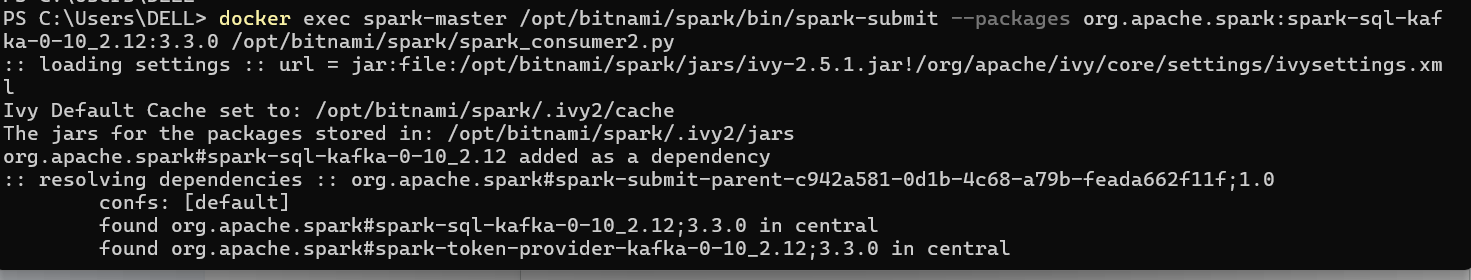
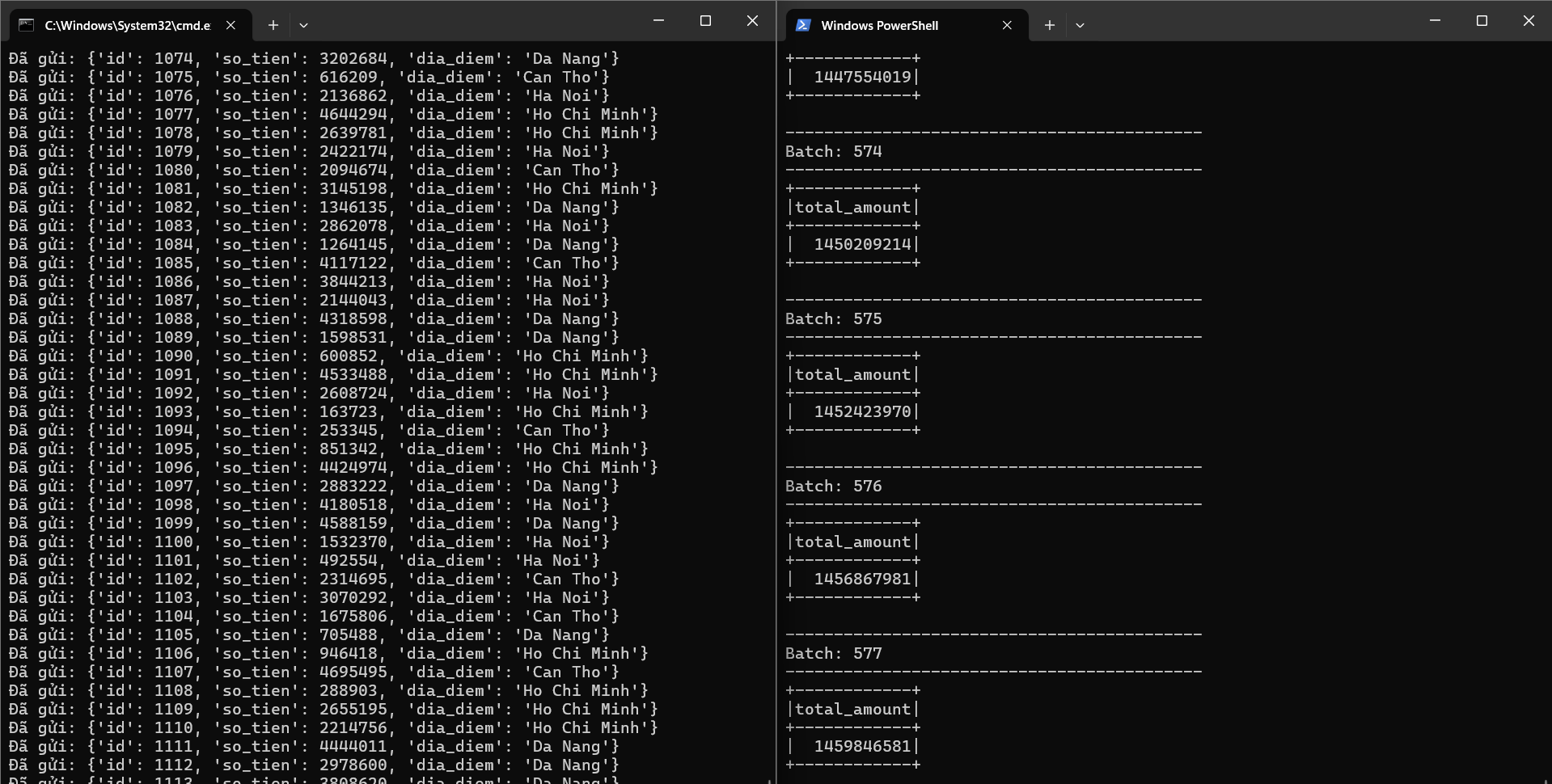


