MSc Data Analytics

Research Project

Configuration Manual

Erin Gilheany

X14106671

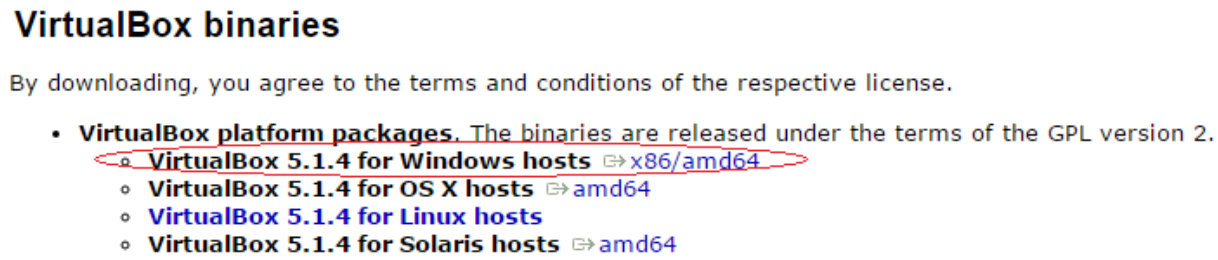
Supervisor: Vikas Sahni

Table of Contents

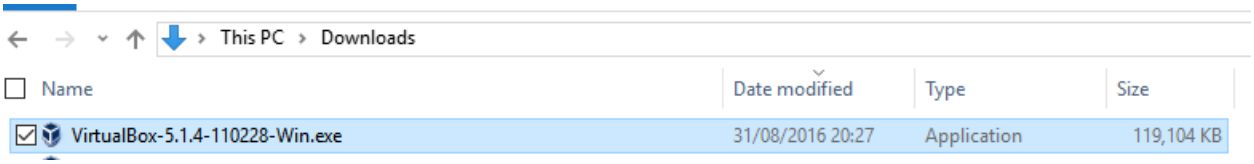
Setting up the Virtual Machines

1. Download and Install Virtual Machine Software

Firstly the software for setting up a virtual machine needs to be downloaded. I used “Oracle VM VirtualBox” which can be found here. The computer I am running this on is a Windows 64-bit so I downloaded the relevant release.

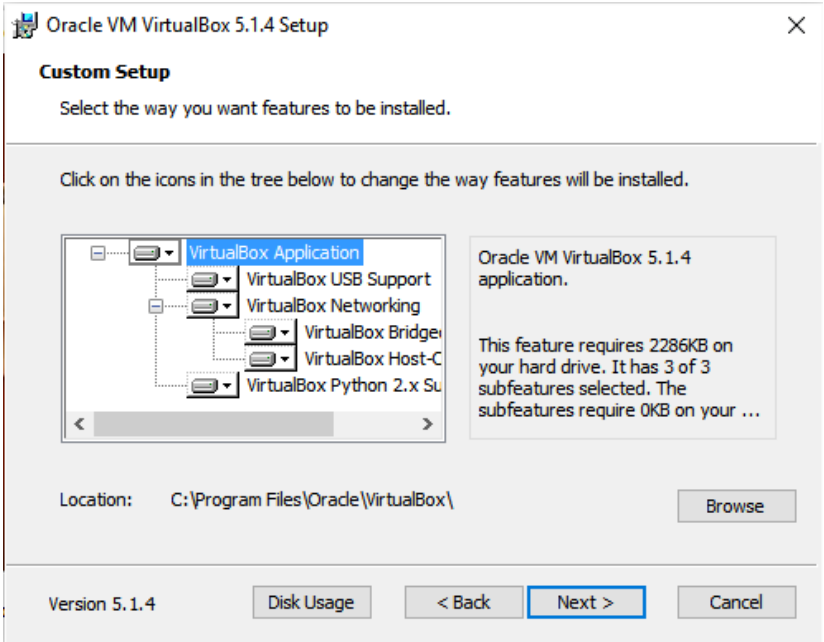


Go to the location where the above has been downloaded to and double-click to run the .exe file.

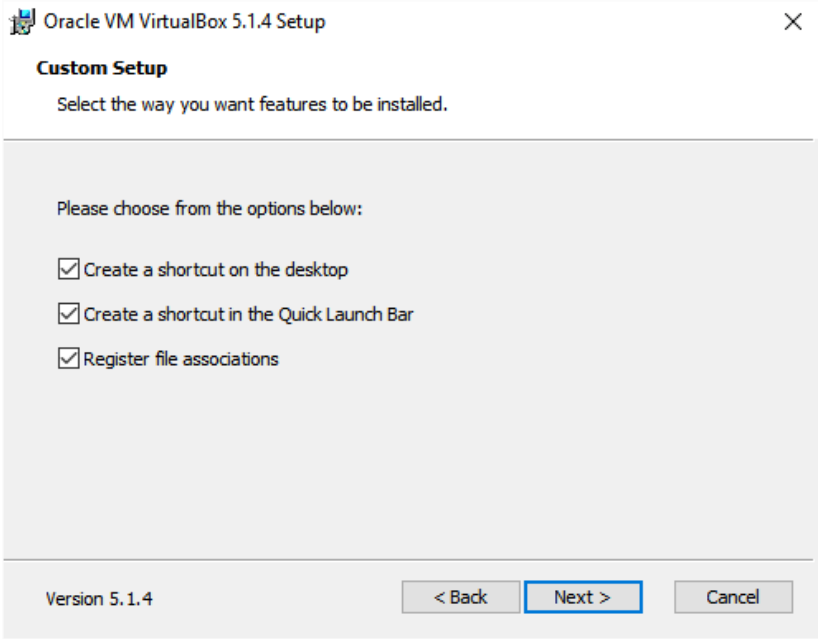


Click next on the below screen.

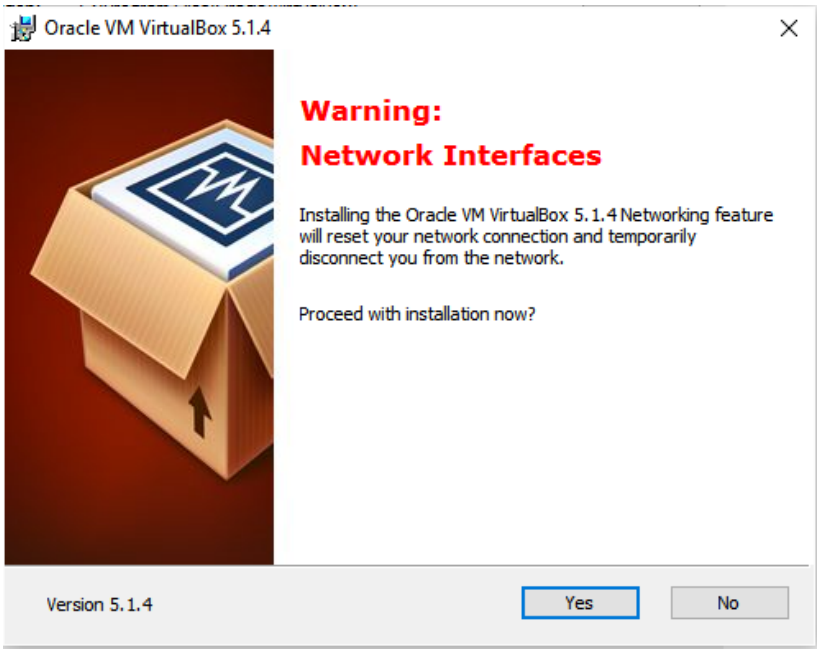
Click next on the below screen.



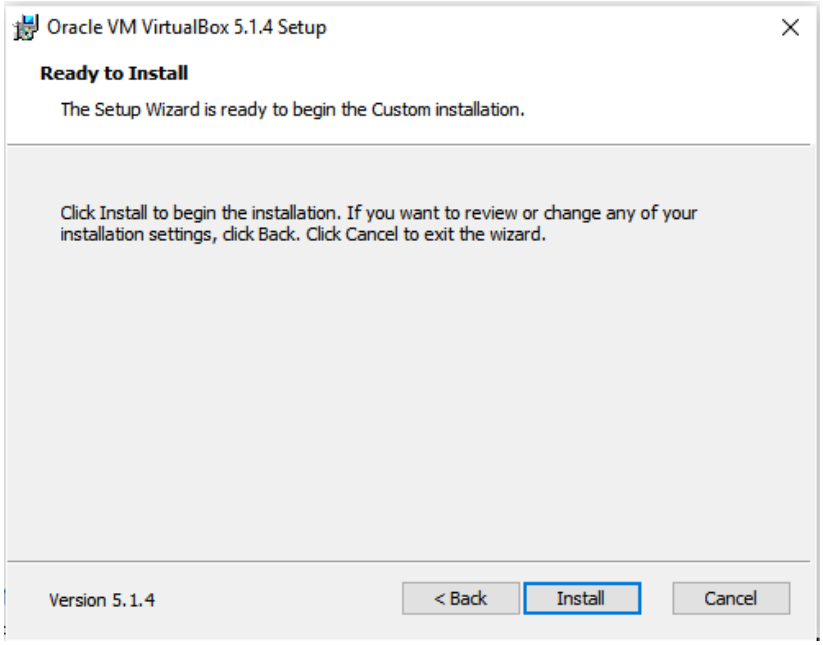
Click next on the below screen.



Click Yes on the below screen.



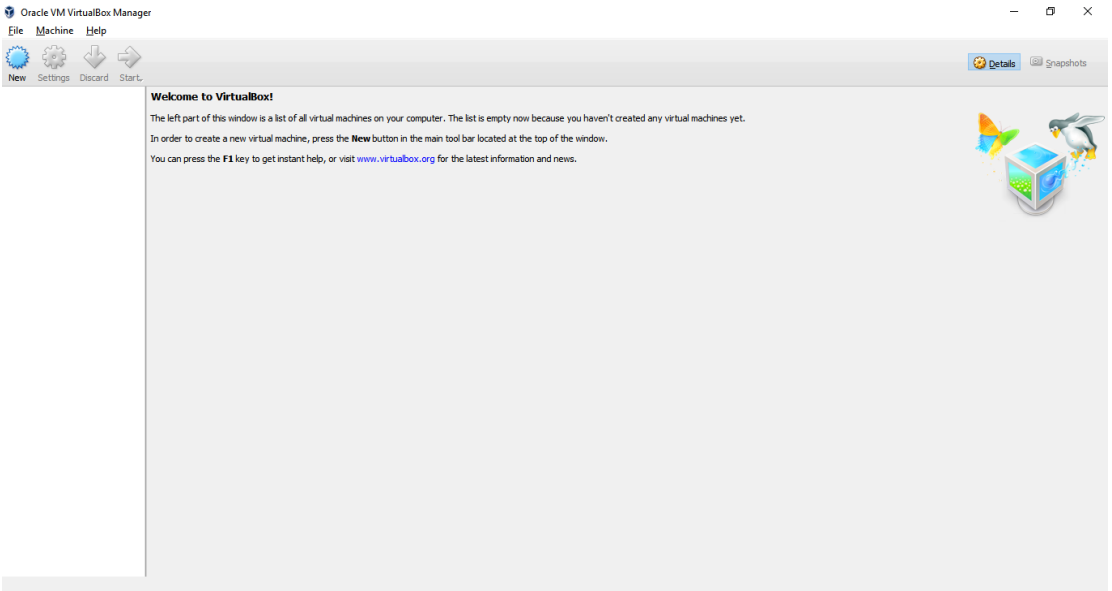
Click Install on the below screen.



The software will install and there should be an icon on your desktop. Please double click on this icon to run the newly installed application.

You should be presented with the below screen.

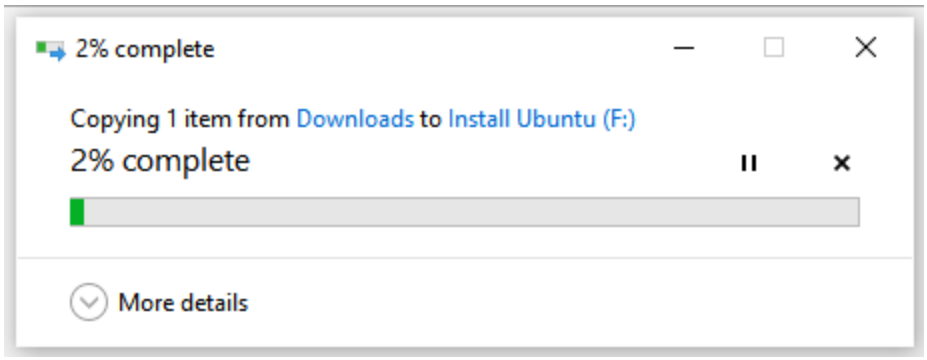
Note: You may need to restart the computer to proceed to this step after installation process has completed.



2. Download Ubuntu Operating System

Download an Ubuntu Desktop .iso file so that we can boot on Virtual Machine from here.

Preferably copy this file to an external disk (to be used when setting up a new VM).

