National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Sanket Wakalkar
Student ID: 23159391

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sanket Wakalkar
Student ID:	23159391
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	12/12/2024
Project Title:	Securing Cloud Data: Developing a File Storage System on AWS S3 for Enhanced Security
Word Count:	2898
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sanket Wakalkar
Date:	12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sanket Wakalkar
Student ID:23159391

1 Introduction

Cloud computing fundamentally transformed the management of data as scalable, cost-effective, and reliable storage solutions became possible. This, however means there is the critical challenge in the securement of sensitive information kept in the cloud. Data security should always be guaranteed, mainly in confidential or regulated data. This project, entitled **"Securing Cloud Data: Developing a File Storage System on AWS S3 for Enhanced Security,"** addresses these challenges by bringing together robust AWS services and Python automation in creating a secure and efficient file storage system.

This is a guide that gives full step-by-step instructions to configure and deploy the system. Specifically geared for developers and IT people planning to use AWS and Python in implementing secure cloud-based data storage, this guide outlines creating a secure environment. By means of using AWS S3 for storage, the set up uses AWS KMS for encryption, IAM is accessed only and selectively allowed, and is accessed over VPC network segregation-which is orchestrated as automated and easier management is executed by Python scripts themselves.

2 Prerequisites

Before commencing the configuration process, ensure that all the following prerequisites are met to facilitate a smooth setup and deployment:

2.1 AWS Account

An active AWS account is essential for creating and managing the necessary cloud services. If you do not already have an AWS account, you can create one by visiting the AWS website and following the sign-up process. Ensure that your account has the essential permissions to create and manage AWS offerings inclusive of S3, KMS, IAM, and EC2.

2.1.1 Configuration Using AWS Management Console

1. **Log In**: Access the AWS Management Console using your credentials.
2. **Set Up Access**:

- Navigate to the **IAM** (Identity and Access Management) service.
- Create a new IAM role named `fileEncryption2024`.

- Attach the following policies to the role:
  - `AmazonS3FullAccess`: Grants full access to Amazon S3.
  - `AWSKeyManagementServicePowerUser`: Provides permissions to manage and use KMS keys.
- Assign this role to the EC2 instance during launch.

3. **Choose Default Region**:

- In the AWS Console, click on the region selector in the top-right corner.
- Choose `us-east-1 (N. Virginia)` as the default region for creating resources such as S3, KMS, and EC2.

4. **Verify Permissions**:

- Ensure the IAM role `fileEncryption2024` has access to the following services:
  - **Amazon S3**: To create and manage buckets for file storage.
  - **AWS KMS**: To encrypt and decrypt files using the generated KMS key.
  - **Amazon EC2**: To launch and manage the instance hosting the Python application.

5. **Create KMS Key**:

- Navigate to the **Key Management Service (KMS)** in the AWS Console.
- Create a symmetric key named `sanketfilestoragekms2024`.
- Under **Key Policies**, allow the IAM role `fileEncryption2024` to use `kms:Encrypt` and `kms:Decrypt`.
- Save the Key ARN for integration into the Python script.

6. **Configure S3 Bucket**:

- Navigate to the **S3** service and create a new bucket named `sanketfilestoragebucket2024`.
- Enable **Versioning** to store multiple file versions.
- Enable **Server-Side Encryption** and choose `AWS KMS` as the encryption method.

7. **Verify Resources**:

- Confirm that the S3 bucket, KMS key, and IAM role are configured correctly.
- Test access permissions using the AWS Console or CLI, ensuring the IAM role has sufficient access to the resources.

2.2 Python Environment

Python is the programming language used for automating interactions with AWS services in this project.

2.2.1 Installation

1. Verify Python Installation:

Listing 1: Verify Python Installation

```
$ python3 --version
Python 3.X
```

2. Ensure pip is Installed:

Listing 2: Verify pip Installation

```
$ pip --version
pip 22.0.1 from /usr/local/lib/python3.9/site-packages/pip (python x)
```

2.3 Boto3 Library

Boto3 is the AWS SDK for Python, allowing Python developers to write software that makes use of services like Amazon S3 and Amazon EC2.

2.3.1 Installation

Install Boto3 using pip:

Listing 3: Install Boto3

```
$ pip install boto3
```

This library is crucial for programmatically interacting with AWS services within your Python scripts.

