DEPLOY FLASK OR PYTHON WEB APPLICATION ISING GIT, GITHUB, JENKINS, TERRAFORM, IN AWS

Prepared by,
T.AJAY

DEPLOY FLASK OR PYTHON WEB APPLICATION ISING GIT, GITHUB, JENKINS, TERRAFORM, ROUTE53 IN AWS

METHOD1:

DEPLOY FLASK OR PYTHON WEB APPLICATION MANUALLY BY USING AWS RESOURCES:

What is FLASK?

1. Flask is a small and ligthtweight python web application framework that provides useful tools and feature that make creating web application in python easier.
2. 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?

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

What is PIP?

1. PIP is a package manager python packages, or modules.

NOTE: If you have python version 3.4 or later PIP is include by default.

PIP stands for Pip install packages.

PRE-REQUISTES:

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

SERVICES AND TOOLS USED:

AWS SERVICES:

- IAM
- VPC
- EC2
- ROUTE53

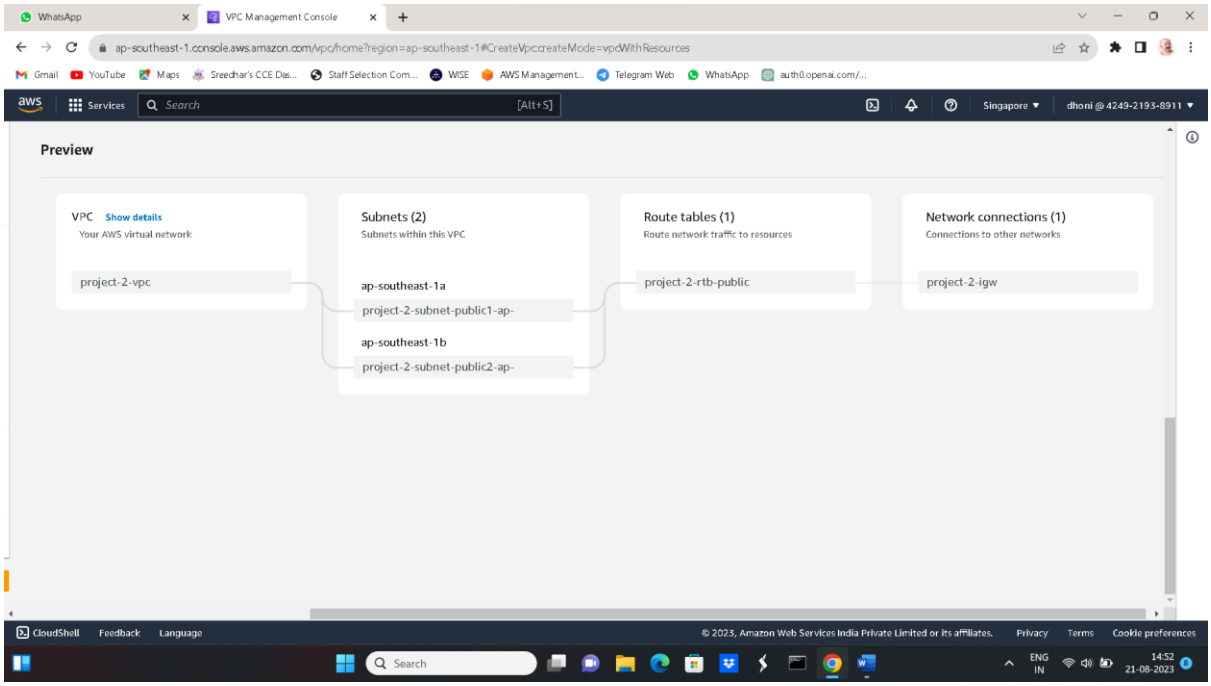
DEVOPS TOOLS:

- Github
- Jenkins
- Terraform

STEP BY STEP PROCEDURE:

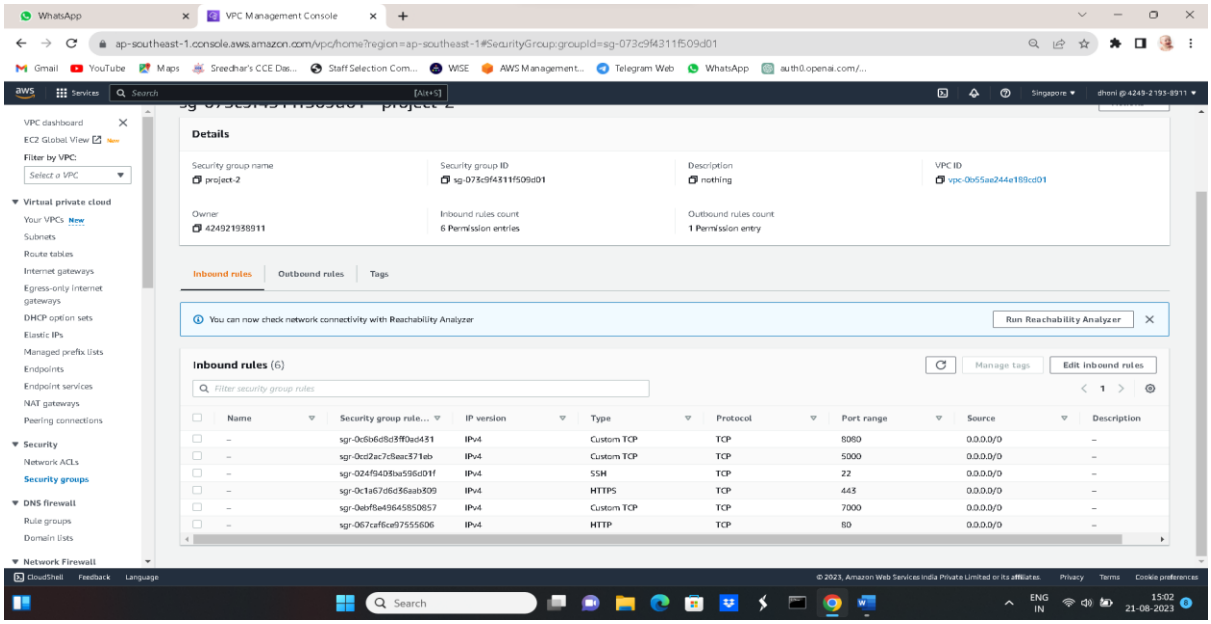
1.Create and login AWS Root Account.

- Create a VPC along with subnets, route tables, internet gateway,elastc ip if required.



