Table of Contents

- Jdk : jdk-8u40-linux-i586.tar.gz
- Hadoop: hadoop-3.1.2.tar.gz

Prelude

Create a common network.

46b42fc00055 spark-net bridge local

Using Docker:

Create a network – spark-net

Container – 0 – master Node

#docker run -it --name hadoop0 --network spark-net -v /Users/henrypotsangbam/Documents/Docker:/opt --privileged -p 8088:8088 -p 9870:9870 -p 9864:9864 -p 8032:8032 -p 8188:8188 -p 8020:8020 --hostname hadoop0 centos:7 /usr/sbin/init

Container – 1 – Slave Node

#docker run -it --name hadoop1 --network spark-net -v /Users/henrypotsangbam/Documents/Docker:/opt --privileged -p 8089:8088 -p 9871:9870 -p 9865:9864 -p 8033:8032 -p 8189:8188 -p 8022:8020 --hostname hadoop1 centos:7 /usr/sbin/init

using Docker Ends.

Launching a Cluster: HDFS – 120 Minutes

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

Configure and install HDFS.

Options:

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

OS should be any Linux of 6 and 7 version.

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:

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

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

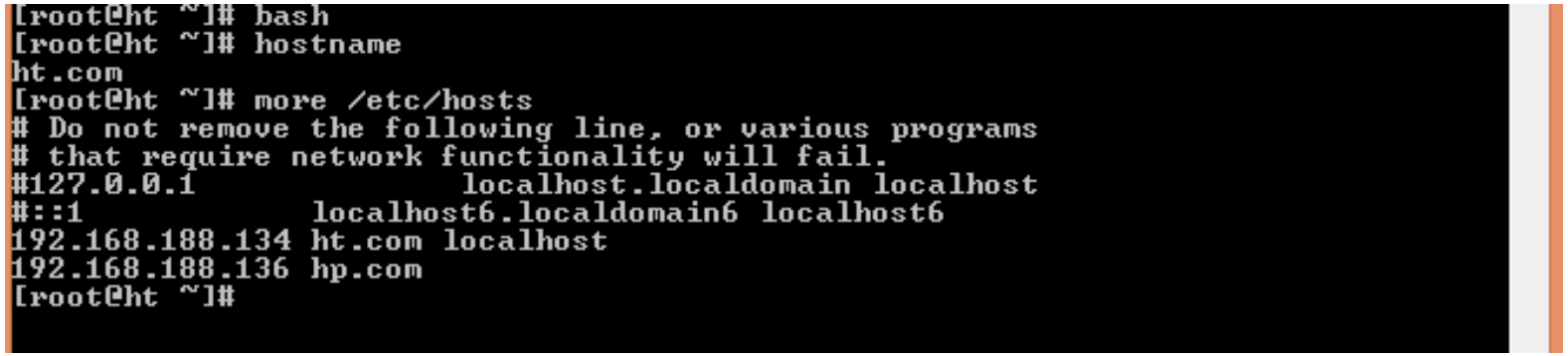
hostname	Container	Remarks
Hadoop0	Hadoop0	HDFS & YARN Services
Hadoop1	Hadoop1	HDFS & YARN Services.

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.



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.

Download Location:
https://hadoop.apache.org/releases.html

Extract hadoop & jdk file as follow:

#tar -xvf hadoop-X.tar.gz -C /opt
#tar -xvf jdk-8* -C /opt

Rename the folder:

#mv hadoo* hadoop
#mv jdk* jdk

Install 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

cd /opt/hadoop

[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.

Use the following, all files will be inside the HADOOP_HOME: Ensure that you replace with the hostname of your machine.

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>
```

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 check that you can ssh to the localhost without a passphrase:

```
#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 and It has to be executed only once.

