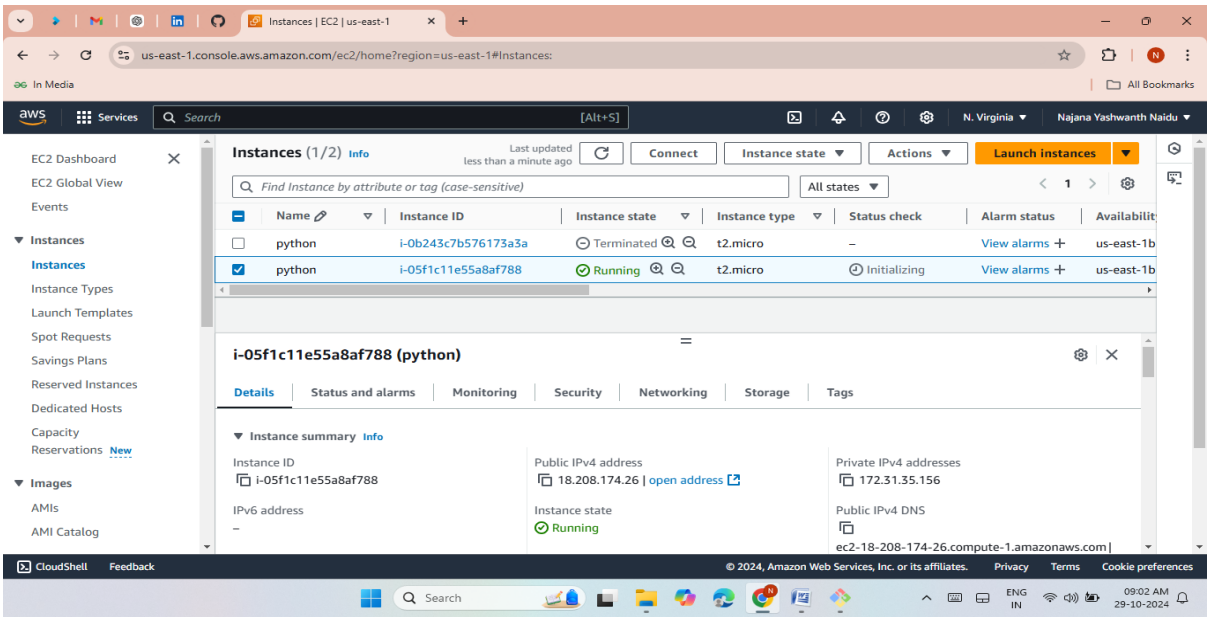
# PROJECT

N. Yashwanth Naidu

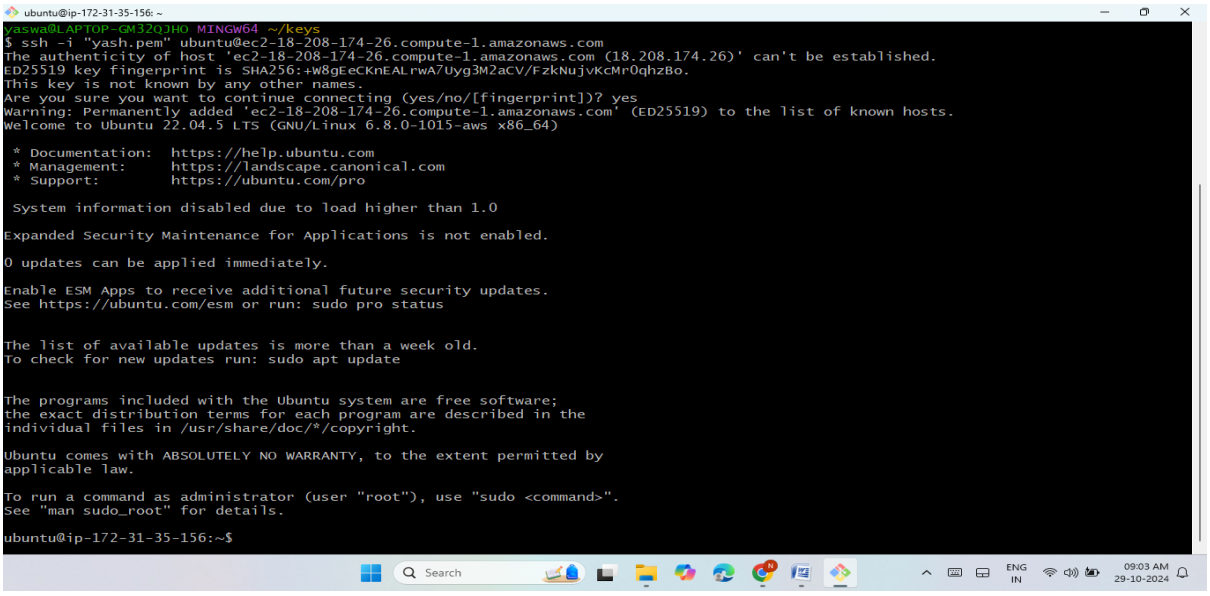
## Method-1

### Deploy flask or multiple python web application manually by using aws resources.

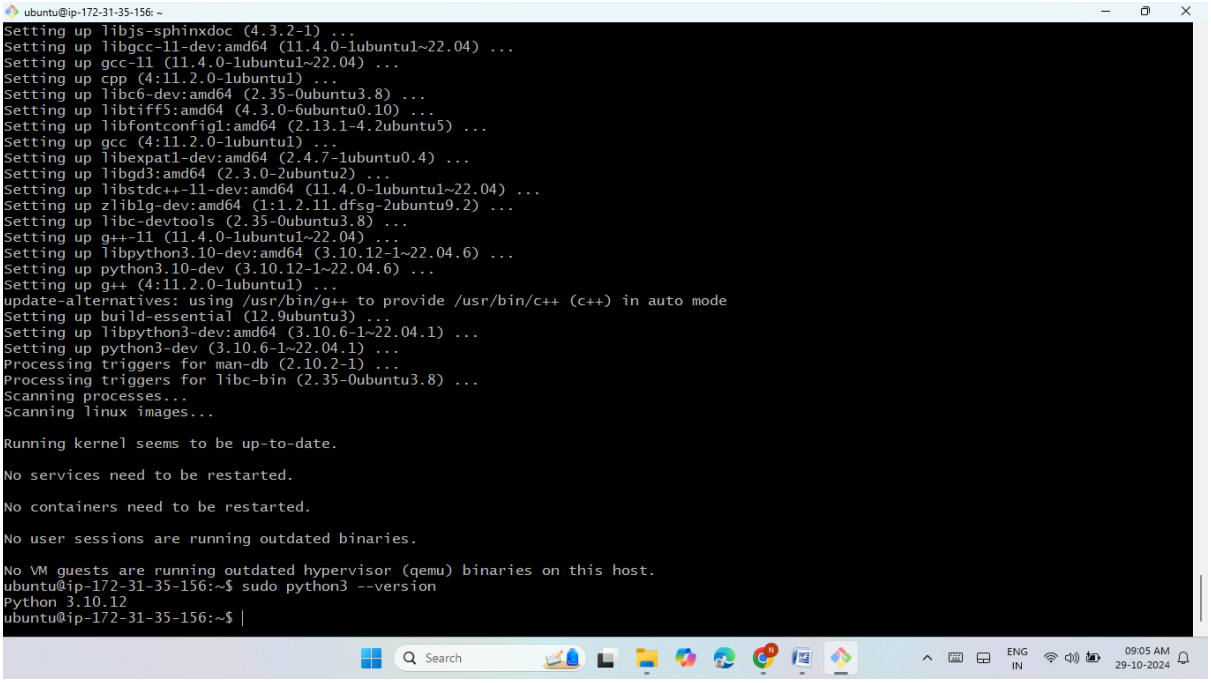
- Deploy flask-library-app web application.
- First launch an ec2 instance with the required ports.



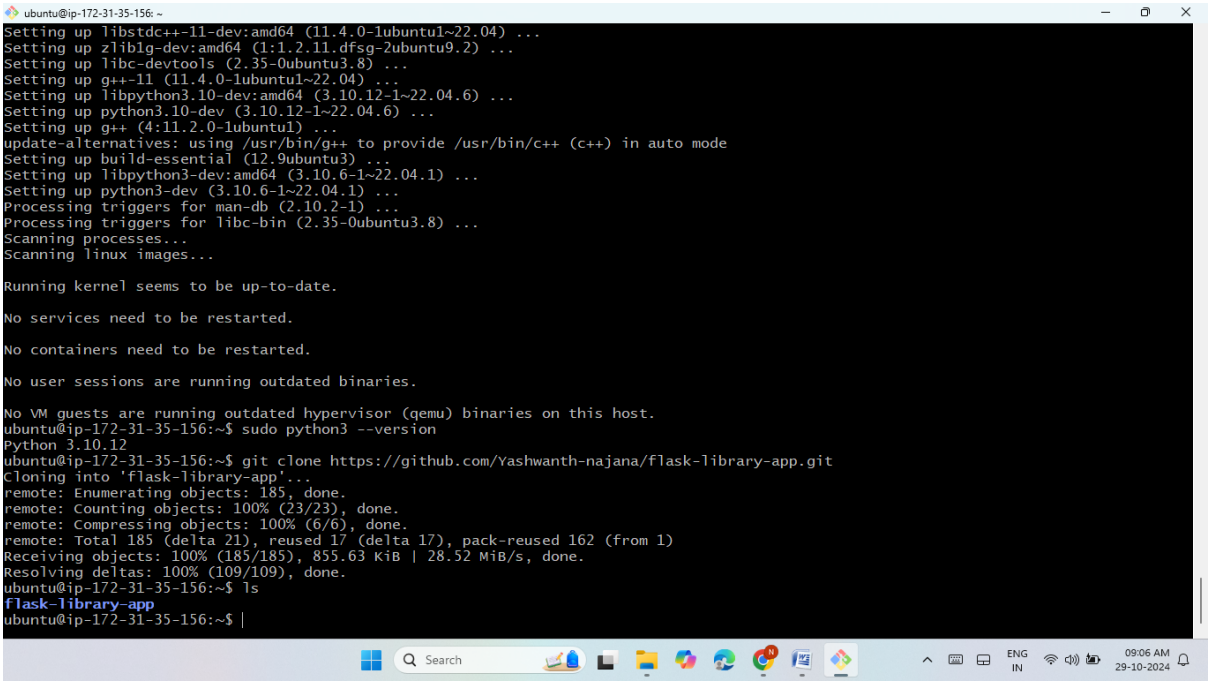

- Now connect the server locally.



- Now install python3 and pip3.

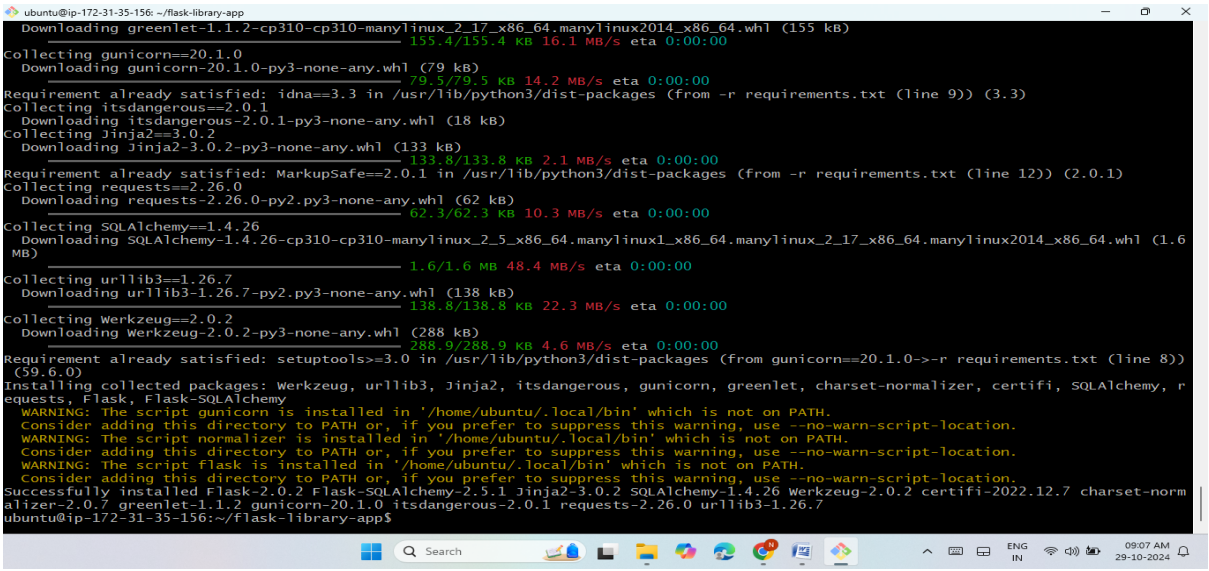



- Now install git then clone the flask-library-app.



- now Install requirements packages with the command.

  **pip3 install –r requirements.txt**




- Here, we want to edit the file app.py with some details.
- Because it will generate localhost IP address.
- We can’t access web app with that IP address.

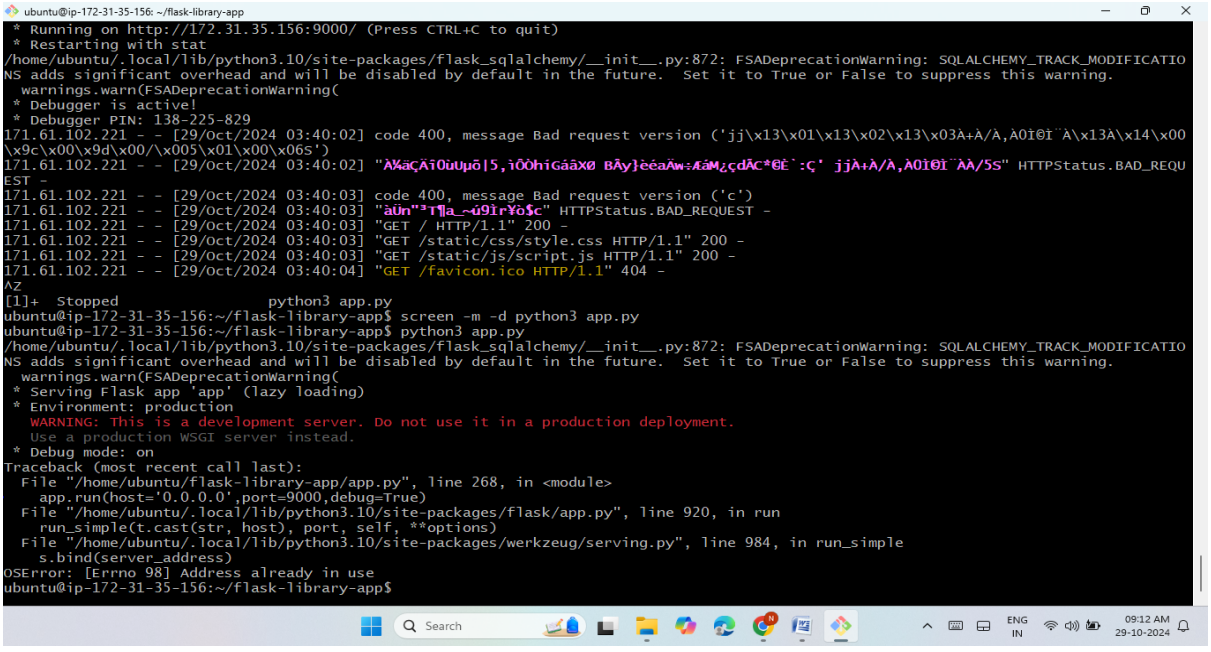
  **Sudo vi app.py**

- Go to very bottom of the file and paste this below txt and save the file.

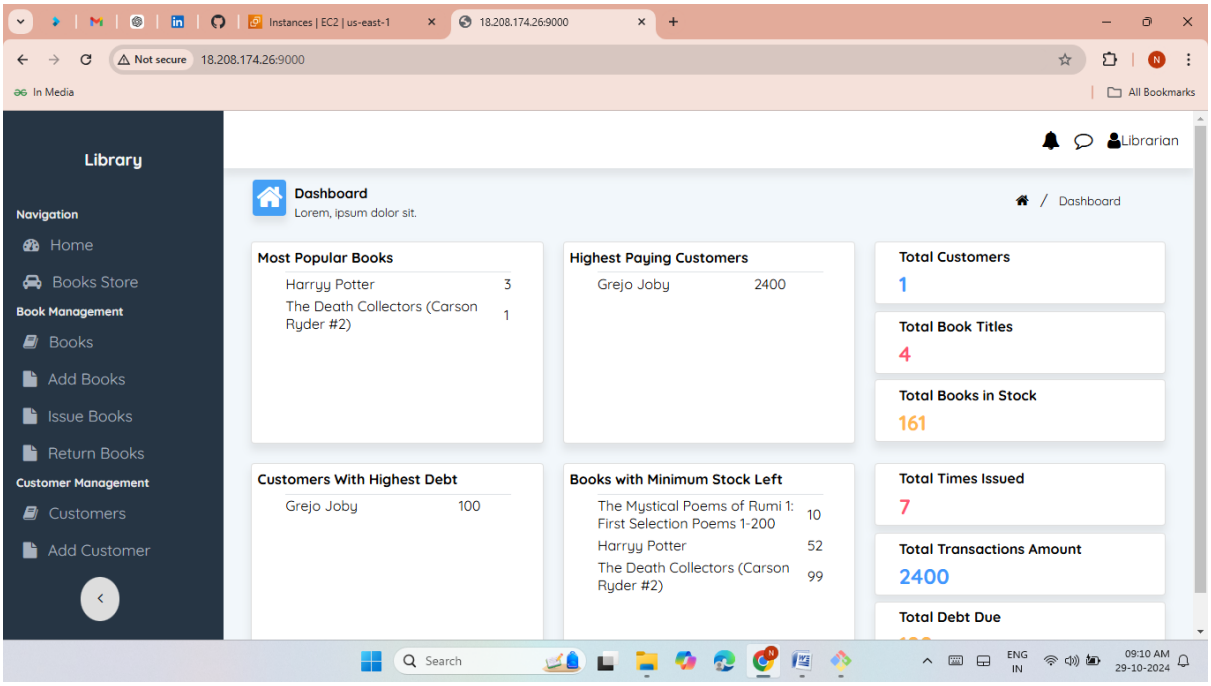
  **app.run(host=’0.0.0.0’, port=8000, debug=True)**

- now run flask-library-app with the command.

  **python3 app.py or screen –m –d python3  app.py**



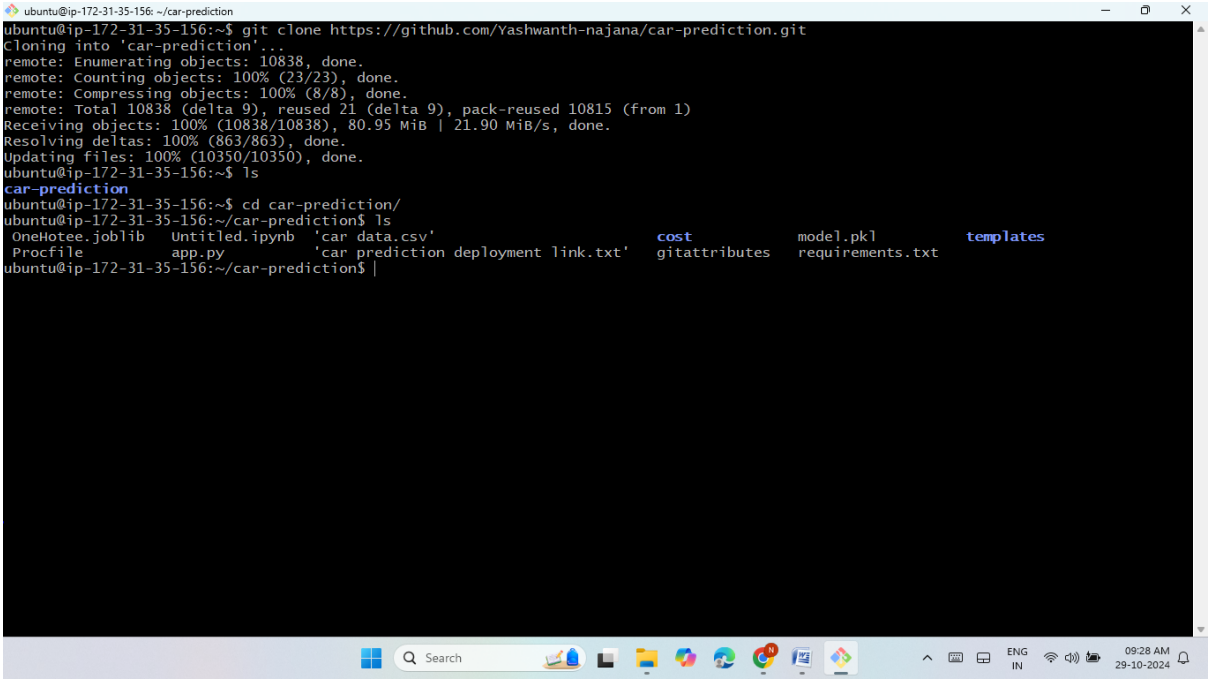
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the flask-library-app is displays.

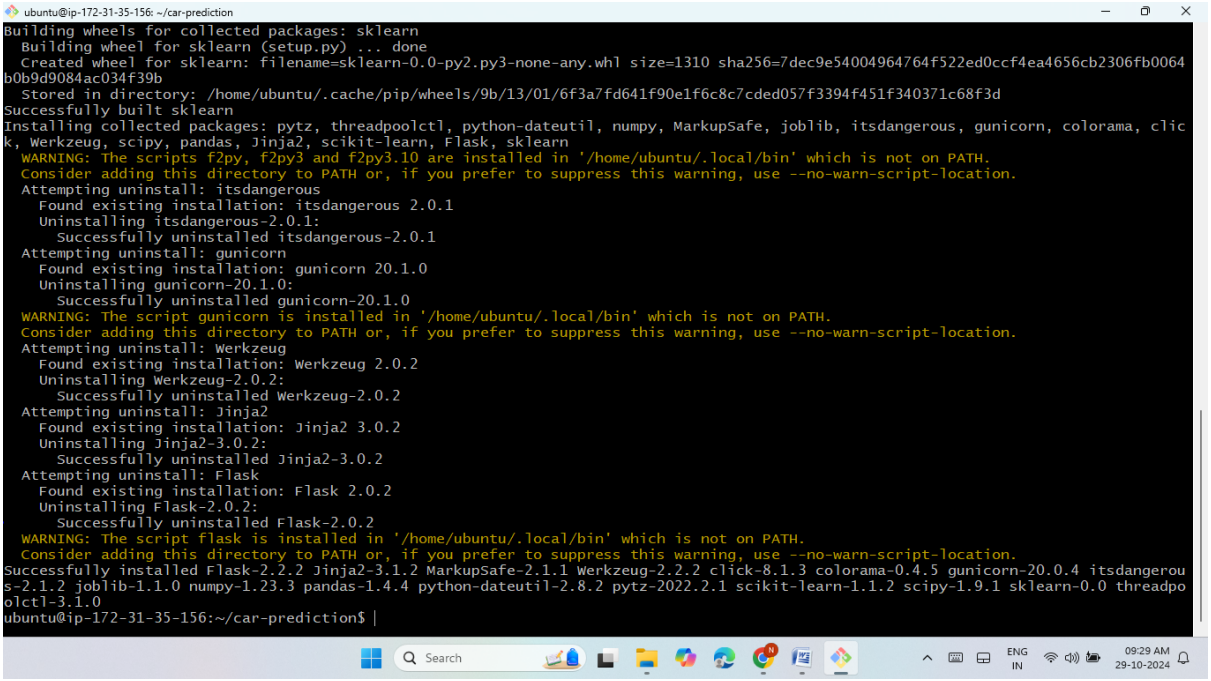


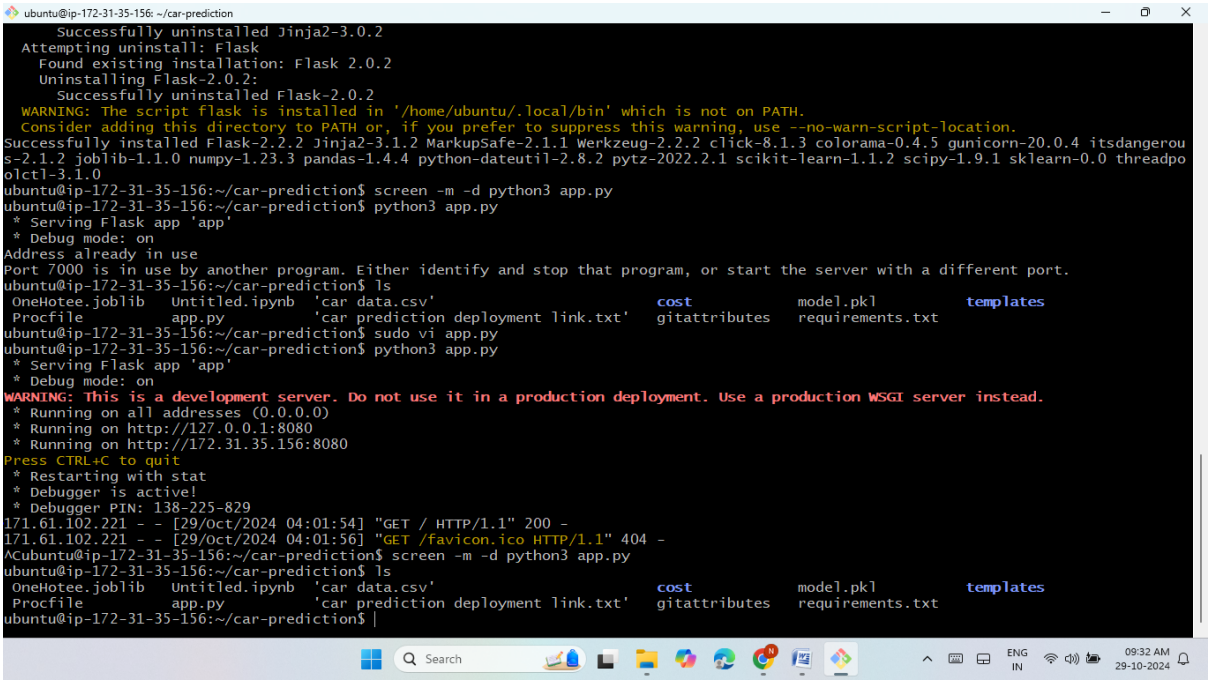

- Now clone the car-prediction python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run car-prediction with the command.

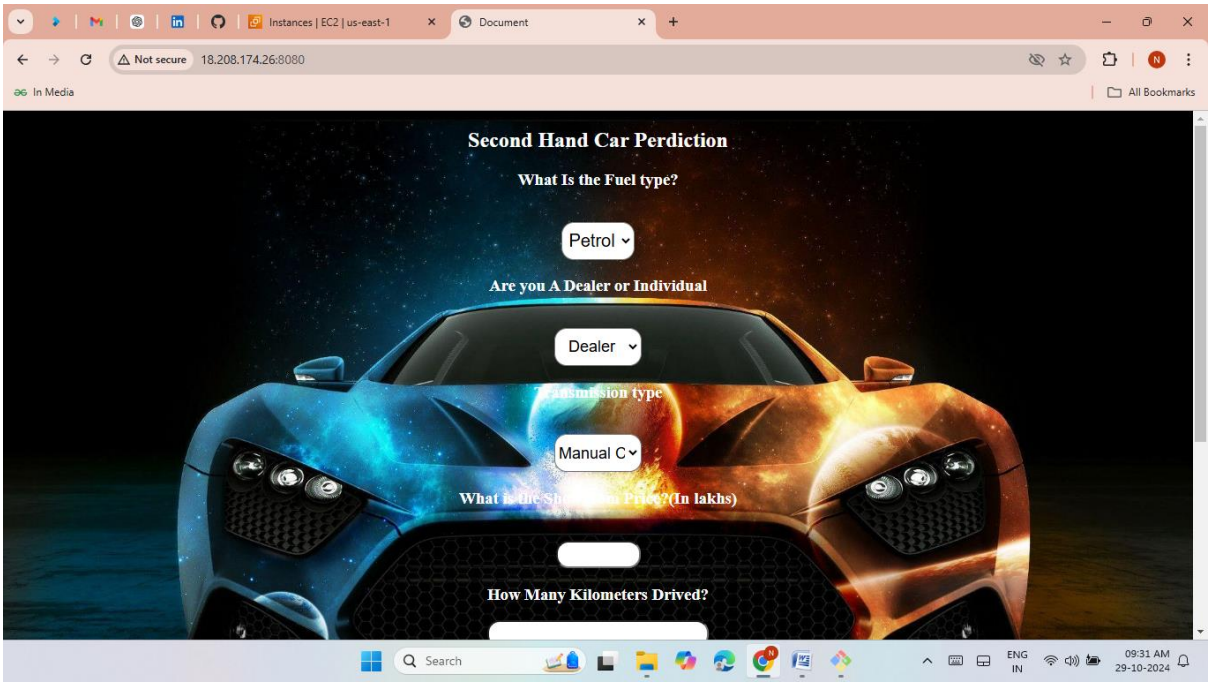
  **python3 app.py or screen –m –d python3  app.py**




ubuntu@ip-172-31-35-156: ~/car-prediction
Building wheels for collected packages: sklearn
Building wheel for sklearn (setup.py) ... done
Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.whl size=1310 sha256=7dec9e54004964764f522ed0ccf4ea4656cb2306fb0064b0b9d9084ac034f39b
Stored in directory: /home/ubuntu/.cache/pip/wheels/9b/13/01/6f3a7fd641f90e1f6c8c7cded057f3394f451f340371c68f3d
Successfully built sklearn
Installing collected packages: pytz, threadpoolctl, python-dateutil, numpy, MarkupSafe, joblib, itsdangerous, gunicorn, colorama, click, Werkzeug, scipy, pandas, Jinja2, scikit-learn, Flask, sklearn
WARNING: The scripts f2py, f2py3 and f2py3.10 are installed in '/home/ubuntu/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Attempting uninstall: itsdangerous
Found existing installation: itsdangerous 2.0.1
Uninstalling itsdangerous-2.0.1:
Successfully uninstalled itsdangerous-2.0.1
Attempting uninstall: gunicorn
Found existing installation: gunicorn 20.1.0
Uninstalling gunicorn-20.1.0:
Successfully uninstalled gunicorn-20.1.0
WARNING: The script gunicorn is installed in '/home/ubuntu/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Attempting uninstall: Werkzeug
Found existing installation: Werkzeug 2.0.2
Uninstalling Werkzeug-2.0.2:
Successfully uninstalled Werkzeug-2.0.2
Attempting uninstall: Jinja2
Found existing installation: Jinja2 3.0.2
Uninstalling Jinja2-3.0.2:
Successfully uninstalled Jinja2-3.0.2
Attempting uninstall: Flask
Found existing installation: Flask 2.0.2
Uninstalling Flask-2.0.2:
Successfully uninstalled Flask-2.0.2
WARNING: The script flask is installed in '/home/ubuntu/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Flask-2.2.2 Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 colorama-0.4.5 gunicorn-20.0.4 itsdangerous-2.1.2 joblib-1.1.0 numpy-1.23.3 pandas-1.4.4 python-dateutil-2.8.2 pytz-2022.2.1 scikit-learn-1.1.2 scipy-1.9.1 sklearn-0.0 threadpoolctl-3.1.0
ubuntu@ip-172-31-35-156:~/car-prediction$
Search
ENG
IN
09:29 AM
29-10-2024


ubuntu@ip-172-31-35-156: ~/car-prediction
Successfully uninstalled Jinja2-3.0.2
Attempting uninstall: Flask
Found existing installation: Flask 2.0.2
Uninstalling Flask-2.0.2:
Successfully uninstalled Flask-2.0.2
WARNING: The script flask is installed in '/home/ubuntu/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Flask-2.2.2 Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 colorama-0.4.5 gunicorn-20.0.4 itsdangerous-2.1.2 joblib-1.1.0 numpy-1.23.3 pandas-1.4.4 python-dateutil-2.8.2 pytz-2022.2.1 scikit-learn-1.1.2 scipy-1.9.1 sklearn-0.0 threadpoolctl-3.1.0
ubuntu@ip-172-31-35-156:~/car-prediction$ screen -m -d python3 app.py
ubuntu@ip-172-31-35-156:~/car-prediction$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
Address already in use
Port 7000 is in use by another program. Either identify and stop that program, or start the server with a different port.
ubuntu@ip-172-31-35-156:~/car-prediction$ ls
OneHotee.joblib Untitled.ipynb 'car data.csv' cost model.pkl templates
Procfile app.py 'car prediction deployment link.txt' gitattributes requirements.txt
ubuntu@ip-172-31-35-156:~/car-prediction$ sudo vi app.py
ubuntu@ip-172-31-35-156:~/car-prediction$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8080
* Running on http://172.31.35.156:8080
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 138-225-829
171.61.102.221 - - [29/Oct/2024 04:01:54] "GET / HTTP/1.1" 200 -
171.61.102.221 - - [29/Oct/2024 04:01:56] "GET /favicon.ico HTTP/1.1" 404 -
^Cubuntu@ip-172-31-35-156:~/car-prediction$ screen -m -d python3 app.py
ubuntu@ip-172-31-35-156:~/car-prediction$ ls
OneHotee.joblib Untitled.ipynb 'car data.csv' cost model.pkl templates
Procfile app.py 'car prediction deployment link.txt' gitattributes requirements.txt
ubuntu@ip-172-31-35-156:~/car-prediction$
Search
ENG
IN
09:32 AM
29-10-2024

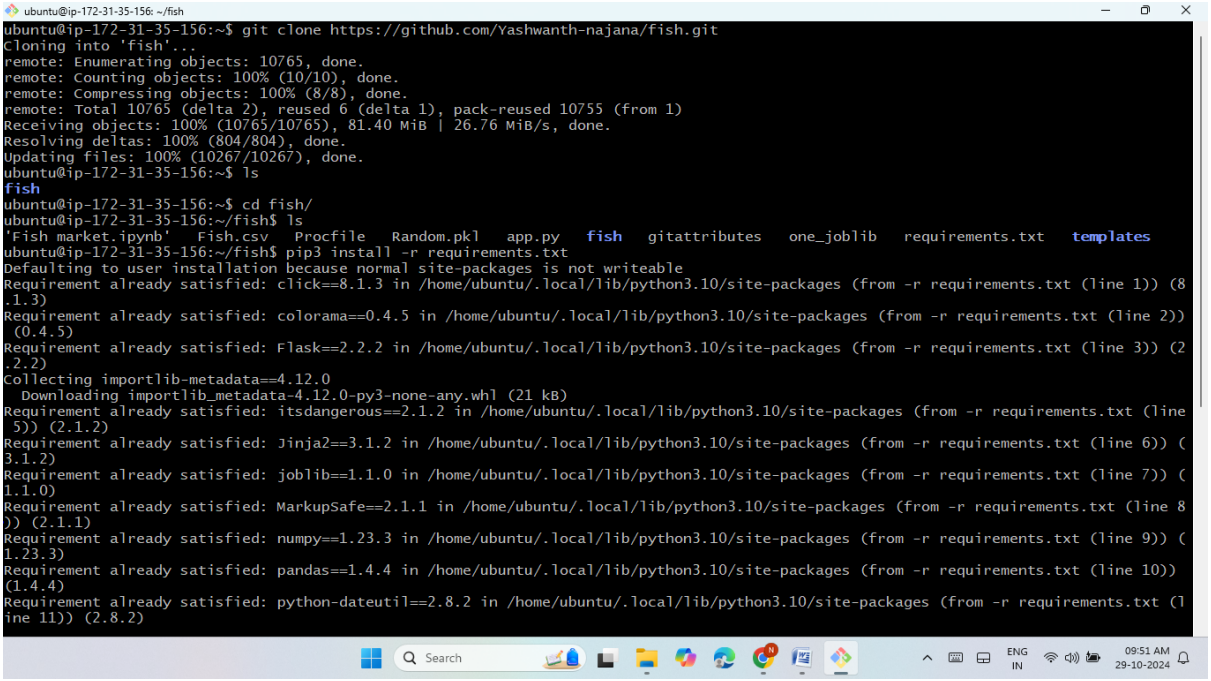
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the car-prediction is displays.

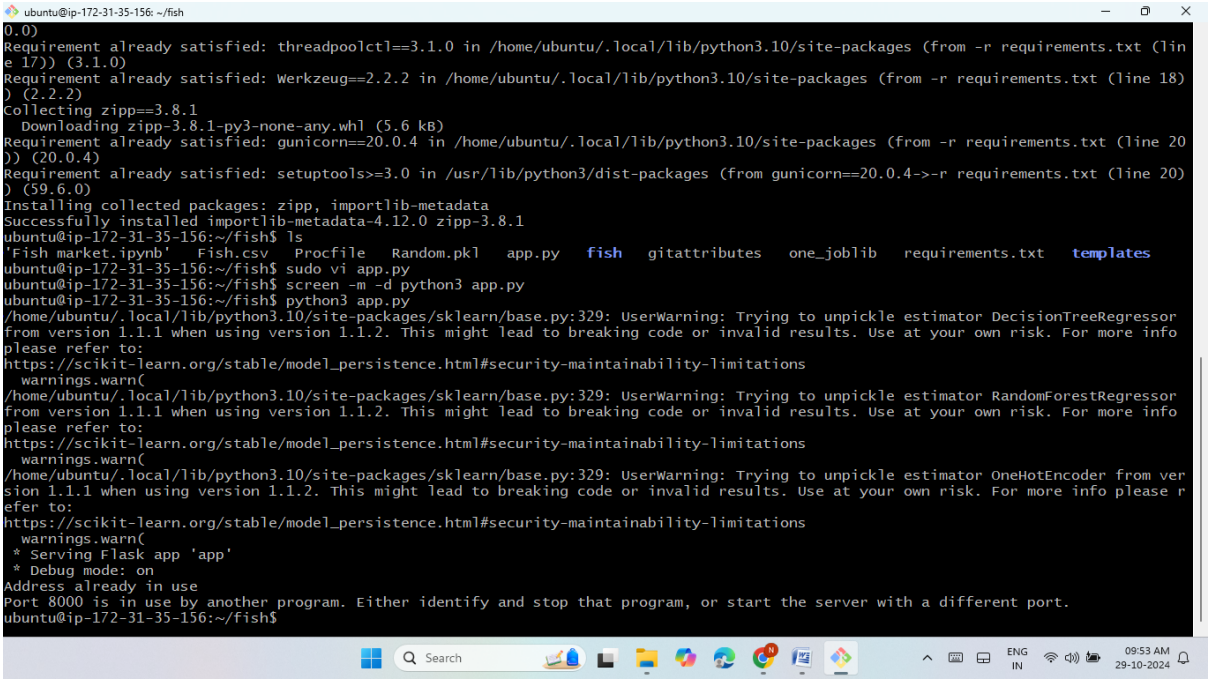



- Now clone the fish python web applications.
- Now install requirements packages with the command.

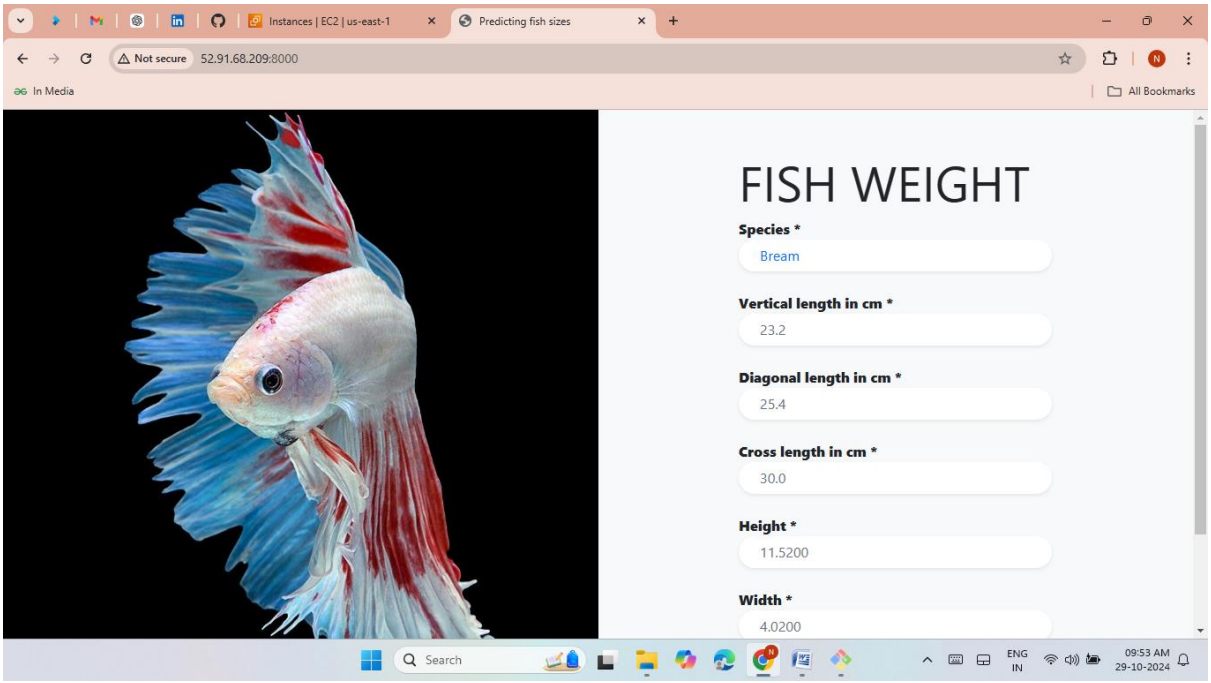
  **pip3 install –r requirements.txt**
- Now run fish with the command.

  **python3 app.py or screen –m –d python3  app.py**





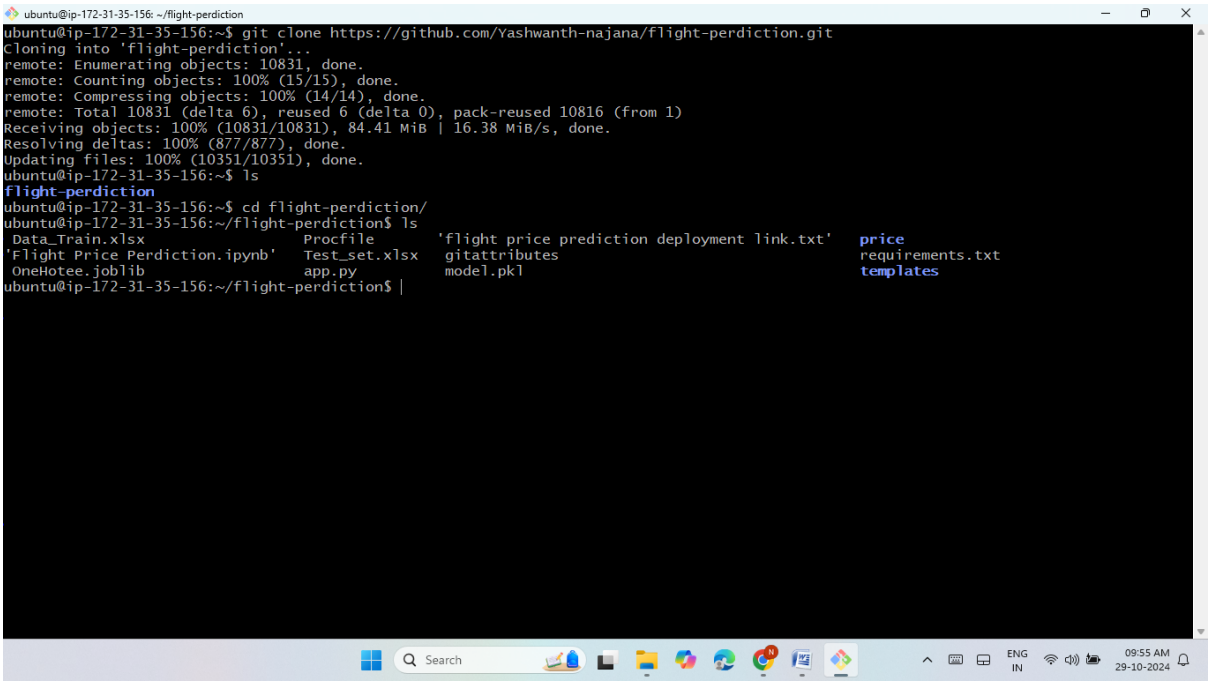

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish is displays.

