REVA UNIVERSITY

BENGALURU INDIA

CLOUD COMPUTING LAB

B18CS6070

6th Semester

B.Tech in Computer Science & engineering

Name	
SRN	
Branch	
Semester	
Section	
Academic Year	

Sl. No.	Lab Sessions
1	Demonstrate on the Virtual Environment on hypervisor. a) Communication between the VM's. b) The backup and restore mechanism.
2	Demonstrate the mechanism of cloning and create a switch with multiple networks having the different VM's.
3	Demonstrates how to simulate a Data Center with one host and run one Cloudlet on it using cloudsim.
4	Demonstrates how to create a datacenter with one host, two virtual machines and run two cloudlets on it. Both virtual machine (vm1,vm2) has same machine configuration.
5	Demonstrates how to create a datacenter with two hosts, two virtual machines and run two cloudlets on it. The cloudlets run in VMs with different MIPS requirements. The second VM will have twice the priority of virtual machine one(VM1) and so cloudlet will receive twice CPU time to complete the execution
6	Demonstrate how to create two datacenters with one host each and run two cloudlets on them.
7	Evaluate the performance of MapReduce program on "word count" for different file size.
8	Evaluate the performance of MapReduce program on "character count" for different file size.
9	Appendix A, B, C, D

Session 1

1	**Problem Statement:**

Demonstrate on the Virtual Environment on hypervisor.

c) Communication between the VM's.
d) The backup and restore mechanism.

2	**Student Learning Outcomes:**

To install and setup the Virtual Machine on same ESXi.

To check the connectivity between the VM's

Demonstration of snapshot

3	**Theoretical Description:**

Virtual Machine (**VM**): is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.
There are different kinds of virtual machines, each with different functions:

- System virtual machines (also termed full virtualization VMs) provide a substitute for a real machine. They provide functionality needed to execute entire operating systems. A hypervisor uses native execution to share and manage hardware, allowing for multiple environments which are isolated from one another, yet exist on the same physical machine. Modern hypervisors use hardware-assisted virtualization, virtualization-specific hardware, primarily from the host CPUs.
- Process virtual machines are designed to execute computer programs in a platform-independent environment.

A VMware snapshot: is a copy of the virtual machine's disk file (VMDK) at a given point in time. Snapshots provide a change log for the virtual disk and are used to restore a VM to a particular point in time when a failure or system error occurs. Snapshots alone do not provide backup.

When you take a snapshot, you capture the state of the virtual machine settings and the virtual disk. If you are taking a memory snapshot, you also capture the memory state of the virtual machine. These states are saved to files that reside with the virtual machine's base files.

A snapshot consists of files that are stored on a supported storage device. A Take Snapshot operation creates .vmdk, -delta.vmdk, .vmsd, and .vmsn files. By default, the first and all delta disks are stored with the base .vmdk file. The .vmsd and .vmsn files are stored in the virtual machine directory.

4	**Requirements**

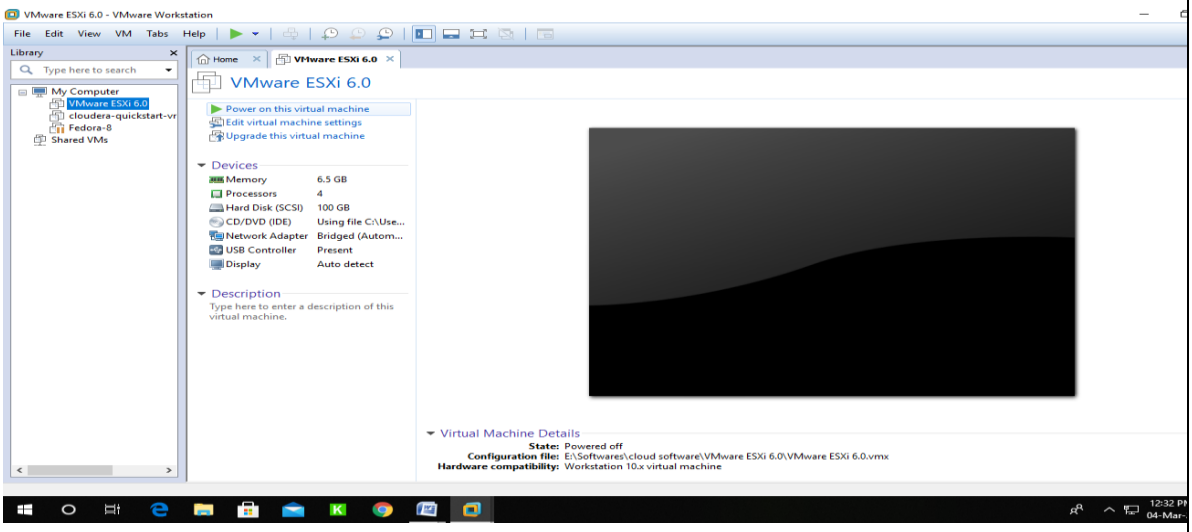
The following list of requirements as a starting point. Like physical computers, the virtual machines running under VMware Workstation generally perform better if they have faster processors and more memory.

- PC Hardware - Standard x86-compatible personal computer, 400 MHz or faster CPU minimum (500 MHz recommended), Multiprocessor systems supported, 64-bit processor support for AMD64 Opteron, Athlon 64 and Intel IA-32e CPU (including "Nocona").
- RAM - 8GB minimum

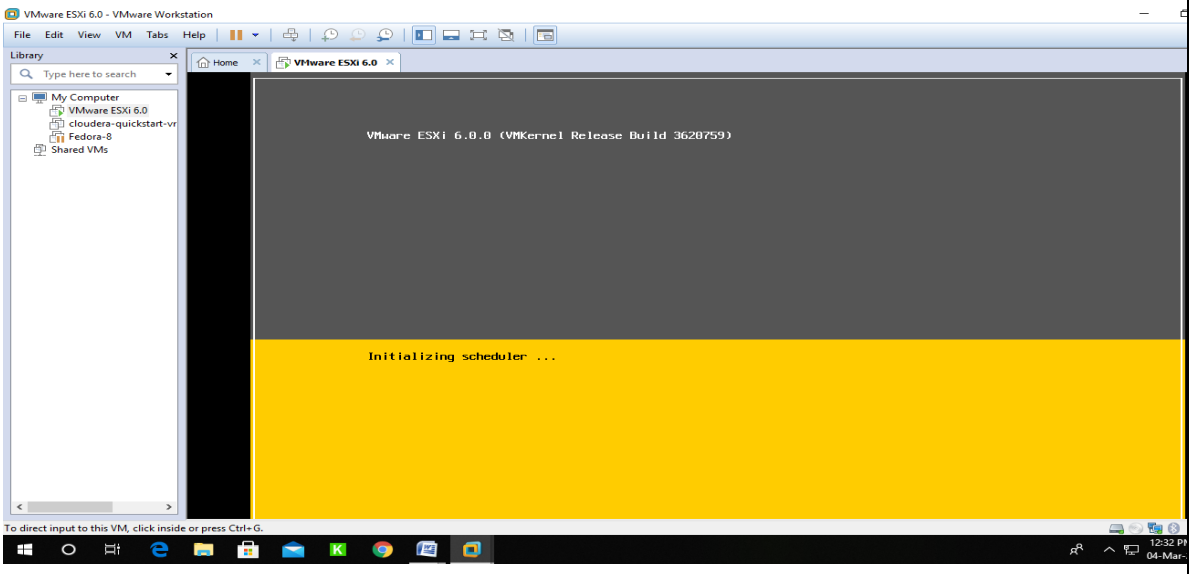
- Disk Drives - IDE and SCSI hard drives supported, up to 950GB capacity, At least 1GB free disk space recommended for each guest operating system and the application software used with it; if you use a default setup, the actual disk space needs are approximately the same as those for installing and running the guest operating system and applications on a physical computer.
- Local Area Networking (Optional)
- Host Operating System

5	**Procedure**

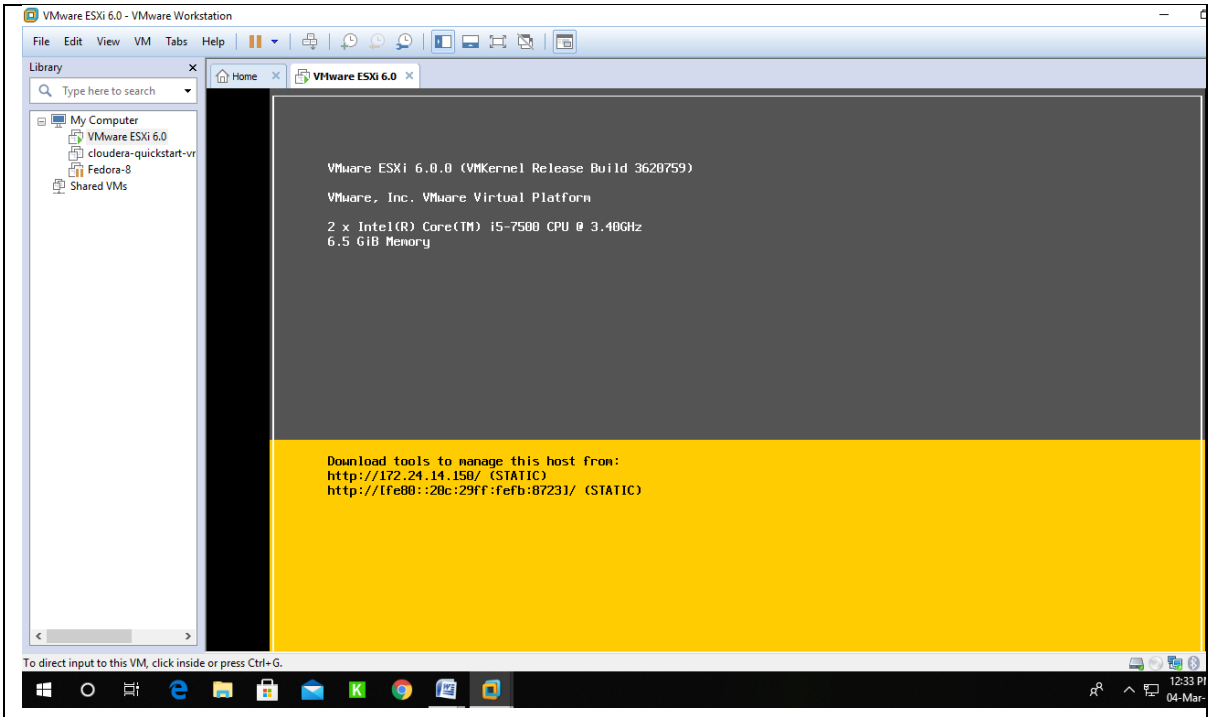
1) Go to VMware Workstation.



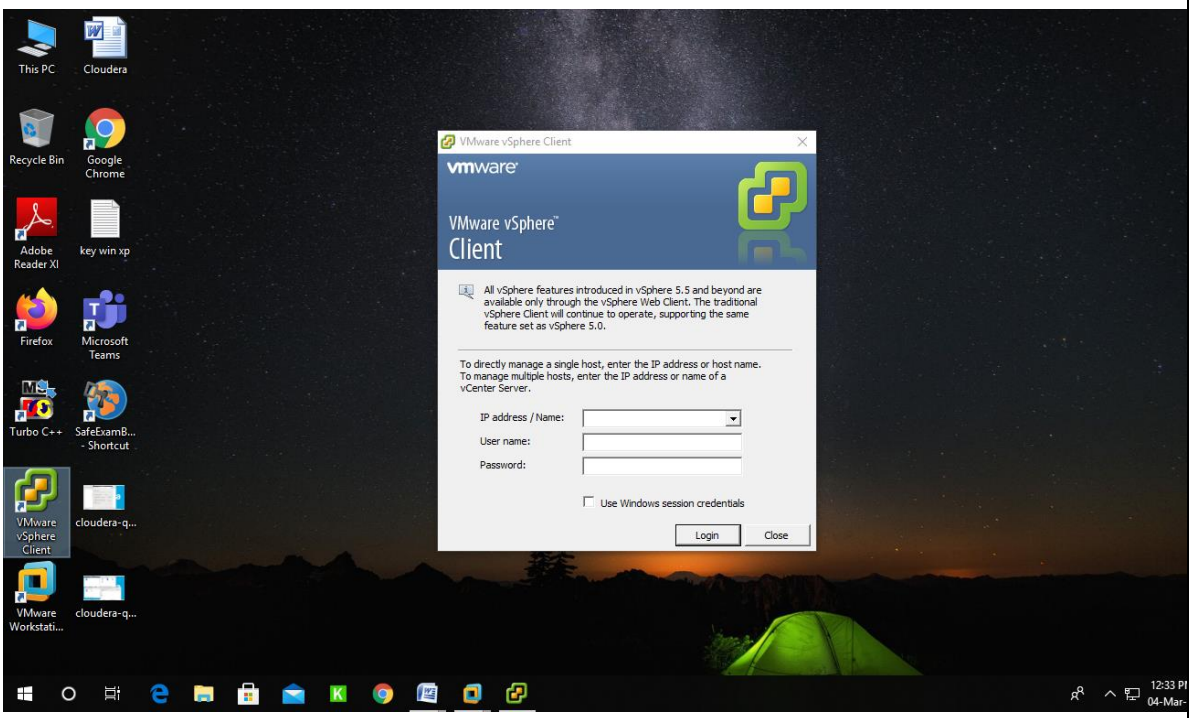
2) Power on the hypervisor (VMWare ESXi)



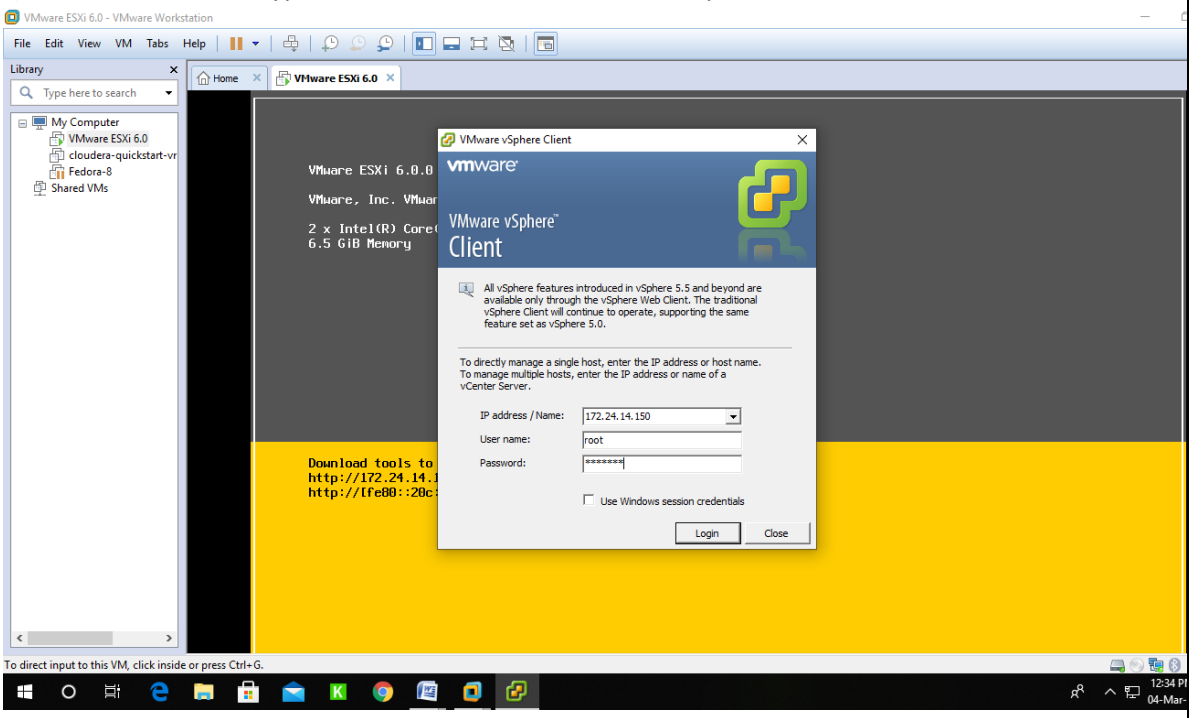
3) Hypervisor is ready. Note down the IPv4 address of hypervisor



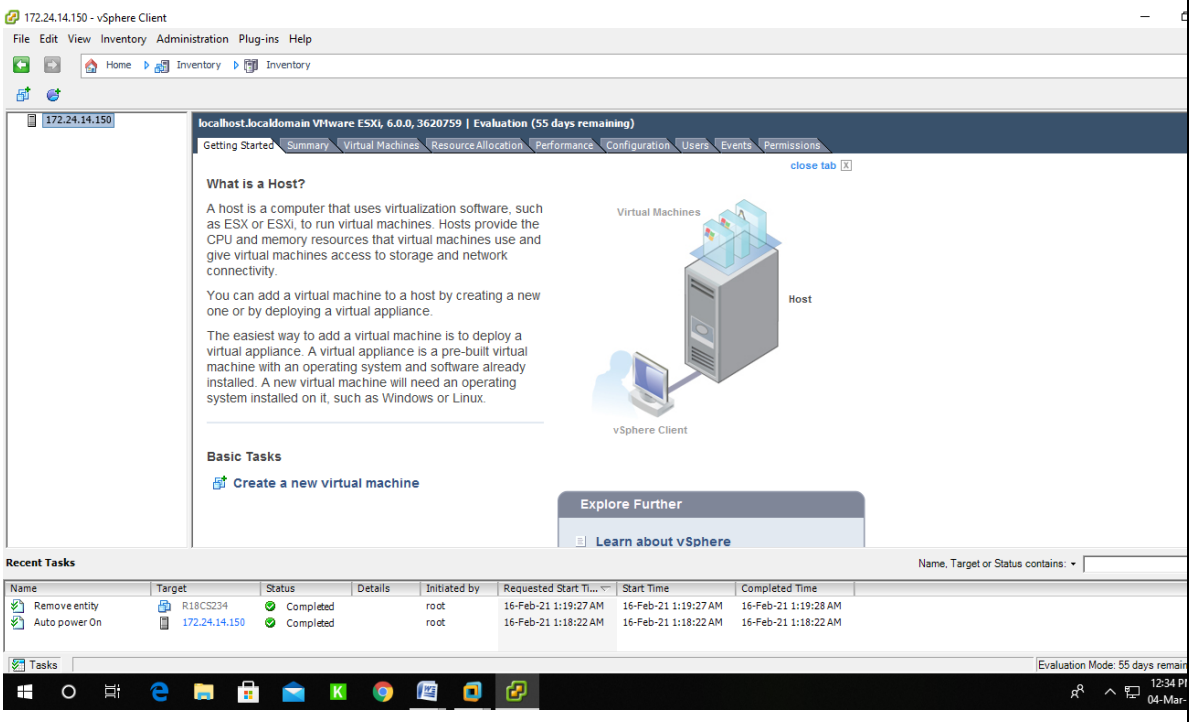
4)Start the vSphere Client.



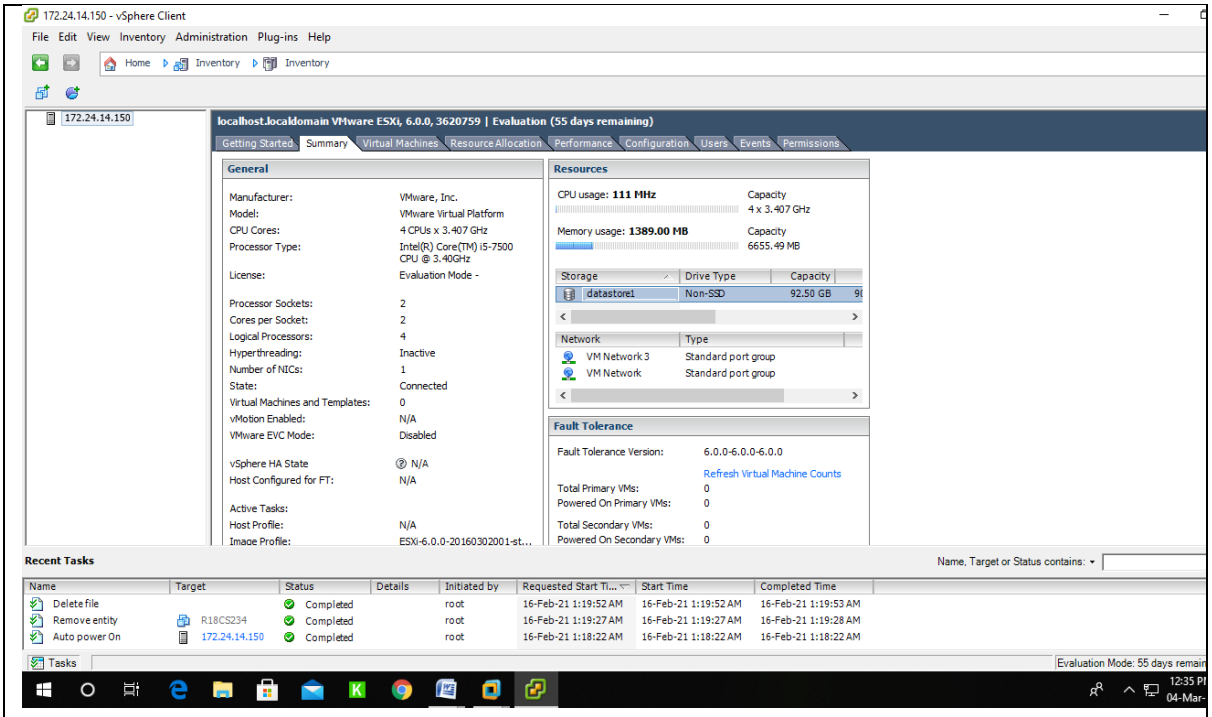
5)Provide the IPv4 of hypervisor and username as root and password as root123



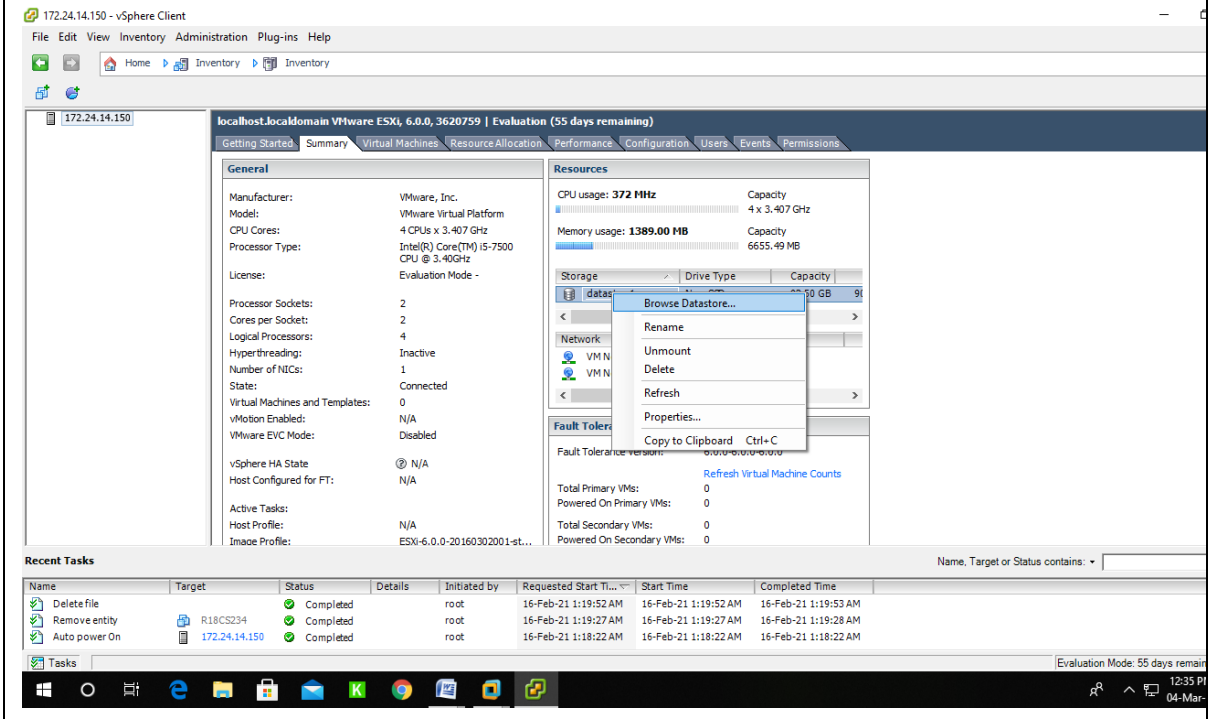
6) An inventory is ready for use



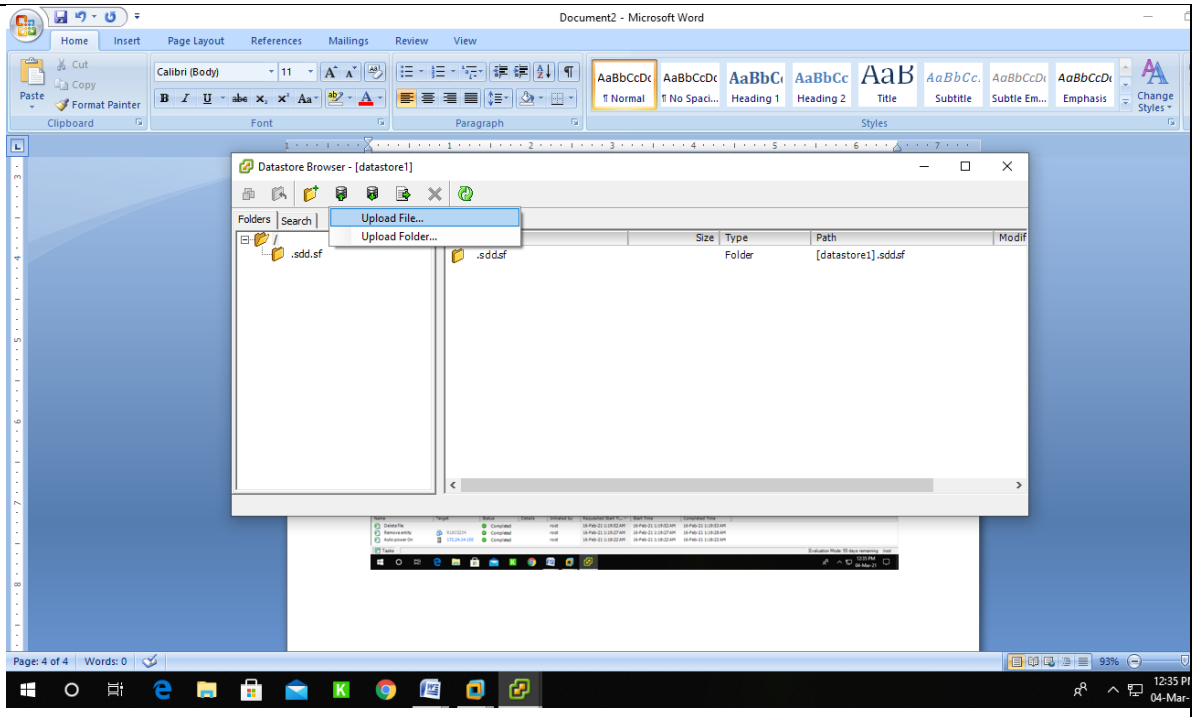
7) Select the IP of hypervisor. Goto Summary tab



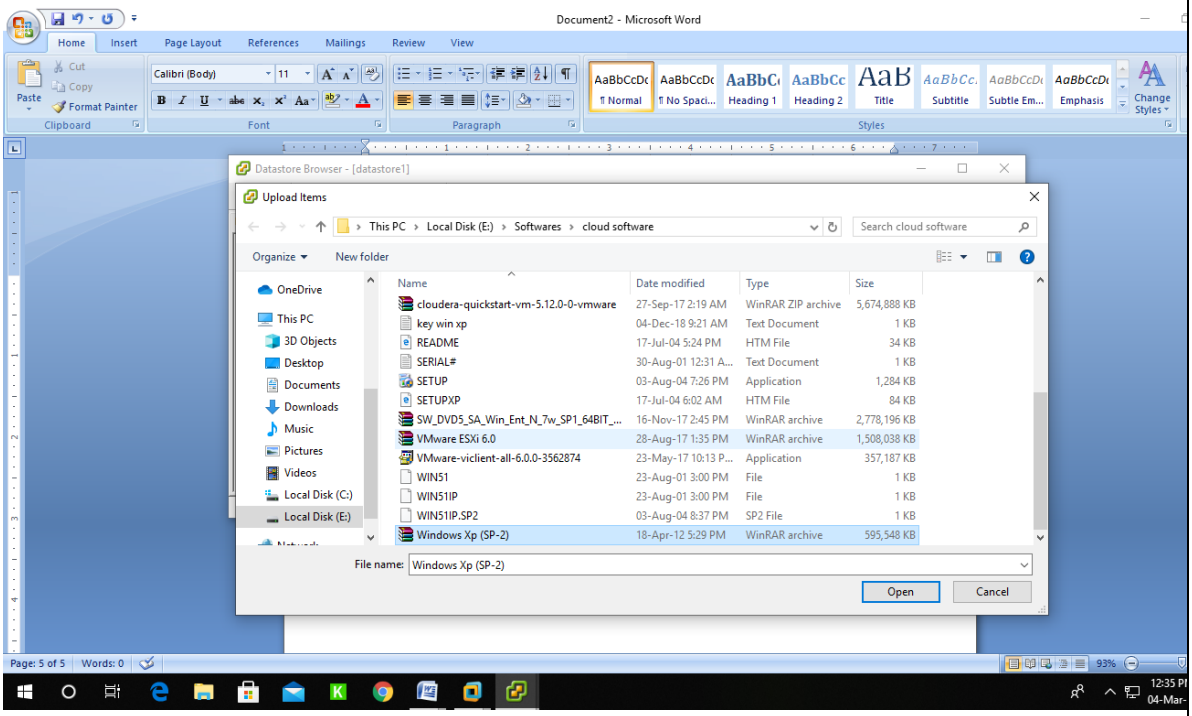
8) Select datastore. Right click and 'Browse Datastore'



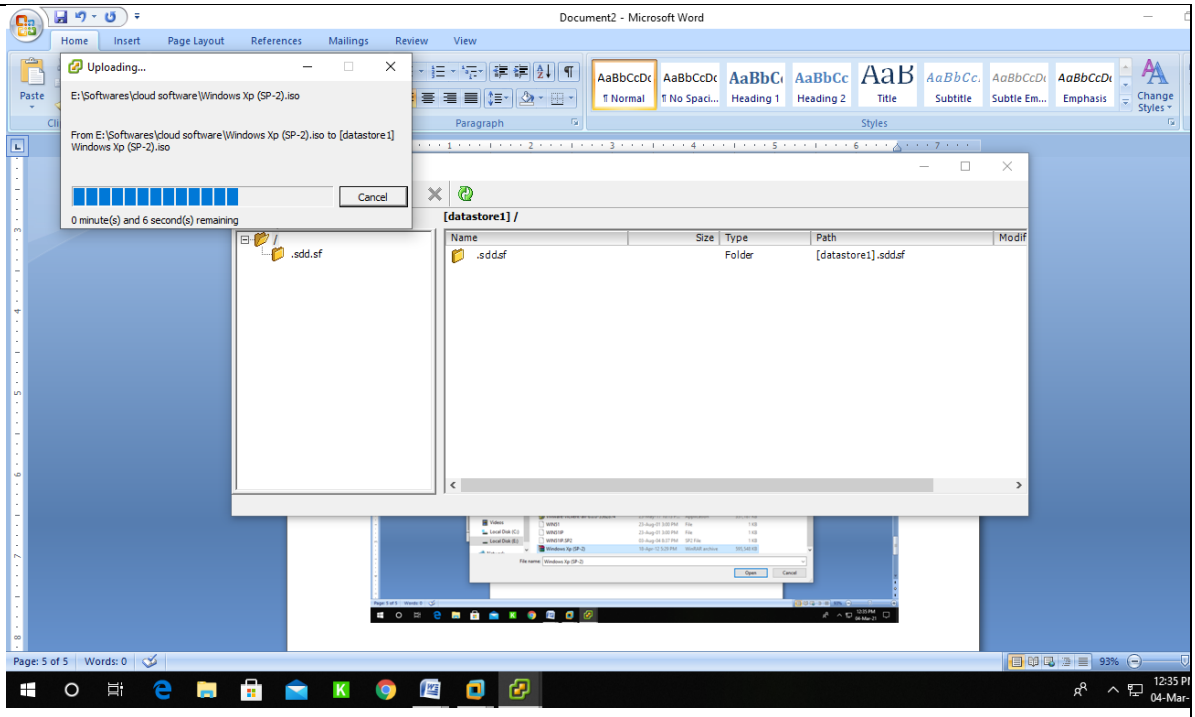
9) Upload the source file, windows xp and iso file.



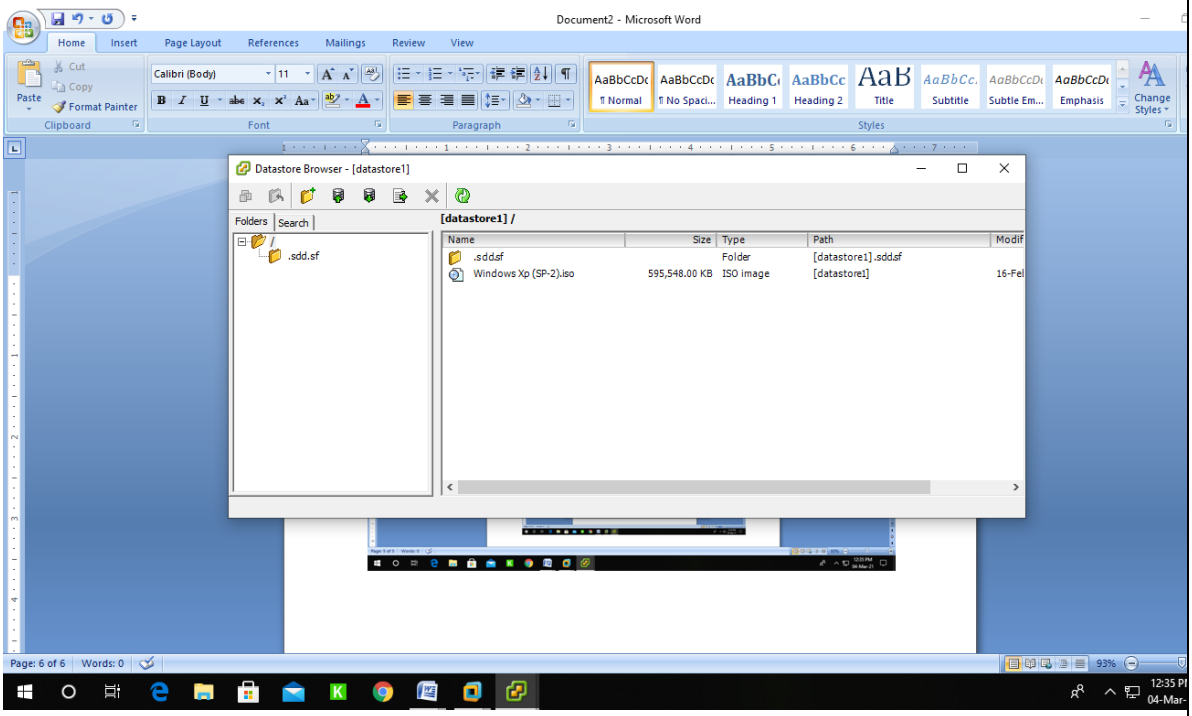
10) select the file from current location and upload.



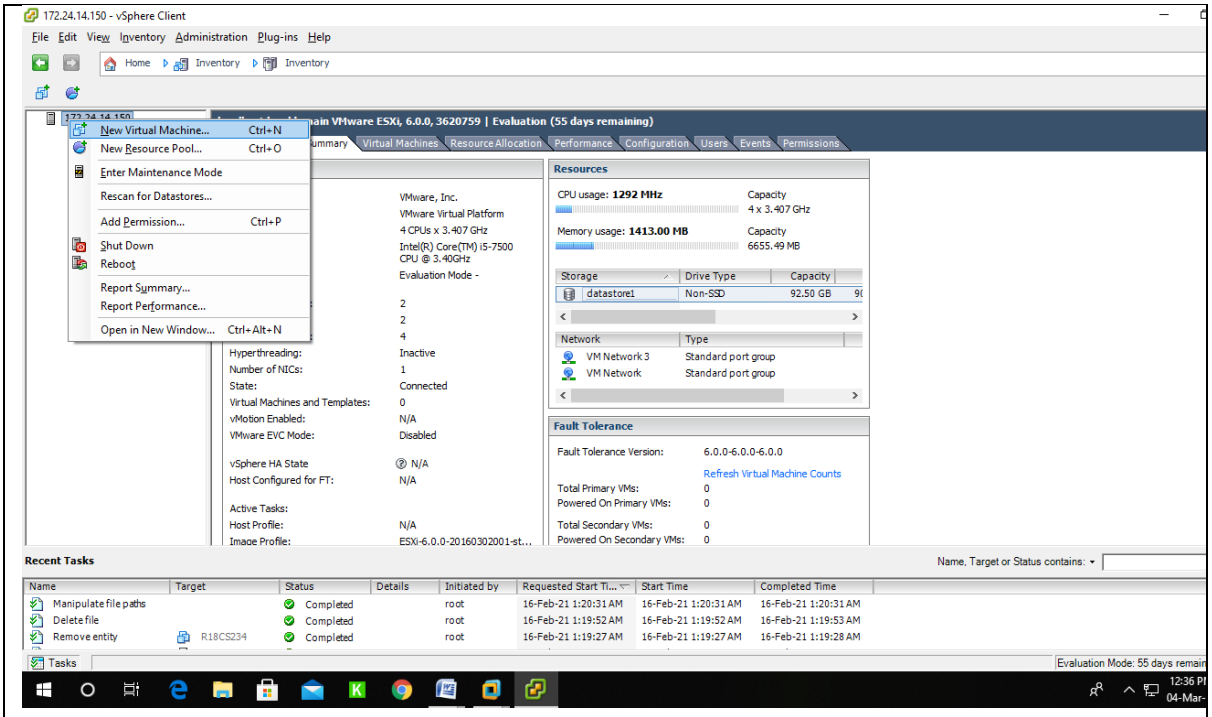
11) The uploading is in progress. May take couple of minutes.



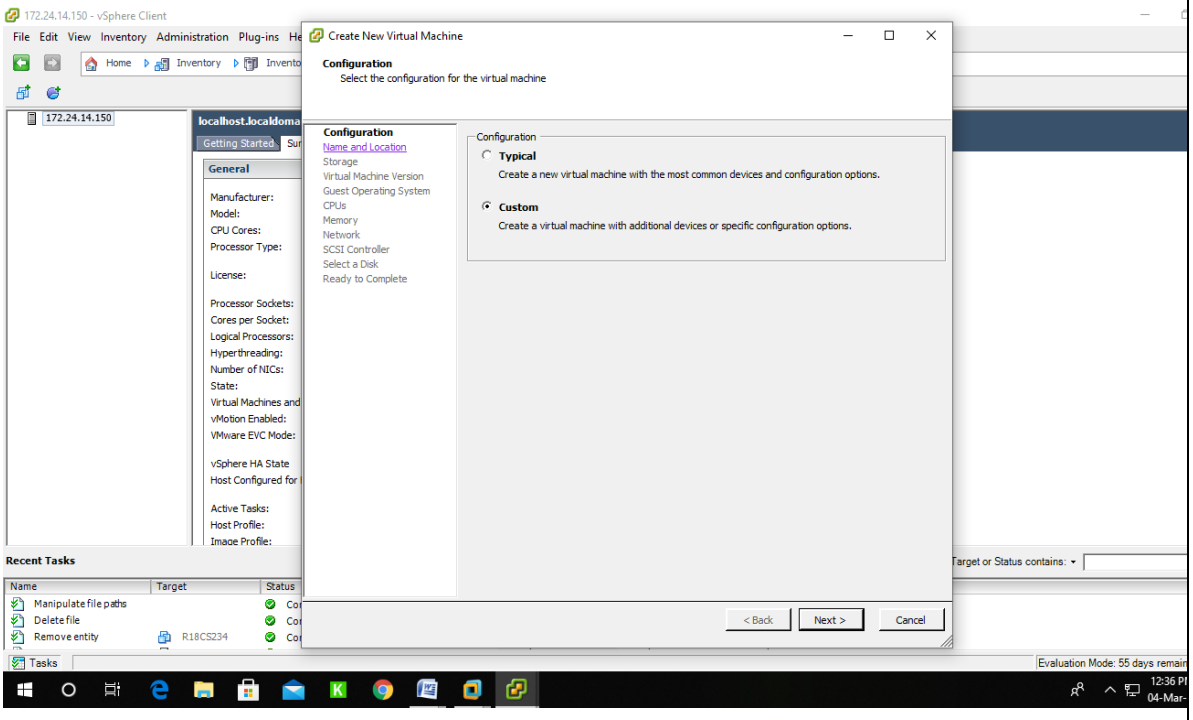
12) The course file is uploaded successfully to datastore.



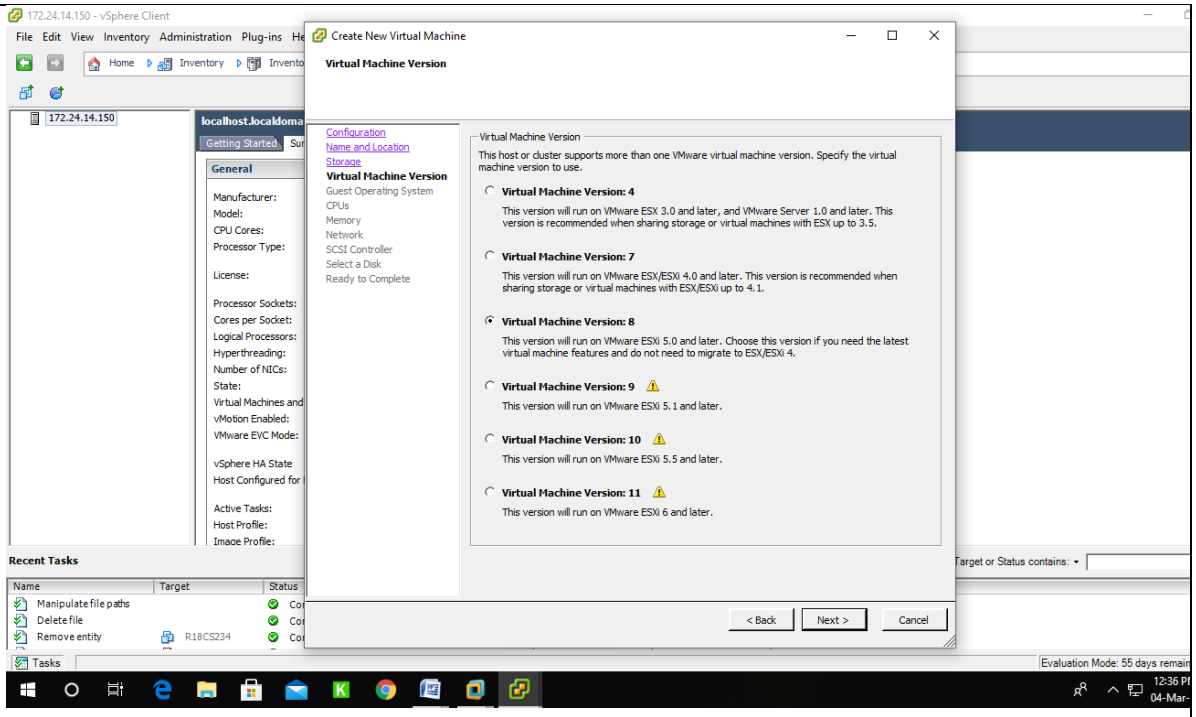
13) Select ip of hypervisor and right click, use the option ‘New Virtual Machine’



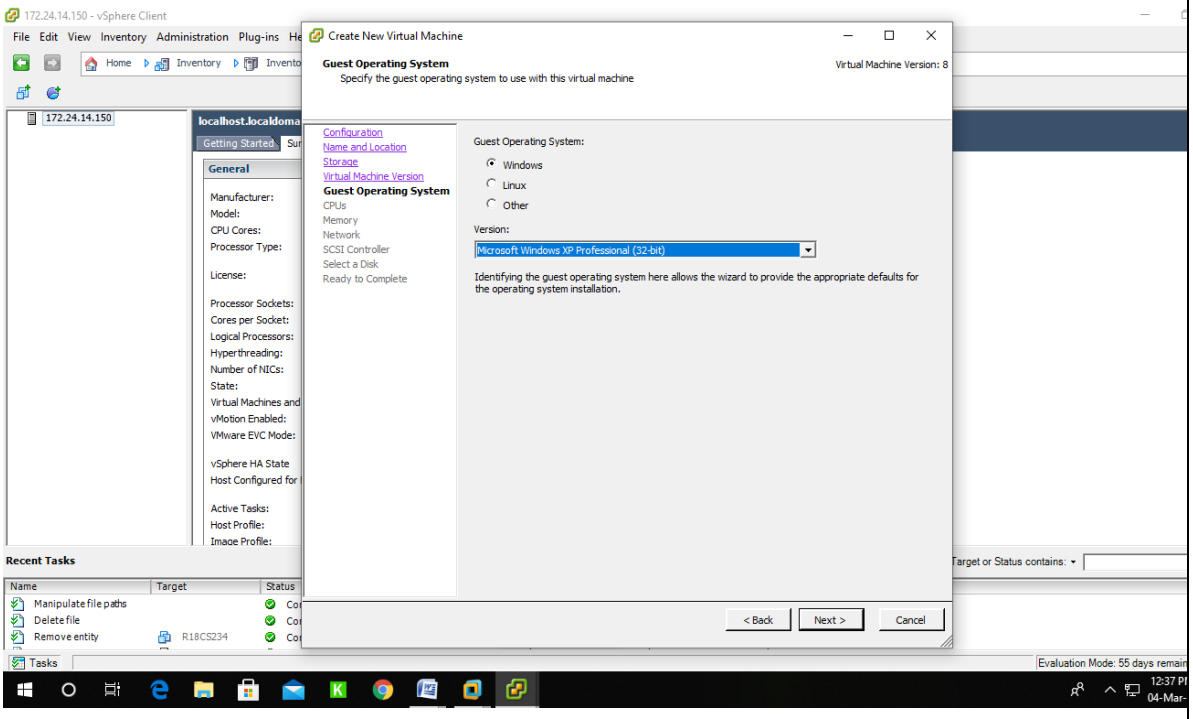
14) Select the Custom option, to set configure based on the physical machine. And proceed further.



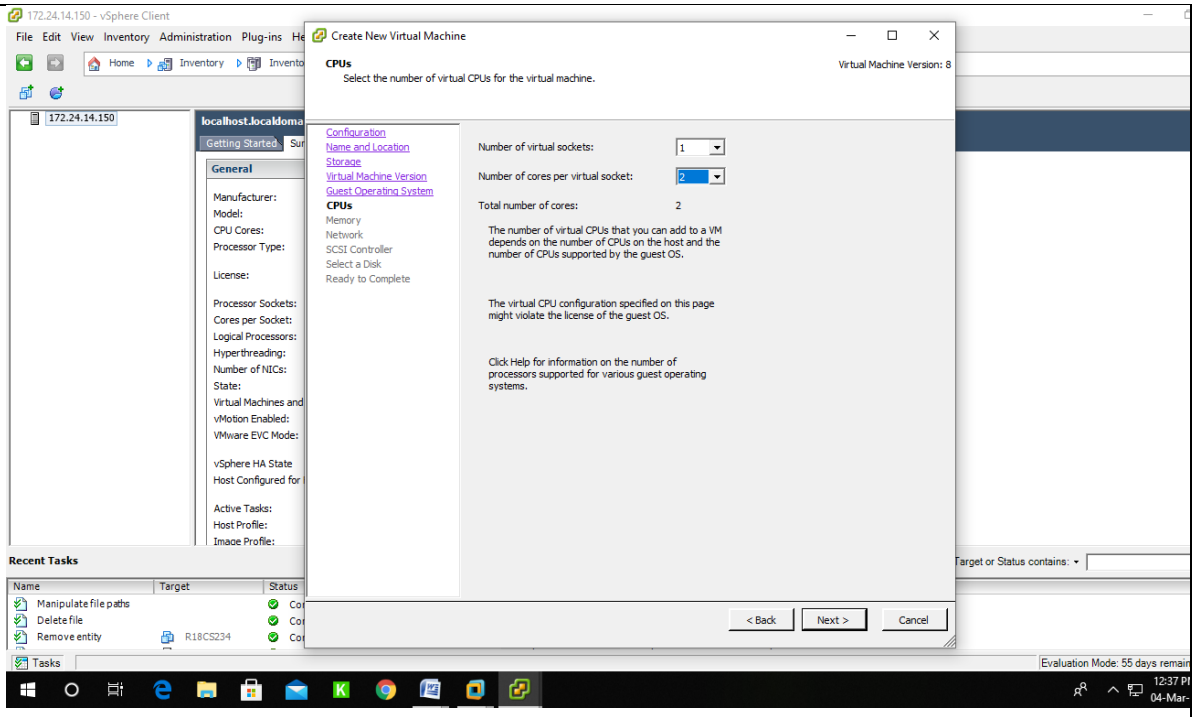
15) Select the VM version compatible to physical machine. Select 8 or less here.



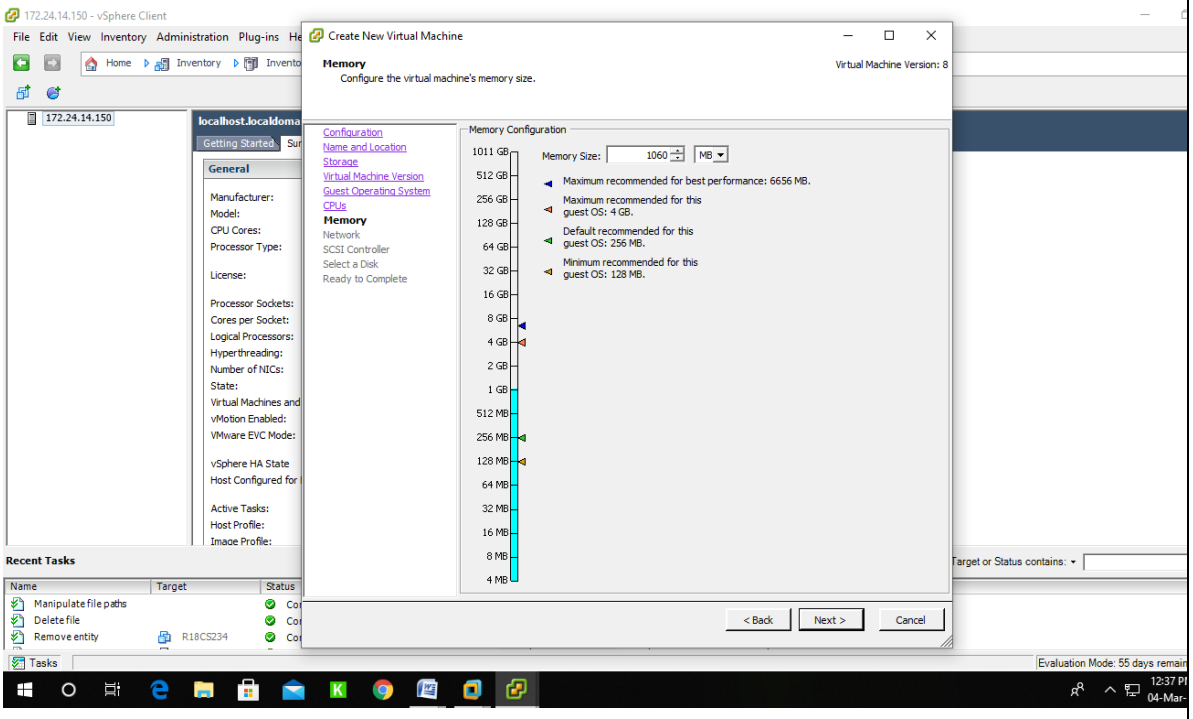
16) Choose the guest operating system. Here we are using windows xp with 32 bit.



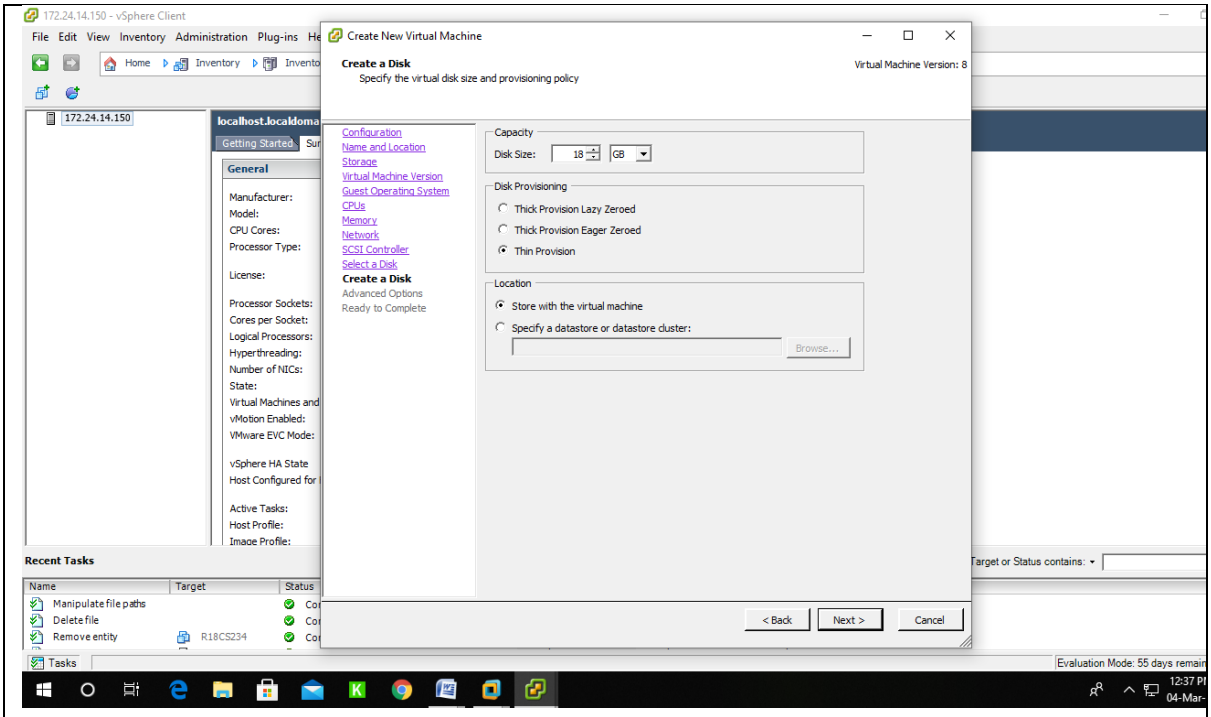
17) Set atleast 2 cores for the VM.



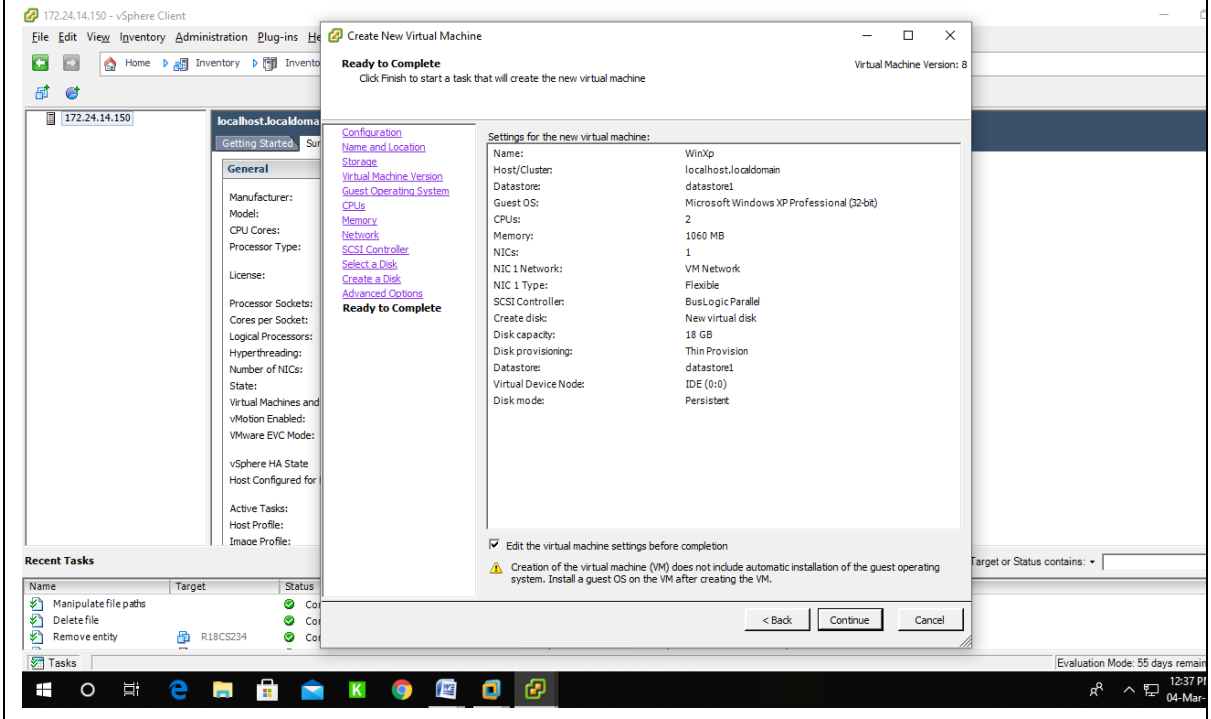
18) Set the RAM capacity for the VM, minimum of 1GB. And proceed further.



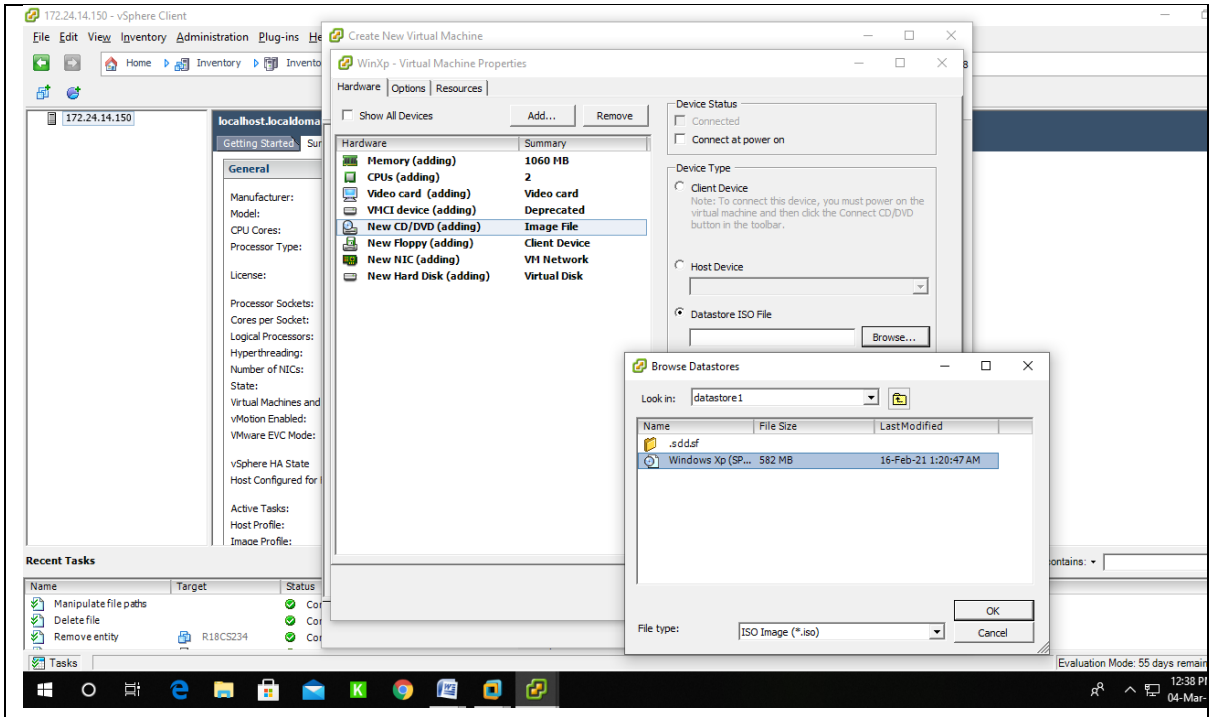
19) Set the hard disk capacity, minimum of 20GB.



