

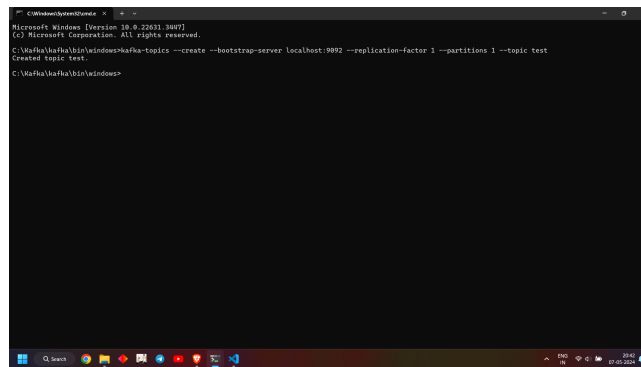
# Kafka Assignment Report

Taruna Sagar Mati (202318045)

May 7, 2024

## Topic Creation

In the process of setting up a Kafka environment for managing e-commerce order data, a crucial step involved the creation of a Kafka topic named "test". This topic serves as a channel for organizing and processing incoming order-related messages, ensuring efficient data handling and analysis within the e-commerce system.

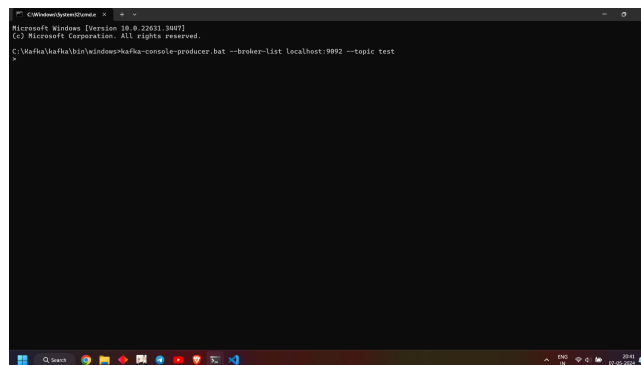


```
Microsoft Windows [Version 10.0.22H21.3847]
(c) Microsoft Corporation. All rights reserved.

C:\kafka\kafka\bin\windows>kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test
Created topic test.

C:\kafka\kafka\bin\windows>
```

Figure 1: test Initialization



```
Microsoft Windows [Version 10.0.22H21.3847]
(c) Microsoft Corporation. All rights reserved.

C:\kafka\kafka\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic test

C:\kafka\kafka\bin\windows>
```

Figure 2: Consumer Console

## Producer 1: Inventory Orders Producer

A Python script named `producer1.py` was created to manage inventory orders. JSON data from `data.json` was loaded and filtered using a custom function. A Kafka producer was initialized to send the filtered inventory messages to the designated Kafka topic, facilitating efficient inventory management within the system.

## Producer 2: Delivery Orders Producer

A Python script named `producer2.py` was developed to handle delivery orders. JSON data was loaded from `data.json` and filtered using a specific function. A Kafka producer was then initialized to dispatch the filtered delivery messages to the designated Kafka topic, streamlining order delivery management within the system.

## Consumer 1: Inventory Data Consumer

A Python script named `consumer1.py` was created to manage inventory data. It initialized a Kafka consumer to subscribe to the designated Kafka topic. The script processed incoming messages, filtering for inventory data, and executed actions to update inventory databases or systems accordingly.

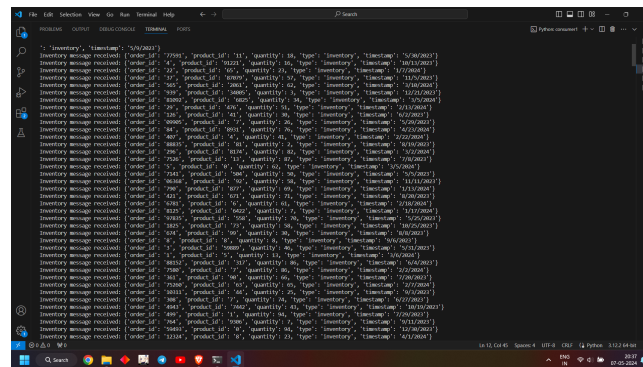


Figure 3: Consumer1 Result

## Consumer 2: Delivery Data Consumer

A Python script named `consumer2.py` was developed to oversee delivery tasks. It initialized a Kafka consumer to subscribe to the specified Kafka topic. The script processed incoming messages, filtering for delivery data, and executed actions such as scheduling deliveries, updating statuses, and notifying customers as necessary.

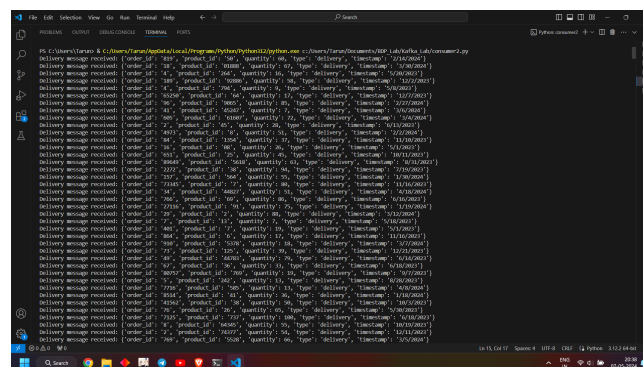


Figure 4: Consumer2 Result

## Filtering Logic

A Python script named `filterlogic.py` was created to showcase message filtering logic using the supplied JSON data. The script loaded JSON data from the provided file `data.json` and defined separate functions to filter inventory and delivery messages. It then printed the filtered inventory and delivery messages for verification purposes.