$ hdfs namenode -format

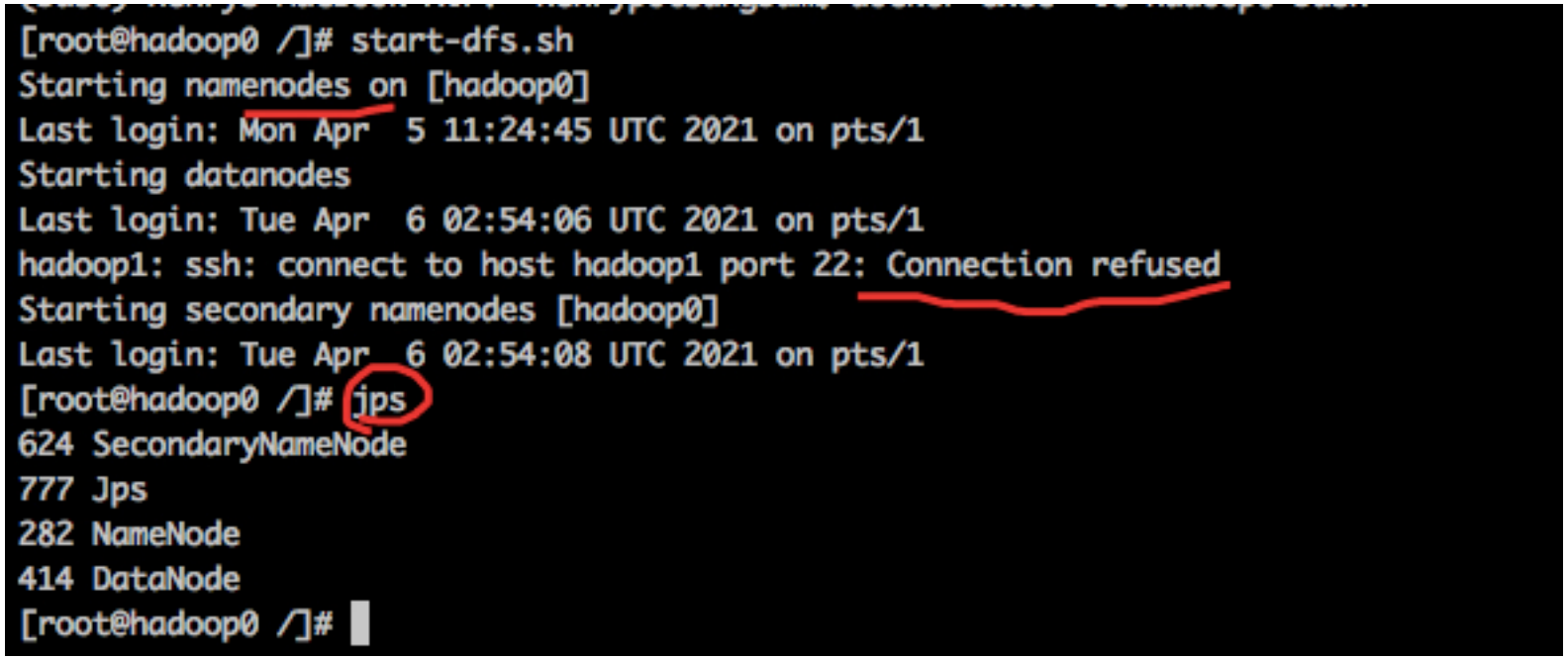
```
2021-04-05 10:16:51,093 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2021-04-05 10:16:51,093 INFO util.GSet: VM type       = 64-bit
2021-04-05 10:16:51,095 INFO util.GSet: 0.029999999329447746% max memory 876.5 MB = 269.3 KB
2021-04-05 10:16:51,095 INFO util.GSet: capacity      = 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 , replace with your hostname

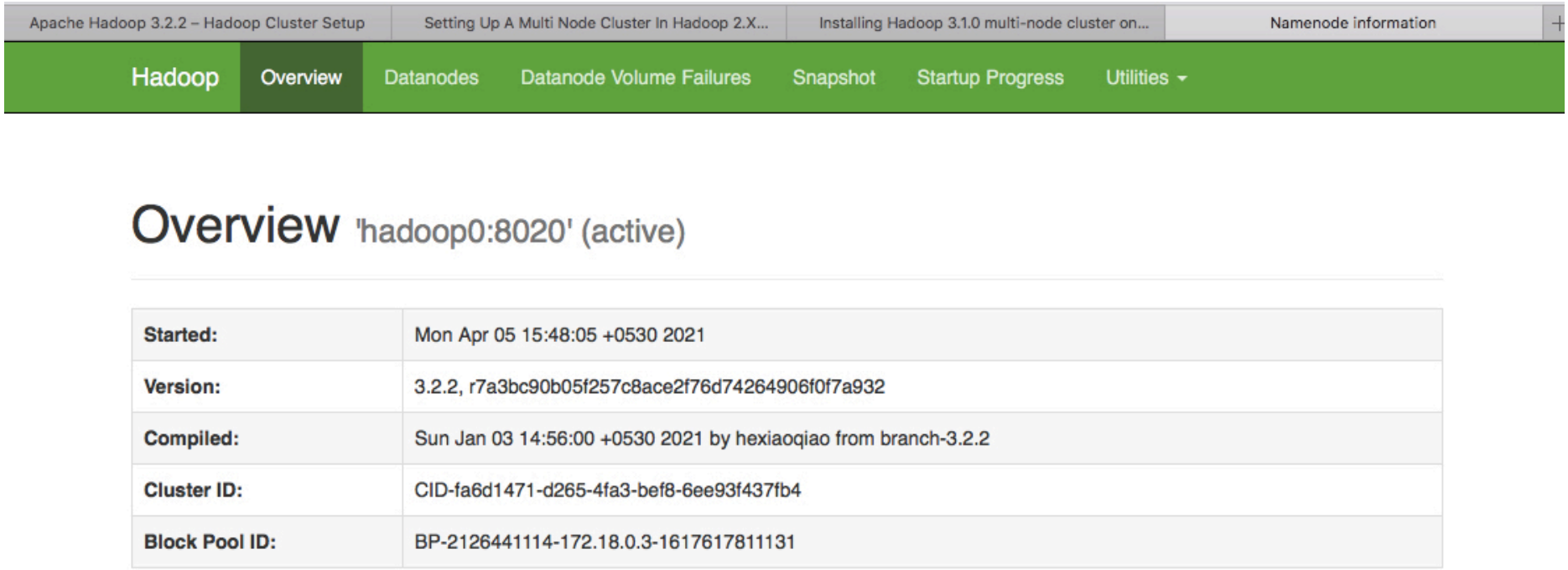
hadoopo

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

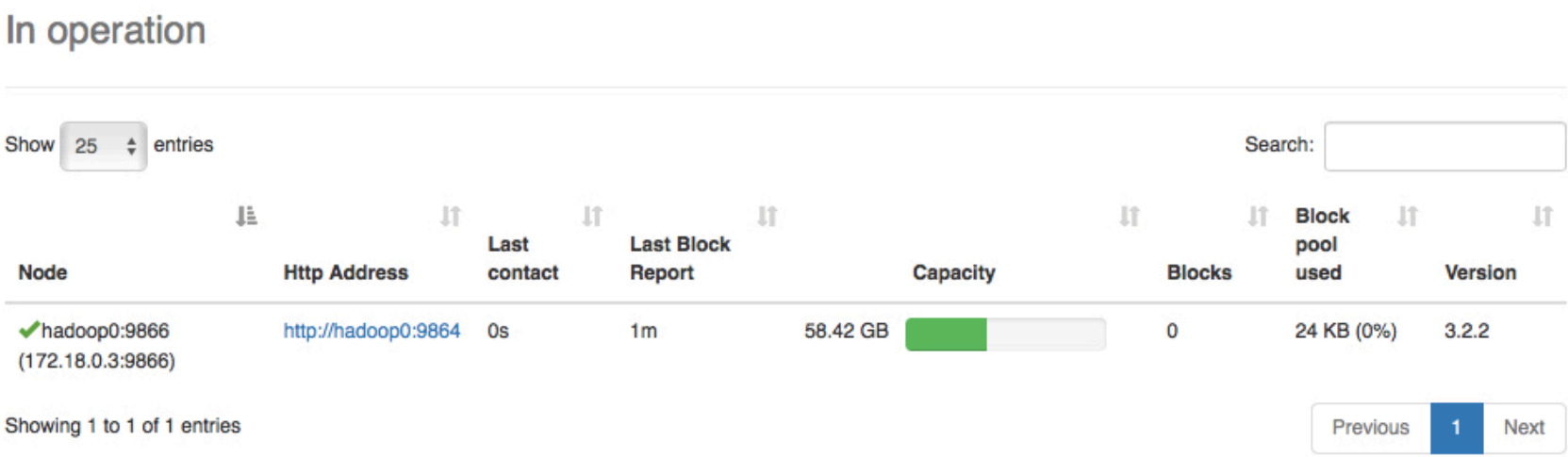
$ start-dfs.sh



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.

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-X.tar.gz -C /opt
#tar -xvf jdk-8u40-linux-i586.tar.gz -C /opt

Rename the folder:

#mv hadoo* hadoop
#mv jdk* jdk

Install 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

Unpack the downloaded Hadoop distribution. 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
cd /opt/hadoop

[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.

Use the following, all files will be inside the HADOOP_HOME: (Enter the hostname of your master - NameNode)

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>

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>

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

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

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

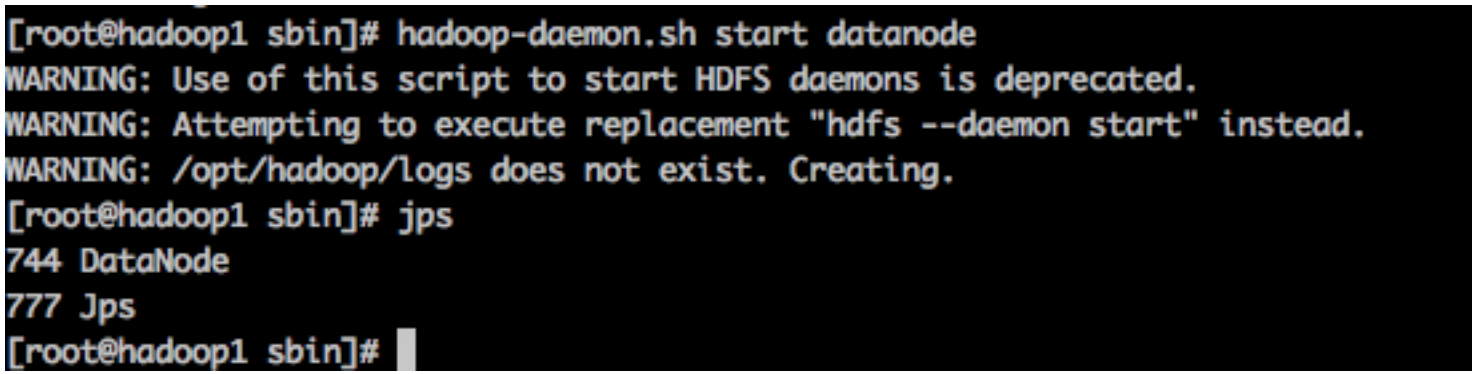
```
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
```

export YARN_NODEMANAGER_USER="root"

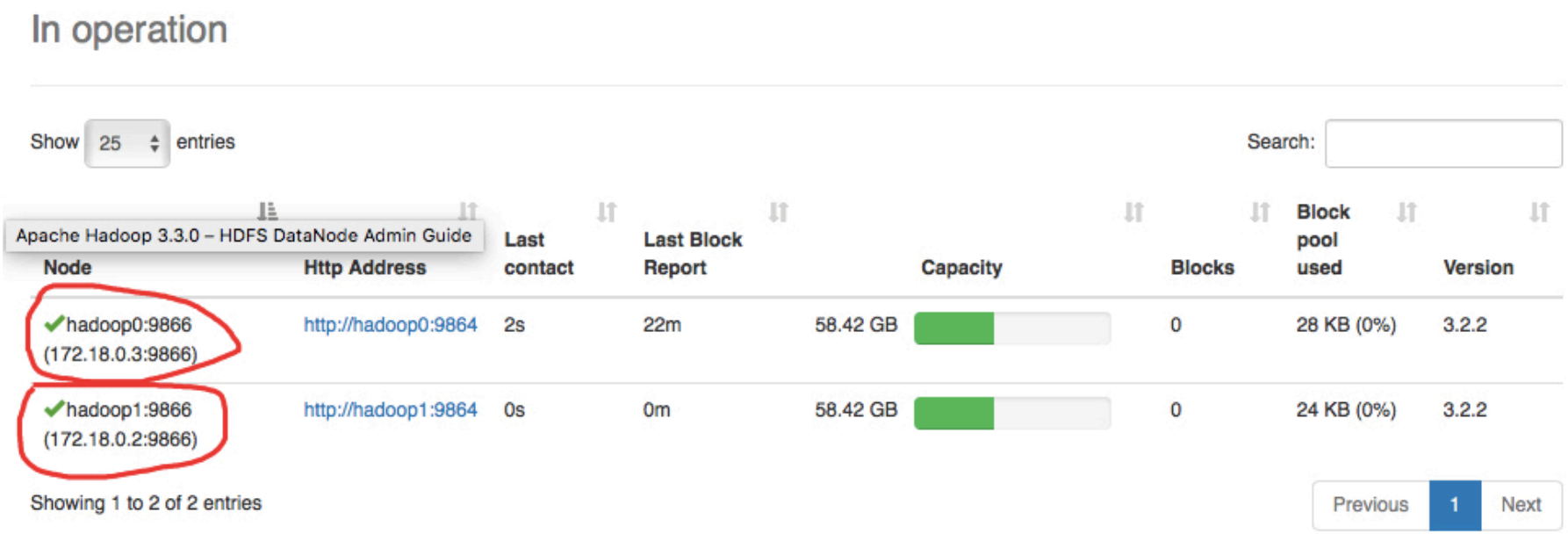
Slave nodes installation ends here.

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

#hadoop-daemon.sh start datanode
or
#hdfs --daemon start datanode
#jps



Now refresh the HDFS console.



Make the HDFS directories required to execute MapReduce jobs:

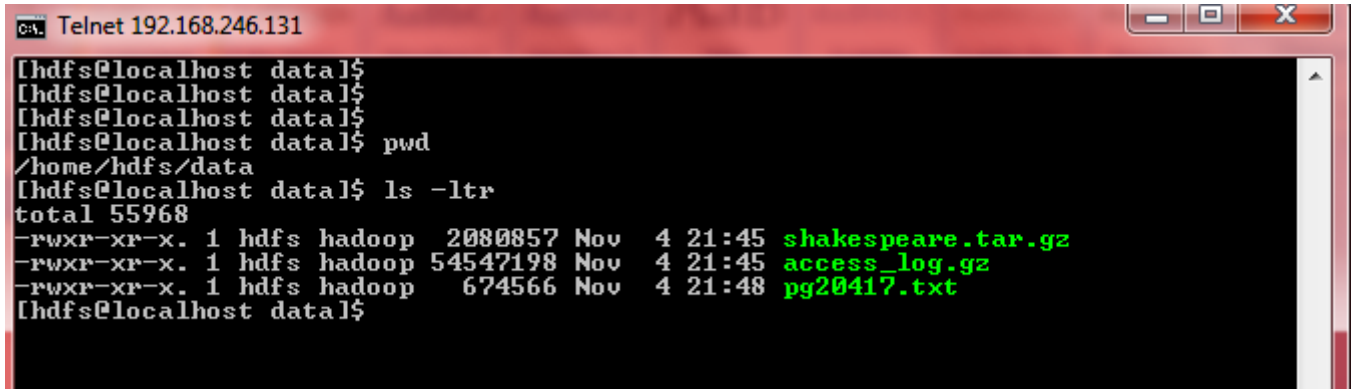
```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/root
```

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

Exploring HDFS – 60 Minutes

All Data files (local) need to copy in your VM. All exercise need 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.

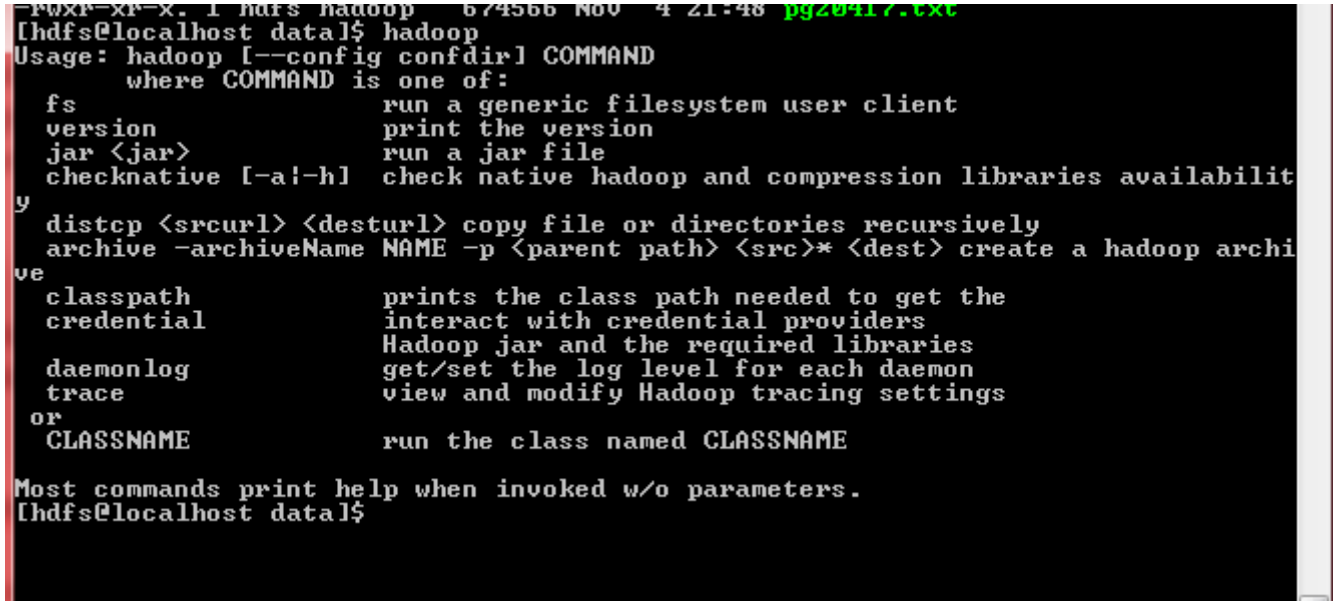
/software/data/shakespeare.tar.gz
/software/data/access_log.gz
/software/data/pg20417.txt



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

$ hadoop

The hadoop 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.



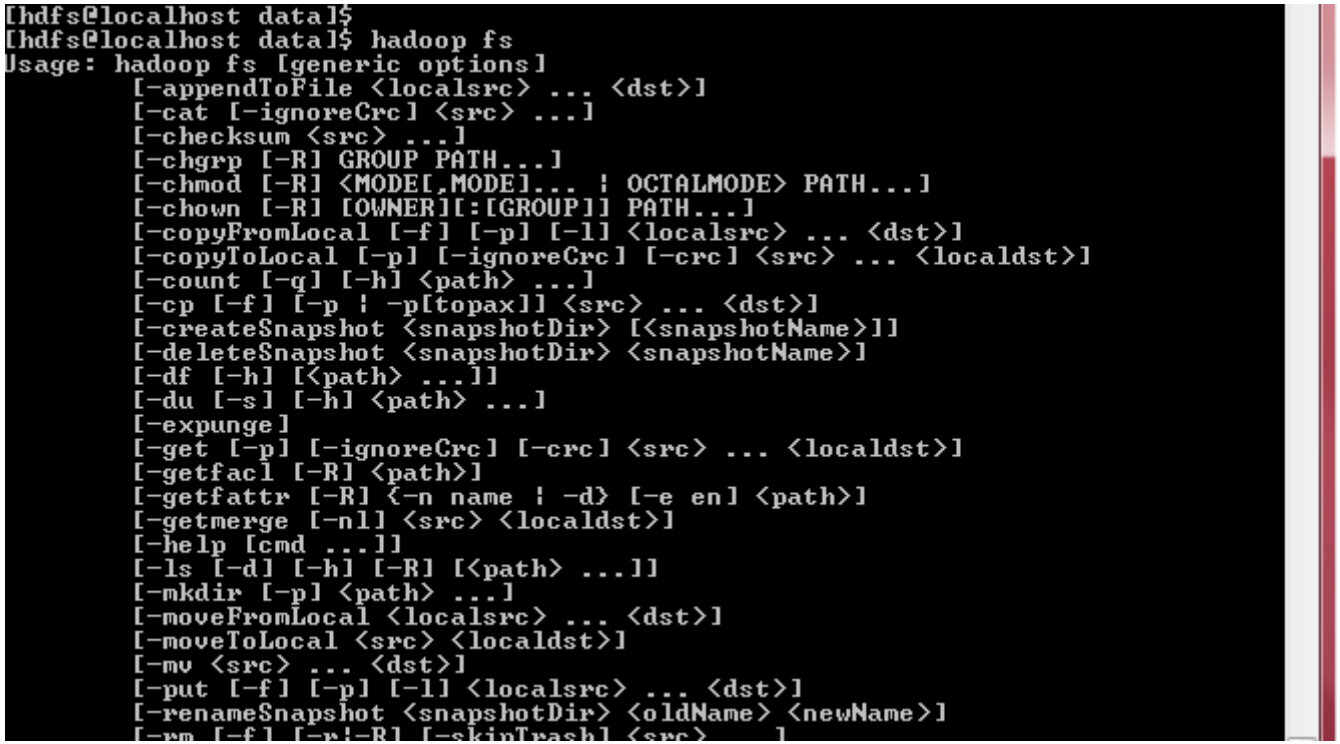
Exploring HDFS

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:

$ hadoop fs
or
hdfs

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



$ hadoop fs -ls /

or

#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.

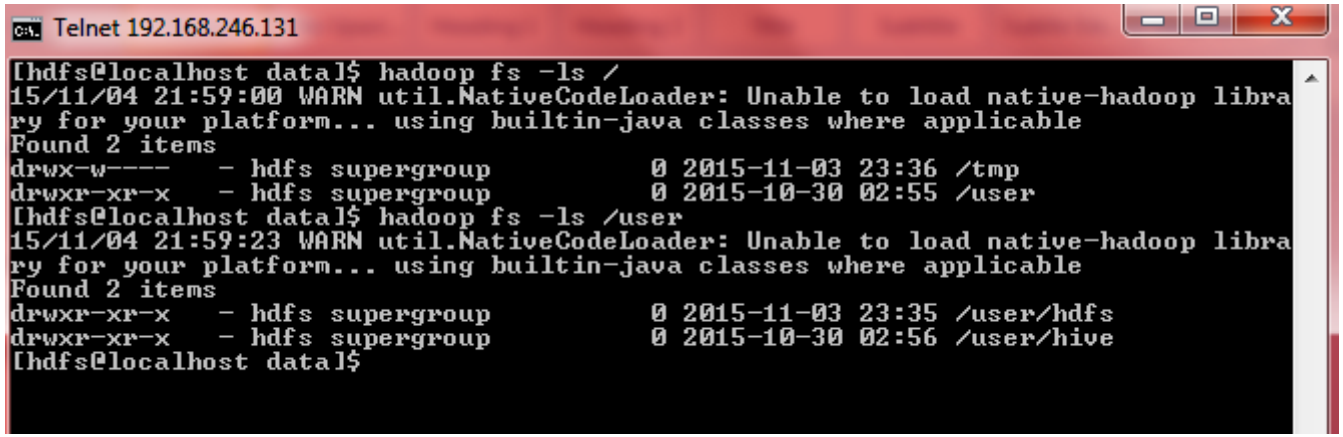
Try viewing the contents of the /user directory by running:

$ hadoop fs -ls /user

or

hdfs dfs -ls /user

You will see your home directory in the directory listing.



List the contents of your home directory by running:

$ hadoop fs -ls /user/hdfs

Try all hadoop command by replacing with hdfs dfs

This is different from running hadoop fs -ls /foo, which refers to a directory that doesn't exist. In this case, an error message would be displayed.
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.

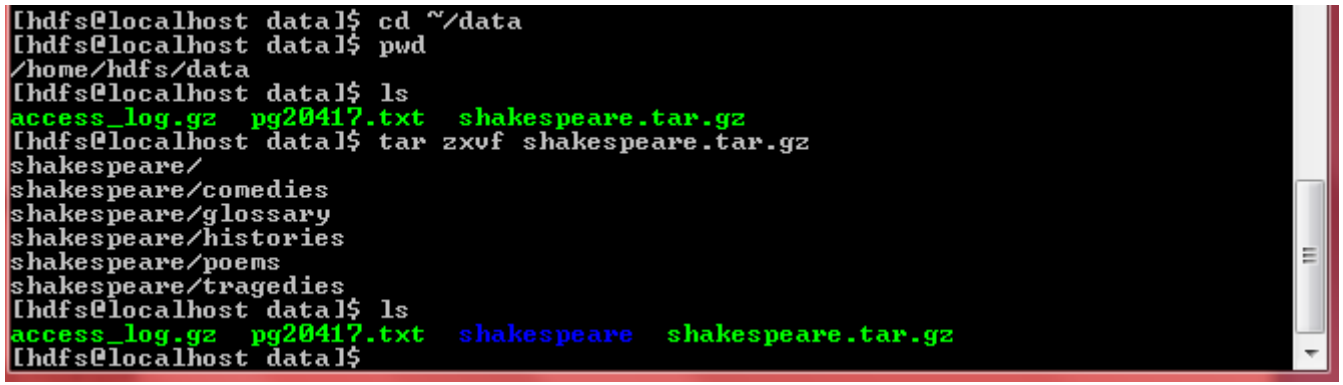
$ cd /Software

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.gz by running with root credentials(su root):

$ tar zxvf shakespeare.tar.gz

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



copy this directory into HDFS using hdfs:

$ hadoop fs -put shakespeare /user/hdfs/shakespeare

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

List the contents of your HDFS home directory now:

$ hadoop fs -ls /user/hdfs

You should see an entry for the shakespeare directory.

