

High-Availability Web Application with AWS EFS Shared Storage

Step 1) We will create EFS file system

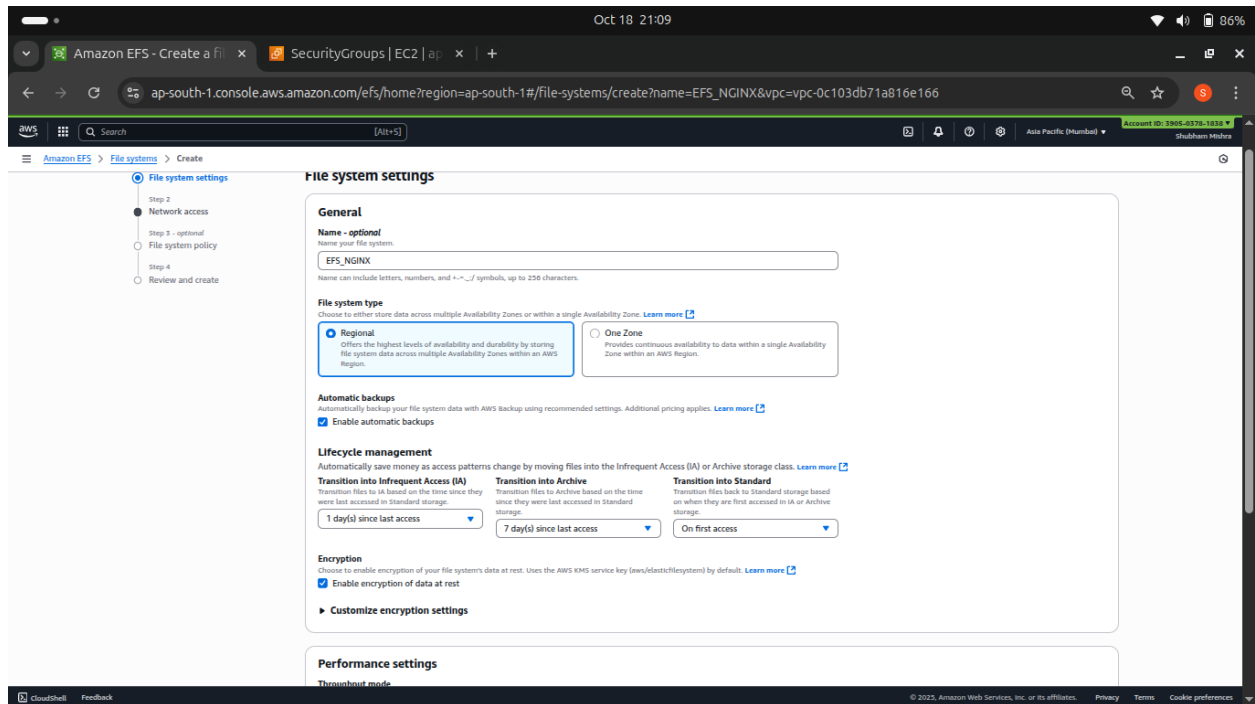
The screenshot shows the AWS Management Console interface for creating a new Elastic File System (EFS). The 'Create file system' modal is open, displaying the following configuration:

- Name - optional:** A text input field containing 'EFS_NGINX'. Below it, a note states: 'Name can include letters, numbers, and +,=,_,/ symbols, up to 256 characters.'
- Virtual Private Cloud (VPC):** A dropdown menu showing 'vpc-0c103db71a816e166' with 'default' written below it.
- Recommended settings:** A section with a disclaimer: 'Your file system is created with the following recommended settings unless you choose to customize the file system. You will be charged for storage and throughput. We recommend reviewing pricing for these features using the [AWS Pricing Calculator](#).' Below this is a table:

