Installing Hadoop 2.7.3 / Yarn, Hive 2.1.0, Scala 2.11.8, and Spark 2.0 on Raspberry Pi Cluster of 3 Nodes

By: Nicholas Propes 2016

NOTES

Please follow instructions PARTS in order because the results are cumulative (i.e. PART I, then PART II, then PART III, then PART IV, then PART V). PARTS III, IV and V are optional.

1. A lot of tools here are for Windows, so substitute your OS equivalent.

2. I am using 3 Raspberry Pi model 3.0 with a 8-port switch. They each have a 32 GB micro sd card (you have to buy this separately) and a case (also bought separately). They also each come with 1 GB RAM (not upgradable). They also have wireless capability built-in, so you may try it without the 8-port switch, but I'm choosing wired.

3. I might forget to put "sudo" in front of commands, so if you get permission errors try putting a "sudo" in front of the command.

4. I attached my switch to my router that was connected to the Internet. I attached each raspberry pi separately, one-by-one, to the switch as I configured it. I didn't connect all of them at once to avoid confusion of which DHCP provided IP belonged to which raspberry pi.

5. I am using precompiled binaries for Hadoop which is 32-bit. If you want to try to compile for 64-bit on the raspberry pi, you can compile Hadoop from source, but it takes a very long time and there are patches (e.g. https://issues.apache.org/jira/browse/HADOOP/fixforversion/12334005/?selectedTab=com.atlassian.jira.jira-projects-plugin:version-summary-panel). Make sure you have the correct versions of src to compile. When trying, I found I had to have this library compiled first-protobuf 2.5- http://pkgs.fedoraproject.org/repo/pkgs/protobuf/protobuf-2.5.0.tar.bz2/a72001a9067a4c2c4e0e836d0f92ece4/ from google code (do not try the newer version, you need 2.5 for Hadoop 2.7.3). You will have to ensure that you install the compilers and libraries you need such as openssl, maven, g++, etc. I kept finding new ones I needed as I tried to compile. My recommendation is not to do this. First, try to get comfortable with precompiled binaries and Hadoop configuration as in these instructions. Then once you get experience, go back and see if you can compile your own version for the raspberry pi platform. You will often see warning messages using the 32-bit precompiled binaries, "WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable" when executing commands. This is OK for my purposes.

6. If you get stuck, you might try these websites for reference though they seem to have errors:

   - http://scn.sap.com/community/bi-platform/blog/2015/04/25/a-hadoop-data-lab-project-on-raspberry-pi--part-14
   - http://scn.sap.com/community/bi-platform/blog/2015/05/03/a-haddop-data-lab-project-on-raspberry-pi--part-24
   - http://scn.sap.com/community/bi-platform/blog/2015/07/10/a-hadoop-data-lab-project-on-raspberry-pi--part-44
   - http://www.widriksson.com/raspberry-pi-hadoop-cluster/
   - http://spark.apache.org/docs/latest/spark-standalone.html

Part I: Basic Setup

1. Download the Raspbian Jessie OS disk image. (I didn't use the lite version, though you could try as this might save disk space--not sure if you will have to install java or other components that might be missing if you use the lite version)

2. Burn the disk image to a micro SD card using Win32 Disk Imager (Windows)

3. Put the micro SD card in the Raspberry Pi and start it up.

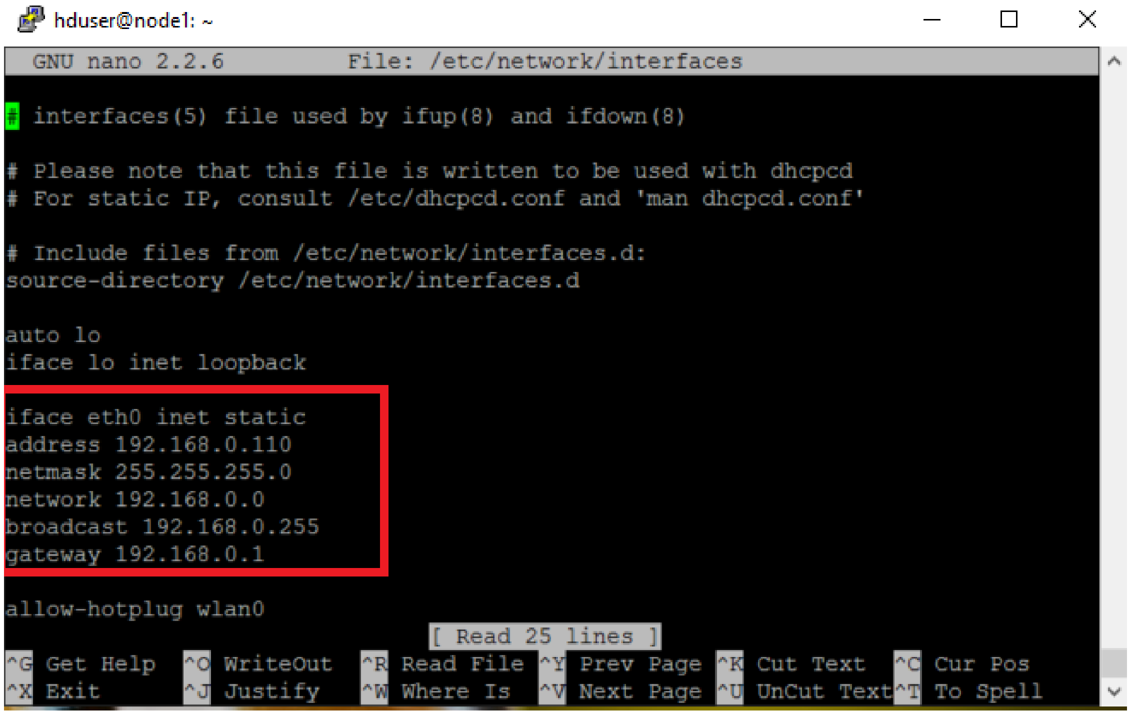
4. SSH into Raspberry Pi using Putty (have to find out what IP is given to it using an network scanning tool, I used one I put on my phone). Default username is "**pi**" and password is "**raspberry**"

