**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**](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

1. **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

1. **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**

1. **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.
2. **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*.
3. **Running the Consumers**
4. **Run Consumers in Separate Terminals:**
   * Terminal#1

(for general\_messages):

python3 /home/izzat/employee\_general\_consumer.py



* + Terminal#2

(for department\_messages):

python3 /home/izzat/employee\_department\_consumer.py



1. **Send Messages on Producer Terminal**
2. 2 terminals (general\_messages & department\_messages) will be open acting as a producer to test the consumer scripts.
3. 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.

1. 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**.

**Screens screenshot of a computer

Description automatically generated**

**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.

Screens screenshot of a computer

Description automatically generated

**Figure 2:** department\_messages