Aim: Develop a Secure identity access management with back up and replicate options: Cloud apps open solution (IaaS, PaaS, and also in SaaS).

For a demo: Demo Video link

1. Introduction

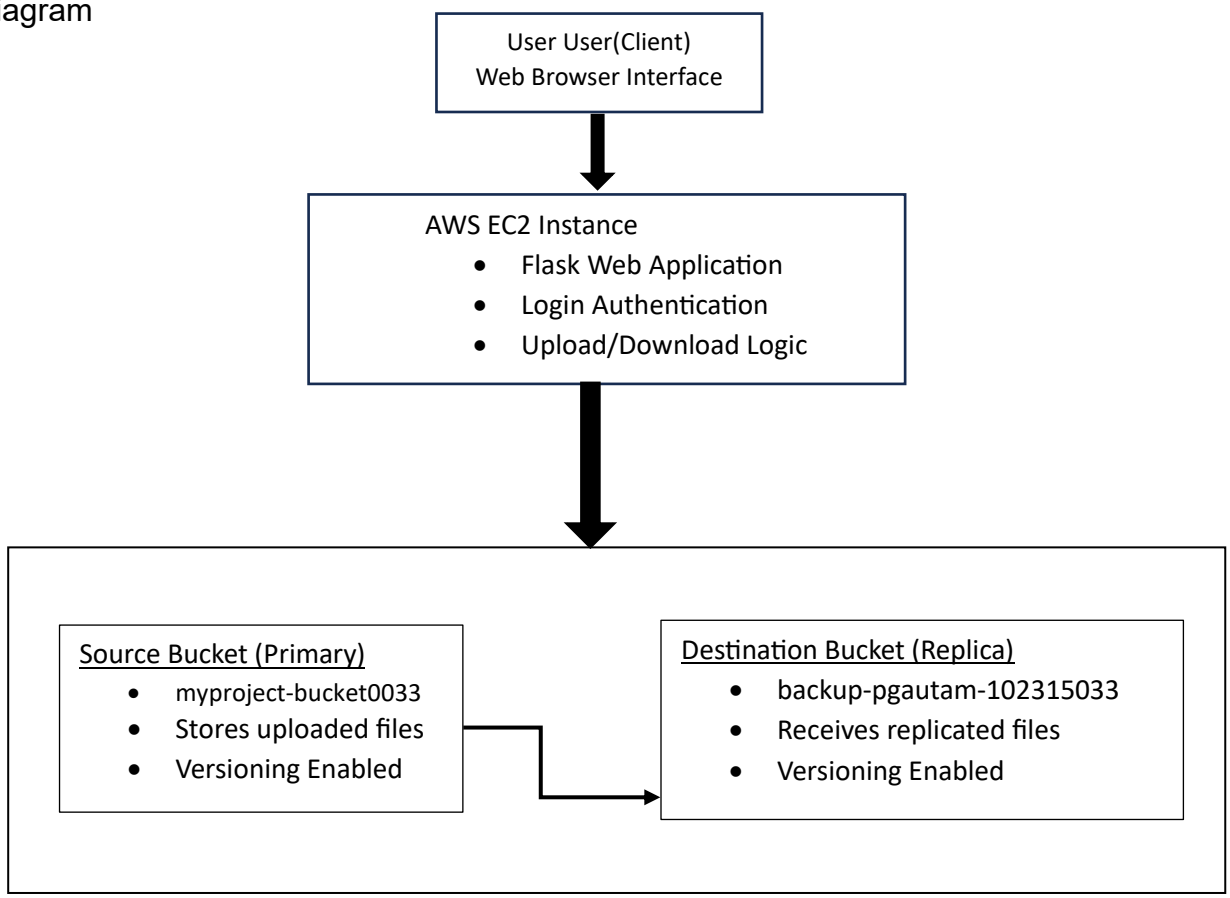
Cloud computing enables scalable, reliable, and cost-efficient access to computing resources over the internet. Among its services, cloud storage plays a crucial role in securely managing and preserving data. This project demonstrates a complete cloud-based file management system using AWS. It includes a web interface for uploading and downloading files, automatic backup through S3 replication, version control for maintaining multiple file revisions, and secure user authentication. The system is deployed on an EC2 instance, showcasing practical implementation of cloud infrastructure, storage, and application hosting.

