WEEK 3 : Configuring to see the no of messages received within a time period

STEPS:

Kafka Setup:

1. Configure Zookeeper server in your VM.
2. Configure Kafka broker server in your VM.
3. Create a topic is not created before using the following command.

```
bin/kafka-topics.sh --create --topic demo_testing2 --bootstrap-server 51.20.105.68:9092 --
replication-factor 1 --partitions 1
```

Node Exporter Setup:

1. Download Node exporter in your local machine or in a EC2 Instance VM.

```
Wget https://github.com/prometheus/node_exporter/releases/download/v1.9.0/node_exporter-
1.9.0.linux-amd64.tar.gz
```

2. Unzip it using TAR.

```
tar xvfz node_exporter-1.9.0.linux-amd64.tar.gz
```

3. Run Node exporter

```
cd node_exporter-1.9.0.linux-amd64

 ./node_exporter
```

NOTE:

It requires LINUX as a subsystem if you want to expose your local machine metrics.

By default, Node Exporter listens on port 9100.

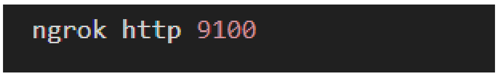
1. **Verify Metrics Exposure**:
   - Open a browser and navigate to http://localhost:9100/metrics to see the metrics being exposed.
2. If you are running it in your local machine, then you would need to expose the port 9100 using NGROK:

If you are running node exporter in a VM , remember the endpoint where the metrics is getting shipped:

Format :   <YOUR_EC2_Instance_PUBLIC_IP>/metrics

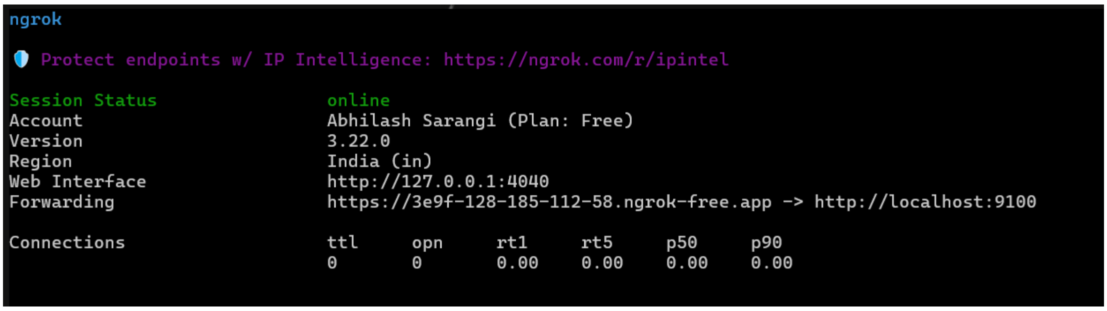
Steps:

1. Sudo apt install NGROK.
2. Configure NGROK key after registering from their website.
3. Execute Command :



```
ngrok http 9100
```

NGROK will give a endpoint for the port.

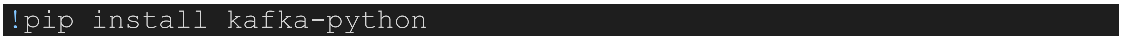


```
ngrok

🛡 Protect endpoints w/ IP Intelligence: https://ngrok.com/r/ipintel

Session Status                online
Account                       Abhilash Sarangi (Plan: Free)
Version                       3.22.0
Region                        India (in)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://3e9f-128-185-112-58.ngrok-free.app -> http://localhost:9100

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

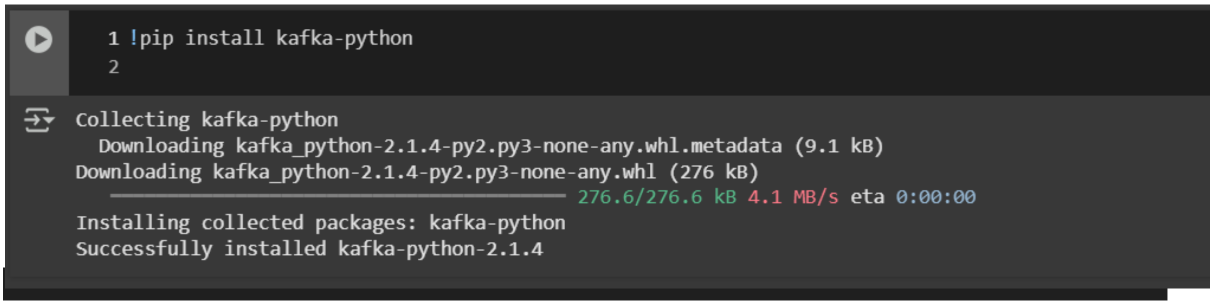
4. Note down the endpoint as we will use it in our kafka consumer script to fetch the metrics from.

Kafka Producer Script:

1. Open a google collab sheet.
2. Download kafka-python



