MAHENDRA COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated by Anna University, Chennai

NAAC Accredited – Recognized U/S 2 (F) & 12 (B) of UGC Act 1956

Salem-Chennai Highway NH 79, Minnampalli, Salem-636106

DEPARTMENT OF INFORMATION TECHNOLOGY

CC335 – CLOUD COMPUTING LABORATORY

RECORD NOTE BOOK

CLASS: III YEAR / VI SEMESTER

MAHENDRA

COLLEGE OF ENGINEERING

NAAC Accredited

Approved by AICTE and Affiliated by Anna University, Chennai

Salem-Chennai Highway NH 79, Minnampalli, Salem-636106

Department of

_____

_____

LABORATORY RECORD

Certified to be the bonafide of work done by

Name: _____Register No: _____

Class: _____Branch: _____

Laboratory Name: _____

HEAD OF THE DEPARTMENT
DATE:

STAFF-IN-CHARGE

Submitted for the University Practical Examination on

_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

S.NO	DATE	TITLE	PAGE NO	MARK	SIGN

INTERNAL MARK

STAFF INCHARGE

MAHENDRA COLLEGE OF ENGINEERING

SALEM-CAMPUS, ATTUR MAIN ROAD, MINNAMPALLI, SALEM -636 106.

DEPARTMENT OF INFORMATION TECHNOLOGY

DEPARTMENT VISION AND MISSION

VISION

To become a department, producing graduates with good technical skills in emerging areas of Information Technology, through value based education and research.

MISSION

- To provide exposure to students to the emerging technologies in Hardware and Software.
- To inculcate students with sound application knowledge.
- To establish strong Industry- Institute Interaction.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

To ensure graduates

- Have proficiency in programming skills to design, develop and apply appropriate techniques, to solve complex engineering problems.
- Have knowledge to build, automate and manage business solutions using cutting edge technologies.
- Have excitement towards research in applied computer technologies

PROGRAM OUTCOMES (POs)

1. **Engineering knowledge**

Apply the knowledge of mathematics science engineering fundamentals and mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**

Identify, formulate, review research literature and analyze complex engineering problems researching substantiated conclusion using first principles of mathematics, natural sciences and engineering sciences.

3. **Design/develop of solution**

Design solutions for complex engineering problems and design systems components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.

4. **Conduct investigation of complex solutions**

Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**

Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineering and society**

Apply reasoning informed by the contextual knowledge to asses societal, health, safety and legal and cultural issues and the consequent responsibilities legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**

Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for the sustainable development and need for sustainable development.

8. **Ethics**

Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**

Function effectively as an individual and as a member or leader in diverse teams and in diverse teams and individual, and in multidisciplinary settings.

10. **Communication**

Communication effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give receive clear instructions.

11. **Project management and finance**

Demonstrate knowledge and understanding of the engineering and knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**

Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CCS335 –CLOUD COMPUTING LABORATORY

COURSE OBJECTIVES:

- To understand the principles of cloud architecture, models and infrastructure.
- To understand the concepts of virtualization and virtual machines.
- To gain knowledge about virtualization Infrastructure.
- To explore and experiment with various Cloud deployment environments.
- To learn about the security issues in the cloud environment.

LIST OF EXPERIMENTS

1. Install Virtualbox/VMware/ Equivalent open source cloud Workstation with different flavours of Linux or Windows OS on top of windows 8 and above.
2. Install a C compiler in the virtual machine created using a virtual box and executes Simple Programs.
3. Install Google App Engine. Create a hello world app and other simple web applications using python/java.
4. Use the GAE launcher to launch the web applications.
5. Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.
6. Find a procedure to transfer the files from one virtual machine to another virtual machine. 7. Install Hadoop single node cluster and run simple applications like wordcount.
8. Creating and Executing Your First Container Using Docker.
9. Run a Container from Docker Hub

TOTAL: 30 PERIODS

SOFTWARE AND HARDWARE REQUIREMENTS:

Software Requirements	Java or equivalent compiler GnuPG, N-Stalker or Equivalent Compiler
Hardware Requirements	Desktop Computer

COURSE OUTCOMES:

CO1: Understand the design challenges in the cloud.
CO2: Apply the concept of virtualization and its types.
CO3: Experiment with virtualization of hardware resources and Docker. CO4: Develop and deploy services on the cloud and set up a cloud environment.
CO5: Explain security challenges in the cloud environment.

CO's-PO & PSO's MAPPING:

CO's	PO's												PSO's		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
1	3	2	1	1	1	-	-	-	2	3	1	3	2	1	3
2	3	1	2	2	1	-	-	-	1	2	1	3	2	2	1
3	2	3	2	3	1	-	-	-	3	1	1	3	1	1	1
4	1	2	3	3	3	-	-	-	3	3	1	2	1	3	3
5	2	3	3	1	3	-	-	-	2	2	1	2	2	2	3
AVg.	2.2	2.2	2.2	2	1.8	-	-	-	2.2	2.2	1	2.6	1.6	1.8	2.2

1-low,2-medium,-high,'-'-nocorrelation

Ex.No.1	**Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.**
Date:	

AIM:

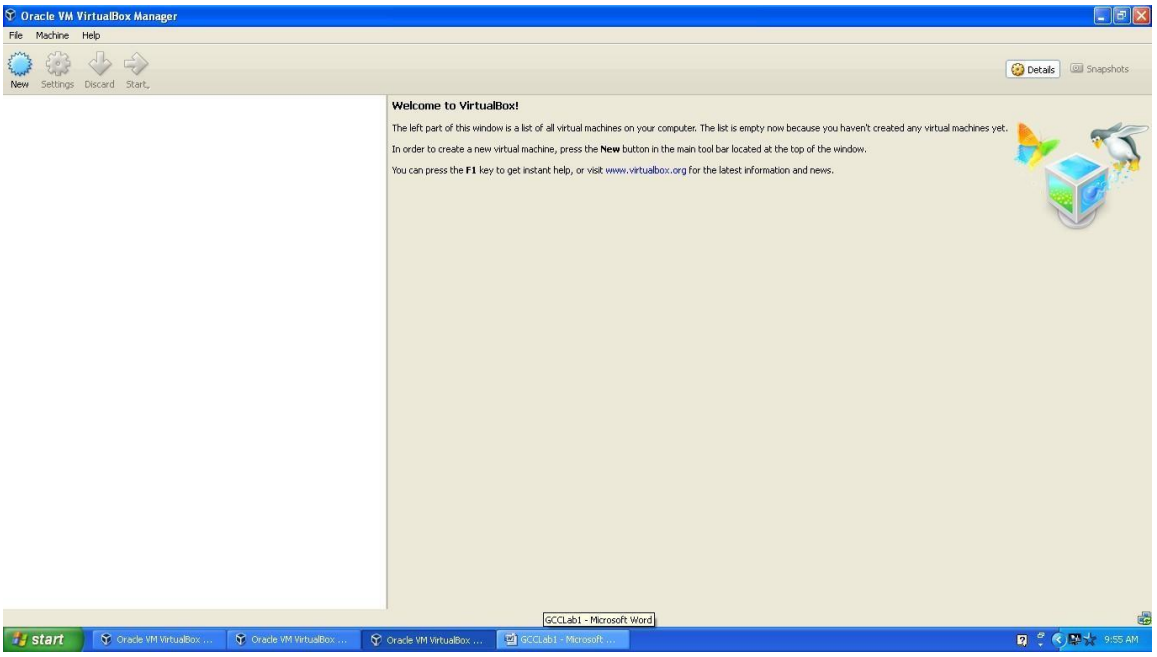
To find procedure to run the virtual machine of different configuration and check how many virtual machines can be utilized at particular time.

PROCEDURE:

Installing Ubuntu using Oracle Virtual Box

Step1:

Open Oracle virtual box manager and click create new -> virtual machine.

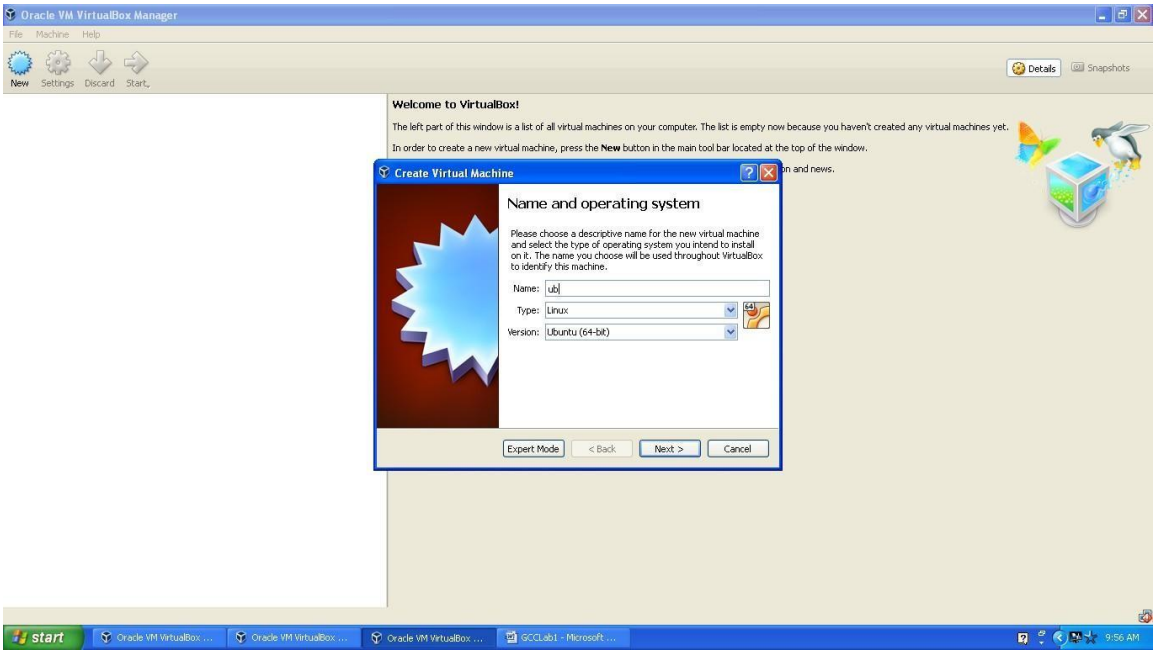


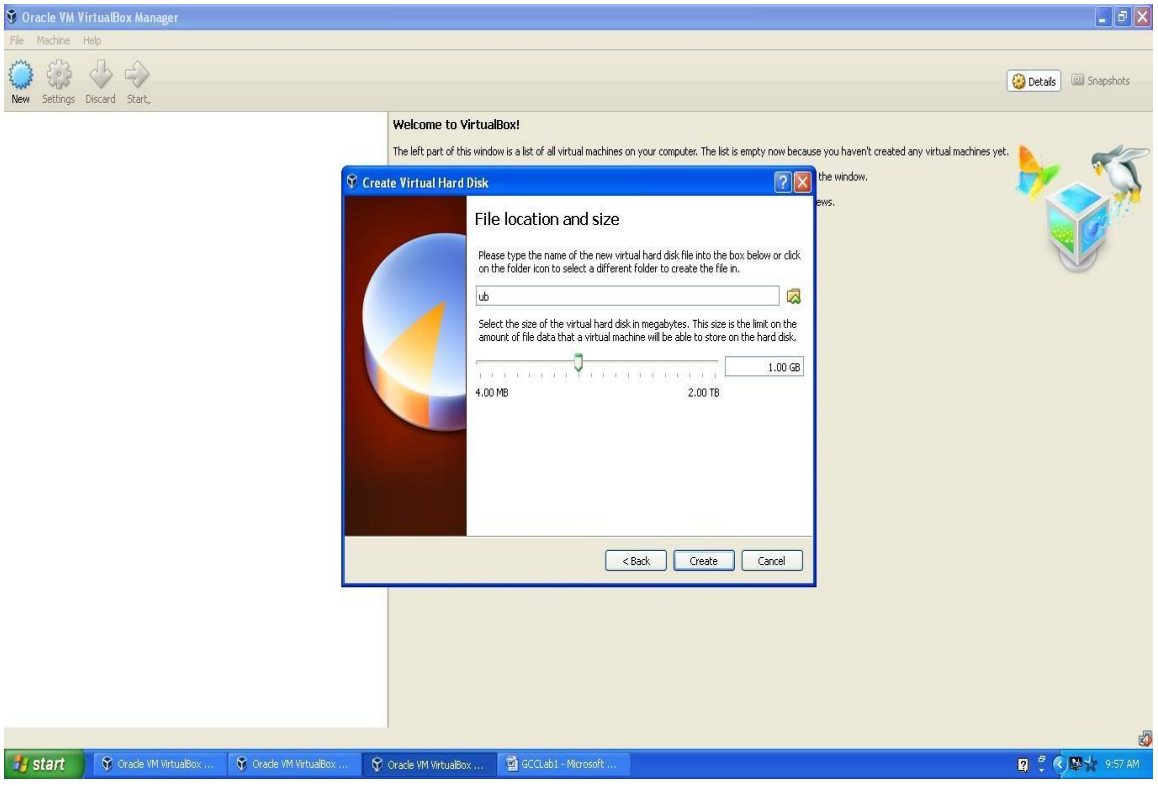
Step 2:

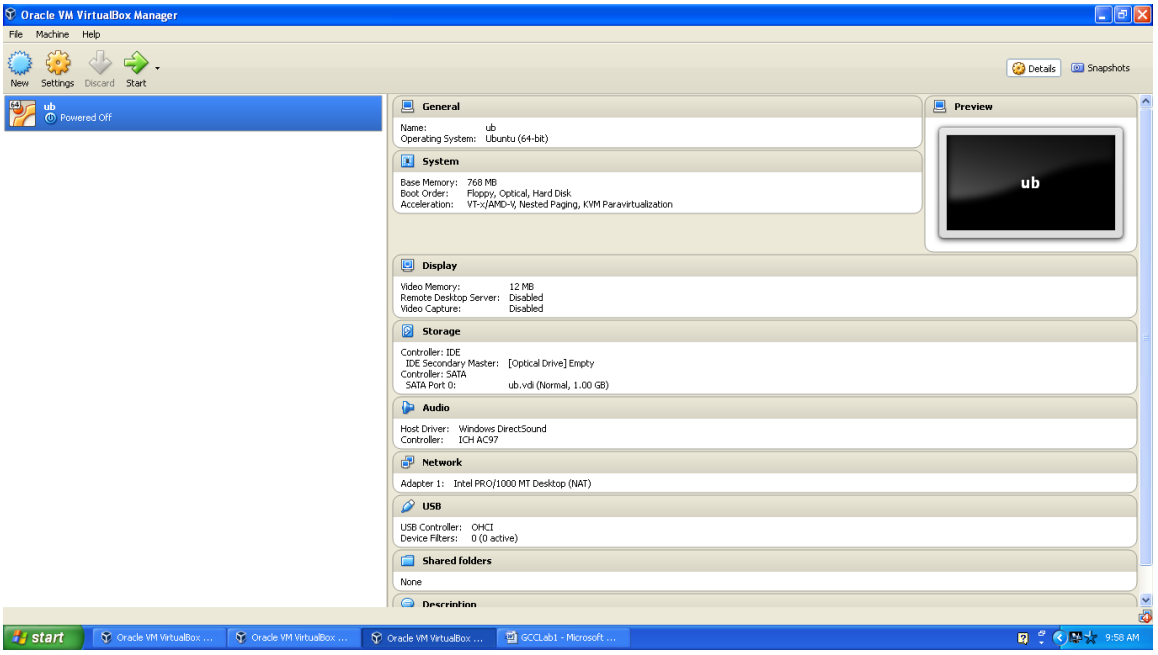
Provide a name for the virtual machine and select the hard disk size for the virtualmachine.

Select the storage size as Dynamically allocated memory size and click OK.The

virtual machine will be created.

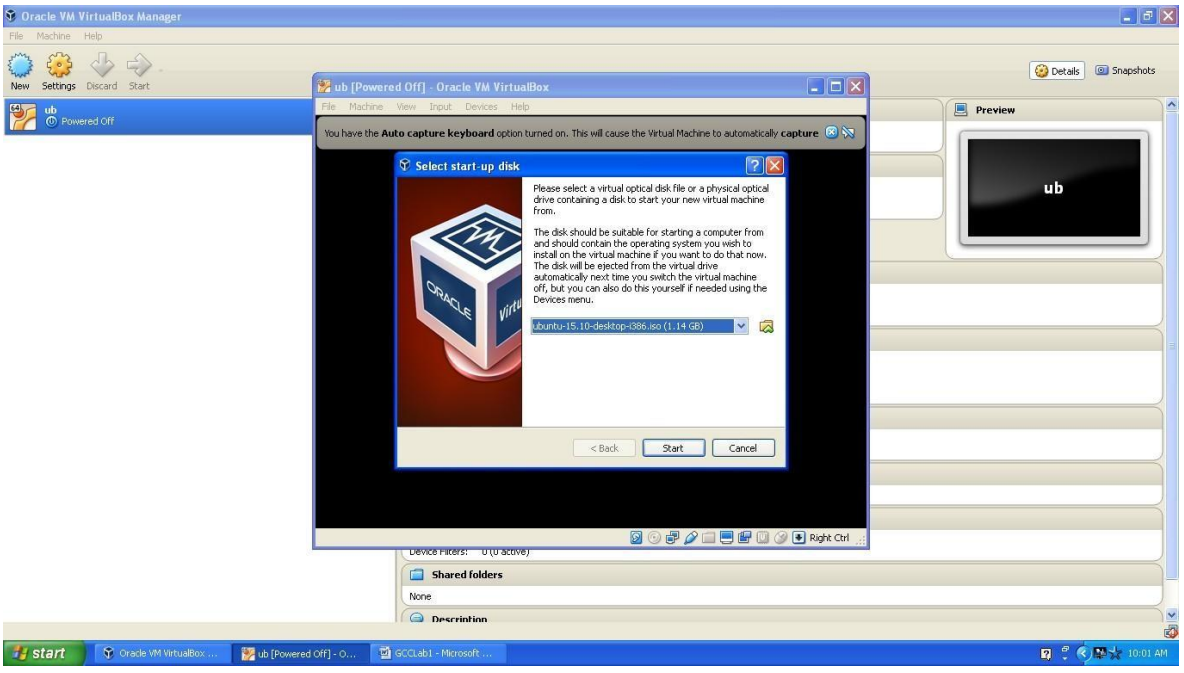

Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
Details Snapshots
Welcome to VirtualBox!
The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.
In order to create a new virtual machine, press the New button in the main tool bar located at the top of the window.
Create Virtual Machine
Name and operating system
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.
Name: ub
Type: Linux
Version: Ubuntu (64-bit)
Expert Mode
< Back
Next >
Cancel
start
9:56 AM


Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Start
Details Snapshots
Welcome to VirtualBox!
The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.
Create Virtual Hard Disk
File location and size
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.
ub
Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.
1.00 GB
4.00 MB
2.00 TB
< Back
Create
Cancel
start
9:57 AM



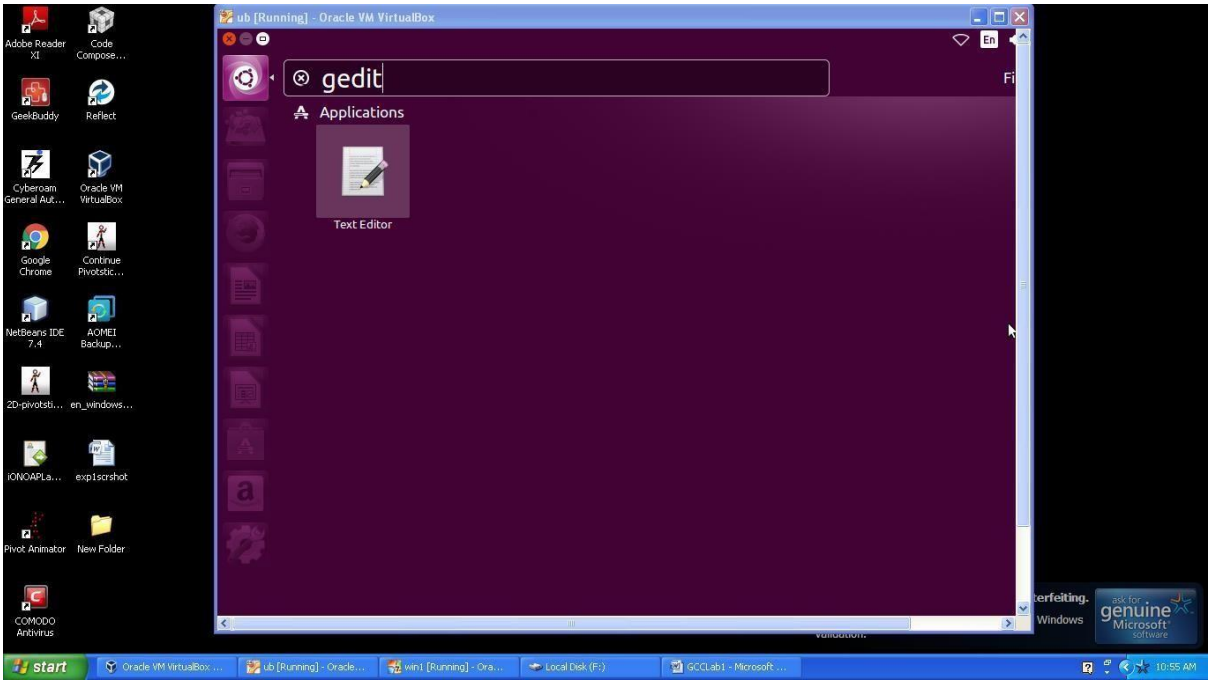
Step 3:

Select the iso file of the virtual OS Ubuntu and click Start.



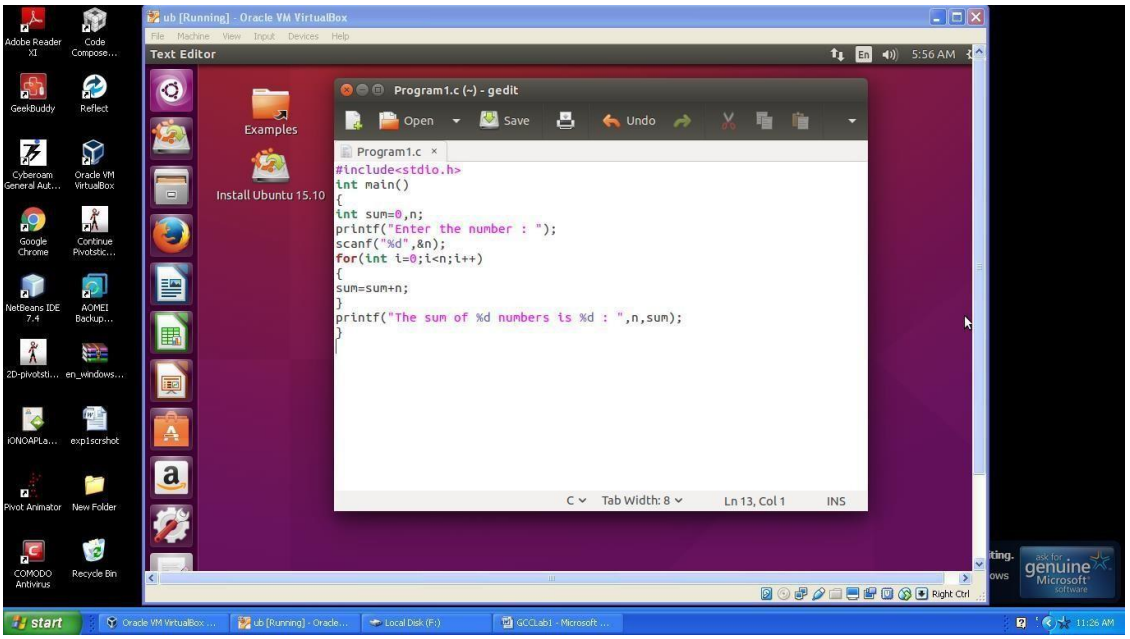
Step 4:

The virtual OS Ubuntu is opened successfully. Now type “gedit” in the search box to open text editor in Ubuntu.



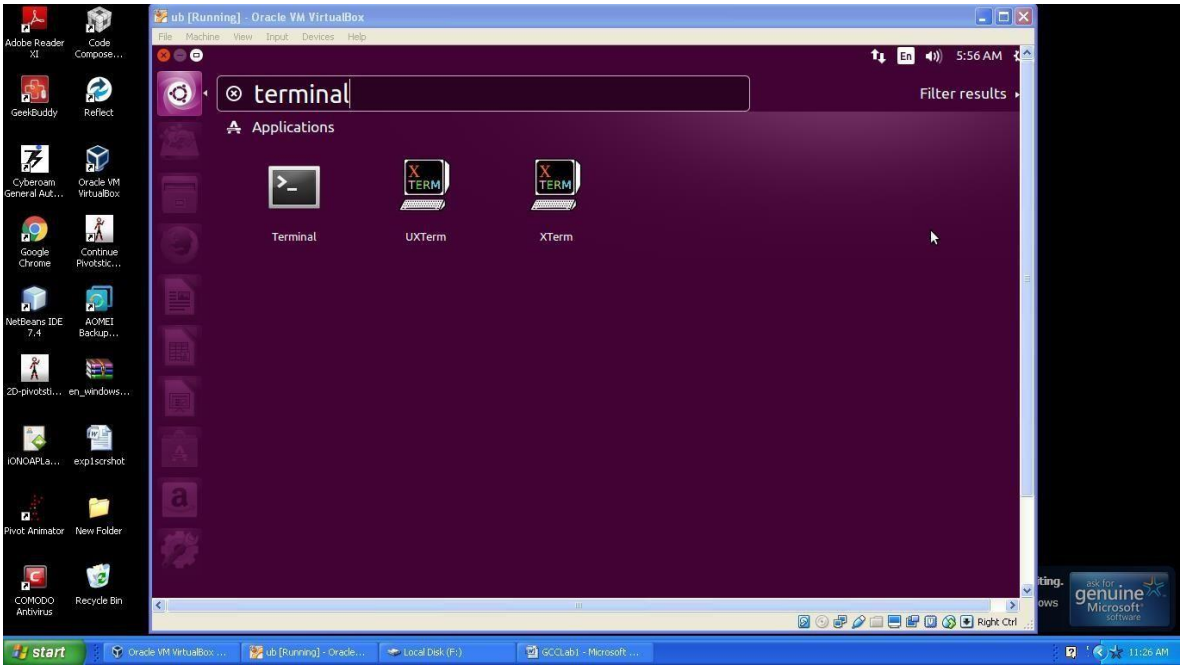
Step 5:

Type your desired C program in text editor and save it with the extension (.c).



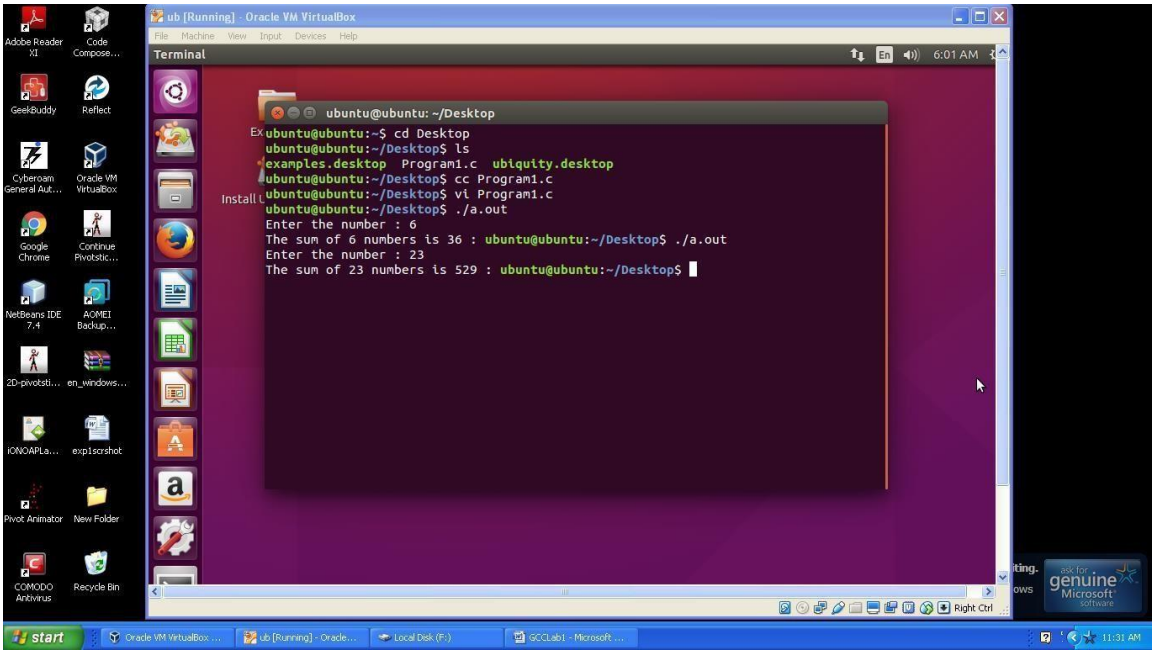
Step 6:

Type “terminal” in the search box to open the command window.



Step 7:

Type the necessary commands to compile and run the C program.

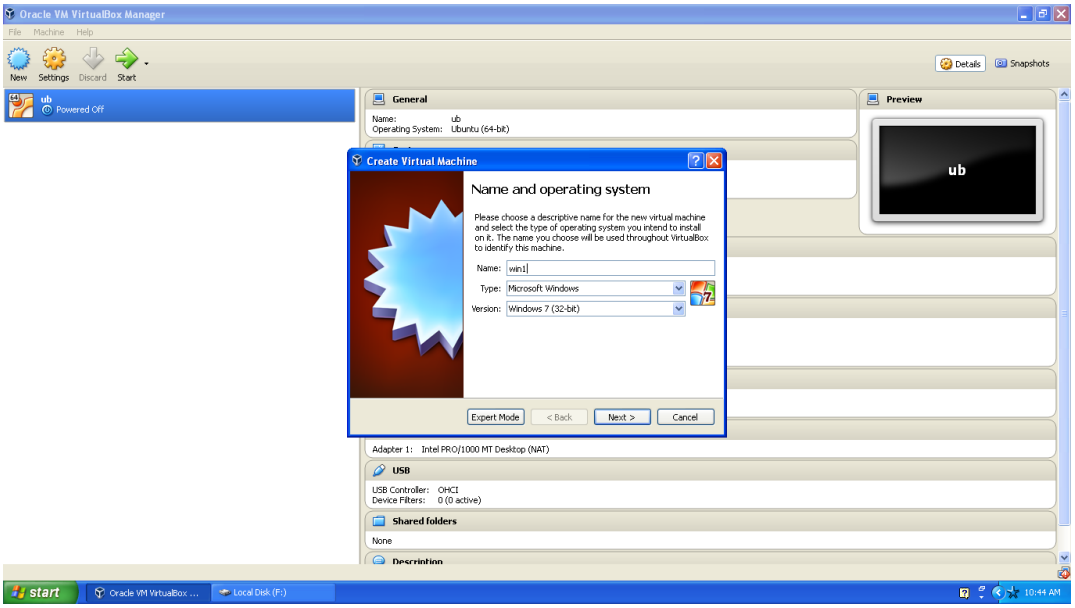


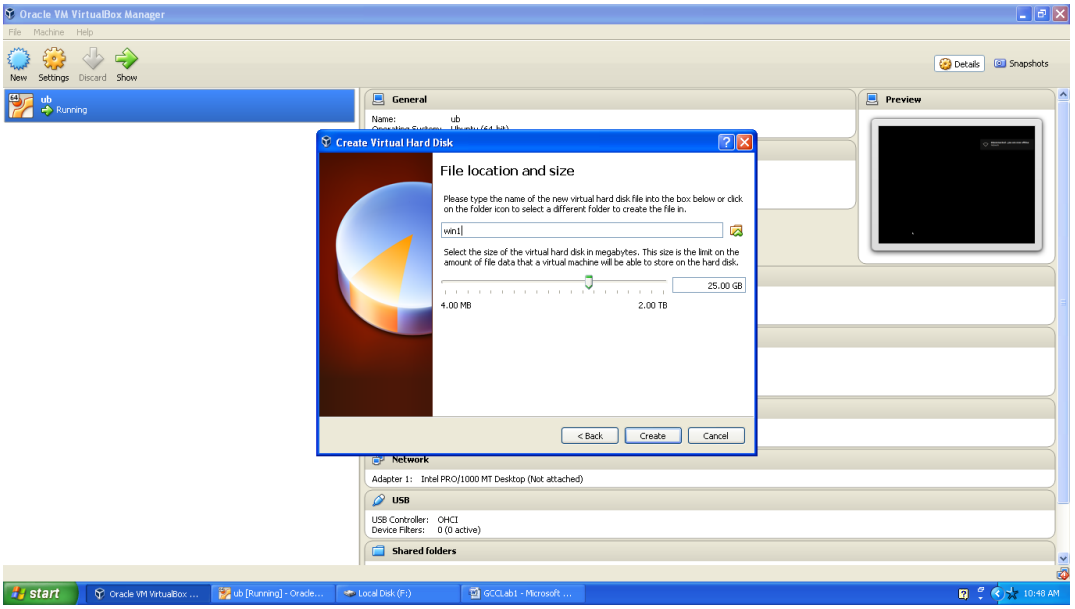
Installing Windows 7 using Oracle Virtual Box

PROCEDURE:

Step1:

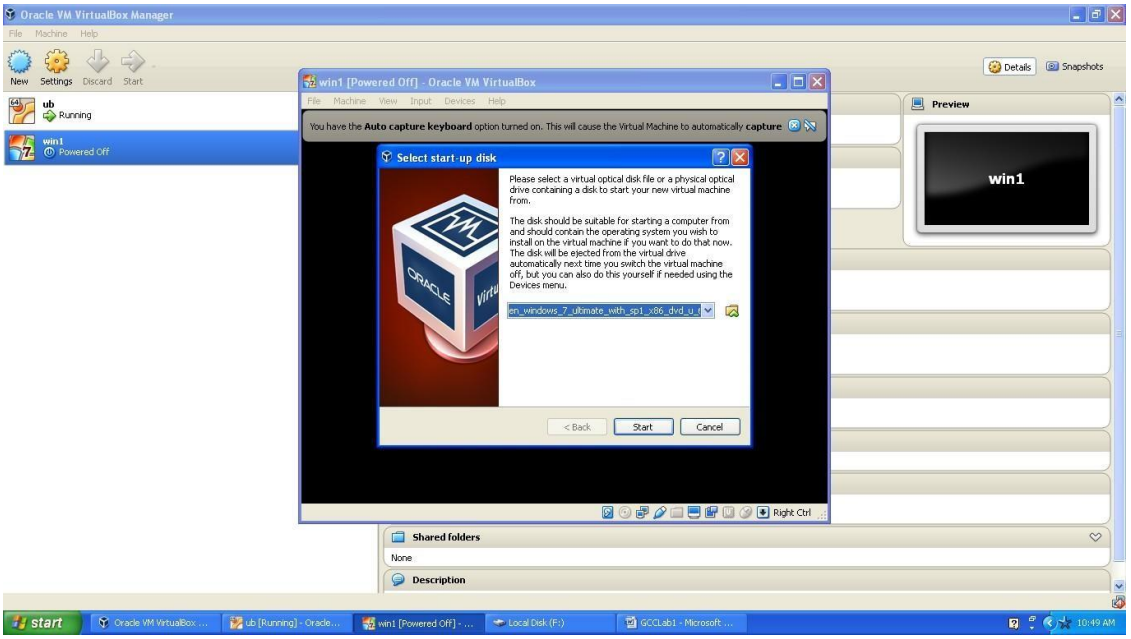
Open Oracle virtual box manager and click create new -> virtual machine. Provide and name for the operating system and select the memory size to be occupied in memory.





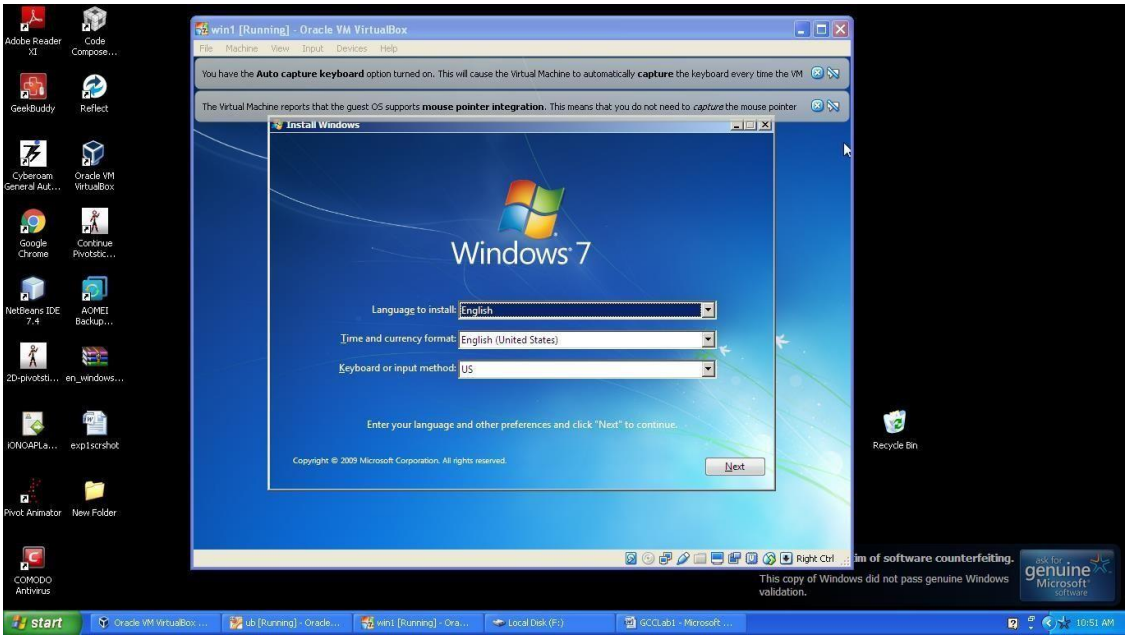
Step 2:

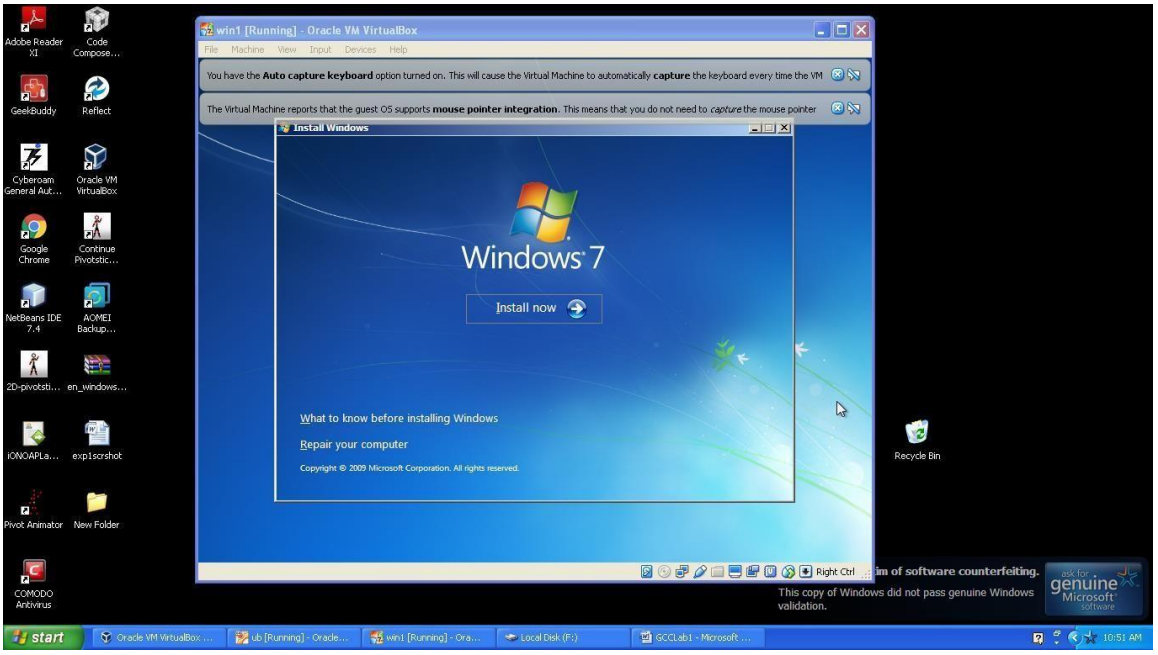
Select the iso file of the virtual OS Windows7 and click Start.



