**BASIC UNDERSTANDING ON CLOUD COMPUTING**

**Introduction to Cloud Computing**

**Cloud computing** is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. The essential characteristics are on-demand self-service, broadcast network access, resource pooling, rapid elasticity, measured service, etc.

**Service Models**

**Software as a Service (SaaS):** The capability provided to the consumer is to use the provider‟s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.

**Platform as a Service (PaaS):** The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.3 The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

**Infrastructure as a Service (IaaS):** The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

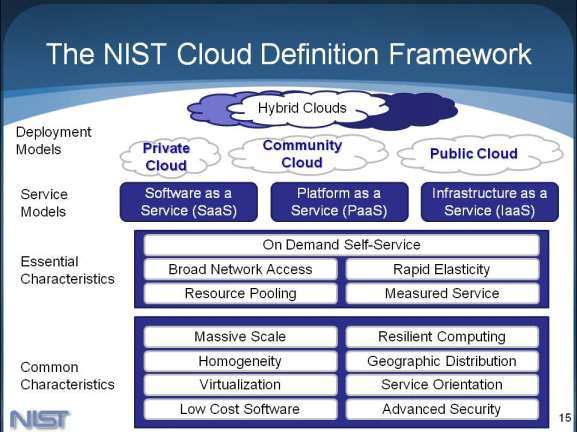
**Deployment Models**

**Private cloud:**The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

**Community cloud:** The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

**Public cloud:** The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

**Hybrid cloud:** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).



**NIST Cloud Definition Framework**

**EX. NO: 1**

**DATE: CREATION AND MANAGEMENT OF VIRTUAL MACHINE IN VIRTUALIZED ENVIRONMENT – VMWAREWORKSTATION**

**AIM**

1. To install VMWare Workstation and configure a virtual machine.
2. To create Virtual Clone.

**THEORY VIRTUALIZATION**

Virtualization is creation of Virtual Machines which can emulate hardware in software or in other words it is the creation of virtual version of something such as a hardware platform, operating system, storage device, or network resources (from Wikipedia). Virtualization is achieved or created with the help of software and this particular software allows you to install any number of OS on your system without using the available hardware directly. When you are running an OS over the top of another on your machine the whole environment acts like a HOST and GUEST OS. The real operating system acts as a HOST and the OS run by virtualization software acts as a GUEST OS.The entire load balancing is actually done by the HOST operating system.

**Types of Virtualization**: There are mainly three types of virtualization.

* + Full virtualization
  + OS level virtualization
  + Para virtualization

**Full virtualization**

As the name suggests everything in a system is virtualized which includes the processor, storage, networking components etc. Virtual Box, VMware are example of “Full Virtualization” solutions.

**OS Level virtualization**

In this type of virtualization only applications are run inside the software. In this case the application is given a platform to work. Isolation is created and the application is made to believe that it is the only thing running on the system.

**Para virtualization**

It‟s a semi-virtualized environment created for the guest OS. A modified guest OS is created using a hypervisor. “The intent of the modified interface is to reduce the portion of the guest‟s execution time spent performing operations which are substantially more difficult to run in a virtual environment compared to a non-virtualized environment. The Para virtualization provides specially defined „hooks‟ to allow the guest(s) and host to request and acknowledge these tasks, which would otherwise be executed in the virtual domain (where execution performance is worse). A successful Para virtualized platform may allow the virtual machine monitor (VMM) to be simpler (by relocating execution of critical tasks from the virtual domain to

the host domain), and/or reduce the overall performance degradation of machine-execution inside the virtual-guest.”

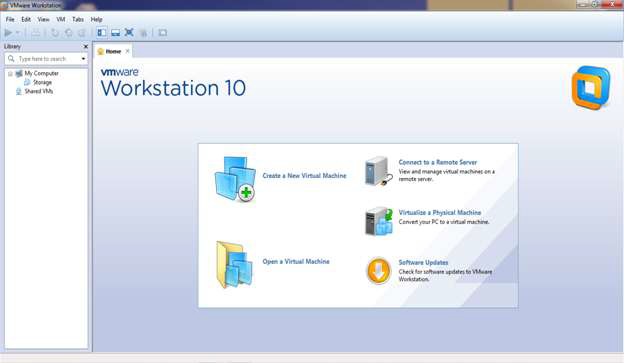
**Advantages of Virtualization**

* One of the biggest advantages of virtualization is scalability i.e. the ability to expand. Whenever there is excessive load on some part of application in a server you can easily create a similar virtual environment on a different server and configure the setup.
* Hardware maintenance cost is reduced because you don‟t need many servers to install different applications.
* You can save a huge amount of energy by running one physical server instead of many and less power backup is required.
* You can get faster and safer backups by taking live snapshot while server is running.
* You will get centralized monitoring of your resources as virtualization provides easy way of connecting and maintaining your virtual servers

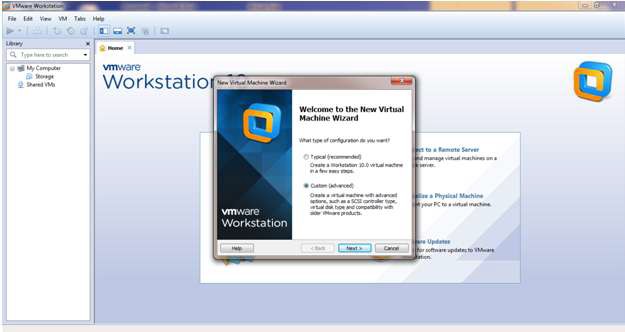
**PRODEDURE**

1. **To install VMWare Workstation and configure a virtual machine.**

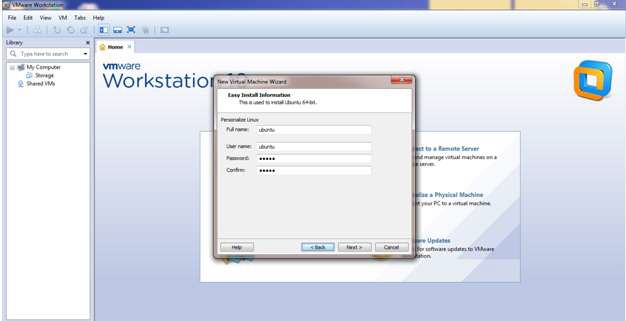
**Step: 1** – Create new virtual machine



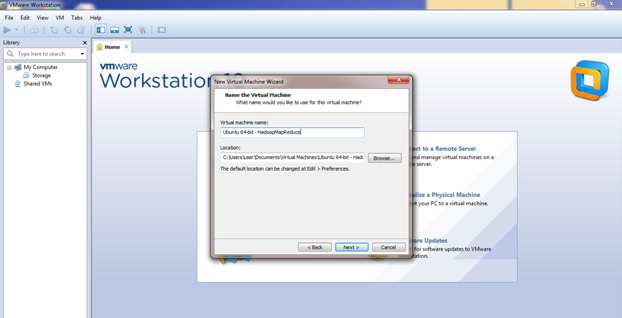
**Step: 2** – Customize the set-up



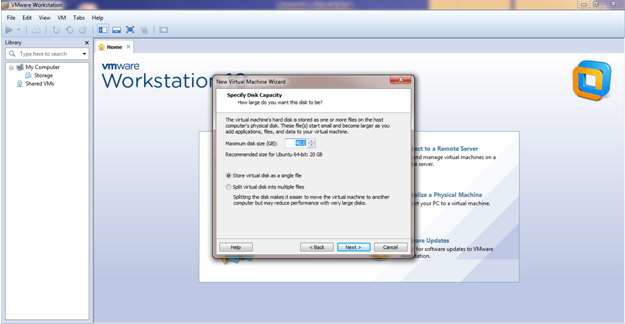
**Step: 3** – Set Username and Password



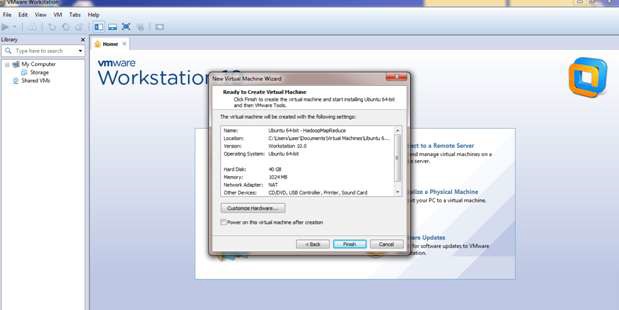
**Step: 4** – Browse for .iso file of an operating system (Ubuntu)



**Step: 5**– Specify the hardware capacity (Hard-disk size)

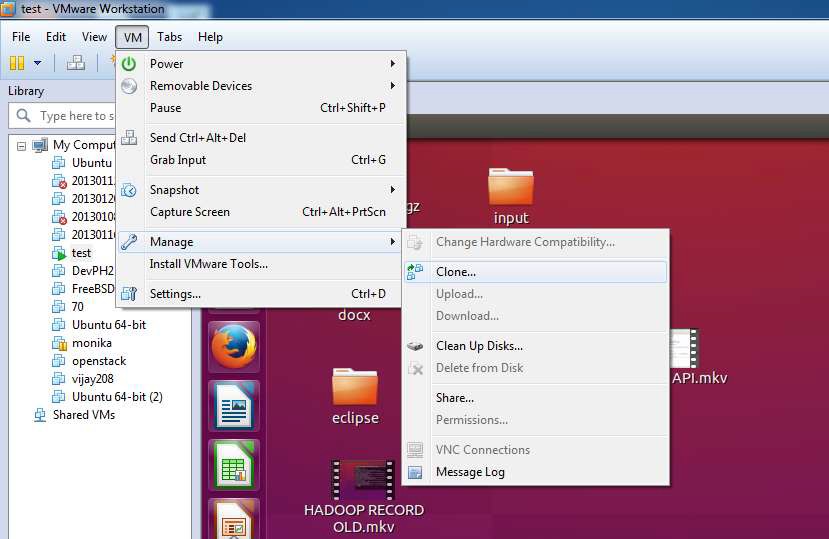


**Step: 6** - Finish



1. **To create Virtual Clone.**

**Step: 1 –**Click VM -> Manage -> Clone



**RESULT**

Thus the procedure to create virtual machine with instance and virtual clone is done successfully.