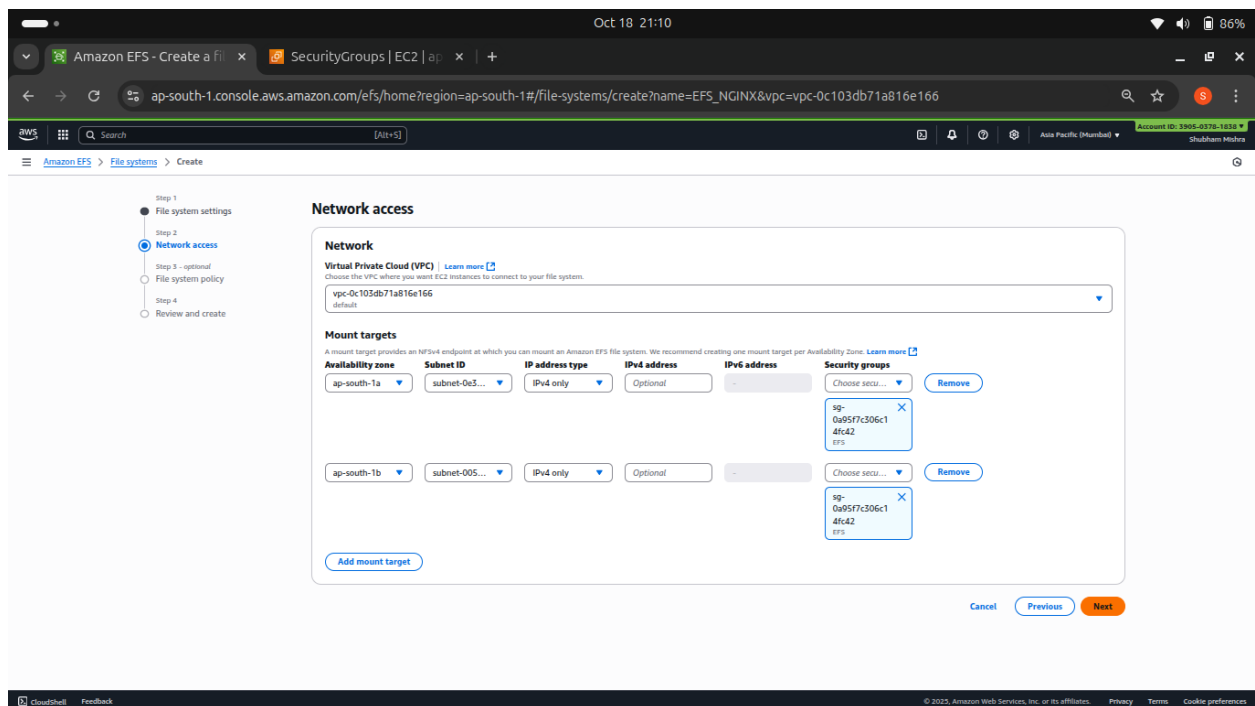
Setting	Value	Editable after creation
Throughput mode Learn more	Elastic	Yes
Transition into Infrequent Access (IA)	30 day(s) since last access	Yes

At the bottom of the modal, there are three buttons: 'Cancel', 'Customize', and 'Create file system' (highlighted in orange).

