Contents

Abstract

This project presents the design and implementation of a Secure Cloud Storage System using Amazon Web Services (AWS). The system is deployed on an EC2 instance and integrates Amazon S3 for scalable object storage. The application allows users to register, log in, upload files, and store them securely in a primary S3 bucket with versioning enabled. Additionally, every uploaded file is automatically backed up to a secondary S3 bucket to ensure data redundancy and fault tolerance.

The system demonstrates practical implementation of cloud computing concepts including Infrastructure as a Service (IaaS), object storage, version control, backup mechanisms, and secure deployment using security groups and public IP configuration.

Introduction

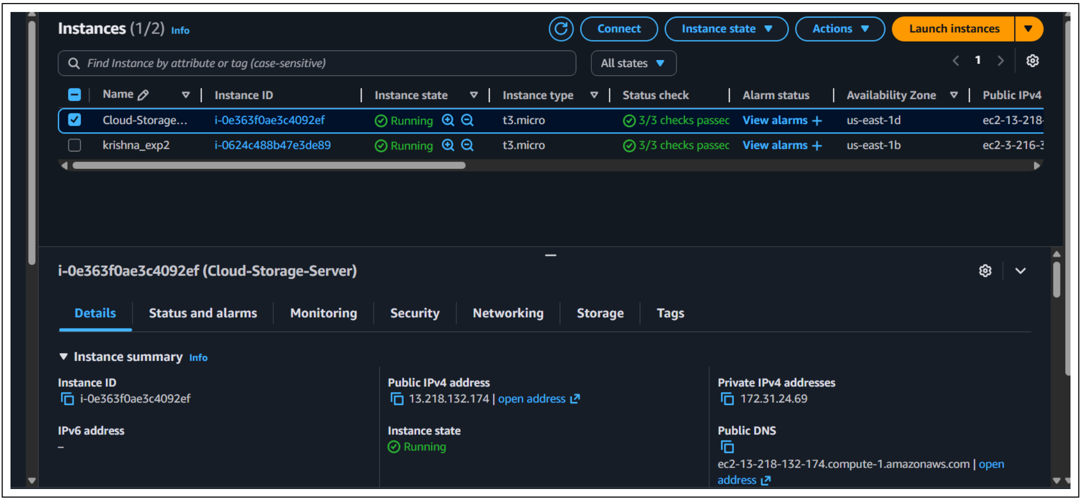
Cloud storage systems provide scalable and reliable storage solutions. This project implements a secure web-based cloud storage application using the following AWS services and features:

- **Amazon EC2** — for hosting the web application on a virtual server instance
- **Amazon S3** — for scalable and durable object storage
- **S3 Versioning** — for data protection against accidental deletion or overwrites
- **Backup Bucket** — for maintaining a redundant copy of all uploaded files

The application is accessible publicly via HTTP on port 80, providing a browser-based interface for end users.

EC2 Instance Deployment

The web application is deployed on an Amazon EC2 instance running Amazon Linux 2023. The instance is configured with a public IPv4 address and appropriate security group rules to allow inbound web traffic.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
Cloud-Storage...	i-0e363f0ae3c4092ef	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1d	ec2-13-218
krishna_exp2	i-0624c488b47e3de89	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1b	ec2-3-216-3

i-0e363f0ae3c4092ef (Cloud-Storage-Server)

Instance ID
i-0e363f0ae3c4092ef

Public IPv4 address
13.218.132.174 | open address

Private IPv4 addresses
172.31.24.69

IPv6 address
–

Instance state
Running

Public DNS
ec2-13-218-132-174.compute-1.amazonaws.com | open address

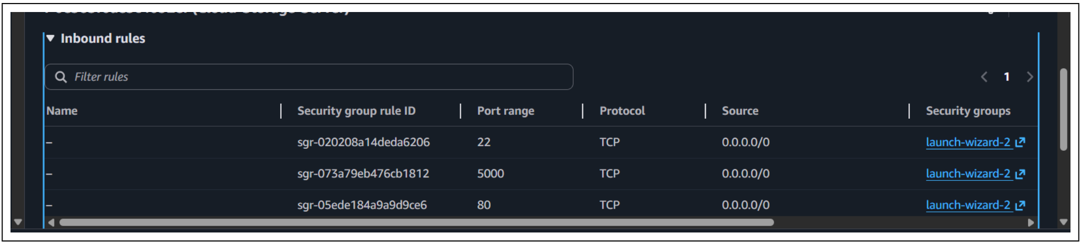
Figure 1: EC2 Instance Running — All System Checks Passed

The instance status confirms that all system and status checks have passed, verifying successful deployment of the virtual machine in the AWS cloud environment.

