

## Table of Contents

Prelude .....	2
<b>Launching a Cluster: HDFS – 120 Minutes</b>	4
Exploring HDFS – 60 Minutes.....	22
<b>Launching a Cluster: YARN – 60 Minutes</b> .....	55
<b>YARN Job on Hadoop Cluster – 60 Minutes</b> .....	64
<b>Capacity Scheduler – 90 minutes</b> .....	77
<b>HDFS High Availability Using the Quorum Journal Manager – 3 Hrs</b> .....	86
<b>ResourceManager high availability – 2 Hrs</b> .....	101
<b>Archival Disk Storage – 60 Minutes</b> .....	113
<b>Hive Installation – 90 Minutes</b> .....	124
<b>Erarata .....</b>	138
Hadoop Version Mismatch with Hive Libraries.....	138

- Jdk : jdk-8u40-linux-i586.tar.gz
- 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
  - <https://downloads.apache.org/db/derby/db-derby-10.14.2.0/db-derby-10.14.2.0-bin.tar.gz>
- <https://mirrors.estointernet.in/apache/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.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 hadoopo --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 hadoopo 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 nomenclature to avoid confusion. Its very important.

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

<b>hostname</b>	<b>Container</b>	<b>Remarks</b>
<b>Hadoopo</b>	Hadoopo	HDFS & YARN Services
<b>Hadoop1</b>	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.

```
[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 ~]#
```

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 the hadoop file as follows:

```
#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
```

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:

etc/hadoop/core-site.xml:

```
<configuration>  
  <property>  
    <name>fs.defaultFS</name>  
    <value>hdfs://hadoop: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_00000000000000000000000000000000 using no compression
2021-04-05 10:16:51,366 INFO namenode.FSImageFormatProtobuf: Image file /opt/hdfs/namenode/current/fsimage.ckpt_00000000000000000000000000000000 of 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

hadoop

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

\$ start-dfs.sh

```
[root@hadoop0 ~]# start-dfs.sh
Starting namenodes on [hadoop0]
Last login: Mon Apr  5 11:24:45 UTC 2021 on pts/1
Starting datanodes
Last login: Tue Apr  6 02:54:06 UTC 2021 on pts/1
hadoop1: ssh: connect to host hadoop1 port 22: Connection refused
Starting secondary namenodes [hadoop0]
Last login: Tue Apr  6 02:54:08 UTC 2021 on pts/1
[root@hadoop0 ~]# jps
624 SecondaryNameNode
777 Jps
282 NameNode
414 DataNode
[root@hadoop0 ~]#
```

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

The screenshot shows a top navigation bar with several tabs: 'Apache Hadoop 3.2.2 – Hadoop Cluster Setup', 'Setting Up A Multi Node Cluster In Hadoop 2.X...', 'Installing Hadoop 3.1.0 multi-node cluster on...', 'Namenode information', and a '+' icon. Below this is a green navigation bar with tabs: 'Hadoop' (selected), 'Overview' (highlighted in green), 'Datanodes', 'Datanode Volume Failures', 'Snapshot', 'Startup Progress', and 'Utilities'. The main content area displays the title 'Overview 'hadoop0:8020' (active)'.

## Overview 'hadoop0:8020' (active)

<b>Started:</b>	Mon Apr 05 15:48:05 +0530 2021
<b>Version:</b>	3.2.2, r7a3bc90b05f257c8ace2f76d74264906f0f7a932
<b>Compiled:</b>	Sun Jan 03 14:56:00 +0530 2021 by hexiaoqiao from branch-3.2.2
<b>Cluster ID:</b>	CID-fa6d1471-d265-4fa3-bef8-6ee93f437fb4
<b>Block Pool ID:</b>	BP-2126441114-172.18.0.3-1617617811131

Click on DataNodes:

## In operation

In operation								
Node		Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ hadoop0:9866 (172.18.0.3:9866)		<a href="http://hadoop0:9864">http://hadoop0:9864</a>	0s	1m	58.42 GB 	0	24 KB (0%)	3.2.2

Showing 1 to 1 of 1 entries

Previous **1** Next

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:

etc/hadoop/core-site.xml:

```
<configuration>  
  <property>  
    <name>fs.defaultFS</name>  
    <value>hdfs://hadoopo: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
```

```
[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.

## In operation

Apache Hadoop 3.3.0 – HDFS DataNode Admin Guide								
Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version	
hadoop0:9866 (172.18.0.3:9866)	<a href="http://hadoop0:9864">http://hadoop0:9864</a>	2s	22m	58.42 GB	0	28 KB (0%)	3.2.2	
hadoop1:9866 (172.18.0.2:9866)	<a href="http://hadoop1:9864">http://hadoop1:9864</a>	0s	0m	58.42 GB	0	24 KB (0%)	3.2.2	

Showing 1 to 2 of 2 entries

Previous **1** Next

Make the HDFS directories required to execute MapReduce jobs:

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

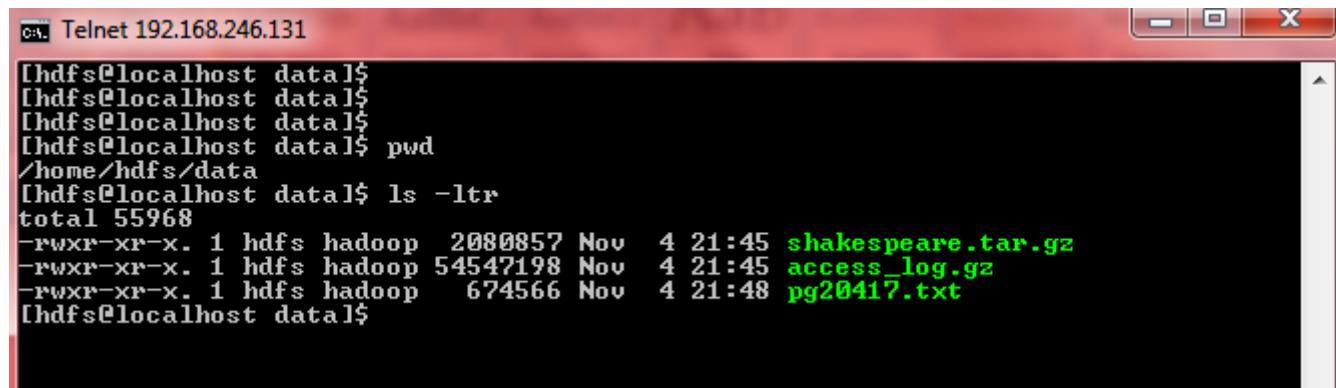
You have successfully configured a Two nodes Hadoop Cluster - HDFS.

----- 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
```



A screenshot of a Windows Telnet window titled "Telnet 192.168.246.131". The window shows a command-line interface where the user has run several commands to list files in the HDFS "data" directory. The output is as follows:

```
[hdfs@localhost data]$  
[hdfs@localhost data]$  
[hdfs@localhost data]$  
[hdfs@localhost data]$ pwd  
/home/hdfs/data  
[hdfs@localhost data]$ ls -ltr  
total 55968  
-rwxr-xr-x. 1 hdfs hadoop 2080857 Nov 4 21:45 shakespeare.tar.gz  
-rwxr-xr-x. 1 hdfs hadoop 54547198 Nov 4 21:45 access_log.gz  
-rwxr-xr-x. 1 hdfs hadoop 674566 Nov 4 21:48 pg20417.txt  
[hdfs@localhost data]$
```

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.