5. Set up raspberry pi using the command "**sudo raspi-config**"
   - International Options->
   - Advanced Options -> Memory Split to 32MB
   - Advanced Options -> SSH -> Enable SHH
   - Advanced Options-> Hostname -> **node1** (or your preference of node name)
   - reboot and log back in using Putty

6. Type "**sudo apt-get update**"

7. Type "**sudo apt-get install python-dev**"

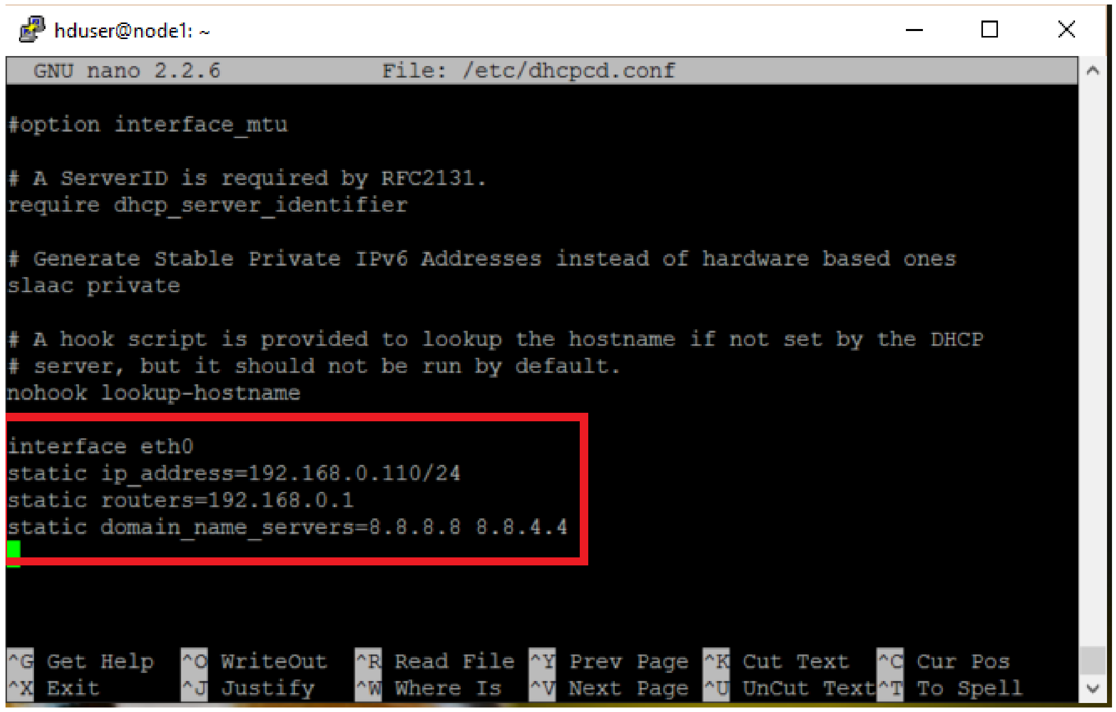
8. Set up network interface (note I'm setting the node1 to address 192.168.0.110):
   - Type "**ifconfig**" and note the inet addr, Bcast, and Mask of eth0
   - Type "**route -n**" and note Gateway and Destination (not 0.0.0.0 on either, the other one).
   - Type "**sudo nano /etc/network/interfaces**" and enter/edit the following:



```
hduser@node1: ~
  GNU nano 2.2.6           File: /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet static
address 192.168.0.110
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.1

allow-hotplug wlan0
                                    [ Read 25 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
```

(to save, press CTRL-X, and press y, and then hit enter, I won't repeat this for nano editing in future)

-Type **"sudo nano /etc/dhcpcd.conf"** and enter/edit the following at the bottom of the file:



```
  GNU nano 2.2.6              File: /etc/dhcpcd.conf

#option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# A hook script is provided to lookup the hostname if not set by the DHCP
# server, but it should not be run by default.
nohook lookup-hostname

interface eth0
static ip_address=192.168.0.110/24
static routers=192.168.0.1
static domain_name_servers=8.8.8.8 8.8.4.4


^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
```

-Type **"sudo nano /etc/hosts"** and delete everything then enter the following:

```
127.0.0.1          localhost
192.168.0.110      node1
```

Make sure that is all that is in that file and no other items exist such as ipv6, etc.

-Type **"sudo nano /etc/hostname"** and make sure it just says:

node1

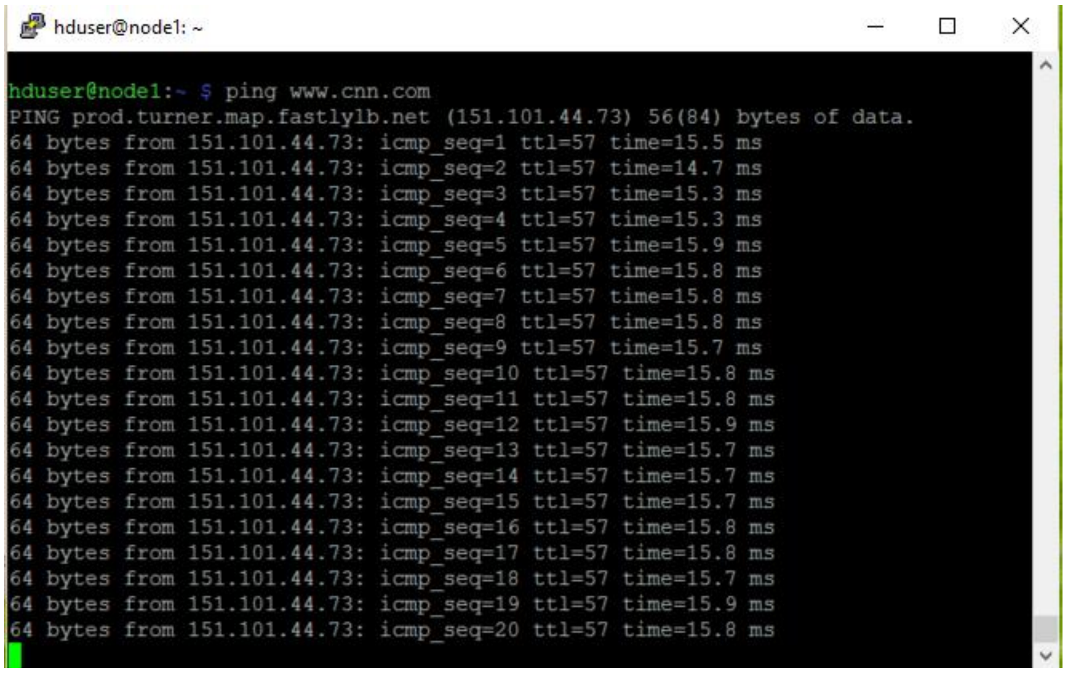
9. Type **"java -version"** and make sure you have the correct java version. I am using java version "1.8.0_64" i.e. Java 8. If you don't have the correct version, type **"sudo apt-get install oracle-java8-jdk"**. You might have multiple Java versions installed. You can use the command **"sudo update-alternatives --config java"** to select the correct one.

