AWS Infrastructure Setup Project

1. Project Overview

This project involves setting up a highly available web application on AWS with the following architecture:

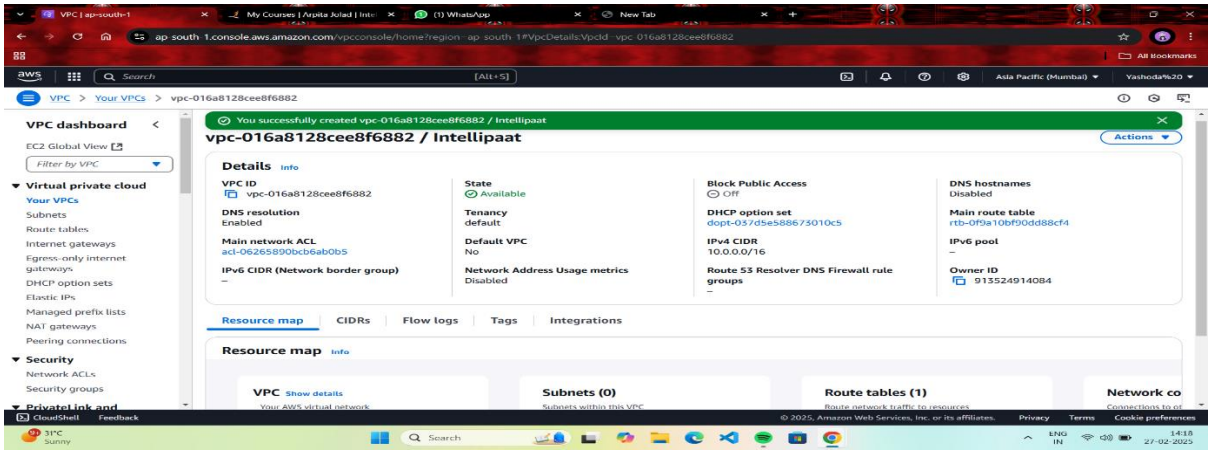
- **VPC with Public Subnets** for networking and high availability.
- **EC2 Instances** hosting Apache and Nginx web servers.
- **Application Load Balancer (ALB)** for traffic distribution with weighted routing.
- **FSx for Lustre** shared storage accessible by both EC2 instances.
- **SNS (Simple Notification Service)** for sending alerts and notifications.

