João Torres Follow
Feb 4, 2020 · 7 min read · Listen



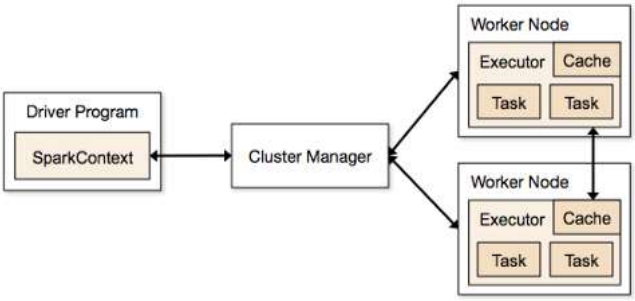
# How to Install and Set Up an Apache Spark Cluster on Hadoop 18.04


In this arcticle I will explain how to install **Apache Spark** on a multi-node cluster, providing step by step instructions.

## Spark Architecture

Apache Spark follows a master/slave architecture with two main daemons and a cluster manager.

- Master Daemon — (Master/Driver Process)
- Worker Daemon –(Slave Process)
- Cluster Manager

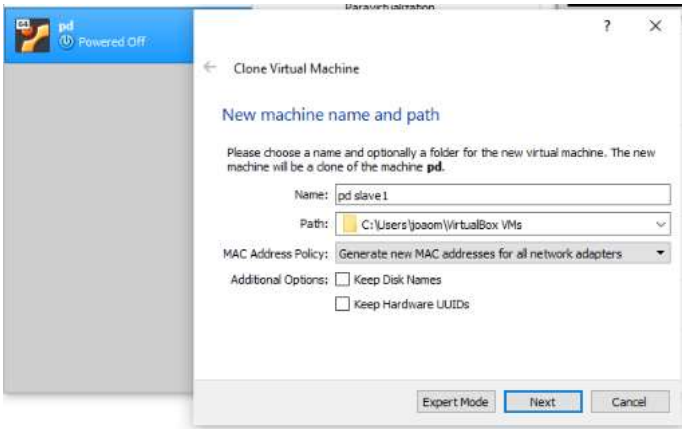


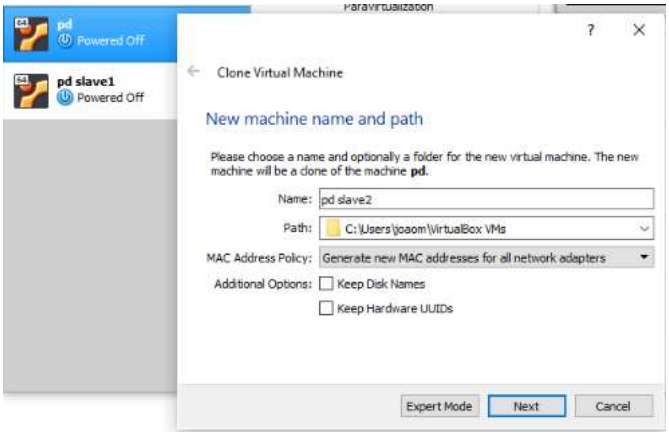

A spark cluster has a single Master and any number of Slaves/Workers. The driver and the executors run their individual Java processes and users can run them on the same horizontal spark cluster or on separate machines.

## Pre-requirements

- Ubuntu 18.04 installed on a virtual machine.

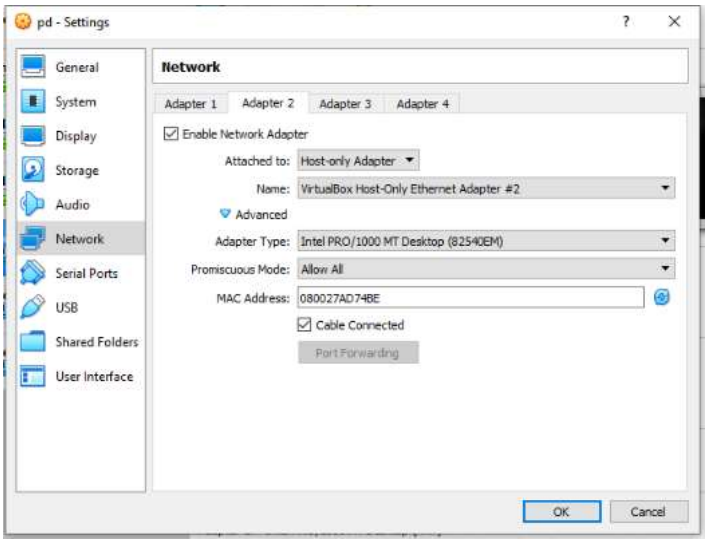
## 1st Step:




**2nd Step:**

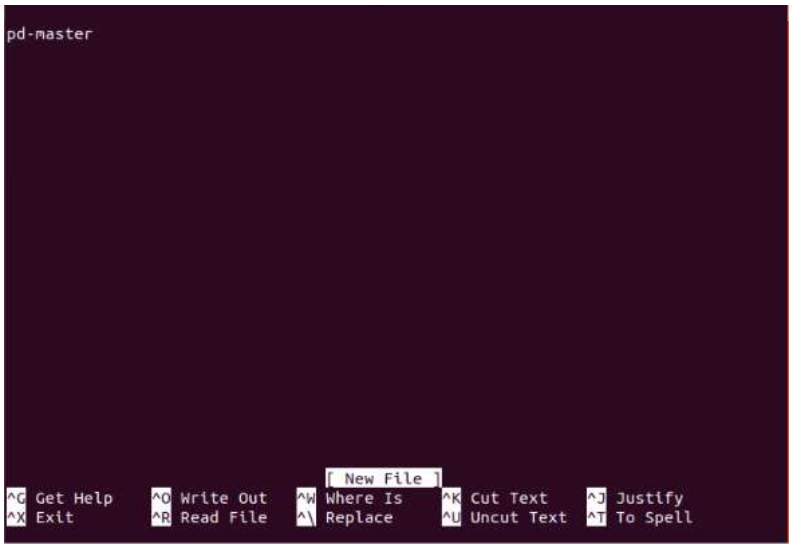
Make sure all the VM's have the following network configuration on Apapter 2:

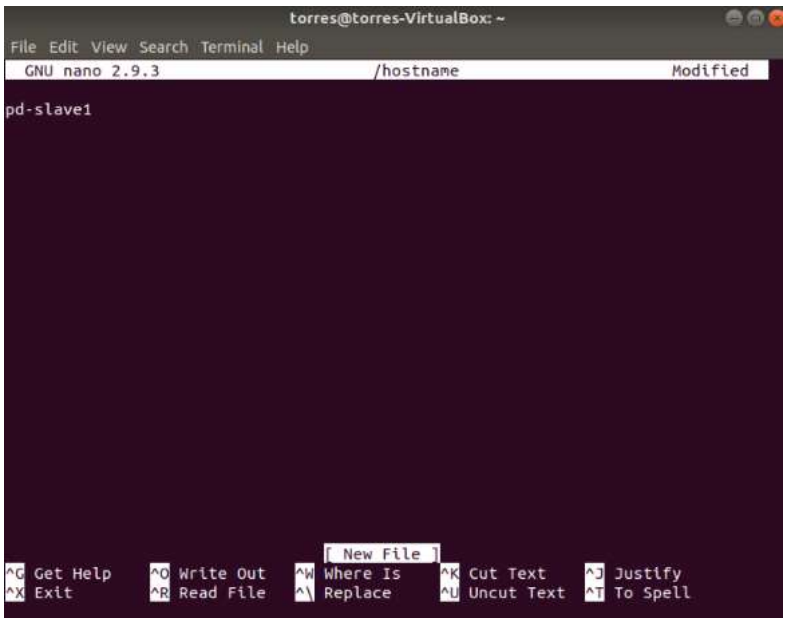


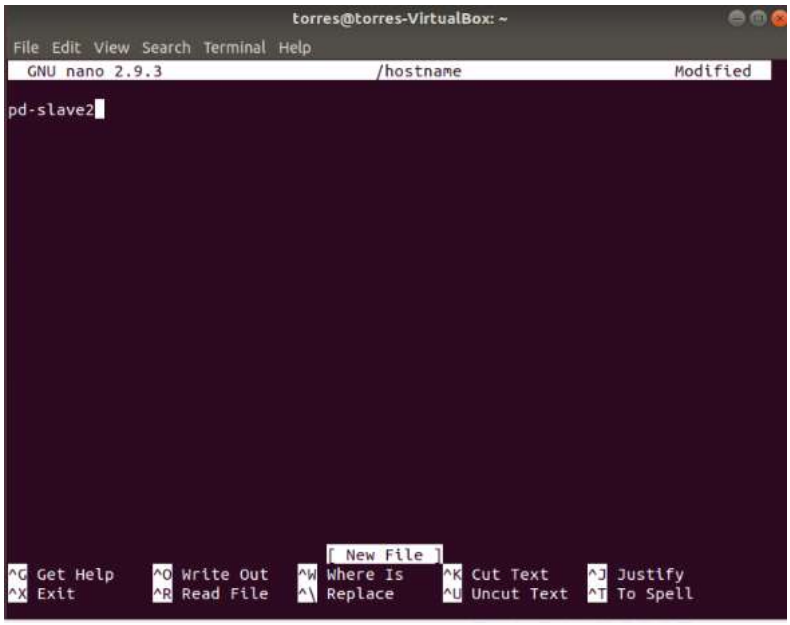

**3rd Step:**

Let's change the hostname on each virtual machine. Open the file and type the name of the machina. Use this command:

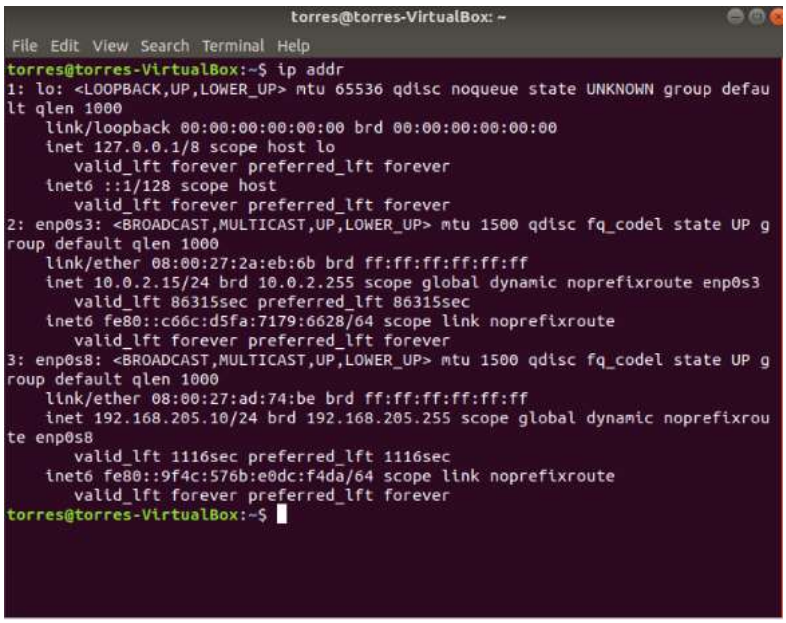
```
sudo nano /hostname
```




**4th Step:**




This is on the master VM, as you can see our IP is 192.168.205.10. For you this will be different.

This means that our IP's are:

master: 192.168.205.10

slave1: 192.168.205.11

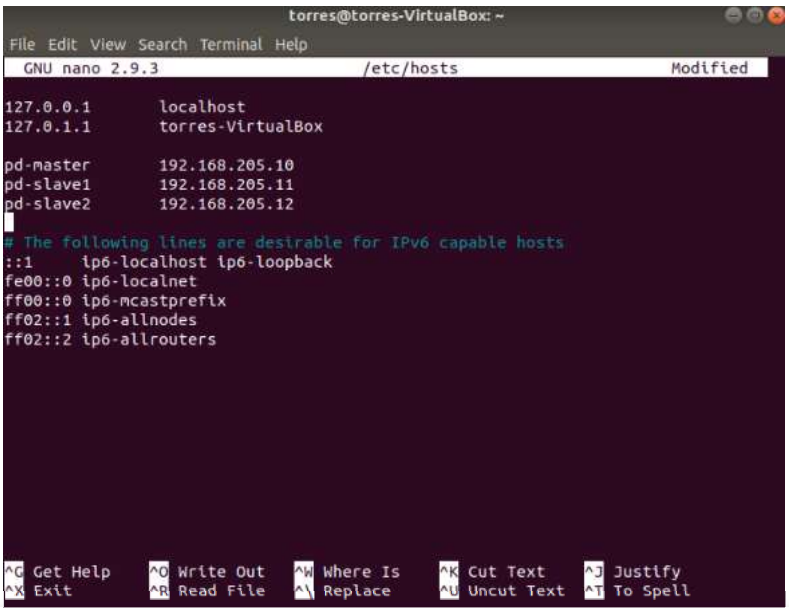
slave2: 192.168.205.12

### 5th Step:

We need to edit the **hosts** file. Use the following command:

```
sudo nano /etc/hosts
```

