SRI SIVASUBRAMANIYA NADAR

COLLEGE OF ENGINEERING

(AFFILIATED TO ANNA UNIVERSITY, CHENNAI)

RAJIV GANDHI SALAI (OMR), KALAVAKKAM – 603110

LABORATORY RECORD

IT8711-FOSS and Cloud Computing Laboratory

Name : .....G Divyadharshini.....

Reg. No. : .....312217205026.....

Dept. : .....IT..... Sem.: .....VII..... Sec.: ...A.....

ANNA UNIVERSITY

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of

the practical work done for the

.....**IT8711-FOSS and Cloud Computing Laboratory**..... lab

by

Name .....G Divyadharshini.....

Register Number .....312217205026.....

of Department of .....INFORMATION TECHNOLOGY.....

Sri Sivasubramaniya Nadar College of Engineering,

Kalavakkam, Chennai

during the academic year ....2020-2021.

Staff In-Charge

Head

Department of .....IT.....

Submitted for the University Examination held at

Sri Sivasubramaniya Nadar College of Engineering on .....

Internal Examiner

External Examiner

INDEX

Name: _____G Divyadharshini_____ Reg. No.: ___312217205026_____

Sem. : _____VII_____ Sec. : _____A_____

Ex: 1 **Make File Creation**

Date: 19.08.2020

Objective:

To understand the Ubuntu software and to understand the working of make command.

Requirement:

Ubuntu software.

Theory:

Make:

make is typically used to build executable programs and libraries from source code. Generally speaking, make is applicable to any process that involves executing arbitrary commands to transform a source file to a target result. For example, make could be used to detect a change made to an image file (the source) and the transformation actions might be to convert the file to some specific format, copy the result into a content management system, and then send e-mail to a predefined set of users that the above actions were performed.

make is invoked with a list of target file names to build as command-line arguments:

make [target]

Without arguments, make builds the first target that appears in its makefile, which is traditionally a target named all. Make decides whether a target needs to be regenerated by comparing file modification times. This solves the problem of avoiding the building of files that are already up to date, but it fails when a file changes but its modification time stays in the past. Such changes could be caused by restoring an older version of a source file, or when a network filesystem is a source of files and its clock or time zone is not synchronized with the machine running make. The user must handle this

situation by forcing a complete build. Conversely, if a source file's modification time is in the future, it may trigger unnecessary rebuilding.

Makefiles:

make searches the current directory for the makefile to use.
GNU make searches files for a file named one of GNUmakefile, makefile, and then Makefile, and runs the specified target(s) from that file.

The makefile language is similar to declarative programming, in which necessary end conditions are described but the order in which actions are to be taken is not important. This may be confusing to programmers used to imperative programming, which explicitly describes how the end result will be reached.

One problem in build automation is the tailoring of a build process to a given platform. For instance, the compiler used on one platform might not accept the same options as the one used on another. This is not well handled by make on its own. This problem is typically handled by generating separate platform-specific build instructions, which in turn may be processed by make. Common tools for this process are autoconf and cmake.

Procedure and implementation:

- Install Ubuntu software.
- Install GCC command.
- Create grades.h, grades.c, adjust.c and Makefile

grades.h:

```
int factorial (int n);
```

grades.c:

```
#include "grades.h"

int factorial (int n) {

if (n! =1)

    return (n * factorial (n - 1));

else
```

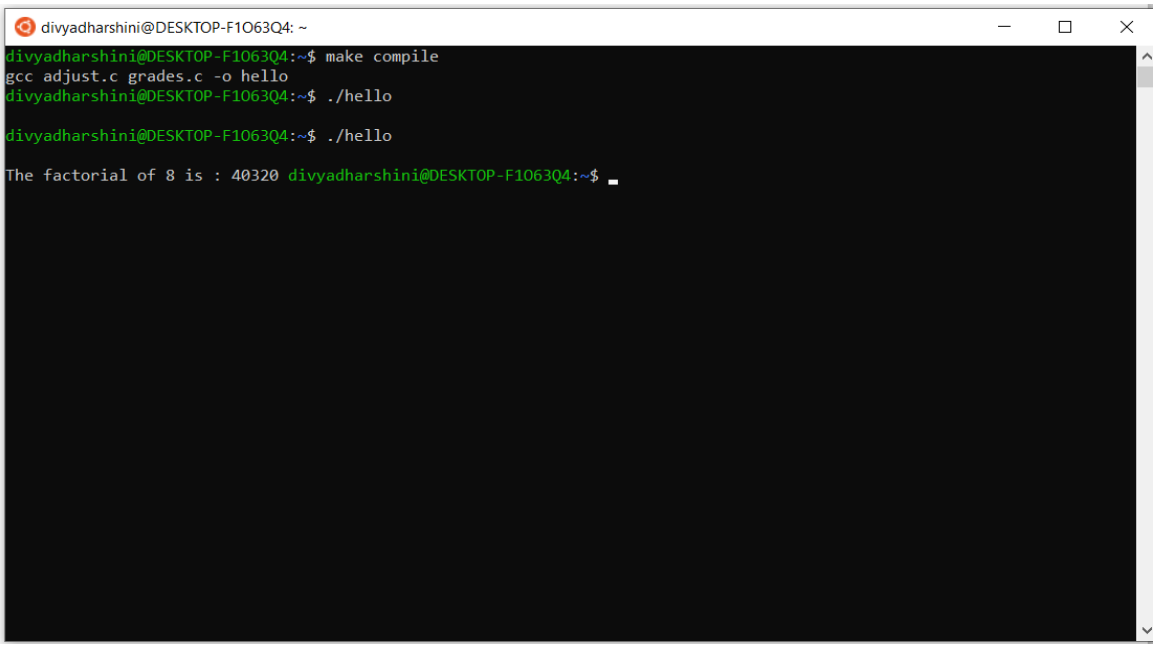
```
    return 1;
}
```

adjust.c:

```
#include<stdio.h>
#include "grades.h"
int main (){
  printf ("\n the factorial of 8 is: %d", factorial (8));
  return 0;
}
```

Makefile

```
all:
compile:
gcc grades.c adjust.c  -o hello using make command.
```



Result:

The make command has been successfully executed using Ubuntu software.

Ex: 2 **Maintaining repositories using Version Control System**

Date: 26.08.2020

Objective:

To get familiar working with Git and Github through Git GUI.

Requirement:

- ❖ Git GUI.
- ❖ Google Chrome.
- ❖ Github account.

Theory:

Version Control System:

Version control systems are a category of software tools that helps record changes to files by keeping a track of modifications done to the code.

Use of Version Control System:

- **A repository:** It can be thought as a database of changes. It contains all the edits and historical versions (snapshots) of the project.
- **Copy of Work (sometimes called as checkout):** It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.

Purpose of Version Control:

- Multiple people can work simultaneously on a single project. Everyone works on and edits their own copy of the files and it is up to them when they wish to share the changes made by them with the rest of the team.
- It also enables one person to use multiple computers to work on a project, so it is valuable even if you are working by yourself.
- It integrates the work that is done simultaneously by different members of the team. In some rare case, when conflicting edits are made by two people to the same line of a file, then human assistance is requested by the version control system in deciding what should be done.
- Version control provides access to the historical versions of a project. This is insurance against computer crashes or data loss. If any mistake is made, you can easily roll back to a previous version. It is also possible to undo

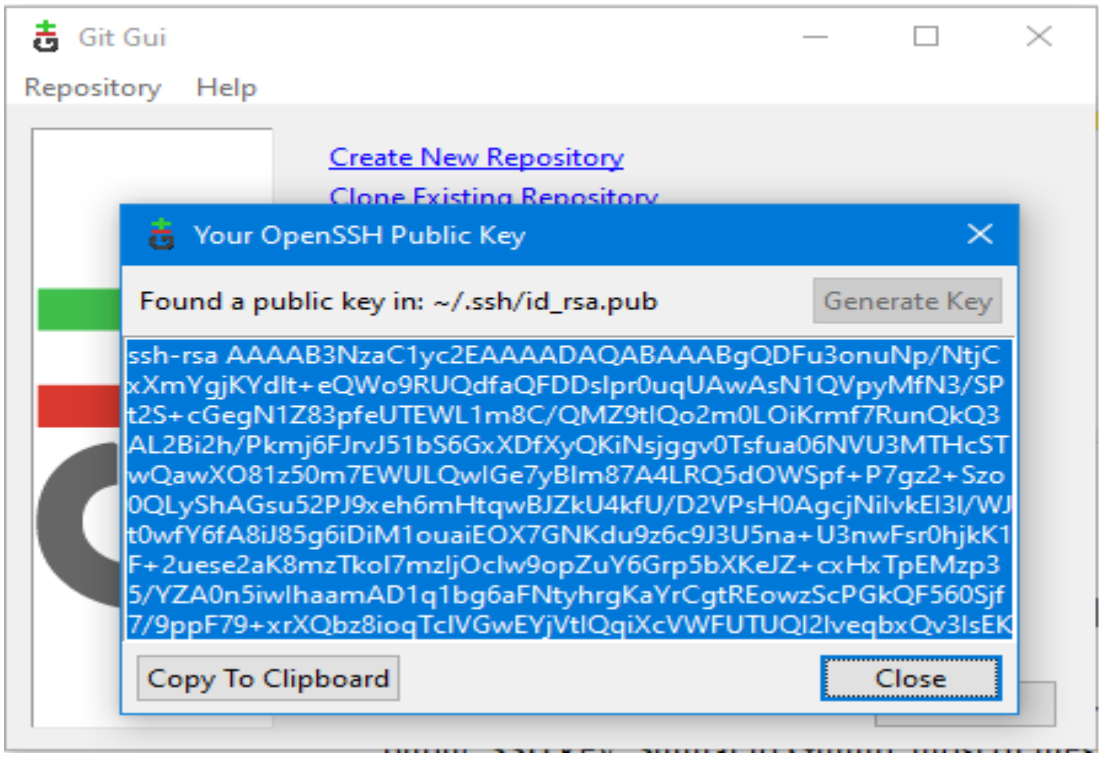
specific edits that too without losing the work done in the meanwhile. It can be easily known when, why, and by whom any part of a file was edited.

GIT:

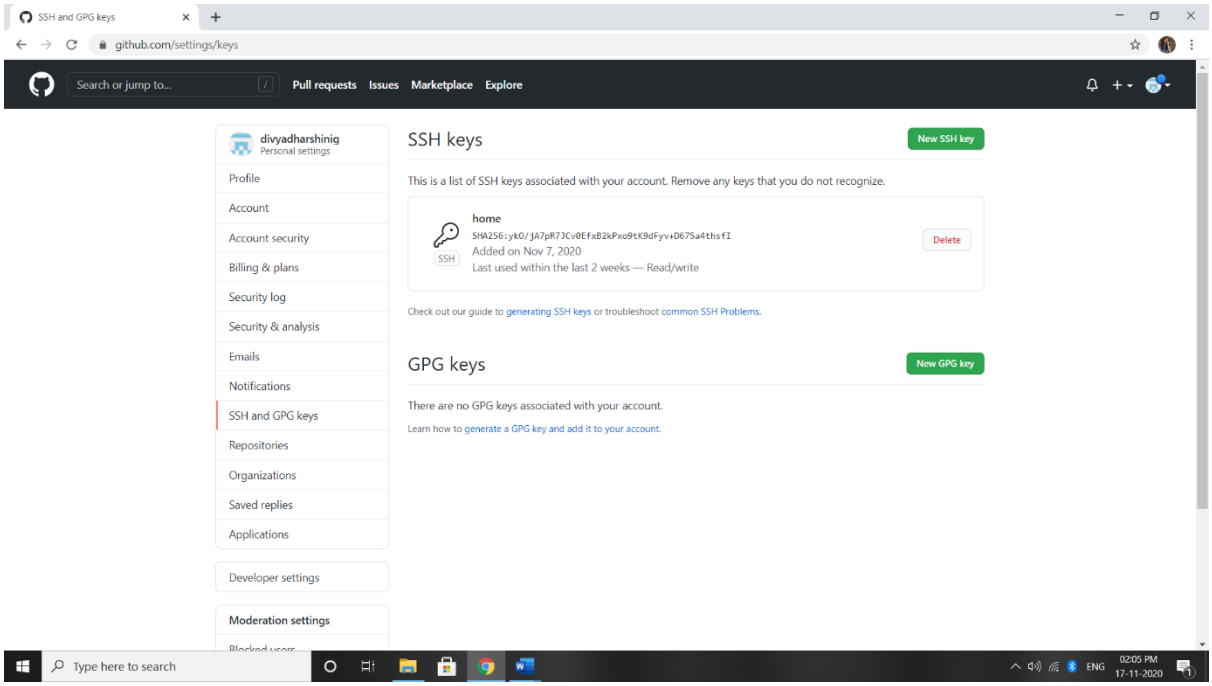
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance.

Procedure and implementation:

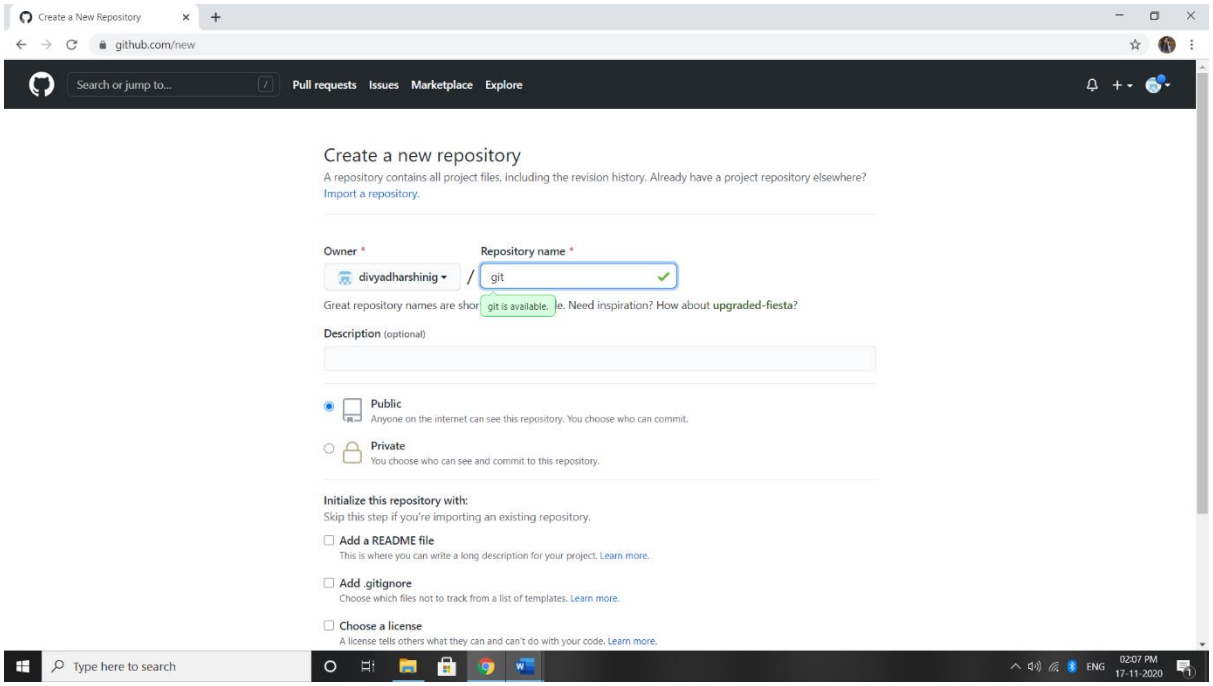
- The first two things you'll want to do are install git and create a free GitHub account.
- Install Git
- First, you need to install on windows; to do so, follow the below steps.
- Download and install the latest version of https://github.com/git-forwindows/git/releases/download/v2.28.0.windows.1/Git-2.28.0-64-bit.exe
- Use the default options for each step in the installation
- Remove Git Bash Desktop Icon
- Go to Start -> All Programs -> Git -> Git GUI and make a Desktop Shortcut.
- Now click on Show SSH Key under the Help Menu.
- It's possible that there is already a SSH key on your computer; it's best to remove or backup the key if you do not know where it came from.
- To do so, simply remove all files within: C:\Users\\.ssh. Be sure to replace with your Windows username. You can generate a SSH key by clicking on the Generate Key button.
- When you do so, you will need to supply a passphrase for security purposes. Remember this passphrase; you will need to use it later.



- click the "Copy to Clipboard" button.
- Next, you need to provide your hosted repo service with your public SSH key. Similar to Github, most of these sites usually have a tab, called "SSH Keys". Click the tab and add your SSH key to the website.

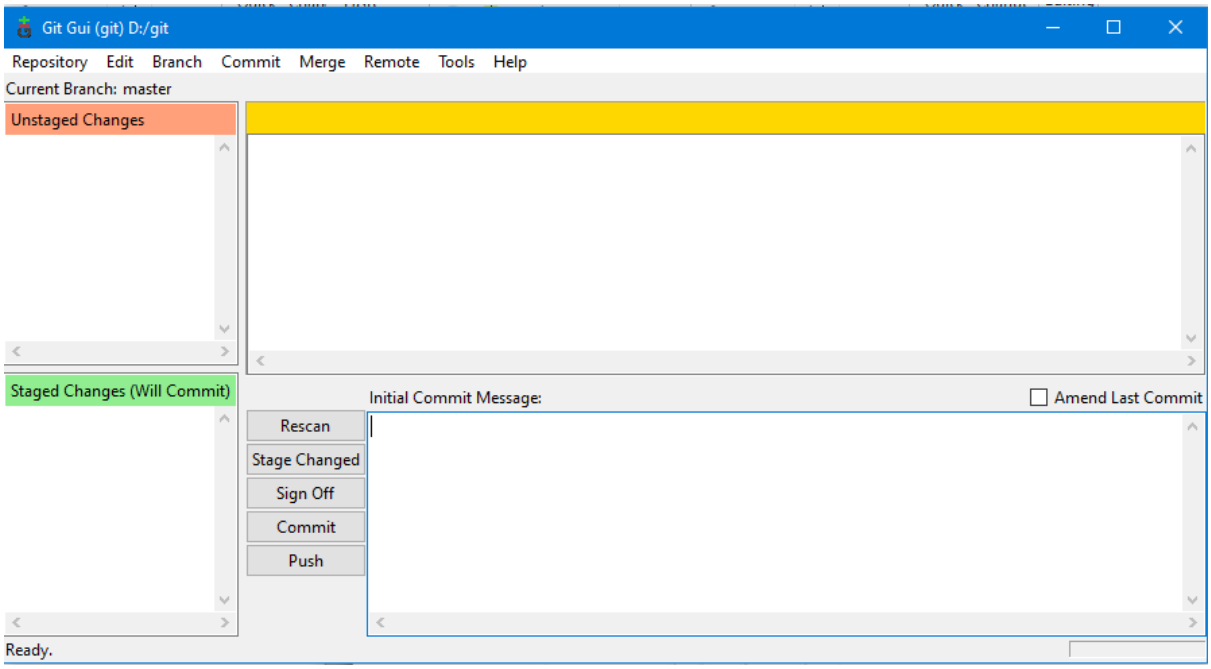


- create new remote repository on Github

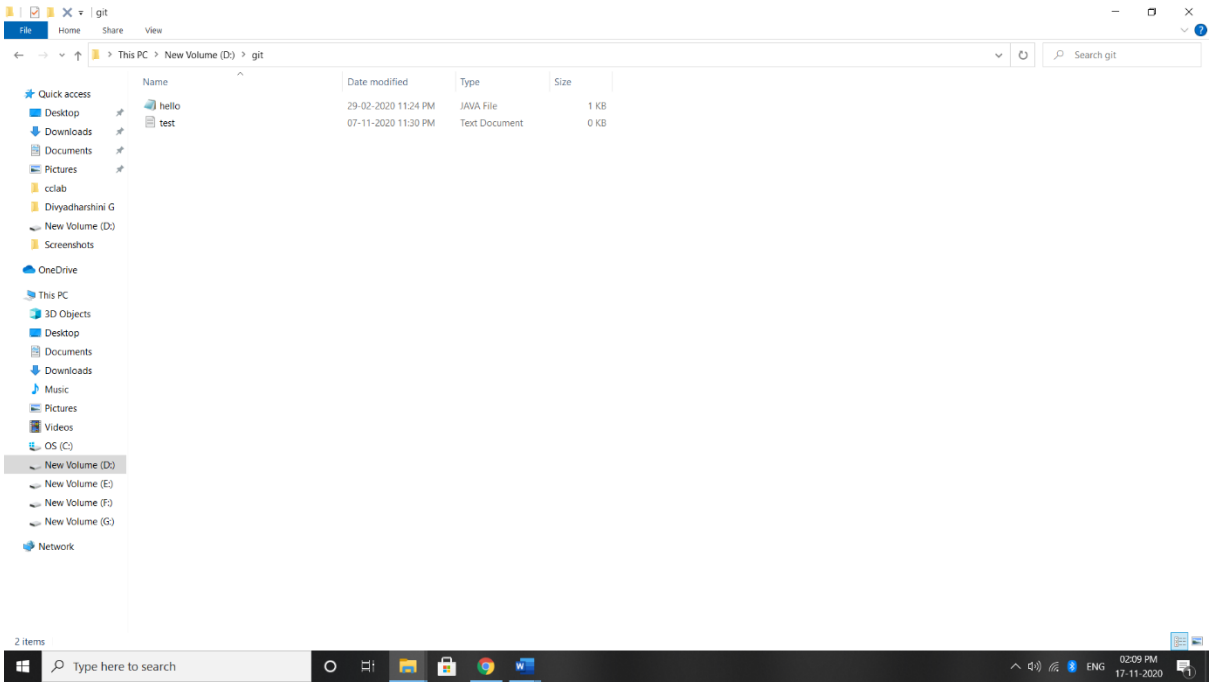


- Create a Local Repository In the Git GUI; clicking on "Create New Repository".

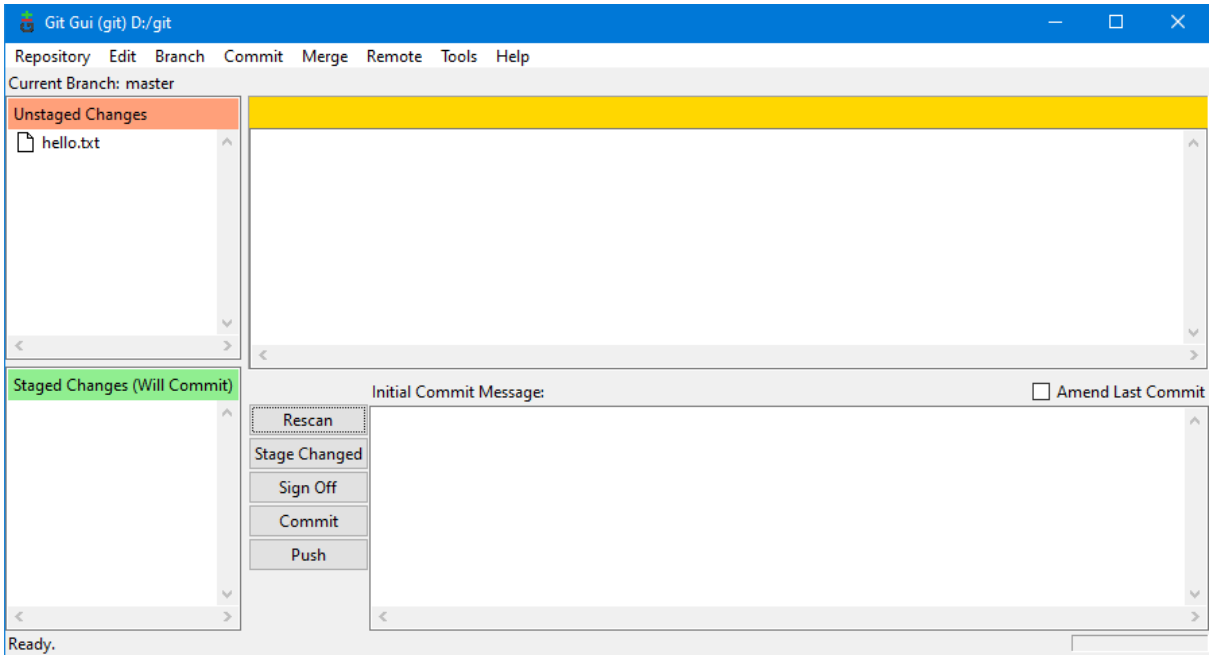
❖ Create new repository



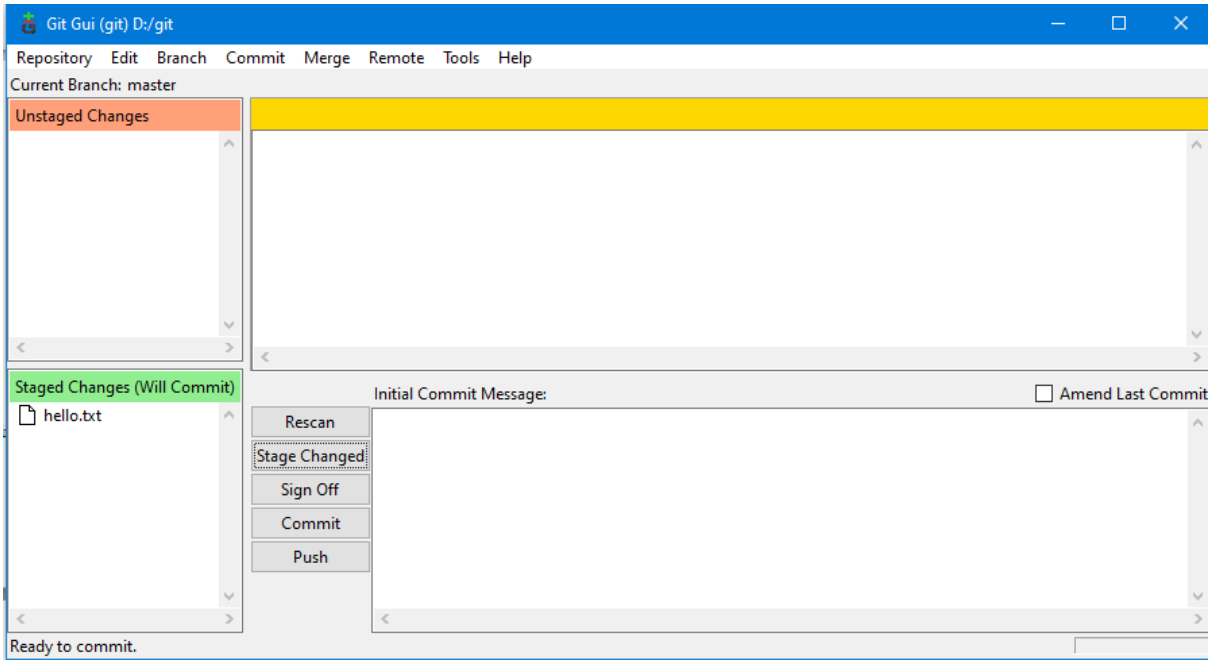
❖ Add a file in that folder.



