**Faculty of Engineering**

**Assignment 1- Semester 7: May 2025**

**Module Name: Cloud Computing Module**

**Number: EC7205**

**Assignment Title: Large-Scale Data Analysis Using MapReduce**

**Team Members:**

1. Dissanayaka R.P.L.M EG/2020/3909
2. Dissanayake D.K.R.C.K EG/2020/3910
3. Balasooriya B.A.L.M EG/2020/3838

**Deadline: 07th June 2025**

**Contents**

[1 Objective 4](#_Toc200204472)

[2 Dataset Selection 4](#_Toc200204473)

[2.1 Dataset Description 4](#_Toc200204474)

[2.1.1 Dataset Columns 4](#_Toc200204475)

[2.2 Dataset Justification 5](#_Toc200204476)

[3 Environment Setup 6](#_Toc200204477)

[3.1 WSL & Ubuntu 6](#_Toc200204478)

[3.2 SSH Setup 7](#_Toc200204479)

[3.3 Install Java jdk 8 7](#_Toc200204480)

[3.4 Hadoop Installation 8](#_Toc200204481)

[3.5 Configuration Files 9](#_Toc200204482)

[3.6 Environment Variables in ~/.bashrc 9](#_Toc200204483)

[3.7 Start Hadoop 10](#_Toc200204484)

[4 MapReduce Task Implementation 11](#_Toc200204485)

[4.1 Task Description 11](#_Toc200204486)

[4.2 Algorithm Design 11](#_Toc200204487)

[4.3 Source Code Summary 12](#_Toc200204488)

[5 Execution and Testing 14](#_Toc200204489)

[5.1 Upload Dataset to HDFS 14](#_Toc200204490)

[5.2 Run MapReduce Job 14](#_Toc200204491)

[5.3 Output Sample 15](#_Toc200204492)

[6 Results and Interpretation 16](#_Toc200204493)

[7 Hadoop Web Interfaces 17](#_Toc200204494)

**List of Figures**

[Figure 2.1 : About the Wine review dataset from Keggle 5](#_Toc200218477)

[Figure 3.1 : WSL Installation via PowerShell 6](#_Toc200218478)

[Figure 3.2 : Ubuntu package update 7](#_Toc200218479)

[Figure 3.3 : Command for remove and install OpenSSH 7](#_Toc200218480)

[Figure 3.4 : Java installation command line 8](#_Toc200218481)

[Figure 3.5 : Java version check command 8](#_Toc200218482)

[Figure 3.6 : Download and exact Hadoop 8](#_Toc200218483)

[Figure 3.7 : Navigating to Hadoop Configuration File core-site.xml using VS Code 9](#_Toc200218484)

[Figure 3.8 : Open .bashrc file 9](#_Toc200218485)

[Figure 3.9 : bashrc file 9](#_Toc200218486)

[Figure 3.10 : Code for source 10](#_Toc200218487)

[Figure 3.11 : Format the HDFS NameNode 10](#_Toc200218488)

[Figure 3.12 : Start the HDFS daemons 10](#_Toc200218489)

[Figure 3.13 : Start the YARN daemons 10](#_Toc200218490)

[Figure 3.14 : Output of the jps command 11](#_Toc200218491)

[Figure 4.1 : AveragePricePerVariety.java file 12](#_Toc200218492)

[Figure 4.2 : AveragePriceReducer.java file code 12](#_Toc200218493)

[Figure 4.3 : PriceMapper.java file code 13](#_Toc200218494)

[Figure 5.1 : Creating HDFS directory and uploading a CSV file in Hadoop 14](#_Toc200218495)

[Figure 5.2 : Hadoop MapReduce job calculating average wine prices per variety 14](#_Toc200218496)

[Figure 5.3 : Command-line interface 15](#_Toc200218497)

[Figure 5.4 : Hadoop MapReduce output: Average wine prices by variety. 15](#_Toc200218498)

[Figure 6.1 : Output terminal of wine varieties and their calculated average prices 16](#_Toc200218499)

[Figure 7.1 : Hadoop web interface (NameNode) 17](#_Toc200218500)

[Figure 7.2 : Hadoop web interface (ResourceManager) 18](#_Toc200218501)

# Objective

This project aims to analyze a large wine review dataset using Hadoop MapReduce to extract meaningful insights. Specifically, it calculates the average price per wine variety, filtering out incomplete or malformed records. The solution is implemented in Java using the Hadoop MapReduce programming model.

# Dataset Selection

## Dataset Description

This project utilizes the Wine Reviews dataset from Kaggle, which contains over 130,000 wine reviews including information such as variety, price, description, region, and rating. The dataset is valuable for analysis due to its mix of structured (e.g., price, points) and unstructured (e.g., description) data. For our project, the focus is on extracting the average price per wine variety using Hadoop MapReduce.

* **Dataset Name**: Wine Reviews
* **Source**: Kaggle (<https://www.kaggle.com/datasets/zynicide/wine-reviews>)
* **Link**: Wine Reviews Dataset
* **File Format**: CSV
* **Size**: 130,000+ reviews
* **License**: CC BY-NC-SA 4.0

### Dataset Columns

* country
* description
* designation
* points
* price
* province
* region\_1
* region\_2
* variety (target group)