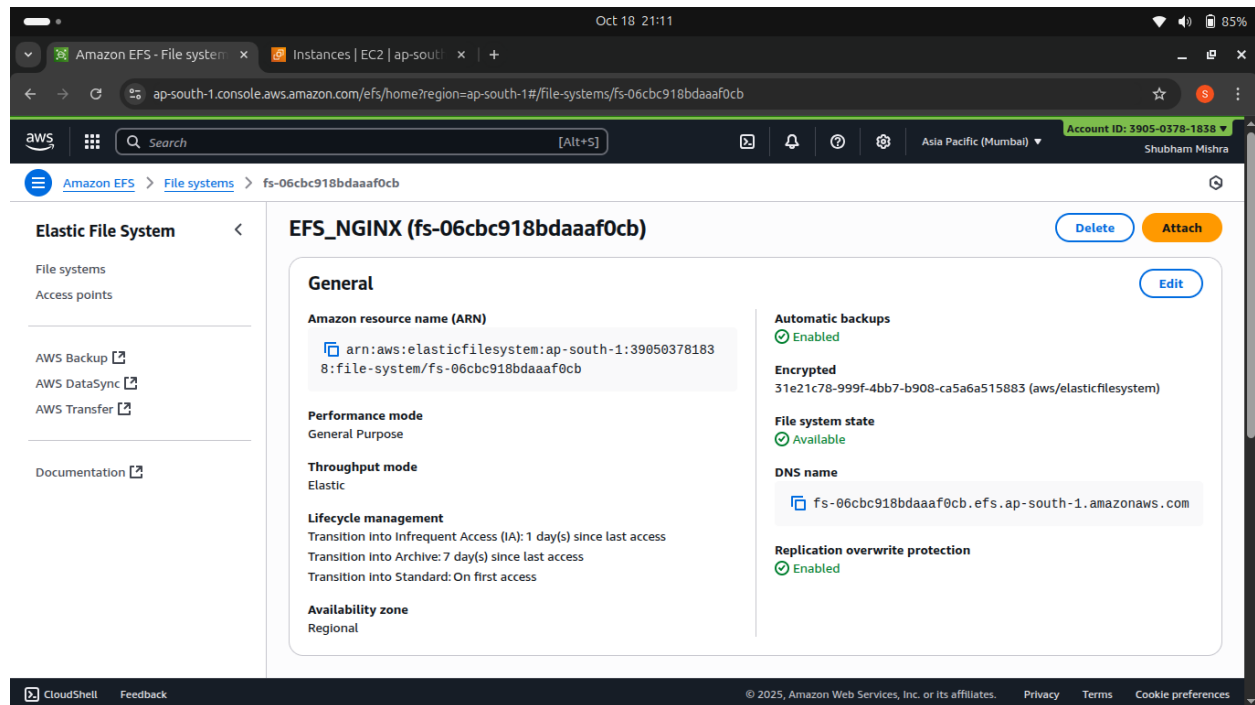
Step 2) We will select default setting



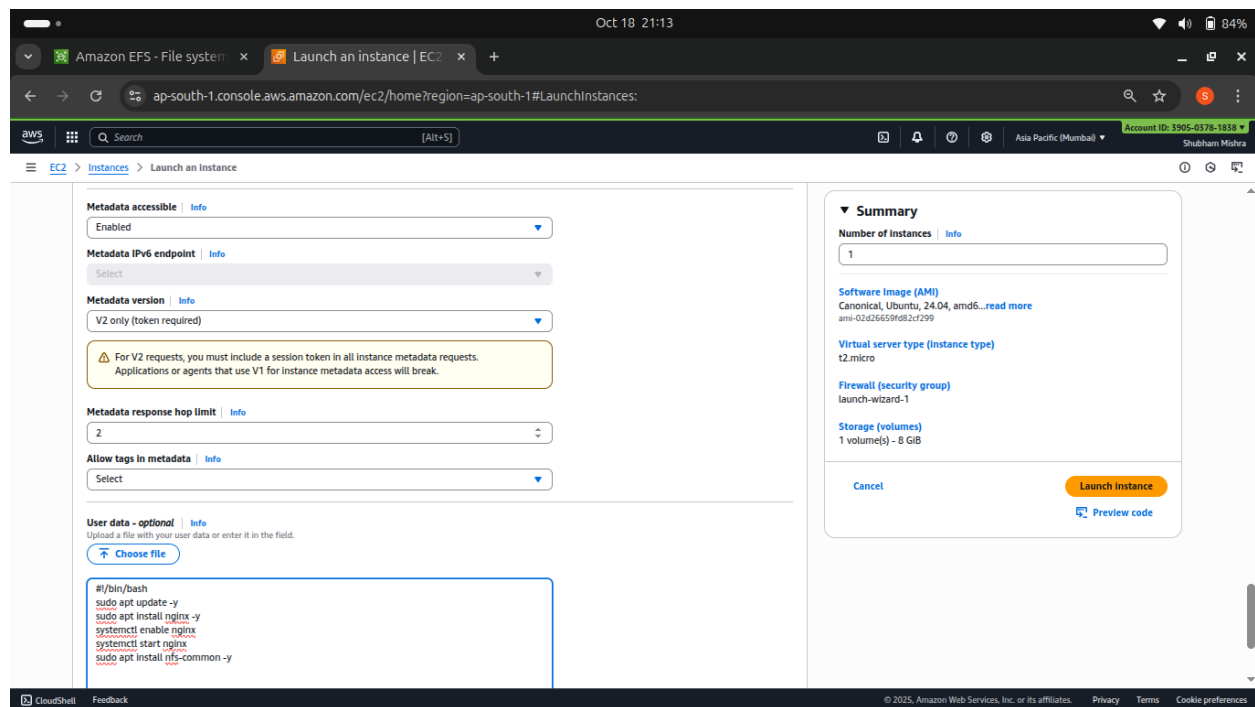
Step 3) We will select subnet for mounting this efs and security group which should allow port 2049 for efs



