Assignment 1

Title

Install Google App Engine. Create hello world app and other simple web applications using Python/Java.

Requirements

1. Google App Engine
2. Python Interpreter (Python2.7.x)
3. Text Editor
4. Browser

Theory

A) Google App Engine

1. Google App Engine (GAE) is a platform-as-a-service product that provides web app developers and enterprises with access to Google's scalable hosting and tier 1 internet service.
2. GAE requires that applications be written in Java or Python, store data in Google Bigtable and use the Google query language.
3. Noncompliant applications require modification to use GAE.
4. GAE provides more infrastructure than other scalable hosting services, such as Amazon Elastic Compute Cloud (EC2).
5. GAE also eliminates some system administration and development tasks to make writing scalable applications easier.
6. Google provides GAE free up to a certain amount of use for resources like CPU, storage, API calls and concurrent requests

B) Google Cloud SDK

1. Google Cloud SDK (Software Development Kit), in simple terms, is a set of tools that are used to manage applications and resources that are hosted on the Google Cloud Platform.
2. It is composed of the gsutil, gcloud, and bqcommand line tools.
3. The gcloudtool is automatically downloaded with the Cloud SDK.

4. Google Cloud SDK run on specific platforms – Windows, Linux, and macOS and requires Python 2.7.x.
5. SDK might have further necessities like Java tools used for the development of Google App Engine needs Java 1.7 or the later one.
6. It can be used to locally deploy and test web applications.

C) Directory Structure for creating hello world application

1. The web applications to be deployed can be organized in the following directory structure

```
root_directory
          |________templates
          |             |_______index.html
          |________static
          |________main.py
          |________app.yaml
```

2. The templates directory can be used to store the web templates of the web application (HTML files).
3. The static directory can be used to store the web static files which contain the styling and the business logic data for the web application (CSS and JS files).
4. The main.py is used to define the routes, rendering logic, data acquisition logic.
5. It provides the WSGI abstraction to the application.
6. The app.yaml file provides the runtime environment, URLs for routes and launch configuration of the application in the form of key value pairs.

Steps

A) Install Google Cloud SDK on Windows or Linux machines

1. Visit the https://cloud.google.com/sdk/docs/install link to download the CLI (Command line interface) tool for the Cloud SDK.
2. Select the appropriate operating system from the installation manual
3. Follow the provided instructions in the displayed section
   a) For Windows users, the executable downloader is provided for downloading.

b) For Ubuntu and Fedora users, terminal commands for installation are provided using apt and dnf repositories respectively.

B) Creating the application

1. The application must be initialized using the above-mentioned directory structure.
2. It is a recommended format for organization and readability of code.
3. The app.yaml file should contain the following content:

Contents of app.yaml

```
runtime : python2
api_version : 1
threadsafe : true

handlers
-   url : /
    script : main.app
```

4. The logic of the application, i.e. the Web server interaction code of the application must be placed in the main.py file.
5. A simple code displaying the hello world on a web page is as follows

Contents of main.py for Hello World application

```
import webapp2

class MainPage(webapp2.RequestHandler) :

        def get(self):
                self.response.write(“Hello World”)

app = webapp2.WSGIApplication(
        [(“/”, MainPage)],
        debug=True
)
```

6. Finally, after saving the above code, the application can be run on the localhost server using the following command. (The command must be run on the Google Cloud Shell or the terminal in case of Ubuntu).

Command:

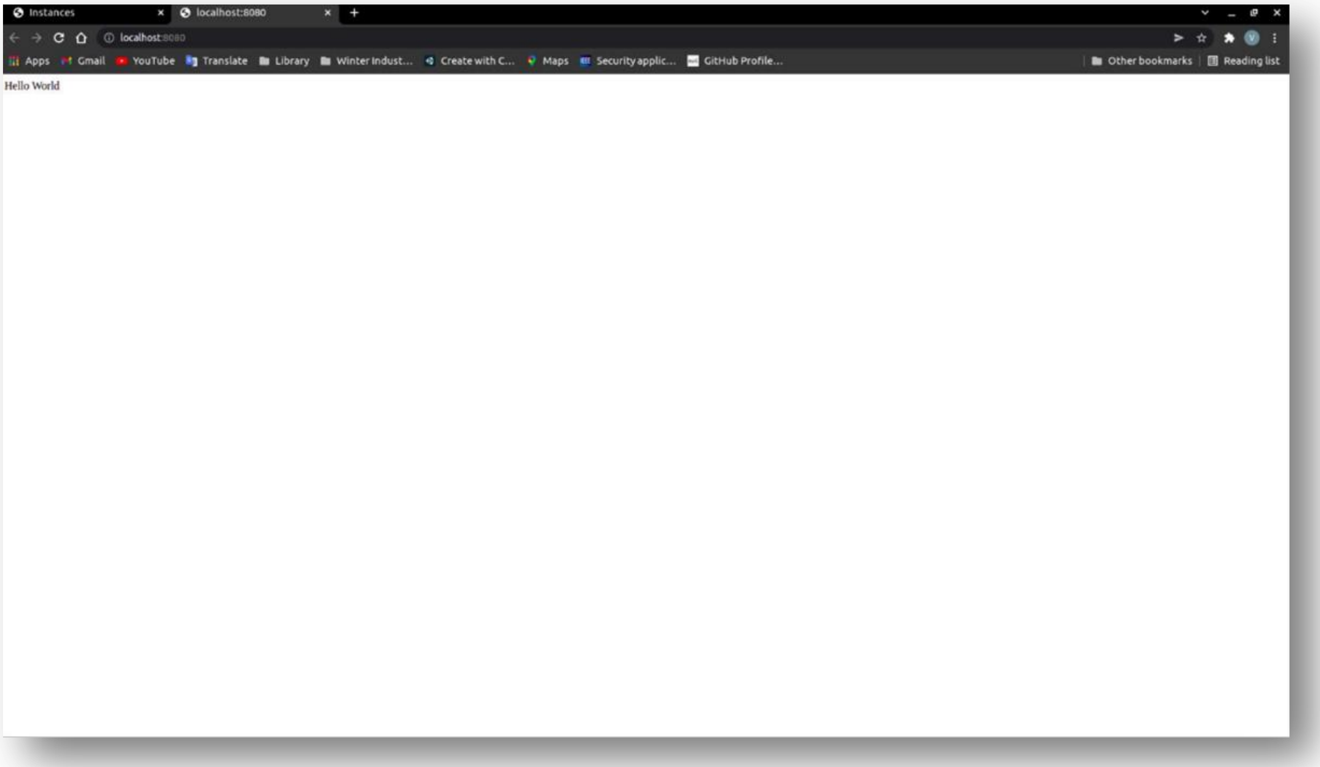
python <path_to_sdk>/bin/devappserver.py
<path_to_application_directory>

Sample Output

1. The application, if no errors are found, is launched on the port 8080 of the localhost server.
2. The cloud console is visible on port 8000 of the localhost server.
3. The URL of localhost:8080 can be typed in the address bar of the browser to view the application

