INSTALLING HADOOP

INSTALL HADOOP FROM THE APACHE PROJECT

Become the hadoop user

```
# su – hadoop
```

Download Hadoop 2.7.1 using wget

```
# wget http://mirrors.advancedhosters.com/apache/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz
```

Uncompress and untar the downloaded file

```
# tar xzf hadoop-2.7.1.tar.gz
```

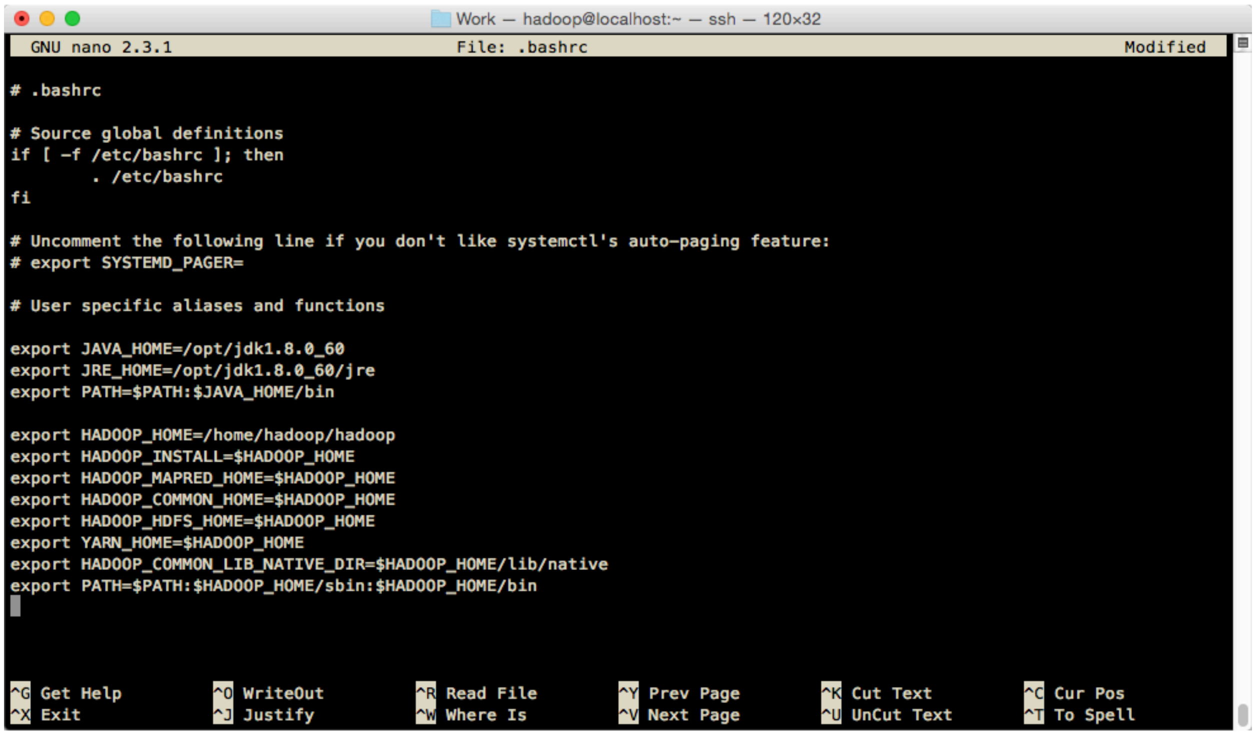
And change the name of the directory containing the Hadoop files to hadoop

```
# mv hadoop-2.7.1 hadoop
```

Set the following environment variables i.e., add them to the hadoop users .bashrc file using nano.

```
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

The figure below shows the nano editor with the .bashrc file after inclusion of these environment variables.