Step 3:

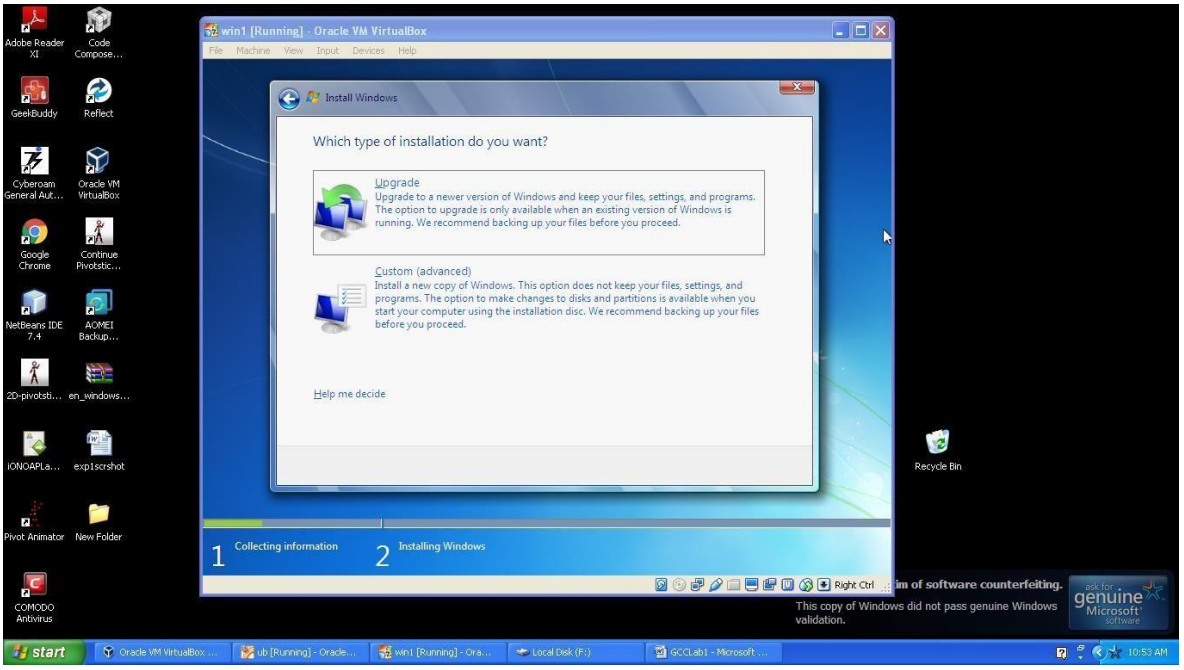
Select the language to use in the Operating System and click Install Now.

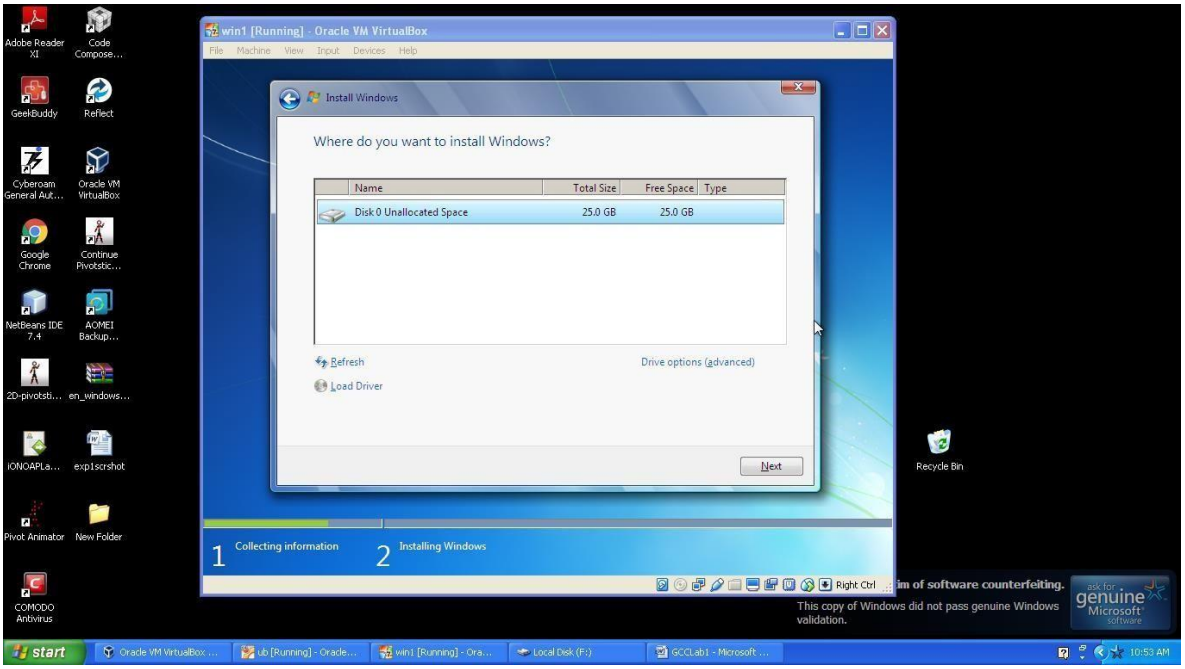




Step 4:

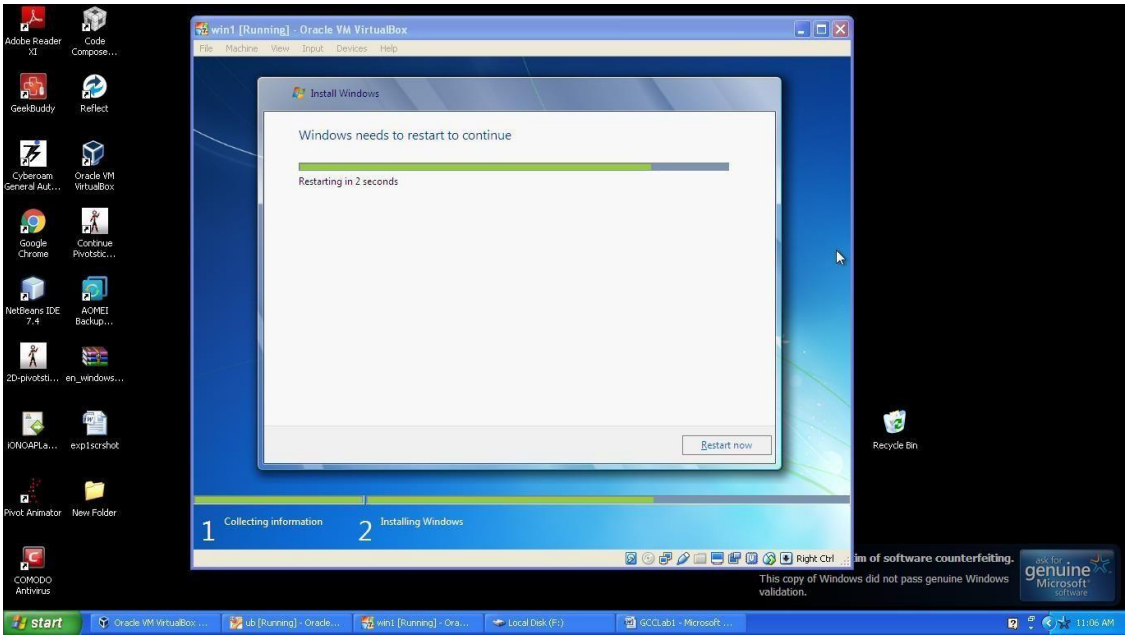
Select the type of installation as Custom for new installation and allocate Diskspace according to your convenience. Click Next to start the installation.

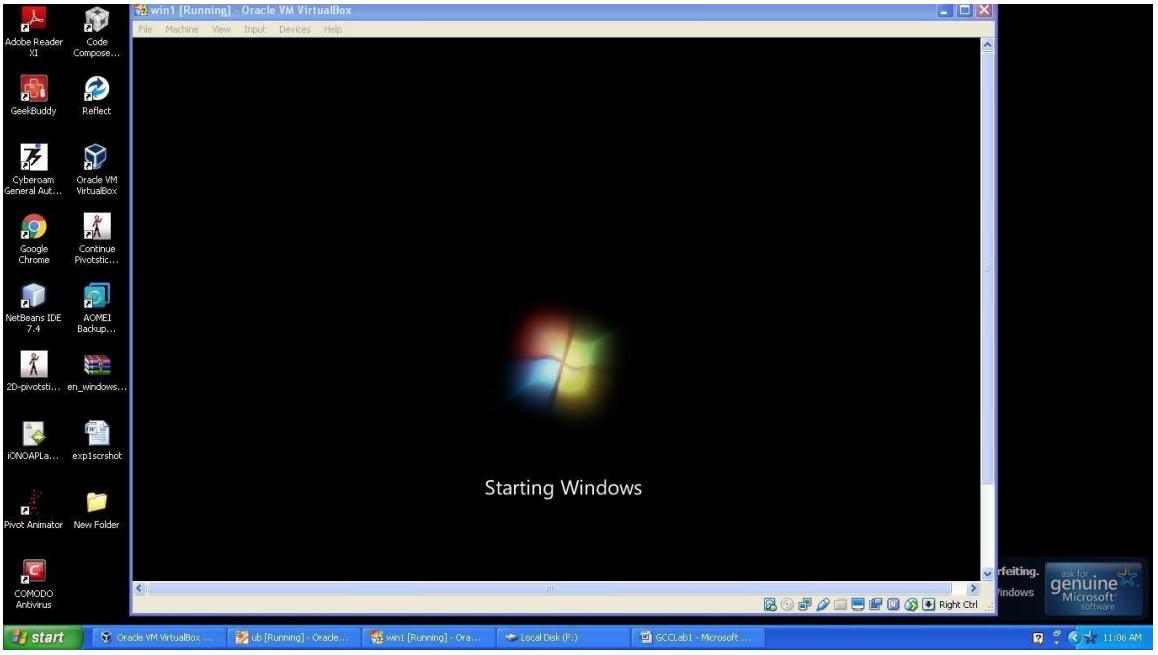


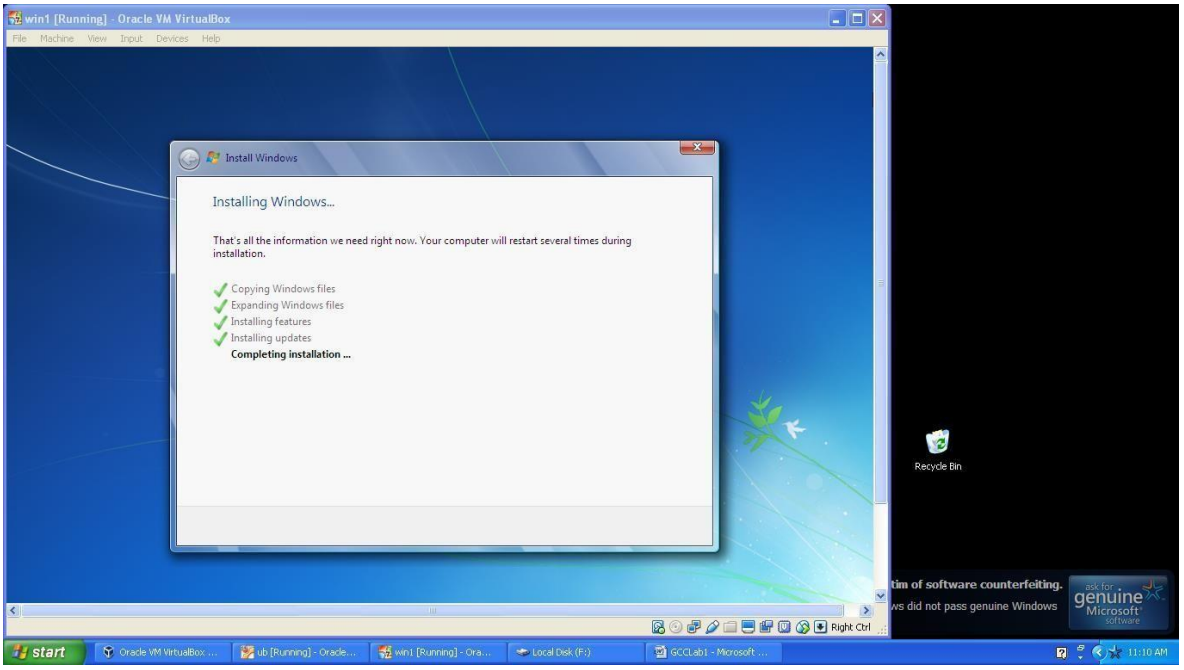


Step 5:

After installation the system will be restarted to complete the installation.

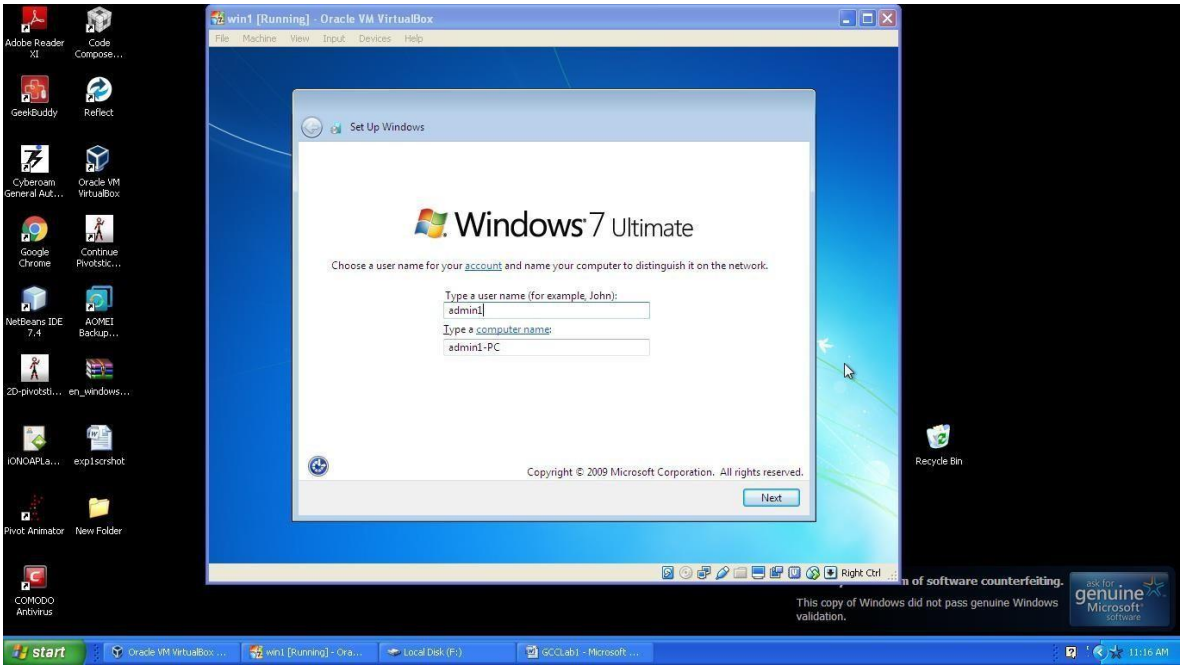






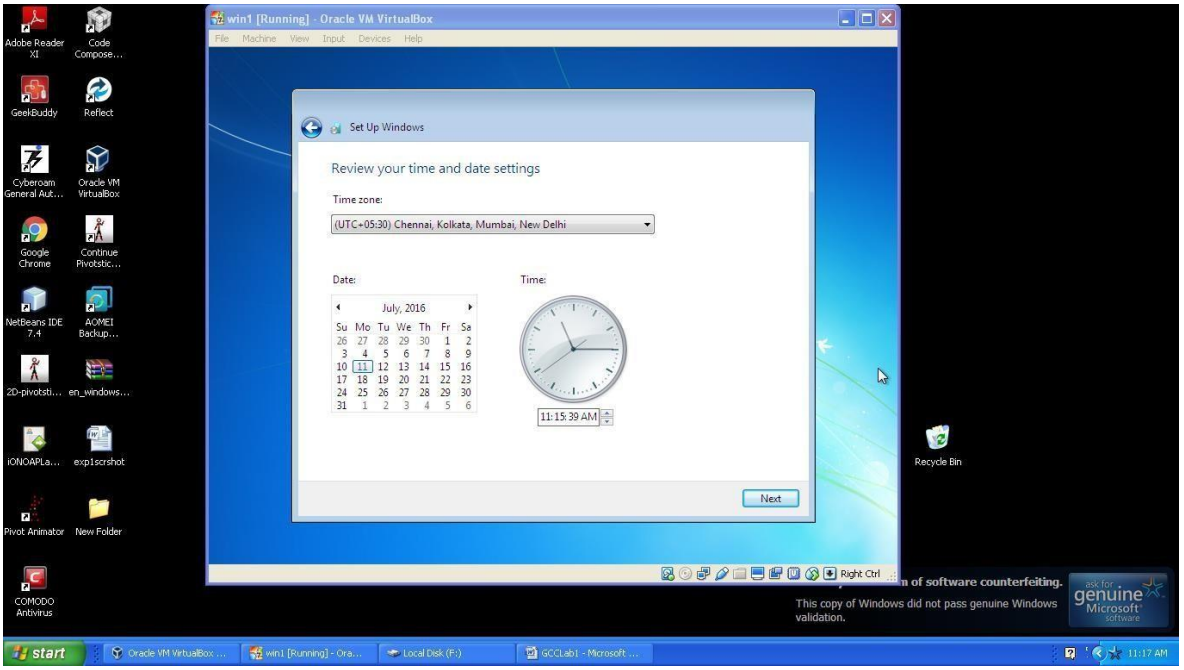
Step 6:

Provide a user name and password(optional) to gain access over the OS.



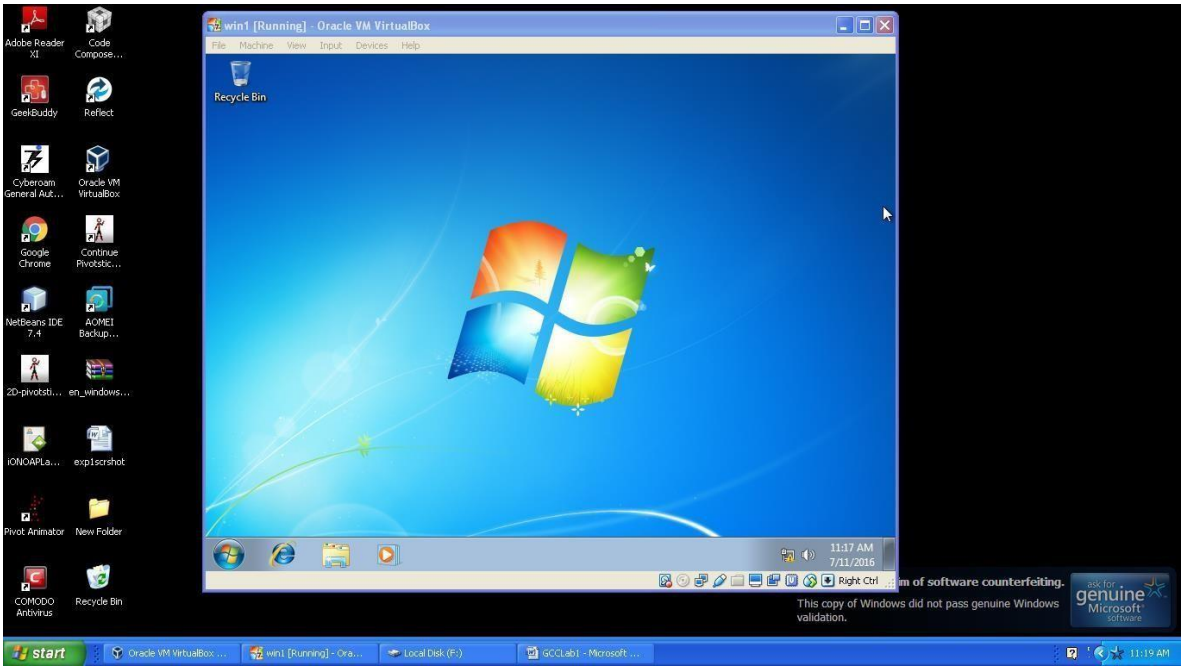
Step 7:

Set the time and date for the new Operating System.



Step 8:

Thus the new Operating System Windows7 will be opened as the virtual machine.



RESULT:

Thus the procedure to run different virtual machines on a single system using Oracle Virtual Box is studied and implemented successfully.

Ex.No.2	**Install a C compiler in the virtual machine created**
Date:	**using virtual box andexecute Simple Programs.**

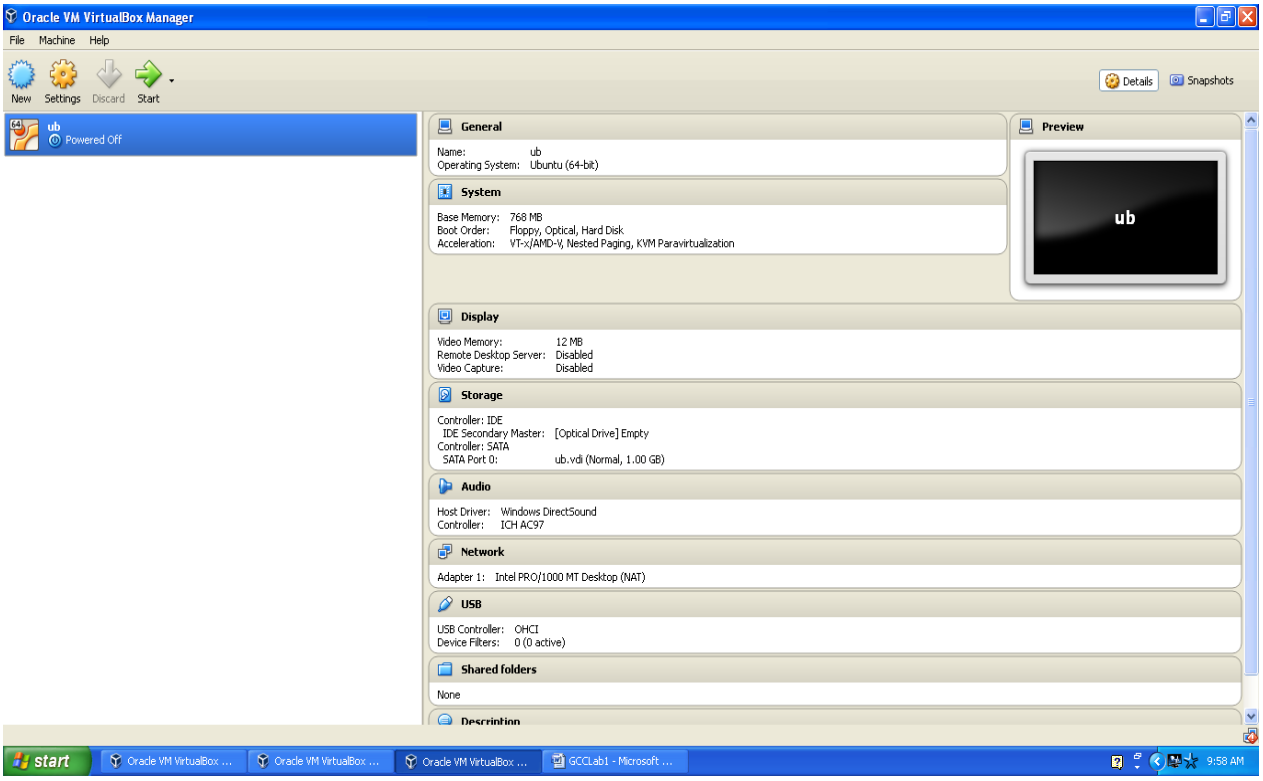
AIM:

To find a procedure to use the C compiler in the virtual machine and execute asample program.

PROCEDURE:

Step 1:

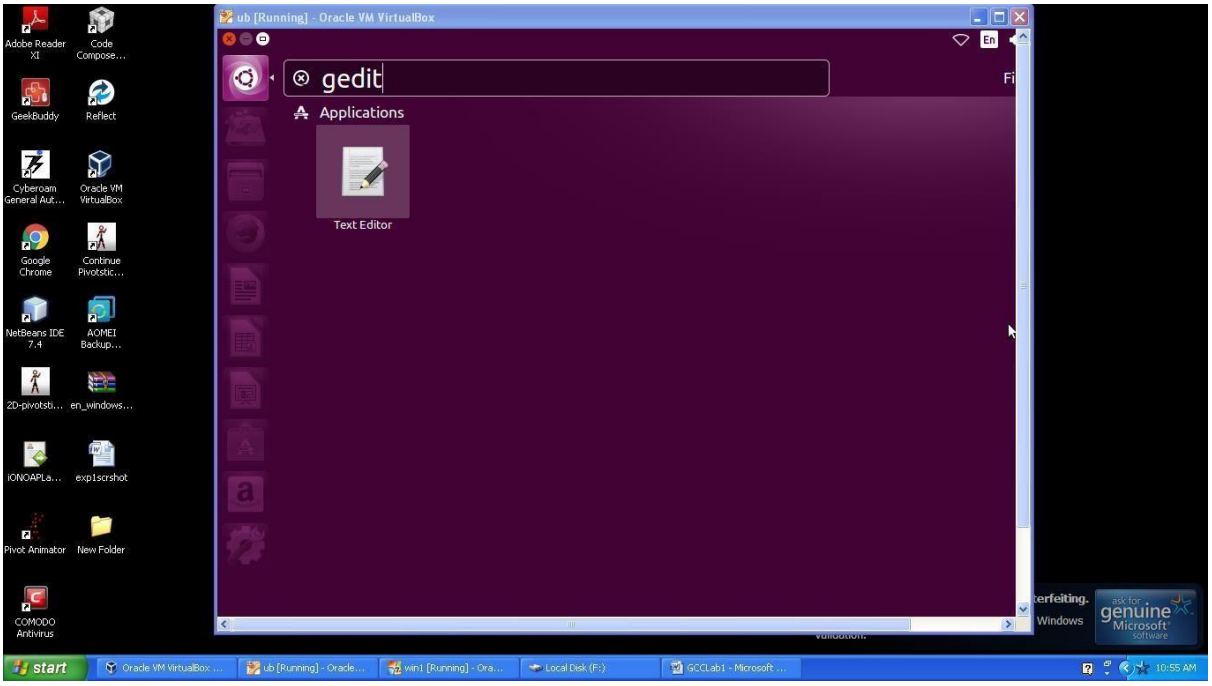
Open the virtual machine in which you want to run the C program.



Step 2:

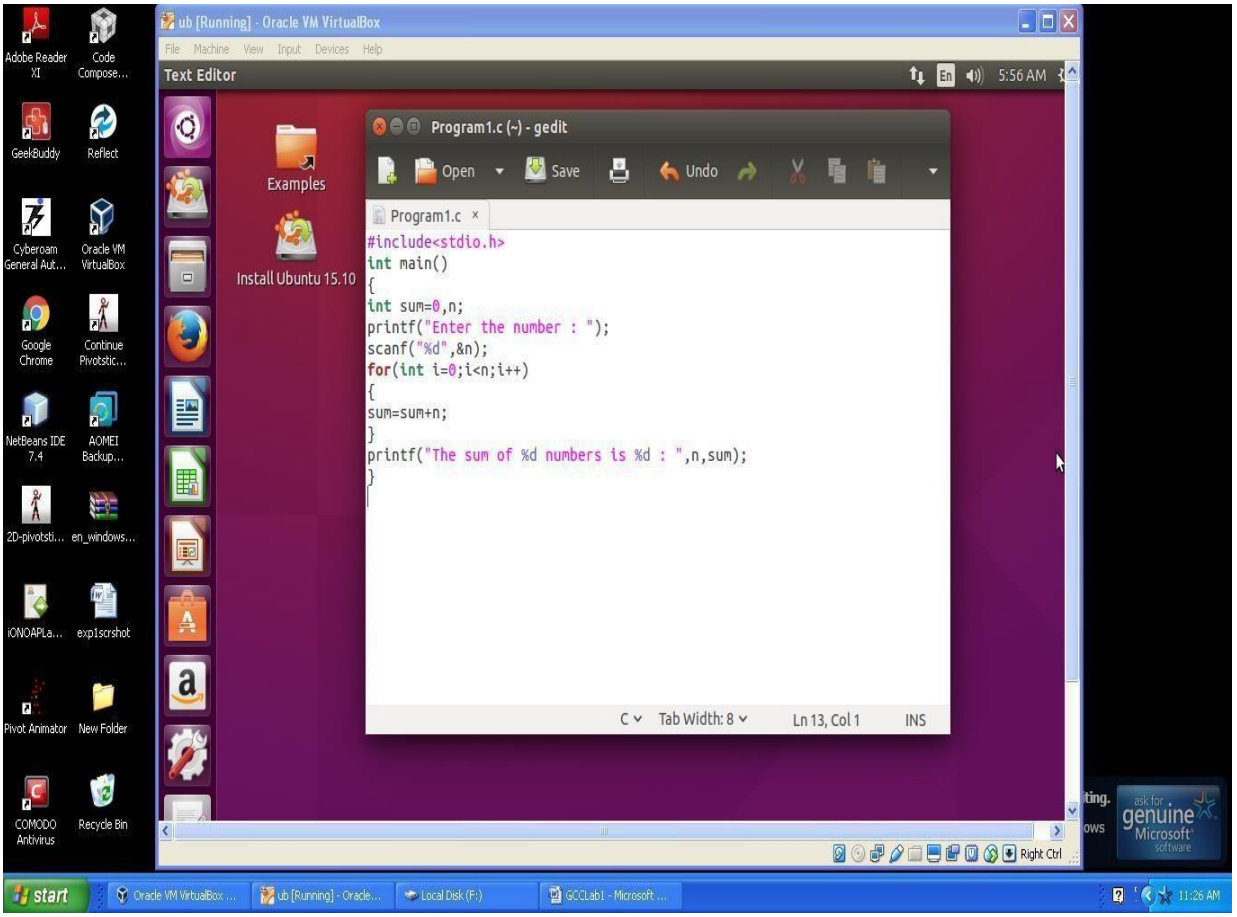
The text editor used by the Ubuntu Operating System is the GEDIT.It can be opened by using the search option by typing gedit in it.

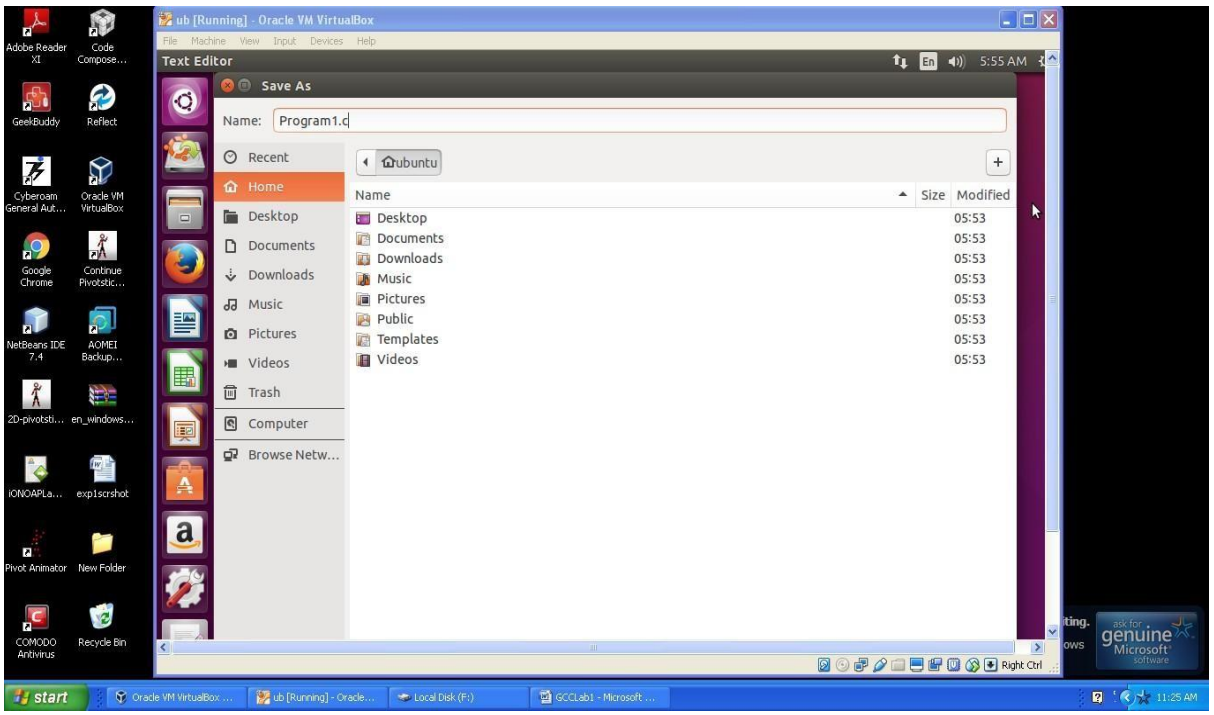
The text editor will be opened now.



Step 3:

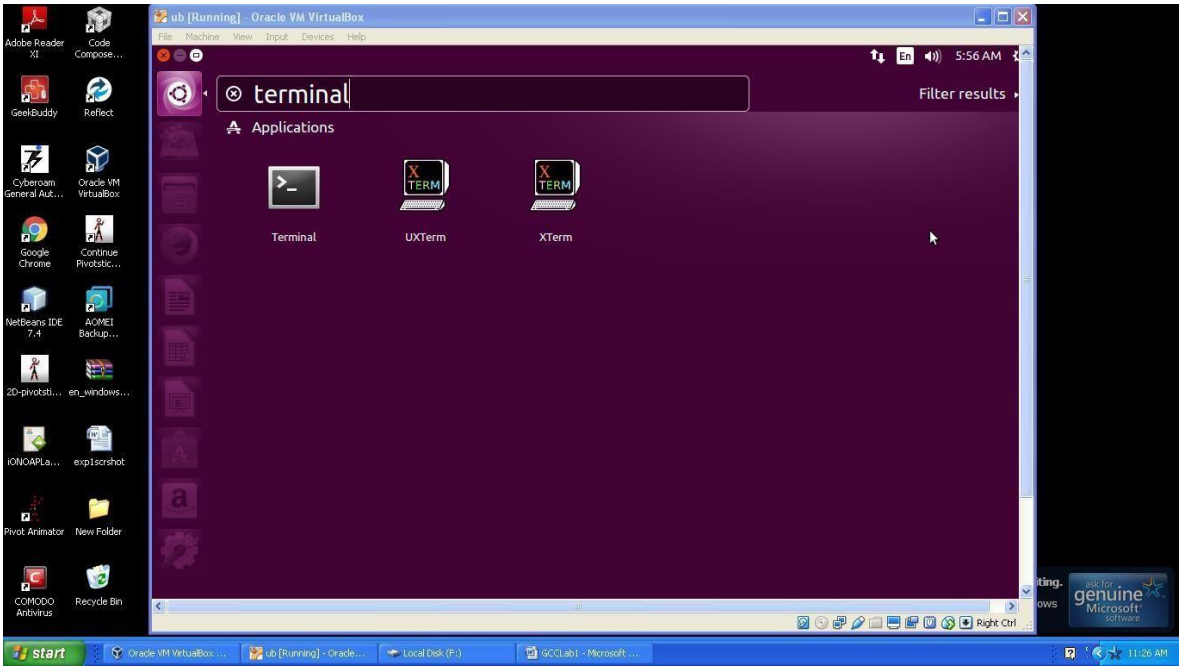
Type your desired C program in the text editor and save it as a C file using theExtension (.c) for C programs.





Step 4:

Type “terminal” in the search box to open the command window.

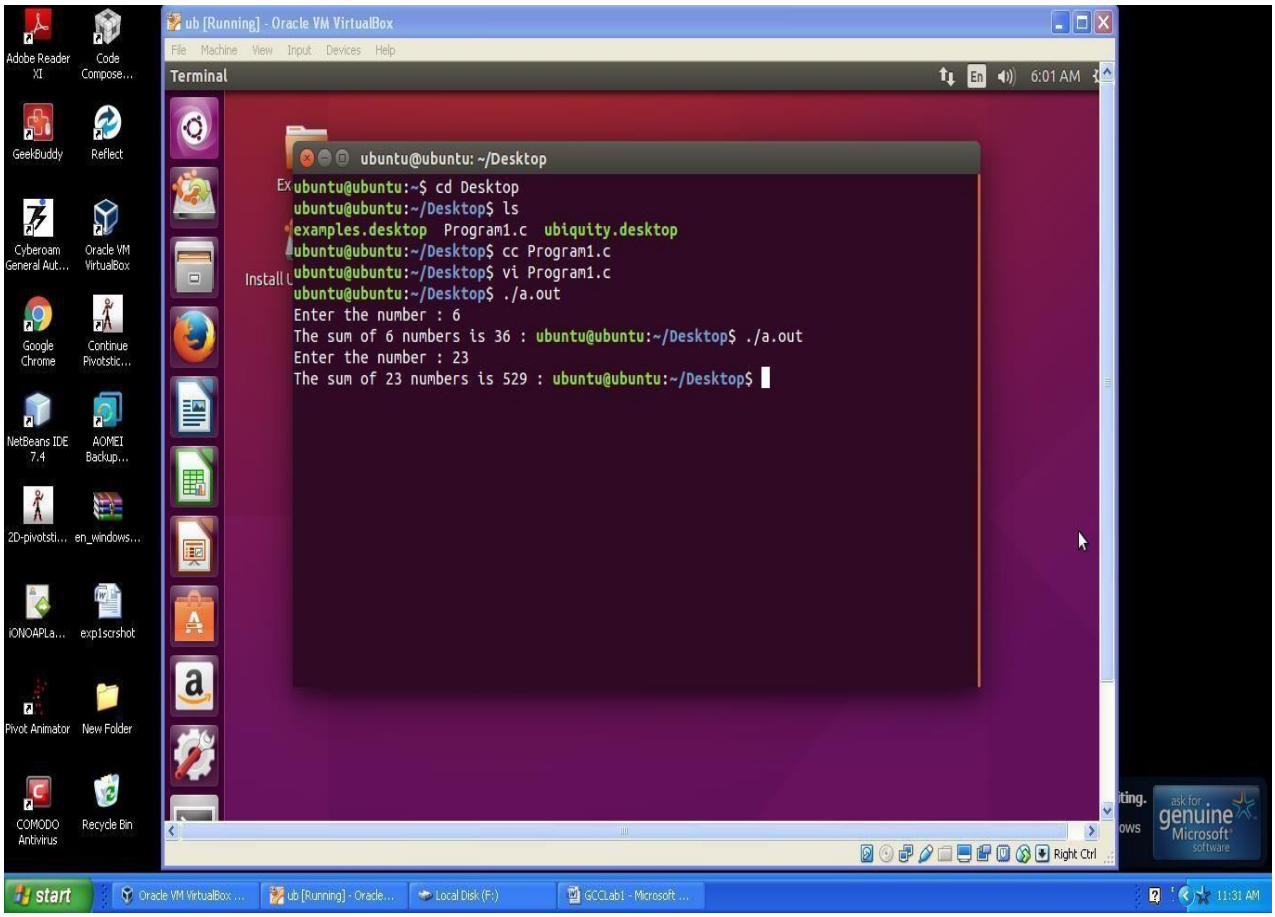


Step 5:

Type the necessary commands to compile and run the C program.

(1).cc_filename to compile the C program.

(2). ./a.out to display the output of the last compiled program.



RESULT:

Thus the procedure to use the C Compiler in the virtual machine and execute a sample program is implemented sucessfully.

Ex.No.**3** **Installing and Running the Google App Engine**

Date: **On Windows**

AIM:

To Installing and Running the Google App Engine On Windows

PROCEDURE:

PROCEDURE:

Install the latest Cloud SDK version (303.0.0)

Step 1: Download the Cloud SDK installer.

```
(New-Object
```

Alternatively, open a PowerShell terminal and run the following PowerShell commands.

```
Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/cha
nnels/ra
pid/GoogleCloudSDKInstaller.exe", "$env:Temp\GoogleCloudSDKInstaller.exe")

& $env:Temp\GoogleCloudSDKInstaller.exe
```

Step 2: Launch the installer and follow the prompts. The installer is signed by Google LLC. Cloud SDK requires Python. Supported versions are 3.5 to 3.7, and 2.7.9 or higher.

The installer will install all necessary dependencies, including the needed Python version. While Cloud SDK currently uses Python 2 by default, you can use an existing Python installation if necessary by unchecking the option to 'Install Bundled Python'. .

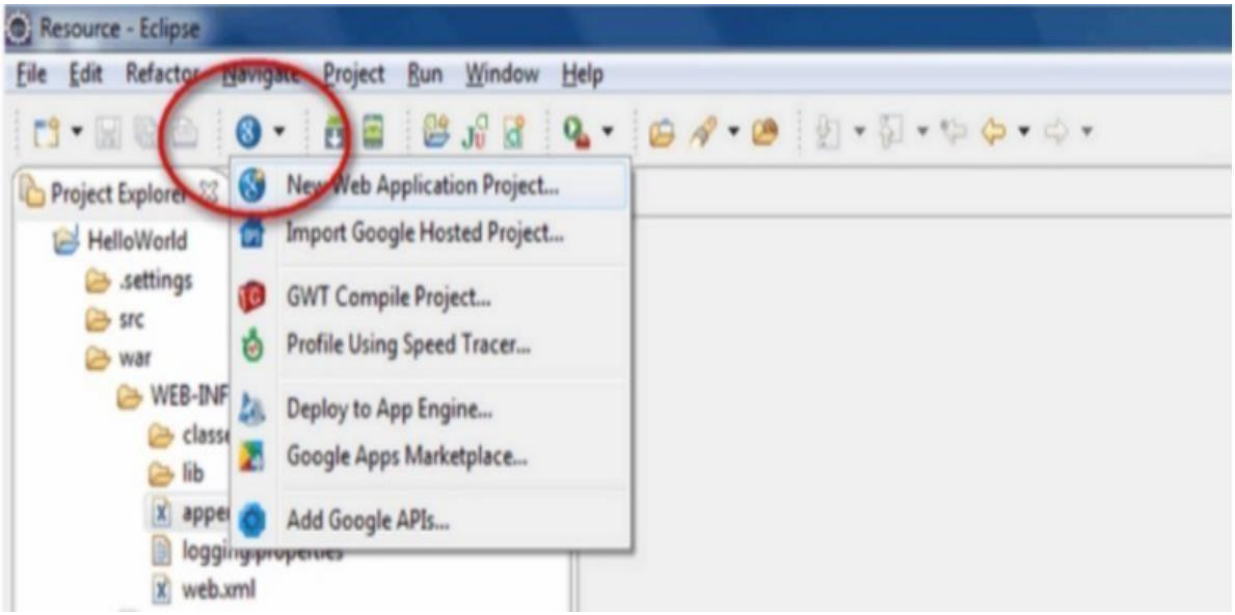
Note: Cloud SDK uses Python 2 by default, but will soon move to Python 3 (run gcloud topic startup for exclusions and more information on configuring your Python interpreter to use a different version). Consider upgrading to Python 3 to avoid disruption in the future.

