Installing HDFS and Hadoop 2.X on a Multi-node cluster with Ubuntu 14.04

This guide is shows step by step how to set up a *multi node cluster* with **Hadoop and HDFS 2.4.1 on Ubuntu 14.04**. It is an update, and takes many parts from previous guides about installing Hadoop&HDFS versions 2.2 and 2.3 on Ubuntu.

Assume we have a 3 nodes cluster, my test case is the following (with IP addresses and shortnames):

```
10.10.10.104  mynode1
10.10.10.105  mynode2
10.10.10.106  mynode3
```

Setup

Make sure you have Oracle JDK 7 or 8 installed. *The following are the commands for java 8, to install java 7 you just need to change the version number*

```
sudo add-apt-repository ppa:webupd8team/java -y
sudo apt-get update
sudo apt-get install oracle-java8-installer
sudo apt-get install oracle-java8-set-default
```

Note: I know some of you are trying to run this guide with debian. I am not sure how much of these guide will apply to that OS, but for this specific case, for debian, the instructions to install Java 8 are here.

While we are installing software, you can find useful to install also `screen` to start sessions of work on remote servers, and `nmap` to check server ports in case something is not working in the cluster networking

```
sudo apt-get install screen nmap
```

Repeat this installation procedure, up to this point, on every node you have in the cluster

The following will be necessary only on the first node: Then we start a screen to work remotely without fear of losing work if disconnected.

```
screen -S installing
```

After the `-S` you can put whatever name for your sessions

Now we are going to actually install the software needed with `maven` with libraries to compile hdfs&hadoop.

```
sudo apt-get install  maven build-essential zlib1g-dev cmake pkg-config libssl-dev protobuf-compiler
```

Among these files, `protoc` or also called `protobuf-compiler` may cause some problems with the version depending on your operating system version. In that case, you can compile and install the correct version (`2.5.0`) from the source.

Hadoop User & Authentication

Next, let's create `hadoop` group and the user `hduser`, which will be also in the sudoers, the following commands have to be run one at at time. In the second step the `adduser` will also ask the login password for `hduser`:

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

Repeat this procedure, up to this point, on every node you have in the cluster

We now log in as the new `hduser` on one node and we will create SSH keys to access the other servers:

```
sudo su - hduser
```

From now on, in the rest of this guide, all commands will be run as the `hduser`.

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Now let's copy these files on the other nodes, e.g, from `mynode1` to `mynode2` and `mynode3`

```
scp -r ~/.ssh  hduser@10.10.10.106:~/
```

Compile the Sources

The following steps will be needed only once. Download hadoop `2.X` stable, to do so you navigate in the List of Mirrors select one and decide what version to download. With `wget` you can run something like the following for hadoop `2.4.1` - from europe:

```
wget http://www.eu.apache.org/dist/hadoop/core/hadoop-2.4.1/hadoop-2.4.1-
src.tar.gz
```

From the U.S. instead

```
wget http://apache.mirror.anlx.net/hadoop/core/hadoop-2.4.1/hadoop-2.4.1-
src.tar.gz
```

Once it has been downloaded, unpack it

```
tar -xvf hadoop-2.4.1-src.tar.gz
```

Then enter the directory and compile

```
cd hadoop-2.4.1-src/
mvn package -Pdist,native -Dmaven.javadoc.skip=true  -DskipTests -Dtar
```

Notice that, if you are behind a proxy, maven needs a `settings.xml` file in the configuration directory in `~/.m2` that contains the basic information of your proxy configuration.

Compiled files will be found in `hadoop-dist/target/hadoop-2.4.1.tar.gz` just put them in the home

```
mv hadoop-dist/target/hadoop-2.4.1.tar.gz ~/
```

Now let's copy these files on the other nodes, e.g, from `mynode1` to `mynode2` and `mynode3`

```
scp ~/hadoop-2.4.1.tar.gz  hduser@10.10.10.105:~/
scp ~/hadoop-2.4.1.tar.gz  hduser@10.10.10.106:~/
```