```
-rwxr-xr-x. 1 hdfs hdfs 674566 Nov  4 21:48 pg20417.txt
[hdfs@localhost data]$ hadoop
Usage: hadoop [--config confdir] COMMAND
    where COMMAND is one of:
        fs                  run a generic filesystem user client
        version             print the version
        jar <jar>           run a jar file
        checknative [-a|-h]  check native hadoop and compression libraries availability
        distcp <srcurl> <desturl> copy file or directories recursively
        archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
        classpath            prints the class path needed to get the
        credential           interact with credential providers
        Hadoop jar and the required libraries
        daemonlog            get/set the log level for each daemon
        trace                view and modify Hadoop tracing settings
        or
        CLASSNAME           run the class named CLASSNAME

Most commands print help when invoked w/o parameters.
[hdfs@localhost data]$
```

## 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:

```
[hdfs@localhost data]$  
[hdfs@localhost data]$ hadoop fs  
Usage: hadoop fs [generic options]  
  [-appendToFile <localsrc> ... <dst>]  
  [-cat [-ignoreCrc] <src> ...]  
  [-checksum <src> ...]  
  [-chgrp [-R] GROUP PATH...]  
  [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]  
  [-chown [-R] [OWNER][:GROUP]] PATH...]  
  [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]  
  [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]  
  [-count [-q] [-h] <path> ...]  
  [-cp [-f] [-p | -p[topax]] <src> ... <dst>]  
  [-createSnapshot <snapshotDir> [<snapshotName>]]  
  [-deleteSnapshot <snapshotDir> <snapshotName>]  
  [-df [-h] [<path> ...]]  
  [-du [-s] [-h] <path> ...]  
  [-expunge]  
  [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]  
  [-getfacl [-R] <path>]  
  [-getattr [-R] {-n name | -d} [-e en] <path>]  
  [-getmerge [-n1] <src> <localdst>]  
  [-help [cmd ...]]  
  [-ls [-d] [-h] [-R] [<path> ...]]  
  [-mkdir [-p] <path> ...]  
  [-moveFromLocal <localsrc> ... <dst>]  
  [-moveToLocal <src> <localdst>]  
  [-mv <src> ... <dst>]  
  [-put [-f] [-p] [-l] <localsrc> ... <dst>]  
  [-renameSnapshot <snapshotDir> <oldName> <newName>]  
  [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
```

```
$ 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.

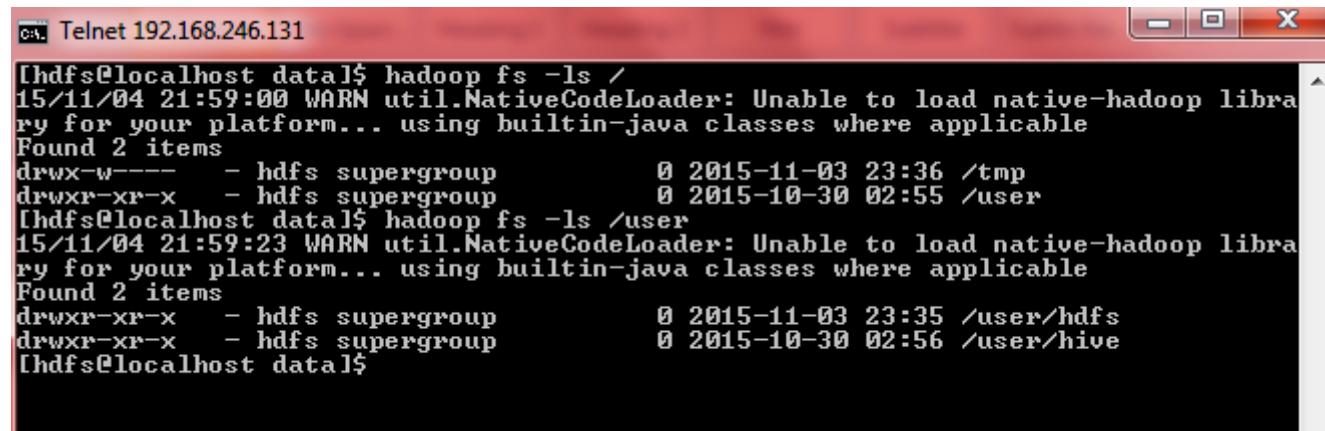
Try viewing the contents of the /user directory by running:

```
$ hadoop fs -ls /user
```

or

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

You will see your home directory in the directory listing.



The screenshot shows a Telnet session connected to 192.168.246.131. The command `hadoop fs -ls /` is run, displaying the contents of the root directory. It shows two items: `/tmp` and `/user`. Both are owned by the hdfs supergroup and were created on 2015-11-03 at 23:36. A warning message about the NativeCodeLoader is displayed twice. The command `hadoop fs -ls /user` is then run, showing the contents of the /user directory. It contains two items: `/user/hdfs` and `/user/hive`, both owned by the hdfs supergroup and created on 2015-10-30 at 02:56. Another warning message about the NativeCodeLoader is displayed.

```
[hdfs@localhost data]$ hadoop fs -ls /
15/11/04 21:59:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwx-w----
```

```
- hdfs supergroup          0 2015-11-03 23:36 /tmp
drwxr-xr-x   - hdfs supergroup          0 2015-10-30 02:55 /user
```

```
[hdfs@localhost data]$ hadoop fs -ls /user
15/11/04 21:59:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x   - hdfs supergroup          0 2015-11-03 23:35 /user/hdfs
drwxr-xr-x   - hdfs supergroup          0 2015-10-30 02:56 /user/hive
[hdfs@localhost data]$
```

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.



A screenshot of a terminal window on a Linux system. The window title is 'Terminal'. The command history shows:

```
[hdfs@localhost data]$ cd ~/data
[hdfs@localhost data]$ pwd
/home/hdfs/data
[hdfs@localhost data]$ ls
access_log.gz pg20417.txt shakespeare.tar.gz
[hdfs@localhost data]$ tar zxvf shakespeare.tar.gz
shakespeare/
shakespeare/comedies
shakespeare/glossary
shakespeare/histories
shakespeare/poems
shakespeare/tragedies
[hdfs@localhost data]$ ls
access_log.gz pg20417.txt shakespeare shakespeare.tar.gz
[hdfs@localhost data]$
```

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]$ hadoop fs -put shakespeare /user/hdfs/shakespeare
15/11/04 22:03:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library 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 library 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_dividends_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-small.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_greater_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.txt
-rw-r--r-- 1 hdfs supergroup 1568 2015-11-03 23:31 /user/hdfs/sample2.txt
drwxr-xr-x - hdfs supergroup 0 2015-11-04 22:03 /user/hdfs/shakespeare
[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
```

Now, extract and upload the file in one step. The `-c` option to gunzip uncompressed to standard output, and the dash (`-`) in the `hadoop fs -put` command takes whatever is being sent to its standard input and places that data in HDFS.

```
$ gunzip -c access_log.gz | hadoop fs -put - weblog/access_log
```

Run the `hadoop fs -ls` command to verify that the log file is in your HDFS home directory.

The access log file is quite large – around 500 MB. Create a smaller version of this file, consisting only of its first 5000 lines, and store the smaller version in HDFS. You can use the smaller version for testing in subsequent labs.

```
$ hadoop fs -mkdir testlog
```

```
$ gunzip -c access_log.gz | head -n 5000 | hadoop fs -put - testlog/test_access_log
```

### Viewing and Manipulating Files

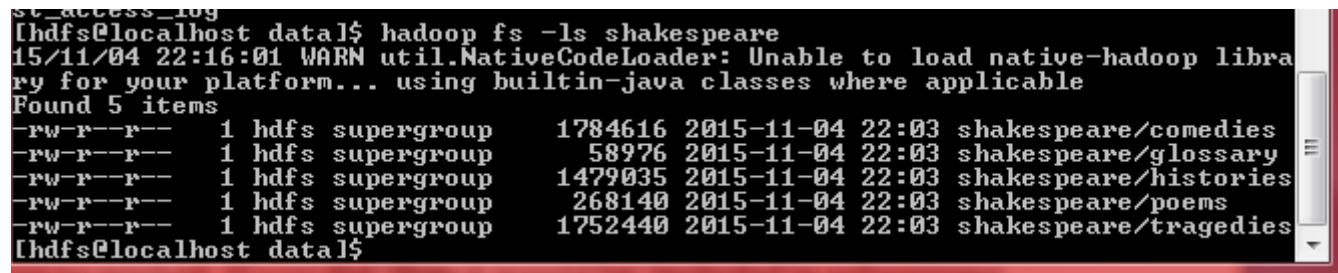
Now let's view some of the data you just copied into HDFS.

Enter:

```
$ hadoop fs -ls shakespeare
```

This lists the contents of the /user/hdfs/shakespeare HDFS directory, which consists of the files comedies, glossary, histories, poems, and tragedies.

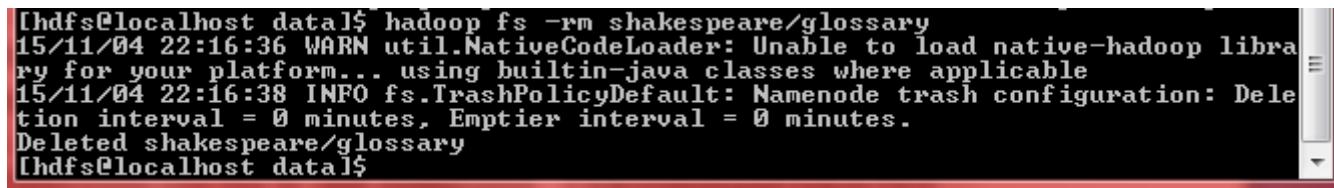
The glossary file included in the compressed file you began with is not strictly a work of Shakespeare, so let's remove it:



```
sc_access_log$ [hdfs@localhost data]$ hadoop fs -ls shakespeare
15/11/04 22:16:01 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 5 items
-rw-r--r-- 1 hdfs supergroup 1784616 2015-11-04 22:03 shakespeare/comedies
-rw-r--r-- 1 hdfs supergroup 58976 2015-11-04 22:03 shakespeare/glossary
-rw-r--r-- 1 hdfs supergroup 1479035 2015-11-04 22:03 shakespeare/histories
-rw-r--r-- 1 hdfs supergroup 268140 2015-11-04 22:03 shakespeare/poems
-rw-r--r-- 1 hdfs supergroup 1752440 2015-11-04 22:03 shakespeare/tragedies
[hdfs@localhost data]$
```

```
$ hadoop fs -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.



```
[hdfs@localhost data]$ hadoop fs -rm shakespeare/glossary
15/11/04 22:16:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
15/11/04 22:16:38 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted shakespeare/glossary
[hdfs@localhost data]$
```

Enter:

```
$ hadoop fs -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.

```
[hdfs@localhost data]$ hadoop fs -cat shakespeare/histories | tail -n 50
15/11/04 22:17:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
RICHMOND      God and your arms be praised, victorious friends,
              The day is ours, the bloody dog is dead.

DERBY    Courageous Richmond, well hast thou acquit thee.
        Lo, here, this long-usurped royalty
        From the dead temples of this bloody wretch
        Have I pluck'd off, to grace thy brows withal:
        Wear it, enjoy it, and make much of it.

RICHMOND      Great God of heaven, say Amen to all!
              But, tell me, is young George Stanley living?

DERBY    He is, my lord, and safe in Leicester town;
        Whither, if it please you, we may now withdraw us.

RICHMOND      What men of name are slain on either side?

DERBY    John Duke of Norfolk, Walter Lord Ferrers,
        Sir Robert Brakenbury, and Sir William Brandon.
```

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:

```
$ hadoop fs -get shakespeare/poems ~/shakepoems.txt
$ less ~/shakepoems.txt
```

```
[Hdfs@localhost data]$  
[Hdfs@localhost data]$  
[Hdfs@localhost data]$ hadoop fs -get shakespeare/poems ~/shakepoems.txt  
15/11/04 22:18:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
[Hdfs@localhost data]$ less ~/shakepoems.txt  
  
SONNETS  
  
TO THE ONLY BEGETTER OF  
THESE INSUING SONNETS  
MR. W. H. ALL HAPPINESS  
AND THAT ETERNITY  
PROMISED BY  
OUR EVER-LIVING POET WISHETH  
THE WELL-WISHING  
ADVENTURER IN  
SETTING FORTH  
T. T.  
  
I.  
FROM S.
```

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:

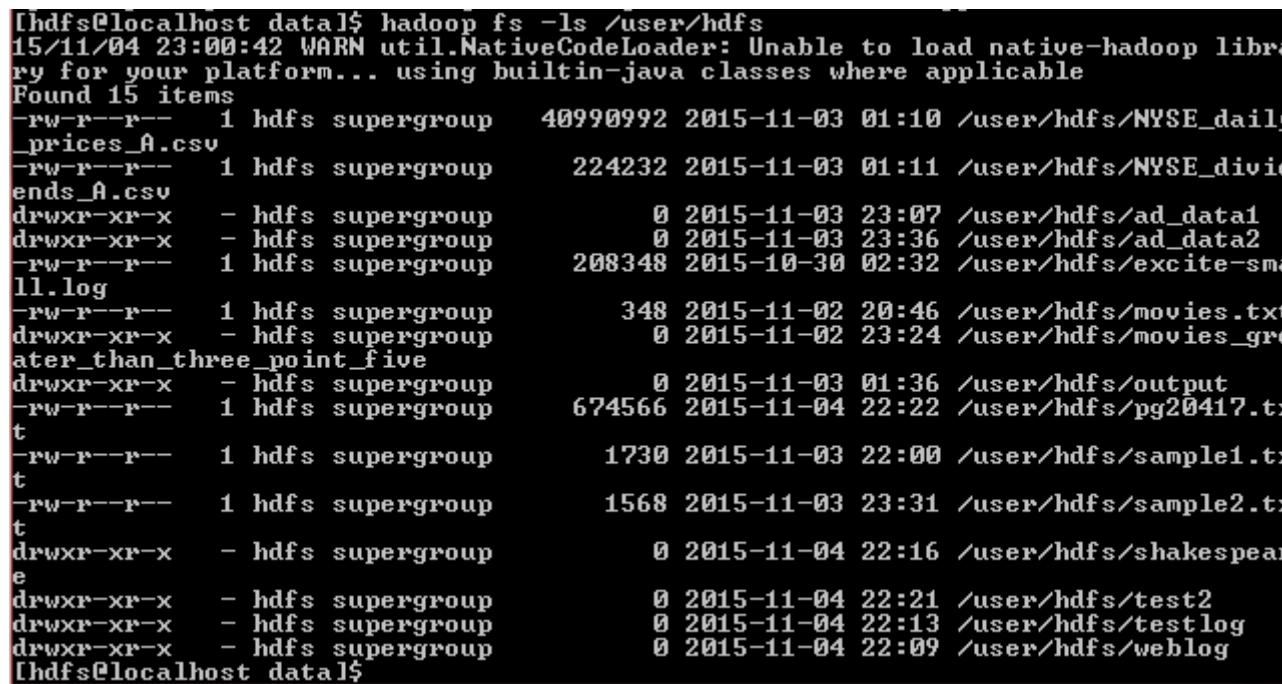
```
hadoop fs -mkdir /user/hdfs/test2  
hadoop fs -ls /user/hdfs | grep test
```

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
```



```
[hdfs@localhost data]$ hadoop fs -ls /user/hdfs
15/11/04 23:00:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 15 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_dividends_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-smal1.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 674566 2015-11-04 22:22 /user/hdfs/pg20417.tx
t
-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 22:16 /user/hdfs/shakespeare
drwxr-xr-x - hdfs supergroup 0 2015-11-04 22:21 /user/hdfs/test2
drwxr-xr-x - hdfs supergroup 0 2015-11-04 22:13 /user/hdfs/testlog
drwxr-xr-x - hdfs supergroup 0 2015-11-04 22:09 /user/hdfs/weblog
[hdfs@localhost data]$
```

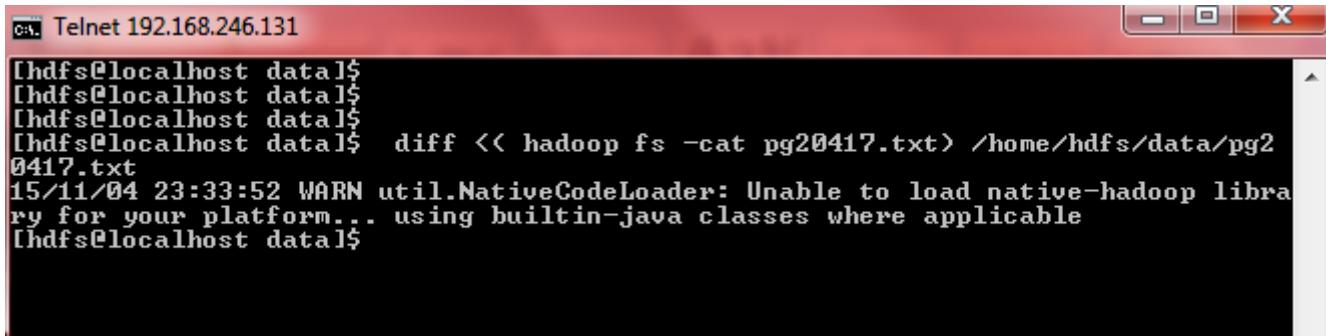
You should now see a new file called /user/hdfs/pg\* listed. In order to view the contents of this file we will use the *-cat* command as follows:

```
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).



The screenshot shows a Telnet session connected to 192.168.246.131. The command entered is:

```
diff << hadoop fs -cat pg20417.txt> /home/hdfs/data/pg20417.txt
```

The output shows a warning message from the Hadoop NativeCodeLoader:

```
15/11/04 23:33:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

## 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
```

```
[hdfs@localhost data]$ hadoop fs -ls -R /user
15/11/04 23:35:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
drwxr-xr-x  - hdfs supergroup          0 2015-11-04 22:22 /user/hdfs
-rw-r--r--  1 hdfs supergroup  40990992 2015-11-03 01:10 /user/hdfs/NYSE_dally_prices_A.csv
-rw-r--r--  1 hdfs supergroup   224232 2015-11-03 01:11 /user/hdfs/NYSE_dividends_A.csv
drwxr-xr-x  - hdfs supergroup          0 2015-11-03 23:07 /user/hdfs/ad_data1
-rw-r--r--  1 hdfs supergroup          0 2015-11-03 23:07 /user/hdfs/ad_data1/_SUCCESS
-rw-r--r--  1 hdfs supergroup    1556 2015-11-03 23:07 /user/hdfs/ad_data1/part-m-00000
drwxr-xr-x  - hdfs supergroup          0 2015-11-03 23:36 /user/hdfs/ad_data2
-rw-r--r--  1 hdfs supergroup          0 2015-11-03 23:36 /user/hdfs/ad_data2/_SUCCESS
-rw-r--r--  1 hdfs supergroup    1506 2015-11-03 23:36 /user/hdfs/ad_data2/part-r-00000
-rw-r--r--  1 hdfs supergroup  208348 2015-10-30 02:32 /user/hdfs/excite-smalldata
```

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
```

```
[hdfs@localhost data]$ hadoop fs -du /user/hdfs
15/11/04 23:36:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
40990992  /user/hdfs/NYSE_daily_prices_A.csv
224232   /user/hdfs/NYSE_dividends_A.csv
1556     /user/hdfs/ad_data1
1506     /user/hdfs/ad_data2
208348   /user/hdfs/excite-small.log
348      /user/hdfs/movies.txt
83       /user/hdfs/movies_greater_than_three_point_five
1992     /user/hdfs/output
674566   /user/hdfs/pg20417.txt
1730     /user/hdfs/sample1.txt
1568     /user/hdfs/sample2.txt
5284231  /user/hdfs/shakespeare
0        /user/hdfs/test2
485246   /user/hdfs/testlog
504941532 /user/hdfs/weblog
[hdfs@localhost data]$ hadoop fs -dus /user/hdfs
dus: DEPRECATED: Please use 'du -s' instead.
15/11/04 23:36:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
552817930 /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>

The screenshot shows the HDFS NameNode Overview page. At the top, there is a navigation bar with tabs: Hadoop, Overview (which is selected and highlighted in green), Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. Below the navigation bar, the title is "Overview 'hadoop0:8020' (active)". A table provides detailed information about the cluster's setup:

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

Below the table, the "Summary" section contains the following information:

- 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 Committed 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

Decommissioning

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

Click on Snapshot



## Snapshot Summary

---

Snapshottable directories: 0

---

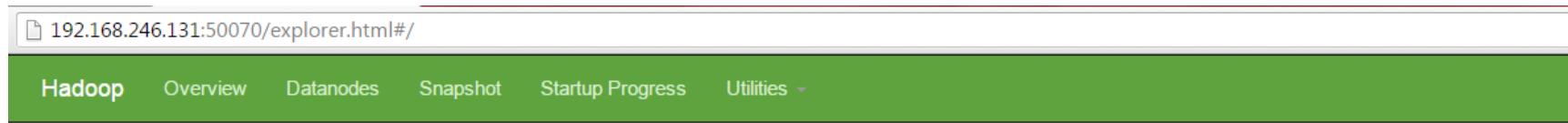
Path	Snapshot Number	Snapshot Quota	Modification Time	Permission	Owner	Group

Snapshotted directories: 0

---

Snapshot ID	Snapshot Directory	Modification Time

Click on Startup Progress and utilities



The screenshot shows a web browser window with the URL `192.168.246.131:50070/explorer.html#/`. The page has a green header bar with the following navigation links: Hadoop, Overview, Datanodes, Snapshot, Startup Progress, and Utilities. The 'Startup Progress' link is highlighted.

## Browse Directory



Permission	Owner	Group	Size	Replication	Block Size	Name
drwx-w----	hdbs	supergroup	0 B	0	0 B	tmp
drwxr-xr-x	hdbs	supergroup	0 B	0	0 B	user

Hadoop, 2014.

## Click on Logs

<http://192.168.246.131:50070/logs/>

← → ⌂ 192.168.246.131:50070/logs/

### Directory: /logs/

[SecurityAuth-hdfs.audit](#)

0 bytes Oct 28, 2015 2:13:24 AM

[hadoop-hdfs-datanode-localhost.localdomain.log](#)

663249 bytes Nov 5, 2015 12:43:38 AM

[hadoop-hdfs-datanode-localhost.localdomain.out](#)

716 bytes Nov 4, 2015 9:55:40 PM

[hadoop-hdfs-datanode-localhost.localdomain.out.1](#)

716 bytes Nov 3, 2015 9:15:58 PM

[hadoop-hdfs-datanode-localhost.localdomain.out.2](#)

3024 bytes Nov 3, 2015 1:43:28 AM

[hadoop-hdfs-datanode-localhost.localdomain.out.3](#)

716 bytes Nov 2, 2015 8:32:23 PM

[hadoop-hdfs-datanode-localhost.localdomain.out.4](#)

716 bytes Oct 30, 2015 3:00:16 AM

[hadoop-hdfs-datanode-localhost.localdomain.out.5](#)

716 bytes Oct 30, 2015 1:36:15 AM

[hadoop-hdfs-namenode-localhost.localdomain.log](#)

1338712 bytes Nov 5, 2015 12:52:09 AM

[hadoop-hdfs-namenode-localhost.localdomain.out](#)

4913 bytes Nov 4, 2015 10:13:01 PM

[hadoop-hdfs-namenode-localhost.localdomain.out.1](#)

716 bytes Nov 3, 2015 9:15:47 PM

[hadoop-hdfs-namenode-localhost.localdomain.out.2](#)

4908 bytes Nov 3, 2015 1:42:15 AM

[hadoop-hdfs-namenode-localhost.localdomain.out.3](#)

716 bytes Nov 2, 2015 8:32:13 PM

[hadoop-hdfs-namenode-localhost.localdomain.out.4](#)

716 bytes Oct 30, 2015 3:00:07 AM

[hadoop-hdfs-namenode-localhost.localdomain.out.5](#)

716 bytes Oct 30, 2015 1:36:07 AM

[hadoop-hdfs-secondarynamenode-localhost.localdomain.log](#)

49486 bytes Oct 29, 2015 4:56:06 AM

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.

```
#hadoop fsck /
```

or

```
# 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?

```
--  
Name: (null) : (1):  
Profile: (null)  
Command: None  
Port: 50070.2:9866 (hadoop0)  
Hostname: hadoop0  
Decommission Status : Normal  
Configured Capacity: 62725623808 (58.42 GB)  
DFS Used: 273555456 (260.88 MB)  
Non DFS Used: 30131851264 (28.06 GB)  
DFS Remaining: 29103501312 (27.10 GB)  
DFS Used%: 0.44%  
DFS Remaining%: 46.40%  
Configured Cache Capacity: 0 (0 B)  
Cache Used: 0 (0 B)  
Cache Remaining: 0 (0 B)  
Cache Used%: 100.00%  
Cache Remaining%: 0.00%  
Xceivers: 1  
Last contact: Thu Apr 15 08:54:31 UTC 2021  
Last Block Report: Thu Apr 15 06:03:47 UTC 2021  
Num of Blocks: 276
```

```
#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-datanode-hadoopmaster.log   hadoop-root-namenode-%computername%.log
hadoop-root-datanode-hadoopmaster.out    hadoop-root-namenode-%computername%.out
hadoop-root-datanode-hadoopmaster.out.1  hadoop-root-namenode-hadoopmaster.log
hadoop-root-datanode-hadoopmaster.out.2  hadoop-root-namenode-hadoopmaster.out
hadoop-root-jobtracker-%computername%.log hadoop-root-namenode-hadoopmaster.out.1
hadoop-root-jobtracker-%computername%.out hadoop-root-namenode-hadoopmaster.log
hadoop-root-jobtracker-hadoopmaster.log   hadoop-root-namenode-hadoopmaster.out
hadoop-root-jobtracker-hadoopmaster.out   hadoop-root-namenode-hadoopmaster.out.1
                                         hadoop-root-secondarynamenode-hadoopmaster.out.2
                                         hadoop-root-tasktracker-hadoopmaster.log
                                         hadoop-root-tasktracker-hadoopmaster.out
                                         hadoop-root-tasktracker-hadoopmaster.out.1
                                         hadoop-root-tasktracker-hadoopmaster.out.2
                                         hadoop.txt
                                         history
                                         userlogs
```

```
# vi hadoop.txt
```

```
[root@hadoopmaster:/hadoop/hadoop/logs]
2 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
drwxrwxrwx  - 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
```

```
[root@hadoop0 var]# hdfs dfs -mkdir henry
2021-04-15 09:08:28,412 INFO Configuration.deprecation: No unit for dfs.client.datanode-restart.timeout(30) assuming SECONDS
mkdir: Cannot create directory /user/root/henry. Name node is in safe mode.
[root@hadoop0 var]#
```

