I am assuming that you have created a network of at least 2 machines with LINUX, you have setup their IP's and performed networking operations between them as discussed in the classroom sessions.

We will be creating a cluster with 2 machines, one with name node and another with data node,
You can create cluster of (2+ machines), one with name node and rest of them with data node,

Configuring the name node

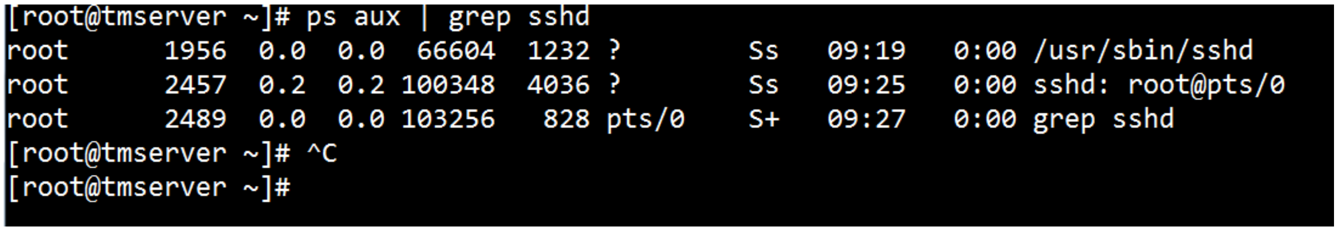
The hostname of my machine is **tmserver**, you can use the (hostname) command to determine the hostname

Login as root user

SSH – Server

Let us find if process sshd is running or not. For that type
ps aux | grep sshd

I got the following output

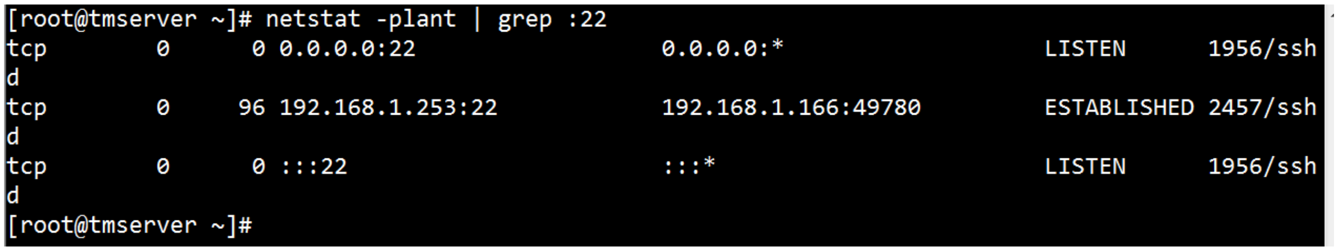


```
[root@tmserver ~]# ps aux | grep sshd
root      1956  0.0  0.0  66604  1232 ?        Ss   09:19   0:00 /usr/sbin/sshd
root      2457  0.2  0.2 100348  4036 ?        Ss   09:25   0:00 sshd: root@pts/0
root      2489  0.0  0.0 103256   828 pts/0    S+   09:27   0:00 grep sshd
[root@tmserver ~]# ^C
[root@tmserver ~]#
```

the /usr/sbin/sshd part tells me that it is running

Now let us check if it running on port 22 for that type
netstat -plant | grep :22

I got the following output



```
[root@tmserver ~]# netstat -plant | grep :22
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN      1956/ssh
d
tcp        0     96 192.168.1.253:22            192.168.1.166:49780         ESTABLISHED 2457/ssh
d
tcp        0      0 :::22                       :::*                        LISTEN      1956/ssh
d
[root@tmserver ~]#
```

The output tells me that the process with id 1956 (ssh) is running on port 22.
Note : in your case the process id might not be 1956, it doesn't matter.

