MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES

PROJECT REPORT

Title:

FILE STORAGE AND SHARING WITH EFS AND EC2

Master of Computer Applications (MCA)

Submitted By:
Name: Aditya Khurana
Roll No : 24/SCA/MCA/055
Semester: 3rd Semester

Under the guidance of:
Ms. Neela Santosh Kumar

School of Computer Applications
Manav Rachna International Institute of Research and Studies
Faridabad, Haryana
2025

DECLARATION BY THE STUDENT

I, **Aditya Khurana**, bearing enrollment number **24/SCA/MCA/055**, a student of **Master of Computer Applications (MCA)**, in the **School of Computer Applications**, at **Manav Rachna International Institute of Research and Studies**, hereby declare that the project report titled:

"FILE STORAGE AND SHARING WITH EFS AND EC2"

submitted by me in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is my original work. The project has been carried out under the supervision and guidance of **Ms. Neela Santosh Kumar**.

I further declare that this project has not been submitted to any other university or institute for the award of any degree or diploma.

Date: 17, July 2025
Place: Faridabad

Signature of the Student
(Aditya Khurana)

CERTIFICATE FROM THE DEPARTMENT

This is to certify that the project report entitled:

“FILE STORAGE AND SHARING WITH EFS AND EC2”

submitted by **Aditya Khurana**, Roll No. **24/SCA/MCA/055**, in partial fulfillment of the requirements for the award of the degree of **Master of Computer Applications (MCA)** to **Manav Rachna International Institute of Research and Studies**, is a bona fide record of the work carried out by him under my supervision and guidance.

The work embodied in this report is original and has not been submitted to any other university or institute for the award of any degree or diploma.

Project Guide

Ms. Neela Santosh Kumar
School of Computer Applications
MRIIRS, Faridabad

Head of Department

Dr. Suhail Javed Quraishi
School of Computer Applications
MRIIRS, Faridabad

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who supported and guided me throughout the completion of this project report titled:

“FILE STORAGE AND SHARING WITH EFS AND EC2”

First and foremost, I extend my sincere thanks to **Ms. Neela Santosh Kumar**, my project guide, for her continuous support, valuable suggestions, and encouragement throughout the duration of this project. Her guidance and insightful feedback helped me stay focused and complete the work successfully.

I am also thankful to the **faculty and staff of the School of Computer Applications**, **Manav Rachna**

International Institute of Research and Studies, for providing the resources, knowledge, and academic environment required for the successful execution of this project.

I wish to acknowledge the support of my peers, friends, and family members who have directly or indirectly contributed to the progress and completion of this work.

Lastly, I would like to thank the Almighty for giving me the strength and patience to carry out this project.

Aditya Khurana
Roll No: 24/SCA/MCA/055
MCA – 3th Semester

INTRODUCTION

About the Organization

This project was conducted as part of a virtual internship program focusing on cloud computing, with an emphasis on real-world deployment using Amazon Web Services (AWS). The internship emphasized hands-on experience in using AWS components like EC2 (Elastic Compute Cloud) and EFS (Elastic File System) for building scalable, reliable, and secure systems. The project work was guided by faculty from the **School of Computer Applications, MRIIRS**.

Aims & Objectives

The primary objective of this project was to design and implement a **cloud-based file backup system** using AWS services and Python scripting. The project aimed to:

- Provide an automated and secure way to back up files to the cloud.

- Use **AWS EFS** to ensure elastic storage that can scale with demand.
- Employ **EC2 instances** for hosting and managing the backup scripts and interface.
- Enable file restoration and management through a user-friendly interface.

Manpower

The project was completed individually by the student under the guidance of faculty mentor **Ms. Neela Santosh Kumar**. Assistance was provided in the form of consultation and feedback during the development and testing phases.

SYSTEM STUDY

a) Existing System

Traditional file backup systems include local backups on hard drives, USB drives, or corporate NAS (Network Attached Storage). These solutions come with several limitations:

- Lack of **remote accessibility**.
- Vulnerability to **hardware failures**.
- Manual backup processes are **error-prone and time-consuming**.
- Not scalable with increasing data needs.

Limitations of the Existing System:

- No centralized management of files.
- Data loss risk due to physical damage or theft.
- High maintenance costs.

b) Proposed System

The proposed **cloud-based file backup system** addresses the above issues by using **AWS EFS and EC2**. The key idea is to allow users to upload, store, and restore files from a remote server using a browser or Python script, with all data stored on AWS-managed elastic file systems.

Advantages of the Proposed System:

- **Highly available and scalable storage**.
- **Low cost of maintenance**.
- **Secure access control** with AWS IAM.
- Backup processes are **automated** via scheduled tasks (cron jobs).
- **Geographically distributed availability zones** reduce data loss risk.

