Hadoop Cluster
Setting up Hadoop in a Cluster

Summary

1. Hadoop version: 2.7.3 link http://hadoop.apache.org/releases.html
2. Java version: java-8
3. Other needed software: ssh, ubuntu, and rsync
4. Objective: Setting up hadoop in a cluster after you have set it up initially

What you need before you get started

Getting hadoop setup

Follow the directions in these labs on how to install hadoop and mapreduce.

Here is the lab3 part 1 and lab3 part 2

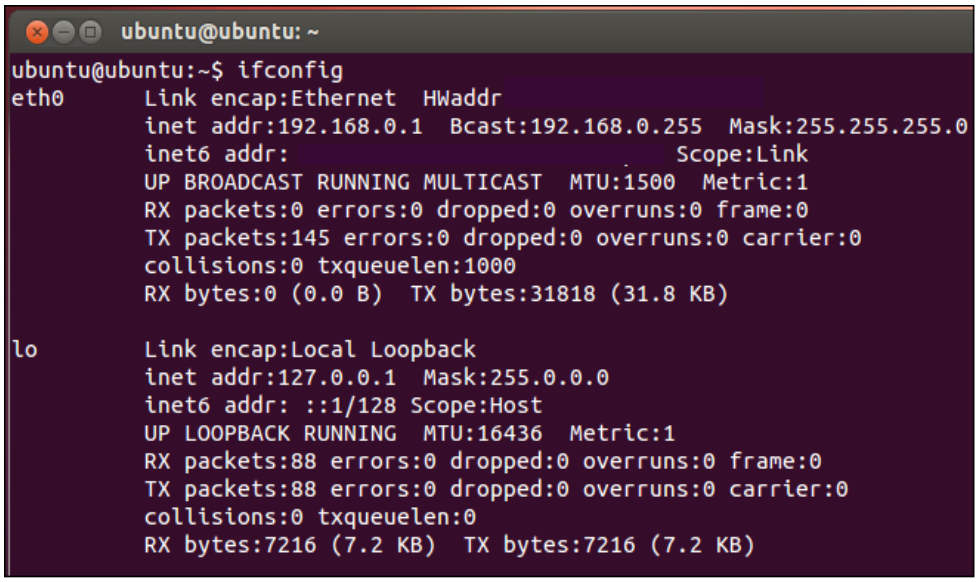
Getting Master and Slave ip addresses

Now you will need to get the ip addresses of your master and slave(s) that you will be using for your hadoop cluster.

To get these addresses just simply type this into your terminal

$ ifconfig

Do not type the $ when typing the commands



Here you can see what the terminal will display when you enter ifconfig.

So in this example the ip for this computer would be 192.168.0.1.

You will need to remember all the eth0 inet address of all your computers that will be used in the cluster.

Setting up hadoop in a cluster

As a side note I prefer to use nano as my text editor, but you can use any that you are familiar with.

Editing your hosts file.

This must be done on all of your computers that are part of the cluster.

The hosts file can be found at the etc directory which you can get there by typing

```
$ sudo /etc
```

Now as root you must add the master and slave(s) ip addresses to hosts

So as an example:

```
$ sudo nano hosts
```

Then add this to the file

```
192.168.0.1 master

192.168.0.2 slave
```

Save and exit.

In this example I added two ip addresses to the file the one ending with 1 was the master while the ip address ending with 2 was the slave.

This must be done on all computers not just the master computer.

This will enable you to be able to just type out master instead of the ip address of that machine every time you need to communicate with it or when you want to connect to the websites that will be setup when the cluster is running.

Editing your hostname file

Now you can edit your hostname file. It is found in the same directory as the hosts file, the /etc directory. You will need to be root to edit this file also, so make sure you are logged in as a user who has sudo privileges.

```
$ sudo nano hostname
```

Now on the master computer put in master, but on the slaves put in the slave name that is next to the ip address that you put in the hosts file

```
master
```

Save and exit.

Now you should see something like this in your terminal

```
ubuntu@master$
```

Changing the owner of hadoop

This section will needed to be done with all the computers not just the master computer.

For hadoop to run in a cluster you will need to change the owner of the hadoop home so that it is a user of your choice, and a new usergroup.

First make a new group called hadoop and then add the user, hduser, to that group.

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
```

Login as that user with su and generate a new ssh key for that user

```
$ su - hduser
$ ssh-keygen -t rsa -P ""
```

Just hit enter when it asks you where you want to save the file, it will save it at the default location which is ~/hduser/.ssh/id_rsa. ~ is short for that user's home dir

Now to enable ssh to access your pc

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Now to connect and test the setup.

Connect to master or slave, the name of the machine you are currently using.

