54- Sahil Kabir

Experiment No.1
Hadoop HDFS Practical
Date of Performance:
Date of Submission:

54- Sahil Kabir

AIM :

Installation, Configuration of Hadoop and performing Basic File Management operations in Hadoop.

THEORY :

Hadoop :

Apache Hadoop is a framework that allows the distributed processing of large data sets across clusters of commodity computers using a simple programming model.
It is an Open-source Data Management with scale-out storage & distributed processing.
The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing.
Hadoop has the capability to handle different modes of data such as structured, unstructured and semi-structured data. It gives us the flexibility to collect, process, and analyse data that our old data warehouses failed to do.

Hadoop Ecosystem:

Hadoop ecosystem is a platform or framework which helps in solving the big data problems. It comprises different components and services (ingesting, storing, analysing, and maintaining) inside of it. Most of the services available in the Hadoop ecosystem are to supplement the main four core components of Hadoop which include HDFS, YARN, MapReduce and Common.

The Hadoop ecosystem includes both Apache Open Source projects and other wide variety of commercial tools and solutions. Some of the well known open source examples include Spark, Hive, Pig, Sqoop and Oozie.

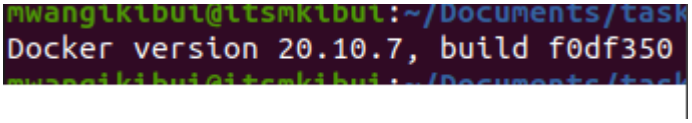
Installation of Hadoop :

To set up this single Hadoop cluster using Docker, ensure that Docker is installed on your computer. Run the following commands to make sure Docker is already set up to run docker-compose

1. To check Docker, run;

```
docker --version
```

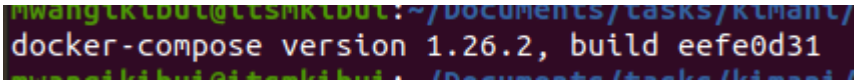
2. If Docker is well set, the output should be similar to;



3. To check docker-compose run;

```
docker-compose --version
```

4. If Docker has docker-compose well set, the output should be similar to;



5. Check whether Docker is working correctly on your system by checking on present running containers if you have any. Run the following command to do so:

```
docker ps
```

6. If you have a running container, it will be logged and listed in the command output. Since I don't have any Docker containers currently on my system, the output will be as follows :

Set up a single Hadoop cluster using docker-compose :

1. Start by cloning this docker-Hadoop repository from Github as follows;

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

The sample repository above has a Hadoop docker-compose.yml set and ready to be deployed to Docker containers. Navigate to the cloned folder, and then run the following command to start the container using docker-compose:

```
docker-compose up -d
```

The docker-compose up will check the containers set in the docker-compose.yml, download them and run them within the Docker engine.

The -d flag will set the container to run in a detachable model, i.e., in the background. After everything is done, you can check the running Hadoop containers using the following command;

```
docker ps
```

2. Check the running Hadoop containerized environment :

To get a visual of a running Hadoop application, you need to get the container IP address. Then test the Hadoop on the browser using the mapped container port. Run this command to get your IP address:

```
ifconfig
```

In the response, your IP is the INET in the second line as follows;

Set up the Hadoop cluster using Docker

From the above example, we have executed the Hadoop cluster using the docker-compose. Alternatively, you can use Docker, run the Hadoop images directly on your Docker engine, and set up a Hadoop cluster.