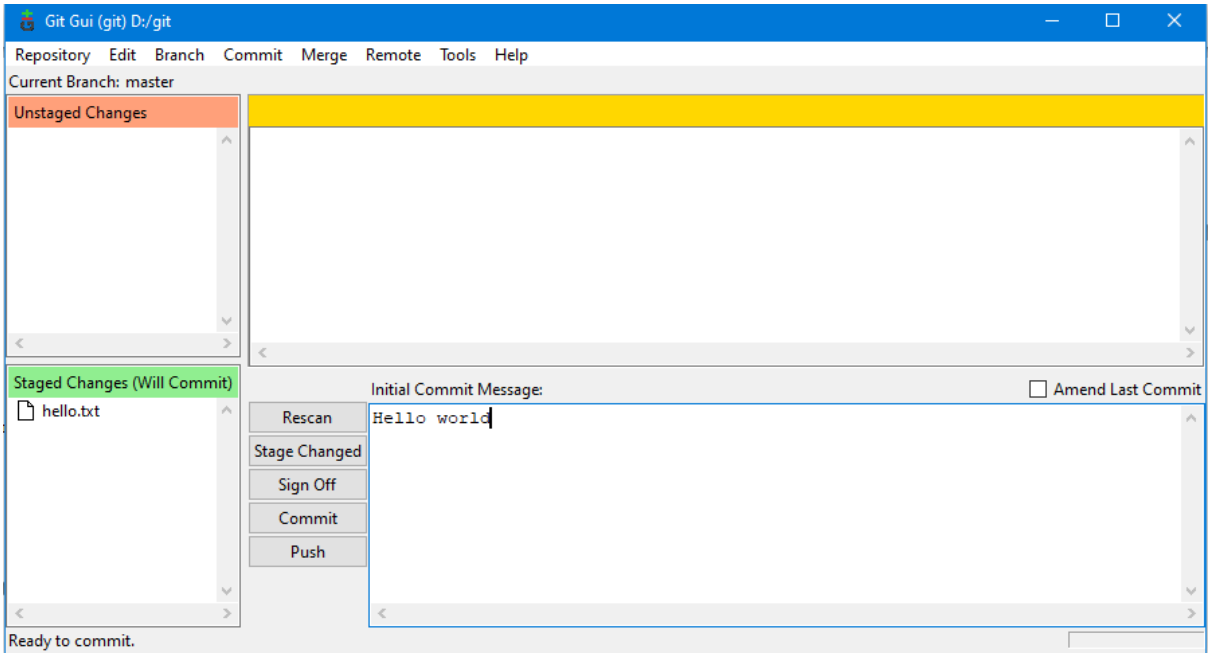
❖ Click rescan.



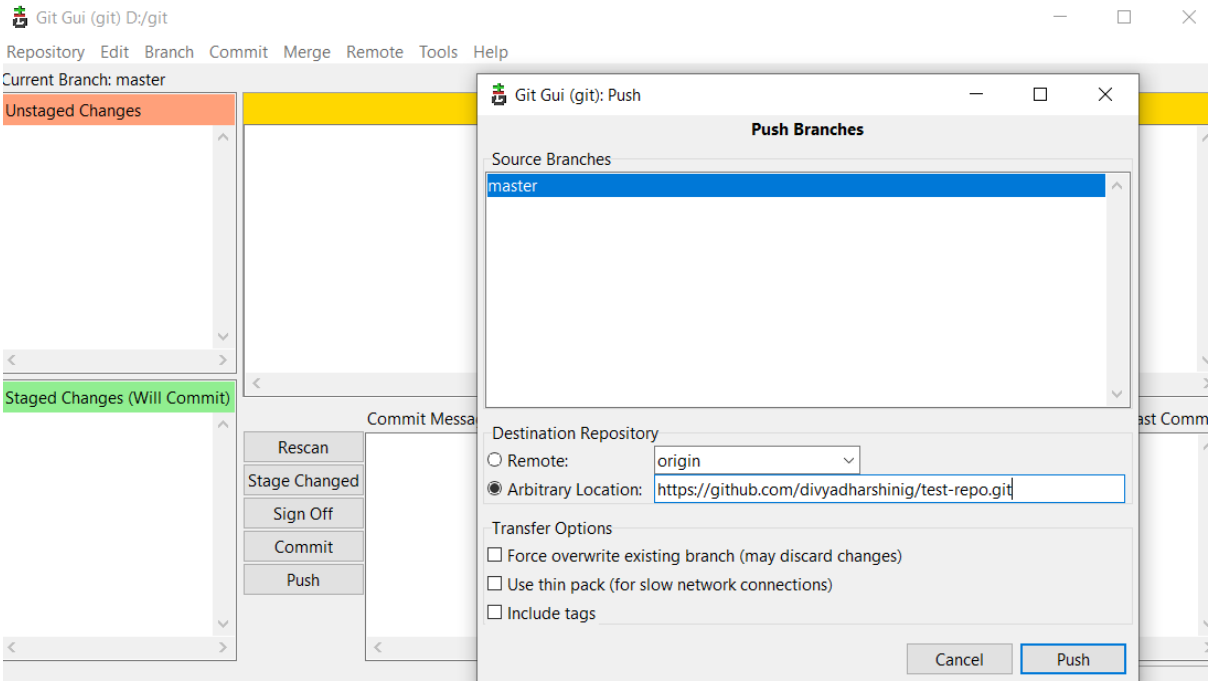
❖ Click Stage Changed.



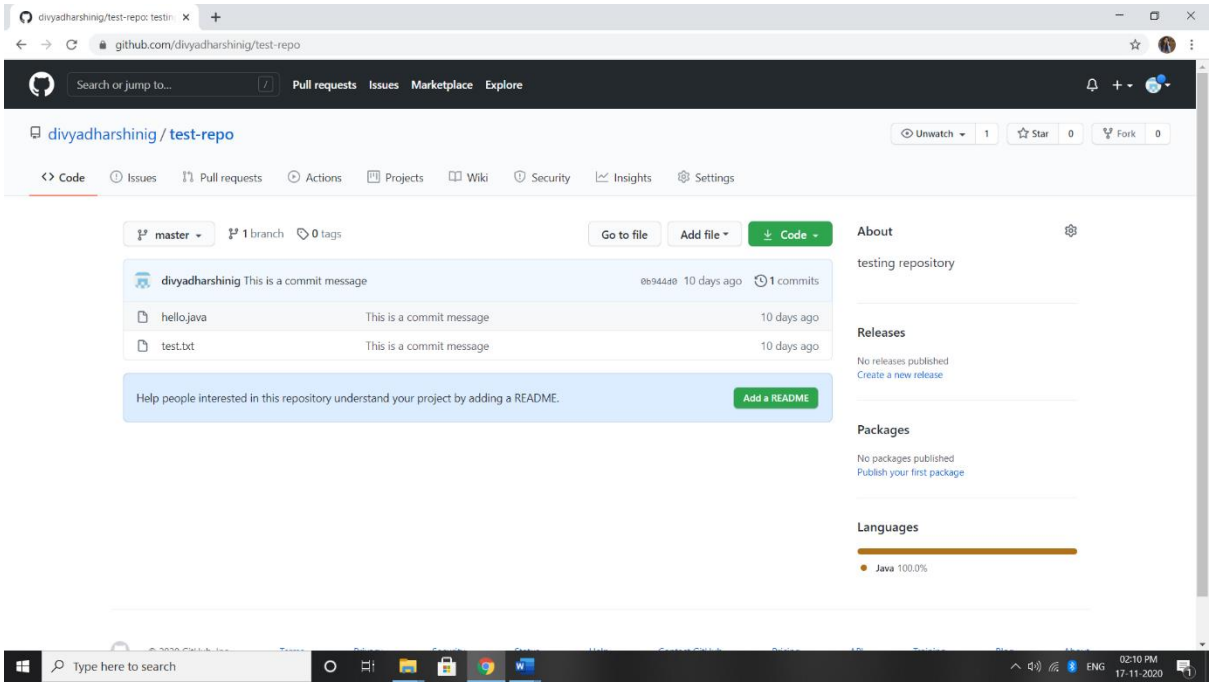
❖ Type message and click commit.



❖ Click Push and add arbitrary location as github repositary link.



❖ Now the files are copied to github repository.



Result:

Thus creating and maintaining repositories using Version control system has been successfully done.

Ex: 3 **Virtualbox /VMware Workstation**

Date: 02.09.2020

Aim:

To understand the working of VMware Workstation and to understand the installation procedure and creation of VMs.

Requirement:

- Google Chrome (or) Related Internet Explorer.
- Ubuntu iso file.

Theory:

Architecture:

VMware Workstation virtualizes I/O devices using a novel design called the *Hosted Virtual Machine Architecture*. The primary feature of this design is that it takes advantage of a pre-existing operating system for I/O device support and still achieves near native performance for CPU-intensive workloads.

VMware Workstation installs like a normal application on an operating system, known as the host operating system. When run, the application portion (**VMApp**) uses a driver loaded into the host operating system (**VMDriver**) to establish the privileged virtual machine monitor component (**VMM**) that runs directly on the hardware. From then on, a given physical processor is executing either the host world or the VMM world, with the VMDriver facilitating the transfer of control between the two worlds. A world switch between the VMM and the host worlds involves saving and restoring *all* user and system visible state on the CPU, and is thus more heavyweight than a normal process switch.

In this architecture, the CPU virtualization is handled by the VMM. A guest application or operating system performing pure computation runs just like a traditional mainframe-style virtual machine system. However, whenever the guest performs an I/O operation, the VMM will intercept it and switch to the host world rather than accessing the native hardware directly. Once in the host world, the VMApp will perform the I/O on behalf of the virtual machine through appropriate system calls. For example, an attempt by the guest to fetch sectors from its disk will become a read() issued to the host for the corresponding data. The VMM also yields control to the host OS upon receiving a hardware interrupt. The hardware interrupt is reasserted in the host world so that the host OS will process the interrupt as if it came directly from hardware.

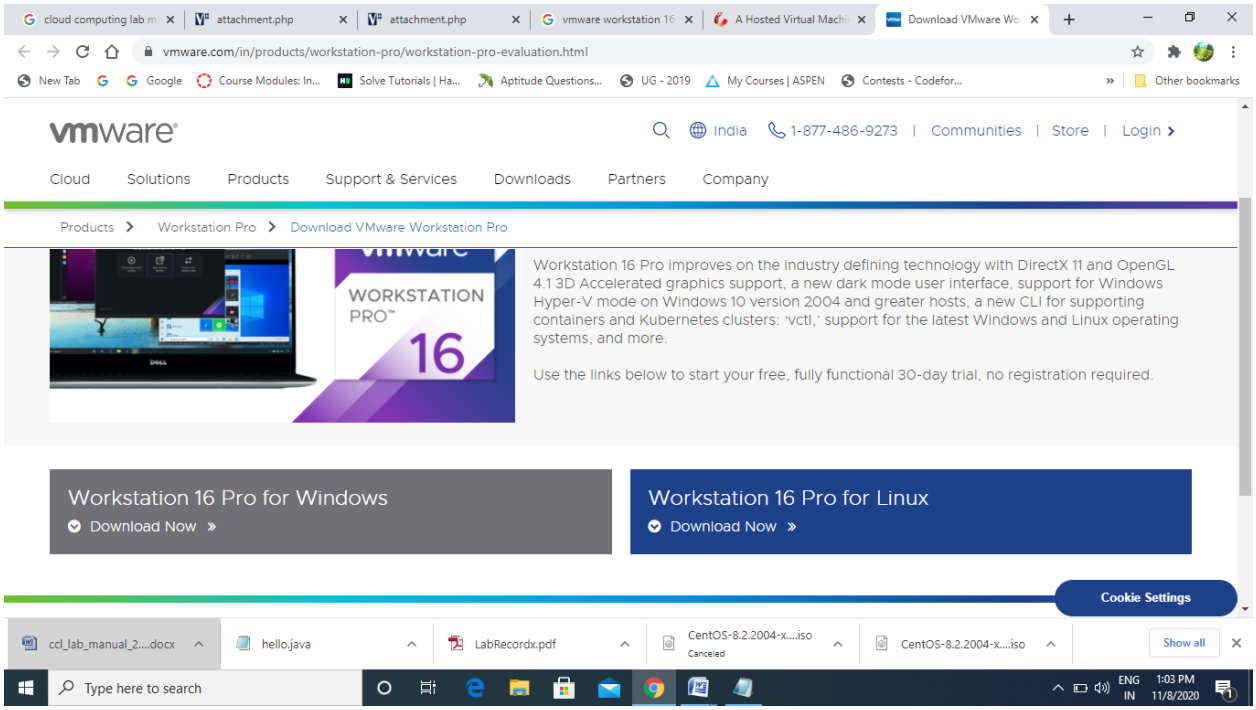
The hosted architecture is a powerful way for a PC-based virtual machine monitor to cope with the vast array of available hardware. One of the primary purposes of an operating system is to present applications with an abstraction of the hardware that allows hardware-independent code to access the underlying devices. For example, a program to play audio CD-ROMs will work on both IDE and SCSI CD-ROM drives because operating systems provide an abstract CD-ROM interface. VMware Workstation takes advantage of this generality to run on whole classes of hardware without itself needing special device drivers for each possible device.

The most significant trade-off of a hosted architecture is in potential I/O performance degradation. Because I/O emulation is done in the host world, a virtual machine executing an I/O intensive workload can accrue extra CPU time switching between the VMM and host worlds, as well as significant time in the host world performing I/O to the native hardware. This increases the CPU overhead associated with any I/O operation.

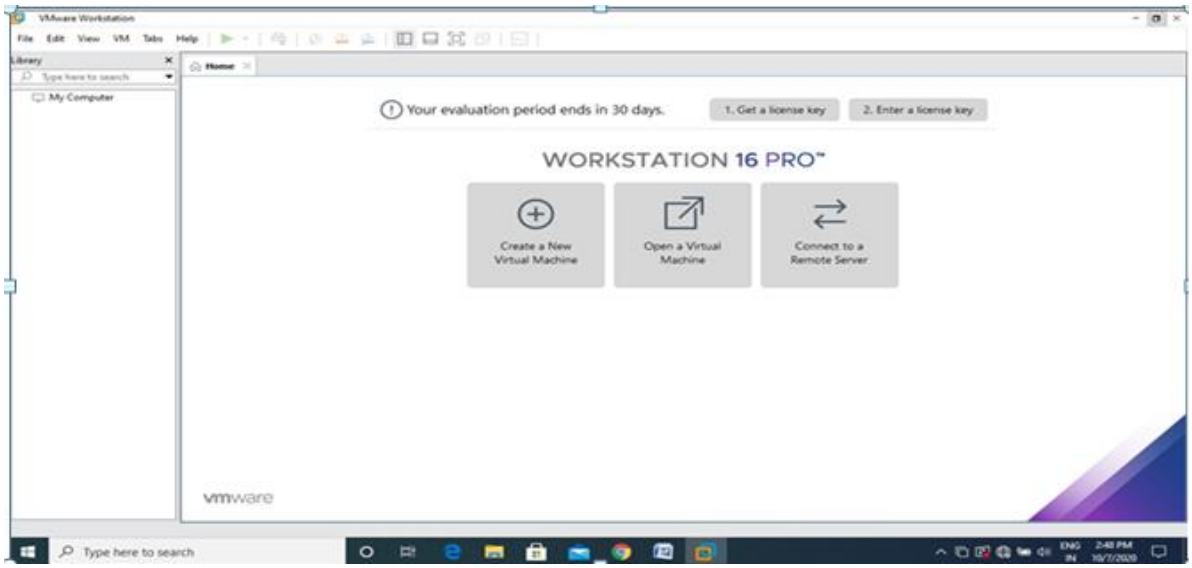
Another trade-off of the hosted architecture is that the host OS is in full control of machine resources. Even though the VMM has full system and hardware privileges, it behaves cooperatively and allows the host OS to schedule it. The host OS can also page out the memory allocated to a particular virtual machine except for a small set of pages that the VMM has pinned on behalf of the virtual machine. This allows VMware Workstation to be treated by the host OS like a regular application, but occasionally at the expense of performance if the host OS makes poor resource scheduling choices for the virtual machine.

Procedure and implementation:

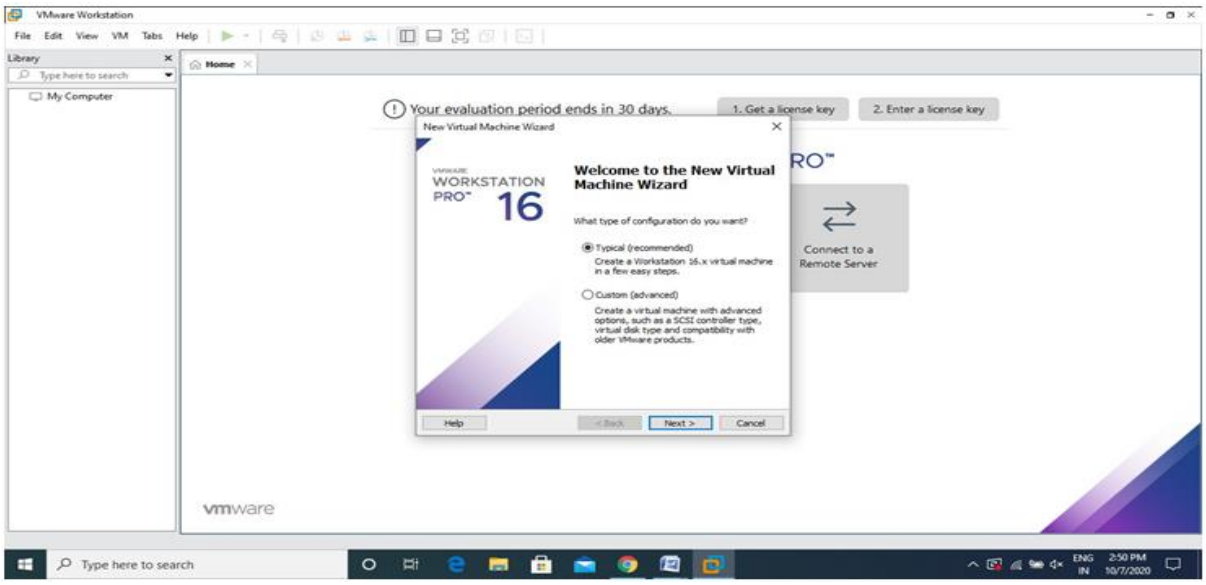
- Go to https://www.vmware.com/in/products/workstation-pro/workstation-pro-evaluation.html

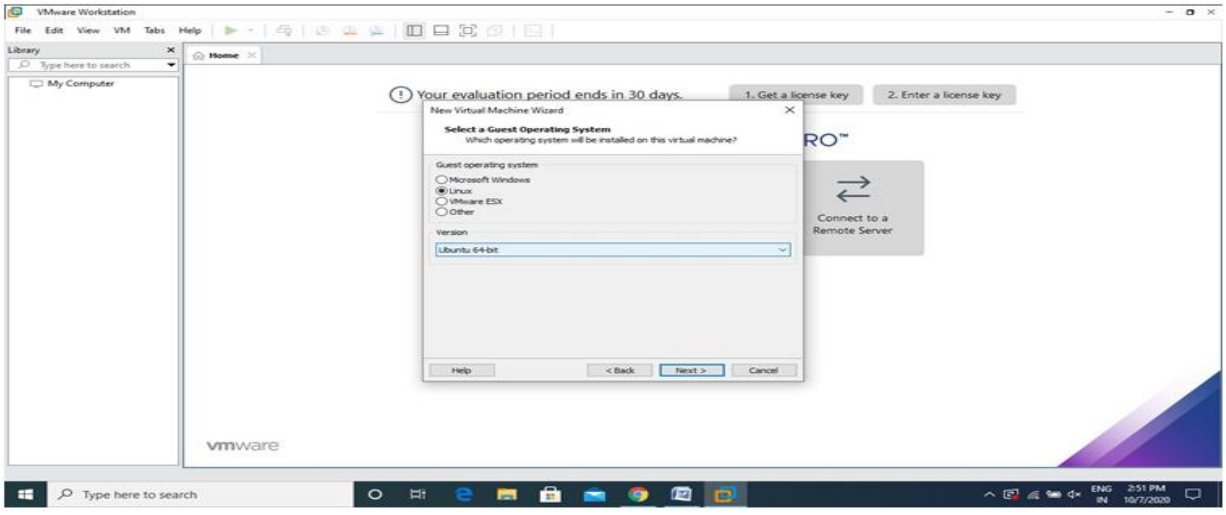


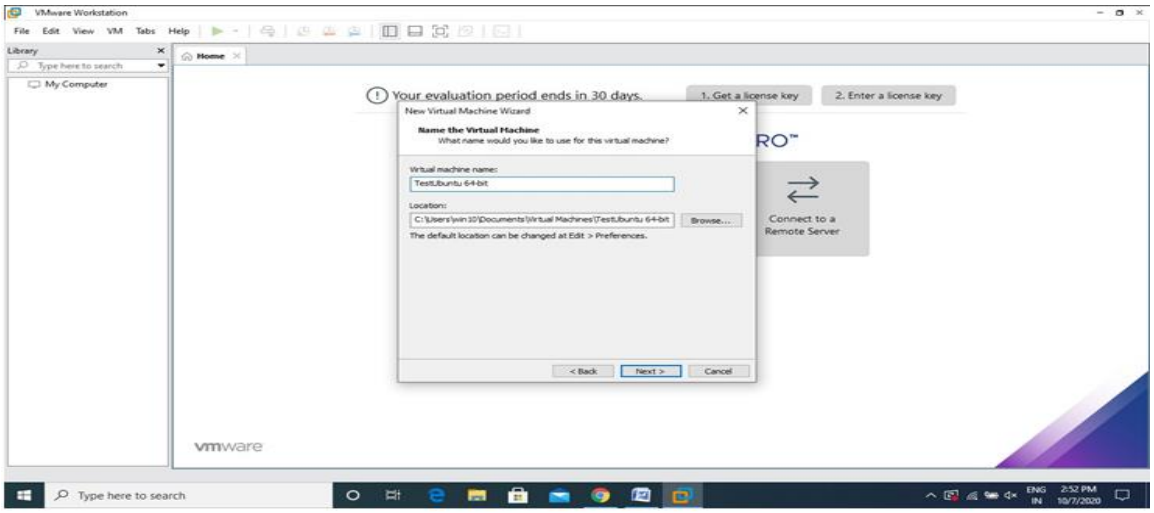
- Download workstation 16 Pro for Windows.
- Once downloaded install it.

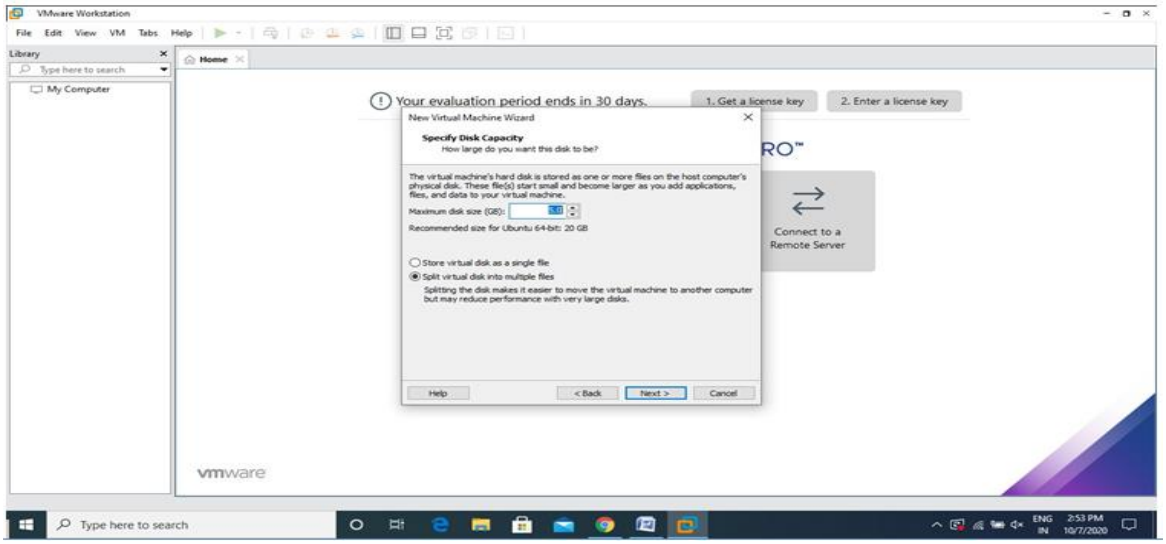


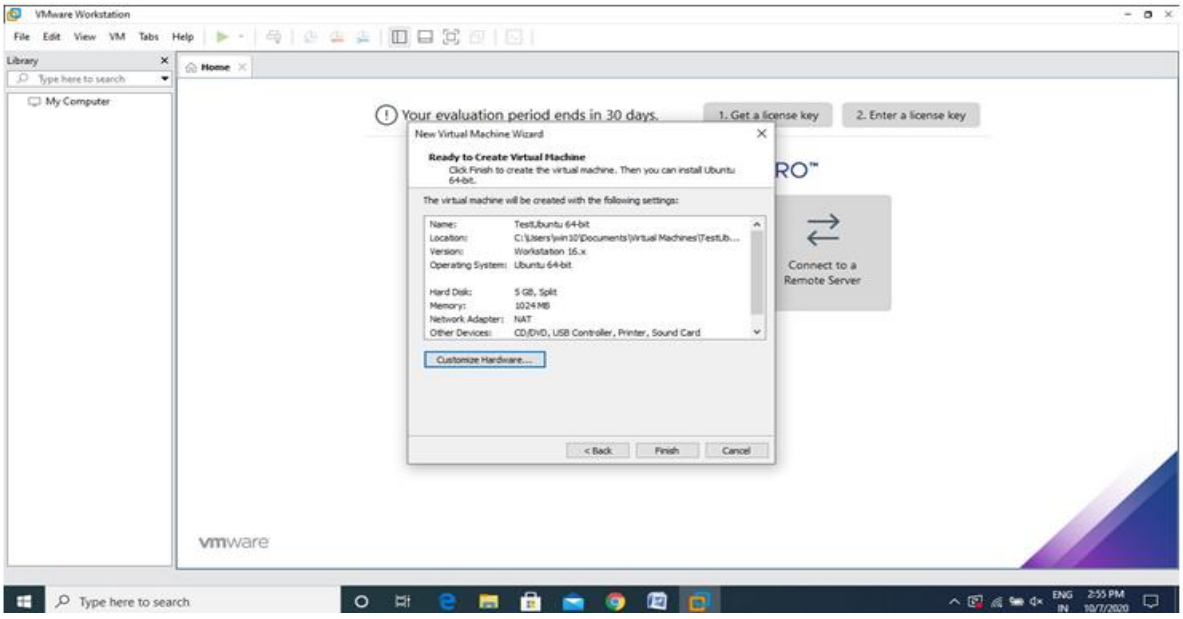
- Create a new Virtual Machine and use ubuntu iso file for creation of ubuntu VM.

