Installation and Configuration of Hadoop

Vaibhav Mishra
Assignment 2
Supervised by: Dr. Shan Suthaharan

March 28, 2018

Abstract

This document will make you understand the installation of Hadoop on a standalone system.

Contents

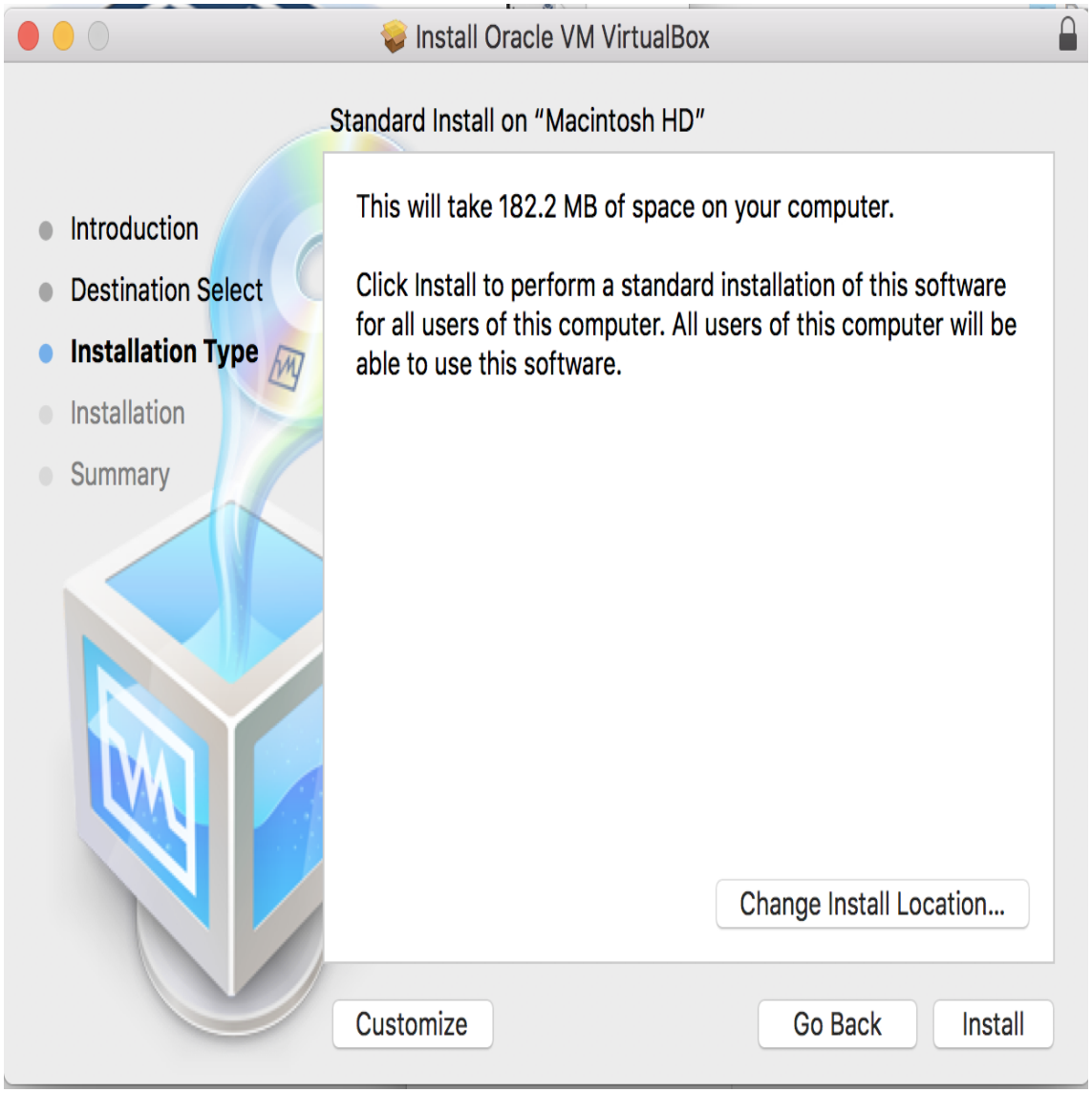
1 Installation of Oracle VirtualBox

Software	VirtualBox
Version	5.2
Download Link	https://www.virtualbox.org/wiki/Downloads
Requirements	At least 1GB of RAM to run the software in addition to what is needed to support

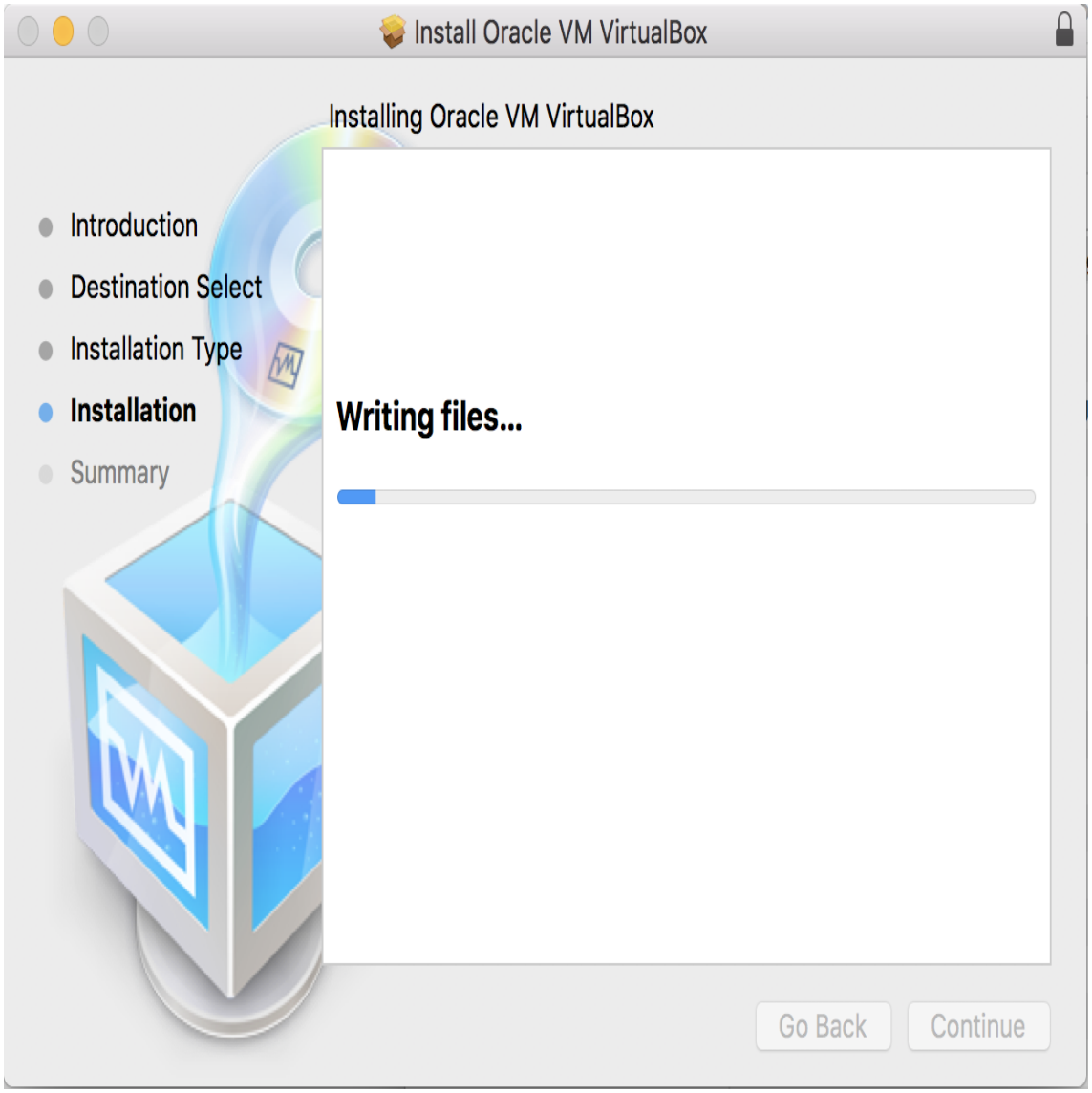
To start the installation process

- Double click on the installation file and select continue. You will see the start up window that is shown in Fig 1.

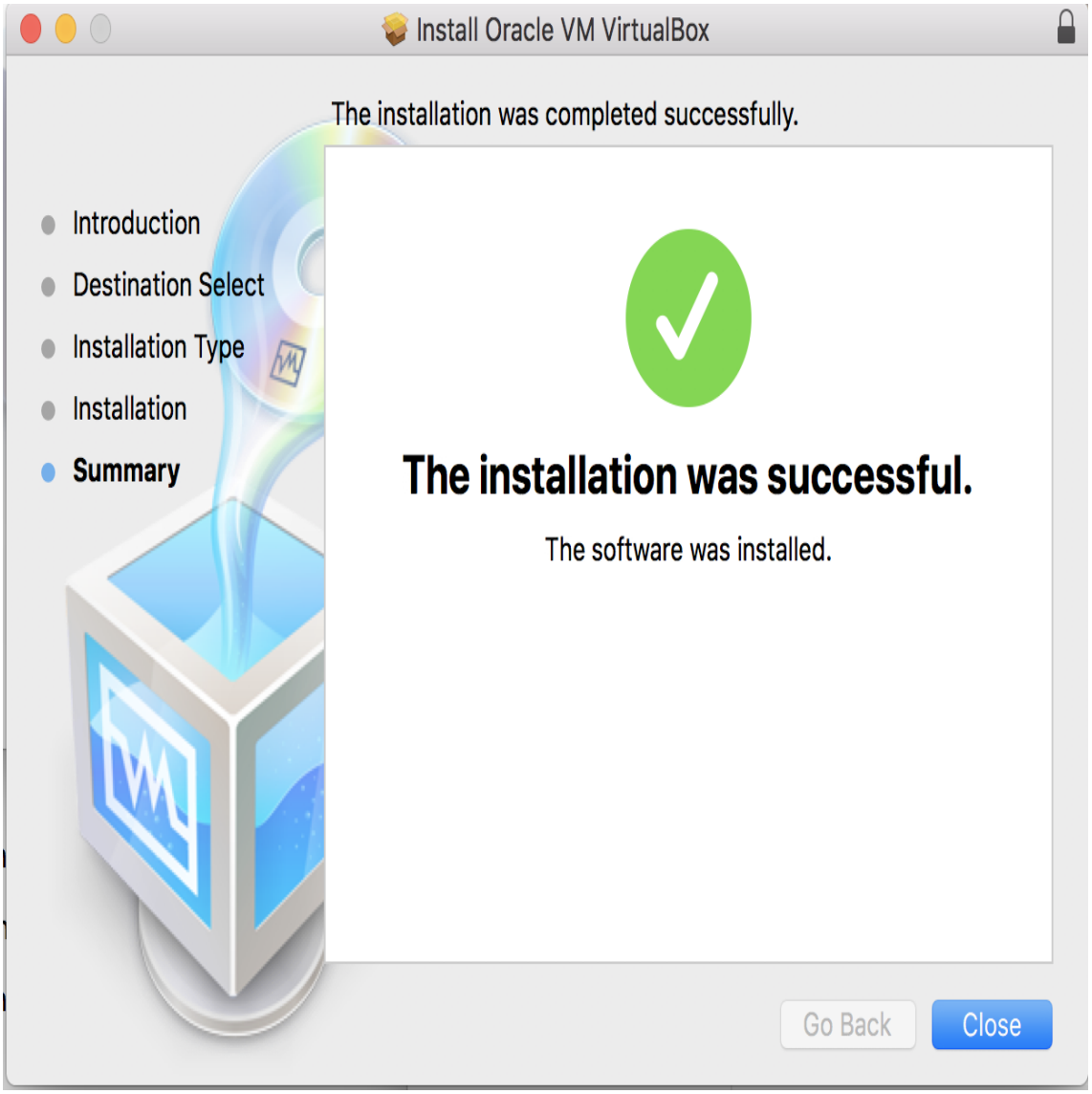
- Select next next as shown in Fig 2.



- Wait for the installation to finish.



- Once installation is finish click on close button.