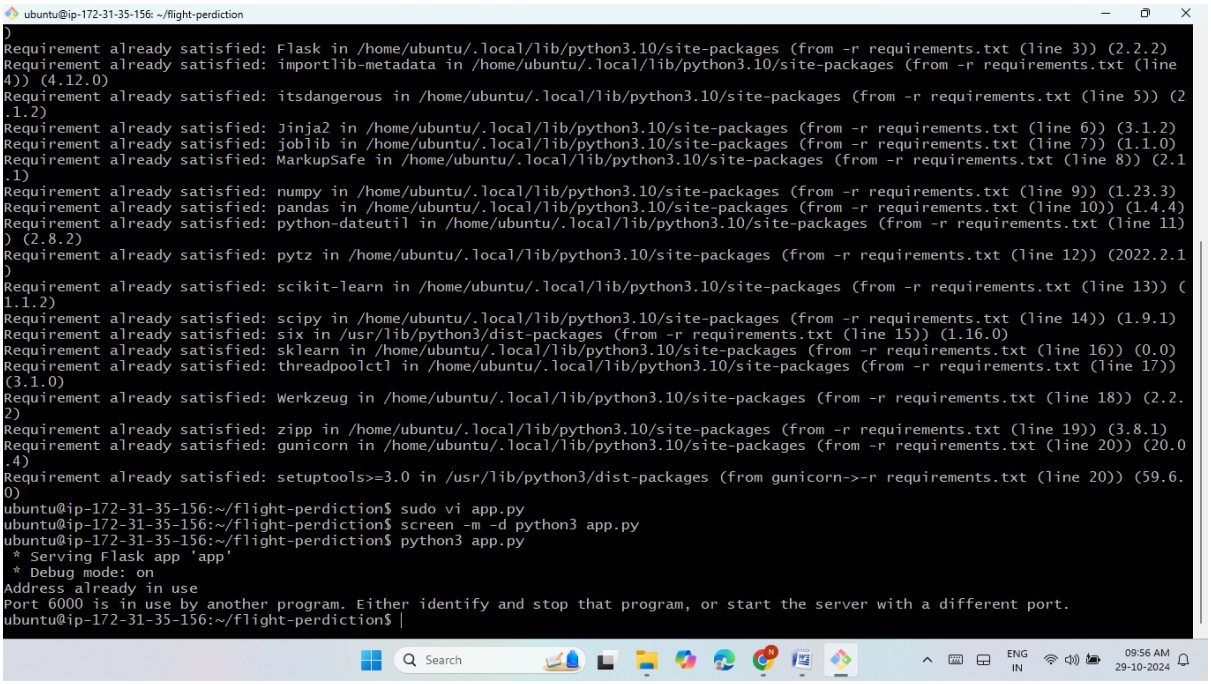


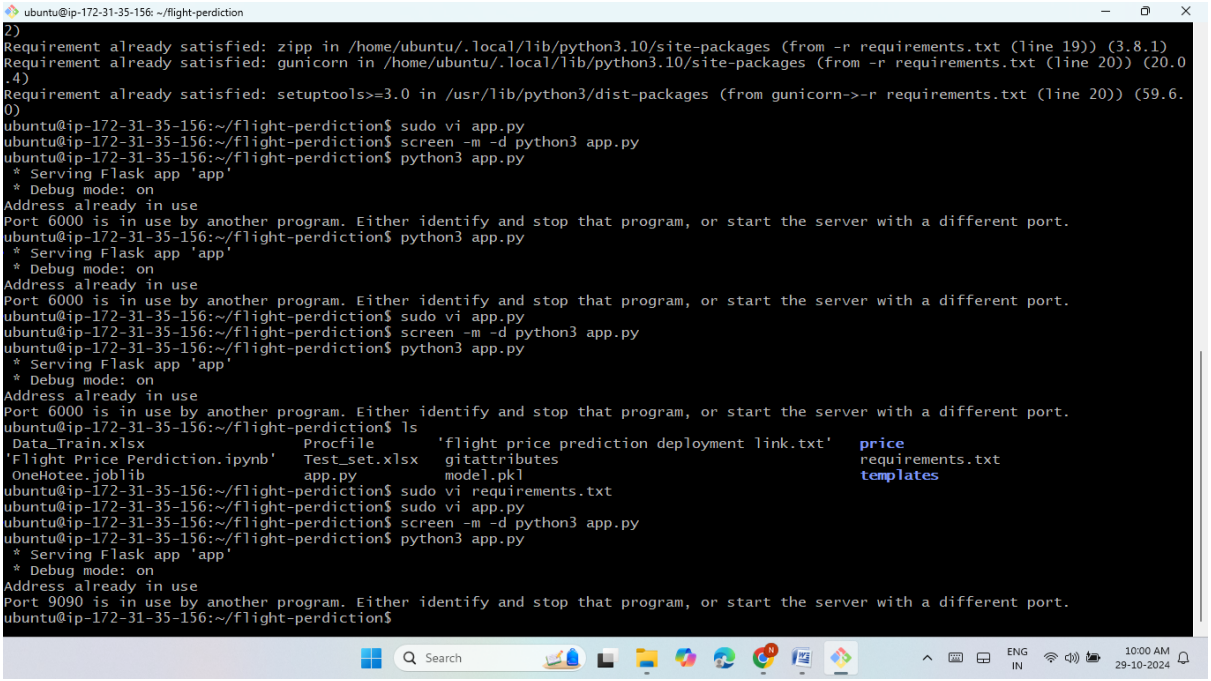
- Now clone the flight-perdiction python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run flight-perdiction with the command.

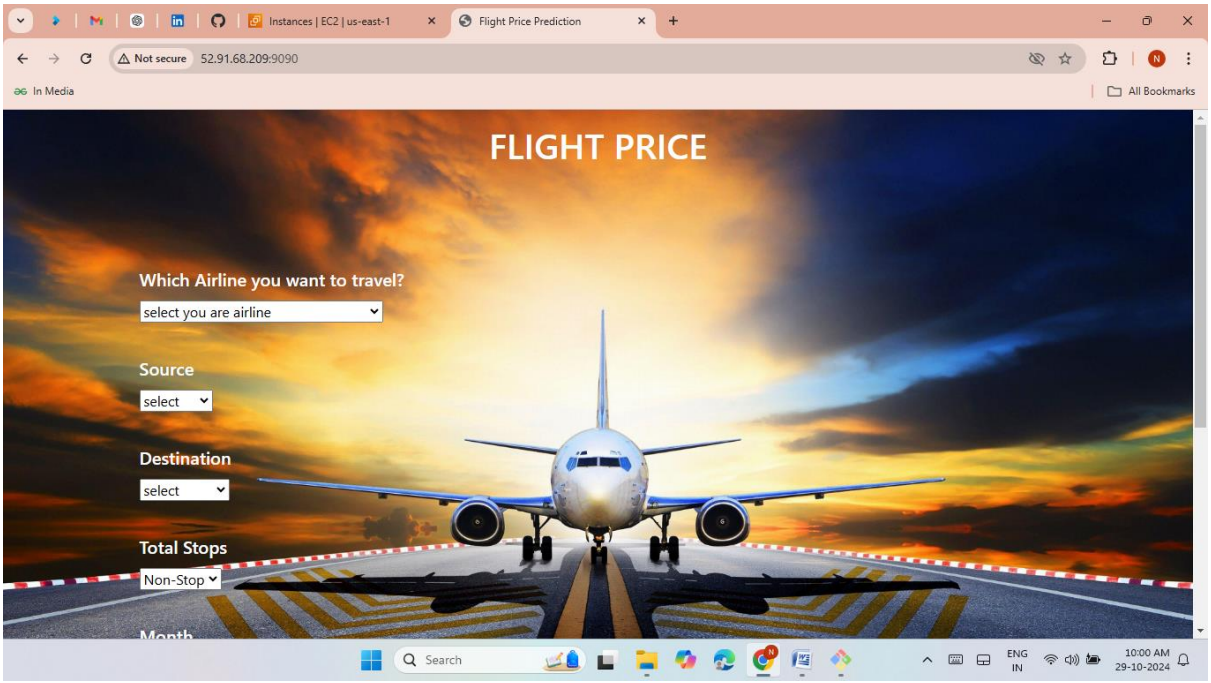
  **python3 app.py or screen –m –d python3  app.py**





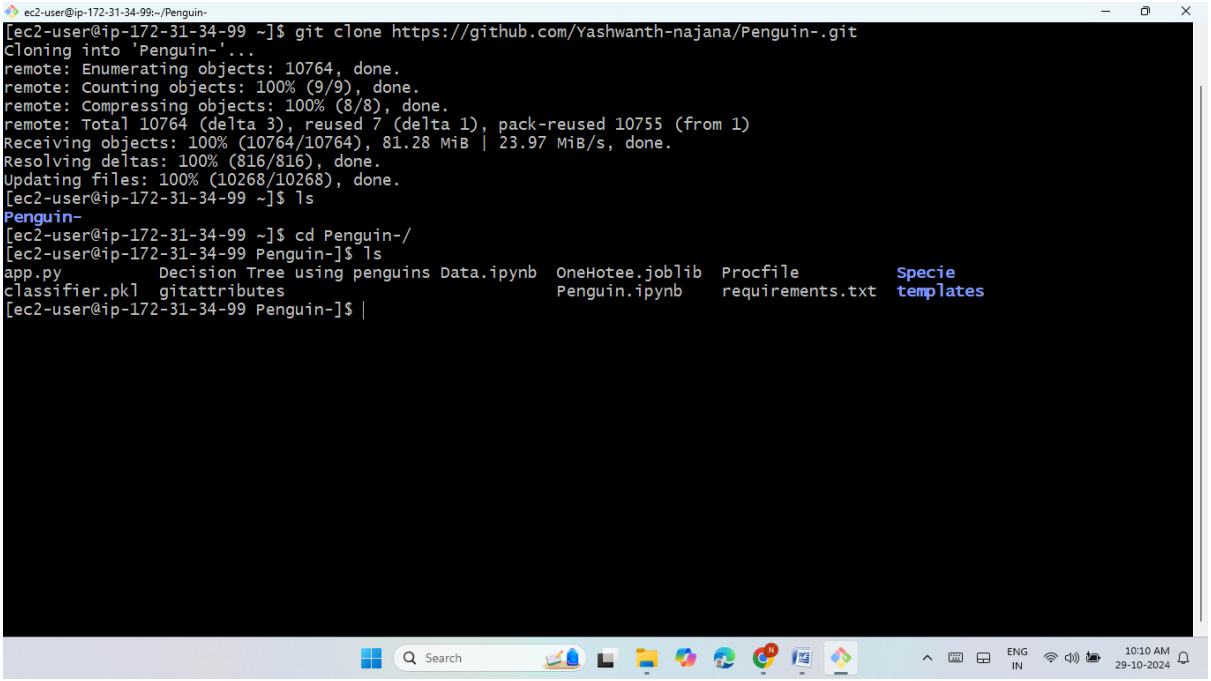

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the flight-perdiction is displays.

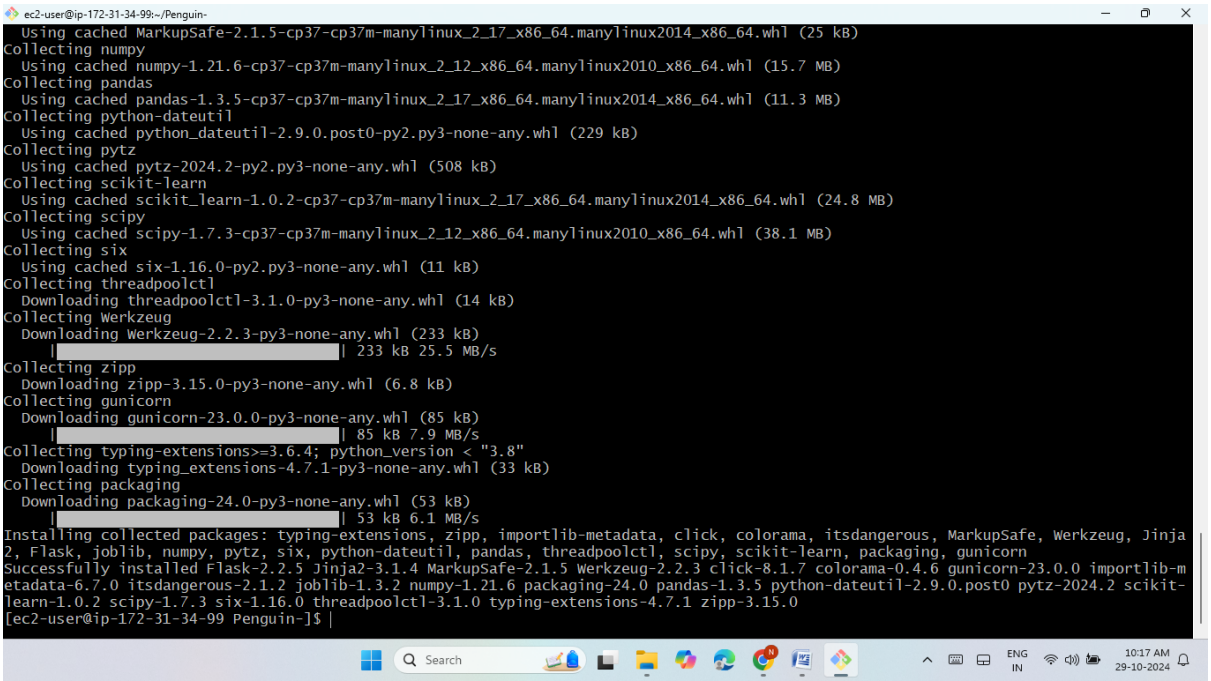


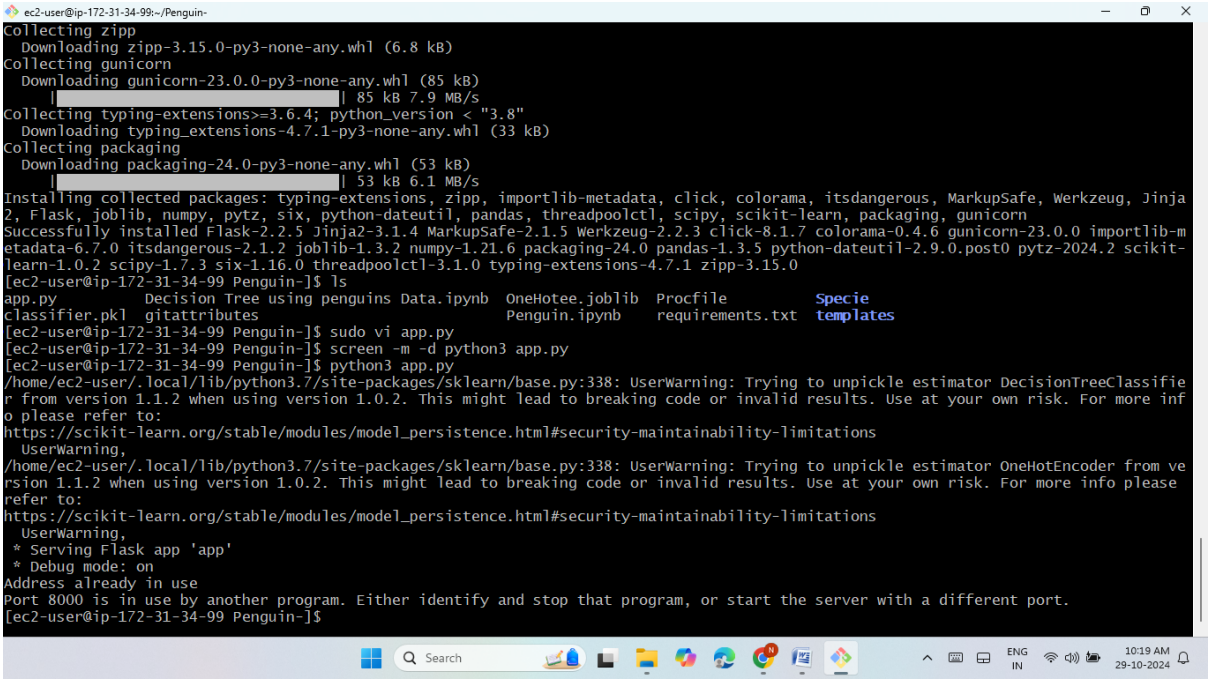
- Now clone the Penguin- python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run Penguin-  with the command.

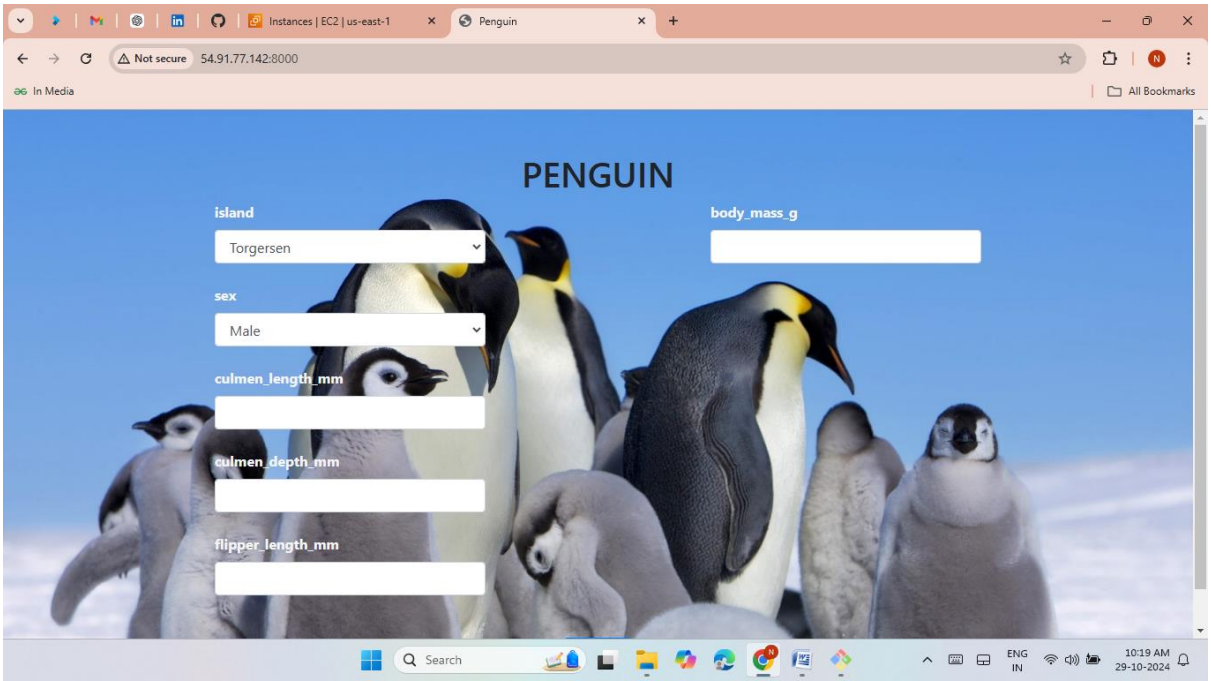
  **python3 app.py or screen –m –d python3  app.py**





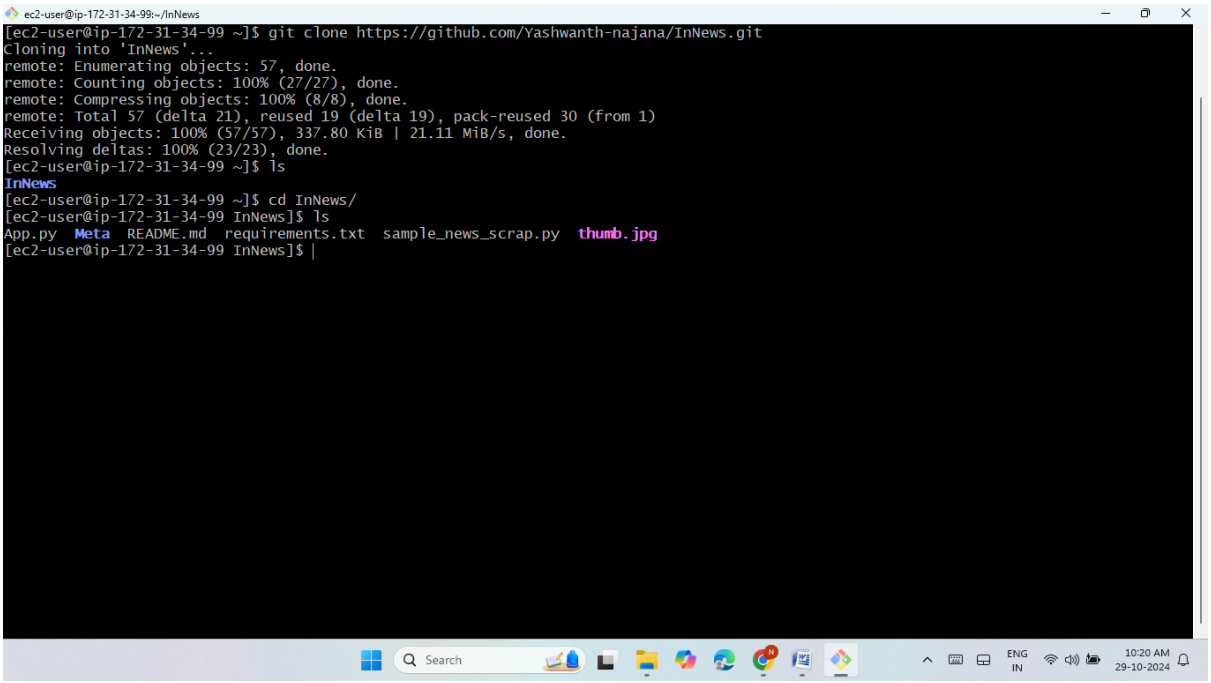

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the Penguin- is displays.

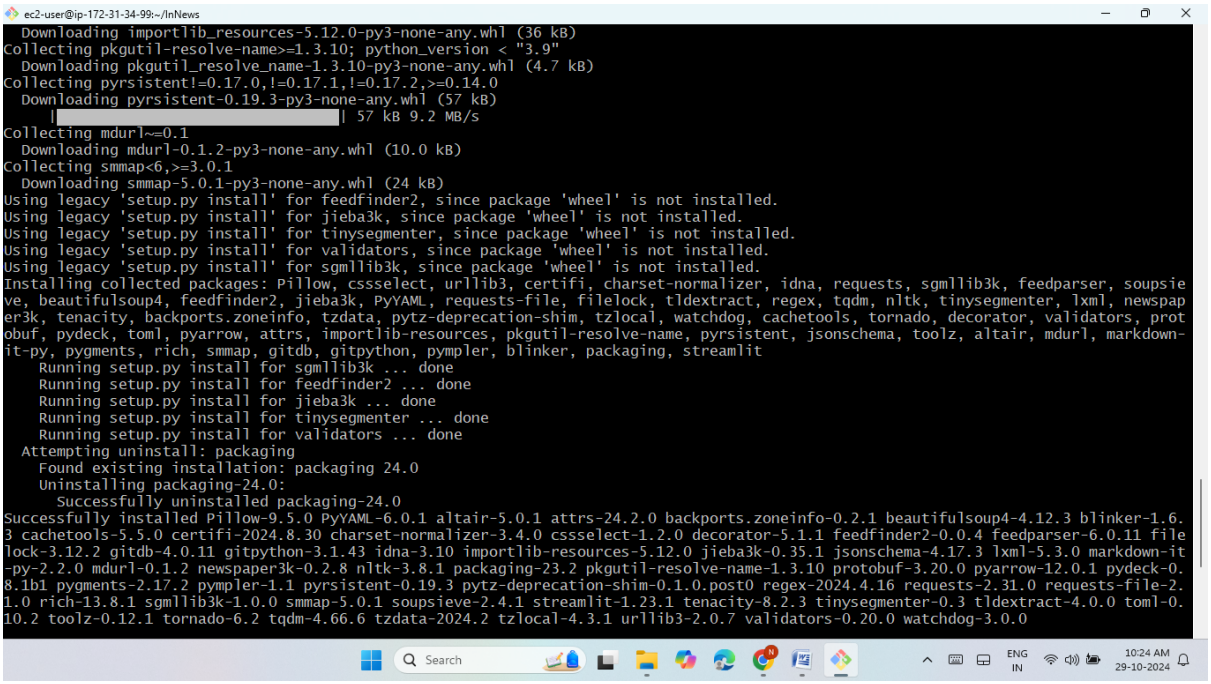


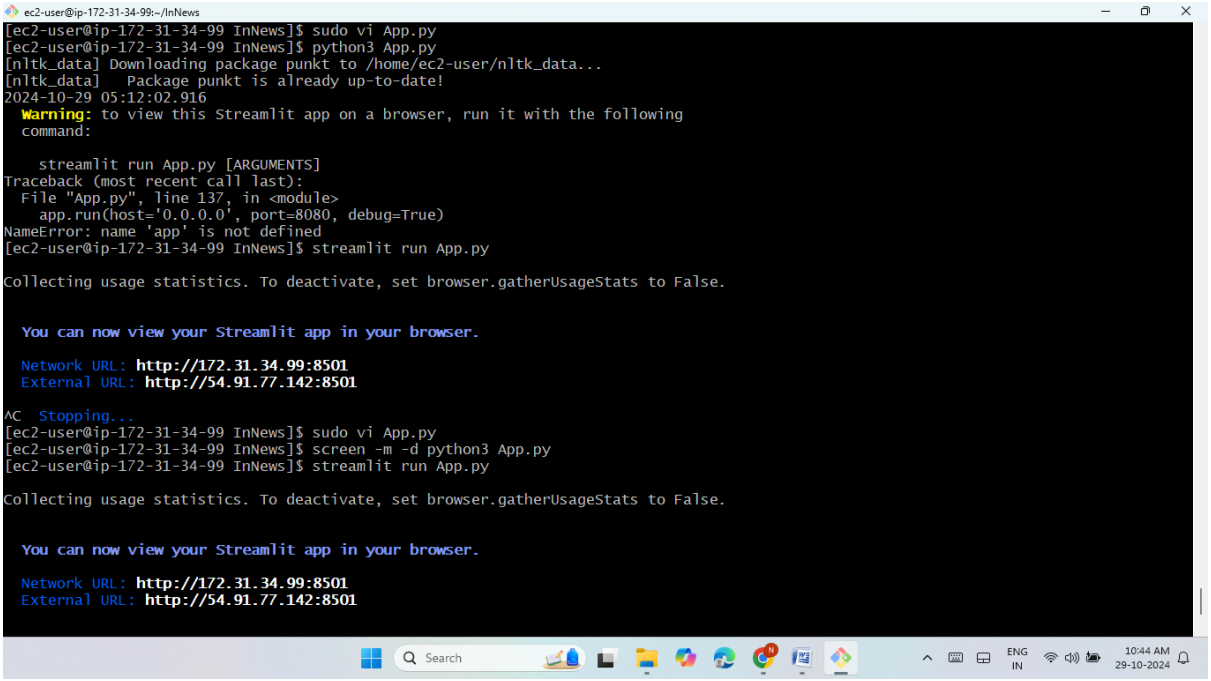
- Now clone the InNews python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run InNews with the command.

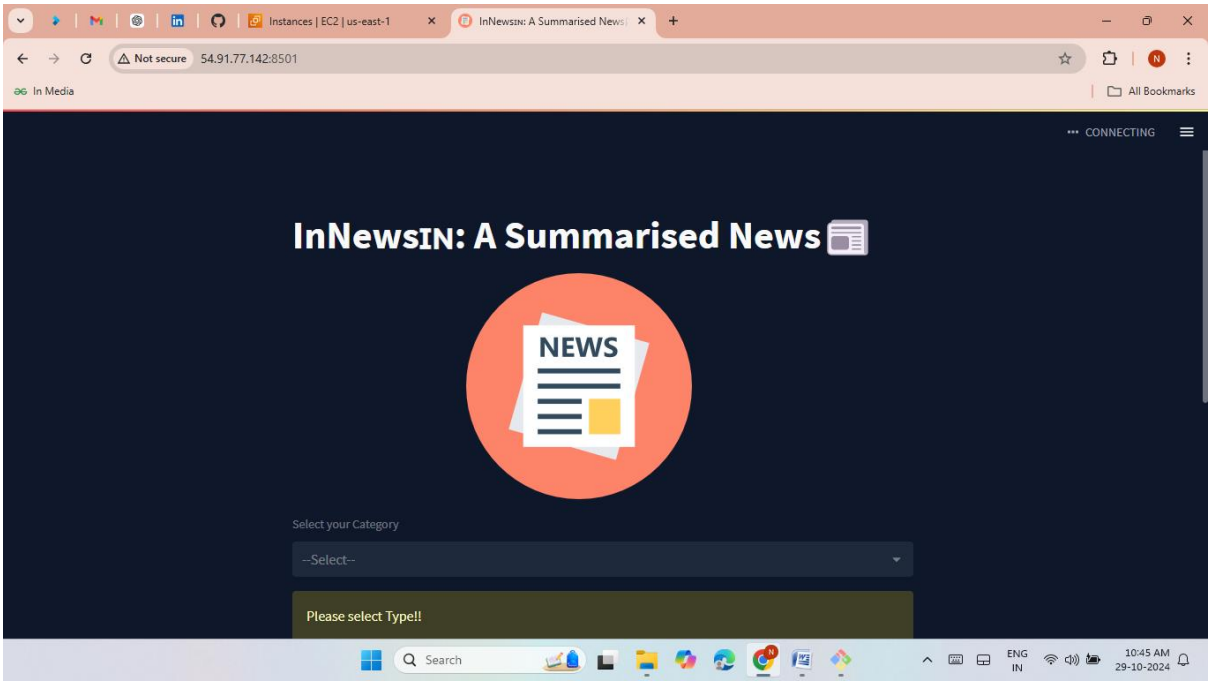
  **python3 app.py or screen –m –d python3  app.py**





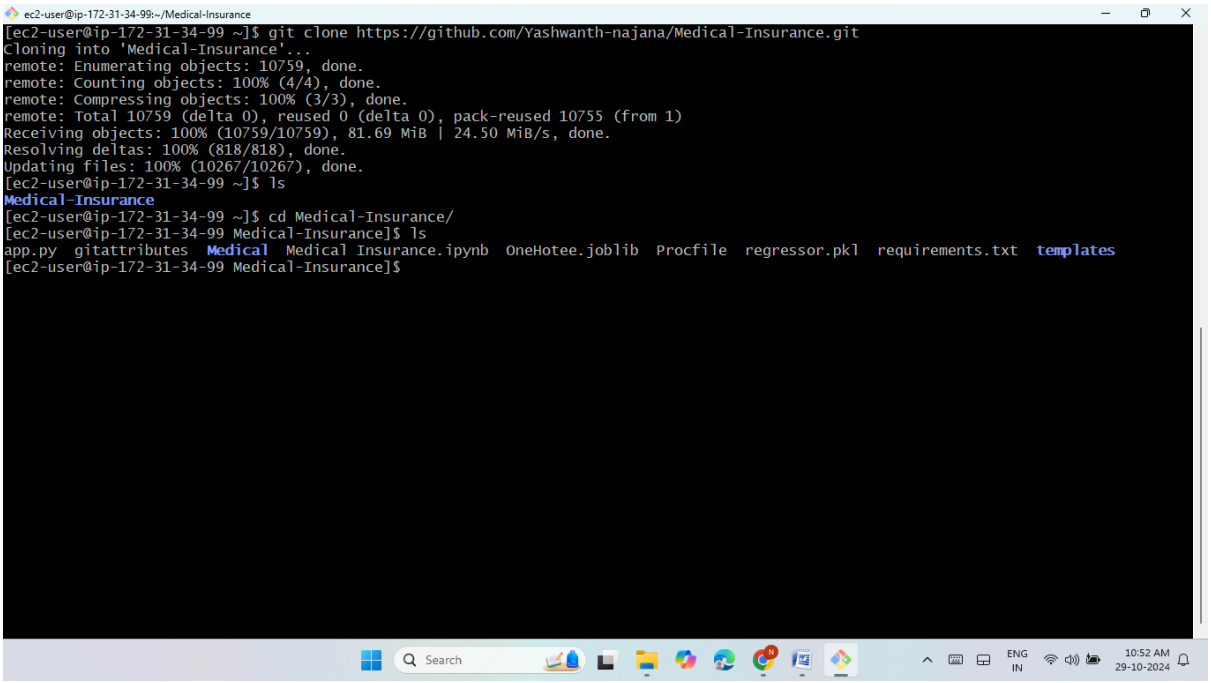

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the InNews is displays.

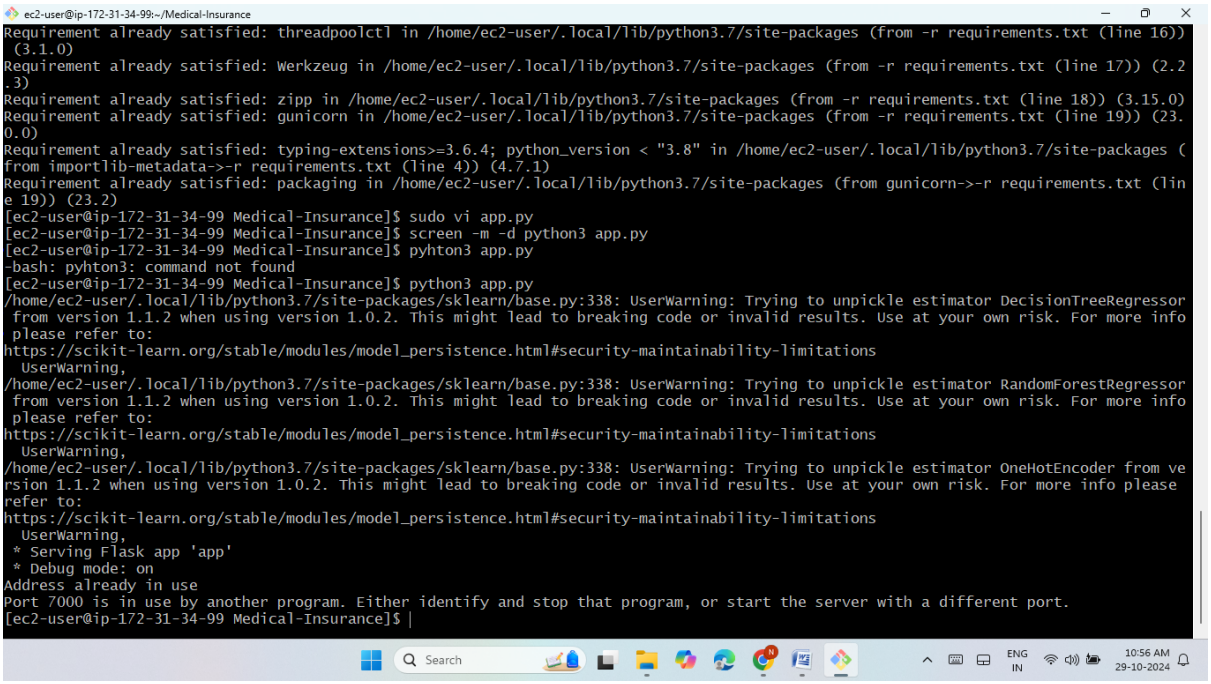


- Now clone the Medical-Insurance python web applications.
- Now install requirements packages with the command.

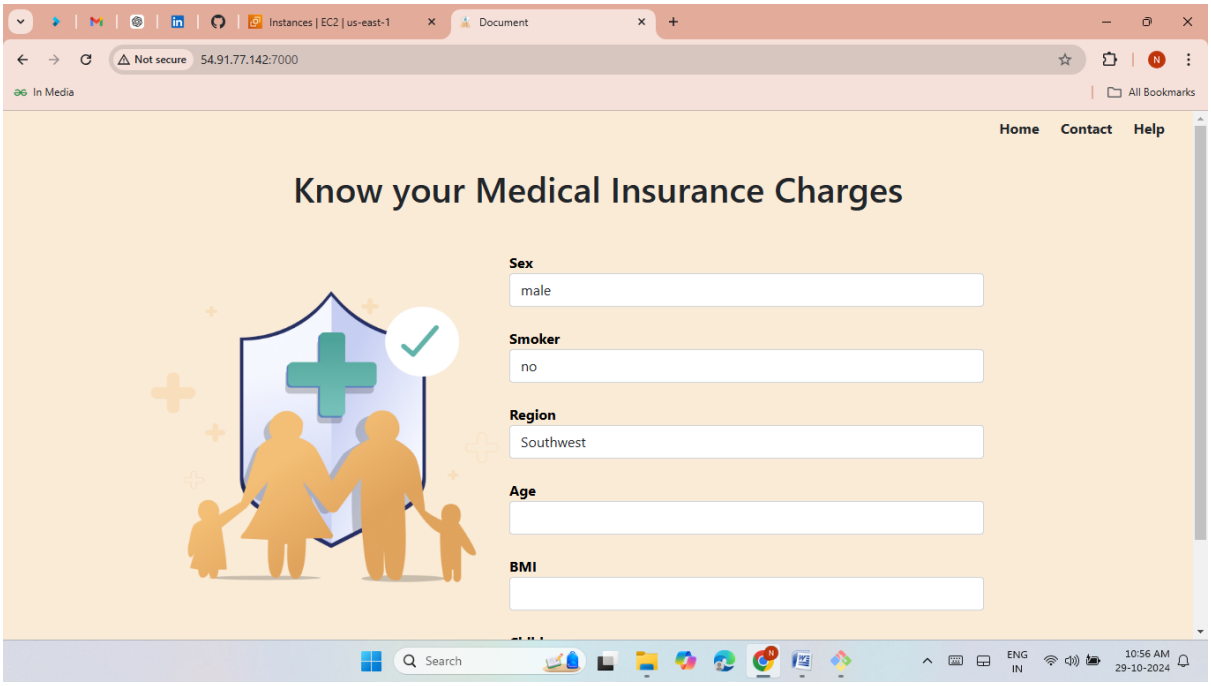
  **pip3 install –r requirements.txt**
- Now run Medical-Insurance with the command.

  **python3 app.py or screen –m –d python3  app.py**



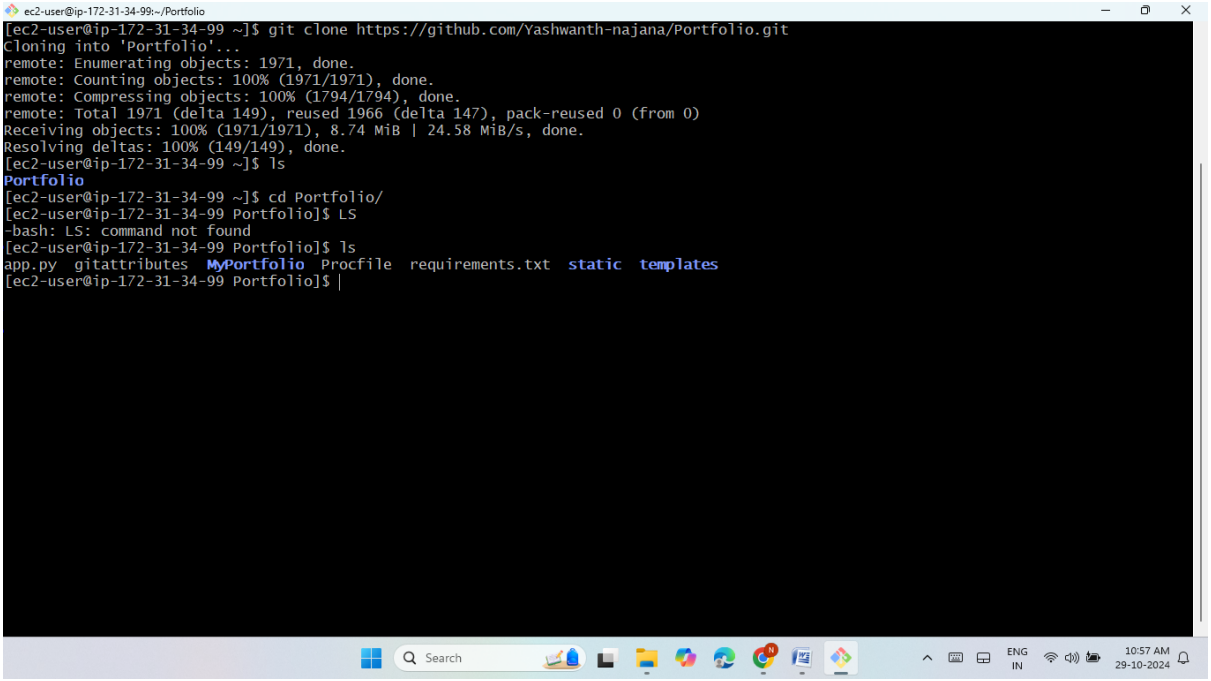
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the Medical-Insurance is displays.



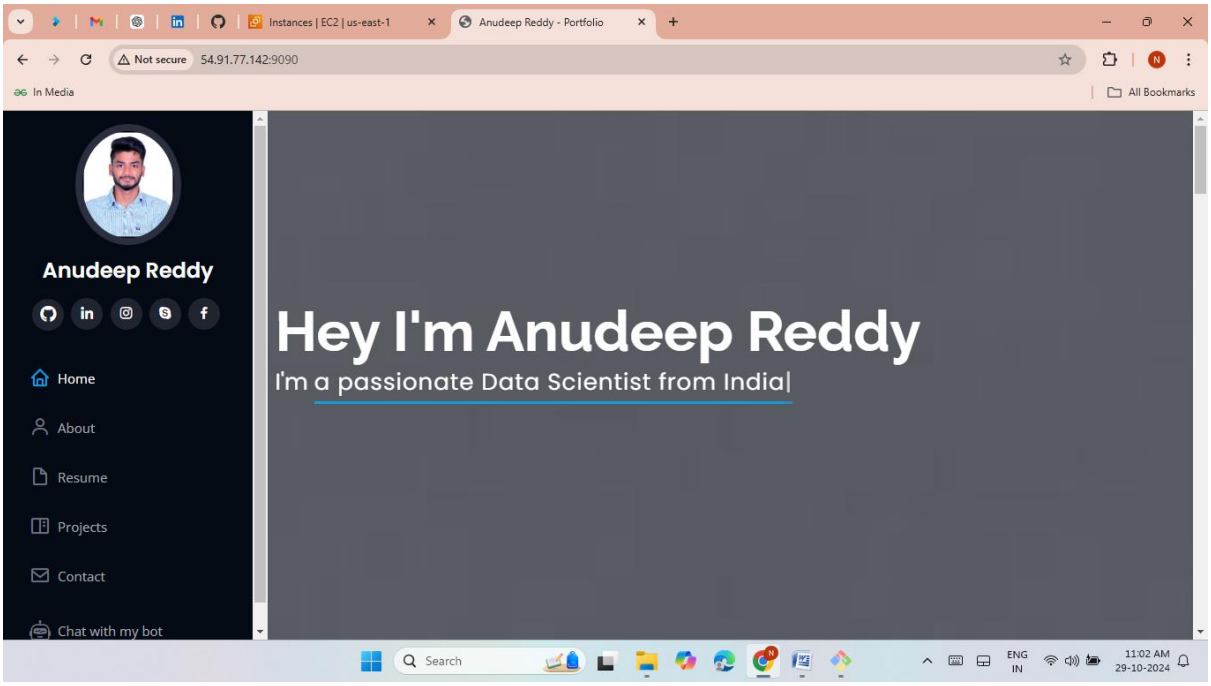

- Now clone the Portfolio python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run Portfolio with the command.

  **python3 app.py or screen –m –d python3  app.py**



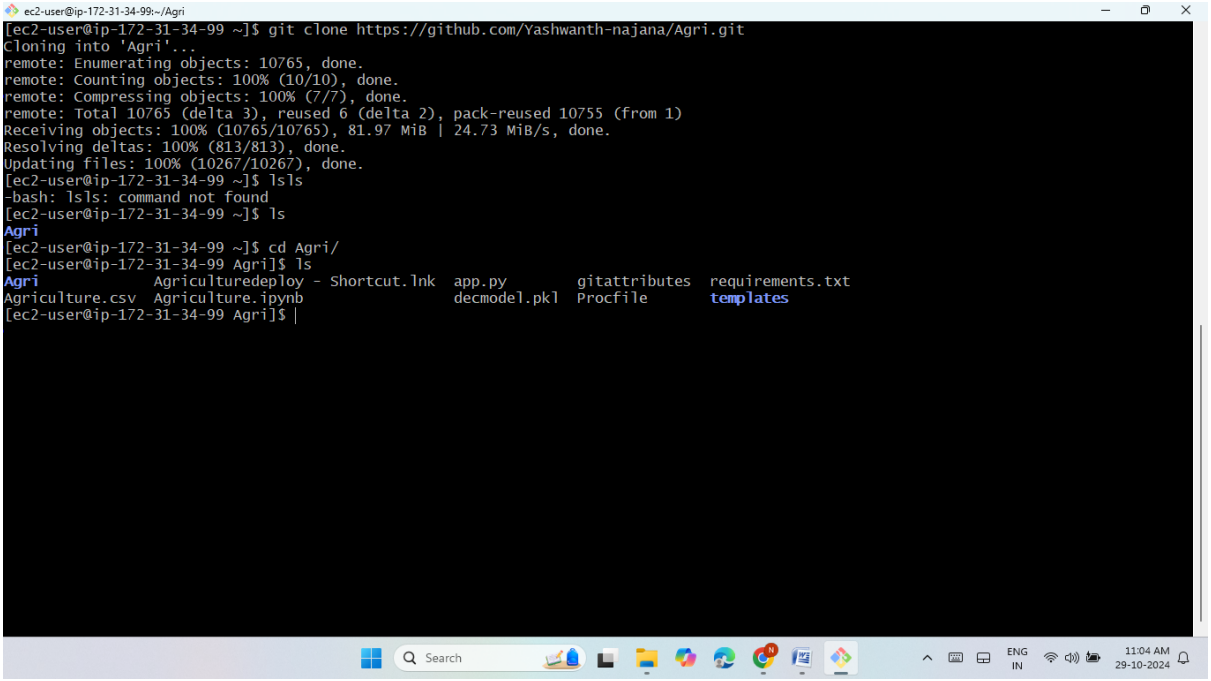
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the Portfolio is displays.



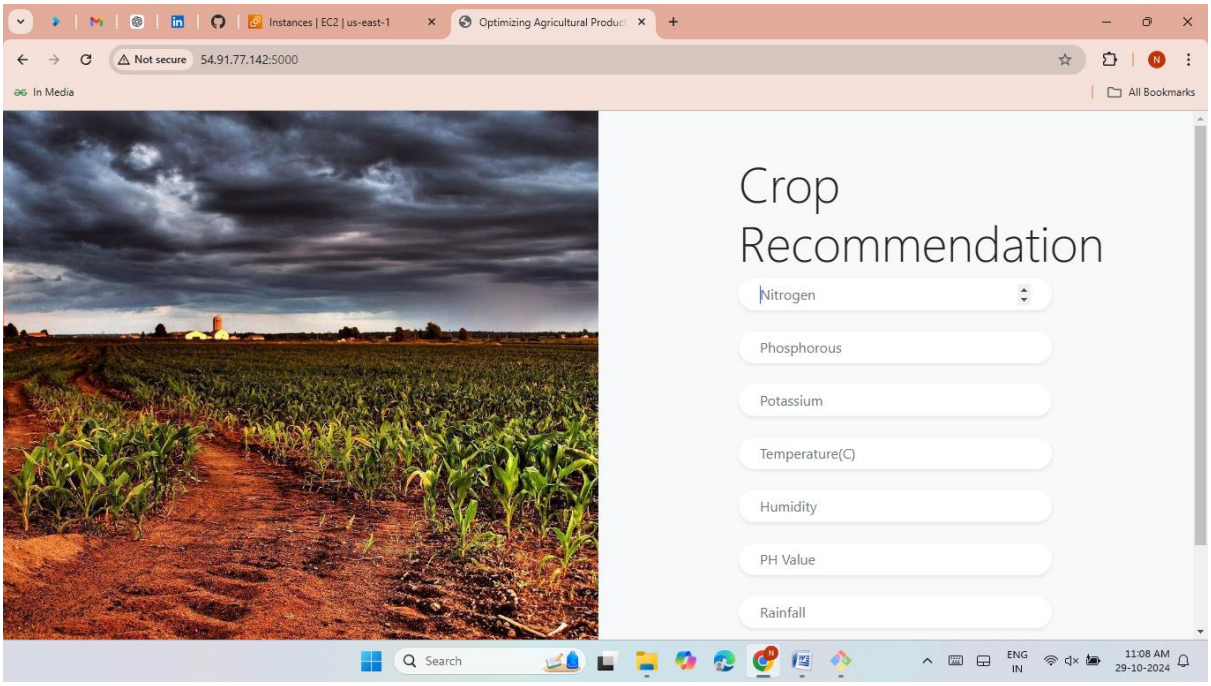

- Now clone the Agri python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run Agri with the command.

