

Name: Mahmood Topiwala

Student ID: 202318030

Real-Time E-commerce Order Processing System Using Kafka

Introduction:

This report outlines the development of a Kafka-based system for managing e-commerce orders in real-time. The system handles inventory management and delivery processing by implementing Kafka producers and consumers.

Kafka Installation:

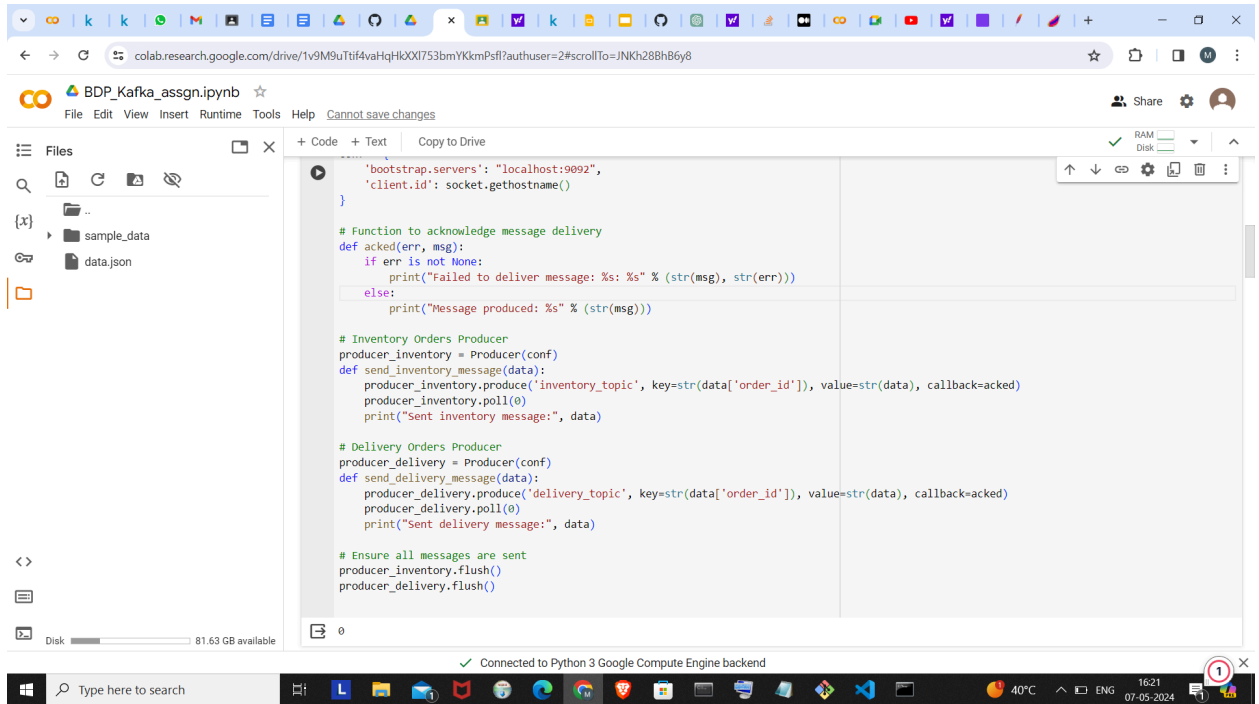
The setup involved installing the necessary libraries and configuring Kafka producers to send messages to Kafka topics.

- 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.

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'.



```
'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()
```

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'.

colab.research.google.com/drive/1v9M9uTt4vHqH0X0753bmYKkmPsfI?authuser=2#scrollTo=aAgh6V85CyZz

