List of Assignments

TITLE
Group A
1. Install Google App Engine. Create hello world app and other simple web applications using python/java. **2.** Use GAE launcher to launch the web applications. **3.** Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim. **4.** Find a procedure to transfer the filesfrom one virtual machine to another virtual machine. **5.** Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version) **6.** Design and deploy a web application in a PaaS environment. **7.** Design and develop custom Application (Mini Project) using Salesforce Cloud. **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.
CASE STUDIES
• Data storage security in private cloud • Application of IoT/Ubiquitous based on cloud • Toolsfor building private cloud

Assignment No : 1

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

Objective:

To learn the cloud computing concepts and installing the GAE.

Outcome:

To implement the hello world in GAE.

Explanation:

Tools used :

1. JDK 1.6
2. Eclipse 3.7 + Google Plugin for Eclipse
3. Google App Engine Java SDK 1.6.3.1

1. Install Google Plugin for Eclipse

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

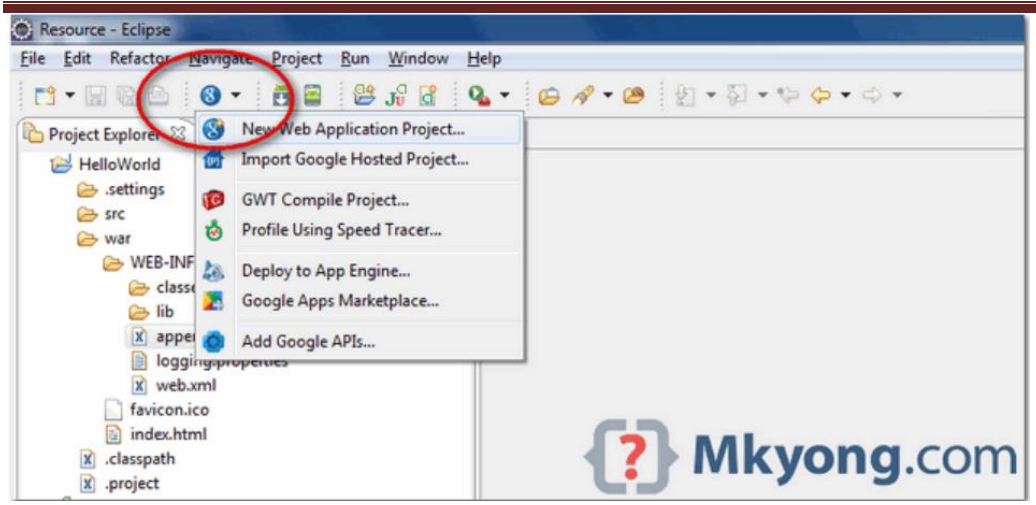
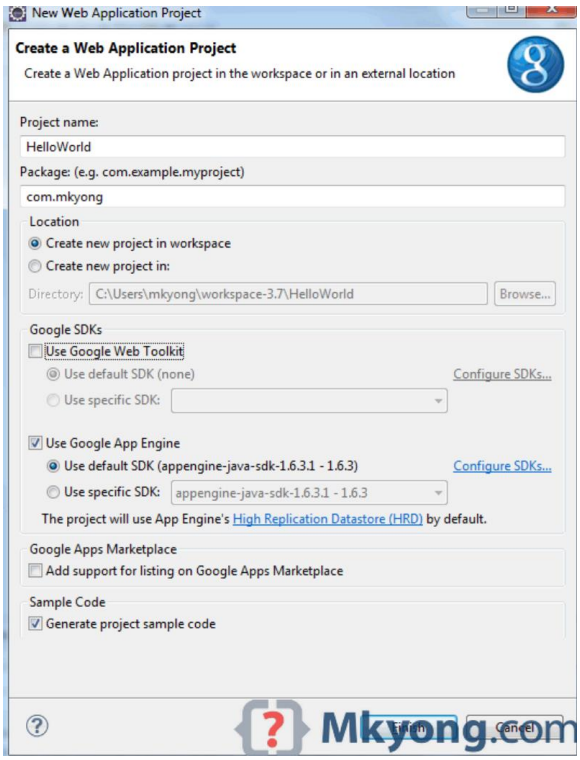


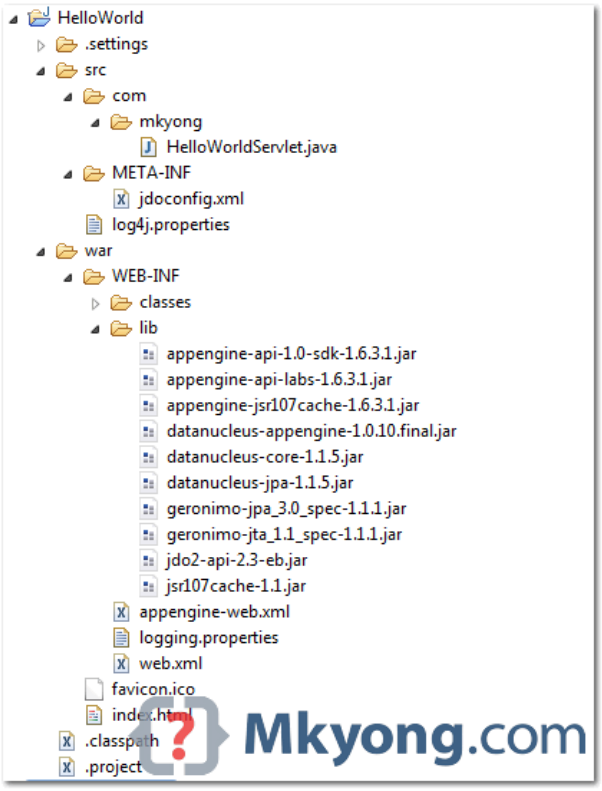
Figure – Deselect the "Google Web ToolKit", and link your GAE Java SDK via the "configure
SDK" link.



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

```
<?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>
```

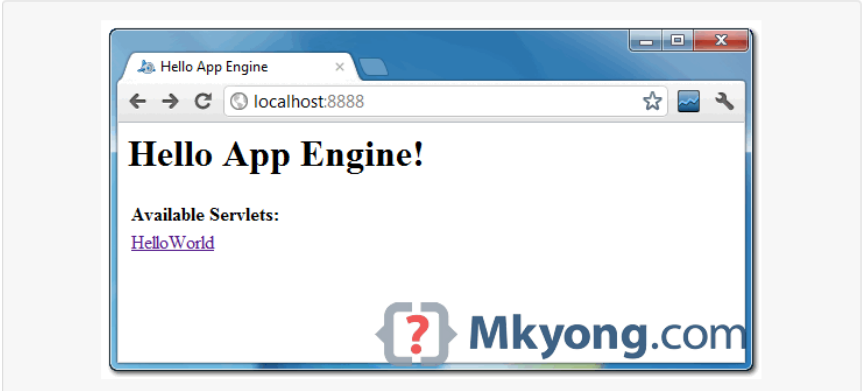
4. Run it local

Right click on the project and run as "**Web Application**".

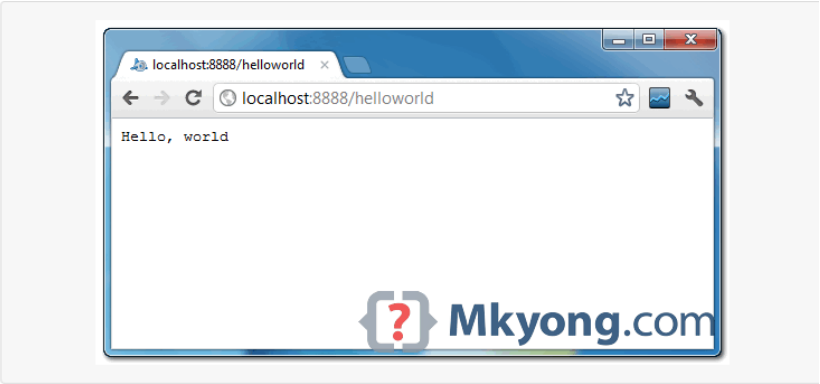
Eclipse console :

```
//...
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



Result:

Thus the simple application created successfully.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

Assignment No 2

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

Objective:

To learn the cloud computing concepts and installing the GAE.

Outcome:

To implement the hello world in GAE.

Explanation:

You can use Google App Engine to host a static website. Static web pages can contain client-side technologies such as HTML, CSS, and JavaScript. Hosting your static site on App Enginecan cost less than using a traditional hosting provider, as App Engine provides a free tier.

Sites hosted on App Engine are hosted on the REGION_ID.r.appspot.com subdomain, such as[my-project-id].uc.r.appspot.com. After you deploy your site, you can map your own domainname to your App Engine-hosted website.

Create a new Cloud Console project or retrieve the project ID of an existing project to use: Goto the Project page.

Install and then initialize the Google Cloud SDK:

Download the SDK

Creating a website to host on Google App Engine

Basic structure for the project

This guide uses the following structure for the project:

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.

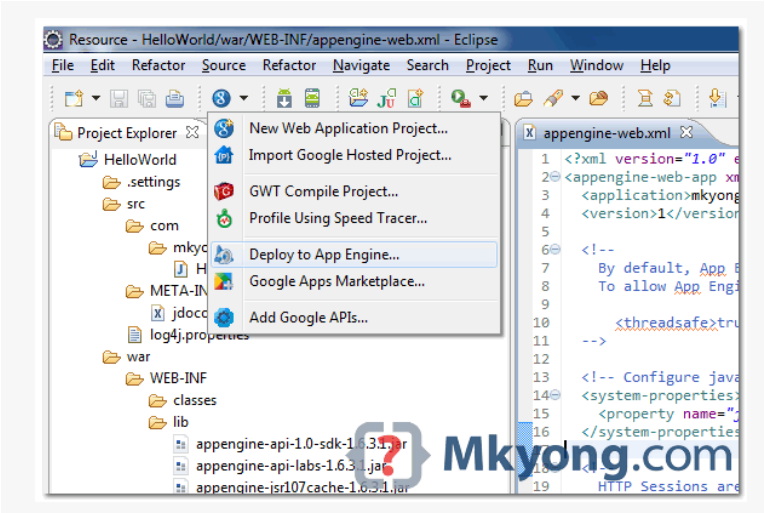


Figure 1.2 – Sign in with your Google account and click on the Deploy button.

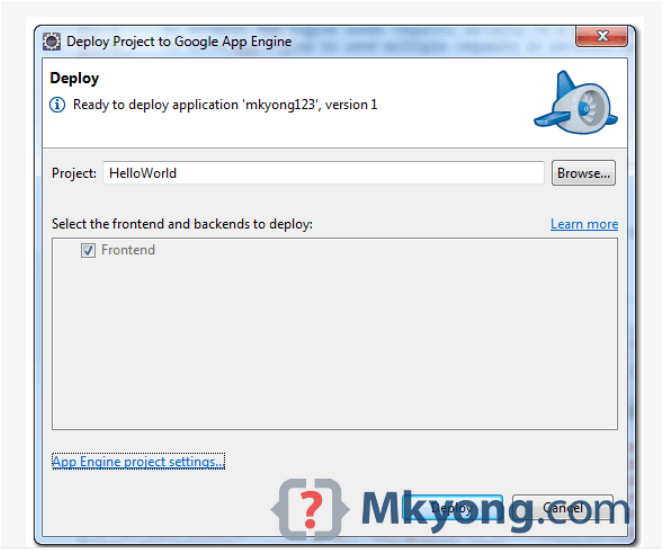


Figure 1.3 – If everything is fine, the hello world web application will be deployed to this URL – http://mkyong123.appspot.com/

Done.

Result:

Successfully web app deployed.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

Assignment No 3

AIM: Simulate a cloud scenario using cloudsim and run a scheduling algorithm that is not present in cloudsim steps.

Objective:

To learn the cloud computing concepts and cloud computing simulation using cloudsim.

Outcome:

To implement simulation using cloudsim.

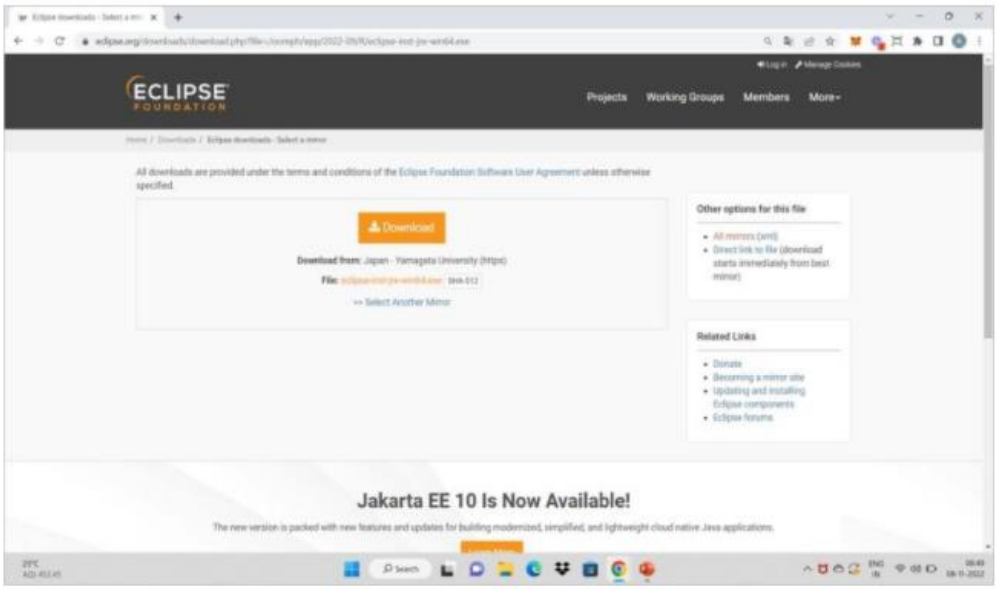
Explanation:

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 modeling 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.

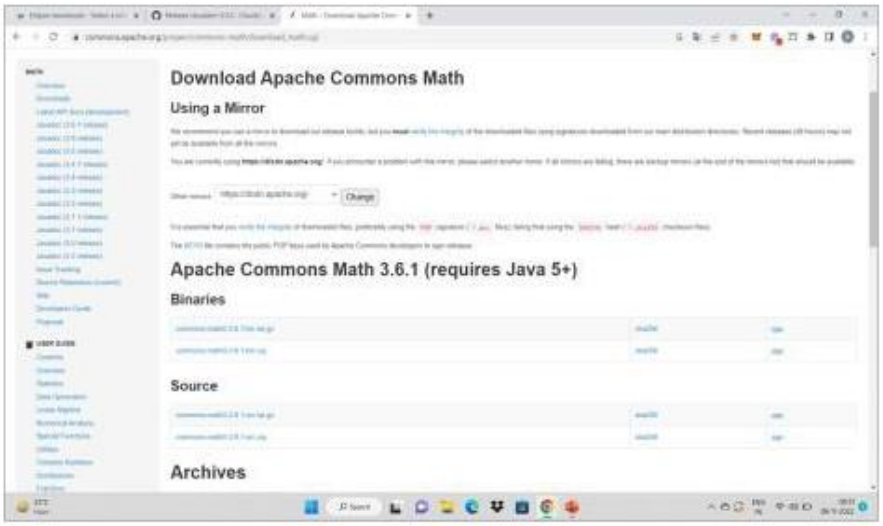
Step-by-Step Implementation

Step 1: Download Eclipse IDE for Java Developers. It is developed by the CLOUDS Lab organization and is written entirely in Java so we would need a Java IDE to simulate a cloud scenario using CloudSim.

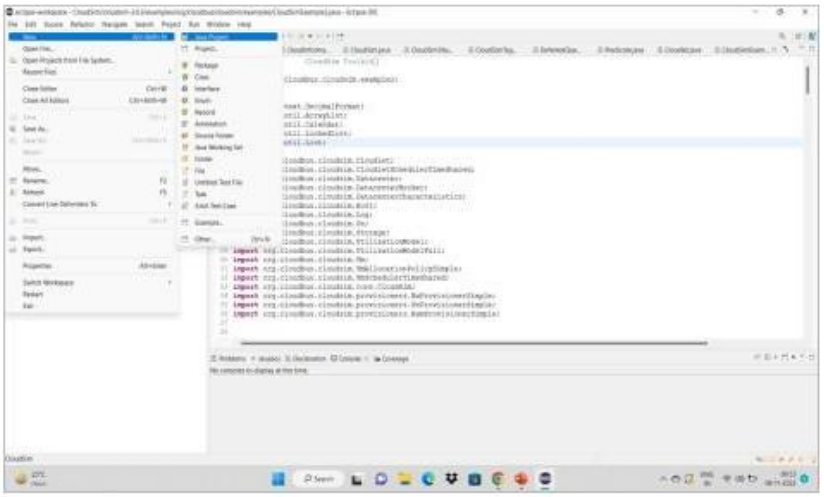


Step 2: Download the CloudSim 3.0.3 zip file from **GitHub** and extract the folders within a folder in our computer storage.

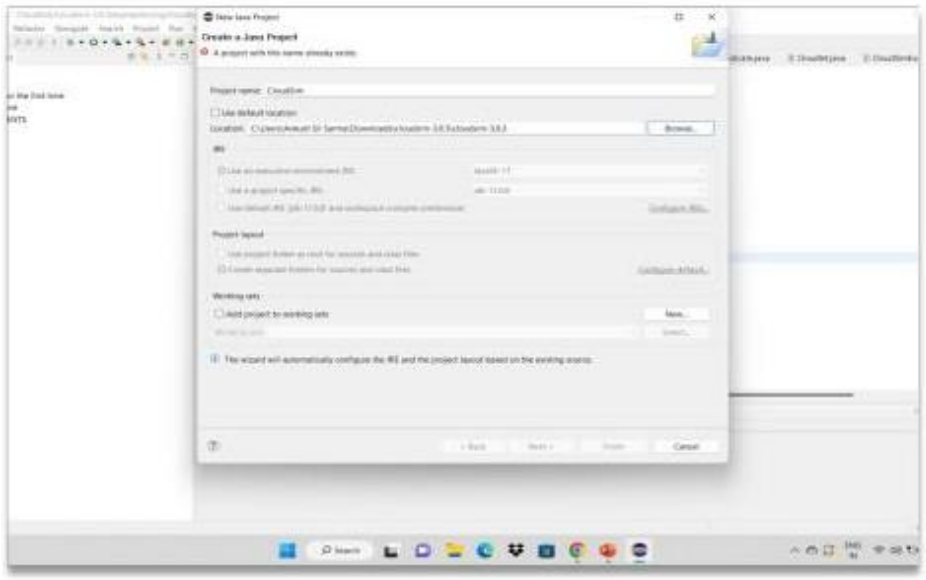
Step 3: Download Apache Commons Math 3.6.1 zip file. It is required as it provides a faster, more accurate, portable alternative to the regular Math and StrictMath classes for large-scale computation in java files.



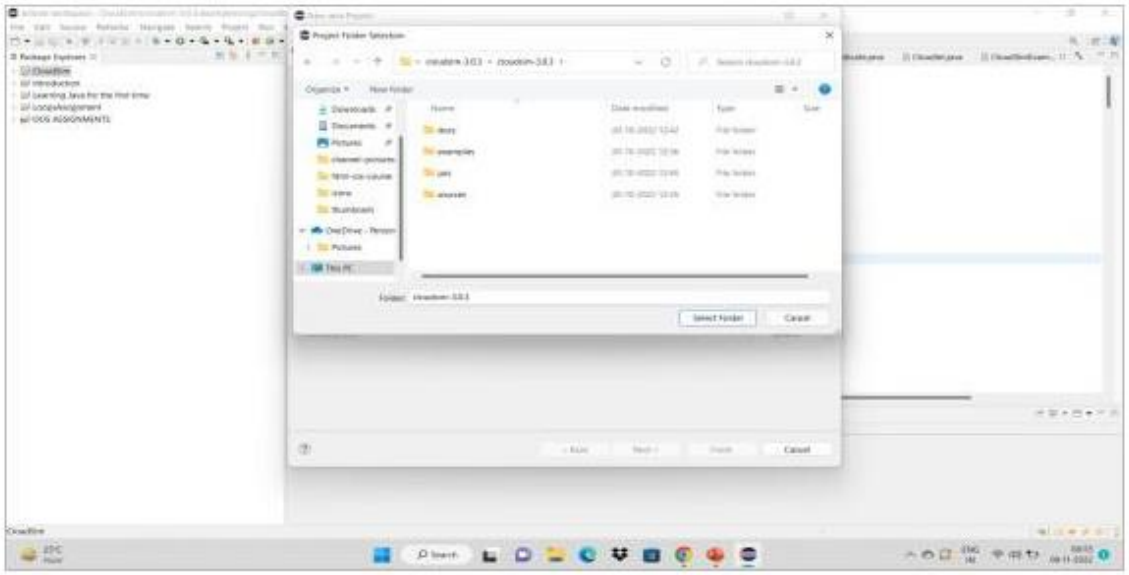
Step 4: Open Eclipse IDE and create a new Java Project.



Step 5: The above-mentioned Java Project should be created with the location of the previously downloaded and extracted CloudSim 3.0.3 folder.

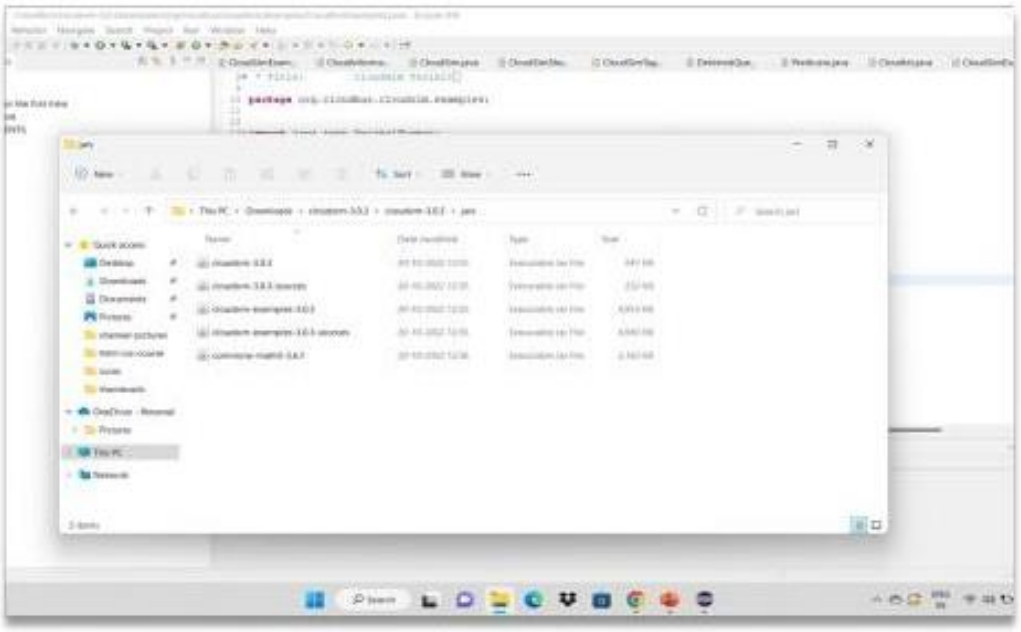


Select name of the project as mentioned and extract CloudSim folder



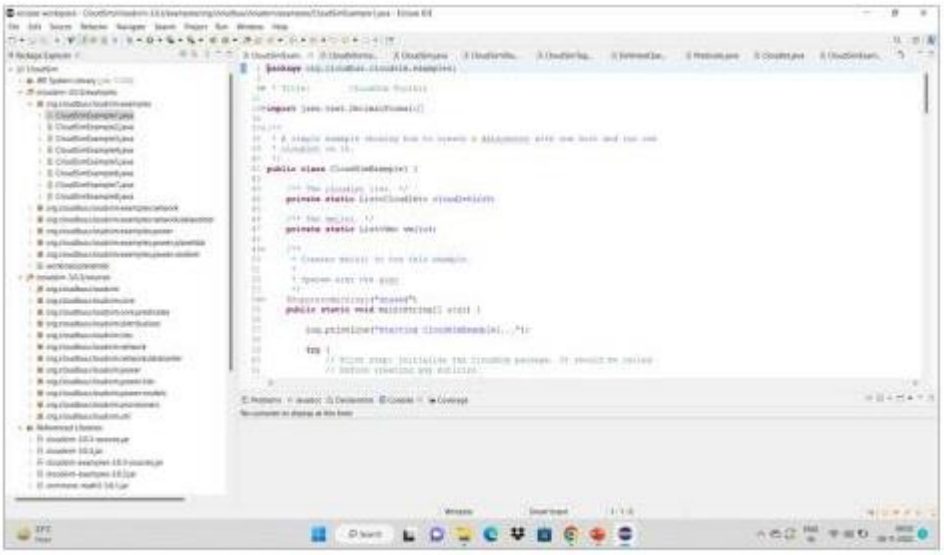
Selecting the CloudSim 3.0.3 folder

Step 6: The JAR file from the extracted Apache Commons Math 3.6.1 folder needs to be added to the JAR files of CloudSim.



Jar files of Apache Commons Math extracted to the Jar files of CloudSim

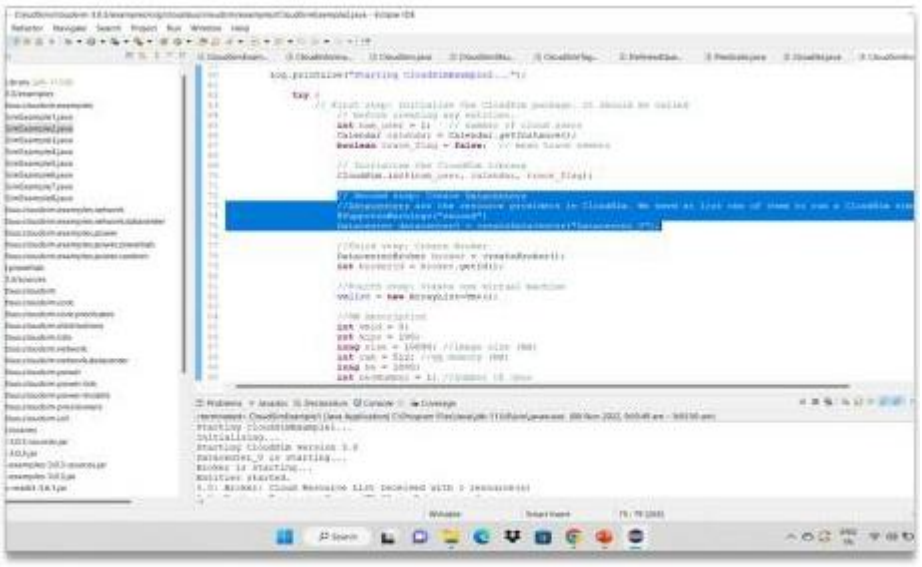
Step 7: Now the CloudSim Environment has been setup in the Eclipse IDE.



CloudSim Environment Has Been Setup

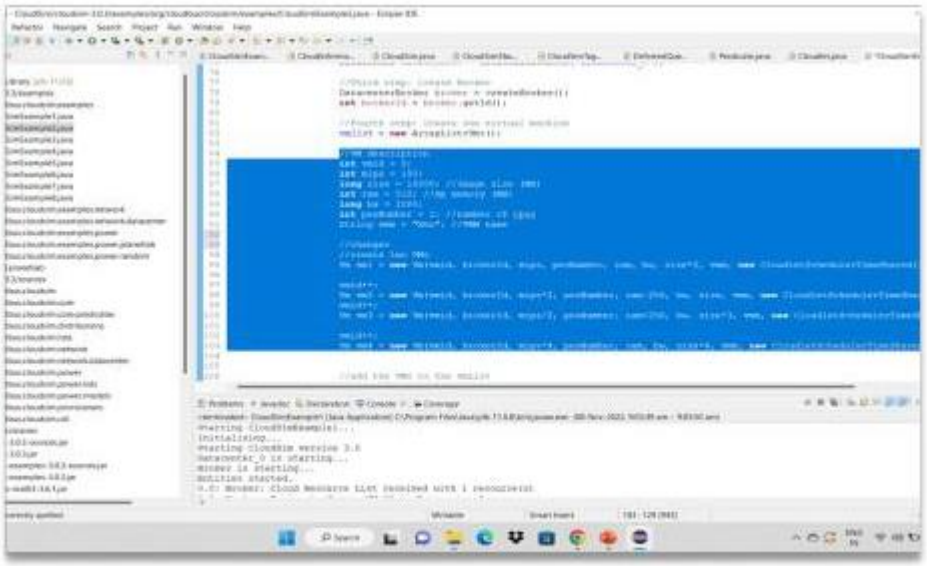
Now Creating Datacenters, Virtual Machines, And Cloudlets In The CloudSim Environment And Checking Its Output

Step 1: We open CloudSimExample2.java from the library on the left and create a data center first.



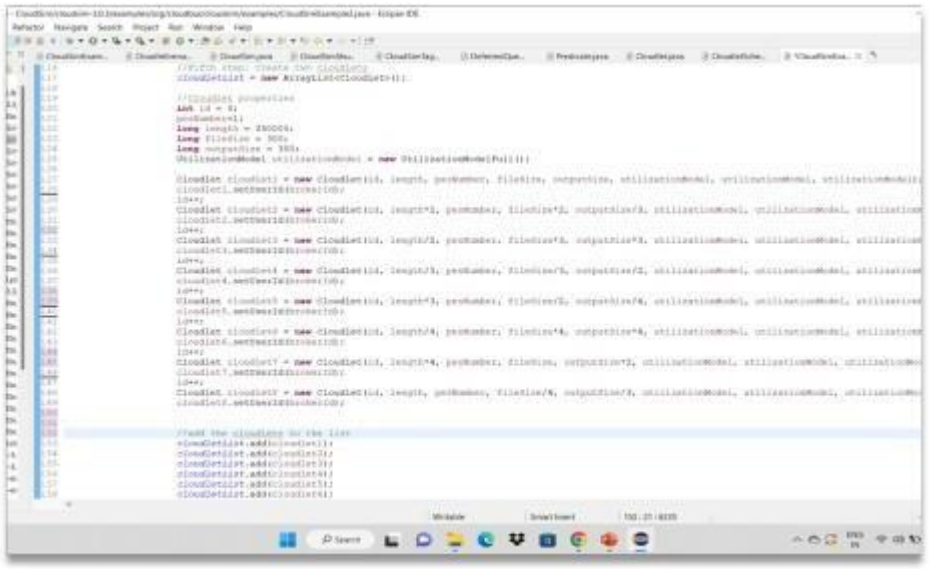
Creating a datacenter in CloudSimExample2.java

Step 2: The initial configuration of one of VM(Virtual machine) is done i.e mips(million instructions), ram(size of RAM), bw(bandwidth), etc. Here we create 4 VMs which are initialized with different configuration and added to vmlist, which is the array list created to store all the 4VMs.



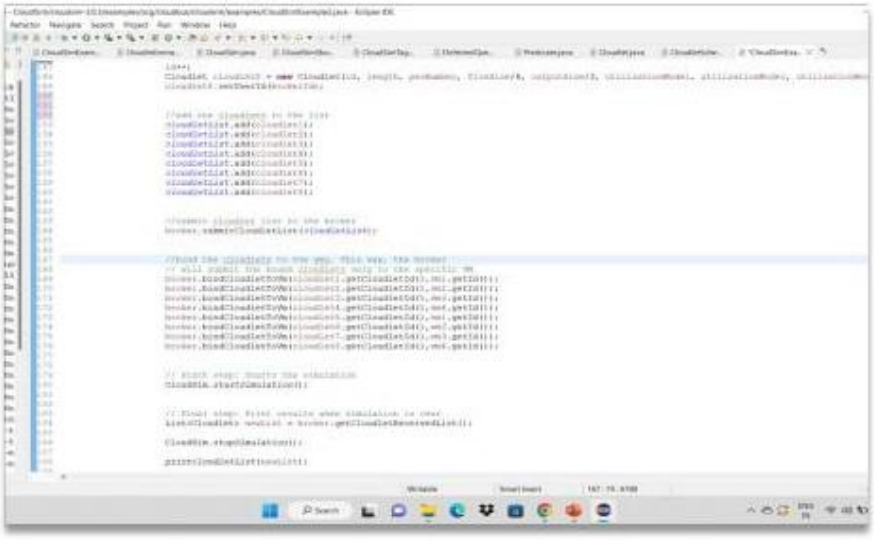
Configuring Virtual Machine

Step 3: We create 8 cloudlets that are initialized with different properties or characteristics i.e id length, outputsize, and filesize.



Creating Cloudlets With Different Characteristics

Step 4: Now the VM-cloudlet binding is done using the broker.



VM-Cloudlet Binding

Below is the code for the above implementation of steps:

***Note**: You have to run this Java program on your Eclipse IDE after extracting CloudSim 3.0.3 and Apache Commons Math 3.6.1 zip file. Only then will the code run properly.*

