**Deploy Flask/Python Web Application Using Git,Github,Jenkins,Terraform,Route53 In Aws**

**what is Flask:**

* Flask is a small and lightweight python web application framework that provides useful tools and feature that make creating web application in python easier.
* It gives developers flexibility and it is a more accessible framework for new developer since you can build a web application quickly using only a single python file.

**What is python:**

* It is a computer programming language often used to build websites and software automate tasks, and conduct data analysis

**What is PIP:**

* PIP is a package manager python packages, or modules

**Note:**  if you have python version3.4 or later PIP is include by default.

**Pre-requisites:**

* AWS account
* IAM user
* Terminal
* Basic understand pf Python/Flask

**Services and Tools used:**

Aws service

* IAM
* VPC
* EC2
* ROUTE53

Devops Tools

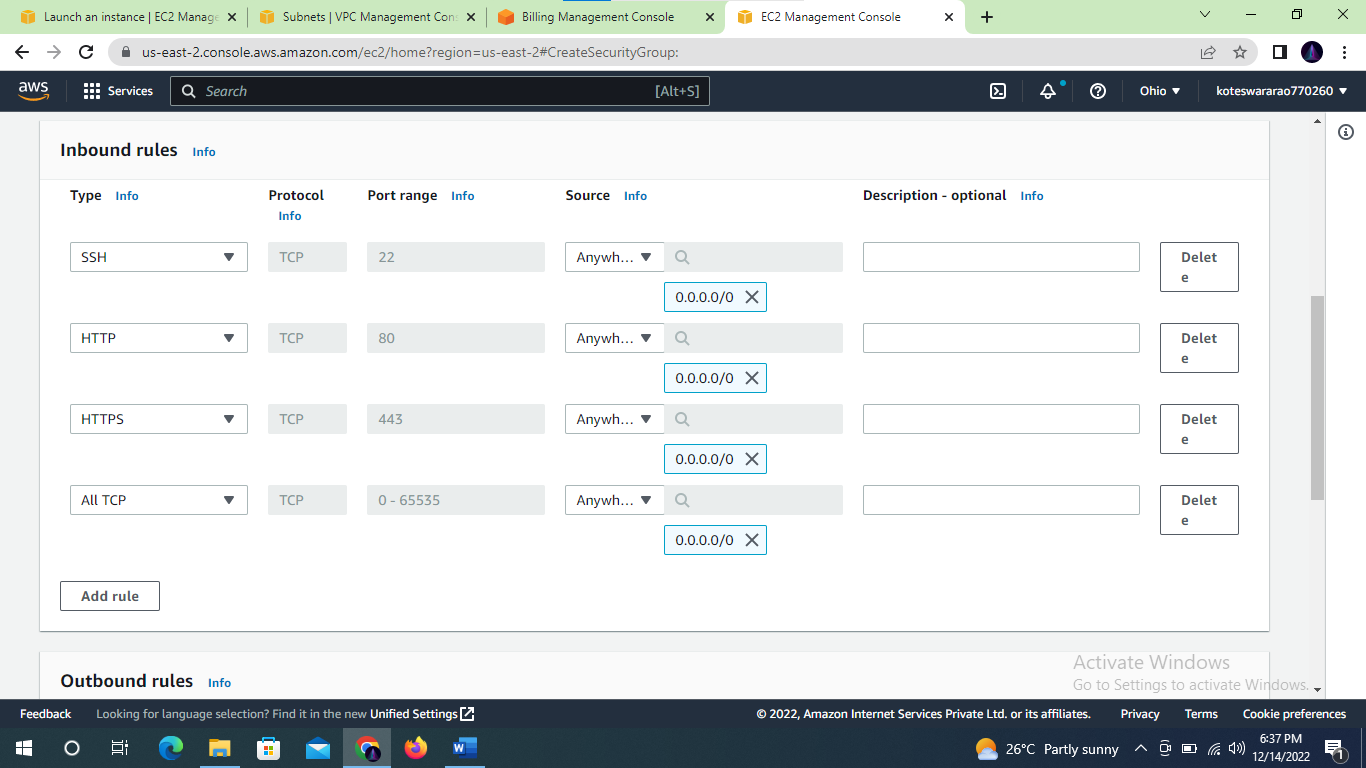
* Github
* Jenkins
* Terraform

**Step By Step Procedure:**

* Create and Login AWS Root Account
* Create a VPC along with subnets, route tables, internet gateway, elastic IP(if required),NACL(optional).



