**Phạm Hoàng Long - MSV:21002218 - KTĐTTH**

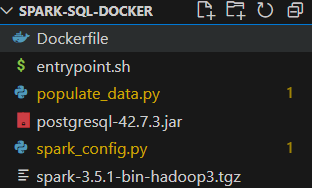
**Hướng dẫn Thiết lập và Sử dụng Docker Container cho SparkSQL và PostgreSQL**

**Mục tiêu**

Mục tiêu của dự án này là thiết lập và chạy SparkSQL trong một Docker container. Docker container sẽ chứa tất cả các phụ thuộc cần thiết để chạy Spark và một cơ sở dữ liệu PostgreSQL để lưu trữ dữ liệu. Người dùng sẽ có thể thực hiện các truy vấn SQL trên cơ sở dữ liệu này thông qua SparkSQL.

**Yêu cầu**

1. Docker: Đảm bảo Docker đã được cài đặt trên hệ thống .Có thể tải xuống và cài đặt Docker từ trang web chính thức của Docker.
2. Apache Spark: Truy cập trang web chính thức của Apache Spark và tải xuống phiên bản spark-3.5.1-bin-hadoop3.tgz. Đảm bảo rằng tệp này được lưu trong thư mục dự án
3. JDBC PostgreSQL: Có thể tải PostgreSQL từ trang web chính thức. Đảm bảo bạn đã tải xuống file JAR PostgreSQL JDBC driver và đặt nó trong cùng thư mục với Dockerfile
4. Python và PySpark: Đã cài đặt
5. Tệp và thư mục: Tạo một thư mục mới cho dự án và di chuyển tệp tin Spark đã tải xuống vào trong thư mục đó. Cấu trúc thư mục nên trông giống như sau



