Case Study:

A company would like to setup a messaging system that automatically broadcasts to all new employees **in a specific order**.

The order of the messages is:

1. Welcome to Company XYZ!
2. Please remember to collect your laptop from the mobile clinic!
3. ANNOUNCEMENT: Stay safe and wash your hands.

Create a Kafka environment that simulates the following:Kafka Broker = Messaging System

Consumers = Employees

All new employees to the company should receive the announcement messages in the specified order above. **Whenever there is a new message, employees should only receive messages that they have never received before.**

Answer:

Below is the details and key step taken:

1. **On WSL, I ensured below dependencies has been installed:**
   - Java for Kafka: sudo apt install default-jdk -y
   - confluent-kafka for Python: pip install confluent-kafka
2. **Kafka version I used:**
   - https://archive.apache.org/dist/kafka/3.5.0/kafka_2.13-3.5.0.tgz
3. **Start Kafka Services:**
   - Start Zookeeper:

     ./kafka/bin/zookeeper-server-start.sh ./kafka/config/zookeeper.properties
   - Start Kafka broker:

     ./kafka/bin/kafka-server-start.sh ./kafka/config/server.properties
4. **Create Topics:**
   - General messages topic (for new employees):

     ./kafka/bin/kafka-topics.sh --create --topic general_messages --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
   - Department messages topic (for specific department *Optional Task):

     ./kafka/bin/kafka-topics.sh --create --topic department_messages --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

5. **Consumer Scripts is created to meet below key requirement:**

Key requirement: **Whenever there is a new message, employees should only receive messages that they have never received before**

a. **General Messages Consumer Script (*employee_general_consumer.py*):**

- Subscribe to the general_messages topic, for new employees.
- Standardize message content by removing extra whitespace and apply a hashing method to identify unique messages.
- Log processed messages in *processed_general_messages.json* to ensure employees receive only new and unique messages which they have never received before.

b. **Department Messages Consumer Script (*employee_department_consumer.py*):**

- Subscribe to the department_messages topic.
- Use the same deduplication method by hashing the message content.
- Log processed messages in *processed_department_messages.json*.

6. **Running the Consumers**

a. **Run Consumers in Separate Terminals:**

- Terminal#1

  (for general_messages):

  ./https://github.com/izzatazfar8/task-assessment/blob/main/kafka_assessment/employee_department_consumer.py

- Terminal#2

  (for department_messages):

  ./https://github.com/izzatazfar8/task-assessment/blob/main/kafka_assessment/employee_general_consumer.py

7. Send Messages on Producer Terminal

a. 2 terminals (general_messages & department_messages) will be open acting as a producer to test the consumer scripts.

b. For general_messages using below command:
./kafka/bin/kafka-console-producer.sh --topic general_messages --bootstrap-server localhost:9092
* The order of the messages is executed as shown in **Figure 1**. Employees won't be received the same messages that they have received before.

c. For department_messages using below command:
./kafka/bin/kafka-console-producer.sh --topic department_messages --bootstrap-server localhost:9092
* Group of messages sent to a specific department of people as shown in **Figure 2**.

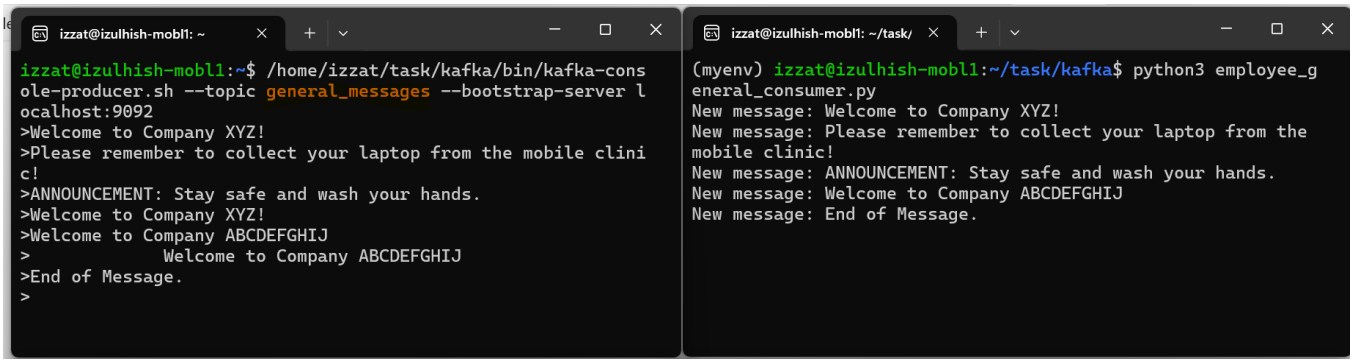

Figure 1: general_messages (new employee)

Optional Task (Bonus points): Create another group message that sends a different set of messages to a specific department of people.

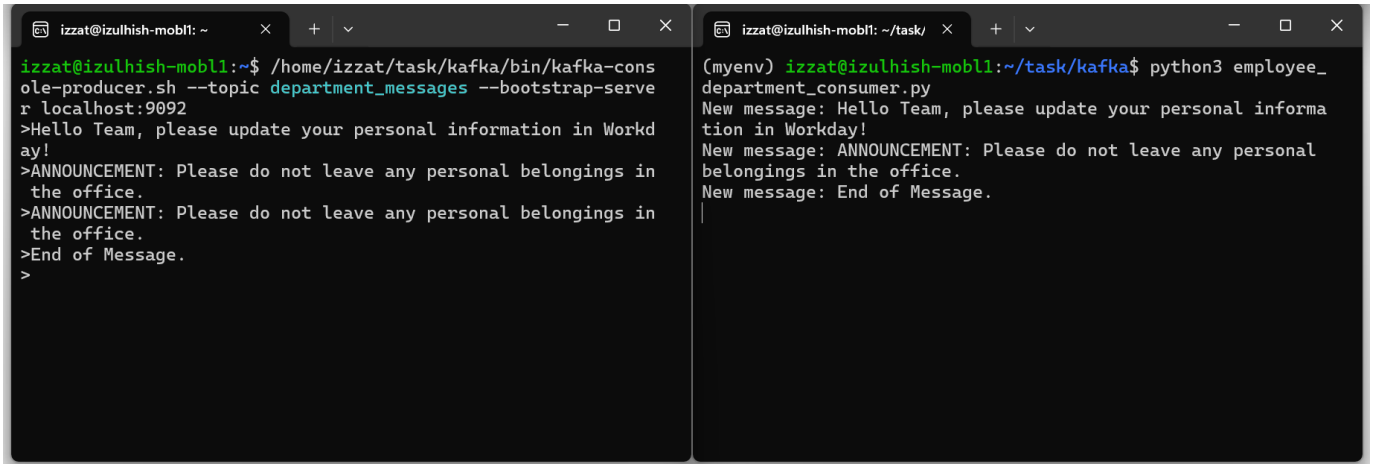

Figure 2: department_messages