# Hadoop Mapreduce

Tanisha Agrawal

A13

Note: Please refer videos for output the screenshots are taken from it.

# STATEMENT:

1. Installation of Apache Hadoop.
2. Creation of two virtual machines and connection using NAT.
3. Implementation of MapReduce method by performing word count program on the client machine using files stored on a server machine.

# 2. ABSTRACT:

Hadoop is an Apache open-source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from a single server to thousands of machines, each offering local computation and storage. In this system, Hadoop is used to demonstrate the word count example using two virtual machines which are connected using Network Address Translator (NAT) and SSH server.

**What is MapReduce in Hadoop?**

**MapReduce** is a software framework and programming model used for processing huge amounts of data. **MapReduce** program work in two phases, namely, Map and Reduce. Map tasks deal with splitting and mapping of data while Reduce tasks shuffle and reduce the data.

Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. The programs of Map Reduce in cloud computing are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster.

The input to each phase is **key-value** pairs. In addition, every programmer needs to specify two functions: **map function** and **reduce function**.

# 3. TOOLS AND LANGUAGES USED:

1. Java: Apache Hadoop works in Java.
2. Virtualbox/VMWare Workstation: VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. It is used to create two virtual instances to demonstrate connectivity and Hadoop.
3. Apache Hadoop: The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

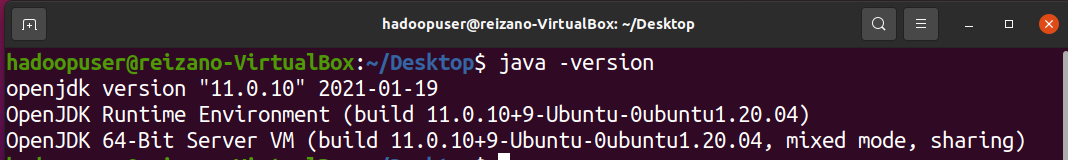
## 3.2 Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows.

Step 1: Download and install Virtualbox.

Step 2: Download target OS’s ISO file from its respective site.

Step 3: Install the particular operating system

## 3.3 Apache Hadoop Installation

1. Prerequisites
   1. Java installation
   2. Creating Dedicated user
   3. Configuring ssh
2. Installation
   1. Downloading and Extracting hadoop binary
   2. bashrc
   3. Changes in hadoop-env.sh and \*-site.xml file
   4. Formatting hdfs
   5. Starting and stopping single-node cluster

### 3.3.1 Prerequisites

#### 3.3.1.a Java installation

Hadoop requires Java 1.5 or above but all the Tutorials available on web insist on Java 1.61[2] and above. For this installation manual Java 11.0.10 is used. Java 11 is available in Ubuntu repository and can be installed using command given in Listing 1



Listing 1: Installing Java 11

Java version can be checked using command in Listing 2 output of command shown in Figure 1



Listing 2: Checking Java Version

Figure 1: Hadoop requires Java 1.5 or higher

#### 3.3.1.b Creating dedicated user

It is advised creating a new dedicated user for using hadoop. New group is created (in this installation report new group created is “hadoop”) and user (in this installation report new user added is “hadoopuser”) can be added to the newly created group using commands in Listing 3



Listing 3: adding ngroup and user for hadoop

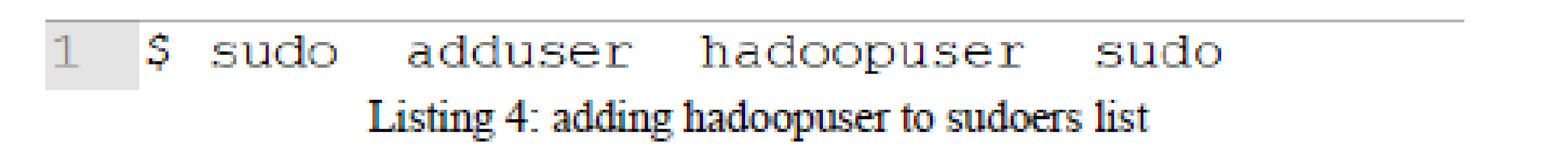
Figure 2 displays the above mentioned commands ‘for creating group and user’ executed on my system. When hadoopuser is added it asks for new UNIX password. This password is password for hadoopuser. Retype the password when prompted and enter the details asked (details are optional). In the end enter ‘y’ to complete the procedure.



Figure 2: Adding group hadoop and user hadoopuser

Some steps require sudo permission.Hence, we grant hadoopuser sudo listing using the command mentioned in Listing 4.





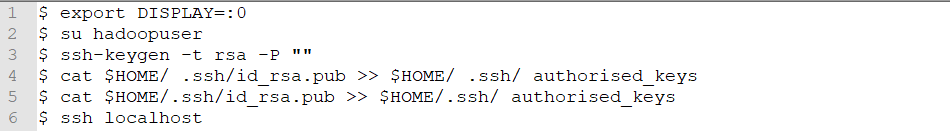
**3.3.1.c Configuring ssh**

Ssh access is required for hadoop to run. In this installation report we will configure ssh access for localhost to user hadoopuser. If ssh server is not installed on the machine, for Ubuntu it can be installed using command given in Listing 5



Listing 5: Installing ssh server

To allow ssh access a SSH Key has to be generated which can be generated for user hadoopuser as follows

Listing 6: Creating keygen and adding localhost to known hosts

The command given above can be explained asWe need to export display while changing from default user to hadoopuser, given in line 2 of Listing 6.

* + 1. Generating keygen, when asked to enter the file to save the key, press enter and key will be saved in default /home/hadoopuser/.ssh/id rsa file, given in line 2 of Listing 6.
    2. Authorizing public key generated as in line 3 of Listing 6.
    3. Adding localhost to list of known hosts using ssh, when prompted for ‘yes/no’, write ‘yes’ and press enter, given in line 4 of Listing 6.
    4. all the above steps is carried out by hadoopuser.

Figure 3 shows the configuration steps for ssh executed on my system.