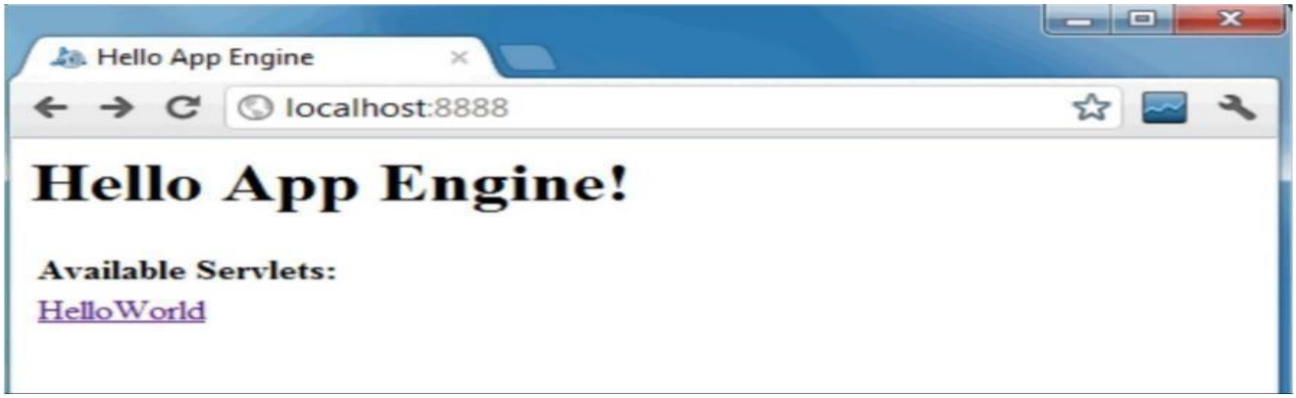
Step 3: After installation has completed, accept the following options:

Start Cloud SDK Shell

Run gcloud init

OUTPUT:


Resource - Eclipse
File Edit Refactor Navigate Project Run Window Help
Project Explorer
HelloWorld
.settings
src
war
WEB-INF
lib
web.xml
New Web Application Project...
Import Google Hosted Project...
GWT Compile Project...
Profile Using Speed Tracer...
Deploy to App Engine...
Google Apps Marketplace...
Add Google APIs...


Hello App Engine
localhost:8888
Hello App Engine!
Available Servlets:
HelloWorld


localhost:8888/helloworld
localhost:8888/helloworld
Hello, world

The installer starts a terminal window and runs the gcloud init command.

Step 4: The default installation does not include the App Engine extensions required to deploy an application using gcloud commands. These components can be installed using the Cloud SDK component manager.

Optional: Enable accessibility features

For a more streamlined screen reader experience, the gcloud command-line tool comes with an accessibility/screen_reader property.

```
gcloud config set accessibility/screen_reader true
```

Percentage progress bars: Progress will be displayed as a percentage, outputted to stderr.

Boxed tables drawn with ASCII characters: Boxed tables are the default output of many list commands. Instead of being drawn with Unicode, they will be rendered using ascii characters. Also, consider using the *--format* flag to define your own format.

To enable these accessibility features, run:

```
$ gcloud config set accessibility/screen_reader true
```

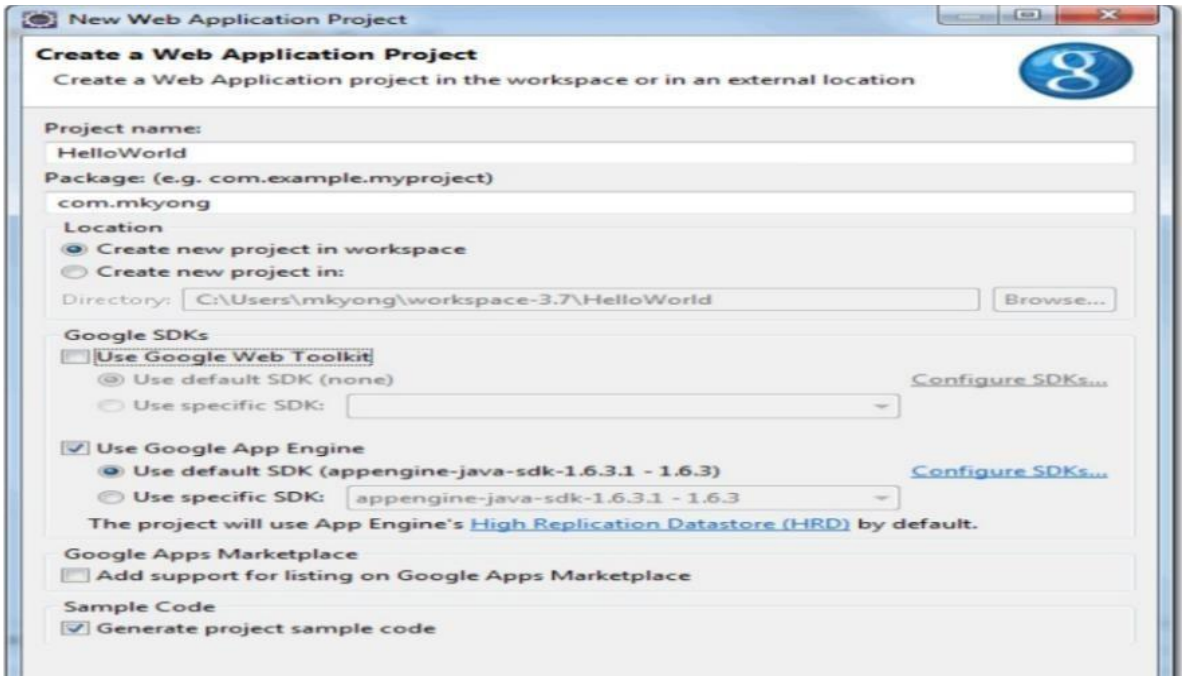
Additional resources

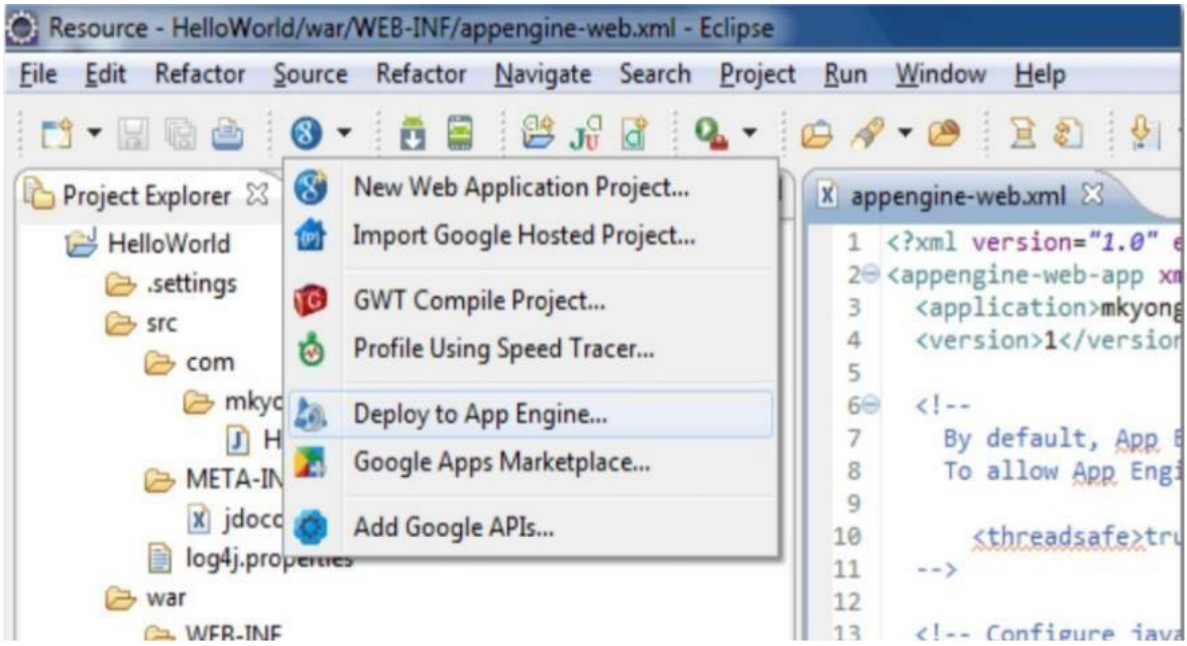
Alternate documentation experiences

- To search the gcloud CLI reference docs for a keyword or command, use *gcloud help*. This runs a search for all commands with help text matching the given argument or arguments.

Refining gcloud CLI output

- To adjust the verbosity level of a command, you can set the gcloud CLI-wide flag --verbosity with either debug, info, warning, error, critical, or none.
- For list commands, you can further refine your output by using the --limit flag to set a maximum number of resources to list. You can also use the --page-size flag to define the number of resources per page if the service lists output in pages. To sort, use the --sort-by flag with the relevant field to sort.
- To disable interactive prompting, use the --quiet flag.


New Web Application Project
Create a Web Application Project
Create a Web Application project in the workspace or in an external location
Project name:
HelloWorld
Package: (e.g. com.example.myproject)
com.mkyong
Location
Create new project in workspace
Create new project in:
Directory:
C:\Users\mkyong\workspace-3.7\HelloWorld
Browse...
Google SDKs
Use Google Web Toolkit
Use default SDK (none)
Configure SDKs...
Use specific SDK:
Use Google App Engine
Use default SDK (appengine-java-sdk-1.6.3.1 - 1.6.3)
Configure SDKs...
Use specific SDK:
appengine-java-sdk-1.6.3.1 - 1.6.3
The project will use App Engine's High Replication Datastore (HRD) by default.
Google Apps Marketplace
Add support for listing on Google Apps Marketplace
Sample Code
Generate project sample code


Resource - HelloWorld/war/WEB-INF/appengine-web.xml - Eclipse
File Edit Refactor Source Refactor Navigate Search Project Run Window Help
Project Explorer
HelloWorld
.settings
src
com
META-IN
war
WEB-INF
New Web Application Project...
Import Google Hosted Project...
GWT Compile Project...
Profile Using Speed Tracer...
Deploy to App Engine...
Google Apps Marketplace...
Add Google APIs...
appengine-web.xml
<?xml version="1.0"
<appengine-web-app
<application>mkyong
<version>1</versio
<!--
By default, App
To allow App Engi
<threadsafe>tru
-->
<!-- Configure java

- To structure and produce more meaningful output, you can use the format, filter and projection flags to fine-tune your output.
    - If you'd like to define just the format of your output, use the `--format` flag to produce a tabulated or flattened version of your output (for interactive display) or a machine-readable version of the output (json, csv, yaml, value).
    - To format a list of keys that select resource data values, use projections.
    - To further refine your output to a criteria you'd like to define, use the `--filter` flag.

Filing feedback

To file a bug, provide feedback, or send suggestions, use gcloud feedback to help improve the gcloud command-line tool experience. This command will open the public issue tracker in a browser window.

You can also directly log issues using issue tracker. Be sure to toggle the 'Type' field to reflect the nature of the issue you're filing (Bug, Feature Request, etc.) to help make responses more efficient.

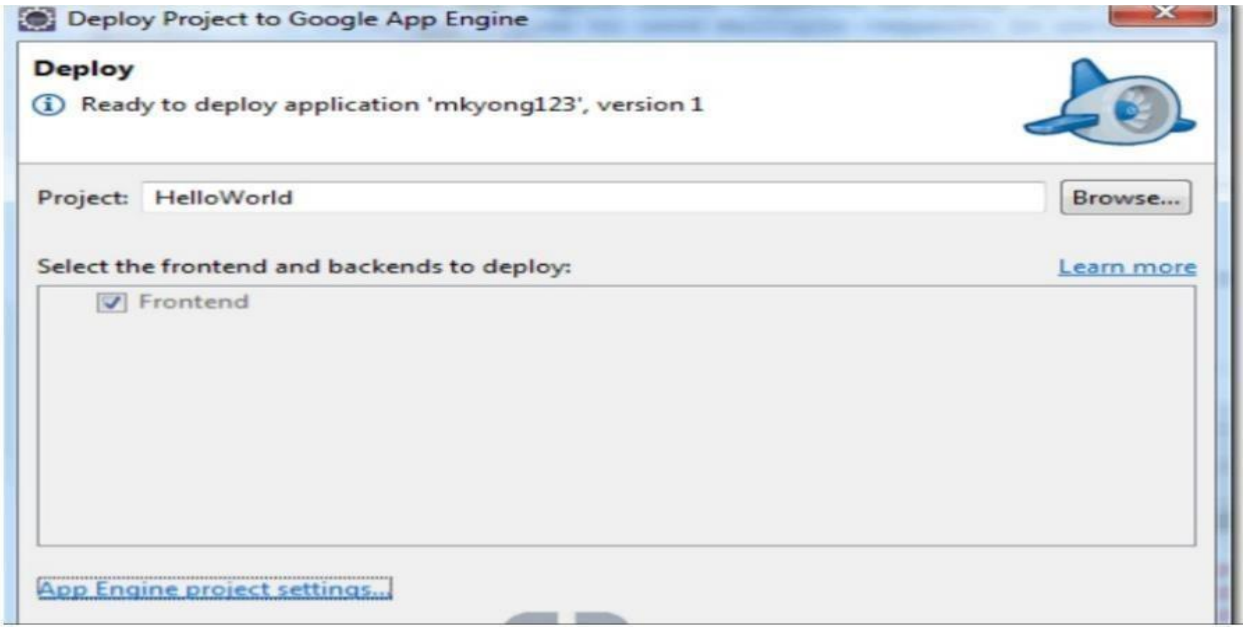
To enable this property, run:

- Download Google Cloud SDKhttps://cloud.google.com/sdk/docs
- Login to google account and verify the account with google for the successful authentication.
- Download webapp2 module from https://pypi.org/project/webapp2/
- Write down simple hello world python program.

```
import webapp2
class MainPage(webapp2.RequestHandler)
def get(self):
   self.response.write("Hello World")

app = webapp2.WSGIApplications([('/',MainPage)], debug=True)
```

- Create app.yaml which has url link



- runtime: python3 api_version: 1
  threadsafe: true

  handlers:
  -url: /
  script: test.app

- Run the gcloud prompt and give Yes option to proceed further.
- Install gcloud components using the command
  gcloud components install cloud-datastore-emulator

RESULT:

Thus the Google App Engine insatlled on WIndows successfully and *hello World* app and other simple web application using python / java executed sucessfully.

Ex.No : 4 **Google App Engine Launch the Web Applications**

Date :

Aim:

To Use GAE launcher to launch the web applications

Procedure:

Setting Up Your Development Environment

To set up your environment for developing on Python 3:

1. Download and install the latest release of Python 3.

   See Python 3 Runtime Environment for a list of the supported versions.

2. Download, install, and then initialize the Cloud SDK. If you already have the Cloud SDK installed, run the `gcloud components update` command to update to the latest release.

   The Cloud SDK provides you the `gcloud` command-line tooling for deploying and managing your apps.

   By downloading, you agree to be bound by the Terms that govern use of the Cloud SDK for App Engine.

   Download and Install the Cloud SDK

Note: Avoid using a package manager such as apt or yum to install the Cloud SDK.

Installing optional tools

For access to code, samples, libraries, and tools in GitHub, install Git:

Download and install Git

Install your preferred tooling or framework, for example you can use any of the following frameworks to develop your Python 3 app:

- Flask

- Django
- Pyramid
- Bottle
- web.py
- Tornado

Creating a website to host on Google App Engine

Basic structure for the project

This guide uses the following structure for the project:

- app.yaml: Configure the settings of your App Engine application.
- www/: Directory to store all of your static files, such as HTML, CSS, images, and JavaScript.
    - css/: Directory to store stylesheets.
        - style.css: Basic stylesheet that formats the look and feel of your site.
    - images/: Optional directory to store images.
    - index.html: An HTML file that displays content for your website.
    - js/: Optional directory to store JavaScript files.
    - Other asset directories.

Creating the app.yaml file

The app.yaml file is a configuration file that tells App Engine how to map URLs to your static files. In the following steps, you will add handlers that will load www/index.html when someone visits your website, and all static files will be stored in and called from the www directory.

Create the app.yaml file in your application's root directory:

1. Create a directory that has the same name as your project ID. You can find your project ID in the Console.

2. In directory that you just created, create a file named app.yaml.

3. Edit the app.yaml file and add the following code to the file:

Creating the *index.html* file

Create an HTML file that will be served when someone navigates to the root page of your website. Store this file in your `www` directory.

```
<html>
  <head>
    <title>Hello, world!</title>
    <linkrel="stylesheet"type="text/css"href="/css/style.css">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <p>
      This is a simple static HTML file that will be served from Google
App
      Engine.
    </p>
  </body>
</html>
```

Deploying your application to App Engine

When you deploy your application files, your website will be uploaded to App Engine. To deploy your app, run the following command from within the root directory of your application where the `app.yaml` file is located:

```
gcloud app deploy
```

Optional flags:

- Include the --project flag to specify an alternate Cloud Console project ID to what you initialized as the default in the gcloud tool. Example: --project [YOUR_PROJECT_ID]
- Include the -v flag to specify a version ID, otherwise one is generated for you. Example: -v [YOUR_VERSION_ID]

To learn more about deploying your app from the command line, see Deploying a Python 2 App.

viewing your application

To launch your browser and view the app at `https://PROJECT_ID.REGION_ID.r.appspot.com`, run the following command:

```
gcloud app browse
```

Step 1 : Download the basic housekeeping stuff

No matter what platform you build products on, there is always some housekeeping stuff you need to put in place before you can hit the ground running. And deploying apps within the Google App Engine is no exception.

Download Python 2.7

As of when this article was written, the Google App Engine standard environment supports Python only upto version 2.7. However, it is only a matter of time before support for Python 3.x is added. You can check the App Engine docs for the latest info.

Download Google Cloud SDK

This will allow you to fork apps onto your local machine, make changes (edit and develop the app), and deploy your app back to the cloud.

Set the Python path in the Google App Engine launcher

After downloading the SDK, launch the App Engine launcher, go to Edit -> Preferences and make sure you set the path for where you installed Python in step 1 above.

Set the Python path in Google App Engine launcher That's all you need. Your local machine should now be ready to build webapps.

Step 2 : App Engine sign-up

This is often the most confusing part of the entire setup. Things you should know when you sign-up:

1. Currently, App Engine offers a free trial for one year.
2. The trial includes $300 of credit that can be used during the one year trial period.

3. You will need to add a credit card to sign-up (for verification purposes).
4. You will not be charged during the sign-up process.
5. You will not be charged during the trial period as long as you do not cross the credit limit offered.

Here are the steps you need to follow to sign-up:

1. Go to the Google Cloud landing page
2. Follow the sign-up process and go to your App Engine dashboard

Most of the hard work is complete after a successful sign-up.

Step 3 : Create a new project

The next step is to create a new Python project that you can work on. Follow the screenshots below to create a new project.

Launch the new project wizard.

Give your app a name and make a note of your project ID.

Hit the create button and Google should take a few minutes to set up all that is necessary for your newly created app.

Step 4 : Fork the app to develop it locally

The next step in the process is to fork the app on your local machine. This will allow you to make changes to the app locally and deploy it whenever you wish to.

Go to Google App Engine launcher and create a new application.

Enter the project ID of your newly created app. Also, provide the folder (local destination) where you wish to store the app locally. Make sure you select the Python 2.7 as your runtime engine.

Hit the create button, and you should see your app listed on the window that follows. You should also check that you now see some files in your local storage (the directory you chose in the screenshot above) after this step.

Step 5 : Run the app locally

Before you go ahead and make some changes to the app, it is important to check whether or not you have executed all the above steps correctly. This can be done by simply running the

app locally.

Select the app and hit the run button on the window.

Wait for a few seconds until you can hit the Browse button. Once the Browse button becomes clickable, click it. This should take you to the browser, and you should see the hello world text appear in your browser window. Alternatively, you can manually go to the browser and use the port specified to access the app.

As long as you see the above screen, you are all set.

Step 6 : Understand the app structure

It is finally time to look at the lines of code which are running this webapp. Open your app folder in the text editor of your choice. I recommend Sublime text or VS Code. However, feel free to choose the one you prefer.

Here is a description of the various files.

app.yaml

This file is a basic markup file that stores information (some metadata) about the app. It is important to note the following crucial parts of the file.

1. application

This is the project ID which you should never change. This is the unique identifier for the app

2. url -> script

This is the homepage for the app. In other words, this file will be rendered in your browser when you launch the app

3. libraries

This is where you can include external libraries to use within the webapp

main.py

This is the homepage of the app (as discussed above). Note that the hello world text in the browser window (step 5) is due to the code you see highlighted below.

Step 7 : Make your changes and deploy the new app

No hello world app is ever complete without the developer changing the hello world text

to something else just to make sure that everything happening behind the scenes is working as it should.

Go ahead and change the text in the above screenshot to something else.
Save the changes, go to the browser and refresh the page. You should see the page with the text "MEOW" displayed.

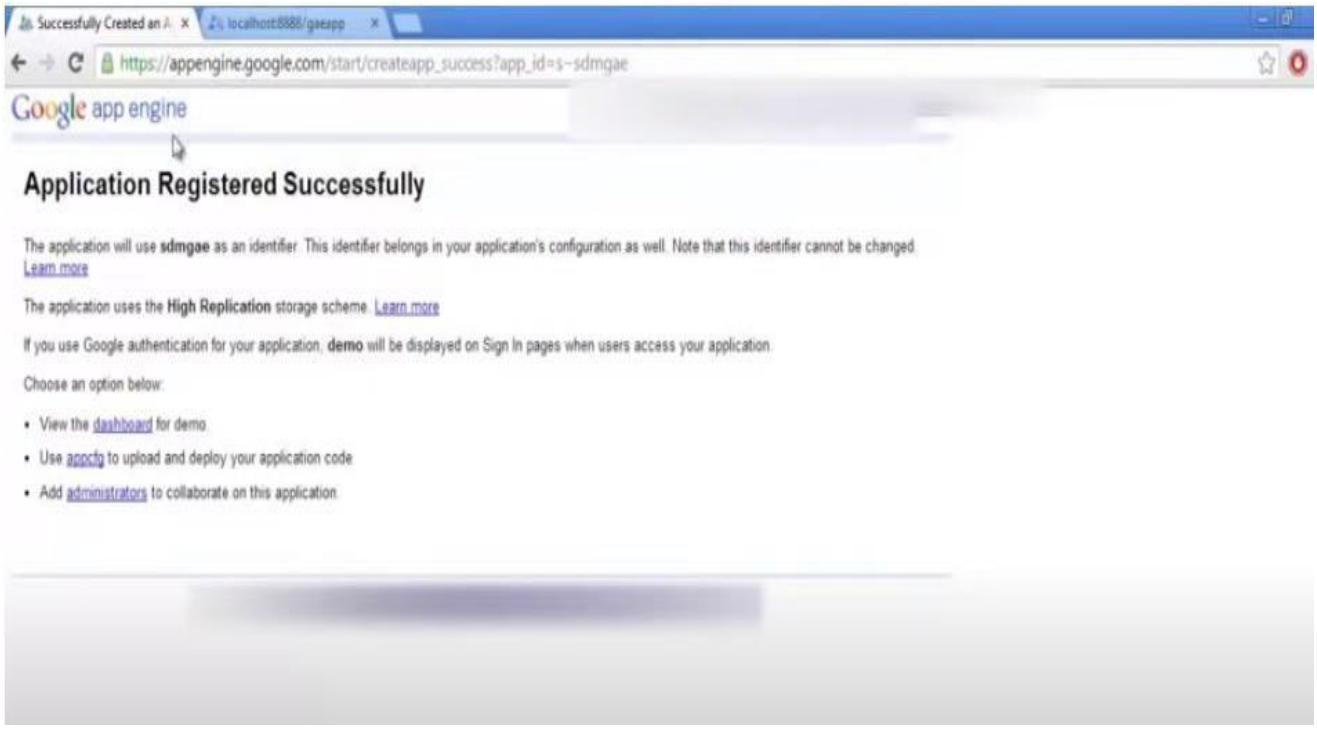
Finally, it is time to deploy your changes to the cloud to make them globally accessible via a URL. Go to the App Engine launcher, select the app, and hit the Deploy button.

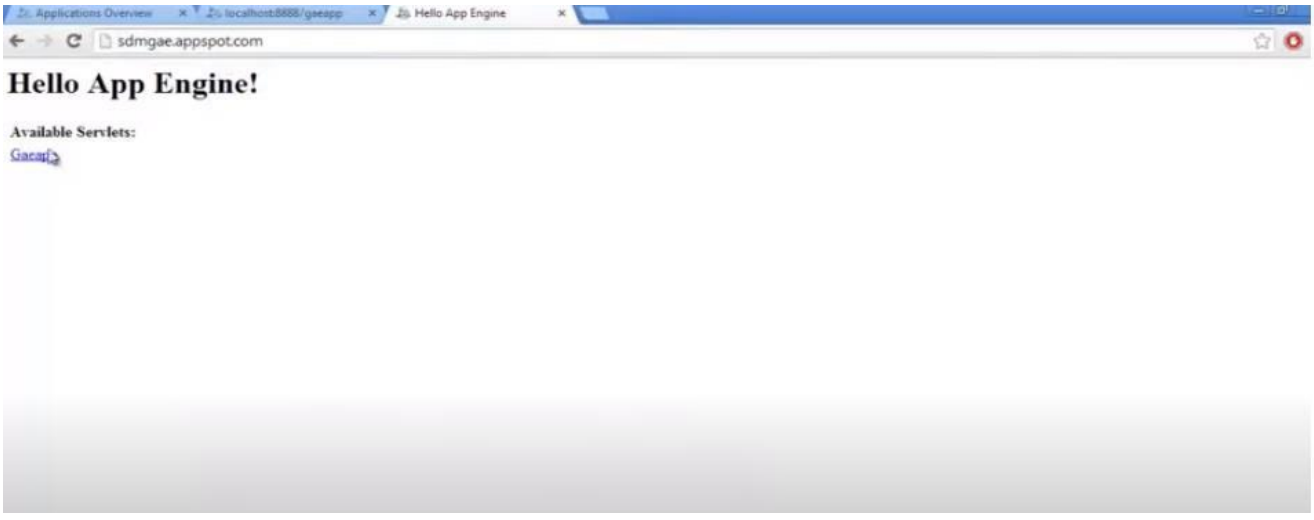
Step 8 : Misc

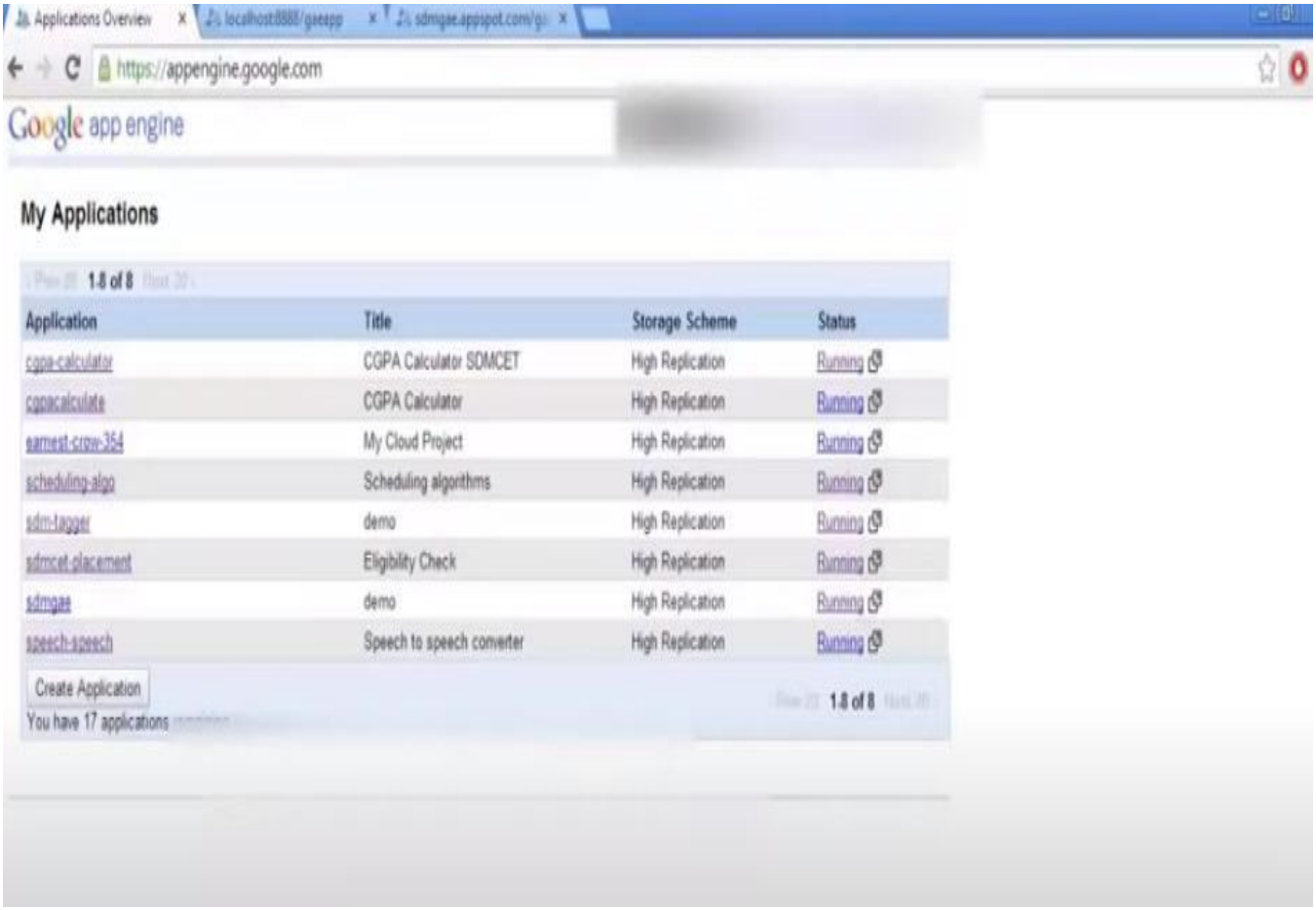
Congratulations, you've finally gotten your first Python webapp deployed on the Google App Engine. Here are some other points which you may find useful.

1. Jinja 2 is an amazing front end templating library for Python that can do some cool stuff, such as passing objects form Python to HTML, using for loops, if conditions, and so on directly out of the box
2. Here's a very useful Udacity course on web development that I have personally found quite resourceful
3. Viewing the logs while running your webapp can be handy to debug and also discover some bugs on the fly.

OUTPUT:


Successfully Created an A
localhost:8888/gaeapp
https://appengine.google.com/start/createapp_success?app_id=s~sdmgae
Google app engine
Application Registered Successfully
The application will use sdmgae as an identifier. This identifier belongs in your application's configuration as well. Note that this identifier cannot be changed.
Learn more
The application uses the High Replication storage scheme. Learn more
If you use Google authentication for your application, demo will be displayed on Sign In pages when users access your application.
Choose an option below:
• View the dashboard for demo
• Use appcfg to upload and deploy your application code
• Add administrators to collaborate on this application


Applications Overview
localhost:8888/gaeapp
Hello App Engine
sdmgae.appspot.com
Hello App Engine!
Available Servlets:
Gacapi



Result :

Thus the GAE launcher to launch the Applications Successfully.

Ex.No : 5	**Simulate a cloud scenario using CloudSim and run a**
Date :	**scheduling algorithm that is not present in CloudSim.**

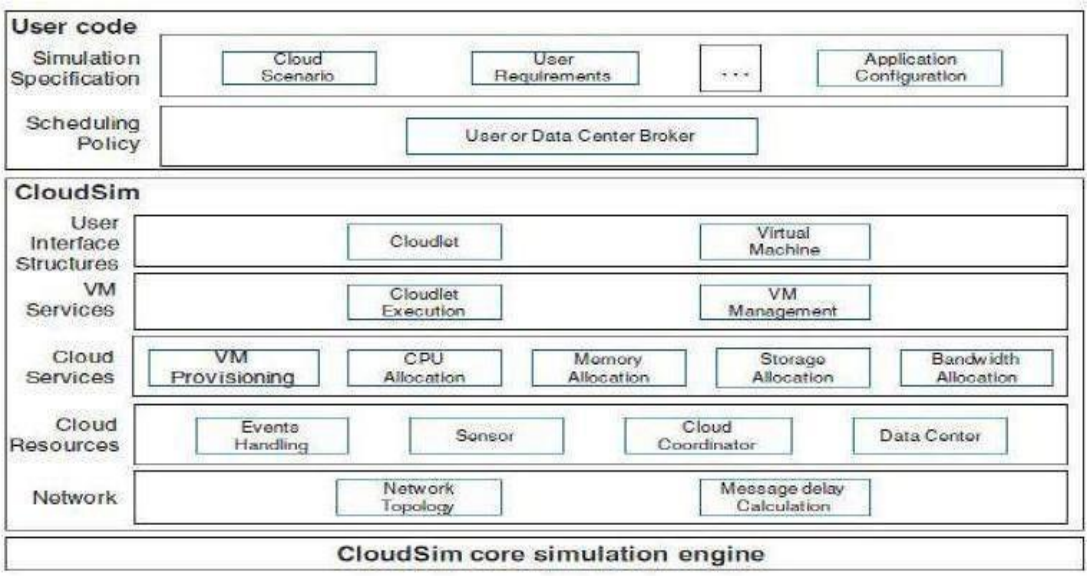
Aim :

To Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim

Introduction:

- **CloudSim**
  - A Framework for modeling and simulation of Cloud Computing Infrastructures and services
  - Originally built at the Cloud Computing Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia
  - It is completely written in JAVA
- **Main Features of CloudSiM**
  - Modeling and simulation
  - Data centre network topologies and message-passing applications
  - Dynamic insertion of simulation elements
  - Stop and resume of simulation
  - Policies for allocation of hosts and virtual machines
- **Cloudsim – Essentials**
  - JDK 1.6 or above http://tinyurl.com/JNU-JAVA
  - Eclipse 4.2 or above http://tinyurl.com/JNU-Eclipse
  - Alternatively NetBeanshttps://netbeans.org/downloads
  - Up & Running with cloudsim guide: https://goo.gl/TPL7Zh
- **Cloudsim-Directory structure**
  - cloudsim/ -- top level CloudSim directory
  - docs/ -- CloudSim API Documentation
  - examples/ -- CloudSim examples
  - jars/ -- CloudSim jar archives
  - sources/ -- CloudSim source code

❖ **Cloudsim - Layered Architecture**



❖ **Cloudsim - Component model classes**

- CloudInformationService.java
- Datacenter.java,Host.java,Pe.java
- Vm.java,Cloudlet.java
- DatacenterBroker.java
- Storage.java,HarddriveStorage.java, SanStorage.java

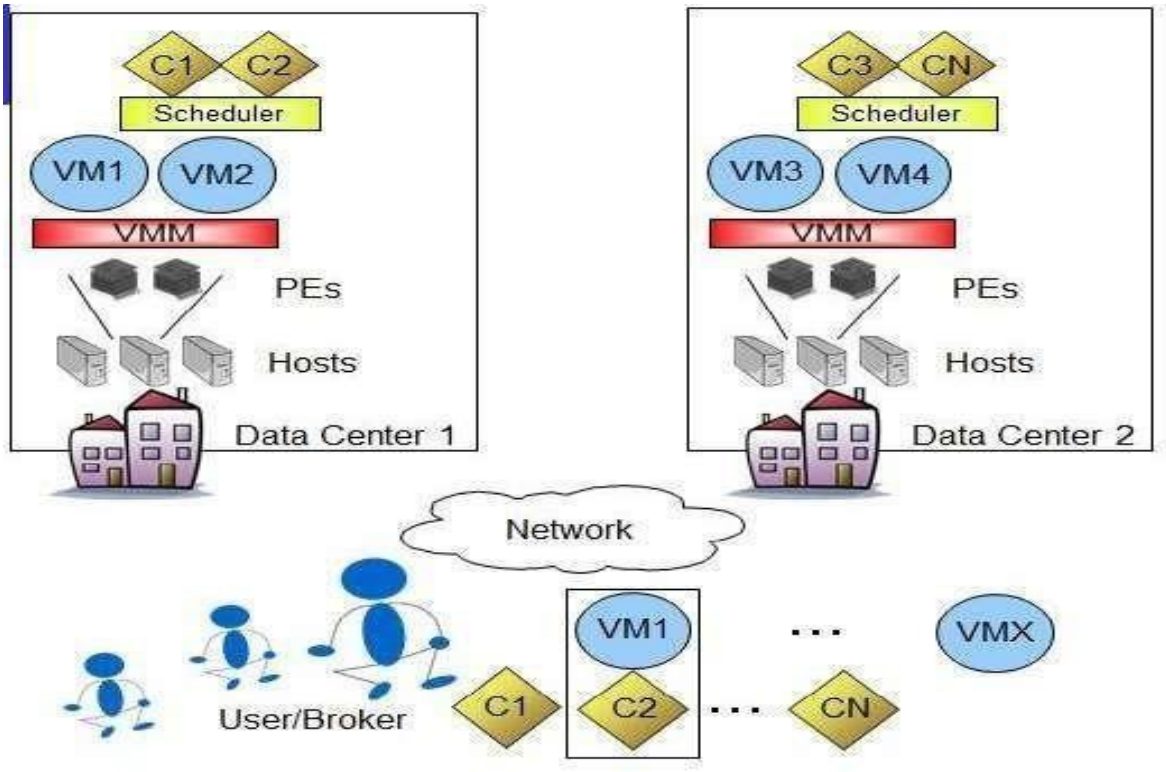
❖ **Cloudsim - Major blocks/Modules**

- org.cloudbus.cloudsim
- org.cloudbus.cloudsim.core
- org.cloudbus.cloudsim.core.predicates
- org.cloudbus.cloudsim.distributions
- org.cloudbus.cloudsim.lists
- org.cloudbus.cloudsim.network
- org.cloudbus.cloudsim.network.datacenter
- org.cloudbus.cloudsim.power
- org.cloudbus.cloudsim.power.lists
- org.cloudbus.cloudsim.power.models
- org.cloudbus.cloudsim.provisioners
- org.cloudbus.cloudsim.util

❖ **Cloudsim - key components**

- Datacenter
- DataCenterCharacteristics
- Host
- DatacenterBroker
- RamProvisioner
- BwProvisioner
- Storage
- Vm
- VMAllocationpolicy
- VmScheduler
- Cloudlet
- CloudletScheduler

- CloudInformationService
- CloudSim
- CloudSimTags
- SimEvent
- SimEntity
- CloudsimShutdown
- FutureQueue
- DefferedQueue
- Predicate and associative classes.