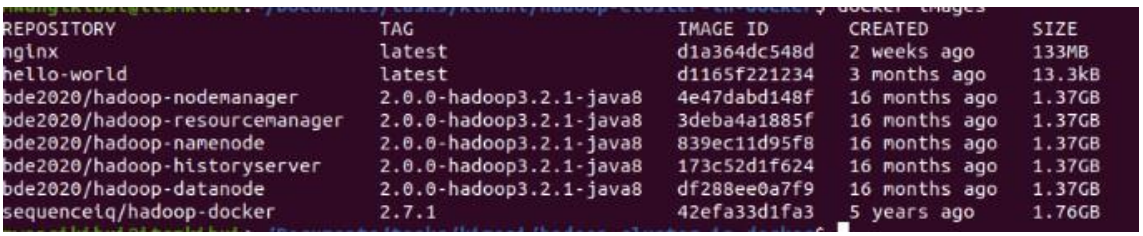
To begin, run the following command to get a Hadoop Docker image from the Docker hub libraries;

```
sudo docker pull sequenceiq/hadoop-docker:2.7.1
```

This will download the Hadoop image with its YARN properties such as the node manager, resource manager, and history server and install it in your computer's Docker engine. Run the below command to see if the Hadoop Docker image was successfully downloaded.

```
docker images
```

If the image was installed successfully, it should be listed in the output as follows;



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	d1a364dc548d	2 weeks ago	133MB
hello-world	latest	d1165f221234	3 months ago	13.3kB
bde2020/hadoop-nodemanager	2.0.0-hadoop3.2.1-java8	4e47dabd148f	16 months ago	1.37GB
bde2020/hadoop-resourcemanager	2.0.0-hadoop3.2.1-java8	3deba4a1885f	16 months ago	1.37GB
bde2020/hadoop-namenode	2.0.0-hadoop3.2.1-java8	839ec11d95f8	16 months ago	1.37GB
bde2020/hadoop-historyserver	2.0.0-hadoop3.2.1-java8	173c52d1f624	16 months ago	1.37GB
bde2020/hadoop-datanode	2.0.0-hadoop3.2.1-java8	df288ee0a7f9	16 months ago	1.37GB
sequenceiq/hadoop-docker	2.7.1	42efa33d1fa3	5 years ago	1.76GB

Let's now build a Hadoop-running Docker container. You can use the following command to create a Hadoop container inside your Docker engine. This creates and runs a single cluster's containers.

```
docker run -it sequenceiq/hadoop-docker:2.7.1 /etc/bootstrap.sh -bash
```

```
Starting sshd:                                             [  OK  ]
21/06/09 03:16:23 WARN util.NativeCodeLoader: Unable to load native-hadoop lib
rary for your platform... using builtin-java classes where applicable
Starting namenodes on [0f4a757b628f]
0f4a757b628f: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root
-namenode-0f4a757b628f.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-da
tanode-0f4a757b628f.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-
root-secondarynamenode-0f4a757b628f.out
21/06/09 03:16:42 WARN util.NativeCodeLoader: Unable to load native-hadoop lib
rary for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemana
ger-0f4a757b628f.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-n
odemanager-0f4a757b628f.out
bash-4.1#
```

If the command is executed without any error (probably due to poor network connections), go ahead and check if Hadoop services are up and running. You can do this by running the jps command :

```
jps
```

```
bash-4.1# jps
654 NodeManager
562 ResourceManager
972 Jps
215 DataNode
124 NameNode
405 SecondaryNameNode
```

You can see that containers are set for NodeManager, DataNode, Resource manager and NameNode.

You can now verify if everything is up and running. On your command terminal, check the currently running containers by the following command;

```
docker ps
```

If your setup is well and running, you will obtain a response similar to;

Testing the Hadoop cluster :

Go over to your terminal tab and run the following command to get the IP address of the running Hadoop Docker container. The IP address will help us to access the Hadoop cluster on the browser. In addition, the local IP address will map to the Hadoop Docker container port number.

