Report on Real-Time E-commerce Order Processing System Utilizing Kafka

Authored by: Kandarp Malkan (202318021)

To create a Kafka-driven system for handling e-commerce orders in real-time, the setup of producers, consumers, and implementation of message filtering logic is imperative. Below delineates the steps to accomplish this:

Step 1: Kafka Setup

1. Install Kafka: Ensure Kafka installation and its running status either on the local system or a designated server.

2. Topic Creation: Establish Kafka topics named "inventory_orders" and "delivery_orders" for the respective producers to dispatch messages to.

Step 2: Implementation of Kafka Producers

1. Inventory Orders Producer (inventory_orders_producer):

- This producer selectively transmits messages identified with the "inventory" type.

- Develop a Kafka producer tasked with extracting inventory-related events from a data repository (such as a database or event stream) and dispatching messages with the type specified as "inventory" to the "inventory_orders" topic.

2. Delivery Orders Producer (delivery_orders_producer):

- This producer exclusively sends messages classified under the "delivery" type.

- Construct a Kafka producer designed to retrieve delivery-related events and dispatch messages marked with the type "delivery" to the "delivery_orders" topic.

Step 3: Implementation of Kafka Consumers

1. Inventory Data Consumer (inventory_data_consumer):

- Configure a Kafka consumer subscribed to the "inventory_orders" topic.

- Implement procedures to process incoming inventory messages by updating corresponding inventory databases or systems.

2. Delivery Data Consumer (delivery_data_consumer):

- Set up a Kafka consumer targeting the "delivery_orders" topic.

- Formulate procedures to handle delivery-related messages, including scheduling deliveries, updating delivery statuses, and informing customers.

Step 4: Development of Message Filtering Logic

1. Producer Message Filtering:

- Embed logic within each producer (inventory_orders_producer and delivery_orders_producer) to sift through messages based on the type field extracted from the incoming data source.

- Only dispatch messages to Kafka if they align with the specified type criteria (i.e., inventory or delivery).

Additional Considerations

- Error Handling: Implement robust error handling mechanisms within producers and consumers to gracefully manage exceptions or failed operations.

- Scalability: Design the system with scalability in mind, incorporating Kafka partitioning, consumer groups, and scaling strategies to accommodate escalating loads.

- Monitoring and Logging: Employ Kafka monitoring utilities and logging frameworks to effectively monitor system performance and troubleshoot issues as they arise.

By adhering to these steps and adhering to best practices, the development of a resilient Kafka-driven e-commerce order management system, proficient in real-time inventory management and delivery processing, is achievable.