Haifa University

Big Data Course

Homework 1

Running MapReduce with Hadoop **submission deadline 25.01.2025**

## Abstract

This assignment aims to familiarize you with the Hadoop framework and concept of Map Reduce. You will need to install it on your laptop's containerization environment, enabling you to run on your computers regardless of the operating system's simple cluster of Hadoop. Later in the assignment, you will learn how to use it to check your implementations of the Map-Reduce-related tasks.

## Instructions The assignment has to be done in pairs or triples, unless you received explicit permit from the lecturer. The submission should include:

* 1. A title page with your names and IDs
  2. Implementation of the code
  3. If needed instructions and explanations should be submitted within word document file
  4. Finally, all homework artifacts have to be submitted compressed with zip   
       
     **NOTE: Please do not submit JAR files**
  5. Submission deadline is 25/01/2025 23:59

## Project Description

In this homework you will learn about Docker the containerization tooling, how to leverage it to setup small cluster of Hadoop nodes on your computers. You are provided with needed configuration files, therefore there is no need for you to write definitions of the containers, but rather reuse existing ones. The goal of this homework is to get you familiar and train implementing tasks with Hadoop and MapReduce and Docker is used only for educational purposes to allow you to run local cluster at your computer.

Once you will run your own Hadoop cluster you will need to make sure it’s up and running learn about control dashboards to check your installation and cluster configuration. After you will get familiar with how to run your own cluster you are given 3 simple MapReduce tasks to implement, and you will need to execute results on you Hadoop cluster and extract results.

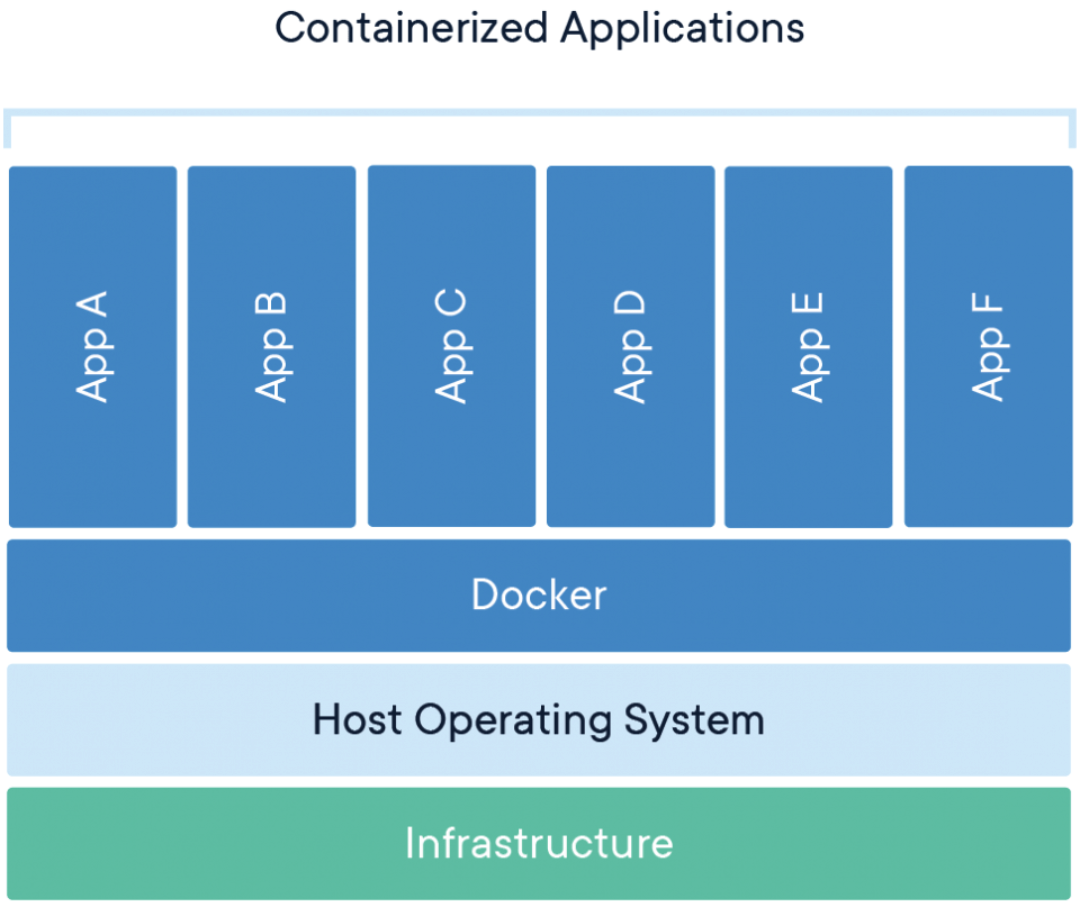
## What is Docker?

Docker is an open-source containerization platform used for developing, deploying, and managing applications in lightweight virtualized environments called containers.

