

Assignment No. – 1

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

1. Install Google Plugin for Eclipse

Read this guide – [how to install Google Plugin for Eclipse](#). If you install the Google App Engine Java SDK together with “**Google Plugin for Eclipse**“, then go to step 2, Otherwise, get the [Google App Engine Java SDK](#) and extract it.

2. Create New Web Application Project

In Eclipse toolbar, click on the Google icon, and select “**New Web Application Project...**”

Figure – New Web Application Project

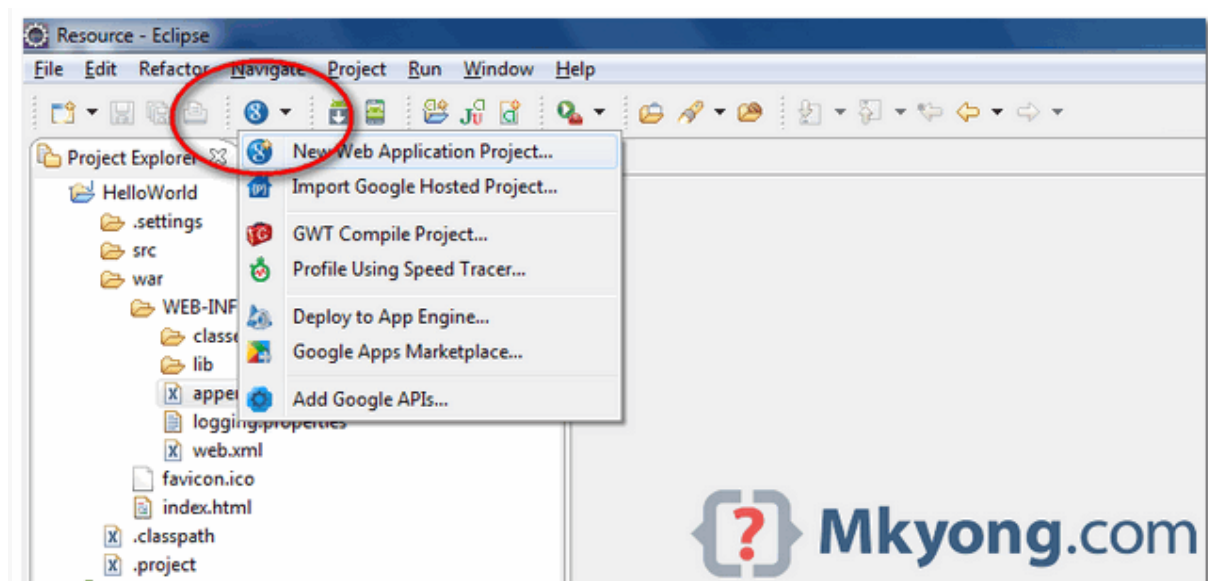
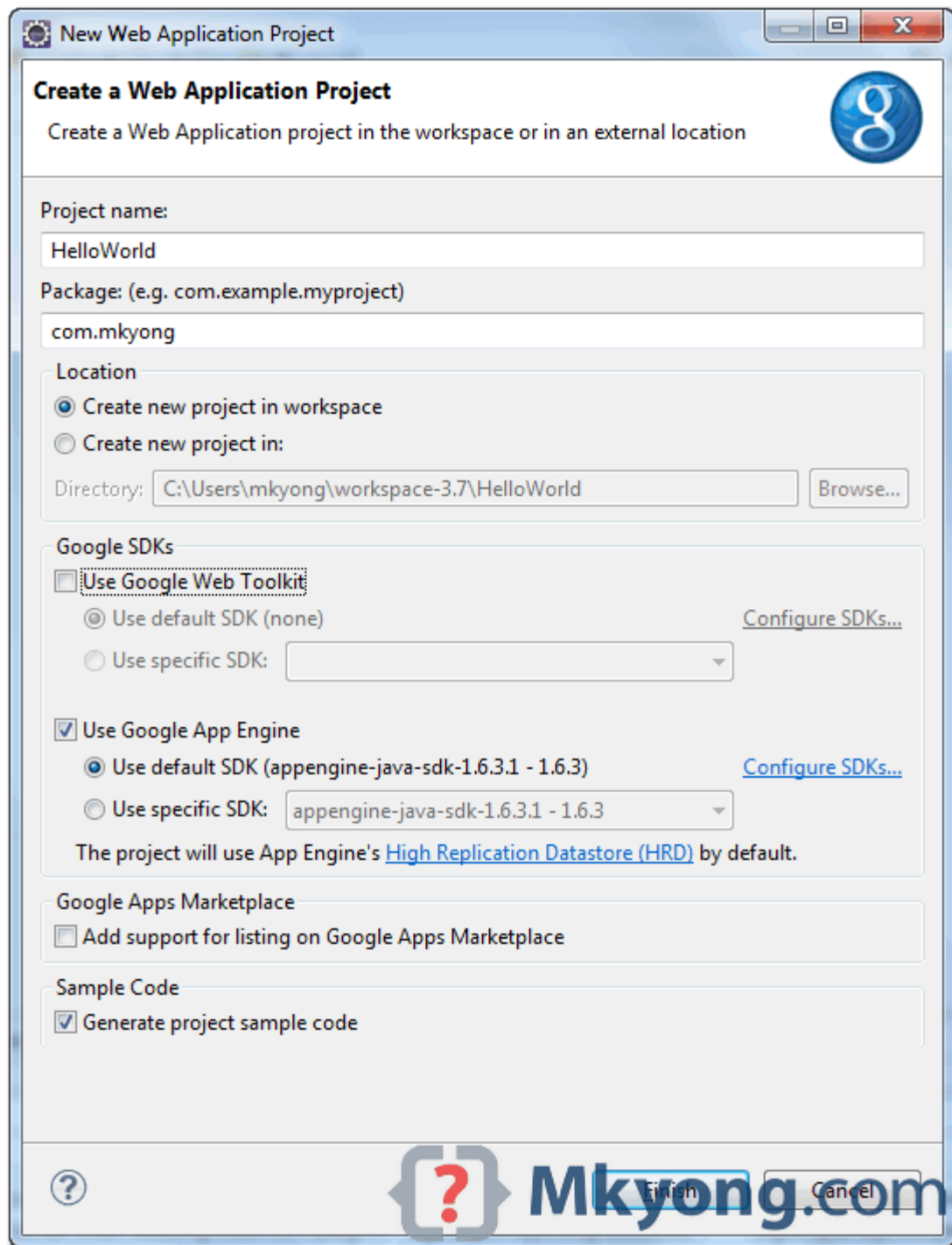


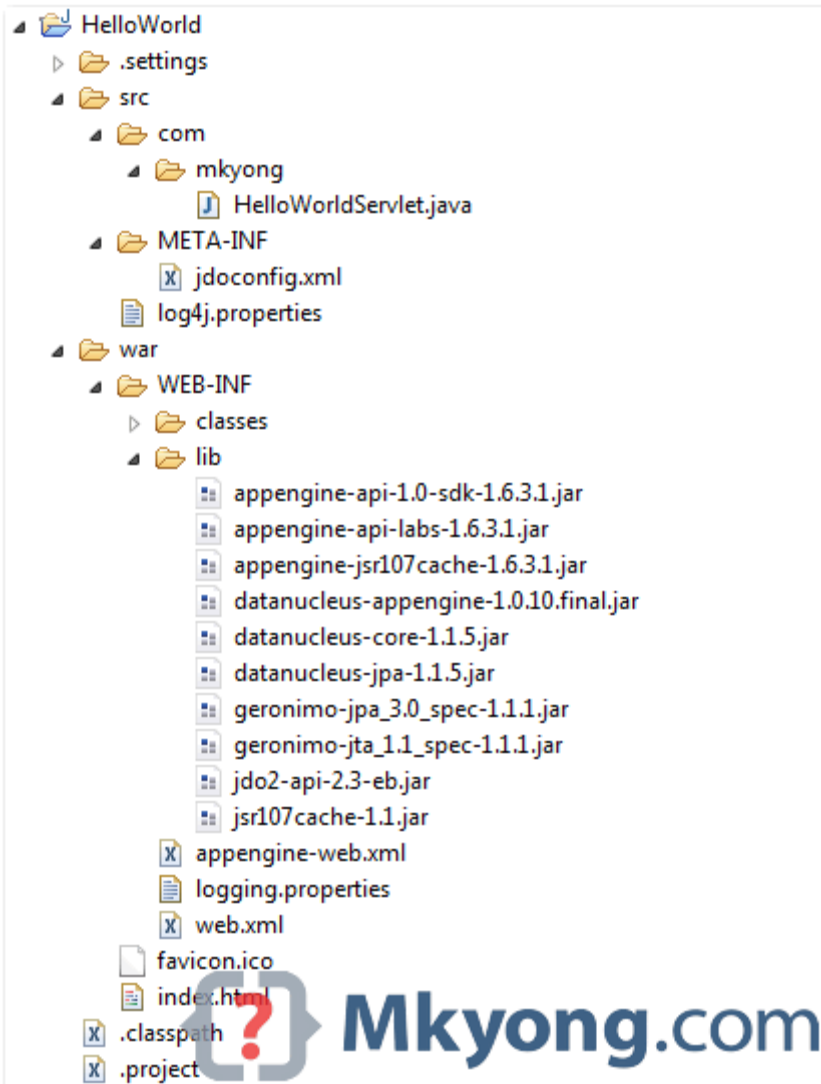
Figure – Deselect the “Google Web Toolkit“, and link your GAE Java SDK via the “configure SDK” link.



Click finished, Google Plugin for Eclipse will generate a sample project automatically.

3. Hello World

Review the generated project directory.



Nothing special, a standard Java web project structure.

```
HelloWorld/  
src/  
...Java source code...  
META-INF/  
...other configuration...  
war/  
...JSPs, images, data files...  
WEB-INF/  
...app configuration...  
lib/  
...JARs for libraries...  
classes/  
...compiled classes...
```

The extra is this file “**appengine-web.xml**“, Google App Engine need this to run and deploy the application.