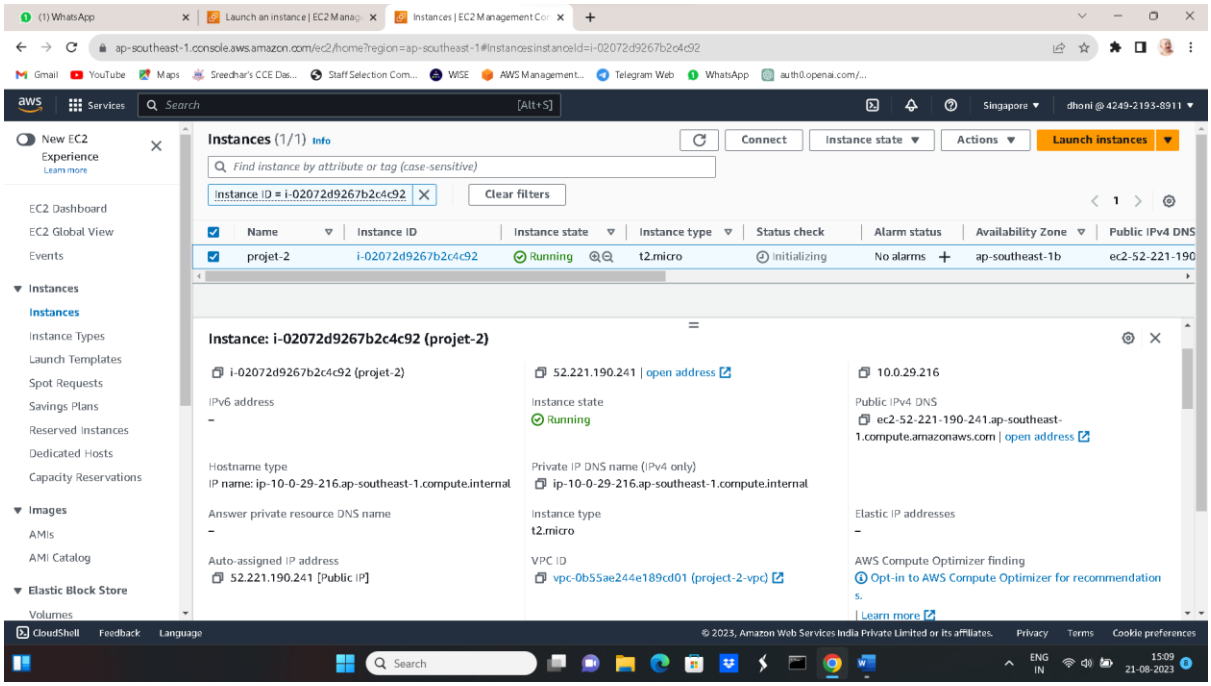
2. Create security group with respective ports

- Ssh-22
- http-80
- httpd-443
- tcp-8080,5000,7000



3. Create EC2 instance launch with SSH

- Amazon Linux, Ubuntu, RHEL



4. Update Ubuntu machine

- sudo apt update

5. Full upgrade the machine.

- sudo apt-get full-update –y

6. Install required packages or tools related for deployment project.

- sudo apt-get install python3-pip

7. Install git and clone the project source code from Github

- sudo git clone https://github.com/ajay77777777/flask-library-app.git

8. Now, go to the source code directory

- cd flask-library-app/

9. Now, install requirements packages

- pip3 install –r requirements.txt

10. Run Flask server

- python3 app.py

11. Here, after running python app.py it will generate localhost IP address. We cant access web app with that IP address. Then we want to edit the file app.py with some details.

sudo vi app.py

12. 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)

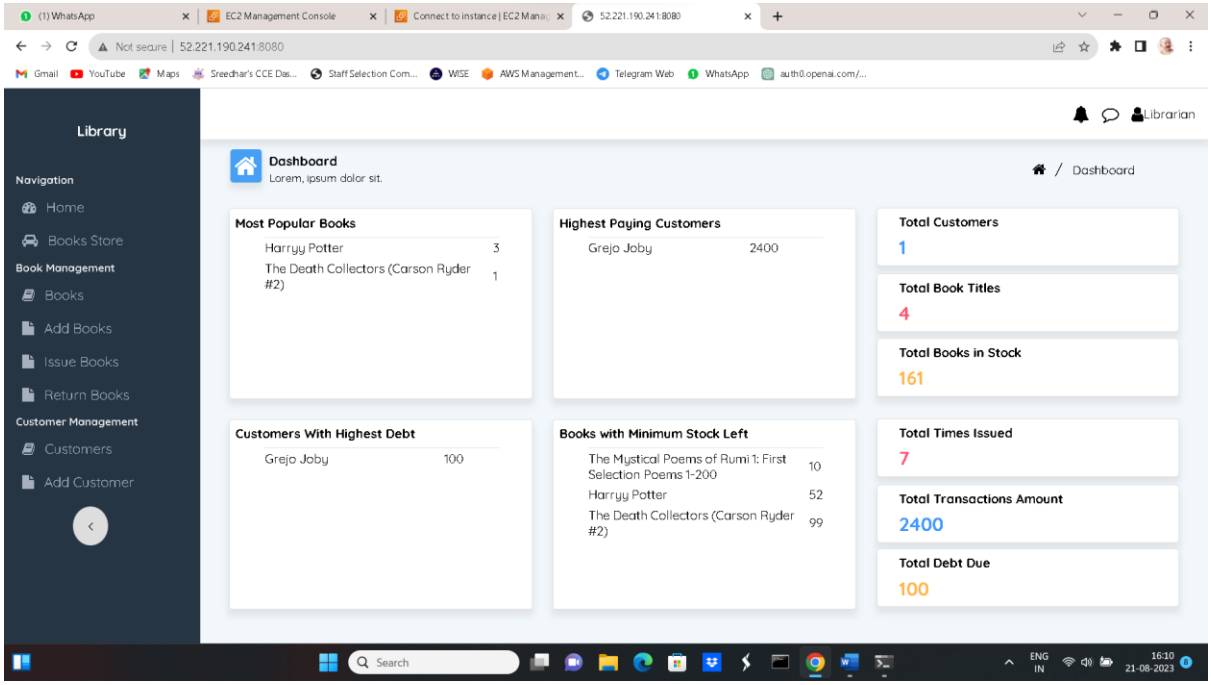
13. Now, again run the Flask server by using below command

    - flask-library-app$python3 app.py

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

    IP:8080

15. We will get output like this.



METHOD 2:

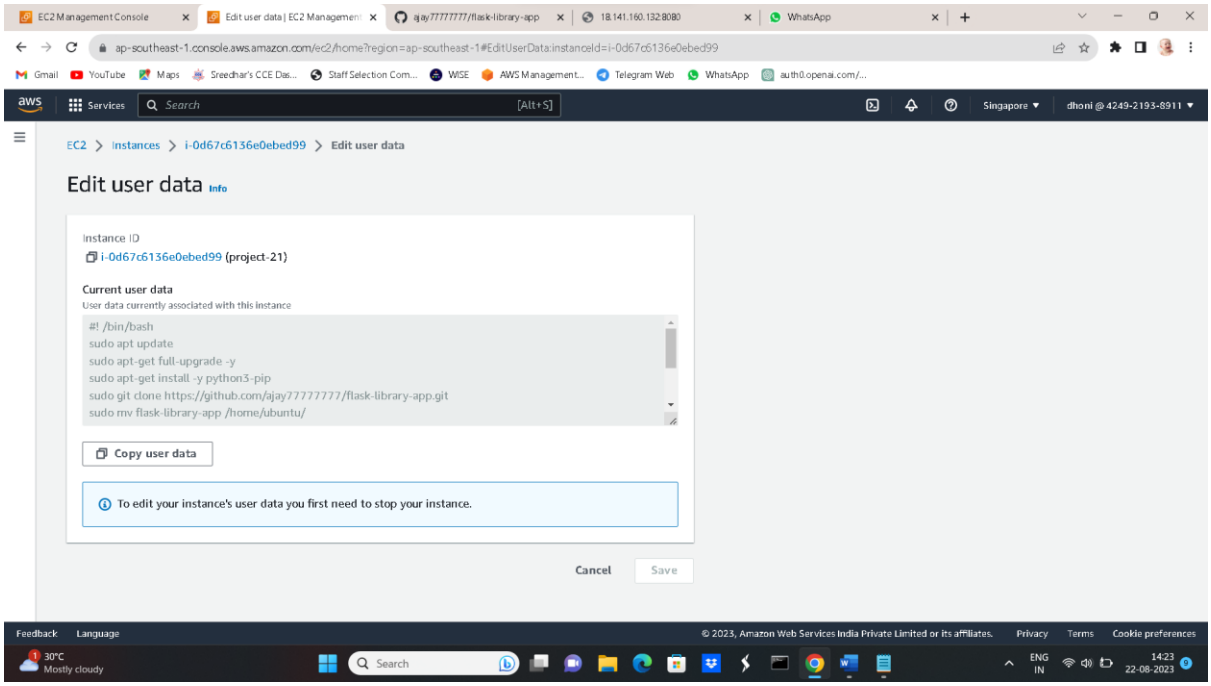
DEPLOYING PYTHON WEB APPLICATION USING USERDATA/BASHSCRIPT.

