Homework 3: Master Kafka

🔍 Objective

The purpose of this homework is to familiarize you with **Apache Kafka**, a distributed streaming platform. You'll learn how to set up Kafka on your Mac, create topics, and develop your own **producer** and **consumer** functions to interact with Kafka streams.

📌 Submission Guidelines

Each group will submit a document containing:

- The steps you followed during setup and execution.
- The **full source code** for both the producer and consumer scripts.
- A brief explanation of how your producer and consumer work.
- A short discussion of how your Kafka setup could be applied in a **real-world scenario**.

Assignment Steps:

Step 1: Install Kafka via Homebrew

1. **Open Terminal.**
2. If you haven't installed Homebrew yet, run:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

3. Update Homebrew with: `brew update`
4. Install Kafka: `brew install kafka`

Step 2: Set Kafka Path in Your Shell Profile

1. Find your Kafka installation path:

```
brew --prefix kafka
```

This usually returns something like /opt/homebrew/opt/kafka (Apple Silicon) or /usr/local/opt/kafka (Intel Macs).

2. Add Kafka's bin directory to your PATH by editing your shell profile file:
    - For Zsh (default on modern macOS):

```
echo 'export PATH="$(brew --prefix kafka)/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

    - For Bash:

```
echo 'export PATH="$(brew --prefix
kafka)/bin:$PATH"' >> ~/.bash_profile
source ~/.bash_profile
```

Step 3: Start ZooKeeper

Kafka needs ZooKeeper to manage brokers and metadata.

1. Open a new terminal window and start ZooKeeper:

```
zookeeper-server-start $(brew --prefix)/etc/kafka/zookeeper.properties
```

Step 4: Start Kafka Broker

a. Open another Terminal window or tab and run:

```
kafka-server-start $(brew --prefix)/etc/kafka/server.properties
```

Note here that by default, your Kafka server will be on `http://localhost:9092.`

Step 5: Create Kafka Topics

b. Create a topic named test with a single partition and replication factor of 1: `kafka-topics --create --topic test --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1`
c. Verify the creation of the topic with:

```
kafka-topics --list --bootstrap-server localhost:9092
```

Note here that you shouldn't create multiple topics in one command; just run the above command with the necessary edits each time you create a new topic.

🚀 Real-Time Application Design

Steps 6 and 7 are the **core of this homework**. Here, you'll use Kafka to simulate a **real-time application**.

I encourage you to go beyond "Hello, World!" and build something meaningful. You'll write a **producer** that continuously sends live or simulated data and a **consumer** that processes it in real time.

💡 Example ideas include:

- A **stock market feed** that streams simulated price data.
- A **clickstream app** that tracks user actions on a website.
- A **weather sensor simulator** streaming live or randomized readings.
- A **chat application**.
- A **machine learning pipeline** that classifies streaming text or predicts something simple based on input.

The more creative, the better!

Step 6: Write Your Own Kafka Producer Function

d. Objective: Write a Python script that acts as a Kafka producer. This script should connect to your Kafka instance and send messages to your topic (or topics).
e. Install the kafka-python library using pip: `pip install kafka-python`
f. Utilize the KafkaProducer object from the kafka-python library to set up the producer.
g. Configure the producer to connect to Kafka on localhost, port 9092. This is the default port used by Kafka for client connections:

```
from kafka import KafkaProducer
producer = KafkaProducer(bootstrap_servers='localhost:9092')
producer.send('test', b'Hello Kafka!')
producer.flush()
```

h. Use your creativity to create an interesting producer. You can choose to send string messages, serialized data, or even generate dynamic content based on some interesting API.

Step 7: Write Your Own Kafka Consumer

i. Write a Python/Pyspark script that acts as a Kafka consumer. This script should read the messages/data sent by your producer and do some processing on them, e.g., run machine learning algorithms.
j. Use the KafkaConsumer object from the kafka-python library to create the consumer. For example:

```
from kafka import KafkaConsumer
consumer = KafkaConsumer(
    'your_topic_name',
    bootstrap_servers=['localhost:9092'],
    auto_offset_reset='earliest',  # start reading at
the earliest message
    group_id='my-group' # consumer group ID
)
```

k. For PySpark, set up the streaming context and connect to Kafka (adjust parameters as needed):

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
        .appName("KafkaConsumerExample") \
        .getOrCreate()

df = spark \
     .readStream \
     .format("kafka") \
     .option("kafka.bootstrap.servers", "localhost:9092")
\
     .option("subscribe", "your_topic_name") \
     .load()
```

📦 Deliverables

A document containing:

- A detailed description of steps 6 and 7.
- The source code for both the producer and consumer scripts.
- A discussion on how these scripts can be used in a real-world application. Propose a simple application scenario where your Kafka setup could be utilized.