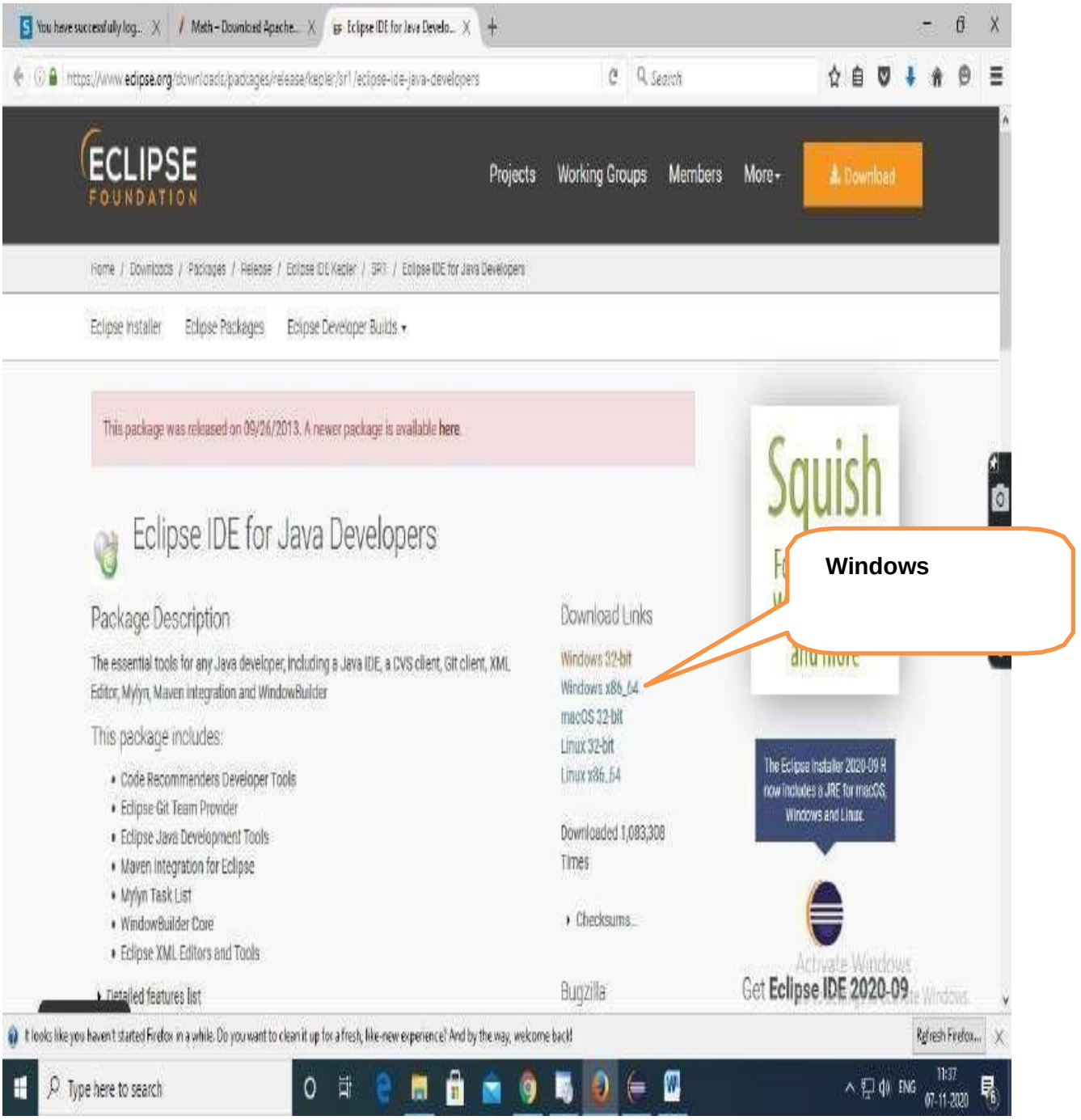


CloudSim Elements/Components

Procedure to import Eclipse, Cloudsim in your system

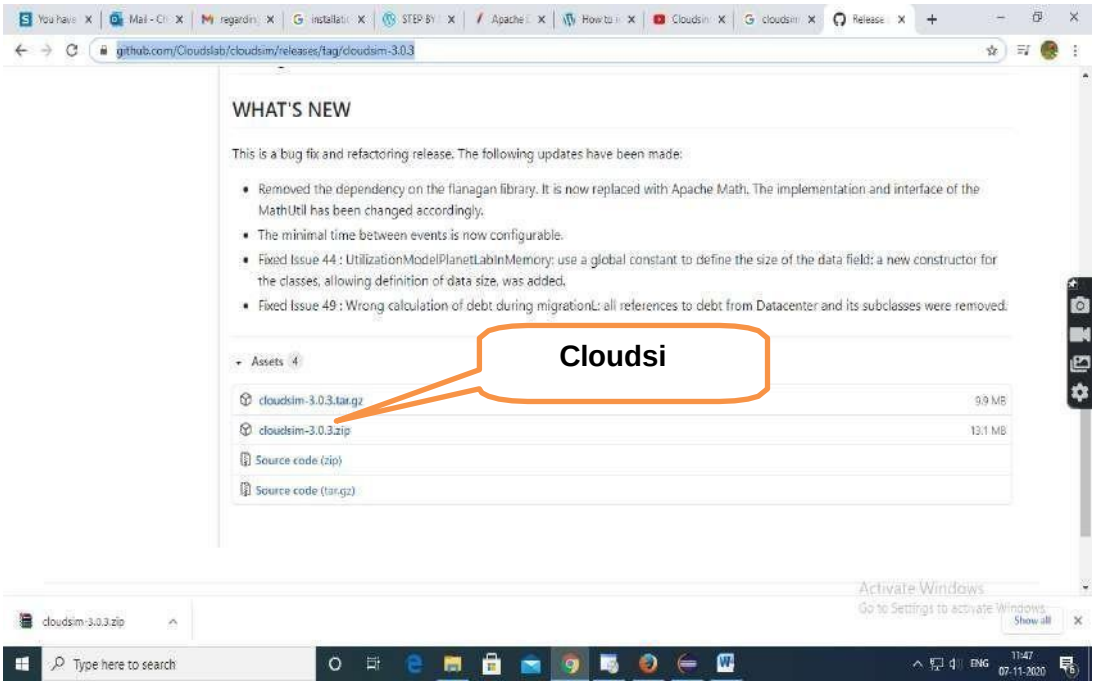
Step 1: Link to download Eclipse and download Eclipse for Windows 64bit into your Local machine

https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers



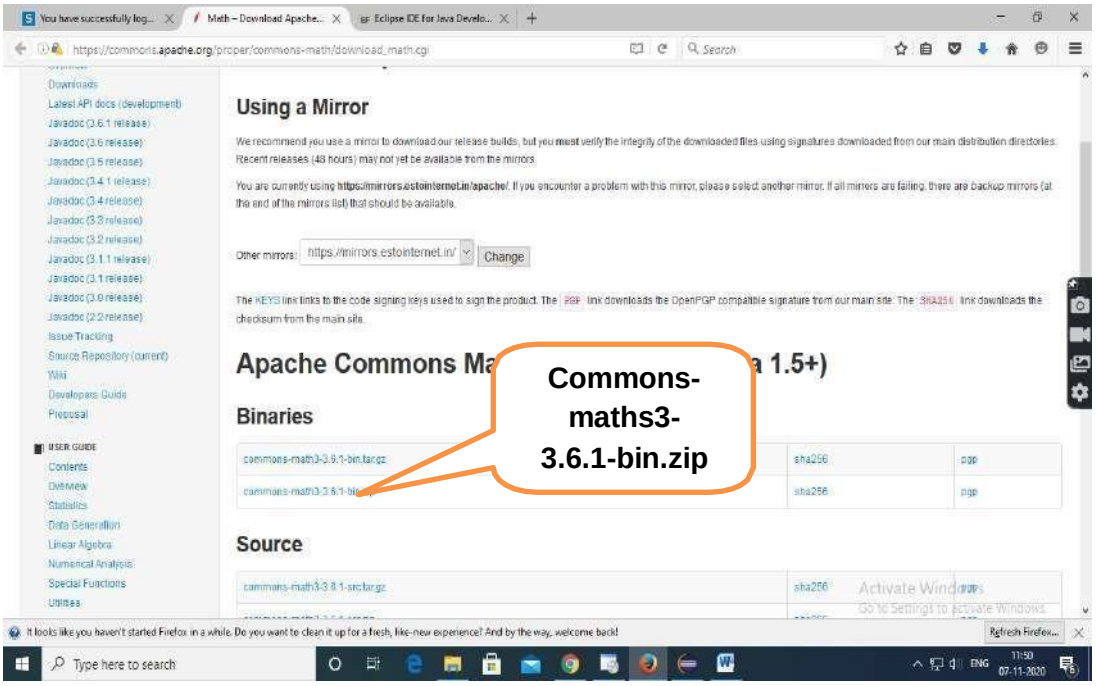
Step 2: Download cloudsim-3.0.3 from git hub repository in your local machine

https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3

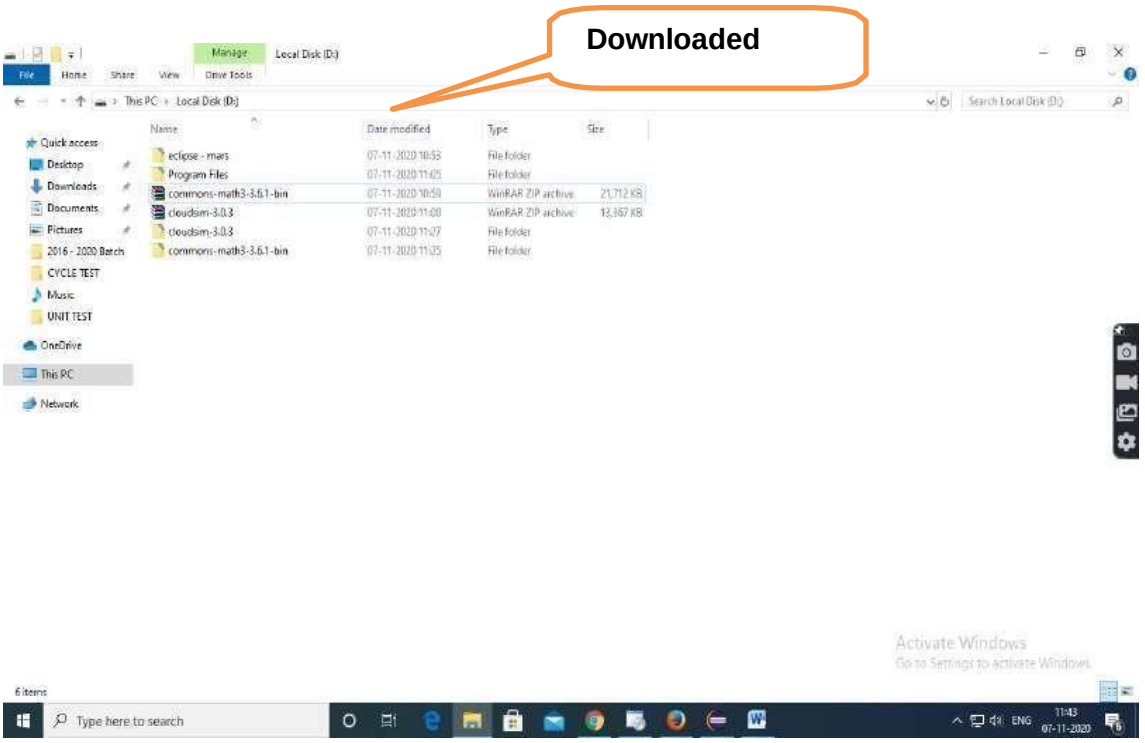


Step 3: Download commons-maths3-3.6.1 from git hub repository in your local machine

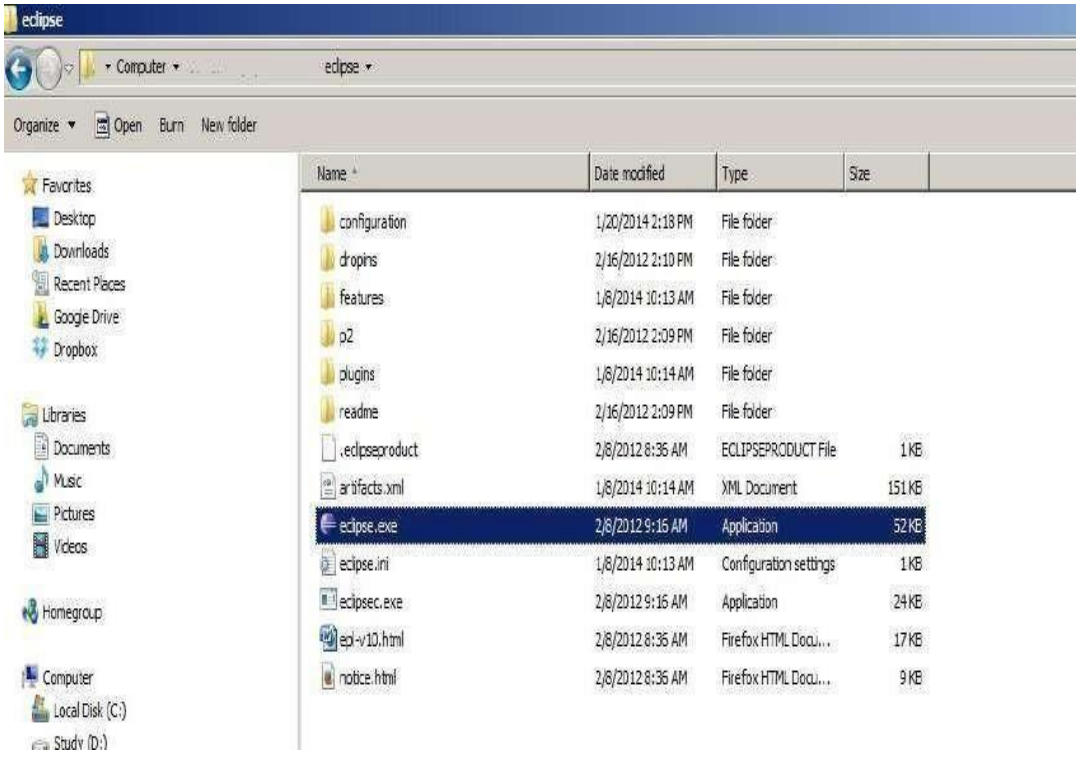
https://commons.apache.org/proper/commons-math/download_math.cgi



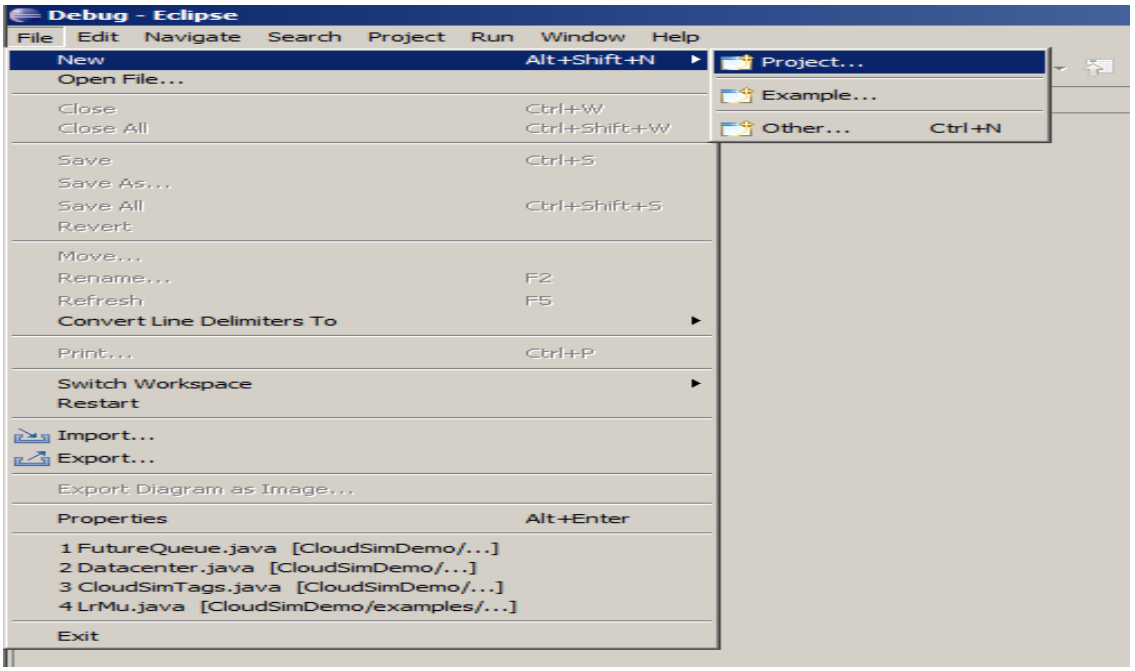
Step 4: Downloaded Eclipse, cloudsim-code-master and Apache Commons Math 3.6.1 in your local machine and extract cloudsim-3.0.3 and Apache Commons Math 3.6.1



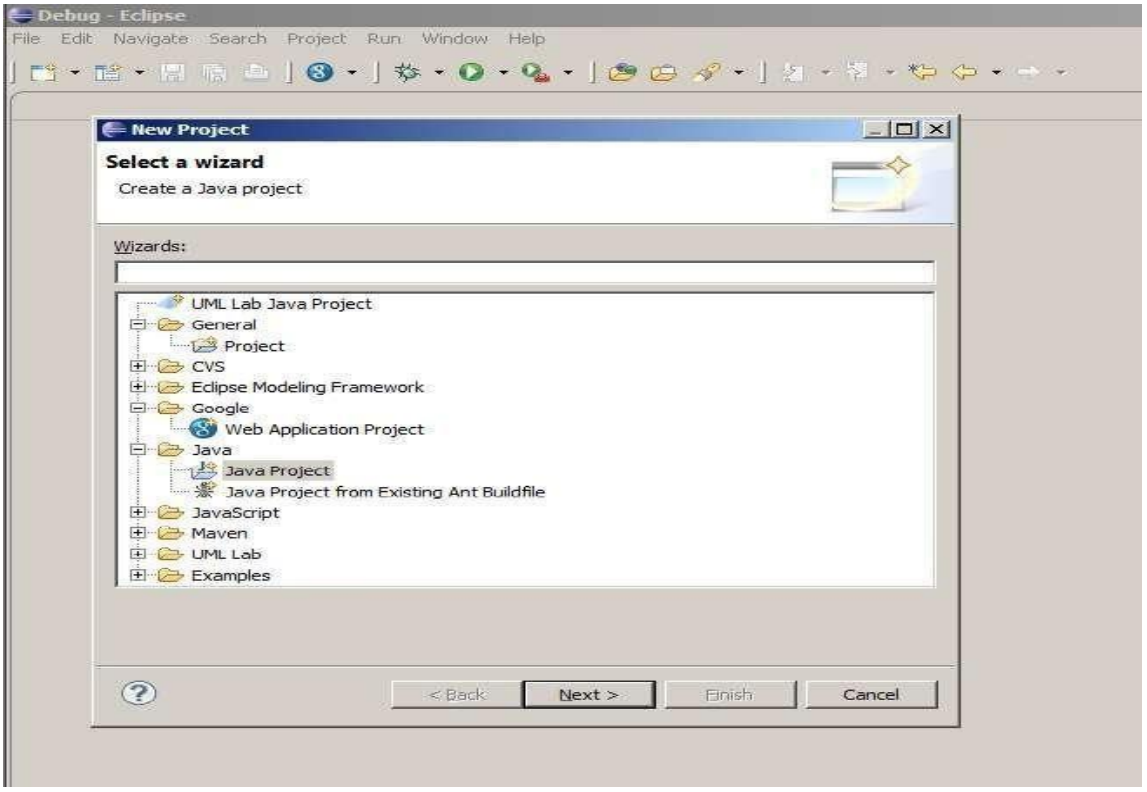
Step 5: First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe



Step 6: Now within Eclipse window navigate the menu: *File -> New -> Project,* to open the new project wizard

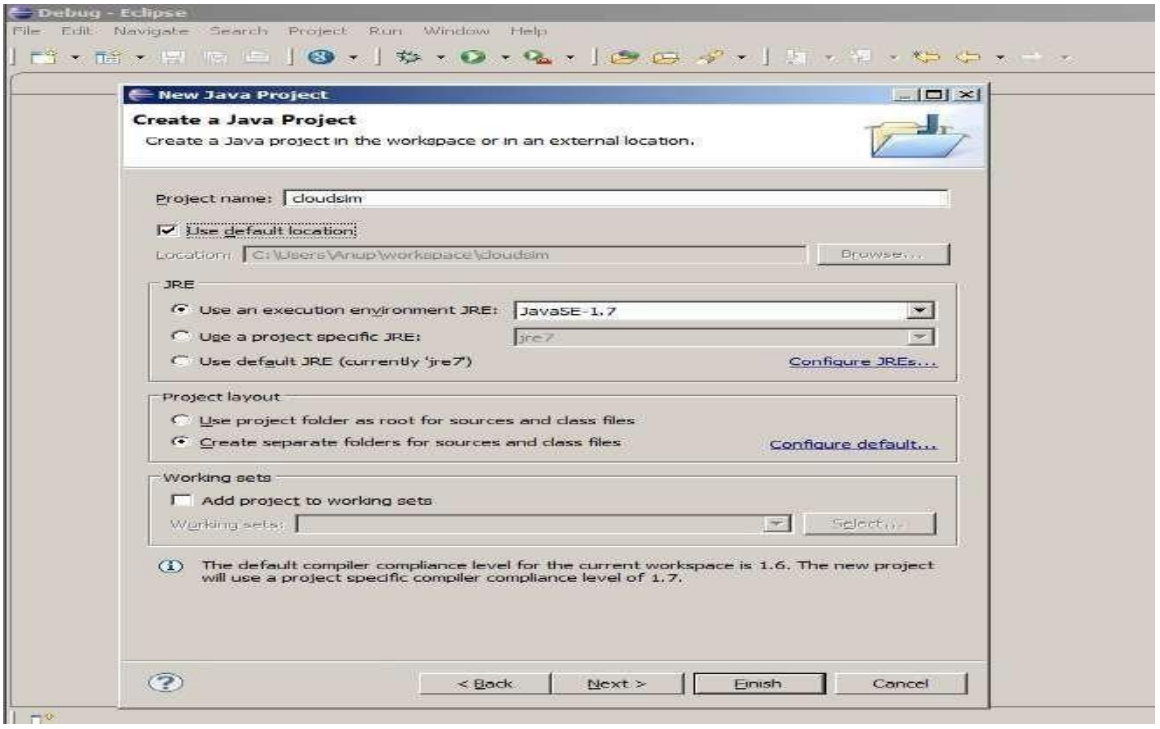


Step 7: A _*New Project*_ wizard should open. There are a number of options displayed and you have to find & select the _*Java Project*_ option, once done click *'Next*_

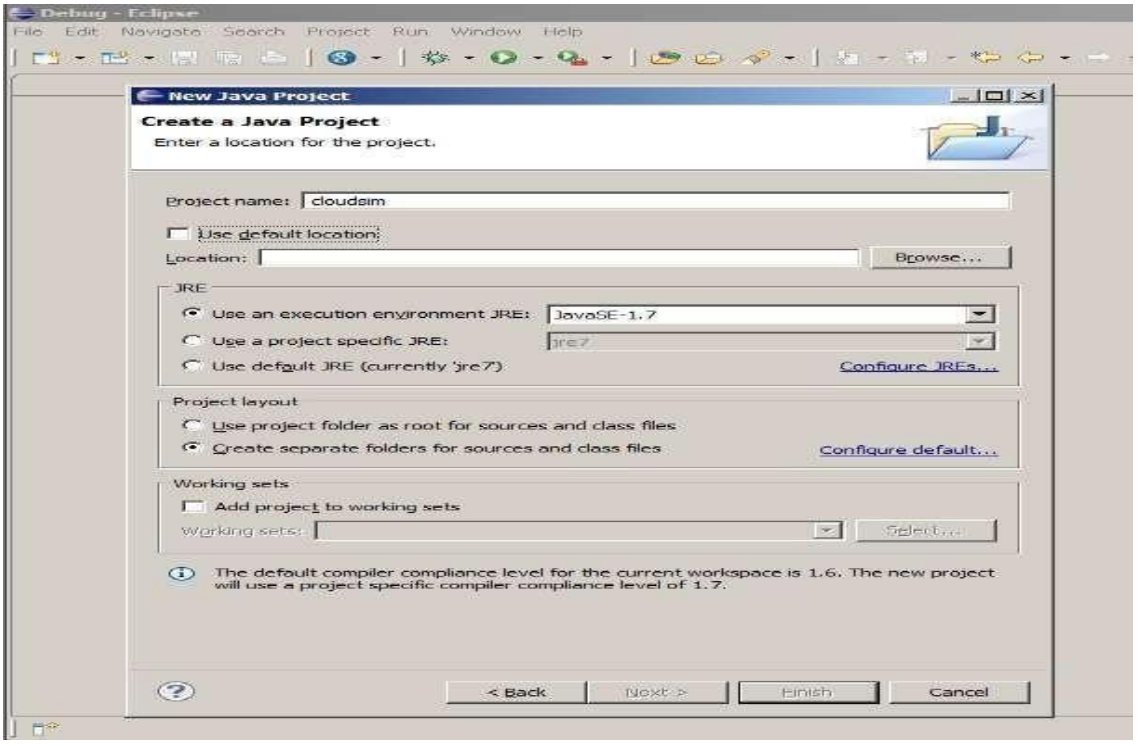


Step 8: Now a detailed new project window will open, here you will provide the project name and the path of CloudSim project source code, which will be done as follows:

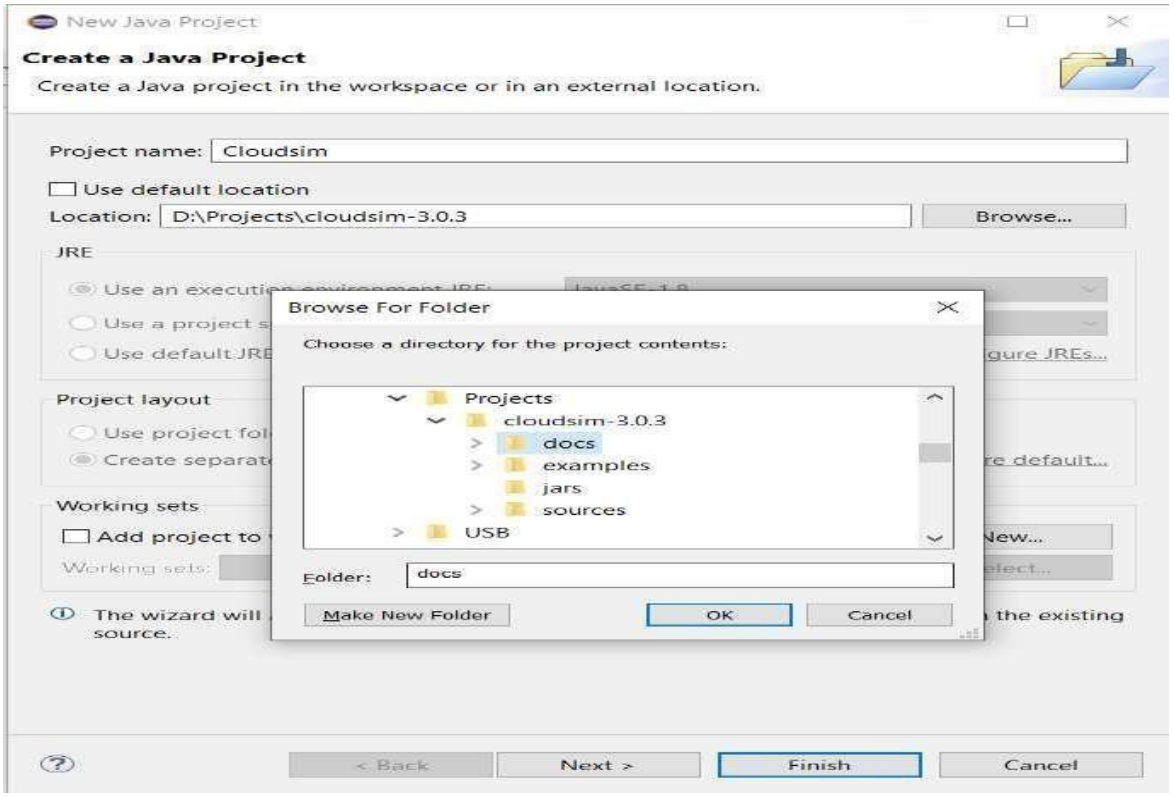
Project Name: CloudSim.



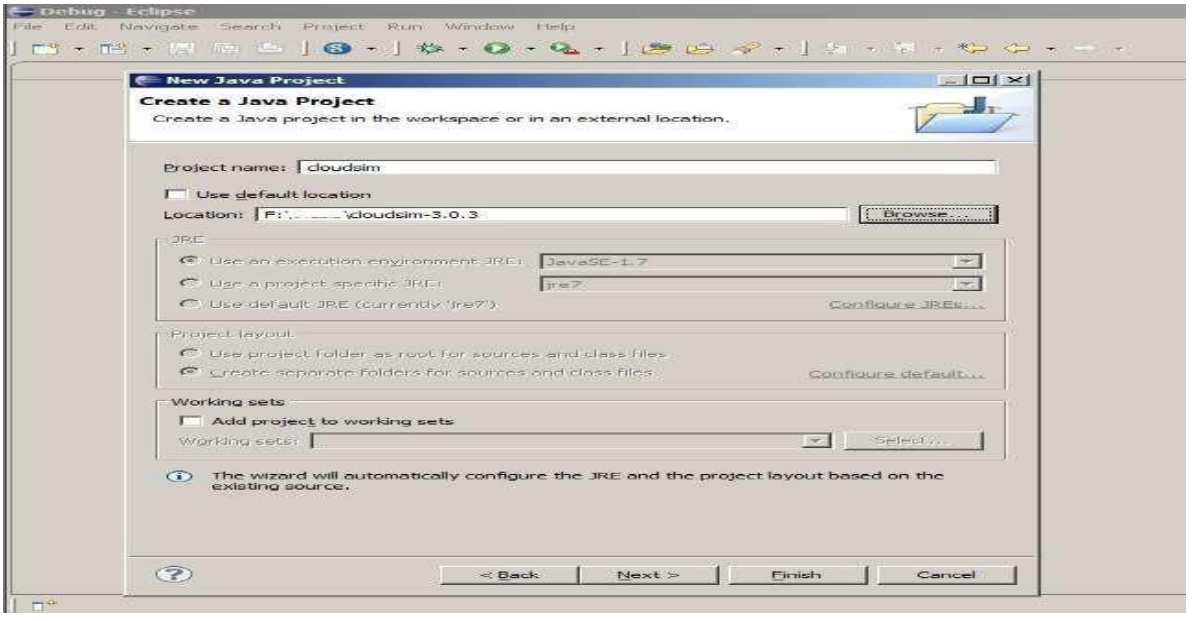
Step 9: Unselect the *'Use default location'* option and then click on *'Browse'* to open the path where you have unzipped the Cloudsim project and finally click Next to set project settings.



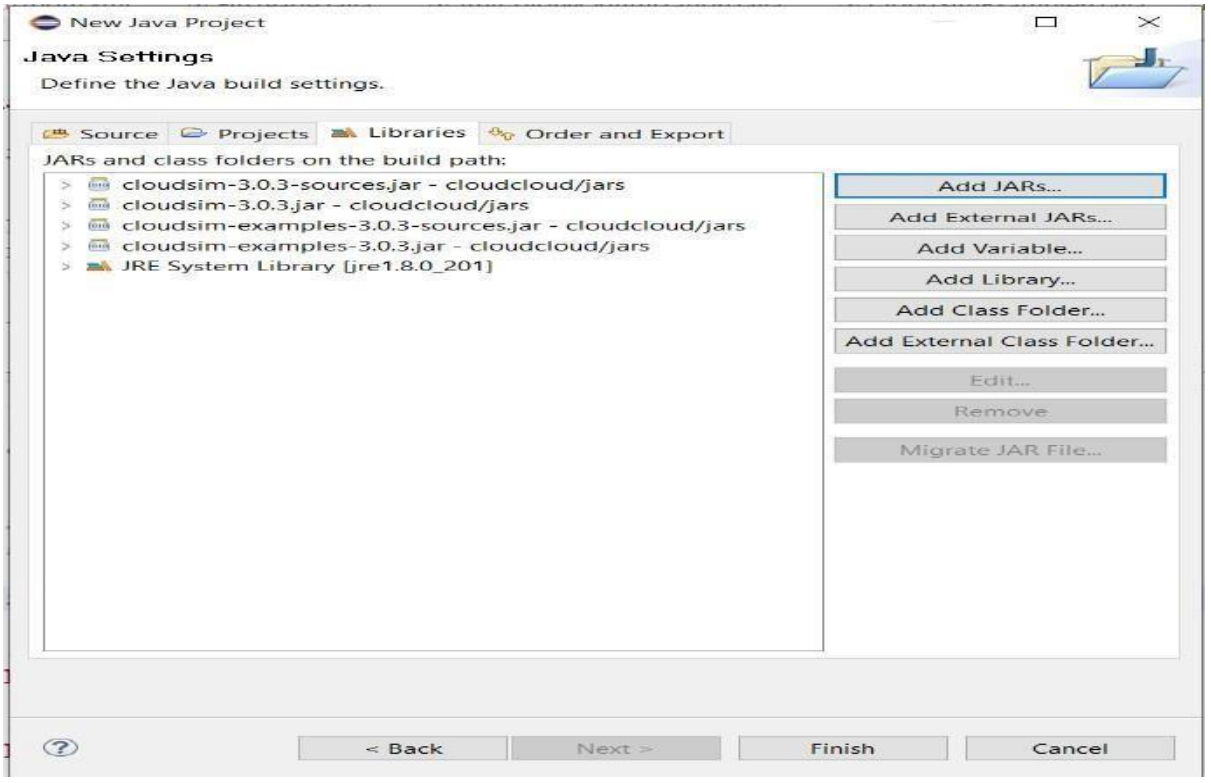
Step 10: Make sure you navigate the path till you can see the bin, docs, examplesetc folder in the navigation plane.



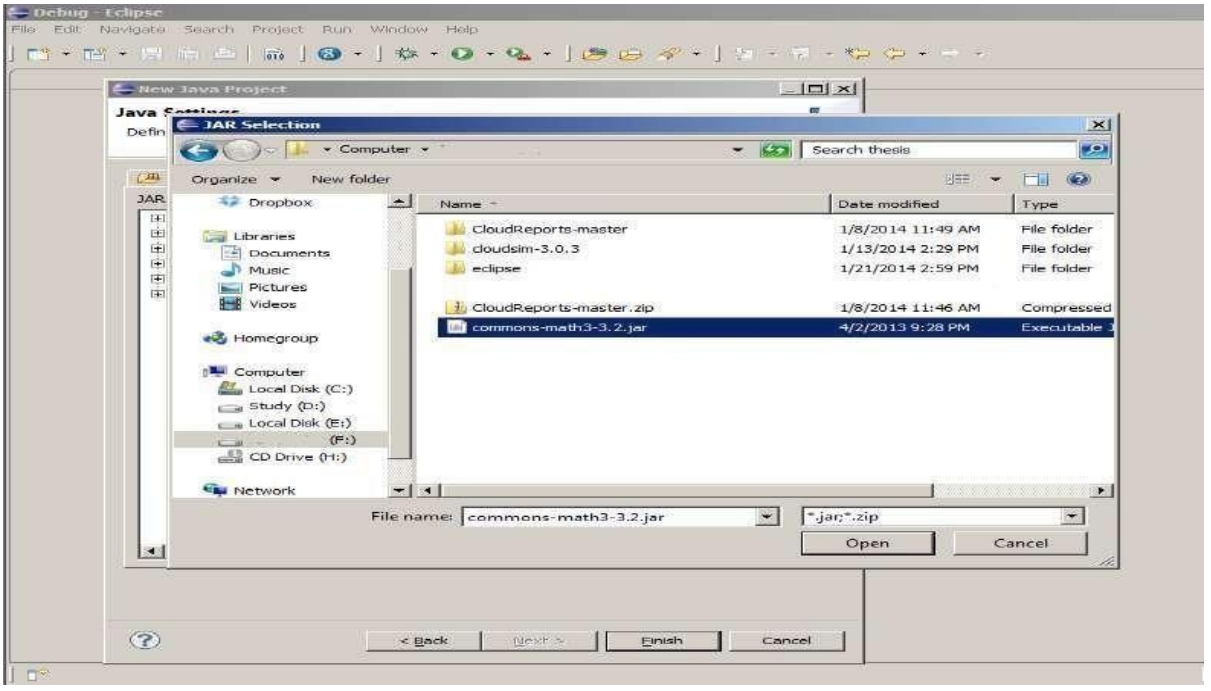
Step 11: Once done finally, click _Next,, to go to the next step i.e. setting up of project settings



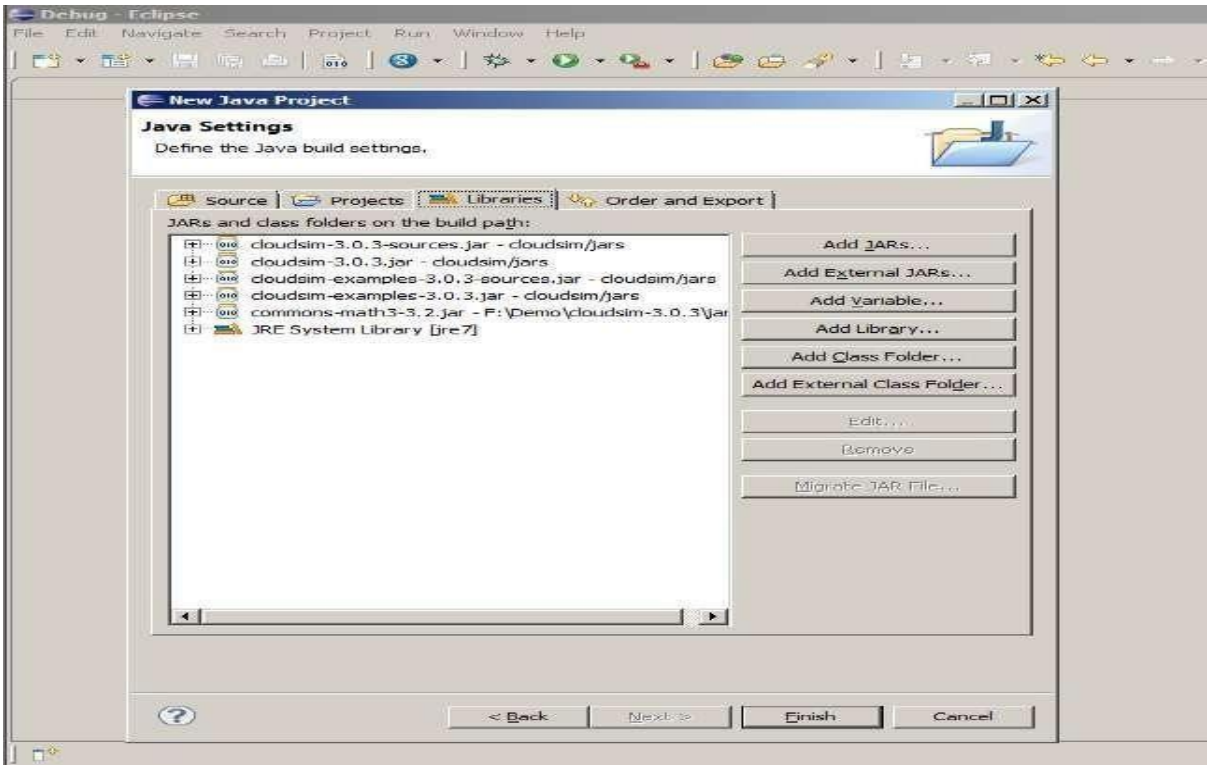
Step 12: Now open *'Libraries'* tab and if you do not find commons-math3-3.x.jar *(here 'x' means the minor version release of the library which could be 2 or greater*) in the list then simply click on *Add External Jar'* (commons-math3-3.x.jar will be included in the project from this step)



Step 13: Once you have clicked on *Add External JAR's* Open the path where you have unzipped the commons-math binaries and select *Commons-math3-3.x.jar*,, and click on open.