Security Group Configuration

A security group acts as a virtual firewall that controls inbound and outbound traffic to the EC2 instance. The following inbound rules were configured:

- **SSH (Port 22)** — For secure remote terminal access to the instance
- **HTTP (Port 80)** — For public web access to the hosted application
- **Custom TCP (Port 5000)** — For initial Flask development and testing



Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-020208a14deda6206	22	TCP	0.0.0.0/0	launch-wizard-2
–	sgr-073a79eb476cb1812	5000	TCP	0.0.0.0/0	launch-wizard-2
–	sgr-05ede184a9a9d9ce6	80	TCP	0.0.0.0/0	launch-wizard-2

Figure 2: Security Group Inbound Rules Configuration

This configuration enables secure remote management of the server while simultaneously providing unrestricted public HTTP access to the web application.

Web Application Interface

The web application is built using the Flask Python framework and provides a clean, functional interface for end users. The key features of the application include:

- **User Registration** — New users can create an account
- **User Login** — Returning users can authenticate securely
- **File Upload** — Authenticated users can upload files to the cloud
- **File Listing** — Users can view and manage their uploaded files

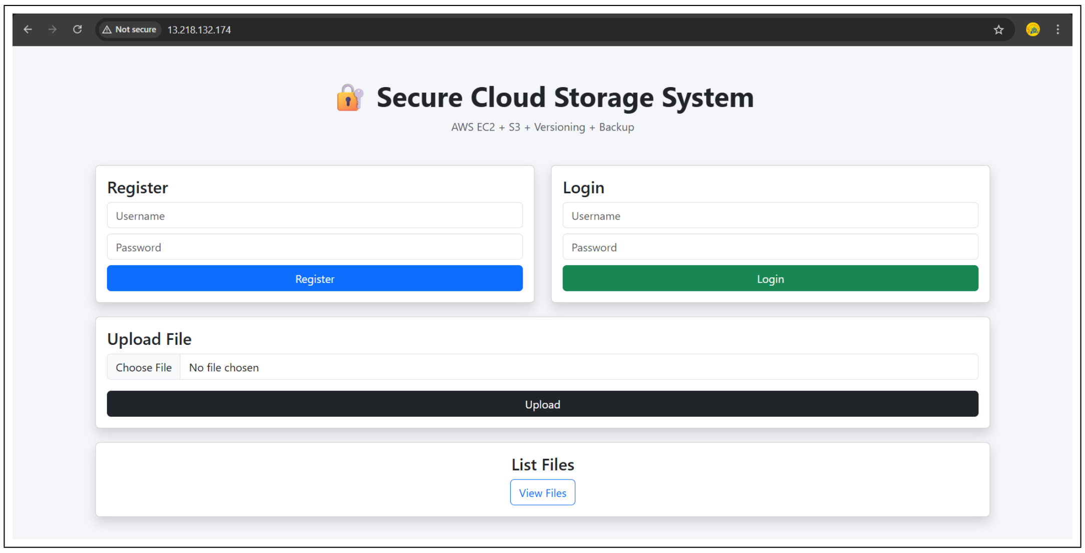


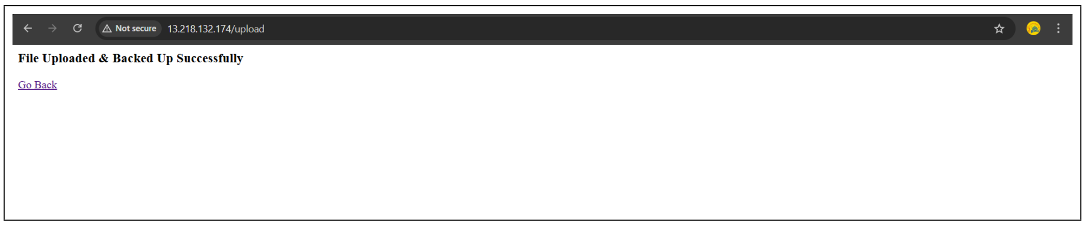
Figure 3: Application Homepage — Accessible via EC2 Public IP

The system is accessed using the public IPv4 address assigned to the EC2 instance, making it reachable from any browser without additional configuration.