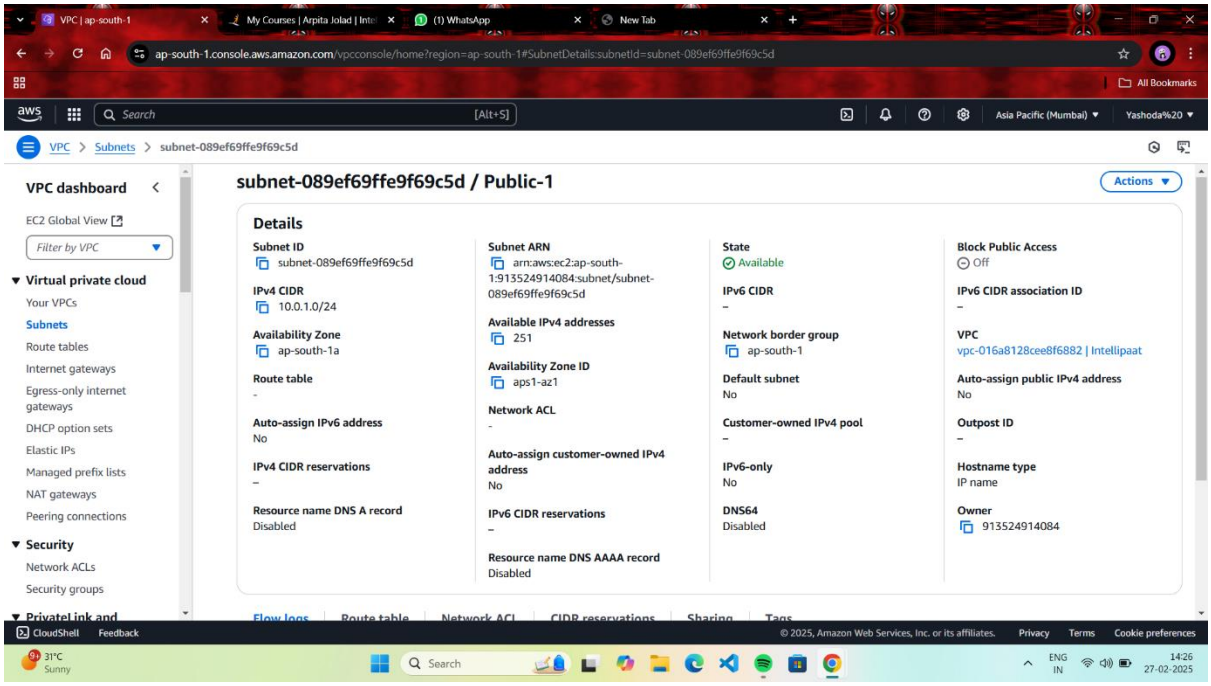
---

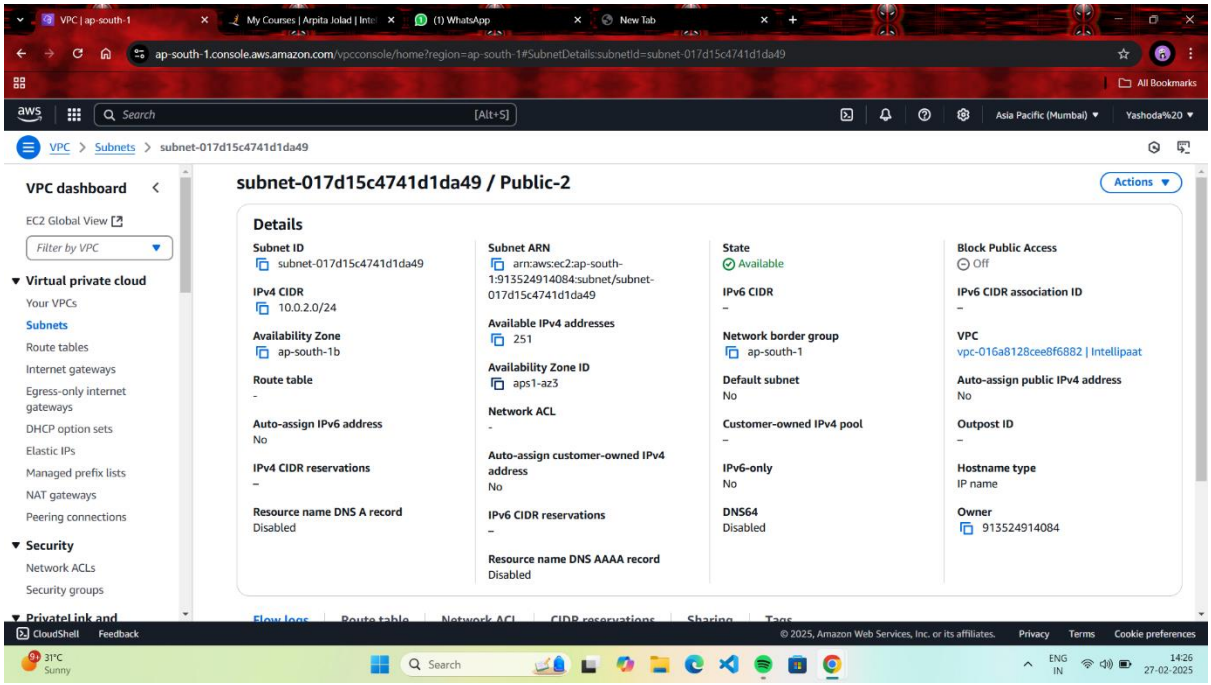
2. VPC and Subnet Setup

- **VPC Name:** Intellipaat
- **VPC CIDR Block:** 10.0.0.0/16
- **Public Subnet 1:**
  - AZ: us-east-1a
  - CIDR: 10.0.1.0/24
- **Public Subnet 2:**
  - AZ: us-east-1b
  - CIDR: 10.0.2.0/24
- **Internet Gateway:** Attached to allow internet access.

