**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**TÌM HIỂU VỀ APACHE SPARK**

**Giáo viên hướng dẫn: Dương Văn Hải**

**Sinh viên thực hiện: 2015585 – Đỗ Hải Anh**

**2015606 – Phạm Lê Minh  
2011438 – Hoàng Ngọc Minh Thắng**

**Đà lạt, …/2023**

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

Đà Lạt, ngày ….. tháng …… năm ……

Giáo viên hướng dẫn

[Ký tên và ghi rõ họ tên]

I. Apache Spark là gì ?

1. Khái niệm

Apache Spark là một framework mã nguồn mở được thiết kế để xử lý và tính toán dữ liệu lớn một cách hiệu quả, mang lại tốc độ và khả năng mở rộng đáng kể. Nó đặc biệt mạnh mẽ trong việc xử lý dữ liệu liên tục, cơ sở dữ liệu đồ thị, machine learning, và các ứng dụng trí tuệ nhân tạo. Được phát triển từ năm 2009 tại AMPLab của Đại học California, Berkeley, Apache Spark đã trở thành một công cụ quan trọng trong ngôi nhà của Apache Software Foundation.

2. Tính Năng Chính:

Tốc Độ Tính Toán: Apache Spark nổi bật với khả năng xử lý dữ liệu gấp 10 lần nhanh hơn so với các công cụ khác. Điều này đạt được thông qua việc sử dụng mô hình tính toán phân tán trên một cụm máy tính (Computer Cluster), giúp tận dụng hiệu suất cao của nhiều máy tính cùng một lúc.

Khả Năng Mở Rộng: Spark hỗ trợ khả năng mở rộng linh hoạt, cho phép mở rộng quy mô hệ thống theo nhu cầu. Điều này làm cho Spark trở thành một lựa chọn ưu việt khi cần xử lý dữ liệu lớn và đạt được hiệu suất tốt trên các cụm máy tính lớn.

Hỗ Trợ Đa Ngôn Ngữ: Spark cung cấp khả năng lập trình bằng nhiều ngôn ngữ như Scala, Java, Python, R, và SQL. Điều này giúp người phát triển có thể chọn ngôn ngữ phù hợp với kỹ năng và yêu cầu cụ thể của dự án.

3. Xử Lý Dữ Liệu Liên Tục (Streaming):

Apache Spark không chỉ mạnh mẽ trong xử lý dữ liệu tĩnh mà còn là một giải pháp xuất sắc cho xử lý dữ liệu liên tục. Module Spark Streaming cho phép xử lý dữ liệu đến từ các nguồn như Apache Kafka, Flume, hay S3. Điều này làm cho Spark thích hợp cho các ứng dụng yêu cầu phản ứng thời gian thực, như theo dõi trực tuyến và phân tích dữ liệu đối với các sự kiện ngay khi chúng xảy ra.

4. Hỗ Trợ Cơ Sở Dữ Liệu Đồ Thị:

Một trong những ưu điểm quan trọng của Apache Spark là khả năng xử lý dữ liệu đồ thị. GraphX, một thư viện tích hợp trong Spark, cung cấp các API để xây dựng và xử lý đồ thị trên cụm máy tính. Điều này mở rộng khả năng ứng dụng của Spark đến lĩnh vực phân tích mạng xã hội, tìm kiếm liên kết, và nhiều ứng dụng khác liên quan đến cơ sở dữ liệu đồ thị.

5. Học Máy và Trí Tuệ Nhân Tạo:

Spark cung cấp thư viện MLlib cho machine learning, giúp người phát triển triển khai các mô hình học máy trên dữ liệu lớn một cách hiệu quả. Điều này làm cho Spark trở thành một nền tảng phù hợp cho việc phát triển và triển khai ứng dụng trí tuệ nhân tạo, kết hợp sức mạnh của xử lý dữ liệu lớn và học máy.

6. Cộng Đồng và Phổ Cập:

Do thuộc dự án Apache, Spark hưởng lợi từ một cộng đồng đa dạng và tích cực. Người dùng có thể tận dụng từ nguồn kiến thức phong phú và sự hỗ trợ từ cộng đồng để giải quyết vấn đề và tối ưu hóa hiệu suất của họ.

7. Kết Luận:

Trong thời đại của big data, Apache Spark đóng vai trò quan trọng trong việc xử lý và phân tích dữ liệu lớn một cách hiệu quả. Với khả năng xử lý dữ liệu liên tục, hỗ trợ đồ thị, và machine learning, Spark đem lại sự linh hoạt và hiệu suất mà nhiều dự án yêu cầu. Bằng cách hỗ trợ nhiều ngôn ngữ lập trình và có một cộng đồng mạnh mẽ, Apache Spark tiếp tục là một trong những công cụ quan trọng giúp các tổ chức và nhà phát triển khai thác toàn bộ tiềm năng của dữ liệu lớn.

II. Apache Spark bao gồm

Apache gồm có các thành phần chính sau đây:

Spark Core

Spark Core là thành phần cốt lõi của Apache Spark, các thành phần khác muốn hoạt động đều cần thông qua Spark Core.

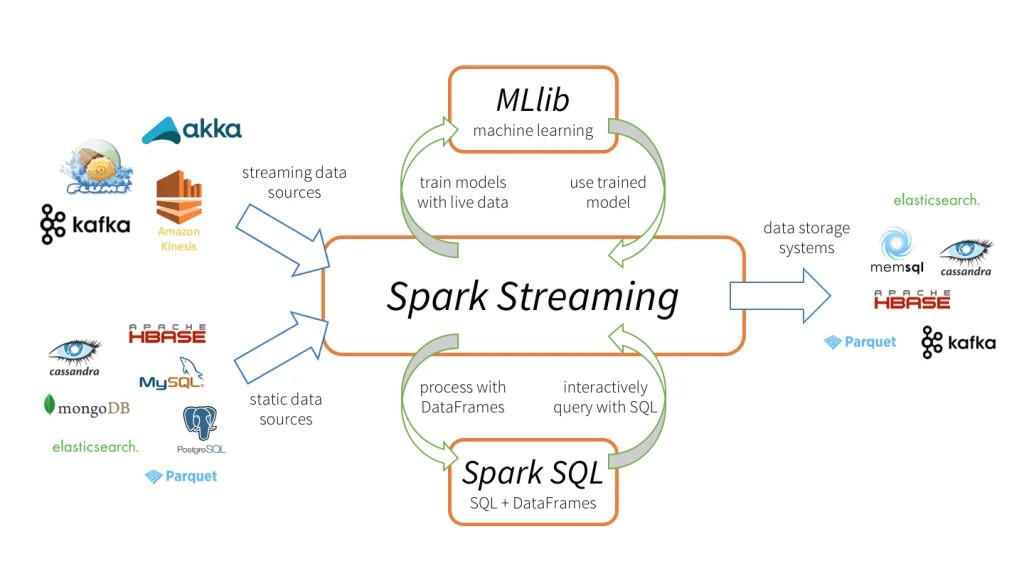
Spark Core cung cấp các chức năng cơ bản để xử lý dữ liệu phân tán (distributed data processing - DDP) bao gồm quản lý bộ nhớ, thiết lập lịch tác vụ và khôi phục lỗi.

Spark SQL

Spark SQL cho phép người dùng truy vấn dữ liệu bằng ngôn ngữ SQL trên các tập dữ liệu lớn. Spark SQL cung cấp một cơ chế để tích hợp với các cơ sở dữ liệu quan hệ như MySQL, Oracle hoặc PostgreSQL và xử lý dữ liệu bằng cách sử dụng các tính năng phân tán của Spark.

Ngoài ra, Spark SQL còn tích hợp tốt với các công cụ ETL và Spark Streaming, điều đó giúp nó trở thành một phần quan trọng trong hệ thống phân tích dữ liệu phân tán và các ứng dụng trực tuyến trong thời gian thực.

Spark Streaming



1. Spark Core: Cốt Lõi Động Cơ của Apache Spark

Spark Core là trái tim của Apache Spark, đóng vai trò là thành phần cốt lõi mà các module khác phải thông qua để hoạt động. Được xem là "động cơ" của hệ thống, Spark Core cung cấp các chức năng quan trọng để xử lý dữ liệu phân tán (Distributed Data Processing - DDP). Trong đó, nổi bật là quản lý bộ nhớ, thiết lập lịch tác vụ và khôi phục lỗi, giúp tối ưu hóa hiệu suất toàn hệ thống.

2. Spark SQL: Kết Nối Dữ Liệu Lớn Với Ngôn Ngữ SQL

Spark SQL mở ra khả năng truy vấn dữ liệu bằng ngôn ngữ SQL trực tiếp trên các tập dữ liệu lớn. Điều này không chỉ giúp tạo sự thuận tiện cho những người quen thuộc với SQL mà còn tích hợp tốt với các cơ sở dữ liệu quan hệ như MySQL, Oracle, hoặc PostgreSQL. Spark SQL trở thành công cụ quan trọng cho việc xử lý dữ liệu lớn, đặc biệt là khi kết hợp với các tính năng phân tán của Spark, và làm nền tảng cho hệ thống phân tích dữ liệu phân tán và ứng dụng trực tuyến.

3. Spark Streaming: Xử Lý Dữ Liệu Trực Tiếp Trong Thời Gian Thực

Module Spark Streaming mang lại khả năng xử lý dữ liệu trực tiếp trong thời gian thực. Nó tách dữ liệu thành các microbatch nhỏ và xử lý chúng qua API Apache Spark. Spark Streaming không chỉ giúp phân tích dữ liệu trực tuyến mà còn tích hợp linh hoạt với các module khác như Spark SQL và MLlib. Ứng dụng rộng rãi của nó bao gồm phân tích dữ liệu trực tuyến, phát hiện sự cố hệ thống và giám sát dữ liệu trong thời gian thực.

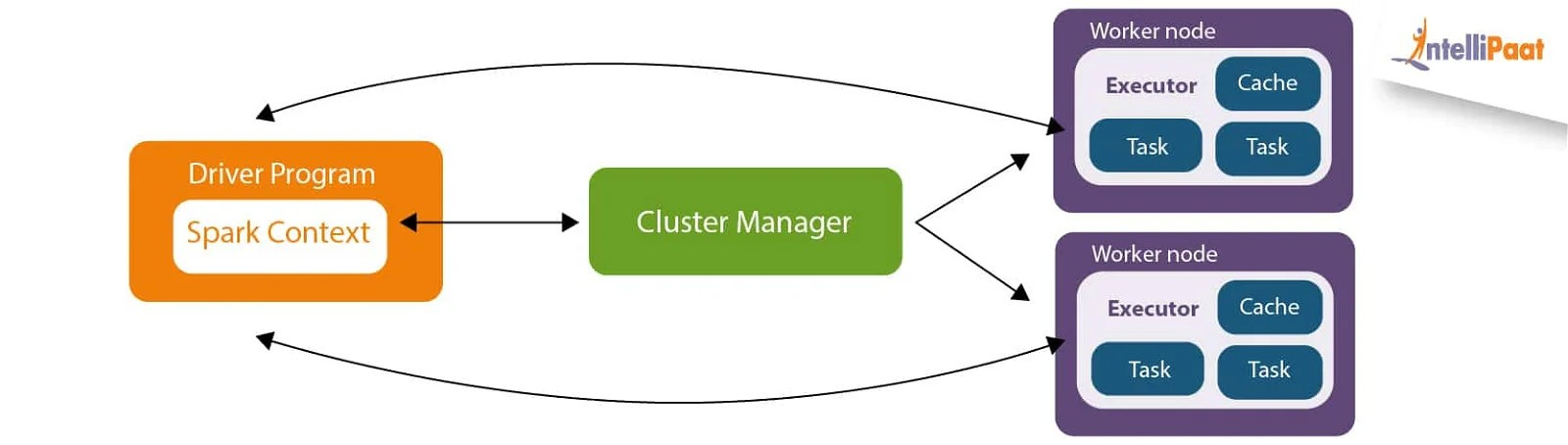
4. Spark MLlib (Machine Learning Library): Sức Mạnh Machine Learning Trên Big Data

Spark MLlib là thư viện Machine Learning tích hợp sẵn trong Apache Spark, mang lại các thuật toán Machine Learning phổ biến và hiệu quả cho việc xử lý big data. Với hỗ trợ cho các thuật toán như Regression, Classification, Clustering, Collaborative Filtering, Dimensionality Reduction, Feature Extraction and Transformation, MLlib là một nguồn tài nguyên quan trọng trong nền tảng Spark. Điều này giúp đơn giản hóa quá trình phát triển và triển khai mô hình học máy trên dữ liệu lớn.

5. GraphX: Xử Lý Đồ Thị Với Công Nghệ Phân Tán

GraphX là thành phần của Apache Spark chuyên về xử lý cấu trúc đồ thị. Với sự tích hợp của RDD, VertexRDD và EdgeRDD, GraphX cung cấp các công cụ mạnh mẽ để thực hiện các tính toán trên đồ thị. Điều này mở ra nhiều ứng dụng trong việc phân tích mạng xã hội, tìm kiếm liên kết và các lĩnh vực liên quan đến cơ sở dữ liệu đồ thị.

III. Kiến trúc của Apache Spark



Apache Spark là một framework mã nguồn mở được thiết kế để xử lý dữ liệu Big Data một cách nhanh chóng và hiệu quả. Kiến trúc của Apache Spark được thiết kế để có thể mở rộng quy mô để đáp ứng nhu cầu của các ứng dụng Big Data lớn.

Kiến trúc của Apache Spark bao gồm hai thành phần chính:

Driver program: Là chương trình chính của ứng dụng Spark, chạy trên một node trong cluster, quản lý và điều khiển quá trình xử lý trên cluster.

Executors: Là các tiến trình chạy trên các node khác trong cluster, được quản lý bởi driver program để thực hiện các nhiệm vụ xử lý dữ liệu.

Driver program

Driver program là thành phần quan trọng nhất trong kiến trúc của Apache Spark. Nó thực hiện các chức năng sau:

Tạo Spark Context: Spark Context là một đối tượng cung cấp các API để truy cập các tính năng của Apache Spark.

Tạo và quản lý các RDD: RDD là một cấu trúc dữ liệu phân tán có thể được sử dụng để lưu trữ và xử lý dữ liệu Big Data.

Phân phối các tác vụ xử lý dữ liệu cho các executors.

Giám sát tiến trình xử lý dữ liệu.

Spark Context

Spark Context là một đối tượng cung cấp các API để truy cập các tính năng của Apache Spark. Nó được tạo bởi driver program và được sử dụng để tạo và quản lý các RDD.

Spark Context có các chức năng sau:

Tạo RDD: RDD có thể được tạo từ nhiều nguồn khác nhau, bao gồm:

File: RDD có thể được tạo từ dữ liệu được lưu trữ trong các file.

Hệ thống tập tin phân tán: RDD có thể được tạo từ dữ liệu được lưu trữ trong các hệ thống tập tin phân tán, chẳng hạn như Hadoop Distributed File System (HDFS).

Bộ nhớ: RDD có thể được tạo từ dữ liệu được lưu trữ trong bộ nhớ.

Truy cập các tính năng của Apache Spark: Spark Context cung cấp các API để truy cập các tính năng của Apache Spark, chẳng hạn như:

Các API xử lý dữ liệu cơ bản: Các API này cung cấp các phương thức để thực hiện các phép toán cơ bản trên dữ liệu, chẳng hạn như lọc, sắp xếp, nhóm, v.v.

Các API xử lý dữ liệu nâng cao: Các API này cung cấp các phương thức để thực hiện các phép toán nâng cao trên dữ liệu, chẳng hạn như phân tích thống kê, học máy, v.v.

Executors

Executors là các tiến trình chạy trên các node khác trong cluster, được quản lý bởi driver program để thực hiện các nhiệm vụ xử lý dữ liệu.

Mỗi executor có thể chứa nhiều task (công việc) được giao để xử lý. Task là đơn vị nhỏ nhất của công việc xử lý dữ liệu trong Apache Spark.

Cách thức hoạt động của Apache Spark

Khi một ứng dụng Apache Spark được khởi chạy, driver program sẽ được tạo và chạy trên một node trong cluster. Driver program sẽ tạo Spark Context và sử dụng Spark Context để tạo các RDD.

Các RDD được phân phối trên các executors. Mỗi executor sẽ thực thi các task được giao bởi driver program. Task sẽ thực hiện các phép toán trên dữ liệu của RDD và trả về kết quả cho driver program.

Driver program sẽ giám sát tiến trình xử lý dữ liệu và thực hiện các hành động cần thiết nếu có lỗi xảy ra.

