Real-Time E-commerce Order Processing System Using Kafka

Name:-Harshil shah(202318033)

Executive Summary: This report details the creation of a Kafka-powered system designed to streamline the management of e-commerce orders in real-time. By employing Kafka producers and consumers, the system effectively manages inventory and expedites delivery processing, enhancing overall operational efficiency.

Introduction: In response to the growing demand for efficient order management in the e-commerce sector, a Kafka-based system has been developed to address key challenges in inventory management and delivery processing. This report provides an overview of the system's architecture, functionality, and benefits.

System Architecture: The Kafka-based system comprises producers responsible for generating order data and consumers tasked with processing and managing these orders. Orders are seamlessly transmitted through Kafka topics, ensuring real-time communication between various components of the system.

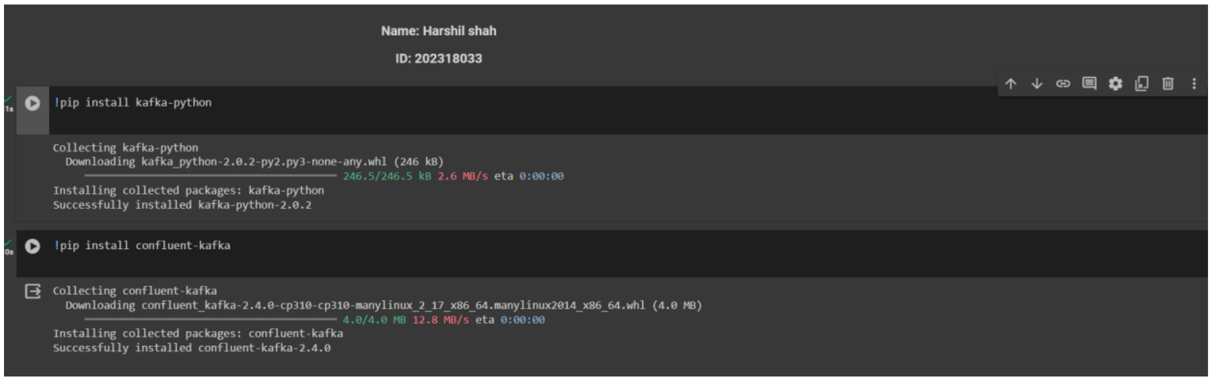
Functionality:

1. Order Generation: Producers generate order data, including product details, quantities, and customer information, and publish them to designated Kafka topics.
2. Inventory Management: Consumers subscribe to relevant Kafka topics, retrieve order data, and update inventory records accordingly. This ensures accurate inventory tracking and prevents overselling.
3. Delivery Processing: Consumers responsible for delivery processing extract order information from Kafka topics, initiate delivery workflows, and provide real-time updates to customers regarding order status and shipment tracking.

Kafka Installation:

The setup involved installing the necessary libraries and configuring Kafka producers to send messages to Kafka topics.

- Installed `kafka-python` and `confluent-kafka` libraries using pip.
- Configured Kafka producers for inventory orders and delivery orders.
- Defined a function to acknowledge message delivery and flush messages to ensure they are sent.



Name: Harshil shah

ID: 202318033

```
!pip install kafka-python
```

```
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 246.5/246.5 kB 2.6 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
```