  **python3 app.py or screen –m –d python3  app.py**



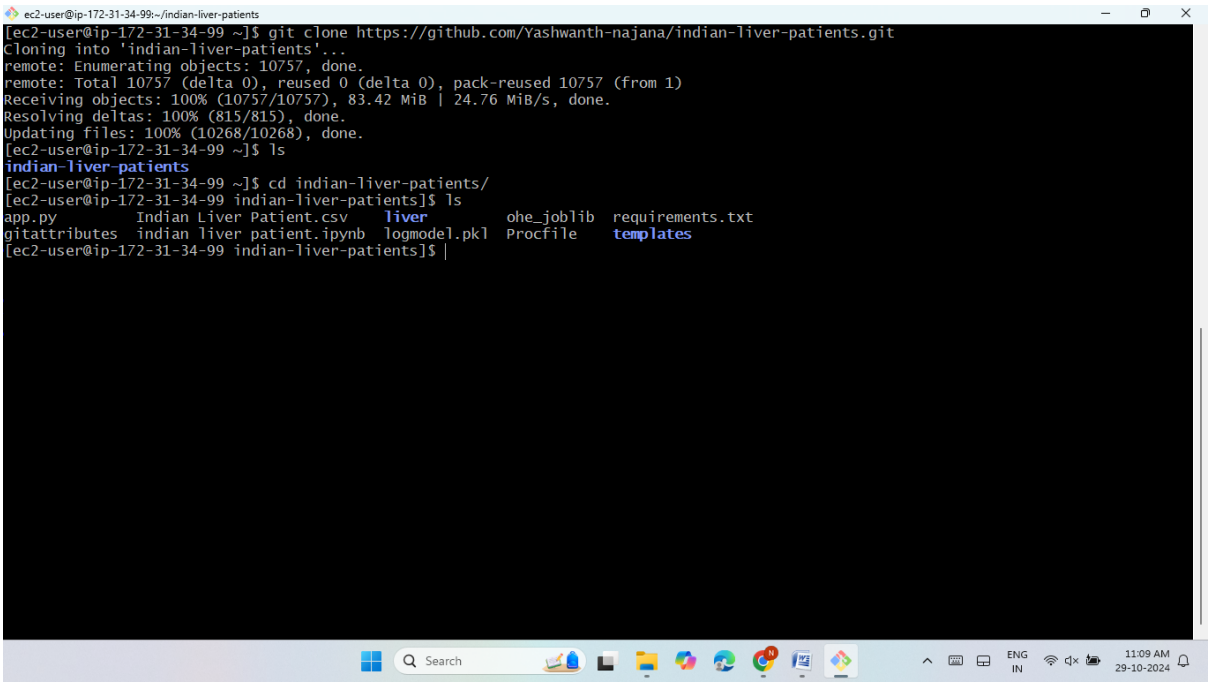
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the Agri is displays.



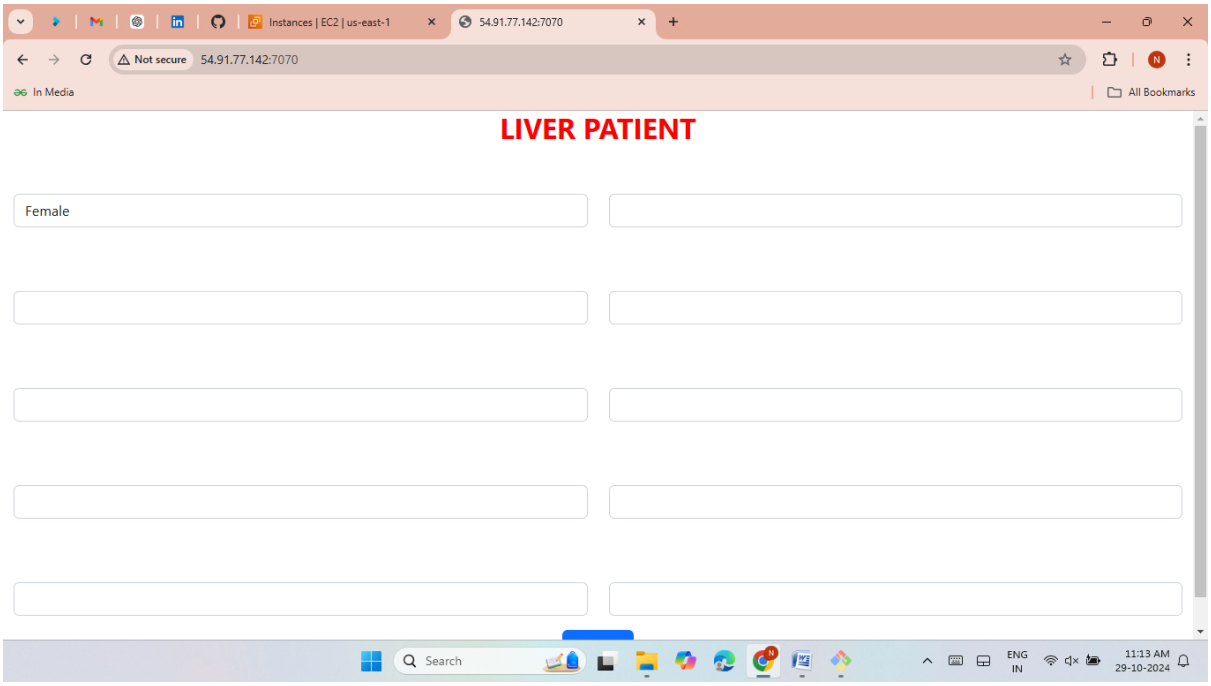

- Now clone the Indian-liver-patients python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run Indian-liver-patients with the command.

  **python3 app.py or screen –m –d python3  app.py**



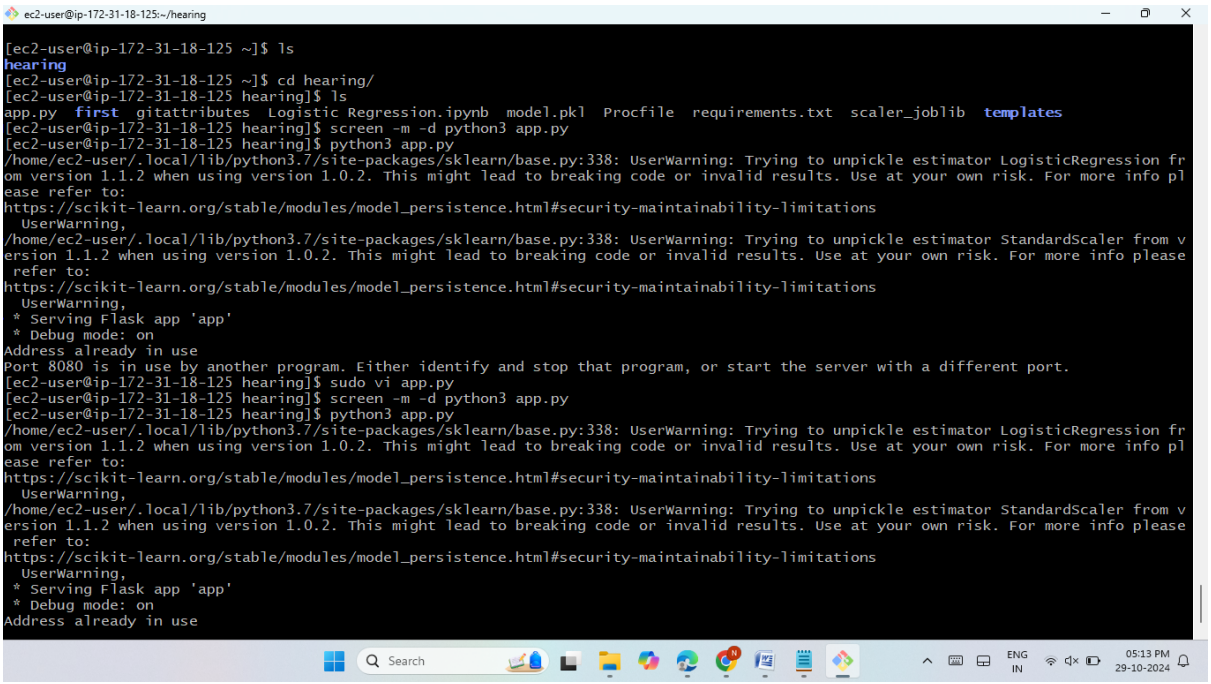
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the Indian-liver-patients is displays.



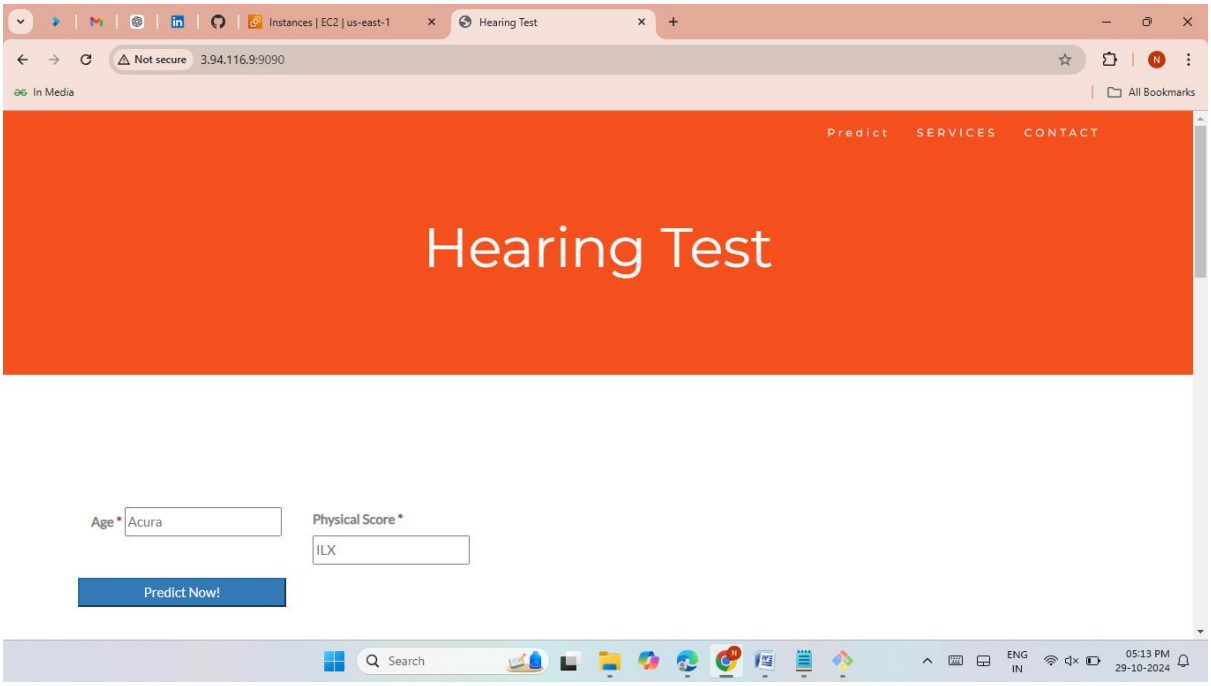

- Now clone the hearing python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run hearing with the command.

  **python3 app.py or screen –m –d python3  app.py**



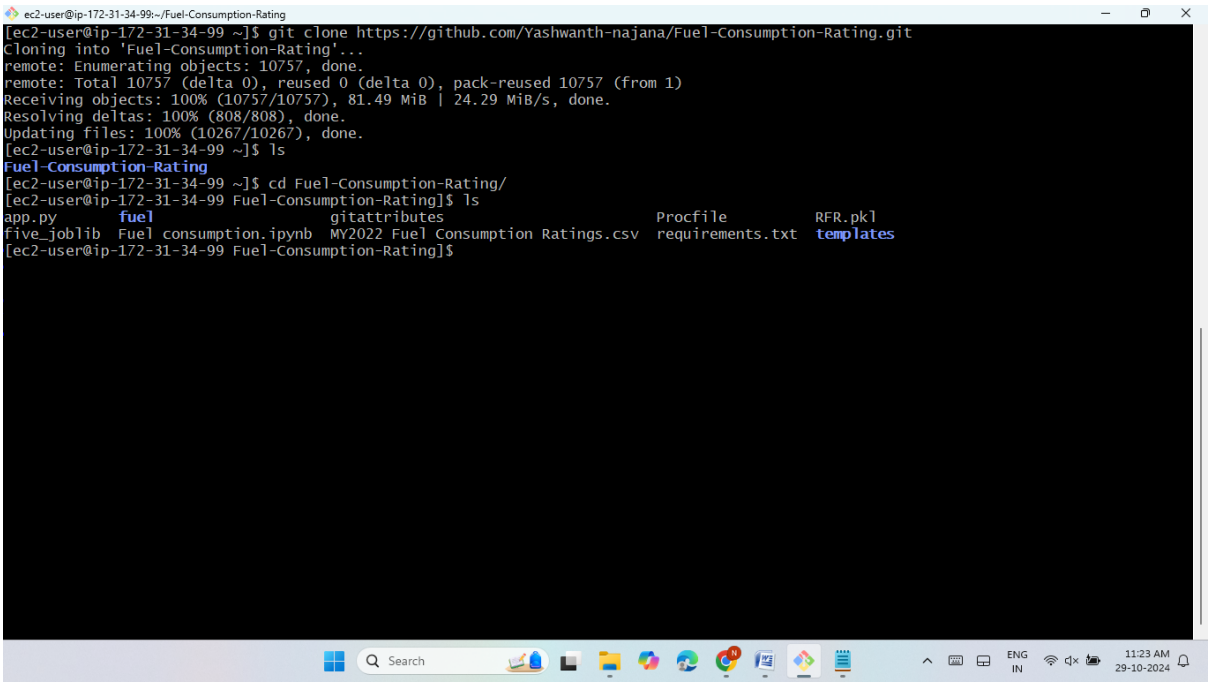
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the hearing is displays.



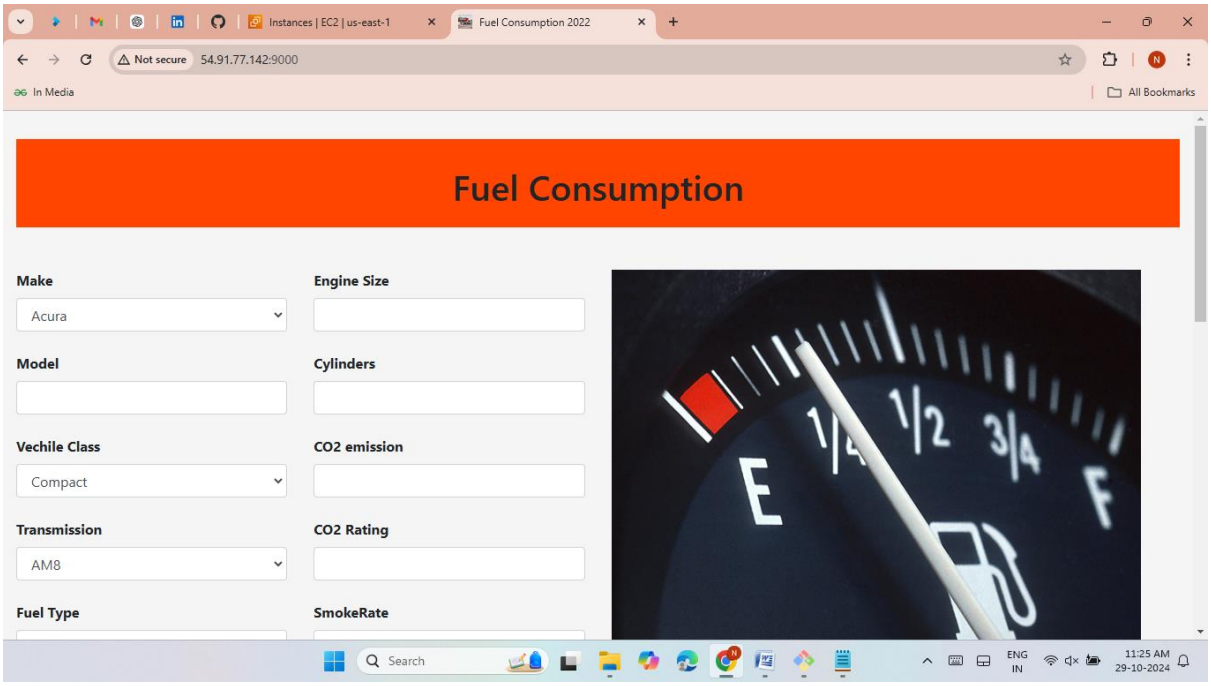

- Now clone the Fuel-Consumption-Rating python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run Fuel-Consumption-Rating with the command.

  **python3 app.py or screen –m –d python3  app.py**



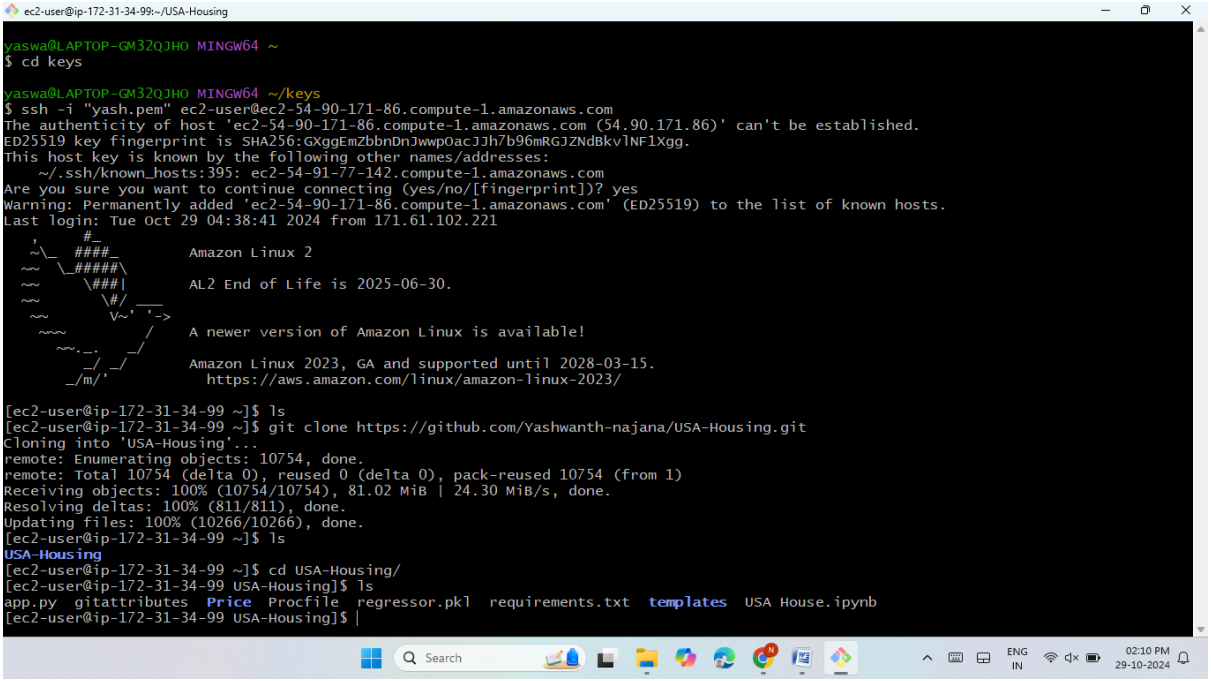
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the Fuel-Consumption-Rating is displays.



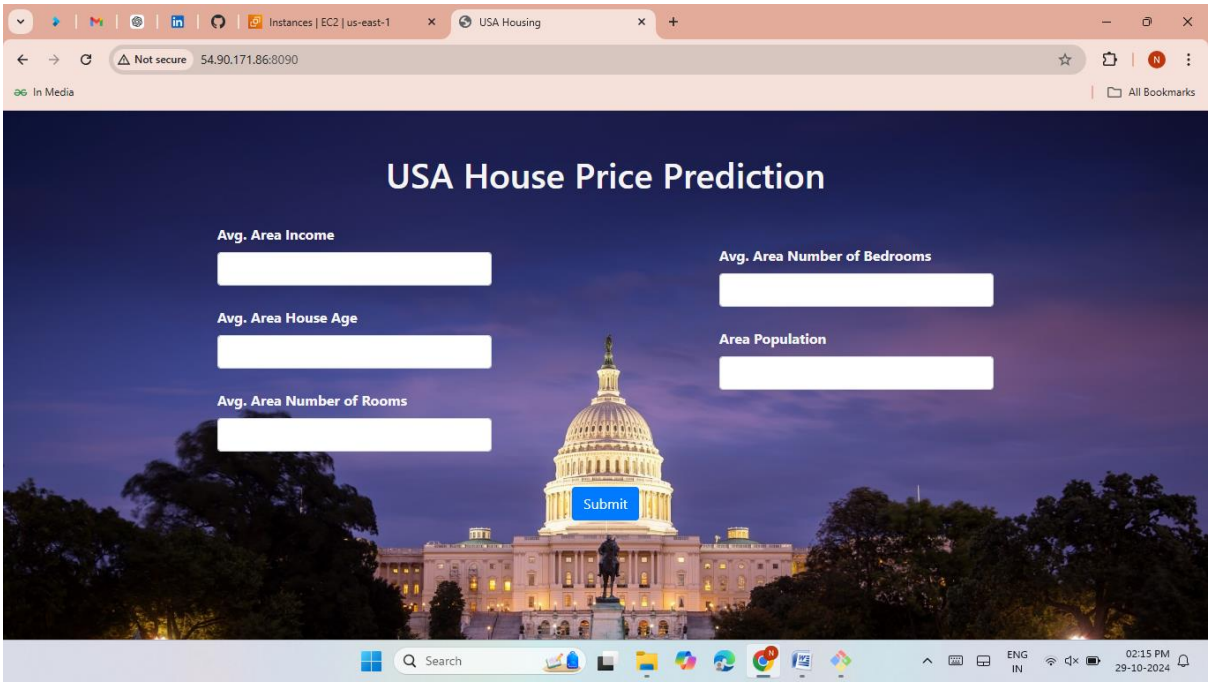

- Now clone the USA-Housing python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run USA-Housing with the command.

  **python3 app.py or screen –m –d python3  app.py**



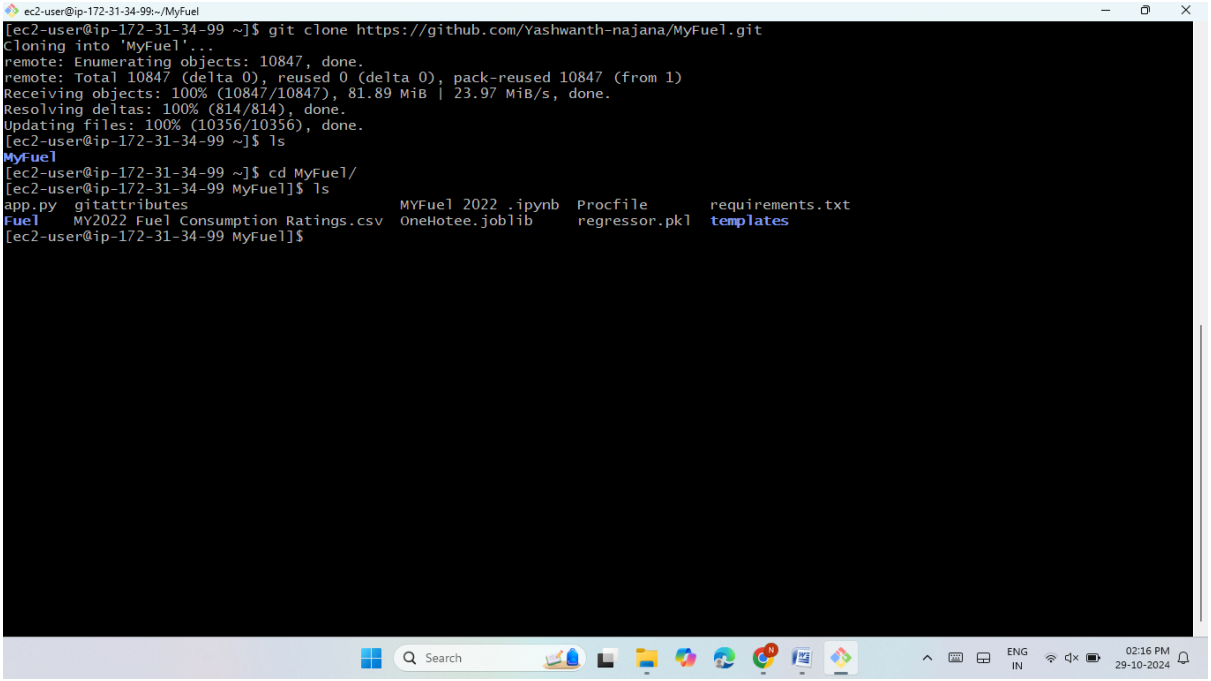
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the USA-Housing is displays.



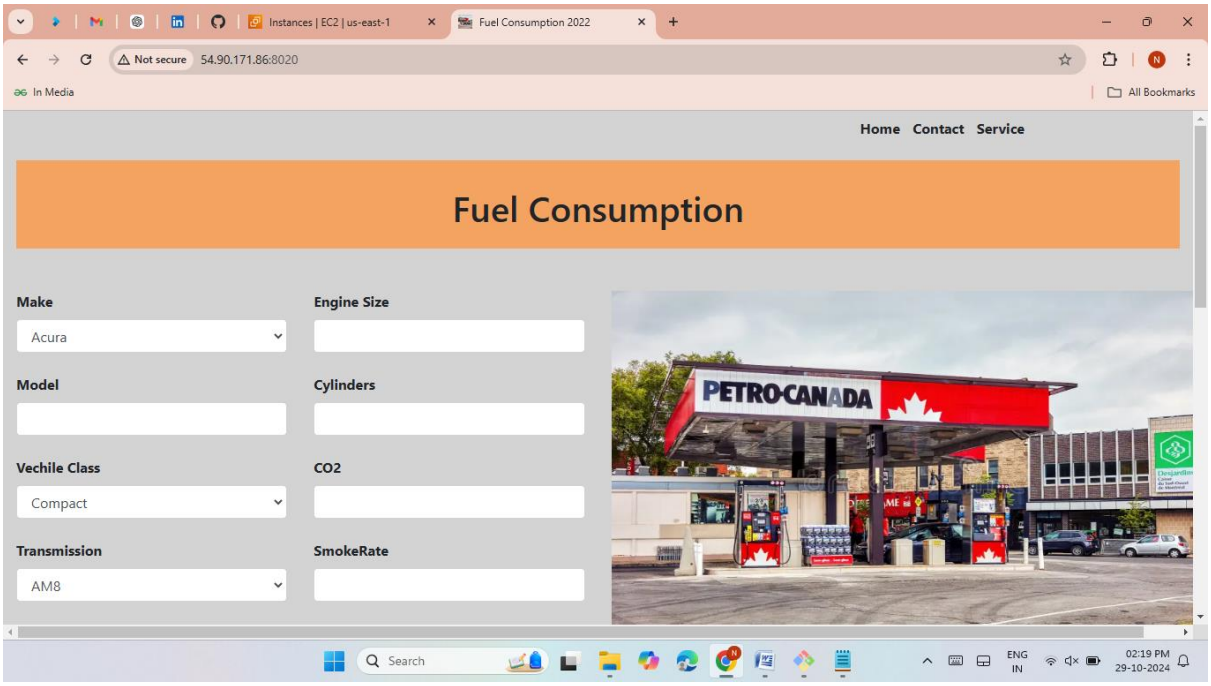

- Now clone the MyFuel python web applications.
- Now install requirements packages with the command.

  **pip3 install –r requirements.txt**
- Now run MyFuel with the command.

  **python3 app.py or screen –m –d python3  app.py**



- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the MyFuel is displays.

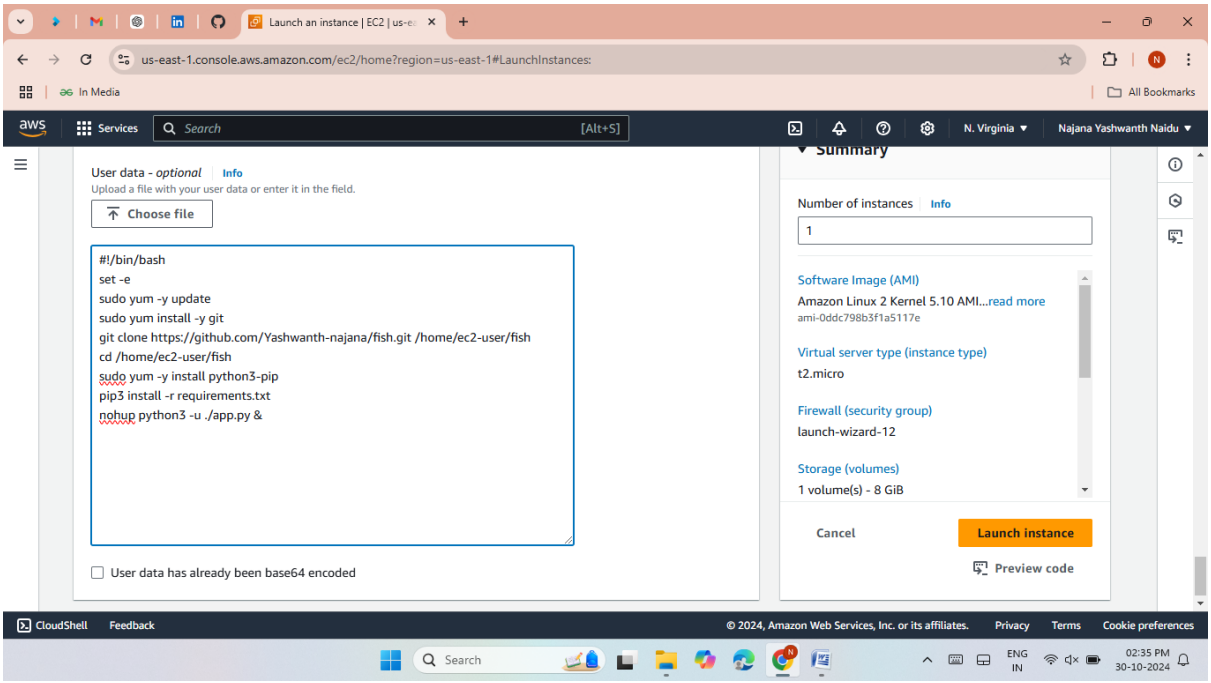



# METHOD-2

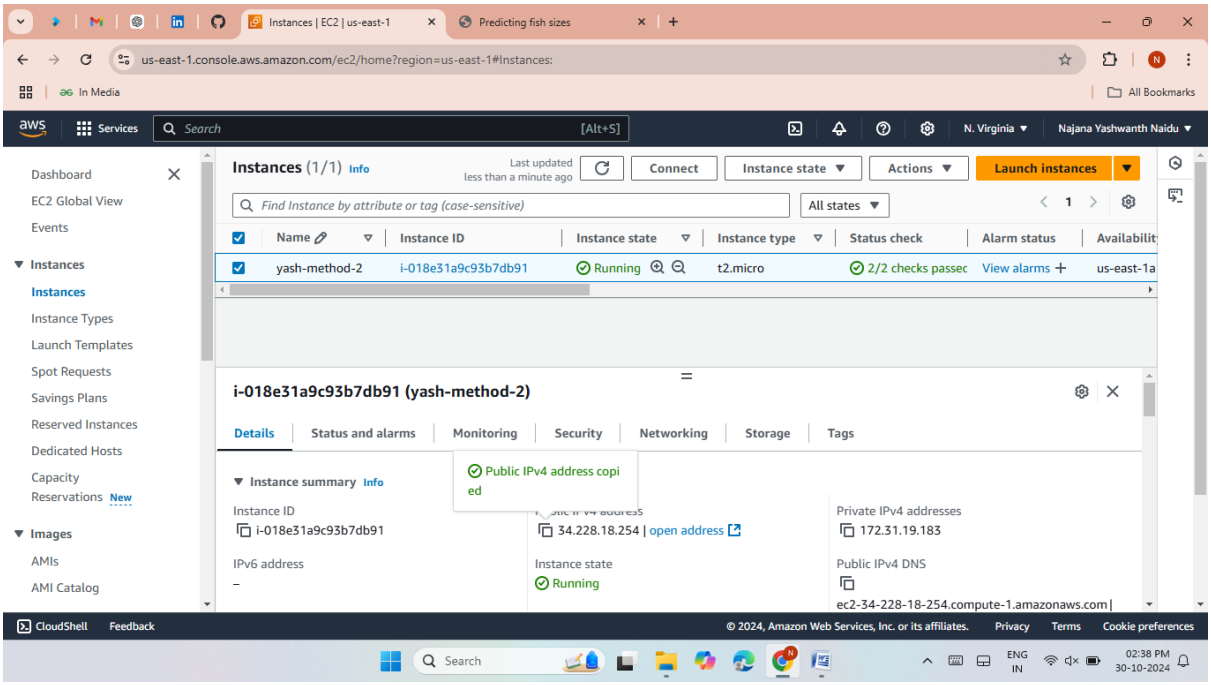
## Build and deploy python applications along with EC2 instance(userdata).

- First launch an ec2 instances with the user data script.

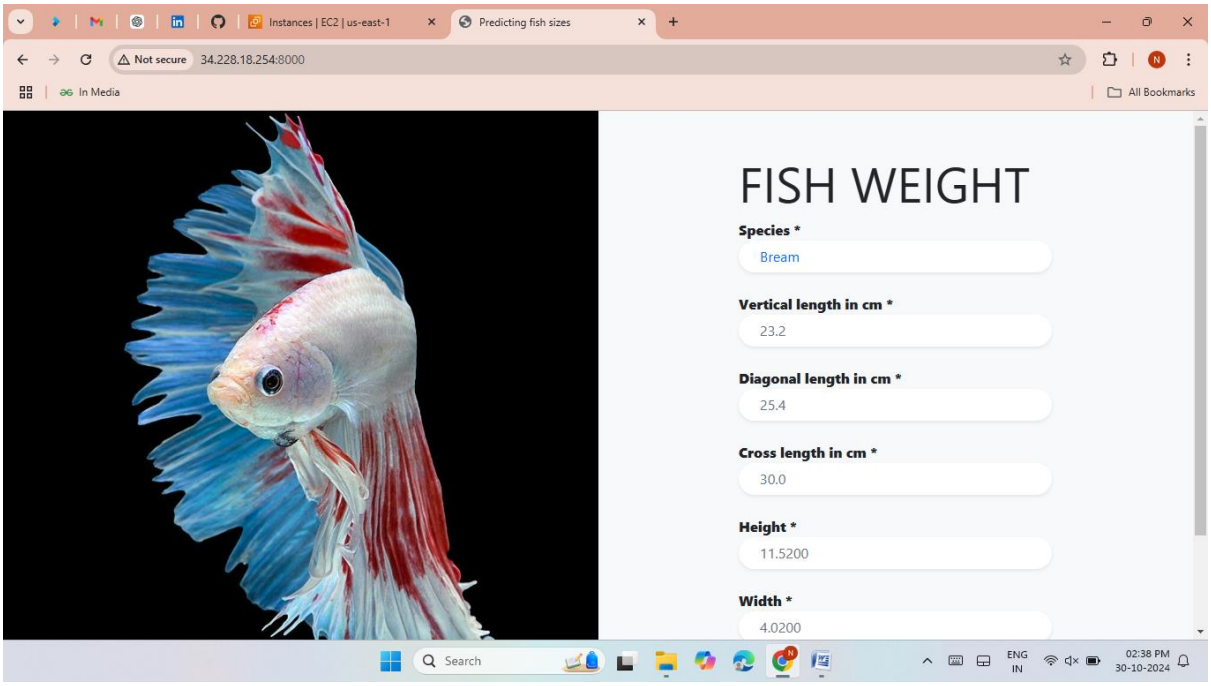
```
#!/bin/bash
set -e
sudo yum -y update
sudo yum install -y git
git clone https://github.com/Yashwanth-najana/fish.git /home/ec2-user/fish
cd /home/ec2-user/fish
sudo yum -y install python3-pip
pip3 install -r requirements.txt
nohup python3 –u ./app.py &
```




➢ Here the instance was launched with the user data.



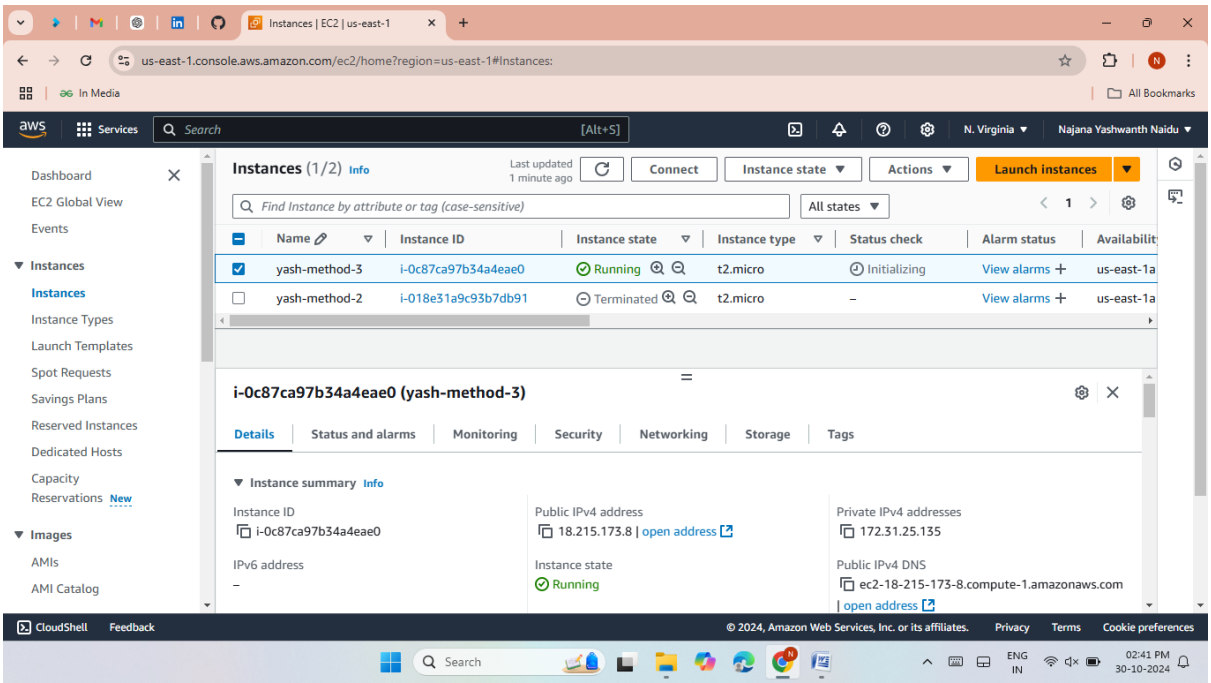
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish is displays.



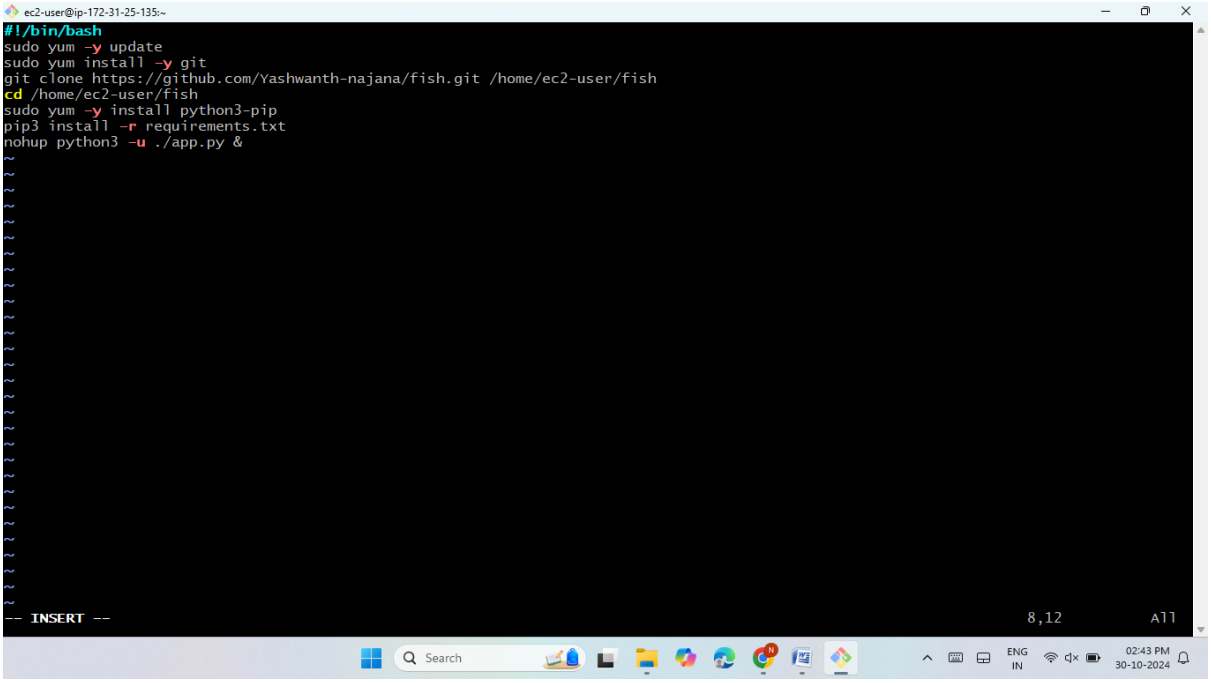

## METHOD-3

### Build and deploy python applications in Bash scripting

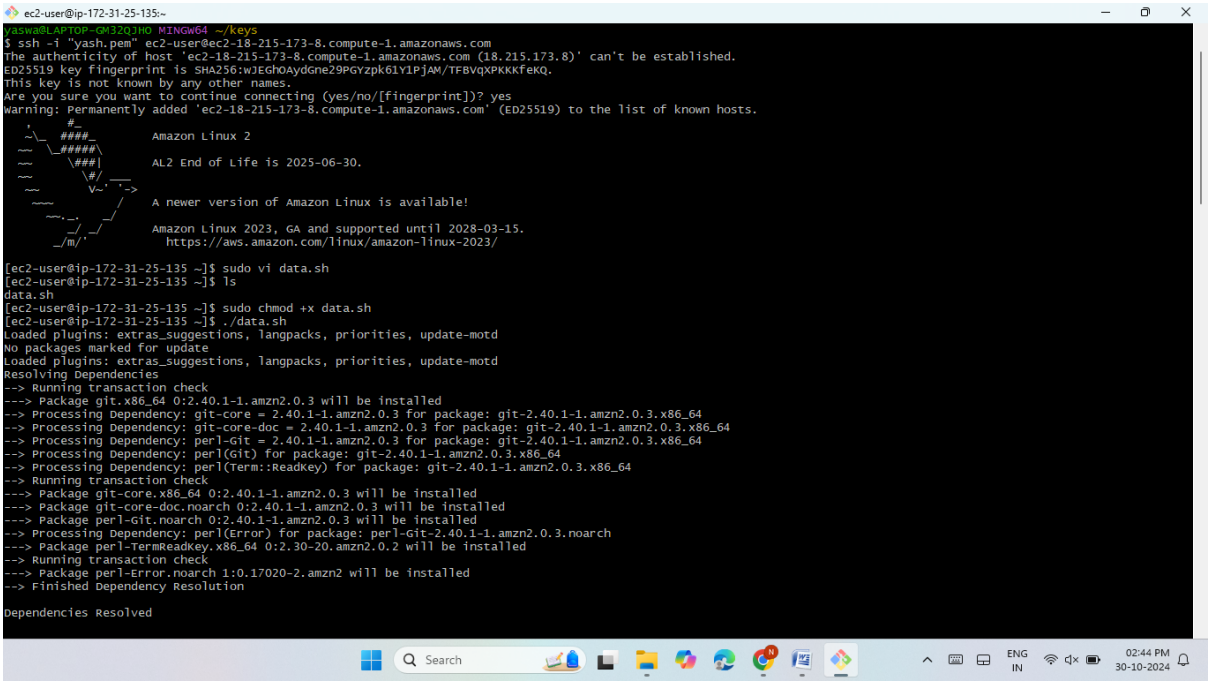
- First launch an EC2 instance with the required ports.

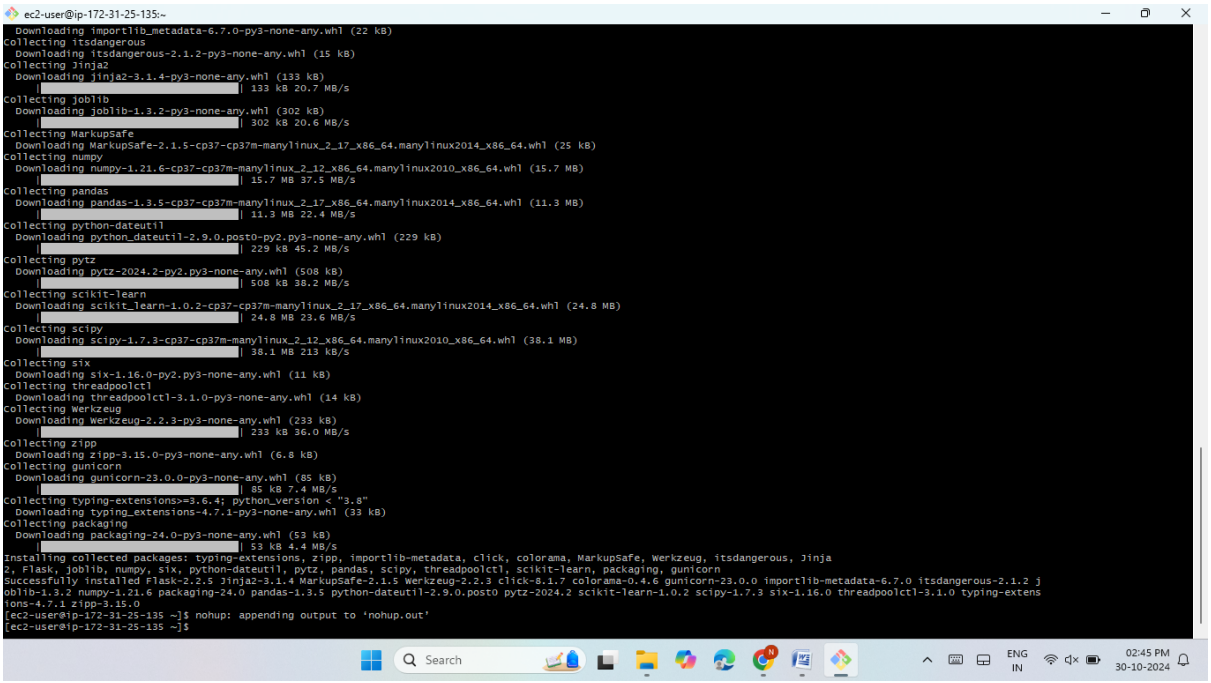


- Now write the bash script for deploy the python application with **sudo vi** mode.