**EX. NO: 2**

**DATE: ATTACHING A VIRTUAL BLOCK**

**AIM**

To find procedure to attach a virtual block to virtual machine and check whether it holds

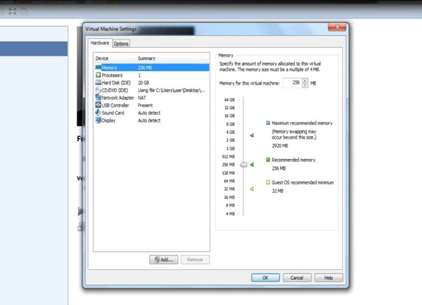
data even after the release of the virtual machine.

a) Adding an additional block to Hard-Disk in Virtual Machine.

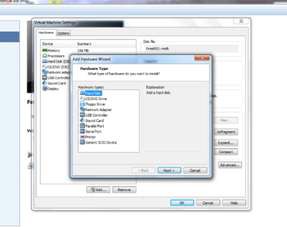
**PROCEDURE**

**a) Adding an additional block to Hard-Disk in Virtual Machine.**

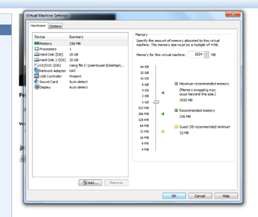
**Step: 1** - Select hard Disk from the virtual machine settings dialog box. Select add button. In the Add Hardware wizard select the hardware type as hard disk and click next. Select the virtual disk type as IDE and click next.



**Step: 2** – In the next page select the Create a new virtual disk option and click next. Provide the disk capacity as 20GB and select store virtual disk as single file option and click next. Specify the disk file and click finish.



**Step: 3** – Additional Hard-Disk



**RESULT**

Thus the procedure to attach a volume to a virtual instance and adding an additional block to Hard-Disk in Virtual Machine is completed successfully.

**EX. NO: 3**

**DATE: INSTALL A C COMPILER IN THE VIRTUAL MACHINE AND**

**EXECUTE A SAMPLE PROGRAM**

**AIM**

To install and run a C Compiler in Ubuntu.

**PROCEDURE**

**Step 1 – Open Console Step 2 – Install gcc**

1. gcc –v
2. sudo apt-get install gcc



1. sudo apt-get install build-essential



C compiler will be installed

**Step 3 - To Install Vim Editor**

1. sudo apt-get update
2. sudo apt-get install vim



**Step 4–Type the C program in terminal**

### hello.c

#include<stdio.h> int main()

{

printf ("Hello"); return 0;

}



**Step 5 – Compile and Execute C Program**

cc hello.c

./a.out

**OUTPUT**

Hello



# RESULT

Thus the procedure to install and run a C Compiler in Ubuntu is done successfully.

**EX. NO: 4**

**DATE: SINGLE-NODE HADOOP INSTALLATION AIM**

To set-up one node Hadoop cluster

* + System Update
  + Install Java
  + Add a dedicated Hadoop user
  + Install SSH and setup SSH certificates
  + Check if SSH works
  + Install Hadoop
  + Modify Hadoop config files
  + Format Hadoop filesystem
  + Start Hadoop
  + Check Hadoop through web UI
  + Stop Hadoop

**THEORY**

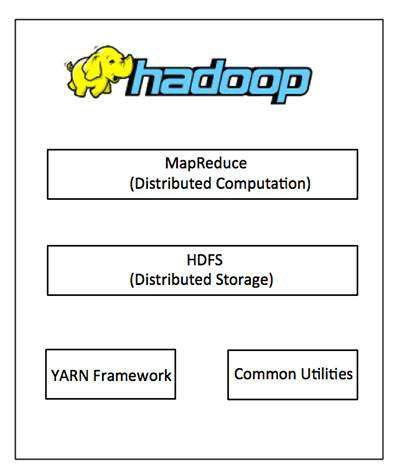
Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

**HADOOP ARCHITECTURE**

Hadoop framework includes following four modules:

* **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
* **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
* **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high-throughput access to application data.
* **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

We can use following diagram to depict these four components available in Hadoop framework.



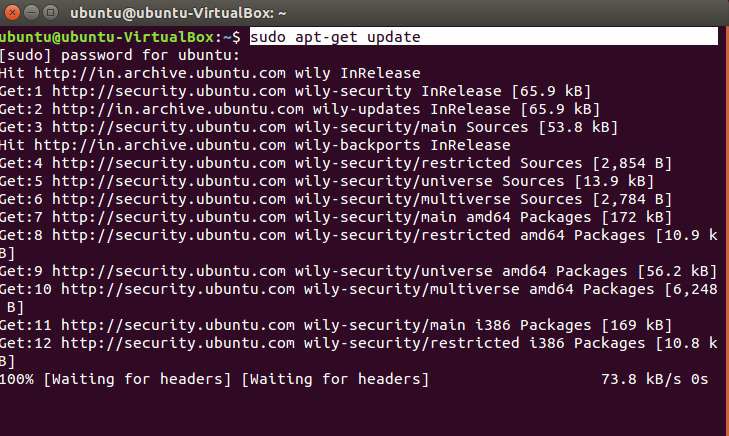
**Hadoop Architecture**

Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

**PROCEDURE**

**Step 1 – System Update**

**$ sudo apt-get update**



**Step 2 – Install Java and Set JAVA\_HOME**

**//This first thing to do is to setup the webupd8 ppa on your system. Run the following command and proceed.**

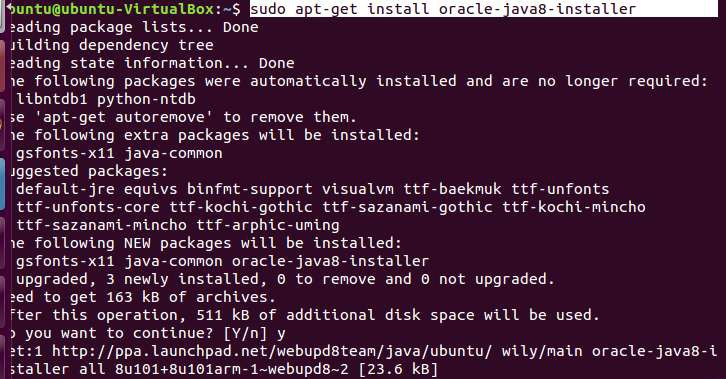
**$ sudo apt-add-repository ppa:webupd8team/java**

**$ sudo apt-get update**

***//*After setting up the ppa repository, update the package cache as well.**

**//Install the Java 8 installer**

**$ sudo apt-get install oracle-java8-installer**

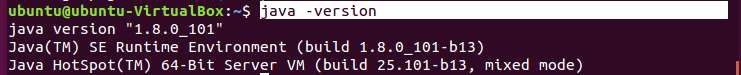


**// After the installation is finished, Oracle Java is setup. Run the java command again to check the version and vendor.**

