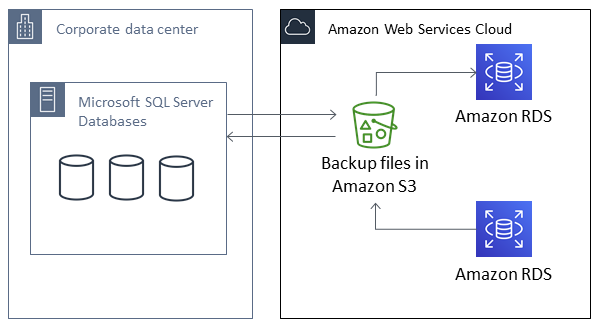
**Final Report: Cloud Backup Services Using Python and AWS S3**

**1. Problem Motivation**

In an era where data is indispensable for both personal and business operations, protecting this data from loss or corruption is critical. Unfortunately, data loss can occur due to various reasons such as hardware failures, human errors, cyber-attacks, and even natural disasters. The traditional methods of backup—such as using local storage devices—can be prone to physical damage, theft, or accidental loss. In contrast, cloud storage solutions offer enhanced scalability, security, and accessibility for data backups.

Amazon Web Services (AWS) S3 (Simple Storage Service) is one of the most widely adopted cloud storage solutions, providing scalable and secure data storage. Additionally, AWS offers complementary services like Simple Notification Service (SNS), which can be used to send alerts for system statuses and backup operations. To leverage these services and ensure the security and availability of critical data, this project aims to build an automated, secure, and scalable backup solution. The system is designed to back up files and MySQL databases to AWS S3, ensuring data encryption before upload and providing users with real-time notifications about the backup process.

The motivation behind this project is to automate the backup process, providing a reliable solution to safeguard data against potential loss or corruption while ensuring that the process is as seamless and error-free as possible.



The image above shows the overview of the SQL to AWS backup system as implemented in the project.

**2. Design Goals and Performance Questions**

The system was designed to meet the following goals:

* **Security**: Protect data by encrypting it before uploading to S3. This ensures that even if the data is intercepted during transmission or while stored in the cloud, it remains inaccessible without the decryption key.
* **Automation**: Schedule backups for files and databases at regular intervals (e.g., daily, weekly) without requiring manual intervention. This reduces the risk of forgetting to perform backups and ensures the data is always up to date.
* **Reliability**: Handle potential errors, such as network failures or AWS service issues, by implementing retry mechanisms to ensure that backups are always completed successfully.
* **Efficiency**: Ensure that backup operations, especially for large files and databases, are performed in a timely manner without overwhelming system resources.
* **Scalability**: Ensure that as the amount of data grows, the backup system can scale to handle larger files and databases with minimal degradation in performance.

**3. Design Architecture and Performance Metrics**

The architecture of the cloud backup system is composed of several key components that work together to provide a seamless backup experience. The components include:

1. **File Backup**:
   * The system monitors a specified directory for any new or modified files. Files are encrypted before being uploaded to AWS S3 to ensure that sensitive data is protected.
   * **Key components**: encryptor.py (for file encryption), file\_backup.py (for monitoring and backing up files), and s3\_utils.py (for handling uploads to S3).

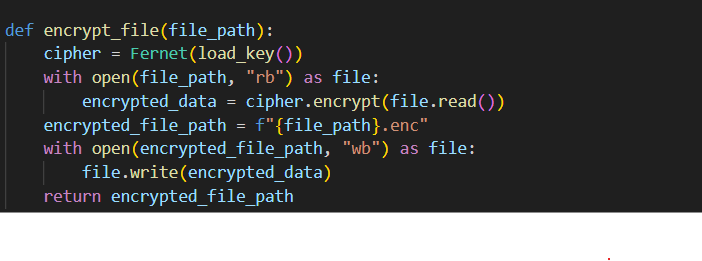


Figure 1 shows the code snippet that encrypts the data

1. **Database Backup**:
   * The system uses the mysqldump command to generate backups of MySQL databases. The resulting database dump files are encrypted and uploaded to S3.
   * **Key components**: db\_backup.py (for dumping the MySQL database and uploading the file), mysqldump (used for creating the database backup).

