

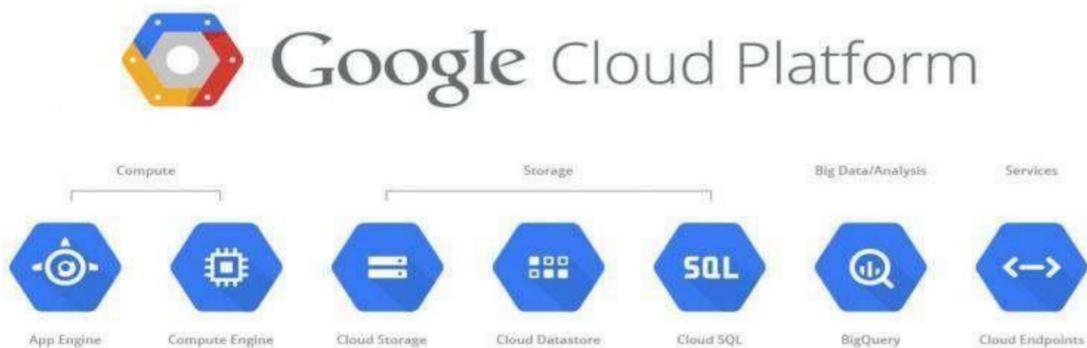
<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>01/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Aim/TITLE:**

Install Google App Engine. Create hello world app and other simple web applications using python/java. Use GAE launcher to launch the web applications

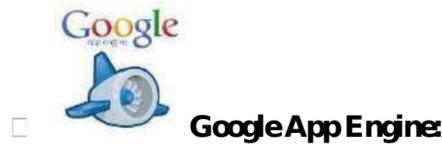
**Theory:****Introduction** **Google Cloud Platform (GCP)**

- **Google Cloud Platform (GCP)**, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube.
- Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning.
- Google Cloud Platform provides infrastructure as a service, platform as a service, and serverless computing environments.

 **Platform as a Service (PaaS)**

- Cloud computing service which provides a computing platform and a solution stack as a service.
- Consumer creates the software using tools and/or libraries from the provider.
- Provider provides the networks, servers, storage, etc.

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>02/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>



### **Google App Engine:**

- Google App Engine was first released as a beta version in April 2008.
- It is a Platform as a Service (PaaS) cloud computing platform for developing and hosting web applications in Google-managed data centers.
- Google's App Engine opens Google's production to any person in the world at no charge.
- Google App Engine is software that facilitates the user to run his web applications on Google infrastructure.
- It is more reliable because failure of any server will not affect either the performance of the end user or the service of the Google.
- It virtualizes applications across multiple servers and data centers.
  - Other cloud-based platforms include offerings such as Amazon Web Services and Microsoft's Azure Services Platform.



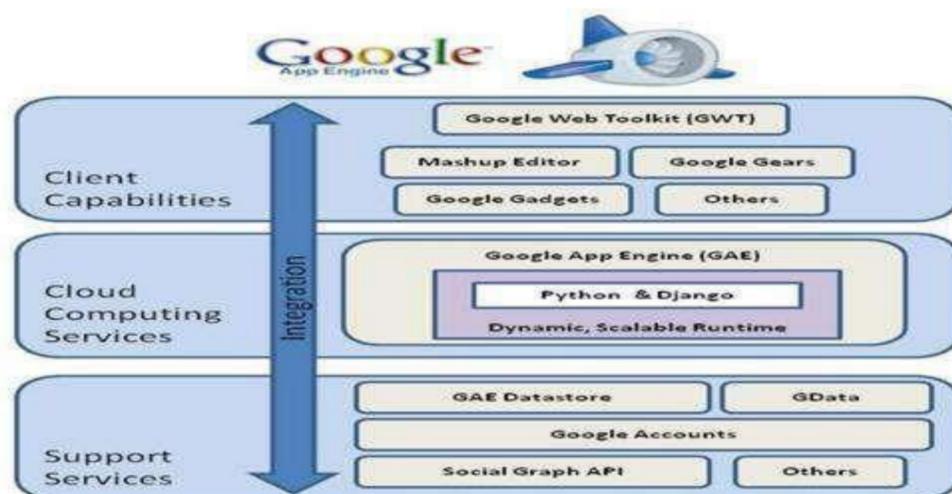
- Google App Engine lets you run your web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: You just upload your application, and it's ready to serve your users.
- You can serve your app from your own domain name (such as <https://www.example.com/>) using Google Apps. Or, you can serve your app using a free name on the [appspot.com](http://appspot.com) domain. You can share your application with the world, or limit access to members of your organization.
- Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language—or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that your application runs quickly, securely, and without interference.

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>03/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

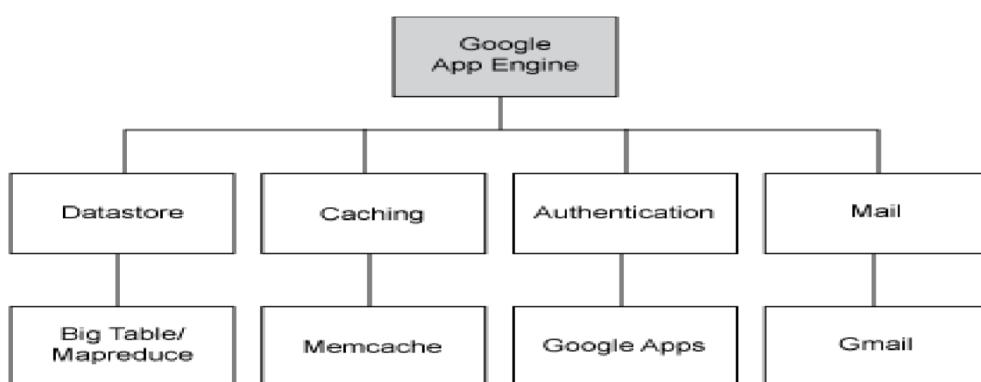
from other apps on the system.

- With App Engine, you only pay for what you use. There are no set-up costs and no recurring fees. The resources your application uses, such as storage and bandwidth, are measured by the gigabyte, and billed at competitive rates. You control the maximum amounts of resources your app can consume, so it always stays within your budget. App Engine costs nothing to get started. All applications can use up to 500 MB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month, absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.

#### **□ Architecture of Google App Engine**



#### **□ Features of Google App Engine**



<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>04/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

 **GAE Application Environment:**

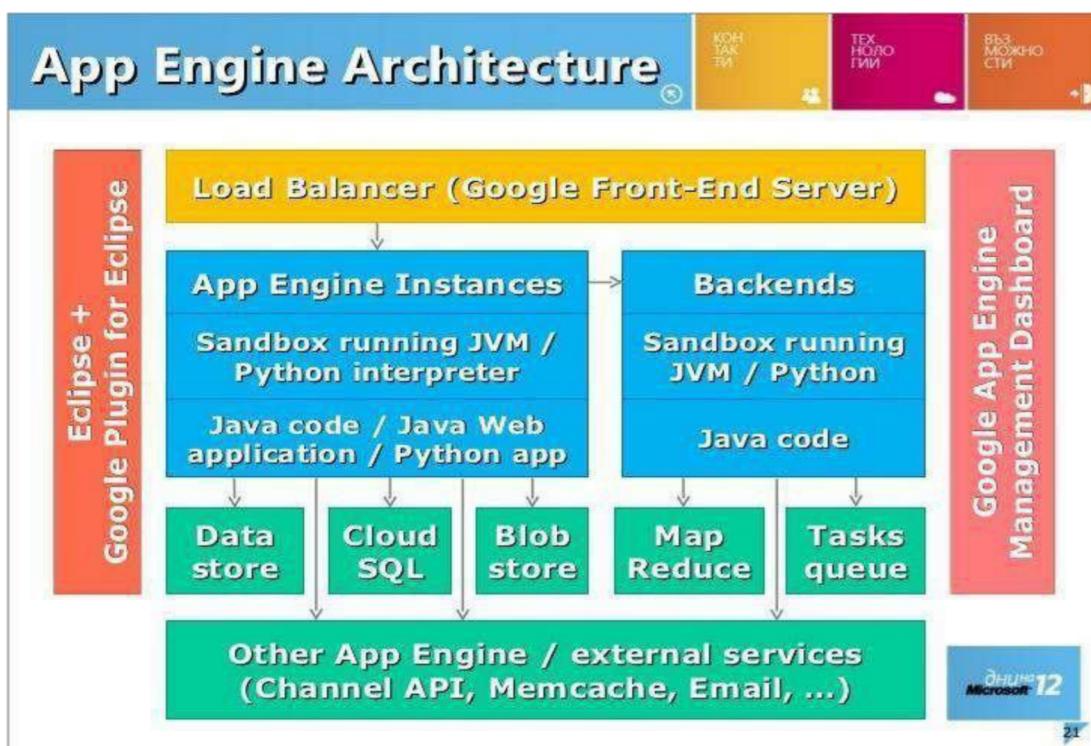
- Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. App Engine includes the following features:
- Persistent storage with queries, sorting and transactions
- Automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- Task queues for performing work outside of the scope of a web request
- Scheduled tasks for triggering events at specified times and regular intervals
- Dynamic web serving, with full support for common web technologies

 **Java Runtime Environment**

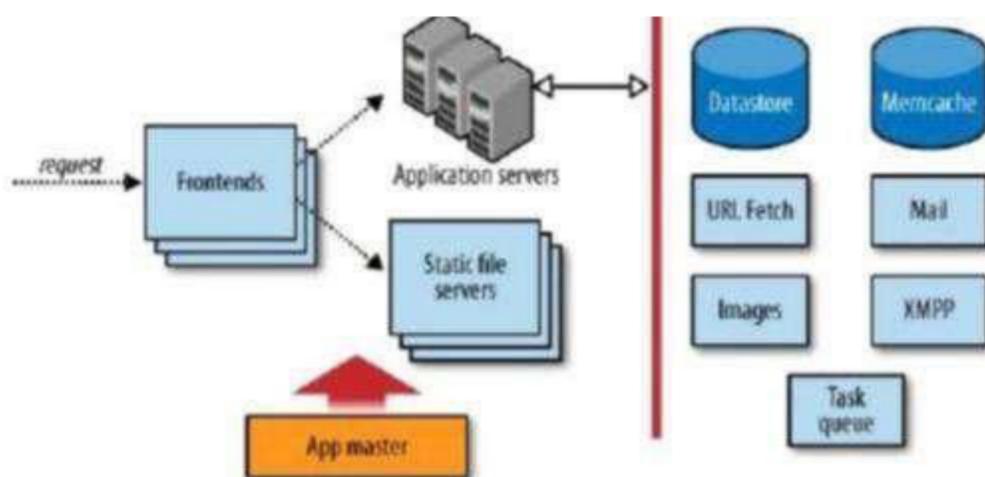
- You can develop your application for the Java runtime environment using common Java web development tools and API standards. Your app interacts with the environment using the Java Servlets standard, and can use common web application technologies such as Java Server Pages
- The Java runtime environment uses Java 6. The App Engine Java SDK supports developing apps using either Java 5 or 6. The environment includes the Java SE Runtime Environment (JRE) 6 platform and libraries. The restrictions of the sandbox environment are implemented in the JVM. An app can use any JVM byte code or library feature, as long as it does not exceed the sandbox restrictions. For instance, byte code that attempts to open a socket or write to a file will throw a runtime exception.
- Your app accesses most App Engine services using Java standard APIs. For the App Engine data store, the Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces. Your app can use the JavaMail API to send email messages with the App Engine Mail service. The java.net HTTP APIs access the App Engine URL fetch service.
- App Engine also includes low-level APIs for its services to implement additional adapters, or to use directly from the application. See the

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>05/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

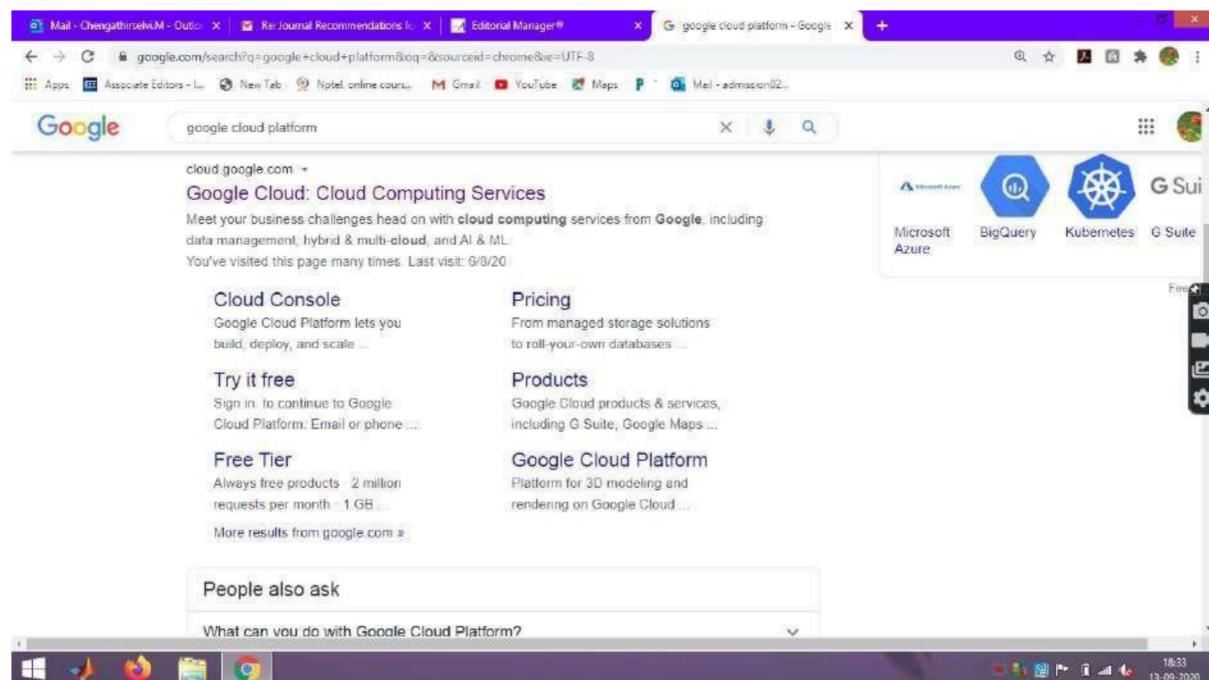
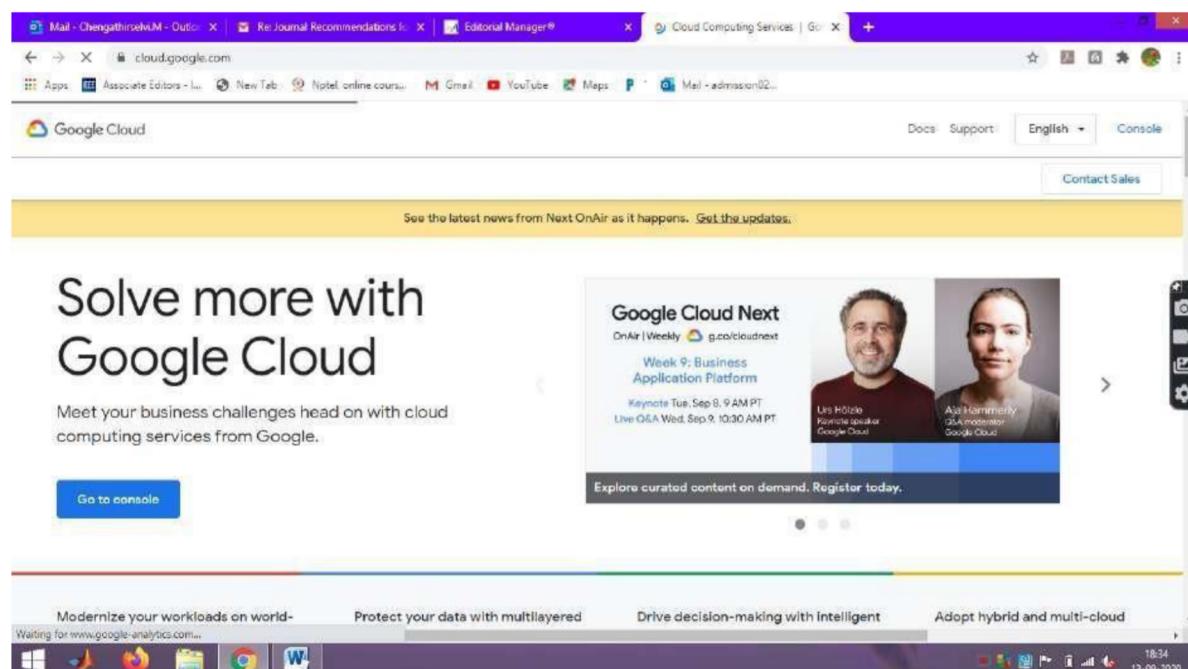
documentation for the data store, memcache, URL fetch, mail, images and Google Accounts APIs. Typically, Java developers use the Java programming language and APIs to implement web applications for the JVM. With the use of JVM-compatible compilers or interpreters, you can also use other languages to develop web applications, such as JavaScript, Ruby.



#### □ Workflow of Google App Engine



<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>06/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step1 : Login to [www.cloud.google.com](http://www.cloud.google.com)****Step2 : Goto Console****PREPARED BY****APPROVED BY****CONTROLLED COPY STAMP****MASTER COPY STAMP**

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>07/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

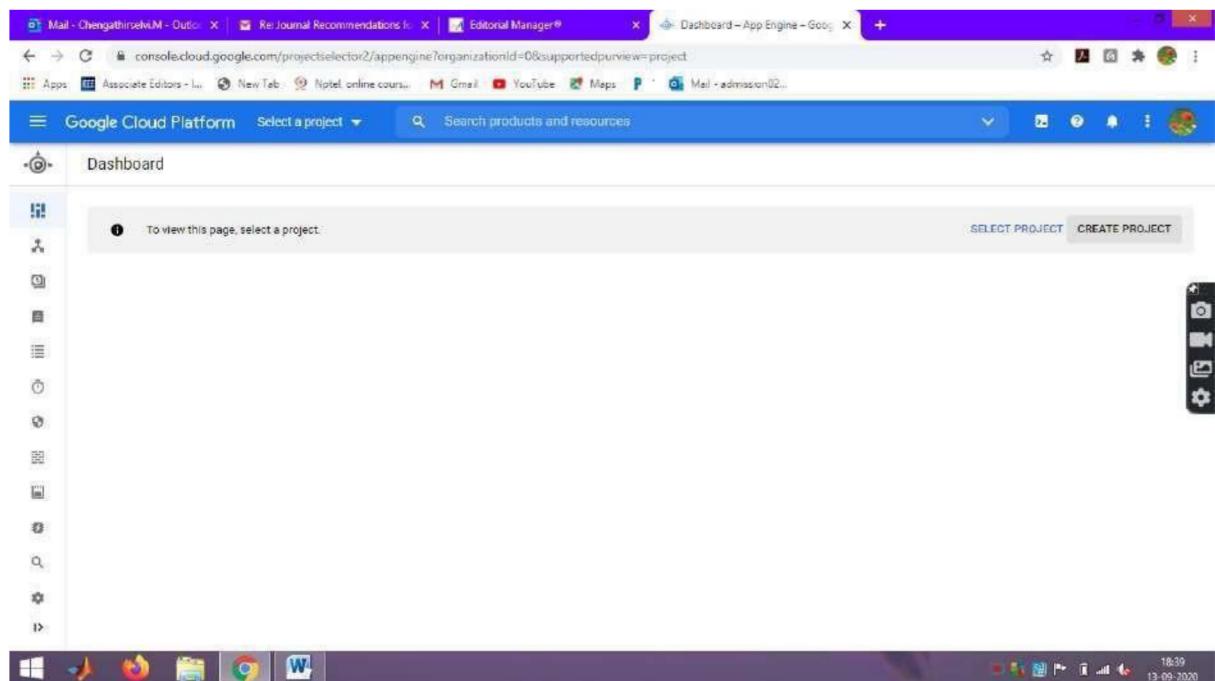
**Step 3 : Google Cloud Platform is shown**

The screenshot shows the Google Cloud Platform dashboard for the project 'Kct-kathir'. The 'Project info' section displays the project name 'Kct-kathir', project ID 'kct-kathir', and project number '625801604998'. The 'API APIs' section shows requests per second over time, with a note that no data is available for the selected time frame. The 'Google Cloud Platform status' section indicates all services are normal. The 'Billing' section shows estimated charges of INR 70.00 for the period 1-13 Sep 2020. The 'Monitoring' section allows setting up alerting policies and creating uptime checks. The dashboard also includes sections for Resources (Storage), Trace, and a sidebar with various Google services like Mail, Calendar, and Sheets.