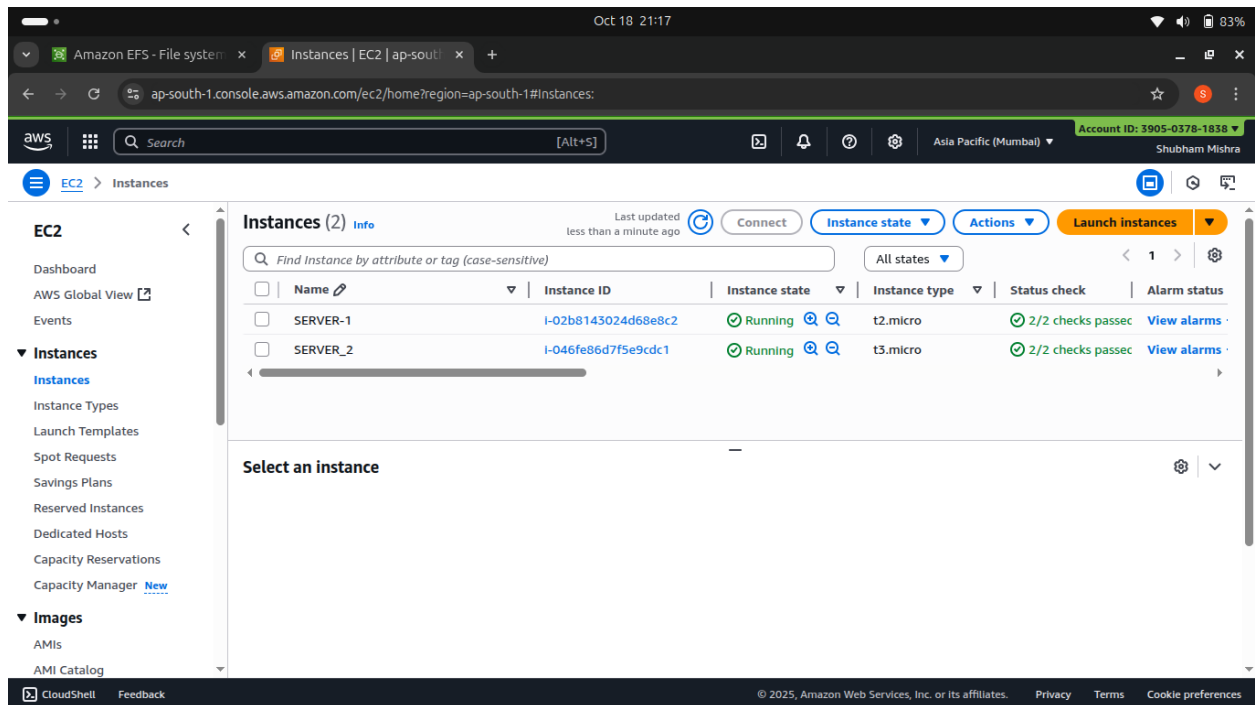
Step 4) We can see EFS is available to use now



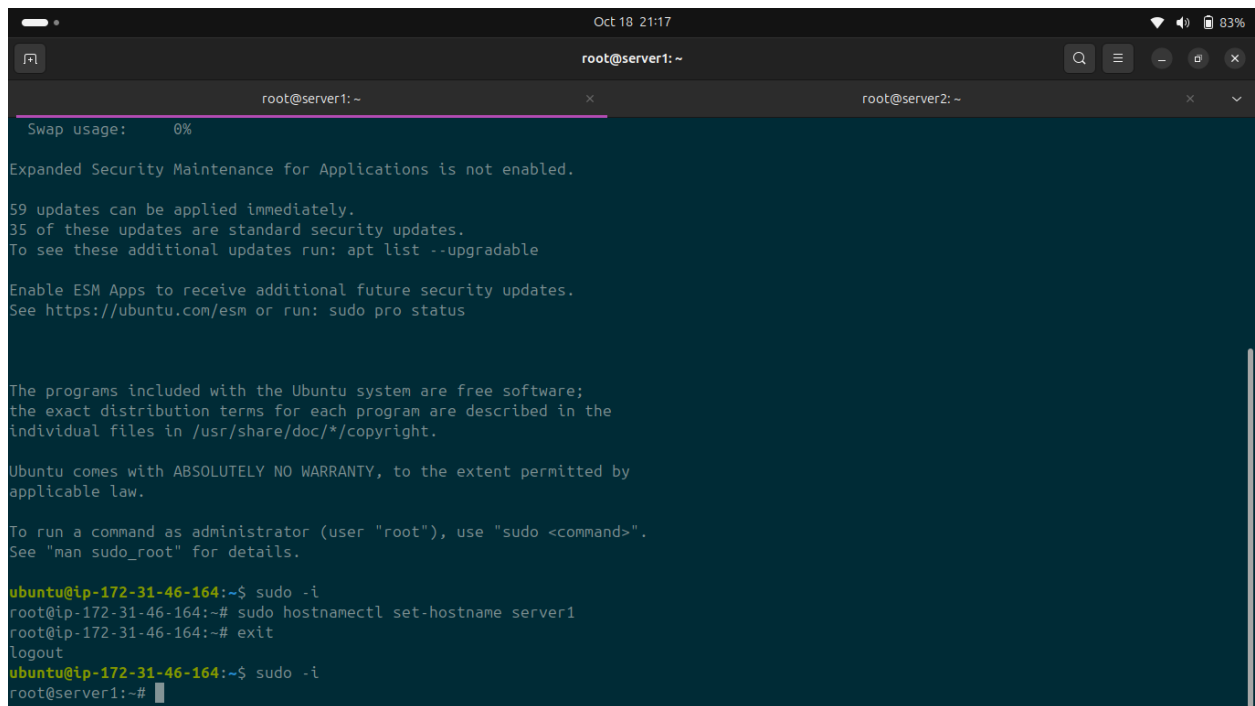
Step 5) Now we will launch two ec2 across different az and enter user data to install nfs and nginx package