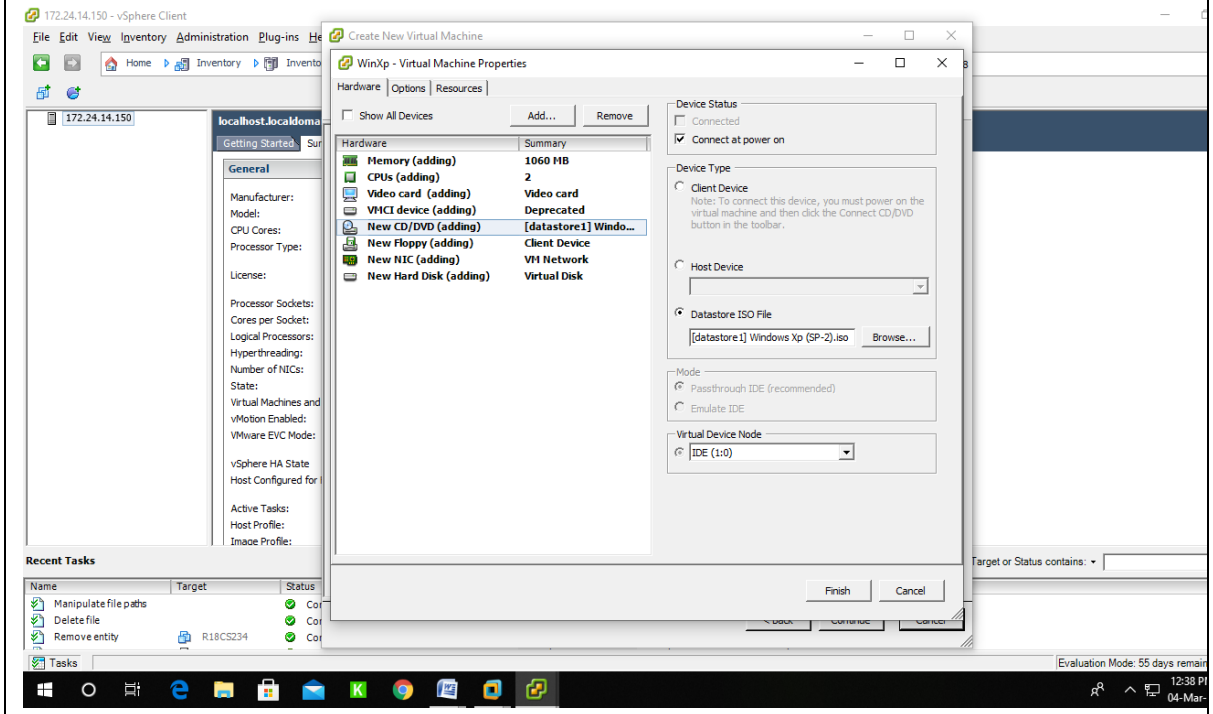
20) Proceed further and select the Edit option, by ticking.



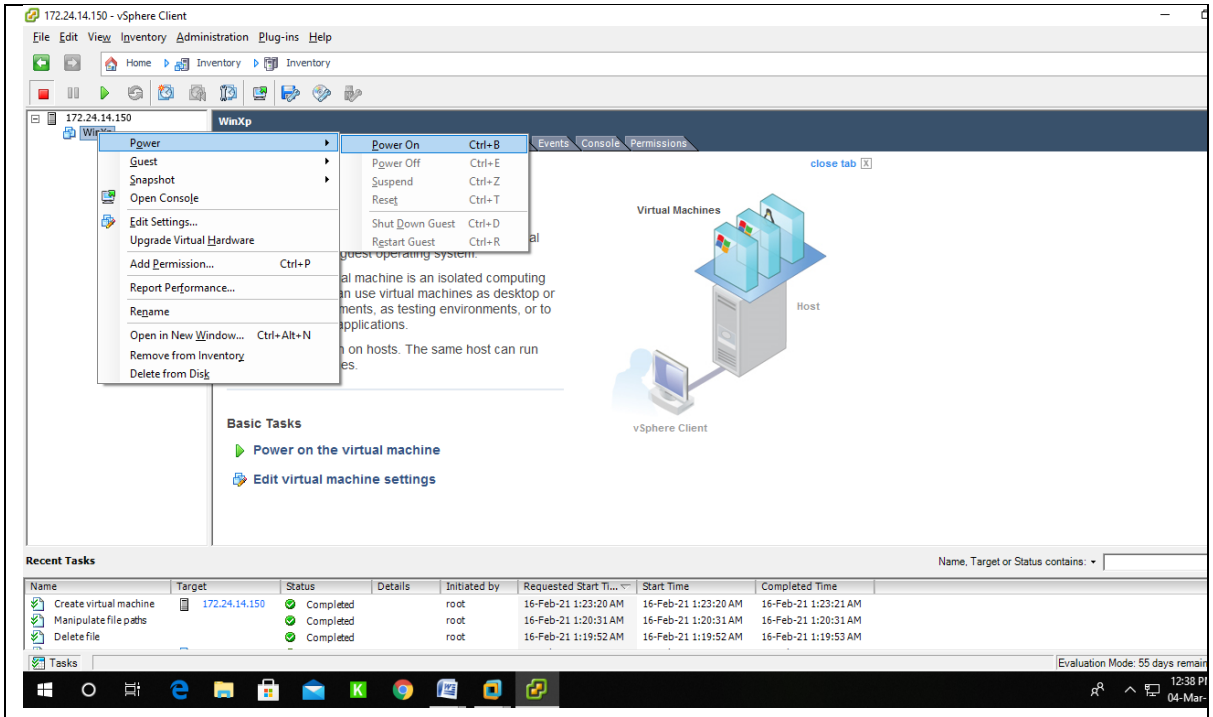
21) Goto the option, CD/DVD. Select datastore ISO file, browse and goto location where the source file is uploaded.



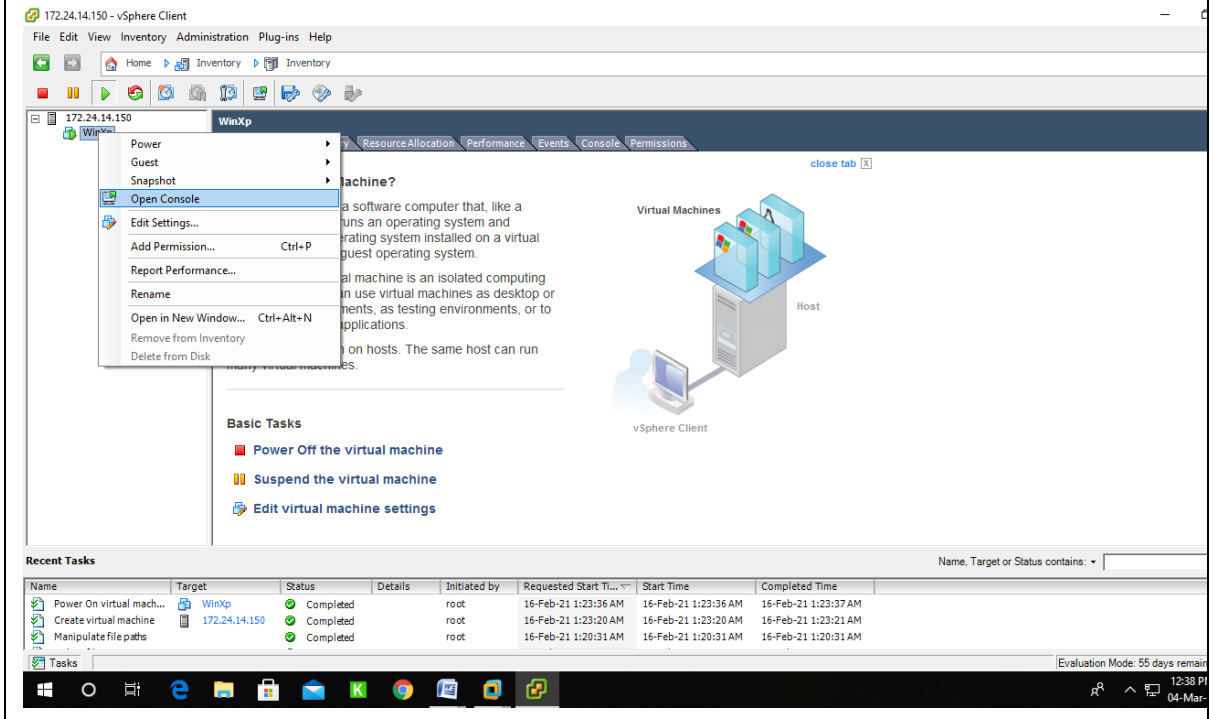
22) Once the path is provided, Check the ‘connect at power on’ and then finish.



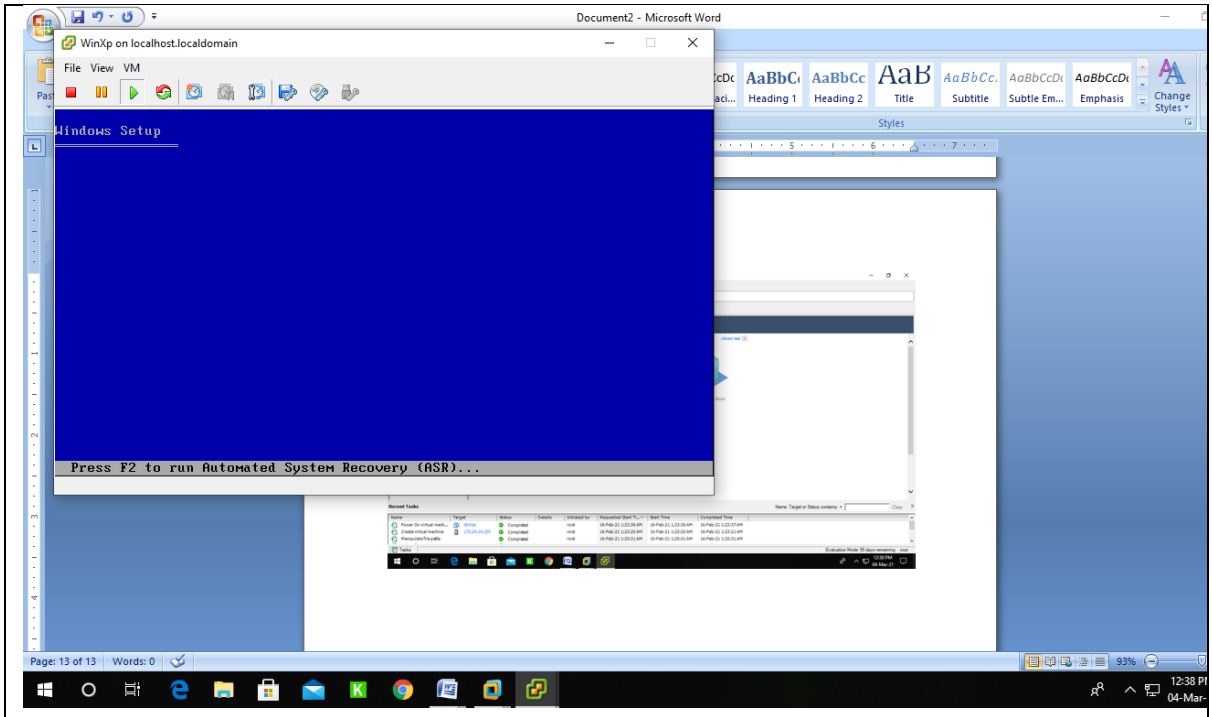
23) Right click on newly created VM. Use the option ‘power on’

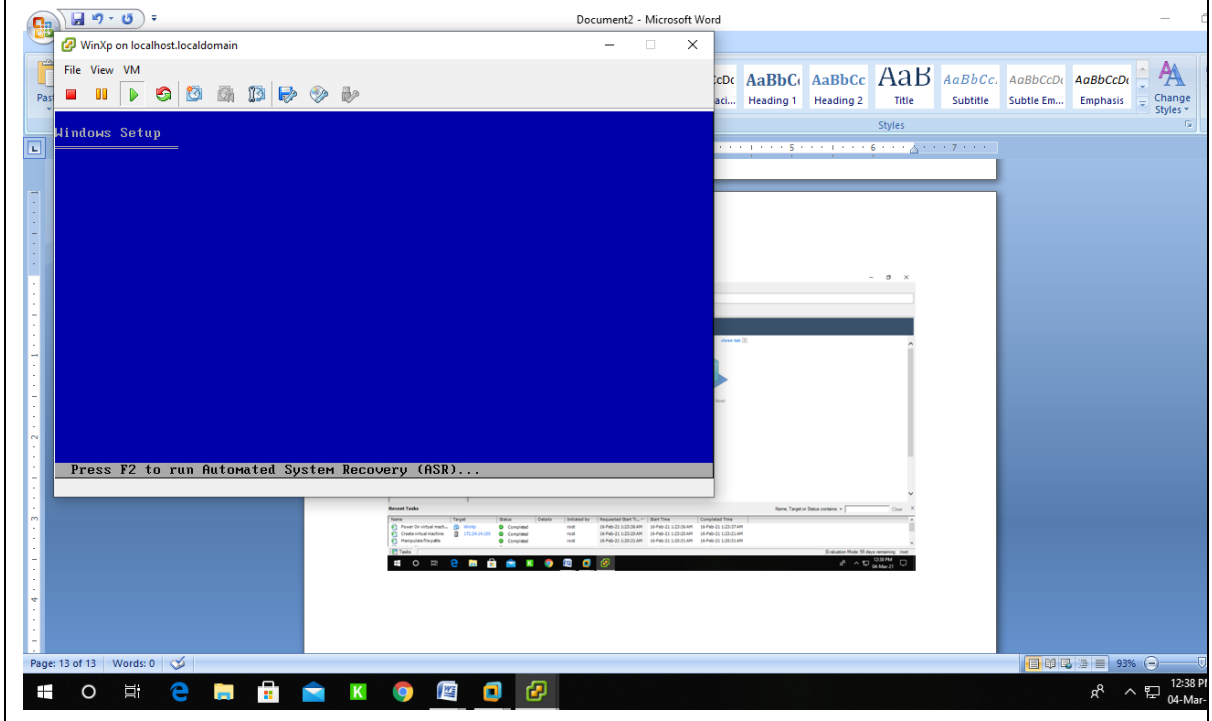


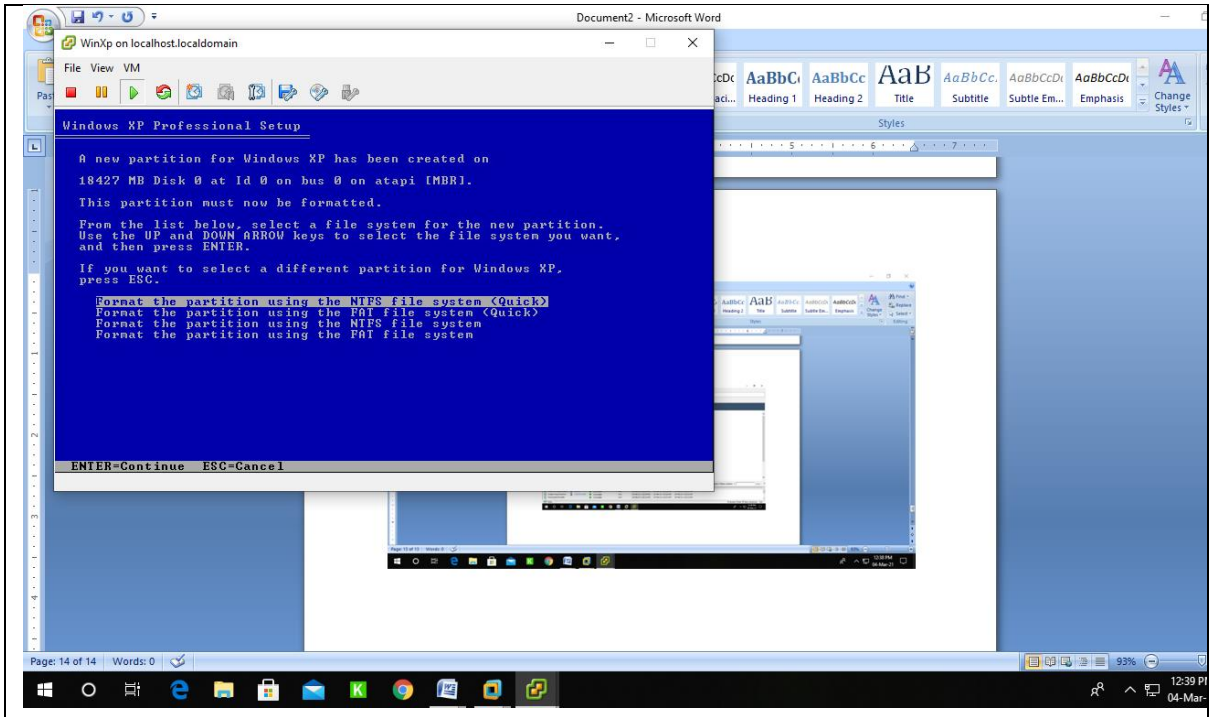
24) Right click on new VM and select ‘open console’

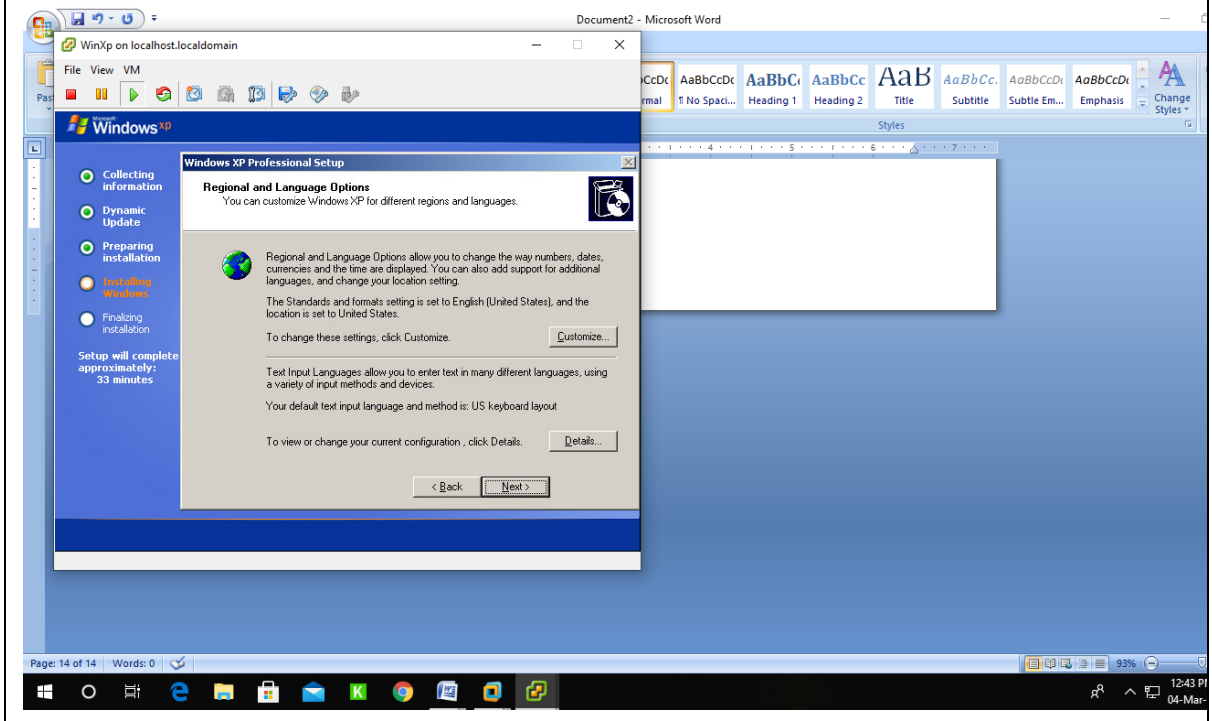


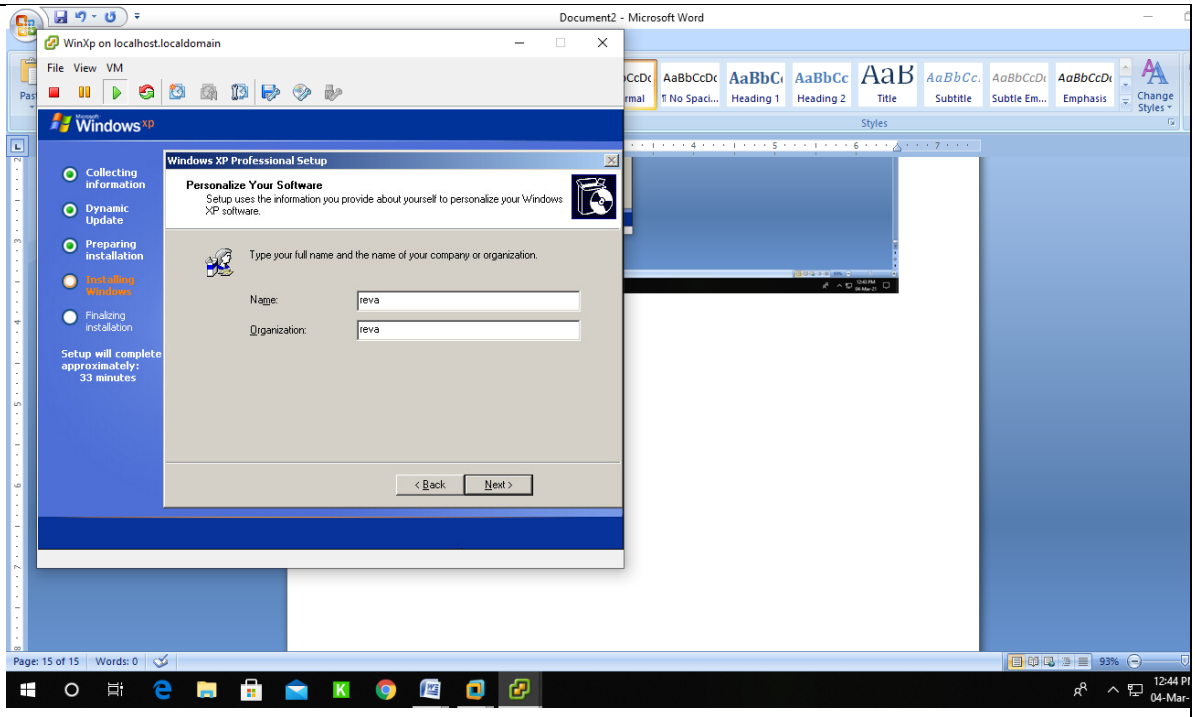
25) Follow the instructions displayed at booting time.

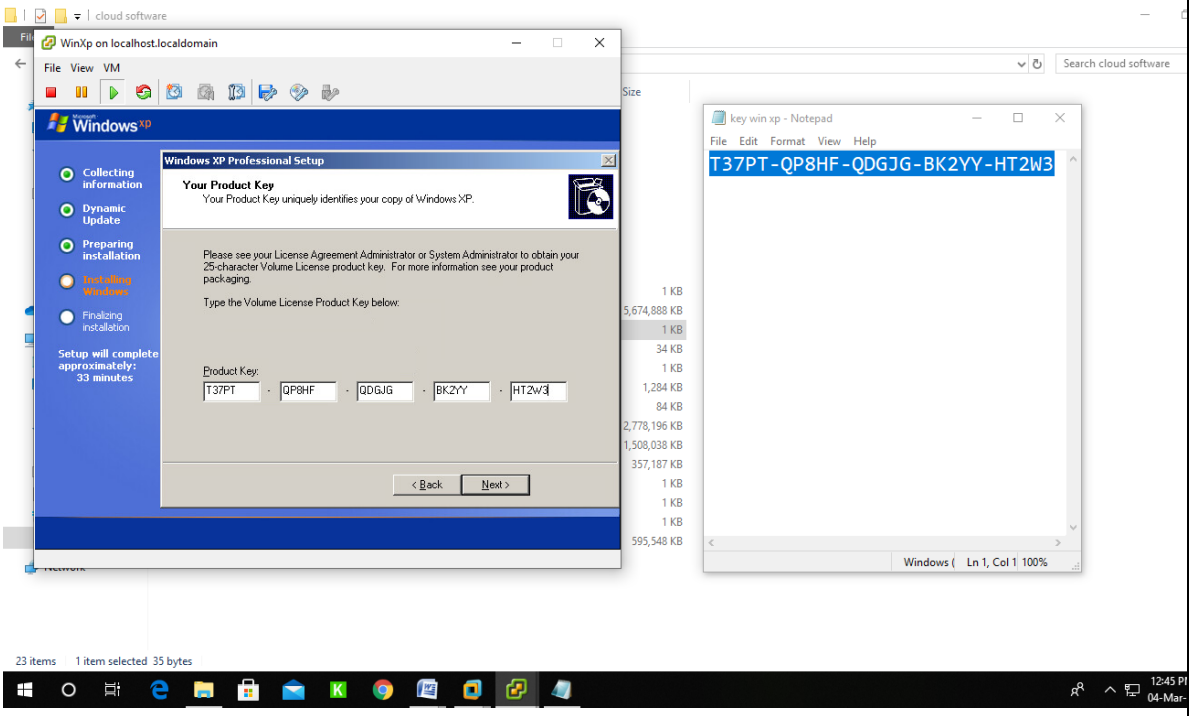

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows Setup
Press F2 to run Automated System Recovery (ASR)...
Page: 13 of 13
Words: 0

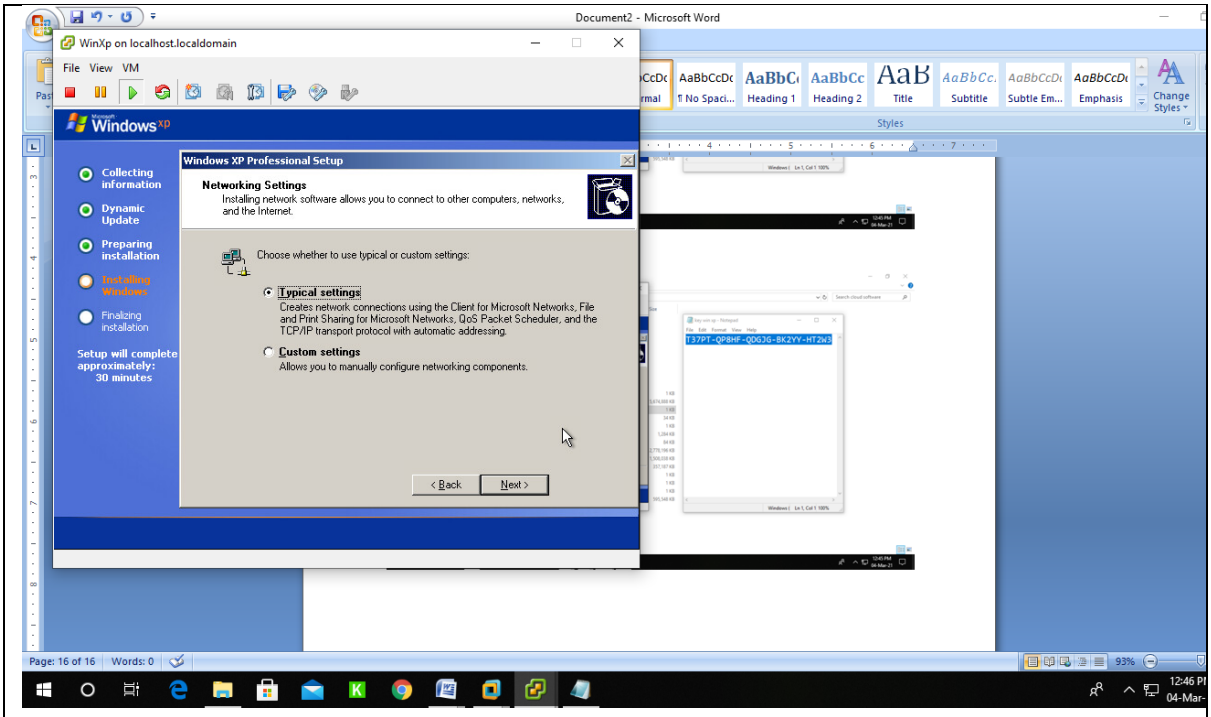

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows Setup
Press F2 to run Automated System Recovery (ASR)...
Page: 13 of 13
Words: 0

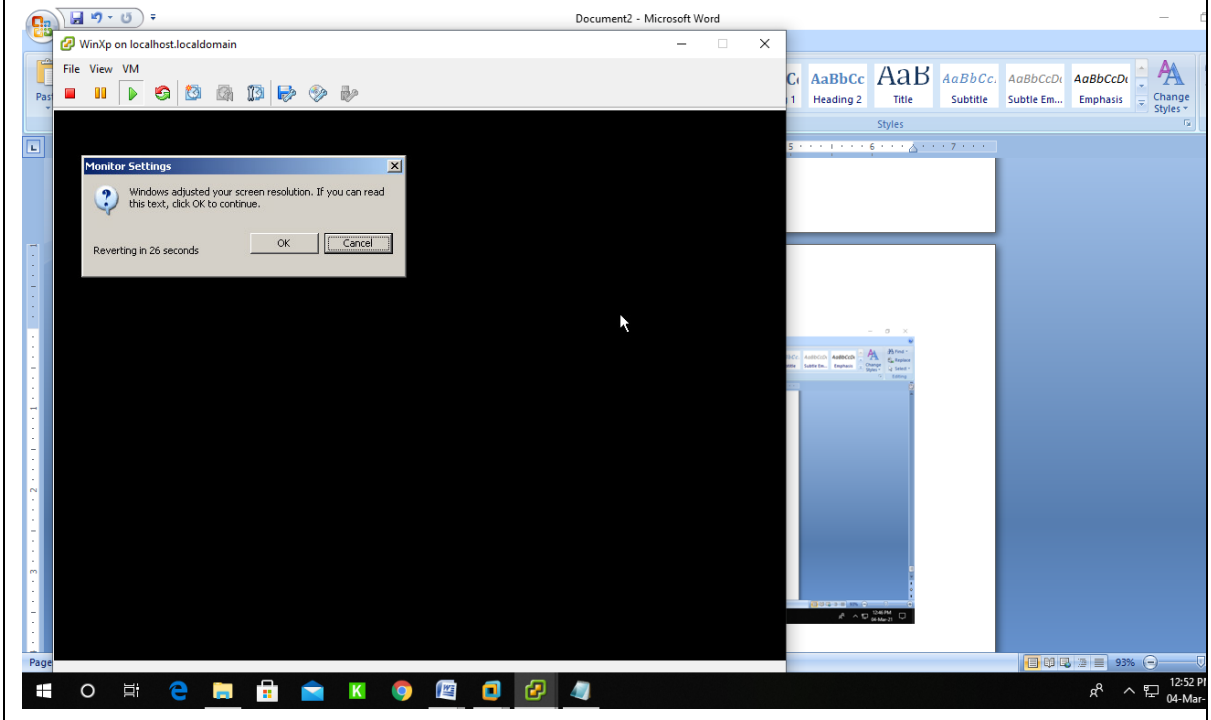

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows XP Professional Setup
A new partition for Windows XP has been created on
18427 MB Disk 0 at Id 0 on bus 0 on atapi [MBR].
This partition must now be formatted.
From the list below, select a file system for the new partition.
Use the UP and DOWN ARROW keys to select the file system you want,
and then press ENTER.
If you want to select a different partition for Windows XP,
press ESC.
Format the partition using the NTFS file system (Quick)
Format the partition using the FAT file system (Quick)
Format the partition using the NTFS file system
Format the partition using the FAT file system
ENTER=Continue ESC=Cancel
Page: 14 of 14
Words: 0
93%

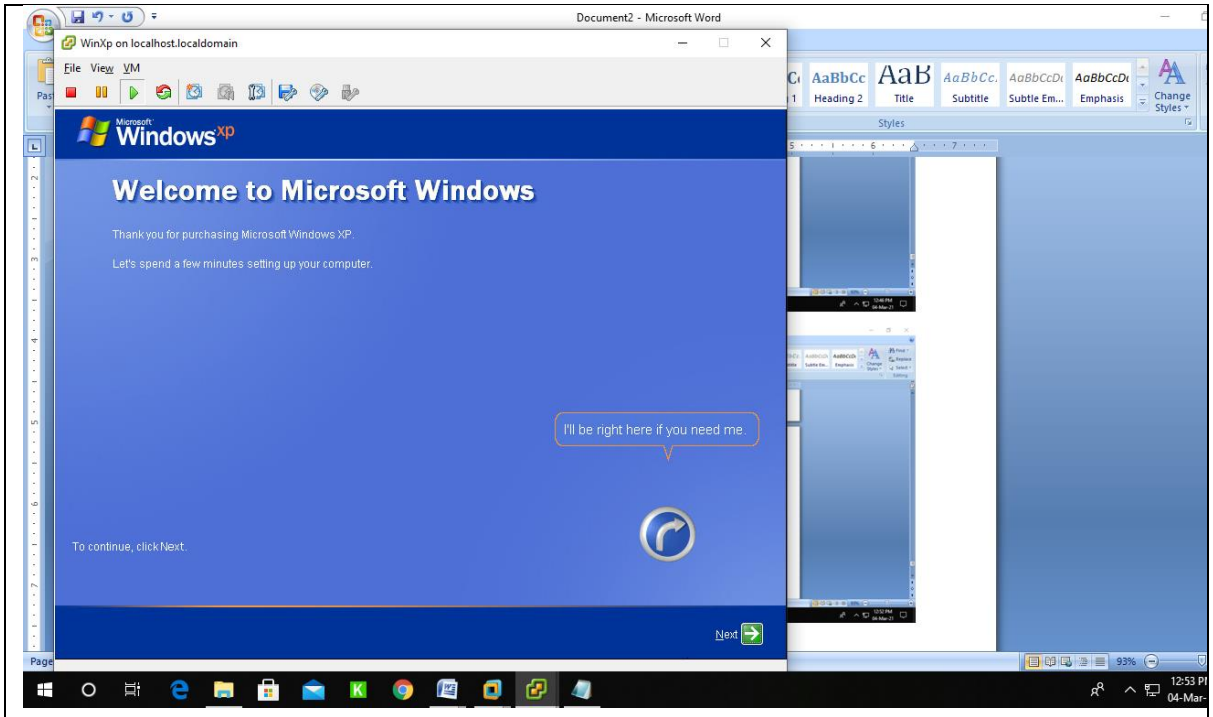

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows XP
Collecting information
Dynamic Update
Preparing installation
Installing Windows
Finalizing installation
Setup will complete approximately: 33 minutes
Windows XP Professional Setup
Regional and Language Options
You can customize Windows XP for different regions and languages.
Regional and Language Options allow you to change the way numbers, dates, currencies and the time are displayed. You can also add support for additional languages, and change your location setting.
The Standards and formats setting is set to English (United States), and the location is set to United States.
To change these settings, click Customize.
Customize...
Text Input Languages allow you to enter text in many different languages, using a variety of input methods and devices.
Your default text input language and method is: US keyboard layout
To view or change your current configuration , click Details.
Details...
< Back
Next >
Page: 14 of 14
Words: 0
93%

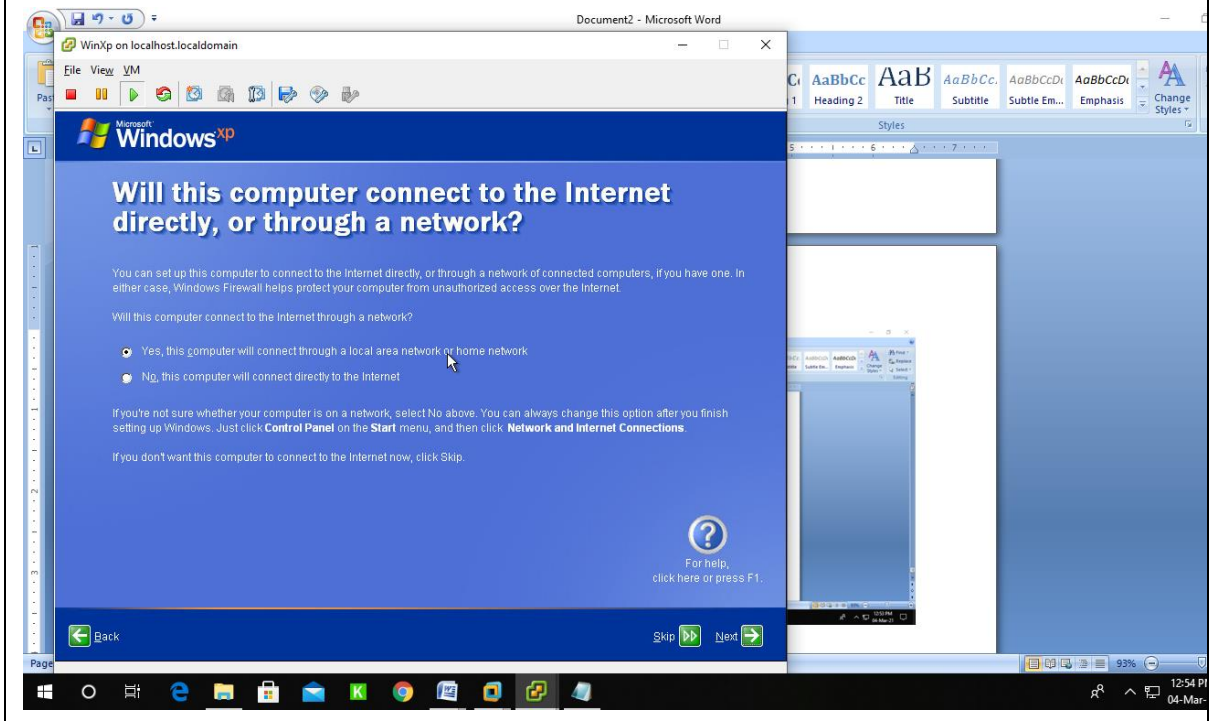

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows XP Professional Setup
Personalize Your Software
Setup uses the information you provide about yourself to personalize your Windows XP software.
Type your full name and the name of your company or organization.
Name:
reva
Organization:
reva
< Back
Next >
Collecting information
Dynamic Update
Preparing installation
Installing Windows
Finalizing installation
Setup will complete approximately: 33 minutes
Page: 15 of 15
Words: 0
93%
12:44 PM
04-Mar-

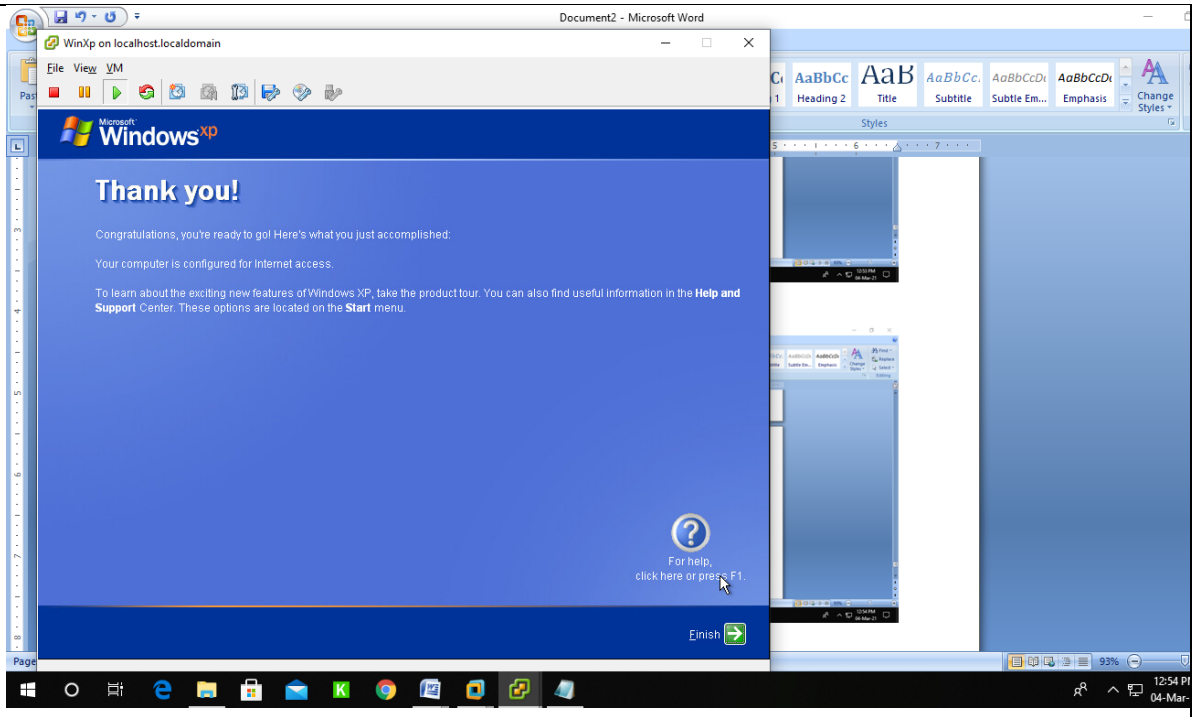

cloud software
WinXp on localhost.localdomain
File View VM
Windows XP Professional Setup
Your Product Key
Your Product Key uniquely identifies your copy of Windows XP.
Please see your License Agreement Administrator or System Administrator to obtain your 25-character Volume License product key. For more information see your product packaging.
Type the Volume License Product Key below:
Product Key:
T37PT
QP8HF
QDGJG
BK2YY
HT2W3
< Back
Next >
key win xp - Notepad
File Edit Format View Help
T37PT-QP8HF-QDGJG-BK2YY-HT2W3
Windows (Ln 1, Col 1 100%
Search cloud software
23 items
1 item selected 35 bytes
12:45 PM
04-Mar-


Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows XP
Windows XP Professional Setup
Networking Settings
Installing network software allows you to connect to other computers, networks, and the Internet.
Collecting information
Dynamic Update
Preparing installation
Installing Windows
Finalizing installation
Setup will complete approximately: 30 minutes
Choose whether to use typical or custom settings:
Typical settings
Creates network connections using the Client for Microsoft Networks, File and Print Sharing for Microsoft Networks, QoS Packet Scheduler, and the TCP/IP transport protocol with automatic addressing.
Custom settings
Allows you to manually configure networking components.
< Back
Next >
Page: 16 of 16
Words: 0
93%

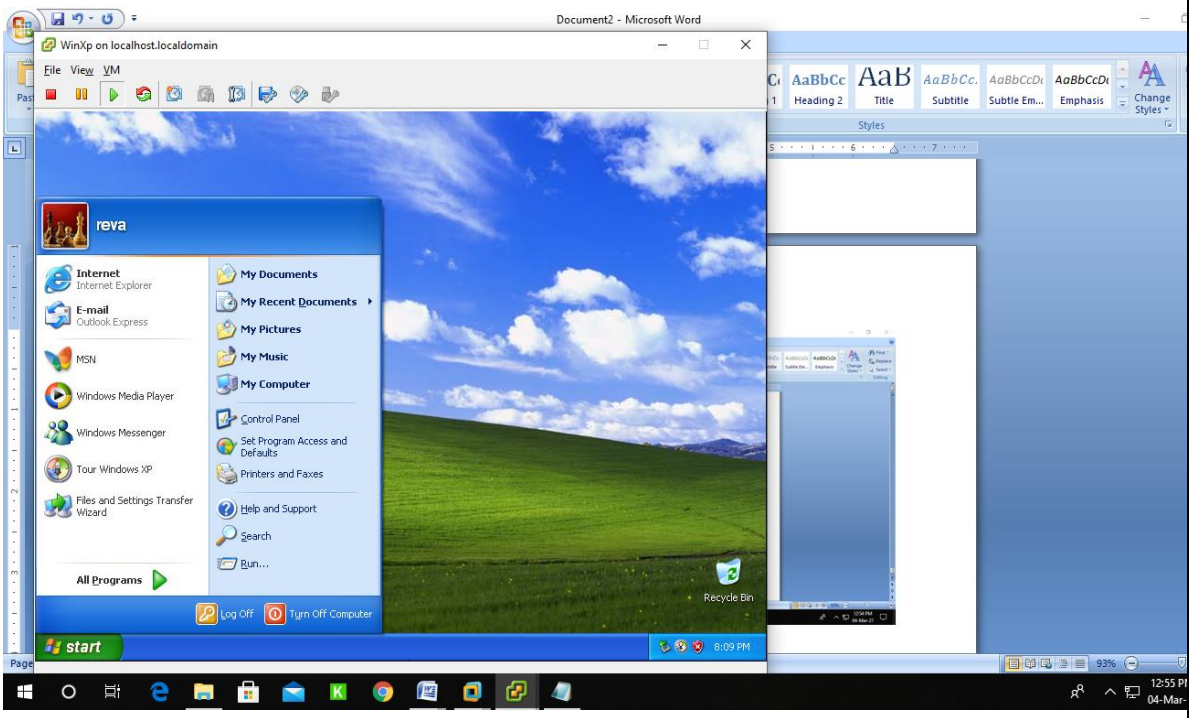

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Monitor Settings
Windows adjusted your screen resolution. If you can read this text, click OK to continue.
Reverting in 26 seconds
OK
Cancel
93%


Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows xp
Welcome to Microsoft Windows
Thank you for purchasing Microsoft Windows XP.
Let's spend a few minutes setting up your computer.
I'll be right here if you need me.
To continue, click Next.
Next

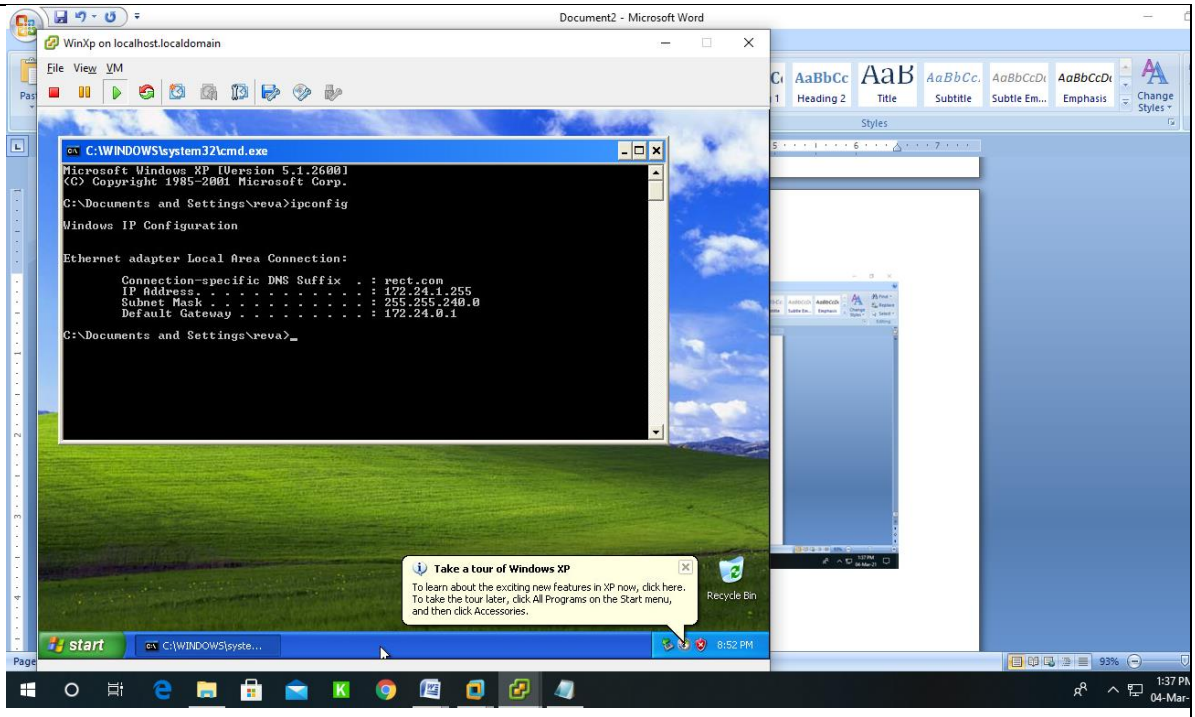

Document2 - Microsoft Word
WinXp on localhost.localdomain
File View VM
Windows xp
Will this computer connect to the Internet directly, or through a network?
You can set up this computer to connect to the Internet directly, or through a network of connected computers, if you have one. In either case, Windows Firewall helps protect your computer from unauthorized access over the Internet.
Will this computer connect to the Internet through a network?
Yes, this computer will connect through a local area network or home network
No, this computer will connect directly to the Internet
If you're not sure whether your computer is on a network, select No above. You can always change this option after you finish setting up Windows. Just click Control Panel on the Start menu, and then click Network and Internet Connections.
If you don't want this computer to connect to the Internet now, click Skip.
For help, click here or press F1.
Back
Skip
Next



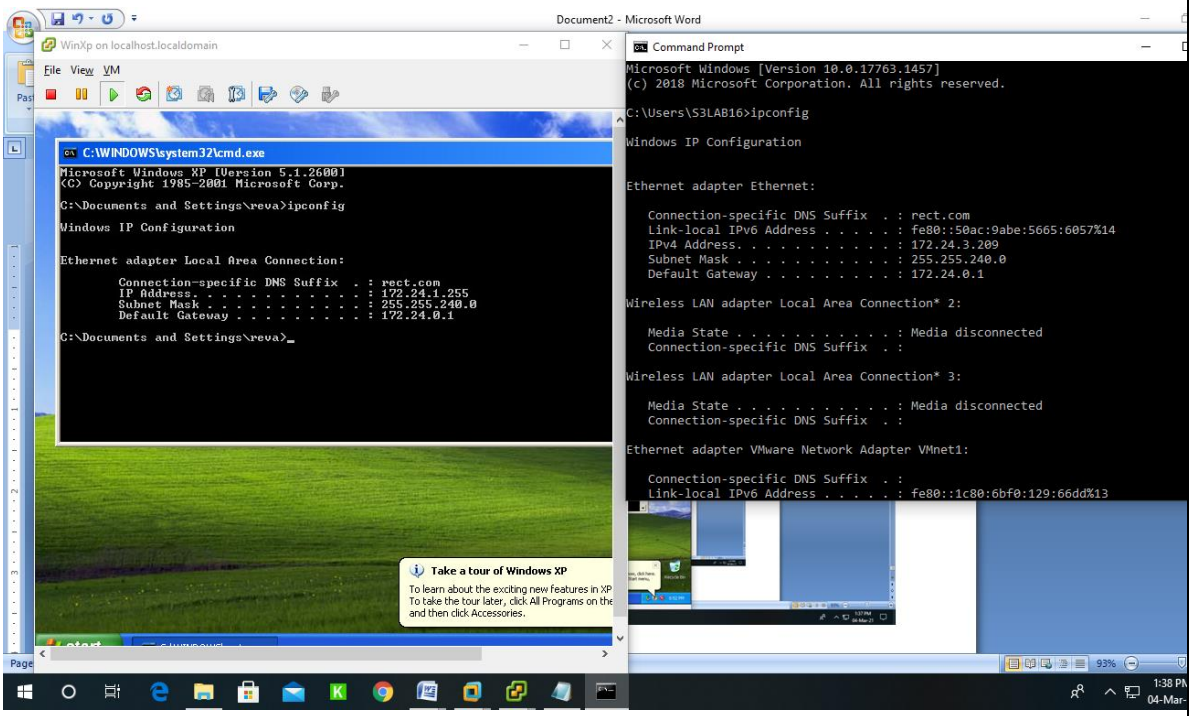
26) Once the VM is created completely, then goto command prompt.



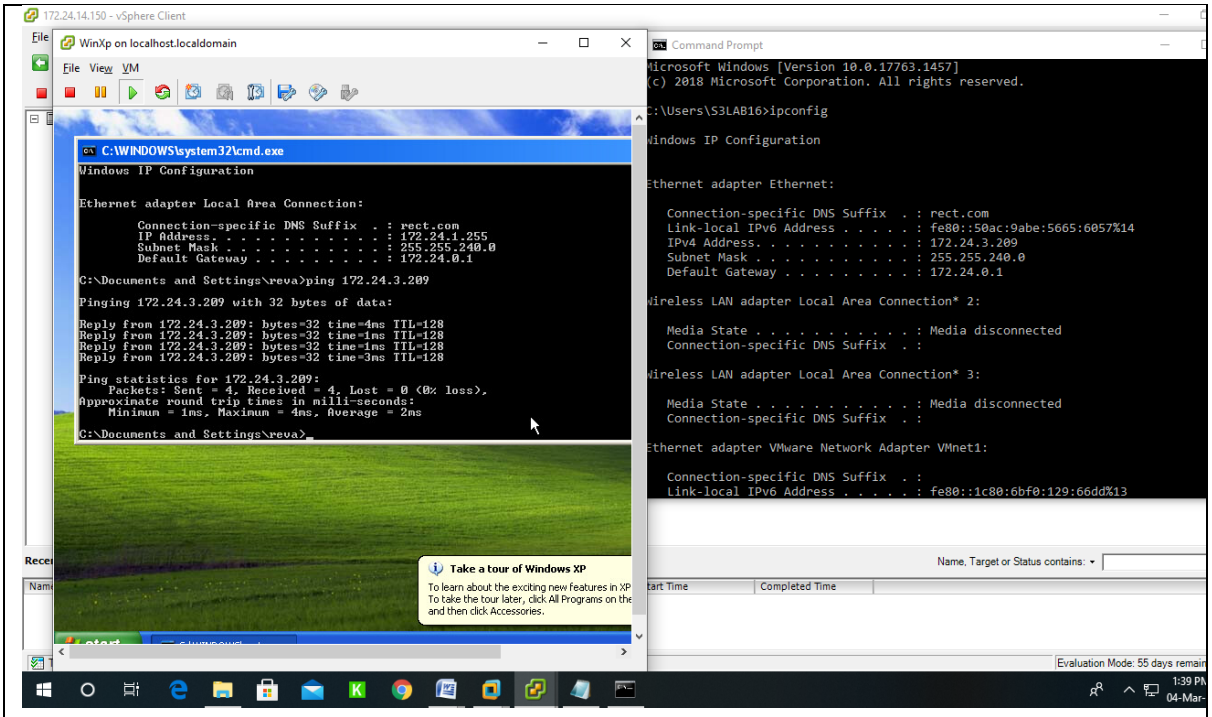
27) use the command ‘ipconfig’ to find its address



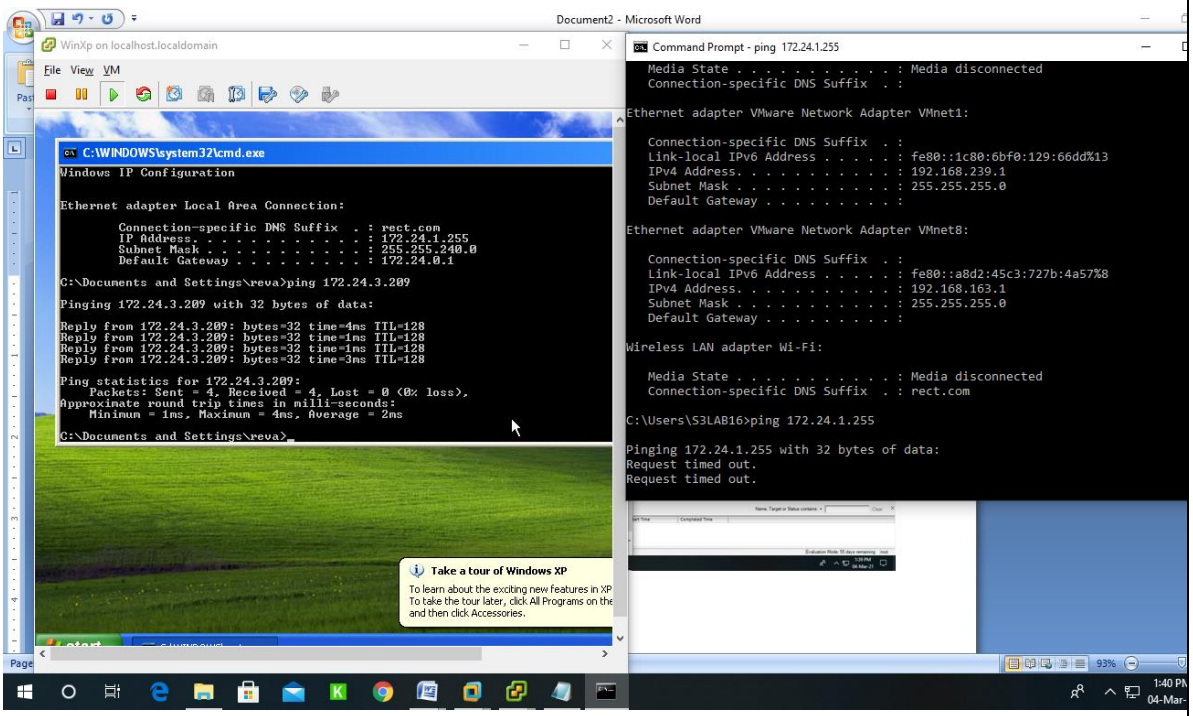
28) Find the IP address of Physical machine also.



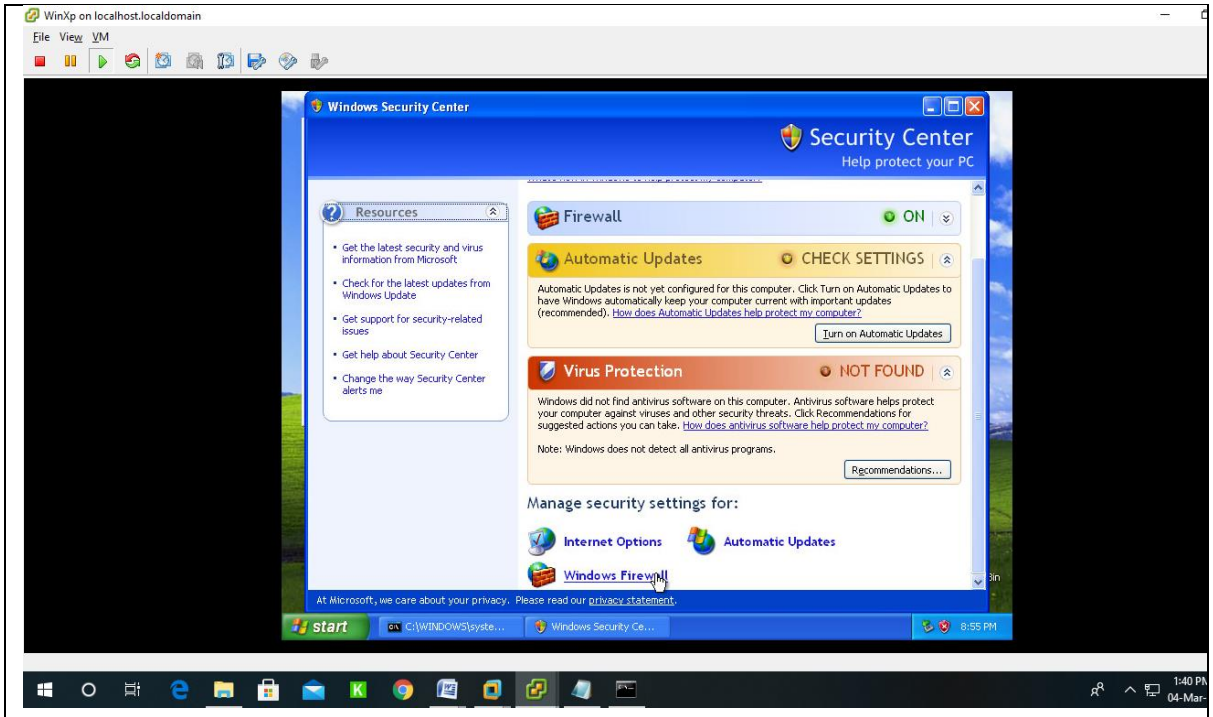
29) Ping from VM to PM

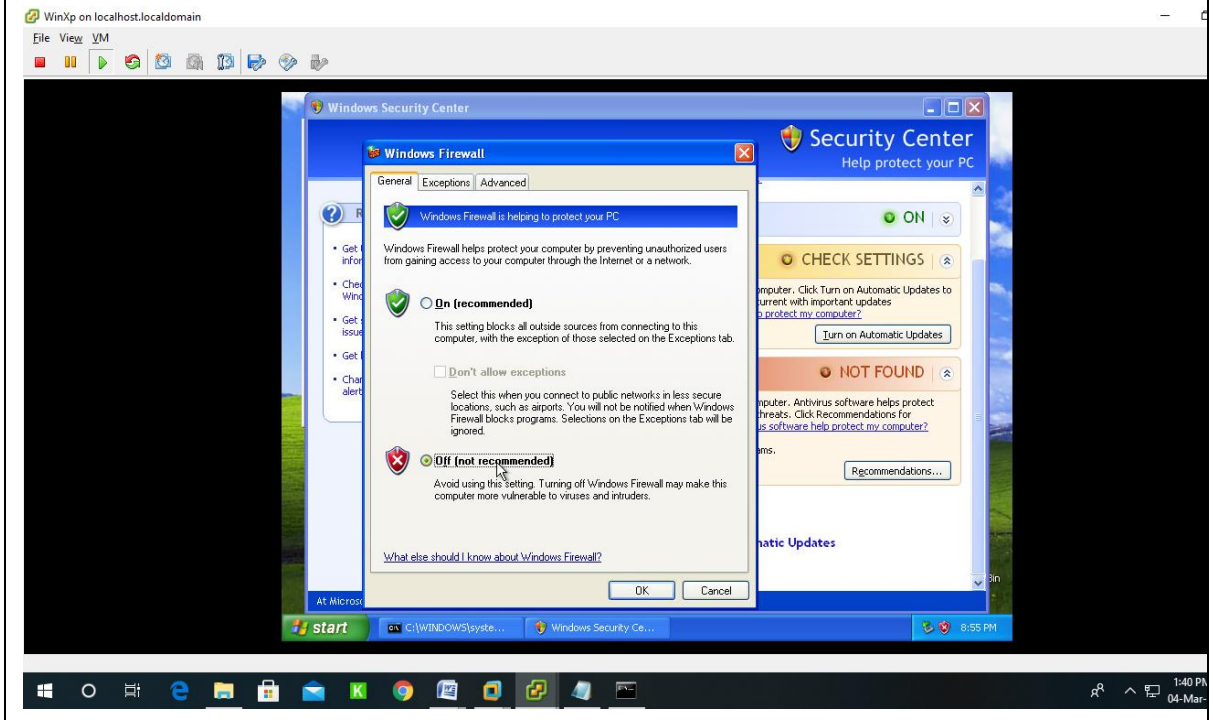


30) Ping from PM to VM may throw an error, due to firewall block at VM.