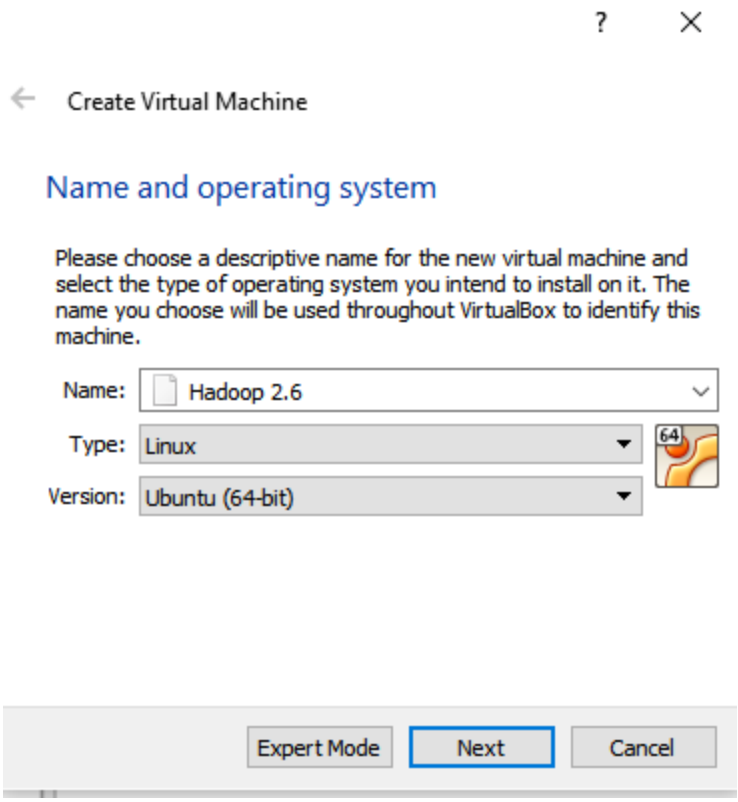


3. Set up our initial Virtual Machine – Hadoop 2.6

We return to our VM Ware application previously downloaded and click on “New” as per screen below.

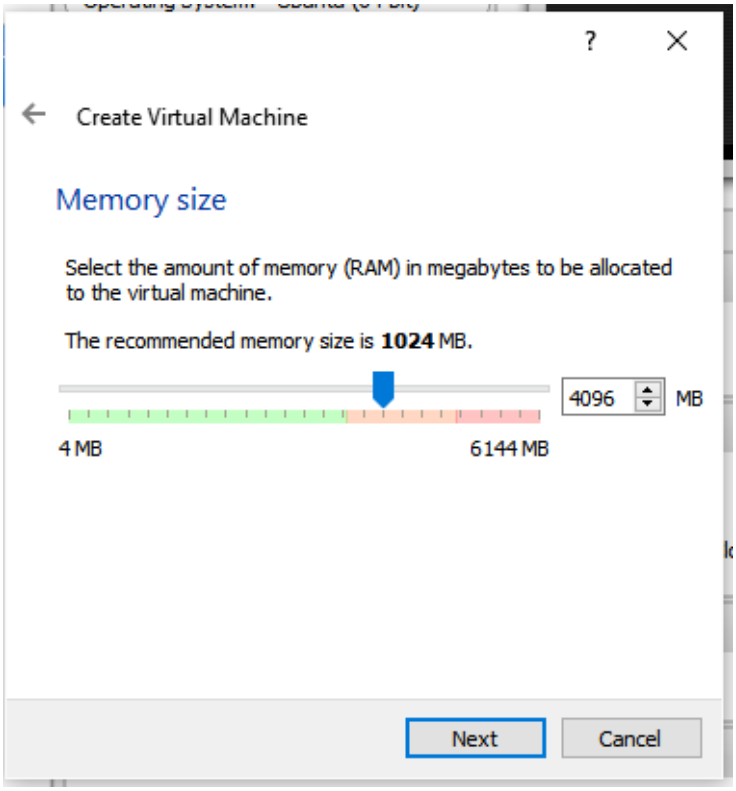
We will start off setting up the virtual machine which we run Hadoop version 2.6 on. Therefore we will name it as per screen below.

Please ensure to enter Linux in the Type dropdown and Ubuntu in the Version dropdown with the bit that matches the host computer.

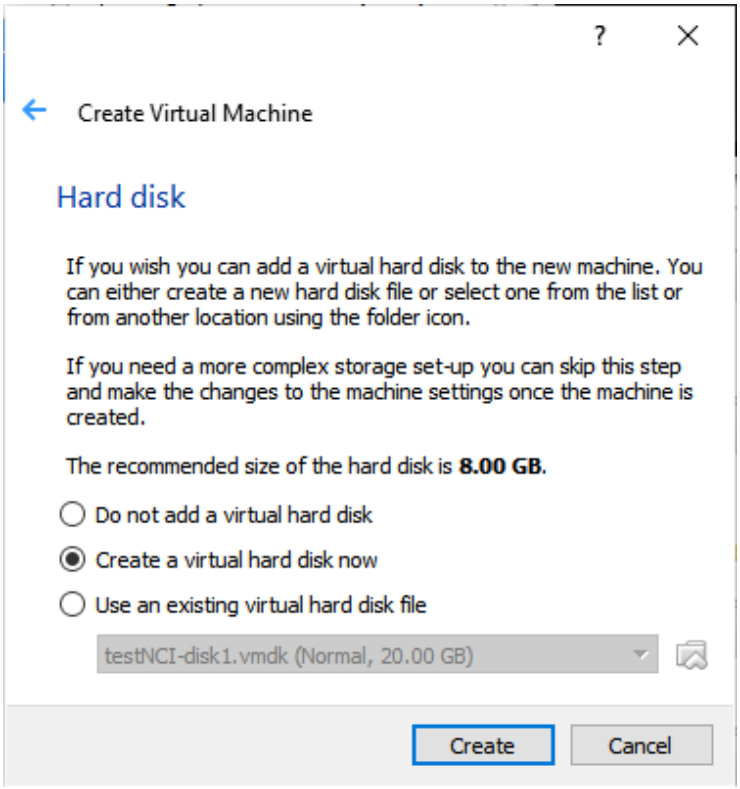


Set the VM memory size to be 4GB or 4096MB.

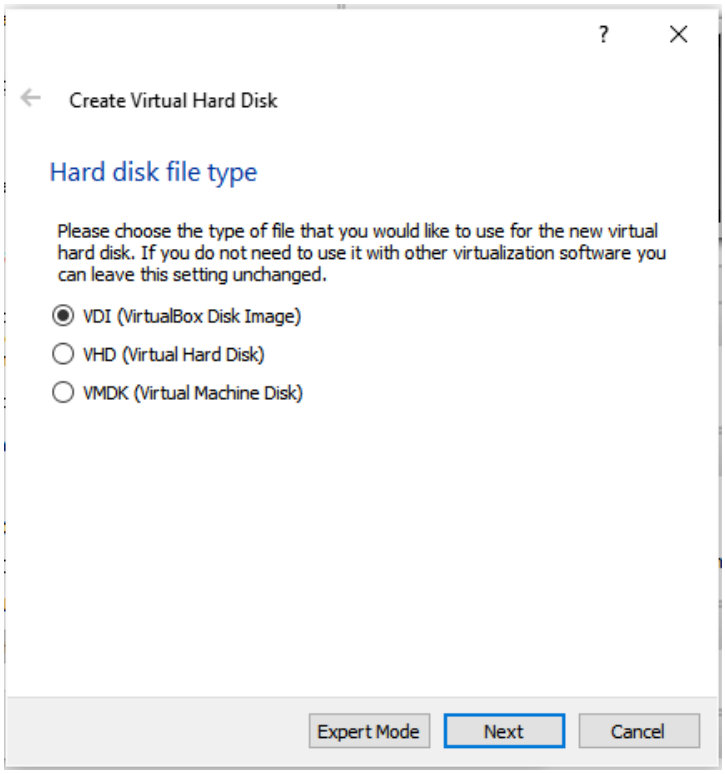
Note: The host machine in this instance has an overall memory of 6GB so this is close to the maximum memory that we can use for the virtual machines.



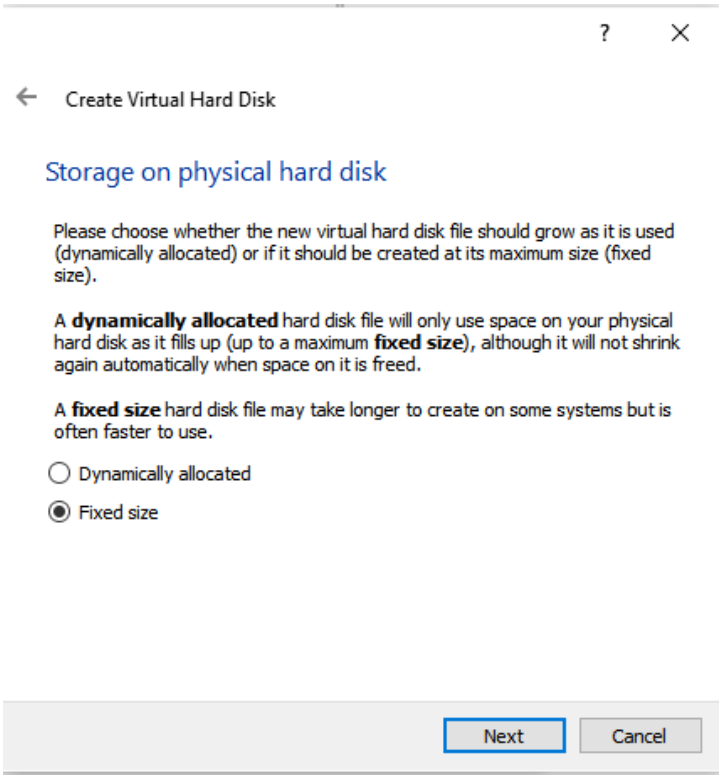
For this instance we have chosen “Create a virtual hard disk now”. Click Create.



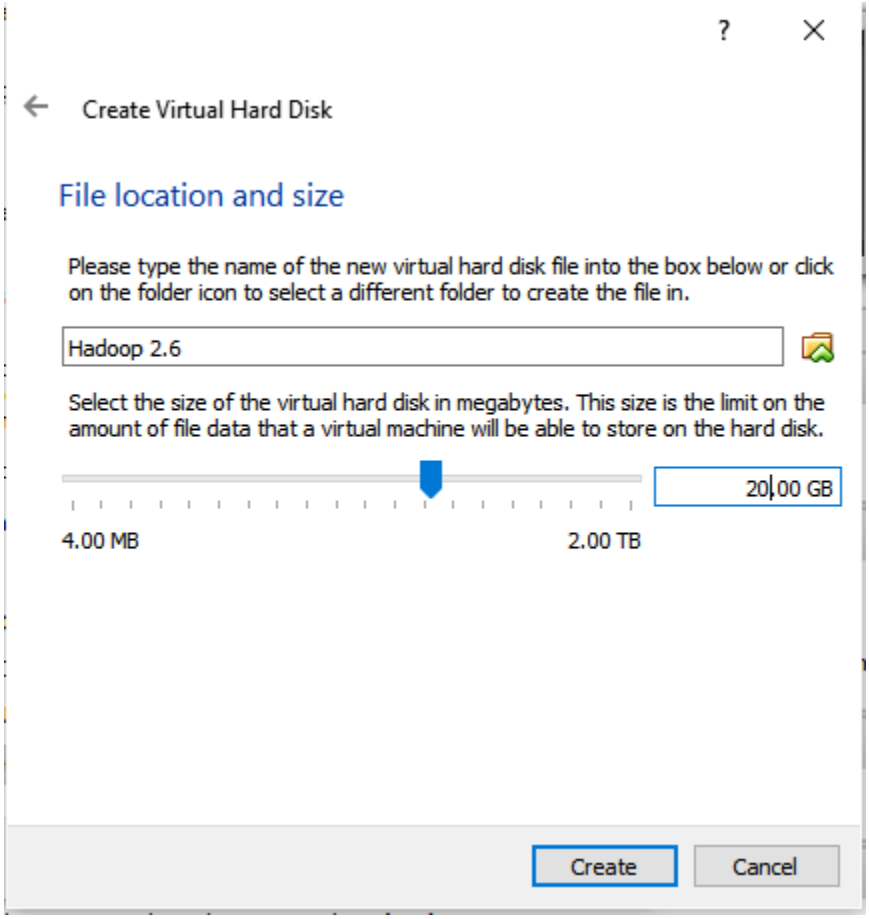
We have chosen to select “VDI (VirtualBox Disk Image)” as per screen below. Click Next.