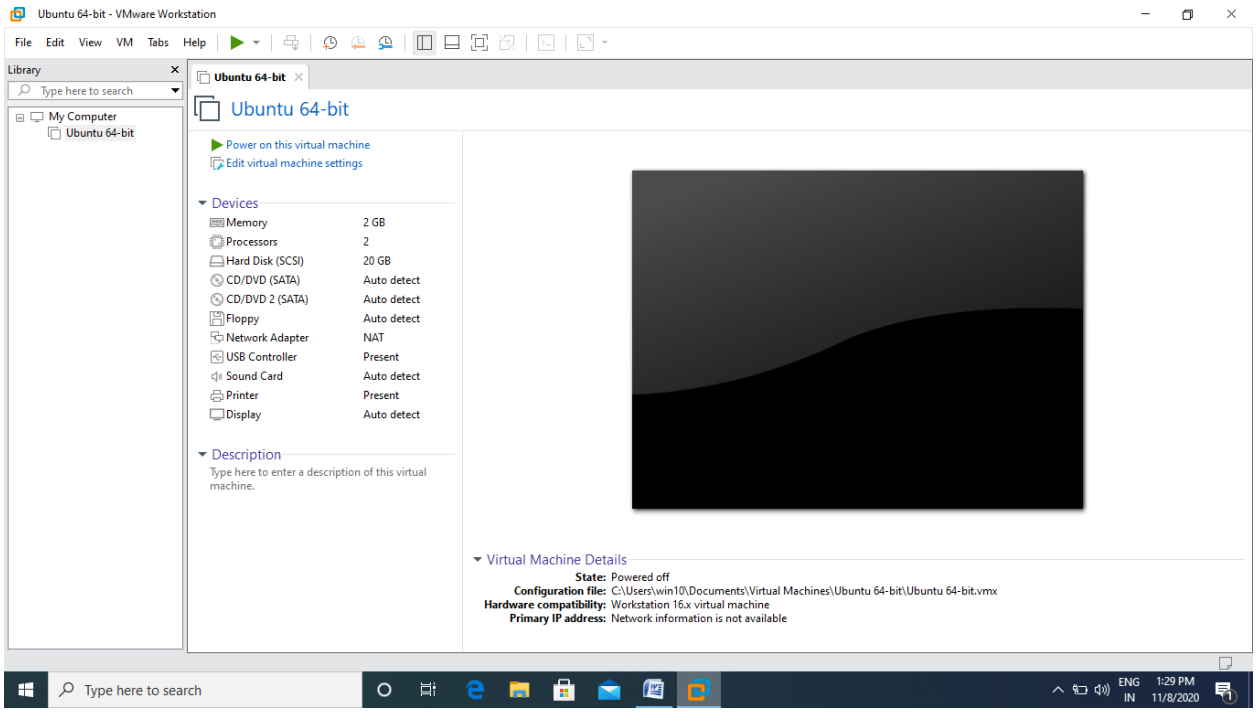

VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Home
Your evaluation period ends in 30 days.
1. Get a license key
2. Enter a license key
New Virtual Machine Wizard
WORKSTATION PRO™ 16
Welcome to the New Virtual Machine Wizard
What type of configuration do you want?
Typical (recommended)
Create a Workstation 16.x virtual machine in a few easy steps.
Custom (advanced)
Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.
Help
< Back
Next >
Cancel
Connect to a Remote Server
vmware
Type here to search
ENG IN 2:50 PM 10/7/2020

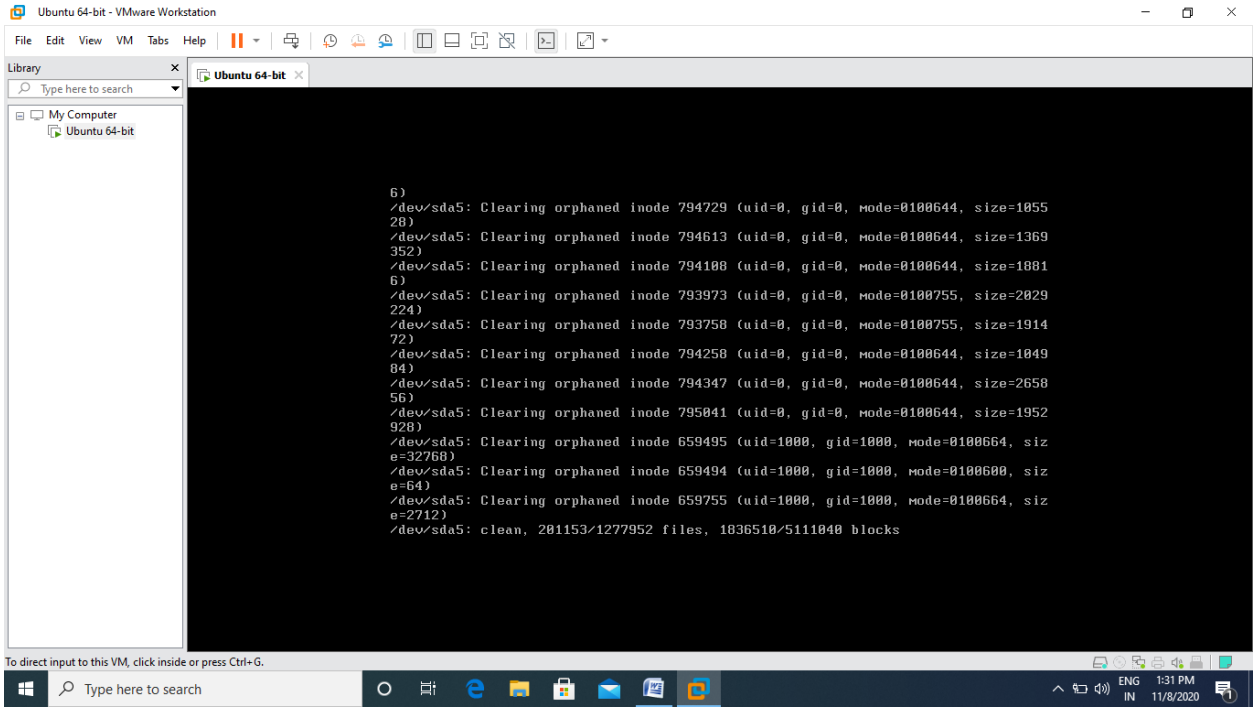

VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Home
Your evaluation period ends in 30 days.
1. Get a license key
2. Enter a license key
New Virtual Machine Wizard
Select a Guest Operating System
Which operating system will be installed on this virtual machine?
Guest operating system
Microsoft Windows
Linux
VMware ESX
Other
Version
Ubuntu 64-bit
Help
< Back
Next >
Cancel
Connect to a Remote Server
vmware
Type here to search
ENG IN 2:51 PM 10/7/2020

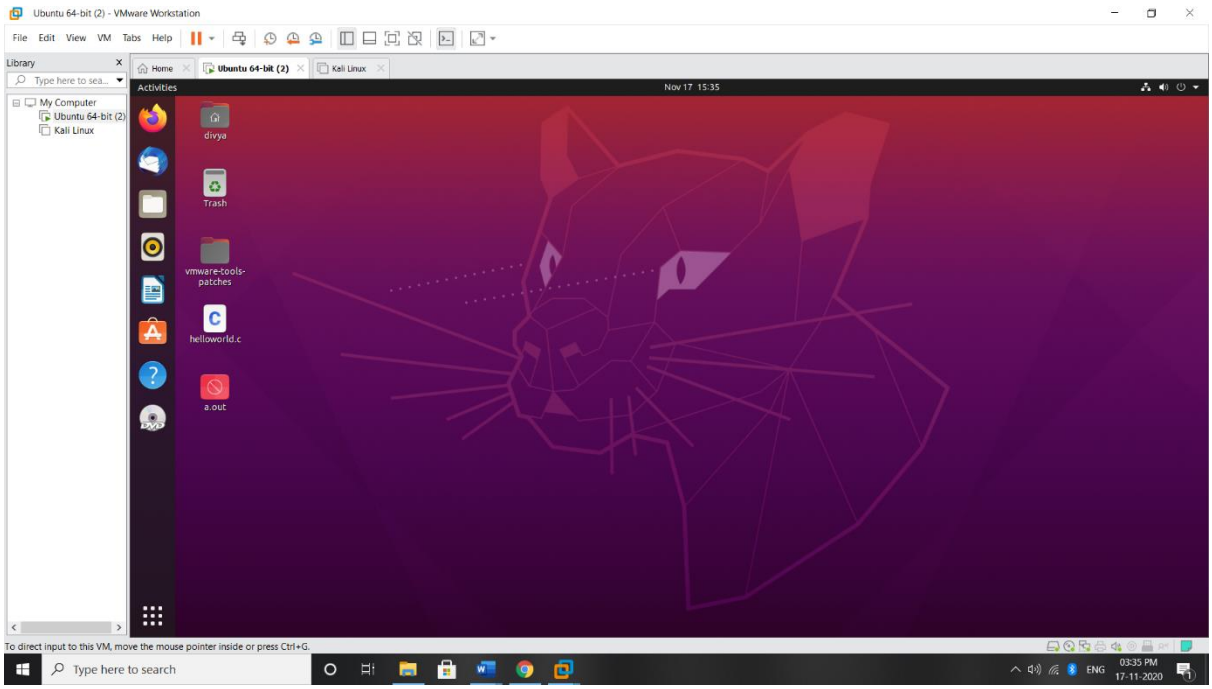

VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Home
Your evaluation period ends in 30 days.
1. Get a license key
2. Enter a license key
New Virtual Machine Wizard
Name the Virtual Machine
What name would you like to use for this virtual machine?
Virtual machine name:
TestUbuntu 64-bit
Location:
C:\Users\win10\Documents\Virtual Machines\TestUbuntu 64-bit
Browse...
The default location can be changed at Edit > Preferences.
< Back
Next >
Cancel
Connect to a Remote Server
vmware
Type here to search
ENG IN 2:52 PM 10/7/2020


VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Home
Your evaluation period ends in 30 days.
1. Get a license key
2. Enter a license key
New Virtual Machine Wizard
Specify Disk Capacity
How large do you want this disk to be?
The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.
Maximum disk size (GB):
Recommended size for Ubuntu 64-bit: 20 GB
Store virtual disk as a single file
Split virtual disk into multiple files
Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.
Help
< Back
Next >
Cancel
Connect to a Remote Server
vmware
Type here to search
ENG IN
2:53 PM
10/7/2020


VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Home
Your evaluation period ends in 30 days.
1. Get a license key
2. Enter a license key
New Virtual Machine Wizard
Ready to Create Virtual Machine
Click Finish to create the virtual machine. Then you can install Ubuntu 64-bit.
The virtual machine will be created with the following settings:
Name: TestUbuntu 64-bit
Location: C:\Users\win10\Documents\Virtual Machines\TestUb...
Version: Workstation 16.x
Operating System: Ubuntu 64-bit
Hard Disk: 5 GB, Split
Memory: 1024 MB
Network Adapter: NAT
Other Devices: CD/DVD, USB Controller, Printer, Sound Card
Customize Hardware...
< Back
Finish
Cancel
Connect to a Remote Server
vmware
Type here to search
ENG IN
2:55 PM
10/7/2020


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu 64-bit
Ubuntu 64-bit
Power on this virtual machine
Edit virtual machine settings
Devices
Memory 2 GB
Processors 2
Hard Disk (SCSI) 20 GB
CD/DVD (SATA) Auto detect
CD/DVD 2 (SATA) Auto detect
Floppy Auto detect
Network Adapter NAT
USB Controller Present
Sound Card Auto detect
Printer Present
Display Auto detect
Description
Type here to enter a description of this virtual machine.
Virtual Machine Details
State: Powered off
Configuration file: C:\Users\win10\Documents\Virtual Machines\Ubuntu 64-bit\Ubuntu 64-bit.vmx
Hardware compatibility: Workstation 16.x virtual machine
Primary IP address: Network information is not available
Type here to search
ENG IN 1:29 PM 11/8/2020


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu 64-bit
Ubuntu 64-bit
6)
/dev/sda5: Clearing orphaned inode 794729 (uid=0, gid=0, mode=0100644, size=1055
28)
/dev/sda5: Clearing orphaned inode 794613 (uid=0, gid=0, mode=0100644, size=1369
352)
/dev/sda5: Clearing orphaned inode 794108 (uid=0, gid=0, mode=0100644, size=1881
6)
/dev/sda5: Clearing orphaned inode 793973 (uid=0, gid=0, mode=0100755, size=2029
224)
/dev/sda5: Clearing orphaned inode 793758 (uid=0, gid=0, mode=0100755, size=1914
72)
/dev/sda5: Clearing orphaned inode 794258 (uid=0, gid=0, mode=0100644, size=1049
84)
/dev/sda5: Clearing orphaned inode 794347 (uid=0, gid=0, mode=0100644, size=2658
56)
/dev/sda5: Clearing orphaned inode 795041 (uid=0, gid=0, mode=0100644, size=1952
928)
/dev/sda5: Clearing orphaned inode 659495 (uid=1000, gid=1000, mode=0100664, siz
e=32768)
/dev/sda5: Clearing orphaned inode 659494 (uid=1000, gid=1000, mode=0100600, siz
e=64)
/dev/sda5: Clearing orphaned inode 659755 (uid=1000, gid=1000, mode=0100664, siz
e=2712)
/dev/sda5: clean, 201153/1277952 files, 1836510/5111040 blocks
To direct input to this VM, click inside or press Ctrl+G.
Type here to search
ENG IN 1:31 PM 11/8/2020



Result:

The VMware workstation 16 Pro has been successfully installed and Ubuntu VM has been successfully created.

Ex: 4 **Programming with VMs**

Date: 09.09.2020

Objective:

To understand the installation procedure of GCC and to program with VMs.

Requirement:

- VMware workstation 16 Pro,
- Ubuntu OS.

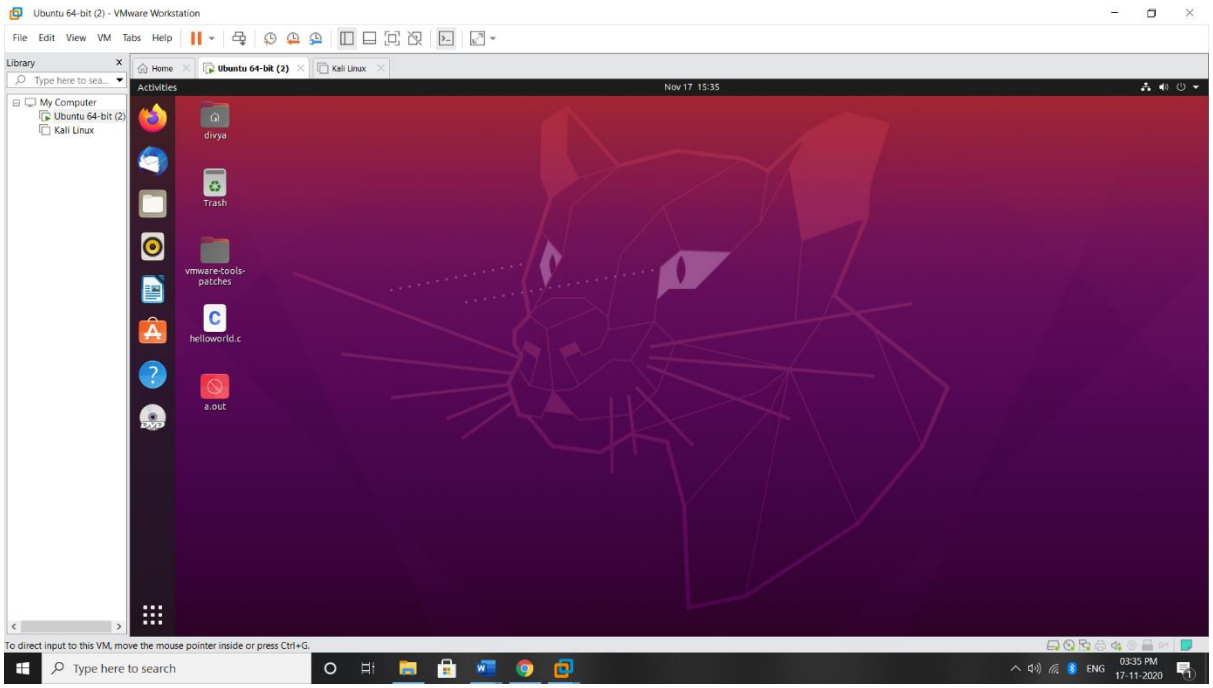
Theory:

Virtual Machine:

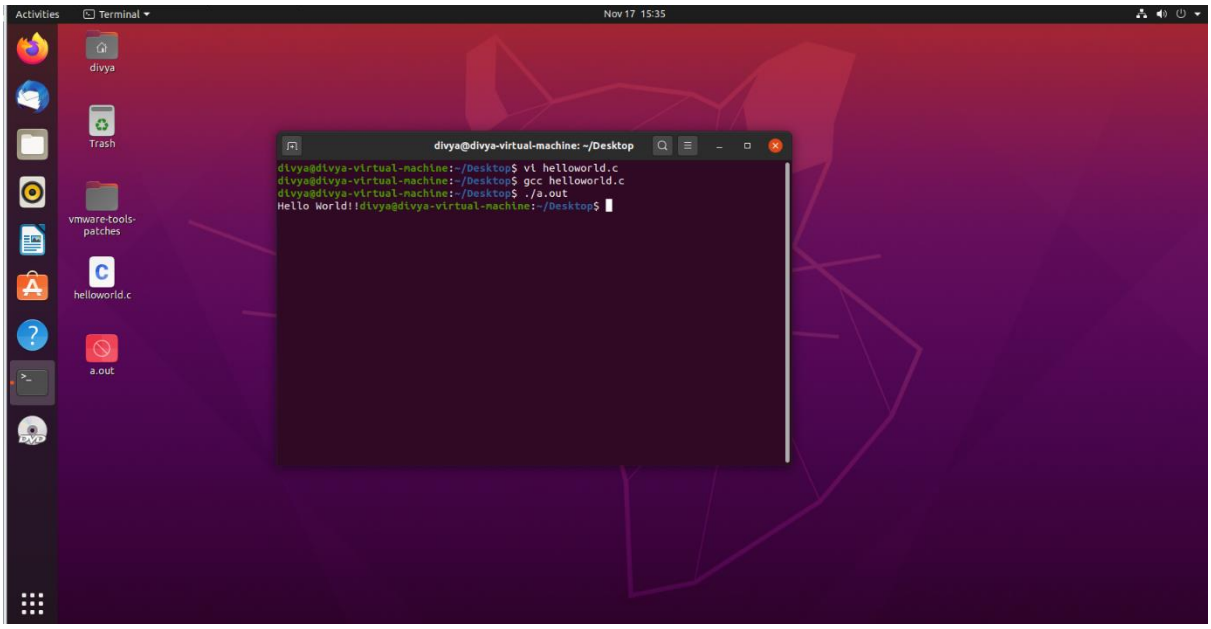
With the introduction of new technologies and newer research models, a lot number of hardware and software products are being launched. Many of the software are platform-dependent hence it is sometimes difficult to debug or check them because of the limited hardware resources.A VM (virtual machine) is an emulation of a computer system, where these machines use computer architectures to provide the functionality of a physical computer. The physical device on which virtual machines work is known as Host, whereas the virtual machines are known as Guest. A single host can have multiple numbers of guests.

Procedure and implementation:

- Open Ubuntu VM in VMware workstation Pro.



- Install gcc using the following command - *sudo apt install gcc*
- Create helloworld.c file.
- Compile and Run the helloworld.c file.



Result:

GCC has been successfully installed and the above program has been successfully executed in Ubuntu Virtual Machine in VMware workstation 16 Pro.

Ex: 5 **Google App Engine installation**

Date: 16.09.2020

Objective:

To understand the working of Google App Engine and to learn the installation procedure of Google App Engine.

Requirement:

- Google Chrome (or) related internet explorer
- JRE 11 (or) higher

Theory:

Architecture:

The Google App Engine (GAE) is Google`s answer to the ongoing trend of Cloud Computing offerings within the industry. In the traditional sense, GAE is a web application hosting service, allowing for development and deployment of web-based applications within a predefined runtime environment. Unlike other cloud-based hosting offerings such as Amazon Web Services that operate on an IaaS level, the GAE already provides an application infrastructure on the PaaS level. This means that the GAE abstracts from the underlying hardware and operating system layers by providing the hosted application with a set of application-oriented services. While this approach is very convenient for developers of such applications, the rationale behind the GAE is its focus on scalability and usage-based infrastructure as well as payment.

Costs:

Developing and deploying applications for the GAE is generally free of charge but restricted to a certain amount of traffic generated by the deployed application. Once this limit is reached within a certain time period, the application stops working. However, this limit can be waived when switching to a billable quota where the developer can enter a maximum budget that can be spent on an application per day. Depending on the traffic, once the free quota is reached the application will continue to work until the maximum budget for this day is reached. Table 1 summarizes some of the in our opinion

most important quotas and corresponding amount per unit that is charged when free resources are depleted and additional, billable quota is desired.

Features:

With a Runtime Environment, the Data store and the App Engine services, the GAE can be divided into three parts. Runtime Environment The GAE runtime environment presents itself as the place where the actual application is executed. However, the application is only invoked once an HTTP request is processed to the GAE via a web browser or some other interface, meaning that the application is not constantly running if no invocation or processing has been done. In case of such an HTTP request, the request handler forwards the request and the GAE selects one out of many possible Google servers where the application is then instantly deployed and executed for a certain amount of time (8). The application may then do some computing and return the result back to the GAE request handler which forwards an HTTP response to the client. It is important to understand that the application runs completely embedded in this described sandbox environment but only as long as requests are still coming in or some processing is done within the application. The reason for this is simple: Applications should only run when they are actually computing, otherwise they would allocate precious computing power and memory without need. This paradigm shows already the GAE‘s potential in terms of scalability. Being able to run multiple instances of one application independently on different servers guarantees for a decent level of scalability. However, this highly flexible and stateless application execution paradigm has its limitations. Requests are processed no longer than 30 seconds after which the response has to be returned to the client and the application is removed from the runtime environment again (8). Obviously this method accepts that for deploying and starting an application each time a request is processed, an additional lead time is needed until the application is finally up and running. The GAE tries to encounter this problem by caching the application in the server memory as long as possible, optimizing for several subsequent requests to the same application. The type of runtime environment on the Google servers is dependent on the programming language used. For Java or other languages that have support for Java-based compilers (such as JRuby, Rhino and Groovy) a Java-based Java Virtual Machine (JVM) is provided. Also, GAE fully supports the Google Web

Toolkit (GWT), a framework for rich web applications. For Python and related frameworks a Python-based environment is used.