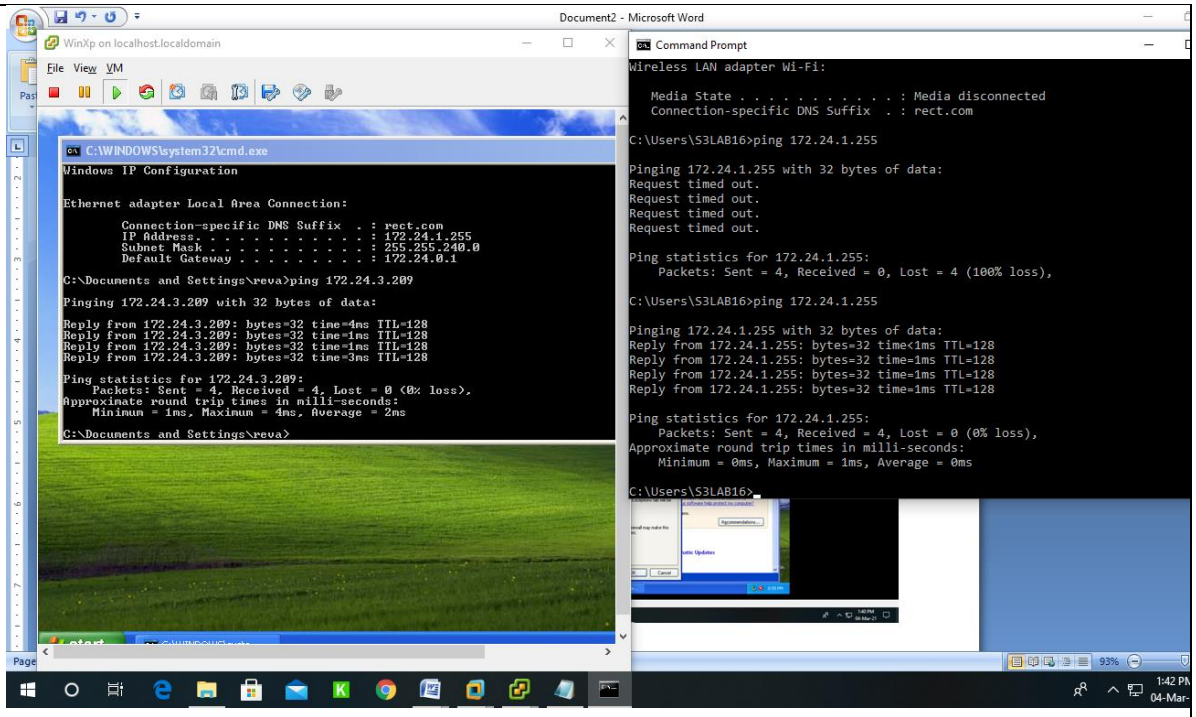


31) Clear the firewall at VM. So that it can allow the request from other machine.

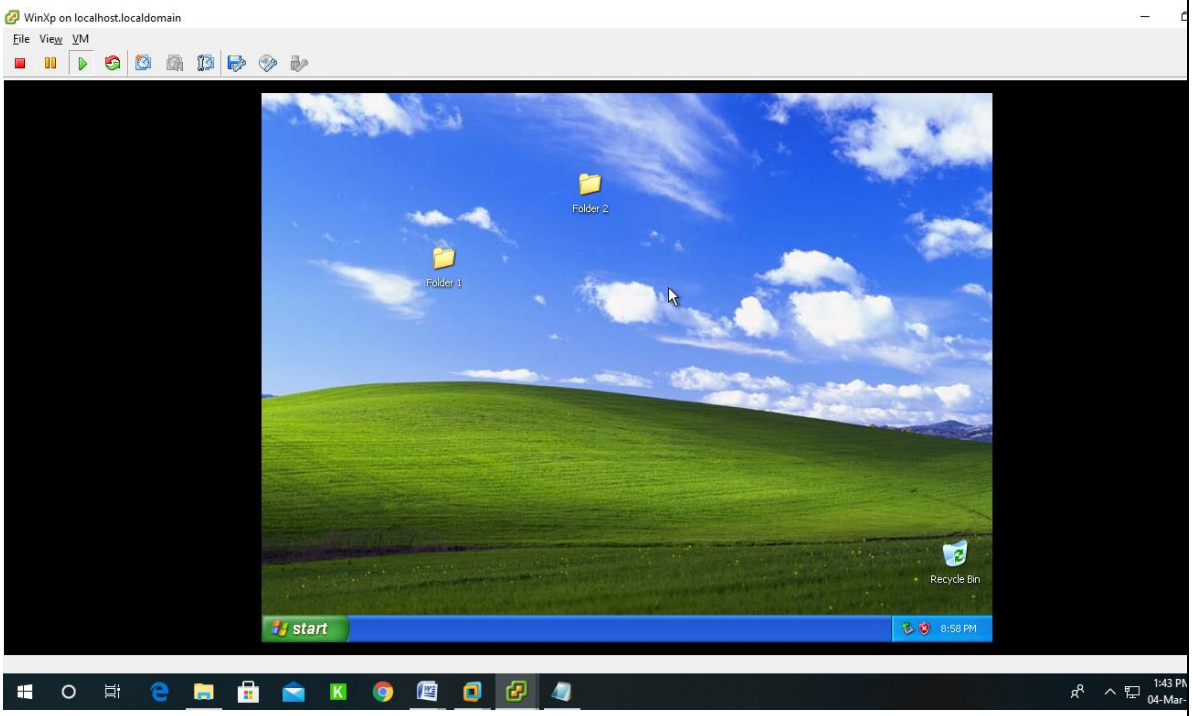




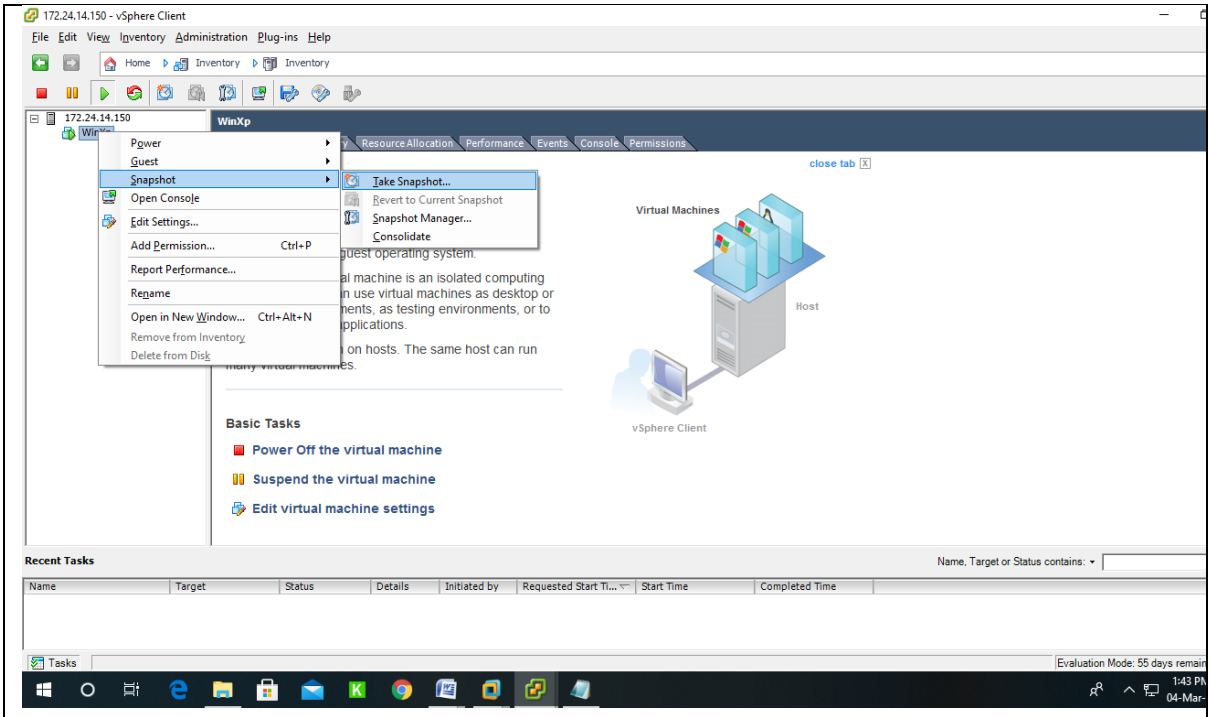
32) Now ping from PM to VM. And its success.



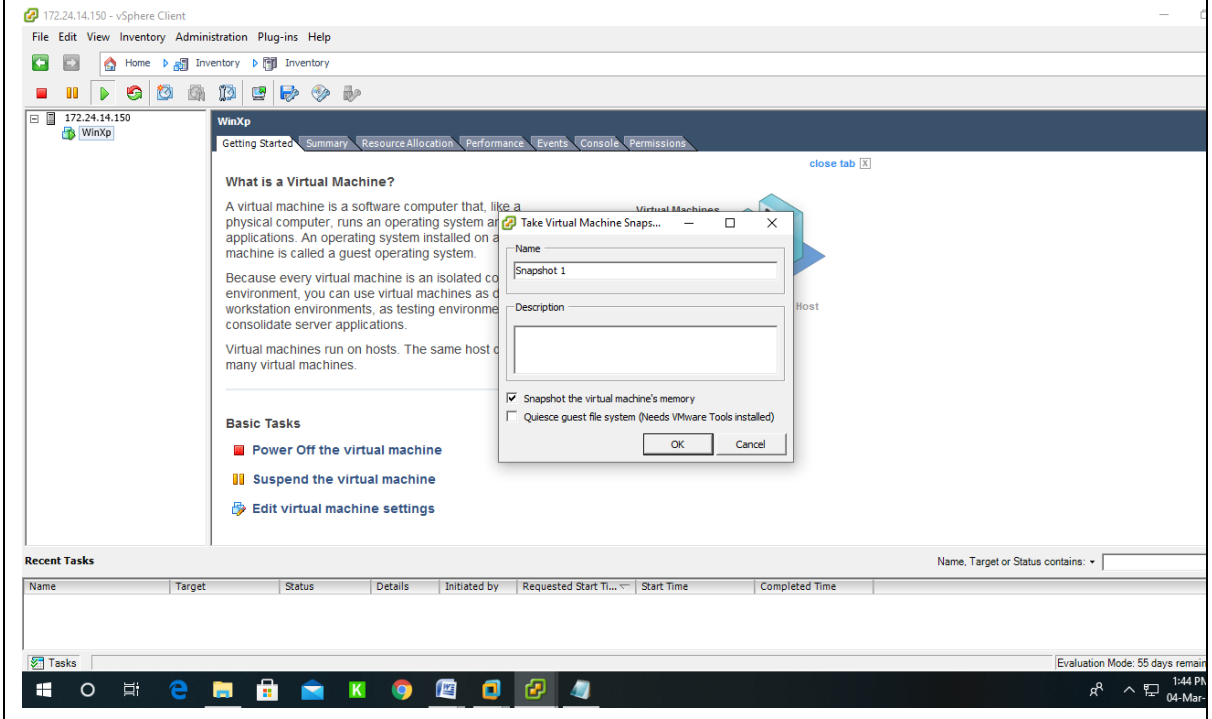
33) To demonstrate about the Snapshot. Create any data files in VM.



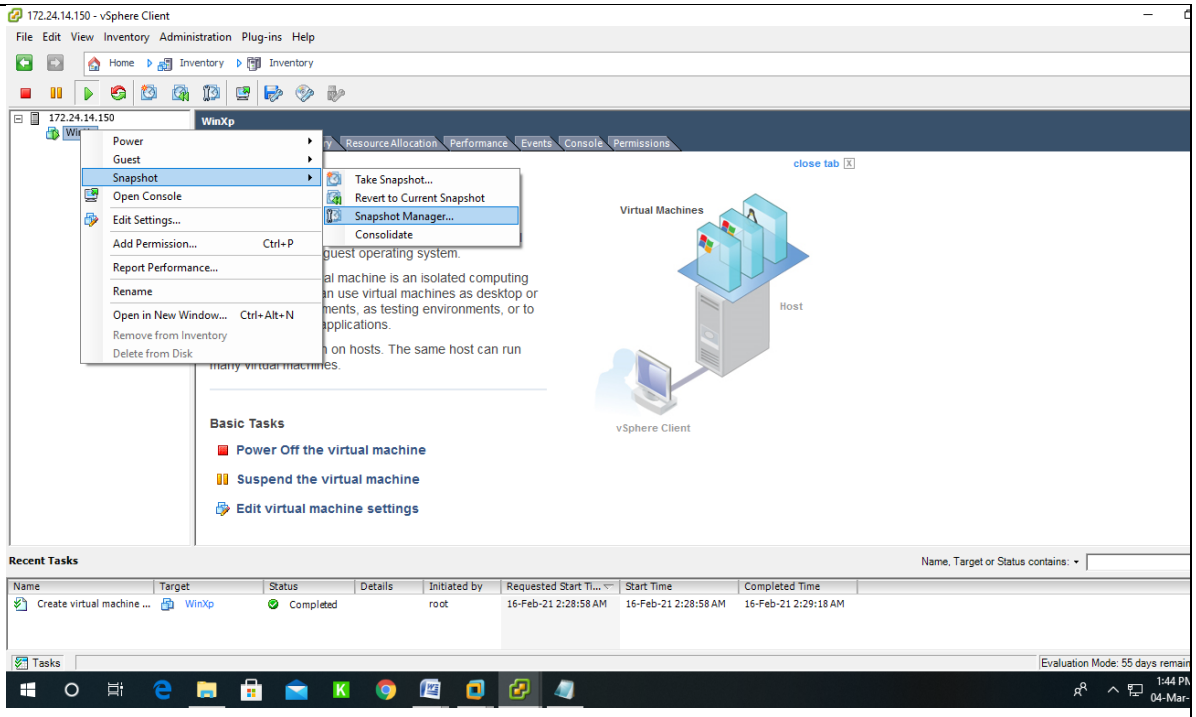
34) Right click on VM, use the option, Snapshot, Take Snapshot.

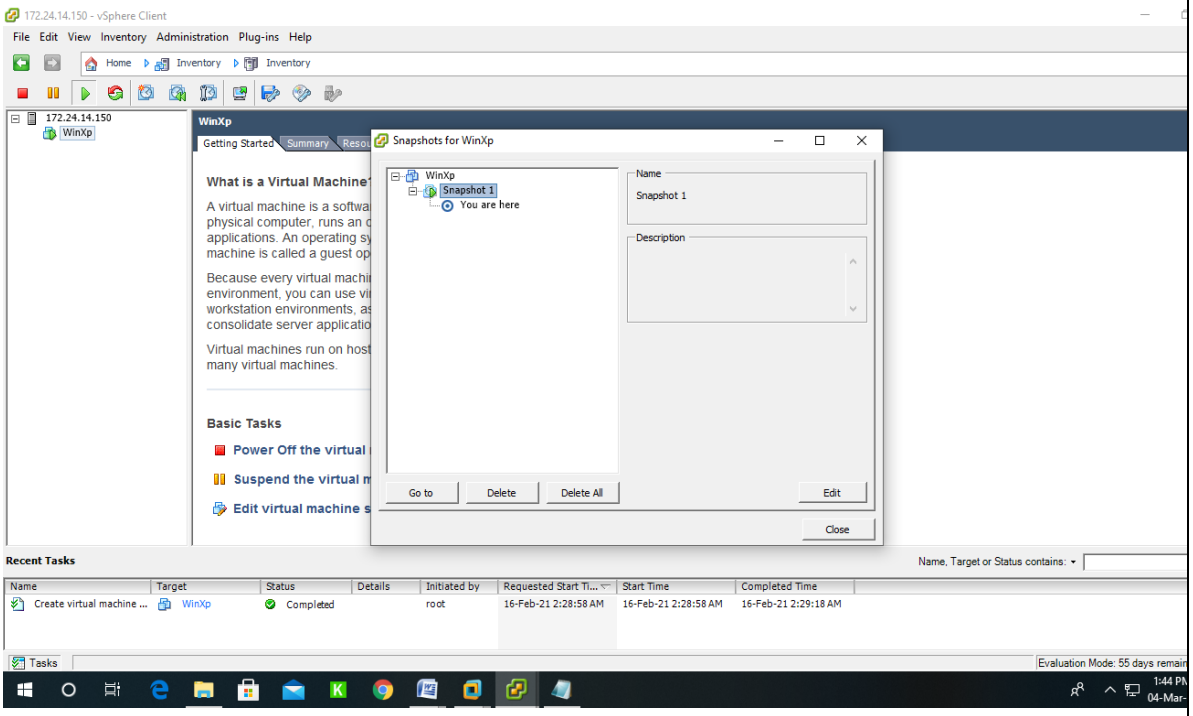


35) Give the name of the snapshot to be stored.

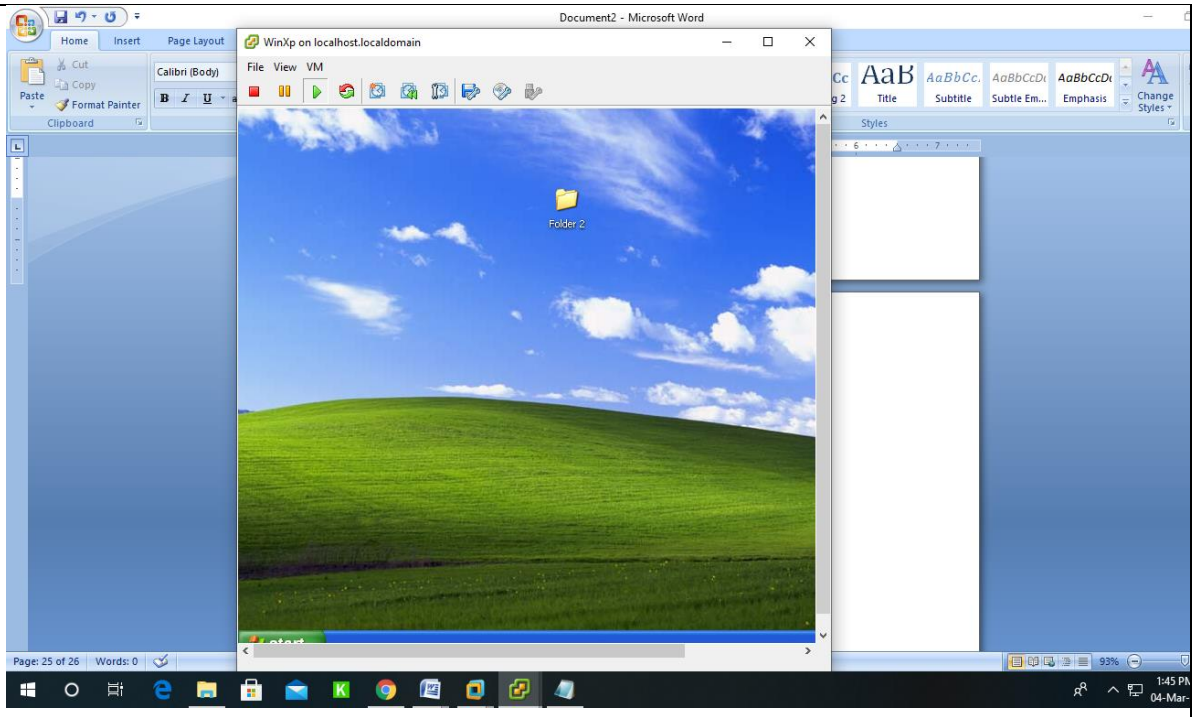


36) All the stored snapshots will be available in Snapshot Manager.

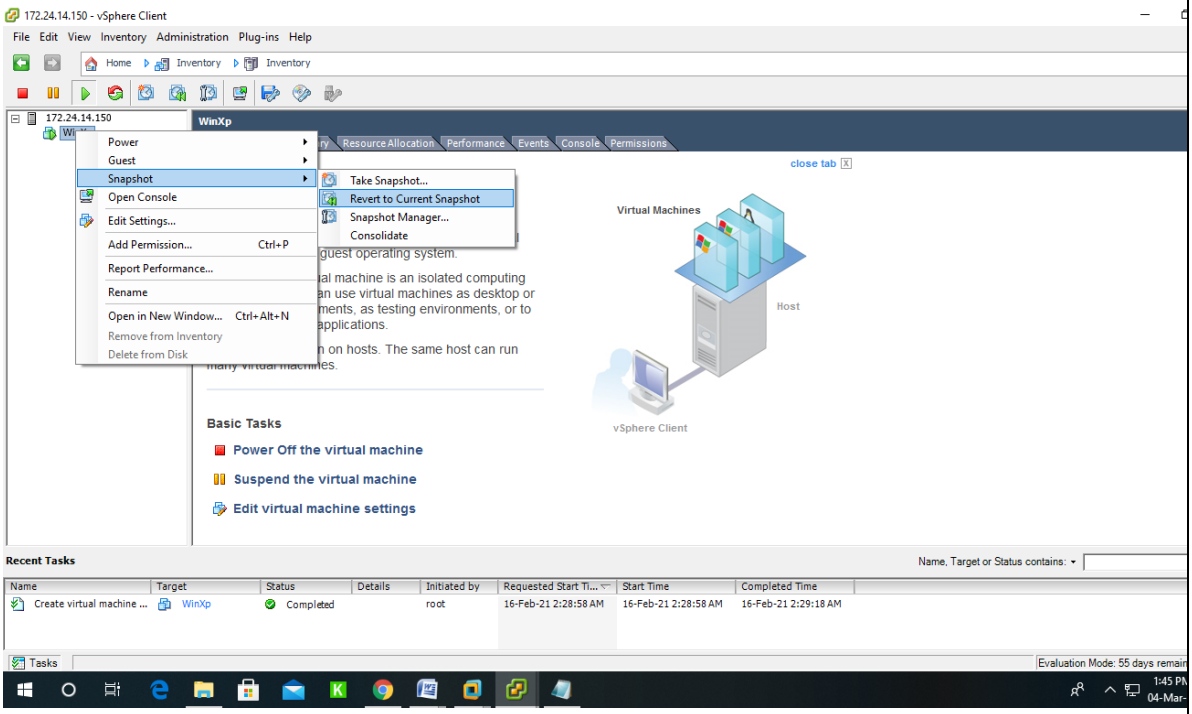


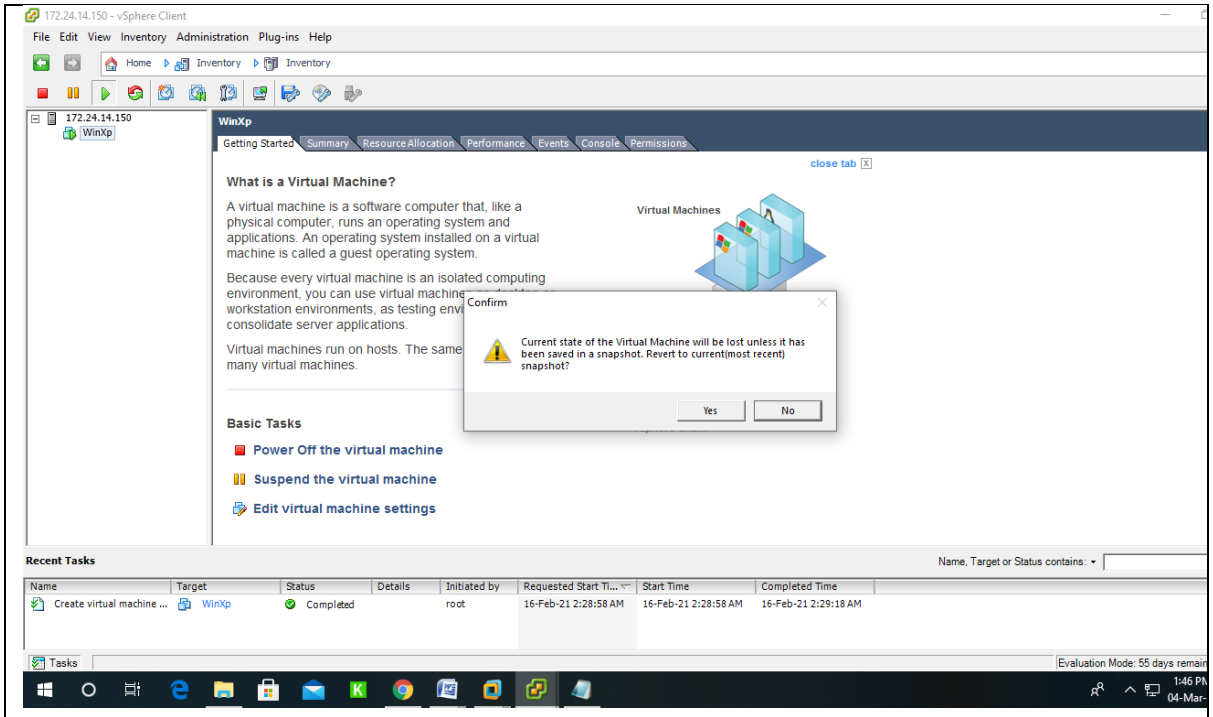


37) If any data is deleted by any failure. It can be retrieved back.

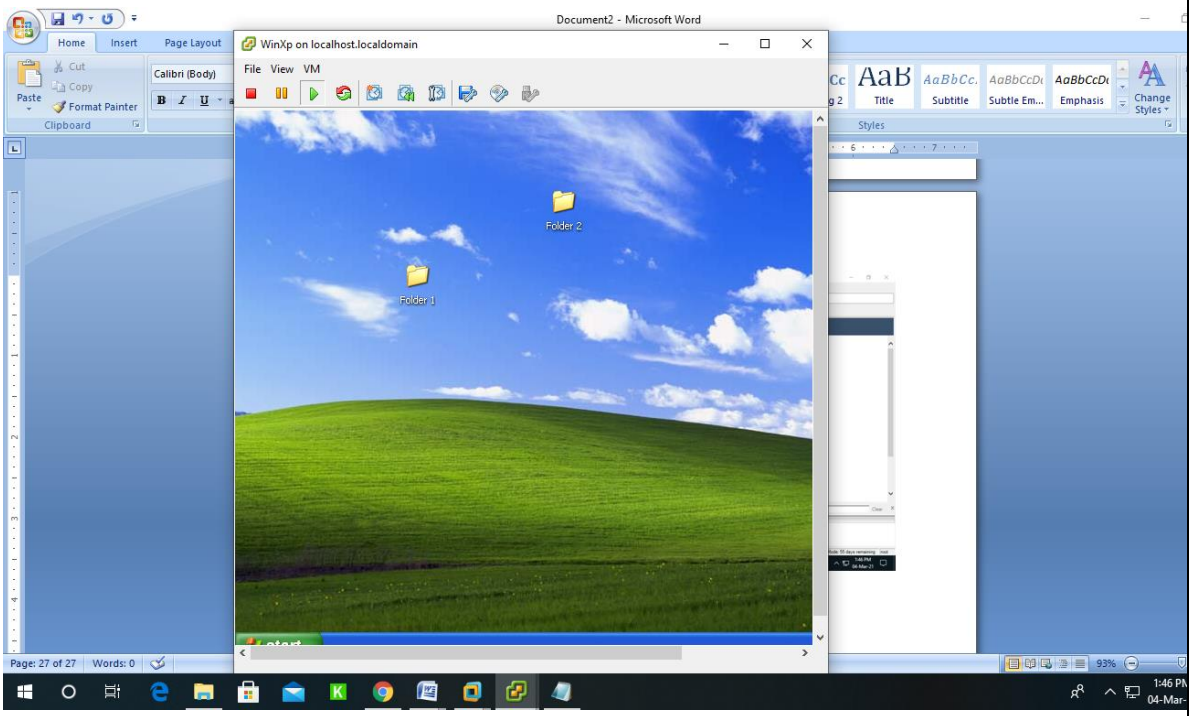


38) Use the option, Revert to Current Snapshot for restoring.





39) The deleted data is restored.



Session 2

1	**Problem Statement:**

Demonstrate the mechanism of cloning and create a switch with multiple networks having the different VM's.

2	**Student Learning Outcomes:**

To configure and maintain the VM's.

To configure a multiple VM's on multiple ESXi's

To provide the adapter for the already created VM's

3	**Theoretical Description:**

Virtual Machine (**VM**): is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.
There are different kinds of virtual machines, each with different functions:

When you take a snapshot, you capture the state of the virtual machine settings and the virtual disk. If you are taking a memory snapshot, you also capture the memory state of the virtual machine. These states are saved to files that reside with the virtual machine's base files.

A snapshot consists of files that are stored on a supported storage device. A Take Snapshot operation creates .vmdk, -delta.vmdk, .vmsd, and .vmsn files. By default, the first and all delta disks are stored with the base .vmdk file. The .vmsd and .vmsn files are stored in the virtual machine directory.

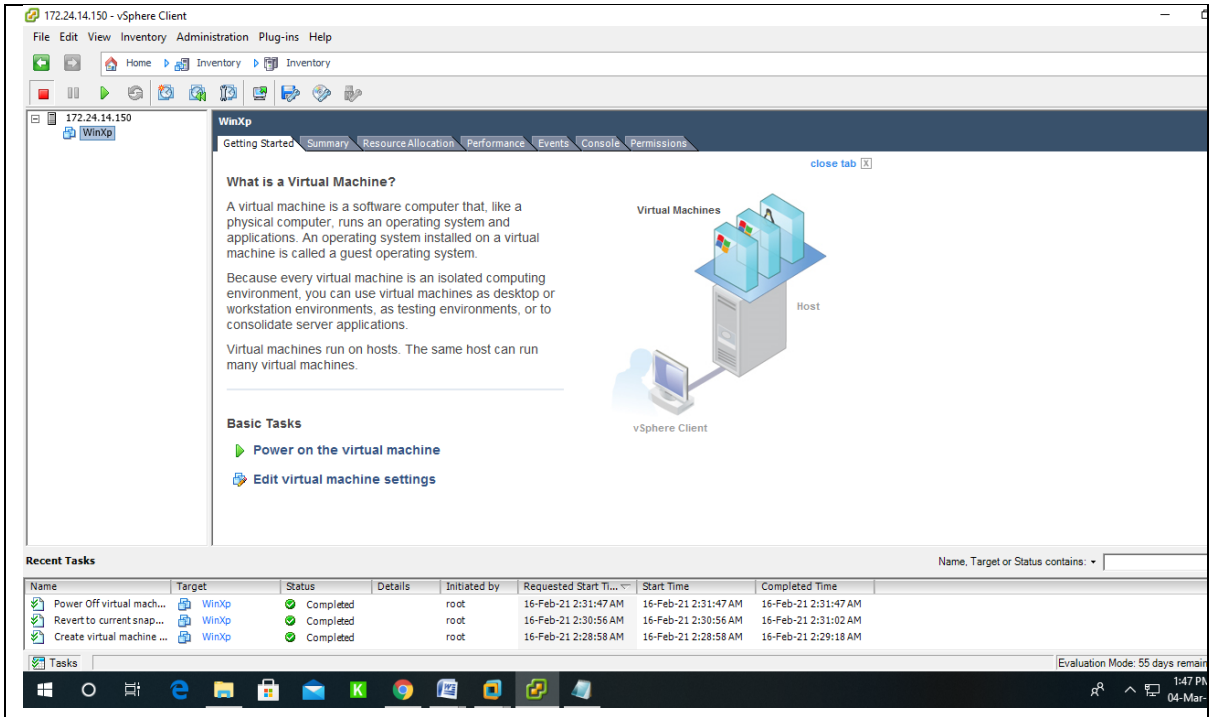
4	**Requirements**

The following list of requirements as a starting point. Like physical computers, the virtual machines running under VMware Workstation generally perform better if they have faster processors and more memory.

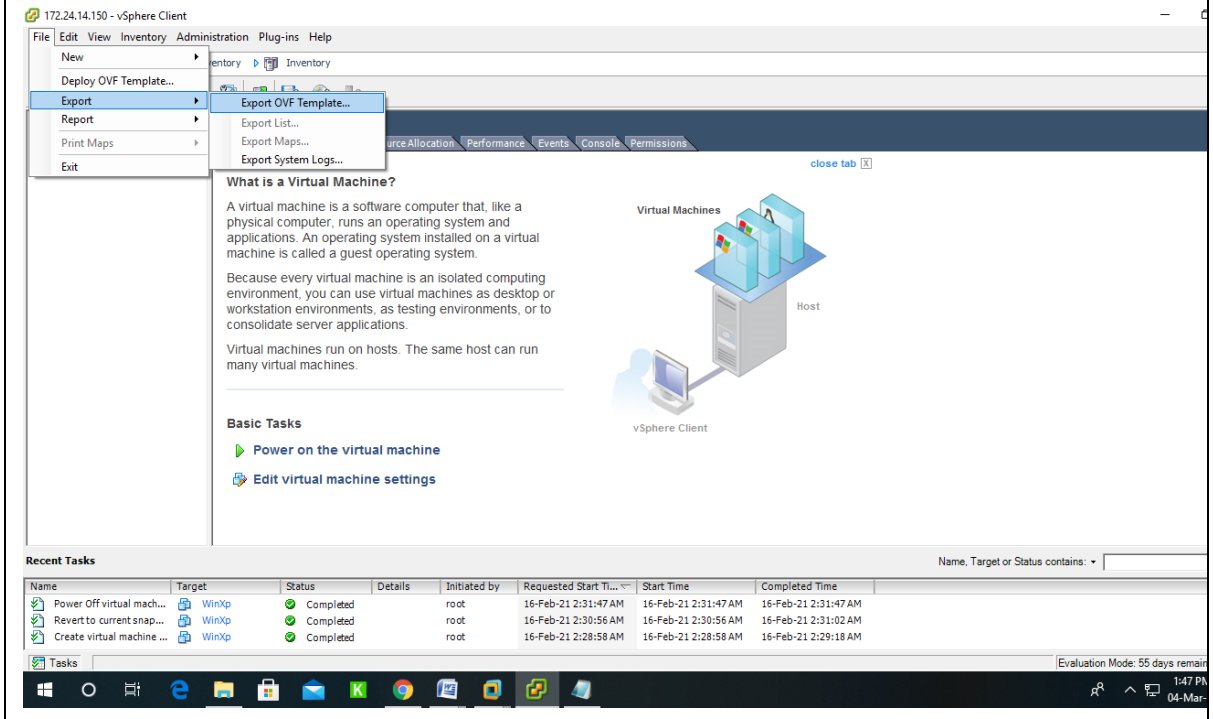
- PC Hardware - Standard x86-compatible personal computer, 400 MHz or faster CPU minimum (500 MHz recommended), Multiprocessor systems supported, 64-bit processor support for AMD64 Opteron, Athlon 64 and Intel IA-32e CPU (including "Nocona").
- RAM - 8GB minimum
- Disk Drives - IDE and SCSI hard drives supported, up to 950GB capacity, At least 1GB free disk space recommended for each guest operating system and the application software used with it; if you use a default setup, the actual disk space needs are approximately the same as those for installing and running the guest operating system and applications on a physical computer.
- Local Area Networking (Optional)
- Host Operating System

5	**Procedure**

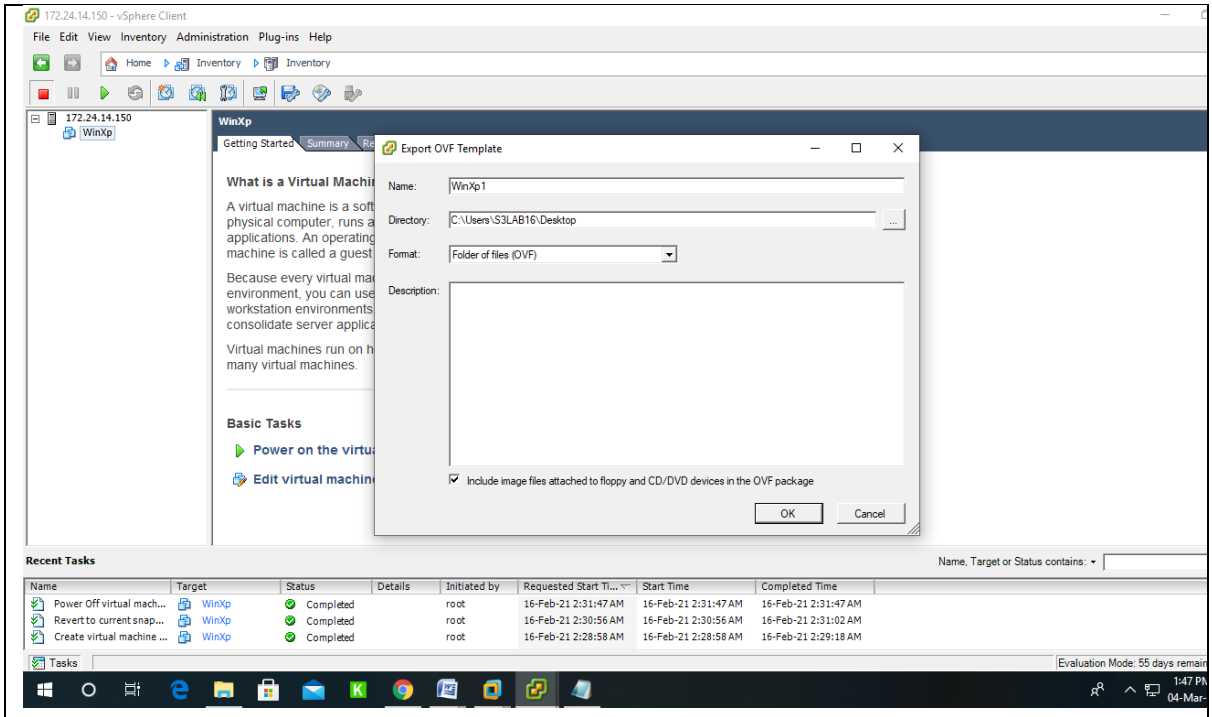
1) Select the VM to be cloned.



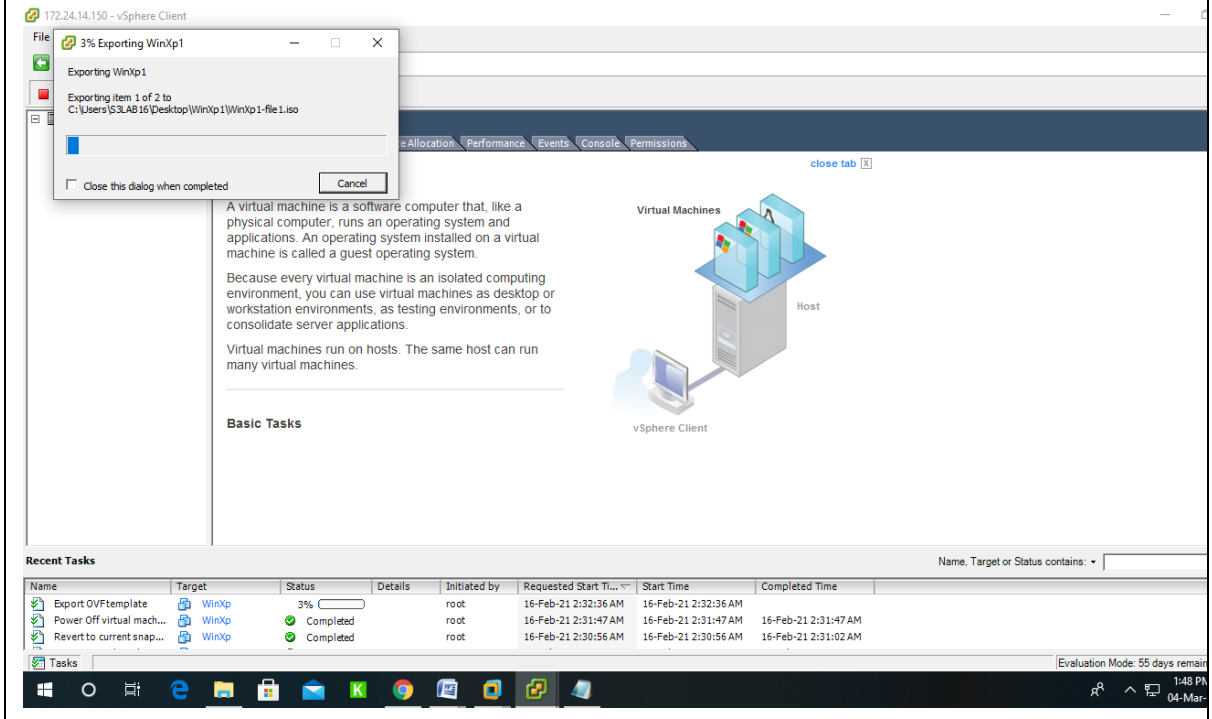
2) Goto file->Export->Export OVF template.



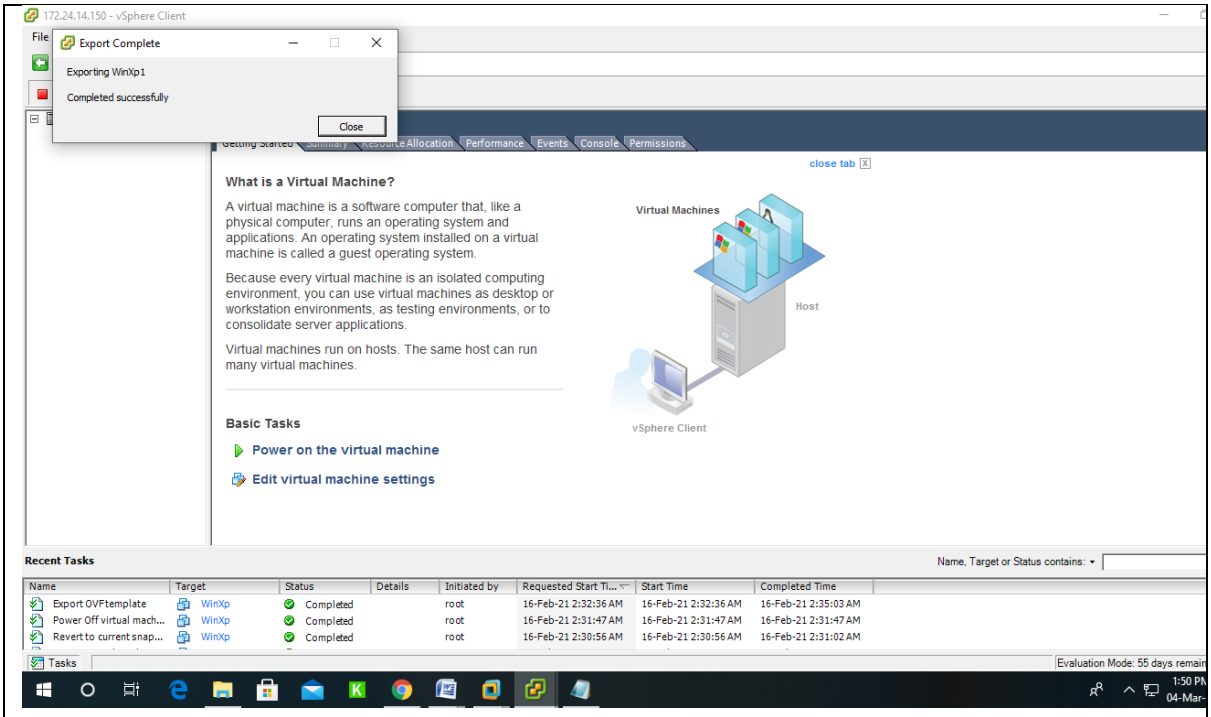
3) Provide the location in physical machine, where the copy of VM to be stored.



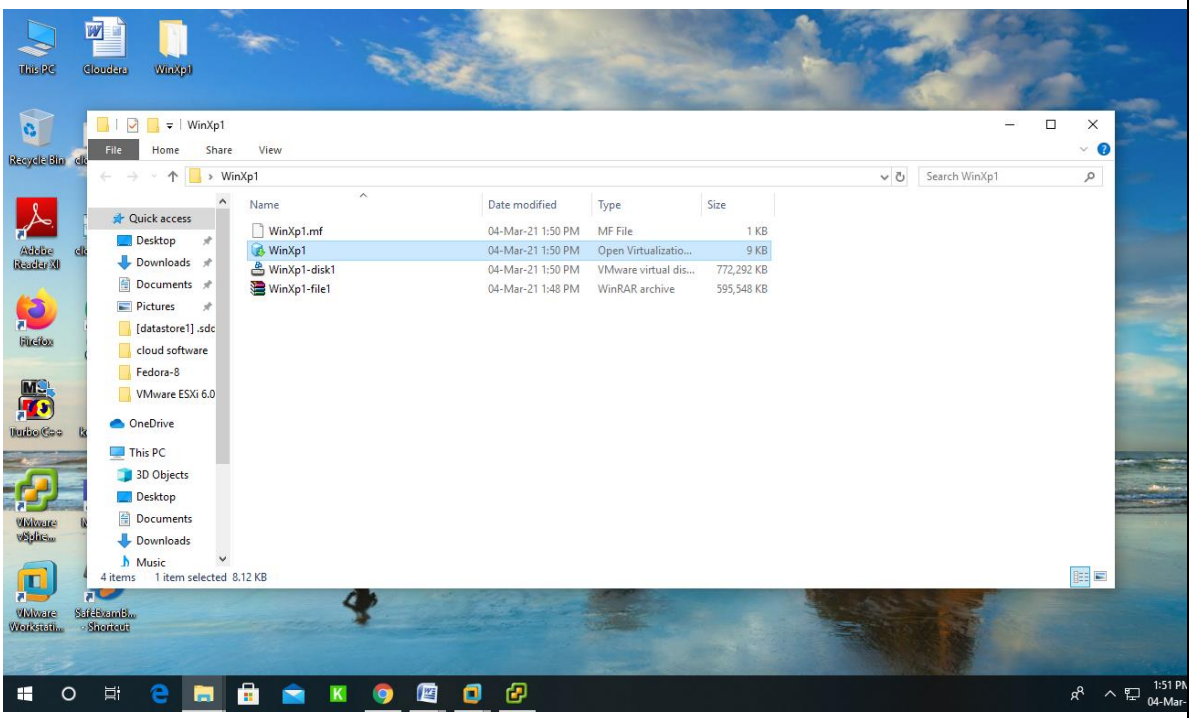
4) The Export is in progress.



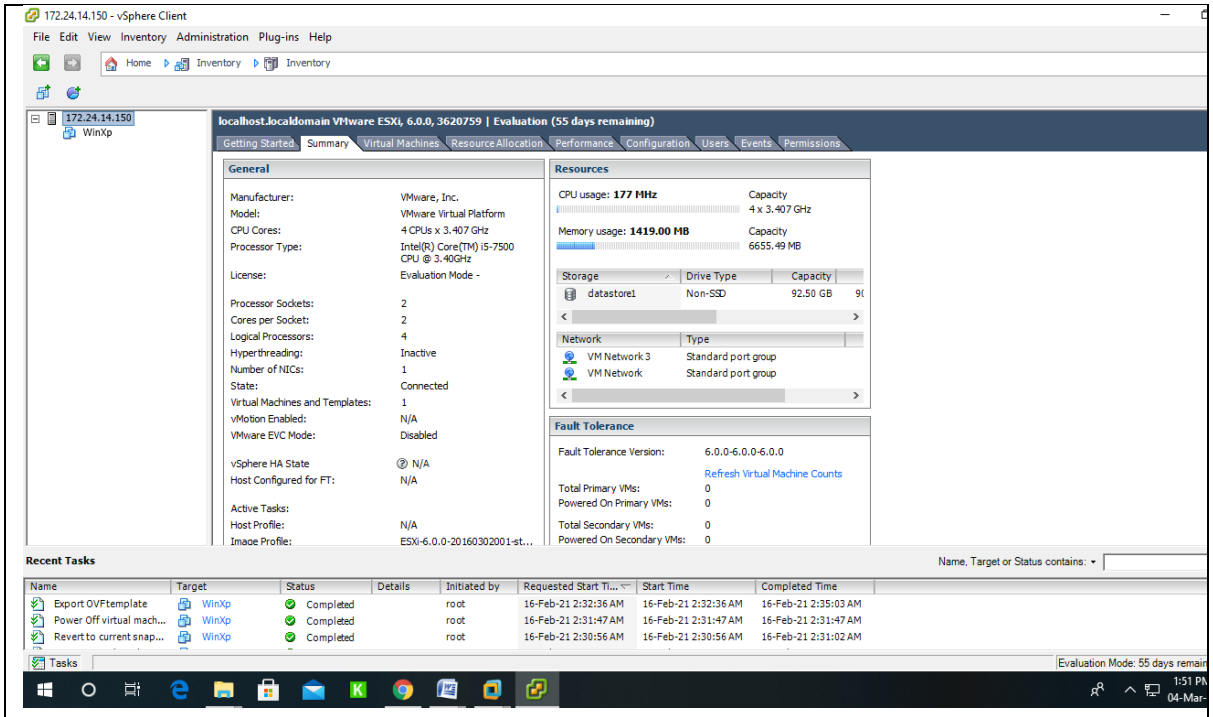
5) The export is completed successfully.



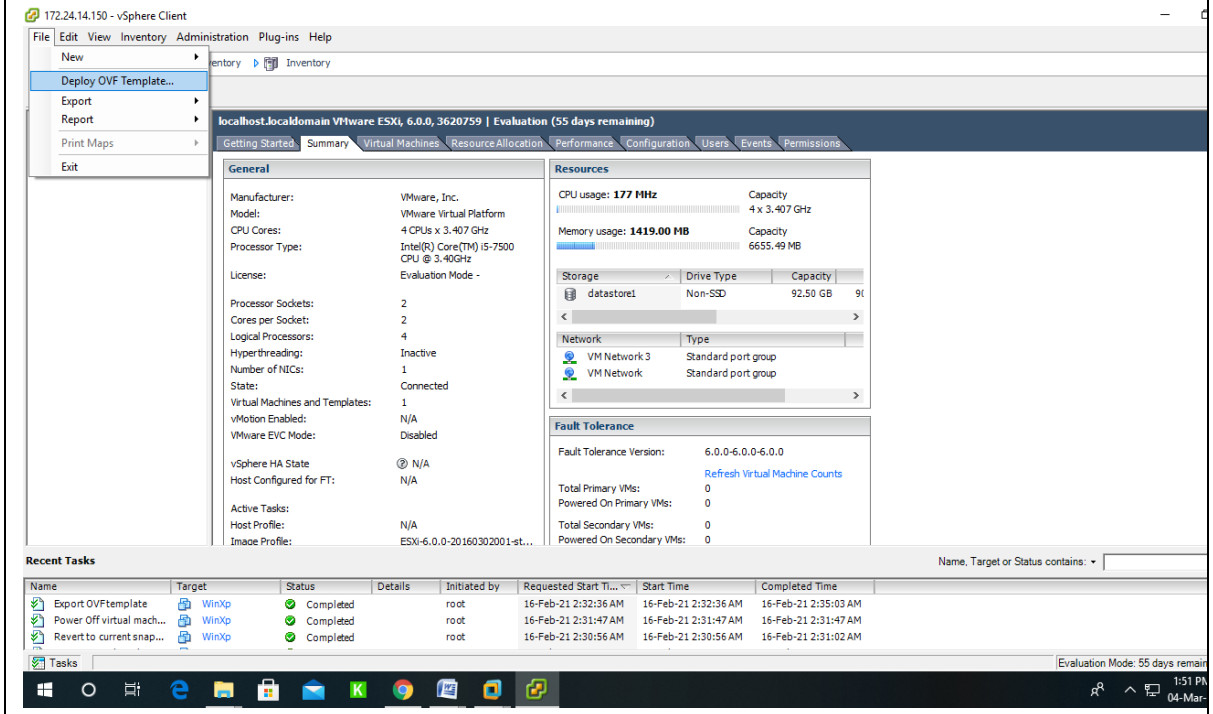
6) Goto location where the VM file is stored. Check for an OVF file in it.



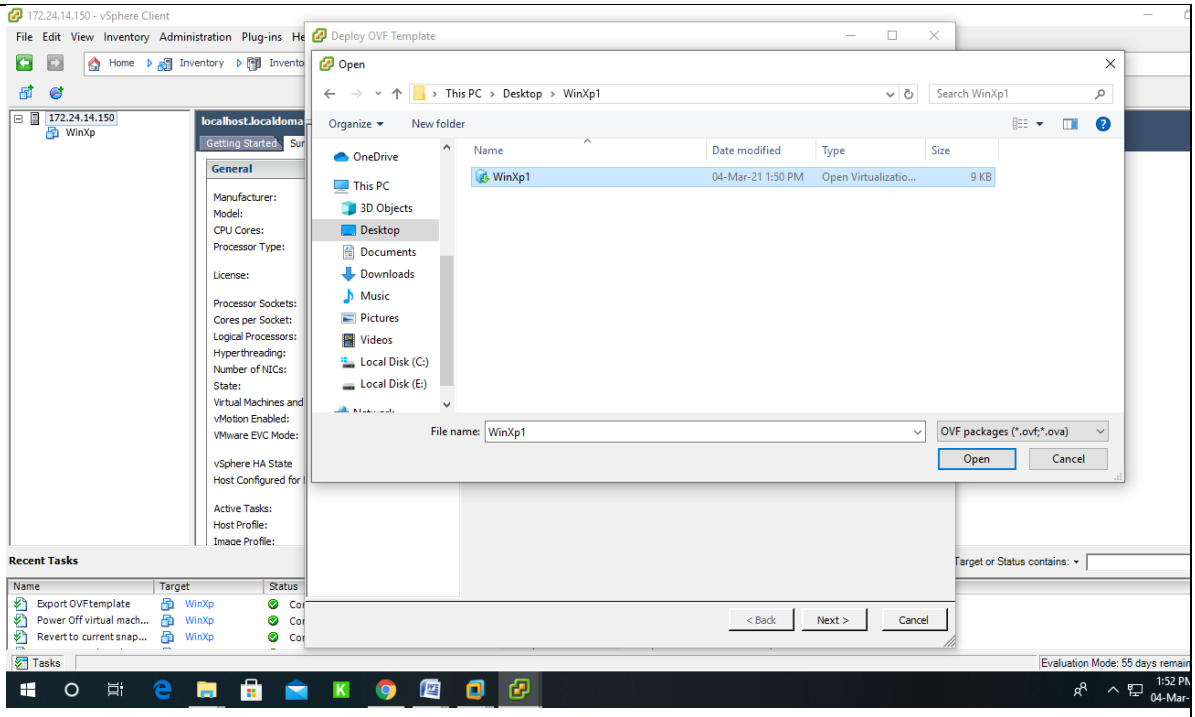
7) Goto inventory, Select the hypervisor on which the VM is tobe cloned.



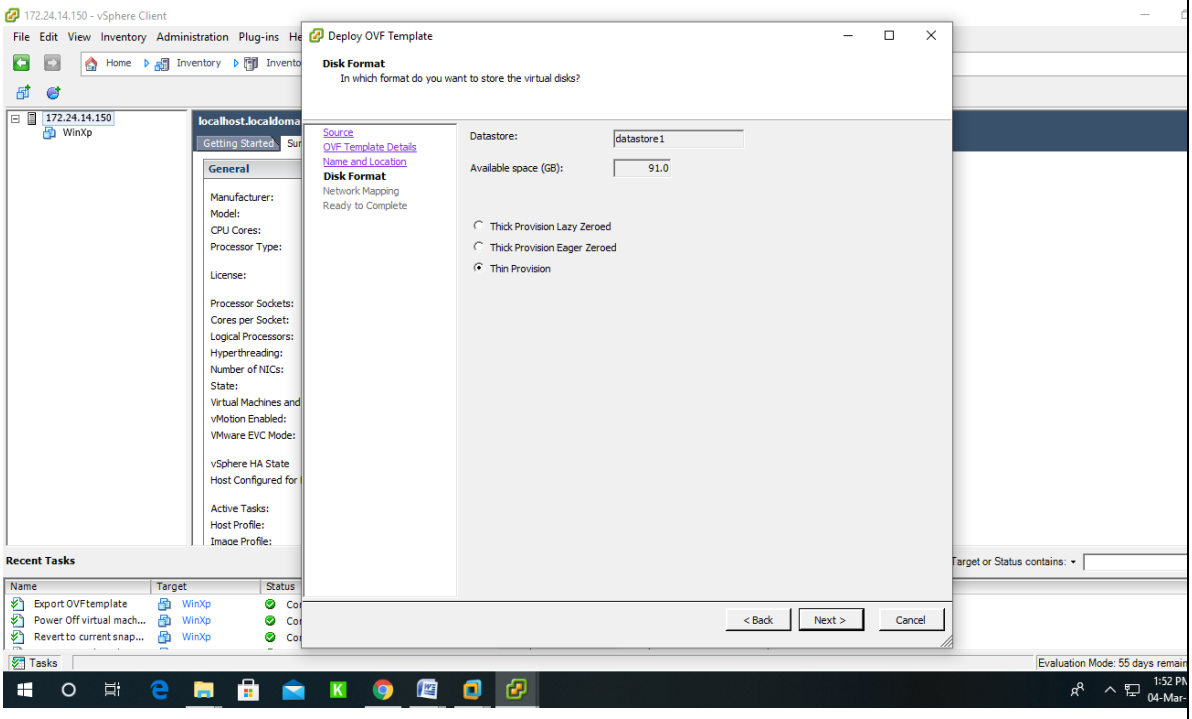
8) Goto file->Deploy OVF template.



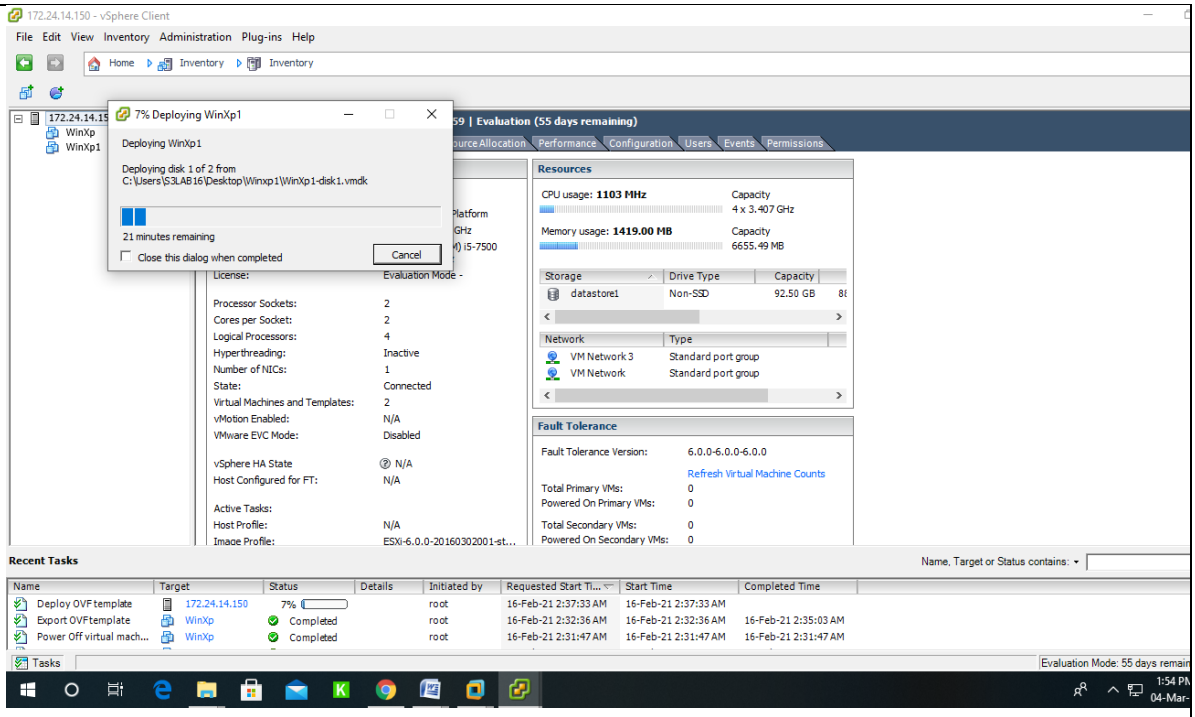
9) Browse for the location where the OVF file is available.



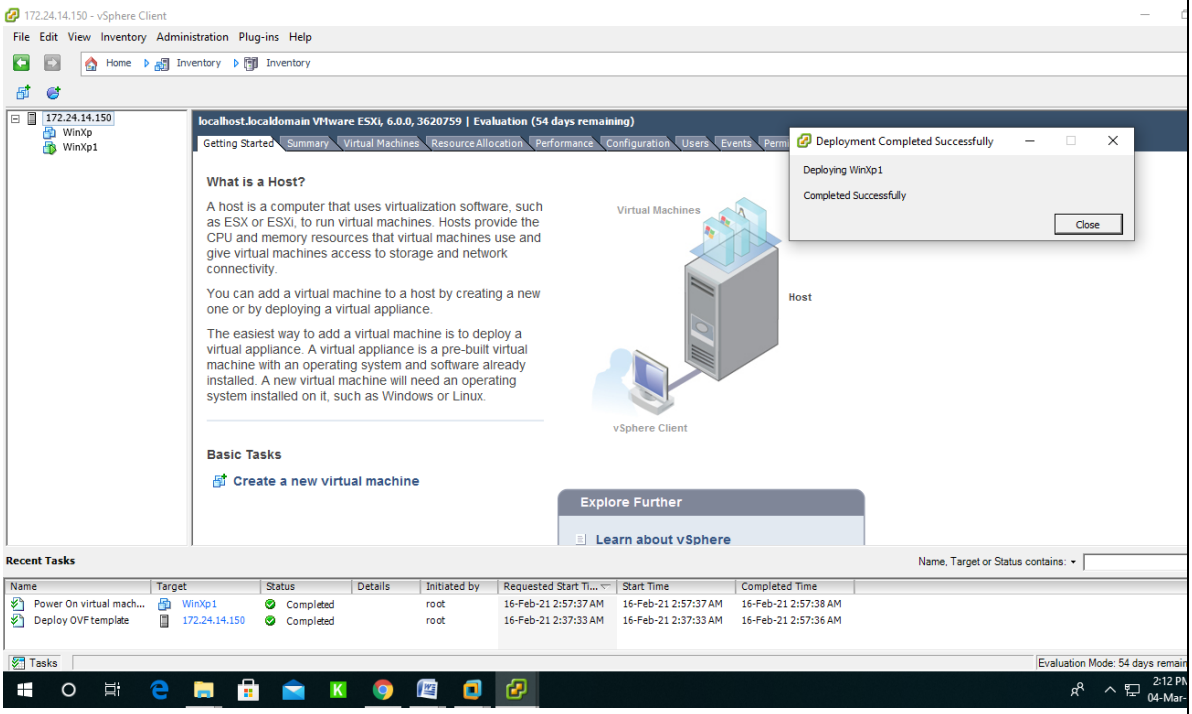
10) Proceed further and select the required options. And finish.