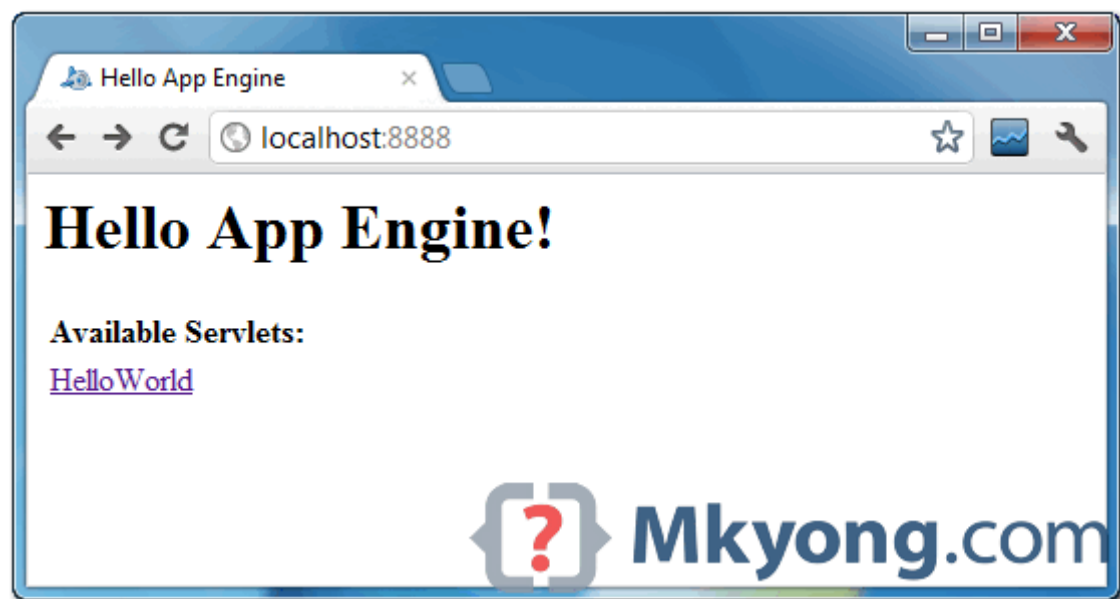
File : appengine-web.xml

4. Run it local

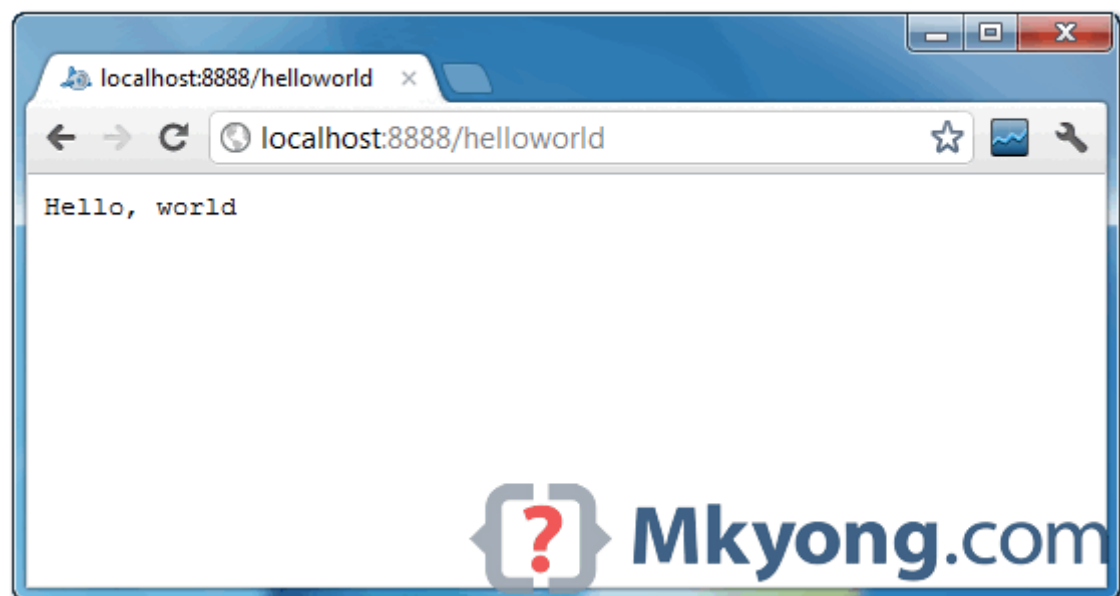
Right click on the project and run as “Web Application“.

Eclipse console :

```
//...  
INFO: The server is running at http://localhost:8888/  
30 Mac 2012 11:13:01 PM com.google.appengine.tools.development.DevAppServerImpl  
start  
INFO: The admin console is running at http://localhost:8888/_ah/admin  
Access URL http://localhost:8888/, see output
```



and also the hello world servlet – <http://localhost:8888/helloworld>



```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application></application>
  <version>1</version>

  <!-- Configure java.util.logging -->
  <system-properties>
    <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/>
  </system-properties>

</appengine-web-app>
```

Assignment No – 2

Use GAE launcher to launch the web applications.

Deploy to Google App Engine

Register an account on <https://appengine.google.com/>, and create an application ID for your web application.

In this demonstration, I created an application ID, named “mkyong123”, and put it in **appengine-web.xml**.

File : appengine-web.xml

```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application>mkyong123</application>
  <version>1</version>

  <!-- Configure java.util.logging -->
  <system-properties>
    <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/>
  </system-properties>

</appengine-web-app>
```

To deploy, see following steps:

Figure 1.1 – Click on GAE deploy button on the toolbar.

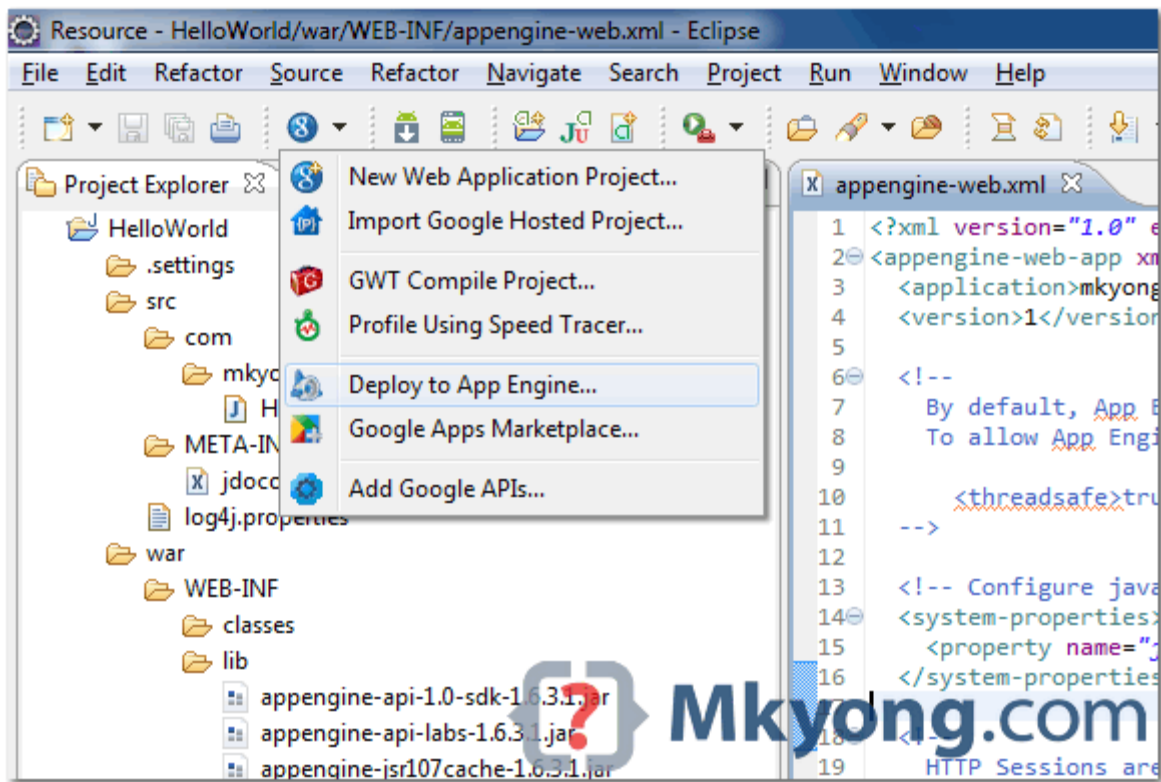


Figure 1.2 – Sign in with your Google account and click on the Deploy button.

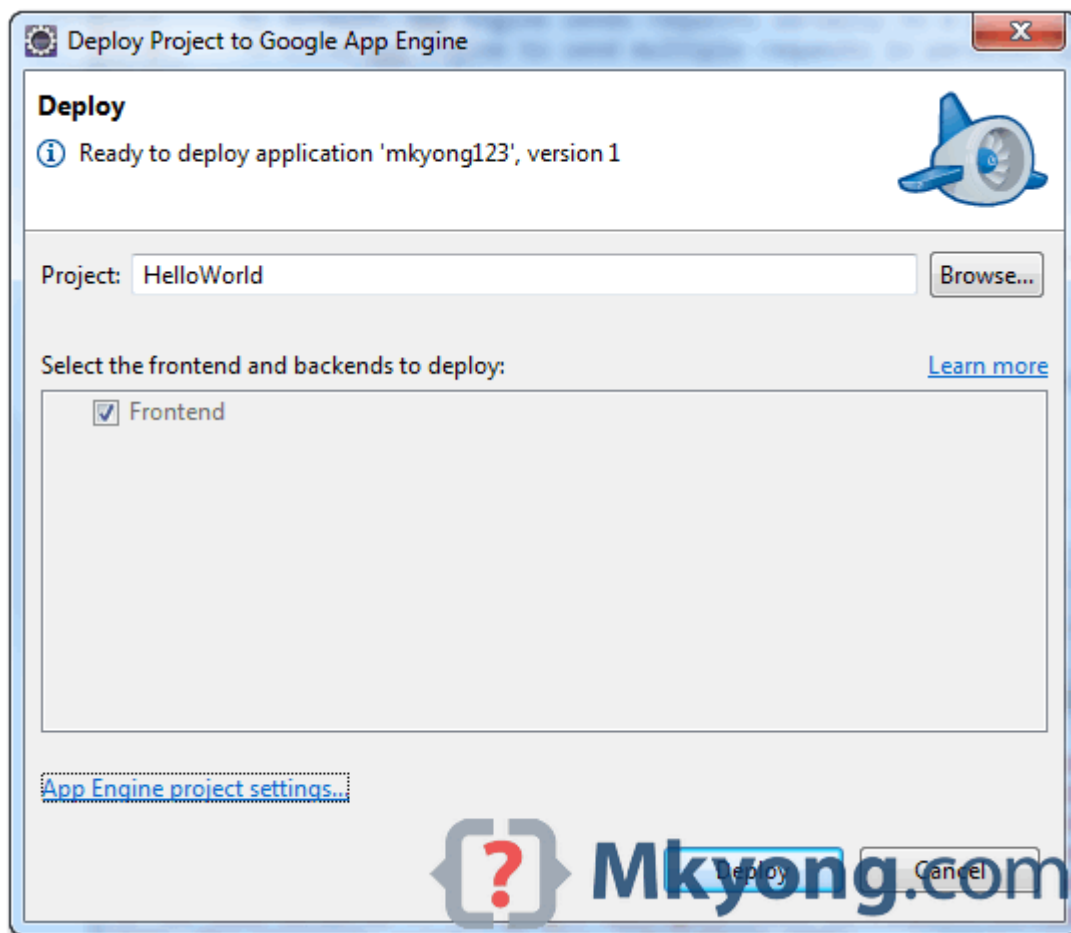


Figure 1.3 – If everything is fine, the hello world web application will be deployed to this URL – <http://mkyong123.appspot.com/>



Done.

Assignment No- 3

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

1. CloudSim

Cloud Computing is one of the hottest topics in town. It has completely transformed how modern-day applications are developed and maintained with high scalability and low latency.

CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. 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.

For example, 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.