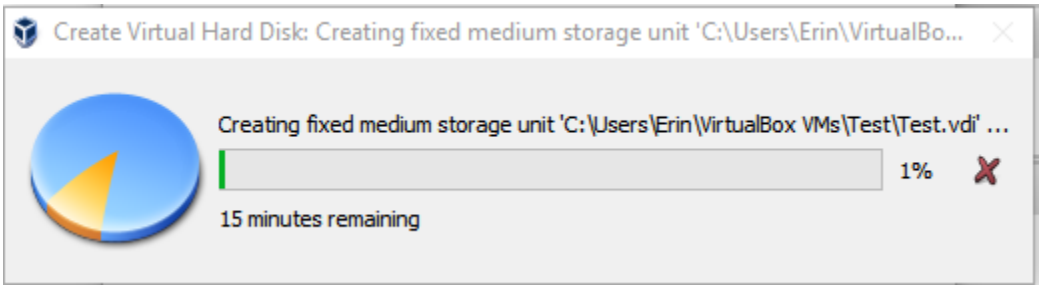
Choose “Fixed Size”. Click Next.



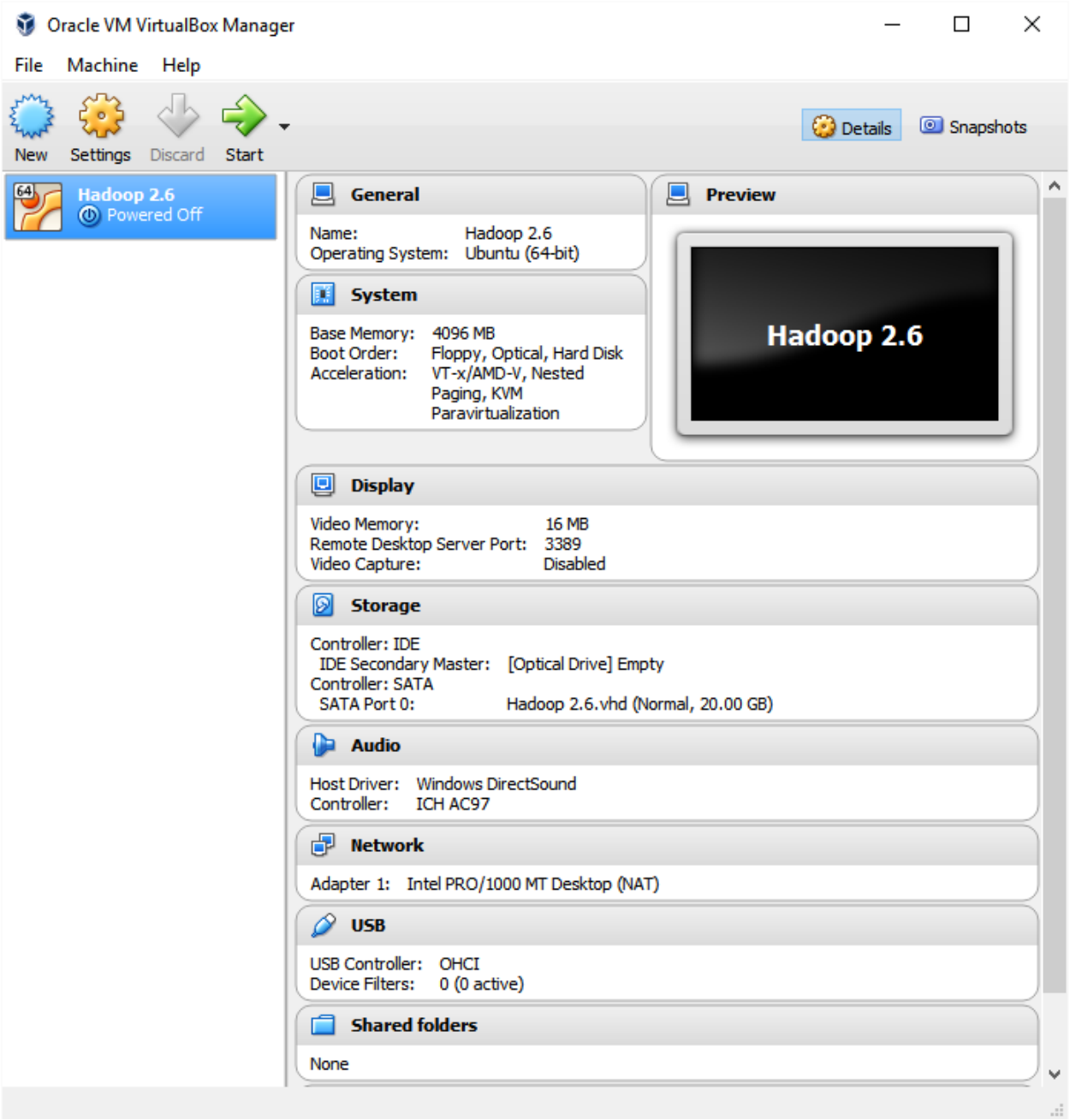
Change hard disk size to 20.00 Gb. Click Create.



The screen below will appear and make take a few minutes to complete.



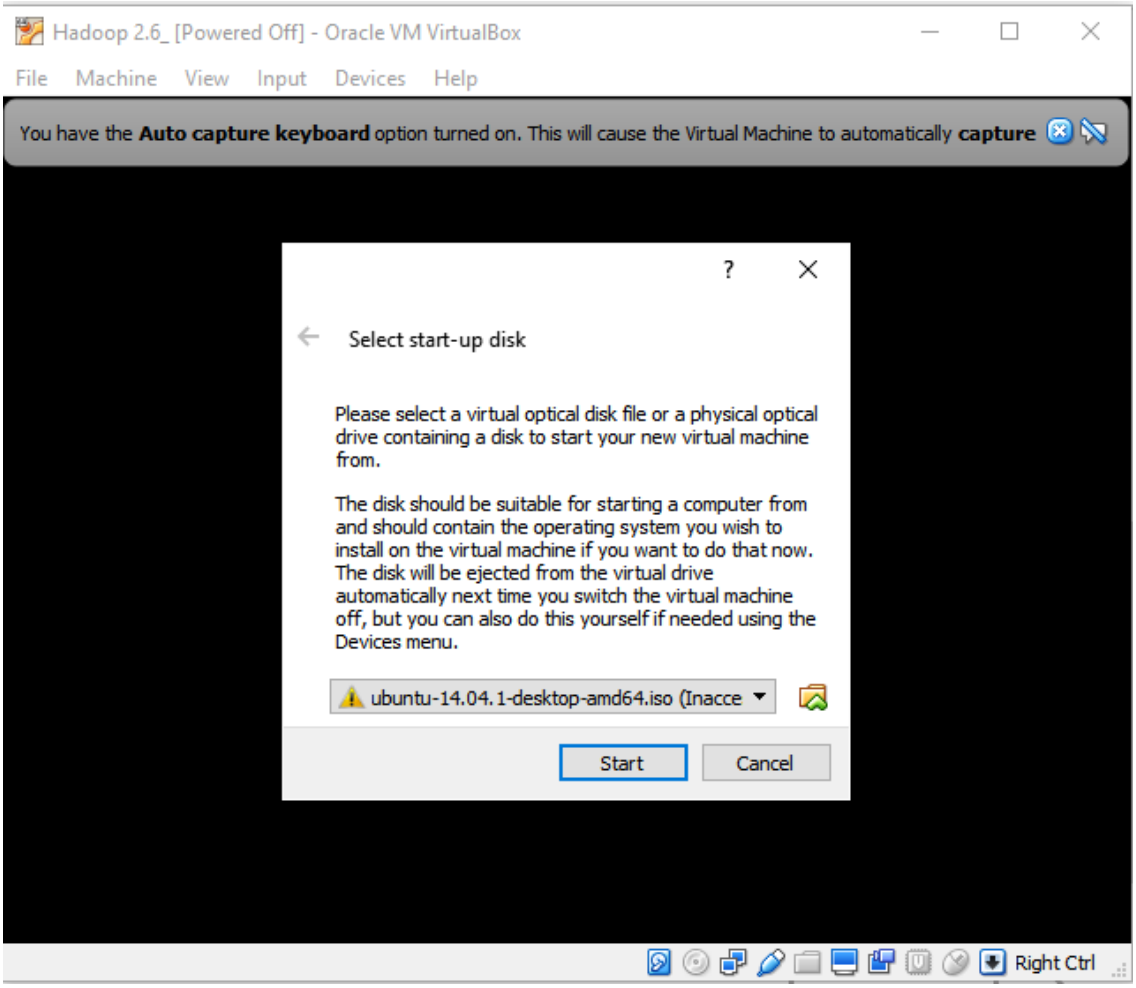
When the above process you will have your Virtual Machine in the screen as per below and ready to boot/start up.

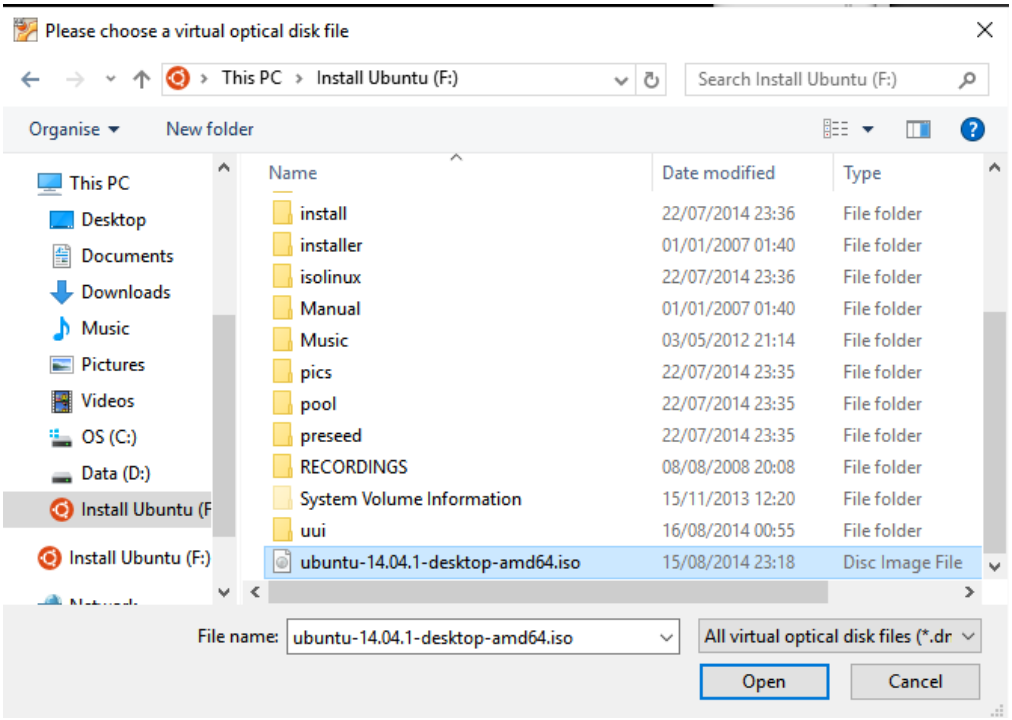


4. Booting up our Initial Virtual Machine

Double click on your “Hadoop 2.6” icon as per above screen.

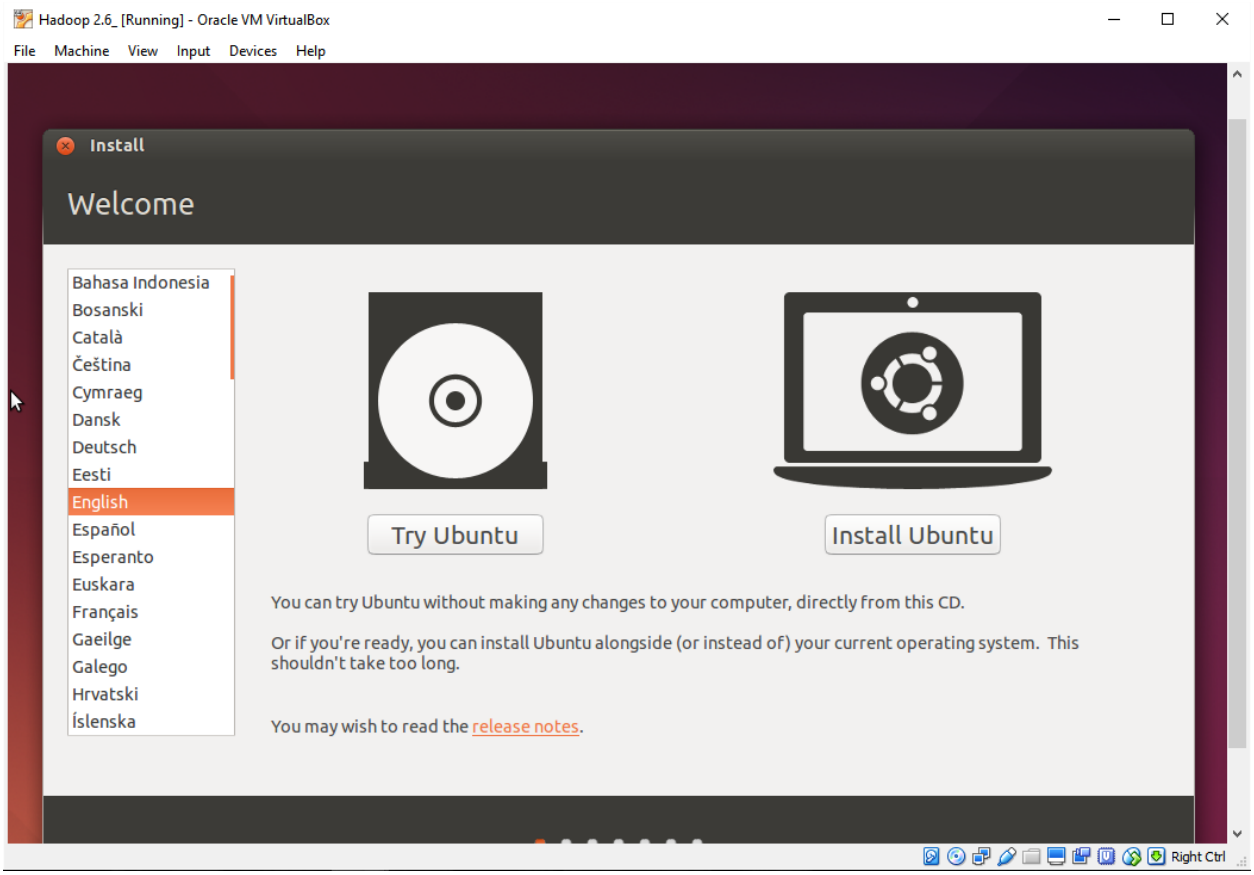
In the screen below, press on the folder icon and find the location where you moved the Ubuntu .iso file if it does not automatically appear here. Click Start.


