**Project 1: Building a Real-Time Data Pipeline with Snowflake**

**Project Overview:**

Please design and implement a real-time data pipeline using Snowflake as the data warehouse. The goal is to ingest, process, and analyze streaming data in real-time, providing valuable insights to end-users.

Project Steps:

Data Source Selection: Identify a suitable data source to simulate streaming data. This could be a web service, IoT devices, or any other data generator that can produce a continuous stream of data. For example, you could use a public API that provides real-time weather data.

Data Ingestion: Set up an AWS service, such as Amazon Kinesis Data Streams or Amazon Kinesis Data Firehose, to ingest the streaming data. Configure the data source to send data to the selected Kinesis service.

Data Transformation: Use AWS Lambda or Amazon Kinesis Data Analytics to process and transform the incoming data into a format suitable for storage in Snowflake. You may need to perform data enrichment, filtering, or aggregation based on the specific use case.

Snowflake Configuration: Set up a Snowflake instance on AWS and create the necessary tables and schema to store the transformed data. Ensure that the Snowflake instance is appropriately scaled to handle the data volume.

Data Loading: Develop a process to load the transformed data from the streaming service into Snowflake. Snowflake provides several options for data loading, such as Snowpipe, COPY INTO, or using external stages.

Real-Time Analytics: Implement real-time analytics using Snowflake's capabilities. You can create views, materialized views, or scheduled tasks to continuously update aggregates or summaries of the data for faster querying.

Visualization and Dashboarding: Connect a business intelligence tool like Tableau or Amazon QuickSight to Snowflake to create interactive dashboards and visualizations. This step will help end-users interact with the real-time data and gain insights from it.

Monitoring and Error Handling: Set up monitoring and alerting for the data pipeline. Monitor data ingestion rates, data loading times, and any potential errors that might occur during the process. Implement appropriate error handling mechanisms.

**Summary** – Collected the weather data from an API URL, Configured it with API Gateway and processed it with AWS services like Lambda, Kinesis, S3, then visualized it using Amazon Quick Sight.

Below is the picture of weather data API URL in JSON Format.

A screenshot of a computer

Description automatically generated

**Below are the series of Steps in Detail with screenshots:**

**Step 1- IAM Role**

Created an IAM role for lambda and attached the policies as shown in the pictures below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Step-2 – Lambda**

Created a Lambda function attaching the above IAM role.

A screenshot of a computer

Description automatically generated

**Step-3 – Kinesis Streams**

Created a Kinesis Stream to load the API data into it.

A screenshot of a computer

Description automatically generated

**Step-4 – API Gateway**

Created a Gateway with a put option to load the data in to lambda by giving a trigger to it.

A screenshot of a computer

Description automatically generated

A screenshot of a search box

Description automatically generated

Below is the picture of connection between lambda and API Gateway.

A screenshot of a computer

Description automatically generated

**Step -5 – Connecting all the services in lambda with a code.**

Now once the connection is established, run the below code in Lambda, where it fetches the data from API, pushes it to Kinesis and then saves it to an S3 Bucket.

import boto3

import json

import urllib.request

def lambda\_handler(event, context):

# Initialize Kinesis and S3 clients

kinesis\_client = boto3.client('kinesis', region\_name='us-east-1')

s3\_client = boto3.client('s3')

# API endpoint URL

api\_url = "https://api.open-meteo.com/v1/gem?latitude=52.52&longitude=13.41&hourly=temperature\_2m"

# Fetch data from the API

with urllib.request.urlopen(api\_url) as response:

data = response.read().decode('utf-8')

# Parse the JSON data

decoded\_data = json.loads(data)

# Iterate through hourly data and push to Kinesis stream

for i, temperature in enumerate(decoded\_data['hourly']['temperature\_2m']):

timestamp = decoded\_data['hourly']['time'][i]

data\_record = {

'latitude': 52.52,

'longitude': 13.41,

'temperature': temperature,

'timestamp': timestamp

}

# Push data record to Kinesis stream

kinesis\_client.put\_record(

StreamName='weather\_data',

Data=json.dumps(data\_record),

PartitionKey=str(i) # Using index as the partition key

)

# Save data to S3 bucket

bucket\_name = 'hvdhsg'

file\_name = 'weather\_data.json'

s3\_client.put\_object(Bucket=bucket\_name, Key=file\_name, Body=data)

return {

"statusCode": 200,

"body": "Data pushed to Kinesis and saved to S3 successfully" }

Below is successful execution result in Lambda.