```
[hdfs@localhost data]$
[hdfs@localhost data]$ hadoop fs -put shakespeare /user/hdfs/shakespeare
15/11/04 22:03:51 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
[hdfs@localhost data]$ hadoop fs -ls /user/hdfs
15/11/04 22:04:15 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 11 items
-rw-r--r--   1 hdfs supergroup   40990992 2015-11-03 01:10 /user/hdfs/NYSE_daily
_prices_A.csv
-rw-r--r--   1 hdfs supergroup     224232 2015-11-03 01:11 /user/hdfs/NYSE_divid
ends_A.csv
drwxr-xr-x   - hdfs supergroup          0 2015-11-03 23:07 /user/hdfs/ad_data1
drwxr-xr-x   - hdfs supergroup          0 2015-11-03 23:36 /user/hdfs/ad_data2
-rw-r--r--   1 hdfs supergroup     208348 2015-10-30 02:32 /user/hdfs/excite-sma
ll.log
-rw-r--r--   1 hdfs supergroup        348 2015-11-02 20:46 /user/hdfs/movies.txt
drwxr-xr-x   - hdfs supergroup          0 2015-11-02 23:24 /user/hdfs/movies_gre
ater_than_three_point_five
drwxr-xr-x   - hdfs supergroup          0 2015-11-03 01:36 /user/hdfs/output
-rw-r--r--   1 hdfs supergroup       1730 2015-11-03 22:00 /user/hdfs/sample1.tx
t
-rw-r--r--   1 hdfs supergroup       1568 2015-11-03 23:31 /user/hdfs/sample2.tx
t
drwxr-xr-x   - hdfs supergroup          0 2015-11-04 22:03 /user/hdfs/shakespear
e
[hdfs@localhost data]$
```

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

$ hadoop fs -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/hdfs.

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 in future labs. This file is currently compressed using GZip. Rather than extract the file to the local disk and then upload it, we will extract and upload in one step.
First, create a directory in HDFS in which to store it:

$ hadoop fs -mkdir weblog

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

$ hadoop fs

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 hadoop fs command. For example to list the / and /app directories you need to input the following commands:

```
hadoop fs -ls /
hadoop fs -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:

```
hadoop fs -mkdir test
```

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

```
hadoop fs -ls /
hadoop fs -ls /user/hdfs
```

