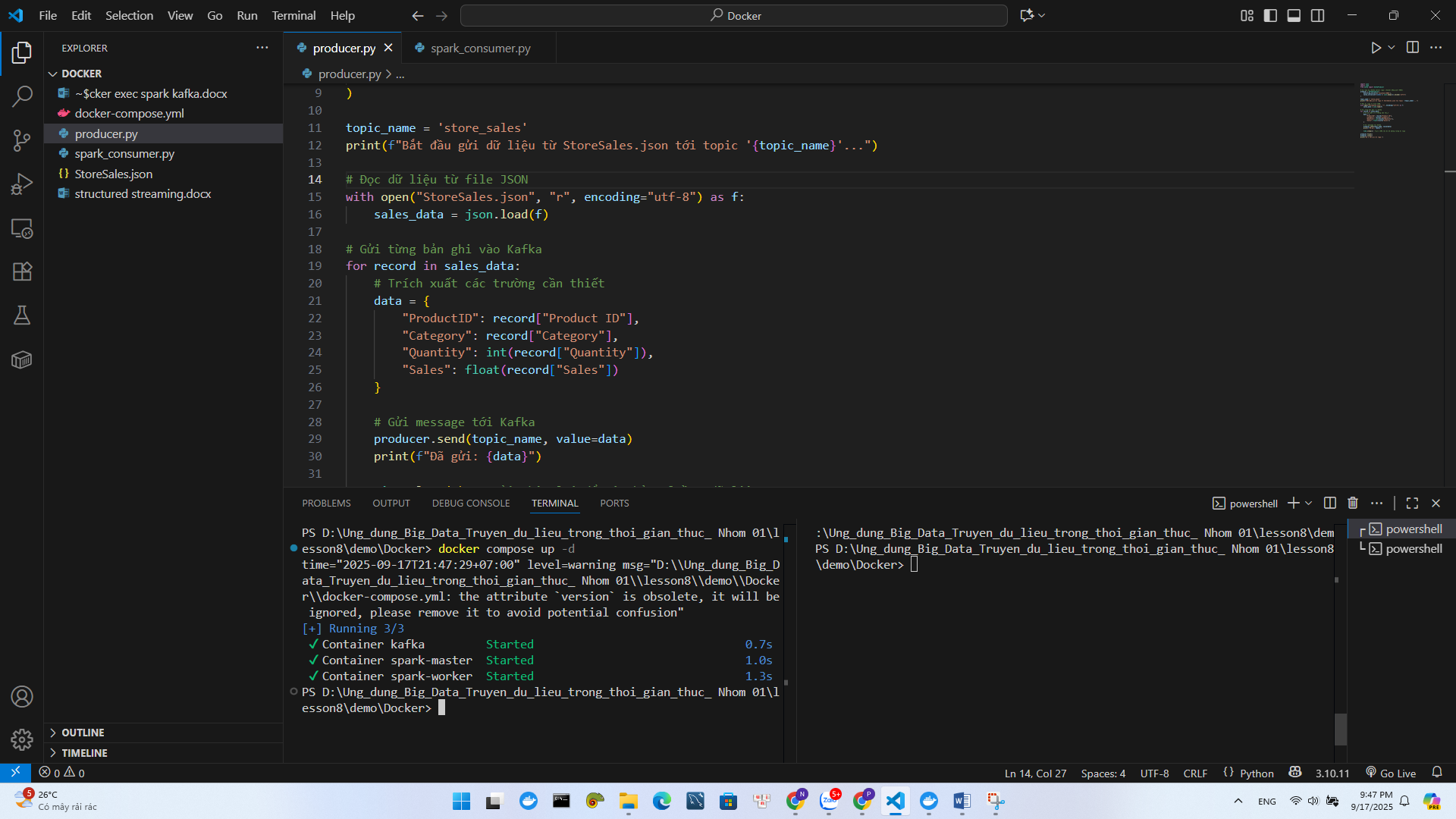
**Xử lý Dữ liệu có cấu trúc với Structured Streaming bằng Docker**