Ưu điểm của kiến trúc của Apache Spark

Kiến trúc của Apache Spark có các ưu điểm sau:

Mở rộng quy mô: Kiến trúc của Apache Spark có thể mở rộng quy mô để đáp ứng nhu cầu của các ứng dụng Big Data lớn. Điều này là do các executors có thể được thêm vào hoặc loại bỏ khỏi cluster một cách dễ dàng.

Tính hiệu quả: Kiến trúc của Apache Spark được thiết kế để có thể xử lý dữ liệu Big Data một cách nhanh chóng và hiệu quả. Điều này là do Apache Spark sử dụng các kỹ thuật xử lý song song để tối ưu hóa hiệu suất của các ứng dụng Big Data.

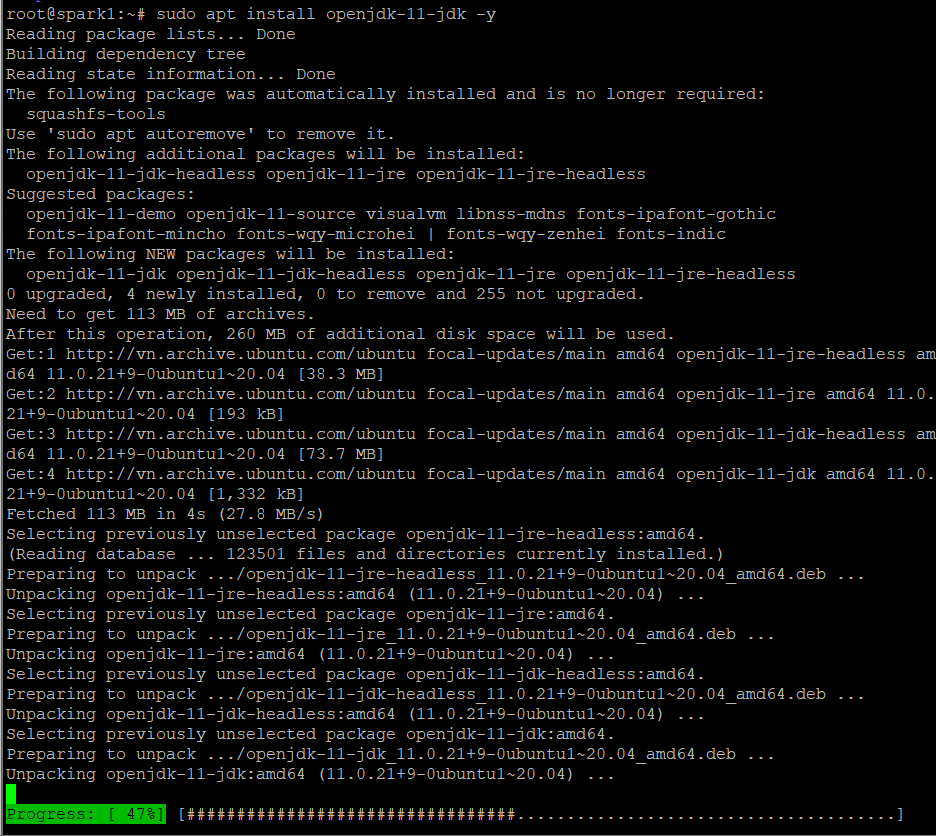
Tính linh hoạt: Kiến trúc của Apache Spark cung cấp khả năng linh hoạt cho các nhà phát triển ứng dụng Big Data. Điều này là do Apache Spark hỗ trợ nhiều loại dữ liệu và các phép toán xử lý dữ liệu.

IV. Nội dung đã làm được và demo

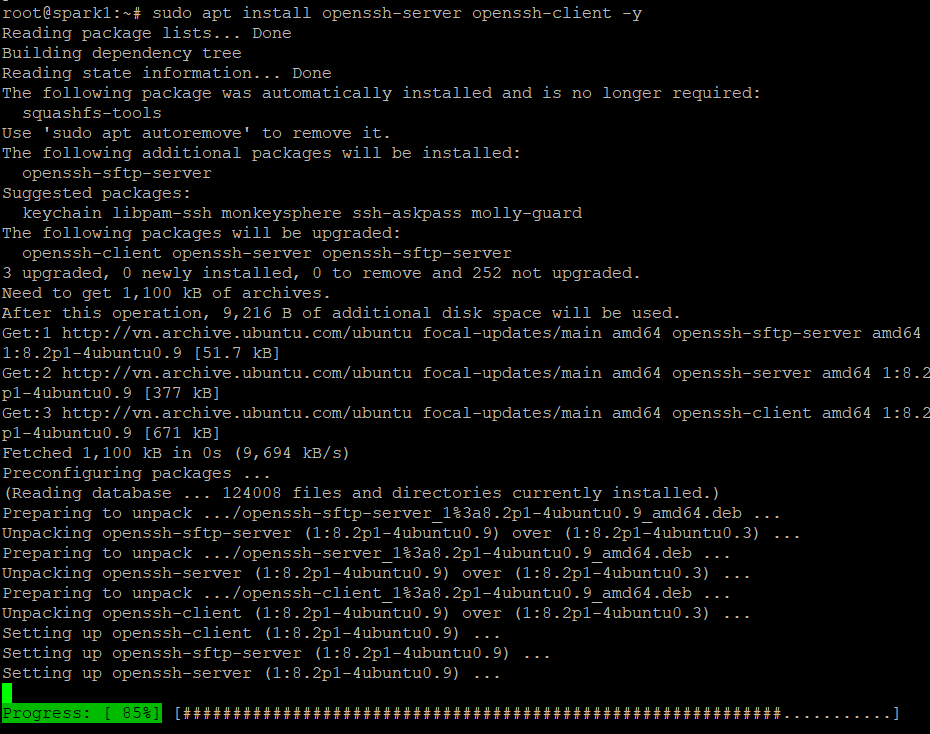
* Cấu hình hệ thống spark master và slave
* Xây dựng chương trình Word Count dựa trên sự phân tán của hệ thống spark master và slave trên hdfs

**Máy 1:**

Cài đặt các cấu hình Hadoop để sử dụng hdfs  
B1: sudo apt install openjdk-11-jdk -y



B2: sudo apt install openssh-server openssh-client -y

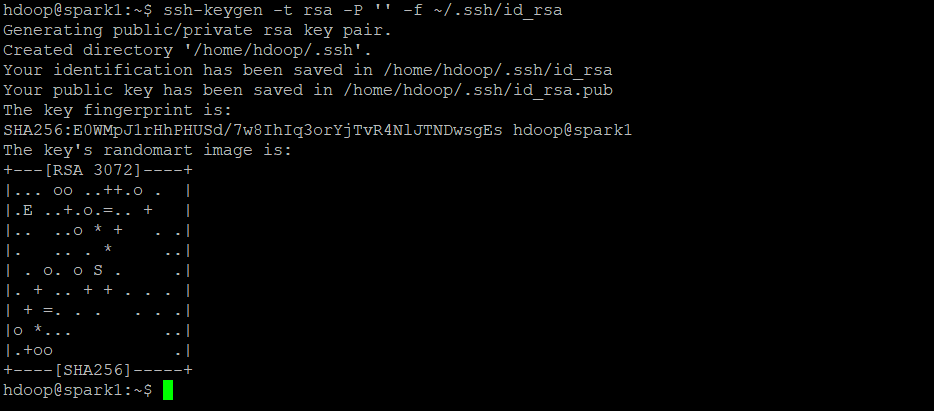


B3: sudo adduser hdoop

su - hdoop



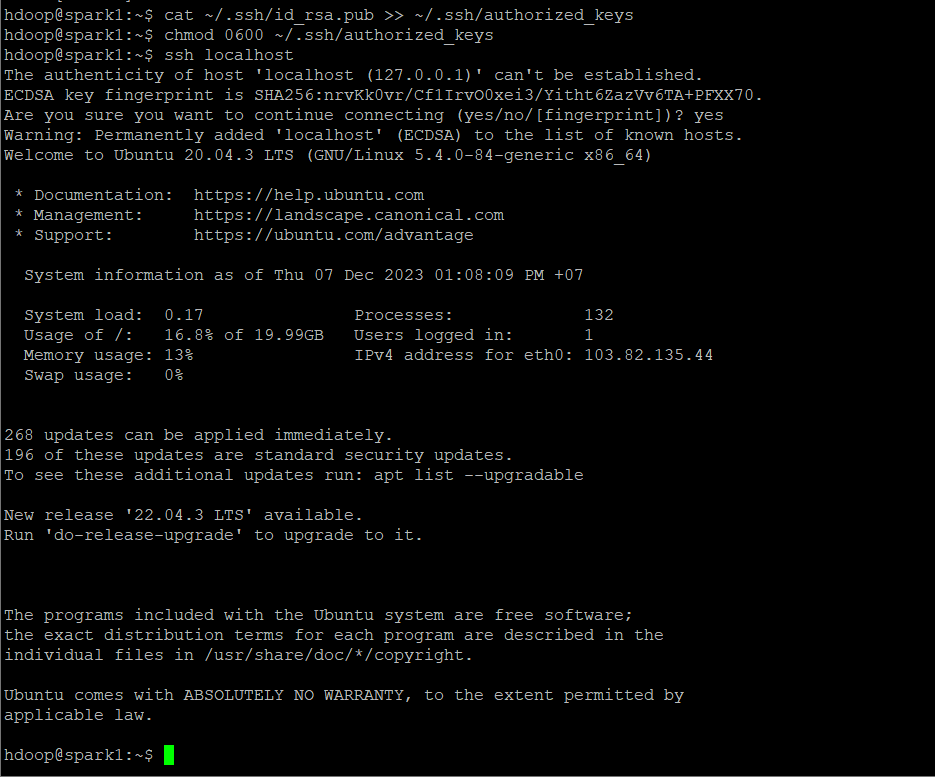
B4: ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa



B5: cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

chmod 0600 ~/.ssh/authorized\_keys

ssh localhost



B6: wget gói của hadoop mới nhất

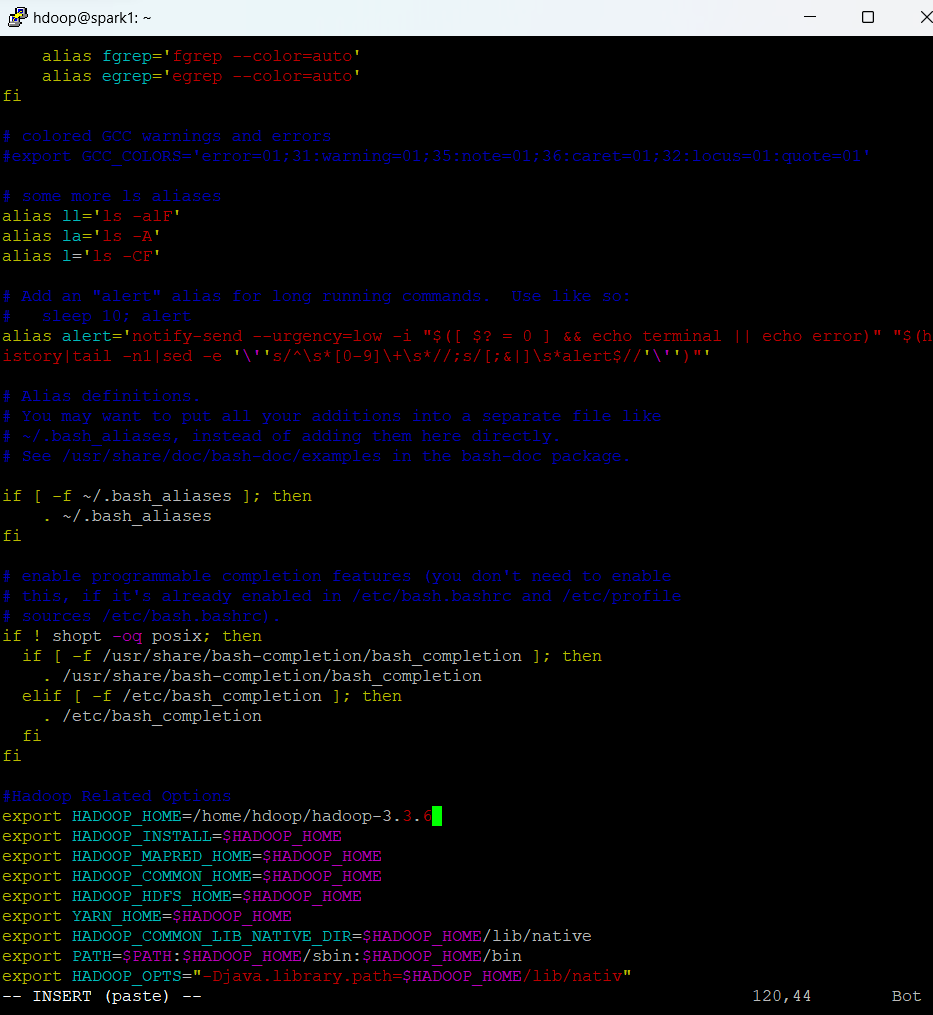


B7: tar xzf hadoop-3.3.6.tar.gz



B8: vi .bashrc và nhập các cấu hình



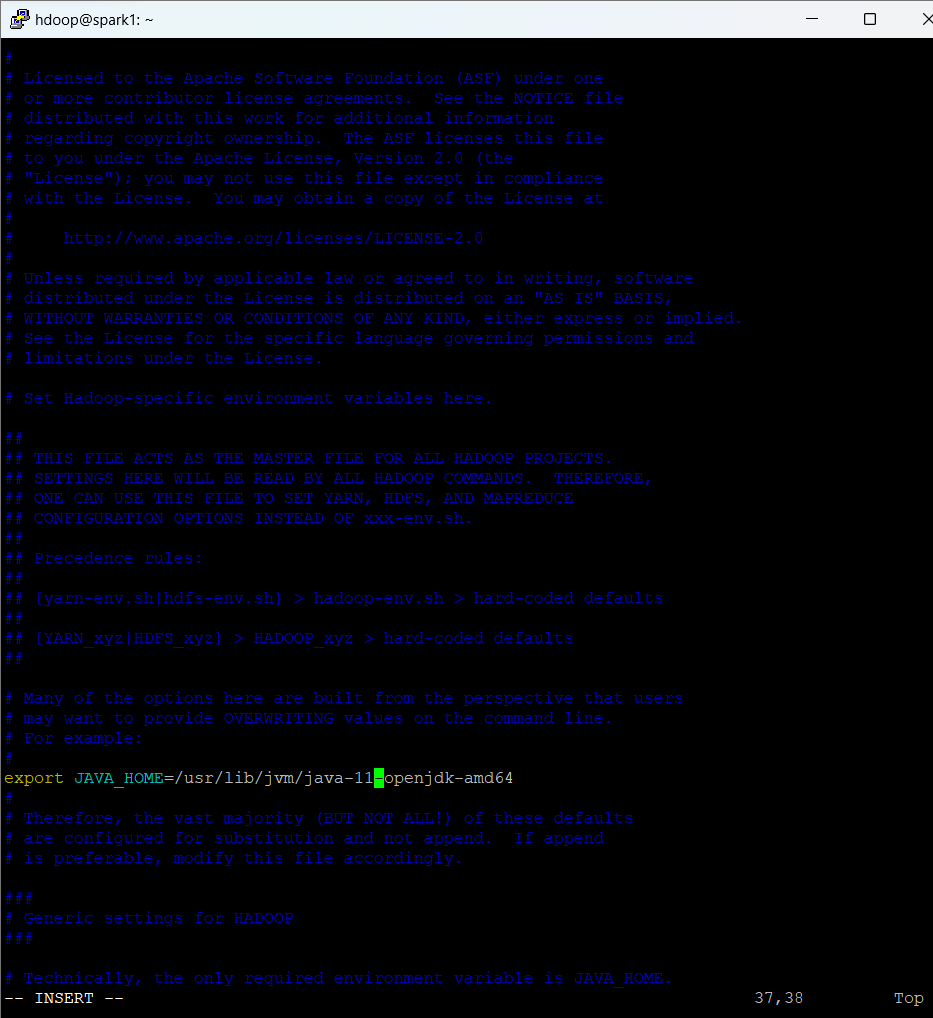


B9: source ~/.bashrc



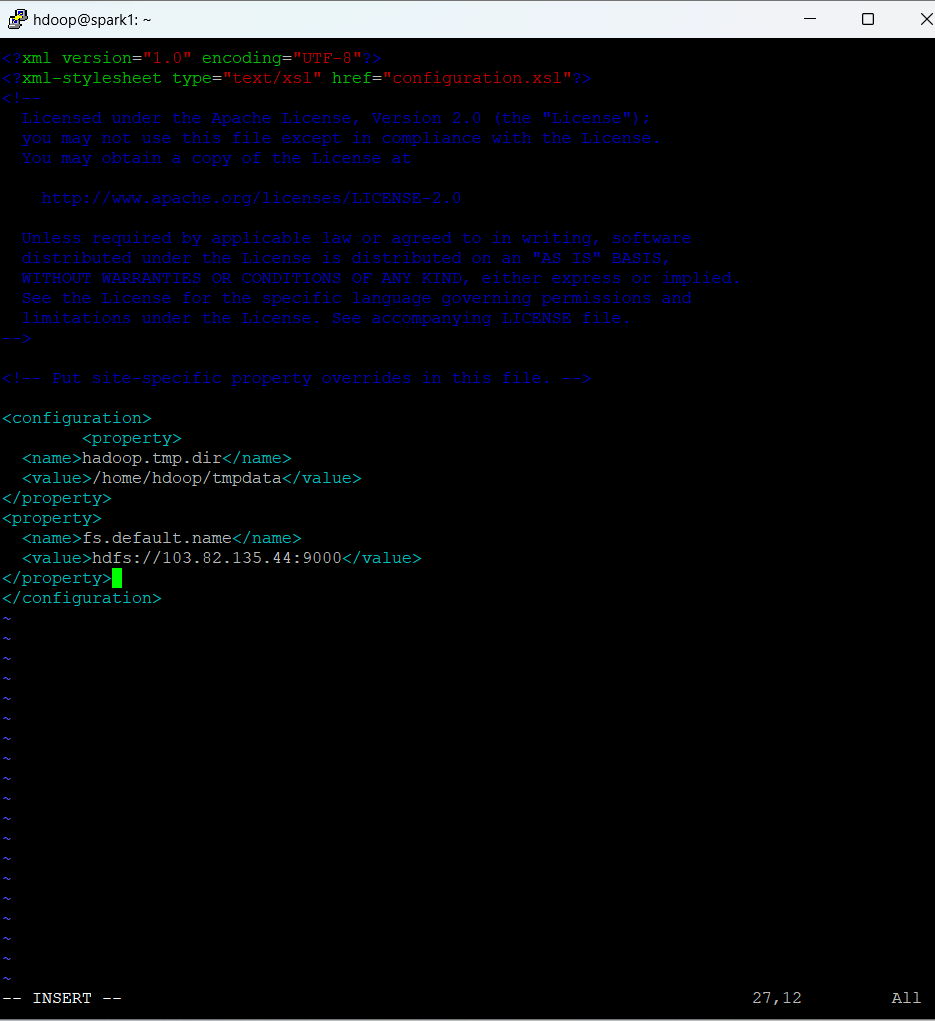
B10: vi hadoop-env.sh và nhập các cấu hình