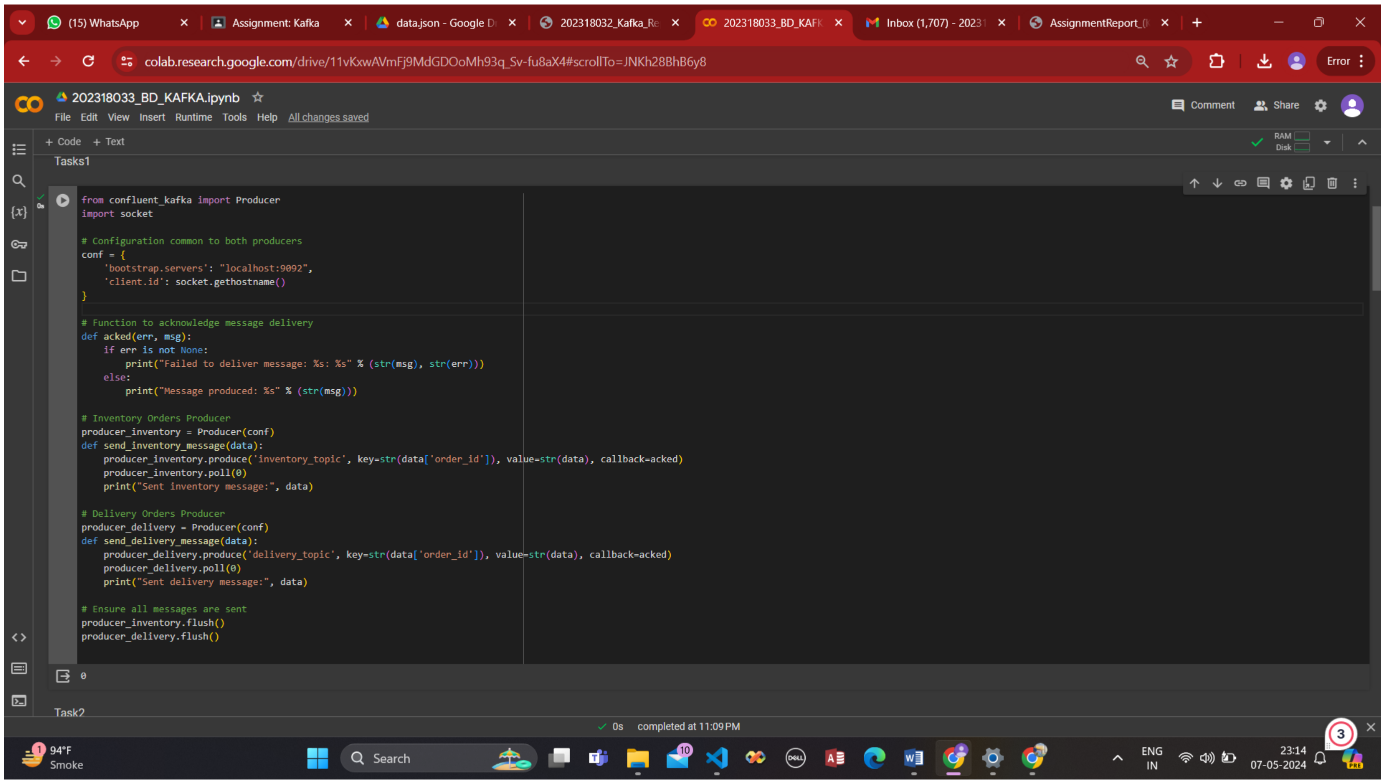
```
!pip install confluent-kafka
```

```
Collecting confluent-kafka
  Downloading confluent_kafka-2.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.0/4.0 MB 12.8 MB/s eta 0:00:00
Installing collected packages: confluent-kafka
Successfully installed confluent-kafka-2.4.0
```

Producer Implementation:

Two Kafka producers were implemented to send messages for inventory orders and delivery orders. Each producer sends messages with a specific type to the respective Kafka topics.

- Implemented a producer for inventory orders to send messages to the 'inventory_topic'.
- Implemented a producer for delivery orders to send messages to the 'delivery_topic'.



Tasks1

```
from confluent_kafka import Producer
import socket

# Configuration common to both producers
conf = {
    'bootstrap.servers': "localhost:9092",
    'client.id': socket.gethostname()
}

# Function to acknowledge message delivery
def acked(err, msg):
    if err is not None:
        print("Failed to deliver message: %s: %s" % (str(msg), str(err)))
    else:
        print("Message produced: %s" % (str(msg)))

# Inventory Orders Producer
producer_inventory = Producer(conf)
def send_inventory_message(data):
    producer_inventory.produce('inventory_topic', key=str(data['order_id']), value=str(data), callback=acked)
    producer_inventory.poll(0)
    print("Sent inventory message:", data)

# Delivery Orders Producer
producer_delivery = Producer(conf)
def send_delivery_message(data):
    producer_delivery.produce('delivery_topic', key=str(data['order_id']), value=str(data), callback=acked)
    producer_delivery.poll(0)
    print("Sent delivery message:", data)

# Ensure all messages are sent
producer_inventory.flush()
producer_delivery.flush()
```