Hadoop 2.6_ [Powered Off] - Oracle VM VirtualBox
File Machine View Input Devices Help
You have the Auto capture keyboard option turned on. This will cause the Virtual Machine to automatically capture
Select start-up disk
Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.
ubuntu-14.04.1-desktop-amd64.iso (Inacce
Start
Cancel
Right Ctrl


Please choose a virtual optical disk file
This PC › Install Ubuntu (F:)
Search Install Ubuntu (F:)
Organise
New folder
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Data (D:)
Install Ubuntu (F
Install Ubuntu (F:)
Name
Date modified
Type
install 22/07/2014 23:36 File folder
installer 01/01/2007 01:40 File folder
isolinux 22/07/2014 23:36 File folder
Manual 01/01/2007 01:40 File folder
Music 03/05/2012 21:14 File folder
pics 22/07/2014 23:35 File folder
pool 22/07/2014 23:35 File folder
preseed 22/07/2014 23:35 File folder
RECORDINGS 08/08/2008 20:08 File folder
System Volume Information 15/11/2013 12:20 File folder
uui 16/08/2014 00:55 File folder
ubuntu-14.04.1-desktop-amd64.iso 15/08/2014 23:18 Disc Image File
File name: ubuntu-14.04.1-desktop-amd64.iso
All virtual optical disk files (*.dr
Open
Cancel

It may take a short time for the next screen to load.

Select language – English in this case – and click “Install Ubuntu”.

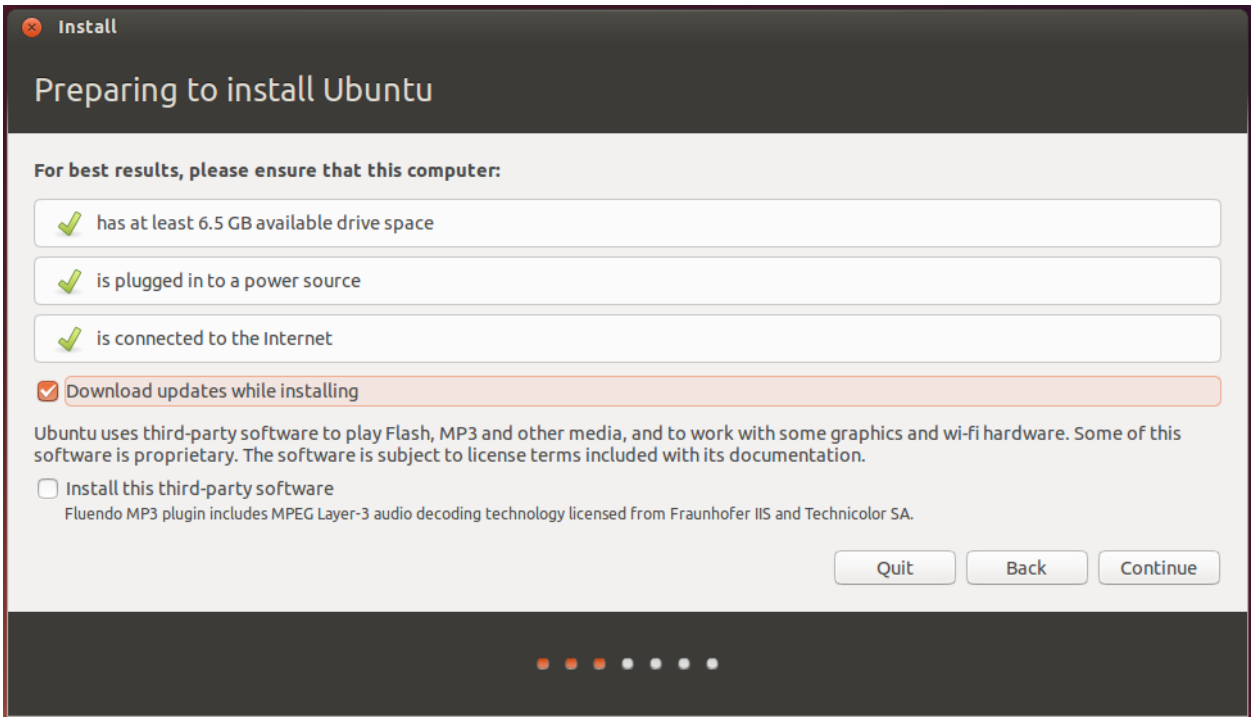


Follow instructions as per screen below by allowing 6.5GB available drive space, plug in your computer to a power source and keep it plugged in, and get and remain connected to the internet.

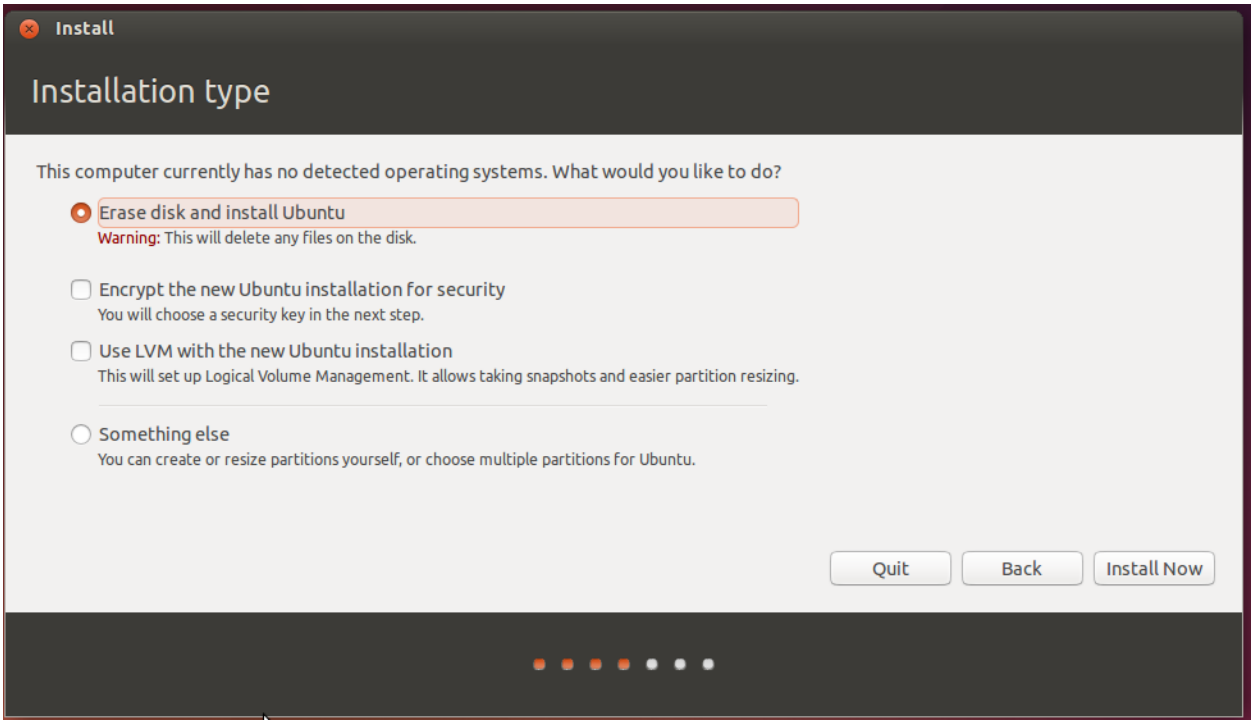
Click on “Download updates while installing”.

Note: Following instructions above will allow for a smoother install if any issues were to arise.

Click Continue.

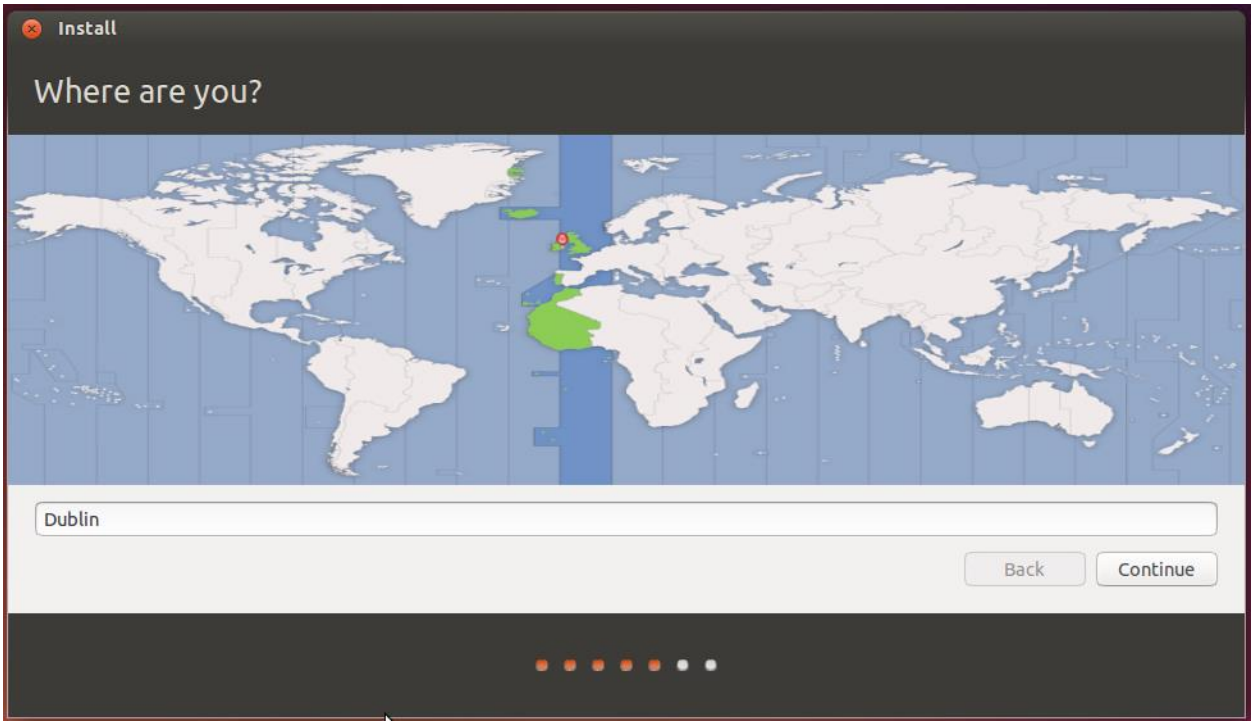


Select “Erase and install Ubuntu” and click “Install Now”.