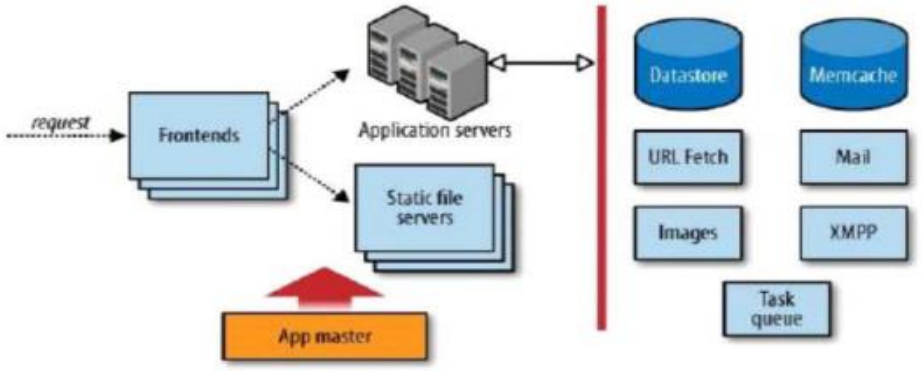
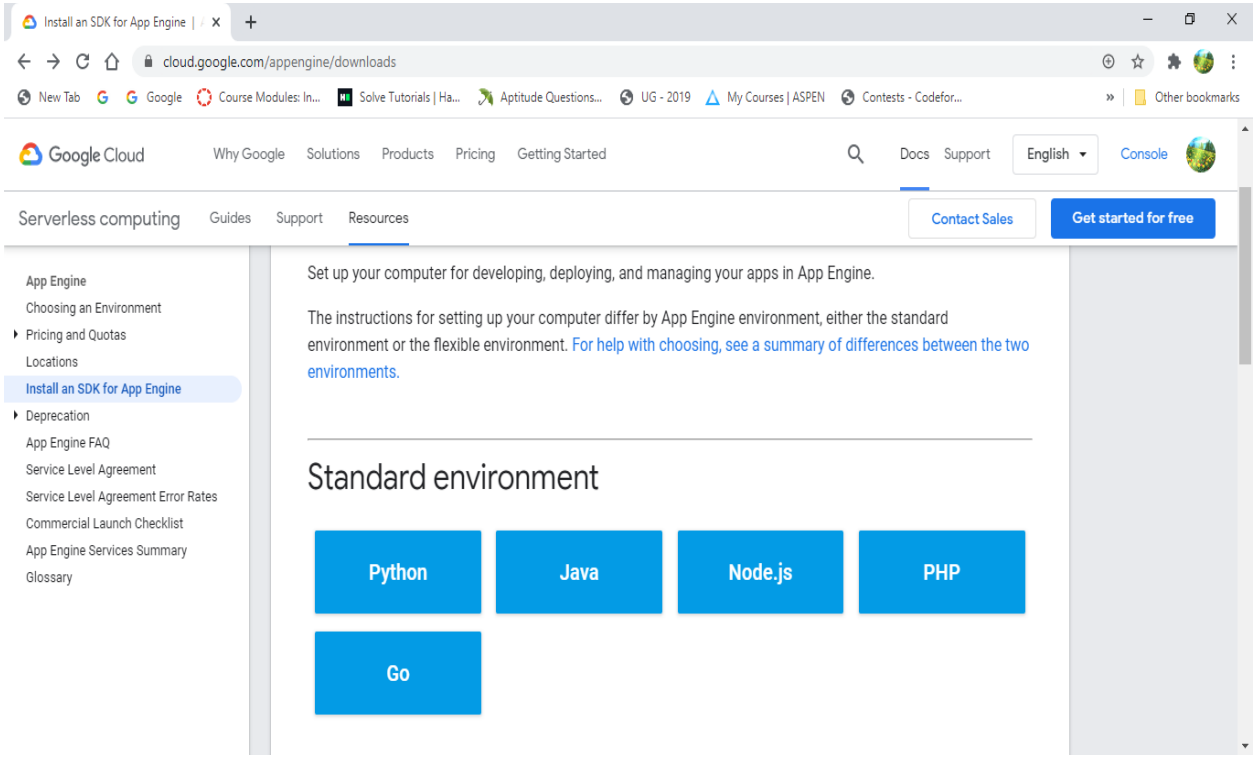


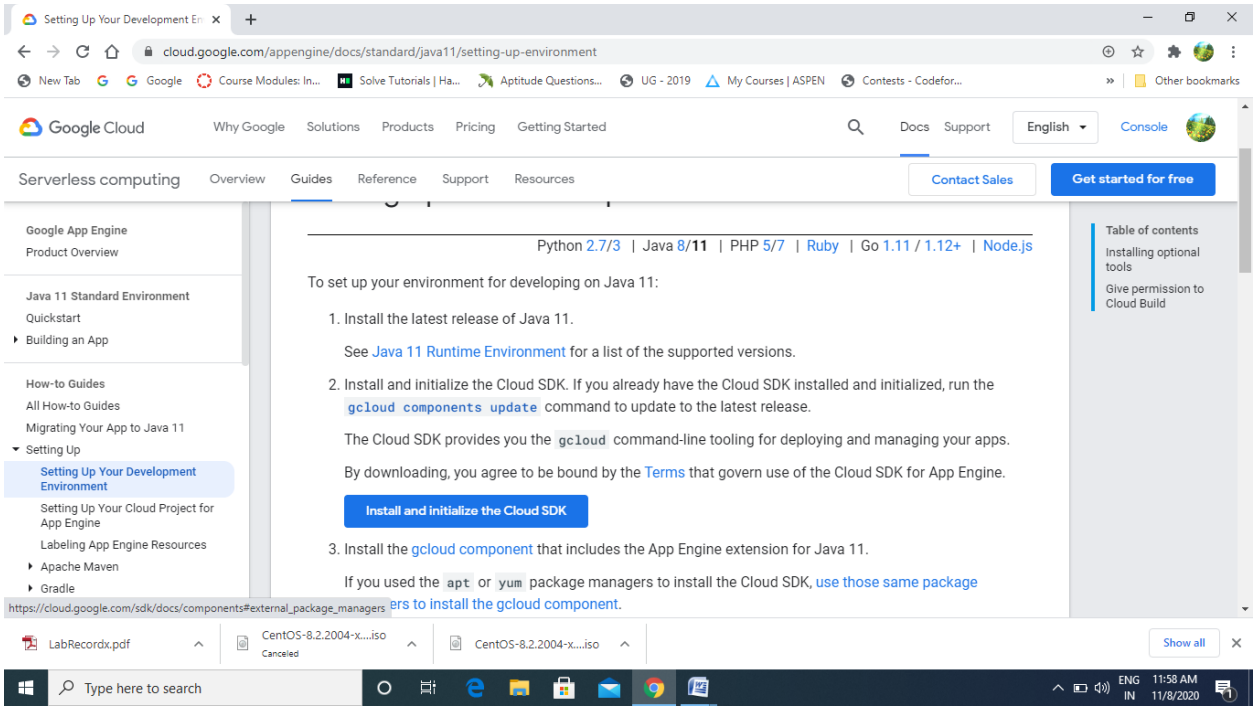
FIGURE 4: STRUCTURE OF GOOGLE APP ENGINE (13)

Procedure and implementation:

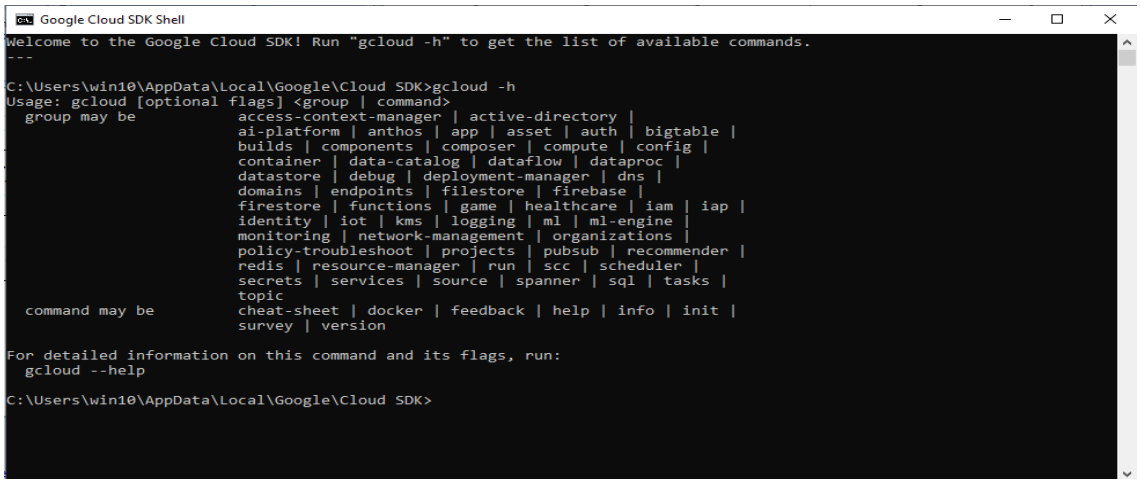
- Go to https://cloud.google.com/appengine/downloads and select java as standard environment.



- Install and Initialize the cloud SDK.



- The google cloud SDK shell will be opened as soon as the installation is over.
- Use *gcloud –h* to view the list of commands.



Result:

The Google App Engine has been successfully installed.

Ex: 6 **Build and Deploy Applications in GAE**

Date: 23.09.2020

Objective:

To understand and work with building and deployment of Applications in Google App Engine.

Requirement:

Google App Engine

Theory:

APPLICATION DEVELOPMENT USING GOOGLE APP ENGINE:

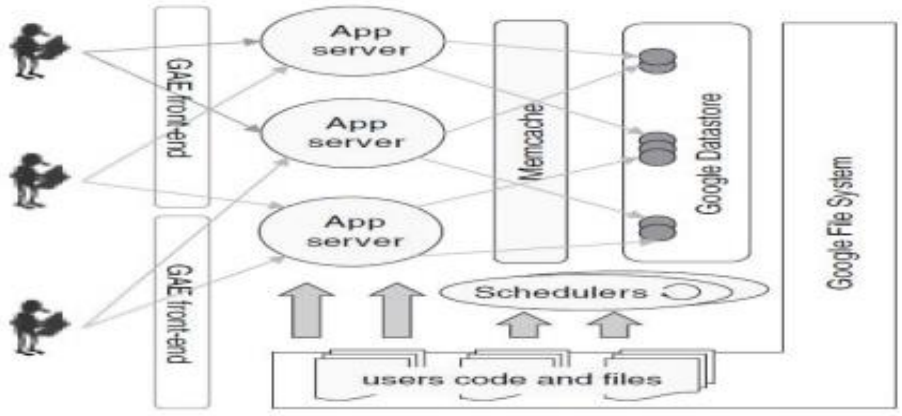
General Idea In order to evaluate the flexibility and scalability of the GAE we tried to come up with an application that relies heavily on scalability, i.e. collects large amounts of data from external sources. That way we hoped to be able to test both persistency and the gathering of data from external sources at large scale. Therefore our idea has been to develop an application that connects people`s delicious bookmarks with their respective Facebook accounts. People using our application should be able to see what their Facebook friends‘ delicious bookmarks are, provided their Facebook friends have such a delicious account. This way a user can get a visualization of his friends‘ latest topics by looking at a generated tag cloud giving him a clue about the most common and shared interests.

PLATFORM AS A SERVICE:

GOOGLE APP ENGINE:

The Google cloud, called Google App Engine, is a ‗platform as a service‘ (PaaS) offering. In contrast with the Amazon infrastructure as a service cloud, where users explicitly provision virtual machines and control them fully, including installing, compiling and running software on them, a PaaS offering hides the actual execution environment from users. Instead, a software platform is provided along with an SDK, using which users develop applications and deploy them on the cloud. The PaaS platform is responsible for executing the applications, including servicing external service requests, as well as running scheduled jobs included in the application. By making the actual execution

servers transparent to the user, a PaaS platform is able to share application servers across users who need lower capacities, as well as automatically scale resources allocated to applications that experience heavy loads. Below Figure depicts a user view of Google App Engine. Users upload code, in either Java or Python, along with related files, which are stored on the Google File System, a very large scale fault tolerant and redundant storage system. It is important to note that an application is immediately available on the internet as soon as it is successfully uploaded (no virtual servers need to be explicitly provisioned as in IaaS).

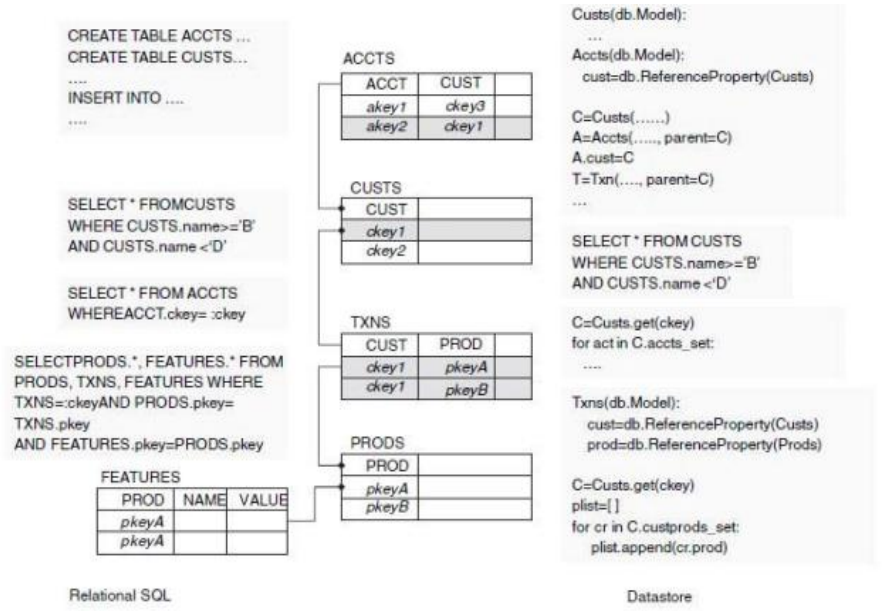


Resource usage for an application is metered in terms of web requests served and CPUhours actually spent executing requests or batch jobs. Note that this is very different from the IaaS model: A PaaS application can be deployed and made globally available 24×7, but charged only when accessed (or if batch jobs run); in contrast, in an IaaS model merely making an application continuously available incurs the full cost of keeping at least some of the servers running all the time. Further, deploying applications in Google App Engine is free, within usage limits; thus applications can be developed and tried out free and begin to incur cost only when actually accessed by a sufficient volume of requests. The PaaS model enables Google to provide such a free service because applications do not run in dedicated virtual machines; a deployed application that is not accessed merely consumes storage for its code and data and expends no CPU cycles. GAE applications are served by a large number of web servers in Google‘s data centers that execute requests from end-users across the globe. The web servers load code from the GFS into memory and serve these requests. Each request to a particular application is served by any one of GAE‘s web servers; there is no guarantee that the same server will serve requests to any two

requests, even from the same HTTP session. Applications can also specify some functions to be executed as batch jobs which are run by a scheduler.

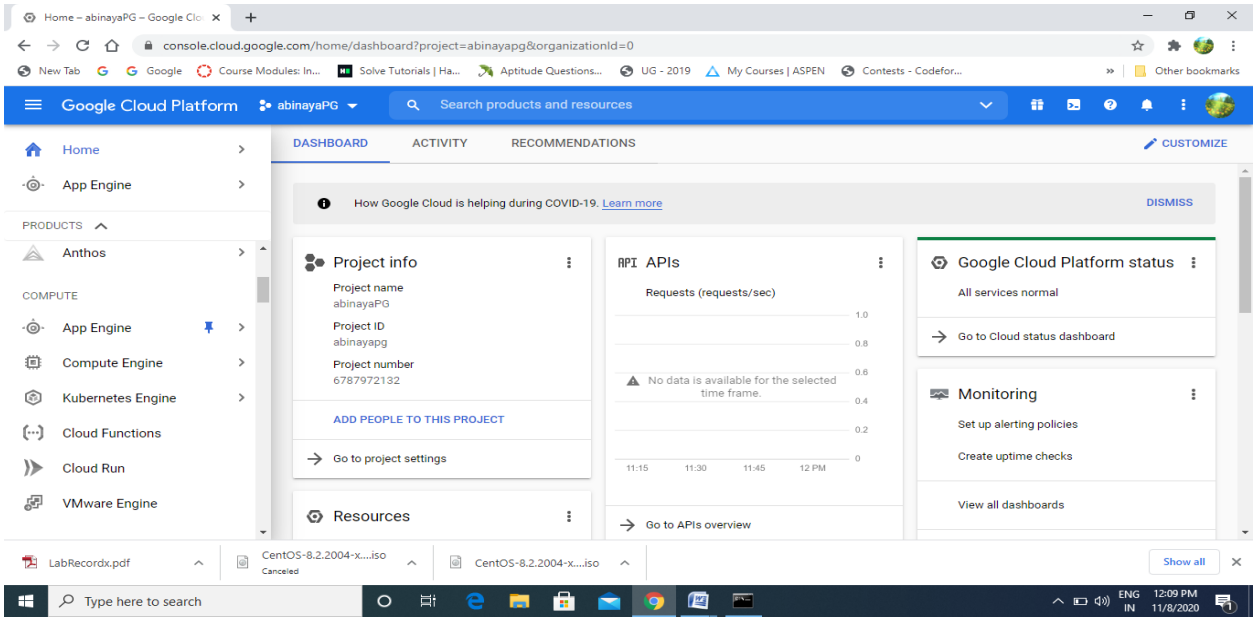
Google Datastore:

Applications persist data in the Google Datastore, which is also (like Amazon SimpleDB) a nonrelational database. The Datastore allows applications to define structured types (called _kinds‘) and store their instances (called _entities‘) in a distributed manner on the GFS file system. While one can view Datastore _kinds‘ as table structures and entities as records, there are important differences between a relational model and the Datastore, some of which are also illustrated in Figure.

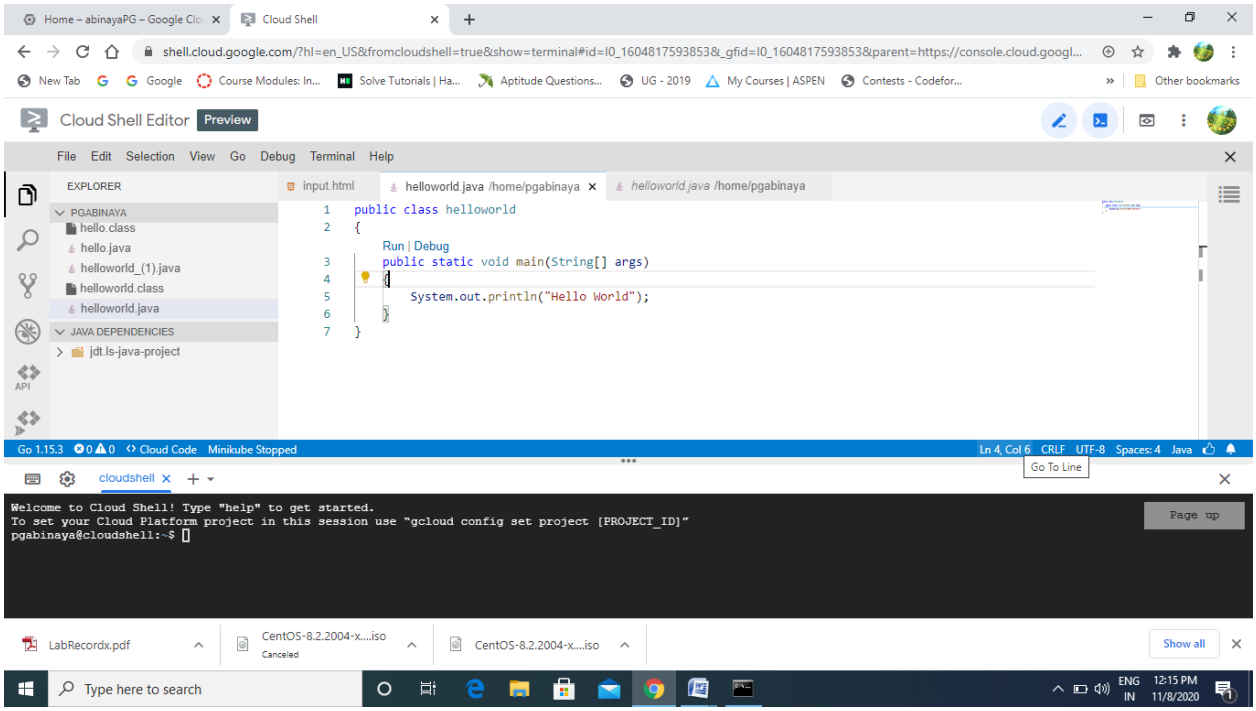


Procedure and implementation:

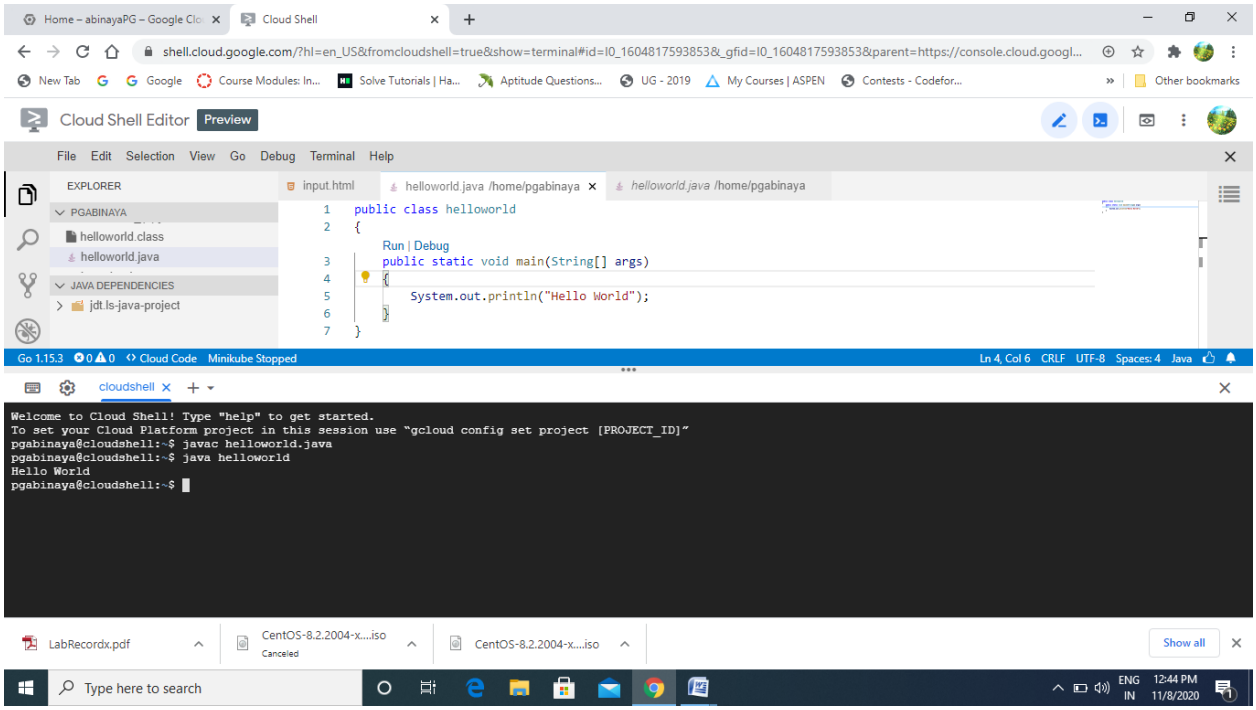
- Go to https://cloud.google.com/appengine/docs/standard/java11/setting-up-environment and go to console.



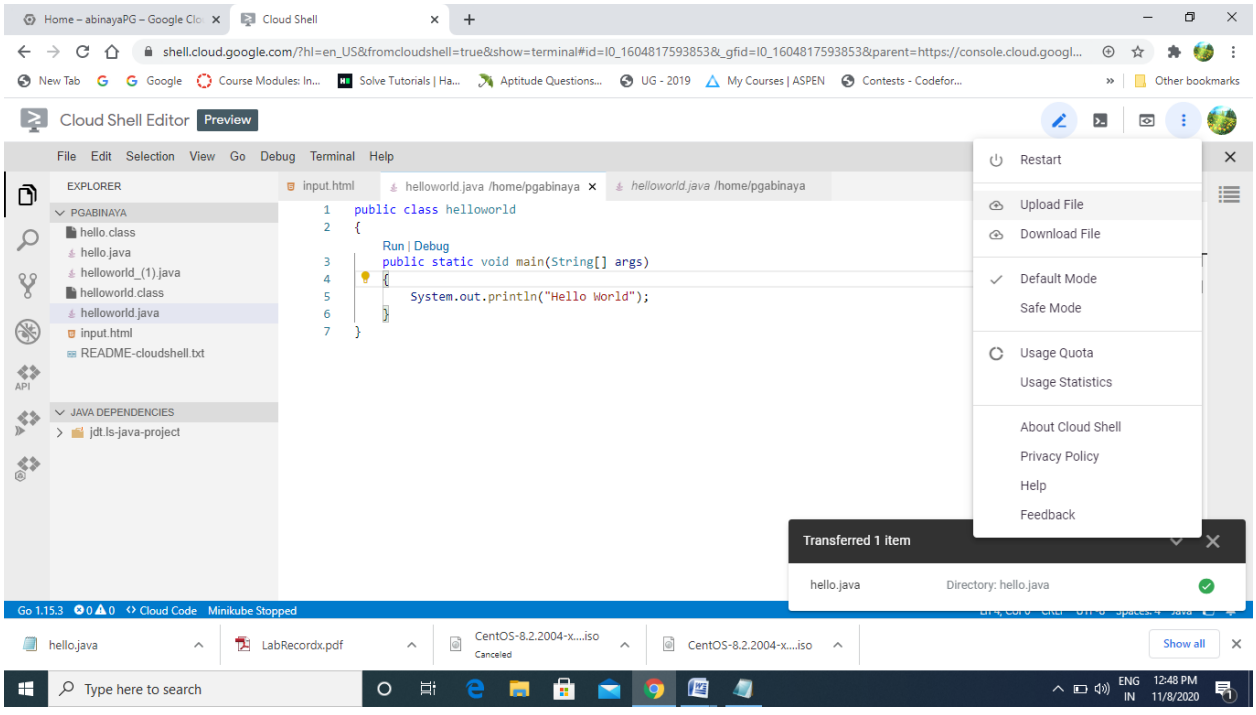
❖ Activate cloud shell and type the application in the editor.

