Kafka

Day 3

Day 3 - Overview

- Create a python kafka producer
- Interface the producer with REST API
- Create a python kafka consumer
- Interface the consumer with websocket
- Specify the partition at producer level
- Multi partition task

Sync your fork for Day 3 activities

Follow the below document to sync your fork and update local repository.

> *https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf*

Start Zookeeper

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

Start Kafka

.\bin\windows\kafka-server-start.bat .\config\server.properties

Setup and Start the application

In VSCode open the kafka day 3 code base.

C:\workspace\flask-microservices-training\kafka-day3\simple-kafka-flask-app

Open a terminal in VSCode and create a python virtual environment

python -m venv venv

Activate the Virtual Environment

.\venv\Scripts\activate

Install all the dependencies

pip install -r requirements.txt

Start Application

python app.py

Producer Code walk through

```
# Messages will be serialized as JSON
def serializer(message):
    return json.dumps(message).encode('utf-8')

# Kafka Producer
producer = KafkaProducer(
    bootstrap_servers=['localhost:9092'],
    value_serializer=serializer
)

def send(topic_name, message):
        # Send it to our 'messages' topic
        producer.send(topic_name, message, partition=0)
```

Consumer Code walk through

```
def read(topic_name):
    # To consume latest messages and auto-commit offsets
    consumer = KafkaConsumer(topic_name,
                             group_id='demo-group',
                             bootstrap_servers=['localhost:9092'])
    for message in consumer:
        # message value and key are raw bytes -- decode if necessary!
        # e.g., for unicode: `message.value.decode('utf-8')`
        print(f"topic: {message.topic}, partition: {message.partition}, offset: {message.offset},
key: {message.key}, value: {message.value}")
```

Task

Point the producer to second-topic which has 2 partitions

Update the producer REST API to take partition number in request arguments
Produce the message to partition specified in the request

Confirm in the listener that messages are consumer from appropriate partition

Q and A