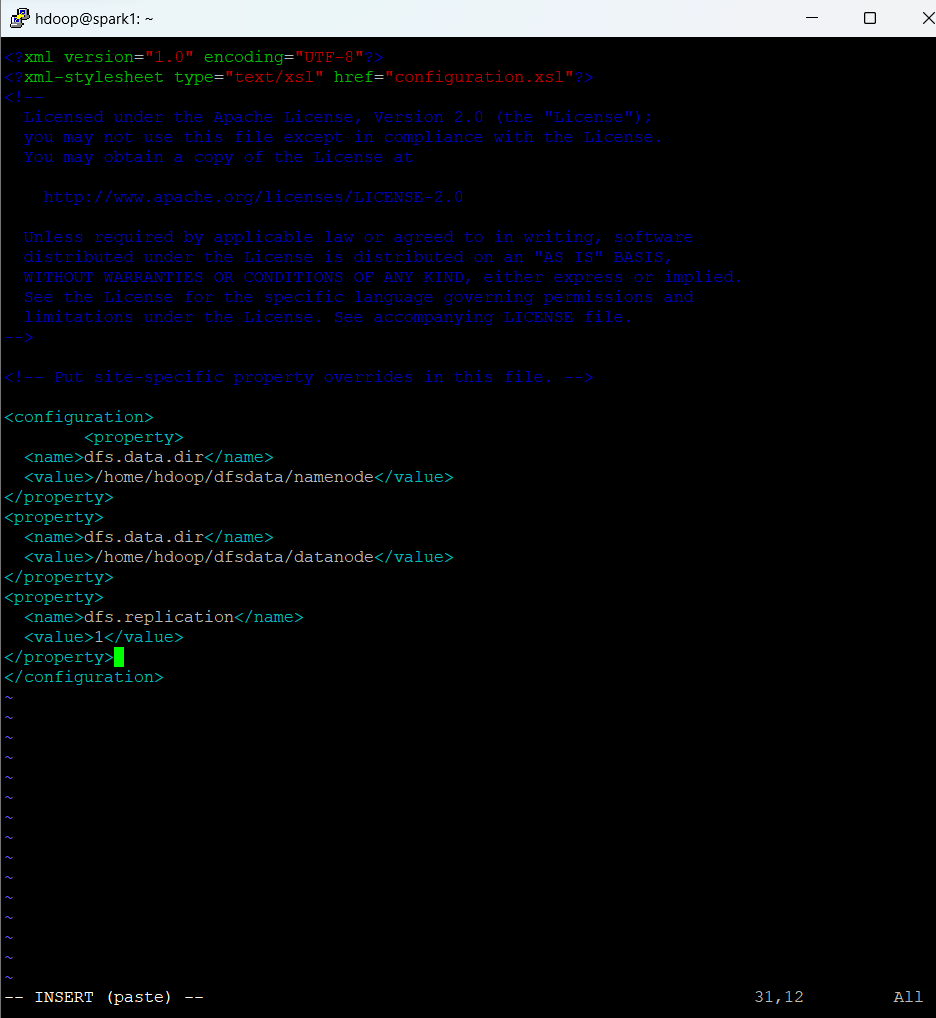
B11: vi core-site.xml và nhập các cấu hình





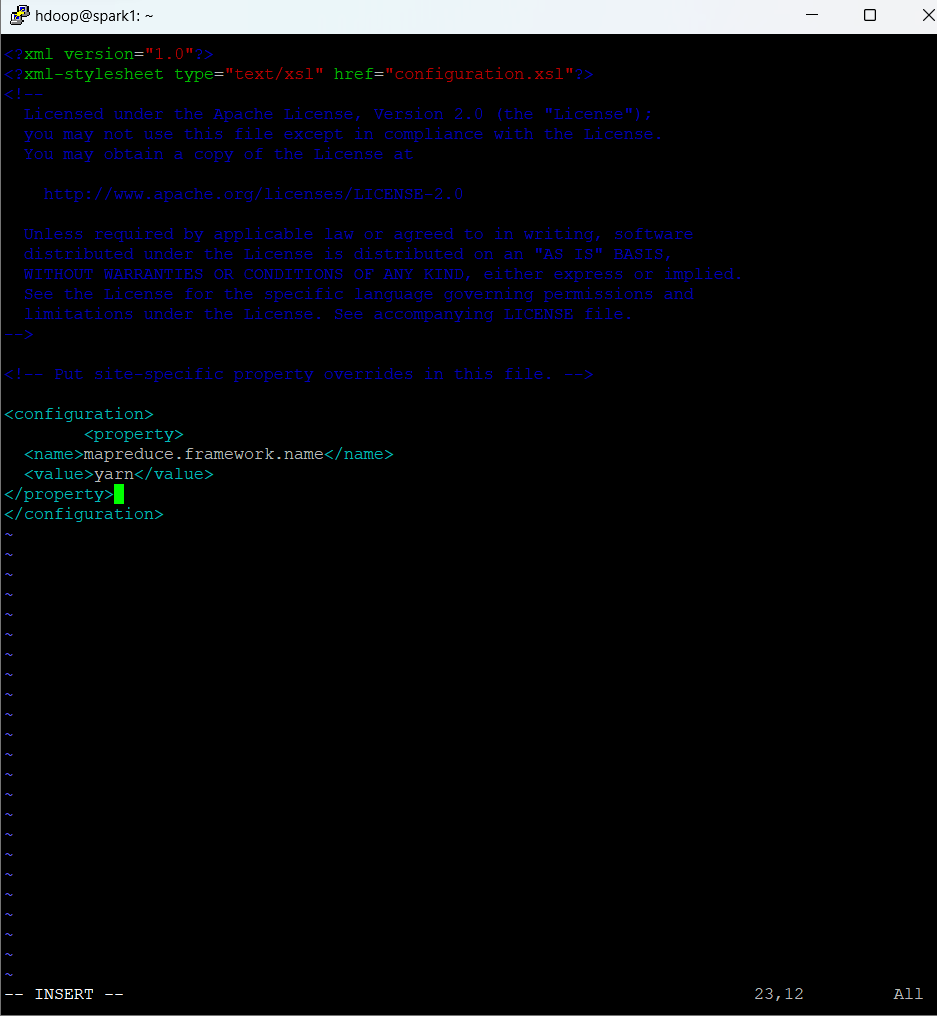
B12: vi hdfs-site.xml và nhập các cấu hình





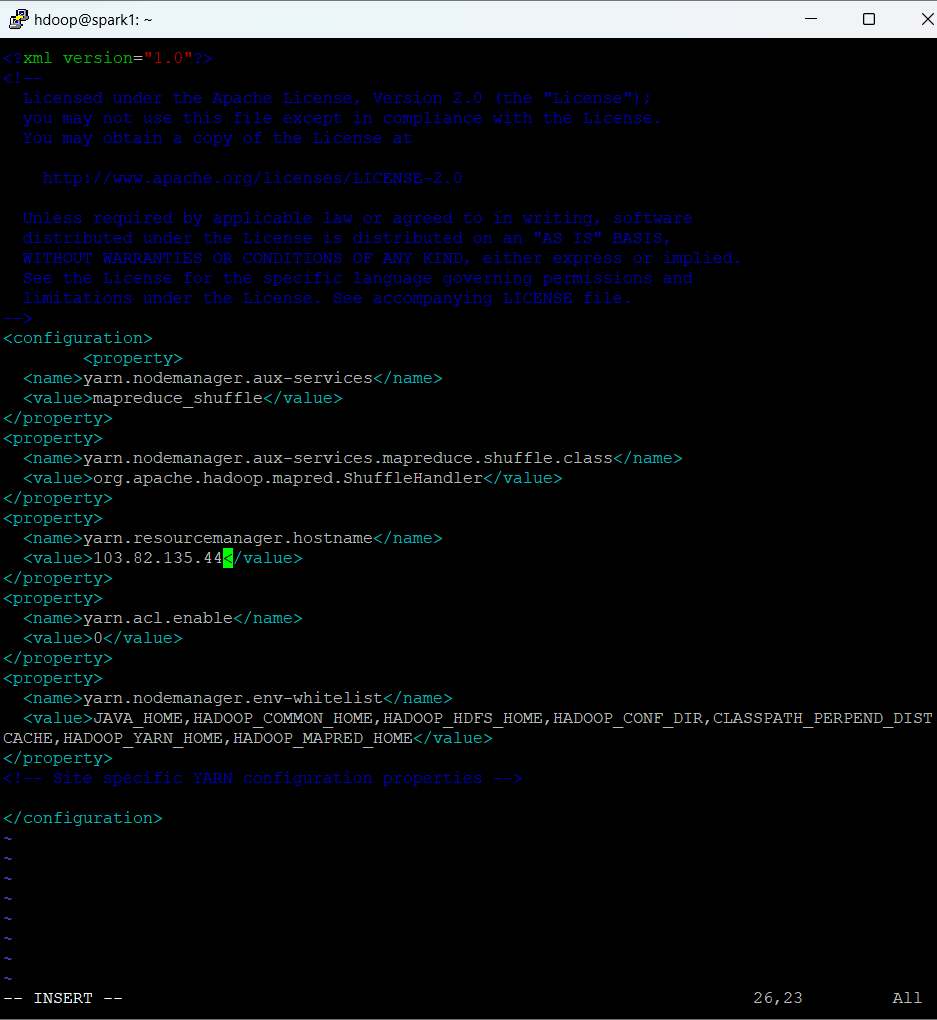
B13: vi mapred-site.xml và nhập các cấu hình