2. Project Objectives

The primary objective of this project is to design and deploy a secure, cloud-based file management system using Amazon Web Services (AWS). The system aims to enable users to upload, store, retrieve, and manage files efficiently through a web interface. Key goals include implementing cloud storage using S3, enabling automatic backup through cross-bucket replication, maintaining multiple file versions for data protection, and securing access through user authentication. The project also focuses on deploying the application on an EC2 instance and validating its functionality through end-to-end testing.

3. System Architecture

- block diagram



- Components:
    - User
    - Web Interface (Flask)
    - EC2 Instance
    - S3 Source Bucket
    - S3 Backup Bucket
    - Replication Rule
    - Versioning

4. Tools & Technologies Used

4.1 Amazon Web Services (AWS)

- Amazon EC2: Used to host and run the Flask web application on a virtual Linux server.
- Amazon S3: Primary cloud storage service for storing uploaded files.
- S3 Replication: Automatically copies files from the source bucket to the backup bucket for redundancy.
- S3 Versioning: Maintains multiple versions of the same file (V1, V2, V3).
- IAM Roles & Policies: Provide secure access for the EC2 instance to interact with S3.

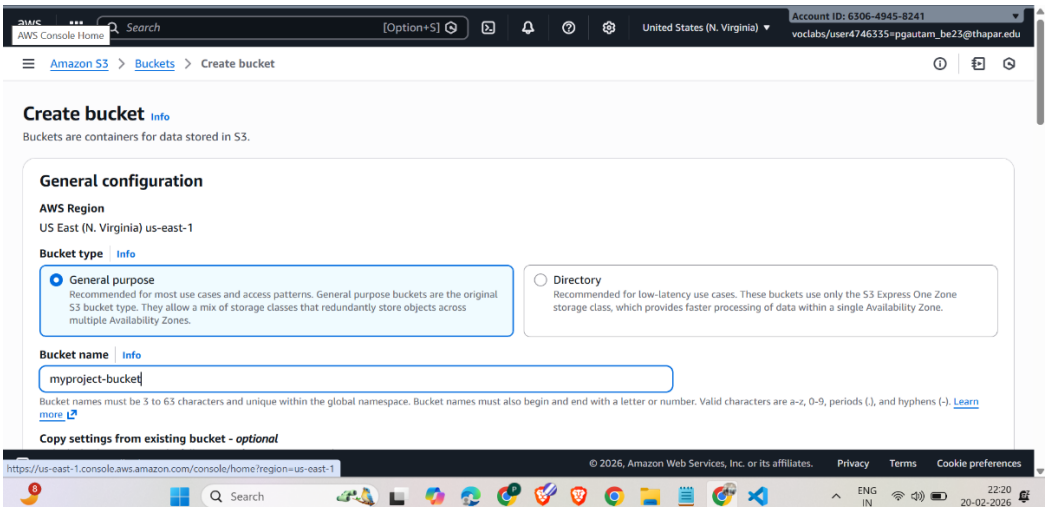
4.2 Backend Technologies

- Python 3: Core programming language used to build the application logic.
- Flask Framework: Lightweight web framework used to create the login page, upload/download interface, and backend routes.
- Boto3 SDK: AWS SDK for Python used to interact with S3 for uploading, downloading, and managing files.