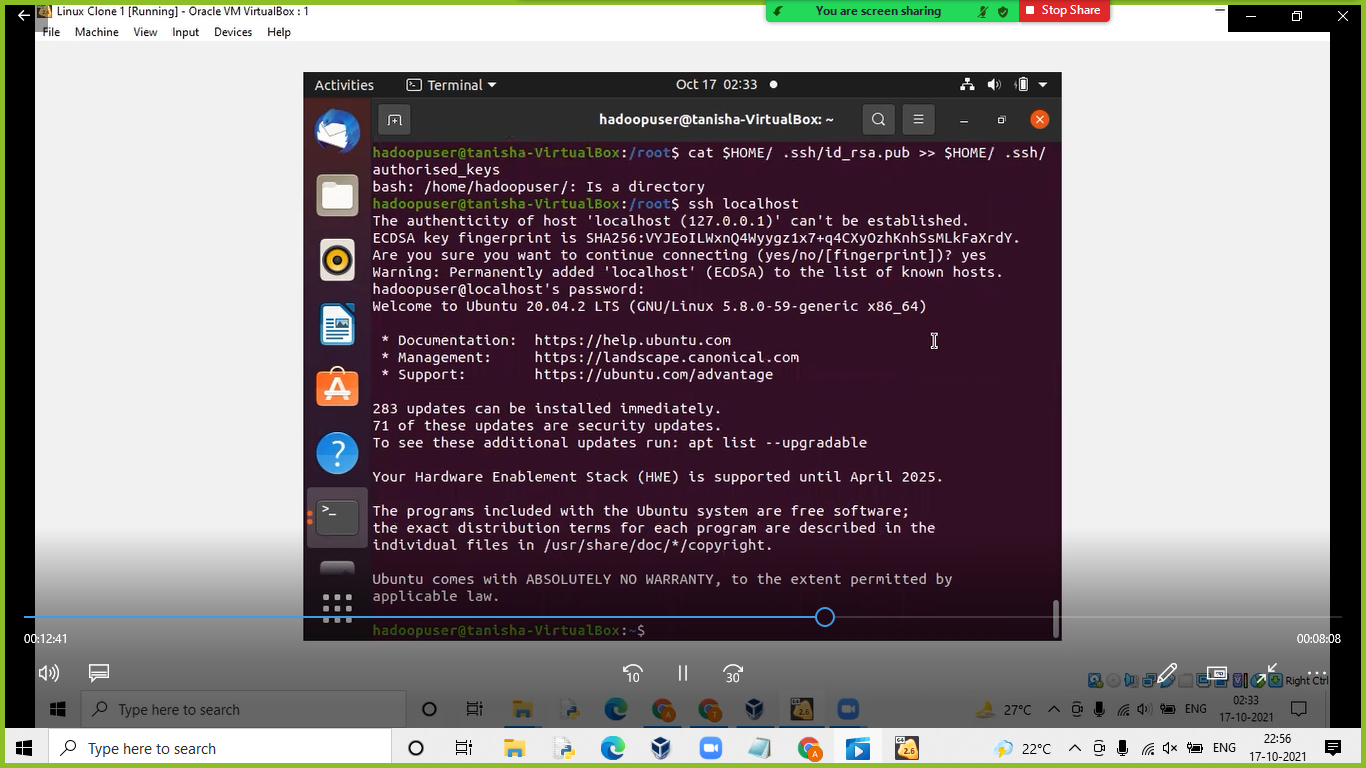
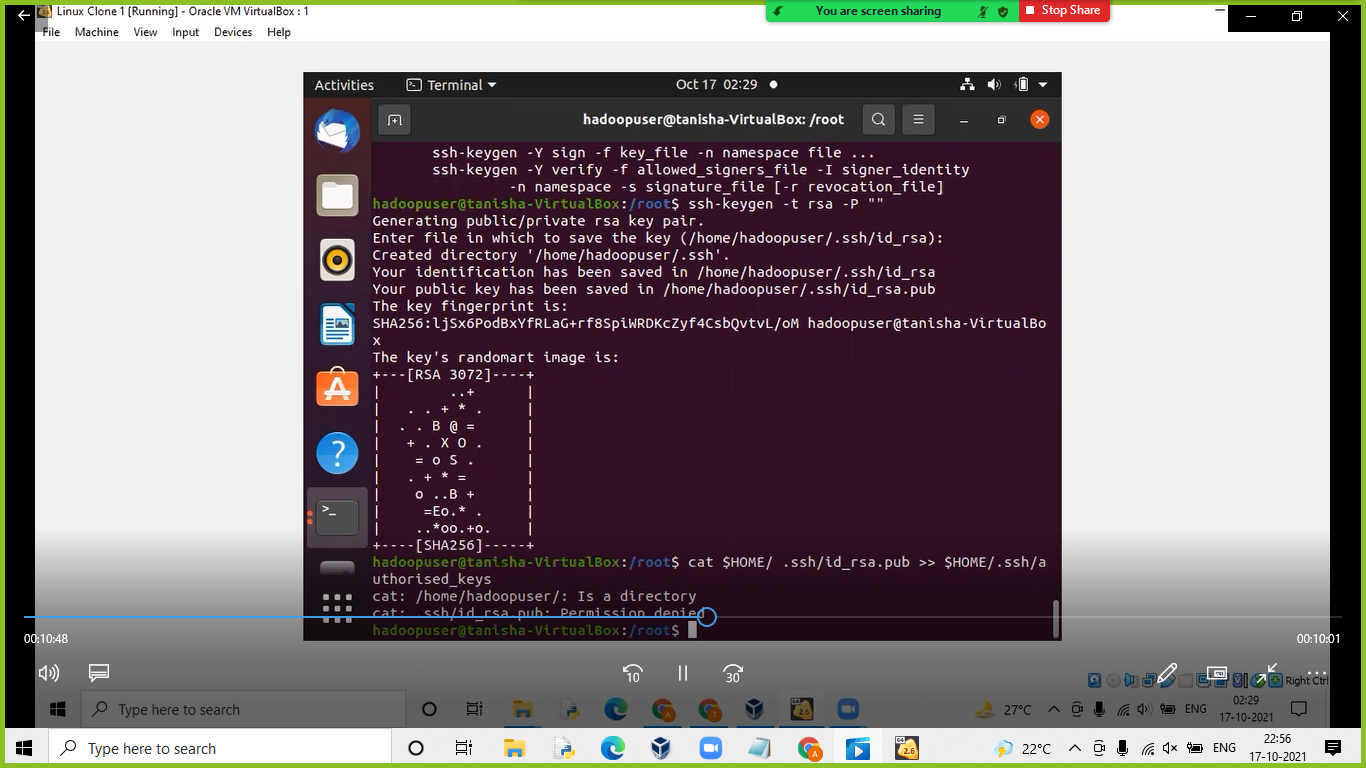
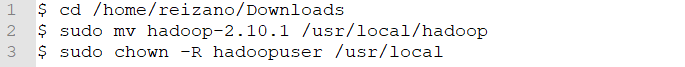


Figure3:Configuring SSH server.