10. Now, we set up a group and user that will be used for Hadoop. We also make the user a superuser.
    -Type **"sudo addgroup hadoop"**
    -Type **"sudo adduser --ingroup hadoop hduser"**
    -Type **"sudo adduser hduser sudo"**

11. Next, we create a RSA key pair so that the master node can log into slave nodes through ssh without password. This will be used later when we have multiple slave nodes.
   -Type "**su hduser**"
   -Type "**mkdir ~/.ssh**"
   -Type "**ssh-keygen -t rsa -P ""**"
   -Type "**cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys**"
   -Verify by typing "**ssh localhost**"

12. Reboot the raspberry pi by typing "**sudo reboot**"

13. Login as hduser and make sure you can access the Internet (note your Putty now should use 192.168.0.110 to access the raspberry pi).
   -Type "**ping www.cnn.com**"
   -Press **CTRL-C** when finished.



```
hduser@node1:~ $ ping www.cnn.com
PING prod.turner.map.fastlylb.net (151.101.44.73) 56(84) bytes of data.
64 bytes from 151.101.44.73: icmp_seq=1 ttl=57 time=15.5 ms
64 bytes from 151.101.44.73: icmp_seq=2 ttl=57 time=14.7 ms
64 bytes from 151.101.44.73: icmp_seq=3 ttl=57 time=15.3 ms
64 bytes from 151.101.44.73: icmp_seq=4 ttl=57 time=15.3 ms
64 bytes from 151.101.44.73: icmp_seq=5 ttl=57 time=15.9 ms
64 bytes from 151.101.44.73: icmp_seq=6 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=7 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=8 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=9 ttl=57 time=15.7 ms
64 bytes from 151.101.44.73: icmp_seq=10 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=11 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=12 ttl=57 time=15.9 ms
64 bytes from 151.101.44.73: icmp_seq=13 ttl=57 time=15.7 ms
64 bytes from 151.101.44.73: icmp_seq=14 ttl=57 time=15.7 ms
64 bytes from 151.101.44.73: icmp_seq=15 ttl=57 time=15.7 ms
64 bytes from 151.101.44.73: icmp_seq=16 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=17 ttl=57 time=15.8 ms
64 bytes from 151.101.44.73: icmp_seq=18 ttl=57 time=15.7 ms
64 bytes from 151.101.44.73: icmp_seq=19 ttl=57 time=15.9 ms
64 bytes from 151.101.44.73: icmp_seq=20 ttl=57 time=15.8 ms
```

If you can't access the Internet something is wrong with your network setup (probably you aren't hooked up to a router, you misspelled something, or your Internet isn't working).

Part II: Hadoop 2.7.3 / Yarn Installation : Single Node Cluster

1. In Windows, go to the Apache Hadoop website: http://hadoop.apache.org/ and click on the "Releases" link on the left. You'll see the list of Hadoop releases for source and binary. I selected the binary tarball release for version 2.7.3 by clicking on the link. Now, you will see a lot of different links for downloading this binary. I wrote down (don't download) the link to one of them such as:

   http://apache.cs.utah.edu/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz

2. Login to the raspberry pi using Putty as hduser.

3. We'll be installing Hadoop / Yarn in the "/opt" directory.
   - -Type "**cd /opt**".

4. Download the binary for Hadoop.
   - -Type typing "**sudo wget <URL in step 1>**" e.g. "wget http://apache.cs.utah.edu/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz"

5. Unzip the tarball.
   - -Type "**sudo tar -xvzf hadoop-2.7.3.tar.gz**"

6. I renamed the download to something easier to type-out later.
   - -Type "**sudo mv hadoop-2.7.3 hadoop**"

7. Make this hduser an owner of this directory just to be sure.
   - -Type "**sudo chown -R hduser:hadoop hadoop**"

8. Now that we have hadoop, we have to configure it before it can launch its daemons (i.e. namenode, secondary namenode, datanode, resourcemanager, and nodemanager). Make sure you are logged in as hduser.
   - -Type "**su hduser**"

9. Now, we will add some environmental variables.
   - -Type "**cd ~**"
   - -Type "**sudo nano .bashrc**"
   - -At the bottom of the .bashrc file, add the following lines

   ```
   export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
   export HADOOP_INSTALL=/opt/hadoop
   export PATH=$PATH:$HADOOP_INSTALL/bin:$HADOOP_INSTALL/sbin
   export HADOOP_USER_CLASSPATH_FIRST=true
   ```

10. Many configuration files for Hadoop and its daemons are located in the /opts/hadoop/etc/hadoop folder. We will edit some of these files for configuration purposes. Note, there are a lot of configuration parameters to explore.
    - -Type "**cd /opts/hadoop/etc/hadoop**"
    - -Type "**sudo nano hadoop-env.sh**"
    - -Edit the line (there should be place to edit an existing line)

      ```
      export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
      ```
    - -Edit the line (there should be place to edit an existing line)

```
export HADOOP_HEAPSIZE=250
```

The default is 1000 MB of heap per daemon launched by HADOOP, but we are dealing with limited memory Raspberry Pi (1GB).