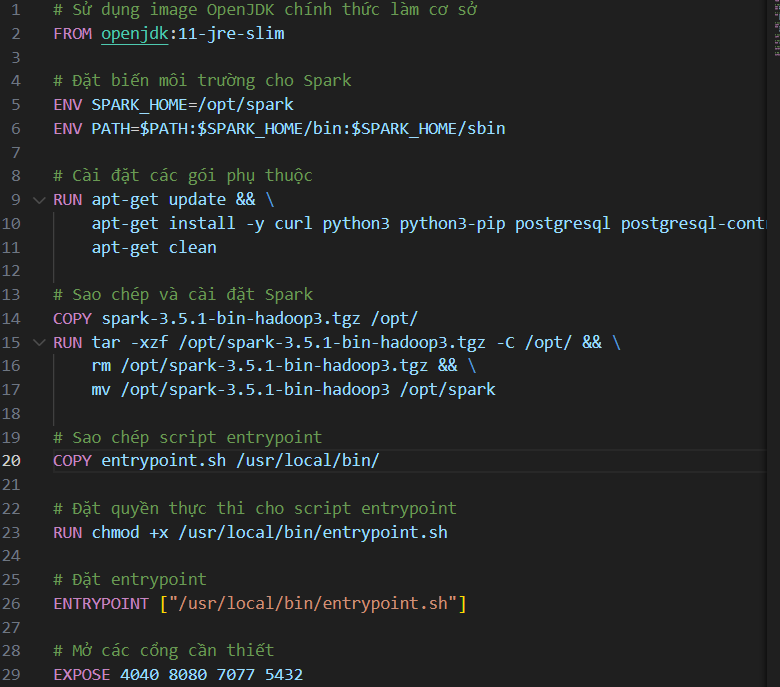
**Thực hành**

**Bước 1: Tạo Dockerfile và Script Entry Point**

**1. Tạo Dockerfile**

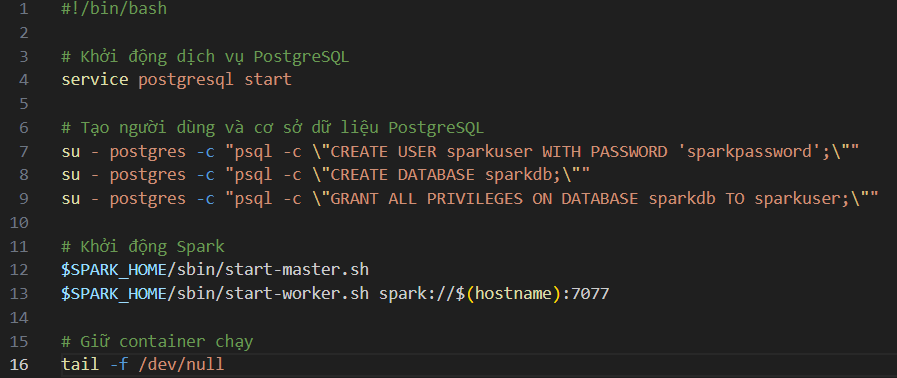
Tạo một tệp có tên là Dockerfile trong thư mục dự án.

Sử dụng `Dockerfile` đã cung cấp để xây dựng Docker image.



**2. Tạo Script Entry Point**

Tạo một tệp có tên là entrypoint.sh trong cùng thư mục với Dockerfile.



**Bước 2: Xây dựng và Chạy Docker Container**

**1. Xây dựng Docker Image**

Mở terminal và điều hướng đến thư mục chứa Dockerfile và các tệp khác. Sau đó, chạy lệnh sau để xây dựng Docker image:

docker build -t spark-sql-docker .

Lệnh này sẽ tạo một Docker image có tên spark-sql-docker sử dụng các chỉ thị trong Dockerfile.

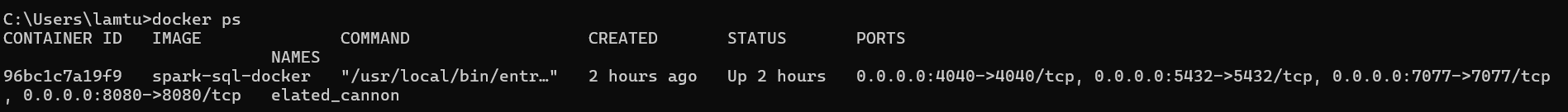
**2. Chạy Docker Container**

Sau khi xây dựng xong, chạy lệnh sau để khởi động Docker container:

docker run -it -p 4040:4040 -p 8080:8080 -p 7077:7077 -p 5432:5432 spark-sql-docker

* -it: Chạy container trong chế độ tương tác với terminal .
* -p 4040:4040 -p 8080:8080 -p 7077:7077 -p 5432:5432: Ánh xạ các cổng của container với các cổng trên máy chủ để có thể truy cập các dịch vụ bên trong container từ bên ngoài.

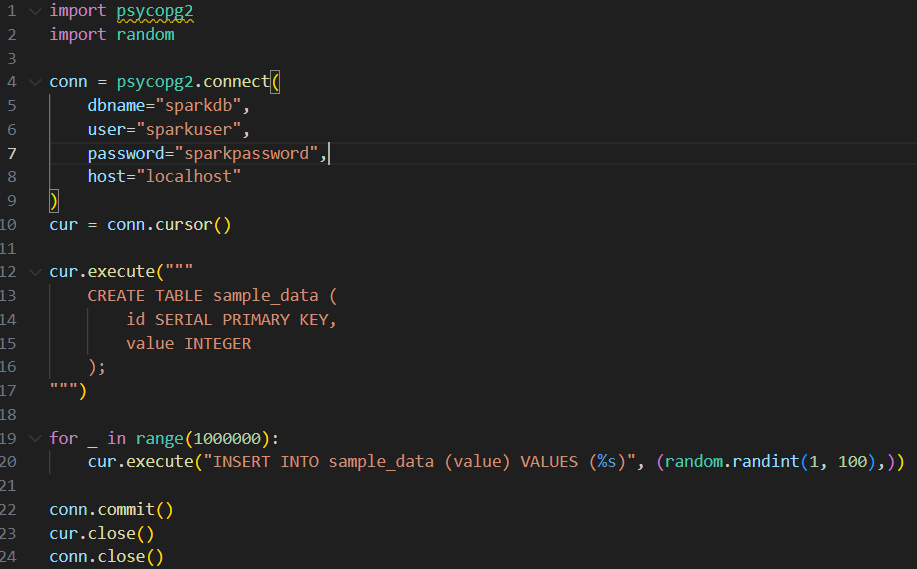
Sau khi chạy 2 lệnh trên, ta sẽ được đưa vào bên trong docker có dạng như sau



