HDFS Simple Docker Installation Guide for Data Science Workflow

Easy step-by-step Installation and Usage of HDFS on your system using the Docker image.

Paras Varshney Mar 19 · 5 min read ★

Photo by Luke Chesser on Unsplash

demonstrate the use-case. Also once the setup is ready to be with on your machine, you can start building your own map-reduce jobs to play around with Hadoop DFS.

Start with cloning the HDFS Project

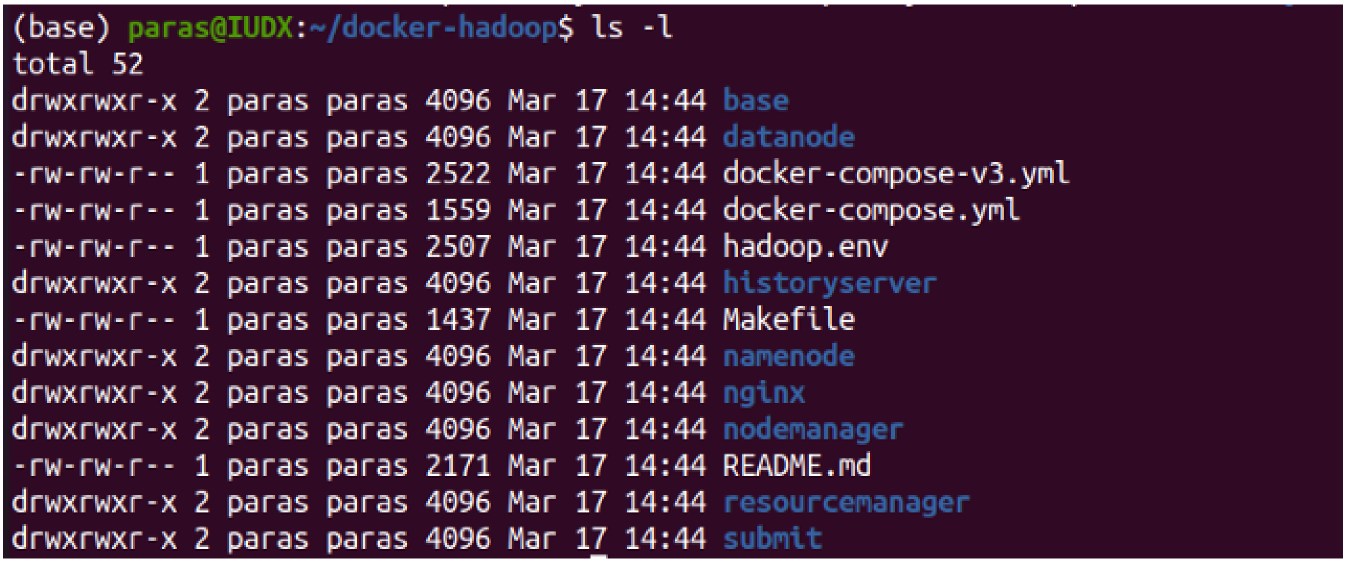
First, you need to **clone** the following **Git repository** into your desired directory in your system. I prefer cloning it into the home for demonstration and easy access.

```
git clone https://github.com/big-data-europe/docker-hadoop
```

then enter the project repository using the **change directory** command:

```
cd docker-hadoop
```

Now on doing an “**ls -l**” command you can see all the files inside this repository as shown below.



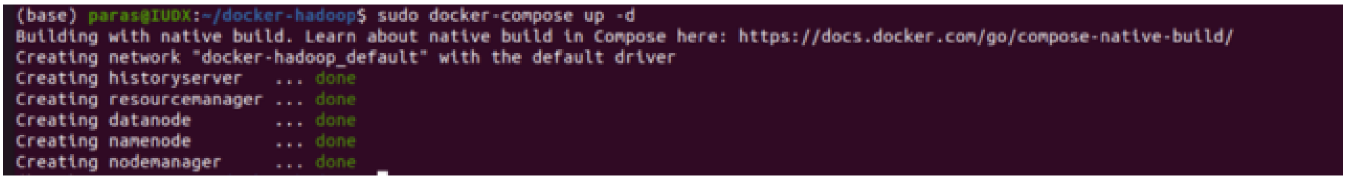
```
(base) paras@IUDX:~/docker-hadoop$ ls -l
total 52
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 base
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 datanode
-rw-rw-r-- 1 paras paras 2522 Mar 17 14:44 docker-compose-v3.yml
-rw-rw-r-- 1 paras paras 1559 Mar 17 14:44 docker-compose.yml
-rw-rw-r-- 1 paras paras 2507 Mar 17 14:44 hadoop.env
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 historyserver
-rw-rw-r-- 1 paras paras 1437 Mar 17 14:44 Makefile
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 namenode
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 nginx
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 nodemanager
-rw-rw-r-- 1 paras paras 2171 Mar 17 14:44 README.md
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 resourcemanager
drwxrwxr-x 2 paras paras 4096 Mar 17 14:44 submit
```

Output of the command “**ls -l**”

To do that, there is a docker-compose.yml file in the project directory, you need to use the “**docker-compose up**” command to download and install the required images from the Docker hub and configure the containers based on the **docker-compose.yml** file. The “**-d**” flag runs the containers in detached mode. If the images are not found locally, Docker downloads them from DockerHub, but if you want to manually download them you can use “**docker-compose pull**”.