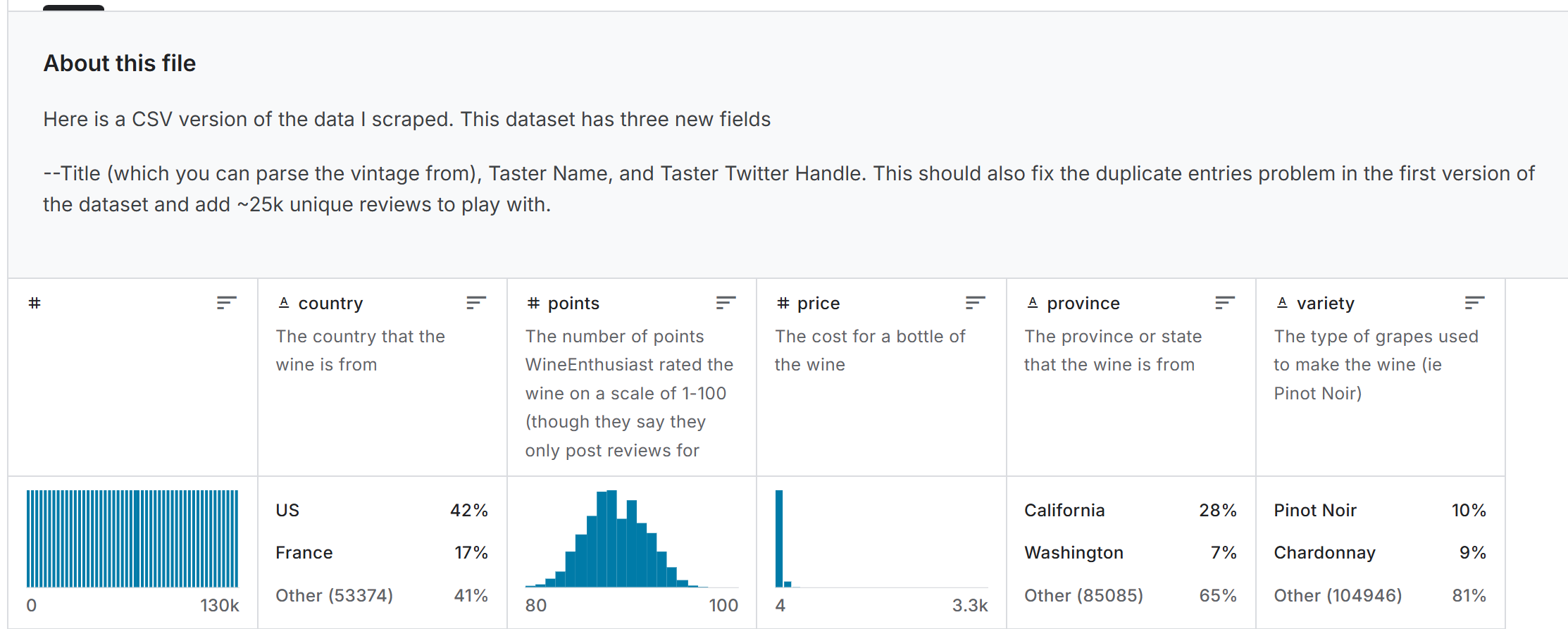


Figure . : About the Wine review dataset from Kaggle

## Dataset Justification

The Wine Reviews dataset is an ideal candidate for distributed data processing with Hadoop MapReduce due to its large volume and diverse structure. It contains a blend of categorical, numerical, and textual data, allowing us to explore a variety of analytical goals. In particular, the inclusion of wine variety and price information enables grouped analysis and aggregation operations that are naturally parallelizable. Processing this data with MapReduce not only improves scalability but also demonstrates how Hadoop can efficiently handle real-world datasets that include missing or inconsistent entries. By computing the average price for each wine variety, this project delivers practical insights that could be valuable to wine retailers, sommeliers, and consumers alike, especially in identifying pricing patterns across different types of wines.

# Environment Setup

Setting up a Hadoop environment for Java-based MapReduce involves configuring several components including WSL (Windows Subsystem for Linux), Java, SSH, and Hadoop binaries. Below is a detailed breakdown of each step required to get the project running smoothly.

## WSL & Ubuntu

Windows Subsystem for Linux (WSL) allows us to run a Linux environment directly on Windows. For this project, we use Ubuntu as the Linux distribution under WSL. This setup enables seamless integration between Linux-native tools and Hadoop. Installation command has shown on below Figure 3.1.

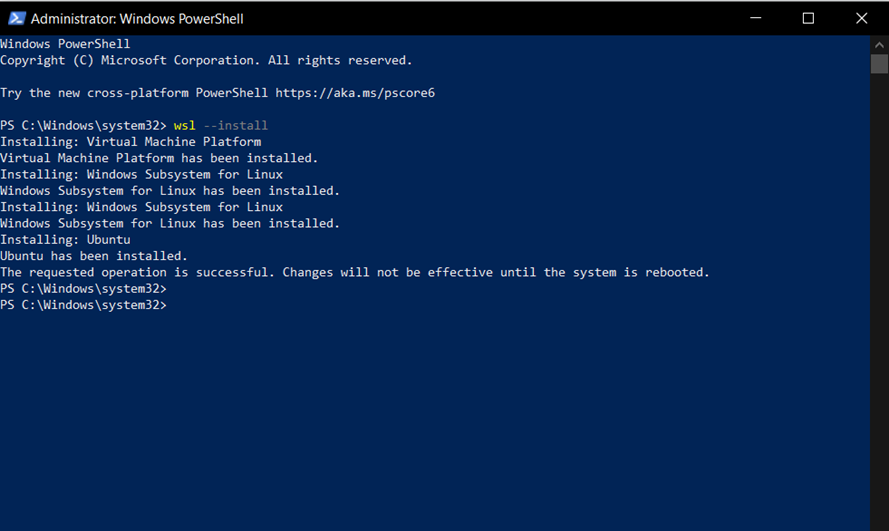


Figure . : WSL Installation via PowerShell

After rebooting, Ubuntu is opened and update packages. This ensures machine have the latest system updates and dependencies. These commands have shown by Figure 3.2.

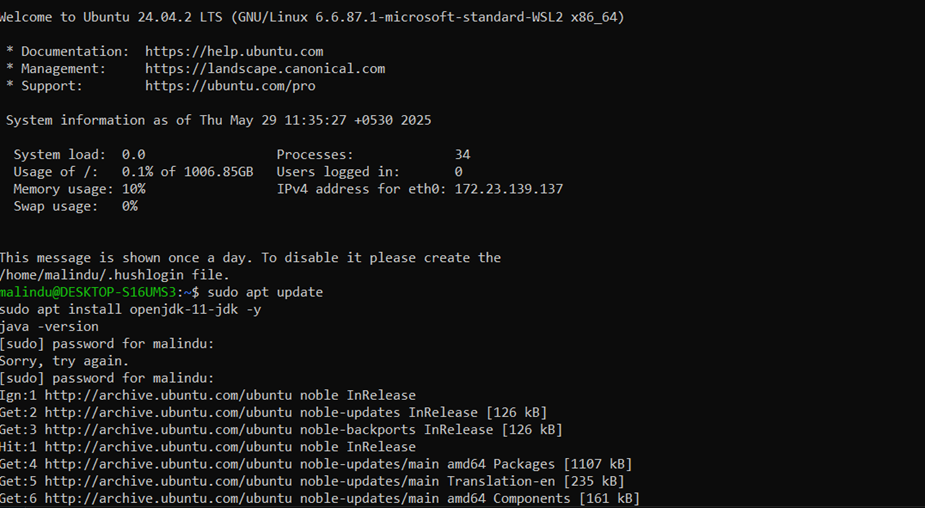


Figure . : Ubuntu package update

## SSH Setup

Hadoop uses SSH for managing cluster nodes even in a pseudo distributed single-node setup. It's essential that passwordless SSH is set up so Hadoop daemons can communicate effectively during start-up and task execution.

Before installing a new version of OpenSSH, it’s a good idea to remove any existing version that might already be installed. This helps prevent potential issues caused by version conflicts or mismatched components.

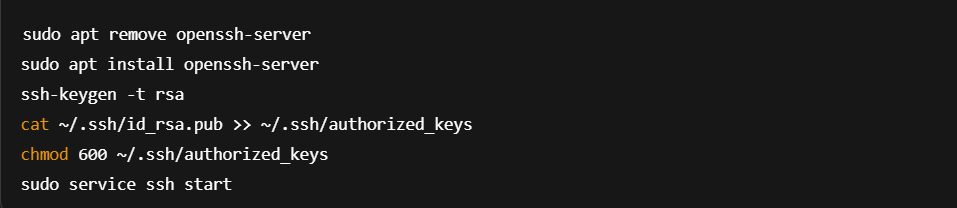


Figure . : Command for remove and install OpenSSH

## Install Java jdk 8

Hadoop requires Java to run all core components, including NameNode, DataNode, and YARN services. OpenJDK 8 has used as it is stable and compatible with Hadoop 3.x.

