

Assignment No. 1

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

Objective:

- 1) To install Google App Engine.
- 2) To create and run a simple hello world web application on Google Cloud SDK Shell.

Theory:

What is Google App Engine?

Google App Engine (GAE) is a platform-as-a-service product that provides web app developers and enterprises with access to Google's scalable hosting and tier 1 internet service. GAE requires that applications be written in Java or Python, store data in Google Bigtable and use the Google query language. Noncompliant applications require modification to use GAE. GAE provides more infrastructure than other scalable hosting services, such as Amazon Elastic Compute Cloud (EC2). GAE also eliminates some system administration and development tasks to make writing scalable applications easier.

Google provides GAE free up to a certain amount of use for the following resources:

- processor (CPU)
- storage
- application programming interface (API) calls
- concurrent requests

How is GAE used?

GAE is a fully managed, serverless platform that is used to host, build and deploy web applications. Users can create a GAE account, set up a software development kit and write application source code. They can then use GAE to test and deploy the code in the cloud.

One way to use GAE is building scalable mobile application back ends that adapt to workloads as needed. Application testing is another way to use GAE. Users can route traffic to different application versions to A/B test them and see which version performs better under various workloads.

What are GAE's key features?

Key features of GAE include the following:

- 1) API selection: GAE has several built-in APIs, including the following five:
 - Blobstore for serving large data objects;
 - GAE Cloud Storage for storing data objects;
 - Page Speed Service for automatically speeding up webpage load times;

- URL Fetch Service to issue HTTP requests and receive responses for efficiency and scaling;
 - Memcache for a fully managed in-memory data store.
- 2) Managed infrastructure: Google manages the back-end infrastructure for users. This approach makes GAE a serverless platform and simplifies API management.
 - 3) Several programming languages: GAE supports a number of languages, including GO, PHP, Java, Python, NodeJS, .NET and Ruby. It also supports custom runtimes.
 - 4) Support for legacy runtimes: GAE supports legacy runtimes, which are versions of programming languages no longer maintained. Examples include Python 2.7, Java 8 and Go 1.11.
 - 5) Application diagnostics: GAE lets users record data and run diagnostics on applications to gauge performance.
 - 6) Security features: GAE enables users to define access policies with the GAE firewall and managed Secure Sockets Layer/Transport Layer Security certificates for free.

Installation of Google App Engine –

The Google Cloud CLI works on Windows 8.1 and later and Windows Server 2012 and later.

- 1) Download the Google Cloud CLI installer from the following link - <https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>
- 2) Launch the installer and follow the prompts. The installer is signed by Google LLC. If you're using a screen reader, check the Turn on screen reader mode checkbox. This option configures gcloud to use status trackers instead of Unicode spinners, display progress as a percentage, and flatten tables. For more information, see the Accessibility features guide.
- 3) Cloud SDK requires Python; supported versions are Python 3 (preferred, 3.5 to 3.8) and Python 2 (2.7.9 or later). By default, the Windows version of Cloud SDK comes bundled with Python 3 and Python 2. To use Cloud SDK, your operating system must be able to run a supported version of Python. The installer installs all necessary dependencies, including the needed Python version. While Cloud SDK installs and manages Python 3 by default, you can use an existing Python installation if necessary by unchecking the option to Install Bundled Python. See gcloud topic startup to learn how to use an existing Python installation.
- 4) After installation is complete, the installer gives you the option to create Start Menu and Desktop shortcuts, start the Google Cloud CLI shell, and configure the gcloud CLI. Make sure that you leave the options to start the shell and configure your installation selected. The installer starts a terminal window and runs the gcloudinit command.
- 5) The default installation doesn't include the App Engine extensions required to deploy an application using gcloud commands. These components can be installed using the gcloud CLI component manager.

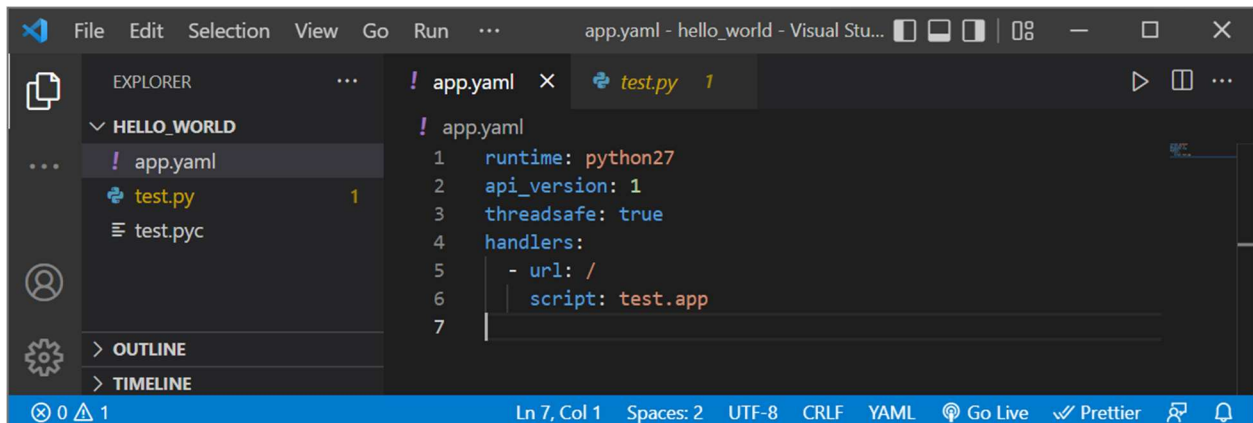
Troubleshooting tips:

- 1) If your installation is unsuccessful due to the find command not being recognized, ensure your PATH environment variable is set to include the folder containing find. Usually, this is C:\WINDOWS\system32;.
- 2) If you uninstalled the gcloud CLI, you must reboot your system before installing the gcloud CLI again.

3) If unzipping fails, run the installer as an administrator.

Code –

app.yaml

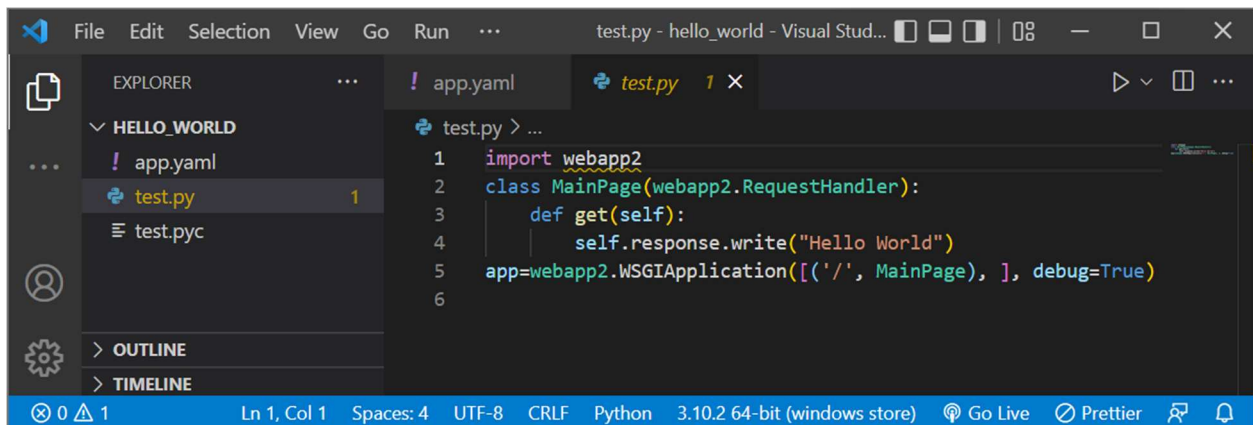


The screenshot shows the Visual Studio Code editor with the 'app.yaml' file open. The Explorer sidebar on the left shows a project named 'HELLO_WORLD' containing 'app.yaml', 'test.py', and 'test.pyc'. The main editor area displays the following YAML code:

```
1 runtime: python27
2 api_version: 1
3 threadsafe: true
4 handlers:
5   - url: /
6     script: test.app
7
```

The status bar at the bottom indicates the cursor is at Line 7, Column 1, with 2 spaces, UTF-8 encoding, CRLF line endings, and the file is a YAML document.

test.py



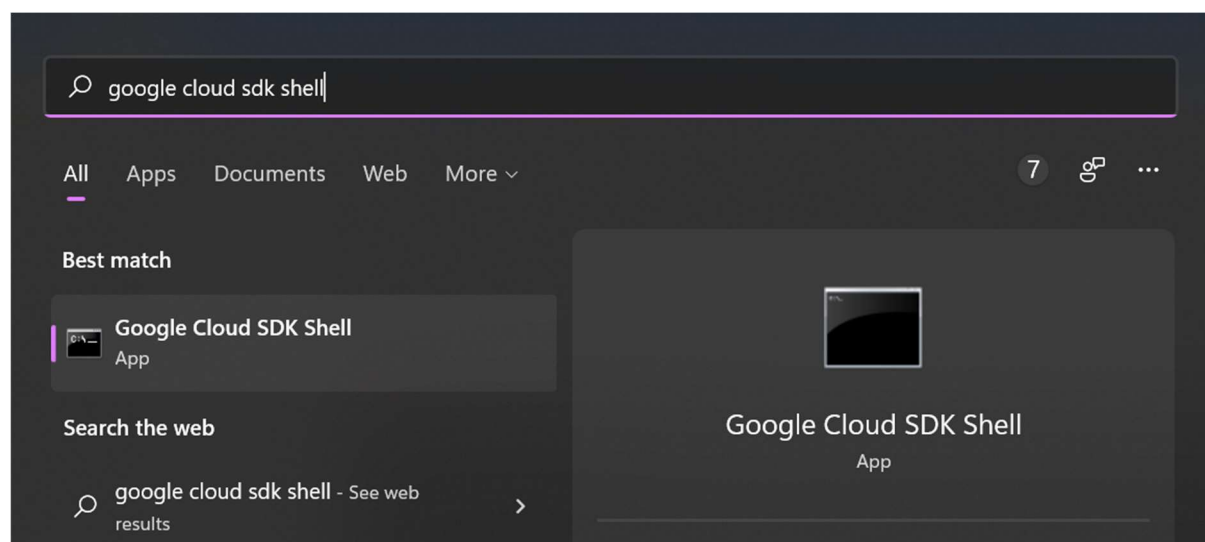
The screenshot shows the Visual Studio Code editor with the 'test.py' file open. The Explorer sidebar on the left shows the same project structure. The main editor area displays the following Python code:

```
1 import webapp2
2 class MainPage(webapp2.RequestHandler):
3     def get(self):
4         self.response.write("Hello World")
5 app=webapp2.WSGIApplication([('/', MainPage)], debug=True)
6
```

The status bar at the bottom indicates the cursor is at Line 1, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the file is a Python 3.10.2 64-bit script.