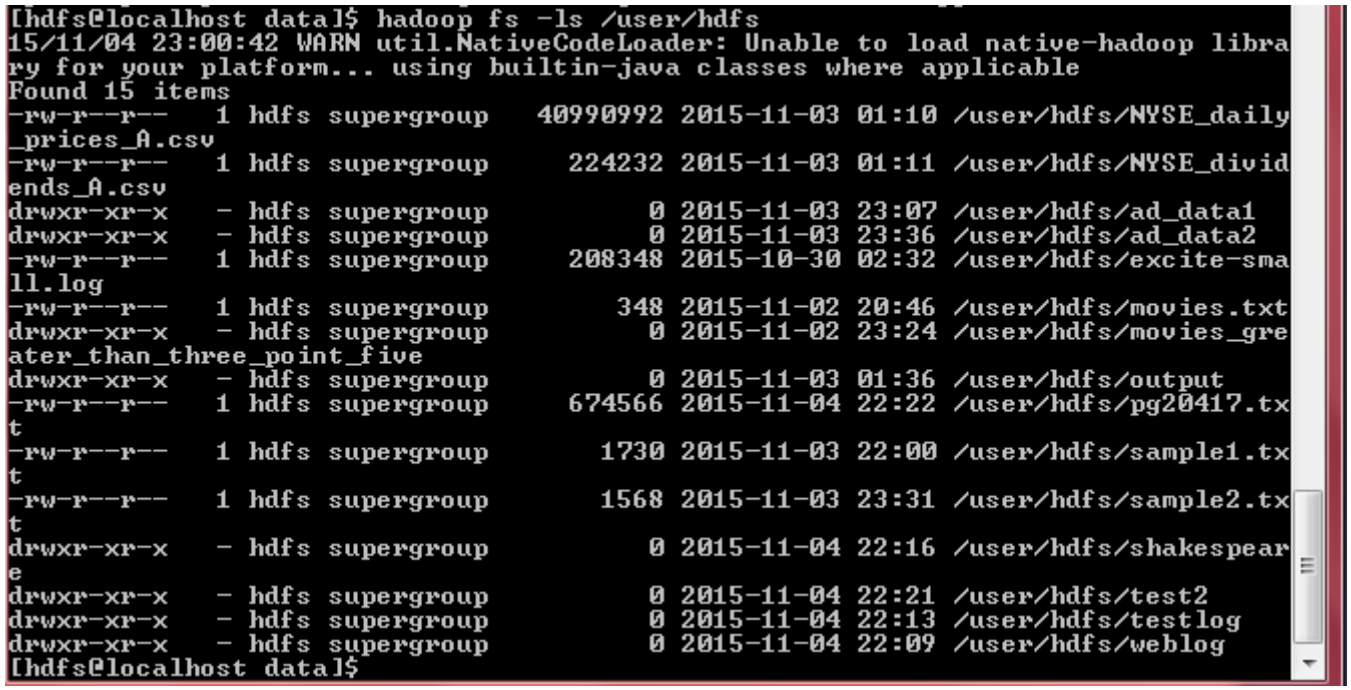
You should be aware that you can pipe (using the | character) any HDFS command to be used with the Linux shell. For example, you can easily use *grep* with HDFS by doing the following:

```
hadoop fs -mkdir /user/hdfs/test2
hadoop fs -ls /user/hdfs | 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

hadoop fs -put /home/hdfs/data/pg20417.txt pg20417.txt
hadoop fs -ls /user/hdfs

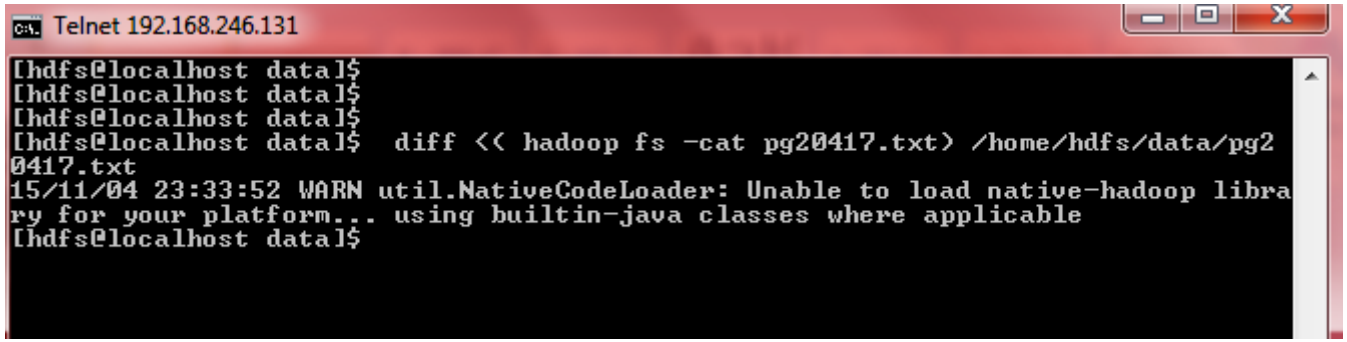


You should now see a new file called /user/hdfs/pg* listed. In order to view the contents of this file we will use the *-cat* command as follows:

hadoop fs -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:

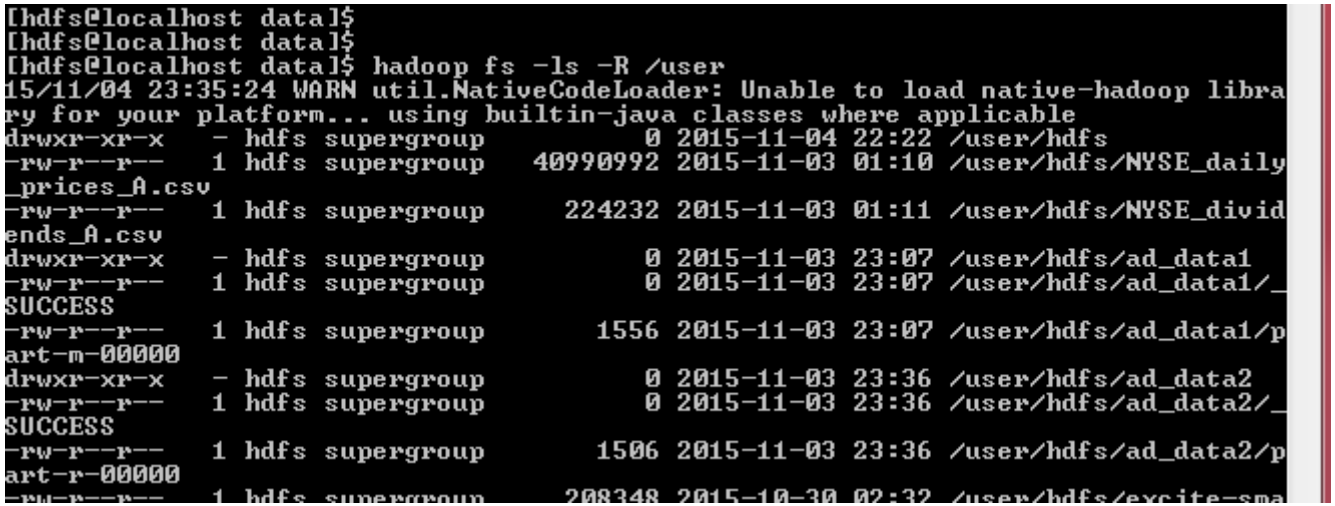
diff <(hadoop fs -cat pg20417.txt) /home/hdfs/data/pg20417.txt
Since the diff command produces no output we know that the files are the same (the diff command prints all the lines in the files that differ).



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:

hadoop fs -ls /user
hadoop fs -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:

hadoop fs -du pg20417.txt