Alternatively we can also use
service   sshd status

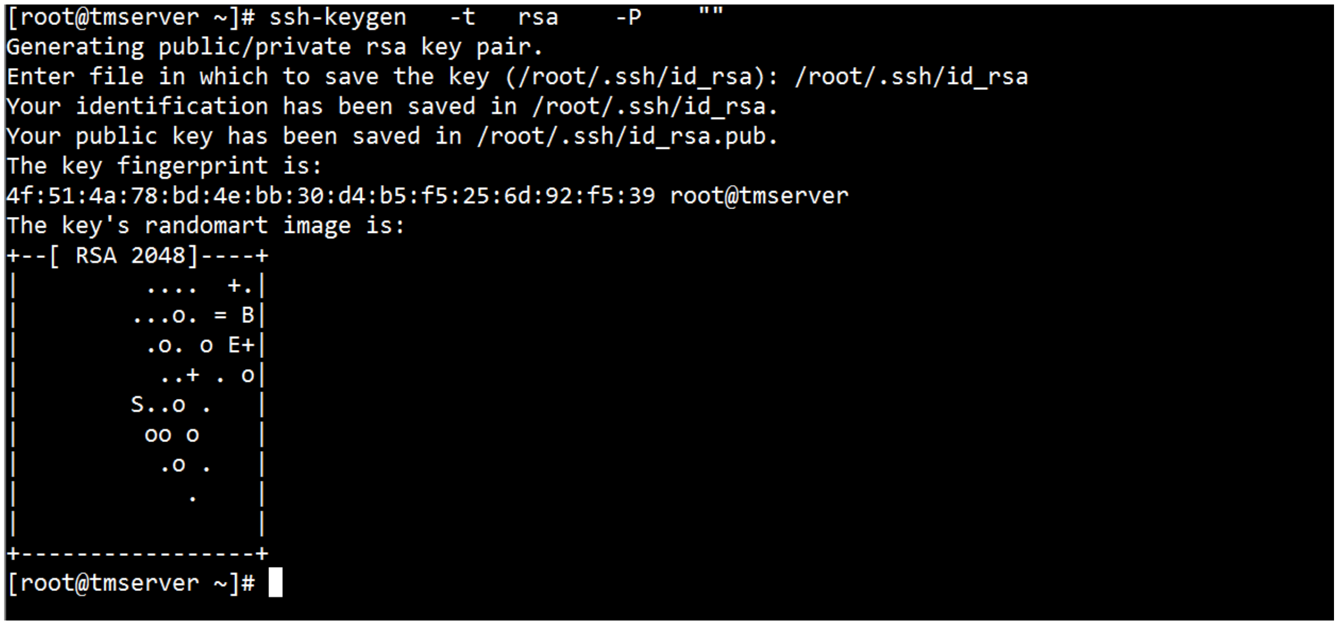
if it is not in started mode, we can type

service sshd start

We will now configure ssh to allow other machines in the cluster to login (without password for the root/hadoop user)
Now let us create a public/private key pair, it will be shared across the cluster.
Right now our working folder is root
For that type
ssh-keygen -t rsa -P ""

you will be asked for file name, provide /root/.ssh/id_rsa

I got the following output



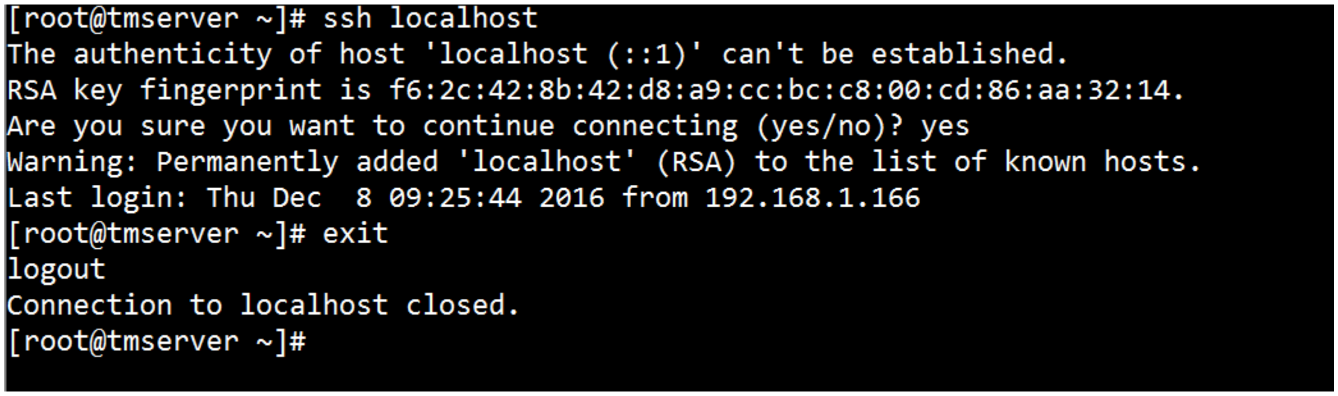
```
[root@tmserver ~]# ssh-keygen   -t   rsa    -P    ""
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/id_rsa
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
4f:51:4a:78:bd:4e:bb:30:d4:b5:f5:25:6d:92:f5:39 root@tmserver
The key's randomart image is:
+--[ RSA 2048]----+
|         ....  +.|
|        ...o. = B|
|         .o. o E+|
|          ..+ . o|
|        S..o .   |
|         oo o    |
|          .o .   |
|            .    |
|                 |
+-----------------+
[root@tmserver ~]#
```

Now let us append the contents of the generated file to (authorized_keys) for password less access.

