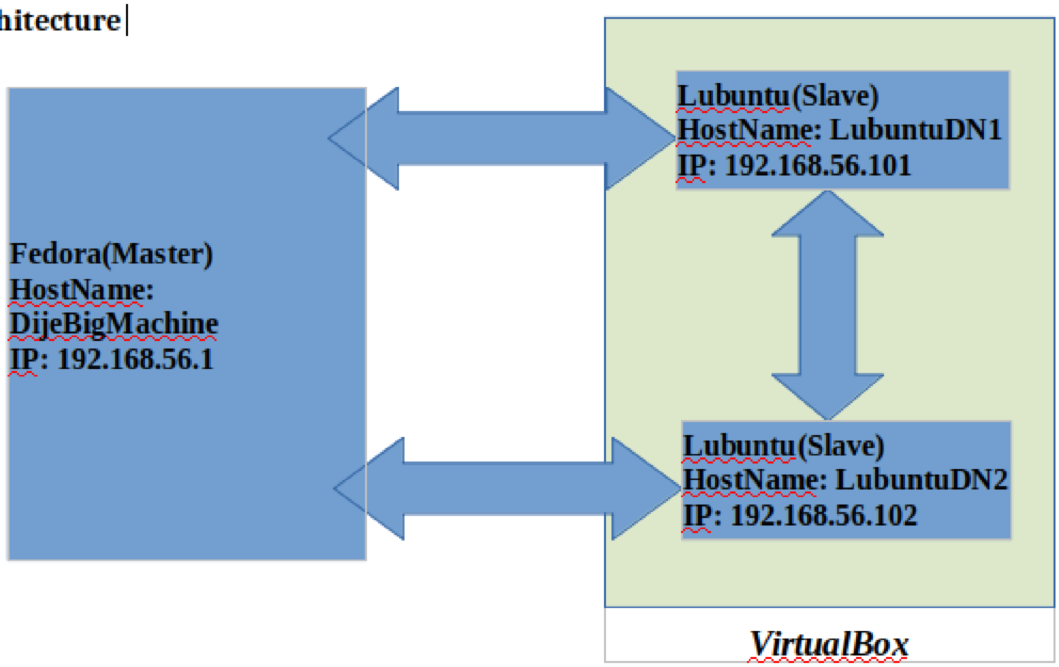
How to create a multinode Hadoop, Spark and Cassandra culster on a Laptop

This guide provides step by step process for setting up a three node Hadoop, Spark and Cassandra cluster on a laptop for learning. This cluster is made up of one host machine and two virtual machines hosted on Oracle VirtualBox.

Operating Systems:
1. Fedora 30 desktop on the laptop for the Spark master
2. Light Ubuntu for 2 Spark slave nodes(lesser machines)

Softwares used:
1. Java 1.8
2. Oracle VirtualBox 5.2
3. Hadoop from Apache foundation
4. Spark 2.3 from Apache foundation
5. Cassandra 3.11.5 from Apache foundation

Architecture



Preparation:

1. Java: Download Java (https://www.java.com/en/download/) and place it at on your har sdrive. Setup .bashrc with the following environment variables.

```
#JAVA HOME
export JAVA_HOME=/home/dijender/Java/jdk1.8.0_14
export CLASSPATH=$JAVA_HOME
export PATH=$PATH:$JAVA_HOME/bin
```

Save .bashrc and source it using the following command from your home directory.

```
source .bashrc
```

To verify if Java is setup and setup with the correct version issue the following command.

```
[dijender@DijeBigMachine ~]$ java -version
java version "1.8.0_141"
Java(TM) SE Runtime Environment (build 1.8.0_141-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
```

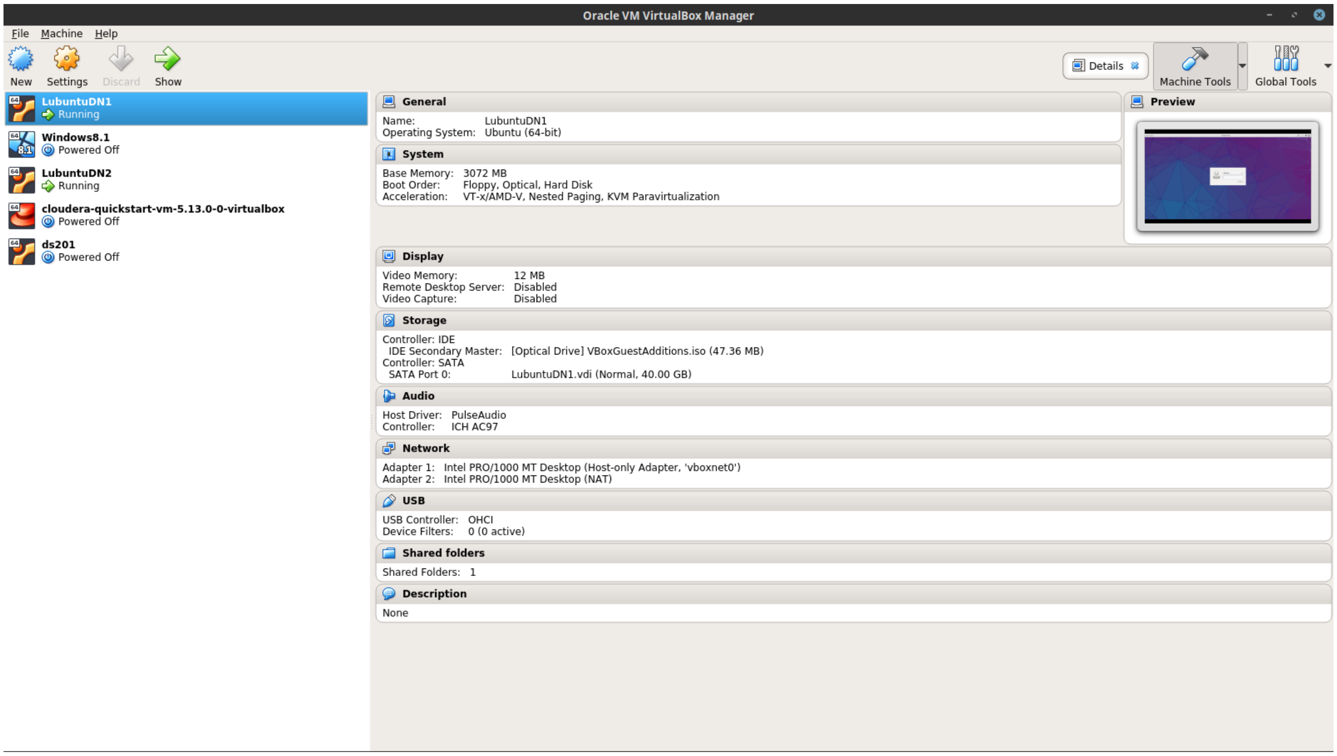
So now our Java is setup.