```
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

// A simple example showing how to create a data center
// with one host and run eight cloudlets on it
public class CloudSimExample1 {
	// The cloudlet list
	private static List<Cloudlet> cloudletList;

	// The vmlist
	private static List<Vm> vmlist;

	@SuppressWarnings("unused")
	public static void main(String[] args)
	{
		Log.printLine("Starting CloudSimExample2...");

		try {
			// First step: Initialize the CloudSim package.
			// It should be called before creating any
			// entities. number of cloud users
			int num_user = 1;
```

```
// Calendar whose fields have been initialized
// with the current date and time.
Calendar calendar = Calendar.getInstance();

// trace events
boolean trace_flag = false;

CloudSim.init(num_user, calendar, trace_flag);

// Second step: Create Datacenters
// Datacenters are the resource providers in
// CloudSim. We need at list one of them to run
// a CloudSim simulation
Datacenter datacenter0
        = createDatacenter("Datacenter_0");

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

// Fourth step: Create four virtual machine
vmlist = new ArrayList<Vm>();

// VM description
int vmid = 0;
int mips = 1000;
long size = 10000; // image size (MB)
int ram = 512; // vm memory (MB)
long bw = 1000; // bandwidth
int pesNumber = 1; // number of cpus
String vmm = "Xen"; // VMM name

// create 4 VMs
Vm vm1
        = new Vm(vmid, brokerId, mips, pesNumber,
                        ram, bw, size, vmm,
                        new CloudletSchedulerTimeShared());
vmid++;
Vm vm2 = new Vm(
        vmid, brokerId, mips * 2, pesNumber,
        ram - 256, bw, size * 2, vmm,
        new CloudletSchedulerTimeShared());
vmid++;
Vm vm3 = new Vm(
```

```
            vmid, brokerId, mips / 2, pesNumber,
            ram + 256, bw, size * 3, vmm,
            new CloudletSchedulerTimeShared());
vmid++;
Vm vm 4
            = new Vm(vmid, brokerId, mips * 4,
                                    pesNumber, ram, bw, size * 4, vmm,
                                    new CloudletSchedulerTimeShared());
vmid++;

// add the VM to the vmList
vmlist.add(vm1);
vmlist.add(vm2);
vmlist.add(vm3);
vmlist.add(vm4);

// submit vm list to the broker
broker.submitVmList(vmlist);

// Fifth step: Create eight Cloudlets
cloudletList = new ArrayList<Cloudlet>();

// Cloudlet properties
int id = 0;
long length = 400000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel
            = new UtilizationModelFull();

Cloudlet cloudlet1 = new Cloudlet(
            id, length, pesNumber, fileSize, outputSize,
            utilizationModel, utilizationModel,
            utilizationModel);
cloudlet1.setUserId(brokerId);
id++;
Cloudlet cloudlet2 = new Cloudlet(
            id, length * 2, pesNumber, fileSize * 2,
            outputSize / 3, utilizationModel,
            utilizationModel, utilizationModel);
cloudlet2.setUserId(brokerId);
id++;
Cloudlet cloudlet3 = new Cloudlet(
            id, length / 2, pesNumber, fileSize * 3,
```

```
                outputSize * 3, utilizationModel,
                utilizationModel, utilizationModel);
        cloudlet3.setUserId(brokerId);
        Cloudlet cloudlet4 = new Cloudlet(
                id, length / 3, pesNumber, fileSize / 3,
                outputSize / 2, utilizationModel,
                utilizationModel, utilizationModel);
        cloudlet4.setUserId(brokerId);
        Cloudlet cloudlet5 = new Cloudlet(
                id, length * 3, pesNumber, fileSize / 2,
                outputSize / 4, utilizationModel,
                utilizationModel, utilizationModel);
        cloudlet5.setUserId(brokerId);
        Cloudlet cloudlet6 = new Cloudlet(
                id, length / 4, pesNumber, fileSize * 4,
                outputSize * 4, utilizationModel,
                utilizationModel, utilizationModel);
        cloudlet6.setUserId(brokerId);
        Cloudlet cloudlet7 = new Cloudlet(
                id, length * 4, pesNumber, fileSize,
                outputSize * 2, utilizationModel,
                utilizationModel, utilizationModel);
        cloudlet7.setUserId(brokerId);
        Cloudlet cloudlet8 = new Cloudlet(
                id, length, pesNumber, fileSize / 4,
                outputSize / 3, utilizationModel,
                utilizationModel, utilizationModel);
        cloudlet8.setUserId(brokerId);

        // add the cloudlet to the list
        cloudletList.add(cloudlet1);
        cloudletList.add(cloudlet2);
        cloudletList.add(cloudlet3);
        cloudletList.add(cloudlet4);
        cloudletList.add(cloudlet5);
        cloudletList.add(cloudlet6);
        cloudletList.add(cloudlet7);
        cloudletList.add(cloudlet8);

        // submit cloudlet list to the broker
        broker.submitCloudletList(cloudletList);

        // bind the cloudlets to the vms,This way the
        // broker will submit the bound cloudlets only
```

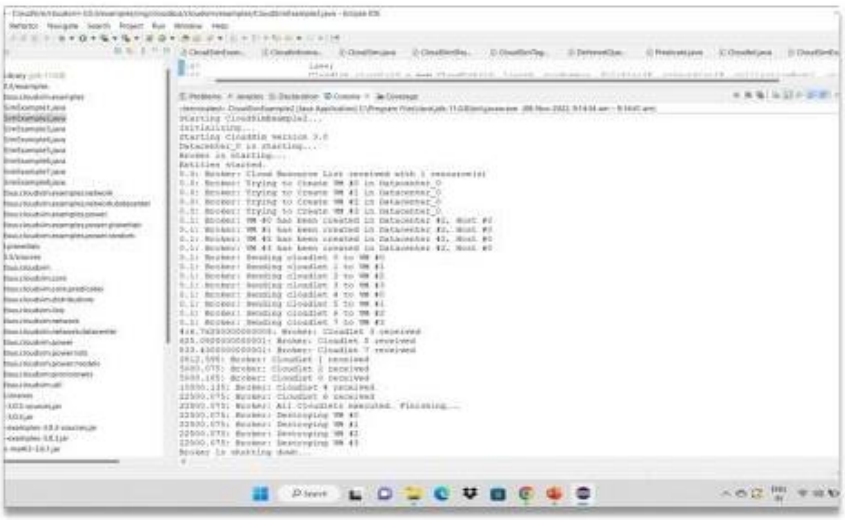
```
                // to the specific VM
                broker.bindCloudletToVm(
                        Cloudlet1.getCloudletId(), vm1.getId());
                broker.bindCloudletToVm(
                        Cloudlet2.getCloudletId(), vm2.getId());
                broker.bindCloudletToVm(
                        Cloudlet3.getCloudletId(), vm3.getId());
                broker.bindCloudletToVm(
                        Cloudlet4.getCloudletId(), vm4.getId());
                broker.bindCloudletToVm(
                        Cloudlet5.getCloudletId(), vm1.getId());
                broker.bindCloudletToVm(
                        Cloudlet6.getCloudletId(), vm2.getId());
                broker.bindCloudletToVm(
                        Cloudlet7.getCloudletId(), vm3.getId());
                broker.bindCloudletToVm(
                        Cloudlet8.getCloudletId(), vm4.getId());

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

                CloudSim.stopSimulation();

                // Final step: Print results when simulation is
                // over
                List<Cloudlet> newList
                        = broker.getCloudletReceivedList();
                printCloudletList(newList);

                Log.printLine("CloudSimExample1 finished!");
        }
        catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
        }
}

private static Datacenter createDatacenter(String name)
{

        // Here are the steps needed to create a
        // PowerDatacenter:
        // 1. We need to create a list to store
        // our machine
```

```
List<Host> hostList = new ArrayList<Host>();

// 2. A Machine contains one or more PEs or
// CPUs/Cores. In this example, it will have only
// one core.
List<Pe> peList = new ArrayList<Pe>();

int mips = 1000;

// 3. Create PEs and add these into a list.
// need to store Pe id and MIPS Rating
peList.add(
        new Pe(0, new PeProvisionerSimple(mips)));

// 4. Create Host 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 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
        = 0.001; // the cost of using storage in this
                        // resource
```

```
                double costPerBw
                        = 0.0; // 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;
        }

        private static DatacenterBroker createBroker()
        {
                DatacenterBroker broker = null;
                try {
                        broker = new DatacenterBroker("Broker");
                }
                catch (Exception e) {
                        e.printStackTrace();
                        return null;
                }
                return broker;
        }

        private static void
```

```
        printCloudletList(List<Cloudlet> list)
        {
                int size = list.size();
                Cloudlet cloudlet;

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

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

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

                                Log.printLine(
                                        indent + indent
                                        + cloudlet.getResourceId() + indent
                                        + indent + indent + cloudlet.getVmId()
                                        + indent + indent
                                        + dft.format(
                                                cloudlet.getActualCPUTime())
                                        + indent + indent
                                        + dft.format(
                                                cloudlet.getExecStartTime())
                                        + indent + indent
                                        + dft.format(cloudlet.getFinishTime()));
                        }
                }
        }
}
```

Step 5: Output after running the program is shown. One can see all the VM and cloudlets are created and their finish time(performance) is different as their VM and cloudlet configurations were different.



If we change the VM-cloudlet binding, we will get different outputs.

RESULT:

Successfully simulation using cloudsim done.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

ASSIGNMENT NO:4

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

Objective:

To learn the installation of virtual machine and transfer the file between them.

Outcome:

To file transfer between two virtual machine.

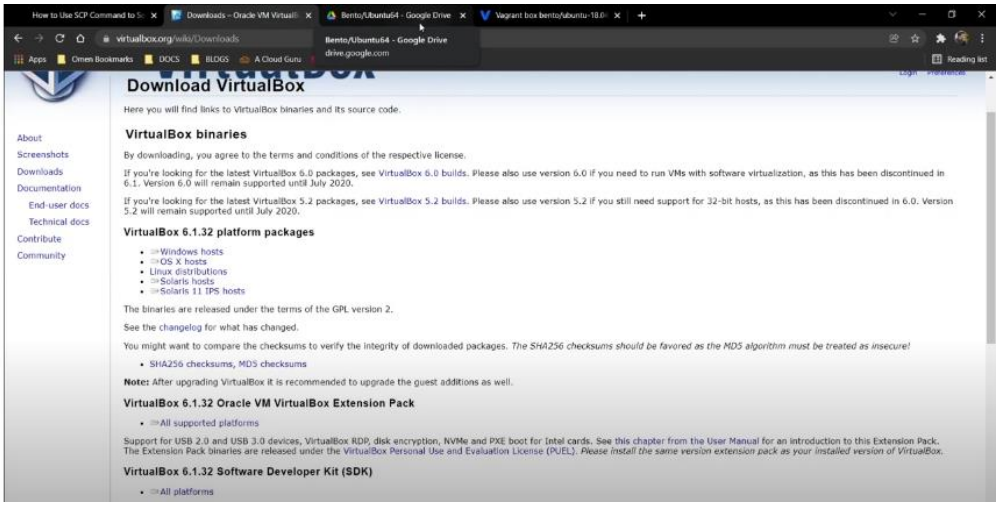
Explanation:

Installation / Download and Reference Material

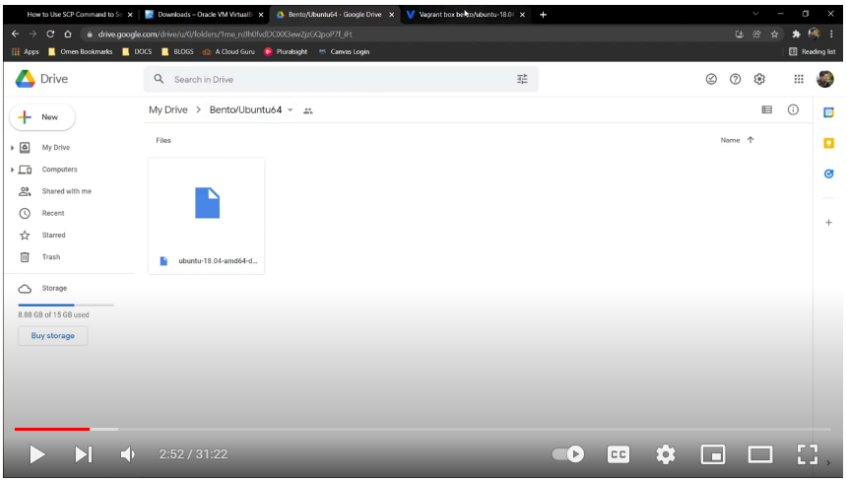
- Oracle's Virtual Box
- Ubuntu 64 VMDK Image
- SCP Article

Steps :

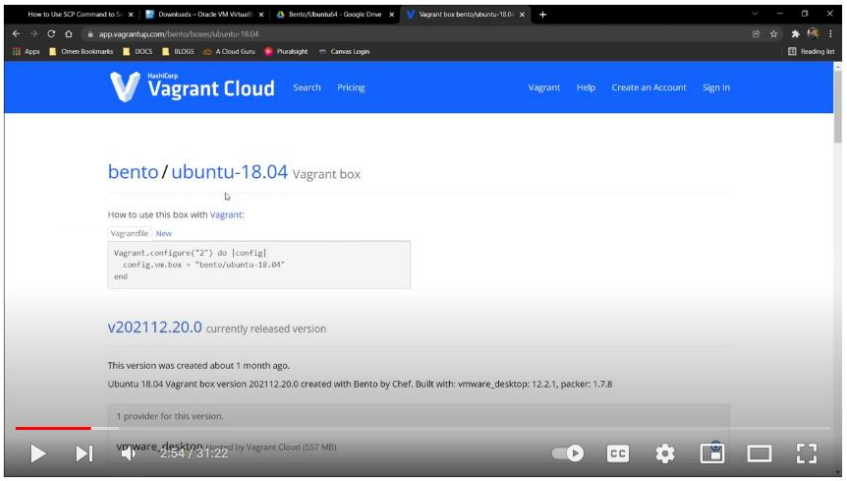
1. Download and install Oracle's Virtual Box. (Reboot needed after installation)



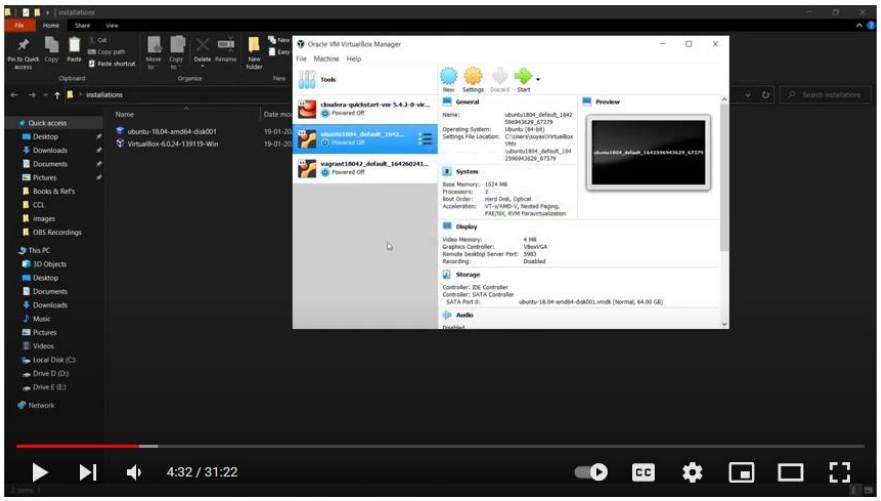
2. Download Ubuntu VMDK Image.



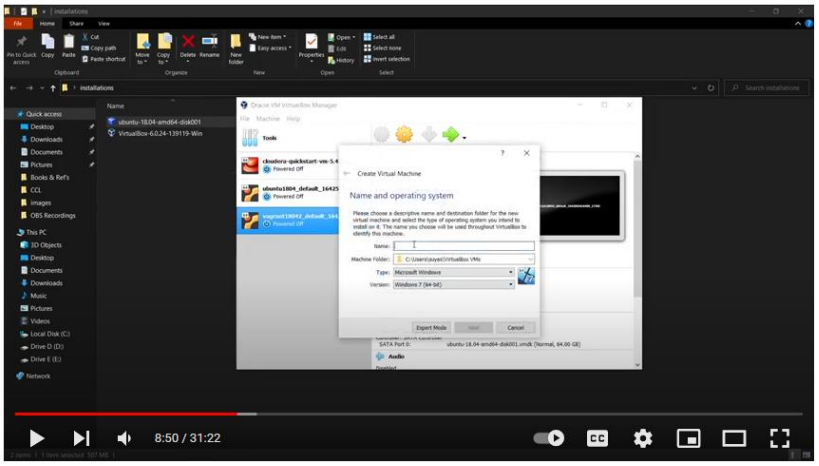
OR



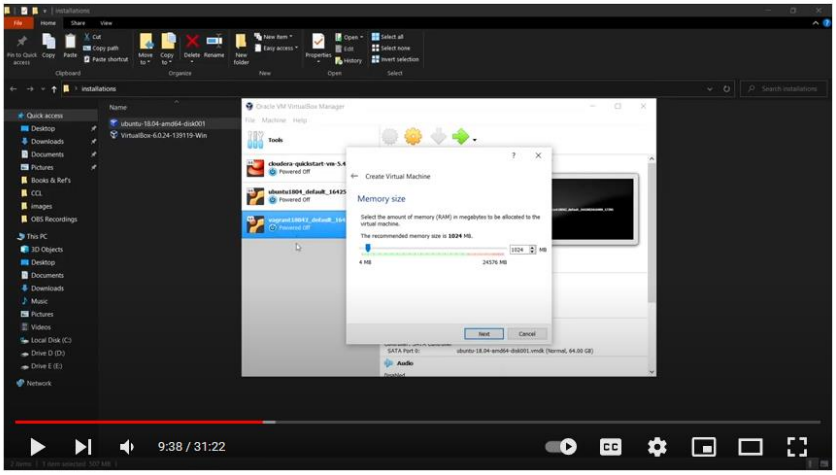
3. Launch Virtualbox and create a new VM.



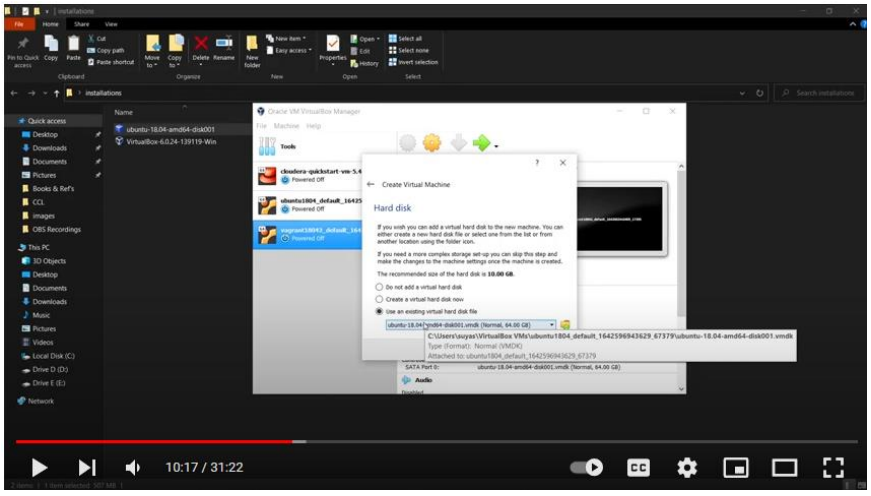
4. Click on new and mention the Name and the machine folder along with the Type and Version of the Machine to be created.



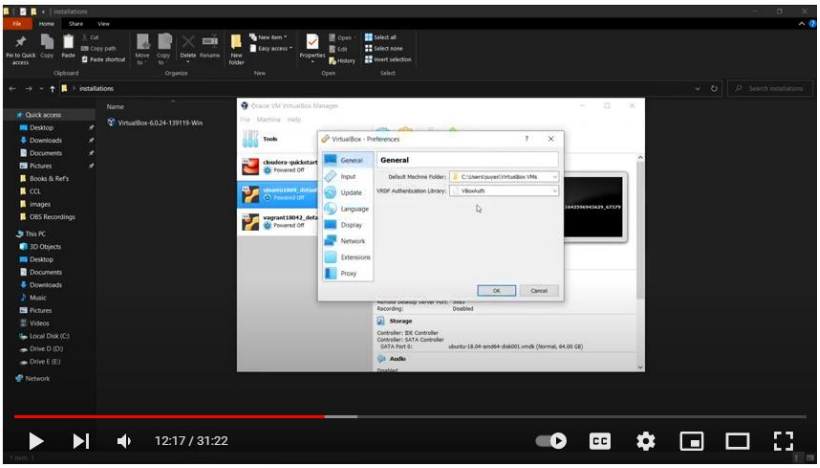
5. Assign memory size for our VM (1024 MB sufficient for now).



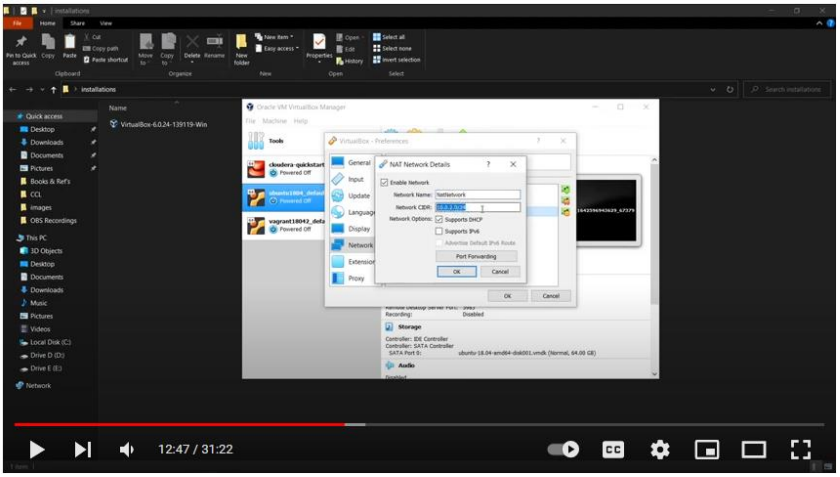
6. Select the option **Use an existing virtual hard disk file** and locate the donwloaded VMDK image below and create VM.



7. Now we have to create a NAT Network so go to **File -> Preferences -> Network -> Add a New NAT Network (Click on +)**



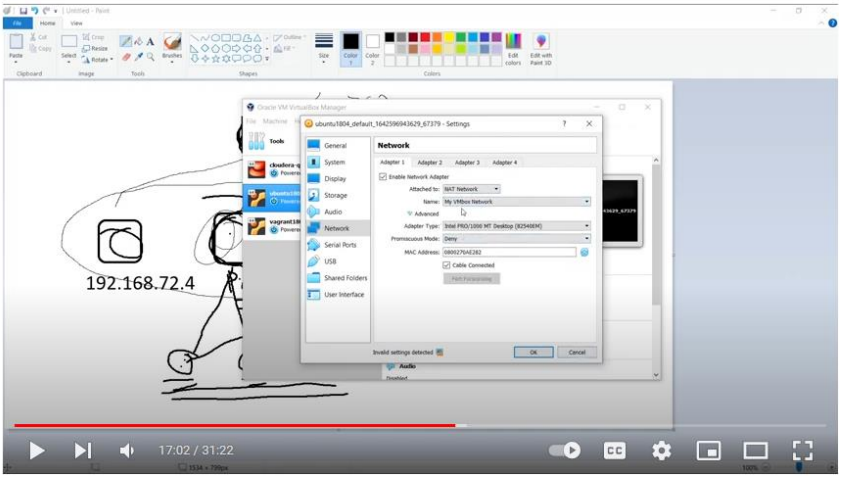
8. Right click and edit the Network name and CIDR if needed. Example :\



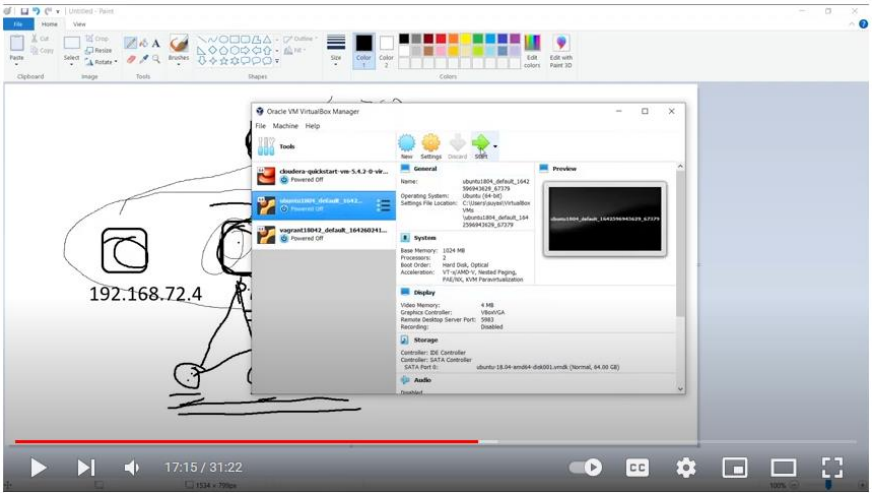