### 3.3.2 Installation

#### 3.3.2.a Hadoop’s folder

Download hadoop-2.10.1 from the official website([here](https://hadoop.apache.org/releases.html)). After completing the download untar (extract) the .tar.gz file. Move the hadoop-2.10.1 folder in a temporary folder used by hadoop’s hdfs file system, In our case it is ‘hadoop’ in /usr/local folder. After that we change the ownership of hadoop and temporary directory just created. Moving the file in /usr/local and changing the ownership requires sudo permission.



Listing 6: Steps to be followed before using Hadoop

The commands executed is given in figure 5

Figure 5: Steps executed with sudo user

#### 3.3.2.b Updating .bashrc for hadoopuser

We can edit .bashrc file for hadoopuser. The edited .bashrc file is shown below:

1 export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

2 export HADOOP\_HOME=/usr/local/hadoop

3 export PATH=$PATH:$HADOOP\_HOME/bin

4 export PATH=$PATH:$HADOOP\_HOME/sbin

5 export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

6 export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

7 export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

8 export YARN\_HOME=$HADOOP\_HOME

9 export

10 HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

11 export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib"

#### 3.3.2.c Changes in Hadoop folder

In Hadoop’s folder we have to edit a few files for hadoop to run. The files can be found in /usr/local/hadoop/etc/hadoop directory. The files are hadoop-env.sh, core- site.xml, hdfs-site.xml, yarn-site.xml and mapred-site.xml. This changes can be done using user ‘hadoopuser’.

##### 3.3.2.c.i hadoop-env.sh

In hadoop-env.sh we have to define a path for JAVA\_HOME. By default it will be commented and it’s value will be set to ${JAVA\_HOME} as shown in Listing 7, un-comment it and change it’s value to the Java to be used. Original and edited hadoop-env.sh files are given in Listing 7 and Listing 8 respectively.



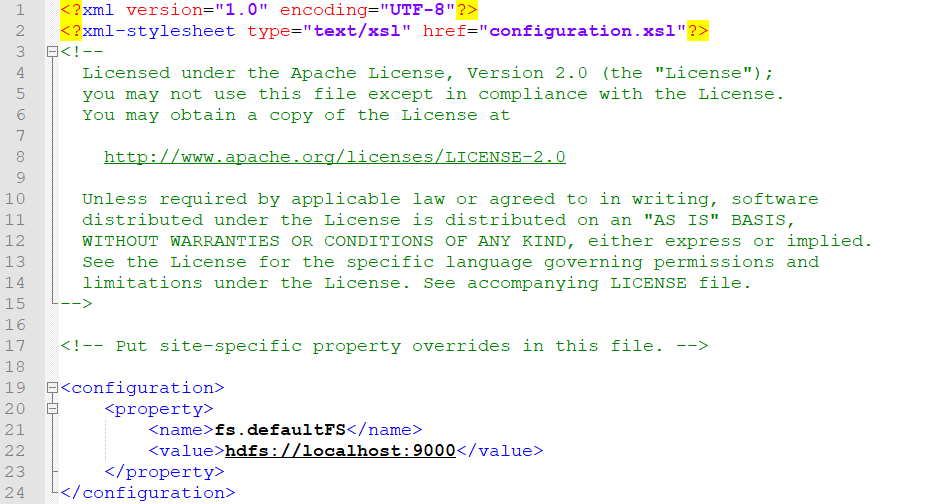
Listing 7: Java path in original hadoop-env.sh



Listing 8: Java path provided in hdfs-env.sh

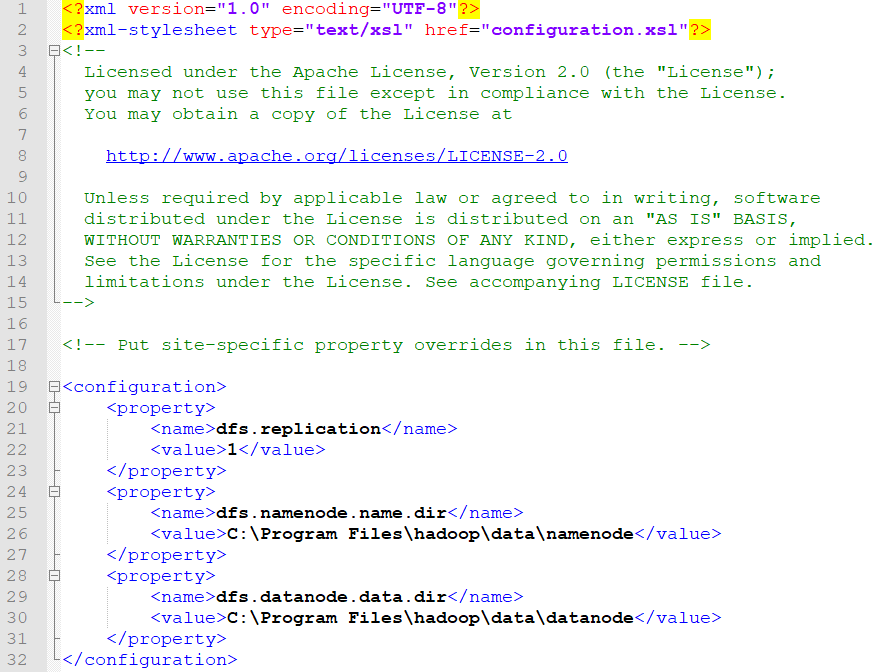
##### 3.3.2.c.ii core-site.xml

Edited core-site.xml is given in Listing 9.