**[or]**

**$ sudo apt-get install default-jdk**

**$ java -version**

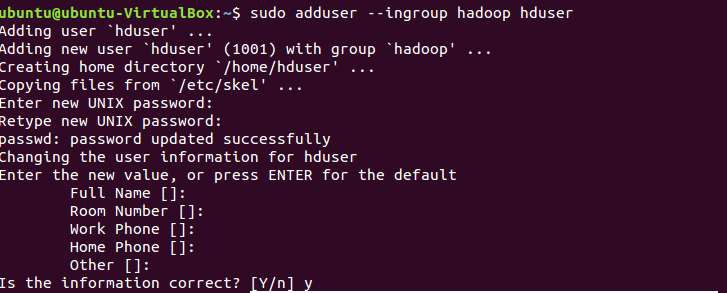


**Step 3 – Add a dedicated Hadoop user**

**$ sudo addgroup hadoop**

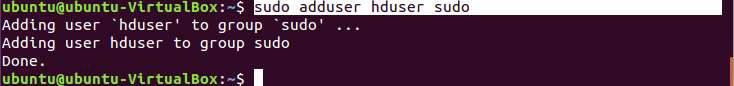


$ **sudo adduser --ingroup hadoop hduser**



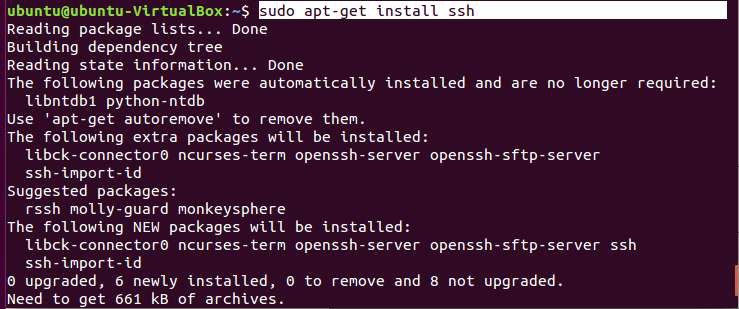
**// Add hduser to sudo user group**

**$ sudo adduser hduser sudo**

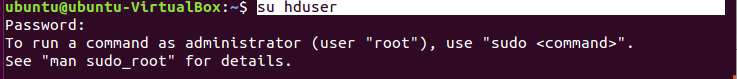


**Step 4 – Install SSH and Create Certificates**

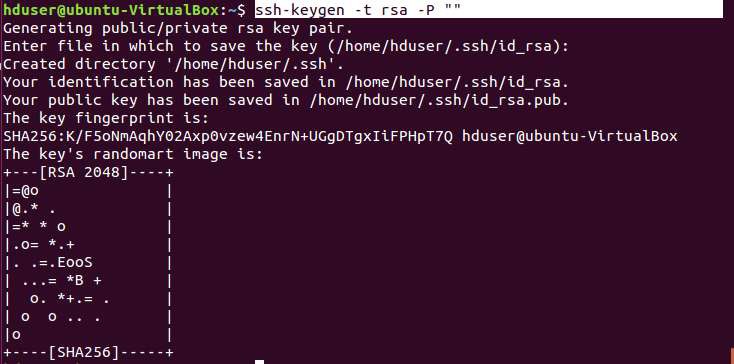
**$ sudo apt-get install ssh**



**$ su hduser**



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



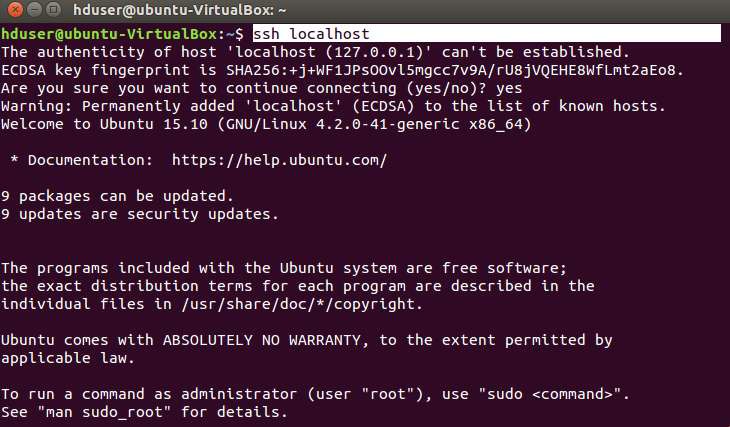
**// Set Environmental variables**

**$ cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys**



**Step 5 – Check if SSH works**

**$ ssh localhost**



**Step 6 – Install Hadoop**

**$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.8.4/hadoop-2.8.4.tar.gz**

**// Extract Hadoop-2.7.2**

**$ sudo tar xvzf hadoop-2.7.2.tar.gz**



**// Create a folder ‘hadoop’ in /usr/local**

**$ sudo mkdir –p /usr/local/hadoop**



**// Move the Hadoop folder to /usr/local/hadoop**

**$ sudo mv hadoop-2.7.2 /usr/local/hadoop**



**// Assigning read and write access to Hadoop folder**

**$ sudo chown –R hduser:hadoop /usr/local/hadoop**



**Step 7 - Modify Hadoop config files**

**//Hadoop Environmental variable setting – The following files will be modified**

1. **~/.bashrc**
2. **/usr/local/hadoop/hadoop-2.7.2/etc/hadoop/hadoop-env.sh**
3. **/usr/local/hadoop/hadoop-2.7.2/etc/hadoop/core-site.xml**
4. **/usr/local/hadoop/hadoop-2.7.2/etc/hadoop/hdfs-site.xml**
5. **/usr/local/hadoop/hadoop-2.7.2/etc/hadoop/yarn-site.xml**
6. **/usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml.template**

**$ sudo nano ~/.bashrc**

**// Add the following lines at the end of the file**

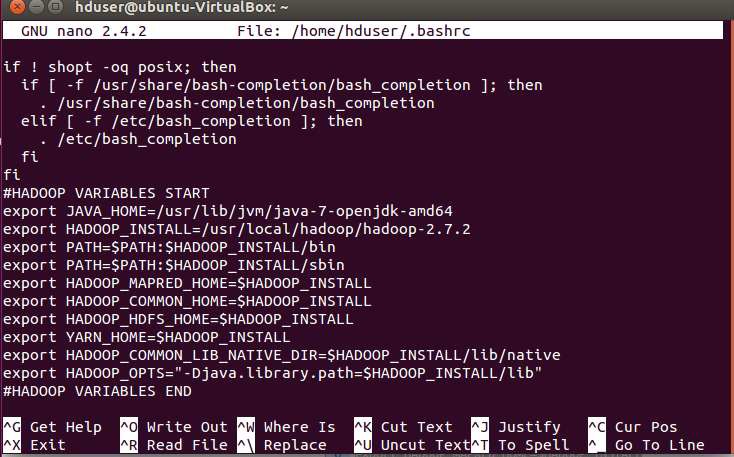
export JAVA\_HOME=/usr/lib/jvm/java-8-oracle

export HADOOP\_HOME=/usr/local/hadoop/hadoop-2.7.2 export PATH=$PATH:$HADOOP\_HOME/bin

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

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME export HADOOP\_COMMON\_HOME=$HADOOP\_HOME export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

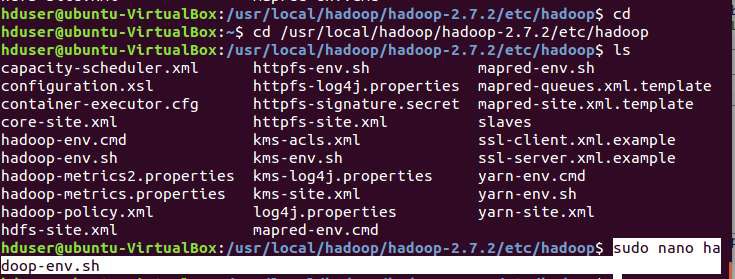
export YARN\_HOME=$HADOOP\_HOME HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native export HADOOP\_OPTS="-D.java.library.path=$HADOOP\_HOME/lib" export PATH=$PATH:/usr/local/hadoop/hadoop-2.7.2/bin



**// Configure Hadoop Files**

**$ cd /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/**

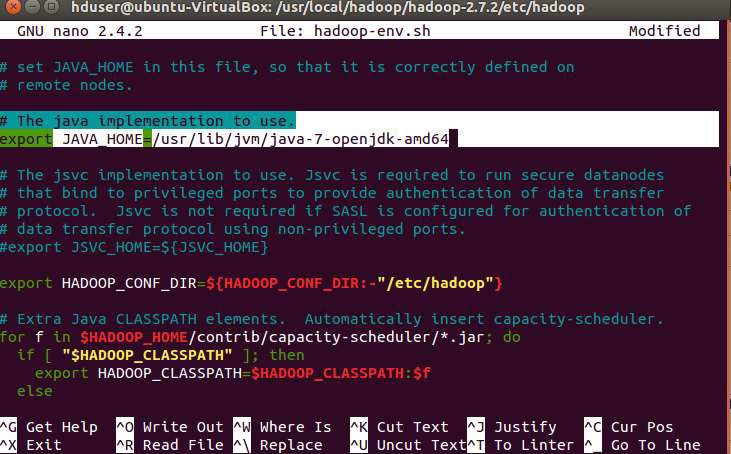
**$ sudo nano hadoop-env.sh**



**// Add following line in hadoop-env.sh – Set JAVA variable in Hadoop**

# The java implementation to use.

export JAVA\_HOME=/usr/lib/jvm/java-8-oracle



**// Create datanode and namenode**

**$ sudo mkdir –p /usr/local/hadoop\_tmp/hdfs/namenode**

**$ sudo mkdir –p /usr/local/hadoop\_tmp/hdfs/datanode**

**// Changing ownership to hadoop\_tmp**

**$ sudo chown –R hduser:hadoop /usr/local/hadoop\_tmp**



**// Edit hdfs-site.xml**

**$ sudo nano hdfs-site.xml**

**// Add the following lines between <configuration> …… </configuration>**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop\_tmp/hdfs/namenode</value>

</property>

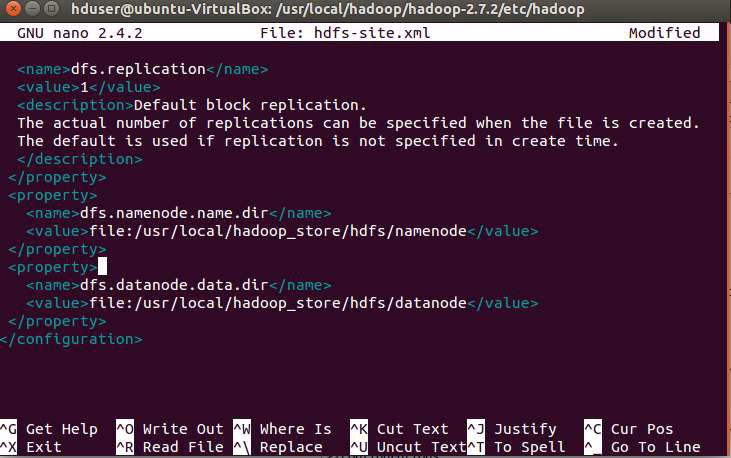
<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop\_tmp/hdfs/datanode</value>

