L1 – Project

Centralized File Sharing and Backup System

Content

Name: SAYELAKSHMI S

Introduction

The objective of this project is to design and implement a centralized file-sharing system on AWS using two EC2 instances connected to a shared Amazon EFS file system. Any file created, modified, or deleted on either EC2 instance is instantly updated in EFS, ensuring seamless collaboration. Additionally, a real-time sync mechanism using *inotifywait* continuously monitors the EFS directory and synchronizes all changes to an S3 bucket. This provides both high availability and secure backup storage for long-term retention.

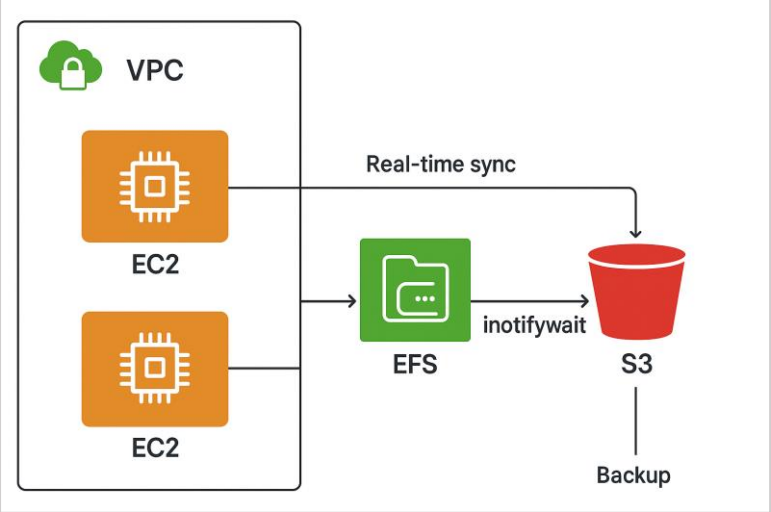
Objectives

The object of this project is to design and implement an AWS architecture where two EC2 instances, hosted in a custom VPC, share an Amazon EFS file system. Any file or directory created or modified inside the mount point **/ share/projects** on either instance must be **automatically** uploaded to an **Amazon S3 Bucket within Seconds**.

AWS Services Used

- Amazon EC2 – two Linux servers
- Amazon VPC – custom private network
- Amazon EFS – shared storage mounted on both EC2
- Amazon S3 – target storage bucket for sync
- AWS IAM – role with S3 Full Access for EC2 instances

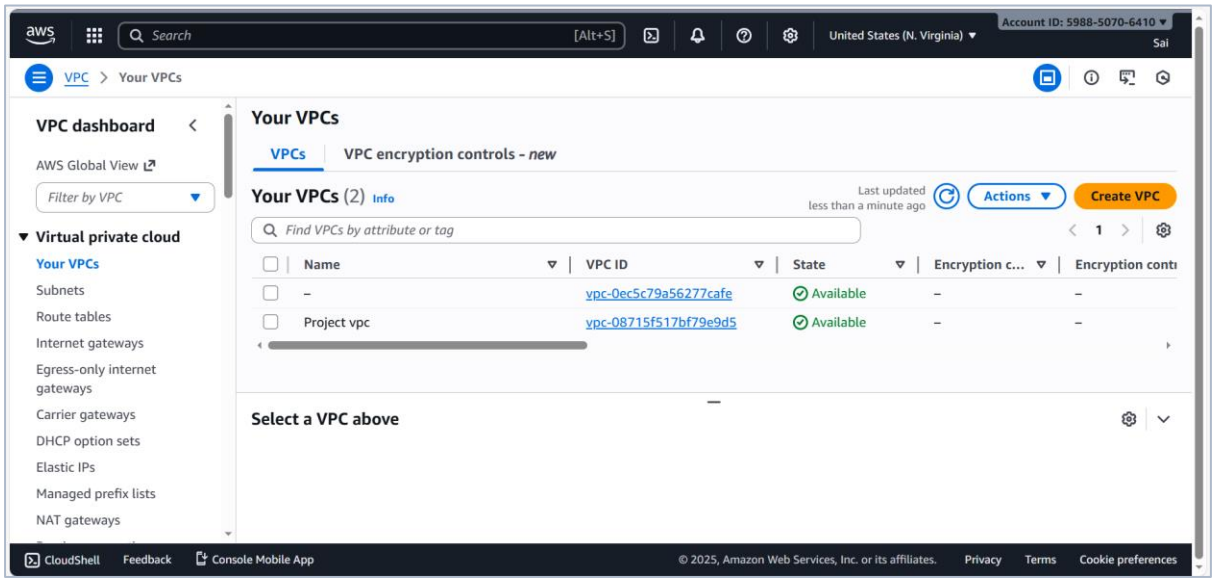
Architecture Diagram



Project Implementation setup:

Step -1 - Setup VPC & EC2 Instances.

