1. Table of Contents

- Jdk : jdk-11.0.16_linux-x64_bin.tar and jdk-8u202-linux-x64.tar
- Hadoop: hadoop-3.1.2.tar.gz
- Hive - apache-hive-3.1.2-bin.tar (Compatible with 3.1.2 hadoop only)

- Apache Derby - 10.14.1.0

Lab-Zookeeper.pdf
Last Modified Date : 20Feb 2022

https://archive.apache.org/dist/hadoop/common/hadoop-3.1.2/
https://mirrors.estointernet.in/apache/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz
https://downloads.apache.org//db/derby/db-derby-10.14.2.0/db-derby-10.14.2.0-bin.tar.gz

2. Prerequisite.

Infrastructure :

You required two server instances. OS should be any Linux of 8 – Centos preferable.

Options:

- VM – using 2 VMs.
- Docker - Create two containers.

For illustration, it has been stated below. Ensure to follow the same nomencleature to avoid confusion. Its very important.

Hostname	Roles	Services
hadoop0	Master	HDFS & YARN Services
hadoop1	Slave	HDFS & YARN Services.

Note: Refer any online document to create VM or install docker in your environment.

By now, you should have two VMs on your workstation or Two containers in a docker environment:

Host/VM	Remarks	Purpose
hp.com	Hdfs & YARN Services	Master
ht.com	Hdfs & YARN Services	Slave

hostname	Container	Remarks
Hadoop0	Hadoop0	HDFS & YARN Services
Hadoop1	Hadoop1	HDFS & YARN Services.

Determine the ips of both the servers.
Ensure that both machines are reachable to each other using IP and Hostname.
Verify the hostname and the /etc/hosts. You need to enter each IP and host name as shown above.
Update details accordingly in your local machine.

```
[root@hp ~]# hostname
hp.com
[root@hp ~]# more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
#127.0.0.1               hp.com localhost
#::1                     hp.com localhost6

192.168.188.134 ht.com
192.168.188.136 hp.com localhost
[root@hp ~]#
```

You should see in your window as follows for ht.com. Changes the IP with that of yours system.

```
[root@ht ~]# bash
[root@ht ~]# hostname
ht.com
[root@ht ~]# more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
#127.0.0.1               localhost.localdomain localhost
#::1             localhost6.localdomain6 localhost6
192.168.188.134 ht.com localhost
192.168.188.136 hp.com
[root@ht ~]#
```

Using Docker:

Create a common network.

46b42fc00055 spark-net bridge local

Container – 0 – master Node

#docker run -it --name hadoop0 --network spark-net -v /Users/henrypotsangbam/Documents/Software:/Software --privileged -p 8088:8088 -p 9870:9870 -p 9864:9864 -p 8032:8032 -p 8188:8188 -p 8020:8020 -p 4040:4040 --hostname hadoop0 centos /usr/sbin/init

Container – 1 – Slave Node

#docker run -it --name hadoop1 --network spark-net -v /Users/henrypotsangbam/Documents/Software:/Software --privileged -p 8089:8088 -p 9871:9870 -p 9865:9864 -p 8033:8032 -p 8189:8188 -p 8022:8020 -p 4041:4040 --hostname hadoop1 centos /usr/sbin/init

Container – 2 – Slave Node

#docker run -it --name hadoop2 --network spark-net -v /Users/henrypotsangbam/Documents/Software:/Software --privileged -p 8090:8088 -p 9872:9870 -p 9866:9864 -p 8034:8032 -p 8190:8188 -p 8023:8020 -p 4042:4040 --hostname hadoop2 centos /usr/sbin/init

using Docker Ends.

yum install passwd
Change the root password
passwd

root123!

Datafolder clean scripts.

#vi clean.sh

```
rm -fr /opt/hdfs/namenode/*
rm -fr /opt/hdfs/datanode/*
rm -fr /opt/yarn/*
rm -fr /opt/journal/*
```

This script can be used to clean the data folders of cluster.

3. Launching a Cluster: HDFS – 180 Minutes (D)

In this lab, you will deploy a two nodes Hadoop Cluster.

Configure and install HDFS.

All the below commands should be executed on hp.com/ master node unless specify.

We are configuring HDFS cluster now.

Change directory to the location where you have the software or download from the following.

Extract the hadoop file as follows:

```
#tar -xvf hadoop-X.tar.gz -C /opt
#tar -xvf jdk-11* -C /opt
```

Rename the folder:

#mv hadoo* hadoop
#mv jdk* jdk

Configure JDK and set Java Home.

To include JAVA_HOME for all bash users , make an entry in /etc/profile.d as follows:

echo "export JAVA_HOME=/opt/jdk/" > /etc/profile.d/java.sh

In the distribution, edit the file /opt/hadoop/etc/hadoop/hadoop-env.sh to define some parameters as follows:

```
# set to the root of your Java installation
 export JAVA_HOME=/opt/jdk
```

[Update bash file:

vi ~/.bashrc

```
export JAVA_HOME=/opt/jdk
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

]
```

Try the following command: It should work.

```
$ hadoop
```

Create some folders required for the configuration.

```
mkdir  -p /opt/hdfs/namenode
mkdir  -p /opt/hdfs/datanode
mkdir -p /opt/yarn/local
```

You need to modify some setting as follows: Replace with your hostname of the master Node accordingly.

Use the following, all files will be inside the HADOOP_HOME:

```
# cd /opt/hadoop

#vi etc/hadoop/core-site.xml

<configuration>
   <property>
      <name>fs.defaultFS</name>
      <value>hdfs://hadoop0:8020</value>
   </property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
<description>Buffer size</description>
</property>
</configuration>
```

#vi etc/hadoop/hdfs-site.xml

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///opt/hdfs/namenode</value>
<description>NameNode directory for namespace and transaction logs storage.</description>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:///opt/hdfs/datanode</value>
<description>DataNode directory</description>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.datanode.use.datanode.hostname</name>
```

<value>false</value>
</property>
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
<value>false</value>
</property>
</configuration>

Now verify that you can **ssh** to the localhost without a passphrase:
Install the ssh package if not done earlier.

```
#yum -y install openssh-server openssh-clients
#systemctl start sshd
```

Setup passphraseless ssh

Change the passwd using passwd command.

```
$ ssh localhost
```

If you cannot ssh to localhost without a passphrase, execute the following commands:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Paraphrase Section Ends Here --------

Export some variables. You can add in the **~/.bashrc** profile.

```
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"
```

Format the HDFS file system: -

Only on the Master node (hadoop0) and It has to be executed only once.

```
$ hdfs namenode -format
```

```
2021-04-05 10:16:51,093 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2021-04-05 10:16:51,093 INFO util.GSet: VM type       = 64-bit
2021-04-05 10:16:51,095 INFO util.GSet: 0.029999999329447746% max memory 876.5 MB = 269.3 KB
2021-04-05 10:16:51,095 INFO util.GSet: capacity      = 2^15 = 32768 entries
2021-04-05 10:16:51,144 INFO namenode.FSImage: Allocated new BlockPoolId: BP-2126441114-172.18.0.3-1617617811131
2021-04-05 10:16:51,162 INFO common.Storage: Storage directory /opt/hdfs/namenode has been successfully formatted.
2021-04-05 10:16:51,207 INFO namenode.FSImageFormatProtobuf: Saving image file /opt/hdfs/namenode/current/fsimage.ckpt_0000000000000
00000 using no compression
2021-04-05 10:16:51,366 INFO namenode.FSImageFormatProtobuf: Image file /opt/hdfs/namenode/current/fsimage.ckpt_0000000000000000000 o
f size 396 bytes saved in 0 seconds .
2021-04-05 10:16:51,381 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2021-04-05 10:16:51,389 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2021-04-05 10:16:51,389 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at hadoop0/172.18.0.3
************************************************************/
[root@hadoop0 hadoop]#
```

Update- workers file as follow (/opt/hadoop/etc/hadoop/workers)

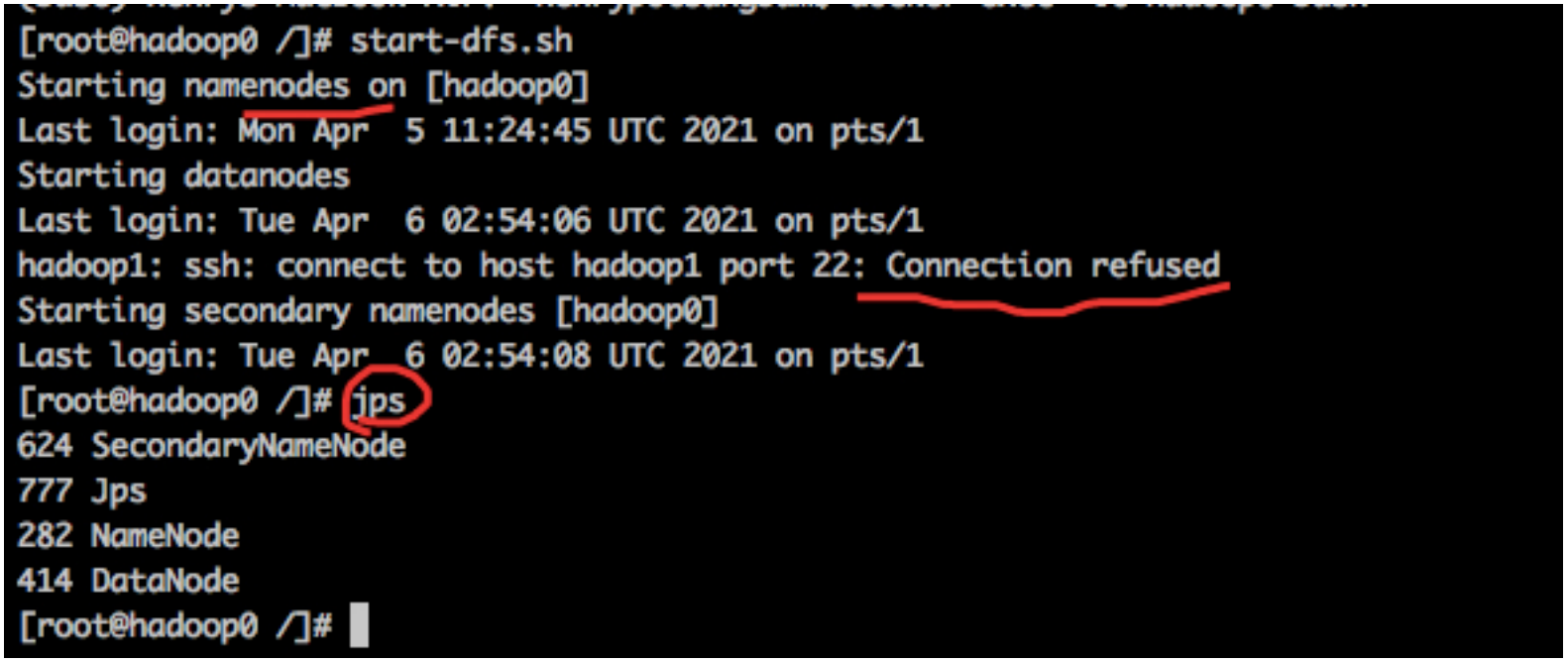
hadoopo

On Master Node only - Start NameNode daemon and DataNode daemon with the following command:

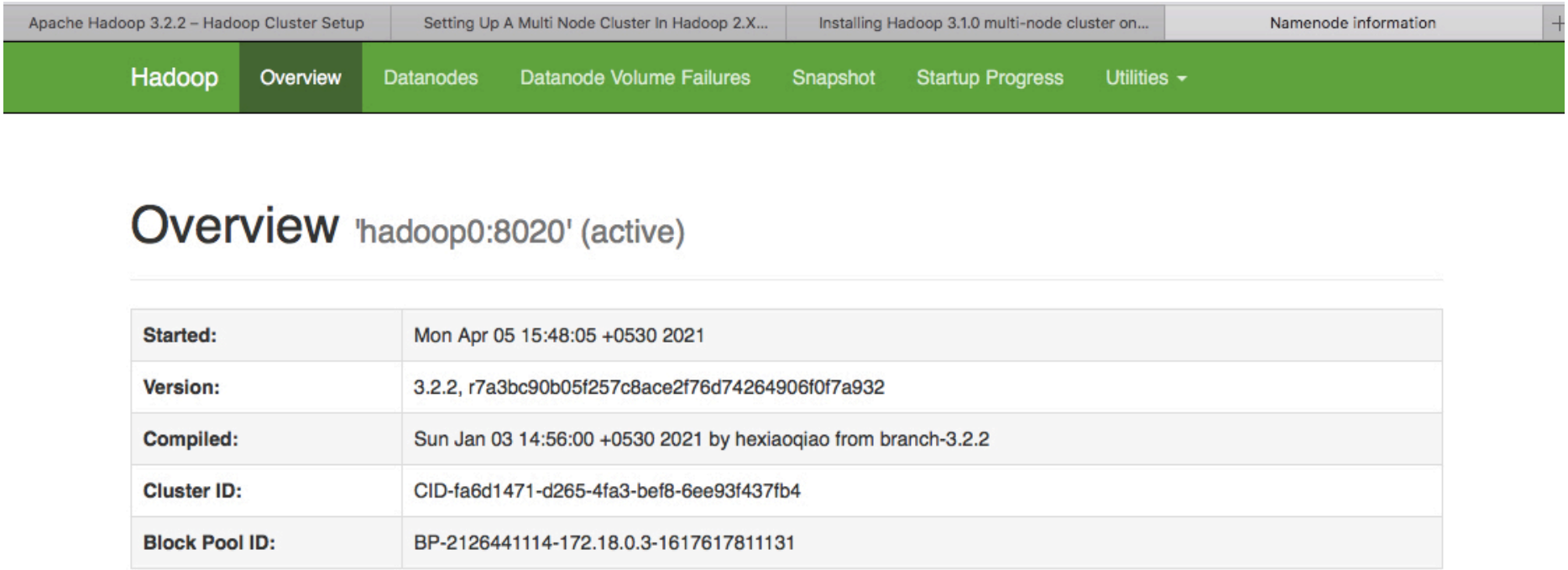
$ start-dfs.sh

Verify the services using

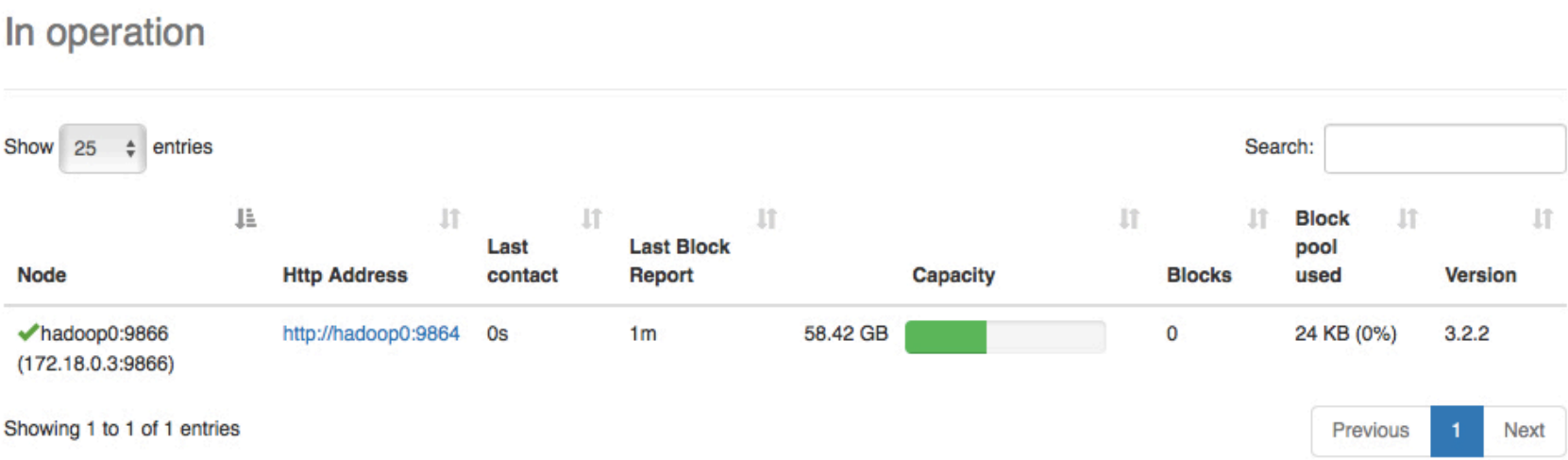
#jps



1. Browse the web interface for the NameNode; by default, it is available at:
    - NameNode - http://localhost:9870/



Click on DataNodes:



You should have only One Data Node.

Add Node

Let us add another Node ie hadoop1

Perform the following in the second node or in all the slave node.

extract the hadoop file as follows:

```
#tar -xvf hadoop-*.tar -C /opt
#tar -xvf jdk-11* -C /opt
```

Rename the folder:

#mv hadoo* hadoop
#mv jdk* jdk

Set Java Home.
To include JAVA_HOME for all bash users , make an entry in /etc/profile.d as follows:

echo "export JAVA_HOME=/opt/jdk/" > /etc/profile.d/java.sh

In the distribution, edit the file /opt/hadoop/etc/hadoop/hadoop-env.sh to define some parameters as follows:

```
# set to the root of your Java installation
 export JAVA_HOME=/opt/jdk
```

[Update - vi ~/.bashrc

```
export JAVA_HOME=/opt/jdk
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

]

Try the following command: It should work.

```
$ hadoop
```

Create some folders required for the configuration.

```
mkdir  -p /opt/hdfs/namenode
mkdir  -p /opt/hdfs/datanode
mkdir -p /opt/yarn/local
```

You need to modify some setting as follows: Replace with your hostname of the master Node accordingly.

```
# cd /opt/hadoop
```

Use the following, all files will be inside the HADOOP_HOME:

```
#vi etc/hadoop/core-site.xml
```

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
```

```
      <value>hdfs://hadoop0:8020</value>
   </property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
<description>Buffer size</description>
</property>
</configuration>
```

#vi etc/hadoop/hdfs-site.xml

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///opt/hdfs/namenode</value>
<description>NameNode directory for namespace and transaction logs storage.</description>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:///opt/hdfs/datanode</value>
<description>DataNode directory</description>
</property>
<property>
```

```
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.datanode.use.datanode.hostname</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
<value>false</value>
</property>
</configuration>
```

Install the necessary software required for ssh.

```
#yum -y install openssh-server openssh-clients
#systemctl start sshd
```

Export some variables. You can add in the ~/.bashrc profile.

#vi ~/.bashrc

```
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"
```

Slave nodes installation ends here.

Execute on the second Node (hadoop1) or on all slave nodes. It starts only the data node services.

#hdfs --daemon start datanode

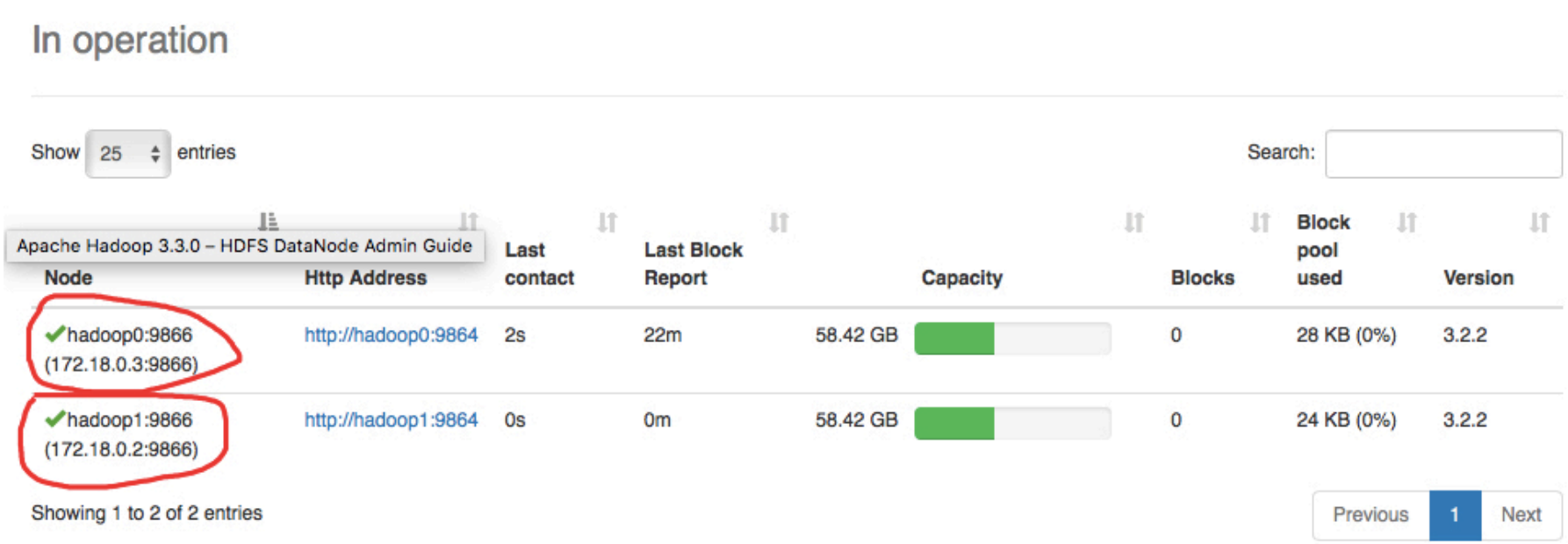
Or

#hadoop-daemon.sh start datanode

#jps

```
[root@hadoop1 sbin]# hadoop-daemon.sh start datanode
WARNING: Use of this script to start HDFS daemons is deprecated.
WARNING: Attempting to execute replacement "hdfs --daemon start" instead.
WARNING: /opt/hadoop/logs does not exist. Creating.
[root@hadoop1 sbin]# jps
744 DataNode
777 Jps
[root@hadoop1 sbin]#
```

Now refresh the HDFS console.



Make the HDFS directories required to execute MapReduce jobs:

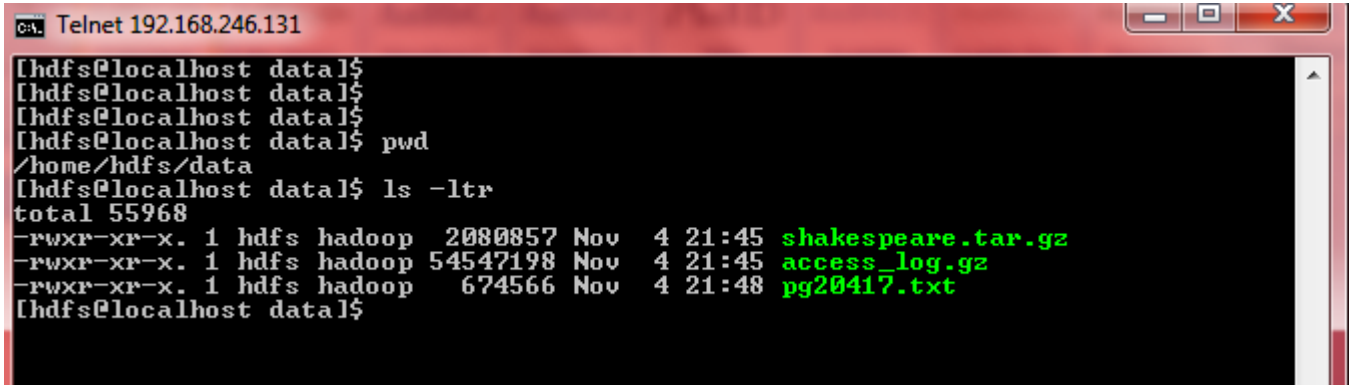
```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/root
```

You have successfully configured a Two nodes Hadoop Cluster - HDFS.
---------- Lab Ends Here ----------

4. Exploring HDFS – 90 Minutes(D)

All Data files (local) need to copy in your VM. All exercise needs to be performed using root logon unless specified. You can create a data folder in your home directory and dump all data inside that folder.

/software/data/shakespeare.tar
/software/data/access_log
/software/data/pg20417.txt



Hadoop is already installed, configured, and running on your machine. Most of your interaction with the system will be through a command--line wrapper called ***hdfs***. If you run this program with no arguments, it prints a help message. To try this, run the following command in a terminal window:

```
$ hdfs
```

The hdfs command is subdivided into several subsystems. For example, there is a subsystem for working with files in HDFS and another for launching and managing MapReduce processing jobs.

Exploring HDFS

The subsystem associated with HDFS in the Hadoop wrapper program is called FsShell. This subsystem can be invoked with the command hadoop fs.

Open a terminal window (if one is not already open) by double--clicking the Terminal icon on the desktop.
In the terminal window, enter:

```
# hdfs
```

You see a help message describing all the commands associated with the FsShell subsystem.
Enter:

```
#hdfs dfs -ls /
```

This shows you the contents of the root directory in HDFS. There will be multiple entries, one of which is /user. Individual users have a "home" directory under this directory, named after their username; your username in this course is hdfs, therefore your home directory is /user/hdfs.