Step-by-step procedure to launch the hello world app on Google App Engine –

Step 1 – Open ‘Google Cloud SDK Shell’

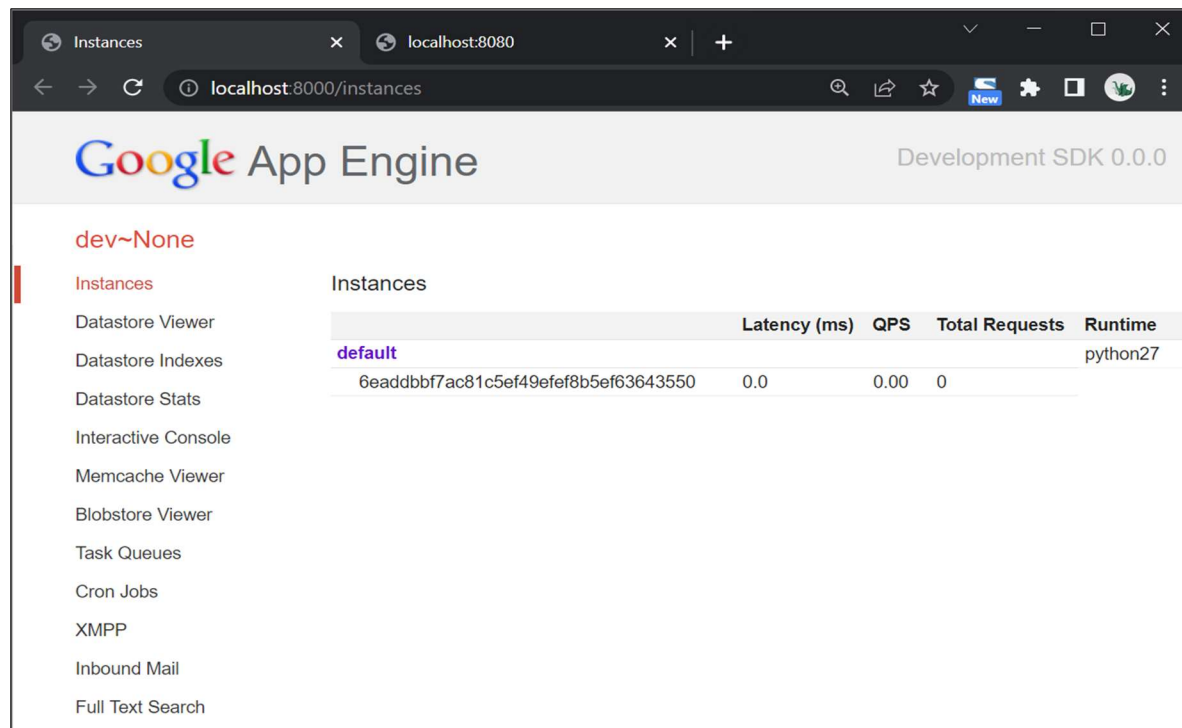


Step 2 – Run the following command,

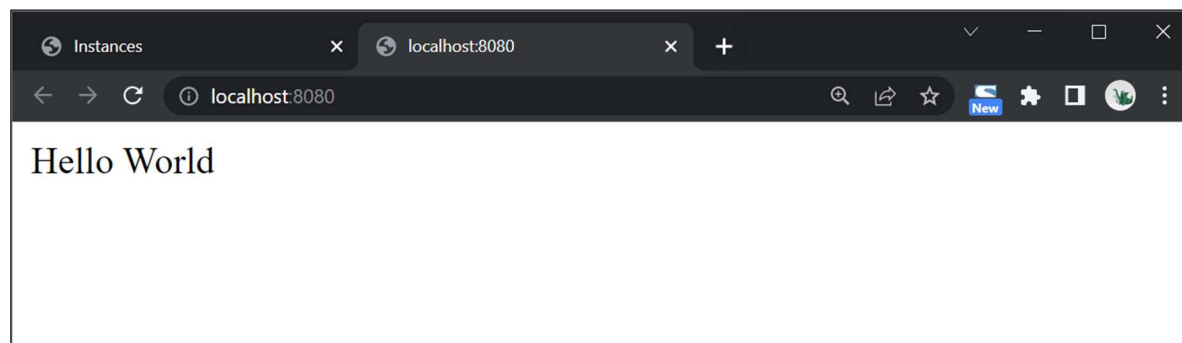
```
py google-cloud-sdk/bin/dev_appserver.py 'Path to the hello world app'
```

```
C:\Users\lenovo\AppData\Roam x + v - □ ×
Welcome to the Google Cloud CLI! Run "gcloud -h" to get the list of available commands.
---
C:\Users\lenovo\AppData\Local\Google\Cloud SDK>py google-cloud-sdk/bin/dev_appserver.py C:\Users\lenovo\Desktop\
hello_world
INFO 2022-04-07 15:06:12,234 devappserver2.py:316] Skipping SDK update check.
WARNING 2022-04-07 15:06:13,285 simple_search_stub.py:1196] Could not read search indexes from c:\users\lenovo\
appdata\local\temp\appengine.None\search_indexes
INFO 2022-04-07 15:06:13,289 <string>:384] Starting API server at: http://localhost:50168
INFO 2022-04-07 15:06:13,296 dispatcher.py:281] Starting module "default" running at: http://localhost:8080
INFO 2022-04-07 15:06:13,301 admin_server.py:150] Starting admin server at: http://localhost:8000
```

Step 3 – Now go to <http://localhost:8000> in your browser. You will see the following dashboard,



Step 4 – Click on the default button. You will be redirected to <http://localhost:8080>



Conclusion: Thus we have successfully created and run a simple hello world web application on Google App Engine.

Assignment -2

Aim : To use Google App Engine to launch a web application.

Objective : To learn design and deployment process involved in creating cloud based application.

Outcome : To design and deploy Web application in cloud environment.

Theory:

Introduction:

Google App Engine is a web application hosting service. By “web application,” we mean an application or service accessed over the Web, usually with a web browser: storefronts with shopping carts, social networking sites, multiplayer games, mobile applications, survey applications, project management, collaboration, publishing, and all the other things we’re discovering are good uses for the Web. App Engine can serve traditional website content too,

Google App Engine:

It is a platform-as-a-service (PaaS) Cloud computing platform that is fully managed and uses inbuilt services to run your apps. You can start development almost instantly after downloading the software development kit (SDK). You can go on to the developer’s guide right away when you click on the language you wish to develop your app in.

Advantages of Google App Engine:

- ☐ Infrastructure for Security
- ☐ Scalability
- ☐ Performance and Reliability
- ☐ Cost Savings
- ☐ Platform Independence

Installation of Google-cloud-sdk shell

The App Engine SDK allows you to run Google App Engine Applications on your local computer. It simulates the run-time environment of the Google App Engine infrastructure.

Pre--Requisites:Python3.10.4

If you don't already have Python 3.10.4 installed in your computer, download and Install Python 3.10.4 from:

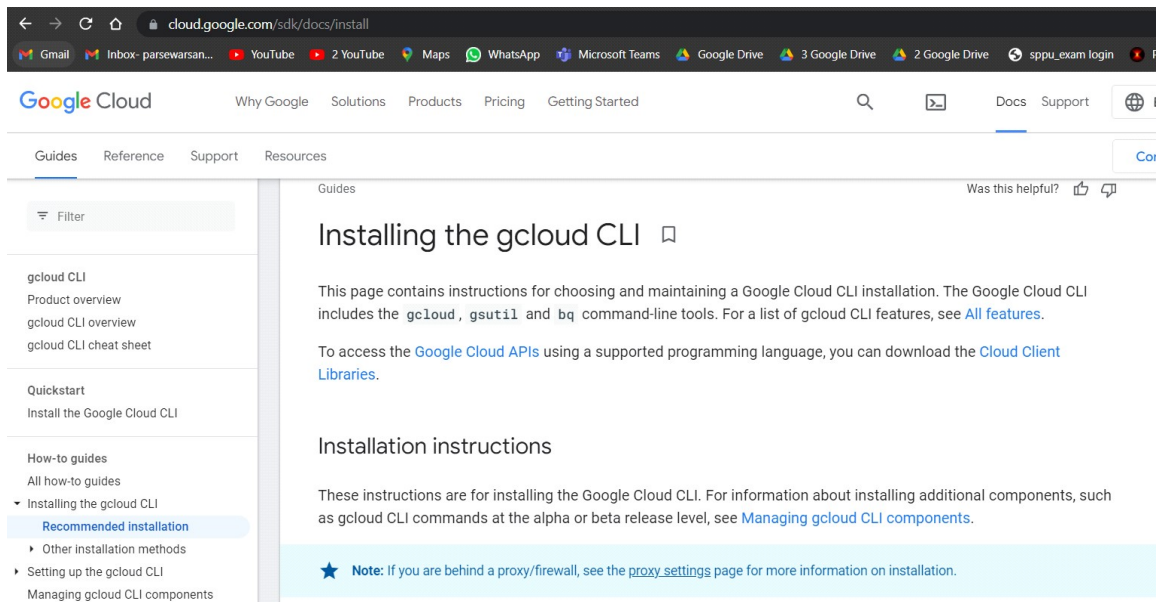
<https://www.python.org/ftp/python/3.10.4/python-3.10.4-amd64.exe>

Download and Install

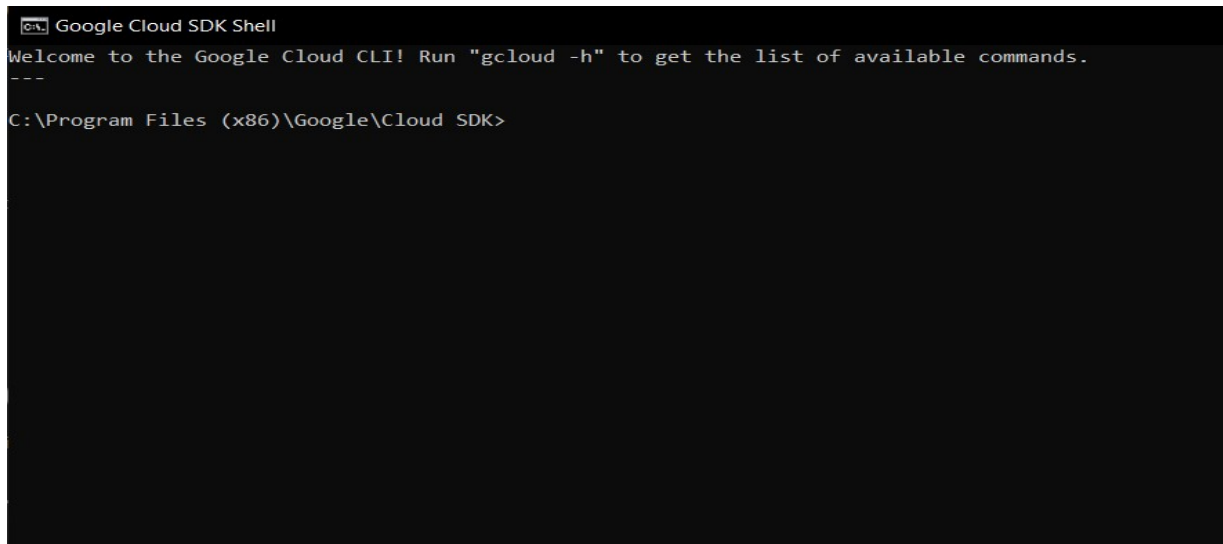
You can download the Google App Engine SDK by going to:

<https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>

and download the appropriate install package.



Window of Google cloud sdk shell.



Making your First Application

Make a folder for your Google App Engine applications' am going to make the Folder on my Desktop called "cc-ass2" –the path to this folder is:

C:\Users\cheta\Desktop\cc-ass2

And then make a sub---folder in withincc-ass2called "www" – the path to this folder would be:

C:\Users\cheta\Desktop\cc-ass2\www

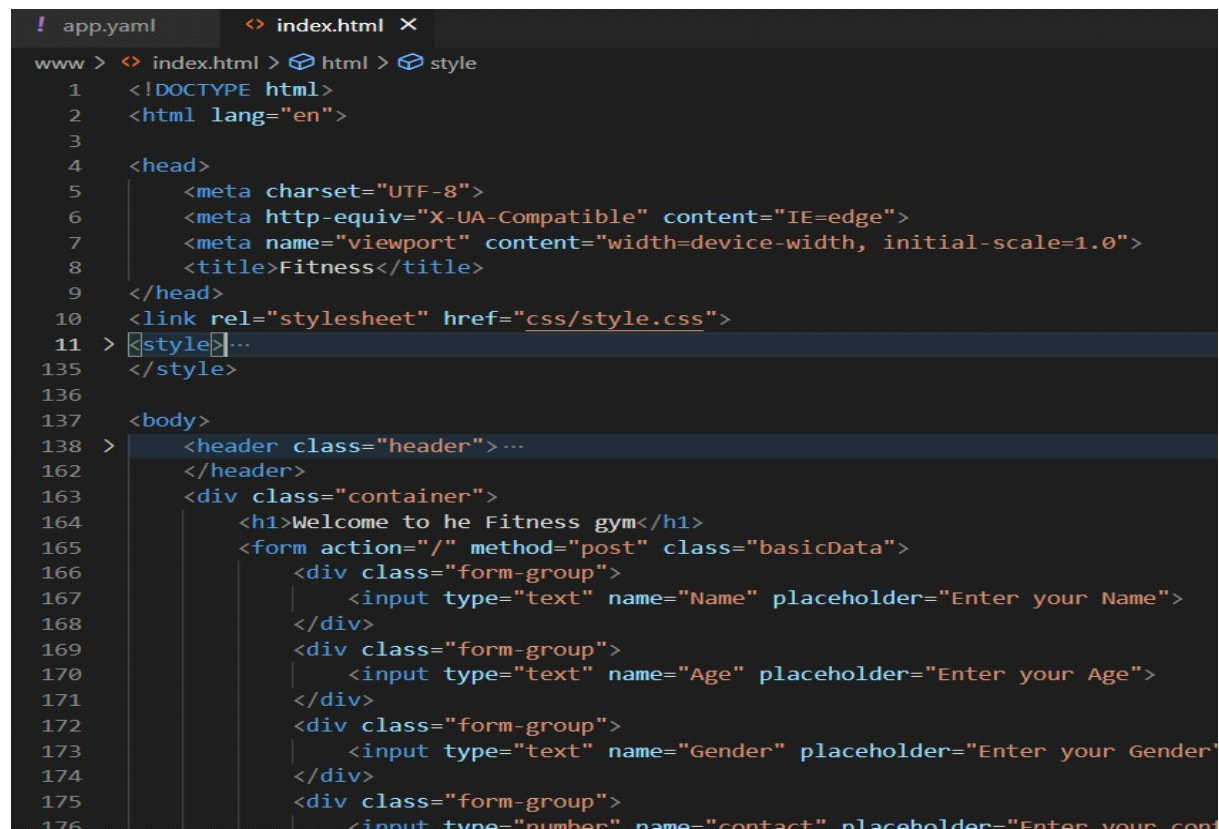
Using text editors such as Visual Studio Code (<https://code.visualstudio.com/>), create a file called **app.yaml** in the **cc-ass2** folder with the following contents:

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

handlers:
- url: /
  static_files: www/index.html
  upload: www/index.html
- url: /(.* )
  static_files: www/\1
  upload: www/(.*)
```

Note: Please do not copy and paste these lines into your text editor – you might end up with strange characters – simply type the min to your editor.

The create a file in the **www** folder called **index.html** with few lines of code in it:

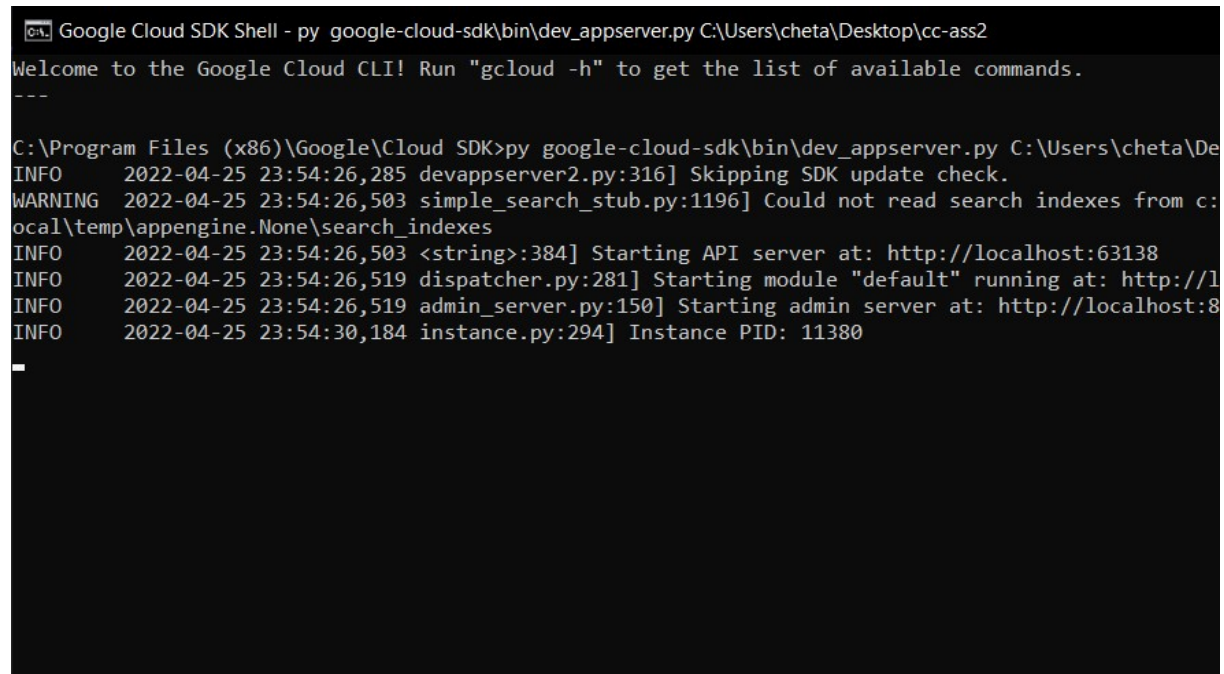


The screenshot shows a Visual Studio Code editor with two files open: **app.yaml** and **index.html**. The **app.yaml** file contains the configuration for a Python 2.7 application. The **index.html** file contains the HTML code for a fitness gym website. The code includes a header, a container with a welcome message, a form with input fields for Name, Age, Gender, and Contact, and a submit button.