2. Steps :

1. Install any of the IDE for running JAVA applications (eclipse recommended)

2. Install JDK and JRE for the same
3. Add the jdk\bin path to the environment variables Open environment variables window, add the following to the path variable

Do include your bin path wherever you have installed JDK

Mine is as following :

> C:\Program Files\Java\jdk-14.0.1\bin

4. Open eclipse in your confined workspace
5. Click on new and open a new **JAVA Project**, give it a name
6. Create a package inside the src folder.
7. Dump in all the files [here](#) inside the package.
8. Now right click on the project and choose **configure build path**.
9. Click on the libraries section and add external jars
10. Extract the Cloudsim.tar file you downloaded
11. Now import the jar files in that Cloudsim.tar into the external jars.
12. Do remember to change the name of the package in all the source files.
13. Now right click on the project and run the project as **JAVA Application**.
14. Select the SJF_Scheduler.java file present in the list.

Constants.java

```
package
<package_name>;

public class Constants {
    public static final int NO_OF_TASKS = 30; // number of Cloudlets;
    public static final int NO_OF_DATA_CENTERS = 5; // number of
Datacenters;
    public static final int POPULATION_SIZE = 25; // Number of
Particles.
}
```

DatacenterCreator.java

```
package
<package_name>;

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
```



```

import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class DatacenterCreator {

    public static Datacenter createDatacenter(String name) {

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

        // 2. A Machine contains one or more PEs or CPUs/Cores.
        // Therefore, should
        //     create a list to store these PEs before creating a
        // Machine.
        List<Pe> peList = new ArrayList<Pe>();

        int mips = 1000;

        // 3. Create PEs and add these into the list.
        peList.add(new Pe(0, new PeProvisionerSimple(mips)));

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

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

```

```

        storage,
        peList,
        new VmSchedulerTimeShared(peList)
    )
); // This is our first machine

// 5. Create a DatacenterCharacteristics object that stores
the
//    properties of a data center: architecture, OS, list of
//    Machines, allocation policy: time- or space-shared, time
zone
//    and its price (G$/Pe time unit).
String arch = "x86";        // system architecture
String os = "Linux";        // operating system
String vmm = "Xen";
double time_zone = 10.0;    // time zone this resource
located
double cost = 3.0;          // the cost of using
processing in this resource
double costPerMem = 0.05;   // the cost of using memory
in this resource
double costPerStorage = 0.1; // the cost of using storage
in this resource
double costPerBw = 0.1;     // the cost of using bw in
this resource
LinkedList<Storage> storageList = new LinkedList<Storage>();
//we are not adding SAN devices by now

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

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

```

```

        e.printStackTrace();
    }
    return datacenter;
}
}

```

GenerateMatrices.java

```

package
<package_name>;

```

```

import java.io.*;

```

```

public class GenerateMatrices {
    private static double[][] commMatrix, execMatrix;
    private File commFile = new File("CommunicationTimeMatrix.txt");
    private File execFile = new File("ExecutionTimeMatrix.txt");

```

```

    public GenerateMatrices() {
        commMatrix = new
double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
        execMatrix = new
double[Constants.NO_OF_TASKS][Constants.NO_OF_DATA_CENTERS];
        try {
            if (commFile.exists() && execFile.exists()) {
                readCostMatrix();
            } else {
                initCostMatrix();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```

    private void initCostMatrix() throws IOException {
        System.out.println("Initializing new Matrices...");
        BufferedWriter commBufferedWriter = new BufferedWriter(new
FileWriter(commFile));

```

```

        BufferedWriter execBufferedWriter = new BufferedWriter(new
        FileWriter(execFile));

        for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
            for (int j = 0; j < Constants.NO_OF_DATA_CENTERS; j++) {
                commMatrix[i][j] = Math.random() * 600 + 20;
                execMatrix[i][j] = Math.random() * 500 + 10;

                commBufferedWriter.write(String.valueOf(commMatrix[i][j]) + ' ');

                execBufferedWriter.write(String.valueOf(execMatrix[i][j]) + ' ');
            }
            commBufferedWriter.write('\n');
            execBufferedWriter.write('\n');
        }
        commBufferedWriter.close();
        execBufferedWriter.close();
    }

```

```

    private void readCostMatrix() throws IOException {
        System.out.println("Reading the Matrices...");
        BufferedReader commBufferedReader = new BufferedReader(new
        FileReader(commFile));

```

```

        int i = 0, j = 0;
        do {
            String line = commBufferedReader.readLine();
            for (String num : line.split(" ")) {
                commMatrix[i][j++] = new Double(num);
            }
            ++i;
            j = 0;
        } while (commBufferedReader.ready());

```

```

        BufferedReader execBufferedReader = new BufferedReader(new
        FileReader(execFile));

```

```

        i = j = 0;
        do {

```

```

        String line = execBufferedReader.readLine();
        for (String num : line.split(" ")) {
            execMatrix[i][j++] = new Double(num);
        }
        ++i;
        j = 0;
    } while (execBufferedReader.ready());
}

public static double[][] getCommMatrix() {
    return commMatrix;
}

public static double[][] getExecMatrix() {
    return execMatrix;
}
}

```

SJFDatacenterBroker.java

```

package
<package_name>;

```

```

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.core.CloudSimTags;
import org.cloudbus.cloudsim.core.SimEvent;

import java.util.ArrayList;
import java.util.List;

public class SJFDatacenterBroker extends DatacenterBroker {

    SJFDatacenterBroker(String name) throws Exception {
        super(name);
    }
}

```

```

    }

    public void scheduleTaskstoVms() {
        int reqTasks = cloudletList.size();
        int reqVms = vmList.size();
        Vm vm = vmList.get(0);

        for (int i = 0; i < reqTasks; i++) {
            bindCloudletToVm(i, (i % reqVms));
            System.out.println("Task" +
cloudletList.get(i).getCloudletId() + " is bound with VM" +
vmList.get(i % reqVms).getId());
        }

        //System.out.println("reqTasks: "+ reqTasks);

        ArrayList<Cloudlet> list = new ArrayList<Cloudlet>();
        for (Cloudlet cloudlet : getCloudletReceivedList()) {
            list.add(cloudlet);
        }

        //setCloudletReceivedList(null);

        Cloudlet[] list2 = list.toArray(new Cloudlet[list.size()]);

        //System.out.println("size :"+list.size());

        Cloudlet temp = null;

        int n = list.size();

        for (int i = 0; i < n; i++) {
            for (int j = 1; j < (n - i); j++) {
                if (list2[j - 1].getCloudletLength() / (vm.getMips() *
vm.getNumberOfPes()) > list2[j].getCloudletLength() / (vm.getMips() *
vm.getNumberOfPes())) {

```

```

        //swap the elements!
        //swap(list2[j-1], list2[j]);
        temp = list2[j - 1];
        list2[j - 1] = list2[j];
        list2[j] = temp;
    }
    // printNumbers(list2);
}
}

ArrayList<Cloudlet> list3 = new ArrayList<Cloudlet>();

for (int i = 0; i < list2.length; i++) {
    list3.add(list2[i]);
}
//printNumbers(list);

setCloudletReceivedList(list);

//System.out.println("\n\tSJFS Broker Schedules\n");
//System.out.println("\n");
}

public void printNumber(Cloudlet[] list) {
    for (int i = 0; i < list.length; i++) {
        System.out.print(" " + list[i].getCloudletId());
        System.out.println(list[i].getCloudletStatusString());
    }
    System.out.println();
}

public void printNumbers(ArrayList<Cloudlet> list) {
    for (int i = 0; i < list.size(); i++) {
        System.out.print(" " + list.get(i).getCloudletId());
    }
    System.out.println();
}

@Override

```

```

protected void processCloudletReturn(SimEvent ev) {
    Cloudlet cloudlet = (Cloudlet) ev.getData();
    getCloudletReceivedList().add(cloudlet);
    Log.println(CloudSim.clock() + ": " + getName() + ":
Cloudlet " + cloudlet.getCloudletId()
        + " received");
    cloudletsSubmitted--;
    if (getCloudletList().size() == 0 && cloudletsSubmitted == 0)
    {
        scheduleTaskstoVms();
        cloudletExecution(cloudlet);
    }
}

protected void cloudletExecution(Cloudlet cloudlet) {

    if (getCloudletList().size() == 0 && cloudletsSubmitted == 0)
    { // all cloudlets executed
        Log.println(CloudSim.clock() + ": " + getName() + ": All
Cloudlets executed. Finishing...");
        clearDatacenters();
        finishExecution();
    } else { // some cloudlets haven't finished yet
        if (getCloudletList().size() > 0 && cloudletsSubmitted ==
0) {
            // all the cloudlets sent finished. It means that some
bount

            // cloudlet is waiting its VM be created
            clearDatacenters();
            createVmsInDatacenter(0);
        }
    }
}

@Override
protected void processResourceCharacteristics(SimEvent ev) {
    DatacenterCharacteristics characteristics =
(DatacenterCharacteristics) ev.getData();

    getDatacenterCharacteristicsList().put(characteristics.getId(),
characteristics);
}

```



```

        if (getDatacenterCharacteristicsList().size() ==
getDatacenterIdsList().size()) {
            distributeRequestsForNewVmsAcrossDatacenters();
        }
    }

    protected void distributeRequestsForNewVmsAcrossDatacenters() {
        int numberOfVmsAllocated = 0;
        int i = 0;

        final List<Integer> availableDatacenters =
getDatacenterIdsList();

        for (Vm vm : getVmList()) {
            int datacenterId = availableDatacenters.get(i++ %
availableDatacenters.size());
            String datacenterName =
CloudSim.getEntityName(datacenterId);

            if (!getVmsToDatacentersMap().containsKey(vm.getId())) {
                Log.println(CloudSim.clock() + ": " + getName() + ":
Trying to Create VM #" + vm.getId() + " in " + datacenterName);
                sendNow(datacenterId, CloudSimTags.VM_CREATE_ACK, vm);
                numberOfVmsAllocated++;
            }
        }

        setVmsRequested(numberOfVmsAllocated);
        setVmsAcks(0);
    }
}

```

SJF_Scheduler.java

```

package
<package_name>;

```

```

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
//import utils.Constants;
//import utils.DatacenterCreator;
//import utils.GenerateMatrices;

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

public class SJF_Scheduler {

    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static Datacenter[] datacenter;
    private static double[][] commMatrix;
    private static double[][] execMatrix;

    private static List<Vm> createVM(int userId, int vms) {
        //Creates a container to store VMs. This list is passed to the
        broker later
        LinkedList<Vm> list = new LinkedList<Vm>();

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

        //create VMs
        Vm[] vm = new Vm[vms];

```

```

        for (int i = 0; i < vms; i++) {
            vm[i] = new Vm(datacenter[i].getId(), userId, mips,
                pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShared());
            list.add(vm[i]);
        }

```

```

        return list;
    }

```

```

    private static List<Cloudlet> createCloudlet(int userId, int
cloudlets, int idShift) {

```

```

        // Creates a container to store Cloudlets
        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

```

```

        //cloudlet parameters
        long fileSize = 300;
        long outputSize = 300;
        int pesNumber = 1;
        UtilizationModel utilizationModel = new
UtilizationModelFull();

```

```

        Cloudlet[] cloudlet = new Cloudlet[cloudlets];