Try viewing the contents of the /user directory by running:

```
# hdfs dfs -ls /user
```

You will see your home directory in the directory listing.

List the contents of your home directory by running:

```
$ hdfs dfs -ls /user/root
```

Note that the directory structure in HDFS has nothing to do with the directory structure of the local filesystem; they are completely separate namespaces.

Uploading Files

Besides browsing the existing filesystem, another important thing you can do with FsShell is to upload new data into HDFS. Change directories to the local filesystem directory containing the sample data we will be using in the homework labs.

```
$ cd /Software
```

If you perform a regular Linux ls command in this directory, you will see a few files, including two named shakespeare.tar.gz and shakespeare-stream.tar.gz. Both of these contain the complete works of Shakespeare in text format, but with different formats and organizations. For now we will work with shakespeare.tar.gz.

Unzip shakespeare.tar by running with root credentials(su root):

```
$ tar xvf shakespeare.tar
```

This creates a directory named shakespeare/ containing several files on your local filesystem.

copy this directory into HDFS using hdfs:

```
$ hdfs dfs -put shakespeare /user/root/shakespeare
```

This copies the local shakespeare directory and its contents into a remote, HDFS directory named /user/root/shakespeare.

List the contents of your HDFS home directory now:

```
$ hdfs dfs -ls /user/root
```

You should see an entry for the shakespeare directory.

Now try the same fs -ls command but without a path argument:

```
$ hdfs dfs -ls
```

You should see the same results. If you don't pass a directory name to the –ls command, it assumes you mean your home directory, i.e. /user/root.

Relative paths
If you pass any relative (non-absolute) paths to FsShell commands (or use relative paths in MapReduce programs), they are considered relative to your home directory.

We will also need a sample web server log file, which we will put into HDFS for use later.

First, create a directory in HDFS in which to store it:

$ hdfs dfs -mkdir weblog
# hdfs dfs -put access_log weblog/

#hdfs dfs -put - weblog/access_log

Run the **hdfs dfs -ls** command to verify that the log file is in your HDFS home directory.

hdfs dfs -ls

Viewing and Manipulating Files
Now let's view some of the data you just copied into HDFS.
Enter:

```
$ hdfs dfs -ls shakespeare
```

This lists the contents of the /user/root/shakespeare HDFS directory, which consists of the files comedies, glossary, histories, poems, and tragedies.
The glossary file included in the compressed file you began with is not strictly a work of Shakespeare, so let's remove it:

```
$ hdfs dfs -rm shakespeare/glossary
```

Note that you *could* leave this file in place if you so wished. If you did, then it would be included in subsequent computations across the works of Shakespeare, and would skew your results slightly. As with many real--world big data problems, you make trade--offs between the labor to purify your input data and the precision of your results.

Enter:

```
$ hdfs dfs -cat shakespeare/histories | tail -n 50
```

This prints the last 50 lines of *Henry IV, Part 1* to your terminal. This command is handy for viewing the output of MapReduce programs. Very often, an individual output file of a MapReduce

program is very large, making it inconvenient to view the entire file in the terminal. For this reason, it's often a good idea to pipe the output of the fs -cat command into head, tail, more, or less.

To download a file to work with on the local filesystem use the fs -get command. This command takes two arguments: an HDFS path and a local path. It copies the HDFS contents into the local filesystem:

```
$ hdfs dfs -get shakespeare/poems ~/shakepoems.txt
$ less ~/shakepoems.txt
```

There are several other operations available with the hdfs fs command to perform most common filesystem manipulations: mv, cp, mkdir, etc.

```
$ hdfs dfs
```

This displays a brief usage report of the commands available within FsShell. Try playing around with a few of these commands if you like.

Basic Hadoop Filesystem commands (Optional)

In order to work with HDFS you need to use the hdfs dfs command. For example to list the / and /app directories you need to input the following commands:

```
hdfs dfs -ls /
hdfs dfs -ls /tmp
```

There are many commands you can run within the Hadoop filesystem. For example to make the directory *test* you can issue the following command:

```
#hdfs dfs -mkdir test
```

Now let's see the directory we've created:

```
hdfs dfs -ls /
hdfs dfs -ls /user/root
```

You should be aware that you can pipe (using the | character) any HDFS command to be used with the Linux shell. For example, you can easily use *grep* with HDFS by doing the following:

```
hdfs dfs -mkdir /user/root/test2
hdfs dfs -ls /user/root | grep test
```

As you can see the grep command only returned the lines which had *test* in them (thus removing the "Found x items" line and oozie-root directory from the listing.

In order to move files between your regular linux filesystem and HDFS you will likely use the *put* and *get* commands. First, move a single file to the hadoop filesystem.
Copy pg20417.txt from software folder to data folder

```
hdfs dfs -put /Software/data/pg20417.txt pg20417.txt
hdfs dfs -ls /user/root
```

You should now see a new file called /user/hdfs/pg* listed. In order to view the contents of this file we will use the *-cat* command as follows:

```
 hdfs dfs -cat pg20417.txt
```

We can also use the linux *diff* command to see if the file we put on HDFS is actually the same as the original on the local filesystem. You can do this as follows:

```
# yum install diff
 #diff <( hdfs dfs -cat pg20417.txt) /Sofware/data/pg20417.txt
```

Since the diff command produces no output we know that the files are the same (the diff command prints all the lines in the files that differ).

Some more Hadoop Filesystem commands

In order to use HDFS commands recursively generally you add an "r" to the HDFS command (In the Linux shell this is generally done with the "-R" argument) For example, to do a recursive listing we'll use the -lsr command rather than just -ls. Try this:

```
hdfs dfs -ls /user
hdfs dfs -ls -R /user
```

In order to find the size of files you need to use the -du or -dus commands. Keep in mind that these commands return the file size in bytes. To find the size of the pg20417.txt file use the following command:

```
hdfs dfs -du pg20417.txt
```

To find the size of all files individually in the /user/root directory use the following command:

```
hdfs dfs -du /user/root
```

To find the size of all files in total of the /user/root directory use the following command:

```
hdfs dfs -dus /user/root
```

If you would like to get more information about a given command, invoke -help as follows:

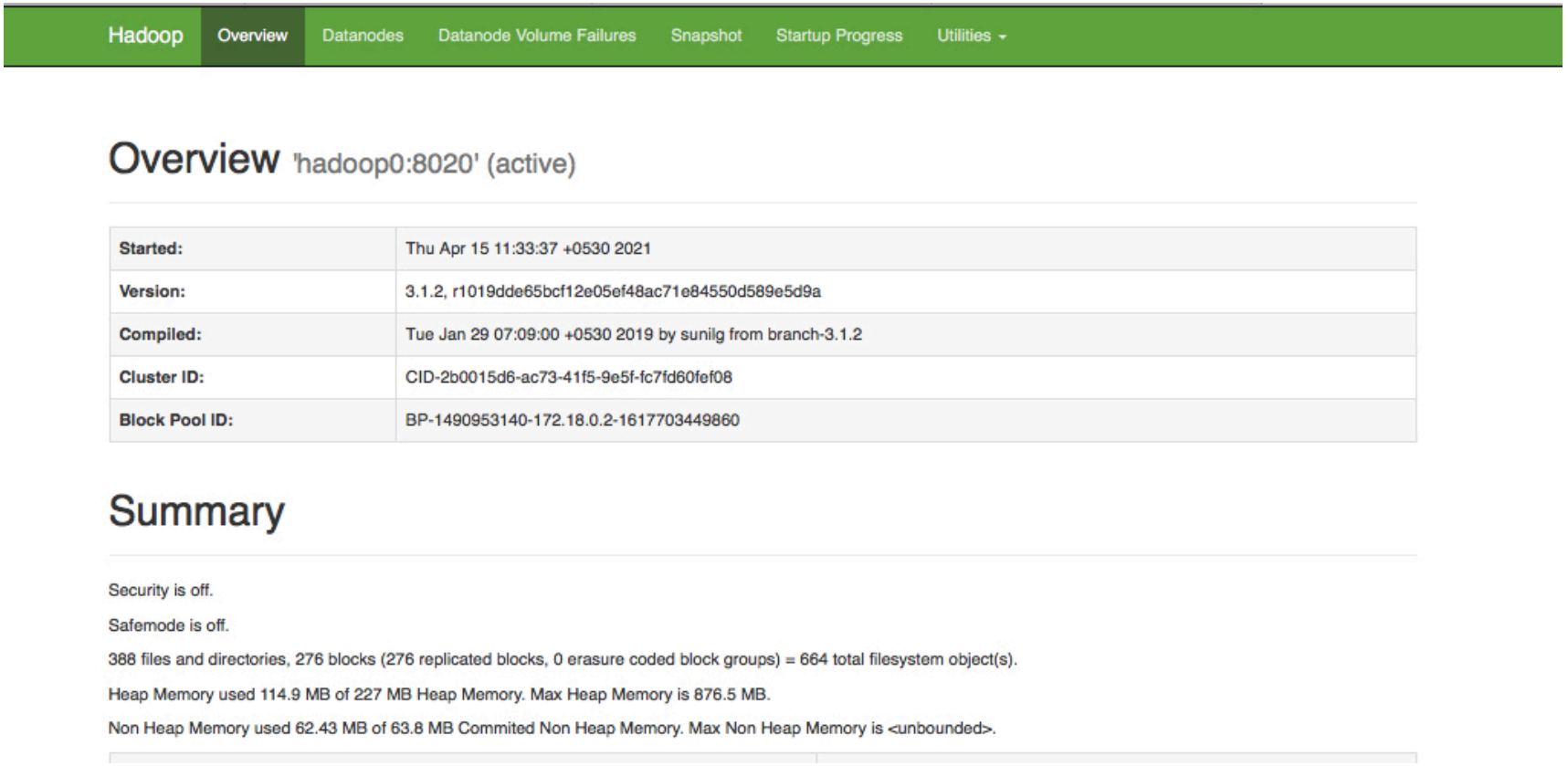
```
hdfs dfs -help
```

For example, to get help on the *dus* command you'd do the following:

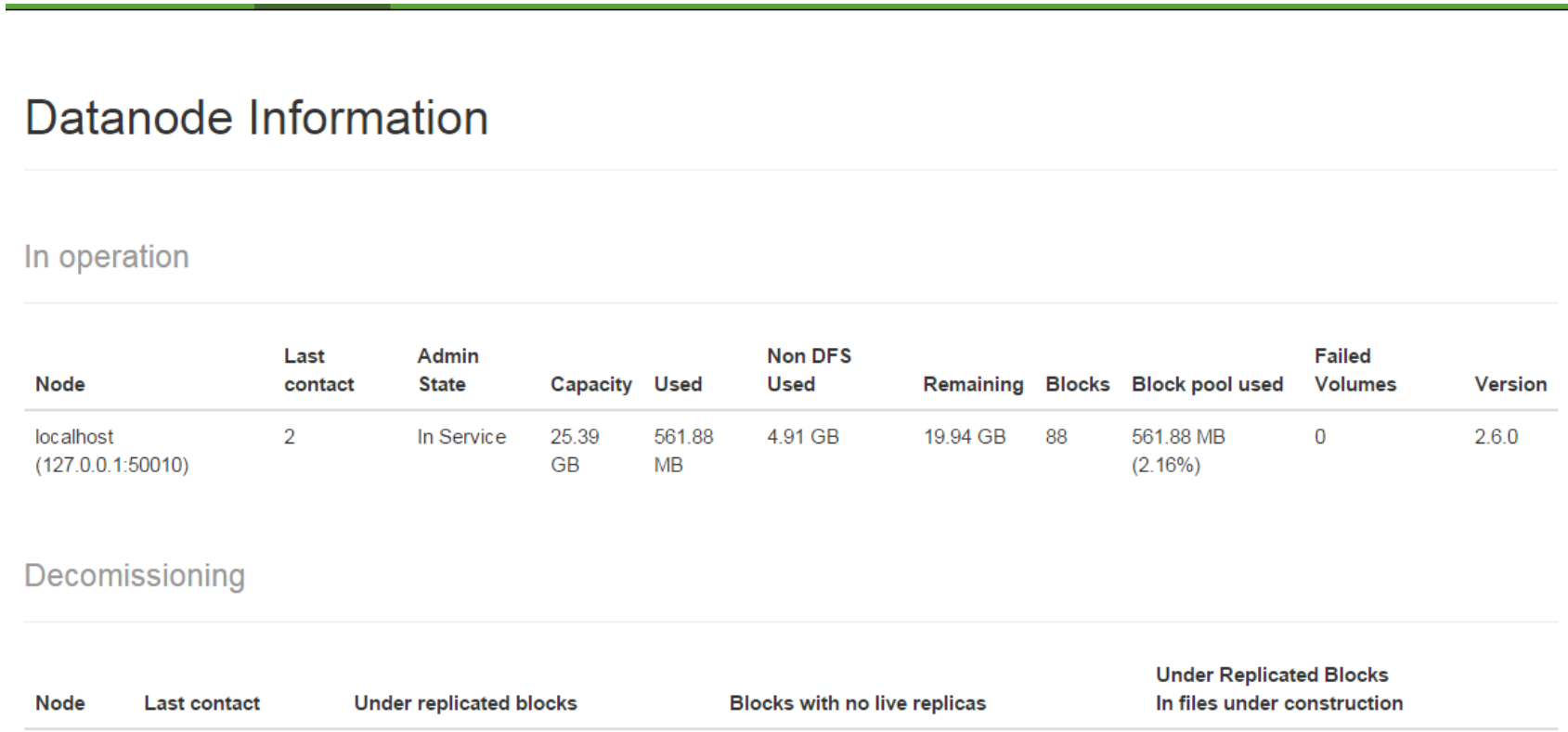
```
hdfs fs -help dus
```

You can observe the HDFS's namenode console as follows:
Familiarize the various options

http://localhost:9870/dfshealth.html#tab-overview


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Overview 'hadoop0:8020' (active)
Started: Thu Apr 15 11:33:37 +0530 2021
Version: 3.1.2, r1019dde65bcf12e05ef48ac71e84550d589e5d9a
Compiled: Tue Jan 29 07:09:00 +0530 2019 by sunilg from branch-3.1.2
Cluster ID: CID-2b0015d6-ac73-41f5-9e5f-fc7fd60fef08
Block Pool ID: BP-1490953140-172.18.0.2-1617703449860
Summary
Security is off.
Safemode is off.
388 files and directories, 276 blocks (276 replicated blocks, 0 erasure coded block groups) = 664 total filesystem object(s).
Heap Memory used 114.9 MB of 227 MB Heap Memory. Max Heap Memory is 876.5 MB.
Non Heap Memory used 62.43 MB of 63.8 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Click on Datanodes



Datanode Information

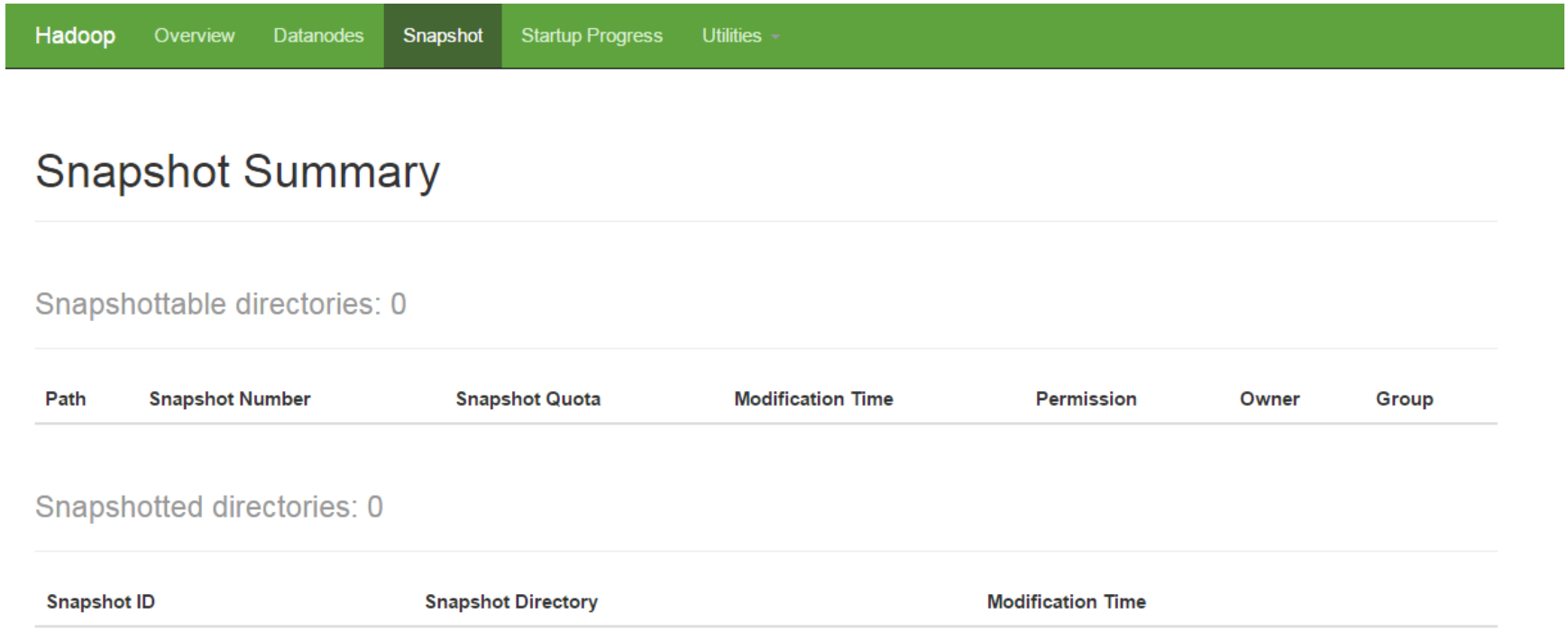
In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
localhost (127.0.0.1:50010)	2	In Service	25.39 GB	561.88 MB	4.91 GB	19.94 GB	88	561.88 MB (2.16%)	0	2.6.0

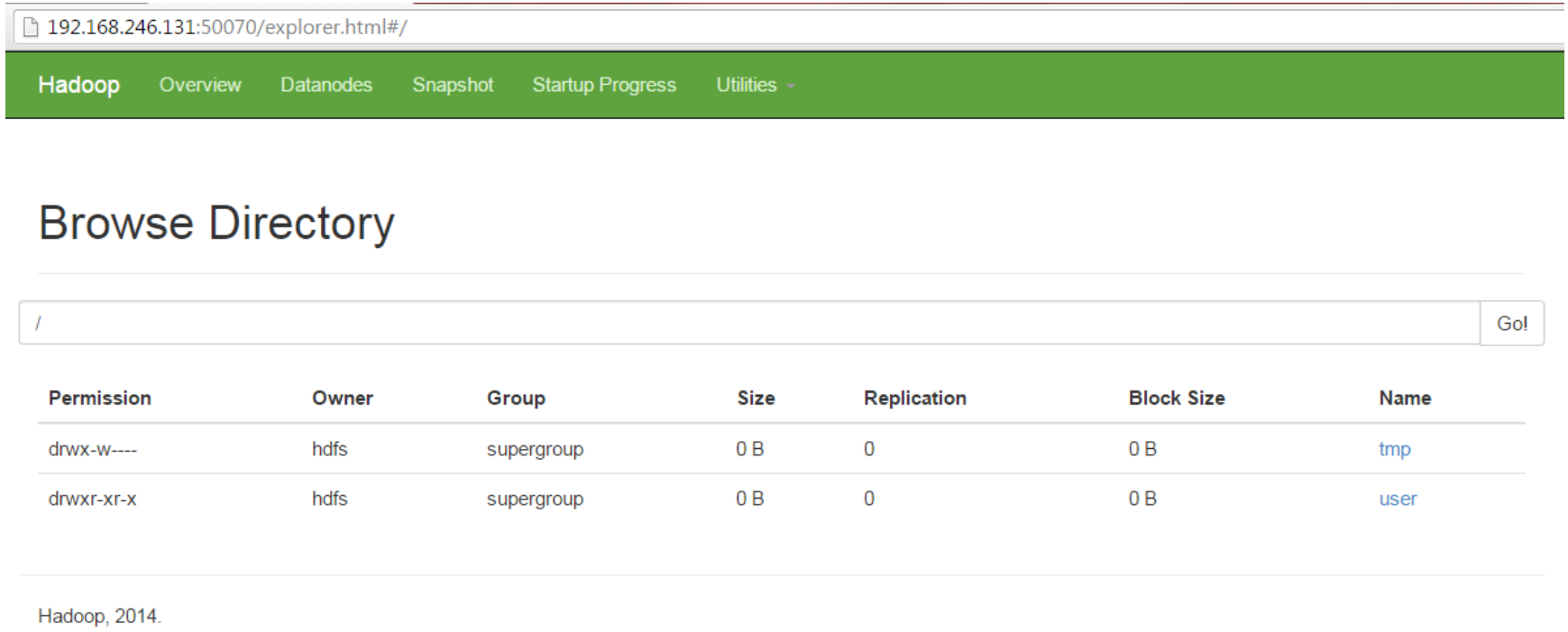
Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction

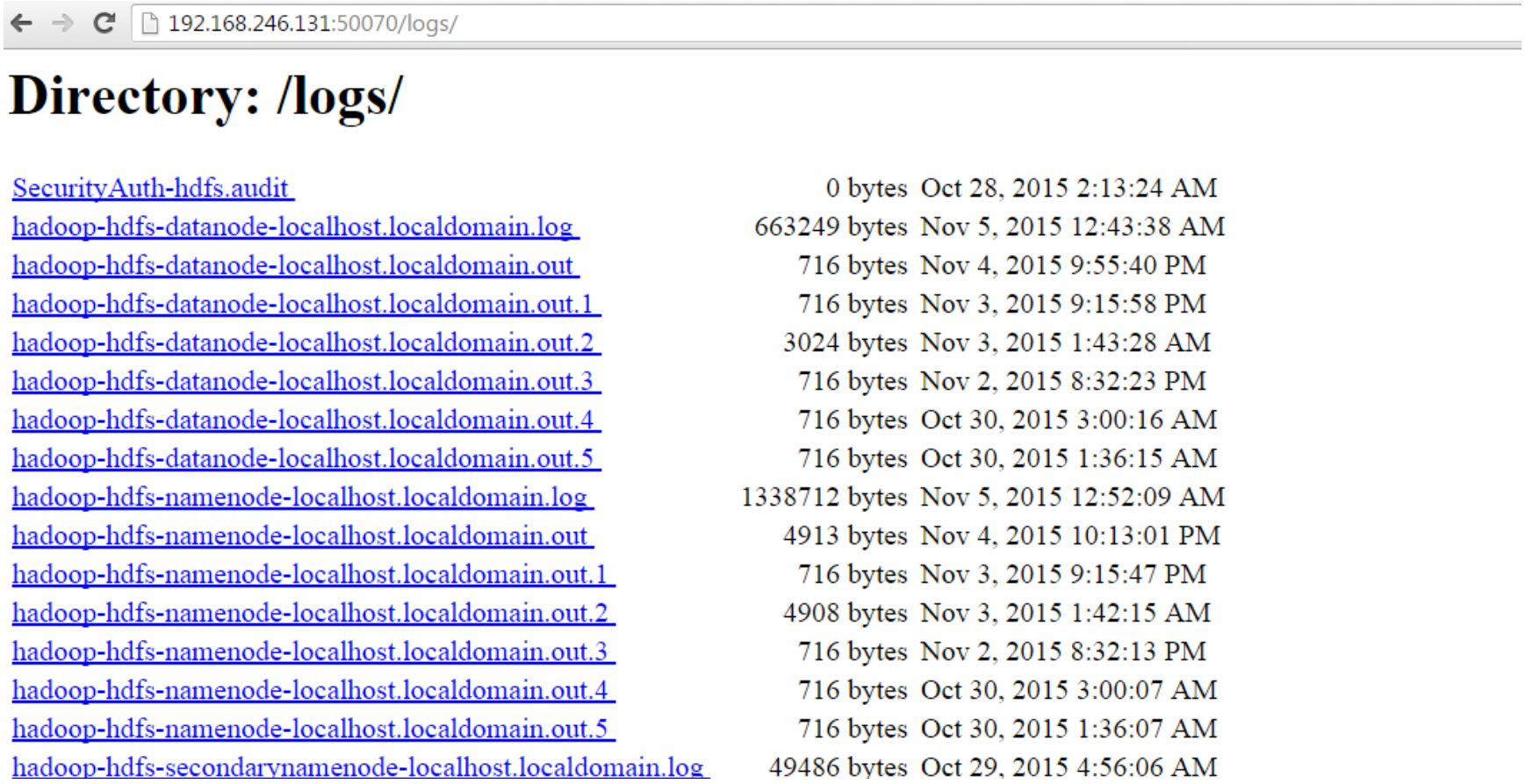
Click on Snapshot



Click on Startup Progress and utlities



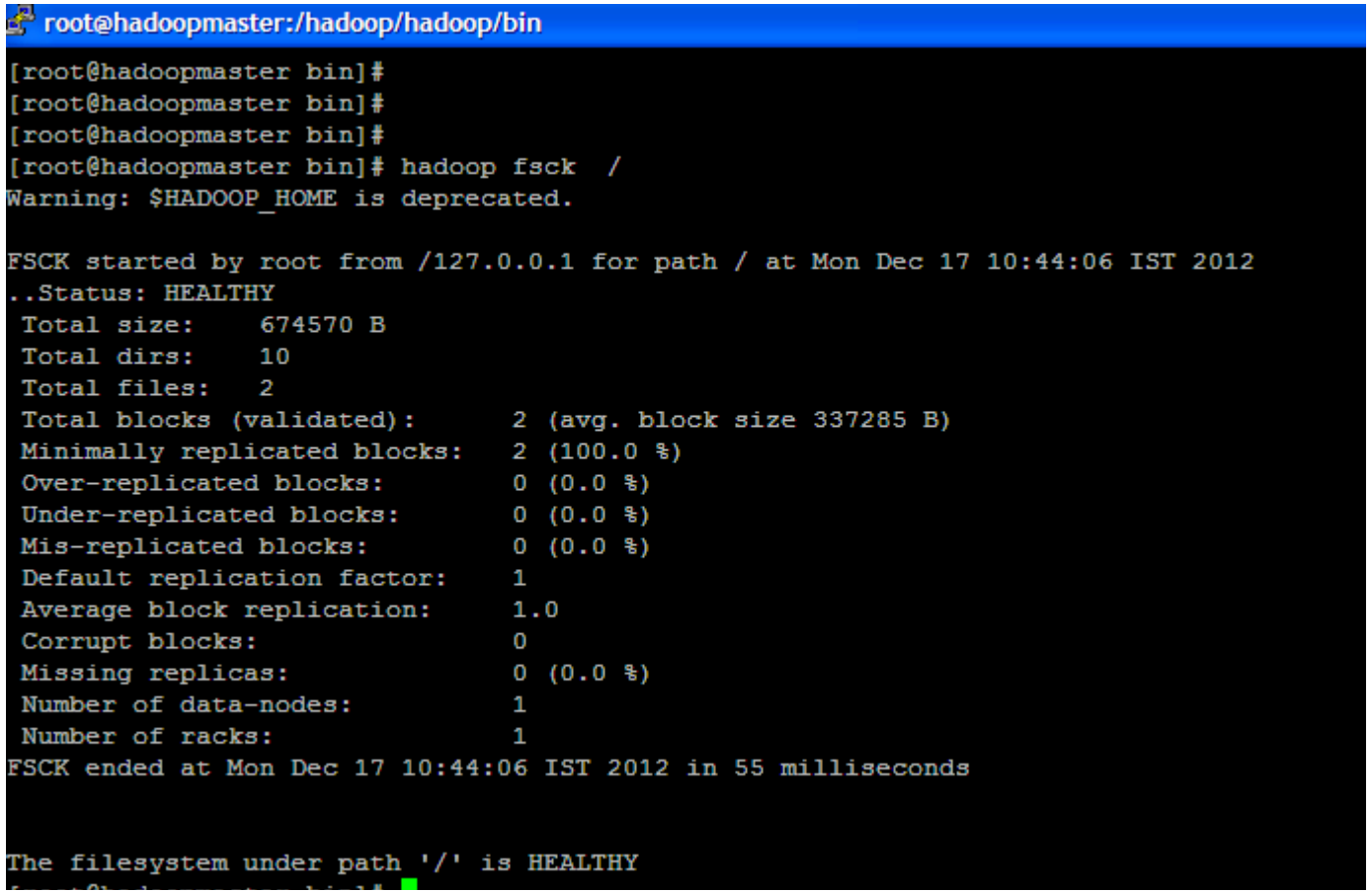
Click on Logs



You can verify the log by clicking on the datanode log file.

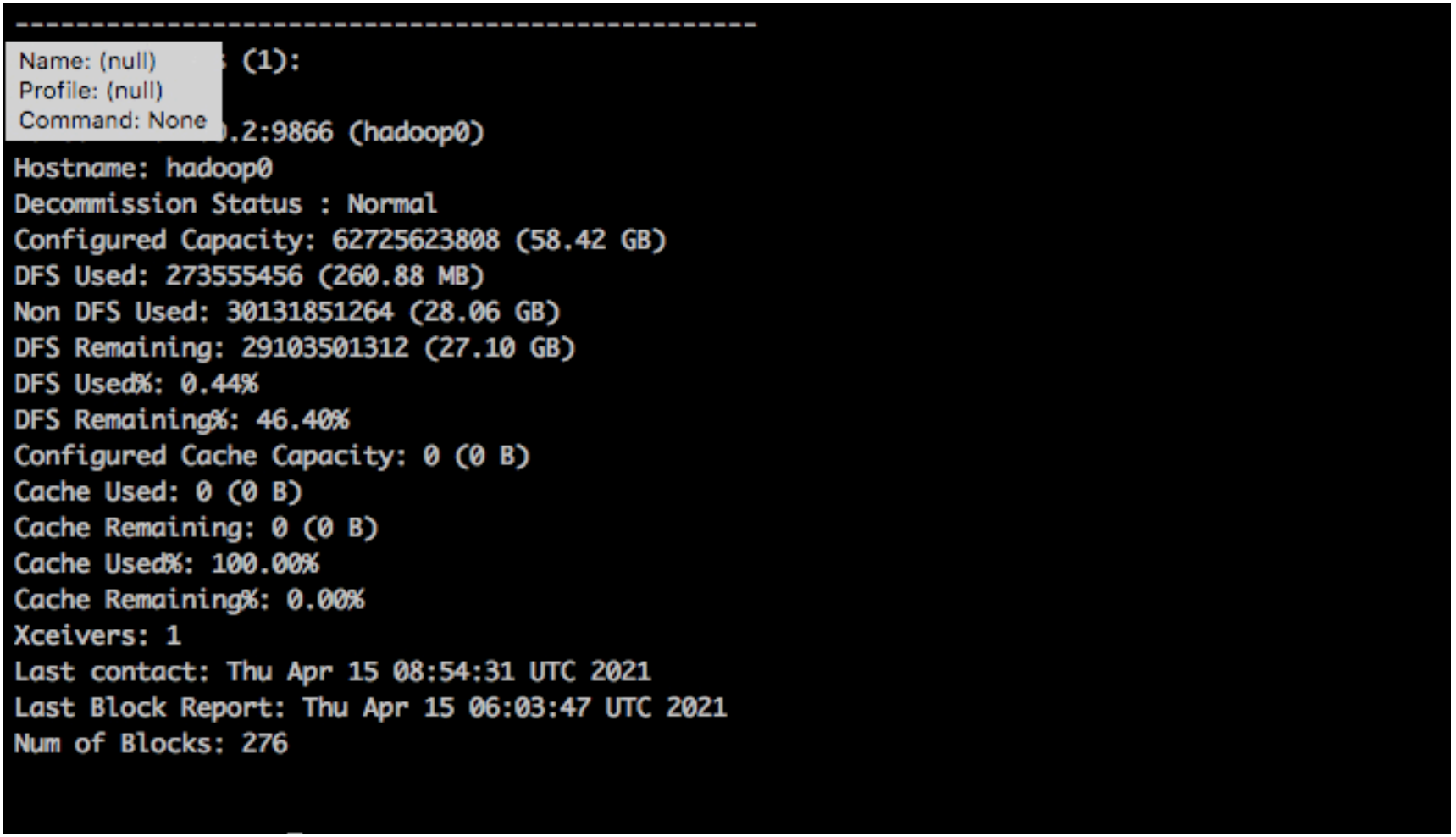
You can verifying the Hadoop File System Health. Check for minimally replicated blocks if any.

```
# hdfs fsck /
```


root@hadoopmaster:/hadoop/hadoop/bin
[root@hadoopmaster bin]#
[root@hadoopmaster bin]#
[root@hadoopmaster bin]#
[root@hadoopmaster bin]# hadoop fsck /
Warning: $HADOOP_HOME is deprecated.
FSCK started by root from /127.0.0.1 for path / at Mon Dec 17 10:44:06 IST 2012
..Status: HEALTHY
Total size: 674570 B
Total dirs: 10
Total files: 2
Total blocks (validated): 2 (avg. block size 337285 B)
Minimally replicated blocks: 2 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Mon Dec 17 10:44:06 IST 2012 in 55 milliseconds
The filesystem under path '/' is HEALTHY

#hdfs dfsadmin -report

This is a dfsadmin command for reporting on each DataNode. It displays the status of Hadoop cluster. Any under replicated blocks or Corrupt replicas?



#hdfs dfsadmin -metasave *hadoop.txt*

This will save some of NameNode's metadata into its log directory under *filename*.

In this metadata, you'll find lists of blocks waiting for replication, blocks being replicated, and blocks awaiting deletion. For replication each block will also have a list of DataNodes being replicated to. Finally, the metasave file will also have summary statistics on each DataNode.

```
[root@hadoop0 opt]# hdfs dfsadmin -metasave hadoop.txt
2021-04-15 09:01:24,497 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
Created metasave file hadoop.txt in the log directory of namenode hdfs://hadoop0:8020
[root@hadoop0 opt]#
```

Go to log folder (Local to the Cluster):
cd /var/hadoop/logs or /opt/hadoop/logs/hadoop.txt

ls

```
[root@hadoopmaster logs]# ls
hadoop1.txt                                 hadoop-root-jobtracker-hadoopmaster.out.1          hadoop-root-secondarynamenode-hadoopmaster.out.2
hadoop-root-datanode-hadoopmaster.log       hadoop-root-namenode-%computername%.log            hadoop-root-tasktracker-hadoopmaster.log
hadoop-root-datanode-hadoopmaster.out       hadoop-root-namenode-%computername%.out            hadoop-root-tasktracker-hadoopmaster.out
hadoop-root-datanode-hadoopmaster.out.1     hadoop-root-namenode-hadoopmaster.log              hadoop-root-tasktracker-hadoopmaster.out.1
hadoop-root-datanode-hadoopmaster.out.2     hadoop-root-namenode-hadoopmaster.out              hadoop-root-tasktracker-hadoopmaster.out.2
hadoop-root-jobtracker-%computername%.log   hadoop-root-namenode-hadoopmaster.out.1            hadoop.txt
hadoop-root-jobtracker-%computername%.out   hadoop-root-secondarynamenode-hadoopmaster.log     history
hadoop-root-jobtracker-hadoopmaster.log     hadoop-root-secondarynamenode-hadoopmaster.out     userlogs
hadoop-root-jobtracker-hadoopmaster.out     hadoop-root-secondarynamenode-hadoopmaster.out.1
```

vi hadoop.txt

```
root@hadoopmaster:/hadoop/hadoop/logs
12 files and directories, 2 blocks = 14 total
Live Datanodes: 1
Dead Datanodes: 0
Metasave: Blocks waiting for replication: 0
Metasave: Blocks being replicated: 0
Metasave: Blocks 0 waiting deletion from 0 datanodes.
Metasave: Number of datanodes: 1
127.0.0.1:50010 IN 11234082816(10.46 GB) 729088(712 KB) 0.01% 6827028480(6.36 GB) Mon Dec 17 10:32:02 IST 2012
~
```

You can get the information of hadoop safe mode.

#hdfs dfsadmin -safemode get

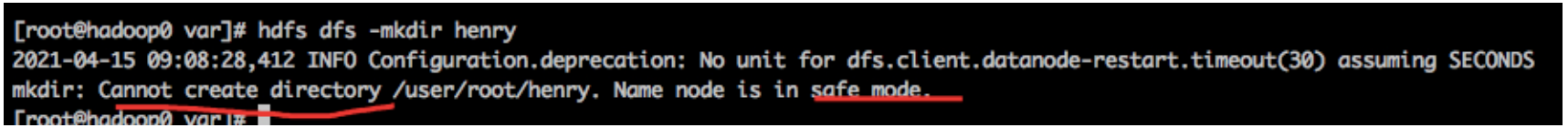
#hdfs dfsadmin -safemode enter

hdfs dfs -ls /

```
[root@hadoop0 var]# hdfs dfsadmin -safemode get
2021-04-15 09:06:58,767 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
Safe mode is OFF
[root@hadoop0 var]# hdfs dfsadmin -safemode enter
2021-04-15 09:07:25,602 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
Safe mode is ON
[root@hadoop0 var]# hdfs dfs -ls /
2021-04-15 09:07:43,677 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
Found 3 items
drwxr-xr-x   - root supergroup          0 2021-04-08 08:57 /apps
drwx-wx-wx   - root supergroup          0 2021-04-07 05:34 /tmp
drwxr-xr-x   - root supergroup          0 2021-04-06 11:05 /user
```

When the cluster is in safe mode, it can perform read operation but not write.

hdfs dfs -mkdir henry



As shown above, it will give error performing any update state on the cluster in safe mode.

#hdfs dfsadmin -safemode leave

Confirm the write operation

#hdfs dfs -mkdir henry

You can determine the version of Hadoop.

#hadoop version

```
[root@hadoop0 var]# hdfs dfsadmin -safemode leave
2021-04-15 09:10:39,577 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
Safe mode is OFF
[root@hadoop0 var]# hadoop version
Hadoop 3.1.2
Source code repository https://github.com/apache/hadoop.git -r 1019dde65bcf12e05ef48ac71e84550d589e5d9a
Compiled by sunilg on 2019-01-29T01:39Z
Compiled with protoc 2.5.0
From source with checksum 64b8bdd4ca6e77cce75a93eb09ab2a9
This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-3.1.2.jar
[root@hadoop0 var]# hdfs version
Hadoop 3.1.2
Source code repository https://github.com/apache/hadoop.git -r 1019dde65bcf12e05ef48ac71e84550d589e5d9a
Compiled by sunilg on 2019-01-29T01:39Z
Compiled with protoc 2.5.0
From source with checksum 64b8bdd4ca6e77cce75a93eb09ab2a9
This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-3.1.2.jar
```

To get a list of files in a directory(/user/root) you would use using REST API:

#curl -i http://lcoalhost:9870/webhdfs/v1/user/root/?op=LISTSTATUS

```
[root@hadoop0 var]# curl -i "http://localhost:9870/webhdfs/v1/user/root/output/?op=LISTSTATUS"
HTTP/1.1 200 OK
Date: Thu, 15 Apr 2021 09:12:42 GMT
Cache-Control: no-cache
Expires: Thu, 15 Apr 2021 09:12:42 GMT
Date: Thu, 15 Apr 2021 09:12:42 GMT
Pragma: no-cache
X-FRAME-OPTIONS: SAMEORIGIN
Content-Type: application/json
Transfer-Encoding: chunked

{"FileStatuses":{"FileStatus":[
{"accessTime":1617780862154,"blockSize":134217728,"childrenNum":0,"fileId":16786,"group":"supergroup","length":0,"modificationTime"
:1617780862161,"owner":"root","pathSuffix":"_SUCCESS","permission":"644","replication":2,"storagePolicy":0,"type":"FILE"},
{"accessTime":1617780861592,"blockSize":134217728,"childrenNum":0,"fileId":16784,"group":"supergroup","length":172,"modificationTim
e":1617780861961,"owner":"root","pathSuffix":"part-r-00000","permission":"644","replication":2,"storagePolicy":0,"type":"FILE"}
]}}
[root@hadoop0 var]#
```

---------- Lab ends here ----------

5. Launching a Cluster: YARN – 60 Minutes(D)

In this lab, we will configure two nodes yarn cluster. This lab depends on the HDFS cluster lab. Ensure that you have completed the above mention lab.

Configure YARN.

Configure parameters as follows:

Update the following in Master and Slave Node.

```
#cd /opt/hadoop
#vi etc/hadoop/mapred-site.xml
```

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_
HOME/share/hadoop/mapreduce/lib/*</value>
  </property>
```

```
</configuration>
```

Only on Master Node.

#vi etc/hadoop/yarn-site.xml

```
<configuration>
<property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>
   <property>
      <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF
_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_H
OME</value>
   </property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
```

```
<name>yarn.nodemanager.local-dirs</name>
<value>file:///opt/yarn/local</value>
</property>
<property>
  <name>yarn.application.classpath</name>
  <value>
$HADOOP_CONF_DIR,$HADOOP_COMMON_HOME/share/hadoop/common/*,$HADOOP_
COMMON_HOME/share/hadoop/common/lib/*,$HADOOP_HDFS_HOME/share/hadoop/hdfs
/*,$HADOOP_HDFS_HOME/share/hadoop/hdfs/lib/*,$HADOOP_MAPRED_HOME/share/ha
doop/mapreduce/*,$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*,

$HADOOP_YARN_HOME/share/hadoop/yarn/*,$HADOOP_YARN_HOME/share/hadoop/yar
n/lib/*
  </value>
</property>
<property>
   <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
    <description>Whether virtual memory limits will be enforced for containers</description>
  </property>
 <property>
   <name>yarn.nodemanager.vmem-pmem-ratio</name>
    <value>4</value>
```

<description>Ratio between virtual memory to physical memory when setting memory limits for containers</description>
</property>
</configuration>

On the Second Node or all slaves/workers node.

#vi etc/hadoop/yarn-site.xml

```
<configuration>
<property>
      <name>yarn.resourcemanager.hostname</name>
      <value>hadoopo</value>
   </property>
<property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>
   <property>
      <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF
_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_H
OME</value>
   </property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

```
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>file:///opt/yarn/local</value>
</property>
<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
   <value>false</value>
   <description>Whether virtual memory limits will be enforced for containers</description>
 </property>
 <property>
  <name>yarn.nodemanager.vmem-pmem-ratio</name>
   <value>4</value>
   <description>Ratio between virtual memory to physical memory when setting memory limits for
containers</description>
 </property>
</configuration>
```

Start ResourceManager daemon and NodeManager daemon: - On Master Node only

```
$ start-yarn.sh
```

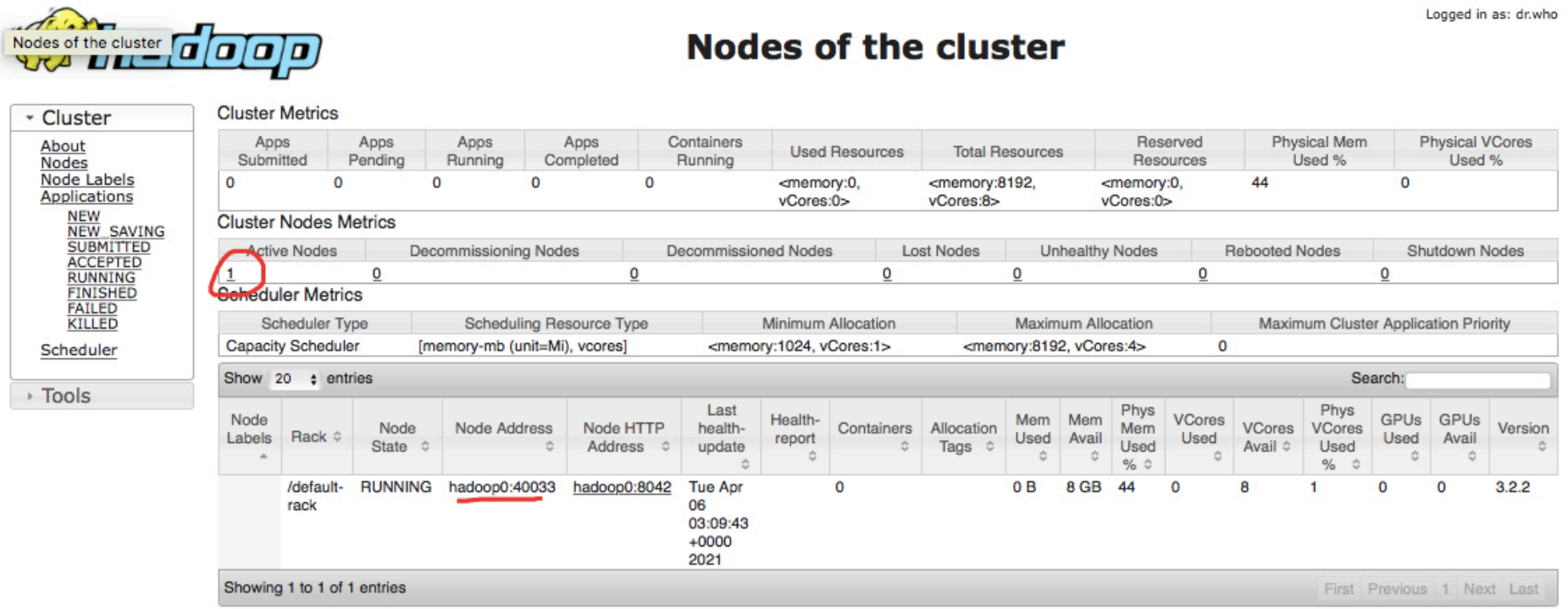
```
[root@hadoop0 hadoop]# sbin/start-yarn.sh
Starting resourcemanager
Last login: Thu Jan 21 08:06:32 UTC 2021 on pts/2
Starting nodemanagers
Last login: Thu Jan 21 08:13:09 UTC 2021 on pts/2
[root@hadoop0 hadoop]#
```

Any issue refer errata below.

Browse the web interface for the ResourceManager; by default, it is available at:

- ResourceManager - http://localhost:8088/

Click on Active Node. You will be able to view one active node as shown below.



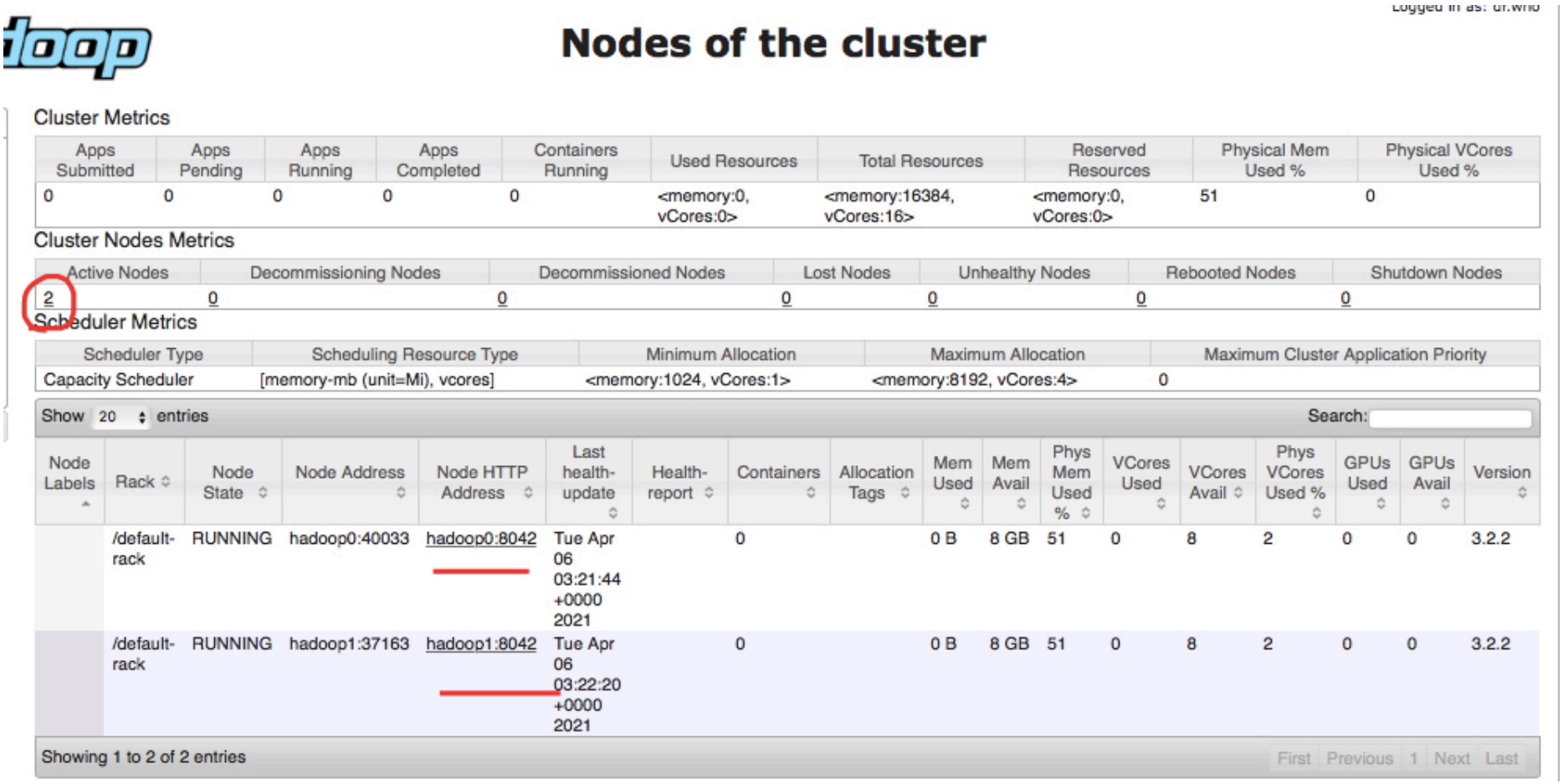
Now Start the Nodemanager on the Second Node or slaves node.