11) The deployment is under progress.

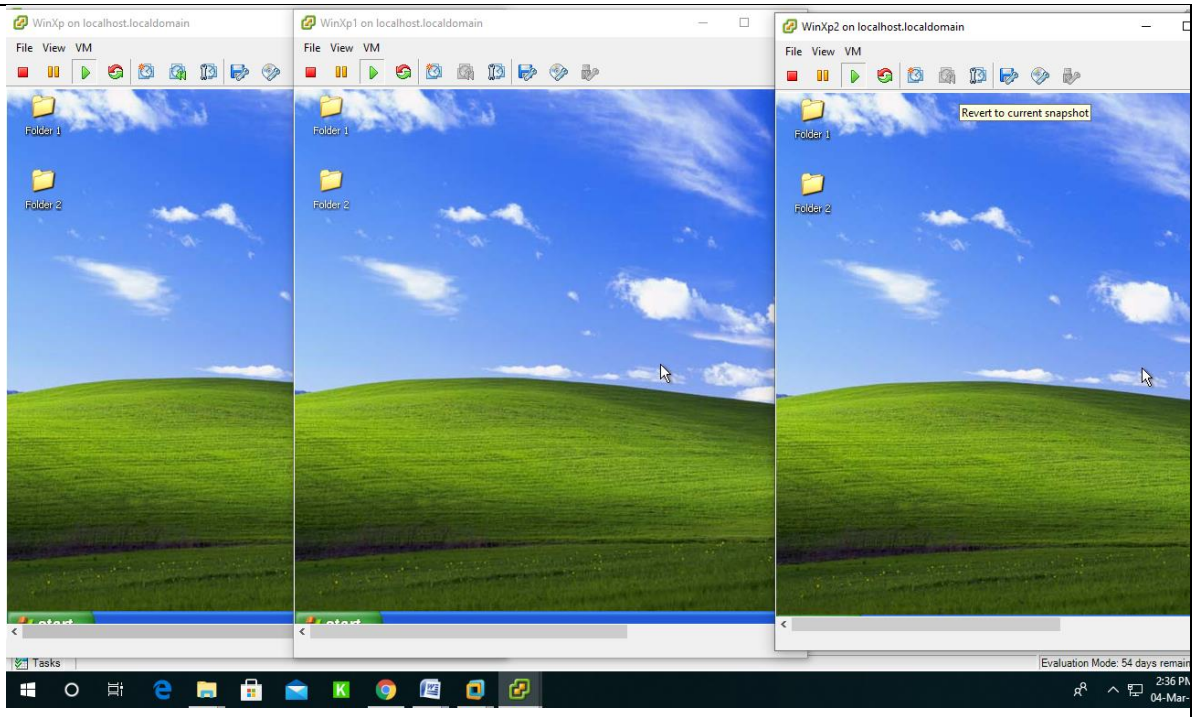


12) The VM is deployed successfully as a new VM.

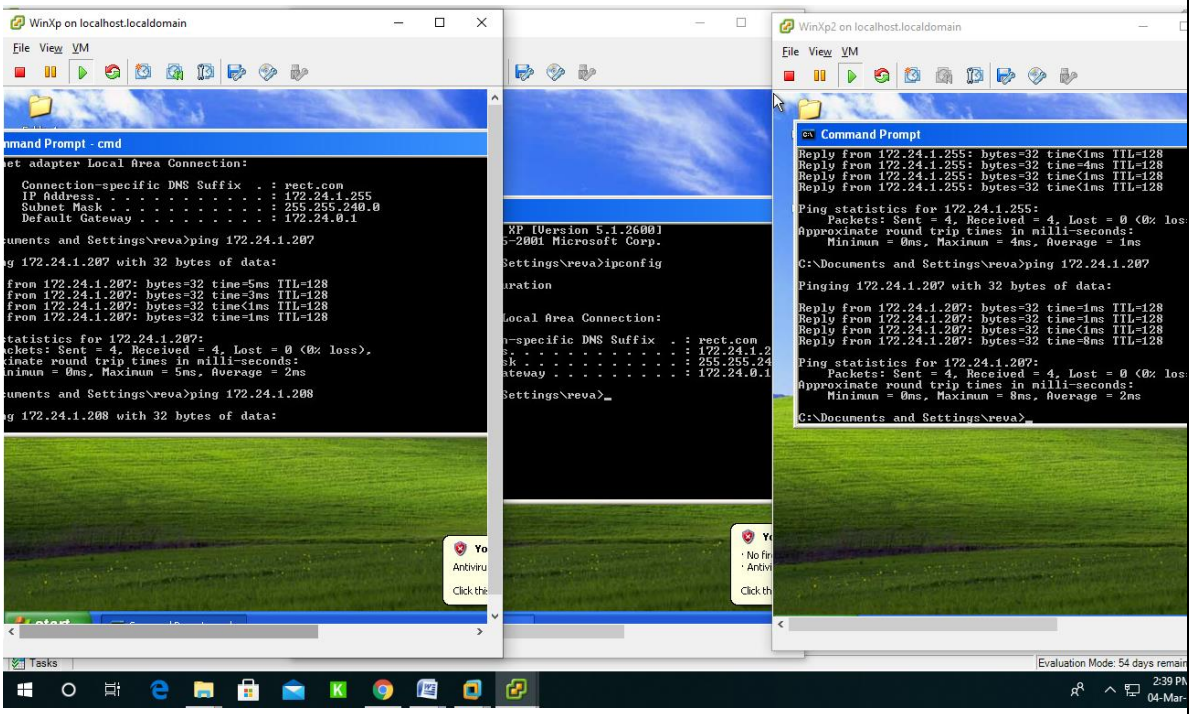


Create one more VM by performing just deployment, no need of export again.

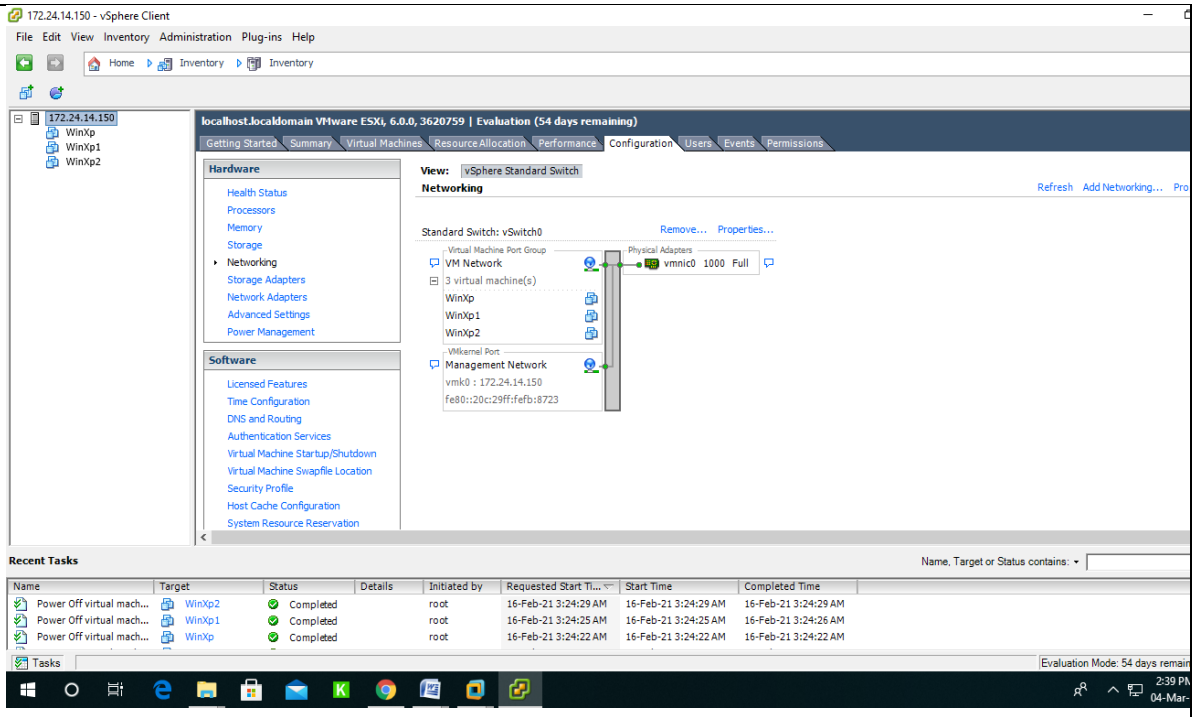
13) Power on and open all 3 VM's. (1 original and 2 cloned VM's)



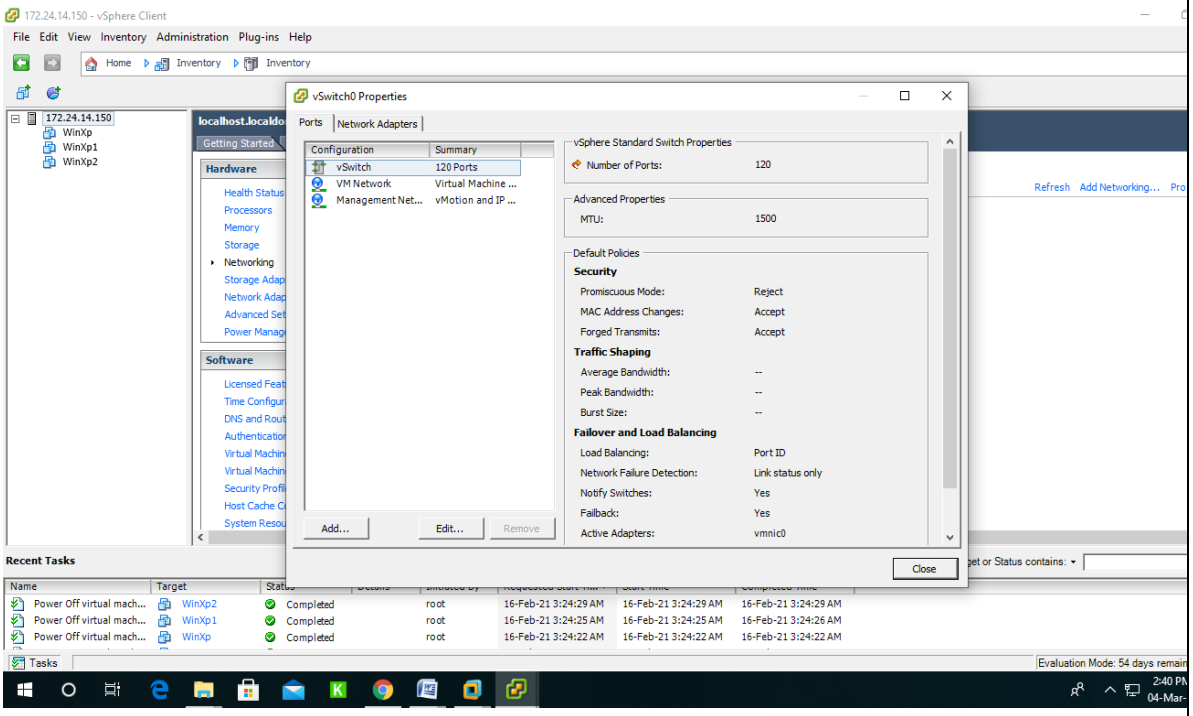
14) Find the IP of those VM's and perform the PING across them and observe the result.



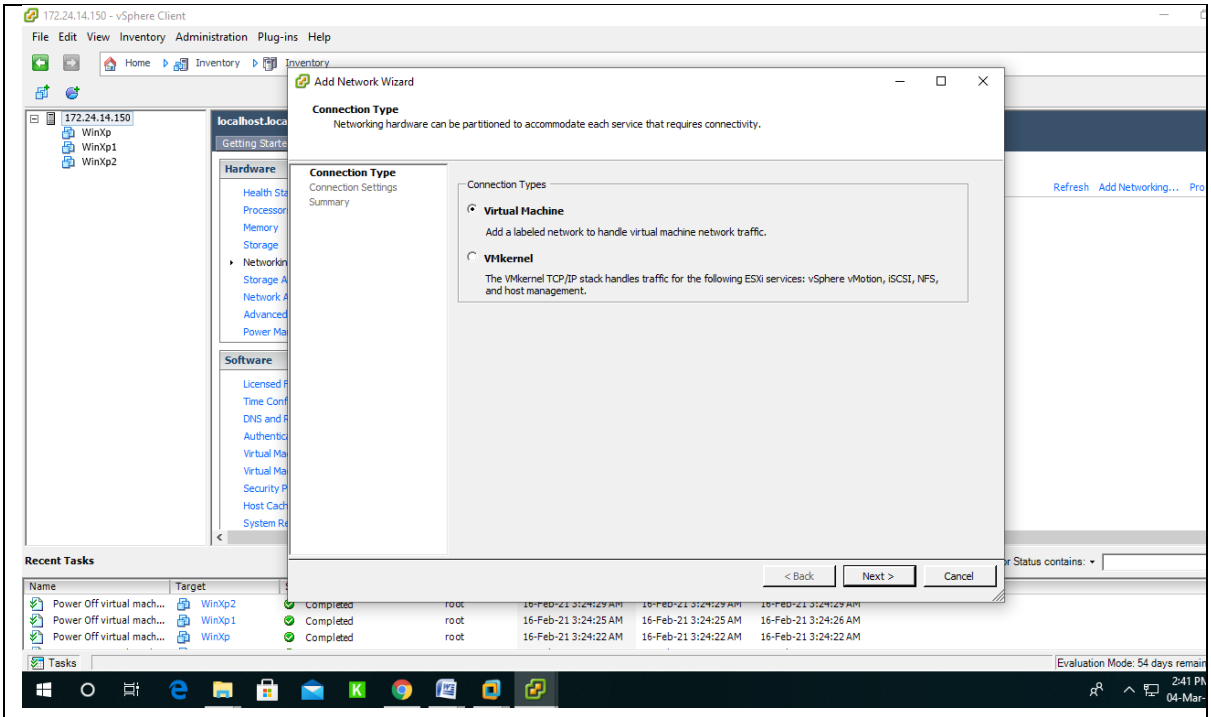
15) For creating the multiple networks on same switch. Select the hypervisor, goto Configuration tab and select Networking.



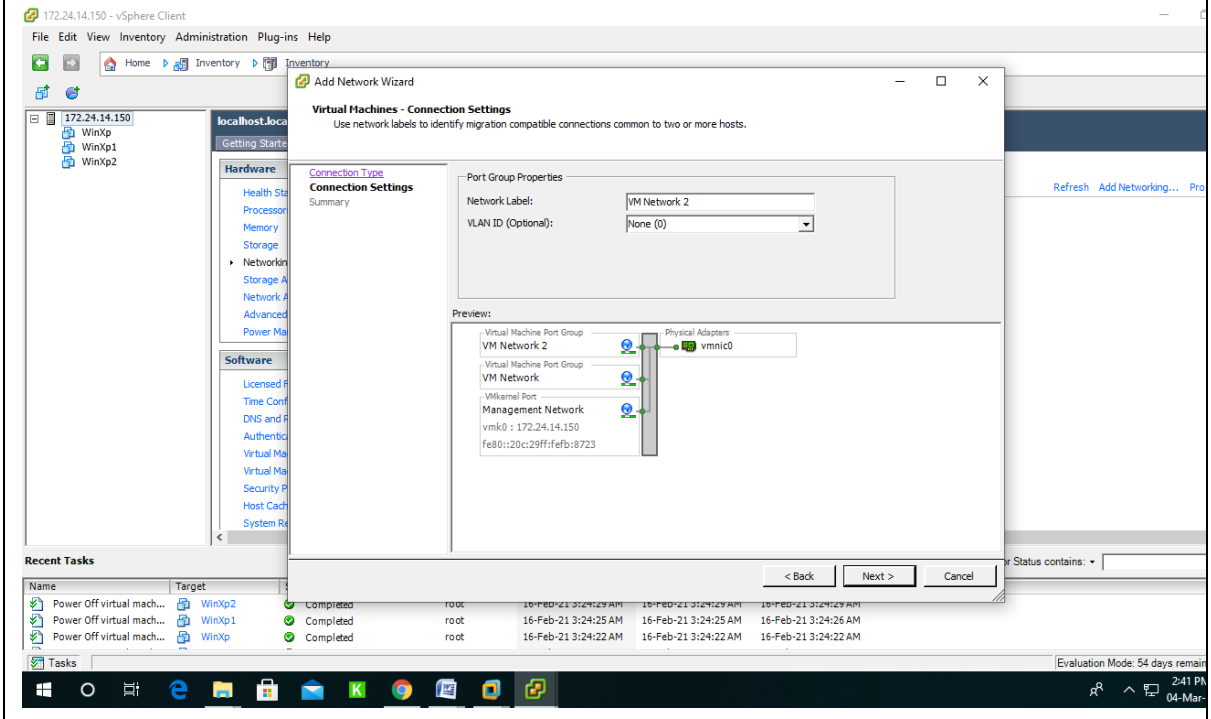
16) Use the option ‘Add’



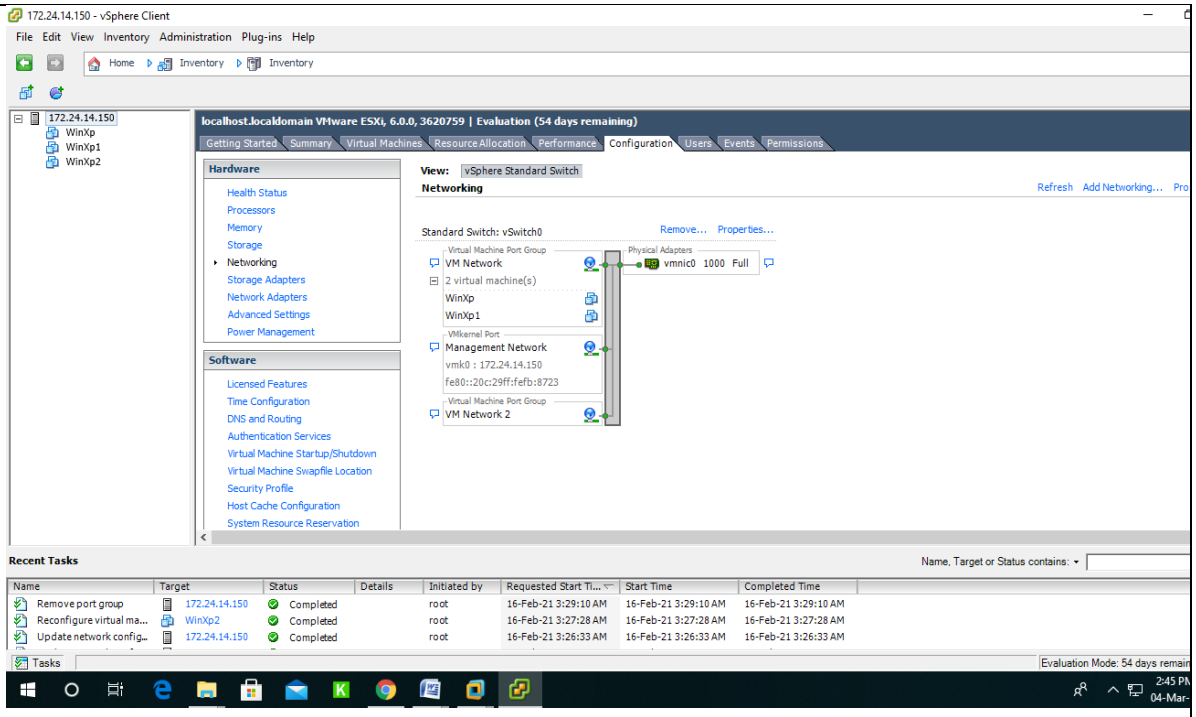
17) Select the Virtual Machine for which the networks is been created.



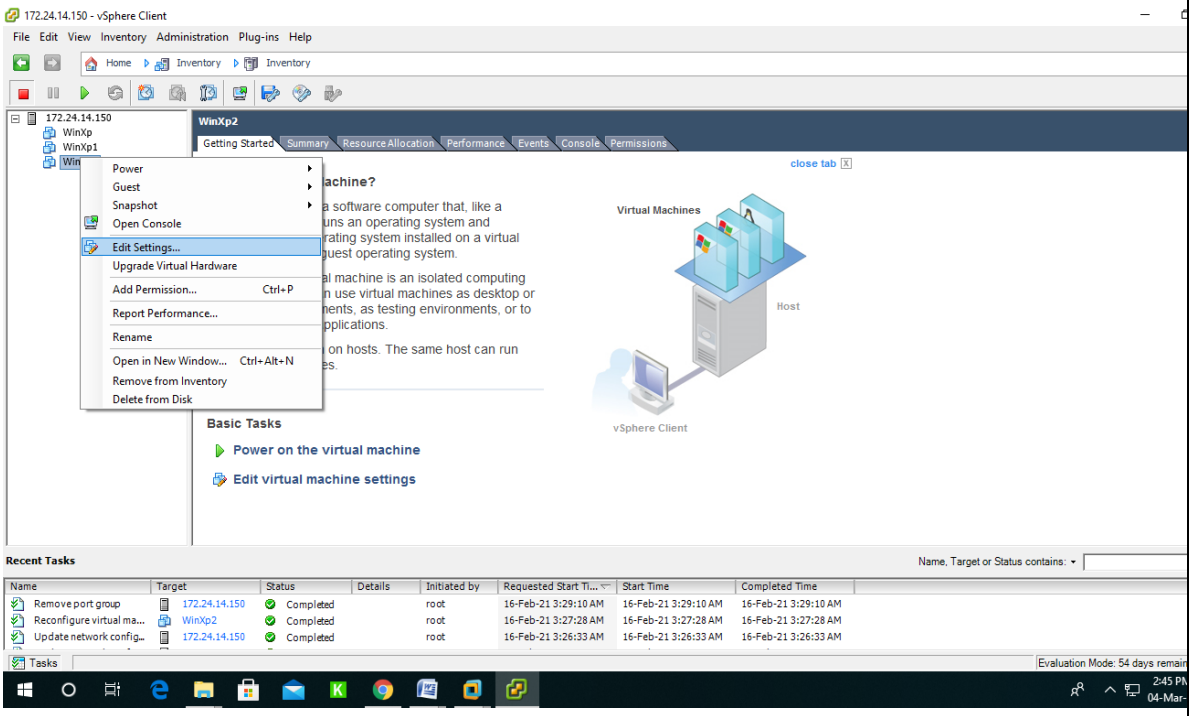
18) Provide the name of the network. And proceed further.



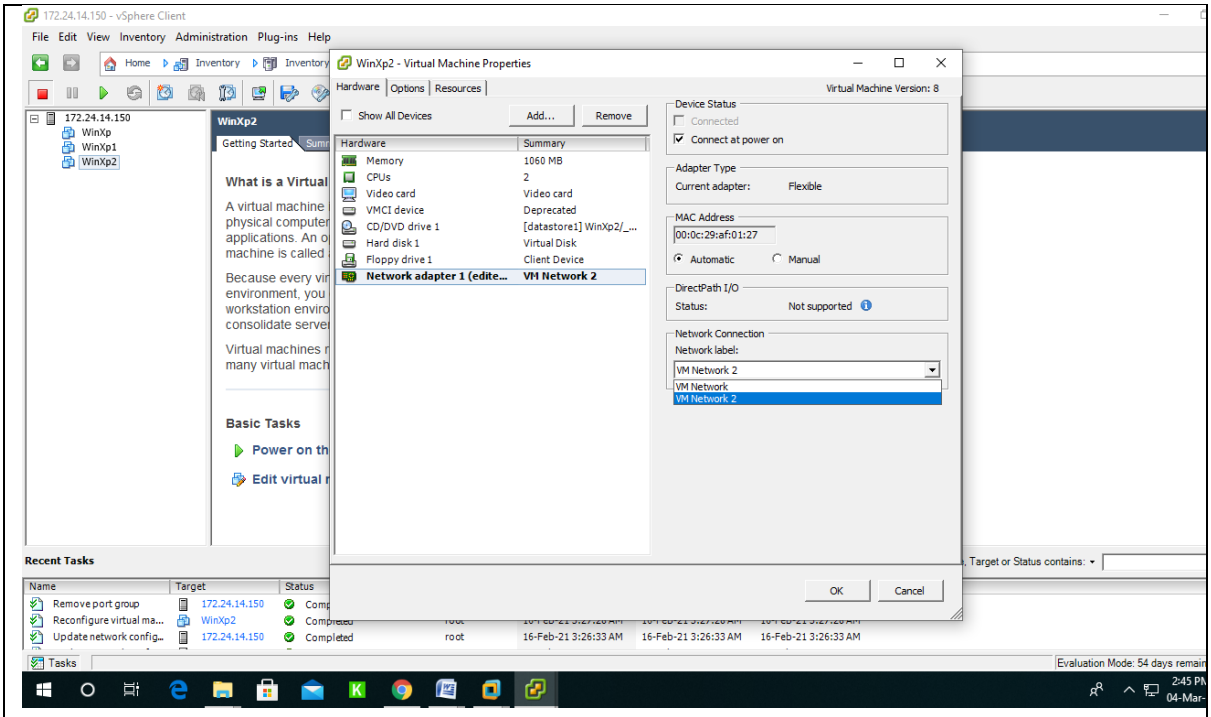
19) Once the multiple networks available on switch.



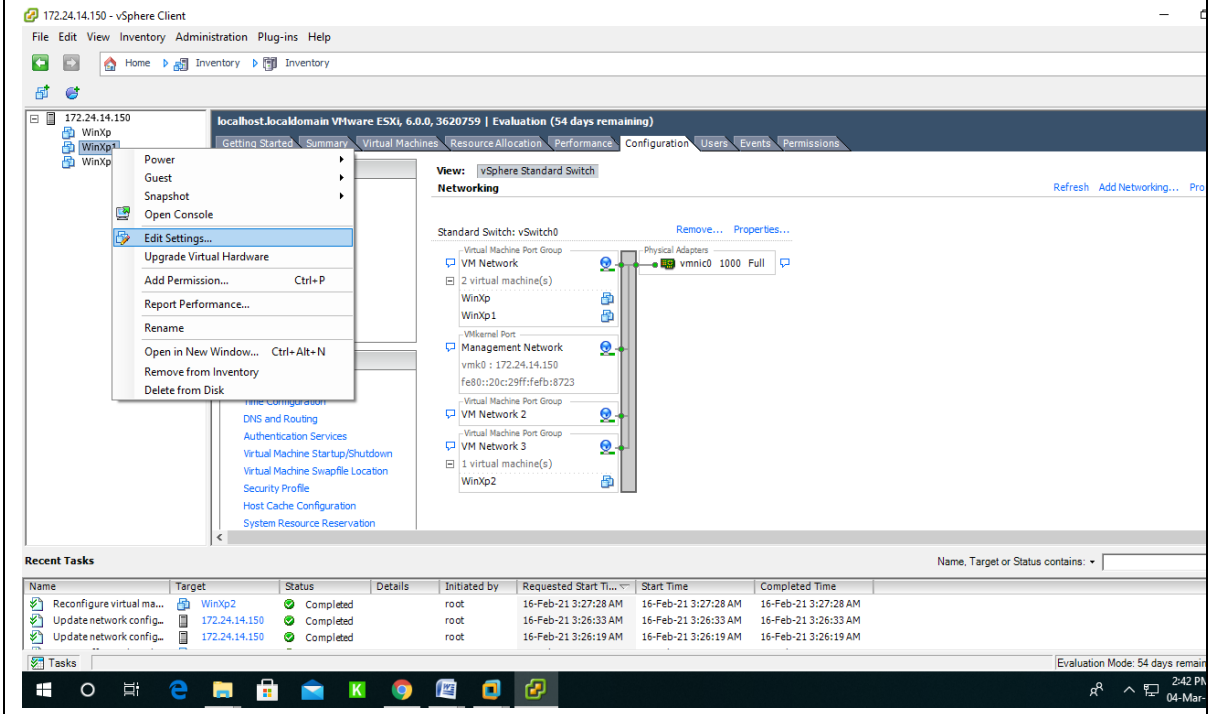
20) Right click on the VM which need to be moved to different network. Select Edit Settings.



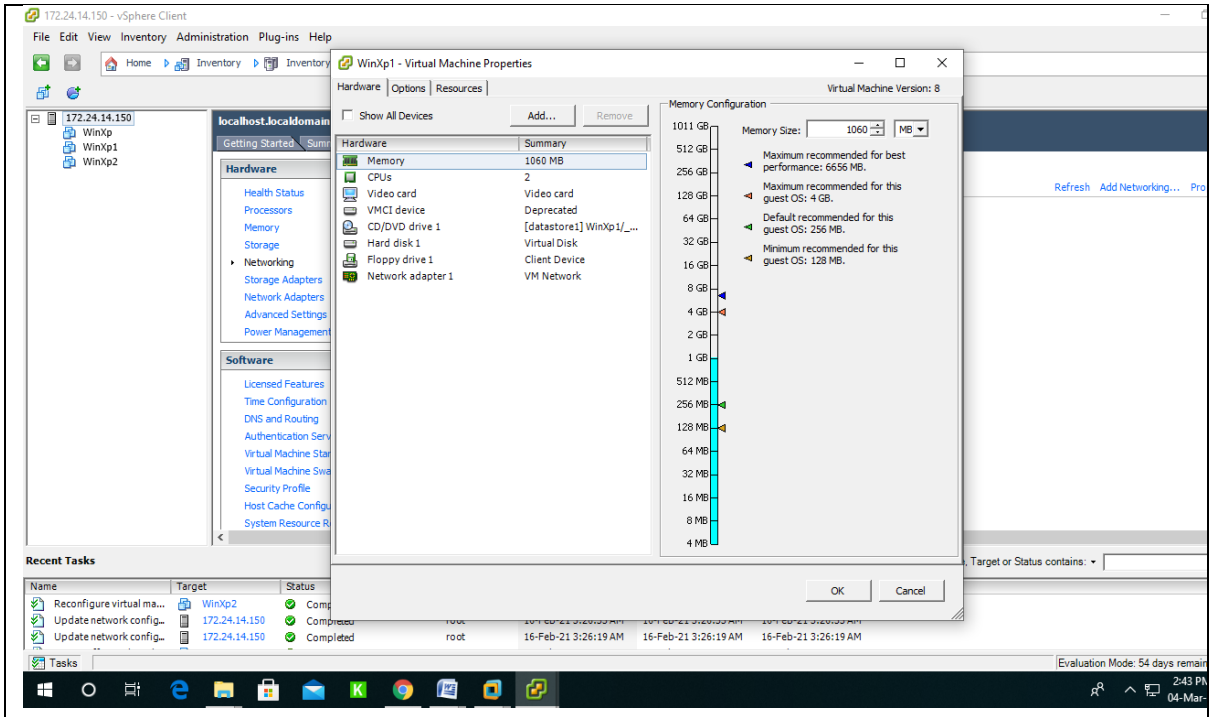
21) Select Network adapter, goto network label and select VMnetwork 2. The different network.



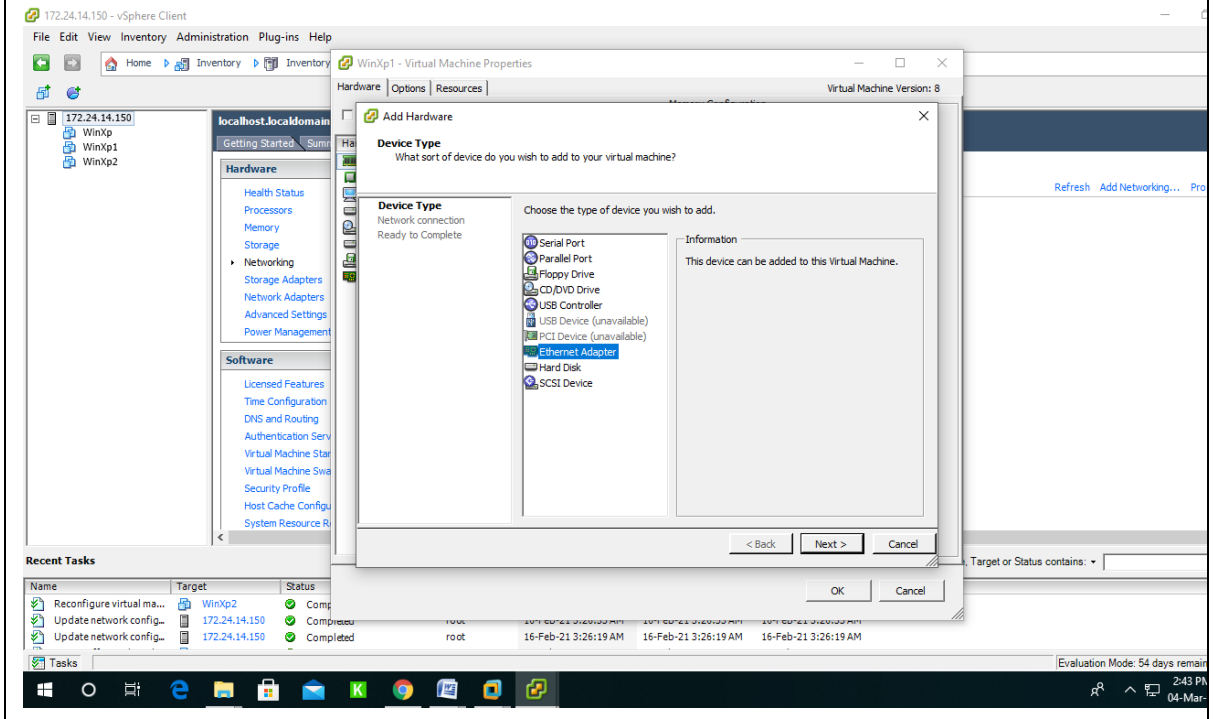
22) To add multiple adapters to a VM, Select and goto edit settings.

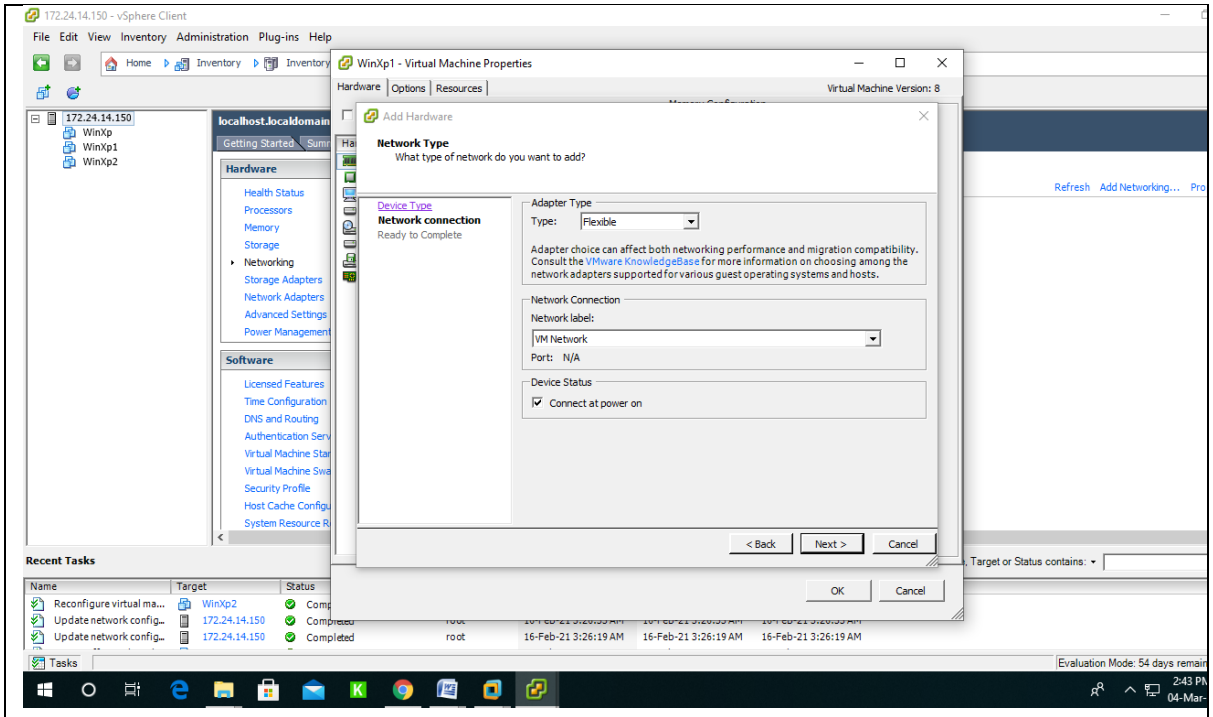


23) Use the option, Add.

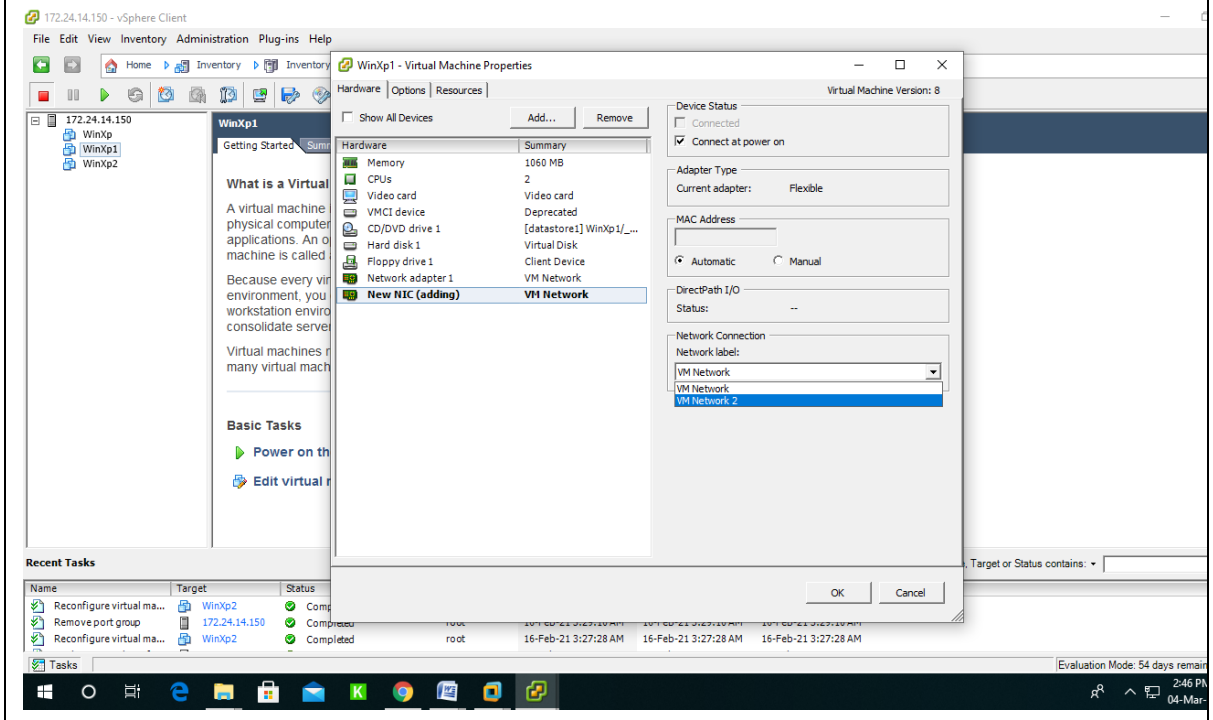


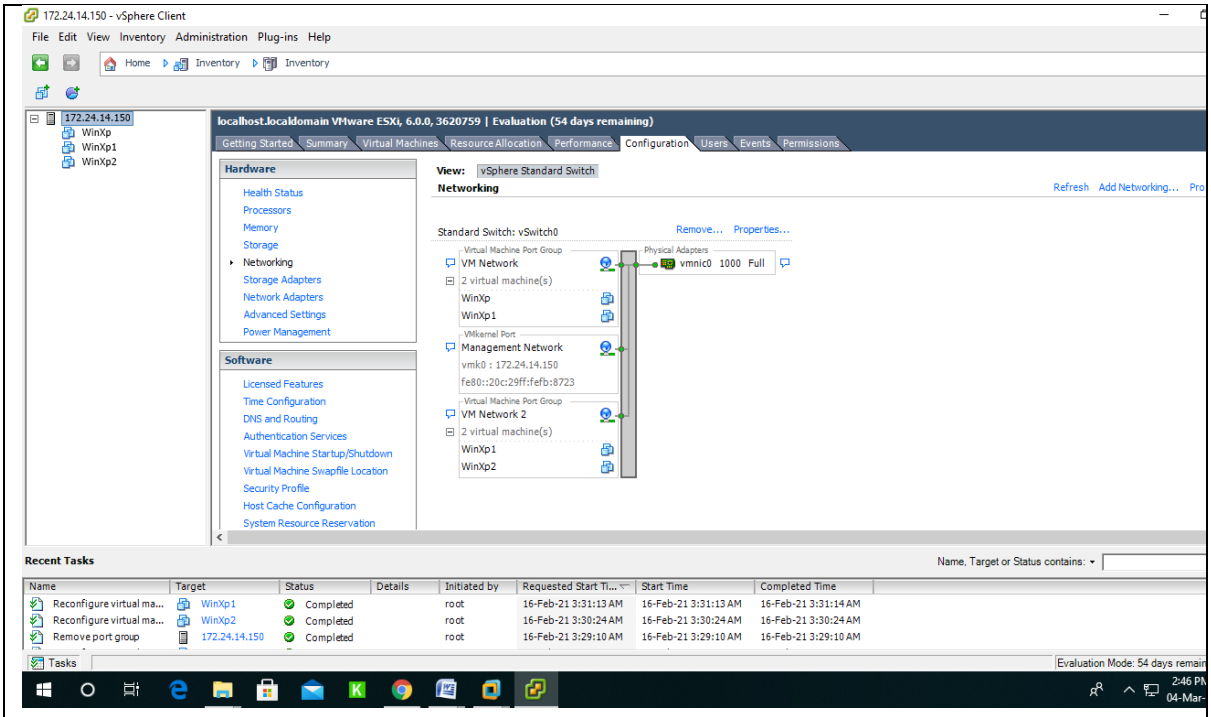
24) Select Ether Adapter. Proceed further and finish.



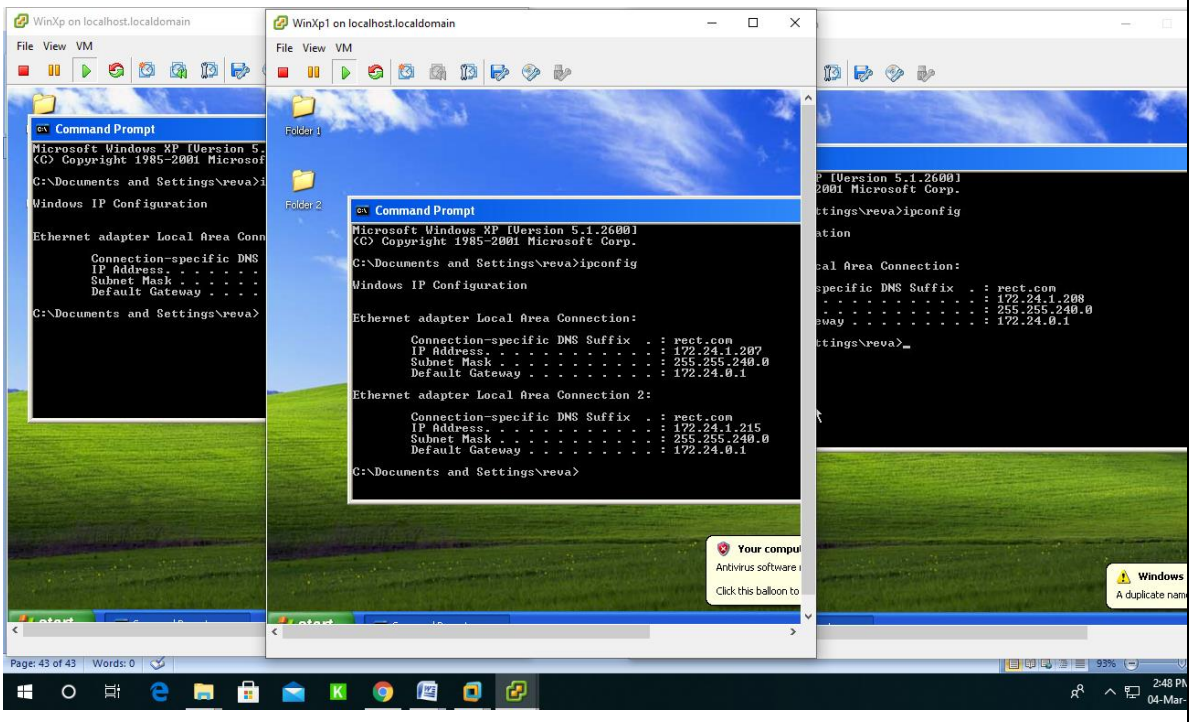


25) Now select the second adapter, network label and choose the different network. Make sure 2 adapters are with 2 networks.





26) Power on all the VM's, Open in console, Ping across them and observe the results.



Session 3

Problem Statement:

Demonstrates how to simulate a Data Center with one host and run one Cloudlet on it using cloudsim.

Student Learning Outcomes:

Simulate a Data Center with one host and to run one Cloudlet using CloudSim.

Theoretical Description:

CloudSim Architecture:

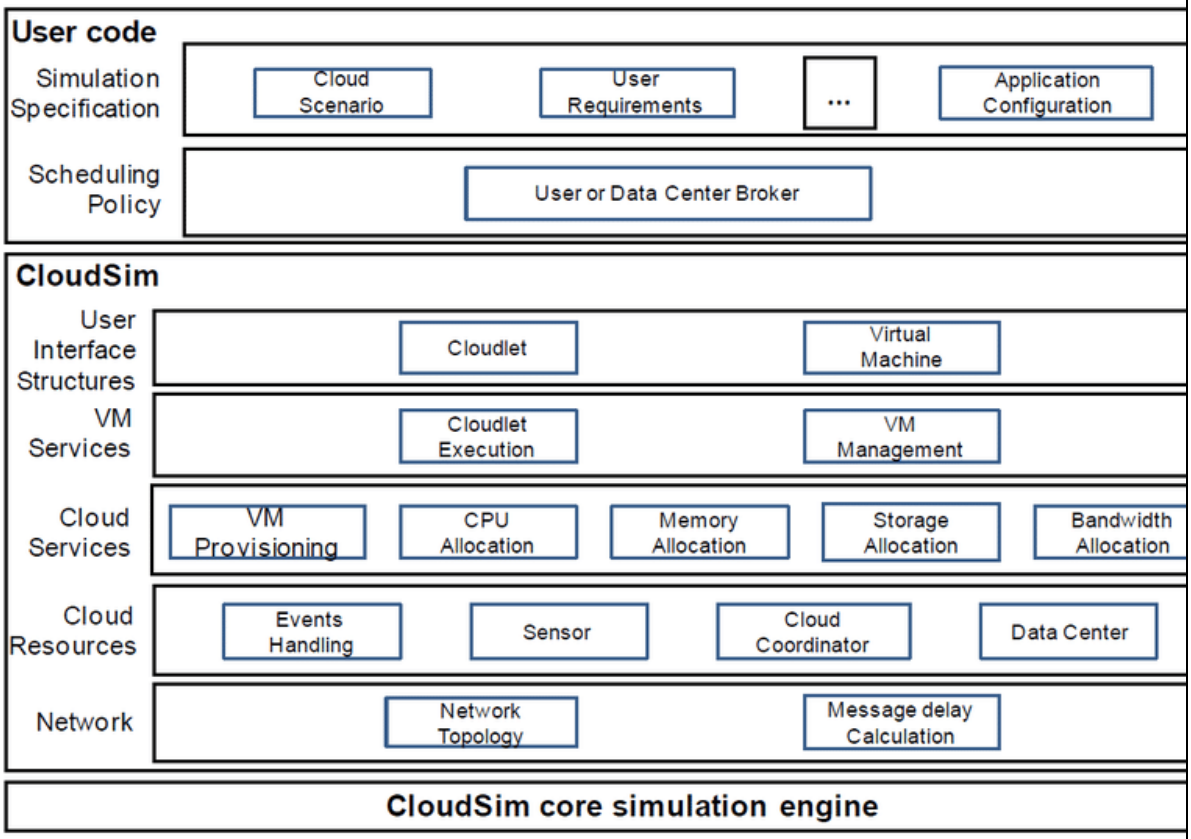


Figure 1: Layered CloudSim Architecture

Above diagram demonstrates about the layered architecture of CloudSim. The **CloudSim Core simulation engine** provides support for modeling and simulation of virtualized Cloud-based data center environments including queuing and processing of events, creation of cloud system entities (like data center, host, virtual machines, brokers, services etc.) communication between components and management of the simulation clock. The **CloudSim layer** provides the dedicated management interfaces for Virtual Machines, memory, storage, and bandwidth. Also it manages the other fundamental issues, such as provisioning of hosts to Virtual Machines, managing application execution, and monitoring dynamic system state(e.g. Network topology, sensors, storage characteristics etc) etc.

And the **User Code layer** is a custom layer where user write their own code to redefine the Characteristics of the simulating environment as per their new research findings.

Design and Implementation of CloudSim

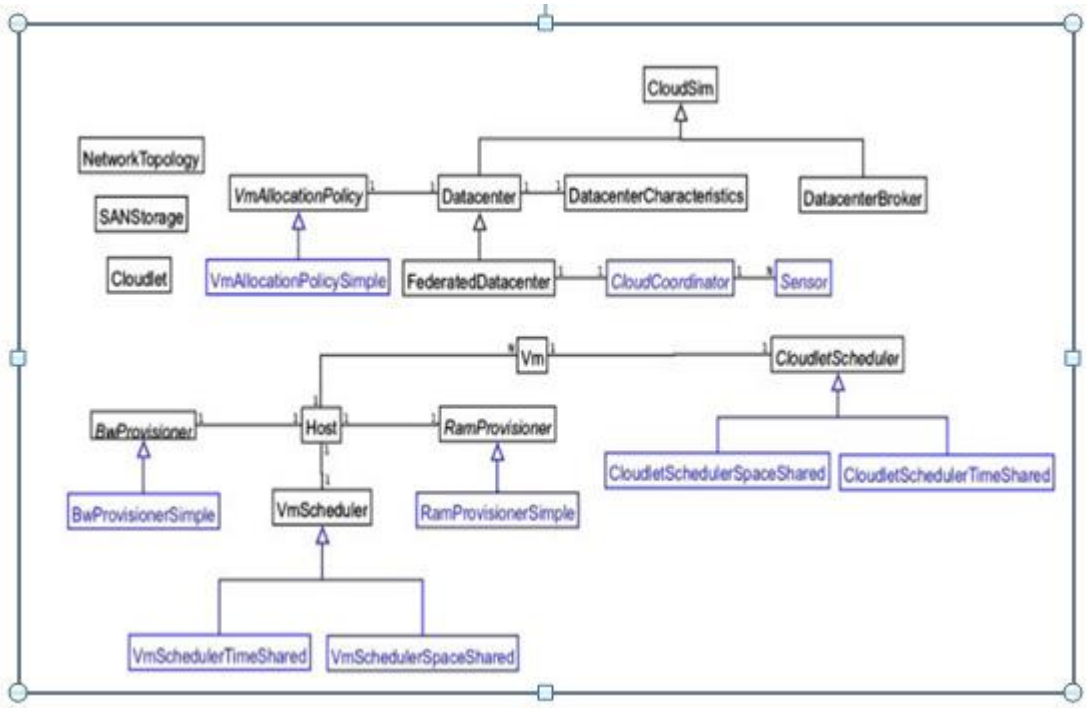


Figure 2: CloudSim Class Design Diagram

The description of all the major classes mentioned in class diagram above is described below:

1. **BwProvisioner**: The main role of this component is to undertake the allocation of network bandwidths to a set of competing VMs that are deployed across the data center. Cloud system developers and researchers can extend this class with their own policies (priority, QoS) to reflect
the needs of their applications.

2. **CloudCoordinator**: It is responsible for periodically monitoring the internal state of data center resources and based on that it undertakes dynamic load-shredding decisions. Developers aiming to deploy their application services across multiple clouds can extend this class for implementing their custom inter-cloud provisioning policies.

3. **Cloudlet**: This class models the Cloud-based application services (program based tasks) such as content delivery, social networking, and business workflow. CloudSim mimics the complexity of an application in terms of its computational requirements. Every application service has a preassigned instruction length and data transfer (both pre and post fetches) overhead that it needs
to undertake during its life cycle.

4. **CloudletScheduler**: This is responsible for implementation of different policies that determine the share of processing power among Cloudlets in a VM. There are two types of provisioning policies offered: space-shared (using **CloudetSchedulerSpaceShared** class) and time-shared(using **CloudletSchedulerTimeShared** class).

5. **Datacenter**: This class models the core infrastructure-level services (i.e. hardware) that are offered by Cloud providers (Amazon, Azure, and App Engine). It encapsulates a set of compute
hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations (memory, cores, capacity, and storage). Also, every Datacenter component instantiates a generalized application provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices to hosts and VMs.

6. **DatacenterBroker or Cloud Broker**: This class models a broker, which is responsible for mediating negotiations between SaaS and Cloud providers; and such negotiations are driven by

QoS requirements. The broker class acts on behalf of applications. It discovers suitable Cloud service providers by querying the CIS and undertakes online negotiations for allocation of resources/services that can meet the application's QoS needs. **Researchers and system developers** must extend this class for evaluating and testing custom brokering policies. The difference between the broker and the CloudCoordinator is that the broker represents the customer (i.e. decisions of these components are made in order to increase user-related performance metrics), whereas the CloudCoordinator acts on behalf of the data center, i.e. it tries to maximize the overall performance of the data center, without considering the needs of specific customers.

7. **DatacenterCharacteristics**: This class contains configuration information of data center resources.

8. **Host**: This class models a physical resource such as a compute or storage server. It encapsulates important information such as the amount of memory and storage, a list and type of processing cores (to represent a multi-core machine), an allocation of policy for sharing the processing power among VMs, and policies for provisioning memory and bandwidth to the VMs.

9. **NetworkTopology**: This class contains the information for inducing network behavior (latencies) in the simulation. It stores the topology information, which is generated using the BRITE topology generator.

10. **RamProvisioner**: This is an abstract class that represents the provisioning policy for allocating primary memory (RAM) to the Virtual Machines. The execution and deployment of VM on a host is feasible only if the RamProvisioner component approves that the host has the required amount of free memory. The RamProvisionerSimple does not enforce any limitation on the amount of memory that a VM may request. However, if the request is beyond the available
memory capacity, then it is simply rejected.

11. **Vm**: This class models a Virtual Machine (VM), which is managed and hosted by a Cloud host component. Every VM component has access to a component that stores the following characteristics related to a VM (i.e.) accessible memory, processor, storage size, and the VM's
internal provisioning policy that is extended from an abstract class called the CloudletScheduler.

12. **VmAllocationPolicy**: This abstract class represents a provisioning policy that a VM Monitor
utilizes for allocating VMs to hosts. The main functionality of the VmAllocationPolicy is to select the available host in a data center that meets the memory, storage, and availability requirement for a VM deployment.

13. **VmScheduler**: This is an abstract class implemented by a Host component that models the
policies (space-shared, time-shared) required for allocating processor cores to VMs. The functionalities of this class can easily be overridden to accommodate application-specific processor sharing policies.

14. **CloudSim**: This is the main class, which is responsible for managing event queues and controlling step-by-step (sequential) execution of simulation events. Every event that is generated by the CloudSim entity at run-time is stored in the queue called future events. These events are sorted by their time parameter and inserted into the queue. Next, the events that are scheduled at each step of the simulation are removed from the future events queue and transferred to the deferred event queue. Following this, an event processing method is invoked for each entity, which chooses events from the deferred event queue and performs appropriate actions. Such an organization allows flexible management of simulation and provides the following powerful capabilities:

a. Deactivation (holding/pausing) of entities.

b. Context switching of entities between different states (e.g. waiting to active). Pausing
and resuming the process of simulation.
c. Creation of new entities at run-time.
d. Aborting and restarting simulation at run-time.

15. **DeferredQueue**: This class implements the deferred event queue used by CloudSim.
16. **FutureQueue**: This class implements the future event queue accessed by CloudSim.
17. **CloudInformationService**: A CIS is an entity that provides resource registration, indexing, and discovering capabilities. CIS supports two basic primitives:
a. **publish()**, which allows entities to register themselves with CIS and
b. **search()**, which allows entities such as CloudCoordinator and Brokers in discovering
status and endpoint contact address of other entities. This entity also notifies the other entities about the end of simulation.
18. **SimEntity**: This is an abstract class, which represents a simulation entity (such as Cloudlet, VM, Host etc) that is able to send messages to other entities and process received messages as well as fire and handle events. SimEntity class provides the ability to schedule new events and send messages to other entities, where network delay is calculated according to the BRITE model. Once created, entities automatically register with CIS. All entities must extend this class and override its three core methods:
a. **startEntity**(), which define actions for entity initialization.
b. **processEvent**(), which define actions processing of events.
c. **shutdownEntity**(),which define actions entity destruction.
19. **CloudSimTags**. This class contains various static event/command tags that indicate the type of action that needs to be undertaken by CloudSim entities when they receive or send events.
20. **SimEvent**: This entity represents a simulation event that is passed between two or more entities. SimEvent stores the following information about an event:
a. type,
b. init time,
c. time at which the event should occur,
d. finish time,
e. time at which the event should be delivered to its destination entity,
f. IDs of the source and destination entities,
g. tag of the event, and
h. Data that have to be passed to the destination entity.
21. **CloudSimShutdown**: This is an entity class that waits for the termination of all end-user and broker entities, and then signals the end of simulation to CIS.