File Upload and Backup Mechanism

The file upload workflow is designed to ensure both persistence and redundancy. When a user uploads a file through the web interface, the following sequence of operations is executed automatically:

1. The file is uploaded and stored in the **Primary S3 Bucket**
2. **S3 Versioning** is applied, generating a unique Version ID for the object
3. A duplicate copy of the file is simultaneously stored in the **Backup S3 Bucket**



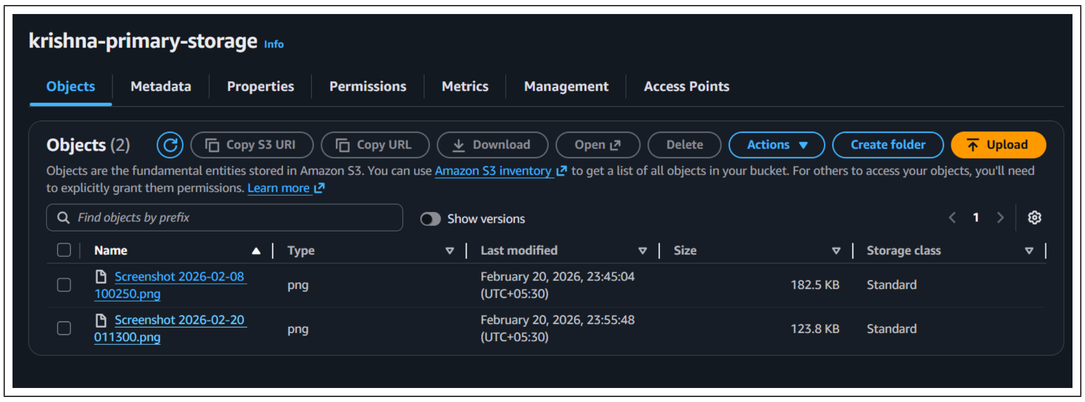
File Uploaded & Backed Up Successfully

Figure 4: Successful File Upload Confirmation

This three-step mechanism guarantees that every uploaded file is protected against accidental loss, overwriting, and infrastructure failure.

Primary S3 Bucket

The primary Amazon S3 bucket serves as the main storage layer of the application. All user-uploaded files are stored here as objects, organized by user-specific key prefixes to ensure logical separation.



Objects (2)

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

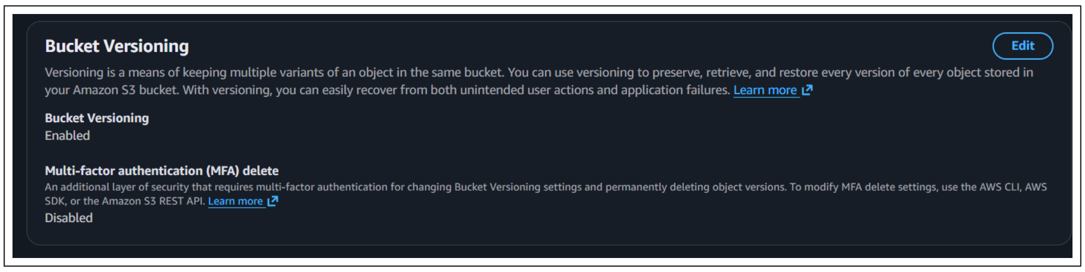
Name	Type	Last modified	Size	Storage class
Screenshot 2026-02-08 100250.png	png	February 20, 2026, 23:45:04 (UTC+05:30)	182.5 KB	Standard
Screenshot 2026-02-20 011300.png	png	February 20, 2026, 23:55:48 (UTC+05:30)	123.8 KB	Standard

Figure 5: Primary S3 Storage Bucket — Object Listing

Amazon S3 provides eleven nines (99.999999999%) of durability, making it a reliable choice for cloud-based file storage.

S3 Versioning

S3 versioning is a built-in feature that maintains a complete history of modifications to every object stored in a bucket. Once enabled, each upload of the same object key generates a new, unique **Version ID** rather than overwriting the existing object.



Bucket Versioning

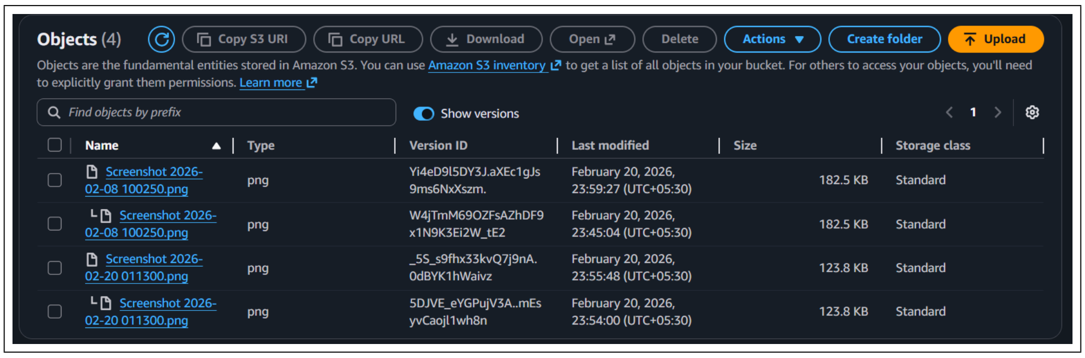
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Enabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled

Figure 6: S3 Versioning Enabled on Primary Bucket

When a file is uploaded multiple times under the same object key, S3 retains all previous versions alongside the latest one. Each version is assigned a distinct Version ID, allowing retrieval or restoration of any specific version at any point in time.



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

Name	Type	Version ID	Last modified	Size	Storage class
Screenshot 2026-02-08 100250.png	png	Yi4eD9l5DY3J.aXEc1gJs9ms6NxXszm.	February 20, 2026, 23:59:27 (UTC+05:30)	182.5 KB	Standard
└ Screenshot 2026-02-08 100250.png	png	W4jTmM69OZFsAZhDF9x1N9K3Ei2W_tE2	February 20, 2026, 23:45:04 (UTC+05:30)	182.5 KB	Standard
Screenshot 2026-02-20 011300.png	png	_5S_s9fhx33kvQ7j9nA.0dBYK1hWaivz	February 20, 2026, 23:55:48 (UTC+05:30)	123.8 KB	Standard
└ Screenshot 2026-02-20 011300.png	png	5DJVE_eYGPujV3A..mEsyvCaojl1wh8n	February 20, 2026, 23:54:00 (UTC+05:30)	123.8 KB	Standard

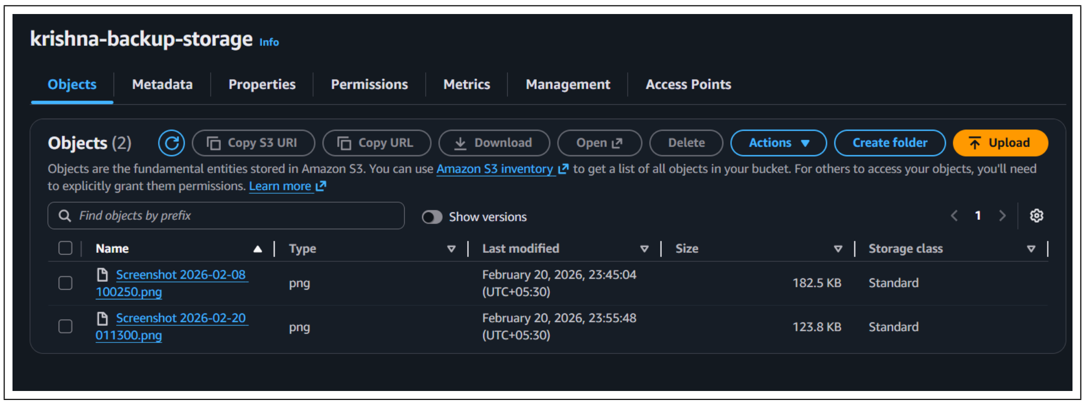
Figure 7: Multiple Object Versions with Unique Version IDs

Key benefits of versioning in this system include:

- **Accidental Deletion Recovery** — Deleted files can be restored from a previous version
- **Overwrite Protection** — Earlier versions of files remain accessible even after updates
- **Audit Trail** — A chronological history of file changes is maintained automatically

Backup Storage Bucket

A secondary S3 bucket is designated as the backup store. For every file upload, the application programmatically copies the object to this backup bucket immediately after the primary upload completes.