#yarn --daemon start nodemanager

You can verify the Nodes using UI and CLI as shown below. In both the cases, you should have 2 Node managers as shown below.
#yarn node -list --all

```
[root@hadoop0 hadoop]# yarn  node -list -all
2021-04-06 03:22:25,164 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Total Nodes:2
         Node-Id             Node-State Node-Http-Address       Number-of-Running-Containers
   hadoop0:40033                RUNNING      hadoop0:8042                                  0
   hadoop1:37163                RUNNING      hadoop1:8042                                  0
[root@hadoop0 hadoop]#
```

Using web console.



When you're done, you can stop the daemons with: Optional.

```
$ stop-yarn.sh
```

#verify java process with jps , all the following services should be there – On the Master Node.
jps

```
[root@hadoop0 hadoop]# export PATH=$PATH:/opt/jdk/bin
[root@hadoop0 hadoop]# jps
1826 NodeManager
919 NameNode
1034 DataNode
1210 SecondaryNameNode
1708 ResourceManager
2174 Jps
[root@hadoop0 hadoop]#
```

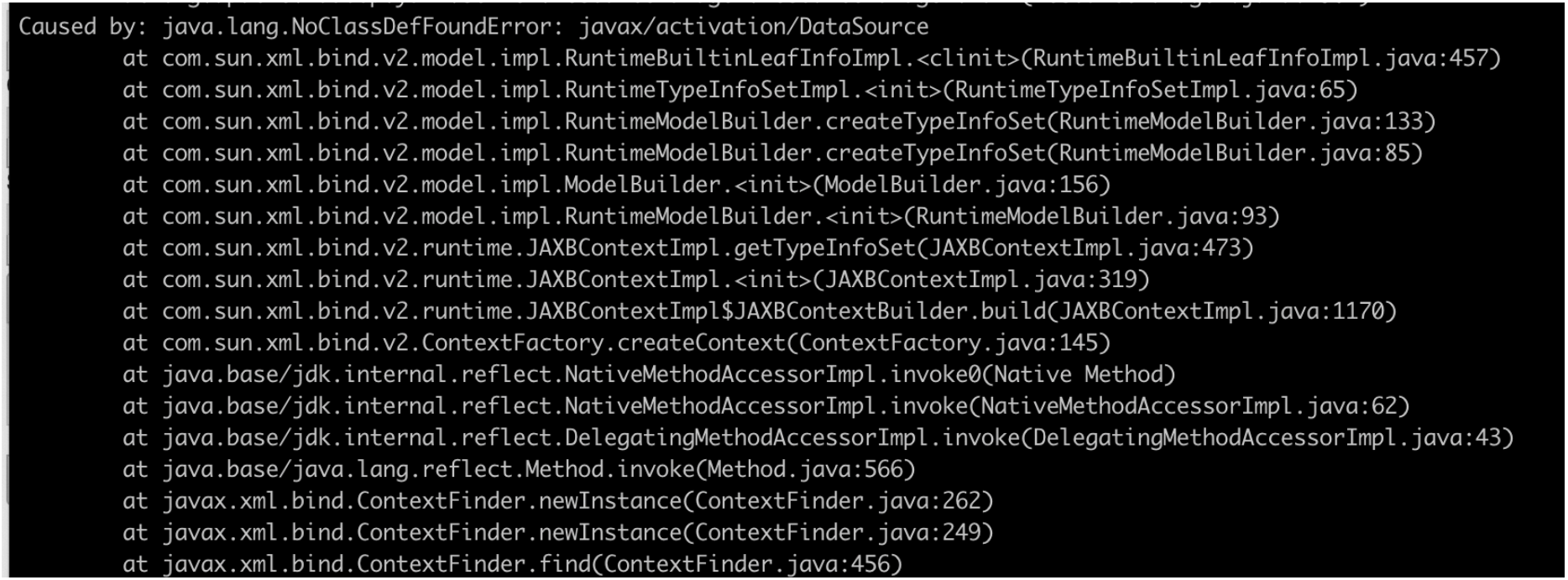
On the other or slave node.

```
[root@hadoop1 logs]# jps
1744 Jps
1523 NodeManager
152 DataNode
[root@hadoop1 logs]#
```

You have successfully configured a Two nodes Hadoop Cluster - YARN.Next let us submit a yarn Job on the above cluster.

Errata:

Observed in the jdk 11.



Solution: Copy : Software/activation-1.1.1.jar to the following folder.

/opt/hadoop/share/hadoop/common

Download

https://search.maven.org/artifact/javax.activation/activation/1.1.1/jar
---------- Lab Ends Here ----------

6. YARN Job on Hadoop Cluster – 60 Minutes(D)

In this lab, we will submit a Yarn job on the cluster, we configured earlier.

Make the HDFS directories required to execute MapReduce jobs:

Verify the folders.

#hdfs dfs -ls -R /

```
[root@hadoop1 logs]# hdfs dfs -ls -R /
drwxr-xr-x   - root supergroup          0 2021-04-05 11:28 /user
drwxr-xr-x   - root supergroup          0 2021-04-05 11:28 /user/root
[root@hadoop1 logs]#
```

If the above folders doesn't exist, then create as shown below else skip.
$ hdfs dfs -mkdir /user
$hdfs dfs -mkdir /user/root

Copy the input files into the distributed filesystem:
$ hdfs dfs -mkdir input
$ hdfs dfs -put /opt/hadoop/etc/hadoop/*.xml input

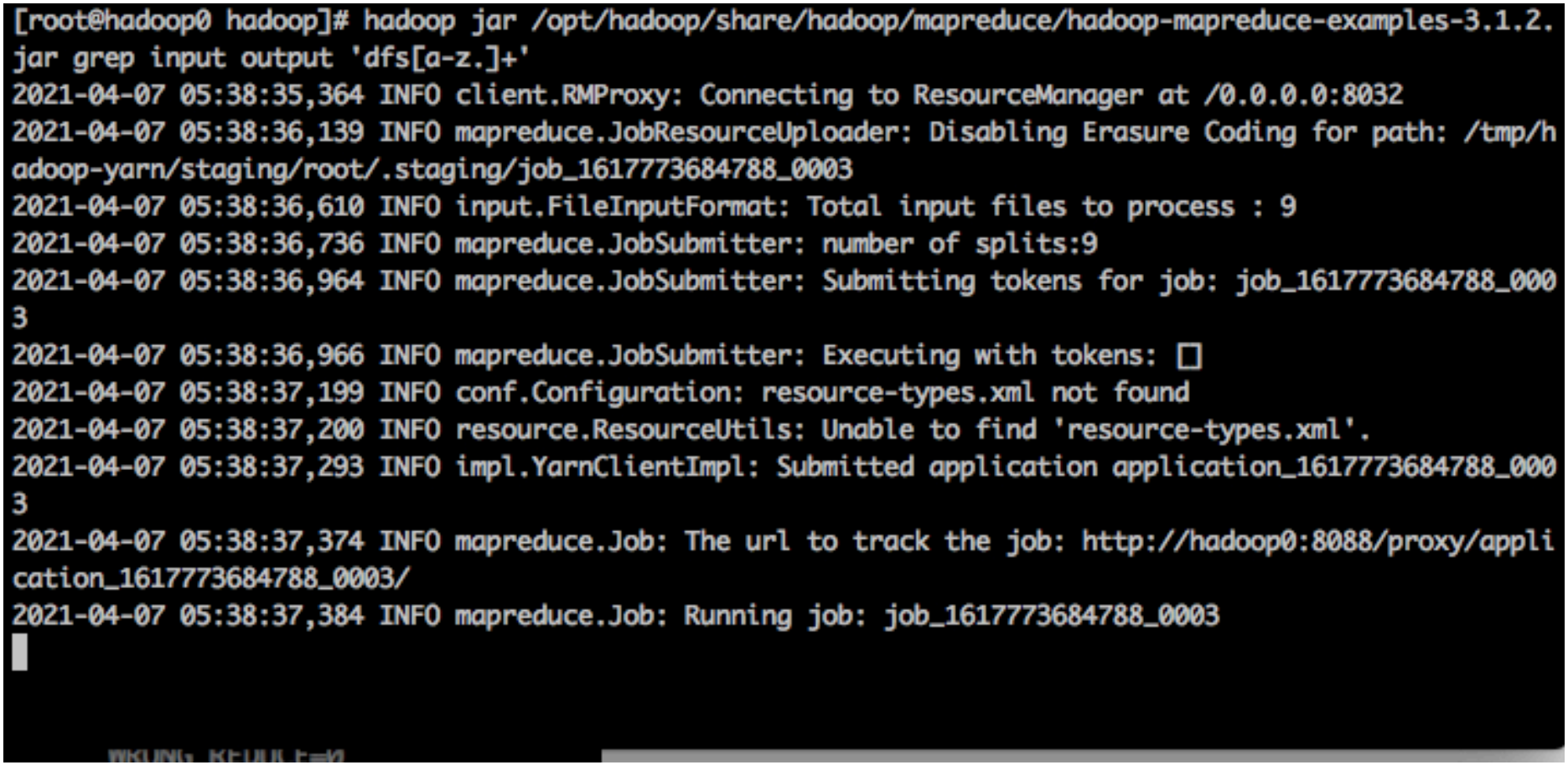
Verify the files

hdfs dfs -ls input

```
[root@hadoop1 logs]# hdfs dfs -ls input
Found 9 items
-rw-r--r--   2 root supergroup       9213 2021-04-06 03:38 input/capacity-scheduler.xml
-rw-r--r--   2 root supergroup        995 2021-04-06 03:38 input/core-site.xml
-rw-r--r--   2 root supergroup      11392 2021-04-06 03:38 input/hadoop-policy.xml
-rw-r--r--   2 root supergroup       1454 2021-04-06 03:38 input/hdfs-site.xml
-rw-r--r--   2 root supergroup        620 2021-04-06 03:38 input/httpfs-site.xml
-rw-r--r--   2 root supergroup       3518 2021-04-06 03:38 input/kms-acls.xml
-rw-r--r--   2 root supergroup        682 2021-04-06 03:38 input/kms-site.xml
-rw-r--r--   2 root supergroup       1060 2021-04-06 03:38 input/mapred-site.xml
-rw-r--r--   2 root supergroup       1210 2021-04-06 03:38 input/yarn-site.xml
[root@hadoop1 logs]#
```

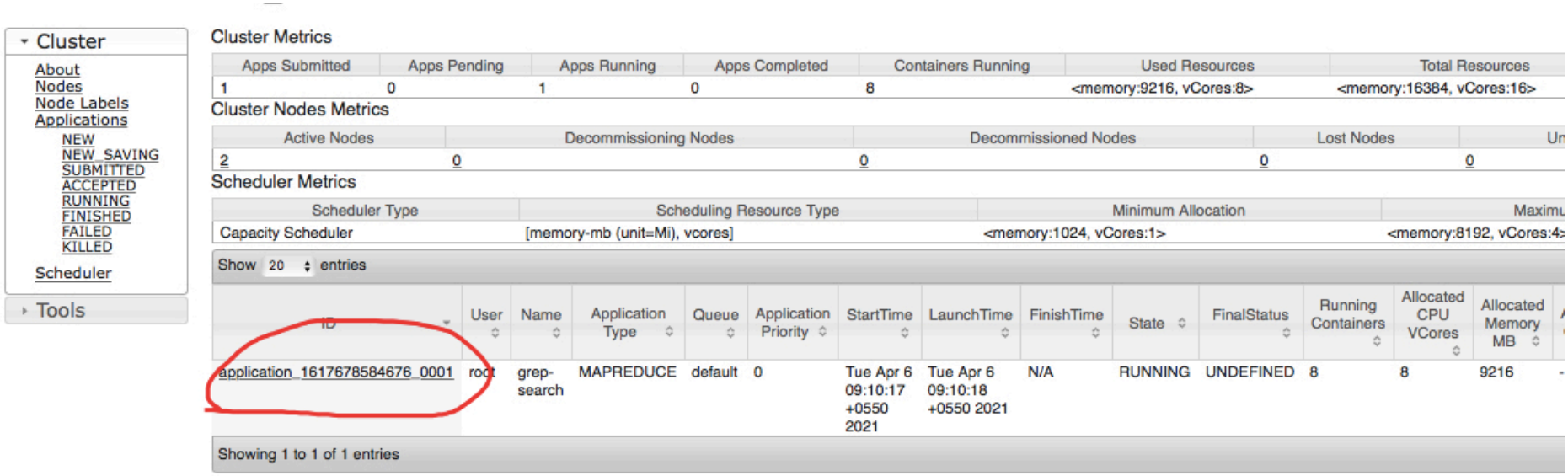
Run some of the examples provided:

#hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar grep input output 'dfs[a-z.]+'


[root@hadoop0 hadoop]# hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.
jar grep input output 'dfs[a-z.]+'
2021-04-07 05:38:35,364 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2021-04-07 05:38:36,139 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/h
adoop-yarn/staging/root/.staging/job_1617773684788_0003
2021-04-07 05:38:36,610 INFO input.FileInputFormat: Total input files to process : 9
2021-04-07 05:38:36,736 INFO mapreduce.JobSubmitter: number of splits:9
2021-04-07 05:38:36,964 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1617773684788_000
3
2021-04-07 05:38:36,966 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-04-07 05:38:37,199 INFO conf.Configuration: resource-types.xml not found
2021-04-07 05:38:37,200 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-04-07 05:38:37,293 INFO impl.YarnClientImpl: Submitted application application_1617773684788_000
3
2021-04-07 05:38:37,374 INFO mapreduce.Job: The url to track the job: http://hadoop0:8088/proxy/appli
cation_1617773684788_0003/
2021-04-07 05:38:37,384 INFO mapreduce.Job: Running job: job_1617773684788_0003

Access the RM web UI.

http://localhost:8088/cluster/apps/RUNNING



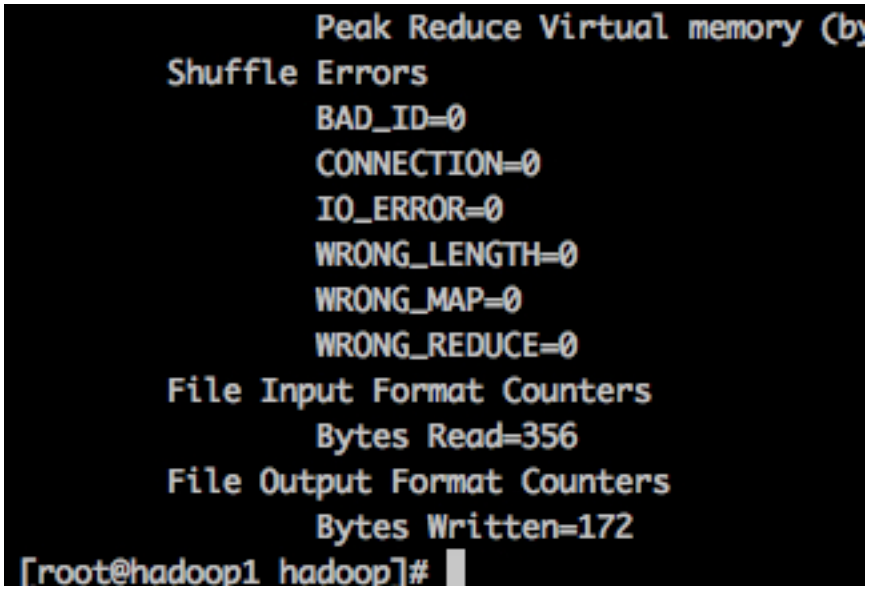
At the end it should have the success state as shown below.

Capacity Scheduler [memory-mb (unit=Mi), vCores] <memory:1024, vCores:1> <memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Re M
application_1617780547234_0003	root	grep-sort	MAPREDUCE	default	0	Wed Apr 7 13:03:46 +0550 2021	Wed Apr 7 13:04:22 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/
application_1617780547234_0002	root	grep-search	MAPREDUCE	default	0	Wed Apr 7 13:02:28 +0550 2021	Wed Apr 7 13:03:43 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/
application_1617780547234_0001	root	grep-search	MAPREDUCE	default	0	Wed Apr 7 12:59:59 +0550 2021	Wed Apr 7 13:01:20 +0550 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/

Showing 1 to 3 of 3 entries



When the job get completed, it should display as shown above.

Examine the output files: Copy the output files from the distributed file system to the local file system and examine them:

```
$ hdfs dfs -get output output
 $ cat output/*
```

```
[root@hadoop1 tmp]# hdfs dfs -get output output
[root@hadoop1 tmp]# cat output/*
1       dfsadmin
1       dfs.replication
1       dfs.permissions
1       dfs.namenode.name.dir
1       dfs.namenode.datanode.registration.ip
1       dfs.datanode.use.datanode.hostname
1       dfs.datanode.data.dir
[root@hadoop1 tmp]#
```

or

View the output files on the distributed filesystem:

$ hdfs dfs -cat output/*

```
[root@hadoop1 tmp]# hdfs dfs -cat output/*
1       dfsadmin
1       dfs.replication
1       dfs.permissions
1       dfs.namenode.name.dir
1       dfs.namenode.datanode.registration.ip
1       dfs.datanode.use.datanode.hostname
1       dfs.datanode.data.dir
[root@hadoop1 tmp]#
```

Execute another program.

Now we can test the Hadoop cluster by running the classic WordCount program.

First, we will create some simple input text files to feed that into the WordCount program:

```
$ mkdir input
$ echo "Hello World" >input/f1.txt
$ echo "Hello Hadoop" >input/f2.txt
```

Now create the input directory on HDFS – Skip this if the folder exist

```
$ hdfs dfs -mkdir -p input1
```

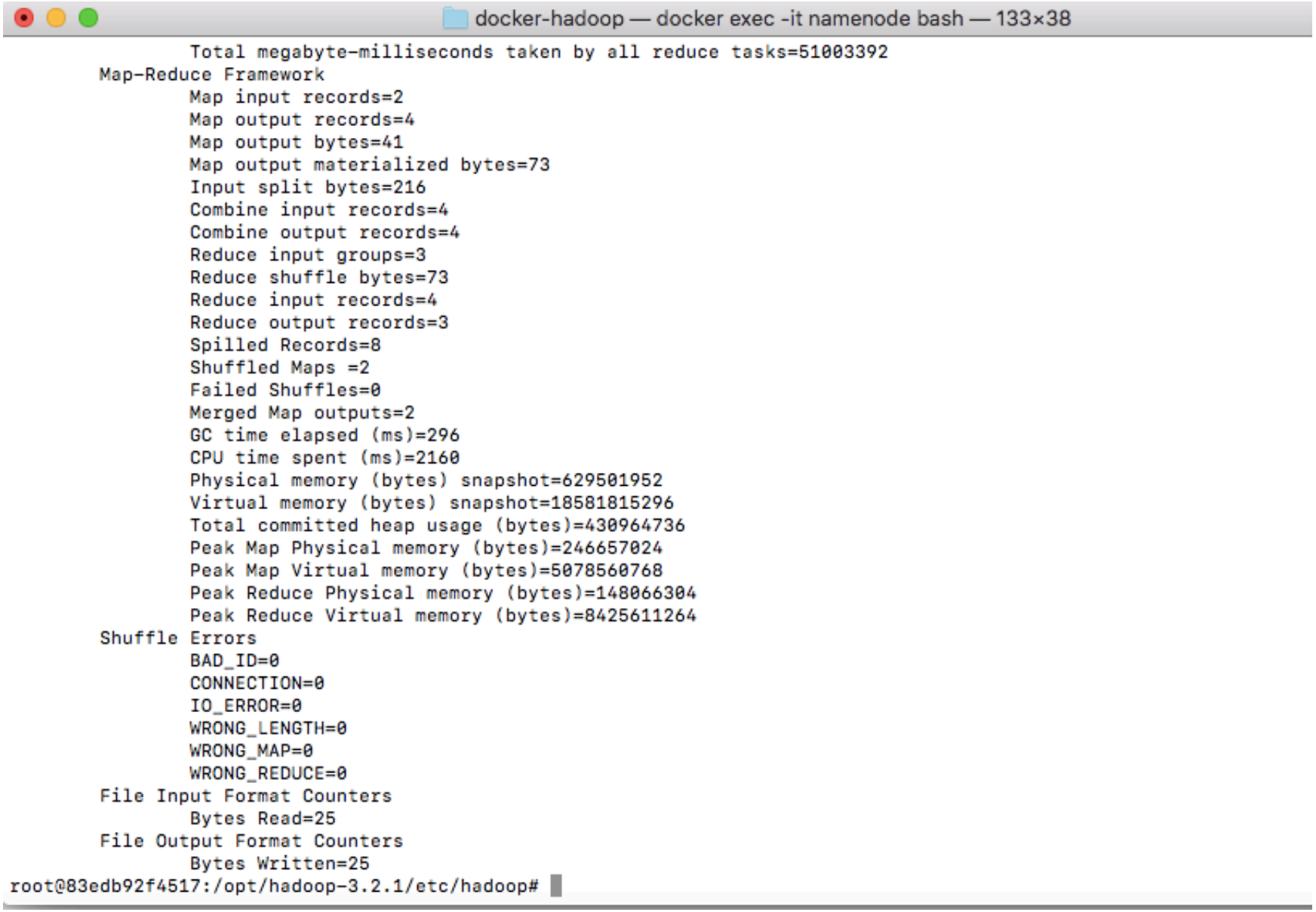
To put the input files to all the datanodes on HDFS, use this command:

```
$ hdfs dfs -put ./input/* input1
```

Now you are ready to run the WordCount program from inside namenode:

```
#hadoop jar /opt/hadoop/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.1.2-sources.jar org.apache.hadoop.examples.WordCount input1 output1
```

```
[root@83edb92f4517:/opt/hadoop-3.2.1/etc/hadoop# hadoop jar /opt/hadoop-3.2.1/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples]
-3.2.1-sources.jar org.apache.hadoop.examples.WordCount input output
2020-07-02 08:31:19,742 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/172.18.0.8:8032
2020-07-02 08:31:20,286 INFO client.AHSProxy: Connecting to Application History server at historyserver/172.18.0.7:10200
2020-07-02 08:31:20,789 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging
/job_1593678172115_0001
2020-07-02 08:31:20,998 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted =
false
2020-07-02 08:31:21,377 INFO input.FileInputFormat: Total input files to process : 2
2020-07-02 08:31:21,481 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted =
false
2020-07-02 08:31:21,563 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted =
false
2020-07-02 08:31:21,621 INFO mapreduce.JobSubmitter: number of splits:2
2020-07-02 08:31:22,037 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted =
false
2020-07-02 08:31:22,117 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1593678172115_0001
2020-07-02 08:31:22,118 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-07-02 08:31:22,474 INFO conf.Configuration: resource-types.xml not found
2020-07-02 08:31:22,477 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2020-07-02 08:31:23,164 INFO impl.YarnClientImpl: Submitted application application_1593678172115_0001
2020-07-02 08:31:23,347 INFO mapreduce.Job: The url to track the job: http://resourcemanager:8088/proxy/application_1593678172115_000
1/
2020-07-02 08:31:23,349 INFO mapreduce.Job: Running job: job_1593678172115_0001
```


docker-hadoop — docker exec -it namenode bash — 133×38
Total megabyte-milliseconds taken by all reduce tasks=51003392
Map-Reduce Framework
Map input records=2
Map output records=4
Map output bytes=41
Map output materialized bytes=73
Input split bytes=216
Combine input records=4
Combine output records=4
Reduce input groups=3
Reduce shuffle bytes=73
Reduce input records=4
Reduce output records=3
Spilled Records=8
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=296
CPU time spent (ms)=2160
Physical memory (bytes) snapshot=629501952
Virtual memory (bytes) snapshot=18581815296
Total committed heap usage (bytes)=430964736
Peak Map Physical memory (bytes)=246657024
Peak Map Virtual memory (bytes)=5078560768
Peak Reduce Physical memory (bytes)=148066304
Peak Reduce Virtual memory (bytes)=8425611264
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=25
File Output Format Counters
Bytes Written=25
root@83edb92f4517:/opt/hadoop-3.2.1/etc/hadoop#

To print out the WordCount program result:

hdfs dfs -cat output1/part-r-00000

```
root@83edb92f4517:/opt/hadoop-3.2.1/etc/hadoop# hdfs dfs -cat output/part-r-00000
2020-07-02 08:34:20,660 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted =
false
Docker  1
Hello   2
World   1
root@83edb92f4517:/opt/hadoop-3.2.1/etc/hadoop#
```

Errata:
[2021-04-07 07:22:02.546]Container [pid=2253,containerID=container_1617779932261_0001_01_000012] is running 261794304B beyond the 'VIRTUAL' memory limit. Current usage: 58.8 MB of 1 GB physical memory used; 2.3 GB of 2.1 GB virtual memory used. Killing container.

