**Xây dựng piline kafka-spark bằng Docker**

**Tổng quan các bước thực hiện**

1. **Khởi chạy môi trường:** Dùng file docker-compose.yml để bật các service Kafka và Spark.
2. **Tạo Kafka Topic:** Tạo một "kênh" để producer gửi tin nhắn vào và consumer đọc tin nhắn ra.
3. **Viết mã Producer (Python):** Tạo một script Python để giả lập việc gửi dữ liệu giao dịch (ID, số tiền, địa điểm) liên tục vào topic Kafka.
4. **Viết mã Consumer (PySpark):** Tạo một script Spark Streaming để đọc dữ liệu từ topic Kafka, xử lý và tính tổng số tiền giao dịch theo từng batch.
5. **Thực thi Demo:** Chạy producer và consumer để thấy luồng dữ liệu hoạt động.

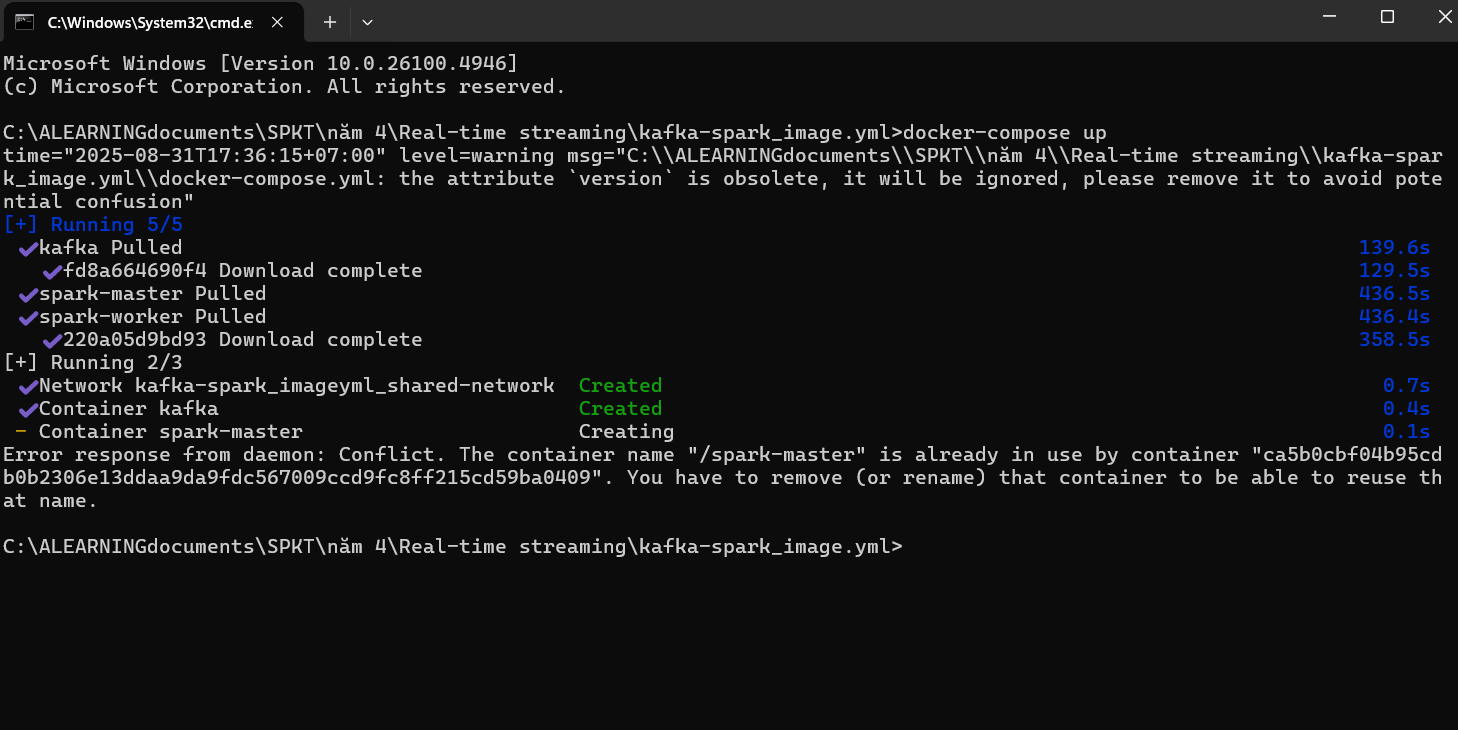
**Bước 1: Chuẩn bị và Khởi chạy Môi trường**

1. Lưu toàn bộ nội dung file cấu hình bạn đã cung cấp vào một file tên là docker-compose.yml.
2. Mở terminal hoặc PowerShell, di chuyển đến thư mục chứa file docker-compose.yml và chạy lệnh sau:

Bash

docker-compose up -d

Lệnh này sẽ tải các image cần thiết và khởi chạy 3 container: kafka, spark-master, và spark-worker ở chế độ nền.