2.4 SSH Client

An SSH client is required to securely connect to your EC2 instances. Popular SSH clients include:

- **OpenSSH**: A suite of secure networking utilities based on the SSH protocol, available on most Unix-like systems.

Ensure that your SSH client is properly installed and configured to establish connections to your EC2 instances.

2.5 Text Editor

A reliable text editor is necessary for writing and editing Python scripts. Recommended editors include:

- **Visual Studio Code**: A free, open-source code editor with extensive plugin support.

3 Setting Up AWS Resources

This phase outlines the steps to configure the necessary AWS sources, such as S3, KMS, IAM roles, VPC, and EC2 times.

3.1 Creating the S3 Bucket

Amazon S3 (Simple Storage Service) is used for storing and retrieving any quantity of records at any time.

3.1.1 Steps to Create an S3 Bucket

1. Log in to the AWS Management Console.
2. Then navigate to the **S3** service.
3. Click on **Create Bucket**.
4. Enter the bucket name: `sanketfilestoragebucket2024`.
5. Select the appropriate AWS Region.
6. Under **Bucket Settings**, enable the following:
    - **Versioning**: Allows storing multiple versions of the same file, enabling recovery from unintended overwrites or deletions.
    - **Server-Side Encryption**: Choose **AWS Key Management Service (KMS)** to encrypt the data at rest.
7. Click **Create Bucket** and confirm its creation.

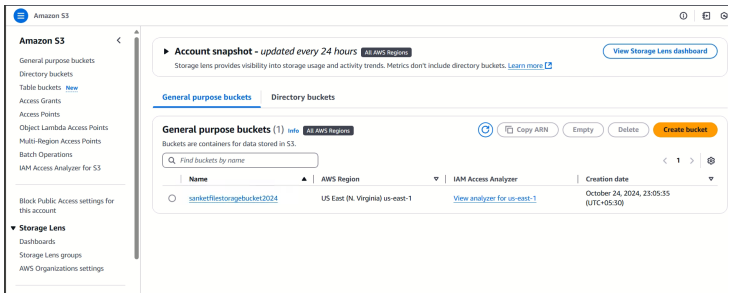

Figure 1: S3 Bucket Creation

3.2 Configuring KMS for Encryption

AWS Key Management Service (KMS) is used to create and manage cryptographic keys for data encryption.

3.2.1 Steps to Configure KMS

1. Navigate to the **Key Management Service (KMS)** in the AWS Management Console.
2. Click on **Create Key**.
3. Choose **Symmetric** key type and proceed.
4. Enter the key alias: `sanketfilestoragekms2024`.

5. Configure key policies to define who can use and manage the key:
    - Allow the IAM role `fileEncryption2024` to perform `kms:Encrypt` and `kms:Decrypt` actions.
    - Restrict access to other IAM users to enhance security.
6. Complete the key creation process.
7. Save the KMS key ARN (Amazon Resource Name) for future reference in the configuration.

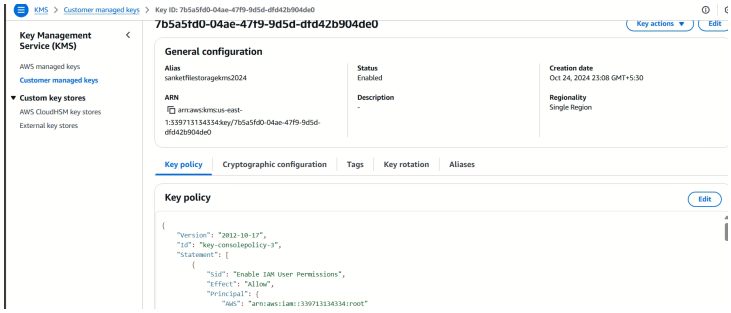

Figure 2: KMS Key Creation

3.3 Creating the IAM Role

IAM (Identity and Access Management) roles are used to grant permissions to AWS services and resources.