Solution 1:

There is a check placed at Yarn level for Virtual and Physical memory usage ratio. Issue is not only that VM doesn't have sufficient physical memory. But it is because Virtual memory usage is more than expected for given physical memory.

Note : This is happening on Centos/RHEL 6 due to its aggressive allocation of virtual memory.

It can be resolved either by :

1. Disable virtual memory usage check by setting **yarn.nodemanager.vmem-check-enabled** to **false**;
2. Increase VM:PM ratio by setting **yarn.nodemanager.vmem-pmem-ratio** to some higher value.

References :

https://issues.apache.org/jira/browse/HADOOP-11364

http://blog.cloudera.com/blog/2014/04/apache-hadoop-yarn-avoiding-6-time-consuming-gotchas/

Add following property in yarn-site.xml

```
<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
   <value>false</value>
   <description>Whether virtual memory limits will be enforced for containers</description>
 </property>
<property>
  <name>yarn.nodemanager.vmem-pmem-ratio</name>
   <value>4</value>
```

<description>Ratio between virtual memory to physical memory when setting memory limits for containers</description>
</property>

Solution 2:

hive -hiveconf tez.am.resource.memory.mb=4096
Another setting to consider tweaking is yarn.app.mapreduce.am.resource.mb

---------- Lab Ends here ----------

7. Capacity Scheduler – 90 minutes(D)

Use 2 Nodes cluster.

In order to configure a Hadoop cluster to utilize a specific amount of memory for YARN node managers, edit the parameter **yarn.nodemanager.resource.memory-mb** in yarn configuration file **/opt/ hadoop/etc/conf/yarn-site.xml**. After the desired value has been defined, restart the YARN services to refresh with the modifications.

yarn-site.xml

```
<property>
 <name>yarn.nodemanager.resource.memory-mb</name>
 <value>4096</value>
</property>
```

Note: In this example, 4 GBs of memory is assigned for utilization by YARN node managers per server.

In order to allow YARN to use capacity scheduler, define the parameter of **yarn.resourcemanager.scheduler.class** in **yarn-site.xml** to use CapacityScheduler

yarn-site.xml

```
<property>
 <name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
```

Setup the queues. CapacityScheduler has a predefined queue called root. All queues in the system are children of the root queue. In **capacity-scheduler.xml**, parameter **yarn.scheduler.capacity.root.queues** is used to define the child queues. For example, to create 3 queues, specify the name of the queues in a comma separated list.

Take a backup of capacity-scheduler.xml.

All update should be done in **capacity-scheduler.xml. Update at the end of the file before </configuration> tag.**

```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>sales,marketing,default</value>
  <description>
  The queues at the this level (root is the root queue).
 </description>
</property>
```

After making the changes above, now specify the queue specific parameters.

Note: Parameters denoting queue specific properties follow a standard set of naming convention and they include the name of the queue for which they are relevant.

Here is an example of general syntax:

yarn.scheduler.capacity.<queue-path>.<parameter>
where :

<queue-path> : identifies the name of the queue.
<parameter> : identifies the parameter whose value is being set.

To set the percentage of cluster resource, which must be allocated to these resources, edit the value of the parameter **yarn.scheduler.capacity.<queue-path>.capacity** in **capacity-scheduler.xml** accordingly. In the example below, the queues are set to use 50%, 30% and 20% of the allocated clustered resources, which was earlier set by **yarn.nodemanager.resource.memory-mb** per nodemanager.

```
<property>
 <name>yarn.scheduler.capacity.root.sales.capacity</name>
 <value>50</value>
 <description>Default queue target capacity.</description>
 </property>

<property>
 <name>yarn.scheduler.capacity.root.marketing.capacity</name>
<value>30</value>
 <description>Default queue target capacity.</description>
 </property>

<property>
 <name>yarn.scheduler.capacity.root.default.capacity</name>
 <value>20</value>
 <description>Default queue target capacity.</description>
 </property>
```

yarn.scheduler.capacity.<queue-path>.state enables the queue to allow jobs or applications to be submitted through them, the state of the queue must be RUNNING. Otherwise an error occurs stating that queue is STOPPED. RUNNING and STOPPED are the permissible values for this parameter.

```
<property>
 <name>yarn.scheduler.capacity.root.sales.state</name>
 <value>RUNNING</value>
 <description>
 The state of the default queue. State can be one of RUNNING or STOPPED.
 </description>
 </property>

 <property>
 <name>yarn.scheduler.capacity.root.marketing.state</name>
 <value>RUNNING</value>
 <description>
 The state of the default queue. State can be one of RUNNING or STOPPED.
 </description>
 </property>

 <property>
 <name>yarn.scheduler.capacity.root.default.state</name>
 <value>RUNNING</value>
 <description>
 The state of the default queue. State can be one of RUNNING or STOPPED.
 </description>
 </property>
```

Bringing the queues in effect:
Once the required parameters are defined in capacity-scheduler.xml file, run the command to bring the changes in effect.

yarn rmadmin -refreshQueues

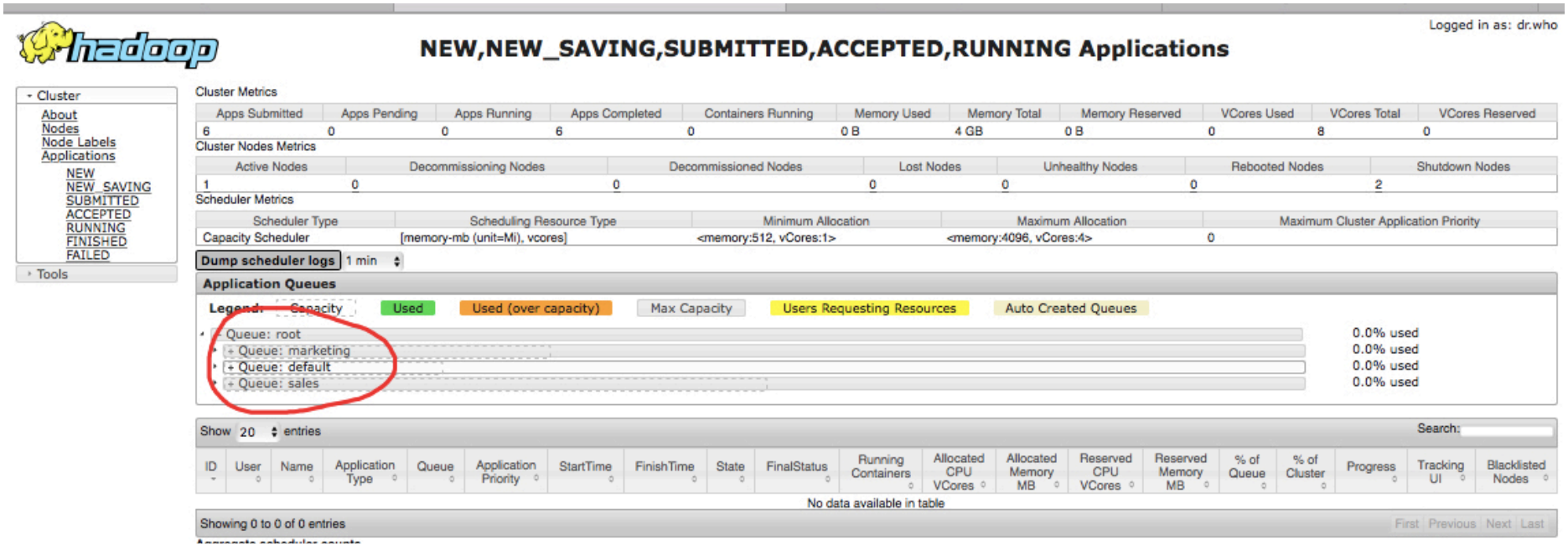
```
[root@master0 hadoop]# yarn rmadmin -refreshQueues
WARNING: YARN_CONF_DIR has been replaced by HADOOP_CONF_DIR. Using value of YARN_CONF_DIR.
2022-06-18 04:26:22,348 INFO client.RMProxy: Connecting to ResourceManager at master0/172.18.0.2:8033
[root@master0 hadoop]#
```

Once the command runs properly, verify if the queues are setup using 2 options:

#mapred queue -list

```
[root@master0 hadoop]# mapred queue -list
2022-06-18 04:27:35,105 INFO client.RMProxy: Connecting to ResourceManager at master0/172.18.0.2:8032
======================
Queue Name : marketing
Queue State : running
Scheduling Info : Capacity: 30.000002, MaximumCapacity: 100.0, CurrentCapacity: 0.0
======================
Queue Name : default
Queue State : running
Scheduling Info : Capacity: 20.0, MaximumCapacity: 100.0, CurrentCapacity: 0.0
======================
Queue Name : sales
Queue State : running
Scheduling Info : Capacity: 50.0, MaximumCapacity: 100.0, CurrentCapacity: 0.0
[root@master0 hadoop]#
```

#Open YARN resourcemanager GUI from Resource Manager GUI, click Scheduler.

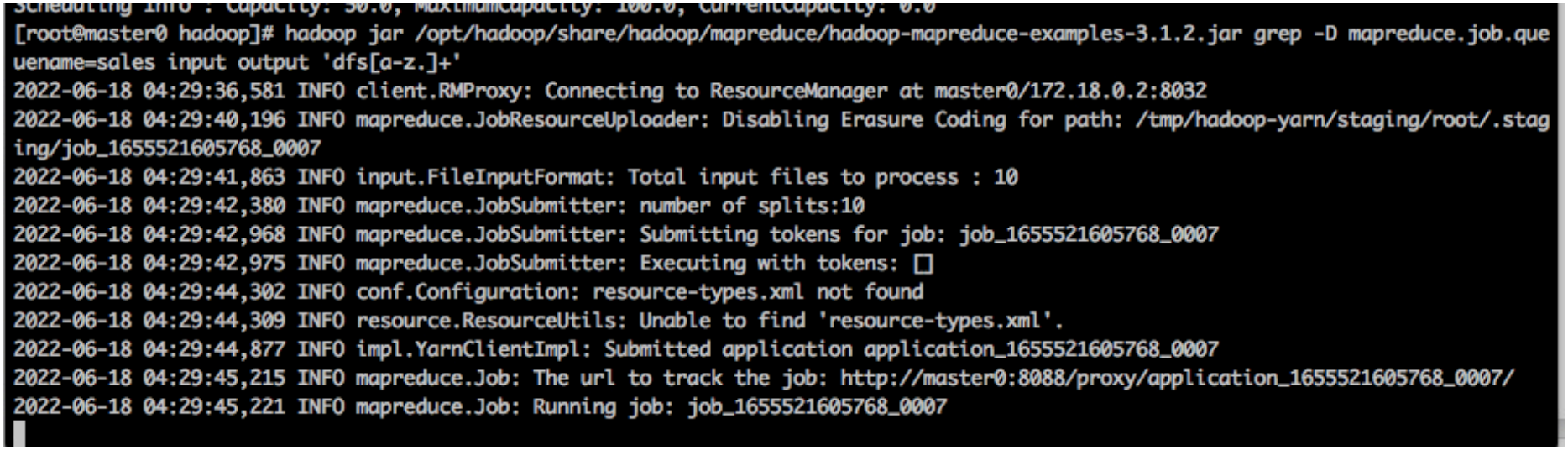


To submit an application use the parameter:

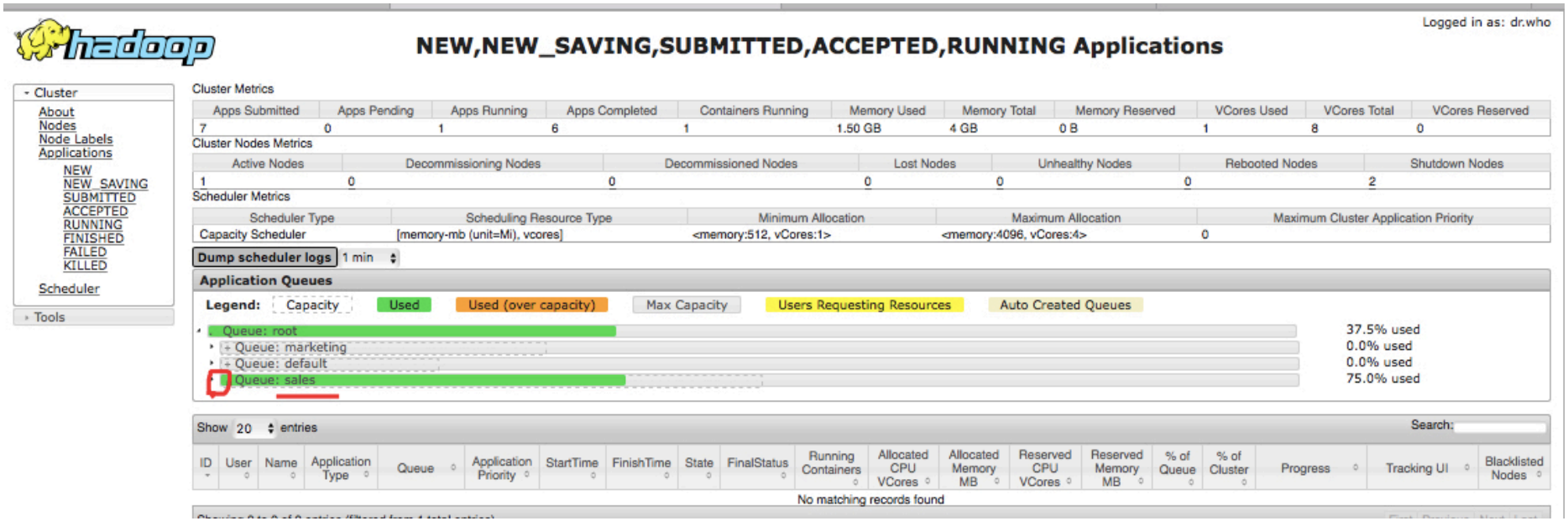
Dmapred.job.queue.name=<queue-name> or -Dmapred.job.queuename=<queue-name>

Example:
Open a Terminal and execute the following(Submitting Job on sales queue)::

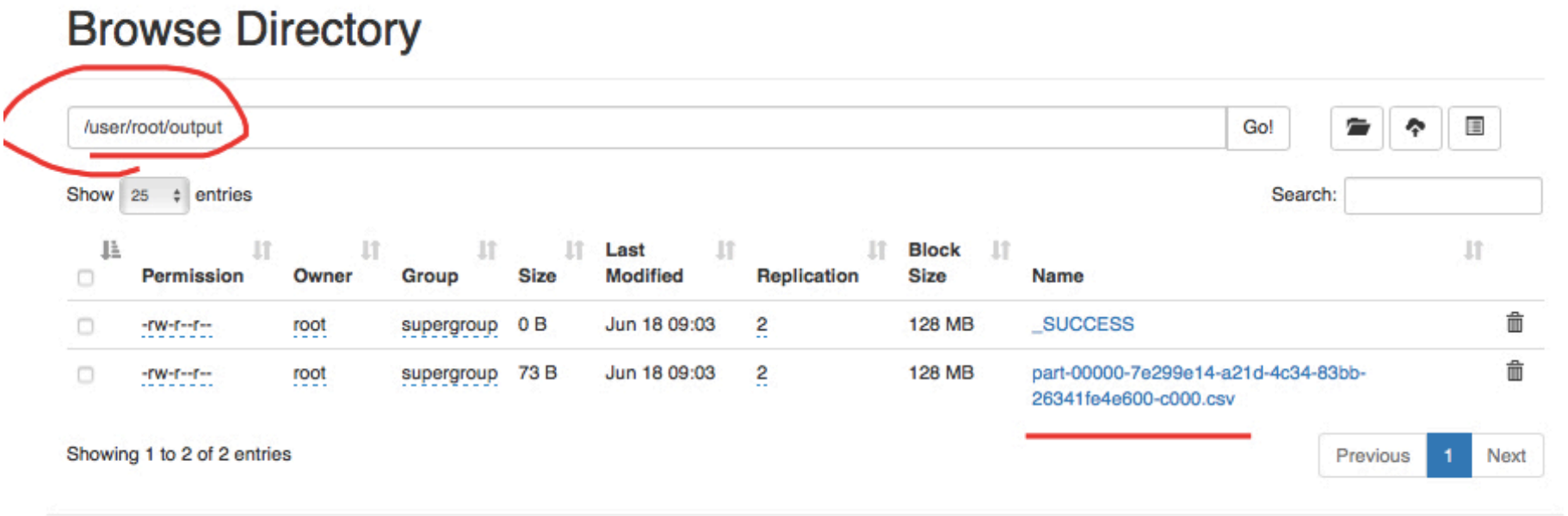
```
# hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar grep -D mapreduce.job.queuename=sales input output2 'dfs[a-z.]+'
```



While the job is executing, monitor the Resource Manger GUI to see queue the job submitted to. Here is a snapshot of the name. In the snapshot below, green color indicates the queue which is being used by the above WordCount application.



Verify the end result from the Namenode UI.



Revert the capacity-schedule file before going ahead.

#cp capacity-scheduler.xml_default capacity-scheduler.xml

Refresh the queue list.
yarn rmadmin -refreshQueues
---------- Lab Ends Here ----------
https://community.pivotal.io/s/article/How-to-configure-queues-using-YARN-capacity-scheduler-xml?language=en_US

8. HDFS High Availability Using the Quorum Journal Manager – 3 Hrs(D)

You require three nodes for configuring Namenode HA. The service layout is as shown below:

hostname	Remarks	Services	Roles
hadoop0	First Machine	NN,DN , JN	Master
hadoop1	Second Machine	NN,DN , JN	Master
hadoop2	New Machine	JN	worker

At this step, you should have configured two nodes HDFS cluster.

Add one More Node.

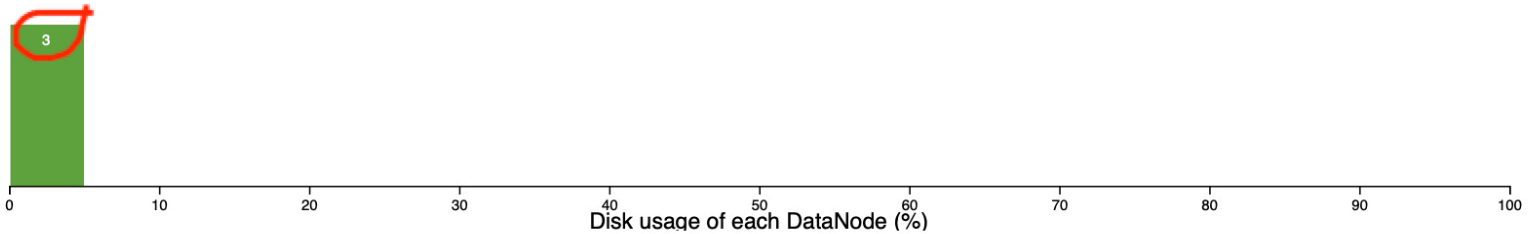
Refer the cluster Node addition step for adding Node.

At the end of the above steps. You should have 3 nodes cluster as shown below.

Datanode Information

✔ In service ● Down ⊘ Decommissioned ⏻ Decommissioned & dead 🔧 In Maintenance 🔧 In Maintenance & dead

Datanode usage histogram



In operation

Show 25 entries Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✔hadoop0:9866 (172.18.0.2:9866)	http://hadoop0:9864	1s	62m	125.93 GB	44	2.79 MB (0%)	3.1.2
✔hadoop1:9866 (172.18.0.3:9866)	http://hadoop1:9864	1s	82m	125.93 GB	44	2.79 MB (0%)	3.1.2
✔hadoop2:9866 (172.18.0.4:9866)	http://hadoop2:9864	2s	0m	125.93 GB	0	24 KB (0%)	3.1.2

Showing 1 to 3 of 3 entries Previous 1 Next

To configure HA NameNodes, you must add several configuration options to your **hdfs-site.xml** configuration file. – On All the nodes.

Take a back up of the hdfs-site.xml before making any changes on all nodes.

The order in which you set these configurations is unimportant, but the values you choose for **dfs.nameservices** and **dfs.ha.namenodes.[nameservice ID]** will determine the keys of those that follow. Thus, you should decide on these values before setting the rest of the configuration options.

dfs.nameservices - the logical name for this new nameservice
Choose a logical name for this nameservice, for example “mycluster”, and use this logical name for the value of this config option. The name you choose is arbitrary. It will be used both for configuration and as the authority component of absolute HDFS paths in the cluster.

```
<property>
  <name>dfs.nameservices</name>
  <value>mycluster</value>
</property>
```

dfs.ha.namenodes.[nameservice ID] - unique identifiers for each NameNode in the nameservice

```
<property>
  <name>dfs.ha.namenodes.mycluster</name>
  <value>nn1,nn2</value>
</property>
```

dfs.namenode.rpc-address.[nameservice ID].[name node ID] - the fully-qualified RPC address for each NameNode to listen on

```
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn1</name>
  <value>hadoop0:8020</value>
</property>
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn2</name>
  <value>hadoop1:8020</value>
</property>
```

dfs.namenode.shared.edits.dir - the URI which identifies the group of JNs where the NameNodes will write/read edits
This is where one configures the addresses of the JournalNodes which provide the shared edits storage, written to by the Active nameNode and read by the Standby NameNode to stay up-to-date with all the file system changes the Active NameNode makes.

```
<property>
  <name>dfs.namenode.shared.edits.dir</name>
<value>qjournal://hadoop0:8485;hadoop1:8485;hadoop2:8485/mycluster</value>
</property>
```

dfs.client.failover.proxy.provider.[nameservice ID] - the Java class that HDFS clients use to contact the Active NameNode

```
<property>
  <name>dfs.client.failover.proxy.provider.mycluster</name>
  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

dfs.journalnode.edits.dir - the path where the JournalNode daemon will store its local state
This is the absolute path on the JournalNode machines where the edits and other local state used by the JNs will be stored.