2. VirtualBox: VirtualBox is a visualization software freely distributed by Oracle. Install it using the instructions given on the VirtualBox website (https://www.virtualbox.org/)

After successful installation, open it see if everything is setup correctly.

3. Light Ubuntu on VirtualBox: OS selected for nodes on the VirtualBox was chosen on purpose to be light as we are only operating with 16 GB of total RAM. The process to install Lubuntu is straight forward. Download the .iso file for the Lubuntu you want to install and then on the VirtualBox, select the .iso file to start installing. Please remember to set up a user and remember the password for that user.

Below are the details for of the Virtual Machine. Don't worry about the network setting, we'll discuss that later in a different section.



LubuntuDN1 — Running
Windows8.1 — Powered Off
LubuntuDN2 — Running
cloudera-quickstart-vm-5.13.0-0-virtualbox — Powered Off
ds201 — Powered Off

General
Name: LubuntuDN1
Operating System: Ubuntu (64-bit)

System
Base Memory: 3072 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display
Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] VBoxGuestAdditions.iso (47.36 MB)
Controller: SATA
SATA Port 0: LubuntuDN1.vdi (Normal, 40.00 GB)

Audio
Host Driver: PulseAudio
Controller: ICH AC97

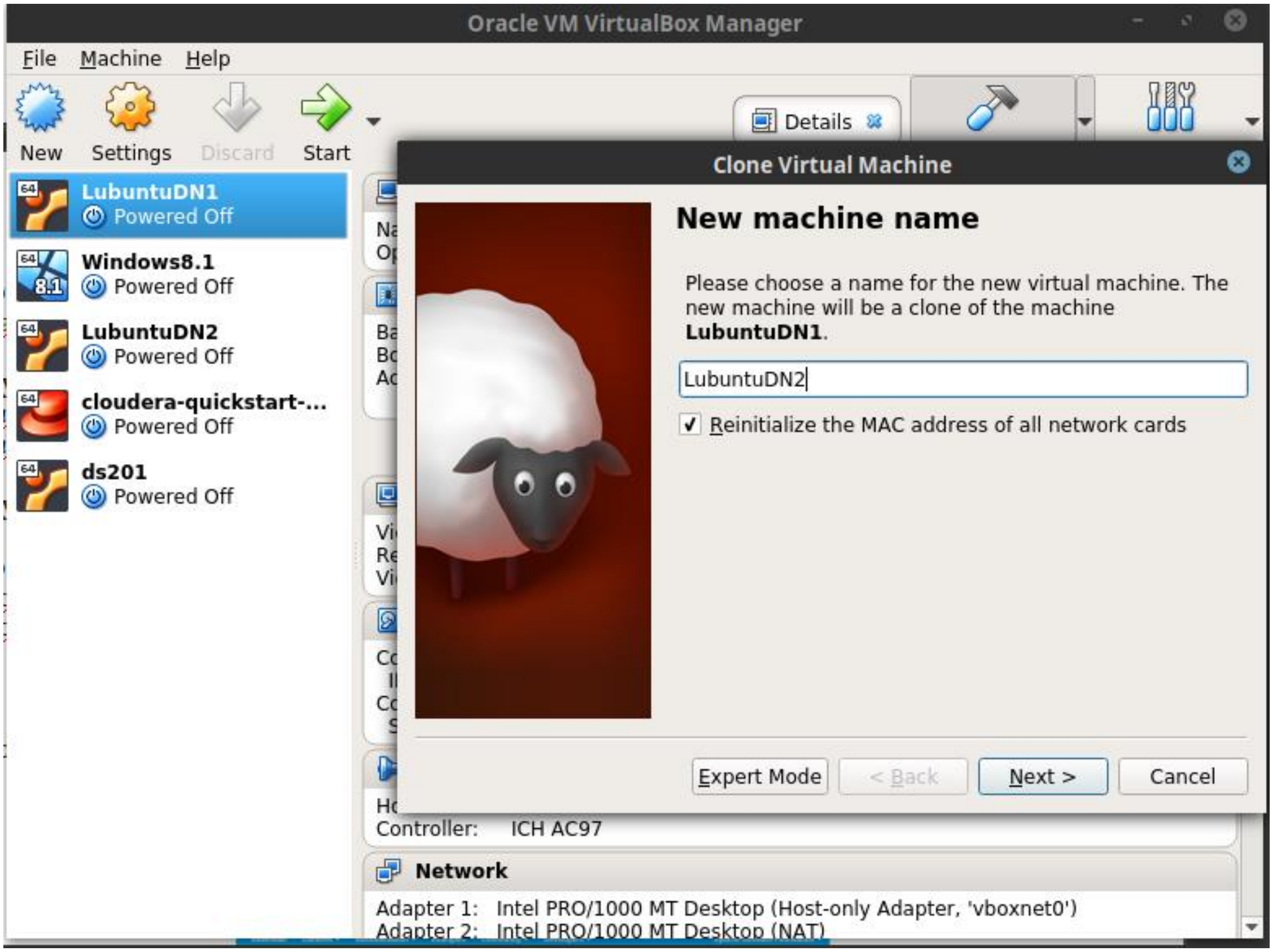
Network
Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'vboxnet0')
Adapter 2: Intel PRO/1000 MT Desktop (NAT)

