Software Design Document

Project Group P1:

1. **Likhitesh A L**
2. **K V Sai Rohith**
3. **Yash Buty**
4. **Mandar Pimparkar**
5. **Neha Waikar**
6. **Yash Gharde**

Creating SNS for S3 Bucket

User Story:

As a team, we should create a notification service using Amazon SNS, so that we get notified once the data is uploaded by the client inside the bucket.

Overview:

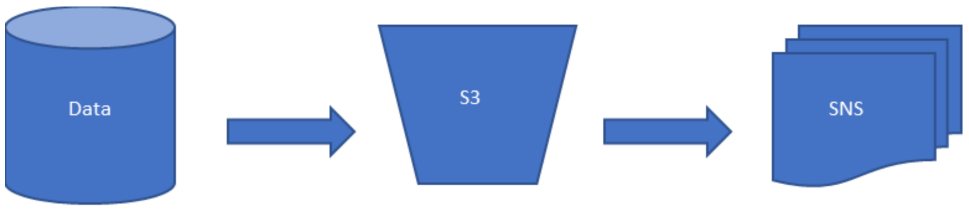
Our team needs to be notified once the client uploads their data to the S3 bucket. We will use Amazon SNS to create a notification service that sends a notification to our team when the client uploads their data to the S3 bucket.

Solution:

We will use the following steps to create a notification service using Amazon SNS:

1. Create an Amazon SNS topic for our team to subscribe to.
2. Create an S3 bucket event that triggers the SNS topic when the client uploads their data to the S3 bucket.
3. Subscribe our team to the SNS topic to receive notifications when the S3 bucket event is triggered.

Flowchart:



Design:

1. We will create an Amazon SNS topic for our team to subscribe to.
2. When the client uploads their data to the S3 bucket, an S3 bucket event will be triggered that sends a notification to the SNS topic.
3. Our team will be subscribed to the SNS topic to receive notifications when the S3 bucket event is triggered.

Security:

We will ensure the security of the system by implementing the following measures:

1. Implement secure access controls to restrict access to the S3 bucket and SNS topic and prevent unauthorized access.

2. Use encryption to protect the IAM access key and secret access key.

Scalability:

The system will be designed to be scalable to meet the client's requirements for uploading their data. We will choose a storage class that meets the client's requirements for durability, availability, and performance, and configure the S3 bucket with lifecycle policies to automatically move data between storage classes based on its age and usage.

Monitoring:

We will monitor the system using AWS CloudWatch metrics, which provide real-time monitoring of system metrics such as number of requests, latency, and errors. We will also monitor the S3 bucket access logs and SNS topic delivery status logs to ensure that the system is functioning correctly and identify any issues.

Creating PySpark Script to analyze the data

User Story:

As a team, we should be able to write a PySpark script for analyzing the data according to user requirements.

Overview:

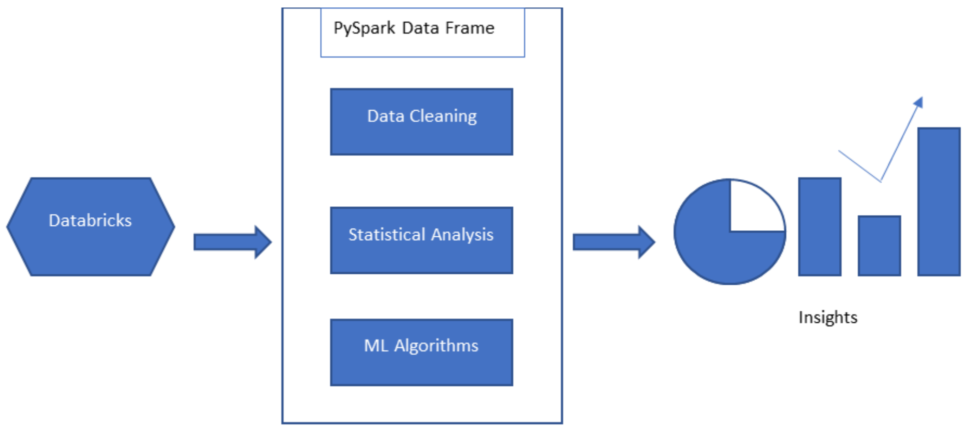
Our team needs to write a PySpark script for analyzing the client data. We will use PySpark, which is a distributed computing framework, to process large volumes of data in parallel and generate insights that meet the user's requirements.

Solution:

We will use the following steps to write a PySpark script for analyzing the data according to user requirements:

1. Load the data from the S3 bucket into Databricks environment.
2. Apply data cleaning and preprocessing techniques to ensure data quality.
3. Apply statistical analysis techniques to extract insights from the data.
4. Apply machine learning algorithms to build predictive models that meet the user's requirements.
5. Generate visualizations and reports to present the insights to the user.