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]#
```

To get a list of files in a directory you would use using REST API:

```
#curl -i http://localhost:9870/webhdfs/v1/user/root/output/?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,"modificationTime":1617780861961,"owner":"root","pathSuffix":"part-r-00000","permission":"644","replication":2,"storagePolicy":0,"type":"FILE"}]}]
[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.

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_HOME</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>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_HOME</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.

The screenshot shows the Hadoop Node Manager UI titled "Nodes of the cluster". The top right corner indicates the user is "Logged in as: dr.who". The left sidebar has sections for Cluster (About Nodes, Node Labels, Applications, Scheduler) and Tools. The main area displays "Cluster Metrics" and "Cluster Nodes Metrics". A red circle highlights the value "1" under "Active Nodes" in the "Cluster Nodes Metrics" table. Below these are "Scheduler Metrics" and a detailed "Node Labels" table. The "Node Labels" table lists one node entry:

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 03:09:43 +0000 2021			0		0 B	8 GB	44	0	8	1	0	0	3.2.2

At the bottom, it says "Showing 1 to 1 of 1 entries" and provides navigation links: First, Previous, 1, Next, Last.

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.

Logged in as: ur.who

# 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 /
```

```
[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]#
```

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>

The screenshot shows the RM web UI interface. On the left, there's a sidebar with a 'Cluster' section containing links for About, Nodes, Node Labels, Applications (with sub-options: NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED), and Scheduler. Below that is a 'Tools' section. The main content area has three tabs: Cluster Metrics, Cluster Nodes Metrics, and Scheduler Metrics. Under Scheduler Metrics, there's a table titled 'Show 20 entries' showing application details. One row is highlighted with a red circle, showing the application ID 'application\_1617678584676\_0001'. The table columns include ID, User, Name, Application Type, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU VCores, and Allocated Memory MB. The application details shown are: User: root, Name: grep-search, Application Type: MAPREDUCE, Queue: default, Application Priority: 0, StartTime: Tue Apr 6 09:10:17 +0550 2021, LaunchTime: Tue Apr 6 09:10:18 +0550 2021, FinishTime: N/A, State: RUNNING, FinalStatus: UNDEFINED, Running Containers: 8, Allocated CPU VCores: 8, and Allocated Memory MB: 9216.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB
application_1617678584676_0001	root	grep-search	MAPREDUCE	default	0	Tue Apr 6 09:10:17 +0550 2021	Tue Apr 6 09:10:18 +0550 2021	N/A	RUNNING	UNDEFINED	8	8	9216

Showing 1 to 1 of 1 entries

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

Capacity Scheduler		[memory_mb (unit=MB), vcores]				[memory_gb (unit=GB), vcores]				[memory_gb (unit=GB), vcores]					
		User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Residual
<a href="#">application_1617780547234_0003</a>	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	
<a href="#">application_1617780547234_0002</a>	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	
<a href="#">application_1617780547234_0001</a>	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	

Showing 1 to 3 of 3 entries

```
Peak Reduce Virtual memory (by)
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=356
File Output Format Counters
  Bytes Written=172
[root@hadoop1 hadoop]#
```

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_0001/
2020-07-02 08:31:23,349 INFO mapreduce.Job: Running job: job_1593678172115_0001
```

```
 docker-hadoop — docker exec -it namenode bash — 133x38
      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 -----

## Capacity Scheduler – 90 minutes

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.

The screenshot shows the Hadoop Job Tracker UI with the title "NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING Applications". On the left, there's a sidebar with "Cluster Metrics", "Cluster Nodes Metrics", "Scheduler Metrics", and a "Dump scheduler logs" button. The main area displays "Application Queues" with a legend: Capacity (grey), Used (green), Used (over capacity) (orange), Max Capacity (light grey), Users Requesting Resources (yellow), and Auto Created Queues (light grey). A red circle highlights the "Queue: root" section, which lists "Queue: marketing", "Queue: default", and "Queue: sales". Below this, there's a table header for application details and a message "No data available in table".

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 output 'dfs[a-z.]+'
```

```
SchedulingInfo : capacity: 50.0, maximumCapacity: 100.0, currentCapacity: 0.0
[root@master0 hadoop]# hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar grep -D mapreduce.job.que
ueueName=sales input output 'dfs[a-z.]+'  
2022-06-18 04:29:36,581 INFO client.RMProxy: Connecting to ResourceManager at master0/172.18.0.2:8032  
2022-06-18 04:29:40,196 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.stag
ing/job_1655521605768_0007  
2022-06-18 04:29:41,863 INFO input.FileInputFormat: Total input files to process : 10  
2022-06-18 04:29:42,380 INFO mapreduce.JobSubmitter: number of splits:10  
2022-06-18 04:29:42,968 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1655521605768_0007  
2022-06-18 04:29:42,975 INFO mapreduce.JobSubmitter: Executing with tokens: □  
2022-06-18 04:29:44,302 INFO conf.Configuration: resource-types.xml not found  
2022-06-18 04:29:44,309 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.  
2022-06-18 04:29:44,877 INFO impl.YarnClientImpl: Submitted application application_1655521605768_0007  
2022-06-18 04:29:45,215 INFO mapreduce.Job: The url to track the job: http://master0:8088/proxy/application_1655521605768_0007/  
2022-06-18 04:29:45,221 INFO mapreduce.Job: Running job: job_1655521605768_0007
```

While the job is executing, monitor the Resource Manager 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.

Logged in as: dr.who

## NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING Applications

The screenshot shows the Hadoop Cluster Metrics interface. On the left, there's a sidebar with links for Cluster (About, Nodes, Node Labels, Applications), Scheduler, and Tools. The Applications section lists various states: NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED. The main area displays several metrics tables and a bar chart for Application Queues.

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved
7	0	1	6	1	1.50 GB	4 GB	0 B	1	8	0

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	2

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=MB), vcores]	<memory:512, vCores:1>	<memory:4096, vCores:4>	0

**Dump scheduler logs** 1 min

**Application Queues**

Legend: Capacity (grey), Used (green), Used (over capacity) (orange), Max Capacity (light grey), Users Requesting Resources (yellow), Auto Created Queues (light yellow).

Queue	Capacity (%)	Used (%)	Over Capacity (%)
Queue: root	37.5%	37.5%	0.0%
Queue: marketing	0.0%	0.0%	0.0%
Queue: default	0.0%	0.0%	0.0%
Queue: sales	75.0%	75.0%	0.0%

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No matching records found																			

Showing 0 to 0 of 0 entries (filtered from 4 total entries). Client | Download | Block | Email

Verify the end result from the Namenode UI.

## Browse Directory

The screenshot shows a HDFS browser interface. At the top, there is a navigation bar with a search bar containing '/user/root/output' and a 'Go!' button. Below the search bar are icons for folder, up, and list. To the right of the search bar is a 'Search:' input field. In the center, there is a table displaying file entries. The table has columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. There are two entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	0 B	Jun 18 09:03	2	128 MB	_SUCCESS
-rw-r--r--	root	supergroup	73 B	Jun 18 09:03	2	128 MB	part-00000-7e299e14-a21d-4c34-83bb-26341fe4e600-c000.csv

At the bottom left, it says 'Showing 1 to 2 of 2 entries'. At the bottom right, there are buttons for 'Previous', '1' (highlighted in blue), and 'Next'.