- First login to the AWS account with respective credentials and go to the EC2 services.

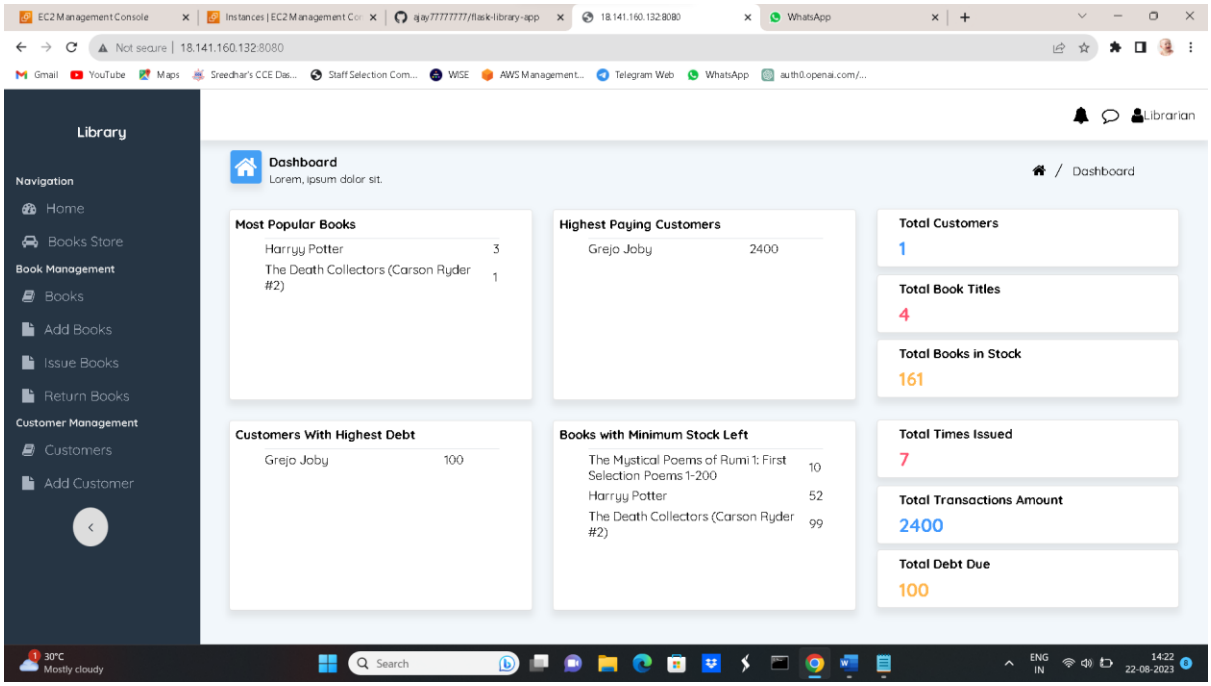
- Create a VPC along with subnets, route tables, internet gateway, elastic IP (if required), NACL (optional).
- Now create and launch EC2 instance by selecting Ubuntu or amazonlinux2 versions with respective ports, they are
  SSH(22), HTTP (80), HTTPS (443) and Custom (8080).
- Now create the bash script at user data field with respective commands, those are

UBUNTU

- **#! /bin/bash**
- **sudo apt update**
- **sudo apt-get full-upgrade -y**
- **sudo apt-get install -y python3-pip**
- **sudo git clone https://github.com/ajay77777777/flask-library-app.git**
- **sudo mv flask-library-app /home/ubuntu/**
- **cd /home/ubuntu**
- **cd flask-library-app/**
- **pip3 install -r requirements.txt**
- **python3 app.py**



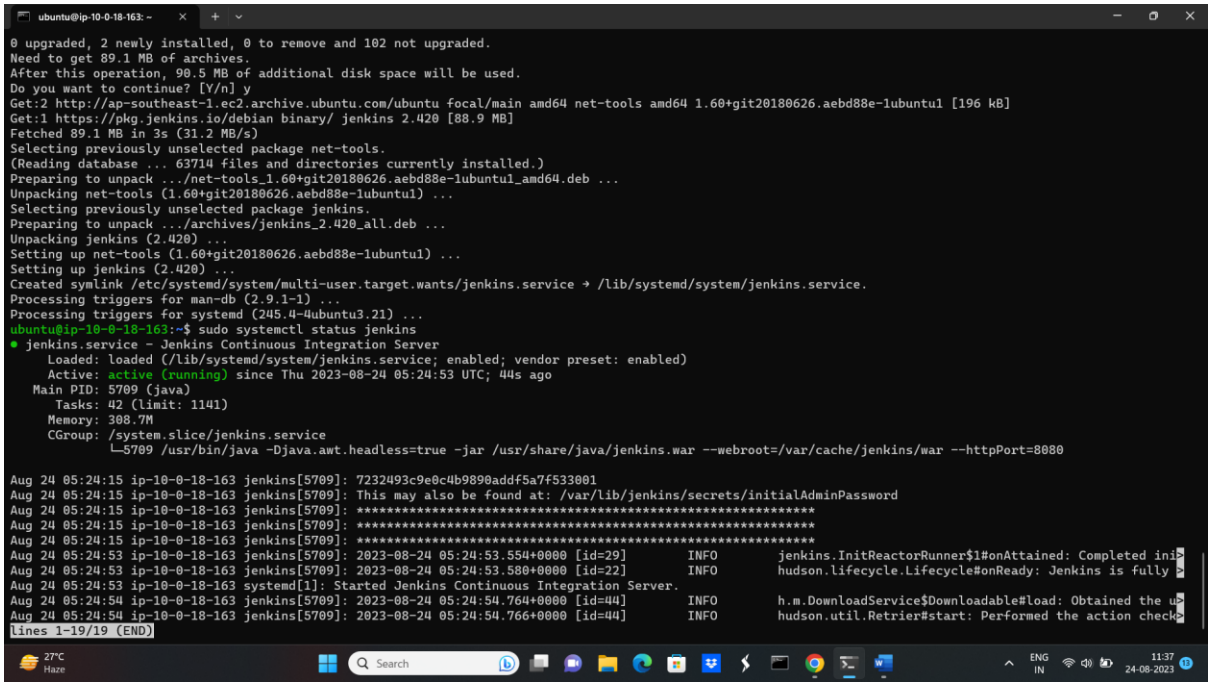
- Now browse the public ip address along with the port number 8080.