</property>

</configuration>



# // Edit core-site.xml

**$ sudo nano core-site.xml**

**// Add the following lines between <configuration> …… </configuration>**

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

**// Edit yarn-site.xml**

**$ sudo nano yarn-site.xml**

**// Add the following lines between <configuration> …… </configuration>**

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.Shuffle-Handler</value>

</property>

</configuration>

# // Edit mapred-site.xml

**$ cp /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml.template**

**/usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml**



**$ sudo nano mapred-site.xml**

**// Add the following lines between <configuration> …… </configuration>**

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

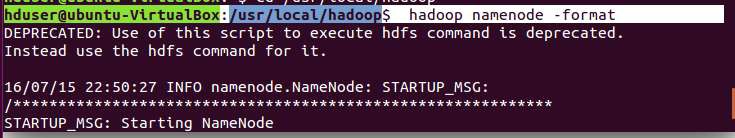
</property>

</configuration>

**Step 8 – Format Hadoop File System**

**$ cd /usr/local/hadoop/hadoop-2.7.2/bin**

**$ hadoop namenode -format**

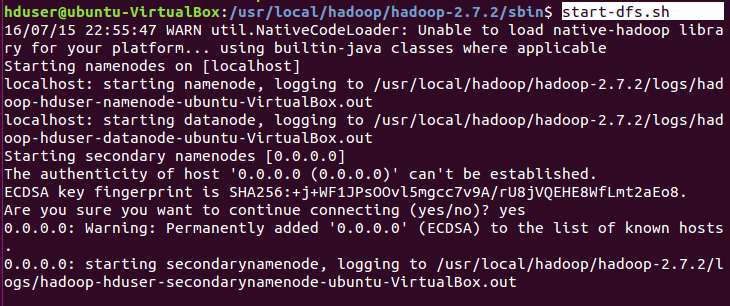


**Step 9 - Start Hadoop**

**$ cd /usr/local/hadoop/hadoop-2.7.2/sbin**

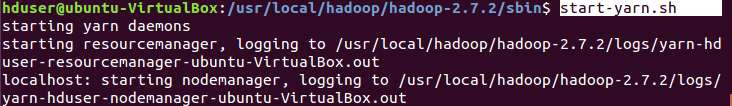
**// Starting dfs services**

**$ start-dfs.sh**



**// Starting mapreduce services**

**$ start-yarn.sh**

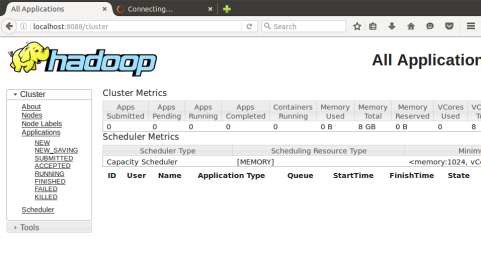


**$ jps**

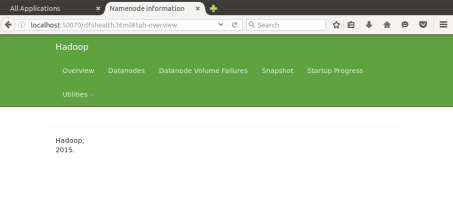


**Step 10 - Check Hadoop through web UI**

Go to browser type **http://localhost:8088 – All Applications Hadoop Cluster**



Go to browser type **http://localhost:50070 – Hadoop Namenode**



# Step 11 - Stop Hadoop

**$ stop-dfs.sh**

**$ stop-yarn.sh**

**RESULT**

Thus the procedure to install single-node Hadoop is executed successfully.

**EX. NO: 5**

**DATE: MOUNT THE ONE NODE HADOOP CLUSTER USING FUSE**

**AIM**

To mount one-node Hadoop cluster using FUSE

**THEORY**

FUSE (Filesystem in Userspace) enables you to write a normal user application as a bridge for a traditional filesystem interface. The hadoop-hdfs-fuse package enables you to use your HDFS cluster as if it were a traditional filesystem on Linux. It is assumed that you have a working HDFS cluster and know the hostname and port that your NameNode exposes.

**PROCEDURE**

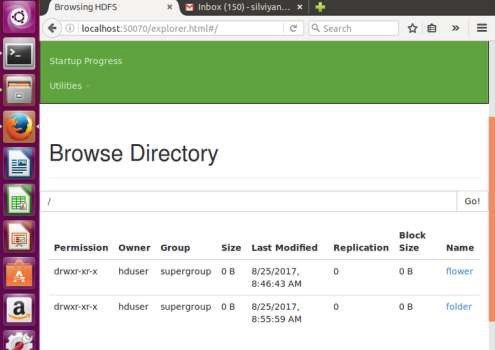
**$ su hduser**

**$ cd /usr/local/hadoop/hadoop-2.7.2/sbin**

**$ start-dfs.sh**

**$ start-yarn.sh**

**$ jps**



**$ wget** [**http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-**](http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-) **repository\_1.0\_all.deb**

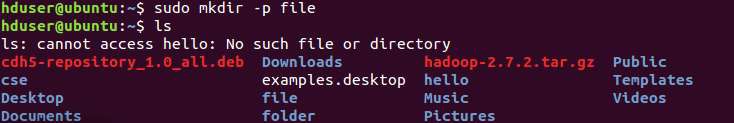
**$ sudo dpkg -i cdh5-repository\_1.0\_all.deb**

**$ sudo apt-get update**

**$ sudo apt-get install hadoop-hdfs-fuse**

**$ sudo mkdir -p <mount\_point>**

**Sample :**sudo mkdir -p file



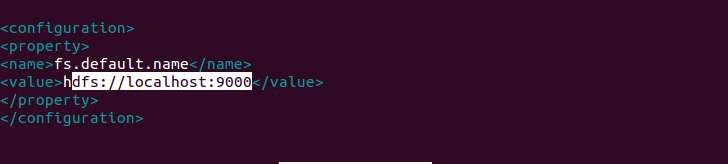
(set folder permission and enable share local network)



(for name\_node\_hostname>:<namenode\_port>

goto (cd /usr/local/hadoop/etc/hadoop/ vi core-site.xml )

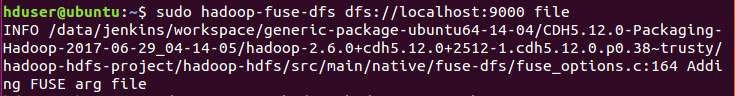


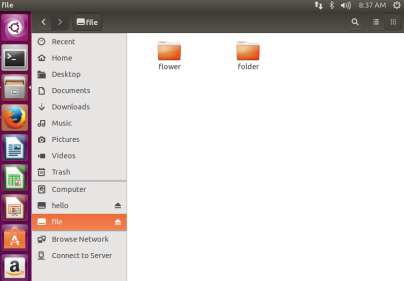


**$ sudo hadoop-fuse-dfs dfs://<name\_node\_hostname>:<namenode\_port>**

**<mount\_point>**

**Sample :** sudo hadoop-fuse-dfs dfs://localhost:9000 file





Once HDFS has been mounted at <mount\_point>, you can use most of the traditional filesystem operations (e.g., cp, rm, cat, mv, mkdir, rmdir, more, scp). However, random write operations such as rsync, and permission related operations such as chmod, chown are not supported in FUSE-mounted HDFS.

**RESULT**

Thus the steps to mount one-node Hadoop cluster using FUSE is done successfully.

**EX. NO: 6**

**DATE: INTERACTION WITH HADOOP API FOR ACCESSING HDFS FROM**

**LOCAL FILE SYSTEM**

**AIM**

To write a program to use the API's of Hadoop for copying File from Local File System

to HDFS and to interact with it.

**PROCEDURE**

**Step 1 – Open New Terminal**

$ cd Desktop/

$ mkdir inputdata

$ cd inputdata/

$ echo “Hai, Hello, How are you? How is your health?” >> file.txt

$ cat >> file.txt

**Step 2 – Download and open eclipse by creating workspace**

Create a new java project.

**Step 3 – Add jar to the project**

You need to remove dependencies by adding jar files in the hadoop source folder. Now Click on **Project** tab and go to Properties.Under Libraries tab, click Add External JARs and select all the jars in the folder (click on 1st jar, and Press Shift and Click on last jat to select all jars in between and click ok)

**/usr/local/hadoop/hadoop-2.7.2/share/hadoop/common** and

**/usr/local/hadoop/hadoop-2.7.2/share/hadoop/mapreduce** folders.

**Step 4 – Program**

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.util.Tool;

import java.io.BufferedInputStream; import java.io.FileInputStream; import java.io.InputStream;

import java.io.OutputStream;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IOUtils; import org.apache.hadoop.util.ToolRunner;

public class HdfsWriter extends Configured implements Tool { public static final String FS\_PARAM\_NAME = "fs.defaultFS";

public int run(String[] args) throws Exception {

if (args.length < 2) {

System.err.println("HdfsWriter /home/rec/Desktop/input/file.txt /hello"); return 1;

}

String localInputPath = args[0];

Path outputPath = new Path(args[1]); Configuration conf = getConf();

System.out.println("configured filesystem = " + conf.get(FS\_PARAM\_NAME)); FileSystem fs = FileSystem.get(conf);

if (fs.exists(outputPath)) { System.err.println("output path exists"); return 1;

}

OutputStream os = fs.create(outputPath);

InputStream is = new BufferedInputStream(new FileInputStream(localInputPath)); IOUtils.copyBytes(is, os, conf);

return 0;

}

public static void main( String[] args ) throws Exception { int returnCode = ToolRunner.run(new HdfsWriter(), args); System.exit(returnCode);

}

}

**Step 5 - Creatr JAR file**

Now Click on the Run tab and click Run-Configurations. Click on New Configuration button on the left-top side and Apply after filling the following properties.