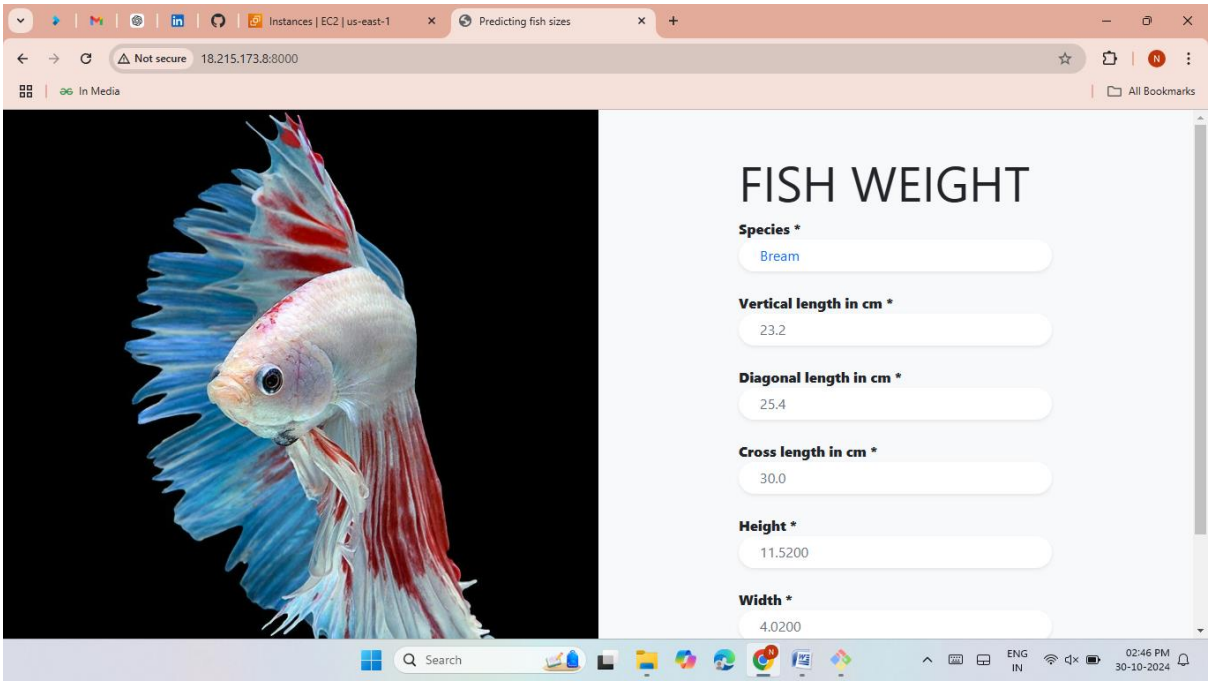



- Now run the bash script with the command **./<file name>.sh**





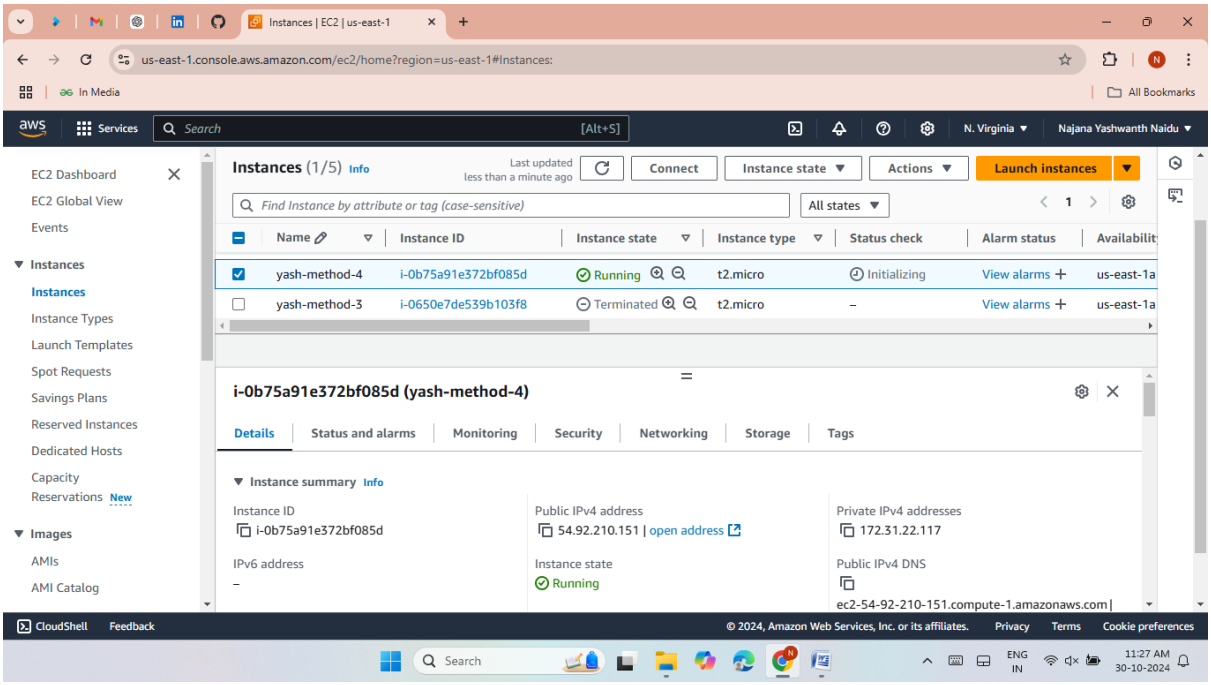

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish is displays.



# METHOD-4

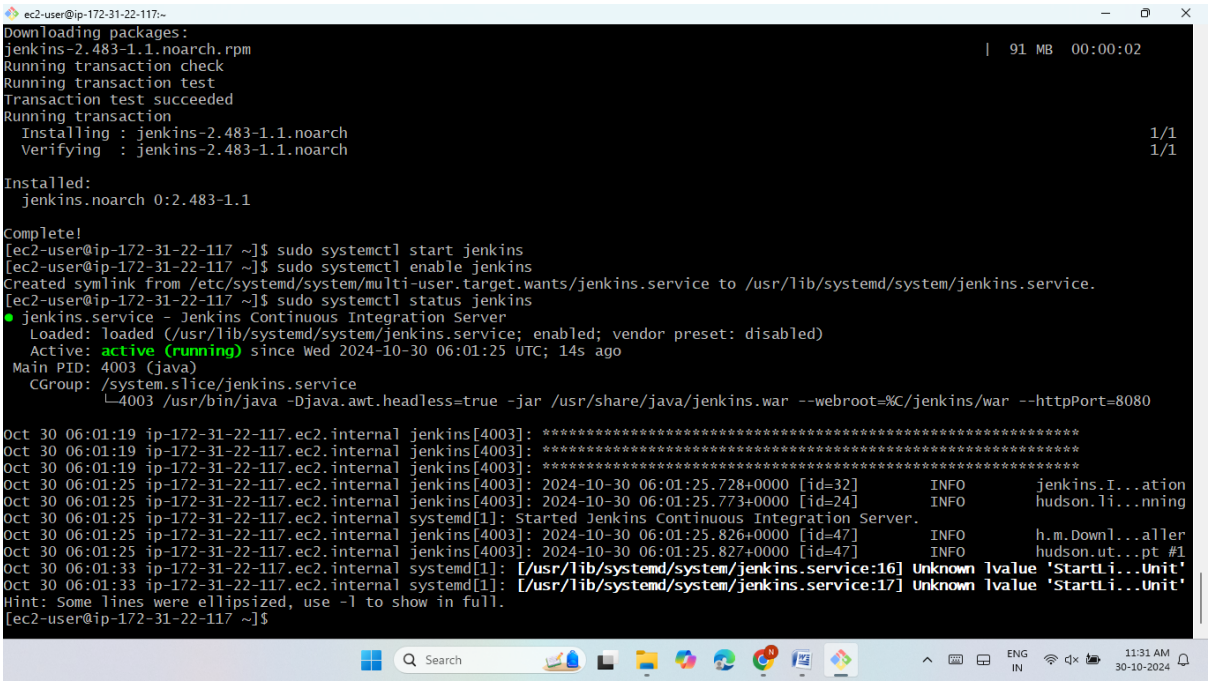
## Build and deploy python applications with Git, Github and Jenkins (execute shell).

- Launch EC2 instance with Jenkins port number(8080) and given python applications port.

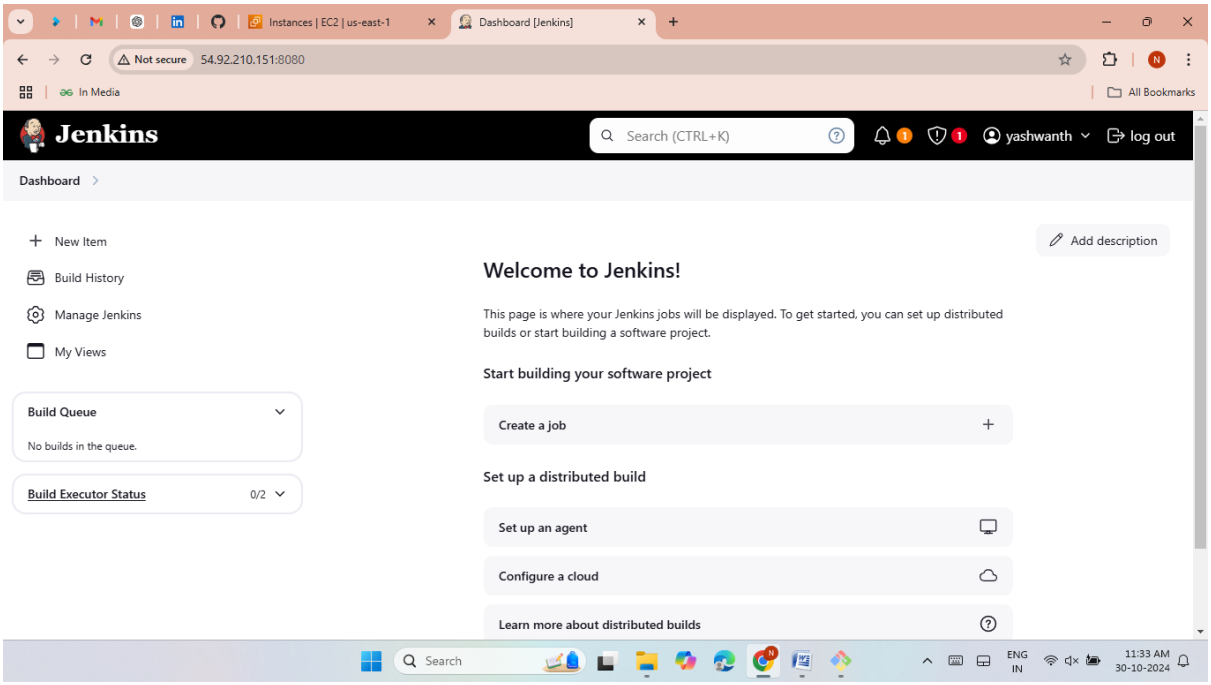



- Now install Jenkins and then start and enable with the commands.

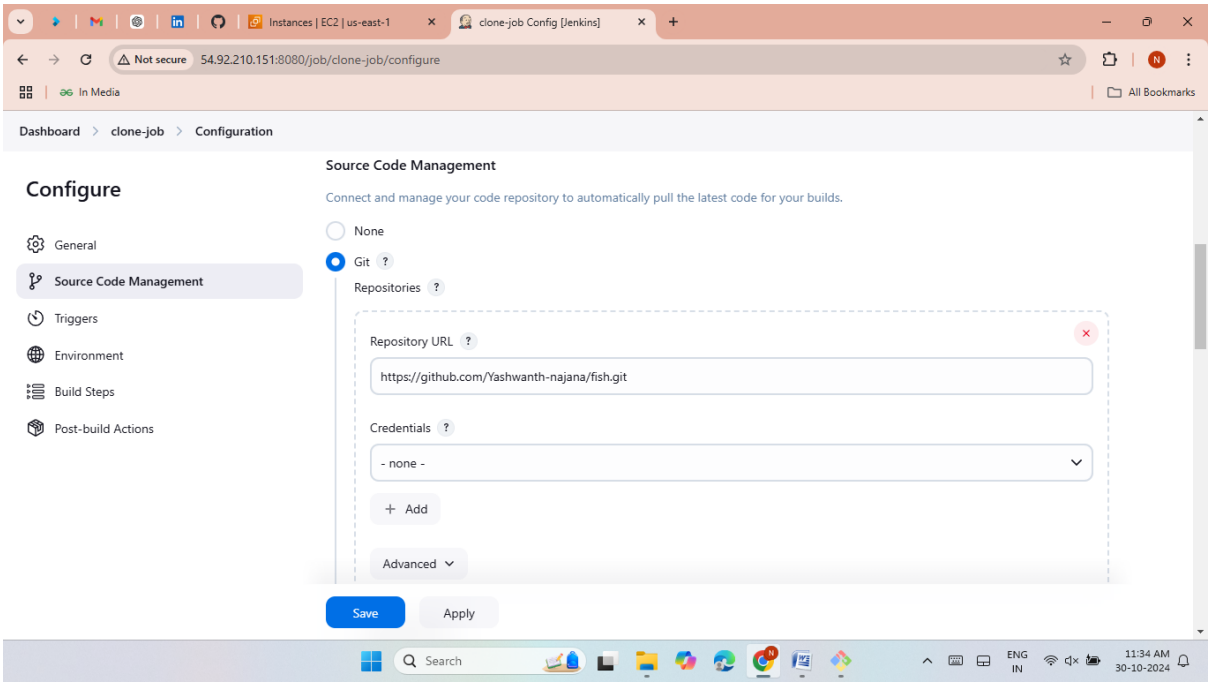
```
sudo yum install –y java-17*
sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io-2023.key
sudo yum upgrade -y
sudo yum install –y jenkins
sudo systemctl start Jenkins
sudo systemctl enable jenkins
```



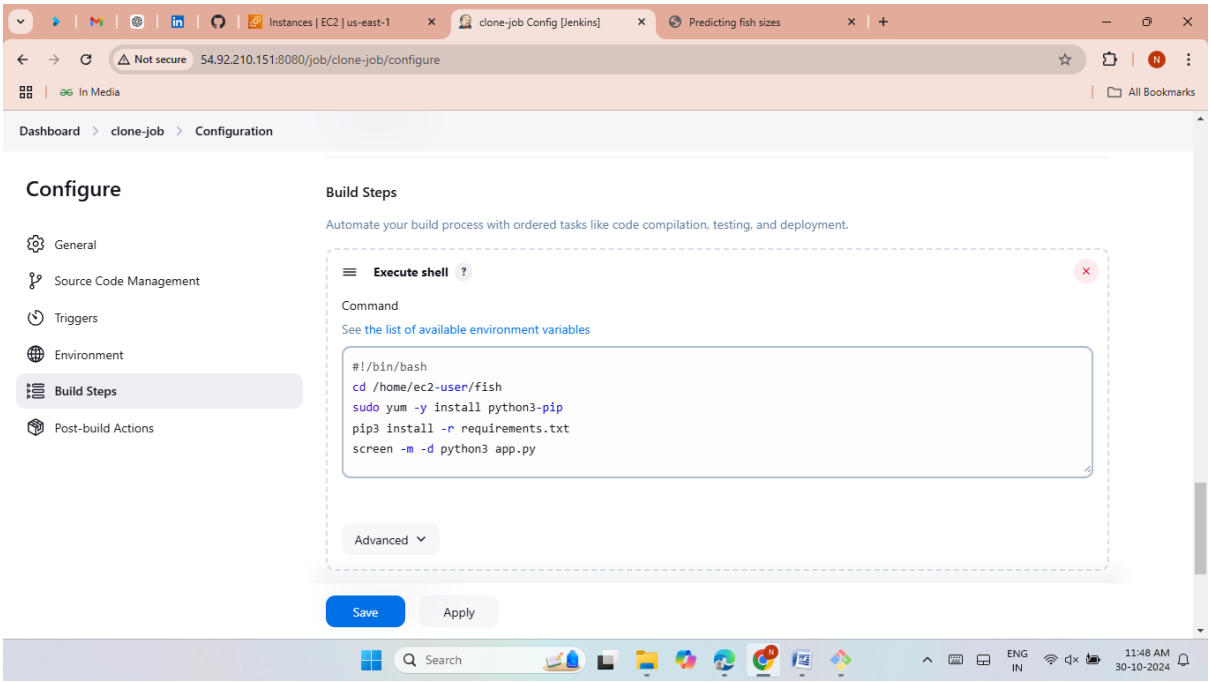

- Here the Jenkins was launched.



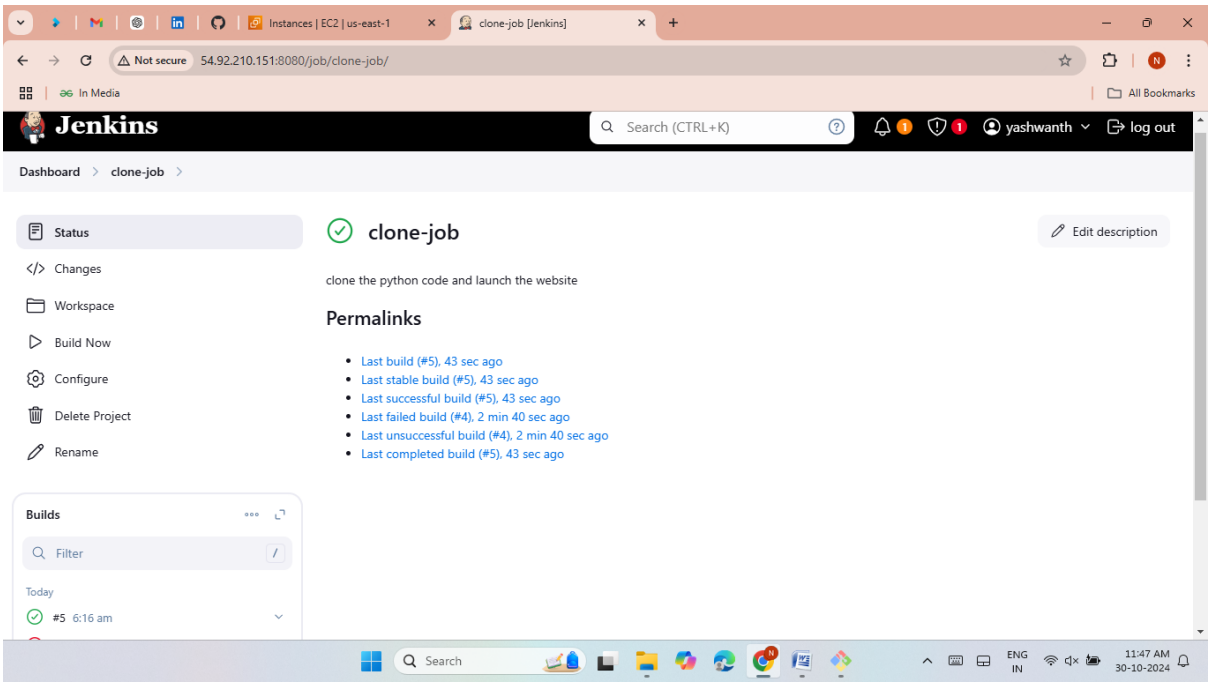
- Create a job for cloned the repo.
- Now Clone the repository URL.



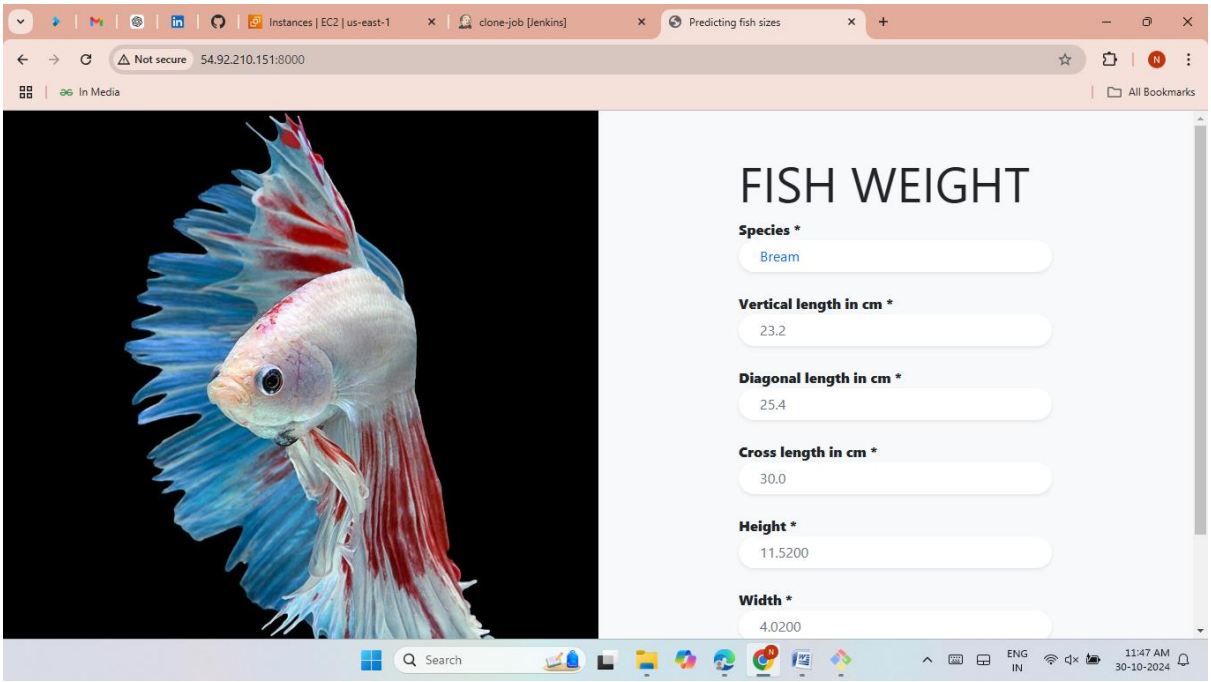

- Now write the script in execute shell.



- Now click on build. Then job is success.



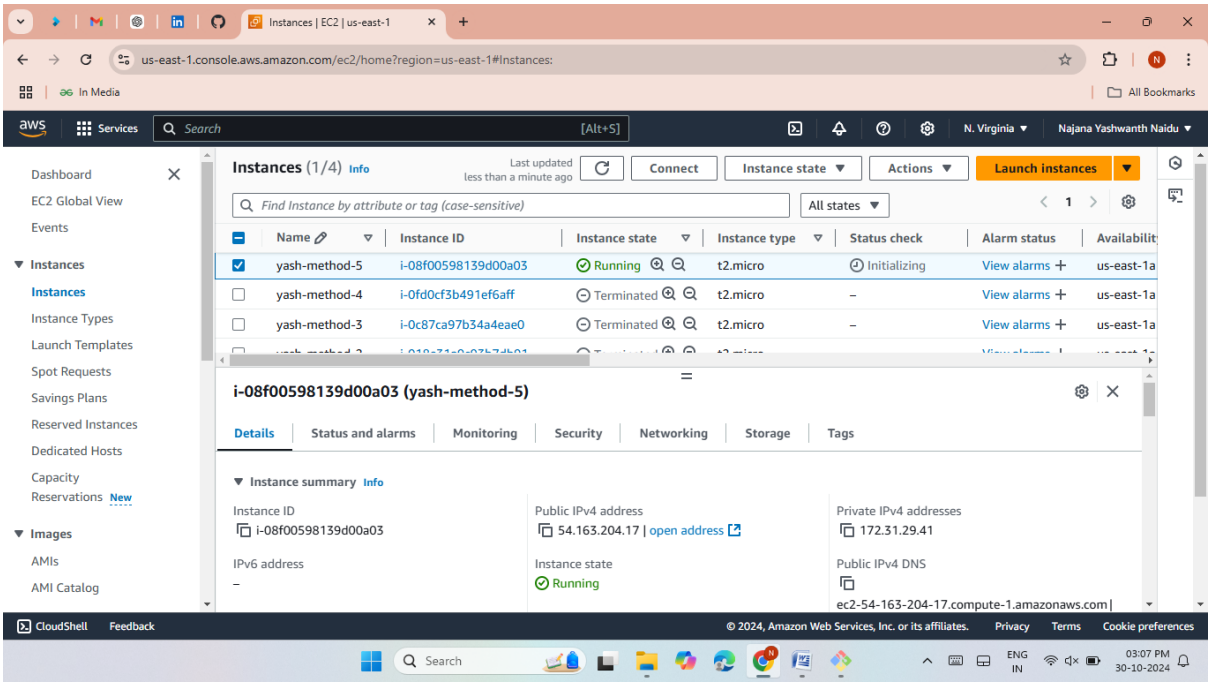

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish python application is displays.



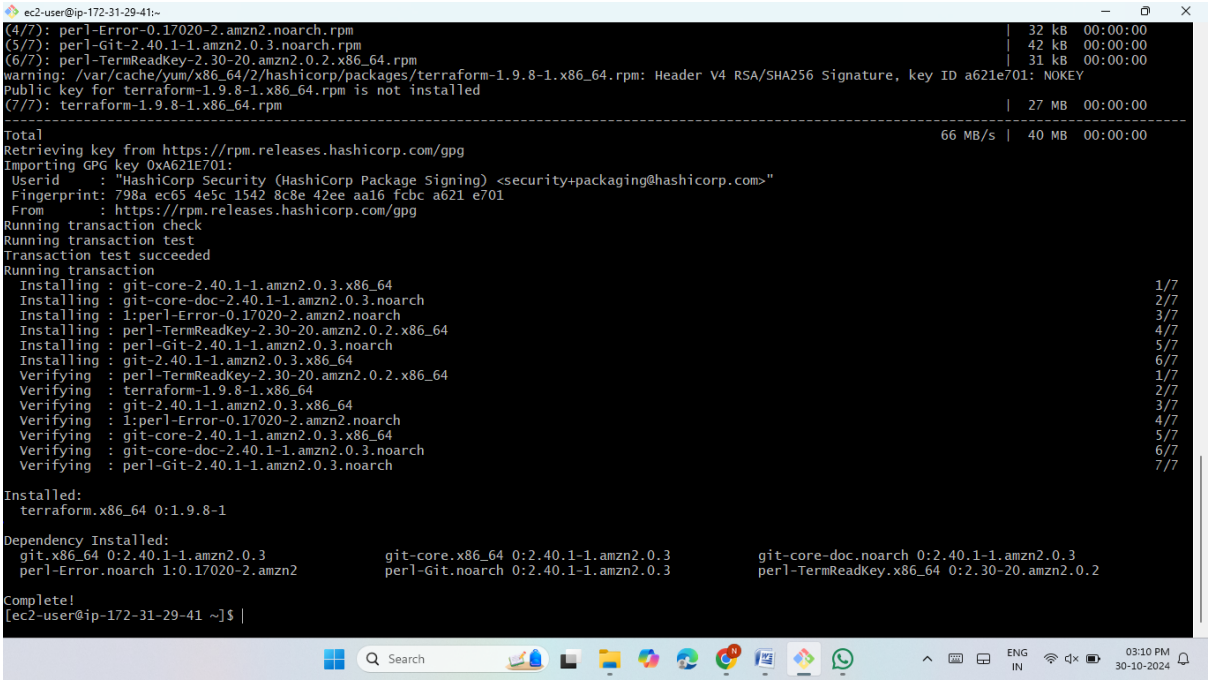
# METHOD-5

**Build and deploy python applications with the Terraform(data.sh) and push the Terraform Scripted files in Github.**

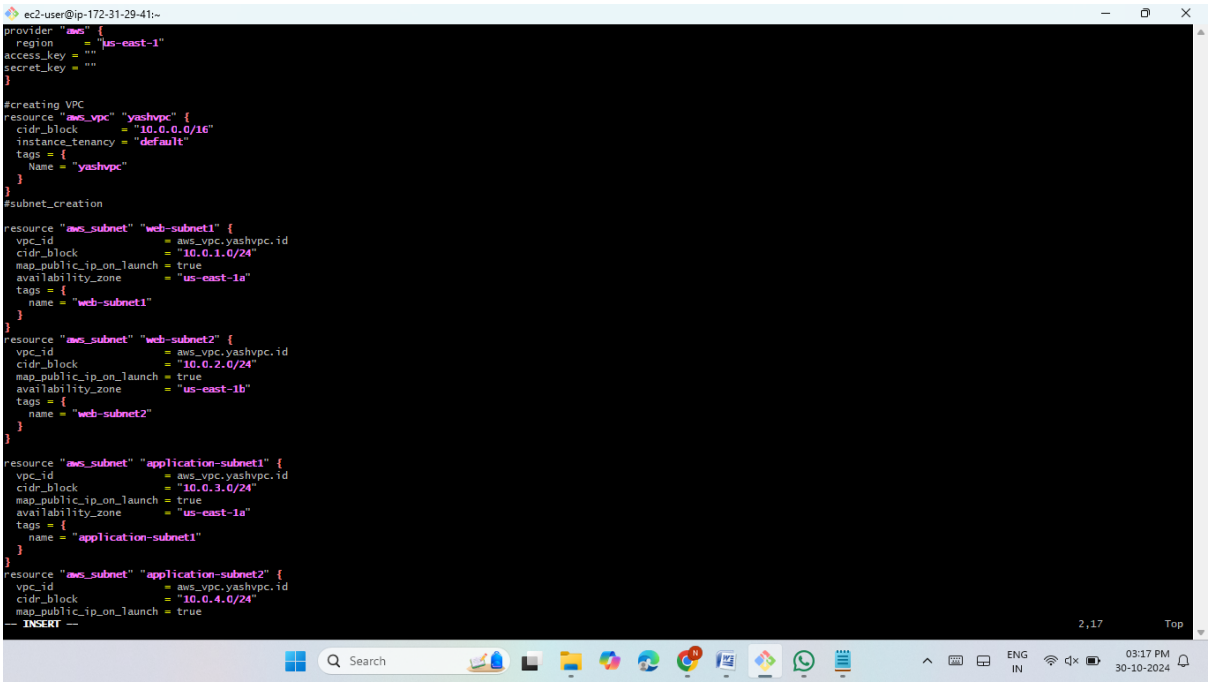
- First Launch an EC2 instance in our AWS account.

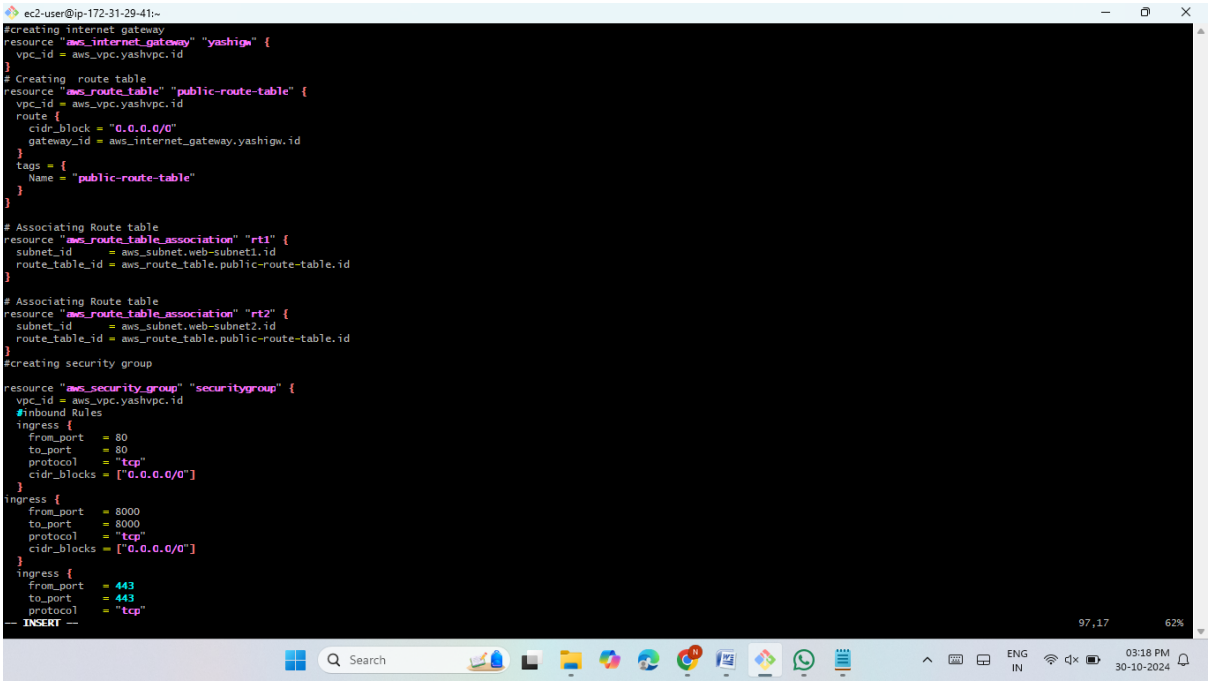


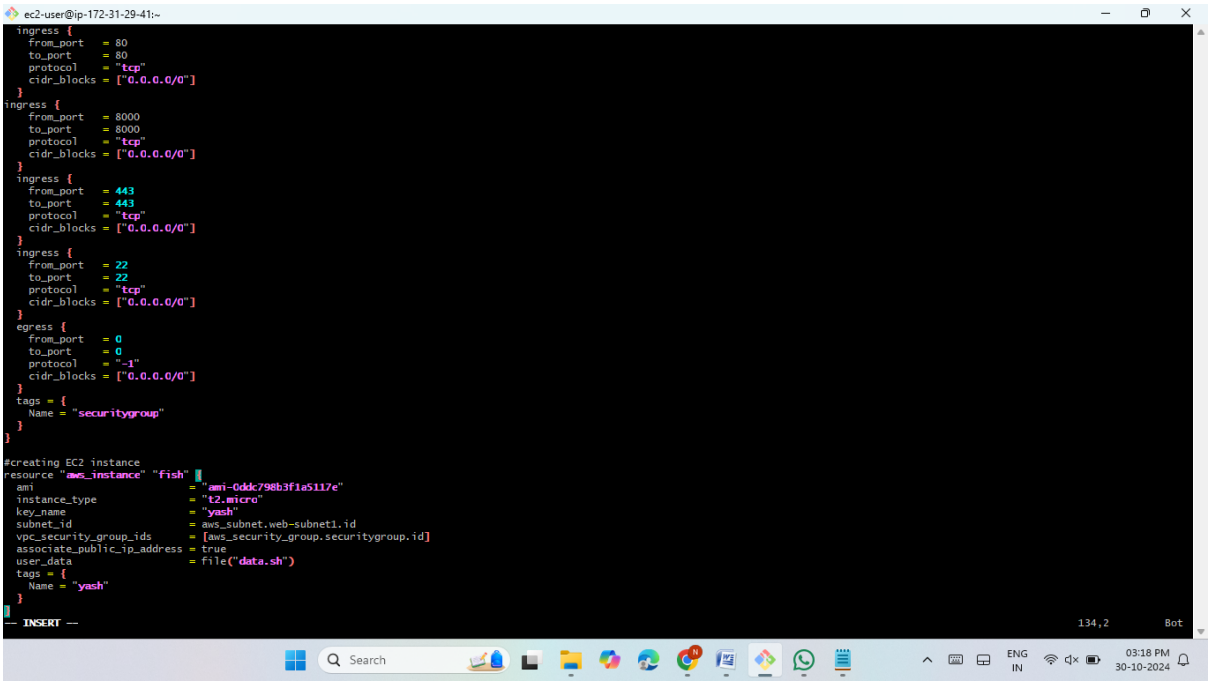

- Now install terraform form Google browser.



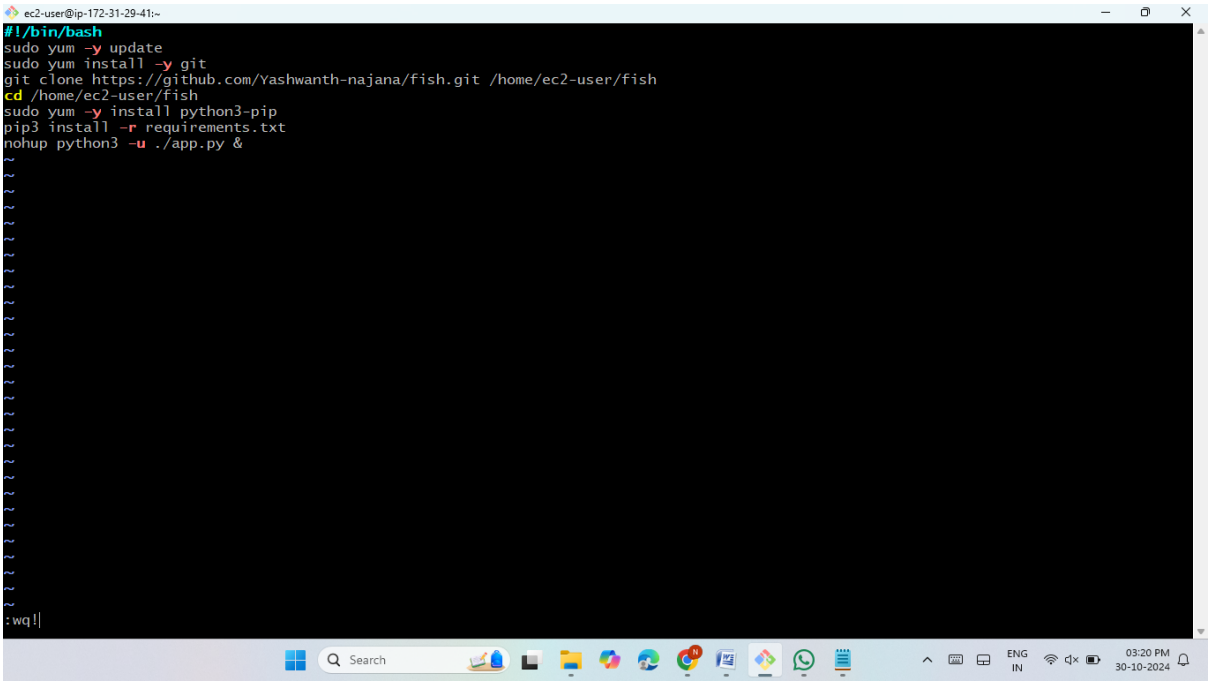
- Write a terraform script for launch an EC2 instance.
- Provide Access key and Secret key.
- Write script for EC2 and Security Groups, VPC, Subnets, Internet Gateway, Route Table.






- Now write the script for data file.
- Clone the fish python application form github.