Listing 9: Edited core-site.xml

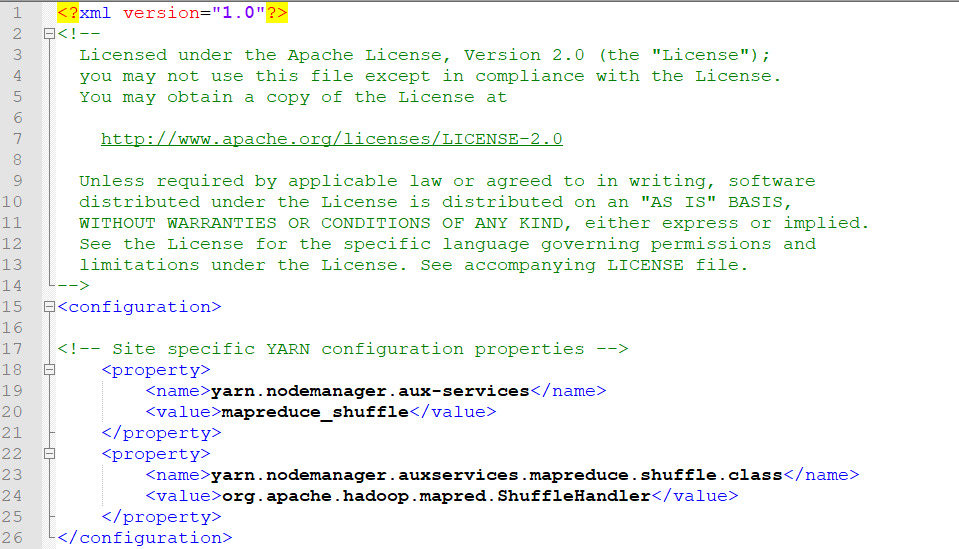
##### 3.3.2.c.iii hdfs-site.xml

Edited hdfs-site.xml file is given in Listing 10

Listing 10: Edited hdfs-site.xml

##### 3.3.2.c.iv yarn-site.xml

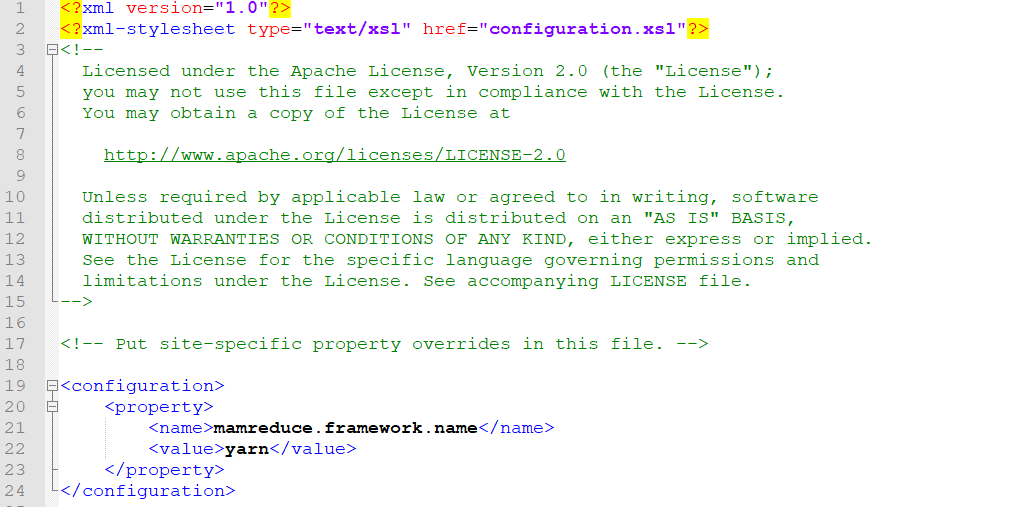
Edited yarn-site.xml is give in Listing 11



Listing 11: Edited yarn-site.xml

##### 3.3.2.c.v mapred-site.xml

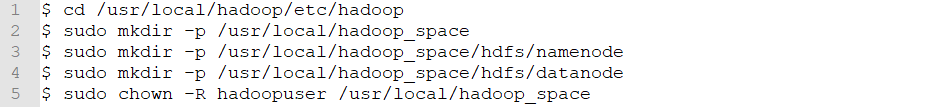
Edited mapred-site.xml file is given in Listing 12



Listing 12: Edited mapred-site.xml

#### 3.3.2.d Formatting hdfs FileSystem

We format the hdfs FileSystem manually by running the commands as shown in Listing 13.



Listing 13: formatting hdfs

We check the name format by using the command in Listing 14.