1.Created a Custom VPC

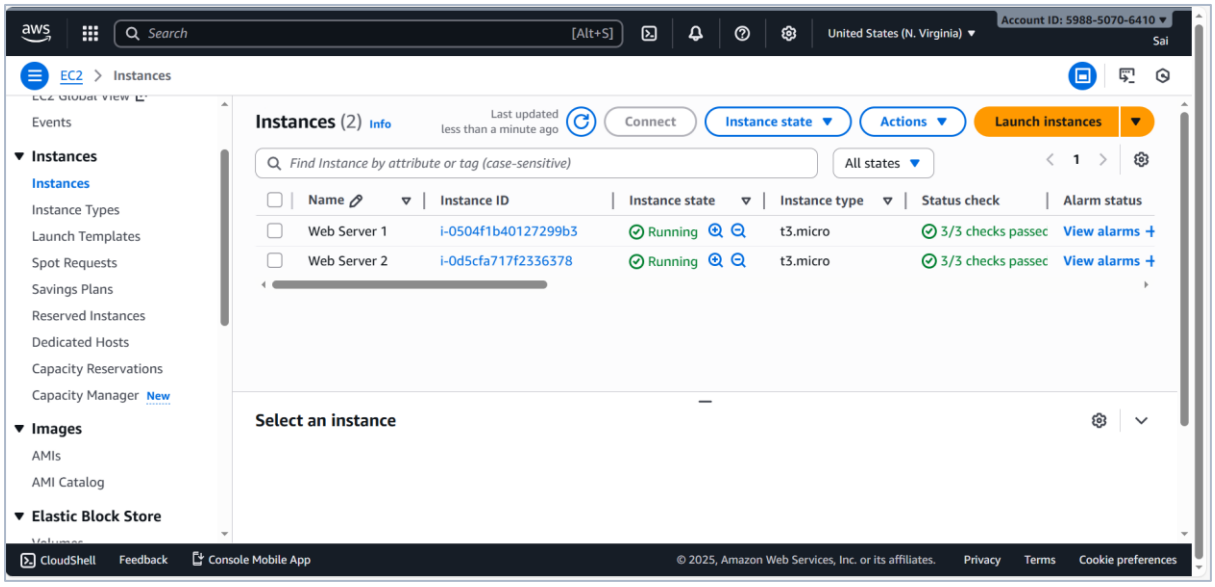


Launch **TWO EC2** instances in this VPC.