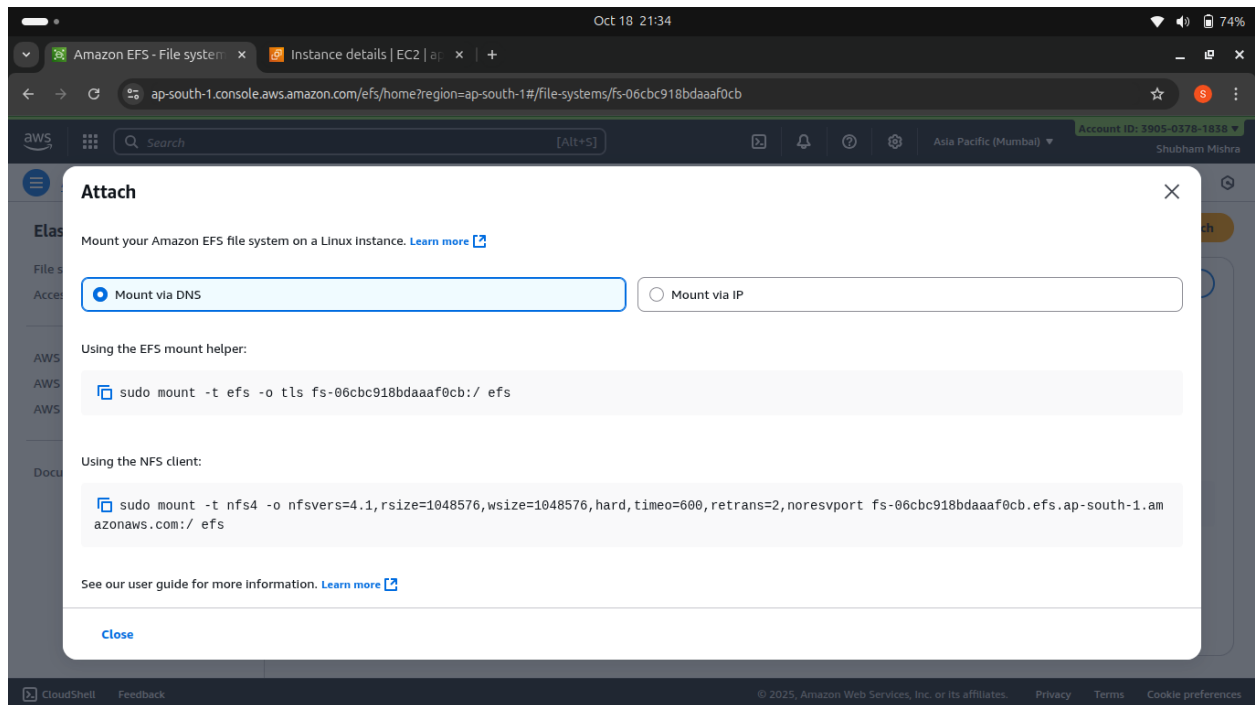
Step 6) See our instances are now available to use