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](https://community.pivotal.io/s/article/How-to-configure-queues-using-YARN-capacity-scheduler-xml?language=en_US)

## HDFS High Availability Using the Quorum Journal Manager – 3 Hrs

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

IP	hostname	Remarks	Services
10.10.20.20	master0	First Machine	NN, DN , JN
10.10.20.21	master1	2 Machine	NN , DN , JN
10.10.20.25	master2	New Machine	JN

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.

In operation

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ master0.hp.com:9866 (172.18.0.2:9866)	<a href="http://master0.hp.com:9864">http://master0.hp.com:9864</a>	0s	2m	465.74 GB	77	21.25 MB (0%)	3.1.2
✓ master1.hp.com:9866 (172.18.0.3:9866)	<a href="http://master1.hp.com:9864">http://master1.hp.com:9864</a>	0s	1m	465.74 GB	34	923.65 KB (0%)	3.1.2
✓ master2.hp.com:9866 (172.18.0.4:9866)	<a href="http://master2.hp.com:9864">http://master2.hp.com:9864</a>	0s	0m	465.74 GB	0	768 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.

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>master0.hp.com:8020</value>
</property>
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn2</name>
  <value>master1.hp.com: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://master0.hp.com:8485;master1.hp.com:8485;master0.hp.com: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 master 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:

```
[root@master0 hadoop]#  
[root@master0 hadoop]#  
[root@master0 hadoop]# jps  
577 SecondaryNameNode  
1154 Jps  
404 DataNode  
293 NameNode  
1096 JournalNode  
[root@master0 hadoop]#
```

You should observe journal process as highlighted above.

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

On Master node, Stop the Namenode (master0) – Jps and Use kill -9 <the process ID>

```
#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 master1 to sync the meta data.

```
# hdfs namenode -bootstrapStandby
```

```
About to bootstrap Standby ID nn2 from:
    Nameservice ID: mycluster
    Other Namenode ID: nn1
Other NN's HTTP address: http://master0.hp.com:9870
Other NN's IPC address: master0.hp.com/172.18.0.2:8020
    Namespace ID: 1550658770
    Block pool ID: BP-940677331-172.18.0.2-1654073176360
    Cluster ID: CID-e4a88a05-b6fd-4f87-bcc7-694102fc7581
    Layout version: -64
    isUpgradeFinalized: true

2022-06-01 08:54:53,708 INFO common.Storage: Storage directory /opt/hdfs1/namenode has been successfully formatted.
2022-06-01 08:54:53,995 INFO namenode.FSEditLog: Edit logging is async:true
2022-06-01 08:54:54,500 INFO namenode.TransferFsImage: Opening connection to http://master0.hp.com:9870/imagetransfer?getimage=1&txid=0&storageInfo=-64:1550658770:1654073176360:CID-e4a88a05-b6fd-4f87-bcc7-694102fc7581&bootstrapstandby=true
2022-06-01 08:54:54,630 INFO common.Util: Combined time for file download and fsync to all disks took 0.02s. The file download took 0.01s at 0.00 KB/s. Synchronous (fsync) write to disk of /opt/hdfs1/namenode/current/fsimage.ckpt_00000000000000000000 took 0.00s.
2022-06-01 08:54:54,634 INFO namenode.TransferFsImage: Downloaded file fsimage.ckpt_00000000000000000000 size 391 bytes.
2022-06-01 08:54:54,803 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at master1/172.18.0.3
************************************************************/
[root@master1 hadoop1]#
```

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

```
Master 1
#hdfs --daemon start namenode
# hdfs haadmin -getAllServiceState
```

```
master1.hp.com:8020          standby
[root@master1 hadoop1]# hdfs haadmin -getAllServiceState
master0.hp.com:8020          standby
master1.hp.com:8020          standby
[root@master1 hadoop1]#
```

Both the Name Nodes are in standby Mode.

Get service state by service ID.

```
# hdfs haadmin -getServiceState nn2
```

```
[root@master1 hadoop1]# hdfs haadmin -getServiceState nn1
standby
[root@master1 hadoop1]# hdfs haadmin -getServiceState nn2
standby
```

You can enable active NN on master0:

```
# 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>

The screenshot shows a web browser window titled "localhost:9870/dfshealth.html#tab-overview". The address bar also displays "localhost:9870/dfshealth.html#tab-overview". The page has a green header bar with tabs: "Hadoop" (selected), "Overview", "Datanodes", "Datanode Volume Failures", "Snapshot", "Startup Progress", and "Utilities". Below the header, the main content area is titled "Overview 'master0.hp.com:8020' (active)". A table provides detailed information about the cluster:

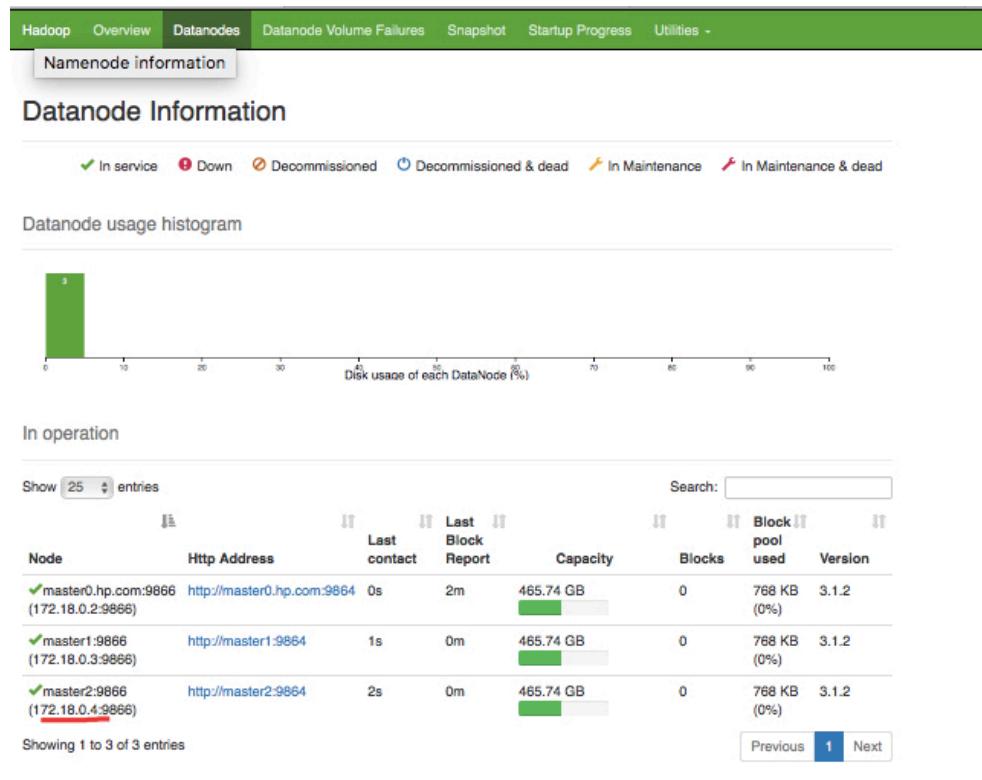
Namespace:	mycluster
Namenode ID:	nn1
Started:	Wed Jun 01 14:26:32 +0530 2022
Version:	3.1.2, r1019dde65bcf12e05ef48ac71e84550d589e5d9a
Compiled:	Tue Jan 29 07:09:00 +0530 2019 by sunilg from branch-3.1.2
Cluster ID:	CID-e4a88a05-b6fd-4f87-bcc7-694102fc7581
Block Pool ID:	BP-940677331-172.18.0.2-1654073176360

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

Start Datanode on all 3 Nodes:

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

Verify the Datanodes:



You should have 3 data nodes in the Cluster.