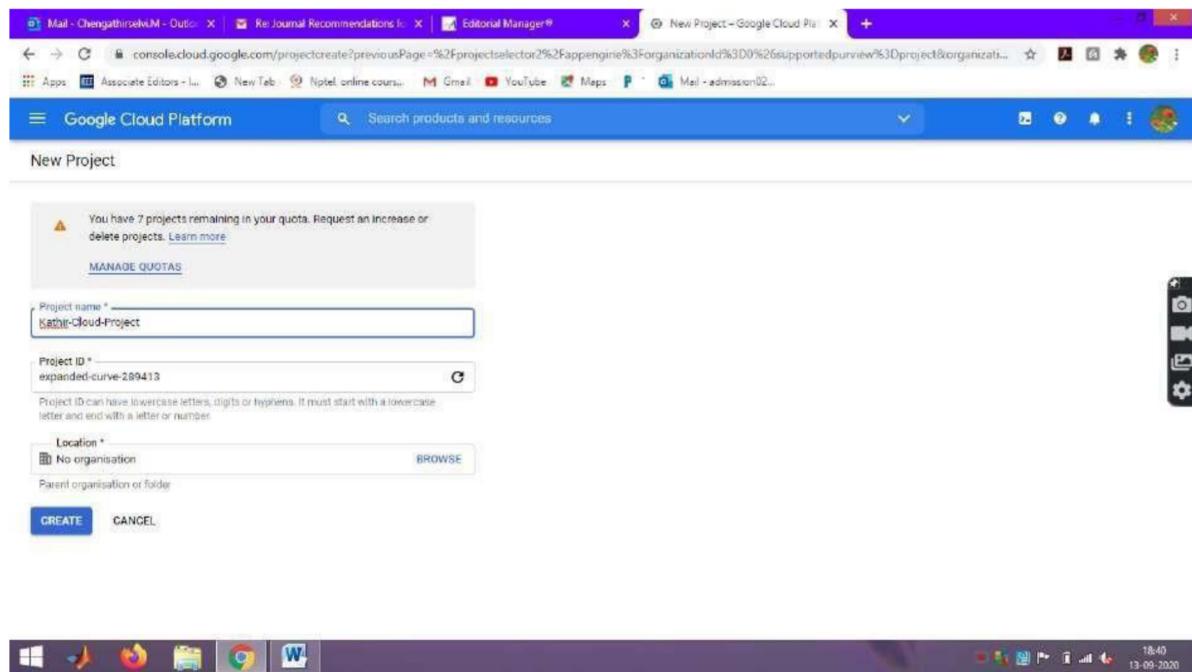
**Step 4 : Click Dashboard in the Google Cloud Platform**

The screenshot shows the Google Cloud Platform dashboard with the 'Dashboard' option selected from the left navigation menu. The main area displays a search bar and a table for managing resources. A message at the top right says 'No resource selected'. Below it are tabs for 'PERMISSIONS', 'LABELS', and 'ACTIVITY'. A note at the bottom left says 'Please select at least one resource.' The dashboard also includes sections for Services, Versions, Instances, Task queues, Cron jobs, Security scans, Firewall rules, Quotas, Memcache, Search, and Settings. The URL in the address bar is https://console.cloud.google.com/appengine?organizationId=0.

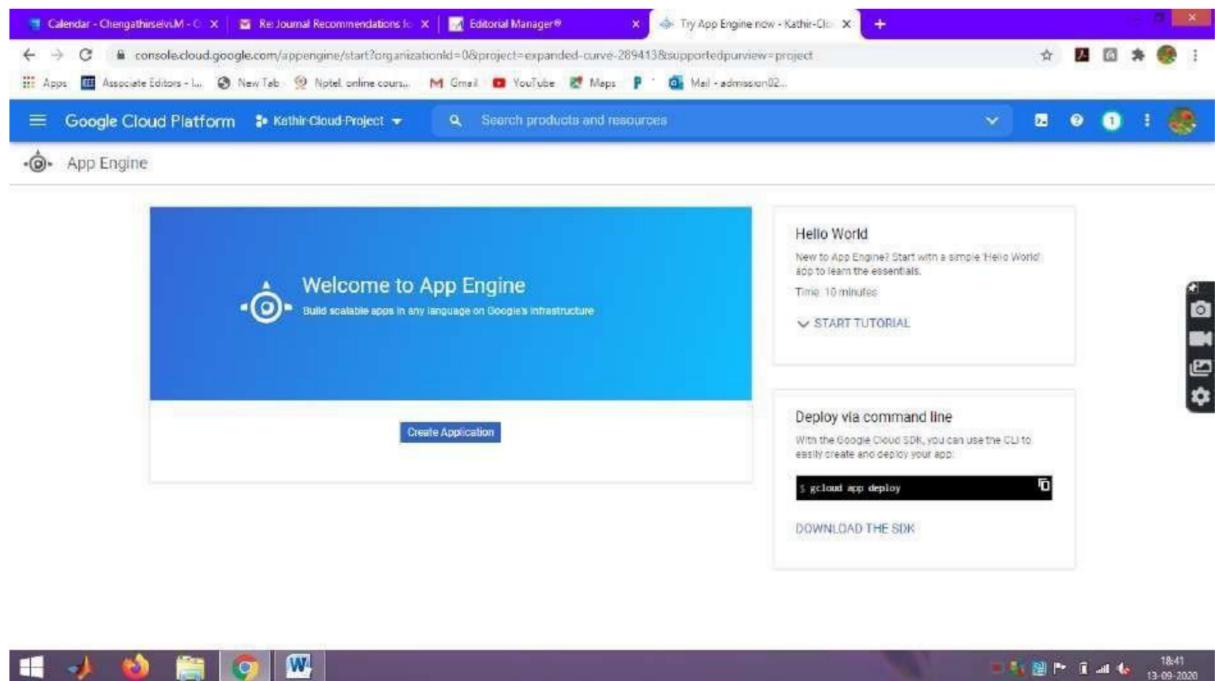
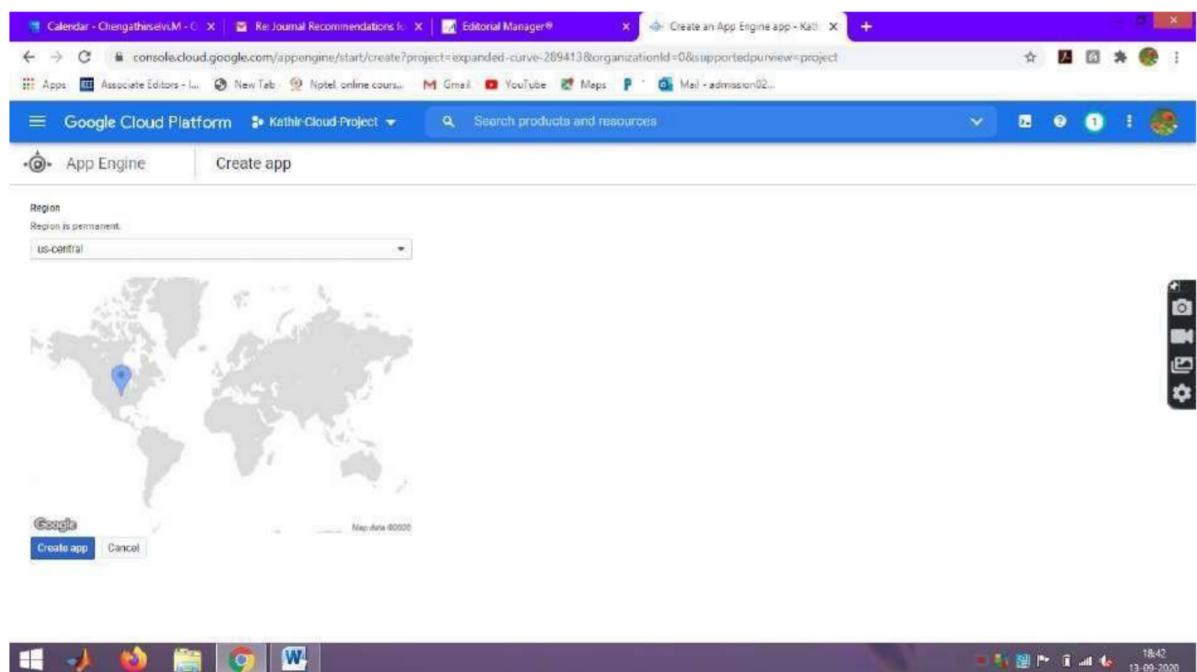
<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>08/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 5 : Dashboard in the Google Cloud Platform****Step 6 : Click New Project and give unique Project Name.**

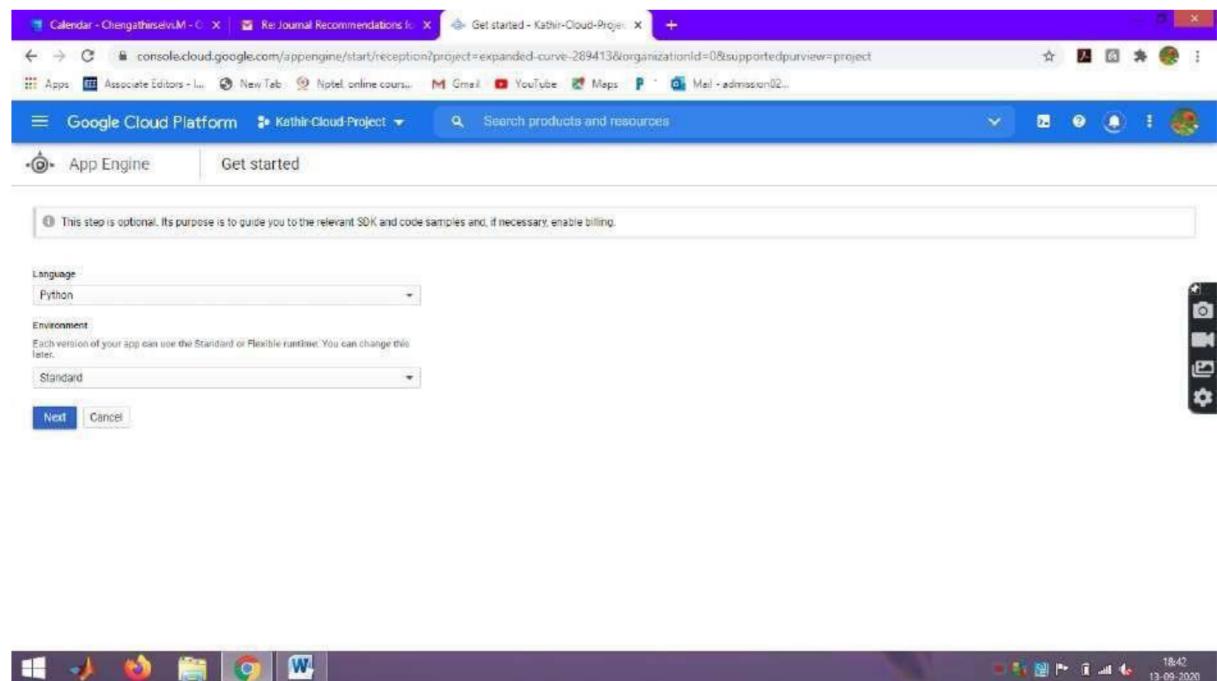
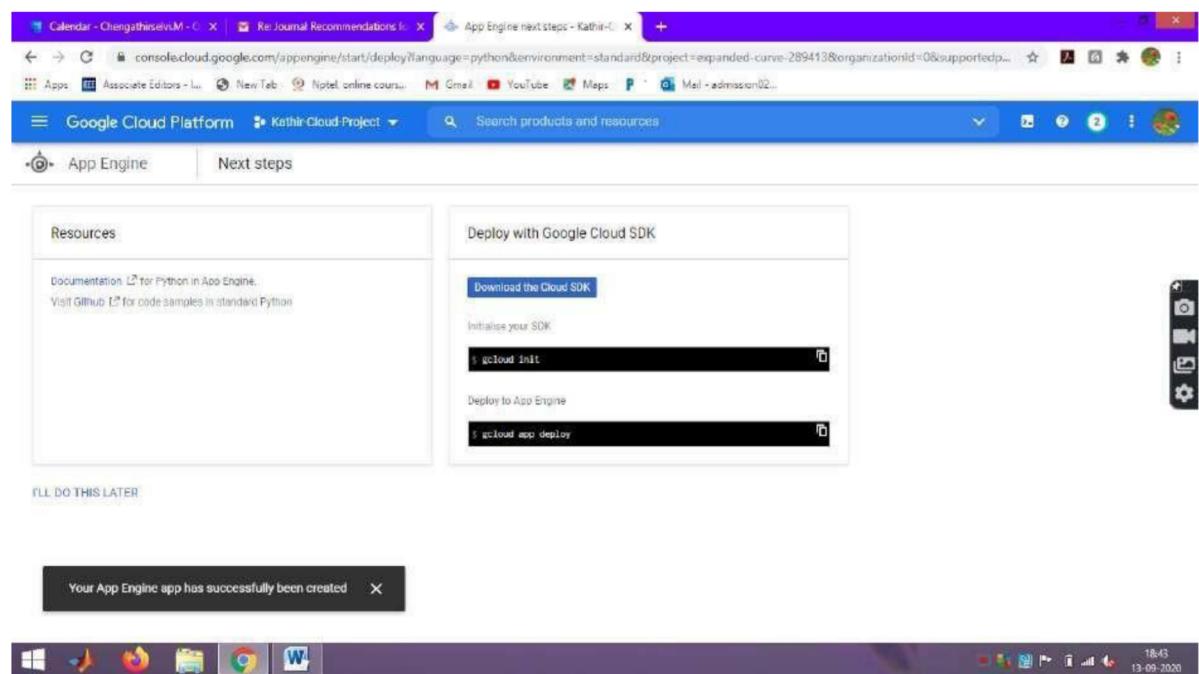
Example : kct-cloud-project



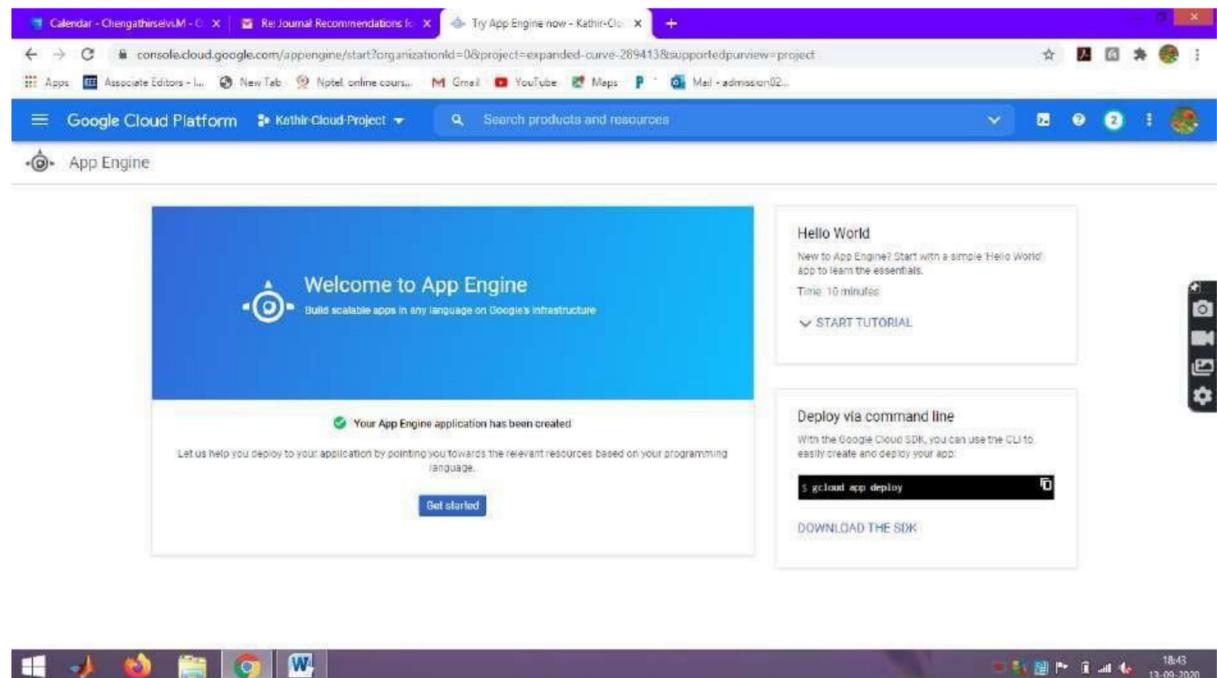
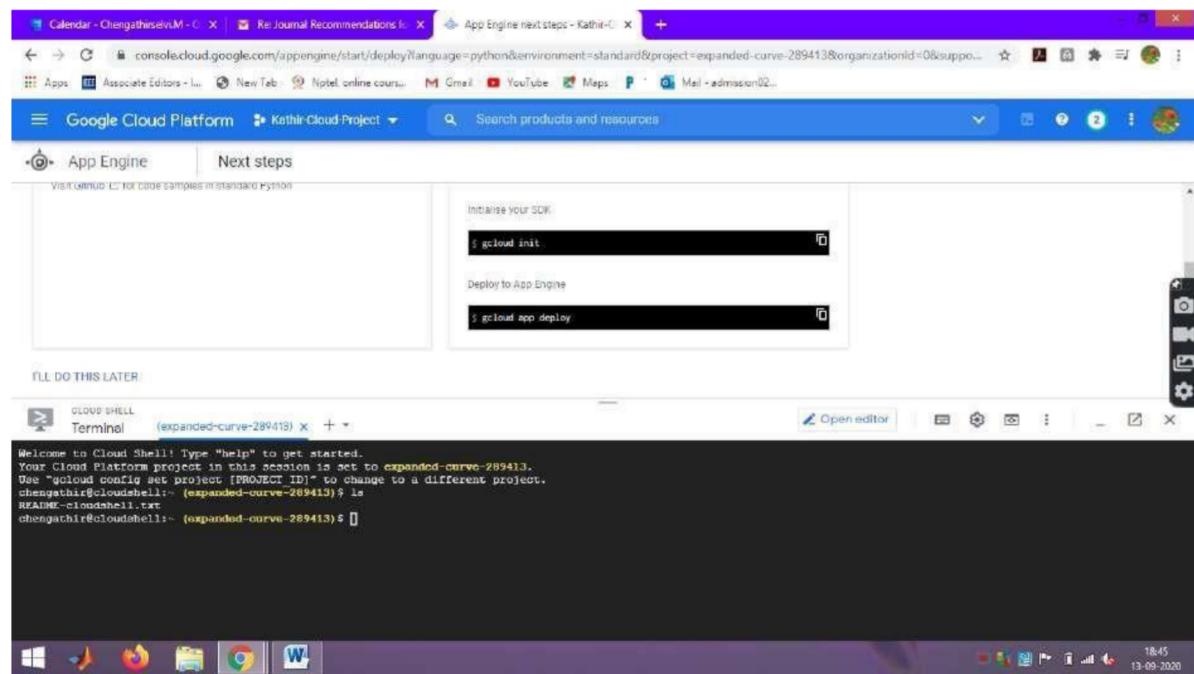
<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>09/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 7 : GoogleApp Engine is initiated****Step 8 : Click create Application**

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>10/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 9 : Create app and Select Language Python****Step 10 : Python app is created in Google App Engine**

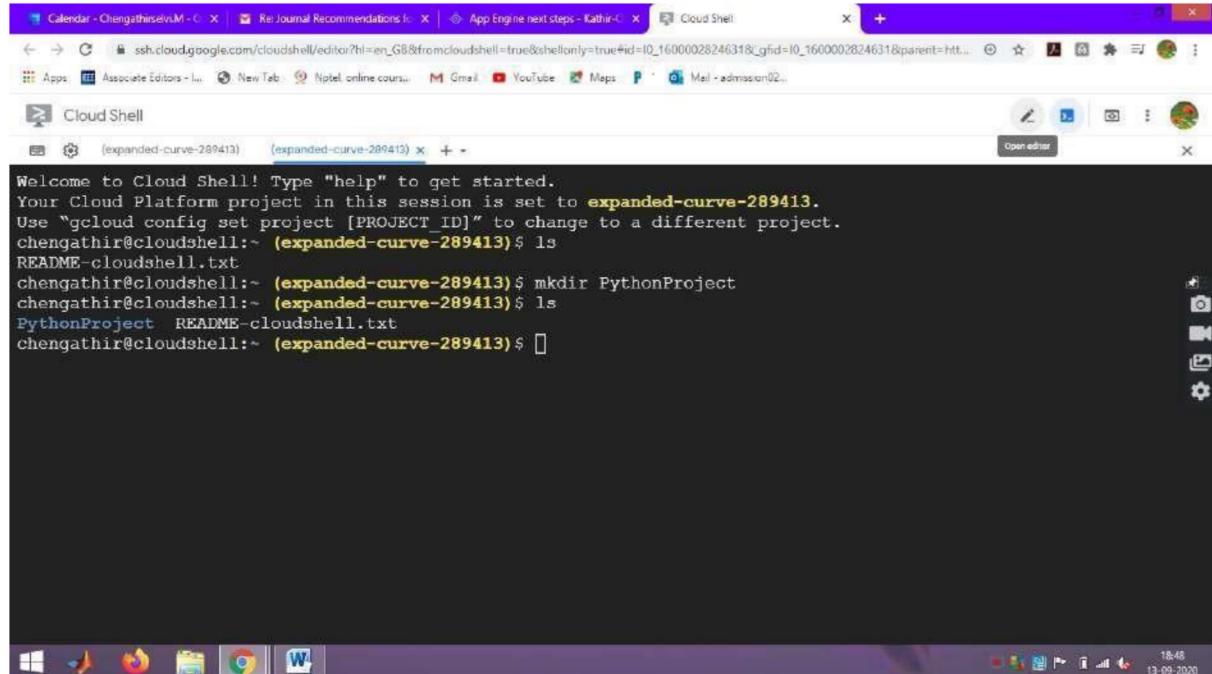
<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>11/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 11 : Python app Engine application is created****Step 12 : Click Cloud Shell in the Kathir-Cloud-Project**