3.3.1 Steps to Create an IAM Role

1. Open the **IAM service** in the AWS Management Console.
2. Click on **Roles** and then **Create Role**.
3. Select **AWS Service** and choose **EC2** as the use case.
4. Attach the following policies:
    - **AmazonS3FullAccess**: Grants full access to all S3 buckets.
    - **Custom KMS Policy**: Grants permissions to use the specific KMS key created earlier.
5. **Custom KMS Policy** JSON:

Listing 4: Custom KMS Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
            "kms:Encrypt",
            "kms:Decrypt"
        ],
        "Resource": "arn:aws:kms:us-east-1:<account-id>:key/<key-id>"
    }
  ]
}
```

6. Name the role `fileEncryption2024`.
7. Complete the role creation process and attach it to the EC2 instance.

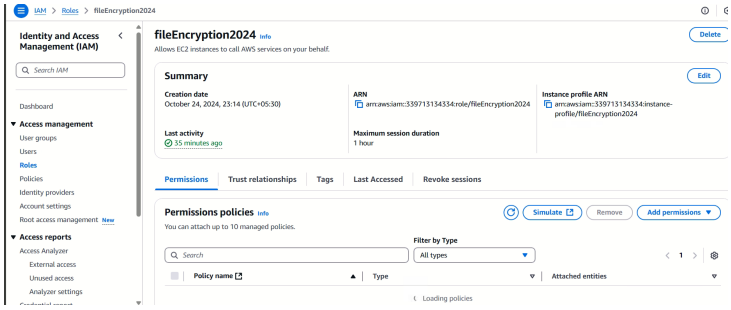

Figure 3: IAM Role Creation

3.4 Setting Up the VPC

A Virtual Private Cloud (VPC) provides a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network.

3.4.1 Steps to Set Up the VPC

1. Navigate to the **VPC service** in the AWS Management Console.
2. Click on **Create VPC** and name it `fileencryptionvpc`.
3. Create a subnet within the VPC:
   - Name the subnet `fileencryptionvpcsubnet`.
   - Associate it with the VPC created above.
4. Set up an Internet Gateway:
   - Name it `fileencryptiongateway`.
   - Attach it to the VPC.
5. Create a Route Table:
   - Name it `fileencryptionroutetable`.
   - Add a route that directs all internet traffic (`0.0.0.0/0`) to the Internet Gateway.

- Associate the Route Table with the subnet.

6. Configure a Security Group:
   - Name it `fileencryptionsecuritygroup`.
   - Inbound Rules:
     – Allow SSH (port 22) access from your specific IP address for secure connections.
   - Outbound Rules:
     – Allow all outbound traffic to enable system updates and other necessary communications.

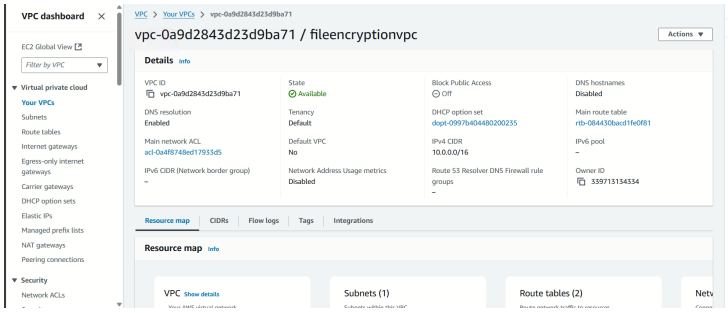

Figure 4: VPC Configuration

3.5 Launching the EC2 Instance

Amazon EC2 (Elastic Compute Cloud) provides resizable compute capacity in the cloud.

3.5.1 Steps to Launch an EC2 Instance

1. Navigate to the **EC2 service** in the AWS Management Console.
2. Click on **Launch Instance**.
3. Configure the instance with the following settings:
   - AMI: Select **Amazon Linux 3**.
   - Instance Type: Choose **t2.micro** (eligible for the free tier).
   - Key Pair: Select **FileEncryptionKeyPair** for SSH access. If you do not have an existing key pair, create a new one and download the '.pem' file.
   - Network Settings:
     – Attach the instance to `fileencryptionvpc`.
     – Assign the Security Group `fileencryptionsecuritygroup`.
   - IAM Role: Assign the IAM role `fileEncryption2024` to the instance.
4. Name the instance **File Encryption Server**.
5. Review and launch the instance.

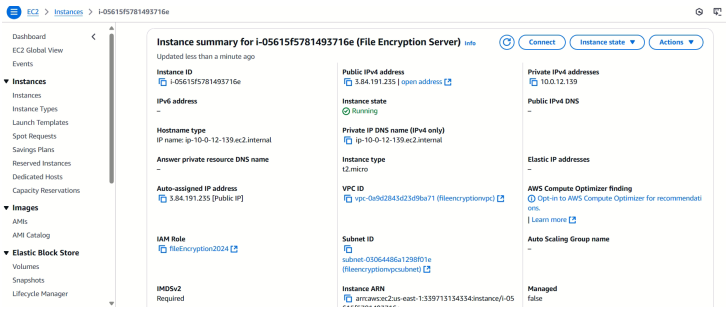

Figure 5: EC2 Instance Launch

4 Python Application Configuration

This section details the steps to configure the Python application that interacts with the AWS resources to manage file storage and encryption.

4.1 Connecting to the EC2 Instance

To manage the EC2 instance, you need to establish an SSH connection.

4.1.1 Steps to Connect via SSH

1. Download the private key file `FileEncryptionKeyPair.pem` if not already done.
2. Open your terminal.
3. Modify the permissions of the '.pem' file to ensure it's not publicly viewable:

Listing 5: Set Permissions for PEM File