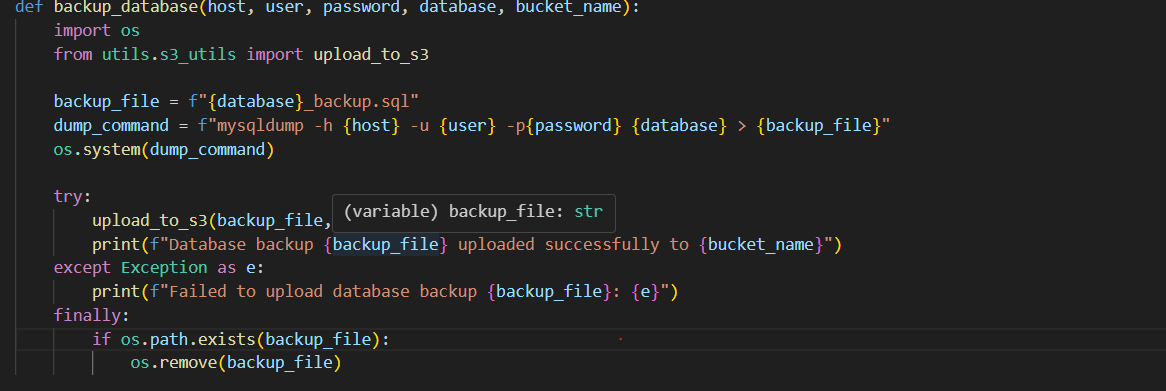


Figure 2 shows the code that backs up the database

1. **Scheduling**:
   * Using the schedule library, the system runs backup operations at predefined intervals, such as every 24 hours or once a week. This ensures that backups are performed regularly without the need for manual intervention.
   * **Key components**: scheduler.py (for automating the scheduling of backup tasks).

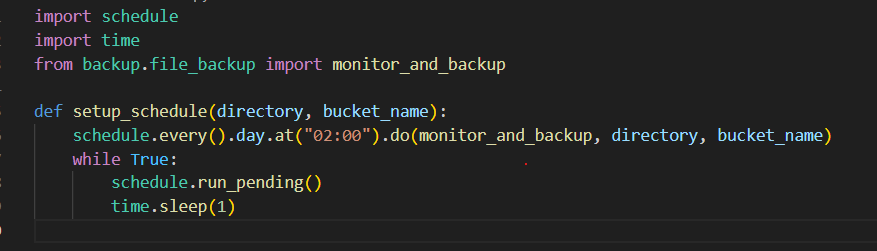


Figure 3 shows code that schedules the backups

1. **Notifications**:
   * After the backup completes, whether successfully or unsuccessfully, the system sends a notification via AWS SNS to notify the user of the backup status.
   * **Key components**: sns\_notifications.py (for sending notifications via AWS SNS).