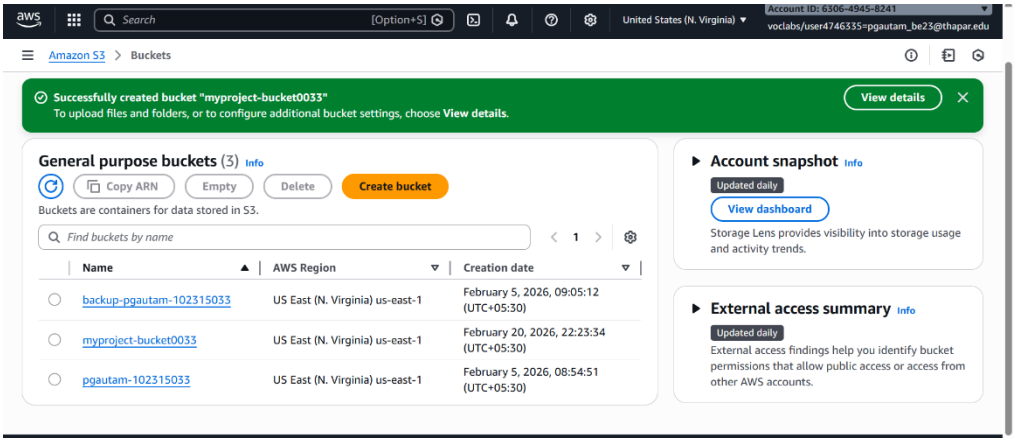
5. Implementation Steps

5.1 Create S3 Buckets

- Source bucket: myproject-bucket0033

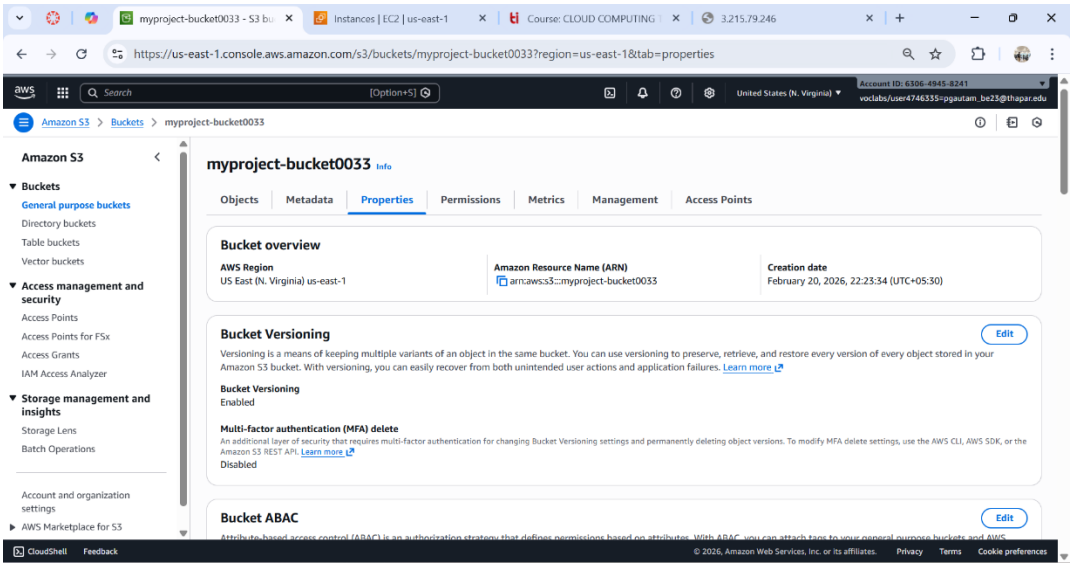


- Backup bucket: backup-pgautam-102315033



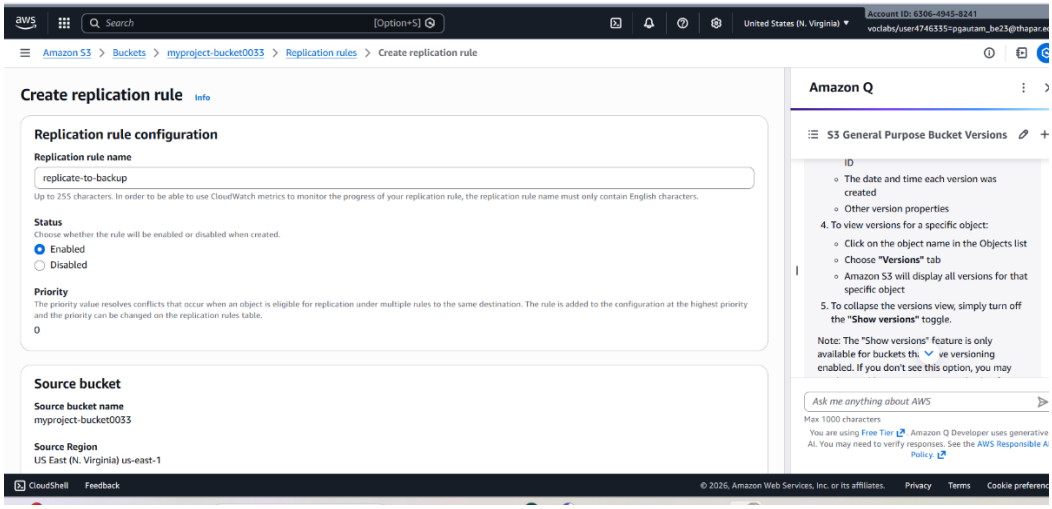
5.2 Enable Versioning