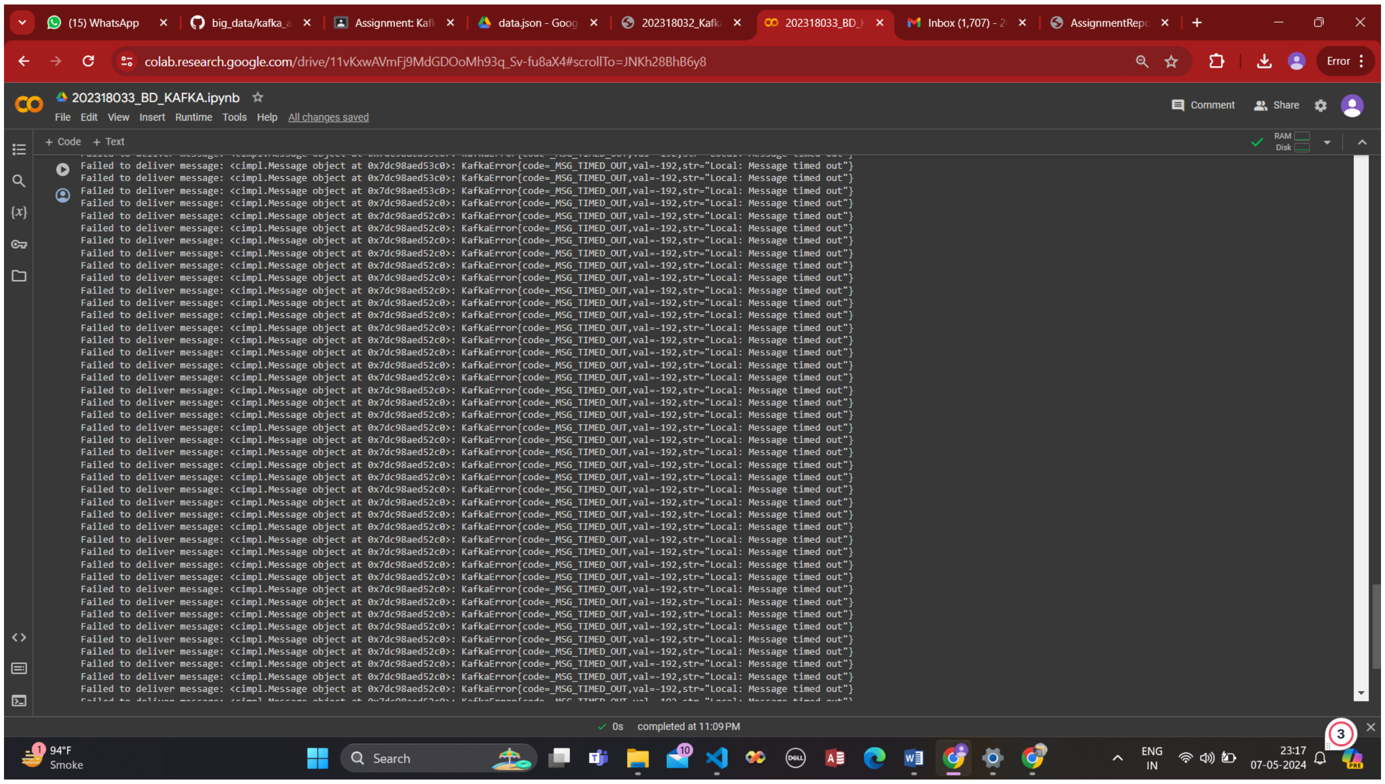
```
0
```

Task2

Message Sending:

Messages were sent to Kafka topics using the implemented producers. Each message contains order details such as order ID, product ID, quantity, and timestamp.

- Inventory messages were sent to the 'inventory_topic'.
- Delivery messages were sent to the 'delivery_topic'.