**Step 6 - Export JAR file**

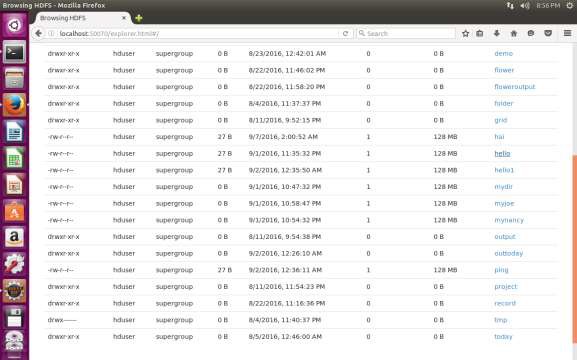
Now click on File tab and select Export. under Java, select Runnable Jar.

In Launch Config – select the config fie you created in **Step 9** (WordCountConfig).

* + Select an export destination (lets say desktop.)
  + Under Library handling, select Extract Required Libraries into generated JAR and click Finish.
  + Right-Click the jar file, go to Properties and under **Permissions**tab, Check Allow executing file as a program. and give Read and Write access to all the users

**Step 7–Execute File**

**hadoop jar /home/rec/Desktop/HelloWrite.jar /home/rec/Desktop/input/file.txt /hello**



**RESULT**

Thus the program to use the API's of Hadoop for copying File from Local File System to HDFS is executed and output verified successfully.

**EX. NO: 7**

**DATE: DEMONSTRATION OF MAP AND REDUCE TASKS**

**AIM**

To write a wordcount program to demonstrate the use of Map and Reduce tasks.

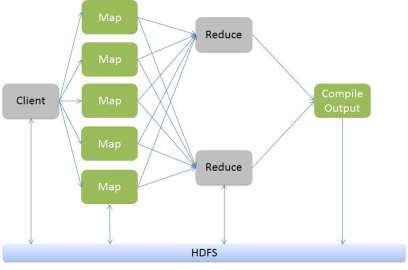
**THEORY**

Hadoop MapReduce is a programming paradigm at the heart of Apache Hadoop for providing massive scalability across hundreds or thousands of Hadoop clusters on commodity hardware. The MapReduce model processes large unstructured data sets with a distributed algorithm on a Hadoop cluster.

The term MapReduce represents two separate and distinct tasks Hadoop programs perform-Map Job and Reduce Job. Map job scales takes data sets as input and processes them to produce key value pairs. Reduce job takes the output of the Map job i.e. the key value pairs and aggregates them to produce desired results. The input and output of the map and reduce jobs are stored in HDFS.

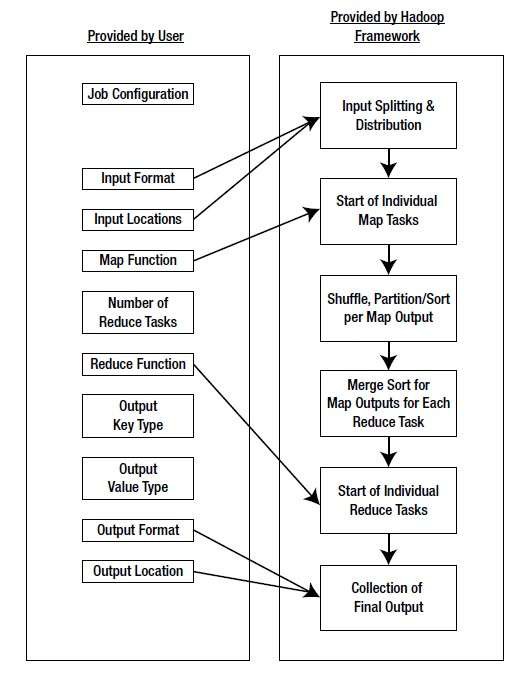
**MAPREDUCE**

Hadoop MapReduce (Hadoop Map/Reduce) is a software framework for distributed processing of large data sets on computing clusters. It is a sub-project of the Apache Hadoop project. Apache Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. MapReduce is the core component for data processing in Hadoop framework. In layman‟s term Mapreduce helps to split the input data set into a number of parts and run a program on all data parts parallel at once. The term MapReduce refers to two separate and distinct tasks. The first is the map operation, takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce operation combines those data tuples based on the key and accordingly modifies the value of the key.



**MAPREDUCE ARCHITECTURE**

The figure shown below illustrates the various parameters and modules that can be configured during a MapReduce operation:



**MapReduce Architecture**

JobConf is the framework used to provide various parameters of a MapReduce job to the Hadoop for execution. The Hadoop platforms executes the programs based on configuration set using JobConf. The parameters being Map Function, Reduce Function, combiner , Partitioning function, Input and Output format. Partitioner controls the shuffling of the tuples when being sent from Mapper node to Reducer nodes. The total number of partitions done in the tuples is equal to the number of reduce nodes. In simple terms based on the function output the tuples are transmitted through different reduce nodes.

Input Format describes the format of the input data for a MapReduce job. Input location specifies the location of the datafile. Map Function/ Mapper convert the data into key value pair. For example let‟s consider daily temperature data of 100 cities for the past 10 years. In this the map function is written such a way that every temperature being mapped to the corresponding

city. Reduce Function reduces the set of tuples which share a key to a single tuple with a change in the value. In this example if we have to find the highest temperature recorded in the city the reducer function is written in such a way that it return the tuple with highest value i.e: highest temperature in the city in that sample data.

The number of Map and Reduce nodes can also be defined. You can set Partitioner function which partitions and transfer the tuples which by default is based on a hash function. In other words we can set the options such that a specific set of key value pairs are transferred to a specific reduce task. For example if your key value consists of the year it was recorded, then we can set the parameters such that all the keys of specific year are transferred to a same reduce task. The Hadoop framework consists of a single master and many slaves. Each master has JobTracker and each slave has TaskTracker. Master distributes the program and data to the slaves. Task tracker, as the name suggests keep track of the task directed to it and relays the information to the JobTracker. The JobTracker monitors all the status reports and re-initiates the failed tasks if any.

Combiner classes are run on map task nodes. It takes the tuples emitted by Map nodes as input. It basically does reduce operation on the tuples emitted by the map node. It is like a pre- reduce task to save a lot of bandwidth. We can also pass global constants to all the nodes using

„Counters‟. They can be used to keep track of all events in map and reduce tasks. For example we can pass a counter to calculate the statistics of an event beyond a certain threshold.

**PROCEDURE**

**Step 1 - Open Terminal**

$ su hduser Password:

**Step 2 - Start dfs and mapreduce services**

$ cd /usr/local/hadoop/hadoop-2.7.2/sbin

$ start-dfs.sh

$ start-yarn.sh

$ jps

**Step 3 - Check Hadoop through web UI**

// Go to browser type http://localhost:8088 – All Applications Hadoop Cluster

// Go to browser type http://localhost:50070 – Hadoop Namenode

**Step 4 – Open New Terminal**

$ cd Desktop/

$ mkdir inputdata

$ cd inputdata/

$ echo “Hai, Hello, How are you? How is your health?” >> hello.txt

$ cat >> hello.txt

**Step 5 – Go back to old Terminal**

$ hadoop fs –mkdir /folder

$ hadoop fs –copyFromLocal /home/hduser/Desktop/inputdata/hello.txt /folder

// Check in hello.txt in Namenode using Web UI

**Step 6 – Download and open eclipse by creating workspace**

Create a new java project.

**Step 7 – Add jar to the project**

You need to remove dependencies by adding jar files in the hadoop source folder. Now Click on **Project** tab and go to Properties.Under Libraries tab, click Add External JARs and select all the jars in the folder (click on 1st jar, and Press Shift and Click on last jat to select all jars in between and click ok)

**/usr/local/hadoop/hadoop-2.7.2/share/hadoop/common**and

**/usr/local/hadoop/hadoop-2.7.2/share/hadoop/mapreduce** folders.

**Step -8 – WordCount Program Create 3 java files named**

* **WordCount.java**
* **WordCountMapper.java**
* **WordCountReducer.java WordCount.java**

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner; import org.apache.hadoop.io.Text;

public class WordCount extends Configured implements Tool {

@Override

public int run(String[] arg0) throws Exception {

// TODO Auto-generated method stub if(arg0.length<2)

{

System.out.println("check the command line arguments");

}

JobConf conf=new JobConf(WordCount.class); FileInputFormat.setInputPaths(conf, new Path(arg0[0]));

FileOutputFormat.setOutputPath(conf, new Path(arg0[1])); conf.setMapperClass(WordMapper.class); conf.setReducerClass(WordReducer.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); JobClient.runJob(conf);

return 0;

}

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

{

int exitcode=ToolRunner.run(new WordCount(), args); System.exit(exitcode);

}

}

**WordCountMapper.java**

import java.io.IOException;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reporter;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.Mapper;

public class WordCountMapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>

{

@Override

public void map(LongWritable arg0, Text arg1, OutputCollector<Text, IntWritable> arg2, Reporter arg3)

throws IOException {

// TODO Auto-generated method stub

String s=arg1.toString(); for(String word:s.split(" "))

{

arg2.collect(new Text(word),new IntWritable(1));

}

}

}

**WordCountReducer.java**

import java.io.IOException; import java.util.Iterator;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter; import org.apache.hadoop.io.Text;

public class WordCountReducer implements Reducer<Text,IntWritable,Text,IntWritable> {

@Override

public void configure(JobConf arg0) {

// TODO Auto-generated method stub

}

@Override

public void close() throws IOException {

// TODO Auto-generated method stub

}

@Override

public void reduce(Text arg0, Iterator<IntWritable> arg1, OutputCollector<Text, IntWritable> arg2, Reporter arg3)

throws IOException {

// TODO Auto-generated method stub int count=0;

while(arg1.hasNext())

{

IntWritable i=arg1.next(); count+=i.get();

}

arg2.collect(arg0,new IntWritable(count));

}

}

**Step 9 - Creatr JAR file**

Now Click on the Run tab and click Run-Configurations. Click on New Configuration button on the left-top side and Apply after filling the following properties.