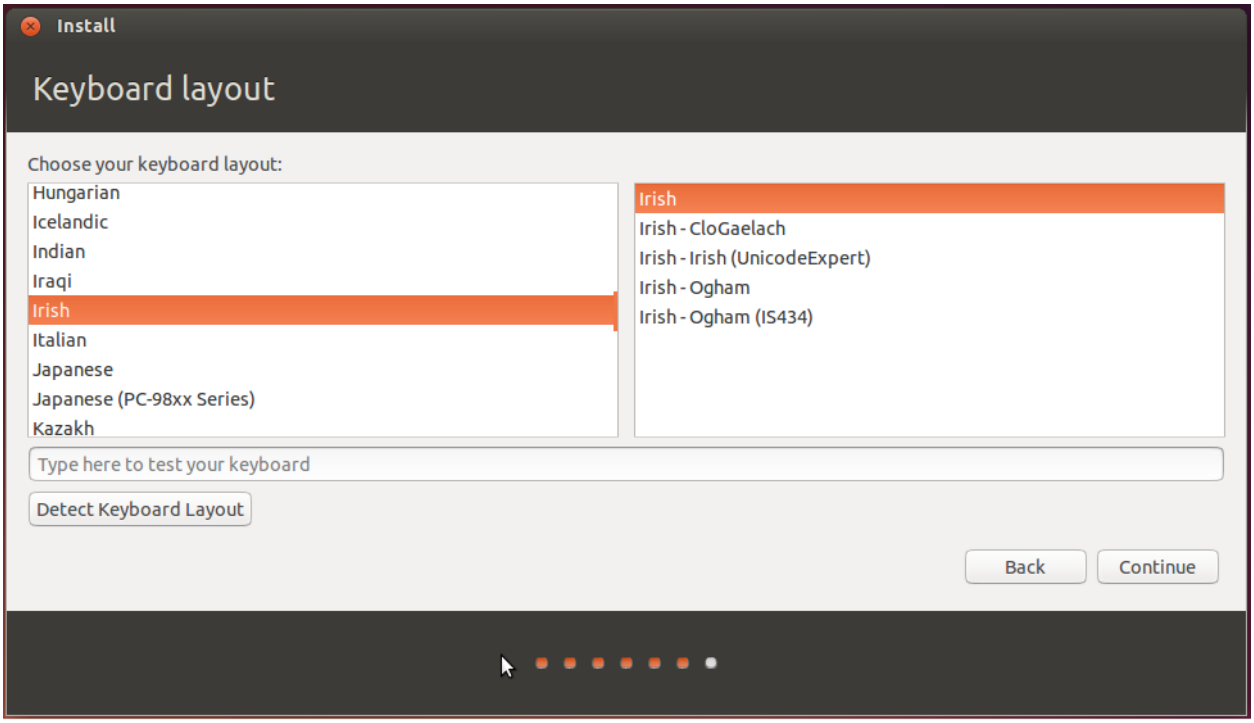
Input your location if this is not automatically picked up – in my case it’s Dublin.

Click Continue.



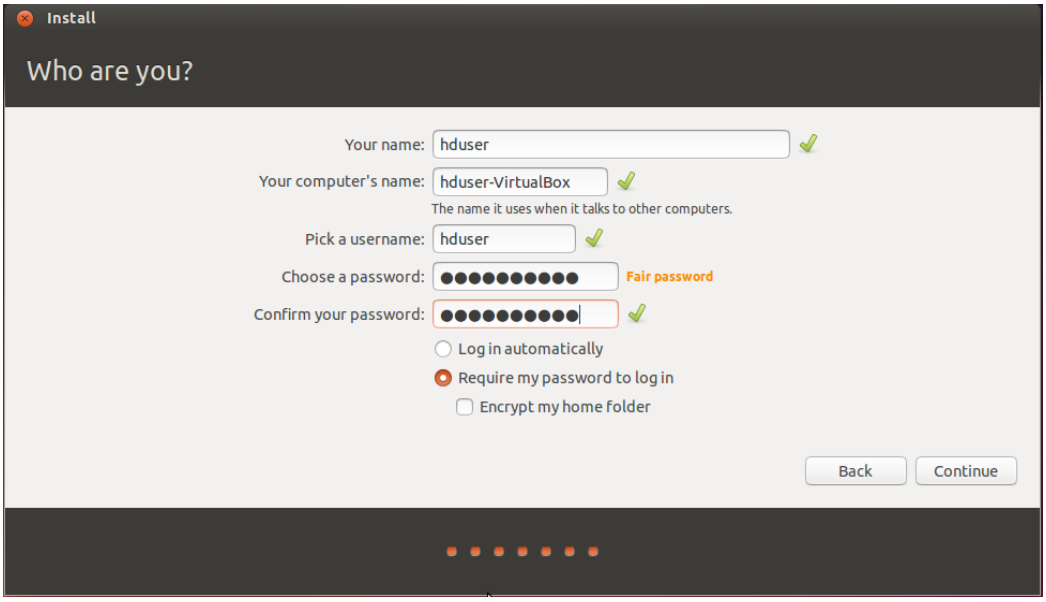
Select your desired Keyboard layout. I have chosen Irish as per below. Click Continue.

Note: This does not refer to Irish as a language, all language returns in English.



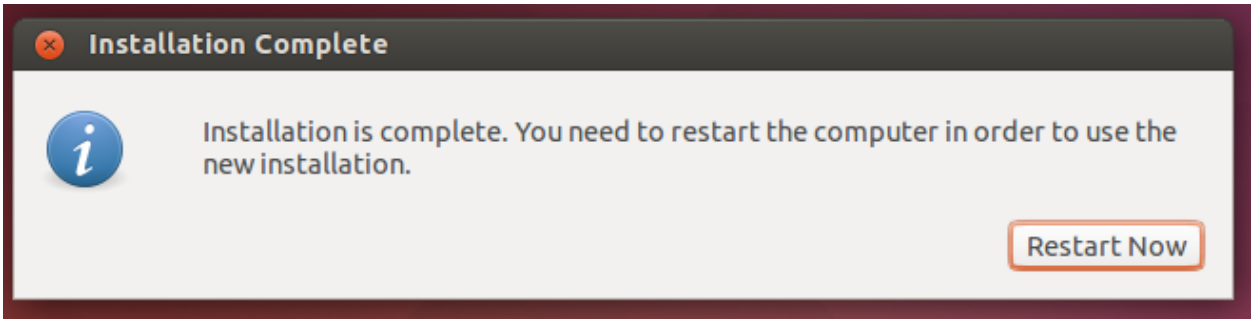
Fill in the fields in screen below with “Require my password to log in”. Click Continue.

Note: Please ensure that a memorable password is selected as we will need to use it later in the process.

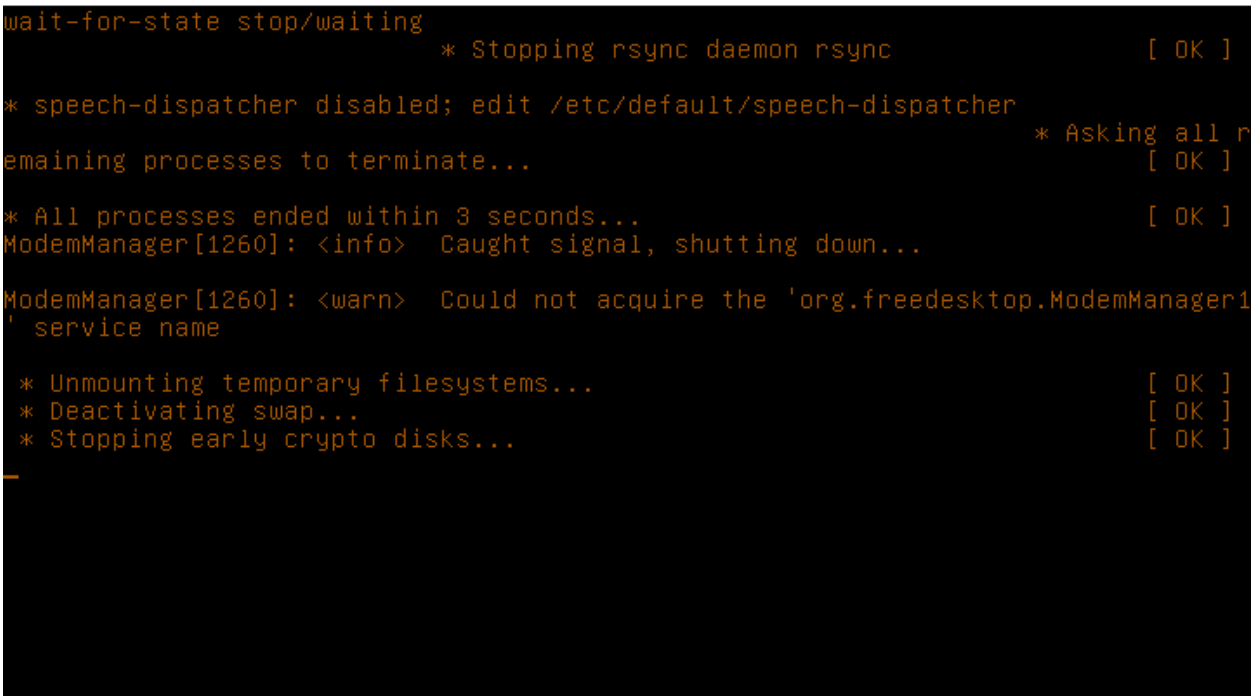


The following screen will appear. We just have to wait until the install finishes. This can take a few minutes.

Click on Restart Now when the screen below appears.

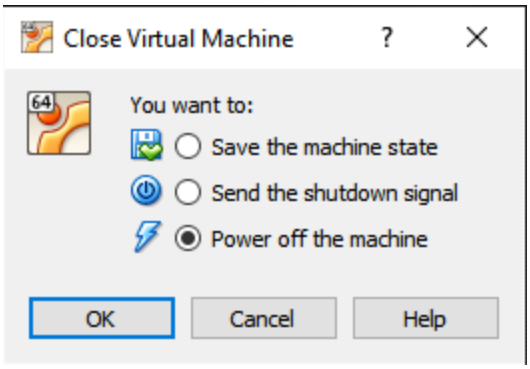


The below screen will appear. We do not have to press anything here.

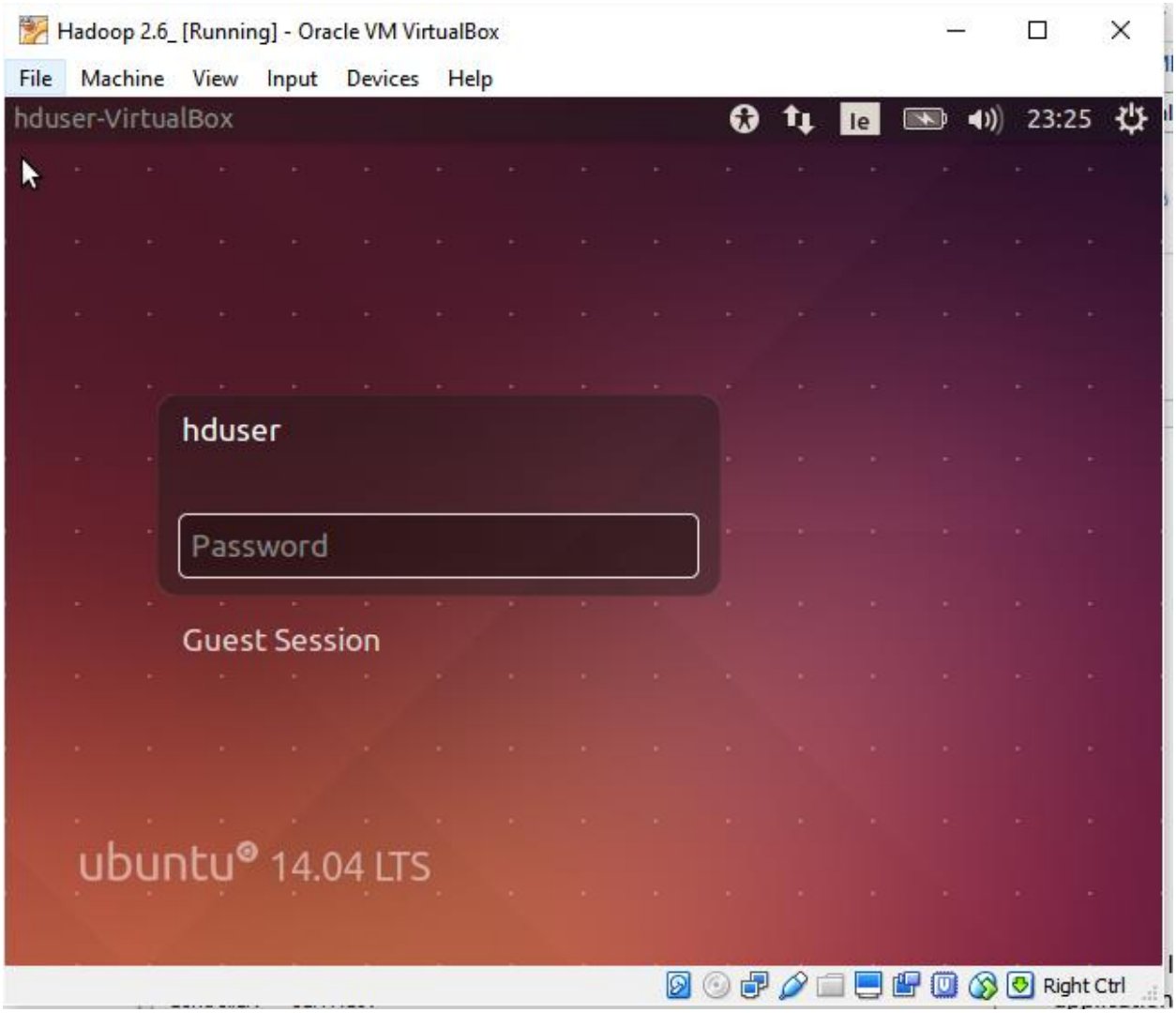


You may need to restart the VM to ensure everything is working correctly. Click on the X at top right hand side of the VM screen.