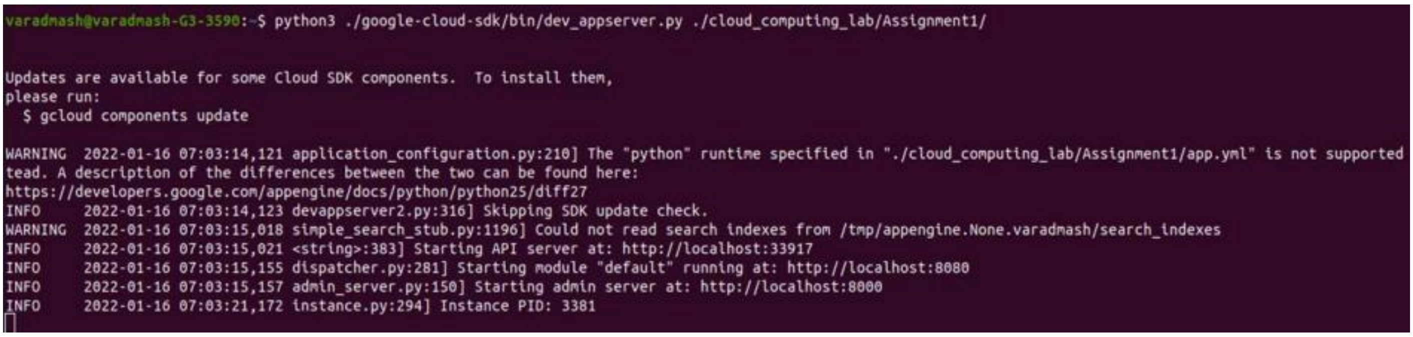
4. Screenshots:

a) Application launch at port 8080



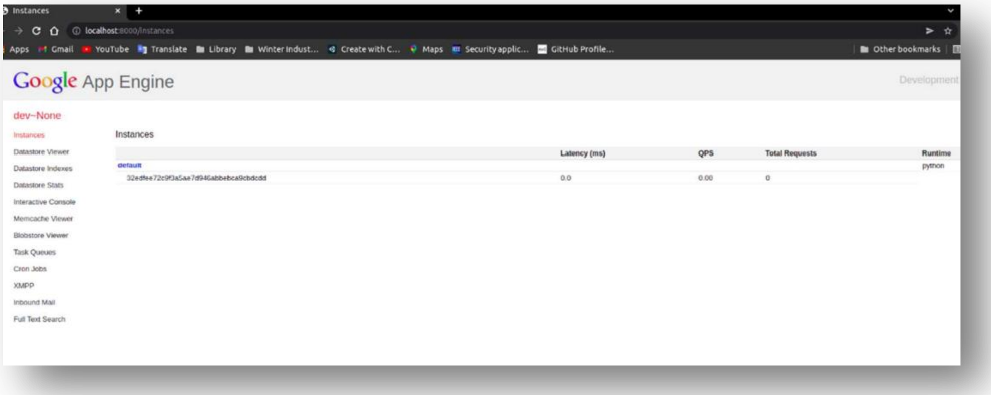
Hello World

b) Terminal / Command Line prompts



```
varadmash@varadmash-G3-3590:~$ python3 ./google-cloud-sdk/bin/dev_appserver.py ./cloud_computing_lab/Assignment1/


Updates are available for some Cloud SDK components.  To install them,
please run:
  $ gcloud components update

WARNING  2022-01-16 07:03:14,121 application_configuration.py:210] The "python" runtime specified in "./cloud_computing_lab/Assignment1/app.yml" is not supported
tead. A description of the differences between the two can be found here:
https://developers.google.com/appengine/docs/python/python25/diff27
INFO     2022-01-16 07:03:14,123 devappserver2.py:316] Skipping SDK update check.
WARNING  2022-01-16 07:03:15,018 simple_search_stub.py:1196] Could not read search indexes from /tmp/appengine.None.varadmash/search_indexes
INFO     2022-01-16 07:03:15,021 <string>:383] Starting API server at: http://localhost:33917
INFO     2022-01-16 07:03:15,155 dispatcher.py:281] Starting module "default" running at: http://localhost:8080
INFO     2022-01-16 07:03:15,157 admin_server.py:150] Starting admin server at: http://localhost:8000
INFO     2022-01-16 07:03:21,172 instance.py:294] Instance PID: 3381
```

c) Cloud Console at port 8000



Instances

	Latency (ms)	QPS	Total Requests	Runtime
default				python
32edfee72c9f3a5ae7d946abbebca9cbdcdd	0.0	0.00	0	

Assignment 2

Title

Use GAE launcher to launch the web applications.

Requirements

1. Google App Engine
2. Python Interpreter (Python2.7.x)
3. Browser

Theory

A) Google App Engine

7. Google App Engine (GAE) is a platform-as-a-service product that provides web app developers and enterprises with access to Google's scalable hosting and tier 1 internet service.
8. GAE requires that applications be written in Java or Python, store data in Google Bigtable and use the Google query language.
9. Noncompliant applications require modification to use GAE.
10. GAE provides more infrastructure than other scalable hosting services, such as Amazon Elastic Compute Cloud (EC2).
11. GAE also eliminates some system administration and development tasks to make writing scalable applications easier.
12. Google provides GAE free up to a certain amount of use for resources like CPU, storage, API calls and concurrent requests

B) Directory Structure for creating web application

7. The web applications to be deployed can be organized in the following directory structure

```
root_directory
          |_______templates
          |           |______index.html
          |           |______results.html
          |_______static
          |_______main.py
          |_______app.yaml
```

8. The templates directory can be used to store the web templates of the web application (HTML files).
9. The static directory can be used to store the web static files which contain the styling and the business logic data for the web application (CSS and JS files).
10. The main.py is used to define the routes, rendering logic, data acquisition logic.
11. It provides the WSGI abstraction to the application.
12. The app.yaml file provides the runtime environment, URLs for routes and launch configuration of the application in the form of key value pairs.

Steps

A) Creating the application

1. The application must be initialized using the above-mentioned directory structure.
2. It is a recommended format for organization and readability of code.
3. Create index.html and results.html as web templates with index.html for taking user input and results.html for displaying response from the API call.
4. The app.yaml file should contain the following content:

   Contents of app.yaml

   ```
   runtime : python2
   api_version : 1
   threadsafe : true

   handlers
   url : /
   script : main.app
   ```

5. The logic of the application, i.e. the Web server interaction code of the application must be placed in the main.py file.
6. The following python code sends request for information to the API and interprets response from the API (here, World Time API is used).