```
$ ssh master
```

Now you can change who the owner of hadoop is.

Go to where you had placed hadoop and change the ownership of it to the new user. Must be root in order to change the owner so just hit control and d at the same time to log out until you are the user who has root privileges, or just start a new terminal.

```
$ cd /usr/local
$ sudo chown -R hduser:hadoop hadoop
```

Now hduser is the owner of hadoop.

Next editing .bashrc

Now make sure that you are logged in as the user who has permission for hadoop

For example my owner of hadoop was hduser. To log in as that user type

```
$ su - hduser
```

The dash will take you to the home dir for that user.

Now you need to edit the .bashrc file

```
$ nano .bashrc
```

At the end of the file add these lines:

```
export PATH=$PATH:$HADOOP_HOME/sbin

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 HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

Save and exit.

These are the needed paths in order for hadoop to operate correctly.

Editing the .xml files

The xml file that need to be edited can all be found at Hadoop_home/etc/hadoop/

```
$ cd /usr/local/hadoop/etc/hadoop/
```

You must be the user who owns hadoop or root to edit these files.

The first file to edit will be the core-site.xml file

```
$ nano core-site.xml
```

Now add these lines between the <configuration> tags

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

Next edit the hdfs-site.xml file

```
$ nano hdfs-site.xml
```

Add these lines between the configuration tags

```
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
```

Here the value is set to two since I was only using two computers for my cluster, change it to the amount of computers you would be using.

The next file to edit will be the mapred-site.xml file

```
$ nano mapred-site.xml
```

Add these lines between the configuration tags

```
<property>
 <name>mapred.job.tracker</name>
 <value>hdfs://master:9000</value>
</property>
<property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
</property>
```

Now for the last xml, yarn-site.xml, file the slaves will get a different input than the master file.

```
$ nano yarn-site.xml
```

For the master file put this in the file between the configuration tags

```
<property>
 <name>yarn.nodemanager.aux-services</name>
 <value>mapreduce_shuffle</value>
</property>
<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value>hdfs://master:8025</value>
</property>
<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>hdfs://master:8030</value>
</property>
<property>
```

```
 <name>yarn.resourcemanager.address</name>
 <value>hdfs://master:8050</value>
</property>
```

And for the slave(s) add this between the configuration tags

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

The reason I had two different formats for the yarn-site.xm file is that the master tends to deal a lot more with yarn while the slaves have very little to do as yarn.

Editing hadoop-env.sh

Several guides that I read advised disabling IPv6 so that is what I did to avoid any issues that may have come up from that.

Now to disable IPv6 in hadoop. To do that go to the hadoop-env.sh file which be found at this path /usr/local/hadoop/etc/hadoop. Same directory as the xml files that we edited earlier.

Edit the file and add this to it so that it will only use IPv4

```
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

Next you may need to edit your Java_home path in this file.

It should look like this

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

This will allow hadoop to be able to find java.

Editing the slave and master file

At Hadoop_home/etc/hadoop edit these two files on all computers

```
$ nano master
```

Add this to the master file

```
master
```

Save and exit.

Next edit the slave files

```
$ nano slave
```

Add this to the slave file

```
master
slave
```

Save and exit.

Since I was only using two computers I had the master as one of the slaves. If using more than two you could remove it from the slave file and add the other slaves to that file.

How to use the Hadoop File System

There will be several hadoop file system commands that will be used in this lab. This is the first one that you will see,

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

This command is similar to the make directory command that is used in ubuntu and many other linix operating systems mkdir. This command will make a directory that all slaves and master can use. The /user is the end directory that is being made, the last directory to be made is the one that is right before the last /. Another command that is used throughout this lab will be the -put command. Here is an example of the -put command,

```
$ bin/hdfs dfs -put etc/hadoop/ input
```

This command takes a directory that is in the host machine and puts it in the hadoop file system, hdfs for short, that is specified by the path that is inserted. Here the path is etc/hadoop/, and it copies the directory input from the host and puts it into the hdfs in the path etc/hadoop. After this command the user can find the input directory at etc/hadoop/input.

Starting and Stopping the Cluster

Starting the Cluster

If you run into any problems check the conclusion section, it will have some usefull info on what might be causing the issue.

These are the needed steps that must be taken every time you start a cluster.

Now that you have all the files edited you should be able to run hadoop in a cluster.

First login as the owner of hadoop, make sure you are on the master computer when you do these next steps.

```
$ su - hduser
```

Next go to hadoop home

```
$ cd /usr/local/hadoop
```

Now to start up the cluster. Enter these commands each one at a time.