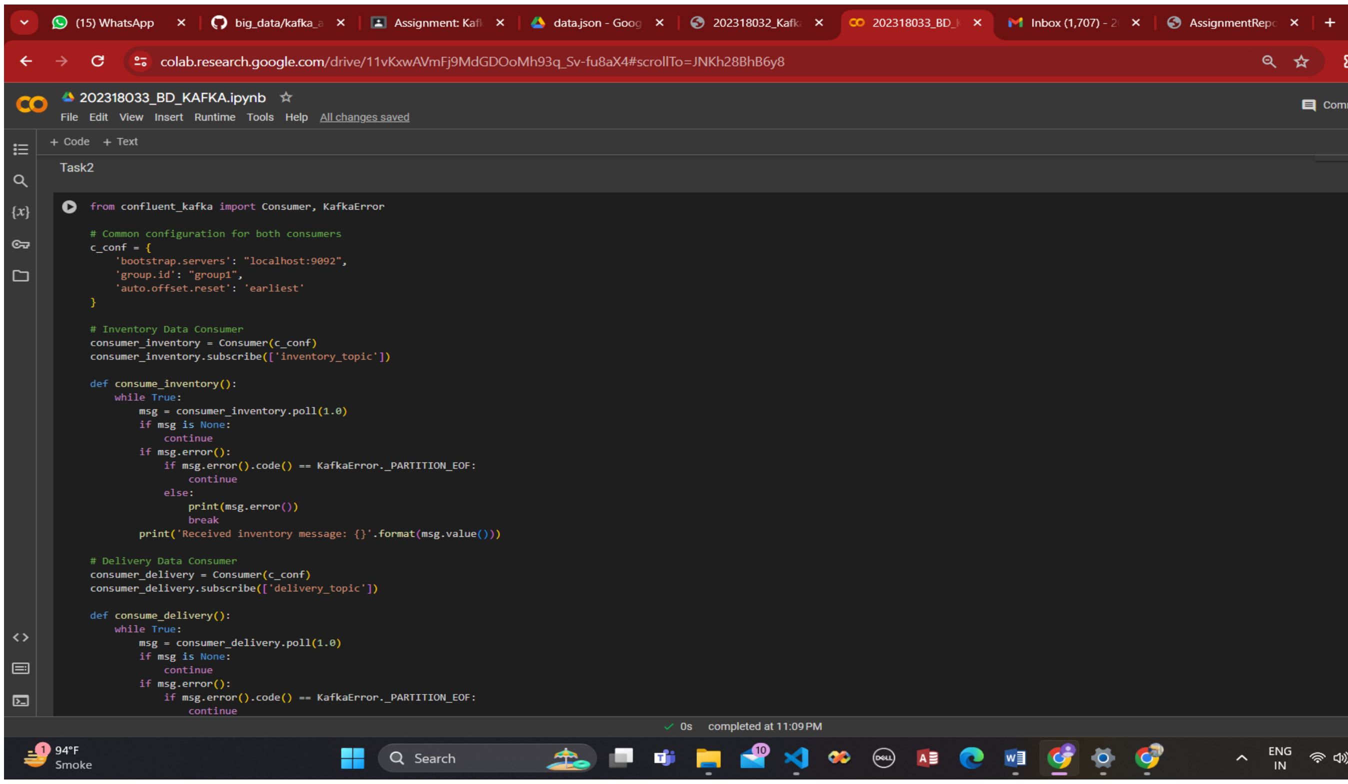


```
Failed to deliver message: <cimpl.Message object at 0x7dc98aed53c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed53c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed53c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
Failed to deliver message: <cimpl.Message object at 0x7dc98aed52c0>: KafkaError{code=_MSG_TIMED_OUT,val=-192,str="Local: Message timed out"}
```