BDP_Kafka_assgn.ipynb

File Edit View Insert Runtime Tools Help Cannot save changes

Files

- sample_data
- data.json

```
Sent delivery order: {'order_id': '96', 'product_id': '9065', 'quantity': 85, 'type': 'delivery', 'timestamp': '2/2/2023'}
Sent delivery order: {'order_id': '41', 'product_id': '45247', 'quantity': 7, 'type': 'delivery', 'timestamp': '3/6/2024'}
Sent delivery order: {'order_id': '605', 'product_id': '61607', 'quantity': 72, 'type': 'delivery', 'timestamp': '3/4/2024'}
Sent delivery order: {'order_id': '2', 'product_id': '45', 'quantity': 28, 'type': 'delivery', 'timestamp': '6/13/2023'}
Sent delivery order: {'order_id': '4973', 'product_id': '8', 'quantity': 51, 'type': 'delivery', 'timestamp': '2/2/2024'}
Sent inventory order: {'order_id': '37', 'product_id': '87079', 'quantity': 57, 'type': 'inventory', 'timestamp': '11/5/2023'}
Sent inventory order: {'order_id': '565', 'product_id': '2061', 'quantity': 62, 'type': 'inventory', 'timestamp': '3/10/2024'}
Sent inventory order: {'order_id': '939', 'product_id': '34005', 'quantity': 3, 'type': 'inventory', 'timestamp': '12/21/2023'}
Sent inventory order: {'order_id': '81092', 'product_id': '6825', 'quantity': 34, 'type': 'inventory', 'timestamp': '3/5/2024'}
Sent delivery order: {'order_id': '84', 'product_id': '1354', 'quantity': 37, 'type': 'delivery', 'timestamp': '11/10/2023'}
Sent delivery order: {'order_id': '16', 'product_id': '08', 'quantity': 26, 'type': 'delivery', 'timestamp': '5/1/2023'}
Sent inventory order: {'order_id': '29', 'product_id': '476', 'quantity': 51, 'type': 'inventory', 'timestamp': '2/13/2024'}
Sent delivery order: {'order_id': '651', 'product_id': '25', 'quantity': 45, 'type': 'delivery', 'timestamp': '10/11/2023'}
Sent delivery order: {'order_id': '89649', 'product_id': '5618', 'quantity': 63, 'type': 'delivery', 'timestamp': '8/31/2023'}
Sent inventory order: {'order_id': '126', 'product_id': '41', 'quantity': 30, 'type': 'inventory', 'timestamp': '6/2/2023'}
Sent inventory order: {'order_id': '09905', 'product_id': '7', 'quantity': 26, 'type': 'inventory', 'timestamp': '5/29/2023'}
Sent inventory order: {'order_id': '84', 'product_id': '8931', 'quantity': 76, 'type': 'inventory', 'timestamp': '4/23/2024'}
Sent delivery order: {'order_id': '2272', 'product_id': '38', 'quantity': 94, 'type': 'delivery', 'timestamp': '7/19/2023'}
Sent delivery order: {'order_id': '157', 'product_id': '564', 'quantity': 55, 'type': 'delivery', 'timestamp': '1/30/2024'}
Sent inventory order: {'order_id': '407', 'product_id': '4', 'quantity': 41, 'type': 'inventory', 'timestamp': '2/22/2024'}
Sent delivery order: {'order_id': '73345', 'product_id': '7', 'quantity': 80, 'type': 'delivery', 'timestamp': '1/16/2023'}
Sent inventory order: {'order_id': '88835', 'product_id': '81', 'quantity': 2, 'type': 'inventory', 'timestamp': '8/19/2023'}
Sent delivery order: {'order_id': '34', 'product_id': '44827', 'quantity': 51, 'type': 'delivery', 'timestamp': '4/18/2024'}
Sent delivery order: {'order_id': '766', 'product_id': '69', 'quantity': 86, 'type': 'delivery', 'timestamp': '6/16/2023'}
Sent inventory order: {'order_id': '296', 'product_id': '8174', 'quantity': 82, 'type': 'inventory', 'timestamp': '3/2/2024'}
Sent delivery order: {'order_id': '2216', 'product_id': '91', 'quantity': 75, 'type': 'delivery', 'timestamp': '1/19/2024'}
Sent inventory order: {'order_id': '7526', 'product_id': '13', 'quantity': 87, 'type': 'inventory', 'timestamp': '7/8/2023'}
Sent delivery order: {'order_id': '29', 'product_id': '2', 'quantity': 88, 'type': 'delivery', 'timestamp': '3/12/2024'}
Sent delivery order: {'order_id': '7', 'product_id': '13', 'quantity': 7, 'type': 'delivery', 'timestamp': '5/18/2023'}
Sent delivery order: {'order_id': '401', 'product_id': '7', 'quantity': 19, 'type': 'delivery', 'timestamp': '5/1/2023'}
Sent inventory order: {'order_id': '5', 'product_id': '0', 'quantity': 62, 'type': 'inventory', 'timestamp': '3/5/2024'}
Sent inventory order: {'order_id': '7141', 'product_id': '504', 'quantity': 50, 'type': 'inventory', 'timestamp': '5/5/2023'}
Sent inventory order: {'order_id': '06308', 'product_id': '92', 'quantity': 58, 'type': 'inventory', 'timestamp': '11/11/2023'}
Sent delivery order: {'order_id': '864', 'product_id': '6', 'quantity': 17, 'type': 'delivery', 'timestamp': '11/16/2023'}
Sent delivery order: {'order_id': '790', 'product_id': '877', 'quantity': 69, 'type': 'delivery', 'timestamp': '1/13/2024'}
Sent inventory order: {'order_id': '790', 'product_id': '877', 'quantity': 69, 'type': 'inventory', 'timestamp': '1/13/2024'}
```

1s completed at 2:30 PM

colab.research.google.com/drive/1v9M9uTt4vHqH0X0753bmYKkmPsfI?authuser=2#scrollTo=aAgh6V85CyZz