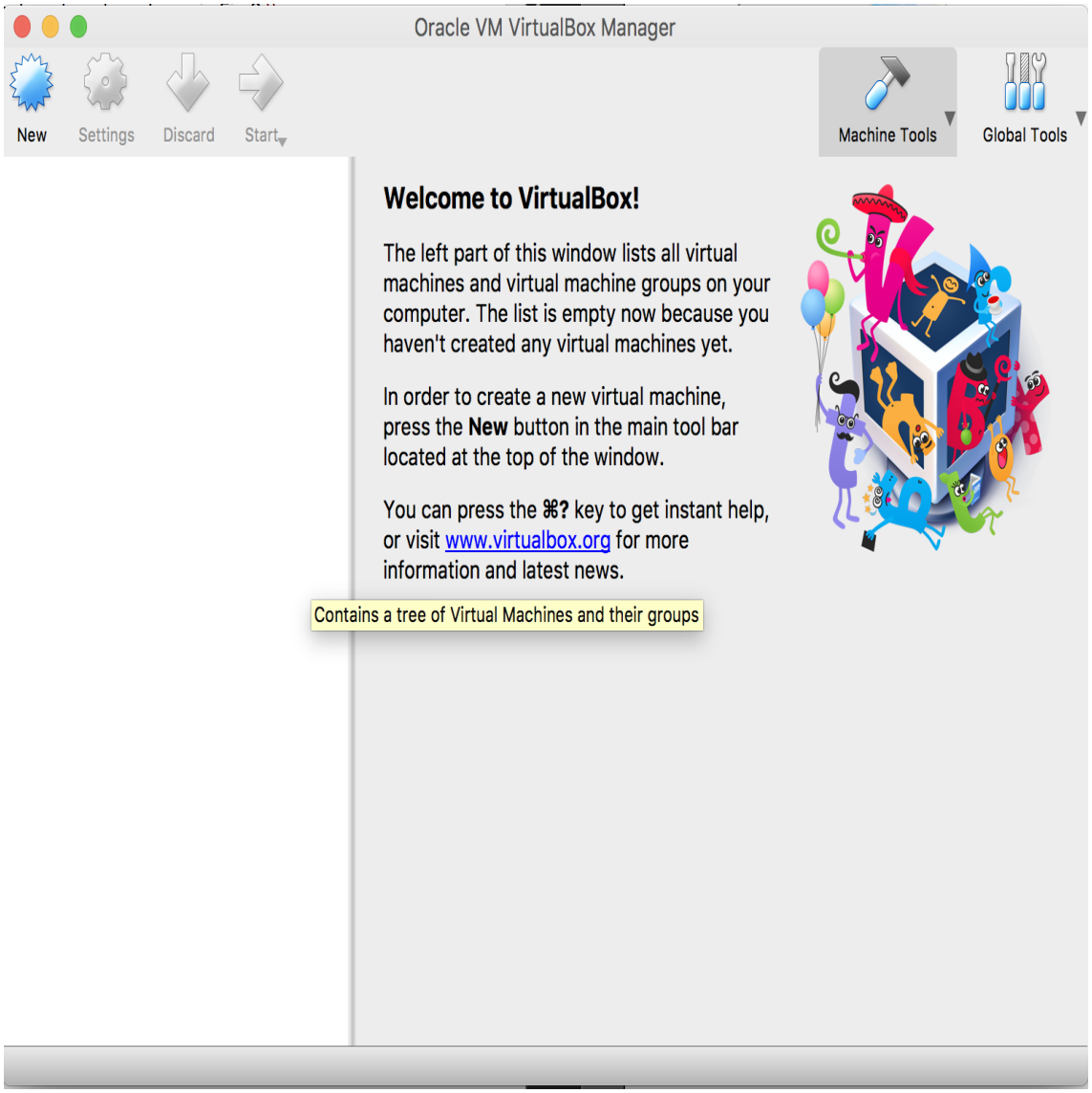


2 Installation of Ubuntu Virtual Machine

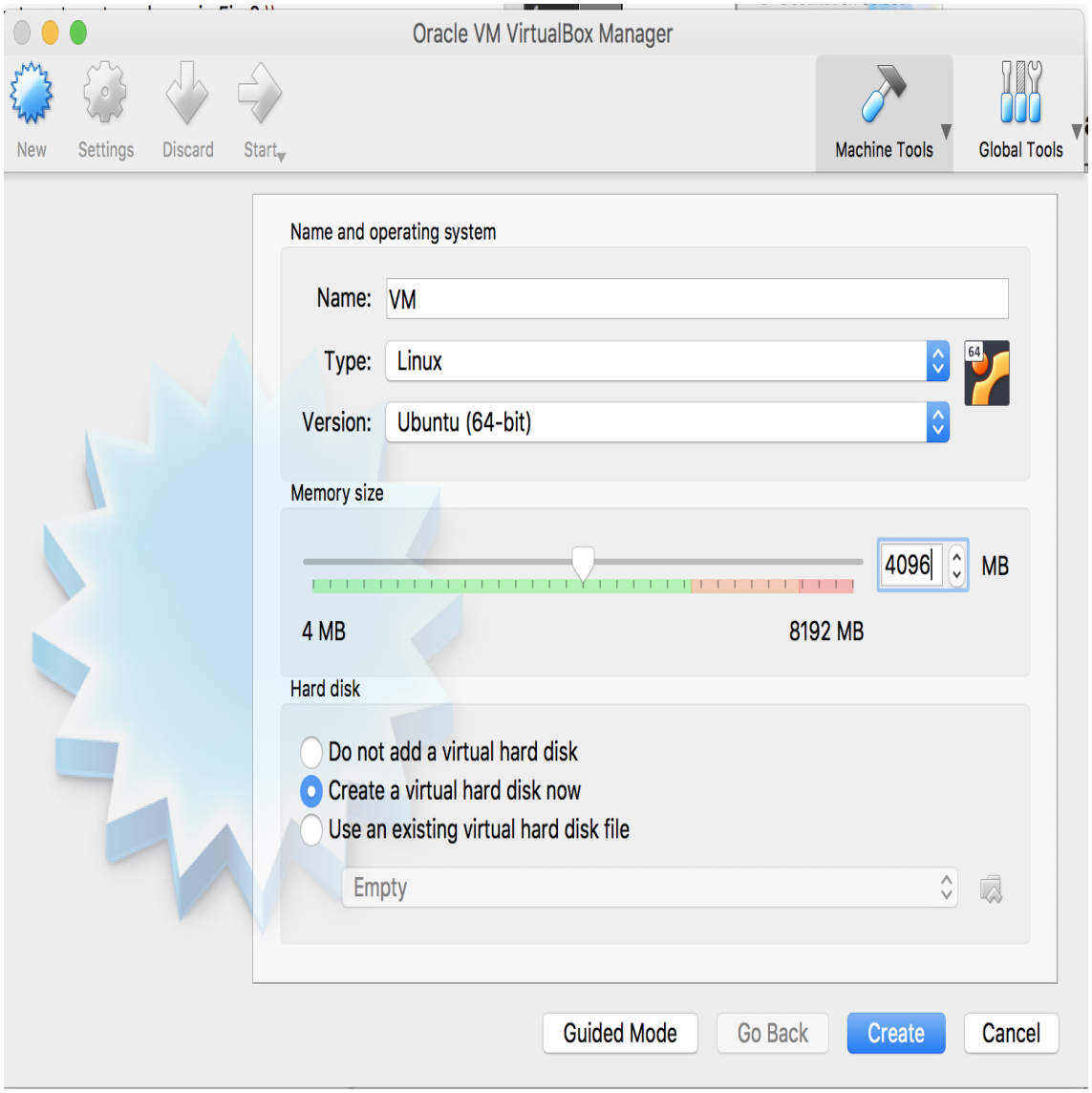
Software	Ubuntu
Version	16.04
Download Link	https://www.ubuntu.com/download
Requirements	2 GHz dual core processor or better processes

For the Ubuntu installation please follow the following steps.

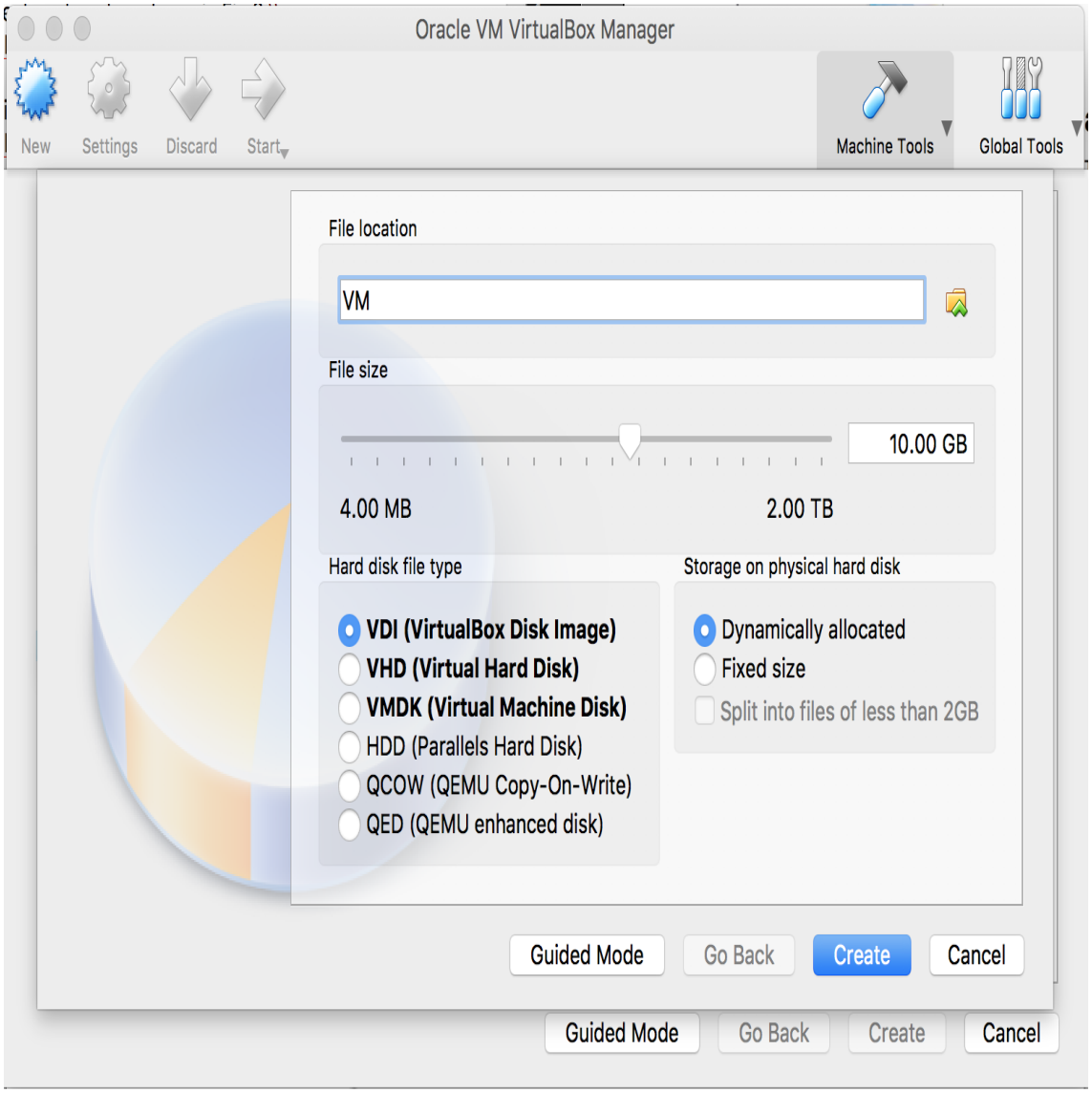
- Launch the virtualBox and select new on virtualBox.



- Provide name for the machine, Type of operating system (in our case it is linux), version of linux to be installed, Provide the RAM size, Let default option be selected and click on create button.