- Show screenshot of “Versioning: Enabled”

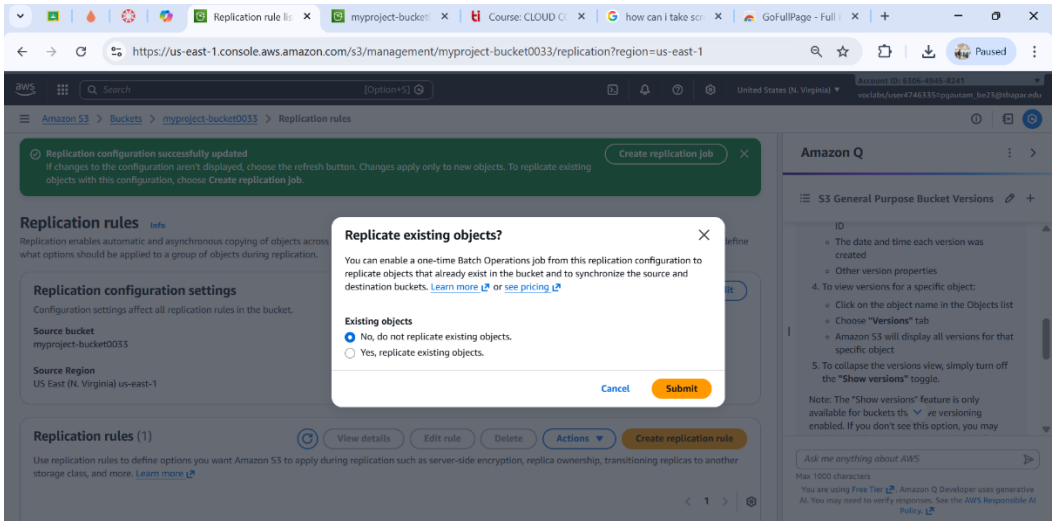


5.3 Configure Replication

- Source → Destination

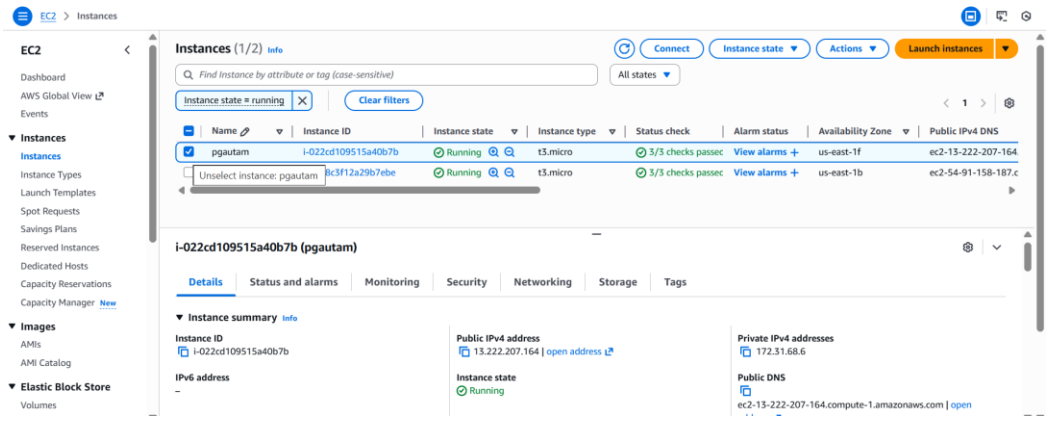


- Add screenshot of replication rule (Yes, Replicate Existing object and submit).