```
<property>
  <name>dfs.journalnode.edits.dir</name>
  <value>/opt/journal/data</value>
</property>
```

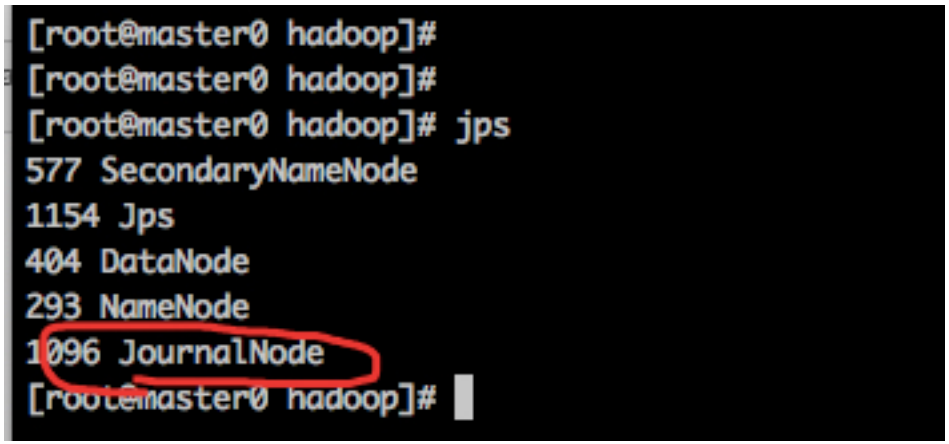
After all the above necessary configuration options have been set, you must start the JournalNode daemons on the set of machines where they will run. This can be done by running the command "*hdfs --daemon start journalnode*" and waiting for the daemon to start on each of the relevant machines.

Create the following folders in all the nodes for the journal to stores its data:

```
#mkdir -p /opt/journal/data
```

On All 3 Nodes start the Journal services:
#hdfs --daemon start journalnode

Verify the process on all nodes using jps:



You should observe journal process as highlighted above in all the nodes.

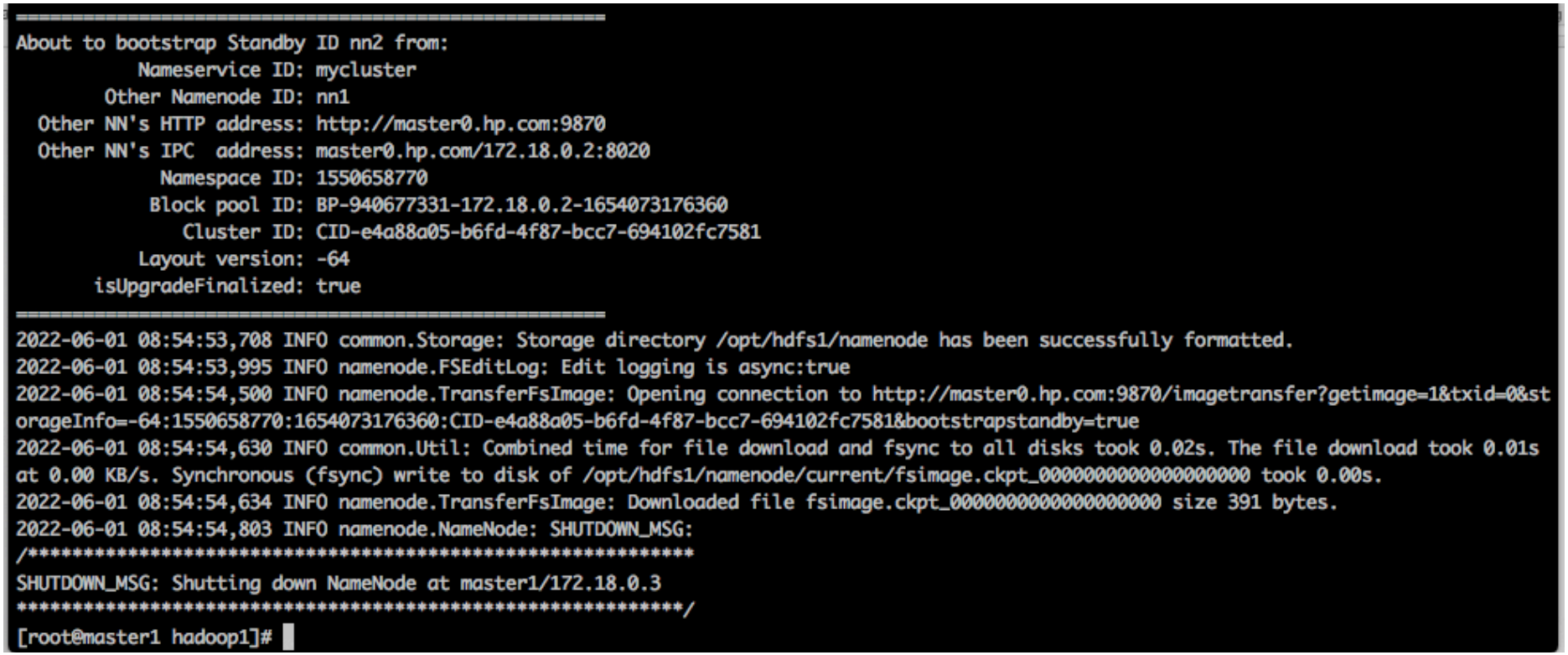
Once the JournalNodes have been started, one must initially synchronize the two HA NameNodes' on-disk metadata.

On Master node, Stop the Namenode (hadoop0) – Jps and Use kill -9 <the process ID>
Accept yes when ask for reformatted.
hdfs --daemon stop namenode
#hdfs namenode -format
#hdfs --daemon start namenode

Our case - If you have already formatted the NameNode, or are converting a non-HA-enabled cluster to be HA-enabled, you should now copy over the contents of your NameNode metadata directories to the other, unformatted NameNode(s) by running the command "*hdfs namenode -bootstrapStandby*" on the unformatted NameNode(s). Running this command will also ensure that the JournalNodes (as configured by **dfs.namenode.shared.edits.dir**) contain sufficient edits transactions to be able to start both NameNodes.

On Secondary namenode; master1/hadoop1 to sync the meta data.

hdfs namenode –bootstrapStandby



At this point you may start all your HA NameNodes as you normally would start a NameNode.

Master 1 / hadoop1

#hdfs --daemon start namenode
hdfs haadmin -getAllServiceState

```
[root@hadoop1 hadoop]# hdfs haadmin -getAllServiceState
hadoop0:8020                                       standby
hadoop1:8020                                       standby
[root@hadoop1 hadoop]#
```

Both the Name Nodes are in standby Mode.

Get service state by service ID.

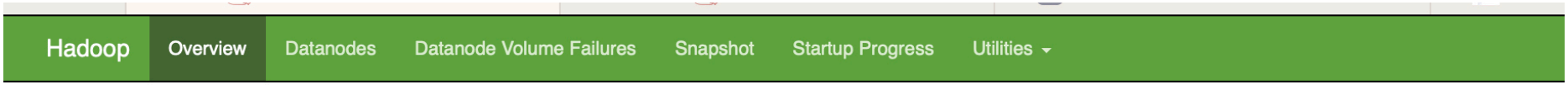
hdfs haadmin -getServiceState nn2

```
[root@hadoop1 hadoop]# hdfs haadmin -getServiceState nn2
standby
[root@hadoop1 hadoop]# hdfs haadmin -getServiceState nn1
standby
[root@hadoop1 hadoop]#
```

You can enable active NN on master0/hadoop0:

```
# hdfs haadmin -transitionToActive nn1
```

You can visit each of the NameNodes' web pages separately by browsing to their configured HTTP addresses. You should notice that next to the configured address will be the HA state of the NameNode (either "standby" or "active".) Whenever an HA NameNode starts, it is initially in the Standby state.
http://localhost:9870/dfshealth.html#tab-overview



Overview 'hadoop0:8020' (active)

Namespace:	mycluster
Namenode ID:	nn1
Started:	Tue Feb 21 10:44:45 +0530 2023
Version:	3.1.2, r1019dde65bcf12e05ef48ac71e84550d589e5d9a
Compiled:	Tue Jan 29 07:09:00 +0530 2019 by sunilg from branch-3.1.2
Cluster ID:	CID-00627d29-7cc2-4504-aabb-64be61d9ba8b
Block Pool ID:	BP-1782287257-172.18.0.2-1676956077648

Clean the directory of all data nodes before starting it else it will get errors due to mismatch in the cluster ID.

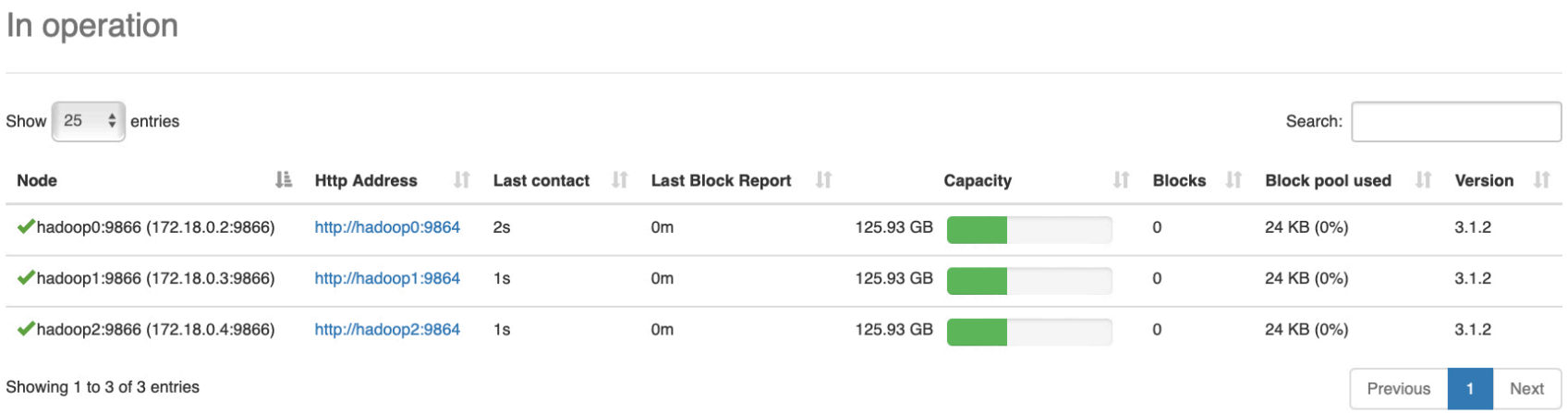
Stop datanode on all 3 nodes.

#hdfs --daemon stop datanode

Start Datanode on all 3 Nodes:

hdfs --daemon start datanode

Verify the Datanodes: You should have all the 3 Nodes listed as shown below else verify the services and logs.



You should have 3 data nodes in the Cluster.

Let us create a directory and move a file to the cluster, From Master0/hadoop0.

```
# hdfs dfs -mkdir /henry
```

Copy Readme file from /opt/hadoop folder.

```
#cd /opt/hadoop
#hdfs dfs -copyFromLocal README.txt /henry
```

Verify the files.

```
#hdfs dfs -ls -R /henry
# hdfs dfs -cat /henry/README.txt
```

Let us switch the NameNode.

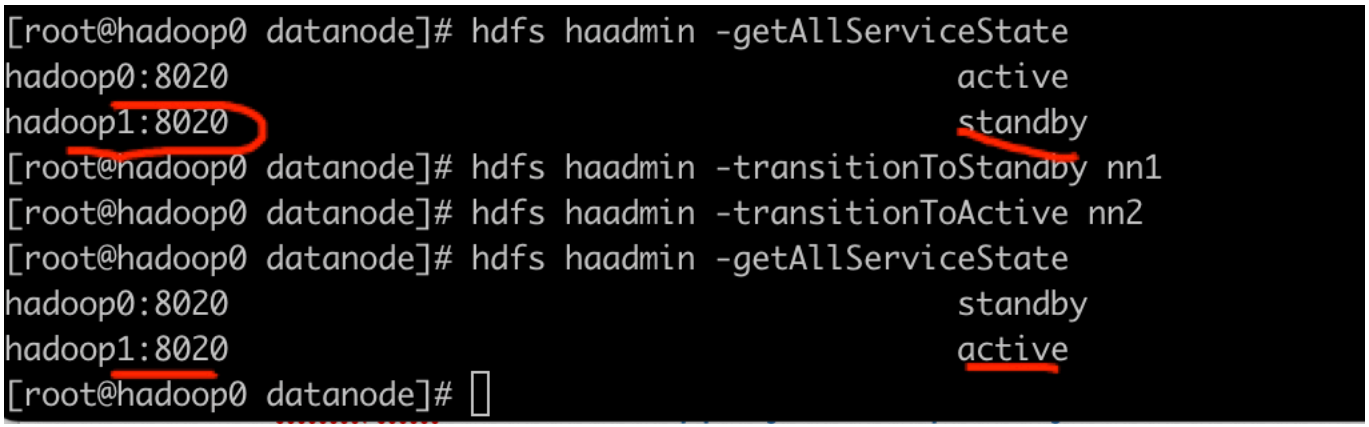
Determine the state of all name Node.

```
#hdfs haadmin -getAllServiceState
```

```
[root@hadoop0 datanode]# hdfs haadmin -getAllServiceState
hadoop0:8020                                       active
hadoop1:8020                                       standby
[root@hadoop0 datanode]#
```

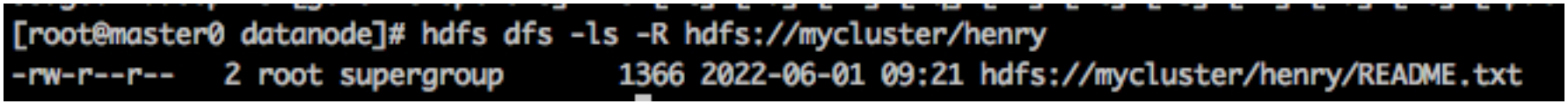
In my case, master0/hadoop0 is active. Let us make the master1/hadoop1 as active NN.
It's a two steps process. Bring the active to standby and standby to active.

hdfs haadmin -transitionToStandby nn1
hdfs haadmin -transitionToActive nn2
hdfs haadmin -getAllServiceState



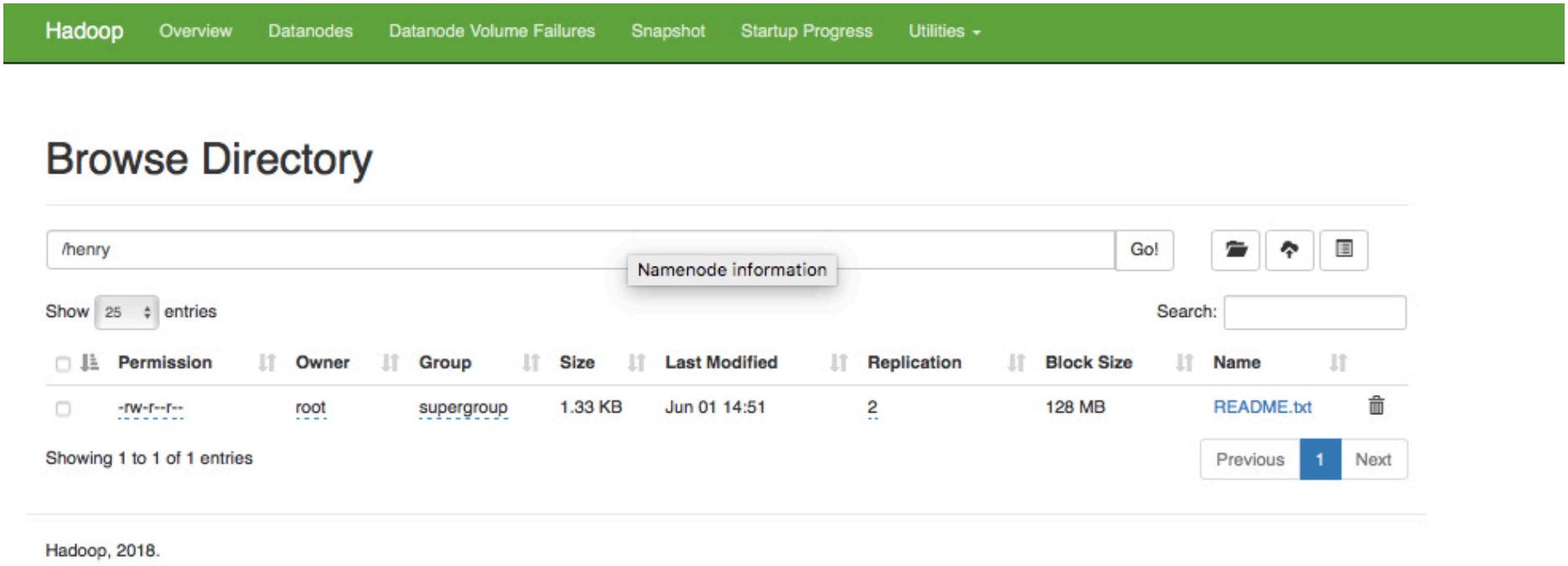
Check that all data in the cluster are intact. Accessing through the cluster Name

hdfs dfs -ls -R hdfs://mycluster/henry



Click on the master1/hadoop1 namenode URL.

http://localhost:9871/explorer.html#/henry



Completed the lab execution.

Reference purpose only.
Core-site.xml

```
<configuration>
   <property>
      <name>fs.defaultFS</name>
      <value>hdfs://hadoop0:8020</value>
   </property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
<description>Buffer size</description>
</property>

</configuration>
```

------- Begin hdfs.site.xml ----------

```
<configuration>
   <property>
      <name>fs.defaultFS</name>
      <value>hdfs://hadoop0:8020</value>
   </property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
<description>Buffer size</description>
</property>

</configuration>
```

```
[root@hadoop1 hadoop]# more hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///opt/hdfs/namenode</value>
<description>NameNode directory for namespace and transaction logs storage.</description>
</property>
<property>
<name>dfs.datanode.data.dir</name>
```

```
<value>file:///opt/hdfs/datanode</value>
<description>DataNode directory</description>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.datanode.use.datanode.hostname</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
<value>false</value>
</property>
<property>
  <name>dfs.nameservices</name>
  <value>mycluster</value>
</property>
<property>
  <name>dfs.ha.namenodes.mycluster</name>
  <value>nn1,nn2</value>
</property>
```

```
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn1</name>
  <value>hadoop0:8020</value>
</property>
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn2</name>
  <value>hadoop1:8020</value>
</property>
<property>
  <name>dfs.namenode.shared.edits.dir</name>
<value>qjournal://hadoop0:8485;hadoop1:8485;hadoop2:8485/mycluster</value>
</property>
<property>
  <name>dfs.client.failover.proxy.provider.mycluster</name>
  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
<property>
  <name>dfs.journalnode.edits.dir</name>
  <value>/opt/journal/data</value>
</property>


</configuration> ------- Ends hdfs.site.xml   ----------------------------
```

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html

---------- Lab Ends Here ----------

9. ResourceManager high availability – 2 Hrs (D)

You must have at least three nodes in the cluster:

Pre-requisite:

Configure HA HDFS Cluster.

Steps to start HA HDFS Cluster:

Start Journal In all the three nodes.

```
#hdfs --daemon start journalnode
```

Start Name Node on hadoop0 and hadoop1

```
#hdfs --daemon start namenode
```

Activate one Namenode as active – hadoop0.

```
#hdfs haadmin -transitionToActive nn1
```

Start All datanodes (hadoop0/hadoop1/hadoop2)

```
#hdfs --daemon start datanode
```

At the end of this lab, you will have services as shown below in the cluster.

hostname	Remarks	Services
Hadoop0	First Machine	NN, DN , JN , RM , NM
hadoop1	2nd Machine	NN , DN , JN , RM , NM
hadoop2	New Machine	JN , ZK – 1 , 2 , 3 , NM

At this point we have only one Resource Manager configured in the cluster.

Stop Resource manager and Yarn Node manager if started.

[

```
yarn --daemon stop resourcemanager
yarn --daemon stop nodemanager
```

]

Add the following configuration in the /opt/hadoop/etc/hadoop/**yarn-site.xml** on all nodes.

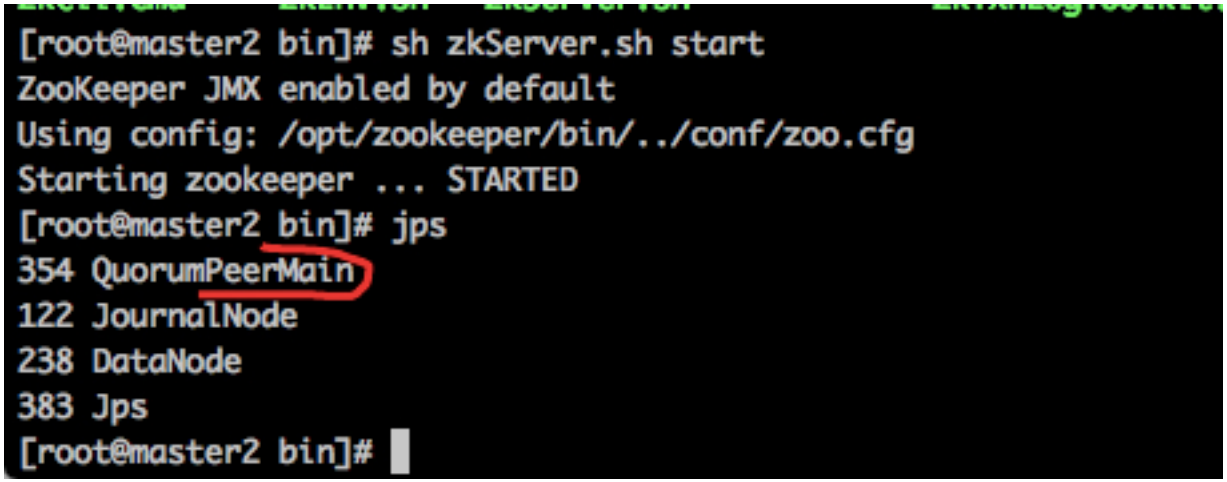
```
<property>
  <name>yarn.resourcemanager.ha.enabled</name>
  <value>true</value>
</property>
<property>
  <name>yarn.resourcemanager.cluster-id</name>
  <value>cluster1</value>
</property>
<property>
```

```
  <name>yarn.resourcemanager.ha.rm-ids</name>
  <value>rm1,rm2</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname.rm1</name>
  <value>hadoop0</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname.rm2</name>
  <value>hadoop1</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm1</name>
  <value>hadoop0:8088</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm2</name>
  <value>hadoop1:8088</value>
</property>
<property>
  <name>hadoop.zk.address</name>
  <value>hadoop2:2181</value>
</property>
<property>
   <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
    <description>Whether virtual memory limits will be enforced for containers</description>
```