```
! app.yaml  < index.html X
www > < index.html > html > style
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <title>Fitness</title>
9  </head>
10 <link rel="stylesheet" href="css/style.css">
11 > <style>...
135 </style>
136
137 <body>
138 > <header class="header">...
162 </header>
163 <div class="container">
164   <h1>Welcome to he Fitness gym</h1>
165   <form action="/" method="post" class="basicData">
166     <div class="form-group">
167       <input type="text" name="Name" placeholder="Enter your Name">
168     </div>
169     <div class="form-group">
170       <input type="text" name="Age" placeholder="Enter your Age">
171     </div>
172     <div class="form-group">
173       <input type="text" name="Gender" placeholder="Enter your Gender">
174     </div>
175     <div class="form-group">
176       <input type="number" name="contact" placeholder="Enter your con
```


Then start the **Google cloud sdk shell** program that can be found under **Applications**. Then follow the command and then continue writing the file path as shown below:

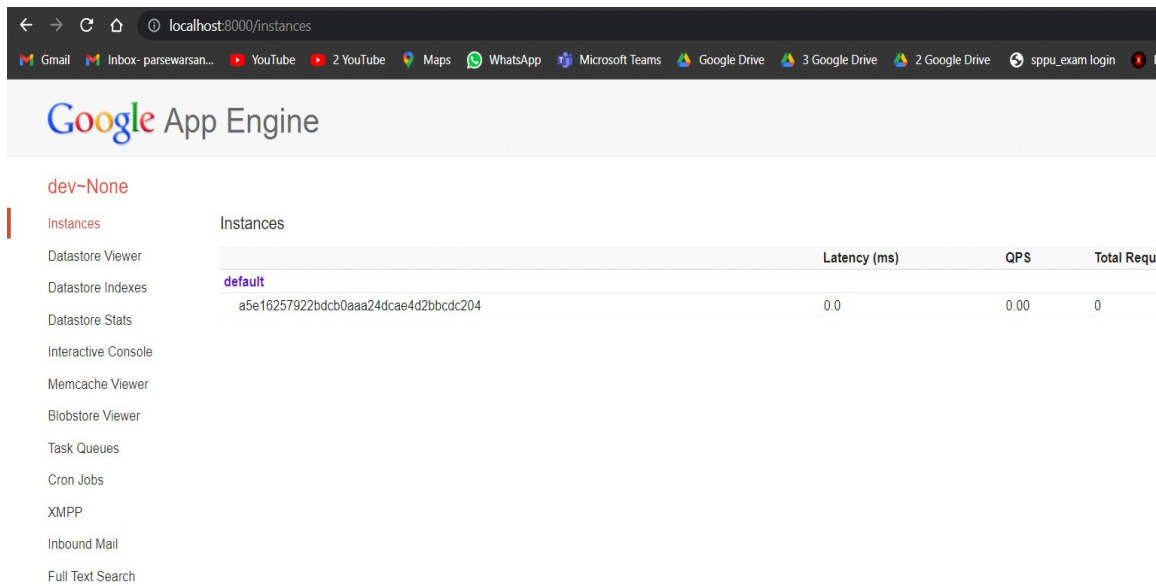
>py google-cloud-sdk\bin\dev_appserver.py C:\Users\cheta\Desktop\cc-ass2



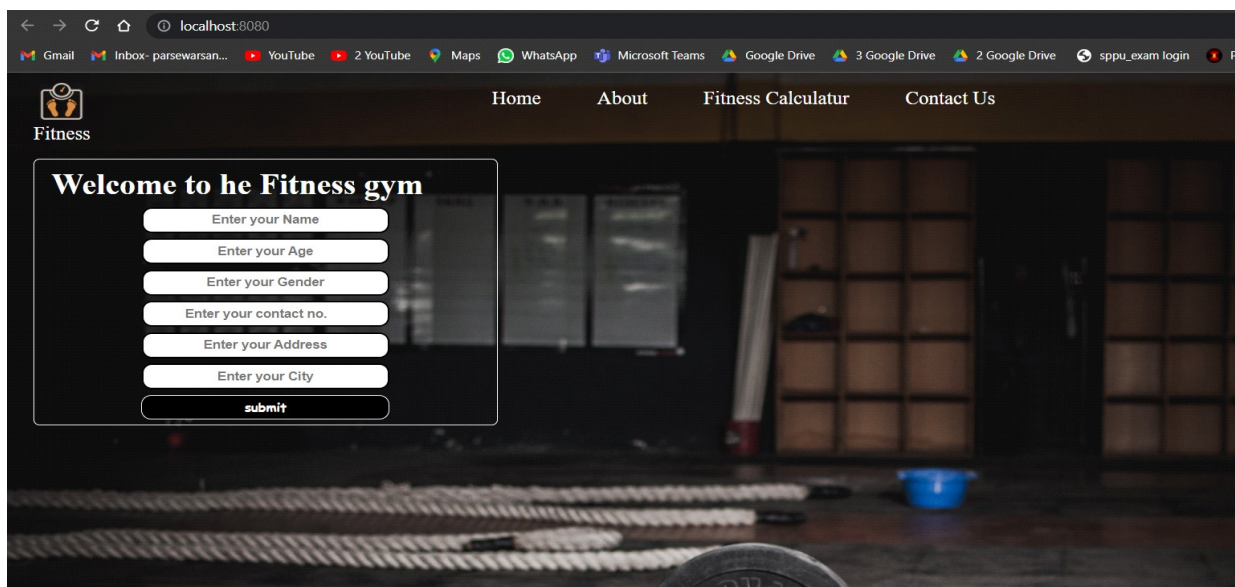
```
Google Cloud SDK Shell - py google-cloud-sdk\bin\dev_appserver.py C:\Users\cheta\Desktop\cc-ass2
Welcome to the Google Cloud CLI! Run "gcloud -h" to get the list of available commands.
---
C:\Program Files (x86)\Google\Cloud SDK>py google-cloud-sdk\bin\dev_appserver.py C:\Users\cheta\Desktop\cc-ass2
INFO      2022-04-25 23:54:26,285 devappserver2.py:316] Skipping SDK update check.
WARNING   2022-04-25 23:54:26,503 simple_search_stub.py:1196] Could not read search indexes from c:\local\temp\appengine.None\search_indexes
INFO      2022-04-25 23:54:26,503 <string>:384] Starting API server at: http://localhost:63138
INFO      2022-04-25 23:54:26,519 dispatcher.py:281] Starting module "default" running at: http://localhost:8000
INFO      2022-04-25 23:54:26,519 admin_server.py:150] Starting admin server at: http://localhost:8080
INFO      2022-04-25 23:54:30,184 instance.py:294] Instance PID: 11380
```

Once you have written command in gcloud sdk shell and press **Enter**. After a moments your application will start. Then press **Browse** to open a browser pointing at your application which is running at <http://localhost:8000/>

Paste <http://localhost:8000/> into your browser and you should see window as follows:



Click on default to run your application.



Reference:http://en.wikipedia.org/wiki/Stack_trace

When you make a mistake in the **app.yaml** file – you must the fix the mistake and attempt to start the application again.

If you make a mistake in a file like **index.html**, you can simply fix the file and press refresh in your browser– there is no need to restart the server.

Conclusion :

Thus we have learnt to design and deploy a web application on Google App Engine .

Assignment 3

Aim: To simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

Objective: To learn the design and development process involved in creating a cloud based application.

Outcome: To simulate a cloud scenario using CloudSim

Theory:

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.

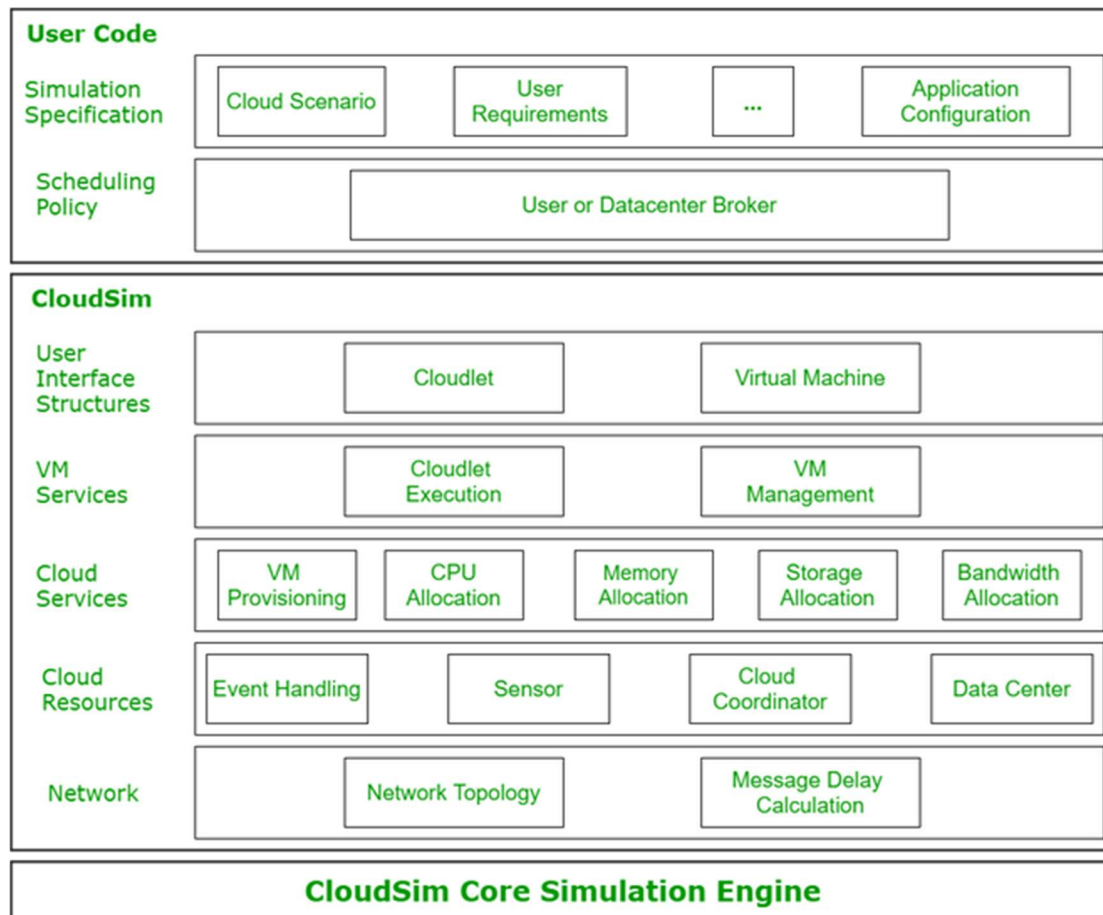
For example, if you were to deploy an application or a website on the cloud and wanted to test the services and load that your product can handle and also tune its performance to overcome bottlenecks before risking deployment, then such evaluations could be performed by simply coding a simulation of that environment with the help of various flexible and scalable classes provided by the CloudSim package, free of cost.

Benefits of Simulation over the Actual Deployment:

Following are the benefits of CloudSim:

- No capital investment involved. With a simulation tool like CloudSim there is no installation or maintenance cost.
- Easy to use and Scalable. You can change the requirements such as adding or deleting resources by changing just a few lines of code.
- Risks can be evaluated at an earlier stage. In Cloud Computing utilization of real testbeds limits the experiments to the scale of the testbed and makes the reproduction of results an extremely difficult undertaking. With simulation, you can test your product against test cases and resolve issues before actual deployment without any limitations.
- No need for try-and-error approaches. Instead of relying on theoretical and imprecise evaluations which can lead to inefficient service performance and revenue generation, you can test your services in a repeatable and controlled environment free of cost with CloudSim.

CloudSim Architecture:



CloudSim Core Simulation Engine provides interfaces for the management of resources such as VM, memory and bandwidth of virtualized Datacenters.

CloudSim layer manages the creation and execution of core entities such as VMs, Cloudlets, Hosts etc. It also handles network-related execution along with the provisioning of resources and their execution and management.

User Code is the layer controlled by the user. The developer can write the requirements of the hardware specifications in this layer according to the scenario.

Some of the most common classes used during simulation are:

- **Datacenter:** used for modelling the foundational hardware equipment of any cloud environment, that is the Data center. This class provides methods to specify the functional requirements of the Data center as well as methods to set the allocation policies of the VMs etc.
- **Host:** this class executes actions related to management of virtual machines. It also defines policies for provisioning memory and bandwidth to the virtual machines, as well as allocating CPU cores to the virtual machines.

- VM: this class represents a virtual machine by providing data members defining a VM's bandwidth, RAM, mips (million instructions per second), size while also providing setter and getter methods for these parameters.
- Cloudlet: a cloudlet class represents any task that is run on a VM, like a processing task, or a memory access task, or a file updating task etc. It stores parameters defining the characteristics of a task such as its length, size, mi (million instructions) and provides methods similarly to VM class while also providing methods that define a task's execution time, status, cost and history.
- Data center Broker: is an entity acting on behalf of the user/customer. It is responsible for functioning of VMs, including VM creation, management, destruction and submission of cloudlets to the VM.
- CloudSim: this is the class responsible for initializing and starting the simulation environment after all the necessary cloud entities have been defined and later stopping after all the entities have been destroyed.

Architecture of CloudSim Simulation

The main scenario of CloudSim simulation is revealed in figure 1. This scenario may be described the detail of each component of CloudSim as follows: that Data centers (DC) provides resource such as more than one host. Host is physical machine which allocates more than one VMs. Virtual machine (VM) are logical equipment on that the cloudlet will be executed. Broker has DC characteristics that allow it to submit virtual machines to the exact host. Cloud Information Service (CIS) is responsible for the listing of resources, indexing, as well as discovering the efficiency of data centers.

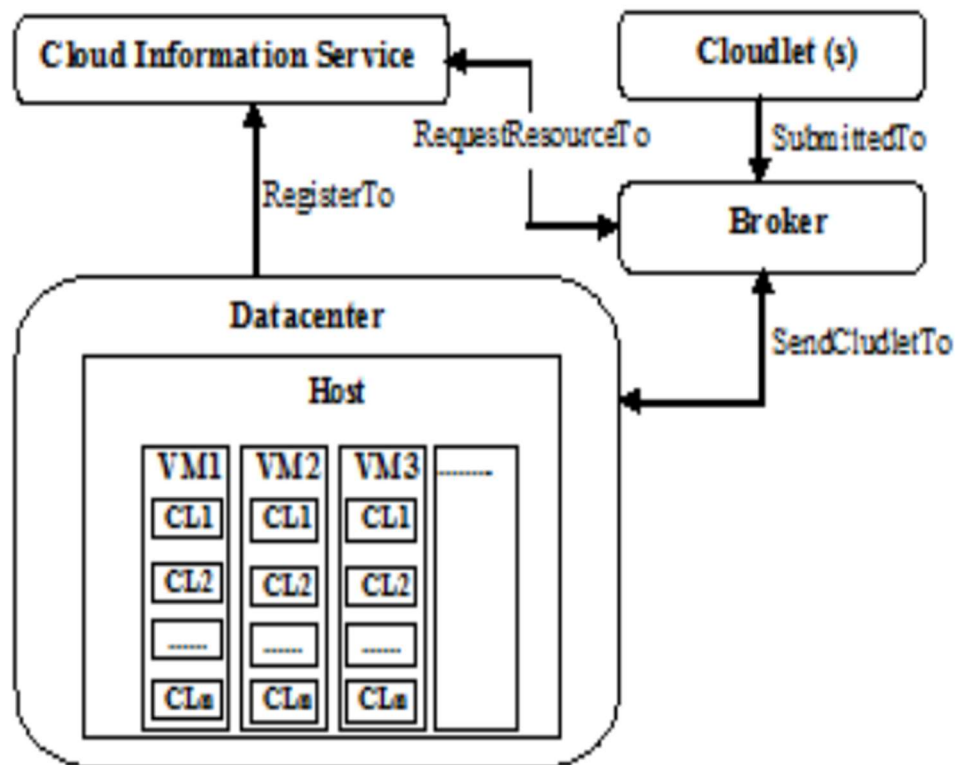


Fig. 1 Basic scenario of Simulation

Installation:

The basic requirement to configure CloudSim as follows:

1. Download

a) CloudSim(3.0.3 or 4.0) installer from

<https://github.com/Cloudslab/cloudsim/releases>

b) Eclipse IDE installer for java developers from

<http://www.eclipse.org/downloads>

c) Java Development Kit(1.6 or 1.7) if it is not available on your desktop or laptop from

<https://www.java.com>

2. Extract

a) CloudSim Installer

b) Eclipse installer

3. Open Eclipse and create a new Java Project:

File -> New-> Java Project -> Write Project Name -> Use execution environment JRE -> Next-> JRE System Library -> Add External file where you extracted the of cloudsim 3.0.3 or 4.0 -> Finish

4. Import CloudSim project into new java project.

Conclusion: Thus we have learnt to simulate a cloud scenario using CloudSim.

Assignment No. 4

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

Objective: To transfer the files from one VM to another VM.

Theory:

What is a virtual machine?

Virtual machine is an application providing a platform independent programming run time that allows applications to execute in the same manner on different platforms. The virtual machine acts as a bridge to the real environment, hiding the details of operating system. Do not confuse this term with system virtual machines, such as VMware, Virtual server, Xen which enables one to run multiple OS on a single piece of hardware.

Types of virtualization techniques:

1. Guest Operating system virtualization
2. Shared Kernel Virtualization
3. Kernel Level Virtualization
4. Hypervisor Virtualization

1. Guest Operating system Virtualization

- This is the most simple and easiest way to do virtualization.
- In this virtualization one host operating system is present and in that, virtualization software is installed in that host operating system.
- The host OS can be anything such as Windows, Mac, or Linux and the virtualization software will run as just like any other application runs on the OS.
- That virtualization software will take care of all the virtualization task and it helps in running the guest operating system.
- One can run multiple OS using that virtualization software, it will take care of all the things like memory management, resource management, hard disk partitioning, etc,
- In the technique, there is no extra configuration is required for the host OS and also the host hardware.
- In the above figure, we can see that the virtualization application is using the same hardware as that of the host operating system and it's running inside the host OS.
- So, the advantage in this case, if there will is no extra cost for the hardware configuration and software configuration, is required
- But since it's the cheapest way of virtualization so there will be issues with performance as there is high levels of abstraction.
- VMware, VirtualBox are the software used for this virtualization.

2. Shared Kernel Virtualization

- This kind of virtualization are also known as the system level or operating system virtualization.
- To understand this technique first we need to understand a few basic terms about the Linux operating system.
- There are two main components of the Linux operating system
1) Kernel 2) root
- Kernel is the one which handles all the important between operating system and system hardware
- Root file contains all the important libraries and files and utilities which are necessary to run the operating system.
- So, under this virtualization technique, each guest Operating system has their own root file and they share the same kernel.
- This root sharing feature is an inbuilt feature which is provided in the Linux operating system and this virtualization techniques is only using this feature.
- So basically, the host and the guest operating system share the same kernel to communicate with the system hardware, but different root files for their own functioning.
- The only drawback in this technique is with the compatibility of the operating system like one wants to run a Windows OS with this method then it will not work, or if someone wants to run a Linux version 2.6 and host OS is 2.4 then again it will not work.
- Example of this kind of techniques are Linux Vserver, Solaris Zones, and containers, etc.

3. Kernel level virtualization

- Is this virtualization technique the guest operating system runs its individual kernel unlike the shared kernel virtualization
- There can be multiple guest operating system and each one will have their own kernel.
- But the guest operating system kernel should have a similar configuration like the host operating system kernel otherwise there will be compatibility issues
- The kernel-level virtualization includes user-mode Linux and kernel-based virtual machine
- The above diagram shows the implementation of the kernel level visualization.

4. Hypervisor Virtualization

- In this virtualization, a program called hypervisor runs directly in the hardware of the CPU which is generally called the ring 0 which is the highest level of privileges provided by the CPU hardware to any software.
- Generally, Operating system only has the privileges to run in ring 0, so in this case, hypervisor runs in ring 0 which is also called type 1 VMM (Virtual machine monitor). As the name suggest it monitors all the guest operating system which are installed in the virtual machine and also provides interfaces for higher-level administration and monitoring.
- So, if the hypervisor runs in ring 0 then the kernel of the guest operating system will not get proper privileges to run so to address this issue hypervisor as a number of different solutions described below

1. Paravirtualization