Install Java using following command has mentioned by Figure 3.4

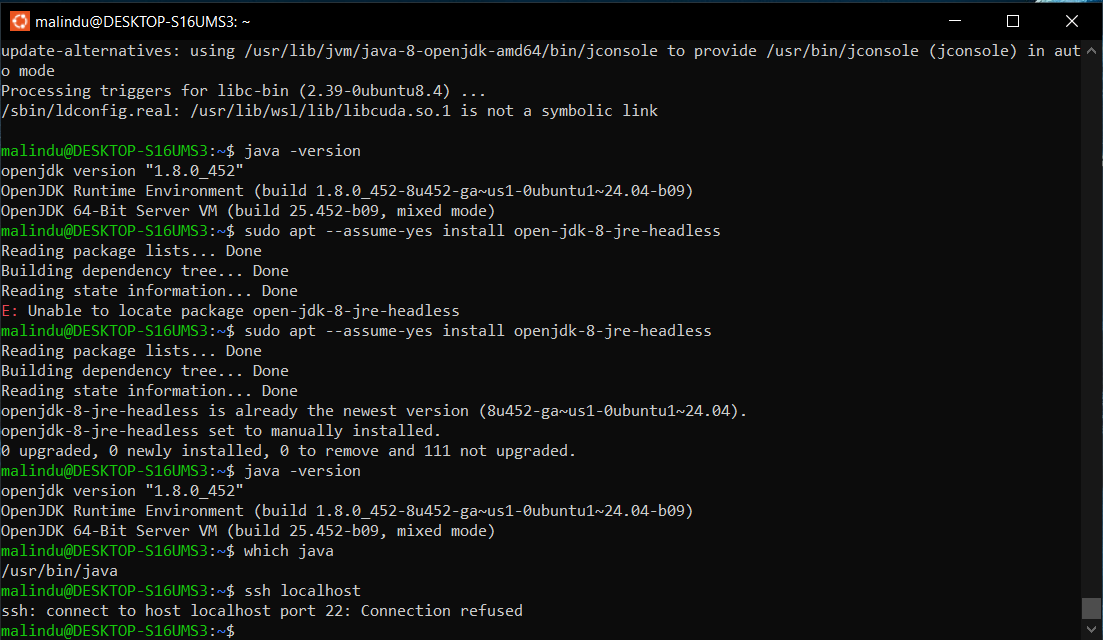


Figure . : Java installation command line

After installation verify the installation with using java -version command

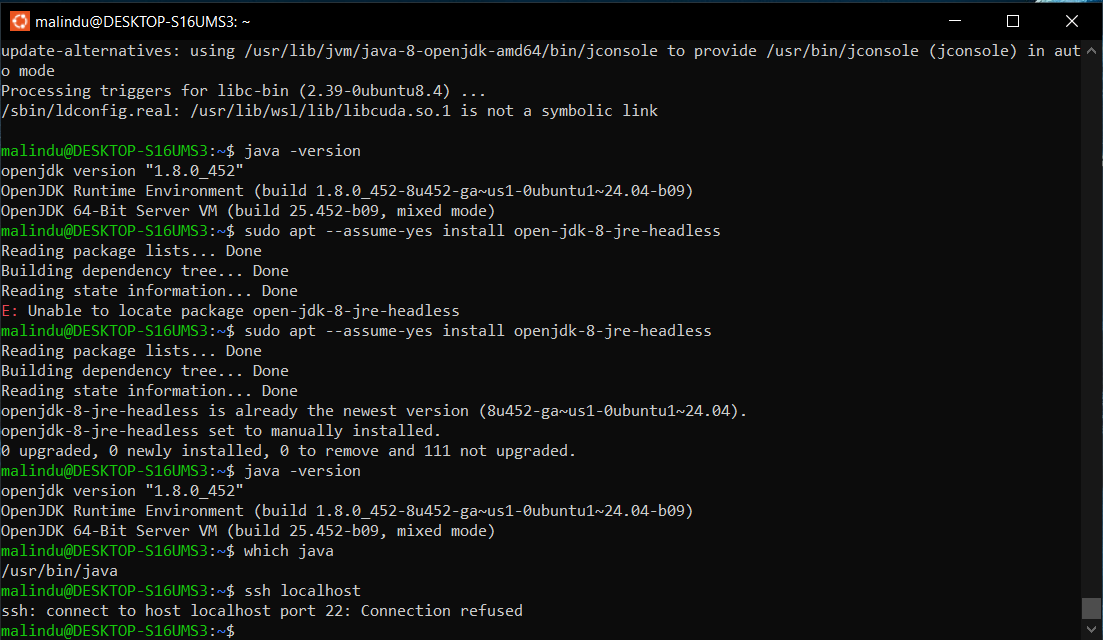


Figure . : Java version check command

## Hadoop Installation

Download and extract Hadoop binaries using the commands below Figure 3.6. We use version 3.3.6 for this project due to its stability and compatibility with recent Java versions.

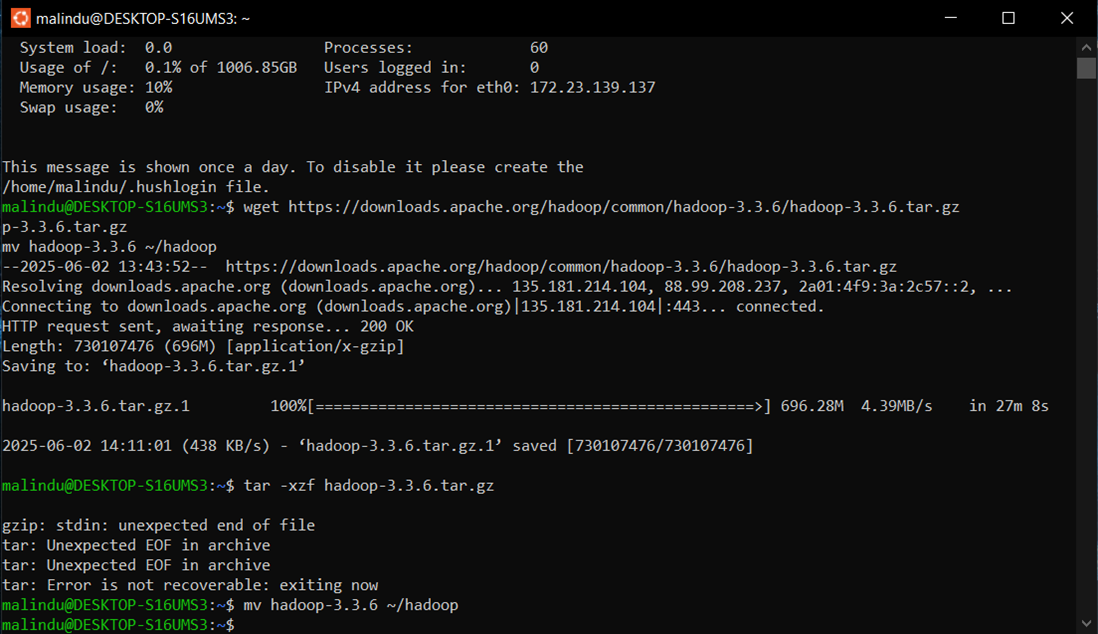


Figure . : Download and exact Hadoop

Ensure the Hadoop folder is accessible and properly located inside your home directory.

## Configuration Files

Hadoop requires configuration files to define file system properties, replication settings, job execution framework, and resource manager communication.

Files to configure:

* **core-site.xml:** Defines the NameNode (HDFS) URI
* **hdfs-site.xml:** Sets HDFS block size, replication, and storage directories
* **mapred-site.xml:** Configures MapReduce to run on YARN
* **yarn-site.xml:** Specifies ResourceManager hostname and NodeManager services