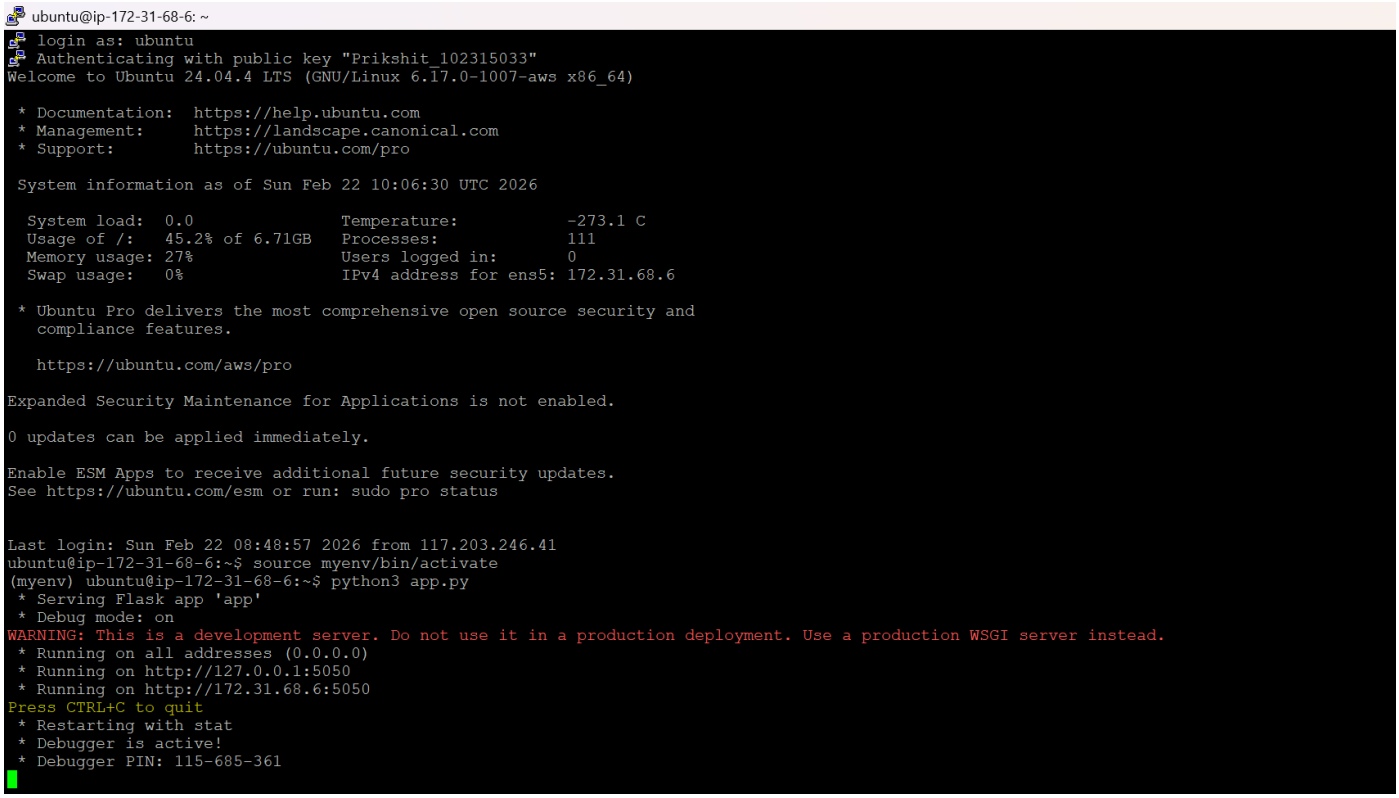


5.4 Launch EC2 Instance

- screenshot of EC2 dashboard



5.5 Install Dependencies (connect EC2 instance to putty, activate the env and launch app.py file)



Requirements:

sudo yum update

sudo yum install python3-pip

pip3 install flask boto3

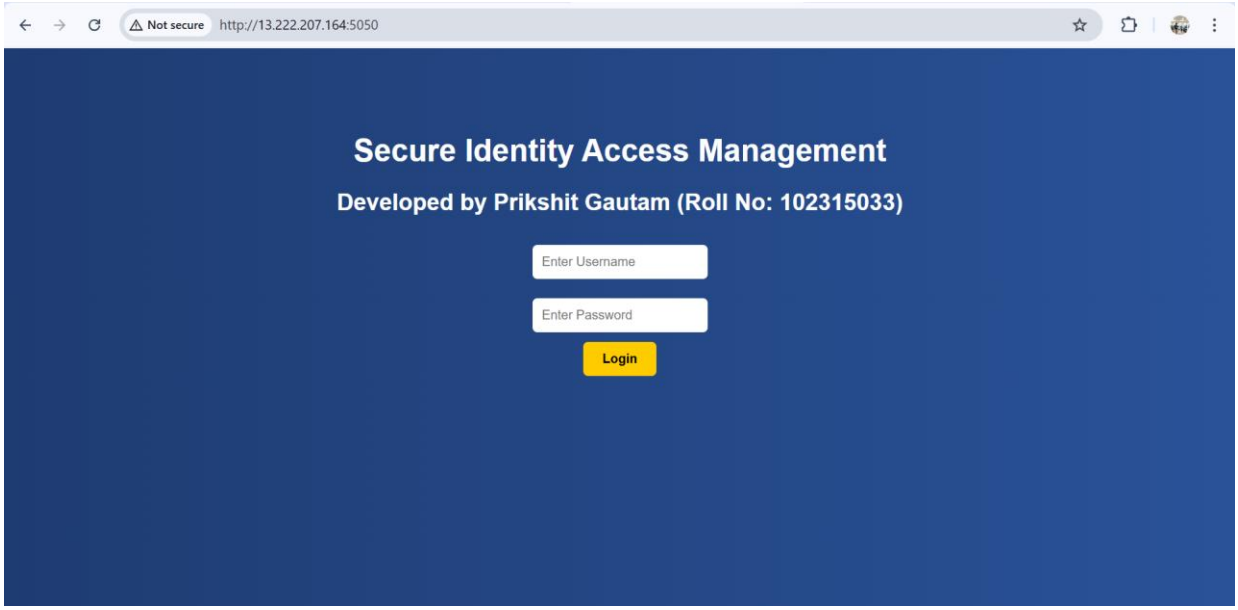
Code for app.py

5.6 Flask Application

- Login page
- Upload route
- Download route
- S3 integration

5.7 Run the Application

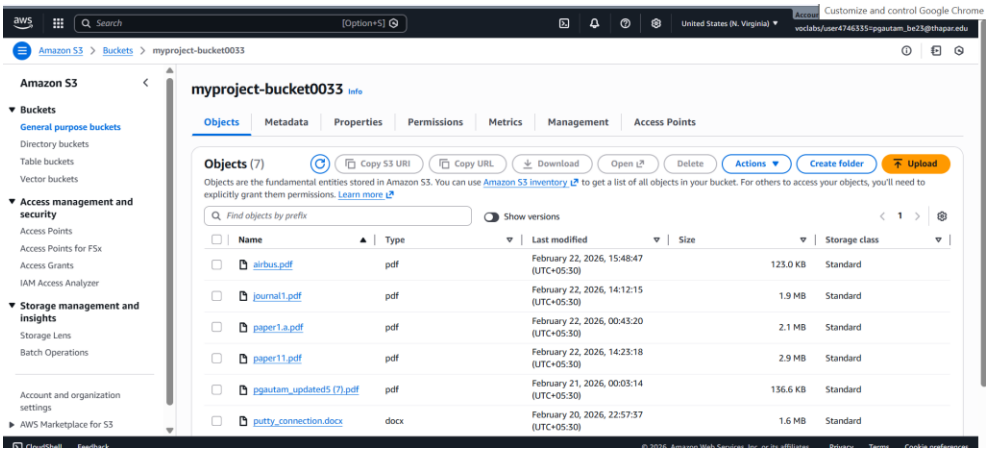
- screenshot of login page



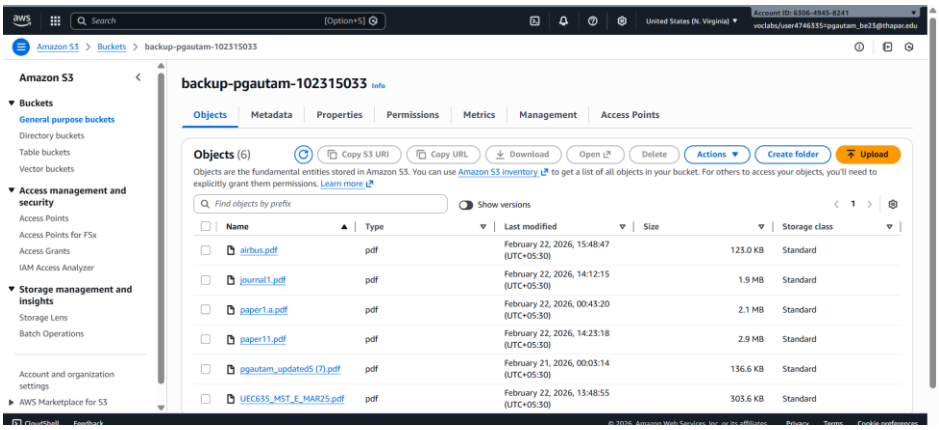
5.8 Upload File

- Screenshot of upload success message

- Screenshot of file in source bucket(added airbus.pdf)