**Step 10 - Export JAR file**

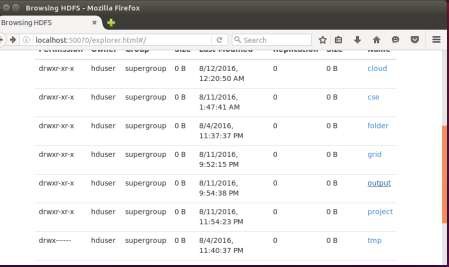
Now click on File tab and select Export. under Java, select Runnable Jar.

In Launch Config – select the config fie you created in **Step 9** (WordCountConfig).

* + Select an export destination (lets say desktop.)
  + Under Library handling, select Extract Required Libraries into generated JAR and click Finish.
  + Right-Click the jar file, go to Properties and under **Permissions**tab, Check Allow executing file as a program. and give Read and Write access to all the users

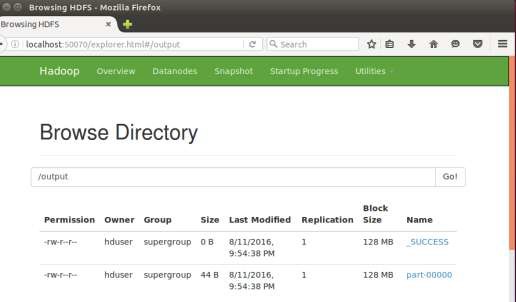
**Step 11 – Go back to old Terminal for Execution of WordCount Program**

**$hadoop jar wordcount**.**jar/usr/local/hadoop/input/usr/local/hadoop/output**



**Step 12 – To view results in old Terminal**

$**hdfs dfs -cat /usr/local/hadoop/output/part-r-00000**



**Step 13 - To Remove folders created using hdfs**

**$ hdfs dfs -rm -R /usr/local/hadoop/output RESULT**

Thus the program to write a wordcount program to that demonstrates the use of Map and Reduce tasks is done and output is verified successfully.

**EX. NO: 8**

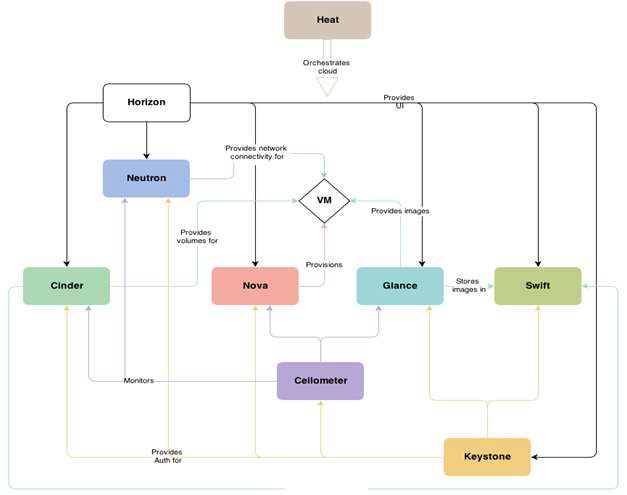
**DATE: CREATION AND MANAGEMENT OF INSTANCE IN OPENSTACK**

**AIM**

1. To install Single node OpenStack-devstack.
2. Creation, Management and Termination of Instances

**THEORY**

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center. All of the above components are managed through a dashboard which gives administrators control while empowering their users to provision resources through a web interface. More specifically though, it is a collection of open source software that allows us to perform certain functions on the cloud. OpenStack was a pilot project launched by Rackspace and NASA which was founded in July 2010. The purpose behind the project was to provide open source software that enables any organization to create and offer cloud computing services running on standardized hardware.



**OpenStack Architecture**

Just like as the Linux distribution has multiple flavors that are supported by different foundations like RedHat and SUSE, it is believed that in the near future there is a strong possibility that OpenStack will have distributions, as well by the leading players and contributors to the open source project, including: RedHat, Ubuntu, and more.

OpenStack has a modular architecture with various code names for its components:

* + Nova (Compute )
  + Swift (Object Storage )
  + Cinder (Block Storage)
  + Glance (Image Service)
  + Neutron (Networking)
  + Horizon (Dashboard )
  + Ceilometer (Telemetry)
  + Heat (Orchestration)

**PRODEDURE**

1. **To install Single node OpenStack-devstack.**

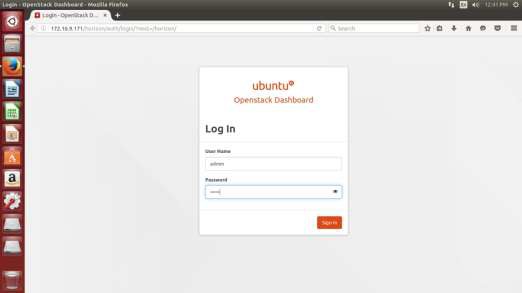
On execution of following commands, the openstack commands like nova, neutron, cinder, glance, keystone, glance, swift, horizon, celiometer would be installed. At the end of installation, IP will be generated.

root@ubuntu: **sudo apt-get update**

root@ubuntu: **sudo apt-get install git**

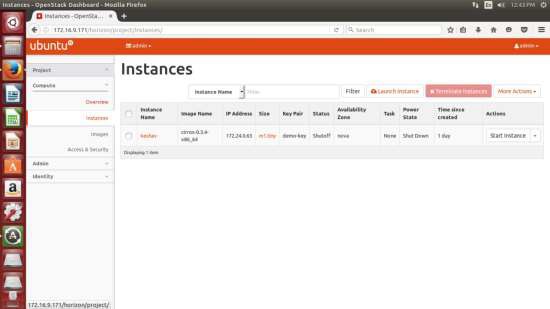
root@ubuntu: **git clone https://git.openstack.org/openstack-devstack/devstack**

root@ubuntu: **cd devstack** root@ubuntu: **ls** root@ubuntu: **./stack.sh**



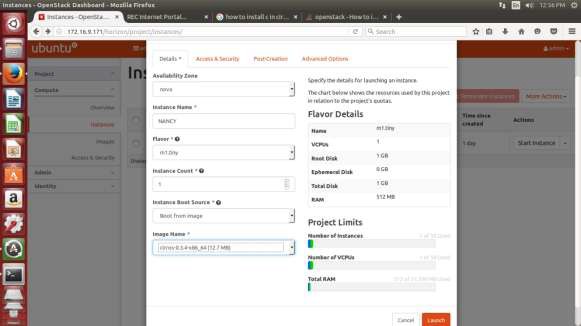
**Openstack Dashboard**

1. **Creation, Management and Termination of Instances Step: 1**–Click Compute -> Instances -> Launch Instance



**Step: 2 –** On clicking Launch Instance, the following dialog appears, which shows number of Instances, VCPUs, and RAM.

* Give an instance name.
* Select Flavor and Instance Count based on requirement
* Select Boot source -> Boot from image
* Launch Instance.



**Step: 3 – Instance in running condition**



**RESULT**

Thus the procedure to install Single node OpenStack-devstack, Creation, Management and Termination of Instances is completed successfully.

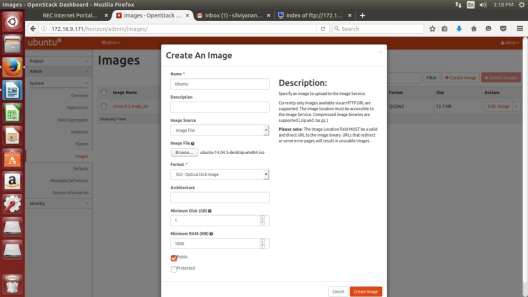
**EX. NO: 9**

**DATE: OPENSTACK - ADD AND DELETE IMAGE AND VOLUME**

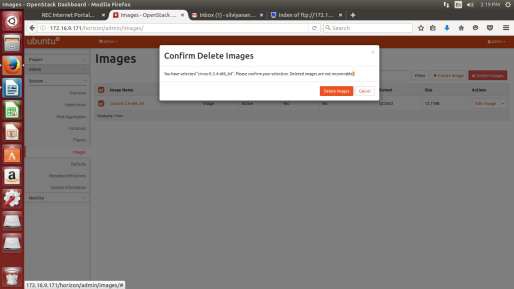
**AIM**

1. To Upload and Deletion of images.
2. Adding Volume

**Step: 1 –**Click System -> Images -> Create Image -> Add details.



**Step: 2 -** Click System -> Images -> delete Image.



1. **Adding Volume**

**Step 1** - Log in to the dashboard.

**Step 2** - Select the appropriate project from the drop down menu at the top left.

**Step 3** - On the Project tab, open the Compute tab and click Volumes category.

**Step 4** - Click Create Volume.

In the dialog box that opens, enter or select the following values. Volume Name: Specify a name for the volume.

Description: Optionally, provide a brief description for the volume. Volume Source: Select one of the following options:

* + No source, empty volume: Creates an empty volume. An empty volume does not contain a file system or a partition table.
  + Image: If you choose this option, a new field for Use image as a source displays. You can select the image from the list.
  + Volume: If you choose this option, a new field for Use volume as a

source displays. You can select the volume from the list. Options to use a snapshot or a volume as the source for a volume are displayed only if there are existing snapshots or volumes.