```

```

        for (int i = 0; i < cloudlets; i++) {
            int dcId = (int) (Math.random() *
Constants.NO_OF_DATA_CENTERS);
            long length = (long) (1e3 * (commMatrix[i][dcId] +
execMatrix[i][dcId]));
            cloudlet[i] = new Cloudlet(idShift + i, length, pesNumber,
fileSize, outputSize, utilizationModel, utilizationModel,
utilizationModel);
            // setting the owner of these Cloudlets
            cloudlet[i].setUserId(userId);
            cloudlet[i].setVmId(dcId + 2);
            list.add(cloudlet[i]);
        }
        return list;
    }

```

```

    public static void main(String[] args) {

```

```

Log.println("Starting SJF Scheduler...");

new GenerateMatrices();
execMatrix = GenerateMatrices.getExecMatrix();
commMatrix = GenerateMatrices.getCommMatrix();

try {
    int num_user = 1;    // number of grid users
    Calendar calendar = Calendar.getInstance();
    boolean trace_flag = false; // mean trace events

    CloudSim.init(num_user, calendar, trace_flag);

    // Second step: Create Datacenters
    datacenter = new Datacenter[Constants.NO_OF_DATA_CENTERS];
    for (int i = 0; i < Constants.NO_OF_DATA_CENTERS; i++) {
        datacenter[i] =
DatacenterCreator.createDatacenter("Datacenter_" + i);
    }

    //Third step: Create Broker
    SJFDatacenterBroker broker = createBroker("Broker_0");
    int brokerId = broker.getId();

    //Fourth step: Create VMs and Cloudlets and send them to
broker
    vmList = createVM(brokerId, Constants.NO_OF_DATA_CENTERS);
    cloudletList = createCloudlet(brokerId,
Constants.NO_OF_TASKS, 0);

    broker.submitVmList(vmList);
    broker.submitCloudletList(cloudletList);

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

    // Final step: Print results when simulation is over

```

```

        List<Cloudlet> newList = broker.getCloudletReceivedList();

//newList.addAll(globalBroker.getBroker().getCloudletReceivedList());

        CloudSim.stopSimulation();

        printCloudletList(newList);

        Log.println(SJF_Scheduler.class.getName() + "
finished!");
    } catch (Exception e) {
        e.printStackTrace();
        Log.println("The simulation has been terminated due to
an unexpected error");
    }
}

    private static SJFDatacenterBroker createBroker(String name)
throws Exception {
        return new SJFDatacenterBroker(name);
    }

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

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

```

```

        indent + "Waiting Time");

    DecimalFormat dft = new DecimalFormat("###.##");
    dft.setMinimumIntegerDigits(2);
    for (int i = 0; i < size; i++) {
        cloudlet = list.get(i);
        Log.print(indent + dft.format(cloudlet.getCloudletId()) +
indent + indent);

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

            Log.println(indent + indent +
dft.format(cloudlet.getResourceId()) +
                indent + indent + indent +
dft.format(cloudlet.getVmId()) +
                indent + indent +
dft.format(cloudlet.getActualCPUTime()) +
                indent + indent +
dft.format(cloudlet.getExecStartTime()) +
                indent + indent + indent +
dft.format(cloudlet.getFinishTime())+
                indent + indent + indent +
dft.format(cloudlet.getWaitingTime()));
        }
    }
    double makespan = calcMakespan(list);
    Log.println("Makespan using SJF: " + makespan);
}

private static double calcMakespan(List<Cloudlet> list) {
    double makespan = 0;
    double[] dcWorkingTime = new
double[Constants.NO_OF_DATA_CENTERS];

    for (int i = 0; i < Constants.NO_OF_TASKS; i++) {
        int dcId = list.get(i).getVmId() %
Constants.NO_OF_DATA_CENTERS;
        if (dcWorkingTime[dcId] != 0) --dcWorkingTime[dcId];
        dcWorkingTime[dcId] += execMatrix[i][dcId] +
commMatrix[i][dcId];
    }
}

```

```

        makespan = Math.max(makespan, dcWorkingTime[dcId]);
    }
    return makespan;
}
}

```

Output-Starting SJF Scheduler...

Reading the 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 #2

0.1: Broker_0: Sending cloudlet 1 to VM #3

0.1: Broker_0: Sending cloudlet 2 to VM #5

0.1: Broker_0: Sending cloudlet 3 to VM #6

0.1: Broker_0: Sending cloudlet 4 to VM #4

0.1: Broker_0: Sending cloudlet 5 to VM #5

0.1: Broker_0: Sending cloudlet 6 to VM #2

0.1: Broker_0: Sending cloudlet 7 to VM #4

0.1: Broker_0: Sending cloudlet 8 to VM #5

0.1: Broker_0: Sending cloudlet 9 to VM #2

0.1: Broker_0: Sending cloudlet 10 to VM #5

0.1: Broker_0: Sending cloudlet 11 to VM #2

0.1: Broker_0: Sending cloudlet 12 to VM #2

0.1: Broker_0: Sending cloudlet 13 to VM #5

0.1: Broker_0: Sending cloudlet 14 to VM #2

0.1: Broker_0: Sending cloudlet 15 to VM #6

0.1: Broker_0: Sending cloudlet 16 to VM #4

0.1: Broker_0: Sending cloudlet 17 to VM #3

0.1: Broker_0: Sending cloudlet 18 to VM #4

0.1: Broker_0: Sending cloudlet 19 to VM #2

0.1: Broker_0: Sending cloudlet 20 to VM #4

0.1: Broker_0: Sending cloudlet 21 to VM #6

0.1: Broker_0: Sending cloudlet 22 to VM #5

0.1: Broker_0: Sending cloudlet 23 to VM #2

0.1: Broker_0: Sending cloudlet 24 to VM #3

0.1: Broker_0: Sending cloudlet 25 to VM #3

0.1: Broker_0: Sending cloudlet 26 to VM #2

0.1: Broker_0: Sending cloudlet 27 to VM #3

0.1: Broker_0: Sending cloudlet 28 to VM #5

```

0.1: Broker_0: Sending cloudlet 29 to VM #3
868.432: Broker_0: Cloudlet 4 received
1584.8519999999999: Broker_0: Cloudlet 2 received
2113.7999999999997: Broker_0: Cloudlet 0 received
2651.948: Broker_0: Cloudlet 3 received
3065.096: Broker_0: Cloudlet 1 received
3243.2479999999996: Broker_0: Cloudlet 6 received
4550.252: Broker_0: Cloudlet 15 received
4782.792: Broker_0: Cloudlet 7 received
4948.0599999999995: Broker_0: Cloudlet 5 received
5149.2480000000005: Broker_0: Cloudlet 21 received
5353.384: Broker_0: Cloudlet 17 received
6204.392: Broker_0: Cloudlet 9 received
7303.8240000000005: Broker_0: Cloudlet 16 received
7371.2199999999999: Broker_0: Cloudlet 8 received
7734.5679999999999: Broker_0: Cloudlet 11 received
9139.0639999999998: Broker_0: Cloudlet 12 received
9307.632: Broker_0: Cloudlet 24 received
9520.328: Broker_0: Cloudlet 10 received
9738.152: Broker_0: Cloudlet 18 received
10369.5399999999999: Broker_0: Cloudlet 14 received
11471.412: Broker_0: Cloudlet 25 received
11966.0719999999998: Broker_0: Cloudlet 19 received
12429.884: Broker_0: Cloudlet 13 received
13122.836: Broker_0: Cloudlet 20 received
13149.664: Broker_0: Cloudlet 27 received
13474.136: Broker_0: Cloudlet 22 received
13920.868: Broker_0: Cloudlet 29 received
15347.0119999999999: Broker_0: Cloudlet 23 received
15570.2200000000001: Broker_0: Cloudlet 28 received
16329.5119999999999: Broker_0: Cloudlet 26 received
16329.5119999999999: Broker_0: All Cloudlets executed. Finishing...
16329.5119999999999: Broker_0: Destroying VM #2
16329.5119999999999: Broker_0: Destroying VM #3
16329.5119999999999: Broker_0: Destroying VM #4
16329.5119999999999: Broker_0: Destroying VM #5
16329.5119999999999: Broker_0: Destroying VM #6
Broker_0 is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Datacenter_2 is shutting down...
Datacenter_3 is shutting down...
Datacenter_4 is shutting down...
Broker_0 is shutting down...
Simulation completed.
Simulation completed.

```

===== OUTPUT =====

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time
Finish Time	Waiting Time				
04	SUCCESS	04	04	868.33	00.1
868.43	00				
02	SUCCESS	05	05	1584.75	00.1
1584.85	00				
00	SUCCESS	02	02	2113.7	00.1
2113.8	00				

03	SUCCESS	06	06	2651.85	00.1
2651.95	00				
01	SUCCESS	03	03	3065	00.1
3065.1	00				
06	SUCCESS	02	02	1129.45	2113.8
3243.25	2113.7				
15	SUCCESS	06	06	1898.3	2651.95
4550.25	2651.85				
07	SUCCESS	04	04	3914.36	868.43
4782.79	868.33				
05	SUCCESS	05	05	3363.21	1584.85
4948.06	1584.75				
21	SUCCESS	06	06	599	4550.25
5149.25	4550.15				
17	SUCCESS	03	03	2288.29	3065.1
5353.38	3065				
09	SUCCESS	02	02	2961.14	3243.25
6204.39	3243.15				
16	SUCCESS	04	04	2521.03	4782.79
7303.82	4782.69				
08	SUCCESS	05	05	2423.16	4948.06
7371.22	4947.96				
11	SUCCESS	02	02	1530.18	6204.39
7734.57	6204.29				
12	SUCCESS	02	02	1404.5	7734.57
9139.06	7734.47				
24	SUCCESS	03	03	3954.25	5353.38
9307.63	5353.28				
10	SUCCESS	05	05	2149.11	7371.22
9520.33	7371.12				
18	SUCCESS	04	04	2434.33	7303.82
9738.15	7303.72				
14	SUCCESS	02	02	1230.48	9139.06
10369.54	9138.96				
25	SUCCESS	03	03	2163.78	9307.63
11471.41	9307.53				
19	SUCCESS	02	02	1596.53	10369.54
11966.07	10369.44				
13	SUCCESS	05	05	2909.56	9520.33
12429.88	9520.23				
20	SUCCESS	04	04	3384.68	9738.15
13122.84	9738.05				
27	SUCCESS	03	03	1678.25	11471.41
13149.66	11471.31				
22	SUCCESS	05	05	1044.25	12429.88
13474.14	12429.78				
29	SUCCESS	03	03	771.2	13149.66
13920.87	13149.56				
23	SUCCESS	02	02	3380.94	11966.07
15347.01	11965.97				
28	SUCCESS	05	05	2096.08	13474.14
15570.22	13474.04				
26	SUCCESS	02	02	982.5	15347.01
16329.51	15346.91				

Makespan using SJF: 5401.400917391503
 custom_package.SJF_Scheduler finished!