```
  </property>
 <property>
   <name>yarn.nodemanager.vmem-pmem-ratio</name>
    <value>4</value>
    <description>Ratio between virtual memory to physical memory when setting memory limits for
containers</description>
  </property>
```

Install zookeeper on master2/hadoop2 node and start it.

Refer the Lab-Zookeeper.pdf

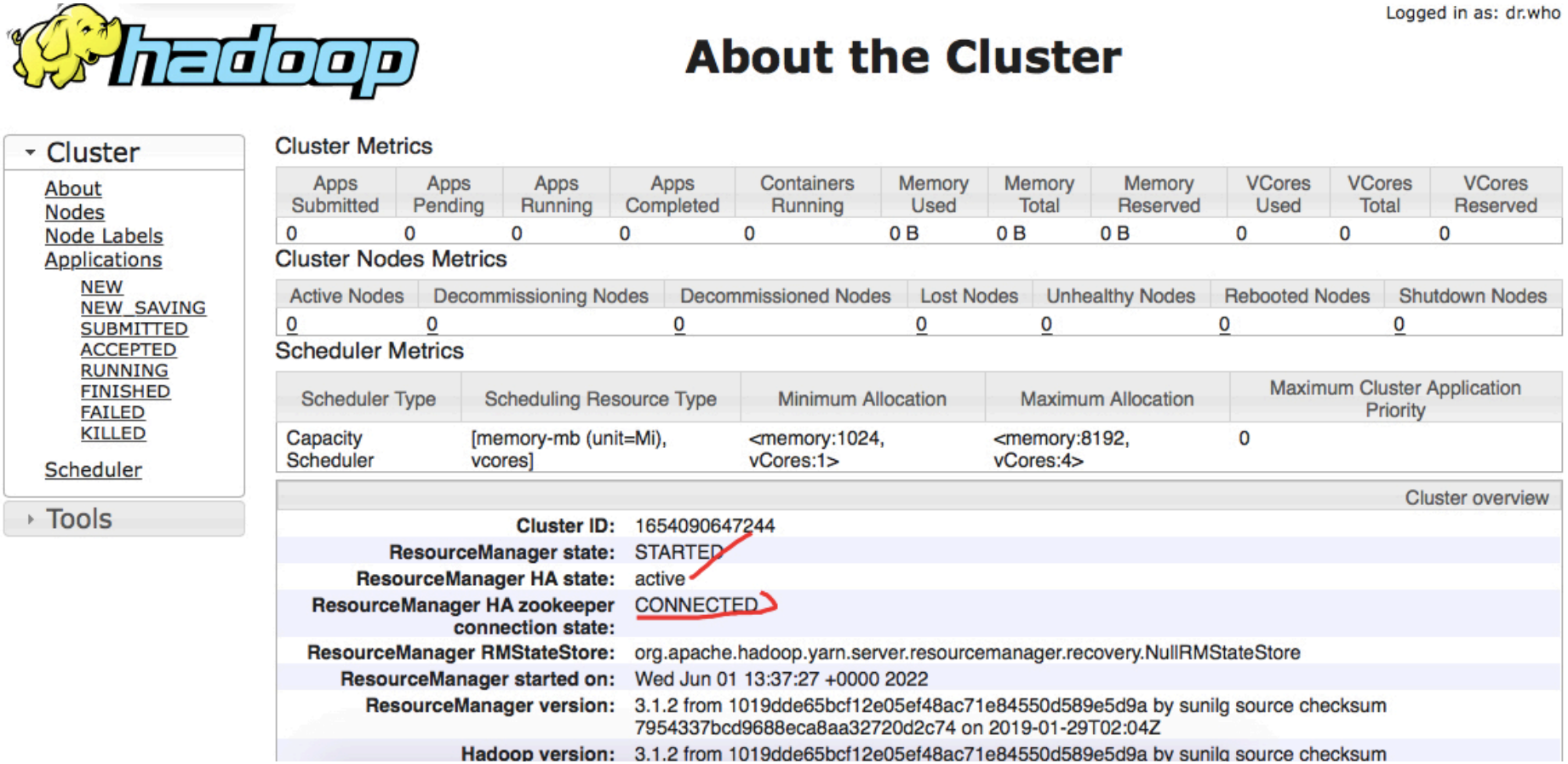


Start Resource Manager on both the node: master0 and master1

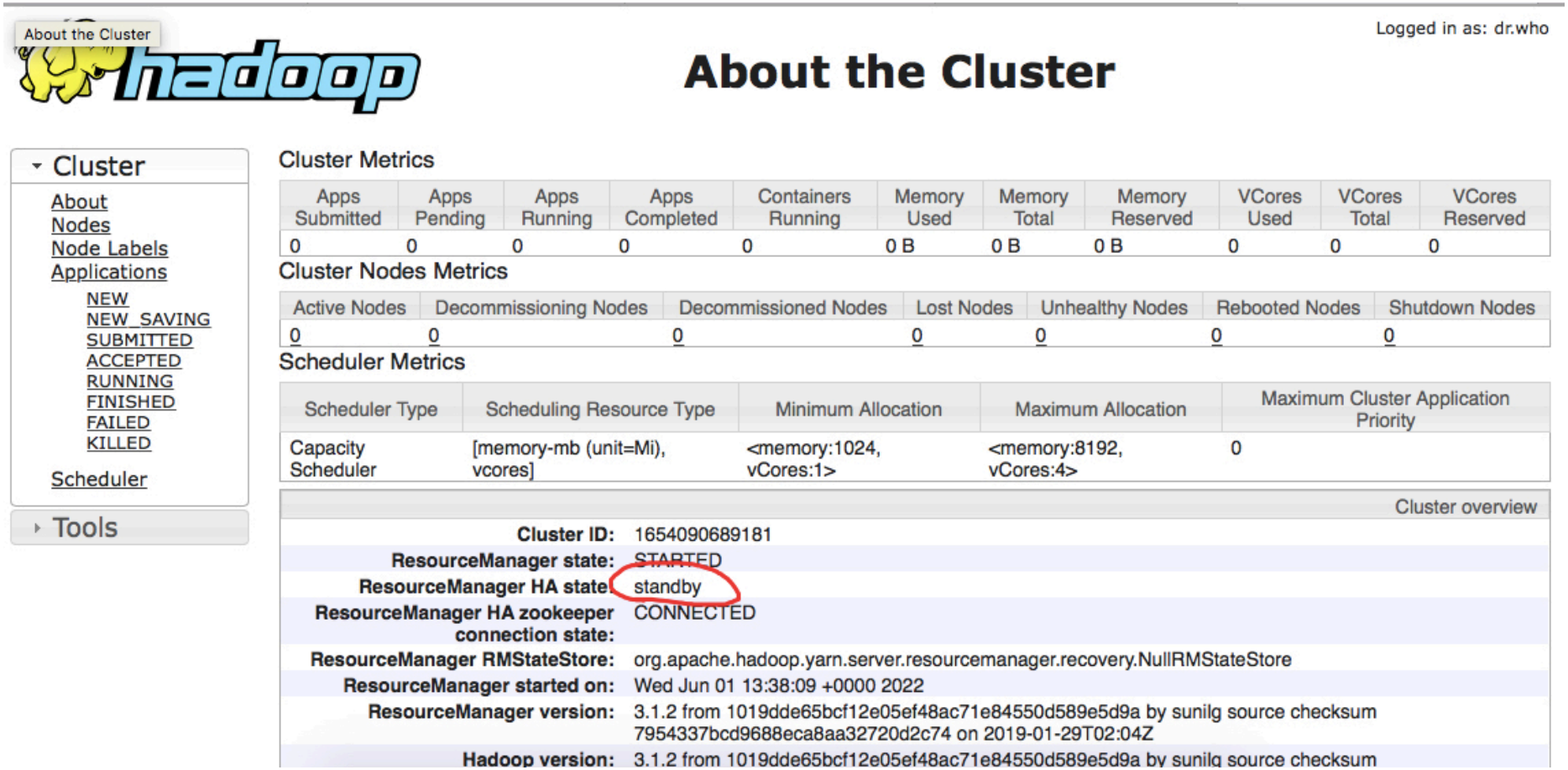
yarn --daemon start resourcemanager

You can verify the resource manager on both the server using the following URL.
http://localhost:8088/cluster/cluster

Master0/hadoop0



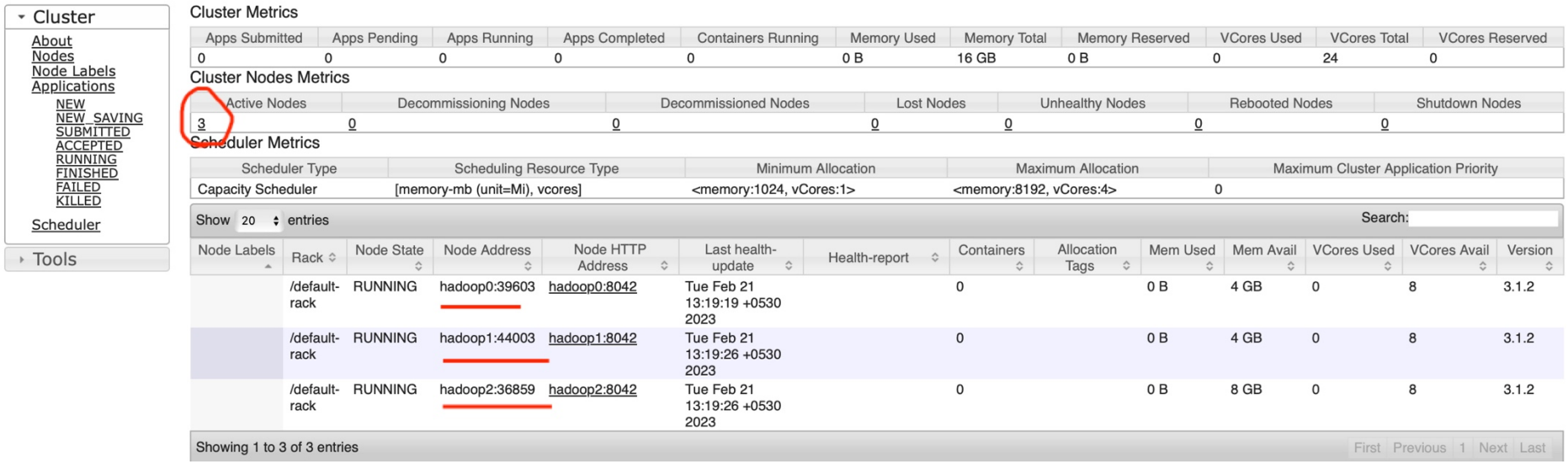
master1/hadoop1 – Standby



Start NodeManager: On all the three nodes.

#yarn --daemon start nodemanager

Verify the number of Nodes using the RM UI.



`yarn rmadmin` has a few HA-specific command options to check the health/state of an RM, and transition to Active/Standby. Commands for HA take service id of RM set by `yarn.resourcemanager.ha.rm-ids` as argument.

$ yarn rmadmin -getServiceState rm1

active

$ yarn rmadmin -getServiceState rm2

standby

Submit a yarn job to the cluster.

Make the HDFS directories required to execute MapReduce jobs:

Verify the folders.

#hdfs dfs -ls -R /

```
[root@hadoop1 logs]# hdfs dfs -ls -R /
drwxr-xr-x   - root supergroup          0 2021-04-05 11:28 /user
drwxr-xr-x   - root supergroup          0 2021-04-05 11:28 /user/root
[root@hadoop1 logs]#
```

If the above folders doesn't exist, the create as shown below.
$hdfs dfs -mkdir /user
$hdfs dfs -mkdir /user/root

Copy the input files into the distributed filesystem:
$ hdfs dfs -mkdir input
$ hdfs dfs -put /opt/hadoop/etc/hadoop/*.xml input

Verify the files

hdfs dfs -ls input

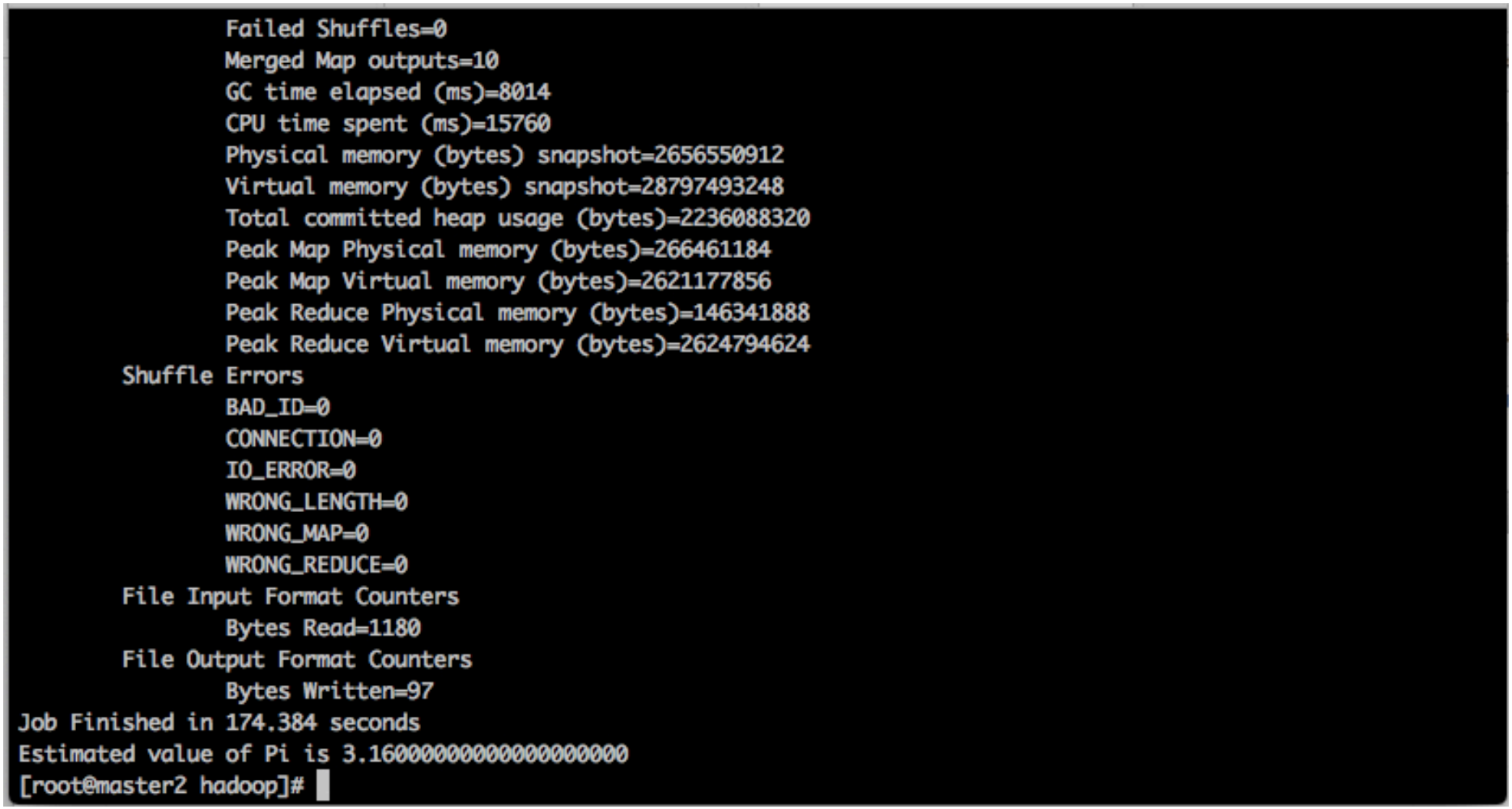
```
[root@hadoop1 logs]# hdfs dfs -ls input
Found 9 items
-rw-r--r--   2 root supergroup       9213 2021-04-06 03:38 input/capacity-scheduler.xml
-rw-r--r--   2 root supergroup        995 2021-04-06 03:38 input/core-site.xml
-rw-r--r--   2 root supergroup      11392 2021-04-06 03:38 input/hadoop-policy.xml
-rw-r--r--   2 root supergroup       1454 2021-04-06 03:38 input/hdfs-site.xml
-rw-r--r--   2 root supergroup        620 2021-04-06 03:38 input/httpfs-site.xml
-rw-r--r--   2 root supergroup       3518 2021-04-06 03:38 input/kms-acls.xml
-rw-r--r--   2 root supergroup        682 2021-04-06 03:38 input/kms-site.xml
-rw-r--r--   2 root supergroup       1060 2021-04-06 03:38 input/mapred-site.xml
-rw-r--r--   2 root supergroup       1210 2021-04-06 03:38 input/yarn-site.xml
[root@hadoop1 logs]#
```

Let's submit a MapReduce job to the cluster

In the Terminal – hadoop2/ master1/hadoop1 node
#export YARN_EXAMPLES=/opt/hadoop/share/hadoop/mapreduce
yarn jar $YARN_EXAMPLES/hadoop-mapreduce-examples-3.1.2.jar pi 10 30

or

#hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar grep input output 'dfs[a-z.]+'



Reference.

yarn-site.xml

```
<configuration>
<property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>
   <property>
      <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASS
PATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HO
ME</value>
   </property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>file:///opt/yarn/local</value>
</property>
<property>
 <name>yarn.application.classpath</name>
```

```
 <value>
$HADOOP_CONF_DIR,$HADOOP_COMMON_HOME/share/hadoop/common/*,$HADOOP_COMMON_H
OME/share/hadoop/common/lib/*,$HADOOP_HDFS_HOME/share
/hadoop/hdfs/*,$HADOOP_HDFS_HOME/share/hadoop/hdfs/lib/*,$HADOOP_MAPRED_HOME/share/ha
doop/mapreduce/*,$HADOOP_MAPRED_HOME/share/hadoop/map
reduce/lib/*,
 $HADOOP_YARN_HOME/share/hadoop/yarn/*,$HADOOP_YARN_HOME/share/hadoop/yarn/lib/*
 </value>
</property>

<!-- Site specific YARN configuration properties -->
 <property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
   <value>false</value>
   <description>Whether virtual memory limits will be enforced for containers</description>
 </property>
 <property>
  <name>yarn.nodemanager.vmem-pmem-ratio</name>
   <value>4</value>
   <description>Ratio between virtual memory to physical memory when setting memory limits for
containers</description>
 </property>
<property>
 <name>yarn.nodemanager.resource.memory-mb</name>
 <value>4096</value>
 </property>
<property>
 <name>yarn.resourcemanager.scheduler.class</name>
```

```
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
 </property>
<property>
  <name>yarn.resourcemanager.ha.enabled</name>
  <value>true</value>
</property>
<property>
  <name>yarn.resourcemanager.cluster-id</name>
  <value>cluster1</value>
</property>
<property>
  <name>yarn.resourcemanager.ha.rm-ids</name>
  <value>rm1,rm2</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname.rm1</name>
  <value>hadoop0</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname.rm2</name>
  <value>hadoop1</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm1</name>
  <value>hadoop0:8088</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm2</name>
```

```
  <value>hadoop1:8088</value>
</property>
<property>
  <name>hadoop.zk.address</name>
  <value>hadoop2:2181</value>
</property>
<property>
   <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
    <description>Whether virtual memory limits will be enforced for containers</description>
  </property>
 <property>
   <name>yarn.nodemanager.vmem-pmem-ratio</name>
    <value>4</value>
    <description>Ratio between virtual memory to physical memory when setting memory limits for
containers</description>
  </property>

</configuration>---------------------------------------- Lab Ends Here ----------------------------------
```

10. Archival Disk Storage – 60 Minutes(D)

Prerequisite: Configure a Two Node HDFS cluster without HA enable.

Shut down the HDFS cluster and YARN Cluster.

Copy the earlier hdfs-site.xml, core-site.xml and yarn-site.xml.

Clean the data directory or Namenode and data node.

Format the Name node and Start the HDFS

On hadoop0

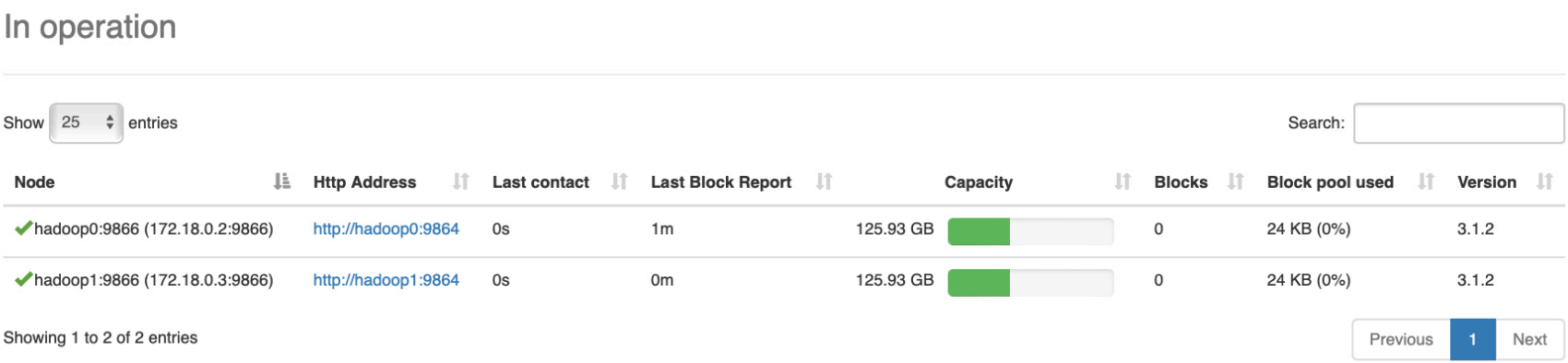
```
#hdfs namenode -format

#start-dfs.sh
```

On hadoop1

```
#hdfs --daemon start datanode
```

At this point, your web UI should be as shown below.



Using the COLD storage policy

Consider a 2-node cluster in which one Node is configured as archive storage. This node has a single disk and limited computing resources.

1. Attach the archival storage On the Second Node(master1).

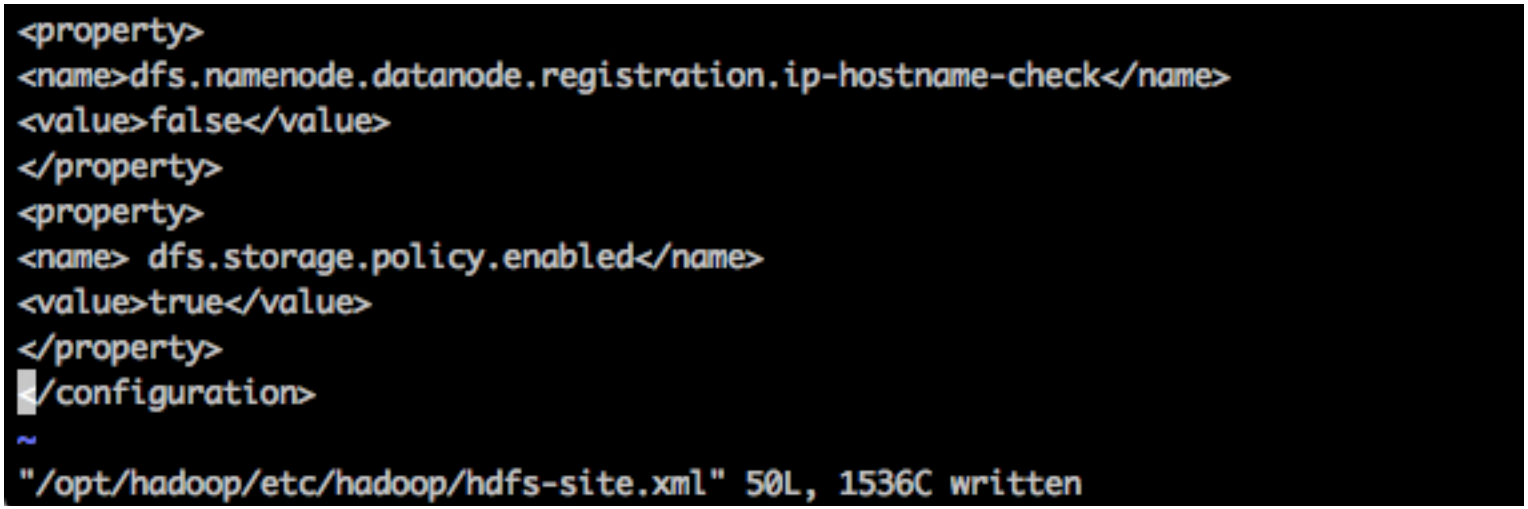
Create the following directory in ***master1/hadoop1*** node using mkdir with root.

#mkdir /opt/hdfs/archive

Enable a storage policy on both the nodes.

Add the dfs.storage.policy.enabled property to the /opt/hadoop/etc/hadoop/hdfs-site.xml.

```
<property>
<name> dfs.storage.policy.enabled</name>
<value>true</value>
</property>
```



Save it. Accept all the default setting.

Configure an archive data node – On the Second Node (master1/hadoop1).

Navigate to **hdfs-site.xml** and replace/add the archival storage volumes to the data node directories, as shown in the following figure:

[ARCHIVE]file:///opt/hdfs/archive

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:///opt/hdfs/namenode</value>
<description>NameNode directory for namespace and transaction logs storage.</description>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>[ARCHIVE]file:///opt/hdfs/archive</value>
<description>DataNode directory</description>
</property>
<property>
```

Save the file.

After the changes are saved, restart the HDFS or kill using -9 the process and start the process .

On hadoop0
start-dfs.sh

On hadoop1
#hdfs --daemon start datanode

Determine an existing file using file viewer or upload using hdfs command.

hdfs dfs -mkdir -p /user/root
#hdfs dfs -put pg20417.txt /user/root

#hdfs dfs -ls -R /user/root

```
[root@master1 opt]# hdfs dfs -ls -R /user/root
-rw-r--r--   2 root supergroup     674566 2022-06-08 13:58 /user/root/pg20417.txt
[root@master1 opt]#
```

In our case we will archive the pg20417.txt file.

Apply the COLD storage policy to a directory. As the hdfs user, assign the COLD storage policy to an HDFS directory by running the following commands from the command line:

```
#hdfs dfs -mkdir /archives
#hdfs dfs -chmod 777 /archives
#hdfs storagepolicies -setStoragePolicy -path /archives -policy COLD
```

After setup is complete, verify the policies by running the following command:

```
#hdfs storagepolicies -getStoragePolicy -path /archives
```

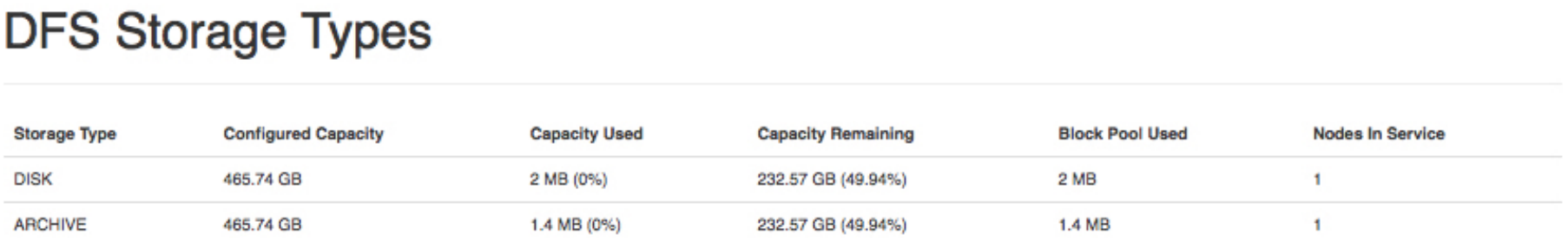
Output from this command will be similar to the following example:

```
[root@tos log]# su - hdfs
Last login: Sat Aug 17 19:48:12 IST 2019
[hdfs@tos ~]$ hdfs dfs -mkdir /archives
[hdfs@tos ~]$ hdfs dfs -chmod 777 /archives
[hdfs@tos ~]$ hdfs storagepolicies -setStoragePolicy -path /archives -policy COL
D
Set storage policy COLD on /archives
[hdfs@tos ~]$ hdfs storagepolicies -getStoragePolicy -path /archives
The storage policy of /archives:
BlockStoragePolicy{COLD:2, storageTypes=[ARCHIVE], creationFallbacks=[], replica
tionFallbacks=[]}
[hdfs@tos ~]$
```

Verify the setup.

http://localhost:9870/dfshealth.html#tab-overview

And scroll to the **DFS Storage Types** Information.