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

hadoop fs -du /user/hdfs

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

hadoop fs -dus /user/hdfs

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

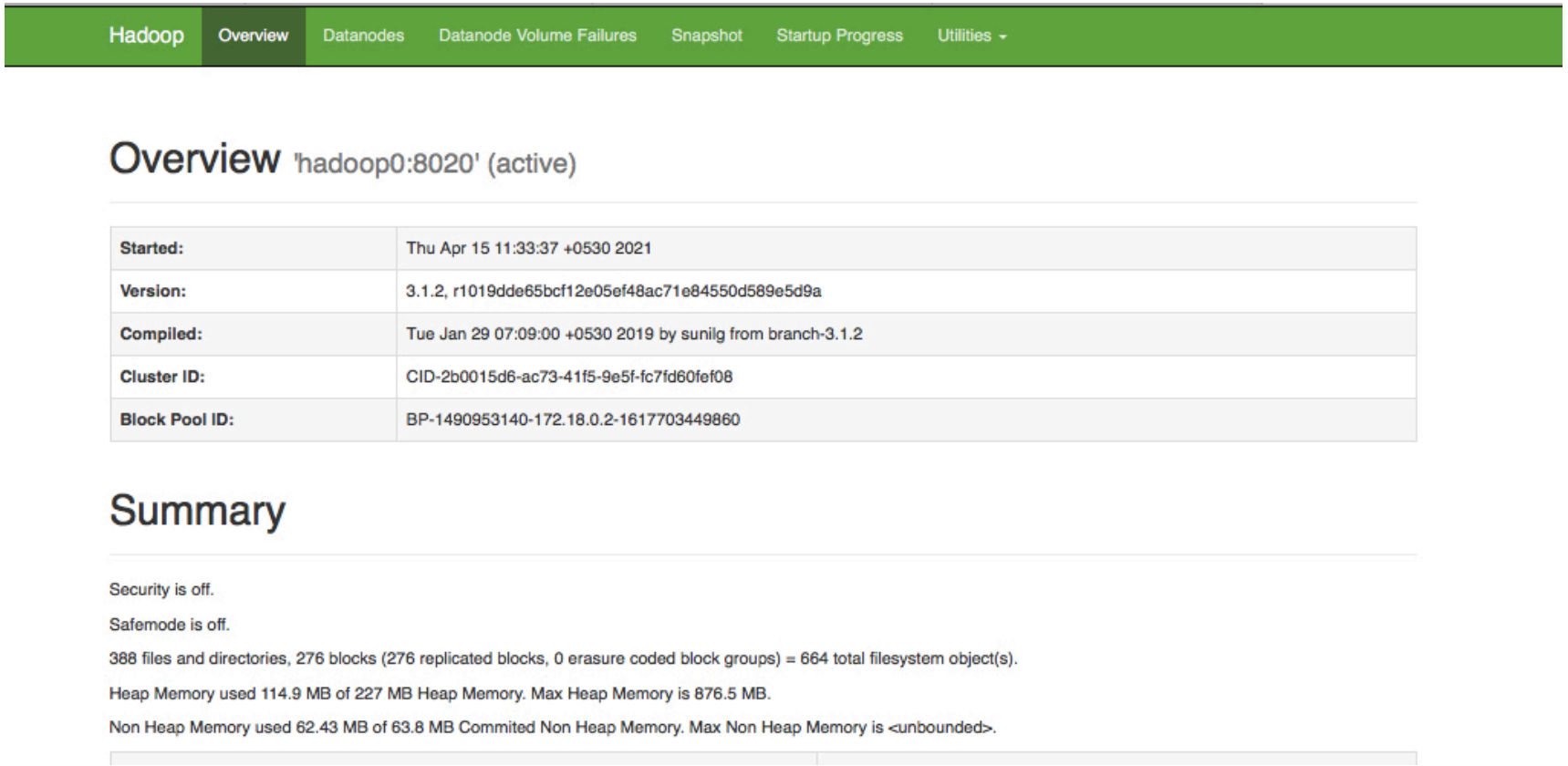
hadoop fs -help

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

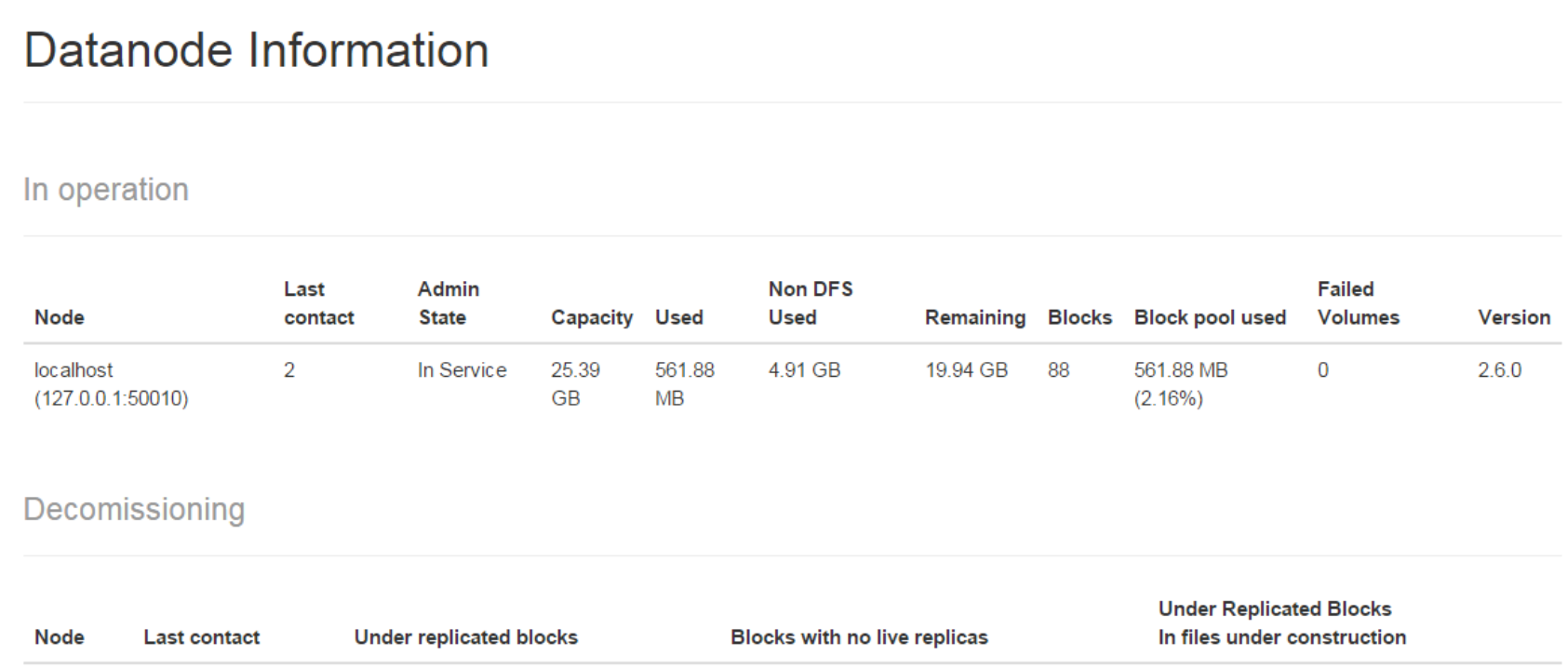
hadoop fs -help dus

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

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



Click on Datanodes



Datanode Information

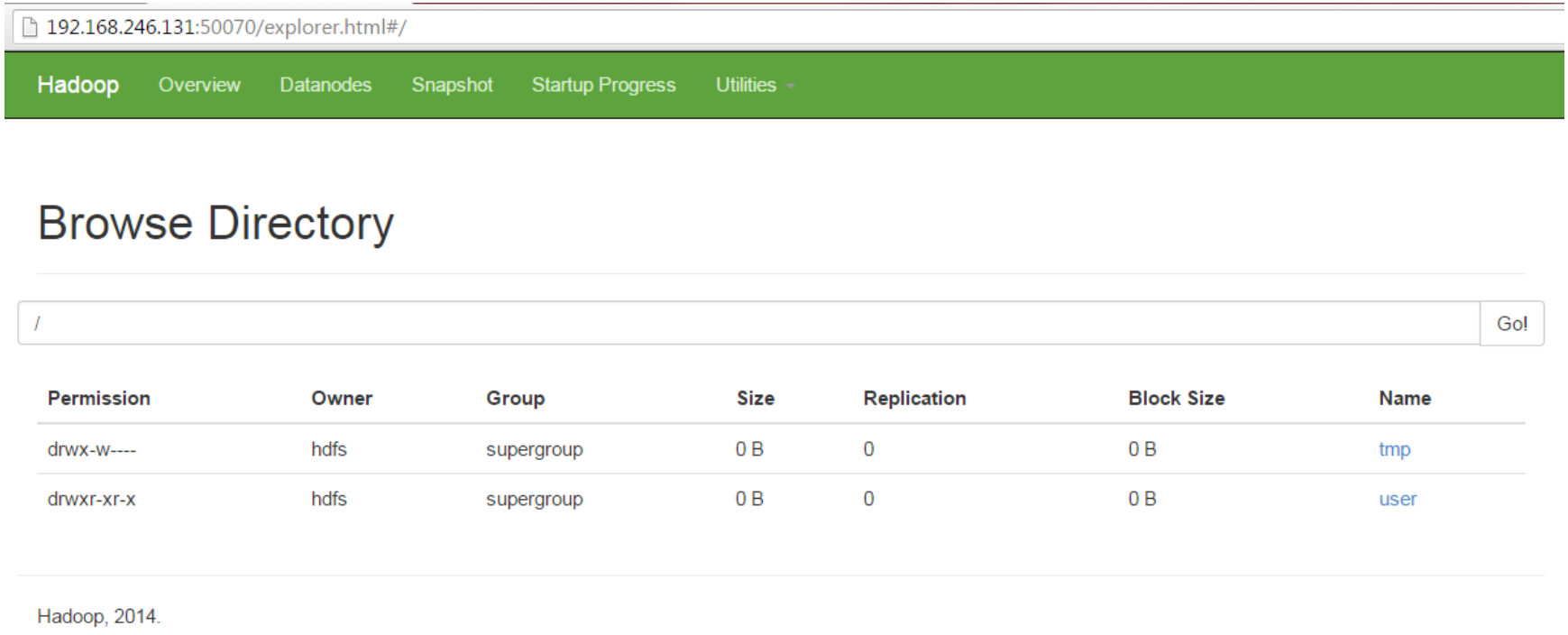
In operation

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

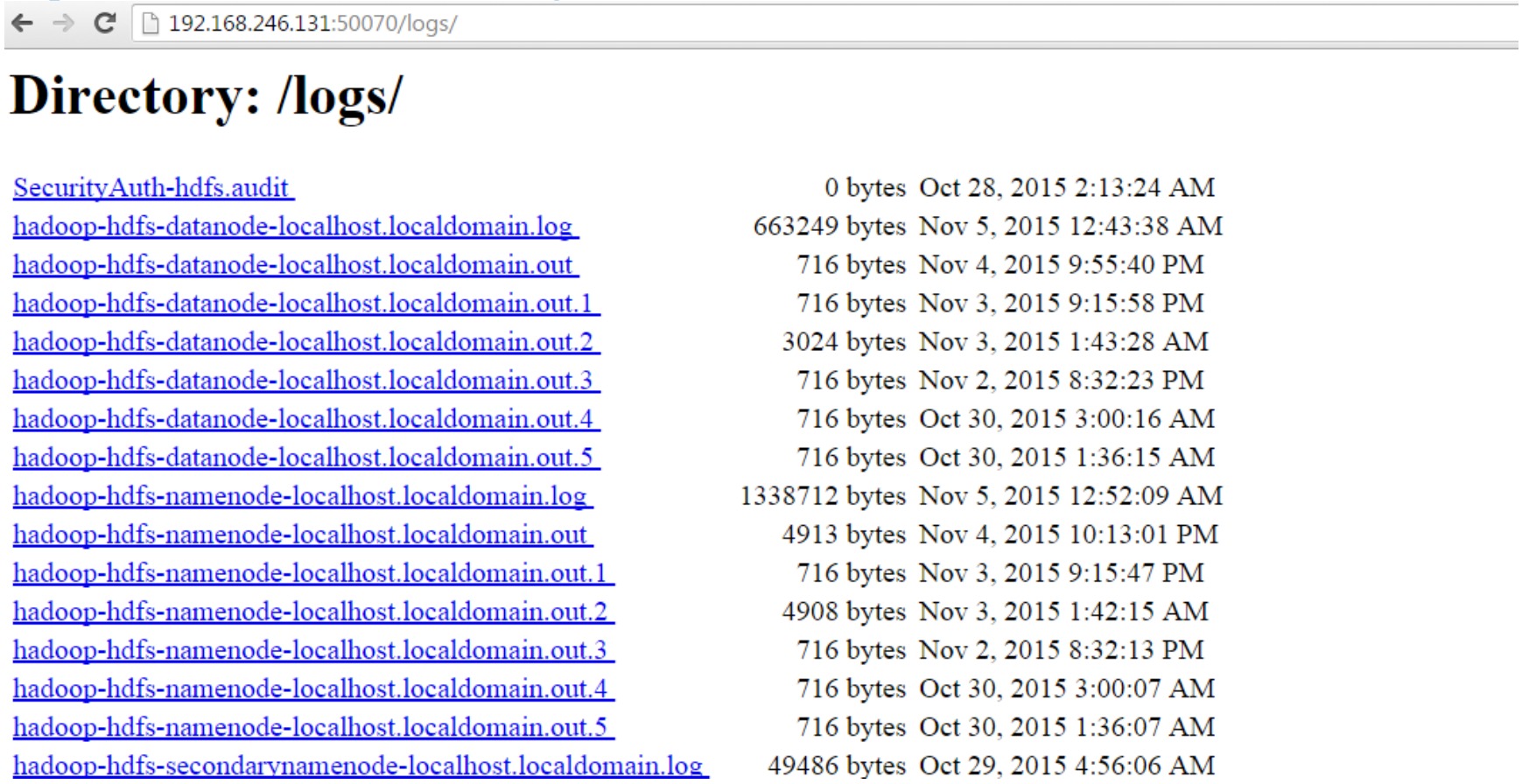
Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction

Click on Startup Progress and utlities



Click on Logs

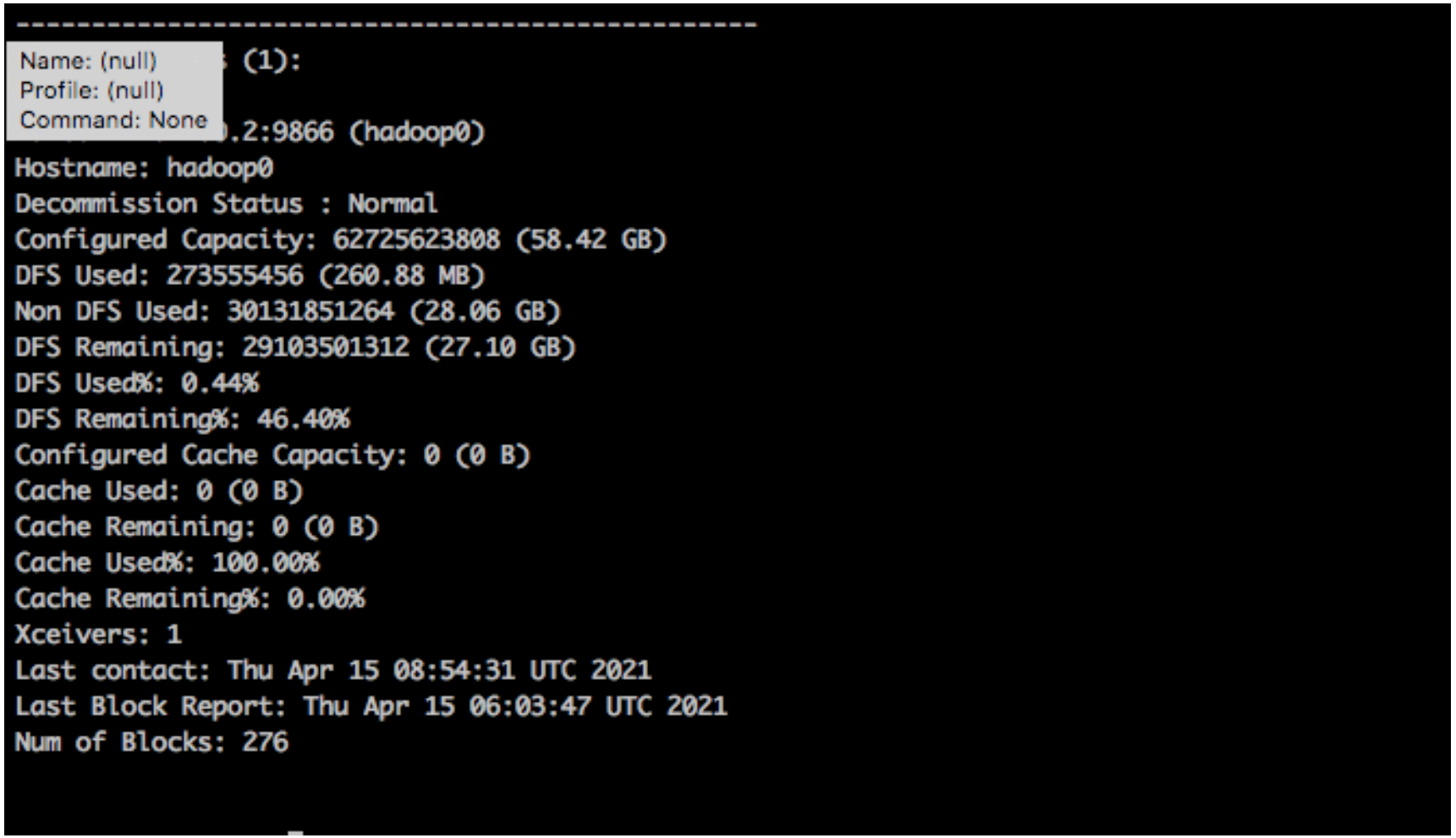
http://192.168.246.131:50070/logs/



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

#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.

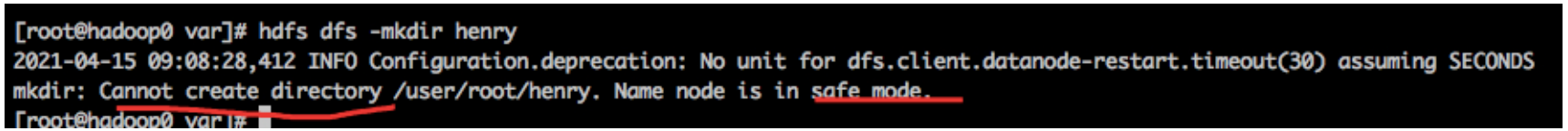
#hdfs dfsadmin -safemode get

#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
```