- Now Enter the following commands

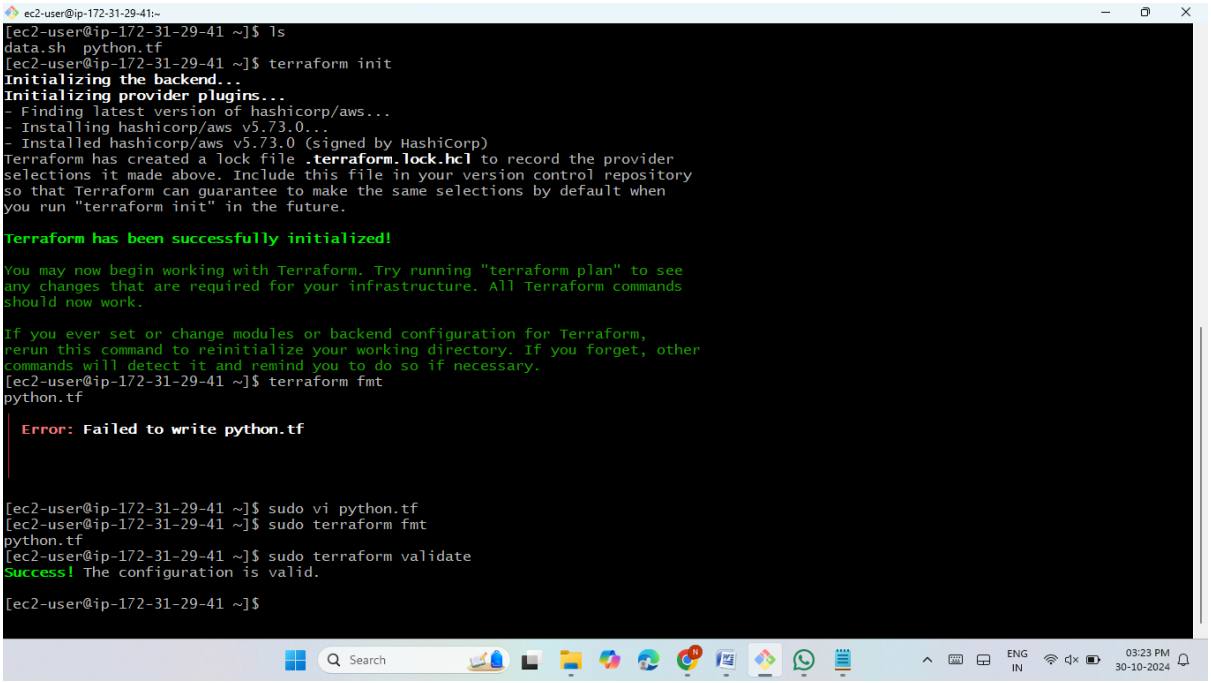
  **terraform init**

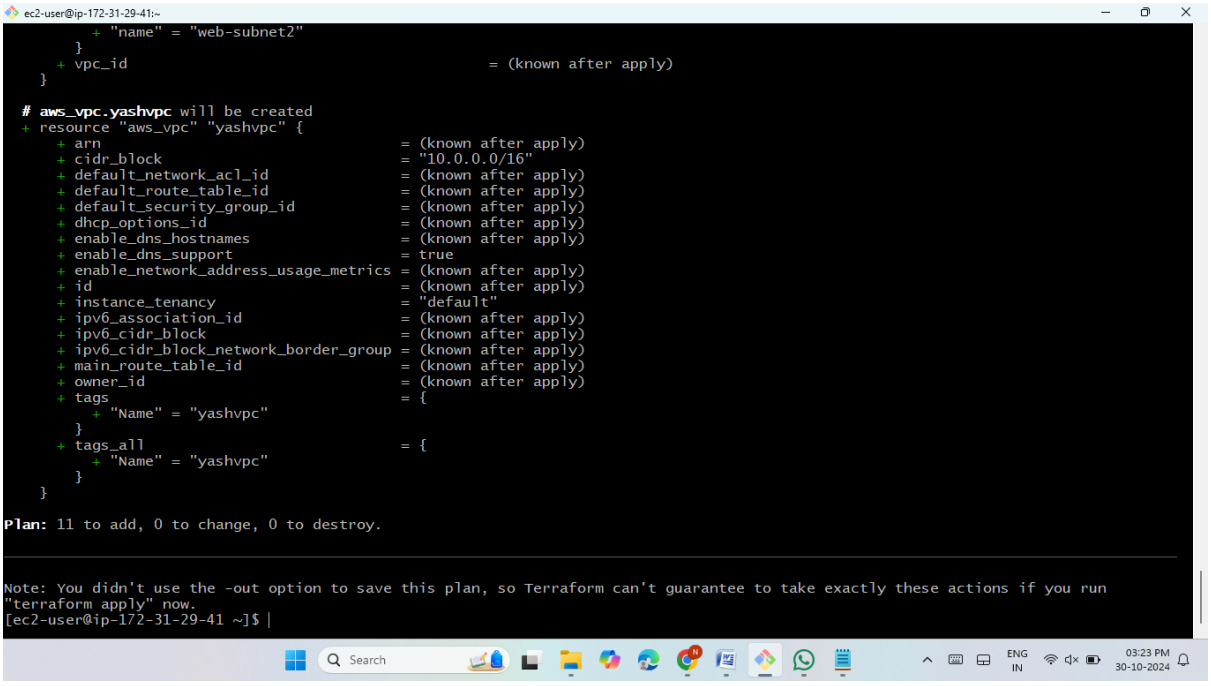
  **terraform fmt**

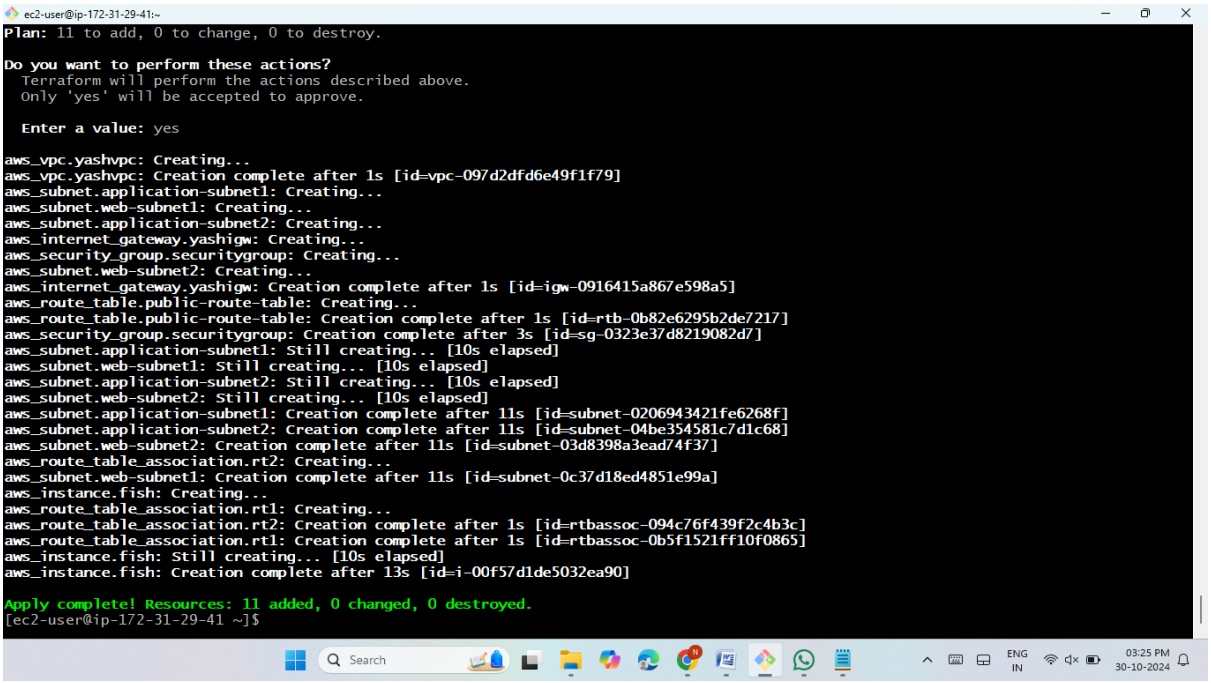
  **terraform validate**

  **terraform plan**

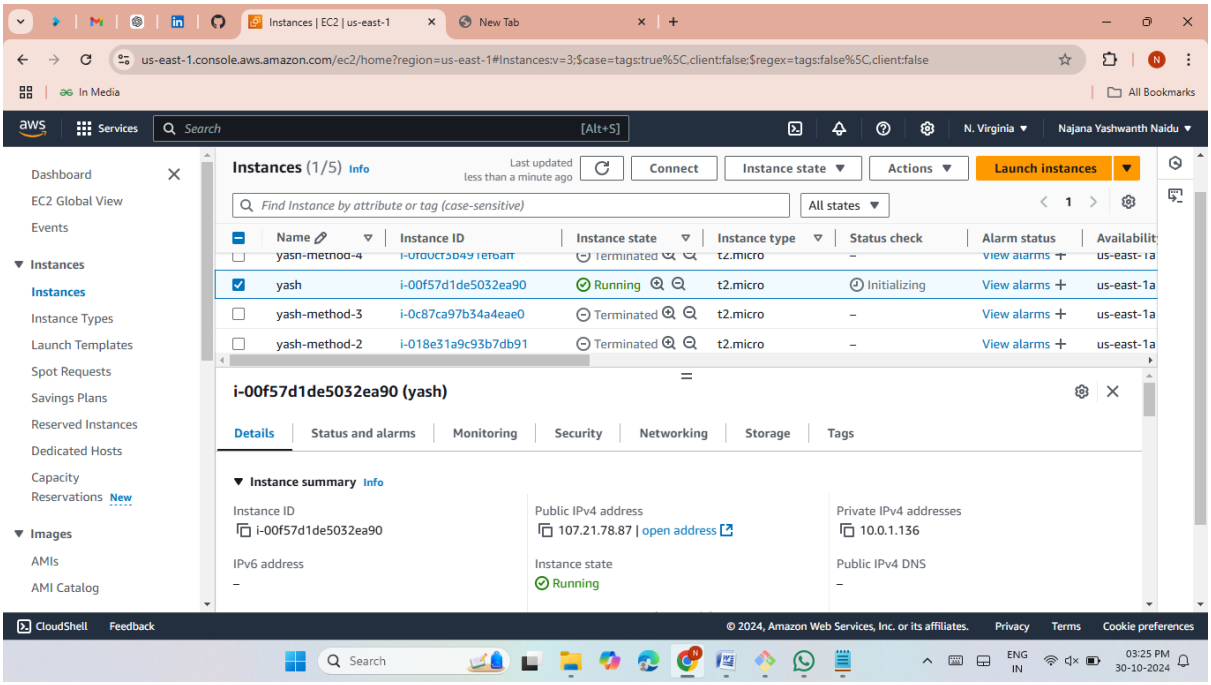
  **terraform apply**



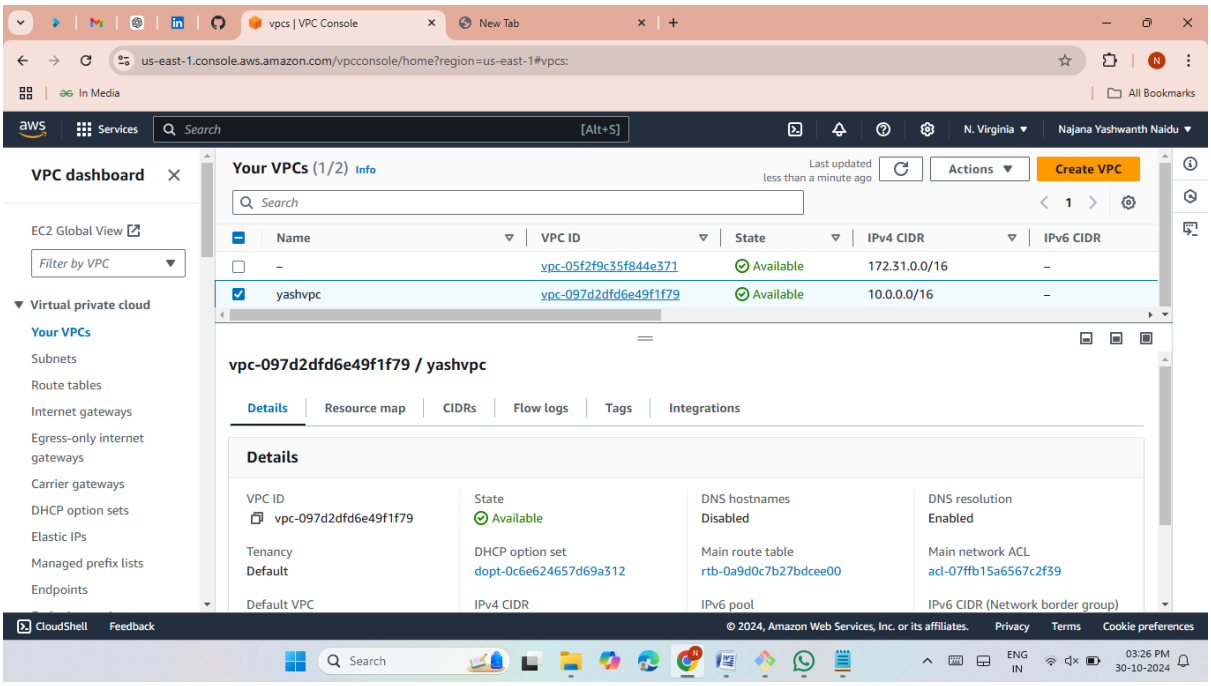





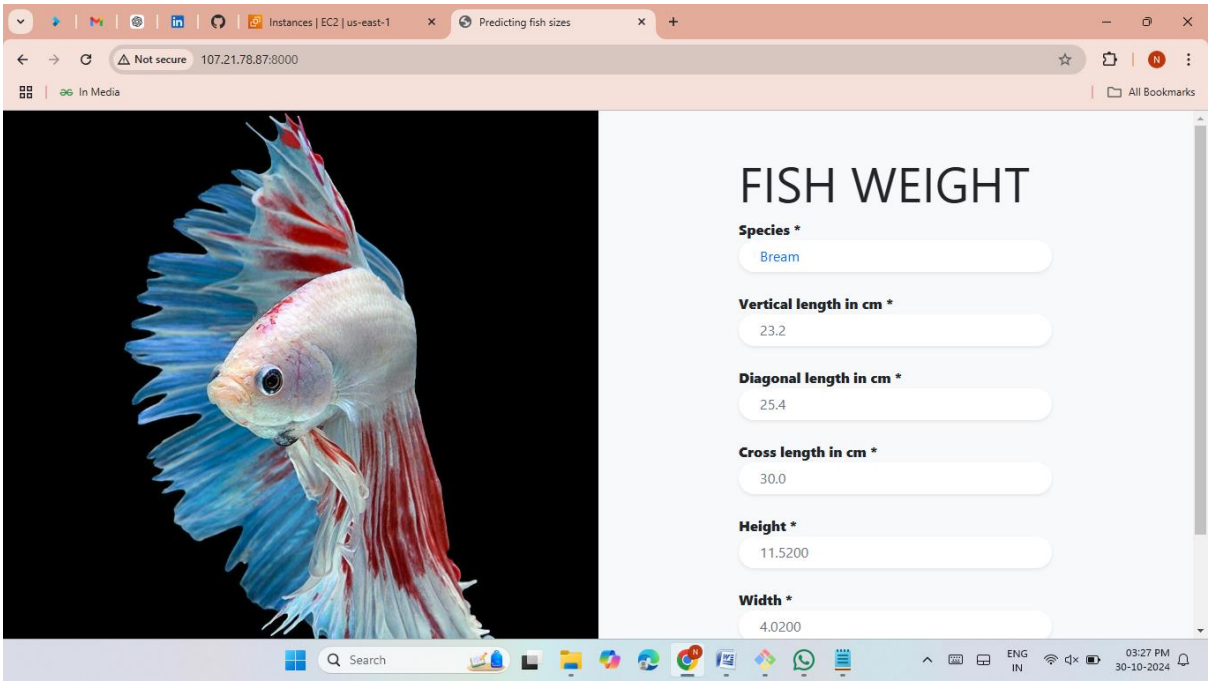

- Here the EC2 instance was launched.



- Here the VPC, Subnet, Internet Gateway, Route Table, Security Groups also created.



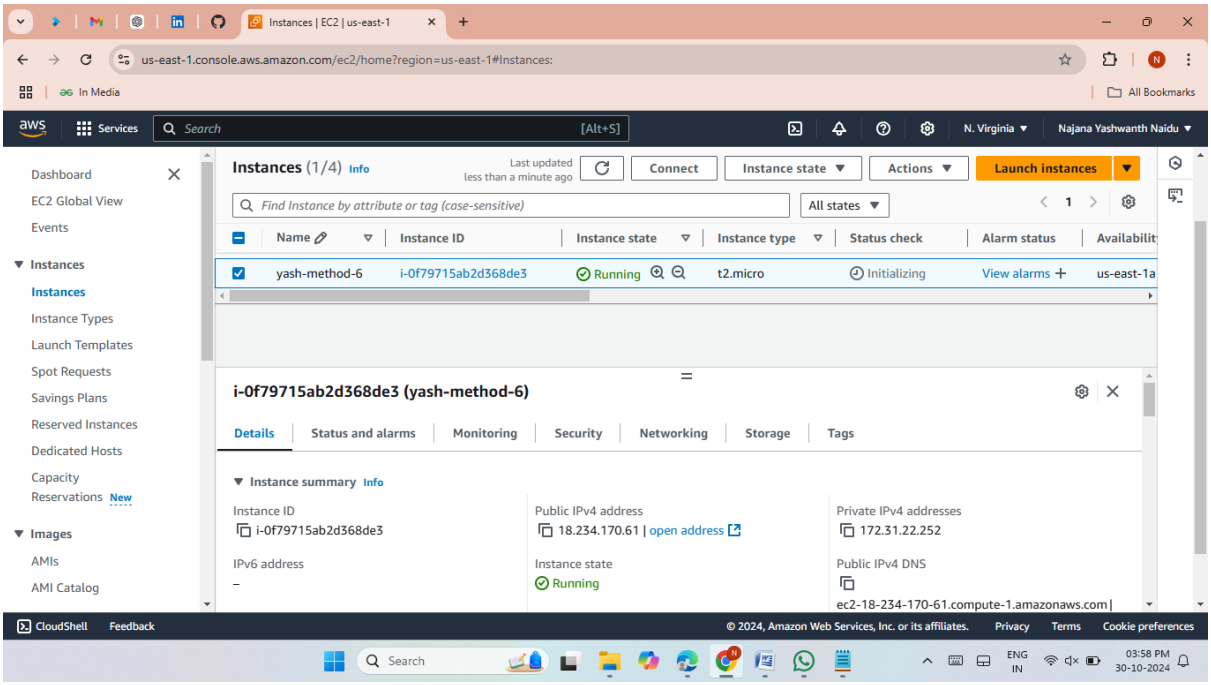

- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish python application is displays.



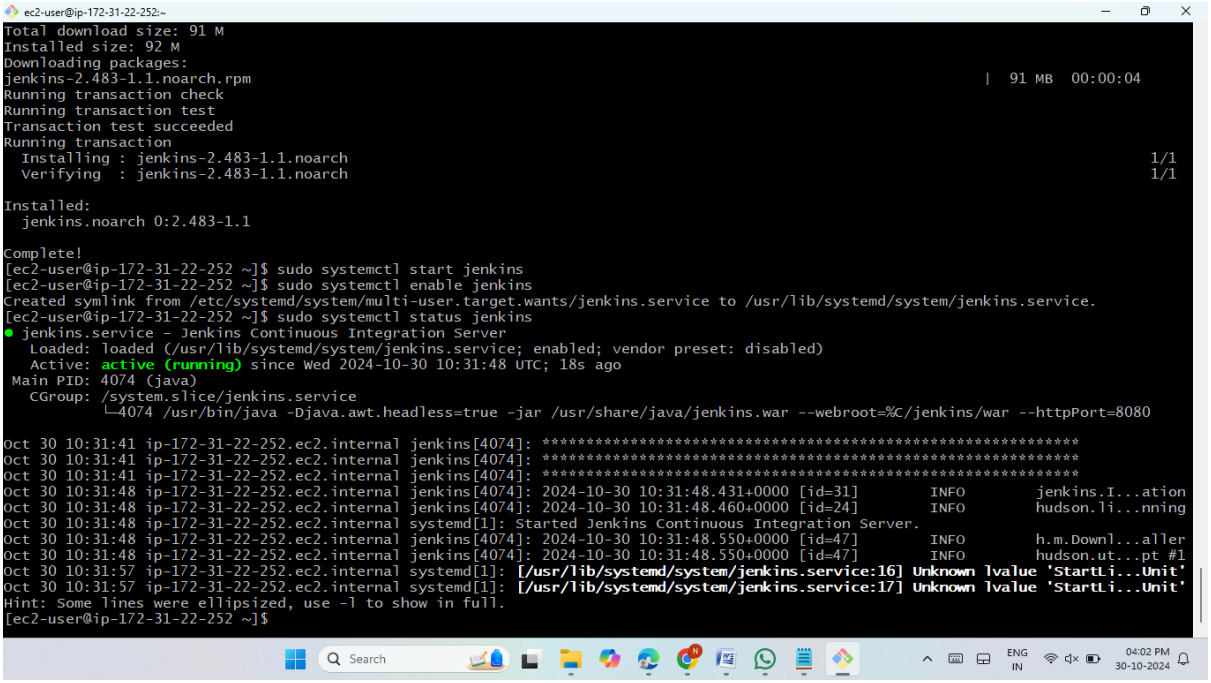
# METHOD-6

## Build and deploy python applications with Git, github, Jenkins and Terraform

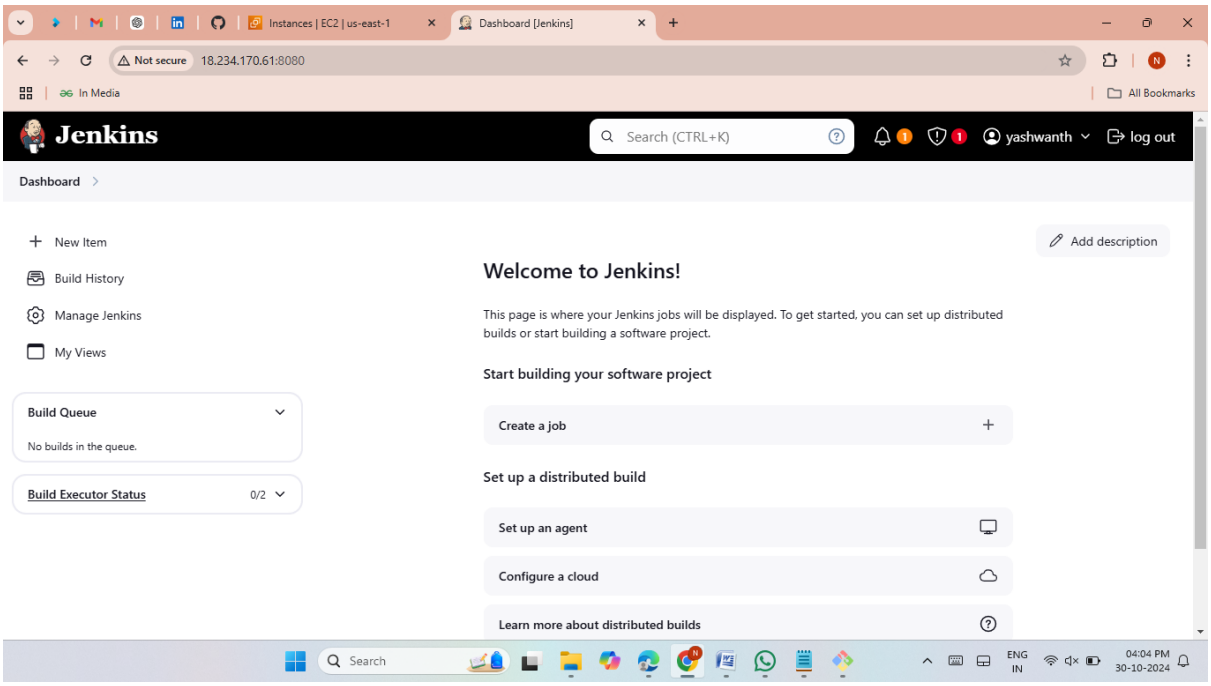
- First launch the EC2 instance with the Jenkins port number(8080) and given required ports.



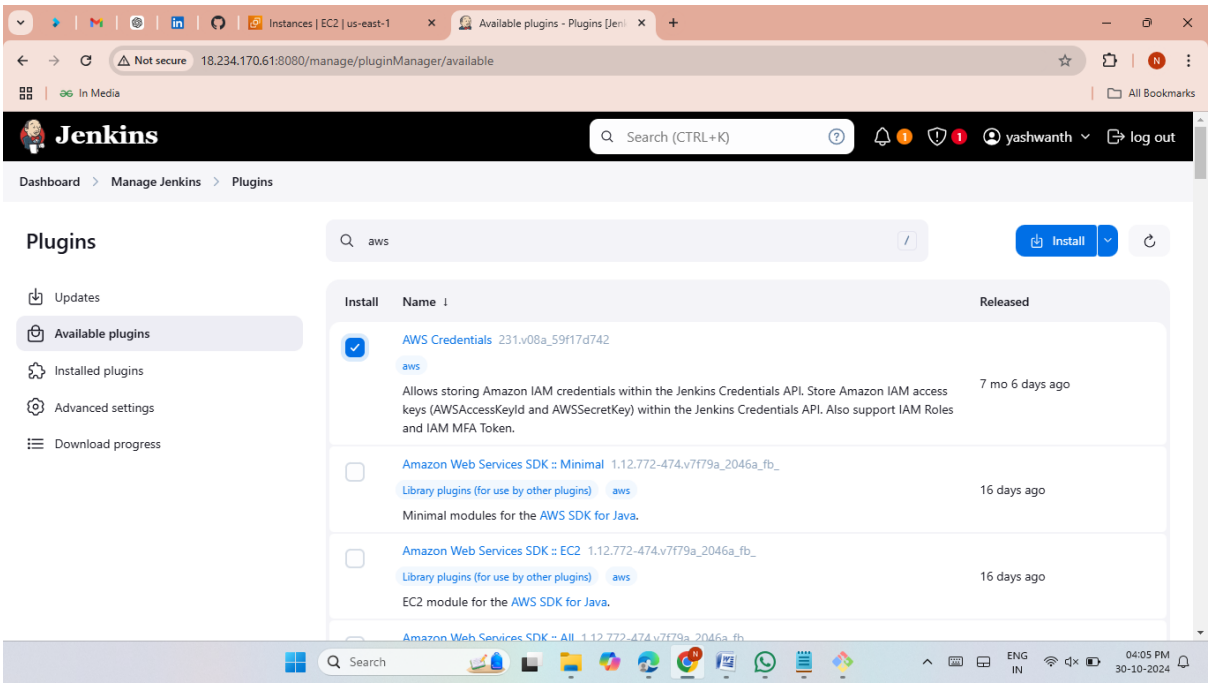

- Now Install git and Jenkins.
- Now start and enable the Jenkins.



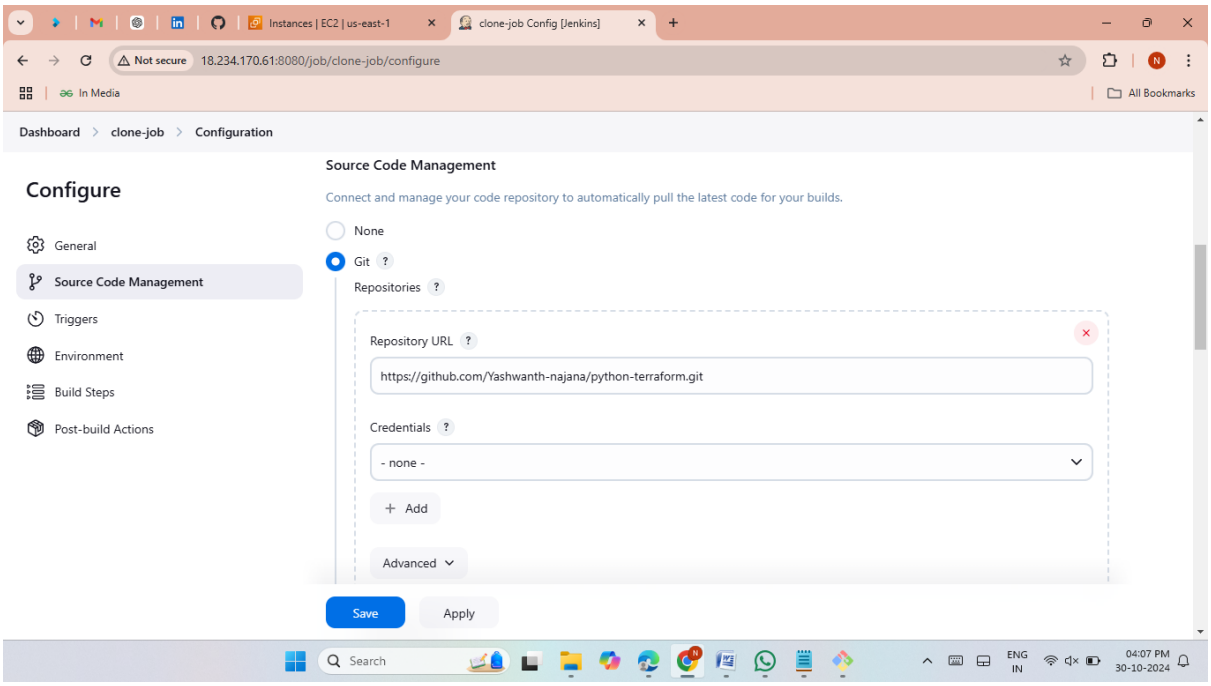
- Here the Jenkins was launched successfully.



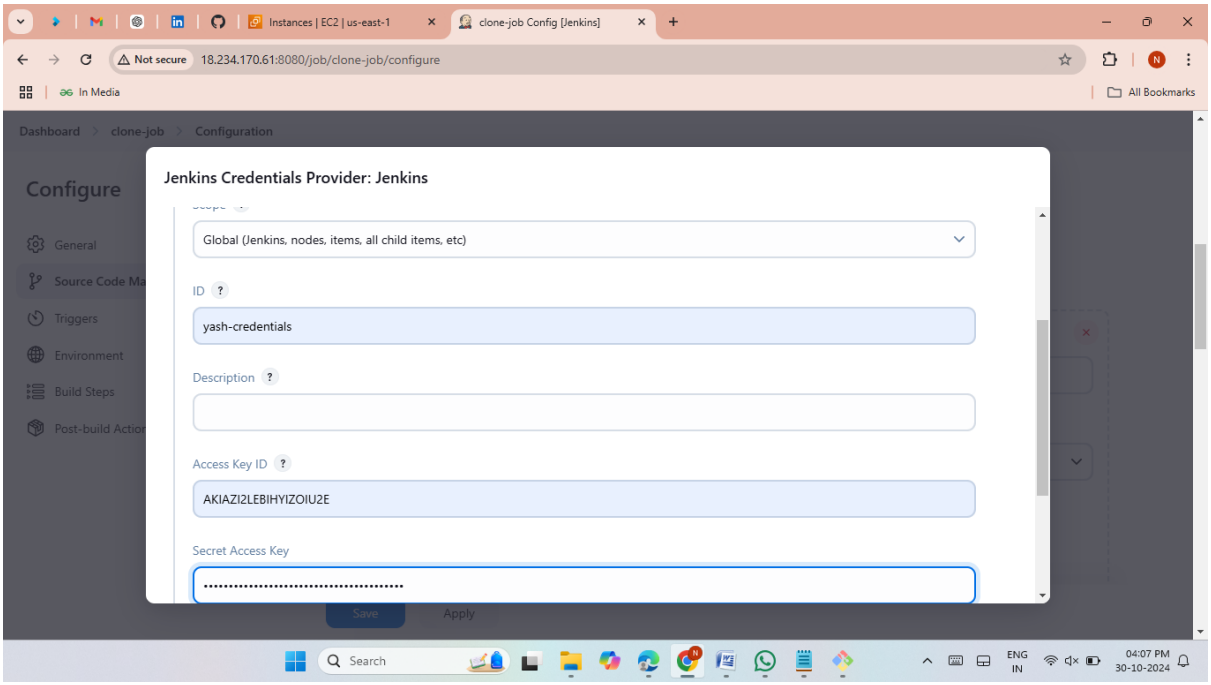

- Now install the AWS credentials plugin.



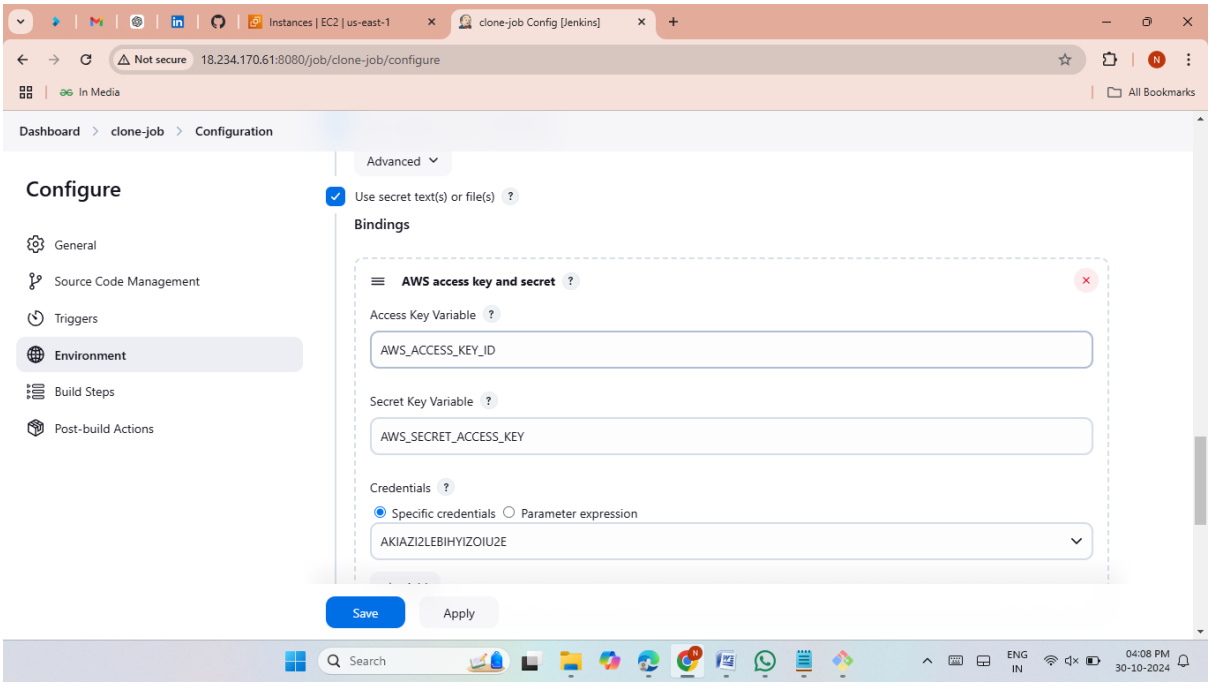
- Now create a job and clone the terraform script form github.




- Then create the credentials.
- Now add the AWS access key and secret key.
- Now apply the changes.



- Now add the AWS access key and secret key.
- Now apply the changes.




- Now install the terraform in execute shell.
- Now add the terraform commands.

**#!/bin/bash**

**sudo yum install -y yum-utils shadow-utils**

**sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo**

**sudo yum -y install terraform**

**sudo chmod + data.sh**

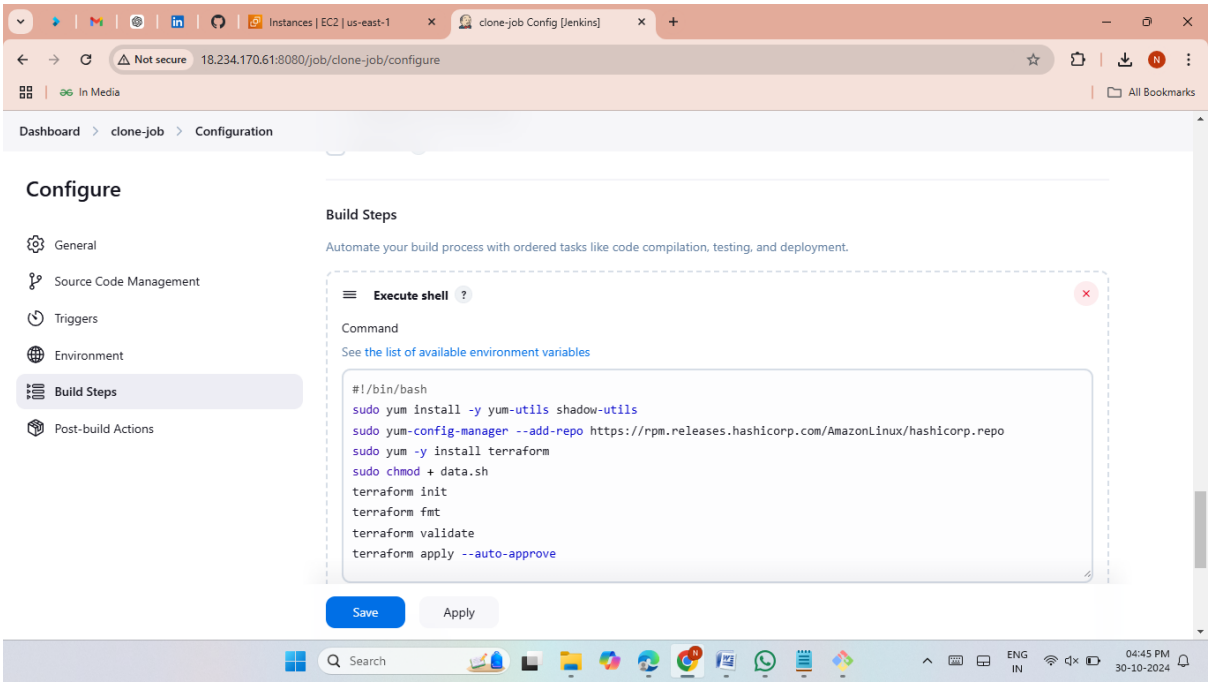
**terraform init**

**terraform fmt**

**terraform validate**

**terraform plan**

**terraform apply --auto-approve**

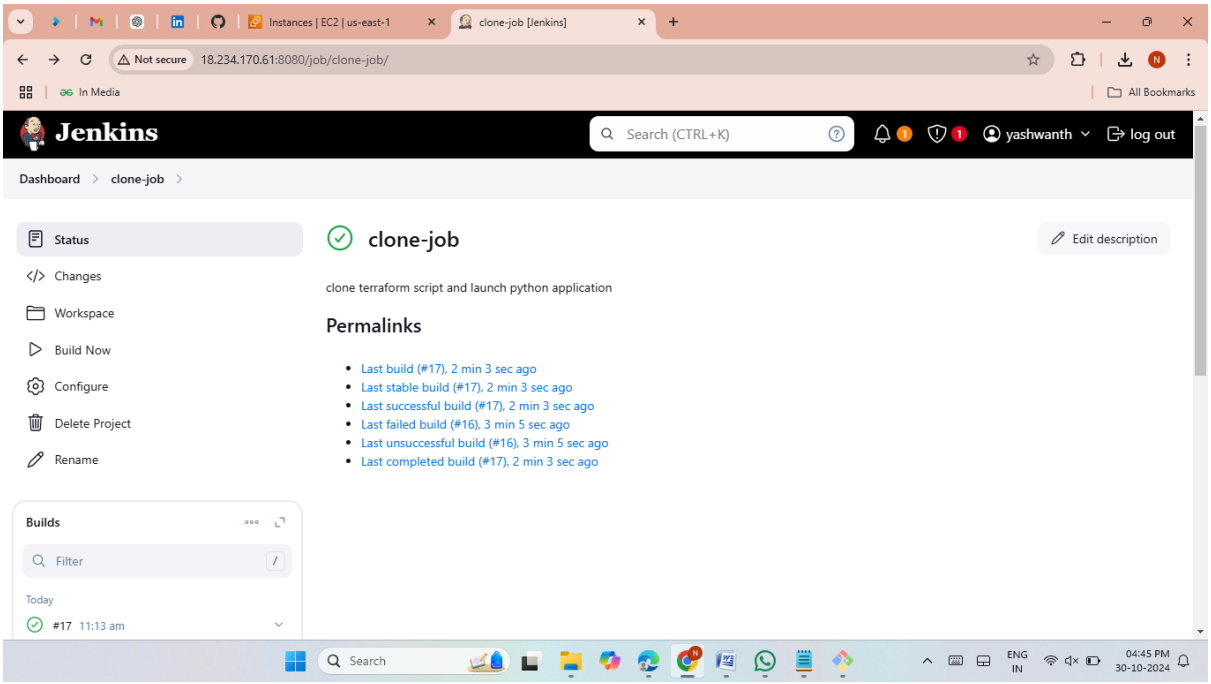



- Now give permissions for Jenkins with the commands

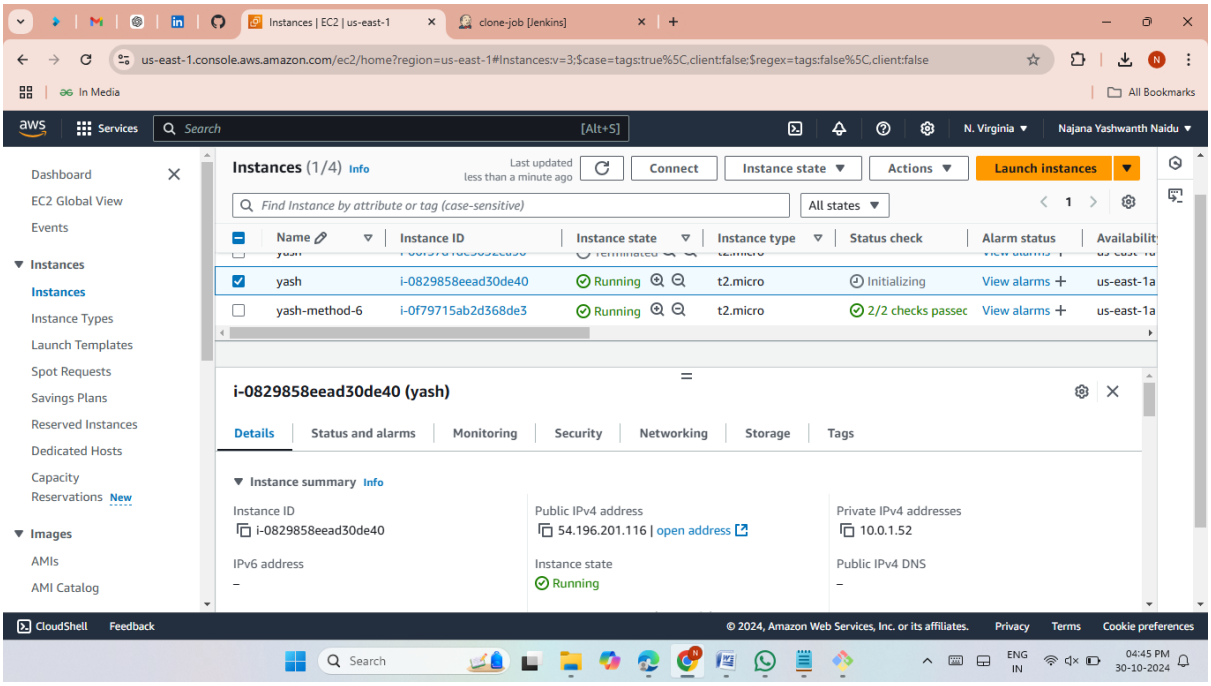
  **sudo visudo**

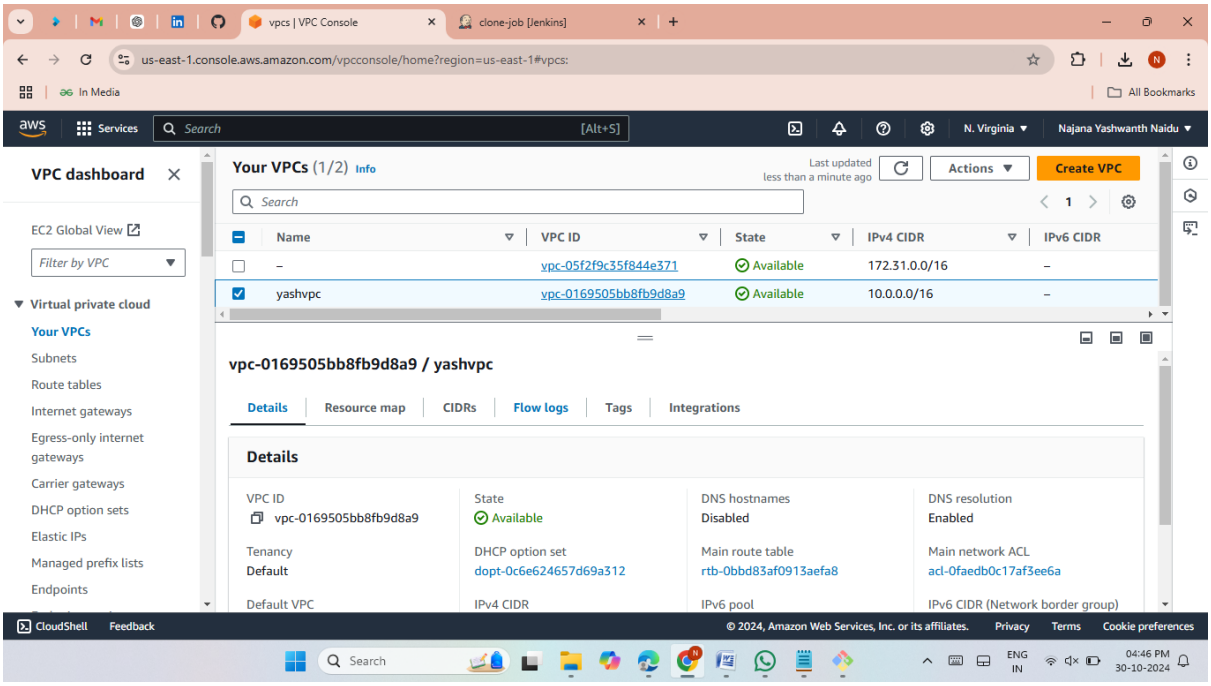
  **jenkins ALL=(ALL) NOPASSWD: ALL**

- Now restart the Jenkins with the command (**sudo systemctl restart Jenkins**).
- Now click on build the job was success.

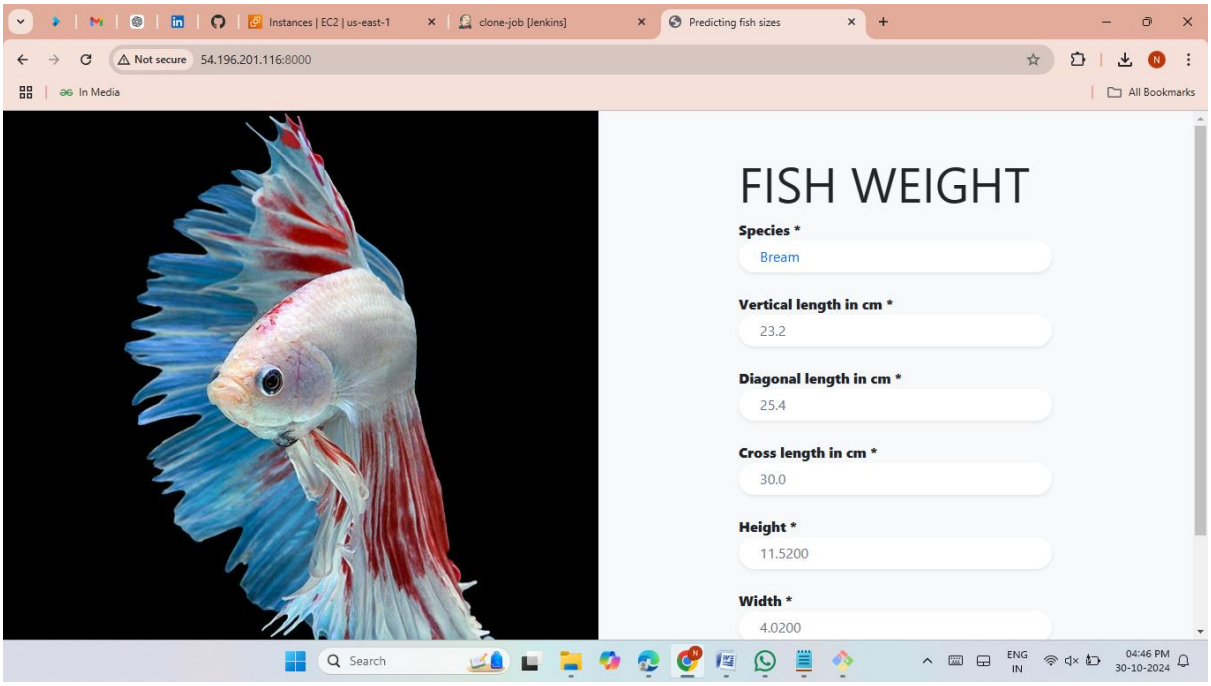


- Here the EC2 instances was automatically launched with help of terraform script.
- Also create the VPC, Subnet, Internet Gateway, Route Table, Security Groups with help of terraform script.



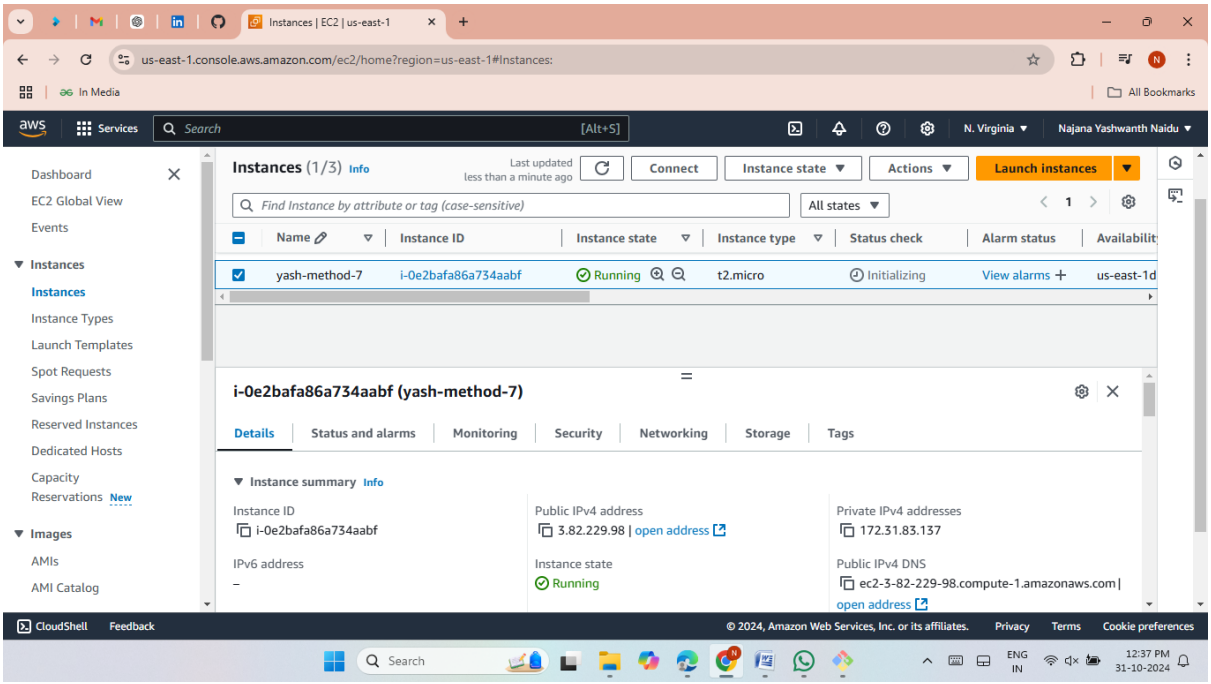
- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish python application is displays.