DFS Storage Types

Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service
DISK	465.74 GB	2 MB (0%)	232.57 GB (49.94%)	2 MB	1
ARCHIVE	465.74 GB	1.4 MB (0%)	232.57 GB (49.94%)	1.4 MB	1

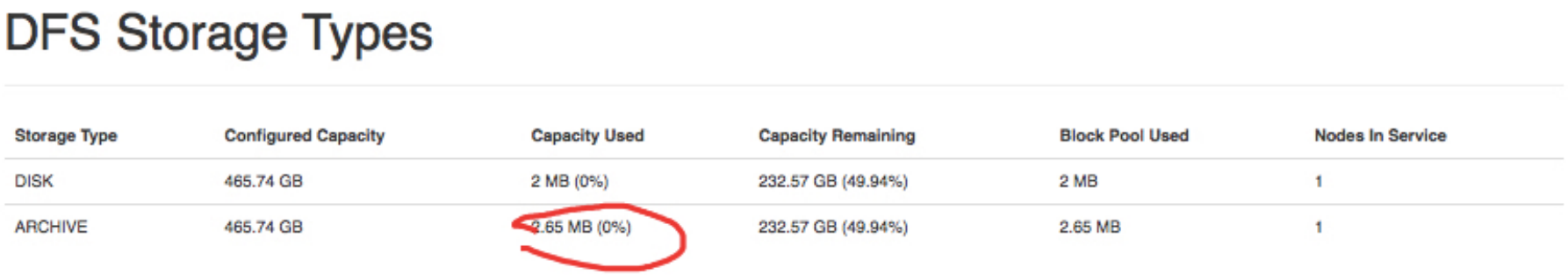
Copy a file to the /archives directory, as shown in the following example:

```
hdfs dfs -cp /user/root/pg*  /archives/
hdfs dfs -setrep 1 /archives/pg*
```

Note: Because this example shows only one archive data node, the replication factor for the file that was uploaded to archive storage must be set to 1; otherwise, there will not be enough data nodes for other replicas.

If you check the blocks on this data node again, it now returns a nonzero value that depends on the size of the uploaded file, as shown in the following figure:

Refresh the datanode page.

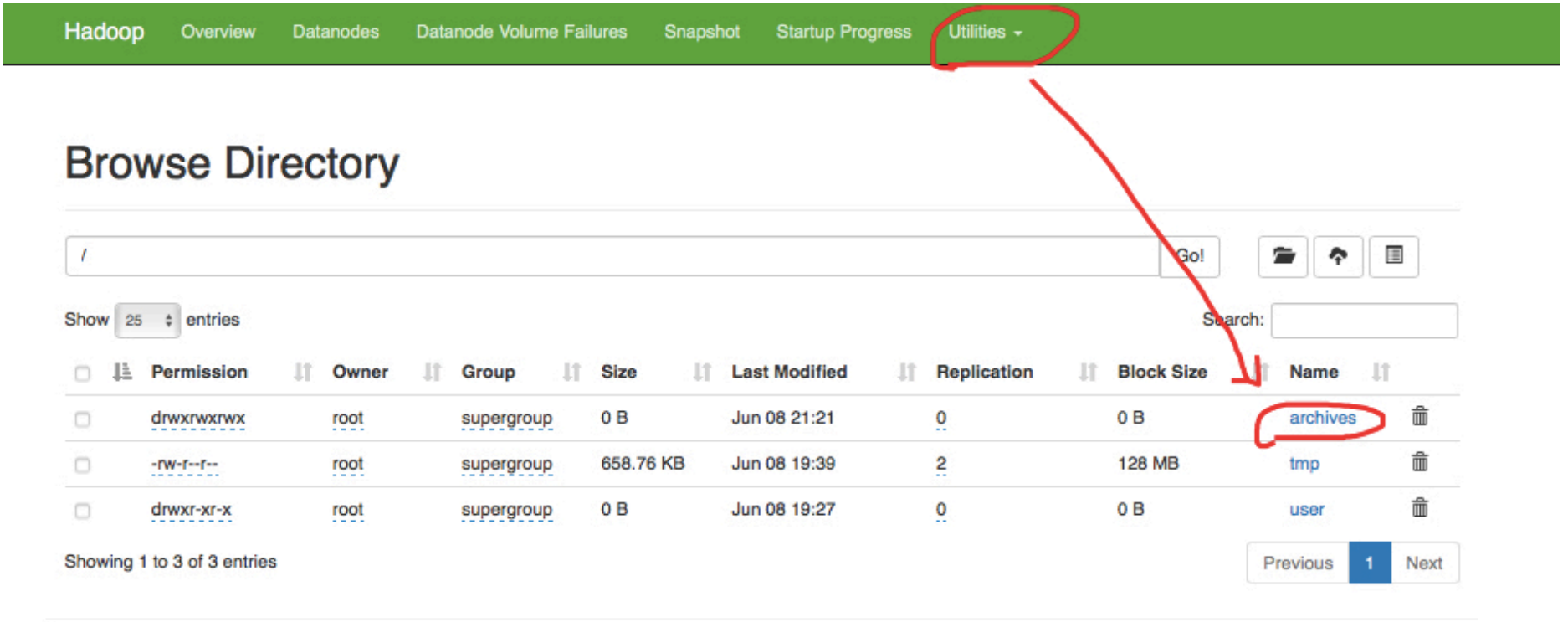


DFS Storage Types

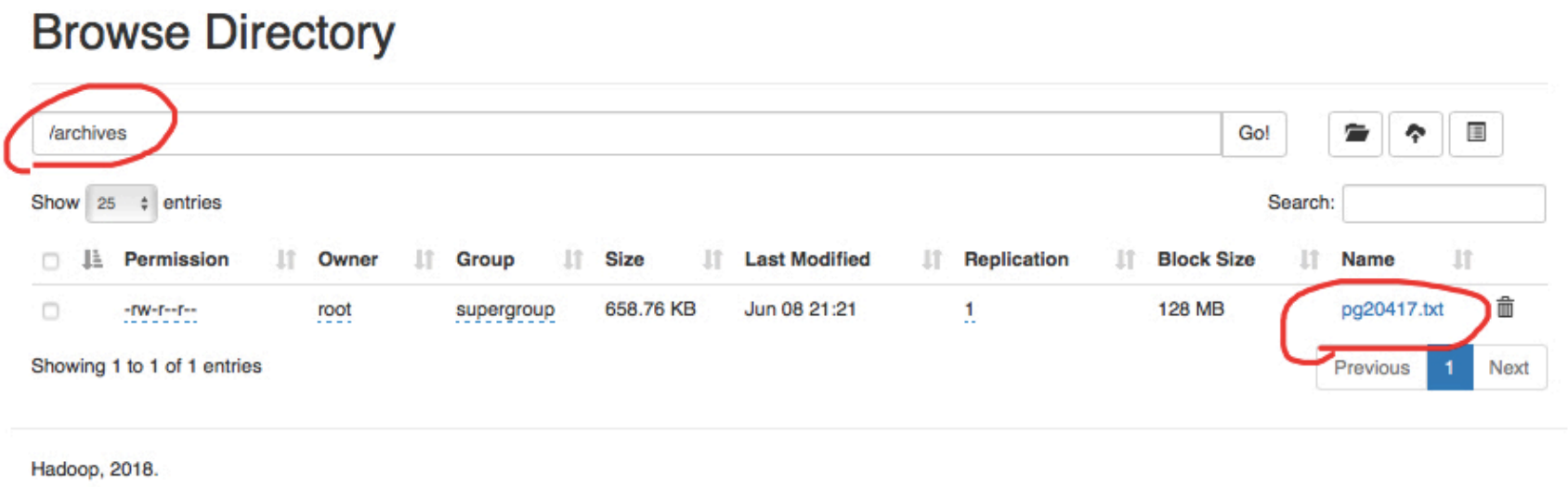
Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service
DISK	465.74 GB	2 MB (0%)	232.57 GB (49.94%)	2 MB	1
ARCHIVE	465.74 GB	2.65 MB (0%)	232.57 GB (49.94%)	2.65 MB	1

This number does not change if you upload files to directories other than /archives.

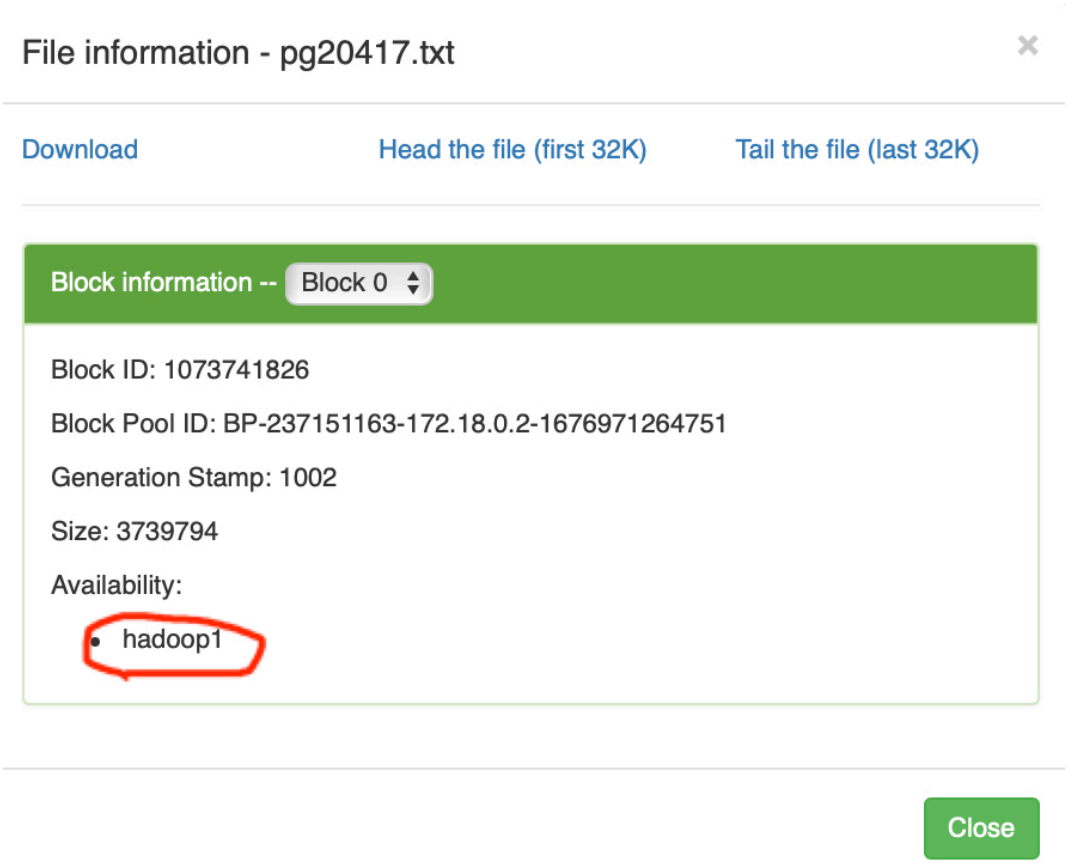
Click on the Utilities



Next → archives -> pg20417.txt



Click on the above file.



As shown above the file get stored in the master1/hadoop1, second node only. Since we have configured archive policy in the above node.

---------- Lab Ends Here ----------

11. Hive Installation – 120 Minutes (D)

In this lab, you will configure Hive for submitting job to YARN Cluster.

All the steps mention below need to be performed On the **hadoop2** node only.

Install jdk 8 and replace the earlier jdk. It only work with jdk 1.8.

Unpack the hive tarball in /opt folder. This will result in the creation of a subdirectory named hive-x.y.z (where x.y.z is the release number):

For manageability, rename the Hive folder. From now onwards HIVE_HOME will be referred to /opt/hive.

```
# tar -xvf apache*hive-* -C /opt
```

```
#cd /opt
```

```
# mv apache-hive-*-bin hive
```

Set the environment variable HIVE_HOME to point to the installation directory:
And add $HIVE_HOME/bin to your PATH:

```
#vi ~/.bashrc
export HIVE_HOME=/opt/hive
```

```
export PATH=$HIVE_HOME/bin:$PATH
```

Running Hive
Hive uses Hadoop, so you must have Hadoop in your path.

```
#export HADOOP_HOME=/opt/hadoop
```

Update the above in bash profile.

Type the following to reinitialize the scripts.

```
#bash
```

Start HDFS cluster – On only hadoopo
```
#start-dfs.sh
```

In addition, you must use below HDFS commands to
create /tmp and /user/hive/warehouse (aka hive.metastore.warehouse.dir) and set some permission to them using chmod g+w before you can create a table in Hive.

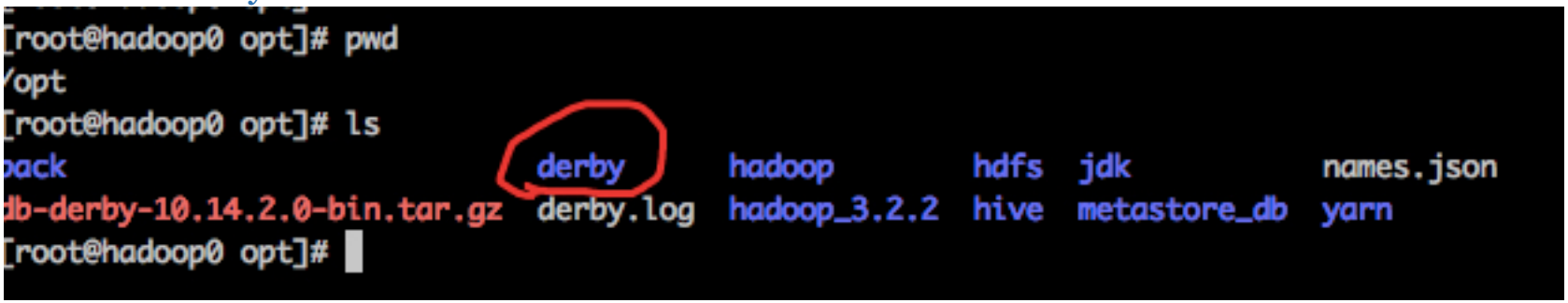
```
$ hdfs dfs -mkdir   /tmp
$ hdfs dfs -mkdir   -p  /user/hive/warehouse
$ hdfs dfs -chmod g+w   /tmp
$ hdfs dfs -chmod g+w   /user/hive/warehouse
```

You may find it useful, though it's not necessary, to set HIVE_HOME:
$ export HIVE_HOME=/opt/hive

In order to store the metadata, we will install and configure Apache Derby.

Configure derby Network.

Extract derby in /opt and rename into derby folder.
tar -xvf db-derby-10.14.2.0-bin.tar -C /opt
#cd /opt
#mv db* derby

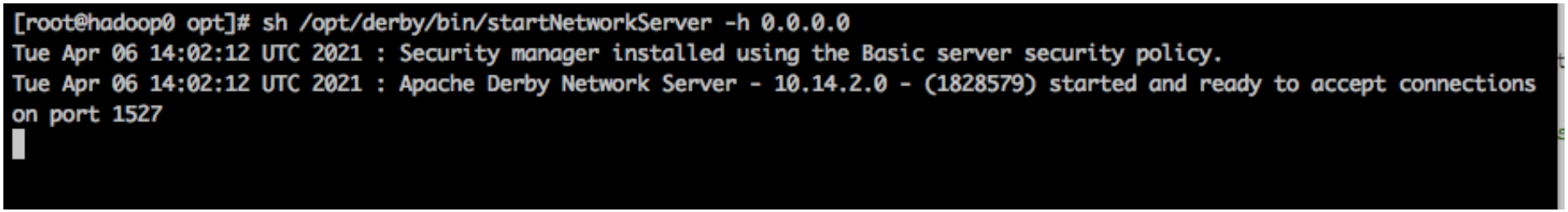


export DERBY_INSTALL=/opt/derby
export DERBY_HOME=/opt/derby

By default, Derby will create databases in the directory it was started from.

Execute the following command always from /opt folder.

#sh /opt/derby/bin/startNetworkServer -h 0.0.0.0



Go to the HIVE_HOME/conf folder and copy the following file. All hive configuration will be done in the following file.

Configure Hive to Use Network Derby

Edit /opt/hive/conf/hive-site.xml as follows. Create the file if it doesn't exist. Replace the hostname of the db server as appropriate

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
<property>

```
 <name>javax.jdo.option.ConnectionURL</name>
 <value>jdbc:derby://hadoop2:1527/metastore_db;create=true</value>
 <description>
JDBC connect string for a JDBC metastore.
 </description>
 </property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>org.apache.derby.jdbc.ClientDriver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>hive.server2.active.passive.ha.enable</name>
<value>true</value> # change false to true
</property>
<property>
<name>hive.server2.enable.doAs</name>
<value>FALSE</value>
<description>Hive
</description>
</property>
</configuration>
```

Copy the following file and add the below last line in the sh file.

```
#cd /opt/hive/conf
#cp hive-env.sh.template hive-env.sh
```

Update the following in the hive-env.sh

```
export HIVE_CONF_DIR=/opt/hive/conf
```

Copy Derby Jar Files

Now since there is a new client you *MUST* make sure Hive has these jar files in the lib directory or in the classpath. The same would be true if you used MySQL or some other DB.

```
#cp /opt/derby/lib/derbyclient.jar /opt/hive/lib
#cp /opt/derby/lib/derbytools.jar /opt/hive/lib
```

Initiate Derby Database - Running HiveServer2 and Beeline

Apache Hive uses the Derby database to store metadata. Initiate the Derby database, from the Hive *bin* directory using the **schematool** command: (metastore_db database will be created in the folder where the derby gets started.)

#cd /opt

#schematool -initSchema -dbType derby –verbose

```
[root@hadoop0 hive]# schematool -initSchema -dbType derby
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:        jdbc:derby://hadoop0:1527/metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.ClientDriver
Metastore connection User:       APP
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
```

```
Initialization script completed
schemaTool completed
[root@hadoop0 hive]#
```

Configure Hive log in HIVE_HOME/conf folder.

Copy the file and update the following.

#cp hive-log4j2.properties.template hive-log4j2.properties
Update the following properties in the above file.

property.hive.log.dir = /opt/log

To run HiveServer2 and Beeline from shell:
#cd /opt

$ hiveserver2

Connect the Hive beeline using the following command.

```
$ beeline -u  jdbc:hive2://hadoop2:10000
```

Execute the following command in the Beeline CLI to create a table and display it.

```
#CREATE TABLE pokes (foo INT, bar STRING);
#SHOW TABLES;
```

```
[root@hadoop0 /]# beeline -u  jdbc:hive2://hadoop0:10000
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/Stat
icLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://hadoop0:10000
Connected to: Apache Hive (version 3.1.2)
Driver: Hive JDBC (version 3.1.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.2 by Apache Hive
0: jdbc:hive2://hadoop0:10000> CREATE TABLE pokes (foo INT, bar STRING);
INFO  : Compiling command(queryId=root_20210407030650_2eb03293-87f7-488d-b286-1e8f72c7d048): CRE
ATE TABLE pokes (foo INT, bar STRING)
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=root_20210407030650_2eb03293-87f7-488d-b286-1e8f72c7
d048); Time taken: 1.114 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=root_20210407030650_2eb03293-87f7-488d-b286-1e8f72c7d048): CRE
ATE TABLE pokes (foo INT, bar STRING)
```

```
0: jdbc:hive2://hadoop0:10000> SHOW TABLES;
INFO  : Compiling command(queryId=root_20210407030707_f1a34c71-a2ce-47de-8194-a78df5a361a7): SHO
W TABLES
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comm
ent:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=root_20210407030707_f1a34c71-a2ce-47de-8194-a78df5a3
61a7); Time taken: 0.144 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=root_20210407030707_f1a34c71-a2ce-47de-8194-a78df5a361a7): SHO
W TABLES
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=root_20210407030707_f1a34c71-a2ce-47de-8194-a78df5a3
61a7); Time taken: 0.049 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-----------+
| tab_name  |
+-----------+
| pokes     |
+-----------+
1 row selected (0.462 seconds)
0: jdbc:hive2://hadoop0:10000> select * from pokes;
```

Execute the following command to load data from a text file which is in local parth

#LOAD DATA LOCAL INPATH '/opt/hive/examples/files/kv1.txt' OVERWRITE INTO TABLE pokes;

#select * from pokes;

```
0: jdbc:hive2://hadoop0:10000> LOAD DATA LOCAL INPATH '/opt/hive/examples/files/kv1.txt' OVERWRI
TE INTO TABLE pokes;
INFO  : Compiling command(queryId=root_20210407033041_0d173e78-25a7-44af-8a66-a865ef4f2800): LOA
D DATA LOCAL INPATH '/opt/hive/examples/files/kv1.txt' OVERWRITE INTO TABLE pokes
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=root_20210407033041_0d173e78-25a7-44af-8a66-a865ef4f
2800); Time taken: 0.267 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=root_20210407033041_0d173e78-25a7-44af-8a66-a865ef4f2800): LOA
D DATA LOCAL INPATH '/opt/hive/examples/files/kv1.txt' OVERWRITE INTO TABLE pokes
INFO  : Starting task [Stage-0:MOVE] in serial mode
INFO  : Loading data to table default.pokes from file:/opt/hive/examples/files/kv1.txt
INFO  : Starting task [Stage-1:STATS] in serial mode
INFO  : Completed executing command(queryId=root_20210407033041_0d173e78-25a7-44af-8a66-a865ef4f
2800); Time taken: 2.76 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
No rows affected (3.069 seconds)
0: jdbc:hive2://hadoop0:10000> select * from pokes;
```

```
0: jdbc:hive2://hadoop0:10000> select * from pokes;
INFO  : Compiling command(queryId=root_20210407033119_22227329-cdf0-4437-b8fa-d8a3249c5dd2): sel
ect * from pokes
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:pokes.foo, type:int, commen
t:null), FieldSchema(name:pokes.bar, type:string, comment:null)], properties:null)
INFO  : Completed compiling command(queryId=root_20210407033119_22227329-cdf0-4437-b8fa-d8a3249c
5dd2); Time taken: 0.397 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=root_20210407033119_22227329-cdf0-4437-b8fa-d8a3249c5dd2): sel
ect * from pokes
INFO  : Completed executing command(queryId=root_20210407033119_22227329-cdf0-4437-b8fa-d8a3249c
5dd2); Time taken: 0.002 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+------------+------------+
| pokes.foo  | pokes.bar  |
+------------+------------+
| 238        | val_238    |
| 86         | val_86     |
| 311        | val_311    |
| 27         | val_27     |
```

If you are able to get the response as shown above, then you have configure Hive to submit job to Hadoop cluster.

---------- Lab Ends Here ----------

12. Erarata

Hadoop Version Mismatch with Hive Libraries.

Error Type: (Exception in thread "main" java.lang.NoSuchMethodError:
com.google.common.base.Preconditions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)

```
[root@hadoop0 bin]# schematool -initSchema -dbType derby
SLF4J: Class path contains multiple SLF4J bin
SLF4J: Found binding in [jar:file:/opt/hive/l                    -impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerB
inder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j
/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkArgumen
t(ZLjava/lang/String;Ljava/lang/Object;)V
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1357)
        at org.apache.hadoop.conf.Configuration.set(Configuration.java:1338)
        at org.apache.hadoop.mapred.JobConf.setJar(JobConf.java:536)
        at org.apache.hadoop.mapred.JobConf.setJarByClass(JobConf.java:554)
        at org.apache.hadoop.mapred.JobConf.<init>(JobConf.java:448)
        at org.apache.hadoop.hive.conf.HiveConf.initialize(HiveConf.java:5141)
        at org.apache.hadoop.hive.conf.HiveConf.<init>(HiveConf.java:5104)
        at org.apache.hive.beeline.HiveSchemaTool.<init>(HiveSchemaTool.java:96)
        at org.apache.hive.beeline.HiveSchemaTool.main(HiveSchemaTool.java:1473)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:323)
```

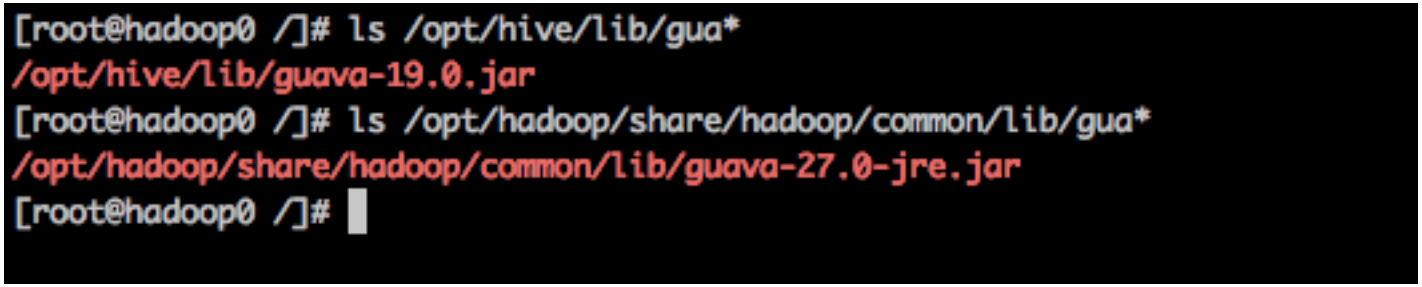
Name: (null)
Profile: (null)
Command: None

Solution:

This issue happened because guava lib version is different in Hive lib folder and Hadoop shared folder.

To fix this, we need to ensure the versions are consistent.

Determine the latest Guava version in both the folders as shown below:



Here, the hadoop's folder guava version is later than that of the hive, so replace with this one.
In this case, delete **guava-19.0.jar** in Hive lib folder, and then copy **guava-27.0-jre.jar** from Hadoop folder to Hive.