```
Name - My VMbox Network
CIDR - 172.168.2.0/24 and save the changes.
```

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 and select the NAT Network you made (in our case : **My VMbox Network**) and click ok.

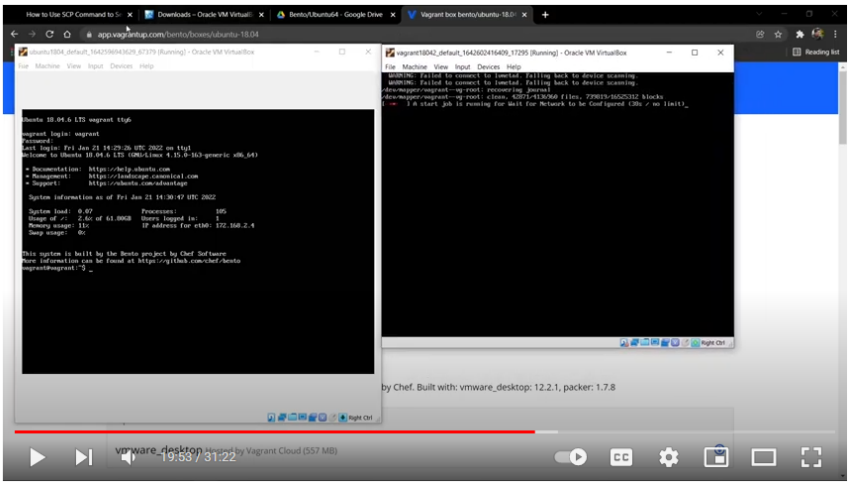


11. Launch the VM now.



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

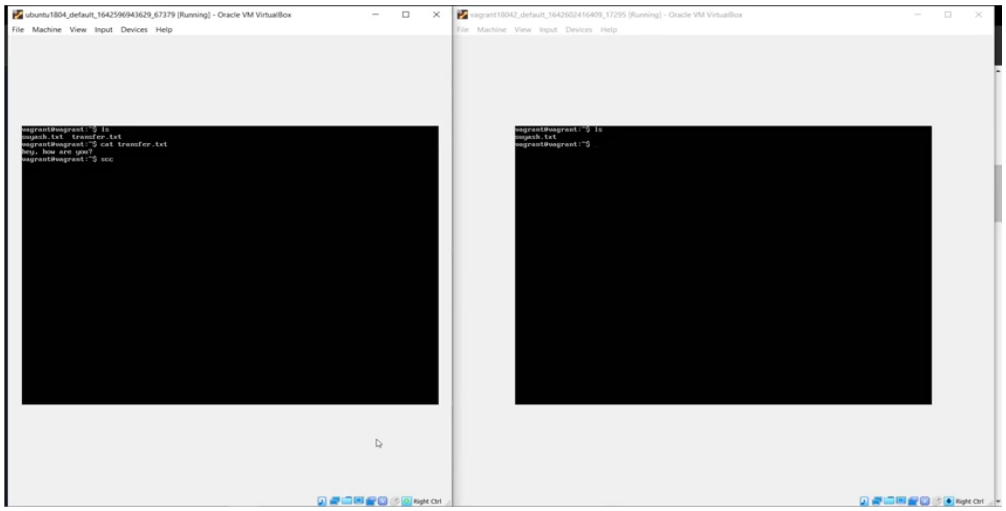
```
$ ifconfig
```

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
```