Commands :

>> ifconfig

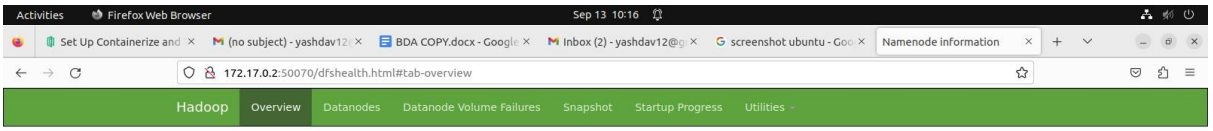
ip-address-response

```
ifconfig
```

```
bash-4.1# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:82 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:12338 (12.0 KiB)  TX bytes:1444 (1.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:3858 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3858 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:551240 (538.3 KiB)  TX bytes:551240 (538.3 KiB)
```

Your IP address will be the INET ADDR value in the third line in the above figure. From your browser, go to: your_ip_address:50070. Make sure you replace your IP address appropriately.



Overview '61a34fb04422:9000' (active)

Started:	Wed Sep 13 00:43:00 EDT 2023
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	CID-5e691286-4de5-4dde-800b-c02a7a8bf44a
Block Pool ID:	BP-1961412683-172.17.0.32-1450036414523

Summary

Security is off.

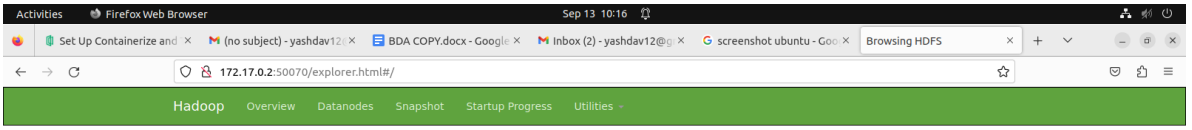
Safe mode is ON. The reported blocks 0 needs additional 31 blocks to reach the threshold 0.9990 of total blocks 31. The number of live datanodes 0 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.

35 files and directories, 31 blocks = 66 total filesystem object(s).

Heap Memory used 83.08 MB of 204 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 28.06 MB of 29.44 MB Commited Non Heap Memory. Max Non Heap Memory is 130 MB.