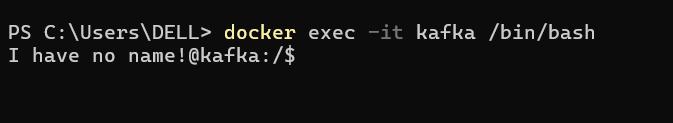
**Bước 2: Tạo Kafka Topic**

Producer cần một topic để gửi dữ liệu vào. Chúng ta sẽ tạo một topic tên là giao\_dich (nghĩa là "transaction").

1. Truy cập vào container kafka đang chạy:

Bash

docker exec -it kafka /bin/bash



1. Sau khi đã vào trong container, chạy lệnh sau để tạo topic. Ở đây, chúng ta sẽ tạo topic với **3 partitions** .

Bash

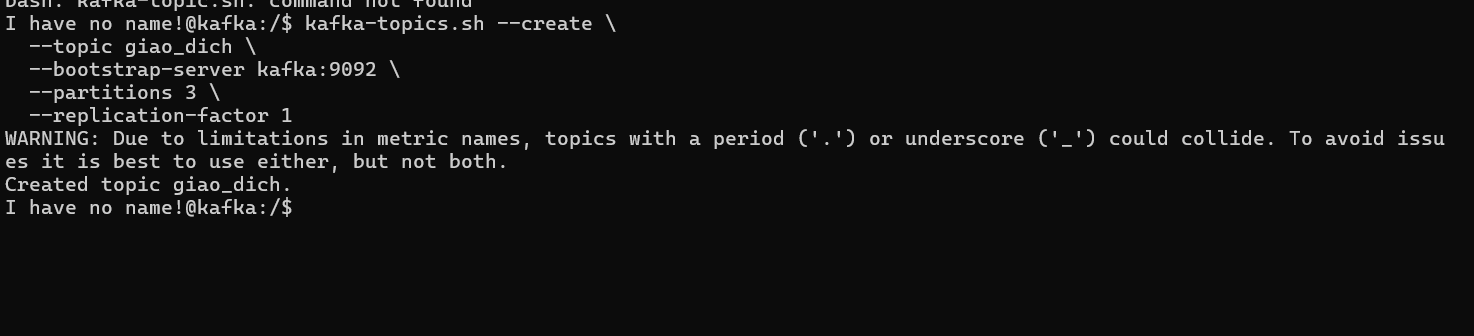
kafka-topics.sh --create \

--topic giao\_dich \

--bootstrap-server kafka:9092 \

--partitions 3 \

--replication-factor 1



1. Gõ exit để thoát khỏi container Kafka.

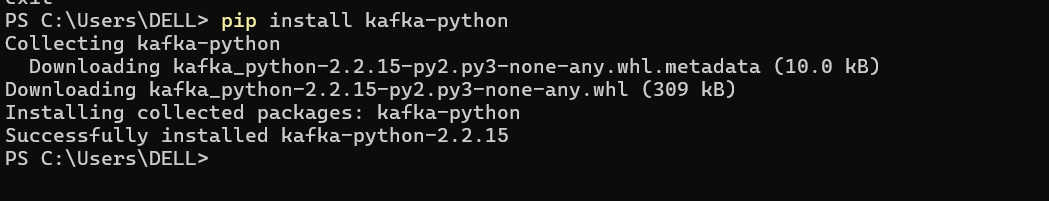
**Bước 3: Viết mã Producer bằng Python**

Script này sẽ liên tục sinh dữ liệu giao dịch giả và gửi vào topic giao\_dich.

1. Đảm bảo bạn đã cài thư viện kafka-python trên máy của mình (không phải trong Docker):

Bash

pip install kafka-python



1. Tạo một file tên producer.py và dán đoạn code sau vào:

Python

import json

import time

import random

from kafka import KafkaProducer

# Tạo một producer kết nối đến Kafka server

# Lưu ý: Vì script này chạy từ máy thật của bạn (bên ngoài Docker),

# chúng ta sẽ kết nối tới port 9093 đã được map ra localhost.

producer = KafkaProducer(

bootstrap\_servers=['localhost:9093'],

value\_serializer=lambda v: json.dumps(v).encode('utf-8')

)

topic\_name = 'giao\_dich'

print(f"Bắt đầu gửi dữ liệu tới topic '{topic\_name}'...")