BDP_Kafka_assgn.ipynb

File Edit View Insert Runtime Tools Help Cannot save changes

Files

- sample_data
- data.json

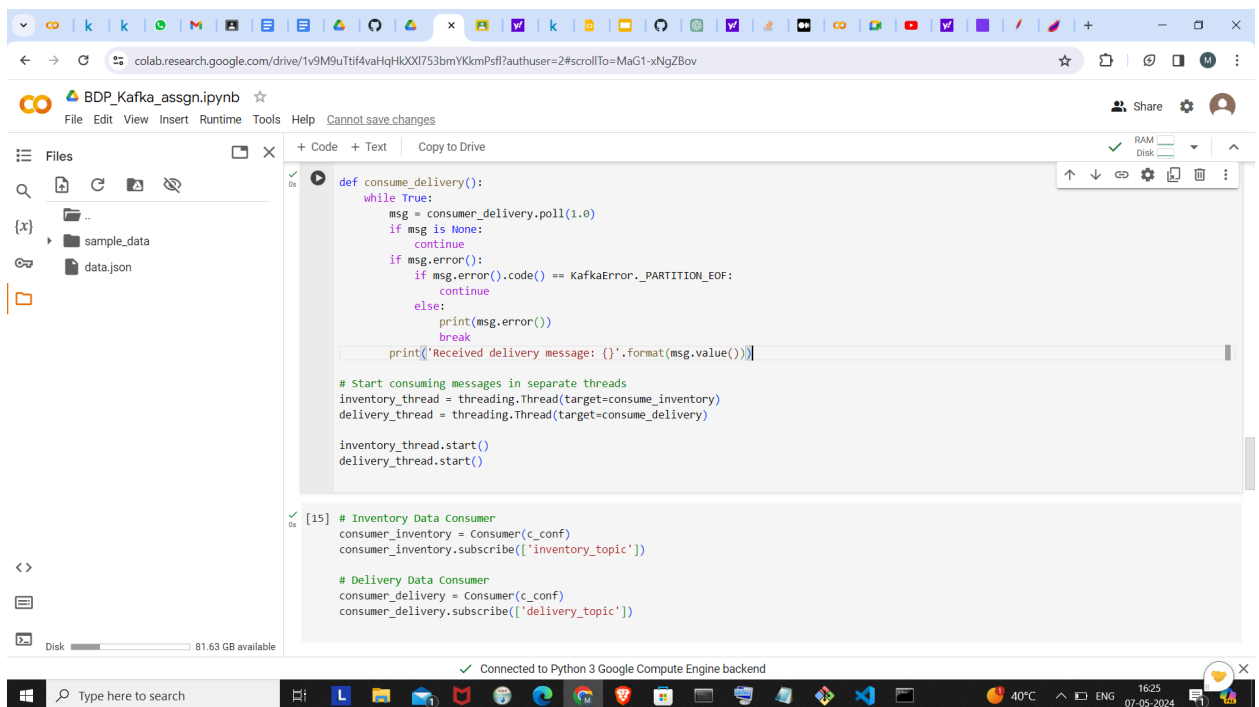
```
Sent delivery order: {'order_id': '804', 'product_id': '2312', 'quantity': 36, 'type': 'delivery', 'timestamp': '5/1/2023'}
Sent inventory order: {'order_id': '6', 'product_id': '2312', 'quantity': 36, 'type': 'inventory', 'timestamp': '5/1/2023'}
Sent inventory order: {'order_id': '6762', 'product_id': '527', 'quantity': 69, 'type': 'inventory', 'timestamp': '7/22/2023'}
Sent delivery order: {'order_id': '031', 'product_id': '63393', 'quantity': 40, 'type': 'delivery', 'timestamp': '8/19/2023'}
Sent inventory order: {'order_id': '537', 'product_id': '7904', 'quantity': 99, 'type': 'inventory', 'timestamp': '10/27/2023'}
Sent inventory order: {'order_id': '72639', 'product_id': '12904', 'quantity': 80, 'type': 'inventory', 'timestamp': '4/10/2024'}
Sent inventory order: {'order_id': '47', 'product_id': '67835', 'quantity': 73, 'type': 'inventory', 'timestamp': '1/17/2024'}
Sent delivery order: {'order_id': '53', 'product_id': '6', 'quantity': 54, 'type': 'delivery', 'timestamp': '3/30/2024'}
Sent inventory order: {'order_id': '0847', 'product_id': '0', 'quantity': 71, 'type': 'inventory', 'timestamp': '3/19/2024'}
Sent delivery order: {'order_id': '3', 'product_id': '2', 'quantity': 64, 'type': 'delivery', 'timestamp': '6/15/2023'}
Sent inventory order: {'order_id': '95', 'product_id': '2', 'quantity': 43, 'type': 'inventory', 'timestamp': '11/14/2023'}
Sent delivery order: {'order_id': '6926', 'product_id': '24863', 'quantity': 25, 'type': 'delivery', 'timestamp': '7/2/2023'}
Sent inventory order: {'order_id': '09337', 'product_id': '4283', 'quantity': 39, 'type': 'inventory', 'timestamp': '10/6/2023'}
Sent inventory order: {'order_id': '2', 'product_id': '2952', 'quantity': 33, 'type': 'inventory', 'timestamp': '10/28/2023'}
Sent delivery order: {'order_id': '4433', 'product_id': '133', 'quantity': 97, 'type': 'delivery', 'timestamp': '2/16/2024'}
Sent delivery order: {'order_id': '82', 'product_id': '702', 'quantity': 40, 'type': 'delivery', 'timestamp': '12/5/2023'}
Sent inventory order: {'order_id': '2', 'product_id': '016', 'quantity': 88, 'type': 'inventory', 'timestamp': '6/24/2023'}
Sent delivery order: {'order_id': '747', 'product_id': '1', 'quantity': 77, 'type': 'delivery', 'timestamp': '5/6/2023'}
Sent delivery order: {'order_id': '8886', 'product_id': '308', 'quantity': 11, 'type': 'delivery', 'timestamp': '1/28/2024'}
Sent inventory order: {'order_id': '9', 'product_id': '7', 'quantity': 66, 'type': 'inventory', 'timestamp': '7/28/2023'}
Sent inventory order: {'order_id': '59793', 'product_id': '643', 'quantity': 39, 'type': 'inventory', 'timestamp': '12/11/2023'}
Sent inventory order: {'order_id': '40', 'product_id': '620', 'quantity': 63, 'type': 'inventory', 'timestamp': '10/27/2023'}
Sent delivery order: {'order_id': '3', 'product_id': '4670', 'quantity': 98, 'type': 'delivery', 'timestamp': '5/7/2023'}