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

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
[root@hadoop0 var]#
```

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

Launching a Cluster: YARN – 60 Minutes

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. All files should be in /opt/hadoop

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.

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>
</configuration>
```

On Second Node or all slaves/workers node.

etc/hadoop/yarn-site.xml:

```
<configuration>
<property>
      <name>yarn.resourcemanager.hostname</name>
      <value>hadoop0</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>
</configuration>
```

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

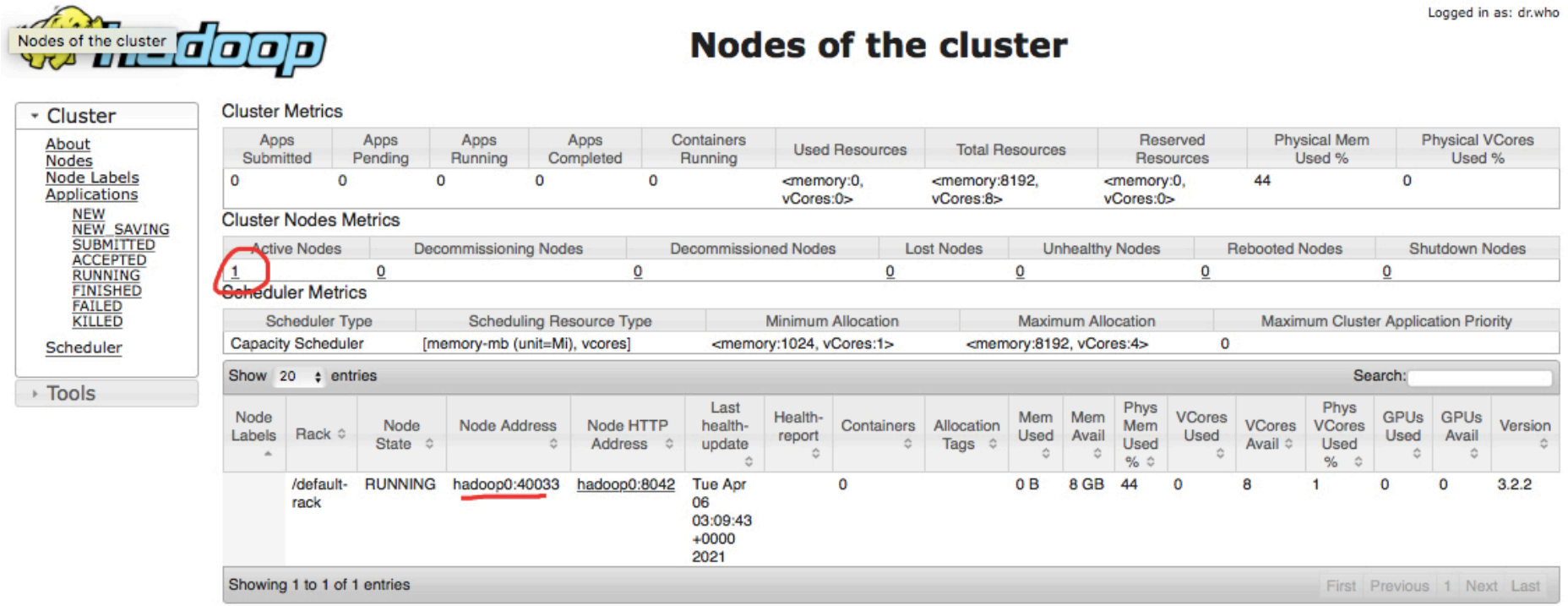
```
$ start-yarn.sh
```

```
[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]#
```

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.