On the below select “Power off the machine” and click ok.



Reopen your newly created Virtual Machine and you will be presented with the following screen.



5. Change Screen Resolution on Virtual Box

Log into machine.

Open Terminal, type in the following:

sudo apt-get install virtualbox-guest-utils virtualbox-guest-x11 virutalbox-guest-dkms

Enter your login password

Enter Y and press enter when it displays: [Do you want to continue]

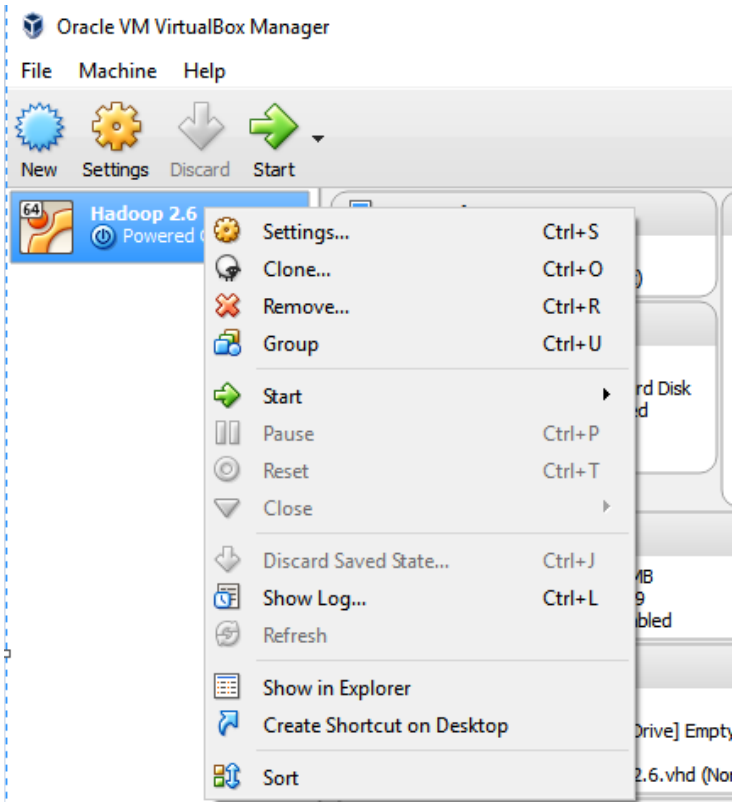
Restart computer and the screen resolution should fit your host machine's screen resolution and will adapt as your maximize or minimize the screen.

This is the last step in setting up the virtual machine. Before we dive into setting up software applications on this virtual machine, we close out of the above because we have to make an identical clone of this VM so that we can separately set up the other version of Hadoop we are going to test, Hadoop 2.7 and subsequently Spark 2.0.0. We will not set up a separate machine for Hadoop 2.7 and Spark 2.0 as Spark will sit on top of this version of Hadoop. It may be possible to set these all up on the one machine but this strategy avoids any confusion and possible mistakes when connecting Spark to HDFS.

6. Clone your initial VM to be used for testing of Hadoop 2.7 and Spark 2.0

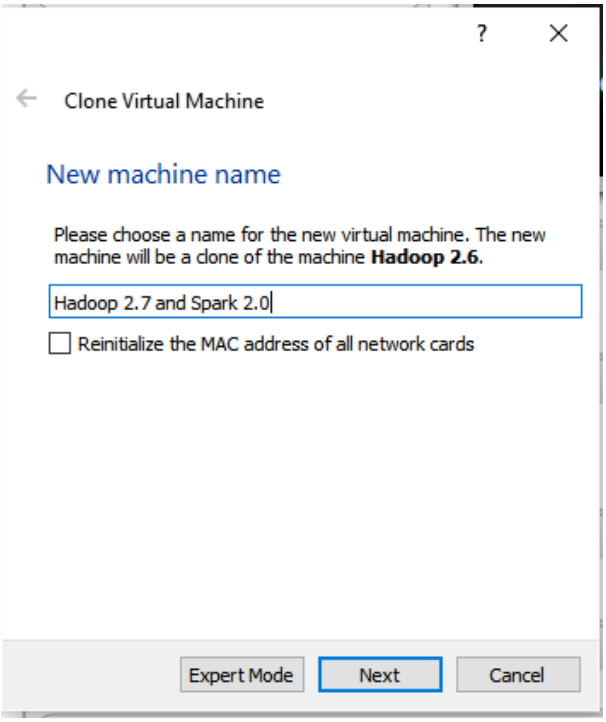
Back in the Oracle VM VirtualBox Manager screen we right click on the Hadoop 2.6 VM icon.

From menu that appears Click Clone.

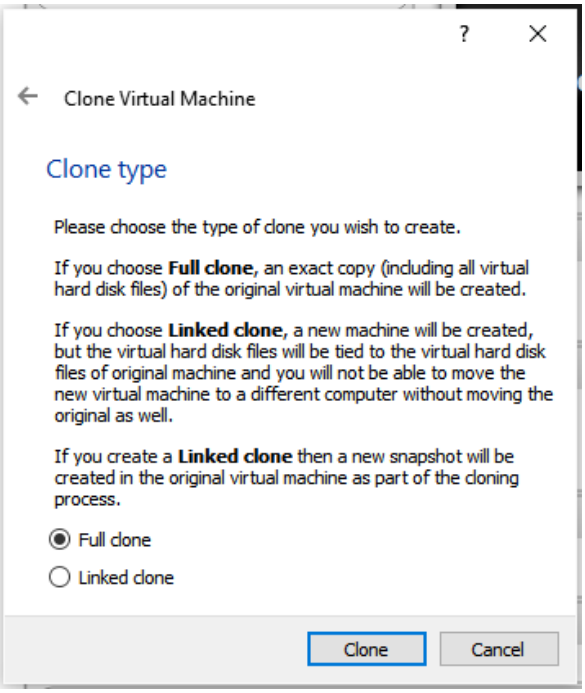


We will name our new VM in line with the application to be installed on it “Hadoop 2.7 and Spark 2.0”.

Click Next

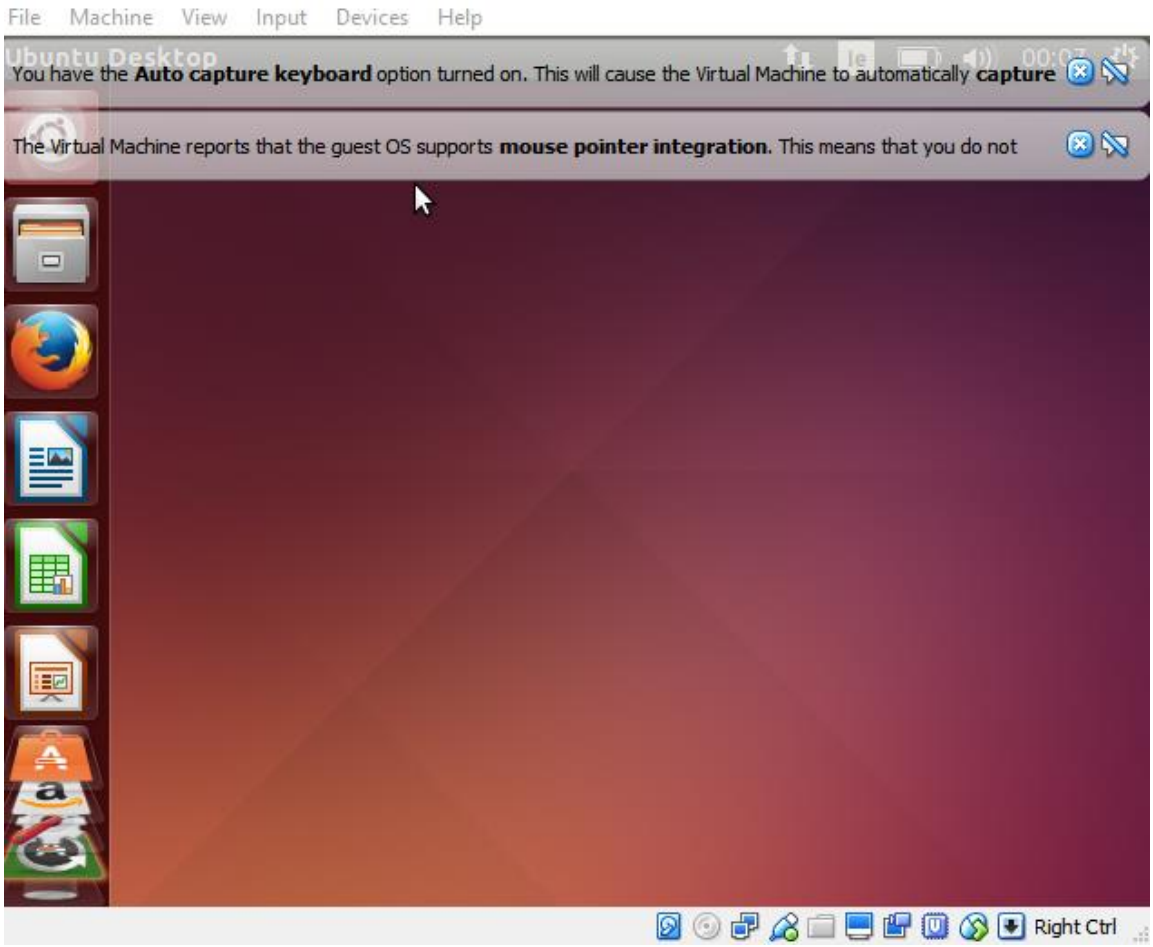


Select “Full clone”. Click Clone.



The following will appear. We just wait until this reaches 100% and our VM is fully cloned.

When complete open up the VM “Hadoop 2.7 and Spark 2.0” to check that the system boots and you can log in. And you will be presented with the below. You click on X on the two banners at the top of the page. Close out of this VM.



Note: This virtual machine will be identical at this point to VM “Hadoop 2.6” including the users and passwords. Anything done in either of the environments will only take effect within that VM, they are not synced or linked to one another.

Hadoop 2.6: Set Up, Configuration and Process Implementation

1. Set up Required Software and Environment

All of below commands will require your log-in password.

Note: At any stage of above commands you are asked "Do you want to continue? [Y/n]". Enter "Y" and hit return.

Open terminal and run the following commands:

sudo apt-get update
sudo apt-get install openjdk-7-jre
sudo apt-get install openjdk-7-jdk
sudo apt-get install ssh
sudo apt-get install rsync

sudo addgroup hadoop

If you have not already set up hduser account:

sudo adduser --ingroup hadoop hduser

If you set up the machine as hduser account:

sudo usermod -a -G hadoop hduser

Run this if you are not already logged in as hduser

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

Enter "/home/hduser/.ssh/authorized_keys" when it asks for a location as per screenshot below and hit return.

```
hduser@hduser-VirtualBox:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
```

ssh localhost

If you are presented with the following message, enter "yes" and hit return.

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is 67:2c:13:da:40:f1:28:bd:1a:47:2f:ce:bc:0a:b1:32.
Are you sure you want to continue connecting (yes/no)? yes
```

You may then have to enter your password and the below will return.

```
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
hduser@localhost's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

We exit out of ssh and log out of the VM OS by running the following commands:

exit

```
hduser@hduser-VirtualBox:~$ exit
logout
Connection to localhost closed.
```

If you have not set up as hduser run the exit command again to logout out of hduser. (Note: not the case for my set up as I only have hduser set up)

exit

Disable IPv6 as Hadoop is not supported for these networks.

sudo gedit /etc/sysct1.conf

Enter password if necessary. An empty document will appear. Enter the below text. Save and close the document.

```
# disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

2. Download, Set Up and Configure Hadoop 2.6.4

We will download Hadoop 2.6.4 itself.

sudo wget http://ftp.heanet.ie/mirrors/www.apache.org/dist/hadoop/common/hadoop-2.6.4/hadoop-2.6.4.tar.gz

sudo cp hadoop-2.6.4.tar.gz /usr/local/
cd /usr/local
sudo tar xvf hadoop-2.6.4.tar.gz
sudo ln -s hadoop-2.6.4 hadoop
sudo chown -R hduser:hadoop hadoop-2.6.4
sudo rm hadoop-2.6.4.tar.gz
sudo update-alternatives --config java

You will see similar text to the below return, please take note of the location where Java is installed, this will be needed later in the process. In my case it is "/usr/lib/jvm/java-7-openjdk-amd64" as per screen below:

```
hduser@hduser-VirtualBox:/usr/local$ sudo update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-op
enjdk-amd64/jre/bin/java
Nothing to configure.
```

Note: the location is not the full text as printed out but up until just before jre directory.

If not currently logged in as hduser do so:

su - hduser
cd /usr/local/hadoop/etc/hadoop
cp mapred-site.xml.template mapred-site.xml

In our hadoop-env.sh we enter the java location we took note of just a few steps ago.

vi hadoop-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

vi mapred-site.xml

Insert the following configuration properties to the file.