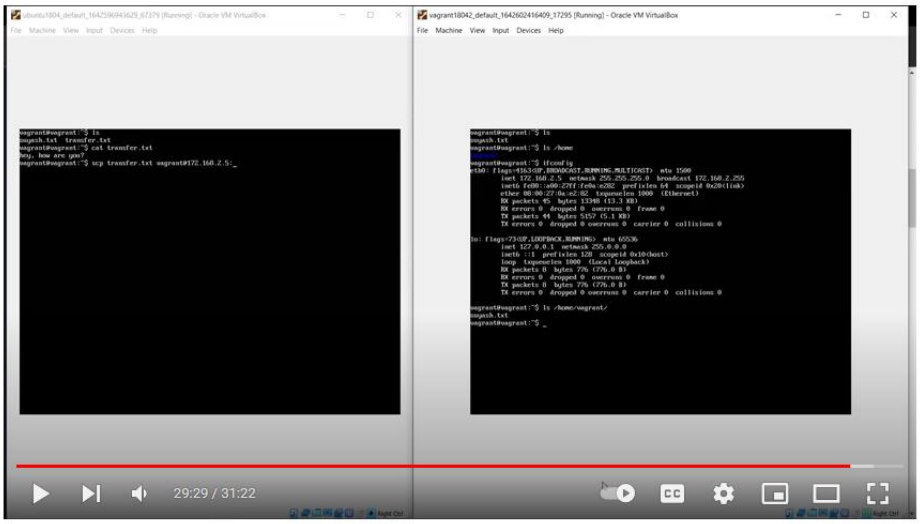


```
Some Commands for Linux Based Distros:

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**.

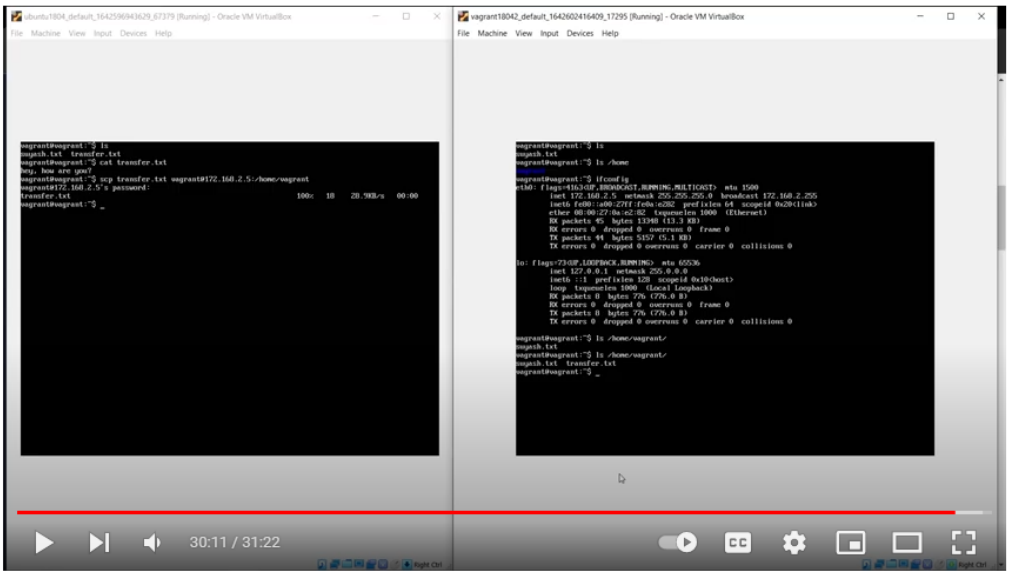
16. Transfer the file using **SCP**

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



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

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



Done..!!!!

RESULT:

File transfer between 2 vm successfully done.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

ASSIGNMENT NO. – 5

AIM: Find a procedure to launch virtual machine using try stack(online open stack Demo Version).

Objective:

To learn the installation of AWS and launch an Amazon EC2 Instance.

Outcome:

Launch a Linux Virtual Machine with Amazon EC2

Explanation:

OpenStackis an open-source software cloud computing platform.
OpenStack is primarily usedfor deploying an infrastructure as a service (IaaS) solution like Amazon Web Service (AWS).
TryStackis the easiest and free way to do it.
Minimum requirements for OpenStack is listed below:
4 GB Of Ram.
4 CPU Units.
30 GB Disk Space.

Prepare the environment for installing OpenStack:

Run the following commands and you will be done with it tobring all your packages to thelatest version and install git so that we can clone OpenStackto our Linux machine

Commands :

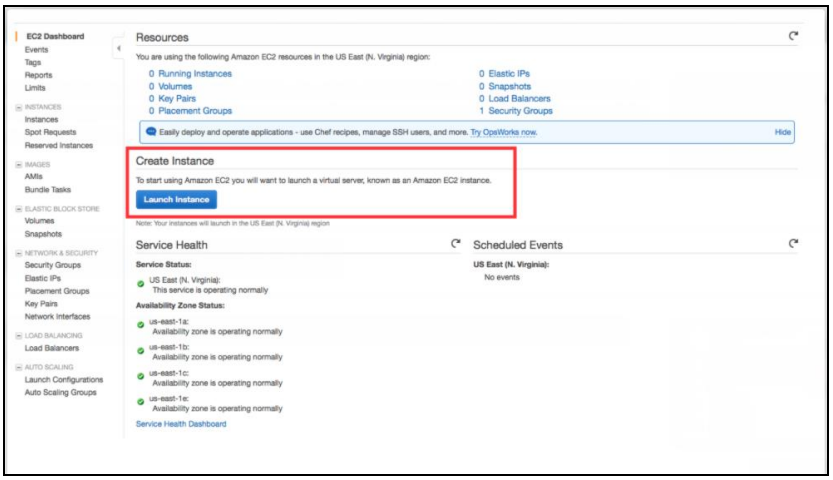
- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get dist-upgrade
- sudo apt-get install git -y
- sudo reboot

Launch a Linux Virtual Machine with Amazon EC2

Amazon Elastic Compute Cloud (EC2) is the web service you use to create and run virtual machines in the cloud. Amazon Web Services calls these virtual machines 'instances'. This step-by-step guide will help you successfully launch a Linux virtual machine on Amazon EC2.

Step 1: Launch an Amazon EC2 Instance

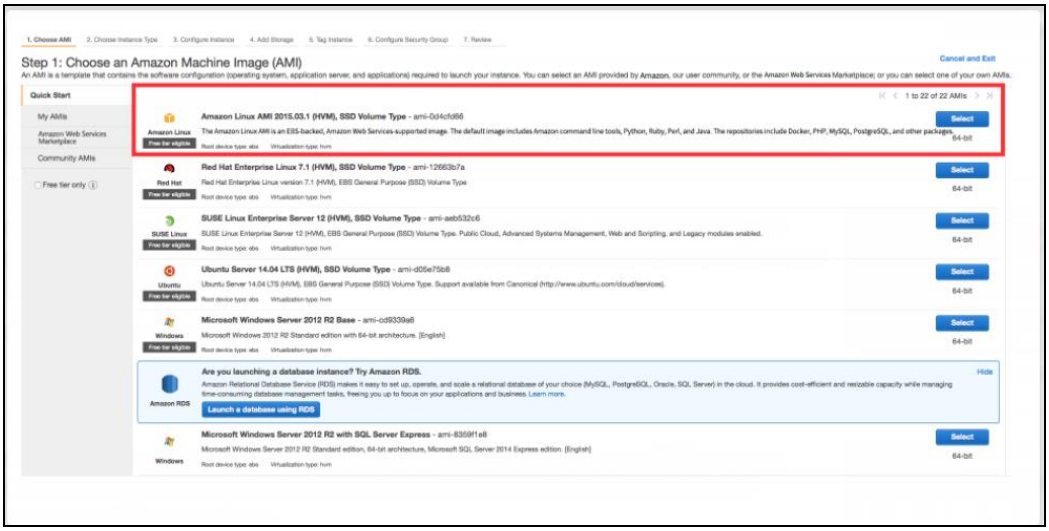
a. Click here to open the Amazon EC2 console and then click Launch Instance to create and configure your virtual machine.



Step 2: Configure your Instance

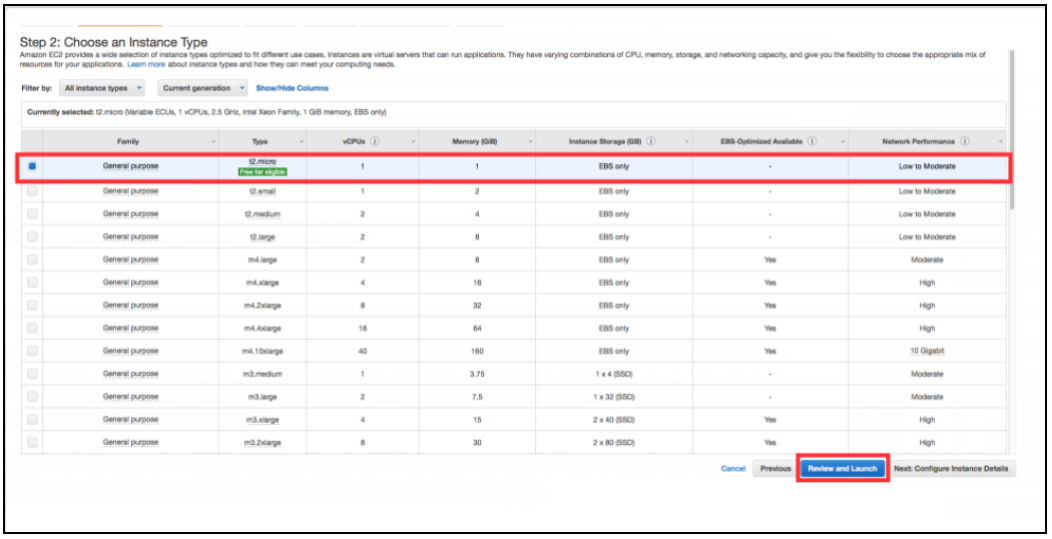
You are now in the Amazon EC2 Launch Instance Wizard, which will help you configure and launch your instance.

a) In this screen, you are shown options to choose an Amazon Machine Image (AMI). AMIs are preconfigured server templates you can use to launch an instance. Each AMI includes an operating system, and can also include applications and application servers.



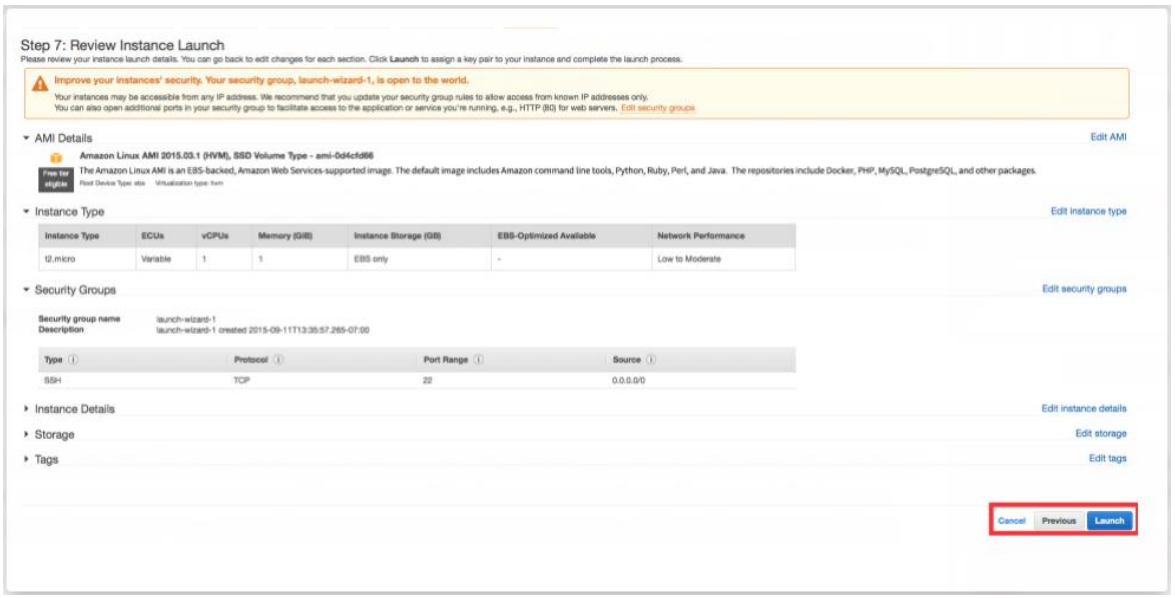
b) You will now choose an instance type. Instance types comprise of varying combinations of CPU, memory, storage, and networking capacity so you can choose the appropriate mix for your applications.

The default option of *t2.micro* should already be checked. This instance type offers enough compute capacity to tackle simple workloads. Click Review and Launch at the bottom of the page.



c) You can review the configuration, storage, tagging, and security settings that have been selected for your instance. While you have the option to customize these settings, we recommend accepting the default values for this tutorial.

Click Launch at the bottom of the page.

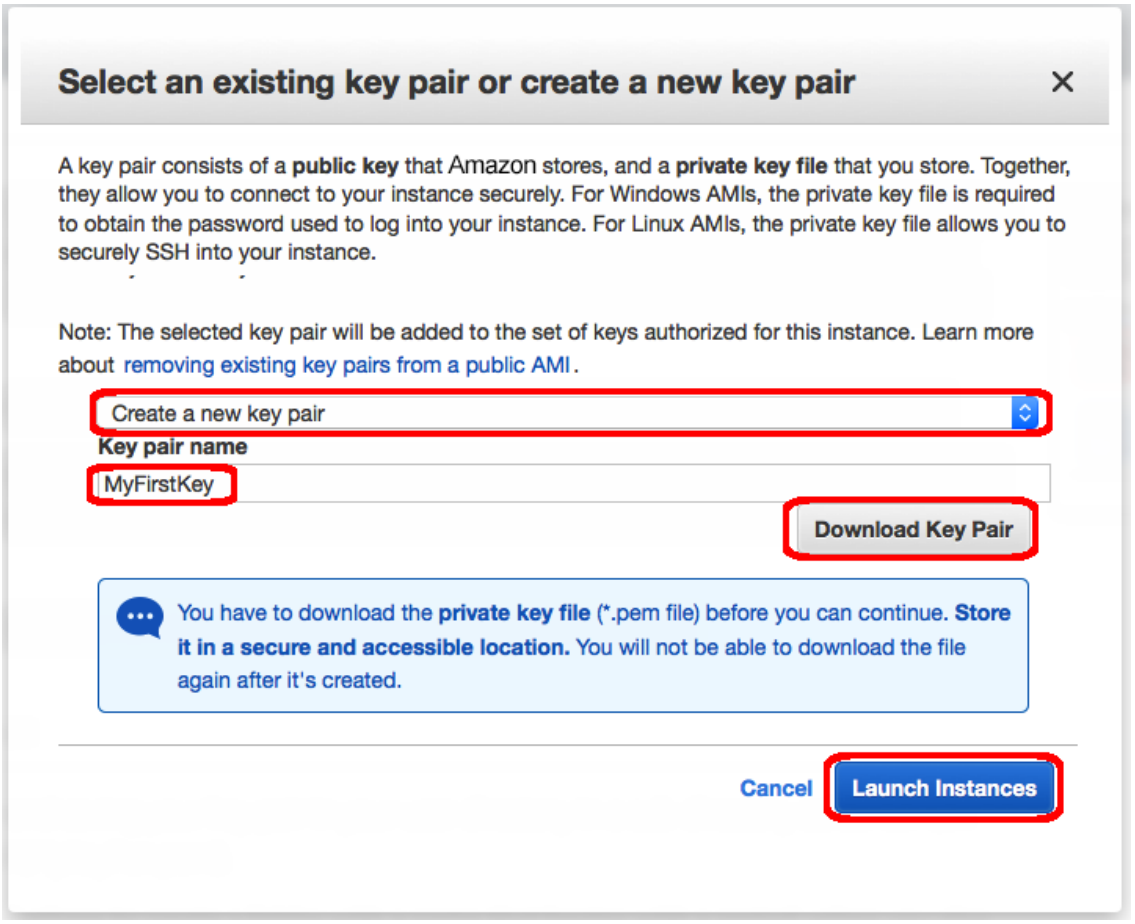


d) On the next screen you will be asked to choose an existing key pair or create a new key pair. A key pair is used to securely access your Linux instance using SSH. Amazon Web Services stores the public part of the key pair which is just like a house lock. You download and use the private part of the key pair which is just like a house key.

Select Create a new key pair and give it the name MyKeyPair. Next click the Download Key Pair button.

After you download the MyKeyPair key, you will want to store your key in a secure location. If you lose your key, you won't be able to access your instance. If someone else gets access to your key, they will be able to access your instance.

Select your operating system below to see instructions on where to save your SSH key.

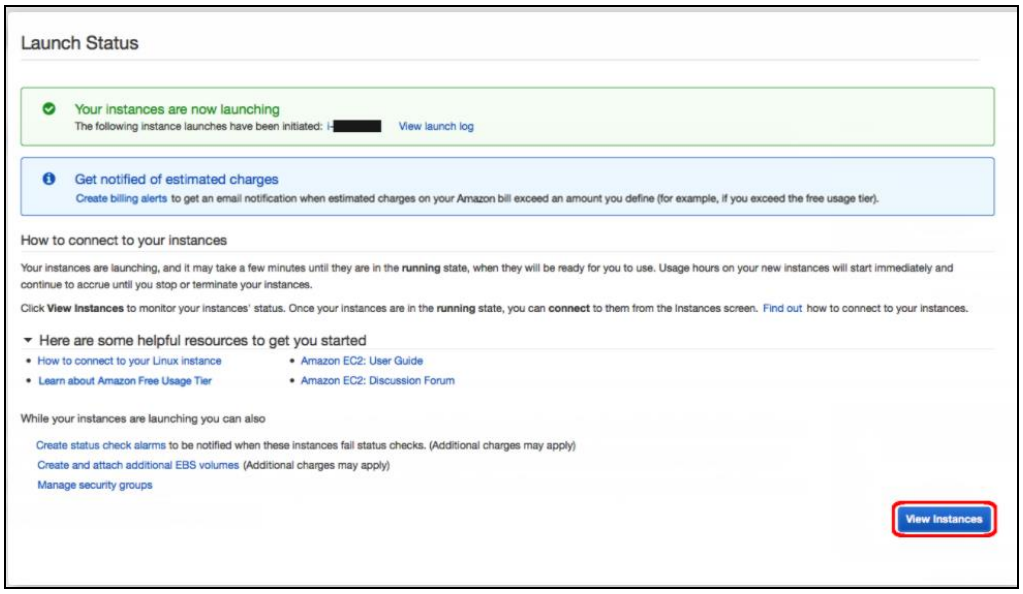


Windows users: We recommend saving your key pair in your user directory in a sub-directory called .ssh (ex. C:\user\{yourusername}\.ssh\MyKeyPair.pem).

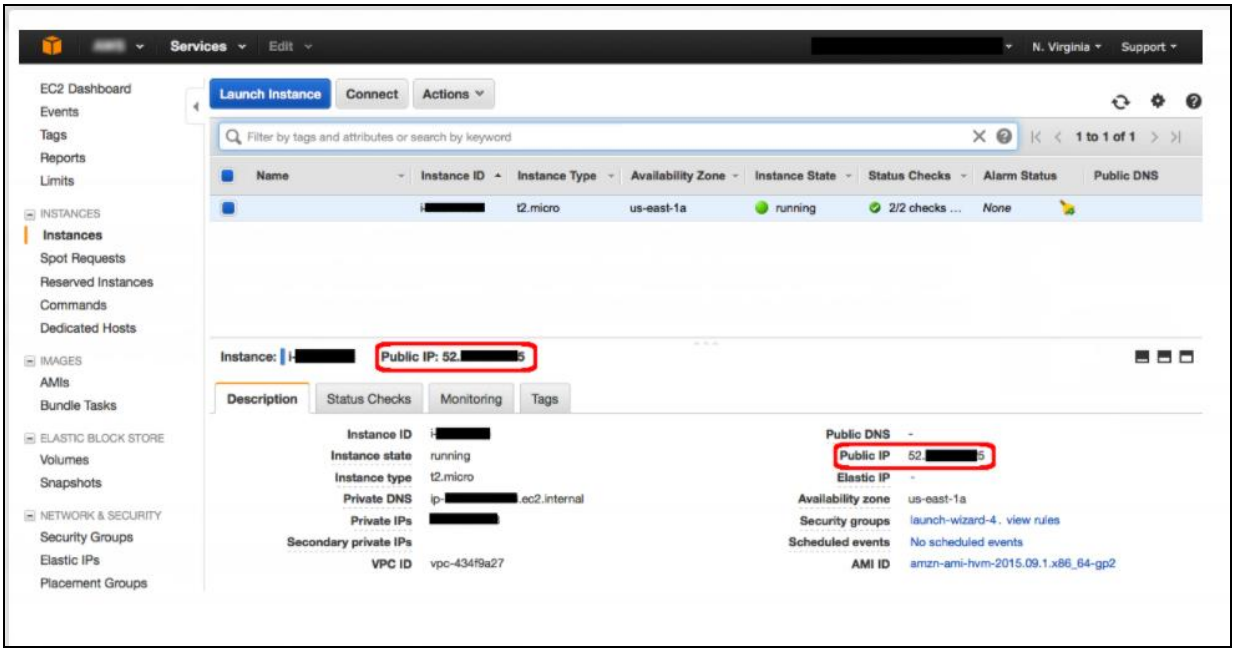
Tip: You can't use Windows Explorer to create a folder with a name that begins with a period unless you also end the folder name with a period. After you enter the name (.ssh.), the final period is removed automatically.After you have stored your key pair, click Launch Instance to start your Linux instance.

After you have stored your key pair, click Launch Instance to start your Linux instance.

e) Click View Instances on the next screen to view your instances and see the status of the instance you have just started.



f) In a few minutes, the *Instance State* column on your instance will change to "*running*" and a Public IP address will be shown. You can refresh these Instance State columns by pressing the refresh button on the right just above the table. Copy the Public IP address of your Amazon Web Services instance, so you can use it when we connect to the instance using SSH in Step 3.

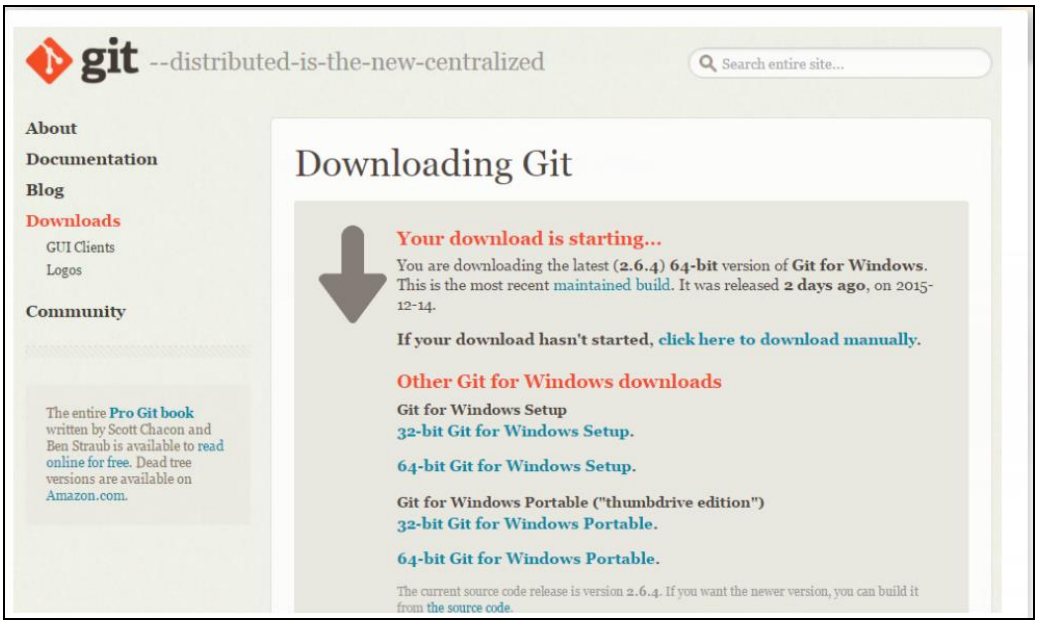


Step 3: Connect to your Instance

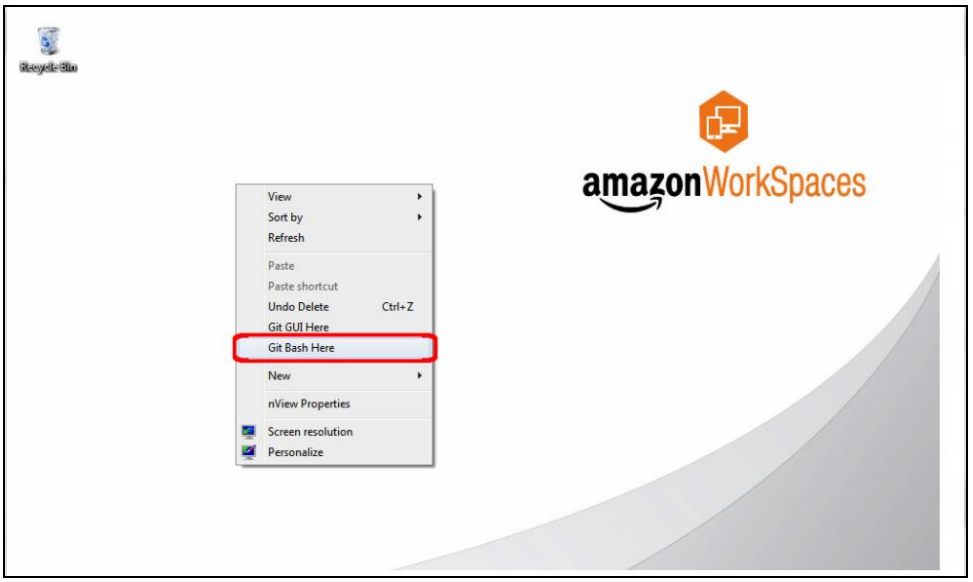
After launching your instance, it's time to connect to it using SSH.

Windows users: Select Windows below to see instructions for installing Git Bash which includes SSH.