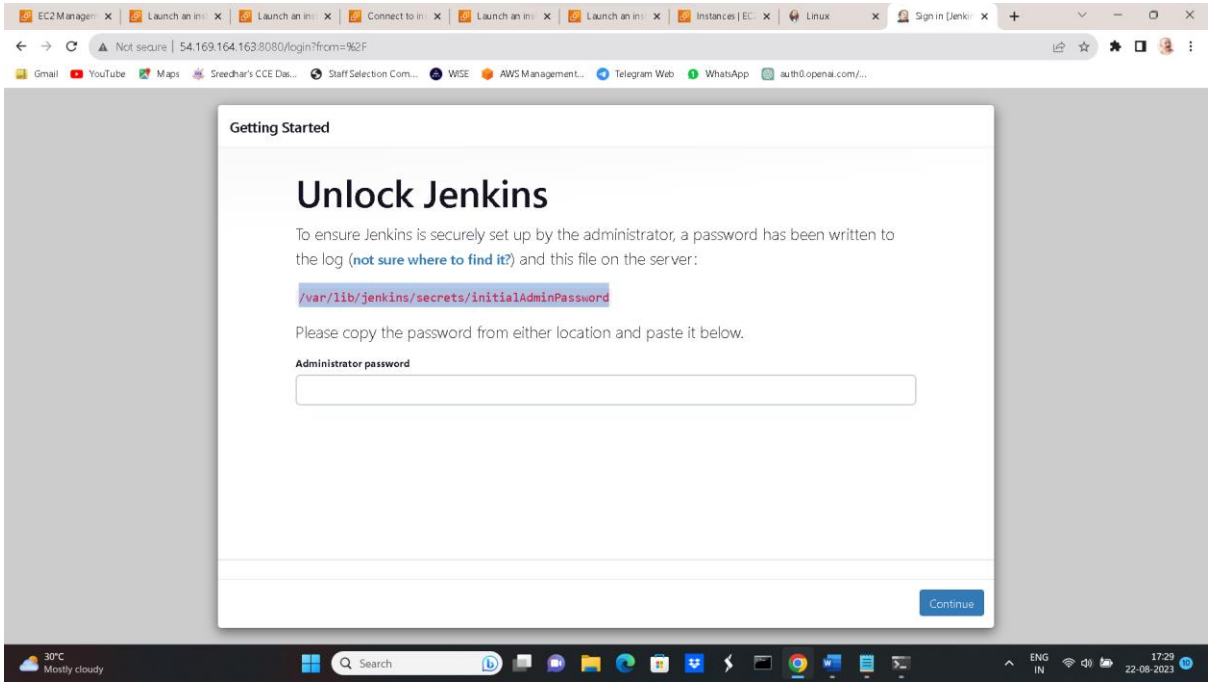
METHOD 3:

DEPLOYING PYTHON WEB APPLICATION USING GIT, GITHUB AND JENKINS:

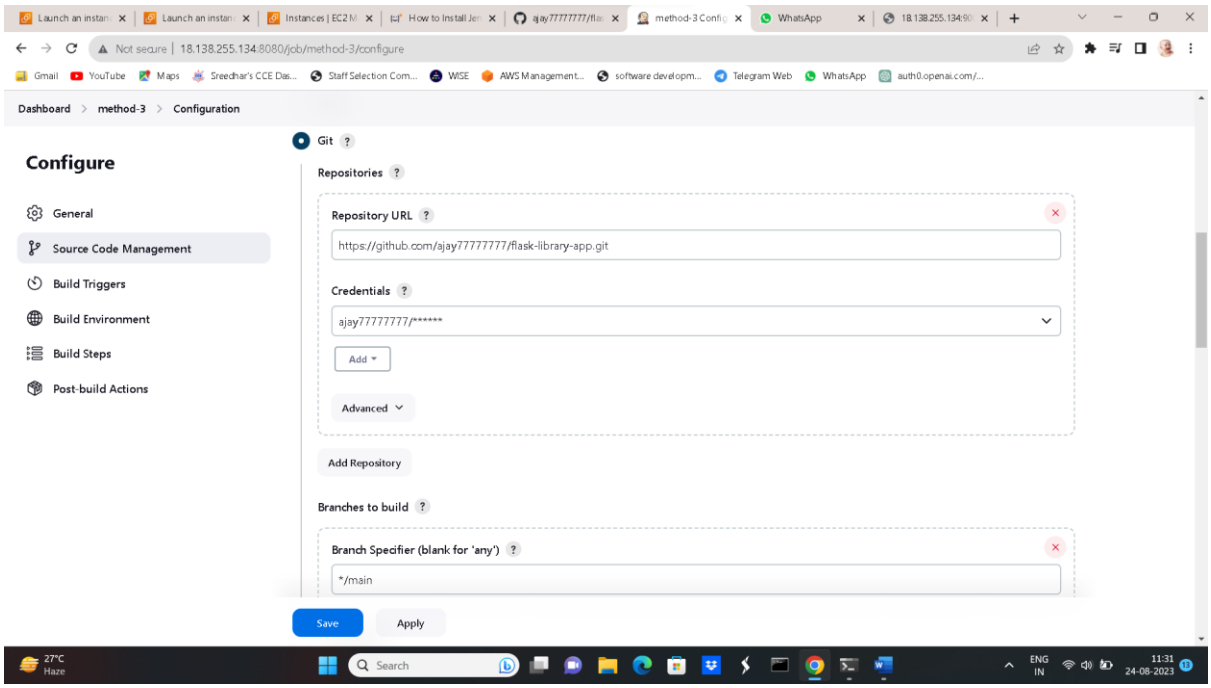
- First login to the AWS account with respective credentials and go to the EC2 services.
- Create a VPC along with subnets, route tables, internet gateway, elastic IP (if required), NACL (optional).
- Now create and launch EC2 instance by selecting Ubuntu or amazonlinux2 versions with respective ports, they are SSH(22), HTTP(80), HTTPS(443), and Custom(8080), (9000).
- Now launch the instance by using the SSH command with Git-bash or putty.
- For Ubuntu linux server use this commands as, This is the Debain package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system.
    - curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
      > /usr/share/keyrings/jenkins-keyring.asc > /dev/null
    - echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
      > https://pkg.jenkins.io/debian binary/ | sudo tee \
      > /etc/apt/sources.list.d/jenkins.list > /dev/null
    - sudo apt-get update
    - sudo apt install openjdk-11-jdk
    - sudo apt-get install Jenkins



- Next launch the Jenkins with browsing public ip along with the port number 8080.



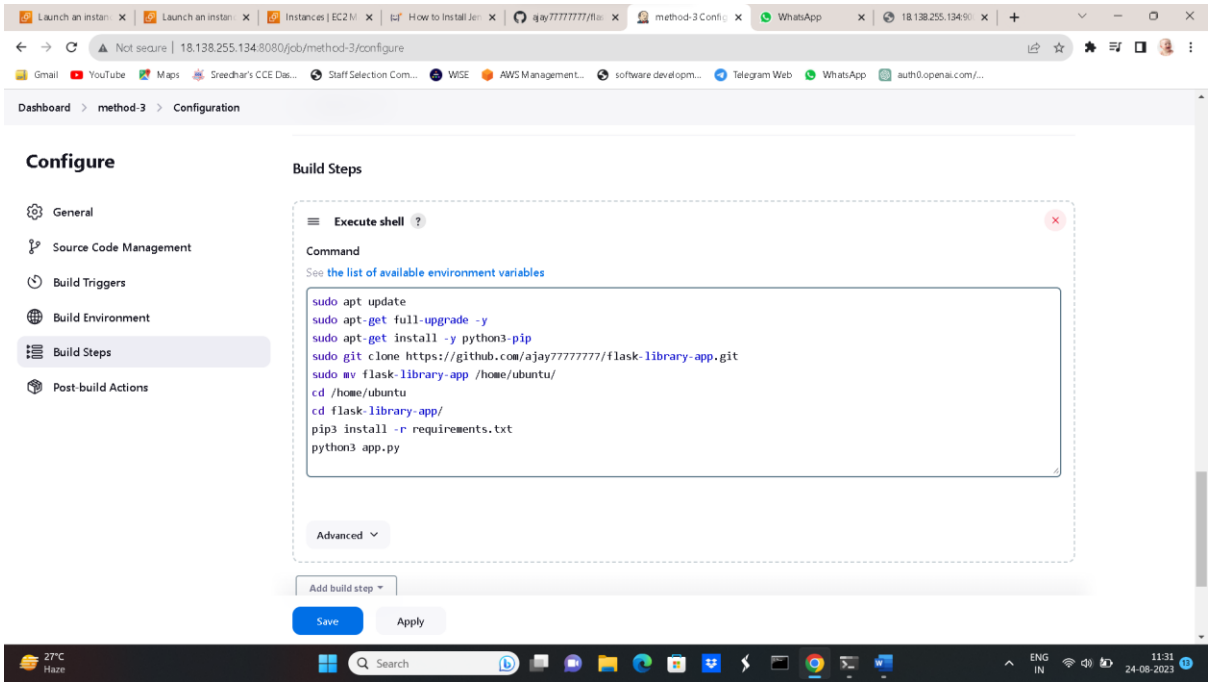
- Now create the job for cloning and clone the python web application repository at the git field where placed under the job and mention branch name also and build the job.