**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:
3. ***docker compose up -d***



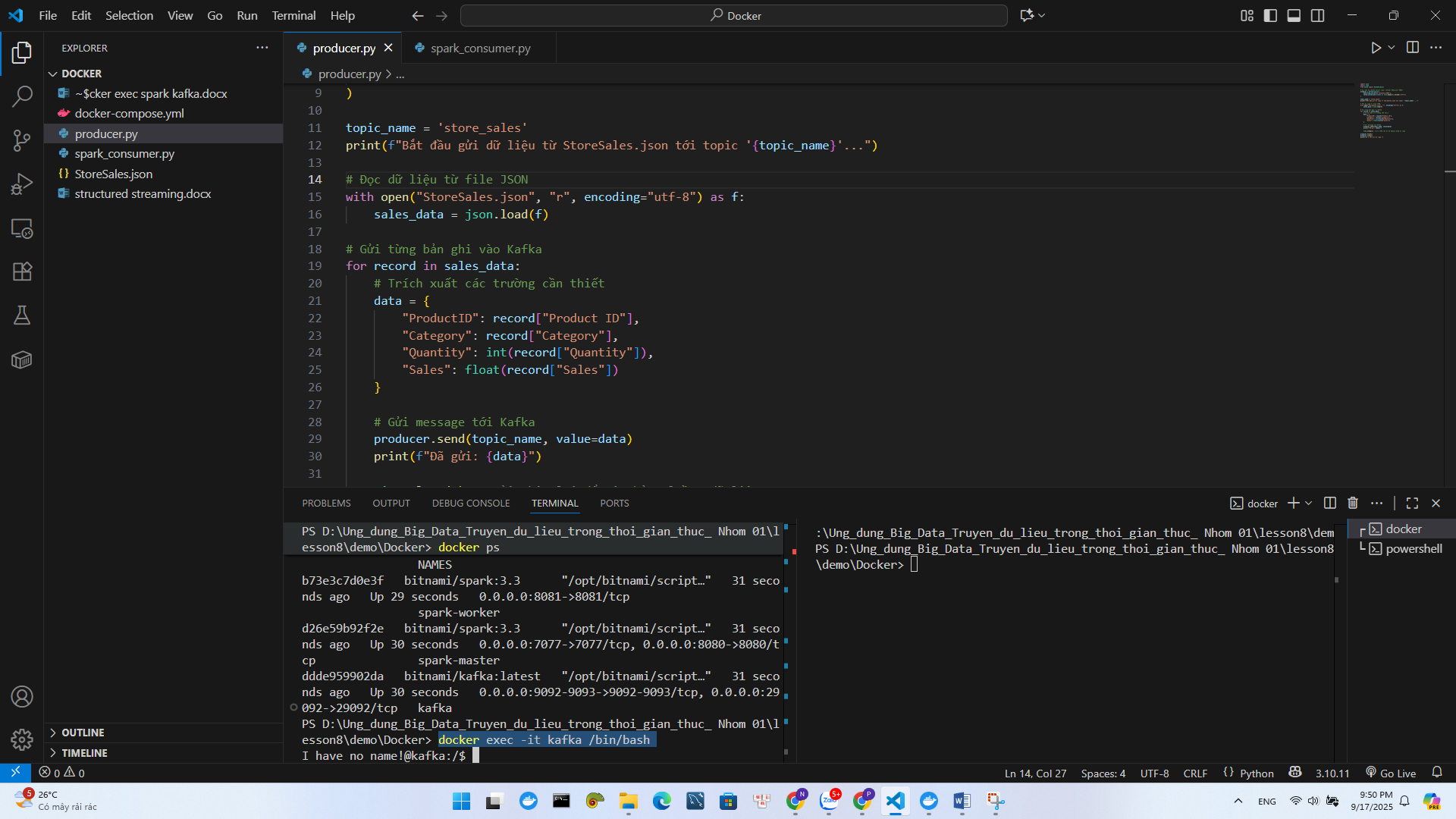
Lệnh này sẽ tải các image cần thiết và khởi chạy 3 container: kafka, spark-master, và spark-worker ở chế độ nền.