```
  GNU nano 2.3.1                              File: .bashrc                                                   Modified

# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions

export JAVA_HOME=/opt/jdk1.8.0_60
export JRE_HOME=/opt/jdk1.8.0_60/jre
export PATH=$PATH:$JAVA_HOME/bin

export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin



^G Get Help        ^O WriteOut        ^R Read File       ^Y Prev Page       ^K Cut Text        ^C Cur Pos
^X Exit            ^J Justify         ^W Where Is        ^V Next Page       ^U UnCut Text      ^T To Spell
```

Now “source” .bashrc so that the environment variables are set for your current login session.

```
# source .bashrc
```

Below is what your screen might look like when typing the contents of you .bashrc file to the screen and then “sourcing” the contents.

```
[hadoop@localhost ~]$ more .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
      . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-
paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions

export JAVA_HOME=/opt/jdk1.8.0_60
export JRE_HOME=/opt/jdk1.8.0_60/jre
export PATH=$PATH:$JAVA_HOME/bin

export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

[hadoop@localhost ~]$ source .bashrc
[hadoop@localhost ~]$
```

Next navigate to the directory $HADOOP_HOME/etc/hadoop/

```
# cd $HADOOP_HOME/etc/hadoop
```

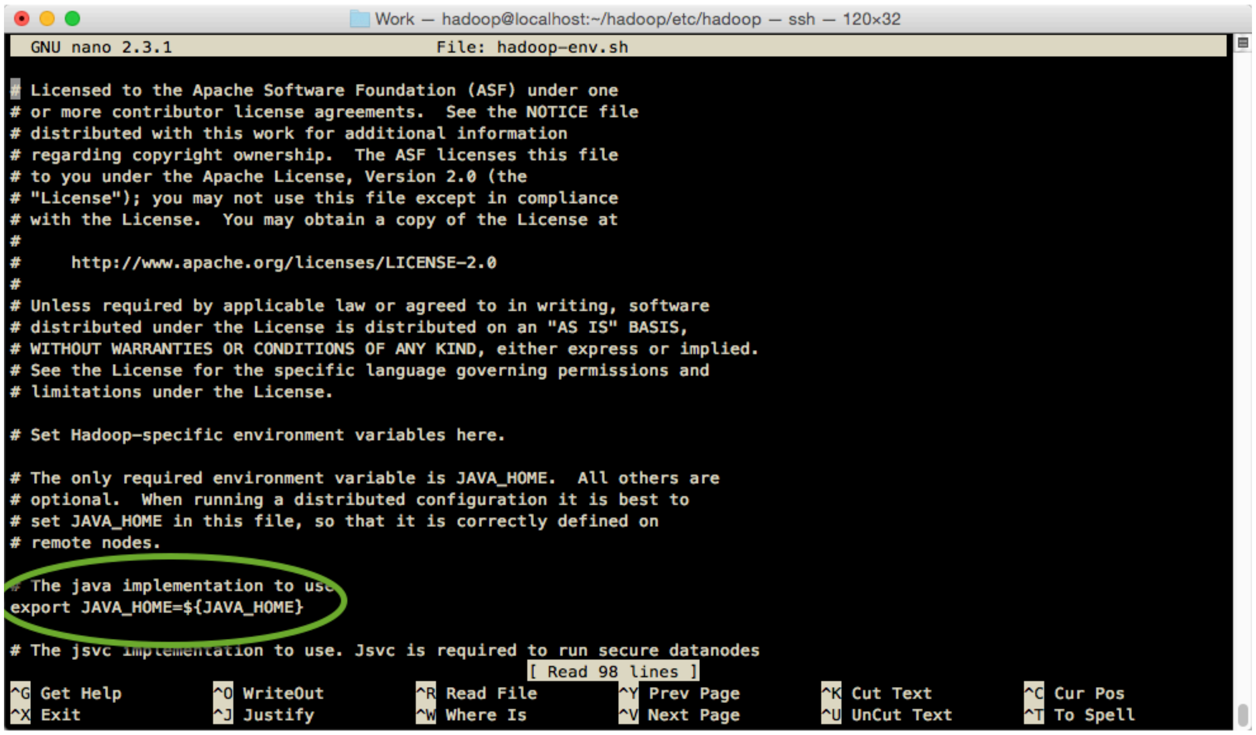
If you list the contents of the directory (enter `ls -la or ll,` or whatever your favorite command is to list the contents of a directory) you will see various configuration files for Hadoop.

Set JAVA_HOME in hadoop-env.sh