11. Many configuration files for Hadoop and its daemons are located in the /opts/hadoop/etc/hadoop folder. We will edit some of these files for configuration purposes. Note, there are a lot of configuration parameters to explore. Now we will edit the core Hadoop configuration in core-site.xml.
    -Type **"sudo nano core-site.xml"**
    -Add the following properties between the <configuration> and </configuration> tags.

```
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/hdfs/tmp</value>
    </property>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://node1:54310</value>
    </property>
</configuration>
```

12. Now edit the hdfs (hadoop file system) configuration in hdfs-site.xml.
    -Type **"sudo nano hdfs-site.xml"**
    -Add the following properties between the <configuration> and </configuration> tags.

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

    We'll be setting this to a different value once we have multiple nodes.

13. Now edit the YARN (Yet Another Resource Negotiator) configuration in yarn-site.xml.
    -Type **"sudo nano hdfs-site.xml"**
    -Add the following properties between the <configuration> and </configuration> tags.

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>node1</value>
    </property>
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>1024</value>
    </property>
    <property>
        <name>yarn.nodemanager.resource.cpu-vcores</name>
        <value>4</value>
    </property>
```

```
</configuration>
```

14. Now edit the map-reduce configuration in mapred-site.xml. Here I believe the default framework for map-reduce is YARN, but I do this anyway (may be optional).
- Type "**sudo cp mapred-site.xml.template mapred-site.xml**"
- Type "**sudo nano mapred-site.xml**"
-Add the following properties between the <configuration> and </configuration> tags.

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

15. Now edit the masters and slaves files.
-Type "**sudo nano slaves**"
-Edit the file so it only contains the following:
node1
-Type "**sudo nano masters**"
-Edit the file so it only contains the following:
node1

16. Reboot by typing "**sudo reboot**"

17. Login as hduser.

18. Create a location for hdfs (see core-site.xml) and format hdfs.
-Type "**sudo mkdir -p /hdfs/tmp**"
-Type "**sudo chown hduser:hadoop /hdfs/tmp**"
-Type "**sudo chmod 750 /hdfs/tmp**"
-Type "**hadoop namenode -format**"

19. Start Hadoop (hdfs) and YARN (resource scheduler). Ignore any warning messages that may occur (as mentioned in notes, most are due to 32-bit binary running on 64-bit platform)
-Type "**cd ~**"
-Type "**start-dfs.sh**"
-Type "**start-yarn.sh**"

20. Test Hadoop and YARN (see if daemons are running)
-Type "**jps**"

You should see something like this:

```
5021 DataNode
4321 NameNode
2012 Jps
1023 SecondaryNameNode
23891 Nodemanager
3211 ResourceManager
```

If you don't see DataNode, SecondaryNameNode, and NameNode, probably something is setup wrong in .bashrc, core-site.xml, or hdfs-site.xml.

If you don't see ResourceManager and Nodemanager, probably something is incorrectly setup in .bashrc, yarn-site.xml, or mapred-site.xml.

21. You can test a calculation using examples provided in the distribution. Here we put a local file into hdfs. Then we execute a Java program that counts the frequency of words in the file located on hdfs now. Then we grab the output file from hdfs and put it on the local computer.
    -Type "**hadoop fs -copyFromLocal /opt/hadoop/LICENSE.txt /license.txt**"
    -Type "**hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount /license.txt /license-out.txt**"
    -Type "**hadoop fs -copyToLocal /license-out.txt**"
    -Type "**more ~/license-out.txt/part-r-00000**"

    Here you can see the output that counts the frequency of words in the LICENSE.txt file.

22. You can view the setup in your Windows browser by following these URLs.

    NAMENODE INFORMATION
    http://192.168.0.110:50070

    ALL APPLICATIONS (YARN)
    http://192.168.0.110:8088

23. There are a lot of commands to explore (there are also hdfs commands which I believe are considered more modern than hadoop commands, but not sure yet). Here are a few to try out:

    - **"hadoop fs -ls /"** shows contents of hdfs
    - **"hadoop fs -rm <file>"** deletes file
    - **"hadoop fs -rm -r -f <directory>"** deletes directory and contents
    - **"hadoop fs -copyFromLocal <local source file> <hdfs destination file>"** copies file from local file system to hdfs.
    - **"hadoop fs -copyToLocal <hdfs source file> <local destination file>"** copies file from hdfs to local file system.
    - **"start-dfs.sh"** starts hdfs daemon (NameNode, Datanode, Secondary NameNode)
    - **"start-yarn.sh"** starts yarn daemon (ResourceManager, NodeManager)
    - **"stop-dfs.sh"** stops hdfs daemon (NameNode, Datanode, Secondary NameNode)
    - **"stop-yarn.sh"** stops yarn daemon (ResourceManager, NodeManager)

Part III: Hadoop 2.7.3 / Yarn Installation : Multi-Node Cluster

1. On node1, login as hduser.

2. Here we will setup a multi-node cluster following on Parts I and II setup. Each node will have Hadoop / Yarn installed on it because we will be cloning node1.
   -Type "**sudo nano /etc/hosts**" and edit it with the following:

   ```
   127.0.0.1        localhost
   192.168.0.110    node1
   192.168.0.111    node2
   192.168.0.112    node3
   ```

   Make sure that is all that is in that file and no other items exist such as ipv6, etc.

3. Remove any data in the /hdfs/tmp folder.
   -Type "**sudo rm -f /hdfs/tmp/***"

4. Shutdown the raspberry pi.
   -Type "**sudo shutdown -h now**"

5. Now we will clone the single node we created onto 2 other SD cards for the other two raspberry pis. Then we will change the configuration for each to setup the cluster. Node 1 will be the master node. Nodes 2 and 3 will be the slave nodes.

6. We will now copy the node1 32 GB micro SD card to the other two blank SD cards.
   -Unplug the raspberry pi from power.
   -Remove the SD card from the raspberry pi.
   -Using a micro SD card reader and Win 32 Disk Imager, "READ" the SD card to an .img file on your Windows computer (you can choose any name for your .img file like node1.img). <u>Warning: this file will be approximately 32 GB so have room where you want to create the image on your Windows computer.</u>
   -After the image is created, put your node1 micro SD card back into the original raspberry pi. Get your other two blank micro SD cards for the other two raspberry pis and "WRITE" the node1 image you just created to them one at a time.
   -After making the images, put the micro SD cards back to their respective raspberry pis and set them aside for now.

7. Now plug in the raspberry pi you want for **node2** to the network and power it up. (it should be the only one attached to the network switch). Login to it using hduser using Putty.

8. Set up network interface for node2 (its ip address will be 192.168.0.111)
   -Type "**sudo nano /etc/network/interfaces**" and change the address from 192.168.0.110 to 192.168.0.111.
   -Type "**sudo nano /etc/dhcpcd.conf**" and change the static ip_address from 192.168.0.110/24 to 192.168.0.111/24.
   -Type "**sudo nano /etc/hostname**" and change the name from node1 to node2.

-Type "**sudo reboot**"

9. Now plug in the raspberry pi you want for **node3** to the network and power it up. (node2 and node3 should be the only one attached to the network switch). Login to it using hduser using Putty.

10. Set up network interface for node3 (its ip address will be 192.168.0.112)
    -Type "**sudo nano /etc/network/interfaces**" and change the address from 192.168.0.110 to 192.168.0.112.
    -Type "**sudo nano /etc/dhcpcd.conf**" and change the static ip_address from 192.168.0.110/24 to 192.168.0.112/24.
    -Type "**sudo nano /etc/hostname**" and change the name from node1 to node3.
    -Type "**sudo reboot**"

11. Now attach **node1** to the network switch and power it up. Login to **node1** (192.168.0.110) using Putty as hduser. You should now see 192.168.0.110, 192.168.0.111, and 192.168.0.112 on your network.

12. Now edit the hdfs configuration in hdfs-site.xml for **node1**.
    -Type "**cd /opt/hadoop/etc/hadoop**"
    -Type "**sudo nano hdfs-site.xml**"
    -Edit the value to 3 for property dfs.replication.