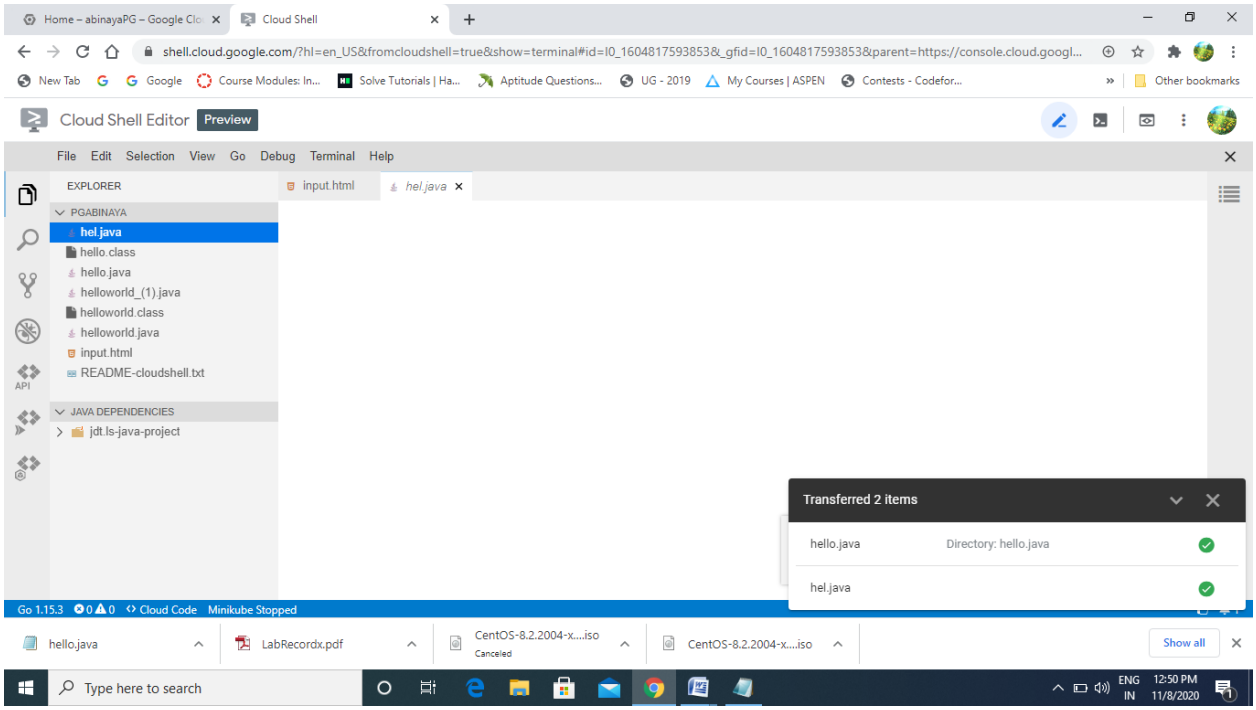


❖ Run/Deploy the application.

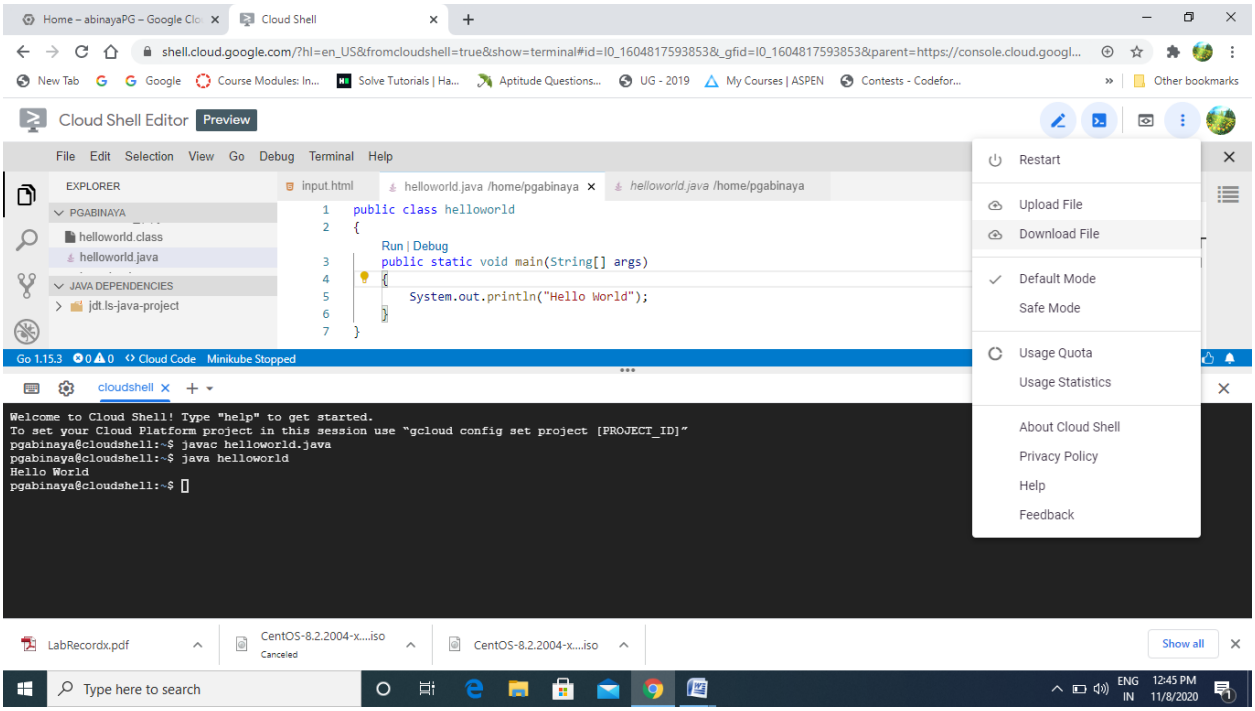


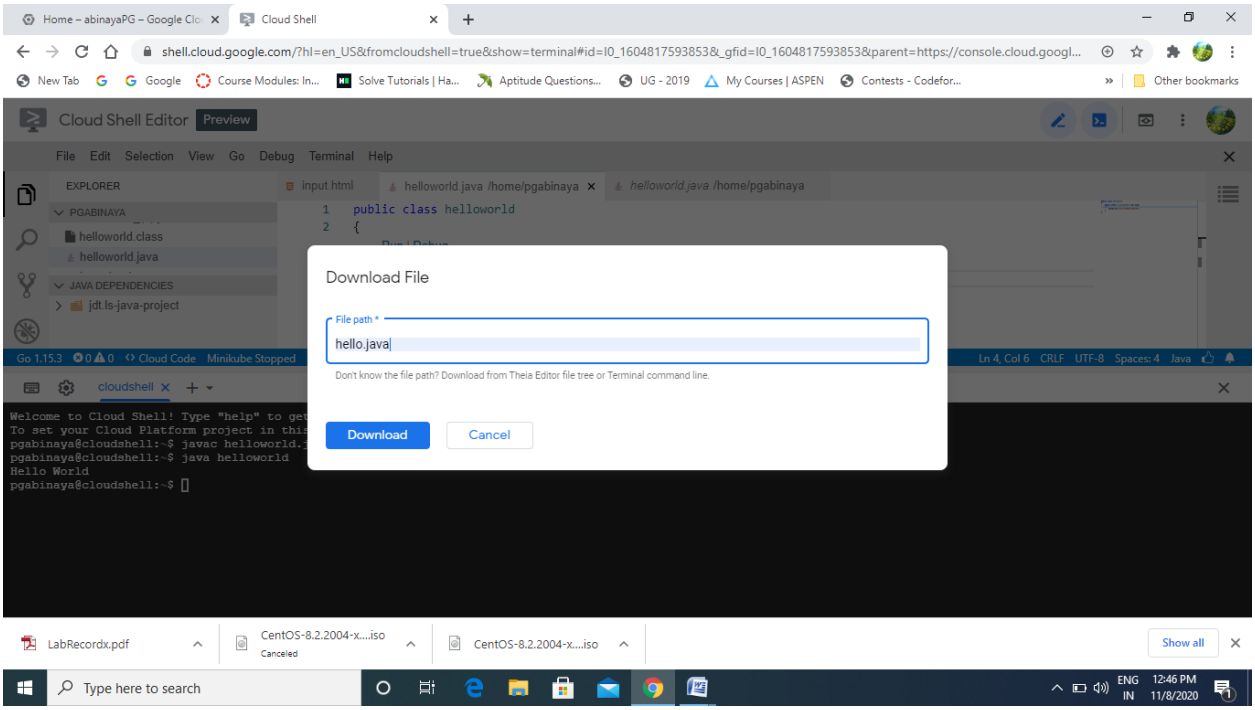
❖ Upload any file from Desktop to GAE.

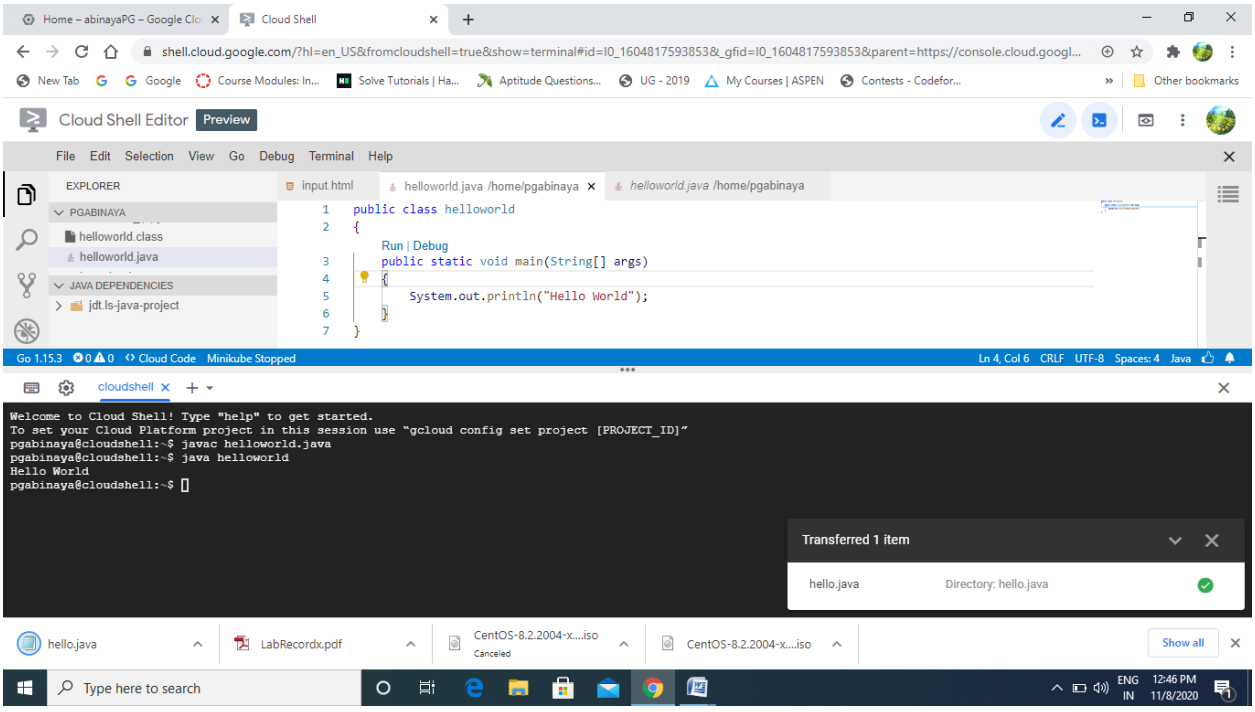




❖ Download the file from GAE to Desktop.







Result:

The above application has been successfully build and deployed in Google App Engine.

Ex: 7 Build and run the scheduling in CloudSim

Date: 30.09.2020

Objective:

To understand the working of cloudsim and to implement scheduling algorithm using cloudsim in eclipse.

Requirement:

- Eclipse (latest version)
- Cloudsim folder
- Common math folder

Theory:

CloudSim is a simulation toolkit that supports the modeling and simulation of the core functionality of cloud, like job/task queue, processing of events, creation of cloud entities(datacenter, datacenter brokers, etc), communication between different entities, implementation of broker policies, etc.

This toolkit allows to:

- Test application services in a repeatable and controllable environment.
- Tune the system bottlenecks before deploying apps in an actual cloud.
- Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques

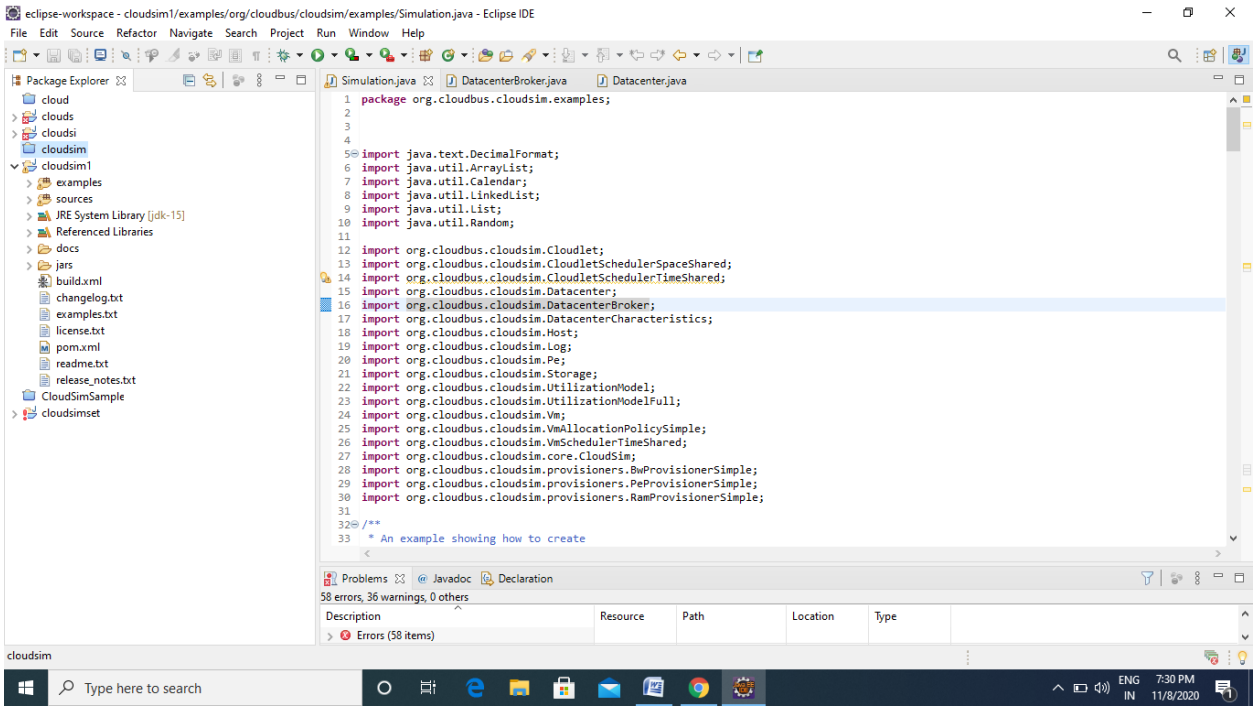
Core features :

- The Support of modeling and simulation of large scale computing environment as federated cloud data centers, virtualized server hosts, with customizable policies for provisioning host resources to virtual machines and energy-aware computational resources
- It is a self-contained platform for modeling cloud's service brokers, provisioning, and allocation policies.
- It supports the simulation of network connections among simulated system elements.
- Support for simulation of federated cloud environment, that inter-networks resources from both private and public domains.

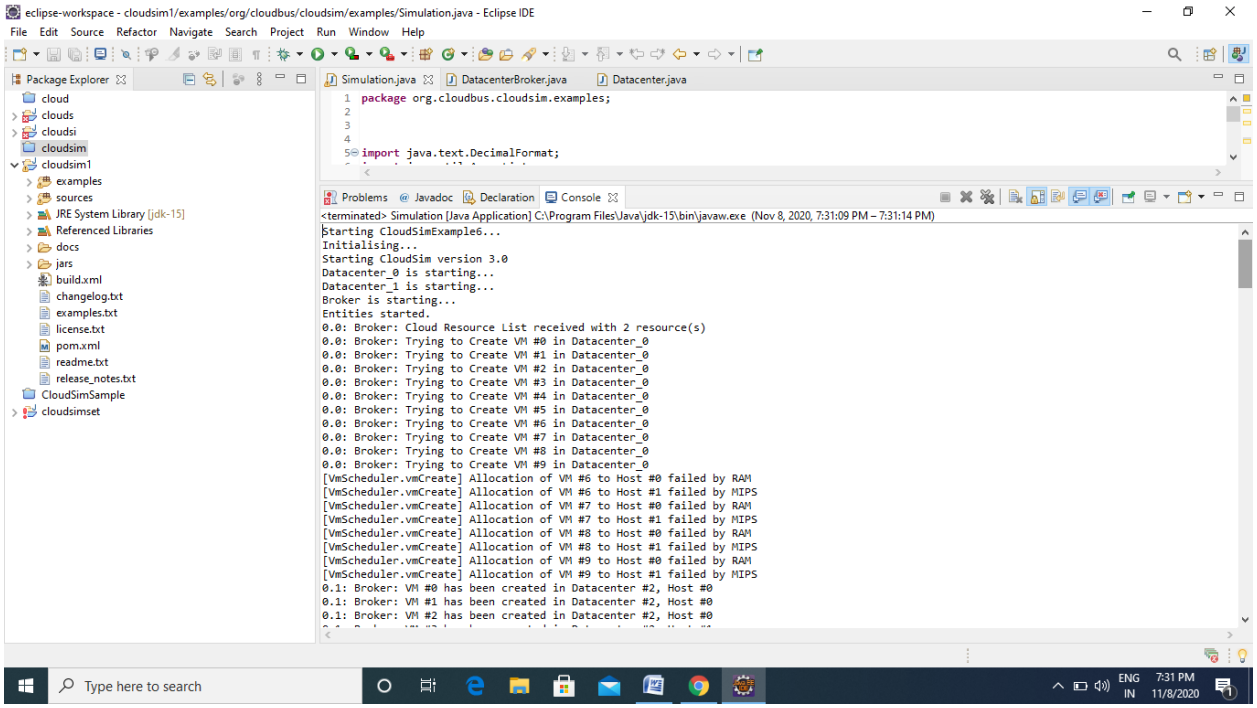
- Availability of a virtualization engine that aids in the creation and management of multiple independent and co-hosted virtual services on a data center node.
- Flexibility to switch between space shared and time shared allocation of processing cores to virtualized services.

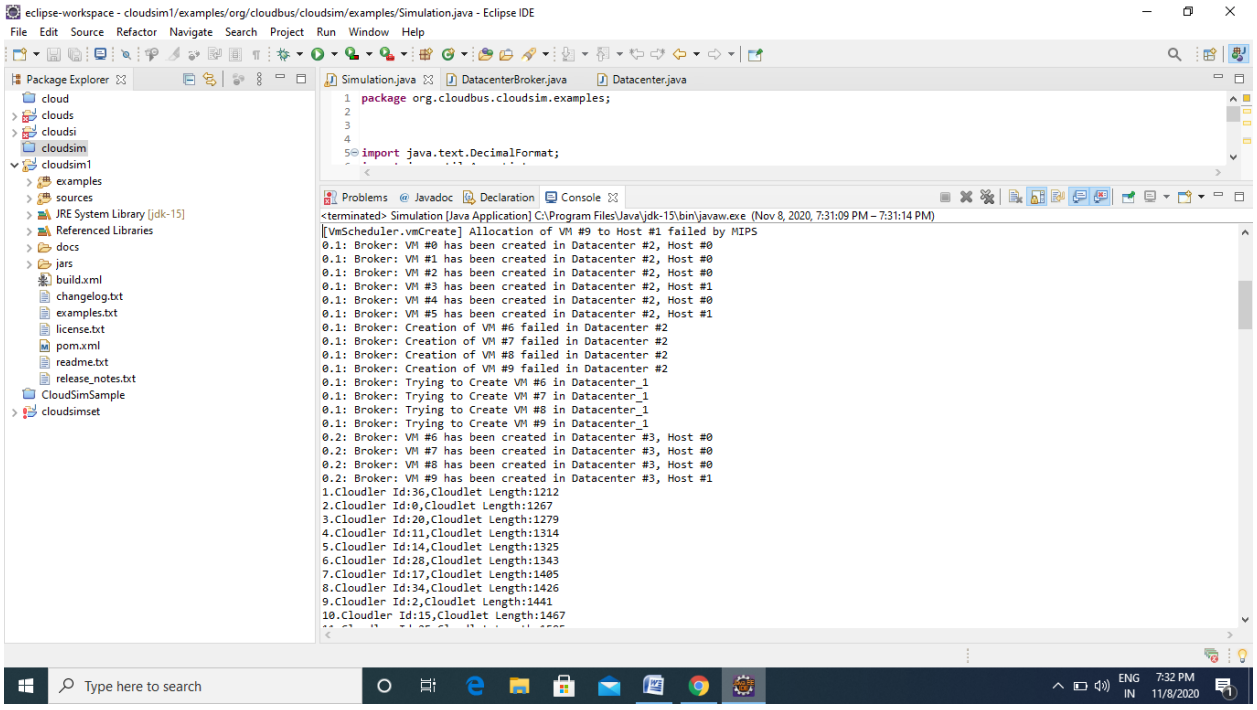
Procedure and implementation:

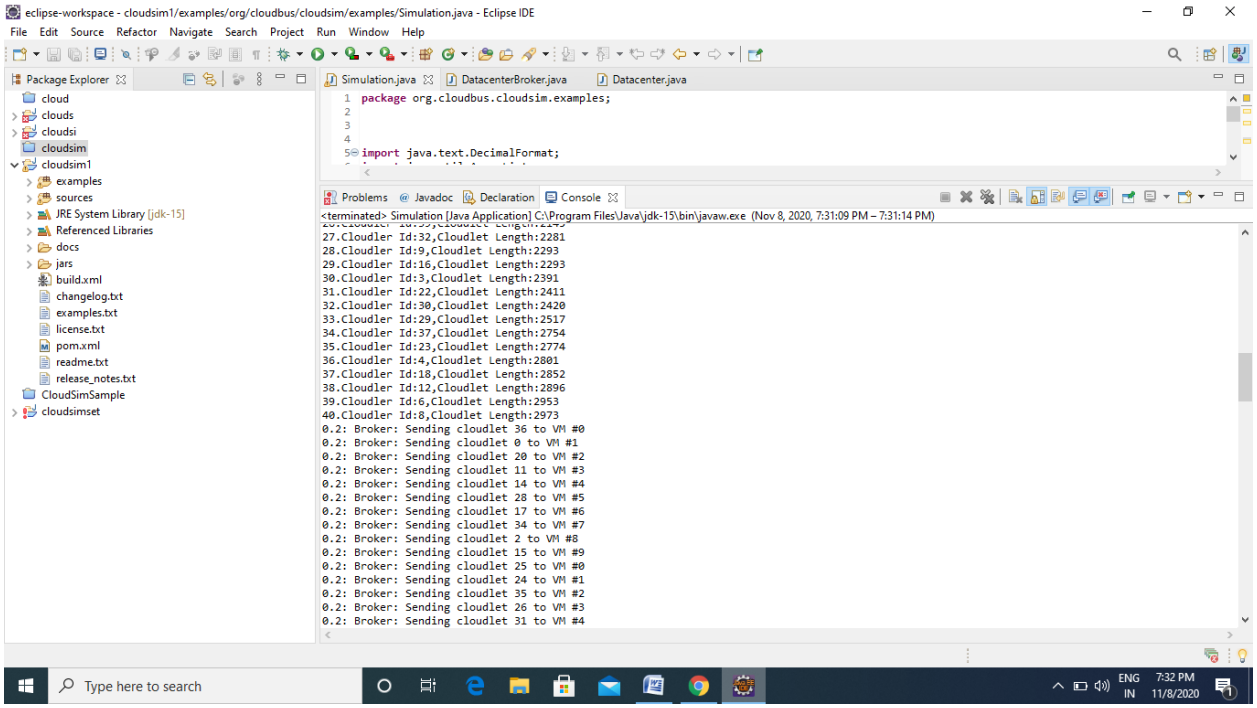
- Create a folder and paste the cloudsim and common math folder in that.
- Open eclipse.
- File->New->java project.
- Browse cloudsim folder and click finish
- Create Simulation.java and update DataCenterBroker.java