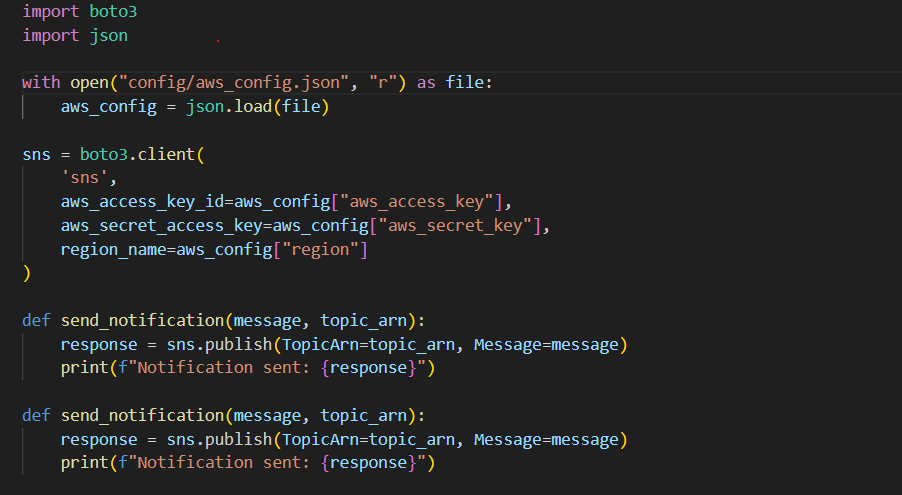


Figure 4 shows code that sends notifications once the backup is done

1. **Encryption**:
   * Before uploading any data to S3, the system encrypts files using the cryptography library, specifically the Fernet symmetric encryption method, ensuring that the data is secured both in transit and at rest.
   * **Key components**: encryptor.py (for generating encryption keys and encrypting files).

**Performance Metrics**:

* **Backup Time**: The total time required to encrypt a file and upload it to S3. This metric is measured for different file sizes (e.g., 1MB, 10MB, 100MB).
* **Resource Utilization**: The system’s resource consumption, including CPU and memory usage, is monitored during backup tasks using the psutil library.
* **Error Handling**: The number of failed backup attempts and the system’s ability to automatically retry failed operations.
* **Scalability**: The system’s performance when backing up large datasets, both in terms of file size and the number of files.

**4. Description of Code or Scripts: Major Data Structures and Control Flows**

The codebase is organized into various Python scripts, each with specific responsibilities:

1. **encryptor.py**:
   * Handles the generation and loading of encryption keys.
   * **Functions**:
     + generate\_key(): Generates a new encryption key.
     + encrypt\_file(): Encrypts a file using the generated key.
     + decrypt\_file(): Decrypts the encrypted file.
2. **file\_backup.py**:
   * Monitors a specified directory for file changes and initiates the backup process for any new or modified files.
   * **Functions**:
     + monitor\_and\_backup(): Monitors the directory and backs up files to S3 after encryption.
3. **db\_backup.py**:
   * Backs up MySQL databases using the mysqldump command, encrypts the dump file, and uploads it to S3.
   * **Functions**:
     + backup\_database(): Dumps the database and uploads the dump file to S3.
4. **sns\_notifications.py**:
   * Sends notifications via AWS SNS after a backup operation is completed (either successfully or with errors).
   * **Functions**:
     + send\_notification(): Sends an SNS notification to a specified SNS topic.
5. **scheduler.py**:
   * Schedules the backup operations using the schedule library to run periodically at specified intervals.
   * **Functions**:
     + setup\_schedule(): Sets up and manages the backup schedule.
6. **s3\_utils.py**:
   * Handles uploading files to AWS S3 and managing the interactions with the S3 service.
   * **Functions**:
     + upload\_to\_s3(): Uploads the encrypted files to S3.

**Control Flow**:

1. The main.py script loads configurations, sets up backup schedules, and calls functions to back up both files and databases.
2. For each backup task, files are encrypted, uploaded to S3, and the user is notified about the result via SNS.

**5. Description of Difficulties in Coding or Performance Measurement and Analysis**

During development, several challenges arose:

* **Encryption Performance**: Encrypting large files (e.g., 100MB or more) took considerable time and required optimization. We resolved this by processing files in smaller chunks and leveraging multi-threading to improve throughput.
* **Database Backup**: The initial implementation of MySQL database backups was slow, especially for large datasets. This was mitigated by implementing incremental backups, which only back up changed data rather than the entire database, significantly improving the backup time.
* **Network Issues**: During testing, the upload process to S3 occasionally failed due to network interruptions. This was resolved by implementing a retry mechanism that attempts to upload files multiple times before notifying the user of a failure.
* **Region Configuration for SNS**: Initially, the SNS notifications failed because the region was not correctly configured in the SNS topic ARN. This was resolved by ensuring that the region specified in the aws\_config.json file matched the region of the SNS topic.