- In this technique, the system calls are made by the kernel of the guest operating system, and those calls are directly handled by the hypervisor and hypervisor, in turn, completes all the tasks.
- The calls between the hypervisor and the guest operating system kernel are called the hypercalls.

2. Full virtualization

- In this case, complete control of the guest operating system is given to the guest operating system.
- The calls are generally controlled and monitored by the hypervisor it provides a CPU emulation to handle and modify the privileges.
- But this scenario is not every efficient and results in the degrading of the system performance compared to paravirtualization.

3. Hardware Virtualization

- With the latest CPU coming into the market INTEL and AMD has devised new CPUs that help in providing an extra layer on top of ring 0 which helps the hypervisor to run and take control of the guest operating system.
- This eliminates the overhead of the CPU emulation
- The above figure shows the working of the hypervisor.
- The hypervisor also runs a management console which helps monitor the guest operating systems that are running in the CPU and it also allows the system administrator to manage the virtual machines
- The different hypervisor available in the market is Microsoft hyper V Xen, VMware ESX Server, etc.

Types of virtual machines:

You can classify virtual machines into two types:

1. System Virtual Machine:

These types of virtual machines gives us complete system platform and gives the execution of the complete virtual operating system. Just like virtual box, system virtual machine is providing an environment for an OS to be installed completely. We can see in below image that our hardware of Real Machine is being distributed between two simulated operating systems by Virtual machine monitor. And then some programs, processes are going on in that distributed hardware of simulated machines separately.

Process Virtual Machine :

While process virtual machines, unlike system virtual machine, does not provide us with the facility to install the virtual operating system completely. Rather it creates virtual environment of that OS while using some app or program and this environment will be destroyed as soon as we exit from that app. Like in below image, there are some apps running on main OS as well some virtual machines are created to run other apps. This shows that as those programs required different OS, process virtual machine provided them with that for the time being those programs are running.

Example –Wine software in Linux helps to run Windows applications.

What is a hypervisor? Types of hypervisors.

The hypervisor or virtual machine monitor (VMM) is a solution, which creates and manages virtual machines (VMs). The hypervisor is what controls and allocates what portion of hardware resources each operating system should get, in order every one of them to get what they need and not to disrupt each other.

There are two types of hypervisors:

- Type 1 hypervisor:** hypervisors run directly on the system hardware – A “bare metal” embedded hypervisor,

- Type 2 hypervisor:** hypervisors run on a host operating system that provides virtualization services, such as I/O device support and memory management.

Type 1 hypervisors:

1. VMware ESX and ESXi

These hypervisors offer advanced features and scalability, but require licensing, so the costs are higher. There are some lower-cost bundles that VMware offers and they can make hypervisor technology more affordable for small infrastructures.

VMware is the leader in the Type-1 hypervisors. Their vSphere/ESXi product is available in a free edition and 5 commercial editions.

2. Microsoft Hyper-V

The Microsoft hypervisor, Hyper-V doesn't offer many of the advanced features that VMware's products provide. However, with XenServer and vSphere, Hyper-V is one of the top 3 Type-1 hypervisors. It was first released with Windows Server, but now Hyper-V has been greatly enhanced with Windows Server 2012 Hyper-V. Hyper-V is available in both a free edition (with no GUI and no virtualization rights) and 4 commercial editions – Foundations (OEM only), Essentials, Standard, and Datacenter. Hyper-V

3. Citrix XenServer

It began as an open source project. The core hypervisor technology is free, but like VMware's free ESXi, it has almost no advanced features. Xen is a type-1 bare-metal hypervisor. Just as Red Hat Enterprise Virtualization uses KVM, Citrix uses Xen in the commercial XenServer. Today, the Xen open source projects and community are at Xen.org. Today, XenServer is a commercial type-1 hypervisor solution from Citrix, offered in 4 editions. Confusingly, Citrix has also branded their other proprietary solutions like XenApp and XenDesktop with the Xen name.

4. Oracle VM

The Oracle hypervisor is based on the open source Xen. However, if you need hypervisor support and product updates, it will cost you. Oracle VM lacks many of the advanced features found in other bare-metal virtualization hypervisors.

Type 2 hypervisor

1. VMware Workstation/Fusion/Player VMware Player is a free virtualization hypervisor.

It is intended to run only one virtual machine (VM) and does not allow creating VMs.

VMware Workstation is a more robust hypervisor with some advanced features, such as record-and-replay and VM snapshot support. VMware Workstation has three major use cases:

- for running multiple different operating systems or versions of one OS on one desktop,
 - for developers that need sandbox environments and snapshots, or
 - for labs and demonstration purposes.
2. VMware Server VMware Server is a free, hosted virtualization hypervisor that's very similar to the VMware Workstation. VMware has halted development on Server since 2009
 3. Microsoft Virtual PC :This is the latest Microsoft's version of this hypervisor technology, Windows Virtual PC and runs only on Windows 7 and supports only Windows operating systems running on it.
 4. Oracle VM VirtualBox
VirtualBox hypervisor technology provides reasonable performance and features if you want to virtualized on a budget. Despite being a free, hosted product with a very small footprint, VirtualBox shares many features with VMware vSphere and Microsoft Hyper-V.
 5. Red Hat Enterprise Virtualization
Red Hat's Kernel-based Virtual Machine (KVM) has qualities of both a hosted and a bare-metal virtualization hypervisor. It can turn the Linux kernel itself into a hypervisor so the VMs have direct access to the physical hardware.

Steps to transfer the files from one virtual machine to another virtual machine:

- Download and install Oracle's Virtual Box<https://www.virtualbox.org/wiki/Downloads>
- Download Ubuntu VMDK Virtual Machine Disk File
Image-<https://app.vagrantup.com/bento/boxes/ubuntu-18.04>
- Launch Virtual box and create a new VM
- Click on new and mention the Name and the machine folder along with the Type and Version of the Machine to be created.
- Assign memory size for our VM (1024 MB sufficient for now).
- Select the option Use an existing virtual hard disk file and locate the downloaded VMDK image and create VM
- Now we have to create a NAT (Network Address Translation) Network so go to File -> Preferences -> Network -> Add a New NAT Network (Click on +)
- Right click and edit the Network name and CIDR if needed.
- Repeat the process of launching the VM for 2 instances
- Now go to the setting, go to the network setting and change the adapter to NAT Network and select the NAT Network you made

- Launch the VM now
- Install the net-tools to know the IP's of the instance
- create a file and write something into it
- If your file is on the VM with IP *172.168.2.4* and the second VM's IP is *172.168.2.5*.

- Transfer the file using *SCP* Secure Copy Protocol

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

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

Output:

On virtual machine 1 (vm1)

On virtual machine 2 (vm2)

For file transfer vm1 and vm2 should be in same network

Vm1 ip is 10.0.2.6

Vm2 ip is 10.0.2.5

That means both are in same network

Create transfer.txt on vm1 To transfer this newly created file type following command

Checking on vm2 whether file is transferred or not

Applications:

1. Virtual Box(Windows/Mac/Linux, Free):

Virtual machine descriptions and parameters are stored entirely in plain text XML files for easy portability and easy folder sharing. User friendly, allowing users to install software on the virtual machine that grants extra privileges to the host machine for tasks like sharing files, sharing drives and peripherals and more.

2. Parallels:

Runs virtualization on windows and Linux. The parallel software's boasts a direct link, thanks to optimization on Intel and AMD chips, to the host computers hardware with selective focus when we jump into the virtual machine to work the host machine automatically processing a power to it.

3. Amazon EC2(elastic cloud compute):

It provides scalable computing capacity in the AWS cloud leveraging it enables organizations to develop and deploy applications faster without needing to invest in hardware upfront. Users can launch virtual servers, configure security and networking and manage cookies from an intuitive dashboard. EC2 is the core compute component of the technology stack. EC2 makes life easier for developers by providing secure and resizable compute capacity in the cloud.

Conclusion:

Thus, we successfully transferred the files from one virtual machine to another virtual machine using some commands.

Assignment no. 05

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

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

Theory :

What is Virtual Machine?

Virtual Machine is like fake computer system operating on your hardware. It partially uses the hardware of your system (like CPU, RAM, disk space, etc.) but its space is completely separated from your main system. Two virtual machines don't interrupt in each other's working and functioning nor they can exceed each other's space which gives an illusion that we are using totally different hardware system.

Types of Virtual Machines:

You can classify virtual machines into two types:

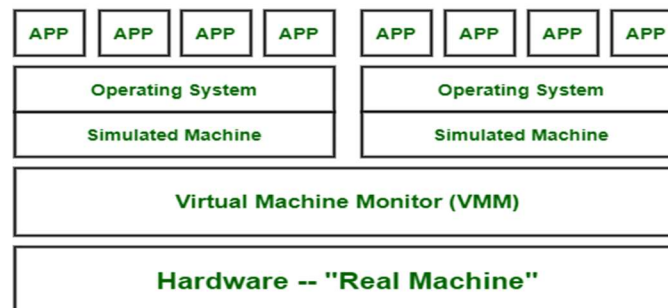
1. System Virtual Machine:

These types of virtual machines give us complete system platform and give the execution of the complete virtual operating system. Just like virtual box,

system virtual machine is providing an environment for an OS to be installed completely. We can see in below image that our hardware of Real Machine is being distributed

between two simulated operating systems by Virtual machine monitor. And then some programs, processes are going on in that distributed hardware of simulated machines separately.

System Virtual Machine



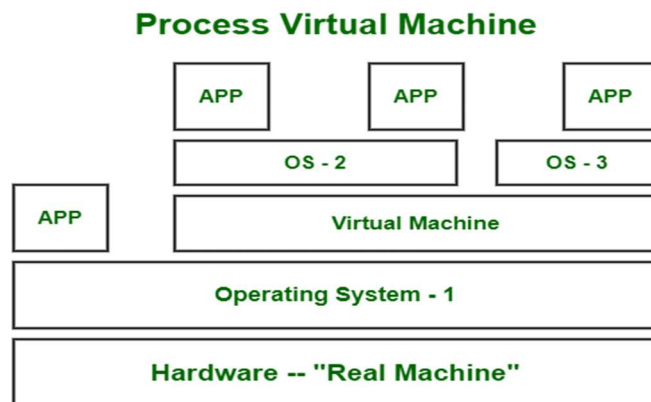
2. Process Virtual Machine :

While process virtual machines, unlike system virtual machine, does not provide us with the facility to install the virtual operating system completely.

Rather it creates virtual environment of that OS while using some app or program and this environment will be destroyed as soon as we exit from that app.

Like in below image, there are some apps running on main OS as well some virtual machines are created to run other apps. This shows that as those programs

required different OS, process virtual machine provided them with that for the time being those programs are running.



How do Virtual Machine work?

Virtualisation Technology allows you to share a system with many virtual environments. The hypervisor manages the hardware and separates the physical resources from the virtual environments.

What is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Deploying Openstack(private cloud) on AWS(public cloud)

Probably everyone with OpenStack hands-on experience would agree that sometimes it could be hard and frustrating to install it and test it. Especially if you do not own a small Data Center or the proper physical infrastructure to support its installation needs... However, wouldn't it be great to use a public cloud provider and its infrastructure to experiment and create POC environments? In this article, I will provide the basic steps of how to spin up a Redhat OpenStack POC environment on AWS and the things you should modify in order to get OpenStack up and running probably on every cloud provider.

Step 0: Prerequisites

Hardware

Instance with at least 8GB RAM and 50GB storage, with hardware virtualization enabled AMI



RHEL-7.6_HVM_GA-20190128-x86_64-0-Hourly2-GP2 - ami-000db10762d0c4c05

Select

Provided by Red Hat, Inc.
64-bit (x86)

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
---	-----------------	-----------	---	----	----------	---	----------	-----

Software

As we are installing RedHat Openstack, We will need RHOSP iso file. Generally RedHat asks for a subscription to download this file. [COPY LINK DRIVE](#)

Network

We need to disable NetworkManager and firewall as they manage network interfaces and the firewall, and Neutron wants to manage them as well. When there are two managers that don't know about each other, you can perhaps imagine that the result is chaos.