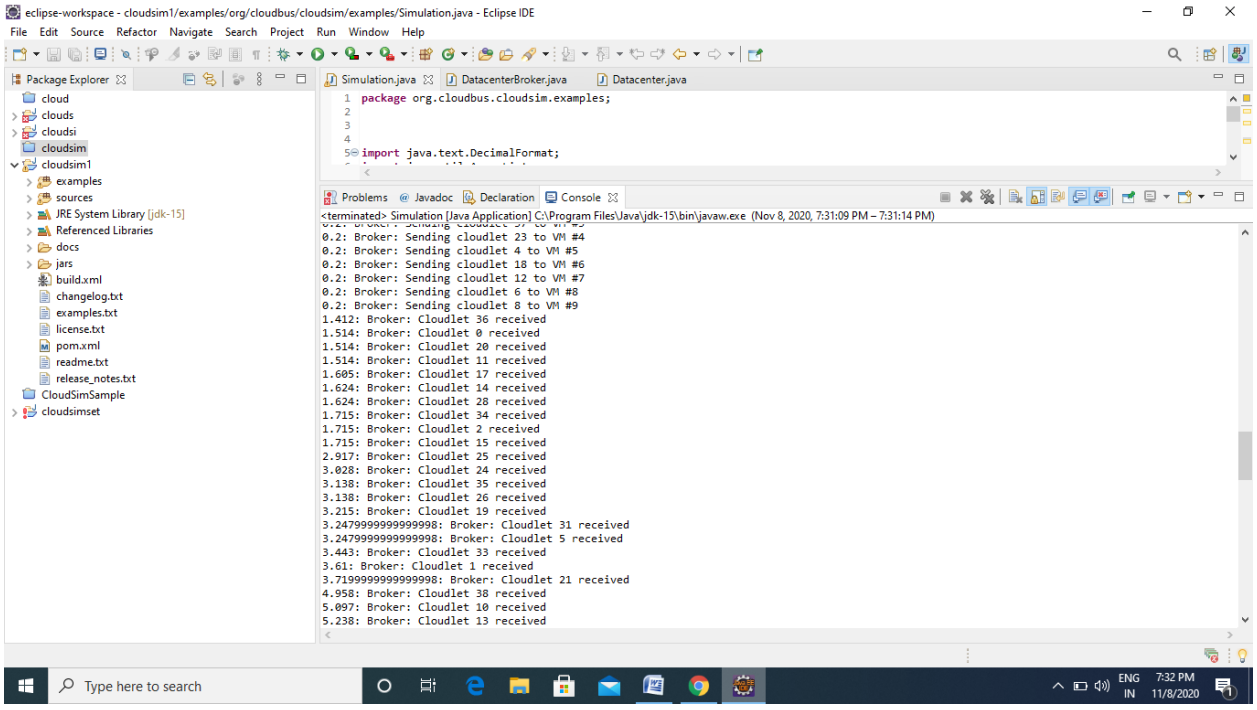


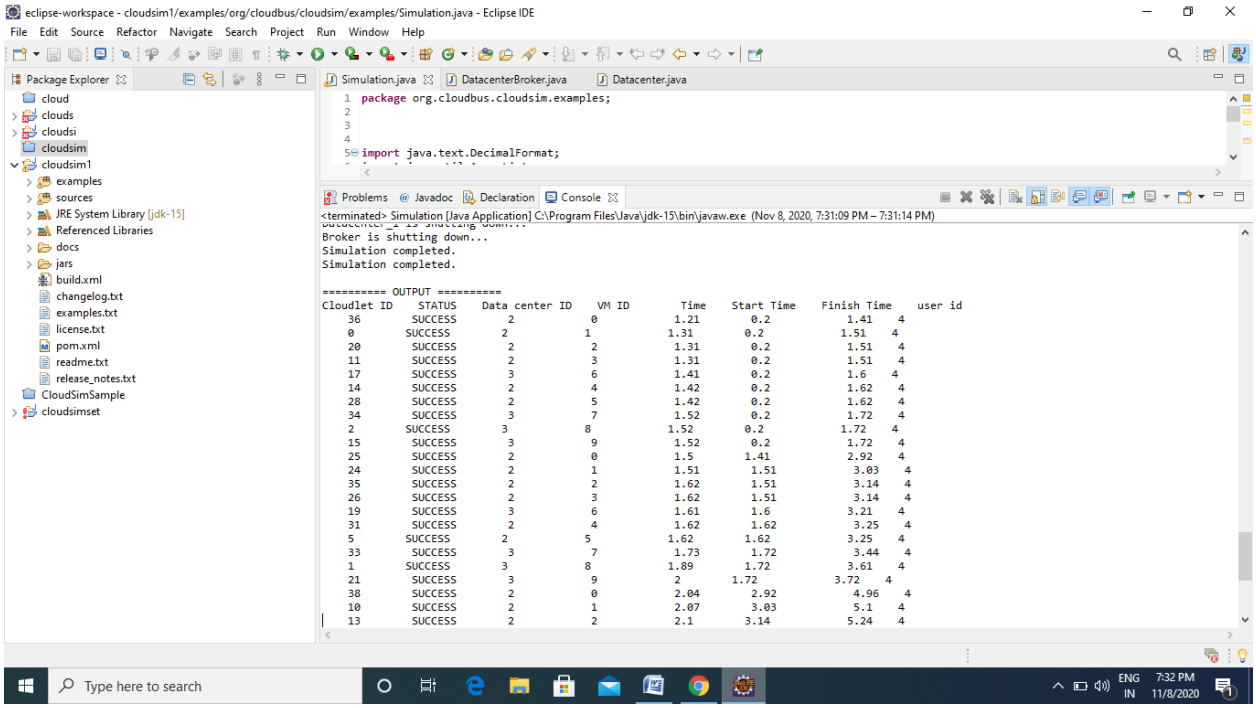
- Run the program


eclipse-workspace - cloudsim1/examples/org/cloudbus/cloudsim/examples/Simulation.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud
clouds
cloudsi
cloudsim
cloudsim1
examples
sources
JRE System Library [jdk-15]
Referenced Libraries
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
CloudSimSample
cloudsimset
Simulation.java
DatacenterBroker.java
Datacenter.java
1 package org.cloudbus.cloudsim.examples;
5 import java.text.DecimalFormat;
Problems Javadoc Declaration Console
<terminated> Simulation [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Nov 8, 2020, 7:31:09 PM – 7:31:14 PM)
Starting CloudSimExample6...
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
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.0: Broker: Trying to Create VM #5 in Datacenter_0
0.0: Broker: Trying to Create VM #6 in Datacenter_0
0.0: Broker: Trying to Create VM #7 in Datacenter_0
0.0: Broker: Trying to Create VM #8 in Datacenter_0
0.0: Broker: Trying to Create VM #9 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #6 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #6 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #7 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #7 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #8 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #8 to Host #1 failed by MIPS
[VmScheduler.vmCreate] Allocation of VM #9 to Host #0 failed by RAM
[VmScheduler.vmCreate] Allocation of VM #9 to Host #1 failed by MIPS
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: VM #2 has been created in Datacenter #2, Host #0
Type here to search
ENG IN 7:31 PM 11/8/2020


eclipse-workspace - cloudsim1/examples/org/cloudbus/cloudsim/examples/Simulation.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud
clouds
cloudsi
cloudsim
cloudsim1
examples
sources
JRE System Library [jdk-15]
Referenced Libraries
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
CloudSimSample
cloudsimset
Simulation.java
DatacenterBroker.java
Datacenter.java
1 package org.cloudbus.cloudsim.examples;
5 import java.text.DecimalFormat;
Problems Javadoc Declaration Console
<terminated> Simulation [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Nov 8, 2020, 7:31:09 PM – 7:31:14 PM)
[VmScheduler.vmCreate] Allocation of VM #9 to Host #1 failed by MIPS
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: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
0.1: Broker: Creation of VM #6 failed in Datacenter #2
0.1: Broker: Creation of VM #7 failed in Datacenter #2
0.1: Broker: Creation of VM #8 failed in Datacenter #2
0.1: Broker: Creation of VM #9 failed in Datacenter #2
0.1: Broker: Trying to Create VM #6 in Datacenter_1
0.1: Broker: Trying to Create VM #7 in Datacenter_1
0.1: Broker: Trying to Create VM #8 in Datacenter_1
0.1: Broker: Trying to Create VM #9 in Datacenter_1
0.2: Broker: VM #6 has been created in Datacenter #3, Host #0
0.2: Broker: VM #7 has been created in Datacenter #3, Host #0
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
1.Cloudlet Id:36,Cloudlet Length:1212
2.Cloudlet Id:0,Cloudlet Length:1267
3.Cloudlet Id:20,Cloudlet Length:1279
4.Cloudlet Id:11,Cloudlet Length:1314
5.Cloudlet Id:14,Cloudlet Length:1325
6.Cloudlet Id:28,Cloudlet Length:1343
7.Cloudlet Id:17,Cloudlet Length:1405
8.Cloudlet Id:34,Cloudlet Length:1426
9.Cloudlet Id:2,Cloudlet Length:1441
10.Cloudlet Id:15,Cloudlet Length:1467
Type here to search
ENG IN 7:32 PM 11/8/2020


eclipse-workspace - cloudsim1/examples/org/cloudbus/cloudsim/examples/Simulation.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud
clouds
cloudsi
cloudsim
cloudsim1
examples
sources
JRE System Library [jdk-15]
Referenced Libraries
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
CloudSimSample
cloudsimset
Simulation.java
DatacenterBroker.java
Datacenter.java
1 package org.cloudbus.cloudsim.examples;
5 import java.text.DecimalFormat;
Problems Javadoc Declaration Console
<terminated> Simulation [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Nov 8, 2020, 7:31:09 PM – 7:31:14 PM)
27.Cloudlet Id:32,Cloudlet Length:2281
28.Cloudlet Id:9,Cloudlet Length:2293
29.Cloudlet Id:16,Cloudlet Length:2293
30.Cloudlet Id:3,Cloudlet Length:2391
31.Cloudlet Id:22,Cloudlet Length:2411
32.Cloudlet Id:30,Cloudlet Length:2420
33.Cloudlet Id:29,Cloudlet Length:2517
34.Cloudlet Id:37,Cloudlet Length:2754
35.Cloudlet Id:23,Cloudlet Length:2774
36.Cloudlet Id:4,Cloudlet Length:2801
37.Cloudlet Id:18,Cloudlet Length:2852
38.Cloudlet Id:12,Cloudlet Length:2896
39.Cloudlet Id:6,Cloudlet Length:2953
40.Cloudlet Id:8,Cloudlet Length:2973
0.2: Broker: Sending cloudlet 36 to VM #0
0.2: Broker: Sending cloudlet 0 to VM #1
0.2: Broker: Sending cloudlet 20 to VM #2
0.2: Broker: Sending cloudlet 11 to VM #3
0.2: Broker: Sending cloudlet 14 to VM #4
0.2: Broker: Sending cloudlet 28 to VM #5
0.2: Broker: Sending cloudlet 17 to VM #6
0.2: Broker: Sending cloudlet 34 to VM #7
0.2: Broker: Sending cloudlet 2 to VM #8
0.2: Broker: Sending cloudlet 15 to VM #9
0.2: Broker: Sending cloudlet 25 to VM #0
0.2: Broker: Sending cloudlet 24 to VM #1
0.2: Broker: Sending cloudlet 35 to VM #2
0.2: Broker: Sending cloudlet 26 to VM #3
0.2: Broker: Sending cloudlet 31 to VM #4
Type here to search
ENG IN 7:32 PM 11/8/2020


eclipse-workspace - cloudsim1/examples/org/cloudbus/cloudsim/examples/Simulation.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud
clouds
cloudsi
cloudsim
cloudsim1
examples
sources
JRE System Library [jdk-15]
Referenced Libraries
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
CloudSimSample
cloudsimset
Simulation.java
DatacenterBroker.java
Datacenter.java
1 package org.cloudbus.cloudsim.examples;
5 import java.text.DecimalFormat;
Problems Javadoc Declaration Console
<terminated> Simulation [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (Nov 8, 2020, 7:31:09 PM – 7:31:14 PM)
0.2: Broker: Sending cloudlet 23 to VM #4
0.2: Broker: Sending cloudlet 4 to VM #5
0.2: Broker: Sending cloudlet 18 to VM #6
0.2: Broker: Sending cloudlet 12 to VM #7
0.2: Broker: Sending cloudlet 6 to VM #8
0.2: Broker: Sending cloudlet 8 to VM #9
1.412: Broker: Cloudlet 36 received
1.514: Broker: Cloudlet 0 received
1.514: Broker: Cloudlet 20 received
1.514: Broker: Cloudlet 11 received
1.605: Broker: Cloudlet 17 received
1.624: Broker: Cloudlet 14 received
1.624: Broker: Cloudlet 28 received
1.715: Broker: Cloudlet 34 received
1.715: Broker: Cloudlet 2 received
1.715: Broker: Cloudlet 15 received
2.917: Broker: Cloudlet 25 received
3.028: Broker: Cloudlet 24 received
3.138: Broker: Cloudlet 35 received
3.138: Broker: Cloudlet 26 received
3.215: Broker: Cloudlet 19 received
3.2479999999999998: Broker: Cloudlet 31 received
3.2479999999999998: Broker: Cloudlet 5 received
3.443: Broker: Cloudlet 33 received
3.61: Broker: Cloudlet 1 received
3.7199999999999998: Broker: Cloudlet 21 received
4.958: Broker: Cloudlet 38 received
5.097: Broker: Cloudlet 10 received
5.238: Broker: Cloudlet 13 received
Type here to search
ENG IN 7:32 PM 11/8/2020



Result:

Thus the scheduling algorithm using cloudsim has been successfully executed.

Ex: 8 **Sharing files between VMs**

Date: 07.10.2020

Objective:

To find a procedure to transfer the files from one virtual machine to another virtual machine.

Requirement:

- ❖ VMware/Virtual Box
- ❖ Ubutu/Linux virtual machine
- ❖ Windows virtual machine

Theory:

To share files between a host computer and a virtual machine or between two virtual machines, you use the networking features of VMware Workstation. If you know how to share files between two physical computers on a network, you already know how to share files with a virtual machine.

This section describes four scenarios for sharing files between two systems, either a host computer and a virtual machine or two virtual machines, where

- Both systems run Windows operating systems, using Windows file sharing
- You are connecting from a Linux system to a Windows system, using smbmount
- You are connecting from a Windows system to a Linux system, using Samba
- Both systems run Linux operating systems, using NFS, FTP and Telnet

You can apply the same principles to share files between virtual machines. Configuration for FreeBSD guests is similar to that for Linux guests.

The following scenarios assume you have set up your virtual machine using NAT networking. Besides giving the virtual machine a direct connection to the host computer's network, NAT networking sets up a virtual network adapter on the host computer. You can use this adapter, which connects to a virtual switch identified as `vmnet8`, to communicate between host and virtual machine. You can also connect two or more virtual machines using `vmnet8`.
In all cases, the user name you used to log in to the system from which you are connecting must be a user on the system to which you want to connect.

Sharing Files Between Two Windows Systems

To share files between two Windows systems (where one machine is a host and the other is a virtual machine, or both are virtual machines), be sure the file and printer sharing service is installed for both operating systems and the folders you want to share are marked as shared. Then you can browse from one system to the shared folder or folders on the other system.

Sharing Files by Connecting to a Windows System from a Linux System

To share files on a Windows system with a Linux system (by connecting to a Windows host from a Linux guest or connecting to a Windows guest from a Linux host), you can mark a folder as shared on the Windows system, then use the `smbmount` utility in the Linux system to mount the shared folder. For example, if you want to share the folder `C:\docs` on a Windows 2000 system called `win2k` with a Linux system at `/mnt/docs`, follow the steps below. You may want to set up a shell script to run these commands.

1. Set up the folder or folders to share on the Windows system.
2. Create a user account on the Windows system for the Linux system user name that you are using to connect to the Windows system.
   Otherwise, if you know the user name and password for a user account that can access the Windows system, you can specify that account on the command line.

3. From your Linux system, log in as root.
4. Add the Windows system's host name and IP address to the hosts file, if the system cannot be found by name.
5. Mount the Windows share on your Linux system. Enter the following command all on one line.
   ```
   mount -t smbfs -o username=<Windows system user account>,password=<password> //win2k/docs /mnt/docs
   ```

   (Substitute the appropriate host name, share and mount point for your systems.)

Now you are connected to the shared folder on the Windows system from your Linux system and can begin to share files between the two.

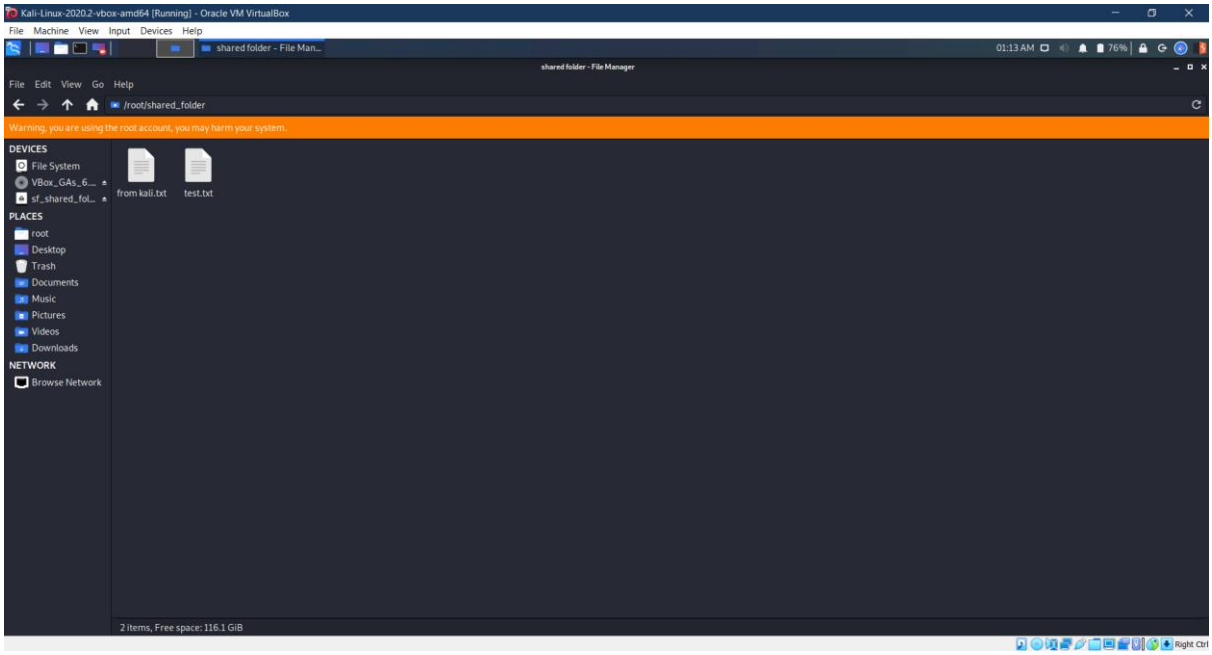
Sharing Files by Connecting to a Linux System from a Windows System

To share files on a Linux system with a Windows system (by connecting to a Linux host from a Windows guest or connecting to a Linux guest from a Windows host), you can run Samba on the Linux system and browse shared directories in the Linux file system from Network Neighborhood in the Windows system.

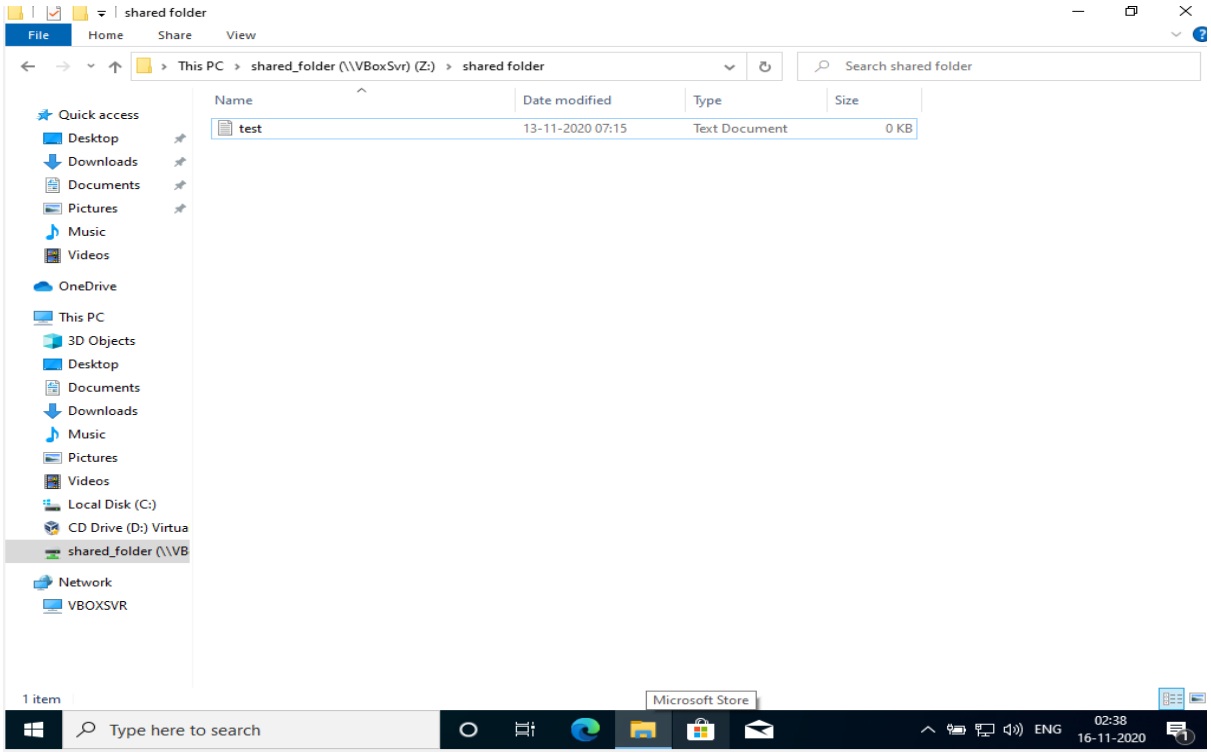
You need to modify Samba on the Linux host operating system so it recognizes the `vmnet8` switch, otherwise you cannot access the Linux file system. You need to do this even if you installed host-only networking (as Samba is installed when you install host-only networking with VMware Workstation).

Procedure and implementation:

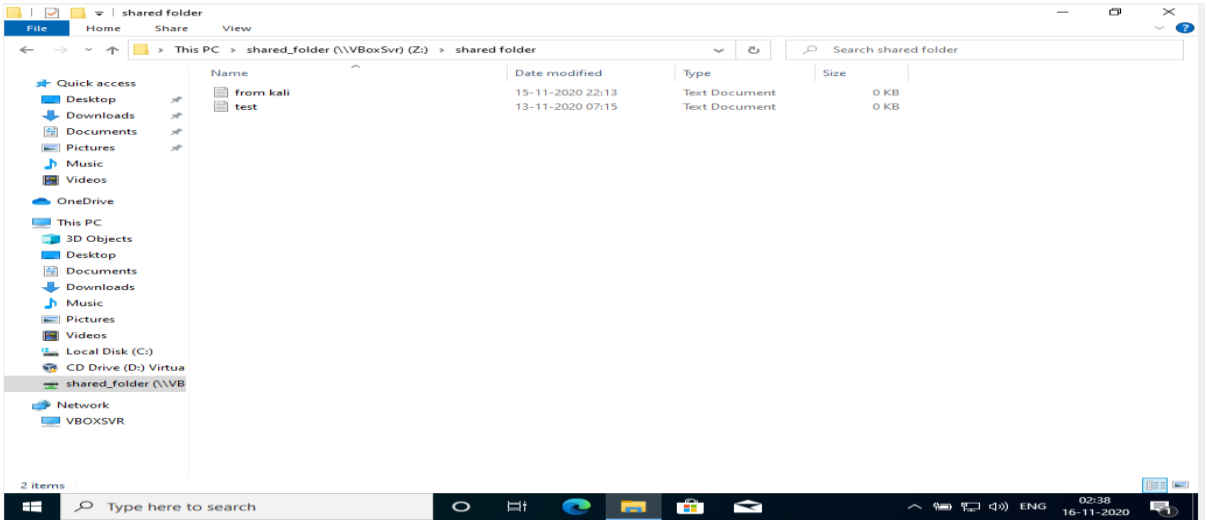
- Download VMware workstation 16 Pro
- Install Virtual Machine using Ubuntu iso file
- Install Virtual Machine using Windows iso file
- Create files in Virtual Machine 1 in shared folder.



- Before file transfer the shared folder in Virtual Machine 2 appears as below



❖ After file transfer the shared folder in Virtual Machine 2 appears as below



Result:

The file transfer between two Virtual Machines in Vmware Workstation 16 Pro has been successfully executed.

Ex: 9 **Trystack virtual machines**

Date: 14.10.2020

Objective:

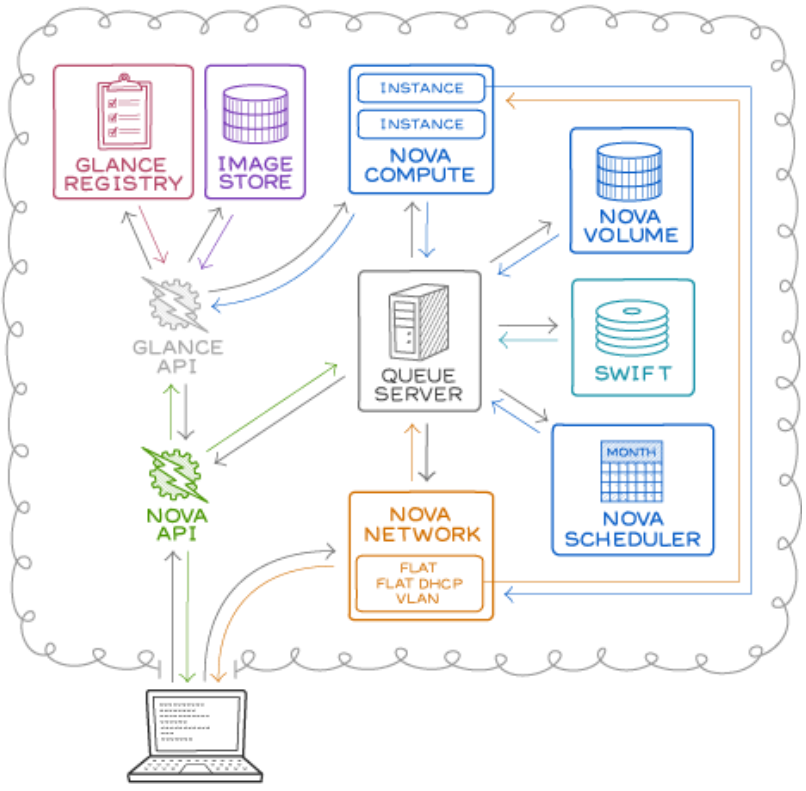
To explore trystack/openstack platform and create a virtual machine on platform managed by Openstack sandbox.

Requirements:

- Web browser
- Open stack Sandbox

Theory:

OpenStack is an open source platform, which offers powerful virtual servers and required services for cloud computing. It is mostly deployed as Infrastructure-as-a-service (IaaS), which aims to provide hardware tools and components for processing, storage, and networking resources throughout a data center.



OpenStack can be understood as a software platform that uses pooled virtual resources to build and manage clouds, both public and private ones. By default, OpenStack offers a couple of cloud-related services like networking, storage, image services, identity, etc., and can be clubbed with a few more to get a customized cloud optimization to support the cloud-native apps.