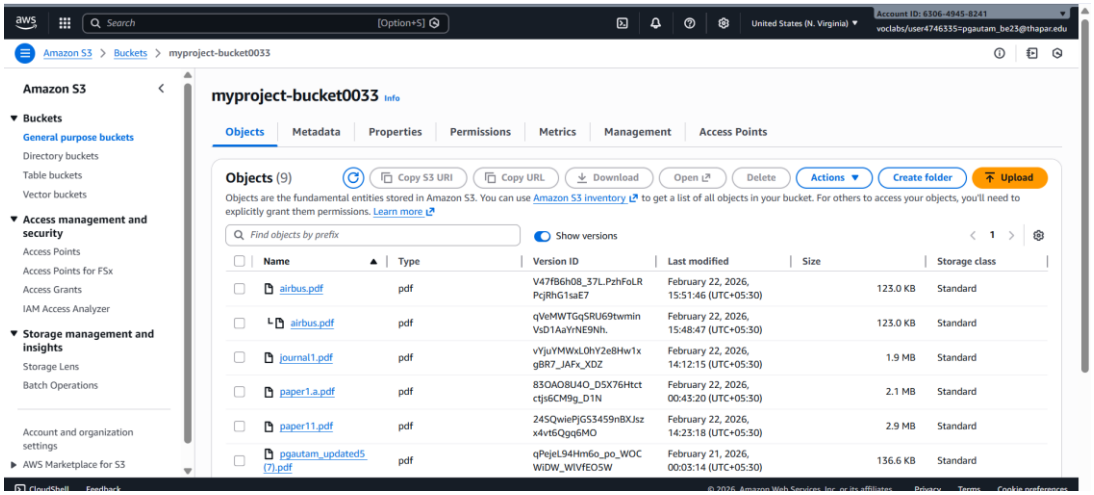


- Screenshot of file in backup bucket



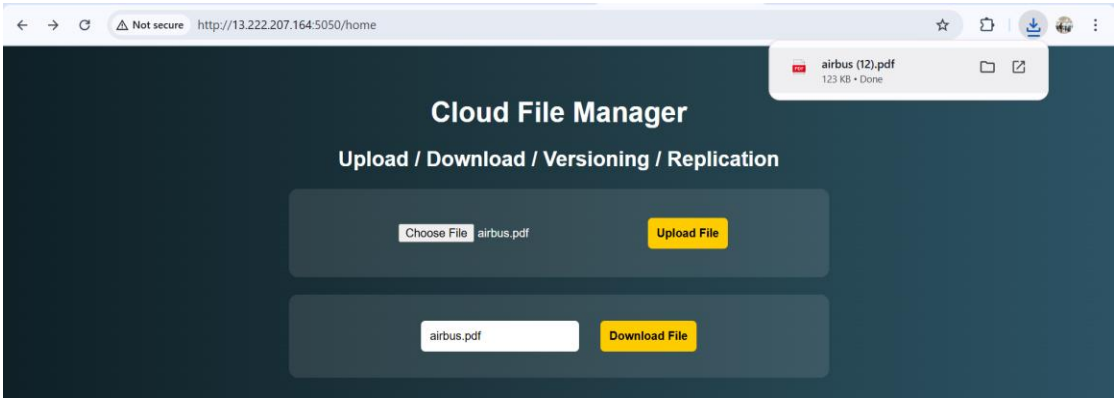
5.9 Versioning Demo

- Upload same file twice
- Screenshot showing V1, V2.



5.10 Download File

- Screenshot of download working



6. Testing

Upload Test

The upload functionality was tested by submitting multiple files through the web interface. Each file successfully appeared in the primary S3 bucket, confirming correct integration between the Flask application and AWS S3.

Replication Test

After uploading a file, the backup bucket was checked to verify automatic replication. Files consistently appeared in the destination bucket within a few seconds, confirming that the S3 replication rule was functioning correctly.

Versioning Test

The same file was uploaded multiple times to test version control. The S3 console displayed multiple versions id's, demonstrating that versioning was enabled and working as expected.

Download Test

The download feature was tested by entering the filename in the download form. Files were successfully retrieved from S3 and downloaded to the local system, confirming correct implementation of the download route.

Authentication Test

Login was tested using valid and invalid credentials. Only the correct username and password allowed access to the system, ensuring secure authentication and restricted access.

Deployment Test

The application was deployed on an AWS EC2 instance and accessed through the public IPv4 address. All features login, upload, replication, versioning, and download worked correctly in the deployed environment, confirming successful cloud deployment.

7. Results