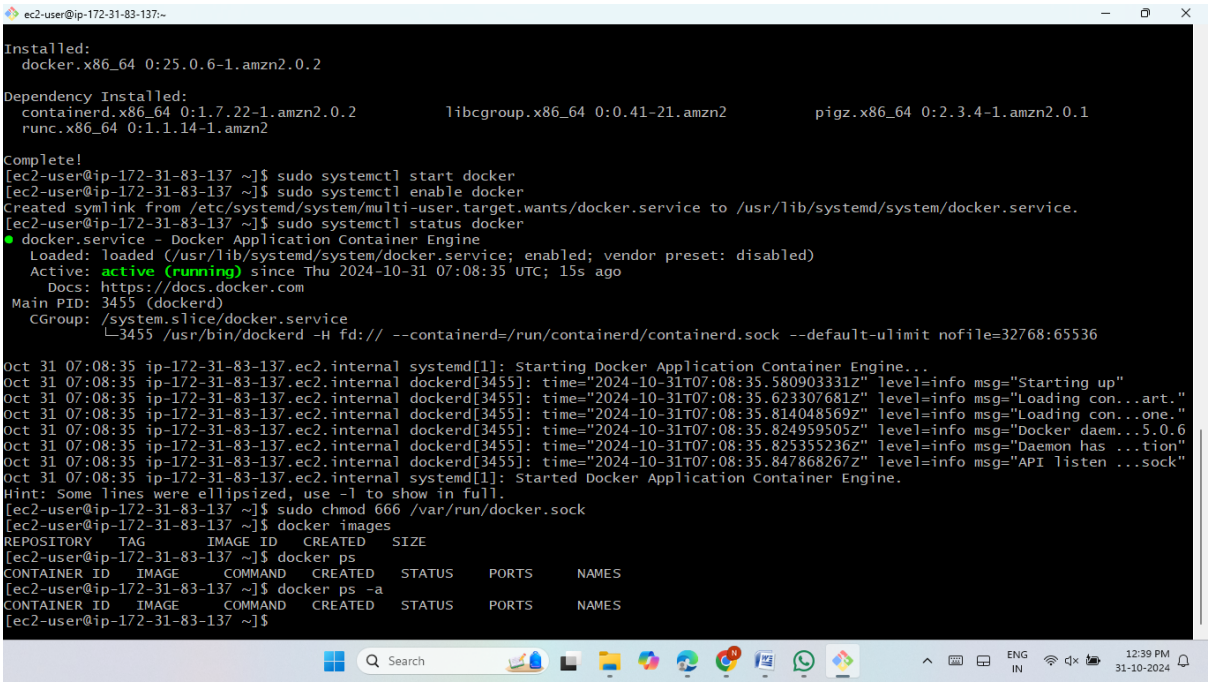

## METHOD-7

**Build and deploy python applications with Docker (Write docker file and create image from docker file and run containers) and push created docker image in docker hub.**

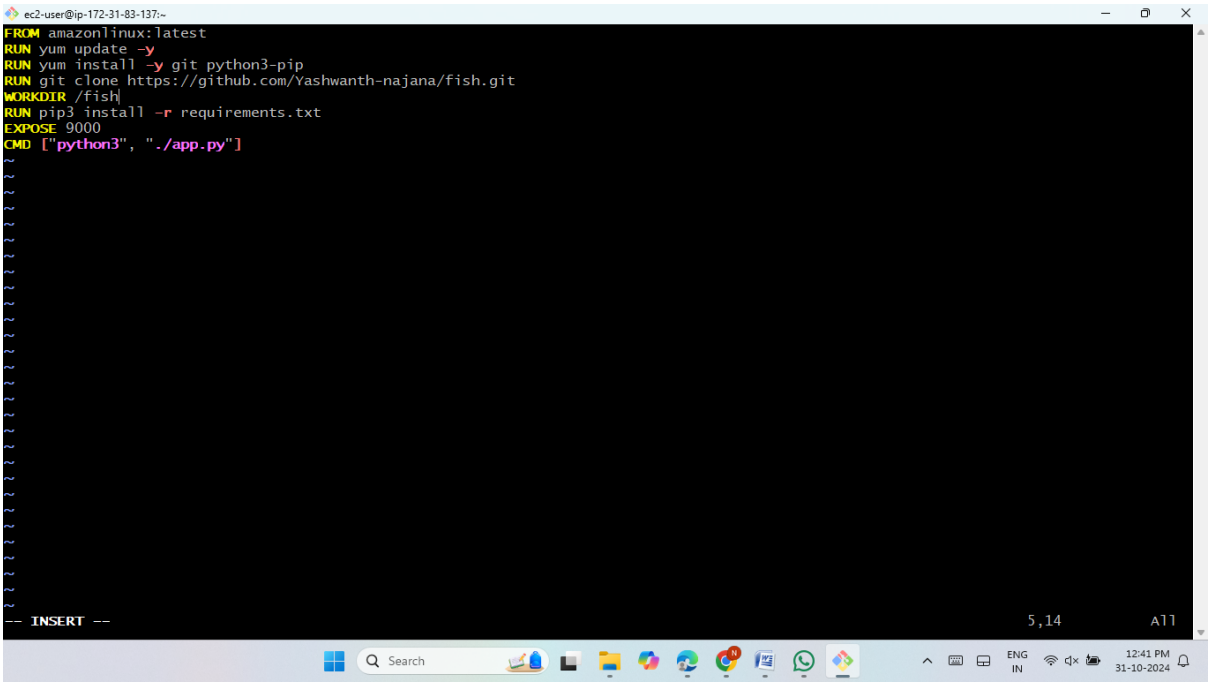
- First launch an EC2 instance.



- Now install docker.
- Now start and enable the docker.
- Give permissions to the docker with the command(**sudo chmod 666 var/run/docker.sock**).
- Now install git.



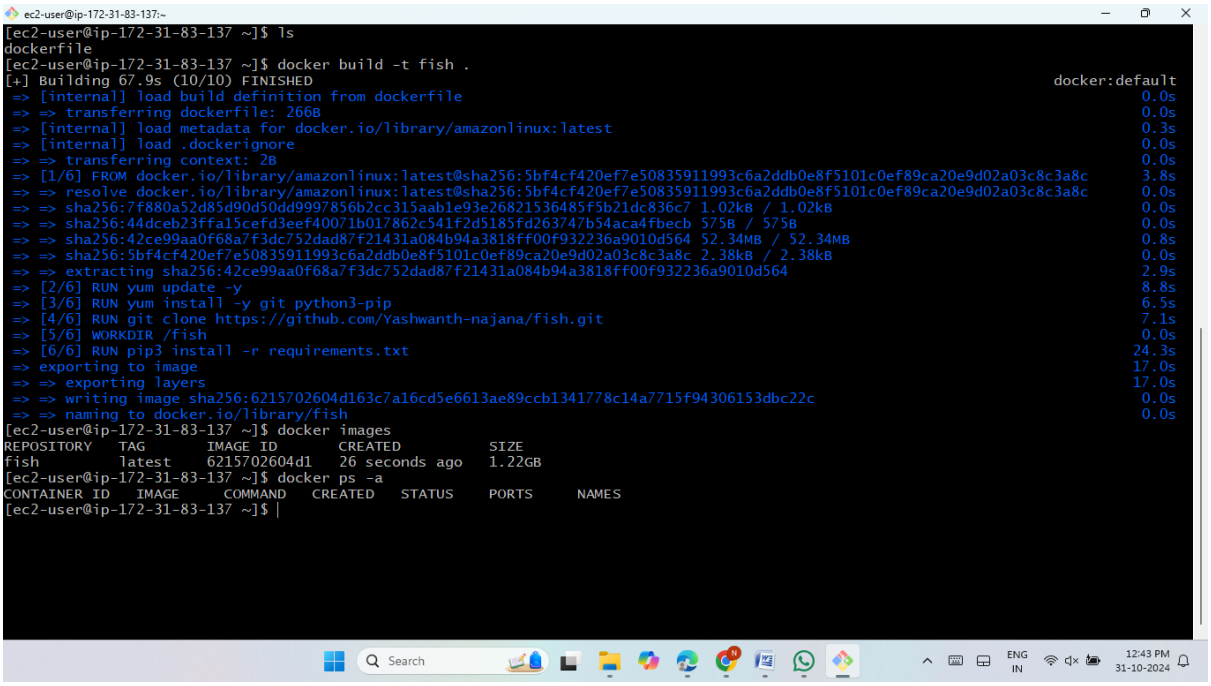

- Now create a docker file and clone the fish python web application.



- Now run the following command.

  **docker build –t <name> .**

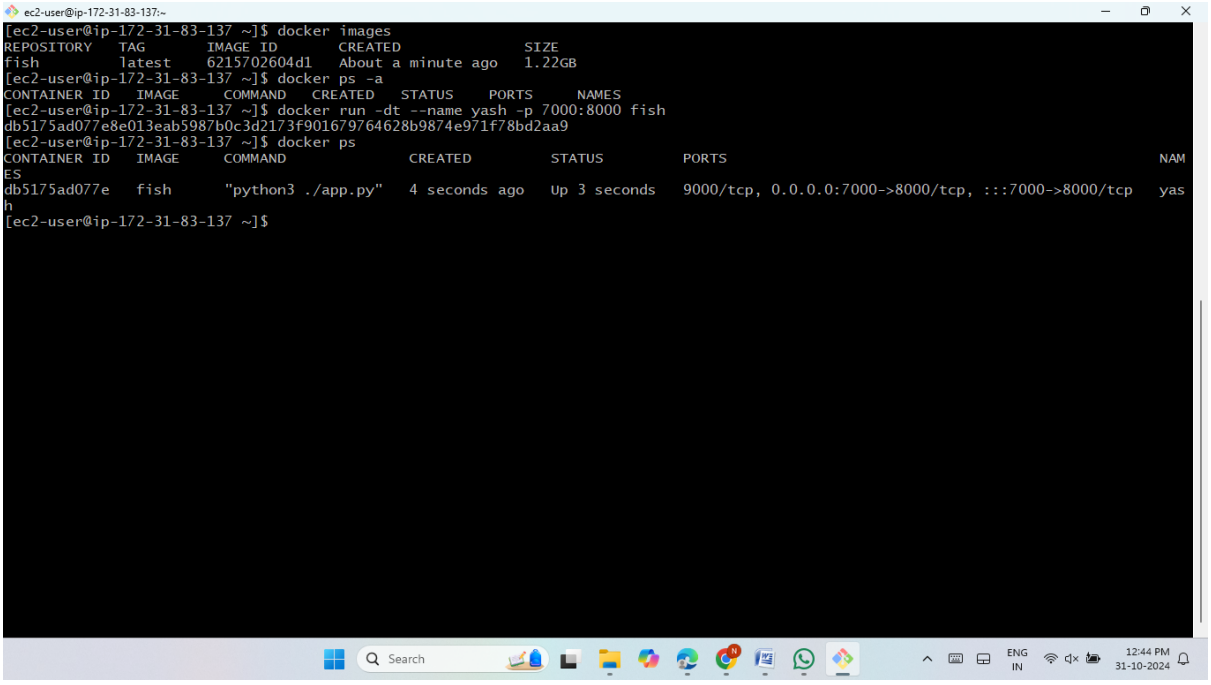
- Here the docker image was created with help of docker file.



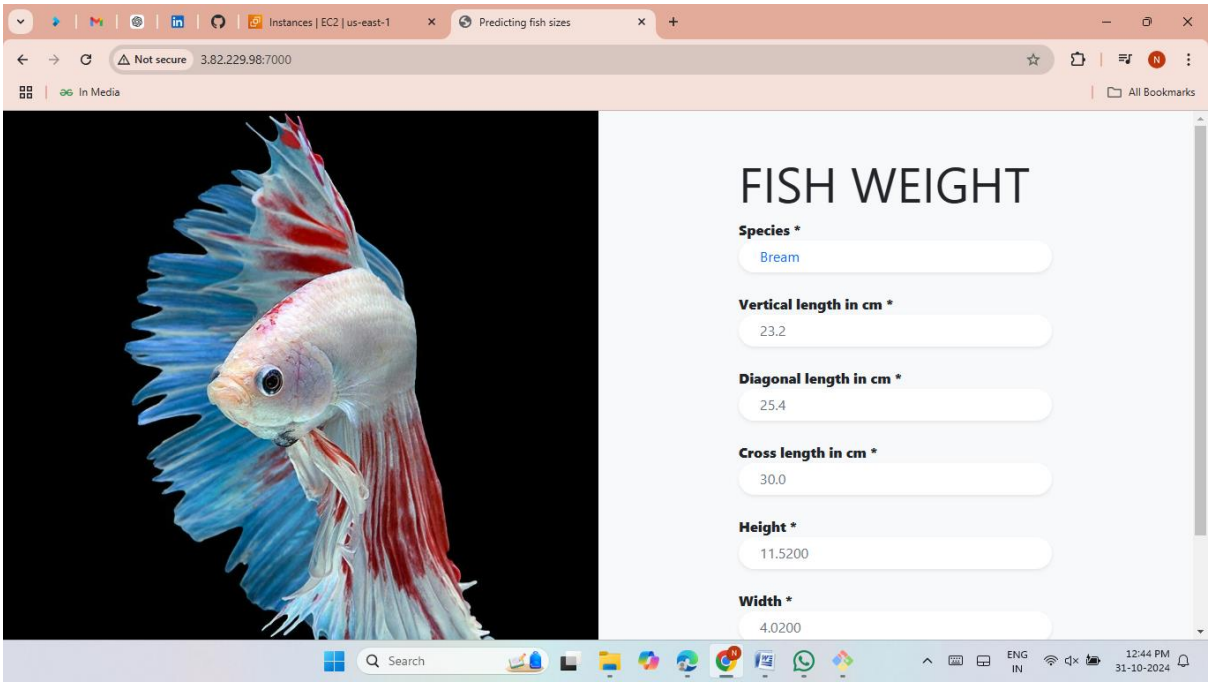

- Now run the command.

  **Docker run –dt --name <name> -p <port>:8000 <image name>**

- Here the docker container was created successfully.



- Now go to EC2 instance and copy public IP and paste it on Google browser along with given port and check the fish python application is displays.

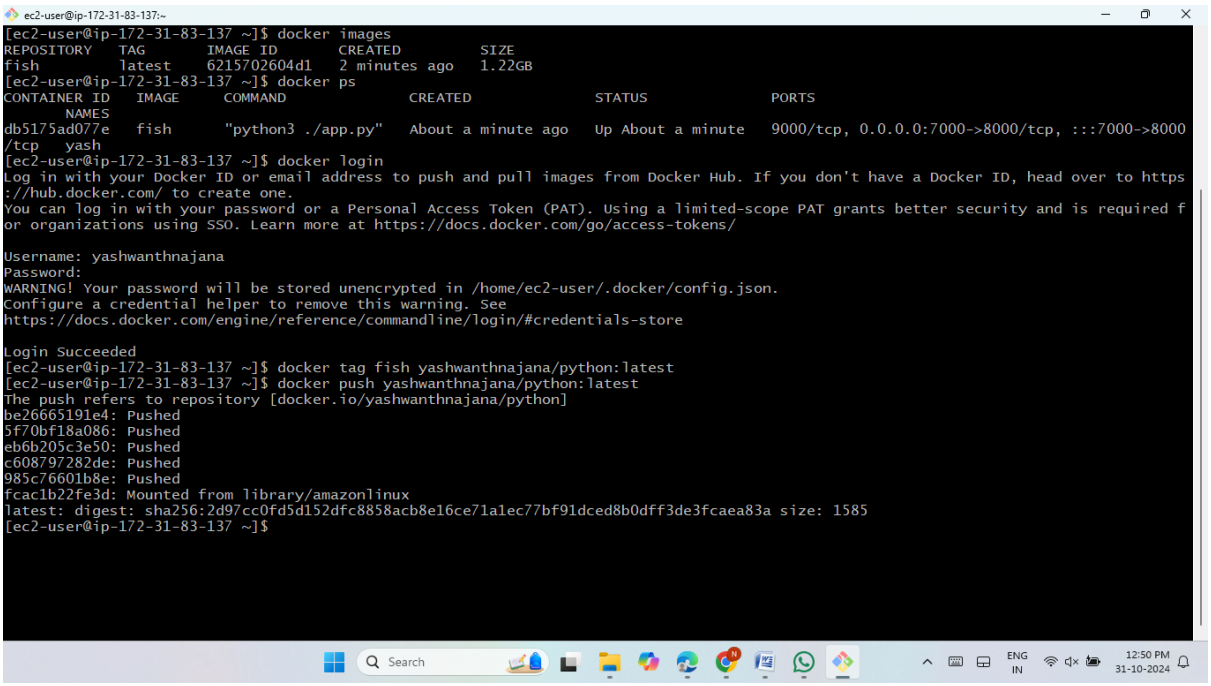



- Now run the following commands to push the docker images into DockerHub platform.

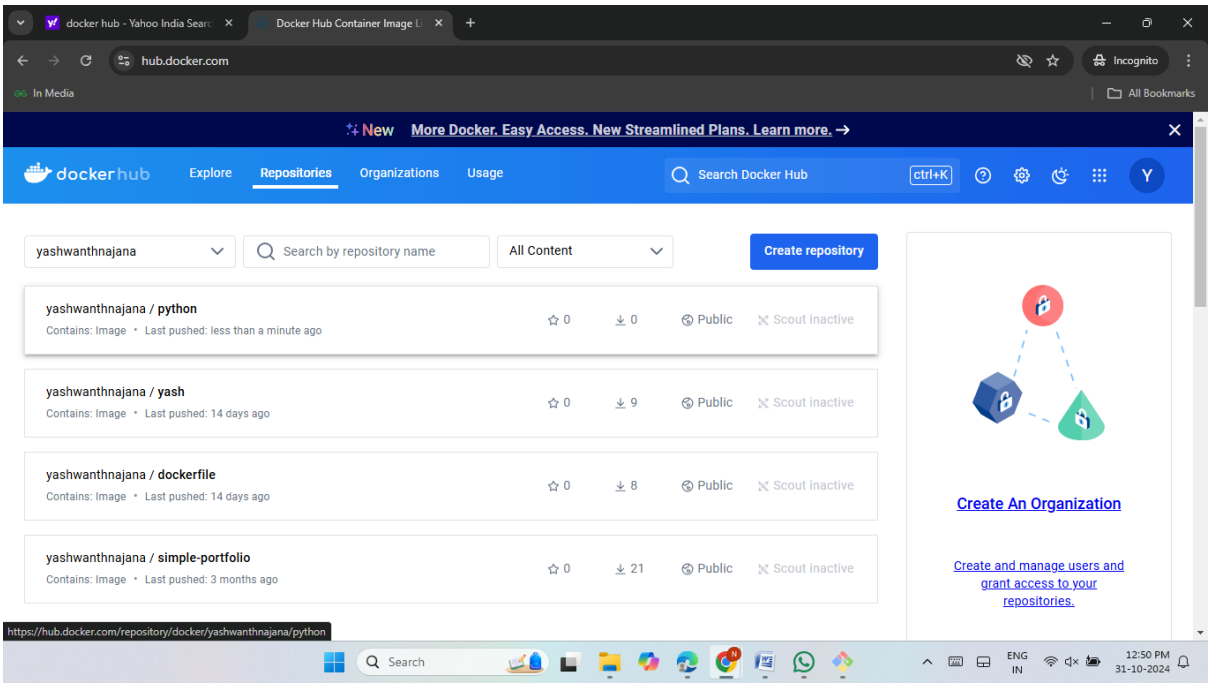
  **docker login (give the DockerHub user and password).**

  **docker tag <image name> <username>/<repo name>:<tag>**

  **docker push <username>/<repo name>:<tag>**



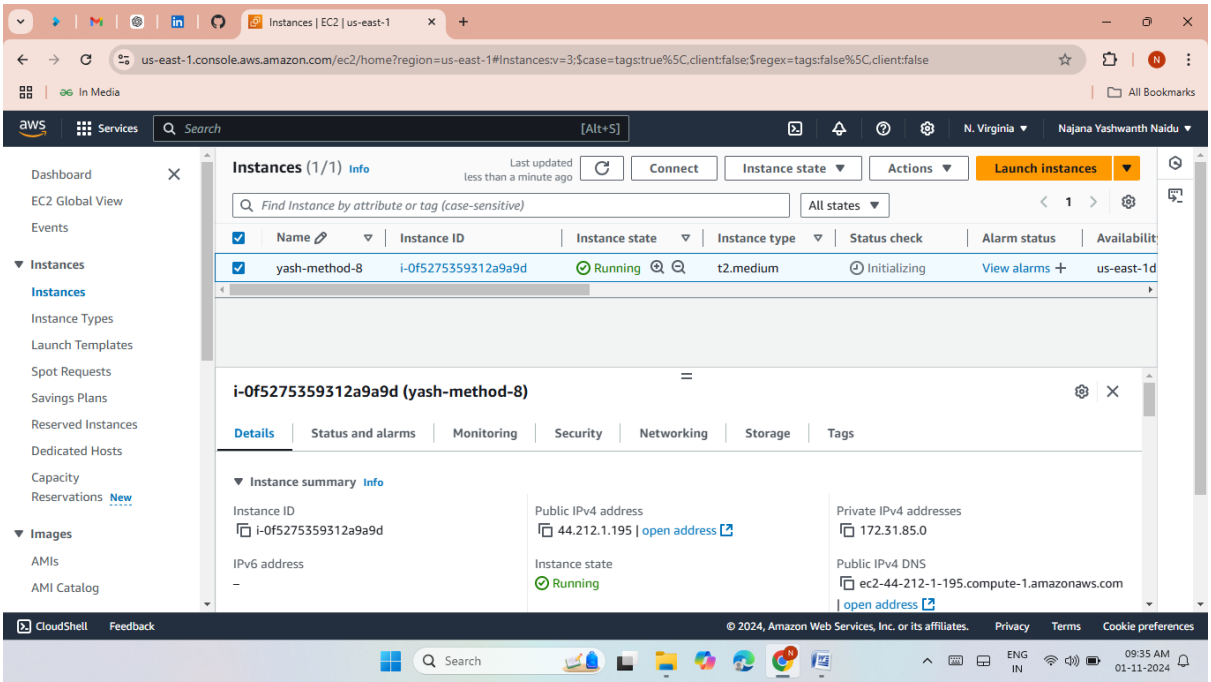
➢ Here the docker image is pushed successfully (python).



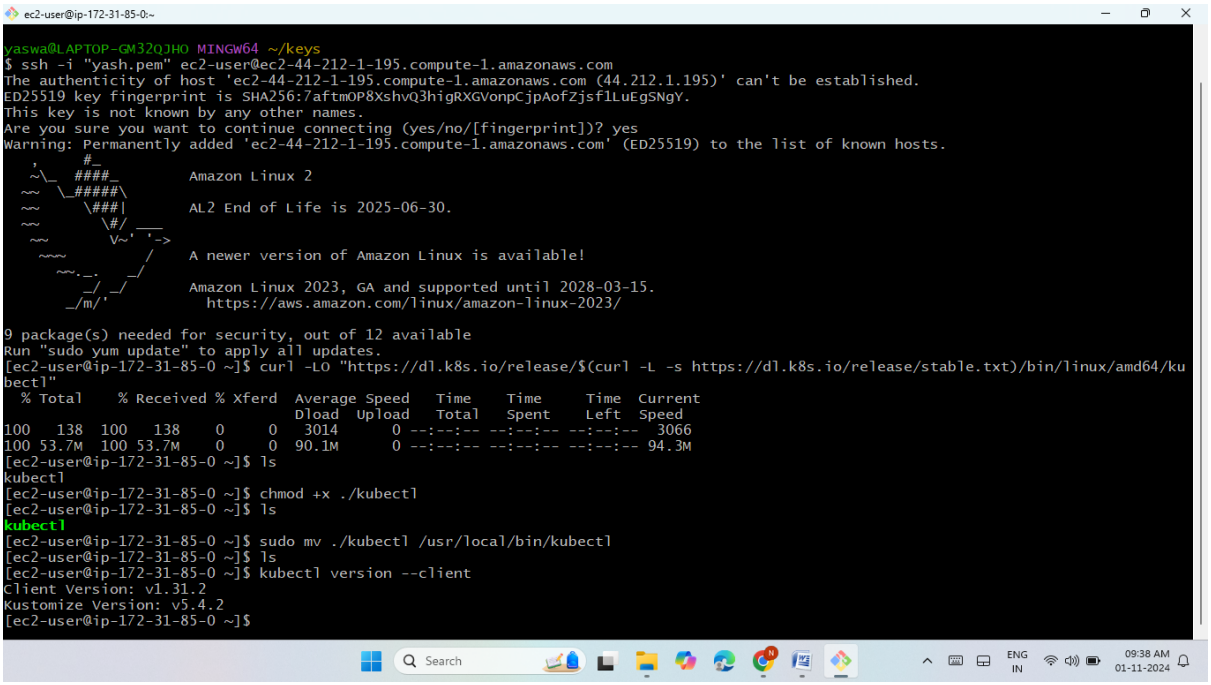

# METHOD -8

## Build and deploy python applications with Docker and k8's (EKS and KOPS) (use Declarative manifest method along with docker image)

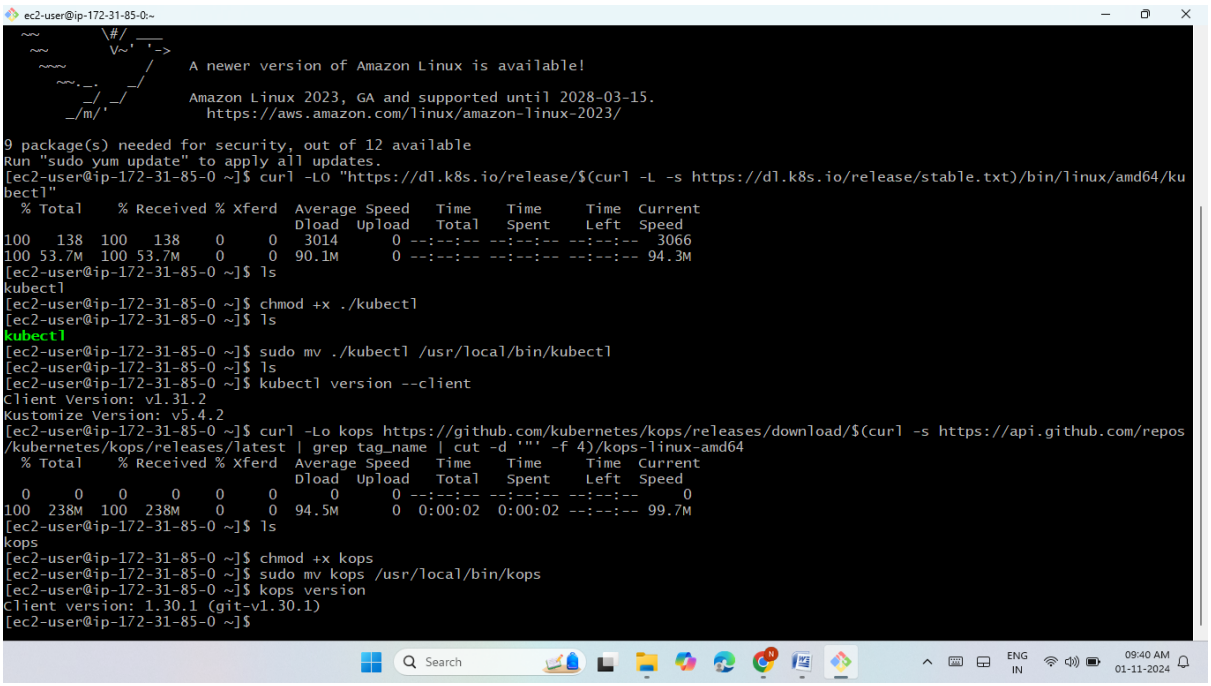
➢ Now launch an EC2 instance for hosting the fish python application in kubernetes.



- Now install aws cli (in linux by default aws cli is there. If we want updated version, we can download).
- Create role with all admin permissions and attach it to the EC2 instance.
- Now first Install kubectl package from Google.
- Then install kubectl.



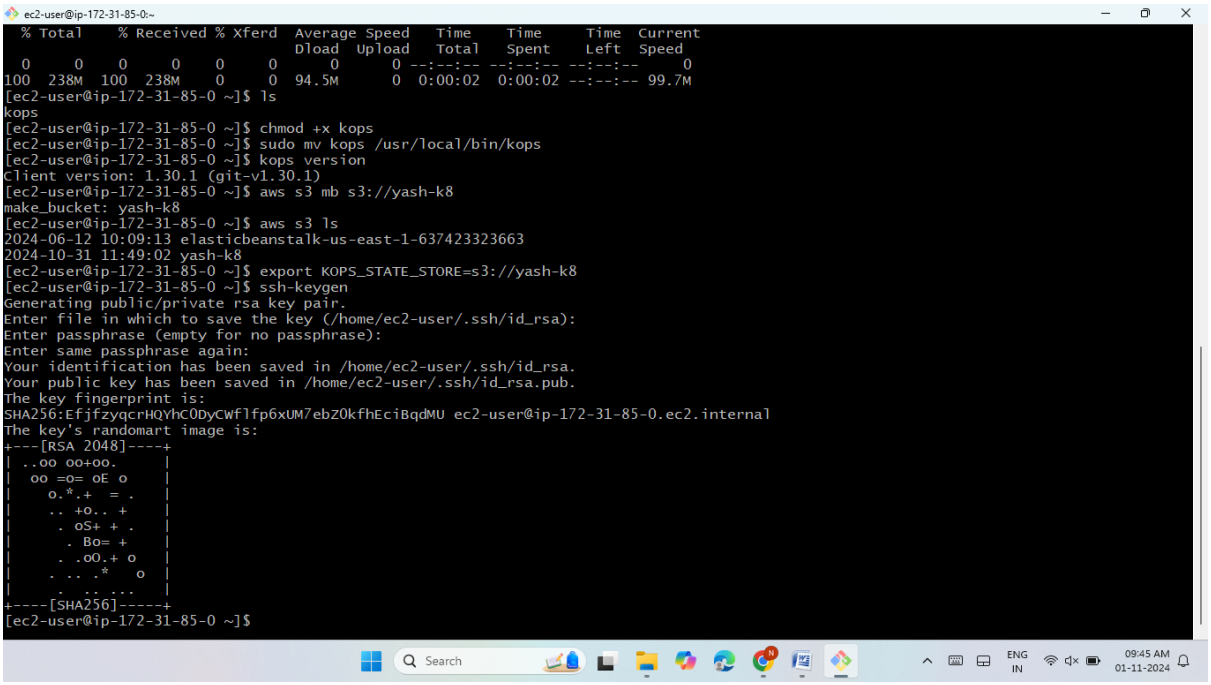

- Now install kops from Google.

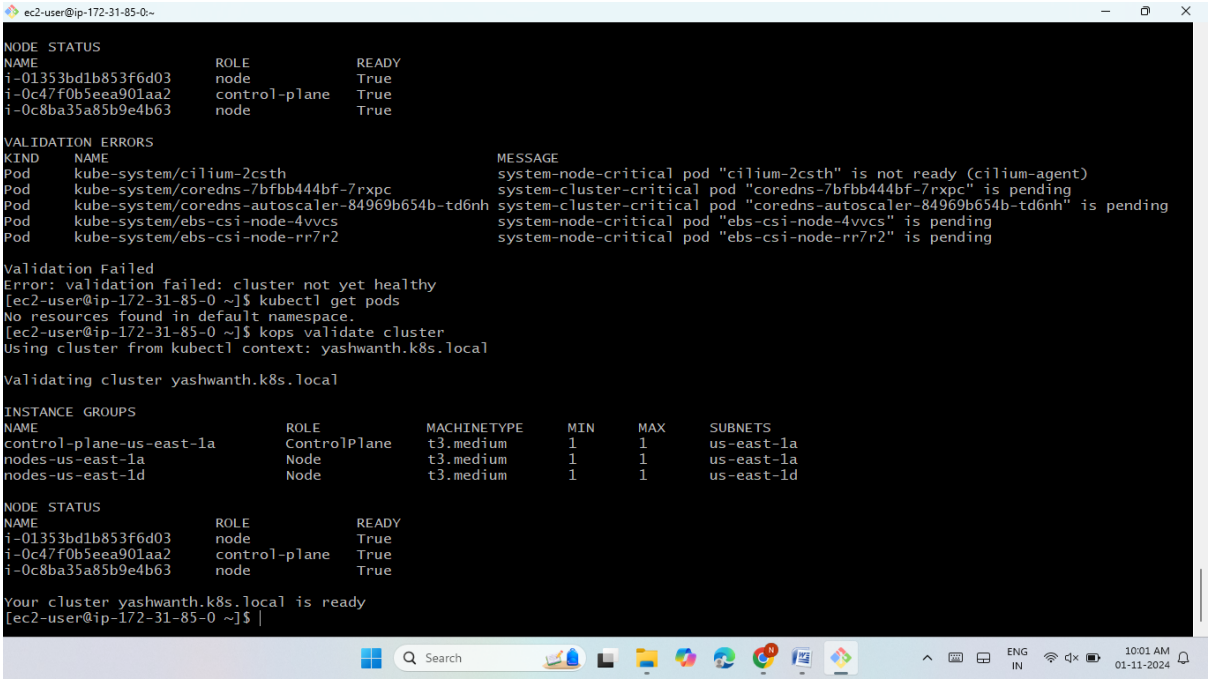


- Now run the following commands.

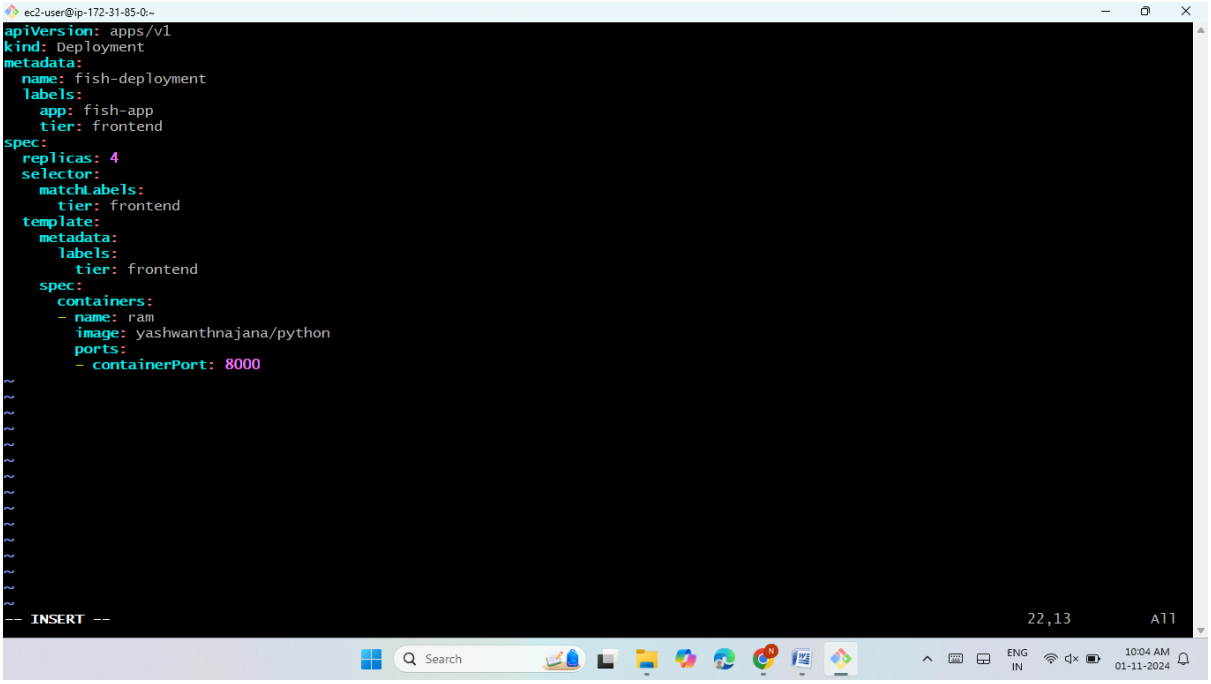
  **aws s3 mb s3://yash-k8 (to create s3 bucket to store the data).**
  **aws s3 ls (to show the s3).**
  **export KOPS_STATE_STORE=s3://yash-k8 (attach s3 bucket to kops for backup).**
  **ssh-keygen (now enter).**
  **kops create cluster --name .k8s.local --state s3:// --zones useast-1b,us-east-1a --node-count 2 --yes** (to create Cluster and it create VPC, Subnet, IG, Route, LoadBalancer, Target Group ,Auto Scalling Group, Instances).

  **kops validate cluster** (to see the details od cluster).

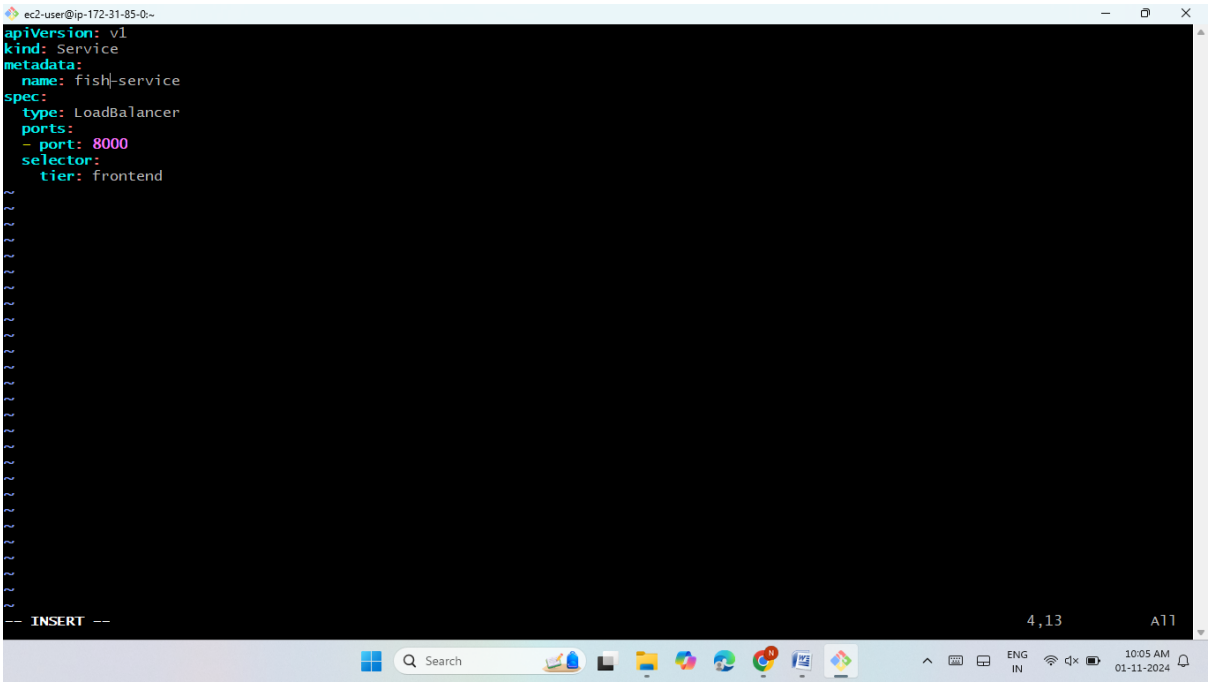


- Now create a deployment.yaml file for run the dockerhub image by using declarative manifest method.




- Now create a service.yaml file for run the dockerhub image by using declarative manifest method.

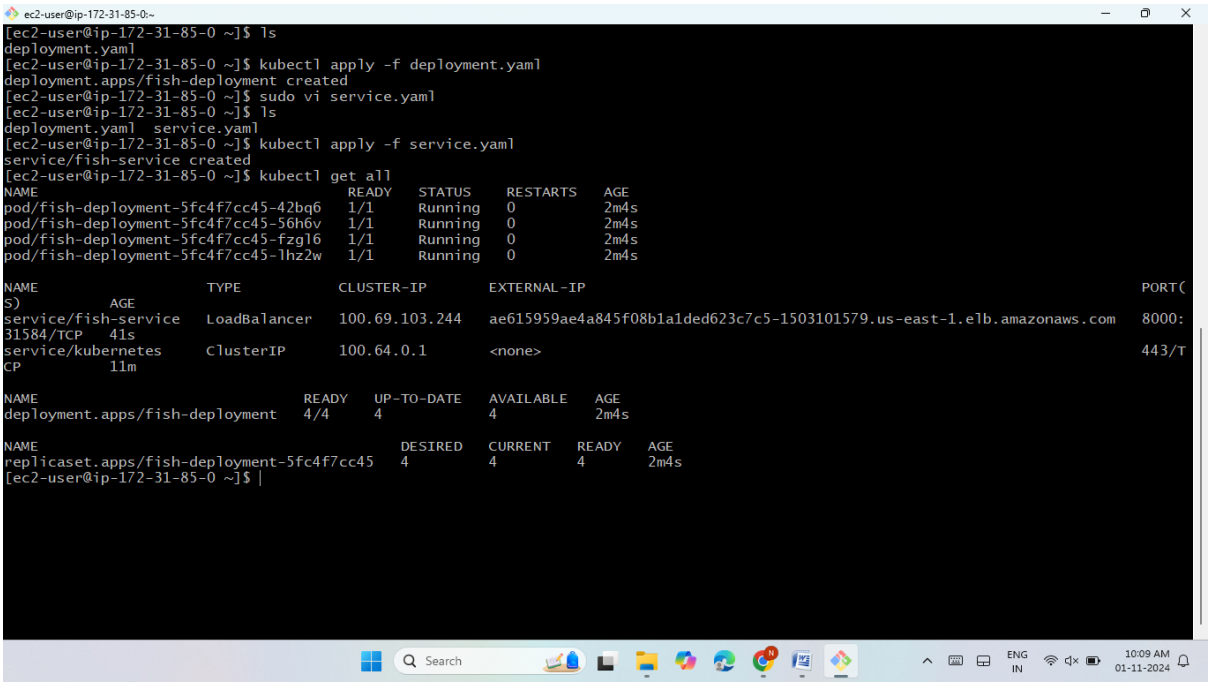


- Now run the following command.

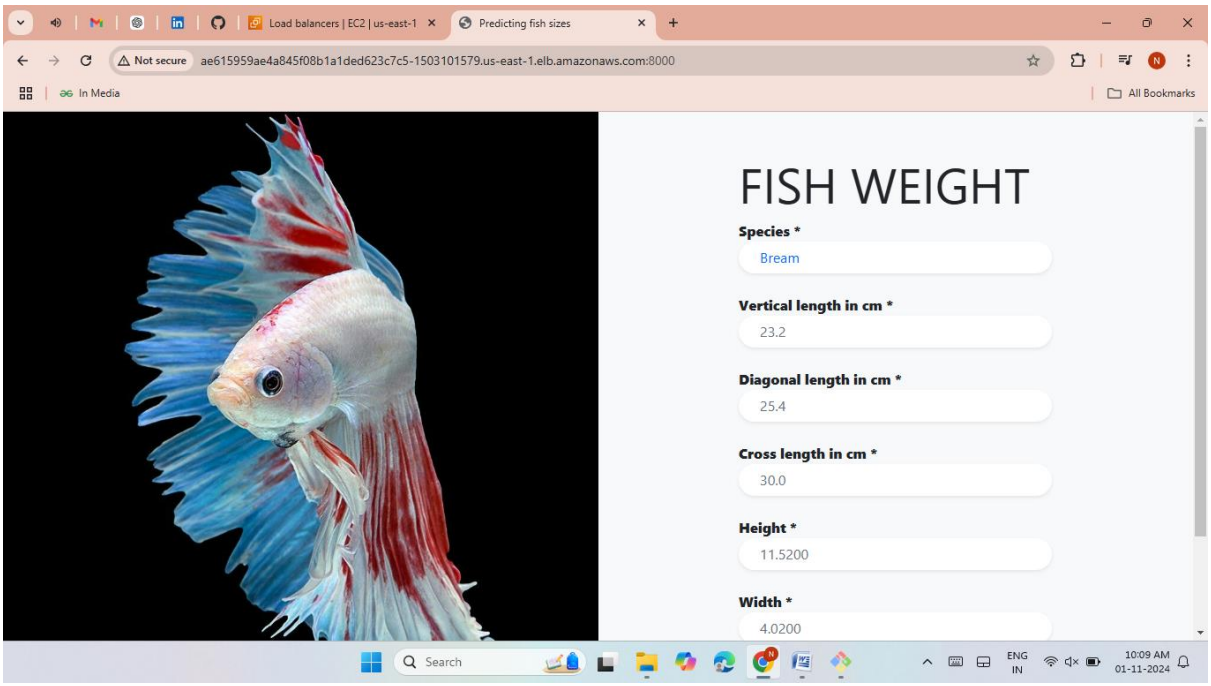
  **kubectl apply –f deployment.yaml**

  **kubectl apply –f service.yaml**

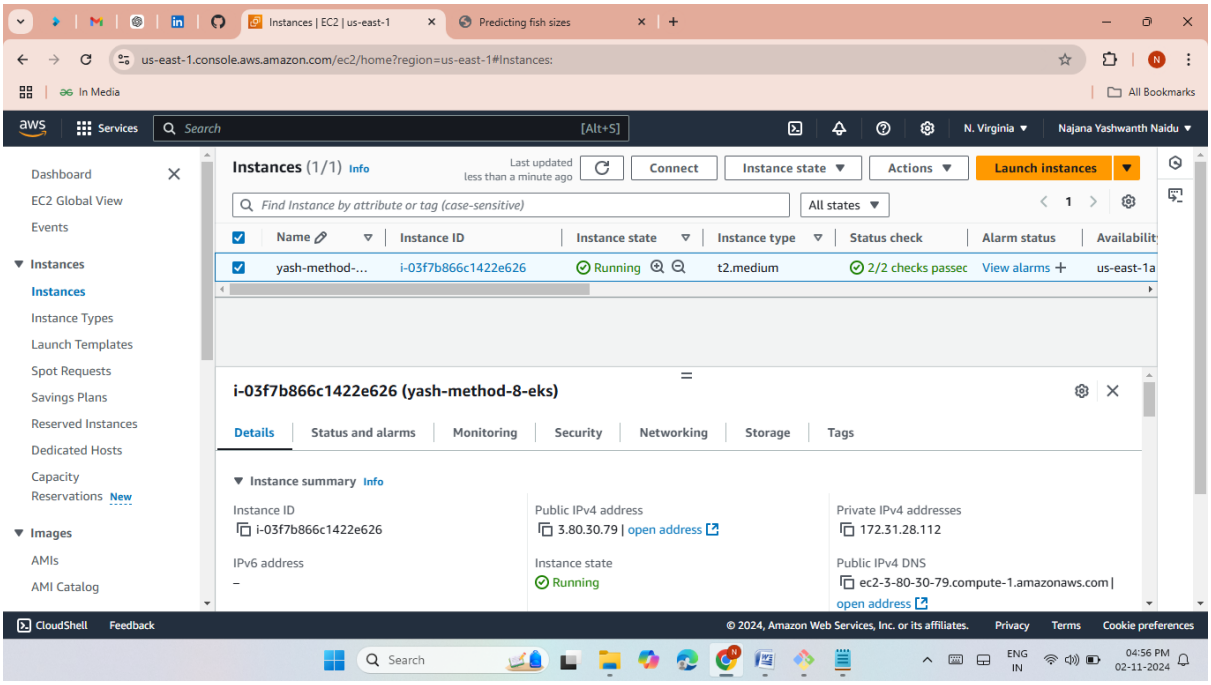
  **kubectl get all**



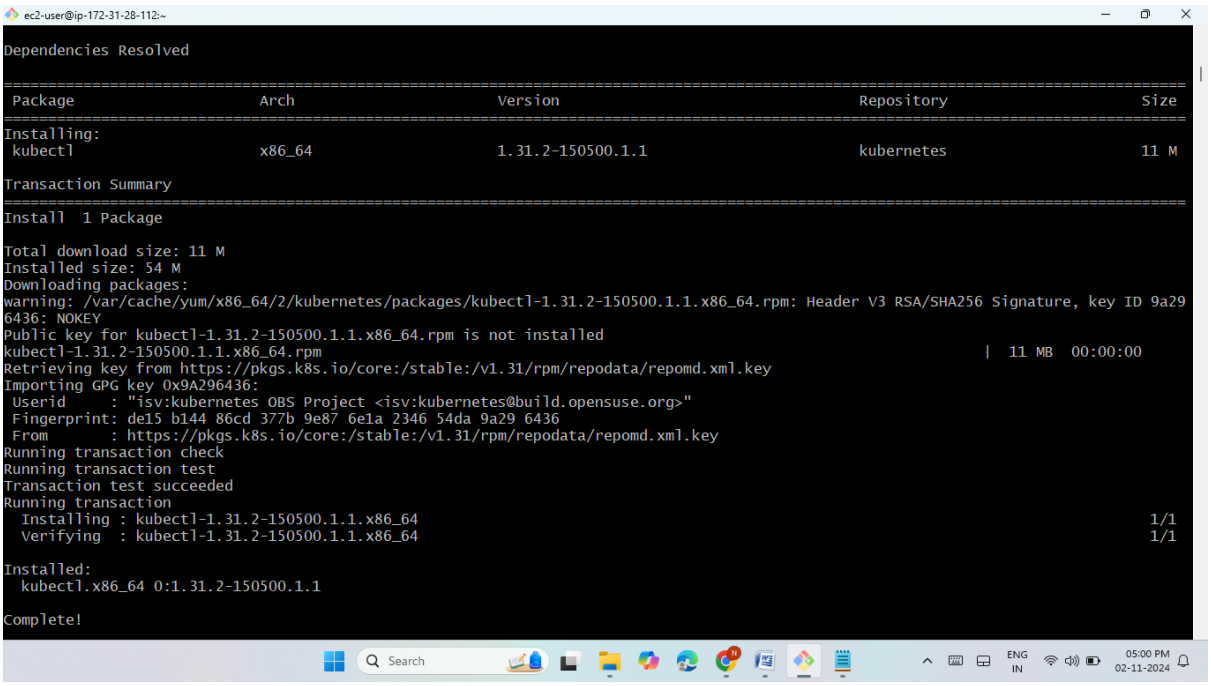

- Now with help of Load Balancer DNS link along with given port the fish python we application was hosted successfully.