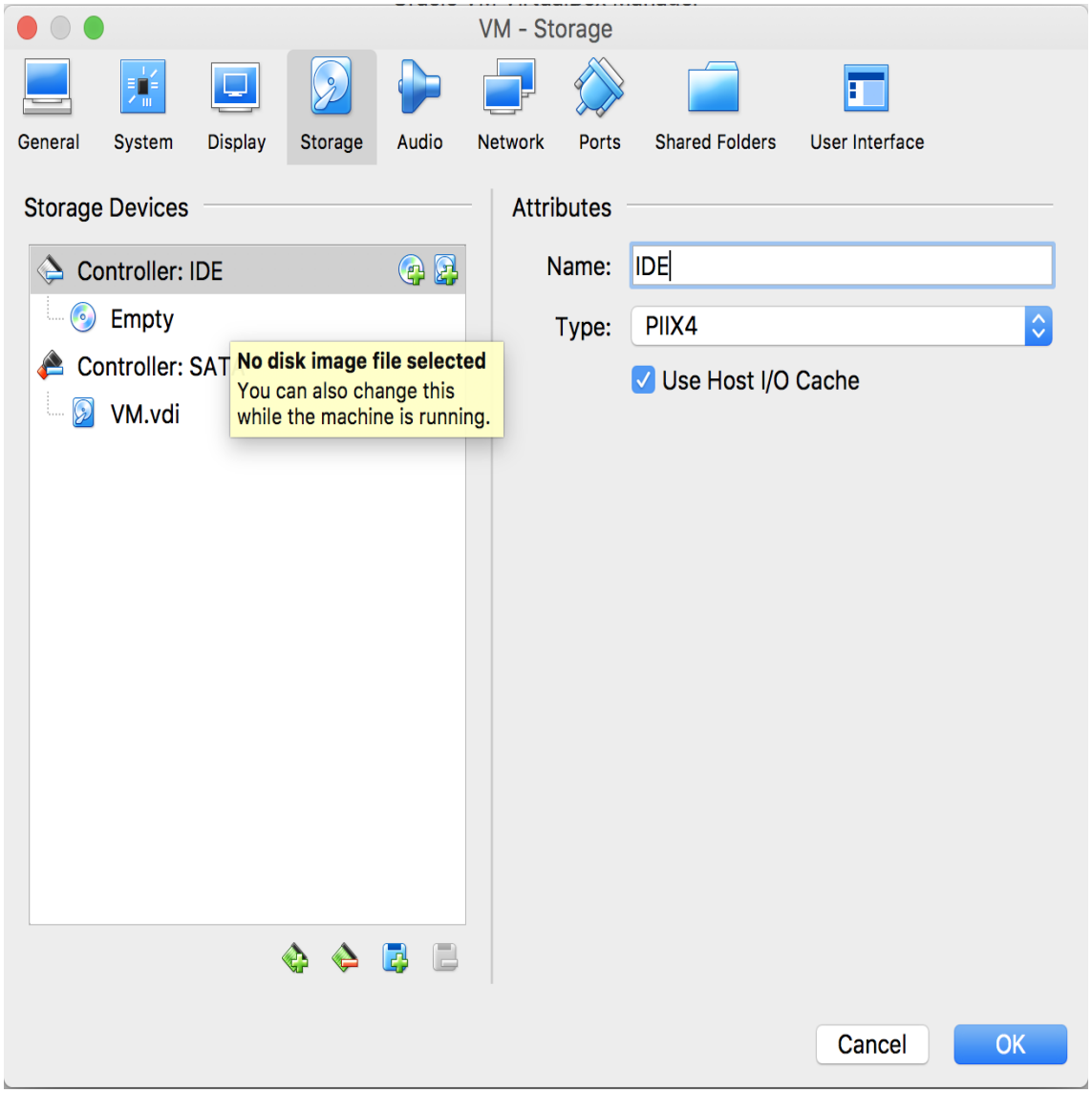


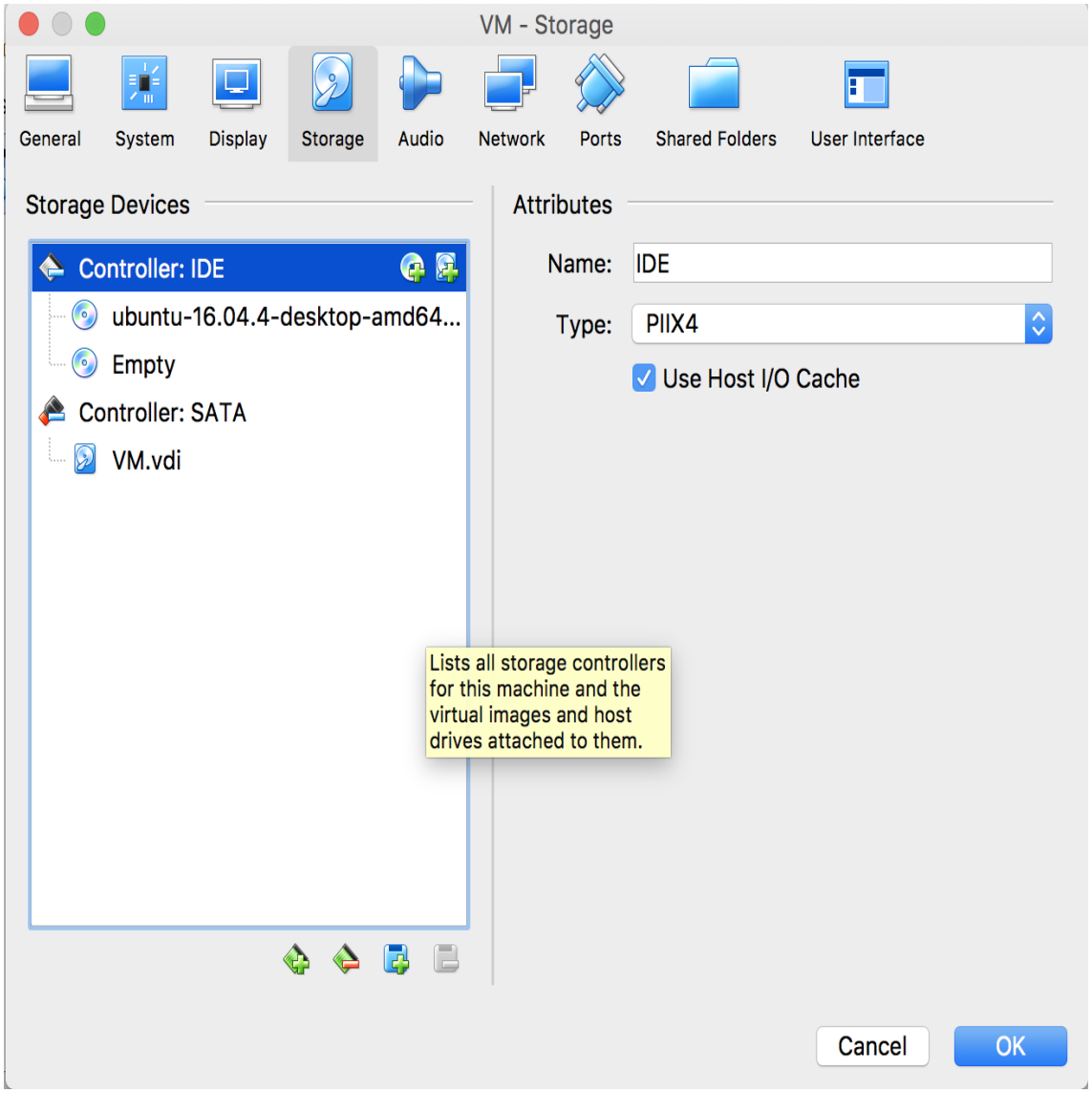
- Provide file location and let all the default option be selected and click on create.



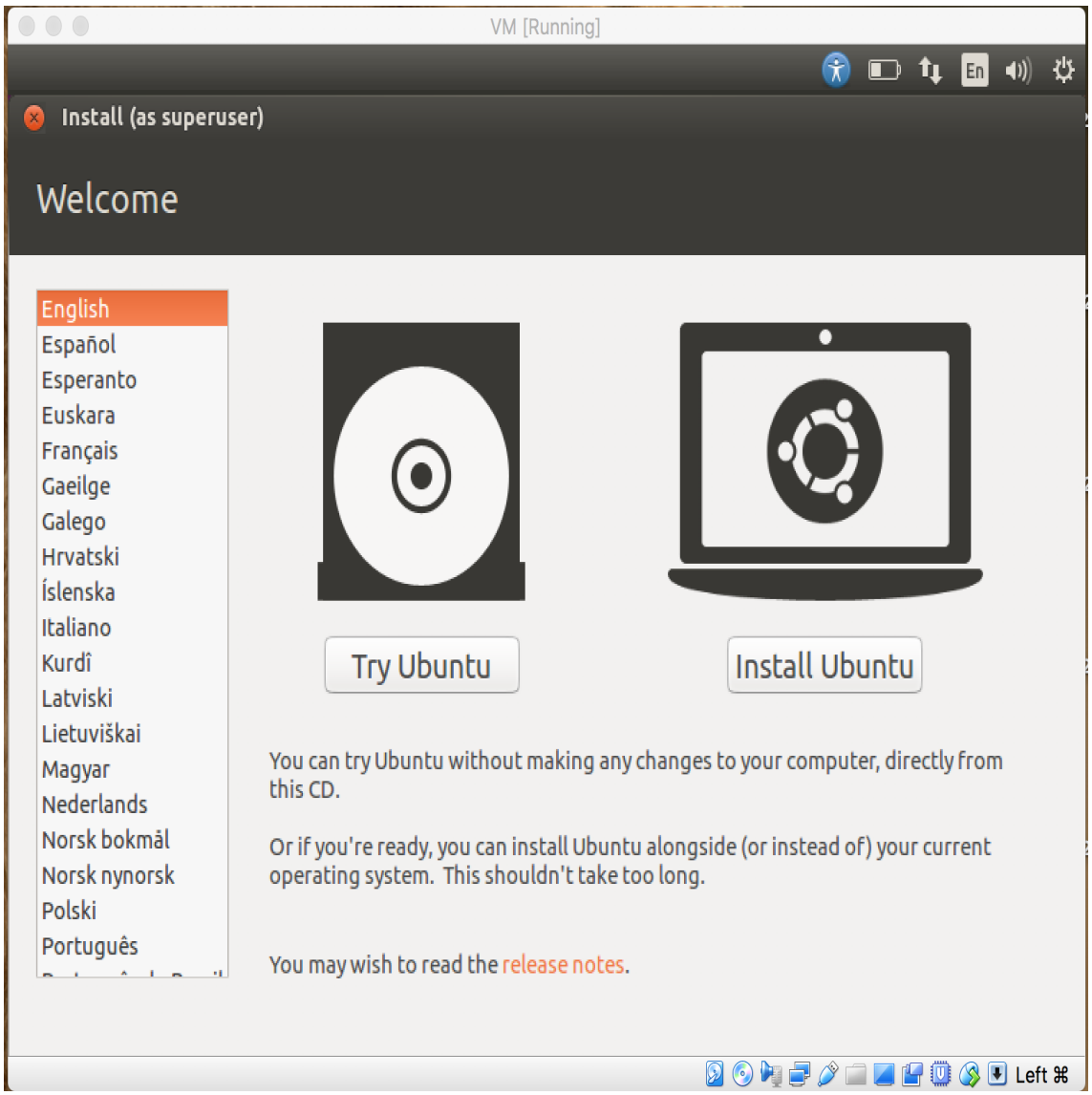
- Now newly machine will be added to the virtualBox, Select setting option to add a image file as shown in figure.

- Select settings option and navigate to storage and add a downloaded image to the controller by selecting "+" option.

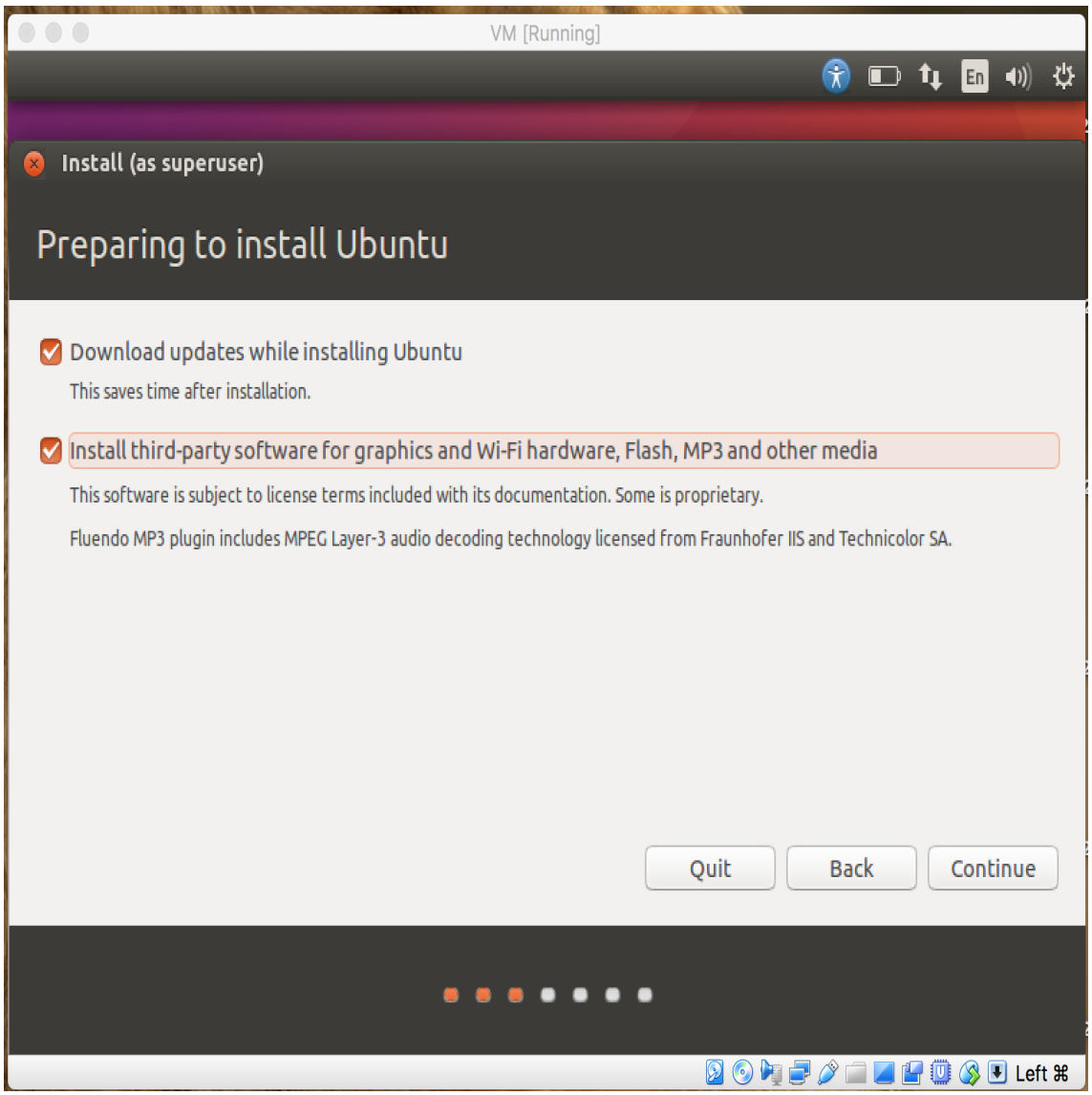

VM - Storage
General
System
Display
Storage
Audio
Network
Ports
Shared Folders
User Interface
Storage Devices
Controller: IDE
Empty
Controller: SAT
VM.vdi
No disk image file selected
You can also change this while the machine is running.
Attributes
Name:
IDE
Type:
PIIX4
Use Host I/O Cache
Cancel
OK