and add your network information:



```
sudo reboot
```

### 7th Step:

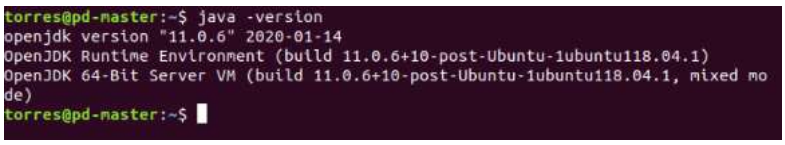
Do this step on all the Machines, master and slaves.

Now, in order to install Java we need to do some things. Follow these commands and give permission when needed:

```
$ sudo apt-get install software_properties_common
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install openjdk-11-jdk
```

To check if java is installed, run the following command.

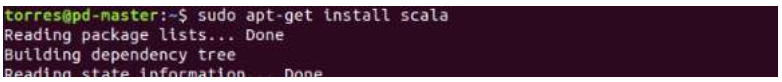
```
$ java -version
```




### 8th Step:

Now let's install Scala on the master and the slaves. Use this command:

```
$ sudo apt-get install scala
```




To check if Scala was correctly installed run this command:

```
$ scala -version
```


As you can see, Scala version 2.11.12 is now installed on my machine.

**9th Step:**

We will configure SSH, but this step in on master only.

We need to install the **Open SSH Server-Client**, use the command:

```
$ sudo apt-get install openssh-server openssh-client
```

```
torres@pd-master:~$ sudo apt-get install openssh-server openssh-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  keychain libpam-ssh monkeysphere ssh-askpass molly-guard rssh
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
The following packages will be upgraded:
  openssh-client
1 upgraded, 4 newly installed, 0 to remove and 466 not upgraded.
Need to get 637 kB/1251 kB of archives.
After this operation, 5316 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Now generate key pairs:

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

```
torres@pd-master:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/torres/.ssh/id_rsa):
Your identification has been saved in /home/torres/.ssh/id_rsa.
Your public key has been saved in /home/torres/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:OZttQJwxh/PurAbQBaXyTms7nhzAAYor9vfB9lNY6uw torres@pd-master
The key's randomart image is:
+---[RSA 2048]----+
|  .   .o+..      |
|.. .   o+=       |
|o   o...+o       |
| . ..+.. ...     |
|o.  o.o S.+      |
|o .  +o. O..     |
|   . .===o+      |
|    .o+++=o      |
|     .=+oE.      |
+----[SHA256]-----+
torres@pd-master:~$
```

Use the following command in order to make this key an authorized one:

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

```
torres@pd-master:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
torres@pd-master:~$
```

```
ssh-copy-id user@pd-master
ssh-copy-id user@pd-slave1
ssh-copy-id user@pd-slave2
```

```
torres@pd-master:~$ ssh-copy-id torres@pd-slave1
The authenticity of host 'pd-slave1 (192.168.205.11)' can't be established.
ECDSA key fingerprint is SHA256:mG56/CnM1qav/9cS7HP8hXNNOEYzzol/7JLKq4cTLbE.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
torres@pd-slave1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'torres@pd-slave1'"
and check to make sure that only the key(s) you wanted were added.

torres@pd-master:~$
```

```
torres@pd-master:~$ ssh-copy-id torres@pd-slave2
The authenticity of host 'pd-slave2 (192.168.205.12)' can't be established.
ECDSA key fingerprint is SHA256:vchan4ALBXytJSnv+3L+nChJiQ1GmER3bWpXnMPy4k0.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
torres@pd-slave2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'torres@pd-slave2'"
and check to make sure that only the key(s) you wanted were added.