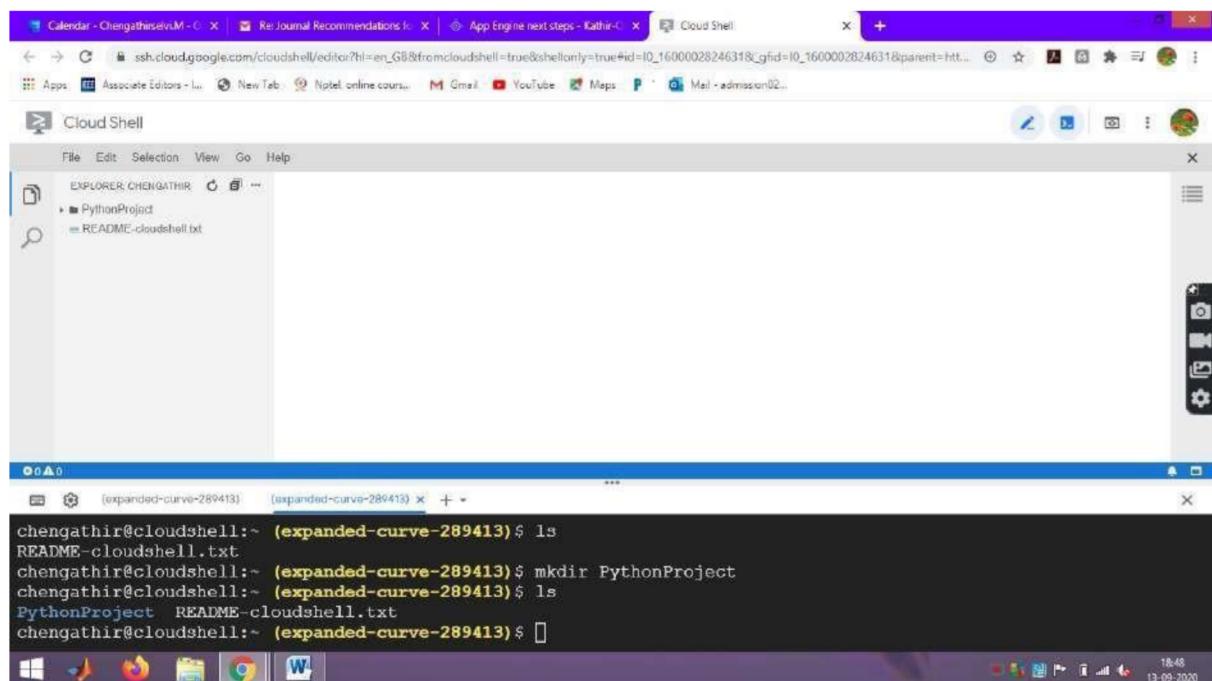
<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>12/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 13 : Create a Directory PythonProject using mkdir command**

Syntax : mkdir PythonProject



```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to expanded-curve-289413.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
chengathir@cloudshell:~ (expanded-curve-289413)$ ls  
README-cloudshell.txt  
chengathir@cloudshell:~ (expanded-curve-289413)$ mkdir PythonProject  
chengathir@cloudshell:~ (expanded-curve-289413)$ ls  
PythonProject README-cloudshell.txt  
chengathir@cloudshell:~ (expanded-curve-289413)$ 
```

**Step 14 : Click Editor to create Python application**

```
File Edit Selection View Go Help  
EXPLORER, CHENGATHIR, ⌂ ⌂ ⌂  
↳ PythonProject  
↳ README-cloudshell.txt  
  
chengathir@cloudshell:~ (expanded-curve-289413)$ ls  
README-cloudshell.txt  
chengathir@cloudshell:~ (expanded-curve-289413)$ mkdir PythonProject  
chengathir@cloudshell:~ (expanded-curve-289413)$ ls  
PythonProject README-cloudshell.txt  
chengathir@cloudshell:~ (expanded-curve-289413)$ 
```

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>13/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 15 : Click New File in the PythonProject Folder (Python file)**

The screenshot shows the Google Cloud Shell interface. In the top navigation bar, there are tabs for 'Calendar - ChengathirselvM - C', 'Re: journal Recommendations fo...', 'App Engine next steps - Kathir...', and 'Cloud Shell'. Below the tabs, the address bar shows the URL: ssh.cloud.google.com/cloudshell/editor?hl=en\_GB&fromCloudShell=true&shellonly=true#id=I0\_1600002824631&gfid=I0\_1600002824631&parent=http://...'. The main area is titled 'Cloud Shell' and contains a file explorer sidebar with 'PythonProject' selected. A context menu is open over a file named 'README'. The menu options include 'New File', 'New Folder', 'Open', 'Select for Compare', 'Find in Folder', 'Copy', 'Paste', 'Copy Download Link', 'Upload Files...', 'Download', 'Delete', 'Duplicate', and 'Rename'. The bottom half of the screen is a terminal window showing the command line:

```
chengathir@cloudshell:~ (expanded-curve-289413)$ ls
README-cloudshell.txt
chengathir@cloudshell:~ (expanded-curve-289413)$ mkdir PythonProject
chengathir@cloudshell:~ (expanded-curve-289413)$ ls
PythonProject README-cloudshell.txt
chengathir@cloudshell:~ (expanded-curve-289413)$
```

**Step 16 : Create main.py file**

The screenshot shows the Google Cloud Shell interface. The top navigation bar and address bar are identical to the previous screenshot. The main area is titled 'Cloud Shell' and contains a file explorer sidebar with 'PythonProject' selected. The 'main.py' file is currently selected in the file list. The code editor shows the following Python code:

```
import logging
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hello World'
if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)
```

The bottom half of the screen is a terminal window showing the command line:

```
chengathir@cloudshell:~ (expanded-curve-289413)$ cd Python[]
```

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>14/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**main.py file**

```
import logging

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'HelloWorld'

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)
```

**Step 17 : Create app.yaml file**

The screenshot shows the Google Cloud Shell interface. In the top navigation bar, there are tabs for Mail, App Engine next steps, Cloud Shell, WhatsApp, and a search bar. Below the navigation bar, there are links for Apps, Associate Editors, New Tab, Nptel online courses, Gmail, YouTube, Maps, and Mail - admission02. The main area is titled "Cloud Shell". It has a file explorer sidebar on the left showing a directory structure: PythonProj/r, app.yaml, main.py, requirements.txt, and README-cloudshell.txt. The central workspace contains an "app.yaml" file with the following content:

```
runtime: python
env: flex
entrypoint: gunicorn -b :$PORT main:app
runtime_config:
  python_version: 3
```

At the bottom, a terminal window shows the command: chengathir@cloudshell:~ (expanded-curve-289413) \$ cd Python

**app.yaml**

```
runtime: python
env: flex
entrypoint: gunicorn -b :$PORT&nbspmain:app

runtime_config:
  python_version: 3
```

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>15/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 18 : Create requirements.txt file**

The screenshot shows the Google Cloud Shell interface. In the terminal window, the user has navigated to a directory named 'PythonProject'. They have opened a text editor and created a new file named 'requirements.txt'. The file contains two lines of text: 'Flask==0.11.1' and 'gunicorn==19.6.0'. The terminal also displays the command 'cd PythonProject'.

```
chengathir@cloudshell:~ (expanded-curve-289413)$ cd PythonProject
```

**requirements.txt**

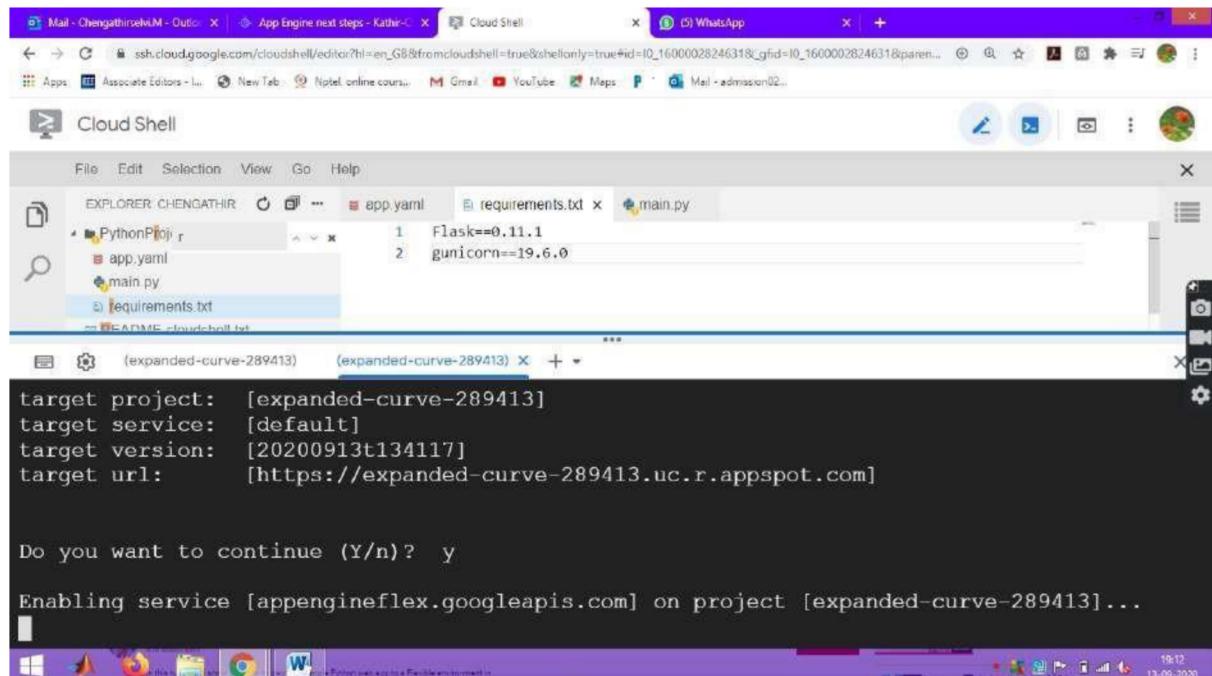
```
Flask==0.11.1
gunicorn==19.6.0
```

**Step 19 : Move to Cloud Shell Environment to run the application**

The screenshot shows the Google Cloud Shell interface again. The user has moved back to the root directory ('~') and run the command 'ls' to list the contents of the 'PythonProject' directory. The output shows files named 'app.yaml', 'main.py', 'requirements.txt', and 'README-cloudshell.txt'.

```
chengathir@cloudshell:~ (expanded-curve-289413)$ ls
app.yaml main.py requirements.txt README-cloudshell.txt
```

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>16/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

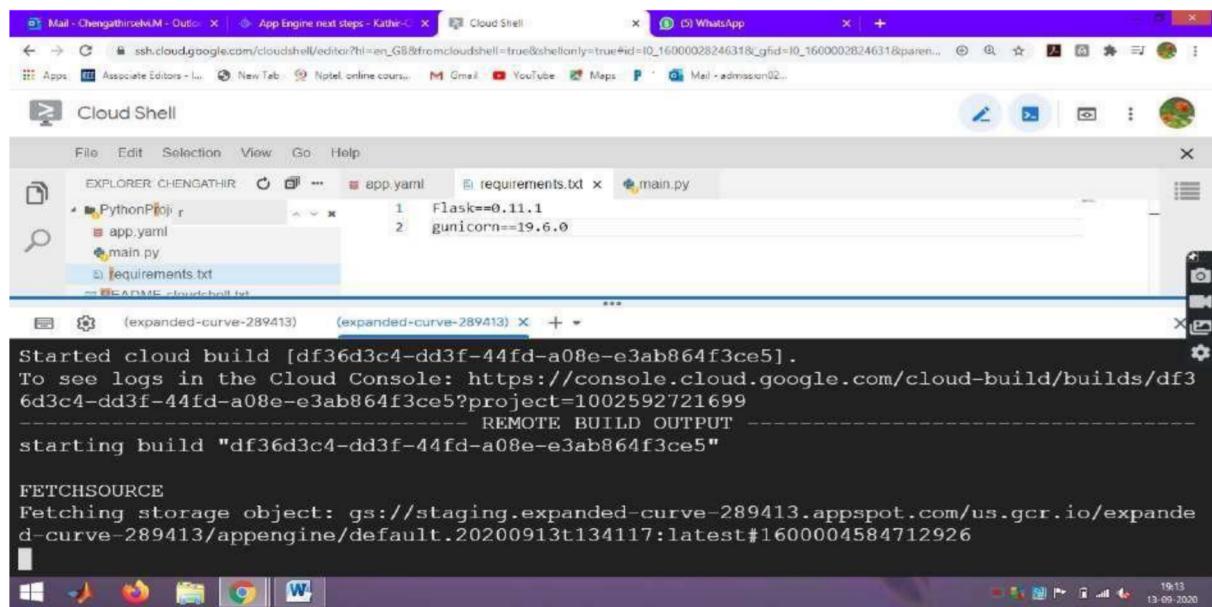
**Step 20 : Move to Cloud Shell Environment to run the application****Syntax : gcloud app deploy**

```
target project: [expanded-curve-289413]
target service: [default]
target version: [20200913t134117]
target url: [https://expanded-curve-289413.uc.r.appspot.com]

Do you want to continue (Y/n)? y

Enabling service [appengineflex.googleapis.com] on project [expanded-curve-289413]...
```

Continue the application. It enable service on the given project



```
Started cloud build [df36d3c4-dd3f-44fd-a08e-e3ab864f3ce5].
To see logs in the Cloud Console: https://console.cloud.google.com/cloud-build/builds/df36d3c4-dd3f-44fd-a08e-e3ab864f3ce5?project=1002592721699
----- REMOTE BUILD OUTPUT -----
starting build "df36d3c4-dd3f-44fd-a08e-e3ab864f3ce5"

FETCHSOURCE
Fetching storage object: gs://staging.expanded-curve-289413.appspot.com/us.gcr.io/expande
d-curve-289413/appengine/default.20200913t134117:latest#1600004584712926
```

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>17/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

It started building the object and fetching the storage object for the created application

The screenshot shows the Google Cloud Shell interface. In the file explorer, there is a folder named 'PythonProject' containing 'app.yaml', 'main.py', and 'requirements.txt'. The terminal window shows the deployment command being run:

```
89e14614ab6b: Layer already exists
18ecc1cf8b2f: Pushed
15942b628b0d: Layer already exists
ed16353646db: Pushed
latest: digest: sha256:a56ba6724428bce0a0f89a6258cab5d985097ed4dd85027c24bcd176ca06d4b6 size: 3457
DONE

Updating service [default] (this may take several minutes)....
```

The status bar at the bottom right indicates the date as 13-09-2020 and the time as 19:16.

It is updating the service

The screenshot shows the Google Cloud Shell interface. In the file explorer, there is a folder named 'PythonProject' containing 'app.yaml', 'main.py', and 'requirements.txt'. The terminal window shows the deployment command completed:

```
DONE

Updating service [default] (this may take several minutes)...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://expanded-curve-289413.uc.r.appspot.com]

You can stream logs from the command line by running:
$ gcloud app logs tail -s default

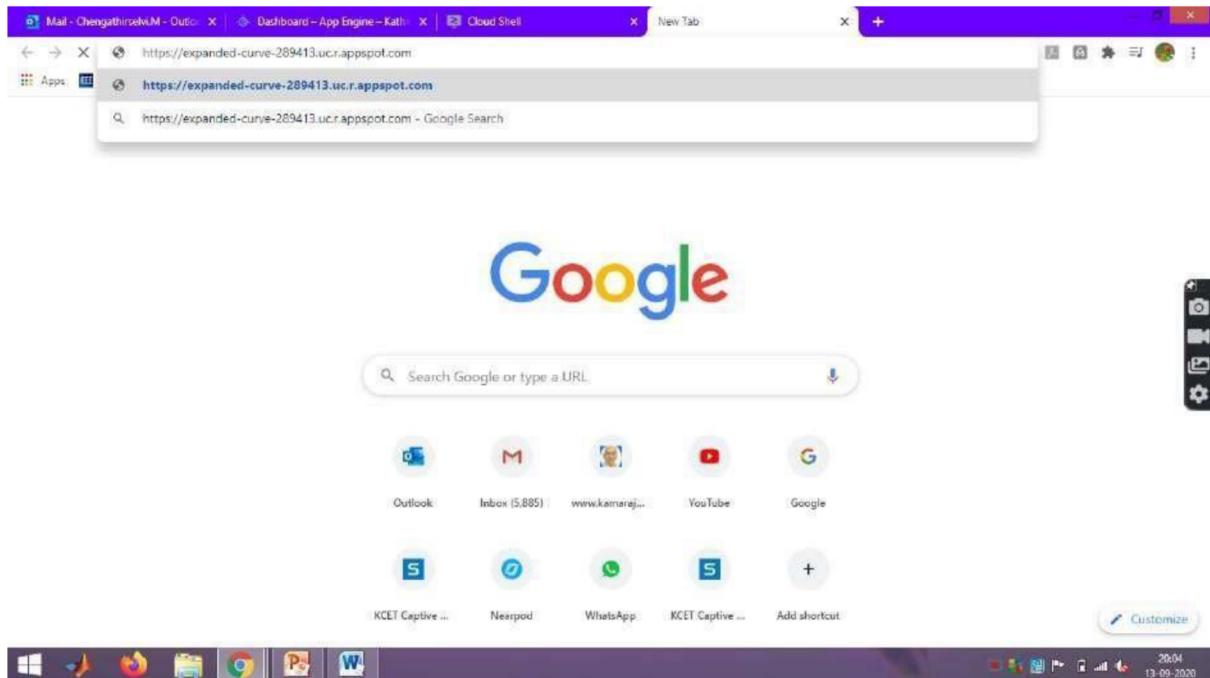
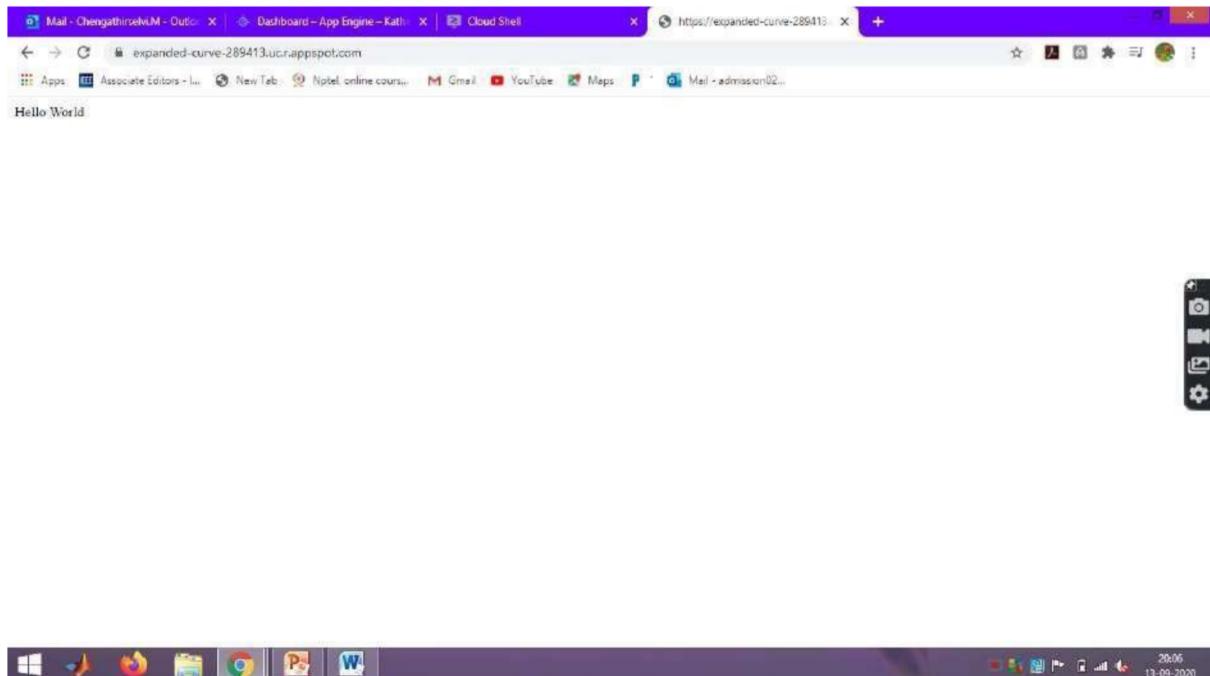
To view your application in the web browser run:
$ gcloud app browse
chengathir@cloudshell:~/PythonProject (expanded-curve-289413)$
```

The status bar at the bottom right indicates the date as 13-09-2020 and the time as 20:04.