We can edit above things using VS Code or nano. For example,

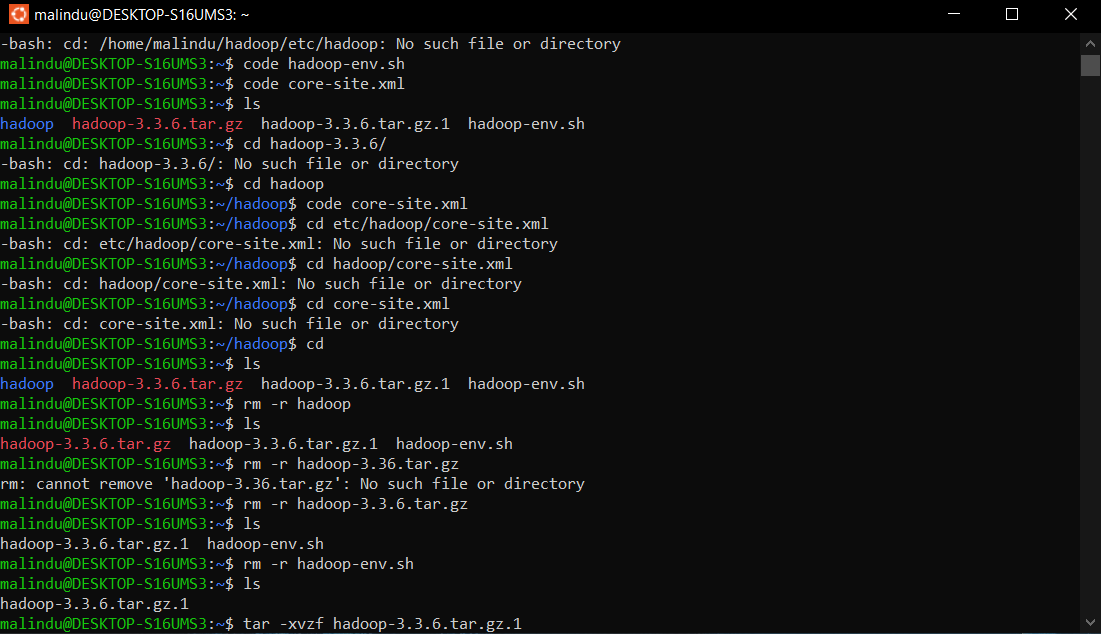


Figure . : Navigating to Hadoop Configuration File core-site.xml using VS Code

## Environment Variables in ~/.bashrc

To allow terminal wide access to Hadoop commands, should add the following environment variables to.bashrc file.

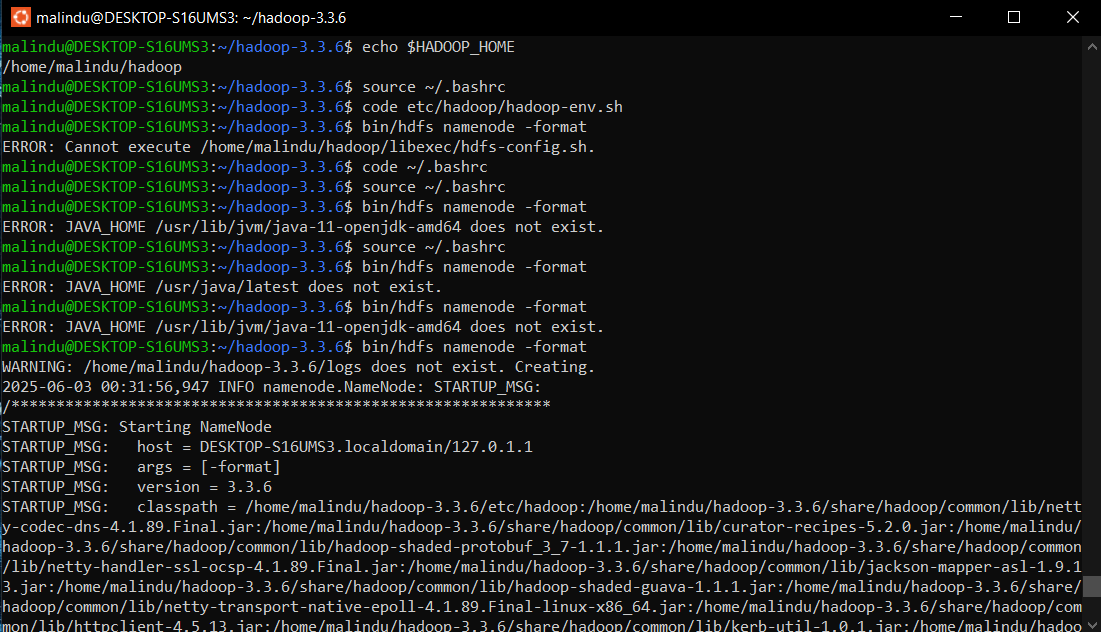


Figure . : Open .bashrc file

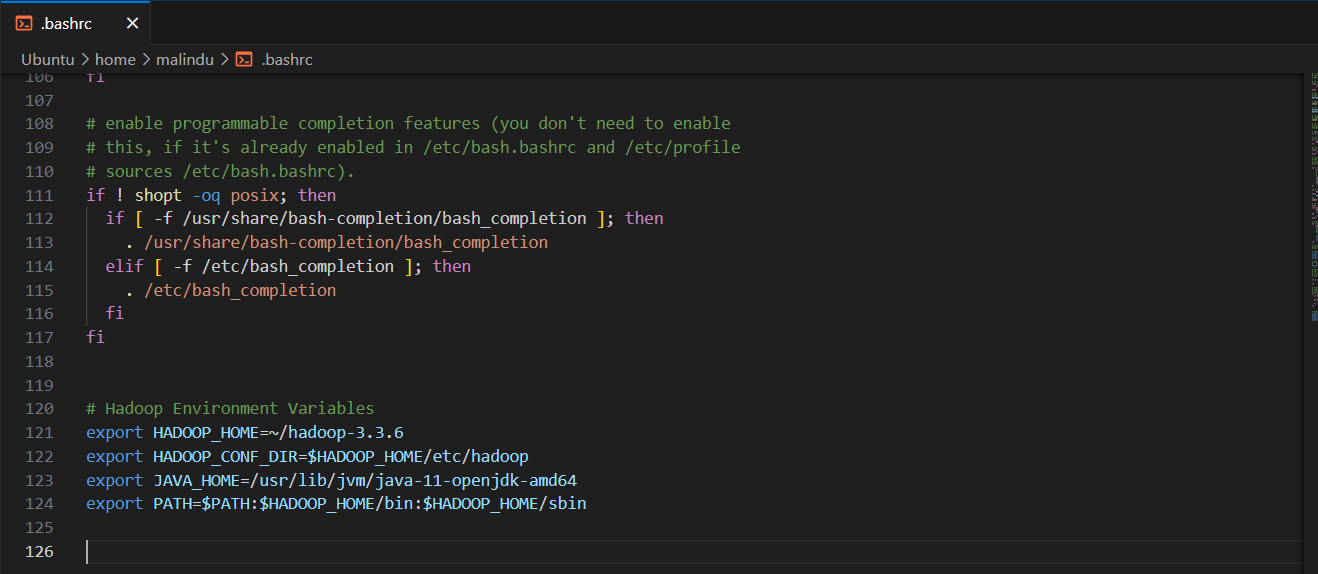


Figure . : bashrc file

After editing, should apply the changes.



Figure . : Code for source

This enables the use of commands like hdfs, yarn, start-dfs.sh, and others globally in the terminal.

## Start Hadoop

To initialize and start the Hadoop file system and YARN services should follow these steps.