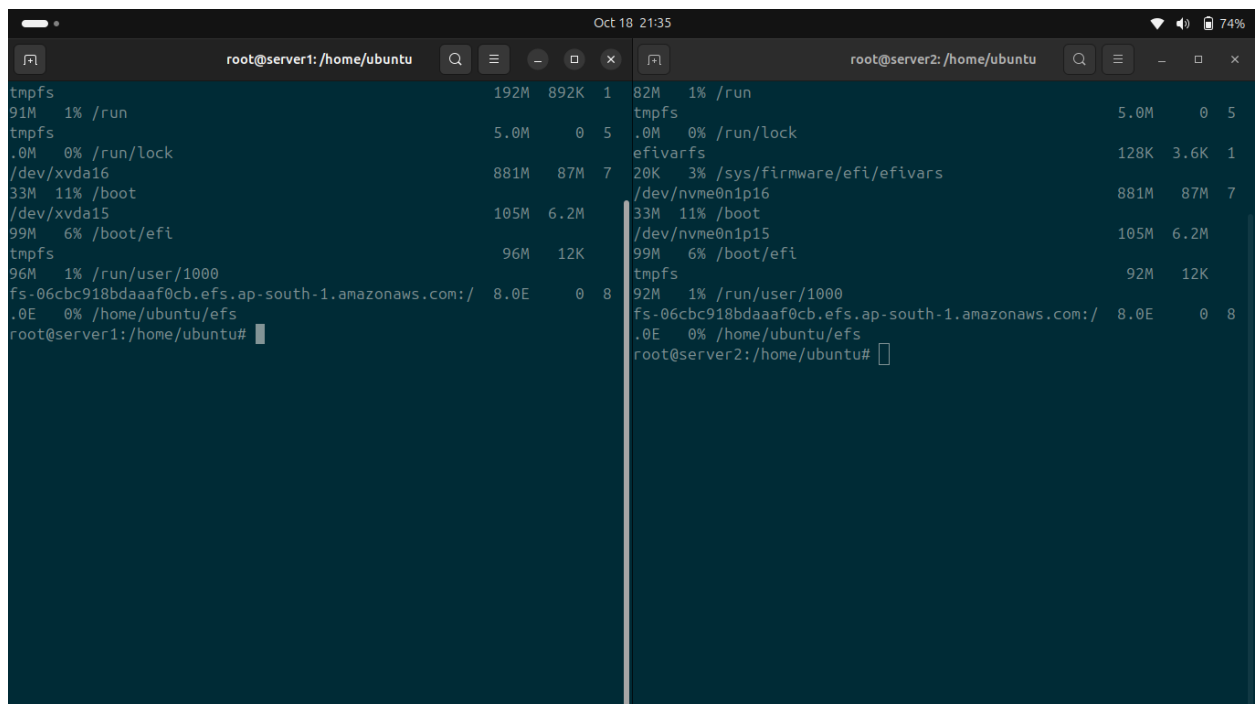
Step 7) We will ssh into both instances



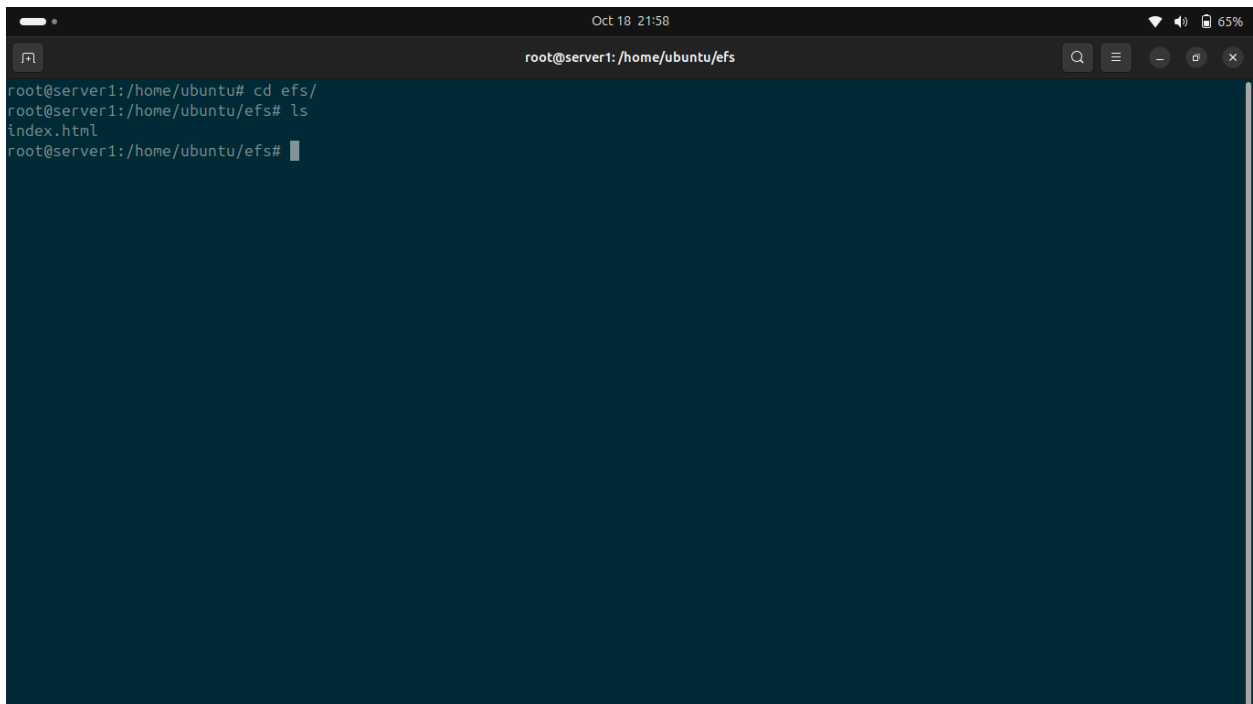
Step 8) Now we'll mount efs by dns on both ec2 instances



