s3-website.ap-south-1.amazonaws.com



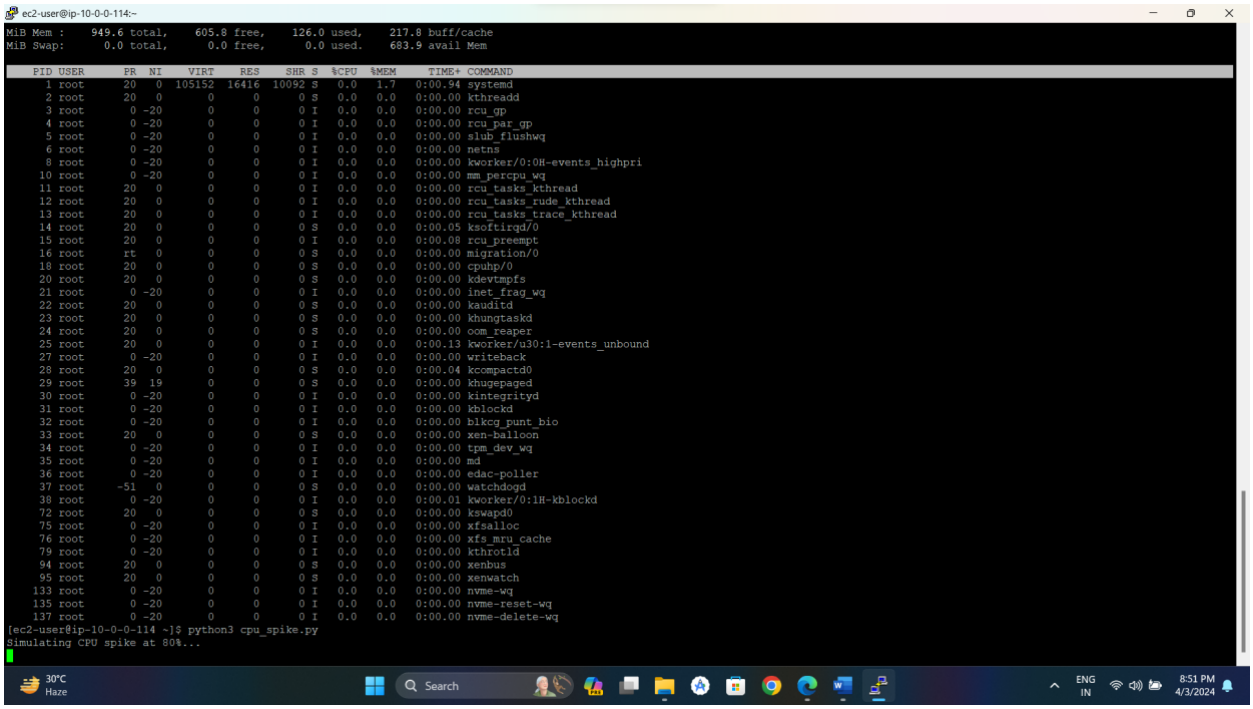
CloudWatch:

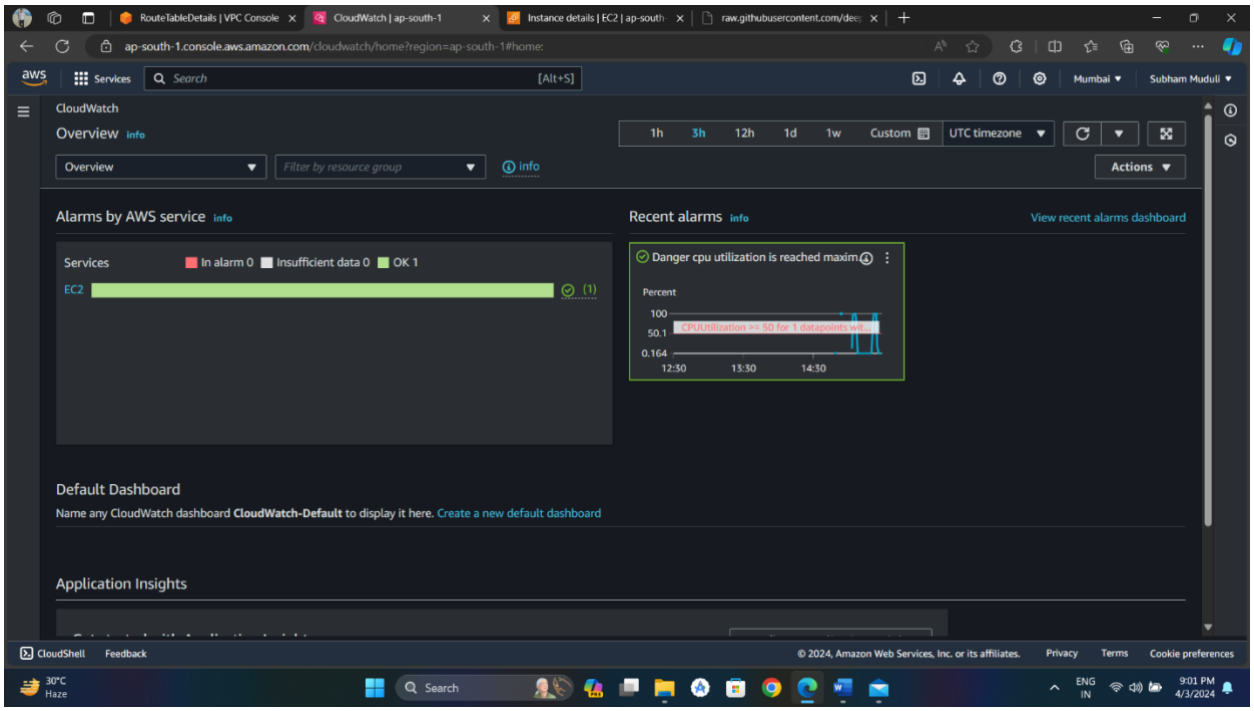
Task 1:

Create a CloudWatch Alarm for an EC2 instance. Trigger the alarm to send notifications to end-users via SNS when it crosses a predefined threshold.

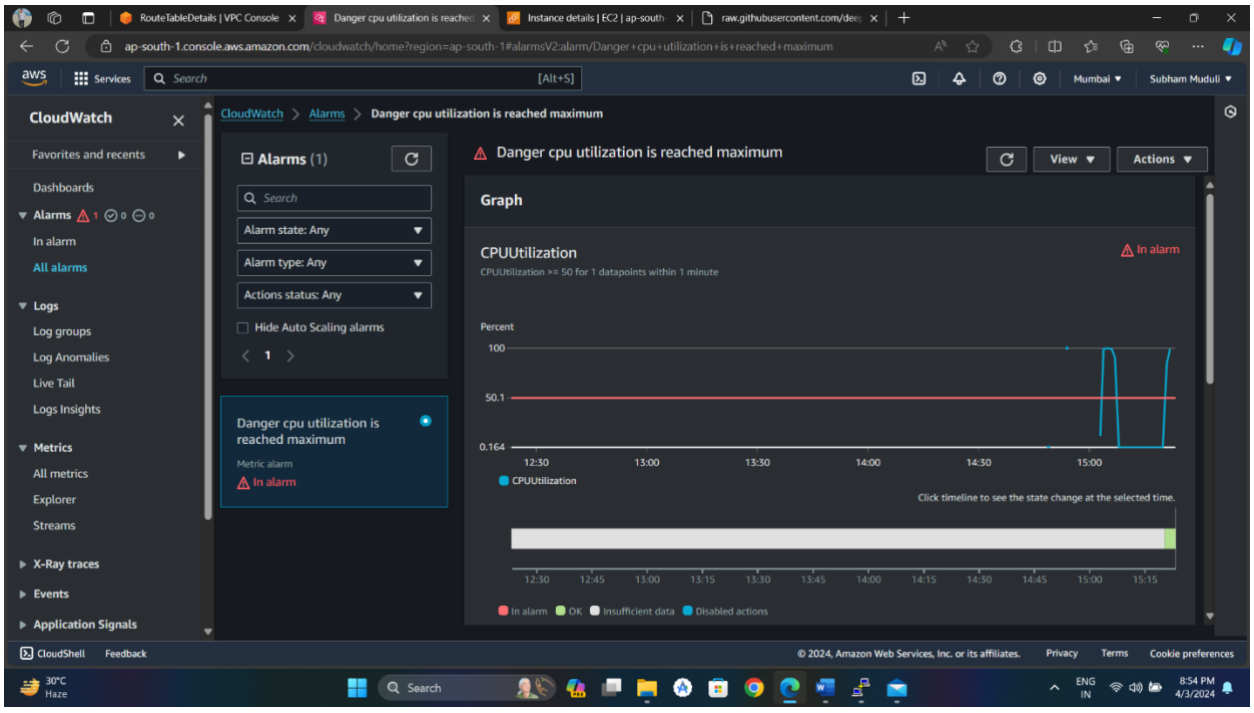
Cpu_spike code Running

Link of Python code:- https://github.com/deepak-401/cpu_spike/blob/main/cpu_spike