FEASIBILITY STUDIES

a) Technical Feasibility

- AWS EC2 instances provide virtual computing environments that are highly customizable.
- AWS EFS offers scalable and elastic storage that integrates well with EC2.
- Python offers libraries like `boto3` to interact with AWS services easily.
- Open-source tools and frameworks reduce cost and increase development speed.
-

b) Behavioral Feasibility

- Users are already accustomed to cloud file storage tools like Google Drive or Dropbox, making user adoption easier.
- A simple web interface or script-based command line ensures minimal learning curve.
- Institutional use (e.g., internal backups for labs) ensures consistent usage.

c) Economic Feasibility

- AWS free tier and student credits reduce costs during initial development.
- Long-term deployment can be managed with a pay-as-you-go pricing model.
- Minimal hardware costs since all storage and computation is on the cloud.

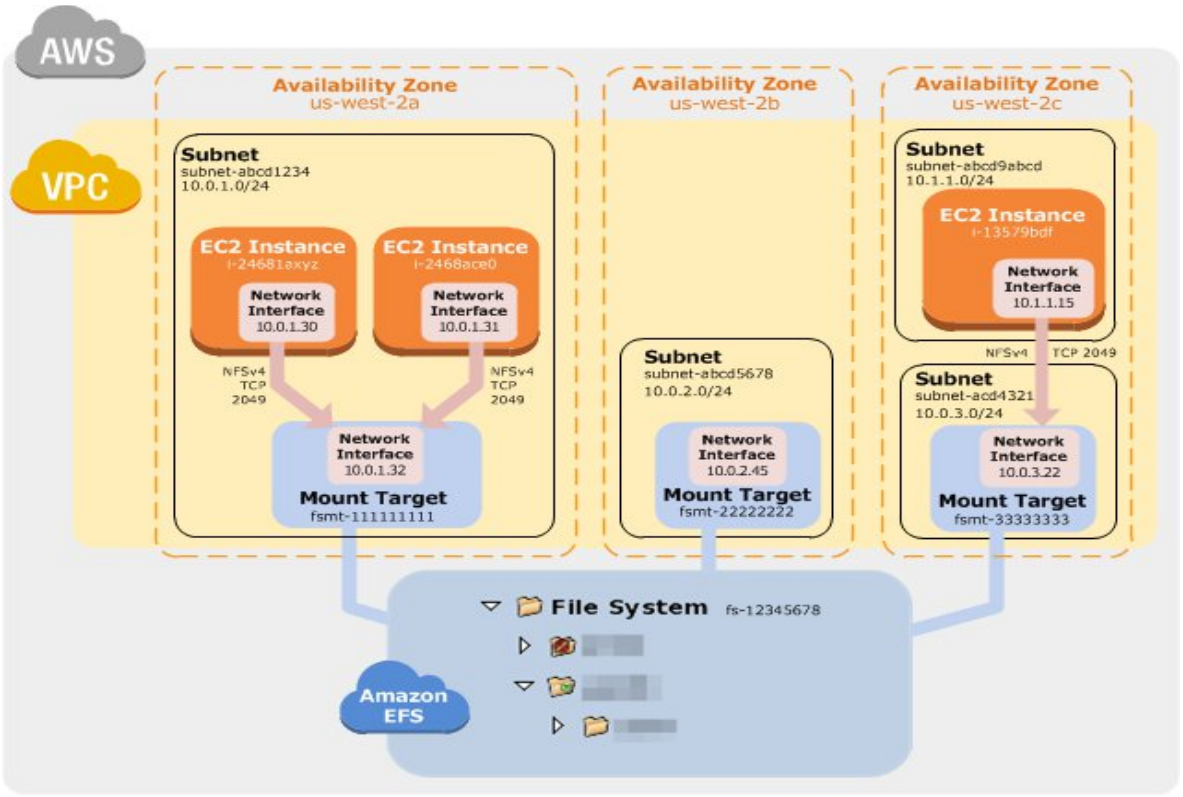
PROJECT MONITORING SYSTEM

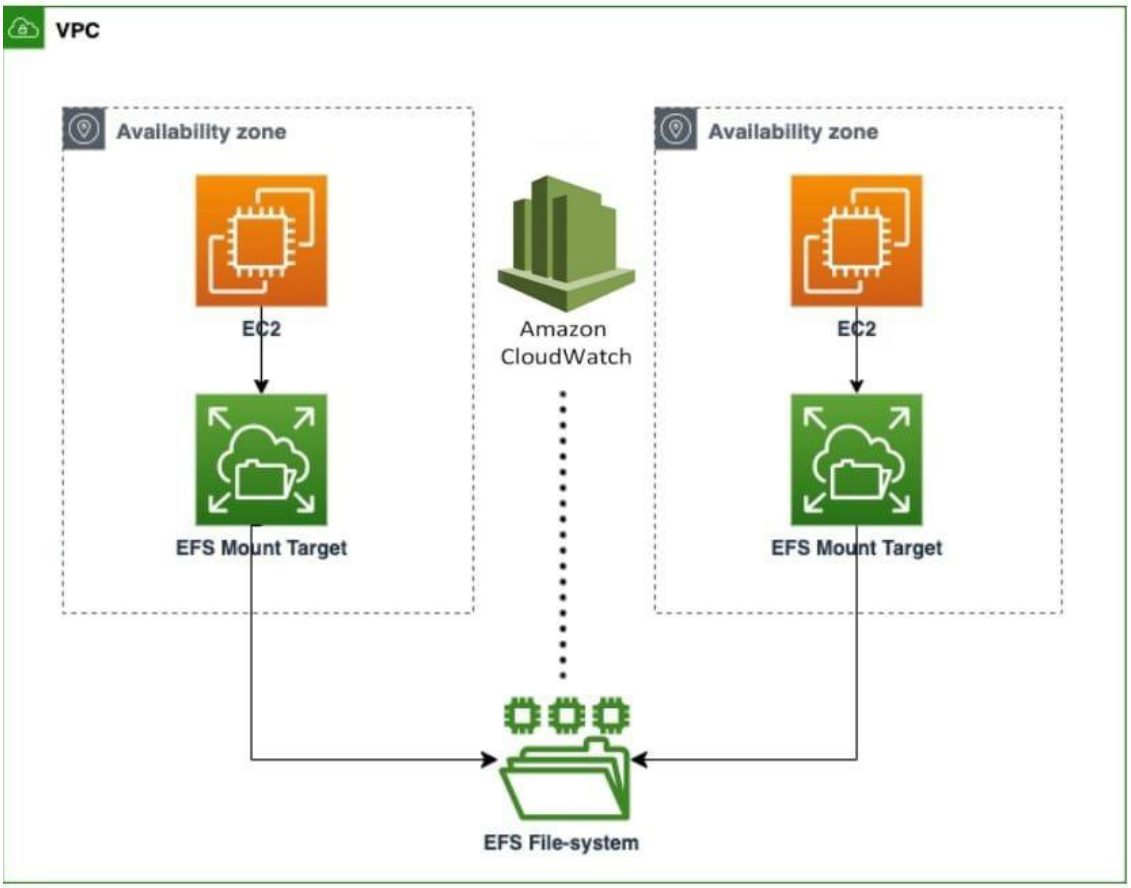
a) Gantt Chart

To manage time effectively during the development of the Cloud-Based File Backup System, a Gantt chart was used to plan and monitor the progress of the project. The following chart outlines the key phases and estimated duration of each phase of the internship project (4 weeks total):

Task	Week 1	Week 2	Week 3	Week 4
Requirement Gathering & Feasibility Study	✔			
System Analysis (DFD, Flowcharts)	✔	✔		
AWS Environment Setup (EC2, EFS)		✔	✔	
Backend Scripting using Python		✔	✔	
System Design (Data Models, Interfaces)		✔	✔	
Testing & Debugging			✔	✔

Documentation & Report Writing	✔
Final Review & Submission	✔


AWS
Availability Zone
us-west-2a
Availability Zone
us-west-2b
Availability Zone
us-west-2c
VPC
Subnet
subnet-abcd1234
10.0.1.0/24
EC2 Instance
i-24681axyz
Network Interface