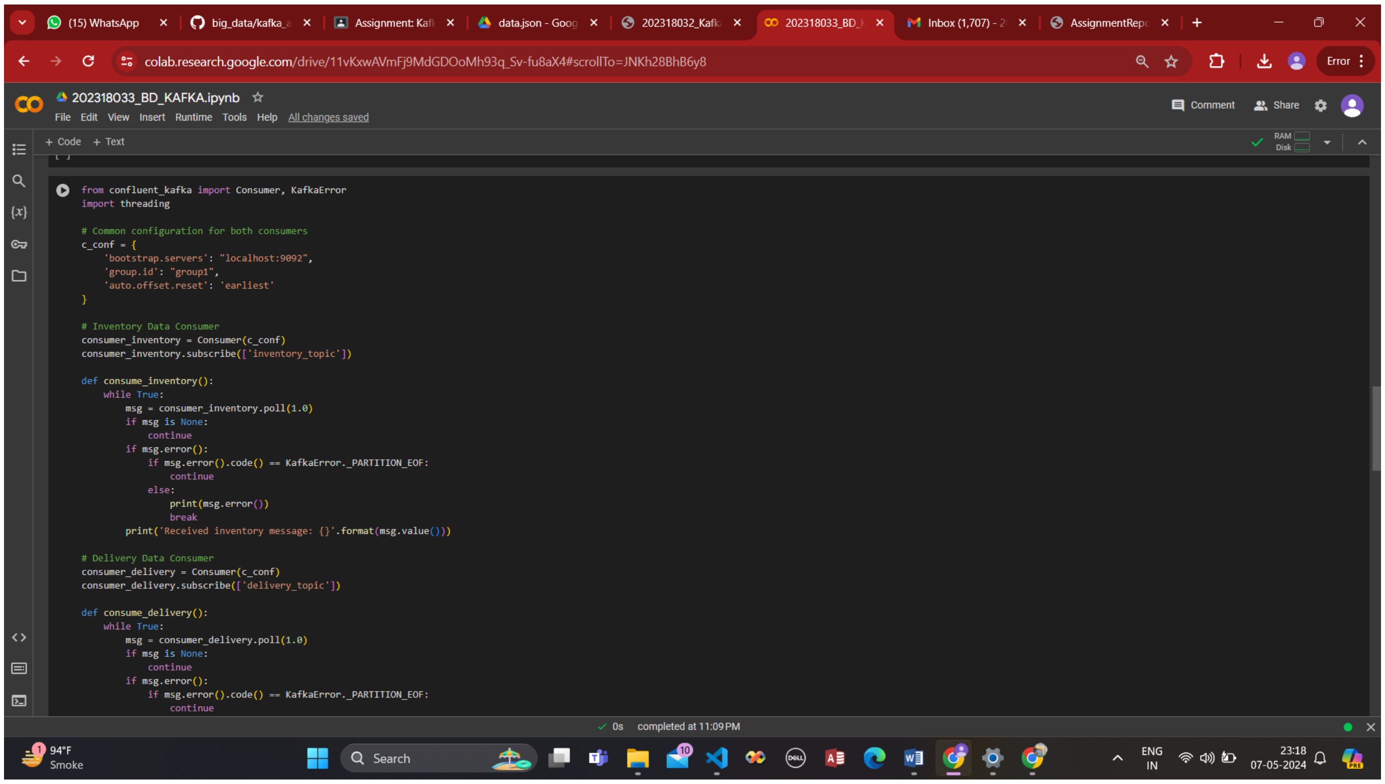
colab.research.google.com/drive/11vKxwAVmFj9MdGDOoMh93q_Sv-fu8aX4#scrollTo=JNKh28BhB6y8
202318033_BD_KAFKA.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s completed at 11:09 PM
23:17 07-05-2024

Consumer Implementation:

Consumers were implemented to consume messages from Kafka topics for inventory data and delivery data. Each consumer listens to its respective Kafka topic and processes incoming messages.

- Implemented consumers for inventory data and delivery data.
- Subscribed consumers to the 'inventory_topic' and 'delivery_topic', respectively.
- Defined functions to consume and process incoming messages.
- 

```
from confluent_kafka import Consumer, KafkaError

# Common configuration for both consumers
c_conf = {
    'bootstrap.servers': "localhost:9092",
    'group.id': "group1",
    'auto.offset.reset': 'earliest'
}

# Inventory Data Consumer
consumer_inventory = Consumer(c_conf)
consumer_inventory.subscribe(['inventory_topic'])

def consume_inventory():
    while True:
        msg = consumer_inventory.poll(1.0)
        if msg is None:
            continue
        if msg.error():
            if msg.error().code() == KafkaError._PARTITION_EOF:
                continue
            else:
                print(msg.error())
                break
        print('Received inventory message: {}'.format(msg.value()))

# Delivery Data Consumer
consumer_delivery = Consumer(c_conf)
consumer_delivery.subscribe(['delivery_topic'])

def consume_delivery():
    while True:
        msg = consumer_delivery.poll(1.0)
        if msg is None:
            continue
        if msg.error():
            if msg.error().code() == KafkaError._PARTITION_EOF:
                continue
```