Contents of main.py for World Time application

```
import os
import json
import urllib
```

```
import webapp2
from google.appengine.ext.webapp import template

class MainPage(webapp2.RequestHandler):
    def get(self):
        template_values = {}
        path = os.path.join(os.path.dirname(__file__), 'templates/index.html')
        self.response.out.write(template.render(path, template_values))

    def post(self):
        region = self.request.get('region')
        area = self.request.get('area')
        url = "http://worldtimeapi.org/api/timezone/"+ region + "/" + area
        data = urllib.urlopen(url).read()
        data = json.loads(data)
        date = data['datetime'][0:10]
        time = data['datetime'][11:19]
        week = data['day_of_week']
        year = data['day_of_year']
        weeknum = data['week_number']
        template_values = {
            "date": date,
            "time": time,
            "week": week,
            "year": year,
            "weeknum": weeknum,
        }
        path = os.path.join(os.path.dirname(__file__),
'templates/results.html')
        self.response.out.write(template.render(path, template_values))


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

7. Finally, after saving the above code, the application can be run on the localhost server using the following command.

Command:

```
python <path_to_sdk>/bin/devappserver.py
<path_to_application_directory>
```

Sample Output

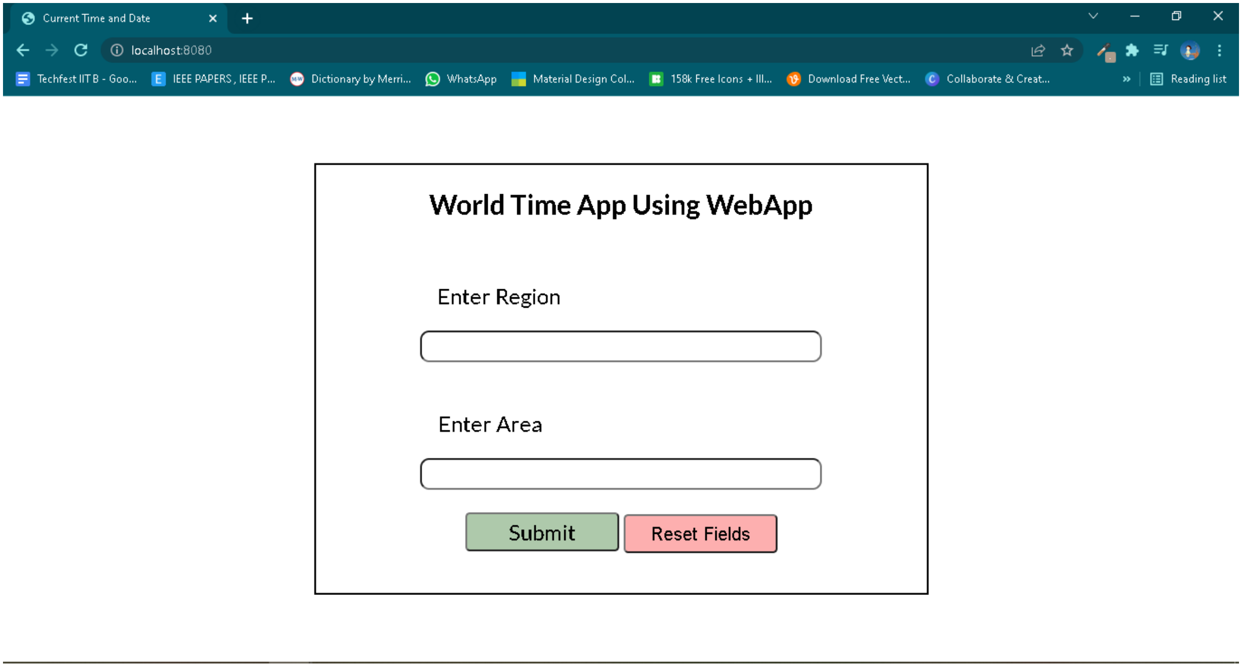
A) The application, if no errors are found, is launched on the port 8080 of the localhost server.

B) The cloud console is visible on port 8000 of the localhost server.

C) The URL of localhost:8080 can be typed in the address bar of the browser to view the application

D) Screenshots:

a) Application launch at port 8080 (Displaying index.html)

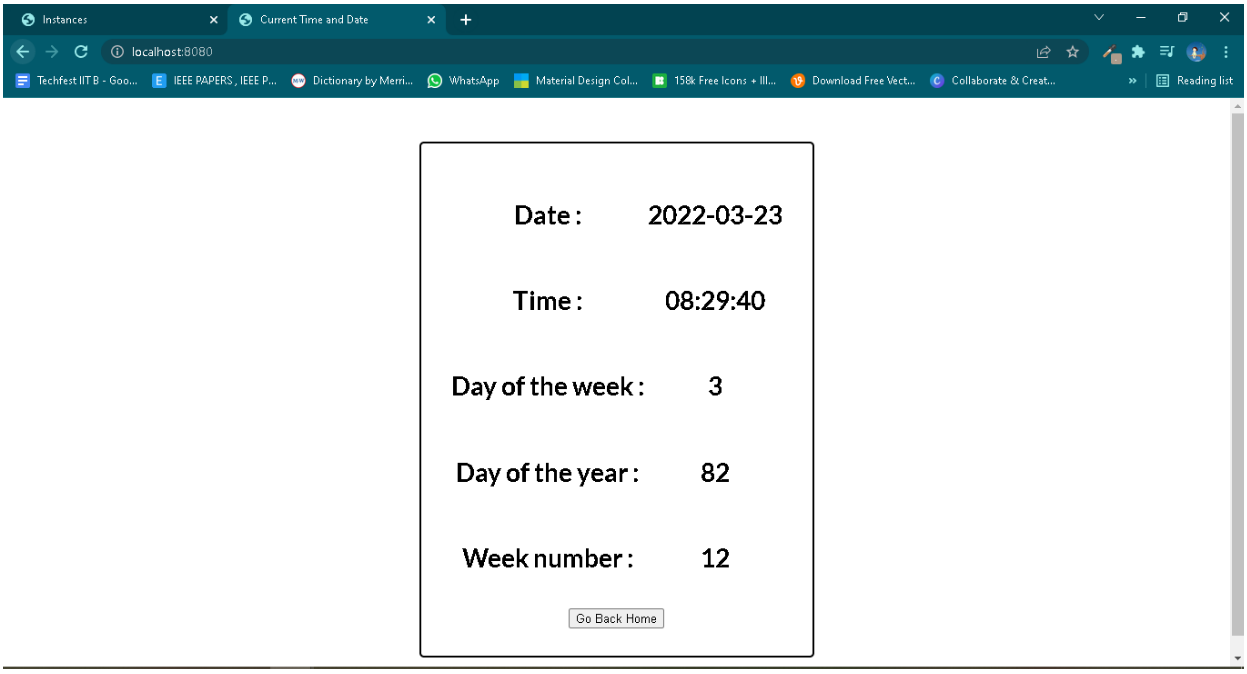


World Time App Using WebApp

Enter Region

Enter Area

b) Displaying results.html to display results from the API call



Date : 2022-03-23

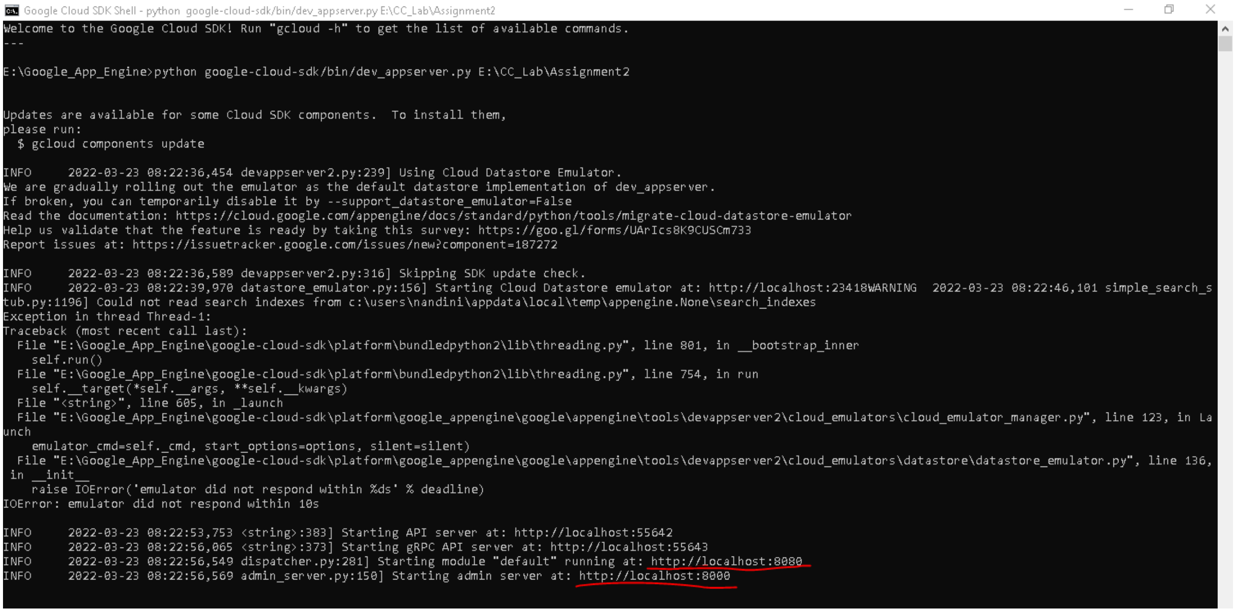
Time : 08:29:40

Day of the week : 3

Day of the year : 82

Week number : 12

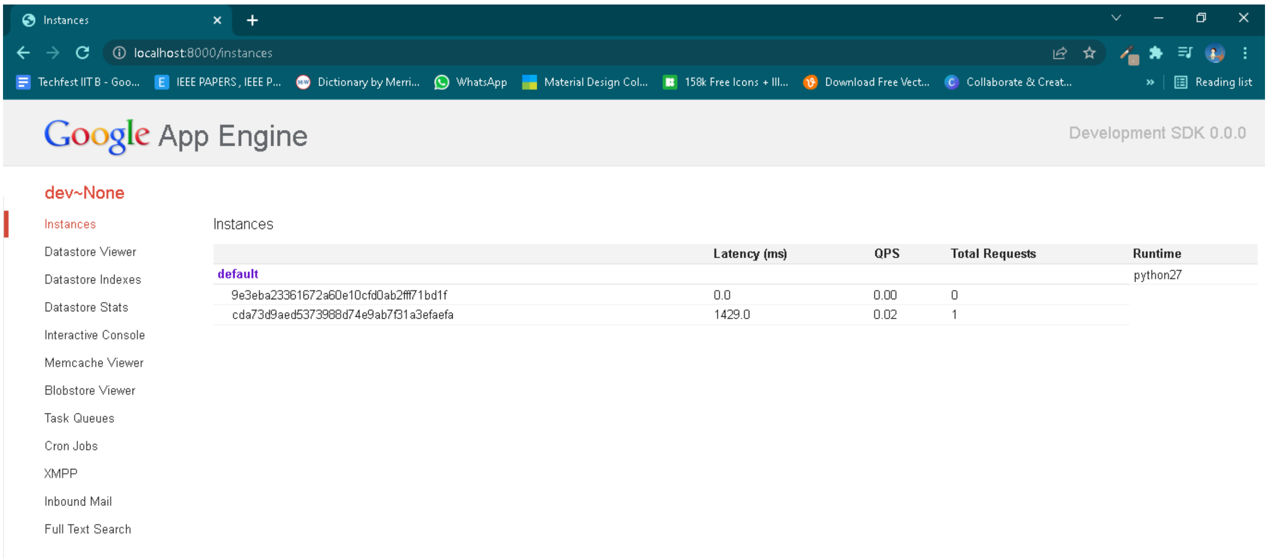
c) Terminal / Command Line prompts



Google Cloud SDK Shell - python google-cloud-sdk/bin/dev_appserver.py E:\CC_Lab\Assignment2