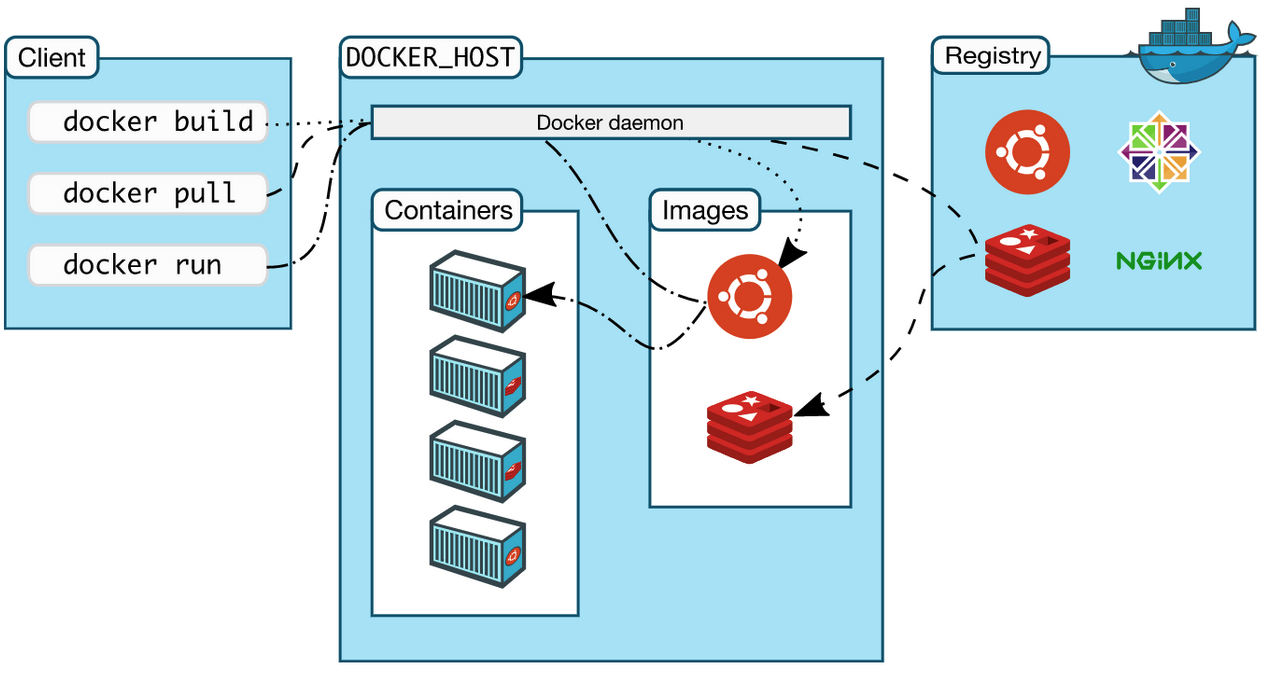
It is mainly used as a software development platform for developing [distributed applications](https://phoenixnap.com/blog/securely-connect-distributed-apps) that work efficiently in different environments. By making the software system agnostic, developers don’t have to worry about compatibility issues. Packaging apps into isolated environments (containers) also makes it easier to develop, deploy, maintain, and use applications.

Since Docker utilizes virtualization to create containers for storing apps, the concept may seem similar to virtual machines. Although both represent isolated virtual environments used for software development, there are important [differences between containers and VMs](https://phoenixnap.com/kb/containers-vs-vms). The most crucial distinction is that Docker containers are lighter, faster, and [more resource efficient](https://phoenixnap.com/kb/docker-memory-and-cpu-limit) than virtual machines.

A Docker container is a pre-configured environment that includes all of the necessary installations for the application running inside it. Each container has a running application, and each container runs on the Docker Engine, which in turn runs on top of the host operating system.



Docker containers use Docker container images to containerize an application. For every application, there is an official Docker container image that you can use to containerize your application dependencies and executables.



## Installing Docker (10pts) Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you:

* 1. **Windows**  
     <https://docs.docker.com/desktop/install/windows-install/>
  2. **MacOS**  
     <https://docs.docker.com/desktop/install/mac-install/>
  3. **Linux**  
     <https://docs.docker.com/desktop/install/linux-install/>