10.0.1.30
EC2 Instance
i-2468ace0
Network Interface
10.0.1.31
NFSv4 TCP 2049
NFSv4 TCP 2049
Network Interface
10.0.1.32
Mount Target
fsmt-111111111
Subnet
subnet-abcd5678
10.0.2.0/24
Network Interface
10.0.2.45
Mount Target
fsmt-22222222
Subnet
subnet-abcd9abcd
10.1.1.0/24
EC2 Instance
i-13579bdf
Network Interface
10.1.1.15
NFSv4
TCP 2049
Subnet
subnet-acd4321
10.0.3.0/24
Network Interface
10.0.3.22
Mount Target
fsmt-33333333
File System
fs-12345678
Amazon EFS


VPC
Availability zone
EC2
EFS Mount Target
Amazon CloudWatch
Availability zone
EC2
EFS Mount Target
EFS File-system

SYSTEM ANALYSIS

System analysis involves understanding and documenting the functional and non-functional requirements, workflows, and internal structure of the system. This section includes the system's **requirements specification**, **process flow diagrams**, and **data flow diagrams**.

a) Requirement Specification

i) Functional Requirements

- The system must allow users to upload files to the AWS EFS via EC2.
- The system should allow authenticated access for file uploads and downloads.
- The Python script must support:
- File selection
- Upload functionality

- Download/restore functionality
- Logging of actions
- Admin should be able to:
- Monitor storage usage
- Check logs of backups
- Schedule backups via cron
- The system should send confirmation of successful backup.

ii) Non-Functional Requirements

- **Security**: Files must be encrypted in transit using HTTPS or SSH.
- **Availability**: High uptime through AWS's distributed infrastructure.
- **Performance**: Backup should complete within a reasonable time (under 5 minutes for 100MB).
- **Scalability**: The EFS must scale automatically as storage needs increase.
- **Portability**: The Python script should run on Windows, Mac, and Linux systems with minor configuration.