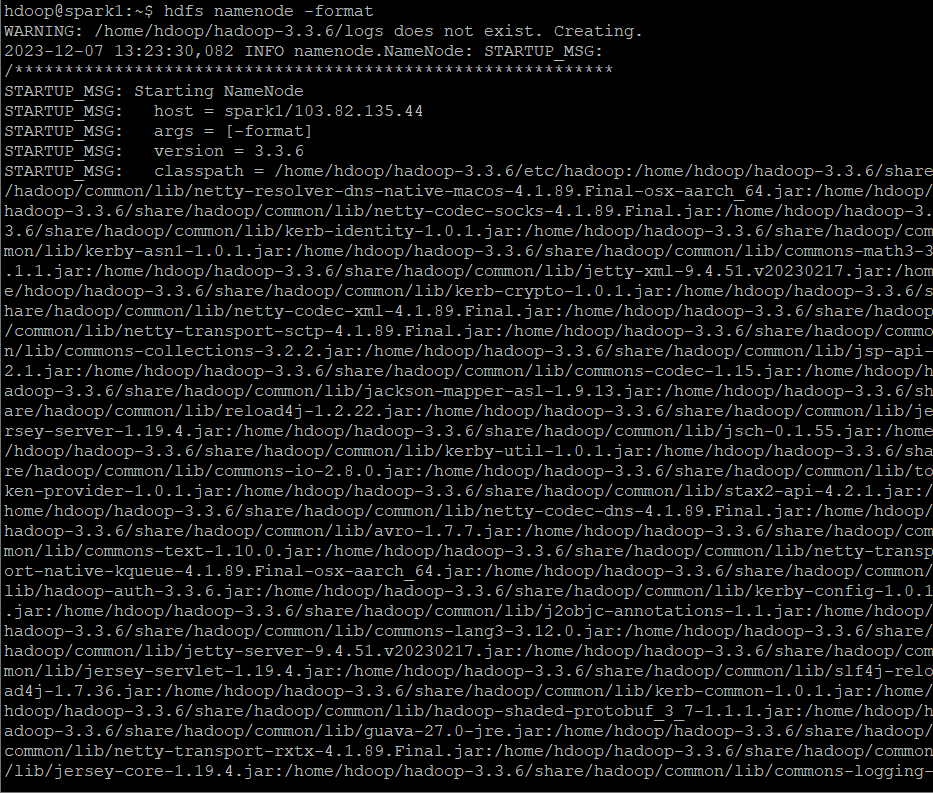


B14: vi yarn-site.xml

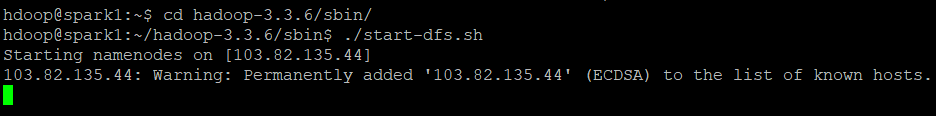




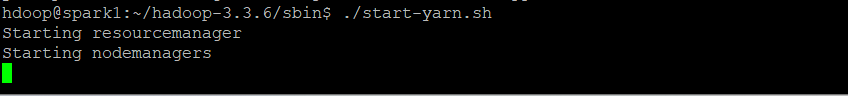
B15: hdfs namenode -format



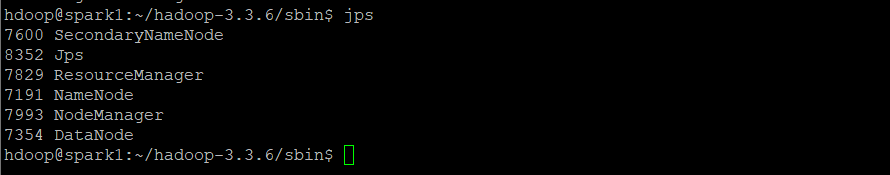
B16: ./start-dfs.sh



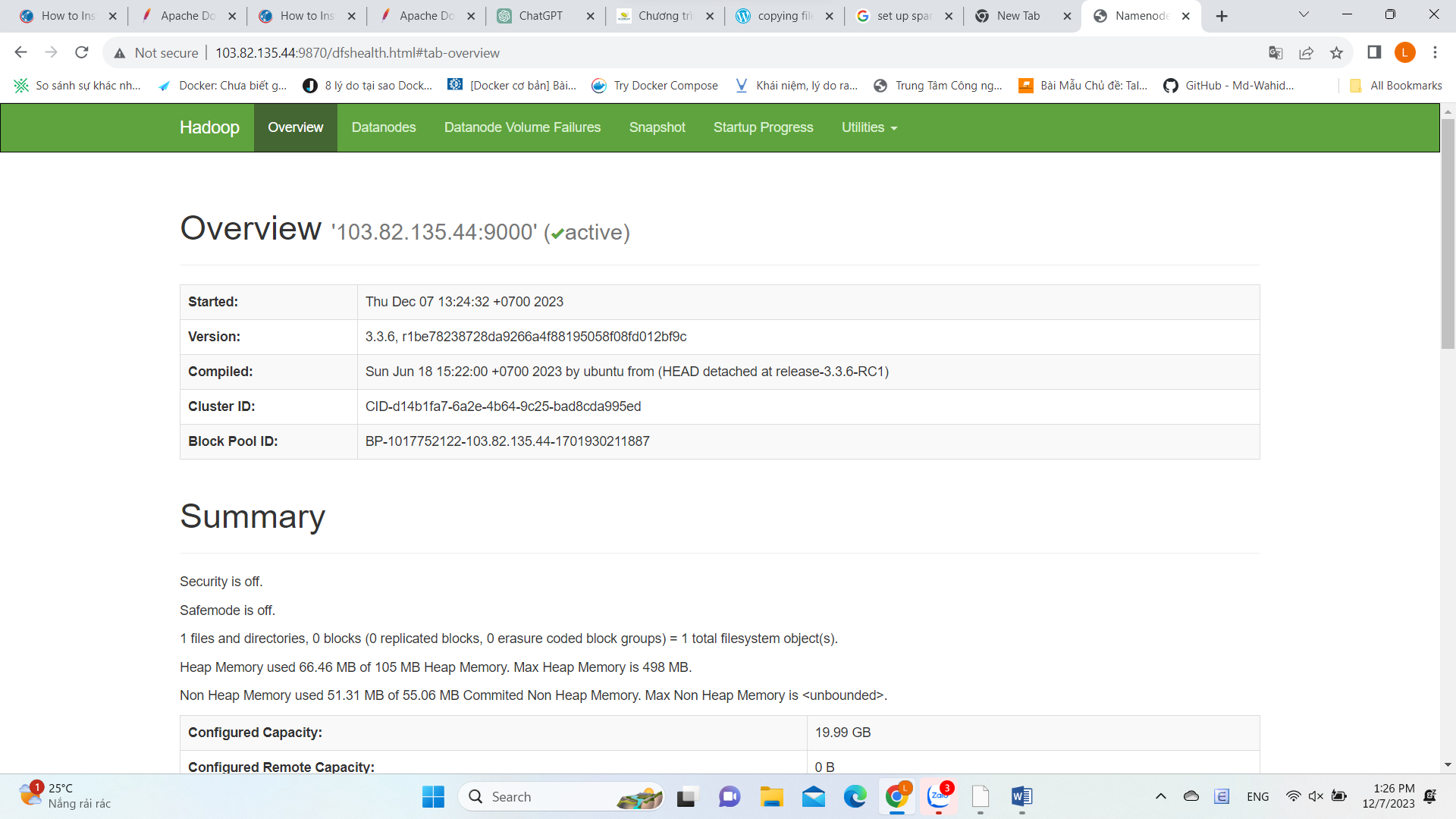
B17: ./start-yarn.sh



B18: jps

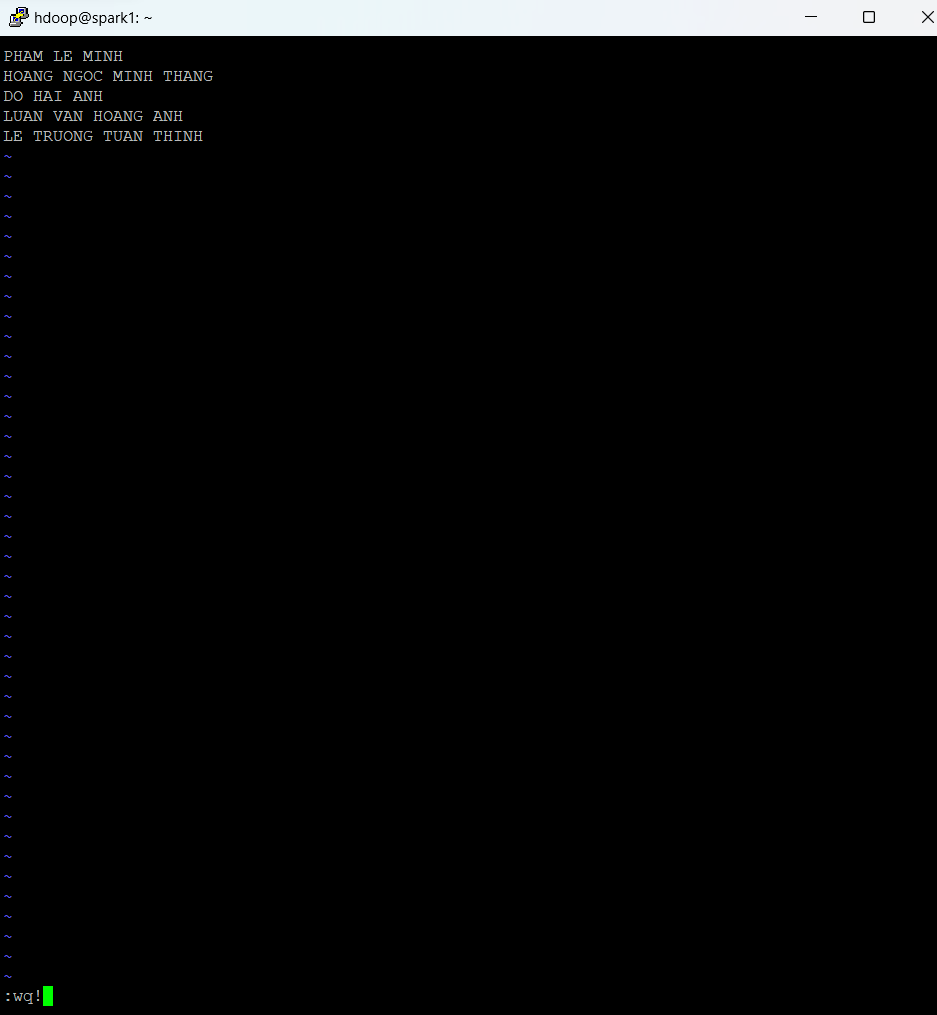


B19: kiểm tra ip 103.82.135.44:9870



B20: Tạo một file txt và nhập nội dung bất kỳ



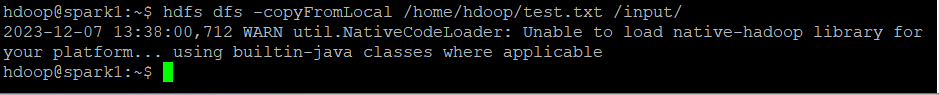


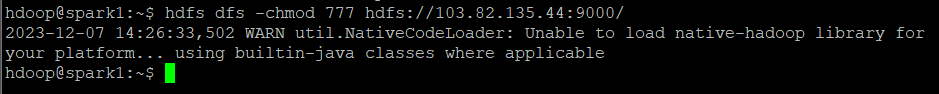
B21: hdfs dfs –mkdir /input/

hdfs dfs –copyFromLocal /home/hdoop/test.txt /input/

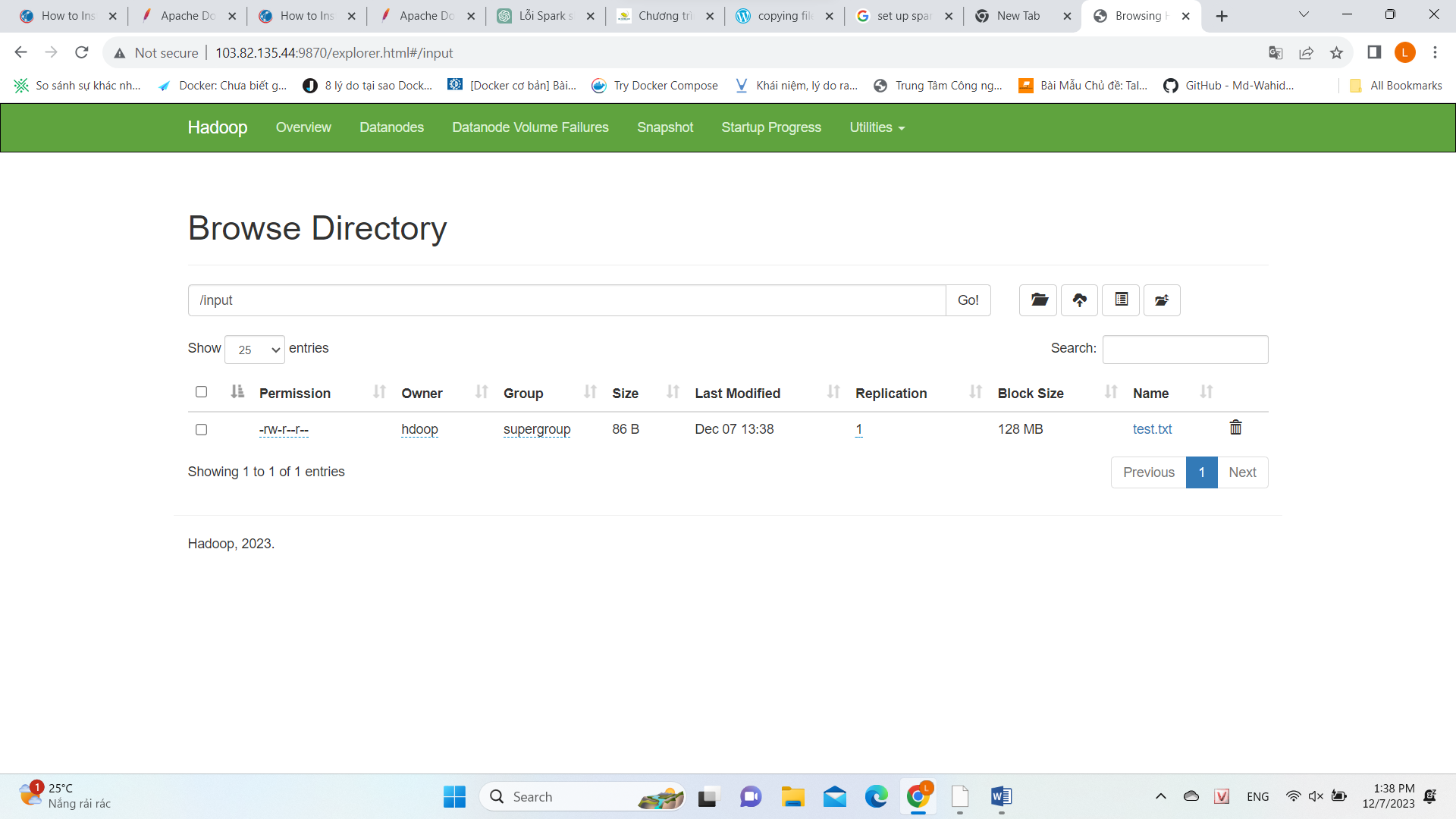
hdfs dfs –chmod 777 hdfs://103.82.135.44:9000/







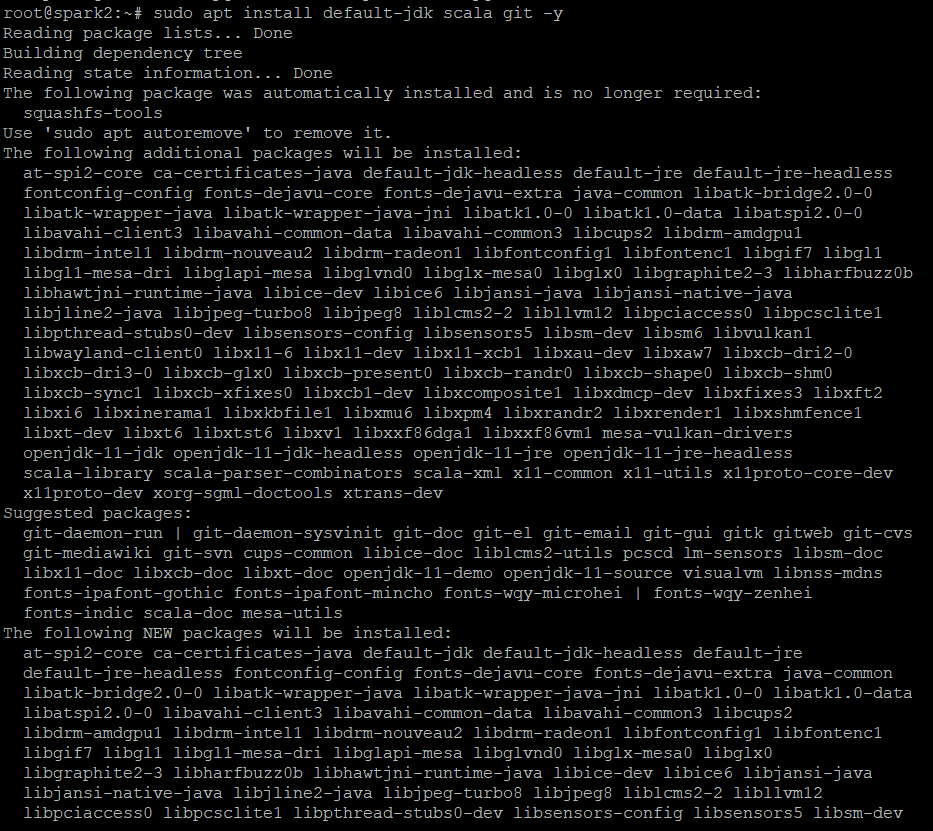
B22: Kiểm tra file txt



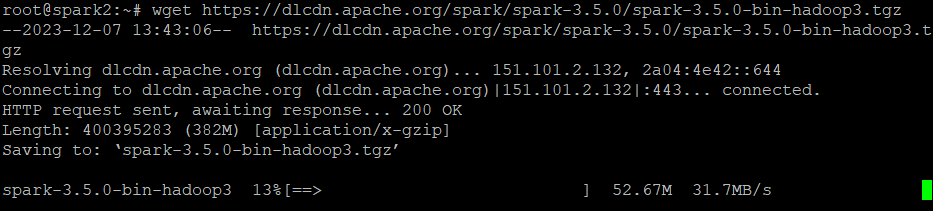
**Máy 2:**

Cài Spark master còn máy 3,4,5 sẽ làm spark slave

B1: sudo apt install default-jdk scala git -y

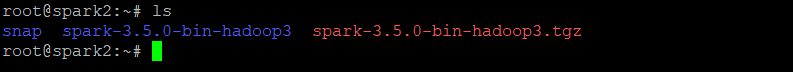


B2: wget gói mới nhất của spark



B3: tar xvf spark-\*





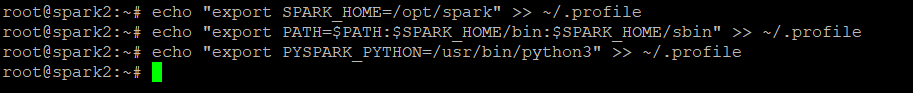
B4: sudo mv spark-3-5-0-bin-hadoop3 /opt/spark