```
$ sudo systemctl disable firewallld
$ systemctl stop firewallld
$ systemctl disable NetworkManager$ systemctl stop NetworkManager
```

Step 1: Installation

I used *gdown* command to download the required iso files inside the instance.

```
sudo yum install python3 -y
sudo pip3 install gdown#gdown needs the file ID to download files from gdrive
the ID can be found in the Gdrive link after file/d/ till /view#gdown --id ID --output rhosp.iso
```

Now, after downloading the Iso file, lets copy the contents to another folder. For this, we have to mount the iso file first

```
mkdir temp openstack
mount rhosp.iso temp
cp -rvf temp/ openstack/
```

After this, we have to create a repo inside the folder

```
Sudo yum install createrepo
cd openstack
createrepo -v .
```

then we have to write a repo file for the newly created repo

```
vi/etc/yum.repos.d/local.repo[openstack]
baseurl=file:///home/ec2-user/openstack
gpgcheck=0
```

We can check the created repo using yum repolist, If the repo isn't shown use yum clean all.


```
[root@ip-172-31-51-195 ec2-user]# yum repolist
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Repository 'openstack' is missing name in configuration, using id
repo id                repo name                status
openstack               openstack                 2,126
rhui-REGION-client-config-server-7/x86_64 Red Hat Update Infrastru 10
rhui-REGION-rhel-server-releases/7Server/x86_64 Red Hat Enterprise Linux 29,219
rhui-REGION-rhel-server-rh-common/7Server/x86_64 Red Hat Enterprise Linux 243
repolist: 31,598
```

Install Openstack using yum install opensack-packstack

Generateanswer	file	using
packstack --gen-answer-file=openstack.txt	After generating,	Run answer file using
packstack --answer-file=openstack.txt		

```
root@ip-172-31-51-195:/home/ec2-user
Preparing Nova VNC Proxy entries [ DONE ]
Preparing OpenStack Network-related Nova entries [ DONE ]
Preparing Nova Common entries [ DONE ]
Preparing Neutron LBaaS Agent entries [ DONE ]
Preparing Neutron API entries [ DONE ]
Preparing Neutron L3 entries [ DONE ]
Preparing Neutron L2 Agent entries [ DONE ]
Preparing Neutron DHCP Agent entries [ DONE ]
Preparing Neutron Metering Agent entries [ DONE ]
Checking if NetworkManager is enabled and running [ DONE ]
Preparing OpenStack Client entries [ DONE ]
Preparing Horizon entries [ DONE ]
Preparing Swift builder entries [ DONE ]
Preparing Swift proxy entries [ DONE ]
Preparing Swift storage entries [ DONE ]
Preparing Gnocchi entries [ DONE ]
Preparing Redis entries [ DONE ]
Preparing Ceilometer entries [ DONE ]
Preparing Aodh entries [ DONE ]
Preparing Puppet manifests [ DONE ]
Copying Puppet modules and manifests [ DONE ]
Applying 172.31.51.195_controller.pp [ | ]
Testing if puppet apply is finished: 172.31.51.195_controller.pp [ | ]
Testing if puppet apply is finished: 172.31.51.195_controller.pp [ | ]
```

We have successfully installed Openstack on AWS

Connecting to Horizon

As Horizon runs on apache web server , We need to configure Apache to use public IP of the instance for accessing openstack.

```

# *****
# Vhost template in module puppetlabs-apache
# Managed by Puppet
# *****

<VirtualHost *:80>
  ServerName ip-172-31-51-195.ec2.internal

  ## Vhost docroot
  DocumentRoot "/var/www/"
  ## Alias declarations for resources outside the DocumentRoot
  Alias /dashboard/static "/usr/share/openstack-dashboard/static"

  ## Directories, there should at least be a declaration for /var/www/

  <Directory "/var/www/">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Require all granted
  </Directory>

  ## Logging
  ErrorLog "/var/log/httpd/horizon_error.log"
  ServerSignature Off
  CustomLog "/var/log/httpd/horizon_access.log" combined

  ## RedirectMatch rules
  RedirectMatch permanent ^/$ /dashboard

  ## Server aliases
  ServerAlias 54.89.193.220
  ServerAlias ip-172-31-51-195.ec2.internal
  ServerAlias localhost
  WSGIApplicationGroup %{GLOBAL}
  WSGIDaemonProcess apache display-name=horizon group=apache processes=3 threads=10 user=apache
  WSGIProcessGroup apache
  WSGIScriptAlias /dashboard "/usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi"
</VirtualHost>