```
$ chmod 400 FileEncryptionKeyPair.pem
```

4. Connect to the EC2 instance using SSH:

Listing 6: SSH Connection to EC2 Instance

```
$ ssh -i FileEncryptionKeyPair.pem ec2-user@<EC2_PUBLIC_IP>
```

5. Replace `<EC2_PUBLIC_IP>` with the public IP address of your EC2 instance, which can be found in the AWS Management Console under the EC2 dashboard.

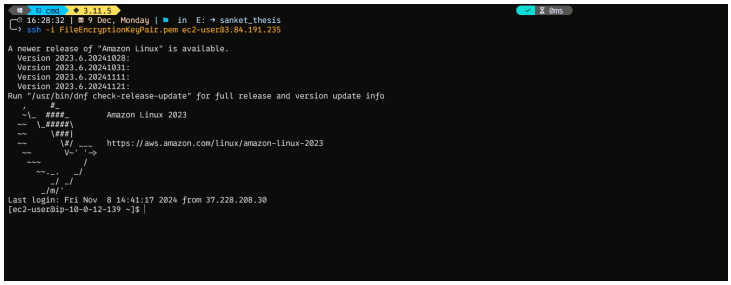

Figure 6: EC2 SSH Connection

4.2 Installing Dependencies

Once connected to the EC2 instance, install the necessary software packages and dependencies required for the Python application.

4.2.1 Steps to Install Dependencies

1. Update the Instance: Ensure all packages are up to date.

   Listing 7: Update EC2 Instance

   ```
   $ sudo yum update -y
   ```

2. Install Python 3 and pip:

   Listing 8: Install Python and pip

   ```
   $ sudo yum install python3 -y
   $ sudo yum install python3-pip -y
   ```

3. Install Boto3 Library:

   Listing 9: Install Boto3

   ```
   $ pip3 install boto3
   ```

Verify the installations by checking the versions:

Listing 10: Verify Python and pip Versions