B5: echo "export SPARK\_HOME=/opt/spark" >> ~/.profile

echo "export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin" >> ~/.profile

echo "export PYSPARK\_PYTHON=/usr/bin/python3" >> ~/.profile



B6: vi .profile kiểm tra

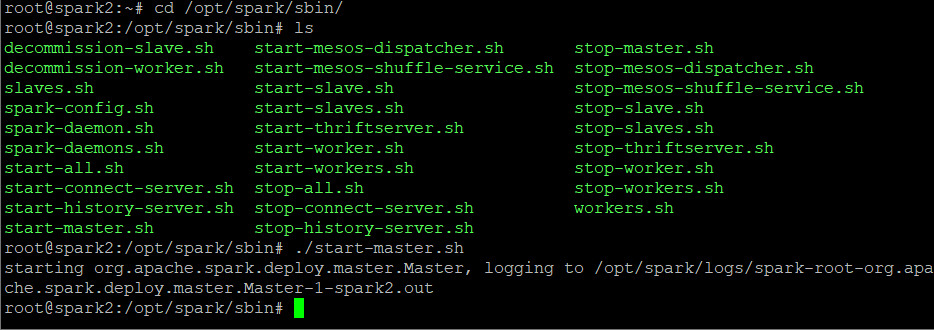




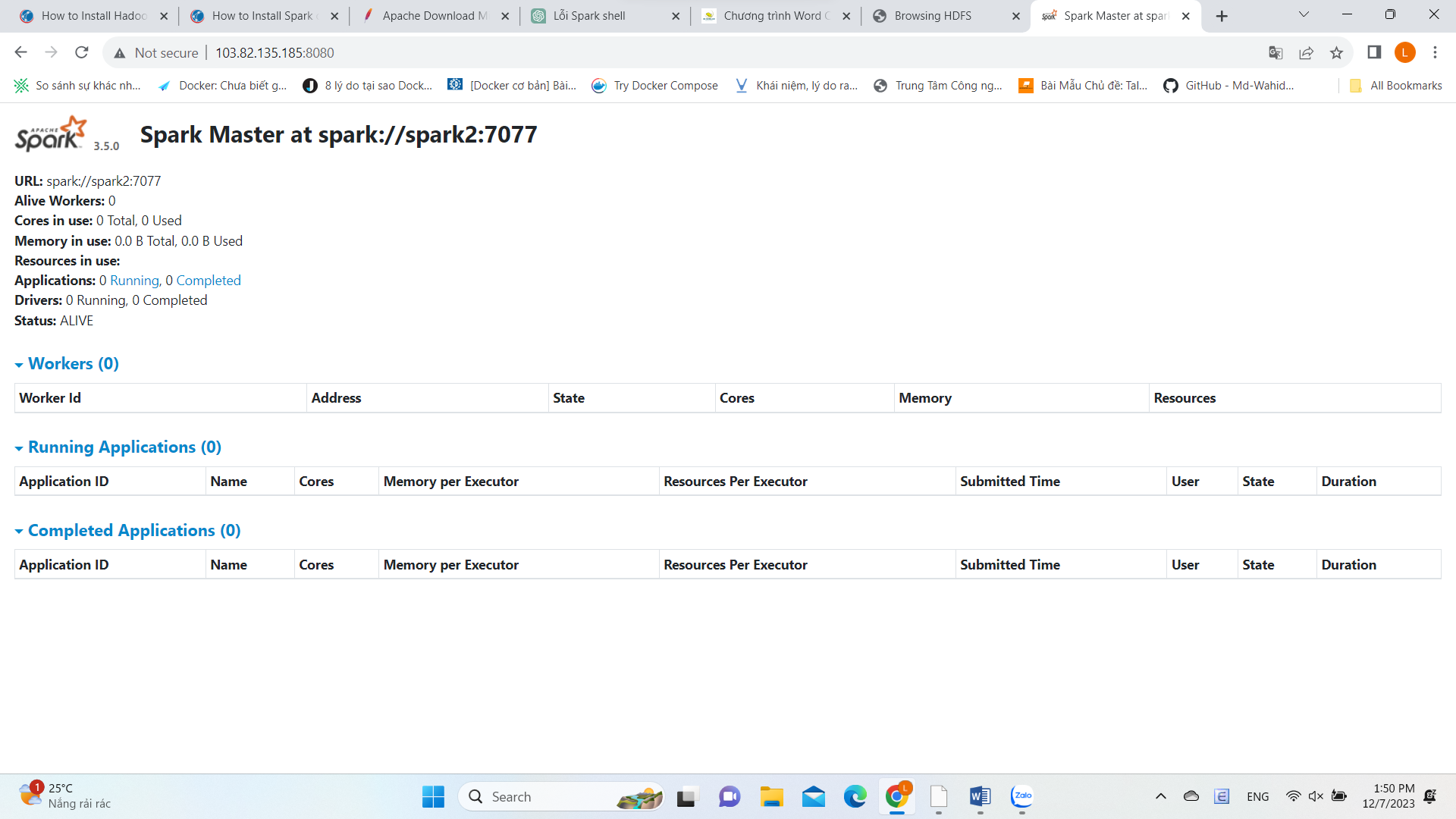
B7: source ~/.profile

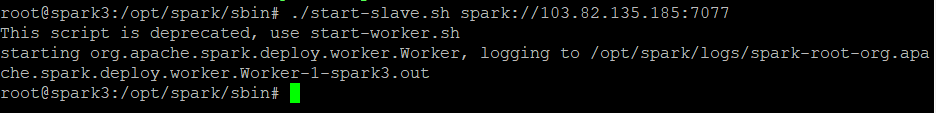


B8: ./start-master.sh

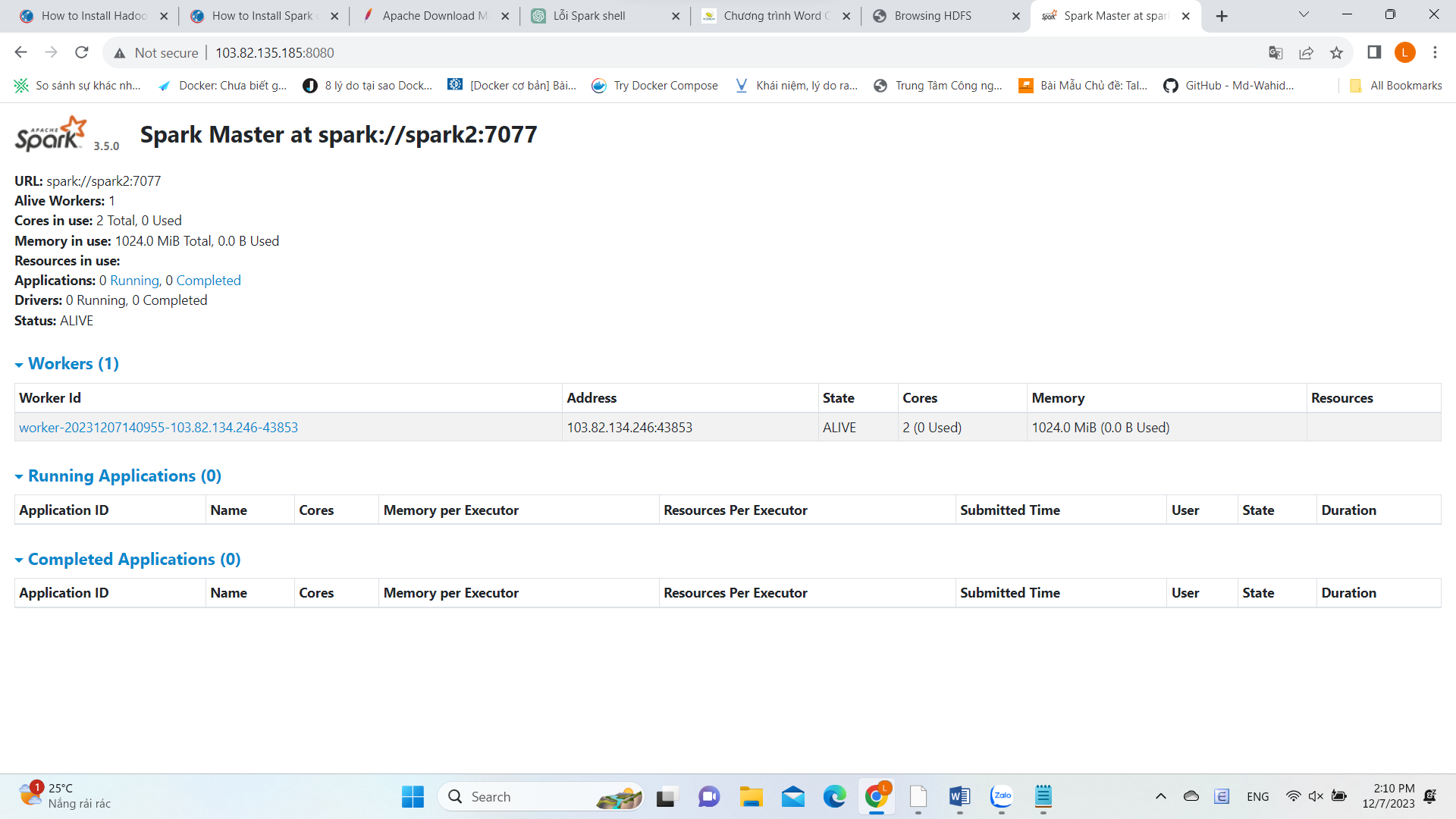


B9: Nhập ip 103.82.135.185:7077 (máy spark master)

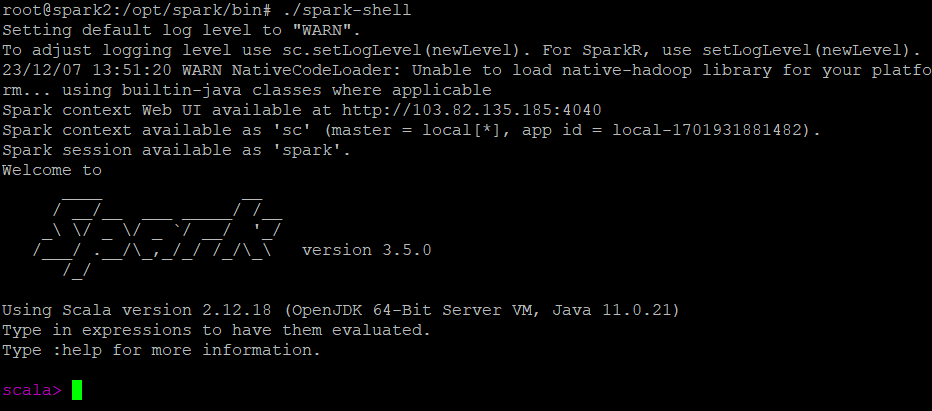


B10: Máy slave cấu hình như máy master nhưng chỉ thay đổi ở phần start ( ./start-slave.sh spark://103.82.135.185:7077  


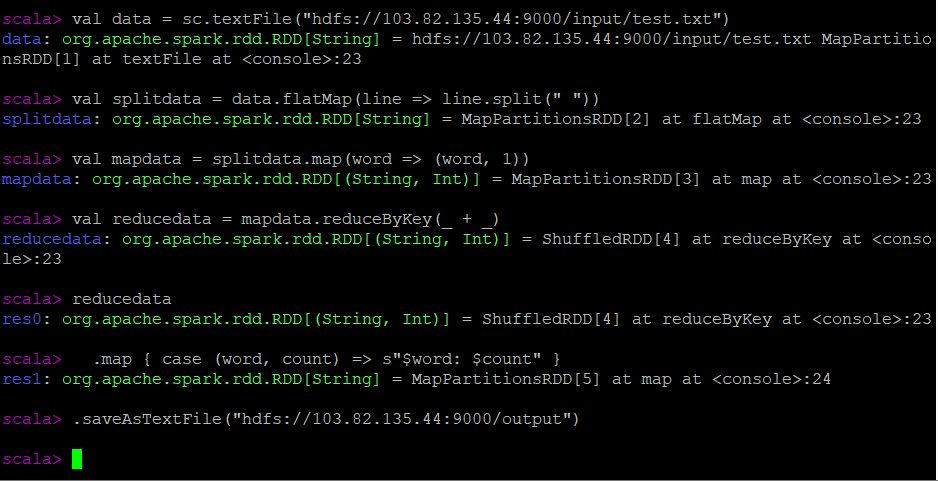
B11: Kiểm tra lại xem slave kết nối tới cụm master chưa



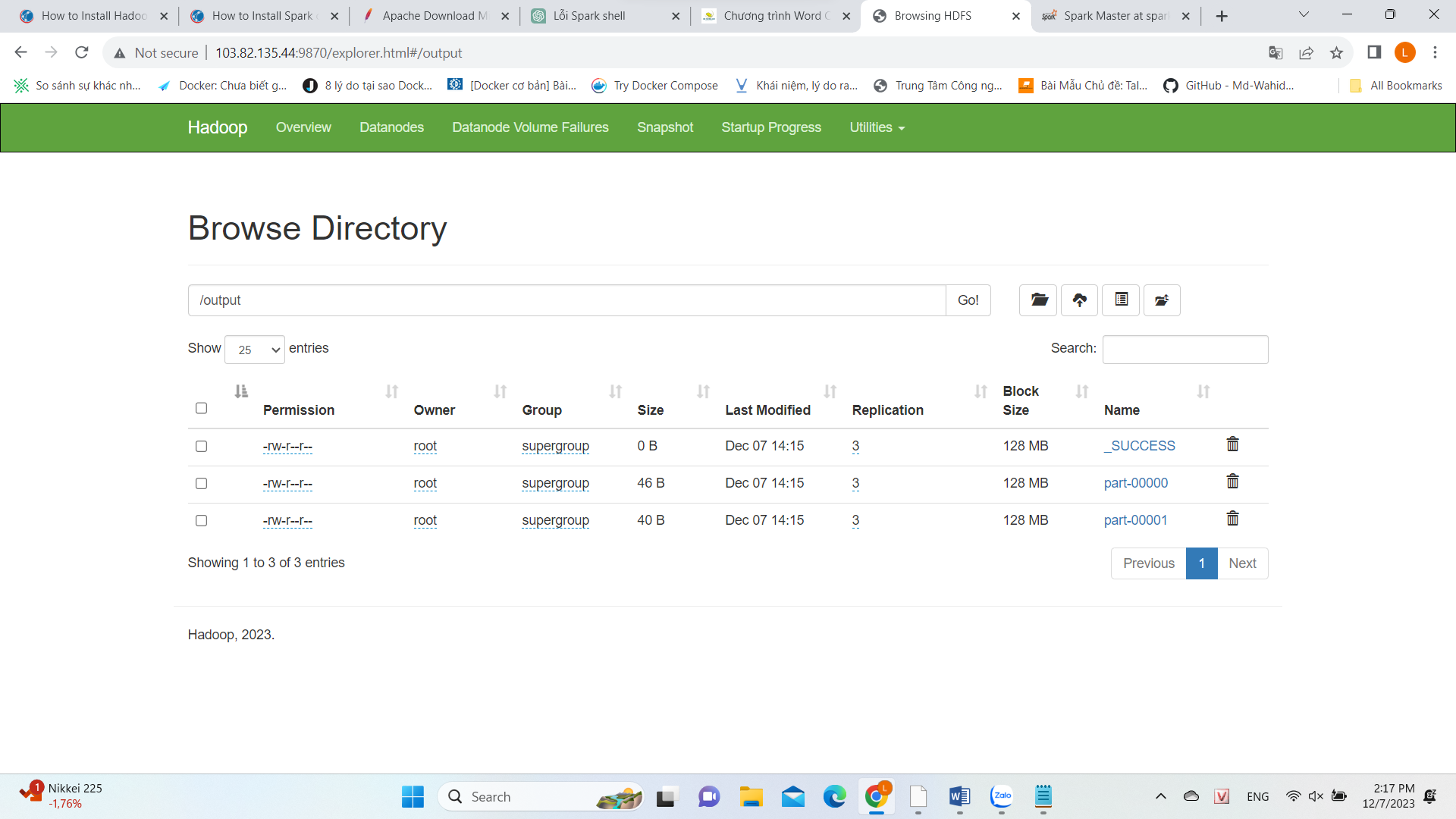
B12: Sử dụng scala: ./spark-shell (Cách 1)



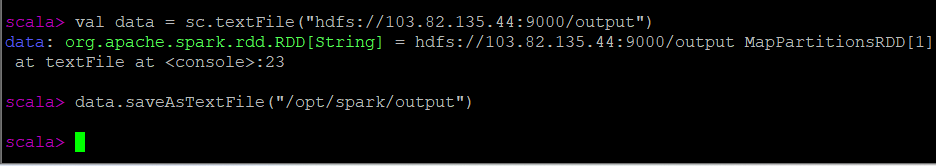
B13: Nhập các lệnh trong hình



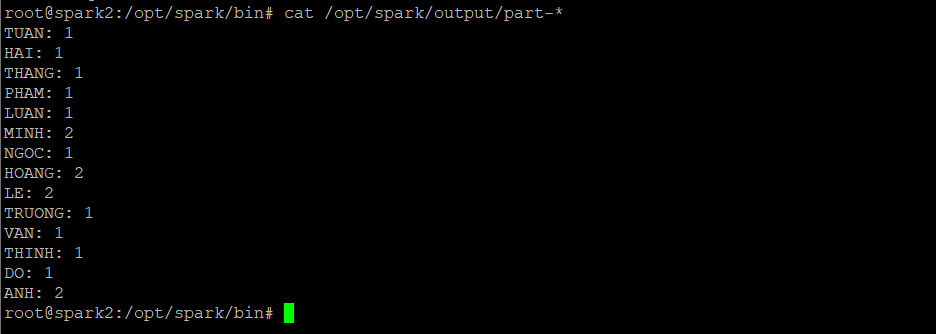
B14: Kiểm tra lại hdfs có file output chưa