1. Format the HDFS NameNode (only required for the first-time setup)

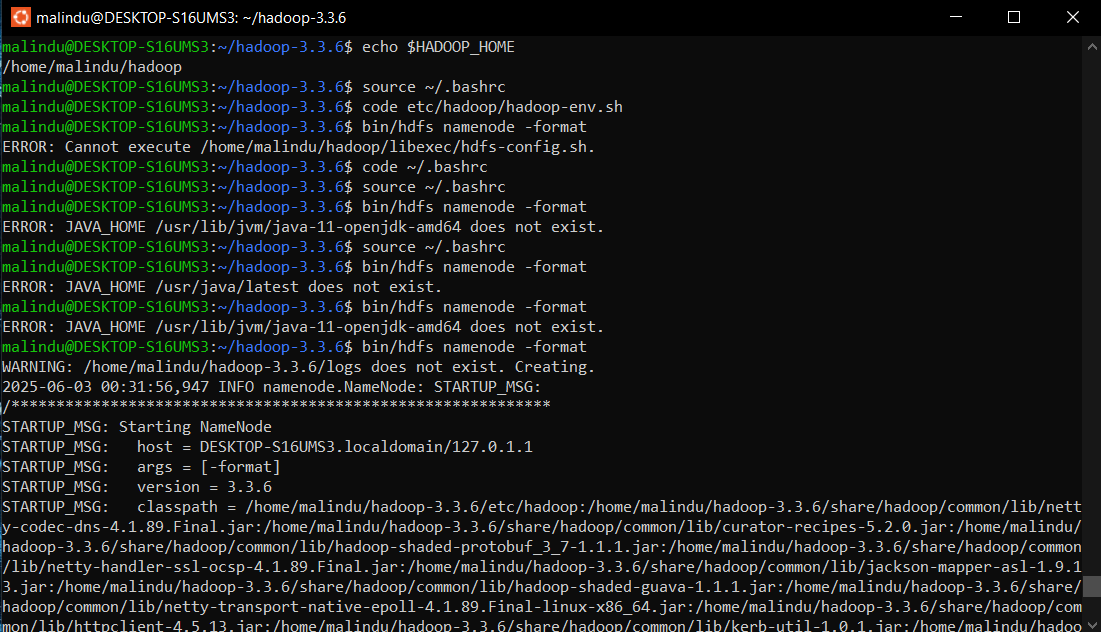


Figure . : Format the HDFS NameNode

This sets up the directory structure and initializes HDFS metadata.

1. Start the HDFS daemons

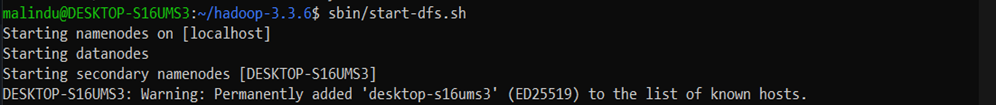


Figure . : Start the HDFS daemons

We should see output that includes both NameNode and DataNode.

1. Start the YARN daemons

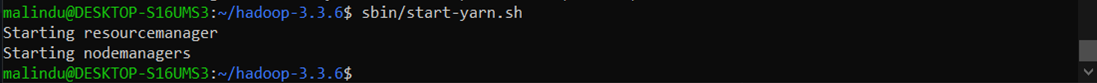


Figure . : Start the YARN daemons

This command starts the NameNode, DataNode, ResourceManager and NodeManager. Verify they are running with following command (Figure 3.14).

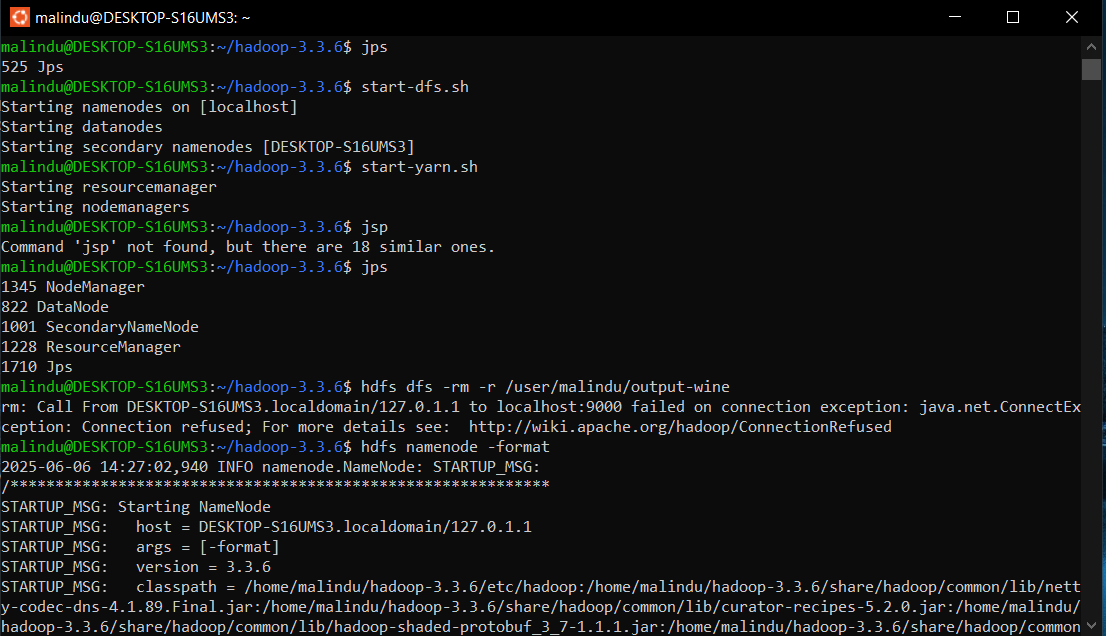


Figure . : Output of the jps command

Once everything is running, you can open the Hadoop Web UIs:

* HDFS NameNode: <http://localhost:9870>
* YARN ResourceManager: <http://localhost:8088>

# MapReduce Task Implementation

MapReduce is a programming model for processing large datasets with a distributed algorithm on a Hadoop cluster. In this project, the MapReduce job is implemented in Java to compute the average wine price grouped by variety. It includes a Mapper to extract key-value pairs, and a Reducer to aggregate and compute averages.

## Task Description

The main objective of this task is to calculate the average wine price per variety using the dataset. Each row of the CSV contains details such as price and wine variety. Rows with missing or invalid prices or varieties are skipped. This ensures only clean and usable data is used in calculations.

The final output is a list of wine varieties along with their respective average prices. This result can help consumers and businesses understand which varieties are priced higher or lower on average.

## Algorithm Design

* **Mapper Function:** Reads each line of the dataset and splits it by commas. Extracts the 'variety' and 'price' columns. Emits the wine variety as the key and its price as the value. Skips malformed rows with missing or non-numeric prices.
* **Reducer Function:** For each unique wine variety key, receives a list of prices. It computes the sum of prices and counts how many prices were received. Then it calculates the average by dividing the sum by the count and emits the variety with its average price.

This two-step process ensures distributed, parallel processing of large datasets with accurate aggregation.

## Source Code Summary

* **Language**: Java
* **Source code**: <https://github.com/MalinduDissanayaka/wine_review_analyzer_project.git>
* **Files**:

1. AveragePricePerVariety.java (Driver)
2. PriceMapper.java
3. AveragePriceReducer.java

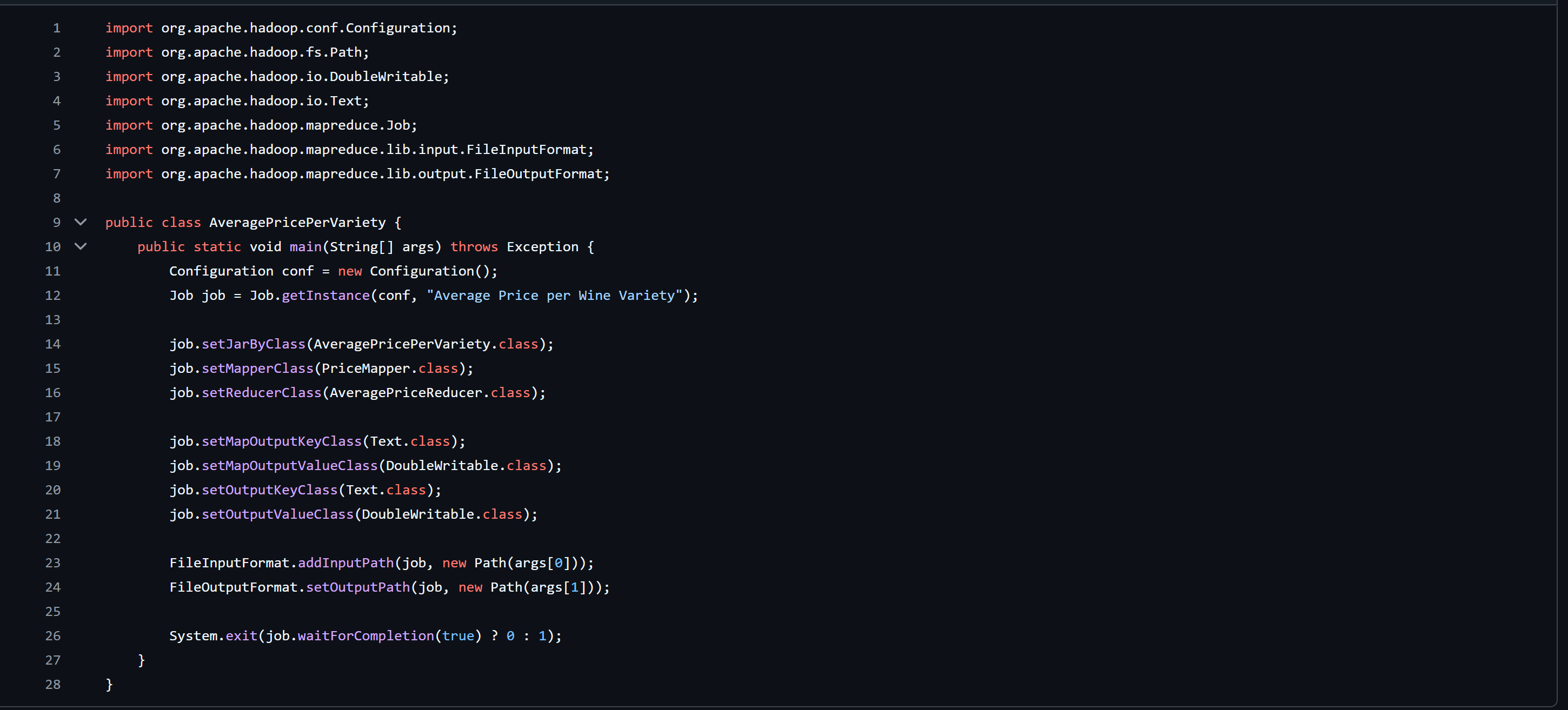


Figure . : AveragePricePerVariety.java file



Figure . : AveragePriceReducer.java file code

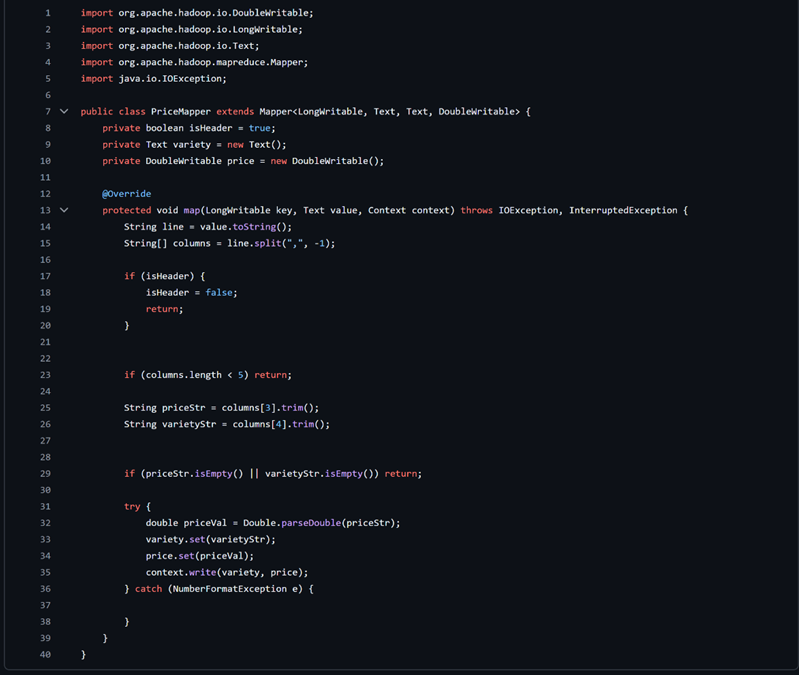


Figure . : PriceMapper.java file code

# Execution and Testing

This section outlines how to execute and test the MapReduce job in the Hadoop ecosystem. It covers uploading the dataset to HDFS, running the compiled JAR file, and inspecting the output to verify successful execution.

## Upload Dataset to HDFS

First, ensure that HDFS is running. Then, create a directory for input if it doesn't already exist and upload the dataset file to that directory.

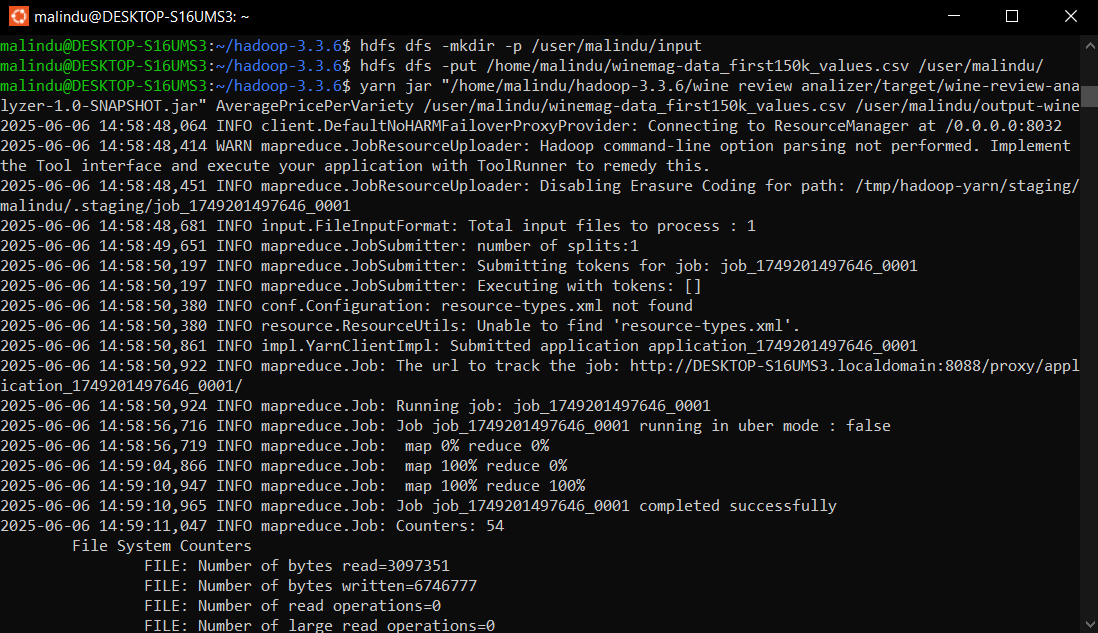


Figure . : Creating HDFS directory and uploading a CSV file in Hadoop

## Run MapReduce Job

Ensure the Java classes are compiled and the JAR file has been created. We can execute MapReduce job using the following command.