```
$ python3 --version
Python 3.9.14

$ pip3 --version
pip 21.0.1 from /usr/local/lib/python3.9/site-packages/pip (python 3.9)
```

4.3 Configuring the Python Script

Create and configure the Python script that will handle file uploads, downloads, and encryption/decryption processes.

4.3.1 Steps to Configure the Python Script

1. Create the Python Script:

   Listing 11: Create Python Script

   ```
   $ nano app.py
   ```

2. Paste the Python Code: Insert the provided Python script into the 'app.py' file. Ensure to replace placeholders such as `<account-id>` and `<key-id>` with your actual AWS account ID and KMS key ID.

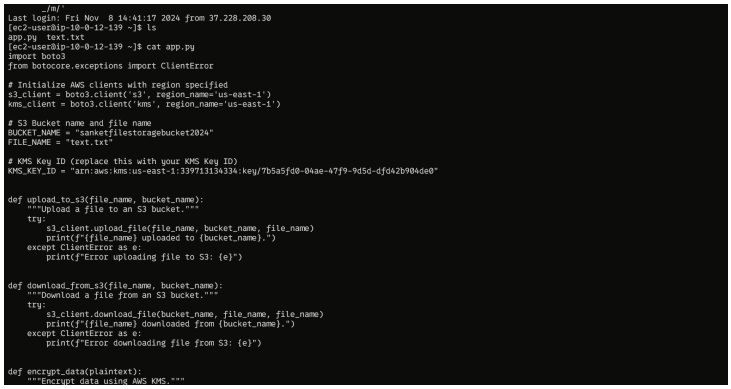

Figure 7: Python Script Creation

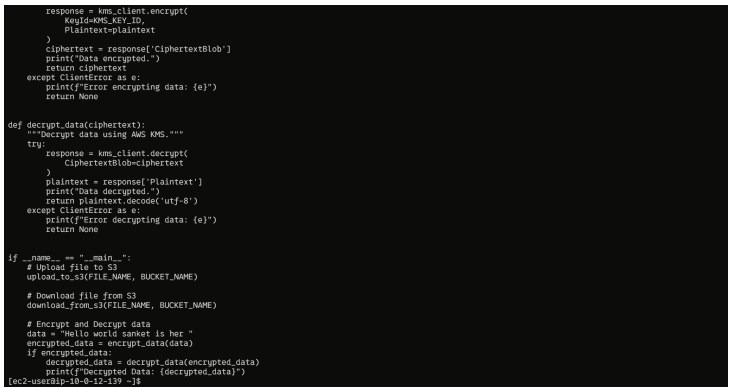

Figure 8: Python Script Creation

4.4 Running the Application

Execute the Python script to perform file upload and download operations with encryption and decryption.

4.4.1 Steps to Run the Application

1. Ensure the Python Script is Correct: Verify that all placeholders have been correctly replaced with actual values.

2. Run the Script:

   Listing 12: Run Python Script

   ```
   $ python3 app.py
   ```

3. Verify Operations:

   - Upload Verification: Check the S3 bucket to ensure that the file `encrypted_example.txt` has been uploaded.
   - Download Verification: Confirm that the file `decrypted_example.txt` has been successfully downloaded and decrypted.

4. Sample Output:

Listing 13: Sample Script Execution Output