```
<configuration>
      <property>
            <name>dfs.replication</name>
            <value>3</value>
      </property>
</configuration>
```

    -Type "**sudo nano slaves**"
    -Edit the file so it only contains the following:

```
      node1
      node2
      node3
```

    -Type "**sudo nano masters**"
    -Edit the file so it only contains the following:

```
      node1
```

13. Copy the RSA keys over to nodes 2 and 3.
    -Type "**sudo ssh-copy-id -i ~/.ssh/id_rsa.pub hduser@node2**"
    -Type "**sudo ssh-copy-id -i ~/.ssh/id_rsa.pub hduser@node3**"

14. Login to **node2** (192.168.0.111) using Putty as hduser.

15. Now edit the hdfs configuration in hdfs-site.xml for **node2**.
    -Type "**cd /opt/hadoop/etc/hadoop**"
    -Type "**sudo nano hdfs-site.xml**"
    -Edit the value to 3 for property dfs.replication.

```
<configuration>
      <property>
            <name>dfs.replication</name>
            <value>3</value>
      </property>
</configuration>
```

    -Type "**sudo nano slaves**"
    -Edit the file so it only contains the following:

      node2

   -Type "**sudo nano masters**"

   -Edit the file so it only contains the following:

      node1

   -Type "**sudo reboot**"

16. Login to **node3** (192.168.0.112) using Putty as hduser.

17. Now edit the hdfs configuration in hdfs-site.xml for **node3**.

   -Type "**cd /opt/hadoop/etc/hadoop**"

   -Type "**sudo nano hdfs-site.xml**"

   -Edit the value to 3 for property dfs.replication.

```
<configuration>
      <property>
            <name>dfs.replication</name>
            <value>3</value>
      </property>
</configuration>
```

   -Type "**sudo nano slaves**"

   -Edit the file so it only contains the following:

      node3

   -Type "**sudo nano masters**"

   -Edit the file so it only contains the following:

      node1

   -Type "**sudo reboot**"

18. Login to **node1** (192.168.0.110) using Putty as hduser.

   -Type "**cd ~**"

   -Type "**hadoop namenode -format**"

   -Type "**sudo reboot**"

19. Login to **node1** (192.168.0.110) using Putty as hduser.

   -Type "**start-dfs.sh**"

   -Type "**start-yarn.sh**"

20. Test Hadoop and YARN (see if daemons are running)

   -Type "**jps**"

   You should see something like this:

```
5021 DataNode
4321 NameNode
2012 Jps
1023 SecondaryNameNode
23891 Nodemanager
3211 ResourceManager
```

21. Login to **node2** (192.168.0.111) using Putty as hduser.

22. Test Hadoop and YARN (see if daemons are running)

   -Type "**jps**"

   You should see something like this:

```
5021 DataNode
2012 Jps
23891 Nodemanager
```

23. Login to **node3** (192.168.0.112) using Putty as hduser.

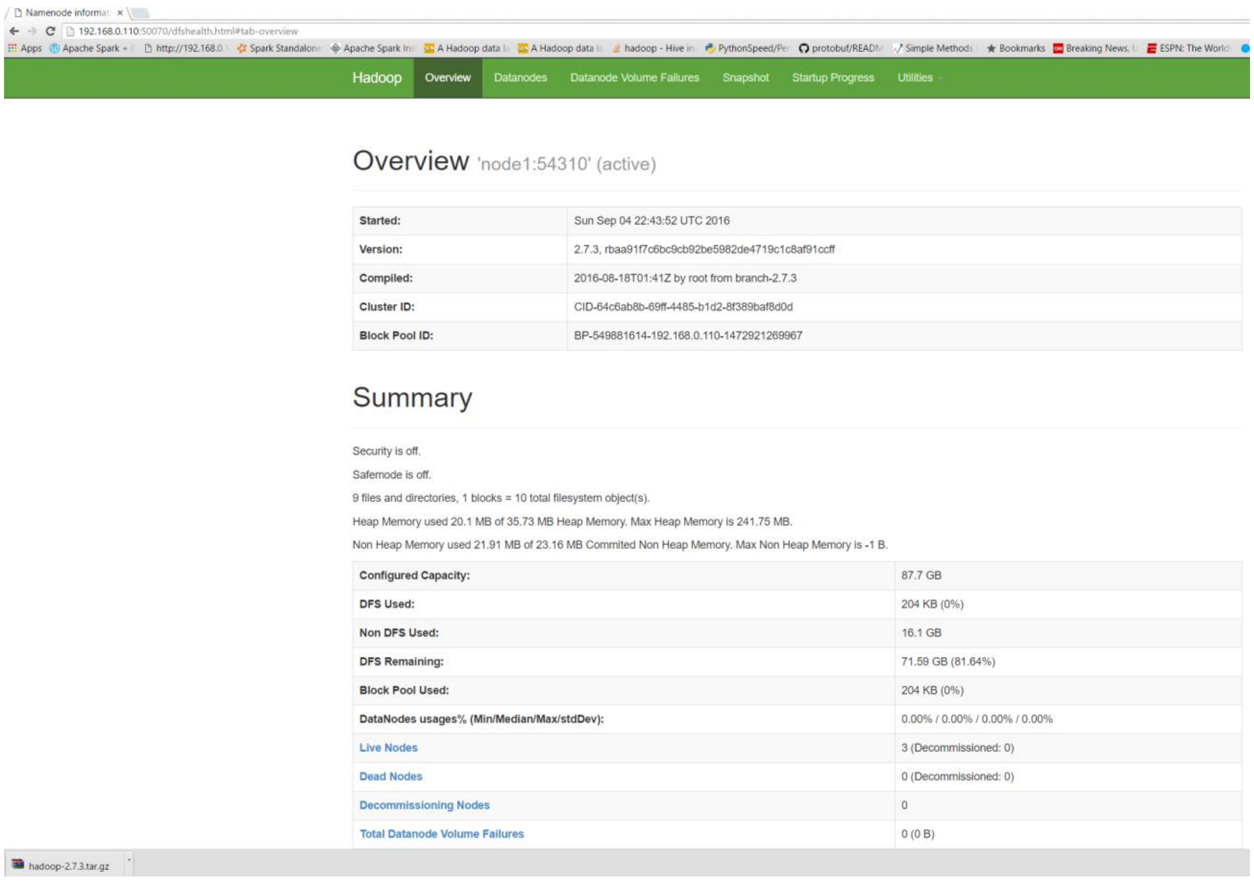
24. Test Hadoop and YARN (see if daemons are running)
    -Type "**jps**"

    You should see something like this:
    5021 DataNode
    2012 Jps
    23891 Nodemanager

25. You can view the setup in your Windows browser by following these URLs.

    NAMENODE INFORMATION
    http://192.168.0.110:50070

    ALL APPLICATIONS (YARN**)**
    http://192.168.0.110:8088



Overview 'node1:54310' (active)

Started:	Sun Sep 04 22:43:52 UTC 2016
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-64c6ab8b-69ff-4485-b1d2-8f389baf8d0d
Block Pool ID:	BP-549881614-192.168.0.110-1472921269967

Summary

Security is off.

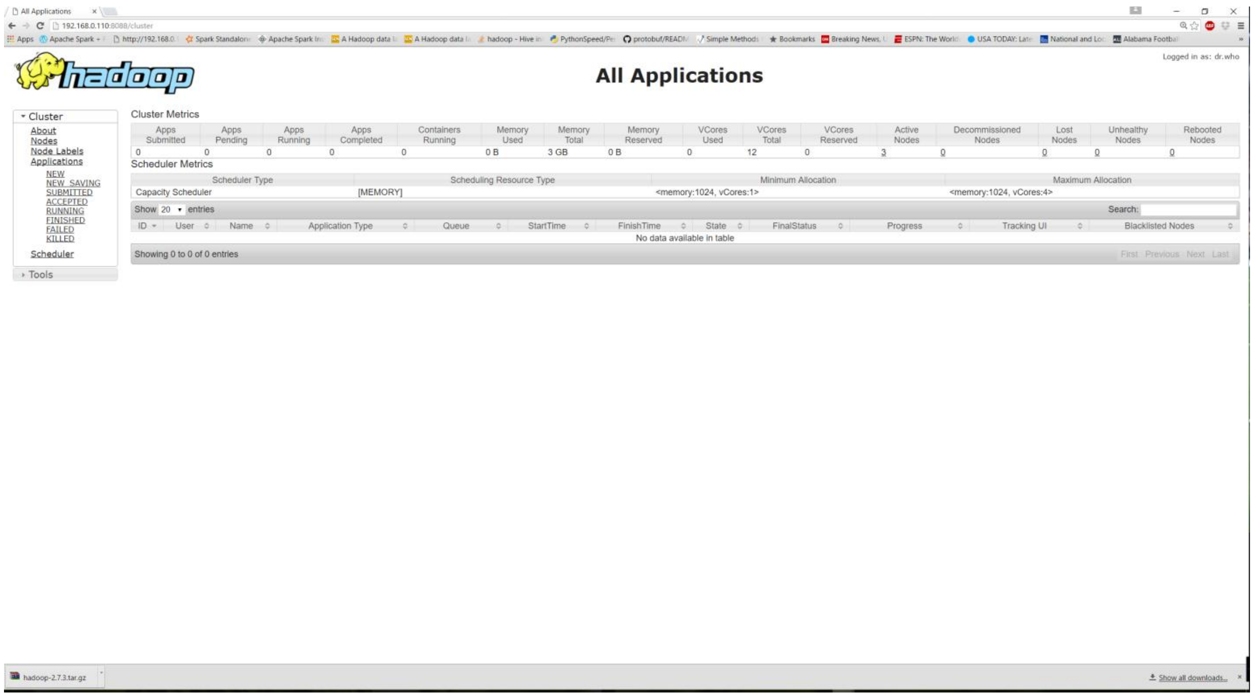
Safemode is off.

9 files and directories, 1 blocks = 10 total filesystem object(s).

Heap Memory used 20.1 MB of 35.73 MB Heap Memory. Max Heap Memory is 241.75 MB.

Non Heap Memory used 21.91 MB of 23.16 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	87.7 GB
DFS Used:	204 KB (0%)
Non DFS Used:	16.1 GB
DFS Remaining:	71.59 GB (81.64%)
Block Pool Used:	204 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)



All Applications

192.168.0.110:8088/cluster

Logged in as: dr.who

All Applications

- 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	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	3 GB	0 B	0	12	0	3	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:1024, vCores:4>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
No data available in table											

Showing 0 to 0 of 0 entries First Previous Next Last

hadoop-2.7.3.tar.gz Show all downloads...

Part IV: Hive 2.1.0 Installation