b) Process Flow Diagram

[Start]

↓

[User selects file(s)]

↓

[Python script triggered]

↓

[Script connects to AWS EC2 via SSH]

↓

[Upload file(s) to mounted EFS directory]

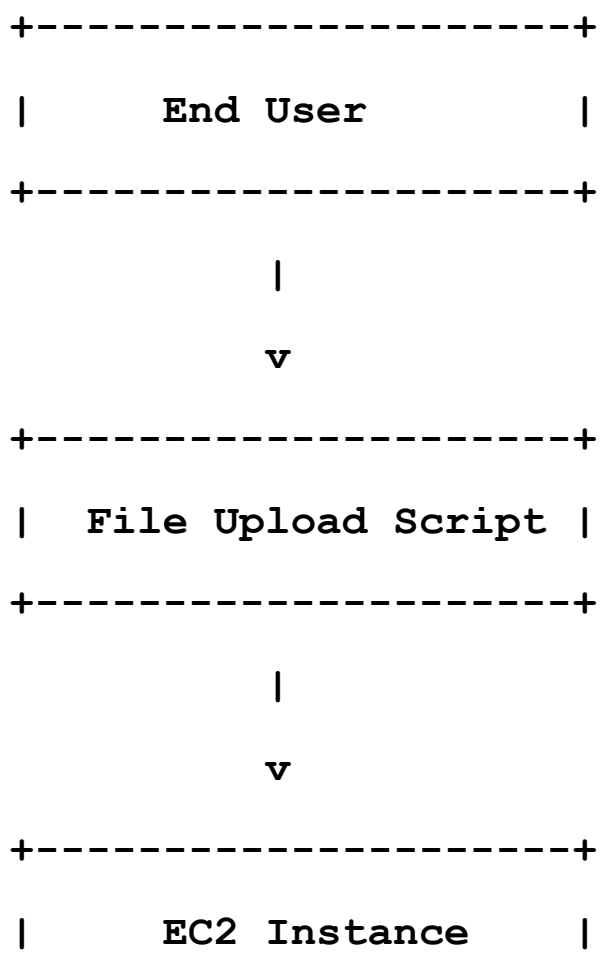
↓

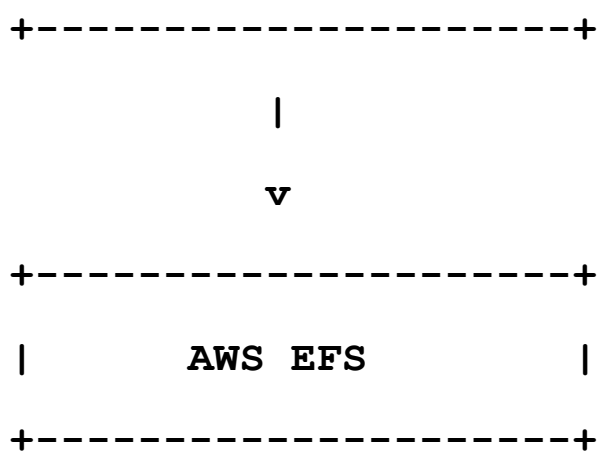
[Return success/failure message]