a. Download Git for Windows here. Run the downloaded installer accepting the default settings (this will install Git Bash as part of Git).



b. Right click on your desktop (not on an icon or file) and select Git Bash Here to open a Git Bash command prompt.



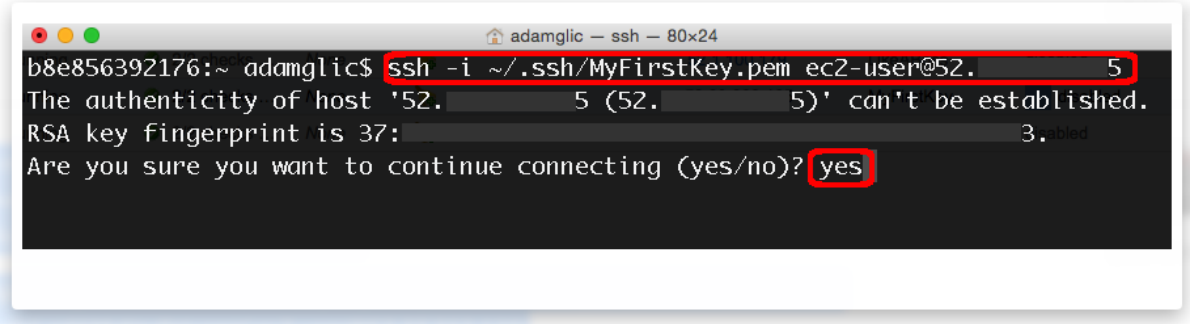
C. Use SSH to connect to your instance. In this case the user name is ec2-user, the SSH key is stored in the directory we saved it to in step 2 part d, and the IP address is from step 2 part f. The format is *ssh -i {full path of your .pem file} ec2-user@{instance IP address}*.

Enter ssh -i 'c:\Users*yourusername*\.ssh\MyKeyPair.pem' ec2-user@*{IP_Address}* (ex. *ssh -i 'c:\Users\adamglic\.ssh\MyKeyPair.pem' ec2-user@52.27.212.125*)

You'll see a response similar to the following:

The authenticity of host 'ec2-198-51-100-1.compute-1.amazonaws.com (10.254.142.33)' can't be established. RSA key fingerprint is 1f:51:ae:28:df:63:e9:d8:cf:38:5d:87:2d:7b:b8:ca:9f:f5:b1:6f. Are you sure you want to continue connecting (yes/no)?

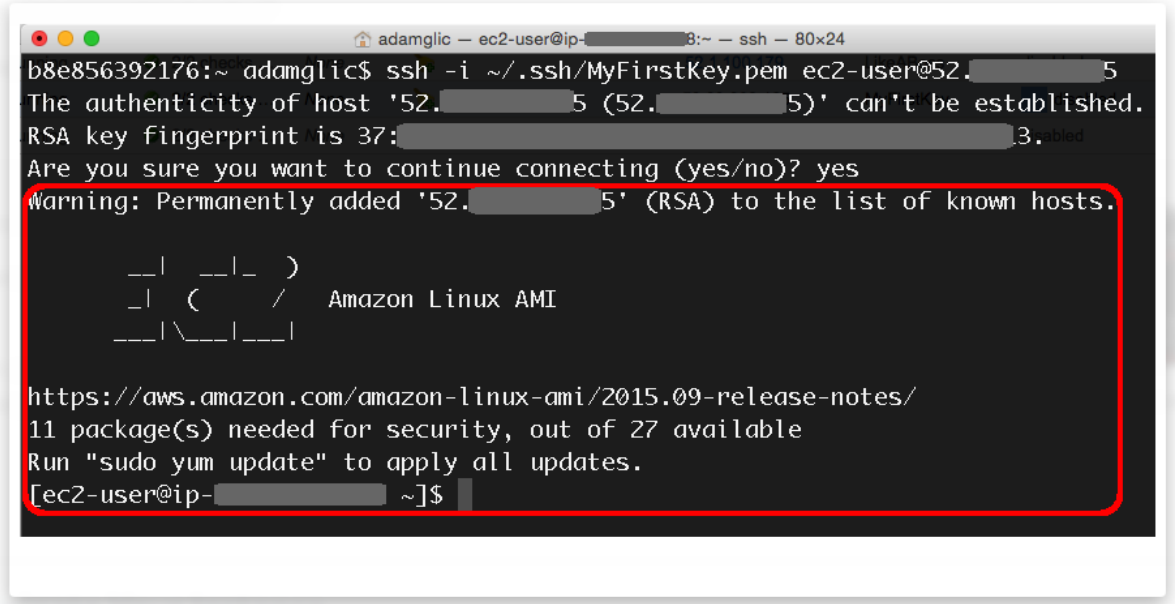
Type yes and press enter.



You'll see a response similar to the following:

Warning: Permanently added 'ec2-198-51-100-1.compute-1.amazonaws.com' (RSA) to the list of known hosts.

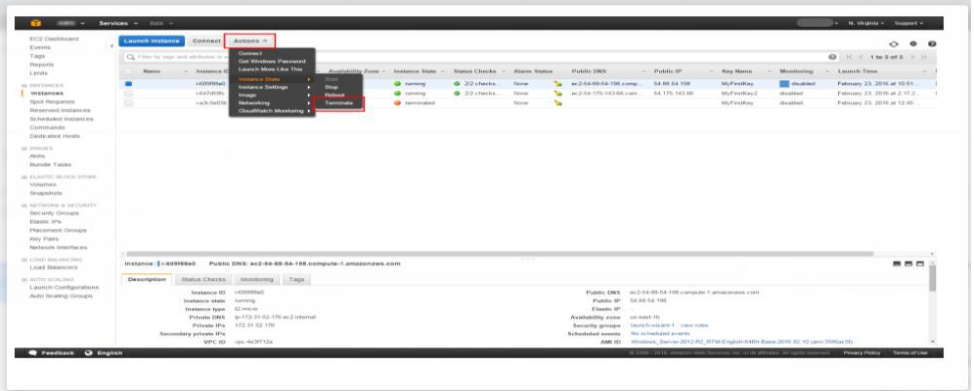
You should then see the welcome screen for your instance and you are now connected to your Amazon Web Services Linux virtual machine in the cloud.



<u>Step 4: Terminate Your Instance:</u>

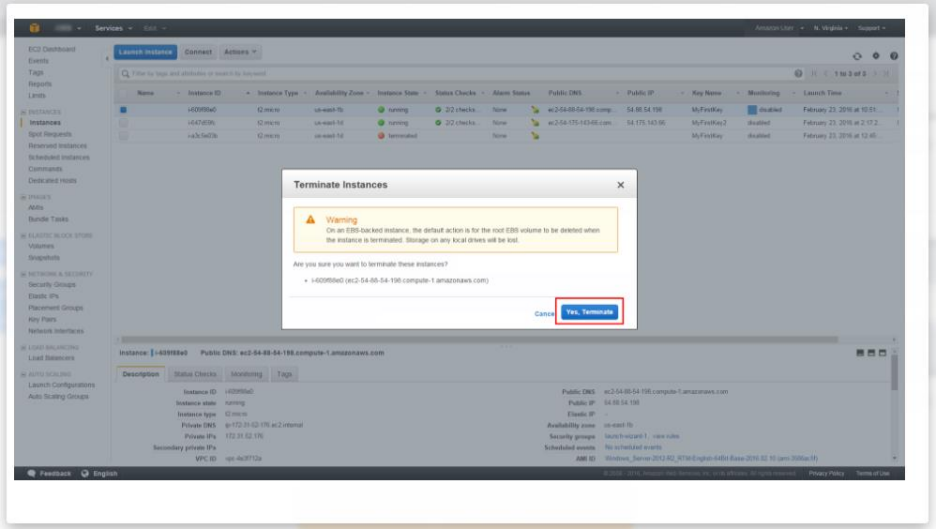
You can easily terminate the instance from the Amazon EC2 console. In fact, it is a best practice to terminate instances you are no longer using so you don't keep getting charged for them.

a. Back on the Amazon EC2 Console, select the box next to the instance you created. Then click the Actions button, navigate to *Instance State*, and click Terminate.



b. You will be asked to confirm your termination - select Yes, Terminate.

Note: This process can take several seconds to complete. Once your instance has been terminated, the Instance State will change to *terminated* on your Amazon EC2 Console.



Result:

Successfully launch virtual machine using trystack.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

ASSIGNMENT NO. – 6

AIM: Design and deploy a web application in a PaaS environment.

Objective:
To learn the installation of AWS and about working on PaaS.

Outcome:
Launching web application in PaaS environment.

Explanation:

What is PaaS?
Platform as a service (PaaS) is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.

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.

You will need the following to steps:

- Introduction to AWS
- Subscription of AWS.
- AWS Toolkit for Visual Studio – I.
- Visual Studio – I will be using 2015.

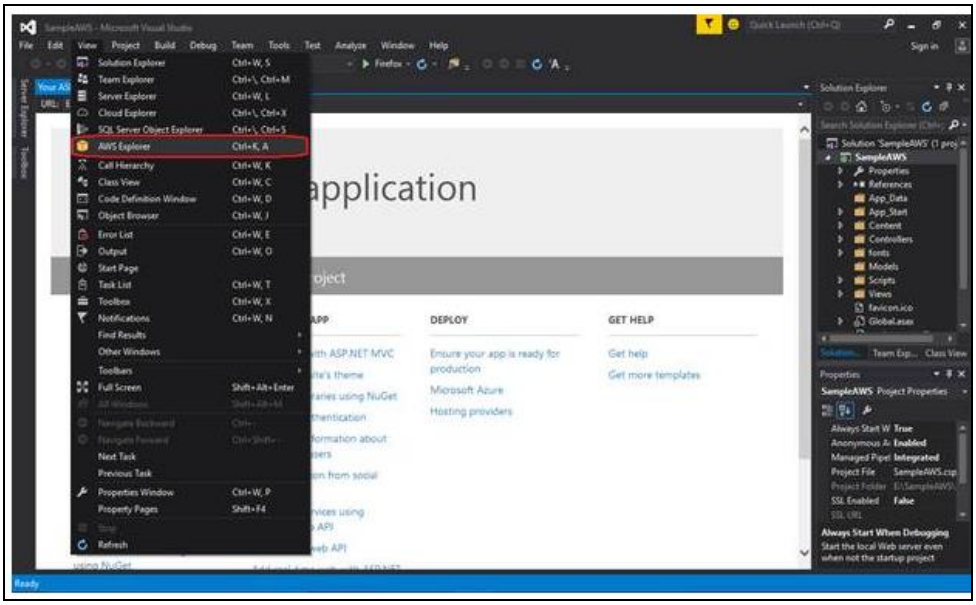
Install AWS Toolkit for Visual Studio

Go to AWS toolkit for Visual Studio,

Download the SDK and install it.

To check whether it is installed in Visual Studio or not, open Visual Studio – View.

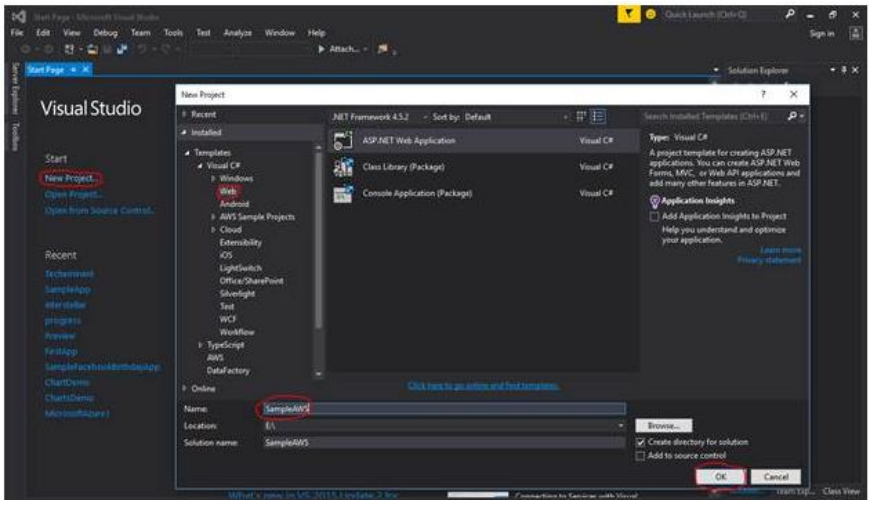
If you see AWS Explorer in the View sub menu that means the AWS toolkit has been installed.



Creating a Project in Visual Studio

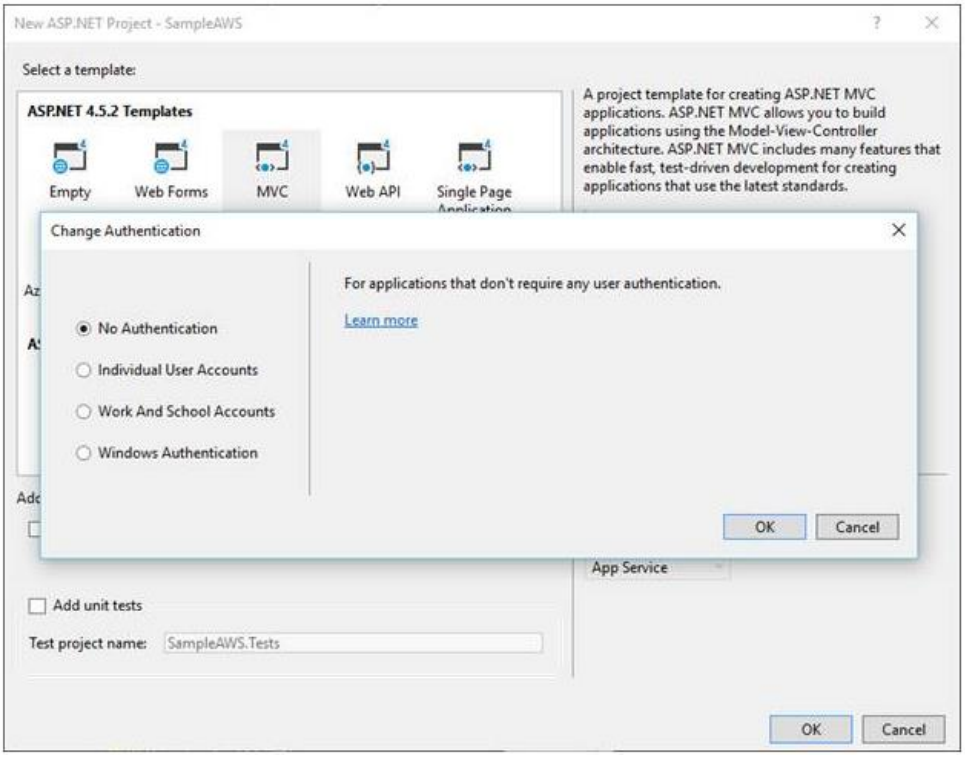
Create a new MVC project.

Click on New Project > Web > ASP.NET Web Application.

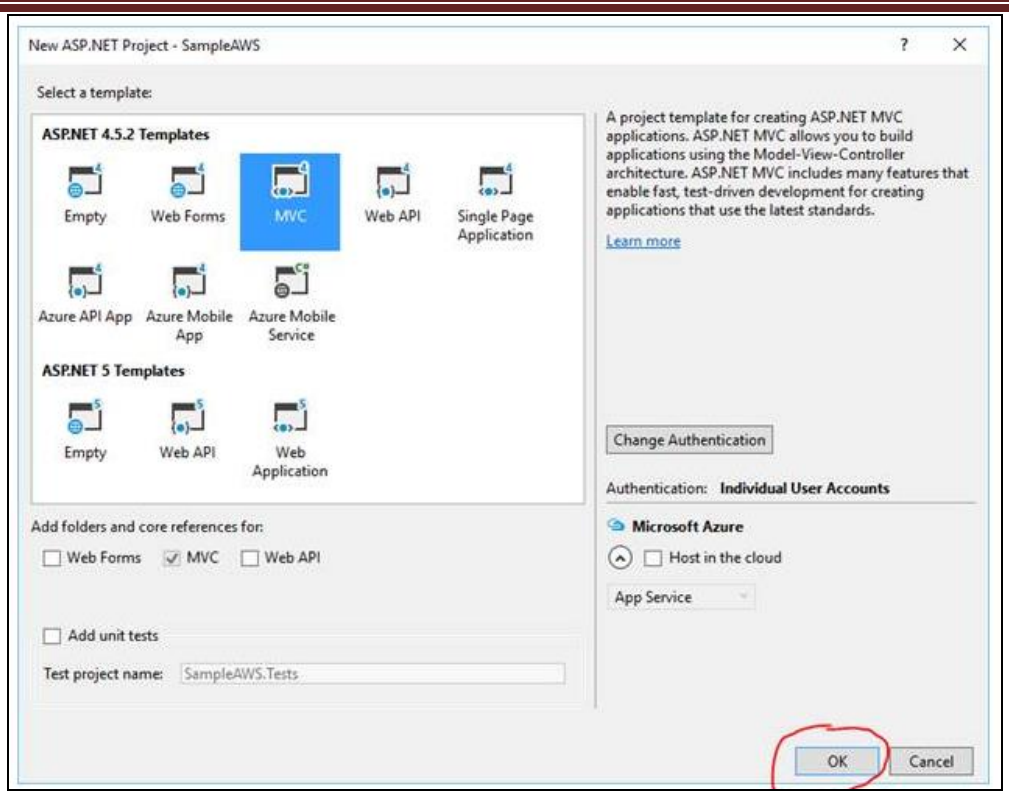


Rename the application and click on OK.

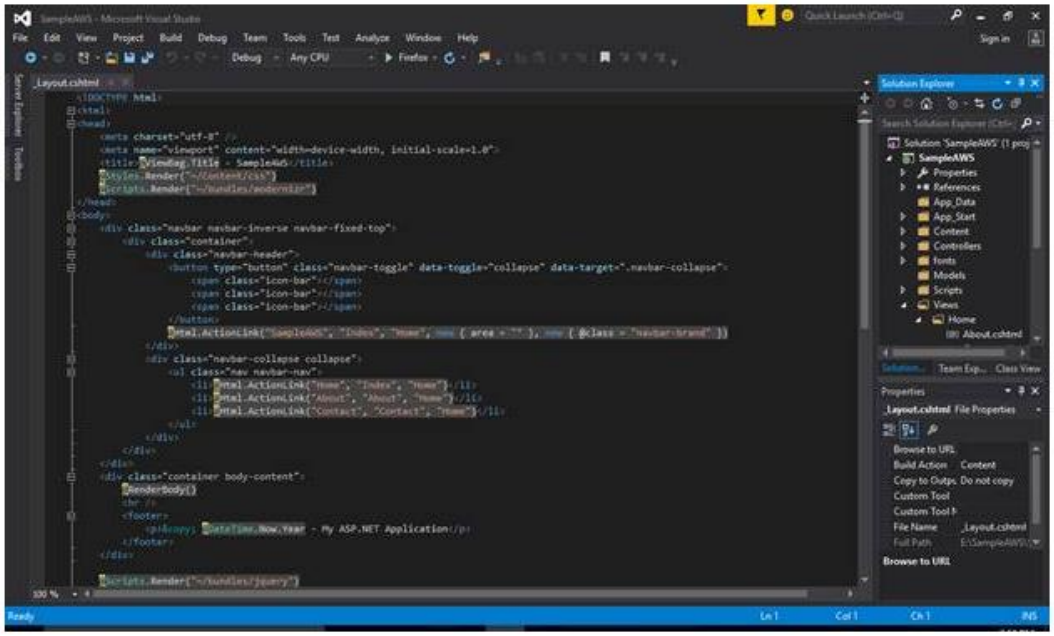
On the next screen you have the choice to choose the web application type. I am choosing MVC with No Authentication.



Choose your web application type and click on OK.

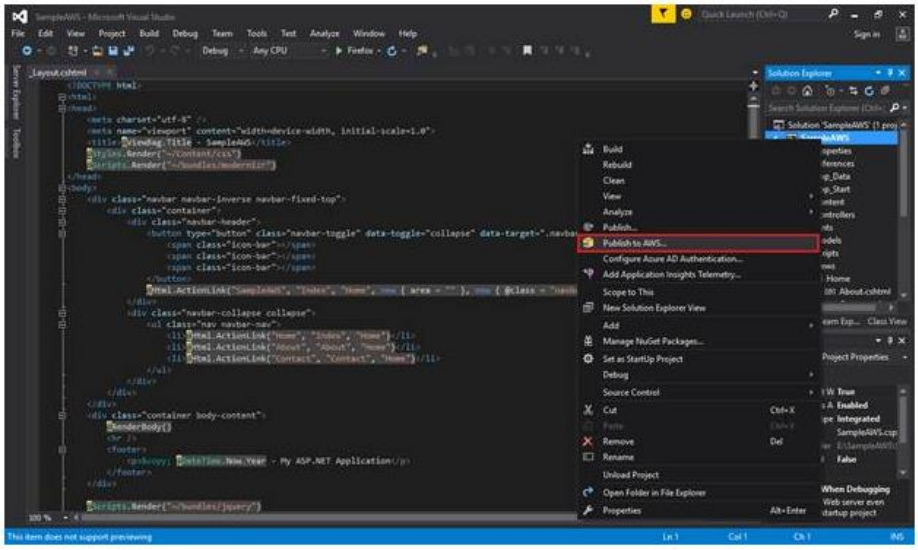


The application is ready now.

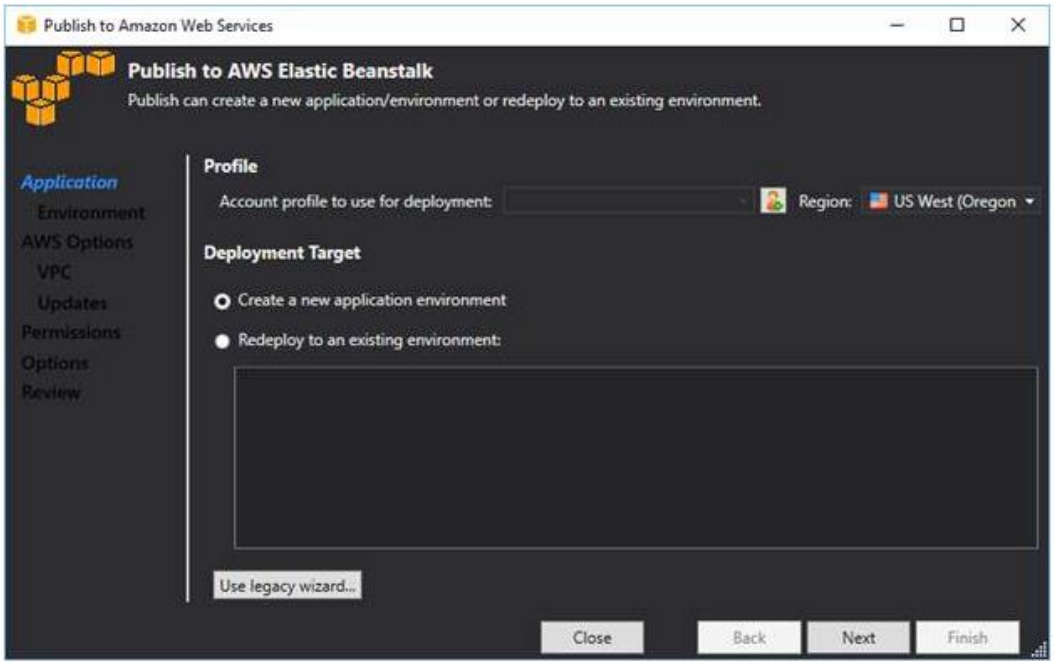


Do some editing if you want. leaving the application as it is and moving ahead to deployment part of the the application.

So, right-click on the project and click on Publish to AWS.



On next screen is for publishing to AWS Elastic Beanstalk.



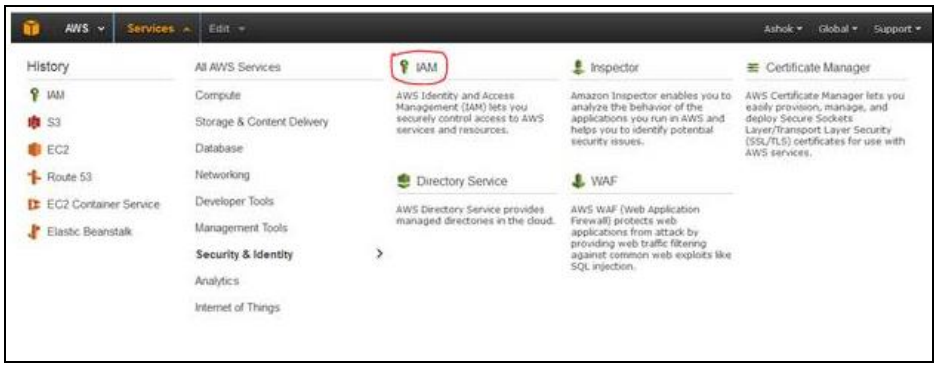
Currently there is no account profile so we have to create one first.

Creating Identity in AWS

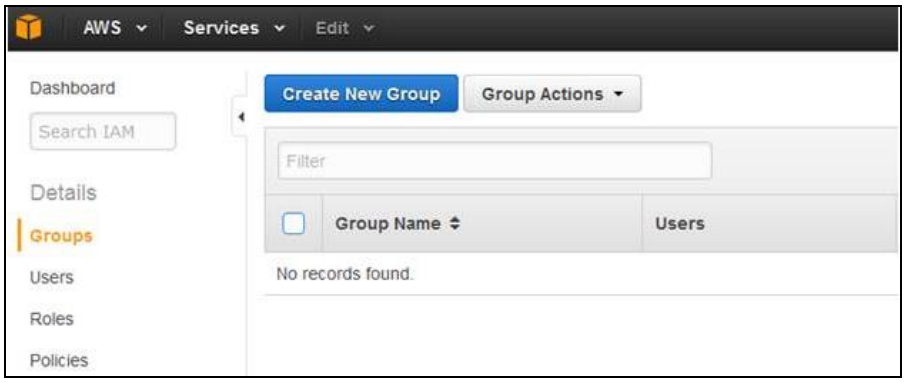
Before deploying an application to AWS, you must have an identity. Identities are created using group and group policies. After a group is created, we can attached users to the group. In then end, AWS creates a pair of Access Key ID and Access Secret Key.

This is the identity of the app that must be attached to the web application when we deploy it to AWS.

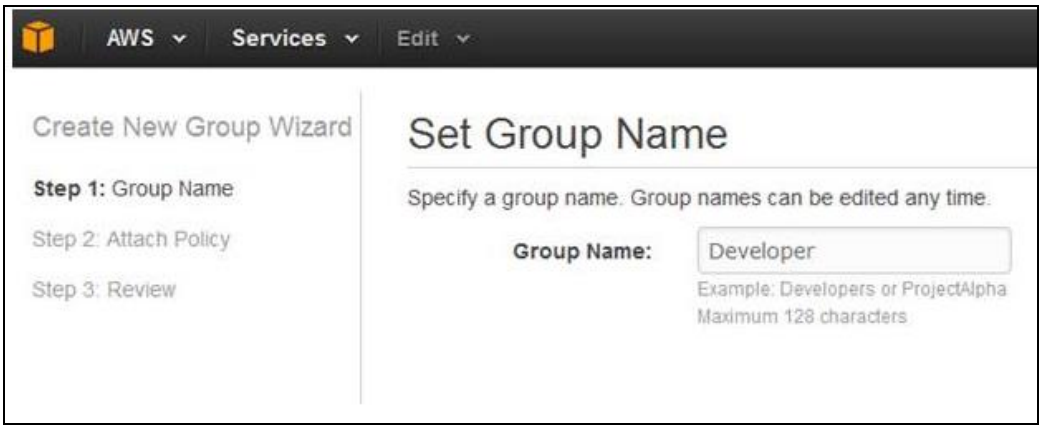
For that first, go to AWS dashboard and under Security & Identity category click on IAM,



Now on IAM dashboard, we need to create a Group and its policies.

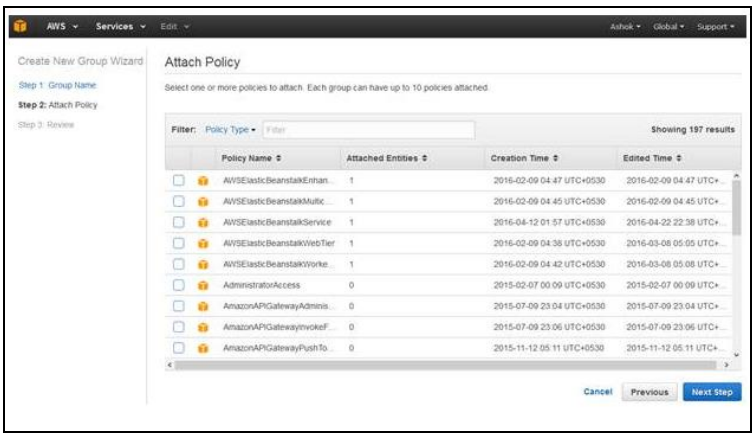


Give a proper name to the group.Usually, its a group that will have access to this application. For example, in your company, if the operations department should have access to this app, then that group needs to be a part of this group.

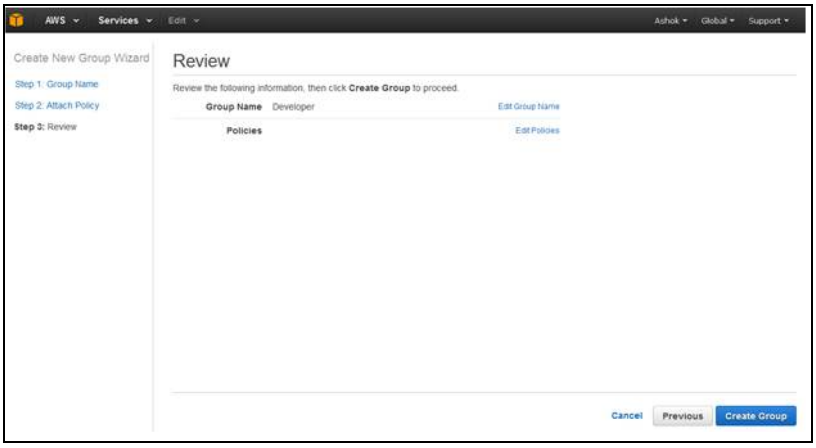


Each group has some group policies. On the next screen, you're asked to attach a policy.

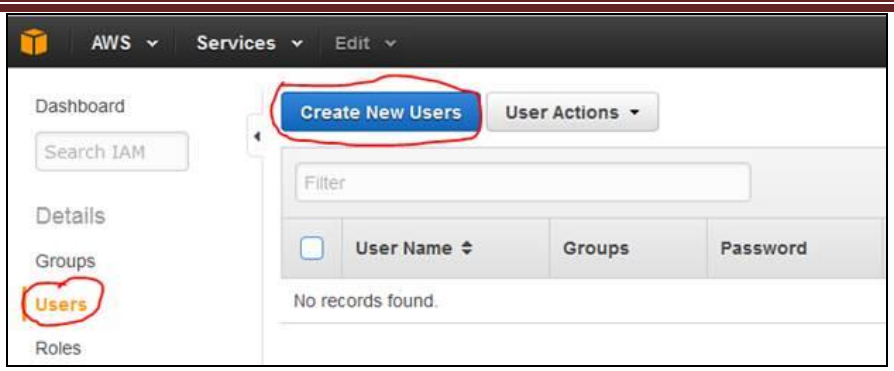
Don’t attach any policy to the group just click on Next Step.



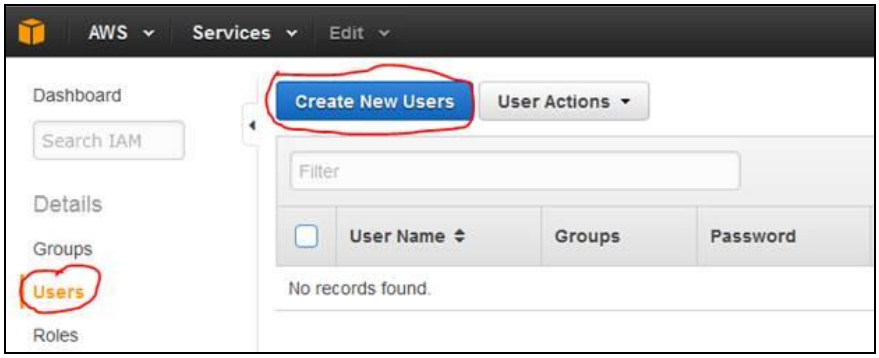
And create a group.



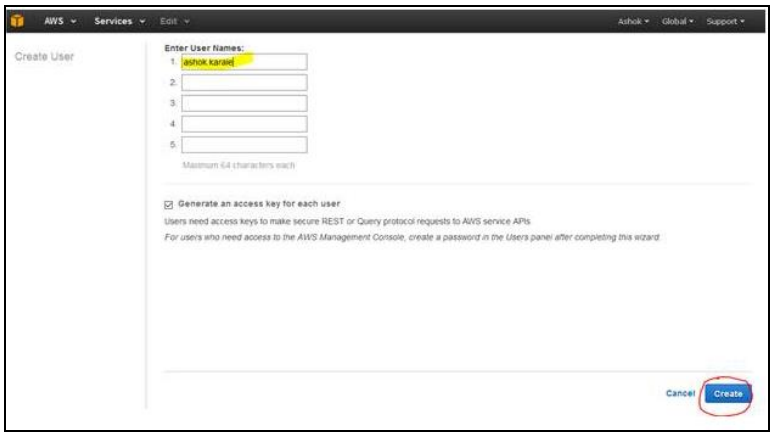