The application is successfully deployed and URL is

<https://expanded-curve-289413.uc.r.appspot.com>

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>18/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 21: Run your program in the browser****Step 22: Hello World Program is sucessfully run in the browser**

<b>CM/ADL-D-05</b>	<b>Google App Engine &amp; GAE launcher</b>	<b>Page</b>	<b>19/19</b>
<b>Experiment No.: 01 &amp; 02</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Result:**

Thus the Google App Engine is installed successfully and a web application to display hello world using python is developed and deployed in the GAE and used GAE Launcher to launch the web applications.

<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>01/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Aim:**

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

**Theory:****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**

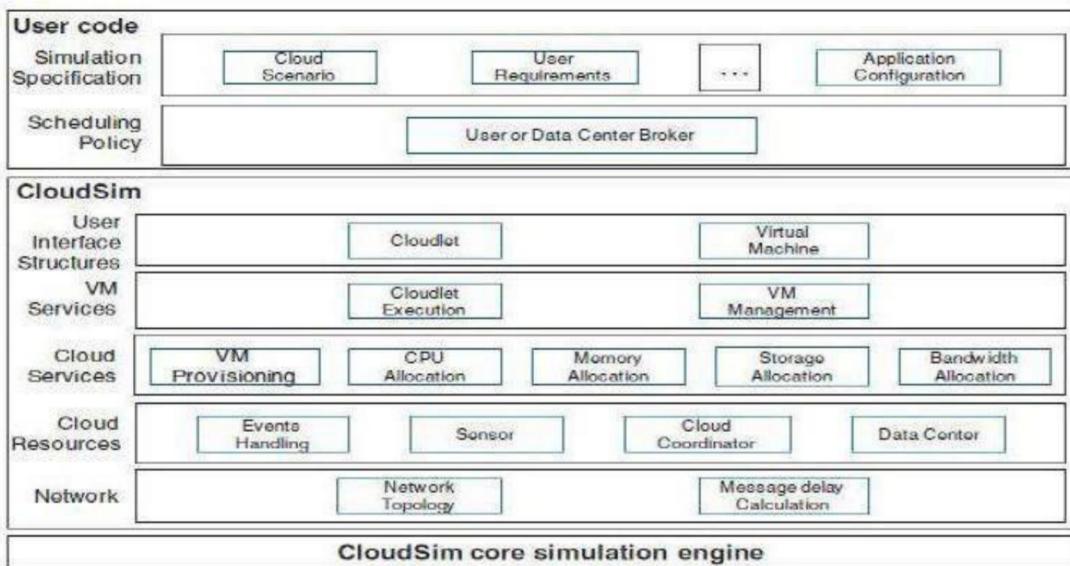
- JDK 1.6 or above <http://tinyurl.com/jNU-JAVA>
- Eclipse 4.2 or above <http://tinyurl.com/jNU-Eclipse>
- Alternatively NetBeans <https://netbeans.org/downloads>
- Up & Running with cloudsim guide: <https://goo.gl/TPL7Zh>

❖ **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**

<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>02/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>



#### ❖ **Cloudsim - Component model classes**

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

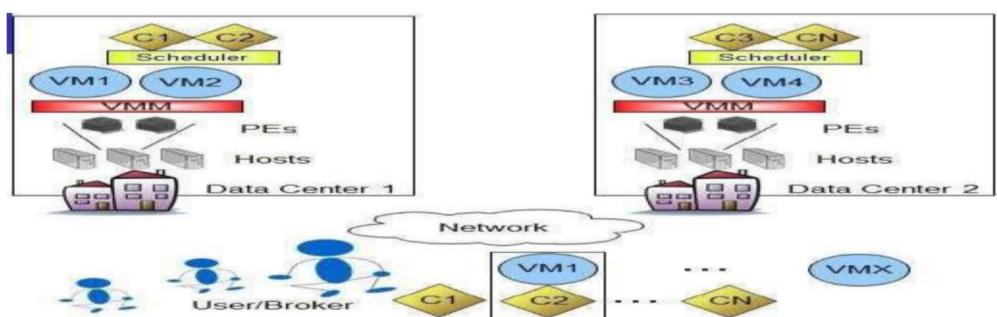
#### ❖ **Cloudsim - Major blocks/Modules**

- 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

<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>03/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

❖ **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.



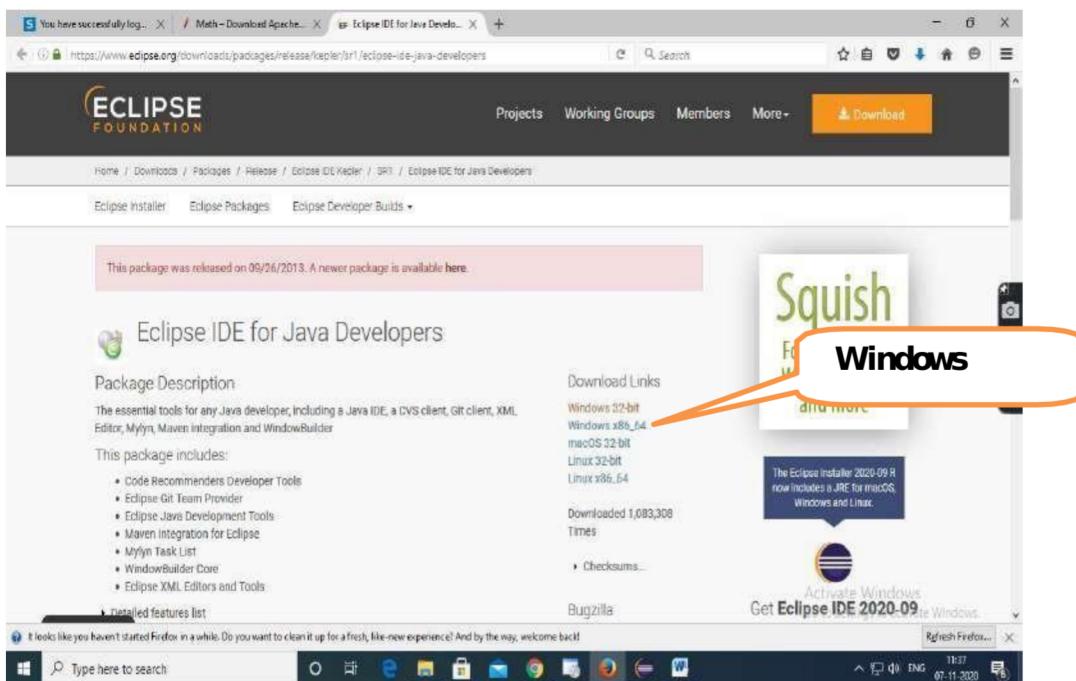
**CloudSim Elements/Components**

**Procedure to import Eclipse, Cloudsim in your system**

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

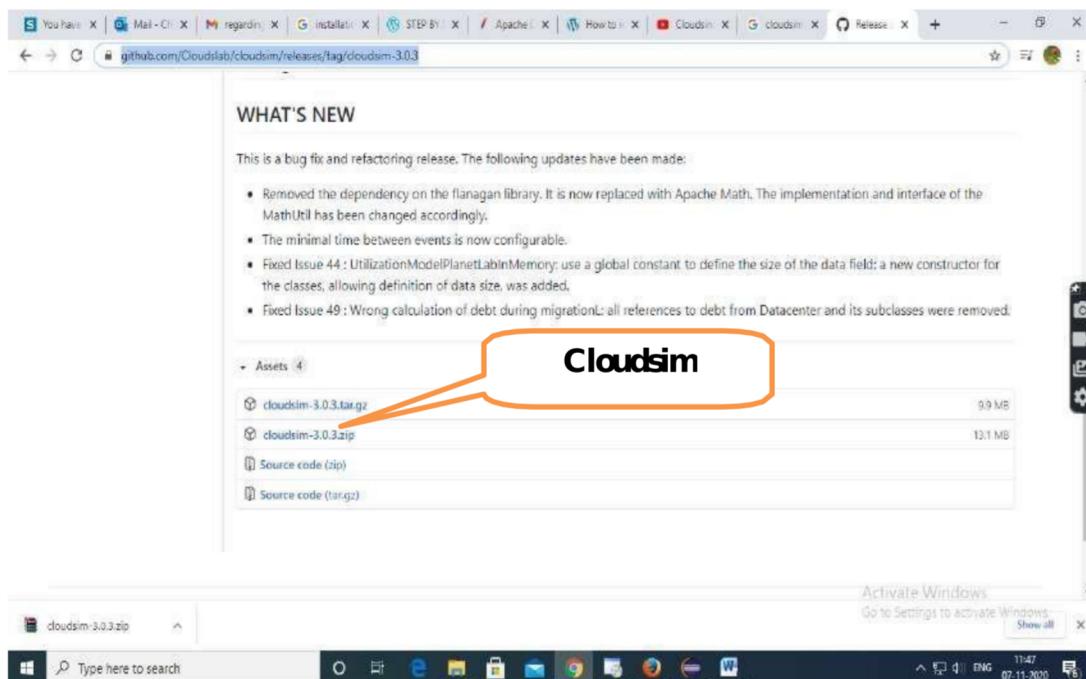
<https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers>

<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>04/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>



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

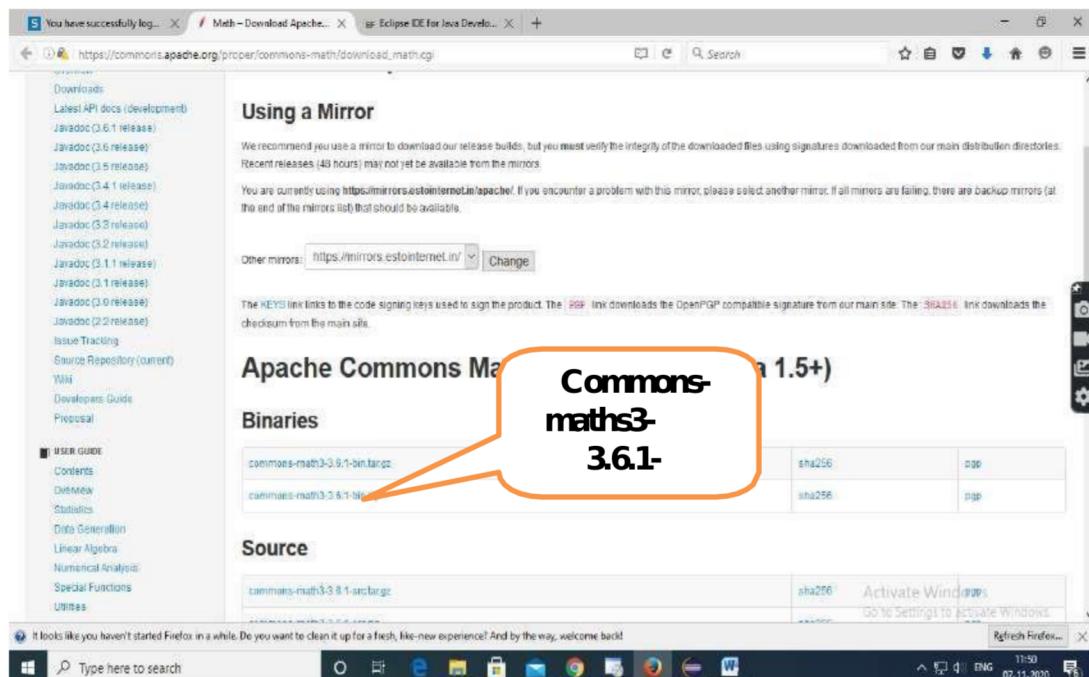
<https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3>



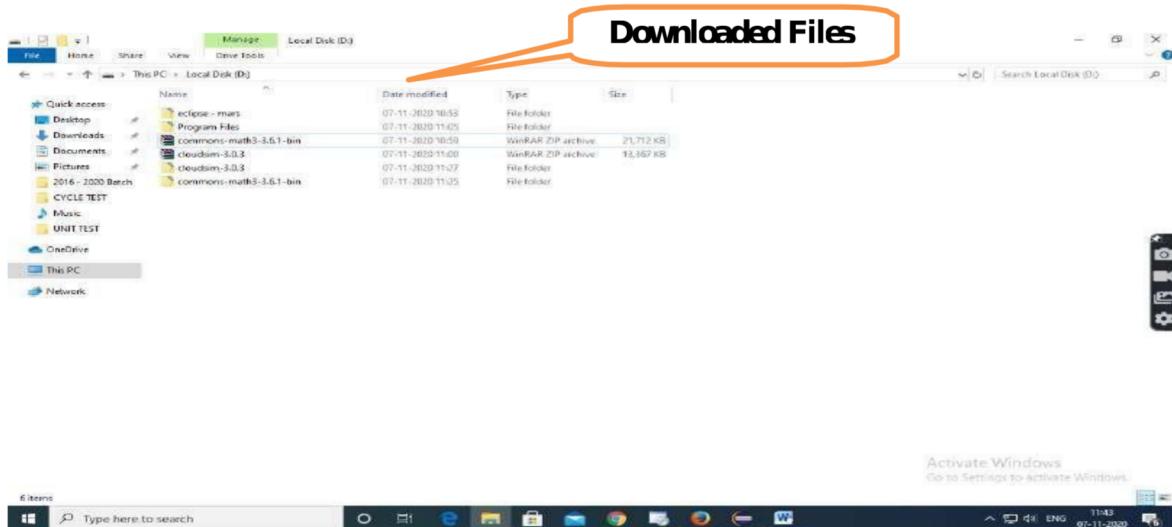
<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>05/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**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](https://commons.apache.org/proper/commons-math/download_math.cgi)

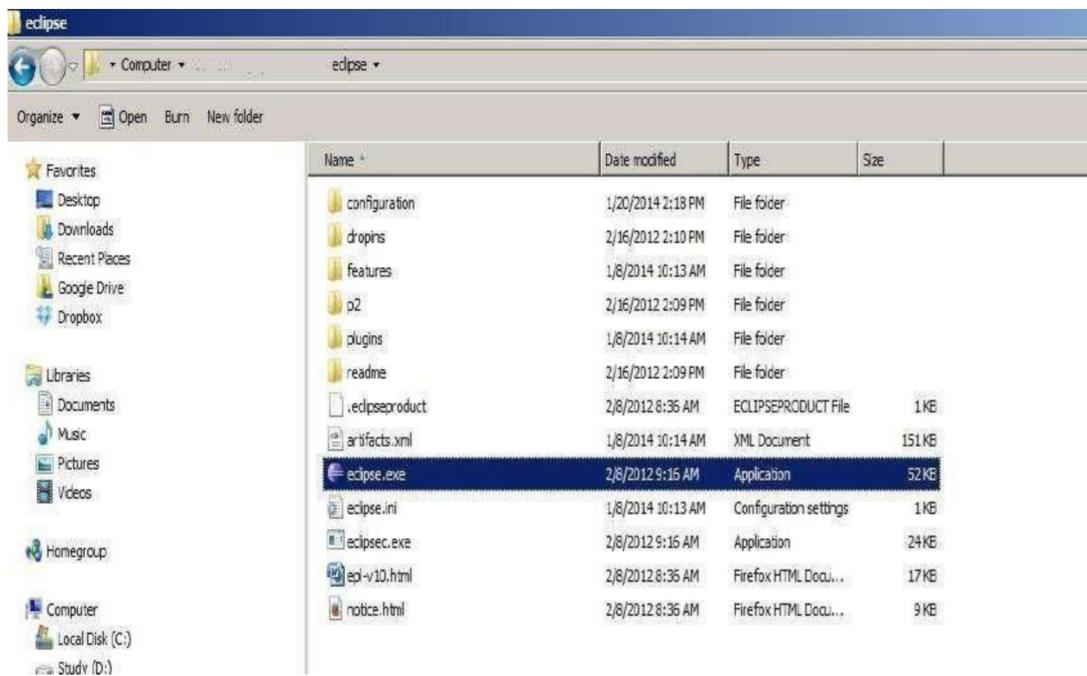


**Step 4:** Downloaded Eclipse, cloudsim-code-master and Apache Commons Math 3.6.1 in your local machine and extract cloudsim-3.0.3 and Apache Commons Math 3.6.1

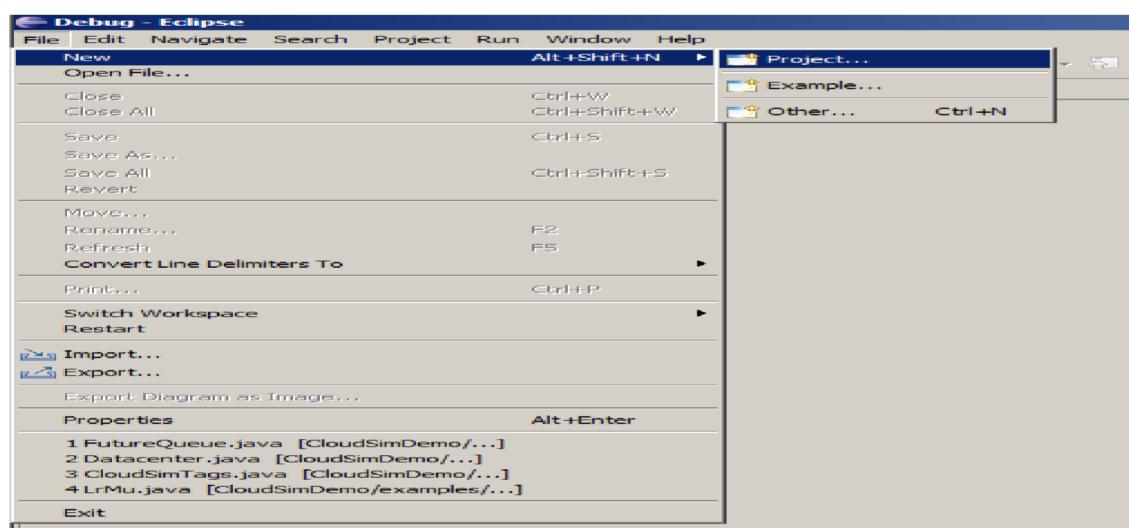


<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>06/14</b>
<b>Experiment No.: 03</b>	<b>Semester - II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