```
Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.
---

E:\Google_App_Engine>python google-cloud-sdk/bin/dev_appserver.py E:\CC_Lab\Assignment2


Updates are available for some Cloud SDK components.  To install them,
please run:
  $ gcloud components update

INFO     2022-03-23 08:22:36,454 devappserver2.py:239] Using Cloud Datastore Emulator.
We are gradually rolling out the emulator as the default datastore implementation of dev_appserver.
If broken, you can temporarily disable it by --support_datastore_emulator=False
Read the documentation: https://cloud.google.com/appengine/docs/standard/python/tools/migrate-cloud-datastore-emulator
Help us validate that the feature is ready by taking this survey: https://goo.gl/forms/UArIcs8K9CUSCm733
Report issues at: https://issuetracker.google.com/issues/new?component=187272

INFO     2022-03-23 08:22:36,589 devappserver2.py:316] Skipping SDK update check.
INFO     2022-03-23 08:22:39,970 datastore_emulator.py:156] Starting Cloud Datastore emulator at: http://localhost:23418WARNING  2022-03-23 08:22:46,101 simple_search_s
tub.py:1196] Could not read search indexes from c:\users\nandini\appdata\local\temp\appengine.None\search_indexes
Exception in thread Thread-1:
Traceback (most recent call last):
  File "E:\Google_App_Engine\google-cloud-sdk\platform\bundledpython2\lib\threading.py", line 801, in __bootstrap_inner
    self.run()
  File "E:\Google_App_Engine\google-cloud-sdk\platform\bundledpython2\lib\threading.py", line 754, in run
    self.__target(*self.__args, **self.__kwargs)
  File "<string>", line 605, in _launch
  File "E:\Google_App_Engine\google-cloud-sdk\platform\google_appengine\google\appengine\tools\devappserver2\cloud_emulators\cloud_emulator_manager.py", line 123, in La
unch
    emulator_cmd=self._cmd, start_options=options, silent=silent)
  File "E:\Google_App_Engine\google-cloud-sdk\platform\google_appengine\google\appengine\tools\devappserver2\cloud_emulators\datastore\datastore_emulator.py", line 136,
 in __init__
    raise IOError('emulator did not respond within %ds' % deadline)
IOError: emulator did not respond within 10s