1. Here we will install Hive on node1. Hive only needs to be installed on the master node.

2. On node1 (192.168.0.110), login as hduser.

3. In your Windows computer, open up a web browser and go to: https://hive.apache.org/downloads.html and click on "Download a release now!" You will see a list of links to download Hive. Click on one of the links. Then click on "hive-2.1.0". Write the link of the bin version down (do not download - e.g. http://apache.claz.org/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz)

4. On node1, we will now download Hive into the /opt directory.
   - -Type "**cd /opt**"
   - -Type "**sudo wget <URL from step 3>**"
   - -Type "**sudo tar -xzvf apache-hive-2.1.0-bin.tar.gz**"
   - -Type "**sudo mv apache-hive-2.1.0-bin hive-2.1.0**"
   - -Type "**sudo chown -R hduser:hadoop /opt/hive-2.1.0**"

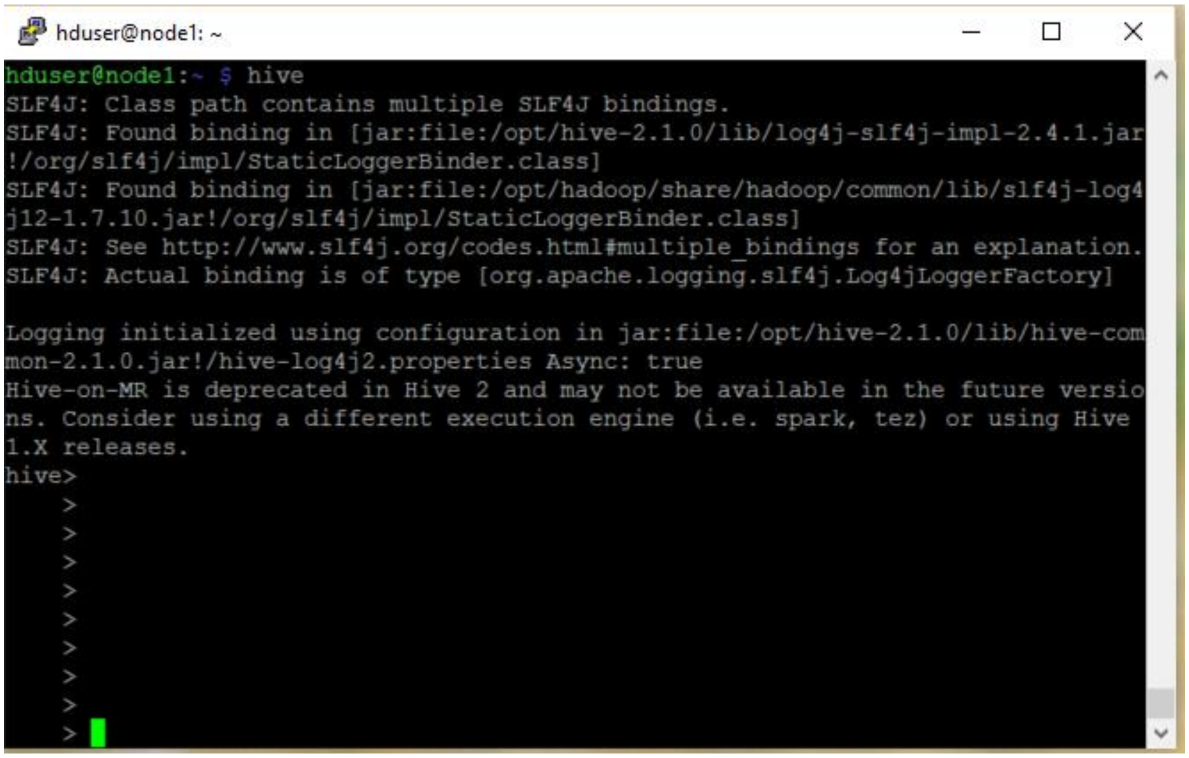
5. On node1, we will add some environmental variables:
   - -Type "**cd ~**"
   - -Type "**sudo nano .bashrc**"
   - -Enter the following additions at the bottom of .bashrc

     ```
     export HIVE_HOME=/opt/hive-2.1.0
     export PATH=$HIVE_HOME/bin:$PATH
     ```
   - -Type "**sudo reboot**"

6. Log back into node1 as hduser. We shall now start up hdfs and yarn services and make some directories.
   - -Type "**start-dfs.sh**"
   - -Type "**start-yarn.sh**"

7. On node1, we will also initialize a database for hive (never delete or modify the metastore_db directory directly-but if you do, you need to do this command again but your data in Hive will be lost).
   - -Type "**cd ~**"
   - -Type "**schematool -initSchema -dbType derby**"

8. On node1, you can start the hive command line interface (cli).
   - -Type "**hive**"


hduser@node1: ~