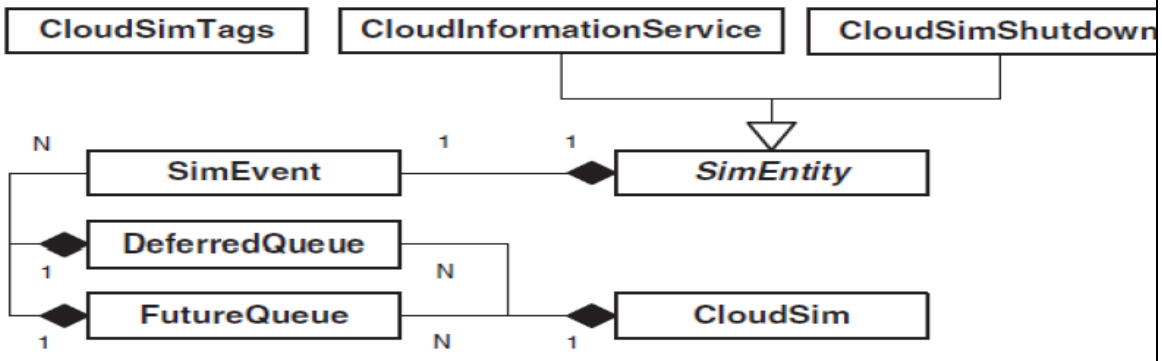


Figure 3: Interrelation of Core Simulation Classes

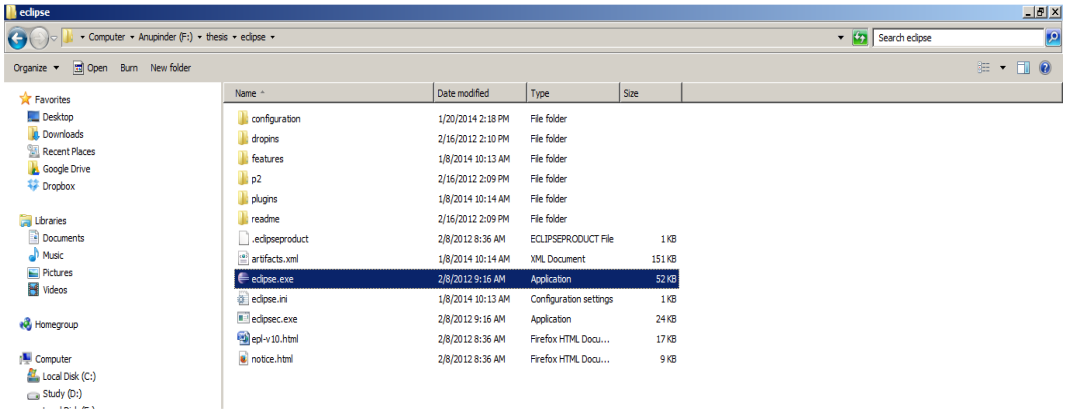
Requirements:

Requirements:

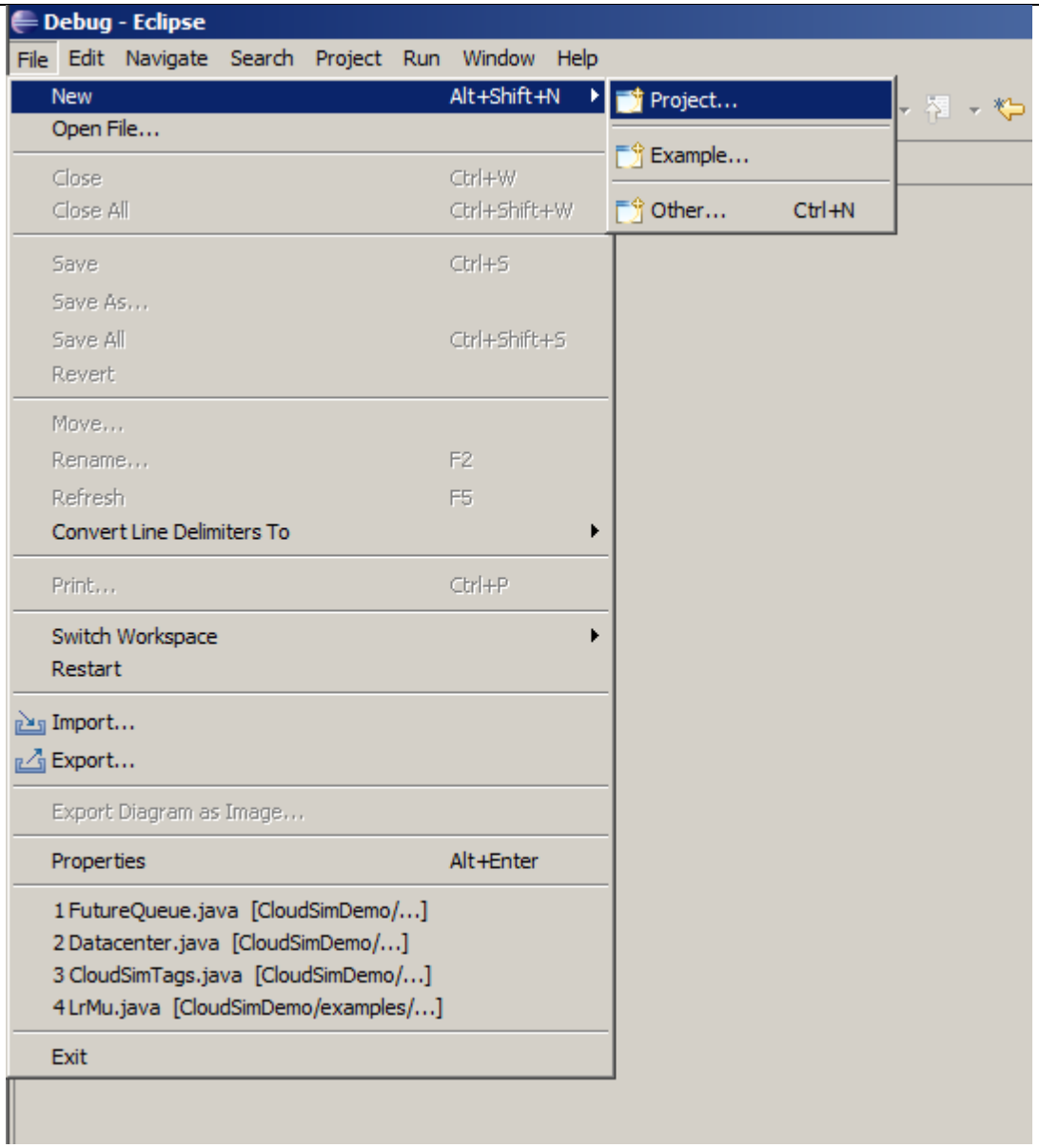
- Eclipse IDE for java developers:
- CloudSim Project code:
- One external requirement of cloudsim i.e. common jar package of math

Procedure:

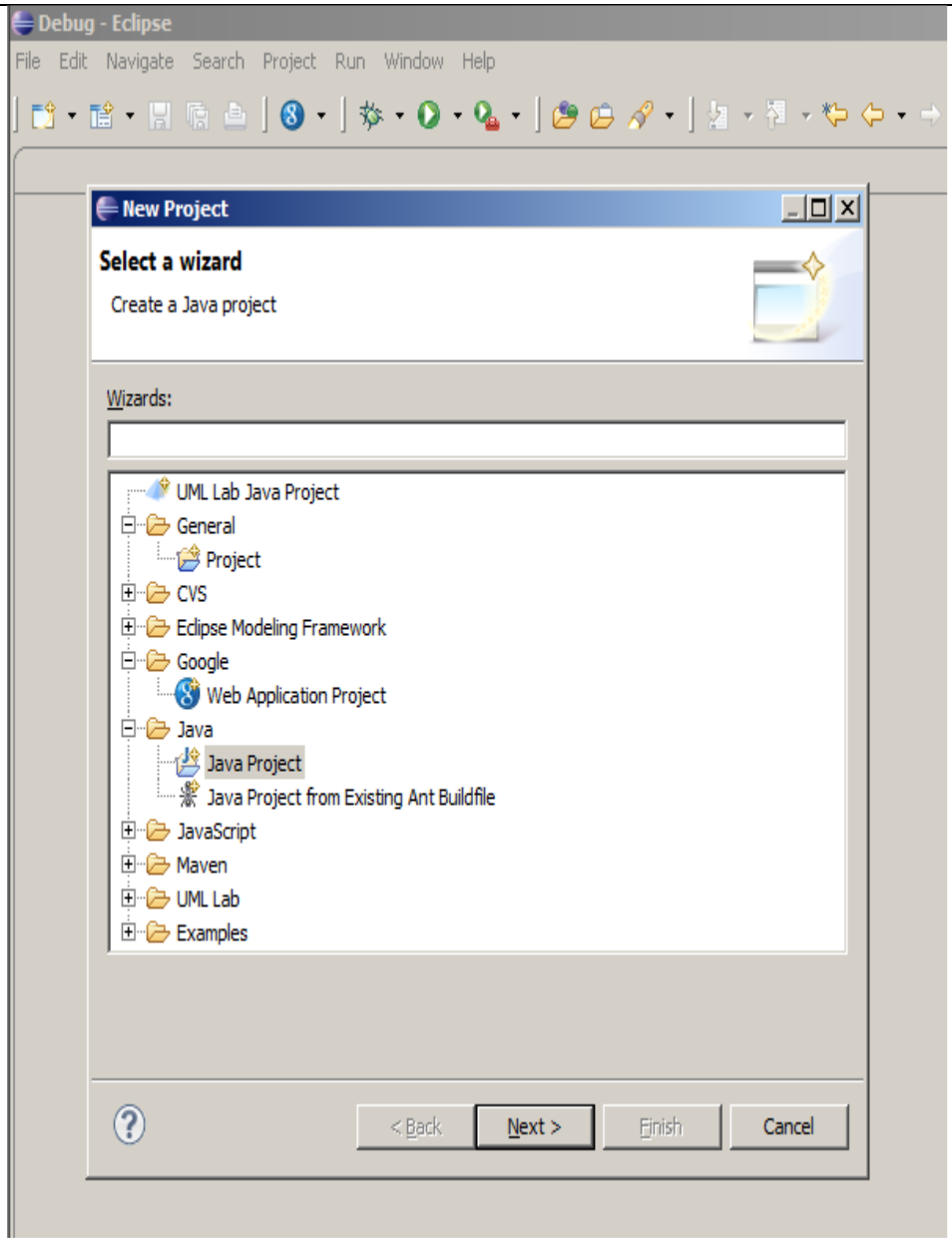
1. Open **eclipse.exe** from eclipse folder:



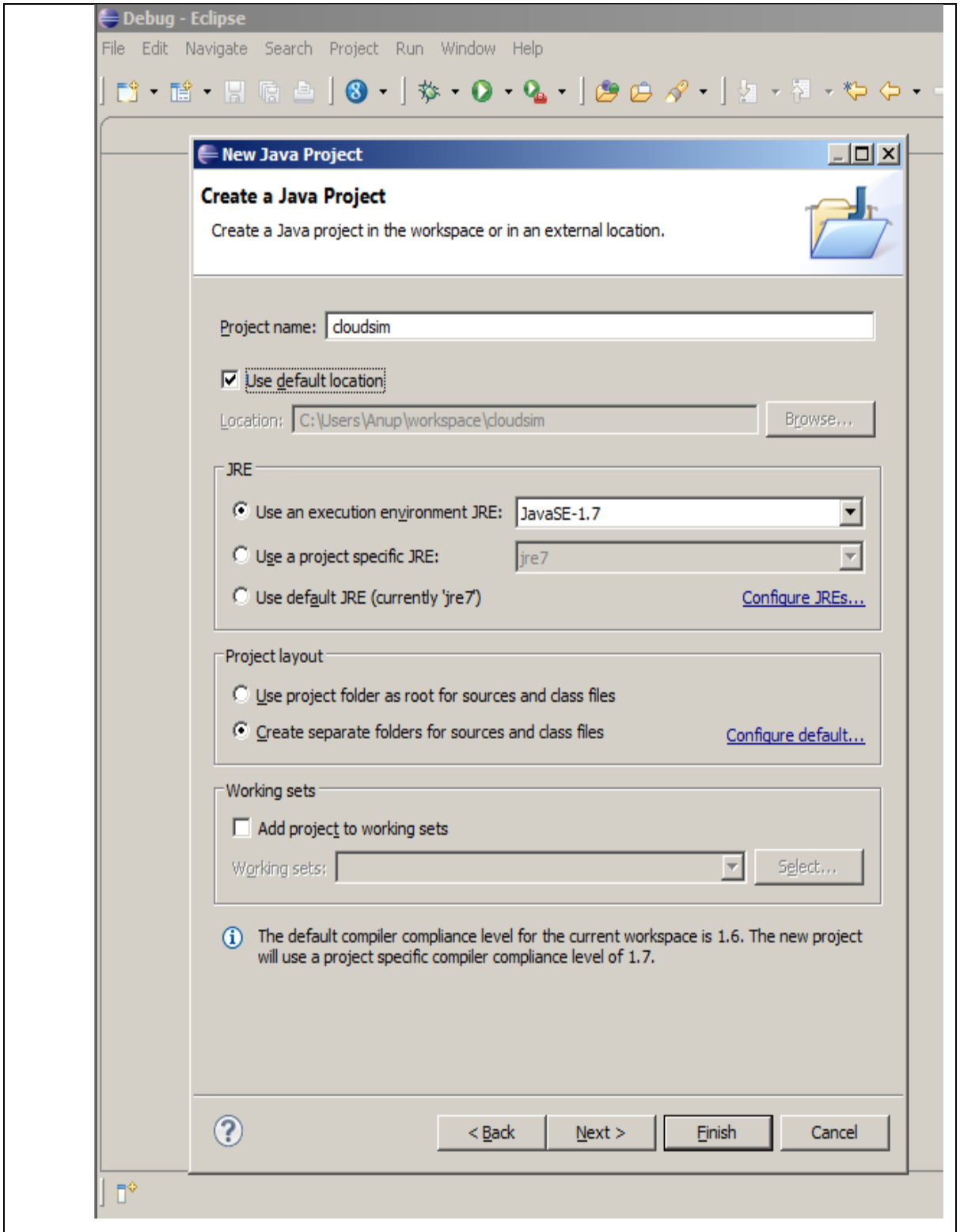
2. Now in Eclipse go to menu File-> New -> Project.

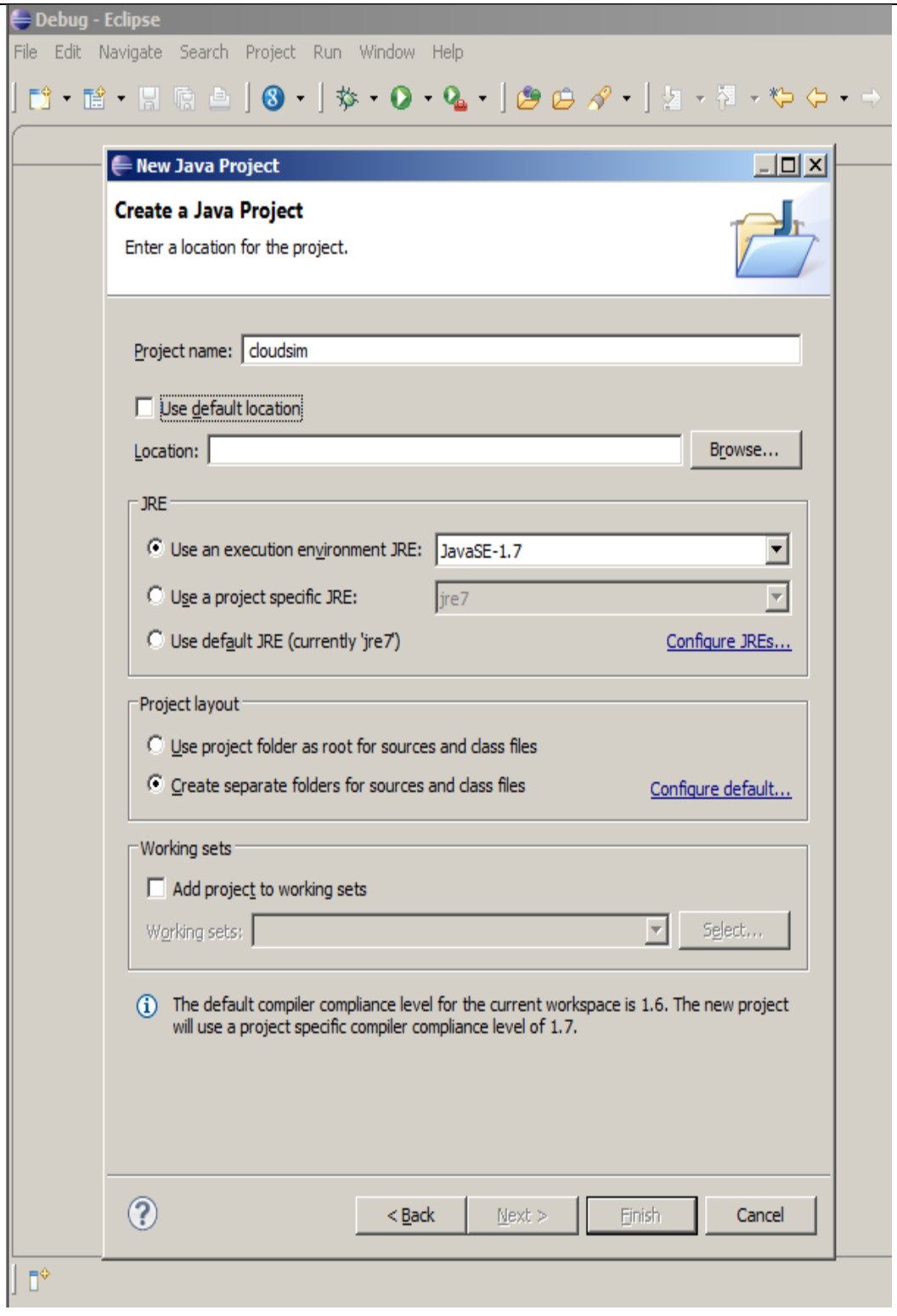


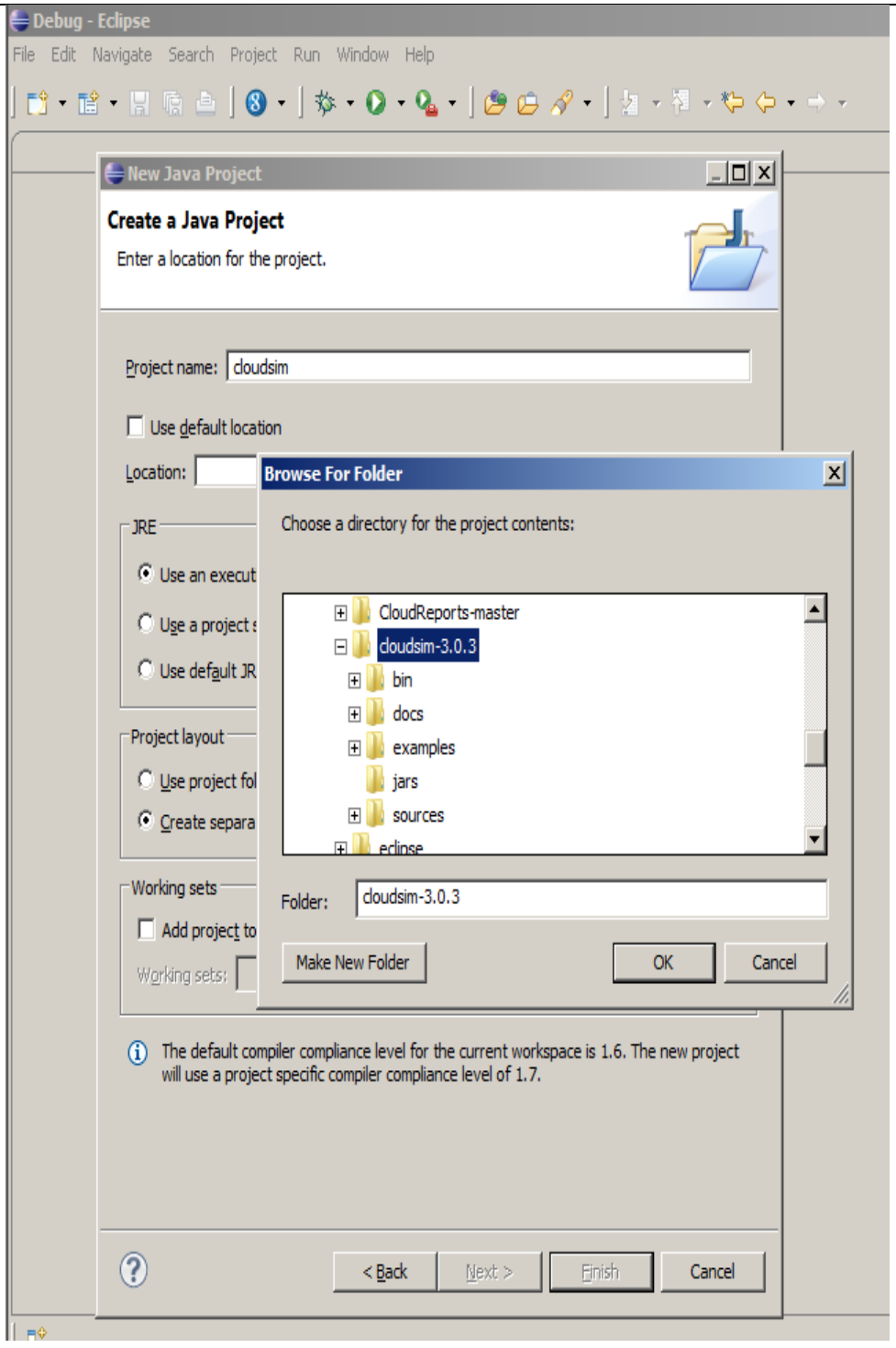
3. Now a project wizard will be opened. From this wizard choose ‘Java Project’ option and click next.

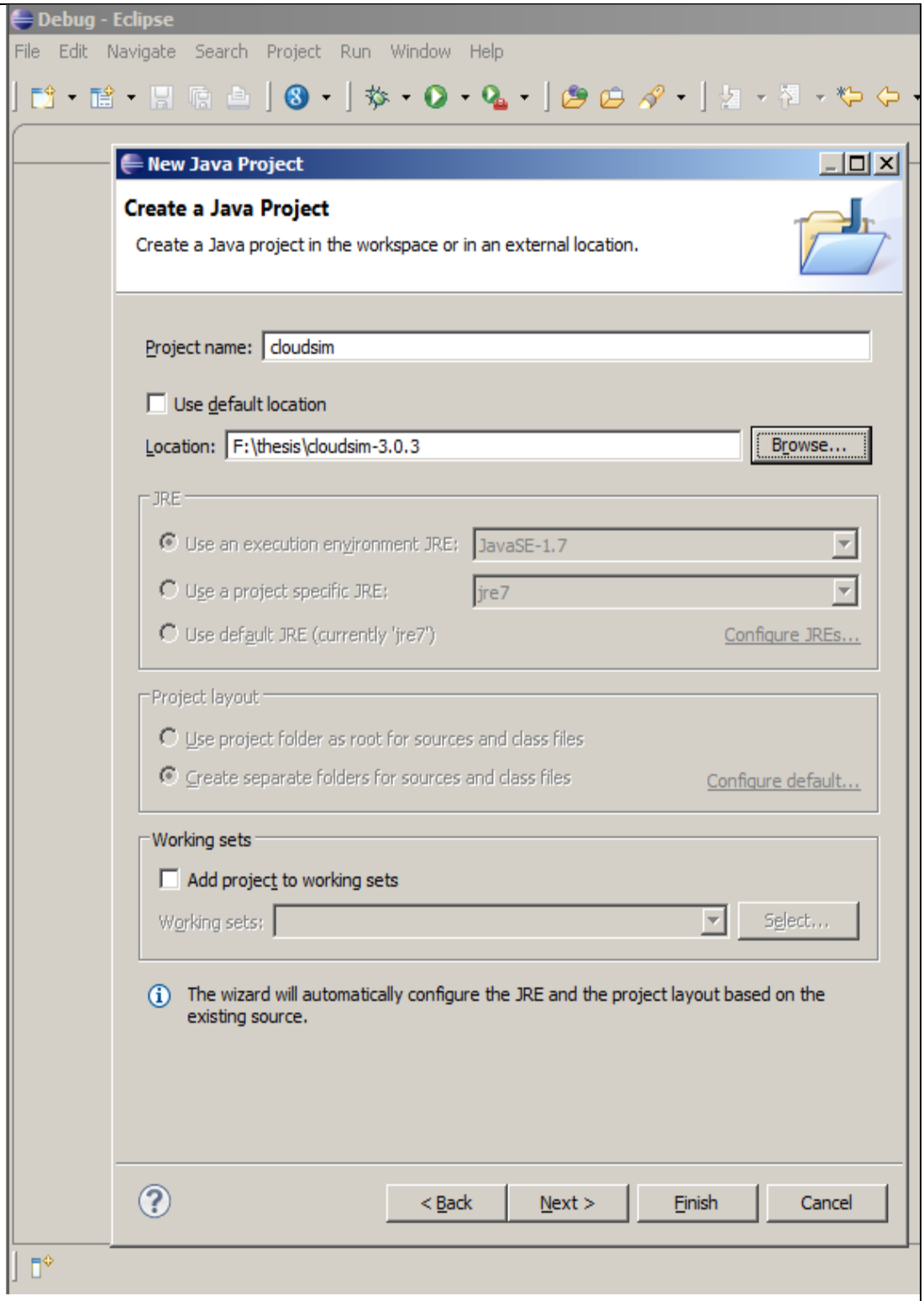


4. Now a detailed new project window will be opened, here you provide project name and the path where you have unzipped the CloudSim project source code or you can put following details as specified:
    a. **Project Name**: CloudSim.
    b. Unselect the 'Use default location' option and then click on browse to open the path where you have unzipped the cloudsim project and finally click Next to set project settings.

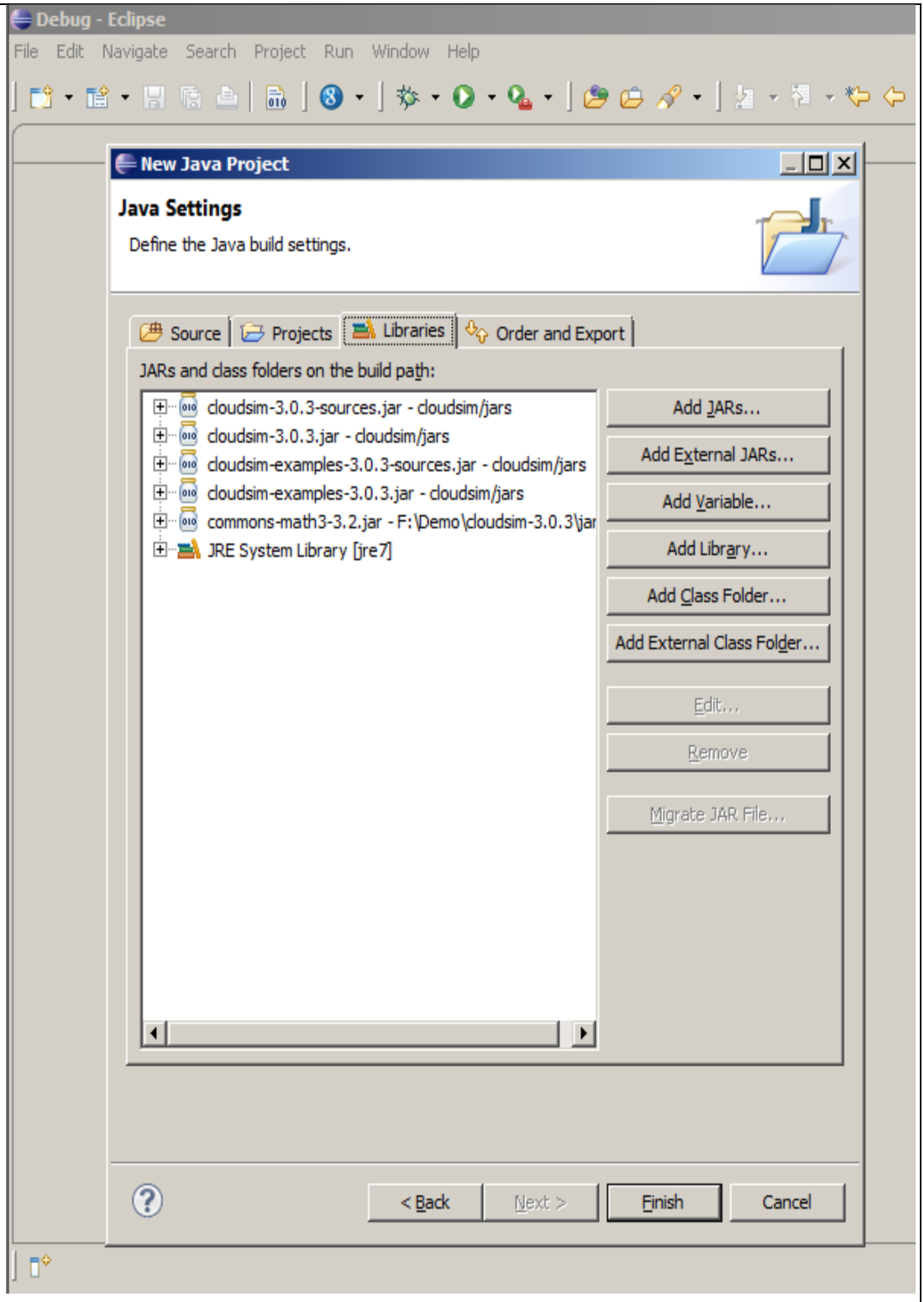

Debug - Eclipse
File Edit Navigate Search Project Run Window Help
New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: cloudsim
Use default location
Location: C:\Users\Anup\workspace\cloudsim
Browse...
JRE
Use an execution environment JRE: JavaSE-1.7
Use a project specific JRE: jre7
Use default JRE (currently 'jre7')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
The default compiler compliance level for the current workspace is 1.6. The new project will use a project specific compiler compliance level of 1.7.
< Back
Next >
Finish
Cancel


Debug - Eclipse
File Edit Navigate Search Project Run Window Help
New Java Project
Create a Java Project
Enter a location for the project.
Project name: cloudsim
Use default location
Location:
Browse...
JRE
Use an execution environment JRE: JavaSE-1.7
Use a project specific JRE: jre7
Use default JRE (currently 'jre7')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
The default compiler compliance level for the current workspace is 1.6. The new project will use a project specific compiler compliance level of 1.7.
< Back
Next >
Finish
Cancel

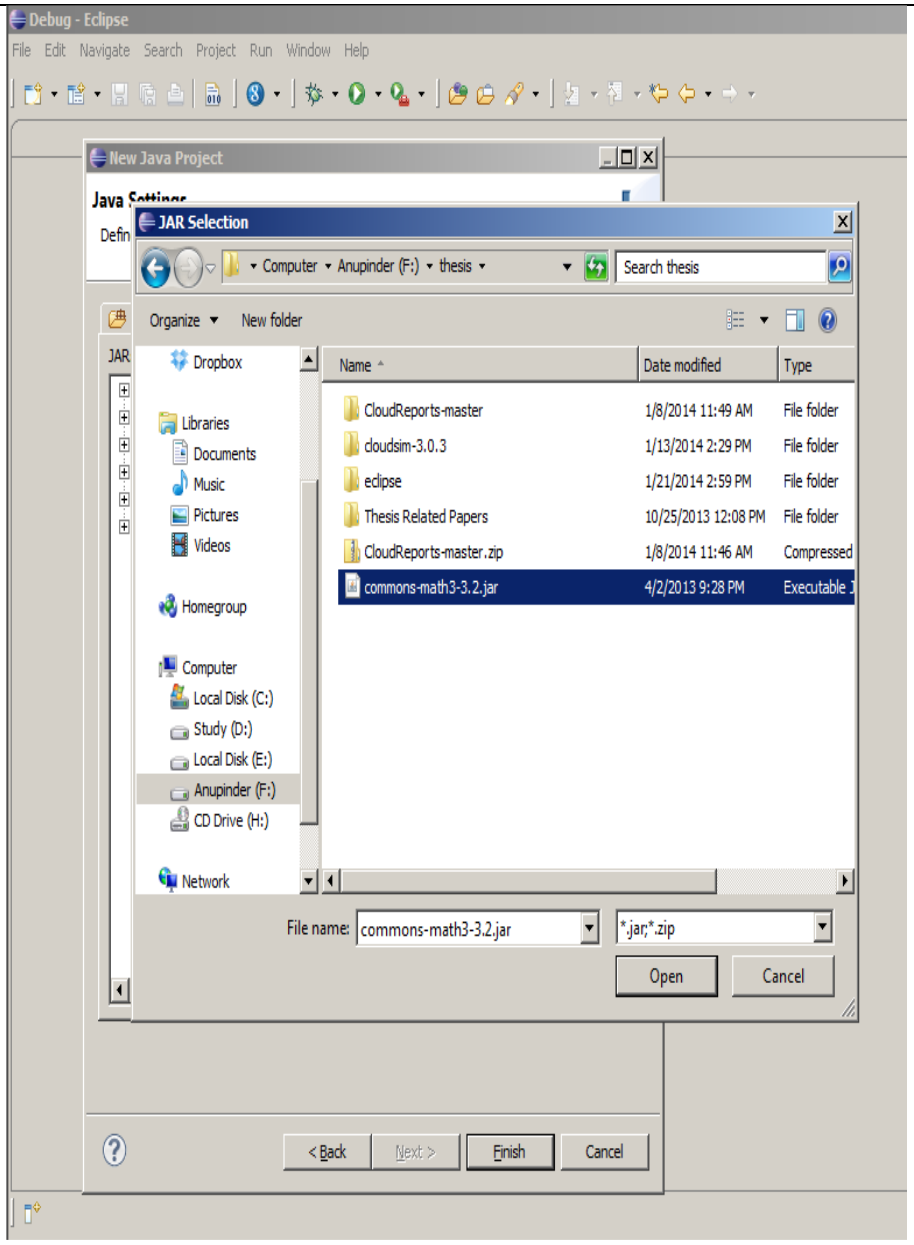

Debug - Eclipse
File Edit Navigate Search Project Run Window Help
New Java Project
Create a Java Project
Enter a location for the project.
Project name: cloudsim
Use default location
Location:
JRE
Use an execut
Use a project s
Use default JR
Project layout
Use project fol
Create separa
Working sets
Add project to
Working sets:
Browse For Folder
Choose a directory for the project contents:
CloudReports-master
cloudsim-3.0.3
bin
docs
examples
jars
sources
eclipse
Folder: cloudsim-3.0.3
Make New Folder
OK
Cancel
The default compiler compliance level for the current workspace is 1.6. The new project will use a project specific compiler compliance level of 1.7.
< Back
Next >
Finish
Cancel



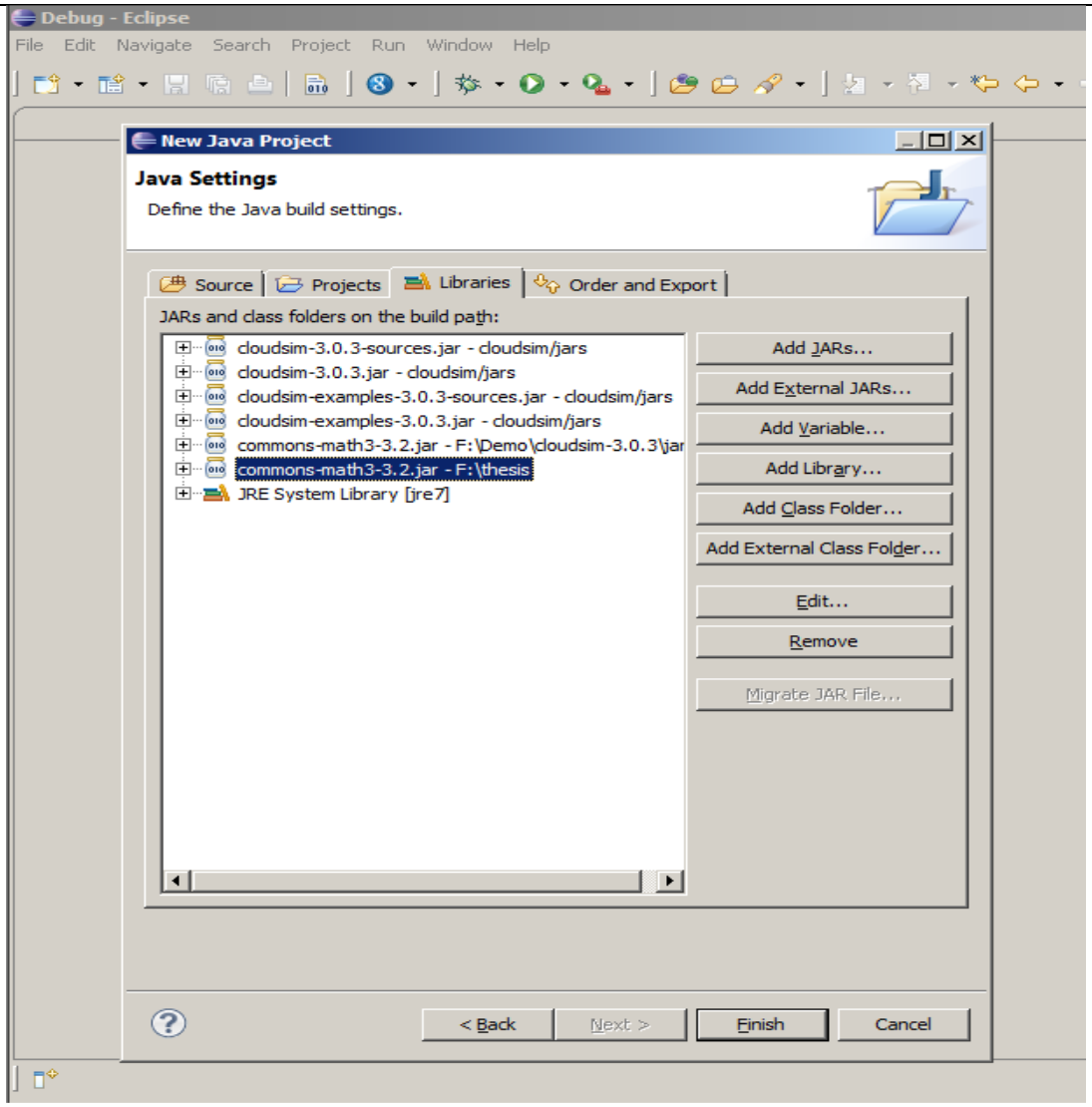
c) Now open ‘Libraries’ tab and click on add external jar (Math common jar will be included in project from this step)



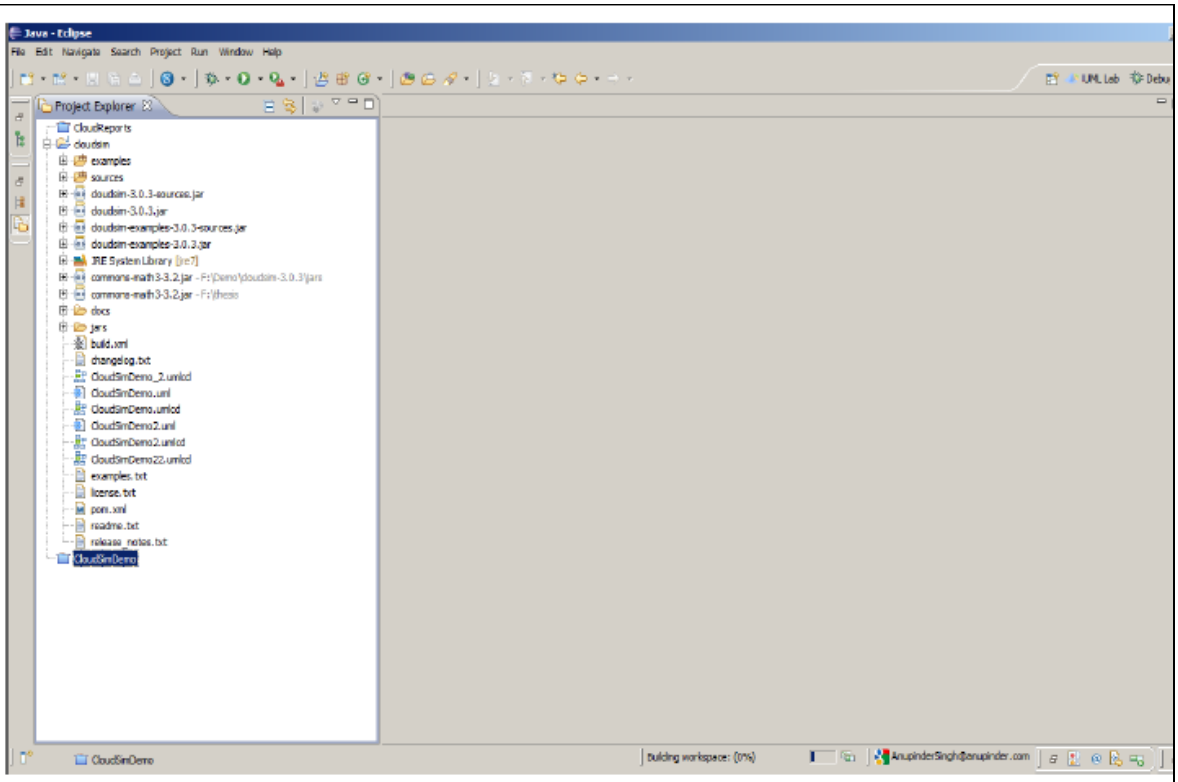
d) Open the path where you have unzipped the math common binaries and select ‘Commonmath 3.3.2.jar’ and click on open.



e. Ensure external jar that you opened in previous step is displayed in list and then click on 'Finish' (your system may take 2-3 minutes to configure the project)



5. Once the project is configured you can open the project explorer and start exploring the cloudsim project. Also for the first time eclipse automatically start building the workspace for
newly configured cloudsim project, which may take some time depending on the configuration
of computer system. Following is the final screen which you will see after cloudsim is configured.



6. Create the program under Examples package.

Output:

Example output of the program:

```
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        400        0.1        400.1
```

```
*****Datacenter: Datacenter_0*****
User id     Debt
3           35.6
**********************************

        CloudSimExample1 finished!
```

Program:

```
//one datacenter, one vm, one cloudlet

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

/**
 * A simple example showing how to create a datacenter with one host and run one
 * cloudlet on it.
 */
public class CloudSimExample1 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example.
	 *
	 * @param args the args
	 */
	@SuppressWarnings("unused")
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample1...");

		try {
```

```
                    // First step: Initialize the CloudSim package. It should be
called
                    // before creating any entities.
                    int num_user = 1; // number of cloud users
                    Calendar calendar = Calendar.getInstance();
                    boolean trace_flag = false; // mean trace events

                    // Initialize the CloudSim library
                    CloudSim.init(num_user, calendar, trace_flag);

                    // Second step: Create Datacenters
                    // Datacenters are the resource providers in CloudSim. We
need at
                    // list one of them to run a CloudSim simulation
                    Datacenter datacenter0 = createDatacenter("Datacenter_0");

                    // Third step: Create Broker
                    DatacenterBroker broker = createBroker();
                    int brokerId = broker.getId();

                    // Fourth step: Create one virtual machine
                    vmlist = new ArrayList<Vm>();

                    // VM description
                    int vmid = 0;
                    int mips = 1000;
                    long size = 10000; // image size (MB)
                    int ram = 512; // vm memory (MB)
                    long bw = 1000;
                    int pesNumber = 1; // number of cpus
                    String vmm = "Xen"; // VMM name

                    // create VM
                    Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());

                    // add the VM to the vmList
                    vmlist.add(vm);

                    // submit vm list to the broker
                    broker.submitVmList(vmlist);

                    // Fifth step: Create one Cloudlet
                    cloudletList = new ArrayList<Cloudlet>();

                    int id = 0;
                    long length = 400000;
                    long fileSize = 300;
                    long outputSize = 300;
                    UtilizationModel utilizationModel = new
UtilizationModelFull();

                    Cloudlet cloudlet = new Cloudlet(id, length, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet.setUserId(brokerId);
                    cloudlet.setVmId(vmid);

                    // add the cloudlet to the list
                    cloudletList.add(cloudlet);

                    // submit cloudlet list to the broker
                    broker.submitCloudletList(cloudletList);
```

```
                // Sixth step: Starts the simulation
                CloudSim.startSimulation();

                CloudSim.stopSimulation();

                //Final step: Print results when simulation is over
                List<Cloudlet> newList = broker.getCloudletReceivedList();
                printCloudletList(newList);

                Log.printLine("CloudSimExample1 finished!");
            } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
            }
        }

        /**
         * Creates the datacenter.
         *
         * @param name the name
         *
         * @return the datacenter
         */
        private static Datacenter createDatacenter(String name) {

            // Here are the steps needed to create a PowerDatacenter:
            // 1. We need to create a list to store
            // our machine
            List<Host> hostList = new ArrayList<Host>();

            // 2. A Machine contains one or more PEs or CPUs/Cores.
            // In this example, it will have only one core.
            List<Pe> peList = new ArrayList<Pe>();

            int mips = 1000;

            // 3. Create PEs and add these into a list.
            peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to
store Pe id and MIPS Rating

            // 4. Create Host with its id and list of PEs and add them to the
list
            // of machines
            int hostId = 0;
            int ram = 2048; // host memory (MB)
            long storage = 1000000; // host storage
            int bw = 10000;

            hostList.add(
                new Host(
                    hostId,
                    new RamProvisionerSimple(ram),
                    new BwProvisionerSimple(bw),
                    storage,
                    peList,
                    new VmSchedulerTimeShared(peList)
                )
            ); // This is our machine

            // 5. Create a DatacenterCharacteristics object that stores the
            // properties of a data center: architecture, OS, list of
```

```
            // Machines, allocation policy: time- or space-shared, time zone
            // and its price (G$/Pe time unit).
            String arch = "x86"; // system architecture
            String os = "Linux"; // operating system
            String vmm = "Xen";
            double time_zone = 10.0; // time zone this resource located
            double cost = 3.0; // the cost of using processing in this
resource
            double costPerMem = 0.05; // the cost of using memory in this
resource
            double costPerStorage = 0.001; // the cost of using storage in
this
                                                                  // resource
            double costPerBw = 0.0; // the cost of using bw in this resource
            LinkedList<Storage> storageList = new LinkedList<Storage>(); // we
are not adding SAN

      // devices by now

            DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
                        arch, os, vmm, hostList, time_zone, cost, costPerMem,
                        costPerStorage, costPerBw);

            // 6. Finally, we need to create a PowerDatacenter object.
            Datacenter datacenter = null;
            try {
                  datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }

            return datacenter;
      }

      // We strongly encourage users to develop their own broker policies, to
      // submit vms and cloudlets according
      // to the specific rules of the simulated scenario
      /**
       * Creates the broker.
       *
       * @return the datacenter broker
       */
      private static DatacenterBroker createBroker() {
            DatacenterBroker broker = null;
            try {
                  broker = new DatacenterBroker("Broker");
            } catch (Exception e) {
                  e.printStackTrace();
                  return null;
            }
            return broker;
      }

      /**
       * Prints the Cloudlet objects.
       *
       * @param list list of Cloudlets
       */
      private static void printCloudletList(List<Cloudlet> list) {
            int size = list.size();
```

```
            Cloudlet cloudlet;

            String indent = "    ";
            Log.printLine();
            Log.printLine("========== OUTPUT ==========");
            Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                        + "Data center ID" + indent + "VM ID" + indent +
"Time" + indent
                        + "Start Time" + indent + "Finish Time");

            DecimalFormat dft = new DecimalFormat("###.##");
            for (int i = 0; i < size; i++) {
                  cloudlet = list.get(i);
                  Log.print(indent + cloudlet.getCloudletId() + indent +
indent);

                  if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                        Log.print("SUCCESS");

                        Log.printLine(indent + indent +
cloudlet.getResourceId()
                                    + indent + indent + indent +
cloudlet.getVmId()
                                    + indent + indent
                                    +
dft.format(cloudlet.getActualCPUTime()) + indent
                                    + indent +
dft.format(cloudlet.getExecStartTime())
                                    + indent + indent
                                    + dft.format(cloudlet.getFinishTime()));
                  }
            }
      }

}
```

Session 4

Problem Statement:

Demonstrates how to create a datacenter with one host, two virtual machines and run two cloudlets on it. Both virtual machine (vm1,vm2) has same machine configuration.

Student Learning Outcomes:

Simulate a Data Center with one host, two VM's and to run one Cloudlet using CloudSim.

Theoretical Description:

Requirements:

Requirements:

- Eclipse IDE for java developers:
- CloudSim Project code:
- One external requirement of cloudsim i.e. common jar package of math

Procedure:

1. Open **eclipse.exe** from eclipse folder:
2. Now in Eclipse go to menu File-> New -> Project.
3. Now a project wizard will be opened. From this wizard choose 'Java Project' option and click next.
4. Now a detailed new project window will be opened, here you provide project name and the path where you have unzipped the CloudSim project source code or you can put following details as specified:

   a. **Project Name**: CloudSim.
   b. Unselect the 'Use default location' option and then click on browse to open the path where you have unzipped the cloudsim project and finally click Next to set project settings.

c) Now open ‘Libraries’ tab and click on add external jar (Math common jar will be included in project from this step)

d) Open the path where you have unzipped the math common binaries and select ‘Commonmath 3.3.2.jar’ and click on open.
e. Ensure external jar that you opened in previous step is displayed in list and then click on ‘Finish’ (your system may take 2-3 minutes to configure the project)

5. Once the project is configured you can open the project explorer and start exploring the
cloudsim project. Also for the first time eclipse automatically start building the workspace for newly configured cloudsim project, which may take some time depending on the configuration of computer system. Following is the final screen which you will see after cloudsim is configured.
6. Create the program under Examples package.

Output:

Example output of the program:

```
Starting CloudSimExample2...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
1000.1: Broker: Cloudlet 0 received
1000.1: Broker: Cloudlet 1 received
1000.1: Broker: All Cloudlets executed. Finishing...
1000.1: Broker: Destroying VM #0
1000.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
   0        SUCCESS        2          0        1000        0.1        1000.1
   1        SUCCESS        2          1        1000        0.1        1000.1
```

```
*****Datacenter: Datacenter_0*****
User id          Debt
3                71.2
************************************

          CloudSimExample2 finished!
```

Program:

```
/*
 * Title:        CloudSim Toolkit
 * Description:  CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation
 *               of Clouds
 * Licence:      GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia
 */

package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;



/**
 * A simple example showing how to create
 * a datacenter with one host and run two
 * cloudlets on it. The cloudlets run in
 * VMs with the same MIPS requirements.
```

```
 * The cloudlets will take the same time to
 * complete the execution.
 */
public class CloudSimExample2 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example
	 */
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample2...");

		try {
		  // First step: Initialize the CloudSim package. It should be called
			// before creating any entities.
			int num_user = 1;   // number of cloud users
			Calendar calendar = Calendar.getInstance();
			boolean trace_flag = false;  // mean trace events

			// Initialize the CloudSim library
			CloudSim.init(num_user, calendar, trace_flag);

			// Second step: Create Datacenters
			//Datacenters are the resource providers in CloudSim. We need at
list one of them to run a CloudSim simulation
			Datacenter datacenter0 = createDatacenter("Datacenter_0");

			//Third step: Create Broker
			DatacenterBroker broker = createBroker();
			int brokerId = broker.getId();

			//Fourth step: Create one virtual machine
			vmlist = new ArrayList<Vm>();

			//VM description
			int vmid = 0;
			int mips = 250;
			long size = 10000; //image size (MB)
			int ram = 512; //vm memory (MB)
			long bw = 1000;
			int pesNumber = 1; //number of cpus
			String vmm = "Xen"; //VMM name

			//create two VMs
			Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());
```

```
                        //the second VM will have twice the priority of VM1 and so will
receive twice CPU time
                        vmid++;
                        Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());

                        //add the VMs to the vmList
                        vmlist.add(vm1);
                        vmlist.add(vm2);

                        //submit vm list to the broker
                        broker.submitVmList(vmlist);


                        //Fifth step: Create two Cloudlets
                        cloudletList = new ArrayList<Cloudlet>();

                        //Cloudlet properties
                        int id = 0;
                        pesNumber=1;
                        long length = 250000;
                        long fileSize = 300;
                        long outputSize = 300;
                        UtilizationModel utilizationModel = new UtilizationModelFull();

                        Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                        cloudlet1.setUserId(brokerId);

                        id++;
                        Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                        cloudlet2.setUserId(brokerId);

                        //add the cloudlets to the list
                        cloudletList.add(cloudlet1);
                        cloudletList.add(cloudlet2);

                        //submit cloudlet list to the broker
                        broker.submitCloudletList(cloudletList);


                        //bind the cloudlets to the vms. This way, the broker
                        // will submit the bound cloudlets only to the specific VM
                        broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
                        broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());

                        // Sixth step: Starts the simulation
                        CloudSim.startSimulation();
```

```
                        // Final step: Print results when simulation is over
                        List<Cloudlet> newList = broker.getCloudletReceivedList();

                        CloudSim.stopSimulation();

                        printCloudletList(newList);

                        //Print the debt of each user to each datacenter
                        datacenter0.printDebts();

                        Log.printLine("CloudSimExample2 finished!");
                }
                catch (Exception e) {
                    e.printStackTrace();
                    Log.printLine("The simulation has been terminated due to an unexpected
error");
                }
            }

                private static Datacenter createDatacenter(String name){

                // Here are the steps needed to create a PowerDatacenter:
                // 1. We need to create a list to store
                    //    our machine
                    List<Host> hostList = new ArrayList<Host>();

                // 2. A Machine contains one or more PEs or CPUs/Cores.
                    // In this example, it will have only one core.
                    List<Pe> peList = new ArrayList<Pe>();

                    int mips = 1000;

                // 3. Create PEs and add these into a list.
                    peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating

                //4. Create Host with its id and list of PEs and add them to the list of machines
                int hostId=0;
                int ram = 2048; //host memory (MB)
                long storage = 1000000; //host storage
                int bw = 10000;

                hostList.add(
                                    new Host(
                                            hostId,
                                            new RamProvisionerSimple(ram),
                                            new BwProvisionerSimple(bw),
                                            storage,
                                            peList,
                                            new VmSchedulerTimeShared(peList)
                                    )
                            ); // This is our machine
```

```
                // 5. Create a DatacenterCharacteristics object that stores the
                //    properties of a data center: architecture, OS, list of
                //    Machines, allocation policy: time- or space-shared, time zone
                //    and its price (G$/Pe time unit).
                String arch = "x86";      // system architecture
                String os = "Linux";          // operating system
                String vmm = "Xen";
                double time_zone = 10.0;         // time zone this resource located
                double cost = 3.0;              // the cost of using processing in this resource
                double costPerMem = 0.05;                // the cost of using memory in this
resource
                double costPerStorage = 0.001; // the cost of using storage in this resource
                double costPerBw = 0.0;                                 // the cost of using bw in this
resource
                LinkedList<Storage> storageList = new LinkedList<Storage>();        //we are
not adding SAN devices by now

                DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                        arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage,
costPerBw);


                // 6. Finally, we need to create a PowerDatacenter object.
                Datacenter datacenter = null;
                try {
                   datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
                } catch (Exception e) {
                   e.printStackTrace();
                }

                return datacenter;
            }

            //We strongly encourage users to develop their own broker policies, to submit
vms and cloudlets according
            //to the specific rules of the simulated scenario
            private static DatacenterBroker createBroker(){

                DatacenterBroker broker = null;
                try {
                        broker = new DatacenterBroker("Broker");
                } catch (Exception e) {
                        e.printStackTrace();
                        return null;
                }
                return broker;
            }

            /**
```

```
     * Prints the Cloudlet objects
     * @param list  list of Cloudlets
     */
    private static void printCloudletList(List<Cloudlet> list) {
        int size = list.size();
        Cloudlet cloudlet;

        String indent = "    ";
        Log.printLine();
        Log.printLine("========== OUTPUT ==========");
        Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
                "Data center ID" + indent + "VM ID" + indent + "Time" + indent + "Start
Time" + indent + "Finish Time");

        DecimalFormat dft = new DecimalFormat("###.##");
        for (int i = 0; i < size; i++) {
            cloudlet = list.get(i);
            Log.print(indent + cloudlet.getCloudletId() + indent + indent);

            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
                Log.print("SUCCESS");

                    Log.printLine( indent + indent + cloudlet.getResourceId() + indent +
indent + indent + cloudlet.getVmId() +
                        indent + indent + dft.format(cloudlet.getActualCPUTime()) + indent +
indent + dft.format(cloudlet.getExecStartTime())+
                        indent + indent + dft.format(cloudlet.getFinishTime()));
            }
        }

    }
}
```

Session 5

Problem Statement:

Demonstrates how to create a datacenter with two hosts, two virtual machines and run two cloudlets on it. The cloudlets run in VMs with different MIPS requirements. The second VM will have twice the priority of virtual machine one(VM1) and so cloudlet will receive twice CPU time to complete the execution

Requirements:

Requirements:

- Eclipse IDE for java developers:
- CloudSim Project code:
- One external requirement of cloudsim i.e. common jar package of math

Program:

```
/*
 * Title:        CloudSim Toolkit
 * Description:  CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation
 *               of Clouds
 * Licence:      GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia
 */

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
```

```
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;


/**
 * A simple example showing how to create
 * a datacenter with two hosts and run two
 * cloudlets on it. The cloudlets run in
 * VMs with different MIPS requirements.
 * The cloudlets will take different time
 * to complete the execution depending on
 * the requested VM performance.
 */
public class CloudSimExample3 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example
	 */
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample3...");

		try {
			// First step: Initialize the CloudSim package. It should be called
			// before creating any entities.
			int num_user = 1;   // number of cloud users
			Calendar calendar = Calendar.getInstance();
			boolean trace_flag = false;  // mean trace events

			// Initialize the CloudSim library
			CloudSim.init(num_user, calendar, trace_flag);

			// Second step: Create Datacenters
			//Datacenters are the resource providers in CloudSim. We need at
list one of them to run a CloudSim simulation
			@SuppressWarnings("unused")
			Datacenter datacenter0 = createDatacenter("Datacenter_0");

			//Third step: Create Broker
			DatacenterBroker broker = createBroker();
```

```
                int brokerId = broker.getId();

                //Fourth step: Create one virtual machine
                vmlist = new ArrayList<Vm>();

                //VM description
                int vmid = 0;
                int mips = 100;
                long size = 10000; //image size (MB)
                int ram = 2048; //vm memory (MB)
                long bw = 1000;
                int pesNumber = 1; //number of cpus
                String vmm = "Xen"; //VMM name

                //create two VMs
                Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());

                //the second VM will have twice the priority of VM1 and so will
receive twice CPU time
                vmid++;
                Vm vm2 = new Vm(vmid, brokerId, mips * 2, pesNumber, ram,
bw, size, vmm, new CloudletSchedulerTimeShared());

                //add the VMs to the vmList
                vmlist.add(vm1);
                vmlist.add(vm2);

                //submit vm list to the broker
                broker.submitVmList(vmlist);


                //Fifth step: Create two Cloudlets
                cloudletList = new ArrayList<Cloudlet>();

                //Cloudlet properties
                int id = 0;
                long length = 40000;
                long fileSize = 300;
                long outputSize = 300;
                UtilizationModel utilizationModel = new UtilizationModelFull();

                Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                cloudlet1.setUserId(brokerId);
```

```
                id++;
                Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                cloudlet2.setUserId(brokerId);

                //add the cloudlets to the list
                cloudletList.add(cloudlet1);
                cloudletList.add(cloudlet2);

                //submit cloudlet list to the broker
                broker.submitCloudletList(cloudletList);


                //bind the cloudlets to the vms. This way, the broker
                // will submit the bound cloudlets only to the specific VM

        broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());

        broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());

                // Sixth step: Starts the simulation
                CloudSim.startSimulation();


                // Final step: Print results when simulation is over
                List<Cloudlet> newList = broker.getCloudletReceivedList();

                CloudSim.stopSimulation();

        printCloudletList(newList);

                Log.printLine("CloudSimExample3 finished!");
            }
            catch (Exception e) {
                e.printStackTrace();
                Log.printLine("The simulation has been terminated due to an
unexpected error");
            }
        }

        private static Datacenter createDatacenter(String name){

            // Here are the steps needed to create a PowerDatacenter:
            // 1. We need to create a list to store
```

```
                //    our machine
                List<Host> hostList = new ArrayList<Host>();

                // 2. A Machine contains one or more PEs or CPUs/Cores.
                // In this example, it will have only one core.
                List<Pe> peList = new ArrayList<Pe>();

                int mips = 1000;

                // 3. Create PEs and add these into a list.
                peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store
Pe id and MIPS Rating

                //4. Create Hosts with its id and list of PEs and add them to the list of
machines
                int hostId=0;
                int ram = 2048; //host memory (MB)
                long storage = 1000000; //host storage
                int bw = 10000;

                hostList.add(
                        new Host(
                                hostId,
                                new RamProvisionerSimple(ram),
                                new BwProvisionerSimple(bw),
                                storage,
                                peList,
                                new VmSchedulerTimeShared(peList)
                        )
                ); // This is our first machine

                //create another machine in the Data center
                List<Pe> peList2 = new ArrayList<Pe>();

                peList2.add(new Pe(0, new PeProvisionerSimple(mips)));

                hostId++;

                hostList.add(
                        new Host(
                                hostId,
                                new RamProvisionerSimple(ram),
                                new BwProvisionerSimple(bw),
                                storage,
                                peList2,
```

```
                                new VmSchedulerTimeShared(peList2)
                        )
                ); // This is our second machine



                // 5. Create a DatacenterCharacteristics object that stores the
                //    properties of a data center: architecture, OS, list of
                //    Machines, allocation policy: time- or space-shared, time zone
                //    and its price (G$/Pe time unit).
                String arch = "x86";      // system architecture
                String os = "Linux";          // operating system
                String vmm = "Xen";
                double time_zone = 10.0;         // time zone this resource located
                double cost = 3.0;              // the cost of using processing in this resource
                double costPerMem = 0.05;           // the cost of using memory in this
resource
                double costPerStorage = 0.001;      // the cost of using storage in this
resource
                double costPerBw = 0.0;                     // the cost of using bw in this
resource
                LinkedList<Storage> storageList = new LinkedList<Storage>();
        //we are not adding SAN devices by now

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage,
costPerBw);

                // 6. Finally, we need to create a PowerDatacenter object.
                Datacenter datacenter = null;
                try {
                        datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                return datacenter;
        }

        //We strongly encourage users to develop their own broker policies, to submit
vms and cloudlets according
        //to the specific rules of the simulated scenario
        private static DatacenterBroker createBroker(){
```

```
                DatacenterBroker broker = null;
                try {
                        broker = new DatacenterBroker("Broker");
                } catch (Exception e) {
                        e.printStackTrace();
                        return null;
                }
                return broker;
        }

        /**
         * Prints the Cloudlet objects
         * @param list  list of Cloudlets
         */
        private static void printCloudletList(List<Cloudlet> list) {
                int size = list.size();
                Cloudlet cloudlet;

                String indent = "    ";
                Log.printLine();
                Log.printLine("========== OUTPUT ==========");
                Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
                                "Data center ID" + indent + "VM ID" + indent + "Time"
+ indent + "Start Time" + indent + "Finish Time");

                DecimalFormat dft = new DecimalFormat("###.##");
                for (int i = 0; i < size; i++) {
                        cloudlet = list.get(i);
                        Log.print(indent + cloudlet.getCloudletId() + indent + indent);

                        if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
                                Log.print("SUCCESS");

                                Log.printLine( indent + indent + cloudlet.getResourceId()
+ indent + indent + indent + cloudlet.getVmId() +
                                                indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent + indent +
dft.format(cloudlet.getExecStartTime())+
                                                indent + indent +
dft.format(cloudlet.getFinishTime()));
                        }
                }

        }
}
```

Output:

```
Starting CloudSimExample3...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
80.1: Broker: Cloudlet 1 received
160.1: Broker: Cloudlet 0 received
160.1: Broker: All Cloudlets executed. Finishing...
160.1: Broker: Destroying VM #0
160.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
   1        SUCCESS        2          1        80        0.1        80.1
   0        SUCCESS        2          0        160        0.1        160.1
CloudSimExample3 finished!
```

Session 6

Problem Statement:

Demonstrate how to create two datacenters with one host each and run two cloudlets on them.