INFO     2022-03-23 08:22:53,753 <string>:383] Starting API server at: http://localhost:55642
INFO     2022-03-23 08:22:56,065 <string>:373] Starting gRPC API server at: http://localhost:55643
INFO     2022-03-23 08:22:56,549 dispatcher.py:281] Starting module "default" running at: http://localhost:8080
INFO     2022-03-23 08:22:56,569 admin_server.py:150] Starting admin server at: http://localhost:8000
```

d) Cloud Console at port 8000



Instances

	Latency (ms)	QPS	Total Requests	Runtime
default				python27
9e3eba23361672a60e10cfd0ab2fff71bd1f	0.0	0.00	0	
cda73d9aed5373988d74e9ab7f31a3efaefa	1429.0	0.02	1	

Assignment 3

Title

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

Requirements

1. Java JDK and JRE
2. CloudSim archives (CloudSim4)
3. Eclipse IDE

Theory

A) CloudSim

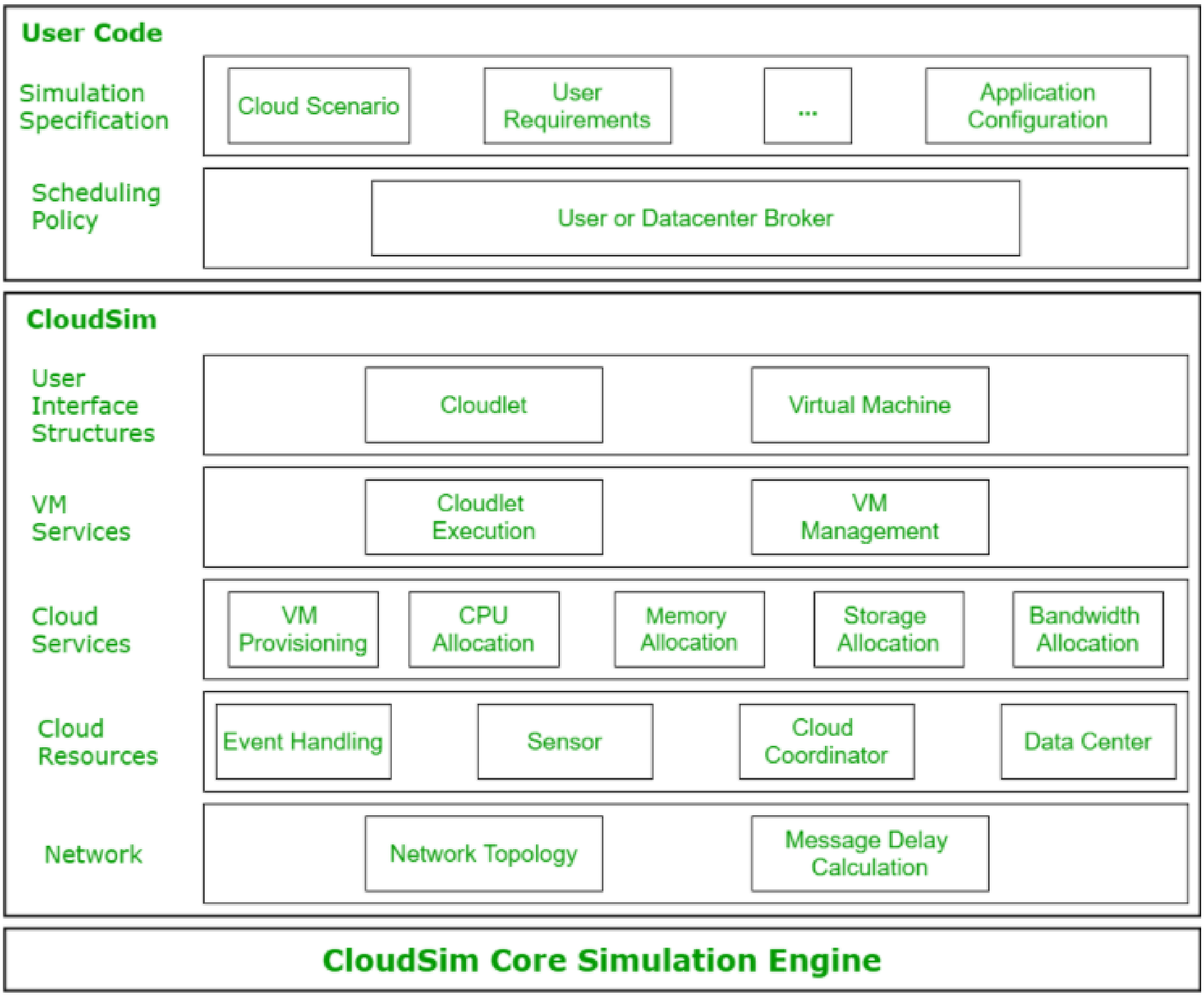
1. CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services.
2. It is developed by the CLOUDS Lab organization and is written entirely in Java.
3. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.
4. If you were to deploy an application or a website on the cloud and wanted to test the services and load that your product can handle and also tune its performance to overcome bottlenecks before risking deployment, then such evaluations could be performed by simply coding a simulation of that environment with the help of various flexible and scalable classes provided by the CloudSim package, free of cost.

B) Benefits of CloudSim

1. No capital investment involved
2. Easy to use and Scalable
3. Risks can be evaluated at an earlier stage
4. No need for try-and-error approaches

C) Architecture

1. CloudSim has a layered architecture which separates the User Code and the simulation environment.
2. It can be depicted as follows



CloudSim Layered Architecture

D) CloudSim Components

- **Datacenter**: used for modelling the foundational hardware equipment of any cloud environment, that is the Datacenter. This class provides methods to specify the functional requirements of the Datacenter as well as methods to set the allocation policies of the VMs etc.
- **Host**: this class executes actions related to management of virtual machines. It also defines policies for provisioning memory and bandwidth to the virtual machines, as well as allocating CPU cores to the virtual machines.
- **VM**: this class represents a virtual machine by providing data members defining a VM's bandwidth, RAM, mips (million instructions per second), size while also providing setter and getter methods for these parameters.
- **Cloudlet**: a cloudlet class represents any task that is run on a VM, like a processing task, or a memory access task, or a file updating task etc. It stores parameters defining the characteristics of a task such as its length, size, mi (million instructions) and provides methods similarly to

VM class while also providing methods that define a task's execution time, status, cost and history.

- **DatacenterBroker**: is an entity acting on behalf of the user/customer. It is responsible for functioning of VMs, including VM creation, management, destruction and submission of cloudlets to the VM.
- **CloudSim**: this is the class responsible for initializing and starting the simulation environment after all the necessary cloud entities have been defined and later stopping after all the entities have been destroyed.

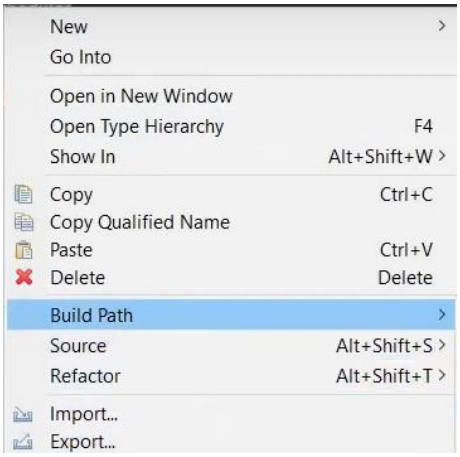
E) SJF algorithm

1. SJF stands for Shortest Job First
2. Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.
3. It is a Greedy Algorithm.
4. It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.
5. It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

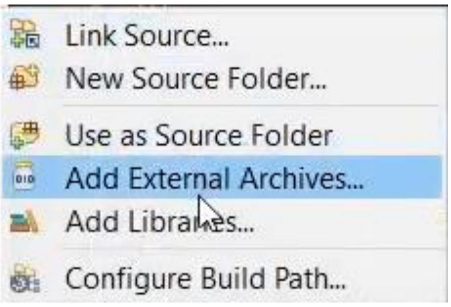
Steps

A) Installation of CloudSim and creation of simulation environment

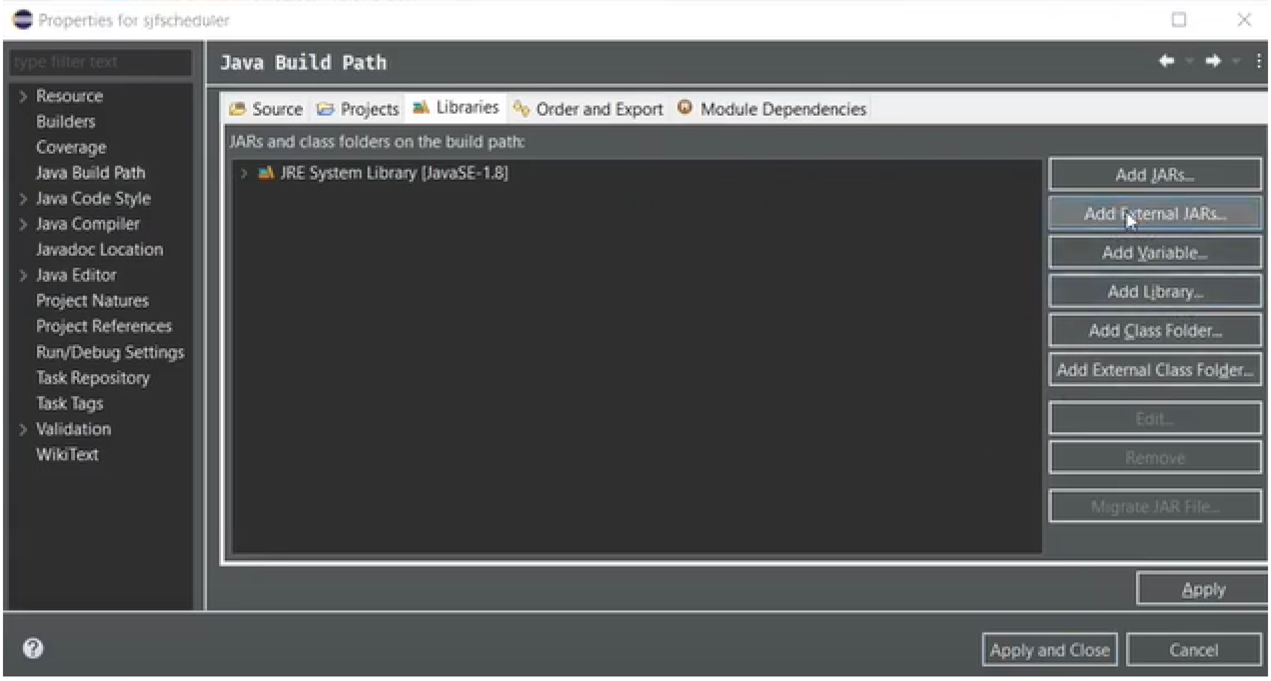
1. Visit https://github.com/Cloudslab/cloudsim/releases to download the CloudSim archives for CloudSim 4.
2. Extract the archive.
3. The jars folder of the extracted archive should contain the following files:
    a) cloudsim-4.0.jar
    b) cloudsim-examples.jar
4. Create a new Java Project using the Eclipse IDE.
5. Right click on the project root and select the Build Path option from the dropdown.



6. Select the Configure Build Path section from the extended dropdown



7. Select the Libraries section and click on Add External JARs field on the pop up

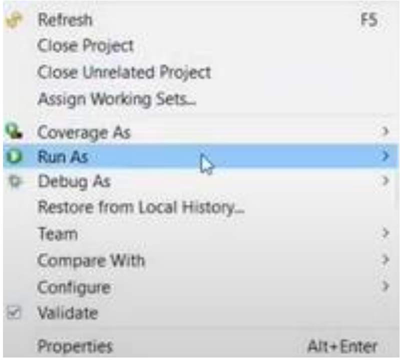


8. Navigate to the jars directory of the CloudSim archive and include the 2 jars in the project.
9. Create a new package in the src directory of the project.

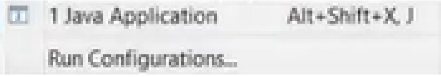
10. Copy the code files for the constants, Data Center Creator, Data Center Broker, Matrix Generator and the SJF scheduler files from the https://github.com/suyash-more/Cloud-Computing-Projects/tree/master/Scheduling-Algorithm-in-CloudSim/src link.
11. Make sure that the package name provided in each file is the same as the previously created package.

B) Execution of code

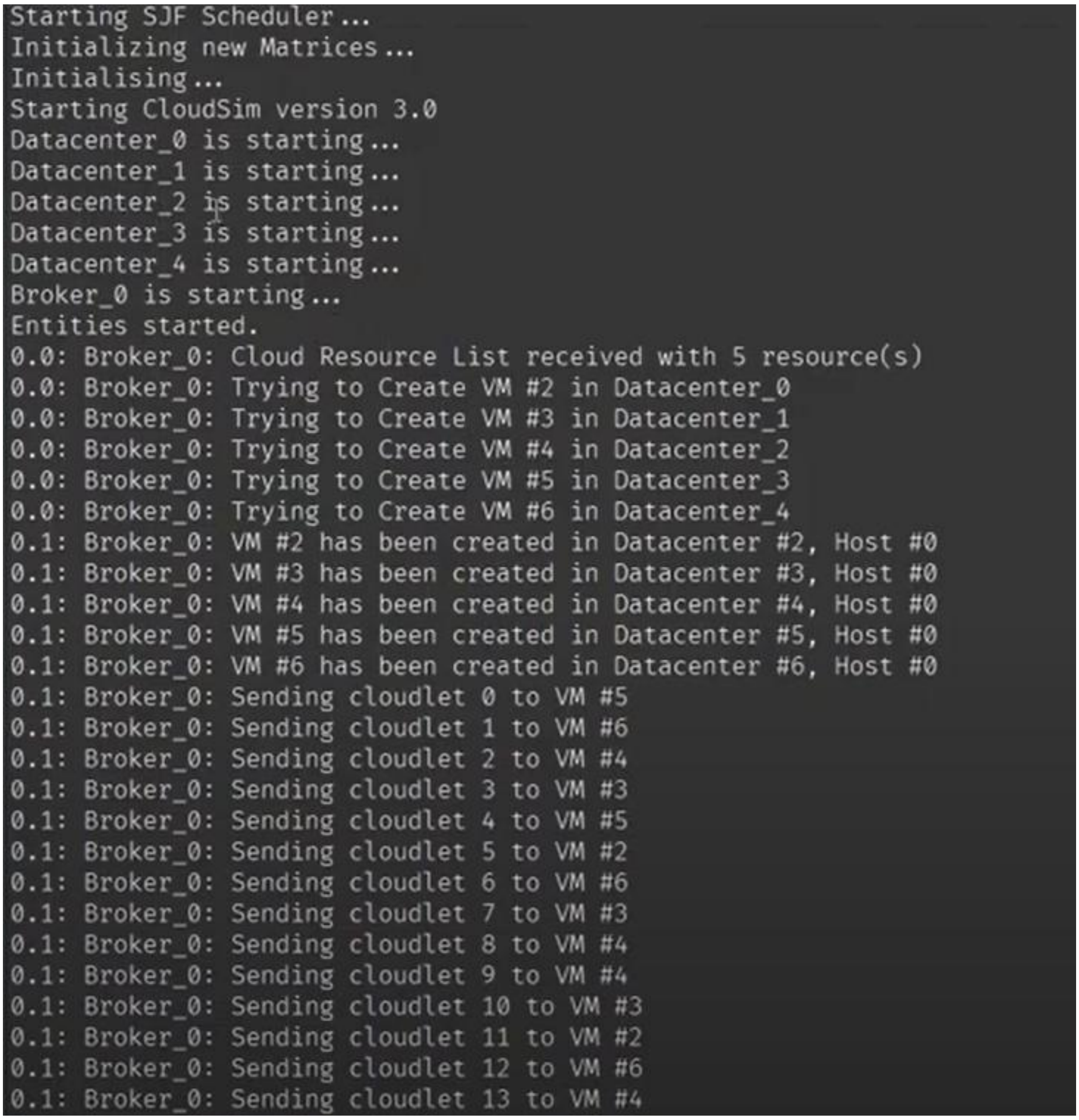
1. Right click on the project



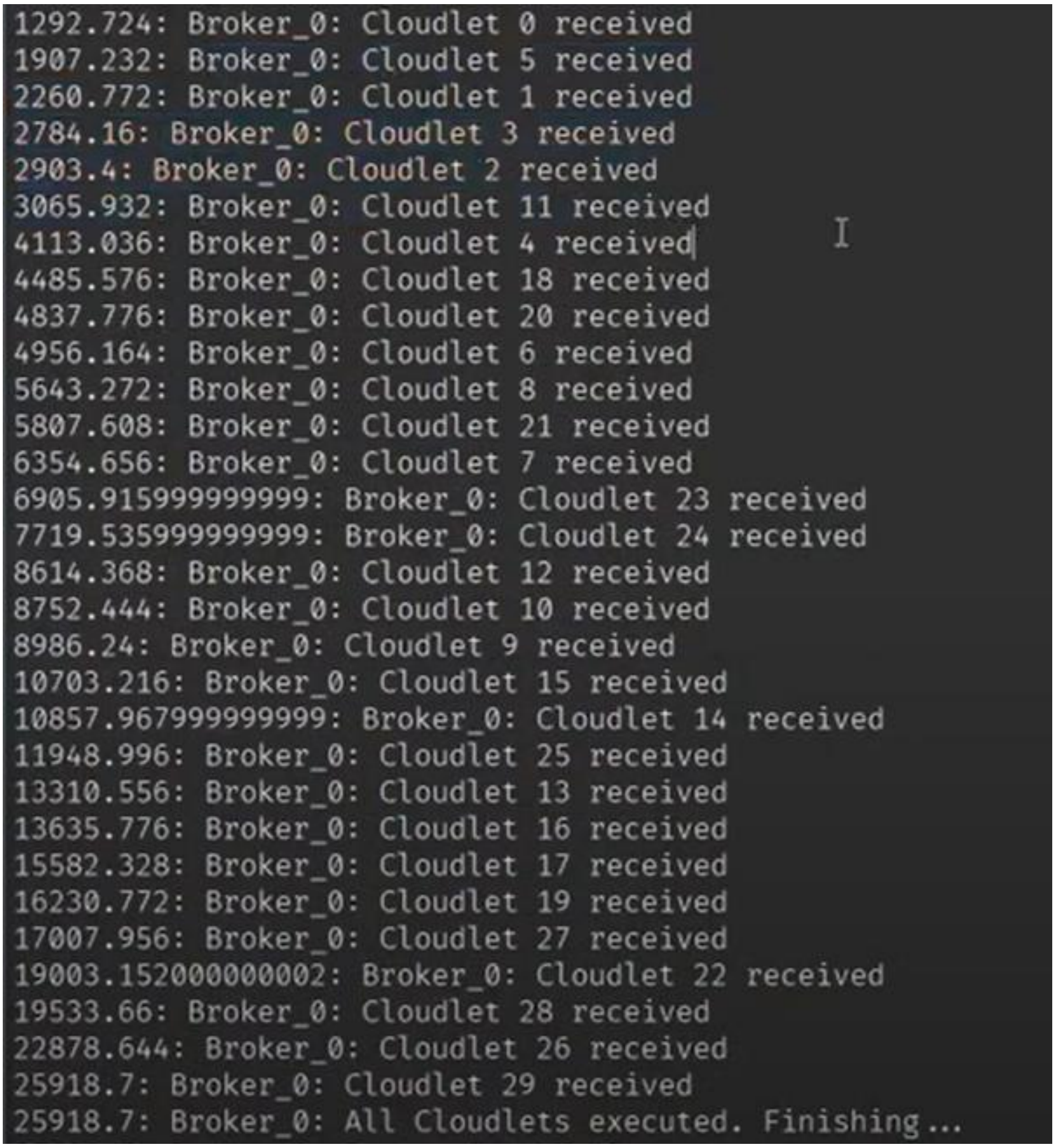
2. Run the project as a Java Application (option in extended dropdown)



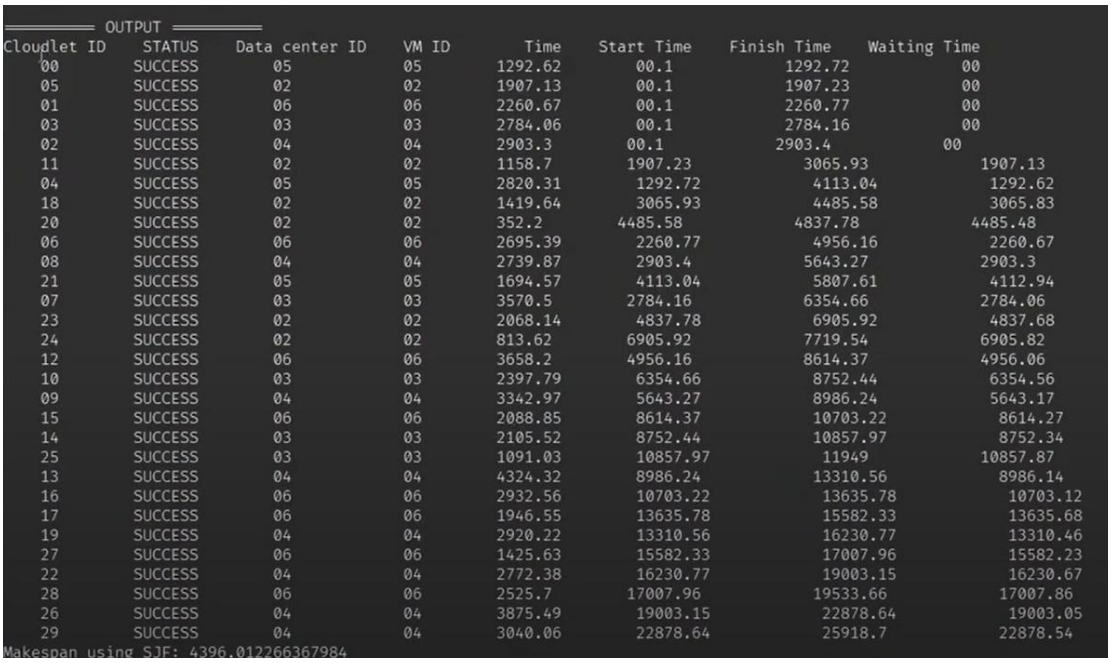
3. The result is displayed on the console