**Bước 3: Thêm Dữ liệu và Thực hiện Truy vấn**

**1. Thêm Dữ liệu vào Cơ sở Dữ liệu**

Tạo một tệp có tên là populate\_data.py để nhập dữ liệu mẫu vào cơ sở dữ liệu PostgreSQL:



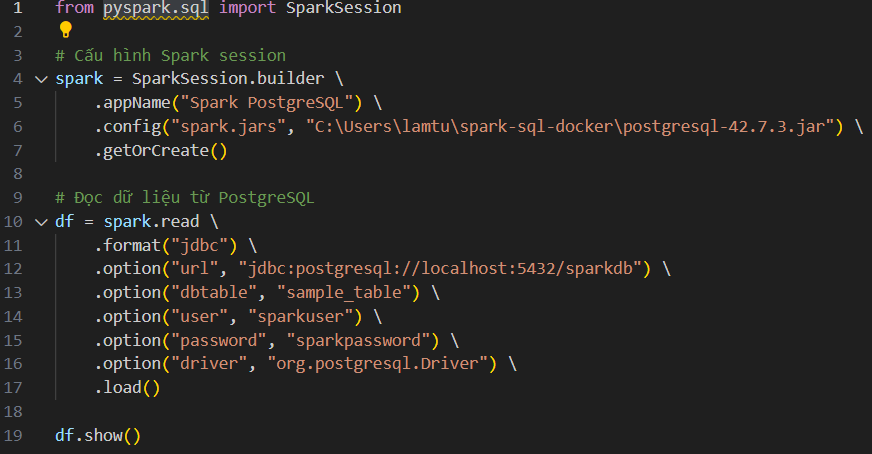
Chạy các lệnh sau trong một terminal mới để sao chép script Python vào container và thực thi nó để nhập dữ liệu mẫu vào cơ sở dữ liệu:

docker cp populate\_data.py <container\_id>:/populate\_data.py

docker exec -it <container\_id> python3 /populate\_data.py

**2. Thực hiện Truy vấn SparkSQL**

Tạo một tệp có tên là spark\_config.py để cấu hình Spark và thực hiện truy vấn SparkSQL:



Chạy các lệnh sau trong một terminal mới để sao chép file spark\_config.py vào container và cấu hình Spark để kết nối với cơ sở dữ liệu PostgreSQL và thực hiện truy vấn:

docker cp spark\_config.py 96bc1c7a19f9:/spark\_config.py

ocker cp C:\Users\lamtu\spark-sql-docker\postgresql-42.7.3.jar 96bc1c7a19f9: C:\Users\lamtu\spark-sql-docker\postgresql-42.7.3.jar

docker exec -it 96bc1c7a19f9 python3 /spark\_config.py

Khi thấy thông báo này, database đã được hoàn thành, ta có thể kiểm tra bằng cách đếm số lượng dòng trong database:

A screen shot of a computer

Description automatically generated

Chúng ta có thể thấy số lượng dòng trong database.

Sau đây là kết quả khi chạy các file ở trên:

A black background with white text

Description automatically generated

Thông báo về việc đã kết nối database và tạo temporary view.

A screen shot of a computer code

Description automatically generated

Kiểm tra thời gian count toàn bộ record và count có điều kiện WHERE và đóng kết nối server với database.