Type: Leave this field blank.

Size (GB): The size of the volume in gibibytes (GiB).

Availability Zone: Select the Availability Zone from the list. By default, this value is set to the availability zone given by the cloud provider (for example, us-west or apac-south). For some cases, it could be nova.

**Step 5** - Click Create Volume.

The dashboard shows the volume on the Volumes tab.



**RESULT**

Thus the procedure to Upload and Deletion of images and adding a volume is completed successfully.

**EX. NO: 10**

**DATE: STORAGE VIRTUALIZATION USING FREEBSD**

**AIM**

To install storage controller and interact with it using FreeBSD.

**PROCEDURE**

**To install storage controller and interact with it using FreeBSD.**

**Step: 1** - Open VMware and select “Create a new Virtual Machine” and increase the RAM size to

4GB

**Step: 2** - In the pop-up box select the installer disc image file (iso) option and browse the file FREENAS.iso and click next.



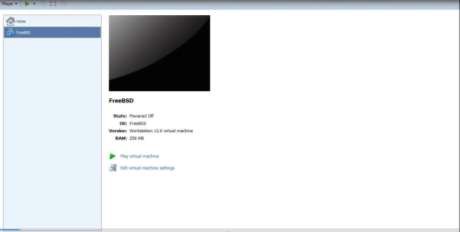
**Step**:**3** - In the next page of the wizard, provide a name for the virtual machine and select the location at which it has to be installed and click next.



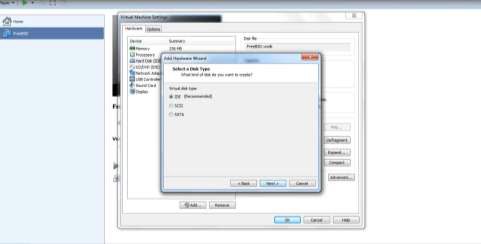
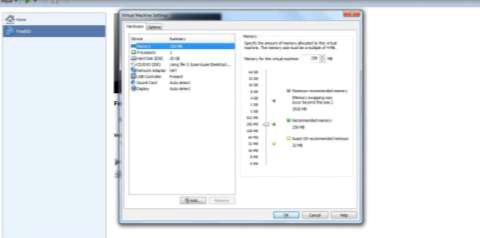
**Step: 4** - In the next page, provide the maximum disc capacity as 20 GB and select store virtual disc as single file option and then click next. Then click finish.



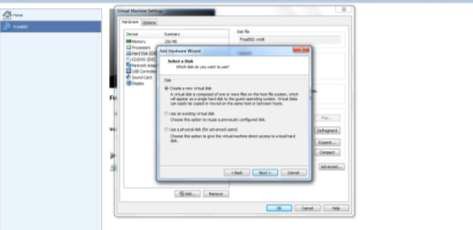
**Step: 5** - A new virtual machine with the provided name would appear in the VMware home screen. Click on edit virtual machine settings.



**Step: 6** - Select hard Disk from the virtual machine settings dialog box. Select add button. In the Add Hardware wizard select the hardware type as hard disk and click next. Select the virtual disk type as IDE and click next.

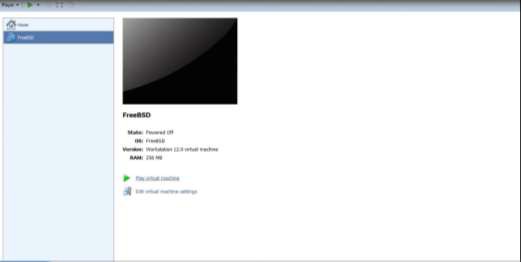


**Step: 7** - In the next page select the Create a new virtual disk option and click next. Provide the disk capacity as 20GB and select store virtual disk as single file option and click next. Specify the disk file and click finish.

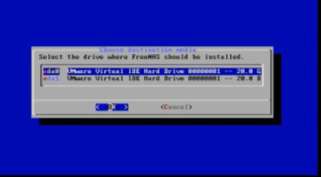


**Step: 8** - Close the virtual machine settings dialog box.

**Step: 9** - Select play virtual machine option available in the home screen of the VMware.



**Step: 10**- Proceed with the normal installation wizard of FREENAS. Wait until the FREENAS gets installed.





**Step: 11** - After successful installation of FREENAS, reboot the VMware.



**Step: 12**– IP will be generated.



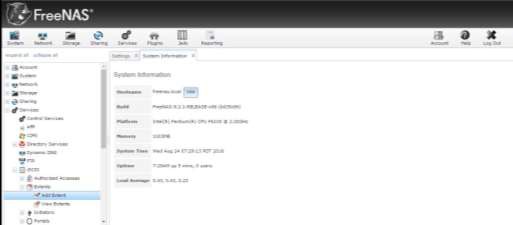
**Step: 13**- Open web Browser and type the IP address 192.168.170.128 to access the FREENAS dashboard. Enter the new password and log in.



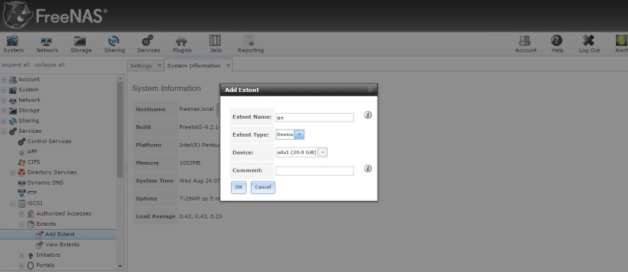
**Step: 14**- In the FREENAS dashboard page, find the menu services in the left pane.



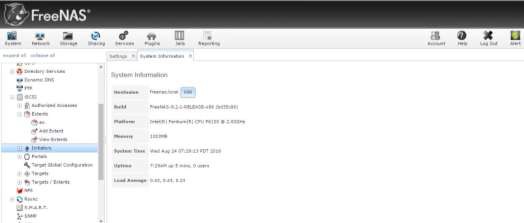
**Step: 15**- Select services->iscsi->Extents->Add extents.



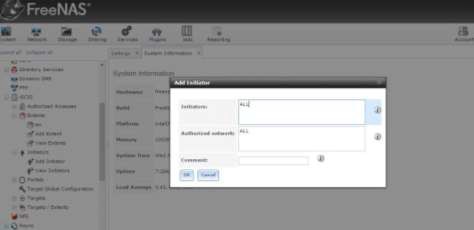
**Step: 16** - In the Add Extent dialog box enter the extent name as en and select the extent type as Device and select ada1(20GB) from the device drop down list box and click ok.



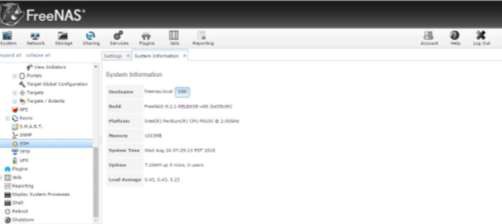
**Step: 17**- Select services->iscsi->Initiators->Add initiators.



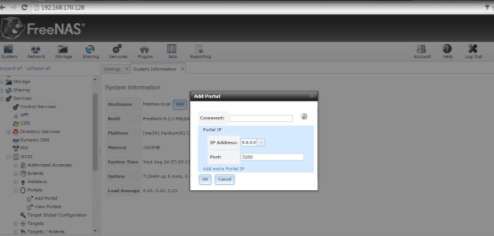
**Step: 18**- In the Add Initiators dialog box enter ALL in the initiators box and ALL in Authorised network box and click ok.



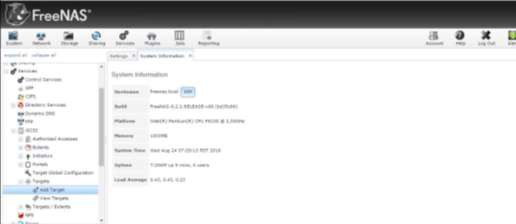
**Step: 19**- Select services->iscsi->Portals->Add portals.



**Step: 20**- In the Add portal dialog box select 0.0.0.0 from the IP Address drop down list box and enter 3260 as Port then click ok.



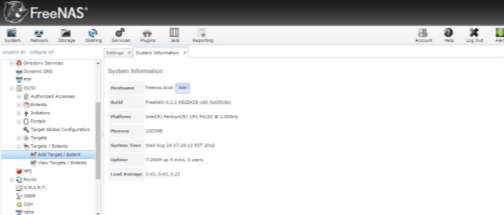
**Step: 21**- Select services->iscsi->Targets->Add Target.



**Step: 22**- In the Add Target dialog box, Enter the target name as t1, select portal Group Id as 1 and Initiator group Id as 1 and then click ok.



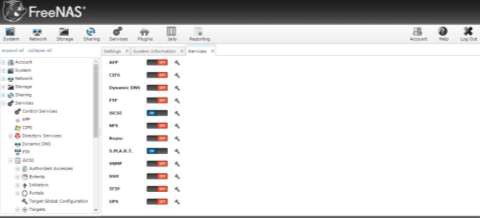
**Step: 23**- Select services->iscsi->Targets/Extents->Add Target/Extent.



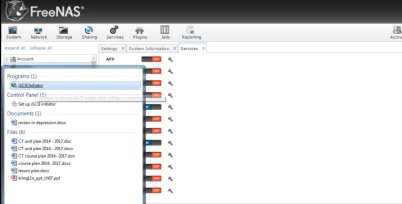
**Step: 24**- In the Add Target/Extent dialog box, Select t1 in the Target list box and en in the Extent list box and click ok.