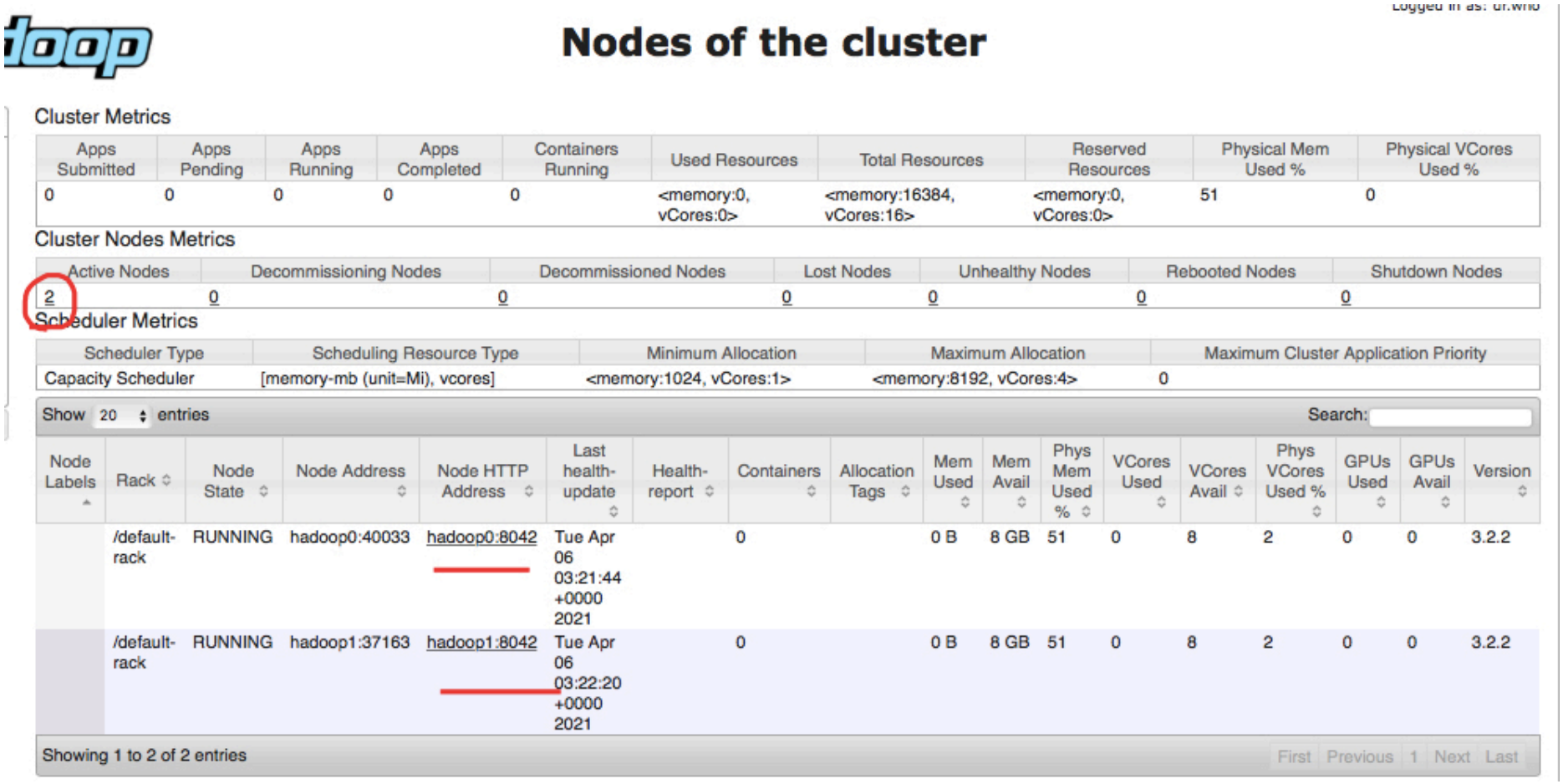


Now Start the Nodemanager on the Second Node or slaves node.
#yarn --daemon start nodemanager

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.



Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %
0	0	0	0	0	<memory:0, vCores:0>	<memory:16384, vCores:16>	<memory:0, vCores:0>	51	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
2	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Phys Mem Used %	VCores Used	VCores Avail	Phys VCores Used %	GPUs Used	GPUs Avail	Version
	/default-rack	RUNNING	hadoop0:40033	hadoop0:8042	Tue Apr 06 03:21:44 +0000 2021		0		0 B	8 GB	51	0	8	2	0	0	3.2.2
	/default-rack	RUNNING	hadoop1:37163	hadoop1:8042	Tue Apr 06 03:22:20 +0000 2021		0		0 B	8 GB	51	0	8	2	0	0	3.2.2

Showing 1 to 2 of 2 entries First Previous 1 Next Last

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

```
$ sbin/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.
---------- Lab Ends Here ----------

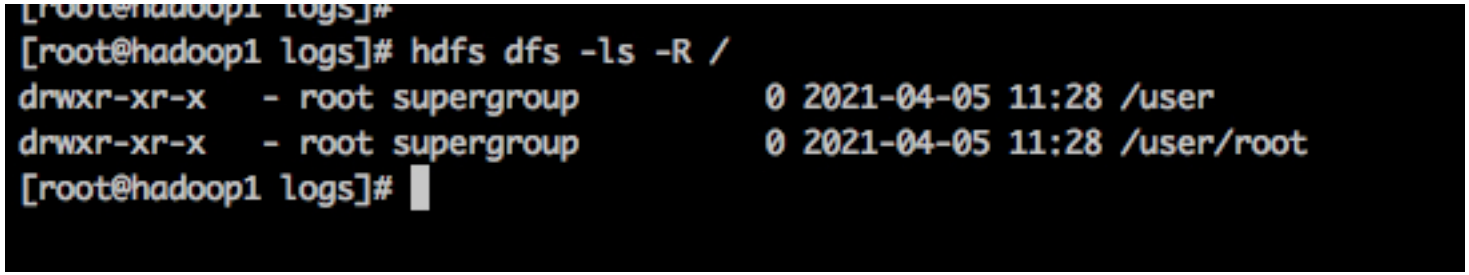
YARN Job on Hadoop Cluster – 60 Minutes

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

Make the HDFS directories required to execute MapReduce jobs:

Verify the folders.

#hdfs dfs -ls -R /



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

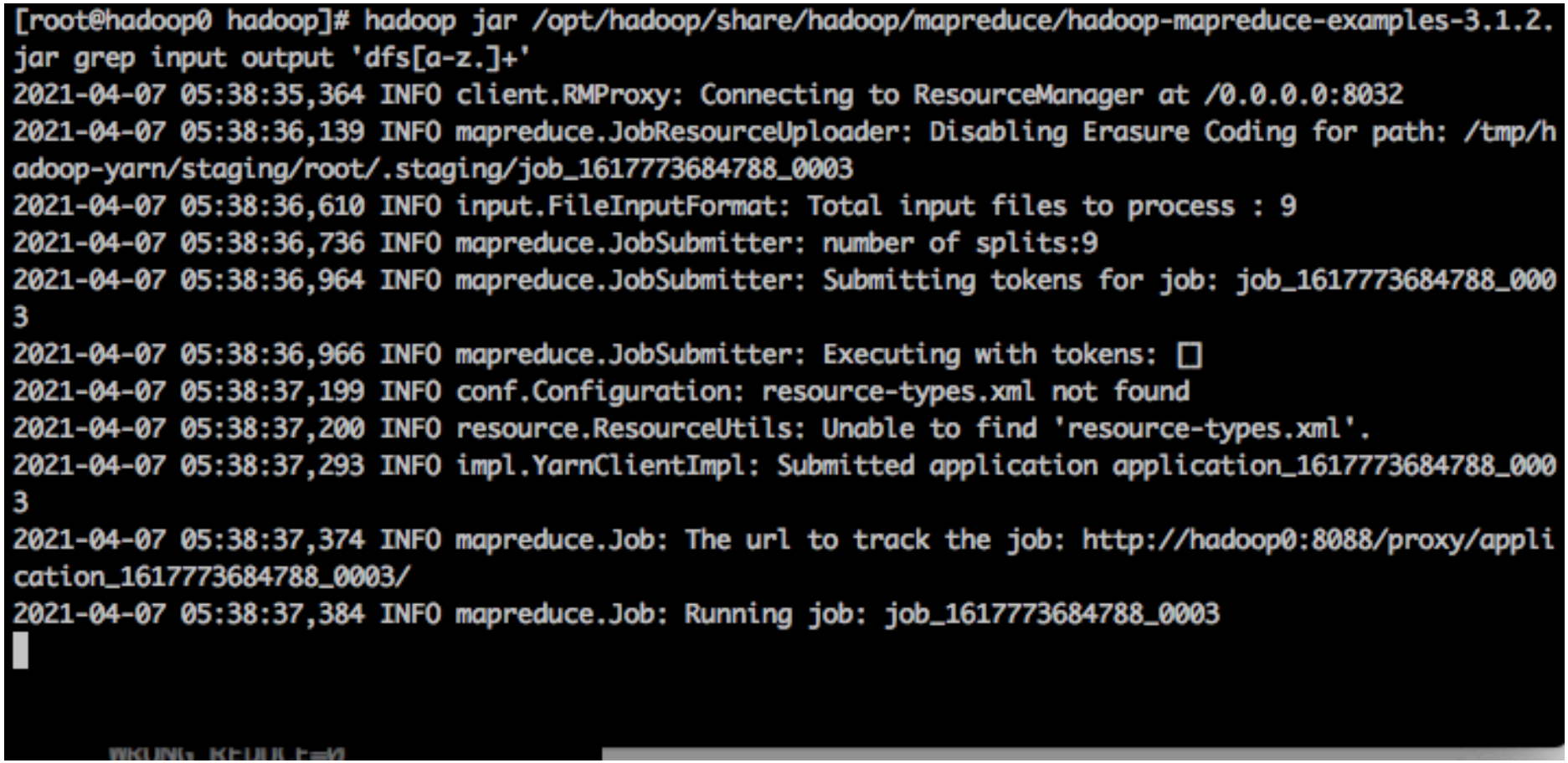
Verify the files

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]#
```

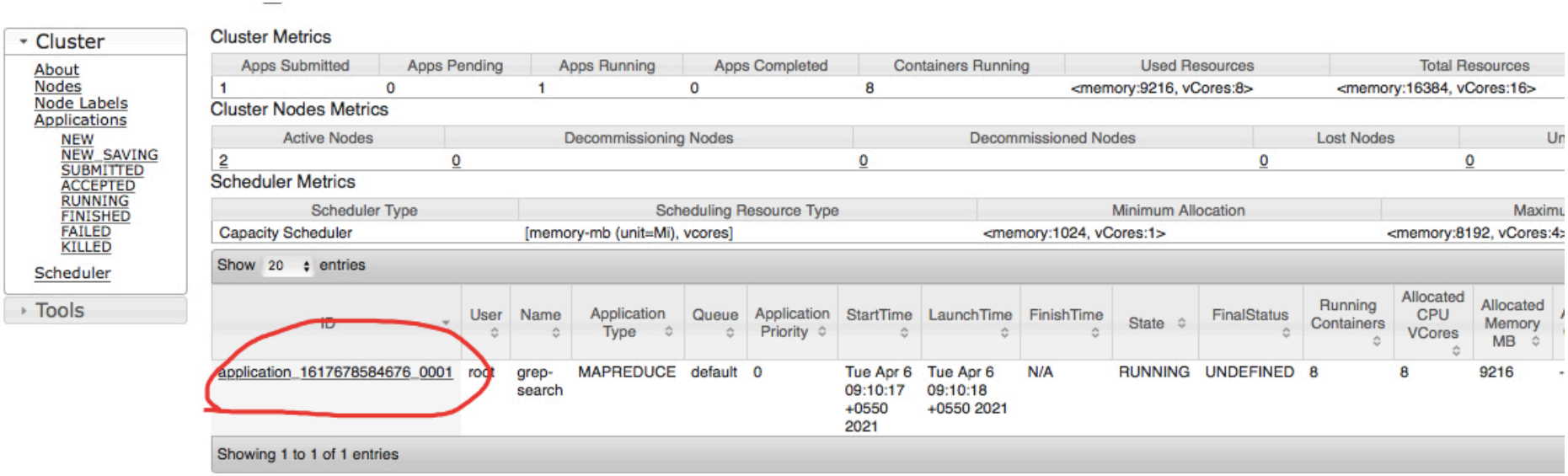
Run some of the examples provided:

#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/hadoop-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_0003
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_0003
2021-04-07 05:38:37,374 INFO mapreduce.Job: The url to track the job: http://hadoop0:8088/proxy/application_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



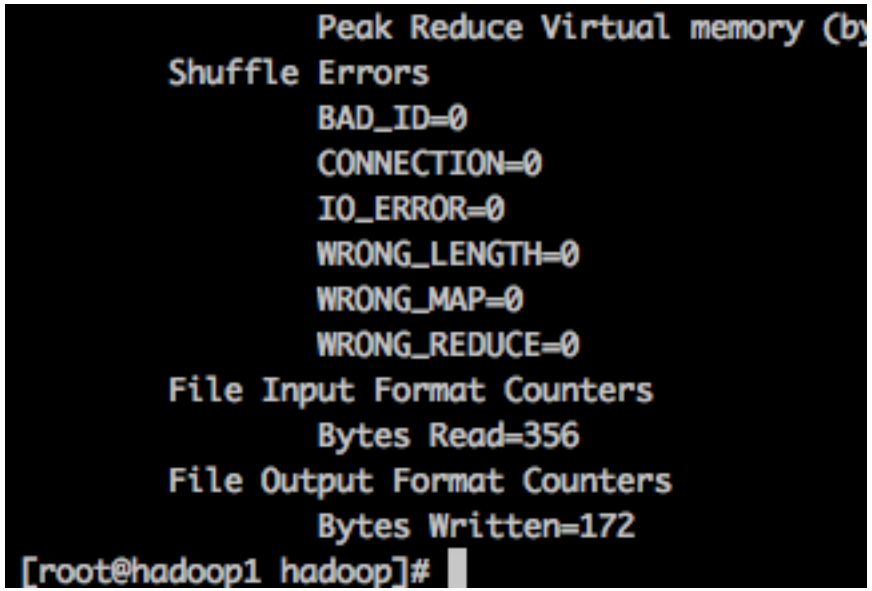
At the end it should have the success state as shown below.