```
$ bin/hdfs namenode -format <cluster_name>
$ sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR --script hdfs start namenode
$ sbin/hadoop-daemons.sh --config $HADOOP_CONF_DIR --script hdfs start datanode
$ sbin/start-dfs.sh
```

Now you will need a filesystem that will work with hadoop so enter these next to set it up.

```
$ bin/hdfs dfs -mkdir /user
$ bin/hdfs dfs -mkdir /user/hduser
$ bin/hdfs dfs -put etc/hadoop/ input
```

These commands will make a directory for hadoop on your master and slave(s) that can be accessed by all in the cluster. It also copies all the files found at etc/hadoop and puts them in a new file called input in the hdfs, hadoop filesystem.

Now you can start the yarn nodes that are needed.

```
$ sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start resourcemanager
$ sbin/yarn-daemons.sh --config $HADOOP_CONF_DIR start nodemanager
$ sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start proxyserver
$ sbin/start-yarn.sh
$ sbin/mr-jobhistory-daemon.sh --config $HADOOP_CONF_DIR start historyserver
```

You can see the output file and all your other files and some interesting info on your cluster nodes at the url

```
master:50070
```

You can see some info on your current, running, pending, done, and many more applications at the url

```
master:8088
```

And to see the jobhistory simple go to this url

```
master:19888
```

50070, 8088, and 19888 are all default ports for these features, however in different versions of hadoop they may be different.

Now you can run an example jar file that they provide you to test if the cluster is working.

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.3.jar grep input output 'dfs[a-z.]+'
```

This should work and you can view the output file from the master:50070 url by going to the browse files tab at the top of the page and moving to the output folder. There you can download the file and view the output of the test run.

Stopping the Cluster

Now you must stop the cluster with these commands.

```
$ sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR --script hdfs stop namenode
$ sbin/hadoop-daemons.sh --config $HADOOP_CONF_DIR --script hdfs stop datanode
$ sbin/stop-dfs.sh
$ sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR stop resourcemanager
$ sbin/yarn-daemons.sh --config $HADOOP_CONF_DIR stop nodemanager
$ sbin/stop-yarn.sh
$ sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR stop proxyserver
$ sbin/mr-jobhistory-daemon.sh --config $HADOOP_CONF_DIR stop historyserver
```

Testing the Cluster

Now that the cluster is setup we can test it with a simple bit of java code that will count the number of occurrences in a text file. Take the code for the java program from the lab 3 that was mentioned at the beginning of the lab. It is important that the input for the code is in the hdfs so that all the computers will have access to it. After you have made the java program move the directory that the program is located in and put it into the hdfs, this can be done with the -put command. So for example my code was called WordCounter its files that I wanted to test it against were in the etc/hadoop/ directory so I moved them there with the -put command,

```
$ bin/hdfs dfs -put etc/hadoop/ input
```

Now that the word files are in the hdfs it can now be run with multiple nodes, and we can see if the cluster is working correctly. To run this on the clusters use this command that is very similar to the other one that had been used previously in this lab,

```
$ bin/hadoop jar wc/wc.jar WordCount input output 'dfs[a-z.]+'
```

You can now see the result of this simple word counter program in the output file, which is in the hdfs so you will need to go to master:50070 and go to the utilities and browse the filesystem to find the output of the program. It should look similar to the output of lab 3, however there will be many more words that it had searched through.

Know Errors and Solutions to problems

I ran into multiple issues if I did not remove the old directory system that I had made. So to get around this issue I had to remove the old filesystem every time. This can be done by going to the /tmp dir. Once you are there as sudo remove all the hadoop-* dirs that are in the /tmp dir. To do this just type this command

$ sudo rm -r hadoop*

Make sure that you are in the right place when you run this command. You can see if it deleted the files by using this piped command

$ ls | grep hadoop

If nothing comes up then the old files are now gone to trash

Conclusion

You now have all the necessary steps to take to have a working two node cluster for hadoop. If you run into some errors you can go to the logs to help pinpoint where they are coming from. The log files are found on the slave and master computers at hadoop_home/logs. Also if you are having troubles connecting to the master from slave or vise versa then make sure that your network is setup to allow devices on the network to connect with each other, you may have to change some settings of your router. With this setup I was able to have a working two node cluster, however there may be better ways of setting up a cluster so I would advise going over and reading some of the websites that I have listed below.

Websites I used to setup the cluster

For single node cluster

I used these two websites to set up the cluster in a single node

1. https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/SingleCluster.html
2. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

The second website uses a different version of hadoop then the one that is used in this lab so be wary if you decide to read that guide.

I used these two for setting up the multi-node cluster

1. https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/ClusterSetup.html
2. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/