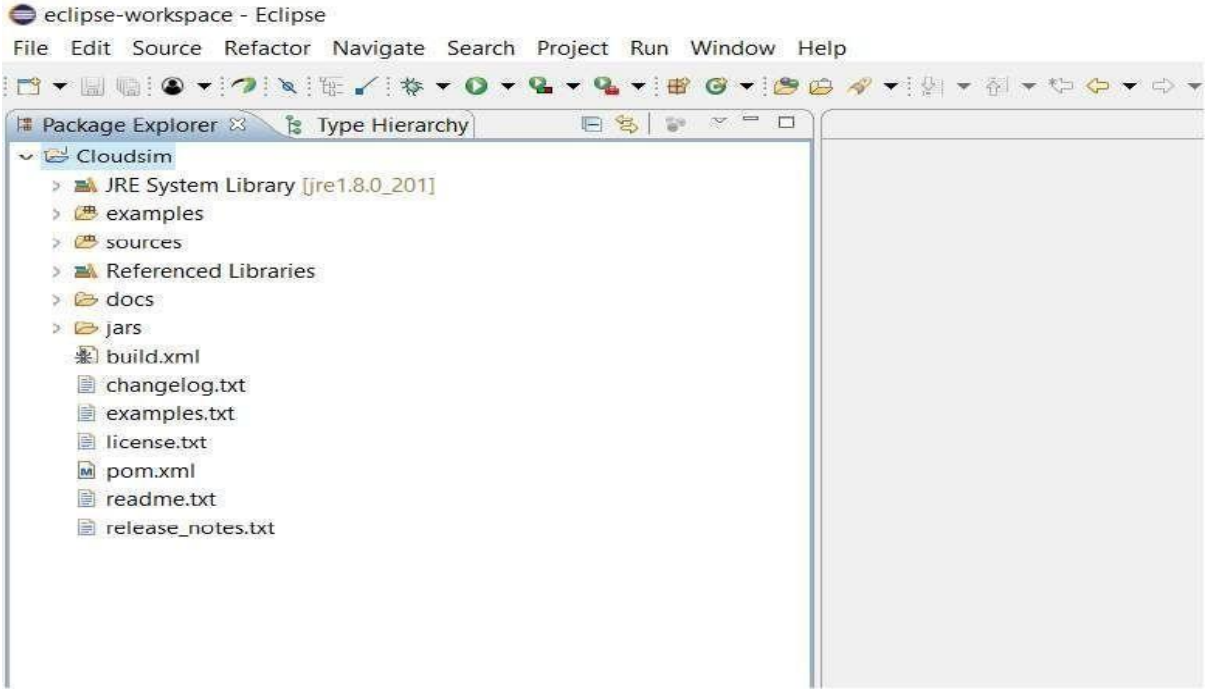


Step 14: Ensure external jar that you opened in the previous step is displayed in the list and then click on _*Finish*„ (your system may take 2-3 minutes to configure the project)

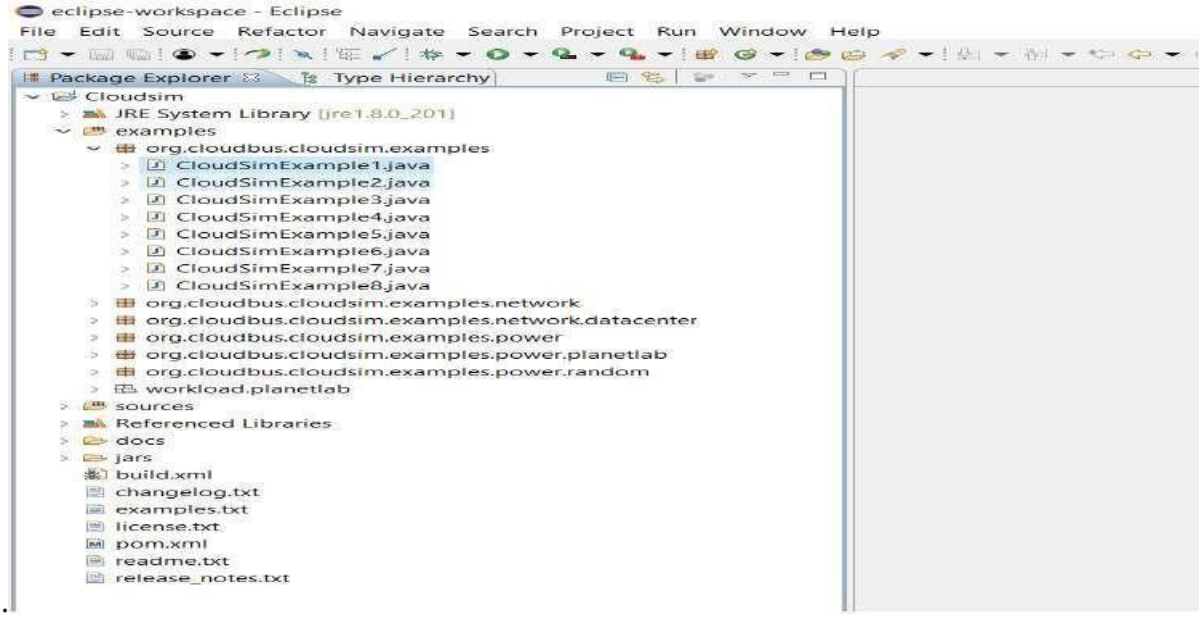


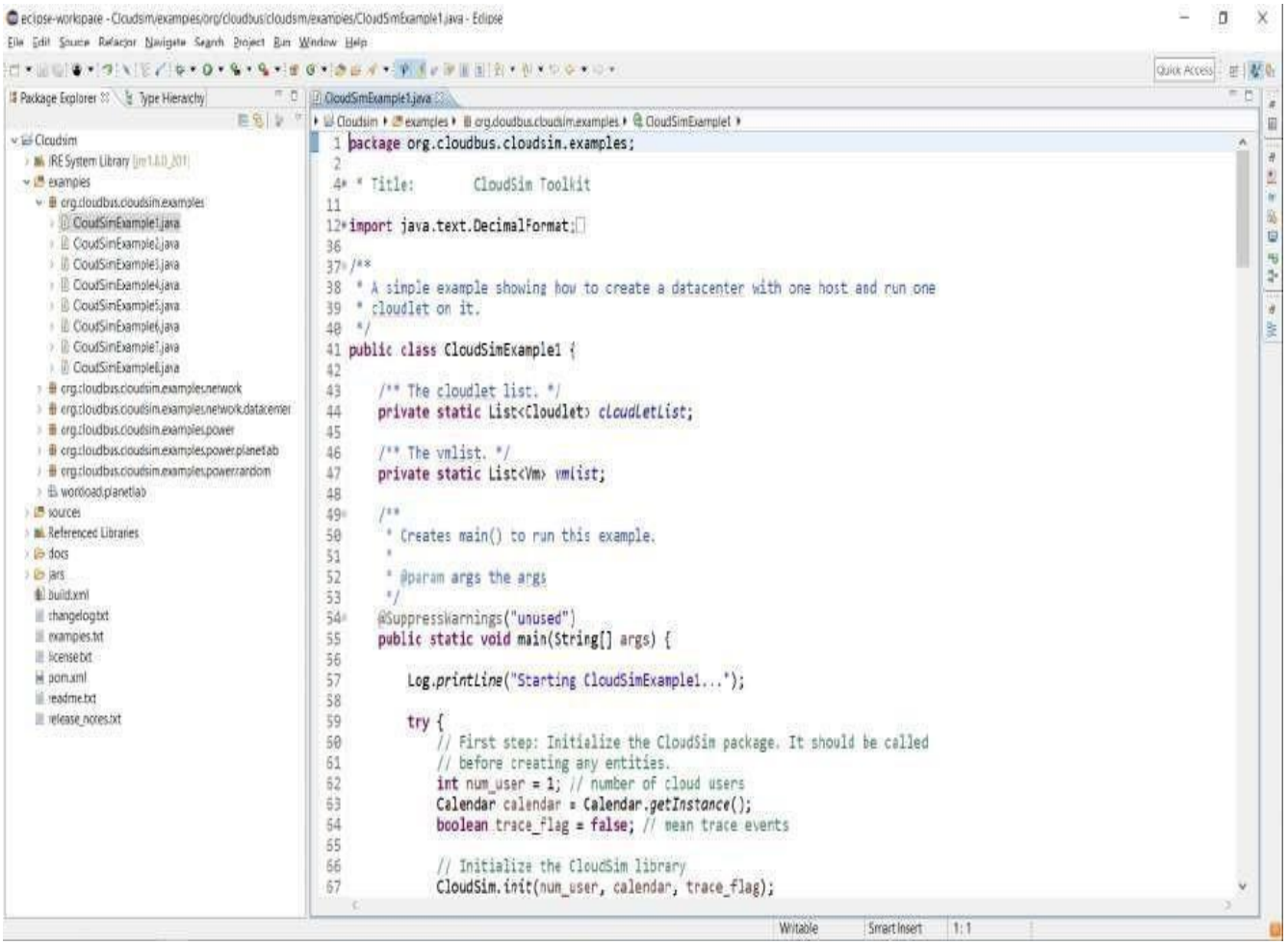
Step 15: Once the project is configured you can open the _*Project Explorer*_and start exploring the Cloudsim project. Also for the first time eclipse automatically start building the workspace for newly configured Cloudsim project, which may take some time depending on the configuration of the computer system.

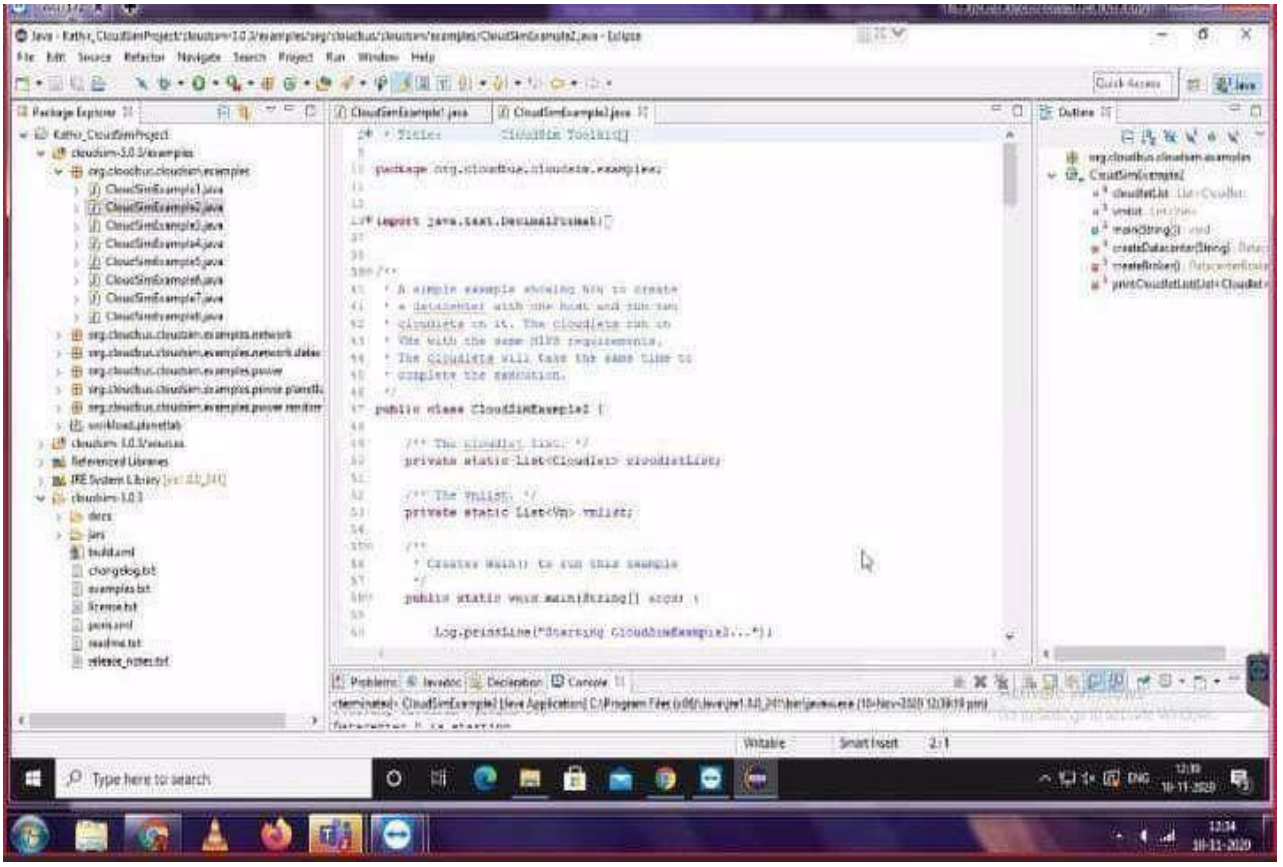
Following is the final screen which you will see after Cloudsim is configured.



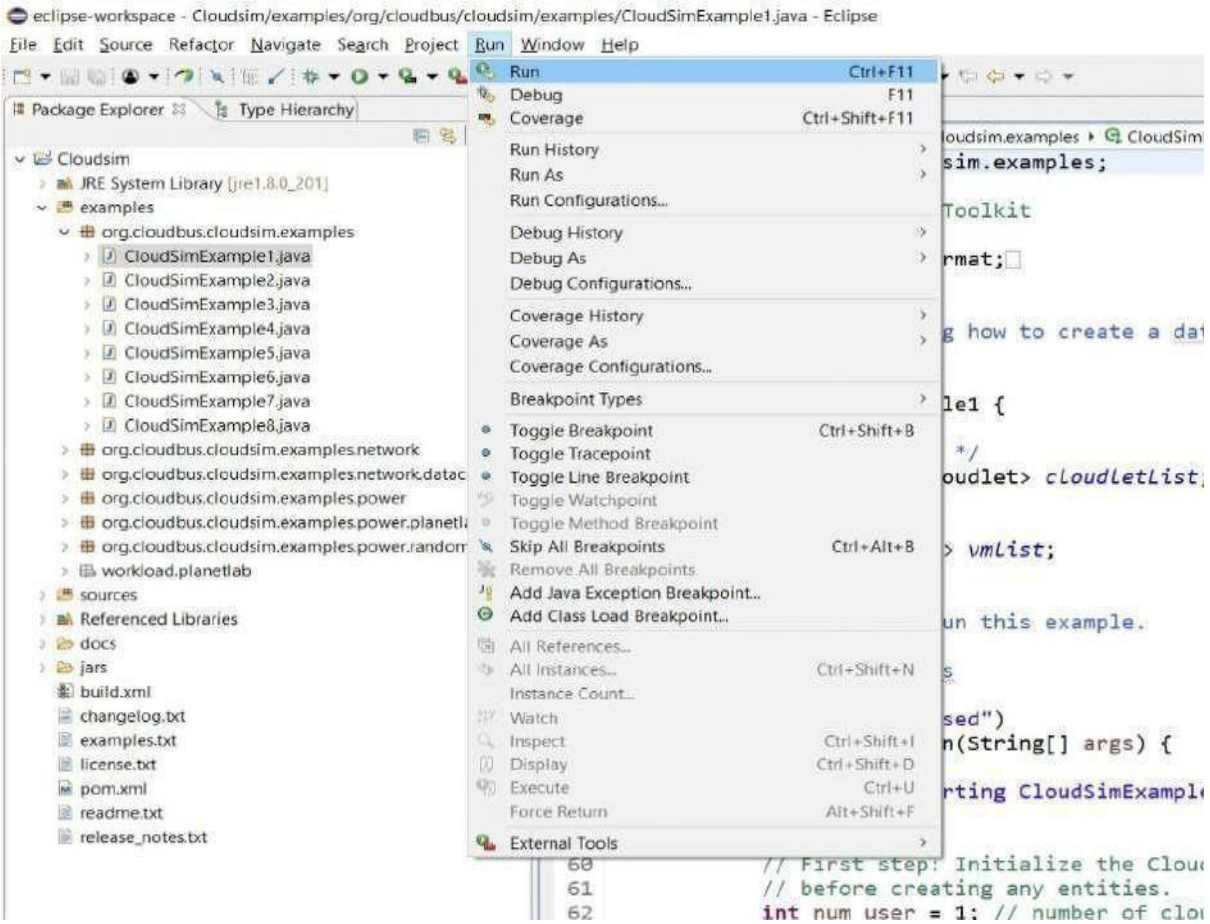
Step 16: Now just to check you within the **_Project Explorer_**, you should navigate to the **_examples_** folder, then expand the package *_org.cloudbus.cloudsim.examples_* and double click to open the *_CloudsimExample1.java_*



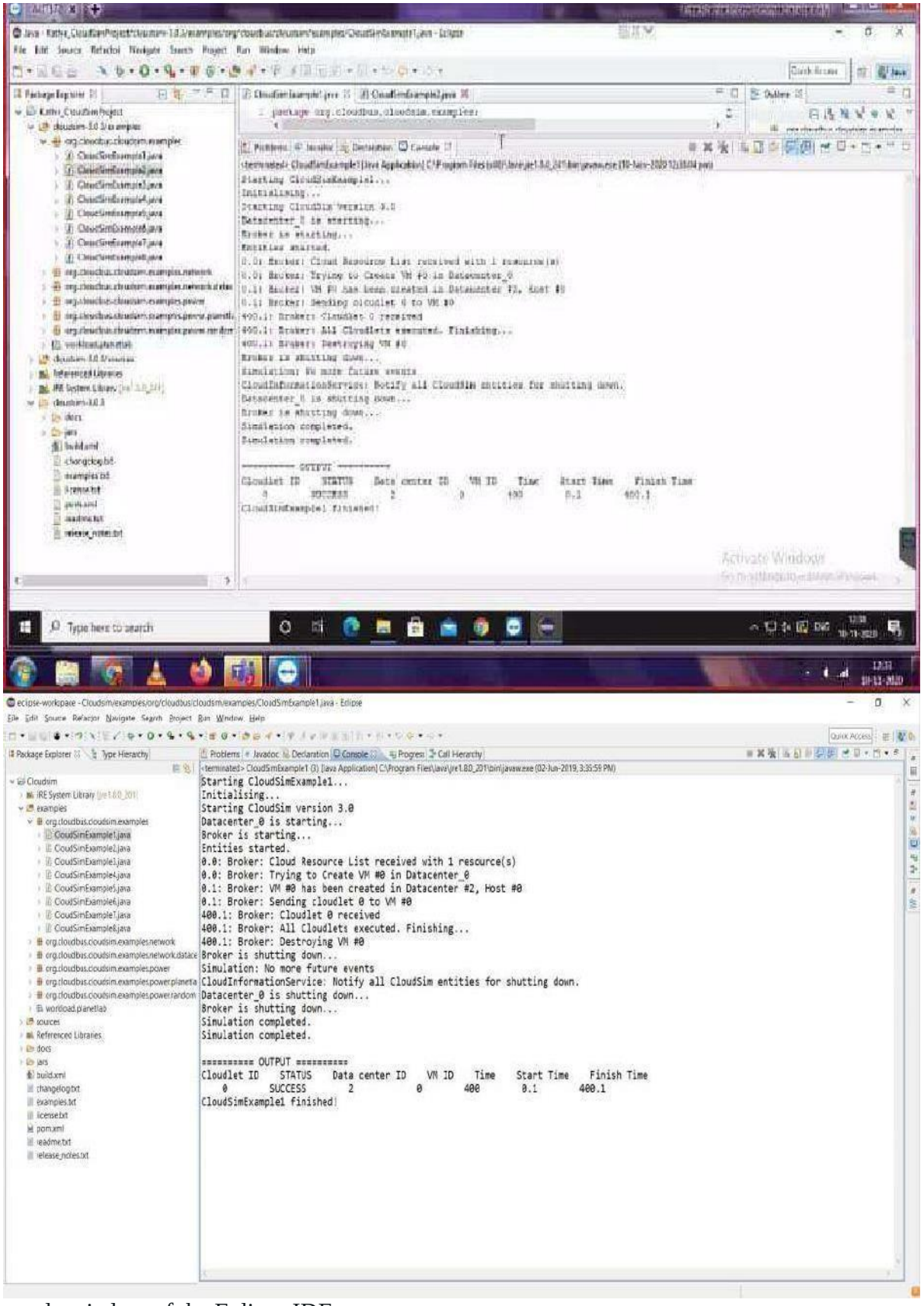




Step 17: Now navigate to the Eclipse menu *_Run ->Run_* or directly use a keyboard shortcut *'Ctrl + F11'* to execute the *_CloudsimExample1.java_*.



Step 18: If it is successfully executed it should be displaying the following type to output inthe



console window of the Eclipse IDE.

Result:

Thus the cloudsim is simulated using Eclipse Environment successfully.

Ex.No : 6 **Find a procedure to transfer the files from one virtual machine to another virtual machine.**

Date :

Aim:

To procedure File Transfer in Client & Server using virtual machine

Steps:

Steps to perform File Transfer in Client & Server using virtual machine.

Step 1: Open a virtual machine to do file transfer.

Step 2: Write the java program for FTP Client and FTP Server.
Step 3: Run the program.

Source Code:

FTPClient.java

```
import java.io.*;
import java.net.*;
import java.util.*;
public class
FTPClient
{
        public static void main(String args[])throws IOException
{
try
 {
        int number;
        Socket s=new Socket("127.0.0.1",10087);
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the file name:");
        String fn=sc.next();
        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
        dos.writeUTF(fn);
        DataInputStream dis=new DataInputStream(s.getInputStream());
        String input=(String)dis.readUTF();
        FileInputStream fis=new FileInputStream(input);
        System.out.println("Even Numbers in the" +fn+" are");
        int i=0;
{
```

```
while((i=fis.read())!=-1)


System.out.println((char)i);

}
s.close();
}
catch(Exception e){
System.out.println("Port not available "+e);
}
}
}
```

FTPServer.java

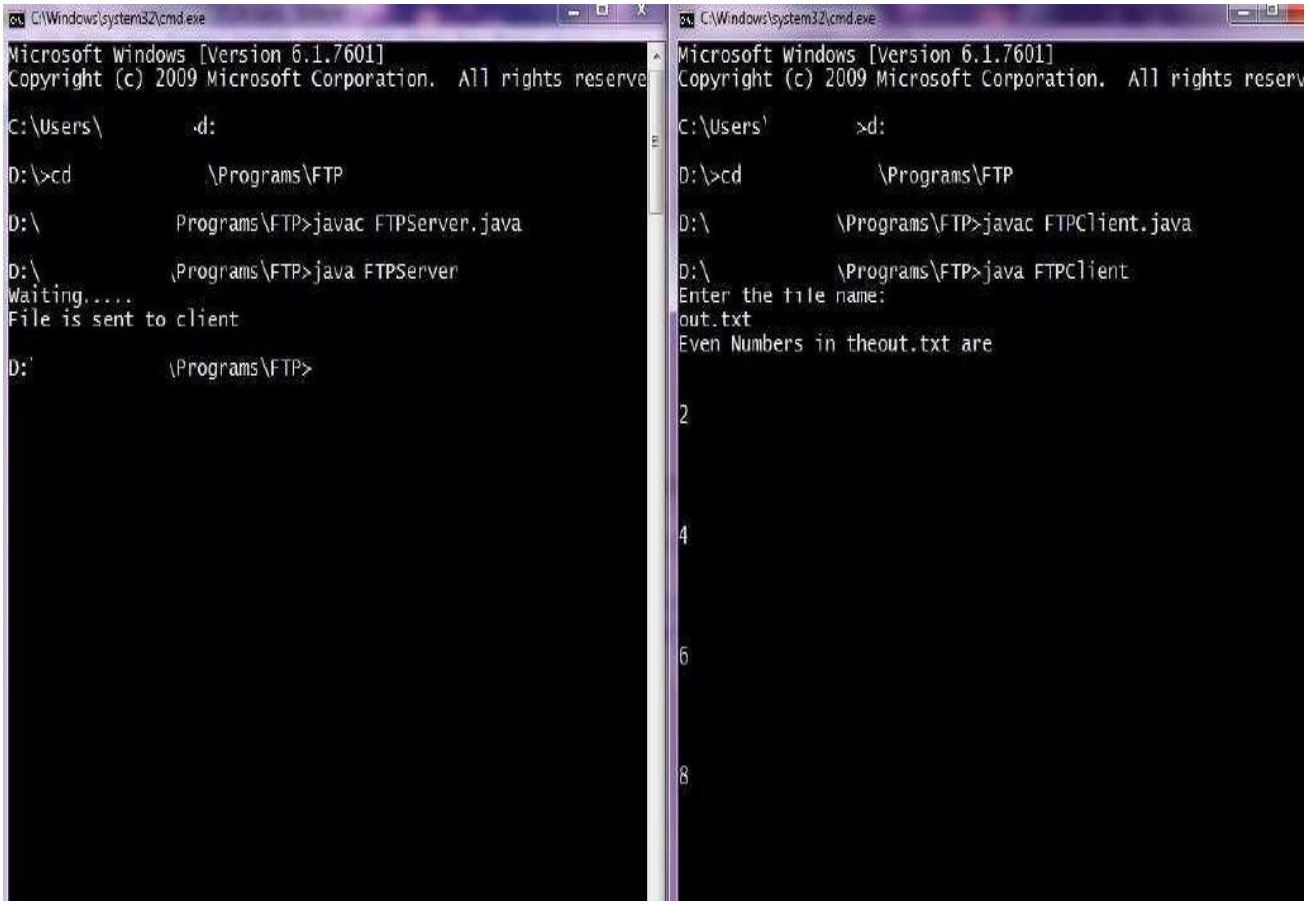
```
import java.io.*;
import java.net.*;
import java.util.*;
public class
FTPServer
{
        public static void main
(
        String args[]
)
        throws IOException
{
 Try
{
        int num;
        Scanner sc=new Scanner(System.in);
        ServerSocket ss=new ServerSocket(10087);
        Socket s=ss.accept();
        System.out.println("Waiting..... ");
        DataInputStream dis=new DataInputStream(s.getInputStream());
        String input=(String)dis.readUTF();
        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
        FileInputStream fis = new FileInputStream("out.txt");
        FileOutputStream fos = new FileOutputStream(input);
        while((num=fis.read())!= -1)
{
```

```
if(num%2==0)
{
        fos.write(num);
}
}
        dos.writeUTF(input);
        System.out.println("File is sent to client");
        ss.close();
        s.close();
}
catch(Exception e)
{
        System.out.println("Port not available"+e);
}
}
}
```

Out.txt

```
1
2
3
4
5
6
7
8
9
```

Output:



Result:

Thus the program to the File transfer operation using virtual machine was successfully executed and verified.

Ex.No :7 **Install Hadoop single node cluster and run simple applications like Word count**

Date :

Aim :

To Install Hadoop single node cluster and run simple applications like wordcount

Procedure

1. Analyze the input file content
2. Develop the code
    a. Writing a map function
    b. Writing a reduce function
    c. Writing the Driver class
3. Compiling the source
4. Building the JAR file
5. Starting the DFS
6. Creating Input path in HDFS and moving the data into Input path
7. Executing the program

Open in any Browser

- Open in any Browser NameNode - http://localhost:50070/
- Open in any Browser JobTracker - http://localhost:50030/
- Open hadoop/hadoop-1.2.1 create a document type something in that document and saveit as test.txt
- bin/hadoop fs -ls /
    - Found 1 items
    - drwxr-xr-x - vishal supergroup 0 2014-04-15 01:13 /tmp
- bin/hadoop fs -mkdir example
- bin/hadoop fs -ls /user/vishal/
    - Found 1 items
    - drwxr-xr-x - vishal supergroup /user/vishal/example
- bin/hadoop fs -copyFromLocal test.txt /user/vishal/example
- bin/hadoop jar hadoop-examples-1.2.1.jar wordcount /user/vishal/example/test.txt /hello

- In Eclipse New → Java Project → Provide Project Name → Next → Select Libraries → Add Externals JARs → Go to Hadoop → hadoop-1.2.1 → select all jar files → againclick on Add External JARs → go to hadoop → hadoop-1.2.1 → lib → select all JAR files → click on Finish.

- Right Click on Src Folder → Select Class → Provide a Class name: WCE → Package name: com.WordCount.Example → Click on Finish.

Program: WordCount.java

```
import java.io.IOException;
import  java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.IntWritable;import
org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job; import
org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>
       {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
StringTokenizer itr = new StringTokenizer(value.toString());while
(itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
```

```
  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
 }
 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

Save the program as

WordCount.java

Step 1: Compile the java program

For compilation we need this hadoop-core-1.2.1.jar file to compile the mapreduce program.

https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1

Assuming both jar and java files in same directory run the following command to compile

root@a4cseh160:/#javac -classpath hadoop-core-1.2.1.jar WordCount.java

Step 2: Create a jar file

Syntax:

jar cf jarfilename.jar MainClassName*.class

Output:

root@a4cseh160:/#jar cf wc.jar WordCount*.class

Step 3: Make directory in hadoop file system

Syntax:

hdfs dfs -mkdir directoryname

Output:

root@a4cseh160:/# hdfs dfs -mkdir /user

Step 4: Copy the input file into hdfs

Syntax:

hdfs dfs -put sourcefile destpath

Output:

root@a4cseh160:/#hdfs dfs -put /input.txt /user

Step 5: **To a run a program**

Syntax:

hadoop jar jarfilename main_class_name inputfile outputpath

Output:

root@a4cseh160:/#hadoop jar wc.jar WordCount /user/input.txt /user/out

Input File: (input.txt)

Cloud and Grid Lab. Cloud and Grid Lab. Cloud Lab.

Output:

18

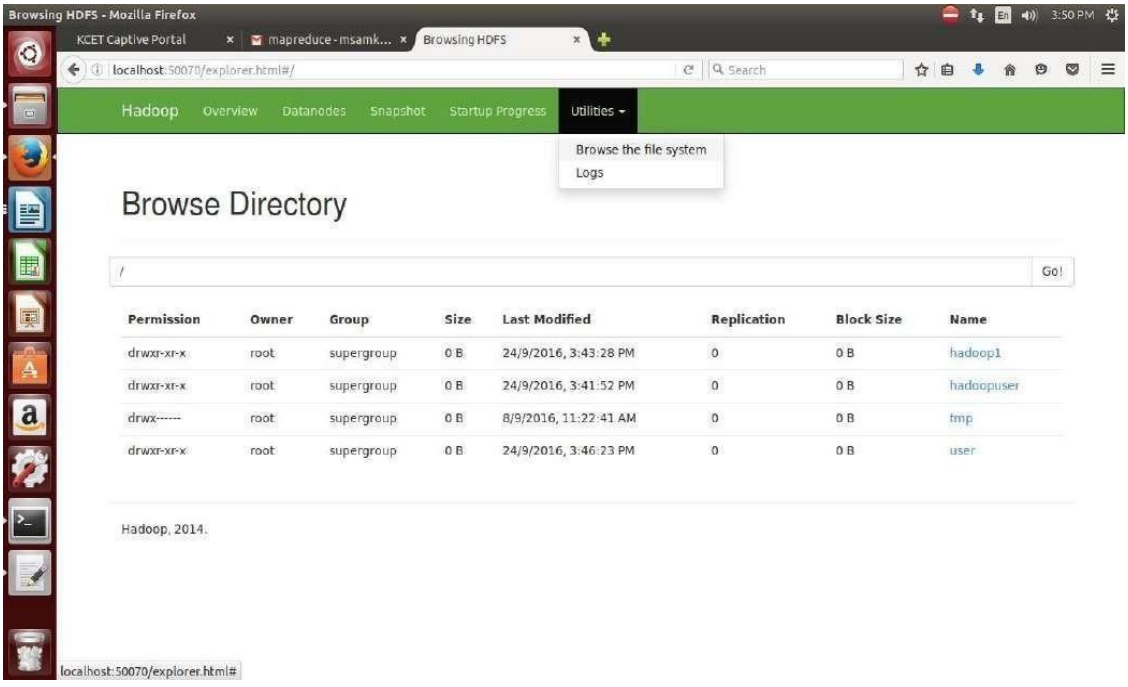
3 Cloud

3 Lab.

2 Grid

2 and

Step 6: Check the output in the Web UI at **http://localhost:50070.**In the Utilities tab select browse file system and select the correct user.The output is available inside the output folder named **user**



Step 7: To Delete an output folder

Syntax:

hdfs dfs -rm -R outputpath

Output:

root@a4cseh160:/#hdfs dfs -rm -R /user/out.txt

Result:

Thus the implemented Map Reduce example such as Word Count program on an file which will count the number of times a word repeats in the given file.

EXNO:8

DATE: **Creating and Executing your First container using Docker**

AIM:

To Creating and Executing your First container using Docker

Procedure:

1. Docker is fast. Unlike a virtual machine, your application can start in a few seconds and stop just as quickly.
2. Docker is multi-platform. You can launch your container on any system.
3. Containers can be built and destroyed faster than a virtual machine.
4. No more difficulties setting up your working environment. Once your Docker is configured, you will never have to reinstall your dependencies manually again. If you change computers or if an employee joins your company, you only have to give them your configuration.
5. You keep your work-space clean, as each of your environments will be isolated and you can delete them at any time without impacting the rest.
6. It will be easier to deploy your project on your server in order to put it online.

Now let's create your first application

- Now that you know what Docker is, it's time to create your first application!
- The purpose of this short tutorial is to create a Python program that displays a sentence. This program will have to be launched through a Dockerfile.
- You will see, it's not very complicated once you understand the process.

Note: You will not need to install Python on your computer. It will be up to the Docker environment to contain Python in order to execute your code.

1. Install Docker on your machine

For Ubuntu:

First, update your packages:

$ sudo apt update

Next, install docker with apt-get:

$ sudo apt install docker.io

Finally, verify that Docker is installed correctly:

$ sudo docker run hello-world

- For MacOSX: you can follow this link.
- For Windows: you can follow this link.

2. Create your project

In order to create your first Docker application, I invite you to create a folder on your computer. It must contain the following two files:

- A 'main.py' file (python file that will contain the code to be executed).

- A 'Dockerfile' file (Docker file that will contain the necessary instructions to create the environment).

Normally you should have this folder architecture:

```
.
├── Dockerfile
└── main.py
0 directories, 2 files
```

3. Edit the Python file

You can add the following code to the ‘main.py’ file:

```
#!/usr/bin/env python3

print("Docker is magic!")
```

Nothing exceptional, but once you see “Docker is magic!” displayed in your terminal you will know that your Docker is working.

3. Edit the Docker file

Some theory: the first thing to do when you want to create your Dockerfile is to ask yourself what you want to do. Our goal here is to launch Python code.

- To do this, our Docker must contain all the dependencies necessary to launch Python. A linux (Ubuntu) with Python installed on it should be enough.
- The first step to take when you create a Docker file is to access the DockerHub website. This site contains many pre-designed images to save your time (for example: all images for linux or code languages).
- In our case, we will type ‘Python’ in the search bar. The first result is the official image created to execute Python. Perfect, we’ll use it!

```
# A dockerfile must always start by importing the base image.
# We use the keyword 'FROM' to do that.
# In our example, we want import the python image.
# So we write 'python' for the image name and 'latest' for the version.
FROM python:latest

# In order to launch our python code, we must import it into our image.
# We use the keyword 'COPY' to do that.
# The first parameter 'main.py' is the name of the file on the host.
# The second parameter '/' is the path where to put the file on the image.
# Here we put the file at the image root folder.
COPY main.py /

# We need to define the command to launch when we are going to run the image.
# We use the keyword 'CMD' to do that.
# The following command will execute "python ./main.py".
CMD [ "python", "./main.py" ]
```

4. Create the Docker image

Once your code is ready and the Dockerfile is written, all you have to do is create your image to contain your application.

```
$ docker build -t python-test .
```

The ’-t’ option allows you to define the name of your image. In our case we have chosen ’python-test’ but you can put what you want.

5. Run the Docker image

- Once the image is created, your code is ready to be launched.
- $ docker run python-test
- You need to put the name of your image after 'docker run'.
- There you go, that's it. You should normally see "Docker is magic!" displayed in your terminal.
- Code is available
- If you want to retrieve the complete code to discover it easily or to execute it, I have put it at your disposal on my GitHub.
  - ➔ GitHub: Docker First Application example
- Useful commands for Docker
- Before I leave you, I have prepared a list of commands that may be useful to you on Docker.

• List your images.
$ docker image ls
• Delete a specific image.
$ docker image rm [image name]
• Delete all existing images.
$ docker image rm $(docker images -a -q)
• List all existing containers (running and not running).
$ docker ps -a
• Stop a specific container.
$ docker stop [container name]
• Stop all running containers.
$ docker stop $(docker ps -a -q)
• Delete a specific container (only if stopped).
$ docker rm [container name]
• Delete all containers (only if stopped).
$ docker rm $(docker ps -a -q)
• Display logs of a container.
$ docker logs [container name]

This is a very simple tutorial for getting started with Docker. I'll try to keep this as simple as possible. In this, we are going to build a basic Flask application and dockerize the application. By the end of this tutorial, you'll get familiar with Docker and a few Docker commands.

```
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/balaji/.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/home/balaji/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/balaji/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/balaji/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  config      Manage Docker configs
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  deploy      Deploy a new stack or update an existing stack
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
```

Docker man page

Overview of Docker

Docker is a platform for developing and running applications. It automates the deployment of applications. It's a tool for running applications in an **isolated** environment. Docker makes it easy to share an application with all of its dependencies across different environments.

Docker Image

Image is a template for creating an environment. The Docker image contains the Operating System, Software, and application code. These are all packaged in a single file. Images are defined with the Dockerfile.

Dockerfile

Dockerfile is built into a docker image

The Dockerfile contains the steps that are needed to package your application(to create the image). These steps include configuring the Operating system, install the required packages or software, copy the files from one place to another.

Create the Flask application

Let's dive into code

Flask is the micro web framework for building small web applications.

The directory structure is as follows:

```
flaskapp/
├──── Dockerfile
├──── app.py
├──── requirements.txt
```

Create a directory with the name of your choice. The command for creating a directory in Linux is,

```
$mkdir flaskapp
```

Navigate inside the directory you created. Now create a file named app.py and copy the following code to it.

app.py

```
from flask import Flask
app = Flask(__name__)@app.route('/')
def hello_world():
    return 'Hello, This is my first Docker app!'if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

This is a simple Hello world Flask app. If you want to learn about flask in detail, then visit this page and explore about flask. Let's concentrate more on Docker now.

Then create the requirements.txt file. Add the following line to it.

requirements.txt

```
Flask==1.1.2
```

Now we are going to create a Dockerfile that dockerizes this application.

Write the Docker File

The Dockerfile should not have any extension.

If you haven't downloaded the docker yet. Get the Docker from here. Detailed installation instructions can be found here.

Create a file called Dockerfile. This must start with capital letter D.

Dockerfile

```
FROM alpine:3.11RUN apk add --update pythonRUN apk add --update py-pipCOPY ./requirements.txt /app/requirements.txtCOPY . /appWORKDIR /appRUN pip install -r requirements.txtEXPOSE 5000CMD python app.py
```

Let’s look at the instructions line by line to have a better understanding

```
FROM alpine:3.11
```

This is our base image. There are a lot of images available in the Dockerhub. We choose alpine since it’s one of a lightweight image.

The **FROM** allows us to initialize the build over the base image. The number after the colon is the version number. A valid Dockerfile always starts with FROM keyword.

```
RUN apk --update add python
RUN apk add --update py-pip
```

These two lines are used to install the python and the pip package respectively. The **RUN** instruction will execute a new layer on top of the current image. This instruction is used to install packages and creating new directories.

```
COPY ./requirements.txt /app/requirements.txtCOPY . /app
```

The **COPY** instruction is used to copy the files from source(.) to the destination(/app). The ‘.’ represents the current directory. This basically copies the flask app into the image.

```
WORKDIR /app
```

The **WORKDIR** sets the working directory.

```
RUN pip install -r requirements.txt
```

This reads the requirements.txt file and installs the specified packages one by one on the host.

```
EXPOSE 5000
```

The **EXPOSE** exposes a port that is used by Flask. When you run the image you’ll get a container that container will run on this port.

```
CMD python app.py
```

CMD is the command that is executed when you start a container. Here, you are using the command to run your Python application. There can be only one CMD per Dockerfile. If you specify more than one, then the last `CMD` will take effect.

When you start a container this command is executed. There should be only one **CMD** instruction in the Dockerfile. If there is more than one, it'll execute the last instruction.

Build the docker image

Enough of information. Let's run the application

To build the Docker image, execute this command in the terminal in the directory we created.

```
$docker build -t flask-app .
```

The `-t` is used to name your image flask-app. The `.` , in the end, represents the current directory. When you execute this for the first time, it'll have to download all of the layers that make up to build the image. After that, it'll use the cache. If you're not an administrator, try to run the command with the `sudo`.

```
Sending build context to Docker daemon  4.096kB
Step 1/9 : FROM alpine:3.11
 ---> f70734b6a266
Step 2/9 : RUN apk add --update python
 ---> Using cache
 ---> e9800bc501b3
Step 3/9 : RUN apk add --update py-pip
 ---> Using cache
 ---> b48cca251e6b
Step 4/9 : COPY ./requirements.txt /app/requirements.txt
 ---> Using cache
 ---> b2c052f9bdc3
Step 5/9 : COPY . /app
 ---> e50bd7a55434
Step 6/9 : WORKDIR /app
 ---> Running in 8488a87861d7
Removing intermediate container 8488a87861d7
 ---> d0d28ec2d9ff
Step 7/9 : RUN pip install -r requirements.txt
 ---> Running in 05ad40e31889
Collecting Flask==1.1.2 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/
Collecting Jinja2>=2.10.1 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/30/9e/f663a2aa66a09d838042ae1a2c5659828bbb9b41ea3a6efa20a20fd92b121/
Collecting click>=5.1 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/
Collecting Werkzeug>=0.15 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/
Collecting itsdangerous>=0.24 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/b9/2e/64db92e53b86efccfaea71321f597fa2e1b2bd3853d8ce658568f7a13094/
Installing collected packages: MarkupSafe, Jinja2, click, Werkzeug, itsdangerous, Flask
  Running setup.py install for MarkupSafe: started
    Running setup.py install for MarkupSafe: finished with status 'done'
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 itsdangerous-1.1.0
Removing intermediate container 05ad40e31889
 ---> b320b914c2c5
Step 8/9 : EXPOSE 5000
 ---> Running in ffbcaffcc010
Removing intermediate container ffbcaffcc010
 ---> 5916f995523c
Step 9/9 : CMD python app.py
 ---> Running in ebe11a334937
Removing intermediate container ebe11a334937
 ---> 785b60cf0cdb
Successfully built 785b60cf0cdb
Successfully tagged flask-app:latest
```

docker build log

To run the application, execute this command

```
$docker run -p 5000:5000 flask-app
```

The -p is used to map the *port running inside the container to your host. Here we're mapping the port 5000. The value before the colon represents the port running on your host and the value after colon represents the port running inside the container.*

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 853-721-745
```

Now, navigate to http://localhost:5000/ in your browser, you'll see the "Hello, This is my first Docker app!" in the window.

Container

To know the container id enter the following command in your terminal.

```
$docker ps
```

This lists all the running containers in the Docker engine. This has information about containers like its ID, created time, status and on which port it's running.

```
$docker ps -a
```

This will list the containers that have been stopped as well.

```
$docker stop [container name]
```

This will stop all the running containers.

Result:

Thus the program Creating and Executing First container using Docker is run successfully

EXNO:9 **Run a container from Docker Hub**

DATE:

AIM:

To run a container from docker hub

PROCEDURE:

Step 1: Get the sample application. If you have git, you can clone the repository for the sample application. ...

Step 2: Explore the Dockerfile. ...

Step 3: Build your first image. ...

Step 4: Run your container. ...

Step 5: Verify that your container is running.

Step 1: Sign up for a Docker account

Start by creating a Docker ID.

A Docker ID grants you access to Docker Hub repositories and allows you to explore images that are available from the community and verified publishers. You'll also need a Docker ID to share images on Docker Hub.

Step 2: Create your first repository

To create a repository:

1. Sign in to Docker Hub.
2. Select **Create a Repository** on the Docker Hub welcome page.
3. Name it **<your-username>/my-private-repo**.
4. Set the visibility to **Private**.
5. Select **Create**.

You've created your first repository.

Step 3: Download and install Docker Desktop

You need to download Docker Desktop to build, push, and pull container images.

1. Download and install Docker Desktop.
2. Sign in to Docker Desktop using the Docker ID you created in step one.

Step 4: Pull and run a container image from Docker Hub

1. In your terminal, run `docker pull hello-world` to pull the image from Docker Hub. You should see output similar to:

```
$ docker pull hello-world

Using default tag: latest

latest: Pulling from library/hello-world

2db29710123e: Pull complete

Digest:
sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dc
a625

Status: Downloaded newer image for hello-world:latest

docker.io/library/hello-world:latest
```

9. Run `docker run hello-world` to run the image locally. You should see output similar to:

```
$ docker run hello-world

Hello from Docker!

This message shows that your installation appears to be working
correctly.