Choose AMI: Amazon Linux

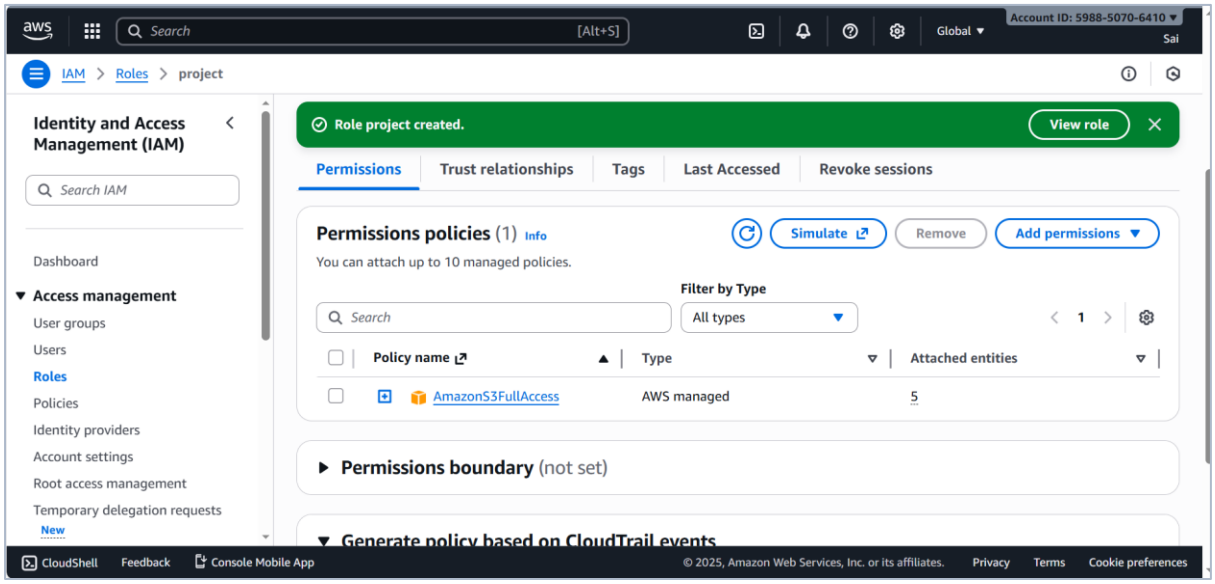
Created Web Server 1 in Subnet -1

And Web Server 2 in subnet 2

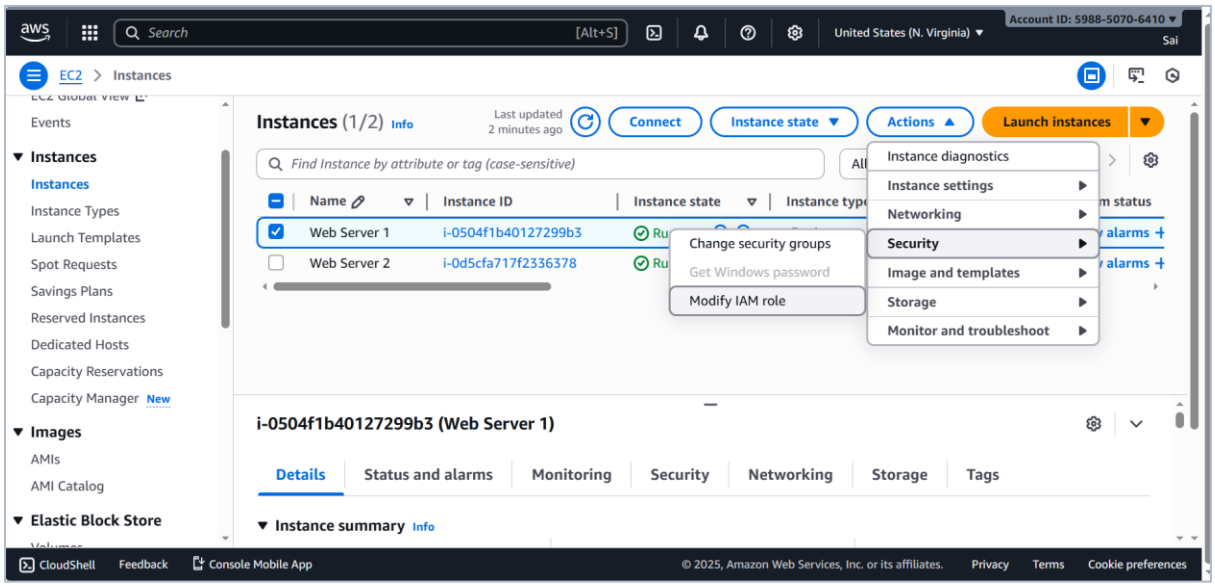


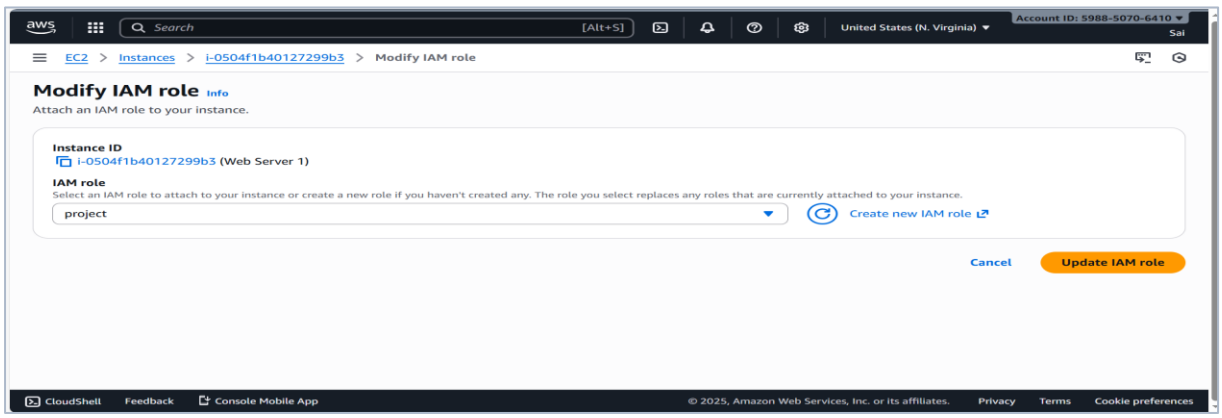
Step: 2 - Create IAM Role

Create IAM role with Amazon S3 Full access permission