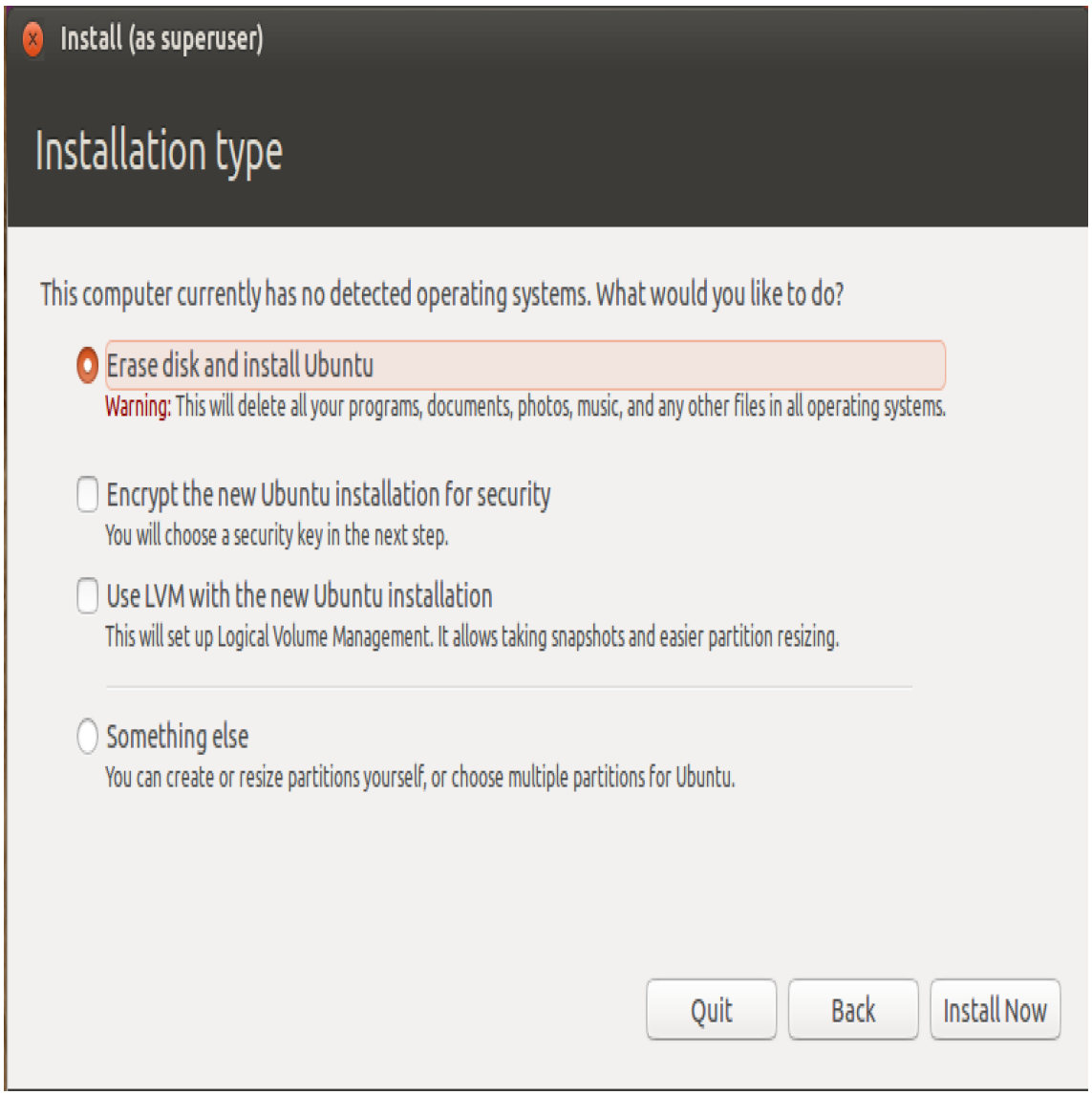
- Start the machine and you should get install option from the list of provided options.



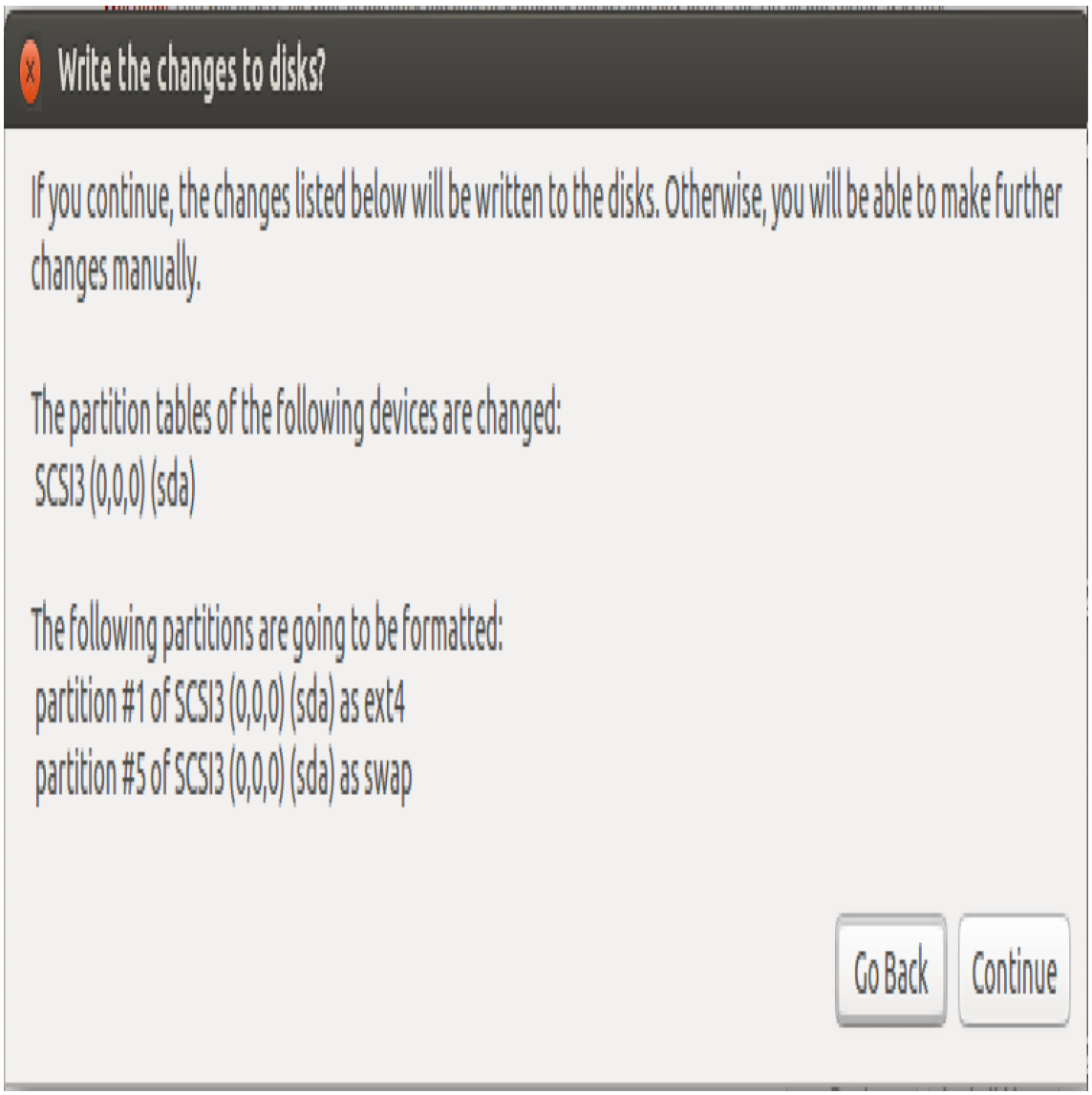
- Select both of the options from the list on next page as shown in figure.



- Select "Erase disk and install Ubuntu" and click on install now as shown in figure.

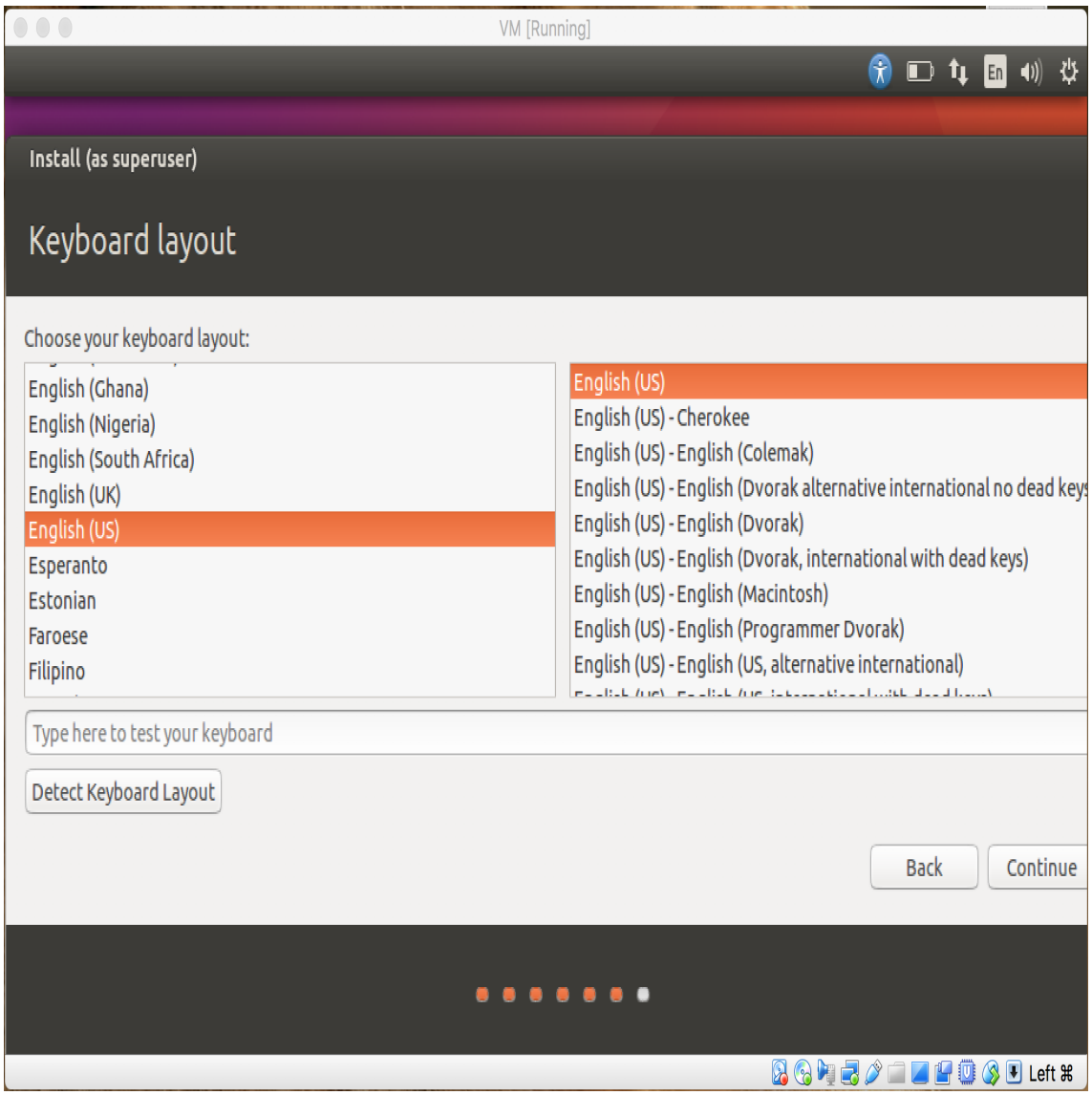


- Select continue on the warning page.