Install the Compiled Code

The following steps will be needed on all the machines We unpack the compiled version and put it n `/usr/local` and we create a shortcut called `/usr/local/hadoop`

```
sudo tar -xvf ~/hadoop-2.4.1.tar.gz -C /usr/local/
sudo ln -s /usr/local/hadoop-2.4.1 /usr/local/hadoop
sudo chown -R hduser:hadoop /usr/local/hadoop-2.4.1
```

Set up ENV Variables

The following steps will be needed on all the machines We update the profile of the shell, i.e., we edit the `.profile` file to put some enviroment variables, in order to upset equally `vim` and `emacs` user we will use a text editor called `nano`

```
nano ~/.profile
```

And we add, at the end, the following

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
export HADOOP_INSTALL=/usr/local/hadoop
export HADOOP_HOME=$HADOOP_INSTALL
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export HADOOP_CONF_DIR=${HADOOP_HOME}"/etc/hadoop"
export YARN_HOME=$HADOOP_INSTALL

alias hfs="hdfs dfs"
```

(To save `CTRL+o` `ENTER` and `CTRL+x`)

Note: If you installed somewhere else hadoop, chec the proper directory path for `$HADOOP_INSTALL`, but do not change `$HADOOP_CONF_DIR`.

Now we made the edit operative by reloading the `.profile` file with

```
source ~/.profile
```

We also have to edit `hadoop-env.sh` files with for the same `$JAVA_HOME` variable, that they seem not able to set up properly, so we open the file in

```
nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

and around line 27 we can replace

```
export JAVA_HOME=${JAVA_HOME}
```

with

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
```

If you want to be sure it worked, you can paste some values, like

```
echo $JAVA_HOME
echo $HADOOP_HOME
```

Set up Data Directory & Logs

We create the directory where `hdfs` data files and logs will be stored, you can create them wherever you like

The first directory is actually needed only on the NameNode (main) machine

```
mkdir -pv /usr/local/hadoop/data/namenode
```

These steps will be needed on all the machines

```
mkdir -pv /usr/local/hadoop/data/datanode
mkdir -pv $HADOOP_INSTALL/logs
```

Edit Configuration Files

These steps will be needed only on the main machine, then we will copy the entire conf directory on the other machines

Then we put this information in the `hdfs-site.xml` file with

```
nano $HADOOP_INSTALL/etc/hadoop/hdfs-site.xml
```

And paste the following between `<configuration>` tag:

```
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///usr/local/hadoop/data/datanode</value>
    <description>DataNode directory</description>
</property>

<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///usr/local/hadoop/data/namenode</value>
    <description>NameNode directory for namespace and transaction logs
storage.</description>
</property>
```

The following are additional configuration parameters to put alongside the previous ones, among them the replication parameter to match the number redundant copy we want - it does not necessarily match the number of nodes in the cluster.

```
<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>
```

Notice: when you will start your HDFS distributed filesystem, you will have a main `NameNode` and a `Secondary NameNode`. The `Secondary NameNode` is *not what you think it is*.

> The term "secondary name-node" is somewhat misleading. It is not a name-node in the sense that data-nodes cannot connect to the secondary name-node, and in no event it can replace the primary name-node in case of its failure. – From Hadoop FAQ

In any case you may want to put the secondary name node on a different machine that is not the master, but maybe one of the workers. Assume you decide your cluster main node is

```
10.10.10.104  mynode1
```

and assume you decide your cluster to have the Secondary NameNode on

```
10.10.10.105  mynode2
```

then we add the following to the `hdfs-site.xml` file :

```
<property>
 <name>dfs.namenode.http-address</name>
 <value>10.10.10.104:50070</value>
 <description>Your NameNode hostname for http access.</description>
</property>

<property>
 <name>dfs.namenode.secondary.http-address</name>
 <value>10.10.10.105:50090</value>
 <description>Your Secondary NameNode hostname for http access.</description>
</property>
```

I thank my colleague Sabeur for helping me with this bit on the Secondary NameNode

Then we also point to `mynode1` IP to for the Hadoop cluster to tell where we host the hadoop `NameNode` by editing:

```
nano $HADOOP_INSTALL/etc/hadoop/core-site.xml
```

and we add inside the `<configuration>` tag the following

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://10.10.10.104/</value>
    <description>NameNode URI</description>
</property>
```