```
export JAVA_HOME=/opt/jdk1.8.0_60/
```

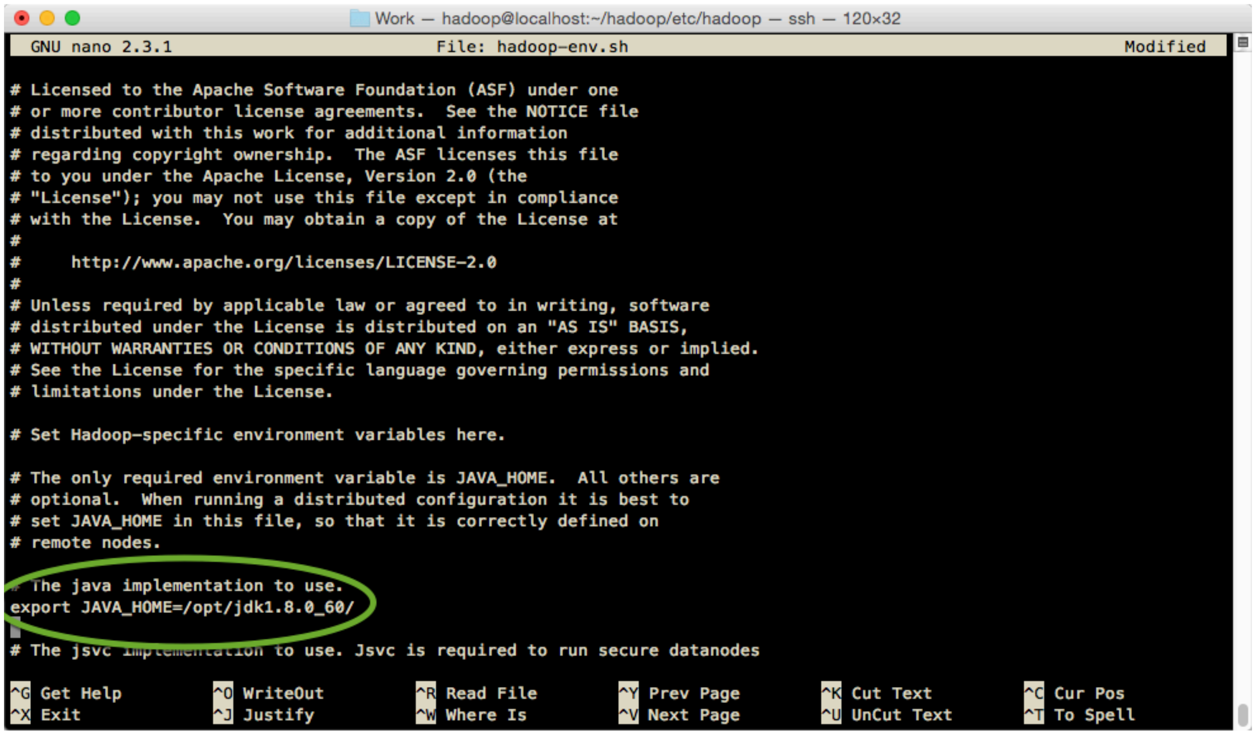
Use the nano editor to update the hadoop-env.sh file

See the line highlighted in the figure below



```
Work — hadoop@localhost:~/hadoop/etc/hadoop — ssh — 120×32
  GNU nano 2.3.1                           File: hadoop-env.sh

# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}

# The jsvc implementation to use. Jsvc is required to run secure datanodes
                                                    [ Read 98 lines ]
^G Get Help         ^O WriteOut         ^R Read File        ^Y Prev Page        ^K Cut Text         ^C Cur Pos
^X Exit             ^J Justify          ^W Where Is         ^V Next Page        ^U UnCut Text       ^T To Spell
```

And change it to what you see highlighted in the figure below