B15: Nhập lệnh sau để lấy về từ hdfs sang máy spark master



B16: cat /output/part-\*



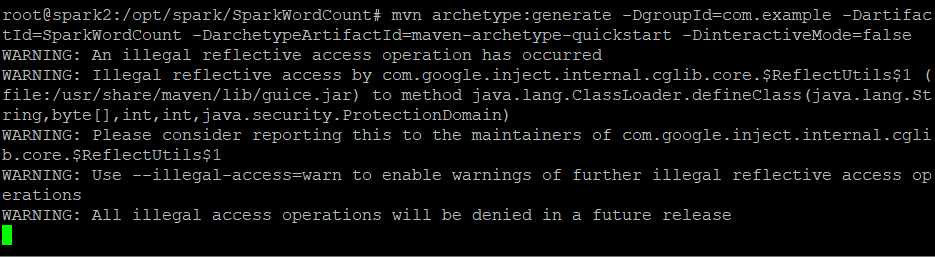
B17: Sử dụng spark-submit (Cách 2)

mkdir SparkWordCount

cd SparkWordCount

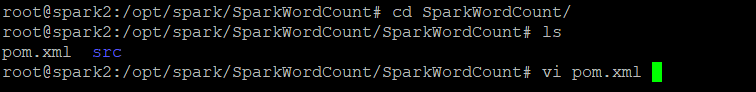


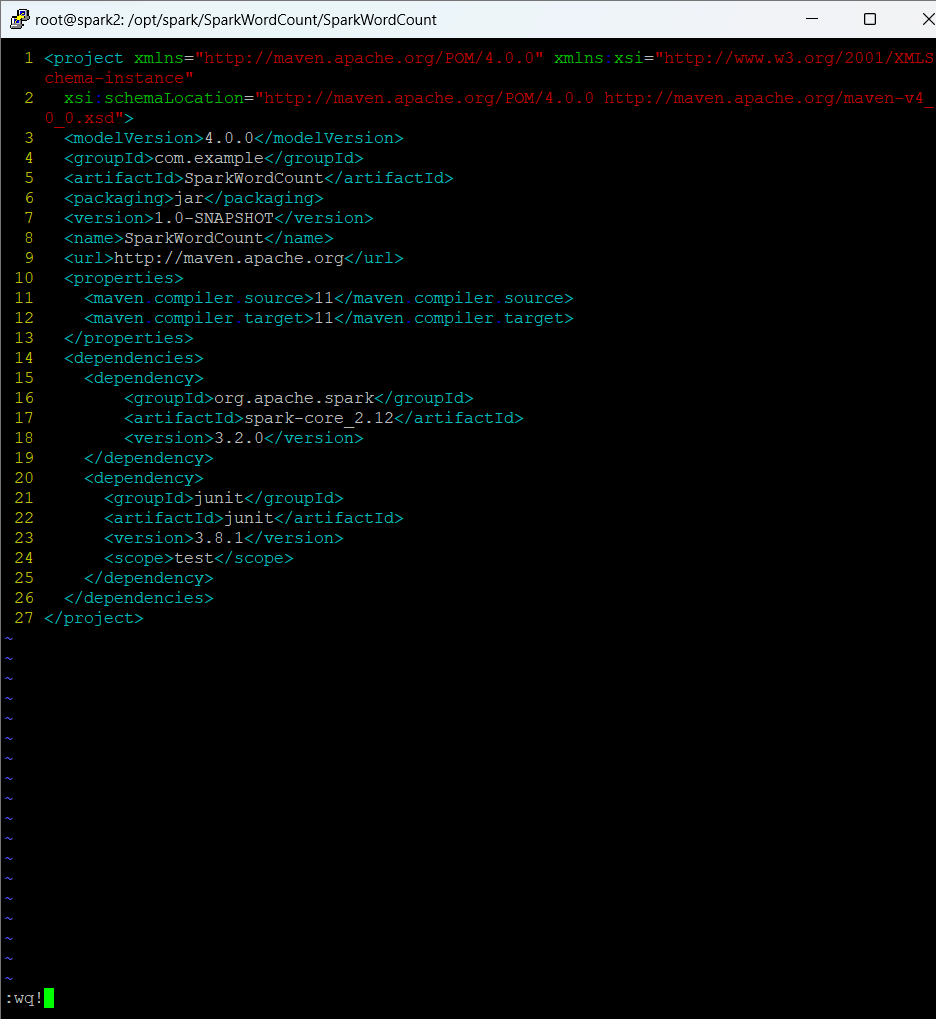
B18: mvn archetype:generate -DgroupId=com.example -DartifactId=SparkWordCount -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false



B19: cd SparkWordCount

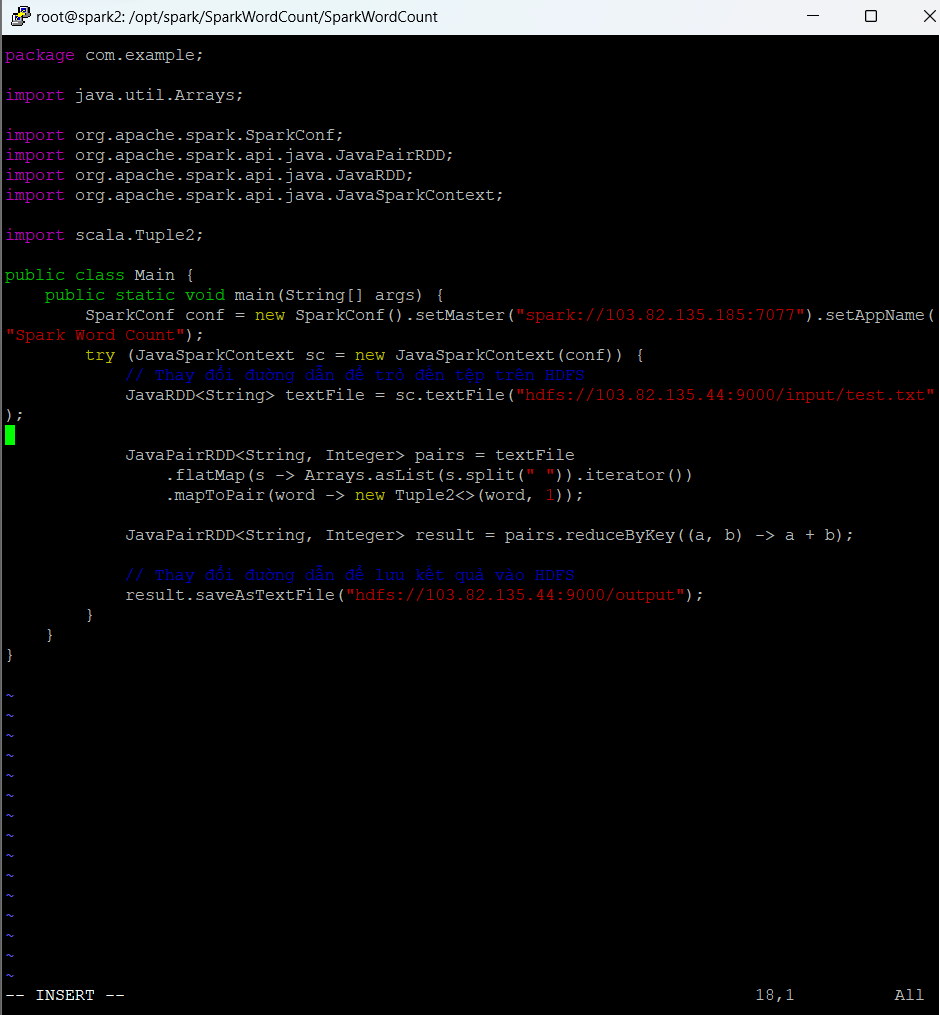
vi pom.xml và nhập các lệnh





B20: vi src/main/java/com/example/Main.java và nhập các lệnh

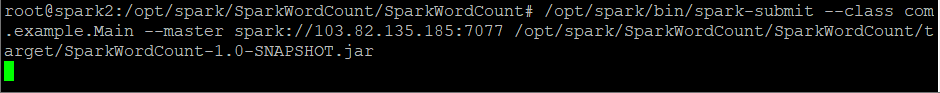




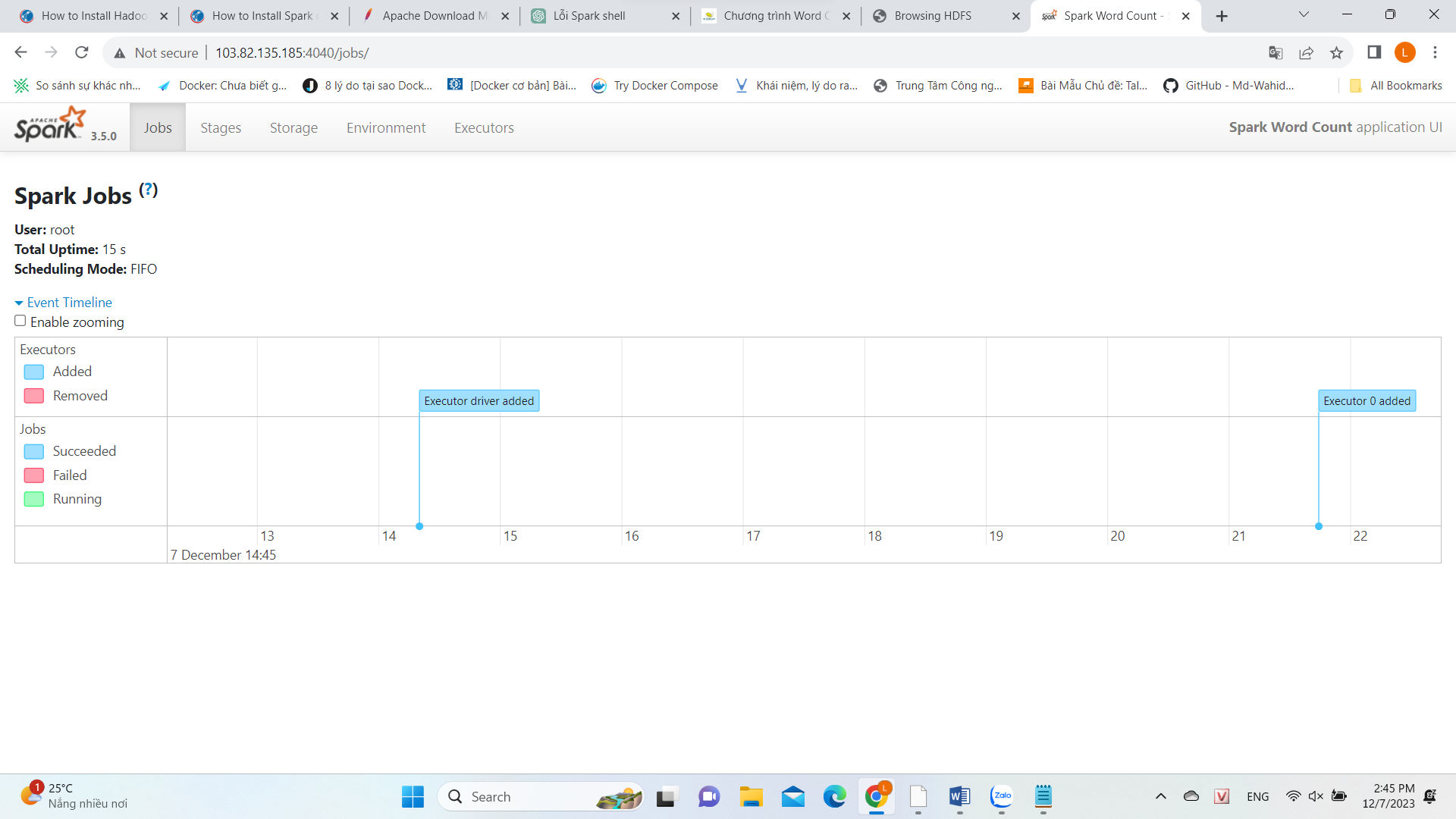
B21: mvn clean package

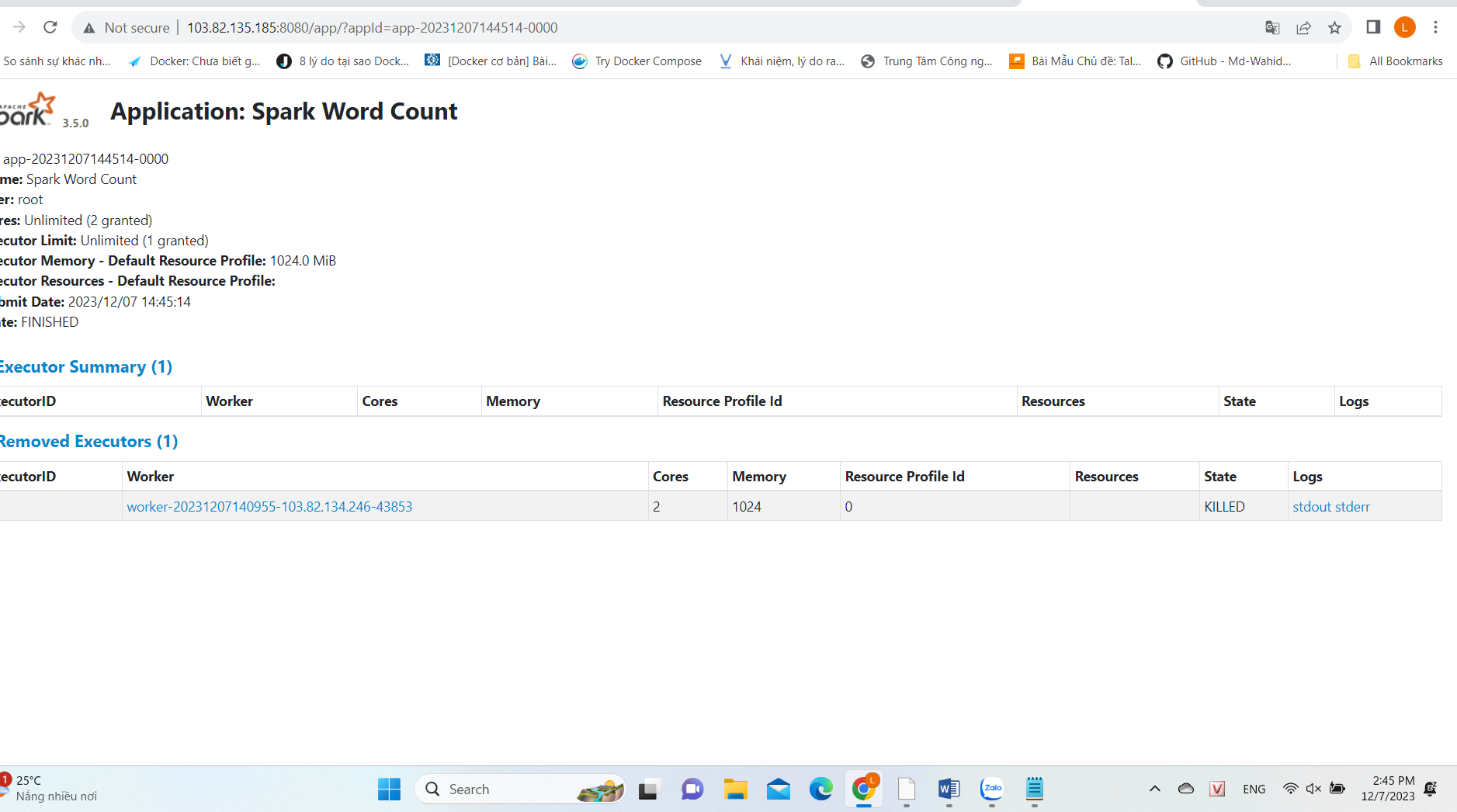


B22: /opt/spark/bin/spark-submit --class com.example.Main --master spark://103.82.135.185:7077 /opt/spark/SparkWordCount/SparkWordCount/target/SparkWordCount-1.0-SNAPSHOT.jar