The implemented system successfully meets all project objectives and performs reliably in a cloud environment. Files uploaded through the web interface are stored securely in the primary S3 bucket, while an automatic replication rule ensures that every file is backed up in a secondary bucket for redundancy. Versioning is enabled, allowing the system to maintain multiple revisions of the same file and protect against accidental overwrites. User authentication restricts access to authorized users, ensuring secure interaction with the application. The entire solution runs smoothly on an AWS EC2 instance, demonstrating effective deployment and end-to-end cloud integration.

8. Conclusion

This project delivers a complete cloud-based file management system built using AWS services and a Flask web interface. It allows users to securely upload, download, and manage files while automatically creating backups and maintaining multiple versions through S3 replication and versioning. The system is useful because it ensures data reliability, redundancy, and secure access, making it suitable for real-world cloud storage scenarios. All assignment requirements are fully met, including cloud storage setup, web interface development, authentication, file operations, replication, version control, and deployment on an EC2 instance.

9. Future Enhancements

The system can be improved by adding a database to manage multiple user accounts and enable personalized access. Email notifications may be integrated to alert users about uploads or updates. File encryption can enhance data security, while introducing multi-user roles would allow different permission levels, making the application more scalable and suitable for organizational use.

10. References

- AWS Documentation-Link
- Flask Documentation-Link
- Python Boto3 Docs-Link