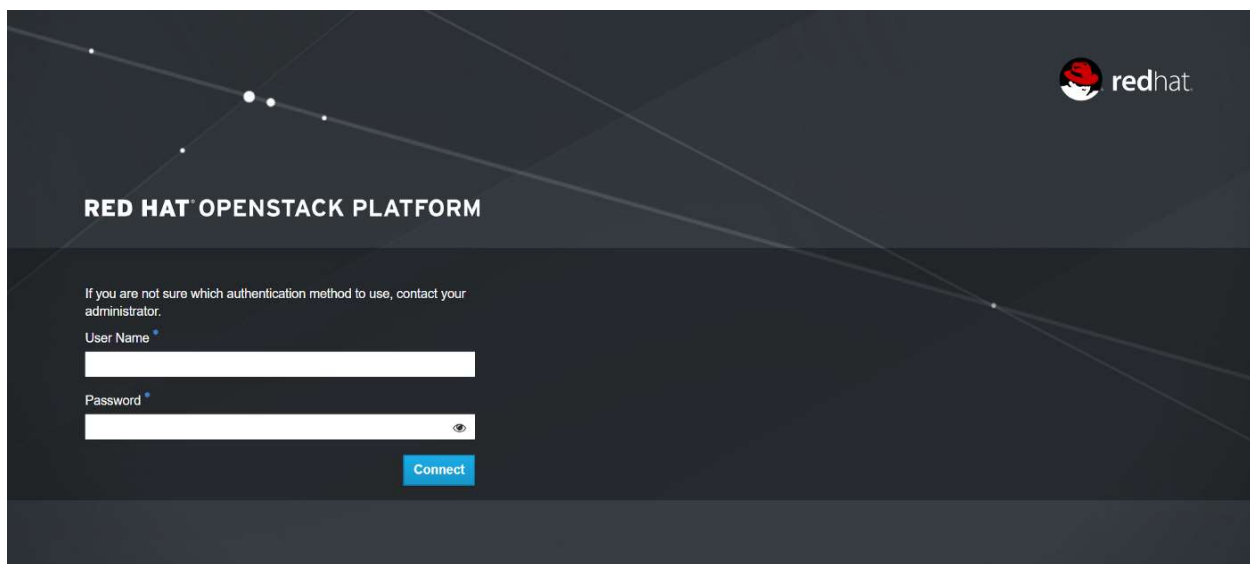
```

Here, we have to change ServerAlias from private IP to public IP.

After that, just restart the webserver using

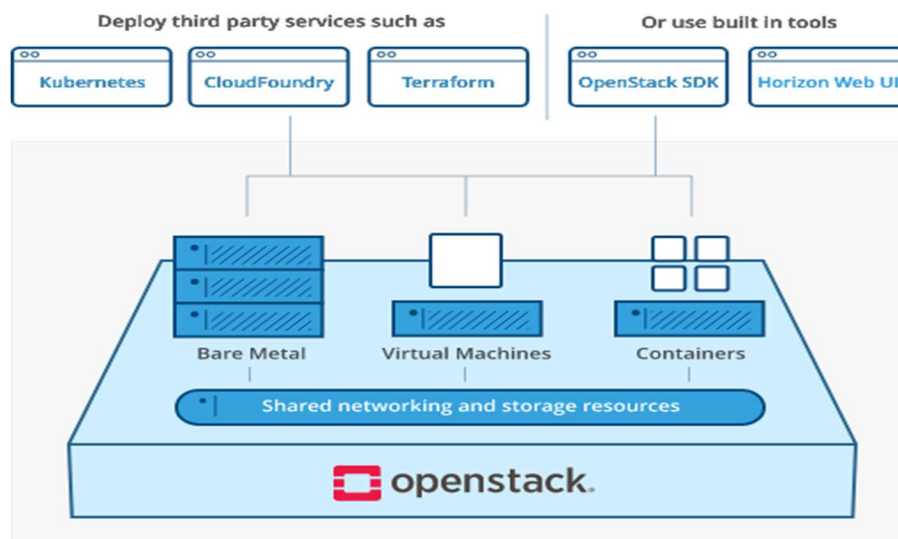
```
systemctl restart httpd
```

Now, We can access Openstack in our web browser using public IP of instance



What is Open Stack?

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.



Vision for OpenStack Clouds :

- **Purpose :** This is a living document. Its purpose is to document the OpenStack community's vision for the output of the OpenStack project as a whole, as it evolves over time.
- **Scope :** The scope of this document is limited to the cloud services that an end-user interacts with. This corresponds to the main 'OpenStack' bucket and parts of the 'OpenStack Operations' bucket in the OpenStack project map.
- **The Pillars of Cloud :** There are at least as many different opinions of what 'cloud' means as there are software developers. However, we can all agree that cloud does mean something. Cloud computing promotes more efficient utilization of resources by reducing the transaction costs involved in provisioning and deprovisioning infrastructure to near zero, and it is able to do so because it differs in qualitative ways from previous models of computing (including virtualization).
- **Self-service :** Clouds are self-service. They provide users with the ability to deploy applications on demand without having to wait for human action or review in the loop. The cloud has no ticket trackers.

Application Control : Clouds allow control of an application's infrastructure to be vested in the application itself. Just as clouds eliminate the need for a human approver in the loop, they also eliminate the need for a human user to be in the loop. While a cloud may have a user interface (graphical or otherwise), it must have an application programming interface.

What is Trystack?

TryStack is a free and easy way for users to try out OpenStack, and set up their own cloud with networking, storage, and computer instances. If you haven't tried it yet, go check it out now—we'll be here when you get back. TryStack is an OpenStack Foundation project; they oversee the political side of it. There are different vendors and companies that have donated resources, such as rack and power and network and servers. At the time we got involved, there wasn't much maintenance being done on the software side of it. I think it was running Ubuntu at the time, and it was running Essex I think. And, so Red Hat came in and donated RHEL subscriptions to run the operating system underneath it, and said they'd help us run RDO on top of it. So, we have a RPM packaging of OpenStack running on Red Hat Enterprise Linux, and we're donating my team's time to manage that. We went through and installed the whole cluster with RHEL, and put the initial release of RDO, which was Folsom, onto that cluster and got it up and running. At that point we had RDO available to the community to use, so there's a block of IP addresses and a block of servers in a datacenter out in San Jose, and Red Hat is managing those servers and running our software on it, to showcase our community-based OpenStack offering.

Steps to launch an Amazon EC2 Instance:

Step 1: Choose an Amazon Machine Image (AMI)

Open Amazon EC2 console and then click Launch Instance to create and configure your virtual machine. 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.

Step 2: 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.

Step 3: Configure Instance Details

You can review the configuration, storage, tagging, and security settings that have been selected for your instance.

Step 4: Add Storage

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.

Step 5: Add Tags

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

Step 6: Configure Security Group

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.

Step 7: Review Instance Launch

Connect to your Instance (either via SSH) Terminate Your Instance (go to the console and select the VM under actions and terminate).

Conclusion : Thus we have learned how to launch virtual machine using trystack(Online Openstack Demo Version). Also we have learned about openstack and trystack.

Assignment No. 6

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

Objective:

- 1] To understand the concept of Platform as a service.
- 2] To understand how we can deploy any web-application on PaaS.

Theory:

PAAS(Platform as a service):

Platform as a Service (PaaS) refers to a cloud computing configuration that helps enterprises operate with an efficient cloud-based strategy. PaaS provides a platform for customers to develop, run, and manage applications without building and maintaining the cloud infrastructure required to develop and launch applications. PaaS permits more efficient application development since the organization can focus on the application itself.

Working of PAAS :

Platform-as-a-service (PaaS) is a type of cloud computing model in which a service provider delivers a platform to customers. The platform enables the organization to develop, run, and manage business applications without the need to build and maintain the infrastructure such software development processes require.

PaaS is offered via a service provider's hosted cloud infrastructure. Users typically access PaaS offerings via a web browser. Customers pay for PaaS on a per-use basis. Some providers will charge a flat monthly fee for access to the platform and applications hosted on the platform.