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

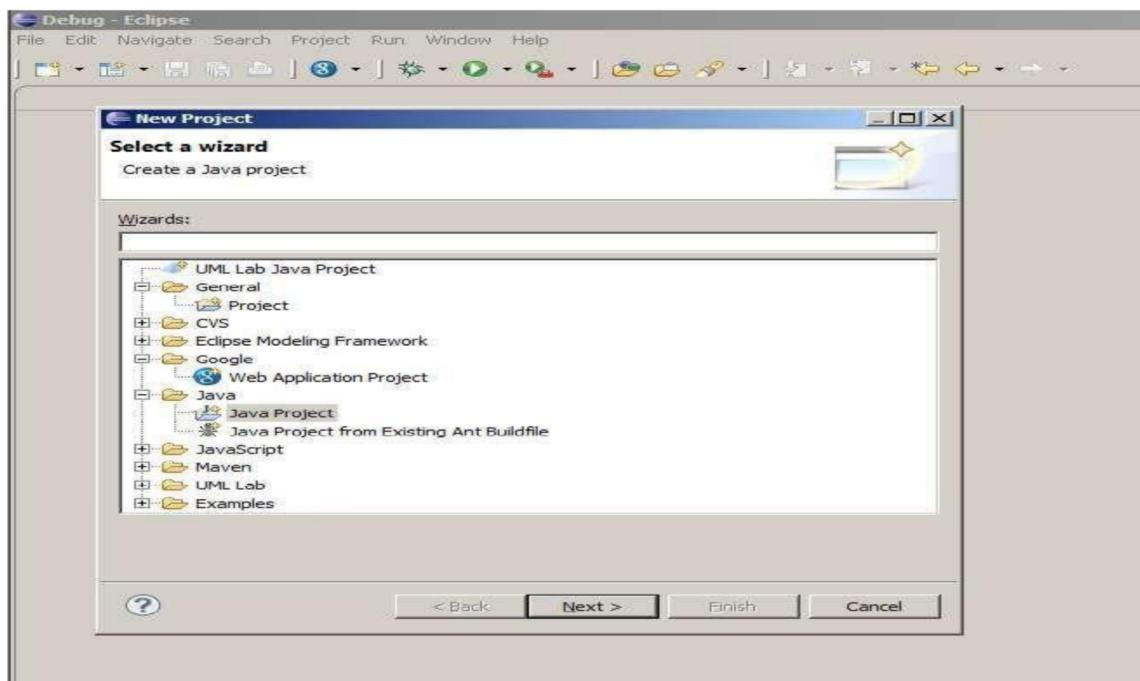


**Step 6:** Now within Eclipse window navigate the menu: *File->New->Project*, to open the new project wizard



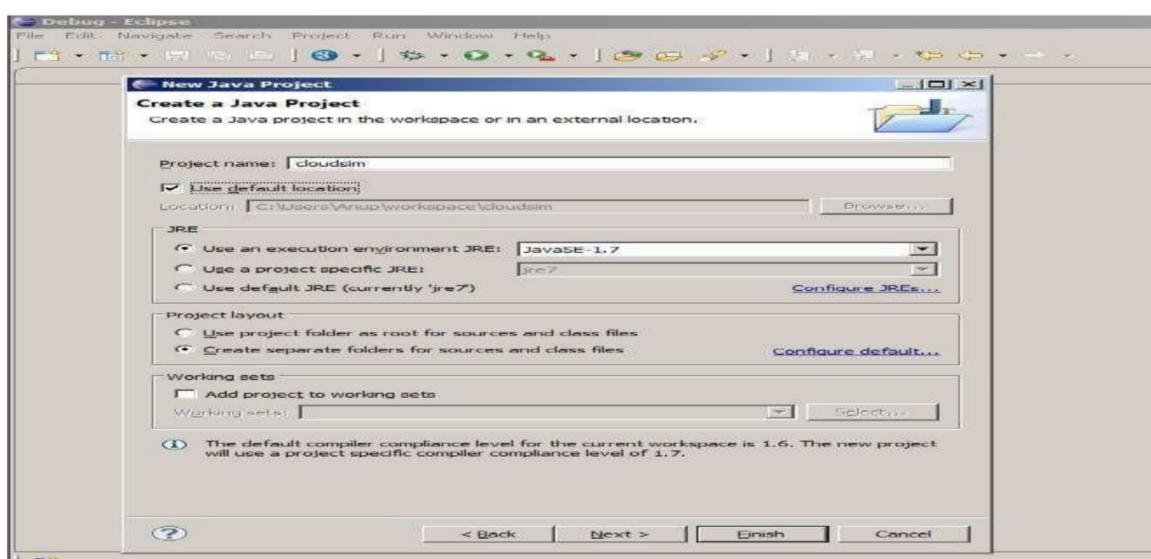
<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>07/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**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'\_



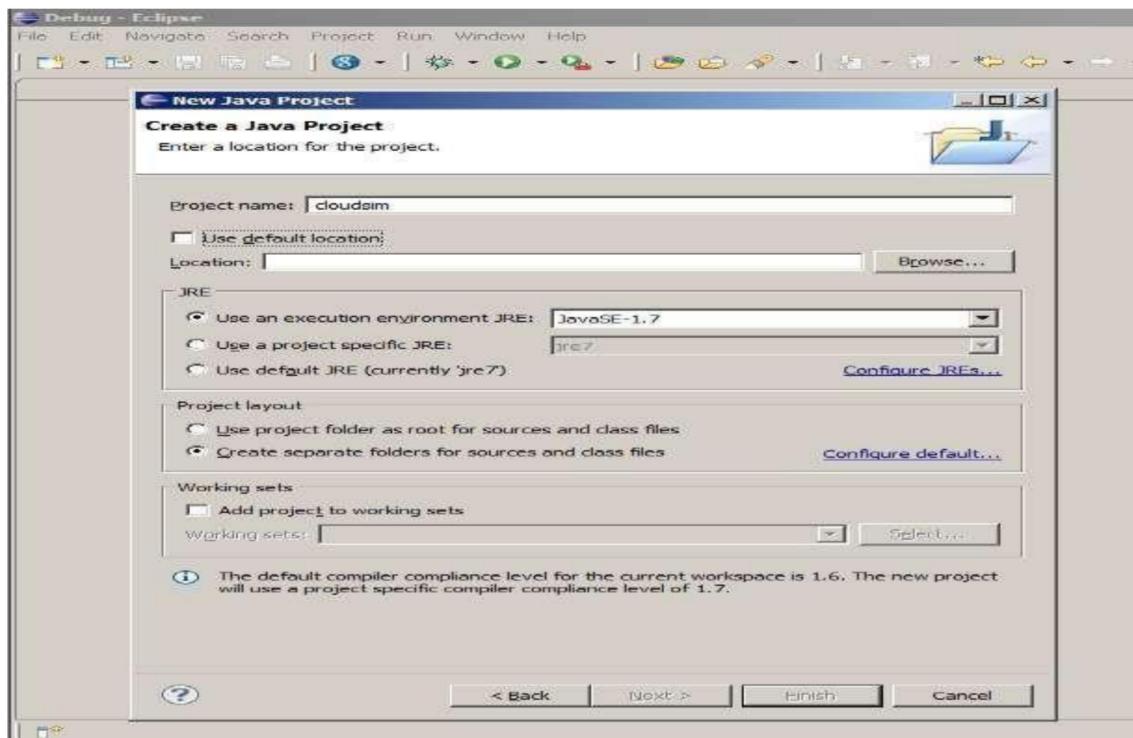
**Step 8:** Now a detailed new project window will open, here you will provide the project name and the path of CloudSim project source code, which will be done as follows:

**Project Name: CloudSim**

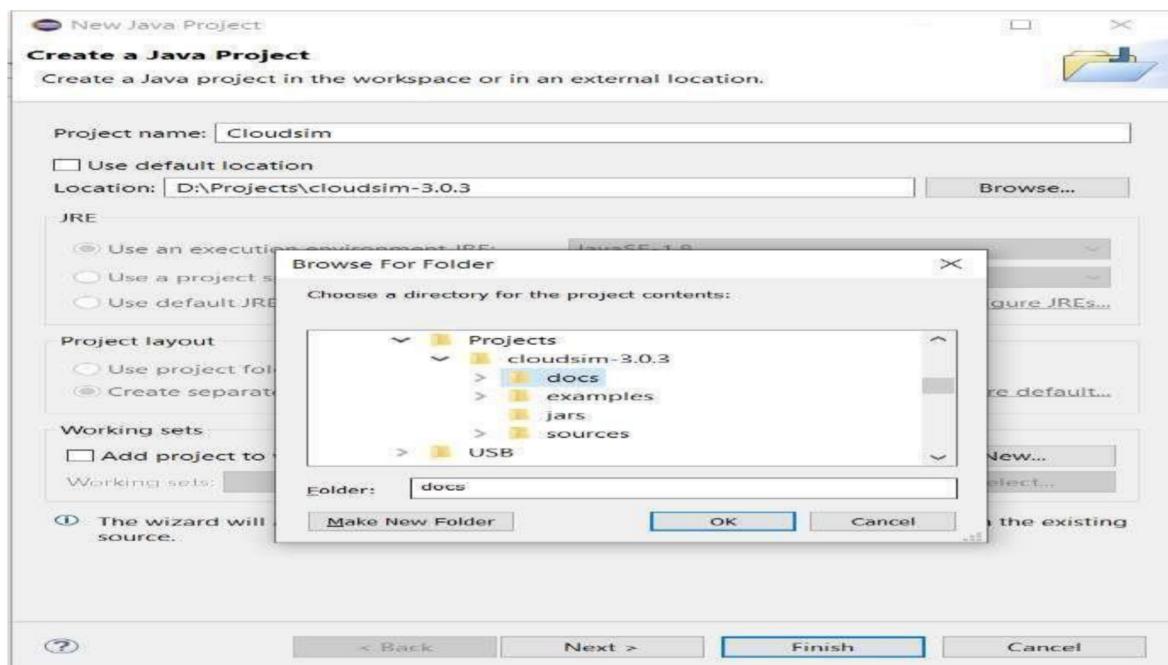


<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>08/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**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.

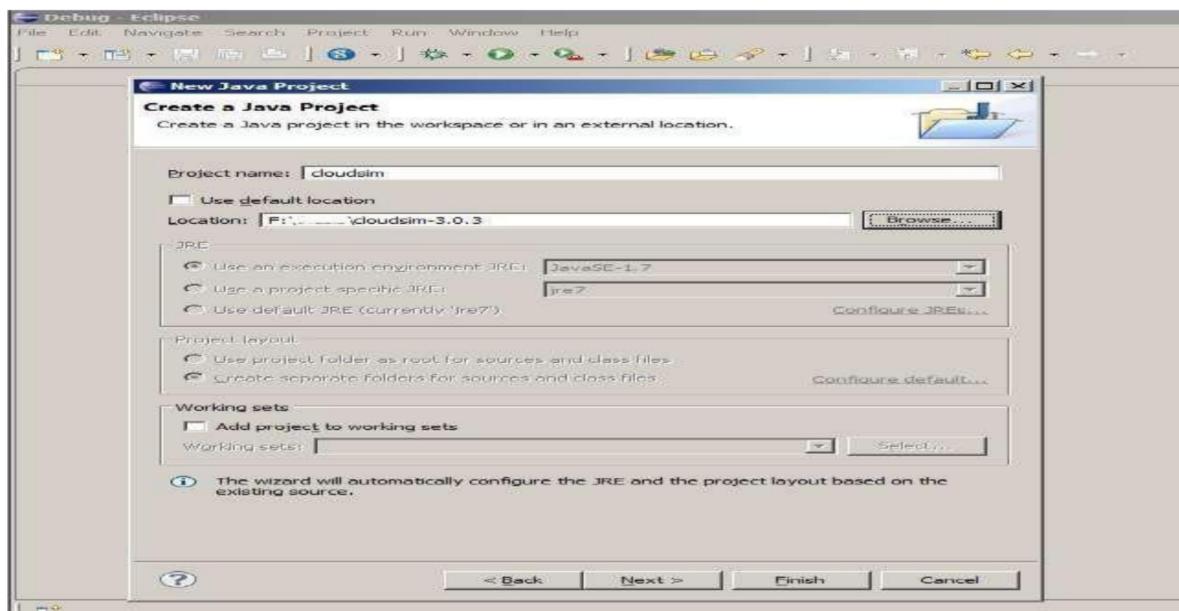


**Step 10:** Make sure you navigate the path till you can see the bin, docs, examples etc folder in the navigation plane.

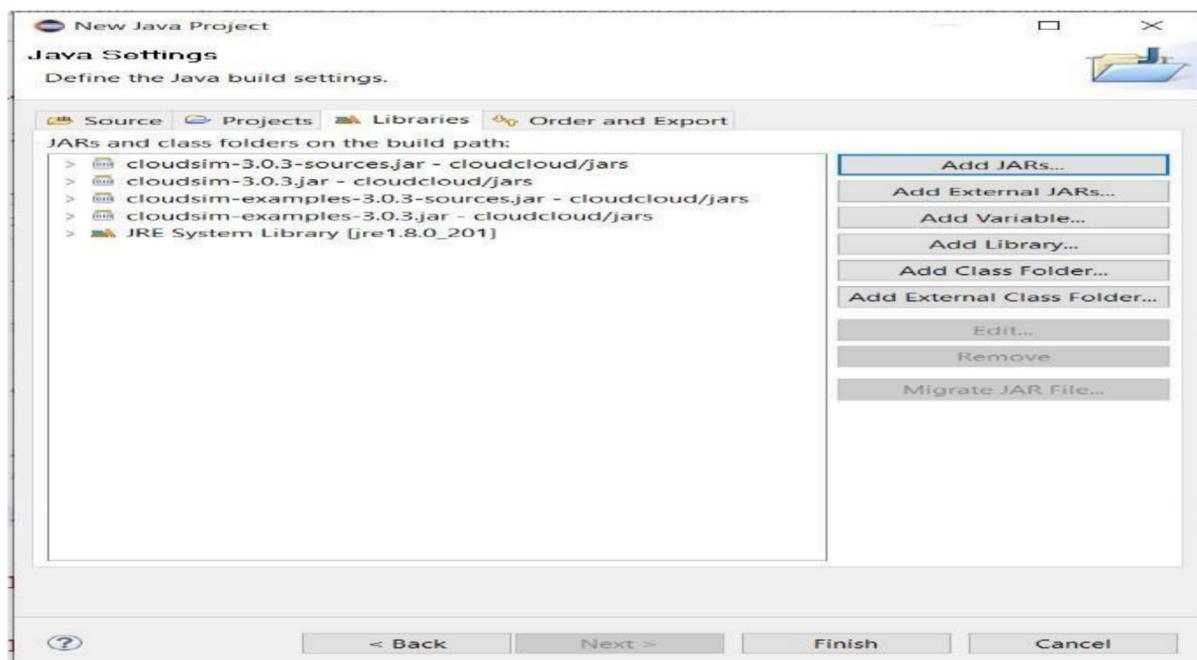


<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>09/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

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

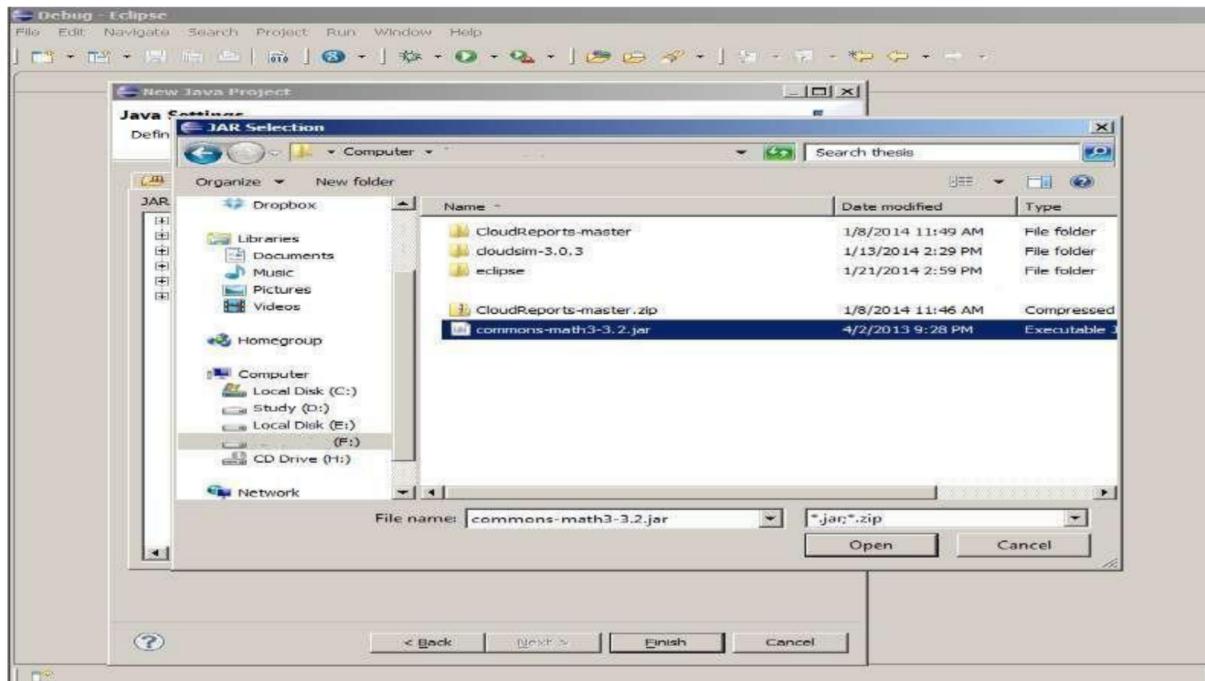


**Step 12:** Now open ‘Libraries’ tab and if you do not find commons-math3-3.x.jar (here ‘x’ means the minor version release of the library which could be 2 or greater) in the list then simply click on ‘Add External Jar’ (commons-math3-3.x.jar will be included in the project from this step)

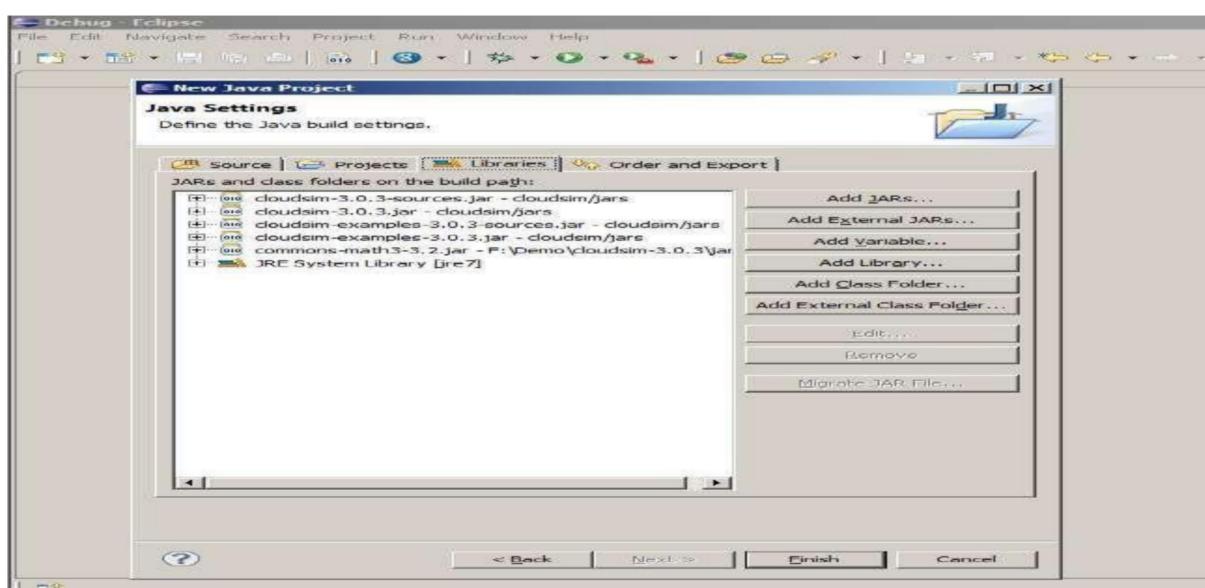


<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>10/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**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.