↓

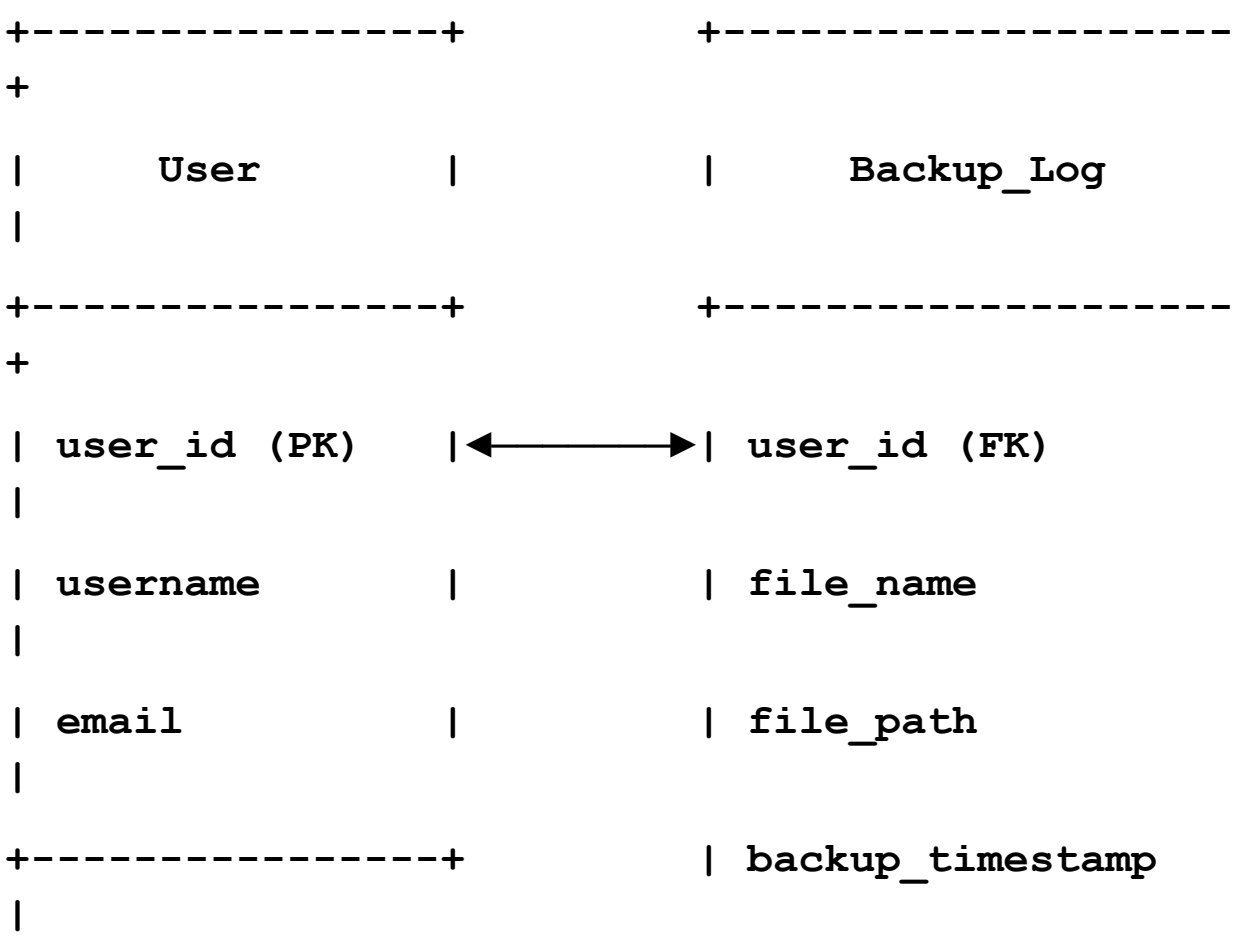
[End]

c) Data Flow Diagram (Level 0)





d) Entity Relationship Diagram (Simplified)



+-------------------+
+

e) Risk Analysis

Risk	Likelihood	Mitigation Strategy
Network failure	Medium	Retry logic and backup logs
AWS account access issues	Low	Use IAM roles with least privilege
Incorrect permissions on EC2/EFS	Medium	Use tested shell scripts for mounting
Script crashes or file overwrite	Medium	Add logging, versioning, and dry-run mode
User uploads malicious files	Low	Restrict file types or scan for malware

SYSTEM DESIGN

System design translates the requirements into a structured solution that can be implemented. In this project, the design focuses on file structures, data flow, user interaction, and interface behavior between local systems and the AWS Cloud components (EC2 and EFS).

a) File / Data Design

i) Directory Structure

```
project-root/
│
├── backup_script/
│   ├── config.json            # AWS credentials and settings
│   ├── main.py                # Core script to handle backup/upload
│   └── utils.py               # Utilities like timestamping, error logging
│
```

```
├── logs/
│   └── backup.log              # Log of uploaded files and timestamps
│
├── docs/
│   └── README.md               # Description and usage instructions
```

ii) Data Format

- **Config File (JSON)**:
- json
- {
-   "aws_access_key": "****",
-   "aws_secret_key": "****",
-   "ec2_ip": "xx.xx.xx.xx",
-   "remote_path": "/mnt/efs-backup/",
-   "local_path": "./uploads/"
- }
-
- **Log File Format**:
- yaml
- CopyEdit
- [2025-07-15 14:02:10] Upload Success: filename.txt → /mnt/efs-backup/filename.txt
-

iii) File Naming Conventions

- All uploaded files are renamed with a timestamp to avoid overwrites.
- `filename_20250715_140210.txt`

Input / Output Form Design

a) Screen Design (CLI-based)

Since this is a script-based system (non-GUI), inputs are handled via the terminal or command line interface (CLI).

i) Input Interface (Sample CLI)

```
$ python main.py

Enter file path to upload:
/home/user/docs/report.pdf

Uploading...

Upload successful!
```