Assignment No – 4

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

Steps:

1. Download and install Oracle's Virtual Box. (Reboot needed after installation)
2. Download Ubuntu VMDK Image.
3. Launch Virtualbox and create a new VM.
4. Click on new and mention the Name and the machine folder along with the Type and Version of the Machine to be created.
5. Assign memory size for our VM (1024 MB sufficient for now).
6. Select the option **Use an existing virtual hard disk file** and locate the downloaded VMDK image below and create VM.
7. Now we have to create a NAT Network so go to **File -> Preferences -> Network -> Add a New NAT Network (Click on +)**
8. Right click and edit the Network name and CIDR if needed. Example :

Name - My VMbox Network

CIDR - 172.168.2.0/24 and save the changes.

9. Repeat the process of launching the VM for 2 instances.
10. Now go to the setting, go to the network setting and change the adapter to NAT Network and select the NAT Network you made (in our case : **My VMbox Network**) and click ok.
11. Launch the VM now.
12. Install the net-tools to know the IP's of the instance.

```
$ sudo apt install net-tools
```

```
$ sudo apt update
```

13. To know the IP address

```
$ ifconfig
```

Now the IP will be in the range of **172.168.2.***

* - any number in the range of 1 to 254 (total 256 addresses)

14. Now create a file and write something into it.

```
$ touch tranfer.txt
```

```
$ nano transfer.txt  
-> hey, How are you?  
ctrl + X and save
```

Some Commands for Linux Based Distro:

ls - list all the files and directories
cat - show the content inside a file
scp - it will help us to copy files from one vm to other
cd - change directory
mkdir - make a new directory
touch - it makes a new file
nano - nano is a editor inside linux os

15. If your file is on the VM with IP **172.168.2.4** and the second VM's IP is **172.168.2.5**.

16. Transfer the file using **SCP**

```
$ scp transfer.txt vagrant@172.168.2.5:/home/vagrant
```

Put in the password of the 2nd VM and done.

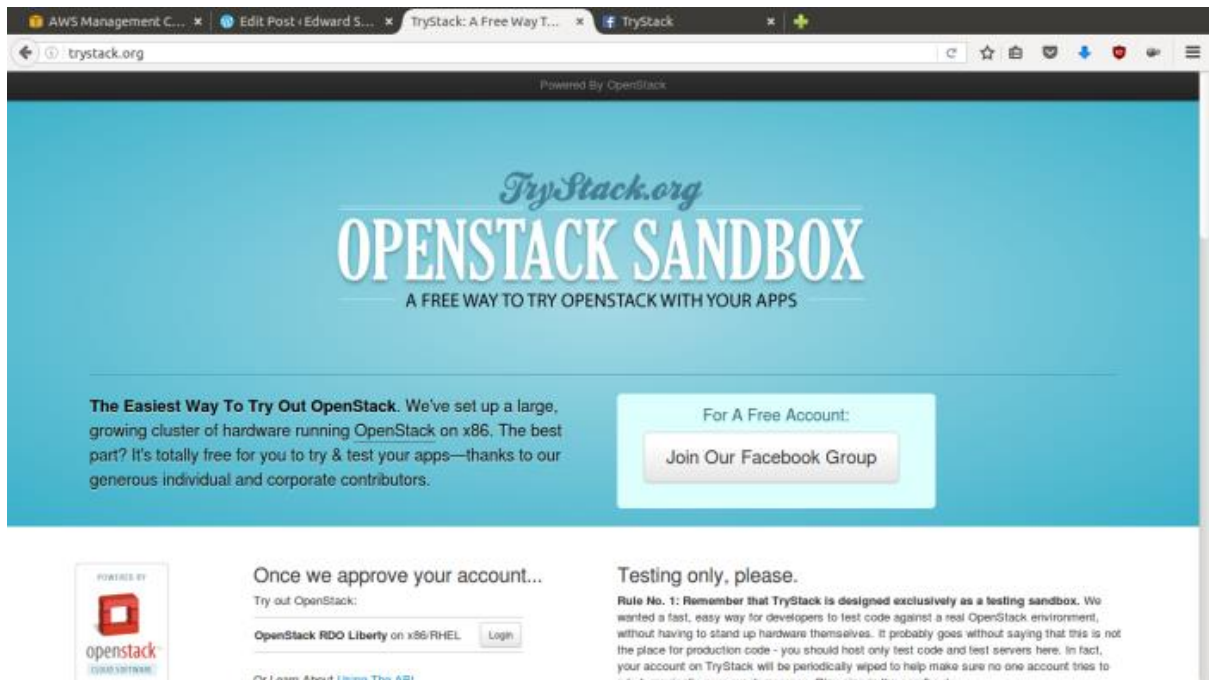
17. Check for the file in the Second VM under the **/home/vagrant** directory.

Assignment – 5

Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)

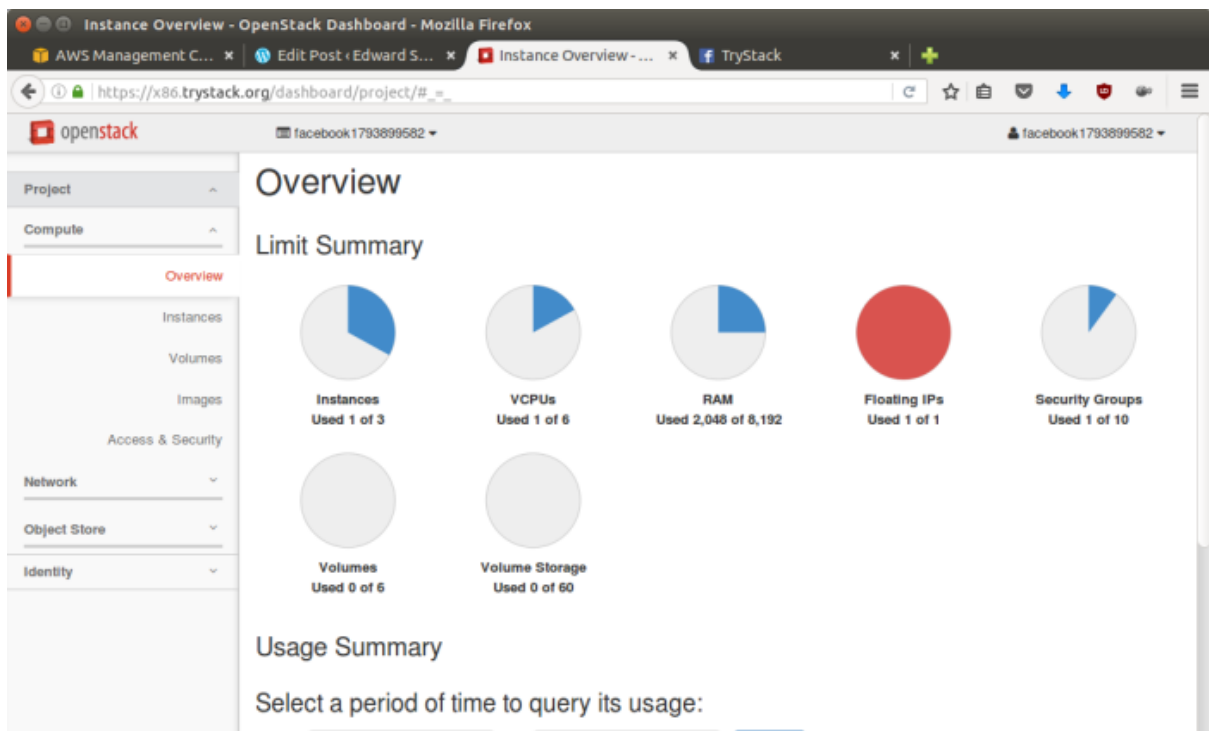
OpenStack is an open-source software cloud computing platform. OpenStack is primarily used for deploying an infrastructure as a service (IaaS) solution like Amazon Web Service (AWS). In other words, you can *make your own AWS* by using OpenStack. If you want to try out OpenStack, **TryStack** is the easiest and free way to do it.

In order to try OpenStack in TryStack, you must register yourself by joining **TryStack Facebook Group**. The acceptance of group needs a couple days because it's approved manually. After you have been accepted in the TryStack Group, you can log in TryStack.



TryStack.org Homepage

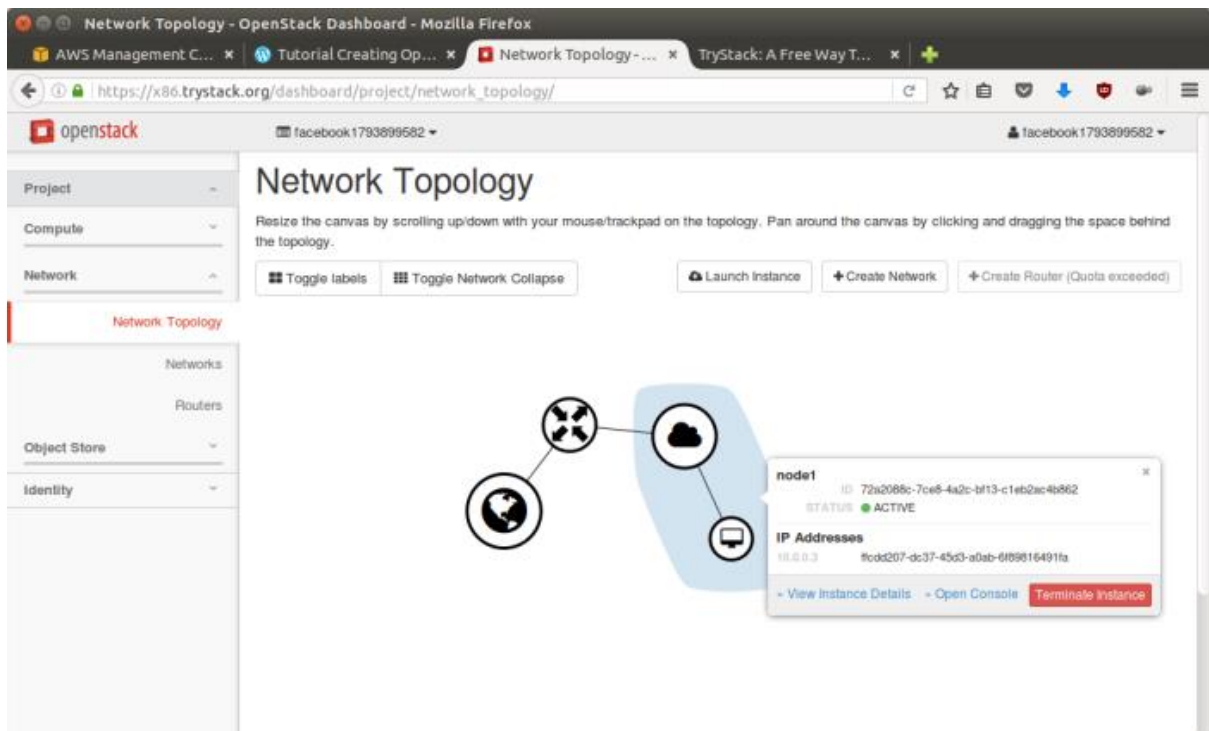
I assume that you already join to the Facebook Group and login to the dashboard. After you log in to the TryStack, you will see the Compute Dashboard like:



OpenStack Compute Dashboard

Overview:

The instance will be accessible through the internet (have a public IP address). The final topology will like:



Network topology

As you see from the image above, the instance will be connected to a local network and the local network will be connected to internet.