OpenStack community has declared around 9 components to be an integral part of OpenStack.

They are:
• Nova: This is the fundamental computing engine of OpenStack. It manages a large number of Virtual machines and other instances, which handle computing tasks.

• Swift: Swift is the storage system of OpenStack. It is used to store the objects and files. Instead of referring to the file and objects through the path, developers can instead refer to them through a unique identifier, which points to a file or piece of information and thereby allow the OpenStack to manage where to store the files. This reduces the effort of the developers to understand and worry about storage distribution. This also ensures that the data is being backed up, if in case of any failure of the machine or network loss.

• Cinder: Cinder is known as the block storage component of OpenStack. This functions in a way, analogous to the traditional ways of locating and accessing specific locations on a disk or a drive.
• Neutron: As the name suggests, Neutron is the component that enables networking in OpenStack. It ensures that each component within the OpenStack is well connected with other components, to establish good communication amongst them.

• Horizon: Horizon is the dashboard of the OpenStack system. It provides all the possibilities for the system administrators to access and manage the cloud. This is the first component that everyone ″sees” upon starting to use the OpenStack. Developers will be able to access and deal with all the components through the Application Programming Interface (API) also, while Horizon is the only place through which the system admins will be interacting with the OpenStack architecture.

• Keystone: Keystone is the component that provides the identity services for all the users. It basically contains a central list of all the users of the OpenStack cloud, mapped to the accessible services of the OpenStack. It provides a way for multiple accesses by allowing the developers to map their existing user access methods to the Keystone.

• Glance: Glance provides the image services in OpenStack, where images refer to the virtual copies of the hard disks. Glance helps in allocating these images to be used as templates while assigning new virtual machine instances.

• Ceilometer: Ceilometer provides telemetry services to its users. It performs a close regulation of each user's cloud components' usage and provides a bill for the services used. Think of Ceilometer as a component to meter the usage and report the same to individual users.

• Heat: Heat is that component of the OpenStack which allows developers to store the requirements of a cloud application in a file so that all the resources necessary for a program are available at hand. It thus provides an infrastructure to manage a cloud application. It is an orchestration instrument of OpenStack.

Procedure and implementation:

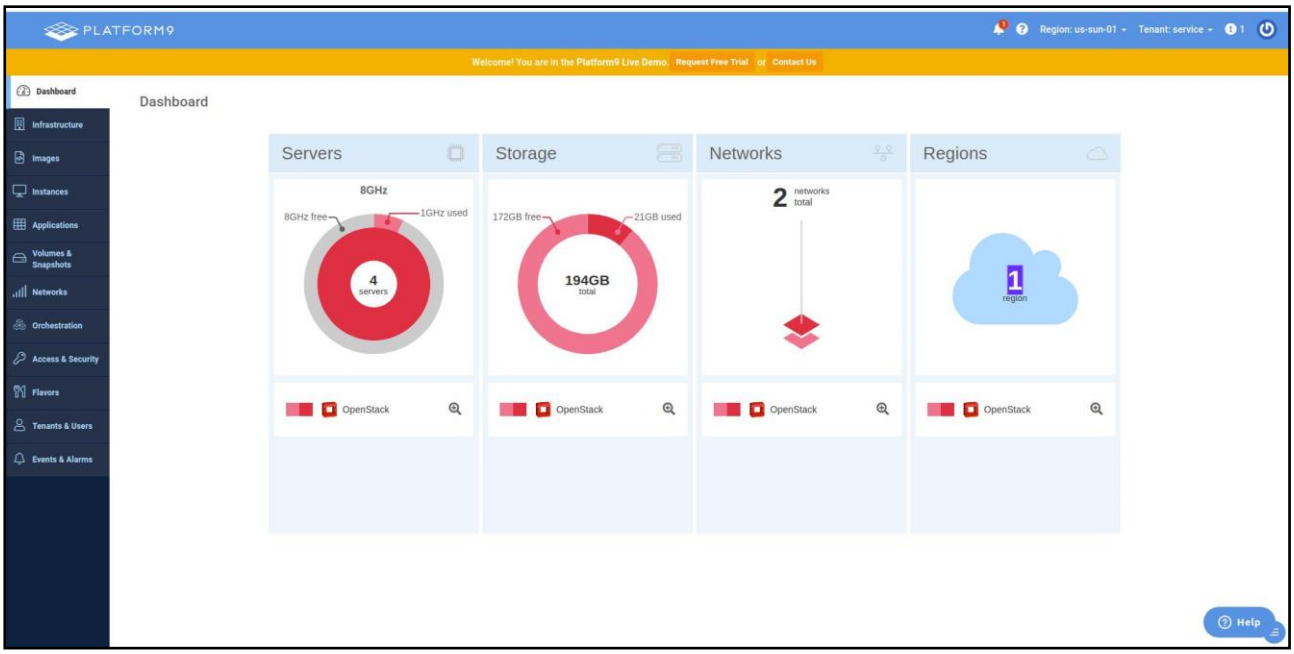
1. Setting up an account:

1.1. Visit https://platform9.com/managed-openstack/ and create a free account.

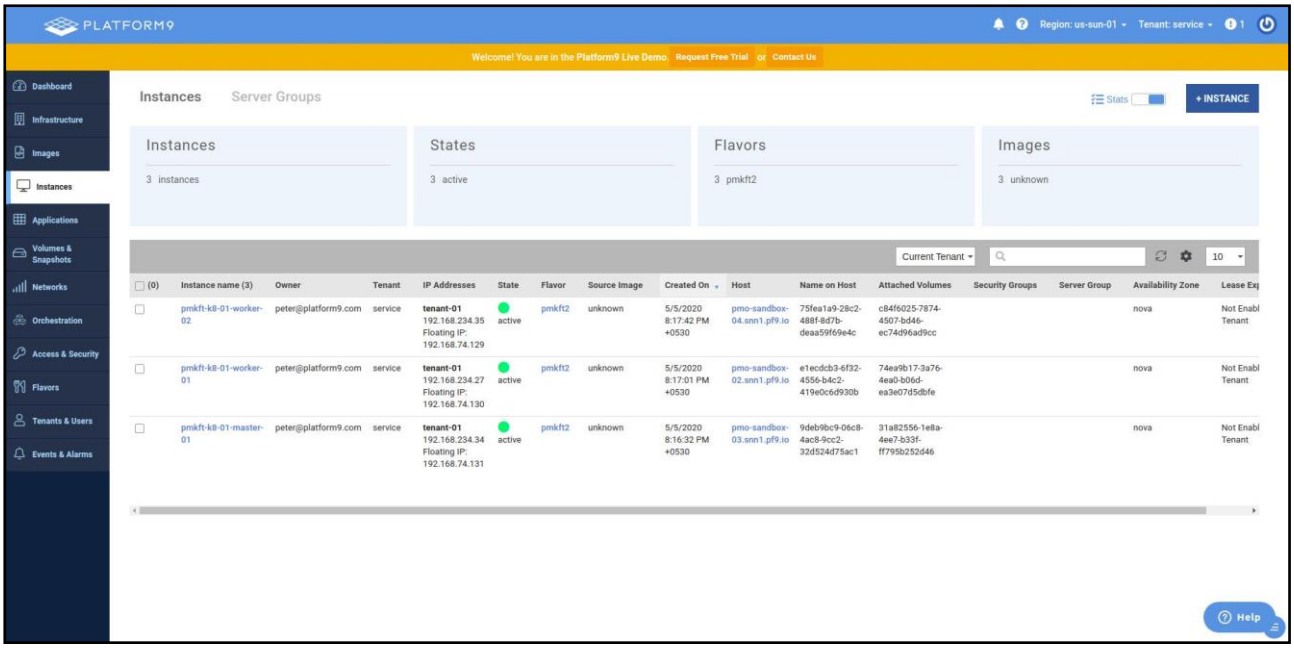
1.2. Login to the account and move to the sandbox mode.

2. Creating a VM instance:

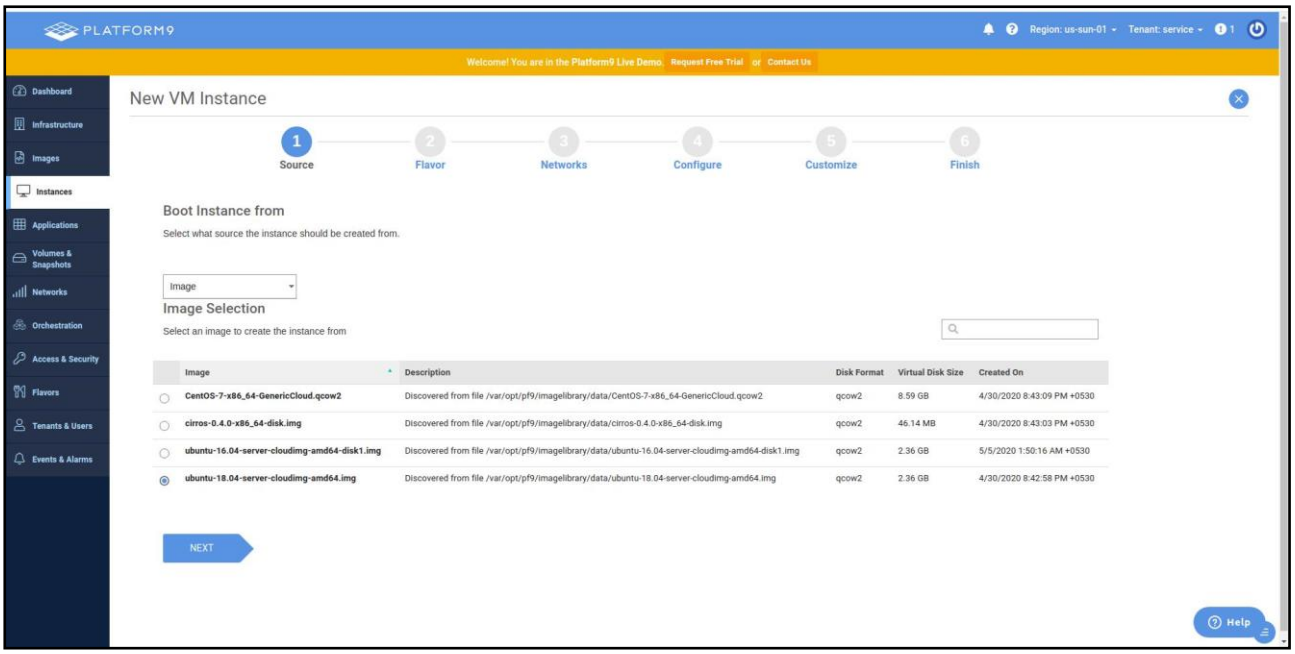
2.1. Login and move to the Dashboard



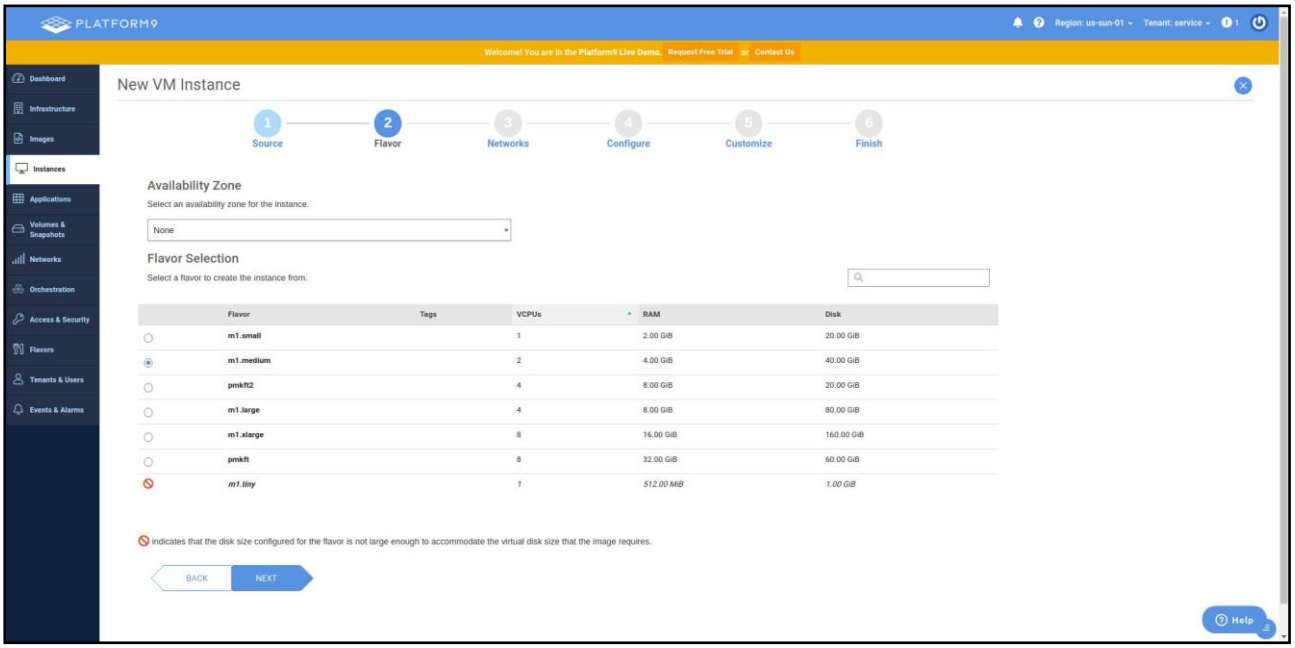
2.2 Click on the instances Tab to view current instances and choose on “create a new instance”.



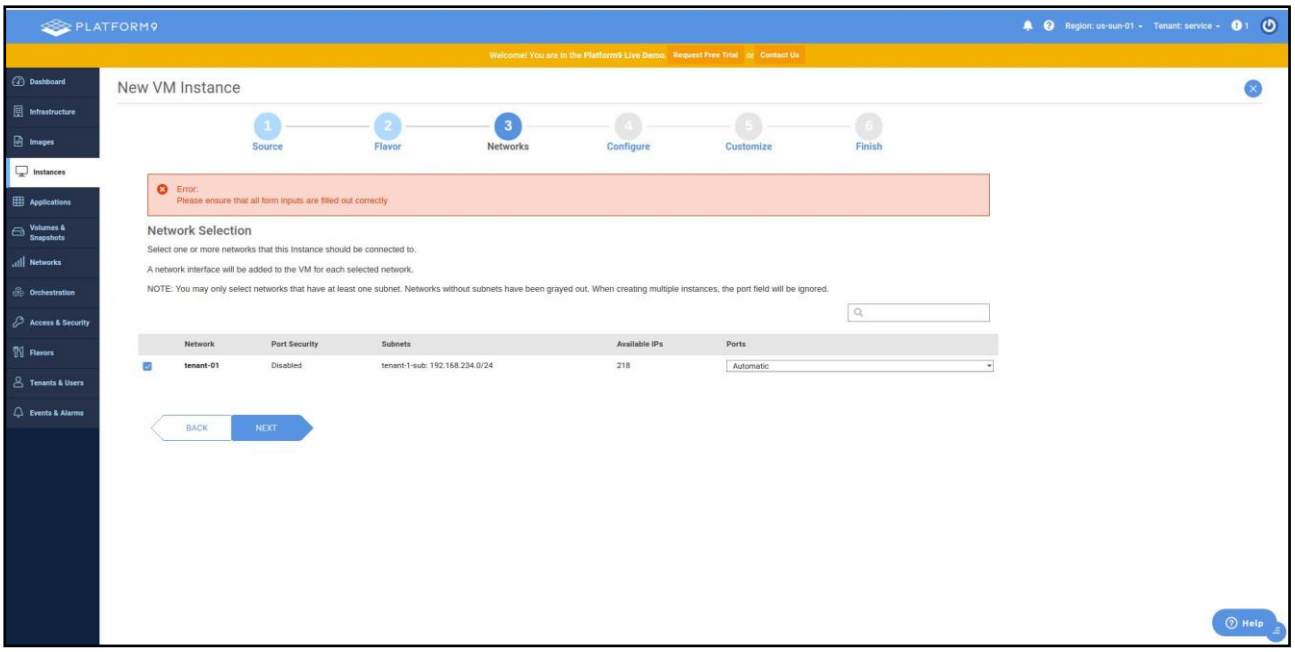
2.3 Choose an OS DISK Image to use.



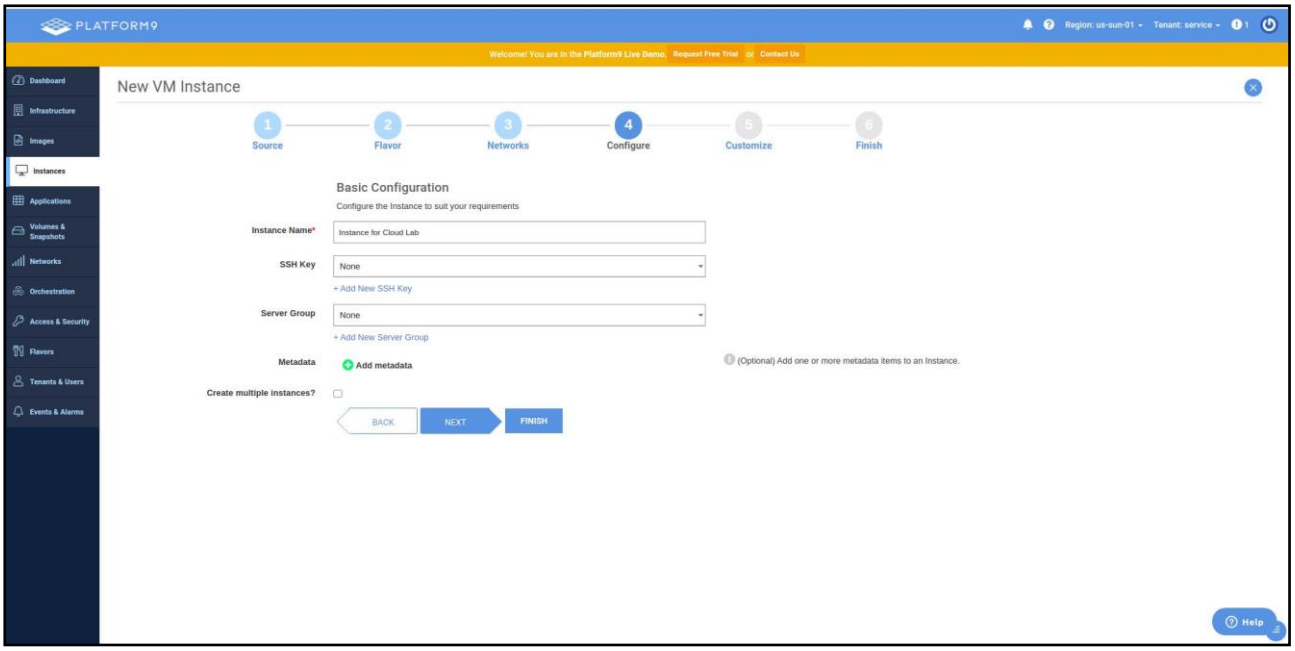
2.4 Next Choose the type of VM based on the CPU and Memory Requirements.



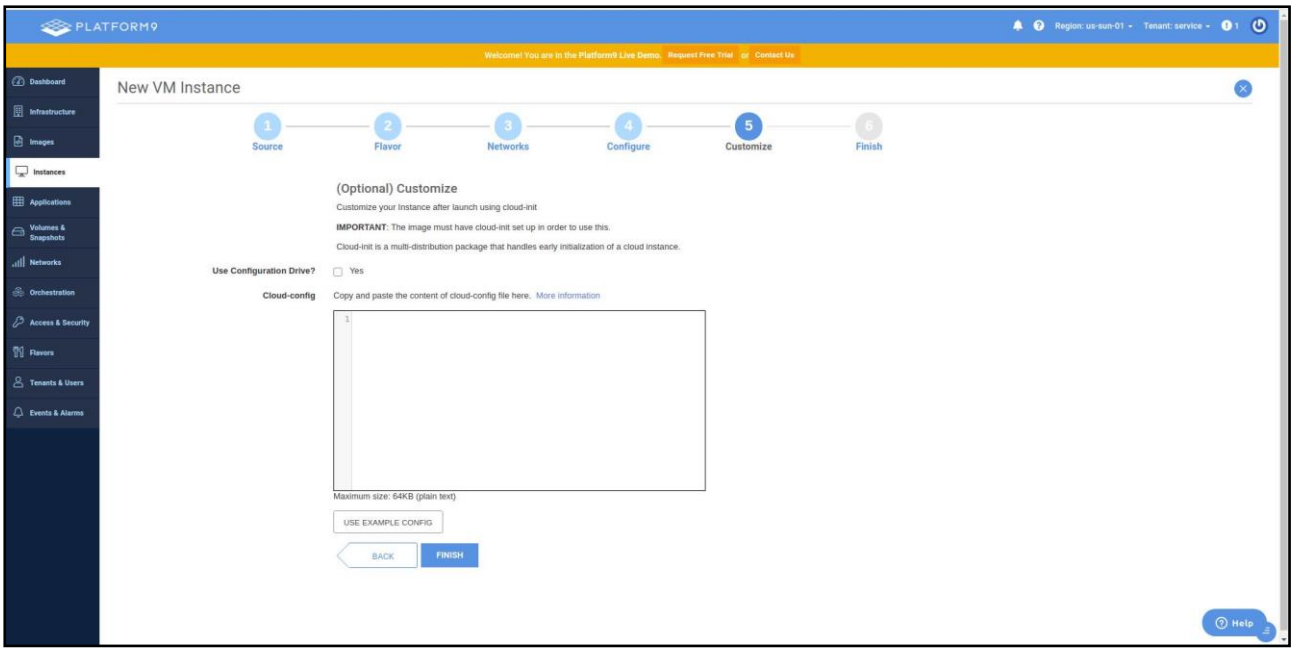
2.5 Set/Select Network configuration for the VM.



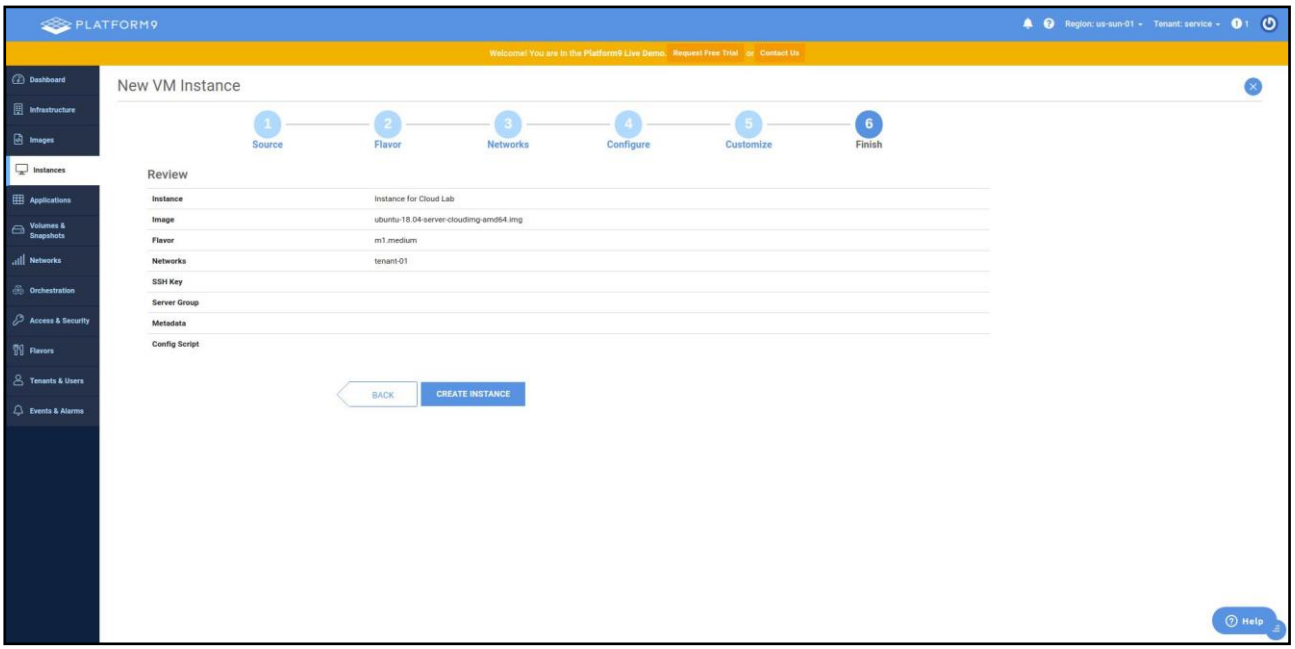
2.6 Set a name for the instance, then choose/create an ssh keypair for connecting to the VM also choose the number of such VMs to be created.



2.7 Optionally add any additional configuration for the VM.



2.8 Finally, Click on “create instance” to successfully create the VM.



Result:

Trystack/Openstack was explored and a VM was successfully created on Platform9 Managed Openstack sandbox.

Ex: 10 **Hadoop and Wordcount**

Date: 21.10.2020

Objective:

To understand the features of Hadoop and to do the program for word count.

Requirement:

Theory:

Map Reduce Model:

THE MAPREDUCE MODEL Traditional parallel computing algorithms were developed for systems with a small number of processors, dozens rather than thousands. So it was safe to assume that processors would not fail during a computation. At significantly larger scales this assumption breaks down, as was experienced at Google in the course of having to carry out many large-scale computations similar to the one in our word counting example. The MapReduce parallel programming abstraction was developed in response to these needs, so that it could be used by many different parallel applications while leveraging a common underlying fault-tolerant implementation that was transparent to application developers. Figure 11.1 illustrates MapReduce using the word counting example where we needed to count the occurrences of each word in a collection of documents. MapReduce proceeds in two phases, a distributed _map‘ operation followed by a distributed _reduce‘ operation; at each phase a configurable number of M _mapper‘ processors and R _reducer‘ processors are assigned to work on the problem (we have usedM = 3 and R = 2 in the illustration). The computation is coordinated by a single master process (not shown in the figure). A MapReduce implementation of the word counting task proceeds as follows: In the map phase each mapper reads approximately 1/M th of the input (in this case documents), from the global file system, using locations given to it by the master. Each mapped then performs a _map‘ operation to compute word frequencies for its subset of documents. These frequencies are sorted by the words they represent and written to the local file system of the mapper. At the next phase reducers are each assigned a subset of words; in our illustration the first reducer is assigned w1 and w2 while the second one handles w3 and w4. In fact during the map

phase itself each mapper writes one file per reducer, based on the words assigned to each reducer, and keeps the master informed of these file locations. The master in turn informs the reducers where the partial counts for their words have been stored on the local files of respective mappers; the reducers then make remote procedure call requests to the mappers to fetch these. Each reducer performs a reduce‘ operation that sums up the frequencies for each word, which are finally written back to the GFS file system.