For that type
cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
Now let us change some rights, for that type
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys

Now to check if password less login is allowed, type
ssh localhost
if prompted with a question type (yes) and you should see that you have logged in.
Note : it is not the login in which you typer (ssh localhost), it is another shell, type exit to get out of it.

Refer to the following screenshot if you are unable to understand



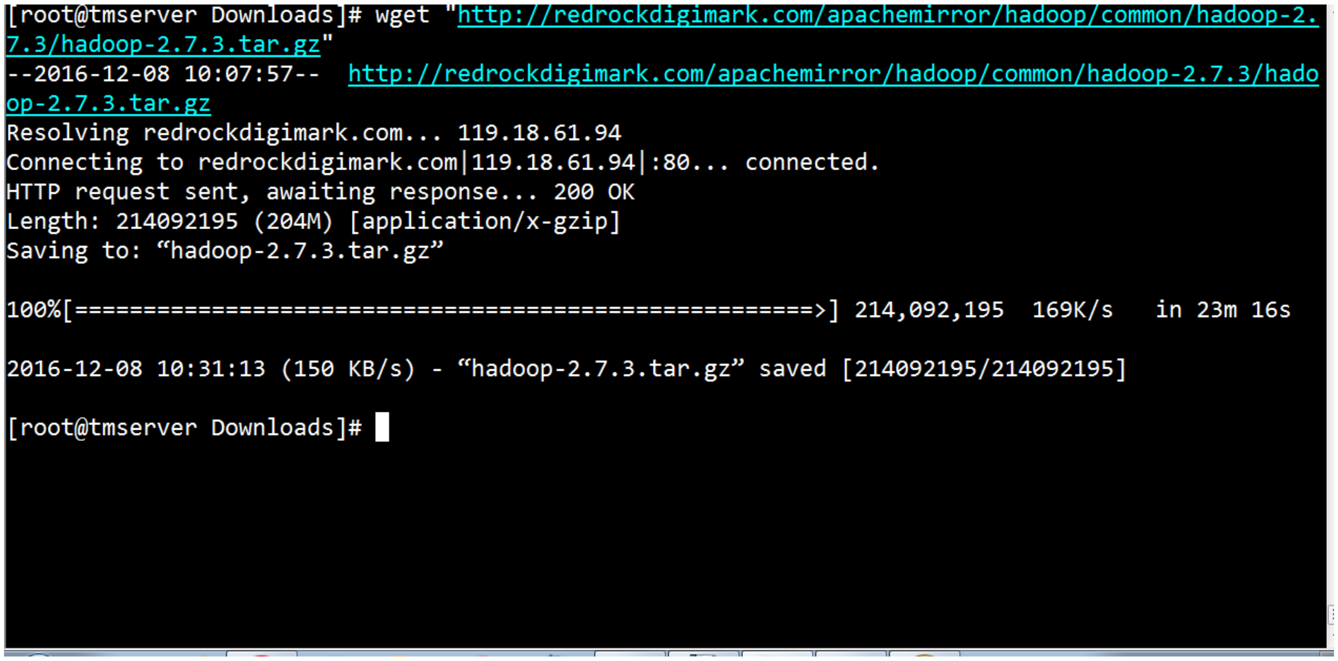
```
[root@tmserver ~]# ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is f6:2c:42:8b:42:d8:a9:cc:bc:c8:00:cd:86:aa:32:14.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Last login: Thu Dec  8 09:25:44 2016 from 192.168.1.166
[root@tmserver ~]# exit
logout
Connection to localhost closed.
[root@tmserver ~]#
```

Now let us download hadoop zip file (I am going to download version 2.7.3) for that type

Move to /root/Downloads folder

wget "http://redrockdigimark.com/apachemirror/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz"

It will take some time as it is about 200 MB zip file, or you can take it from somebody else, in pen drive, mount it and copy it to your downloads folder.



```
[root@tmserver Downloads]# wget "http://redrockdigimark.com/apachemirror/hadoop/common/hadoop-2.
7.3/hadoop-2.7.3.tar.gz"
--2016-12-08 10:07:57--  http://redrockdigimark.com/apachemirror/hadoop/common/hadoop-2.7.3/hado
op-2.7.3.tar.gz
Resolving redrockdigimark.com... 119.18.61.94
Connecting to redrockdigimark.com|119.18.61.94|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 214092195 (204M) [application/x-gzip]
Saving to: “hadoop-2.7.3.tar.gz”

100%[=====================================================>] 214,092,195  169K/s   in 23m 16s

2016-12-08 10:31:13 (150 KB/s) - “hadoop-2.7.3.tar.gz” saved [214092195/214092195]

[root@tmserver Downloads]#
```