```
Starting SJF Scheduler...
Initializing new Matrices...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Datacenter_2 is starting...
Datacenter_3 is starting...
Datacenter_4 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 5 resource(s)
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_1
0.0: Broker_0: Trying to Create VM #4 in Datacenter_2
0.0: Broker_0: Trying to Create VM #5 in Datacenter_3
0.0: Broker_0: Trying to Create VM #6 in Datacenter_4
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #4 has been created in Datacenter #4, Host #0
0.1: Broker_0: VM #5 has been created in Datacenter #5, Host #0
0.1: Broker_0: VM #6 has been created in Datacenter #6, Host #0
0.1: Broker_0: Sending cloudlet 0 to VM #5
0.1: Broker_0: Sending cloudlet 1 to VM #6
0.1: Broker_0: Sending cloudlet 2 to VM #4
0.1: Broker_0: Sending cloudlet 3 to VM #3
0.1: Broker_0: Sending cloudlet 4 to VM #5
0.1: Broker_0: Sending cloudlet 5 to VM #2
0.1: Broker_0: Sending cloudlet 6 to VM #6
0.1: Broker_0: Sending cloudlet 7 to VM #3
0.1: Broker_0: Sending cloudlet 8 to VM #4
0.1: Broker_0: Sending cloudlet 9 to VM #4
0.1: Broker_0: Sending cloudlet 10 to VM #3
0.1: Broker_0: Sending cloudlet 11 to VM #2
0.1: Broker_0: Sending cloudlet 12 to VM #6
0.1: Broker_0: Sending cloudlet 13 to VM #4
```



