

L1 -Project Report

Centralized File Sharing and Backup System

Contents

1. Introduction
2. Objectives
3. Services Used
4. Architecture
5. Project Implementation details
6. Output Screenshot

Project submitted by
Jayashree L

Centralized File Sharing and Backup System

1. Introduction:

The Centralized File Sharing and Backup System is designed to enable seamless collaboration between two EC2 instances by providing a shared storage environment using Amazon EFS. Both instances access the same directory structure, ensuring consistent project data. To maintain reliable backups and prevent data loss, all changes within the shared directory are automatically replicated to Amazon S3 within seconds.

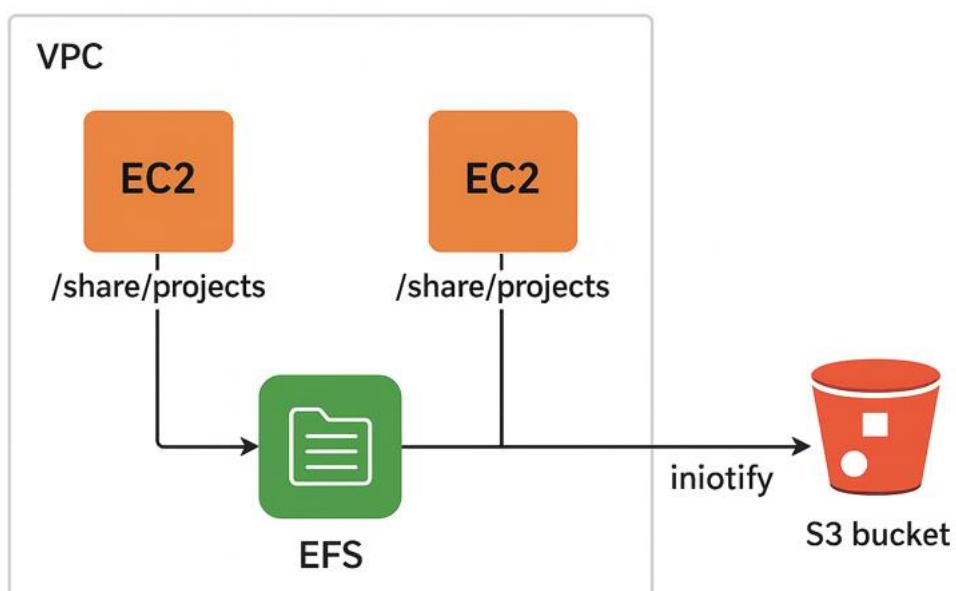
2. Objectives:

The objective 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.

3. Services used:

- Amazon EC2
- Amazon VPC
- Amazon EFS
- Amazon S3
- IAM

4. Architecture



5. Project Implementation details

Step 1: Setup VPC and EC2 Instances

1. Create a custom VPC

The screenshot shows the AWS VPC dashboard. On the left sidebar, under 'Virtual private cloud', 'Your VPCs' is selected. The main area displays 'Your VPCs (1/2)'. There are two entries:

Name	VPC ID	State	Block Public Access	IPv4 CIDR
vpc-03fdd26ccac8607f2	Available	Off	172.31.0.0/16	
myvpc789	Available	Off	10.0.0.0/16	

Below the table, a detailed view for 'myvpc789' is shown with tabs for Details, Resource map, CIDRs, Flow logs, Tags, and Integrations. The 'Details' tab is selected, showing the following information:

VPC ID	State	Block Public Access	DNS hostnames
vpc-030d3d25961a0eb9b	Available	Off	Enabled

2. Launch two EC2 instances in this VPC.

- Choose AMI: Amazon Linux
- Create instance1 in subnet -1
and instance 2 in subnet- 2.

The screenshot shows the AWS EC2 Instances page. On the left sidebar, 'Instances' is selected. The main area displays 'Instances (2)'. There are two entries:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
inst1	i-06409a7c7ded9a424	Running	t3.micro	Initializing	View alarms +
inst2	i-086057c2825080749	Running	t3.micro	Initializing	View alarms +

Step 2 : Create an IAM role

Create an IAM role with AmazonS3Fullaccess permission.