Sent delivery order: {'order_id': '9432', 'product_id': '91418', 'quantity': 37, 'type': 'delivery', 'timestamp': '3/6/2024'}
Sent delivery order: {'order_id': '87', 'product_id': '719', 'quantity': 69, 'type': 'delivery', 'timestamp': '2/6/2024'}
Sent delivery order: {'order_id': '54375', 'product_id': '450', 'quantity': 24, 'type': 'delivery', 'timestamp': '7/2/2023'}
Sent inventory order: {'order_id': '7225', 'product_id': '4', 'quantity': 86, 'type': 'inventory', 'timestamp': '12/11/2023'}
Sent inventory order: {'order_id': '72', 'product_id': '4656', 'quantity': 97, 'type': 'inventory', 'timestamp': '4/26/2023'}
Sent inventory order: {'order_id': '36', 'product_id': '6268', 'quantity': 98, 'type': 'inventory', 'timestamp': '1/15/2024'}
Sent delivery order: {'order_id': '24', 'product_id': '36629', 'quantity': 46, 'type': 'delivery', 'timestamp': '5/31/2023'}
Sent delivery order: {'order_id': '95737', 'product_id': '9', 'quantity': 95, 'type': 'delivery', 'timestamp': '7/17/2023'}
Sent delivery order: {'order_id': '18', 'product_id': '6783', 'quantity': 78, 'type': 'delivery', 'timestamp': '2/10/2024'}
Sent delivery order: {'order_id': '43464', 'product_id': '652', 'quantity': 62, 'type': 'delivery', 'timestamp': '1/16/2024'}
Sent delivery order: {'order_id': '29', 'product_id': '70', 'quantity': 1, 'type': 'delivery', 'timestamp': '11/11/2023'}
Sent delivery order: {'order_id': '8', 'product_id': '41', 'quantity': 97, 'type': 'delivery', 'timestamp': '1/22/2024'}
Sent inventory order: {'order_id': '401', 'product_id': '7', 'quantity': 19, 'type': 'inventory', 'timestamp': '5/1/2023'}
```

1s completed at 2:30 PM

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.



```
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()

[15] # 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'])
```

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'.

The screenshot shows a Google Colab notebook interface. The browser address bar displays a Google Drive link. The notebook title is "BDP_Kafka_assgn.ipynb". The left sidebar shows a file explorer with a folder named "sample_data" containing a file "data.json". The main area contains Python code for interacting with Kafka. The code includes two main functions: "consume_inventory" and "consume_delivery", both using a "while True" loop to poll messages from Kafka. It handles "KafkaError_PARTITION_EOF" and prints received messages. At the bottom, it starts two separate threads: "inventory_thread" and "delivery_thread". The status bar at the bottom indicates "Connected to Python 3 Google Compute Engine backend".

```
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
            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)
```

Conclusion:

The real-time e-commerce order processing system using Kafka was successfully implemented. Kafka producers and consumers were configured to handle inventory and delivery orders, ensuring efficient management of e-commerce operations.