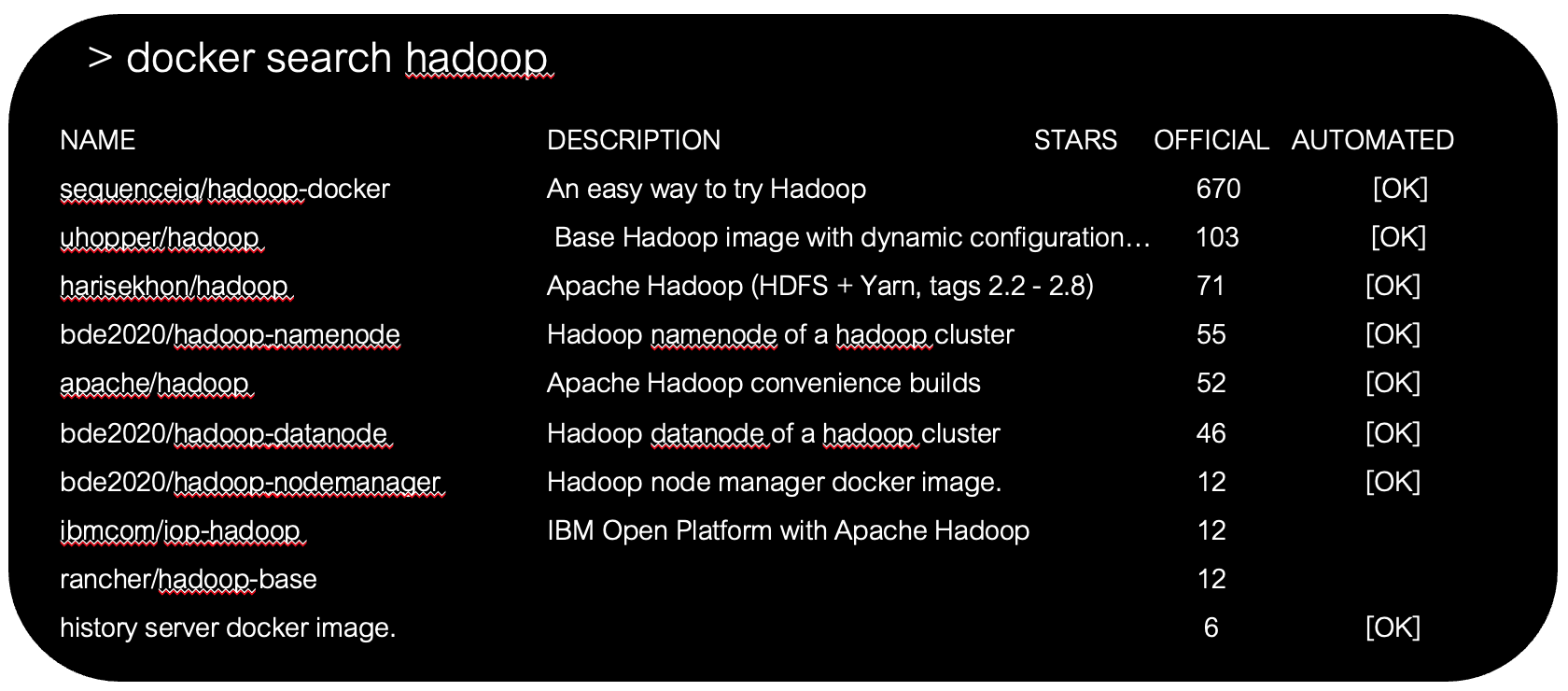
In order to check your installation, you can run at your terminal following command:

docker run -it --rm hello-world  
  
It will run example container which will print greetings and system information about your docker installation, note first time it might take a while since it downloads the image from Docker Hub (repository of images).

In case something goes wrong please read documentation here:

1. <https://docs.docker.com/get-started/>
2. <https://docs.docker.com/install/>

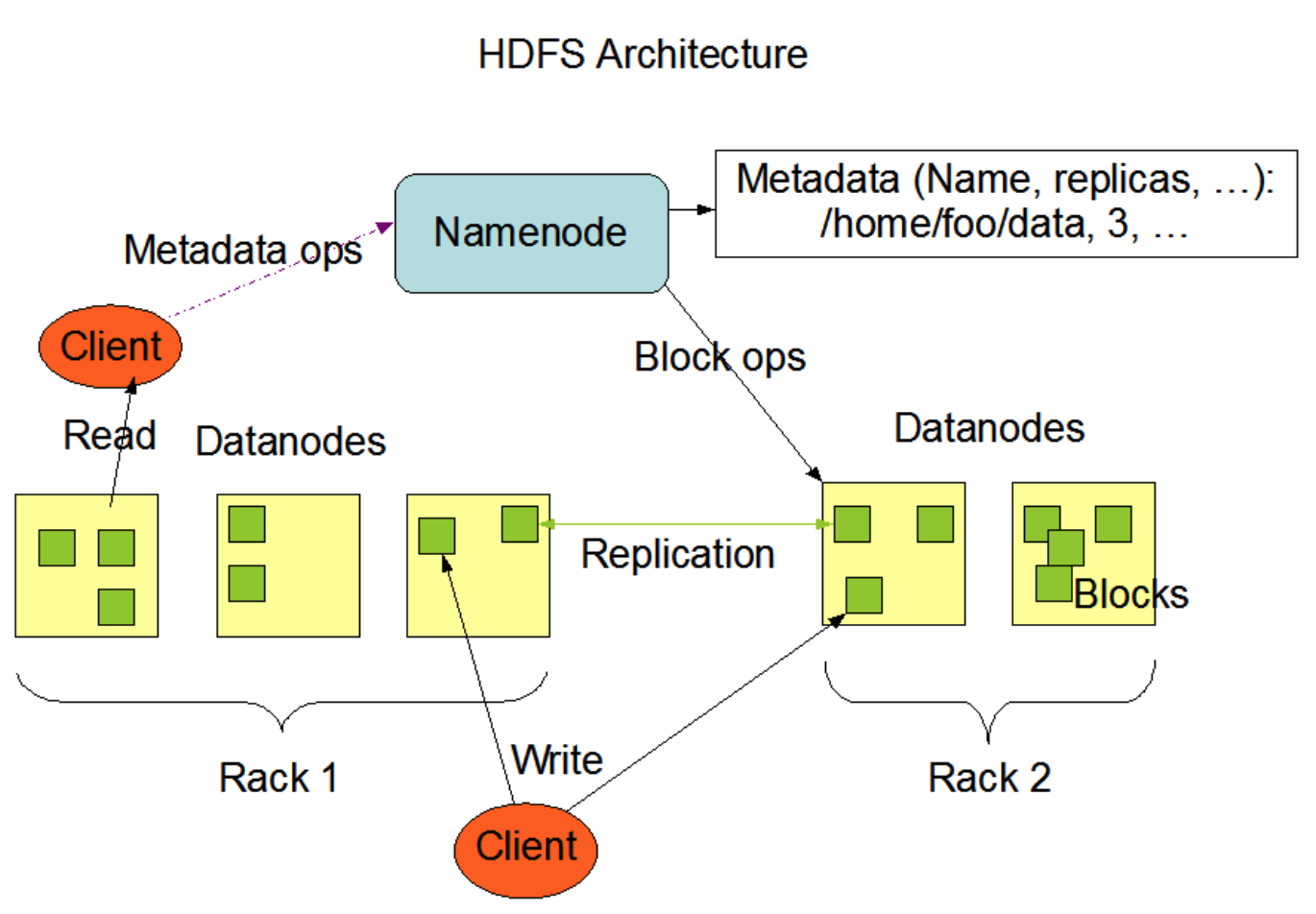
Once docker is installed you can explore available docker images that encapsulates different functionalities, for instance images for Hadoop:

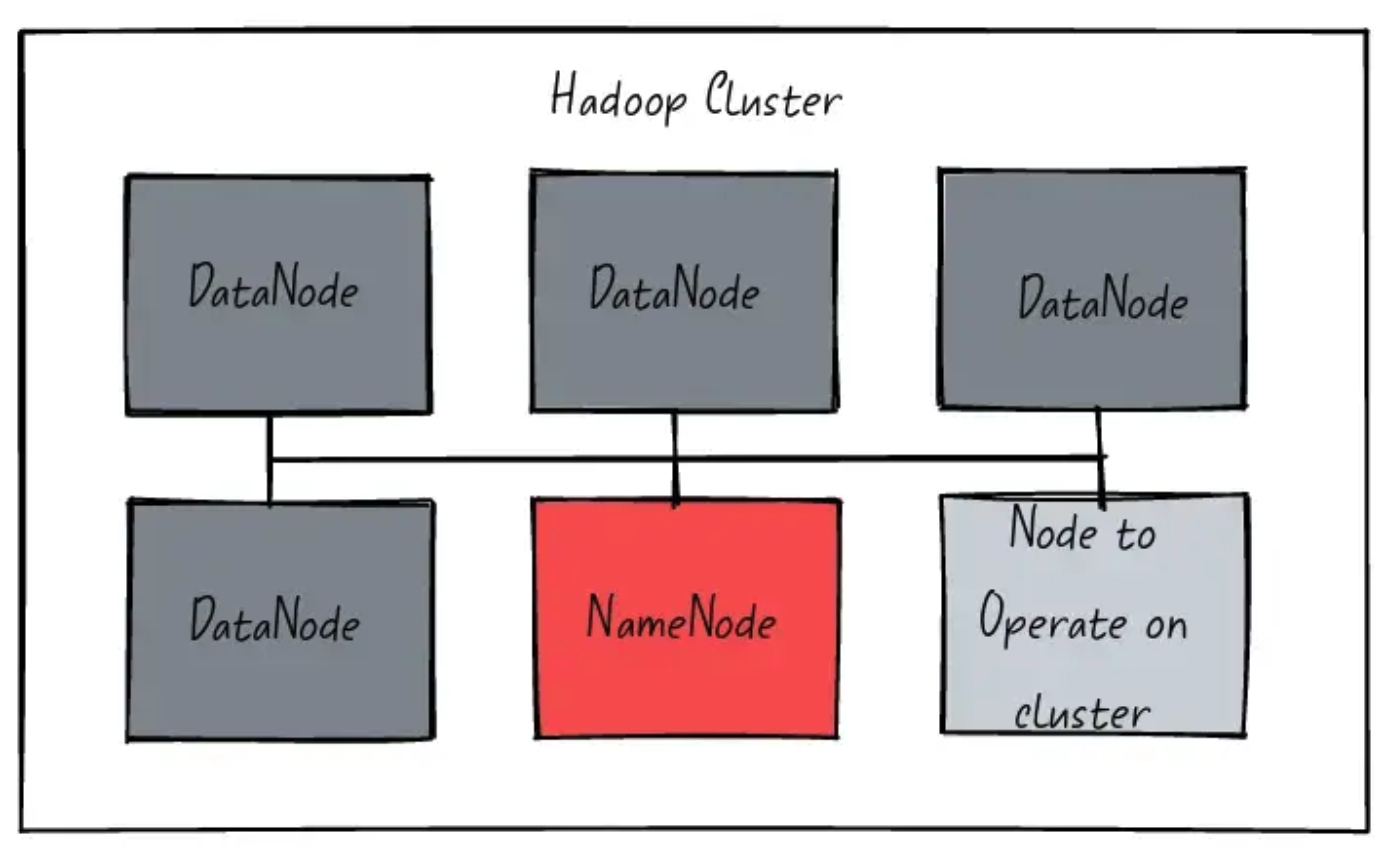


## Docker Compose (10pts) Docker Compose is a useful tool designed to simplify the process when running and managing multiple containers simultaneously. It strings multiple containers needed to work together and controls them through a single coordinated command.