Listing 14: Checking the namenode format

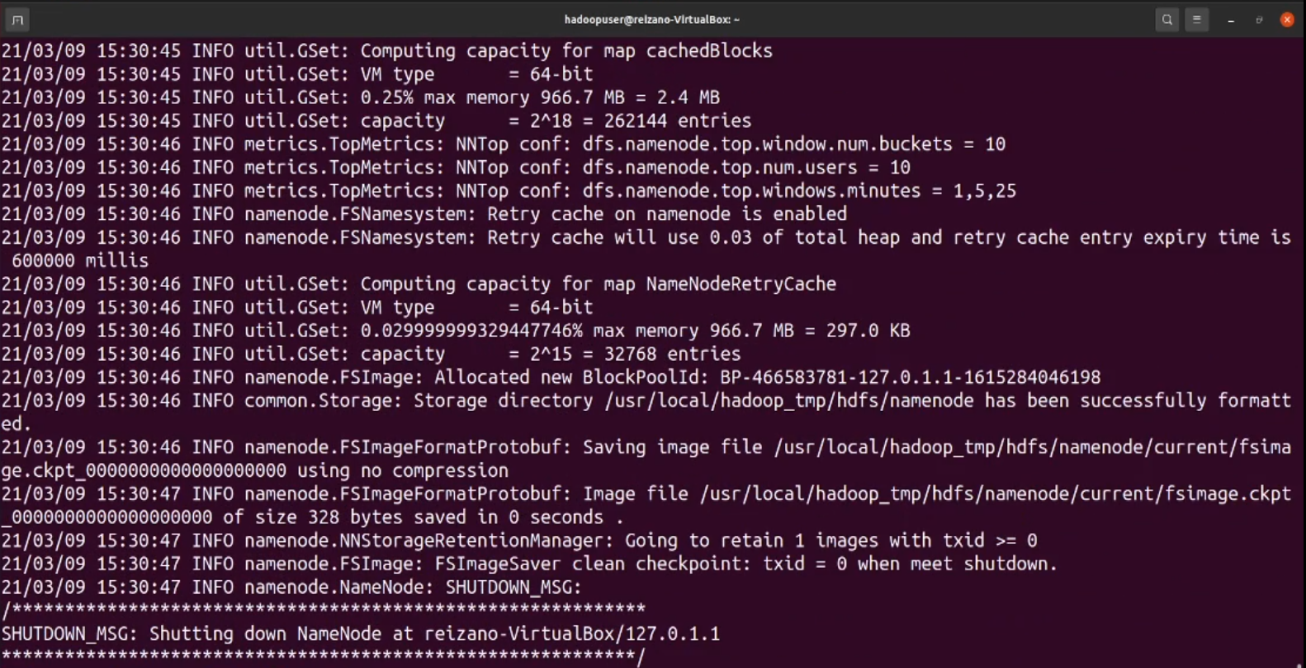


Figure 6: Output when hdfs is formatted

#### 3.3.2.e Starting and stopping hdfs

After completing all prerequisites, installation steps mentioned and formatting hdfs, hadoop is ready for use. Hadoop can be started and stopped using the start and stop script available in bin directory (done using hadouser). Script to start and stop hadoop when run on my system are given in figure 7 and figure 8 respectively. The command to start hadoop services is (it is assumed you are in /usr/local/hadoop directory). Figure 7 also mentions jps, jps is a tool available in Java used to check the services started. When start script is executed the services started are DataNode, SecondaryNameNode, NameNode and JPS

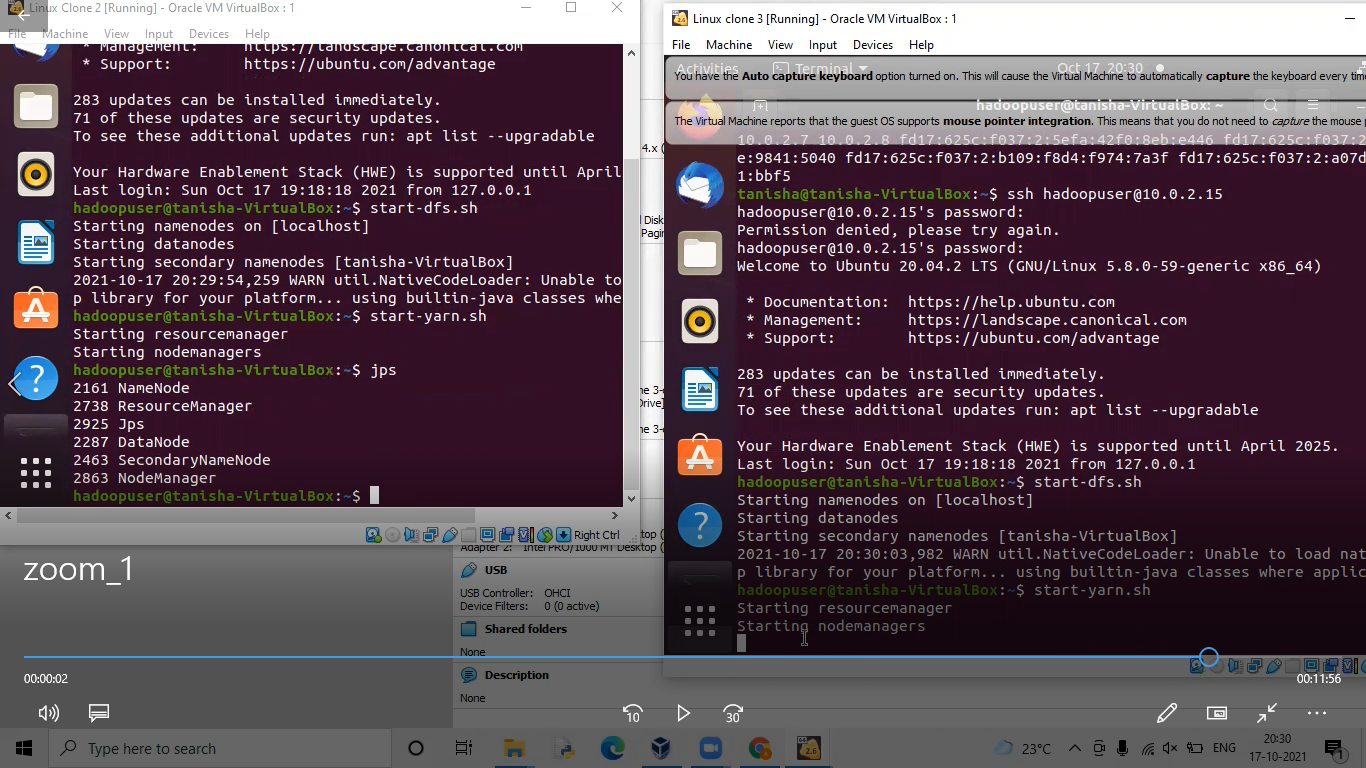


Figure 7: Starting hadoop and checking the status of started processes using jps

The processes can be stopped by using the command shown in the Listing 15.



Listing 15: Stopping hdfs

## 4.2 The Apache Hadoop

It is a software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data. Hadoop YARN: A framework for job scheduling and cluster resource management. Hadoop MapReduce: A YARN-based system for parallel processing of large data sets.MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. Here, MapReduce is performed to count the frequency of each word in a document.

It is a collection of open-source software utilities that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. Hadoop was originally designed for computer clusters built from commodity hardware, which is still in common use. It has since also found use on clusters of higher-end hardware.All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.The core of Apache Hadoop consists of a storage part Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

## 4.3 VirtualBox

Oracle VM VirtualBox (formerly Sun VirtualBox, Sun xVM VirtualBox and Innotek VirtualBox) is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [hosted](https://en.wikipedia.org/wiki/Hypervisor#Classification) [hypervisor](https://en.wikipedia.org/wiki/Hypervisor) for [x86 virtualization](https://en.wikipedia.org/wiki/X86_virtualization), developed by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation). Created by Innotek, it was acquired by [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) in 2008, which was in turn acquired by Oracle in 2010.