```
Work — hadoop@localhost:~/hadoop/etc/hadoop — ssh — 120×32
  GNU nano 2.3.1                           File: hadoop-env.sh                                        Modified

# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/opt/jdk1.8.0_60/

# The jsvc implementation to use. Jsvc is required to run secure datanodes

^G Get Help         ^O WriteOut         ^R Read File        ^Y Prev Page        ^K Cut Text         ^C Cur Pos
^X Exit             ^J Justify          ^W Where Is         ^V Next Page        ^U UnCut Text       ^T To Spell
```

EDIT THE HADOOP CONFIGURATION FILES

The Hadoop configuration files are located in the directory $HADOOP_HOME/etc/hadoop which is the sample place where hadoop-env.sh is. You will need to edit four more files xml files. Once these changes have been made you will format the hdfs namenode.

If necessary, change to the $HADOOP_HOME/etc/hadoop directory

```
# cd $HADOOP_HOME/etc/hadoop
```

Edit core-site.xml so that from <configuration> to </configuration> contains

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Edit hdfs-site.xml so that from <configuration> to </configuration> contains

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>

<property>
  <name>dfs.name.dir</name>
  <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
</property>

<property>
  <name>dfs.data.dir</name>
  <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
</property>
</configuration>
```

Edit mapred-site.xml.template

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

Edit yarn-site.xml

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

Now format the namenode

```
# hdfs namenode –format
```

Some of the output will look like

Adsf

Adsfadsf

```
hadoop@localhost hadoop]$ hdfs namenode -format
15/07/21 08:24:36 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.1
STARTUP_MSG:   classpath =

…
…

15/07/21 08:24:38 INFO util.GSet: Computing capacity for map
NameNodeRetryCache
15/07/21 08:24:38 INFO util.GSet: VM type       = 64-bit
15/07/21 08:24:38 INFO util.GSet: 0.029999999329447746% max memory
966.7 MB = 297.0 KB
15/07/21 08:24:38 INFO util.GSet: capacity      = 2^15 = 32768
entries
15/07/21 08:24:38 INFO namenode.FSImage: Allocated new BlockPoolId:
BP-2094905875-127.0.0.1-1437488678644
15/07/21 08:24:38 INFO common.Storage: Storage directory
/home/hadoop/hadoopdata/hdfs/namenode has been successfully
formatted.
15/07/21 08:24:39 INFO namenode.NNStorageRetentionManager: Going to
retain 1 images with txid >= 0
15/07/21 08:24:39 INFO util.ExitUtil: Exiting with status 0
15/07/21 08:24:39 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/127.0.0.1
************************************************************/
[hadoop@localhost hadoop]$
```

START HDFS AND YARN

Run these commands as the hadoop user

Go to the hadoop/sbin directory

```
# cd /home/hadoop/hadoop/sbin
```

Note the “shell script” files that start and stop the daemons associated with Hadoop by listing the contents of the directory.

Start HDFS

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

You should see something like

```
[hadoop@localhost sbin]$ start-dfs.sh
15/08/20 01:26:13 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to
/home/hadoop/hadoop/logs/hadoop-hadoop-namenode-
localhost.localdomain.out
localhost: starting datanode, logging to
/home/hadoop/hadoop/logs/hadoop-hadoop-datanode-
localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is
38:29:c7:bd:bd:4a:3d:c1:40:c3:da:8d:97:eb:3b:49.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of
known hosts.
0.0.0.0: starting secondarynamenode, logging to
/home/hadoop/hadoop/logs/hadoop-hadoop-secondarynamenode-
localhost.localdomain.out
15/08/20 01:26:37 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
[hadoop@localhost sbin]$
```

Next start YARN

```
# start-yarn.sh
```

You should see something like

```
hadoop@localhost sbin]$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/hadoop/hadoop/logs/yarn-
hadoop-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to
/home/hadoop/hadoop/logs/yarn-hadoop-nodemanager-
localhost.localdomain.out
[hadoop@localhost sbin]$
```