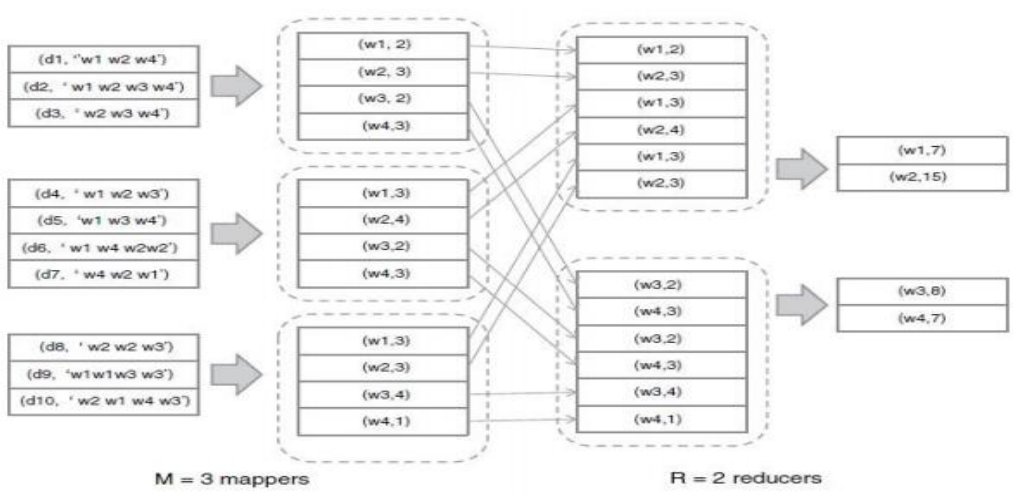


FIGURE 11.1. MapReduce model

The MapReduce programming model generalizes the computational structure of the above example. Each map operation consists of transforming one set of key-value pairs to another: Map: (k1, v1) → [(k2, v2)]…… (11.4) In our example each map operation takes a document indexed by its id and emits a list if wordcount pairs indexed by word-id: (dk, [w1 . . .wn]) → [(wi, ci)]. The reduce operation groups the results of the map step using the same key k2 and performs a function f on the list of values that correspond to each 43 Reduce: (k2, [v2]) → (k2, f ([v2]))……. (11.5) In our example each reduce operation sums the frequency counts for each word. The implementation also generalizes. Each mapper is assigned an input-key range (set of values for k1) on which map operations need to be performed. The mapper writes results of its map operations to its local disk in R partitions, each corresponding to the output-key range (values of k2) assigned to a particular reducer, and informs the master of these locations. Next each reducer fetches these pairs from the respective mappers and performs reduce operations for each key k2 assigned to it. If a processor fails during the execution, the master detects this through regular heartbeat communications it maintains with each worker, wherein

updates are also exchanged regarding the status of tasks assigned to workers. If a mapper fails, then the master reassigns the key-range designated to it to another working node for re-execution. Note that re-execution is required even if the mapper had completed some of its map operations, because the results were written to local disk rather than the GFS. On the other hand if a reducer fails only its remaining tasks (values k2) are reassigned to another node, since the completed tasks would already have been written to the GFS. Finally, heartbeat failure detection can be fooled by a wounded task that has a heartbeat but is making no progress: Therefore, the master also tracks the overall progress of the computation and if results from the last few processors in either phase are excessively delayed, these tasks are duplicated and assigned to processors who have already completed their work. The master declares the task completed when any one of the duplicate workers complete. Such a fault-tolerant implementation of the MapReduce model has been implemented and is widely used within Google; more importantly from an enterprise perspective, it is also available as an open source implementation through the Hadoop project along with the HDFS distributed file system. The MapReduce model is widely applicable to a number of parallel computations, including database-oriented tasks which we cover later. Finally we describe one more example, that of indexing a large collection of documents, or, for that matter any data including database records: The map task consists of emitting a word-document/record id pair for each word: (dk, [w1 . . .wn]) → [(wi, dk)]. The reduce step groups the pairs by word and creates an index entry for each word: [(wi, dk)] → (wi, [di1 . . . dim]). Indexing large collections is not only important in web search, but also a critical aspect of handling structured data; so it is important to know that it can be executed efficiently in parallel using MapReduce. Traditional parallel databases focus on rapid query execution against data warehouses that are updated infrequently; as a result these systems often do not parallelize index creation sufficiently well.

Procedure and implementation:

- Download the Java 8 Package. Save this file in your home directory.
- Extract the Java Tar File.

- ❖ Download the Hadoop 2.7.3 Package
- ❖ Extract the Hadoop tar File.
- ❖ Add the Hadoop and Java paths in the environmental variable setting
- ❖ To make sure that Java and Hadoop have been properly installed on your system, execute the java -version and Hadoop version commands.
- ❖ Edit the Hadoop Configuration files.
- ❖ Open *core-site.xml* and edit the property mentioned below inside configuration tag:

core-site.xml informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS &MapReduce.

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

- ❖ Edit *hdfs-site.xml* and edit the property mentioned below inside configuration tag:

hdfs-site.xml contains configuration settings of HDFS daemons (i.e.NameNode,

DataNode, Secondary NameNode). It also includes the replication factor and block sizeof HDFS.

```
<configuration>
<property>
<name>dfs.replication</name>
 <value>1</value>
 </property>
 <property>
 <name>dfs.namenode.name.dir</name>
 <value>file:///C:/hadoop-3.3.0/data/namenode</value>
 </property>
 <property>
 <name>dfs.datanode.data.dir</name>
 <value>/C:/hadoop-3.3.0/data/datanode</value>
 </property>
</configuration>
```

❖ Edit the *mapred-site.xml* file and edit the property mentioned below inside configuration tag: *mapred-site.xml* contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc. In some cases, mapred-site.xml file is not available. So, we have to create the mapredsite.

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

❖ Edit *yarn-site.xml* and edit the property mentioned below inside configuration tag:
yarn-site.xml contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

```
<configuration>

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
  </property>
  <property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>

</configuration>
```

❖ Edit *hadoop-env.sh* and add the Java Path as mentioned below:
hadoop-env.sh contains the environment variables that are used in the script to run Hadoop like Java home path, etc.

```
@rem The java implementation to use.  Required.
set JAVA_HOME=C:\java\jdk1.8.0_271

@rem The jsvc implementation to use. Jsvc is required to run secure datanodes.
@rem set JSVC_HOME=%JSVC_HOME%

@rem set HADOOP_CONF_DIR=

@rem Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
if exist %HADOOP_HOME%\contrib\capacity-scheduler (
  if not defined HADOOP_CLASSPATH (
    set HADOOP_CLASSPATH=%HADOOP_HOME%\contrib\capacity-scheduler\*.jar
  ) else (
    set HADOOP_CLASSPATH=%HADOOP_CLASSPATH%;%HADOOP_HOME%\contrib\capacity-scheduler\*.jar
  )
)
```

❖ Go to Hadoop home directory and format the NameNode.

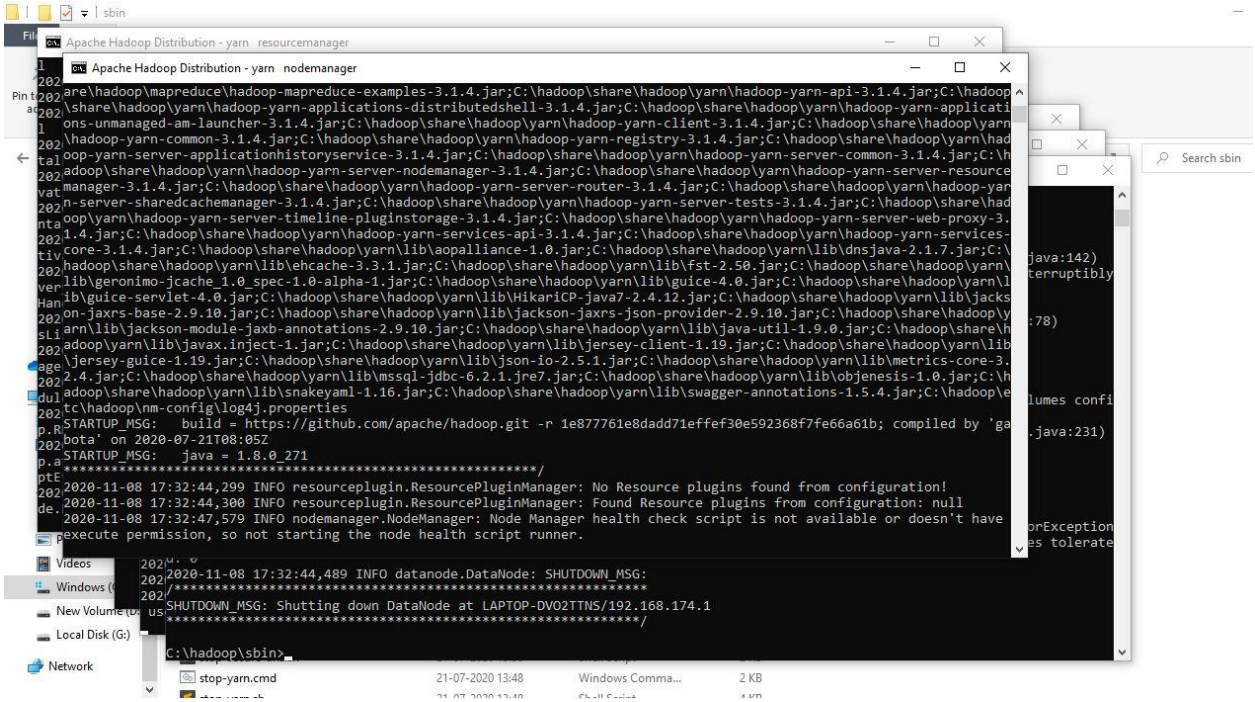
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>hdfs namenode -format
2020-11-16 19:39:45,026 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = DESKTOP-2N7VHQK/192.168.175.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.0
STARTUP_MSG:   classpath = C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\share\hadoop\common;C:\hadoop-3.3.0\share\hadoop\c
ommon\lib\accessors-smart-1.2.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop-
3.3.0\share\hadoop\common\lib\asm-5.0.4.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\audience-annotations-0.5.0.jar;C:\ha
doop-3.3.0\share\hadoop\common\lib\avro-1.7.7.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\checker-qual-2.5.2.jar;C:\hado
op-3.3.0\share\hadoop\common\lib\commons-beanutils-1.9.4.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-cli-1.2.jar
;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-codec-1.11.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-collecti
ons-3.2.2.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-compress-1.19.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\
commons-configuration2-2.1.1.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-daemon-1.0.13.jar;C:\hadoop-3.3.0\share
\hadoop\common\lib\commons-io-2.5.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-lang3-3.7.jar;C:\hadoop-3.3.0\shar
e\hadoop\common\lib\commons-logging-1.1.3.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-math3-3.1.1.jar;C:\hadoop-
3.3.0\share\hadoop\common\lib\commons-net-3.6.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-text-1.4.jar;C:\hadoop
-3.3.0\share\hadoop\common\lib\curator-client-4.2.0.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\curator-framework-4.2.0.
jar;C:\hadoop-3.3.0\share\hadoop\common\lib\curator-recipes-4.2.0.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\dnsjava-2.
1.7.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\failureaccess-1.0.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\gson-2.2.4
.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\guava-27.0-jre.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\hadoop-annotatio
ns-3.3.0.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\hadoop-auth-3.3.0.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\hadoo
p-shaded-protobuf_3_7-1.0.0.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\htrace-core4-4.1.0-incubating.jar;C:\hadoop-3.3.
0\share\hadoop\common\lib\httpclient-4.5.6.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\httpcore-4.4.10.jar;C:\hadoop-3.3
.0\share\hadoop\common\lib\j2objc-annotations-1.1.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\jackson-annotations-2.10.3
.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\jackson-core-2.10.3.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\jackson-cor
```

❖ Once the NameNode is formatted, go to hadoop-2.7.3/sbin directory and start all the daemons.

Command: cd hadoop-2.7.3/sbin

❖ Either you can start all daemons with a single command or do it individually.

Command: ./start-all.cmd



❖ Now hadoop has been successfully installed.To run the word count program, first create a input directory and move the input file to that directory using the command:

hadoop fs -mkdir /input_dir

hadoop fs -put C:/input_file.txt /input_dir

Administrator: Command Prompt

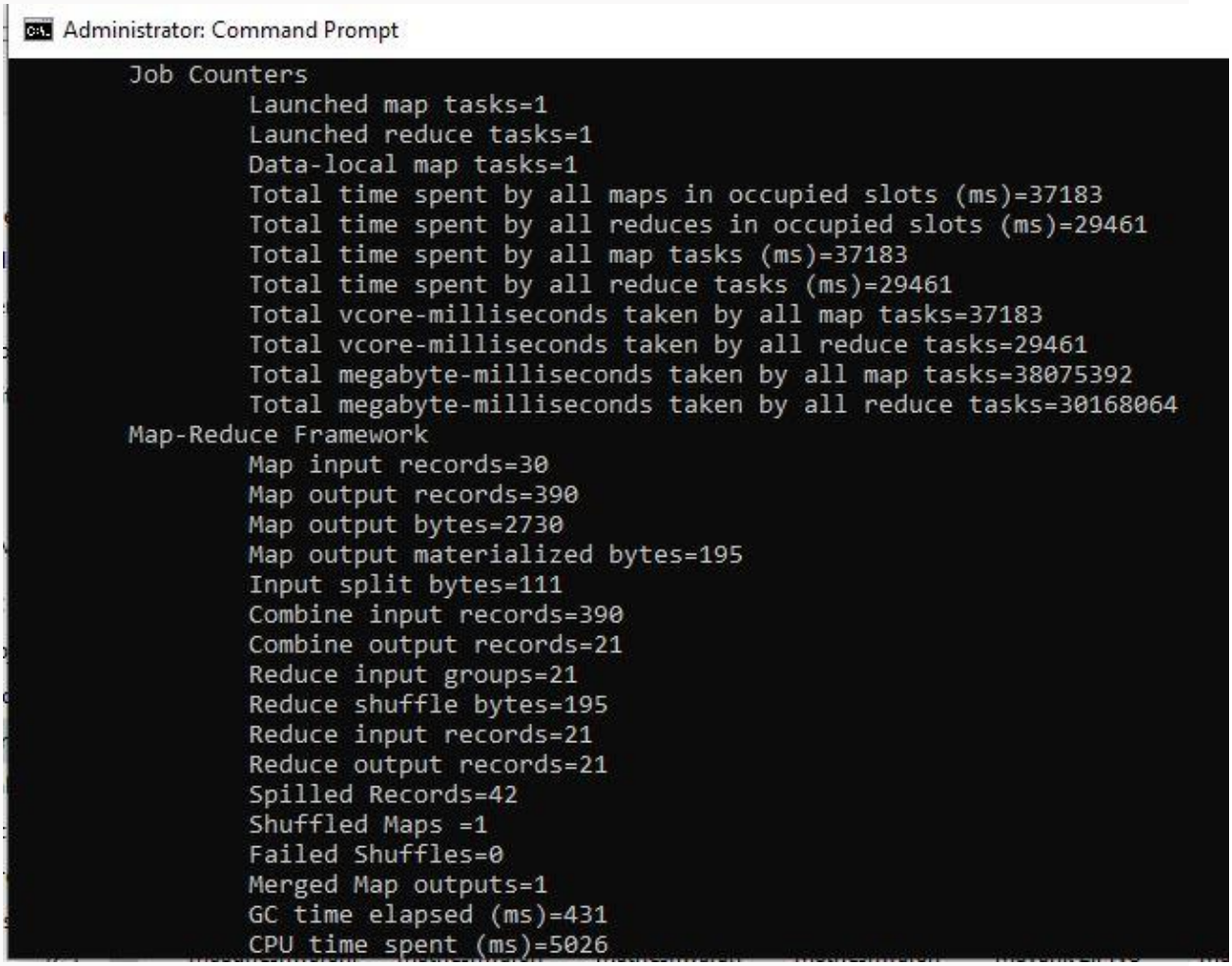
```
C:\WINDOWS\system32>cd
C:\WINDOWS\system32

C:\WINDOWS\system32>cd ..

C:\Windows>cd ..

C:\>cd hadoop-3.3.0

C:\hadoop-3.3.0>cd sbin

C:\hadoop-3.3.0\sbin>start-dfs

C:\hadoop-3.3.0\sbin>start-yarn
starting yarn daemons

C:\hadoop-3.3.0\sbin>jps
2640 NameNode
8260 ResourceManager
2232 NodeManager
3432 DataNode
12188 Jps

C:\hadoop-3.3.0\sbin>stop-dfs
SUCCESS: Sent termination signal to the process with PID 6552.
SUCCESS: Sent termination signal to the process with PID 1928.

C:\hadoop-3.3.0\sbin>stop-yarn
stopping yarn daemons
SUCCESS: Sent termination signal to the process with PID 5952.
SUCCESS: Sent termination signal to the process with PID 6324.

INFO: No tasks running with the specified criteria.

C:\hadoop-3.3.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-3.3.0\sbin>cd/
```

Administrator: Command Prompt

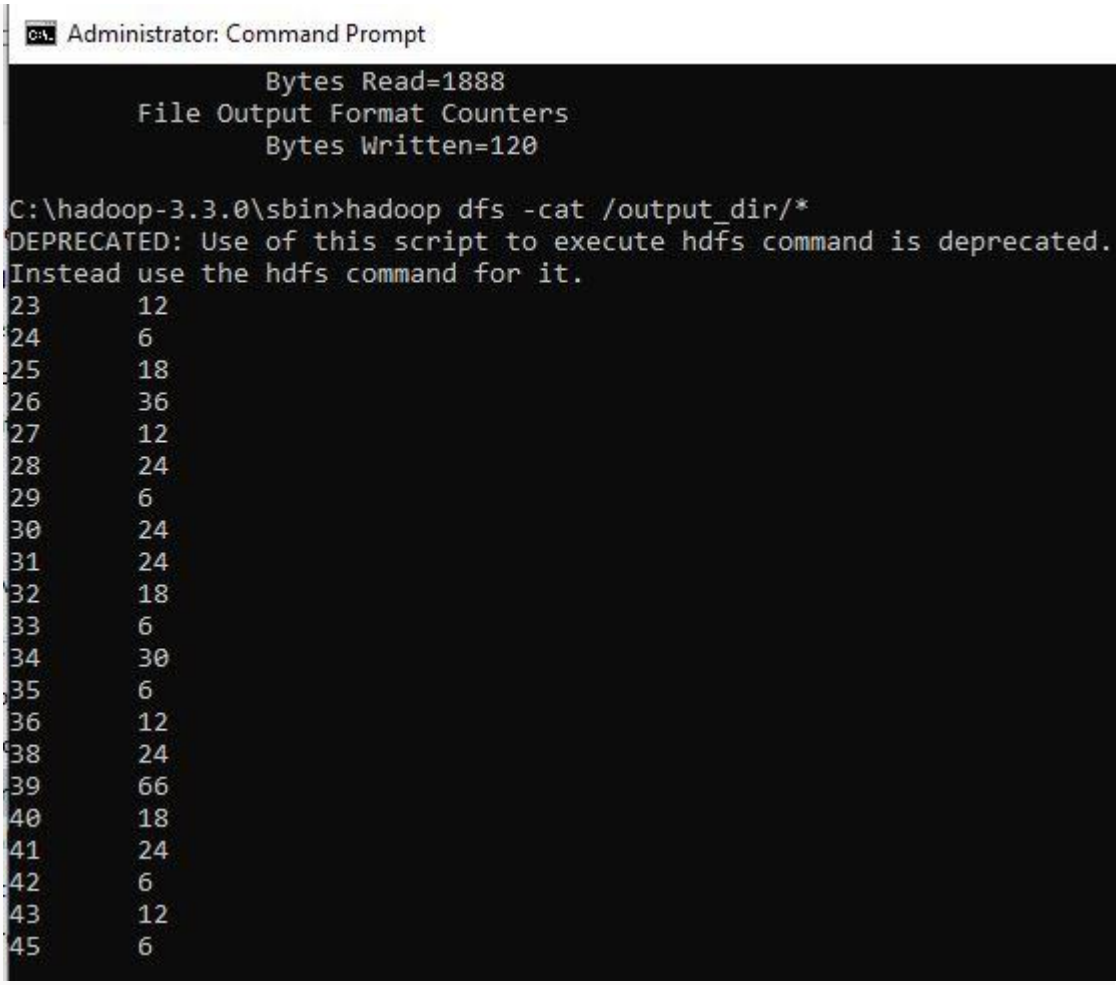
```
C:\>hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Safe mode is OFF

C:\>hadoop fs -mkdir /input_dir

C:\>hadoop fs -put C:/input_file.txt /input_dir

C:\>hadoop fs -ls /input_dir/
Found 1 items
-rw-r--r--   1 nandhini supergroup       1888 2020-11-15 14:35 /input_dir/input_file.txt

C:\>hadoop dfs -cat /input_dir/input_file.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
31   32   32  32   33   34   35   36   36   34   34   34  34
39   38   39  39   39   41   42   43   40   39   38   38  40
38   39   39  39   39   41   41   41   28   40   39   39  45
23   23   27  43   24   25   26   26   26   26   25   26  25
26   27   28  28   28   30   31   31   31   30   30   30  29
```

- Run the mapreduce program to compute the word count in the input file using the command :

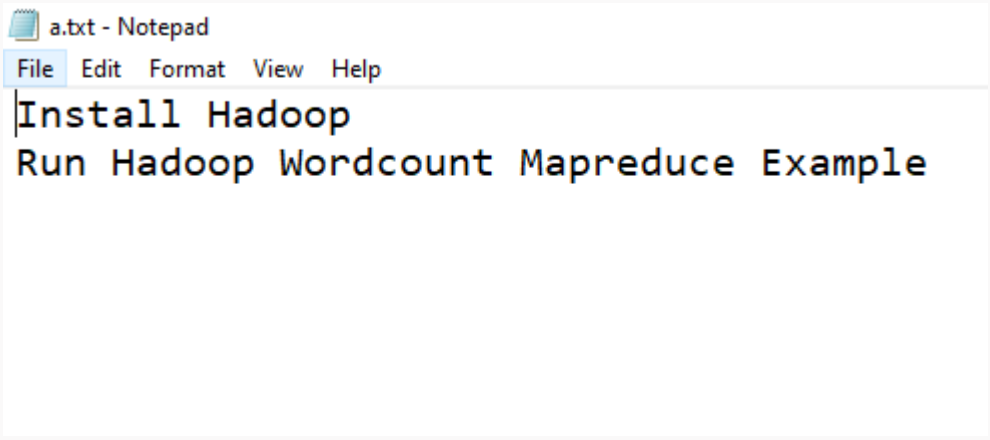
hadoop jar C:/MapReduceClient.jar wordcount /input_dir /output_dir

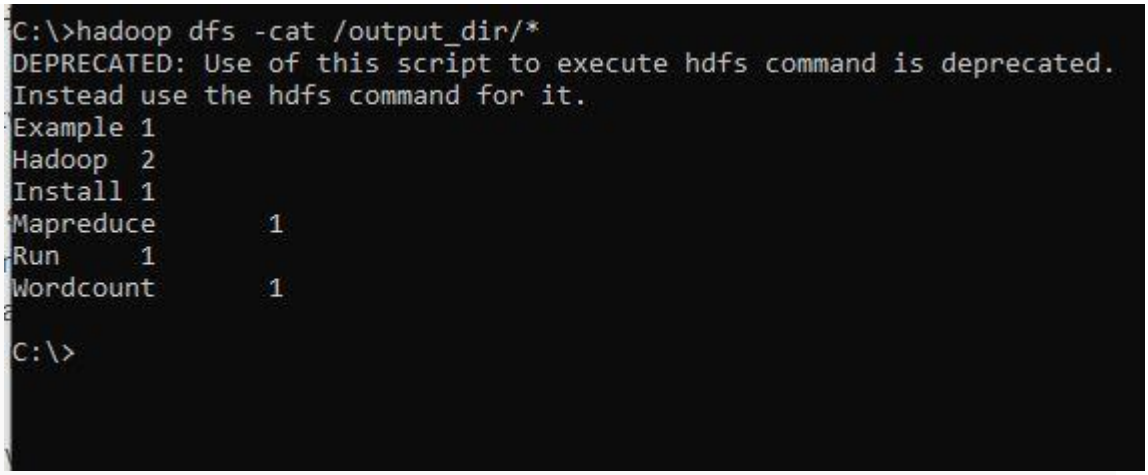


- Display the output


Administrator: Command Prompt
Bytes Read=1888
File Output Format Counters
Bytes Written=120
C:\hadoop-3.3.0\sbin>hadoop dfs -cat /output_dir/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
23 12
24 6
25 18
26 36
27 12
28 24
29 6
30 24
31 24
32 18
33 6
34 30
35 6
36 12
38 24
39 66
40 18
41 24
42 6
43 12
45 6

❖ Input file:


a.txt - Notepad
File Edit Format View Help
Install Hadoop
Run Hadoop Wordcount Mapreduce Example



Result:

Thus the word count program has been executed successfully using Hadoop.