* Create security group with respective ports
* SSH = 22
* HTTP = 80
* HTTPS = 443
* TCP = 8080,5000,7000



* Create EC2 instance launch with SSH
* Amazon Linux, Ubuntu, RHEL
* Update ubuntu machine.

***ubuntu@ip-10-0-0-218:~$ sudo apt update***

* Full upgrade the machine.

***ubuntu@ip-10-0-0-218:~$ sudo apt-get full-upgrade -y***

* Install required packages or tools related for deployment project.

***ubuntu@ip-10-0-0-218:~$ sudo apt-get install python3-pip***

* Install git and clone the project source code from Github

***ubuntu@ip-10-0-0-218:~$ sudo git clone https://github.com/koteswararao798988/flask-library-app.git***

* Now, go to the source code directory

***ubuntu@ip-10-0-0-218:~$ cd flask-library-app/***

* Now, install requirements packages

***ubuntu@ip-10-0-0-218:~/flask-library-app$ pip3 install -r requirements.txt***

* Run Flask server

***ubuntu@ip-10-0-0-218:~/flask-library-app$ python3 app.py***

* Here, after running python app.py it will generate localhost IP address. We can’t access web app with that IP address. Then we want to edit the file app.py with some details.

***ubuntu@ip-10-0-0-218:~/flask-library-app$ sudo vi app.py***

* Go to very bottom of the file and paste this below text and save the file.

***app.run(host='0.0.0.0', port=8080, debug=True)***

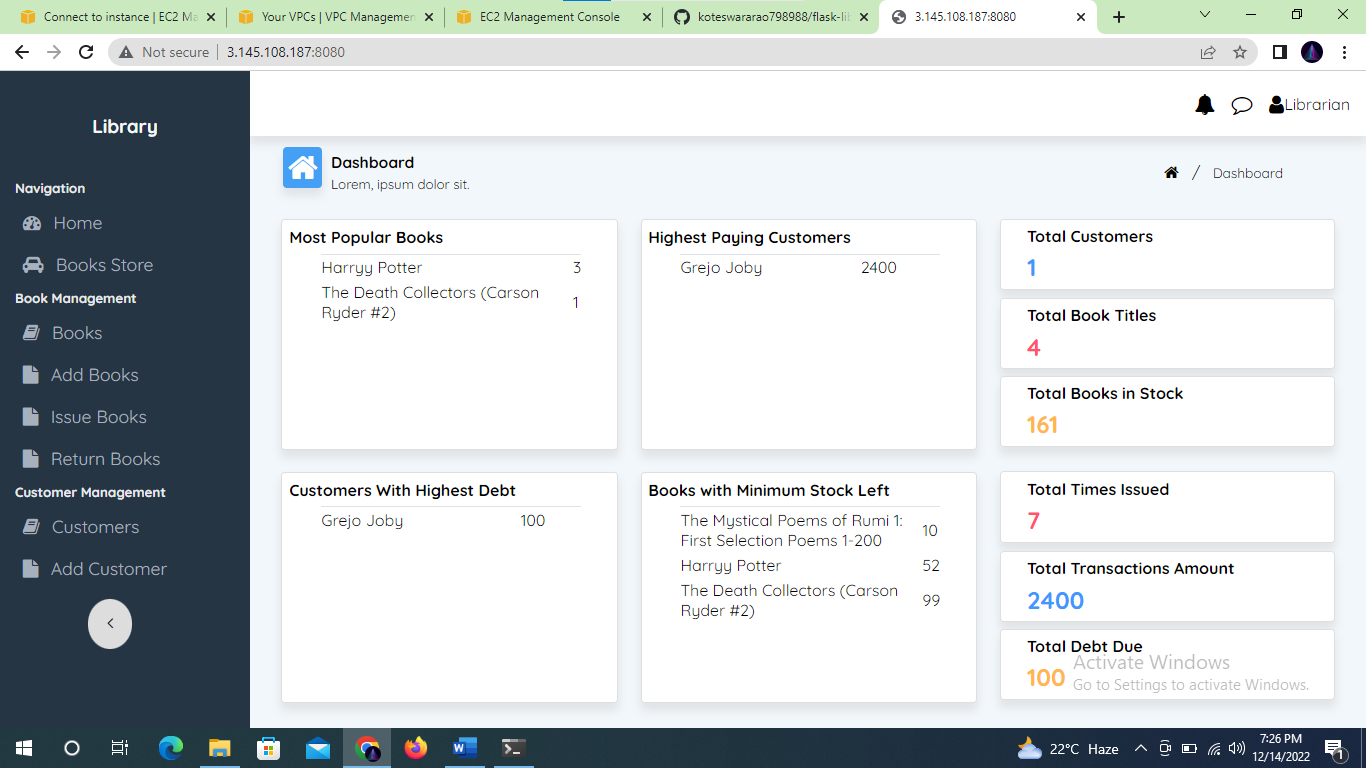
* Now, again run the Flask server by using below command