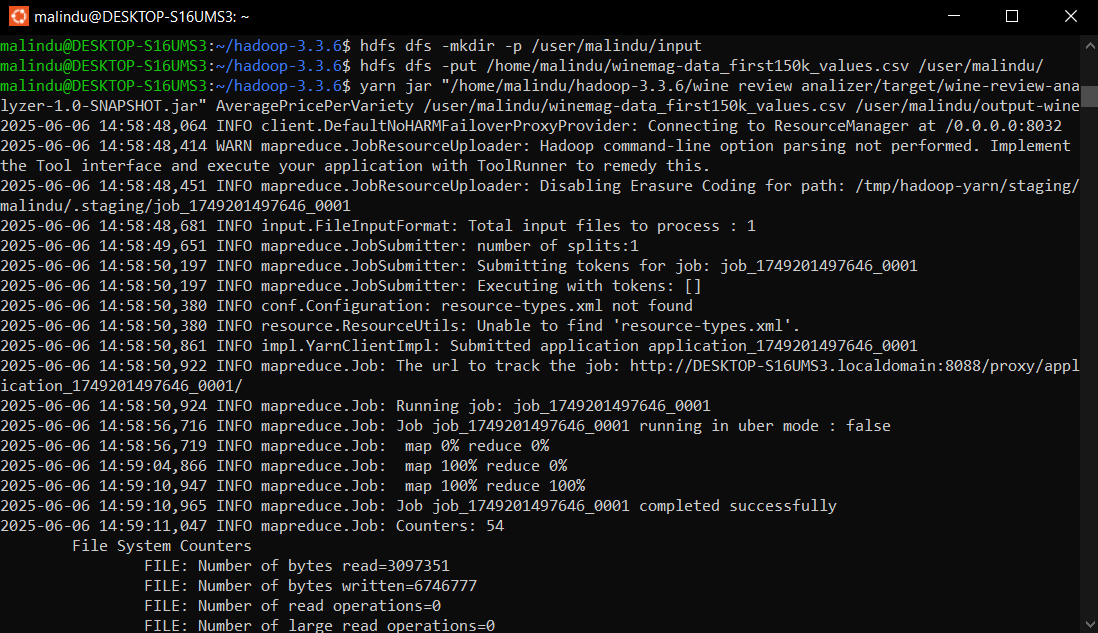


Figure . : Hadoop MapReduce job calculating average wine prices per variety

This command runs the AveragePricePerVariety driver class using the specified input and output paths in HDFS.

If the output directory already exists, delete it using following Figure 5.3 showing code.



Figure . : Command-line interface

Job status can be check via the YARN ResourceManager UI at <http://localhost:8088>.

## Output Sample

Once the job finishes, we can read the output stored in the HDFS output directory. It has shown by below Figure 5.4.

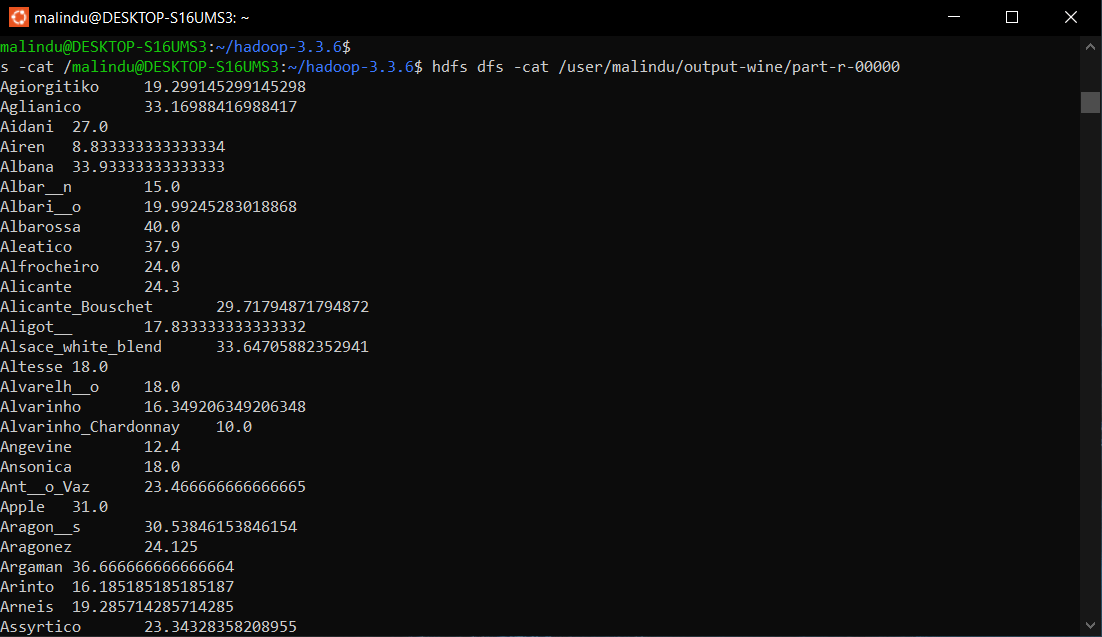


Figure . : Hadoop MapReduce output: Average wine prices by variety.

This confirms successful grouping and averaging of wine prices by variety.

# Results and Interpretation

The output consists of wine varieties and their calculated average prices. It demonstrates the ability of Hadoop MapReduce to extract analytical insights from structured but noisy datasets.