torres@pd-master:~$
```


```
torres@pd-master:~$ ssh-copy-id torres@pd-slave2
The authenticity of host 'pd-slave2 (192.168.205.12)' can't be established.
ECDSA key fingerprint is SHA256:vchan4ALBXytJSnv+3L+nChJiQ1GmER3bWpXnMPy4k0.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
torres@pd-slave2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'torres@pd-slave2'"
and check to make sure that only the key(s) you wanted were added.
```


Let's check if everything went well, try to connect to the slaves:

```
$ ssh slave01
$ ssh slave02
```

```
torres@pd-master:~$ ssh pd-slave1
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.18.0-15-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

463 packages can be updated.
250 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
torres@pd-slave1:~$
```

```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

463 packages can be updated.
250 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
torres@pd-slave2:~$
```

As you can see everything went well, to exit just type the command:

```
exit
```

```
torres@pd-slave2:~$ exit
logout
Connection to pd-slave2 closed.
```

### 10th Step:

Now we Download the latest version of Apache Spark.

**NOTE: Everything inside this step must be done on all the virtual machines.**

Use the following command :

```
$ wget http://www-us.apache.org/dist/spark/spark-2.4.4/spark-2.4.4-bin-hadoop2.7.tgz
```

This is the most recent version as of the writing of this arcticle, it might have changed if you try it later. Anyway, I think you'll still be good using this one.

```
torres@pd-master:~$ wget http://www-us.apache.org/dist/spark/spark-2.4.4/spark-
2.4.4-bin-hadoop2.7.tgz
--2020-02-03 19:41:41--  http://www-us.apache.org/dist/spark/spark-2.4.4/spark-
2.4.4-bin-hadoop2.7.tgz
Resolving www-us.apache.org (www-us.apache.org)... 40.79.78.1
Connecting to www-us.apache.org (www-us.apache.org)|40.79.78.1|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 230091034 (219M) [application/x-gzip]
Saving to: 'spark-2.4.4-bin-hadoop2.7.tgz'

.4.4-bin-hadoop2.7.   0%[                    ] 856,50K   231KB/s    eta 16m 45s
```

**Extract the Apache Spark file you just downloaded**

Use the following command to extract the Spark tar file:

```
$ tar xvf spark-2.4.4-bin-hadoop2.7.tgz
```

```
spark-2.4.4-bin-hadoop2.7/R/lib/sparkr.zip
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/INDEX
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/html/
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/html/R.css
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/html/00Index.html
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/help/
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/help/aliases.rds
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/help/AnIndex
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/help/SparkR.rdx
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/help/SparkR.rdb
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/help/paths.rds
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/worker/
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/worker/worker.R
spark-2.4.4-bin-hadoop2.7/R/lib/SparkR/worker/daemon.R
```

**Move Apache Spark software files**

Use the following command to move the spark software files to respective directory (*/usr/local/bin*)

```
$ sudo mv spark-2.4.4-bin-hadoop2.7 /usr/local/spark
```

```
torres@pd-master:~$ sudo mv spark-2.4.4-bin-hadoop2.7 /usr/local/spark
[sudo] password for torres:
torres@pd-master:~$
```

**Set up the environment for Apache Spark**

Edit the *bashrc* file using this command:

```
$ sudo gedit~/.bashrc
```

```
torres@pd-master:~$ sudo gedit ~/.bashrc
```

Add the following line to the file. This adds the location where the spark software file are located to the PATH variable.

```
export PATH = $PATH:/usr/local/spark/bin
```

```
Open ▾ *.bashrc ~/ Save

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal ||
echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]
\s*alert$//'\'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export PATH= $PATH:/usr/local/spark/bin
```

**Note:** this screenshot has a mistake, when you're doing this don't leave a space like I did. Just write "PATH=$PATH".

```
torres@pd-master:~$ source ~/.bashrc
torres@pd-master:~$
```

### 11th Step:

Apache Spark Master Configuration (do this step on the Master VM only)

**Edit spark-env.sh**

Move to spark *conf* folder and create a copy of the template of *spark-env.sh* and rename it.