```
1292.724: Broker_0: Cloudlet 0 received
1907.232: Broker_0: Cloudlet 5 received
2260.772: Broker_0: Cloudlet 1 received
2784.16: Broker_0: Cloudlet 3 received
2903.4: Broker_0: Cloudlet 2 received
3065.932: Broker_0: Cloudlet 11 received
4113.036: Broker_0: Cloudlet 4 received
4485.576: Broker_0: Cloudlet 18 received
4837.776: Broker_0: Cloudlet 20 received
4956.164: Broker_0: Cloudlet 6 received
5643.272: Broker_0: Cloudlet 8 received
5807.608: Broker_0: Cloudlet 21 received
6354.656: Broker_0: Cloudlet 7 received
6905.915999999999: Broker_0: Cloudlet 23 received
7719.535999999999: Broker_0: Cloudlet 24 received
8614.368: Broker_0: Cloudlet 12 received
8752.444: Broker_0: Cloudlet 10 received
8986.24: Broker_0: Cloudlet 9 received
10703.216: Broker_0: Cloudlet 15 received
10857.967999999999: Broker_0: Cloudlet 14 received
11948.996: Broker_0: Cloudlet 25 received
13310.556: Broker_0: Cloudlet 13 received
13635.776: Broker_0: Cloudlet 16 received
15582.328: Broker_0: Cloudlet 17 received
16230.772: Broker_0: Cloudlet 19 received
17007.956: Broker_0: Cloudlet 27 received
19003.152000000002: Broker_0: Cloudlet 22 received
19533.66: Broker_0: Cloudlet 28 received
22878.644: Broker_0: Cloudlet 26 received
25918.7: Broker_0: Cloudlet 29 received
25918.7: Broker_0: All Cloudlets executed. Finishing...
```