USB
USB Controller: OHCI
Device Filters: 0 (0 active)

Shared folders
Shared Folders: 1

Description
None

4. Clone the VM to create another VM with the same configuration. See image below.



New machine name

Please choose a name for the new virtual machine. The new machine will be a clone of the machine **LubuntuDN1**.

LubuntuDN2

☑ Reinitialize the MAC address of all network cards

Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'vboxnet0')
Adapter 2: Intel PRO/1000 MT Desktop (NAT)

5. Setup Host names for all three machines:

```
hostnamectl set-hostname <HostName>
```

Host Fedora Machine name: DijeBigMachine
Virtual Machine1: LubuntuDN1
Virtual Machine2: LubuntuDN2

6. Network setting: This is one the most crucial step in setting up a cluster. There are few considerations that we need to make while creating our cluster.

1. All machines should be able to communicate with each other i.e. all machine should have two way communication enabled
2. All communication should be free of password. As all nodes need to communication continuously, machines needs to trust each other. We'll see that in the section **"Setting up trusted machines in a network".**
3. IP addresses can be changed everytime we start a machine and for virtual machines the IP addresses with be picked from a pool and machines can be allocated different ones depending on the order in which they are fired up. We'll setup fixed IP s for all our machines. We'll see that in the section **"Setting up fixed IP addresses".**
4. After the IP addresses have been fixed we need to update */etc/hosts* file in each machine with the fixed IP addresses.

```
[dijender@DijeBigMachine ~]$ cat /etc/hosts
127.0.0.1		localhost.localdomain localhost
192.168.56.1 DijeBigMachine
192.168.56.101 LubuntuDN1
192.168.56.102 LubuntuDN2
```

Setting up hostnames and fixed IP on trusted machines

1. Setting up fixed IP on the Host machine(DijeBigMachine)

First find the network name of your computer by using the command ifconfig.

```
[dijender@DijeBigMachine ~]$ ifconfig
eno1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      ether d4:be:d9:60:d5:0f  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
      device interrupt 20  memory 0xf7e00000-f7e20000
```

As seen above, the name of the network is **eno1**. Edit /etc/sysconfig/network-scripts/ifcfg-eno1 to put the static IP address 192.168.56.1 in it.

```
[dijender@DijeBigMachine ~]$ cat /etc/sysconfig/network-scripts/ifcfg-eno1
HWADDR=D4:BE:D9:60:D5:0F
TYPE=Ethernet
```

```
BOOTPROTO=static
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=eno1
UUID=8d557efd-839e-418d-8be8-62f5a5cf5014
IPADDR=192.168.56.1
NETMASK=255.255.255.0
BROADCAST=192.168.56.255
NETWORK=192.168.56.0
ONBOOT=yes
```

Restart your machine.

2. Setting up two way communication on the virtual machines(LubuntuDN1 and LubuntuDN2).

On VirtualBox application -
Go to File->Preference->Network->Host-Only Adapter(tab)
add a Host-Only Adapter
Don't enable DHCP server

On VM -
Settings->Network
Adapter 1 - Host-Only network(Select the Host-only adapter as seen above)
Adapter 2 – NAT

Start the VM
See the Adapter names using the commands --
ls /sys/class/net
ifconfig -all

Make sure which one is Host-only and which one is NAT.

I got
enp0s3 enp0s8 lo

enp0s3 -> Host only
enp0s8 -> NAT

Now edit /etc/network/interfaces and it should look like this with static IP set as 192.168.56.101

```
-----------------------------------------------------------
dijender@LubuntuDN1:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
```

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

#Host-only interface
auto enp0s3
iface enp0s3 inet static
        address 192.168.56.101
        netmask 255.255.255.0
        network 192.168.56.0
        broadcast 192.168.56.255

#NAT interface
auto enp0s8
iface enp0s8 inet dhcp
```

restart networking busing command - /etc/init.d/networking restart

Setting up trusted machines in a network

1. Generate public using command - ssh-keygen
2. Copy the generated key in cat ~/.ssh/id_rsa.pub
3. Paste it in the .ssh/authorized_keys file on the machine you want to connect to.
4. This needs to be done between all pair of machines.
5. After machines have shared the public key, you should be able to login using ssh to each other without the password as seen below.

```
[dijender@DijeBigMachine ~]$ ssh LubuntuDN1
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

123 packages can be updated.
91 updates are security updates.