```
<configuration>
	<property>
		<name>mapreduce.jobtracker.address</name>
		<value>local</value>
	</property>
</configuration>
```

mkdir ~/tmp
mkdir ~/hdfs
chmod 750 ~/hdfs

vi core-site.xml

Enter the below properties to this file.

```
<configuration>
	<property>
		<name>hadoop.tmp.dir</name>
		<value>/home/hduser/tmp</value>
	</property>
	<property>
		<name>fs.defaultFS</name>
		<value>hdfs://localhost:54310</value>
	</property>
</configuration>
```

vi hdfs-site.xml

```
<configuration>
	<property>
```

```
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/home/hduser/hdfs</value>
    </property>
</configuration>
```

cd /usr/local/hadoop
bin/hdfs namenode -format
sbin/start-dfs.sh

*Enter password multiple times and type "yes" and return if asked to continue.

sbin/start-yarn.sh

jps

You should see the following return:

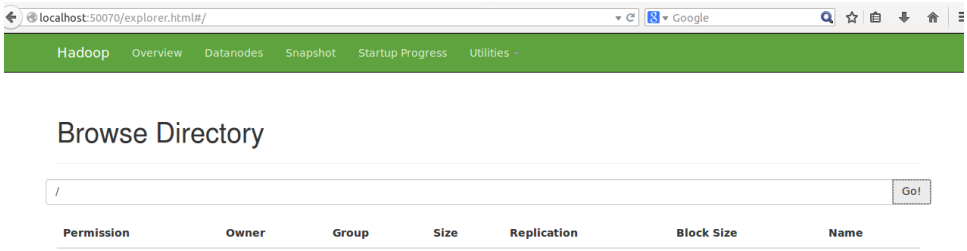
```
hduser@hduser-VirtualBox:/usr/local/hadoop$ jps
13154 DataNode
13327 SecondaryNameNode
13867 Jps
13466 ResourceManager
13773 NodeManager
13013 NameNode
```

A useful resource for all information on Hadoop Environment and Administration is to open a web browser/tab in web browser and type in the following location:

localhost:50070

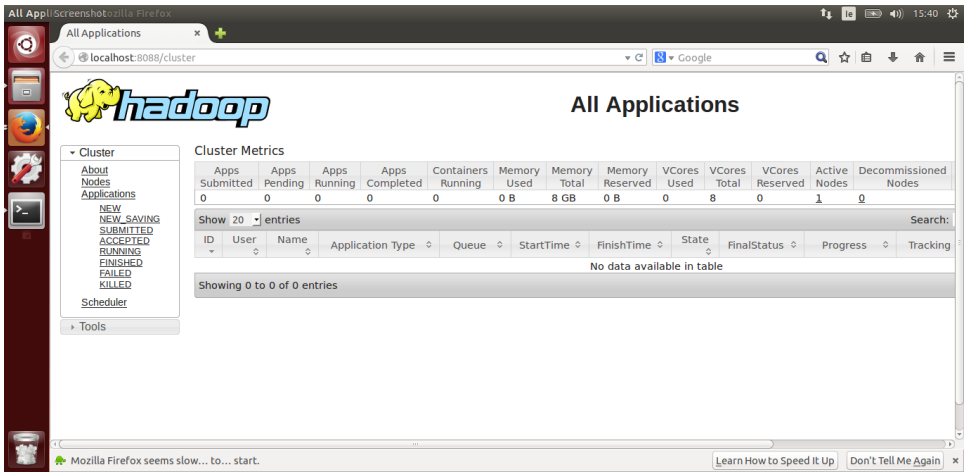
If you click on "Utilities" tab at top right and the "Browse the file system" you will see a page like screenshot below. Click Go to browse through file system or go directly to a file or directory.

Note: This is empty at the moment because we have created any directories or added any files to hdfs.



Verify All Applications for Cluster Browser is running as expected. Enter the following address into the web browser and you will see the screen below return:

localhost:8088



3. Download and Set Up Mahout 0.12.2

In terminal enter

cd ~
sudo wget http://mirrors.whoishostingthis.com/apache/mahout/0.12.2/apache-mahout-distribution-0.12.2.tar.gz

*Enter log-in password

sudo cp apache-mahout-distribution-0.12.2.tar.gz /usr/local
cd /usr/local

sudo tar xvf apache-mahout-distribution-0.12.2.tar.gz
sudo ln -s apache-mahout-distribution-0.12.2 mahout
sudo chown -R hduser:hadoop apache-mahout-distribution-0.12.2
sudo rm apache-mahout-distribution-0.12.2.tar.gz

Update bashrc file to ensure all of the following variables are included:

vi ~/.bashrc

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
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=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END

export MAHOUT_HOME=/usr/local/mahout
```

After you have updated the bashrc file as above you will need to restart the computer for changes to take effect.

Once you have restarted the computer re-enter the following commands to start up hadoop services again.

cd /usr/local/hadoop
sbin/start-dfs.sh
sbin/start-yarn.sh

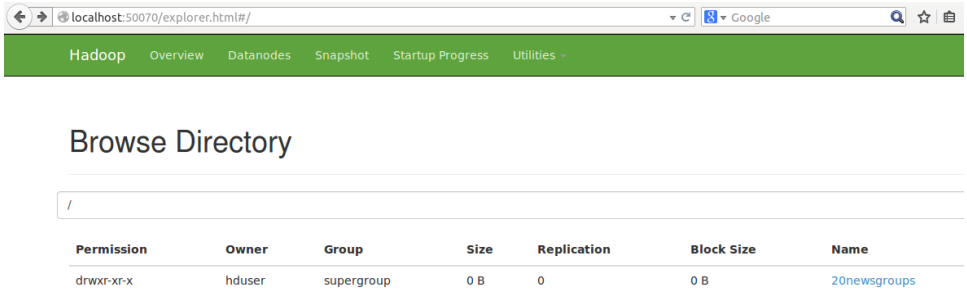
4. Download Dataset and move to HDFS

cd ~

sudo mkdir ~/20newsgroups
cd 20newsgroups
sudo wget http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate.tar.gz
sudo tar xvf 20news-bydate.tar.gz
cd ~
sudo chown -R hduser 20newsgroups

cd /usr/local/hadoop
bin/hdfs dfs -mkdir /20newsgroups
bin/hdfs dfs -copyFromLocal ~/20newsgroups//* /20newsgroups*

If we look back in our Hadoop Administration @ localhost:50070 and go to Utilities and Browse the file system, you will now see the directory as follows:



And when you click on 20newsgroups you will see subdirectories (20 news group categories) as follows (only some categories/subdirectories show because all cannot fit on screenshot, scroll down to view all of them):

localhost:50070/explorer.html#/20newsgroups

Browse Directory

/20newsgroups

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	alt.atheism
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comp.graphics
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comp.os.ms-windows.misc
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comp.sys.ibm.pc.hardware
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comp.sys.mac.hardware
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	comp.windows.x
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	misc.forsale
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	rec.autos
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	rec.motorcycles
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	rec.sport.baseball
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	rec.sport.hockey

Each subdirectory contains a large number of documents within as per below:

localhost:50070/explorer.html#/20newsgroups/alt.atheism

Browse Directory

/20newsgroups/alt.atheism Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	11.61 KB	1	128 MB	49960
-rw-r--r--	hduser	supergroup	31.3 KB	1	128 MB	51060
-rw-r--r--	hduser	supergroup	4 KB	1	128 MB	51119
-rw-r--r--	hduser	supergroup	1.59 KB	1	128 MB	51120
-rw-r--r--	hduser	supergroup	773 B	1	128 MB	51121
-rw-r--r--	hduser	supergroup	4.8 KB	1	128 MB	51122
-rw-r--r--	hduser	supergroup	618 B	1	128 MB	51123
-rw-r--r--	hduser	supergroup	1.42 KB	1	128 MB	51124
-rw-r--r--	hduser	supergroup	2.7 KB	1	128 MB	51125
-rw-r--r--	hduser	supergroup	427 B	1	128 MB	51126
-rw-r--r--	hduser	supergroup	742 B	1	128 MB	51127

You can manually download these files if you wish to see text within by clicking on Name and then download on the pop up that appears.

5. Run Mahout Algorithms

Create folder to store results/output of process

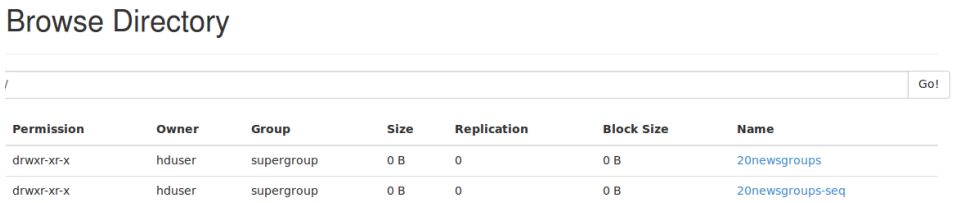
```
mkdir ~/mahout
```

Run transformation to sequence process

```
cd /usr/local/mahout
```

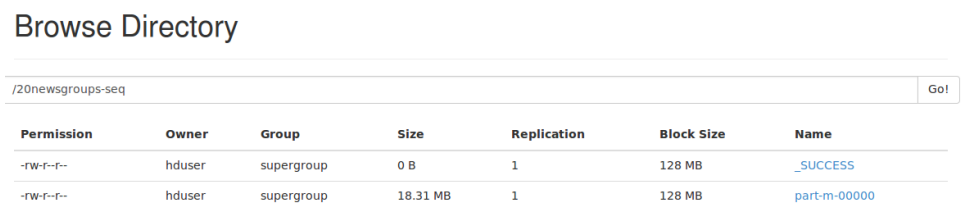
bin/mahout seqdirectory -i /20newsgroups -o /20newsgroups-seq -ow

After this is successfully run we will see a new directory in HDFS files system (go to localhost:50070 -> Utilities -> Browse the file system)



Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups-seq

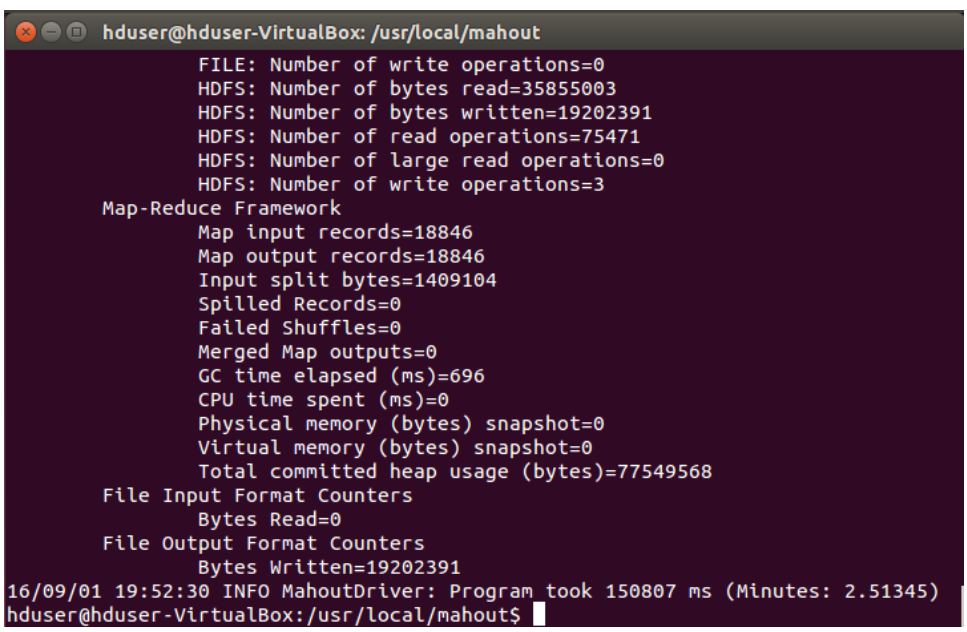
And within this directory two files:



Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	18.31 MB	1	128 MB	part-m-00000

Return to the terminal take note of the output of previous command.

Copy and paste this into a file as we will use this for analysis later on.



1. TFIDF on Hadoop

Run the following commands:

bin/mahout seq2sparse -i /20newsgroups-seq -o /20newsgroups-vectors -lnorm -nv -wt tfidf

Browse Directory

/ Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups-seq
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups-vectors

Browse Directory

/20newsgroups-vectors Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	df-count
-rw-r--r--	hduser	supergroup	1.85 MB	1	128 MB	dictionary.file-0
-rw-r--r--	hduser	supergroup	1.8 MB	1	128 MB	frequency.file-0
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	tf-vectors
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	tfidf-vectors
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	tokenized-documents
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	wordcount

2. Naïve Bayes on Hadoop

bin/mahout trainnb -i /20newsgroups-vectors/tfidf-vectors -o /model -li /labelindex -ow

Browse Directory

/ Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups-seq
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	20newsgroups-vectors
-rw-r--r--	hduser	supergroup	658 B	1	128 MB	labelindex
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	model
drwxr-xr-x	hduser	supergroup	0 B	0	0 B	user

Save output of command in terminal to a file for later analysis.