🖼️ **[Screenshot: VPC and Subnet Details]**







3. EC2 Instances Setup

3.1 Launch EC2 Instances

- **app1:** Apache Web Server
  - AMI: Ubuntu 22.04
  - Subnet: Public-1
- **app2:** Nginx Web Server

- AMI: Ubuntu 22.04
- Subnet: Public-2

3.2 Configure Web Servers

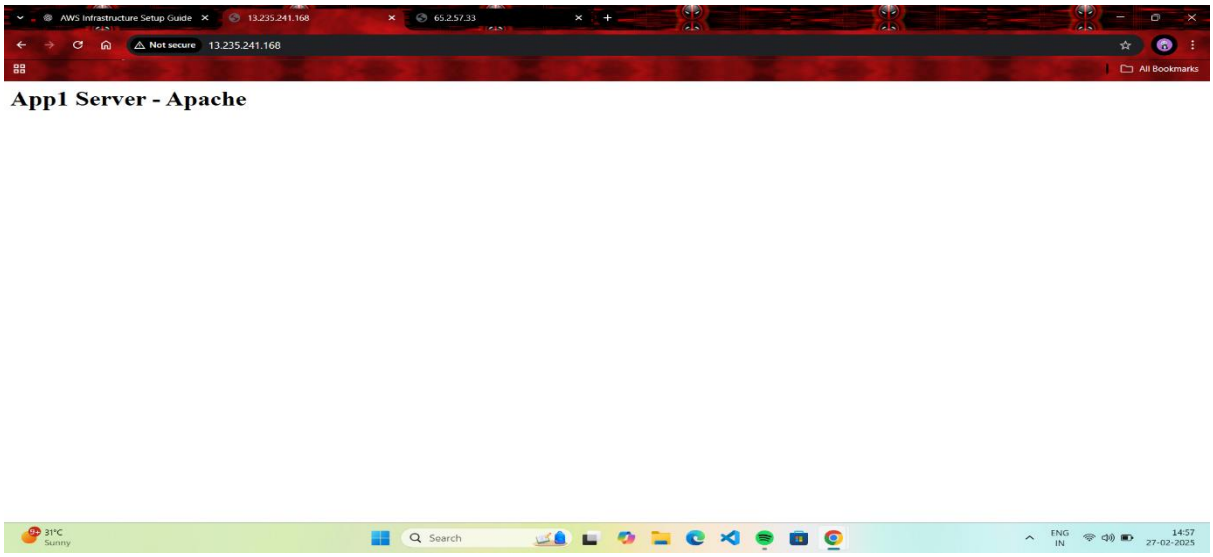
For app1 (Apache):

```
sudo apt update -y
sudo apt install -y apache2
echo "<h1>App1 Server - Apache</h1>" | sudo tee /var/www/html/index.html
sudo systemctl start apache2
sudo systemctl enable apache2
```

For app2 (Nginx):

```
sudo apt update -y
sudo apt install -y nginx
echo "<h1>App2 Server - Nginx</h1>" | sudo tee /usr/share/nginx/html/index.html
sudo systemctl start nginx
sudo systemctl enable nginx
```

🖼️ **[Screenshot: Web Pages from app1 and app2]**



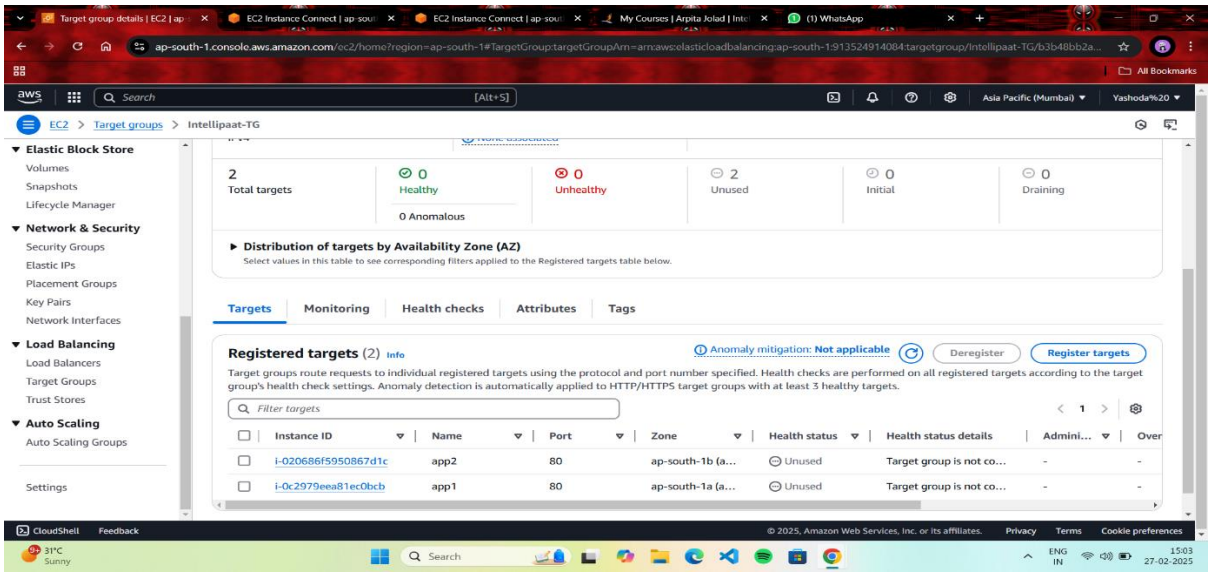
4. Application Load Balancer (ALB) Setup