Docker Compose launches, executes, communicates, and closes containers with a command. This is done using a YAML file configuring the application's services.

For example, a Hadoop cluster is composed from multiple nodes each exhibit different functionality, regular Hadoop cluster looks as following:



You are provided with ready and configured docker compose file with Hadoop cluster, in next sections you will see instruction on how to run it. In a nutshell your local cluster will look like the following:  


* **NameNode** and **DataNode** are nothing but processes running on a system. Thus, we can instantiate or delete a **NameNode** or **DataNode** as we please.
* To access these or operate on the cluster we connect an additional node to the cluster that has the same configurations as other and is connected to the same network.

So, all you need to do for creating your own basic Hadoop cluster is to have a bunch of machines connected over the same network, put some Hadoop configuration files in those, and start the process for HDFS (NameNode and DataNode). If you have configured it properly, Hadoop utilities such as YARN or Hadoop (command) should work without a problem. Let’s see how we do that.

## Installing Docker-Compose

The easiest and recommended way to get Docker Compose is to install Docker Desktop. Docker Desktop includes Docker Compose along with Docker Engine and Docker CLI which are Compose prerequisites.

* 1. Windows  
     <https://docs.docker.com/desktop/install/windows-install/>
  2. MacOS  
     <https://docs.docker.com/desktop/install/mac-install/>
  3. Linux  
     <https://docs.docker.com/desktop/install/linux-install/>

## Running of the Docker-Compose Hadoop project (10pts)

In this project you are provided with the archive **docker-hadoop.zip**, which contains docker compose configuration of the Hadoop cluster. You need to unzip the content of the archive and open your terminal window, from within the terminal you need to step into the folder and run following command:

docker-compose up

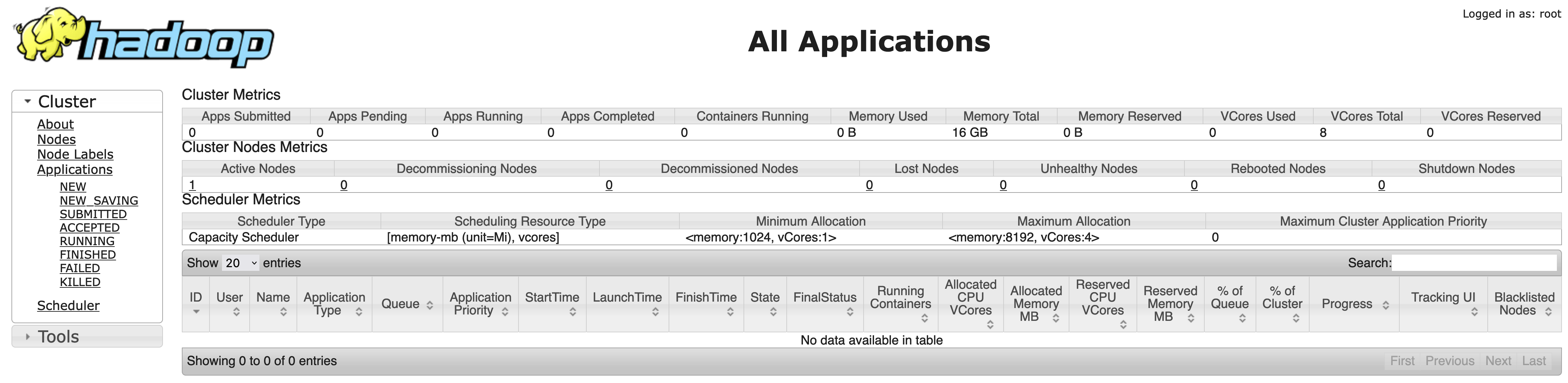
This will start running the Hadoop nodes and spanning the cluster on your local laptop, the process takes about few minutes, therefore be patient and wait until its complete bootstrapping the cluster. Once cluster will be running you will see in your terminal following output:

**Scheduler recovery is done. Start allocating new containers.**

In order to validate you have working cluster you can open your browser and enter into address bar:

http://localhost:8089/cluster

you should see following **UI**:



**Please click on About and attach screenshot to your final report.**

Next, you can check application history server, which is service maintaining history of all jobs ran on your cluster, in your case it will be empty, since you just started, go in your browser to:

http://localhost:8188/applicationhistory

Next, is the **NodeManager** which exposes all information about nodes in your cluster:

http://localhost:8042/node

Here it is, your Hadoop cluster is up and running and you can start working with it. For the time being while you do not need your cluster, you better to shut it down, with following command:

docker-compose down -v

Now, you are ready to implement your first Map-Reduce programs.