***ubuntu@ip-10-0-0-218:~/flask-library-app$ python3 app.py***

* Now , copy EC2 instance public IP and give port number and search in web browser.

***IP:8080***

* We will get output like this



**Deploy Flask/Python Web Application Using Jenkins In Aws**

**Installation of JENKINS:**

* Create EC2 instance launch with SSH
* AMI : Ubuntu, Linux, Red-hat
* Security-group: 22,8080,8000
* Update the machine

***ubuntu@ip-10-0-0-211:~$ sudo apt-get update***

* Install java version 11 runtime

***ubuntu@ip-10-0-0-211:~$ sudo apt install openjdk-11-jre-headless***

* Install require repo and packages for Jenkins

***ubuntu@ip-10-0-0-211:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \***

***/usr/share/keyrings/jenkins-keyring.asc > /dev/null***

***ubuntu@ip-10-0-0-211:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \***

***https://pkg.jenkins.io/debian-stable binary/ | sudo tee \***

***/etc/apt/sources.list.d/jenkins.list > /dev/null***

* Install Jenkins

***ubuntu@ip-10-0-0-211:~$ sudo apt-get install jenkins***

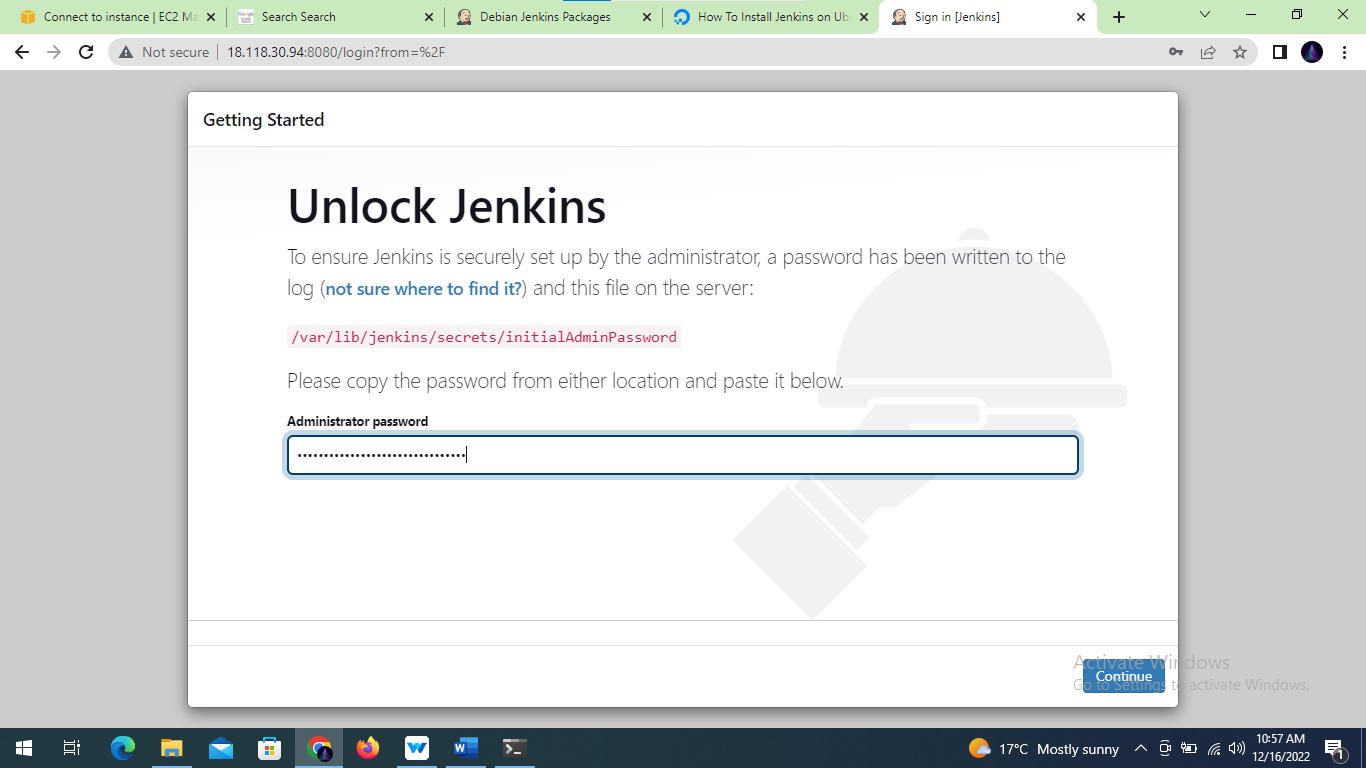
* Start the Jenkins

***ubuntu@ip-10-0-0-211:~$ sudo systemctl start jenkins***

* Copy the public IP of instance

Public IP:8080

* Copy the path for getting password

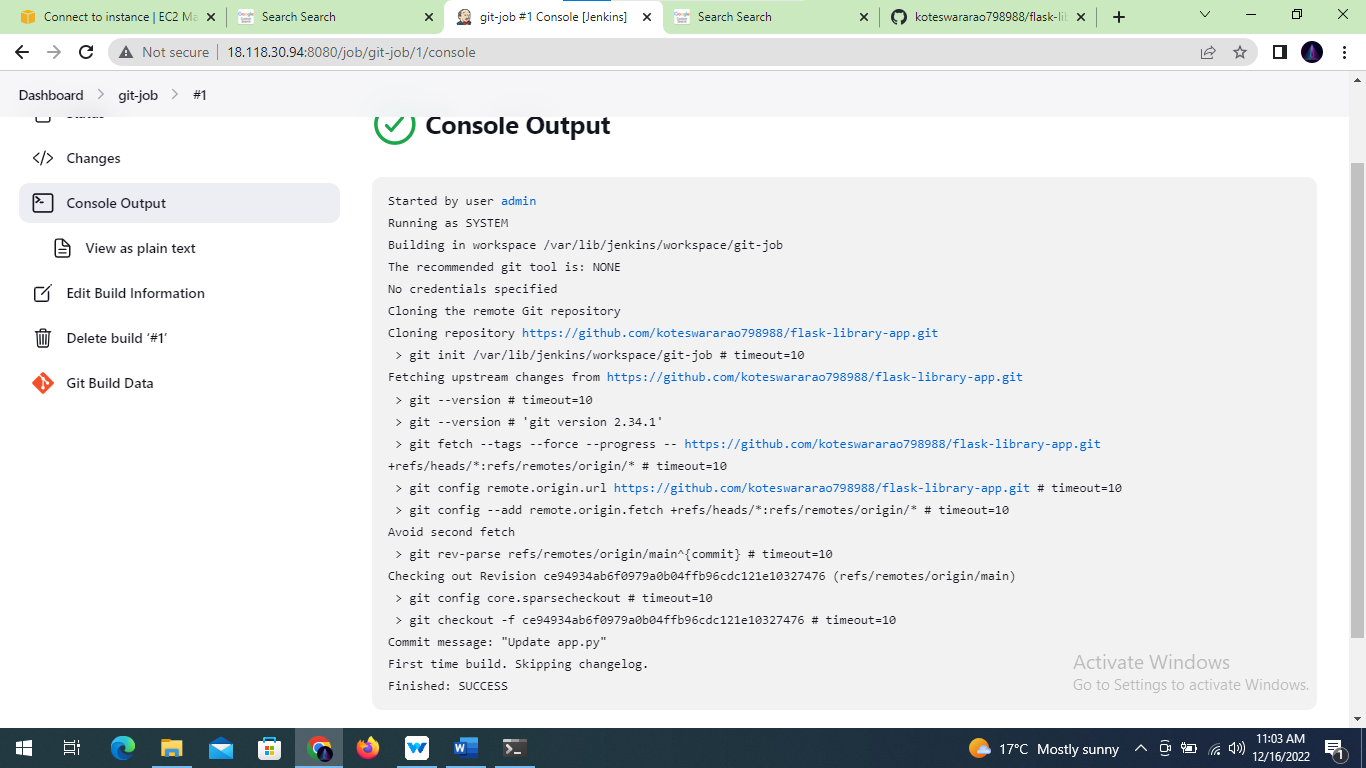


* Paste it in terminal and get a password and copy it

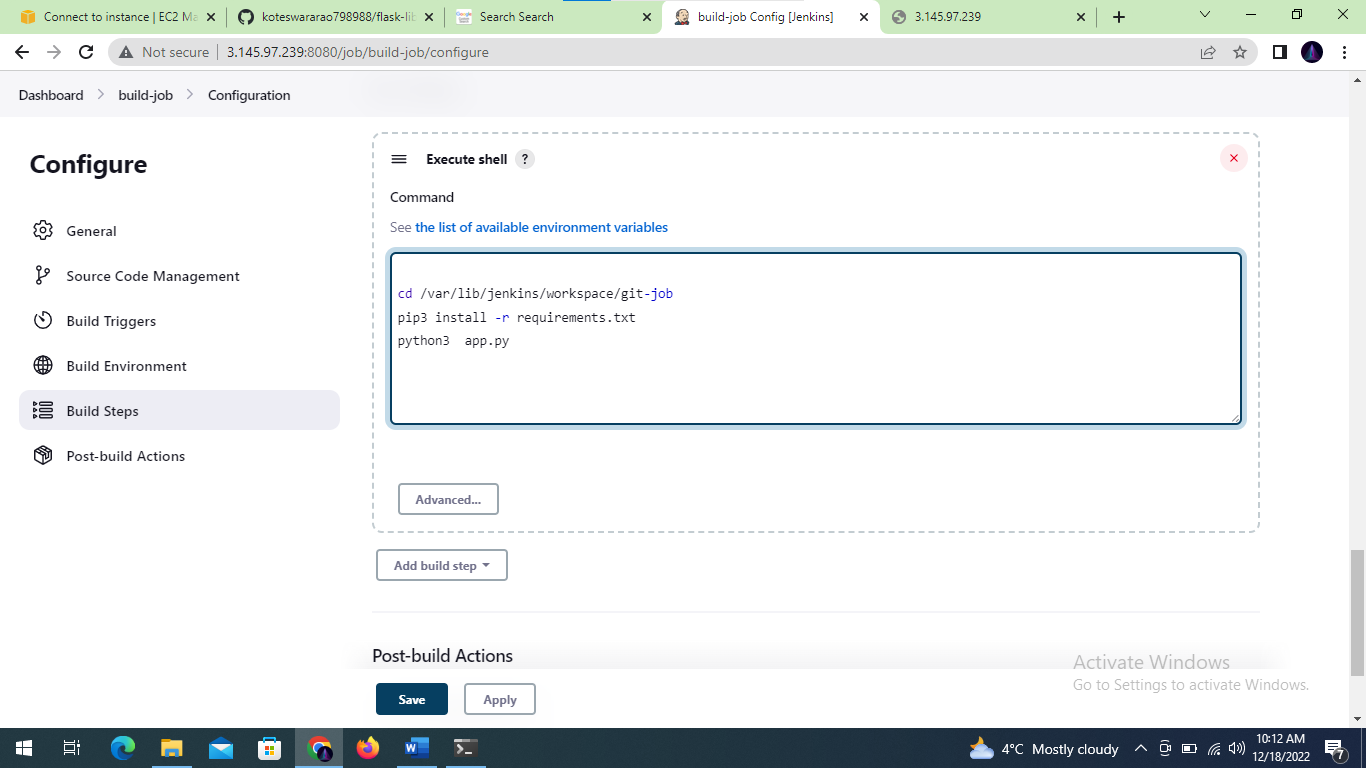
***ubuntu@ip-10-0-0-211:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword***