**Bước 2: Tạo Kafka Topic**

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

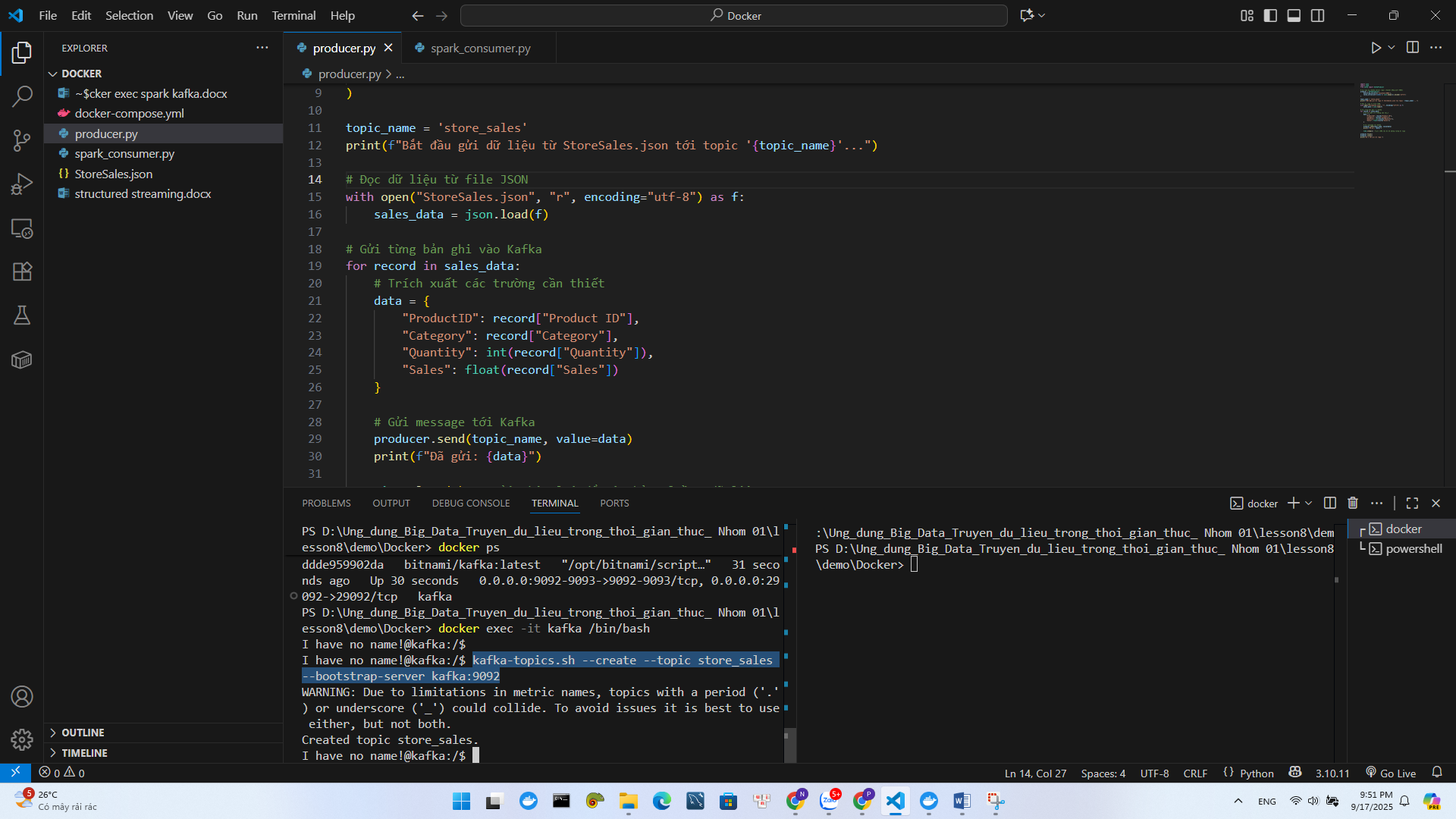
1. Truy cập vào container kafka đang chạy:

***docker exec -it kafka /bin/bash***

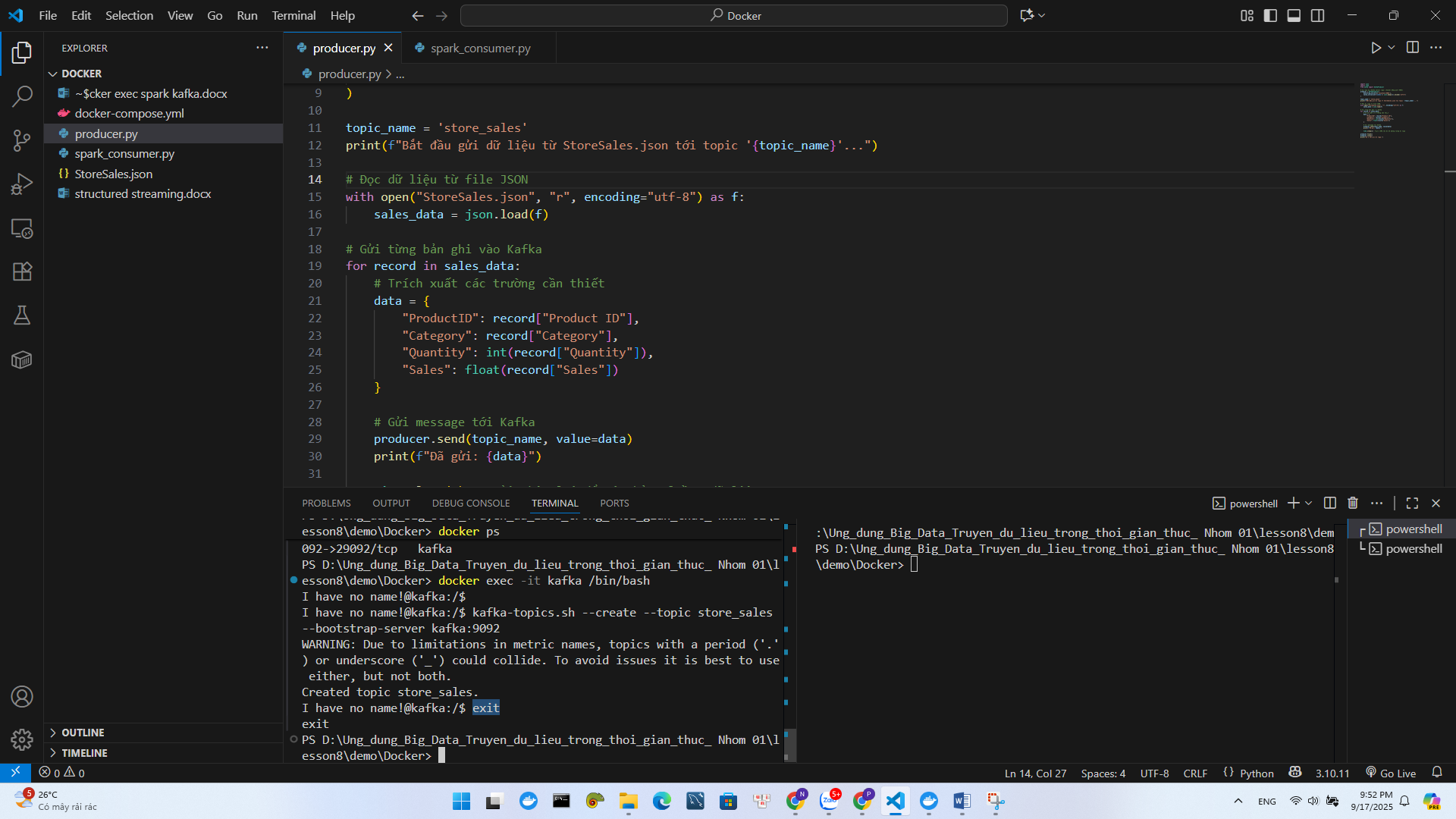


1. Sau khi đã vào trong container, chạy lệnh sau để tạo topic. Ở đây, chúng ta sẽ tạo topic với **3 partitions** để phục vụ cho câu hỏi tương tác ở cuối bài.