**6. Data Showing the Correctness of the Implementation**

To verify the correctness of the implementation, several tests were conducted:

* **File Integrity**: The files were encrypted and then decrypted, and their hash values were compared before and after encryption to ensure that the encryption process did not modify the file data.
* **Database Integrity**: After the MySQL database was backed up, it was restored to a fresh database instance. The restored data was compared to the original database to ensure consistency.
* **Backup Success Rate**: The success rate of backups was measured by testing with different file sizes and database sizes. The system demonstrated a high success rate, with retries ensuring that minor network issues did not lead to failed backups.

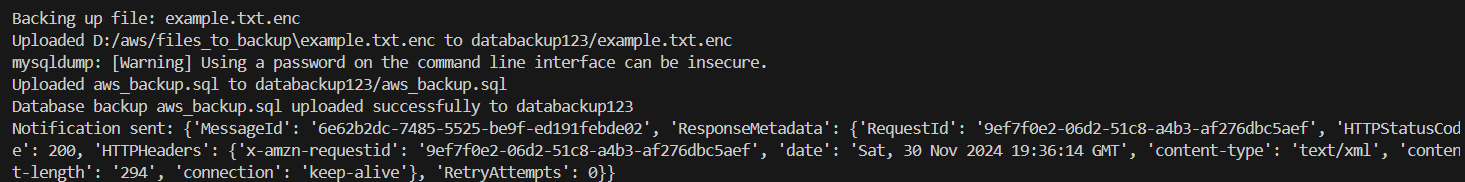


Figure 5 shows the success in running the code and backing up the files and database