***d5496fc9d2854188a0d33ba171dd5537***

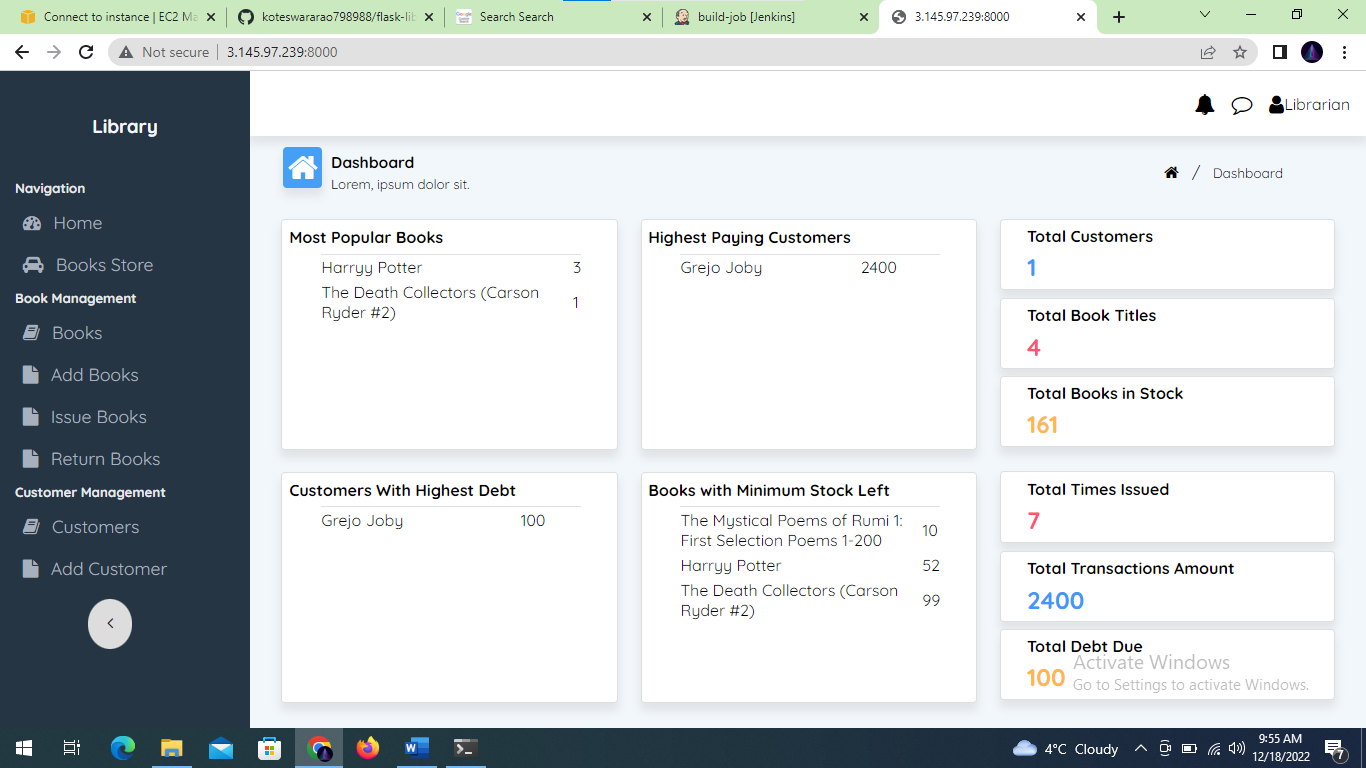
* Paste it in Jenkins page
* Give credentials
* In Jenkins page, create a first job in freestyle
* Clone the flask code from git hub
* Save the job and build now