Last login: Sat Jun 24 17:37:13 2017 from 192.168.56.102b
dijender@LubuntuDN1:~$
```

For details refer to https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server

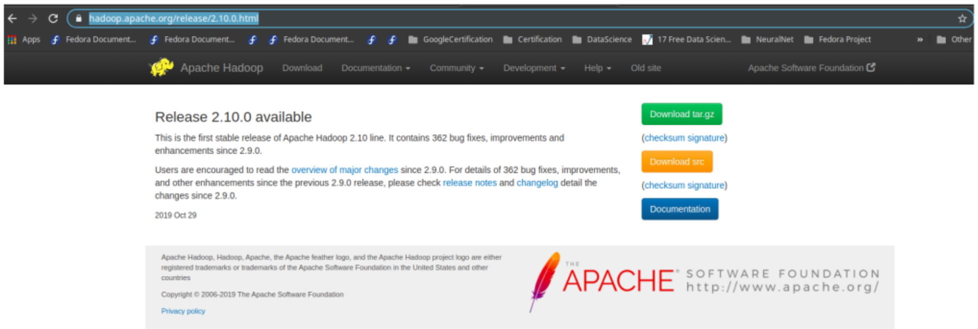
After this, our machines are all setup in one network and we are ready to install our software on them.

Installation

For installation of all softwares, I have a separate folder which I use to keep all related softwares. I call it /home2/BigData

1. Hadoop

We start with installation of Hadoop. This includes HDFS and yarn. Download latest Hadoop distribution from Apache foundation(https://hadoop.apache.org/release/2.10.0.html) and untar it at a location on your harddrive.



Release 2.10.0 available

This is the first stable release of Apache Hadoop 2.10 line. It contains 362 bug fixes, improvements and enhancements since 2.9.0.

Users are encouraged to read the overview of major changes since 2.9.0. For details of 362 bug fixes, improvements, and other enhancements since the previous 2.9.0 release, please check release notes and changelog detail the changes since 2.9.0.

2019 Oct 29

Apache Hadoop, Hadoop, Apache, the Apache feather logo, and the Apache Hadoop project logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and other countries

Copyright © 2006-2019 The Apache Software Foundation

Privacy policy

I place it at */home2/BigData/Hadoop*.

I also create */home2/BigData/hadoop_store/hdfs/namenode* and */home2/BigData/hadoop_store/hdfs/datanode* folders for name node and data node files. These values go in the configuration file hdfs-site.xml.

Update .bashrc as shown below

```
# HADOOP VARIABLES START
export HADOOP_INSTALL=/home2/BigData/hadoop
export HADOOP_HOME=$HADOOP_INSTALL
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native"
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/common/*
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/common/lib/*
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/mapreduce/*:$HADOOP_HOME/share/hadoop/mapreduce/lib/*
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/yarn/*:$HADOOP_HOME/share/hadoop/yarn/lib/*
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/hdfs/*:$HADOOP_HOME/share/hadoop/hdfs/lib/*
```

Source .bashrc by running

```
[dijender@DijeBigMachine ~]$ source .bashrc
```

Update core-site.xml at the location $HADOOP_HOME/etc/hadoop and set

```
 <property>
      <name>fs.defaultFS</name>
      <value>hdfs://DijeBigMachine:9000</value>
   </property>
```

Update hdfs-site.xml at $HADOOP_HOME/etc/hadoop and set the following properties.

1. Replication factor:

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

2. Namenode location:

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/home2/BigData/hadoop_store/hdfs/namenode</value>
 </property>
```

3. Datanode location:

```
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/home2/BigData/hadoop_store/hdfs/datanode</value>
 </property>
```

Create a file called workers at $HADOOP_HOME/etc/hadoop with name of the nodes.

```
[dijender@DijeBigMachine hadoop]$ cat workers
DijeBigMachine
```

```
LubuntuDN1
LubuntuDN2
```

Source the .bashrc file and test if you can start hadoop by running start-dfs.sh followed by start-yarn.sh

If you don't see any errors, try checking the hadoop file system by running

```
hadoop fs -mkdir /testDir
hadoop fs -ls /
```

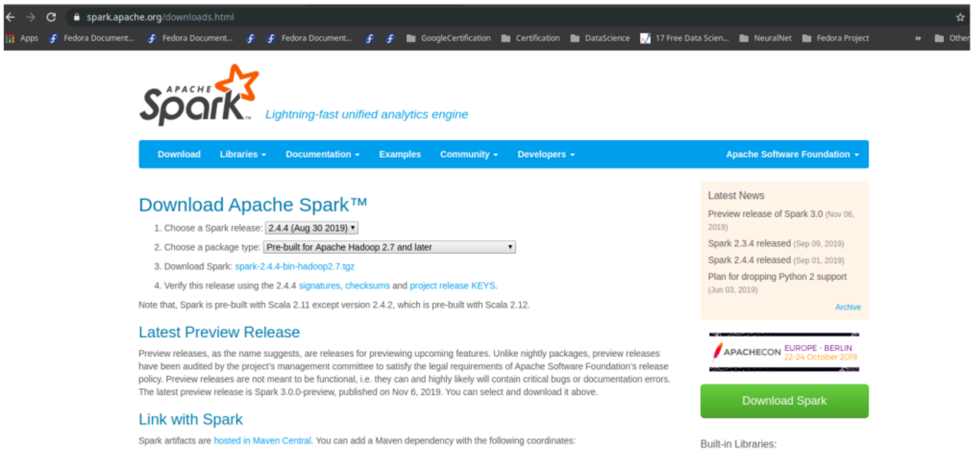
If you see /testDir created. Your Hadoop is correctly setup.