```
!pip install kafka-python
```



```
!pip install kafka-python

Collecting kafka-python
  Downloading kafka_python-2.1.4-py2.py3-none-any.whl.metadata (9.1 kB)
Downloading kafka_python-2.1.4-py2.py3-none-any.whl (276 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 276.6/276.6 kB 4.1 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.1.4
```

```
from kafka import KafkaConsumer
import json

#Kafka consumer
consumer = KafkaConsumer(
    'demo_testing2',  # Topic name
    bootstrap_servers='51.20.105.68:9092',  # Kafka broker (EC2 public IP)
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    value_deserializer=lambda v: v.decode('utf-8')  # Decode as plain text
first
)

print("Listening for messages on topic 'demo_testing2'...")


for message in consumer:
    try:
        json_message = json.loads(message.value)
        print(f"Received JSON message: {json_message}")
    except json.JSONDecodeError:
        print(f"Received non-JSON message: {message.value}")
```

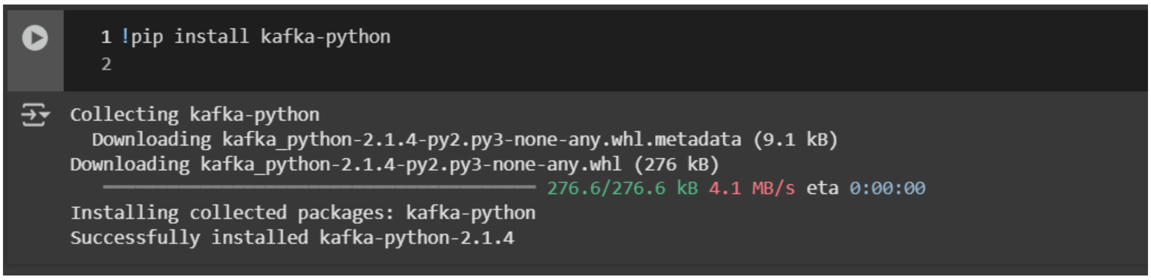
KafkaConsumer setup:

- Listens to topic demo_testing2.
- Connects to a Kafka broker running on the specified IP (51.20.105.68) and port 9092.
- auto_offset_reset='earliest': Starts reading from the beginning of the topic if no offset is stored.
- enable_auto_commit=True: Automatically commits offsets, marking messages as read.
- value_deserializer=lambda v: v.decode('utf-8'): Converts incoming byte data into readable strings.

Kafka Producer Script:

1. Open another google collab sheet.
2. Download kafka-python

```
!pip install kafka-python
```



```
!pip install kafka-python
```

```
Collecting kafka-python
  Downloading kafka_python-2.1.4-py2.py3-none-any.whl.metadata (9.1 kB)
Downloading kafka_python-2.1.4-py2.py3-none-any.whl (276 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 276.6/276.6 kB 4.1 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.1.4
```

3. Run the following script in another code block.

```
import time
from kafka import KafkaProducer
import requests
import json

# Kafka Producer Configuration
producer = KafkaProducer(
    bootstrap_servers='51.20.105.68:9092', # Replace with your Kafka broker address
    value_serializer=lambda v: json.dumps(v).encode('utf-8')   # Serialize dictionary to JSON
)

while True:
    # Fetch Node Exporter Metrics
    response = requests.get('https://12e2-128-185-112-58.ngrok-free.app/metrics')   # Replace with Node Exporter URL
    metrics = response.text

    # Send a subset of metrics to Kafka
    for line in metrics.splitlines():
        if line.startswith('node_cpu_seconds_total'):
            data = {"metric": line}   # Wrap metric in a dictionary
            producer.send('node_metrics', value=data)

    print("Metrics sent to Kafka topic 'node_metrics'")
    producer.flush()