Step 1: Create Network

Network? Yes, the network in here is our own local network. So, your instances will be not mixed up with the others. You can imagine this as your own LAN (Local Area Network) in the cloud.

1. Go to **Network > Networks** and then click **Create Network**.
2. In **Network** tab, fill **Network Name** for example internal and then click **Next**.
3. In **Subnet** tab,
 1. Fill **Network Address** with appropriate CIDR, for example 192.168.1.0/24. Use **private network CIDR block** as the best practice.
 2. Select **IP Version** with appropriate IP version, in this case **IPv4**.
 3. Click **Next**.
4. In **Subnet Details** tab, fill **DNS Name Servers** with 8.8.8.8 (Google DNS) and then click **Create**.

Step 2: Create Instance

Now, we will create an instance. The instance is a virtual machine in the cloud, like AWS EC2. You need the instance to connect to the network that we just created in the previous step.

1. Go to **Compute > Instances** and then click **Launch Instance**.
2. In **Details** tab,
 1. Fill **Instance Name**, for example **Ubuntu 1**.
 2. Select **Flavor**, for example **m1.medium**.
 3. Fill **Instance Count** with **1**.
 4. Select **Instance Boot Source** with **Boot from Image**.
 5. Select **Image Name** with **Ubuntu 14.04 amd64 (243.7 MB)** if you want install Ubuntu 14.04 in your virtual machine.
3. In **Access & Security** tab,
 1. Click **[+]** button of **Key Pair** to import key pair. This key pair is a public and private key that we will use to connect to the instance from our machine.
 2. In **Import Key Pair** dialog,
 1. Fill **Key Pair Name** with your machine name (for example **Edward-Key**).
 2. Fill **Public Key** with your **SSH public key** (usually is in **~/.ssh/id_rsa.pub**). See description in **Import Key Pair** dialog box for more information. If you are using Windows, you can use **Puttygen** to generate key pair.
 3. Click **Import key pair**.
 3. In **Security Groups**, mark/check **default**.
4. In **Networking** tab,
 1. In **Selected Networks**, select network that have been created in Step 1, for example **internal**.
5. Click **Launch**.
6. If you want to create multiple instances, you can repeat step 1-5. I created one more instance with instance name **Ubuntu 2**.

Step 3: Create Router

I guess you already know what router is. In the step 1, we created our network, but it is isolated. It doesn't connect to the internet. To make our network has an internet connection, we need a router that running as the gateway to the internet.

1. Go to **Network > Routers** and then click **Create Router**.
2. Fill **Router Name** for example **router1** and then click **Create router**.
3. Click on your **router name link**, for example **router1**, **Router Details** page.
4. Click **Set Gateway** button in upper right:
 1. Select **External networks** with **external**.
 2. Then **OK**.
5. Click **Add Interface** button.
 1. Select **Subnet** with the network that you have been created in Step 1.
 2. Click **Add interface**.

6. Go to **Network > Network Topology**. You will see the network topology. In the example, there are two network, i.e. external and internal, those are bridged by a router. There are instances those are joined to internal network.

Step 4: Configure Floating IP Address

Floating IP address is public IP address. It makes your instance is accessible from the internet. When you launch your instance, the instance will have a private network IP, but no public IP. In OpenStack, the public IPs is collected in a pool and managed by admin (in our case is TryStack). You need to request a public (floating) IP address to be assigned to your instance.

1. Go to **Compute > Instance**.
2. In one of your instances, click **More > Associate Floating IP**.
3. In **IP Address**, click Plus [+].
4. Select **Pool** to **external** and then click **Allocate IP**.
5. Click **Associate**.
6. Now you will get a public IP, e.g. 8.21.28.120, for your instance.

Step 5: Configure Access & Security

OpenStack has a feature like a firewall. It can whitelist/blacklist your in/out connection. It is called *Security Group*.

1. Go to **Compute > Access & Security** and then open **Security Groups** tab.
2. In **default** row, click **Manage Rules**.
3. Click **Add Rule**, choose **ALL ICMP** rule to enable ping into your instance, and then click **Add**.
4. Click **Add Rule**, choose **HTTP** rule to open HTTP port (port 80), and then click **Add**.
5. Click **Add Rule**, choose **SSH** rule to open SSH port (port 22), and then click **Add**.
6. You can open other ports by creating new rules.

Step 6: SSH to Your Instance

Now, you can SSH your instances to the floating IP address that you got in the step 4. If you are using Ubuntu image, the SSH user will be ubuntu.

Assignment No. – 6

Design and deploy a web application in a PaaS environment.

What AWS Amplify is?

AWS Amplify is a set of purpose-built tools and features that lets frontend web and mobile developers quickly and easily build full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services as your use cases evolve. With Amplify, you can configure a web or mobile app backend, connect your app in minutes, visually build a web frontend UI, and easily manage app content outside the AWS console. Ship faster and scale effortlessly—with no cloud expertise needed.

Steps :

1. Login to the AWS console
2. Find for AWS Amplify in the services.
3. Get Started with Amplify service.
4. Click on Host a Web App.
5. Then choose to launch it with Github and authenticate your GitHub account for the same..
6. After that choose the Repository containing your source code (subfolder if needed)
7. Then Launch the application with the default configurations provided by [AWS Amplify](#)
8. Configurations may be different on type of framework / technology you are launching your application

Assignment No. – 7

Design and develop custom Application (Mini Project) using Salesforce Cloud

Salesforce Org

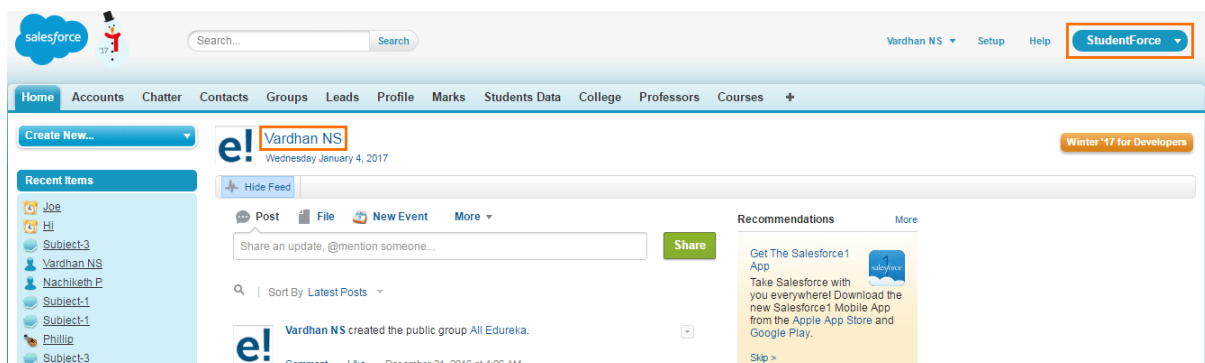
The cloud computing space offered to you or your organization by Force.com is called Salesforce org. It is also called Salesforce environment. Developers can create custom Salesforce Apps, objects, workflows, data sharing rules, Visualforce pages and Apex coding on top of Salesforce Org. Get your [Salesforce sales cloud certification](#) today to become certified.

Let us now deep dive into Salesforce Apps and understand how it functions.

Salesforce Apps

The primary function of a Salesforce app is to manage customer data. Salesforce apps provide a simple UI to access customer records stored in objects (tables). Apps also help in establishing relationship between objects by linking fields.

Apps contain a set of related tabs and objects which are visible to the end user. The below screenshot shows, how the *StudentForce* app looks like.

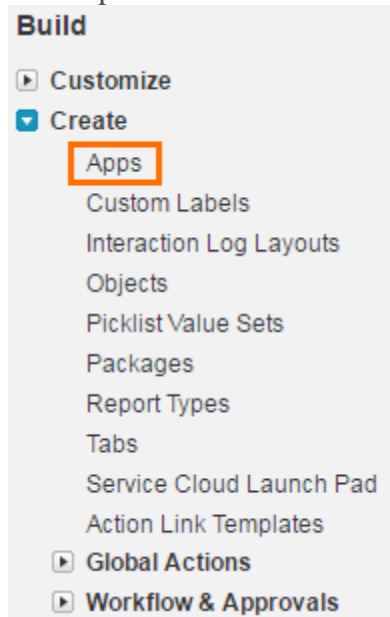


The highlighted portion in the top right corner of the screenshot displays the app name: *StudentForce*. The text highlighted next to the profile pic is my username: *Vardhan NS*.

Before you create an object and enter records, you need to set up the skeleton of the app. You can follow the below instructions to set up the app.

Steps To Setup The App

1. Click on *Setup* button next to app name in top right corner.
2. In the bar which is on the left side, go to *Build* → select *Create* → select *Apps* from the drop down menu.



3. Click on *New* as shown in the below screenshot.

Apps Help for this Page ?

An app is a group of tabs that work as a unit to provide functionality. Users can switch between apps using the Force.com app drop-down menu at the top-right corner of every page. You can customize existing apps to match the way you work, or build new apps by grouping standard and custom tabs.

Custom apps work in conjunction with User Profile Tab Visibility settings. [View User Profiles now.](#)

Action	App Label	Console	Custom	Description
Edit	App Launcher	<input type="checkbox"/>	<input type="checkbox"/>	App Launcher tabs
Edit	Call Center	<input type="checkbox"/>	<input type="checkbox"/>	State-of-the-Art On-Demand Customer Service
Edit	Community	<input type="checkbox"/>	<input type="checkbox"/>	Salesforce CRM Communities

4. Choose *Custom App*.

5. Enter the *App Label*. *StudentForce* is the label of my app. Click on *Next*.
New Custom App

[Help for this Page](#)

Step 2. Enter the Details Step 2 of 5

Fill in the fields below to define the custom app.

Custom App Information Required Information

App Label Example: HRforce, Financeforce, Bugforce

App Name

Description

[Previous](#) [Next](#) [Cancel](#)

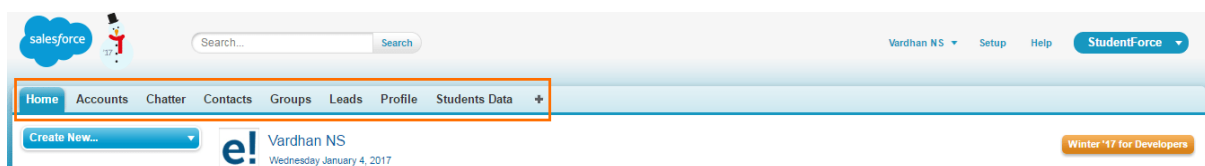
6. Choose a profile picture for your app. Click *Next*.
7. Choose the tabs you deem necessary. Click *Next*.
8. Select the different profiles you want the *app* to be assigned to. Click *Save*.

In steps 7 and 8, you were asked to choose the relevant tabs and profiles. Tabs and profiles are an integral part of Salesforce Apps because they help you to manage objects and records in Salesforce.

In this salesforce tutorial, I will give you a detailed explanation of Tabs, Profiles and then show you how to create objects and add records to it.

Salesforce Tabs

Tabs are used to access objects (tables) in the Salesforce App. They appear on top of the screen and are similar to a toolbar. It contains shortcut links to multiple objects. On clicking the object name in a tab, records in that object will be displayed. Tabs also contain links to external web content, custom pages and other URLs. The highlighted portion in the below screenshot is that of Salesforce tabs.