## MapReduce tasks (70pts)

In today’s digital world, mobile applications have become an integral part of our daily lives. Understanding mobile app usage patterns can offer valuable insights into user behavior, app performance, and engagement metrics. This homework is designed to help students apply Hadoop and MapReduce to analyze large datasets efficiently, uncovering hidden patterns and generating meaningful insights. Students will use real-world data taken from Kaggle, a popular platform for data science competitions, to practice processing, aggregating, and analyzing data at scale.

### 9.1 Dataset description

This DataSet Contains Detailed Insights into **Mobile App Usage Patterns, including ScreenTime**, notifications received, and app openings. The data spans multiple days in August and some popular apps, offering a granular view of digital behavior.

## Features:

#### 1. Date: The date of the recorded data.

#### 2. App: The name of the mobile application.

#### 3. Usage (minutes): Total minutes spent using the app on a given day.

#### 4. Notifications: Number of notifications received from the app.

#### 5. Times Opened: How many times the app was launched.

### 9.2 Tasks:

### Top 5 Most Used Apps by Screen Time (10pts)

**Objective:** Identify the top 5 most used apps based on total screen time across all dates.

**Steps to Follow:**

1. Use MapReduce to process the dataset.

2. Sum up the Usage (minutes) for each app across all dates.

3. Sort the apps by total screen time in descending order.

4. Return the top 5 apps with the highest screen time.

**Expected Output Example:**

1. Instagram - 1200 minutes

2. TikTok - 950 minutes

3. YouTube - 850 minutes

4. WhatsApp - 800 minutes

5. Facebook - 750 minutes

### Average Notifications per App (10pts)

**Objective:** Calculate the average number of notifications received from each app.

**Steps to Follow:**

1. Use MapReduce to sum the total notifications for each app.

2. Count the number of days each app appears in the dataset.

3. Divide the total notifications by the number of days to calculate the average.

**Expected Output Example:**

Instagram - 25.3 notifications/day

TikTok - 18.7 notifications/day

YouTube - 12.1 notifications/day

### Daily App Launch Patterns (10pts)

**Objective:** Find the app that was opened the most times on each day.

**Steps to Follow:**

1. Group the data by Date and App.

2. Calculate the total Times Opened for each app on each day.

3. Identify the app with the highest number of launches for each day.

**Expected Output Example:**

2024-08-07 - Instagram (57 times)

2024-08-08 - TikTok (65 times)

2024-08-09 - YouTube (45 times)

### Notifications vs. Usage Analysis (20 pts)

**Objective:** Analyze the relationship between notifications received and screen time to identify if a correlation exists.

**Steps to Follow:**

1. Use MapReduce to calculate the total Usage (minutes) and total Notifications for each app.

2. Generate a report that shows whether apps with more notifications also have higher usage.

**Expected Output Example:**

App Total Usage (minutes) Total Notifications

Instagram 1200 540

TikTok 950 430

YouTube 850 320

### Most Frequent App Launch Days (20pts)

**Objective:** Determine which day of the week users launch apps the most frequently.

**Steps to Follow:**

1. Convert the Date field to a day of the week (e.g., Monday, Tuesday).

2. Use MapReduce to count the total app launches (Times Opened) for each day of the week.

3. Identify the most frequent day for app launches.

**Expected Output Example:**

Monday - 2500 launches

Tuesday - 2700 launches

Wednesday - 2300 launches

Thursday - 2800 launches

Friday - 3100 launches

Saturday - 2900 launches

Sunday - 2400 launches

### 9.3. Submission Requirements

• Submit your Hadoop and MapReduce code for each task.

• Include a report with the results generated by your code.

• Provide a brief explanation of how your code works and the insights you obtained from the data.

### 9.4 Evaluation Criteria

|  |  |
| --- | --- |
| **Criteria** | **Weightage** |
| Code Functionality | 40% |
| Code Efficiency | 20% |
| Accuracy of Results | 20% |
| Report Clarity and Insights | 10% |
| Code Documentation | 10% |

## Compile and Run Code in Local Hadoop cluster (10pts)

Before you start, please make sure your local Hadoop cluster is up and running and in case it was stop, restart it again.

To complete you work and see how to run it on Hadoop, you need to log in into one of the cluster nodes to run commands from it, please run following command in your command line prompt:

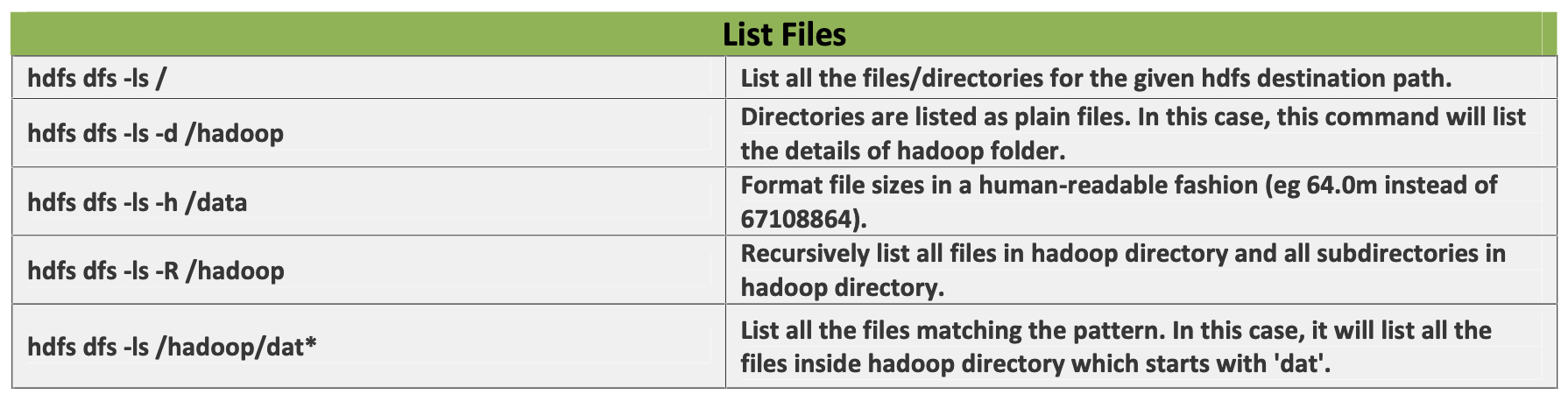
docker exec -it namenode /bin/bash

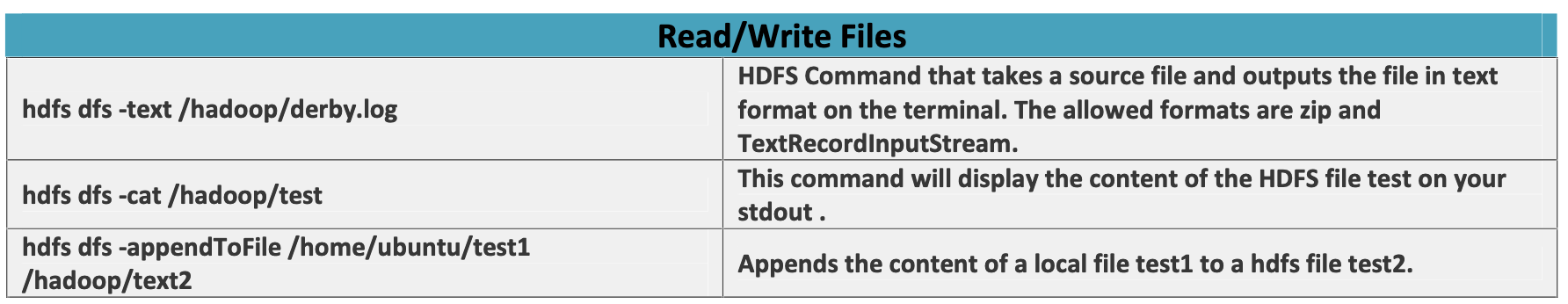
This will log you into one of the cluster nodes, next you need to learn how to work with HDFS so you can copy your files into it and then read outputs.

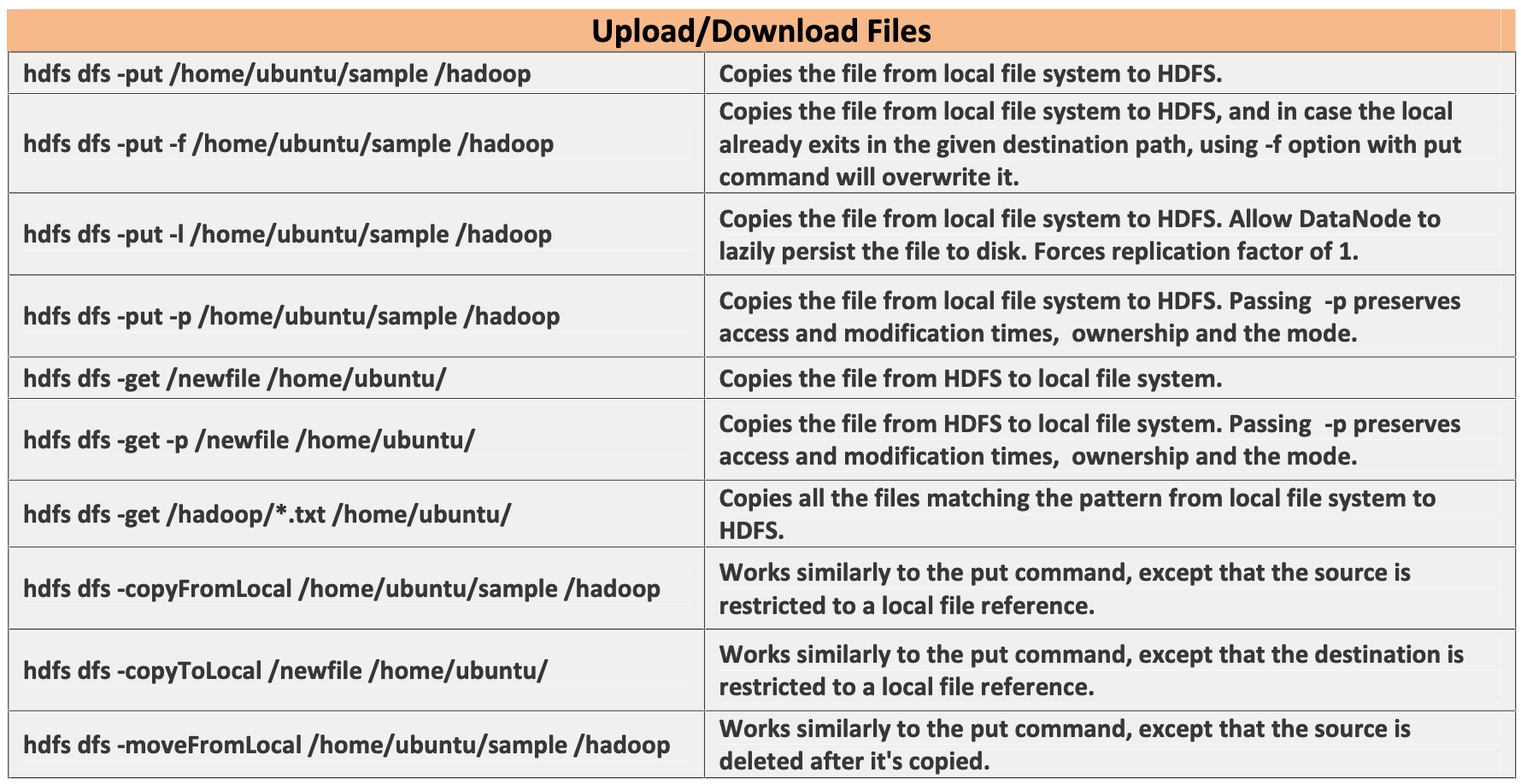
### Working with HDFS.