- **ALB Name:** Intellipaat-ALB
- **Type:** Internet-facing
- **Subnets:** Public-1 & Public-2

4.1 Create Target Groups

- **Intellipaat-TG:** Target group for app1
- **Intellipaat-TG2:** Target group for app2

🖼️ [Screenshot: Target Group Configuration]

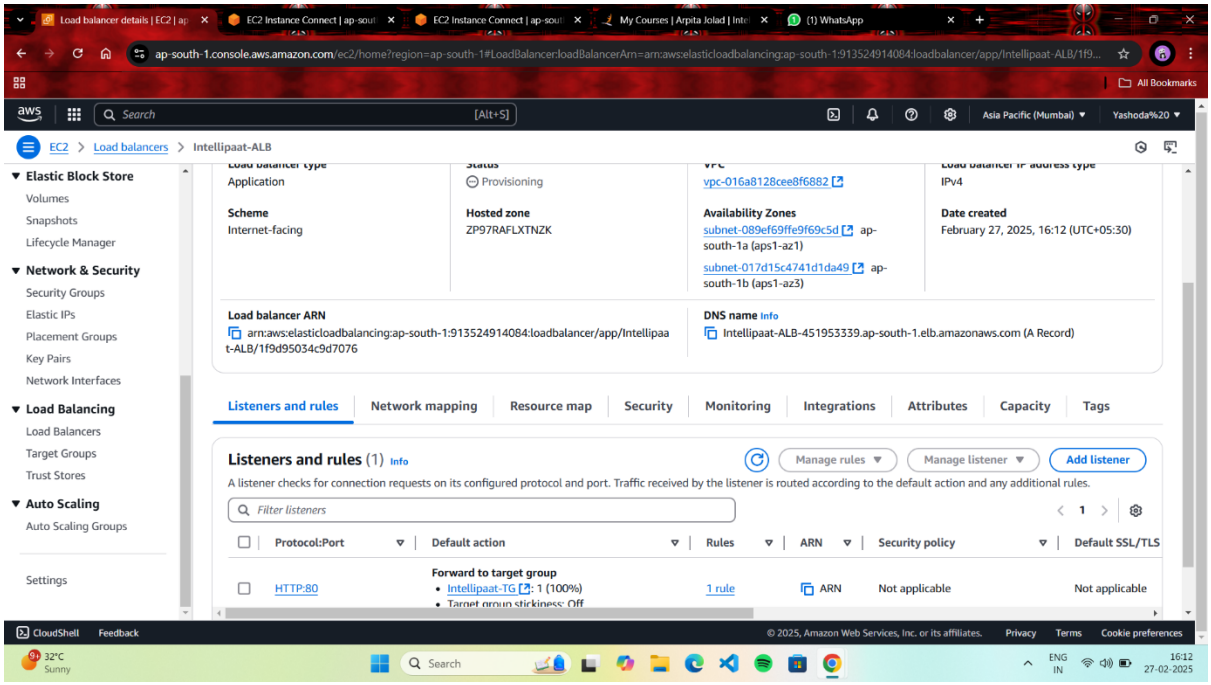


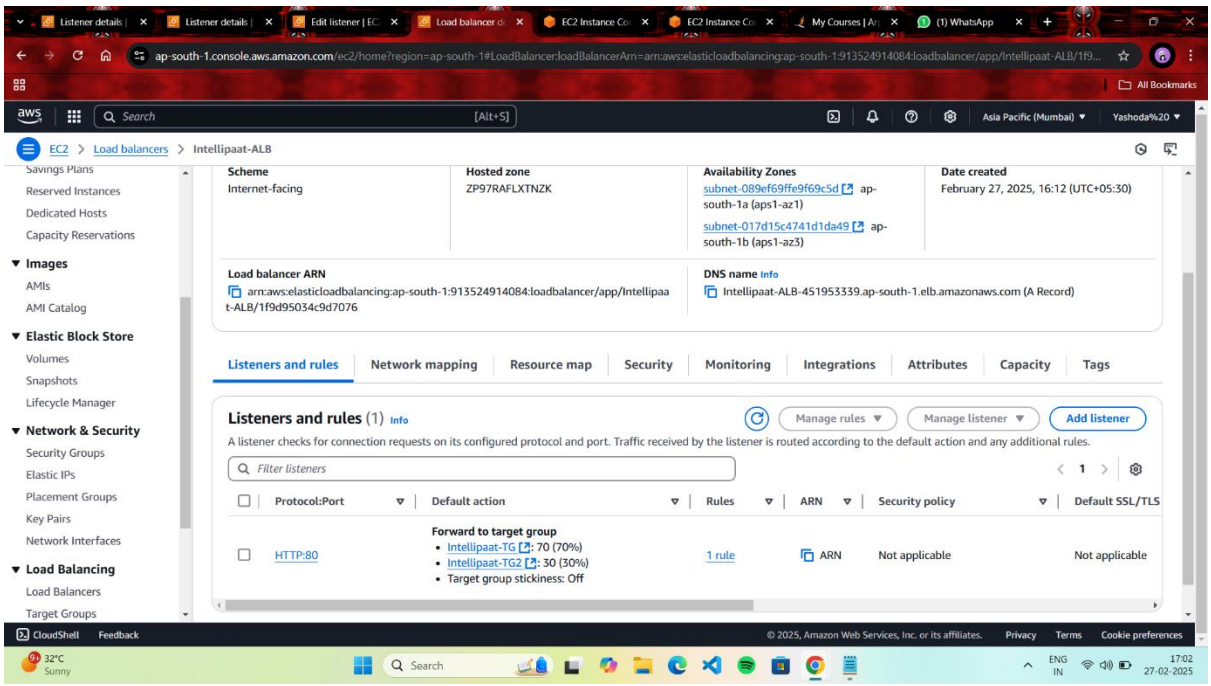
4.2 Configure Weighted Routing