We put the IP addresses of the nodes to be used as `DataNodes` in the `slaves` file, we open it with

```
    nano $HADOOP_INSTALL/etc/hadoop/slaves
```

And we put the list of server addresses one per line, note that in this case also the master is used, so we put there the following list:

```
    10.10.10.104
    10.10.10.105
    10.10.10.106
```

Up to here was mainly about `HDFS`, now we configure the `yarn` cluster, i.e., the execution engine, we then edit the `yarn-site.xml`.

```
nano $HADOOP_INSTALL/etc/hadoop/yarn-site.xml
```

Again we add the following inside the `<configuration>` tag

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>10.10.10.104:8025</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>10.10.10.104:8030</value>
</property>
<property>
    <name>yarn.resourcemanager.address</name>
    <value>10.10.10.104:8050</value>
</property>
```

Now is time to update all the nodes with this news configuration, thus we copy from `mynode1` to `mynode2` and `mynode3` the directory with the following command (*note the destination directory*)

```
scp -r  $HADOOP_INSTALL/etc/hadoop  hduser@10.10.10.105:$HADOOP_INSTALL/etc/
scp -r  $HADOOP_INSTALL/etc/hadoop  hduser@10.10.10.106:$HADOOP_INSTALL/etc/
```

Initialize HDFS

These commands will be used only on the main node

If all went well we should be able to run the following command

```
hadoop version
```

and obtain something like

```
Hadoop 2.4.1
Subversion Unknown -r Unknown
Compiled by hduser on 2014-08-23T15:29Z
Compiled with protoc 2.5.0
From source with checksum bb7ac0a3c73dc131f4844b873c74b630
This command was run using /usr/local/hadoop-2.4.1/share/hadoop/common/hadoop-
common-2.4.1.jar
```

Now the first step is to `format` the NameNode, this will basically initialize the `hdfs` file system. So on the main node you run:

```
hdfs namenode -format
```

> Hadoop NameNode is the centralized place of an HDFS file system which keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. In short, it keeps the metadata related to datanodes. When we format namenode it formats the meta-data related to data-nodes. – From StackOverflow

Start and test the Cluster!

These commands will be used only on the main node Now we can start the `hdfs` cluster with the command

```
start-dfs.sh
```

And if the preivious command didn't complain about anythign, we can create a random directory in our `HDFS` filesystem with

```
hadoop fs -mkdir -p /datastore
```

Note that we used the full `hadoop fs` command, but in our profile we added an alias with `hfs`

now check the size of the files inside the `datanode` directory

```
du -sh /usr/local/hadoop/data/datanode
```

and we can put inside a new directory and, as a test, the `.tar.gz` file of hadoop

```
hfs -mkdir -p /datastore/test
hfs -copyFromLocal ~/hadoop-2.4.1.tar.gz /datastore/
```

now check again the size of the files inside the `datanode` directory, you can run the same command on all nodes, and see that the file is also on those other servers (*all of it or part, it depends on the replication level and the number of nodes you have*)

```
du -sh /usr/local/hadoop/data/datanode
```

You can check the content of the `hdfs` directory with

```
hfs -ls /datastore
```

and remove the all the files with

```
hfs -rm /datastore/test/*
```

In case you want to delete an entire directory you can instead use

```
hfs -rm -r /datastore/test
```

This is the distributed file system running, and you can check the processes with

```
jps
```

Which will give you, on the main node, something like

```
18755 DataNode
18630 NameNode
18969 SecondaryNameNode
19387 Jps
```

Up to here we set up the distributed filesystem, this will be come handy not only for `hadoop`, but also for other distributed computation engines, like Spark or Flink - which was Stratosphere.

Finally to start the actual `hadoop yarn` execution engine you just go with

```
start-yarn.sh
```

Configure Hostnames

As a side note, in this guide, we used IP addresses in configuration files, if you want to use instead the shortnames you shall first update the `/etc/hosts` so that all of them are listed with their shortname.

```
10.10.10.104  mynode1
10.10.10.105  mynode2
10.10.10.106  mynode3
```

In this case, make sure that there, the only appeareance of the ip `127.0.0.1` is with `localhost`. This is very important, so if in you `hosts` file there is a line like

```
127.0.0.1 mynode1
```