In this lab, we set up a **Kafka streaming pipeline** using **Docker**, integrating **Kafka**, **Zookeeper**, and **Spark** to process streaming data.

Bài toán: viết và phân tích dữ liệu mua sắm của khách hàng

**Step 1: Set Up Docker Compose**

The first step is to configure **Docker Compose** to create a network of services that includes **Zookeeper**, **Kafka**, and **Spark**. These components are essential for a streaming application where Kafka serves as the message broker, Zookeeper manages Kafka, and Spark is used to process streaming data.

**Explanation of Services**

* **Zookeeper**: Kafka's dependency to manage its distributed nodes.
* **Kafka**: The message broker. It connects to Zookeeper and listens on port 9092.
* **Spark**: The Spark master container, where we'll run Spark jobs.
* **Spark Worker**: The Spark worker node, connected to the Spark master.
* **Kafka-Spark Network**: A custom bridge network to ensure that all containers can communicate.

**Step 2: Configure Kafka Producer**

The Kafka producer sends streaming events to Kafka topics. Here you can write a Python script to send data to Kafka using kafka-python.

**Step 3: Configure Kafka Consumer in Spark**

The **Kafka consumer** is set up within Spark to process the streaming data. Spark will read from the Kafka topic (ecommerce-events) and process it in real-time.

**Step 4: Create Dockerfile for Producer and Consumer**

To containerize both the producer and consumer, create the **Dockerfile** in the same directory as your producer.py and spark\_consumer.py.

Tạo image, ví dụ:

docker build -t kafka-spark-producer-consumer .

**Step 6: Run the Entire Environment with Docker Compose**

Now that we have the **Kafka**, **Zookeeper**, **Spark**, and our **producer** and **consumer** scripts ready, we can run everything together using Docker Compose.

ví dụ:

docker-compose up -d

kiểm tra network và container :

docker network ls

docker-compose ps

**Run the Producer** to send events to Kafka. Use the following command to run the producer in the Kafka Spark network:

Ví dụ:

docker run --network kafka\_spark\_net kafka-spark-producer-consumer python producer.py

Nếu chạy đúng sẽ ra như sau:

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '0', 'user\_id': 'user\_0'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '1', 'user\_id': 'user\_1'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '2', 'user\_id': 'user\_2'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '3', 'user\_id': 'user\_3'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '4', 'user\_id': 'user\_4'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '5', 'user\_id': 'user\_5'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '6', 'user\_id': 'user\_6'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '7', 'user\_id': 'user\_7'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '8', 'user\_id': 'user\_8'}

Sent event: {'event\_type': 'add\_to\_cart', 'product\_id': '9', 'user\_id': 'user\_9'}

**Run the Spark Streaming Consumer** to process the events from Kafka:

Ví dụ:

docker run --network kafka\_spark\_net kafka-spark-producer-consumer python spark\_consumer.py

Lưu ý: kiểm tra kết nối của spark và kafka

Một số tài liệu tham khảo:

Giới thiệu về kafka:

<https://kafka.apache.org/intro>

Chạy kafka streaming trên Databricks:

https://www.databricks.com/blog/2017/04/04/real-time-end-to-end-integration-with-apache-kafka-in-apache-sparks-structured-streaming.html