# Vòng lặp vô hạn để gửi dữ liệu

transaction\_id = 1

while True:

try:

# Tạo dữ liệu giao dịch giả

data = {

'id': transaction\_id,

'so\_tien': random.randint(10000, 5000000), # Số tiền ngẫu nhiên từ 10k đến 5m

'dia\_diem': random.choice(['Ha Noi', 'Ho Chi Minh', 'Da Nang', 'Can Tho'])

}

# Gửi message tới Kafka

producer.send(topic\_name, value=data)

print(f"Đã gửi: {data}")

transaction\_id += 1

time.sleep(1) # Tạm dừng 1 giây

except KeyboardInterrupt:

print("\nĐã dừng producer.")

break

except Exception as e:

print(f"Có lỗi xảy ra: {e}")

break

# Đảm bảo tất cả message đã được gửi đi trước khi thoát

producer.flush()

producer.close()

**Bước 4: Viết mã Consumer bằng Spark Streaming (PySpark)**

Script này sẽ là ứng dụng Spark đọc dữ liệu từ Kafka và tính tổng số tiền.

1. Tạo một file tên spark\_consumer.py và dán đoạn code sau vào:

Python

import json

from pyspark.sql import SparkSession

from pyspark.sql.functions import from\_json, col, sum

from pyspark.sql.types import StructType, StructField, StringType, IntegerType, LongType

# Khởi tạo SparkSession, entry point mới cho Spark 2.x trở lên

spark = SparkSession \

    .builder \

    .appName("KafkaSparkStructuredStreamingDemo") \

    .getOrCreate()

spark.sparkContext.setLogLevel("WARN")

# Đọc dữ liệu từ Kafka dưới dạng một DataFrame streaming

df = spark \

    .readStream \

    .format("kafka") \

    .option("kafka.bootstrap.servers", "kafka:9092") \

    .option("subscribe", "giao\_dich") \

    .load()

# Định nghĩa cấu trúc (schema) của dữ liệu JSON trong message

schema = StructType([

    StructField("id", IntegerType()),

    StructField("so\_tien", LongType()),

    StructField("dia\_diem", StringType()),

])

# Xử lý DataFrame

# 1. Chuyển cột 'value' từ dạng binary sang string

# 2. Parse chuỗi JSON thành các cột riêng biệt dựa trên schema

# 3. Gom nhóm và tính tổng cột "so\_tien"

result\_df = df.selectExpr("CAST(value AS STRING)") \

    .select(from\_json(col("value"), schema).alias("data")) \

    .select("data.so\_tien") \

    .agg(sum("so\_tien").alias("total\_amount"))

# Xuất kết quả ra console

query = result\_df \

    .writeStream \

    .outputMode("complete") \

    .format("console") \

    .start()

query.awaitTermination()

**Bước 5: Hướng dẫn chạy Demo**

Bây giờ bạn sẽ chạy song song cả producer và consumer.

1. **Chạy Producer (trên máy của bạn):** Mở một terminal, di chuyển đến thư mục chứa file producer.py và chạy:

Bash

python producer.py

Bạn sẽ thấy các dòng log "Đã gửi: ..." xuất hiện liên tục. Cứ để terminal này chạy.

1. **Chạy Consumer (bên trong Spark container):**
   * Đầu tiên, bạn cần sao chép file spark\_consumer.py vào container spark-master:

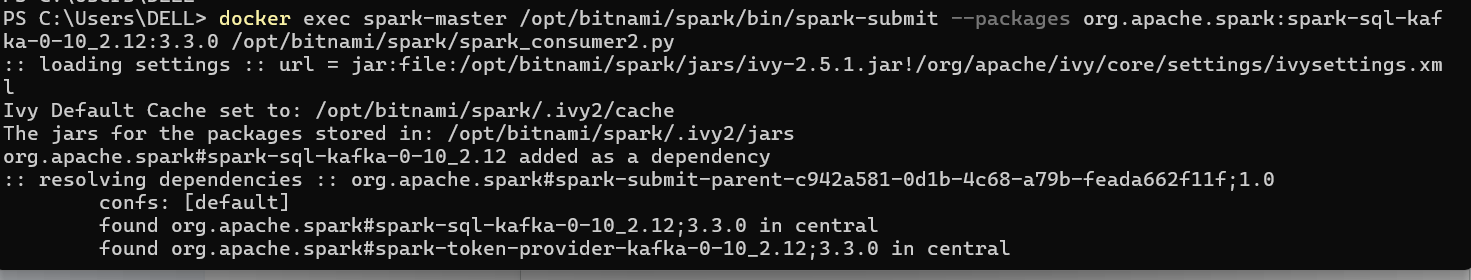
Bash

docker cp spark\_consumer.py spark-master:/opt/bitnami/spark/

* + Mở một terminal **mới** và thực thi spark-submit. Lệnh này sẽ chạy ứng dụng Spark của bạn và chỉ định các gói thư viện cần thiết để kết nối với Kafka.

Bash

docker exec spark-master /opt/bitnami/spark/bin/spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.3.0 /opt/bitnami/spark/spark\_consumer.py



* + Sau khi chạy, bạn sẽ thấy output từ Spark, cứ mỗi vài giây nó sẽ in ra tổng số tiền giao dịch nhận được trong khoảng thời gian đó.