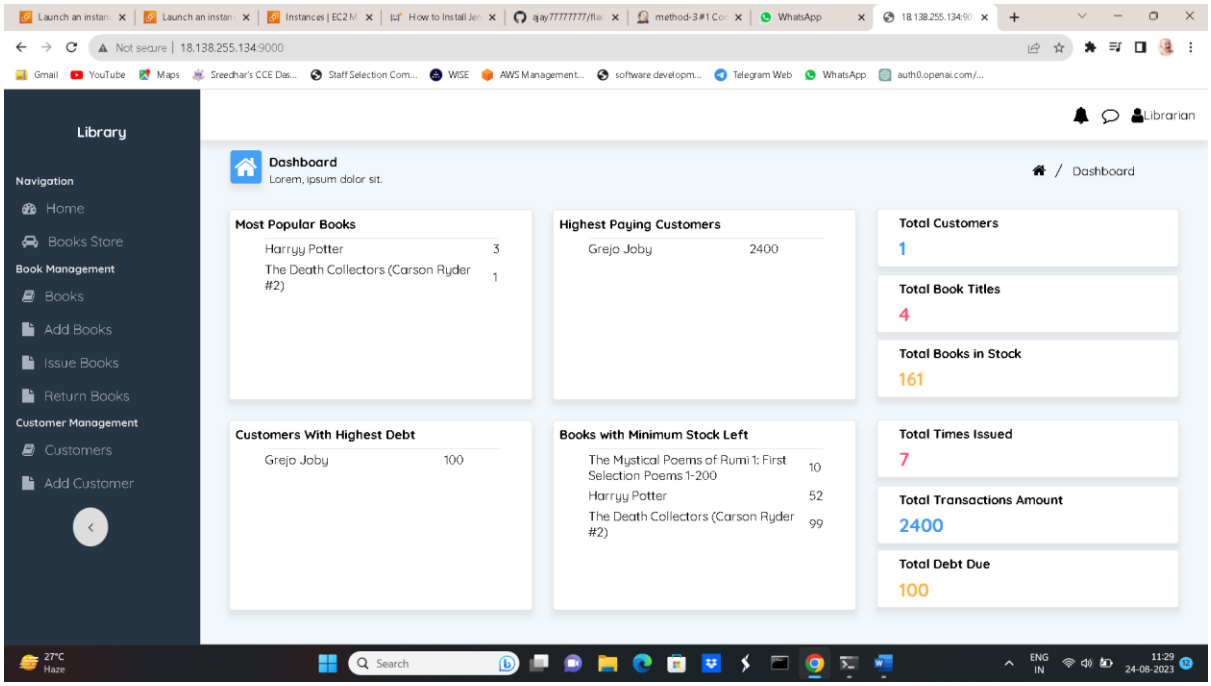
- Next create build job in this copy the clone while creating build job and write the bash script for python web application deployment in the under build trigger field at Execute-shell option and build the job.
- Before going to execute the script you need to add Jenkins user for sudo permissions in the sudoers file under /etc directory.

UBUNTU:

- sudo apt update
- sudo apt-get full-upgrade -y
- sudo apt-get install -y python3-pip
- sudo git clone https://github.com/ajay77777777/flask-library-app.git
- sudo mv flask-library-app /home/ubuntu/
- cd /home/ubuntu
- cd flask-library-app/
- pip3 install -r requirements.txt
- python3 app.py



- Finally just go with public ip of launched instance copy it and browse it along with port 9000.



METHOD 4:

DEPLOYING PYTHON WEB APPLICATION USING GIT,GITHUB AND TERRAFORM:

- ✓ First login to the AWS account with respective credentials and go the EC2 services.
- ✓ Create a VPC along with subnets, route tables
- ✓ Now create and launch EC2 instance by selecting Ubuntu or amazonlinux2 versions with
- ✓ Now launch the instance by using the SSH command with Git-bash or putty.
- ✓ Now install the terraform by using below commands.
    - Wget –O-https://apt.releases.hashicorp.com/gpg –dearmor | sudo tee /usr/share/keyrings/hashicorp (Search from Google)
- ✓ Create terraform files for creating resources in aws by using fallowing terraform script

#creating vpc.tf

```
resource "aws_vpc" "demovpc" {
cidr_block= "10.0.0.0/16"
instance_tenancy = "default"
tags = {
Name="demovpc"
}
}
```

#creating subnet.tf

```
 resource "aws_subnet" "public-subnet-1" {
 vpc_id                  = aws_vpc.demovpc.id
 cidr_block              = "10.0.1.0/24"
 map_public_ip_on_launch = true
 availability_zone       = "us-east-1a"
 tags = {
   Name = "Web Subnet 1"
 }
}
```

#creating security group.tf

```
 resource "aws_security_group""demosg" {
vpc_id=aws_vpc.demovpc.id
ingress{
from_port=80
to_port=80
protocol="TCP"
cidr_blocks=["0.0.0.0/0"]
}
ingress{
from_port=8080
to_port=8080
protocol="TCP"
cidr_blocks=["0.0.0.0/0"]
}
ingress{
```

```
from_port=9000
to_port=9000
protocol="TCP"
cidr_blocks=["0.0.0.0/0"]
}
ingress{
from_port=22
to_port=22
protocol="TCP"
cidr_blocks=["0.0.0.0/0"]
}
egress{
from_port=0
to_port=0
protocol="-1"
cidr_blocks=["0.0.0.0/0"]
}
tags={
Name="web sg"
}
}
```

#creating route table.tf file

```
 resource "aws_route_table""route"{
vpc_id=aws_vpc.demovpc.id
tags={
Name="route to internet"
}
}
resource "aws_route""default_route"{
route_table_id =aws_route_table.route.id
destination_cidr_block = "0.0.0.0/0"
gateway_id=aws_internet_gateway.demogateway.id
}
resource"aws_route_table_association""rtl"{
subnet_id=aws_subnet.public-subnet-1.id
route_table_id=aws_route_table.route.id
}
```

#creating internet gateway igw.tf

```
 resource "aws_internet_gateway" "demogateway" {
vpc_id = aws_vpc.demovpc.id
}
```

#creating key pair key.tf

```
resource "aws_key_pair" "ajay" {
key_name = "ajay"
public_key = tls_private_key.rsa.public_key_openssh
}
resource "tls_private_key" "rsa" {
algorithm = "RSA"
rsa_bits =4096
}
resource "local_file" "tf-key" {
content = tls_private_key.rsa.private_key_pem
filename = "ajay"
}
```