All applications will have a *Home* tab by default. Standard tabs can be chosen by clicking on '+' in the Tab menu. *Accounts*, *Contacts*, *Groups*, *Leads*, *Profile* are the standard tabs offered by Salesforce. For example, *Accounts* tab will show you the list of accounts in the SFDC org and *Contacts* tab will show you the list of contacts in the SFDC org.

Steps To Add Tabs

1. Click on '+' in the tab menu.
2. Click on *Customize tabs*, which is present on the right side.

3. Choose the tabs of your choice and click on *Save*.

Customize My Tabs

Choose the tabs that will display in each of your apps.

Custom App:
StudentForce ▼

Available Tabs		Selected Tabs
App Launcher	Add ▶ Remove ◀	Home (default)
Assets		Accounts
Campaigns		Chatter
Cases		Contacts
Console		Groups
Content		Leads
Contracts		Profile
Contribute		Students
D&B Companies		
Dashboards		
Data.com		

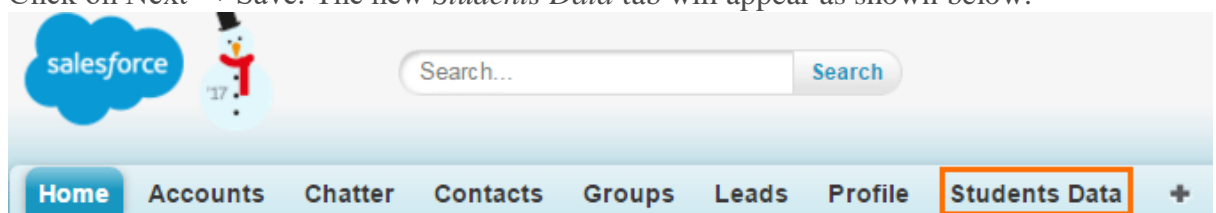
Up
▲
▼
Down

Save Cancel

Besides standard tabs, you can also create custom tabs. *Students* tab that you see in the above screenshot is a custom tab that I have created. This is a shortcut to reach the custom object: *Students*.

Steps To Create Custom Tabs

1. Navigate to Setup → Build → Create → Tabs.
2. Click on *New*.
3. Select the object name for which you are creating a tab. In my case, it is *Students Data*. This is a custom object which I have created (the instructions to create this object is covered later in this blog).
4. Choose a tab style of your preference and enter a description.
5. Click on Next → Save. The new *Students Data* tab will appear as shown below.



Salesforce Profiles

Every user who needs to access the data or SFDC org will be linked to a profile. A profile is a collection of settings and permissions which controls what a user can view, access and modify in Salesforce.

A profile controls user permissions, object permissions, field permissions, app settings, tab settings, apex class access, Visualforce page access, page layouts, record types, login hour and login IP addresses.

You can define profiles based on the background of the user. For example, different levels of access can be set for different users like system administrator, developer and sales representative.

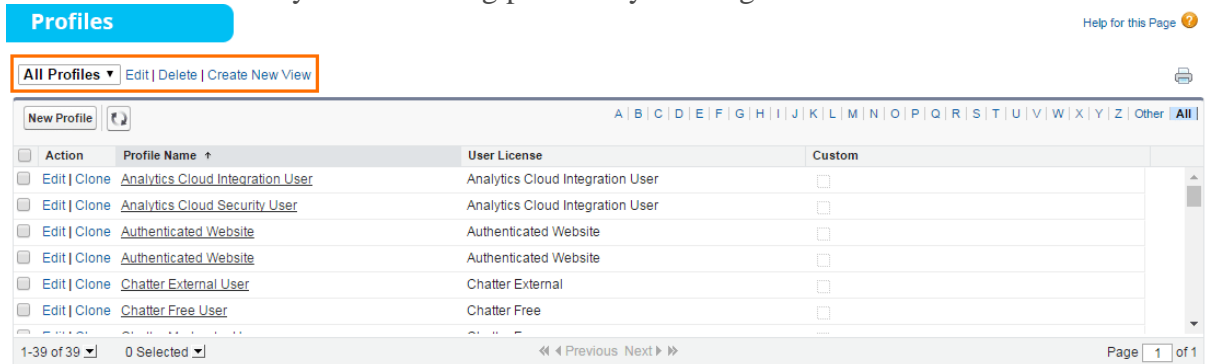
Similar to tabs, we can use any standard profile or create a custom profile. By default, the available standard profiles are: read only, standard user, marketing user, contract manager, solution manager and system administrator. If you want to create custom profiles, you have to first clone standard profiles and then edit that profile. Do note that one profile can be assigned to many users, but one user cannot be assigned many profiles.



Steps To Create A Profile

1. Click on Setup → Administer → Manage users → Profiles

- You can then clone any of the existing profiles by clicking on *Edit*.



Once the tabs and profiles are set up for your App, you can load data into it. The next section of this Salesforce tutorial will thus cover how data is added to objects in the form of records and fields.

Enroll in [Salesforce training](#) to fast forward your career.

Objects, Fields And Records In Salesforce

Objects, Fields and Records are the building blocks of Salesforce. So, it is important to know what they are and what role they play in building Apps.

Objects are the database tables in Salesforce where data is stored. There are two types of objects in Salesforce:

- Standard objects:** The objects provided by Salesforce are called standard objects. For example, Accounts, Contacts, Leads, Opportunities, Campaigns, Products, Reports, Dashboard etc.
- Custom objects:** The objects created by users are called custom objects.

Objects are a collection of records and records are a collection of fields.

Every row in an object consists of many fields. Thus a record in an object is a combination of related fields. Look at the below excel for illustration.

	A	B	C	D	E	F
1	Student Name	Email ID	Phone Number	Country	City	Department
2	student1	student2@gmail.	1234354657	India	Bangalore	Science
3	student2					
4	student3					
5	student4					
6	student5					
7	student6					

Diagram illustrating the relationship between Object, Field, and Record:

- Object:** The entire table (all rows and columns).
- Field:** A single column (e.g., Student Name, Email ID).
- Record:** A single row (e.g., student1, student2@gmail., 1234354657, India, Bangalore, Science).

I will create an object called *Students Data* which will contain personal details of students.

Steps to create a custom object:

1. Navigate to Setup → Build → Create → Object
2. Click on *New Custom Object*.
3. Fill in the *Object Name* and *Description*. As you can see from the below image, the object name is *Students Data*.

Custom Object Definition Edit [Save] [Save & New] [Cancel]

Custom Object Information ⓘ = Required Information

The singular and plural labels are used in tabs, page layouts, and reports.

Label Example: Account

Plural Label Example: Accounts

Starts with vowel sound ☐

The Object Name is used when referencing the object via the API.

Object Name Example: Account

Description

Context-Sensitive Help Setting ☒ Open the standard Salesforce.com Help & Training window ☐ Open a window using a Visualforce page

4. Click on *Save*.

If you want to add this custom object to the tab menu, then you can follow the instructions mentioned earlier in this Salesforce tutorial blog.

After creating the object, you need to define various fields in that object. e.g. the fields in a student's record will be student name, student phone number, student email ID, the department a student belongs to and his native city.

You can add records to objects only after defining the fields.

Steps To Add Custom Fields

1. Navigate to Setup → Build → Create → Objects
2. Select the object to which you want to add fields. In my case, it is *Students Data*.
3. Scroll down to Custom Fields & Relationships for that object and click on *New* as shown in the below screenshot.

Custom Fields & Relationships [New] [Field Dependencies] Custom Fields & Relationships Help ⓘ

No custom fields defined

4. You need to choose the data type of that particular field and then click *Next*. I have chosen *text* format because I will be storing letters in this field.
The different data types of fields have been explained in detail in the next section of this blog.
5. You will then be prompted to enter the name of the field, maximum length of that field and description.
6. You can also make it an optional/ mandatory field and allow/ disallow duplicate values for different records by checking on the check boxes. See the below screenshot

to get a better understanding.

Step 2. Enter the details Step 2 of 4

[Previous](#) [Next](#) [Cancel](#)

Field Label: ⓘ

Please enter the maximum length for a text field below.

Length:

Field Name: ⓘ

Description:

Help Text: ⓘ

Required: ☒ Always require a value in this field in order to save a record

Unique: ☐ Do not allow duplicate values

☒ Treat "ABC" and "abc" as duplicate values (case insensitive)
☐ Treat "ABC" and "abc" as different values (case sensitive)

External ID: ☐ Set this field as the unique record identifier from an external system

Default Value: [Show Formula Editor](#)

Use formula syntax: e.g., Text in double quotes: "hello", Number: 25, Percent as decimal: 0.10, Date expression: Today()
+ 7

[Previous](#) [Next](#) [Cancel](#)

7. Click on *Next*.
8. Select the various profiles who can edit that text field at a later point of time.
Click *Next*.
9. Select the page layouts that should include this field.
10. Click *Save*.

As you can see from the below screenshot, there are two types of fields. Standard fields created for every object by default and Custom fields created by myself. The four fields which I have created for *Students Data* are City, Department, Email ID and Phone No. You will notice that all custom fields are suffixed with ‘__C’ which indicates that you have the power to edit and delete those fields. Whereas some standard fields can be edited, but not deleted.

Standard Fields						Standard Fields Help ?
Action	Field Label	Field Name	Data Type	Controlling Field	Indexed	
	Created By	CreatedBy	Lookup(User)			
	Last Modified By	LastModifiedBy	Lookup(User)			
Edit	Owner	Owner	Lookup(User,Queue)			✓
Edit	Student Name	Name	Text(80)			✓

Custom Fields & Relationships							Custom Fields & Relationships Help ?
		New Field Dependencies					
Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By	
Edit Del	City	City__c	Text(20)			Vardhan NS , 12/27/2016 6:42 AM	
Edit Del	Department	Department__c	Text(20)			Vardhan NS , 12/27/2016 9:36 PM	
Edit Del	Email ID	Email_ID__c	Email (Unique)	✓		Vardhan NS , 12/27/2016 6:39 AM	
Edit Del	Phone No.	Phone_No__c	Phone			Vardhan NS , 12/27/2016 6:38 AM	

You can now add student records (complete row) to your object.

Steps To Add A Record

1. Go to the object table from the tab menu. *Students Data* is the object to which I will add records.

- As you can see from the below image, there are no existing records. Click on *New* to add new student records.

Students Home

View: All Go! Edit | Create New View

Recent Students [New](#) [Recently Viewed](#)

No recent records. Click Go or select a view from the dropdown to display records.

- Add student details into different fields as shown in the below screenshot. Click on *Save*.

Student Data Edit New Student Data

Student Data Edit [Save](#) [Save & New](#) [Cancel](#)

Information ! = Required Information

Student Name Owner Vardhan NS

Department

Phone No.