VirtualBox may be installed on Windows, macOS, Linux, Solaris and OpenSolaris. There are also ports to FreeBSD and Genode.It supports the creation and management of guest virtual machines running Windows, Linux, BSD, OS/2, Solaris, Haiku, and OSx86, as well as limited virtualization of macOS guests on Apple hardware.For some guest operating systems, a "Guest Additions" package of device drivers and system applications is available,which typically improves performance, especially that of graphics.

# 5. IMPLEMENTATION

## 5.2 Using Hadoop via Remote login between 2 VMs using SSH

### MapReduce method

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System (see [HDFS Architecture Guide](https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html)) are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

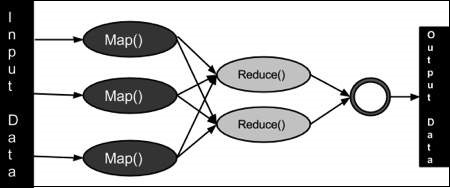
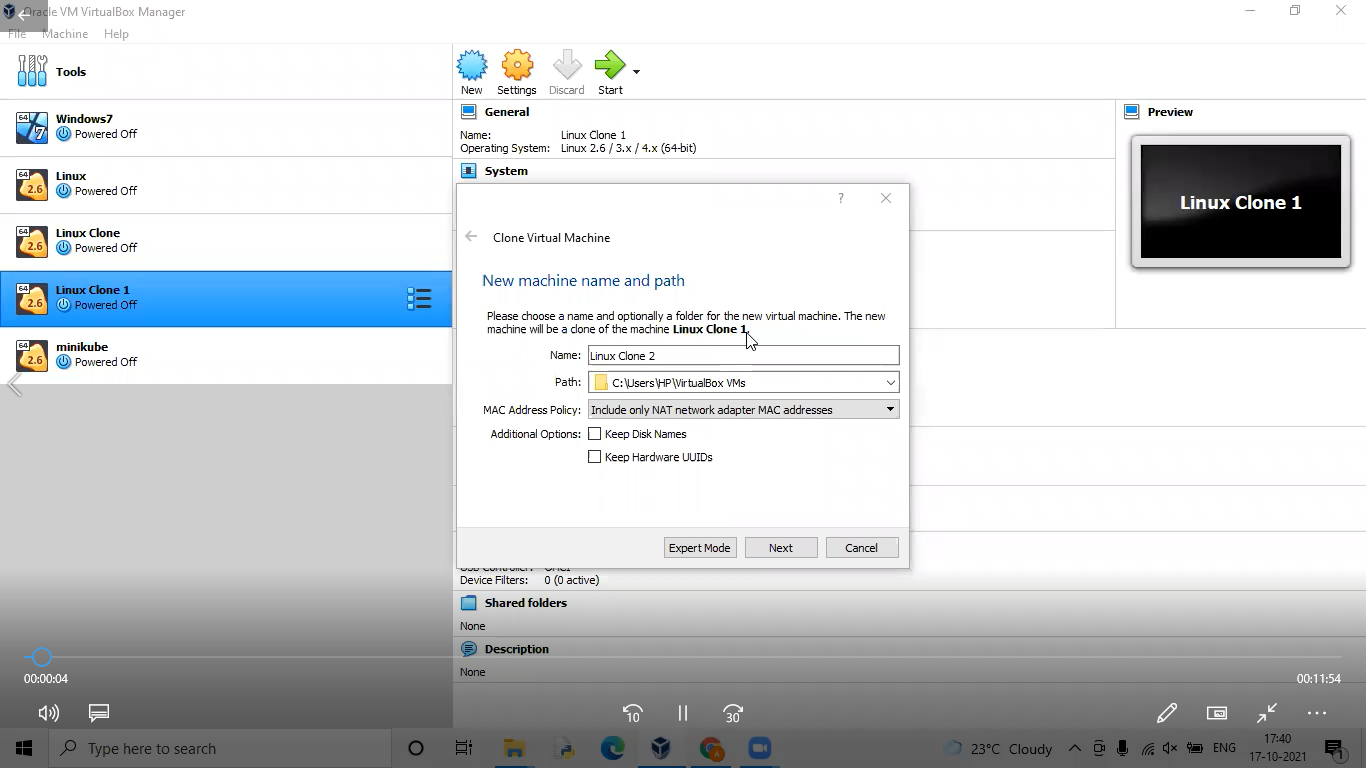


Figure 14 : MapReducer

### 

### 5.2.1 Creating a NAT Network

### 

****

### 

### 5.2.2 Adding created NAT Network to all the VMs and start them

### 

### 

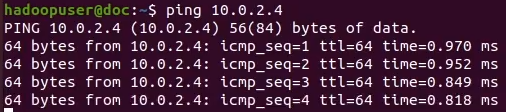
### 5.2.3 Finding and Pinging each others ip

****

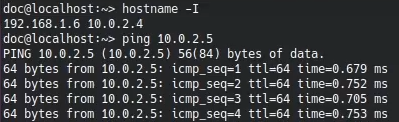
Listing 8 : checking ip connections to host vm (ubuntu)

****

Listing 9 : checking ip connections to client vm (opensuse)



Listing 10 : Pinging client via hosts at 10.0.2.4

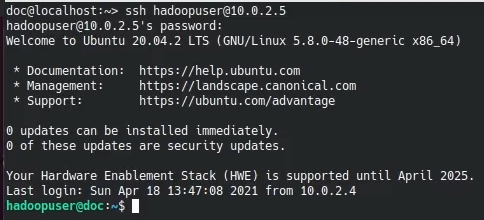


Listing 11 : Pinging hosts via client at 10.0.2.5

### 5.2.4 install and login using ssh



Listing 12 : Installing ssh server on both VMs.

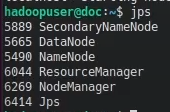
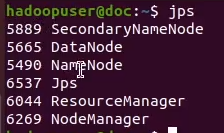
****

Listing 13 : Login to the host VM from client using ssh at host’s username and ip.