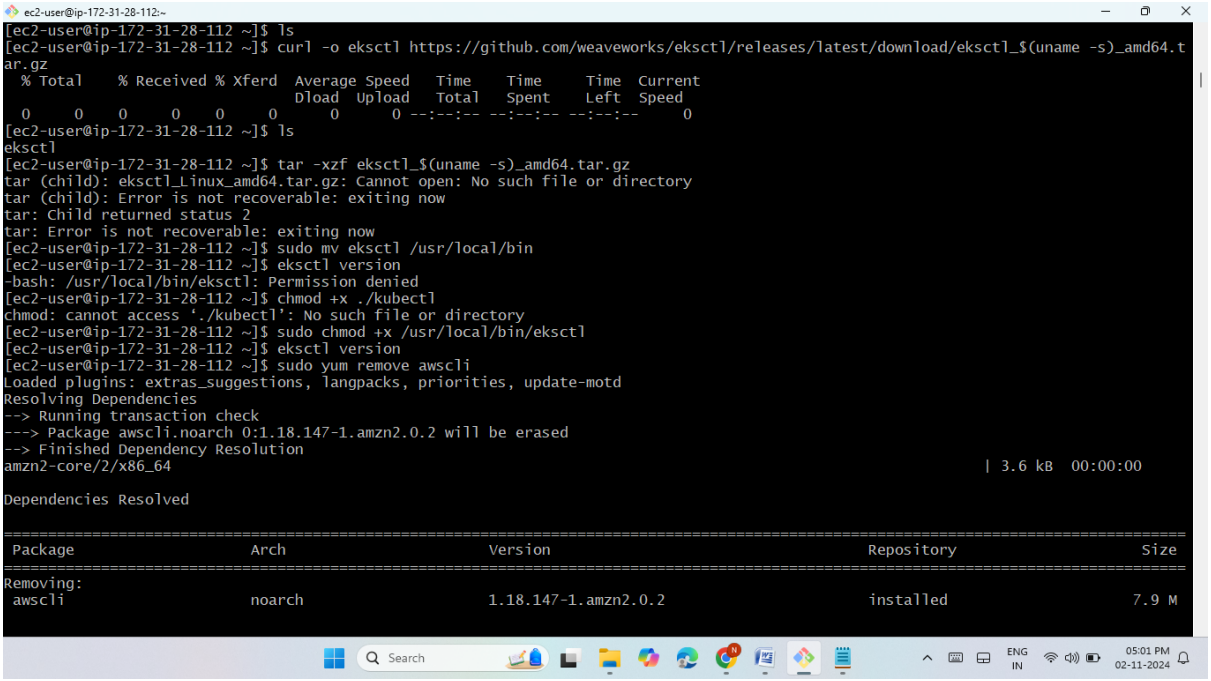


- Now launch an EC2 instance for hosting the fish python application in kubernetes.




- Now install aws cli (in linux by default aws cli is there. If we want updated version, we can download).
- Now first Install kubectl package from Google.
- Then install kubectl.
- Then Install eksctl (CLI tool for creating and managing EKS clusters).






- Now run the following command.

  **aws configure**

  **Aws access keys**

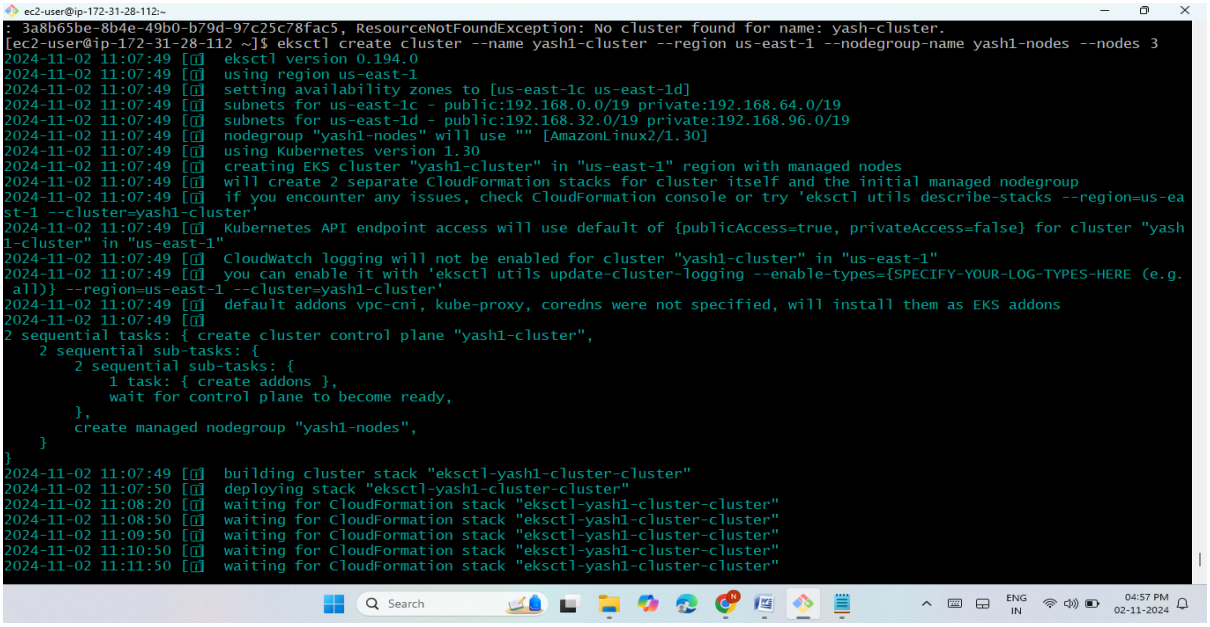
  **Aws secret keys**

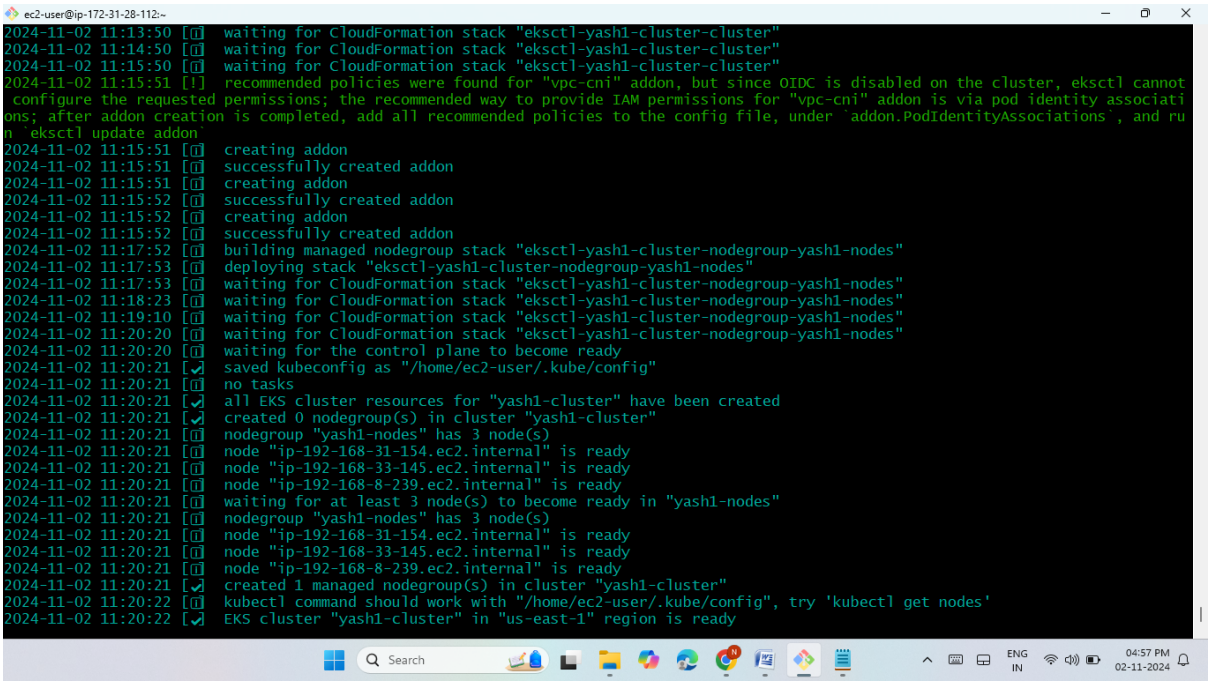
  **Region**

  **Output format**

- Now create cluster with the command.

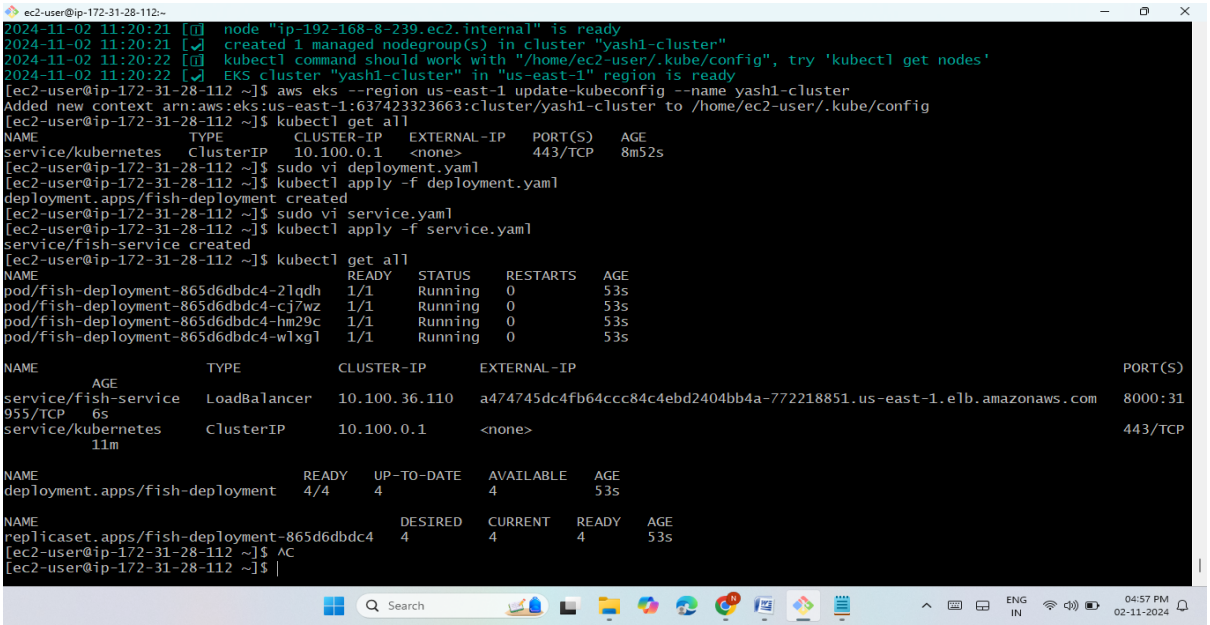
  **eksctl create cluster --name <cluster name> --region <region> --nodegroup-name <nodegroup name> --nodes <count>**

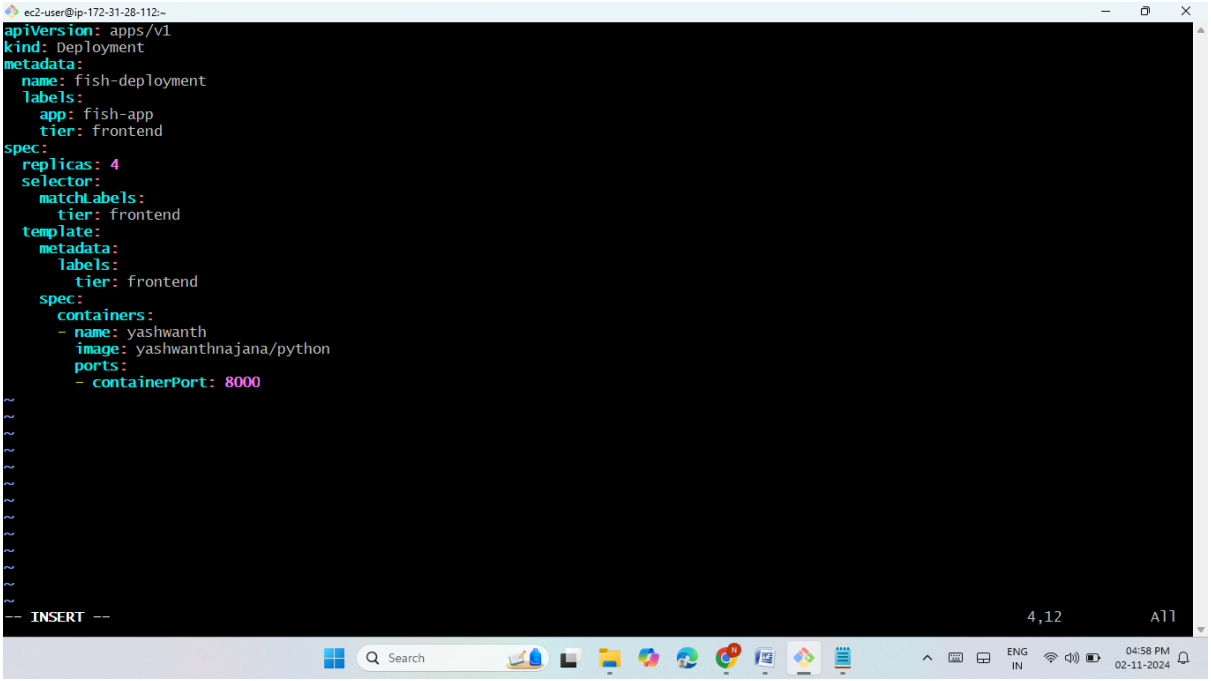


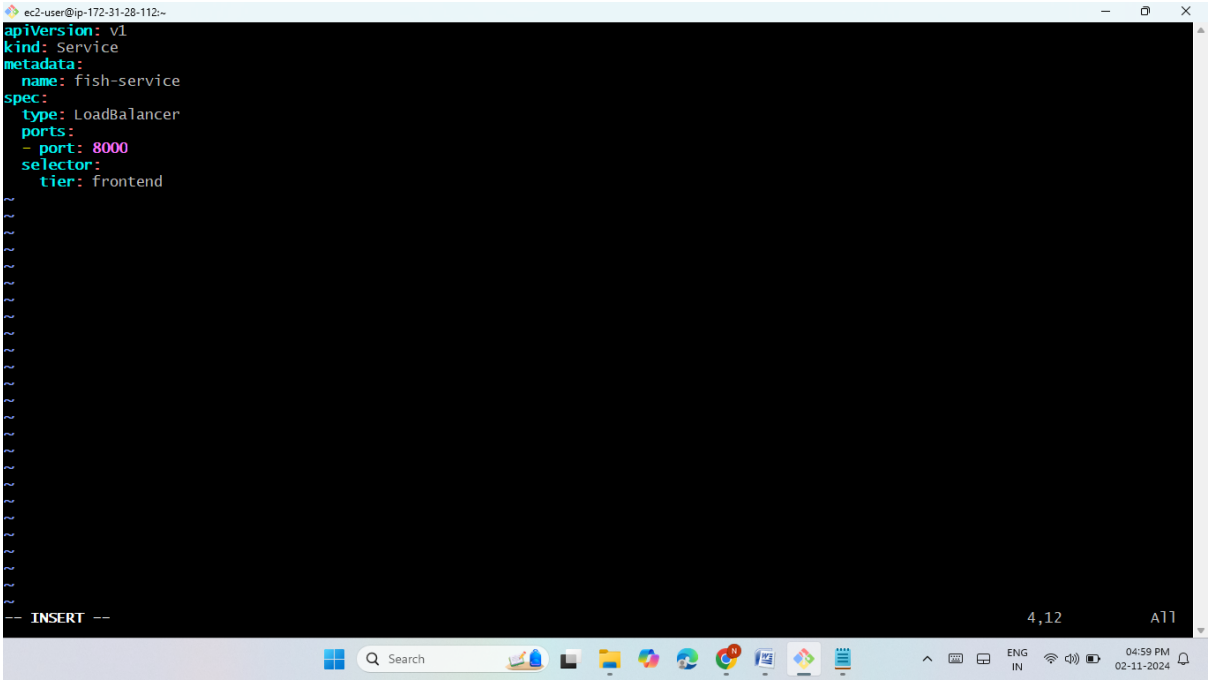



- After the cluster is created, configure kubectl to use the new cluster with the command **aws eks --region <region> update-kubeconfig --name <cluster name>**
- Now create a deployment.yaml file for run the dockerhub image by using declarative manifest method.
- Now create a service.yaml file for run the dockerhub image by using declarative manifest method.
- Now run the following command.

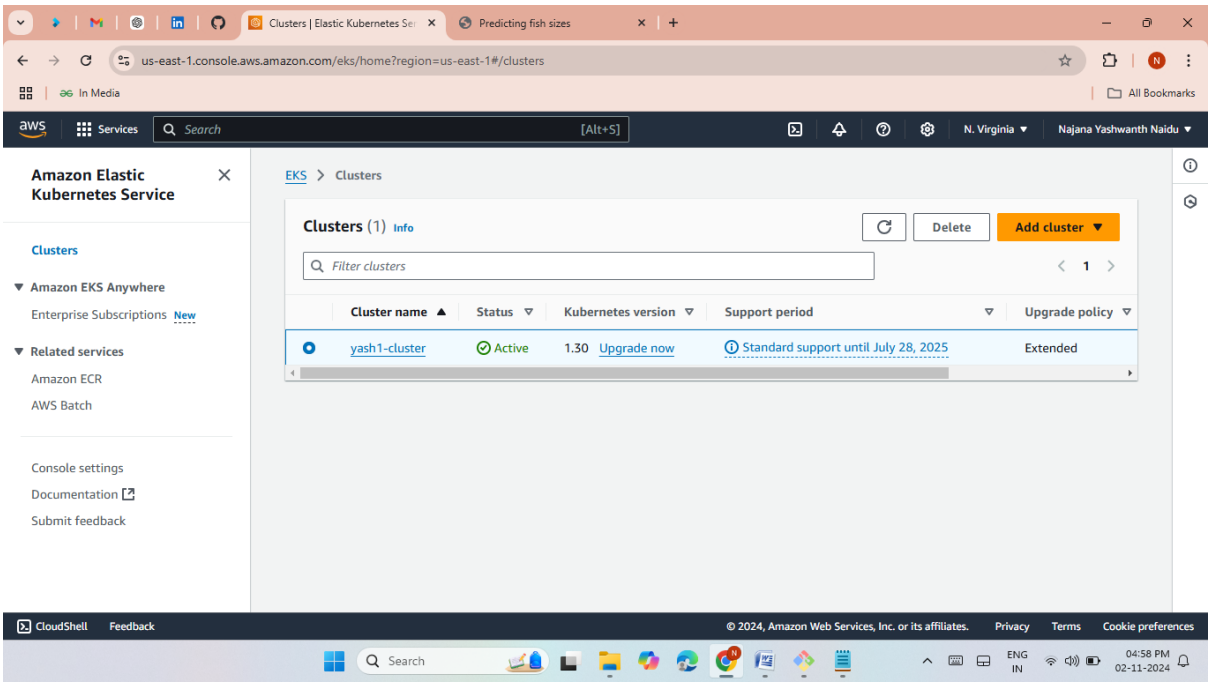
  **kubectl apply –f deployment.yaml**
  **kubectl apply –f service.yaml**
  **kubectl get all**



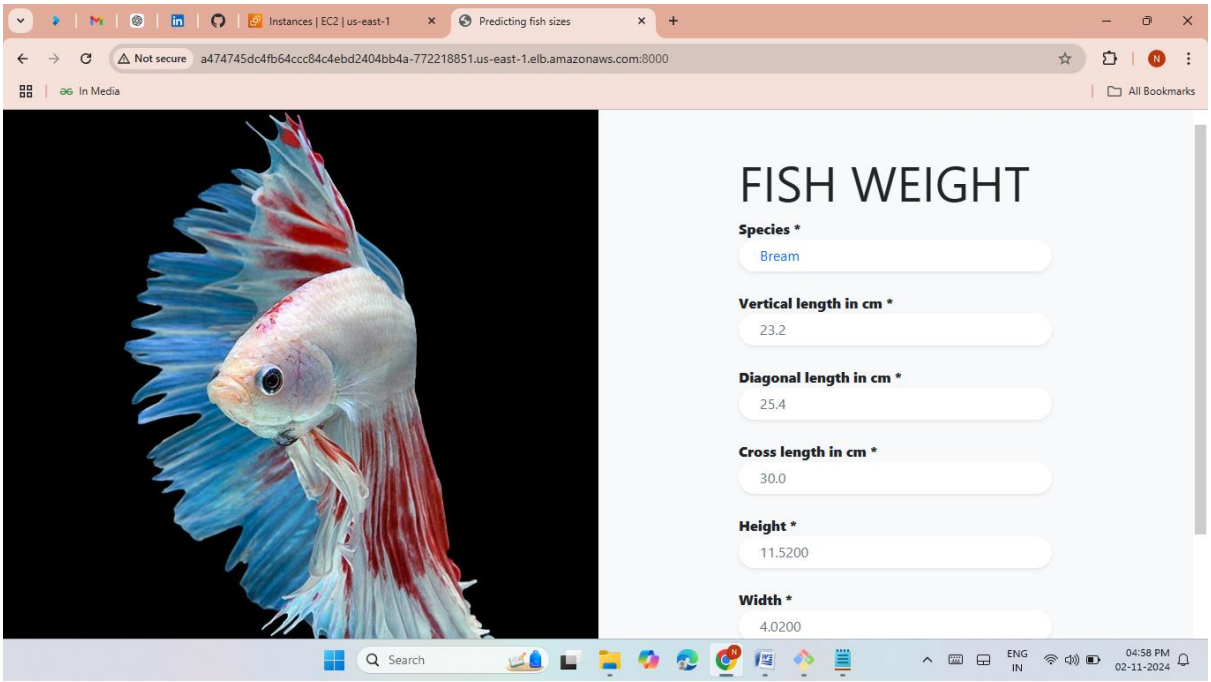

ec2-user@ip-172-31-28-112:~
apiVersion: apps/v1
kind: Deployment
metadata:
name: fish-deployment
labels:
app: fish-app
tier: frontend
spec:
replicas: 4
selector:
matchLabels:
tier: frontend
template:
metadata:
labels:
tier: frontend
spec:
containers:
- name: yashwanth
image: yashwanthnajana/python
ports:
- containerPort: 8000
-- INSERT --
4,12
All
Search
ENG
IN
04:58 PM
02-11-2024


ec2-user@ip-172-31-28-112:~
apiVersion: v1
kind: Service
metadata:
name: fish-service
spec:
type: LoadBalancer
ports:
- port: 8000
selector:
tier: frontend
-- INSERT --
4,12
All
Search
ENG
IN
04:59 PM
02-11-2024

➢ Here the AWS EKS Cluster was created.



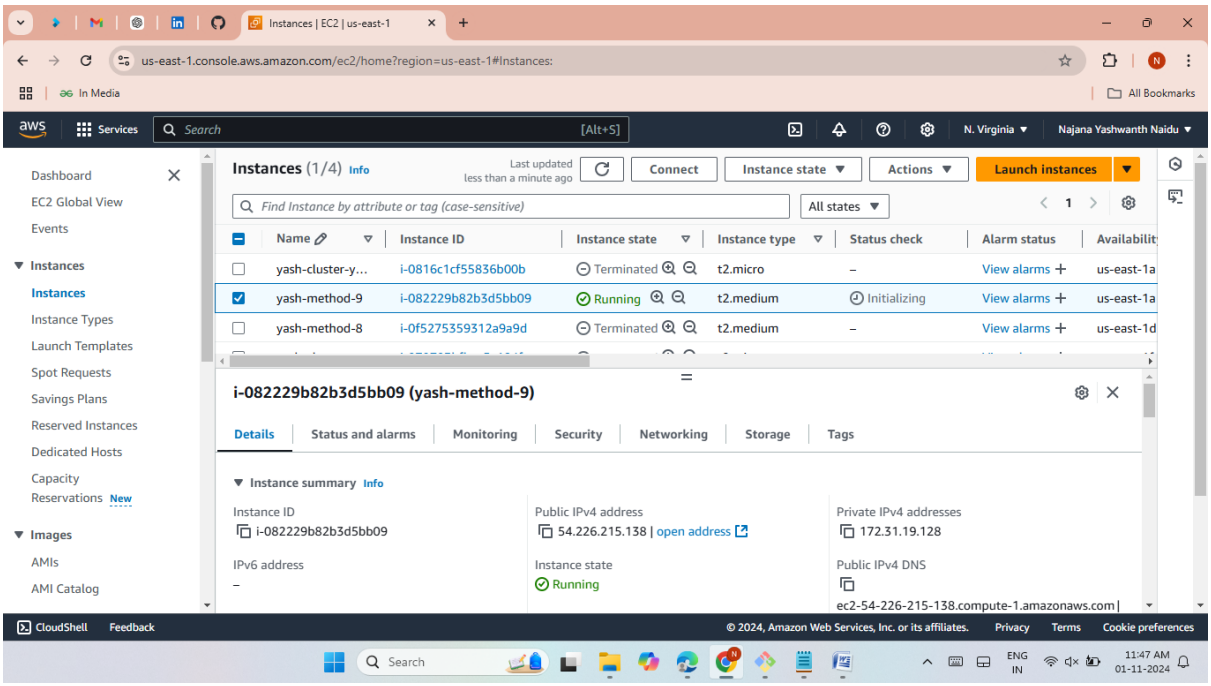

➢ Now with help of Load Balancer DNS link along with given port the fish python we application was hosted successfully.



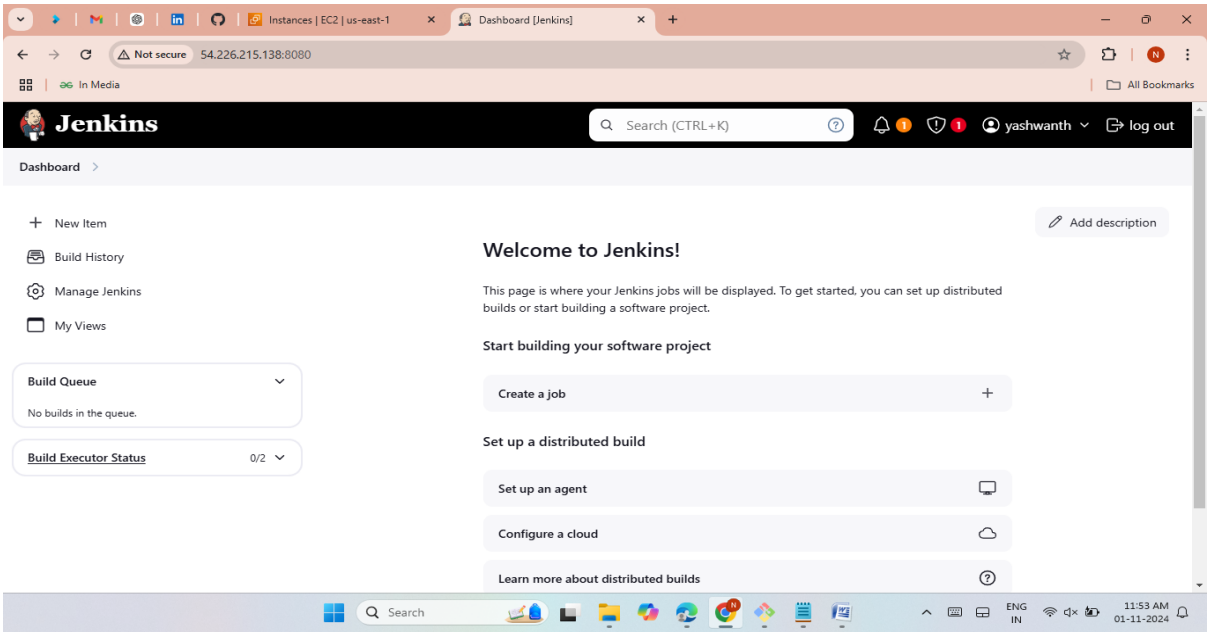
# METHOD-9

## Build and deploy python applications with the Git, github, Jenkins and Terraform (with Build periodically, pollscm and webhooks)

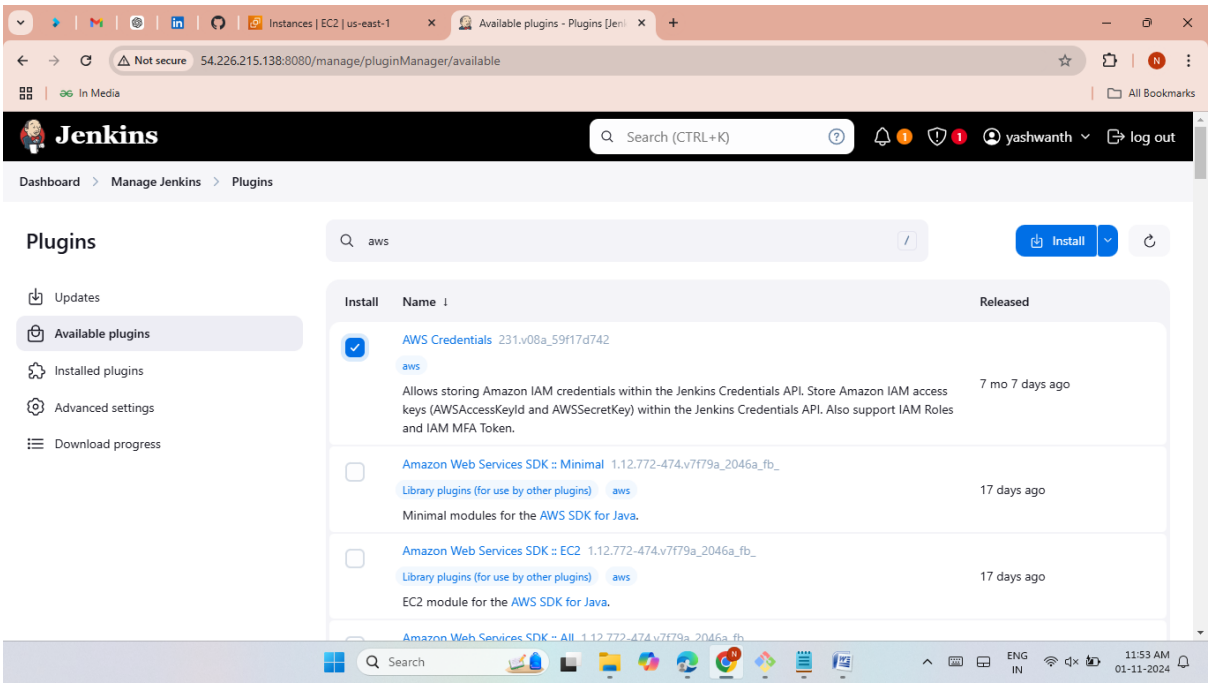
- First launch the EC2 instance with the Jenkins port number(8080) and required ports.



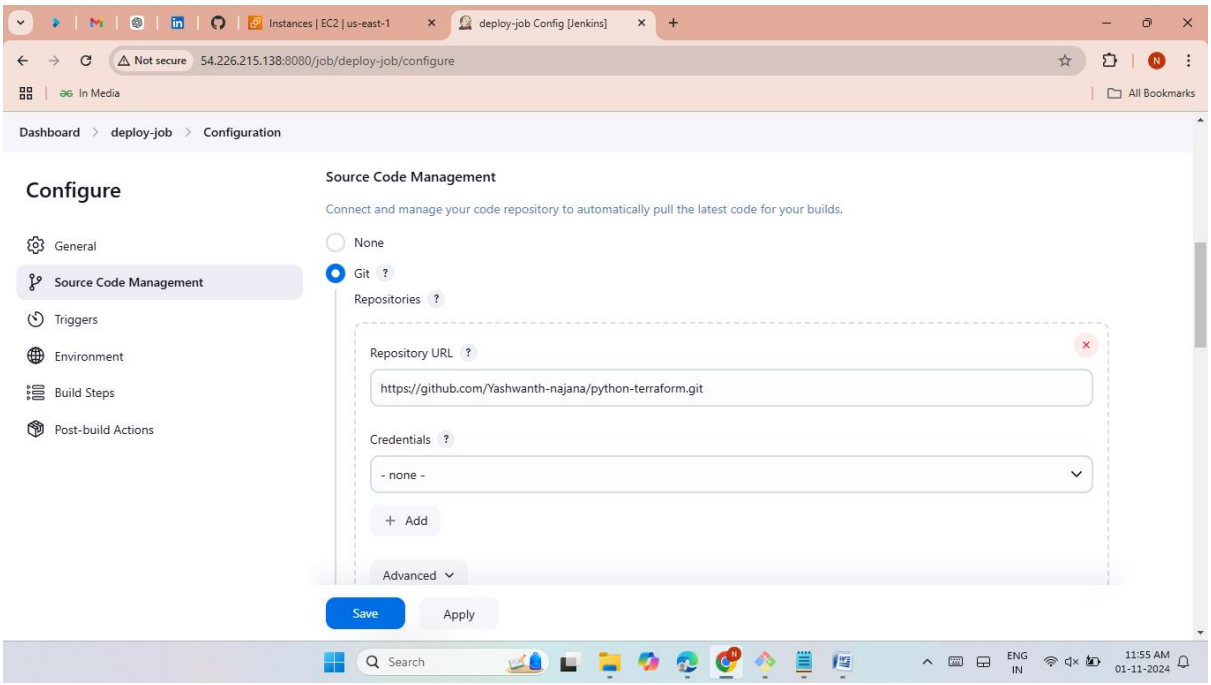

- Now Install git and Jenkins.
- Now start and enable the Jenkins.
- Here the Jenkins was launched successfully.



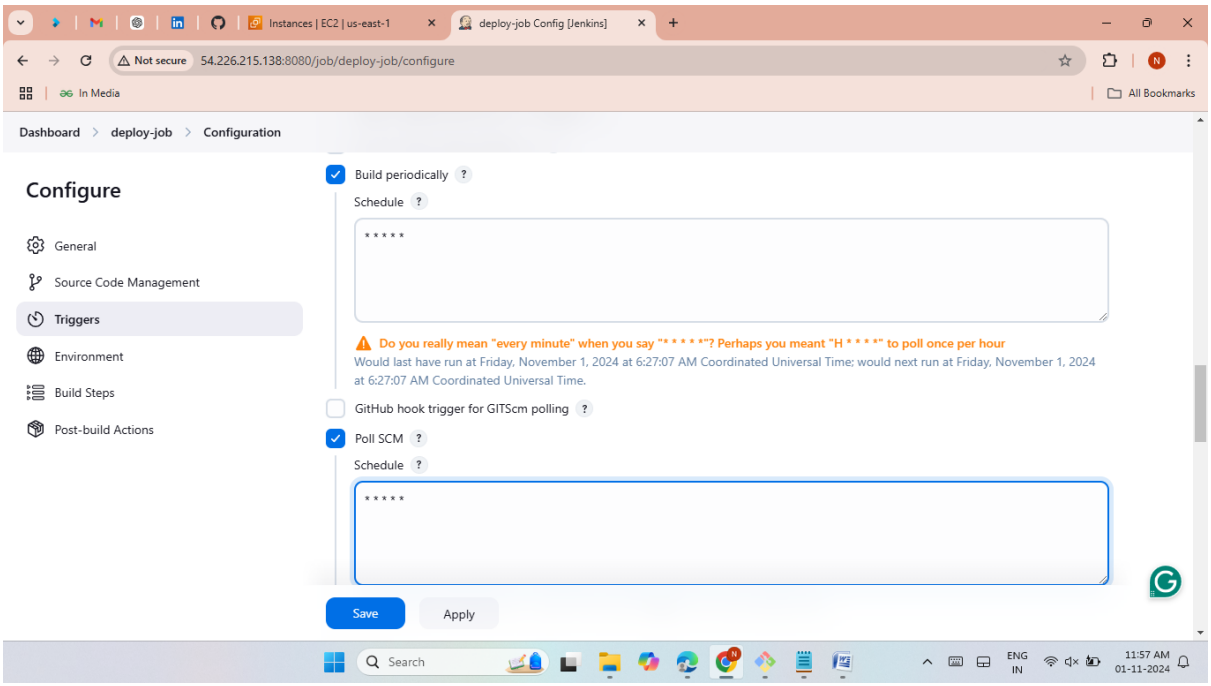
➢ Now install the AWS credentials plugin.



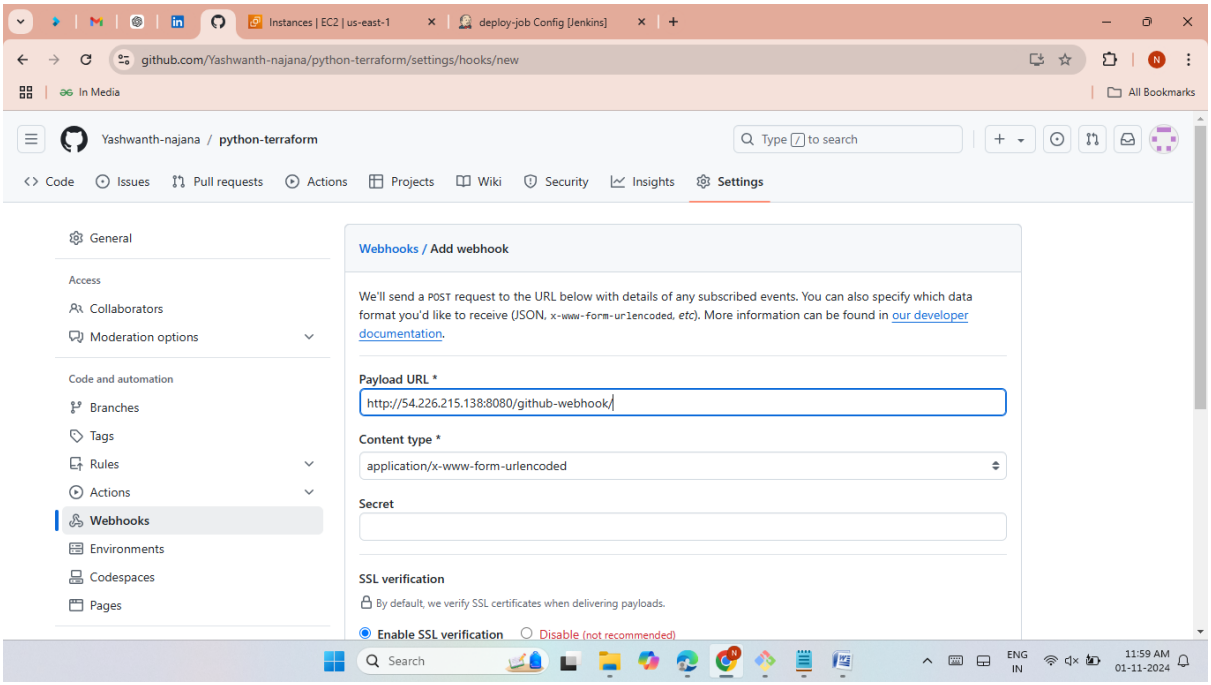

➢ Now create a job and clone the terraform script form github.



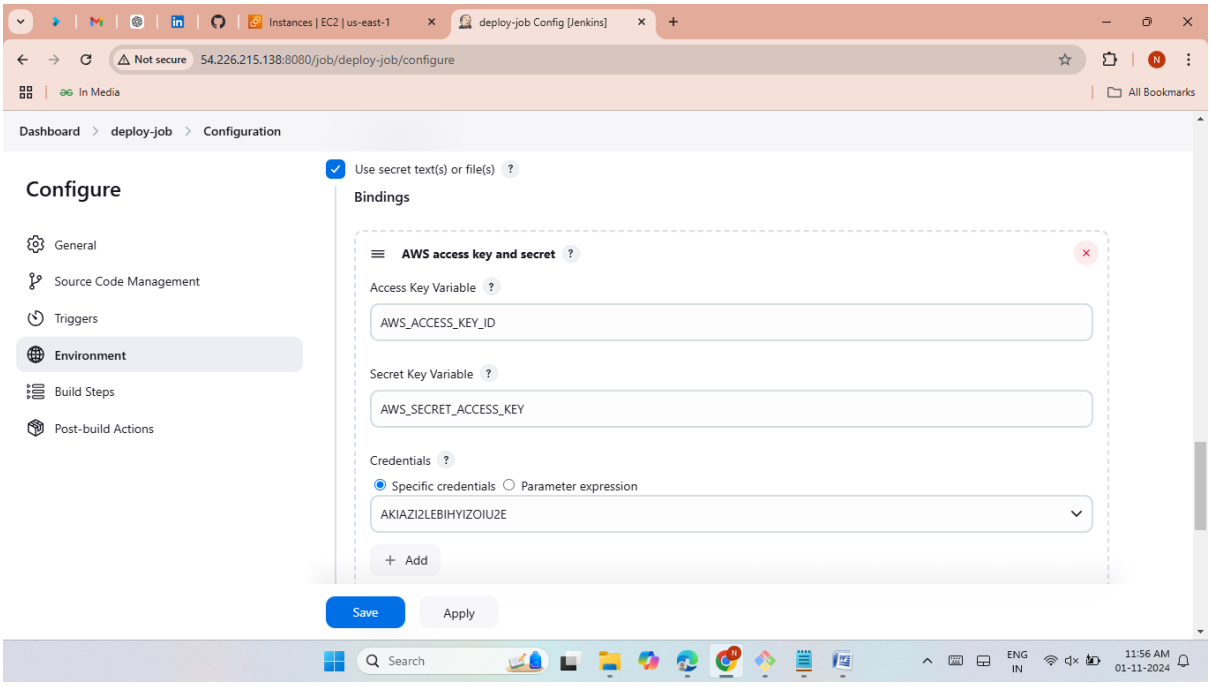
- Now Add the Build Periodically and Poll SCM.




- Create webhooks.
- Github→repo→settings→webhooks→add webhooks→under payload→ip-adress with Jenkins port/github-webhook/



- Then create the AWS credentials.
- Now connect the access keys and secret keys.




- Now install the terraform in execute shell.
- Now add the terraform commands.

**#!/bin/bash**

**sudo yum install -y yum-utils shadow-utils**

**sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo**

**sudo yum -y install terraform**

**sudo chmod + data.sh**

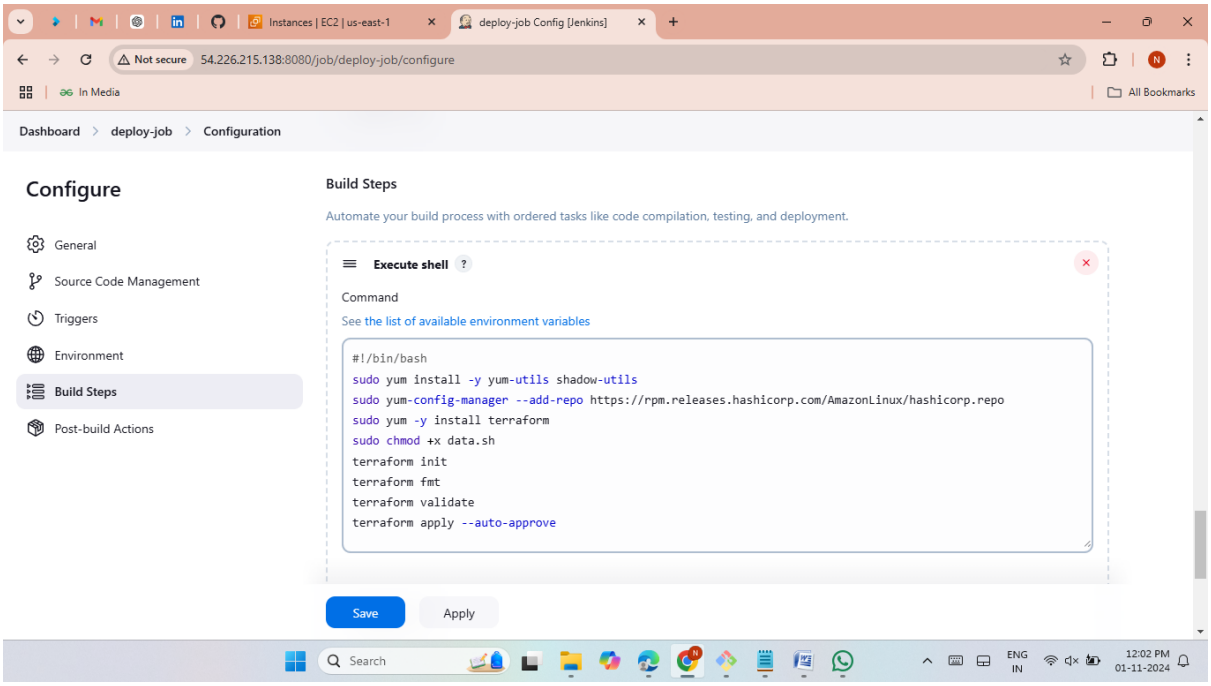
**terraform init**

**terraform fmt**

**terraform validate**

**terraform plan**

**terraform apply --auto-approve**

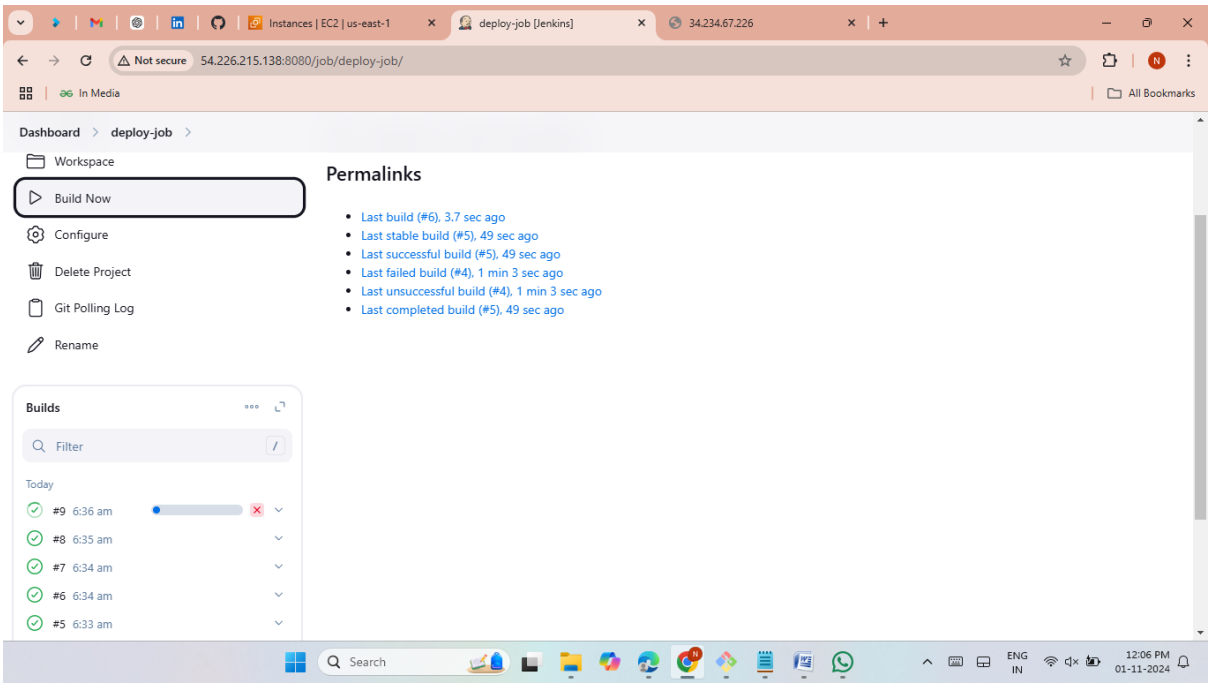



- Give permissions to jenkins.

