Project Report:
File Backup and Sync to AWS S3

1. Introduction

The objective of this project is to develop a Python-based script that can automatically back up files or directories to AWS S3, with added features like scheduling backups, encryption, and versioning. This will help users safeguard their important files by ensuring they are backed up periodically, encrypted for security, and versioned for consistency.

2. Problem Statement

File Backup and Sync: Develop a script or application that automatically backs up files or directories to AWS S3. Additional features like scheduling, encryption, and versioning should be implemented to provide a robust backup solution.

3. Solution Overview

The solution provides an automated backup system that supports:

- **Backup of files and directories** to an S3 bucket.
- **Encryption** of files during backup to ensure data privacy.
- **Versioning** of backups in S3 to keep track of changes.
- **Scheduling** to run backups periodically, e.g., once, minutely, hourly, daily, etc.

The solution uses Python, AWS EC2, S3, and other tools to achieve these functionalities.

---

4. System Architecture

The solution architecture consists of the following components:

- **AWS EC2 Instance**: A virtual server that runs the backup script.
- **AWS S3 Bucket**: A storage service where the backed-up files are stored.
- **IAM Role**: Provides necessary permissions to EC2 to interact with the S3 bucket.
- **Python Script**: Performs the backup operations, including encryption, versioning, and scheduling.

5. Implementation Steps

Step 1: AWS Setup

1. **Create an S3 Storage Bucket**:
    - Log into the AWS Management Console and create an S3 bucket where the files will be backed up. This bucket will serve as the central location for storing backup files.
2. **Create an EC2 Instance**:
    - Launch an EC2 instance (e.g., using Amazon Linux 2).
    - Use the instance to run the backup script.
    - Secure the EC2 instance by creating and downloading an SSH key pair.
3. **Configure IAM Role for EC2**:
    - Create a new IAM role with the following policies:
        - **AmazonS3FullAccess**: Allows EC2 to read/write to the S3 bucket.
        - **CloudWatchLogsFullAccess** (optional): Provides access to CloudWatch for logging purposes.
    - Attach this IAM role to the EC2 instance.
4. **SSH into EC2 Instance**:

    Access the EC2 instance using SSH:
    code
    ```
    ssh -i <your-key>.pem ec2-user@<EC2-public-ip>
    ```

5. **Install Required Software**:

    Update the EC2 instance's package list:

    ```
    sudo yum update -y
    ```
    Install Python 3 and pip:

    ```
    sudo yum install python3.11 -y
    sudo yum install pip -y
    ```

    Install necessary Python libraries:

    ```
    pip3 install boto3 cryptography schedule
    ```

    -

Step 2: Backup Script Implementation

The backup script (`main.py` and `backuplib.py`) is responsible for backing up files from the EC2 instance to the S3 bucket. The main features include:

- **Backup**: The script can back up a single file or an entire directory to the S3 bucket.
- **Encryption**: It supports optional encryption using the Fernet encryption scheme.
- **Versioning**: The script can enable or disable versioning on the S3 bucket to track different versions of the same file.
- **Scheduling**: The script can schedule backups to run periodically (once, minutely, hourly, etc.).

`main.py` (Entry point to the backup process):

```
import backuplib, sys
if len(sys.argv) < 2:
    backuplib.welcome()
    backuplib.interactive_backup()
else:
    backuplib.main()
```

`backuplib.py` (Core Backup Logic):

1. **Argument Parsing**: The script uses `argparse` to define command-line arguments for different operations, such as:
    - `backup`: Perform a backup of a directory/file.
    - `encrypt`: Encrypt a specified file.
    - `decrypt`: Decrypt a file using a specified encryption key.
    - `help`: Show help about available commands.

2. **Backup Logic**: The `cmd_backup` function backs up files to S3, optionally encrypting them before upload. It handles the logic for both single file and directory backups.
3. **Encryption & Decryption**: Files are encrypted using the `cryptography.fernet` library before being uploaded to S3, ensuring that sensitive data is protected. Decryption is handled when restoring files.
4. **Versioning**: The `configure_versioning` function manages versioning in S3, enabling it for the backup files. This ensures that multiple versions of files can be stored.
5. **Scheduling**: The script uses the `schedule` library to allow backups to run at specified intervals (e.g., daily, monthly, hourly, etc.).

---

6. Testing

The backup process was tested in several scenarios:

- **Single file backup**: A single file was backed up to the S3 bucket, with optional encryption and versioning enabled.
- **Directory backup**: A directory containing multiple files was successfully backed up.
- **Encryption**: Files were encrypted before being uploaded to the S3 bucket, ensuring that sensitive data remained protected.
- **Scheduling**: Backups were successfully scheduled to run at different intervals (daily, weekly) using the schedule library.
- **Versioning**: The versioning feature was tested by uploading the same file multiple times, and verifying that S3 correctly stored different versions.

---

7. Challenges Faced

- **Permission Issues**: Initially, EC2 did not have sufficient permissions to write to S3. This was resolved by correctly setting up the IAM role with appropriate policies.
- **Encryption Key Management**: Managing encryption keys securely was challenging, especially when automated backups are scheduled. We provided an option to generate a new key or use a user-provided key.
- **Scheduling**: Ensuring that scheduled backups ran as expected without overlapping or conflicts required careful configuration of the `schedule` library.

---

8. Future Enhancements

- **Error Handling**: Improve error handling to gracefully manage potential failures such as connection issues or invalid file paths.
- **Backup Monitoring**: Integrate with AWS CloudWatch to monitor backup status and notify the user in case of failures.
- **Backup Optimization**: Implement differential or incremental backups to reduce the amount of data transferred during each backup.
- **Web Interface**: Develop a web interface for users to configure backup settings, schedules, and view backup logs.

---

9. Conclusion

The project successfully meets the requirements outlined in the problem statement. It provides a robust backup solution with support for scheduling, encryption, and versioning, ensuring that files are securely backed up to AWS S3. The use of AWS EC2 and S3 along with Python provides a cost-effective and scalable solution for file backup automation.