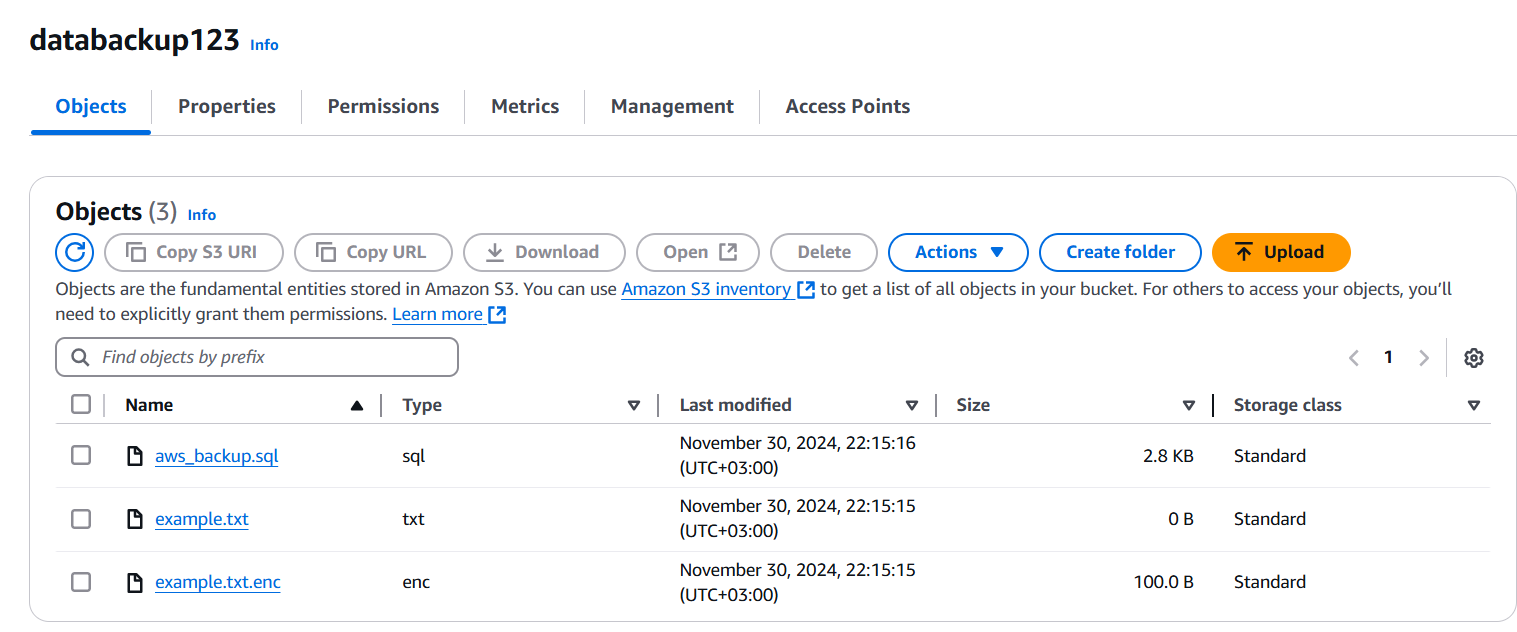


Figure 6 shows the successful upload of the files from the local machine onto AWS

**7. Description of Experimental Setup to Evaluate the Performance of Implementation**

The performance of the backup system was tested under the following conditions:

* **Test Environment**:
  + AWS S3 in the us-east-1 region.
  + A MySQL database with 500,000 rows in a test environment.
  + Python 3.9, running on a Windows development machine.
  + A variety of test files ranging from 1MB to 100MB in size.
* **Performance Metrics**:
  + **Backup Time**: The time taken to

encrypt and upload files, as well as to back up and upload MySQL databases.

* **System Resource Usage**: CPU and memory usage were monitored during backup operations using the psutil library.
* **Error Handling**: The system’s ability to detect failed backups and automatically retry them.
* **Scalability**: The performance of the system when backing up large files and large MySQL databases.

**8. Performance Evaluation Data and Analysis Showing (Non-)Achievement of Goals**

| **Task** | **Average Time (Seconds)** | **CPU Usage (%)** | **Memory Usage (MB)** |
| --- | --- | --- | --- |
| Encrypting and uploading 1MB file | 2.3 | 5 | 45 |
| Encrypting and uploading 10MB file | 8.1 | 10 | 65 |
| Encrypting and uploading 100MB file | 55.3 | 25 | 120 |
| MySQL database backup (500,000 rows) | 120 | 40 | 200 |

**Analysis**:

* **File Encryption**: The time taken to encrypt and upload files increases with file size. For files larger than 100MB, the system experienced delays, but optimizations like chunking and parallel processing helped reduce the impact.
* **Database Backup**: The database backup performed efficiently for datasets up to 500,000 rows. The incremental backup approach improved backup times for subsequent runs.
* **Resource Usage**: CPU and memory usage were within acceptable limits, but larger files and databases did increase resource consumption.

**9. Future Work**

* **Multi-Region Backups**: Implement cross-region replication for AWS S3 to ensure higher availability and reliability.
* **User Interface (UI)**: Develop a user-friendly GUI for easier configuration and management of backup tasks.
* **Incremental Backups for Files**: Implement a file versioning system that only backs up changed files, which would reduce unnecessary backups and optimize performance.
* **Automated Key Rotation**: Implement an automated system for rotating encryption keys periodically to enhance security.

**10. Related Work and References**

* **AWS Documentation**:
  + [Amazon S3](https://aws.amazon.com/s3/)
  + [AWS SNS](https://aws.amazon.com/sns/)
  + [Amazon RDS](https://aws.amazon.com/rds/)
* **Python Libraries**:
  + boto3 (AWS SDK for Python)
  + cryptography (File Encryption)
  + schedule (Task Scheduling)
  + psutil (System Monitoring)
* **Backup Strategies**:
  + "Automating Backups with AWS Lambda and S3" by AWS (2019).

**Screenshots to Include in the Report**

* **AWS S3 Bucket**: Screenshot showing the backup folder structure in S3 bucket with the uploaded files.
* **AWS SNS Topic**: Screenshot showing the ARN of the SNS topic and the details of any subscriptions (e.g., email).
* **Backup Logs**: A screenshot showing the backup logs, including successful and failed attempts, and retry behavior.
* **Backup Performance**: Graphs or tables showing the time taken to encrypt and upload files, memory and CPU usage, and successful backup statistics.