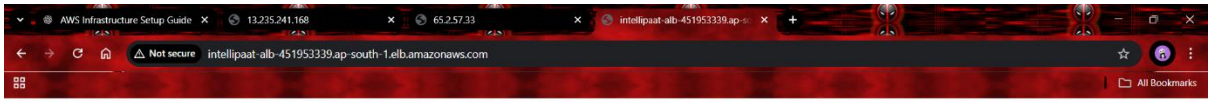
Forwarding Rule:

- **Intellipaat-TG:** 70% Traffic
- **Intellipaat-TG2:** 30% Traffic

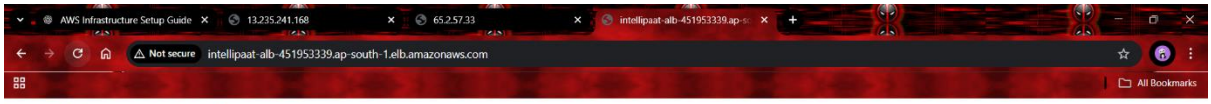
🖼️ [Screenshot: ALB Weighted Routing Configuration]







App2 Server - Nginx



App1 Server - Apache

5. FSx for Lustre Setup (Shared Storage)

- **Create FSx for Lustre**
- **Attach to Both EC2 Instances**

Mount FSx on app1 and app2:

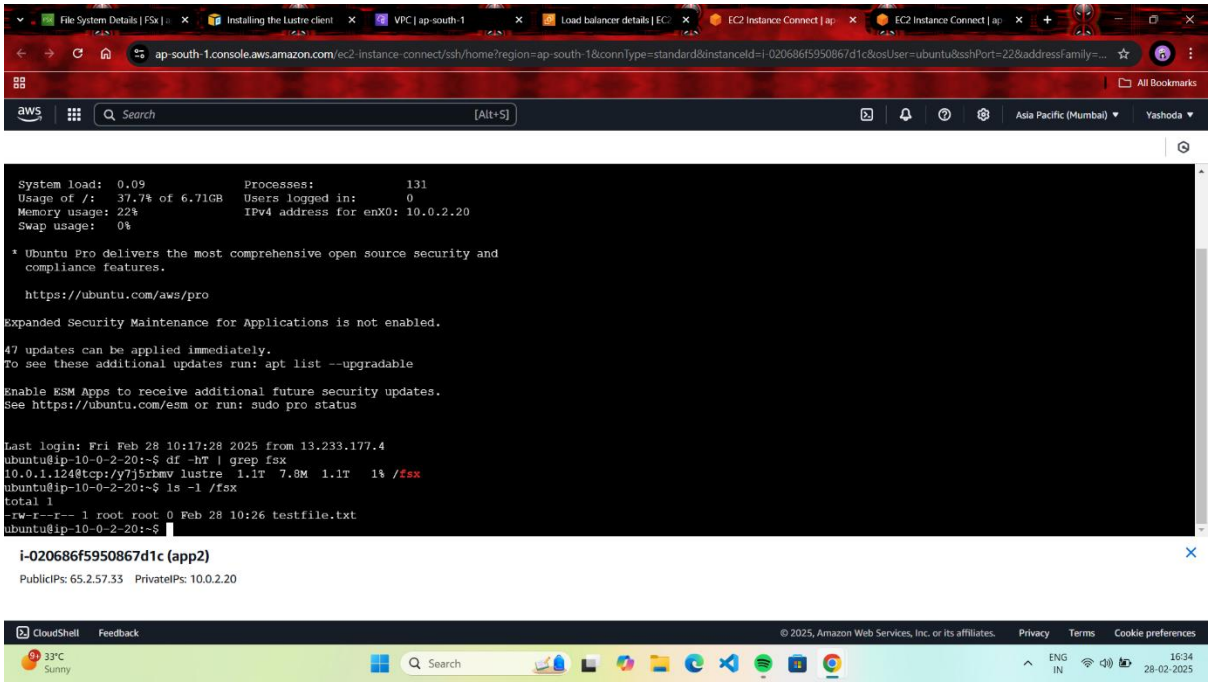
sudo mkdir /fsx

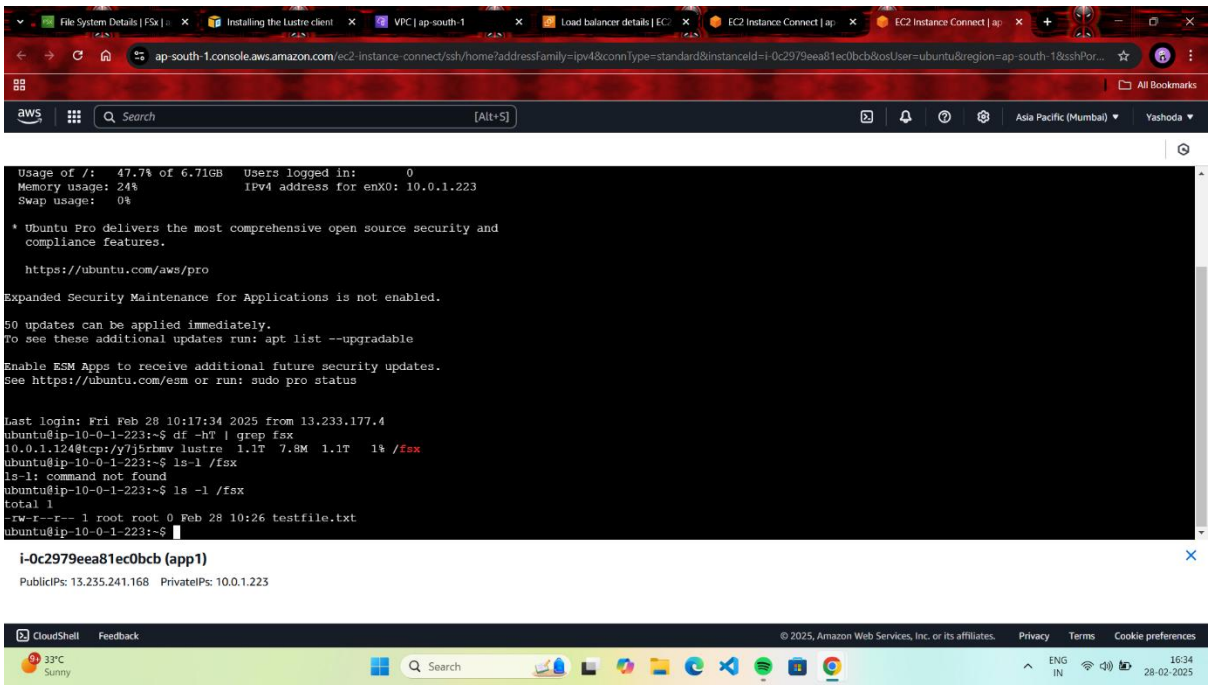
sudo mount -t lustre <FSX_MOUNT_POINT>:/fsx /fsx