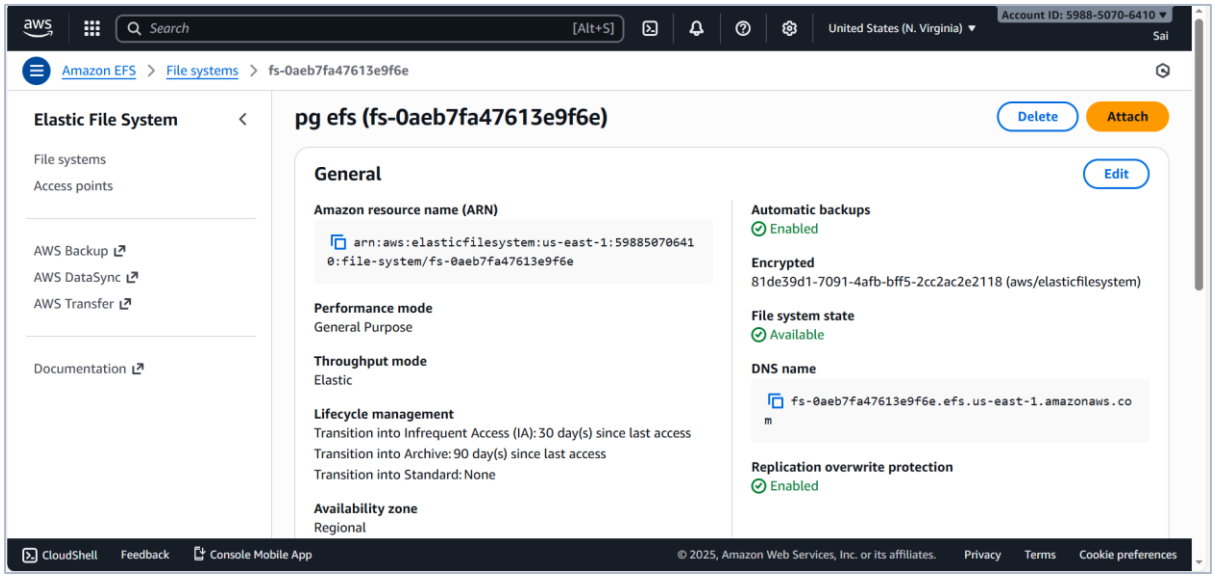
Attach the Role to two EC2 Server already created.





Step: 3 - Create & Mount EFS

1.Create an EFS File System in the same VPC



Create a mount target in the same subnet as yours EC2

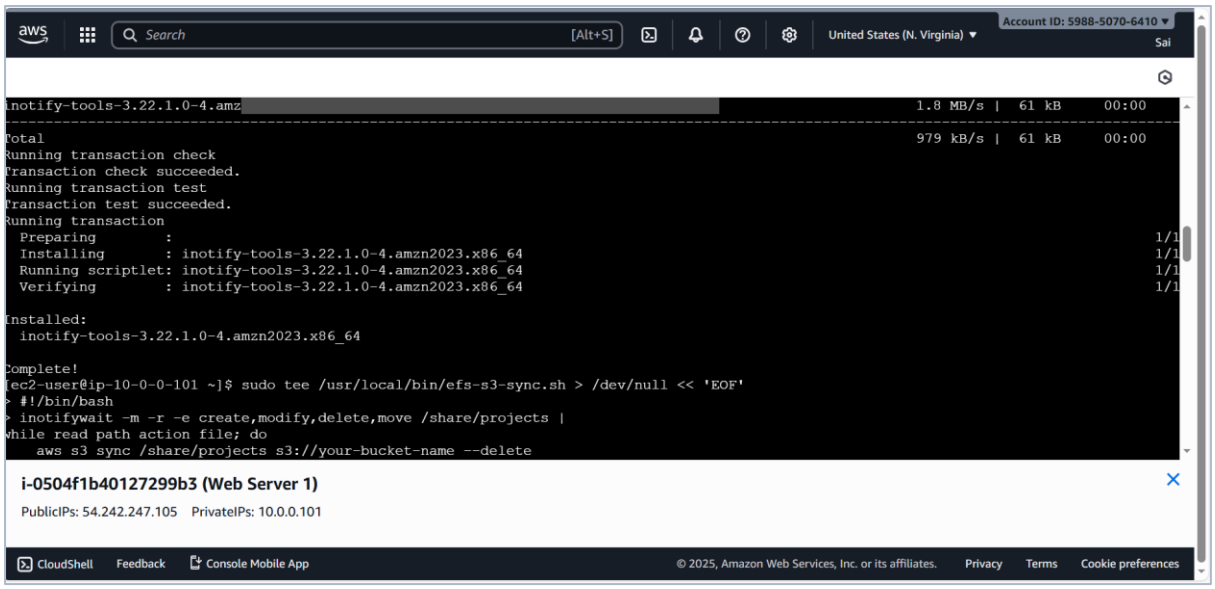
Mount EFS on both EC2 servers:

sudo yum install -y amazon-efs-utils

sudo mkdir -p /share/projects

sudo mount -t efs fs-0aeb7fa47613e9f6e:/ /share/projects

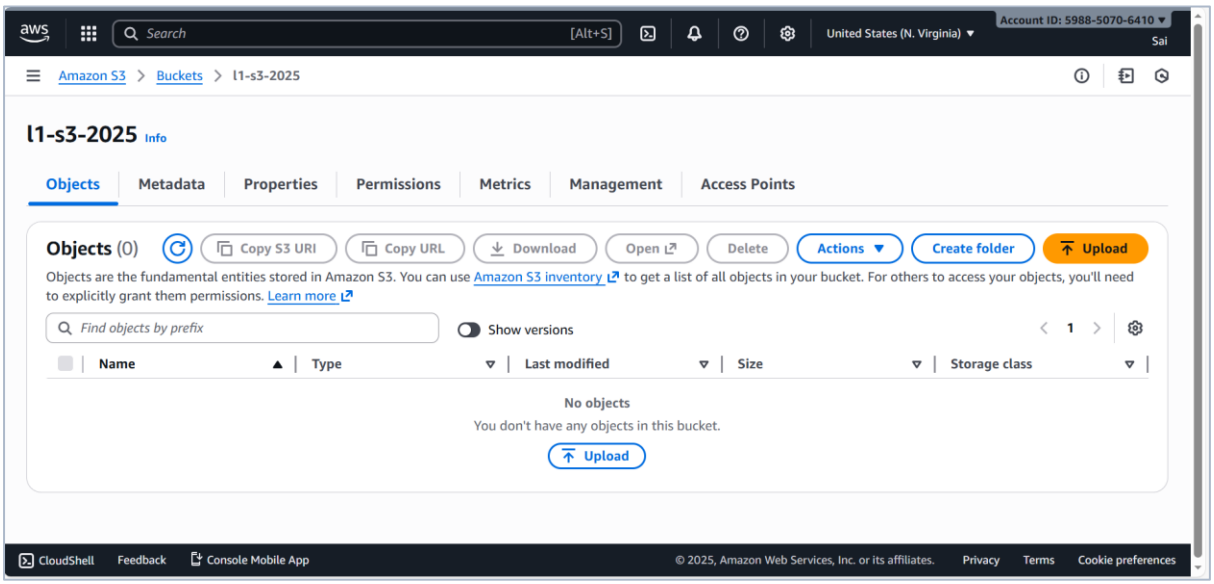
/share/projects will be our centralized folder.



Make the folder Writable:

Step 4: - Create S3 Bucket

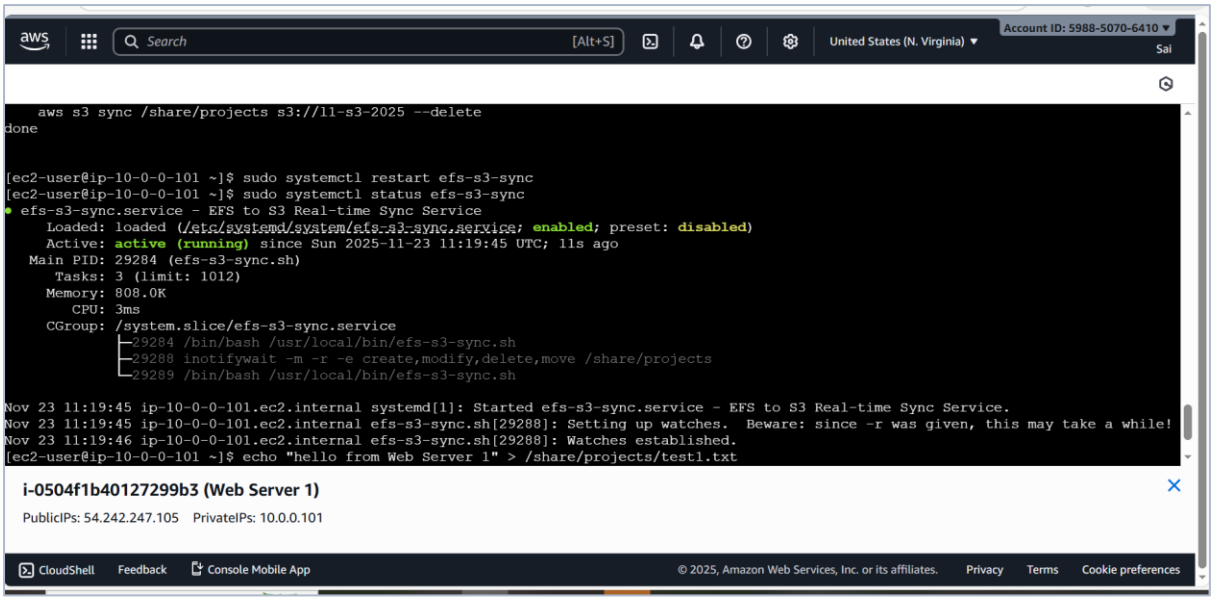
Bucket: s3:// **l1-s3-2025**



Make sure EC2 IAM Role can access it.