Next start the Job History Server

```
      # mr-jobhistory-daemon.sh start historyserver
```

You should see

```
[hadoop@localhost sbin]$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/hadoop/hadoop/logs/mapred-
hadoop-historyserver-localhost.localdomain.out
[hadoop@localhost sbin]$
```

The "jps" command, which is part of the java software distribution, will list the running java process. You can use it to verify that the hadoop processes are running.

```
[hadoop@localhost sbin]$ jps
26880 JobHistoryServer
26960 Jps
26098 DataNode
25975 NameNode
26264 SecondaryNameNode
26424 ResourceManager
26525 NodeManager
[hadoop@localhost sbin]$
```

The DataNode, NameNode, and SecondaryNameNode are the processes associated with HDFS. ResourceManager and NodeManager are YARN processes. JobHistoryServer is associated with the Job History Server, obviously.

Note: that the order the processes are displayed and the 5 digit pids (process id) may differ on your system.

To stop the three services, use these commands

```
      # mr-jobhistory-daemon.sh stop historyserver
      # stop-yarn.sh
      # stop-dfs.sh
```

Firewall concerns for Web interfaces to Hadoop

Other than the Linux command line, the Hadoop software provides a Web interface to its software systems. Each of these software services is exposed to the web on different network ports. However, out of the box CentOS blocks all ports with the exception of port 22, which is used for secure shell, ssh, so that you can remotely access the machine.

To gain web access to the Hadoop services the following ports need to be opened.

1. Port 50070 for HDFS
2. Port 50090 for the HDFS Secondary Namenode
3. Port 50075 for DataNode
4. Port 8088 for YARN
5. Port 19888 for the Job History Server

In distributions of CentOS before 7, ipconfig was used to manage the firewall for opening network ports. However, with CentOS 7 one uses firewall-cmd.

Using the firewall-cmd you can open ports for the current session, which will revert to the original settings when the OS is rebooted. You can also make the changes permanent, but the changes will only take effect after the system is rebooted.

You must be the root user to affect changes to the firewall.

The five commands (one for each port) that you need to execute to open the ports for the current session are