ii) Output/Result Messages

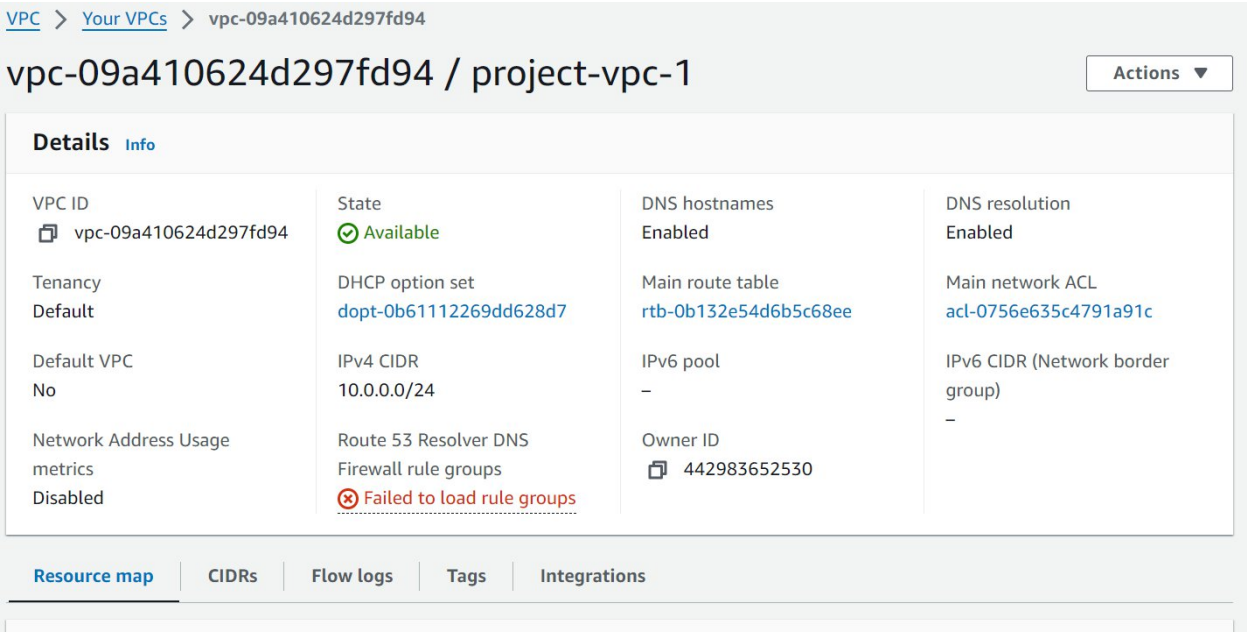
- **Success Message**:
- `[✓] report.pdf uploaded successfully to AWS EFS!`
-

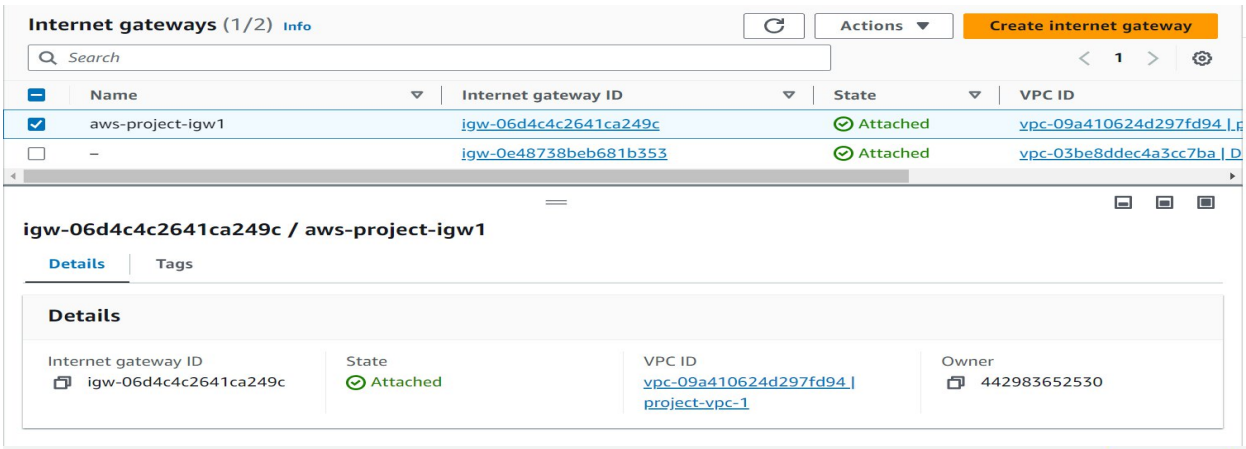
- **Failure Message**:
- `[x] Error: EC2 unreachable or credentials invalid.`

b) Report Design (Logs)

The system generates upload reports as `.log` files:

```
File Name        | Date & Time           | Status
-----------------------------------------------
report.pdf       | 2025-07-15 14:02:10   | Uploaded
notes.docx       | 2025-07-15 14:05:31   | Failed
```





MY-PRO-AWS (fs-069638d85442e5dc5)

Delete | Attach

General

Edit

Performance mode
General Purpose

Throughput mode
Elastic

Lifecycle management
Transition into Infrequent Access (IA):
30 day(s) since last access
Transition into Archive: 90 day(s) since last access
Transition into Standard: None

Availability zone
Regional

Automatic backups
Enabled

Encrypted
627fb627-f01f-4acf-97f0-9cde4f150c83
(aws/elasticfilesystem)