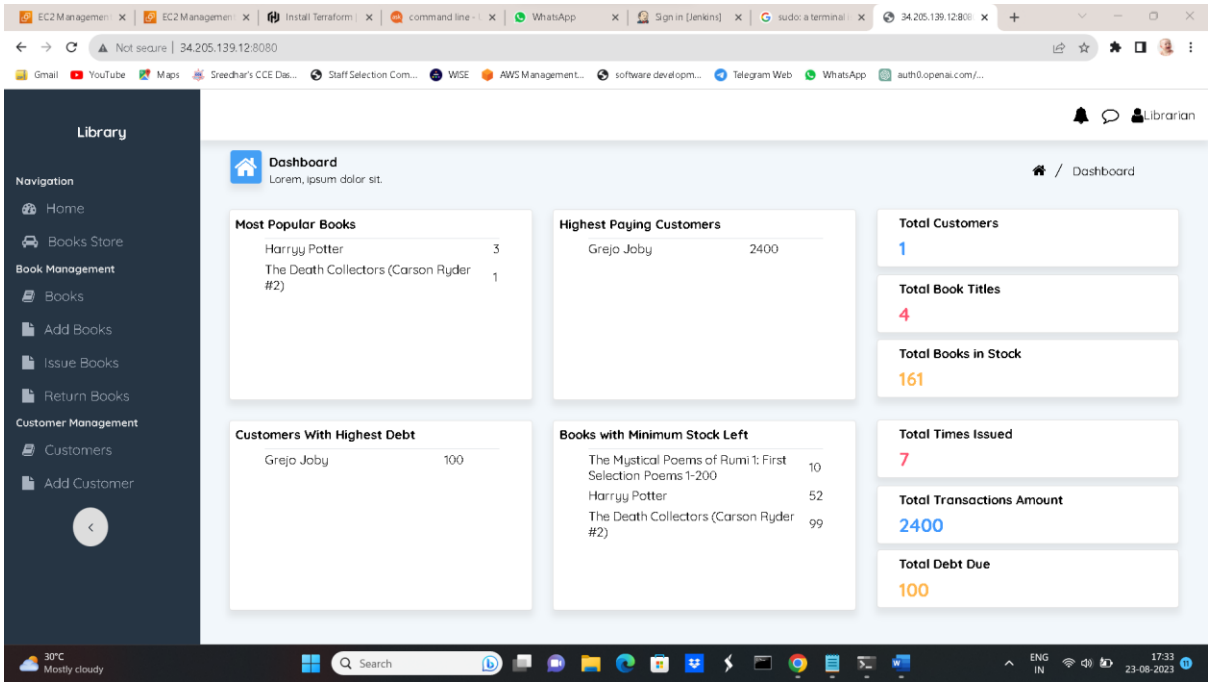
#creating user data data.sh

```
 #! /bin/bash
sudo apt update
sudo apt-get full-upgrade -y
sudo apt-get install -y python3-pip
sudo git clone https://github.com/ajay77777777/flask-library-app.git
sudo mv flask-library-app /home/ubuntu/
cd /home/ubuntu
cd flask-library-app/
pip3 install -r requirements.txt
python3 app.py
```

#creating ec2 instance ec2.tf

```
resource "aws_instance""demoinstance2" {
ami="ami-0261755bbcb8c4a84"
instance_type="t2.micro"
key_name="ajay"
vpc_security_group_ids=["${aws_security_group.demosg.id}"]
subnet_id=aws_subnet.public-subnet-1.id
associate_public_ip_address=true
user_data=file("data.sh")
tags={
Name="my public insatance2"
}
```

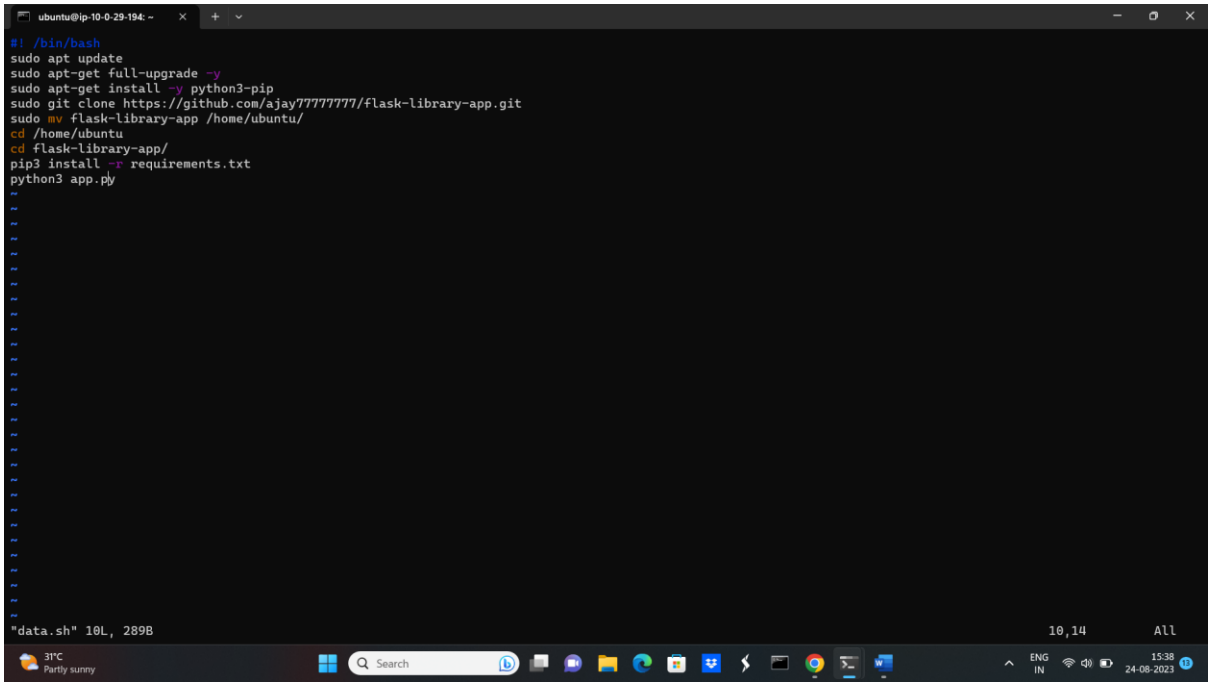
- ✓ After creating terraform files give the fallowing commands
  - **TERRAFORM INIT**
  - **TERRAFORM VALIDATE**
  - **TERRAFORM APPLY**
- ✓ Now go to copy public ip of created server and browse it along with port number - 8080.