Let us create a directory and move a file to the cluster, From Mastero .

```
# 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@master2 hadoop]# hdfs haadmin -getAllServiceState  
master0.hp.com:8020          active  
master1.hp.com:8020          standby  
[root@master2 hadoop]#
```

In my case, mastero is active. Let us make the master1 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
```

```
[root@master2 hadoop]# hdfs haadmin -transitionToStandby nn1
[root@master2 hadoop]# hdfs haadmin -transitionToActive nn2
[root@master2 hadoop]# hdfs haadmin -getAllServiceState
master0.hp.com:8020                      standby
master1.hp.com:8020                      active
[root@master2 hadoop]#
```

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

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

```
[root@master0 datanode]# hdfs dfs -ls -R hdfs://mycluster/henry
-rw-r--r--  2 root supergroup      1366 2022-06-01 09:21 hdfs://mycluster/henry/README.txt
```

Click on the master1 namenode URL.

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

The screenshot shows the Hadoop Web UI interface. At the top, there's a navigation bar with links: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. Below the navigation bar, the main title is "Browse Directory". In the search bar, the path "/henry" is entered. To the right of the search bar are buttons for "Go!", "File", "Edit", and "Delete". Below the search bar, there are filters: "Show 25 entries" and a "Search" input field. A "Namenode information" button is also present. The main content area is a table listing the directory entries. The columns are: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The single entry is: "-rw-r--r--" (permissions), "root" (owner), "supergroup" (group), "1.33 KB" (size), "Jun 01 14:51" (last modified), "2" (replication), "128 MB" (block size), and "README.txt" (name). At the bottom left, it says "Showing 1 to 1 of 1 entries". At the bottom right, there are "Previous", "1", and "Next" buttons.

Hadoop, 2018.

Completed the lab execution.

Core-site.xml

```
<configuration>
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://mycluster</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>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>master0.hp.com:8020</value>
</property>
<property>
<name>dfs.namenode.rpc-address.mycluster.nn2</name>
<value>master1.hp.com:8020</value>
</property>
<property>
<name>dfs.namenode.shared.edits.dir</name>
<value>qjournal://master0.hp.com:8485;master1.hp.com:8485;master2.hp.com: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>
<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>
----- Ends hdfs.site.xml -----
```

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

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

## ResourceManager high availability – 2 Hrs

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

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

Activate one Namenode as active.

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

Start All datanodes

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

IP	hostname	Remarks	Services
10.10.20.20	master0	First Machine	NN, DN , JN , RM , NM
10.10.20.21	master1	2nd Machine	NN , DN , JN , RM , NM
10.10.20.25	master2	New Machine	JN , ZK – 1 , 2 , 3 , NM

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

Stop Yarn Node manager if started.

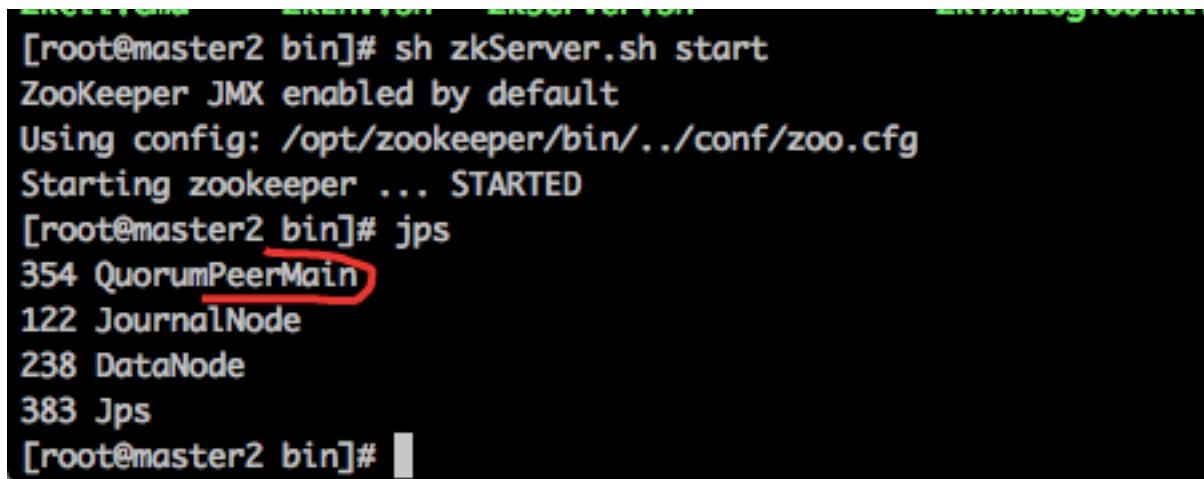
Add the following configuration in the yarn-site.xml on all nodes.

```
<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>master0</value>
</property>
<property>
<name>yarn.resourcemanager.hostname.rm2</name>
<value>master1</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address.rm1</name>
<value>master0:8088</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address.rm2</name>
<value>master1:8088</value>
</property>
<property>
<name>hadoop.zk.address</name>
<value>master2:2181</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 node and start it.



```
[root@master2 bin]# sh zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@master2 bin]# jps
354 QuorumPeerMain
122 JournalNode
238 DataNode
383 Jps
[root@master2 bin]#
```

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://master0:8088/cluster/cluster>

Mastero

Logged in as: dr.who



## About the Cluster

**Cluster**

- [About](#)
- [Nodes](#)
- [Node Labels](#)
- [Applications](#)
  - [NEW](#)
  - [NEW\\_SAVING](#)
  - [SUBMITTED](#)
  - [ACCEPTED](#)
  - [RUNNING](#)
  - [FINISHED](#)
  - [FAILED](#)
  - [KILLED](#)
- [Scheduler](#)

**Tools**

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	0 B	0 B	0	0	0

**Cluster Nodes Metrics**

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

**Cluster overview**

Cluster ID:	1654090647244
ResourceManager state:	STARTED
ResourceManager HA state:	active
ResourceManager HA zookeeper connection state:	CONNECTED
ResourceManager RMStateStore:	org.apache.hadoop.yarn.server.resourcemanager.recovery.NullRMStateStore
ResourceManager started on:	Wed Jun 01 13:37:27 +0000 2022
ResourceManager version:	3.1.2 from 1019dde65bcf12e05ef48ac71e84550d589e5d9a by sunilg source checksum 7954337bcd9688eca8aa32720d2c74 on 2019-01-29T02:04Z
Hadoop version:	3.1.2 from 1019dde65bcf12e05ef48ac71e84550d589e5d9a by sunilg source checksum

master1 – Standby

The screenshot shows the 'About the Cluster' page of a Hadoop cluster. The top navigation bar includes links for 'About the Cluster', 'Nodes', 'Logs', 'Metrics', 'Scheduler', and 'Tools'. The main content area is titled 'About the Cluster' and displays various metrics and configuration details.

**Cluster Metrics:**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	0 B	0 B	0	0	0

**Cluster Nodes Metrics:**

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

**Cluster overview:**

Cluster ID:	1654090689181
ResourceManager state:	STARTED
ResourceManager HA state:	standby
ResourceManager HA zookeeper connection state:	CONNECTED
ResourceManager RMStateStore:	org.apache.hadoop.yarn.server.resourcemanager.recovery.NullRMStateStore
ResourceManager started on:	Wed Jun 01 13:38:09 +0000 2022
ResourceManager version:	3.1.2 from 1019dde65bcf12e05ef48ac71e84550d589e5d9a by sunilg source checksum 7954337bcd9688eca8aa32720d2c74 on 2019-01-29T02:04Z
Hadoop version:	3.1.2 from 1019dde65bcf12e05ef48ac71e84550d589e5d9a by sunilg source checksum

Start NodeManager: On all the three nodes.

```
#yarn --daemon start nodemanager
```

Verify the number of Nodes using the RM UI.