Configured Capacity:	0 B
DFS Used:	0 B (100%)
Non DFS Used:	0 B

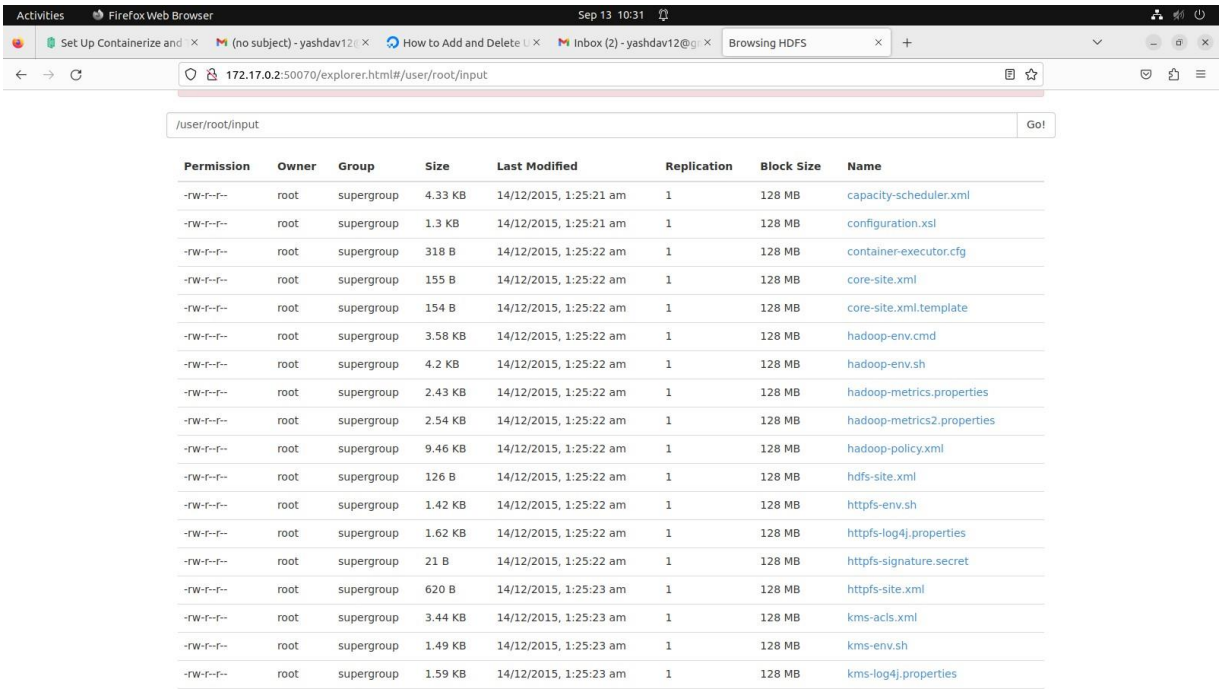


Browse Directory

/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	14/12/2015, 1:24:30 am	0	0 B	user

Hadoop, 2015.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	4.33 KB	14/12/2015, 1:25:21 am	1	128 MB	capacity-scheduler.xml
-rw-r--r--	root	supergroup	1.3 KB	14/12/2015, 1:25:21 am	1	128 MB	configuration.xsl
-rw-r--r--	root	supergroup	318 B	14/12/2015, 1:25:22 am	1	128 MB	container-executor.cfg
-rw-r--r--	root	supergroup	155 B	14/12/2015, 1:25:22 am	1	128 MB	core-site.xml
-rw-r--r--	root	supergroup	154 B	14/12/2015, 1:25:22 am	1	128 MB	core-site.xml.template
-rw-r--r--	root	supergroup	3.58 KB	14/12/2015, 1:25:22 am	1	128 MB	hadoop-env.cmd
-rw-r--r--	root	supergroup	4.2 KB	14/12/2015, 1:25:22 am	1	128 MB	hadoop-env.sh
-rw-r--r--	root	supergroup	2.43 KB	14/12/2015, 1:25:22 am	1	128 MB	hadoop-metrics.properties
-rw-r--r--	root	supergroup	2.54 KB	14/12/2015, 1:25:22 am	1	128 MB	hadoop-metrics2.properties
-rw-r--r--	root	supergroup	9.46 KB	14/12/2015, 1:25:22 am	1	128 MB	hadoop-policy.xml
-rw-r--r--	root	supergroup	126 B	14/12/2015, 1:25:22 am	1	128 MB	hdfs-site.xml
-rw-r--r--	root	supergroup	1.42 KB	14/12/2015, 1:25:22 am	1	128 MB	httpfs-env.sh
-rw-r--r--	root	supergroup	1.62 KB	14/12/2015, 1:25:22 am	1	128 MB	httpfs-log4j.properties
-rw-r--r--	root	supergroup	21 B	14/12/2015, 1:25:22 am	1	128 MB	httpfs-signature.secret
-rw-r--r--	root	supergroup	620 B	14/12/2015, 1:25:23 am	1	128 MB	httpfs-site.xml
-rw-r--r--	root	supergroup	3.44 KB	14/12/2015, 1:25:23 am	1	128 MB	kms-acls.xml
-rw-r--r--	root	supergroup	1.49 KB	14/12/2015, 1:25:23 am	1	128 MB	kms-env.sh
-rw-r--r--	root	supergroup	1.59 KB	14/12/2015, 1:25:23 am	1	128 MB	kms-log4j.properties

If everything worked correctly, you should receive a Hadoop UI on your browser.

CONCLUSION :

In our Hadoop HDFS practical, we implemented a distributed file system to store and manage large volumes of data efficiently. We witnessed the scalability and fault tolerance advantages of HDFS while performing data operations. This hands-on experience underscored the importance of Hadoop HDFS as a foundational component in big data processing, laying the groundwork for further data analysis and insights.