To generate this message, Docker took the following steps:

 1. The Docker client contacted the Docker daemon.

 2. The Docker daemon pulled the "hello-world" image from the
Docker Hub.

 (amd64)

 3. The Docker daemon created a new container from that image
which runs the

 executable that produces the output you are currently reading.

 4. The Docker daemon streamed that output to the Docker client,
which sent

 it to your terminal.

To try something more ambitious, you can run an Ubuntu container
with:

 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker
ID:

 https://hub.docker.com/
```

```
28.  For more examples and ideas, visit:
29.   https://docs.docker.com/get-started/
```

Step 5: Build and push a container image to Docker Hub from your computer

1. Start by creating a Dockerfile to specify your application as shown below:

```
2. # syntax=docker/dockerfile:1
3. FROM busybox
4. CMD echo "Hello world! This is my first Docker image."
```

5. Run `docker build -t <your_username>/my-private-repo .` to build your Docker image.
6. Run `docker run <your_username>/my-private-repo` to test your Docker image locally.
7. Run `docker push <your_username>/my-private-repo` to push your Docker image to Docker Hub. You should see output similar to:

```
[redacted] cat > Dockerfile <<EOF
FROM busybox
CMD echo "Hello world! This is my first Docker image."
EOF
[redacted] docker build -t mobythewhale/my-private-repo .
[+] Building 1.2s (5/5) FINISHED
 => [internal] load build definition from Dockerfile                    0.0s
 => => transferring dockerfile: 110B                                    0.0s
 => [internal] load .dockerignore                                       0.0s
 => => transferring context: 2B                                         0.0s
 => [internal] load metadata for docker.io/library/busybox:latest       1.2s
 => CACHED [1/1] FROM docker.io/library/busybox@sha256:a9286defaba7b3a519  0.0s
 => exporting to image                                                  0.0s
 => => exporting layers                                                 0.0s
 => => writing image sha256:dcdb1fd928bfb257bfc0122ea47accd911a3a386ce618  0.0s
 => => naming to docker.io/mobythewhale/my-private-repo                 0.0s
[redacted] docker run mobythewhale/my-private-repo
Hello world! This is my first Docker image.
[redacted] docker push mobythewhale/my-private-repo
The push refers to repository [docker.io/mobythewhale/my-private-repo]
d2421964bad1: Layer already exists
latest: digest: sha256:7604fbf8eeb03d866fd005fa95cdbb802274bf9fa51f7dafba6658294
efa9baa size: 526
[redacted]
```

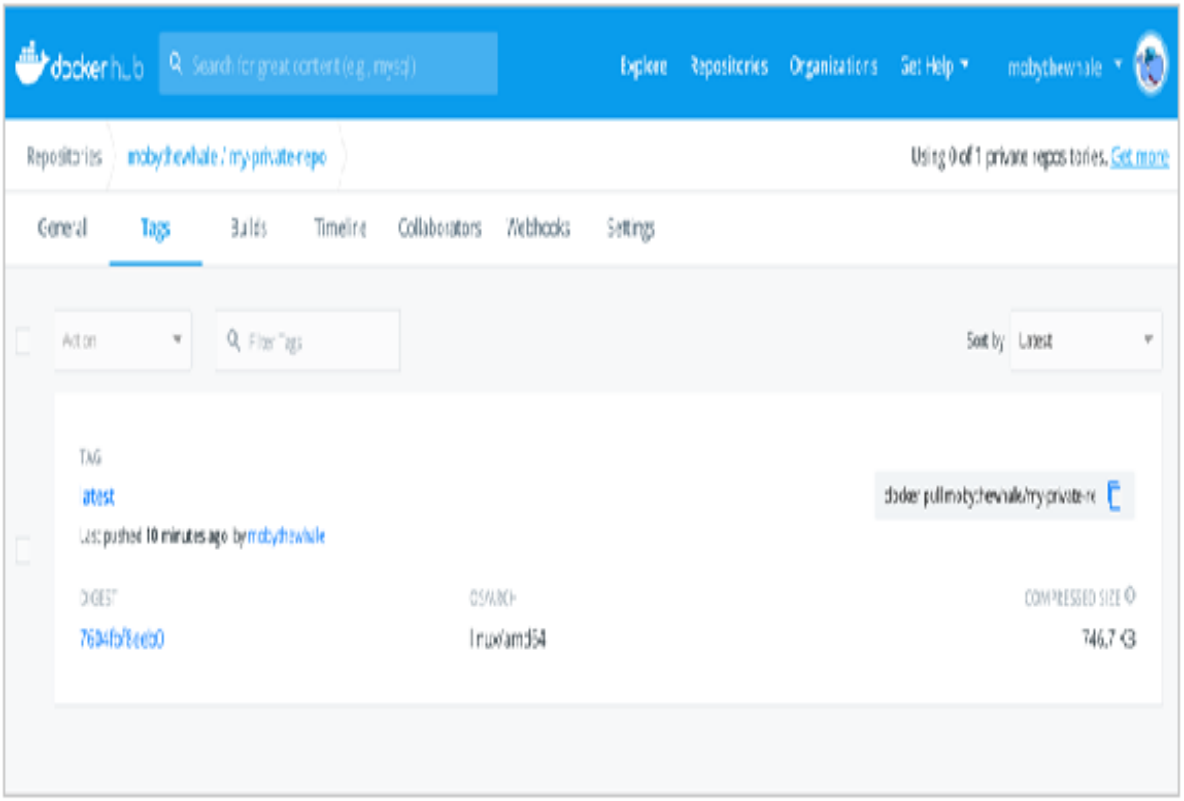
Note

You must be signed in to Docker Hub through Docker Desktop or the command line, and you must also name your images correctly, as per the above steps.

Result

Thus the Run a container from Docker Hub is run successfully and output is verified

8. Your repository in Docker Hub should now display a new `latest` tag under **Tags**:



You've successfully:

- Signed up for a Docker account
- Created your first repository
- Pulled an existing container image from Docker Hub
- Built your own container image on your computer
- Pushed it successfully to Docker Hub

Next steps

- Create an organization to use Docker Hub with your team.
- Automatically build container images from code through builds.
- Explore official & publisher images.
- Upgrade your subscription to push additional private Docker images to Docker Hub.

- Docker Desktop
- Docker Extensions
- Docker Engine
- Docker Build
- Docker Compose

Docker Hub

Result

Thus the Run a container from Docker Hub is run successfully and output is verified

CONTENT BEYOND TOPICS

Ex No: 10
Date :

KVM AND OPENSTACK INSTALLATION

Aim:

To install the KVM and Opensatck in Ubuntu 14.04 version an d creation of virtual amchine.

Mandatory prerequisite:

1. Linux 64 bit Operating System (The commands mentioend are for Ubuntu Linux Operating System latest version).

Installing KVM (Hypervisor for Virtualization)

1. Check if the Virtualization flag is enabled in BIOS

Run the command in terminal

egrep -c '(vmx|svm)' /proc/cpuinfo

If the result is any value higher than 0, then virtualization is enabled.

If the value is 0, then in BIOS enable Virtualization – Consult system administrator for this step.

2. To check if your OS is 64 bit,

Run the command in terminal

uname -m

If the result is x86_64, it means that your Operating system is 64 bit Operating system.

3. Few KVM packages are availabe with Linux installation.

To check this, run the command,

ls /lib/modules/{press tab}/kernel/arch/x86/kvm

nagarajan@JBL01**:~$ ls /lib/modules/4.4.0-21-generic/kernel/arch/x86/kvm**

The three files which are installed in your system will be displayed

kvm-amd.ko kvm-intel.ko kvm.ko

4. Install the KVM packages

1. Switch to root (Administrator) user

sudo -i

export http_proxy=http://172.16.0.3:8080

2. To install the packages, run the following commands,

apt-get update

apt-get install qemu-kvm
apt-get install libvirt-bin
apt-get install bridge-utils
apt-get install virt-manager
apt-get install qemu-system

5. To verify your installation, run the command

virsh -c qemu:///system list

it shows output