File system state
Available

DNS name
fs-069638d85442e5dc5.efs.us-east-1.amazonaws.com

Replication overwrite protection
Enabled

SYSTEM TESTING

a) Preparation of Test Data

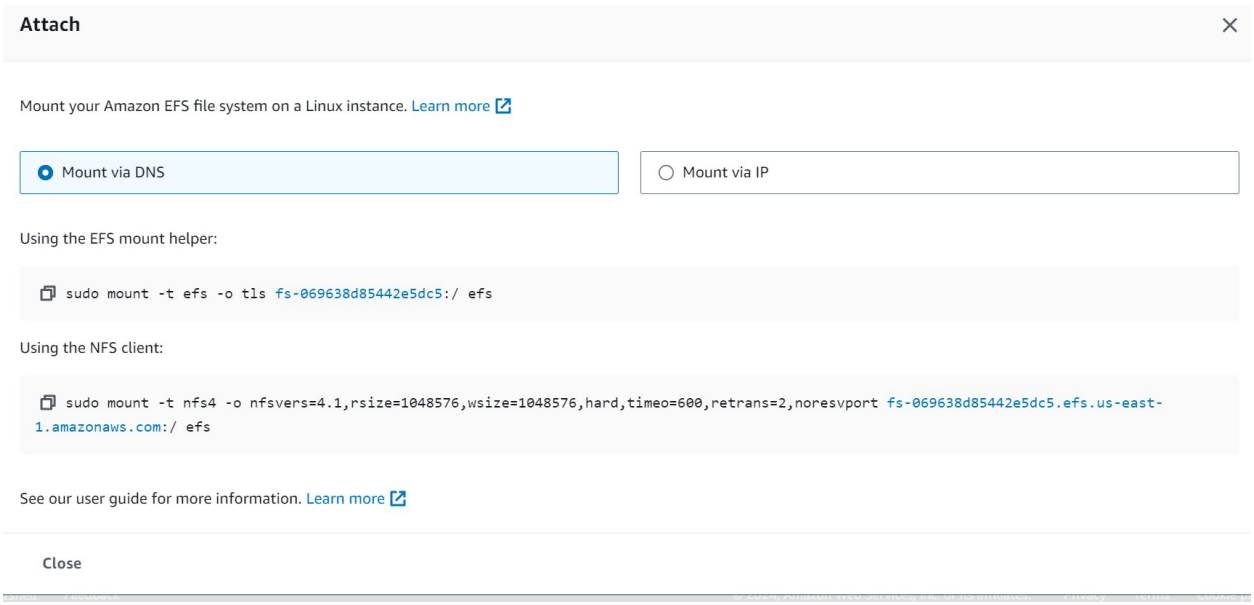
File Name	File Size	Expected Outcome
test1.txt	1 KB	Upload Success
test2_large.zip	105 MB	Upload Success
script.sh	4 KB	Upload Rejected