> *Note: Use “**sudo** ” as the prefix of these commands, if you get a permission error.*

```
# download images required for setting up HDFS and spin up necessary
# containers.
docker-compose up -d
```



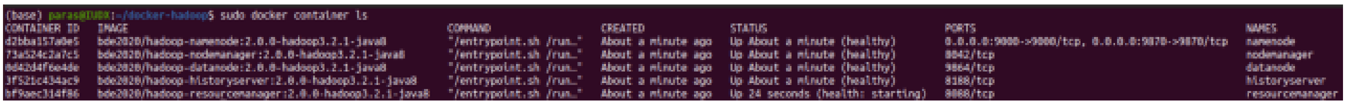
```
(base) paras@IUDX:~/docker-hadoop$ sudo docker-compose up -d
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Creating network "docker-hadoop_default" with the default driver
Creating historyserver   ... done
Creating resourcemanager ... done
Creating datanode        ... done
Creating namenode        ... done
Creating nodemanager     ... done
```

Output from the command “**sudo docker-compose up -d**”

The above command will download all the necessary Docker images from the docker hub for setting up HDFS containers. It might take a little while to download images depending on your internet speed.

Now, to have a look at your current running Docker containers, use the command to list all active containers.

```
# List all the available running docker containers.
docker container ls
```



```
(base) paras@IUDX:~/docker-hadoop$ sudo docker container ls
CONTAINER ID   IMAGE                                                    COMMAND                  CREATED              STATUS                             PORTS                                            NAMES
d2bba157a0e5   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"   About a minute ago   Up About a minute (healthy)        0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp   namenode
73a524c2a7c5   bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   About a minute ago   Up About a minute (healthy)        8042/tcp                                         nodemanager
6d42d4f6e4de   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"   About a minute ago   Up About a minute (healthy)        9864/tcp                                         datanode
3f521c434ac9   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8     "/entrypoint.sh /run…"   About a minute ago   Up About a minute (healthy)        8188/tcp                                         historyserver
bf9aec314f86   bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8   "/entrypoint.sh /run…"   About a minute ago   Up 24 seconds (health: starting)   8088/tcp                                         resourcemanager
```

Output of the command “**sudo docker container ls**” this example, I have copied files in **/tmp**), now you would go inside the **namenode** using the following command in an interactive terminal mode in bash mode.

```
# Enter inside namenode and open its bash
docker exec -it namenode /bin/bash
```

Example: **sudo docker cp my_input.txt namenode:/tmp/**

Copy necessary JAR and Input files

Now we need to copy the jar files which contains our **map-reduce jobs** and copy them inside the namenode (which will be running your jobs) in HDFS using the following Docker command:

```
docker cp <file_name> namenode:/<path>
```

Interact with namenode

Once you enter the name node in an interactive terminal, use the following HDFS commands to interact with the **namenode**.

```
# HDFS list commands to show all the directories in root "/"
hdfs dfs -ls /

# Create a new directory inside HDFS using mkdir tag.
hdfs dfs -mkdir -p /user/root

# Copy the files to the input path in HDFS.
hdfs dfs -put <file_name> <path>

# Have a look at the content of your input file.
hdfs dfs -cat <input_file>
```

Output of above mentioned commands

Run Hadoop Map Reduce Jobs

Now you can run your map-reduce job using the following command:

```
# Run map reduce job from the path where you have the jar file.
hadoop jar <jar_file_name> <class_name> input output
```

Example: **hadoop jar word_counter.jar org.apache.hadoop.examples.WordCount input output**

Once this command runs **successfully**, you will notice that the map-reduce job completes its execution with some information on the console about the process.

Check Your Output

Once the job is executed successfully, you can check your output using the cat command in HDFS:

```
# Check the content of the output file after running the job
hdfs dfs -cat <output_file>
```

You will see the word frequency of your word counter job should be printed on the console.

Congratulations! You have successfully configured and created your first Hadoop HDFS map-reduce job!

Bonus Tip

You can access the HDFS namenode's **UI dashboard** on your localhost at port 9870. Use the following link:

> http://<your_ip_address>:9870

Namenode UI: **http://<your_ip_address>:9870**

Conclusion:

Docker-based HDFS set up and start writing your own map-reduce jobs to execute various tasks. If you are not very familiar with the map-reduce jobs so I have attached a few useful links. Enjoy!

1. https://en.wikipedia.org/wiki/MapReduce
2. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
3. https://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.html

Hope you found the article helpful!
Thank You!

Big Data Data Science Machine Learning Hadoop Docker