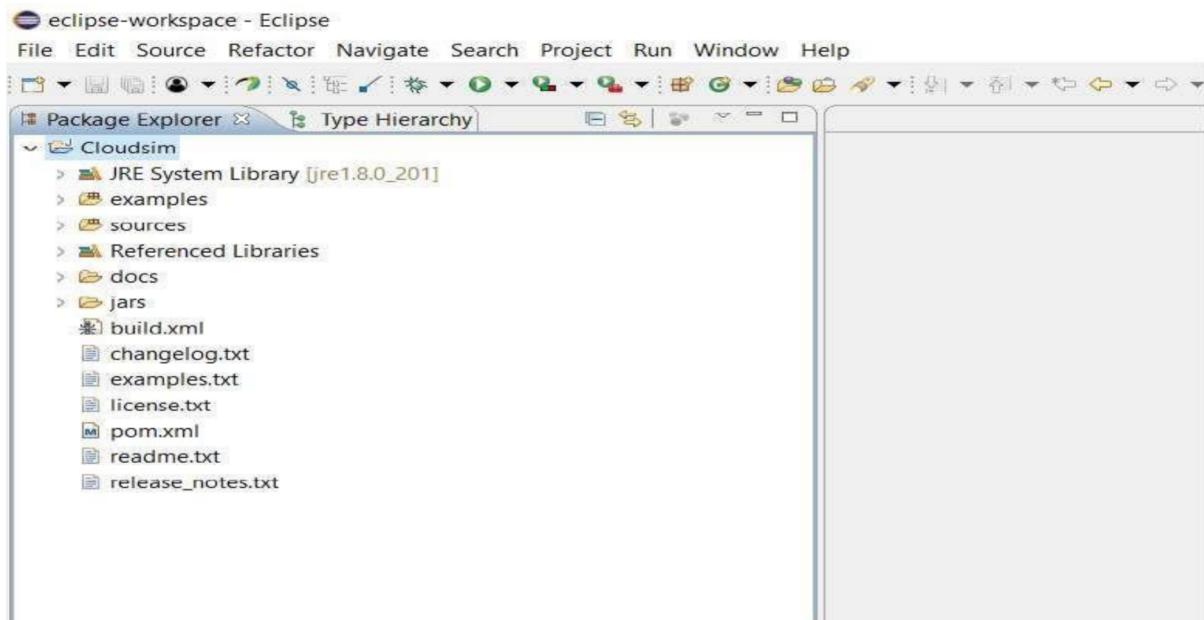
**Step 14:** Ensure external jar that you opened in the previous step is displayed in the list and then click on Finish (your system may take 2-3 minutes to configure the project)



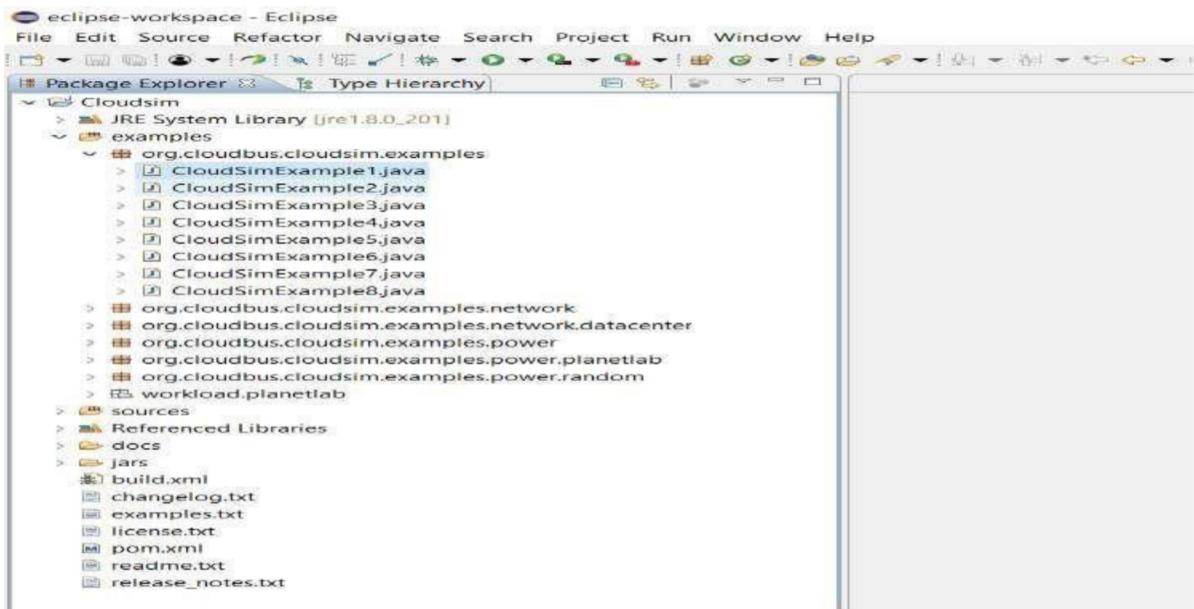
<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>11/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**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



<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>12/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

The screenshot shows the Eclipse IDE interface with the CloudSimExample1.java file open in the editor. The code implements a simple example to create a datacenter with one host and run one cloudlet on it. It includes imports for DecimalFormat, a main method, and CloudSim Toolkit. The main method initializes the CloudSim package, creates a cloudlet list, and runs it.

```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;

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

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

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

    /**
     * Creates main() to run this example.
     *
     * @param args the args
     */
    @SuppressWarnings("unused")
    public static void main(String[] args) {
        Log.println("Starting CloudSimExample1...");

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

            // Initialize the CloudSim library
            CloudSim.init(num_user, calendar, trace_flag);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The screenshot shows the Eclipse IDE interface with the CloudSimExample2.java file open in the editor. This code is similar to CloudSimExample1 but includes comments explaining the creation of a datacenter with one host and two cloudlets. It also includes imports for DecimalFormat and CloudSim Toolkit.

```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;

/*
 * A simple example showing how to create
 * a datacenter with one host and run two
 * cloudlets in it. The cloudlets run in
 * the host with the same MIPS requirements,
 * i.e. The cloudlets will take the same time to
 * complete the execution.
 */
public class CloudSimExample2 {

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

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

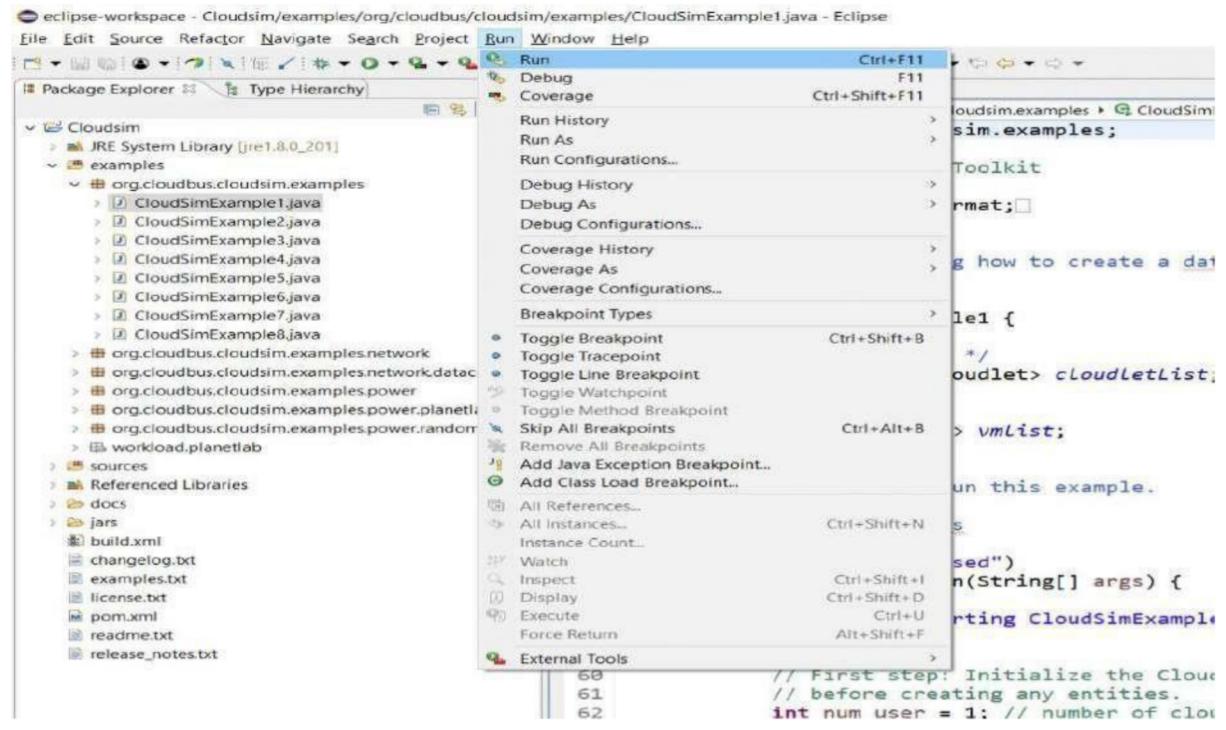
    /**
     * Creates main() to run this example
     */
    public static void main(String[] args) {
        Log.println("Starting CloudSimExample2...");

        try {
            // First step: Initialize the CloudSim package. It should be called
            // before creating any entities.
            int num_user = 2; // number of cloud users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false; // mean trace events

            // Initialize the CloudSim library
            CloudSim.init(num_user, calendar, trace_flag);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

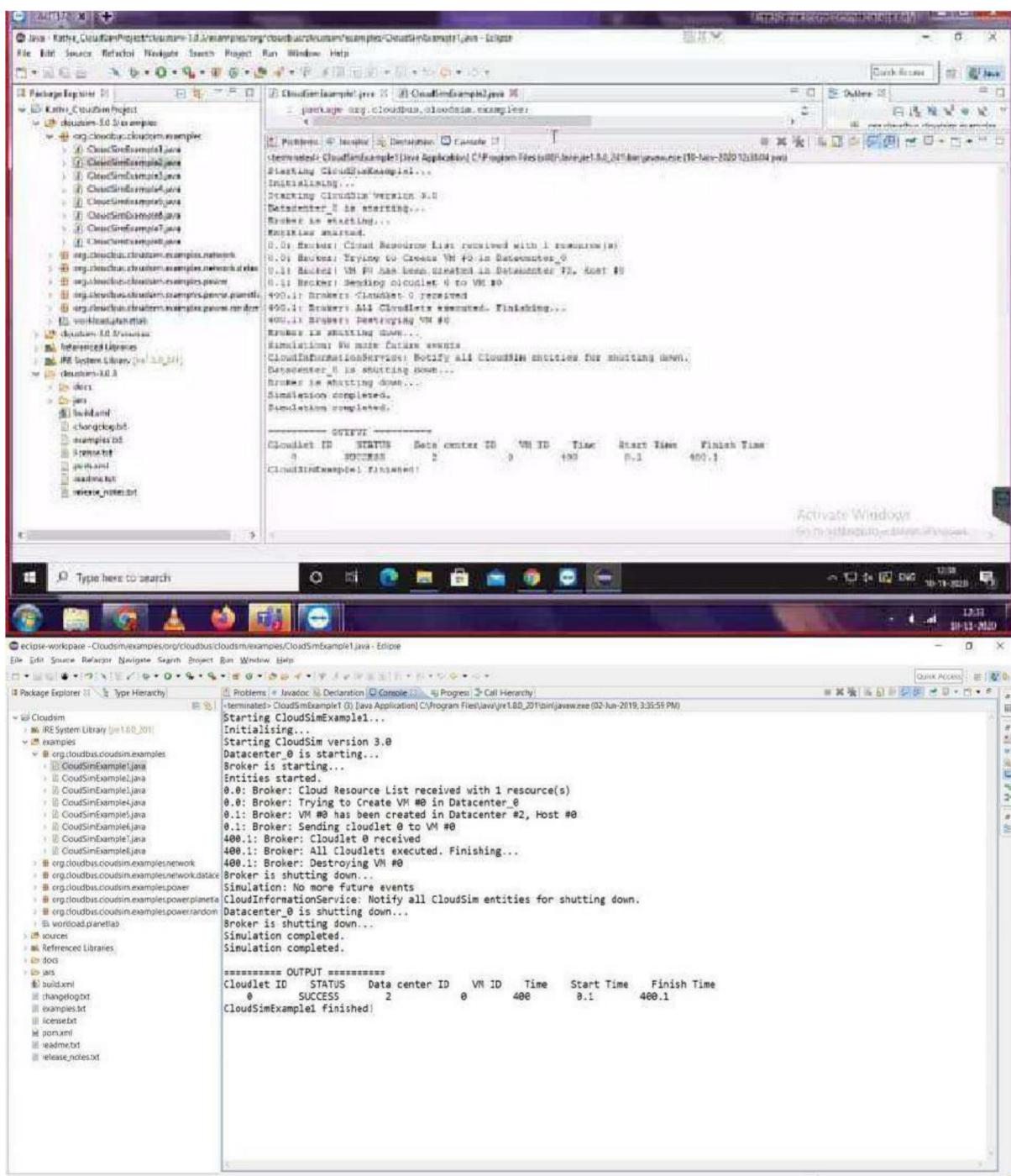
<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>13/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**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 in the console window of the Eclipse IDE.

<b>CM/ADL-D-05</b>	Simulate a cloud scenario using CloudSim	<b>Page</b>	<b>14/14</b>
<b>Experiment No.: 03</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Result:**

Thus the cloudsim is simulated using Eclipse Environment successfully.

<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>01/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

## Aim:

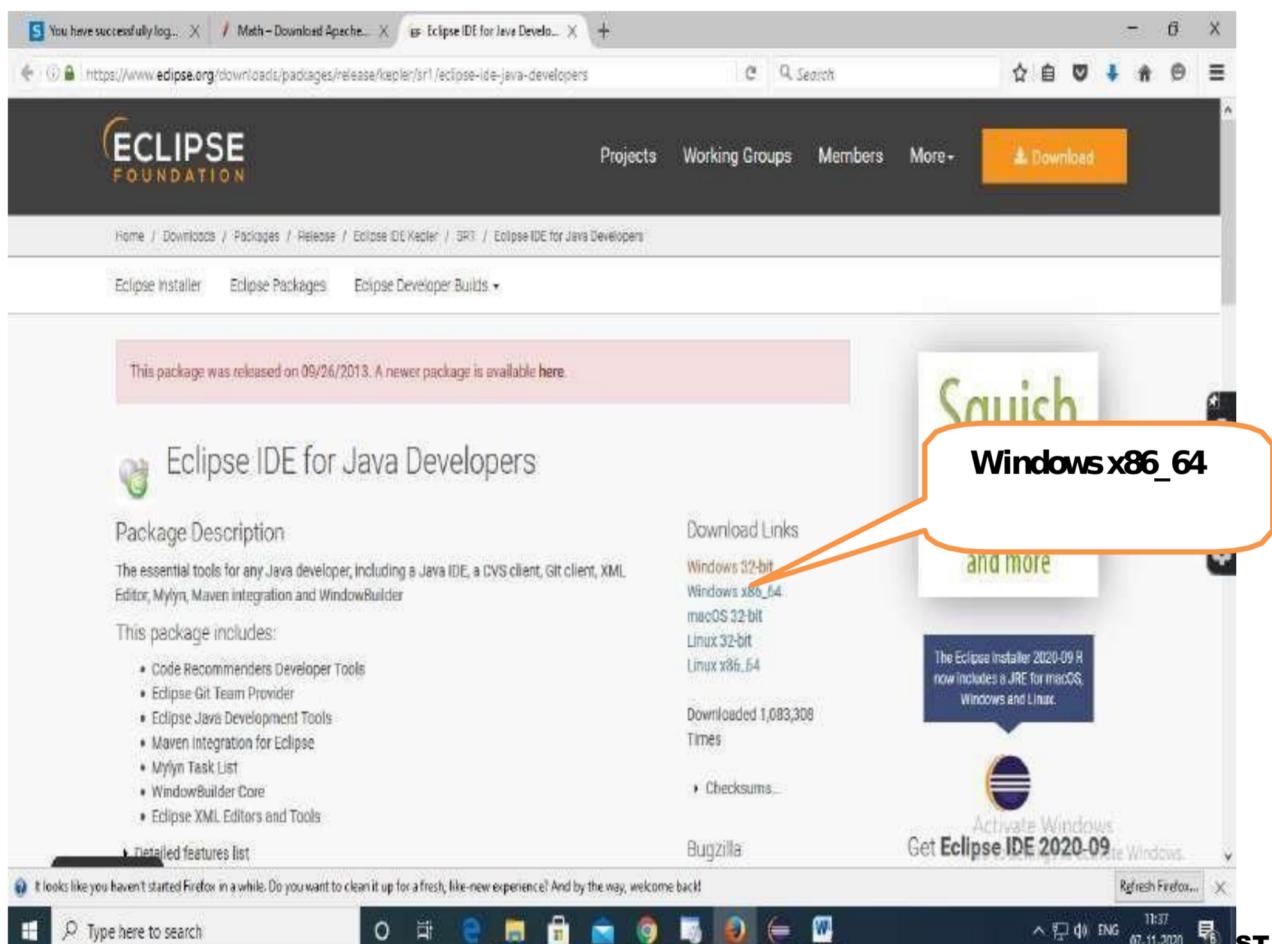
Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

## Theory:

### Procedure to import Eclipse, running scheduling algorithms in your system

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

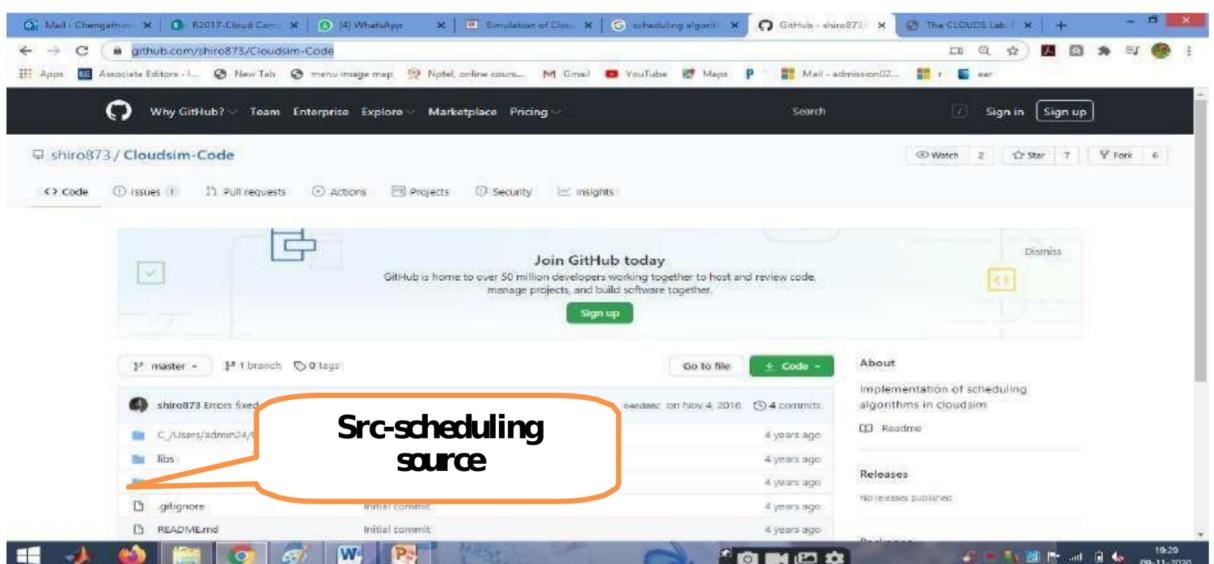
<https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers>



<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>02/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

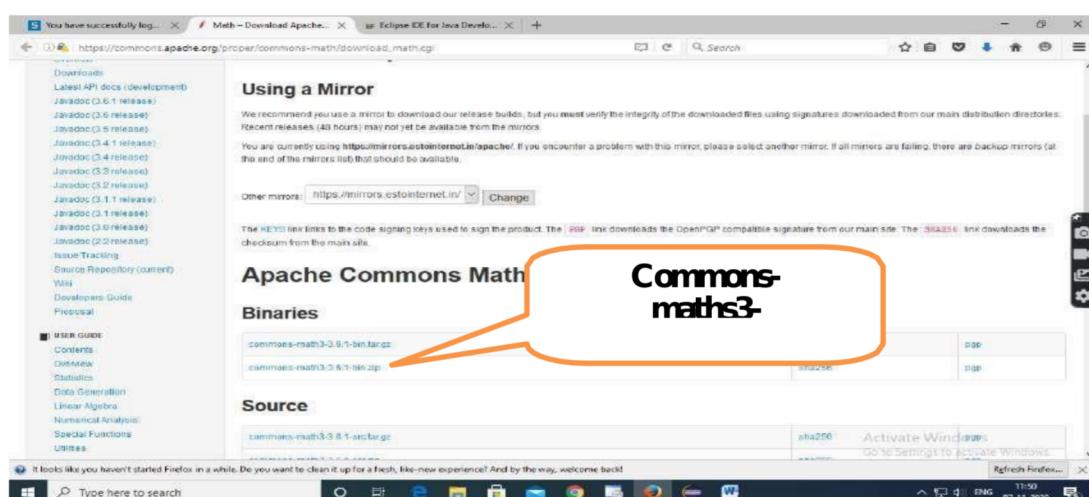
**Step 2:** Download scheduling source code **cloudsim-code-master** from git hub repository in your local machine

<https://github.com/shiro873/Cloudsim-Code>



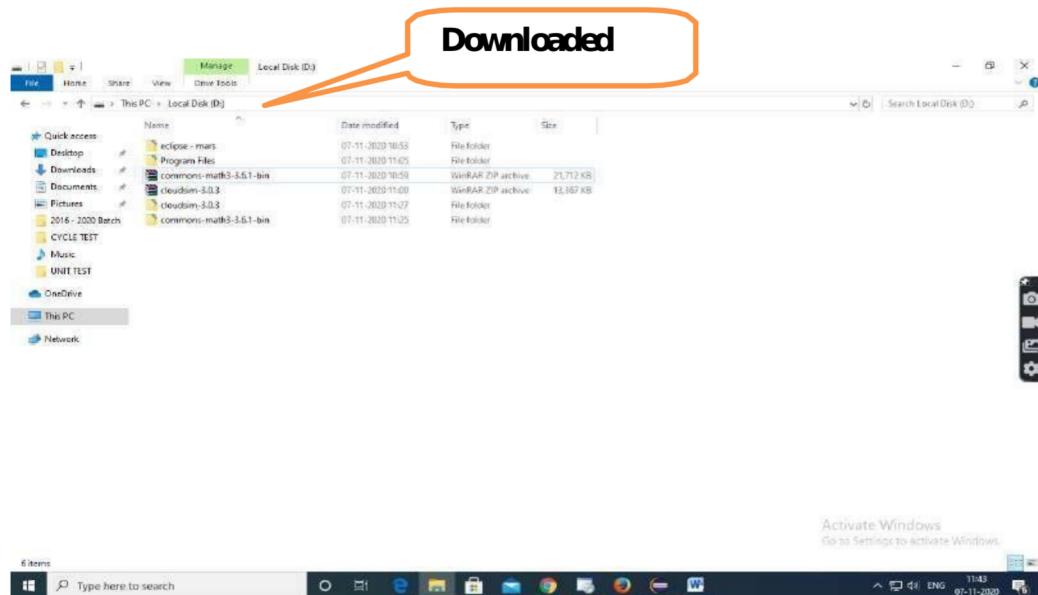
**Step 3:** Download commons-math3-3.6.1 from git hub repository in your local machine