A screenshot of a computer

Description automatically generated

Below is the picture of S3 Bucket where data has been sent successfully.

A screenshot of a computer

Description automatically generated

**Step- 6 – QuickSight**

Create a database to Analize the weather data and then add the file from the S3 Bucket to the database.

Add the fields and graph as required, Below is the graph for the weather data from the API Url.

A screenshot of a computer

Description automatically generated

**Optional –** Since I don’t have full access to snowflake we were not able to establish connection b/w snowflake and lambda. If all access were there we can do the below to send the data to snowflake and then send it to Quick Sight for visualization.

If we want to store the data in Snowflake from S3.

Below code helps snowflake to collect the data from S3. This code will be added to the above Lambda code.

# Initialize the Snowflake REST API endpoint

snowflake\_endpoint = "https://ws94209.ca-central-1.aws.snowflakecomputing.com"

# Snowflake account credentials

snowflake\_credentials = {

"data": {

"ACCOUNT\_NAME": "SW56669",

"USERNAME": "MounikaSabbisetty",

"PASSWORD": "\*\*\*\*\*\*\*\*\*",

"AUTHENTICATOR": "SNOWFLAKE"

}

}

# Authenticate and obtain the session token

login\_data = json.dumps(snowflake\_credentials).encode('utf-8')

login\_request = urllib.request.Request(f"{snowflake\_endpoint}/session/v1/login-request", data=login\_data, method='POST')

with urllib.request.urlopen(login\_request) as login\_response:

login\_result = login\_response.read().decode('utf-8')

session\_token = json.loads(login\_result)["data"]["token"]

# Execute the Snowpipe ingestion process

snowpipe\_command = {"command": "COPY INTO my\_pipe"}

snowpipe\_headers = {"Authorization": f"Bearer {session\_token}"}

snowpipe\_data = json.dumps(snowpipe\_command).encode('utf-8')

snowpipe\_request = urllib.request.Request(f"{snowflake\_endpoint}/pipe/v1/execute", data=snowpipe\_data, headers=snowpipe\_headers, method='POST')

with urllib.request.urlopen(snowpipe\_request) as snowpipe\_response:

if snowpipe\_response.status == 200:

print("Snowpipe executed successfully")

else:

print("Error executing Snowpipe:", snowpipe\_response.read().decode('utf-8'))

return {

"statusCode": 200,

"body": "Lambda function executed successfully"

}

Once that is successful we can add the below codes in snowflake sql.

Snowflake Stage:

-- Create an external stage for Lambda data

CREATE OR REPLACE STAGE lambda\_stage

URL = 's3://hvdhsg/weather\_data.json'

CREDENTIALS = (

AWS\_KEY\_ID = 'AKIA5QYNWMNK7KHUJ6OT',

AWS\_SECRET\_KEY = 'R4XiobmCVYX03aULDdZh1XPHt8X7XoFoQte4mwja'

);

Snowflake Pipe:

-- Create a pipe to copy data from the external table to the target table

CREATE OR REPLACE PIPE weather\_data\_pipe

AUTO\_INGEST = TRUE

AS

COPY INTO weather\_data

FROM @lambda\_stage

FILE\_FORMAT = (TYPE = 'JSON')

MATCH\_BY\_COLUMN\_NAME = CASE\_INSENSITIVE

ON\_ERROR = CONTINUE;

The above helps us to add the data to Snowflake.

A screenshot of a computer

Description automatically generated

Above picture shows the database, Schema and table creation in Snowflake with sql script.

A screenshot of a computer

Description automatically generatedOnce the PIPE is successfully created in Snowflake, which is shown in the above picture, the data gets transferred from S3 to Snowflake.