The screenshot shows the AWS IAM Role details page for 'EC2toS3proj'. The role allows EC2 instances to call AWS services on behalf of the user. It was created on November 18, 2025, at 17:03 (UTC+05:30). The ARN is arn:aws:iam::952421619651:role/EC2toS3proj. The instance profile ARN is arn:aws:iam::952421619651:instance-profile/EC2toS3proj. The last activity was 13 minutes ago. The maximum session duration is 1 hour. The 'Permissions' tab is selected, showing one managed policy attached. Other tabs include Trust relationships, Tags, Last Accessed, and Revoke sessions.

and attach the role to the two EC2 instances created already.

The screenshot shows the AWS EC2 Instances page. There are two instances listed: 'inst1' (i-06409a7c7ded9a424) and 'inst2' (i-086057c2825080749). The 'Actions' dropdown menu for 'inst1' is open, showing options like Change security groups, Get Windows password, and Modify IAM role. The 'Modify IAM role' option is highlighted. The 'Details' tab is selected for 'inst1', showing its instance ID, public and private IP addresses, and other basic details.

The screenshot shows the 'Modify IAM role' dialog for instance 'inst1'. The 'Instance ID' is set to i-06409a7c7ded9a424 (inst1). The 'IAM role' dropdown is set to 'EC2toS3proj'. A 'Create new IAM role' button is available if no role is selected. At the bottom, there are 'Cancel' and 'Update IAM role' buttons.

Step 3: Create and Mount EFS

1. Create an **EFS file system** in the same VPC.

The screenshot shows the 'File system settings' step of the EFS creation wizard. On the left, a navigation tree includes 'File system settings' (selected), 'Network access', 'File system policy', and 'Review and create'. The main area is titled 'File system settings' under 'General'. It shows a 'Name - optional' field containing 'EFSprojsyncS3' and a note that names can include up to 256 characters. Below it, the 'File system type' section offers two options: 'Regional' (selected) and 'One Zone'. 'Regional' is described as offering high availability and durability. 'One Zone' is described as providing continuous availability within a single Availability Zone. At the bottom, there's a note about automatic backups and a checkbox for 'Enable automatic backups'.

The screenshot shows the 'Elastic File System' details page for the file system 'EFSprojsyncS3 (fs-06318216ec4487de1)'. The left sidebar lists 'File systems' and 'Access points'. The main panel displays the file system's ARN (arn:aws:elasticfilesystem:us-east-1:95242161965:file-system/fs-06318216ec4487de1), performance mode (General Purpose), throughput mode (Elastic), lifecycle management (last access 7 days ago, archive none, standard on first access), availability zone (Regional), and other details like encryption and replication protection. Buttons for 'Delete' and 'Attach' are at the top right.

2. Create a **mount target** in the same subnet as your EC2s.

3. Mount EFS on both EC2 instances:

```
sudo yum install -y amazon-efs-utils # Amazon Linux
```

```
sudo mkdir -p /share/projects
```

```
sudo mount -t efs fs-xxxxxx:/ /share/projects
```

- /share/projects will be our **centralized folder**.

```

Total                                         35 MB/s | 4.9 MB   00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing      :
Installing    : stunnel-5.58-1.amzn2023.0.2.x86_64          1/1
Running scriptlet: stunnel-5.58-1.amzn2023.0.2.x86_64          1/2
Installing    : amazon-efs-utils-2.4.0-1.amzn2023.x86_64        1/2
Running scriptlet: amazon-efs-utils-2.4.0-1.amzn2023.x86_64        2/2
Verifying      : amazon-efs-utils-2.4.0-1.amzn2023.x86_64        2/2
Verifying      : stunnel-5.58-1.amzn2023.0.2.x86_64          2/2

Installed:
amazon-efs-utils-2.4.0-1.amzn2023.x86_64           stunnel-5.58-1.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-10-0-0-89 ~]$ sudo mkdir -p /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo mount -t efs fs-06318216ec4487del:/ /share/projects
[ec2-user@ip-10-0-0-89 ~]$ 
```