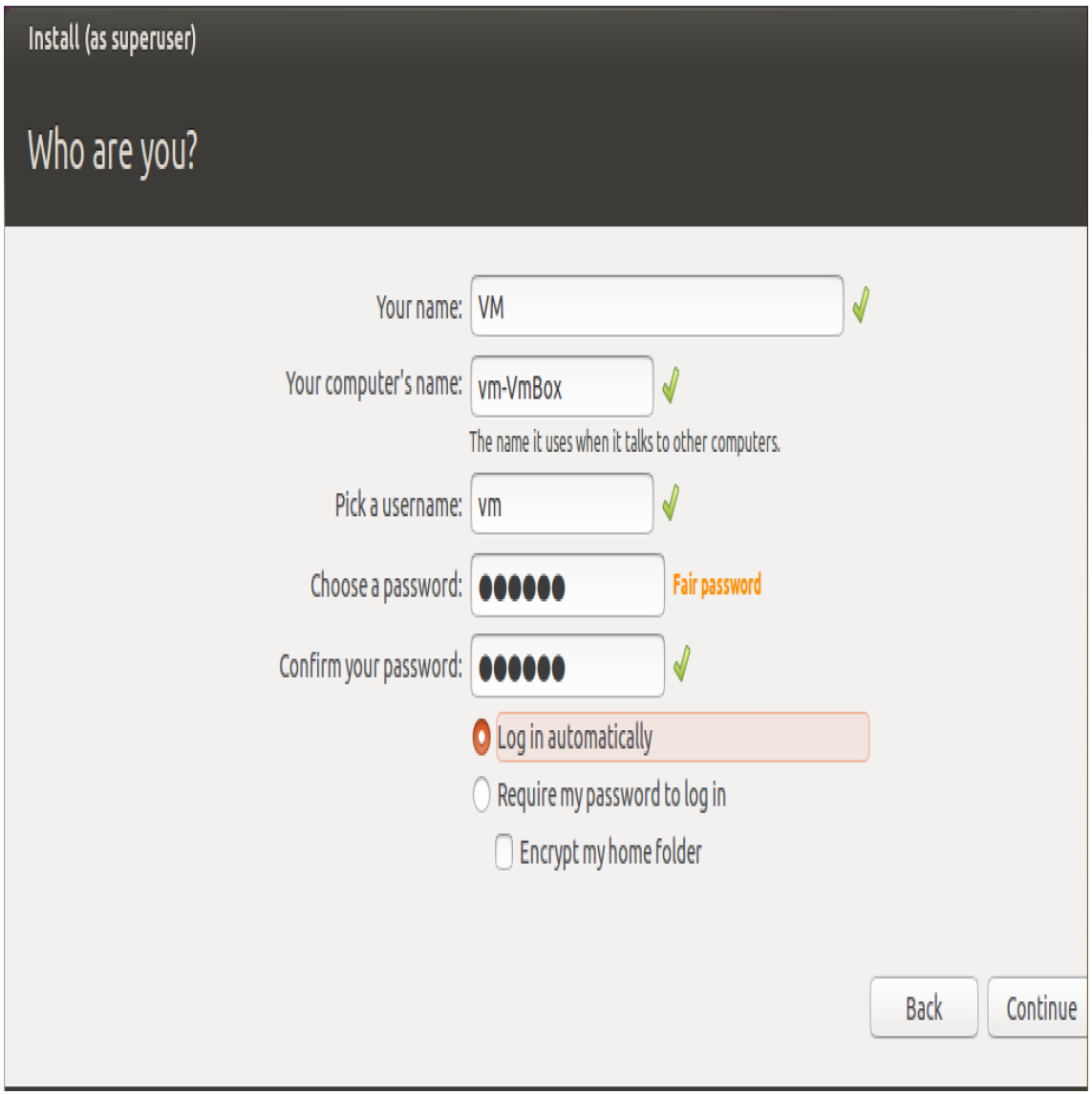


- Provide the time zone information, we have provided new York for "EST".

- Select the keyboard layout from list if provided options.



- Provide the useful information such as username, system name and Password and select continue to complete the installation.



3 HADOOP INSTALLATION

This section refers to the installation settings of Hadoop on a standalone system as well as on a system existing as a node in a cluster.

3.1 SINGLE-NODE INSTALLATION

3.1.1 Running Hadoop on Ubuntu (Single node cluster setup)

The report here will describe the required steps for setting up a single-node Hadoop cluster backed by the Hadoop Distributed File System, running on Ubuntu Linux. Hadoop is a framework written in Java for running applications on large clusters of

commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Hadoops HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications that have large data sets.

Let's understand the meaning of the following:

3.1.2 DataNode:

A DataNode stores data in the Hadoop File System. A functional file system has more than one DataNode, with the data replicated across them.

3.1.3 NameNode:

The NameNode is the centrepiece of an HDFS file system. It keeps the directory of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these file itself.

3.1.4 Jobtracker:

The Jobtracker is the service within hadoop that farms out MapReduce to specific nodes in the cluster, ideally the nodes that have the data, or atleast are in the same rack.

3.1.5 TaskTracker:

A TaskTracker is a node in the cluster that accepts tasks- Map, Reduce and Shuffle operatons from a Job Tracker.

3.1.6 Secondary Namenode:

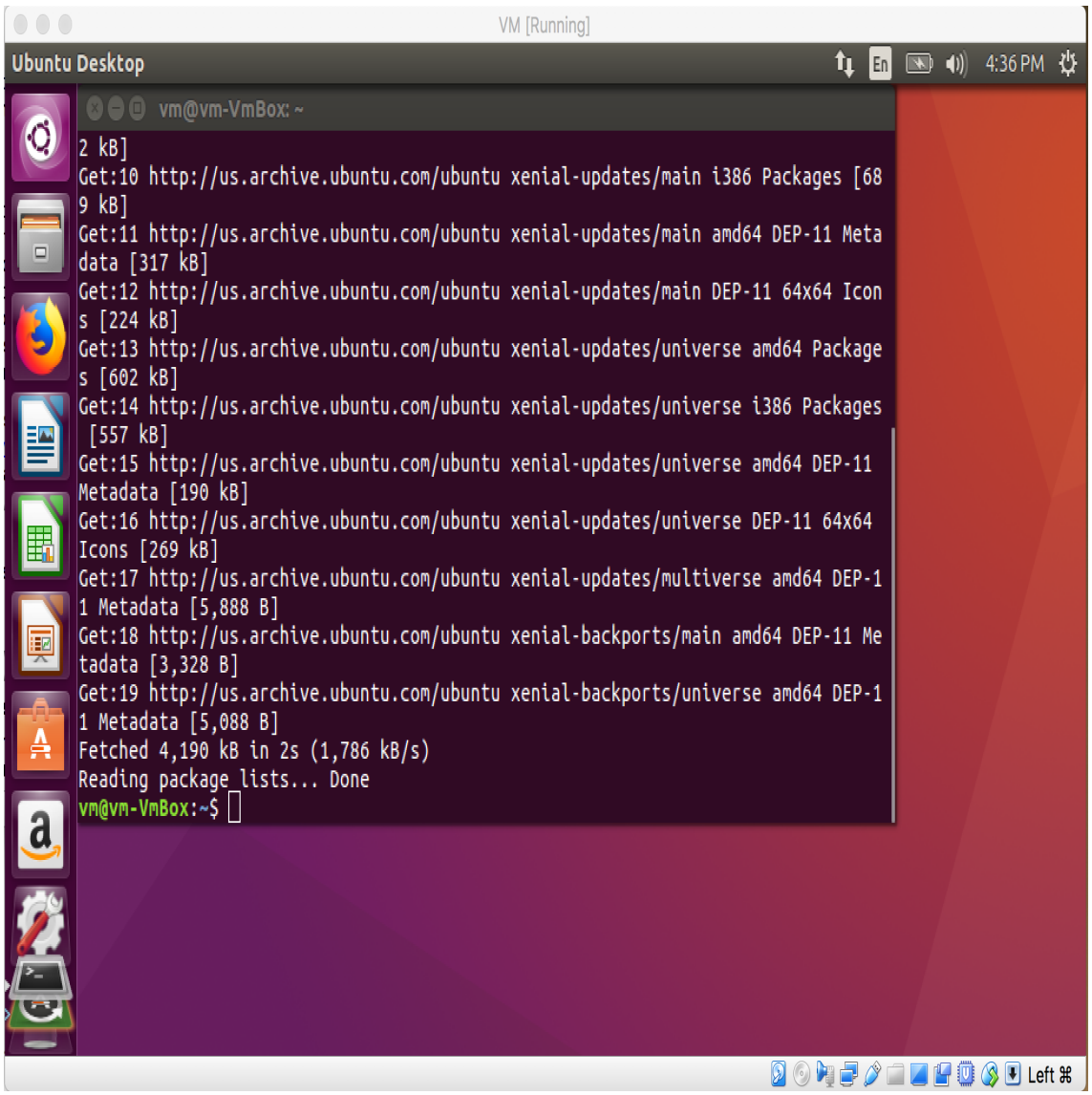
Secondary Namenode whole purpose is to have a checkpoint in HDFS. It is just a helper node for namenode.

3.2 Prerequisites

3.2.1 Java 8 JDK

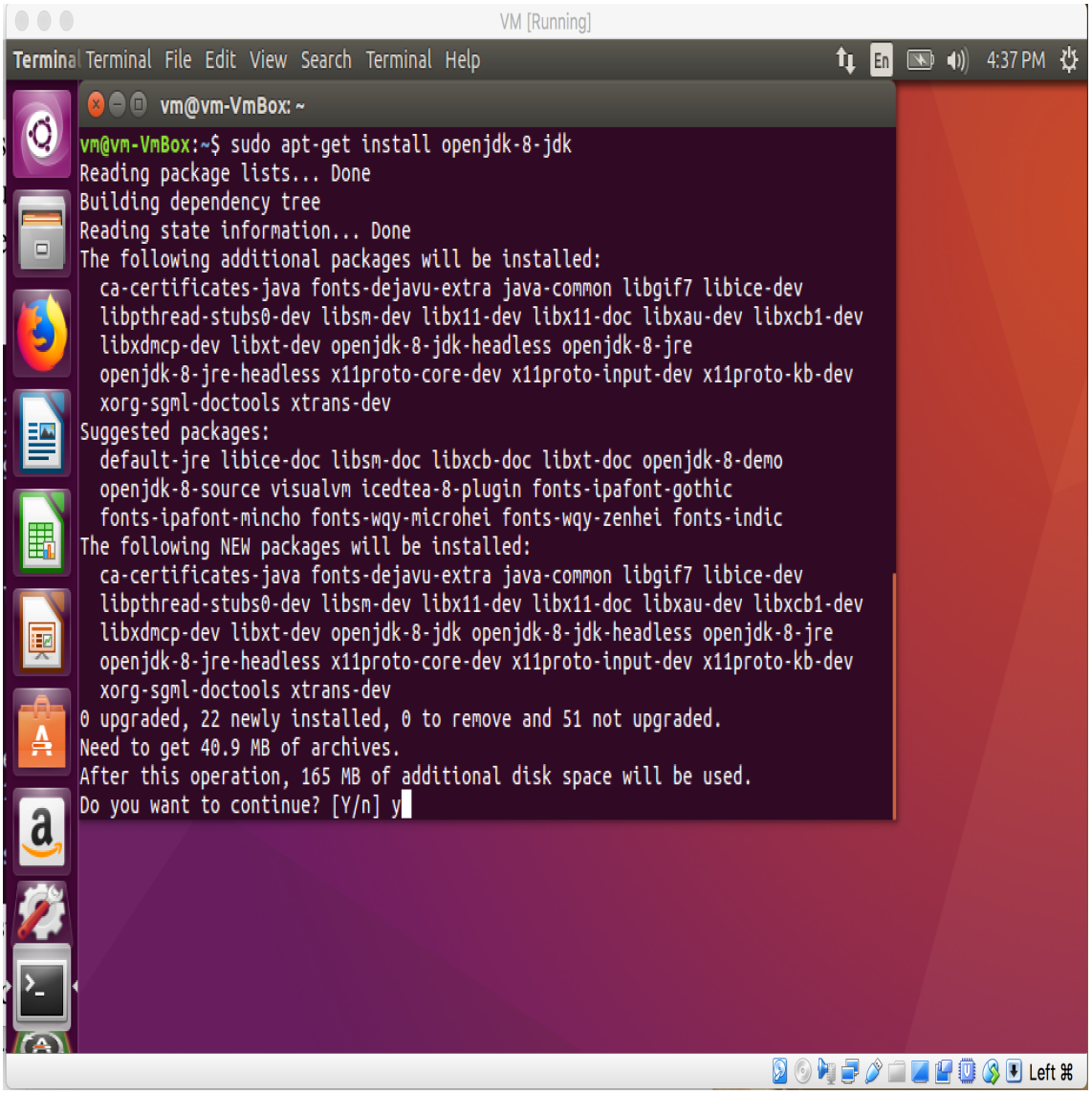
Hadoop requires a working Java 1.5+ (aka Java 5) installation.
Update the source list.

```
$ sudo apt-get update
```