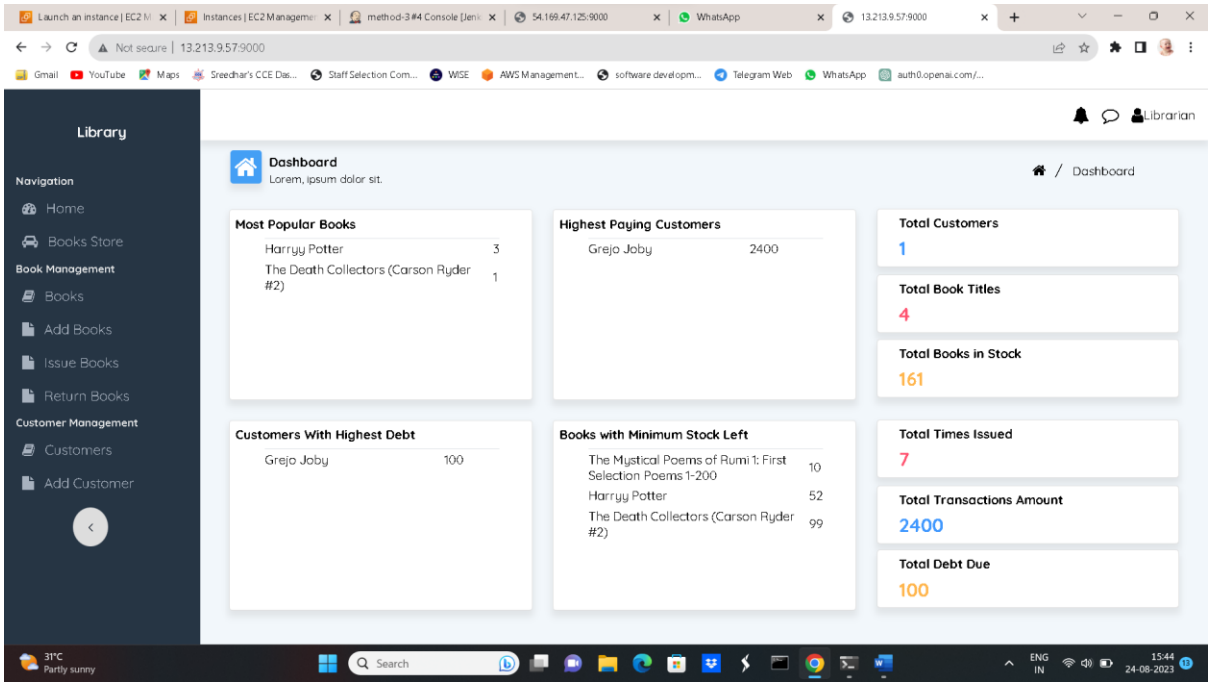
METHOD 5:

DEPLOYING PYTHON WEB APPLICATION USING GIT,GITHUB AND BASH SCRIPT:

- ✓ First login to the AWS account with respective credentials and go the EC2 services.
- ✓ Create a VPC along with subnets, route tables
- ✓ Now create and launch EC2 instance by selecting Ubuntu or amazonlinux2 versions with
- ✓ Now launch the instance by using the SSH command with Git-bash or putty.
- ✓ Now create bash file for bash scripting by using vi command and the bash script for UBUNTU server
    - #!/bin/bash
    - sudo apt update
    - sudo apt-get full-upgrade –y
    - git clone https://github.com/GOUSERABBANI44/flask-library-app.git
    - sudo apt-get install python3-pip –y
    - cd flask-library-app
    - pip3 install –r requirements.text
    - nohup python3 –u ./app.py &



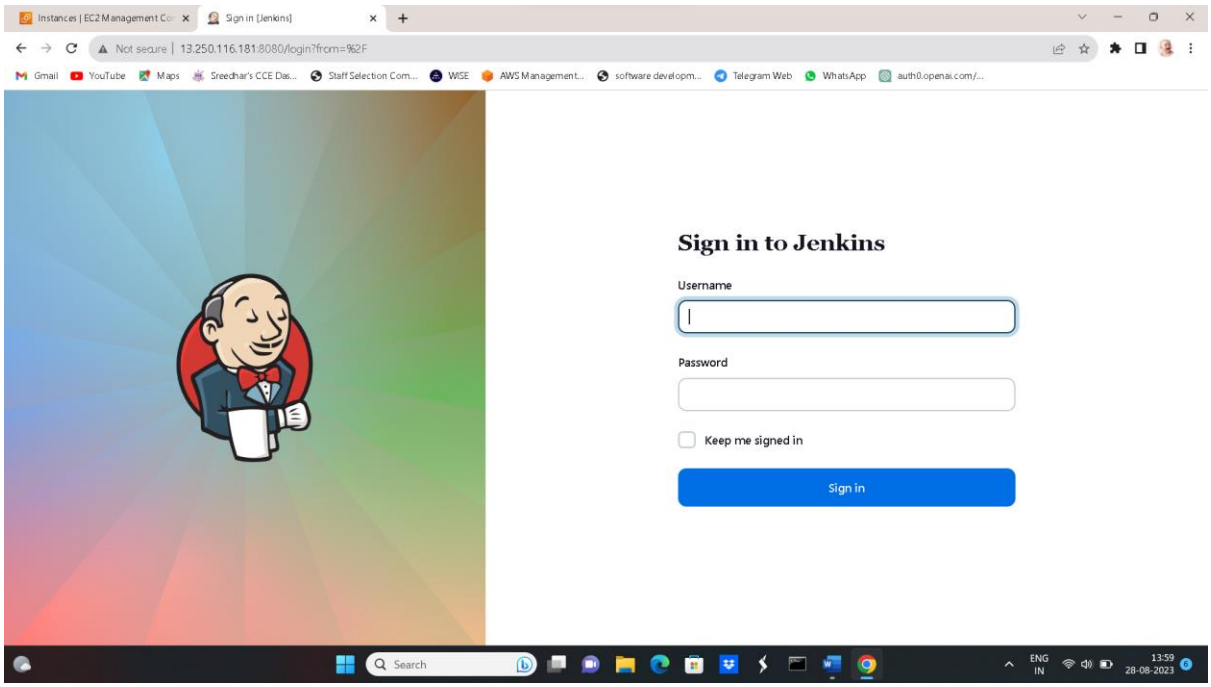
- now you need to add the execute permissions for the bash script file
    - chmod +x data.sh
- now run the bash script file by using ./data.sh command
- Finally just go with public ip of launched instance copy it and browse it along with port 9000.



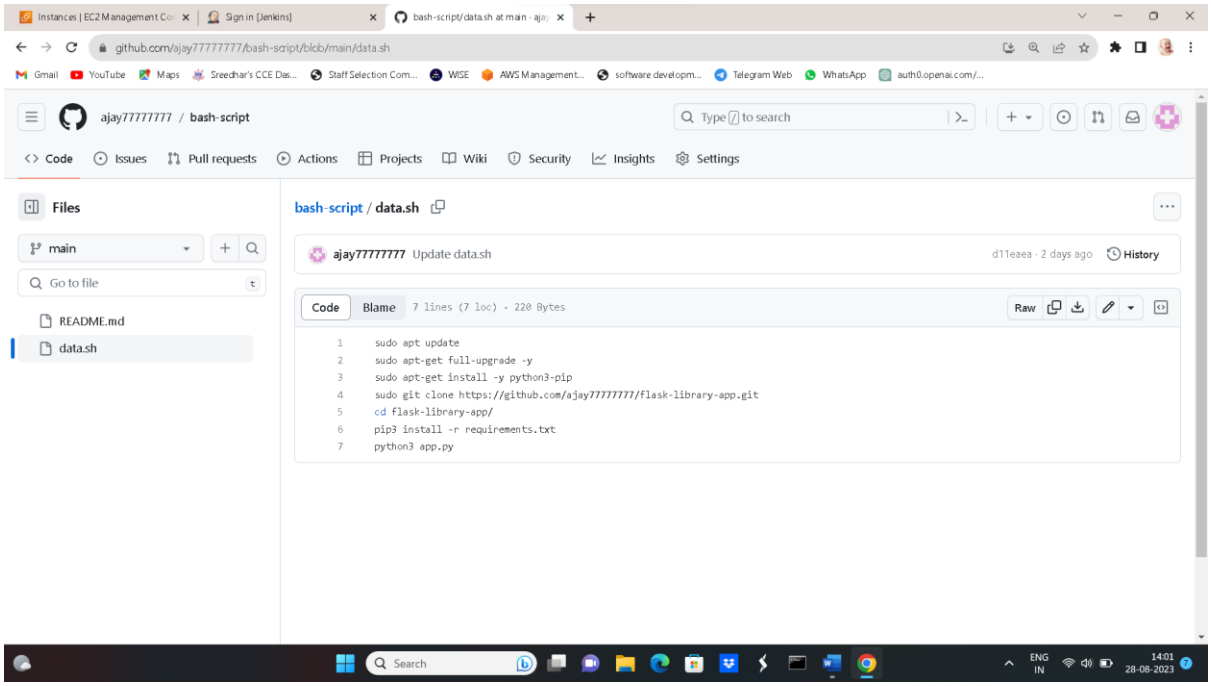
METHOD 6:

DEPLOYING PYTHON WEB APPLICATION USING GIT, GITHUB and Jenkins.