```
Id          Name            State
----------------------------
```

If Vms are running, then it shows name of VM. If VM is not runnign, the system shows blank output, whcih means your KVM installation is perfect.

6. Run the command

virsh --connect qemu:///system list --all

7. Working with KVM

run the command
virsh
version (this command displays version of software tools installed)
nodeinfo (this command displays your system information)
quit (come out of the system)

8. To test KVM installation - we can create Virtual machines but these machines are to be done in manual mode. Skipping this, Directly install Openstack.

Installation of Openstack

1. add new user named stack – This stack user is the adminstrator of the openstack services.
To add new user – run the command as root user.

adduser stack

2. run the command

apt-get install sudo -y || install -y sudo

3. Be careful in running the command – please be careful with the syntax. If any error in thsi following command, the system will crash beacause of permission errors.

echo "stack ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

4. Logout the system and login as stack user

5. Run the command (this installs git repo package)
Run in Root

export http_proxy=http://172.16.0.3:8080
sudo apt-get install git

6. Run the command (This clones updatesd version of dev-stack (which is binary auto-installer package of Openstack)

stack@JBL01:/$ **export http_proxy=http://172.16.0.3:8080**
stack@JBL01:/$ **export https_proxy=http://172.16.0.3:8080**
stack@JBL01:/$ **git config --global http.proxy $http_proxy**
stack@JBL01**:/$ git config --global https.proxy $http_proxy**
git clone http://git.openstack.org/openstack-dev/devstack
ls (this shows a folder named devstack)
cd devstack (enter into the folder)

7. create a file called **local.conf**. To do this run the command,
nano local.conf

stack@JBL01:/devstack$ **sudo nano local.conf**

8. In the file, make the following entry (Contact Your Network Adminstrator for doubts in these values)

[[local|localrc]]
FLOATING_RANGE=192.168.1.224/27
FIXED_RANGE=10.11.11.0/24
FIXED_NETWORK_SIZE=256
FLAT_INTERFACE=eth0
ADMIN_PASSWORD=root
DATABASE_PASSWORD=root
RABBIT_PASSWORD=root
SERVICE_PASSWORD=root
SERVICE_TOCKEN=root

9. Save this file

stack@JBL01:/devstack$ **sudo gedit stackrc**
Save this file
Change File Permission:
stack@JBL01:~$ **chown stack * -R**

10. Run the command (This installs Opentack)

./stack.sh

11. If any error occurs, then run the command for uninistallation

./unstack.sh

1. update the packages

apt-get update

2. Then reinstall the package

./stack.sh

12. Open the browser, http://IP address of your machine, you will get the openstack portal.

13. If you restart the machine, then to again start open stack

open terminal,

su stack

cd devstack

run ./rejoin.sh

14. Again you can access openstack services in the browser, http://IP address of your machine,

VIRTUAL MACHINE CREATION

Launch an instance

1. Log in to the dashboard

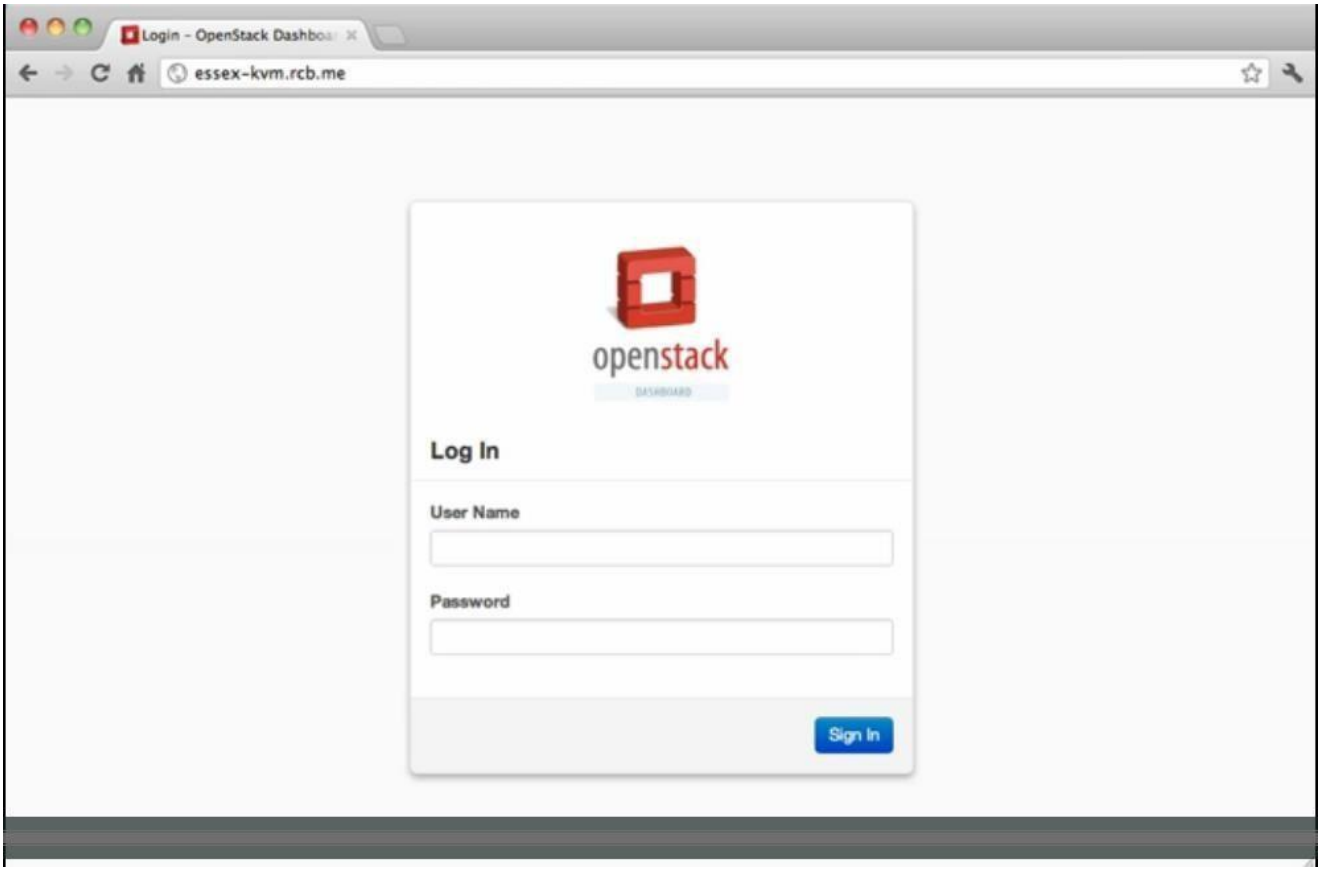
2. Select the appropriate project from the drop down menu at the top left.

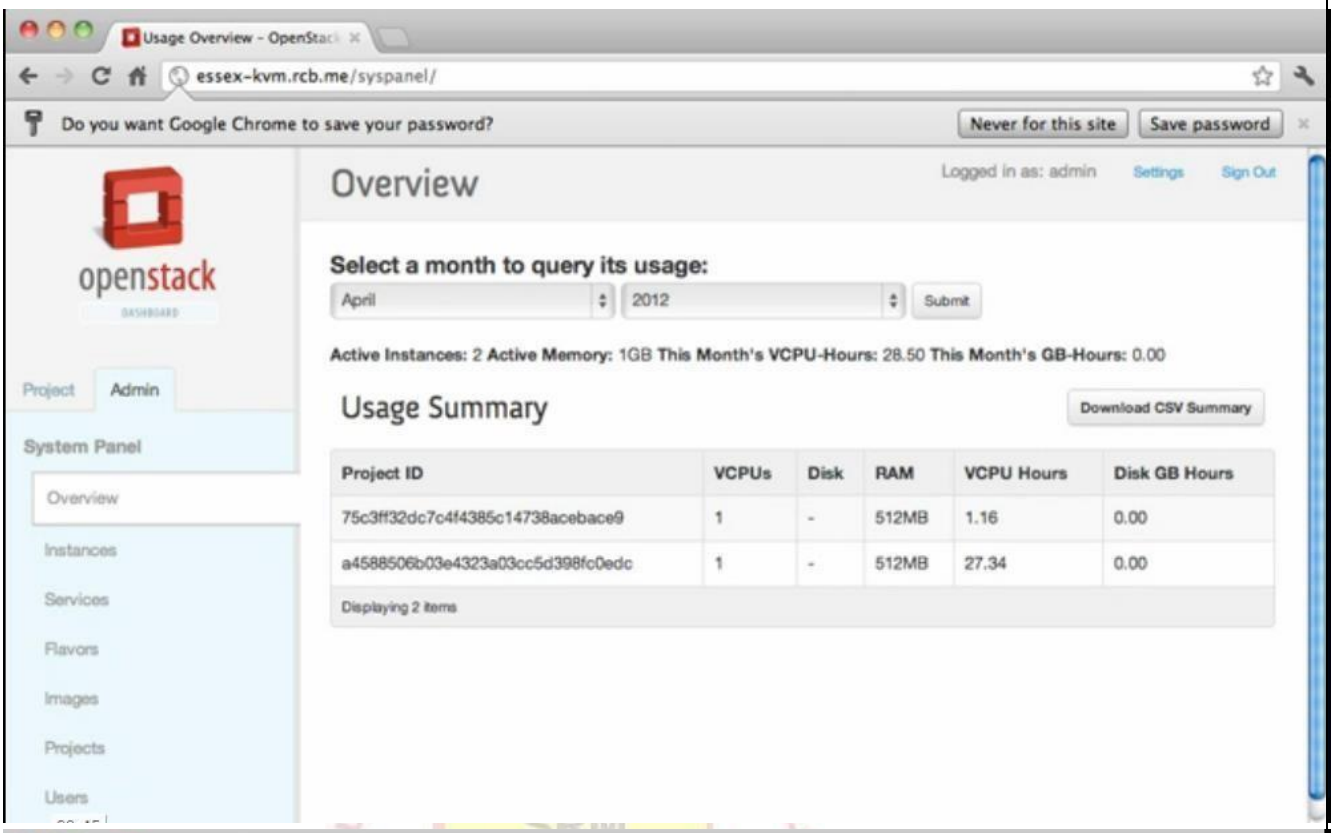
3. On the Project tab, open the Compute tab and click Instances category.

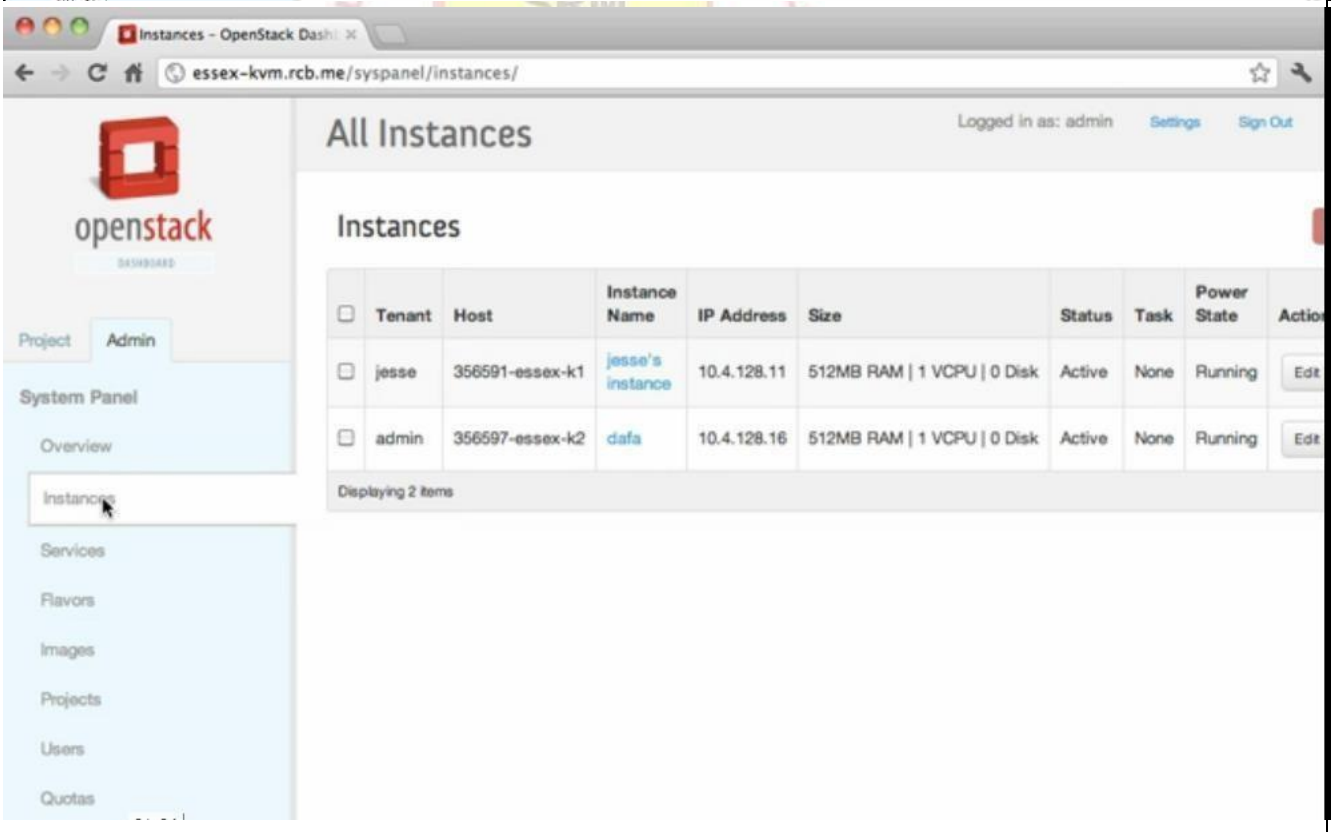
The dashboard shows the instances with its name, its private and floating IP addresses, size, status, task, power state, and so on.

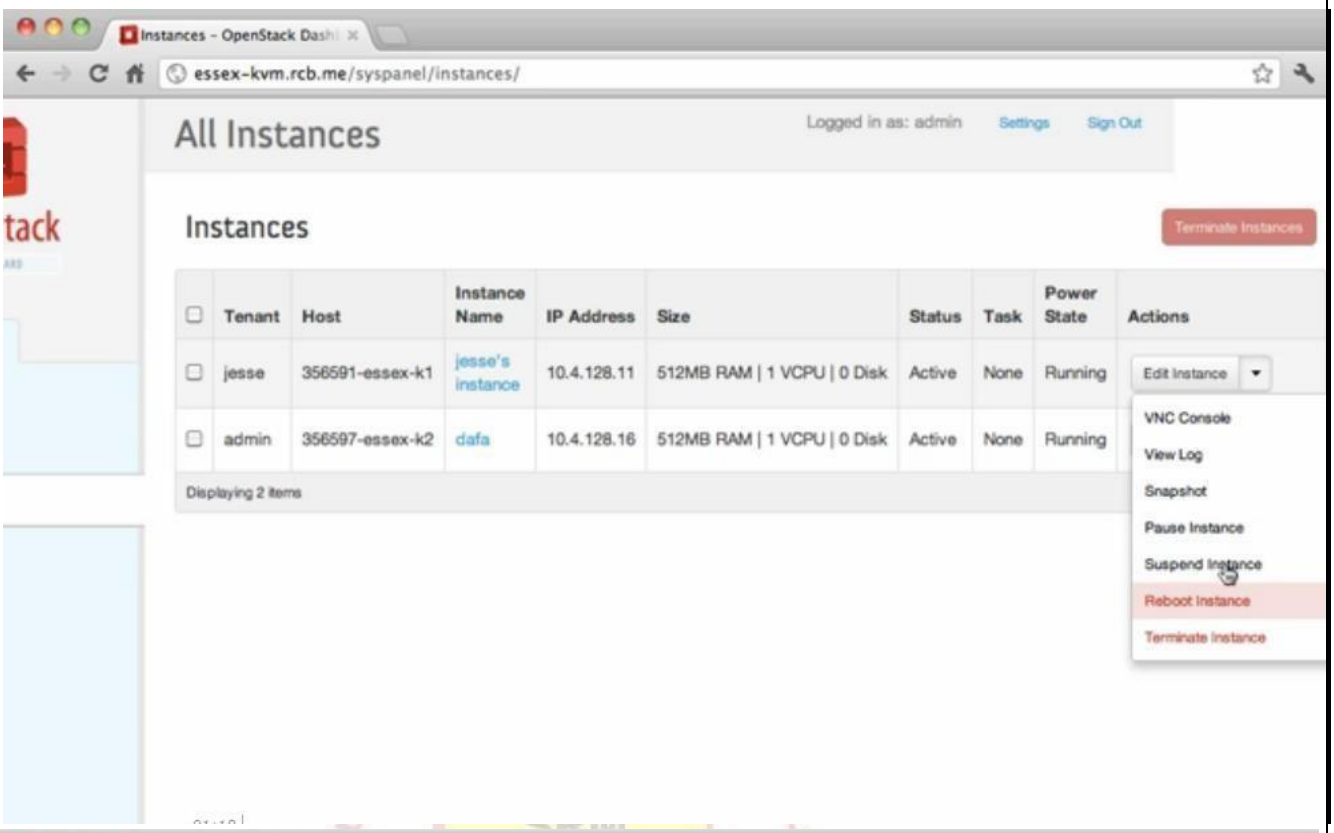
4. Click Launch Instance.

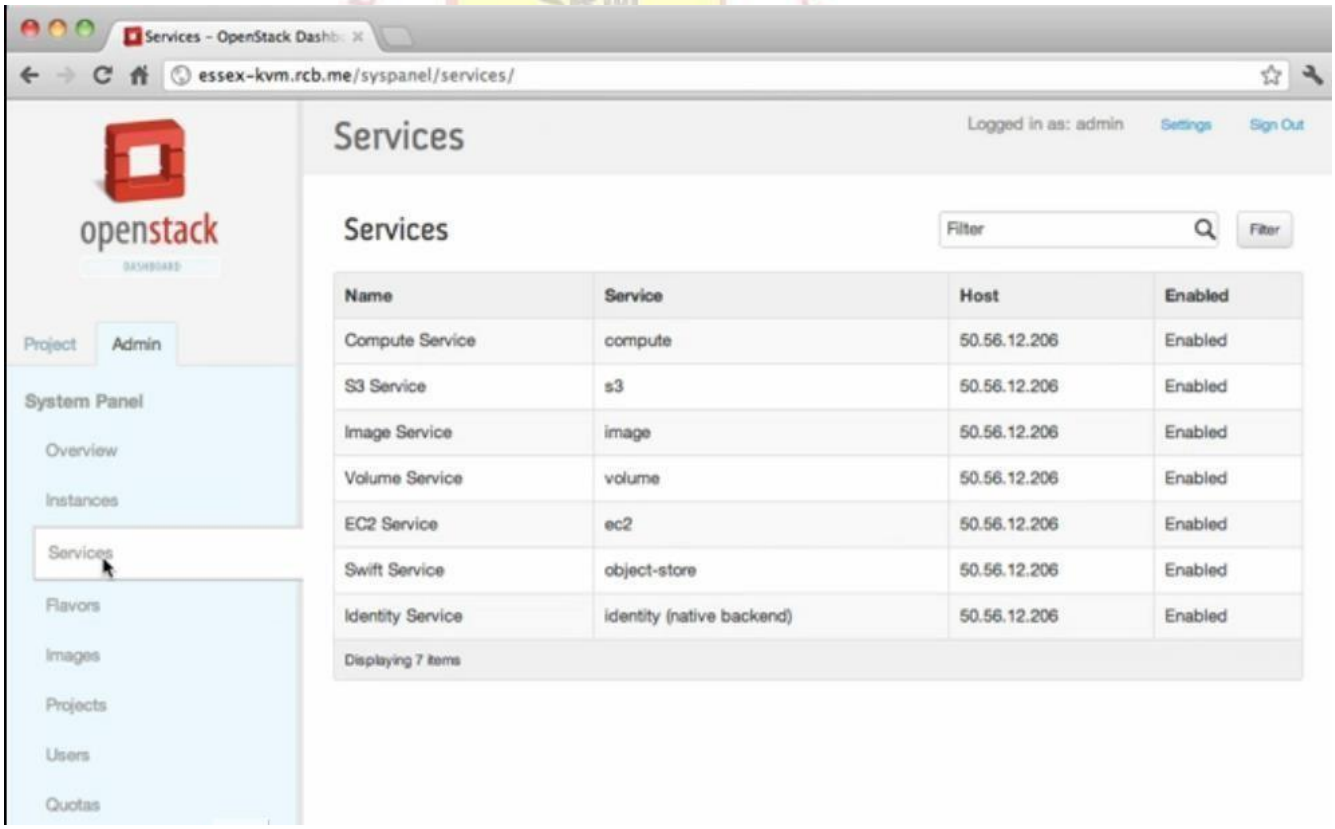
5. In the Launch Instance dialog box, specify the following values:

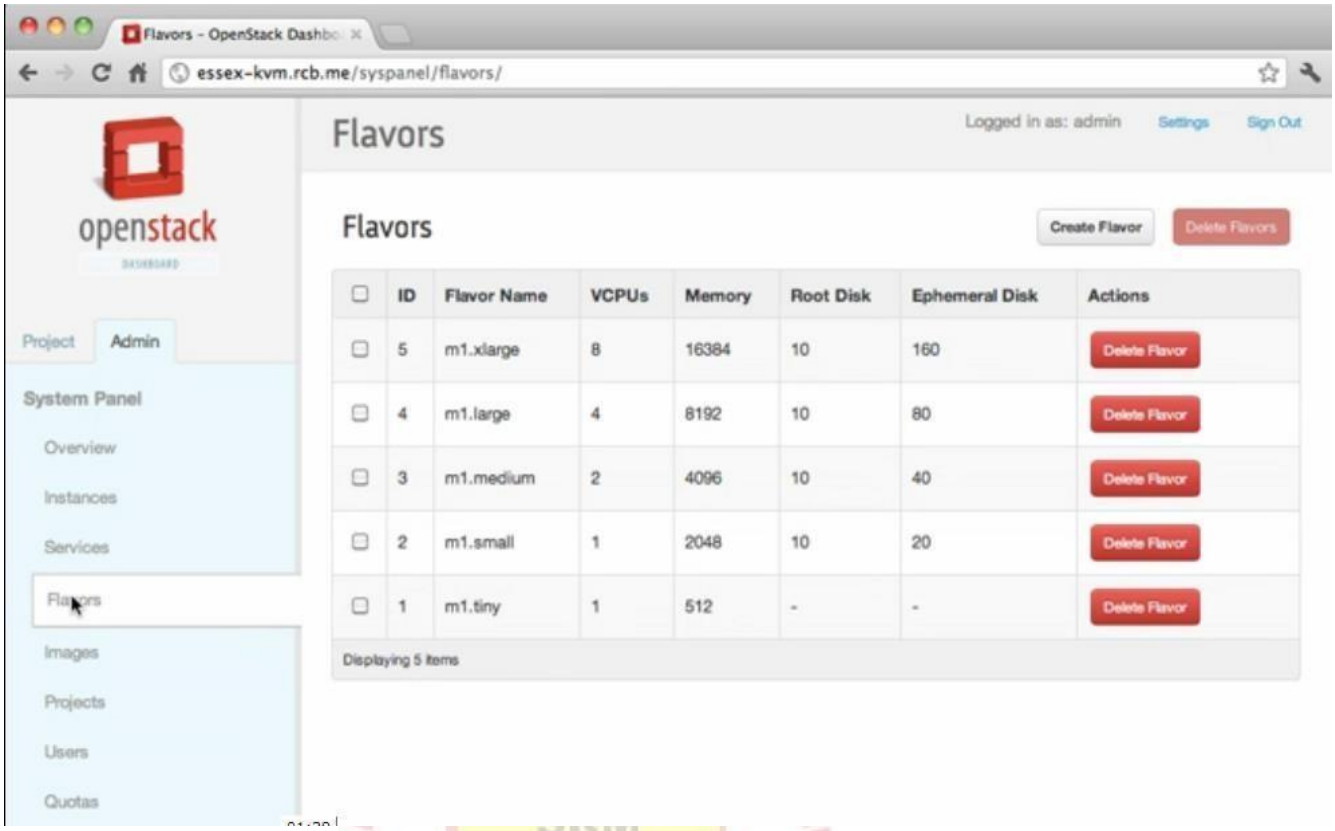

Login - OpenStack Dashboard
essex-kvm.rcb.me
openstack
Log In
User Name
Password
Sign In

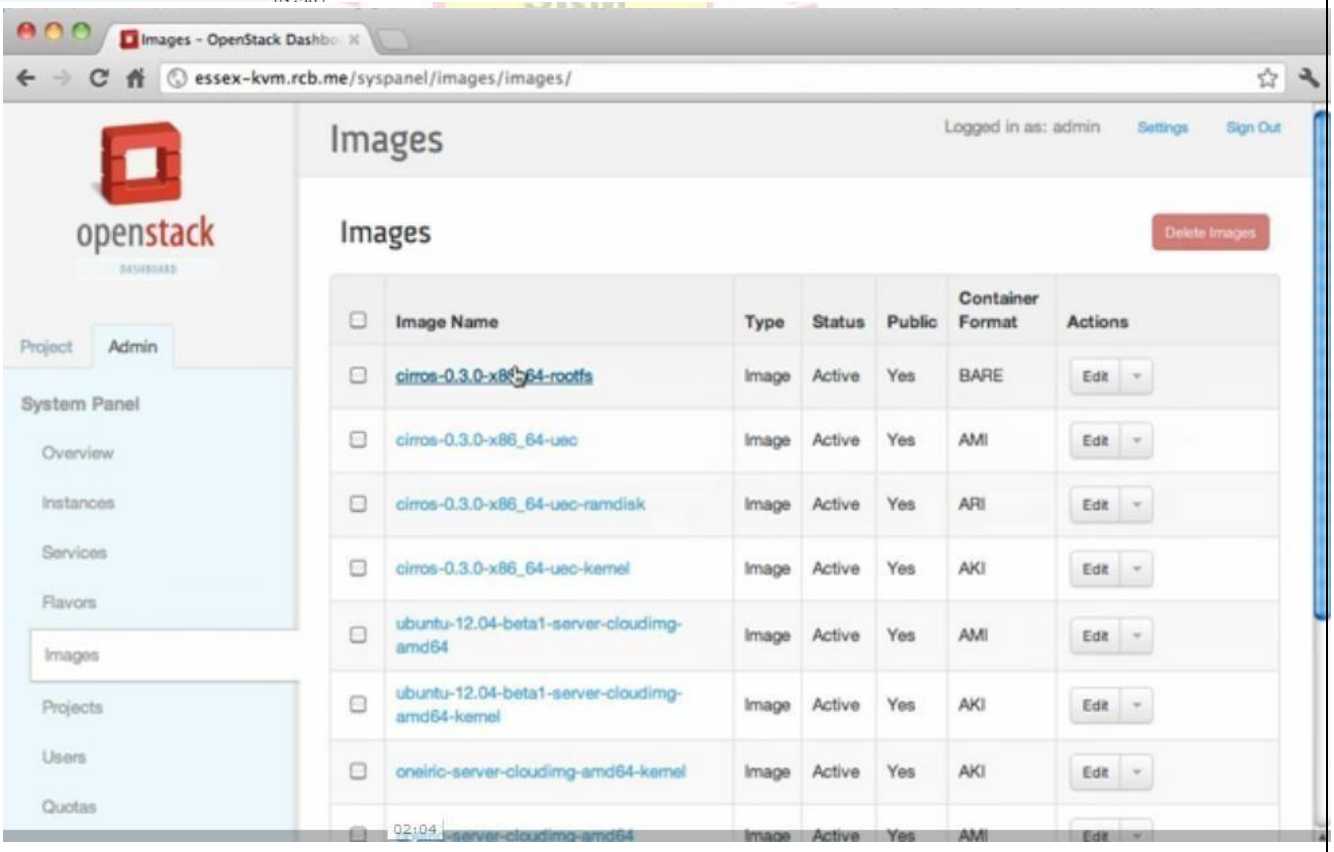

Usage Overview - OpenStack
essex-kvm.rcb.me/syspanel/
Do you want Google Chrome to save your password?
Never for this site
Save password
openstack
DASHBOARD
Overview
Logged in as: admin
Settings
Sign Out
Select a month to query its usage:
April
2012
Submit
Active Instances: 2 Active Memory: 1GB This Month's VCPU-Hours: 28.50 This Month's GB-Hours: 0.00
Usage Summary
Download CSV Summary
Project ID
VCPUs
Disk
RAM
VCPU Hours
Disk GB Hours
75c3ff32dc7c4f4385c14738acebace9
1
-
512MB
1.16
0.00
a4588506b03e4323a03cc5d398fc0edc
1
-
512MB
27.34
0.00
Displaying 2 items
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users

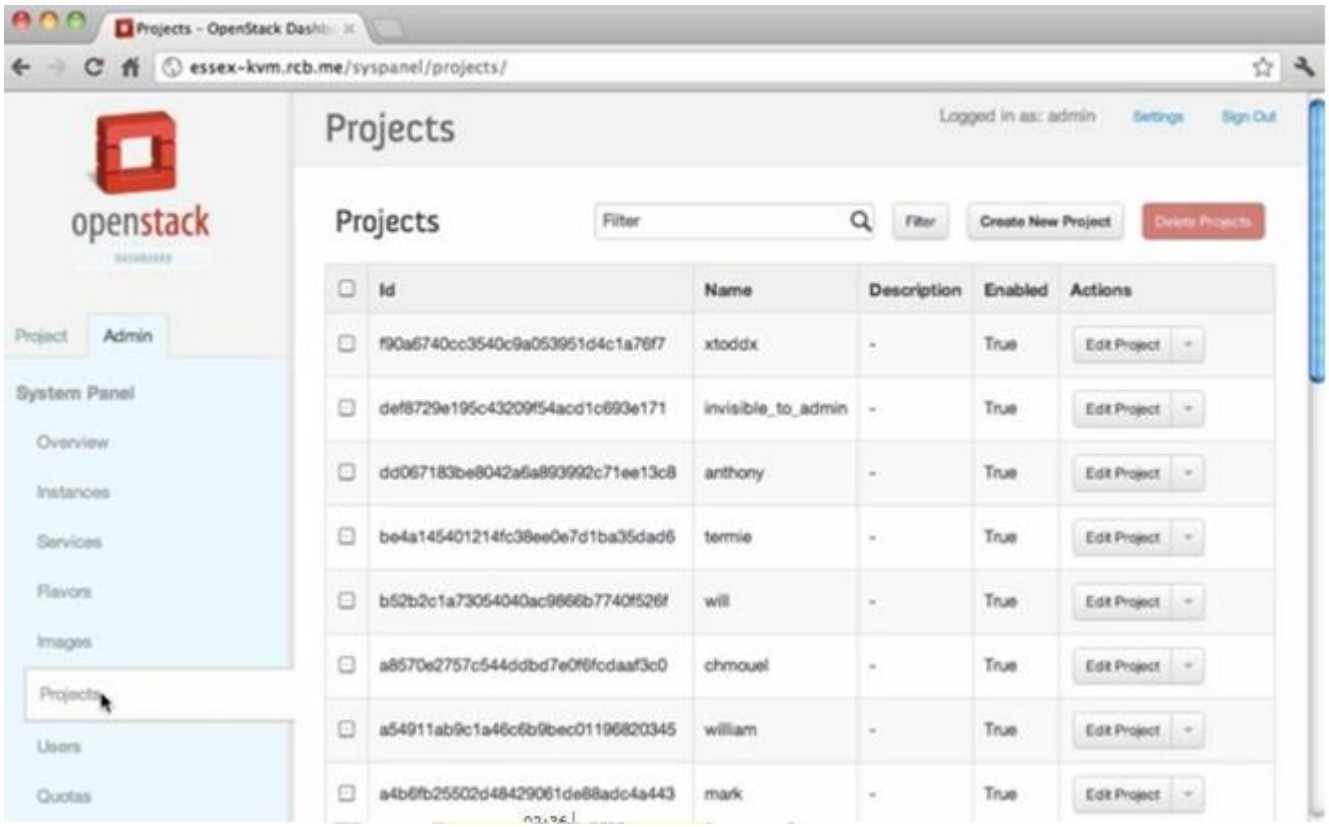

Instances - OpenStack Dash
essex-kvm.rcb.me/syspanel/instances/
openstack
DASHBOARD
All Instances
Logged in as: admin
Settings
Sign Out
Instances
Tenant
Host
Instance Name
IP Address
Size
Status
Task
Power State
jesse
356591-essex-k1
jesse's instance
10.4.128.11
512MB RAM | 1 VCPU | 0 Disk
Active
None
Running
admin
356597-essex-k2
dafa
10.4.128.16
512MB RAM | 1 VCPU | 0 Disk
Active
None
Running
Displaying 2 items
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Quotas

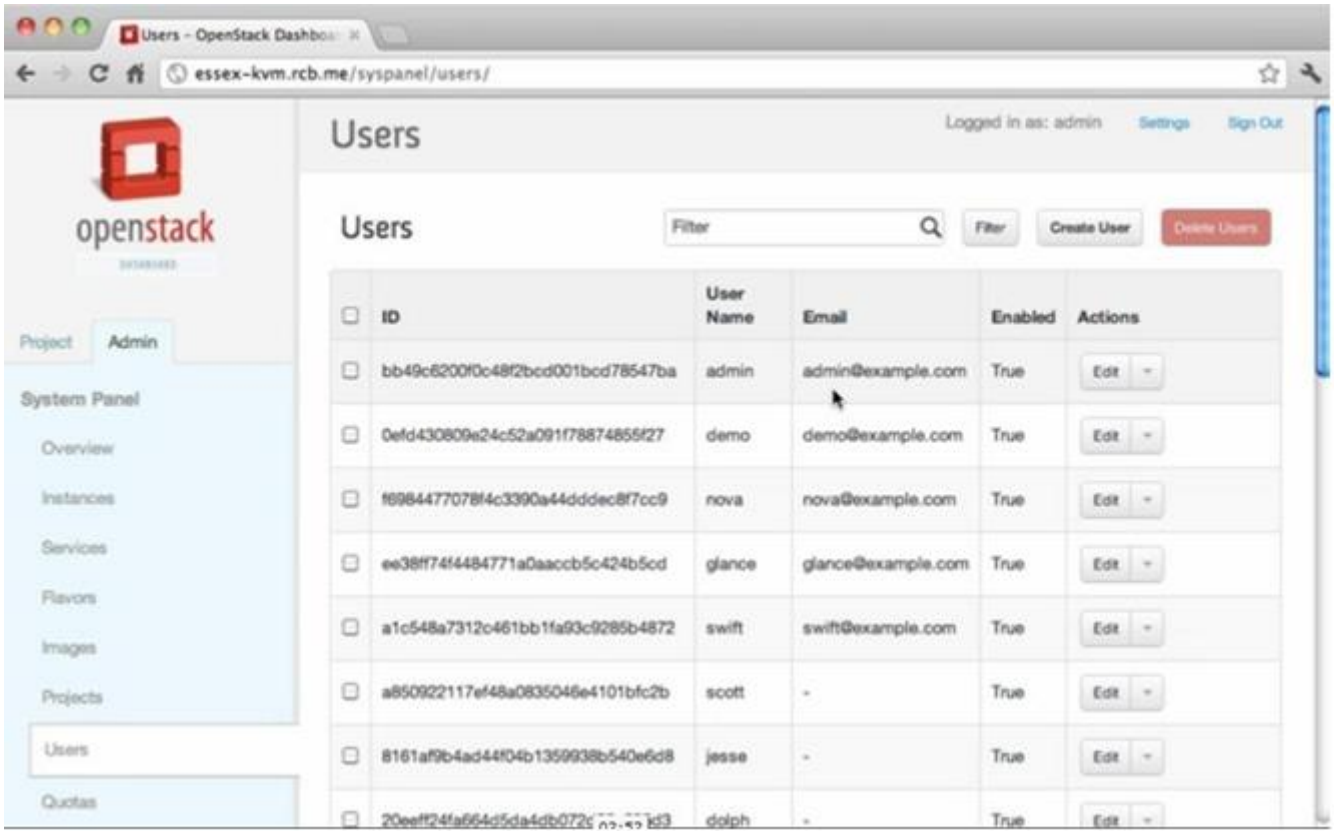

Instances - OpenStack Dash
essex-kvm.rcb.me/syspanel/instances/
All Instances
Logged in as: admin
Settings
Sign Out
Instances
Terminate Instances
Tenant
Host
Instance Name
IP Address
Size
Status
Task
Power State
Actions
jesse
356591-essex-k1
jesse's instance
10.4.128.11
512MB RAM | 1 VCPU | 0 Disk
Active
None
Running
Edit Instance
admin
356597-essex-k2
dafa
10.4.128.16
512MB RAM | 1 VCPU | 0 Disk
Active
None
Running
Displaying 2 items
VNC Console
View Log
Snapshot
Pause Instance
Suspend Instance
Reboot Instance
Terminate Instance

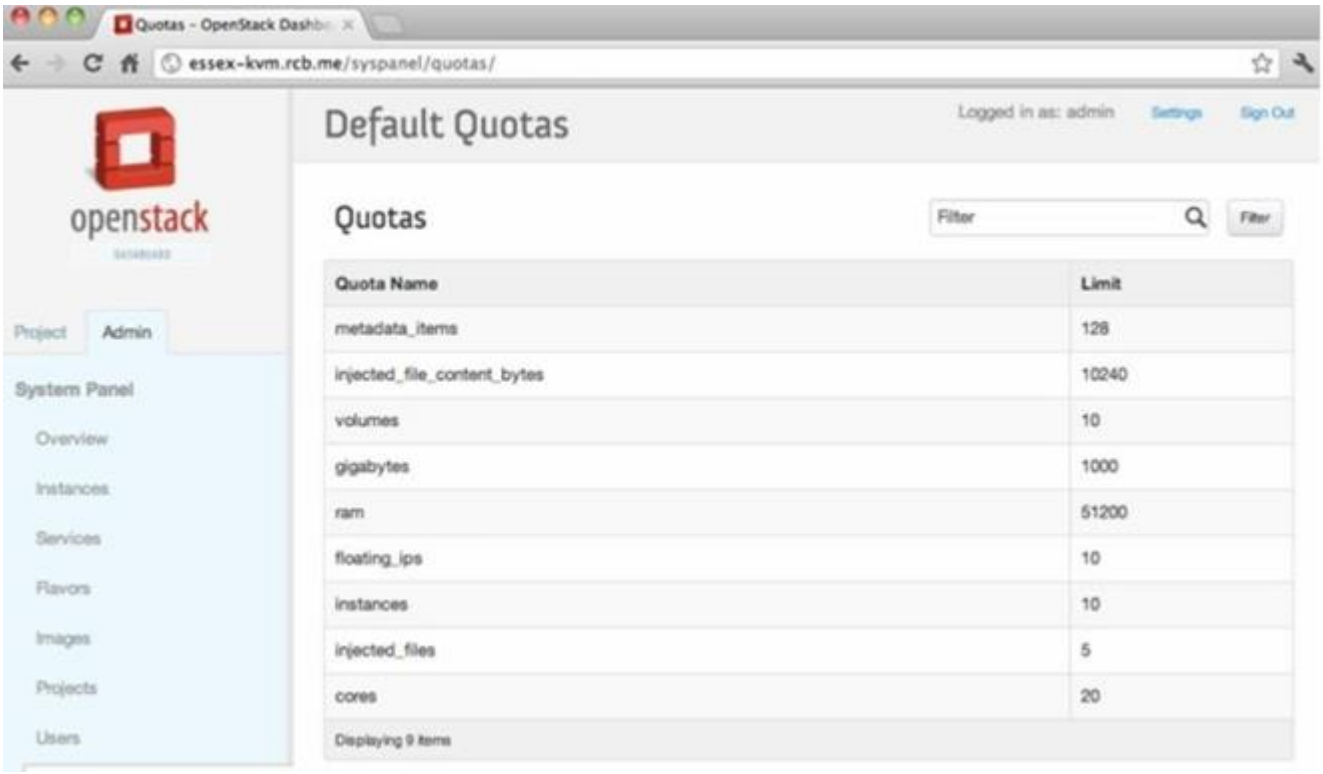

Services - OpenStack Dash
essex-kvm.rcb.me/syspanel/services/
Services
Logged in as: admin
Settings
Sign Out
openstack
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Quotas
Services
Filter
Filter
Name
Service
Host
Enabled
Compute Service
compute
50.56.12.206
Enabled
S3 Service
s3
50.56.12.206
Enabled
Image Service
image
50.56.12.206
Enabled
Volume Service
volume
50.56.12.206
Enabled
EC2 Service
ec2
50.56.12.206
Enabled
Swift Service
object-store
50.56.12.206
Enabled
Identity Service
identity (native backend)
50.56.12.206
Enabled
Displaying 7 items

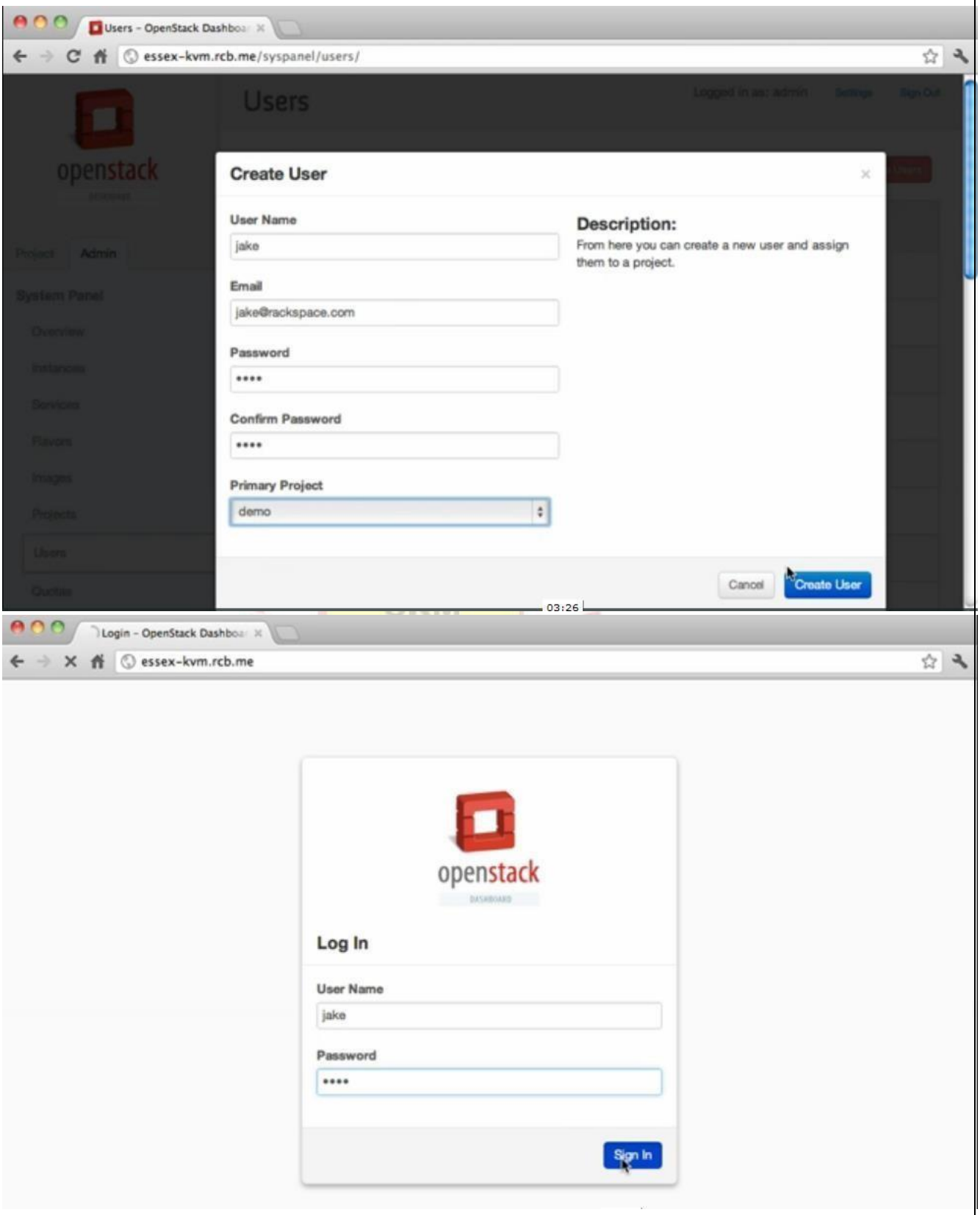

Flavors - OpenStack Dashbo
essex-kvm.rcb.me/syspanel/flavors/
openstack
DASHBOARD
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Quotas
Flavors
Logged in as: admin
Settings
Sign Out
Flavors
Create Flavor
Delete Flavors
ID
Flavor Name
VCPUs
Memory
Root Disk
Ephemeral Disk
Actions
5
m1.xlarge
8
16384
10
160
Delete Flavor
4
m1.large
4
8192
10
80
Delete Flavor
3
m1.medium
2
4096
10
40
Delete Flavor
2
m1.small
1
2048
10
20
Delete Flavor
1
m1.tiny
1
512
-
-
Delete Flavor
Displaying 5 items

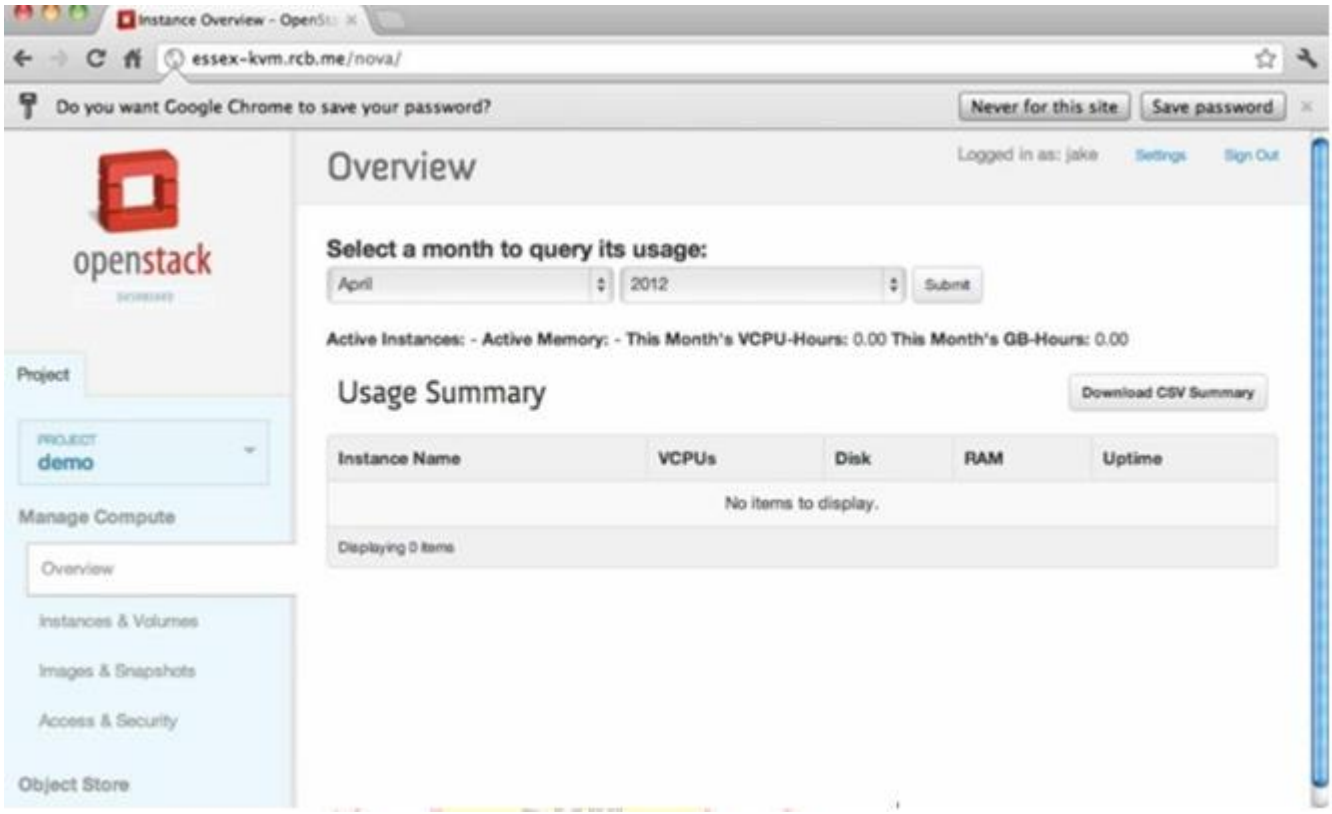

Images - OpenStack Dashbo
essex-kvm.rcb.me/syspanel/images/images/
openstack
DASHBOARD
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Quotas
Images
Logged in as: admin
Settings
Sign Out
Images
Delete Images
Image Name
Type
Status
Public
Container Format
Actions
cirros-0.3.0-x86_64-rootfs
Image
Active
Yes
BARE
Edit
cirros-0.3.0-x86_64-uec
Image
Active
Yes
AMI
Edit
cirros-0.3.0-x86_64-uec-ramdisk
Image
Active
Yes
ARI
Edit
cirros-0.3.0-x86_64-uec-kernel
Image
Active
Yes
AKI
Edit
ubuntu-12.04-beta1-server-cloudimg-amd64
Image
Active
Yes
AMI
Edit
ubuntu-12.04-beta1-server-cloudimg-amd64-kernel
Image
Active
Yes
AKI
Edit
oneiric-server-cloudimg-amd64-kernel
Image
Active
Yes
AKI
Edit

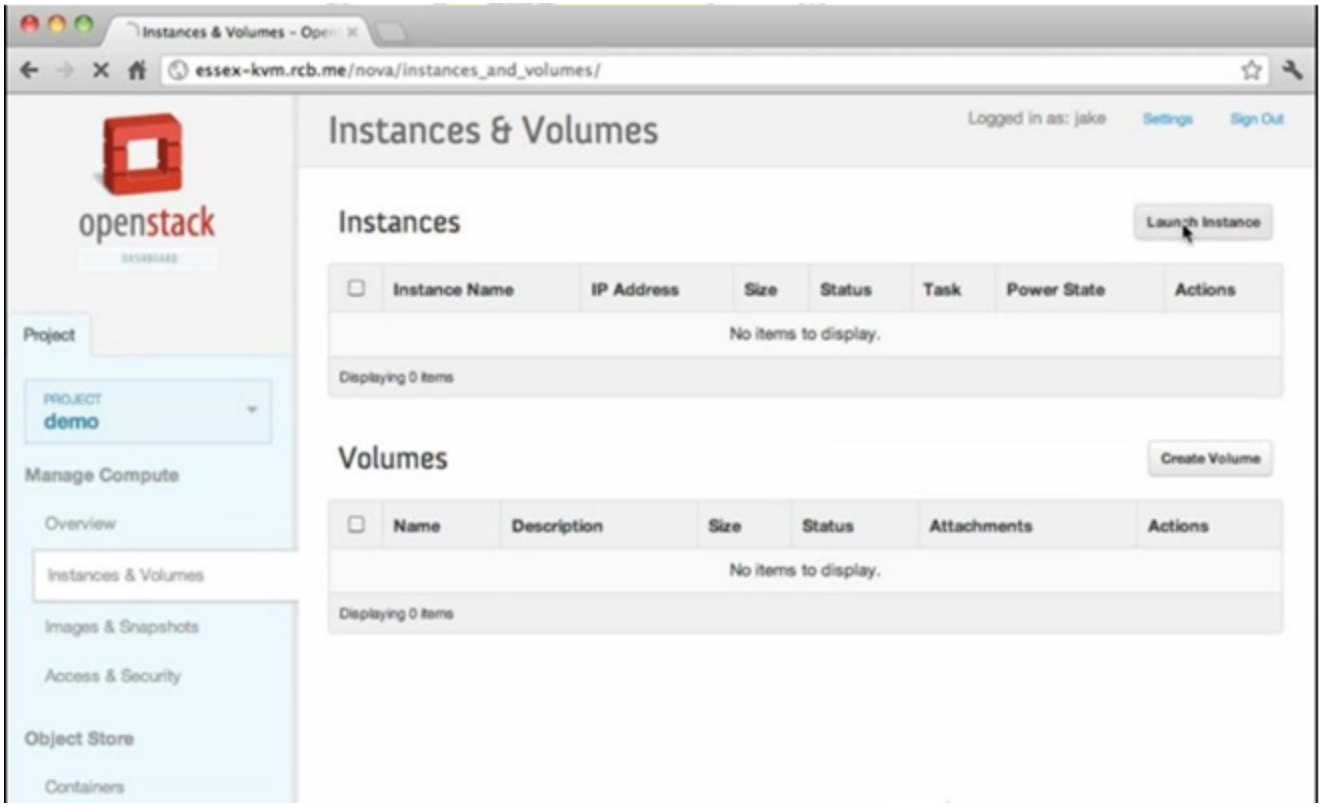

Projects - OpenStack Dashb
essex-kvm.rcb.me/syspanel/projects/
openstack
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Quotas
Projects
Logged in as: admin
Settings
Sign Out
Projects
Filter
Filter
Create New Project
Delete Projects
Id
Name
Description
Enabled
Actions
f90a6740cc3540c9a053951d4c1a76f7
xtoddx
-
True
Edit Project
def8729e195c43209f54acd1c693e171
invisible_to_admin
-
True
Edit Project
dd067183be8042a6a893992c71ee13c8
anthony
-
True
Edit Project
be4a145401214fc38ee0e7d1ba35dad6
termie
-
True
Edit Project
b52b2c1a73054040ac9866b7740f526f
will
-
True
Edit Project
a8570e2757c544ddbd7e0f6fcdaaf3c0
chmouel
-
True
Edit Project
a54911ab9c1a46c6b9bec01196820345
william
-
True
Edit Project
a4b6fb25502d48429061de88adc4a443
mark
-
True
Edit Project

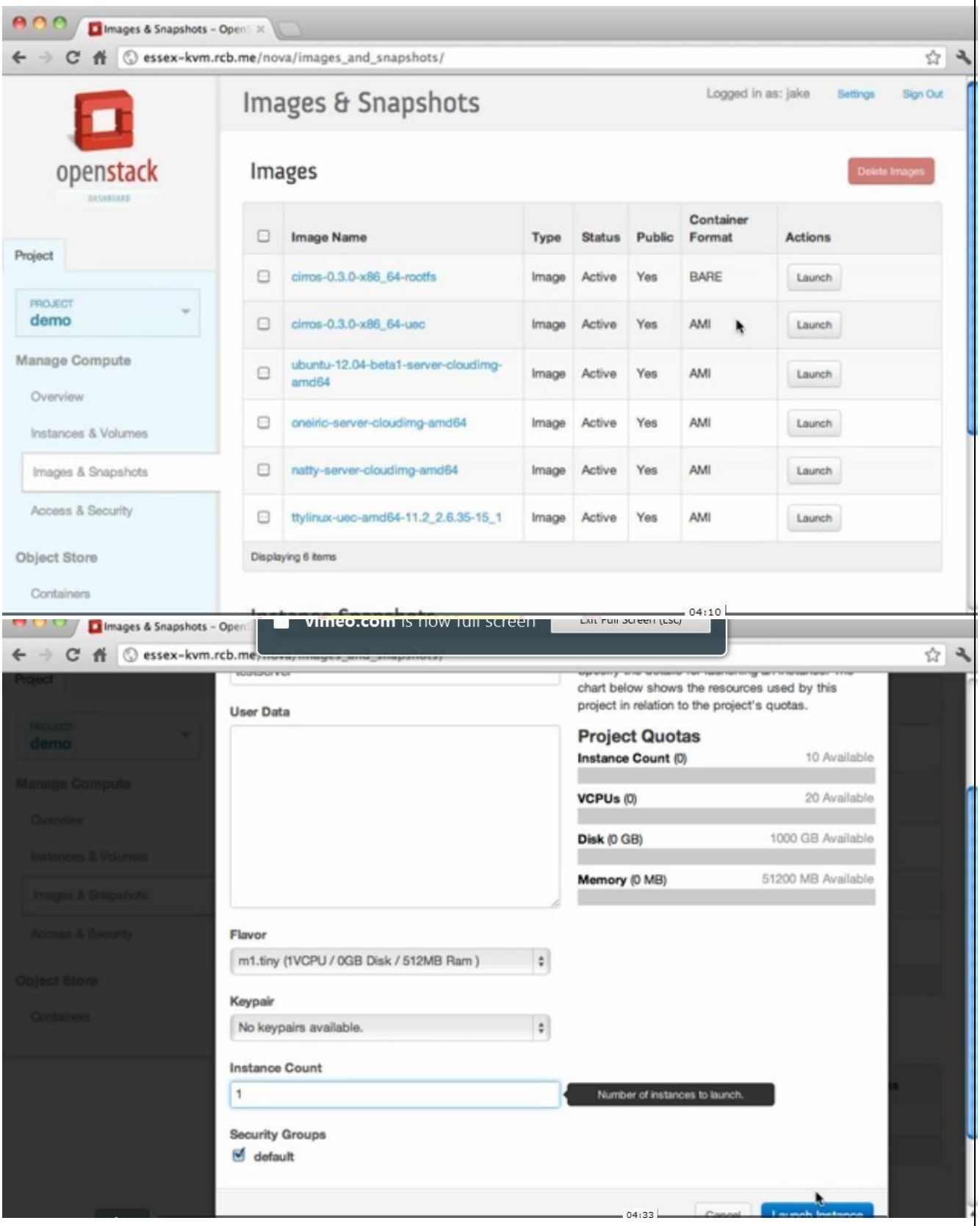

Users - OpenStack Dashboa
essex-kvm.rcb.me/syspanel/users/
openstack
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Quotas
Users
Logged in as: admin
Settings
Sign Out
Users
Filter
Filter
Create User
Delete Users
ID
User Name
Email
Enabled
Actions
bb49c6200f0c48f2bcd001bcd78547ba
admin
admin@example.com
True
Edit
0efd430809e24c52a091f78874855f27
demo
demo@example.com
True
Edit
f6984477078f4c3390a44dddec8f7cc9
nova
nova@example.com
True
Edit
ee38ff74f4484771a0aaccb5c424b5cd
glance
glance@example.com
True
Edit
a1c548a7312c461bb1fa93c9285b4872
swift
swift@example.com
True
Edit
a850922117ef48a0835046e4101bfc2b
scott
-
True
Edit
8161af9b4ad44f04b1359938b540e6d8
jesse
-
True
Edit
20eeff24fa664d5da4db072c
dolph
-
True
Edit

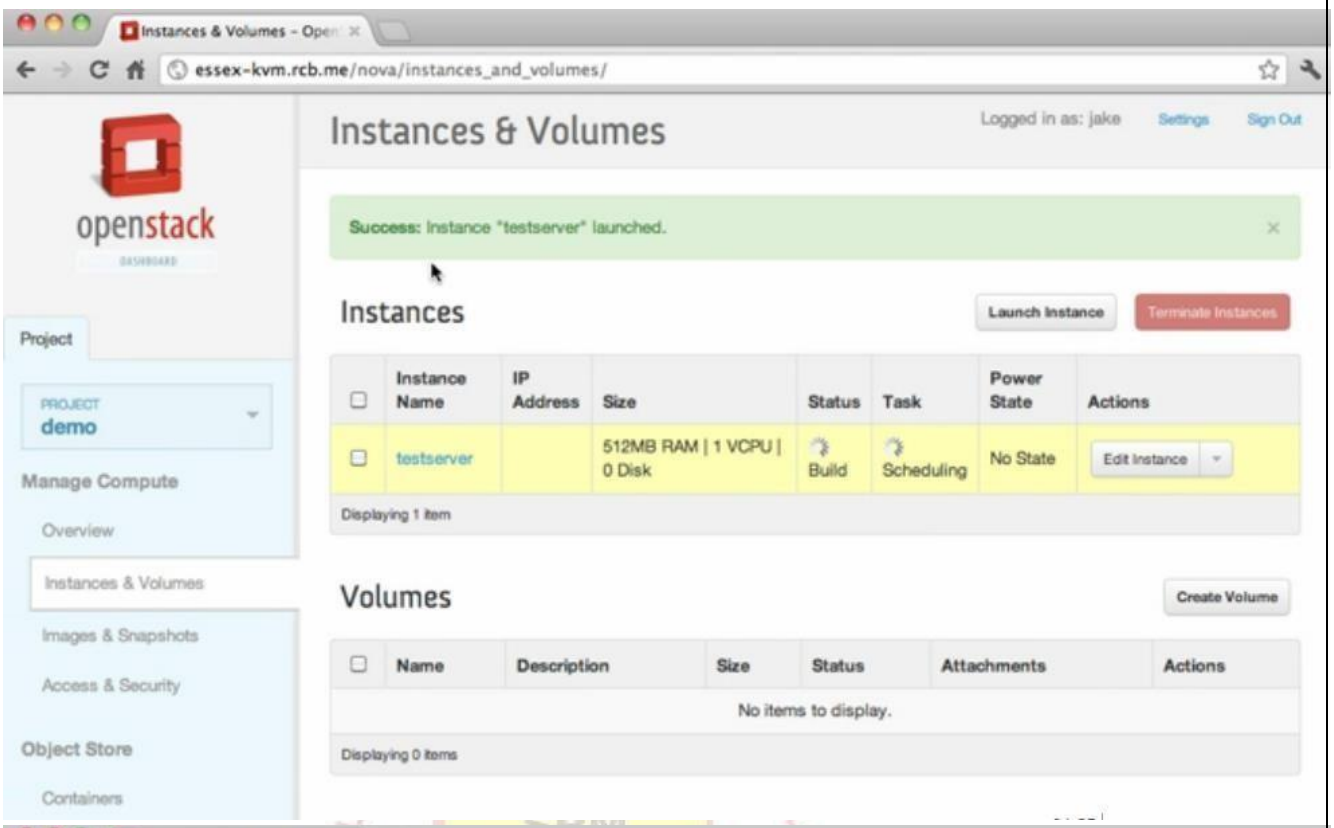

Quotas - OpenStack Dashb
essex-kvm.rcb.me/syspanel/quotas/
openstack
Project
Admin
System Panel
Overview
Instances
Services
Flavors
Images
Projects
Users
Default Quotas
Logged in as: admin
Settings
Sign Out
Quotas
Filter
Filter
Quota Name
Limit
metadata_items
128
injected_file_content_bytes
10240
volumes
10
gigabytes
1000
ram
51200
floating_ips
10
instances
10
injected_files
5
cores
20
Displaying 9 items

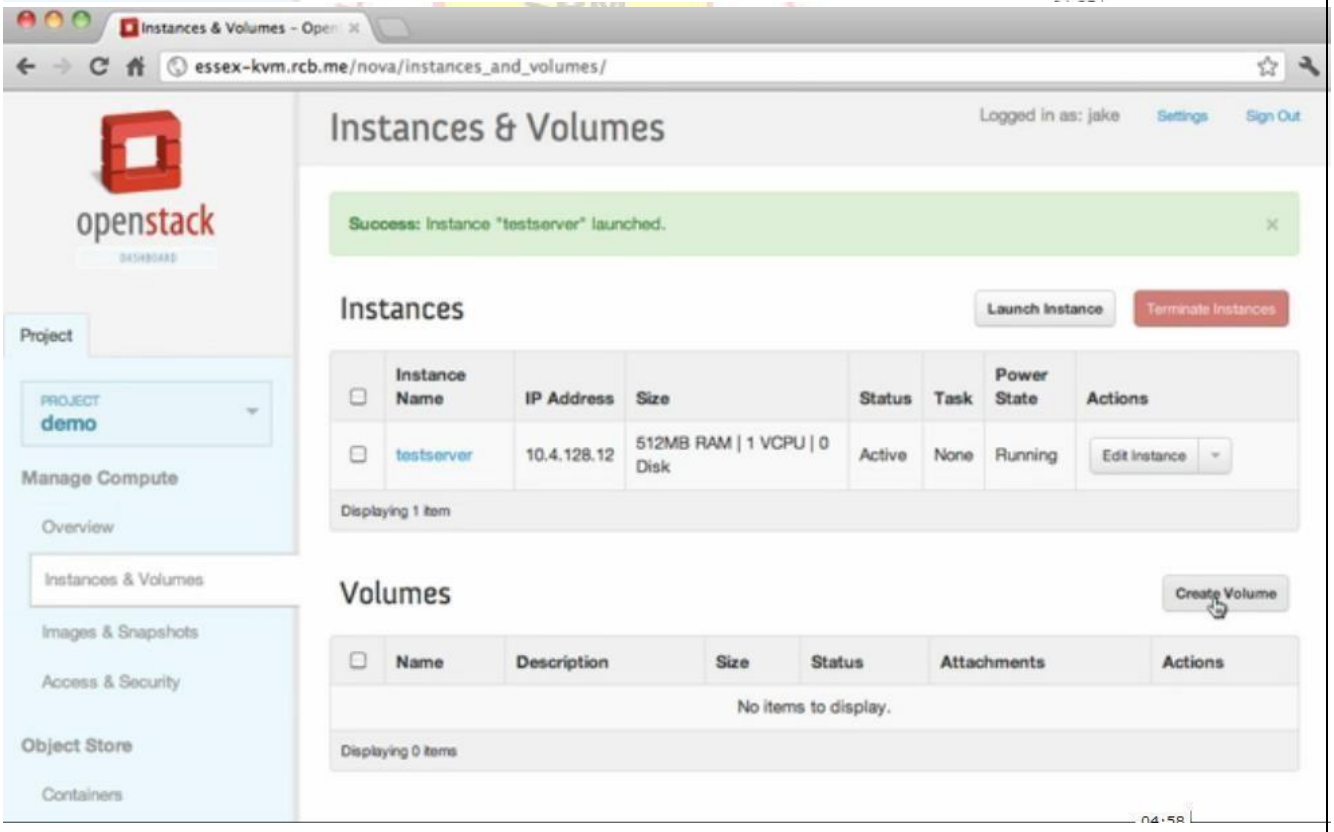

Users - OpenStack Dashboard
essex-kvm.rcb.me/syspanel/users/
Users
openstack
Create User
User Name
jake
Email
jake@rackspace.com
Password
Confirm Password
Primary Project
demo
Description:
From here you can create a new user and assign them to a project.
Cancel
Create User
03:26
Login - OpenStack Dashboard
essex-kvm.rcb.me
openstack
Log In
User Name
jake
Password
Sign In

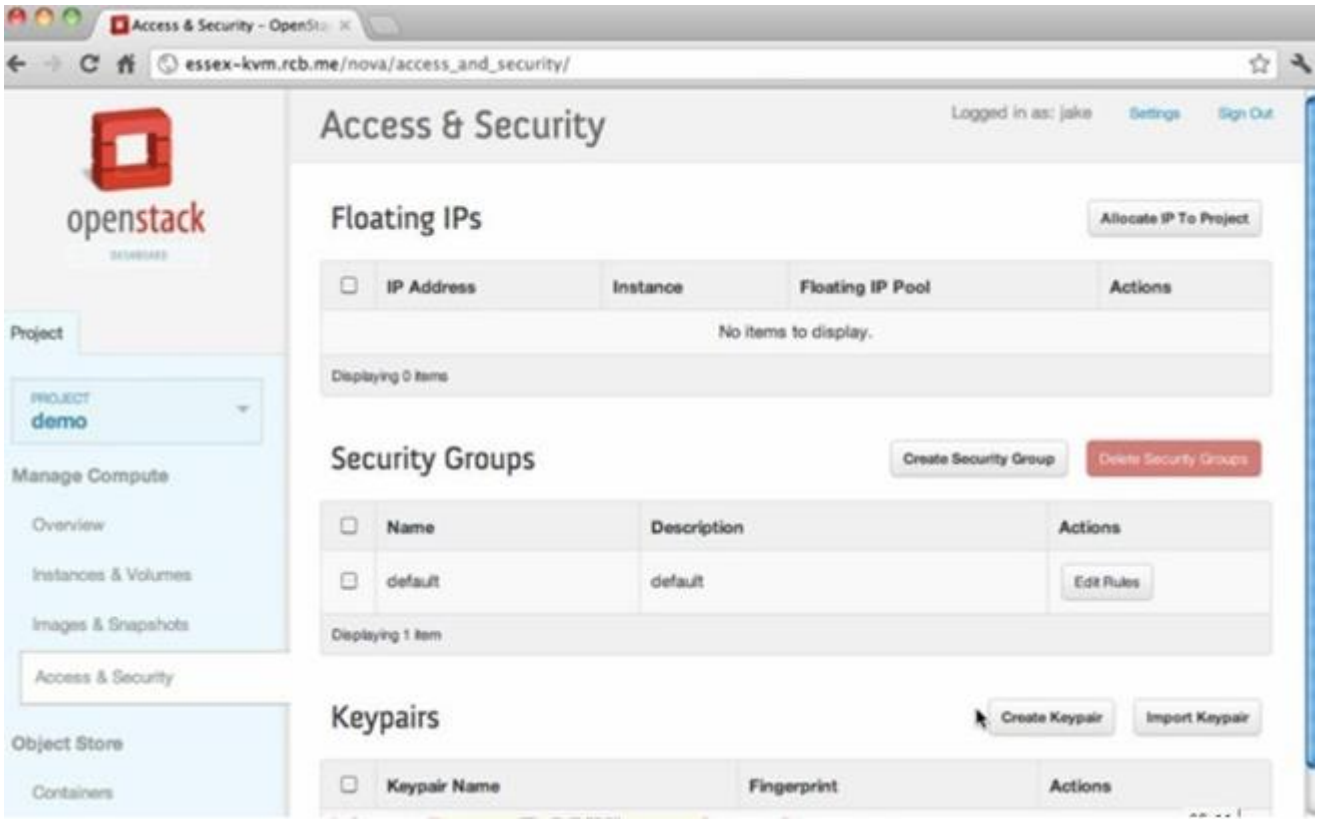

Instance Overview - OpenSt
essex-kvm.rcb.me/nova/
Do you want Google Chrome to save your password?
Never for this site
Save password
openstack
Overview
Logged in as: jake
Settings
Sign Out
Select a month to query its usage:
April
2012
Submit
Active Instances: - Active Memory: - This Month's VCPU-Hours: 0.00 This Month's GB-Hours: 0.00
Usage Summary
Download CSV Summary
Instance Name
VCPUs
Disk
RAM
Uptime
No items to display.
Displaying 0 items
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store

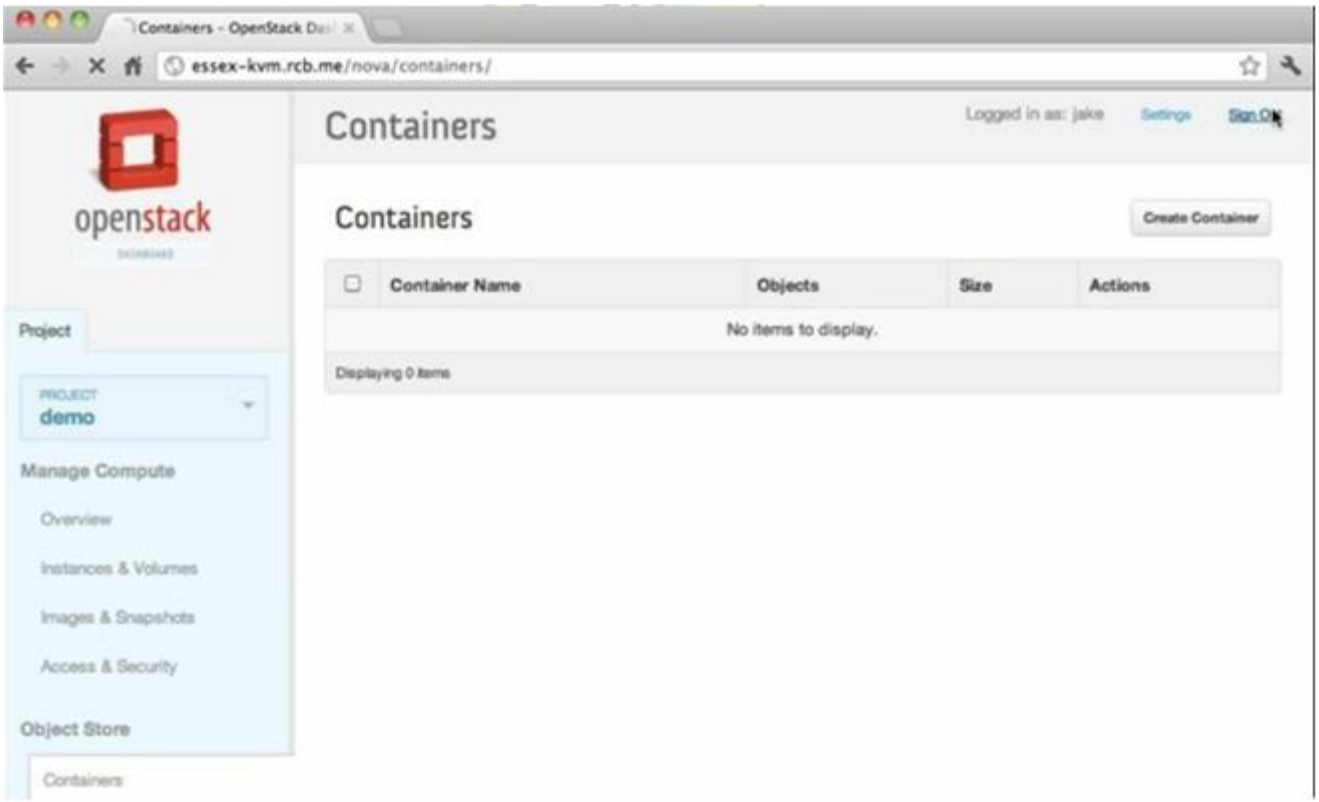

Instances & Volumes - Open
essex-kvm.rcb.me/nova/instances_and_volumes/
openstack
Instances & Volumes
Logged in as: jake
Settings
Sign Out
Instances
Launch Instance
Instance Name
IP Address
Size
Status
Task
Power State
Actions
No items to display.
Displaying 0 items
Volumes
Create Volume
Name
Description
Size
Status
Attachments
Actions
No items to display.
Displaying 0 items
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store
Containers


Images & Snapshots - Open
essex-kvm.rcb.me/nova/images_and_snapshots/
openstack
Images & Snapshots
Logged in as: jake
Settings
Sign Out
Images
Delete Images
Image Name
Type
Status
Public
Container Format
Actions
cirros-0.3.0-x86_64-rootfs
cirros-0.3.0-x86_64-uec
ubuntu-12.04-beta1-server-cloudimg-amd64
oneiric-server-cloudimg-amd64
natty-server-cloudimg-amd64
ttylinux-uec-amd64-11.2_2.6.35-15_1
Launch
Displaying 6 items
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store
Containers
04:10
Vimeo.com is now full screen
Exit Full Screen (Esc)
User Data
chart below shows the resources used by this project in relation to the project's quotas.
Project Quotas
Instance Count (0)
10 Available
VCPUs (0)
20 Available
Disk (0 GB)
1000 GB Available
Memory (0 MB)
51200 MB Available
Flavor
m1.tiny (1VCPU / 0GB Disk / 512MB Ram)
Keypair
No keypairs available.
Instance Count
1
Number of instances to launch.
Security Groups
default
04:33
Cancel
Launch Instance


Instances & Volumes - Open
essex-kvm.rcb.me/nova/instances_and_volumes/
openstack
Instances & Volumes
Logged in as: jake
Settings
Sign Out
Success: Instance "testserver" launched.
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store
Containers
Instances
Launch Instance
Terminate Instances
Instance Name
IP Address
Size
Status
Task
Power State
Actions
testserver
512MB RAM | 1 VCPU | 0 Disk
Build
Scheduling
No State
Edit Instance
Displaying 1 item
Volumes
Create Volume
Name
Description
Size
Status
Attachments
Actions
No items to display.
Displaying 0 items


Instances & Volumes - Open
essex-kvm.rcb.me/nova/instances_and_volumes/
openstack
Instances & Volumes
Logged in as: jake
Settings
Sign Out
Success: Instance "testserver" launched.
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store
Containers
Instances
Launch Instance
Terminate Instances
Instance Name
IP Address
Size
Status
Task
Power State
Actions
testserver
10.4.128.12
512MB RAM | 1 VCPU | 0 Disk
Active
None
Running
Edit Instance
Displaying 1 item
Volumes
Create Volume
Name
Description
Size
Status
Attachments
Actions
No items to display.
Displaying 0 items


Access & Security - OpenSt
essex-kvm.rcb.me/nova/access_and_security/
openstack
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store
Containers
Access & Security
Logged in as: jake
Settings
Sign Out
Floating IPs
Allocate IP To Project
IP Address
Instance
Floating IP Pool
Actions
No items to display.
Displaying 0 items
Security Groups
Create Security Group
Delete Security Groups
Name
Description
Actions
default
default
Edit Rules
Displaying 1 item
Keypairs
Create Keypair
Import Keypair
Keypair Name
Fingerprint
Actions


Containers - OpenStack Das
essex-kvm.rcb.me/nova/containers/
openstack
Project
PROJECT
demo
Manage Compute
Overview
Instances & Volumes
Images & Snapshots
Access & Security
Object Store
Containers
Containers
Logged in as: jake
Settings
Sign Out
Containers
Create Container
Container Name
Objects
Size
Actions
No items to display.
Displaying 0 items

Result:

Thus the virtual machine is launched in Ubuntu 14.04.

Ex No: 11
Date :

BUILDING MODULAR PROGRAMS USING MAKE

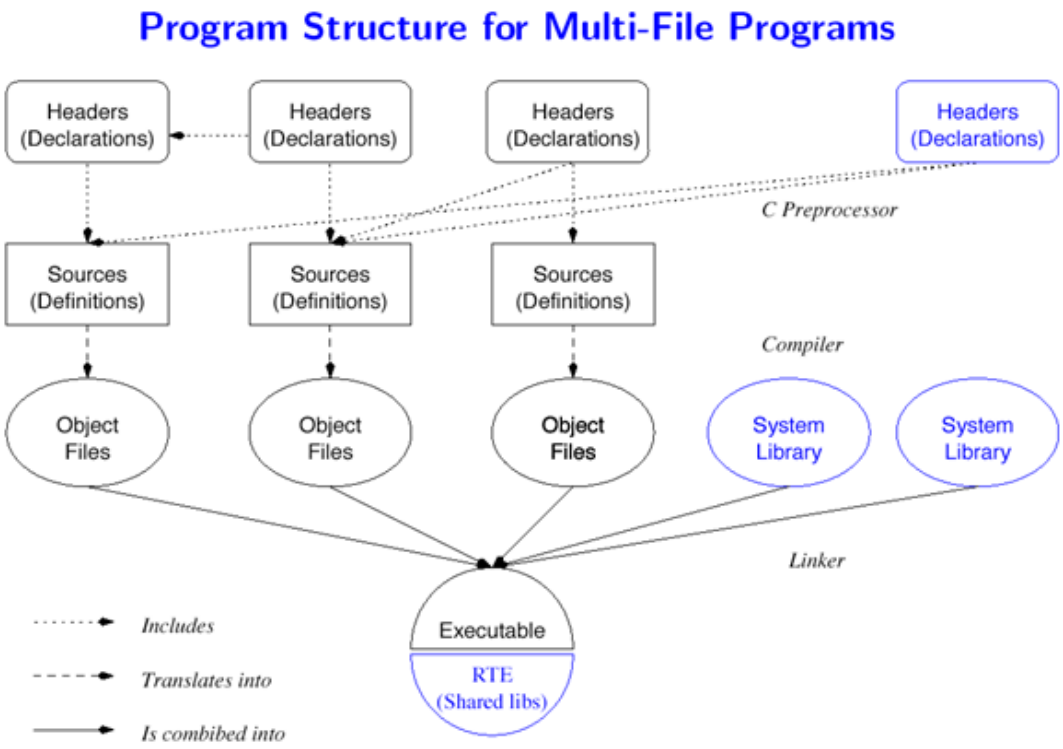
Aim:

To use gcc to compile c-programs. Split the programs to different modules and create an application using make command.

Procedure:

- A Monolithic Program
  - Consider this monolithic program Monolithic.c
  - Placing the code into one source file has a number of shortcomings:
    - It doesn't reflect the modularity of the design.
      There is no clear separation of the ADTs from each other and the main program.
    - There is no information hiding.
      The main program contains details of the PersonType representation which it doesn't need to know.
    - It doesn't scale. Storing a large program as a single file produces large files that are hard to navigate. Every time we make a change to one part, we are forced to recompile the whole program.
  - The mechanisms that enable us to reflect modularity in our code are *interface and implementation* and *separate compilation*.
- Interface and Implementation
  - A module's *interface* is a description of the data and functions that are visible to code external to the module. These descriptions are usually just the type signatures of the externally visible data and functions.
  - A module's *implementation* comprises the data structures and code that make the module work.
  - Separating the *interface* from the *implementation* is the way we reflect *abstraction*.
  - When we use a module, we need to know only what effects accessing the data and routines in its *interface* will have and we can ignore the details of how the module is *implemented*.
- A Modular Program
  - We can split the code above into 4 files as follows:
    - CString.h - defines the CString type
    - Person.h - defines the PersonType type
    - PrettyPrint.h - defines the PRETTY_PRINT macro
    - Person.c - implements the person manipulation
    - Modular.c - main program
  - This separates the interface of the ADT from the implementation.
  - The files can all be compiled together
    - gcc -o Modular Modular.c Person.c
  - The "declare before main, define afterwards" style of programming makes the split easy to do later. Compare these two ...

        - ProgramStructure.c
        - ProgramStructure2.c
- Separate Compilation



    - Separating the interface from the implementation means compiling multiple small files, and some code is compiled multiple times.
    - Some files need not be recompiled when changes are made.
    - The UNIX commands to compile and link separately are
        - gcc -c Source.c to compile
        - gcc -o Executable Object1.o Object2.o ... to link
    - For the example above ... these are the file dependencies
    - The compiler has some options that support multi-file program development.
- make
    - Large programs broken up may use many object and library files
    - Changing a source file means updating the objects, libraries and executables that depend on the source file.
    - The UNIX make utility keeps track of such dependencies, and recreates only the dependant files.
    - To determine which source files is part of the *target* to be built, and which tools to use to build the target, make uses a file called Makefile in the current directory.
        - A simple Makefile for the modular program
        - Lines starting with a # are comments

- A Makefile is a sequence of rules in the form
- target: dependencies
  action

  Note that action must be preceded by a tab character; this is a historical artifact that no one has bothered to fix.

- Actions are just standard UNIX commands that are fed to a shell
- If a dependency does not have a target entry, it is expected to exist as a result of external activity, e.g., editing

- Makefiles are processed from the top down, trying to create a target. By default the first target is used, but any target can be specified on the command line.