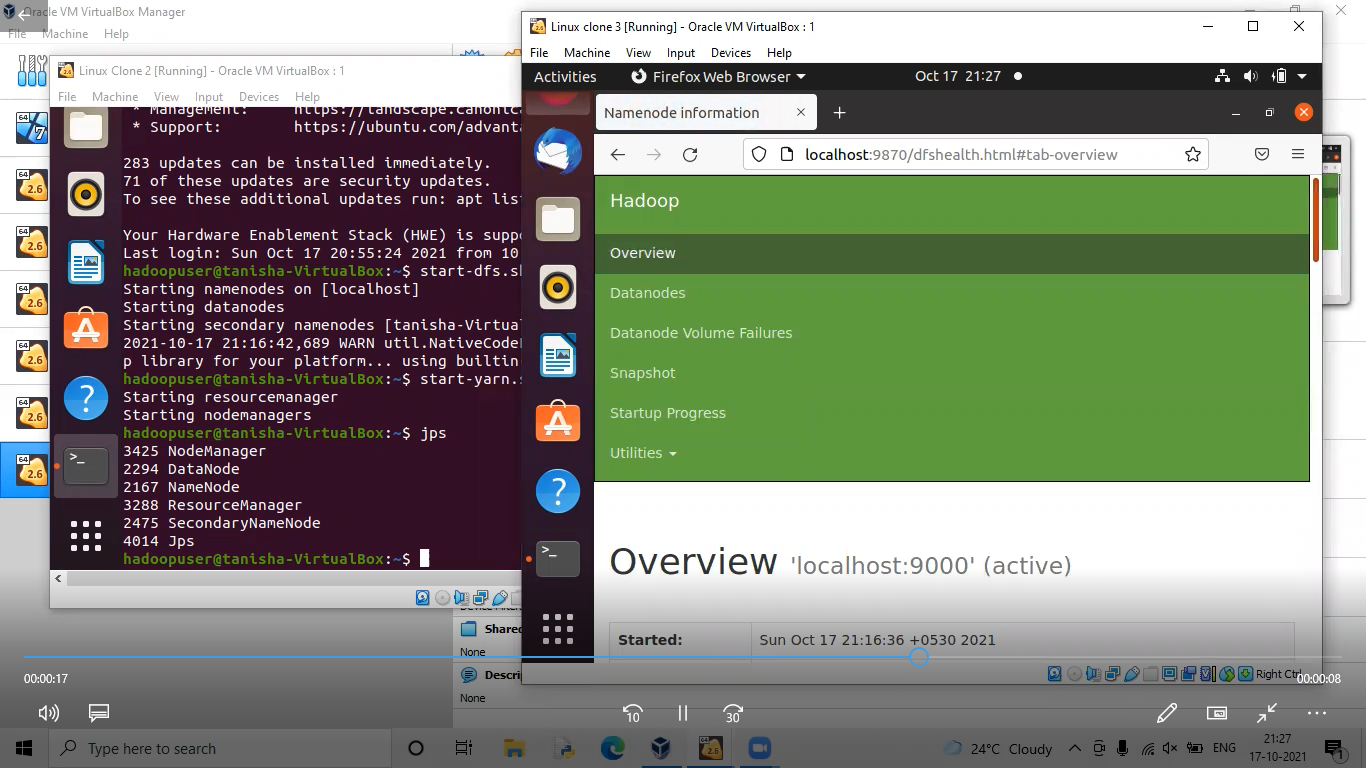
### 5.2.5 Starting hadoop services

****

Listing 14 : Starting hadoop services.

****

Listing 15 : Checking if hadoop systems are working properly from both VMs.