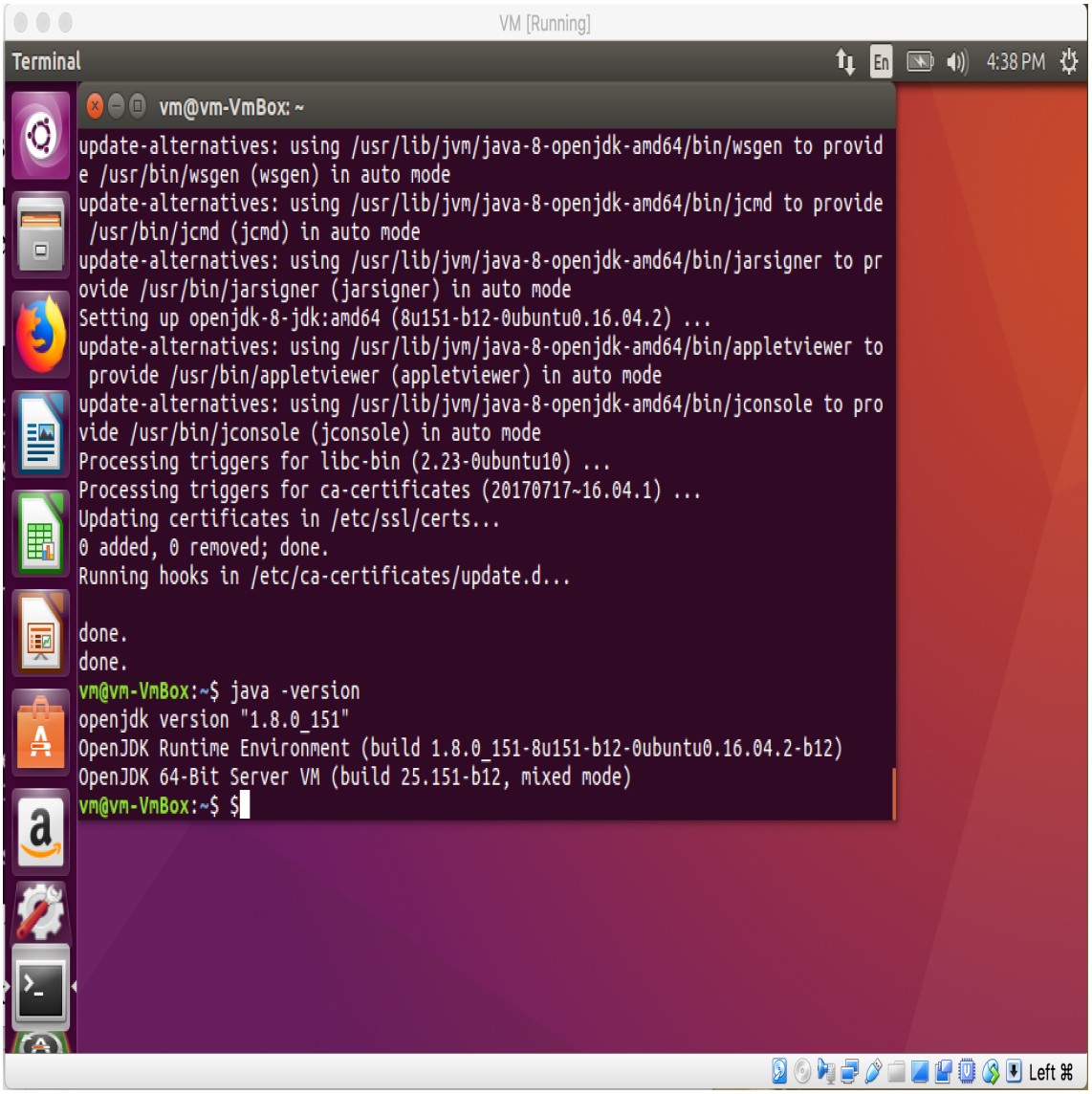
If you already have Java JDK installed on your system, then you need not run the above command. To install it

```
$ sudo apt-get install openjdk-8-jre
$ sudo apt-get install openjdk-8-jdk
```



The full JDK which will be placed in /usr/lib/jvm/java-8-openjdk-amd64 After installation, check whether java JDK is correctly installed or not, with the following command

```
$ java -version
```



3.3 Adding a dedicated Hadoop system user

We will use a dedicated Hadoop user account for running Hadoop. Please free to install it using "root" user.
Following commands can be used to create user and provide

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

We need access to the jdk file. Therefore, we use the following command. Later, we create other short hands for starting the Hadoops services.

```
$ cd /usr/lib/jvm
$ sudo ln -s java-8-openjdk-amd64 jdk
```

3.4 Configuring SSH

The hadoop control scripts rely on SSH to peform cluster-wide operations. For example, there is a script for stopping and starting all the daemons in the clusters. To work seamlessly, SSH needs to be setup to allow password-less login. for the hadoop user from machines in the cluster. The simplest way to achive this is to generate a public/private key pair, and it will be shared across the cluster. Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost for the hduser user we created in the earlier.

To install ssh following commands can be used:

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

SSH key can be generated for hduser using following command

```
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
```

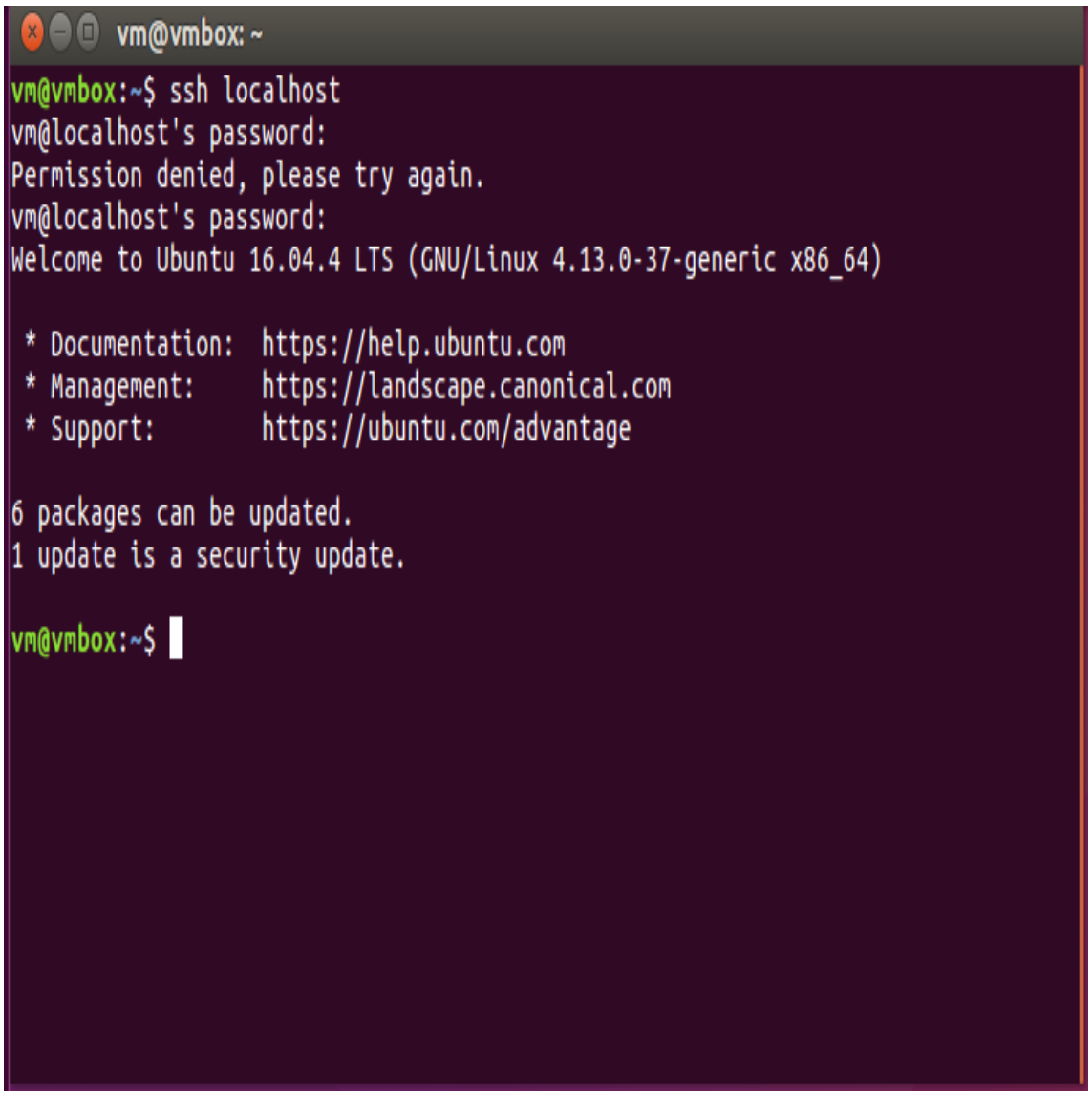
This key should be authorized so that you can access ssh on machine.

```
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

SSH can be testes using following command:

```
$ ssh localhost
```

on successful installation you should get following output:



If the SSH connection fails, we can try the following (optional):

- Enable debugging with ssh -vvv localhost and investigate the error in detail.
- Check the SSH server configuration in

  S **cd** /etc/ssh/sshd_config

  If you made any changes to the SSH server

  $ config- uration

  file, you can force a configuration reload with

  $ sudo /etc/init.d/ssh

  reload.

3.5 Installation of HADOOP using hduser

- Startthe process by switching to hduser

```
$ su - hduser
```

- Just navigate to download folder or you can create your folder to download all the binaries

- Now, download and extract Hadoop 2.7.5 using following command:

```
$ wget http://mirrors.koehn.com/apache/hadoop/common/hadoop-2.7.5/hadoo
$ sudo tar vxzf hadoop-2.7.1.tar.gz –C /usr/local
```

- Now, move to the folder of Hadoop and setup the ownership and permissions.

```
$ cd /usr/local
$ sudo mv hadoop-2.7.5 hadoop
$ sudo chown –R hduser:hadoop hadoop
```

- Setup Environment Variables for Hadoop.
  Add the following entries to .bashrc file

```
$ sudo nano ~/.bashrc
$ export HADOOP_HOME=/usr/local/hadoop
$ export JAVA_HOME=/usr/lib/jvm/jdk/
$ 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_HOME/lib/native
$ export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

- In case if there is a plan to integrate spark with hadoop following path needs to be updated in .bashrc:

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

3.6 Configuration

Next step will be is to edit different configuration files:

- **Hadoop-env.sh**

  Change the file: conf/hadoop-env.sh

  ```
  $ sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
  $ export JAVA_HOME=/usr/lib/jvm/jdk/
  ```

- **conf/*-site.xml** first create temp folder

  ```
  $ sudo mkdir -p /app/hadoop/tmp
  $ sudo chown hduser:hadoop /app/hadoop/tmp
  $ sudo chmod 750 /app/hadoop/tmp
  ```

  Make following changes to core-site.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3  <!--
4     Licensed under the Apache License, Version 2.0 (the "License
          ");
5     you may not use this file except in compliance with the
          License.
6     You may obtain a copy of the License at
7
8       http://www.apache.org/licenses/LICENSE-2.0
9
10    Unless required by applicable law or agreed to in writing,
          software
11    distributed under the License is distributed on an "AS IS"
          BASIS,
12    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
           or implied.
13    See the License for the specific language governing
          permissions and
14    limitations under the License. See accompanying LICENSE file
          .
15 -->
16
17 <!-- Put site-specific property overrides in this file. -->
18
19 <configuration>
20 <property>
21    <name>hadoop.tmp.dir</name>
22    <value>/app/hadoop/tmp</value>
23    <description>A base for other temporary directories.</
          description>
24 </property>
25 <property>
26     <name>fs.default.name</name>
27     <value>hdfs://hdfs:@localhost:9000/</value>
28 </property>
```

```
</configuration>
```

- **mapred-site.xml**
  Make following changes.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed 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. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
</configuration>
```

- **hdfs-site.xml**
  Make following changes.
  To enable interaction with HDFS, we need directories for datanode and namenode. Use the following commands to create these directories.

```
$ cd ~
$ mkdir -p mydata/hdfs/namenode
$ mkdir -p mydata/hdfs/datanode
```

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed 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. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

The final step is to build the Hadoop file structure with the following command.

```
$ hdfs namenode -format
$ mkdir -p mydata/hdfs/namenode
$ mkdir -p mydata/hdfs/datanode
```

- **yarn-site.xml**

  Make following changes.

```
<?xml version="1.0"?>
<!--
```

```
3    Licensed under the Apache License, Version 2.0 (the "License
         ");
4    you may not use this file except in compliance with the
         License.
5    You may obtain a copy of the License at
6
7      http://www.apache.org/licenses/LICENSE-2.0
8
9    Unless required by applicable law or agreed to in writing,
         software
10   distributed under the License is distributed on an "AS IS"
         BASIS,
11   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
          or implied.
12   See the License for the specific language governing
         permissions and
13   limitations under the License. See accompanying LICENSE file
         .
14 -->
15 <configuration>
16
17 <!-- Site specific YARN configuration properties -->
18 <property>
19     <name>yarn.nodemanager.aux-services</name>
20     <value>mapreduce_shuffle</value>
21 </property>
22 <property>
23     <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class
           </name>
24     <value>org.apache.hadoop.mapred.ShuffleHandler</value>
25 </property>
26 </configuration>
```