[https://commons.apache.org/proper/commons-math/download\\_math.cgi](https://commons.apache.org/proper/commons-math/download_math.cgi)

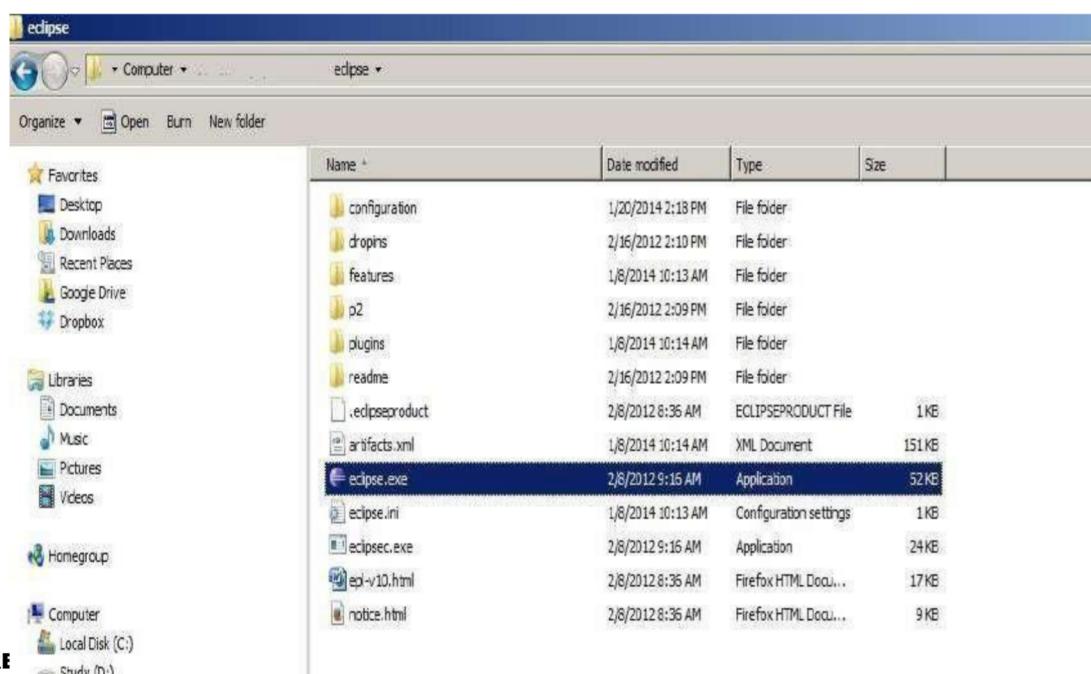


<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>03/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date 15-06-17</b>

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

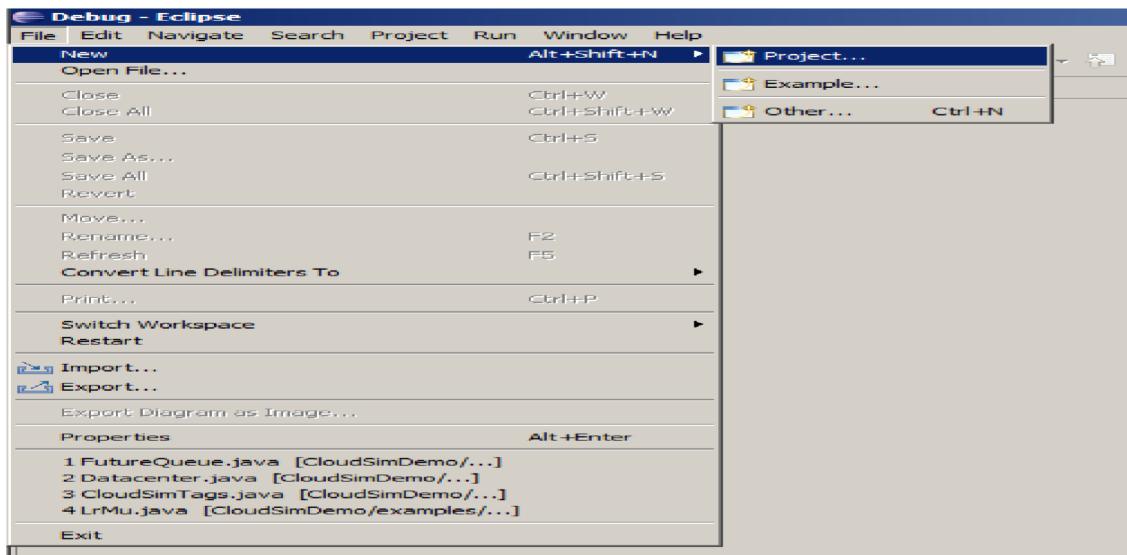


**Step 5:** First of all, navigate to the folder where you have unzipped the eclipse folder and open Eclipse.exe

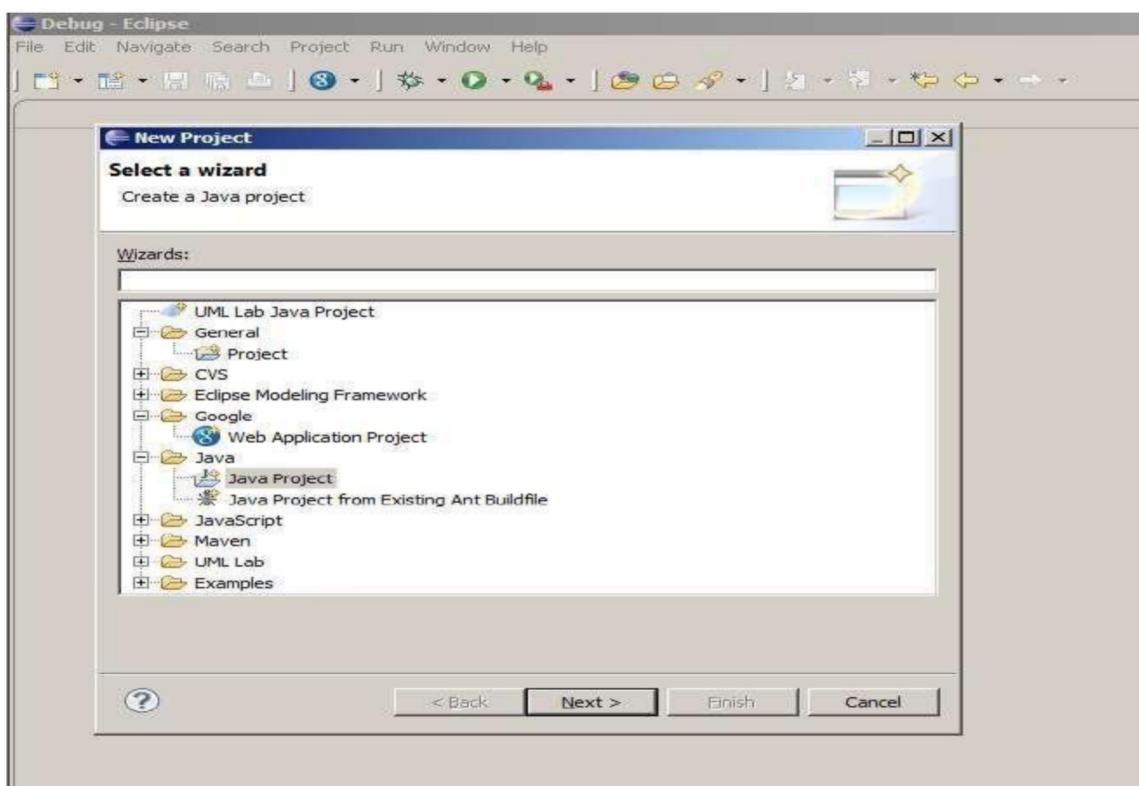


<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>04/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date 15-06-17</b>

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



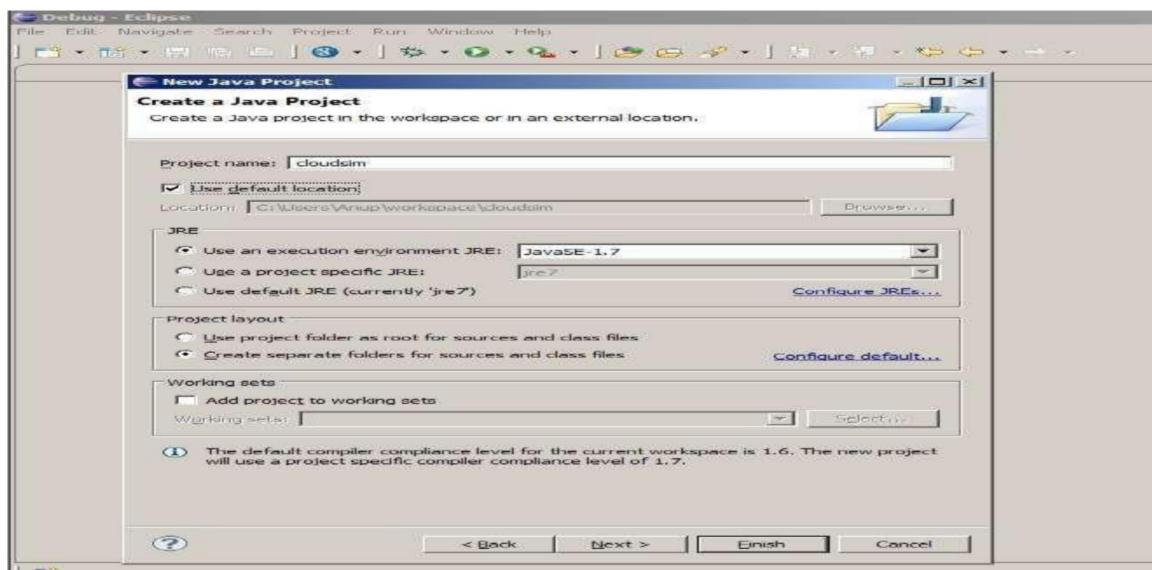
**Step 7:** A New Project wizard should open. There are a number of options displayed and you have to find & select the Java Project option, once done click 'Next\_



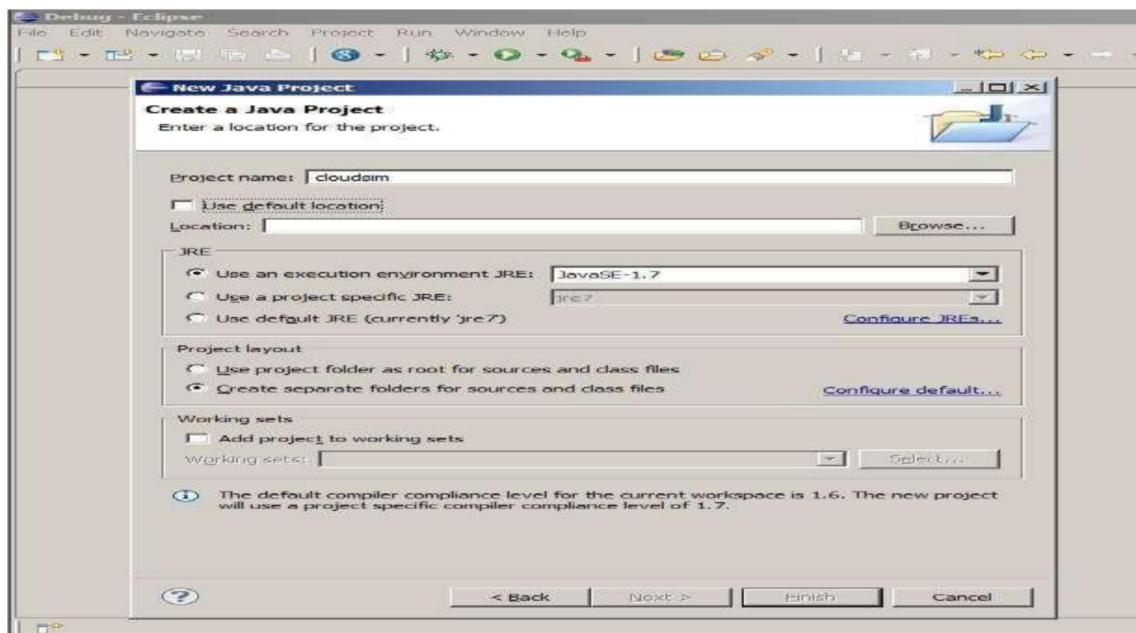
<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>05/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

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

**Project Name: CloudSim**

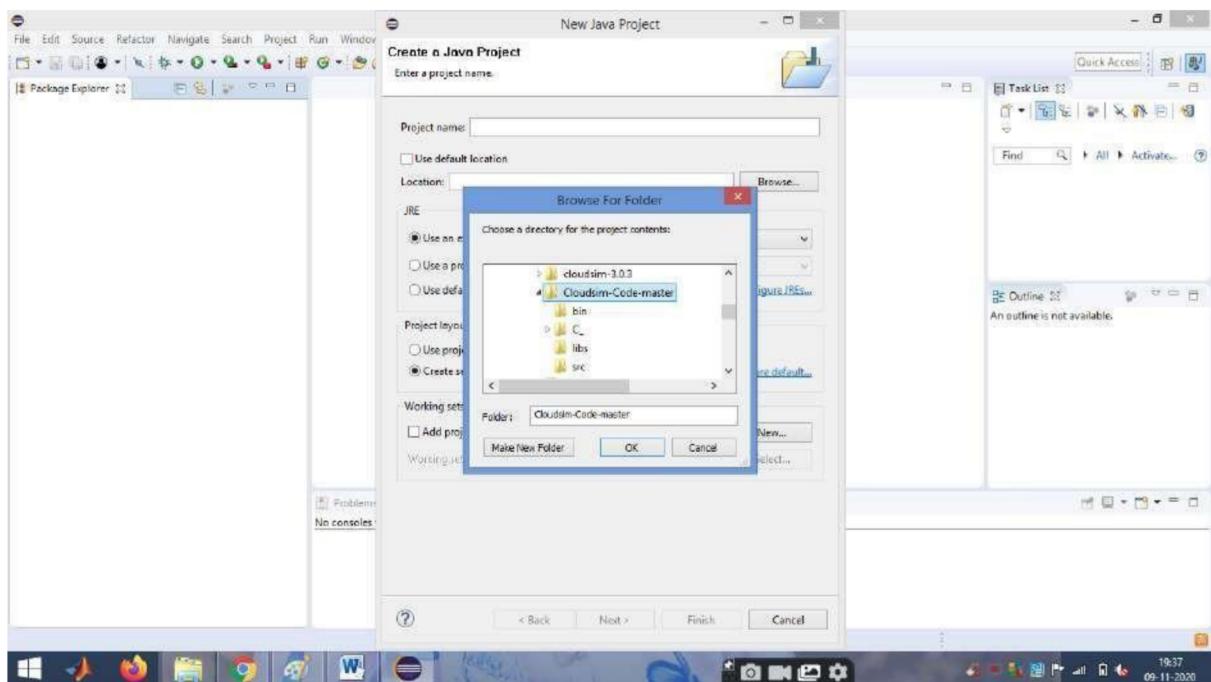


**Step 9:** Unselect the 'Use default location' option and then click on 'Browse' to open the path where you have unzipped the Cloudsim-code-master project and finally click Next to set project settings.

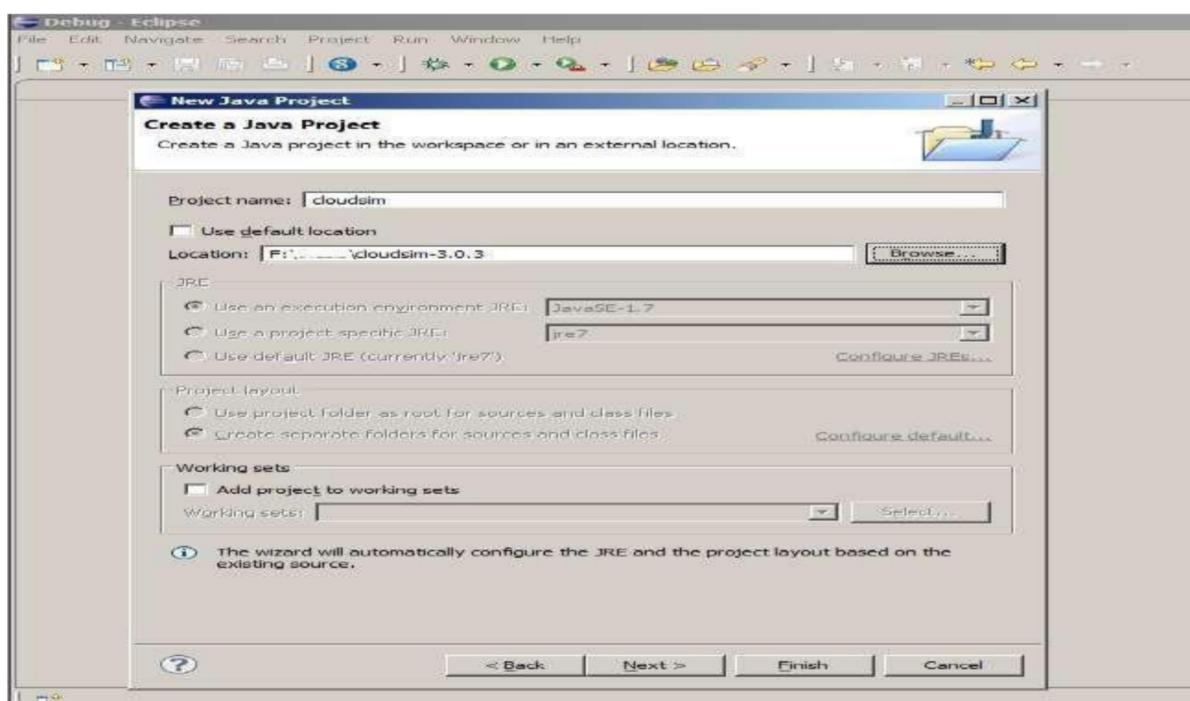


<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>06/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

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



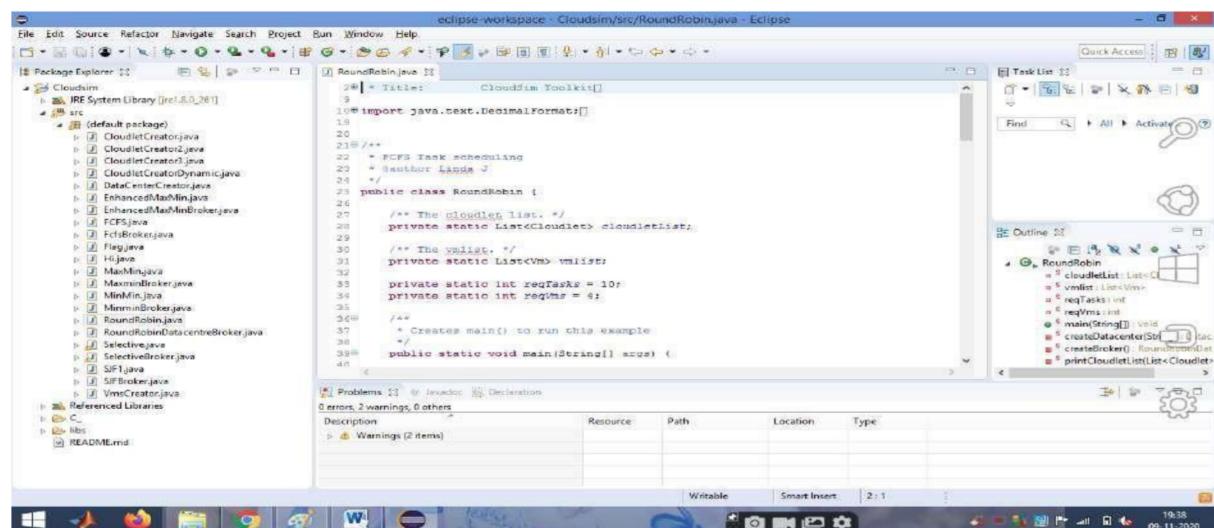
**Step 11:** Once done finally, click ‘Next’ to go to the next step i.e. setting up of project settings



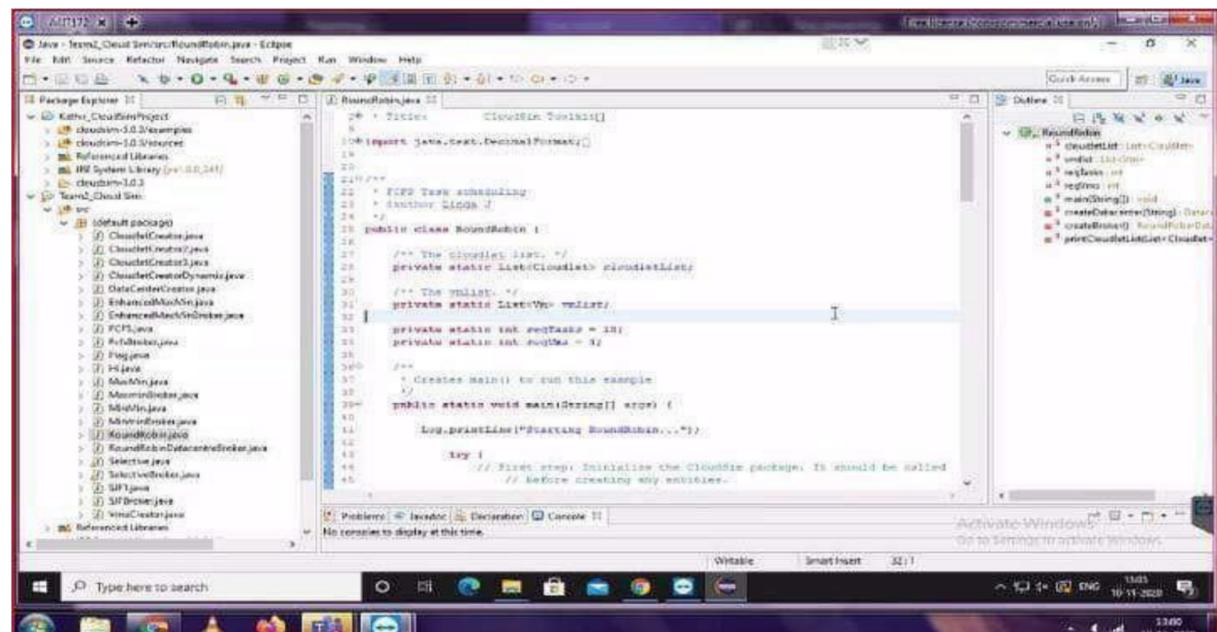
<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>07/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 12:** 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.