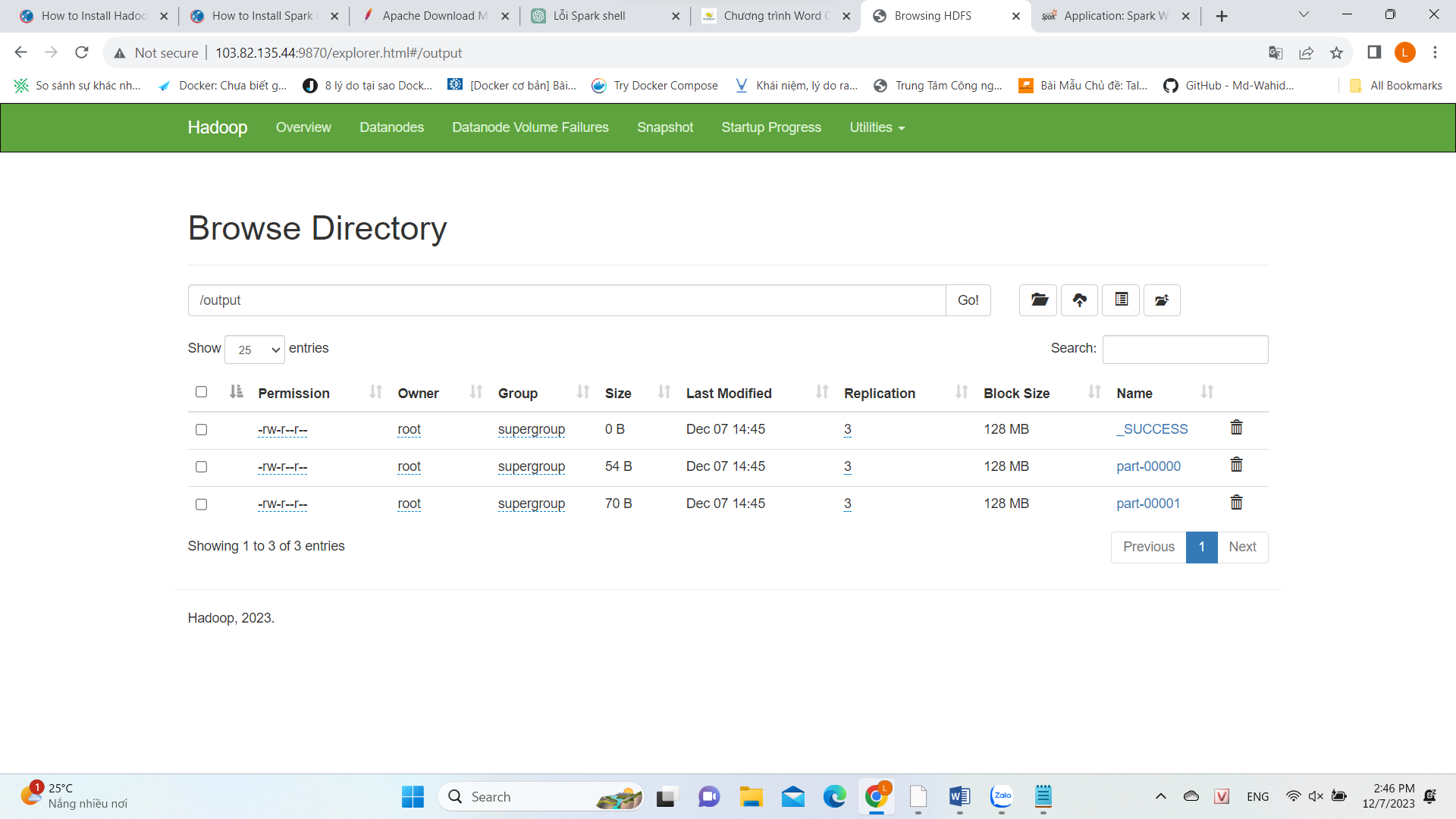


B23: Kiểm tra trên ip spark master sự phân tán

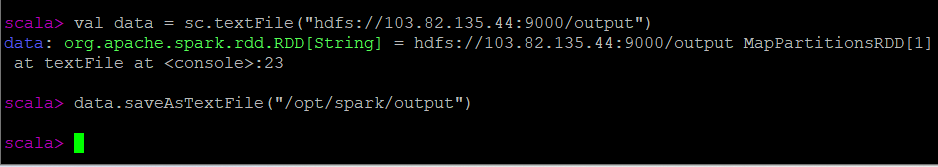




B24: Kiểm tra có file output/part-\* chưa



B25: Lấy về từ hdfs cho máy spark master



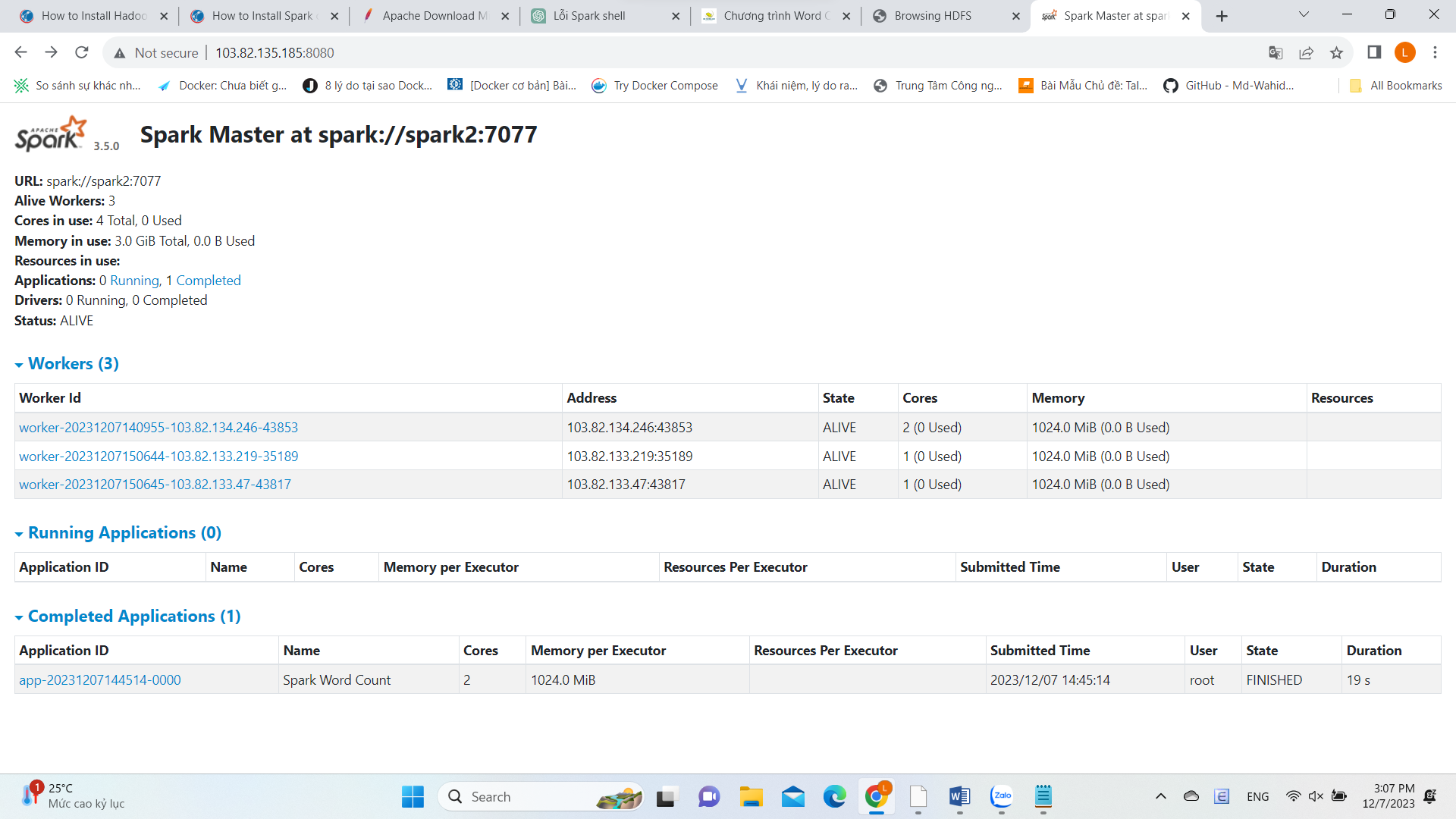
B26: Kiểm tra

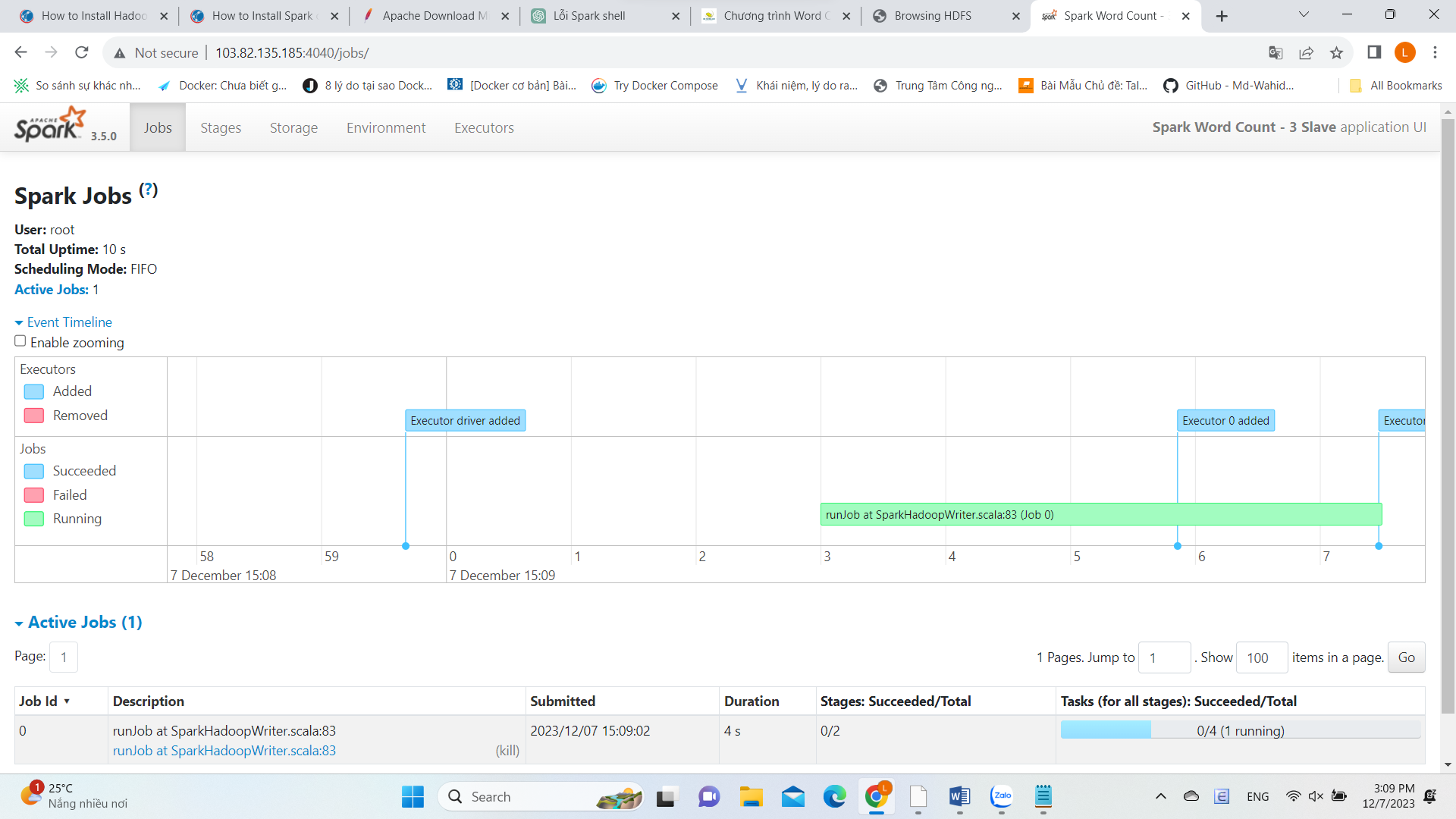


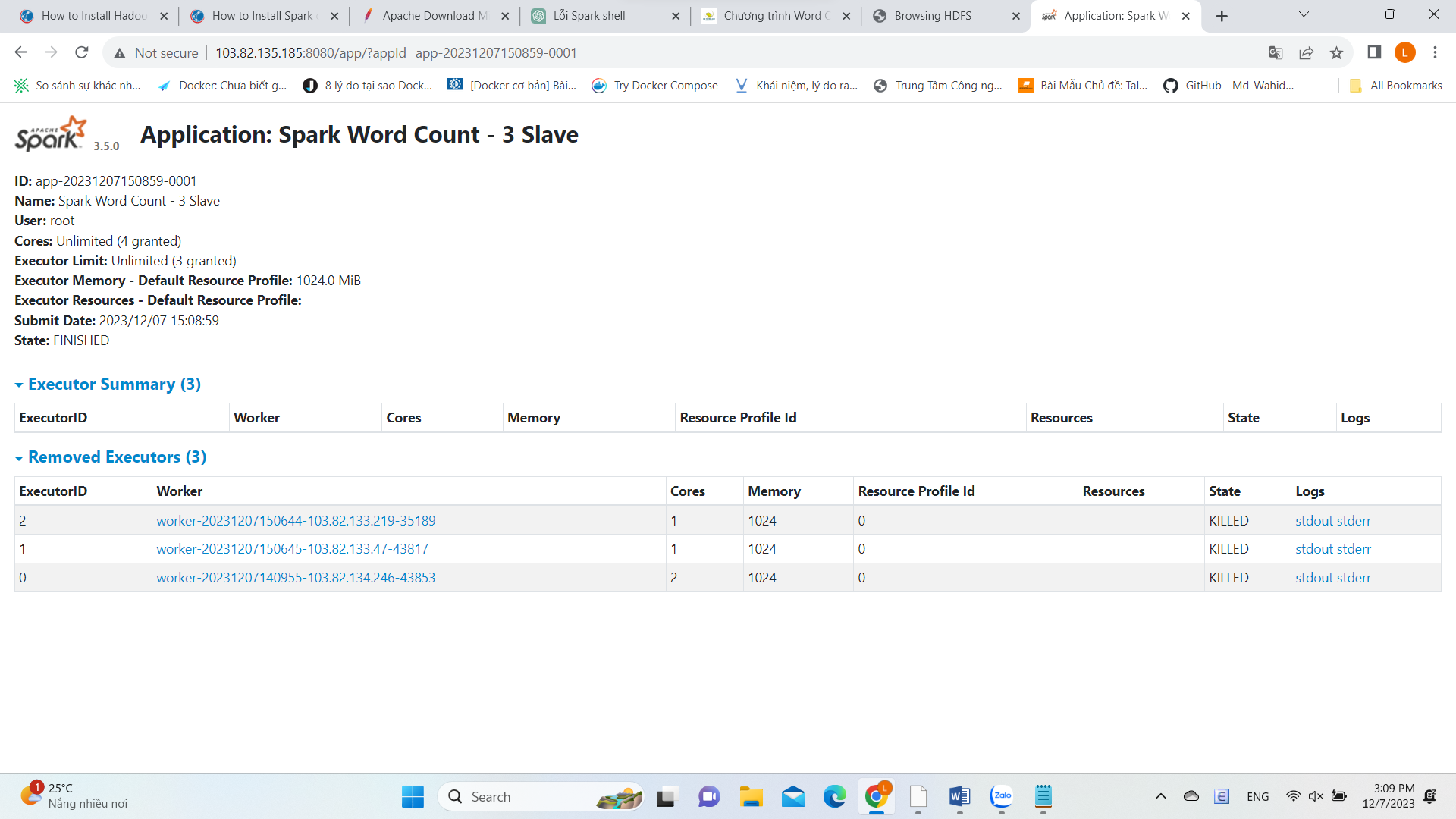
B27: nếu muốn xóa file output trên hdfs

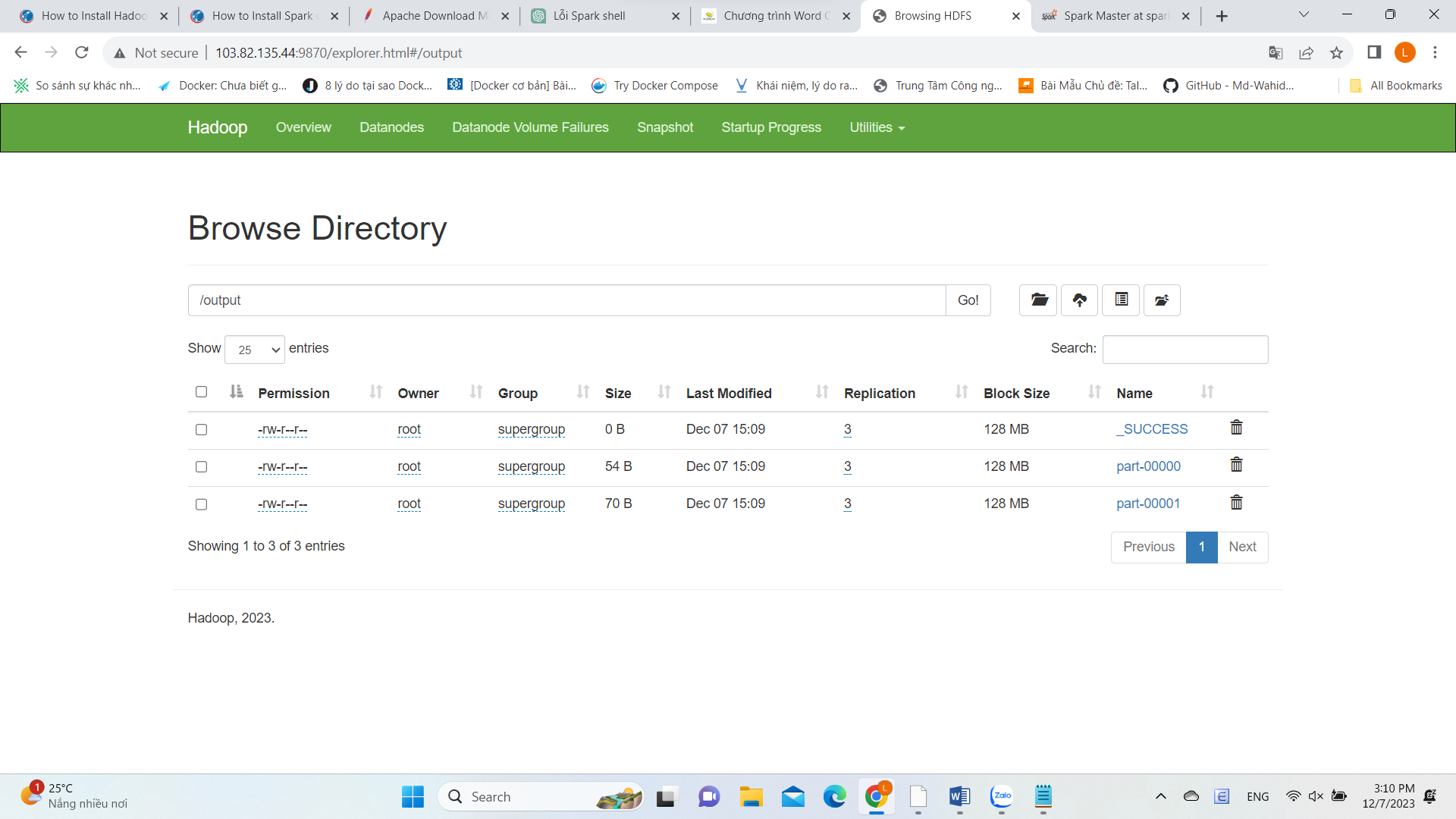


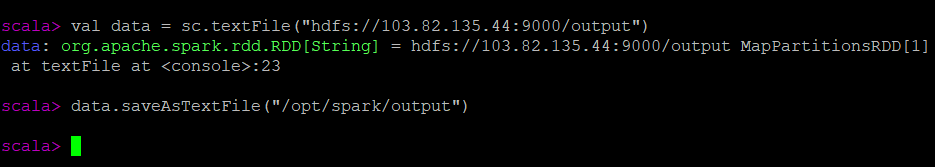
B28: Test xem thêm 2 máy spark slave còn lại và làm tương tự spark-submit