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

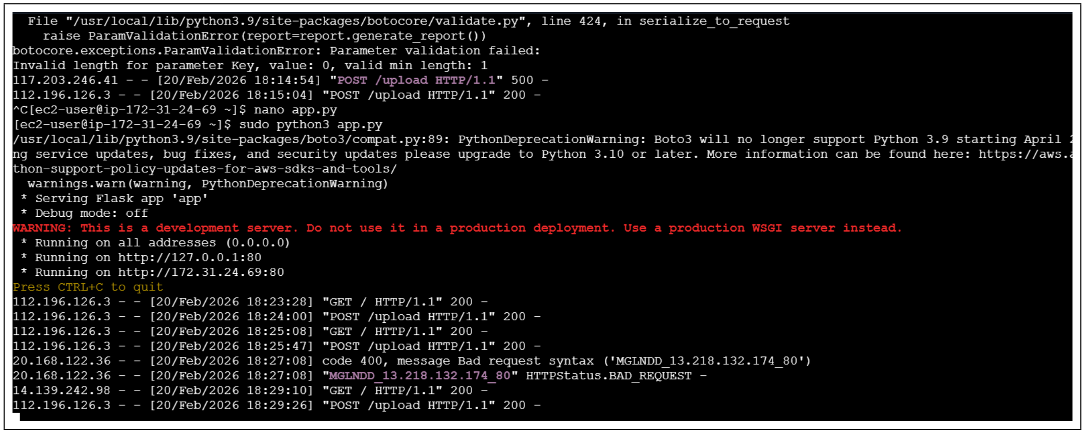
Name	Type	Last modified	Size	Storage class
Screenshot 2026-02-08 100250.png	png	February 20, 2026, 23:45:04 (UTC+05:30)	182.5 KB	Standard
Screenshot 2026-02-20 011300.png	png	February 20, 2026, 23:55:48 (UTC+05:30)	123.8 KB	Standard

Figure 8: Backup S3 Bucket — Redundant File Copies

This dual-bucket architecture ensures high availability and fault tolerance. In the event of accidental deletion or corruption of the primary bucket, all data remains intact and retrievable from the backup location.

Application Running on Port 80

The Flask application is configured to bind on port 80, enabling direct HTTP access without requiring users to specify a port number in the URL. This is achieved by running the application with root privileges on the EC2 instance.



```
  File "/usr/local/lib/python3.9/site-packages/botocore/validate.py", line 424, in serialize_to_request
    raise ParamValidationError(report=report.generate_report())
botocore.exceptions.ParamValidationError: Parameter validation failed:
Invalid length for parameter Key, value: 0, valid min length: 1
117.203.246.41 - - [20/Feb/2026 18:14:54] "POST /upload HTTP/1.1" 500 -
112.196.126.3 - - [20/Feb/2026 18:15:04] "POST /upload HTTP/1.1" 200 -
^C[ec2-user@ip-172-31-24-69 ~]$ nano app.py
[ec2-user@ip-172-31-24-69 ~]$ sudo python3 app.py
/usr/local/lib/python3.9/site-packages/boto3/compat.py:89: PythonDeprecationWarning: Boto3 will no longer support Python 3.9 starting April
ng service updates, bug fixes, and security updates please upgrade to Python 3.10 or later. More information can be found here: https://aws.
thon-support-policy-updates-for-aws-sdks-and-tools/
  warnings.warn(warning, PythonDeprecationWarning)
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.24.69:80
Press CTRL+C to quit
112.196.126.3 - - [20/Feb/2026 18:23:28] "GET / HTTP/1.1" 200 -
112.196.126.3 - - [20/Feb/2026 18:24:00] "POST /upload HTTP/1.1" 200 -
112.196.126.3 - - [20/Feb/2026 18:25:08] "GET / HTTP/1.1" 200 -
112.196.126.3 - - [20/Feb/2026 18:25:47] "POST /upload HTTP/1.1" 200 -
20.168.122.36 - - [20/Feb/2026 18:27:08] code 400, message Bad request syntax ('MGLNDD_13.218.132.174_80')
20.168.122.36 - - [20/Feb/2026 18:27:08] "MGLNDD_13.218.132.174_80" HTTPStatus.BAD_REQUEST -
14.139.242.98 - - [20/Feb/2026 18:29:10] "GET / HTTP/1.1" 200 -
112.196.126.3 - - [20/Feb/2026 18:29:26] "POST /upload HTTP/1.1" 200 -
```

Figure 9: Flask Application Running on Port 80 — Terminal Output

The terminal output confirms that the Flask development server is active, listening on all network interfaces (0.0.0.0:80), and ready to serve incoming HTTP requests from the public internet.

Conclusion

The Secure Cloud Storage System successfully demonstrates the integration of multiple AWS services to build a functional, scalable, and resilient cloud storage solution. The project highlights the following key accomplishments:

- Successful deployment of a Flask web application on an **AWS EC2** instance
- Integration with **Amazon S3** for durable and scalable object storage
- Implementation of **S3 Versioning** for data history and recovery
- Establishment of an **automatic backup mechanism** using a secondary S3 bucket
- Secure public accessibility configured via **Security Groups** and port rules

This project reflects a practical understanding of cloud infrastructure design, storage management strategies, and real-world deployment practices on AWS. The modular architecture used here can be extended to include features such as encryption at rest, IAM-based access control, CloudFront CDN integration, and lifecycle policies for automated data archival.