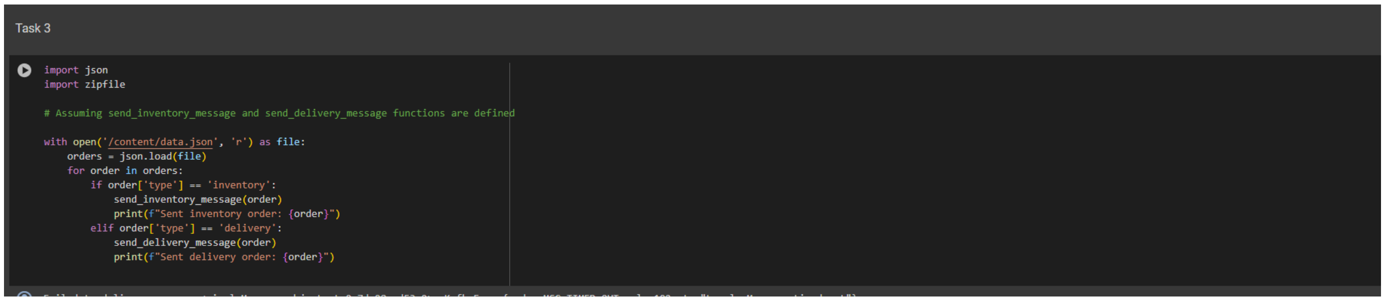


```
from confluent_kafka import Consumer, KafkaError
import threading

# Common configuration for both consumers
c_conf = {
    'bootstrap.servers': "localhost:9092",
    'group.id': "group1",
    'auto.offset.reset': 'earliest'
}

# Inventory Data Consumer
consumer_inventory = Consumer(c_conf)
consumer_inventory.subscribe(['inventory_topic'])

def consume_inventory():
    while True:
        msg = consumer_inventory.poll(1.0)
        if msg is None:
            continue
        if msg.error():
            if msg.error().code() == KafkaError._PARTITION_EOF:
                continue
            else:
                print(msg.error())
                break
        print('Received inventory message: {}'.format(msg.value()))

# Delivery Data Consumer
consumer_delivery = Consumer(c_conf)
consumer_delivery.subscribe(['delivery_topic'])

def consume_delivery():
    while True:
        msg = consumer_delivery.poll(1.0)
        if msg is None:
            continue
        if msg.error():
            if msg.error().code() == KafkaError._PARTITION_EOF:
                continue
```

Message Processing:

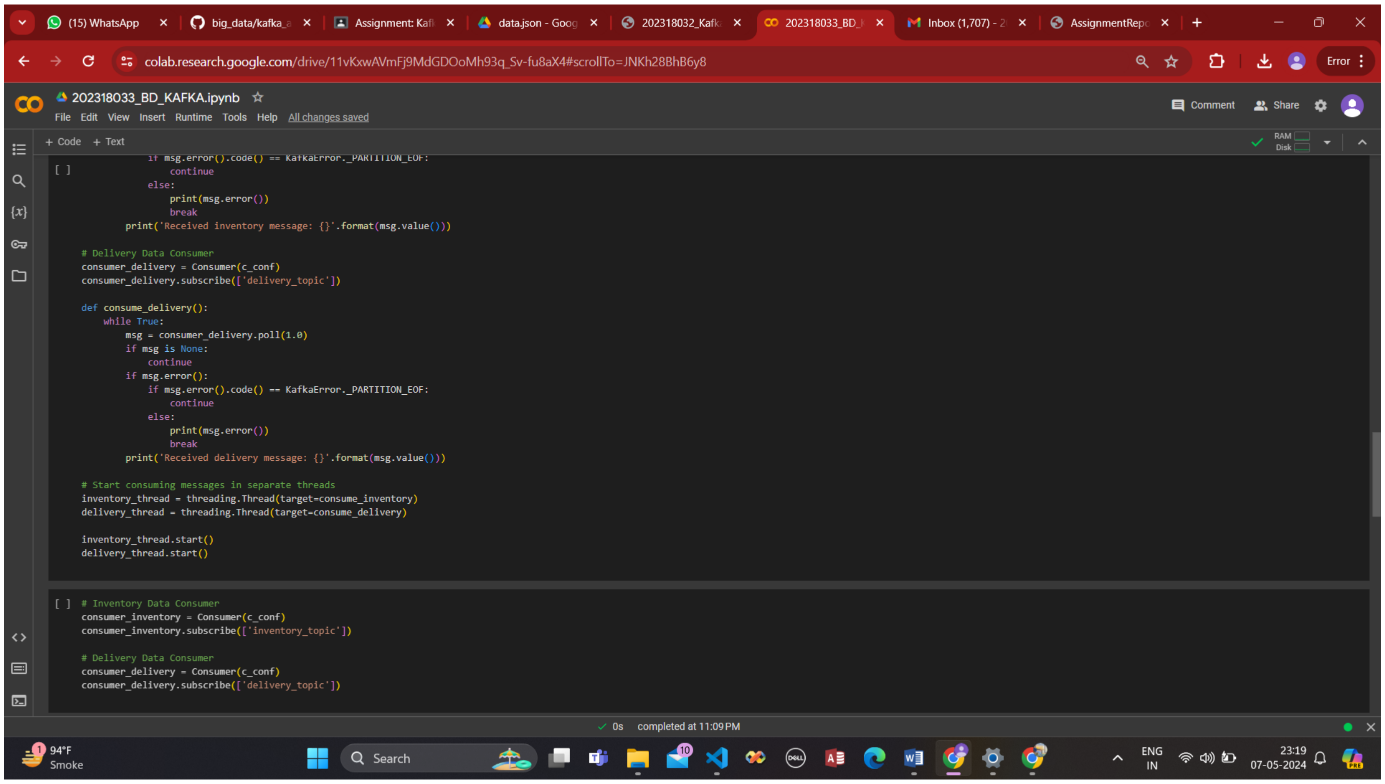
Incoming messages were processed by the consumers to update inventory databases, schedule deliveries, update delivery status, and notify customers as per the message type.