```
$ cd /usr/local/spark/conf
$ cp spark-env.sh.template spark-env.sh
```

```
torres@pd-master:/usr/local/spark/conf$ cp spark-env.sh.template spark-env.sh
torres@pd-master:/usr/local/spark/conf$
```

Now edit the configuration file *spark-env.sh.*

```
$ sudo vim spark-env.sh
```

And add the following parameters:

```
export SPARK_MASTER_HOST='<MASTER-IP>'export JAVA_HOME=<Path_of_JAVA_installation>
```

**Add Workers**

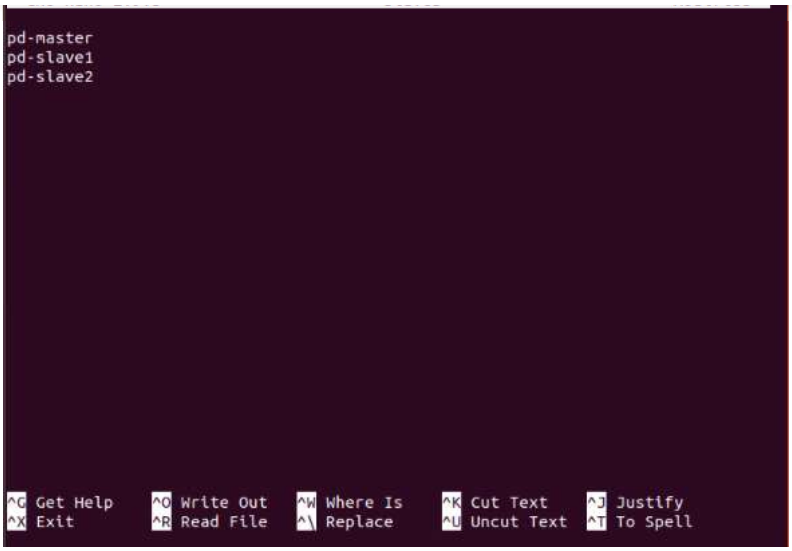
Edit the configuration file *slaves* in (*/usr/local/spark/conf*).

```
$ sudo nano slaves
```

```
torres@pd-master:/usr/local/spark/conf$ sudo nano slaves
```

And add the following entries.

```
pd-master
pd-slave01
pd-slave02
```




### 12th Step:

**Let's try to start our Apache Spark Cluster, hopefully everything is ok!**

To start the spark cluster, run the following command on master.:

```
$ cd /usr/local/spark
$ ./sbin/start-all.sh
```



I won't stop it, but in case you want to stop the cluster, this is the command:

```
$ ./sbin/stop-all.sh
```

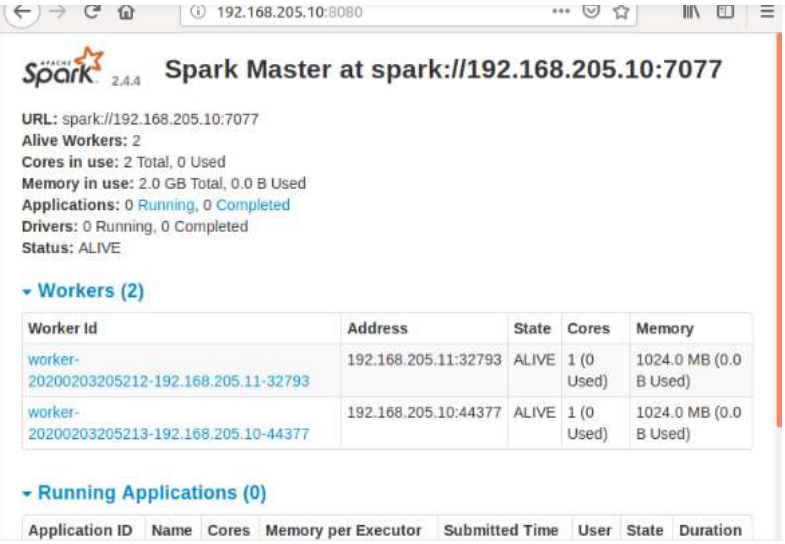
### 13th Step:

To check if the services started we use the command:

```
$ jps
```


### 14th Step:

Browse the Spark UI to know about your cluster. To do this, go to your browser and type:




As you can see we have 2 Alive Workers, our slaves, which means it's all done!

**Final considerations:**

Hopefully you managed to successfully follow this tutorial and have a perfectly working Apache Spark Cluster.

Any doubts feel free to ask me.

See you later!