Test Shared Storage:
echo "Hello from app1" | sudo tee /fsx/testfile.txt

cat /fsx/testfile.txt # Run from app2

🖼️ [Screenshot: FSx Mount and Shared File Access]





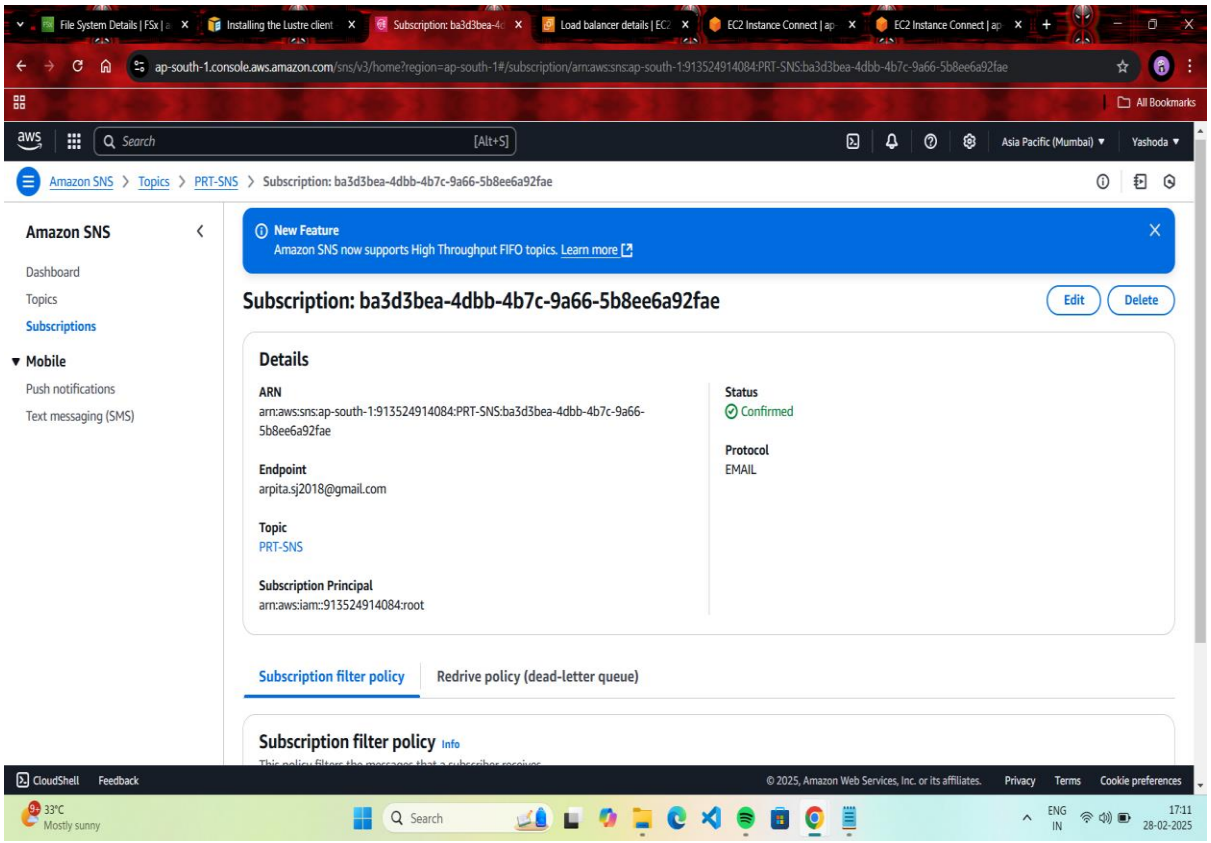
6. Create SNS Topic Using CloudFormation

