TP07 : Apache Kafka

kiram mohammed ali

Introduction

This report presents the practical work of TP07 on Apache Kafka, based on the official instructions provided in the TP document.[1] The goal is to understand distributed messaging, create Kafka topics, use producers and consumers, and build an IoT simulation system using Kafka.

1. Starting Zookeeper

Kafka requires Zookeeper to manage metadata and broker coordination.

Command

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

Screenshot

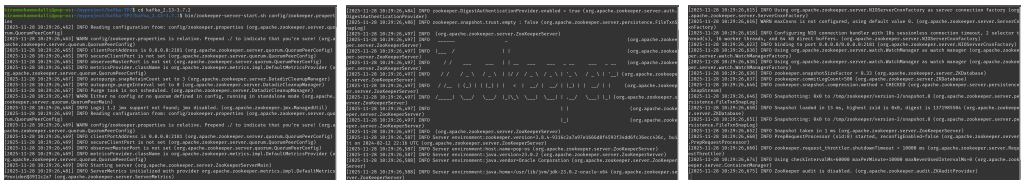

Figure 1: Zookeeper successfully started

[1] See TP07 PDF instructions.

2. Starting Kafka Broker

Command

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

Screenshot

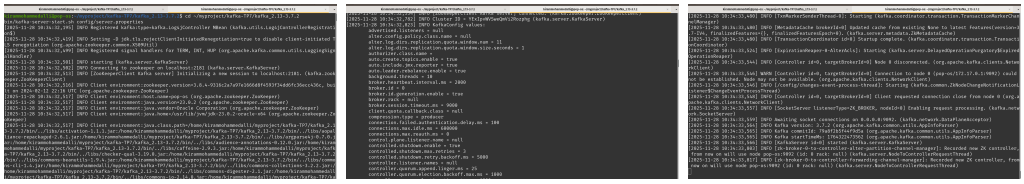

Figure 2: Kafka broker running

3. Creating a Kafka Topic

The topic used in this TP is named `my-first-topic`.

Create Topic

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

List Topics

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

Screenshot

Figure 3: Topic creation and verification

4. Producer - Sending Messages

Command

```
bin/kafka-console-producer.sh \
  --topic my-first-topic \
  --bootstrap-server localhost:9092
```

After running the command, messages like `hello kafka` are typed.

Screenshot

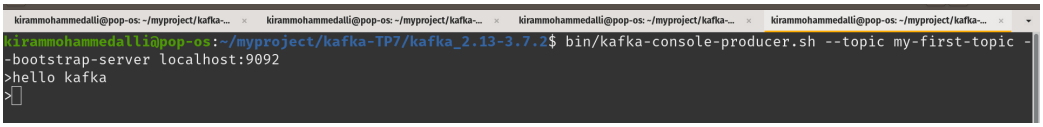

Figure 4: Kafka Console Producer sending messages

5. Consumer - Reading Messages

Command

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

Screenshot

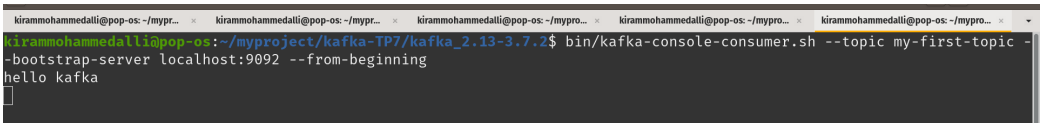

Figure 5: Kafka Console Consumer receiving messages

6. IoT + Kafka Project

According to the TP instructions,[2] a simulated IoT system sends machine temperature and vibration data to Kafka. Alerts are raised when thresholds are exceeded.

Producer Code (Python)

```
from kafka import KafkaProducer
import json
import random
import time

producer = KafkaProducer(
    bootstrap_servers='localhost:9092',
    value_serializer=lambda v: json.dumps(v).encode('utf-8')
)

def generate_machine_data(machine_id):
    return {
        "machine_id": machine_id,
        "temperature": round(random.uniform(40, 110), 2),
        "vibration": round(random.uniform(0, 5), 2),
        "status": "OK"
    }

if __name__ == "__main__":
    while True:
```

[2]See TP07 IoT section.

```
        for machine_id in range(1, 4):
            data = generate_machine_data(machine_id)
            print("Send:", data)
            producer.send('machines-data', data)
        time.sleep(1)
```