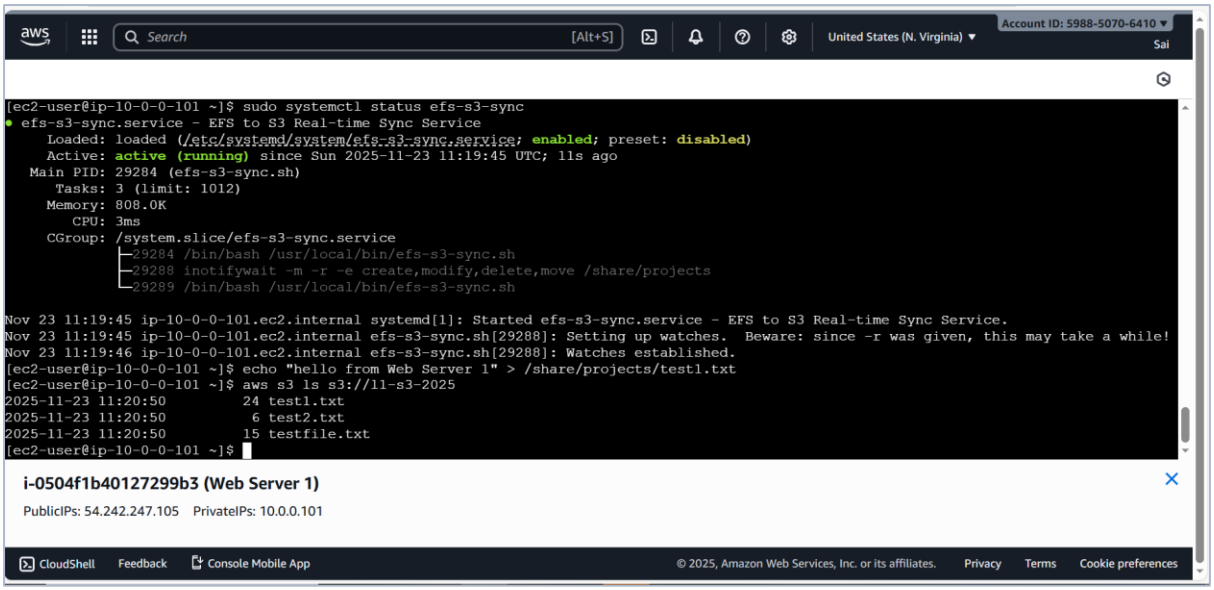
Step:5 - Install Dependencies

Sudo yum install -y inotify-tools



Ensure AWS CLI works with our IAM role:

aws s3 ls s3:// l1-s3-2025



Step: 6 - Create inotify sync script

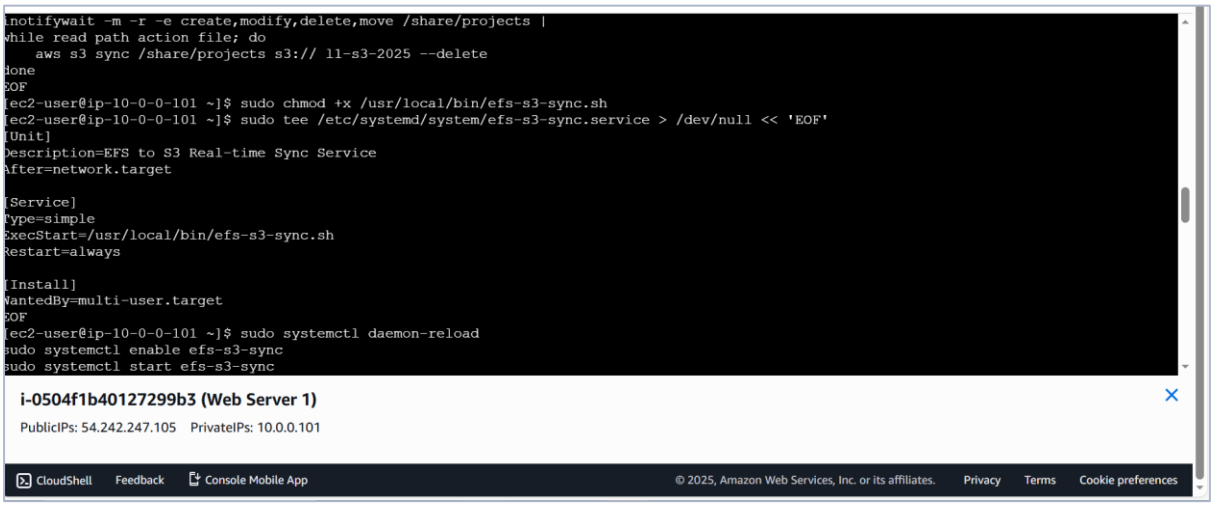
Implement a near real-time sync mechanism using inotify-based monitoring

1. Create the script file:

/usr/local/bin/efs-s3-sync.sh

Make it Executable:

chmod +x /home/ec2-user/efs-s3-watch.sh



Step:7 - Run the Script in Background

This script that syncs EFS → S3

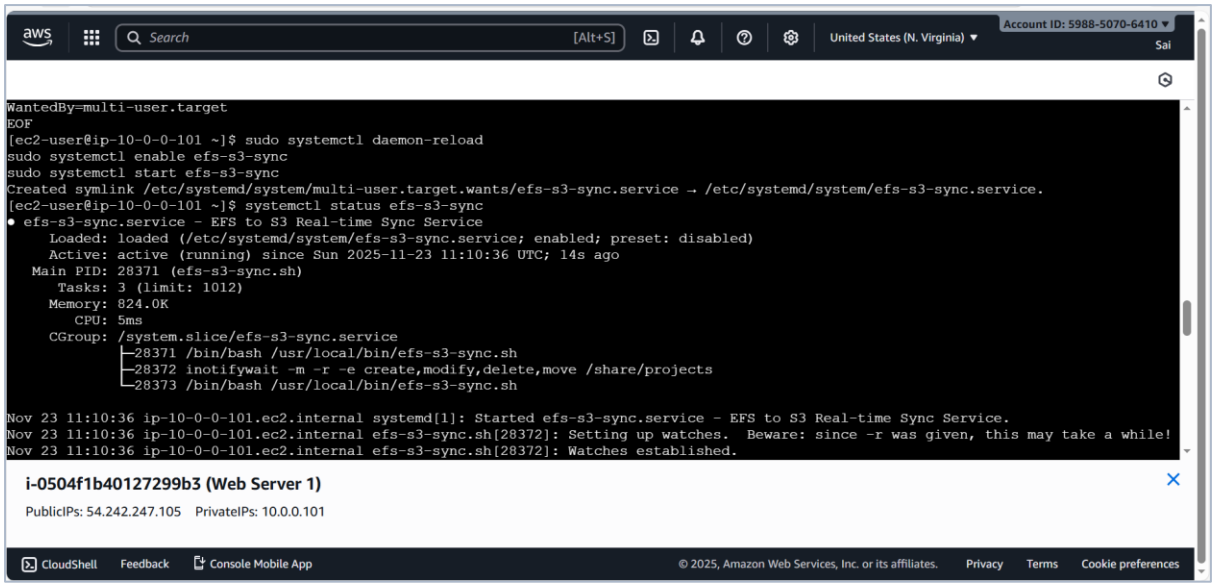
- /usr/local/bin/efs-s3-sync.sh

Runs the script in background after you log out

- /etc/systemd/system/efs-s3-sync.service

Check that it's running

- systemctl status efs-s3-sync



Step:8 - Test Automatic Sync

1.Create a file in EFS:

echo "hello from Web Server 1" > /share/projects/test1.txt

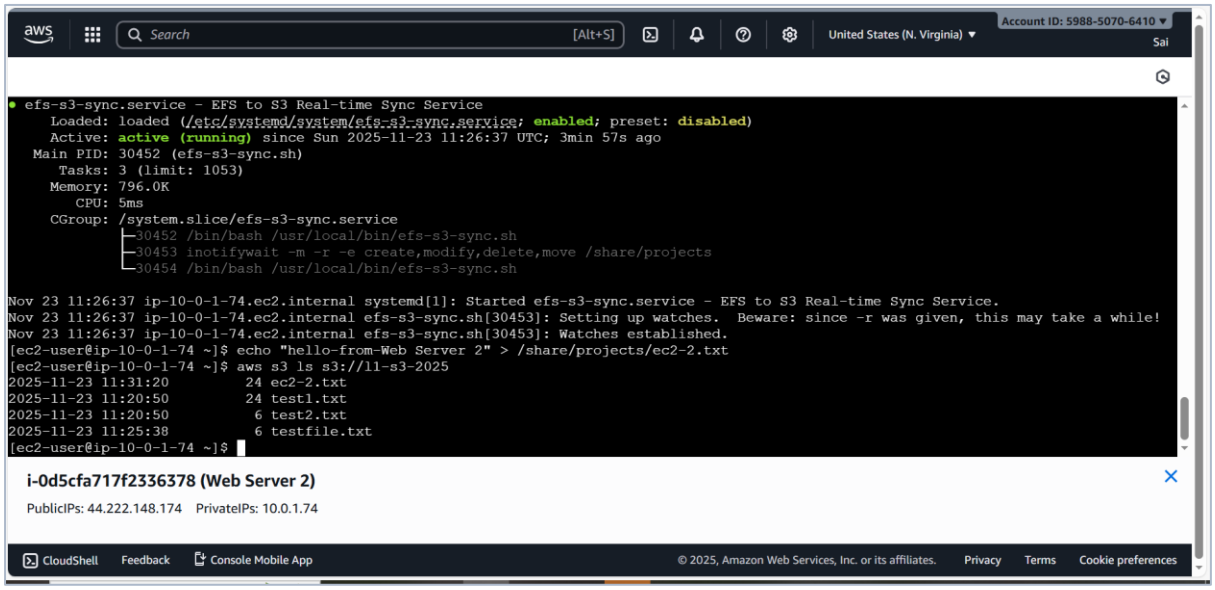
2. Wait a few seconds.