The screenshot shows the YARN Resource Manager (RM) UI. The left sidebar has sections for Cluster (About, Nodes, Node Labels, Applications, Scheduler), Tools, and a expanded Applications section showing NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED. The Applications section is circled in red. The main area shows Cluster Metrics (0 Apps Submitted, 0 Pending, 0 Running, 0 Completed, 0 Containers Running, 0 B Memory Used, 24 GB Total, 0 Reserved, 0 Vcores Used, 24 Total, 0 Reserved) and Cluster Nodes Metrics (3 Active Nodes, 0 Decommissioning, 0 Decommissioned, 0 Lost, 0 Unhealthy, 0 Rebooted, 0 Shutdown). The Scheduler Metrics table shows Capacity Scheduler, Scheduling Resource Type [memory-mb (unit=Mi), vcores], Minimum Allocation <memory:1024, vcores:1>, Maximum Allocation <memory:8192, vcores:4>, and Maximum Cluster Application Priority 0. The Cluster Nodes table lists three nodes:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Vcores Used	Vcores Avail	
/default-rack		RUNNING	master1:46517	master1:8042	Wed Jun 01 13:46:30 +0000 2022		0		0 B	8 GB	0	8	3.1
/default-rack		RUNNING	master2:34515	master2:8042	Wed Jun 01 13:46:36 +0000 2022		0		0 B	8 GB	0	8	3.1
/default-rack		RUNNING	master0:36789	master0:8042	Wed Jun 01 13:46:13		0		0 B	8 GB	0	8	3.1

---

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]#  
[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 – master2 node

```
#export YARN_EXAMPLES=/opt/hadoop2/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.]+'
```

```
Failed Shuffles=0
Merged Map outputs=10
GC time elapsed (ms)=8014
CPU time spent (ms)=15760
Physical memory (bytes) snapshot=2656550912
Virtual memory (bytes) snapshot=28797493248
Total committed heap usage (bytes)=2236088320
Peak Map Physical memory (bytes)=266461184
Peak Map Virtual memory (bytes)=2621177856
Peak Reduce Physical memory (bytes)=146341888
Peak Reduce Virtual memory (bytes)=2624794624
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=1180
File Output Format Counters
Bytes Written=97
Job Finished in 174.384 seconds
Estimated value of Pi is 3.160000000000000000000000000000
[root@master2 hadoop]#
```

## yarn-site.xml

```
<configuration>

    <!-- Site specific YARN configuration properties -->
    <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>master0</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname.rm2</name>
        <value>master1</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address.rm1</name>
        <value>master0:8088</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address.rm2</name>
        <value>master1:8088</value>
    </property>
```

```

<property>
  <name>hadoop.zk.address</name>
  <value>master2:2181</value>
</property>
<property>
<property>
  <name>yarn.nodemanager.local-dirs</name>
  <value>file:///mydata/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/hadoop/mapreduce/*,$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/*,$HADOOP_YARN_HOME/share/hadoop/yarn/lib/*
  </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_HOME</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</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 -----

## Archival Disk Storage – 60 Minutes

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

Shut down the HDFS 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.

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 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 hdfs-site.xml.

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

```
<property>
<name>dfs.namenode.datanode.registration.ip-hostname-check</name>
<value>false</value>
</property>
<property>
<name> dfs.storage.policy.enabled</name>
<value>true</value>
</property>
</configuration>
~/opt/hadoop/etc/hadoop/hdfs-site.xml" 50L, 1536C written
```

Save it. Accept all the default setting.

## ***Configure an archive data node – On the Second Node (master1).***

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

[ARCHIVE]/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.

```
# stop-dfs.sh
# start-dfs.sh
```

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 COLD
Set storage policy COLD on /archives
[hdfs@tos ~]$ hdfs storagepolicies -getStoragePolicy -path /archives
The storage policy of /archives:
BlockStoragePolicy{COLD:2, storageTypes=[ARCHIVE], creationFallbacks=[], replicationFallbacks=[]}
[hdfs@tos ~]$
```

### ***Verify the setup.***

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

And scroll to the volume 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

The screenshot shows the Hadoop Web UI interface. At the top, there is a navigation bar with links: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. A red circle highlights the 'Utilities' link. Below the navigation bar is a search bar with the path '/'. To the right of the search bar are icons for Go!, folder, upload, and download. Underneath the search bar, there are buttons for Show (set to 25 entries) and Search (with an empty input field). The main area displays a table of file system entries:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	Action
<input type="checkbox"/>	drwxrwxrwx	root	supergroup	0 B	Jun 08 21:21	0	0 B	archives	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	658.76 KB	Jun 08 19:39	2	128 MB	tmp	
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Jun 08 19:27	0	0 B	user	

At the bottom left, it says 'Showing 1 to 3 of 3 entries'. At the bottom right, there are buttons for Previous, Next, and a page number '1'.

Next → archives -> pg20417.txt

## Browse Directory

The screenshot shows a 'Browse Directory' interface. At the top, there is a search bar containing '/archives' with a red oval around it. To the right of the search bar are buttons for 'Go!', a folder icon, an upload icon, and a refresh icon. Below the search bar, there are buttons for 'Show 25 entries' and a 'Search:' input field. The main area displays a table of file entries:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	658.76 KB	Jun 08 21:21	1	128 MB	pg20417.txt	

Below the table, a message says 'Showing 1 to 1 of 1 entries'. At the bottom right, there are navigation buttons for 'Previous', a blue '1' button, and 'Next'.

Hadoop, 2018.

Click on the above file.

The screenshot shows a 'File information - pg20417.txt' dialog box. At the top, there are three buttons: 'Download', 'Head the file (first 32K)', and 'Tail the file (last 32K)'. Below this is a green header bar with 'Block information -- Block 0'. The main content area displays the following information:

- Block ID: 1073741828
- Block Pool ID: BP-1846743683-172.18.0.2-1654695684877
- Generation Stamp: 1004
- Size: 674566
- Availability:
  - master1.hp.com

A red oval highlights the 'Availability' section. In the bottom right corner of the content area, there is a watermark that reads 'Apache Hadoop 3.3.3 – A'. At the very bottom right of the dialog box is a green 'Close' button.

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

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

## Hive Installation – 90 Minutes

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

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

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 -xzvf apache*hive-* .gz -C /opt
```

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

Set the environment variable `HIVE_HOME` to point to the installation directory:

```
$ export HIVE_HOME=/opt/hive
```

Finally, add `$HIVE_HOME/bin` to your `PATH`:

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

## Running Hive

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

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

Start HDFS cluster

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.

```
[root@hadoop0 opt]# pwd  
/opt  
[root@hadoop0 opt]# ls  
back derby hadoop hdfs jdk names.json  
db-derby-10.14.2.0-bin.tar.gz derby.log hadoop_3.2.2 hive metastore_db yarn  
[root@hadoop0 opt]#
```

```
export DERBY_INSTALL=/opt/derby  
export DERBY_HOME=/opt/derby  
export HADOOP
```

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
```

```
[root@hadoop0 opt]# sh /opt/derby/bin/startNetworkServer -h 0.0.0.0  
Tue Apr 06 14:02:12 UTC 2021 : Security manager installed using the Basic server security policy.  
Tue Apr 06 14:02:12 UTC 2021 : Apache Derby Network Server - 10.14.2.0 - (1828579) started and ready to accept connections  
on port 1527
```

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

```
#cp hive-default.xml.template hive-default.xml
```

### 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://mastero.hp.com: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
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.)

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

To run HiveServer2 and Beeline from shell:

```
$ hiveserver2
```

```
[root@hadoop0 hadoop]# hiveserver2
/opt/hive/bin/hive: line 289: which: command not found
2021-04-06 10:35:33: Starting HiveServer2
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/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]
Hive Session ID = 280590e1-f7e6-4c3d-9311-0ee5f4c53c0c
|
```

Connect the Hive beeline using the following command.

```
$ beeline -u jdbc:hive2://hadoop0:10000
```

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/StaticLoggerBinder.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): CREATE TABLE pokes (foo INT, bar STRING)
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retryal = false)
INFO : Returning Hive schema: Schema(fieldSchemas:null, properties:null)
INFO : Completed compiling command(queryId=root_20210407030650_2eb03293-87f7-488d-b286-1e8f72c7d048); 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): CREATE 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 (retryal = 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 path

```
#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' OVERWRITE INTO TABLE pokes;
INFO : Compiling command(queryId=root_20210407033041_0d173e78-25a7-44af-8a66-a865ef4f2800): LOAD 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-a865ef4f2800); 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): LOAD 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-a865ef4f2800); 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 YARN cluster.

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

Hive:

<https://data-flair.training/blogs/configure-hive-metastore-to-mysql/>

## Erarata

Hadoop Version Mismatch with Hive Libraries.

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

```
[root@hadoop0 bin]# schematool -initSchema -dbType derby
SLF4J: Class path contains multiple SLF4J bindings
SLF4J: Found binding in [jar:file:/opt/hive/lib/slf4j-log4j12-1.7.25.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/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.checkNotNull(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)
```

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:

```
[root@hadoop0 /]# ls /opt/hive/lib/gua*
/opt/hive/lib/guava-19.0.jar
[root@hadoop0 /]# ls /opt/hadoop/share/hadoop/common/lib/gua*
/opt/hadoop/share/hadoop/common/lib/guava-27.0-jre.jar
[root@hadoop0 /]#
```

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.