To stop Hadoop, run the following

```
stop-dfs.sh
stop-yarn.sh
```

2. Spark

Download Spark from Apache foundation as seen in image below.



Download Apache Spark™

1. Choose a Spark release: 2.4.4 (Aug 30 2019)
2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later
3. Download Spark: spark-2.4.4-bin-hadoop2.7.tgz
4. Verify this release using the 2.4.4 signatures, checksums and project release KEYS.

Note that, Spark is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12.

Latest Preview Release

Preview releases, as the name suggests, are releases for previewing upcoming features. Unlike nightly packages, preview releases have been audited by the project's management committee to satisfy the legal requirements of Apache Software Foundation's release policy. Preview releases are not meant to be functional, i.e. they can and highly likely will contain critical bugs or documentation errors. The latest preview release is Spark 3.0.0-preview, published on Nov 6, 2019. You can select and download it above.

Link with Spark

Spark artifacts are hosted in Maven Central. You can add a Maven dependency with the following coordinates:

Latest News

Preview release of Spark 3.0 (Nov 06, 2019)

Spark 2.3.4 released (Sep 09, 2019)

Spark 2.4.4 released (Sep 01, 2019)

Plan for dropping Python 2 support (Jun 03, 2019)

Archive

Built-in Libraries:

Un-tar it at a location in your hard drive. I untar it at /home2/Bigdata/spark.

Set up .bashrc as follows

```
#Add Spark path
export SPARK_HOME=/home2/BigData/spark
export SPARK_CONF_DIR=$SPARK_HOME/conf
export SPARK_JAR=$SPARK_HOME/jars/*
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Create master and slaves files and put master and slaves host names in them.

```
[dijender@DijeBigMachine conf]$ pwd
```

```
/home2/BigData/spark/conf
[dijender@DijeBigMachine conf]$ cat master
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# A Spark Master will be started on each of the machines listed below.
DijeBigMachine

[dijender@DijeBigMachine conf]$ cat slaves
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# A Spark Worker will be started on each of the machines listed below.
DijeBigMachine
LubuntuDN1
LubuntuDN2
```

I keep everything else as default in the configuration file.s For testing and using purpose it works good. Feel free to fiddle with the values in the config file and see the impact.

Verify the installation

You don’t need to run anything on the worker nodes. We will start Hadoop and Spark on the master node.

1. Fire up both the Lubuntu VMs.
2. On the master machine(Fedora), run the following commands to start HDFS and Yarn.

```
[dijender@DijeBigMachine ~]$ start-dfs.sh
Starting namenodes on [DijeBigMachine]
Starting datanodes
Starting secondary namenodes [DijeBigMachine]
[dijender@DijeBigMachine ~]$
[dijender@DijeBigMachine ~]$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
[dijender@DijeBigMachine ~]$
[dijender@DijeBigMachine hadoop]$ hdfs dfsadmin -report
Configured Capacity: 570065395712 (530.91 GB)
Present Capacity: 186425126912 (173.62 GB)
DFS Remaining: 180737851392 (168.33 GB)
DFS Used: 5687275520 (5.30 GB)
DFS Used%: 3.05%
Replicated Blocks:
        Under replicated blocks: 0
        Blocks with corrupt replicas: 0
        Missing blocks: 0
        Missing blocks (with replication factor 1): 0
        Pending deletion blocks: 0
Erasure Coded Block Groups:
        Low redundancy block groups: 0
        Block groups with corrupt internal blocks: 0
        Missing block groups: 0
        Pending deletion blocks: 0

-------------------------------------------------
Live datanodes (3):

Name: 192.168.56.101:9866 (LubuntuDN1)
Hostname: LubuntuDN1
Decommission Status : Normal
Configured Capacity: 38970851328 (36.29 GB)
DFS Used: 1895112704 (1.76 GB)
Non DFS Used: 7607930880 (7.09 GB)
DFS Remaining: 27464609792 (25.58 GB)
DFS Used%: 4.86%
DFS Remaining%: 70.47%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Nov 14 12:15:59 IST 2019
Last Block Report: Thu Nov 14 11:57:32 IST 2019
```

```
Num of Blocks: 51


Name: 192.168.56.102:9866 (LubuntuDN2)
Hostname: LubuntuDN2
Decommission Status : Normal
Configured Capacity: 38970851328 (36.29 GB)
DFS Used: 1895112704 (1.76 GB)
Non DFS Used: 8426008576 (7.85 GB)
DFS Remaining: 26646532096 (24.82 GB)
DFS Used%: 4.86%
DFS Remaining%: 68.38%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Nov 14 12:16:00 IST 2019
Last Block Report: Thu Nov 14 11:57:33 IST 2019
Num of Blocks: 51