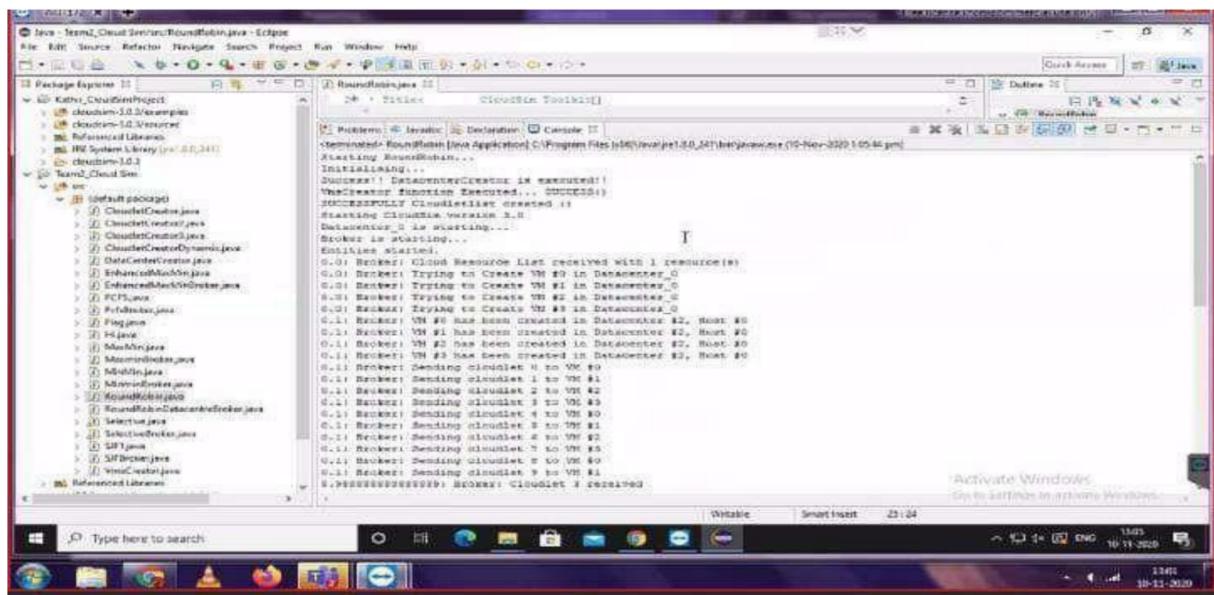


**Step 13:** Now just to check you within the Project Explorer, you should navigate to the src folder, then expand the package default package and double click to open the RoundRobin.java.



<b>CM/ADL-D-010</b>	Simulate a cloud scenario using CloudSim and running a scheduling algorithm	<b>Page</b>	<b>08/08</b>
<b>Experiment No.: 04</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**Step 14:** Now navigate to the Eclipse menu *Run ->Run* or directly use a keyboard shortcut '*Ctrl + F11*' to execute the '*RoundRobin.java*'. If it is successfully executed it should be displaying the following type of output in the console window of the Eclipse IDE.



## **Result:**

Thus the scheduling algorithm is executed in cloudsim is simulated using Eclipse Environment successfully.

<b>CM/ADL-D-09</b>	<b>Procedure File Transfer in Client &amp; Server using virtual machine.</b>	<b>Page</b>	<b>01/03</b>
<b>Experiment No.: 05</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**AimTitle:**

To procedure File Transfer in Client & Server using virtual machine

**Theory:****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)
```

<b>CM/ADL-D-09</b>	<b>Procedure File Transfer in Client &amp; Server using virtual machine.</b>	<b>Page</b>	<b>02/03</b>
<b>Experiment No.: 05</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

```
        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();
        }
    }
}
```

<b>CM/ADL-D-09</b>	<b>Procedure File Transfer in Client &amp; Server using virtual machine.</b>	<b>Page</b>	<b>03/03</b>
<b>Experiment No.: 02-C</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

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

## Out.txt

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

## Output:

The image shows two Microsoft Windows command-line windows side-by-side. Both windows are titled 'C:\Windows\system32\cmd.exe' and show the Windows logo watermark.

**Left Window (Server Side):**

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\          .d:
D:\>cd      \Programs\FTP
D:\          Programs\FTP>javac FTPServer.java
D:\          Programs\FTP>java FTPServer
Waiting.....
File is sent to client
D:\          \Programs\FTP>
```

**Right Window (Client Side):**

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\          .d:
D:\>cd      \Programs\FTP
D:\          \Programs\FTP>javac FTPClient.java
D:\          \Programs\FTP>java FTPClient
Enter the file name:
out.txt
Even Numbers in theout.txt are

2
4
6
8
```

## Result:

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

CM/ADL-D-13	Find a procedure to launch virtual machine using Openstack	Page	01/09
Experiment No.: 06	Semester – II	Rev.: 00	Date: 15-06-17

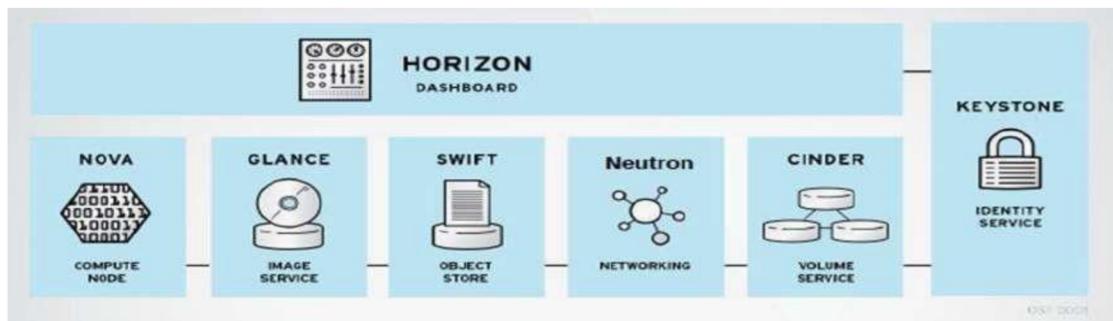
**Aim:**

Find a procedure to launch virtual machine using Openstack

**Introduction:**

- ❖ OpenStack was introduced by Rackspace and NASA in July 2010.
- ❖ OpenStack is an Infrastructure as a Service known as Cloud Operating System, that take resources such as Compute, Storage, Network and Virtualization Technologies and control those resources at a data center level
- ❖ The project is building an open source community - to share resources and technologies with the goal of creating a massively scalable and secure cloud infrastructure.
- ❖ The software is open source and limited to just open source APIs such as Amazon.

The following figure shows the OpenStack architecture



OpenStack architecture

- It is modular architecture
- Designed to easily scale out
- Based on (growing) set of core services

**The major components are**

- 1. Keystone**
- 2. Nova**
- 3. Glance**
- 4. Swift**
- 5. Quantum**
- 6. Cinder**

<b>CM/ADL-D-13</b>	<b>Find a procedure to launch virtual machine using Openstack</b>	<b>Page</b>	<b>02/09</b>
<b>Experiment No.: 06</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

- **KEYSTONE :**

- Identity service
- Common authorization framework
- Manage users, tenants and roles
- Pluggable backends (SQL,PAM,LDAP, IDM etc)

- **NOVA**

- Core compute service comprised of
  - Compute Nodes – hypervisors that run virtual machines
    - Supports multiple hypervisors KVM,Xen,LXC,Hyper-V and ESX
  - Distributed controllers that handle scheduling, API calls, etc
    - Native OpenStack API and Amazon EC2 compatible API

- **GLANCE**

- Image service
- Stores and retrieves disk images (Virtual machine templates)
- Supports RAW,QCOW,VHD,ISO,OVF & AMI/AKI
- Backend Storage: File System, Swift, Gluster, Amazon S3

- **SWIFT**

- Object Storage service
- Modeled after Amazon's Service
- Provides simple service for storing and retrieving arbitrary data
- Native API and S3 compatible API

- **NEUTRON**

- Network service
- Provides framework for Software Defined Network
- Plugin architecture
  - Allows integration of hardware and software based network solutions

Open vSwitch, Cisco UCS, Standard Linux Bridge, NiCira NVP

- **CINDER**

- Block Storage (Volume) service
- Provides block storage for Virtual machines(persistent disks)
- Similar to Amazon EBS service

<b>CM/ADL-D-13</b>	<b>Find a procedure to launch virtual machine using Openstack</b>	<b>Page</b>	<b>03/09</b>
<b>Experiment No.: 06</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

- Plugin architecture for vendor extensions
  - NetApp driver for cinder

- **HORIZON**

- Dashboard
- Provides simple self service UI for end-users
- Basic cloud administrator functions
  - Define users, tenants and quotas
  - No infrastructure management

- **HEAT OpenStack Orchestration**

- Provides template driven cloud application orchestration
- Modeled after AWS Cloud Formation
- Targeted to provide advanced functionality such as high availability and auto scaling
- Introduced by Redhat

- **CEILOMETER** – OpenStack Monitoring and Metering

- Goal: To Provide a single infrastructure to collect measurements from an entire OpenStack Infrastructure; Eliminate need for multiple agents attaching to multiple OpenStack Projects
- Primary targets metering and monitoring; Provided extensibility

❖ **Steps in Installing**

**Openstack Step 1:**

- Download and Install Oracle Virtual Box latest version & Extension package
  - <https://virtualbox.org/wiki/downloads>

**Step 2:**

- Download CentOS 7 OVA(Open Virtual Appliance) from
  - Link : <https://linuxvmimages.com/images/centos-7>
- Import CentOS 7 OVA(Open Virtual Appliance) into Oracle Virtual Box

CM/ADL-D-13	Find a procedure to launch virtual machine using Openstack	Page	04/09
Experiment No.: 06	Semester – II	Rev.: 00	Date: 15-06-17

1. Create a Virtual Machine on your VM Ware or Oracle Virtual Box.



### **Step 3: Login into CentOS 7**

- Login Details
  - **User name : centos**
  - **Password : centos**
- To change into root user in Terminal

**#sudosu–**

```
File Edit View Search Terminal Help
[jedureka@localhost ~]$ su
Password:
[root@localhost edureka]#
```

CM/ADL-D-13	Find a procedure to launch virtual machine using Openstack	Page	05/09
Experiment No.: 06	Semester – II	Rev.: 00	Date: 15-06-17

**Step 4:** Installation Steps for OpenStack

**Step5:** Command to disable and stop firewall

```
# systemctl disable  
firewalld #systemctl stop  
firewalld
```

```
[root@localhost ~]# systemctl disable firewalld  
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.  
Removed symlink /etc/systemd/system/basic.target.wants/firewalld.service.  
[root@localhost ~]# systemctl stop firewalld  
[root@localhost ~]# █
```

I

**Step 6:** Command to disable and stop Network Manager

```
# systemctl disable  
NetworkManager # systemctl  
stop NetworkManager
```

```
[root@localhost ~]# systemctl disable NetworkManager  
Removed symlink /etc/systemd/system/multi-user.target.wants/NetworkManager.service.  
Removed symlink /etc/systemd/system/dbus-org.freedesktop.NetworkManager.service.  
Removed symlink /etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service.  
[root@localhost ~]# systemctl stop NetworkManager  
[root@localhost ~]# █
```

I

CM/ADL-D-13	Find a procedure to launch virtual machine using Openstack	Page	06/09
Experiment No.: 06	Semester – II	Rev.: 00	Date: 15-06-17

**Step 7:** Enable and start Network

```
#systemctl enable  
network #systemctl  
start network
```

```
[root@localhost ~]# systemctl enable network  
network.service is not a native service, redirecting to /sbin/chkconfig.  
Executing /sbin/chkconfig network on  
[root@localhost ~]# systemctl start network  
[root@localhost ~]#
```

**Step 8:** OpenStack will be deployed on your Node with the help of **PackStack** package provided by **rdo** repository (**RPM Distribution of OpenStack**). In order to enable **rdo** repositories on Centos 7 run the below command.

```
#yum install -y https://rdoproject.org/repos/rdo-release.rpm
```

```
[root@localhost ~]# yum install -y centos-release-openstack-newton
```

CM/ADL-D-13	Find a procedure to launch virtual machine using Openstack	Page	07/09
Experiment No.: 06	Semester - II	Rev.: 00	Date: 15-06-17

**Step 9:** Update Current packages**#yum update -y**

```
[root@localhost ~]# yum update -y
Loaded plugins: fastestmirror, langpacks
centos-ceph-jewel
centos-openstack-newton
centos-qemu-ev
(1/3): centos-ceph-jewel/7/x86_64/primary_db      2.9 kB  00:00:00
(2/3): centos-qemu-ev/7/x86_64/primary_db        2.9 kB  00:00:00
(3/3): centos-openstack-newton/x86_64/primary_db   2.9 kB  00:00:00
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net                  63 kB  00:00:01
 * extras: centos.excellmedia.net                52 kB  00:00:00
 * updates: mirrors.viethosting.com            853 kB  00:00:02
```

**Step 10:** Install OpenStack Release for CentOS**#yum install -y openstack-packstack**

```
[root@localhost ~]# yum install -y openstack-packstack
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: mirrors.viethosting.com
```

**Step 11:** Start packstack to install OpenStack Newton**#packstak --allinone**

```
[root@localhost ~]# packstack --allinone
Welcome to the Packstack setup utility

The installation log file is available at: /var/tmp/packstack/20170314-065810-b8cxch/openstack-setup.log
Packstack changed given value to required value /root/.ssh/id_rsa.pub

Installing:
Clean Up                                [ DONE ]
Discovering ip protocol version          [ DONE ]
Setting up ssh keys                      [ DONE ]
Preparing servers                        [ DONE ]
Pre installing Puppet and discovering hosts' details [ DONE ]
Preparing pre-install entries           [ DONE ]
Setting up CACERT                         [ DONE ]
Preparing AMQP entries                   [ DONE ]
Preparing MariaDB entries                [ DONE ]
Fixing Keystone LDAP config parameters to be undef if empty[ DONE ]
Preparing Keystone entries               [ DONE ]
Preparing Glance entries                 [ DONE ]
Checking if the Cinder server has a cinder-volumes vg[ DONE ]
Preparing Cinder entries                 [ DONE ]
Preparing Nova API entries              [ DONE ]
```

CM/ADL-D-13	Find a procedure to launch virtual machine using Openstack	Page	08/09
Experiment No.: 06	Semester - II	Rev.: 00	Date: 15-06-17

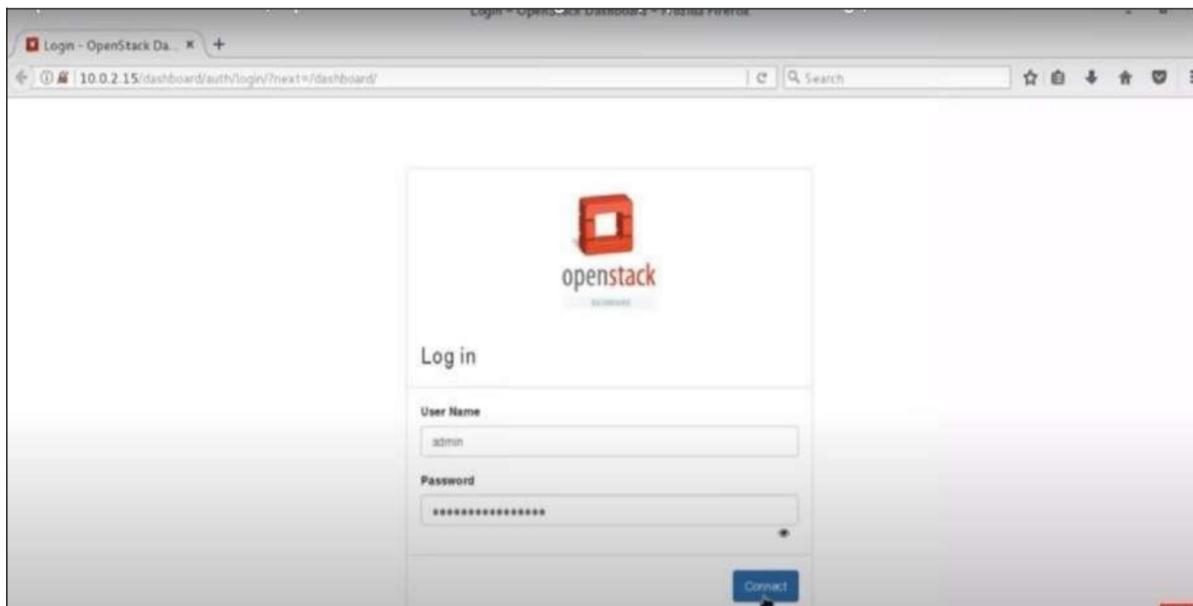
**Step 12:** Note the user name and password from kestonerc\_admin

```
[root@localhost ~]# ls
anaconda-ks.cfg      kestonerc_admin  packstack-answers-20170314-065812.txt
initial-setup-ks.cfg  kestonerc_demo
[root@localhost ~]# cat kestonerc_admin
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin
export OS_PASSWORD=cdc897f8cb7f4dda
export OS_AUTH_URL=http://10.0.2.15:5000/v2.0
export PS1='[\u@\h \W(keystone_admin)]\$ '

export OS_TENANT_NAME=admin
export OS_REGION_NAME=RegionOne
[root@localhost ~]#
```

**#cat kestonerc\_admin**

**Step 13:** Click the URL and enter the user name and password to start OpenStack



<b>CM/ADL-D-13</b>	<b>Find a procedure to launch virtual machine using Openstack</b>	<b>Page</b>	<b>09/09</b>
<b>Experiment No.: 06</b>	<b>Semester – II</b>	<b>Rev.: 00</b>	<b>Date: 15-06-17</b>

**OpenStack is successfully launched in your machine**

Name	Description	Project ID	Domain Name	Enabled	Actions
services	Tenant for the openstack services	2e6451398c8240bb81294e079b74e483	Default	Yes	Manage Members
admin	admin tenant	58095a144065471d88982e869d82bc94	Default	Yes	Manage Members
demo	default tenant	a5d60a950e484c98977d5d59cf71562d	Default	Yes	Manage Members

Name	Description	Project ID	Domain Name	Enabled	Actions
services	Tenant for the openstack services	2e6451398c8240bb81294e079b74e483	Default	Yes	Manage Members
admin	admin tenant	58095a144065471d88982e869d82bc94	Default	Yes	Manage Members
demo	default tenant	a5d60a950e484c98977d5d59cf71562d	Default	Yes	Manage Members

## **Result:**

Thus the OpenStack Installation is executed successfully.