```
hduser@node1:~ $ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive-2.1.0/lib/log4j-slf4j-impl-2.4.1.jar
!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4
j12-1.7.10.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]

Logging initialized using configuration in jar:file:/opt/hive-2.1.0/lib/hive-com
mon-2.1.0.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive>
    >
    >
    >
    >
    >
    >
    >
    >
    >
```

Part IV: Spark 2.0 Installation

1. Here we will install Spark (Standalone Mode) on node1, node2, and node3. Then we will configure each node separately. node1 will be master node for spark and node2 and node3 slave nodes for spark. Before we install spark, we will install Scala on node1.

2. Find Scala by going to the http://www.scala-lang.org/download/2.11.8.html page in Windows and at the bottom find the link to the scala-2.11.8.deb package (e.g. http://downloads.lightbend.com/scala/2.11.8/scala-2.11.8.deb)

3. Install Scala on node1. Login to node1 as hduser.
   -Type "**sudo wget http://downloads.lightbend.com/scala/2.11.8/scala-2.11.8.deb**"
   -Type "**sudo dpkg -i scala-2.11.8.deb**"

4. Find Spark by going to the http://spark.apache.org/downloads.html page in Windows. Right click on the spark-2.0.0-bin-hadoop2.7.tgz link (step #4 on webpage) and paste into notebook to see the address (e.g. http://d3kbcqa49mib13.cloudfrontnet/spark-2.0.0-bin-hadoop2.7.tgz).

5. Install Spark on node1. Login to node1 as hduser.
   -Type "**cd /opt**"
   -Type "**sudo wget http://d3kbcqa49mib13.cloudfrontnet/spark-2.0.0-bin-hadoop2.7.tgz**"
   -Type "**sudo tar -xvzf spark-2.0.0-bin-hadoop2.7.tgz**"
   -Type "**sudo mv spark-2.0.0-bin-hadoop2.7 spark**"
   -Type "**sudo chown -R hduser:hadoop /opt/spark**"

6. Repeat step 5 for both node2 and node3.

7. On node1 we configure Spark. Login to node1 as hduser.
   -Type "**cd ~**"
   -Type "**sudo nano .bashrc**"
   Add the following to bottom of .bashrc file
   ```
   export PATH=$PATH:/opt/spark/bin
   export PATH=$PATH:/opt/spark/sbin
   export SPARK_HOME=/opt/spark
   export PATH=$SPARK_HOME:$PATH
   export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH
   ```
   -Type "**cd /opt/spark/conf**"
   -Type "**sudo cp slaves.template slaves**"
   -Type "**sudo nano slaves**"
   Add the following to bottom of slaves file
   ```
   node1
   node2
   node3
   ```
   -Type "**sudo reboot**"

8. On node2 we configure Spark. Login to node1 as hduser.
   -Type "**cd ~**"
   -Type "**sudo nano .bashrc**"
   Add the following to bottom of .bashrc file
   ```
   export PATH=$PATH:/opt/spark/bin
   ```
   -Type "**sudo reboot**"

9. On node3 we configure Spark. Login to node1 as hduser.
   - -Type "**cd ~**"
   - -Type "**sudo nano .bashrc**"

     Add the following to bottom of .bashrc file

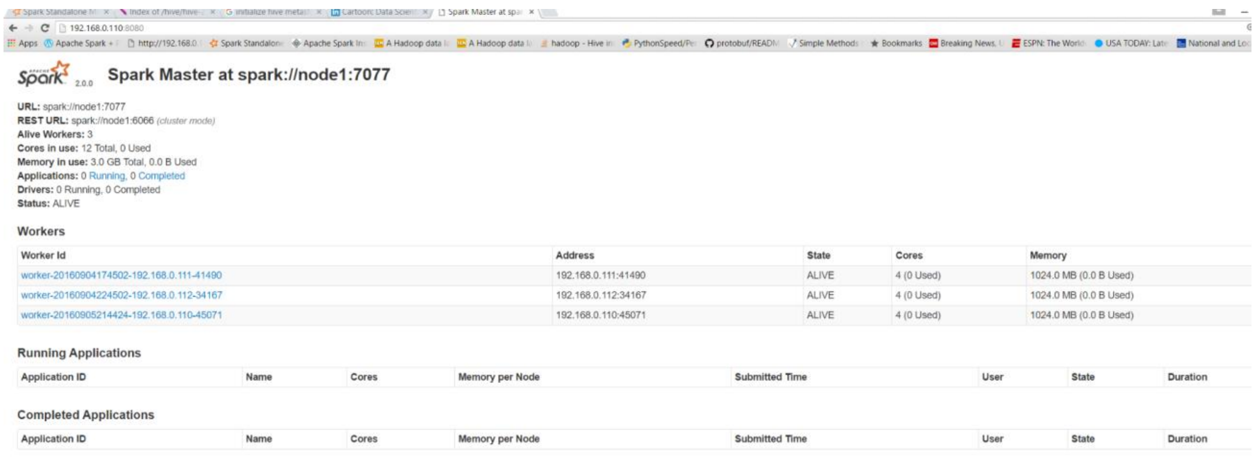
     ```
     export PATH=$PATH:/opt/spark/bin
     ```
   - -Type "**sudo reboot**"

10. On node1, login as hduser.
    - -Type "**/opt/spark/sbin/start-all.sh**"

11. To test we check jps.
    - -Type "**jps**"

    Note that we should see Master and Worker processes on node1 and Worker process only on nodes 2 and 3.

12. You can check the spark monitoring website as well in a web browser at http://192.168.0.110:8080.



192.168.0.110:8080

Spark Master at spark://node1:7077

URL: spark://node1:7077
REST URL: spark://node1:6066 *(cluster mode)*
Alive Workers: 3
Cores in use: 12 Total, 0 Used
Memory in use: 3.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20160904174502-192.168.0.111-41490	192.168.0.111:41490	ALIVE	4 (0 Used)	1024.0 MB (0.0 B Used)
worker-20160904224502-192.168.0.112-34167	192.168.0.112:34167	ALIVE	4 (0 Used)	1024.0 MB (0.0 B Used)
worker-20160905214424-192.168.0.110-45071	192.168.0.110:45071	ALIVE	4 (0 Used)	1024.0 MB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

13. You can also launch pyspark.
    - -Type "**pyspark**"