PaaS can be delivered through public, private, or hybrid clouds. With a public cloud PaaS, the customer controls software deployment while the cloud provider delivers all the major IT components needed for running applications. These components can include servers, storage systems, networks, operating systems, and databases. With a private cloud offering, PaaS is delivered as software or an appliance behind a customer's firewall, typically in its on-premises data center. Hybrid cloud PaaS offers a mix of the two types of cloud service.

Rather than replace an organization's entire IT infrastructure for software development, PaaS provides key services such as application hosting or Java development. Some PaaS offerings include application design, development, testing, and deployment. PaaS services can also include web service integration, development team collaboration, database integration, and information security. PaaS includes multiple underlying cloud infrastructure components, including servers, networking equipment, operating systems, storage services, middleware, and databases.

All of these technology offerings are owned, operated, configured, and maintained by the service providers. Customers can avoid having to lay out investments in these foundational IT components that they might not be able to use to the fullest extent possible. PaaS also includes widely used resources such as development tools, programming languages, libraries, containers, database management systems, and other tools.

Advantages of PAAS :

Platform as a Service hardware and software provide benefits like streamlining development tools, reducing infrastructure cost, working on multiple operating systems, and supporting various programming languages.

Platform as a Service benefits outlined by research firm Gartner include:

- **API development and management** – Companies can use PaaS solutions to manage application programming interfaces as well as micro services. This includes security, development, creating new APIs, and end-to-end API management.
- **Business analytics/intelligence** – Some PaaS solutions include tools which empower enterprises to analyze their data for business insights and patterns of behavior. These tools give the organization the required information to make better decisions and more accurately predict things like market demand for products.
- **Business process management (BPM)** – Organizations can access a BPM platform delivered as a service through PaaS solutions. BPM suites integrate IT components needed for process management, including data, business rules, and service-level agreements.
- **Communications** – PaaS can deliver communications platforms. This allows developers to add communication features to applications, such as voice, video, and messaging.
- **Databases** – A PaaS provider can deliver database services to an organization, such as set-up and maintenance. Database PaaS is an on-demand, secure, and scalable self-service database model. Provisioning can be automated, as well as administration of databases, according to analyst firm Forrester Research.

• **Primary data management (PDM)** – Primary data management (MDM/PDM) software tracks the most essential company-wide data points, providing a single point of reference for data. From this point of reference, the software provides insights related to company operations, clients, and goals. Such data might include reference data such as information about customer transactions, and analytical data to support decision making. Users can then implement that data as they see fit, keep records of data history, and make projections based on findings.

Examples of PAAS:

Google AppEngine:

Amazon Amplify:

Digital Ocean:

Dokku:

North-Flank:

Heroku:

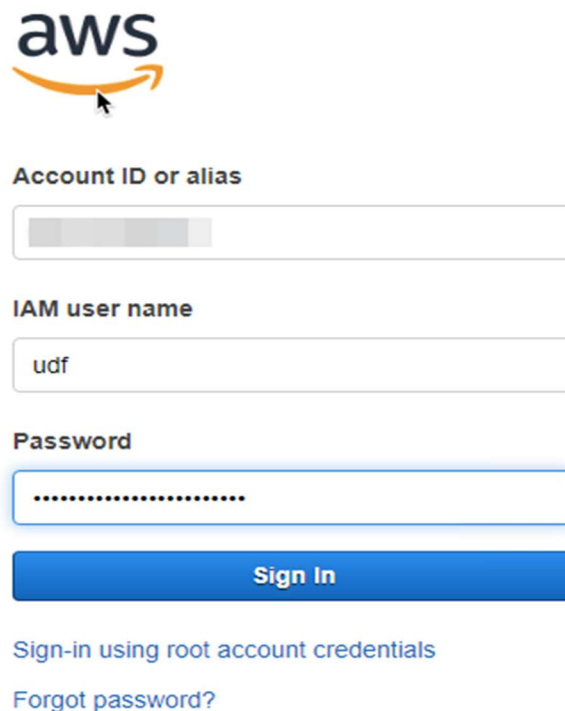
AWS Elastic Beanstalk:

Procedure:

To Deploy Web-Application on PAAS:

1.Login to the AWS

console



The image shows the AWS login page. At the top is the AWS logo. Below it are three input fields: 'Account ID or alias' (with a greyed-out placeholder), 'IAM user name' (containing 'udf'), and 'Password' (with masked dots). A blue 'Sign In' button is below the password field. At the bottom are two links: 'Sign-in using root account credentials' and 'Forgot password?'.

aws

Account ID or alias

IAM user name

udf

Password

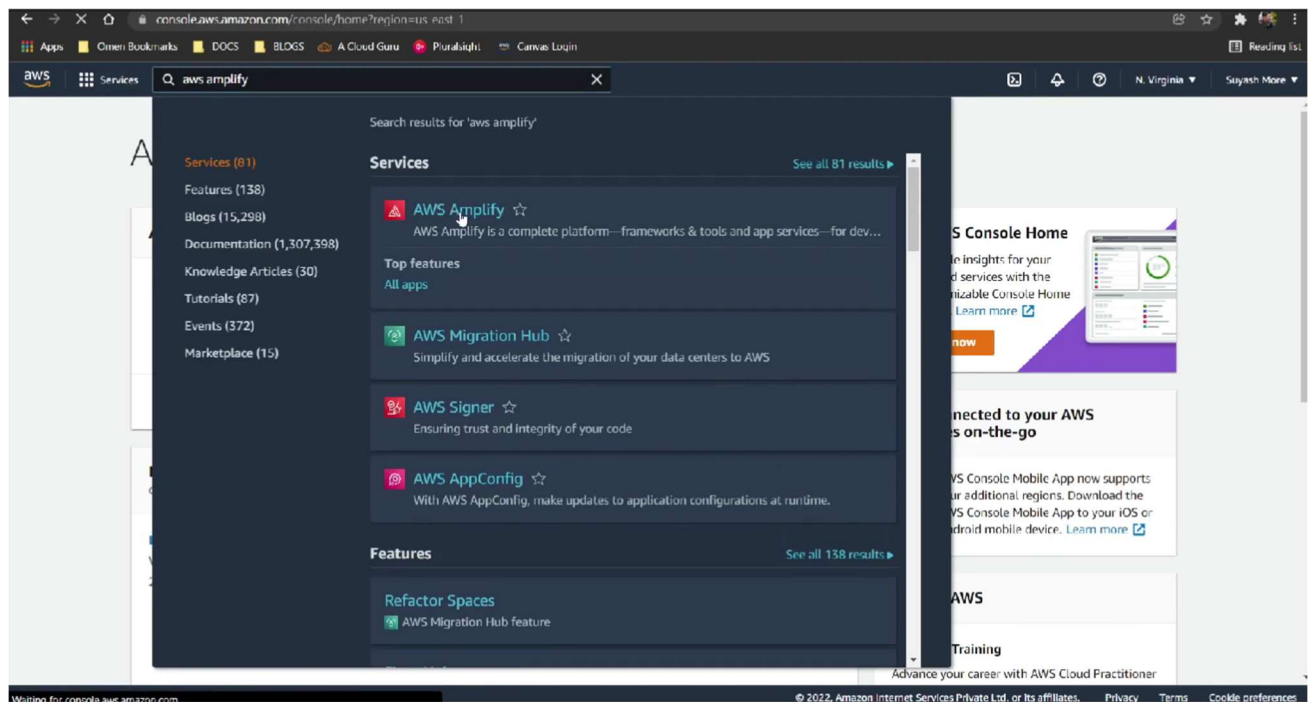
.....

Sign In

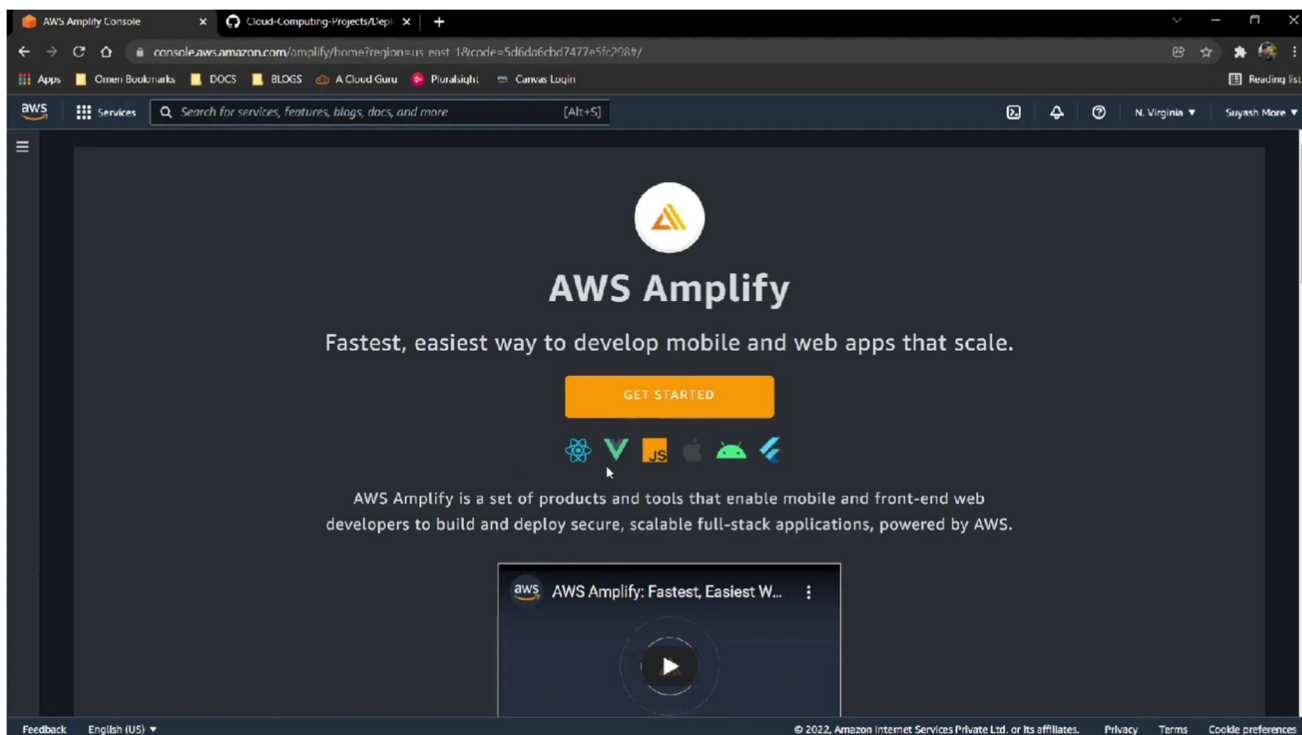
[Sign-in using root account credentials](#)

[Forgot password?](#)

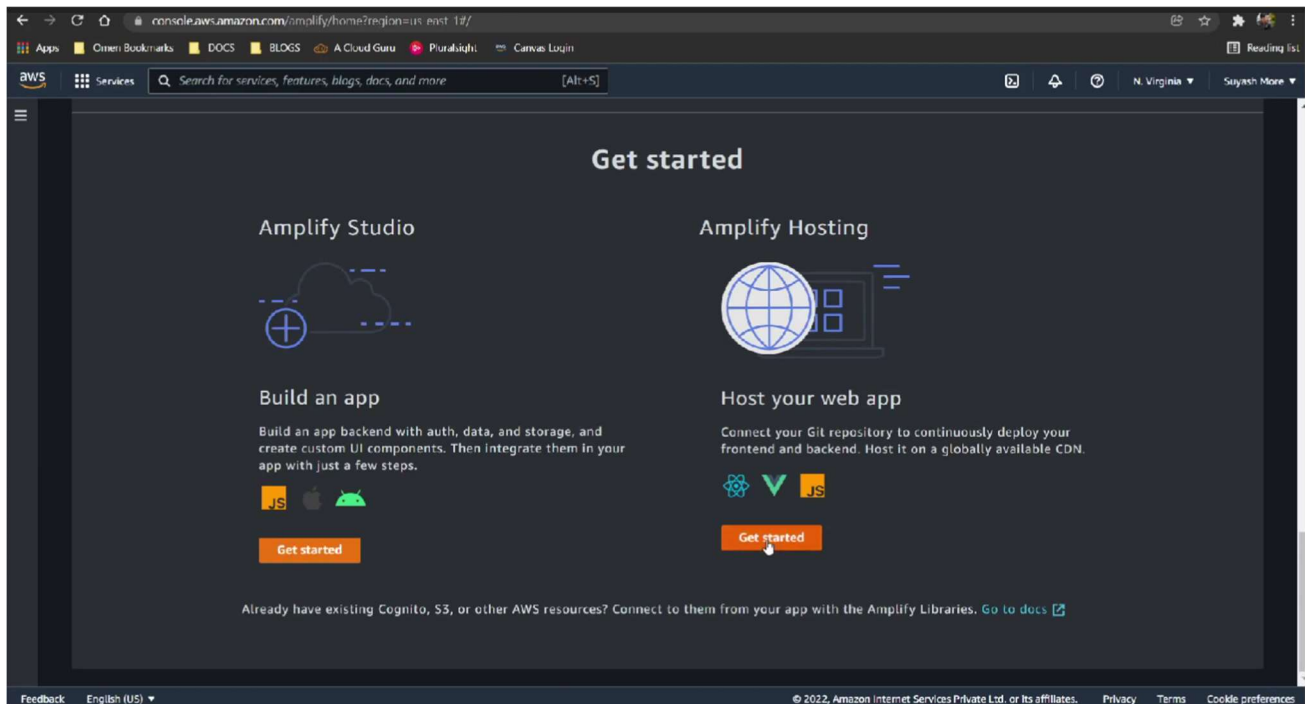
2. Find for AWS Amplify in the services.



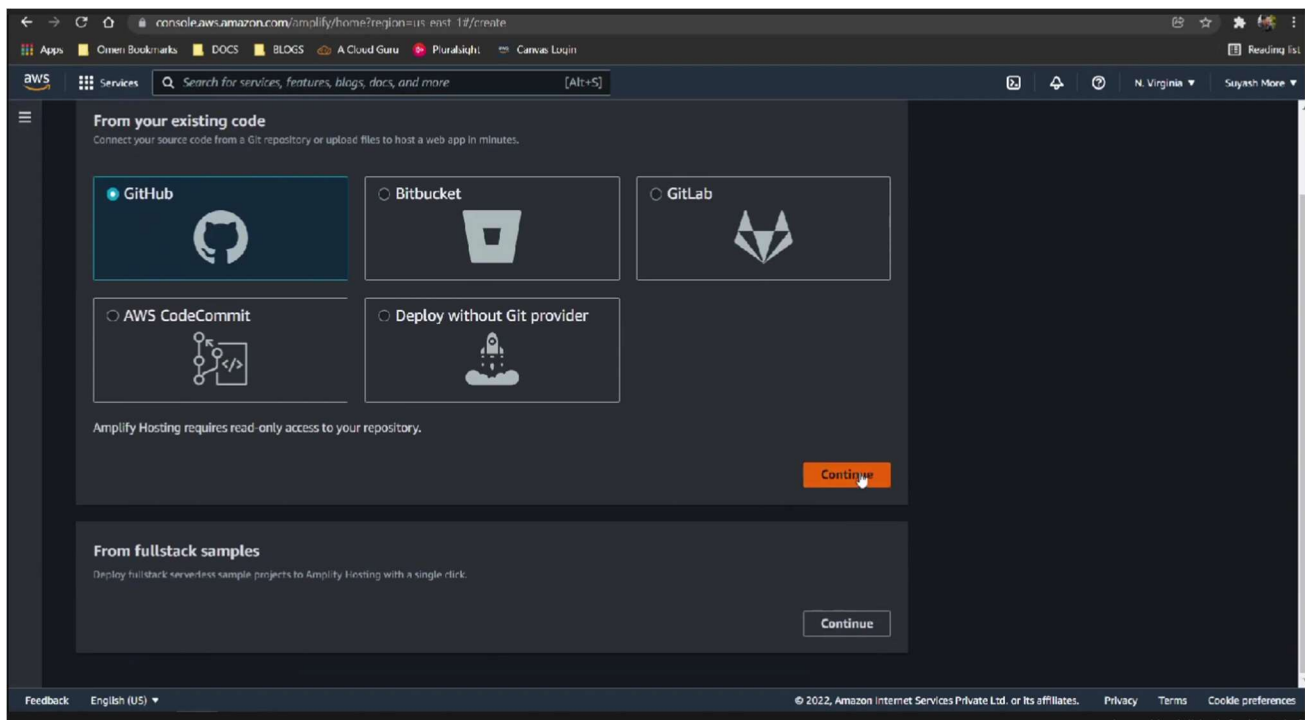
3. Get Started with Amplify service.



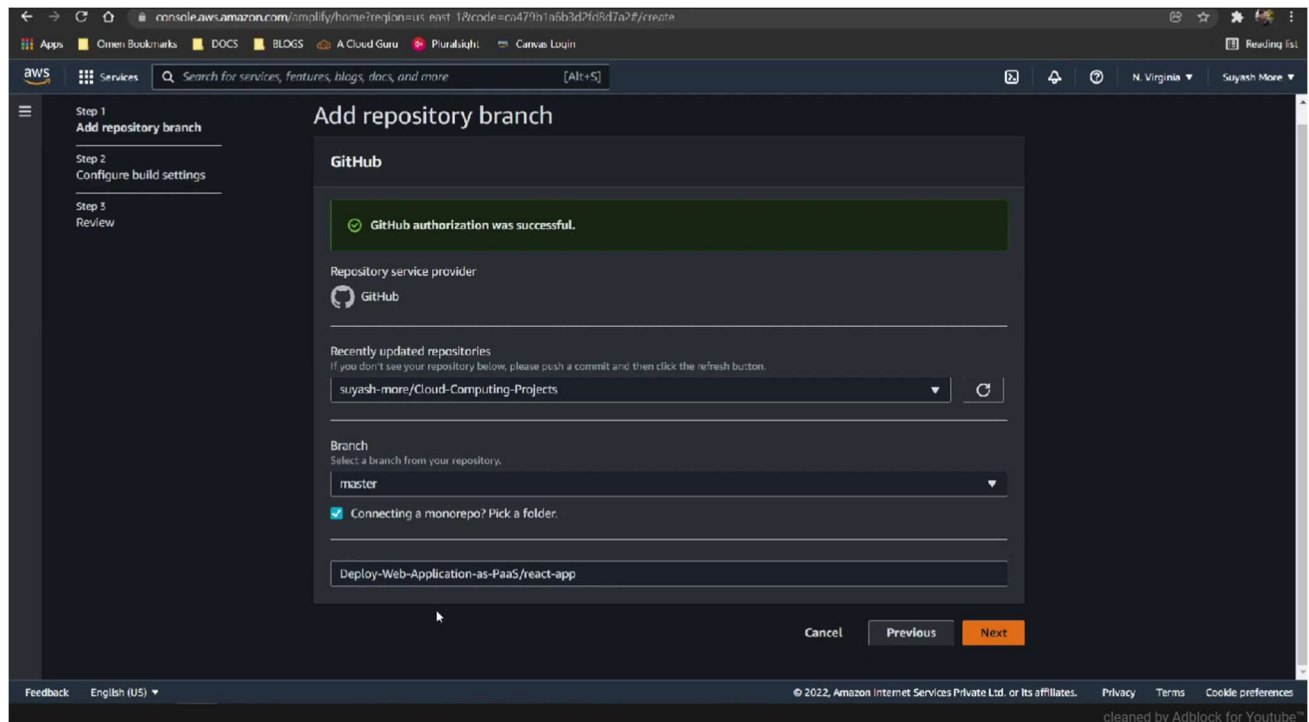
4. Click on Host a Web App.



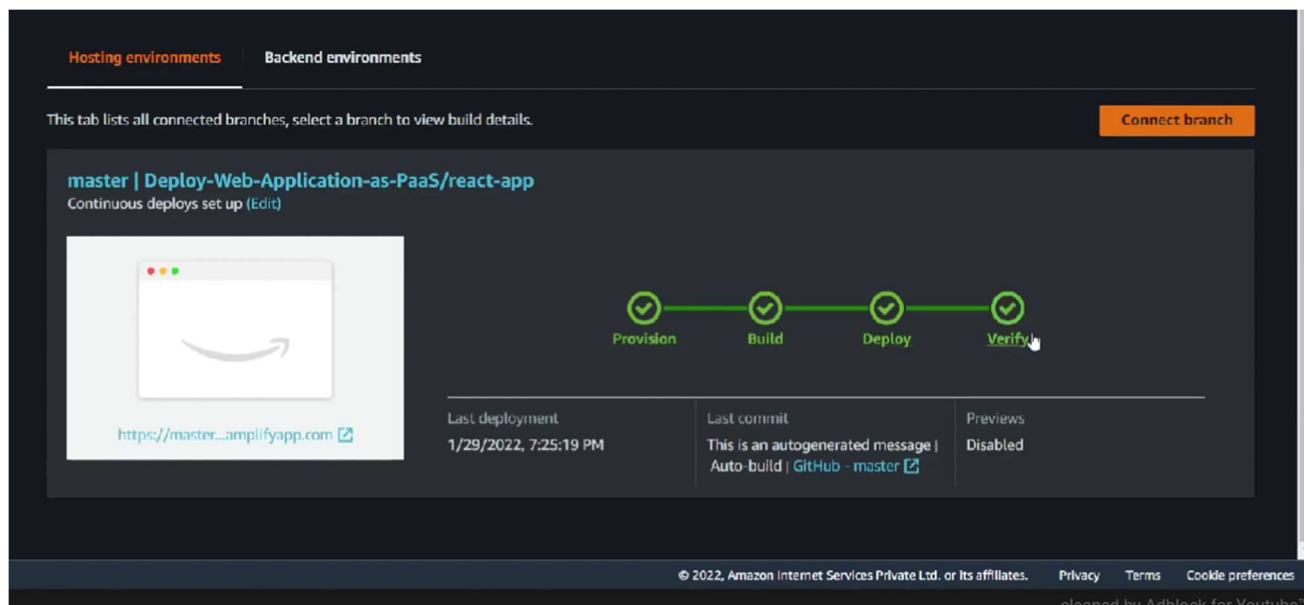
5. Then choose to launch it with Git-hub and authenticate your GitHub account for the same..



6. After that choose the Repository containing your source code (subfolder if needed)



7. After 4-Pipelines Launch the application with the default configurations provided by [AWS Amplify] (<https://aws.amazon.com/amplify/#:~:text=AWS%20Amplify%20is%20a%20set,as%20your%20use%20cases%20evolve.>)



8.Configurations may be different on type of framework / technology you are launching your application.

Conclusion:

There-fore we have successfully deployed our Web-Application on Amazon-Amplify which is one of the popular Platform as a service.