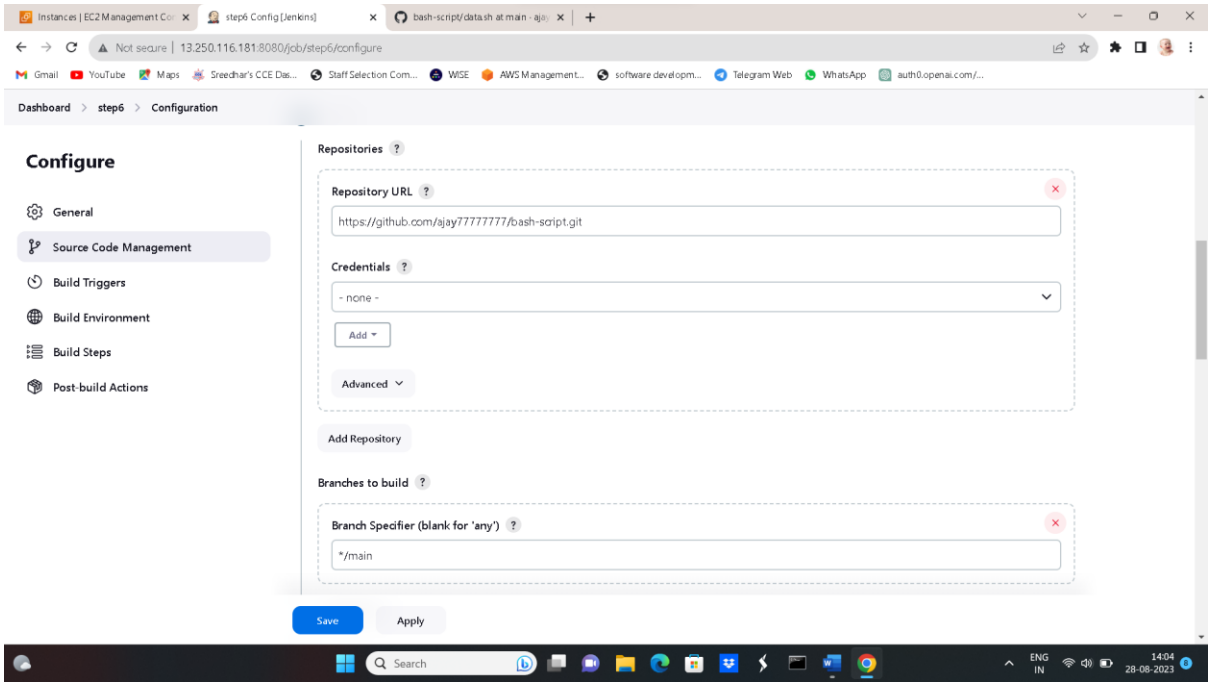
- First login to the AWS account with respective credentials and go to the EC2 services.
- Create a VPC along with subnets, route tables, internet gateway, elastic IP (if required), NACL (optional).
- Now create and launch EC2 instance by selecting Ubuntu or amazonlinux2 versions with respective ports, they are SSH(22), HTTP(80), HTTPS(443), and Custom(8080), (9000).
- Now launch the instance by using the SSH command with Git-bash or putty.
- For Ubuntu linux server use this commands as, This is the Debain package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system.
    - curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
      > /usr/share/keyrings/jenkins-keyring.asc > /dev/null
    - echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
      > https://pkg.jenkins.io/debian binary/ | sudo tee \
      > /etc/apt/sources.list.d/jenkins.list > /dev/null
    - sudo apt-get update
    - sudo apt install openjdk-11-jdk
    - sudo apt-get install Jenkins
- now browse the ip along with port number 8080.



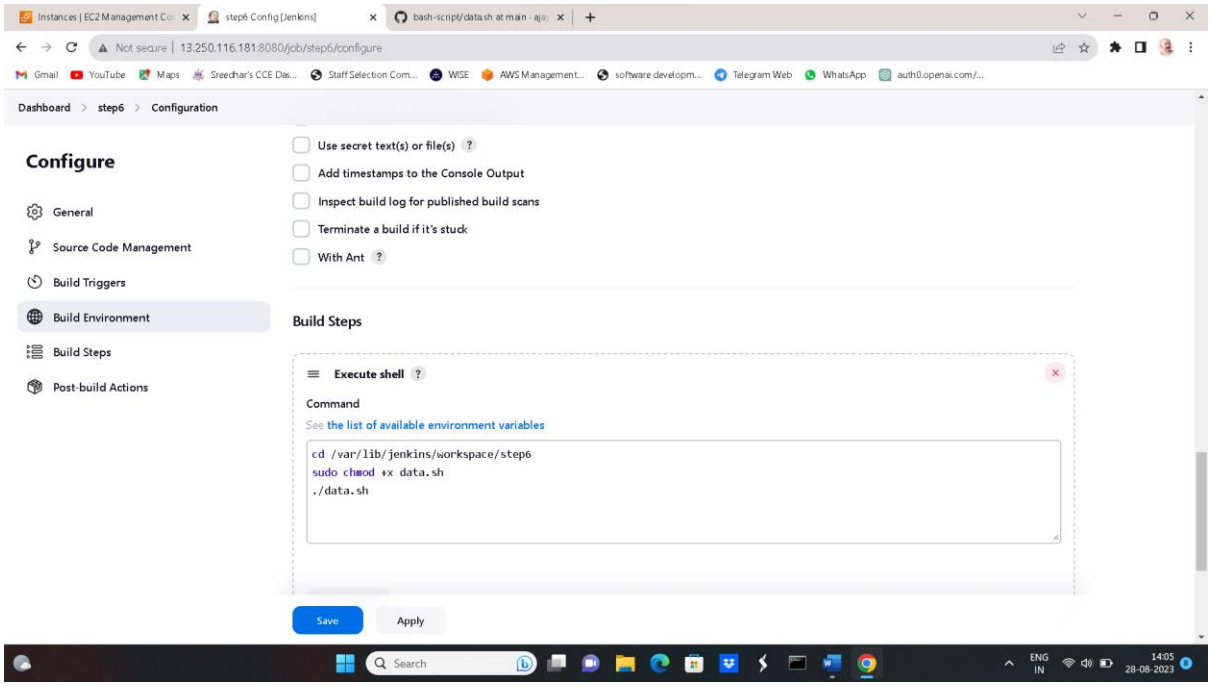
- now create one repository in github



- now you can create one job in Jenkins for cloning the repository



- in the next build step select execute shell and add script to run the shell script.



- after saving the job click on build now .now you can browse the ip along with port number 9000.