Requirements:

Requirements:

- Eclipse IDE for java developers:
- CloudSim Project code:
- One external requirement of cloudsim i.e. common jar package of math

Program:

```
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerSpaceShared;

import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;


/**
 * A simple example showing how to create
 * two datacenters with one host each and
 * run two cloudlets on them.
 */
public class CloudSimExample4 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example
	 */
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample4...");
```

```
            try {
                    // First step: Initialize the CloudSim package. It should be
called
                    // before creating any entities.
                    int num_user = 1;   // number of cloud users
                    Calendar calendar = Calendar.getInstance();
                    boolean trace_flag = false;  // mean trace events

                    // Initialize the GridSim library
                    CloudSim.init(num_user, calendar, trace_flag);

                    // Second step: Create Datacenters
                    //Datacenters are the resource providers in CloudSim. We
need at list one of them to run a CloudSim simulation
                    @SuppressWarnings("unused")
                    Datacenter datacenter0 = createDatacenter("Datacenter_0");
                    @SuppressWarnings("unused")
                    Datacenter datacenter1 = createDatacenter("Datacenter_1");

                    //Third step: Create Broker
                    DatacenterBroker broker = createBroker();
                    int brokerId = broker.getId();

                    //Fourth step: Create one virtual machine
                    vmlist = new ArrayList<Vm>();

                    //VM description
                    int vmid = 0;
                    int mips = 250;
                    long size = 10000; //image size (MB)
                    int ram = 512; //vm memory (MB)
                    long bw = 1000;
                    int pesNumber = 1; //number of cpus
                    String vmm = "Xen"; //VMM name

                    //create two VMs
                    Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());

                    vmid++;
                    Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw,
size, vmm, new CloudletSchedulerTimeShared());

                    //add the VMs to the vmList
                    vmlist.add(vm1);
                    vmlist.add(vm2);

                    //submit vm list to the broker
                    broker.submitVmList(vmlist);


                    //Fifth step: Create two Cloudlets
                    cloudletList = new ArrayList<Cloudlet>();

                    //Cloudlet properties
                    int id = 0;
                    long length = 250000;
                    long fileSize = 300;
                    long outputSize = 300;
                    UtilizationModel utilizationModel = new
UtilizationModelFull();
```

```
                    Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet1.setUserId(brokerId);

                    id++;

                    Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
                    cloudlet2.setUserId(brokerId);

                    //add the cloudlets to the list
                    cloudletList.add(cloudlet1);
                    cloudletList.add(cloudlet2);

                    //submit cloudlet list to the broker
                    broker.submitCloudletList(cloudletList);


                    //bind the cloudlets to the vms. This way, the broker
                    // will submit the bound cloudlets only to the specific VM

      broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());

      broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());

                    // Sixth step: Starts the simulation
                    CloudSim.startSimulation();


                    // Final step: Print results when simulation is over
                    List<Cloudlet> newList = broker.getCloudletReceivedList();

                    CloudSim.stopSimulation();

            printCloudletList(newList);

                    Log.printLine("CloudSimExample4 finished!");
            }
            catch (Exception e) {
                    e.printStackTrace();
                    Log.printLine("The simulation has been terminated due to an
unexpected error");
            }
      }

      private static Datacenter createDatacenter(String name){

            // Here are the steps needed to create a PowerDatacenter:
            // 1. We need to create a list to store
            //    our machine
            List<Host> hostList = new ArrayList<Host>();

            // 2. A Machine contains one or more PEs or CPUs/Cores.
            // In this example, it will have only one core.
            List<Pe> peList = new ArrayList<Pe>();

            int mips = 1000;

            // 3. Create PEs and add these into a list.
            peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to
store Pe id and MIPS Rating
```

```
            //4. Create Host with its id and list of PEs and add them to the
list of machines
            int hostId=0;
            int ram = 2048; //host memory (MB)
            long storage = 1000000; //host storage
            int bw = 10000;


            //in this example, the VMAllocatonPolicy in use is SpaceShared. It
means that only one VM
            //is allowed to run on each Pe. As each Host has only one Pe, only
one VM can run on each Host.
            hostList.add(
                  new Host(
                        hostId,
                        new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw),
                        storage,
                        peList,
                        new VmSchedulerSpaceShared(peList)

                  )
            ); // This is our first machine

            // 5. Create a DatacenterCharacteristics object that stores the
            //    properties of a data center: architecture, OS, list of
            //    Machines, allocation policy: time- or space-shared, time
zone
            //    and its price (G$/Pe time unit).
            String arch = "x86";      // system architecture
            String os = "Linux";          // operating system
            String vmm = "Xen";
            double time_zone = 10.0;         // time zone this resource
located
            double cost = 3.0;              // the cost of using processing in
this resource
            double costPerMem = 0.05;        // the cost of using memory in
this resource
            double costPerStorage = 0.001;   // the cost of using storage in
this resource
            double costPerBw = 0.0;                 // the cost of using bw in
this resource
            LinkedList<Storage> storageList = new LinkedList<Storage>();
      //we are not adding SAN devices by now

             DatacenterCharacteristics characteristics = new
DatacenterCharacteristics(
                        arch, os, vmm, hostList, time_zone, cost, costPerMem,
costPerStorage, costPerBw);


            // 6. Finally, we need to create a PowerDatacenter object.
            Datacenter datacenter = null;
            try {
                  datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }

            return datacenter;
```

```
	}

	//We strongly encourage users to develop their own broker policies, to
submit vms and cloudlets according
	//to the specific rules of the simulated scenario
	private static DatacenterBroker createBroker(){

		DatacenterBroker broker = null;
		try {
			broker = new DatacenterBroker("Broker");
		} catch (Exception e) {
			e.printStackTrace();
			return null;
		}
		return broker;
	}

	/**
	 * Prints the Cloudlet objects
	 * @param list  list of Cloudlets
	 */
	private static void printCloudletList(List<Cloudlet> list) {
		int size = list.size();
		Cloudlet cloudlet;

		String indent = "    ";
		Log.printLine();
		Log.printLine("========== OUTPUT ==========");
		Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
				"Data center ID" + indent + "VM ID" + indent + "Time"
+ indent + "Start Time" + indent + "Finish Time");

		DecimalFormat dft = new DecimalFormat("###.##");
		for (int i = 0; i < size; i++) {
			cloudlet = list.get(i);
			Log.print(indent + cloudlet.getCloudletId() + indent +
indent);

			if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
				Log.print("SUCCESS");

				Log.printLine( indent + indent +
cloudlet.getResourceId() + indent + indent + indent + cloudlet.getVmId() +
					indent + indent +
dft.format(cloudlet.getActualCPUTime()) + indent + indent +
dft.format(cloudlet.getExecStartTime())+
					indent + indent +
dft.format(cloudlet.getFinishTime()));
			}
		}

	}
}
```

Output:

Starting CloudSimExample4...
Initialising...
Starting CloudSim version 3.0

```
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Creation of VM #1 failed in Datacenter #2
0.1: Broker: Trying to Create VM #1 in Datacenter_1
0.2: Broker: VM #1 has been created in Datacenter #3, Host #0
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
160.2: Broker: Cloudlet 0 received
160.2: Broker: Cloudlet 1 received
160.2: Broker: All Cloudlets executed. Finishing...
160.2: Broker: Destroying VM #0
160.2: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
   0        SUCCESS        2          0        160        0.2        160.2
   1        SUCCESS        3          1        160        0.2        160.2
CloudSimExample4 finished!
```

Session 7

Problem Statement:

Implement and Evaluate the performance of MapReduce program on word count for different file size.

Student Learning Outcomes:

Execute MapReduce programs on Hadoop and analyze the results.

Theoretical Description:

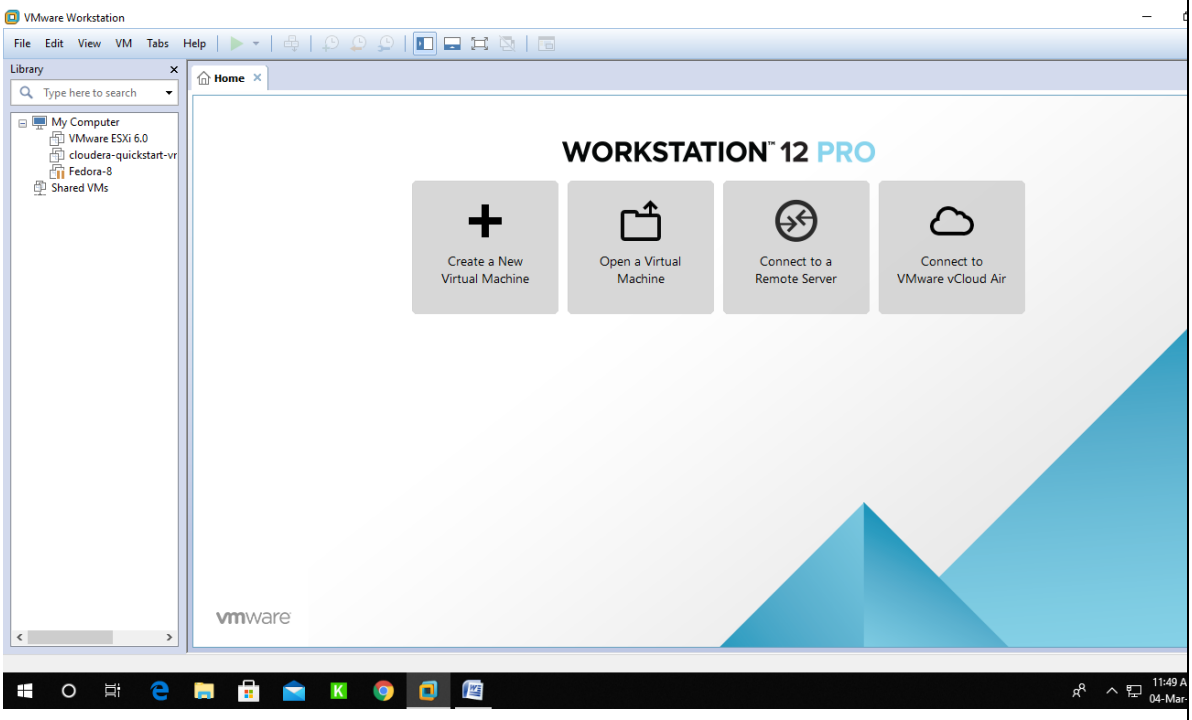
MapReduce: A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored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

Requirements:

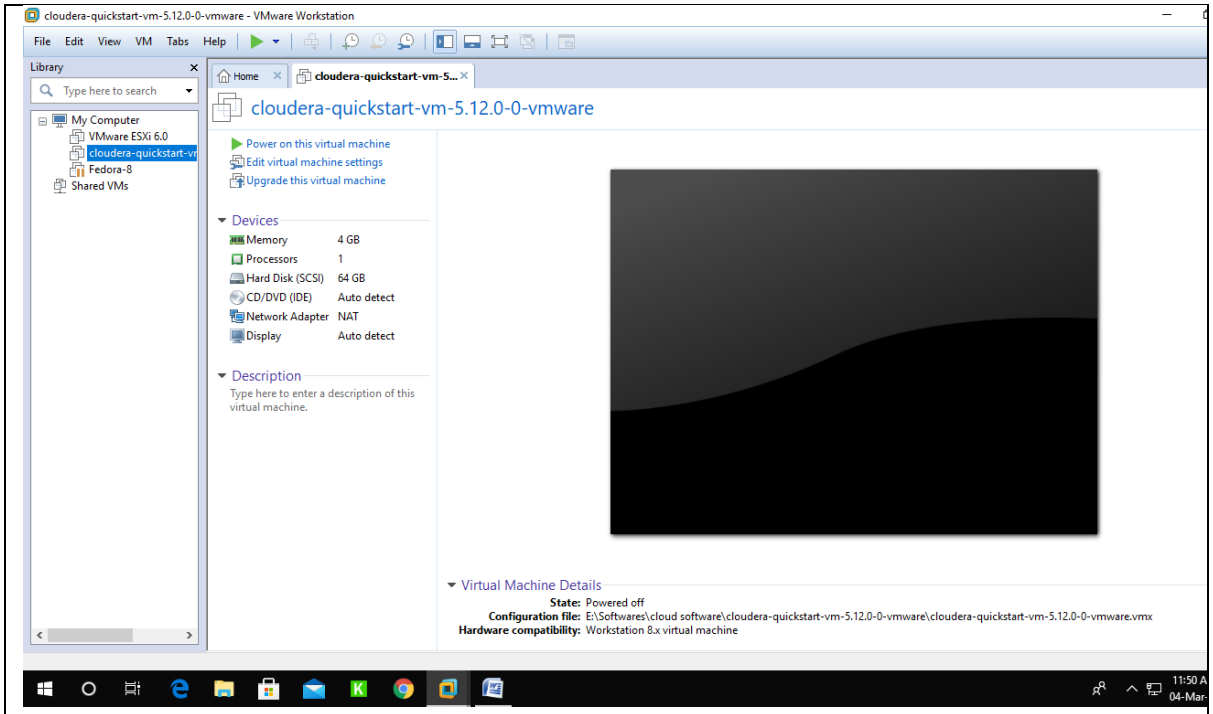
- Physical Computer with minimum of 4GB RAM and other specifications.
- Cloudera, Hadoop.
- JDK, Eclipse.

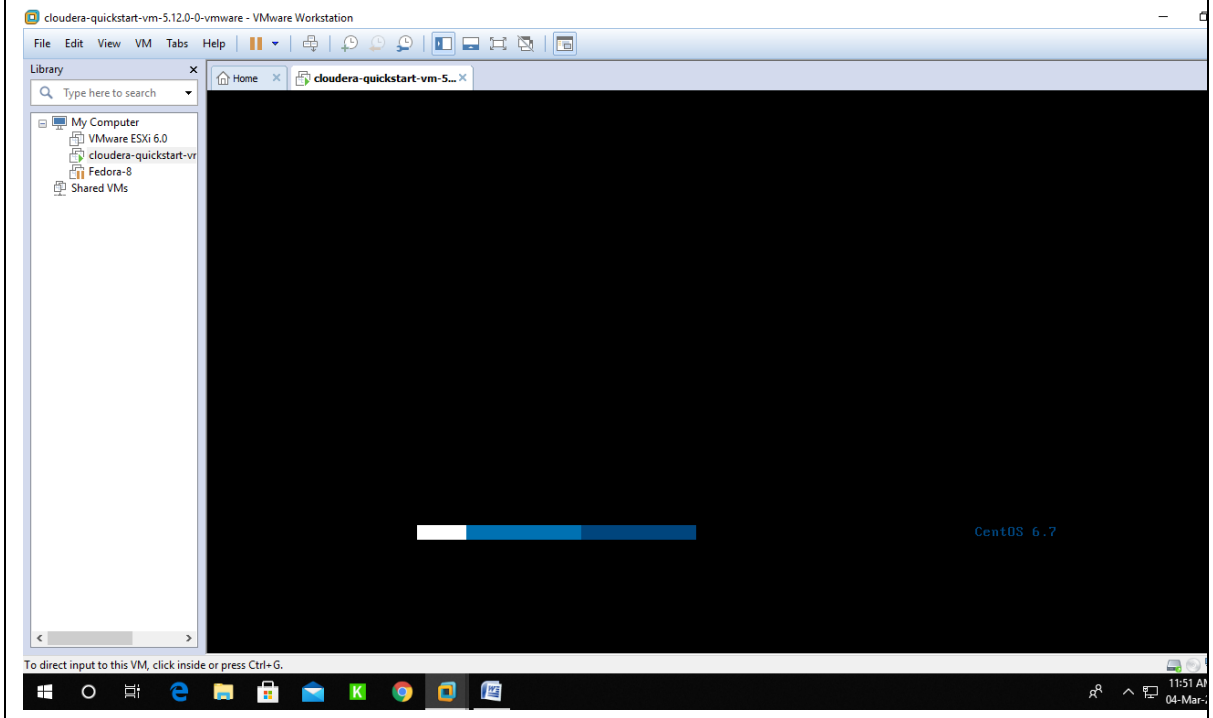
Procedure:

1) Switch on the VMware workstation.

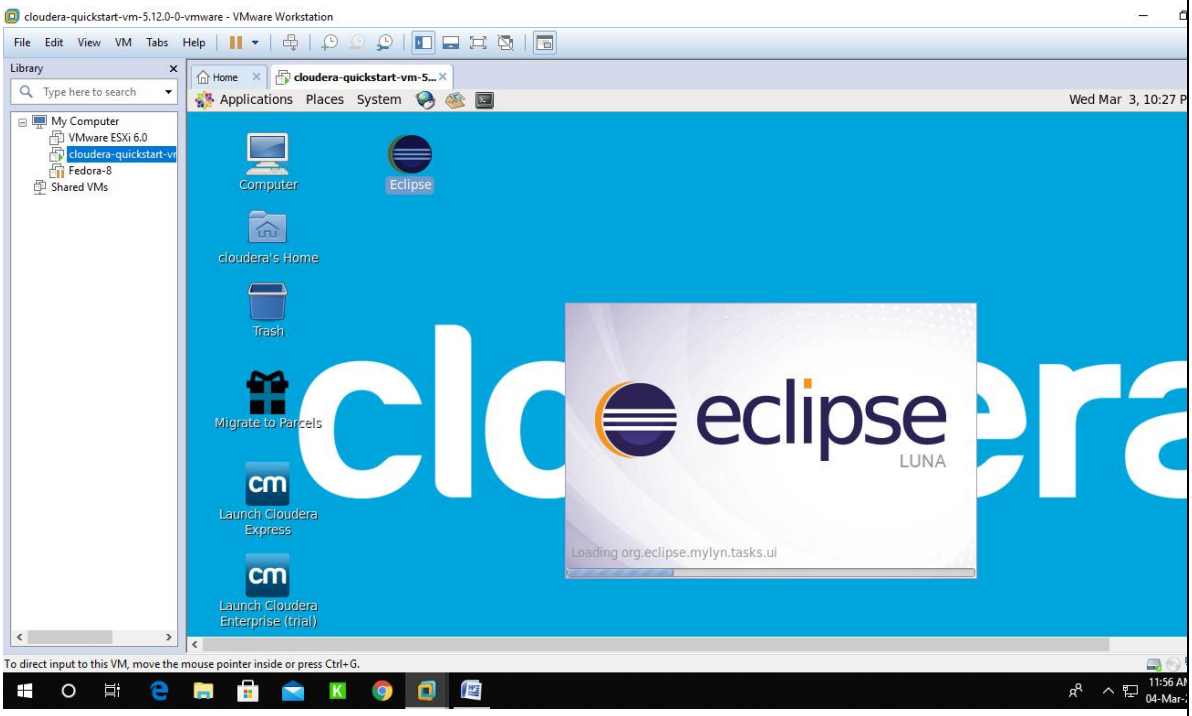


2) Power on the Cloudera.

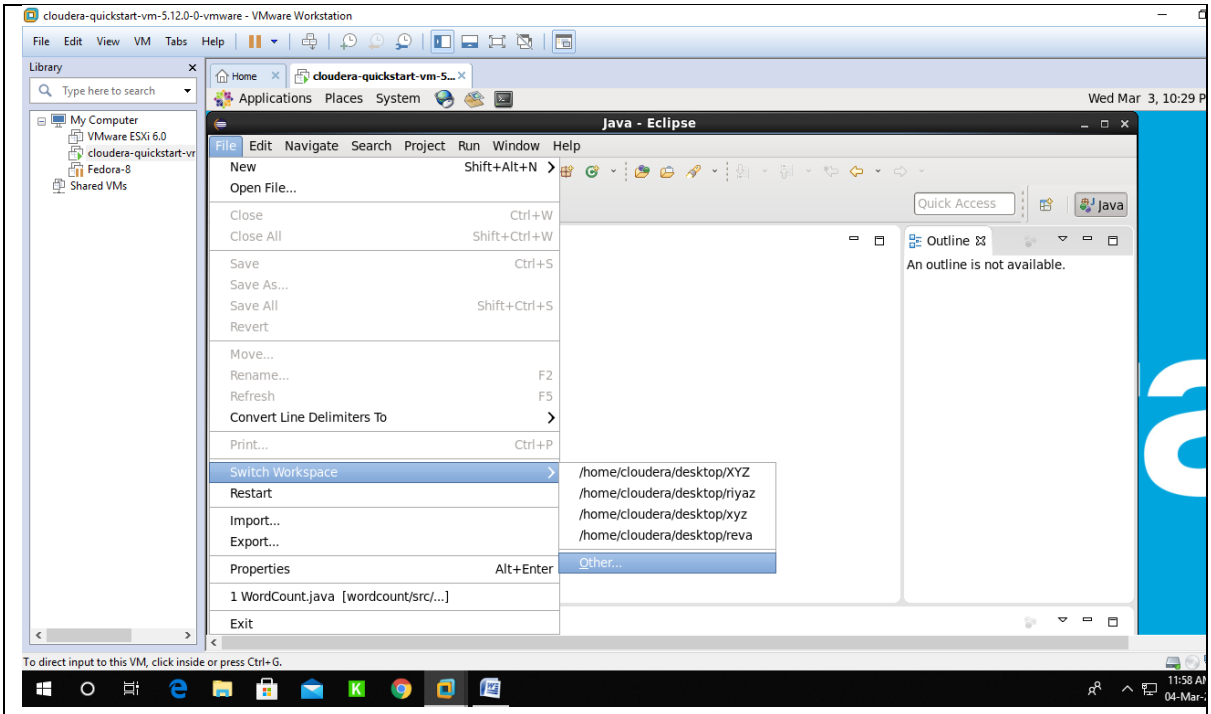

cloudera-quickstart-vm-5.12.0-0-vmware - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
VMware ESXi 6.0
cloudera-quickstart-vr
Fedora-8
Shared VMs
Home
cloudera-quickstart-vm-5...
cloudera-quickstart-vm-5.12.0-0-vmware
Power on this virtual machine
Edit virtual machine settings
Upgrade this virtual machine
Devices
Memory 4 GB
Processors 1
Hard Disk (SCSI) 64 GB
CD/DVD (IDE) Auto detect
Network Adapter NAT
Display Auto detect
Description
Type here to enter a description of this virtual machine.
Virtual Machine Details
State: Powered off
Configuration file: E:\Softwares\cloud software\cloudera-quickstart-vm-5.12.0-0-vmware\cloudera-quickstart-vm-5.12.0-0-vmware.vmx
Hardware compatibility: Workstation 8.x virtual machine


cloudera-quickstart-vm-5.12.0-0-vmware - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
VMware ESXi 6.0
cloudera-quickstart-vr
Fedora-8
Shared VMs
Home
cloudera-quickstart-vm-5...
CentOS 6.7
To direct input to this VM, click inside or press Ctrl+G.

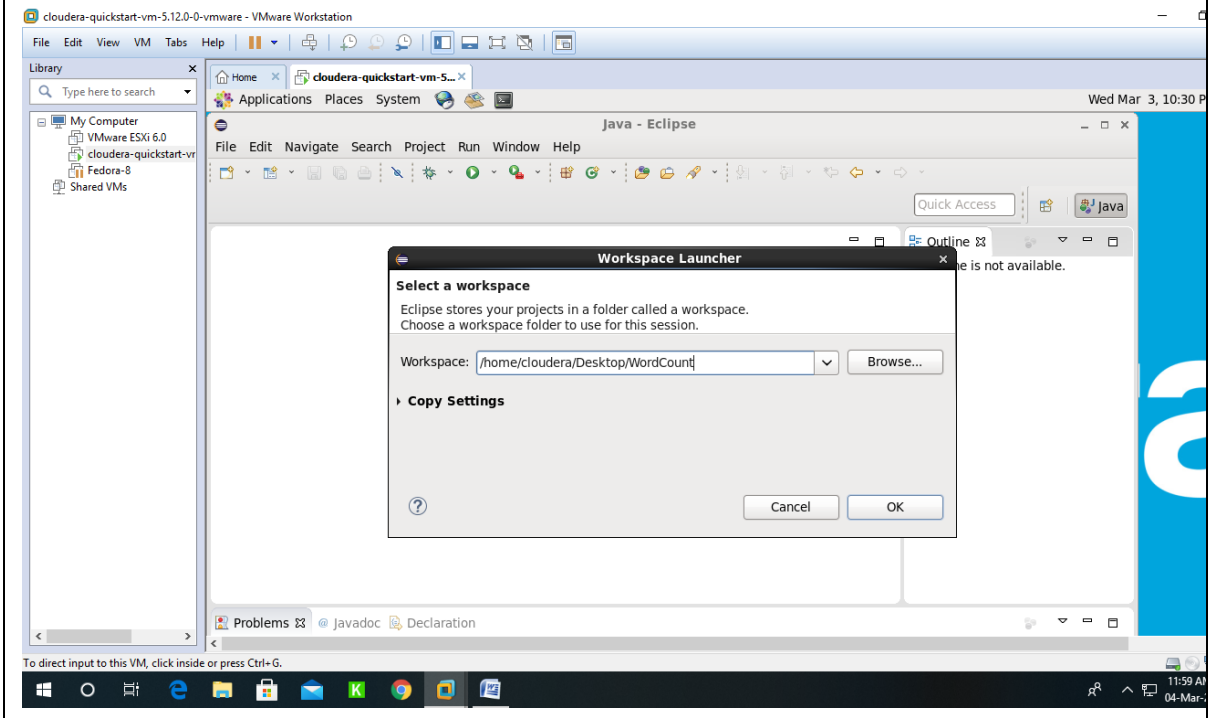
3) Switch on the Eclipse.



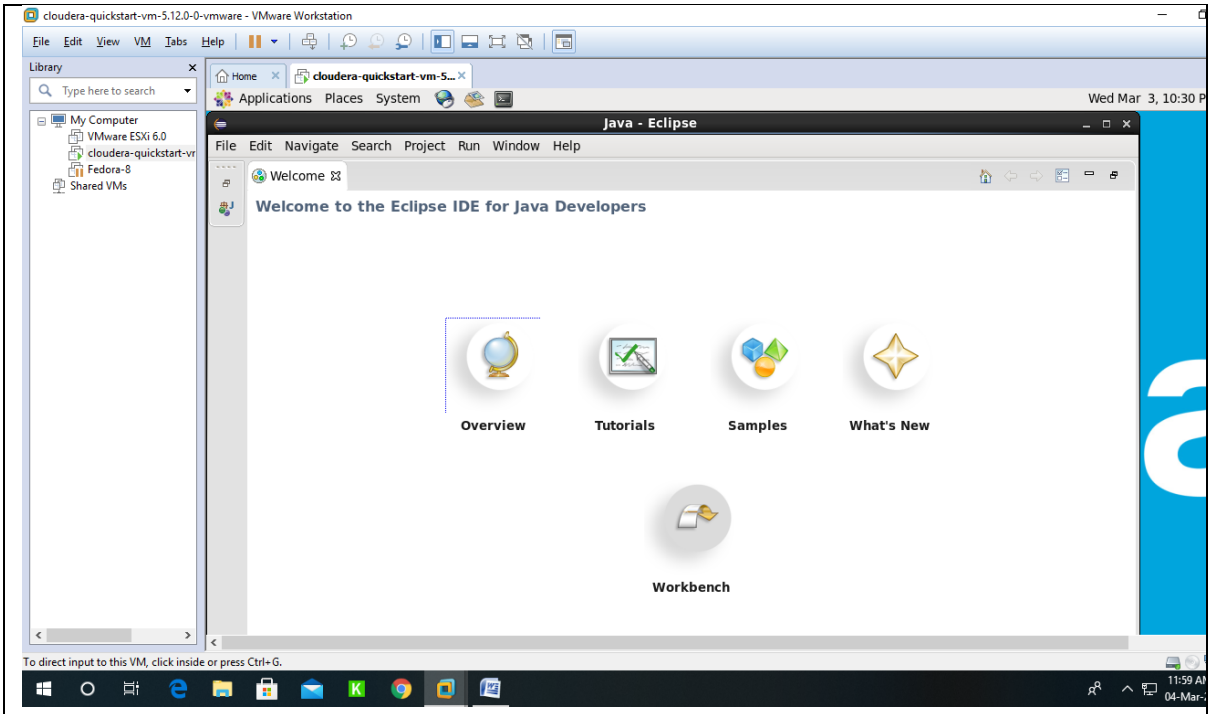
4) Goto File -> Switch workspace - >other.



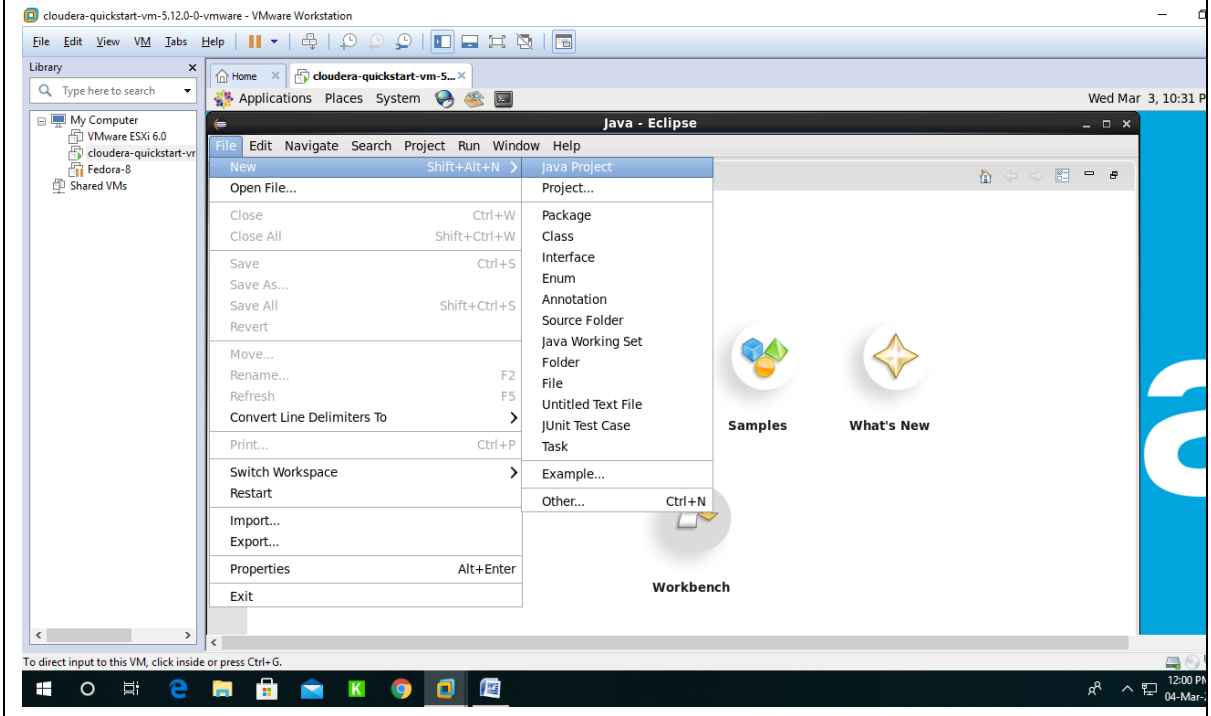
5) Set the path to store the workspace.



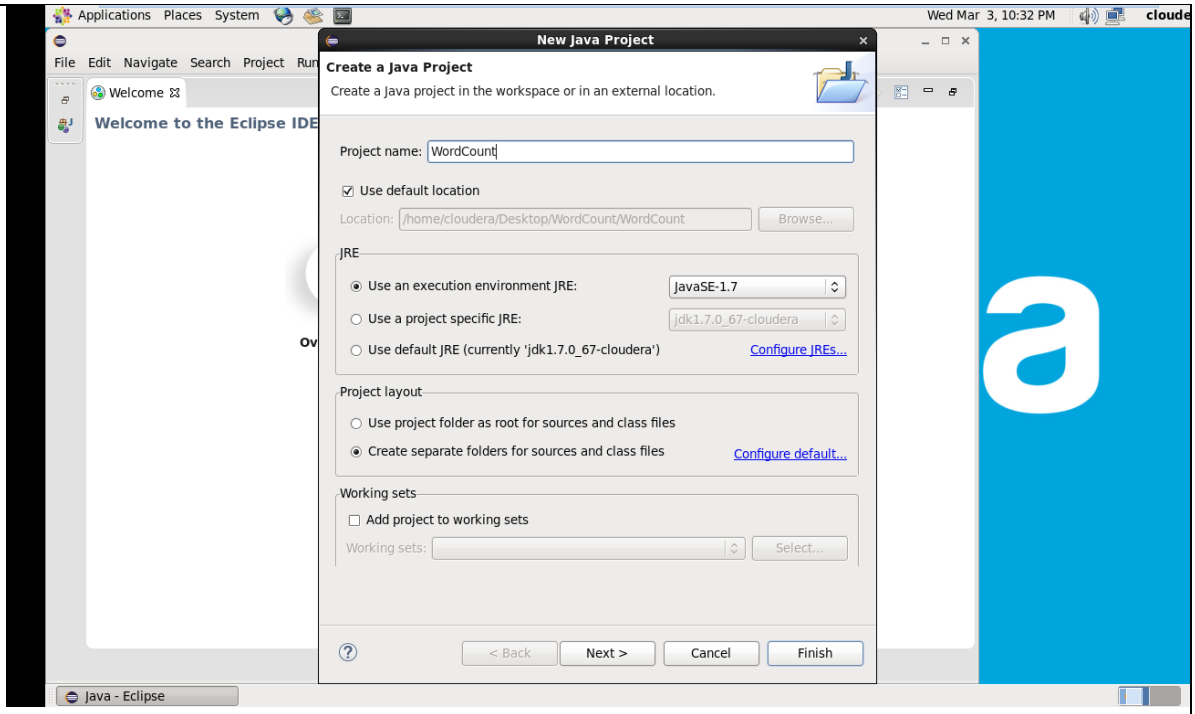
6) Restore the workspace.



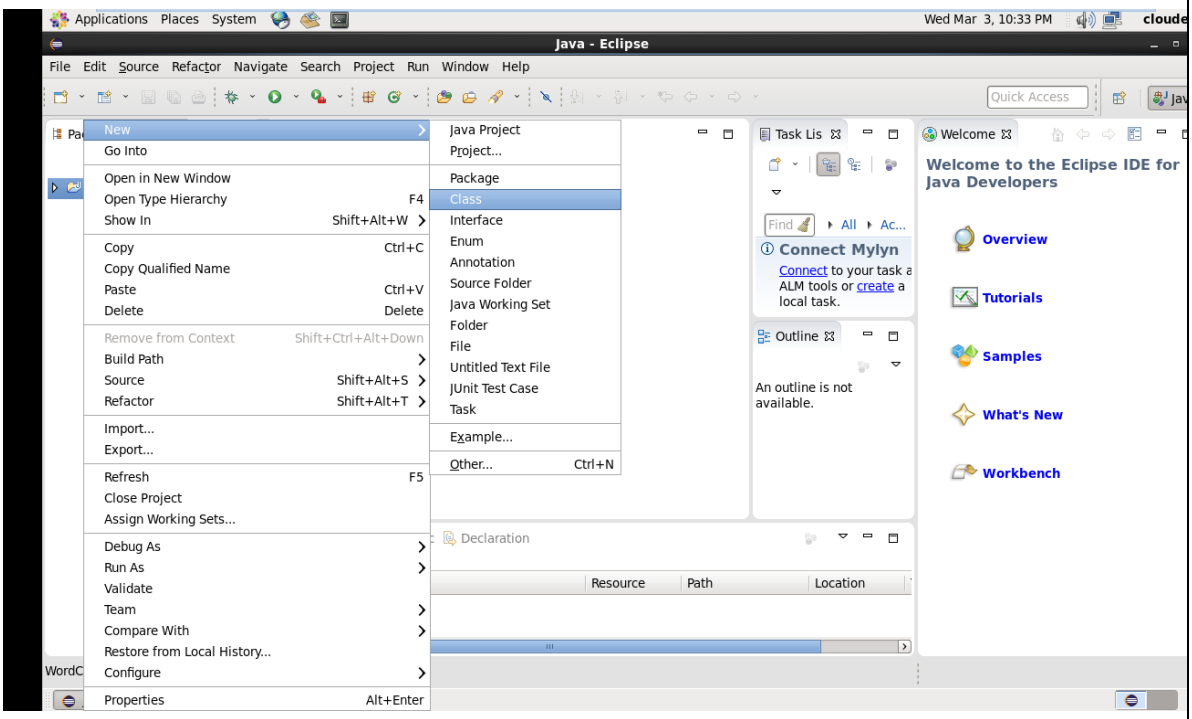
7) Goto File-> New-> Java Project.



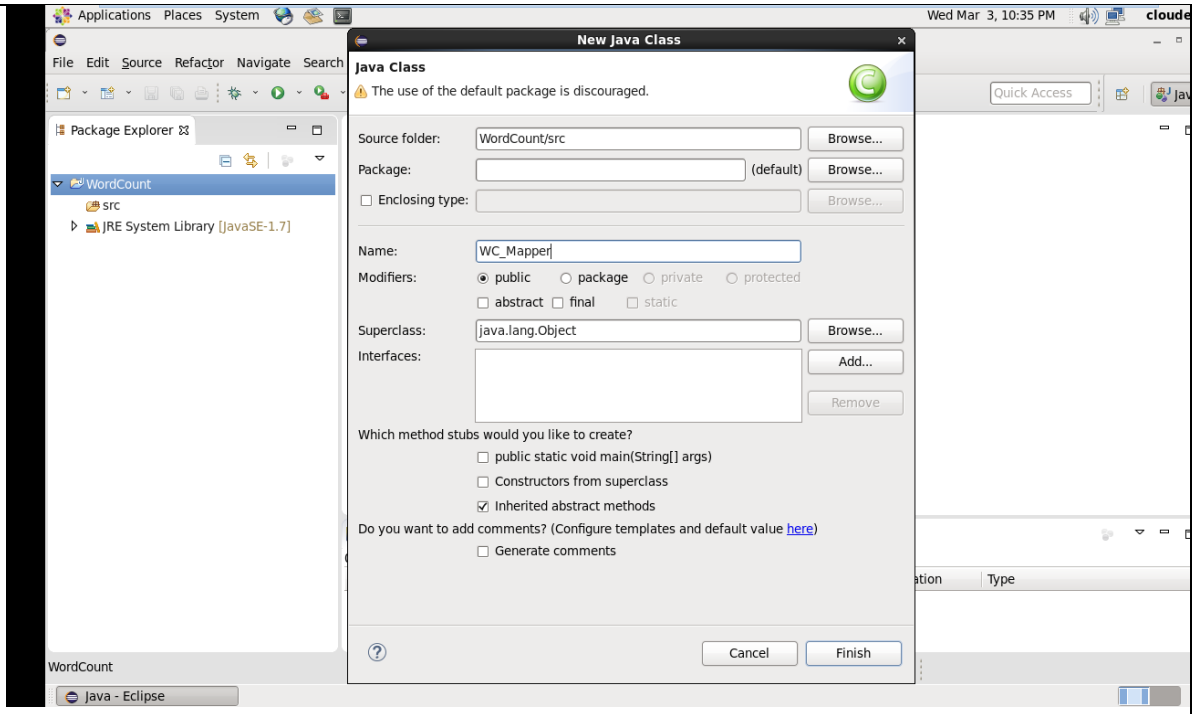
8) Provide the title of the project, preferably as ‘WordCount’



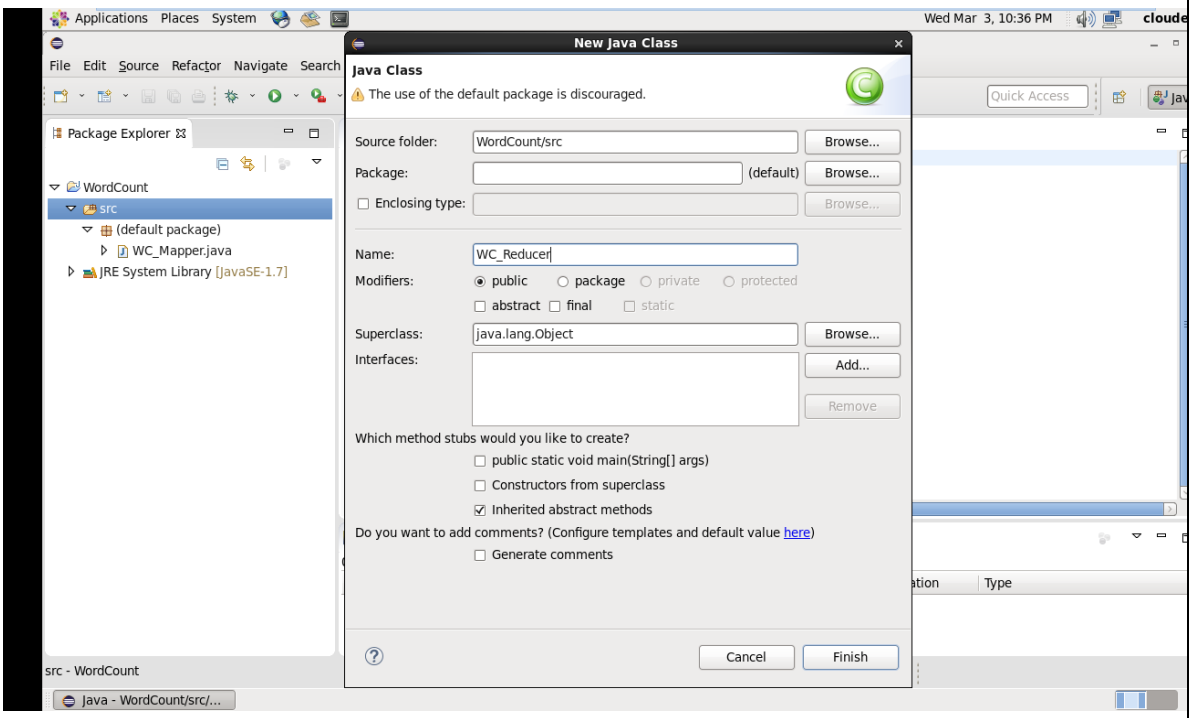
9) Right click on the created project, New- >Class

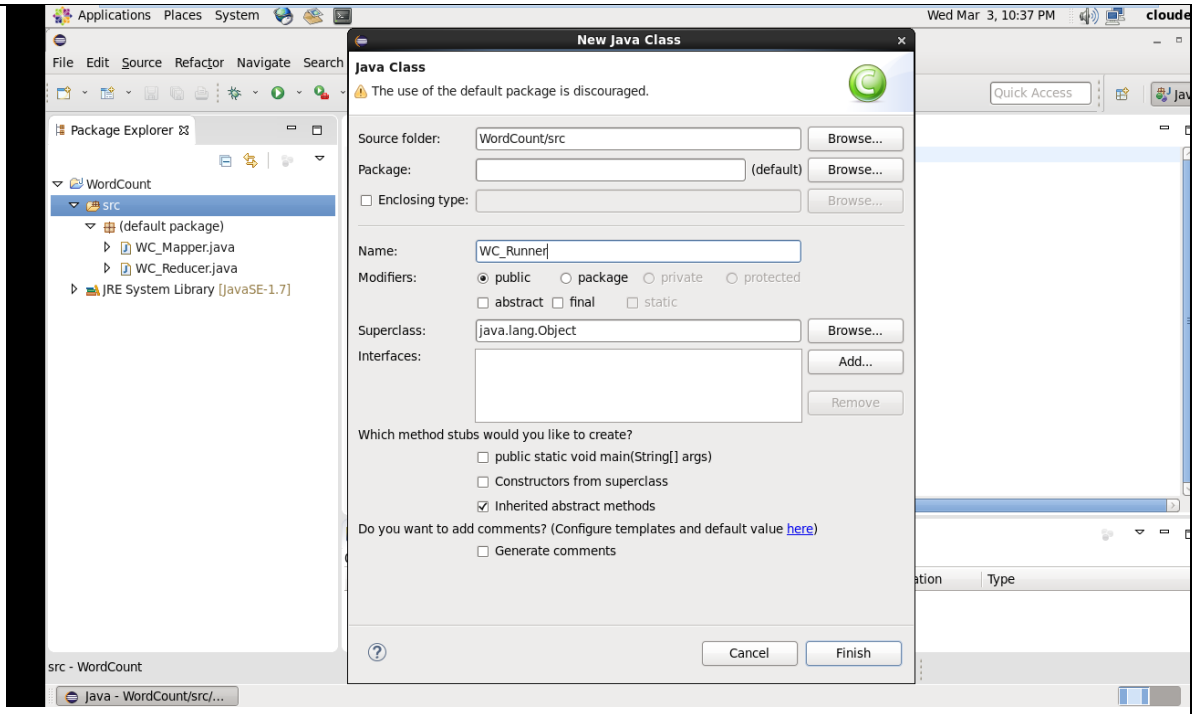


10) Provide the name of class file, preferably as WC_Mapper

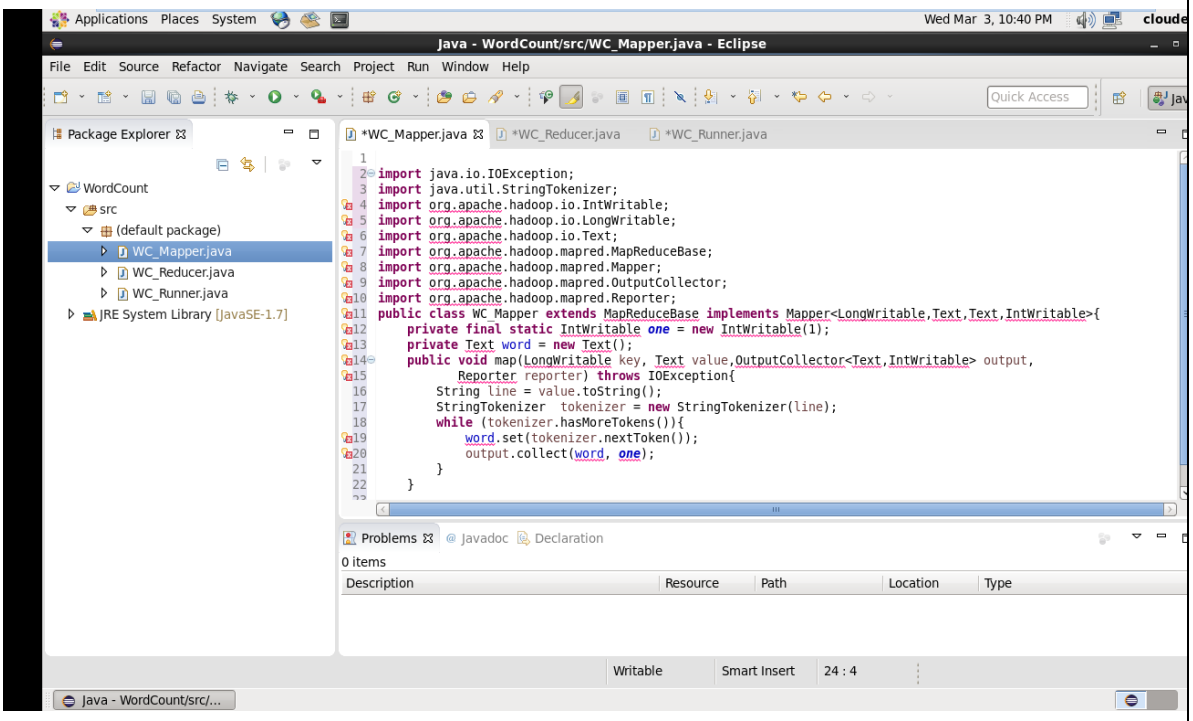


11) Create two more class files and provide the names as reducer and runner.

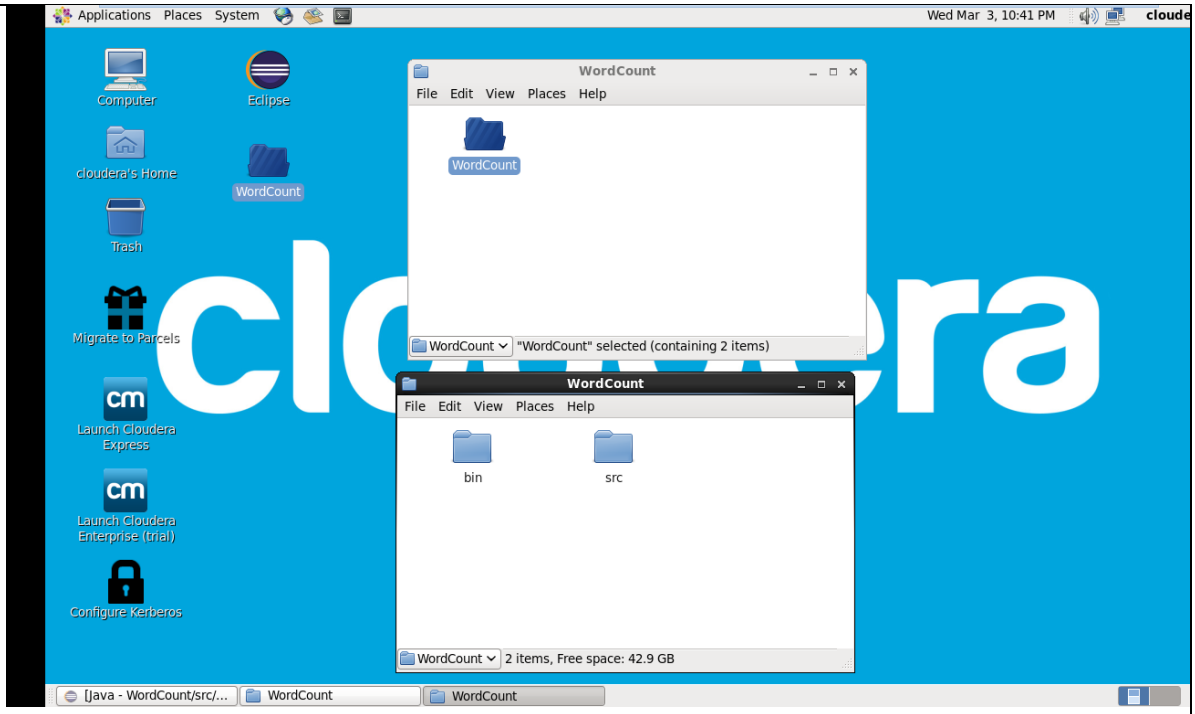




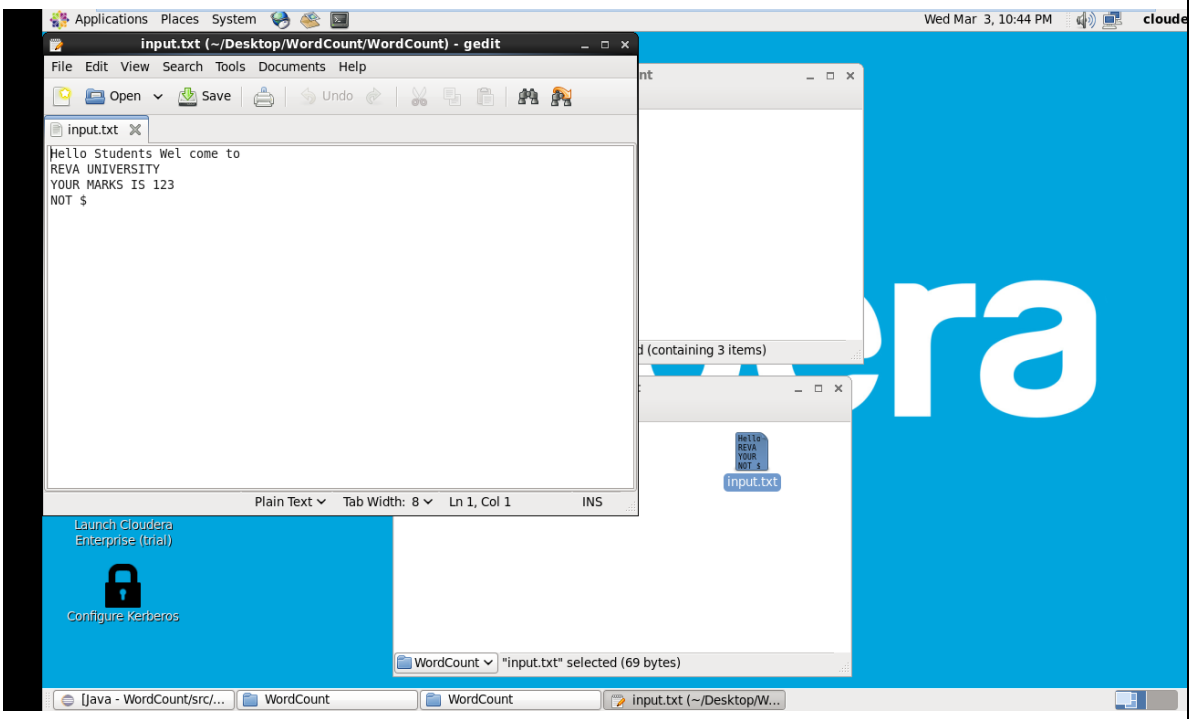
12) Write the program code.



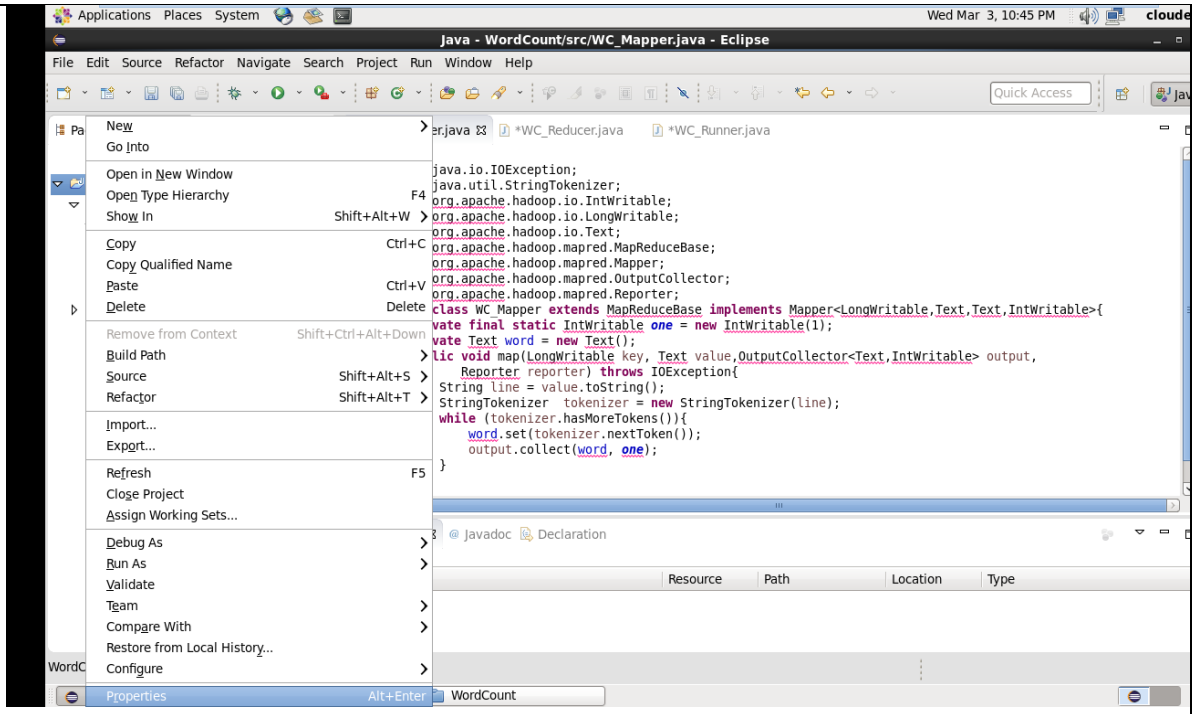
13) Goto workspace created. Goto src folder.



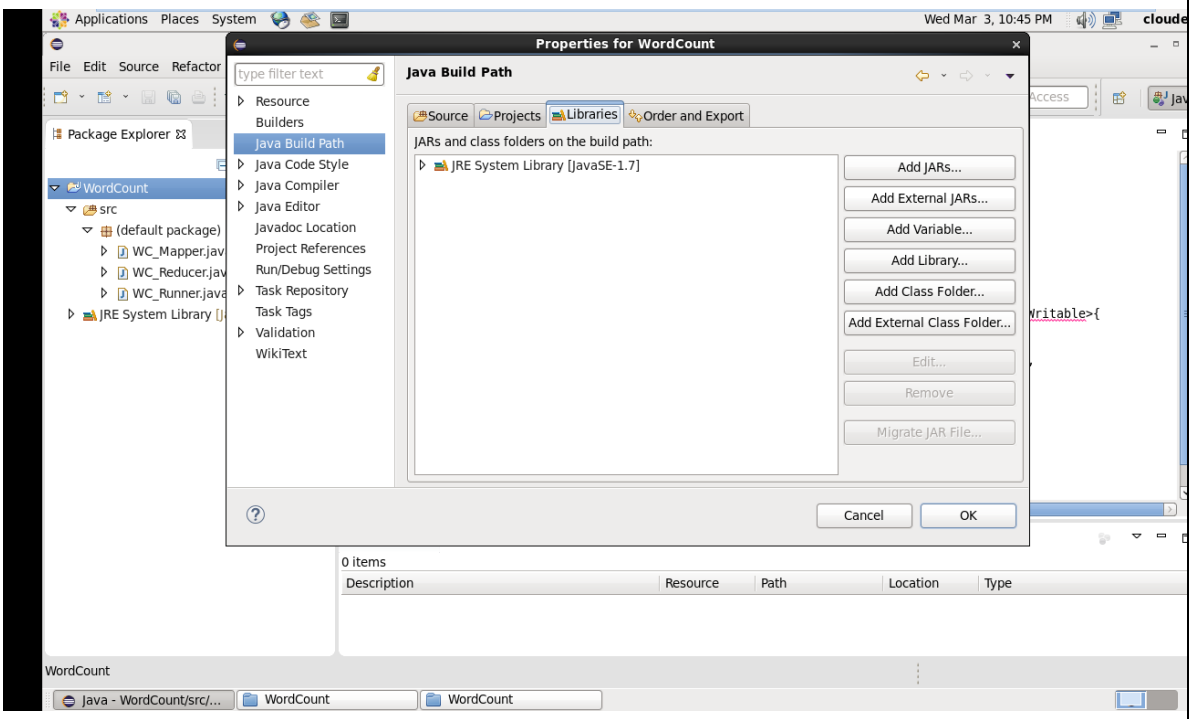
14) Create 'input.txt' file to have the nay text data as an input to the program.



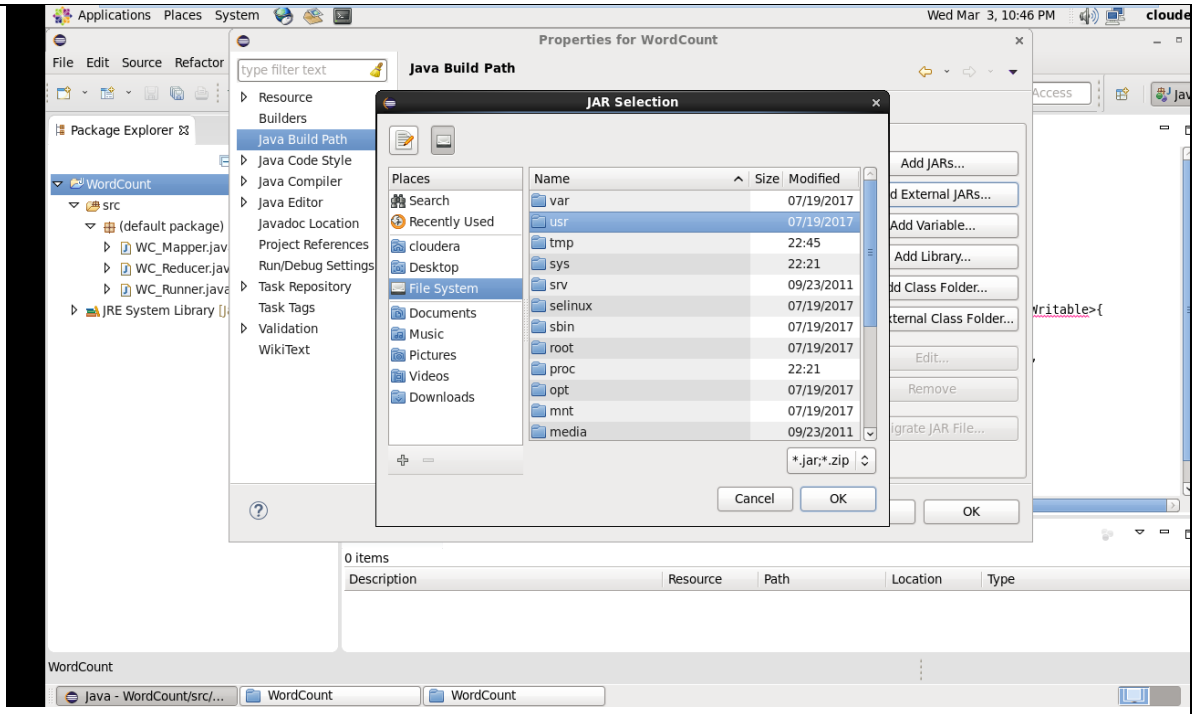
15) Right click on WordCount Project, goto properties.



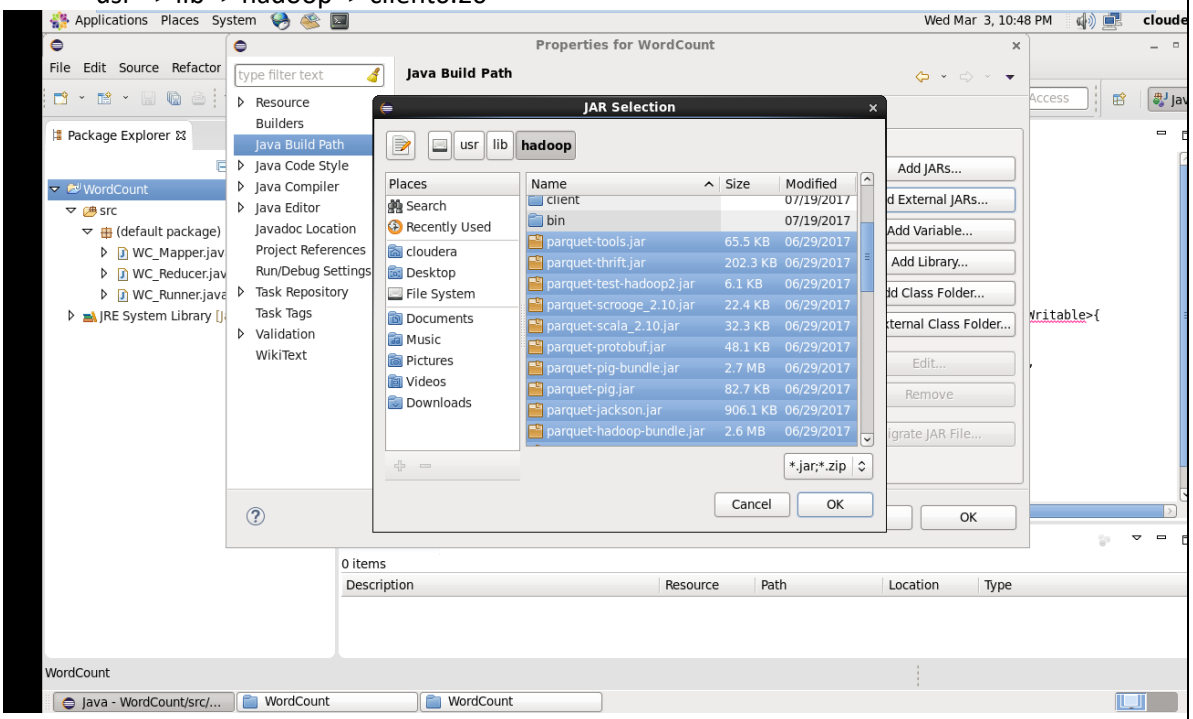
16) Goto Java Build Path -> Libraries-> Add External JARs.