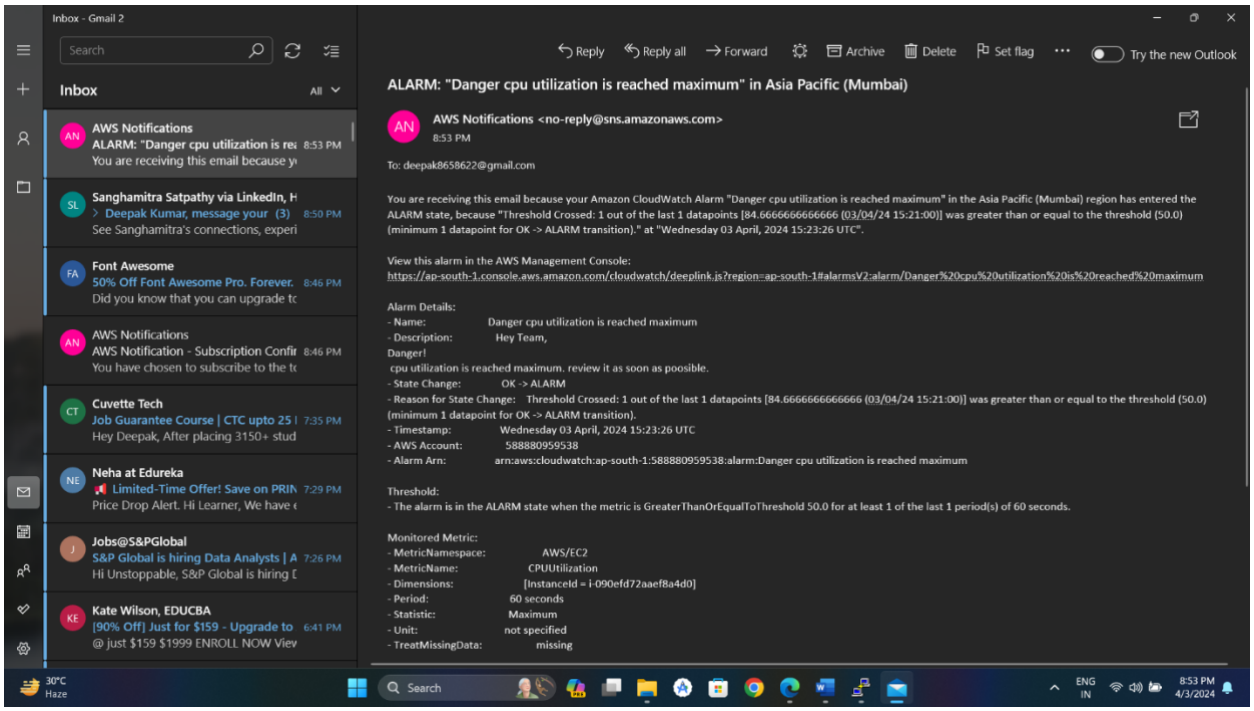




In Alaram State



E-mail



Monitor the following metrics:

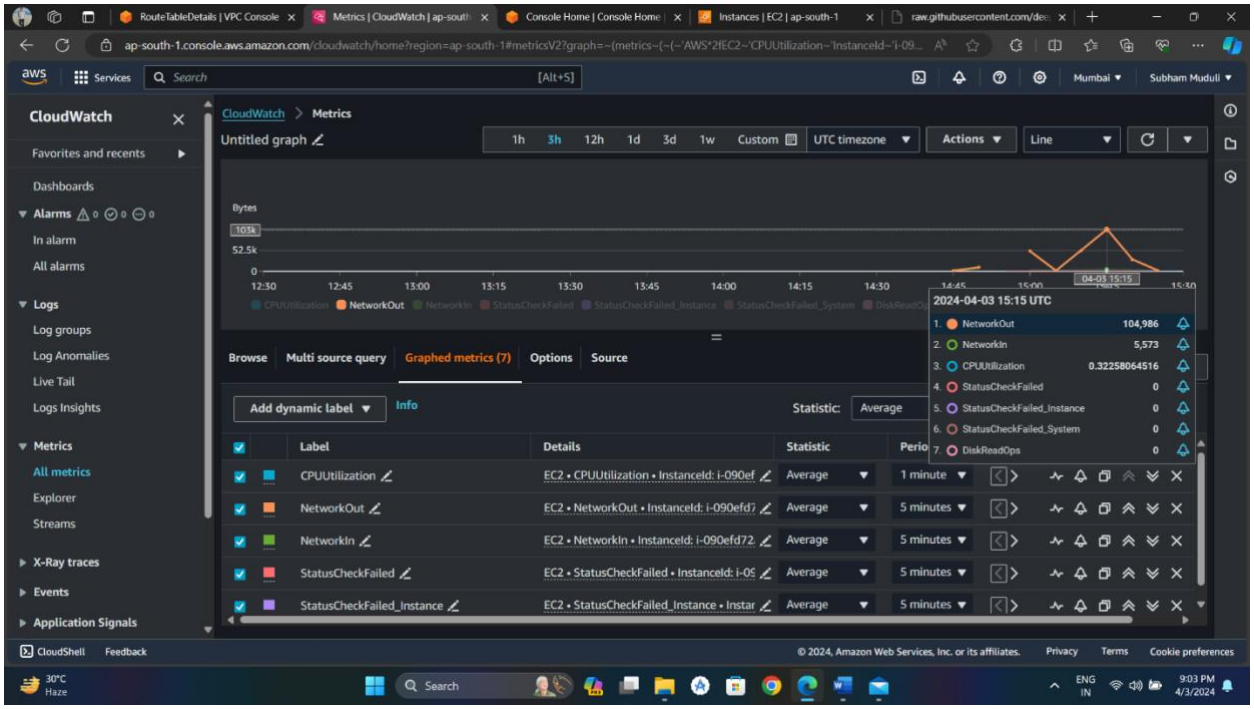
1. CPU Utilization

2. Memory Utilization

3. Disk Utilization

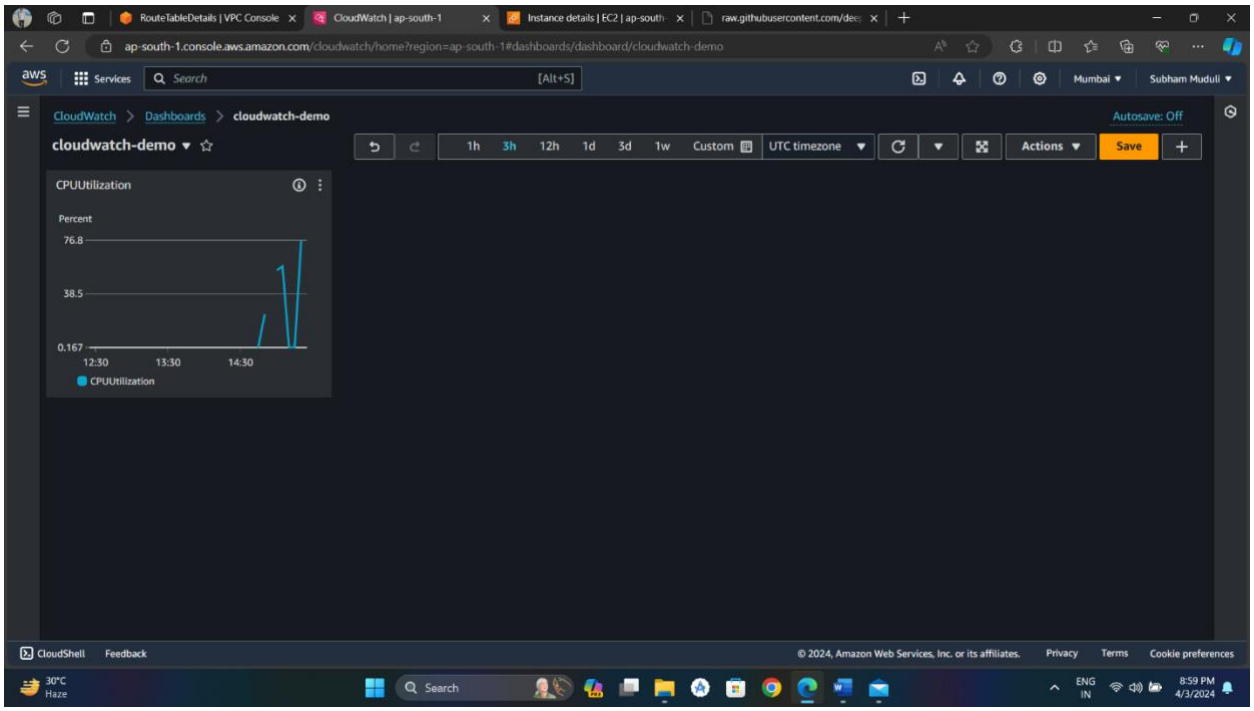
4. Network Out and In

5. Instance and System Status Check Failed



Task 2:

Create a dashboard to monitor CPU utilization and Memory Utilization for a specific EC2 instance.


RouteTableDetails | VPC Console
CloudWatch | ap-south-1
Instance details | EC2 | ap-south
raw.githubusercontent.com/dee
ap-south-1.console.aws.amazon.com/cloudwatch/home?region=ap-south-1#dashboards/dashboard/cloudwatch-demo
aws
Services
Search
[Alt+S]
Mumbai
Subham Mudull
CloudWatch > Dashboards > cloudwatch-demo
Autosave: Off
cloudwatch-demo
1h
3h
12h
1d
3d
1w
Custom
UTC timezone
Actions
Save
CPUUtilization
Percent
76.8
38.5
0.167
12:30
13:30
14:30
CPUUtilization
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
30°C
Haze
Search
ENG
IN
8:59 PM
4/3/2024