```
make(Target) {
   foreach Dependency do {
      if (IsATarget(Dependency)) {
         make(Dependency)
      }
   }
   if (!FileExists(Target) || AnyNewer(Dependencies,Target)) {
      DoActions(Target)
   }
         }
```

- Make variables can hold values like environment variables. make automatically sets the value of certain variables like $@. Variables of this ilk include:
  - $@ The file name of the target of the rule.
  - $< The name of the first dependency.
  - $^ The names of all the dependencies with spaces between them.
  - $? The list of dependencies that are out of date.
- make has rules that it applies by default, specified in /usr/share/lib/make/make.rules. For example, make knows that to get from a .c file to a .o file, it must run the C compiler with the -c flag.
  - Makefile using defaults
- In addition to targets that build programs, make can be made to perform other housekeeping by specifying targets that are not necessarily programs. For example, a common target is
- clean:
- rm -f *.o *.bak *~
  which causes make to delete object files and any backup files (*.bak and *~) created by other programs such as editors.

- makedepend or mkdep
  - Examines each of the source files in its command line and looks for #include directives.

        - Generates a list of source files that are needed by the compiler to produce the resulting target.
        - Result is placed in .depend (FreeBSD) or appended to makefile (Linux)
    - A more realistic Makefile for the example
- Creating a library
    - .o files can be combined into libraries using the ar utility.
    - Library files are given a .a extension.
    - The object files can be used by the compiler by specifying the location of the library file with the -L flag, and giving the library file name.
    - Example
    - prompt> gcc -c *.c
    - prompt> ar -rv libmystuff.a *.o
    - prompt> pwd
    - /home/geoff/c
    - prompt> gcc MyMain.c -L/home/geoff/c -lmystuff -o MyMain

Exercises

- Below is the content of a makefile. Assume that the files part1.c and part3.c have just been modified. Write down the sequence of command activations in the correct order when make is run.

```
all: part1.o part2.o
        gcc part1.o part2.o -o whole

part1.o: part1.c part3.o
        gcc -c part1.c
        gcc part1.o part3.o -o part3.out

part2.o: part2.c
        gcc -c part2.c

part3.o: part3.c
        gcc -c part3.c
```

Program

Monolithic.c

```
//.....
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_CSTRING 128
```

```
typedef char CString[MAX_CSTRING];

typedef struct {
   CString FamilyName;
   CString GivenName;
} PersonType;

#define PRETTY_PRINT(This) printf("\n----------\n%s\n---------- \n",(This))

//-------------------------------------------------------------------------
void SetNames(CString NewFamilyName,CString NewGivenName,PersonType *APerson) {

   strcpy(APerson->FamilyName,NewFamilyName);
   strcpy(APerson->GivenName,NewGivenName);
}
//-------------------------------------------------------------------------
void GetFullName(PersonType APerson,CString FullName) {

   strcpy(FullName,APerson.GivenName);
   strcat(FullName," ");
   strcat(FullName,APerson.FamilyName);
}
//-------------------------------------------------------------------------
int main(void) {

   PersonType MyPerson;
   CString InputFamilyName;
   CString InputGivenName;
   CString  FullName;
   CString OutputLine;

   printf("Please enter the given name and family name : ");
   scanf(" %s %s",InputGivenName,InputFamilyName);
   SetNames(InputFamilyName,InputGivenName,&MyPerson);
   GetFullName(MyPerson,FullName);
   sprintf(OutputLine,"The full name is %s",FullName);
   PRETTY_PRINT(OutputLine);
   return(EXIT_SUCCESS);
}
//-------------------------------------------------------------------------
CString.h

#ifndef CSTRING_H
#define CSTRING_H
//-------------------------------------------------------------------------
#define MAX_CSTRING 128
```

```
typedef char CString[MAX_CSTRING];
//---------------------------------------------------------------------------
#endif
```

Person.h

```
#ifndef PERSON_H
#define PERSON_H
//---------------------------------------------------------------------------
#include "CString.h"

typedef struct {
   CString FamilyName;
   CString GivenName;
} PersonType;
//---------------------------------------------------------------------------
void SetNames(CString NewFamilyName,CString NewGivenName,PersonType *APerson);
void GetFullName(PersonType APerson,CString FullName);
//---------------------------------------------------------------------------
#endif
\
```

PrettyPrint.h

```
#ifndef PRETTYPRINT_H
#define PRETTYPRINT_H
//---------------------------------------------------------------------------
#define PRETTY_PRINT(This) printf("\n----------------\n%s\n---------------- \n",(This))
//---------------------------------------------------------------------------
#endif
```

Person.c

```
//---------------------------------------------------------------------------
#include <string.h>
#include "CString.h"
#include "Person.h"
//---------------------------------------------------------------------------
void SetNames(CString NewFamilyName,CString NewGivenName,
PersonType *APerson) {

// ---Comment
   strcpy(APerson->FamilyName,NewFamilyName);
   strcpy(APerson->GivenName,NewGivenName);
}
```

```
//-------------------------------------------------------------------------
void GetFullName(PersonType APerson,CString FullName) {

   strcpy(FullName,APerson.GivenName);
   strcat(FullName," ");
   strcat(FullName,APerson.FamilyName);
}
//-------------------------------------------------------------------------
```

Modular.c

```
//-------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include "CString.h"
#include "Person.h"
#include "PrettyPrint.h"
//-------------------------------------------------------------------------
int main(void) {
   PersonType MyPerson;
   CString InputFamilyName;
   CString InputGivenName;
   CString  FullName;
   CString OutputLine;
   printf("Please enter the given name and family name : ");
   scanf(" %s %s",InputGivenName,InputFamilyName);
   SetNames(InputFamilyName,InputGivenName,&MyPerson);
   GetFullName(MyPerson,FullName);
   sprintf(OutputLine,"The full name is %s",FullName);
   PRETTY_PRINT(OutputLine);

   return(EXIT_SUCCESS);
}
//-------------------------------------------------------------------------
```

Result

Thus the program use make command executed and answer id verified

Ex.No:12

Date:

CONTROL SYSTEMS COMMAND TO CLONE, COMMIT, PUSH, FETCH, PULL, CHECKOUT, RESET, AND DELETE

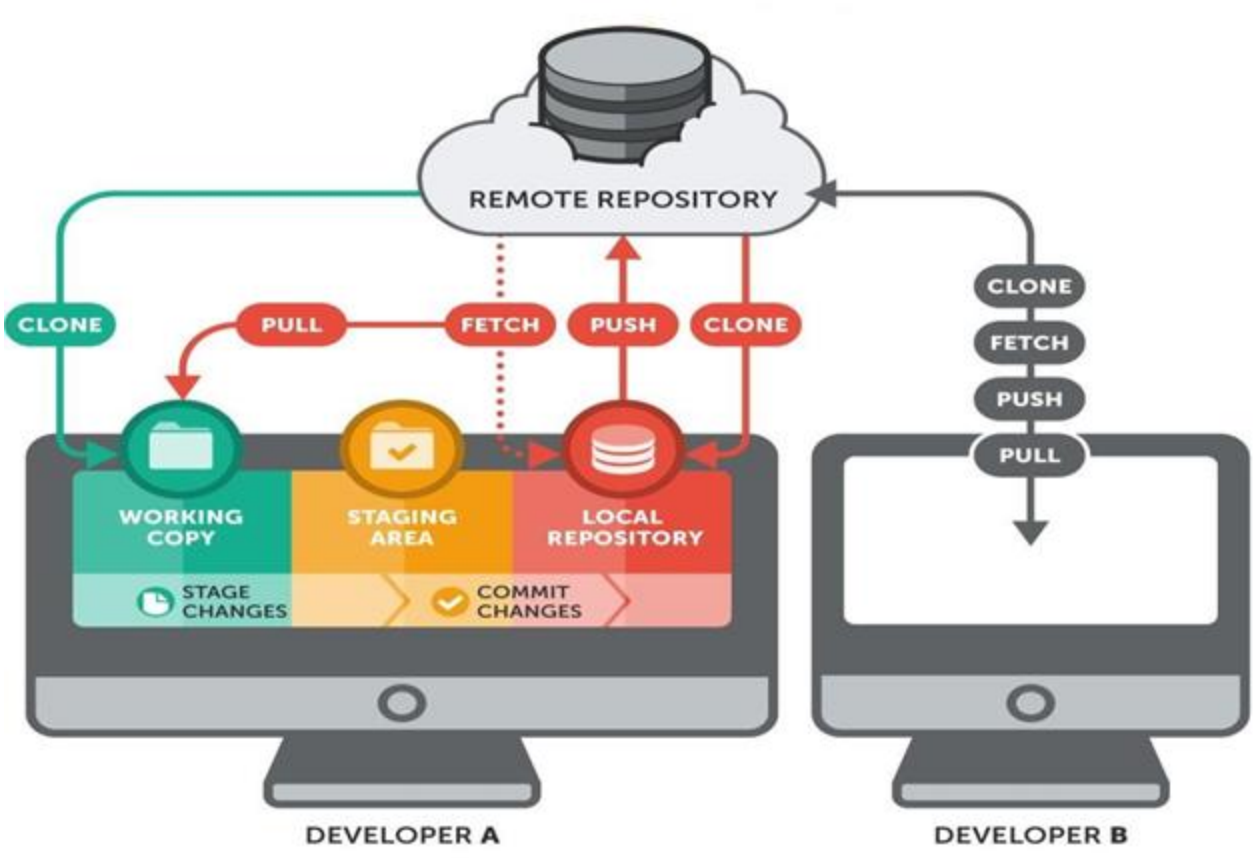
Aim:

Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories.

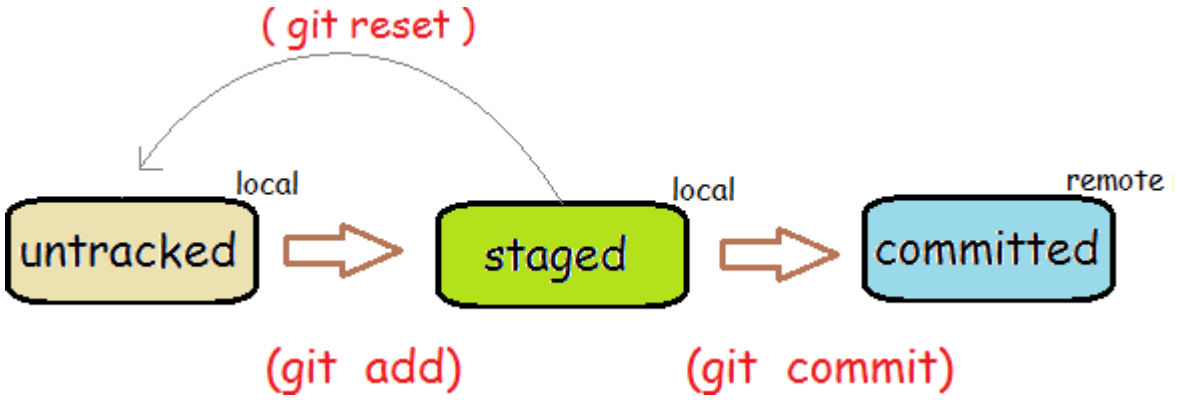
Description:

Git is a *version control system (software)* and **GitHub** is a *source code hosting service*. Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.

Lets Now, Understand the working of **Git**



A file in git goes through the following stages:



Procedure

Setting Up Git

You need to setup Git on your local machine, as follows:

1. Download & Install:
   - For Windows and Mac, download the installer from http://git-scm.com/downloads and run the downloaded installer.
   - For Ubuntu, issue command "sudo apt-get install git".

   For Windows, use the "Git Bash" command shell bundled with Git Installer to issue commands. For Mac/Ubuntu, use the "Terminal".
2. Customize Git:
   Issue "git config" command (for Windows, run "Git Bash" from the Git installed directory. For Ubuntu/Mac, launch a "Terminal"):
3. // Set up your username and email (to be used in labeling your commits)
4. $ git config --global user.name "your-name"
   $ **git config --global user.email "*your-email@youremail.com*"**
   The settings are kept in "<GIT_HOME>/etc/gitconfig" (of the GIT installed directory) and "<USER_HOME>/.gitconfig" (of the user's home directory. You can issue "git config --list" to list the settings:
   $ **git config --list**
   user.email=xxxxxx@xxxxxx.com
   user.name=xxxxxx

Git Commands

Git provides a set of simple, distinct, standalone commands developed according to the "Unix toolkit" philosophy - build small, interoperable tools.

To issue a command, start a "Terminal" (for Ubuntu/Mac) or "Git Bash" (for Windows):

$ **git <command> <arguments>**

The commonly-used commands are:

1. **init**, **clone**, **config**: for starting a Git-managedproject.

2. **add**, **mv**, **rm**: for staging file changes.
3. **commit**, **rebase**, **reset**, **tag**:
4. **status**, **log**, **diff**, **grep**, **show**: show status
5. **checkout**, **branch**, **merge**, **push**, **fetch**, **pull**

Help and Manual

The best way to get help these days is certainly *googling*.

To get help on Git commands:

```
$ git help <command>
// or
$ git <command> --help
```

The GIT manual is bundled with the software (under the "doc" directory), and also available online @ http://git-scm.com/docs.

Initialize a new Git Repo (git init)

To manage a project under Git, run "git init" at the project *root* directory (i.e., "hello-git") (via "Git Bash" for Windows, or "Terminal" for Ubuntu/Mac):

```
// Change directory to the project directory
$ cd /path-to/hello-git

// Initialize Git repo for this project
$ git init
Initialized empty Git repository in /path-to/hello-git/.git/

$ ls -al
drwxr-xr-x   1 xxxxx   xxxxx    4096 Sep 14 14:58 .git
-rw-r--r--   1 xxxxx   xxxxx     426 Sep 14 14:40 Hello.class
-rw-r--r--   1 xxxxx   xxxxx     142 Sep 14 14:32 Hello.java
-rw-r--r--   1 xxxxx   xxxxx      66 Sep 14 14:33 README.md
```

A hidden sub-directory called ".git" will be created under your project *root* directory (as shown in the above "ls -a" listing), which contains ALL Git related data.

Git Storage Model

The local repo after "git init" is empty. You need to explicitly deposit files into the repo.
Before we proceed, it is important to stress that Git manages changes to files between so-called commits. In other words, it is a version control system that allows you to keep track of the file changes at the commits.

Staging File Changes for Tracking (git add <file>...)

Issue a "git status" command to show the status of the files:

```
$ git status
On branch master
```

```
Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      Hello.class
      Hello.java
      README.md
nothing added to commit but untracked files present (use "git add" to track)
```

By default, we start on a *branch* called "master". We will discuss "branch" later.
In Git, the files in the working tree are either *untracked* or *tracked*. Currently, all 3 files are *untracked*. To stage a new file for tracking, use "git add <file>..." command.

```
// Add README.md file
$ git add README.md

$ git status
On branch masterInitial commit

 Changes to be committed:
(use "git rm --cached <file>..." to unstage)new file:
README.md

 Untracked files:
(use "git add <file>..." to include in what will be committed)Hello.class
Hello.java

 // You can use wildcard * in the filename
 // Add all Java source files into Git repo
 $ git add *.java

 // You can also include multiple files in the "git add"
 // E.g.,
 // git add Hello.java README.md

 $ git status
On branch masterInitial commit

 Changes to be committed:
(use "git rm --cached <file>..." to unstage)new file:
Hello.java
new file:  README.md

 Untracked files:
```

```
  (use "git add <file>..." to include in what will be committed)
        Hello.class
```

The command "git add <file>..." takes one or more filenames or pathnames with possibly wildcards pattern. You can also use "git add ." to add all the files in the current directory (and all sub-directories). But this will include "Hello.class", which we do not wish to be tracked.
When a new file is added, it is *staged* (or *index*ed, or *cached*) in the *staging area* (as shown in the GIT storage model), but NOT yet *committed*.

Git uses two stages to commit file changes:

1. "git add <file>" to stage file changes into the *staging area*, and
2. "git commit" to commit ALL the file changes in the *staging area* to the *local repo*.

The staging area allows you to group related file changes and commit them together.

Committing File Changes (git commit)

The "git commit" command commits ALL the file changes in the *staging area*. Use a -m option to provide a *message* for the commit.

```
$ git commit -m "First commit" // -m to specify the commit message
[master (root-commit) 858f3e7] first commit
 2 files changed, 8 insertions(+)
 create mode 100644 Hello.java
 create mode 100644 README.md

// Check the status
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      Hello.class
nothing added to commit but untracked files present (use "git add" to track)
```

Commit ALL staged file changes via "git commit":

```
$ git commit -m "Second commit"
[master 96efc96] Second commit
 1 file changed, 1 insertion(+)

$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      Hello.class
nothing added to commit but untracked files present (use "git add" to track)
```

Once the file changes are committed, it is marked as *unmodified* in the staging area (not shown in "Changes to be committed").
Both "git diff" and "git diff --staged" return empty output, signalling there is no "unstaged" and "staged" changes.

More on "git checkout" and Detached HEAD

"git checkout" can be used to checkout a branch, a commit, or files. The syntaxes are:

```
$ git checkout <branch-name>
$ git checkout <commit-name>
$ git checkout <commit-name> <filename>
```

When you checkout a commit, Git switches into so-called "Detached HEAD" state, i.e., the HEAD detached from the tip of a branch. Suppose that you continue to work on the detached HEAD on commit-5, and wish to merge the commit-5 back to master. You checkout the master branch, but there is no branch name for your to reference the commit-5!!!

In Summary, you can use "git checkout <commit-name>" to inspect a commit. BUT you should always work on a branch, NOT on a detached HEAD.

More on "git reset" and "git reset --hard"

```
$ git reset <file>
  // Unstage the changes of <file> from staging area,
  // not affecting the working tree.

$ git reset
  // Reset the staging area
  // Remove all changes (of all files) from staging area,
  // not affecting the working tree.

$ git reset --hard
  // Reset the staging area and working tree to match the
  // recent commit (i.e., discard all changes since the
  // last commit).

$ git reset <commit-name>
  // Move the HEAD of current branch to the given commit,
  // not affecting the working tree.

$ git reset --hard <commit-name>
  // Reset both staging area and working tree to the given
  // commit, i.e., discard all changes after that commit.
```

[TODO] Diagram

[TODO] --soft option

Summary of Work Flows

Setting up GIT and "Edit/Stage/Commit/Push" Cycle

Step 1: Install GIT.

- For Windows and Mac, download the installer from http://git-scm.com/downloads and run the downloaded installer.

- For Ubuntu, issue command "sudo apt-get install git".

For Windows, use "git-bash" command shell provided by Windows installer to issue command.
For Mac/Ubuntu, use "Terminal".

Step 2: Configuring GIT:

```
// Setup your username and email to be used in labeling commits
$ git config --global user.email "your-email@yourmail.com"
$ git config --global user.name "your-name"
```

Step 3: Set up GIT repo for a project. For example, we have a project called "olas1.1" located at "/usr/local/olas/olas1.1".

```
$ cd /usr/local/olas/olas1.1

// Initialize the GIT repo
$ git init

$ ls -al
  // Check for ".git" directory
```

Create a "README.md" (or "README.textile" if you are using Eclipse's WikiText in "textile" markup) under your project directory to describe the project.

Step 4: Start "Edit/Stage/Commit/Push" cycles.

Create/Modify files. Stage files into the staging area via "git add <file>".

```
// Check the status
$ git status
......

// Add files into repo
$ git add README.md
$ git add www
......

// Check the status
$ git status
......
```

Step 5: Create a ".gitignore" (in the project base directory) to exclude folders/files from being tracked by GIT. Check your "git status" output to decide which folders/files to be ignored.

For example,

```
# ignore files and directories beginning with dot
.*

# ignore directories beginning with dot (a directory ends with a slash)
.*/

# ignore these files and directories
www/test/
```

```
www/.*
www/.*/
```

The trailing slash indicate directory (and its sub-directories and files).

If you want the ".gitignore" to be tracked (which is in the ignore list):

```
$ git add -f .gitignore
    // -f to override the .gitignore
```

Step 6: Commit.

```
$ git status
......

// Commit with a message
$ git commit -m "Initial Commit"
......

$ git status
......
```

Step 7: Push to the Remote Repo (for backup, version control, and collaboration).
You need to first create a repo (says olas) in a remote GIT host, such as GitHub or BitBucket.
Take note of the remote repo URL, e.g., https://*username@hostname*.org/*username*/olas.git.

```
$ cd /path-to/local-repo

// Add a remote repo name called "origin" mapped to the remote URL
$ git remote add origin https://hostname/username/olas.git

// Push the "master" branch to the remote "origin"
// "master" is the default branch name of your local repo after init.
$ git push origin master
```

Check the remote repo for the files committed.

Step 8: Work on the source files, make changes, commit and push to remote repo.

```
// Check the files modified
$ git status
......

// Stage for commit the modified files
$ git add ....
......

// Commit (with a message)
$ git commit -m "commit-message"

// Push to remote repo
```

```
$ git push origin master
```

Step 9: Create a "tag" (for version number).

```
// Tag a version number to the current commit
$ git tag -a v1.1 -m "Version 1.1"
     // -a to create an annotated tag, -m to provide a message

// Display all tags
$ git tag
......

// Push the tags to remote repo
// ("git push -u origin master" does not push the tags)
$ git push origin --tags
```

.

Fetch/Merge Changes from remote (git fetch/merge)

The "git fetch" command imports commits from a remote repo to your local repo, without updating your local working tree. This gives you a chance to review changes before updating (merging into) your working tree. The fetched objects are stored in remote branches, that are differentiated from the local branches.

```
$ cd /path-to/working-directory

$ git fetch <remote-name>
   // Fetch ALL branches from the remote repo to your local repo

$ git fetch <remote-name> <branch-name>
   // Fetch the specific branch from the remote repo to your local repo

// List the local branches
$ git branch
* master
  devel
   // * indicates current branch

// List the remote branches
$ git branch -r
  origin/master
  origin/devel

// You can checkout a remote branch to inspect the files/commits.
// But this put you into "Detached HEAD" state, which prevent you
// from updating the remote branch.

// You can merge the fetched changes into local repo
$ git checkout master
```

```
  // Switch to "master" branch of local repo
$ git merge origin/master
  // Merge the fetched changes from stored remote branch to local
```

git pull

As a short hand, "git pull" combines "git fetch" and "git merge" into one command, for convenience.

```
$ git pull <remote-name>
  // Fetch the remote's copy of the current branch and merge it
  // into the local repo immediately, i.e., update the working tree

// Same as
$ git fetch <remote-name> <current-branch-name>
$ git merge <remote-name> <current-branch-name>

$ git pull --rebase <remote-name>
  // linearize local changes after the remote branch.
```

The "git pull" is an easy way to *synchronize* your local repo with origin's (or upstream) changes (for a specific branch).

Pushing to Remote Repo (revision)

The "git push <remote-name> <branch-name>" is the counterpart of "git fetch", which exports commits from local repo to remote repo.

```
$ git push <remote-name> <branch-name>
  // Push the specific branch of the local repo

$ git push <remote-name> --all
  // Push all branches of the local repo

$ git push <remote-name> --tag
  // Push all tags
  // "git push" does not push tags

$ git push -u <remote-name> <branch-name>
  // Save the remote-name and branch-name as the
  // reference (or current) remote-name and branch-name.
  // Subsequent "git push" without argument will use these references.
```

REFERENCES & RESOURCES

1. GIT mother site @ http://git-scm.com and GIT Documentation @ http://git-scm.com/doc.

Result:

Thus the control commands executed in GIT

VIVA QUESTIONS

VIVA QUESTIONS AND ANSWERS

1.Define Cloud Computing with example.

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

2.What is the working principle of Cloud Computing?

The cloud is a collection of computers and servers that are publicly accessible via the Internet. This hardware is typically owned and operated by a third party on a consolidated basis in one or more data center locations. The machines can run any combination of operating systems.

3.What are the advantages and disadvantages of Cloud Computing?

Advantages

- Lower-Cost Computers for Users
- Improved Performance
- Lower IT Infrastructure Costs
- Fewer Maintenance Issues
- Lower Software Costs
- Instant Software Updates
- Increased Computing Power
- Unlimited Storage Capacity
- Increased Data Safety
- Improved Compatibility Between Operating Systems
- Improved Document Format Compatibility
- Easier Group Collaboration
- Universal Access to Documents
- Latest Version Availability
- Removes the Tether to Specific Devices

Disadvantages

- Requires a Constant Internet Connection
- Doesn't Work Well with Low-Speed Connections
- Can Be Slow
- Features Might Be Limited
- Stored Data Might Not Be Secure
- If the Cloud Loses Your Data, You're Screwed

4.What is distributed system?

A *distributed system* is a software system in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal.

Three significant characteristics of distributed systems are:

- ✓ Concurrency of components
- ✓ Lack of a global clock
- ✓ Independent failure of components
- ✓ What is cluster?
- ✓ Acomputingclusterconsistsofinterconnectedstand-alonecomputerswhichworkcooperativelyasasingleintegratedcomputingresource.Inthepast,clusteredcomputersystemshavedemonstrated

5.What is grid computing?

Grid Computing enables virtuals organizations to share geographically distributed resources as they pursue common goals, assuming the absence of central location, central control, omniscience, and an existing trust relationship.

(or)

- ✓ Gridtechnologydemandsnewdistributedcomputingmodels,software/middlewaresupport,networkprotocols,andhardwareinfrastructures.
- ✓ Nationalgridprojectsarefollowedbyindustrialgridplatformdevelopmentby IBM,Microsoft,Sun,HP,Dell,Cisco,EMC,PlatformComputing,andothers. Newgridserviceproviders(GSPs)andnewgridapplicationshaveemergedrapidly,similartothegrowthofInternetandwebservicesinthepasttwodecades.
- ✓ gridsystemsareclassifiedinessentiallytwocategories:computationalordatagridsandP2Pgrids.

6.What are the business areas needs in Grid computing?

- ✓ Life Sciences
- ✓ Financial services
- ✓ Higher Education
- ✓ Engineering Services
- ✓ Government
- ✓ Collaborative games

7.List out the Grid Applications:

- ✓ Application partitioning that involves breaking the problem into discrete pieces
- ✓ Discovery and scheduling of tasks and workflow
- ✓ Data communications distributing the problem data where and when it is required
- ✓ Provisioning and distributing application codes to specific system nodes
- ✓ Autonomic features such as self-configuration, self-optimization, self-recovery and self-management

8.List some grid computing toolkits and frameworks?

- ✓ Globus Toolkit Globus Resource Allocation Manager(GRAM)
- ✓ Grid Security Infrastructure(GSI)
- ✓ Information Services
- ✓ Legion, Condor and Condor-G
- ✓ NIMROD, UNICORE, NMI.

9.What are Desktop Grids?

These are grids that leverage the compute resources of desktop computers.
Because of the true (but unfortunate) ubiquity of Microsoft® Windows® operating system in corporations, desktop grids are assumed to apply to the Windows environment.
The Mac OS™ environment is supported by a limited number of vendors.

10.What are Server Grids?

- ✓ Some corporations, while adopting Grid Computing , keep it limited to server resources that are within the purview of the IT department.
- ✓ Special servers, in some cases, are bought solely for the purpose of creating an internal "utility grid" with resources made available to various departments.
- ✓ No desktops are included in server grids. These usually run some flavor of the Unix/Linux operating system.

11.Define Opennebula.

OpenNebula is an open source management tool that helps virtualized data centers oversee private clouds, public clouds and hybrid clouds. ... OpenNebula is vendor neutral, as well as platform- and API-agnostic. It can use KVM, Xen or VMware hypervisors.

12.Define Eclipse.

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment.

13.Define Netbeans.

NetBeans is an open-source integrated development environment (IDE) for developing with Java, PHP, C++, and other programming languages. NetBeans is also referred to as a platform of modular components used for developing Java desktop applications.

14.Define Apache Tomcat.

Apache Tomcat (or Jakarta Tomcat or simply Tomcat) is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run."

15.What is private cloud?

The *private cloud* is built within the domain of an intranet owned by a single organization. Therefore, they are client owned and managed. Their access is limited to the owning clients and their partners. Their deployment was not meant to sell capacity over the Internet through publicly accessible interfaces. Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative domains.

16.What is public cloud?

A *public cloud* is built over the Internet, which can be accessed by any user who has paid for the service. Public clouds are owned by service providers. They are accessed by subscription. Many companies have built public clouds, namely Google App Engine, Amazon AWS, Microsoft Azure, IBM Blue Cloud, and Salesforce Force.com. These are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure.

17. What is hybrid cloud?

A *hybrid cloud* is built with both public and private clouds, Private clouds can also support a *hybrid cloud* model by supplementing local infrastructure with computing capacity from an external public cloud. For example, the *research compute cloud* (RC2) is a private cloud built by IBM.

18.What is a Community Cloud ?

A community cloud in computing is a collaborative effort in which infrastructure is shared between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party and hosted internally or externally. This is controlled and used by a group of organizations that have shared interest. The costs are spread over fewer users than a public cloud (but more than a private cloud

19.Define IaaS?

The IaaS layer offers storage and infrastructure resources that is needed to deliver the Cloud services. It only comprises of the infrastructure or physical resource. Top IaaS Cloud Computing Companies: Amazon (EC2), Rackspace, GoGrid, Microsoft, Terremark and Google.

20.Define PaaS?

PaaS provides the combination of both, infrastructure and application. Hence, organisations using PaaS don't have to worry for infrastructure nor for services. Top PaaS Cloud Computing Companies: Salesforce.com, Google, Concur Technologies, Ariba, Unisys and Cisco..

21.Define SaaS?

In the SaaS layer, the Cloud service provider hosts the software upon their servers. It can be defined as a in model in which applications and softwares are hosted upon the server and made available to customers over a network. Top SaaS Cloud Computing Companies: Amazon Web Services, AppScale, CA Technologies, Engine Yard, Salesforce and Windows Azure.

22.What is meant by virtualization?

Virtualizationisacomputerarchitecturetechnologybywhichmultiplevirtualmachines (VMs)are multipl exedin the same hardwar emachine.Theideaof VMs canbe dated back to the 1960s.The purpose of a VM is to enhance resource sharing by many users and improve computer performance interms of resource utilization and application flexibility.

23.What are the implementation levels of virtualization?

The virtualization types are following

1. OS-level virtualization
2. ISA level virtualization
3. User-ApplicationLevel virtualization
4. hardware level virtualization
5. library level virtualization

24.List the requirements of VMM?

There are three requirements for a VMM.

First, a VMM should provide an environment for programs which is essentially identical to the original machine.

Second, programs run in this environment should show, at worst, only minor decreases in speed.

Third, a VMM should be in complete control of the system resources.

25.Explain Host OS and Guest OS?

A comparison of the differences between a host system, a guest system, and a virtual machine within a virtual infrastructure.

A host system (host operating system) would be the primary & first installed operating system. If you are using a bare metal Virtualization platform like Hyper-V or ESX, there really isn't a host operating system besides the Hypervisor. If you are using a Type-2 Hypervisor like VMware Server or Virtual Server, the host operating system is whatever operating system those applications are installed into.

A guest system (guest operating system) is a virtual guest or virtual machine (VM) that is installed under the host operating system. The guests are the VMs that you run in your virtualization platform.

26.Write the steps for live VM migration?

The five steps for live VM migration is

Stage 0: *Pre-Migration*
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: *Reservation*
Initialize a container on the target host
Stage 2: *Iterative pre-copy*
Enable shadow paging
Copy dirty pages in successive rounds.
Stage 3: *Stop and copy*
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: *Commitment*
VM state on Host A is released
Stage 5: *Activation*
VM starts on Host B
Connects to local devices

Resumes normal operation

27..Define Globus Toolkit: Grid Computing Middleware

- ✓ Globus is open source grid software that addresses the most challenging problmes in distributed resources sharing.
- ✓ The Globus Toolkit includes software services and libraries for distributed security, resource management, monitoring and discovery, and data management.

28.Define Blocks in HDFS

- ✓ A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size. Filesystem blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes. This is generally transparent to the filesystem user who is simply reading or writing a file—of whatever length.

29.Define Namenodes and Datanodes

- ✓ An HDFS cluster has two types of node operating in a master-worker pattern:
  - ➢ a *namenode* (the master) and
  - ➢ a number of *datanodes*(workers).
- ✓ The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log.
- ✓ The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.

30.Define HADOOP.

Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

31.Define HDFS.

Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers. HDFS, MapReduce, and YARN form the core of Apache™ Hadoop®.

32.Write about HADOOP.

Hadoop was created by Doug Cutting and Mike Cafarella in 2005. Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant. It was originally developed to support distribution for the Nutch search engine project.

33.Definition of *Grid Portal*:

A *Grid Portal* provides an efficient infrastructure to put Grid-empowered applications on corporate Intranet/Internet.

34.Define GAE.

Google App Engine (often referred to as GAE or simply App Engine) is a Platform as a Service and cloud computing platform for developing and hosting web applications in Google-managed data centers. Applications are sandboxed and run across multiple servers. App Engine offers automatic scaling for web applications—as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand.

35.What is Cloudsim?

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

36.Core features of CloudSim are:

- 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.

37.Uses of Cloudsim.

- Load Balancing of resources and tasks
- Task scheduling and its migrations
- Optimizing the Virtual machine allocation and placement policies
- Energy-aware Consolidations or Migrations of virtual machines
- Optimizing schemes for Network latencies for various cloud scenarios

38.Define OpenStack.

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.A dashboard is also available, giving administrators control while empowering their users to provision resources through a web interface.

39.Define Trystack.

TryStack is a great way to take OpenStack for a spin without having to commit to a full deployment.

This free service lets you test what the cloud can do for you, offering networking, storage and compute instances, without having to go all in with your own hardware.

It's a labor of love spearheaded by three Red Hat OpenStack experts Will Foster, Kambiz Aghaiepour and Dan Radez.

TryStack's set-up must bear the load of anyone who wants to use it, but instead of an equally boundless budget and paid staff, it was originally powered by donated equipment and volunteers from Cisco, Dell, Equinix, NetApp, Rackspace and Red Hat who pulled together for this OpenStack Foundation project.

40.Define Hadoop.

Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.