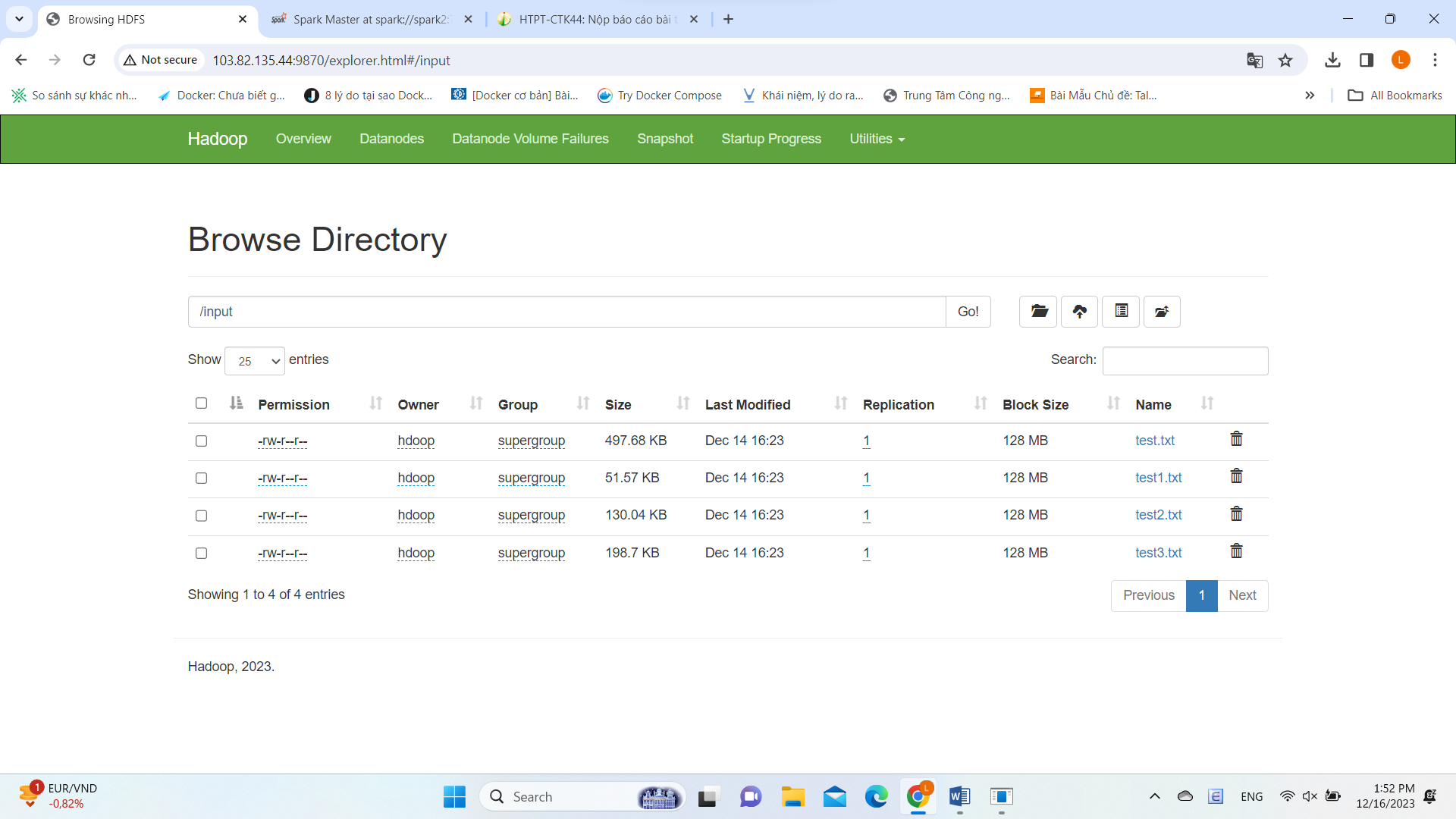




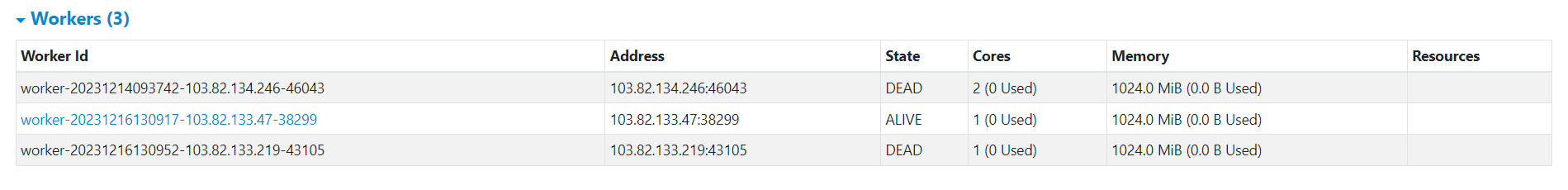


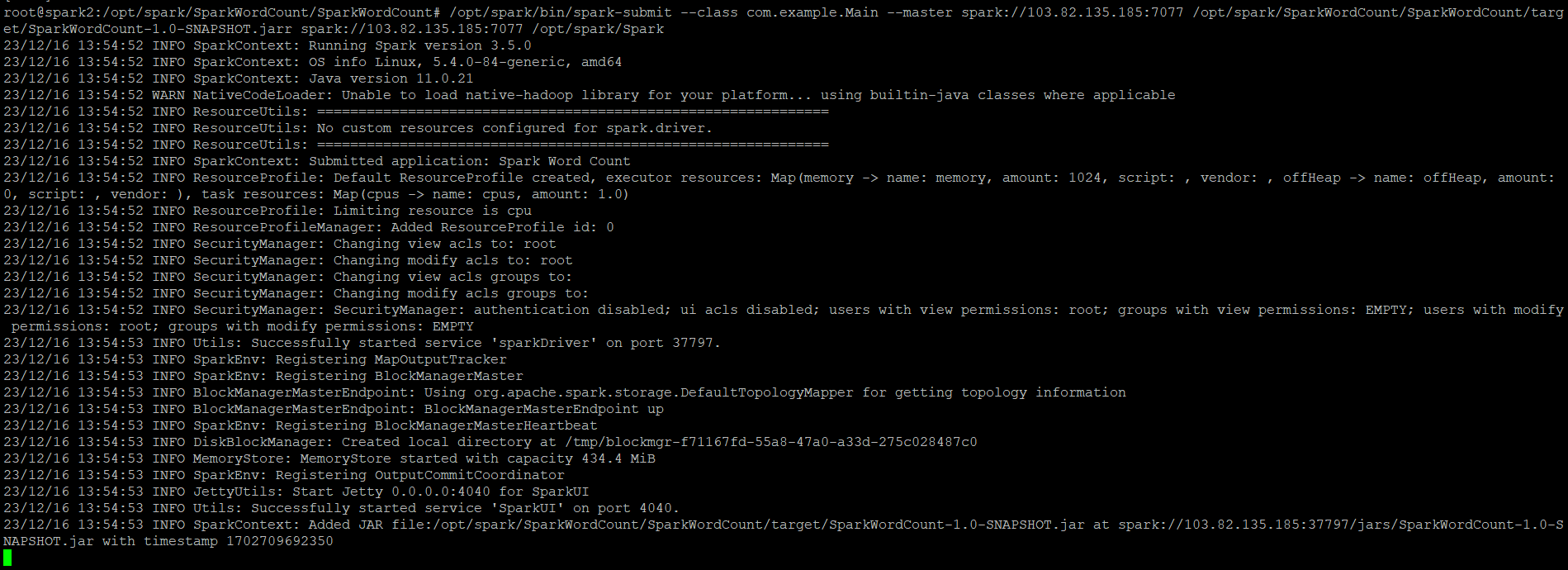
Ở trên nhóm chúng em đã cấu hình và thực hiện chương trình word count đếm file text đơn giản, tiếp theo chúng em sẽ làm rõ sự khác biệt thời gian xử lý giữa một máy slave và nhiều máy slave:

Đầu tiên tạo nhiều file txt có chứa nhiều nội dung

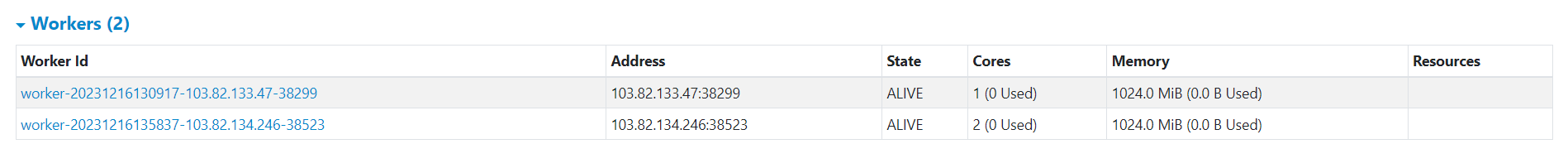


Sau đó ta stop-slave 2 máy slave rồi chạy spark-submit với 1 máy slave



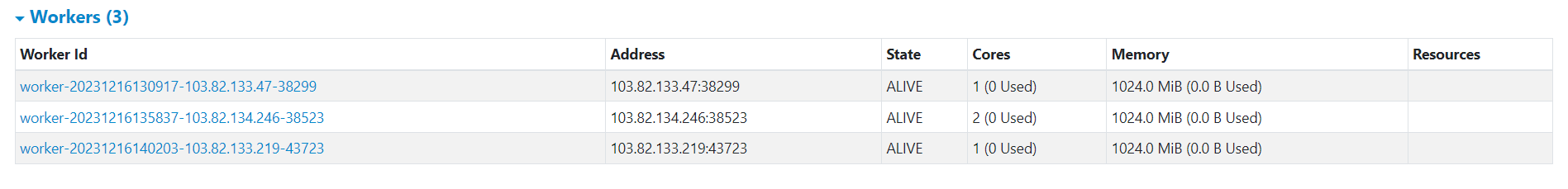


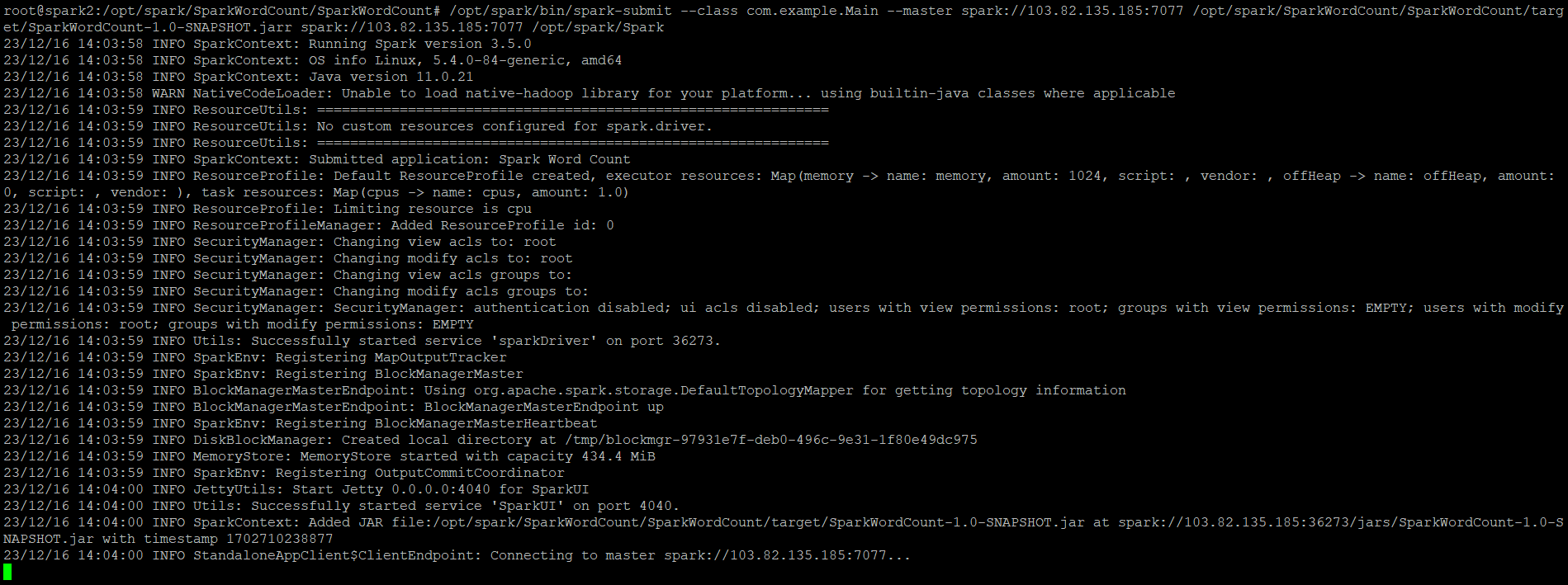
Sau đó ta start-slave thêm 1 máy slave và chạy spark-submit với 2 máy slave



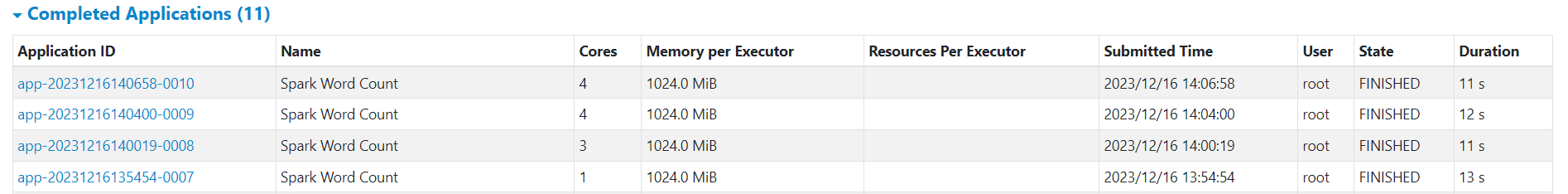


Sau đó ta start-slave thêm 1 máy slave và chạy spark-submit với 3 máy slave





Và đây là kết quả chạy giữa một máy, hai máy và 3 máy slave



Ở đây nhóm chúng em chạy 2 lần spark-submit với 3 máy slave do lần 1 bị treo mạng và sau đó chúng em thử lần 2 thì thời gian nhanh hơn một giây so với lần 1

Ta sẽ thấy thời gian chạy chương trình Word Count sẽ khác nhau giữa 1 máy slave chạy với 2 máy slave và 3 máy slave khi chạy, tuy nhiên do vấn đề mạng sẽ làm nó khó nhìn sự chênh lệch

Và đây là kết quả chương trình Word Count