docker exec spark-master /opt/bitnami/spark/bin/spark-submit --packages org.apache.spark:spark-streaming-kafka-0-10\_2.12:3.3.0 /opt/bitnami/spark/spark\_consumer.py

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\_consumer2.py

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





Chạy bên dưới :

|

|

|

V

**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** để phục vụ cho câu hỏi tương tác ở cuối bài.

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 import SparkContext

from pyspark.streaming import StreamingContext

from pyspark.streaming.kafka import KafkaUtils

# Hàm để xử lý và tính tổng số tiền trong mỗi RDD

def process\_rdd(rdd):

if not rdd.isEmpty():

total\_amount = rdd.map(lambda record: json.loads(record[1])) \

.map(lambda data: data.get('so\_tien', 0)) \

.reduce(lambda a, b: a + b)

print("-------------------------------------------")

print(f"Tổng số tiền trong batch này: {total\_amount:,} VND")

print("-------------------------------------------")

# Cấu hình Spark

sc = SparkContext(appName="KafkaSparkStreamingDemo")

sc.setLogLevel("WARN") # Giảm bớt log thừa cho dễ nhìn

# Tạo StreamingContext với batch interval là 5 giây

ssc = StreamingContext(sc, 5)

# Thông số kết nối Kafka

# Lưu ý: Vì Spark chạy trong Docker network, nó sẽ kết nối tới Kafka

# qua tên service 'kafka' và port 9092.

kafka\_params = {

"bootstrap.servers": "kafka:9092",

"group.id": "spark-streaming-group" # Định danh consumer group

}

topic = "giao\_dich"

# Tạo Direct Stream từ Kafka

# Đây chính là cách sử dụng KafkaUtils.createDirectStream

kafka\_stream = KafkaUtils.createDirectStream(

ssc,

[topic],

kafka\_params

)

# Xử lý mỗi RDD trong DStream

kafka\_stream.foreachRDD(process\_rdd)

# Bắt đầu xử lý

ssc.start()

ssc.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-streaming-kafka-0-8\_2.11:2.4.8 \

/opt/bitnami/spark/spark\_consumer.py

*(Lưu ý: Version của gói thư viện có thể cần thay đổi nếu version Spark của bạn khác. Image bitnami/spark:3.3 thường đi với Scala 2.12. Bạn có thể thử --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.3.0 nếu lệnh trên không hoạt động).*

* + Sau khi chạy, bạn sẽ thấy output từ Spark, cứ mỗi 5 giây nó sẽ in ra tổng số tiền giao dịch nhận được trong khoảng thời gian đó.