Hadoop 2.7: Set Up, Configuration and Process Implementation

1. Updates to Process Needed for Hadoop 2.7.3

Shut down and close out of VM “Hadoop 2.6”.

Open up VM “Hadoop 2.7 and Spark 2.0”.

The process for this will be the exact same as the steps we toke for Hadoop 2.6 except for 1 thing – hadoop version references.

We just need to update the command:

sudo wget http://ftp.heanet.ie/mirrors/www.apache.org/dist/hadoop/common/hadoop-2.6.4/hadoop-2.6.4.tar.gz

with

sudo wget ***http://www-eu.apache.org/dist/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz***

Also please ensure that any command which contain reference to Hadoop version 2.6.4, update this command to reference Hadoop 2.7.3.

Spark 2.0: Set Up, Configuration and Process Implementation

1. Download, Set Up and Configure Spark 2.0.0

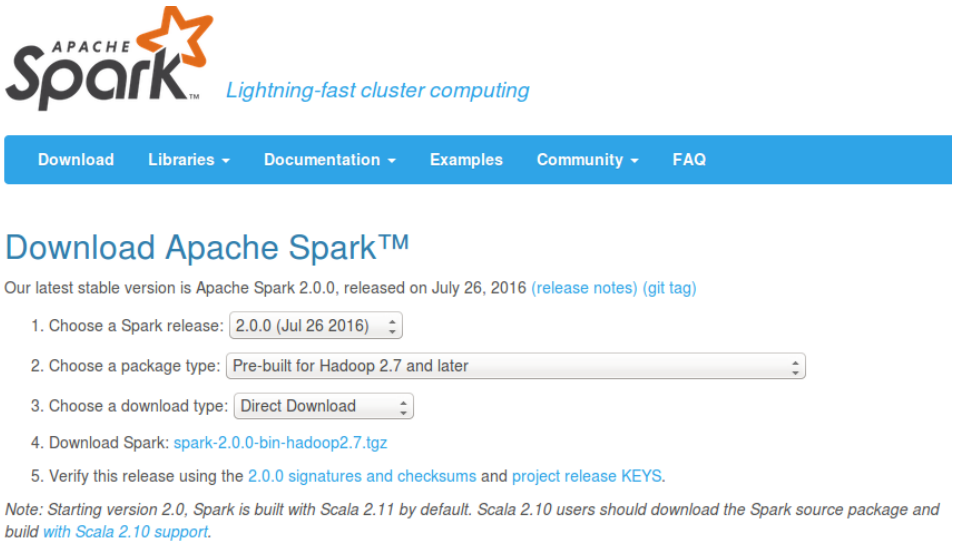
We are still using VM "Hadoop 2.7 and Spark 2.0", so ensure we are still logged in to this after completing Set Up, Configuration and Process Implementation for Hadoop 2.7.

Our implementation of Spark will connect directly into the same data we used for the Hadoop experiment, which sits in hdfs (hadoop file system).

So we download Spark from the following location

http://spark.apache.org/downloads.html

And ensure that we have selected the correct versions of both Spark (2.0.0) and Hadoop (2.7)



We then move the downloaded Spark file to a specific directory. In this case I have moved it to the same location as Hadoop: */usr/local/*

Using the Linux commands below:

cd ~/Downloads
sudo cp spark-2.0.0-bin-hadoop2.7.tgz /usr/local/

We then go to this directory, extract the file and set up a symbolic link (and name it Spark) for easier navigation to spark.

cd /usr/local

sudo tar –xvzf spark-2.0.0-bin-hadoop2.7.tgz
sudo ln -s spark-2.0.0-bin-hadoop2.7 spark

(Optional) I then removed the original downloaded .tgz file from this directory just to keep the folder as clean as possible.

sudo rm spark-2.0.0-bin-hadoop2.7.tgz

The user needs to be given permission to be able to use spark.

sudo chown -R hduser:hadoop /usr/local/spark

2. Install Scala and check Spark Set Up

First of all we will test Spark in local mode to ensure it is running correctly.

In order to do so we need to have the latest version of Scala installed as the Spark Shell uses this programming language so we download Scala from the following location:

sudo apt-get remove scala-library scala
sudo wget http://www.scala-lang.org/files/archive/scala-2.11.8.deb
sudo dpkg -i scala-2.11.8.deb
sudo apt-get update
sudo apt-get install scala

To test that the local install is working correctly we navigate to the spark directory and open the spark shell

cd /usr/local/spark
bin/spark-shell

We see a screen that appears as follows:

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.7.0_95)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Enter ":quit" and hit return to exit shell.

We now need to configure Spark so that it connects with the Hadoop set up that has already been set up which currently consists of one datanode as follows

```
hduser@ncistudent-VirtualBox:/usr/local/spark$ jps
24285 DataNode
24507 SecondaryNameNode
24683 ResourceManager
31322 Jps
24821 NodeManager
24146 NameNode
```

We add the following lines to the bashrc file, opening it using the command:

vi ~/.bashrc

#SPARKVARIABLES

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

Restart the VM so bashrc changes take effect. Log in to VM again.

We must configure the file "spark-env.sh" using the commands as follows:

cd /usr/local/spark/conf

Here there is a file named "spark-env.sh.template". We use this as the basis to our "spark-env.sh" as it does not yet exist. So we duplicate and remove "template" in the name and save it in the same location (/usr/local/spark/conf/)

sudo cp spark-env.sh.template spark-env.sh

export HADOOP_CONF_DIR=/usr/local/hadoop/conf

Set up a log folder and ensure permission is granted to user hduser:hadoop

cd /usr/local/spark
sudo mkdir logs
sudo chown hduser:hadoop logs

Restart all services

cd /usr/local/hadoop
sbin/stop-all.sh
sbin/start-dfs.sh
sbin/start-yarn.sh

On top of the Hadoop services we also have to start our Spark Master and Worker

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

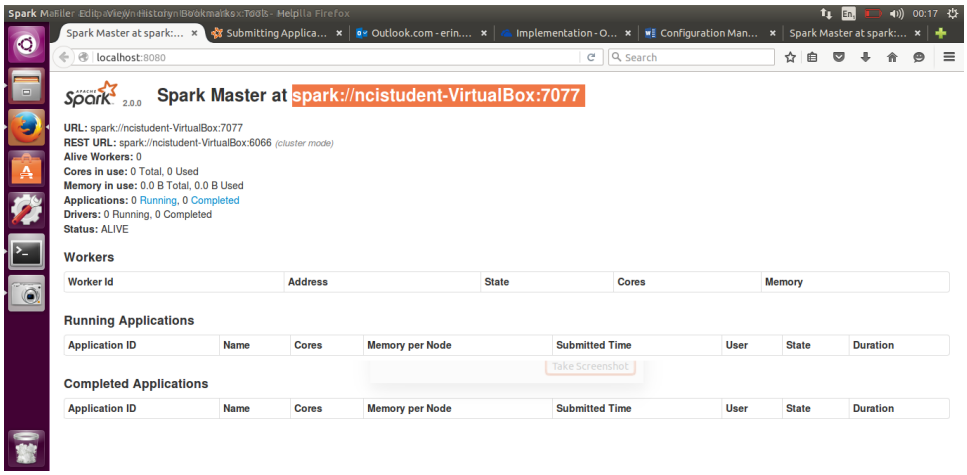
jps

This is to ensure Master is running as in below:

```
hduser@ncistudent-VirtualBox:/usr/local/spark$ jps
2899 NodeManager
11646 Jps
2763 ResourceManager
3473 SparkSubmit
2559 SecondaryNameNode
3241 Master
11606 Worker
2228 NameNode
2367 DataNode
```

You can view more in info relating to Spark master at address in web browser:

localhost:8080



Run spark-shell in order to use machine learning

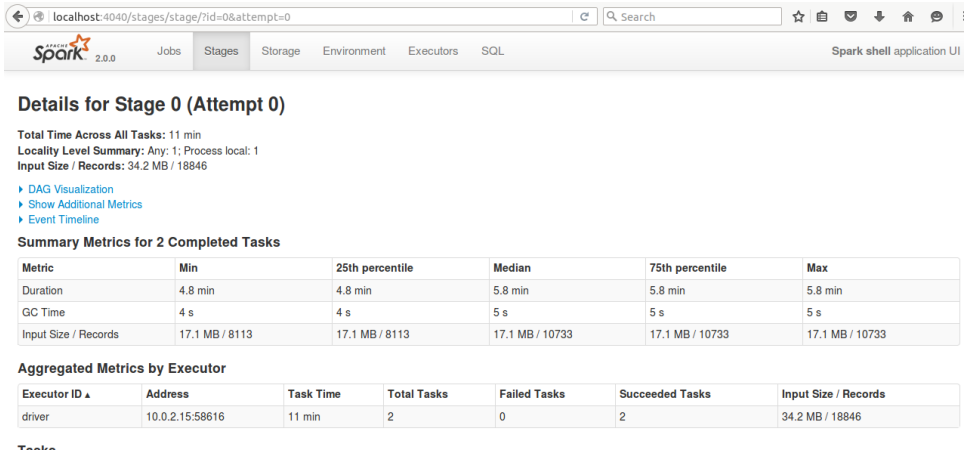
bin/spark-shell --master yarn --driver-memory 1g --executor-memory 3g

```
hduser@ncistudent-VirtualBox:/usr/local/spark$ bin/spark-shell --master local[4]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
16/09/02 00:04:46 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
16/09/02 00:04:47 WARN util.Utils: Your hostname, ncistudent-VirtualBox resolves
 to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface eth0)
16/09/02 00:04:47 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to ano
ther address
16/09/02 00:04:50 WARN spark.SparkContext: Use an existing SparkContext, some co
nfiguration may not take effect.
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[4], app id = local-1472771090074
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.7.0_95)
Type in expressions to have them evaluated.
Type :help for more information.
```

Please take note of address as per above: Spark Context Web UI available at http://10.0.2.15:4041 as this is where we will see our metrics for analysis.

This may be different in your implementation. Enter this address into web browser and you will see the following UI.



3. Run MLlib Algorithms

Staying in the spark shell -> Import required packages.

```
import org.apache.spark.mllib.feature.{HashingTF, IDF}
import org.apache.spark.mllib.linalg.Vector
import org.apache.spark.rdd.RDD
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.classification.{NaiveBayes, NaiveBayesModel}
```

Import data input files from HDFS into an RDD of (String,String)

```
val example = sc.wholeTextFiles("hdfs://127.0.0.1:54310/20newsgroups/*")
```

Manipulate and transform this RDD so that it is in the correct format for running TD-IDF as well as Naïve Bayes

```
val example1 = example.map{case(directory, text) => (text,directory)}
val example2 = example1.map{case(text, directory) =>
(if(directory contains "alt.atheism"){1}
else if(directory contains "comp.graphics"){2}
else if(directory contains "comp.os.ms-windows.misc"){3}
else if(directory contains "comp.sys.ibm.pc.hardware"){4}
else if(directory contains  "comp.sys.mac.hardware"){5}
```

```
else if(directory contains "comp.windows.x"){6}
else if(directory contains "misc.forsale"){7}
else if (directory contains "rec.autos"){8}
else if(directory contains "rec.motorcycles"){9}
else if(directory contains "rec.sport.baseball"){10}
else if(directory contains "rec.sport.hockey"){11}
else if(directory contains "sci.crypt"){12}
else if(directory contains "sci.electronics"){13}
else if(directory contains "sci.med"){14}
else if(directory contains "sci.space"){15}
else if (directory contains "soc.religion.christian"){16}
else if(directory contains "talk.politics.guns"){17}
else if(directory contains "talk.politics.mideast"){18}
else if(directory contains "talk.politics.misc"){19}
else if(directory contains "talk.religion.misc"){20}
else {0}, text)}
```

```
val labels = example2.map{case(labels,text) => labels}
val pretf = example2.map{case(labels,text) => text.split(" ").toSeq}
```

3. TFIDF on Spark

Run IDF and subsequently TFIDF on output from transformations on input data above.

```
val hashingTF = new HashingTF()
pretf.cache()
val tf: RDD[Vector] = hashingTF.transform(pretf)
val idf = new IDF().fit(tf)
```

The latest command will be where we measure our performance of TFIDF. From here we perform 4 runs of the commands which create spark tasks, ignoring the first one and record these results for our

findings. Our results will be taken from localhost:4040 (or respective address as per spark-shell set up).

4. Naïve Bayes on Spark

We will prepare the data further for input into Naïve Bayes

pretf.unpersist()
val tfidf = idf.transform(tf)
val training = labels.zip(tfidf)

Make labeled points for input of Naïve Bayes and run multinomial Naïve Bayes.

val training1 = training.map{case(x,y) => LabeledPoint(x,y)}
training1.cache()
val model = NaiveBayes.train(training1, lambda = 1.0, modelType = "multinomial")

From here we perform 4 runs of the commands which create spark tasks, ignoring the first one and record these results for our findings. Our results will be taken from localhost:4040 (or respective address as per spark-shell set up).