i-06409a7c7ded9a424 (inst1)
Public IPs: 34.207.56.168 Private IPs: 10.0.0.89

4. Make the folder writable:

```

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing      :
Installing    : stunnel-5.58-1.amzn2023.0.2.x86_64          1/1
Running scriptlet: stunnel-5.58-1.amzn2023.0.2.x86_64          1/2
Installing    : amazon-efs-utils-2.4.0-1.amzn2023.x86_64        1/2
Running scriptlet: amazon-efs-utils-2.4.0-1.amzn2023.x86_64        2/2
Verifying      : amazon-efs-utils-2.4.0-1.amzn2023.x86_64        2/2
Verifying      : stunnel-5.58-1.amzn2023.0.2.x86_64          2/2

Installed:
amazon-efs-utils-2.4.0-1.amzn2023.x86_64           stunnel-5.58-1.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-10-0-0-89 ~]$ sudo mkdir -p /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo mount -t efs fs-06318216ec4487del:/ /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo chown -R ec2-user:ec2-user /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo chmod -R 777 /share/projects
[ec2-user@ip-10-0-0-89 ~]$ 
```

i-06409a7c7ded9a424 (inst1)
Public IPs: 34.207.56.168 Private IPs: 10.0.0.89

Step 4: Create an S3 Bucket

- bucket: s3://amzn-projsync

The screenshot shows the Amazon S3 console with the following details:

- Breadcrumbs:** Amazon S3 > Buckets > amzn-projsync
- Bucket Name:** amzn-projsync
- General purpose buckets:**
 - Directory buckets
 - Table buckets
 - Vector buckets
 - Access Grants
 - Access Points (General Purpose Buckets, FSx file systems)
 - Access Points (Directory Buckets)
 - Object Lambda Access Points
 - Multi-Region Access Points
 - Batch Operations
 - IAM Access Analyzer for S3
- Actions:**
 - Objects (0)
 - Create folder
 - Upload (highlighted)
- Information:** Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)
- Search:** Find objects by prefix
- Sort:** Name, Type, Last modified, Size, Storage class
- Status:** No objects
- Upload:** A large blue button labeled "Upload".

- Make sure EC2 IAM role can access it.

Step 5: Install Dependencies

sudo yum install -y inotify-tools

```
[ec2-user@ip-10-0-0-89 ~]$ sudo mkdir -p /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo mount -t efs fs-06318216ec4487del:/ /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo chown -R ec2-user:ec2-user /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo chmod -R 777 /share/projects
[ec2-user@ip-10-0-0-89 ~]$ sudo yum install -y inotify-tools
Last metadata expiration check: 0:04:12 ago on Wed Nov 19 18:37:32 2025.
Dependencies resolved.

Transaction Summary
Install 1 Package

Total download size: 61 k
Installed size: 144 k
Downloading Packages:
inotify-tools-3.22.1.0-4.amzn2023.x86_64.rpm           1.6 MB/s | 61 kB     00:00

i-06409a7c7ded9a424 (inst1)
PublicIPs: 34.207.56.168 PrivateIPs: 10.0.0.89
```

which aws

- Ensure AWS CLI works with our IAM role:

aws s3 ls s3://amzn-projsync

```
aws
[ec2-user@ip-10-0-0-89 ~]$ Search [Alt+S] United States (N. Virginia) Account ID: 9524-2161-9651 jayashreewc17

Downloading Packages:
inotify-tools-3.22.1.0-4.amzn2023.x86_64.rpm           1.6 MB/s | 61 kB     00:00
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Installing : inotify-tools-3.22.1.0-4.amzn2023.x86_64 1/1
Running scriptlet: inotify-tools-3.22.1.0-4.amzn2023.x86_64 1/1
Verifying   : inotify-tools-3.22.1.0-4.amzn2023.x86_64 1/1

Installed:
inotify-tools-3.22.1.0-4.amzn2023.x86_64

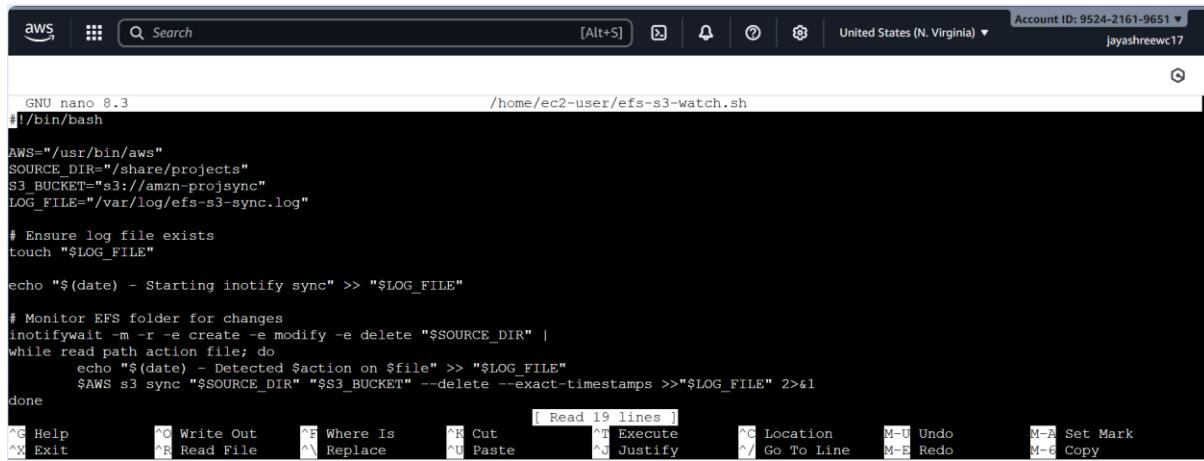
Complete!
[ec2-user@ip-10-0-0-89 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-0-89 ~]$ aws s3 ls s3://amzn-projsync
[ec2-user@ip-10-0-0-89 ~]$ i-06409a7c7ded9a424 (inst1)
PublicIPs: 34.207.56.168 PrivateIPs: 10.0.0.89
```

Step 6: Create Inotify Sync Script

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

1. Create the script file:

nano /home/ec2-user/efs-s3-watch.sh



```

GNU nano 8.3                               /home/ec2-user/efs-s3-watch.sh
#!/bin/bash

AWS="/usr/bin/aws"
SOURCE_DIR="/share/projects"
S3_BUCKET="s3://amzn-projsync"
LOG_FILE="/var/log/efs-s3-sync.log"

# Ensure log file exists
touch "$LOG_FILE"

echo "$(date) - Starting inotify sync" >> "$LOG_FILE"

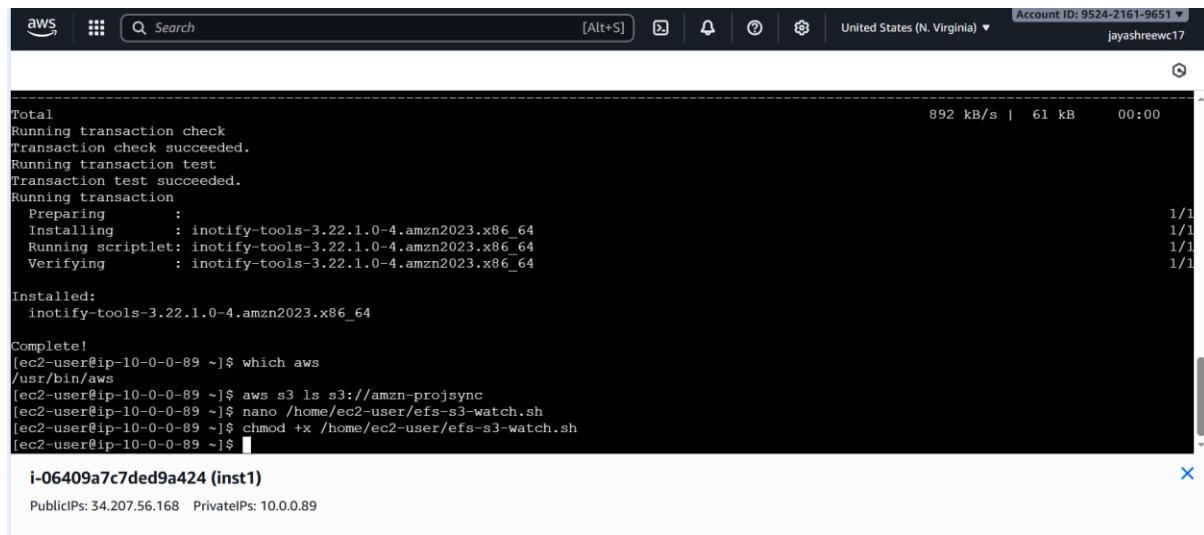
# Monitor EFS folder for changes
inotifywait -m -r -e create -e modify -e delete "$SOURCE_DIR" |
while read path action file; do
    echo "$(date) - Detected $action on $file" >> "$LOG_FILE"
    $AWS s3 sync "$SOURCE_DIR" "$S3_BUCKET" --delete --exact-timestamps >>"$LOG_FILE" 2>&1
done