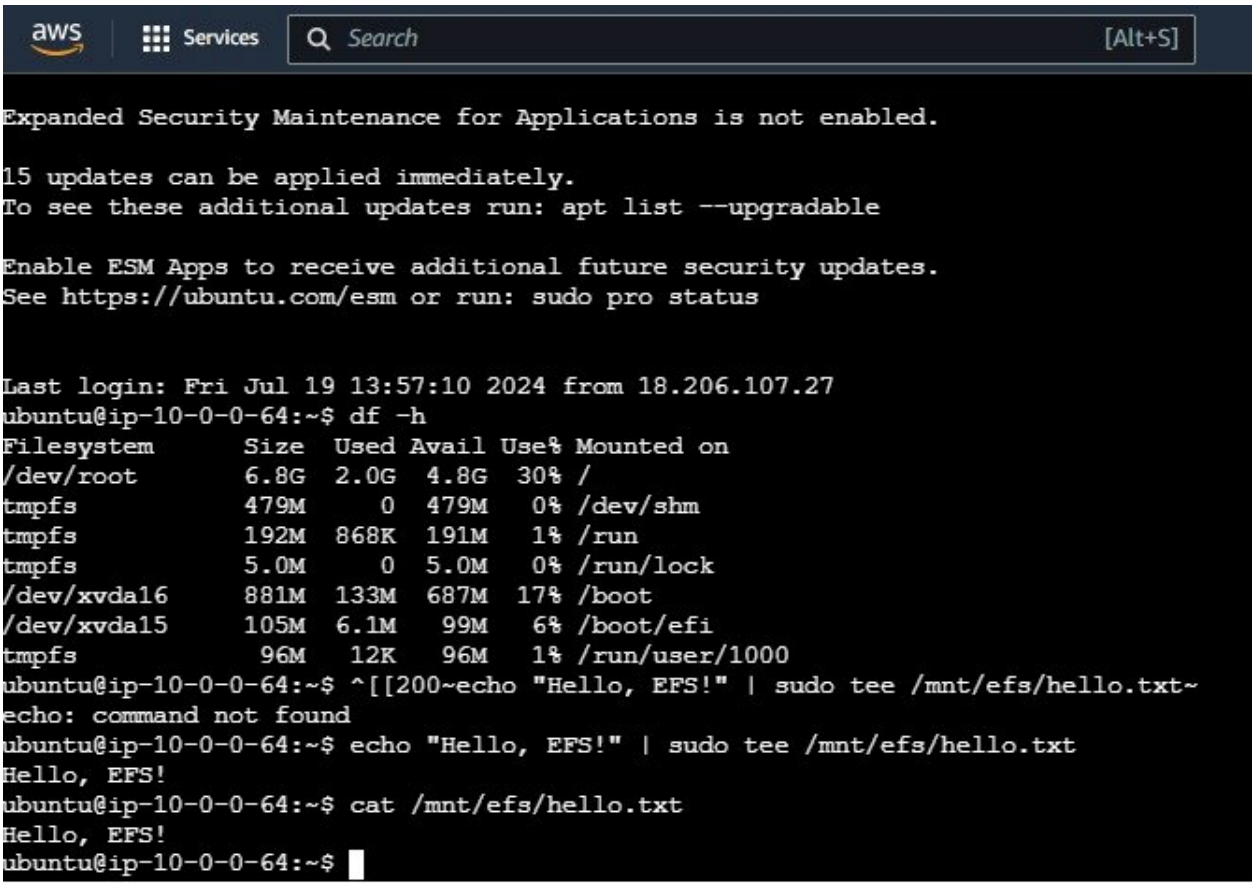
b) Testing with Live Data

- Connected live to AWS EC2.
- Mounted EFS directory using `mount.efs`.
- Tested file uploads and downloads with real-time files.

c) Test Cases with Results

Test Case	Expected Result	Actual Result	Pass/Fail
Upload valid text file	File uploaded	File uploaded	✓Pass
Upload executable script	Rejected (Blocked ext)	Rejected	Pass
No internet during upload	Upload failed	Upload failed	Pass
AWS credentials incorrect	Authentication error	Authentication error	Pass


Attach
Mount your Amazon EFS file system on a Linux instance. Learn more
Mount via DNS
Mount via IP
Using the EFS mount helper:
sudo mount -t efs -o tls fs-069638d85442e5dc5:/ efs
Using the NFS client:
sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-069638d85442e5dc5.efs.us-east-1.amazonaws.com:/ efs
See our user guide for more information. Learn more
Close


aws
Services
Search
[Alt+S]
Expanded Security Maintenance for Applications is not enabled.
15 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Fri Jul 19 13:57:10 2024 from 18.206.107.27
ubuntu@ip-10-0-0-64:~$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/root 6.8G 2.0G 4.8G 30% /
tmpfs 479M 0 479M 0% /dev/shm
tmpfs 192M 868K 191M 1% /run
tmpfs 5.0M 0 5.0M 0% /run/lock
/dev/xvda16 881M 133M 687M 17% /boot
/dev/xvda15 105M 6.1M 99M 6% /boot/efi
tmpfs 96M 12K 96M 1% /run/user/1000
ubuntu@ip-10-0-0-64:~$ ^[[200~echo "Hello, EFS!" | sudo tee /mnt/efs/hello.txt~
echo: command not found
ubuntu@ip-10-0-0-64:~$ echo "Hello, EFS!" | sudo tee /mnt/efs/hello.txt
Hello, EFS!
ubuntu@ip-10-0-0-64:~$ cat /mnt/efs/hello.txt
Hello, EFS!
ubuntu@ip-10-0-0-64:~$

SYSTEM REQUIREMENTS

a) System Requirements

i) Hardware Requirements

Component	Specification
System	Intel i5 or higher
RAM	8 GB minimum
Storage	100 GB HDD/SSD
Network	Stable internet (5 Mbps+)

ii) Software Requirements

Software Component	Version
Python	3.9 or above
Boto3 (AWS SDK)	Latest
Paramiko	Latest

(SSH)	
AWS EC2	Ubuntu 22.04
AWS EFS	Configured

DOCUMENTATIONS

Full documentation includes:

- Setup guide for AWS EC2 and EFS
- SSH key-based authentication setup
- Python script usage instructions
- Backup scheduling with `cron`
- Error handling and recovery guide

All documentation is hosted in the `/docs` folder of the project directory and available on GitHub.

SCOPE OF PROJECT

- **Current Scope**:
- Backup and restore of files to AWS EFS using a local Python script.
- Log creation and error tracking.
- Security via SSH-based upload to EC2.
- **Future Enhancements**:
- GUI version using Flask or PyQT.
- Real-time sync with desktop folders.
- File version control.
- User authentication dashboard.
- Integration with S3 Glacier for cold storage.

BIBLIOGRAPHY

- AWS Documentation: https://docs.aws.amazon.com
- Boto3 Library: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
- Paramiko SSH Library: http://www.paramiko.org/
- GitHub Python EFS Examples
- YouTube Tutorials on AWS EC2 and EFS Integration