Listing 16 : Accessing hadoop UI at localhost:50070

### 

### 

### 

### 

### 

### 

### 

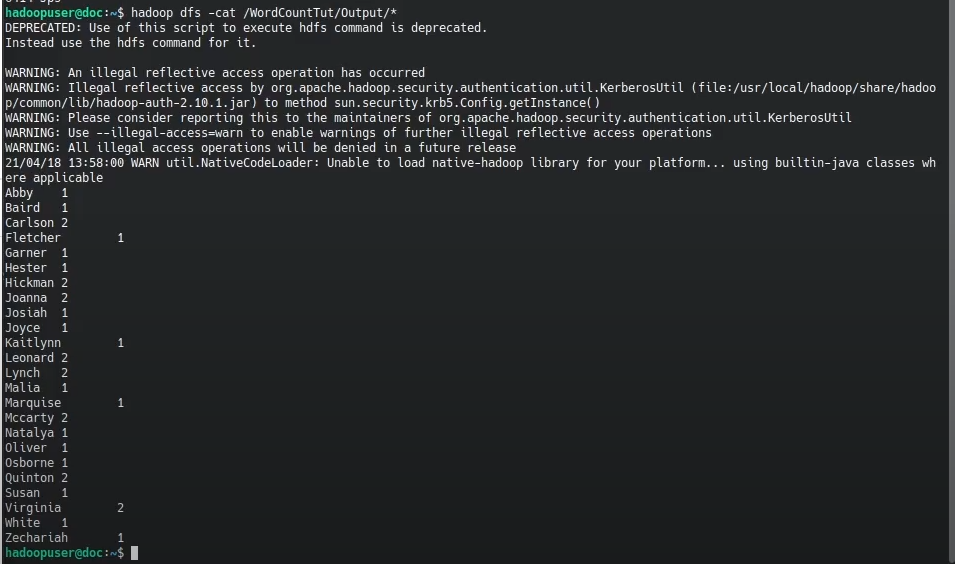
### 

### 

### 

### 

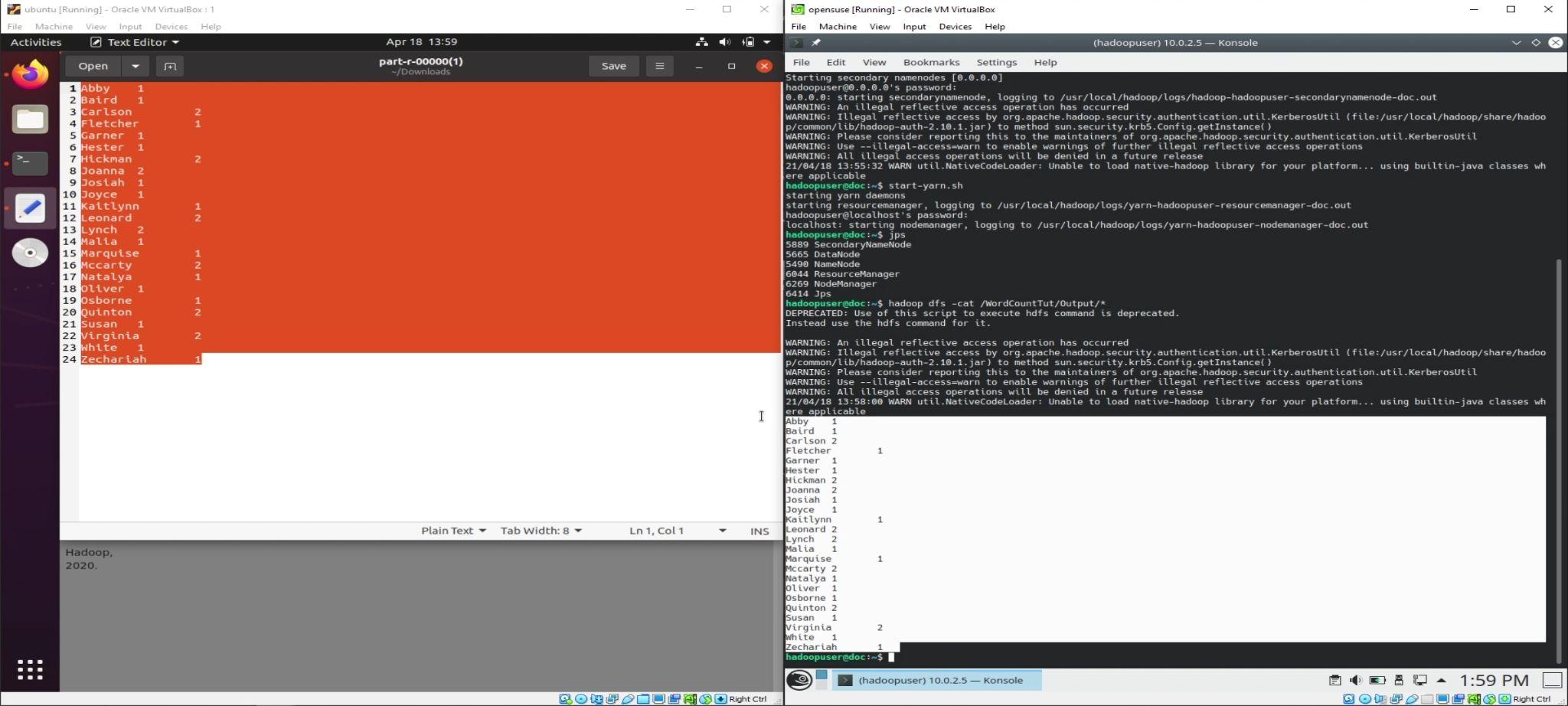
### 5.2.6 Executing word count program



Listing 17 : Executing word count example

//

Listing 18 : Downloading output file

****

### Listing 19 : The output in terminal matches with the output file

### Code:

### *https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:\_WordCount\_v1.0*

#### WordCount.java

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

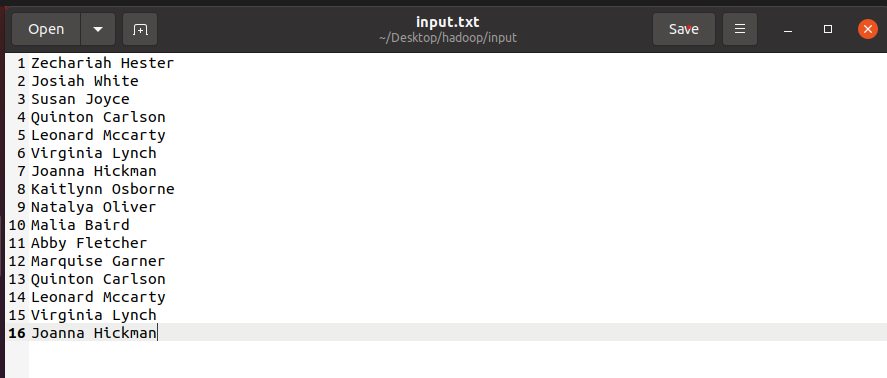
FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

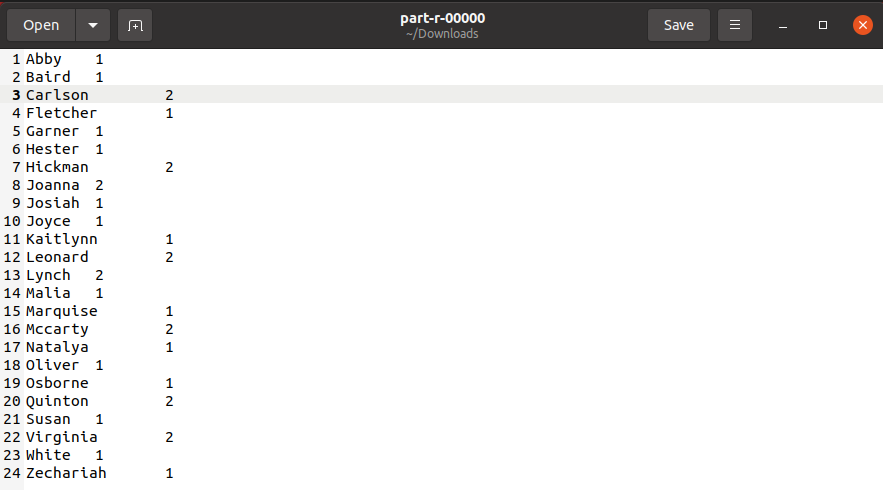
System.exit(job.waitForCompletion(true) ? 0 : 1);

}}

#### 



#### figure 15 : input.txt

 figure 16 : output.txt

**Note: The localhost:50070 has been changed to localhost:9870 in new version of hadoop**