***kafka-topics.sh --create --topic store\_sales --bootstrap-server kafka:9092***



1. Gõ exit để thoát khỏi container Kafka.

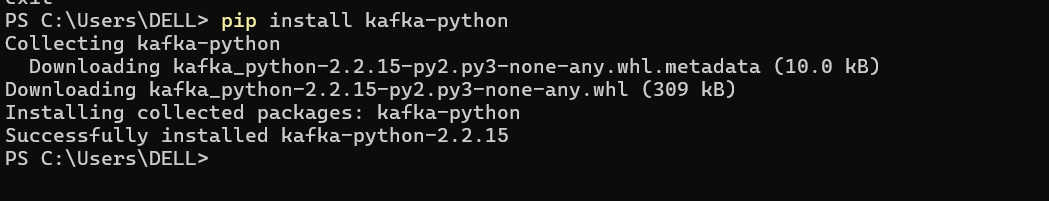


**Bước 3: Viết mã Producer bằng Python**

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

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

***pip install kafka-python***



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

Python

import json

import time

from kafka import KafkaProducer

# Kết nối đến Kafka server (ngoài Docker dùng port 9093)

producer = KafkaProducer(

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

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

)

topic\_name = 'store\_sales'

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

# Đọc dữ liệu từ file JSON

with open("StoreSales.json", "r", encoding="utf-8") as f:

    sales\_data = json.load(f)

# Gửi từng bản ghi vào Kafka

for record in sales\_data:

    # Trích xuất các trường cần thiết

    data = {

        "ProductID": record["Product ID"],

        "Category": record["Category"],

        "Quantity": int(record["Quantity"]),

        "Sales": float(record["Sales"])

    }

    # Gửi message tới Kafka

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

    print(f"Đã gửi: {data}")

    time.sleep(1)  # gửi chậm lại để mô phỏng luồng dữ liệu

producer.flush()

producer.close()

print("Hoàn tất gửi dữ liệu.")

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

Script này sẽ là ứng dụng Spark đọc dữ liệu từ Kafka và tính tổng doanh thu theo từng loại sản phẩm.

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, expr

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

# 1. Khởi tạo SparkSession

spark = SparkSession \

    .builder \

    .appName("KafkaSparkStructuredStreamingDemo") \

    .getOrCreate()

spark.sparkContext.setLogLevel("WARN")

# 2. Tạo streaming DataFrame (Đọc dữ liệu từ Kafka)

df = spark \

    .readStream \

    .format("kafka") \

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

    .option("subscribe", "store\_sales") \

    .load()

# Định nghĩa schema

schema = StructType([

    StructField("ProductID", StringType()),

    StructField("Category", StringType()),

    StructField("Quantity", IntegerType()),

    StructField("Sales", DoubleType()),

])

# 3. Thao tác trên streaming DataFrame (Parse JSON và xử lý)

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

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

    .select("data.\*")

# Tính doanh thu = Quantity \* Sales theo từng Category

result\_df = parsed\_df \

    .withColumn("Revenue", col("Quantity") \* col("Sales")) \

    .groupBy("Category") \

    .agg(sum("Revenue").alias("Total\_Revenue"))

# 4. Khởi động truy vấn trực tuyến (xuất kết quả ra console)

query = result\_df \

    .writeStream \

    .outputMode("complete") \

    .format("console") \

    .option("checkpointLocation", "/tmp/spark-checkpoints/store\_sales") \

    .start()

# 5. Quản lý truy vấn + 6. Theo dõi truy vấn

print("=== Query Status ===")

print(query.status)          # Thông tin trạng thái hiện tại

print("=== Query Last Progress ===")

print(query.lastProgress)    # Thông tin lần cập nhật gần nhất

# Chạy liên tục cho đến khi stop

query.awaitTermination()