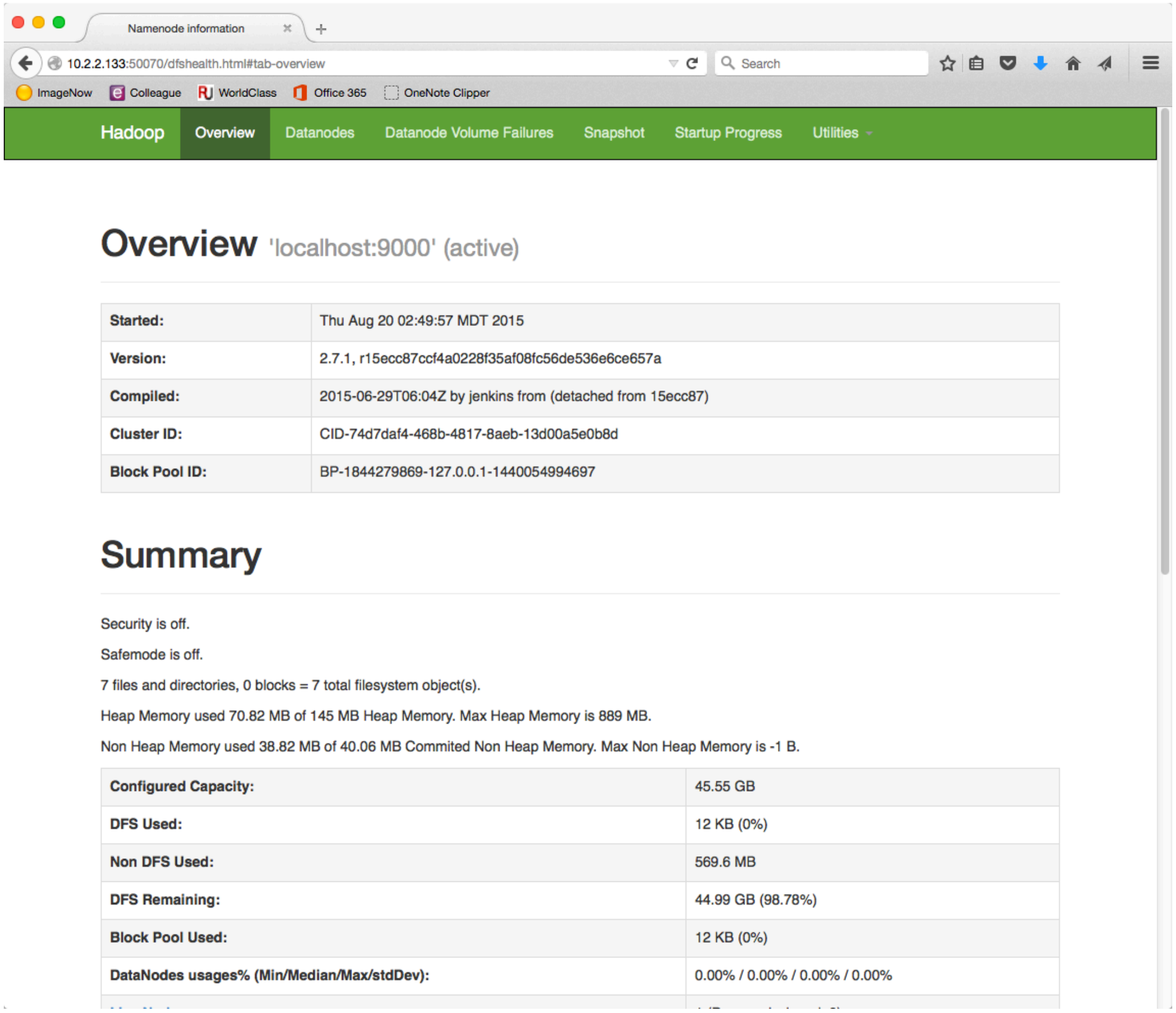
```
# firewall-cmd --zone=public --add-port=50070/tcp
# firewall-cmd --zone=public --add-port=50090/tcp
# firewall-cmd --zone=public --add-port=50075/tcp
# firewall-cmd --zone=public --add-port=8088/tcp
# firewall-cmd --zone=public --add-port=19888/tcp
```

Using these five commands will make the changes permanent

```
# firewall-cmd --zone=public --permanent --add-port=50070/tcp
# firewall-cmd --zone=public --permanent --add-port=50090/tcp
# firewall-cmd --zone=public --permanent --add-port=50075/tcp
# firewall-cmd --zone=public --permanent --add-port=8088/tcp
# firewall-cmd --zone=public --permanent --add-port=19888/tcp
```

If needed, start the Hadoop services

Once you have opened the ports you can access the web interfaces to your server using the web browser on you host computer. For example, if I enter 10.2.2.133:50070 as the address in Firefox I see the figure below. This is the web interface to HDFS.



Overview 'localhost:9000' (active)

Started:	Thu Aug 20 02:49:57 MDT 2015
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	CID-74d7daf4-468b-4817-8aeb-13d00a5e0b8d
Block Pool ID:	BP-1844279869-127.0.0.1-1440054994697

Summary

Security is off.

Safemode is off.