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	R 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.

Docker : Enter into the running namenode container by executing this command: (

docker exec -it hadoop0 bash

)

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 input

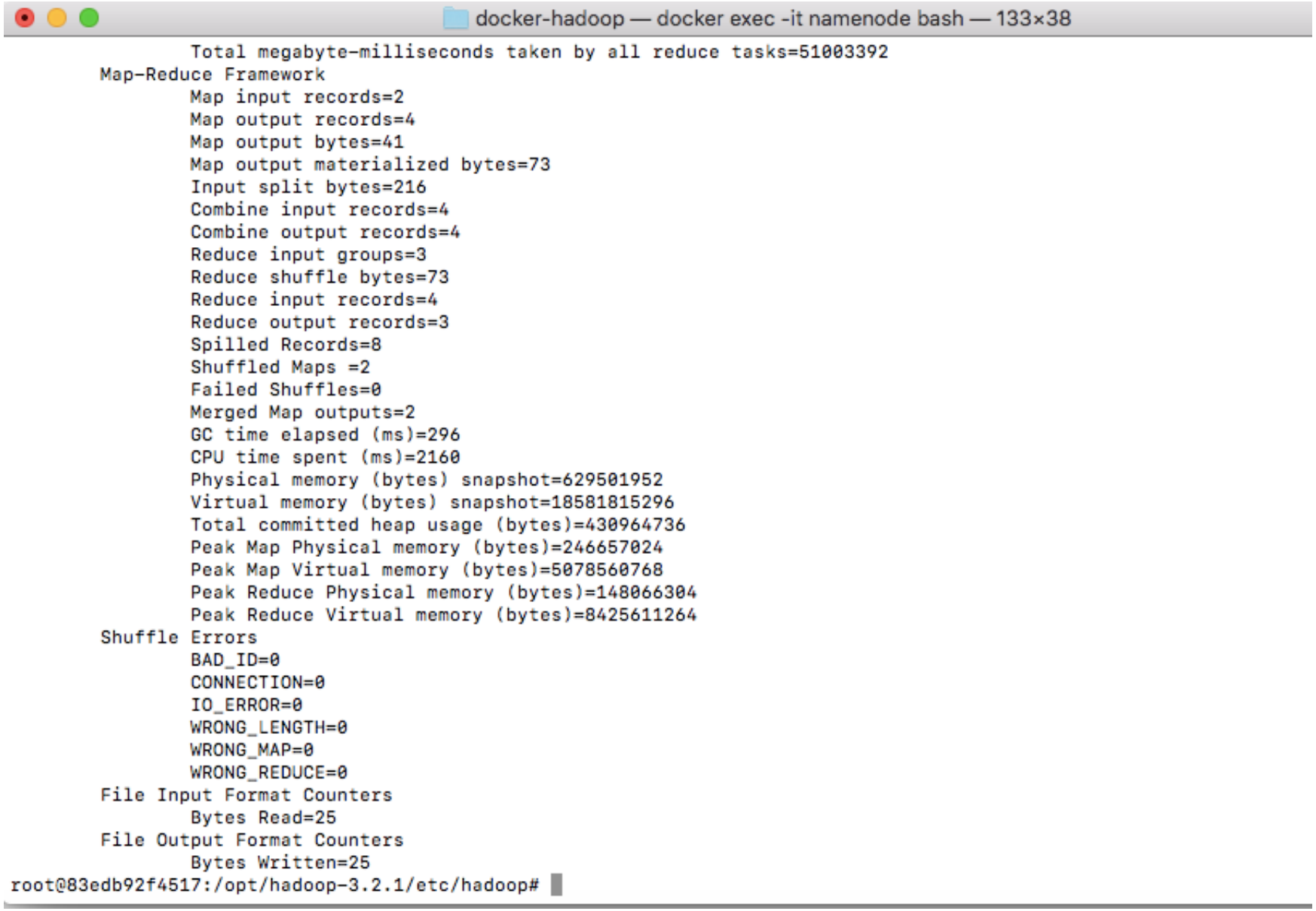
To put the input files to all the datanodes on HDFS, use this command:

$ hdfs dfs -put ./input/* input

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 input output

```
[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 output/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:

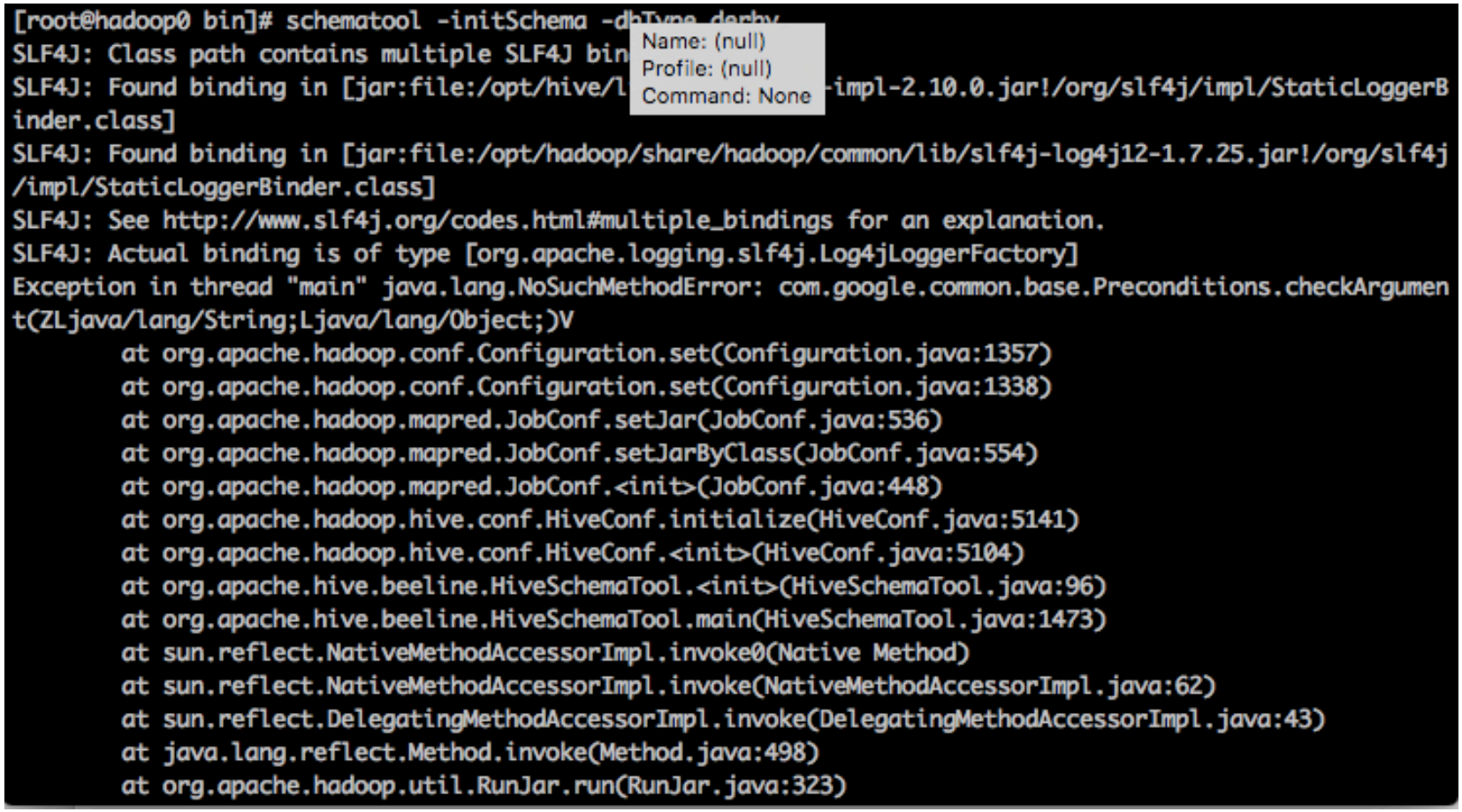
hive -hiveconf tez.am.resource.memory.mb=4096
Another setting to consider tweaking is yarn.app.mapreduce.am.resource.mb

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

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;)

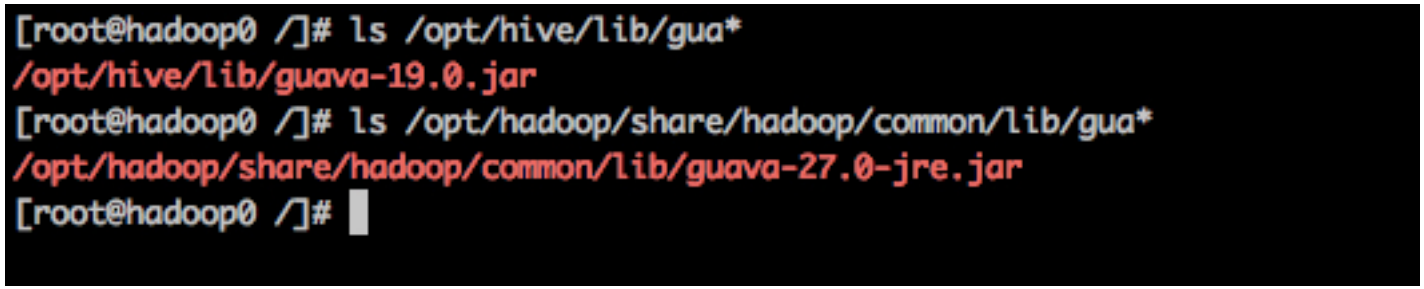


Solution:

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.