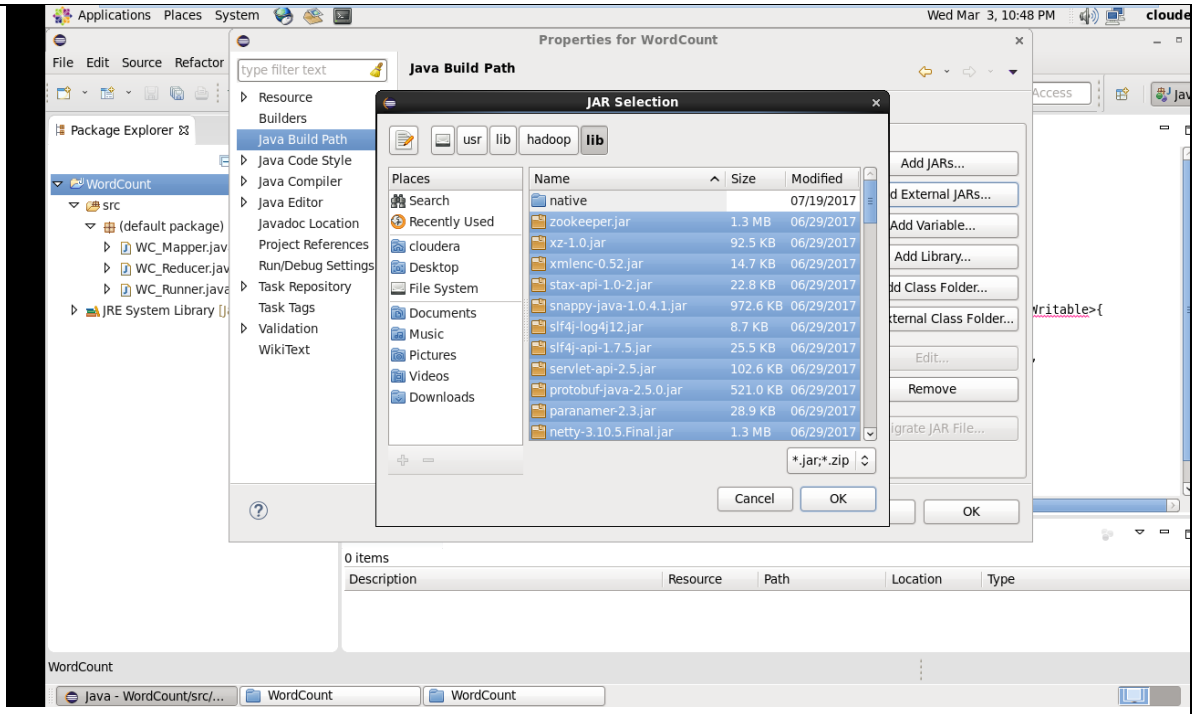


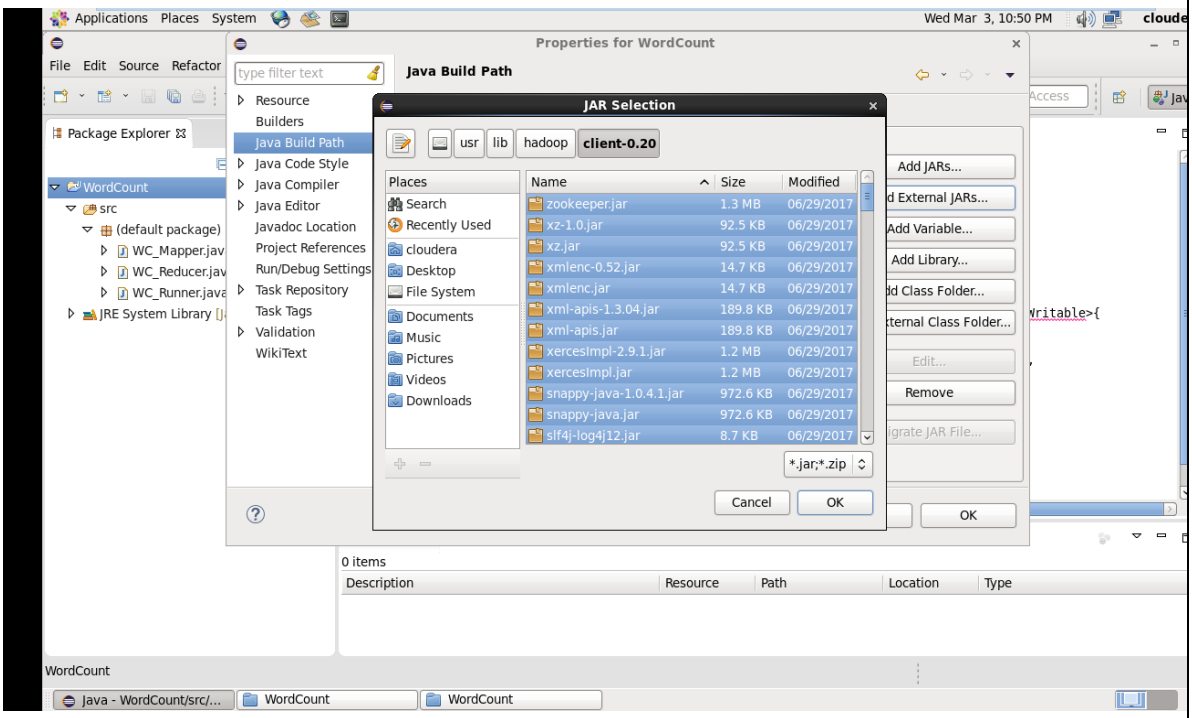
17)File System -> usr



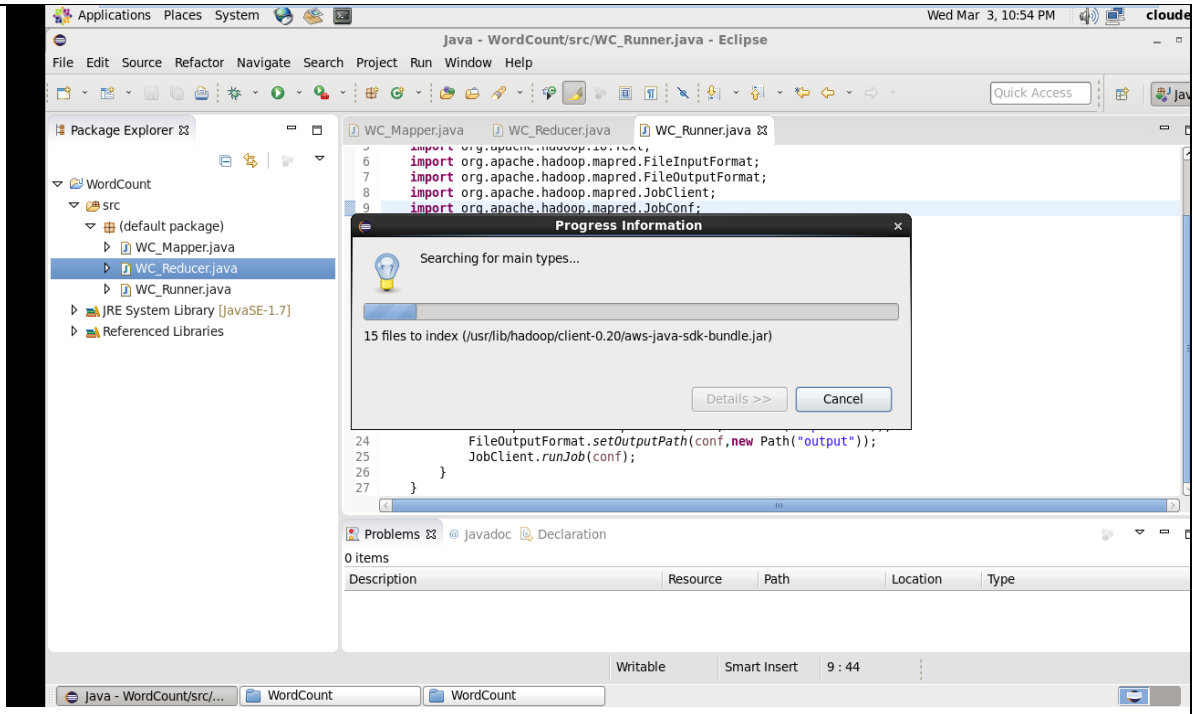
17) Fetch the JAR files from 3 locations as
    usr - > lib -> hadoop
    usr - > lib -> hadoop -> lib
    usr - > lib -> hadoop -> client0.20



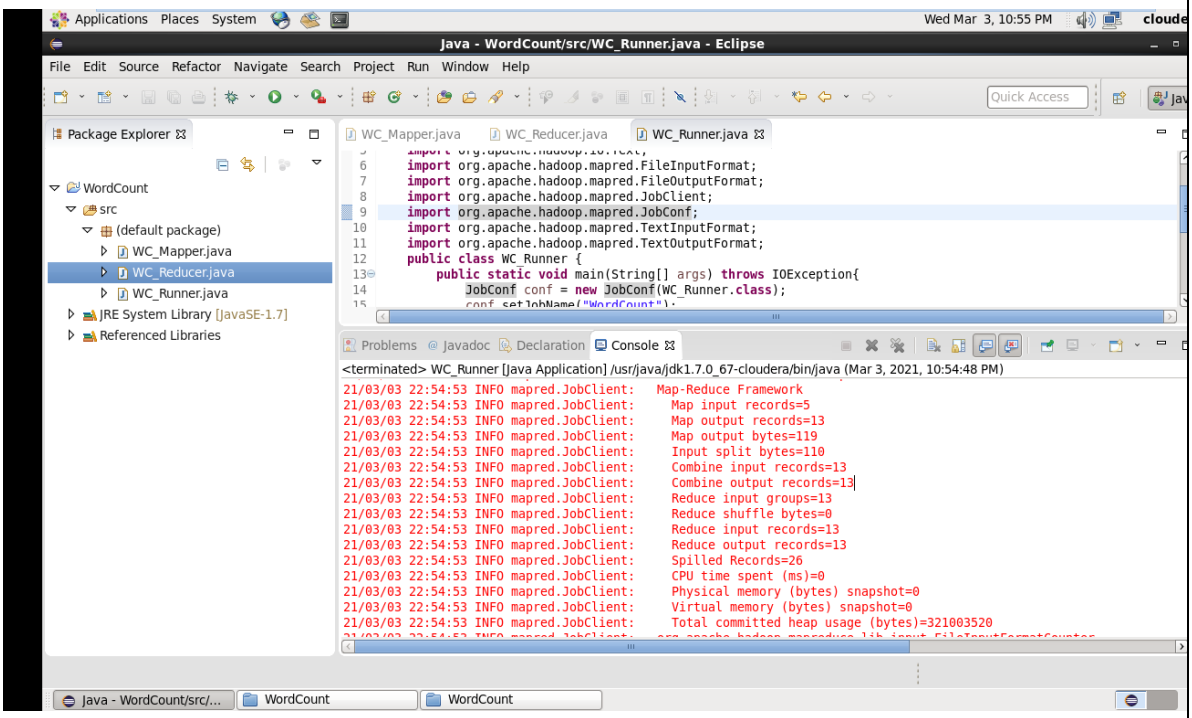




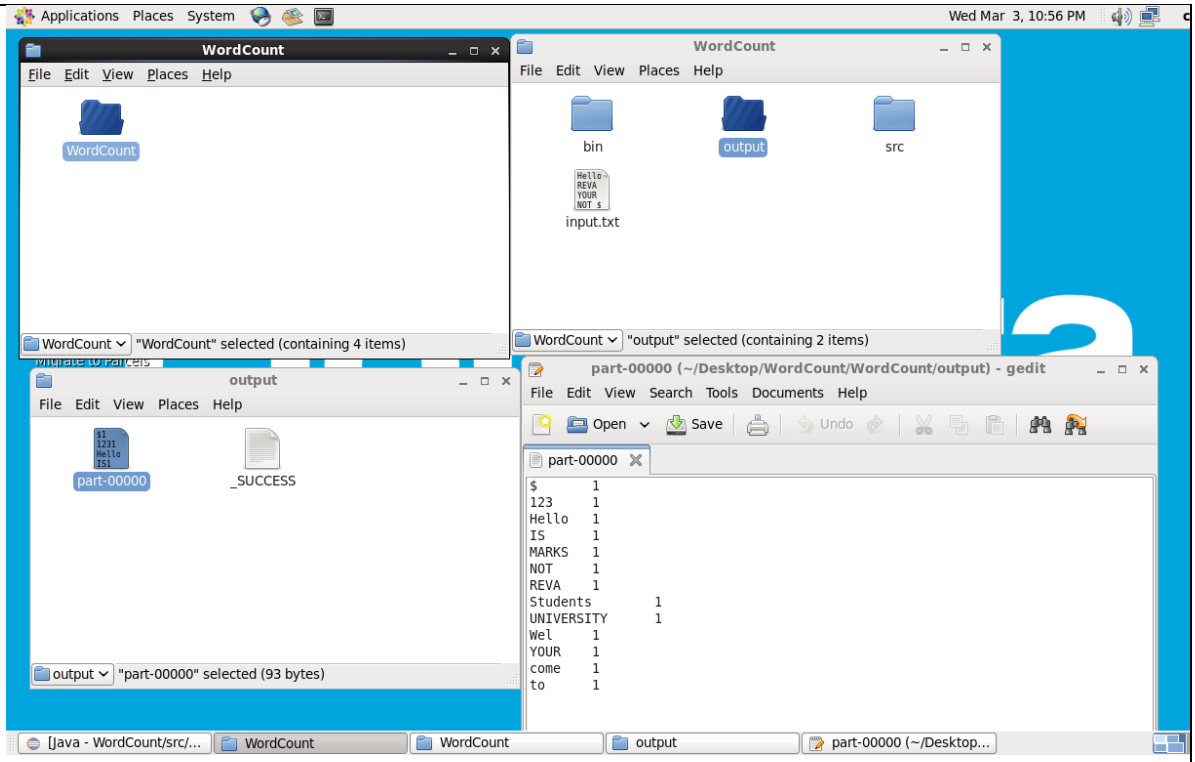
18) Once the JAR files are attached, make sure all errors are rectified, and then RUN.



19) Executed successfully.



20) Goto workspace. Open the output folder. And a part file, you can see the final result.



Program:

WC_Mapper.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
         Reporter reporter) throws IOException{
      String line = value.toString();
      StringTokenizer  tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()){
         word.set(tokenizer.nextToken());
         output.collect(word, one);
      }
   }

}
```

WC_Reducer.java

```
import java.io.IOException;
   import java.util.Iterator;
   import org.apache.hadoop.io.IntWritable;
   import org.apache.hadoop.io.Text;
   import org.apache.hadoop.mapred.MapReduceBase;
```

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

   public class WC_Reducer  extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
   public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable>
output,
    Reporter reporter) throws IOException {
   int sum=0;
   while (values.hasNext()) {
   sum+=values.next().get();
   }
   output.collect(key,new IntWritable(sum));
   }
   }
```

WC_Runner.java

```
 import java.io.IOException;
   import org.apache.hadoop.fs.Path;
   import org.apache.hadoop.io.IntWritable;
   import org.apache.hadoop.io.Text;
   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.mapred.TextInputFormat;
   import org.apache.hadoop.mapred.TextOutputFormat;
   public class WC_Runner {
      public static void main(String[] args) throws IOException{
         JobConf conf = new JobConf(WC_Runner.class);
         conf.setJobName("WordCount");
         conf.setOutputKeyClass(Text.class);
         conf.setOutputValueClass(IntWritable.class);
         conf.setMapperClass(WC_Mapper.class);
         conf.setCombinerClass(WC_Reducer.class);
         conf.setReducerClass(WC_Reducer.class);
         conf.setInputFormat(TextInputFormat.class);
         conf.setOutputFormat(TextOutputFormat.class);
         FileInputFormat.setInputPaths(conf,new Path("input.txt"));
         FileOutputFormat.setOutputPath(conf,new Path("output"));
         JobClient.runJob(conf);
      }
   }
```

Session 8

Problem Statement:

Implement and Evaluate the performance of MapReduce program on character count for different file size.

Student Learning Outcomes:

Execute MapReduce programs on Hadoop and analyze the results.

Theoretical Description:

MapReduce: 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.

Requirements:

- Physical Computer with minimum of 4GB RAM and other specifications.
- Cloudera, Hadoop.
- JDK, Eclipse.

Procedure: Follow same as Experiment 7.

1. Switch on the VMware workstation.
2. Power on the Cloudera.
3. Switch on the Eclipse.
4. Goto File -> Switch workspace - >other.
5. Set the path to store the workspace.
6. Restore the workspace.
7. Goto File-> New-> Java Project.
8. Provide the title of the project, preferably as 'WordCount'
9. Right click on the created project, New- >Class
10. Provide the name of class file, preferably as WC_Mapper
11. Create two more class files and provide the names as reducer and runner.
12. Write the program code.
13. Goto workspace created. Goto src folder.
14. Create 'input.txt' file to have the nay text data as an input to the program.
15. Right click on WordCount Project, goto properties.
16. Goto Java Build Path -> Libraries-> Add External JARs.
17. File System -> usr
18. Fetch the JAR files from 3 locations as
    usr - > lib -> hadoop
    usr - > lib -> hadoop -> lib
    usr - > lib -> hadoop -> client0.20
19. Once the JAR files are attached, make sure all errors are rectified, and then RUN.
20. Executed successfully.
21. Goto workspace. Open the output folder. And a part file, you can see the final result.

Program:

WC_Mapper.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
  public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable> {
    public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
          Reporter reporter) throws IOException{
        String line = value.toString();
        String  tokenizer[] = line.split("");
        for(String SingleChar : tokenizer)
        {
            Text charKey = new Text(SingleChar);
            IntWritable One = new IntWritable(1);
            output.collect(charKey, One);
        }
    }

}
```

WC_Reducer.java

```
import java.io.IOException;
    import java.util.Iterator;
    import org.apache.hadoop.io.IntWritable;
    import org.apache.hadoop.io.Text;
    import org.apache.hadoop.mapred.MapReduceBase;
    import org.apache.hadoop.mapred.OutputCollector;
    import org.apache.hadoop.mapred.Reducer;
    import org.apache.hadoop.mapred.Reporter;

    public class WC_Reducer  extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable>
output,
     Reporter reporter) throws IOException {
    int sum=0;
    while (values.hasNext()) {
    sum+=values.next().get();
    }
    output.collect(key,new IntWritable(sum));
    }
    }
```

WC_Runner.java

```
 import java.io.IOException;
    import org.apache.hadoop.fs.Path;
    import org.apache.hadoop.io.IntWritable;
    import org.apache.hadoop.io.Text;
    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.mapred.TextInputFormat;
    import org.apache.hadoop.mapred.TextOutputFormat;
    public class WC_Runner {
        public static void main(String[] args) throws IOException{
            JobConf conf = new JobConf(WC_Runner.class);
            conf.setJobName("CharCount");
```