* Create second job
* In exec shell, give commands



* Save it and build now
* Copy the public IP and browse it.



**Deploy Flask/Python Web Application Using Terraform In Aws**

**Install Terraform In Server:**

* Create EC2 instance Launch with SSH

AMI : Ubuntu, Amazon Linux , Red-Hat

* Install terraform into instance

***ubuntu@ip-10-0-0-150:~$ wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg***

* Create VPC with terraform script

**Main.tf**

# terraform update

# aws is my provider

provider "aws" {

access\_key = "AKIAX4NLJCBATVMI6WIF"

secret\_key = "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"

region = "us-east-2"

}

**VPC.tf**

# create vpc in aws

resource "aws\_vpc" "demovpc" {

cidr\_block = "20.0.0.0/16"

}

# create internet gateway

resource "aws\_internet\_gateway" "demoigw" {

vpc\_id = aws\_vpc.demovpc.id

}

# create subnet in vpc

resource "aws\_subnet" "public-subnet-11" {

vpc\_id = aws\_vpc.demovpc.id

cidr\_block = "20.0.0.0/24"

map\_public\_ip\_on\_launch = true

availability\_zone = "us-east-2a"

}

# create second subnet in vpc

resource "aws\_subnet" "private-subnet-11" {

vpc\_id = aws\_vpc.demovpc.id

cidr\_block = "20.0.1.0/24"

availability\_zone = "us-east-2b"

}

# create route table

resource "aws\_route\_table" "route" {

vpc\_id = aws\_vpc.demovpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.demoigw.id

}

}

# associating with subnet

resource "aws\_route\_table\_association" "rt1" {

subnet\_id = aws\_subnet.public-subnet-11.id

route\_table\_id = aws\_route\_table.route.id

}

**Security groups.tf**

# create security group

resource "aws\_security\_group" "demosg" {

vpc\_id = aws\_vpc.demovpc.id

# inbounnd rule

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

# https

ingress {

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

# ssh

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

# out bound rules

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

**Data.sh**

**Ec2.tf**

# create ec2 instance

resource "aws\_instance" "demoinstance" {

ami = "ami-0283a57753b18025b"

instance\_type = "t2.micro"

key\_name = "workshop"

subnet\_id = aws\_subnet.public-subnet-11.id

vpc\_security\_group\_ids = ["${aws\_security\_group.demosg.id}"]

associate\_public\_ip\_address = true

user\_data = file("data.sh")

}

**Terraform plan** :- the terraform plan command lets you to preview the actions terraform would take to modify your infrastructure, or save a speculative plan which you can apply later.

***ubuntu@ip-10-0-0-150:~$ terraform plan***

**Terraform apply :-** the terraform apply command executes the actions proposed in a terraform plan. It is used to deploy your infrastructure.

***ubuntu@ip-10-0-0-150:~$ terraform apply --auto-approve***