Now unzip it using
tar xzvf hadoop-2.7.3.tar.gz

now the downloads directory should have a folder named as hadoop-2.7.3

now let us move to /usr/local with a new name as hadoop, for that type

mv hadoop-2.7.3 /usr/local/hadoop

Now let us create a directory structure for name node, for that type

mkdir -p /usr/local/hadoop_work/hdfs/namenode

Note : the -p option forces creation of parent folders that don't exist

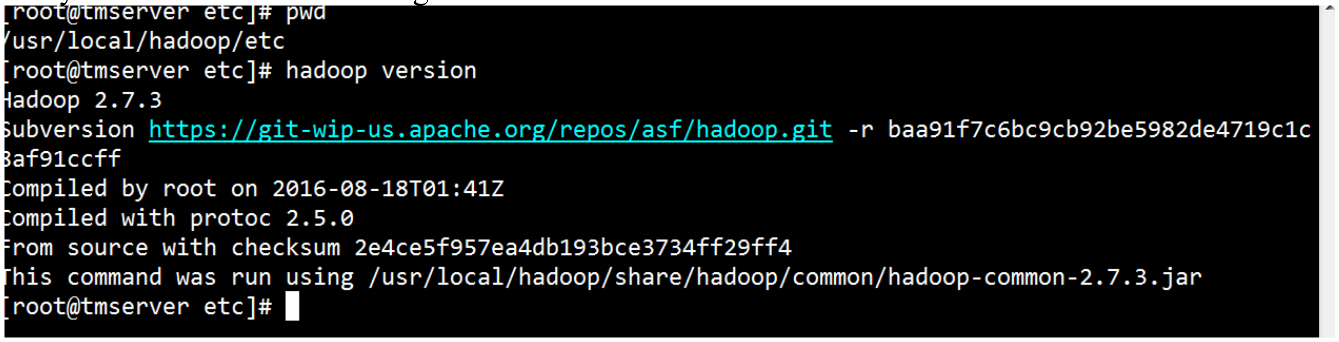
Now let us create a structure for namesecondary, for that type

mkdir -p /usr/local/hadoop_work/hdfs/namesecondary

now while staying in the root directory open the .bashrc file and append the following to its end

```
export   HADOOP_HOME=/usr/local/hadoop
export   PATH=$PATH:$HADOOP_HOME/bin
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   CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.
export   HADOOP_OPTS="$HADOOP_OPTS    -Djava.security.egd=file:/dev/../dev/urandom"
```

then type
source .bashrc
Now move to the /usr/local/hadoop/etc folder and type
hadoop version
and you should see the following



```
[root@tmserver etc]# pwd
/usr/local/hadoop/etc
[root@tmserver etc]# hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be5982de4719c1c
3af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.3.jar
[root@tmserver etc]#
```

Now let us configure the core-site.xml

This core-site.xml is a configuration file used to specify I/O settings that are common to HDFS and MapReduce.

move to /usr/local/hadoop/etc/hadoop
now edit the **core-site.xml** file and put the following in between configuration tag

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://tmserver:8020/</value>
</property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
</property>
```

Note : replace tmserver with your hostname

Now edit the **hdfs-site.xml** file
using this configuration file we will specify the name node directory and the number of replications required.

in it put the following between the configuration tag

```
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_work/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_work/hdfs/datanode</value>
</property>
<property>
<name>dfs.namenode.checkpoint.dir</name>
<value>file:/usr/local/hadoop_work/hdfs/namesecondary</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.block.size</name>
<value>134217728</value>
</property>
```

Now copy the mapred-site.xml.template to mapred-site.xml using

```
cp   mapred-site.xml.template   mapred-site.xml
```

now edit the mapred-site.xml and put the following between the configuration tag
We are telling that we will be using YARN framework

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobhistory.address</name>
<value>tmserver:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>tmserver:19888</value>
</property>
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/user/app</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Djava.security.egd=file:/dev/../dev/urandom</value>
</property>
```

Note : replace tmserver with your hostname

Now edit the yarn-site.xml and put the following between the configuration tag

```
<property>
<name>yarn.resourcemanager.hostname</name>
<value>tmserver</value>
</property>
<property>
<name>yarn.resourcemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<property>
<name>yarn.nodemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<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.log-aggregation-enable</name>
<value>true</value>
</property>
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>file:/usr/local/hadoop_work/yarn/local</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>file:/usr/local/hadoop_work/yarn/log</value>
</property>
<property>
<name>yarn.nodemanager.remote-app-log-dir</name>
<value>hdfs://tmserver:8020/var/log/hadoop-yarn/apps</value>
</property>
```

Note : replace tmserver with your hostname
Now create a file named as **masters** and type the hostname in it, in my case the masters file contains only one line

```
tmserver
```

Now we will format the namenode before proceeding, for that type

```
/usr/local/hadoop/bin/hadoop    namenode   -format
```

Somewhere in the output you should see a message that says Storage directory /usr/local/hadoop_work/hdfs/namenode has been successfully formatted

Now let us configure the data node.