Flowchart:



Design:

We will design the PySpark script for analyzing the data according to user requirements as follows:

1. Load the data from the S3 bucket into a PySpark Data Frame using the Pyspark script.
2. Use PySpark's data cleaning and preprocessing libraries such as PySpark SQL and PySpark Data Frames to clean and preprocess the data.
3. Use PySpark's visualization library such as PySpark SQL and PySpark Data Frames to generate visualizations and reports to present the insights to the user.

Security:

We will ensure the security of the system by implementing the following measures:

1. Implement secure access controls to restrict access to the S3 bucket and PySpark script and prevent unauthorized access.
2. Implement authentication and authorization measures to ensure that only authorized users can access the data and PySpark script.

Scalability:

The system will be designed to be scalable to meet the client's requirements for processing and analyzing their data. We will use PySpark's distributed computing capabilities to process large volumes of data in parallel and ensure high performance.

Monitoring:

We will monitor the system using AWS CloudWatch metrics, which provide real-time monitoring of system metrics such as CPU usage, memory usage, and disk usage. We will also monitor the PySpark application logs to ensure that the system is functioning correctly and identify any issues.

Creating UI and Connecting it to S3 Bucket

User Story:

As a client, I should be able to upload and retrieve data using UI so that the team can analyze and give processed data for making good business decisions.

Overview:

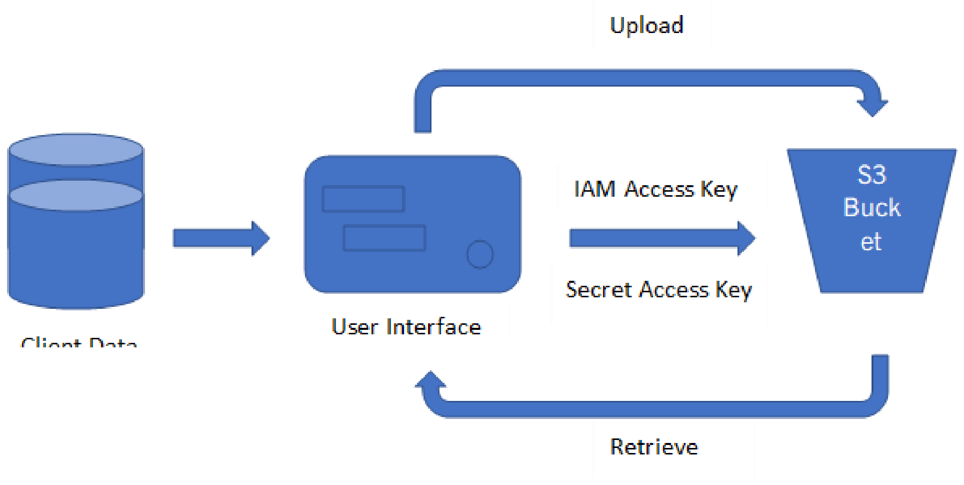
Our client needs an easy-to-use UI to upload and retrieve their data, so that our team can analyze the data and provide processed data for making good business decisions. We need a solution to design a user interface that allows the client to upload and retrieve their data.

Solution:

We will use the following steps to create a user interface for uploading and retrieving data:

1. Design a user interface that allows the client to upload their data to the S3 bucket.
2. We are connecting S3 bucket with the UI using IAM access key and secret access key.
3. When the client clicks the upload button, the file will get uploaded, the data will be stored directly in the S3 bucket.
4. Using the same user interface, the client should be able to retrieve the processed file from S3 bucket.

Flowchart:



Design:

1. We will design a user interface that allows the client to upload their data to the S3 bucket using the Upload button.
2. Once the upload process is complete, the data will be stored directly in the S3 bucket.
3. We will also design functionality on the same user interface that allows the client to retrieve the processed file.

Security:

We will ensure the security of the system by implementing the following measures:

1. Implement secure access controls to restrict access to the S3 bucket and prevent unauthorized access.
2. Use encryption to protect the IAM access key and secret access key.

Scalability:

The system will be designed to be scalable to meet the client's requirements for uploading and retrieving their data. We will choose a storage class that meets the client's requirements for durability, availability, and performance, and configure the S3 bucket with lifecycle policies to automatically move data between storage classes based on its age and usage.

Monitoring:

We will monitor the system using AWS CloudWatch metrics, which provide real-time monitoring of system metrics such as number of requests, latency, and errors. We will also monitor the S3 bucket access logs to record all requests made to the S3 bucket and monitor the logs for any suspicious activity.