```

^G Help ^O Write Out ^F Where Is ^R Cut ^T Execute ^C Location M-U Undo
 ^X Exit ^R Read File ^A Replace ^U Paste ^J Justify ^L Go To Line M-E Redo
 M-A Set Mark M-G Copy

2. Make it executable:

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



```

aws  Search Account ID: 9524-2161-9651
[Alt+S] United States (N. Virginia) jayashreewc17

Total                                         892 kB/s | 61 kB   00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing :                                                 1/1
  Installing : inotify-tools-3.22.1.0-4.amzn2023.x86_64      1/1
  Running scriptlet: inotify-tools-3.22.1.0-4.amzn2023.x86_64  1/1
  Verifying  : inotify-tools-3.22.1.0-4.amzn2023.x86_64       1/1

Installed:
  inotify-tools-3.22.1.0-4.amzn2023.x86_64

Complete!
[ec2-user@ip-10-0-0-89 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-0-89 ~]$ aws s3 ls s3://amzn-projsync
[ec2-user@ip-10-0-0-89 ~]$ nano /home/ec2-user/efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ chmod +x /home/ec2-user/efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ 
```

i-06409a7c7ded9a424 (inst1)

PublicIPs: 34.207.56.168 PrivateIPs: 10.0.0.89

Step 7: Run the Script in Background

nohup /usr/bin/bash /home/ec2-user/efs-s3-watch.sh &

- This keeps it running after you log out.
- Check that it's running:

ps aux | grep efs-s3-watch.sh

```

Installing      : inotify-tools-3.22.1.0-4.amzn2023.x86_64
Running scriptlet: inotify-tools-3.22.1.0-4.amzn2023.x86_64
Verifying       : inotify-tools-3.22.1.0-4.amzn2023.x86_64

Installed:
  inotify-tools-3.22.1.0-4.amzn2023.x86_64

Complete!
[ec2-user@ip-10-0-0-89 ~]$ which aws
/usr/bin/aws
[ec2-user@ip-10-0-0-89 ~]$ aws s3 ls s3://amzn-projsync
[ec2-user@ip-10-0-0-89 ~]$ nano /home/ec2-user/efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ chmod +x /home/ec2-user/efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ nohup /usr/bin/bash /home/ec2-user/efs-s3-watch.sh &
[1] 28581
[ec2-user@ip-10-0-0-89 ~]$ nohup: ignoring input and appending output to 'nohup.out'

[ec2-user@ip-10-0-0-89 ~]$ ps aux | grep efs-s3-watch.sh
ec2-user 28581 0.0 0.3 222964 3400 pts/0 S 18:55 0:00 /usr/bin/bash /home/ec2-user/efs-s3-watch.sh
ec2-user 28585 0.0 0.1 222964 1668 pts/0 S 18:55 0:00 /usr/bin/bash /home/ec2-user/efs-s3-watch.sh
ec2-user 28656 0.0 0.2 222336 2236 pts/0 S+ 18:55 0:00 grep --color=auto efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ \q

```

i-06409a7c7ded9a424 (inst1)

Public IPs: 34.207.56.168 Private IPs: 10.0.0.89

Step 8: Test Automatic Sync

1. Create a file in EFS:

```
echo "Hello from EC2-1" > /share/projects/testfile1.txt
```

2. Wait a few seconds.
3. Check S3:

```
aws s3 ls s3://amzn-projsync
```

- File should appear automatically.
- Check log:

```
cat /var/log/efs-s3-sync.log
```