Name: 192.168.56.1:9866 (DijeBigMachine)
Hostname: DijeBigMachine
Decommission Status : Normal
Configured Capacity: 492123693056 (458.33 GB)
DFS Used: 1897050112 (1.77 GB)
Non DFS Used: 338594594816 (315.34 GB)
DFS Remaining: 126626709504 (117.93 GB)
DFS Used%: 0.39%
DFS Remaining%: 25.73%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Nov 14 12:15:58 IST 2019
Last Block Report: Thu Nov 14 11:56:53 IST 2019
Num of Blocks: 51
```

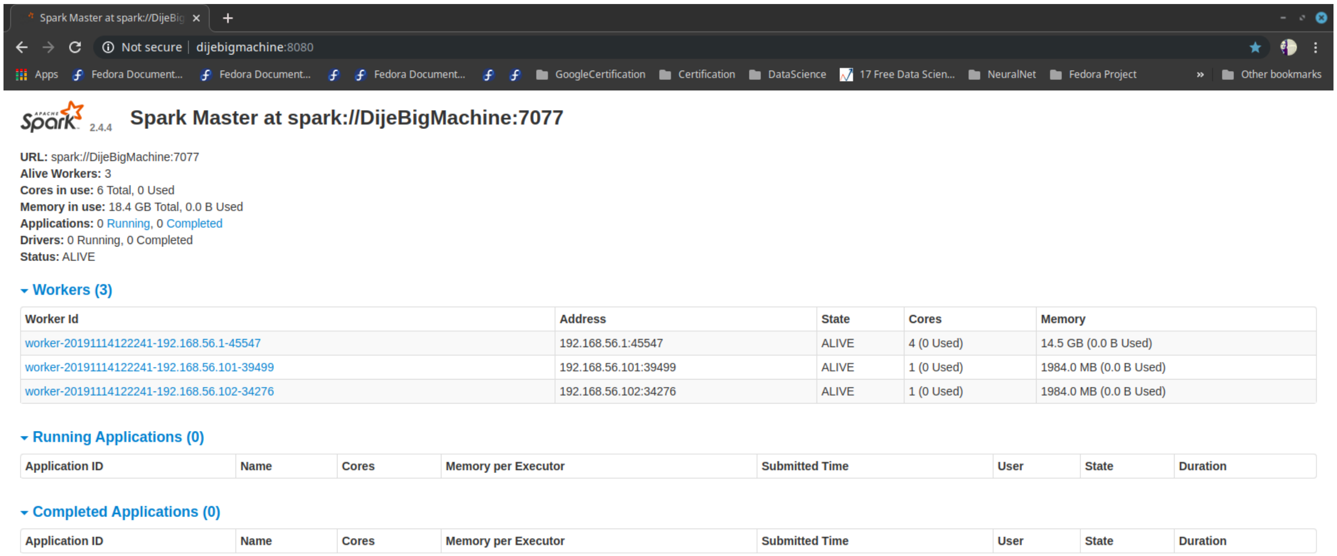
We can see we have 3 datanodes live.

Start spark using the following commands on the Master machine. To start the master and workers.

```
[dijender@DijeBigMachine ~]$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /home2/BigData/spark/logs/spark-dijender-org.apache.spark.deploy.master.Master-1-DijeBigMachine.out
[dijender@DijeBigMachine ~]$ start-slaves.sh
LubuntuDN1: starting org.apache.spark.deploy.worker.Worker, logging to /home2/BigData/spark/logs/spark-dijender-org.apache.spark.deploy.worker.Worker-1-LubuntuDN1.out
LubuntuDN2: starting org.apache.spark.deploy.worker.Worker, logging to /home2/BigData/spark/logs/spark-dijender-org.apache.spark.deploy.worker.Worker-1-LubuntuDN2.out
DijeBigMachine: starting org.apache.spark.deploy.worker.Worker, logging to /home2/BigData/spark/logs/spark-dijender-org.apache.spark.deploy.worker.Worker-1-DijeBigMachine.out
```