3. Check S3:

aws s3 ls s3://l1-s3-2025

- File should appear automatically.
- Check log: cat /var/log/efs-s3-sync.log

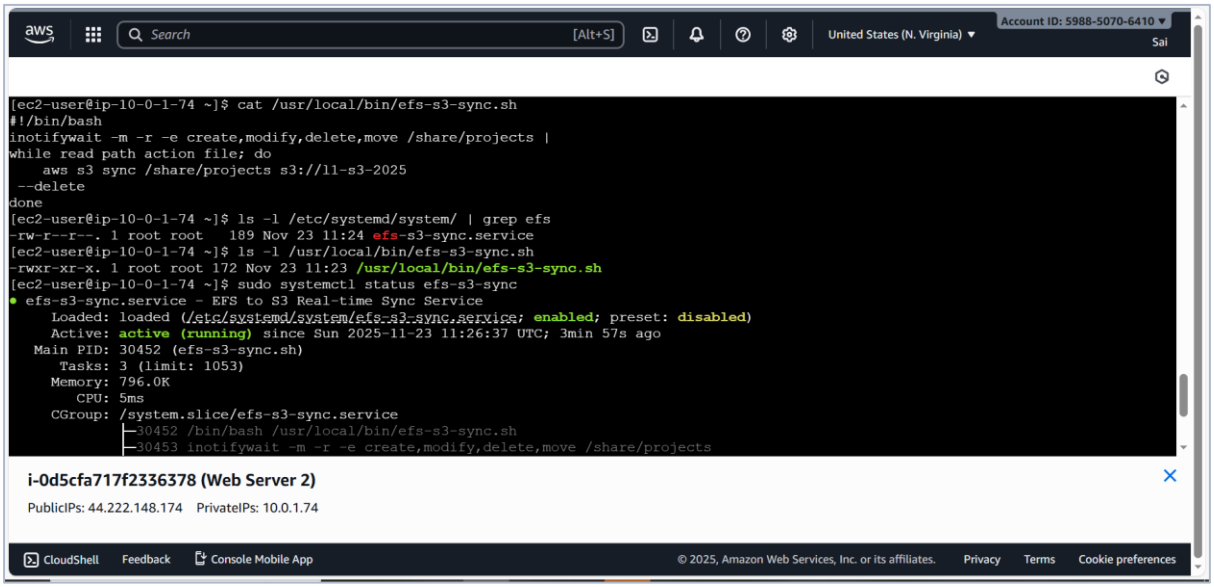
Step: 9 - Same command run into the Web Server 2 (EC2-2)



Step:10 - Make Script Start on Boot (systemd service)

1. Create a systemd service file:

ls -l /etc/systemd/system/ | grep efs



Enable auto-start at boot

- sudo systemctl enable efs-s3-sync

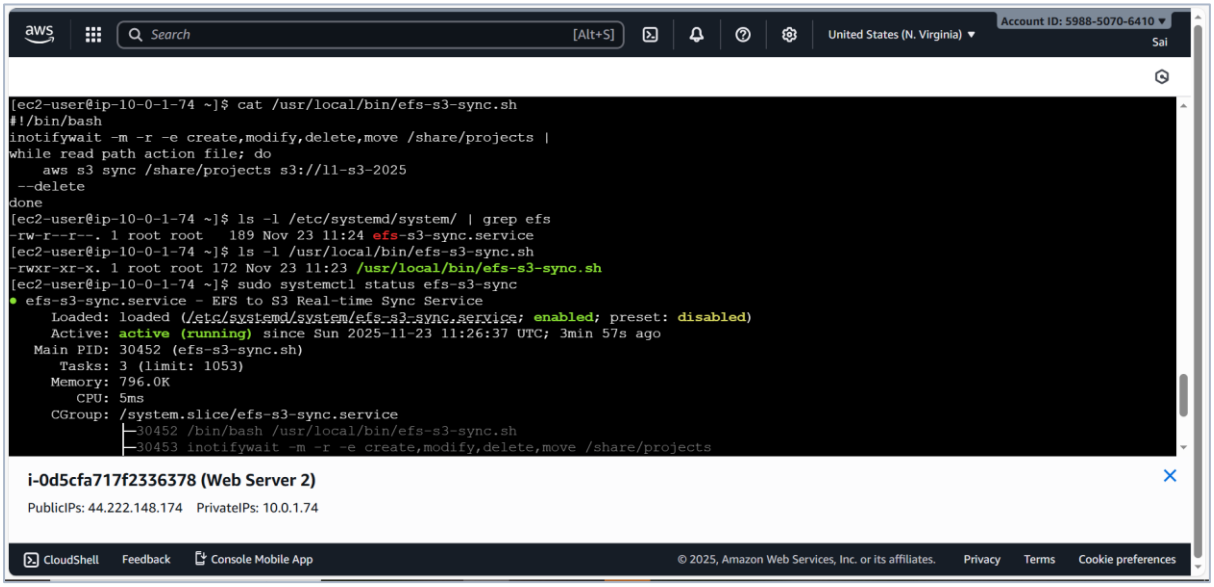
Start service now

- sudo systemctl start efs-s3-sync

Check if running

- systemctl status efs-s3-sync

Now the service is active (running),Now test file sync.



Result - The Instances automatically uploaded to an Amazon S3 Bucket within seconds.