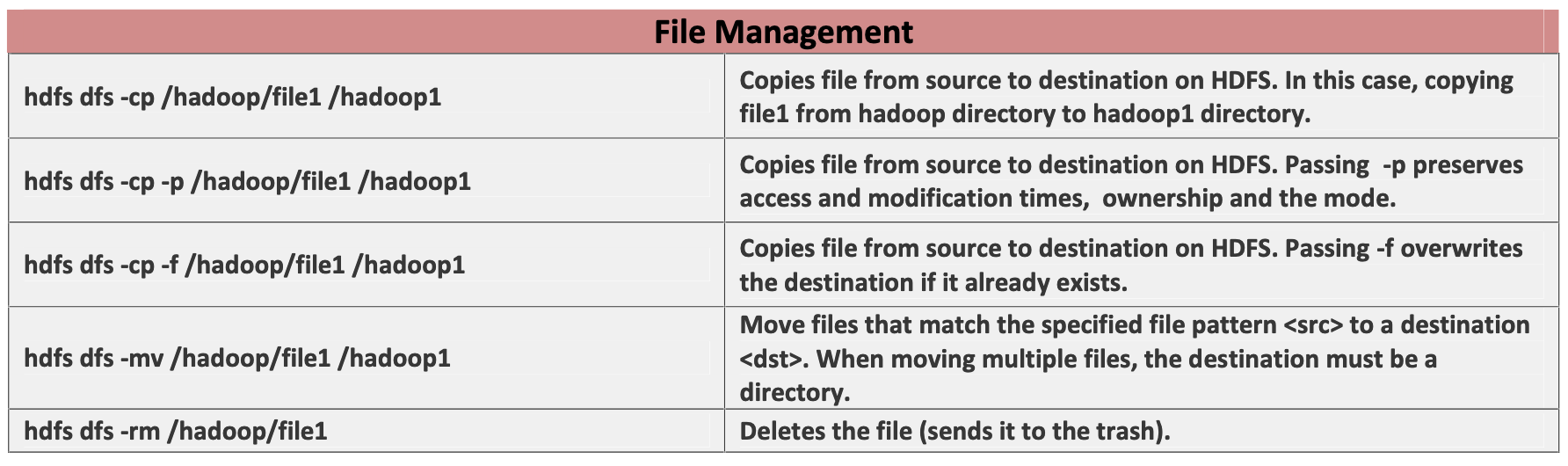
HDFS is a Hadoop distributed file system, which used as an abstraction to work with the inputs and outputs reading and writing results into it. In order to run your applications you need to learn how to operate with HDFS, therefore here are examples of the command you need to know in order to interact with HDFS within your local cluster.

Below list of commands to work with HDFS from command line in order to create folders, place files into HDFS and read content.









### Running code Map Reduce

* 1. You need to compile you Maven project with:   
     mvn clean install
  2. Resulted artifact will be generated into target folder
  3. You need to copy artifact into namenode container:  
     docker cp java\_project.jar namenode:/.
  4. If you need to copy text or input files:  
     docker cp textfile.txt namenode:/.
  5. Now you can log into namenode container:  
     docker exec -it namenode /bin/bash
  6. Create folder on HDFS:  
     hdfs dfs -mkdir -p /user/root
  7. Copy input file from #4:  
     hdfs dfs -copyFromLocal textfile.txt /user/root/textfile.txt
  8. Now you can run your Hadoop job:  
     hadoop jar java\_project.jar Java.Class.Name.Of.Driver <args>

After these steps you job will be submitted into Hadoop cluster for execution and scheduling, usually it takes a few minutes to get you job scheduled and to complete execution.