Step 9: Repeat on Second EC2 Instance

- Mount the same EFS on EC2-2.
- Run the **same inotify script** there.

```

[ec2-user@ip-10-0-1-4 ~]$ sudo mkdir -p /share/projects
[ec2-user@ip-10-0-1-4 ~]$ sudo mount -t efs fs-06318216ec4487de1:/ /share/projects
[ec2-user@ip-10-0-1-4 ~]$ sudo chown -R ec2-user:ec2-user /share/projects
[ec2-user@ip-10-0-1-4 ~]$ sudo chmod -R 777 /share/projects
[ec2-user@ip-10-0-1-4 ~]$ sudo yum install -y inotify-tools
Last metadata expiration check: 0:05:19 ago on Wed Nov 19 18:37:35 2025.
Dependencies resolved.

=====
 Package          Architecture      Version           Repository      Size
 =
 
 Installing:
  inotify-tools      x86_64          3.22.1.0-4.amzn2023      amazonlinux      61
 k

 Transaction Summary
 =====
 Install 1 Package

i-086057c2825080749 (inst2)
Public IPs: 35.170.78.67 Private IPs: 10.0.1.4

```

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

1. Create a systemd service file:

```
sudo nano /etc/systemd/system/efs-s3-sync.service
```

The screenshot shows a terminal window in the AWS CloudShell interface. The user is creating a systemd service file named `efs-s3-sync.service`. The terminal output shows the nano editor being used to edit the file, and then the contents of the file being displayed. The file contains the configuration for the EFS to S3 automatic sync service.

```
[ec2-user@ip-10-0-0-89 ~]$ nano /home/ec2-user/efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ chmod +x /home/ec2-user/efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ nohup /usr/bin/bash /home/ec2-user/efs-s3-watch.sh &
[1] 28581
[ec2-user@ip-10-0-0-89 ~]$ nohup: ignoring input and appending output to 'nohup.out'

[ec2-user@ip-10-0-0-89 ~]$ ps aux | grep efs-s3-watch.sh
ec2-user 28581 0.0 0.3 222964 3400 pts/0 S 18:55 0:00 /usr/bin/bash /home/ec2-user/efs-s3-watch.sh
ec2-user 28585 0.0 0.1 222964 1668 pts/0 S 18:55 0:00 /usr/bin/bash /home/ec2-user/efs-s3-watch.sh
ec2-user 28656 0.0 0.2 222336 2236 pts/0 S+ 18:55 0:00 grep --color=auto efs-s3-watch.sh
[ec2-user@ip-10-0-0-89 ~]$ Hello from EC2-1" > /share/projects/testfile1.txt
[ec2-user@ip-10-0-0-89 ~]$ aws s3 ls s3://amzn-projsync
[ec2-user@ip-10-0-0-89 ~]$ cat /var/log/efs-s3-sync.log
cat: /var/log/efs-s3-sync.log: No such file or directory
[ec2-user@ip-10-0-0-89 ~]$ ^C
[ec2-user@ip-10-0-0-89 ~]$ sudo mkdir -p /var/log
[ec2-user@ip-10-0-0-89 ~]$ sudo touch /var/log/efs-s3-sync.log
[ec2-user@ip-10-0-0-89 ~]$ sudo chmod 666 /var/log/efs-s3-sync.log
[ec2-user@ip-10-0-0-89 ~]$ cat /var/log/efs-s3-sync.log
[ec2-user@ip-10-0-0-89 ~]$ LOGFILE="/var/log/efs-s3-sync.log"
[ec2-user@ip-10-0-0-89 ~]$ sudo nano /etc/systemd/system/efs-s3-sync.service
[ec2-user@ip-10-0-0-89 ~]$ i-06409a7c7ded9a424 (inst1)
PublicIPs: 34.207.56.168 PrivateIPs: 10.0.0.89
```

The screenshot shows a terminal window in the AWS CloudShell interface displaying the contents of the `efs-s3-sync.service` file. The file is a systemd unit file with the following configuration:

```
[Unit]
Description=EFS to S3 Automatic Sync
After=network.target
[Service]
Type=simple
User=ec2-user
ExecStart=/usr/bin/bash /home/ec2-user/efs-s3-watch.sh Restart=always [Install] WantedBy=multi-user.target
```