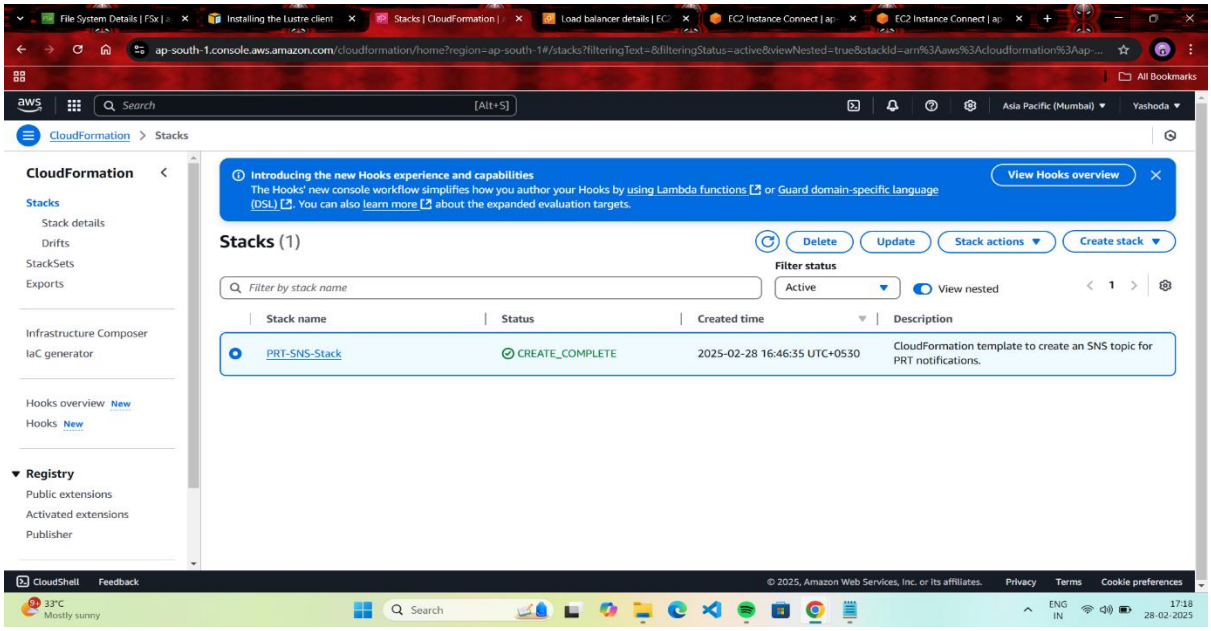
CloudFormation Template (YAML):

```
Resources:
  SNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName: PRT-SNS

  SNSSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref SNSTopic
      Protocol: email
      Endpoint: your-email@example.com
```

🖼️ **[Screenshot: CloudFormation Stack Creation and SNS Topic]**



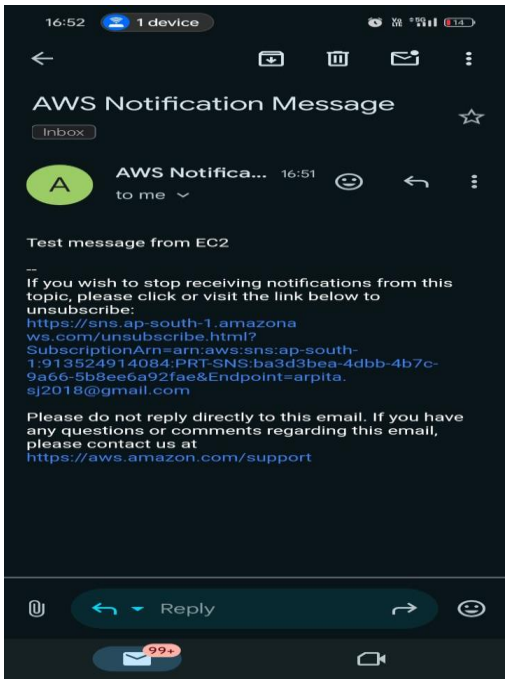


7. Publish Test Notification

Publish from EC2:

aws sns publish --topic-arn <SNS_TOPIC_ARN> --message "Test notification from EC2"

🖼️ [Screenshot: SNS Notification Received in Email]



9. Conclusion

This project successfully demonstrates the deployment of a highly available and scalable web application infrastructure on AWS, integrating key services like VPC, EC2, ALB, FSx for Lustre, and SNS. By setting up a well-structured network with public subnets across different Availability Zones and attaching an Internet Gateway, we ensured seamless internet access and high availability. The EC2 instances, hosting Apache and Nginx web servers, were strategically placed in separate subnets and registered with an Application Load Balancer, which efficiently distributed traffic using weighted routing (70% to app1 and 30% to app2). This setup allows optimized performance and load management, critical for real-world production environments.

Additionally, the integration of FSx for Lustre provided a powerful shared storage solution, enabling both EC2 instances to access and modify data in real time. The deployment of an SNS topic through CloudFormation further enhanced the architecture by introducing an automated notification system for critical events. With thorough testing and verification, including the validation of web server responses, shared storage functionality, and SNS email notifications, this project stands as a well-rounded implementation of AWS services. The architecture is scalable and efficient, making it a strong foundation for future enhancements and real-world applications.