**Set Up AWS EC2 for Kafka Broker**

**Step 1**: Launch an EC2 Instance on AWS

* Choose an instance type that meets your resource needs (e.g., t2.medium or larger).
* Ensure that the security group allows traffic on ports 9092 (Kafka) and 9100 (Prometheus node exporter).

**Step 2**: Install Kafka on the EC2 Instance

1. Install Java (required by Kafka):

sudo yum install java-1.8.0-openjdk -y

1. Download and extract Kafka:

wget https://archive.apache.org/dist/kafka/2.8.0/kafka\_2.13-2.8.0.tgz

tar -xvzf kafka\_2.13-2.8.0.tgz

cd kafka\_2.13-2.8.0

**Step 3**: Start Zookeeper and Kafka

1. Start Zookeeper:

bin/zookeeper-server-start.sh config/zookeeper.properties

1. In another terminal, start Kafka:

bin/kafka-server-start.sh config/server.properties

**Step 4**: Create the Kafka Topic (test2)

bin/kafka-topics.sh --create --topic test2 --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

**Step 5**: Update the Security Group to Allow Inbound Traffic

* Make sure ports 9092 (Kafka) and 9100 (Prometheus node exporter) are open in the security group of the EC2 instance.

**Prometheus Node Exporter Setup**

**Step 1**: Install Prometheus Node Exporter on the EC2 Instance

1. Download the Prometheus Node Exporter:

wgethttps://github.com/prometheus/node\_exporter/releases/download/v1.2.0/node\_exporter-1.2.0.linux-amd64.tar.gz

1. Extract the tarball:

tar -xvzf node\_exporter-1.2.0.linux-amd64.tar.gz

cd node\_exporter-1.2.0.linux-amd64

1. Run the Node Exporter:

./node\_exporter &

* This will expose system metrics on http://<ec2-public-ip>:9100/metrics.

**Kafka Producer Setup in Google Colab**

**Step 1**: Install Required Libraries

In Google Colab, install the necessary libraries for Kafka and HTTP requests:

!pip install kafka-python

**Step 2**: Configure the Producer

Update the configuration with the relevant information:

KAFKA\_BROKER = "3.94.214.180:9092" # Public IP of your EC2 instance

TOPIC = "test2" # Kafka topic to send data

NODE\_EXPORTER\_URL = "http://44.210.145.128:9100/metrics" # Node Exporter URL

**Step 3**: Implement the Kafka Producer Logic

Add the following code to fetch system metrics and send them to Kafka every 5 seconds:

**Producer.ipynb:**

from kafka import KafkaProducer

import requests

import time

import json

import socket

import re

# Kafka Configuration

KAFKA\_BROKER = "3.94.214.180:9092"

TOPIC = "test2"

HOSTNAME = socket.gethostname()

NODE\_EXPORTER\_URL = "http://44.210.145.128:9100/metrics"

producer = KafkaProducer(

bootstrap\_servers=KAFKA\_BROKER,

value\_serializer=lambda v: json.dumps(v).encode("utf-8")

)

def fetch\_metrics():

response = requests.get(NODE\_EXPORTER\_URL)

if response.status\_code != 200:

print(f"Error fetching metrics: {response.status\_code}")

return None

metrics = response.text.split("\n")

parsed\_metrics = {}

for line in metrics:

if line.startswith("#") or not line.strip():

continue

match = re.match(r'([\w:]+)\s+([\d.]+)', line)

if match:

key, value = match.groups()

parsed\_metrics[key] = float(value)

return parsed\_metrics

def extract\_relevant\_metrics(metrics):

return {

"cpuload": metrics.get("node\_cpu\_seconds\_total", 0),

"memory\_used": metrics.get("node\_memory\_MemTotal\_bytes", 0) - metrics.get("node\_memory\_MemFree\_bytes", 0),