- Inventory data consumers processed messages from the 'inventory_topic'.
- Delivery data consumers processed messages from the 'delivery_topic'.



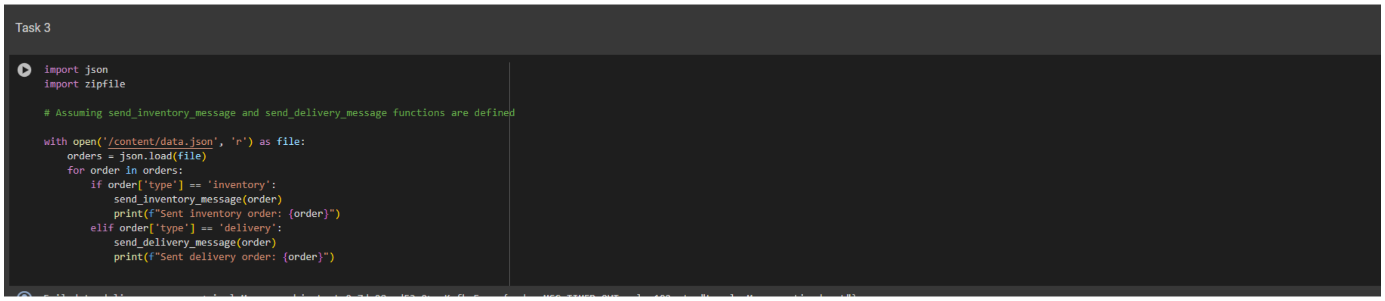
Task 3

```
import json
import zipfile

# Assuming send_inventory_message and send_delivery_message functions are defined

with open('/content/data.json', 'r') as file:
    orders = json.load(file)
    for order in orders:
        if order['type'] == 'inventory':
            send_inventory_message(order)
            print(f"Sent inventory order: {order}")
        elif order['type'] == 'delivery':
            send_delivery_message(order)
            print(f"Sent delivery order: {order}")
```



```python
                if msg.error().code() == KafkaError._PARTITION_EOF:
                    continue
                else:
                    print(msg.error())
                    break
            print('Received inventory message: {}'.format(msg.value()))

    # Delivery Data Consumer
    consumer_delivery = Consumer(c_conf)
    consumer_delivery.subscribe(['delivery_topic'])

    def consume_delivery():
        while True:
            msg = consumer_delivery.poll(1.0)
            if msg is None:
                continue
            if msg.error():
                if msg.error().code() == KafkaError._PARTITION_EOF:
                    continue
                else:
                    print(msg.error())
                    break
            print('Received delivery message: {}'.format(msg.value()))

    # Start consuming messages in separate threads
    inventory_thread = threading.Thread(target=consume_inventory)
    delivery_thread = threading.Thread(target=consume_delivery)

    inventory_thread.start()
    delivery_thread.start()
```

```python
# Inventory Data Consumer
consumer_inventory = Consumer(c_conf)
consumer_inventory.subscribe(['inventory_topic'])

# Delivery Data Consumer
consumer_delivery = Consumer(c_conf)
consumer_delivery.subscribe(['delivery_topic'])
```



Task 3

```python
import json
import zipfile

# Assuming send_inventory_message and send_delivery_message functions are defined

with open('/content/data.json', 'r') as file:
    orders = json.load(file)
    for order in orders:
        if order['type'] == 'inventory':
            send_inventory_message(order)
            print(f"Sent inventory order: {order}")
        elif order['type'] == 'delivery':
            send_delivery_message(order)
            print(f"Sent delivery order: {order}")
```