4 Hadoop Startup

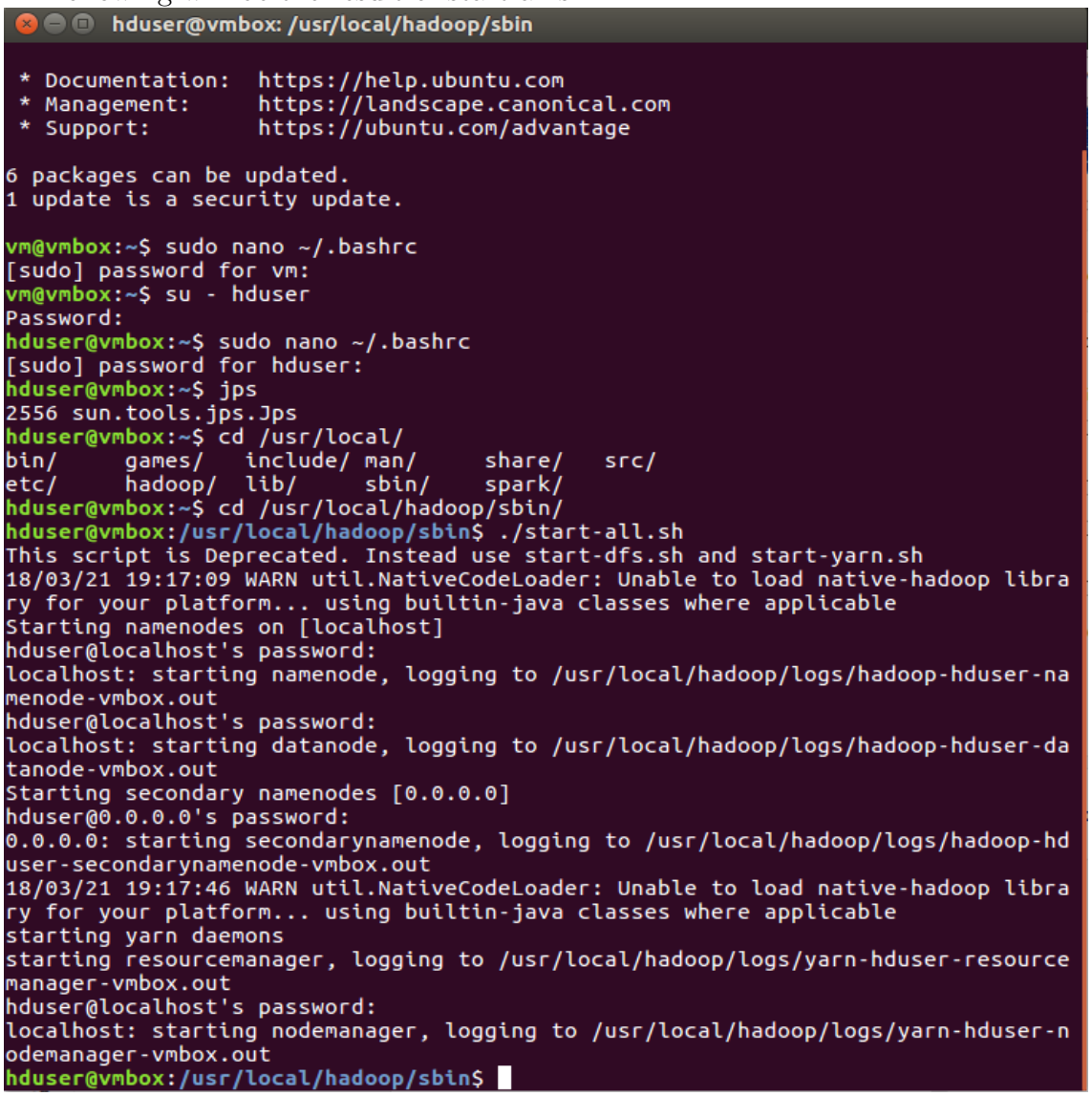
Now, the Hadoop is ready and we can run its services by the following commands:

```
$ cd /
$ cd /usr/local/hadoop/bin
$ start-dfs.sh
$ start-yarn.sh
```

Alternatively, use the following command to run all the services. Once started, you should see

```
$ cd /
$ sd /usr/local/hadoop/bin/start-all.sh
```

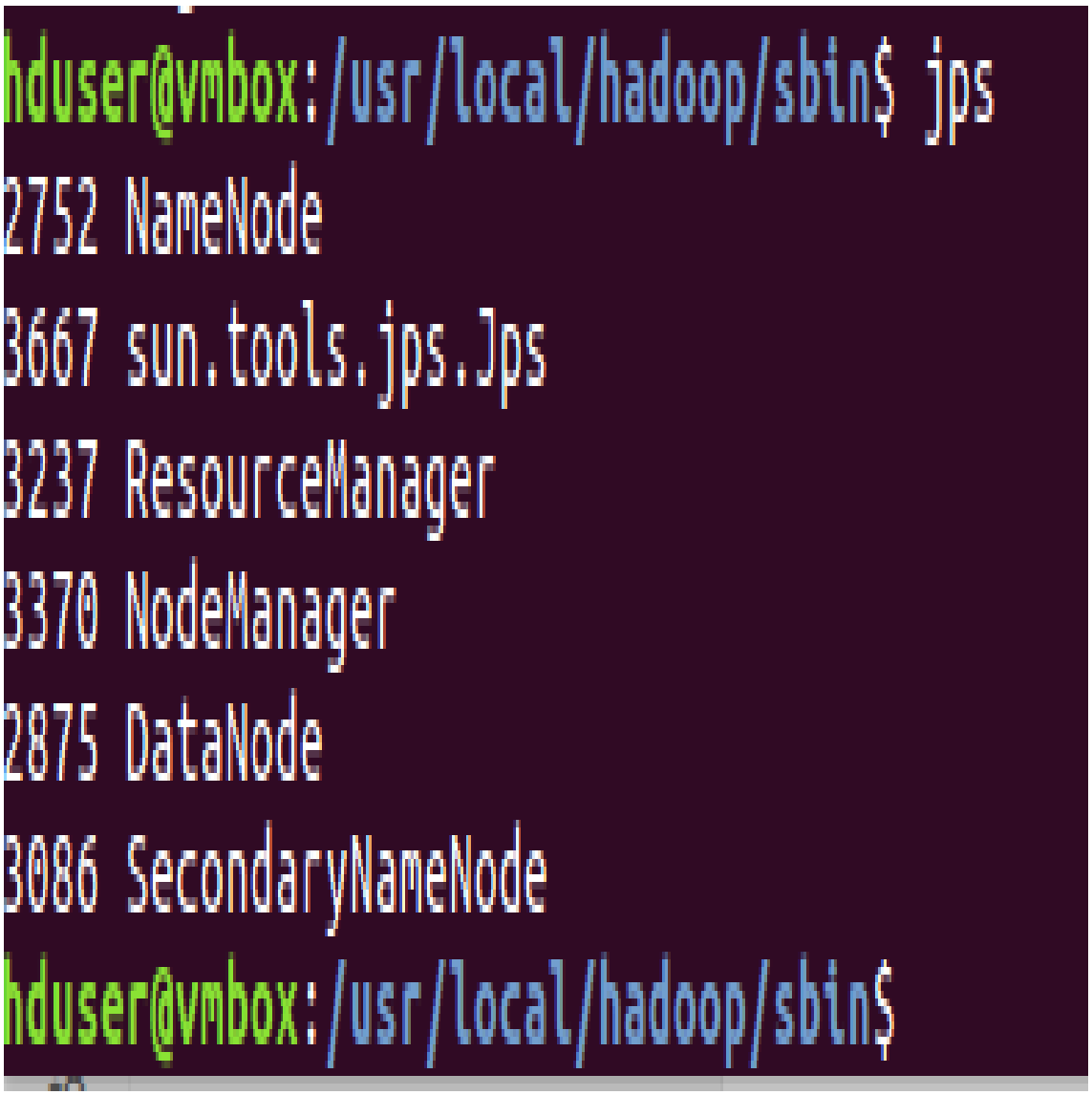
Following will be the result of start-all.sh.



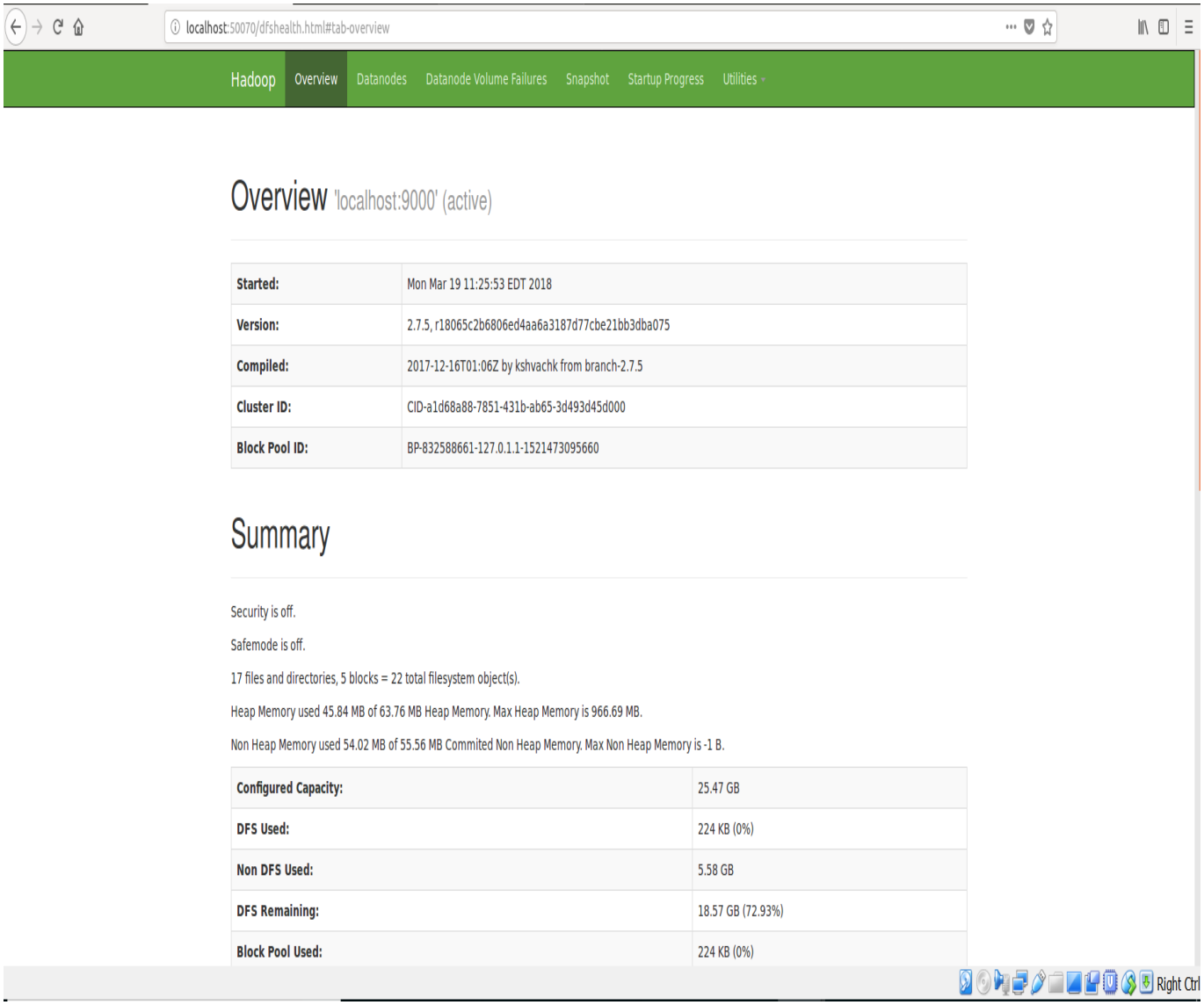
To list all of the services of hadoop following commands can be used

```
$ jps
```

Result will look like:



The final check for the flawless operation of Hadoop on your system is checking the Hadoop web service on this address: http://localhost:50070/