Email ID

City

[Save](#) [Save & New](#) [Cancel](#)

- You can create any number of student records. I have created 4 student records as shown in the below screenshot.

salesforce Search... Vardhan NS Setup Help StudentForce

Home Accounts Chatter Contacts Groups Leads Profile Marks **Students Data** College Professors Courses +

Create New... [All](#) Edit | Delete | Create New View

[New Student Data](#) [Change Owner](#) [A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#) Other [All](#)

Action	Student Name	Country	Email ID	College
Edit Del	Joe	USA	joe1@gmail.com	
Edit Del	Kate	France	kate1@gmail.com	Cambridge
Edit Del	Mike	USA	mike1@gmail.com	Harvard
Edit Del	Sam	UK	sam1@gmail.com	Oxford

- In case you want to edit the student details, you can click on *Edit* as shown in the below screenshot.

salesforce Search... Vardhan NS Setup Help StudentForce

Home Accounts Chatter Contacts Groups Leads Profile Marks **Students Data** College Professors Courses +

Create New... [All](#) Edit | Delete | Create New View

Student Data [Joe](#)

[Back to List: Students Data](#) [Marks \(0\)](#) | [Courses \(1\)](#)

Student Data Detail [Edit](#) [Delete](#) [Clone](#)

Student Name Joe Owner [e! Vardhan NS \[Change\]](#)

Department Medicine

Phone No. (457) 857-8676

Email ID joe1@gmail.com

City New York

Country USA

College

Created By [Vardhan NS](#) 12/29/2016 3:11 AM Last Modified By [Vardhan NS](#) 12/29/2016 3:11 AM

[Edit](#) [Delete](#) [Clone](#)


[Recycle Bin](#)

Data Types Of Fields

Data type controls which type of data can be stored in a field. Fields within a record can have different data types. For example:

- If it is a phone number field, you can choose *Phone*.
- If it is a name or a text field, you can choose *Text*.
- If it is a date/ time field, you can choose *Date/Time*.
- By choosing *Picklist* as data type for a field, you can write predefined values in that field and create a drop-down.

You can choose any one of the data types for custom fields. Below is a screenshot listing the different data types.

<input type="radio"/> Auto Number	A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.
<input type="radio"/> Formula	A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.
<input type="radio"/> Roll-Up Summary 	A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.
<input type="radio"/> Lookup Relationship	Creates a relationship that links this object to another object. The relationship field allows users to click on a lookup icon to select a value from a popup list. The other object is the source of the values in the list.
<input type="radio"/> Master-Detail Relationship	Creates a special type of parent-child relationship between this object (the child, or "detail") and another object (the parent, or "master") where: <ul style="list-style-type: none"> • The relationship field is required on all detail records. • The ownership and sharing of a detail record are determined by the master record. • When a user deletes the master record, all detail records are deleted. • You can create rollup summary fields on the master record to summarize the detail records. The relationship field allows users to click on a lookup icon to select a value from a popup list. The master object is the source of the values in the list.
<input type="radio"/> External Lookup Relationship	Creates a relationship that links this object to an external object whose data is stored outside the Salesforce org.
<input type="radio"/> Checkbox	Allows users to select a True (checked) or False (unchecked) value.
<input type="radio"/> Currency	Allows users to enter a dollar or other currency amount and automatically formats the field as a currency amount. This can be useful if you export data to Excel or another spreadsheet.
<input type="radio"/> Date	Allows users to enter a date or pick a date from a popup calendar.
<input type="radio"/> Date/Time	Allows users to enter a date and time, or pick a date from a popup calendar. When users click a date in the popup, that date and the current time are entered into the Date/Time field.
<input type="radio"/> Email	Allows users to enter an email address, which is validated to ensure proper format. If this field is specified for a contact or lead, users can choose the address when clicking Send an Email. Note that custom email addresses cannot be used for mass emails.
<input type="radio"/> Geolocation	Allows users to define locations. Includes latitude and longitude components, and can be used to calculate distance.
<input type="radio"/> Number	Allows users to enter any number. Leading zeros are removed.
<input type="radio"/> Percent	Allows users to enter a percentage number, for example, "10" and automatically adds the percent sign to the number.
<input type="radio"/> Phone	Allows users to enter any phone number. Automatically formats it as a phone number.
<input type="radio"/> Picklist	Allows users to select a value from a list you define.
<input type="radio"/> Picklist (Multi-Select)	Allows users to select multiple values from a list you define.

Data types like *Lookup Relationship*, *Master-Detail Relationship* and *External Lookup Relationship* are used to create links/ relationships between one or more objects. Relationships between objects is the next topic of discussion in this Salesforce tutorial blog.

Assignment No. – 8

Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store

Before you Begin-

1. Install [Git](#), [Python 2.7](#), and [virtualenv](#). For more information on setting up your Python development environment, such as installing the latest version of Python, refer to [Setting Up a Python Development Environment](#) for Google Cloud.
2. If you're new to Google Cloud, [create an account](#) to evaluate how our products perform in real-world scenarios. New customers also get \$300 in free credits to run, test, and deploy workloads.

3. In the Google Cloud Console, on the project selector page, select or [create a Google Cloud project](#).

Note: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

[Go to project selector](#)

4. [Install](#) and [initialize](#) the Google Cloud CLI.

If you have already installed and initialized the SDK to a different project, set the gcloud project to the App Engine project ID you're using for Firenotes. See [Managing Google Cloud SDK Configurations](#) for specific commands to update a project with the gcloud tool.

Cloning the sample app

To download the sample to your local machine:

1. Clone the sample application repository to your local machine:

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git
```

Alternatively, you can [download the sample](#) as a zip file and extract it.

2. Navigate to the directory that contains the sample code:

```
cd python-docs-samples/appengine/standard/firebase/firenotes
```

To configure FirebaseUI and enable identity providers:

3. Add Firebase to your app by following these steps:
 - a. Create a Firebase project in the [Firebase console](#).
 - If you don't have an existing Firebase project, click **Add project** and enter either an existing Google Cloud project name or a new project name.
 - If you have an existing Firebase project that you'd like to use, select that project from the console.
 - b. From the project overview page, click **Add Firebase to your web app**. If your project already has an app, select **Add App** from the project overview page.
 - c. Use the Initialize Firebase section of your project's customized code snippet to fill out the following section of the frontend/main.js file:

```
appengine/standard/firebase/firenotes/frontend/main.js
```

[View on GitHub](#)

```
// Obtain the following from the "Add Firebase to your web
app" dialogue
// Initialize Firebase
var config = {
  apiKey: "<API_KEY>",
  authDomain: "<PROJECT_ID>.firebaseapp.com",
  databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
  projectId: "<PROJECT_ID>",
  storageBucket: "<BUCKET>.appspot.com",
  messagingSenderId: "<MESSAGING_SENDER_ID>"
};
```

1. In the frontend/main.js file, configure the [FirebaseUI login widget](#) by selecting which providers you want to offer your users.

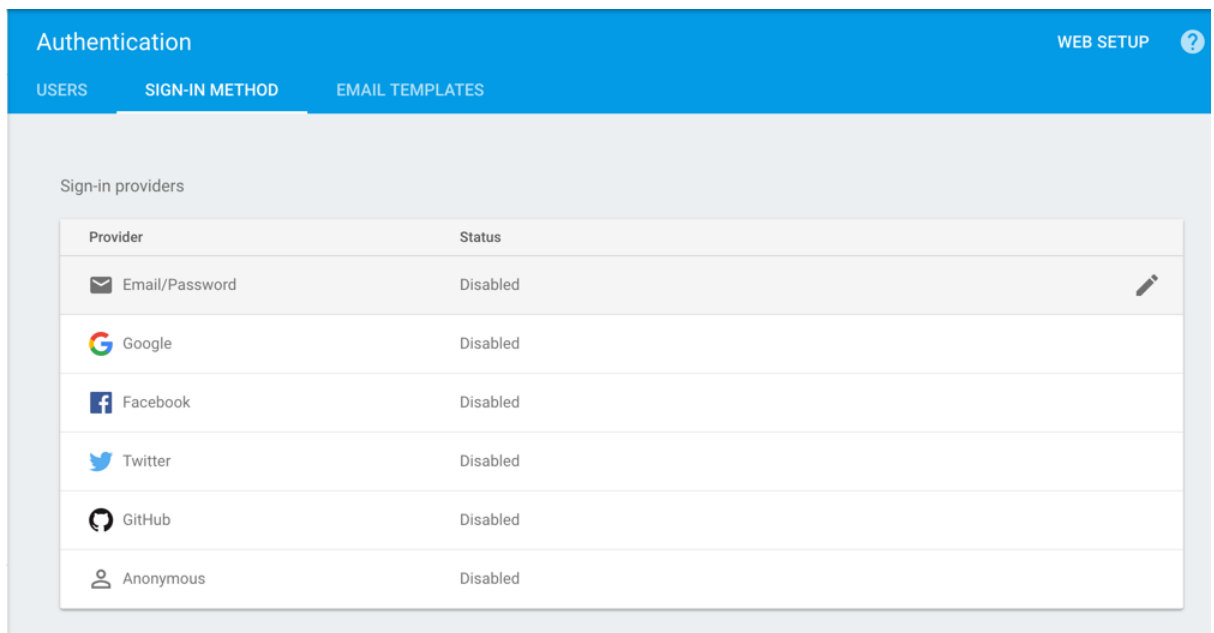
[appengine/standard/firebase/firenotes/frontend/main.js](#)

[View on GitHub](#)

```
// Firebase log-in widget
function configureFirebaseLoginWidget() {
  var uiConfig = {
    'signInSuccessUrl': '/',
    'signInOptions': [
      // Leave the lines as is for the providers you want to
      offer your users.
      firebase.auth.GoogleAuthProvider.PROVIDER_ID,
      firebase.auth.FacebookAuthProvider.PROVIDER_ID,
      firebase.auth.TwitterAuthProvider.PROVIDER_ID,
      firebase.auth.GithubAuthProvider.PROVIDER_ID,
      firebase.auth.EmailAuthProvider.PROVIDER_ID
    ],
    // Terms of service url
    'tosUrl': '<your-tos-url>',
  };

  var ui = new firebaseui.auth.AuthUI(firebase.auth());
  ui.start('#firebaseui-auth-container', uiConfig);
}
```

2. Enable the providers you have chosen to keep in the [Firebase console](#) by clicking **Authentication** > **Sign-in method**. Then, under **Sign-in providers**, hover the cursor over a provider and click the pencil icon.



1.

- a. Toggle the **Enable** button and, for third-party identity providers, enter the provider ID and secret from the provider's developer site. The Firebase docs give specific instructions in the "Before you begin" sections of the [Facebook](#), [Twitter](#), and [GitHub](#) guides. After enabling a provider, click **Save**.

Facebook

Enable ☒

App ID

App secret

To complete set up, add this OAuth redirect URI to your Facebook app configuration. [Learn more](#)

`https://test-84e93.firebaseio.com/_/auth/handler`

CANCEL SAVE

- b. In the Firebase console, under **Authorized Domains**, click **Add Domain** and enter the domain of your app on App Engine in the following format:

`[PROJECT_ID].appspot.com`

Do not include `http://` before the domain name.

Installing dependencies

1. Navigate to the backend directory and complete the application setup:

```
cd backend/
```

2. Install the dependencies into a lib directory in your project:

```
pip install -t lib -r requirements.txt
```

3. In `appengine_config.py`, the `vendor.add()` method registers the libraries in the lib directory.