Ok now you have a group. Click on the group you just created. It will show that there are no users in the group. So we will be required to create a user. Click on Users, which is on the left-hand side menu. Then click on Create new users. If you already have users in your company account, you may see list of availalbe users.



Ok now you have a group. Click on the group you just created. It will show that there are no users in the group. So we will be required to create a user. Click on Users, which is on the left-hand side menu. Then click on Create new users. If you already have users in your company account, you may see list of availalbe users.



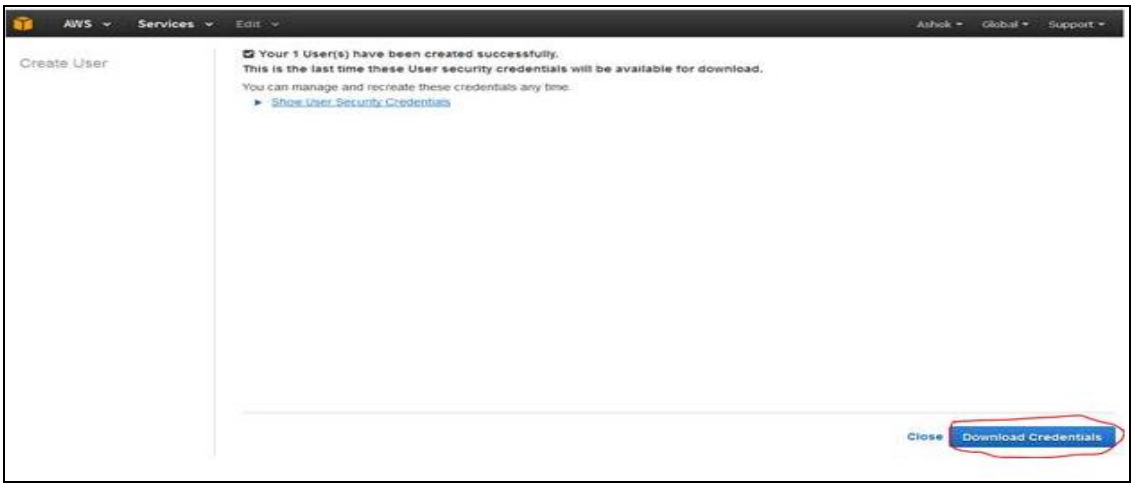
You can add multiple users at a time but I am just creating one. So just enter the user name and click on Create.



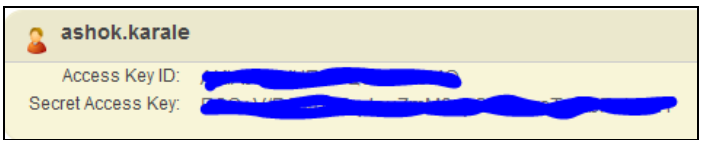
What this Create button does it, it creates an identity for the app, which is required when we publish/deploy this app to AWS.

Download the credentials and save them at an accessible location.

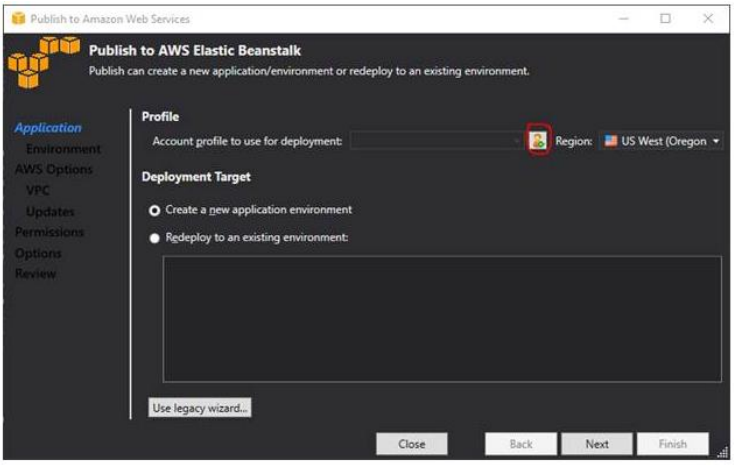
See Download Credentials button below.



Finally, we have an Access Key ID and and Secret Access Key.



Now go back to the Visual Studio on Publish to AWS Elastic Beanstalk wizard and click on Add another account profile.

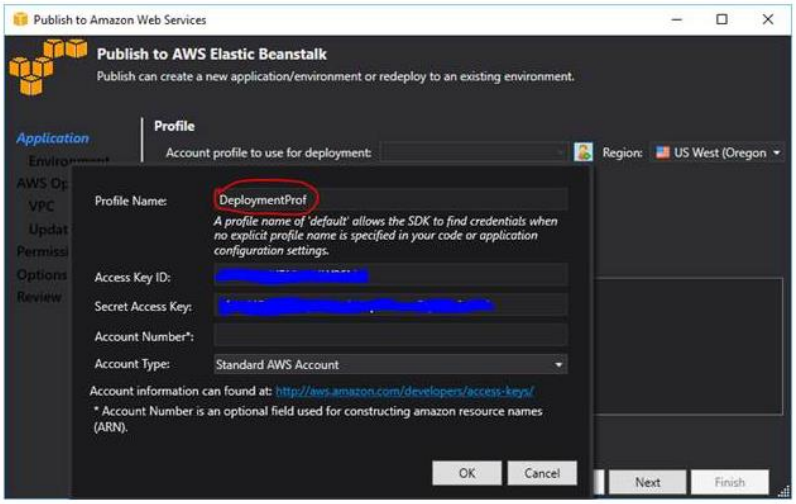


A new screen will pop up to fill the profile credential. Fill all the details.

- Profile name – Give a app profile name. It can be anything.
- Access Key ID and Secret Access Key - Enter what we had just created in the previous step.

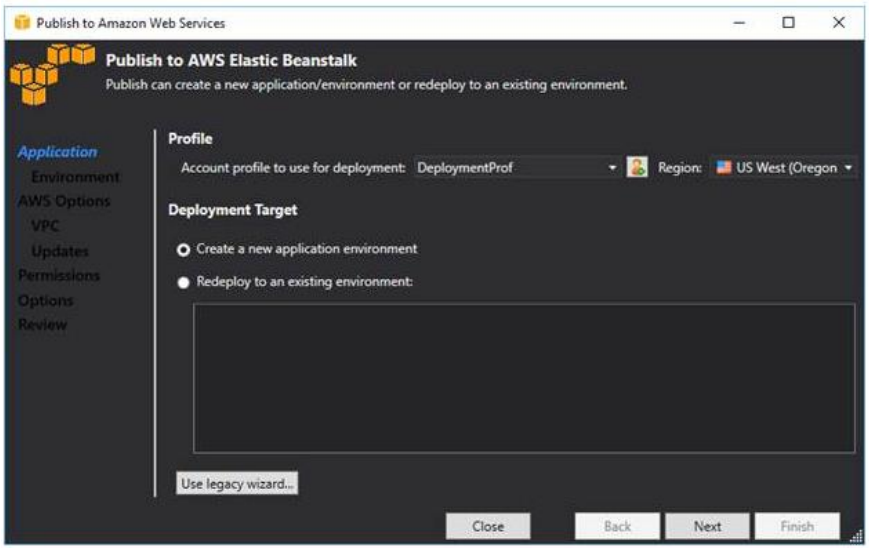
- Account number- optional.
- Account Type - Default is selected. You will see more if you've multiple account types.

Click OK once you've filled all the fields.



Now we have a profile attached with this app that is going to be used deploy our application to AWS.

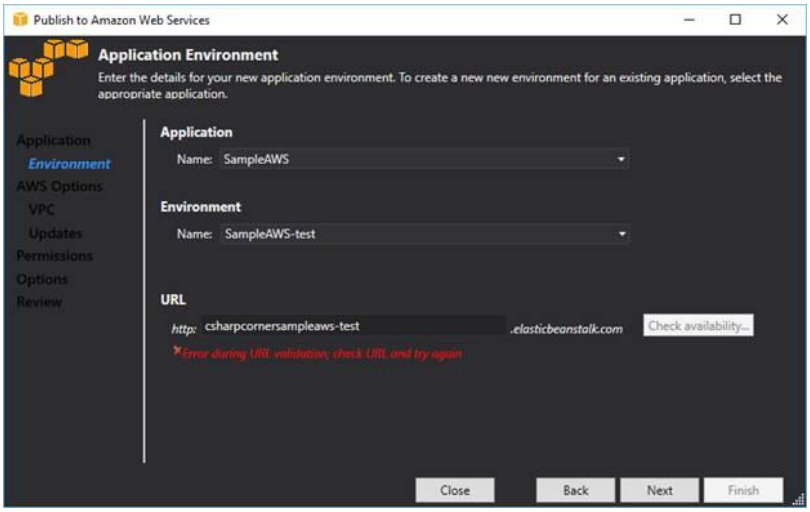
Now make sure the profile we have created is selected and Create a new application environment is selected. Click on next.



On this screen, you have to select the environment or you can enter your own as per requirement. I am choosing a test as I am just testing this.

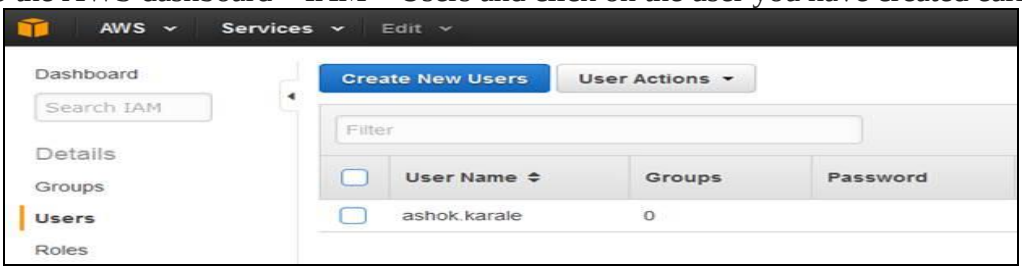
Now enter the URL. URL will be extended by AWS default URL .elasticbeanstalk.com. Your final public URL will be - YOURURL.elasticbeanstalk.com. This is similar to what we get for an Azure application.

You will notice that whatever unique URL you are entering, it will show an error, “Error during URL validation; check URL and try again”. It’s strange. I have spent so much time on this part and finally got the solution. I have intentionally let this error come to make you understand why this is happening.



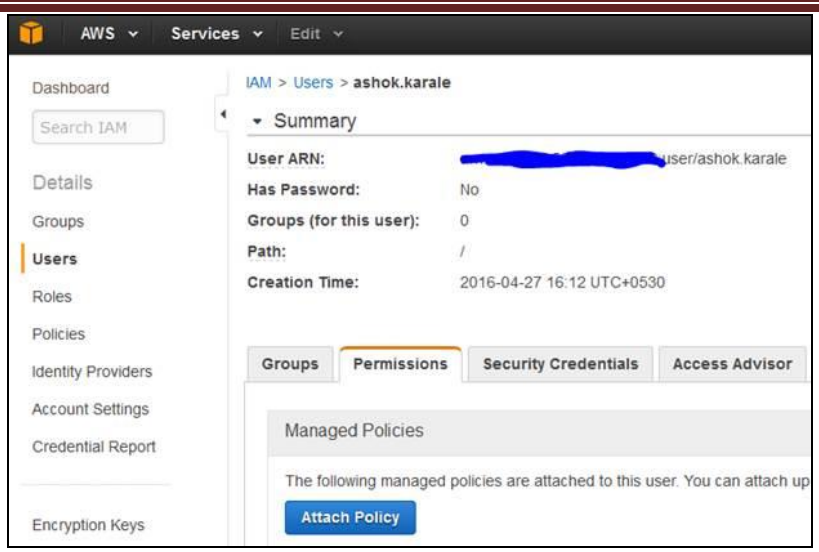
We are getting this error because of the missing settings while we created a new user on the AWS dashboard. So let’s fix it.

Go to the AWS dashboard – IAM – Users and click on the user you have created earlier.



We are getting the error because we have created a user but didn’t assign any permissions or policies which are required for every user we will create. So keep that in mind!

Once you clicked on the user you have created it will take you to the user page like below.

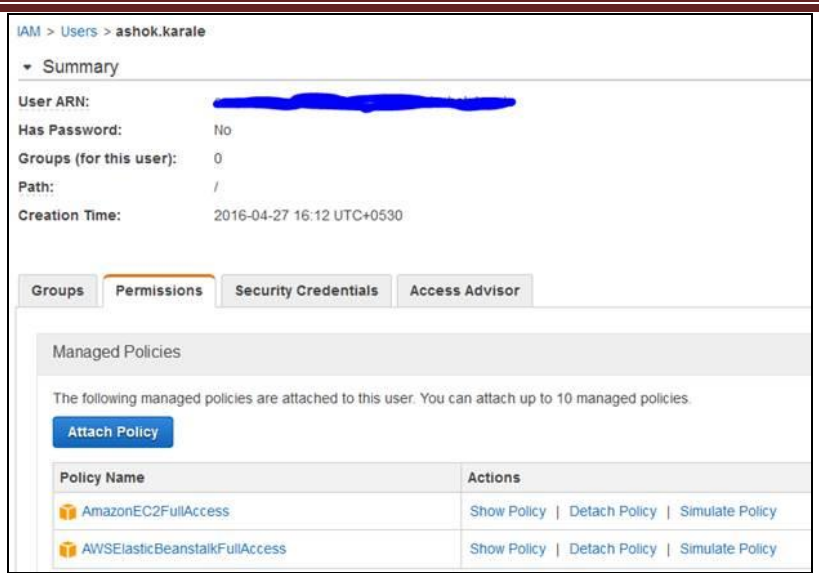


Click on Attach Policy.

We have to give full access to the user to AmazonEC2FullAccess & AWSElasticBeanstalkFullAccess.

Once you select both the policies, click on Attach Policy.

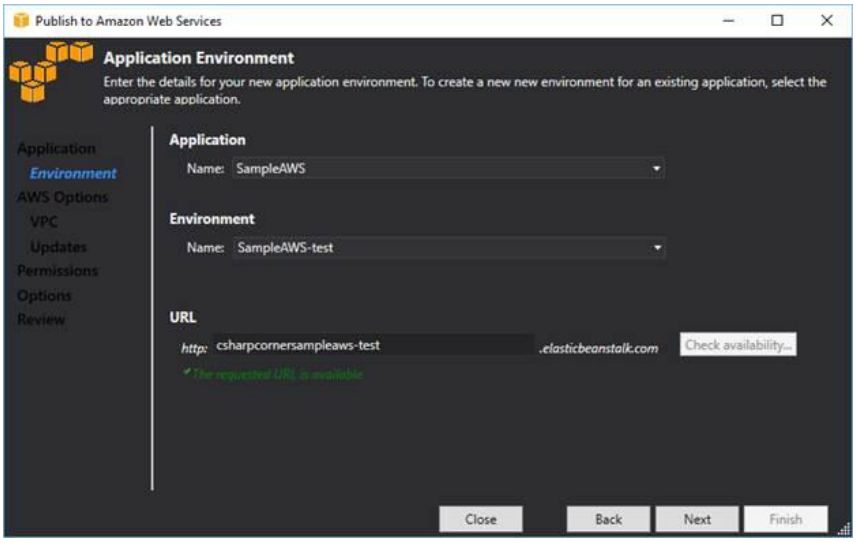
Finally, we have access to policies we need for the user.



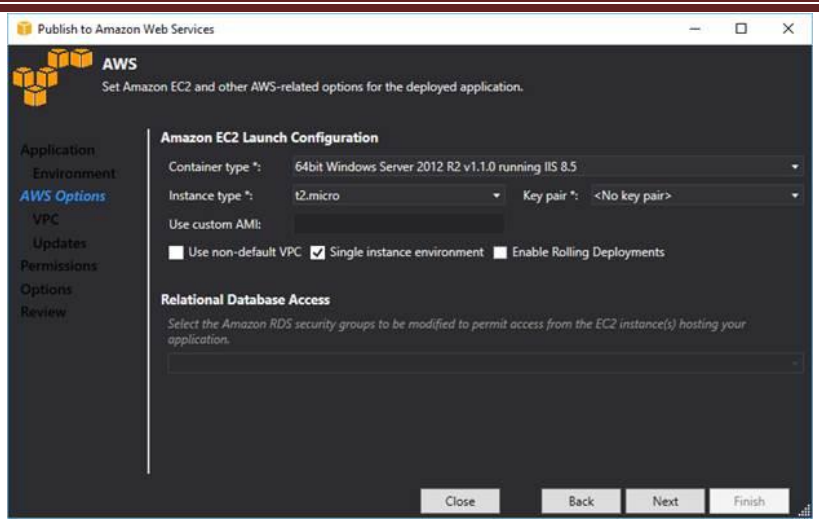
We are done with the AWS dashboard.

Now go back to Visual Studio screen where we had left and click on Back and Next buttons again.

You will notice URL is valid now.

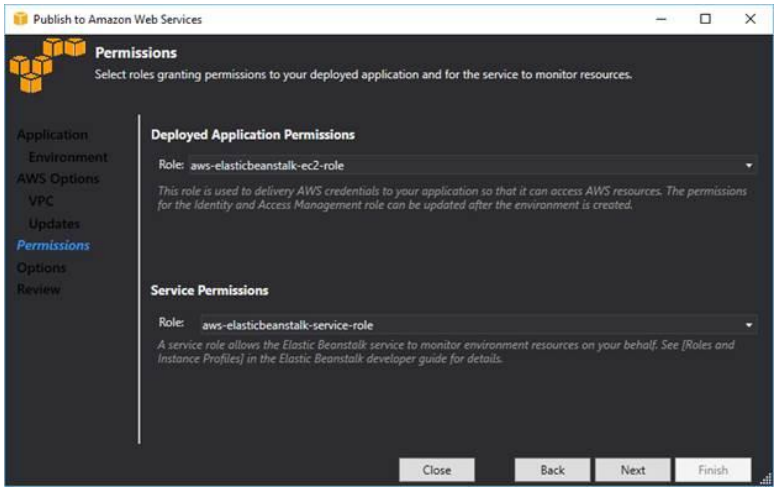


Just click on Next now.

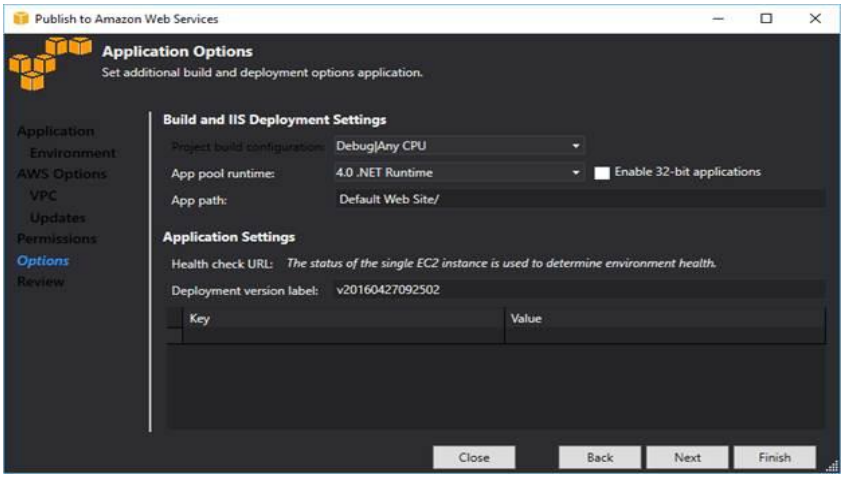


On this screen we are not doing anything for now. Just click Next.

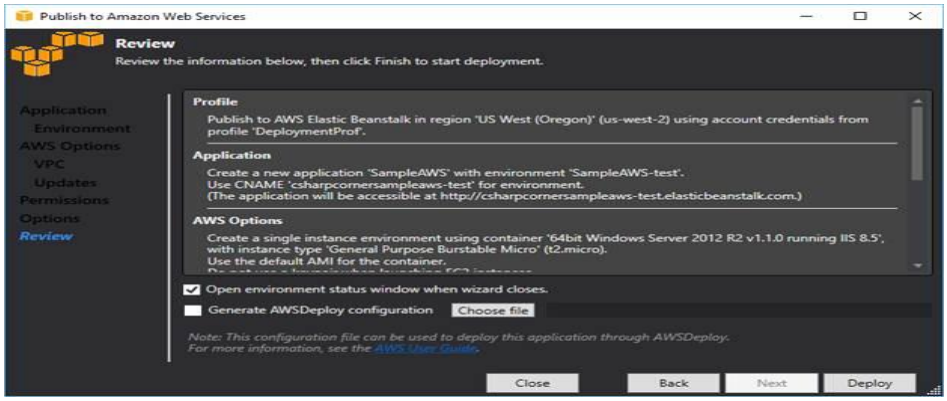
On the next screen also we are not going to do anything as of now. Just click Next.



again Click on Next.

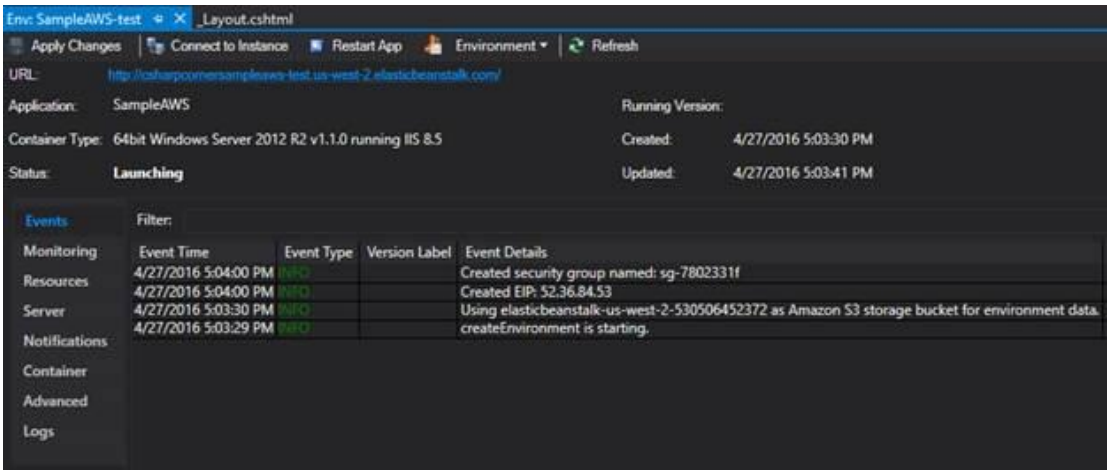


Click on Finish and on the next screen just click Deploy.

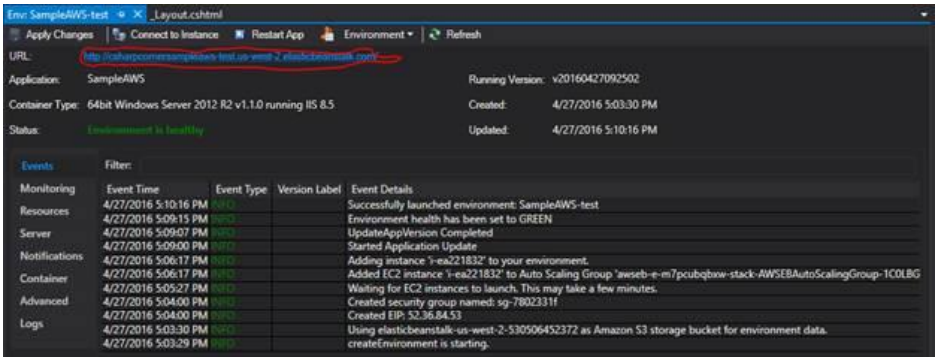


You are done. Now next everything will be done by AWS.

It will take time to finish the deployment. Every event will be shown to you like below.



When AWS complete its work, it will show all event with its status and show below results with a link to our application.

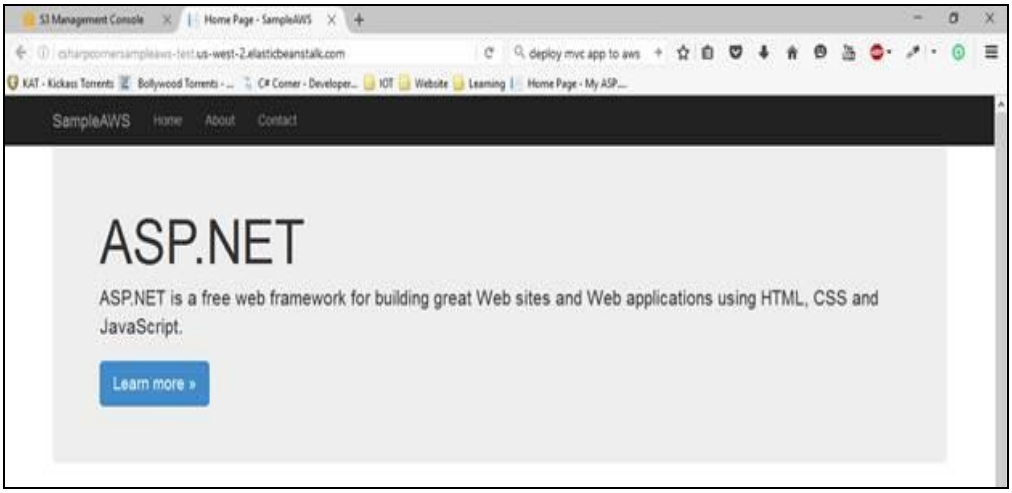


That is the URL you can type or copy in your browser to access the application we had just deployed.

Test the application

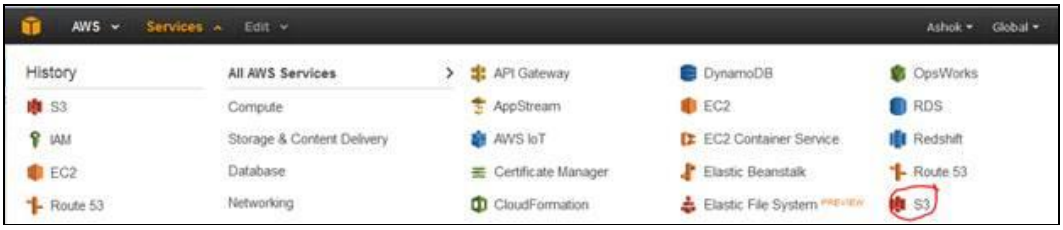
Go to the URL we had entered at the time of deployment.

It's working!!



So finally, we have deployed an ASP.NET application to AWS. You can find your files uploaded on AWS on AWS dashboard – S3.

Click on S3 below and you will see your application files there.



Result:

Done..!! You have launched your app successfully

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

ASSIGNMENT NO:7

AIM: Design and deploy a custom application(mini project) using salesforce cloud .

Objective:

To create an custom application on Salesforce Lightning platform. Software / Hardware Requirements: OS - Windows / Ubuntu, Google Chrome.

