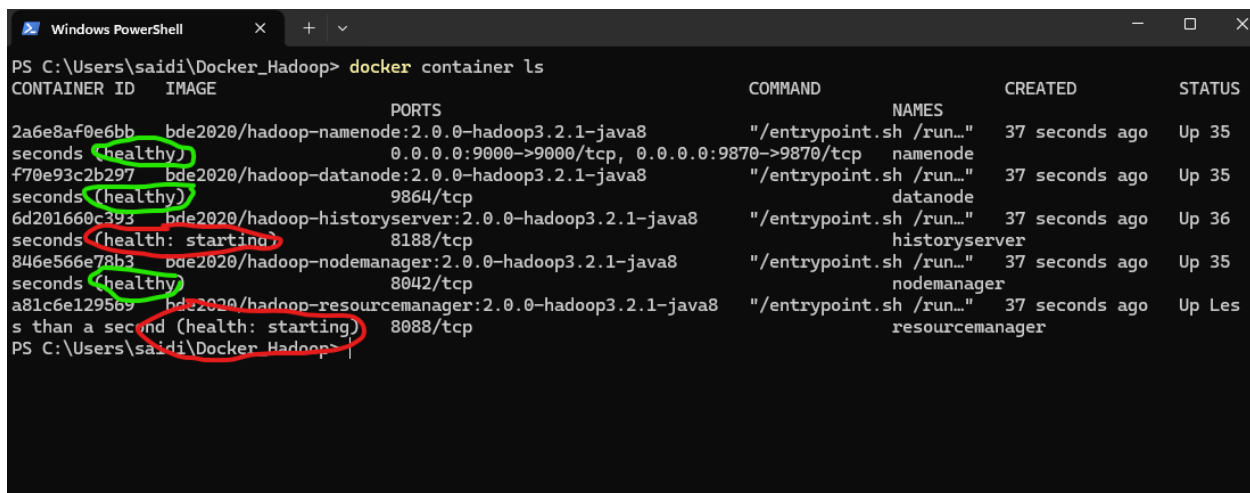


Hadoop Presentation Day – 1 Detailed Notes

- By the point of reading this document, I hope everyone has installed hadoop using docker and execute “**docker compose up -d**” having the .yaml file inside the path folder under terminal.
- Make sure that all the nodes are up and running using:
 - **docker container ls**



```
PS C:\Users\saidi\Docker_Hadoop> docker container ls
CONTAINER ID   IMAGE                                PORTS                                COMMAND                                NAMES                                CREATED      STATUS
2a6e8af0e6bb   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8  0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp  "/entrypoint.sh /run..."          namenode                                37 seconds ago Up 35
seconds (healthy)
f70e93c2b297   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8  9864/tcp                               "/entrypoint.sh /run..."          datanode                                37 seconds ago Up 35
seconds (healthy)
6d201660c393   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8  8188/tcp                               "/entrypoint.sh /run..."          historyserver                            37 seconds ago Up 36
seconds (health: starting)
846e566e78b3   bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8  8042/tcp                               "/entrypoint.sh /run..."          nodemanager                             37 seconds ago Up 35
seconds (healthy)
a81c6e129569   bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8  8088/tcp                               "/entrypoint.sh /run..."          resourcemanager                         37 seconds ago Up Les
s than a second (health: starting)
```

As it can be seen in the above picture make sure every node is “healthy” and if it still showing “health: starting” give it some time and you will find it changes to “healthy” once you rerun the “**docker container ls**” command.









- Once every node is healthy and running now you can work on your “nodename” container using:
 - **docker exec -it namenode bash** (or)
 - **docker exec -i -t namenode bash** (they mean the same)

- After Executing that you will find yourself operating the terminal under the container namenode (Note: This is not an hdfs file system it is the container namenode that contains an hdfs file system which we can access)

```
PS C:\Users\saidi\Hadoop> docker exec -i -t namenode bash
root@2a6e8af0e6bb:/# |
```

After you see the above screen in the terminal it means you are inside the container “2a6e8af0e6bb” (containerId) as “root” user

- Now you can access your Hdfs file system and its contents in two ways:
 - **Way – 1 (Through the namenode server url that can be found in docker as follows):**

	nodemanager bde2020/hadoop-no Running	■	⋮	🗑
	datanode bde2020/hadoop-da Running	■	⋮	🗑
	historyserver bde2020/hadoop-his Running	■	⋮	🗑
	namenode bde2020/hadoop-na Running 9000:9000  9870:9870  9870:9870  Show less	■	⋮	🗑
	resourcemanager bde2020/hadoop-res Running	■	⋮	🗑

-
- Once you click on that link or navigate to <http://localhost:9870/> you will find yourself in following web page (**Default port that the webserver runs is 9870**)

Overview 'namenode:9000' (active)

Started:	Tue Mar 19 16:53:31 -0500 2024
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 10:56:00 -0500 2019 by rohithsharmaks from branch-3.2.1
Cluster ID:	CID-5c1c3661-52b5-4e95-934f-325132c6389e
Block Pool ID:	BP-716078944-172.23.0.6-1709243821843

- So, The way to access the hdfs file system is by clicking on the “utilities” dropdown and selecting “browse the file system” option. And now you can see following screen:

Browse Directory

/

Go!