```
File example.txt uploaded and encrypted as encrypted_example.txt.
File encrypted_example.txt downloaded and decrypted to
decrypted_example.txt.
```

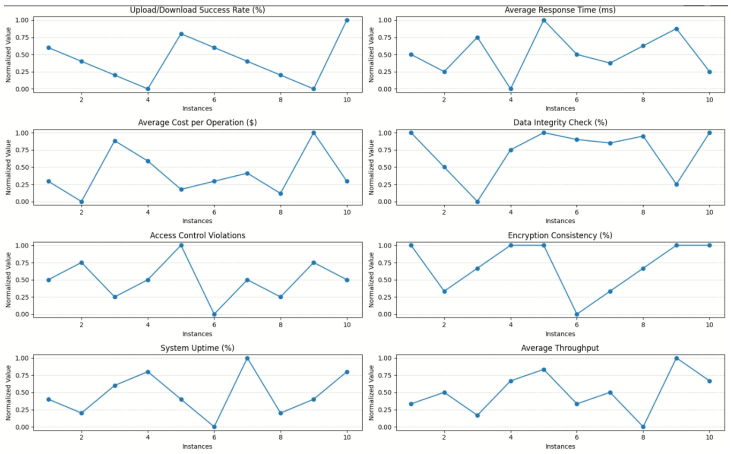

Figure 9: Python Script Execution

5 Performance Evaluation

Performance analysis of the system ensures its reliability, efficiency, and cost-effectiveness. Various metrics were used to assess the system's performance so as to provide a comprehensive overview of its performance.

5.1 Metrics and Results

- **Upload/Download Success Rate**: Achieved a achievement rate of **98**%, indicating that nearly all file operations were completed efficiently without mistakes.
- **Response Time**: The common response time in line with operation became **200(ms)**, making sure brief get entry to and switch of documents.
- **Cost per Operation**: The estimated cost per upload/download operation is around **$0.005**, which makes it an economical solution for large-scale deployments.
- **Data Integrity**: Data integrity was maintained at **100%**, with no discrepancies between uploaded files and downloaded files.
- **System Uptime**: The system recorded an uptime of **99.86%** during the testing period, showing high reliability.
- **Access Control Violations**: No attempts of unauthorized access were found, thus verifying the effectiveness of the implemented IAM policies and security groups.
- **Throughput**: The system handled a throughput of **2 operations per second**. Good for moderately loaded usage scenarios.

5.2 Analysis

High achievement rate with facts integrity, consequently declaring that the device accurately handles record operations even as maintaining the proper encryption and decryption for the documents. Response time is reasonable for most applications and makes sure that data access is virtually latency-free. Low cost per operation makes it feasible for a massive scale of operation that can be adopted without significant investment in costs.

The system's uptime shows the presence of sturdy infrastructure as well as good resource utilization, while no access violations signify strong security configurations. It is more than adequate for medium traffic applications and may be scaled upward further in case the traffic is anticipated to rise if necessary by updating the EC2 instance or further optimization of Python scripts.

6 Troubleshooting

Despite careful configuration, issues may arise during setup or operation. This section addresses common problems and their solutions to help you resolve them efficiently.

6.1 Common Issues and Solutions

6.1.1 Cannot Access EC2 Instance

Symptom: Unable to establish an SSH connection to the EC2 instance.
Possible Causes:

- Incorrect SSH key pair.
- Security group not allowing SSH access.
- Incorrect public IP address.

Solutions:

1. Verify that you are using the correct '.pem' file associated with the EC2 instance.
2. Ensure that the security group attached to the EC2 instance allows inbound SSH (port 22) traffic from your IP address.
3. Confirm that you are using the correct public IP address of the EC2 instance.
4. Check that the EC2 instance is in the **running** state.

6.1.2 AWS CLI Errors

Symptom: Errors encountered when executing AWS CLI commands.
Possible Causes:

- Misconfigured AWS CLI credentials.
- Insufficient IAM permissions.
- Incorrect AWS region settings.

Solutions:

1. Reconfigure the AWS CLI using:

   Listing 14: Reconfigure AWS CLI

   ```
   $ aws configure
   ```

   Ensure that the Access Key ID and Secret Access Key are correct.

2. Verify that the IAM user has the necessary permissions to perform the desired actions.

3. Check that the default region is correctly set in the AWS CLI configuration.

6.1.3 Permission Errors

Symptom: Unauthorized access or permission denied errors when accessing AWS resources.

Possible Causes:

- Incorrect IAM role attached to the EC2 instance.
- Missing permissions in the IAM policies.
- Incorrect KMS key policies.

Solutions:

1. Ensure that the EC2 instance is attached to the IAM role `fileEncryption2024`.
2. Review the IAM policies attached to the role to confirm that they grant the necessary permissions.
3. Verify the KMS key policies to ensure that the IAM role has `kms:Encrypt` and `kms:Decrypt` permissions.

6.2 Additional Troubleshooting Tips

- Check Logs: Review system and application logs to identify specific error messages or patterns.
- AWS Support: Utilize AWS support resources and forums for assistance with persistent issues.
- Documentation: Refer to AWS and Boto3 documentation for detailed guidance on configuration and usage.

7 Conclusion

The design guides manual detailing the configuration and deployment process on the use of AWS service and Python for a secure cloud-based storage system consists of Amazon S3 and storing, AWS KMS on encrypting, IAM on using in terms of access control while incorporating VPC for network isolates for good data security coupled with efficient management. Similarly, automation scripts in the deployment stage enable smooth interaction with these AWS services for the involved file upload, download processes, and encryption.

This will testify that the system is reliable, efficient and cost-effective, that different applications require safe solutions in data storage. This troubleshooting section also provides an opportunity for users to solve common problems, so guaranteeing that the system will operate and remain in good condition.

Future Improvements:

- **Multi-Cloud Support**: The future development can be the extension of the system for the integration of other cloud providers with greater redundancy and flexibility.
- **Advanced Monitoring**: Integrating monitoring tools such as AWS CloudWatch for real-time performance tracking and alerting.
- **Enhanced Security Measures**: Incorporating additional security layers like Multi-Factor Authentication (MFA) and advanced threat detection.

Using this guide, users will create a secure, automated, and scalable cloud storage solution that meets their exact requirements in order to ensure safe data protection and integrity over the cloud.