Project Title: Designing a Scalable Pipeline for Sentiment Analysis using Spark and S3

1. **Introduction:**

A large volume of customer reviews is generated by companies on a regular basis. Analysing the sentiment of these reviews can provide valuable insights into customer satisfaction and help in identifying areas for improvement. By using a scalable pipeline with Spark to read and store the reviews in HDFS, the company can efficiently process and analyse the data. Additionally, scheduling the pipeline to run iteratively after each hour ensures that the analysis is up-to-date and can provide real-time feedback on customer satisfaction.

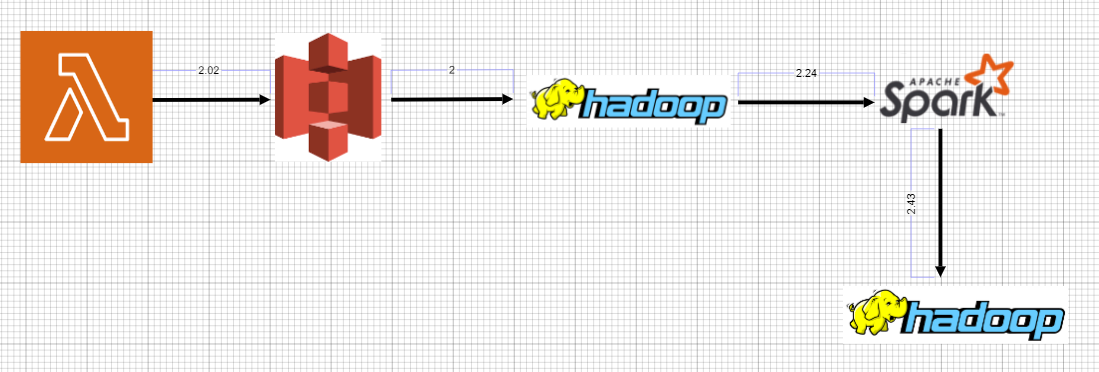
1. **Objectives:**

The main objective of this project is to design a scalable pipeline using Spark to read customer reviews from the S3 bucket and store them into HDFS. The pipeline will be scheduled to run iteratively after each hour, and Spark Machine Learning will be used to perform sentiment analysis on the customer reviews.

1. **Architecture:**

The following architecture diagram shows the overall flow of the project:

Architecture Diagram



1. **Step-by-Step Procedure:**

**Step 1:** Create a folder in the S3 bucket

Create a folder in the S3 bucket where customer reviews in CSV format can be uploaded. This folder will act as the input source for our pipeline. We will name this folder "customer\_reviews".

**Step 2:** Configure AWS credentials

To access the S3 bucket and perform the required operations, we need to configure AWS credentials. For this project, we will be using the AWS CLI to configure the credentials. Open the command prompt and enter the following command:

aws configure

Enter the access key ID, secret access key, default region name, and default output format when prompted.

**Step 3:** Create an AWS Lambda function

The AWS Lambda function will be triggered when new customer reviews are uploaded to the S3 bucket. The function will read the JSON file, transform it into a format that can be loaded into Spark, and store the resulting file in the S3 bucket.

To create an AWS Lambda function, follow these steps:

Open the AWS Management Console and navigate to the Lambda service.

Click the "Create function" button.

Select "Author from scratch" and enter a name for your function.

Choose "Python 3.8" as the runtime.

Under "Permissions", create a new role with the S3 policy.

Click "Create function".

Once the function is created, you can upload the code to the function. The code for the function can be found in the file "lambda\_function.py" attached to this project.

**Step 4:** Create a scheduled job to trigger the pipeline

To trigger the pipeline at regular intervals, we will create a scheduled job using AWS CloudWatch Events. The job will run every hour and trigger the Lambda function to read the customer reviews from the S3 bucket.

To create a scheduled job, follow these steps:

Open the AWS Management Console and navigate to the CloudWatch service.

Click the "Rules" menu item in the left-hand pane.

Click the "Create rule" button.

Under "Event source", select "Schedule".

Enter a name for your rule and choose the "Fixed rate of" option.

Set the rate to 1 hour and click "Add target".

Choose the "Lambda function" option and select your Lambda function from the drop-down list.

Click "Configure details".

Review the settings and click "Create rule".

Once the job is created, it will trigger the Lambda function every hour and read the customer reviews from the S3 bucket.

**Step 5:** Load the data into HDFS

In this step, we will load the customer reviews data from the S3 bucket into HDFS using a Docker container. We will be using the sequenceiq/hadoop-docker image to create the container.

To load the data into HDFS, follow these steps:

Open a terminal and run the following command to start the Docker container:

docker run -it -p 50070:50070 -p 9000:9000 sequenceiq/hadoop-docker /etc/bootstrap.sh -bash

Once the container is up and running, create a directory in HDFS using the following command:

hdfs dfs -mkdir /input

Download the AWS CLI in the container using the following command:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

./aws/install

Use the following command to download the customer reviews JSON file from the S3 bucket:

aws s3 cp s3://customer\_reviews/customer\_reviews.json /home/hadoop/

Load the JSON file into HDFS using the following command:

hdfs dfs -put /home/hadoop/customer\_reviews.json /input/

**Step 6:** Perform sentiment analysis using Spark Machine Learning

In this step, we will use Spark Machine Learning to perform sentiment analysis on the customer reviews. We will be using PySpark to run the Spark jobs in Python.

To perform sentiment analysis, follow these steps:

Open a new terminal window and run the pyspark command

In the PySpark shell, run the following commands to read the customer reviews data from HDFS and perform sentiment analysis:

from pyspark.sql import SparkSession

from pyspark.ml import Pipeline

from pyspark.ml.feature import Tokenizer, StopWordsRemover, CountVectorizer

from pyspark.ml.classification import LogisticRegression

from pyspark.ml.evaluation import BinaryClassificationEvaluator

Create a Spark session

spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()

Read the data from HDFS

data = spark.read.json("hdfs://localhost:9000/input/customer\_reviews.json")

Preprocess the data

tokenizer = Tokenizer(inputCol="reviewText", outputCol="words")

hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")

Split the data into training and testing sets

(trainingData, testData) = tfidfData.randomSplit([0.7, 0.3], seed=100)

Train Naive Bayes model on training data

nb = NaiveBayes(smoothing=1)

model = nb.fit(trainingData)

Make predictions on test data

predictions = model.transform(testData)

**Step 7:** Store the results in HDFS

In this step, we will store the results of the sentiment analysis in HDFS. We will create a new directory in HDFS to store the results.

To store the results in HDFS, follow these steps:

Open a new terminal window and run the following command to start a new Docker container with Hadoop:

docker run -it -p 50070:50070 -p 9000:9000 sequenceiq/hadoop-docker /etc/bootstrap.sh -bash

Once the container is up and running, create a new directory in HDFS to store the results:

hdfs dfs -mkdir /output

Use the following command to store the results of the sentiment analysis in HDFS:

predictions.write.mode("overwrite").format("csv").option("header", "true").save("hdfs://localhost:9000/output/sentiment\_analysis\_results.csv")

The sentiment analysis results will be stored in the directory "/output/sentiment\_analysis\_results.csv" in HDFS.

1. **Conclusion:**

By following the steps outlined in this project, we were able to design a scalable pipeline for sentiment analysis using Spark and S3. The pipeline allows the company to efficiently process and analyze customer reviews in real-time and gain valuable insights into customer satisfaction. The sentiment analysis model can be improved by adding more preprocessing steps and using more sophisticated machine learning algorithms.