    # Add delay to control frequency of fetching metrics
    time.sleep(10)   # Fetch metrics every 10 seconds
```

Explanation:

- **Kafka Broker**:
    1. Ensure the IP and port (51.20.105.68:9092) are accessible.
    2. Make sure security groups/firewall rules allow traffic.
- **Serialization**:
    1. value_serializer uses json.dumps to encode Python dicts into JSON (Kafka messages must be byte-encoded).

Metrics Fetching

- **Endpoint**:
    1. Metrics are pulled from a Prometheus Node Exporter exposed via Ngrok.
    2. URL: https://12e2-128-185-112-58.ngrok-free.app/metrics (replace with your actual endpoint in production).
- **Parsing**:
    1. Only metrics starting with node_cpu_seconds_total are processed and sent.
    2. Good for reducing noise from other metrics.

Kafka Publishing

- **Topic**: Messages are sent to Kafka topic 'node_metrics'.
- **Structure**: Each message is a JSON object like:
  {"metric": "node_cpu_seconds_total{...} 12345.0"}
- **Flush**: Ensures messages are not stuck in buffer before the next loop.

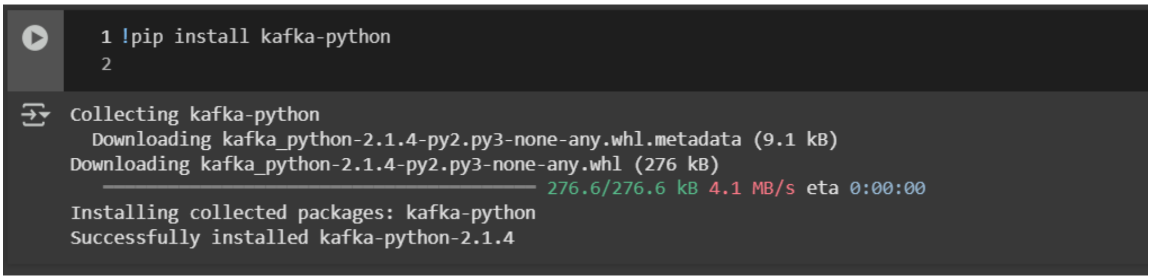
Timing

- **Loop Interval**:
    1. Metrics are fetched and sent every 10 seconds using time.sleep(10).
    2. You can tune this depending on your monitoring frequency needs.

Confluent Kafka Script to fetch the no of messages sent within a time period

1. Open another google collab sheet.
2. Download kafka-python

```
!pip install kafka-python
```



```
!pip install kafka-python

Collecting kafka-python
  Downloading kafka_python-2.1.4-py2.py3-none-any.whl.metadata (9.1 kB)
Downloading kafka_python-2.1.4-py2.py3-none-any.whl (276 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 276.6/276.6 kB 4.1 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.1.4
```

3. Run the following script

```
from confluent_kafka import Consumer, TopicPartition
from datetime import datetime, timedelta

# Kafka Configuration
KAFKA_BROKER = '51.20.105.68:9092'
TOPIC = 'node_metrics'
GROUP_ID = 'message-counter-group'

def get_message_count_last_minute():
    # Create a Kafka consumer
    consumer = Consumer({
        'bootstrap.servers': KAFKA_BROKER,
        'group.id': GROUP_ID,
        'auto.offset.reset': 'earliest',
        'enable.auto.commit': False
    })

    # Fetch metadata to get the partitions for the topic
    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()

    # Get the timestamp for 1 minute ago
    one_minute_ago = int((datetime.now() -
timedelta(minutes=1)).timestamp() * 1000)

    # Create TopicPartition objects with the timestamp
    topic_partitions = [TopicPartition(TOPIC, p, one_minute_ago) for
p in partitions]
```

```
    # Fetch offsets for the timestamp (1 minute ago)
    offsets_for_time = consumer.offsets_for_times(topic_partitions)

    # Calculate the number of messages in the last minute
    total_messages = 0
    for tp in topic_partitions:
        # Find the offset for the timestamp
        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:
            # Get the high watermark (latest offset)
            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}")
```

Explanation:

This script connects to a Kafka broker and counts how many messages were produced to the node_metrics topic in the **last minute**.

How it works:

1. **Connects to Kafka** and checks the topic exists.
2. Calculates the **timestamp for 1 minute ago**.
3. For each partition:
   - Gets the **offset** closest to that timestamp.
   - Gets the **latest offset** (high watermark).
   - Subtracts the two to find how many messages were added since.
4. **Sums up** messages from all partitions.

Finally, it prints the total message count in the last minute.