Outcome:

Developing custom Application.

Explanation:

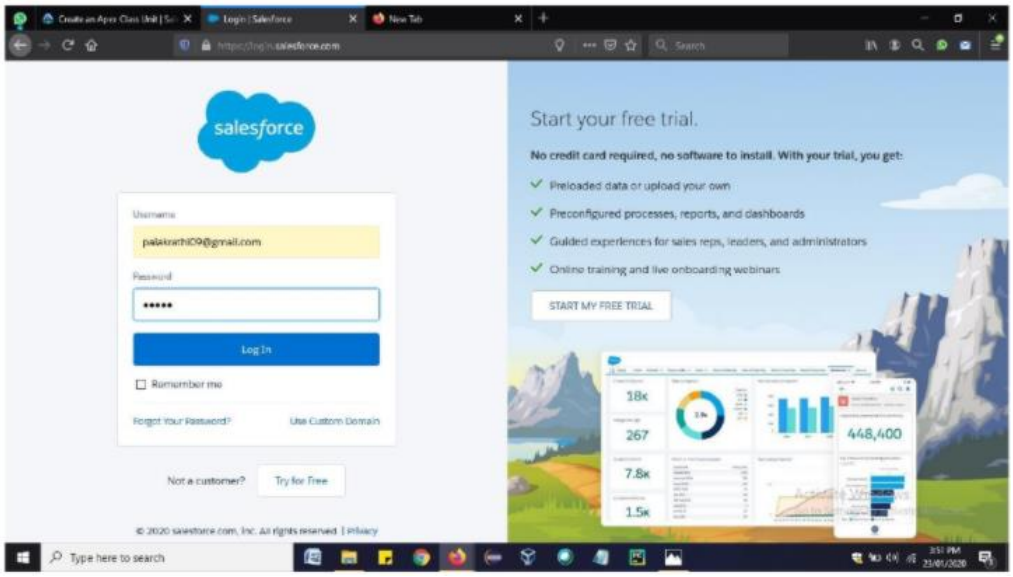
1. Salesforce:

Salesforce is a cloud computing service as a software (SaaS) company that specializes incustomer relationship management (CRM). Salesforce's services allow businesses to usecloud technology to better connect with customers, partners and potential customers. The software has become the number one for customer success and helps businessestrack customer activity, market to customers and many more services. Salesforce is acustomer relationship management solution that brings companies and customers together. It is one integrated CRM platform that gives all your departments — including marketing,sales, commerce, and service — a single, shared view of every customer.

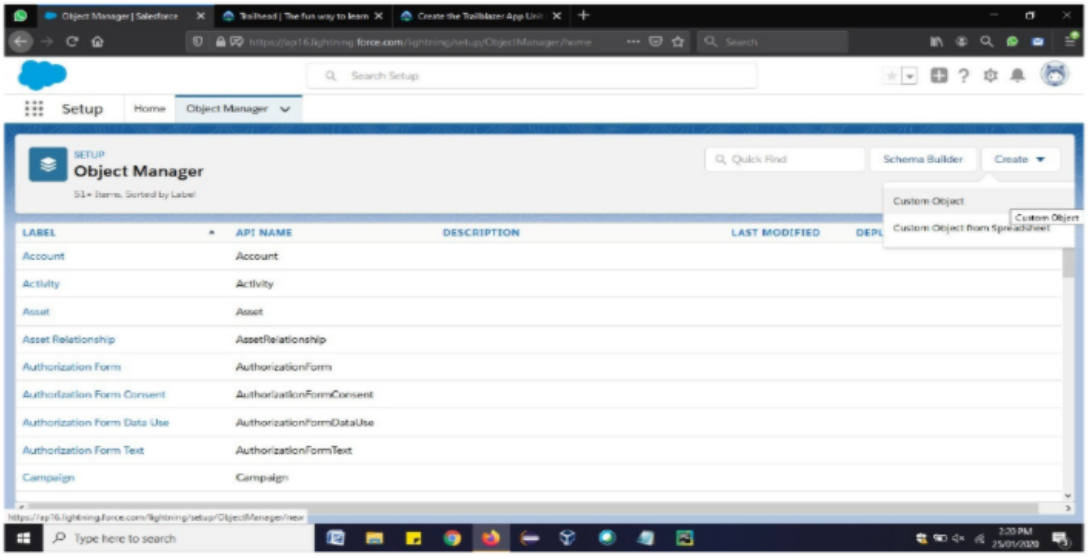
2. Salesforce Lightning Experience:

Salesforce Lightning Experience is simply referred to as “Lightning”. When working with lightning Salesforce we will learn about different Salesforce lightning topics like Lightning Login, Lightning App Builder, Lightning for Outlook, Salesforce Lightning Components,Lightning Sync and many more. Some of them are applicable in Lightning Experience only,but some others will work in both Lightning Experience and older Classic user interfaces.Lightning Sync is used to sync your user contacts and events between your email server with Salesforce.

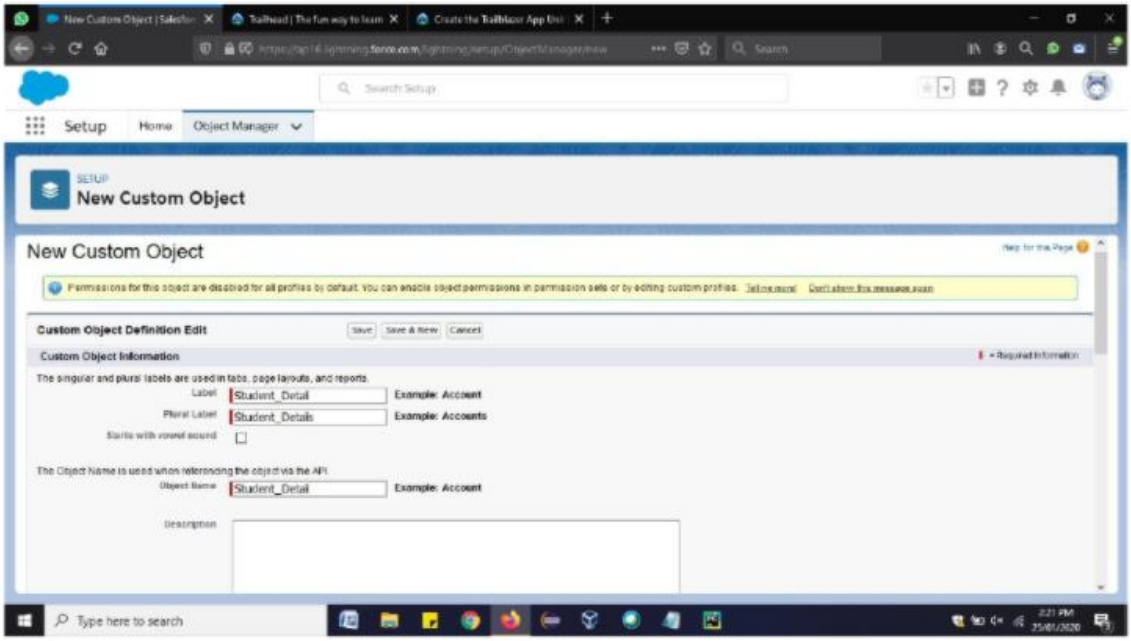
Step 1: Log into Salesforce Developer account

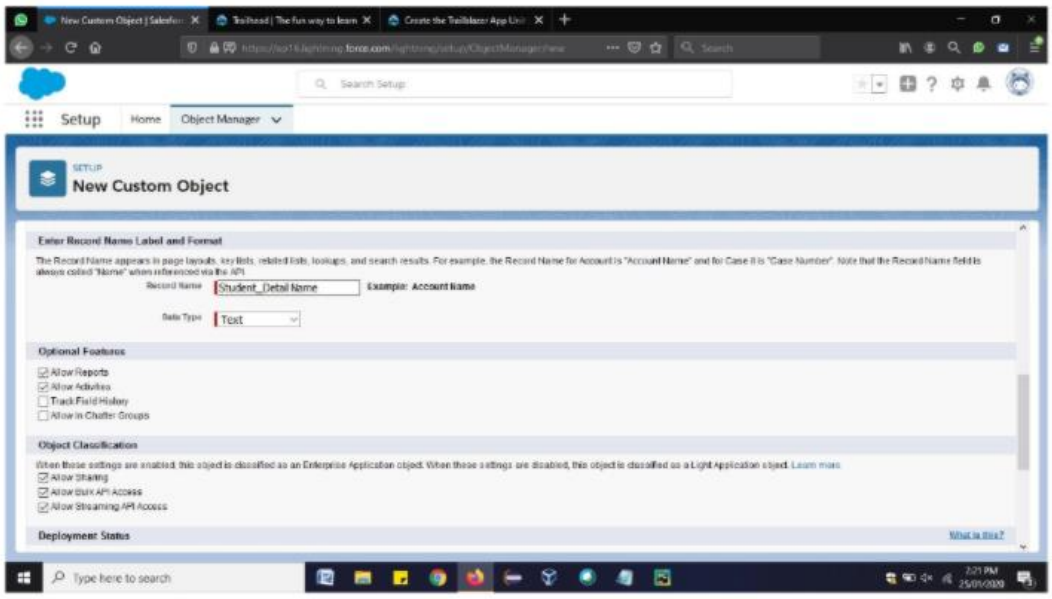


Step 2: Open Salesforce Lightning platform and click on Object Manager => Create => Custom Object

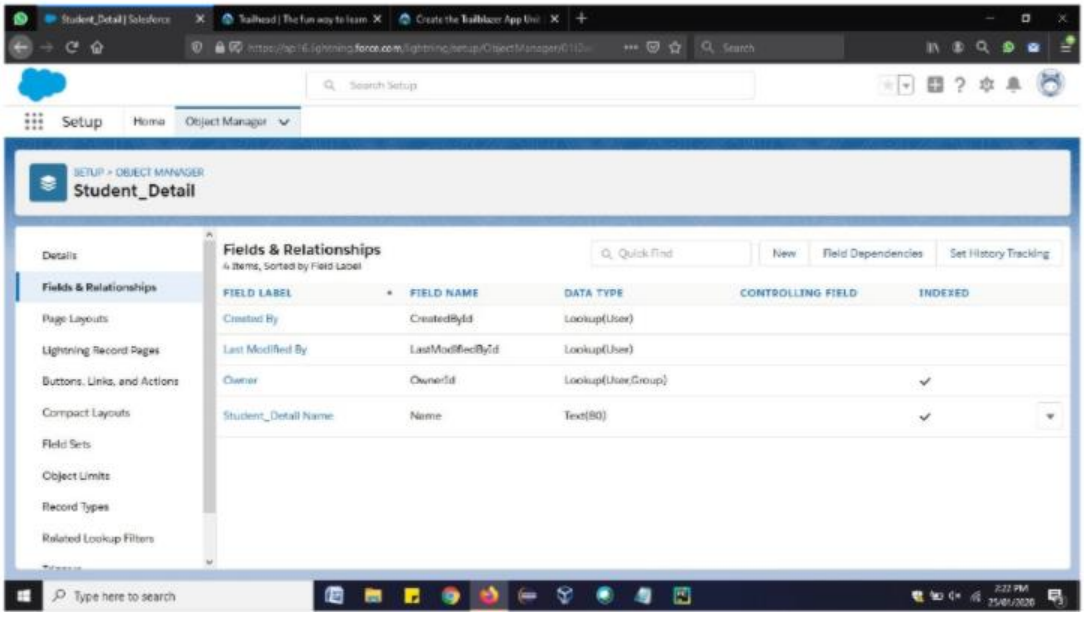


Step 3: Fill in the required fields and under Optional Features, select Allow Reports and Allow Activities. Click Save

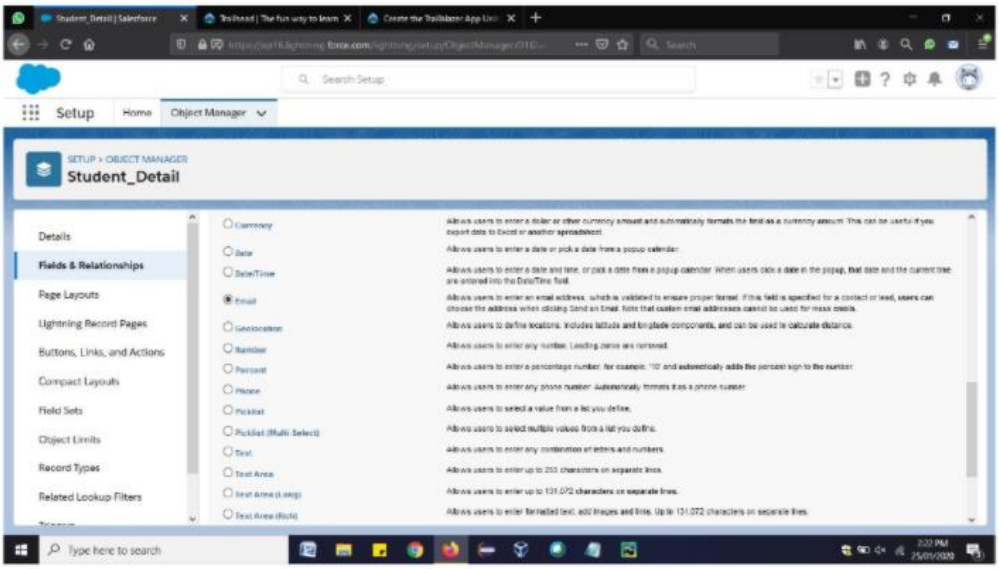




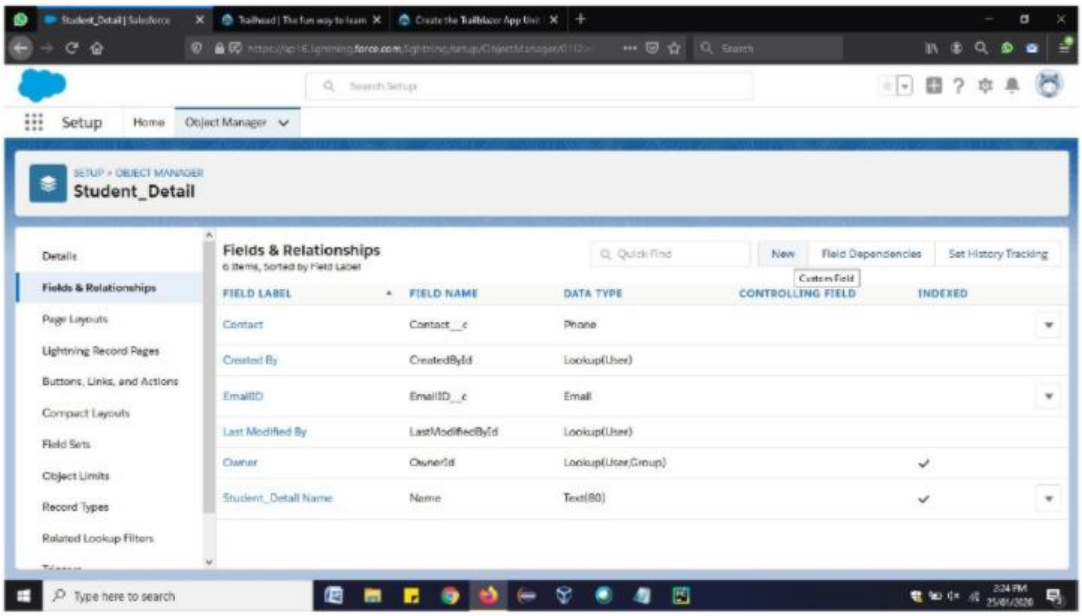
Step 4: Now, Click on Fields & Relations => New]



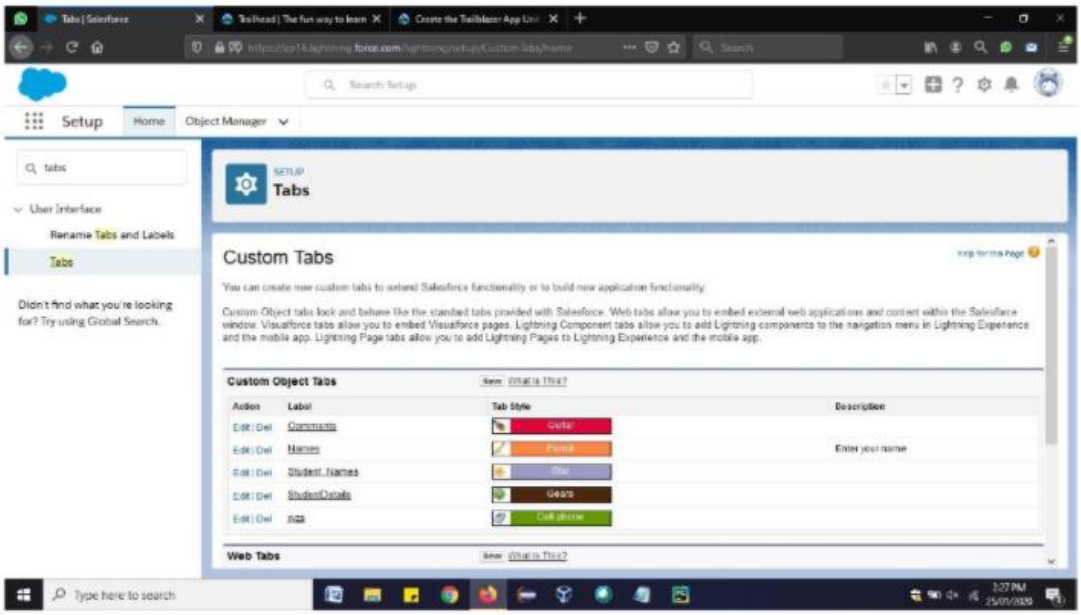
Step 5: Then select option “Email” and click Next-> Next -> Save.



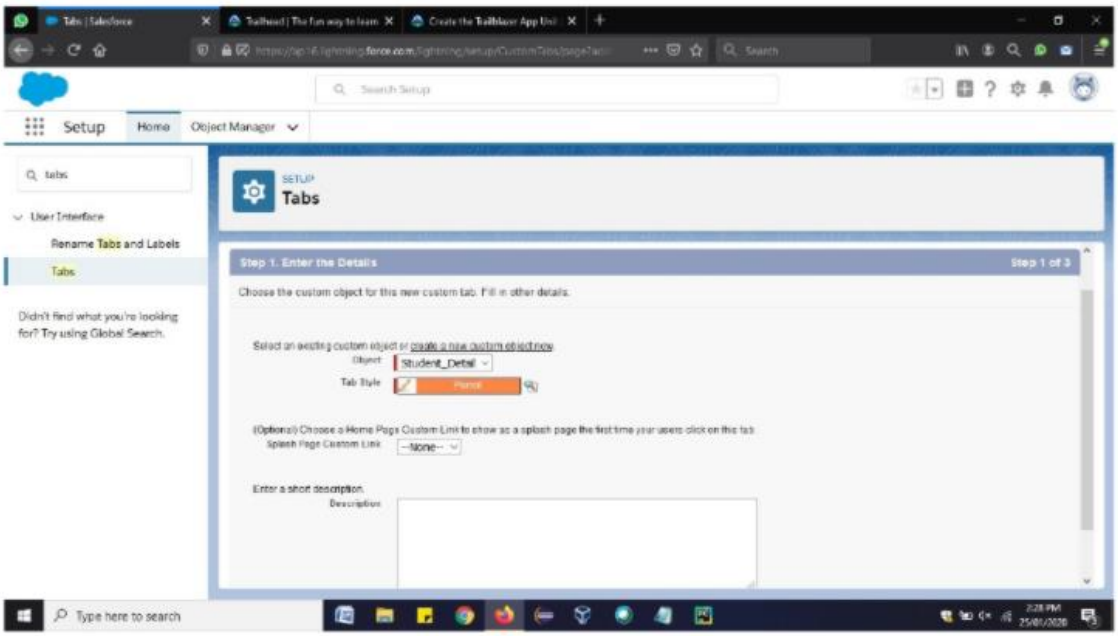
Step 6: Similarly. Repeat steps 4 and 5 to add more fields like Phone, Date of Birth. This is how the custom object will have the various fields.



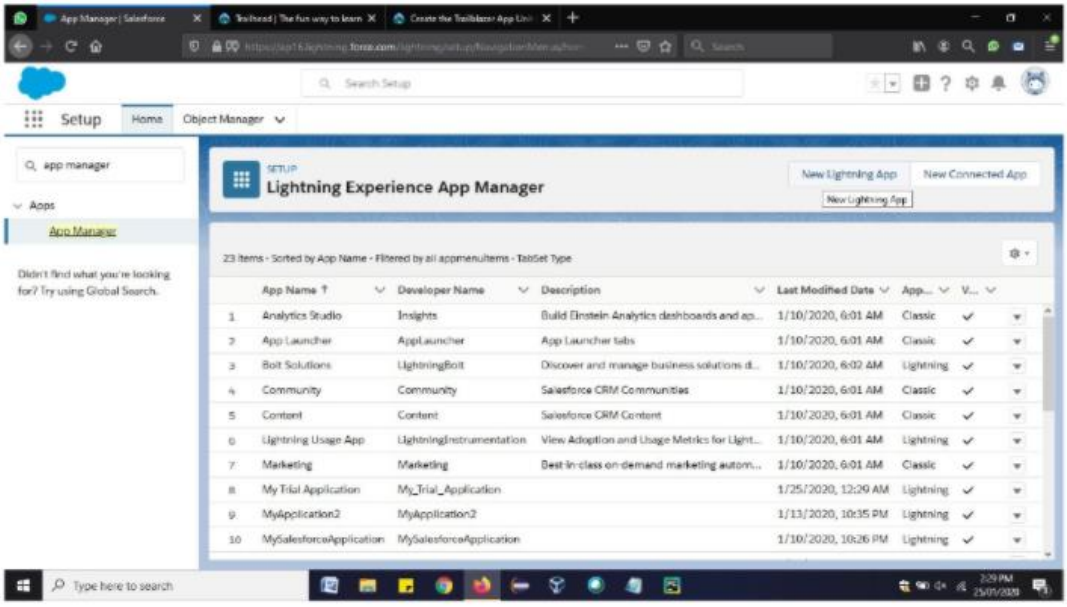
Step 7: Now go to Home => search for “Tabs”. Click on New.



Step 8: Enter the Object name and select any icon for tab style. Leave all defaults as it is. ClickNext, Next, and Save.

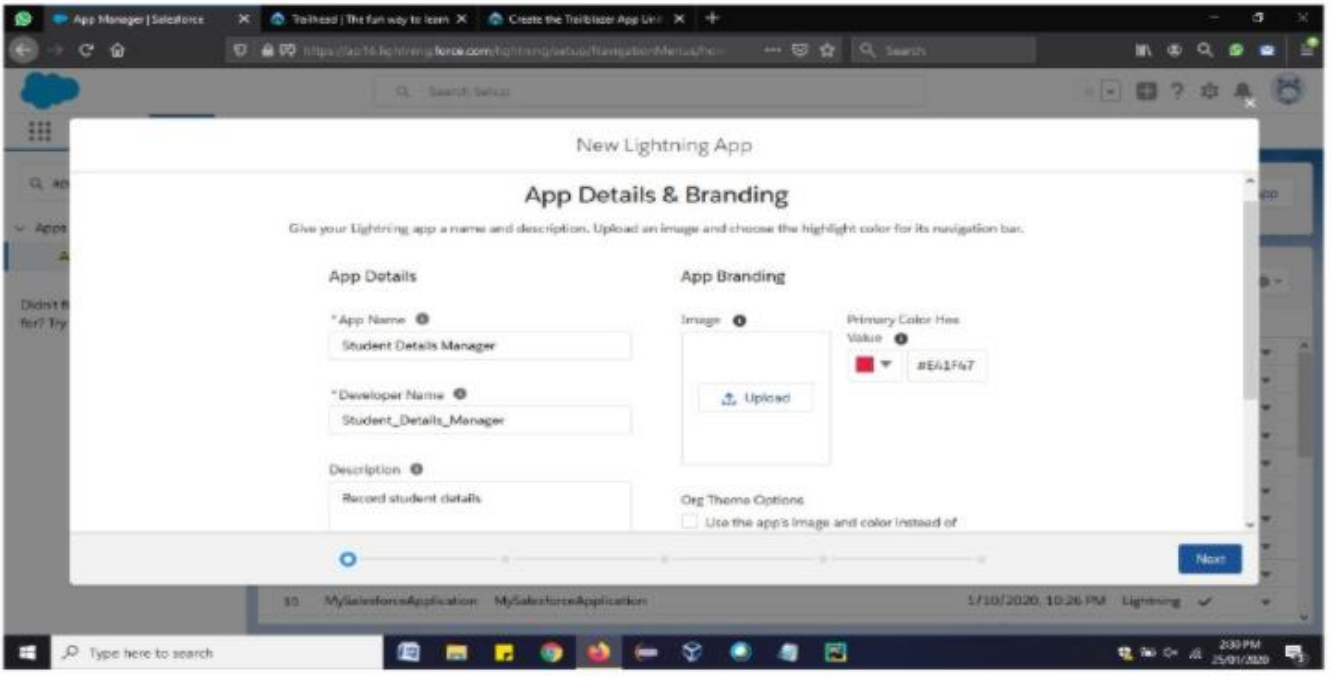


Step 9: In Setup, click Home. Enter "App Manager" in Quick Find and select App Manager. Click New Lightning App.

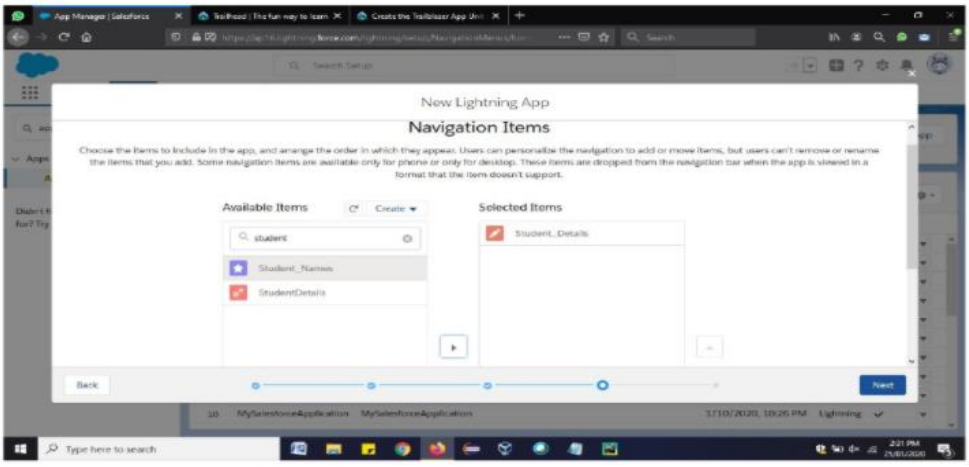


Step 10: Define the new Lightning app as follows:

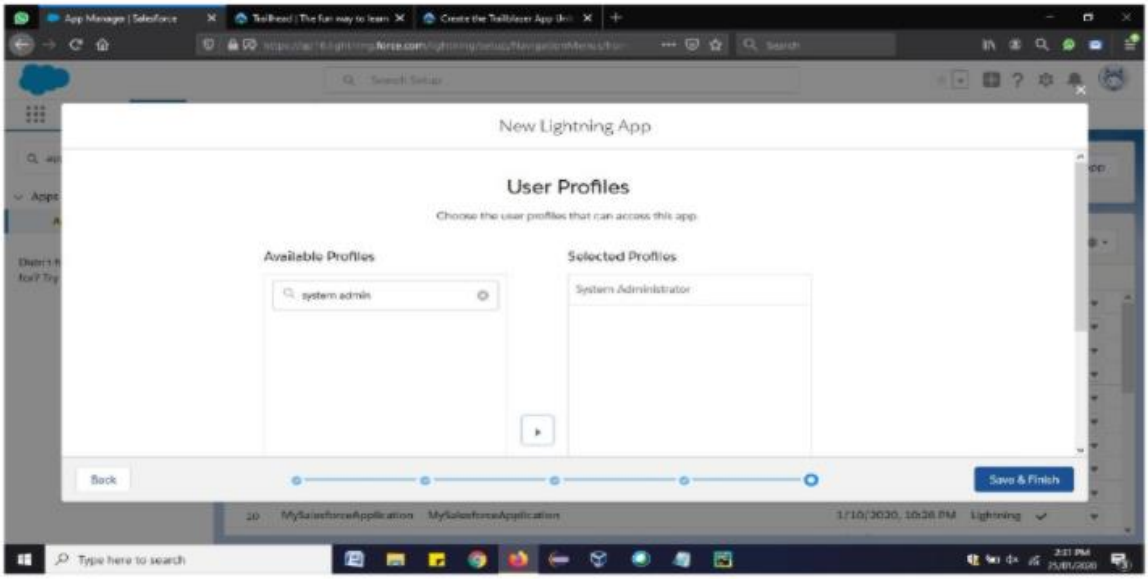
App Name: Student Details Manager Developer Name: Student_Details_Manager. Click Next



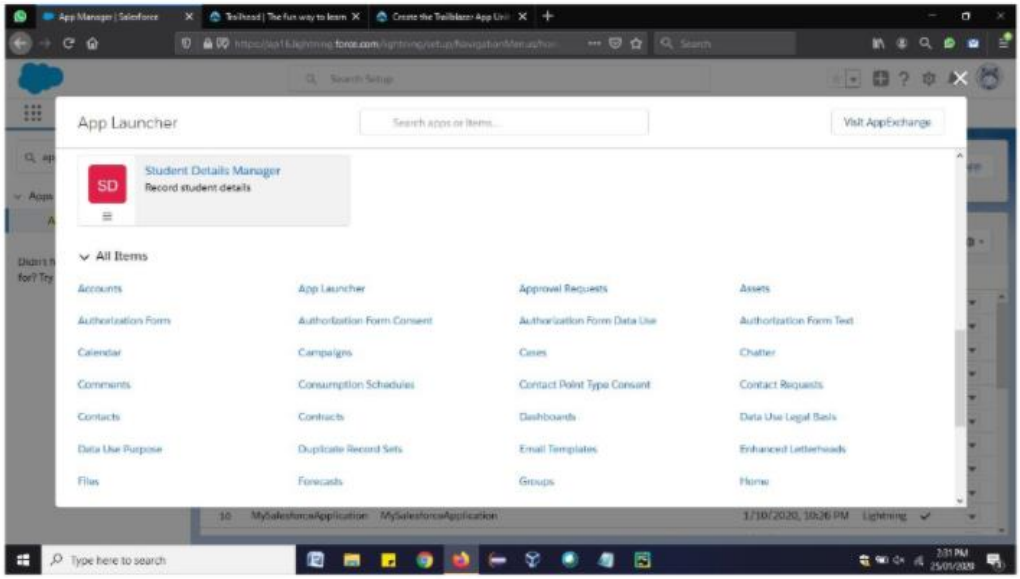
Step 11: On the App Options screen, leave the defaults as is and click Next. On the Utility Items screen, leave the defaults as is and click Next.
On the Navigation Items screen, select Student_Detail and move them to the Selected Items box. Then click Next.

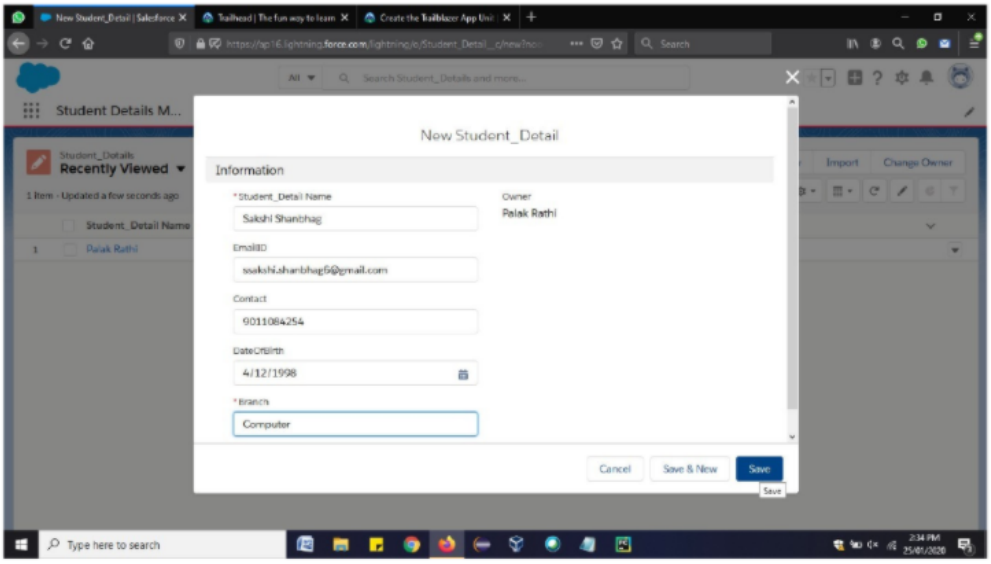


Step 12: On the Assign to User Profiles screen, select System Administrator and move it toSelected Profiles. Then click Save & Finish.

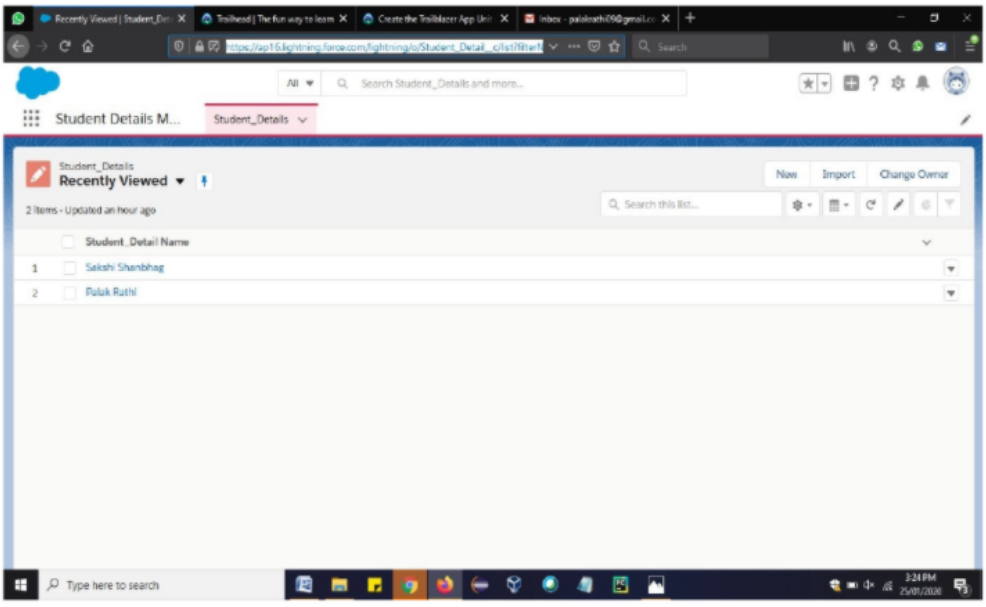


Step 13: Click on App launcher and Open the custom application created.

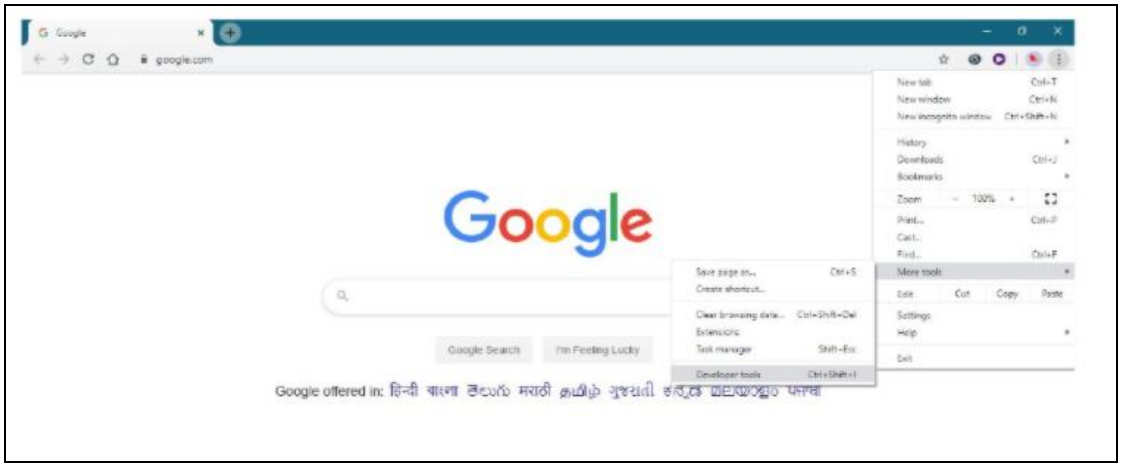


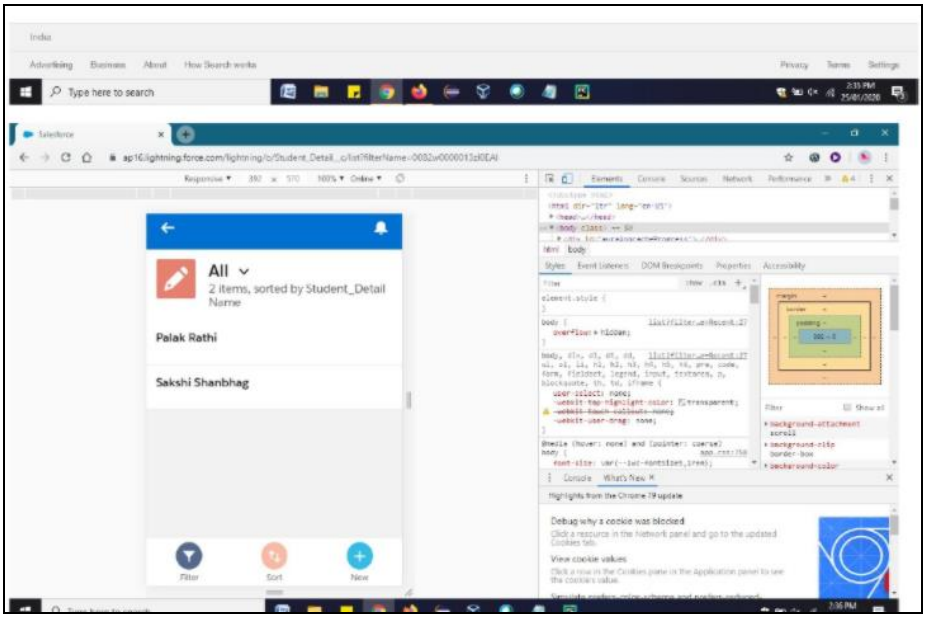


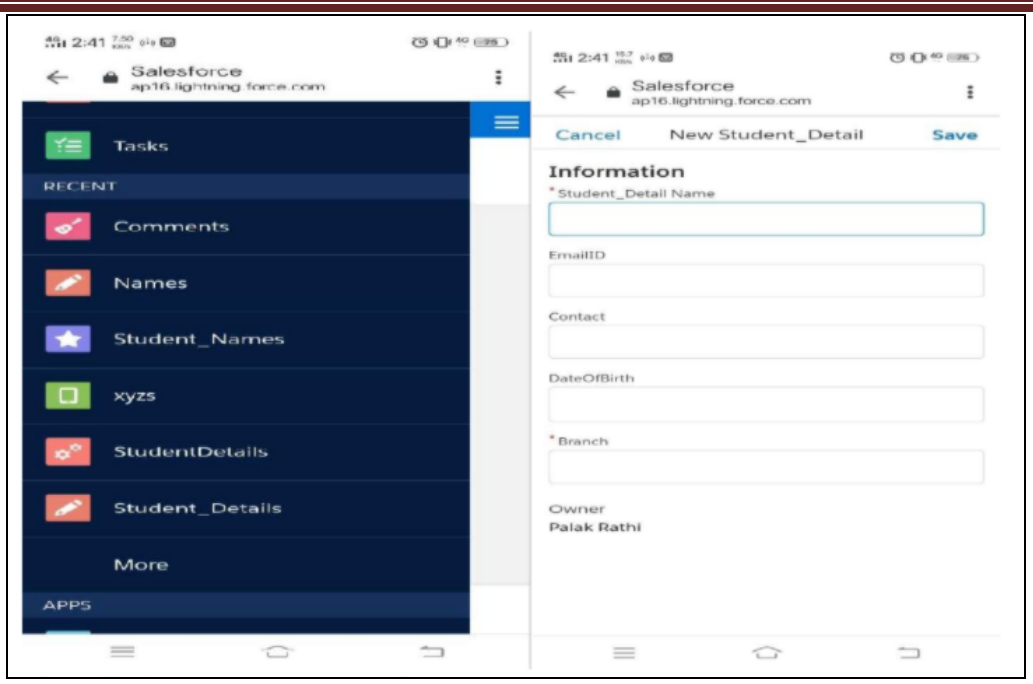
Step 14: Click on Student_Details => New => fill the specified details and copy the URL.

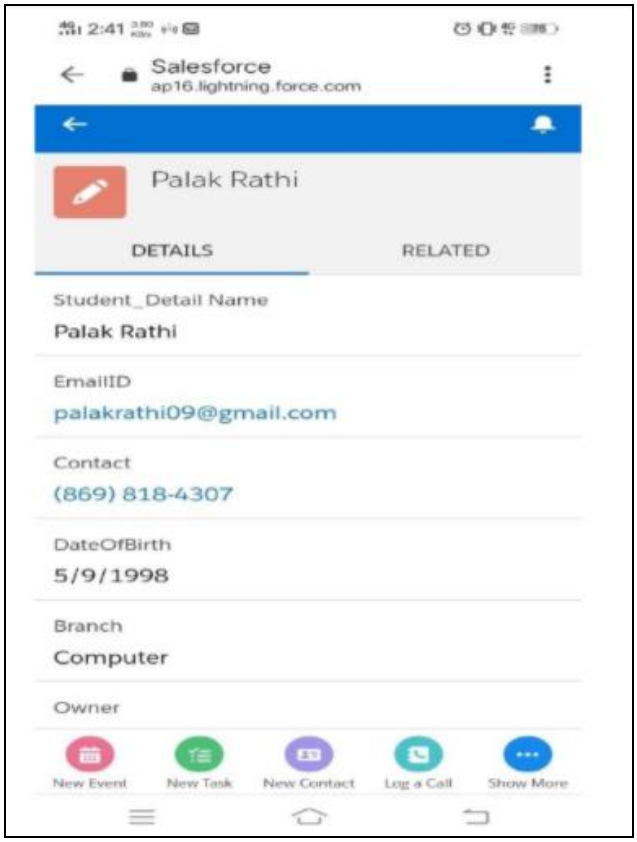


Step 15: Open Google Chrome new Tab => More Tools => Developer Tools and then paste the URL of the application, copied in the previous step.


Google
Google Search
I'm Feeling Lucky
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ


All
2 items, sorted by Student_Detail Name
Palak Rathi
Sakshi Shanbhag
Filter
Sort
New


Salesforce
ap16.lightning.force.com
Tasks
RECENT
Comments
Names
Student_Names
xyzs
StudentDetails
Student_Details
More
APPS
Salesforce
ap16.lightning.force.com
Cancel
New Student_Detail
Save
Information
*Student_Detail Name
EmailID
Contact
DateOfBirth
*Branch
Owner
Palak Rathi


Salesforce
ap16.lightning.force.com
Palak Rathi
DETAILS
RELATED
Student_Detail Name
Palak Rathi
EmailID
palakrathi09@gmail.com
Contact
(869) 818-4307
DateOfBirth
5/9/1998
Branch
Computer
Owner
New Event
New Task
New Contact
Log a Call
Show More

Result:

We have learnt to create custom application using salesforce Lightning platform.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	

ASSIGNMENT NO:8

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

Objective: By completing this tutorial, you'll accomplish the following:

- Configure the Firebase Authentication user interface.
- Obtain an Firebase ID token and verify it using server-side authentication.
- Store user credentials and associated data in Datastore.
- Query a database using the NDB client library.
- Deploy an app to App Engine.

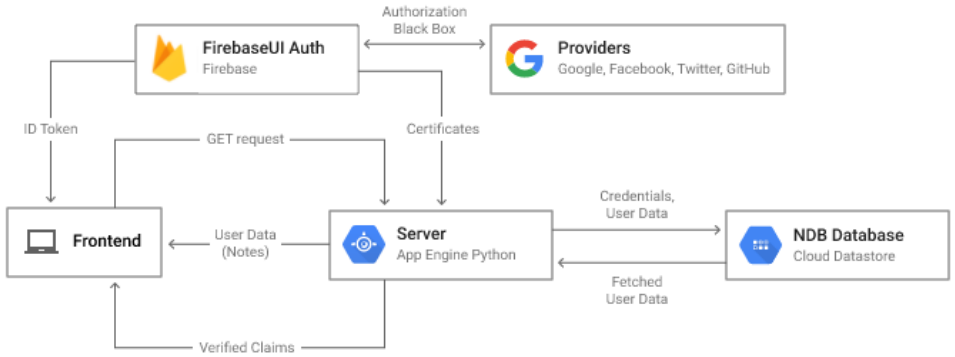
Outcome:

Deploying app using GAE.

Explanation:

Before you begin

The following diagram shows how the frontend and backend communicate with each other and how user credentials travel from Firebase to the database.



Step-1: Install Git,

