

Delivery Processing System Report

To stay ahead in today's cutthroat market, businesses need lightning-fast delivery processing. Apache Kafka, a powerful distributed streaming platform, offers the key to unlocking this efficiency. This report details how we implemented Kafka producers and consumers, coupled with intelligent message filtering, to create a seamless system for managing inventory and delivery orders.

1.Kafka Producers Implementation:

Producer 1: Inventory Orders Producer:

Leveraging Kafka client libraries, the application establishes a connection to the Kafka broker and selectively publishes messages to the "inventory_orders" topic. A filtering mechanism ensures only messages categorized as "inventory" are transmitted, streamlining data flow and optimizing message delivery.

```
1  from confluent_kafka import Producer
2  import json
3
4  file_path = '/FileStore/shared_uploads/202318032@daiict.ac.in/data-2.json'
5  df = spark.read.json(file_path)
6
7  # Define Kafka producer configuration
8  bootstrap_servers = 'localhost:9092'
9  producer_config = {
10     'bootstrap.servers': bootstrap_servers
11 }
12 inventory_producer = Producer(producer_config)
13
14 # Function to produce a message for inventory orders
15 def produce_inventory_order(order):
16     message = json.dumps(order)
17     inventory_producer.produce('inventory_orders', message.encode('utf-8'))
18
19 # Assuming `df` is the DataFrame containing orders
20 for row in df.collect():
21     order = row.asDict()
22     if order['type'] == 'inventory':
23         produce_inventory_order(order)
24
25 # Flush producer to ensure all messages are sent
26 inventory_producer.flush()
```

Producer 2: Delivery Orders Producer

Created a Kafka producer responsible for handling messages related to the delivery of orders.

Incorporated filtering functionality to send only messages where the type is "delivery".

Integrated error handling mechanisms to ensure reliable message delivery.

```
1  # Create Kafka producer for delivery orders
2  delivery_producer = Producer(producer_config)
3
4  # Function to produce a message for delivery orders
5  def produce_delivery_order(order):
6      message = json.dumps(order)
7      delivery_producer.produce('delivery_orders', message.encode('utf-8'))
8
9  for row in df.collect():
10     order = row.asDict()
11     if order['type'] == 'delivery':
12         produce_delivery_order(order)
13
14  # Flush producer to ensure all messages are sent
15  delivery_producer.flush()
```

2. Kafka Consumers Implementation:

Consumer 1: Inventory Data Consumer:

Developed a Kafka consumer application to handle inventory data.

Configured the consumer to listen for messages with the type "inventory".

Implemented logic to process messages and update inventory databases or systems accordingly.

```

1  from confluent_kafka import Consumer, KafkaError
2  consumer_config = {
3      'bootstrap.servers': bootstrap_servers,
4      'group.id': 'inventory_consumer_group',
5      'auto.offset.reset': 'earliest'
6  }
7
8  # Create Kafka consumer for inventory data
9  inventory_consumer = Consumer(consumer_config)
10 inventory_consumer.subscribe(['inventory_orders'])
11
12 # Function to handle inventory data
13 def handle_inventory_data(order):
14     # Process inventory data here
15     print("Processing inventory data:", order)
16
17 # Poll for new messages in the Kafka topic and process them
18 try:
19     while True:
20         msg = inventory_consumer.poll(timeout=1.0)
21         if msg is None:
22             continue
23         if msg.error():
24             if msg.error().code() == KafkaError._PARTITION_EOF:
25                 continue
26             else:
27                 print(msg.error())
28                 break
29         order = json.loads(msg.value().decode('utf-8'))
30         if order['type'] == 'inventory':
31             handle_inventory_data(order)
32
33 except KeyboardInterrupt:
34     pass
35
36 finally:
37     # Close Kafka consumer
38     inventory_consumer.close()

```

Consumer 2: Delivery Data Consumer:

Implemented a Kafka consumer to manage delivery tasks.

Configured the consumer to listen for messages with the type "delivery".

Developed functionalities to schedule deliveries, update delivery status, and notify customers based on received messages.

```
1 delivery_consumer = Consumer(consumer_config)
2 delivery_consumer.subscribe(['delivery_orders'])
3
4 def handle_delivery_data(order):
5     print("Processing delivery data:", order)
6
7 try:
8     while True:
9         msg = delivery_consumer.poll(timeout=1.0)
10        if msg is None:
11            continue
12        if msg.error():
13            if msg.error().code() == KafkaError._PARTITION_EOF:
14                continue
15            else:
16                print(msg.error())
17                break
18        order = json.loads(msg.value().decode('utf-8'))
19        if order['type'] == 'delivery':
20            handle_delivery_data(order)
21
22 except KeyboardInterrupt:
23     pass
24
25 finally:
26     delivery_consumer.close()
```
