Step-by-Step Workflow

July 24, 2025

End-to-End Data Streaming Pipeline with Kafka, AWS S3, Glue, and Athena **Prepared by Your Name**

Introduction

This document outlines the step-by-step process of building an end-to-end data streaming pipeline using Apache Kafka on AWS. The pipeline streams data from a local machine to a Kafka broker on an EC2 instance, stores it in Amazon S3, catalogs it with AWS Glue, and enables querying with Amazon Athena. This professional workflow is designed to be clear, reproducible, and scalable for enterprise use.

Step-by-Step Workflow

Set Up an EC2 Instance for Kafka Broker

- **Launch EC2 Instance**: Create an Amazon EC2 instance using Amazon Linux or Ubuntu with sufficient resources (e.g., t2.micro for testing).
- **Install Java**: Install Java, a prerequisite for Kafka and Zookeeper:

```
sudo yum install java-1.8.0-openjdk -y  % For Amazon Linux

```

- **Install Kafka**: Download and set up Apache Kafka:

```
wget https://archive.apache.org/dist/kafka/3.2.0/kafka_2.13-3.2.0.tgz
tar -xzf kafka_2.13-3.2.0.tgz
cd kafka_2.13-3.2.0

```

- **Configure Security Group**: Allow traffic on:
  - Port 9092 (Kafka broker)
  - Port 2181 (Zookeeper)
- **Start Zookeeper and Kafka**:

```
bin/zookeeper-server-start.sh config/zookeeper.properties &
bin/kafka-server-start.sh config/server.properties &

```

Connect to EC2 Instance via SSH

- **Access EC2**: Use the provided SSH key (.pem file) to connect from your local machine:

```
ssh -i "your-key.pem" ec2-user@your-ec2-public-ip

```

- **Verify Connection**: Ensure you can access the EC2 instance terminal.

Create a Kafka Topic

- **Create Topic**: On the EC2 instance, create a Kafka topic named `demo_test`:

```
bin/kafka-topics.sh --create --topic demo_test --bootstrap-server localhost:9092 --
    partitions 1 --replication-factor 1

```

- **Verify Topic**: Check the topic creation:

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092

```

Set Up a Python Producer on Local Machine

- **Install Libraries**: Install required Python libraries locally:

```
pip install kafka-python pandas

```

- **Create Producer Script**: Write a Python script to read from `indexProcessed.csv` and send data to the Kafka topic:

```
from kafka import KafkaProducer
from json import dumps
from time import sleep
import pandas as pd

```

```python
# Load CSV data
df = pd.read_csv('indexProcessed.csv')

# Initialize Kafka Producer
producer = KafkaProducer(
    bootstrap_servers=['your-ec2-public-ip:9092'],
    value_serializer=lambda x: dumps(x).encode('utf-8')
)

# Send data to Kafka
while True:
    data = df.sample(1).to_dict(orient="records")[0]
    producer.send('demo_test', value=data)
    sleep(1)

```

- **Run Producer**: Execute the script to stream data to Kafka.

Set Up a Kafka Consumer on EC2

- **Run Console Consumer**: On the EC2 instance, verify data reception:

```
bin/kafka-console-consumer.sh --topic demo_test --bootstrap-server localhost:9092 --
   from-beginning

```

- **Confirm Data Flow**: Ensure messages from the local producer are received.

Store Consumed Data in Amazon S3

- **Create Consumer Script**: Write a Python script to consume Kafka messages and upload to S3:

```python
from kafka import KafkaConsumer
import boto3
from json import loads

# Initialize Kafka Consumer
consumer = KafkaConsumer(
    'demo_test',
    bootstrap_servers=['your-ec2-public-ip:9092'],
    auto_offset_reset='earliest',
    value_deserializer=lambda x: loads(x.decode('utf-8'))
)

# Initialize S3 client
s3_client = boto3.client('s3')

# Consume and upload to S3
for message in consumer:
    data = message.value
    s3_client.put_object(
        Bucket='your-bucket-name',
        Key=f'data/message_{message.offset}.json',
        Body=str(data)
    )

```

- **Configure AWS Credentials**: Set up AWS CLI or environment variables for S3 access.
- **Run Consumer**: Execute the script to store data in S3.

Set Up AWS Glue Crawler

- **Create Crawler**: In AWS Glue, configure a crawler to scan the S3 bucket (e.g., `s3://your-bucket-name/data/`).
- **Run Crawler**: Execute the crawler to create a table in the AWS Glue Data Catalog.
- **Verify Table**: Confirm the table is available in the Glue Data Catalog.

Query Data Using Amazon Athena

- **Access Athena**: In the AWS Management Console, navigate to Amazon Athena.
- **Select Database**: Choose the database created by the Glue crawler.

- **Query Data**: Run SQL queries to analyze the data:

```sql
SELECT * FROM your_table_name LIMIT 10;

```

- **Verify Results**: Confirm the streamed data is queryable.

Architecture Overview

The pipeline follows this flow:

- Local Python producer streams CSV data to a Kafka topic (`demo_test`) on EC2.
- A Kafka consumer reads messages and uploads them to an S3 bucket.
- AWS Glue Crawler catalogs the S3 data into a table.
- Amazon Athena enables SQL-based querying of the cataloged data.

Conclusion

This pipeline demonstrates a robust, end-to-end data streaming solution using Kafka and AWS services. It is scalable and can be extended with automation or additional integrations as needed.