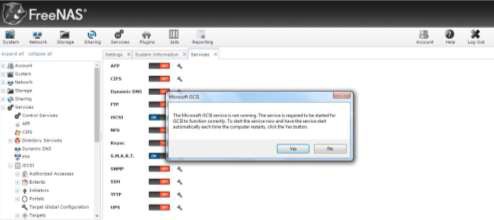


**Step: 25**- From the menu bar select services and turn ON iSCSI and S.M.A.R.T.



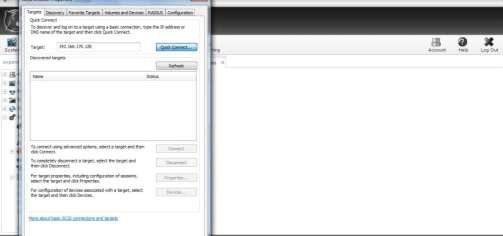
**Step: 26**- Search for the program iSCSI initiator and click yes in the pop-up box.

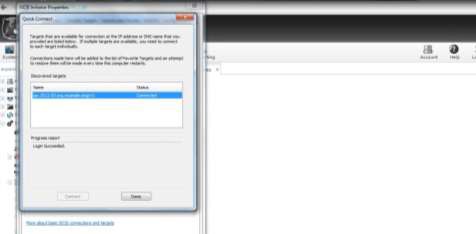


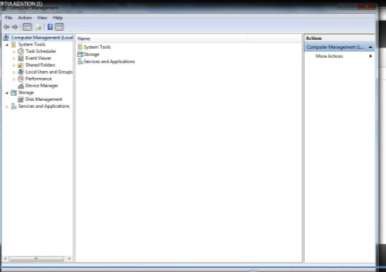


**Step: 27**- In the iSCSI initiator properties dialog box , enter the IP Address 192.168.170.128 in the Target box and click Quick connect. After successful connection the Quick connect dialog box appears with the status as connected. Click Done and close the iSCSI properties window.

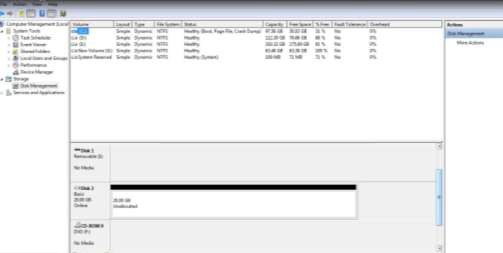
**Step: 28**- Open Computer Management by right clicking on My computer and selecting Manage. Select Disk Management from the right pane of Computer Management.

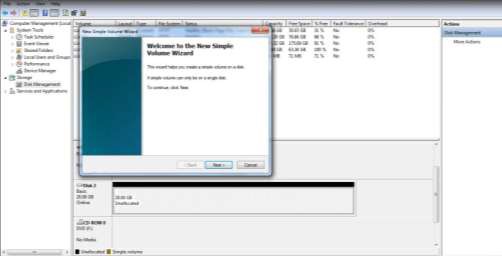




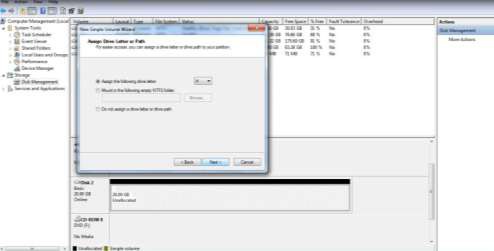


**Step: 29**- In the disk volume information page right click on the 20GB unallocated Disk and select New Simple Volume.

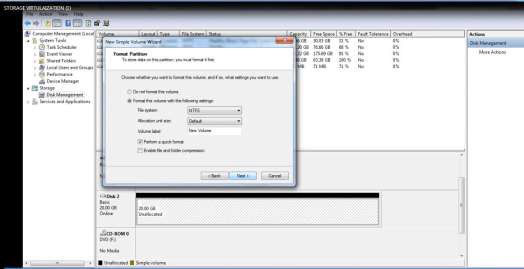


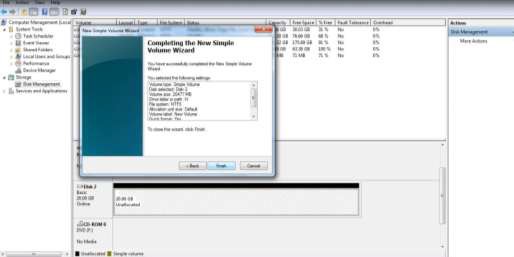


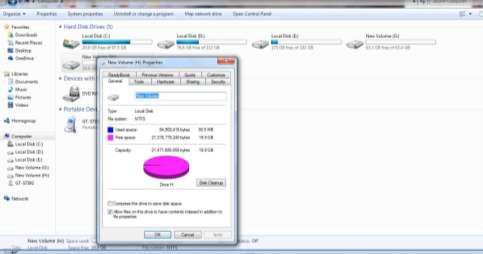
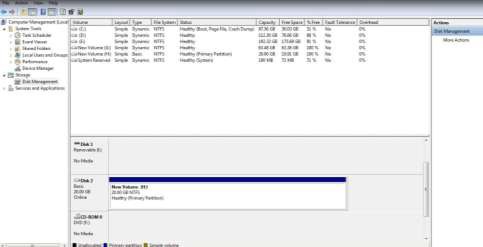
**Step: 30**- In the New Simple Volume wizard click next and select The Drive letter to be allocated as H and click next.



**Step: 31**- In the Format partition page select Format Partition with the following Settings option with file system as NTFS and check perform Quick format option and click next. Click finish.







**RESULT**

Thus the procedure to install storage controller and interaction is done successfully using FreeBSD.

**BASIC UNDERSTANDING ON GRID COMPUTING**

**Grid Computing**

Computational Grid is a collection of distributed, possibly heterogeneous resources which can be used as an ensemble to execute large-scale applications

* Computational Grid also called metacomputer
* Term computational grid comes from an analogy with the electric power grid:
  + Electric power is ubiquitous
  + Don't need to know the source (transformer, generator) of the power or the power company that serves it

**Grid applications include Distributed Supercomputing**

* Distributed Supercomputing applications couple multiple computational resources - supercomputers and/or workstations
* Distributed supercomputing applications include SFExpress (large-scale modeling of battle entities with complex interactive behavior for distrtibuted interactive simulation), Climate Modeling (modeling of climate behavior using complex models and long time- scales)

**High-Throughput Applications**

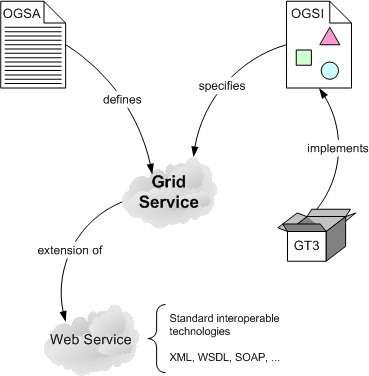
* Grid used to schedule large numbers of independent or loosely coupled tasks with the goal of putting unused cycles to work
* High-throughput applications include RSA keycracking, seti@home (detection of extra- terrestrial communication)

**Data-Intensive Applications**

* Focus is on synthesizing new information from large amounts of physically distributed data
* Examples include NILE (distributed system for high energy physics experiments using data from CLEO), SAR/SRB applications, digital library applications

**GLOBUS TOOLKIT**

The Globus Toolkit is a software toolkit that allows us to program grid-based applications. The third and latest version of the toolkit is based on something called Grid Services. Before defining Grid Services, we're going to see how Grid Services are related to a lot of acronyms you've probably heard (OGSA, OGSI for example), but aren't quite sure what they mean exactly. The following diagram summarizes the major players in the Grid Service world:



**Grid Service**

Grid Services are defined by OGSA. The Open Grid Services Architecture (OGSA) aims to define a new common and standard architecture for grid-based applications. Right at the center of this new architecture is the concept of a Grid Service. OGSA defines what Grid Services are, what they should be capable of, what types of technologies they should be based on, but doesn't give a technical and detailed specification (which would be needed to implement a Grid Service).

Grid Services are specified by OGSI. The Open Grid Services Infrastructure is a formal and technical specification of the concepts described in OGSA, including Grid Services. The Globus Toolkit 3 is an implementation of OGSI. GT3 is a usable implementation of everything that is specified in OGSI (and, therefore, of everything that is defined in OGSA). Grid Services are based on Web Services. Grid Services are an extension of Web Services. Grid Services provide some cool features which are not available with standard Web Services.

1. Grid Services are Stateful unlike Web Services which are stateless.
2. Grid Services implement Factory Pattern. (Since you have already studied Patterns I won't describe it further).
3. Two implementation approaches: A Grid Service can be implemented either by inheriting from a skeleton class or by using a delegation model, where incoming calls are delegated to a series of classes called operation providers.
4. Lifecycle management: Grid Services provide the necessary tools, such as callback functions during special moments in a Grid Service's lifetime (creation time, destruction time, etc.), to effectively manage its lifecycle (for example, to make Grid Services persistent).
5. Service Data: A Grid Service can have a set of associated service data that describes it. This is not to be confused with WSDL, which describes details like methods, protocols, etc. Service data is particularly useful to index Grid Services according to their characteristics and capabilities.
6. Notifications: We can configure a Grid Service to be a notification source, and certain clients to be notification sinks (or subscribers). This means that if a change occurs in the Grid Service, that change is notified to all the subscribers.