WEEK 3: Node Exporter → Kafka → Prometheus Setup with Message Counter

This document explains the integration of **Node Exporter**, **Kafka**, and **Prometheus** to collect, produce, consume, and count metrics from a Linux system. It includes:

- A **producer** that fetches Node Exporter metrics and sends them to Kafka.
- A **consumer** that reads metrics from Kafka.
- A **message counter script** that checks how many messages were sent in the last minute.

---

1. Kafka Producer Script

This script fetches `node_cpu_seconds_total` metrics from a Node Exporter instance and sends them to a Kafka topic named `node_metrics`.

`producer.py`

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

producer = KafkaProducer(
    bootstrap_servers='kafka_private_ip:9092',  # Replace with your Kafka
broker IP
    value_serializer=lambda v: json.dumps(v).encode('utf-8')
)

# Replace with the actual Node Exporter URL
while True:
    response = requests.get('http://54.226.0.3:9100/metrics')
    metrics = response.text

    for line in metrics.splitlines():
        if line.startswith('node_cpu_seconds_total'):
            data = {"metric": line}
            producer.send('node_metrics', value=data)

    print("Metrics sent to Kafka topic 'node_metrics'")
    producer.flush()
    time.sleep(10)  # Send metrics every 10 seconds
```

Run the Producer

```
python3 producer.py
```

---

2. Kafka Consumer Script

This script listens to the `node_metrics` topic and prints each metric consumed.

consumer.py

```
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer(
    'node_metrics',
    bootstrap_servers='kafka_private_ip:9092',  # Replace with your Kafka
broker IP
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    value_deserializer=lambda v: json.loads(v.decode('utf-8'))
)

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

try:
    for message in consumer:
        metric_data = message.value
        print(f"Received Metric: {metric_data}")
        metric_name = metric_data.get("metric", "N/A")
        print(f"Processed Metric: {metric_name}")
except KeyboardInterrupt:
    print("Consumer stopped.")
finally:
    consumer.close()
```

Run the Consumer

```
python3 consumer.py
```

---

3. Kafka Message Counter Script

This script counts how many messages were received in the last **1 minute** on the `node_metrics` topic.

Installation

```
pip install confluent-kafka
```

count_messages.py

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

KAFKA_BROKER = '51.20.105.68:9092'  # Replace with your Kafka broker IP
TOPIC = 'node_metrics'
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}")
```

Run the Script

```
python3 count_messages.py
```

OUTPUT SCREENSHOT