Show

25

entries

Search

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxr-xr-x	root	root	0 B	Feb 29 16:06	0	0 B	app-logs	
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Feb 29 15:57	0	0 B	rmstate	
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Feb 29 16:06	0	0 B	tmp	
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Mar 19 11:23	0	0 B	user	

Showing 1 to 4 of 4 entries

Previous

1

Next

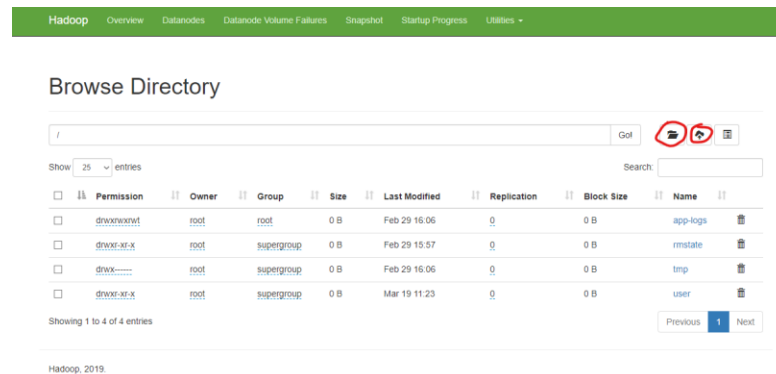
Hadoop, 2019

Hadoop, 2019.

So as you can see we are able to see the hdfs file system hierarchy and contents of the file system (**Note: You might not be able to see all the files shown in the picture as they haven't been created yet**)

- To create a folder or upload a file using this method is simple just click on either of these two things as per your requirements (either to create a folder or upload a file from

your machine).



○ **Way – 2 (Accessing hdfs using the terminal from the namenode container):**

- So when you are in the container namenode as a root user and your terminal looks like this:

```
PS C:\Users\saidi\Docker_Hadoop> docker exec -i -t namenode bash
root@2a6e8af0e6bb:/# |
```

- You can perform all linux operations on the machine like:
 - cd – change directory
 - ls – list the contents of the directory/path.
 - mkdir – to make a directory.
 - rm – to remove a file.
 - rm -rf to remove a directory that is not empty.
 - **And many more....**
- So, lets see how to access hadoop file system and create a file inside the hdfs instead of creating it inside the namenode container that we are currently in:
 - To access the hadoop file system and perform any operation the default command syntax is as follows:
 - **hdfs dfs –[command] [arguments]**

So, I hope everyone is clear until this part and gone through all the basic Linux commands. If not, don't worry, follow these next steps as well and you will get practical experience in what we are doing.

Word Count using Hadoop Map Reduce

- **Step-1: Creating an User Folder in container and hdfs**
 - So, Let us create a new folder in both container and hdfs. So we get a better understanding of **how to create a folder in the container vs how to create a folder in hdfs of the container**.
 - Once you are in the container as root user:
 - Perform 'ls' operation to view the contents of that folder you are in –

```
root@2a6e8af0e6bb:/# ls
KEYS  boot  entrypoint.sh  hadoop      home  lib64  mnt  proc  run  sbin  sys  usr
bin   dev   etc            hadoop-data lib  media  opt  root  run.sh  srv  tmp  var
root@2a6e8af0e6bb:/#
```

- As, it can be seen in the above picture now we don't have any "user" folder in the list of contents so let us make one using:

mkdir <yourName> (replace <yourName> with your name or any name your prefer your folder to be). In my case I have done "**mkdir dinesh**" and performed "**ls**" operation again and I am able to see my folder in the contents list now

```
root@2a6e8af0e6bb:/# mkdir dinesh
root@2a6e8af0e6bb:/# ls
KEYS  boot  dinesh  etc      hadoop-data  lib  media  opt  root  run.sh  srv  tmp  var
bin   dev   entrypoint.sh  hadoop  home      lib64  mnt  proc  run  sbin  sys  usr
root@2a6e8af0e6bb:/# |
```

- Now you have created a user folder in the container namenode. Now lets do the same command with little addition to create a folder inside hdfs. To check the contents of hdfs initially before trying to create a folder you can perform:

hdfs dfs -ls /

(Note that the difference between the ls command used in the container namenode's bash shell and the above command is the addition of "hdfs dfs -" and a "/" after the ls command so what does they mean?

hdfs – states that we are performing an operation on hdfs

dfs – states we are accessing the distributed file system of hdfs

- - States that we are performing this following command on specified path inside hdfs

/ - home path of hdfs

So all together combined "**hdfs dfs -ls /**" Will list the contents of the home directory of the hdfs present inside the namenode container

In the same way we perform the operations on hdfs using

hdfs dfs -[command] [options] [arguments]

options: options differ from command to commands for example:

in linux command "**ls -l**" **[-l]** is an option that tells to list all the contents in a detailed view. Detailed view contains following information:

<filetype><permissions>	<numberOfHardLinks>	<owner>	<group>
<fileSizeinBytes>	<LastModifiedDate>	<file/directory name>	

It looks like this:

```

root@2a6e8af0e6bb:/# ls -l
total 416
-rw-r--r-- 1 root root 325083 Feb  4 2020 KEYS
drwxr-xr-x 1 root root  4096 Feb  4 2020 bin
drwxr-xr-x 2 root root  4096 Sep  8 2019 boot
drwxr-xr-x 5 root root   340 Mar 19 21:53 dev

```

- Coming back to where we left off “**hdfs dfs -ls /**”. Once you perform this operation in bash inside your name node container you will find the list of contents in the home directory of hdfs in your namenode.
- **Create a user folder in hdfs using:**
 - **hdfs dfs -mkdir /user/<yourName>**
- Create two folders **Input**, **Output** that contains our input file for the hadoop job and output file that generates after job execution
 - **hdfs dfs -mkdir /user/<yourName>/Input**
 - **hdfs dfs -mkdir /user/<yourName>/Output**
- Now when you run “**hdfs dfs -ls /user/<yourName>/**”. Your output looks like follows:

```

root@2a6e8af0e6bb:/# hdfs dfs -ls /user/saidinesh
Found 2 items
drwxr-xr-x - root supergroup 0 2024-03-19 22:58 /user/saidinesh/Input
drwxr-xr-x - root supergroup 0 2024-03-19 22:58 /user/saidinesh/Output

```

By this time, You might have learnt how to create a folder in the namenode container and the hdfs that the container holds.

Now, let us try to copy the file from your local machine to the container as well the hdfs it holds:

Copying a File from Local Machine to Container/hdfs:

- First come back to operating the terminal on your local machine either using “**Ctrl + z**” or “**Cmd + z**”. if either of them don’t work, Use “exit” command as follows:

```
root@2a6e8af0e6bb:/# exit
exit
PS C:\Users\saidi\Docker_Hadoop> |
```

- Now you will find yourself back in your local machine. Now create any text file or you can use the data.txt file under the repository we have posted and it is under path as follows:
<Path_to_the_Repository>/code/Hadoop_code/input/**data.txt**
- Create another text file “**manifest.txt**” that contains “Main-Class:WordCount” as its content and keep it aside.

- **FIRST_STEP: MOVING FILE INTO THE CONTAINER**

- Perform “**docker cp <path_to_data.txt> namenode:<yourName>**”
 - yourName = folder you created initially in container
- do “**docker cp <path_to_WordCount.java> namenode:<yourName>**”
 - path to WordCount =
 <path_to_repository>/code/Hadoop_Code/HadoopWordCount/src/WordCount.java
- do the same to copy the manifest file using “**docker cp <path_to_manifest.txt> namenode:<yourName>**”
- once these three operations are done go back to the container node using
 - **docker exec -it namenode bash**
- check the copied files are present using “**ls <yourName> /**”

```
root@2a6e8af0e6bb:/# ls dinesh/
WordCount.java  data.txt  manifest.txt
root@2a6e8af0e6bb:/# |
```

You will see this output

- **SECOND_STEP: MOVING FILE FROM CONTAINER INTO HDFS**

- To move the file from the local machine of container to hdfs we use **-put** command of hdfs as follows:
 - **hdfs dfs -put <path_to_file> <path_to_destination>**
- so perform this operation and send the data.txt inside the hdfs folder we created as “**Input**” as it is the data we are using to perform mapReduce function on

- **hdfs dfs -put /<yourName>/data.txt /user/<yourName>/Input/**
 - **/user/<yourName>/Input/** = hdfs path of created Input folder it might vary based on your path you created the folder in
- Once this is done perform “**hdfs dfs -ls /user/<yourName>/Input**” and your output should be as follows:

```
root@2a6e8af0e6bb:/dinesh# hdfs dfs -put data.txt /user/saidinesh/Input
2024-03-19 23:30:01,859 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localTrusted = false
root@2a6e8af0e6bb:/dinesh# hdfs dfs -ls /user/saidinesh/Input
Found 1 items
-rw-r--r--  3 root supergroup      1666 2024-03-19 23:30 /user/saidinesh/Input/data.txt
```

- Now run the following commands to perform the MapReduce function and view the output (make sure you are in the folder containing all the files we copied from our local system using “ls” command):
 - **javac -classpath \$HADOOP_HOME/share/hadoop/common/hadoop-common-3.2.1.jar:\$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.1.jar -d . WordCount.java**
 - **jar cvmf manifest.txt wc.jar *.class**
 - **hadoop jar wc.jar WordCount /user/<yourName>/Input/data.txt /user/<yourName>/Output/WordCountOutput**
 - after job is successfully done
 - **hdfs dfs -cat /user/<yourName>/Output/WordCountOutput/part-r-00000**
 - You will see the output of mapReduce

Will be revisiting the topic presentation from this point of creating a jar file. Any doubts or clarifications will be answered. 😊

Thank you.