Step 9) See efs successfully mounted on both ec2 instances



Step 10) We have created folder in efs and added index.html to be accessed from both ec2 instances



```
root@server1:/home/ubuntu# cd efs/
root@server1:/home/ubuntu/efs# ls
index.html
root@server1:/home/ubuntu/efs#
```

Step 11) Fix the permission and everything as shown below



```
bash

# Give execute permission on parent directories
sudo chmod o+x /home
sudo chmod o+x /home/ubuntu

# Set proper permissions on EFS directory
sudo chmod 755 /home/ubuntu/efs
sudo chmod 644 /home/ubuntu/efs/index.html

# Test as www-data user
sudo -u www-data cat /home/ubuntu/efs/index.html
```

Step 2: Configure Nginx on BOTH instances

```
bash

# Edit the default site config
sudo nano /etc/nginx/sites-available/default
```

Update to:

```
nginx

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /home/ubuntu/efs;
    index index.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Step 3: Test and restart Nginx on BOTH instances

```
bash

# Test configuration
sudo nginx -t

# Restart Nginx
sudo systemctl restart nginx

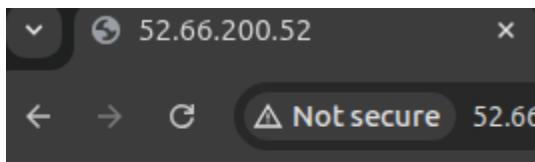
# Test locally
curl http://localhost
```

Step 12) Now we curl localhost on both ec2 instances and can see both serves same file

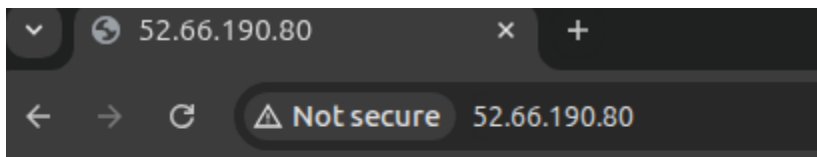
```
Oct 18 22:06
root@server1:/home/ubuntu/efs# curl localhost
<h1>Hello from EFS</h1>
root@server1:/home/ubuntu/efs#