```
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf,new Path("input.txt"));
        FileOutputFormat.setOutputPath(conf,new Path("output"));
        JobClient.runJob(conf);
    }
}
```

Appendix A

1	**Problem Statement:**

Install and Configure VMware vSphere ESXi.

2	**Student Learning Outcomes:**

To identify the different ways to install and setup the VMware ESXi, which can act as server for creating multiple VM's.

3	**Theoretical Description:**

vSphere: formerly VMware Infrastructure, is VMware's cloud computing virtualization platform.

ESXi: formerly ESX, is an enterprise-class, type-1 hypervisor developed by VMware for deploying and serving virtual computers. As a type-1 hypervisor, ESXi is not a software application that one installs in an operating system (OS); instead, it includes and integrates vital OS components, such as a kernel.

4	**Requirements**

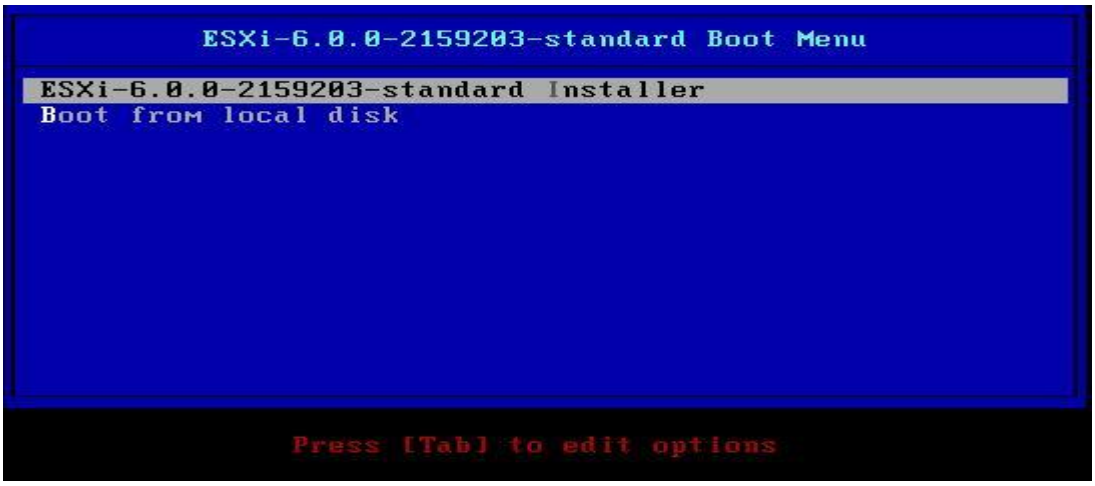
To install or upgrade ESXi 6.0, your hardware and system resources must meet the following requirements:

- Supported server platform. For a list of supported platforms, see the *VMware Compatibility Guide* at http://www.vmware.com/resources/compatibility.
- ESXi 6.0 requires a host machine with at least two CPU cores.
- ESXi 6.0 supports 64-bit x86 processors released after September 2006. This includes a broad range of multi-core processors.
- ESXi 6.0 requires the NX/XD bit to be enabled for the CPU in the BIOS.
- ESXi requires a minimum of 4GB of physical RAM. It is recommended to provide at least 8 GB of RAM to run virtual machines in typical production environments.
- To support 64-bit virtual machines, support for hardware virtualization (Intel VT-x or AMD RVI) must be enabled on x64 CPUs.
- One or more Gigabit or faster Ethernet controllers. For a list of supported network adapter models.
- SCSI disk or a local, non-network, RAID LUN with unpartitioned space for the virtual machines.
- For Serial ATA (SATA), a disk connected through supported SAS controllers or supported on-board SATA controllers. SATA disks will be considered remote, not local. These disks will not be used as a scratch partition by default because they are seen as remote.

5	**Procedure**

I. Installation of ESXi 6.0.0 on Bare metal

Insert the VMWARE ESXi6.0.0DVD and Power on the Computer, the computer will BOOT from the CDROM, and the following BOOT screen will be displayed

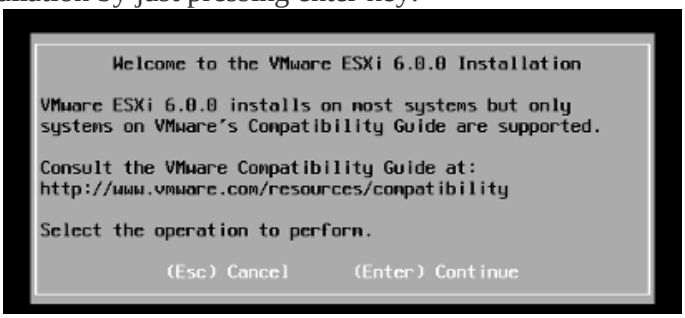


The default option is to BOOT ESXi-6.0.0-2159203-standard Installer after 10 seconds, hit any key will pause the BOOT process.

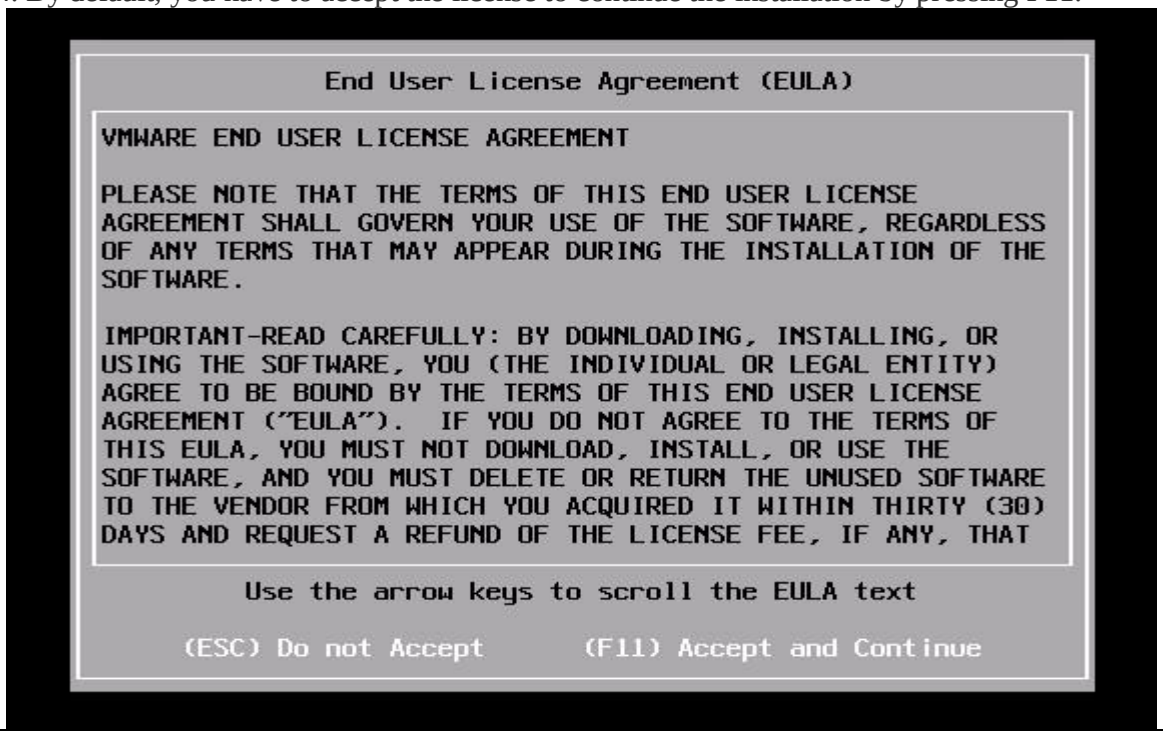
1. ESXi installer will load the necessary files for installation.

2. Once all the modules are loaded successfully, you will get the below screen with server configuration.

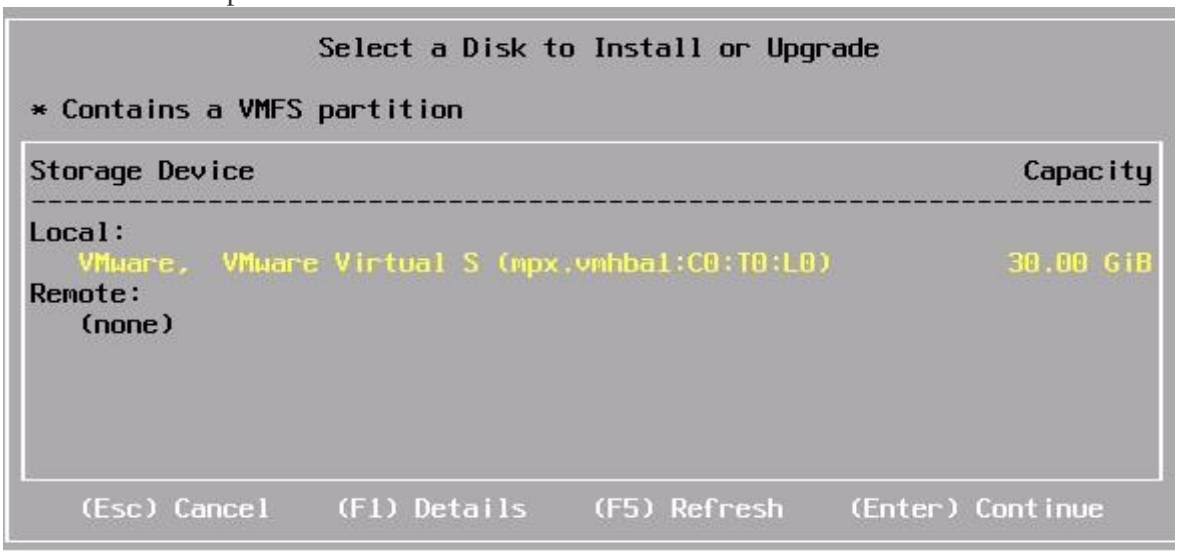
3. Continue the installation by just pressing enter key.



4. By default, you have to accept the license to continue the installation by pressing F11.

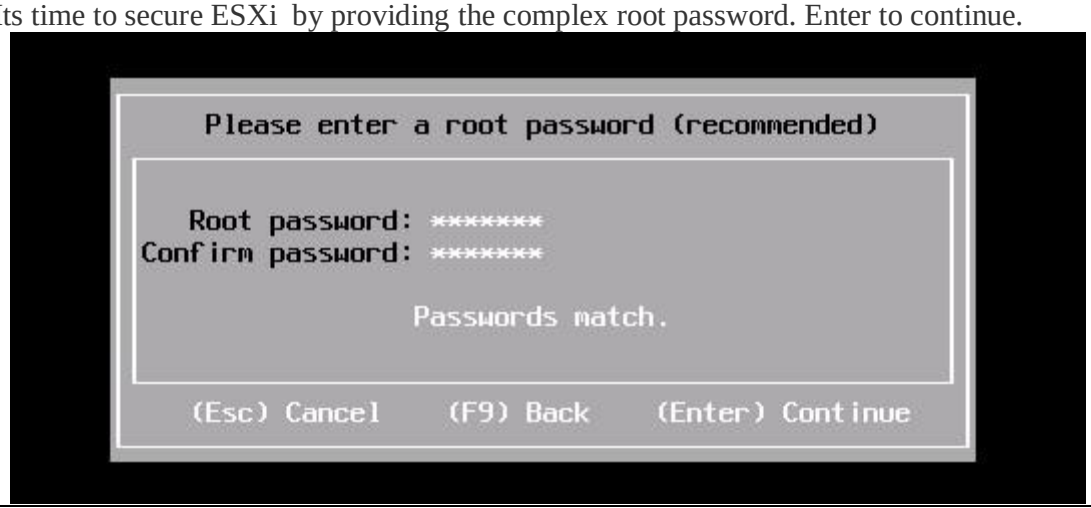


5. Select the disk in which you would like to install VMWARE ESXi 5.1.It requires minimum 4~ to 5~GB disk space. Enter to continue.

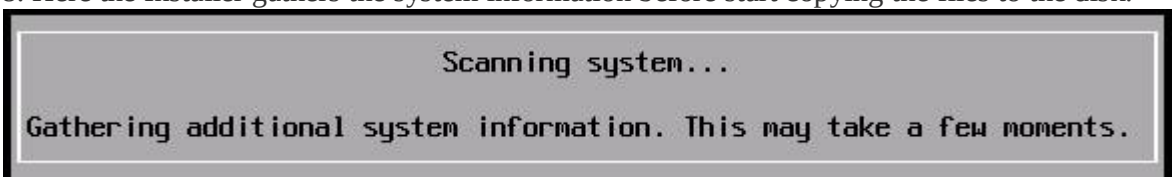


6. Select the keyboard layout. Here i am selecting “US Default”.

7. Its time to secure ESXi by providing the complex root password. Enter to continue.

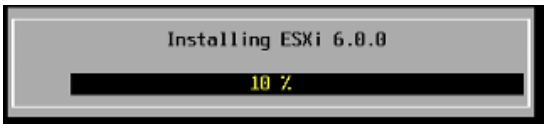


8. Here the Installer gathers the system information before start copying the files to the disk.

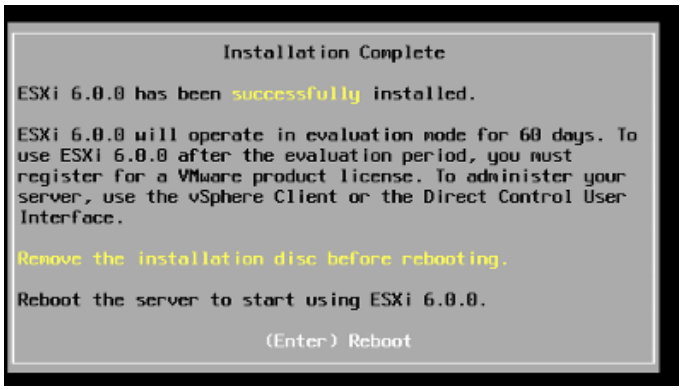


9. Here will be the final confirmation you need to give before destroying any data on that disk. Press F11 to continue the installation.

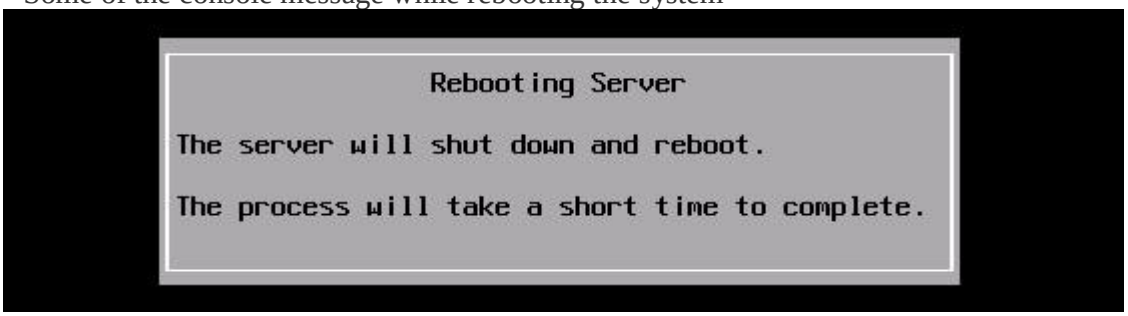
10. Installation begins.



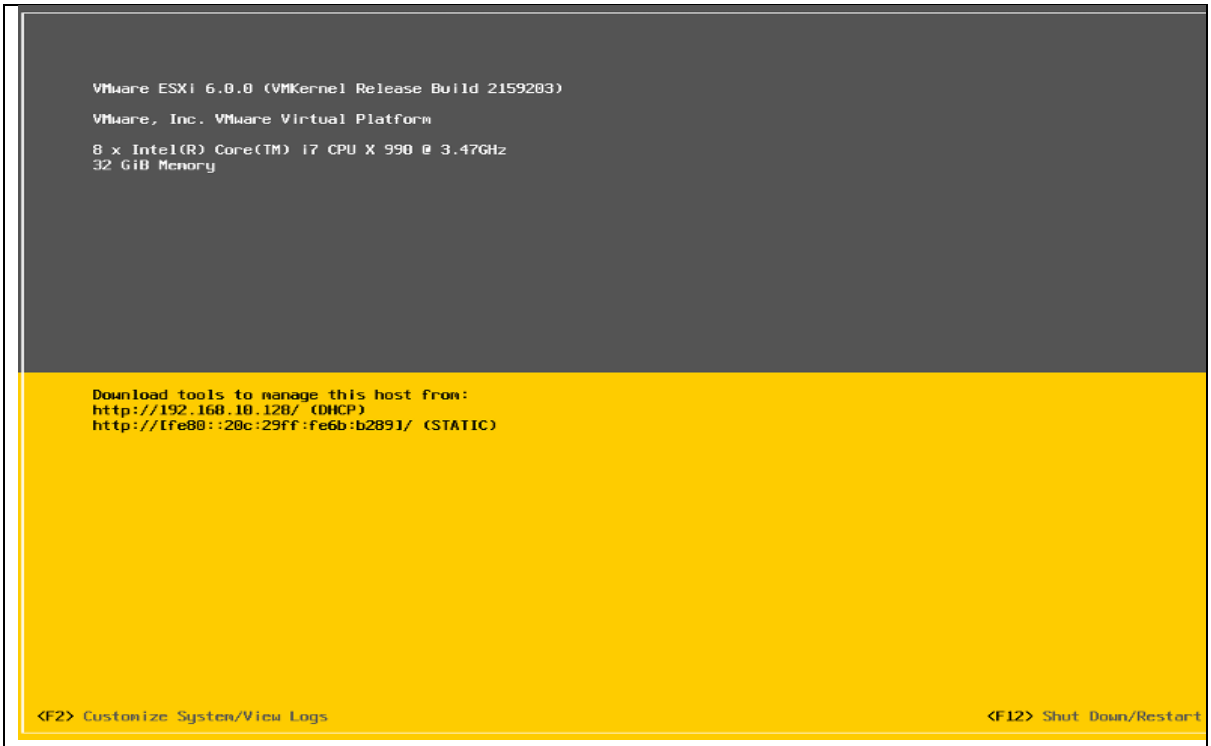
11. Reboot the system to complete the installation. Make sure that you have removed the installation media from the server.



Some of the console message while rebooting the system

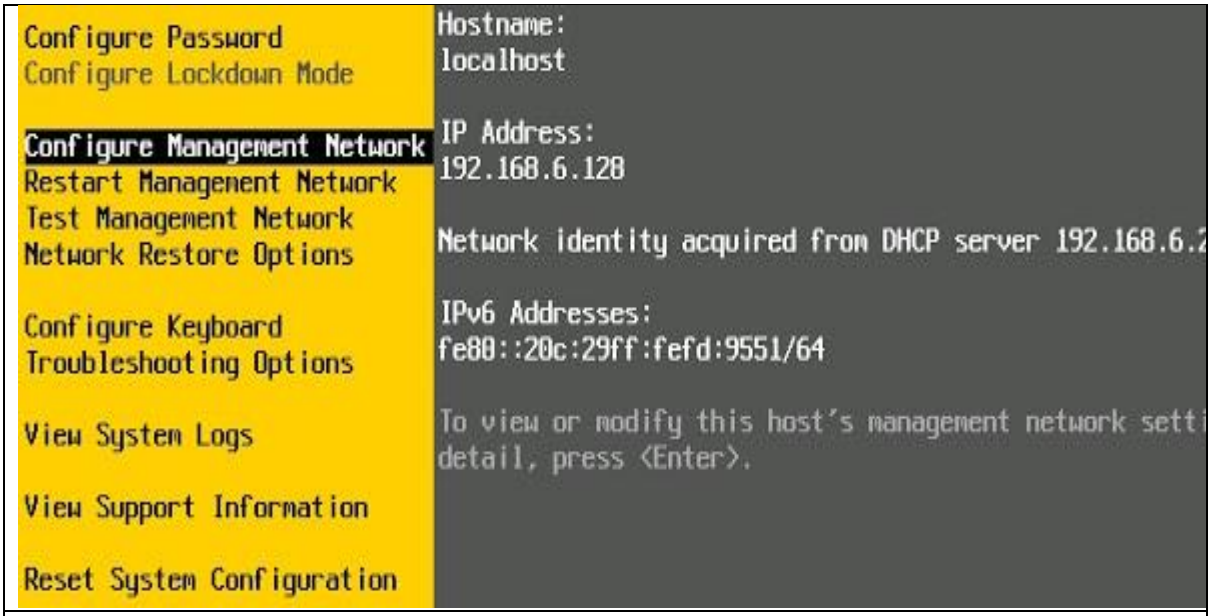


12. Now vmware EXSi 5.1 will boot from hard disk and you will get the below screen, once it's completely up.



13. Many of them wonder that, how to login to the system. You can get login screen by just pressing F2.

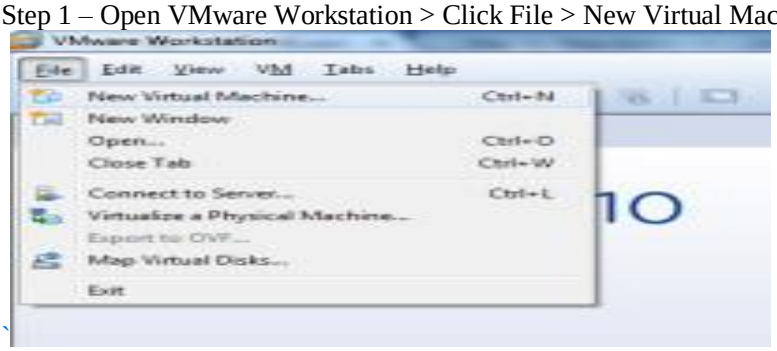
14. After logging to the system, you can see there are many options provided for configuring the ESXi and viewing the system logs. It also provided troubleshooting options as well.



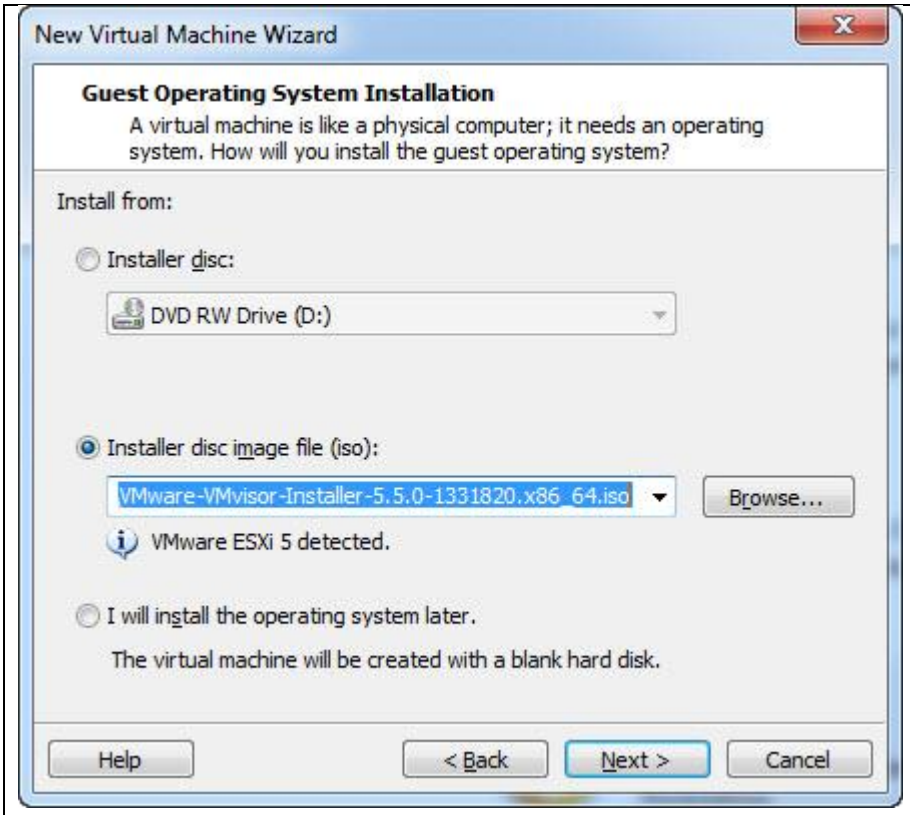
II. Set-Up Virtual Machine in VMware Workstation

The ESXi can also be installed and configured on VMware Workstation, not directly on hardware of the physical machine.

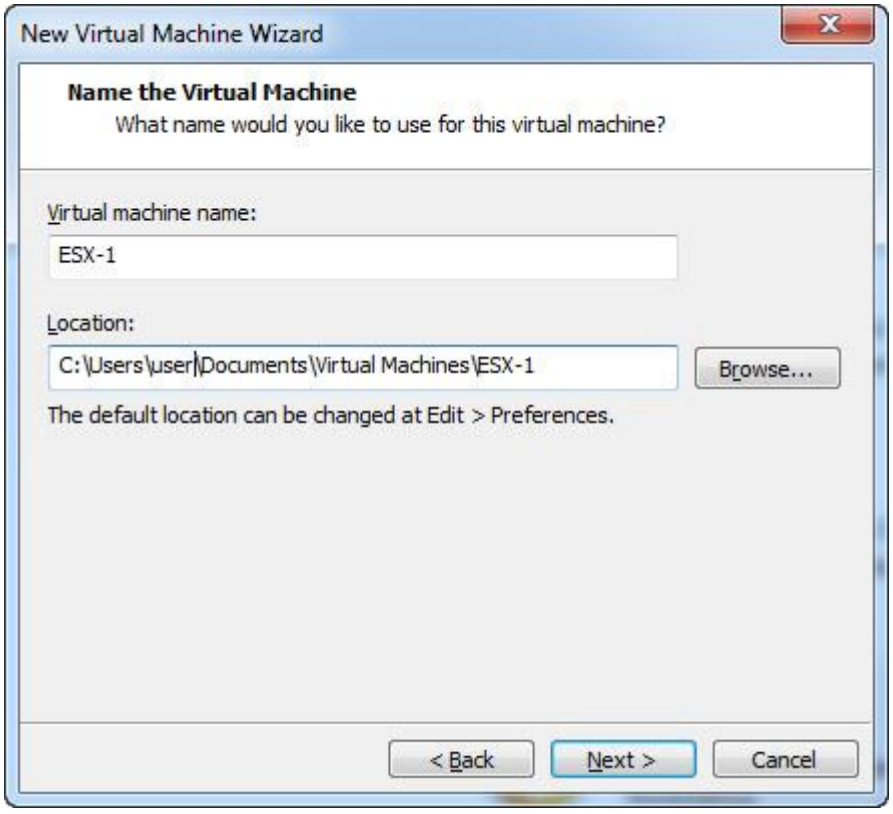
Step 1 – Open VMware Workstation > Click File > New Virtual Machine (Wizard)



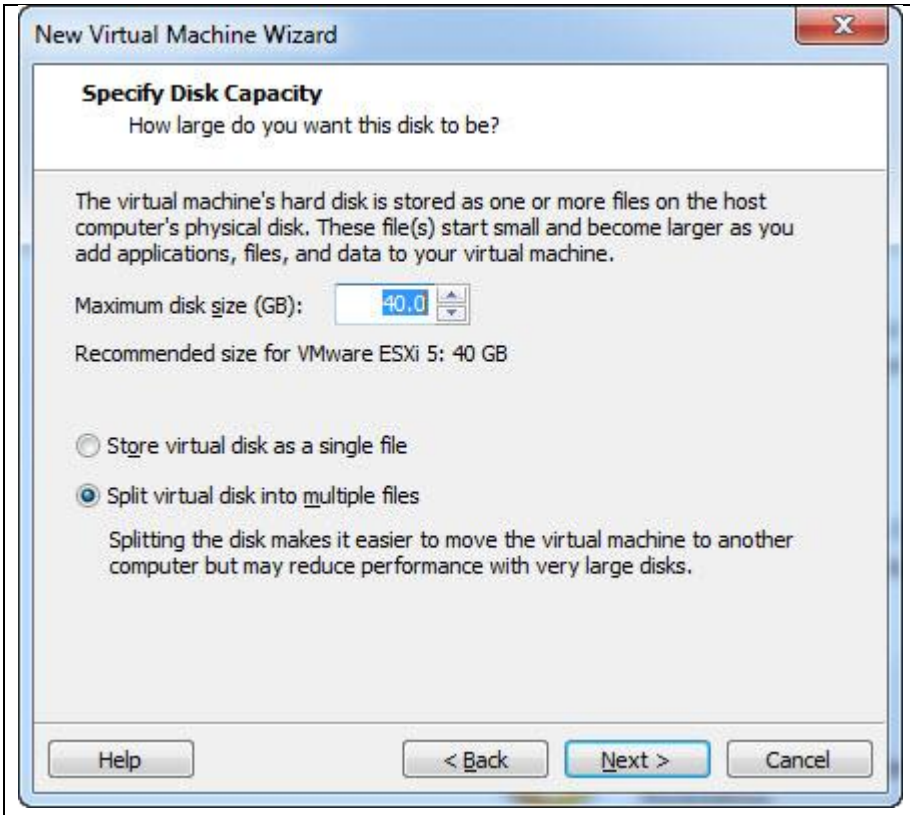
Step 2 – The New Virtual Machine will appear > Select Installer Disk Image File (ISO) > Browse & Select the Downloaded ESXi 5.5 ISO Image from above.



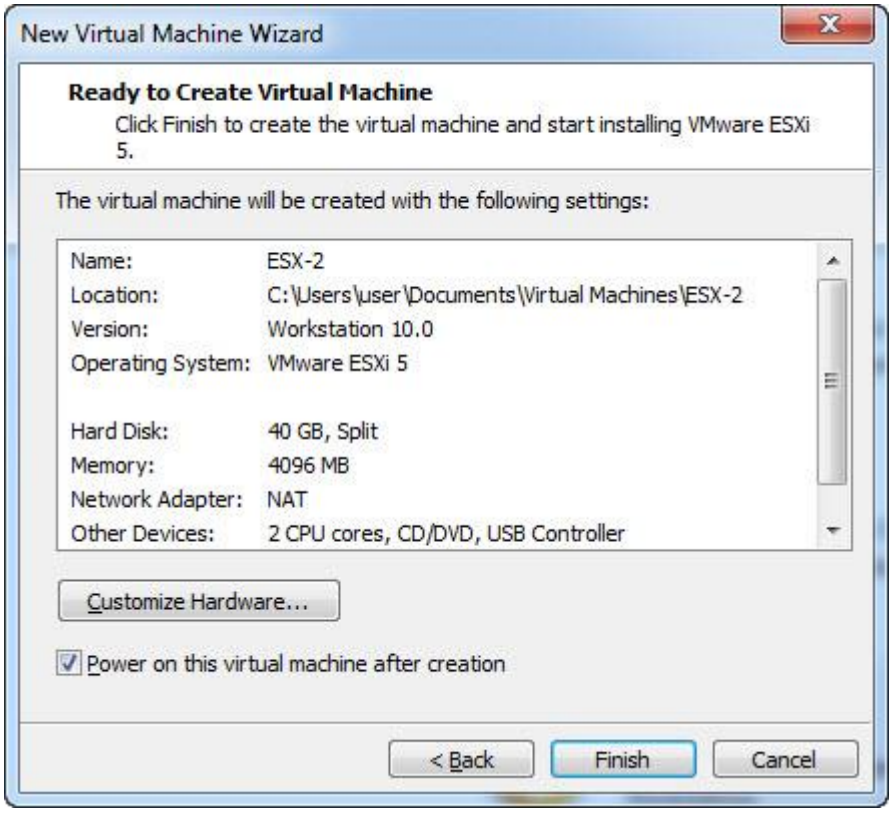
Step 3 – Give your virtual machine a name. For the purposes of this LAB it will be ESX-1.



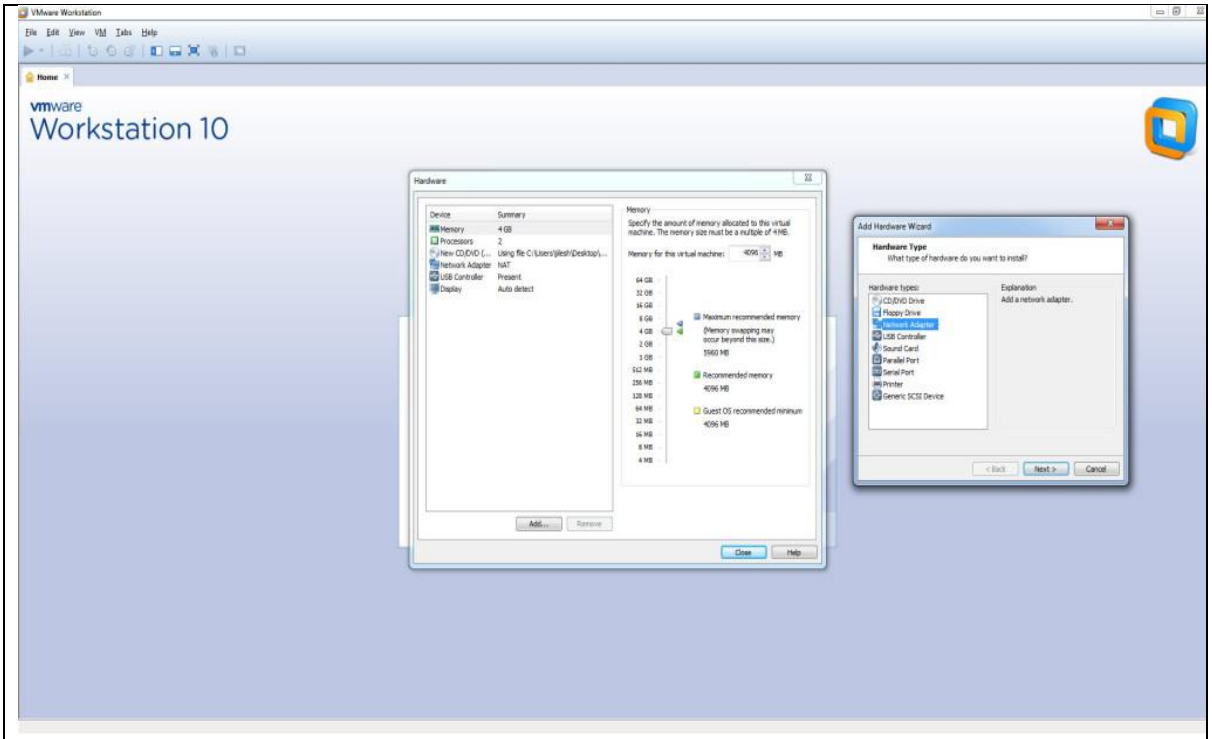
Step 4 – Specify the disk size. 40GB is the default and is more than enough.



Step 5 – Before you click Finish, select "Customise Hardware".



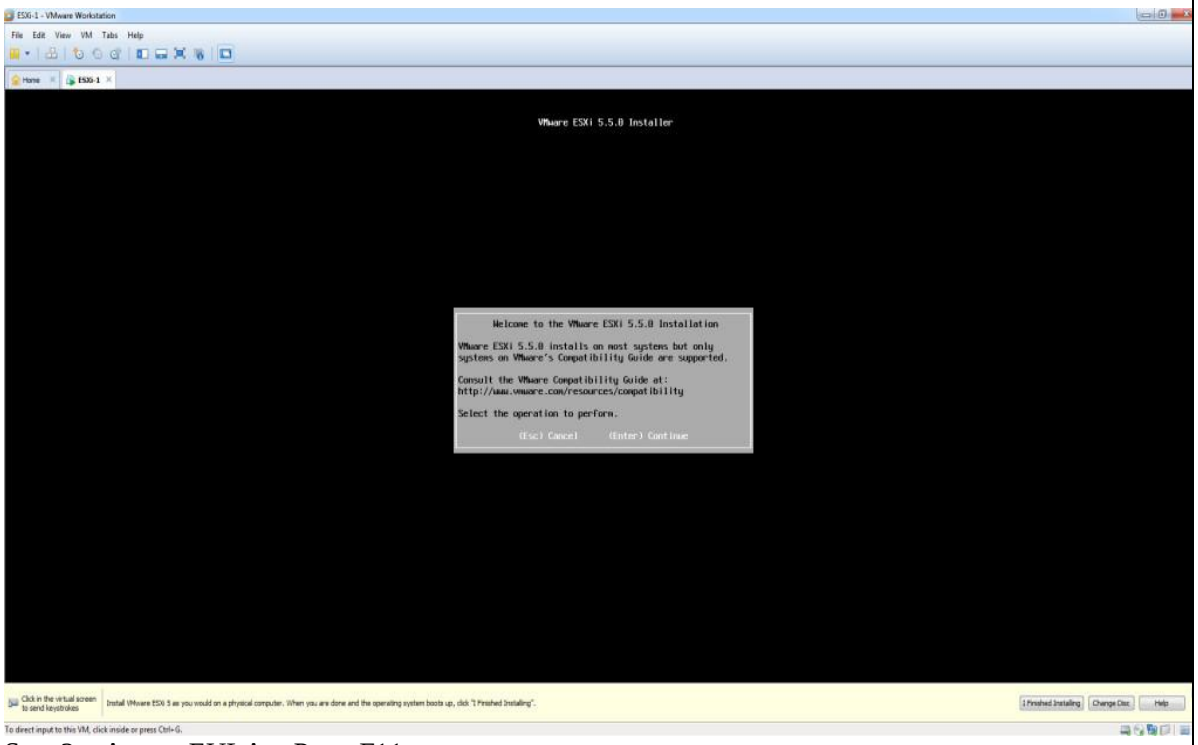
Step 6 – 3 Virtual Network Cards are required for your lab per ESXi Host. Select "Add" to open the Add Hardware Wizard and follow the steps to add 2 more network cards to your virtual machine.



Step 7 – Click on Finish, and wait for your virtual machine to Power on Successfully.

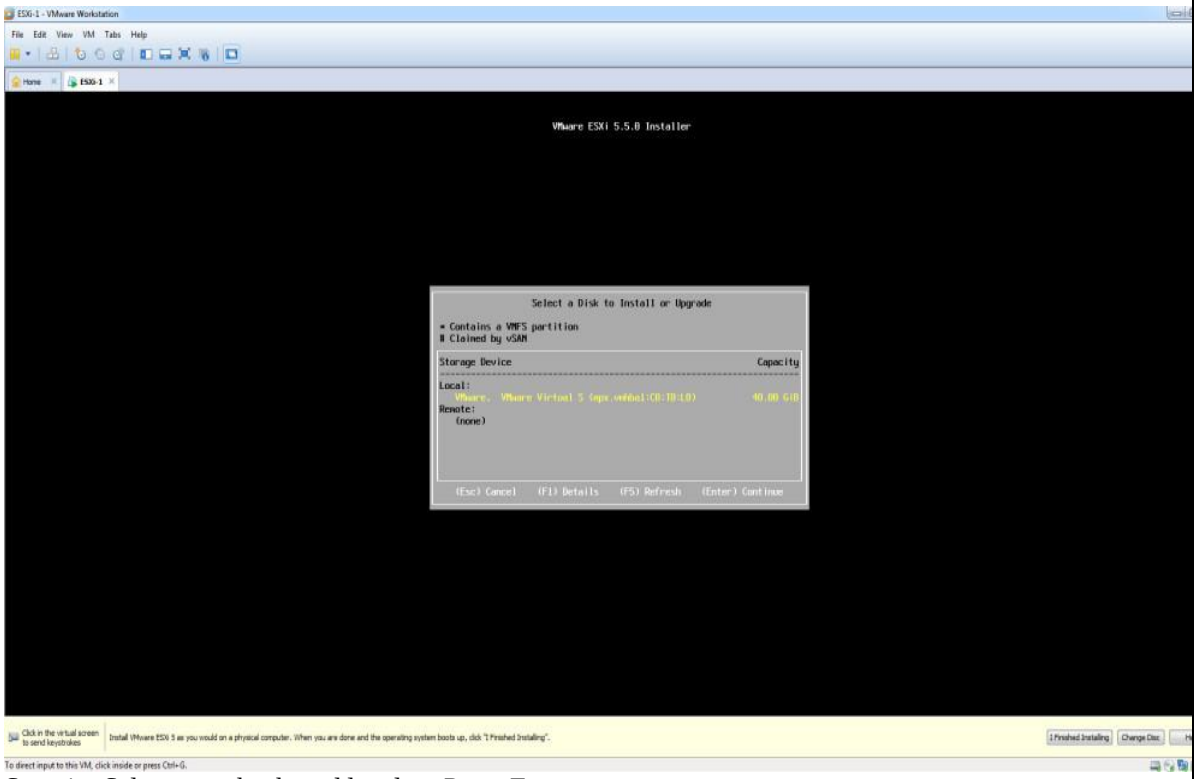
INSTALL ESXI 5.5

Step 1 – Power on your Virtual / Physical Machine, and boot from the ESXi 5.5 ISO Image > Press Enter to Start the installation wizard.

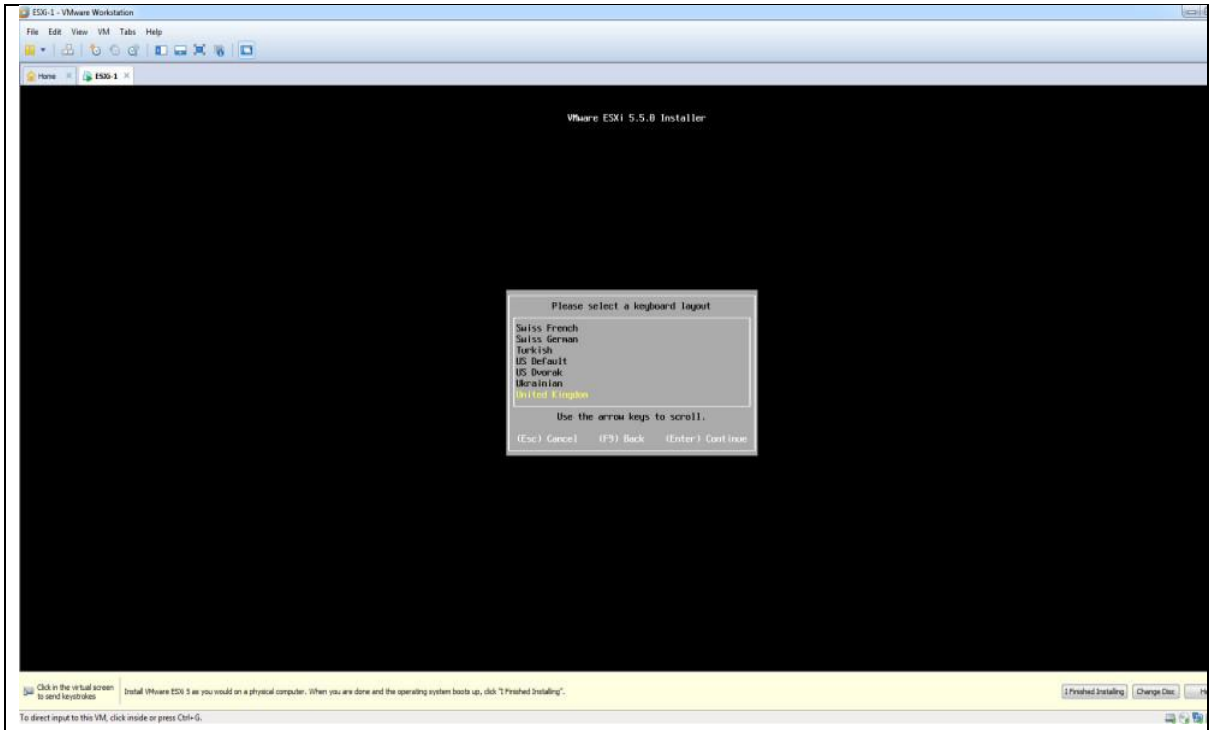


Step 2 – Accept EULA > Press F11

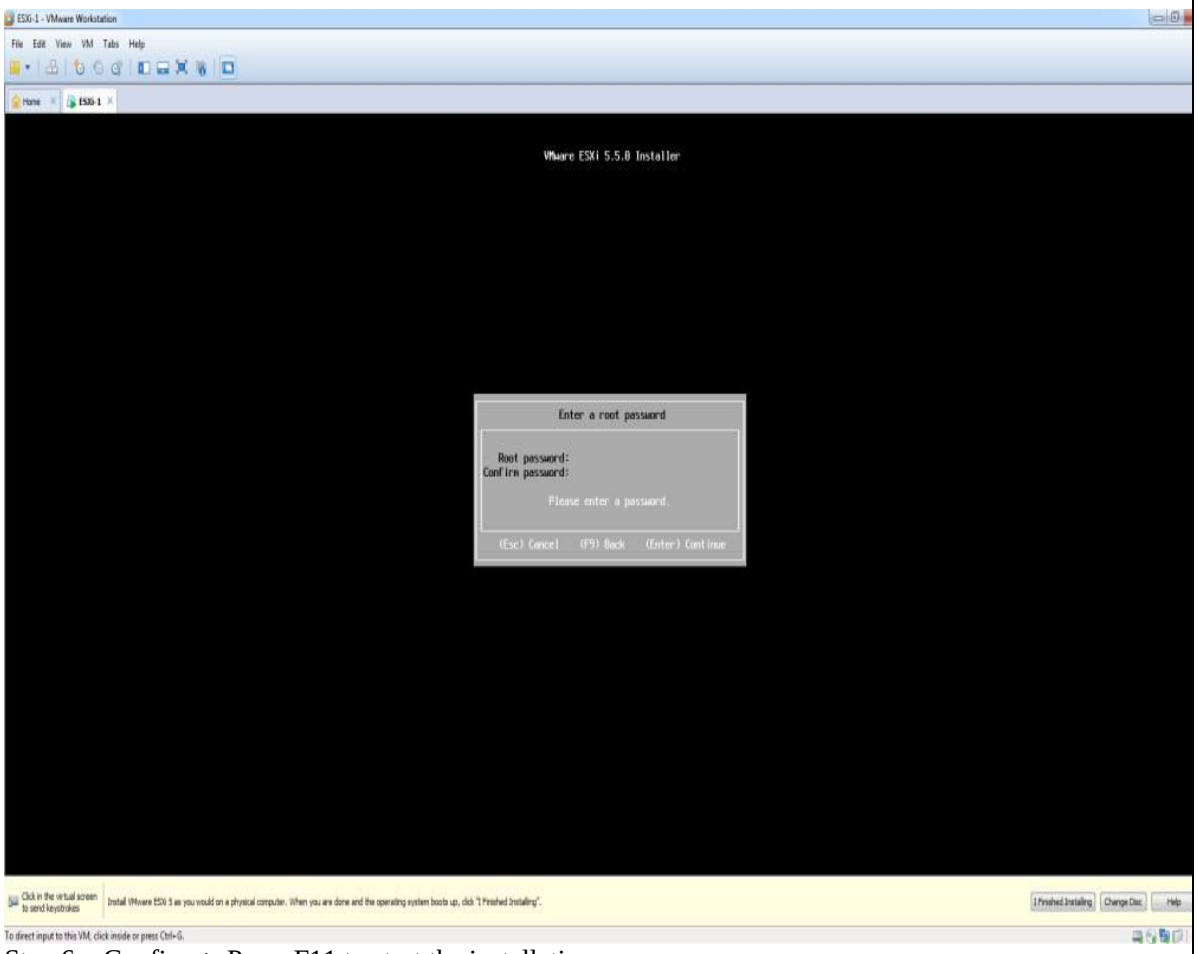
Step 3 – Select the Local Disk for the Installation > Press Enter



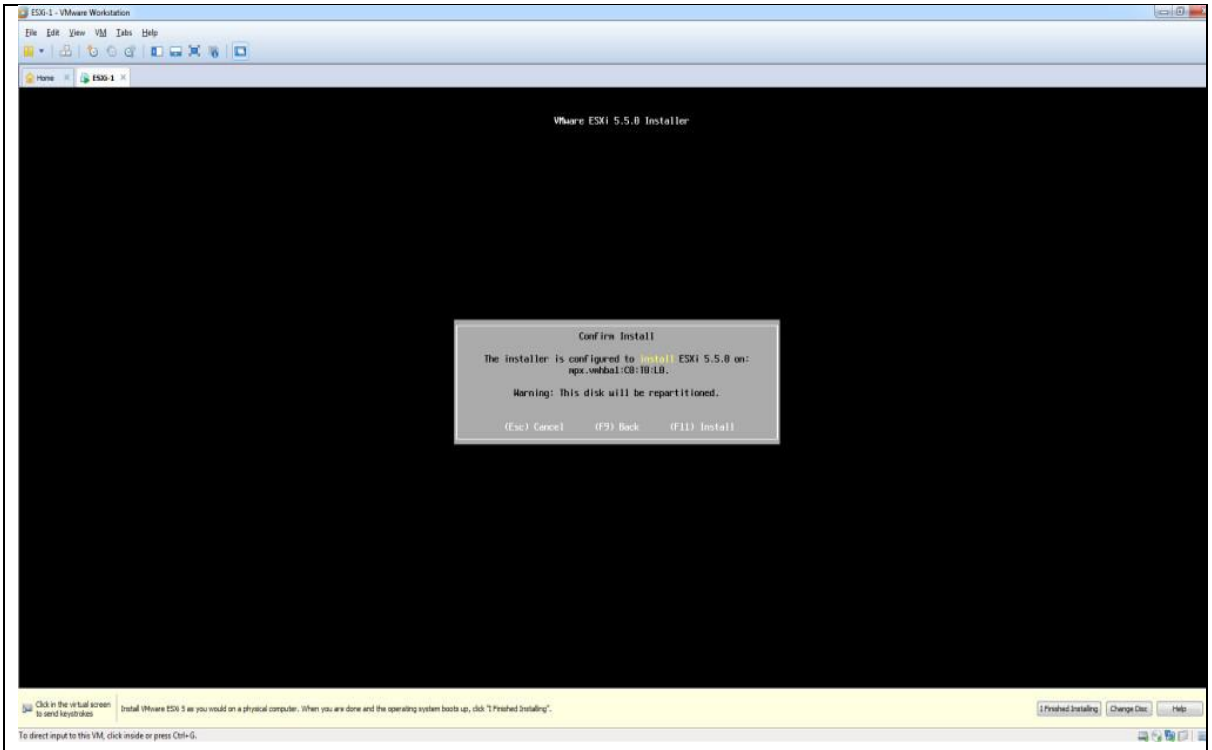
Step 4 – Select your keyboard locale > Press Enter



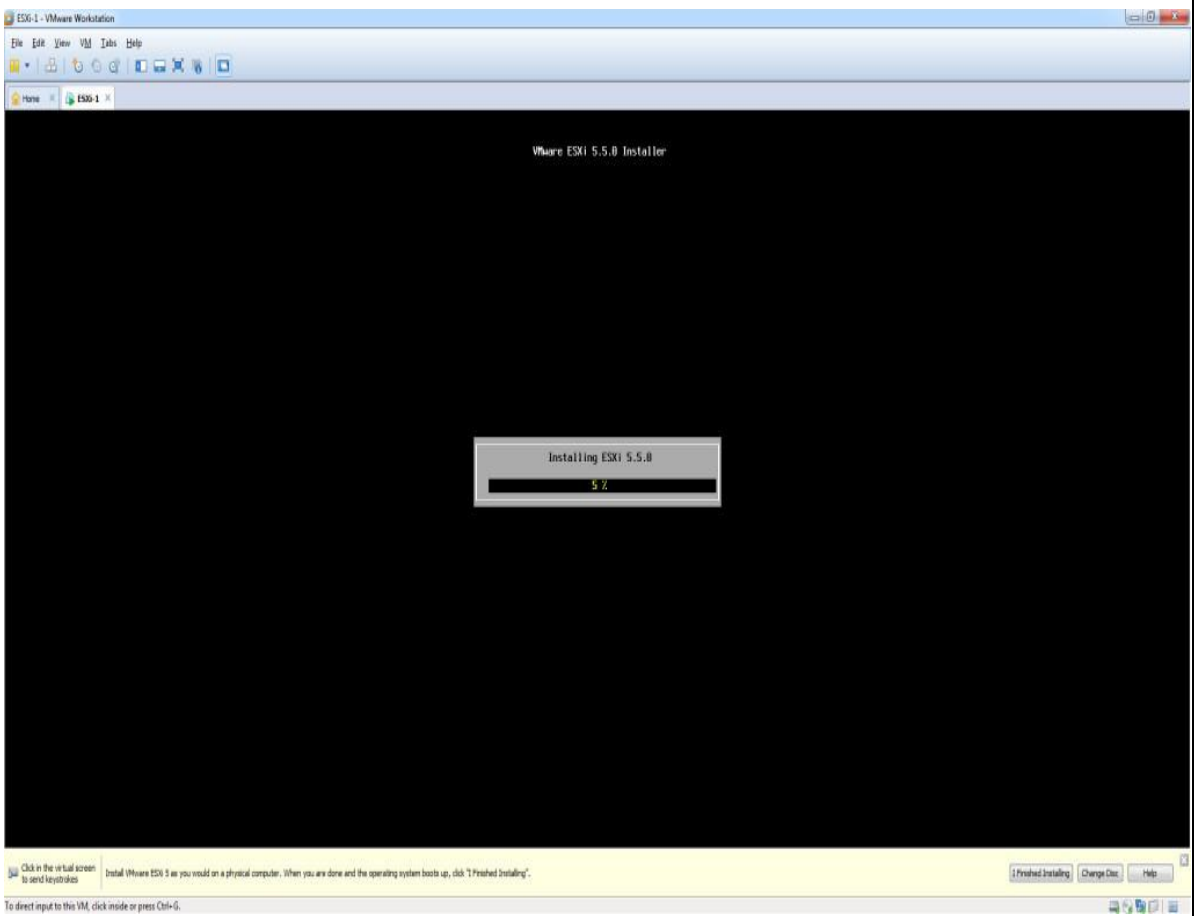
Step 5 – Type a complex password, with a minimum of 7 characters > Press Enter



Step 6 – Confirm > Press F11 to start the installation

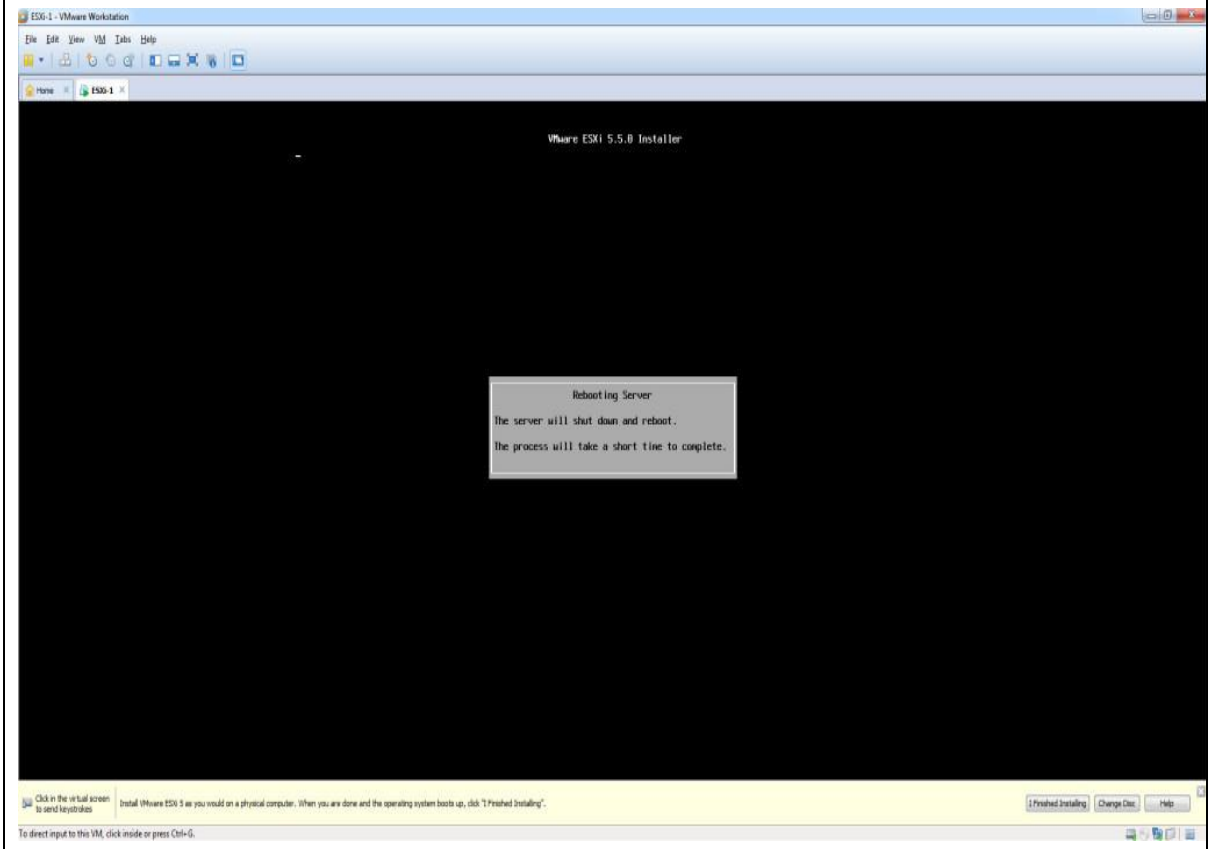


Step 7 – Installation will take about 2-5 minutes to complete.



Step 8 – Press Enter to Reboot once the installation has completed successfully.

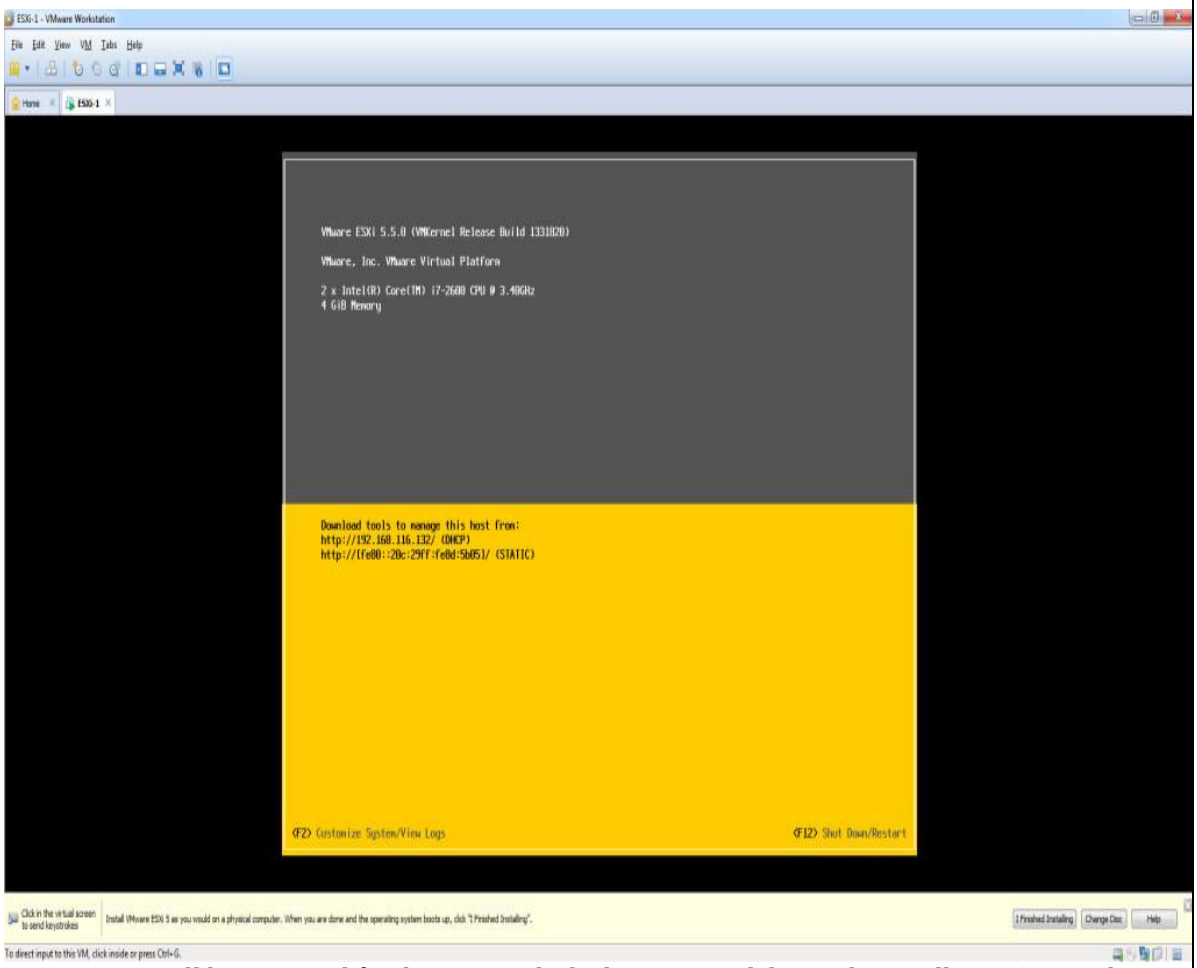
Step 9 – Wait for reboot to complete



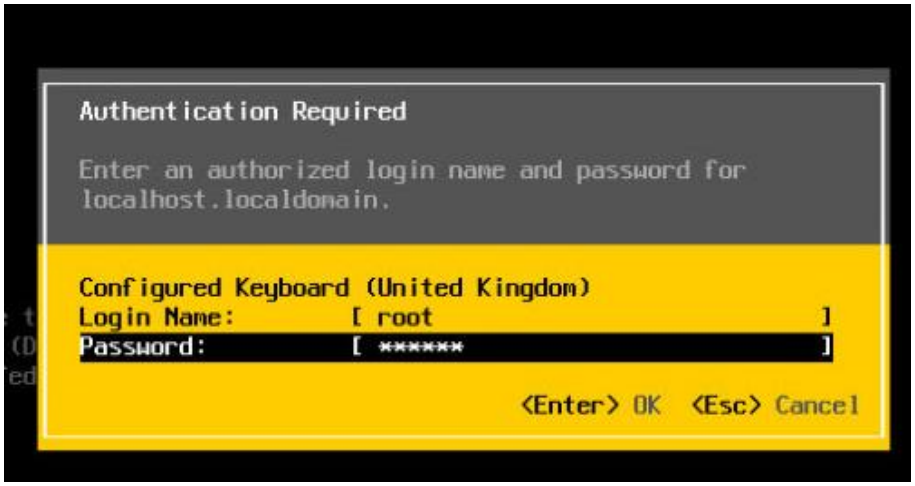
III. CONFIGURE – HOSTNAME, IP ADDRESS, GATEWAY, DNS, PASSWORD ON ESXI 5.5

Once reboot has completed, your installation is now complete. The next step is to configure your ESXi 5.5 Host.

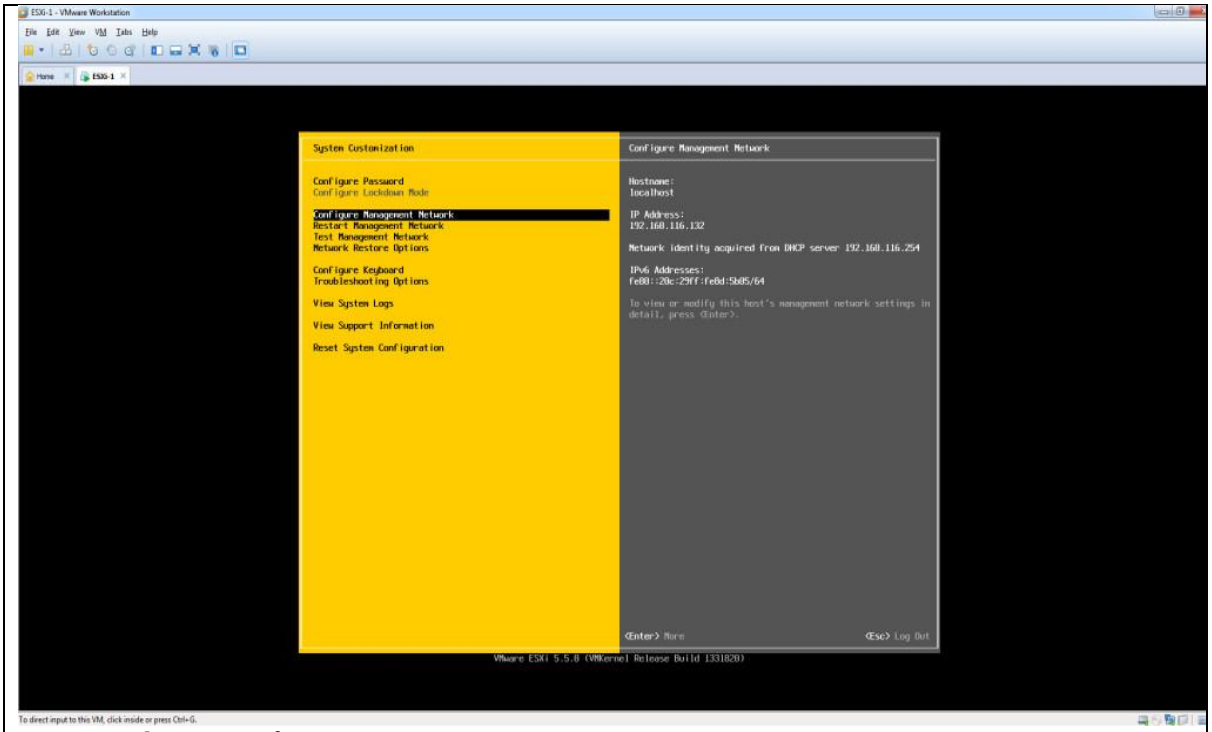
Step 1 – Press F2 to enter the configuration mode



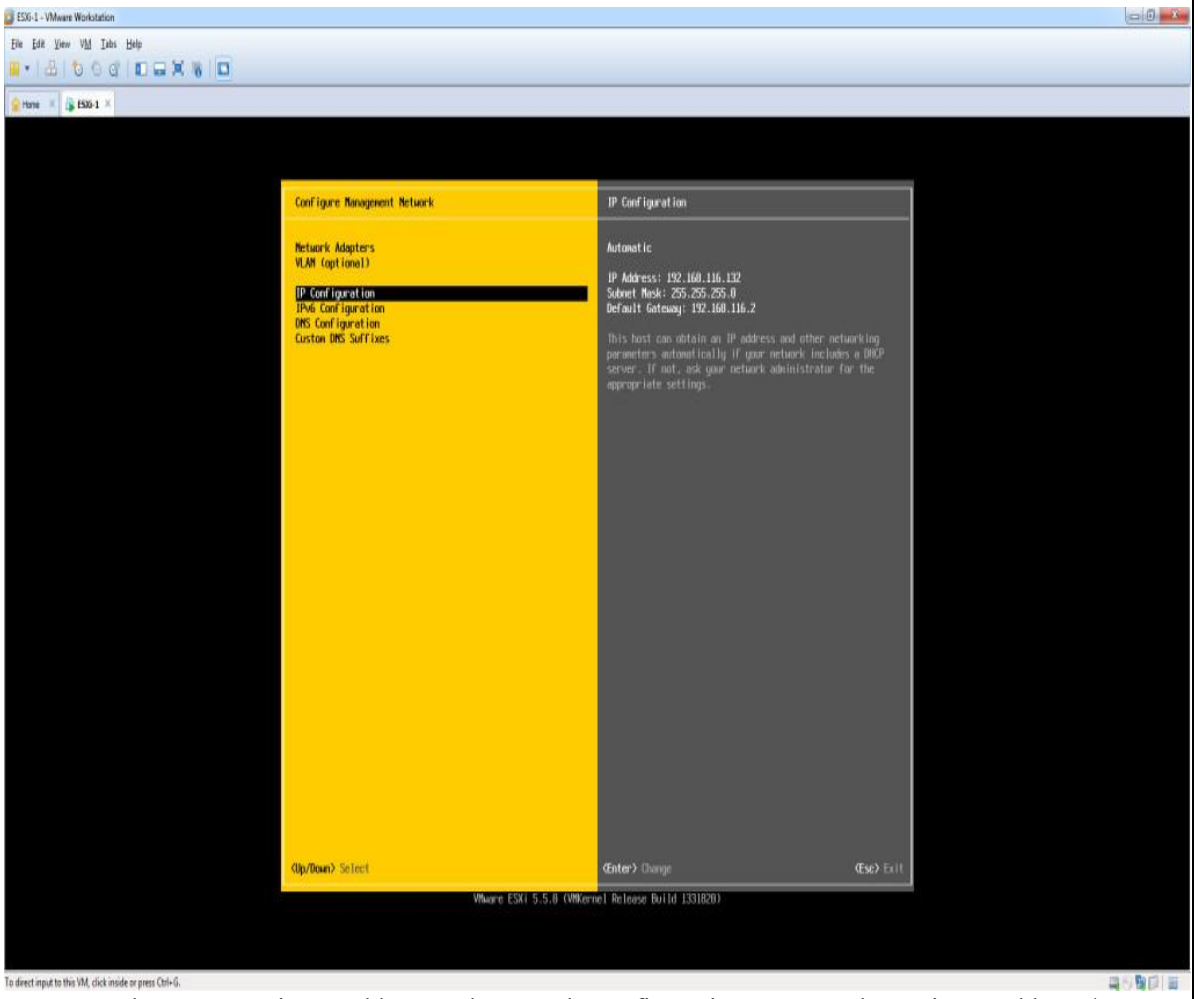
Step 2 – You will be promoted for the password which you created during the installation (note – the default user name is – root).



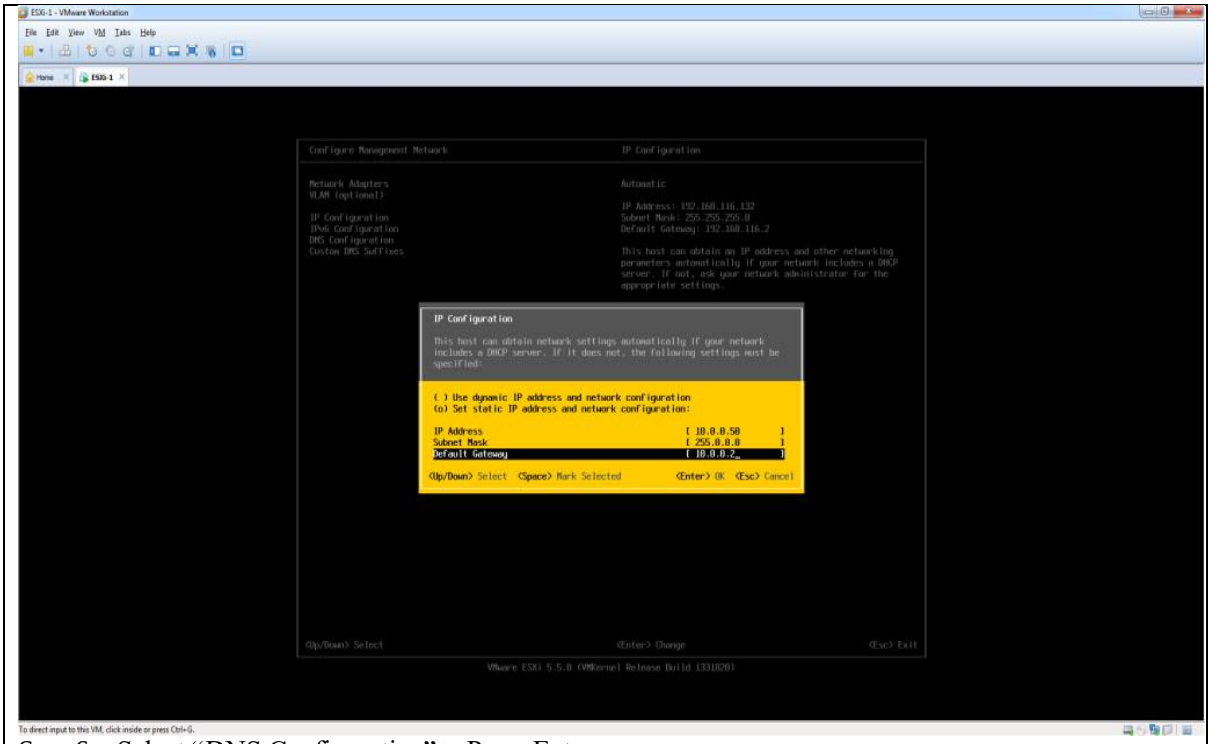
Step 3 – The next step is to configure the management network interface. Select “Configure Management Network” > Press Enter.



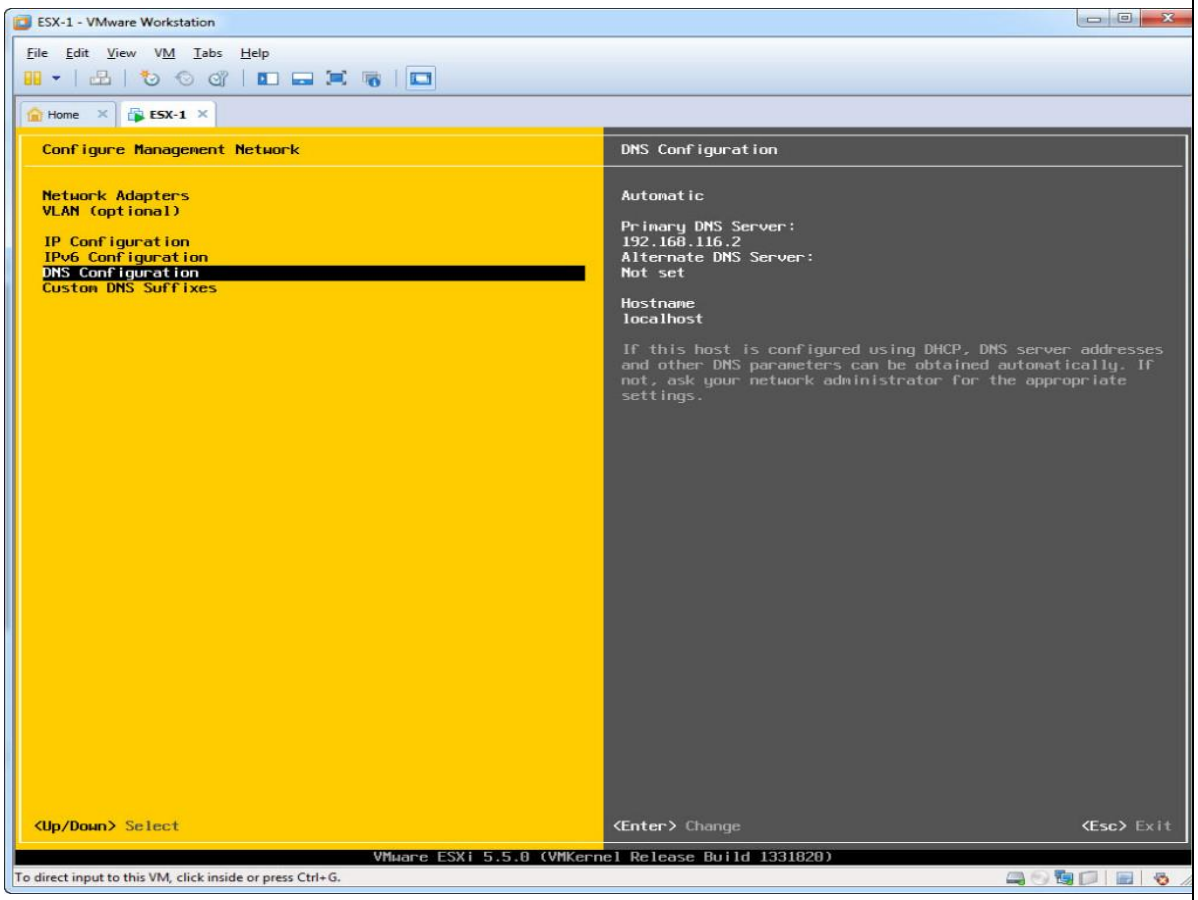
Step 4 – Select IP Configuration > Press Enter



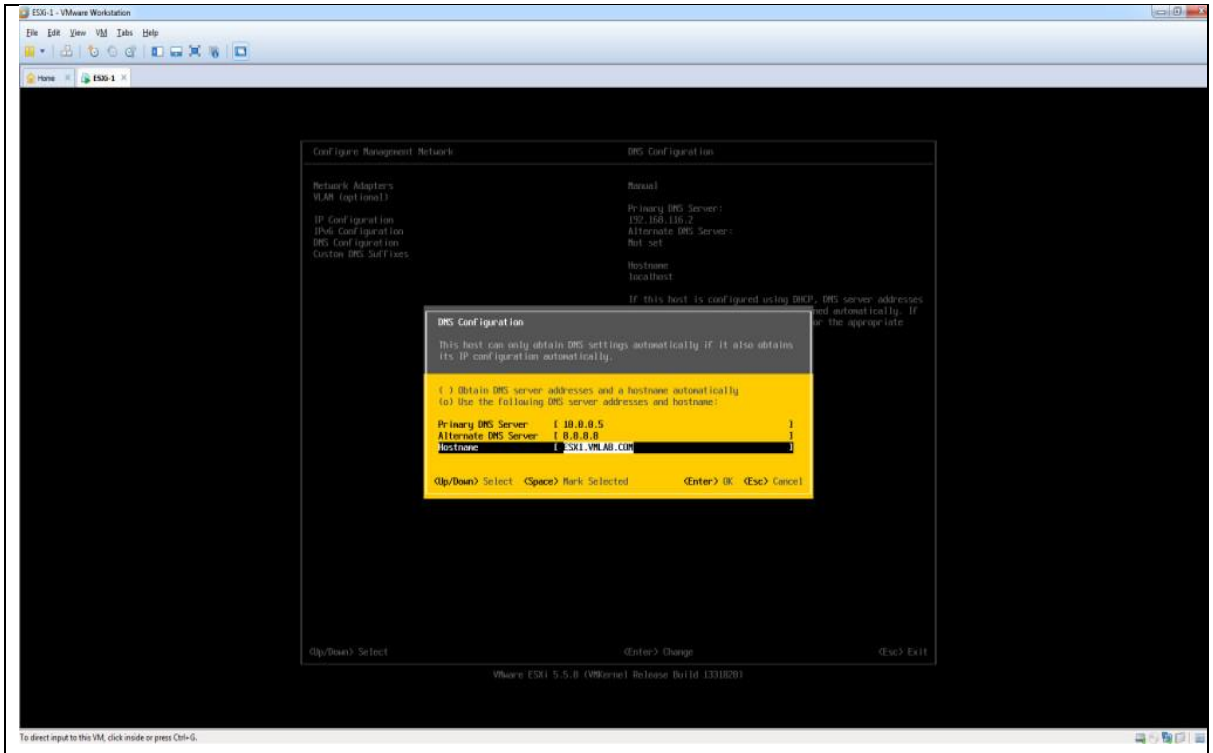
Step 5 – Select “Set Static IP Address and Network Configuration” > Type the static IP Address / Subnet mask and default gateway for ESX1 from the top, or one of your choice > Press Enter once complete.



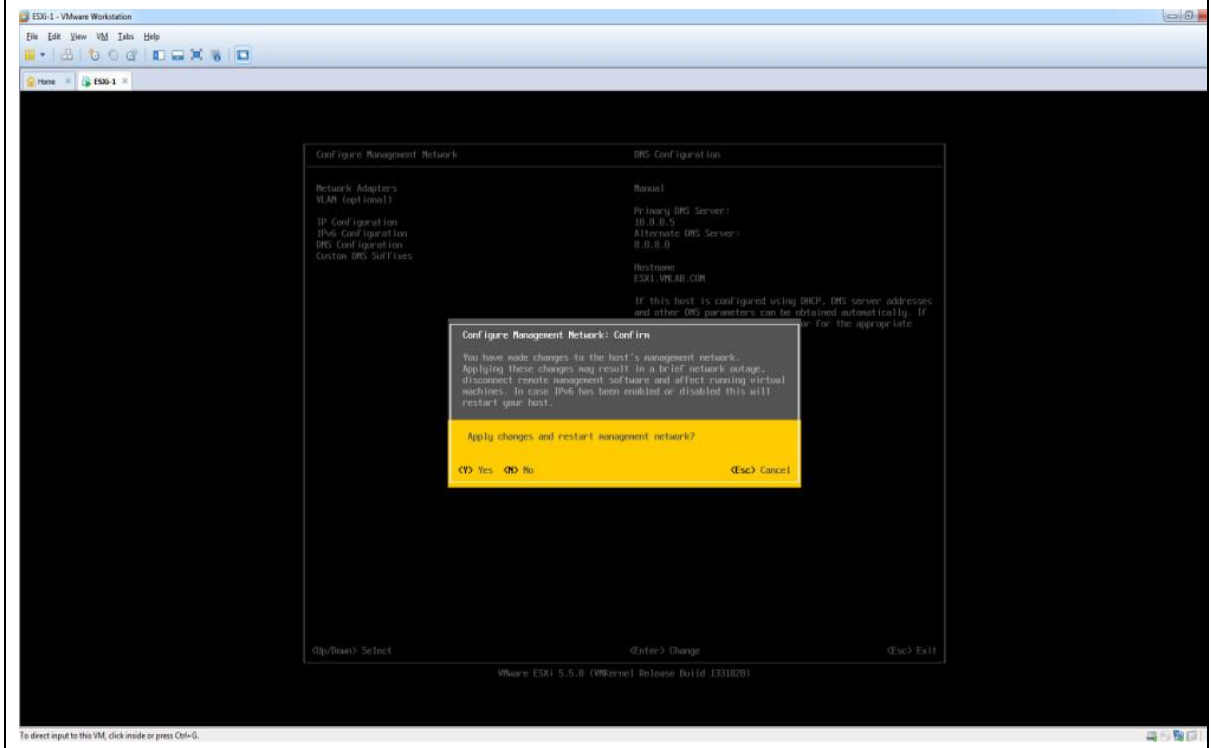
Step 6 – Select “DNS Configuration” > Press Enter



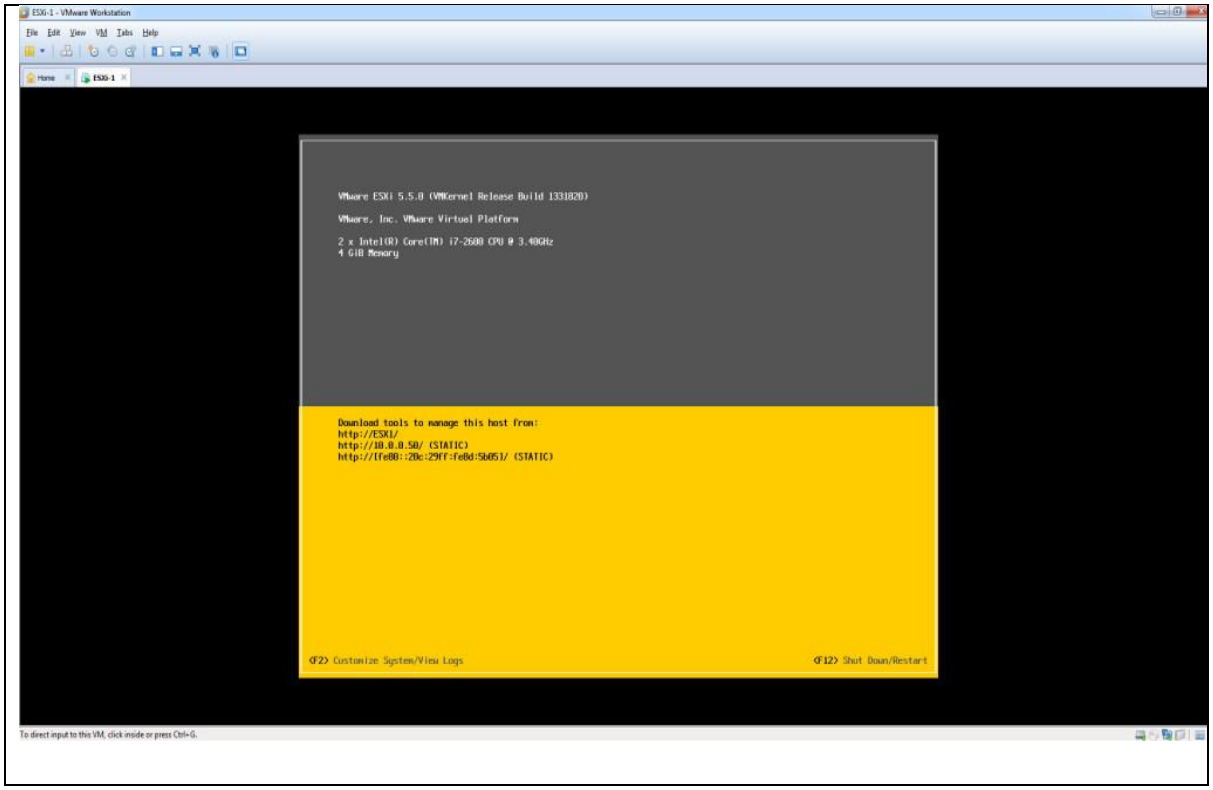
Step 7 – Type the Primary DNS Server / Alternate DNS Server and Hostname for ESX1 from the top, or one of your choice > Press Enter once complete.



Step 8 – You will currently be in the “Configure Management Network” Menu. You have completed all configuration required for ESX 1. To Exit this menu and SAVE configuration > Press ESC > Type [Y] to “Apply changes and restart management network”. This will allow you to successfully exit the ESXi 5.5 configuration menu, back to the home screen.



Step 9 – You have successfully completed configuration for ESXi 5.5 with the host name ESX1.


ESXi-1 - VMware Workstation
File Edit View VM Tabs Help
Home
ESXi-1
VMware ESXi 5.5.0 (VMkernel Release Build 1331820)
VMware, Inc. VMware Virtual Platform
2 x Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
4 GiB Memory
Download tools to manage this host from:
http://ESXI/
http://10.0.0.50/ (STATIC)
http://[fe80::20c:29ff:fe0d:5b051/ (STATIC)
<F2> Customize System/View Logs
<F12> Shut Down/Restart
To direct input to this VM, click inside or press Ctrl+G.

Appendix B

1	**Problem Statement:**

Installation of VMware vSphere Client

2	**Student Learning Outcomes:**

To install and setup the VMware vSphere vClient.

3	**Theoretical Description:**

The vSphere Client is an interface for administering vCenter Server and ESXi.

The vSphere Client user interface is configured based on the server to which it is connected:

When the server is a vCenter Server system, the vSphere Client displays all the options available to the vSphere environment, according to the licensing configuration and the user permissions.
When the server is an ESXi host, the vSphere Client displays only the options appropriate to single host management

4	**Requirements**

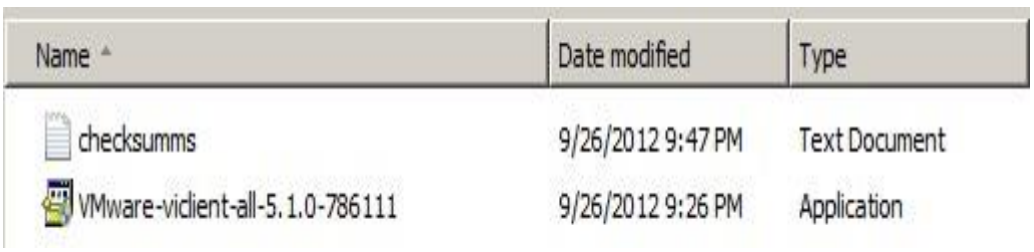
Make sure that the vSphere Client hardware meets the minimum requirements.

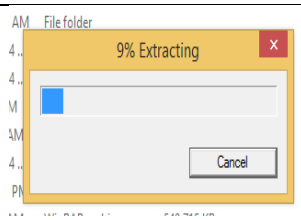
- CPU – 1 CPU
- Processor – 500MHz or faster Intel or AMD processor (1GHz recommended)
- Memory – 1GB RAM
- Disk Storage – 1.5GB free disk space for a complete installation, which includes the following components:
  - Microsoft .NET 2.0
  - Microsoft .NET 3.0 SP1
  - Microsoft Visual J#
- Remove any previously installed versions of Microsoft Visual J# on the system where you are installing the vSphere Client.
- vSphere Client 6
- Networking – Gigabit connection recommended

5	**Procedure**

Login to any windows operating system where you want to configure vSphere Client to manage vCenter Server.

1. Copy the “VMware vSphere Client 6” directory to windows host and execute the exe file.

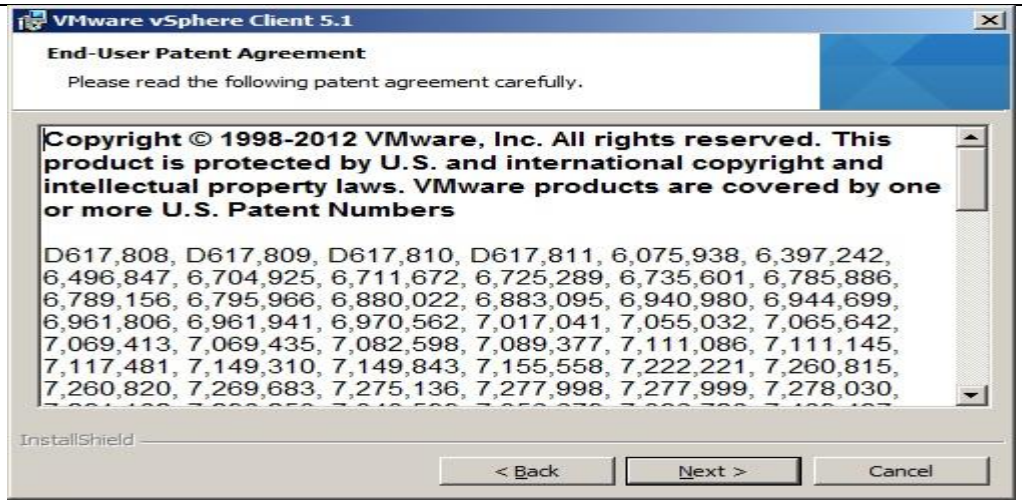




2. Press OK to continue the installation in English.

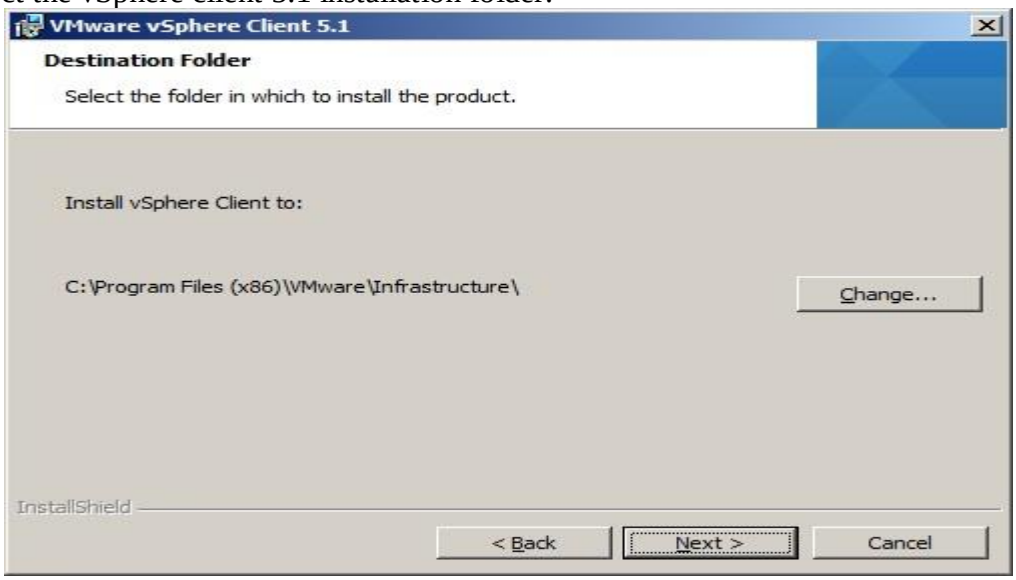
3.Click Next to continue the installation.

4.End User patent agreement, Click next to continue.

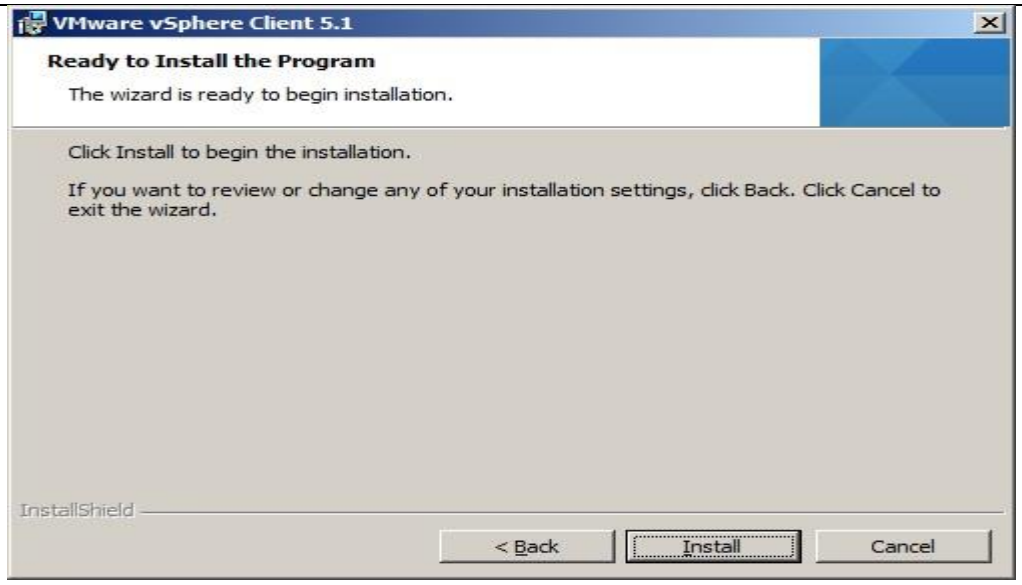


5.Agree the license agreement and continue the installation.

6.Select the vSphere client 5.1 installation folder.

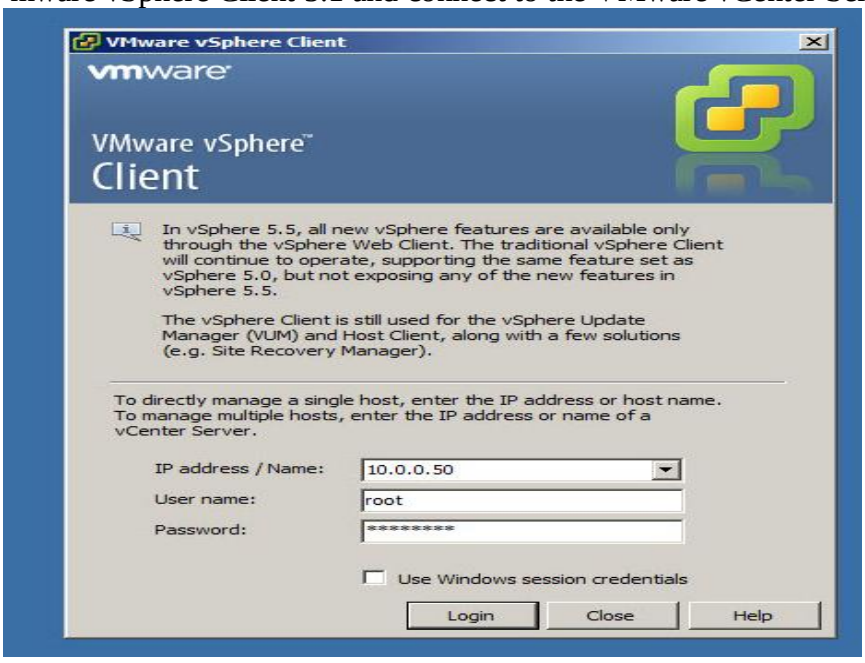


7.Click install to begin the installation.

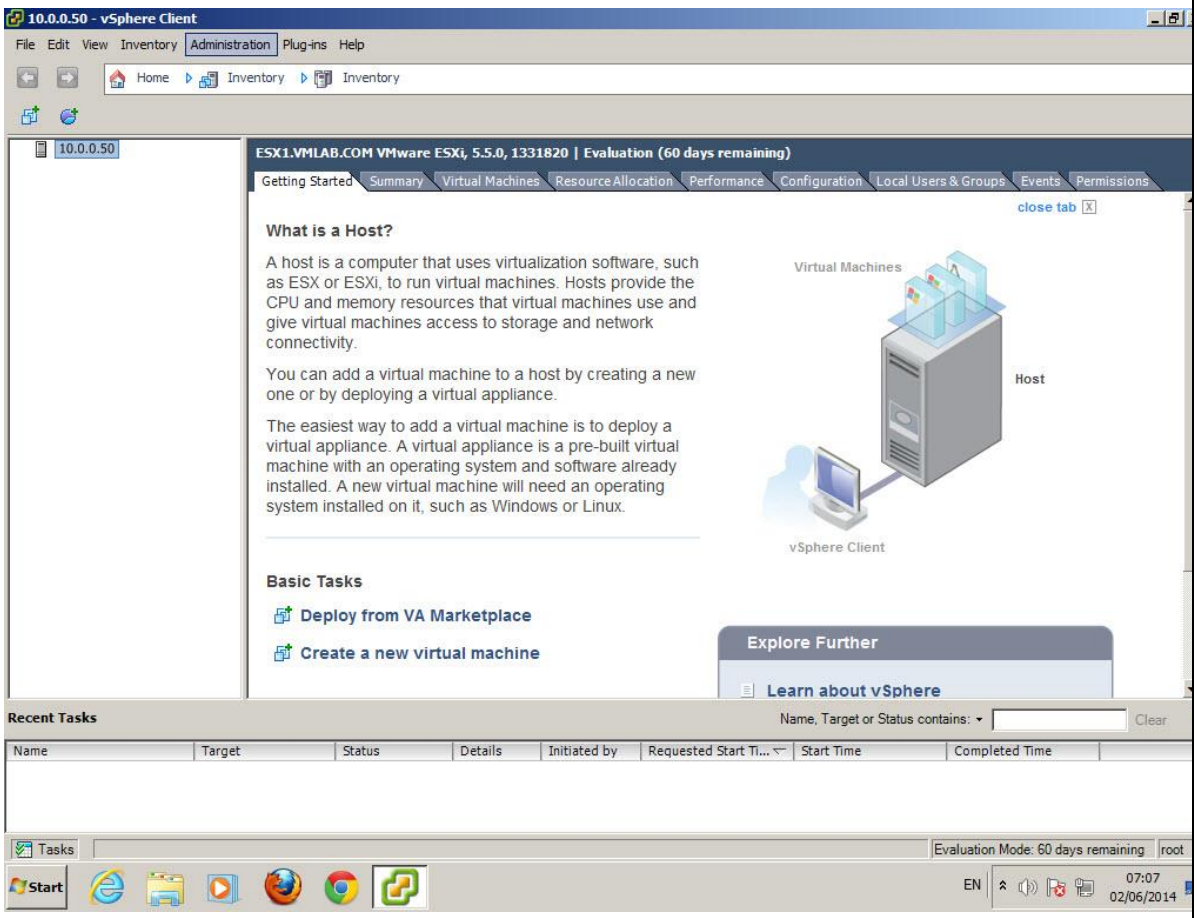


8. Click finish to complete the installation. The installation is so simple like other windows software's.

9.Start the Vmware vSphere Client 5.1 and connect to the VMware vCenter Server.



10.You are done with vSphere client installation and verified by logging in to vCenter server.


10.0.0.50 - vSphere Client
File Edit View Inventory Administration Plug-ins Help
Home Inventory Inventory
10.0.0.50
ESX1.VMLAB.COM VMware ESXi, 5.5.0, 1331820 | Evaluation (60 days remaining)
Getting Started Summary Virtual Machines Resource Allocation Performance Configuration Local Users & Groups Events Permissions
close tab
What is a Host?
A host is a computer that uses virtualization software, such as ESX or ESXi, to run virtual machines. Hosts provide the CPU and memory resources that virtual machines use and give virtual machines access to storage and network connectivity.
You can add a virtual machine to a host by creating a new one or by deploying a virtual appliance.
The easiest way to add a virtual machine is to deploy a virtual appliance. A virtual appliance is a pre-built virtual machine with an operating system and software already installed. A new virtual machine will need an operating system installed on it, such as Windows or Linux.
Virtual Machines
Host
vSphere Client
Basic Tasks
Deploy from VA Marketplace
Create a new virtual machine
Explore Further
Learn about vSphere
Recent Tasks
Name, Target or Status contains:
Clear
Name Target Status Details Initiated by Requested Start Ti... Start Time Completed Time
Tasks
Evaluation Mode: 60 days remaining
root
Start
EN
07:07
02/06/2014

Appendix C

1	**Problem Statement:**

Install and configure the vSphere Web client

2	**Student Learning Outcomes:**

To install and configure the vSphere web Client to access the vCenter server

Create a final data center for different VM's

3	**Theoretical Description:**

vSphere Web client is a

- Web application.
- Cross platform.
- Can connect to only vCenter Server.
- Subset of full functionality, focused on virtual machine deployment and basic monitoring functions. Cannot configure hosts, clusters, networks, datastores, or datastore clusters
- Extensible plug-in-based architecture.

4	**Requirements**

- Download the vCenter Server installer.
- Verify that you are a member of the Administrators group on the system.
- Verify that the system has an Internet connection.
- Verify that the system meets the software requirements for the vSphere Web Client. See vSphere Client and vSphere Web Client Software Requirements. The vSphere Web Client requires a 64-bit operating system for installation.

5	**Procedure**

1. In the software installer directory, double-click the autorun.exe file to start the installer.
2. Select VMware vSphere® Web Client (Server) and click Install.
3. Follow the wizard prompts to complete the installation
   When the vSphere Web Client installation is finished, a browser opens.
4. Register one or more vCenter Servers on the vSphere Web Client Administration Application page in the browser.
5. If the browser fails to open or to display the Administration Application page correctly, open the application from the shortcut
   From the Windows Start menu, select Programs > VMware > VMware vSphere Web Client > vSphere Administration Application.

Appendix D

1	**Problem Statement:**

Illustration of vMotion to move the VM's from ESXi to ESXi

2	**Student Learning Outcomes:**

To install and configure the vSphere web Client to access the vCenter server

Create a final data center for different VM's

Identify the way of moving the VM's from one ESXi to another.

Migrate the VM's from one to another ESXi.

3	**Theoretical Description:**

VMware VMotion enables the live migration of running virtual machines from one physical server to another with zero downtime, continuous service availability, and complete transaction integrity. It is transparent to users.

VMotion lets you:

- Automatically optimize and allocate entire pools of resources for maximum hardware utilization and availability.
- Perform hardware maintenance without any scheduled downtime.
- Proactively migrate virtual machines away from failing or underperforming servers.

4	**Requirements**

- Migration with vMotion requires a correctly configured vMotion network interface on both source and target hosts.
- Configure each host with at least one vMotion network interface. To ensure secure data transfer, the vMotion network must be a secure network, accessible only to trusted parties. Because vMotion performance improves significantly with additional bandwidth, dedicate at minimum a physical 1 Gigabit Ethernet (GigE) NIC to vMotion. As a best practice, provision at least one additional physical NIC as a failover NIC.
- Ensure that virtual machines have access to the same subnets on source and destination hosts.
- If you are using standard switches for networking, ensure that the network labels used for virtual machine port groups are consistent across hosts.
- During a migration with vMotion, vCenter Server assigns virtual machines to port groups. Configure hosts for vMotion with shared storage to ensure that virtual machines are accessible to both source and target hosts.
- If you use vMotion to migrate virtual machines with raw device mapping (RDM) files, ensure that the LUN IDs for RDMs are consistent across all participating hosts.

5	**Procedure**

Configure a vMotion interface using vSphere Client

To configure a vMotion Interface:

1. Log into the vCenter Server using vSphere Client.
2. Click to select the host.
3. Click the **Configuration** tab.
4. Click **Networking** under Hardware.
5. Click **Add Networking**.
6. Select **VMkernel** and click **Next**.
7. Select the existing vSwitch, or select **Create a vSphere standard switch** to create a new vSwitch and click **Next**.
8. Enter a name in the **Network Label** to identify the network that vMotion uses.
9. Select a VLAN ID from the **VLAN ID (Optional)** dropdown if applicable.
10. Select **Use this port group for vMotion** and click **Next**.
11. Enter the **IP address** and **Subnet Mask** of the host's vMotion Interface.
12. Click **Next**, then click **Finish**.

    Note: For multiple hosts, repeat steps 2 to 12. Use a unique IP address for each host vMotion interface.

Configure a vMotion interface using vSphere Web Client:

1. In the vSphere Web Client, navigate to the Host.
2. Under **Manage**, select **Networking** and then select **VMkernel adapters**.
3. Click **Add host networking**.
4. On the Select connection type page, select **VMkernel Network Adapter** and click **Next**.
5. On the Select target device page, select either an **existing standard switch** or a **New vSphere standard switch**.
6. On the **Port properties**, enable vMotion Traffic and select **Next.**
7. Configure network for the vMotion VMkernel interface and click **Next**.
8. Review the settings and click **Finish**.