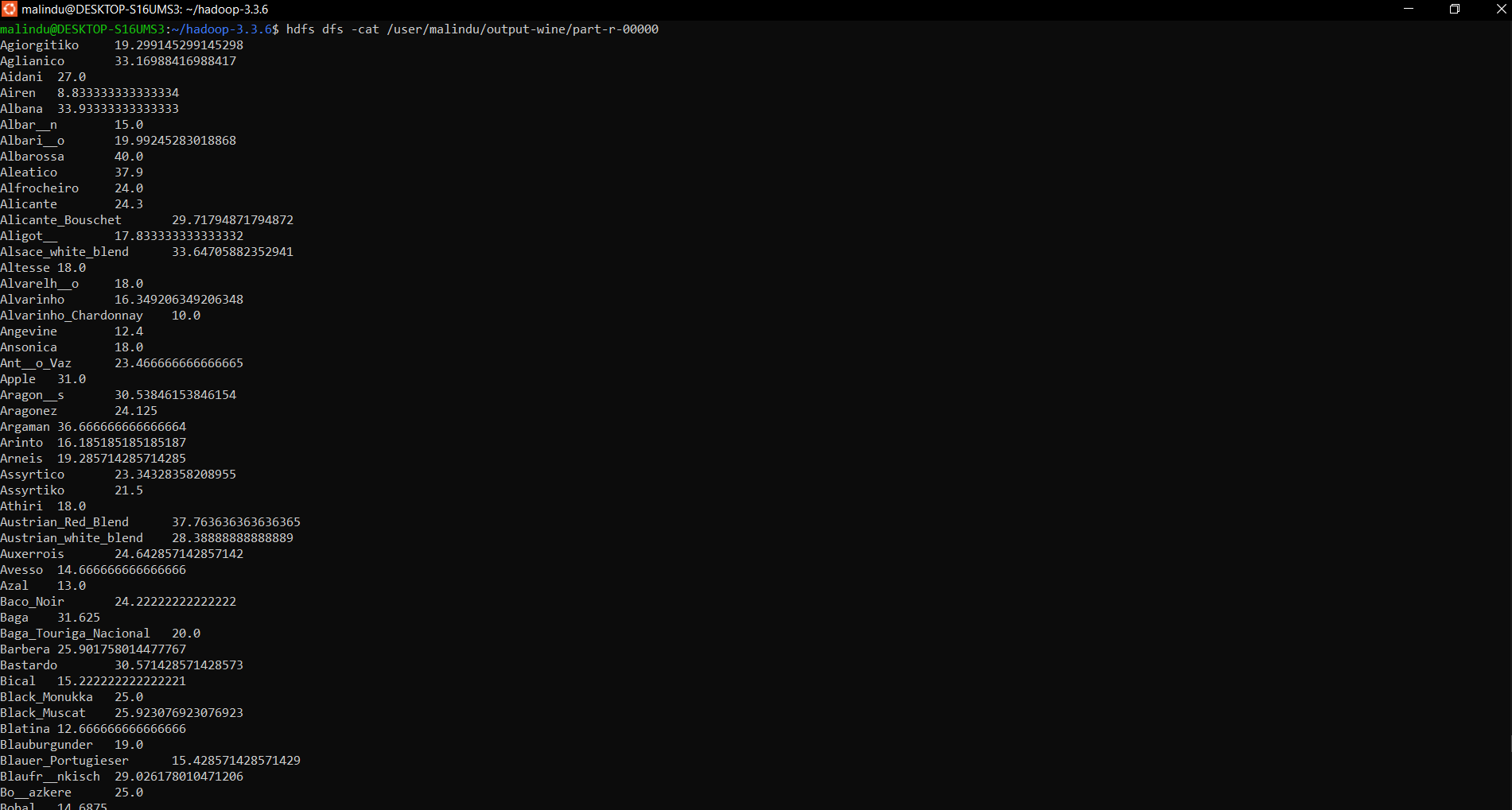


Figure . : Output terminal of wine varieties and their calculated average prices

# Hadoop Web Interfaces

* **NameNode**: <http://localhost:9870>

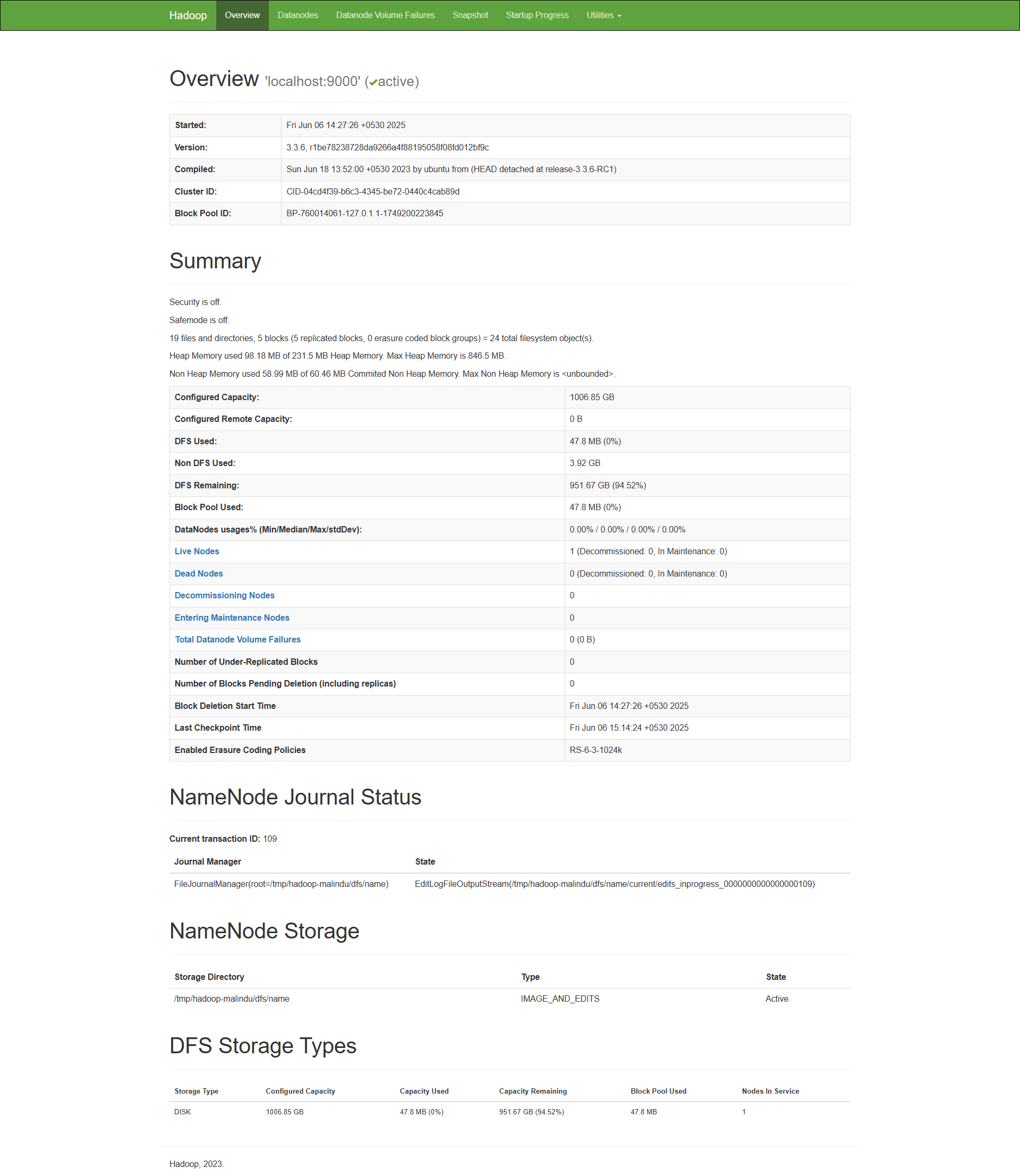


Figure . : Hadoop web interface (NameNode)

* **ResourceManager**: <http://localhost:8088>

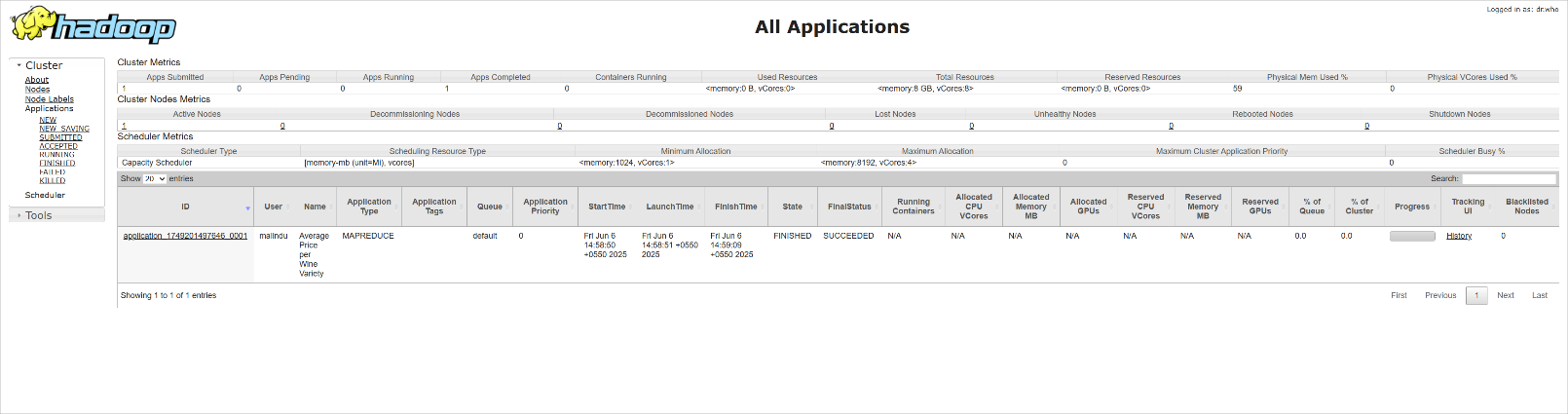


Figure . : Hadoop web interface (ResourceManager)