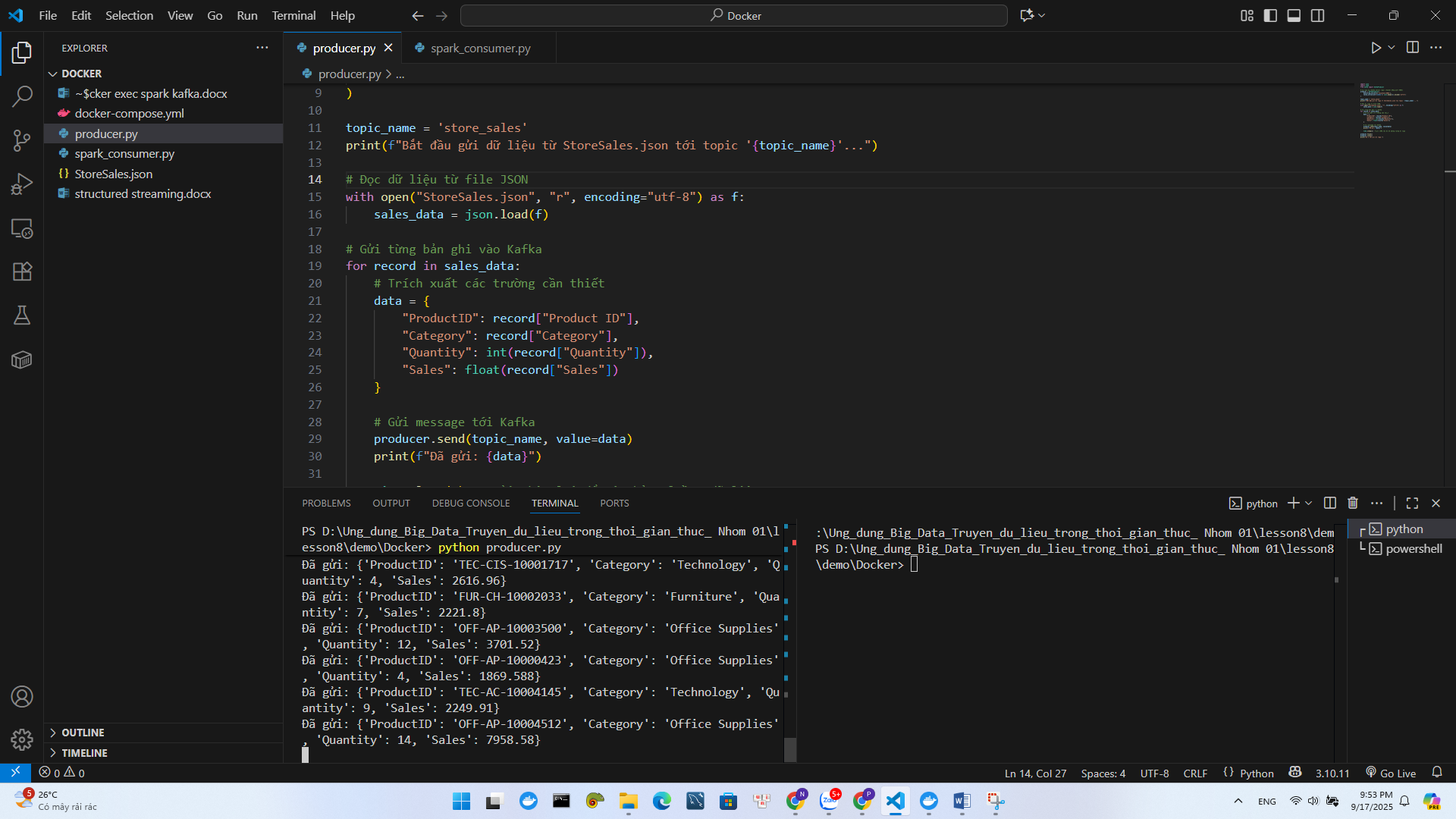
**Bước 5: Hướng dẫn chạy Demo**

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

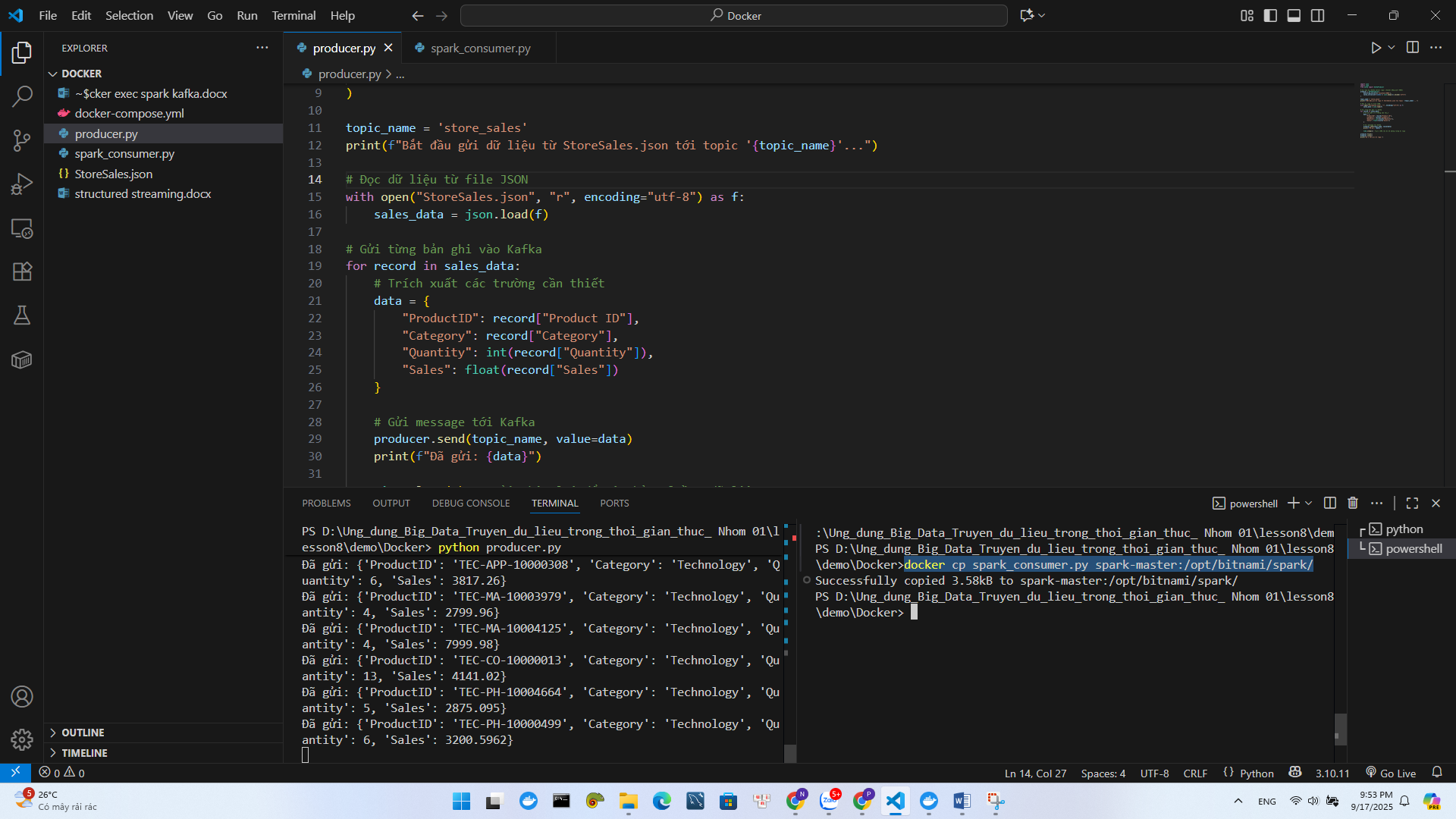
**python producer.py**



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:

**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.

***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***