Very very important

The hostname of my machine that I have decided to use as data node is linux2server and its ip is 192.168.1.172
And my namenode machine on which I have already configured the SSH server with key and have also installed hadoop is tmserver and its ip is 192.168.1.253

Now mount a pendrive and copy the /root/.ssh/id_rsa.pub to the pen drive and rename it to authorized_ keys

Then unmount the pen drive and copy the authorized_keys file to the data node machine under /root/.ssh folder

Understand this part very very very very clearly

My Name node Machine with hostname as tmserver and ip as 192.168.1.253 (/etc/hosts) file has the following in the end

```
192.168.1.253   tmserver
192.168.1.172   linux2server
```

My Data node machine with hostname as linux2server and ip as 192.168.1.172 (/etc/hosts) file has the following in the end

```
192.168.1.172   linux2server
192.168.1.253   tmserver
```

Now login into the data node machine and set the /etc/hosts as discussed earlier

On the datanode machine type the following

```
mkdir   -p   /usr/local/hadoop_work/hdfs/datanode
mkdir   -p   /usr/local/hadoop_work/yarn/local
mkdir   -p   /usr/local/hadoop_work/yarn/log
```

Now on the NameNode machine in my case tmserver, type

```
ssh   linux2server
```

and you will be able to login without password, so conclusion we are able to login from NameNode machine to DataNode machine without password.

Now the magic part, let us copy the hadoop folder from the NameNode machine to the DataNode machine using scp (copy through ssh)

for that on the namenode server (tmserver in my case) move to /usr/local and then type

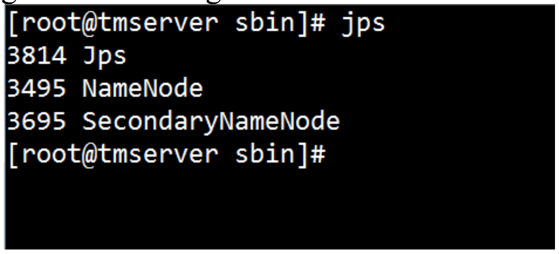
```
scp  -r   hadoop   linux2server:/usr/local
```

If the above doesn't work for some reason then just copy the hadoop folder under /usr/local from the

name node machine to the datanode machines /usr/local folder using a pen drive or something or copy the zip file and unzip it as done earlier
Now we are working on linux2server (the data node)
Now edit the .bashrc file under the root folder and add the following to it

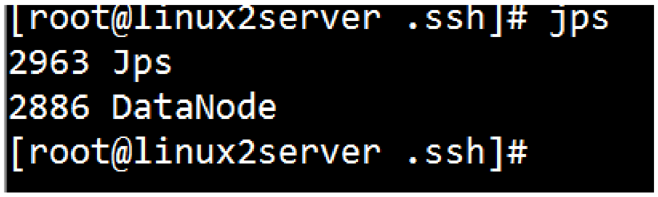
```
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.
export HADOOP_OPTS="$HADOOP_OPTS -Djava.security.egd=file:/dev/../dev/urandom"
```

then type source .bashrc
If you have more data nodes, then do the same for them.
Now let us start the hadoop distributed system, we are on name node (tmserver in my case)
move to
/usr/local/hadoop/sbin and type
start_dfs.sh
if or whenever asked for yes/no, type yes
Now wait for some time
then when the prompt appears, type
jps on namenode and datanode to check the status
on the tmserver (namenode) I got the following



```
[root@tmserver sbin]# jps
3814 Jps
3495 NameNode
3695 SecondaryNameNode
[root@tmserver sbin]#
```

On the linux2server (data node I got the following)



```
[root@linux2server .ssh]# jps
2963 Jps
2886 DataNode
[root@linux2server .ssh]#
```

That's it, we have successfully configured everything. Some more configurations are required, but we will do it later on. To stop type stop-dfs.sh and then again check the status with jps on namenodes and data nodes. This time you won't see the DataNode and NameNode part in the result of jps, which means that we have shutdown the hadoop services.