root@server2:/etc/nginx/sites-available# curl localhost
<h1>Hello from EFS</h1>
root@server2:/etc/nginx/sites-available#
```

Step 13) After accessing public ip on both ec2 instances got same file from efs



Hello from EFS



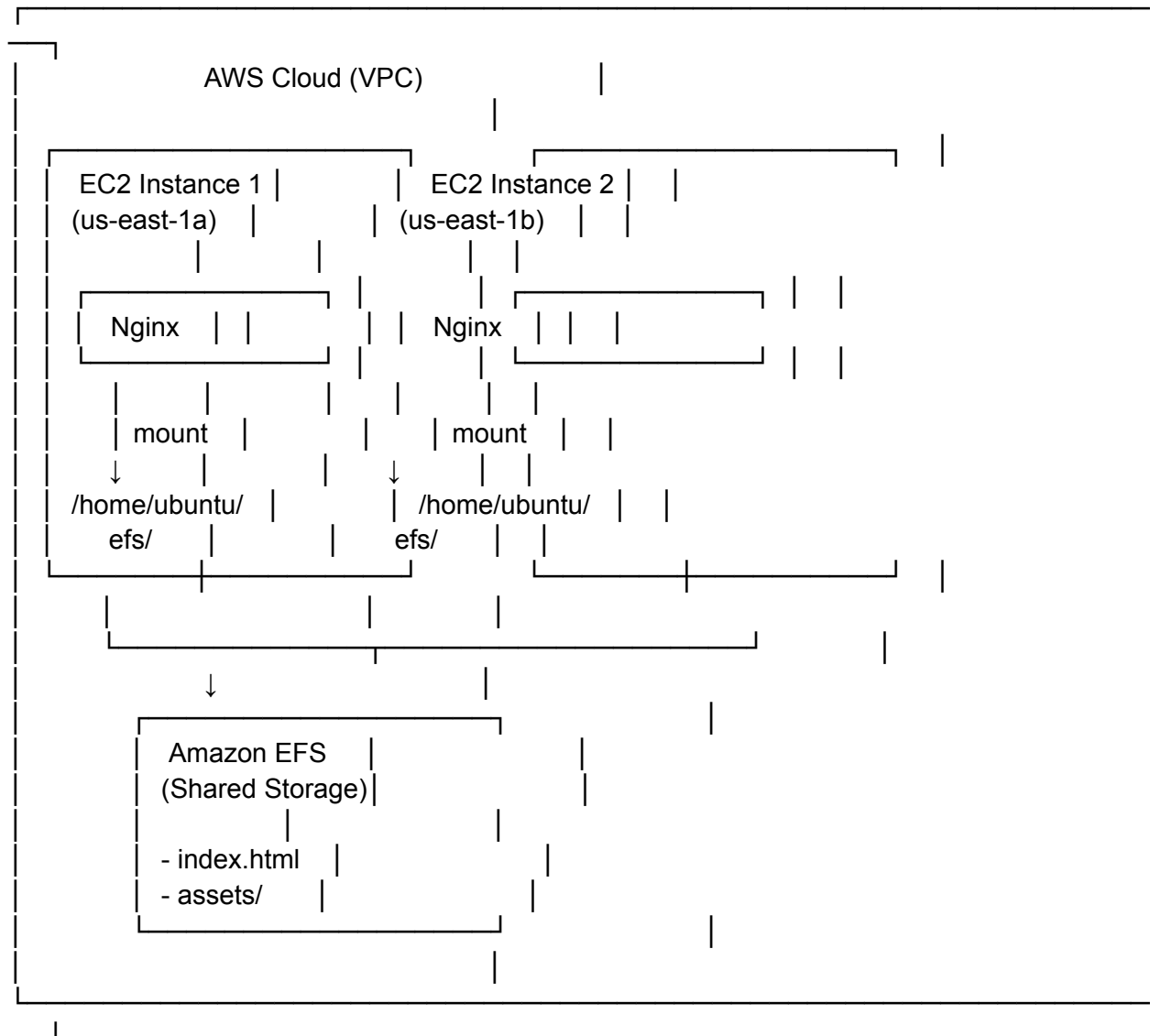
Hello from EFS

Technologies Used

Technology	Purpose
AWS EC2	Virtual servers hosting web applications
AWS EFS	Network File System for shared storage
Nginx	High-performance web server
Linux (Ubuntu)	Operating system
NFS v4.1	Network file sharing protocol
Bash Scripting	Automation and configuration

Key Features

1. **High Availability:** Multiple web servers ensure service continuity
2. **Shared Storage:** Centralized content management via EFS
3. **Real-time Synchronization:** Changes propagate instantly across all servers
4. **Scalability:** Easy to add more EC2 instances to the cluster
5. **Cost-Effective:** Pay-as-you-go EFS storage with no minimum commitments
6. **Automatic Failover:** If one server fails, others continue serving traffic
7. **Persistent Mounts:** EFS automatically mounts on instance reboot



This project successfully demonstrates the implementation of a production-ready, highly available web infrastructure on AWS using shared storage principles. By leveraging Amazon EFS across multiple EC2 instances, I created a scalable architecture that ensures content consistency and high availability while maintaining cost efficiency.