```
[dijender@DijeBigMachine ~]$ jps
77953 ResourceManager
77313 SecondaryNameNode
78112 NodeManager
76692 NameNode
81462 Master
81627 Worker
76909 DataNode
81695 Jps
[dijender@DijeBigMachine ~]$
```

Verify Spark is installed correctly in the browser.



Spark Master at spark://DijeBigMachine:7077

URL: spark://DijeBigMachine:7077
Alive Workers: 3
Cores in use: 6 Total, 0 Used
Memory in use: 18.4 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory
worker-20191114122241-192.168.56.1-45547	192.168.56.1:45547	ALIVE	4 (0 Used)	14.5 GB (0.0 B Used)
worker-20191114122241-192.168.56.101-39499	192.168.56.101:39499	ALIVE	1 (0 Used)	1984.0 MB (0.0 B Used)
worker-20191114122241-192.168.56.102-34276	192.168.56.102:34276	ALIVE	1 (0 Used)	1984.0 MB (0.0 B Used)

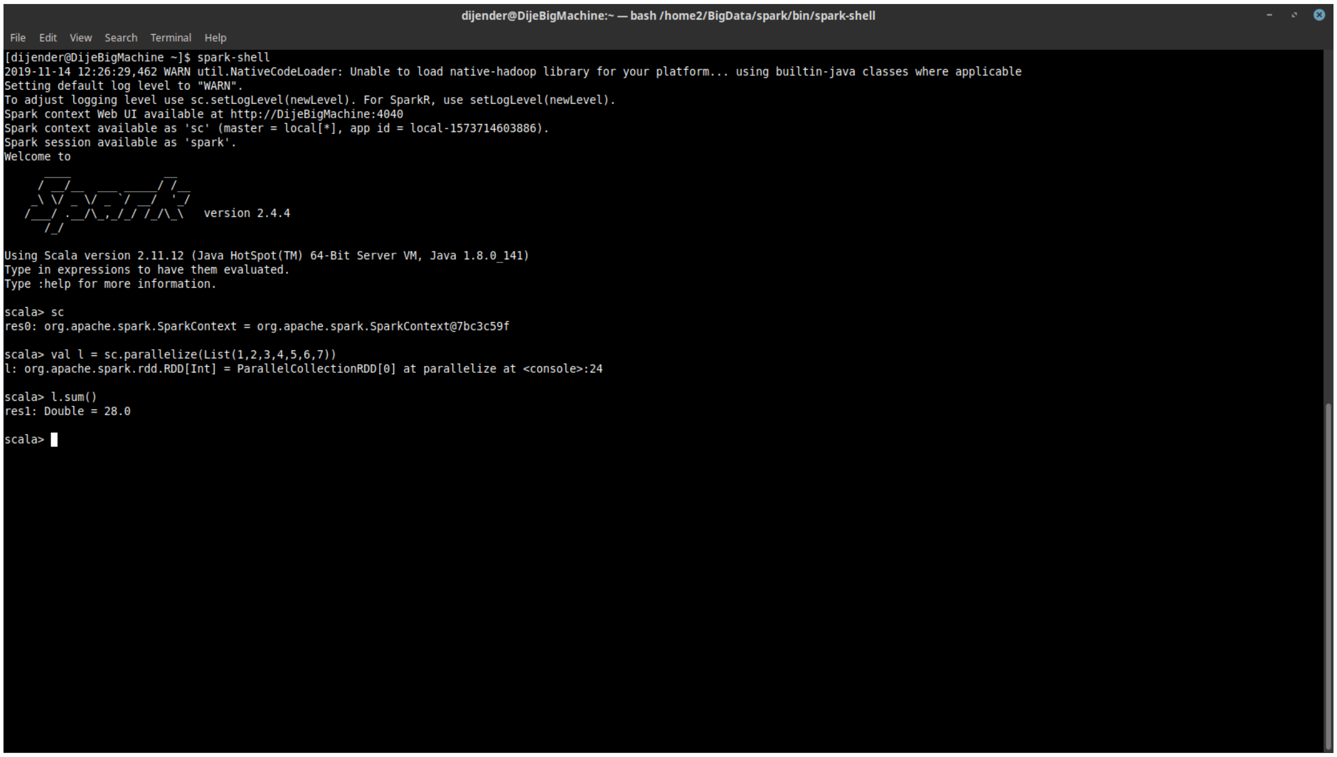
Running Applications (0)

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

Completed Applications (0)

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

Running Spark code using spark-shell to verify that installation is working fine. Run spark-shell on the command prompt.b

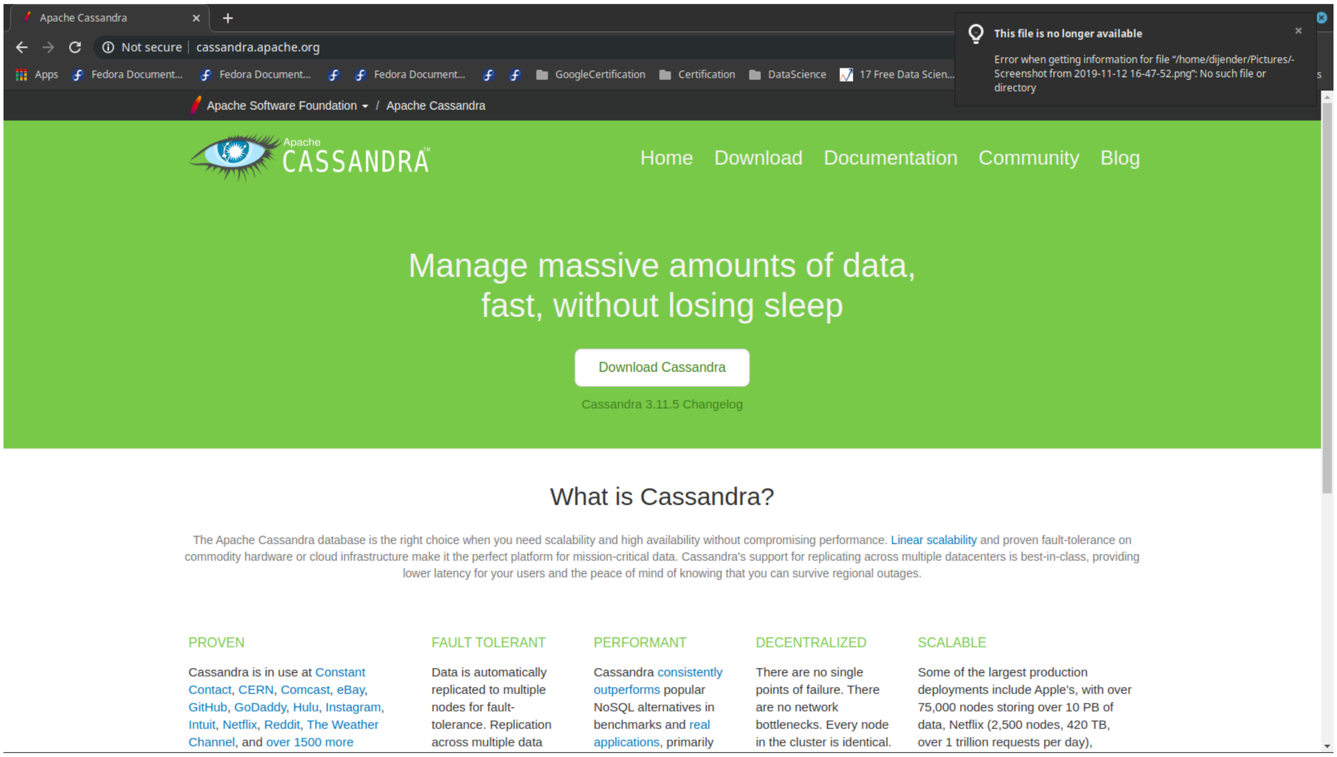

dijender@DijeBigMachine:~ — bash /home2/BigData/spark/bin/spark-shell
File
Edit
View
Search
Terminal
Help