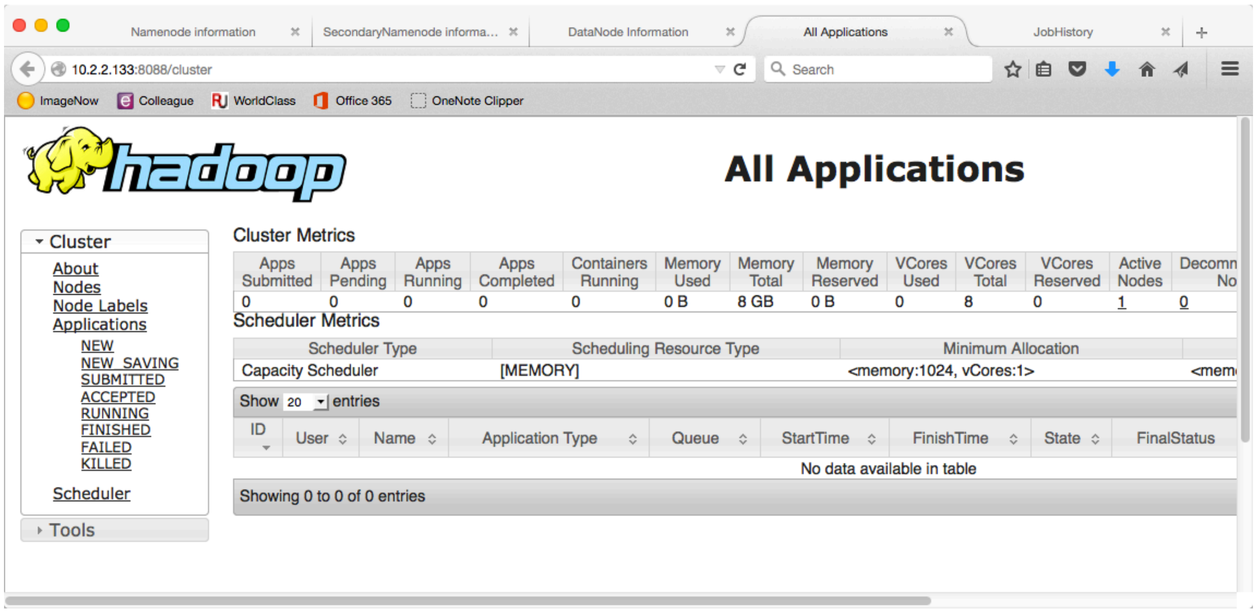
7 files and directories, 0 blocks = 7 total filesystem object(s).

Heap Memory used 70.82 MB of 145 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 38.82 MB of 40.06 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	45.55 GB
DFS Used:	12 KB (0%)
Non DFS Used:	569.6 MB
DFS Remaining:	44.99 GB (98.78%)
Block Pool Used:	12 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%

For YARN, the web interface is at 10.2.2.133:8088 and looks like



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decomn No
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<mem

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
No data available in table								

Showing 0 to 0 of 0 entries

HDFS, YARN, and Job History are probably the most useful interfaces.

TESTING YOUR HADOOP INSTALLATION

The following commands can be used to build a

1. An HDFS user directory for the hadoop user
2. Download the complete works of Shakespeare as a text file
3. Create an input directory and copy the Shakespeare.txt file to it
4. Run the hadoop wordcount example program on the HDFS input file
5. Type the results to the console
6. More the output from HDFS to the local file system
7. Type the output file to the console

As you use the hadoop fs command you will see a warning that looks like

```
WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
```

The processes will run correctly in spite of this warning.

Bonus points go to anyone who can come up with how to correct this concern. Fewer bonus point go to anyone who can figure out how to suppress the warning. BTW, there are no bonus points! ☺

```
# cd ~
# hadoop fs -ls
# hadoop fs -mkdir -p /user/hadoop
# hadoop fs -ls
# wget http://norvig.com/ngrams/shakespeare.txt
# hadoop fs -mkdir shakespeare
# hadoop fs -mkdir shakespeare/input
# hadoop fs -copyFromLocal ~/shakespeare.txt shakespeare/input
# hadoop fs -ls shakespeare/input
# hadoop jar hadoop-mapreduce-examples-2.7.1.jar wordcount
shakespeare/input                                shakespeare/output
# hadoop fs -ls shakespeare/output
# hadoop fs -cat shakespeare/output/part-r-00000
# hadoop fs -copyToLocal shakespeare/output/part-r-00000
~/shakespeareoutput.txt
      # more shakespeareoutput.txt
```