```
======== OUTPUT ========
Cloudlet ID    STATUS    Data center ID    VM ID       Time    Start Time    Finish Time    Waiting Time
    00        SUCCESS         05            05      1292.62      00.1          1292.72           00
    05        SUCCESS         02            02      1907.13      00.1          1907.23           00
    01        SUCCESS         06            06      2260.67      00.1          2260.77           00
    03        SUCCESS         03            03      2784.06      00.1          2784.16           00
    02        SUCCESS         04            04      2903.3      00.1          2903.4           00
    11        SUCCESS         02            02      1158.7      1907.23         3065.93          1907.13
    04        SUCCESS         05            05      2820.31      1292.72         4113.04          1292.62
    18        SUCCESS         02            02      1419.64      3065.93         4485.58          3065.83
    20        SUCCESS         02            02      352.2      4485.58         4837.78          4485.48
    06        SUCCESS         06            06      2695.39      2260.77         4956.16          2260.67
    08        SUCCESS         04            04      2739.87      2903.4         5643.27          2903.3
    21        SUCCESS         05            05      1694.57      4113.04         5807.61          4112.94
    07        SUCCESS         03            03      3570.5      2784.16         6354.66          2784.06
    23        SUCCESS         02            02      2068.14      4837.78         6905.92          4837.68
    24        SUCCESS         02            02      813.62      6905.92         7719.54          6905.82
    12        SUCCESS         06            06      3658.2      4956.16         8614.37          4956.06
    10        SUCCESS         03            03      2397.79      6354.66         8752.44          6354.56
    09        SUCCESS         04            04      3342.97      5643.27         8986.24          5643.17
    15        SUCCESS         06            06      2088.85      8614.37         10703.22          8614.27
    14        SUCCESS         03            03      2105.52      8752.44         10857.97          8752.34
    25        SUCCESS         03            03      1091.03      10857.97         11949          10857.87
    13        SUCCESS         04            04      4324.32      8986.24         13310.56          8986.14
    16        SUCCESS         06            06      2932.56      10703.22         13635.78          10703.12
    17        SUCCESS         06            06      1946.55      13635.78         15582.33          13635.68
    19        SUCCESS         04            04      2920.22      13310.56         16230.77          13310.46
    27        SUCCESS         06            06      1425.63      15582.33         17007.96          15582.23
    22        SUCCESS         04            04      2772.38      16230.77         19003.15          16230.67
    28        SUCCESS         06            06      2525.7      17007.96         19533.66          17007.86
    26        SUCCESS         04            04      3875.49      19003.15         22878.64          19003.05
    29        SUCCESS         04            04      3040.06      22878.64         25918.7          22878.54
Makespan using SJF: 4396.012266367984
```