3. Enable and start:

The screenshot shows a terminal window in the AWS CloudShell interface. The user is enabling and starting the `efs-s3-sync.service` unit. The terminal output shows the commands being run and the resulting messages indicating the unit is loaded and active.

```
[ec2-user@ip-10-0-0-89 ~]$ sudo touch /var/log/efs-s3-sync.log
[ec2-user@ip-10-0-0-89 ~]$ sudo chmod 666 /var/log/efs-s3-sync.log
[ec2-user@ip-10-0-0-89 ~]$ cat /var/log/efs-s3-sync.log
[ec2-user@ip-10-0-0-89 ~]$ LOGFILE="/var/log/efs-s3-sync.log"
[ec2-user@ip-10-0-0-89 ~]$ sudo nano /etc/systemd/system/efs-s3-sync.service
[ec2-user@ip-10-0-0-89 ~]$ sudo systemctl daemon-reload
[ec2-user@ip-10-0-0-89 ~]$ sudo systemctl enable efs-s3-sync.service
The unit files have no installation config (WantedBy=, RequiredBy=, Also=, Alias= settings in the [Install] section, and DefaultInstance= for template units). This means they are not meant to be enabled using systemctl.
Possible reasons for having this kind of units are:
• A unit may be statically enabled by being symlinked from another unit's .wants/ or .requires/ directory.
• A unit's purpose may be to act as a helper for some other unit which has a requirement dependency on it.
• A unit may be started when needed via activation (socket, path, timer, D-Bus, udev, scripted systemctl call, ...).
• In case of template units, the unit is meant to be enabled with some instance name specified.
[ec2-user@ip-10-0-0-89 ~]$ sudo systemctl start efs-s3-sync.service
[ec2-user@ip-10-0-0-89 ~]$ sudo systemctl status efs-s3-sync.service
i-06409a7c7ded9a424 (inst1)
PublicIPs: 34.207.56.168 PrivateIPs: 10.0.0.89
```

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

```
Active: active (running) since Wed 2025-11-19 19:05:48 UTC; 7s ago
Main PID: 29637 (bash)
Tasks: 3 (limit: 1012)
Memory: 816.0K
CPU: 7ms
CGroup: /system.slice/efs-s3-sync.service
└─29637 /usr/bin/bash /home/ec2-user/efs-s3-watch.sh Restart=always "[Install]" WantedBy=multi-user.target
  ├─29644 inotifywait -m -r -e create -e modify -e delete /share/projects
  └─29647 /usr/bin/bash /home/ec2-user/efs-s3-watch.sh Restart=always "[Install]" WantedBy=multi-user.target

Nov 19 19:05:48 ip-10-0-0-89.ec2.internal systemd[1]: Started efs-s3-sync.service - EFS to S3 Automatic Sync.
Nov 19 19:05:48 ip-10-0-0-89.ec2.internal bash[29644]: Setting up watches. Beware: since -r was given, this may take a while!
Nov 19 19:05:48 ip-10-0-0-89.ec2.internal bash[29644]: Watches established.
[ec2-user@ip-10-0-0-89 ~]$ echo "test" > /share/projects/test.txt
[ec2-user@ip-10-0-0-89 ~]$ aws s3 ls s3://amzn-projsync
2025-11-19 19:07:28      5 test.txt
2025-11-19 19:07:27      17 testfile1.txt
[ec2-user@ip-10-0-0-89 ~]$ aws s3 ls s3://amzn-projsync
2025-11-19 19:07:28      5 test.txt
2025-11-19 19:10:09      5 test2.txt
2025-11-19 19:07:27      17 testfile1.txt
[ec2-user@ip-10-0-0-89 ~]$
```

i-06409a7c7ded9a424 (inst1)

PublicIPs: 34.207.56.168 PrivateIPs: 10.0.0.89

6. Output screenshots

Amazon S3 > Buckets > amzn-projsync

Objects (3)

Name	Type	Last modified	Size	Storage class
test.txt	txt	November 20, 2025, 00:37:28 (UTC+05:30)	5.0 B	Standard
test2.txt	txt	November 20, 2025, 00:40:09 (UTC+05:30)	5.0 B	Standard
testfile1.txt	txt	November 20, 2025, 00:37:27 (UTC+05:30)	17.0 B	Standard