```
sudo apt install git-all
```

Step-II: Install Python 2.7

Step-III: Sign in to your Google Account.
If you don't already have one, sign up for a new account.

Step-IV: Select or create a GCP project.

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

GO TO THE MANAGE RESOURCES PAGE

Step-V: Install and initialize the Cloud SDK.

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 Cloud SDK Configurations for specific commands to update a project with the gcloud tool.

Step-VI: Cloning the sample app

To download the sample to your local machine:

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.

Navigate to the directory that contains the sample code:

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

Step-VII: Adding the Firebase Authentication user interface

To configure FirebaseUI and enable identity providers:

1. 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 Platform 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:

GO to given Pathappengine/standard/firebase/firenotes/frontend/main.js

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>"
};
```

2. Edit the backend/app.yaml file and enter your Firebase project ID in the environment variables:

appengine/standard/firebase/firenotes/backend/app.yaml

VIEW ON GITHUB

```
runtime: python27
api_version: 1
threadsafe: true
service: backend

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

env_variables:
  # Replace with your Firebase project ID.
  FIREBASE_PROJECT_ID: '<PROJECT_ID>'
```

3. In the frontend/main.js file, configure the FirebaseUI login widget by selecting which providers you want to offer your users.

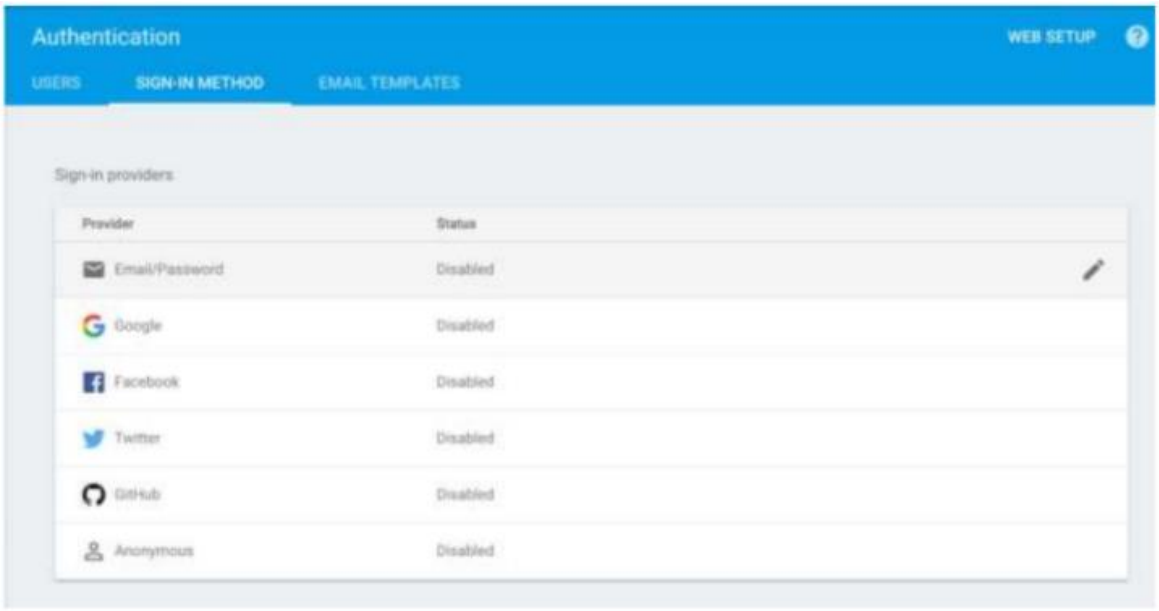
appengine/standard/firebase/firenotes/frontend/main.js

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.
```

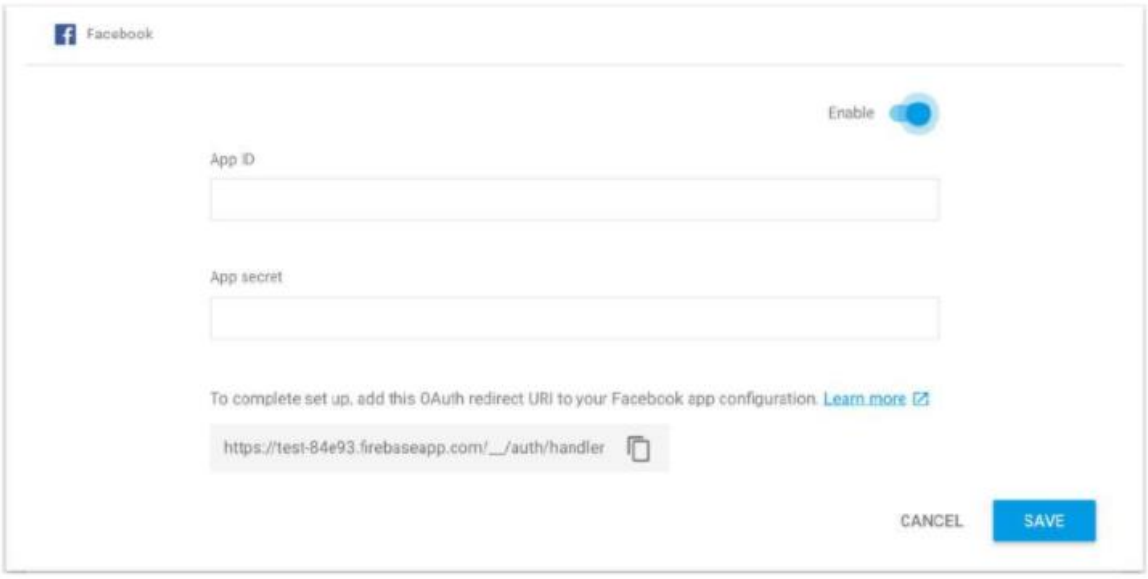
```
      firebase.auth.GoogleAuthProvider.PROVIDER_ID,
      firebase.auth.FacebookAuthProvider.PROVIDER_ID,
      firebase.auth.TwitterAuthProvider.PROVIDER_ID,
      firebase.auth.GithubAuthProvider.PROVIDER_ID,
      firebase.auth.EmailAuthProvider.PROVIDER_ID
    ],
    // Terms of service url
    'tosUrl': '<your-tos-url>',
  };

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

3. 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.

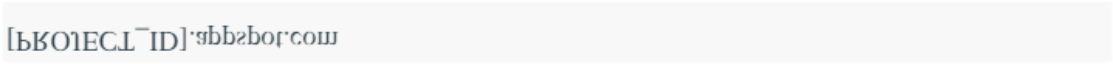


- Toggle the Enable button and, for third-party identity providers, enter the provider ID and secret from the provider's developer site.

The Firebase docs give specific instructions in the "Before you begin" sections of theFacebook, Twitter, and GitHub guides. After enabling a provider, click Save.



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



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

Reference:

https://cloud.google.com/appengine/docs/standard/python/authenticating-users-firebase-appengine

Result:

Successfully app deployed using Google App Engine.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	