```
[dijender@DijeBigMachine ~]$ spark-shell
2019-11-14 12:26:29,462 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DijeBigMachine:4040
Spark context available as 'sc' (master = local[*], app id = local-1573714603886).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.4
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_141)
Type in expressions to have them evaluated.
Type :help for more information.

scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@7bc3c59f

scala> val l = sc.parallelize(List(1,2,3,4,5,6,7))
l: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> l.sum()
res1: Double = 28.0

scala> 
```

As you can see, we created a simple RDD and ran a sum on the list of number.

3. Cassandra

1. Download Cassandra tar ball from Apache foundation.



This file is no longer available

Error when getting information for file "/home/dijender/Pictures/-Screenshot from 2019-11-12 16-47-52.png": No such file or directory

Manage massive amounts of data, fast, without losing sleep

What is Cassandra?

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.

PROVEN

Cassandra is in use at Constant Contact, CERN, Comcast, eBay, GitHub, GoDaddy, Hulu, Instagram, Intuit, Netflix, Reddit, The Weather Channel, and over 1500 more

FAULT TOLERANT

Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data

PERFORMANT

Cassandra consistently outperforms popular NoSQL alternatives in benchmarks and real applications, primarily

DECENTRALIZED

There are no single points of failure. There are no network bottlenecks. Every node in the cluster is identical.

SCALABLE

Some of the largest production deployments include Apple's, with over 75,000 nodes storing over 10 PB of data, Netflix (2,500 nodes, 420 TB, over 1 trillion requests per day),

2. Extract it at a location. I have extracted it at /home2/BigData/apache-cassandra-3.11.5
3. Setup .bashrc as shown below and source it.

```
#Cassandra
export CASSANDRA_HOME=/home2/BigData/apache-cassandra-3.11.5
export PATH=$PATH:$CASSANDRA_HOME/bin
```

4. Extract the tar ball on the VMs as well and setup .bashrc as shown above.
5. We only need to setup **cassandra.yaml** to make Cassandra work

```
[dijender@DijeBigMachine ~]$ cd $CASSANDRA_HOME
[dijender@DijeBigMachine apache-cassandra-3.11.5]$ cd conf
[dijender@DijeBigMachine conf]$ ls -l
total 160
-rw-r--r--. 1 dijender dijender 18250 Oct 24 22:13 cassandra-env.ps1
-rw-r--r--. 1 dijender dijender 12535 Oct 24 22:13 cassandra-env.sh
-rw-r--r--. 1 dijender dijender   148 Oct 24 22:13 cassandra-jaas.config
-rw-r--r--. 1 dijender dijender  1200 Oct 24 22:13 cassandra-rackdc.properties
-rw-r--r--. 1 dijender dijender  1358 Oct 24 22:13 cassandra-topology.properties
-rw-r--r--. 1 dijender dijender 58539 Nov 12 11:08 cassandra.yaml
-rw-r--r--. 1 dijender dijender  2082 Oct 24 22:13 commitlog_archiving.properties
-rw-r--r--. 1 dijender dijender  6360 Oct 24 22:13 cqlshrc.sample
-rw-r--r--. 1 dijender dijender  2757 Oct 24 22:13 hotspot_compiler
```

```
-rw-r--r--. 1 dijender dijender  9956 Oct 24 22:13 jvm.options
-rw-r--r--. 1 dijender dijender  1195 Oct 24 22:13 logback-tools.xml
-rw-r--r--. 1 dijender dijender  3809 Oct 24 22:13 logback.xml
-rw-r--r--. 1 dijender dijender  1603 Oct 24 22:13 metrics-reporter-config-sample.yaml
-rw-r--r--. 1 dijender dijender   291 Oct 24 22:13 README.txt
drwxr-xr-x. 2 dijender dijender  4096 Nov 11 23:04 triggers
[dijender@DijeBigMachine conf]$
```

6. As there’s no Master-Slave concept in Cassandra so all the nodes will be setup in the same manner.

Seeds: "192.168.56.101" (You need to add any one machine in the node here, with the gossip protocol it would be known to all nodes in the ring).
rpc_address: 127.0.0.1
listen_address: 192.168.56.1 (This would be the node’s own IP address).

7. Once Cassandra.yaml is setup, we need to start Cassandra in all the machines. This is different from Spark where we need to start process only on the master. Start cassandra using cassandra command.

8. Once Cassansdra is started on all machine, you can verify that all nodes are up and running using the following command.

```
[dijender@DijeBigMachine conf]$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address         Load       Tokens       Owns (effective)  Host ID                               Rack
UN  192.168.56.1    277.83 KiB  256          69.8%             848e362e-009e-4b74-a8e1-a8c34526632b  rack1
UN  192.168.56.101  235.76 KiB  256          67.5%             e0c411bb-5bc2-412f-808f-652ee7bb56fc  rack1
UN  192.168.56.102  303.24 KiB  256          62.7%             2df44d96-af2b-4e90-a024-18d903eeee69  rack1
```

Now we have a HDFS, Spark and Cassandra all up and running on the three nodes.