Producer Output

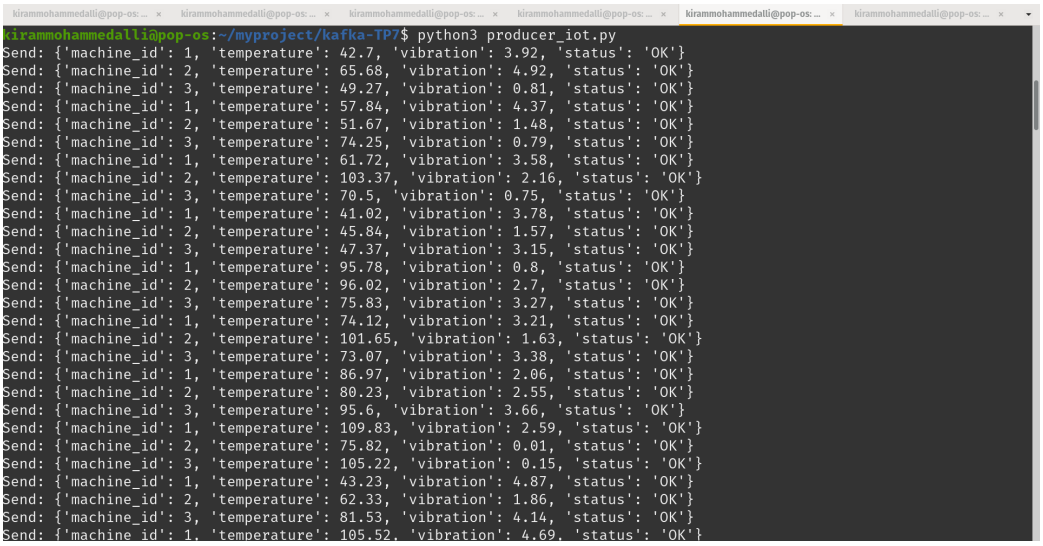

Figure 6: IoT Producer sending machine sensor data

Consumer Code (Python)

```
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer(
    'machines-data',
    bootstrap_servers='localhost:9092',
    value_deserializer=lambda m: json.loads(m.decode('utf-8'))
)

TEMP_LIMIT = 90
```

```
VIB_LIMIT = 3

print("Listening for machine data...")

for message in consumer:
    data = message.value
    machine_id = data["machine_id"]
    temp = data["temperature"]
    vib = data["vibration"]

    alert = []

    if temp > TEMP_LIMIT:
        alert.append(f"Temp haute: {temp}")
    if vib > VIB_LIMIT:
        alert.append(f"Vibration élevée: {vib}")

if alert:
    print(f"[WARNING] ALERTE machine {machine_id}: {', '.join(alert)}")
else:
    print(f"[OK] machine {machine_id} OK - temp={temp}, vib={vib}")
```

Consumer Output

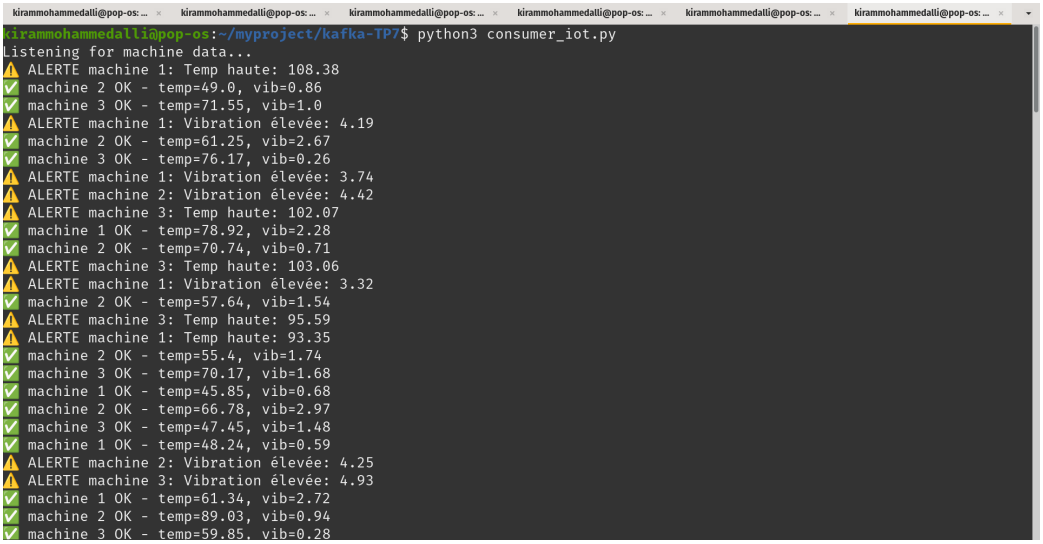

Figure 7: IoT Consumer generating alerts based on thresholds

Conclusion

This TP demonstrates the full workflow of Apache Kafka: starting Zookeeper, launching Kafka brokers, creating topics, using producers and consumers, and building an IoT real-time data processing system. Kafka proves to be a reliable and powerful distributed streaming platform for real-time analytics.