"process\_memory": metrics.get("node\_memory\_Active\_bytes", 0),

"page\_cache": metrics.get("node\_memory\_Cached\_bytes", 0),

"buffer\_cache": metrics.get("node\_memory\_Buffers\_bytes", 0),

"uptime": metrics.get("node\_time\_seconds", 0),

"disk\_write": metrics.get("node\_disk\_written\_bytes\_total", 0),

"disk\_read": metrics.get("node\_disk\_read\_bytes\_total", 0),

"net\_out": metrics.get("node\_network\_transmit\_bytes\_total", 0),

"net\_in": metrics.get("node\_network\_receive\_bytes\_total", 0),

"logins": metrics.get("node\_users\_logged", 0),

"swap\_avail": metrics.get("node\_memory\_SwapFree\_bytes", 0),

"cpu\_frequency": metrics.get("node\_cpu\_frequency\_hertz", 0) / 1e6

}

def format\_metrics(metrics, hostname):

formatted = []

for key, value in metrics.items():

formatted.append(f"{hostname} : {key} = {value}")

return "\n".join(formatted)

while True:

raw\_metrics = fetch\_metrics()

if raw\_metrics:

metrics = extract\_relevant\_metrics(raw\_metrics)

formatted\_output = format\_metrics(metrics, HOSTNAME)

# Send to Kafka

producer.send(TOPIC, {"raw": metrics, "formatted": formatted\_output})

print(formatted\_output)

print("\n")

time.sleep(5)

**Step 4**: Run the Producer Code

Execute the producer script within the Colab environment to begin sending metrics to the Kafka broker

**5. Kafka Consumer Setup in Google Colab**

**Step 1**: Install the Consumer Library

Install the necessary libraries for Kafka consumption:

!pip install kafka-python

!pip install confluent\_kafka

**Step 2**: Implement the Consumer Logic

Add the following code to receive messages from Kafka:

**Consumer.ipynb:**

from kafka import KafkaConsumer

import json

KAFKA\_BROKER = "3.94.214.180:9092"

TOPIC = "test2"

consumer = KafkaConsumer(

TOPIC,

bootstrap\_servers=KAFKA\_BROKER,

value\_deserializer=lambda v: json.loads(v.decode("utf-8"))

)

for message in consumer:

print(f"Received metrics: {message.value}")

**Step 3**: Run the Consumer Code

Execute the consumer script in a separate Colab notebook or cell:

**Kafka Message Count Consumer**

**Step 1**: Install Required Libraries

!pip install confluent\_kafka

**Step 2**: Implement the Message Count Consumer

Add the following code to count the number of messages in the last minute:

from confluent\_kafka import Consumer, TopicPartition

from datetime import datetime, timedelta

KAFKA\_BROKER = '3.94.214.180:9092'

TOPIC = 'test2'

GROUP\_ID = 'message-counter-group'

def get\_message\_count\_last\_minute():

consumer = Consumer({

'bootstrap.servers': KAFKA\_BROKER,

'group.id': GROUP\_ID,

'auto.offset.reset': 'earliest',

'enable.auto.commit': False

})

metadata = consumer.list\_topics(timeout=10)

if TOPIC not in metadata.topics:

print(f"Topic '{TOPIC}' does not exist.")

consumer.close()

return

partitions = metadata.topics[TOPIC].partitions.keys()

one\_minute\_ago = int((datetime.now() - timedelta(minutes=1)).timestamp() \* 1000)

topic\_partitions = [TopicPartition(TOPIC, p, one\_minute\_ago) for p in partitions]

offsets\_for\_time = consumer.offsets\_for\_times(topic\_partitions)

total\_messages = 0

for tp in topic\_partitions:

offset\_info = next((o for o in offsets\_for\_time if o.topic == tp.topic and o.partition == tp.partition), None)

if offset\_info is not None and offset\_info.offset != -1:

low, high = consumer.get\_watermark\_offsets(TopicPartition(TOPIC, tp.partition))

total\_messages += high - offset\_info.offset

consumer.close()

return total\_messages

if \_\_name\_\_ == '\_\_main\_\_':

count = get\_message\_count\_last\_minute()

if count is not None:

print(f"Messages received in the last minute: {count}")

**Step 3**: Run the Message Count Consumer Script

Execute the message count consumer to track the number of messages within the last minute:

**Monitoring and Troubleshooting**

* **Logs and Error Handling**: Monitor logs for connection or deserialization issues.
* **Topic Management**: Use Kafka’s built-in utilities to manage topics.

bin/kafka-topics.sh --describe --topic test2 --bootstrap-server localhost:9092