4.1 Testing of MapReduce

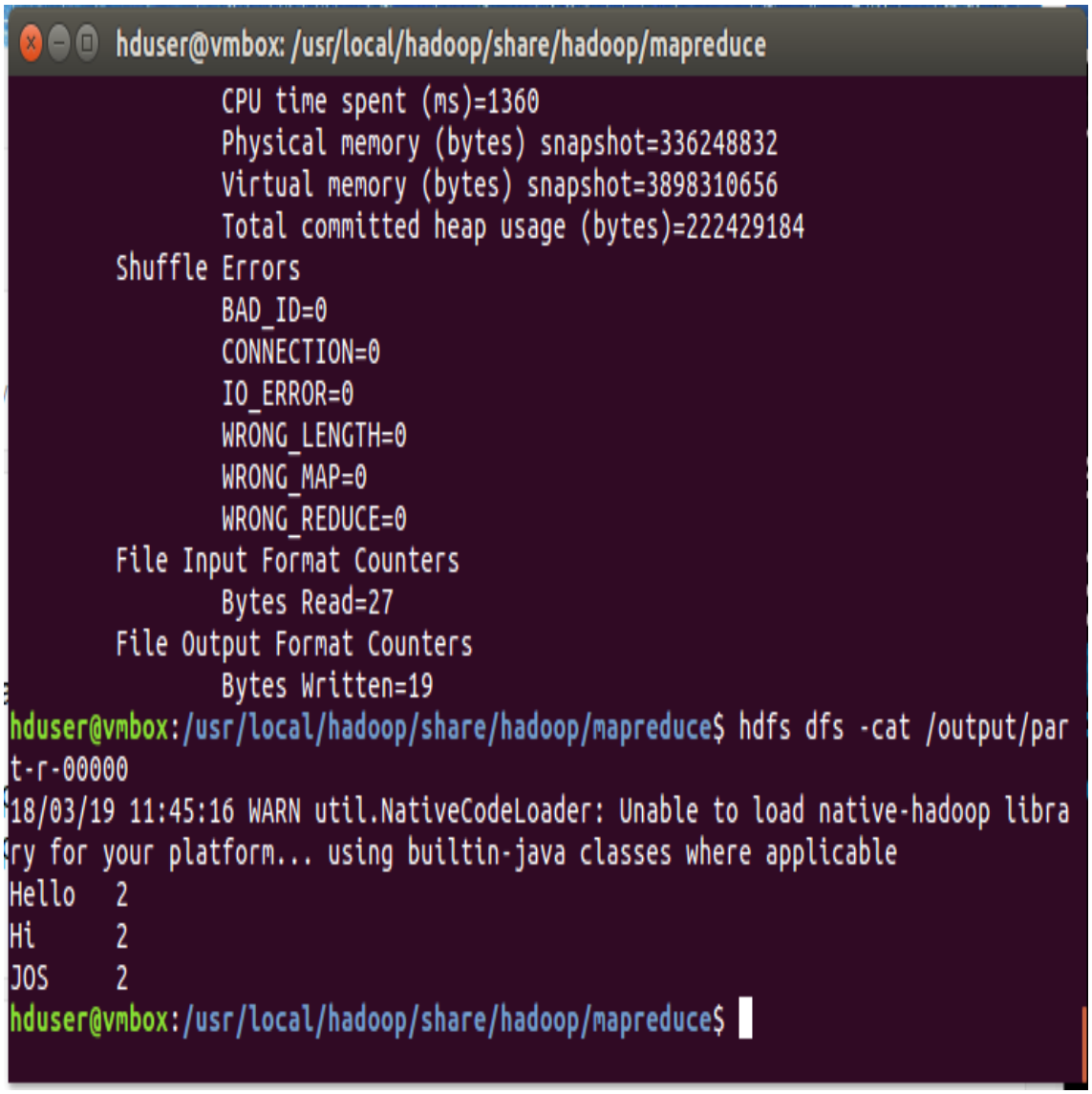
Steps to test Word count program:

- Create a input directory on HDFS using following command :
  `$ hdfs dfs –mkdir /input`
- Navigate to following directory
  `$ cd /usr/local/hadoop/share/hadoop/mapreduce`
- Run following command

```
$ hadoop jar hadoop-mapreduce-examples-2.7.5.jar wordcount /input/text
```

- To check the wordcount program result using following command

```
$ hdfs dfs -cat /output/part-r-00000
```



5 R Integration with Hadoop

When it comes to Statistical Analysis, R is one of the most preferred option and by integrating it with Hadoop, we can successfully use it for Big Data Analytics. In this post, we will be discussing the step-by-step explanation for integrating R with Hadoop and will be performing various operations on HDFS using R console.

RHadoop is a collection of three R packages for providing large data operations with an R environment. RHadoop is available with three main R packages, where each of them offer different Hadoop features:

5.1 Rhdfs:

Rhdfs is an R package that provides the basic connectivity to the Hadoop Distributed File System. R programmers can browse, read, write, and modify files stored in HDFS from within R. Rhdfs package calls the HDFS API in the backend to operate on the data sources stored in the HDFS. This package should be installed only on the node that will run the R client.

5.2 Rmr:

Rmr is an R package that allows R developers to perform Statistical Analysis in R via Hadoops MapReduce functionality on a Hadoop cluster. With the help of this package, the job of a R programmer has been reduced, where they just need to divide their application logic into the map and reduce phases and submit it with the Rmr methods. After that, the Rmr calls the Hadoop streaming MapReduce API with several job parameters such as input directory, output directory, mapper, reducer, and so on, to perform the R MapReduce job over the Hadoop cluster. This package should be installed on every node in the cluster.

5.3 Required Packages for Installing

To install the latest version of R package, CRAN repository should be added to the system. Use the following code for this purpose:

```
$ sudo sh -c 'echo "deb␣http://archive.linux.duke.edu/cran/␣/bin/linux/ubu
```

To install R following commands can be used:

```
$ sudo apt-get update
$ sudo apt-get install r-base
$ sudo apt-get install r-base-dev
```

Now R can be tested using:

```
$ R
```

To install visual studio following commands can be used:

```
$ sudo apt-get install gdebi-core
$ wget https://download2.rstudio.org/rstudio-server-1.0.44-amd64.deb
$ sudo gdebi rstudio-server-1.0.44-amd64.deb
```

The RStudio server is ready to use. To execute RStudio server, I used the http://10.0.2.15:8787.

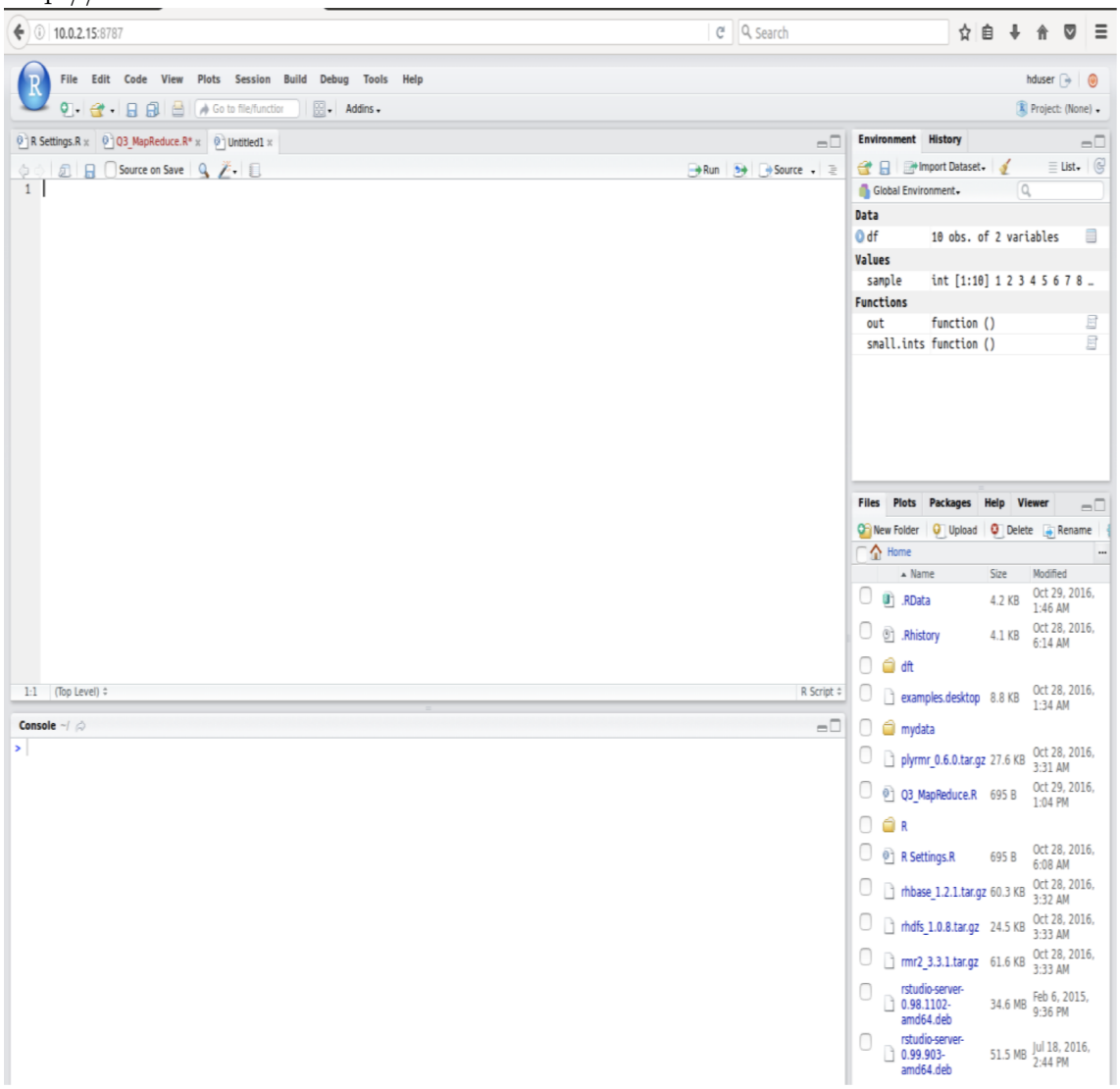


Fig 3. The interface of RStudio Server

We require several R packages to be installed for connecting R with Hadoop. The list of packages are as follows:

- rJava
- RJSONIO
- itertools
- digest
- Rcpp

- httr
- functional
- devtools
- plyr
- reshape2

5.4 Using install.packages from R Console:

```
$ install.packages( c('rJava','RJSONIO', 'itertools',
'digest','Rcpp ','httr','functional',
'devtools', 'plyr','reshape2'),
dependencies=TRUE, repos='http://cran.rstudio.com/')
```

5.5 Downloading Packages and installing through R cmd:

Download the required packages from the below link.
`https://drive.google.com/open?id=0B5dejdhAYHztRkgzbGZOeUdXdVE`

After downloading the packages, extract them and use the below command:

```
$ unzip Rhadoop_packages.zip
```

To install these packages, we will be using R cmd.

```
$ HADOOP_CMD="/usr/local/hadoop/bin/hadoop"
$ R CMD INSTALL <package.rar>
```

5.6 Prerequisite for Rhbase:

Rhabse requires thrift package to be installed for which following link can be followed: `http://thrift-tutorial.readthedocs.io/en/latest/installation.html`

5.7 Rhbase

Rhbase is an R interface for operating the Hadoops HBase data source, stored at the distributed network via a Thrift server. The Rhbase package is designed with several methods for initialization and read/write and table manipulation operations. In this post, we will look in to the Rhdfs package that provides the basic connectivity to the Hadoop Distributed File System. Before delving deeper.

Right now we have struggling with the installation of Rhbase package. working to rectify it :

```
vm@vmbox:~/Downloads$ sudo R CMD INSTALL rhbase_1.2.1.tar.gz
[sudo] password for vm:
* installing to library '/usr/local/lib/R/site-library'
* installing *source* package 'rhbase' ...
** libs
g++ -I/usr/share/R/include -DNDEBUG     -I. -g  -DHAVE_UINTPTR_T -DHAVE_NETDB_H=1 -fpermissive -DHAVE_INTTYPES_H -DHAVE_NETINET_IN_H -I./gen_cpp `pkg-config --cflags thrift` -Wall -fpic  -g -O2 -fstack-pr
otector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -g  -c Hbase.cpp -o Hbase.o
In file included from Hbase.cpp:7:0:
Hbase.h:80:30: error: field 'iface_' has incomplete type 'boost::shared_ptr<apache::hadoop::hbase::thrift::HbaseIf>'
   boost::shared_ptr<HbaseIf> iface_;
                              ^
In file included from /usr/include/boost/throw_exception.hpp:42:0,
                 from /usr/include/boost/numeric/conversion/converter_policies.hpp:16,
                 from /usr/include/boost/numeric/conversion/converter.hpp:14,
                 from /usr/include/boost/numeric/conversion/cast.hpp:33,
                 from /usr/local/include/thrift/transport/TTransportException.h:23,
                 from /usr/local/include/thrift/transport/TTransport.h:25,
                 from /usr/local/include/thrift/protocol/TProtocol.h:28,
                 from /usr/local/include/thrift/TProcessor.h:24,
                 from Hbase.h:10,
                 from Hbase.cpp:7:
/usr/include/boost/exception/exception.hpp:148:11: note: declaration of 'class boost::shared_ptr<apache::hadoop::hbase::thrift::HbaseIf>'
    class shared_ptr;
          ^
In file included from Hbase.cpp:7:0:
Hbase.h:5438:61: error: field 'piprot_' has incomplete type 'boost::shared_ptr<apache::thrift::protocol::TProtocol>'
   boost::shared_ptr< ::apache::thrift::protocol::TProtocol> piprot_;
                                                             ^
In file included from /usr/include/boost/throw_exception.hpp:42:0,
                 from /usr/include/boost/numeric/conversion/converter_policies.hpp:16,
                 from /usr/include/boost/numeric/conversion/converter.hpp:14,
                 from /usr/include/boost/numeric/conversion/cast.hpp:33,
                 from /usr/local/include/thrift/transport/TTransportException.h:23,
                 from /usr/local/include/thrift/transport/TTransport.h:25,
                 from /usr/local/include/thrift/protocol/TProtocol.h:28,
                 from /usr/local/include/thrift/TProcessor.h:24,
                 from Hbase.h:10,
```

References

[1] `https://www.uncg.edu/cmp/downloads/`

[2] Kotipalli, K., Suthaharan, S., 2014. Modeling of class imbalance using an empirical approach with spambase dataset and random forest classification, in: Proceedings of the 3rd annual conference on Research in information technology, ACM. pp. 7580

[3] Suthaharan, S., 2014. Big data classification: Problems and challenges in network intrusion pre- diction with machine learning. ACM SIGMETRICS Performance Evaluation Review 41, 7073

[4] Suthaharan, S., 2015. Machine Learning Models and Algorithms for Big Data Classification: Think- ing with Examples for Effective Learning. volume 36. Springer