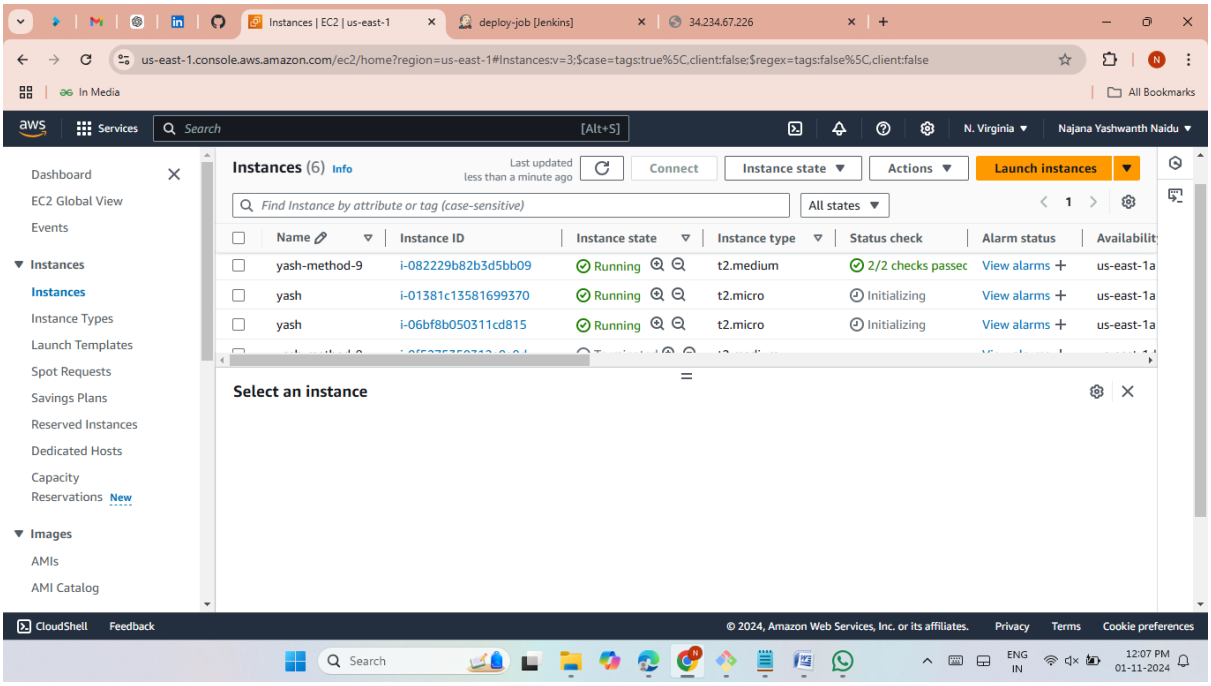
  **Sudo visudo**

  **jenkins ALL=(ALL) NOPASSWD: ALL**

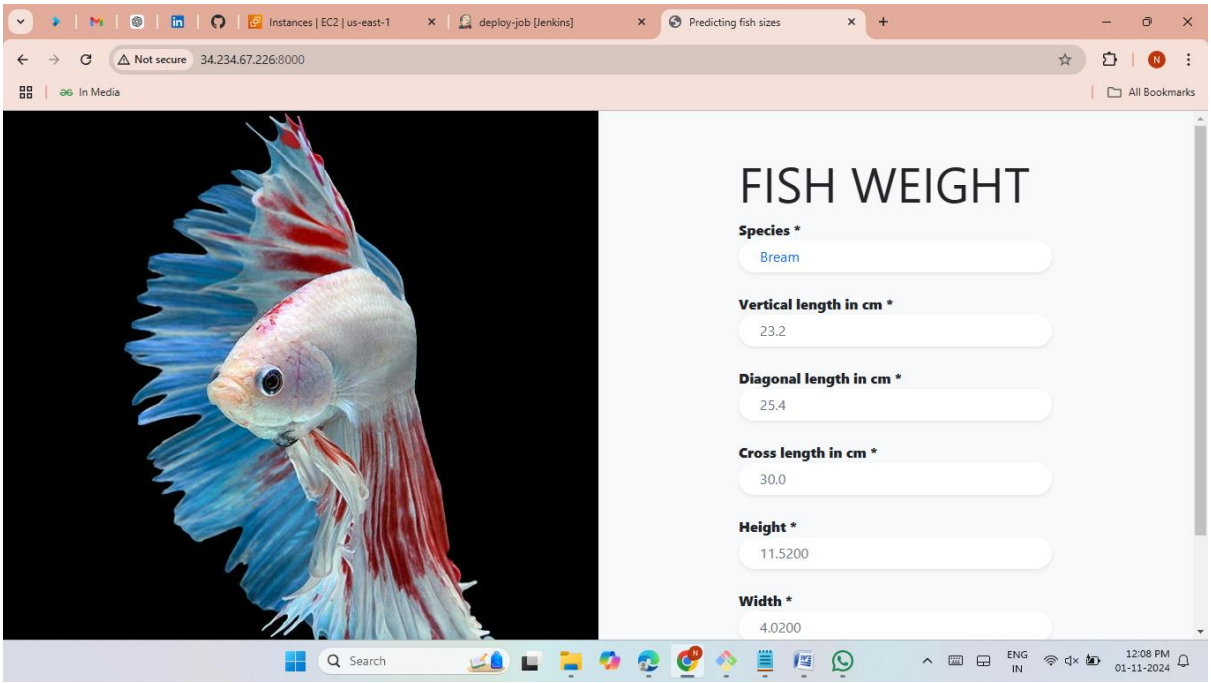
- Now restart the Jenkins with the command (**sudo systemctl restart Jenkins**).
- Now click on build the job was success.
- It runs every MINUTE HOUR DOM(DAY OF MONTH) MONTH DOW(DAY OF WEEK).
- Because I have given (* * * * *).



- Here the instances was automatically launched every one min.
- Also create the VPC, Subnet, Internet Gateway, Route Table, Security Groups with help of terraform script.
- Because of Build Periodically, Poll SCM and Webhooks.



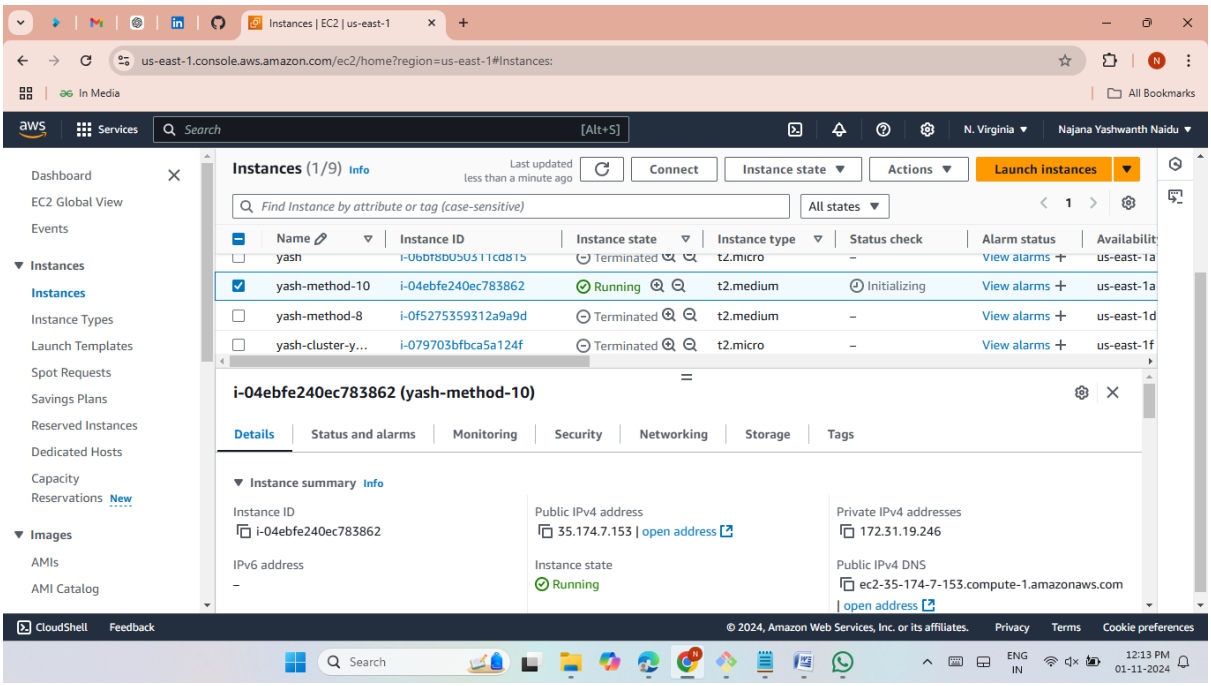

- Here the all fish python applications was launced.



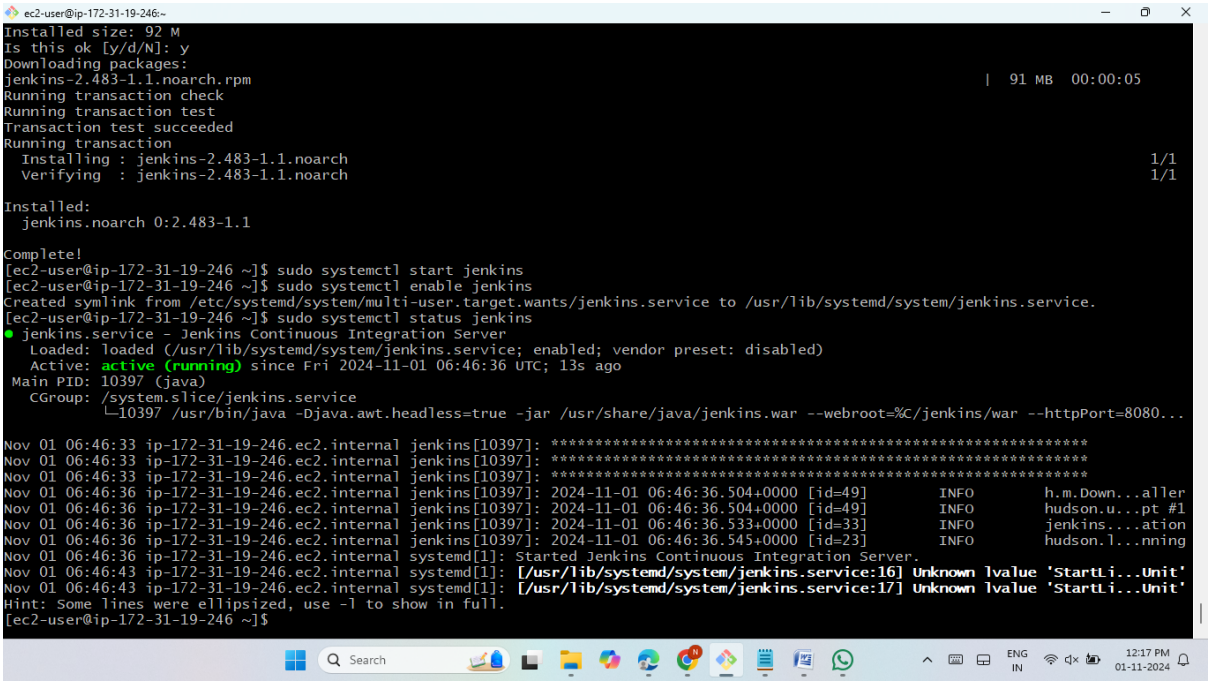
# METHOD-10

## Build and deploy python applications with pipeline method (create clone job and Build job) with Build periodically, pollscm and webhooks

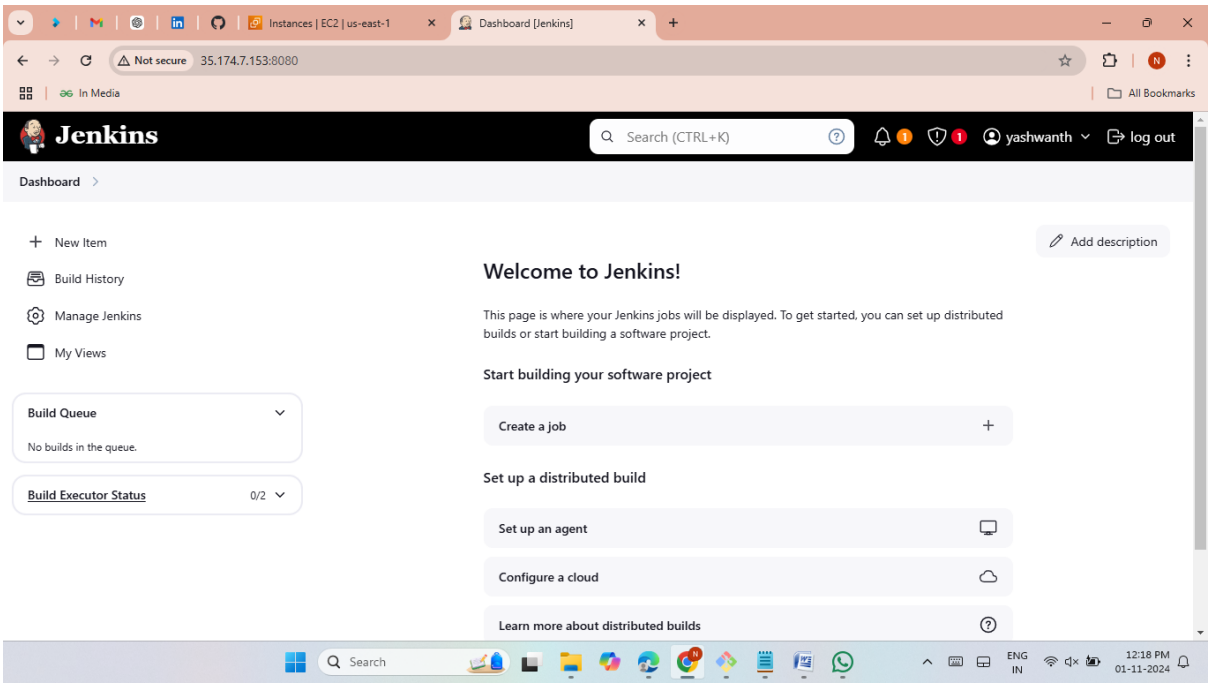
- First launch the EC2 instance with the Jenkins port number(8080) and required ports.



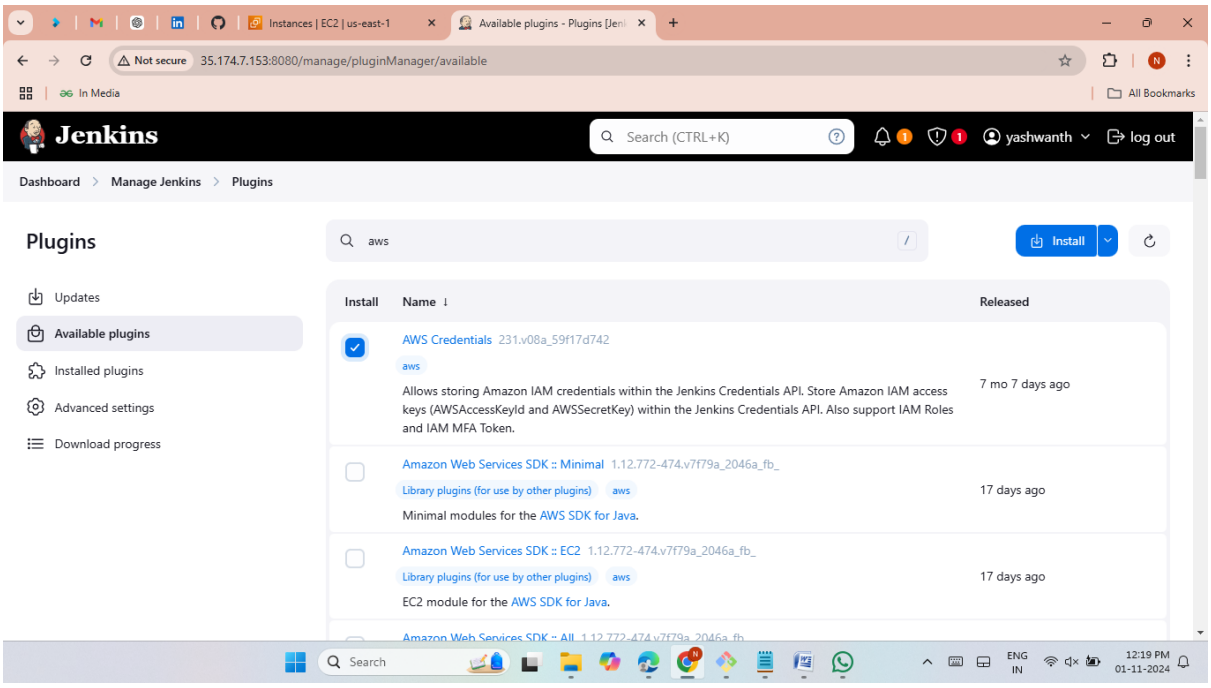

- Now Install git and Jenkins.
- Now start and enable the Jenkins.



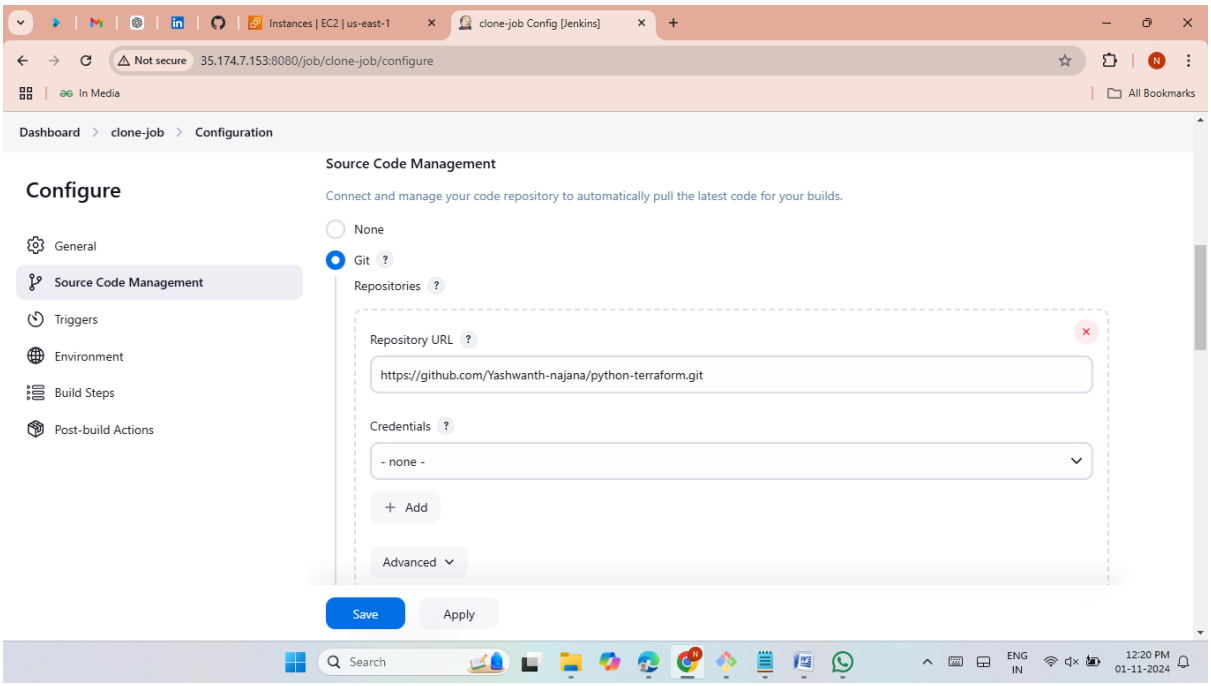
➢ Here the Jenkins was launched successfully.



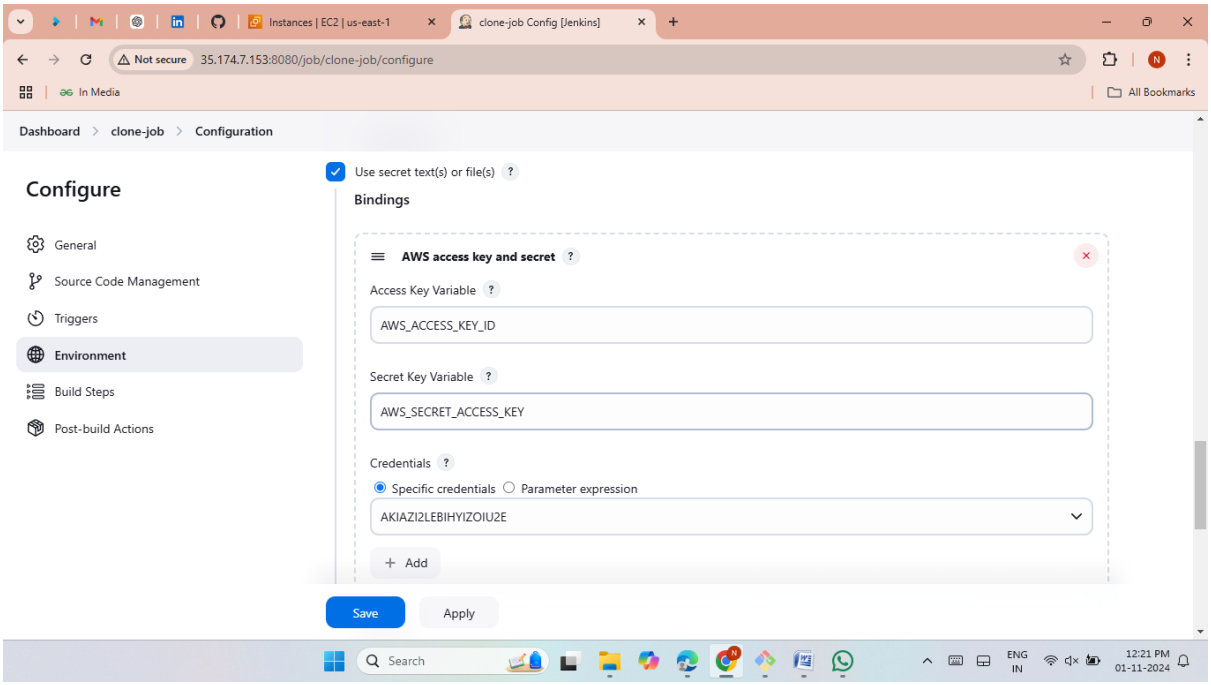

➢ Now install the AWS credentials plugin.



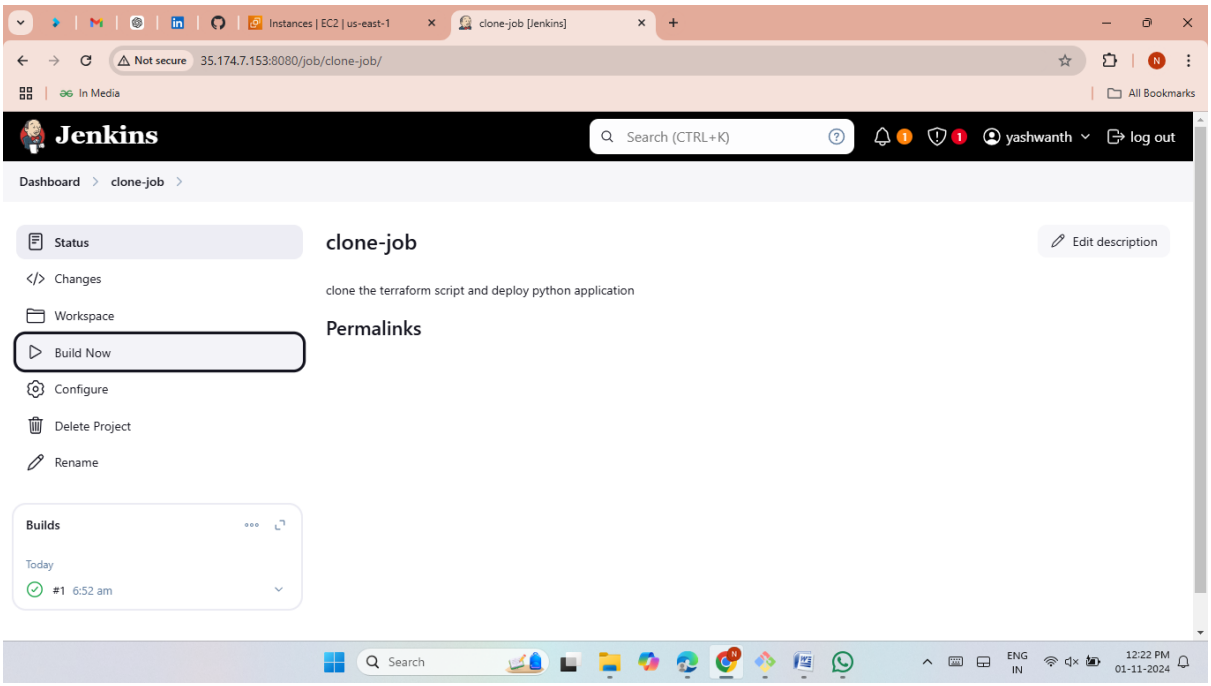
- Now create a clone-job and clone the terraform script form github.



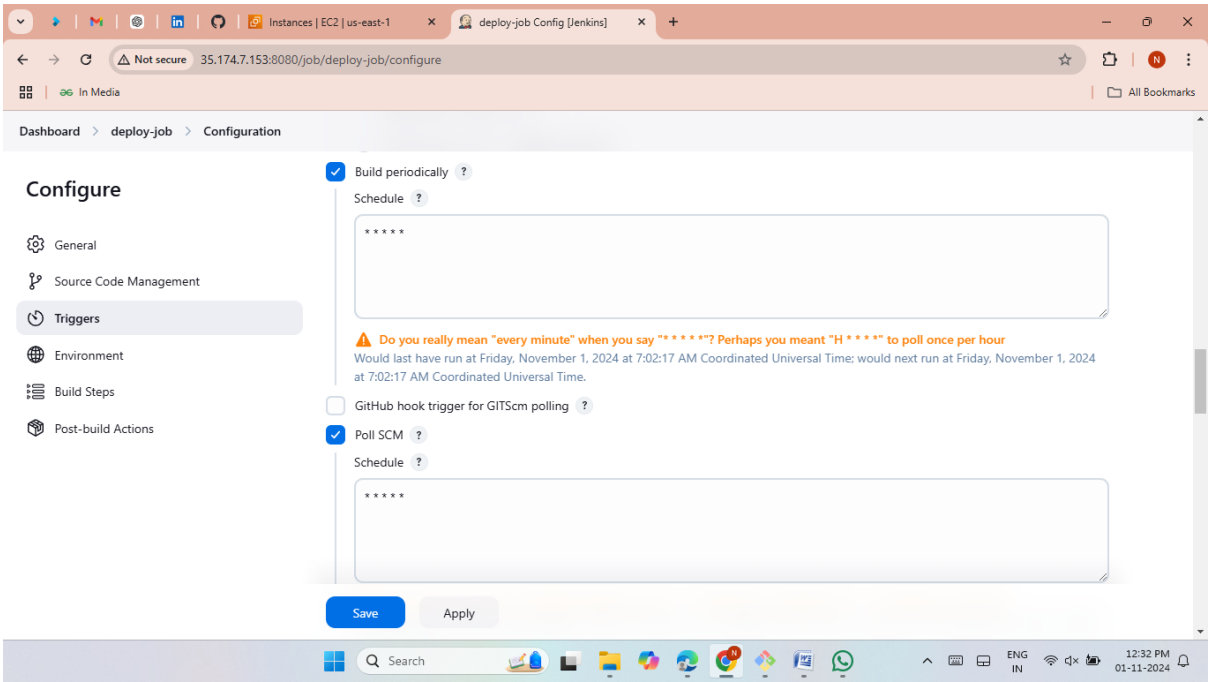

- Then create the AWS credentials.
- Now connect the access keys and secret keys.
- Now apply the changes.



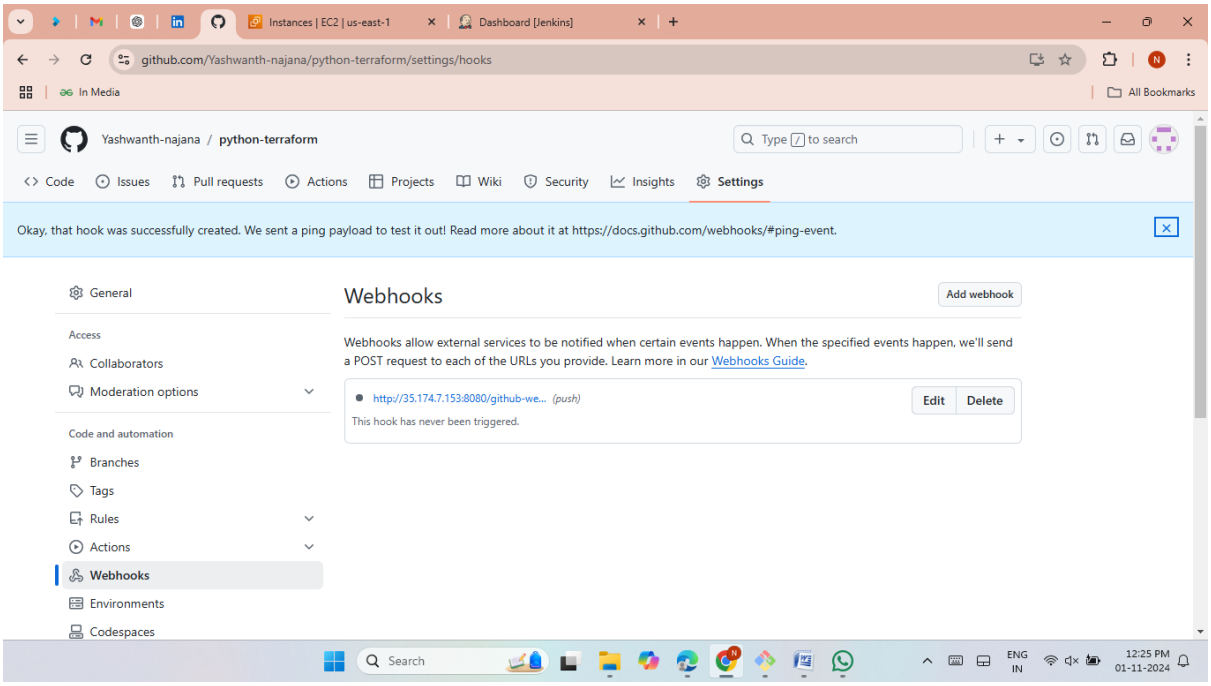
- Here the clone job is success.




- Now create new job and connect to the clone-job.
- Now Add the Build Periodically and Poll SCM.



- Now Create Webhooks.
- Github→repo→settings→webhooks→add webhooks→under payload→ip-adress with Jenkins port/github-webhook/




- Now install the terraform in execute shell.
- Now add the terraform commands.

**#!/bin/bash**

**sudo yum install -y yum-utils shadow-utils**

**sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo**

**sudo yum -y install terraform**

**sudo chmod + data.sh**

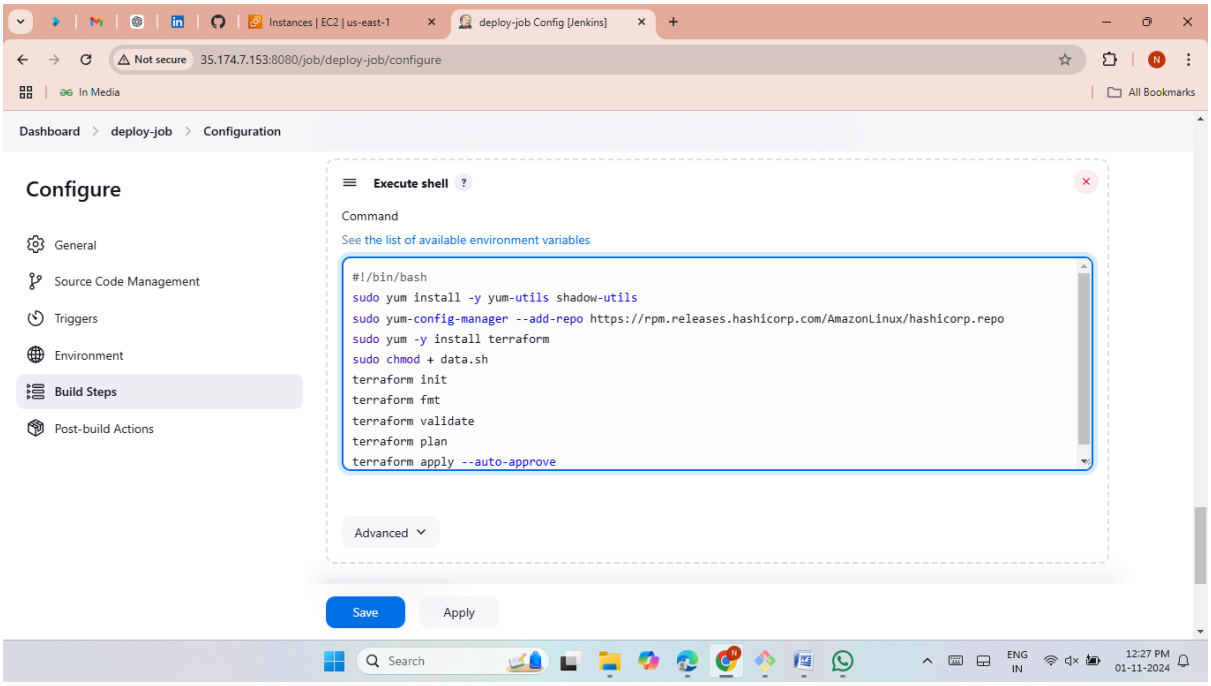
**terraform init**

**terraform fmt**

**terraform validate**

**terraform plan**

**terraform apply --auto-approve**

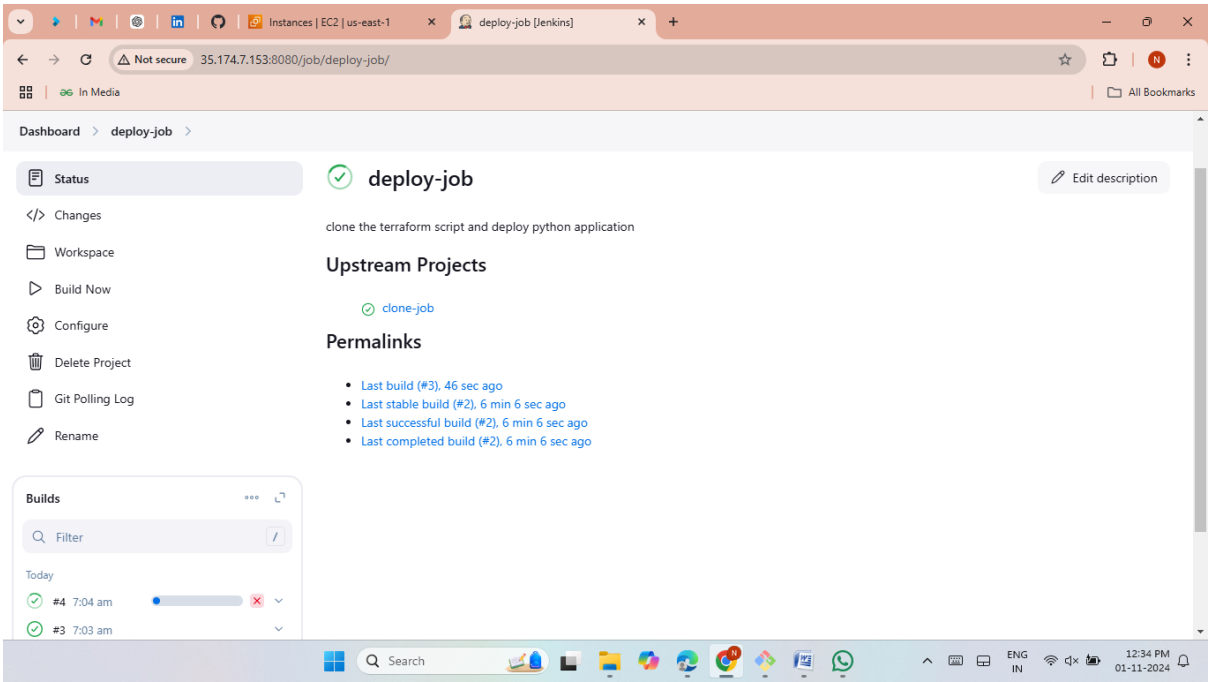



- Give permissions to jenkins.

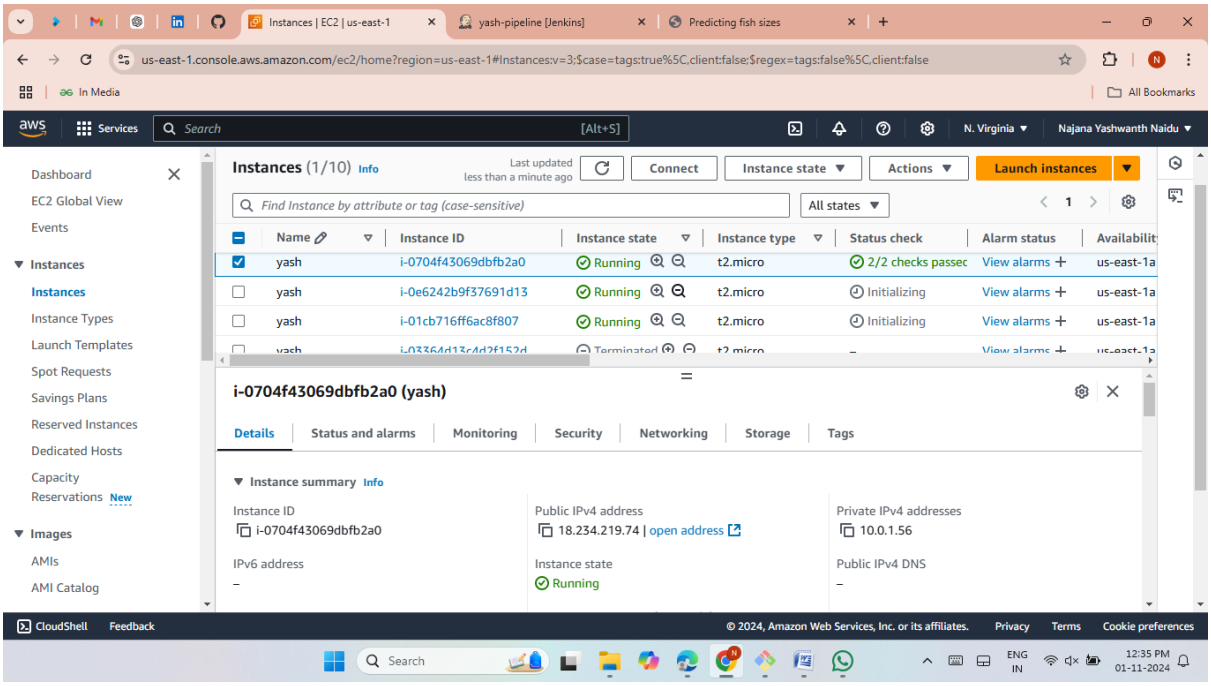
  **Sudo visudo**

  **jenkins ALL=(ALL) NOPASSWD: ALL**

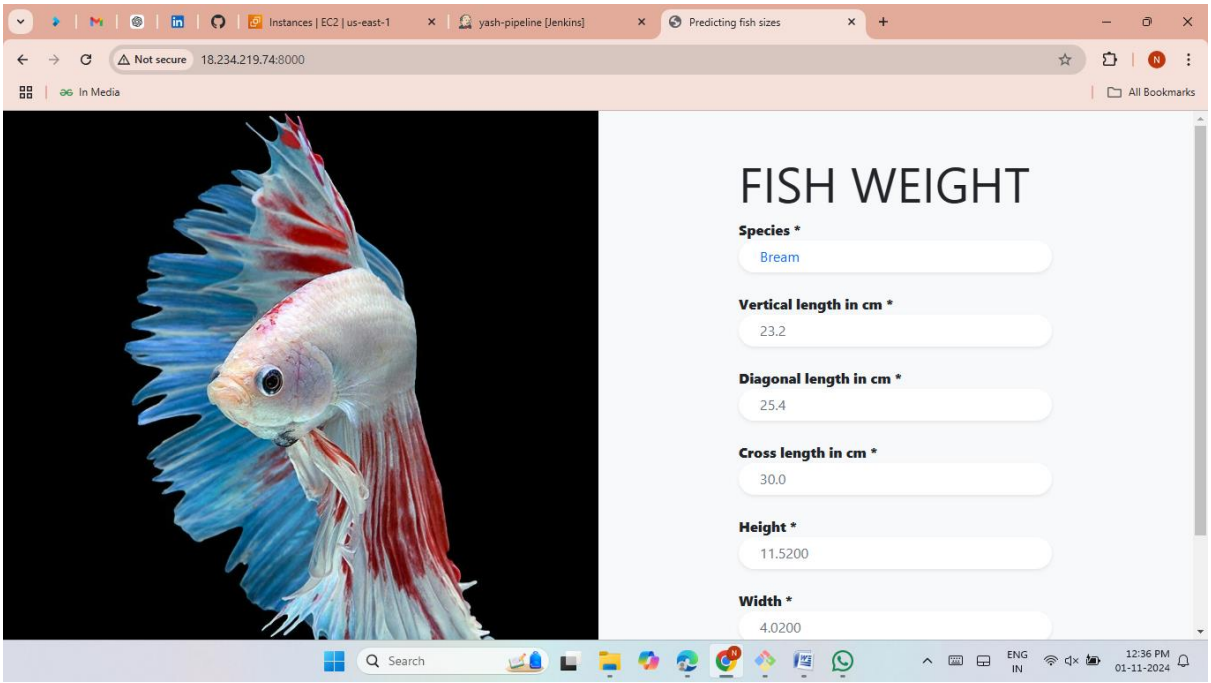
- Now restart the Jenkins with the command (**sudo systemctl restart Jenkins**).
- Now click on build the job was success.
- It runs every MINUTE HOUR DOM(DAY OF MONTH) MONTH DOW(DAY OF WEEK).
- Because I have given (* * * * *).



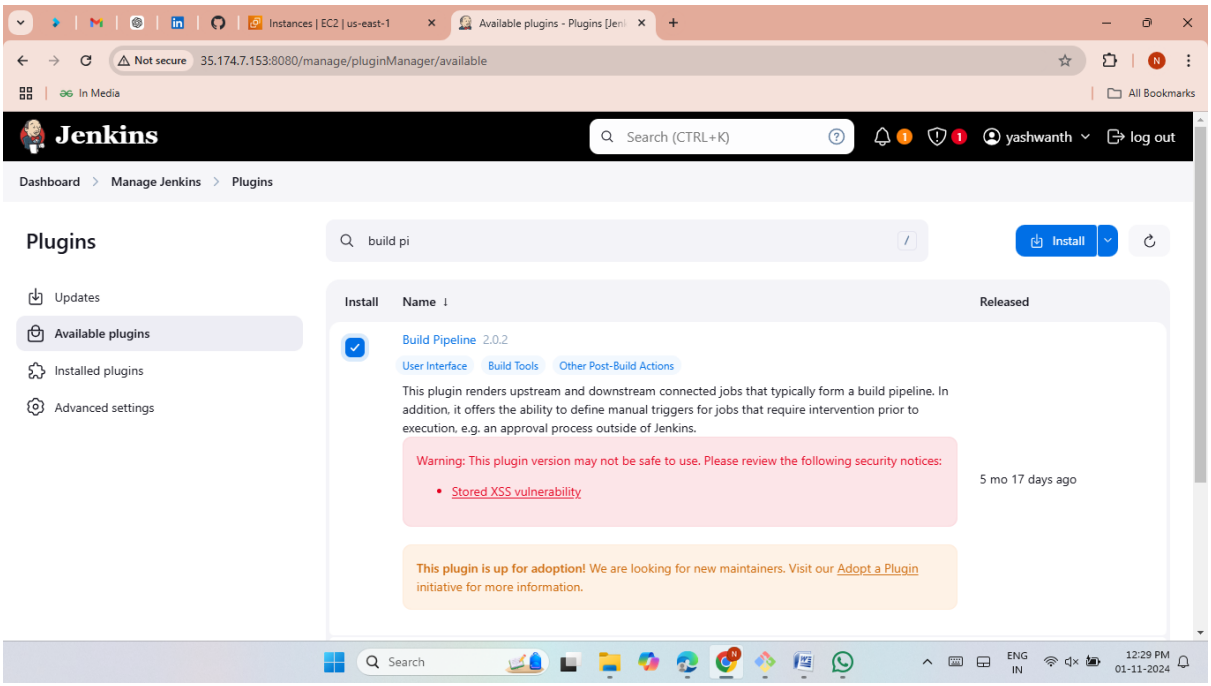
- Here the instances was automatically launched every one min.
- Also create the VPC, Subnet, Internet Gateway, Route Table, Security Groups with help of terraform script.
- Because of Build Periodically, Poll SCM and Webhooks.



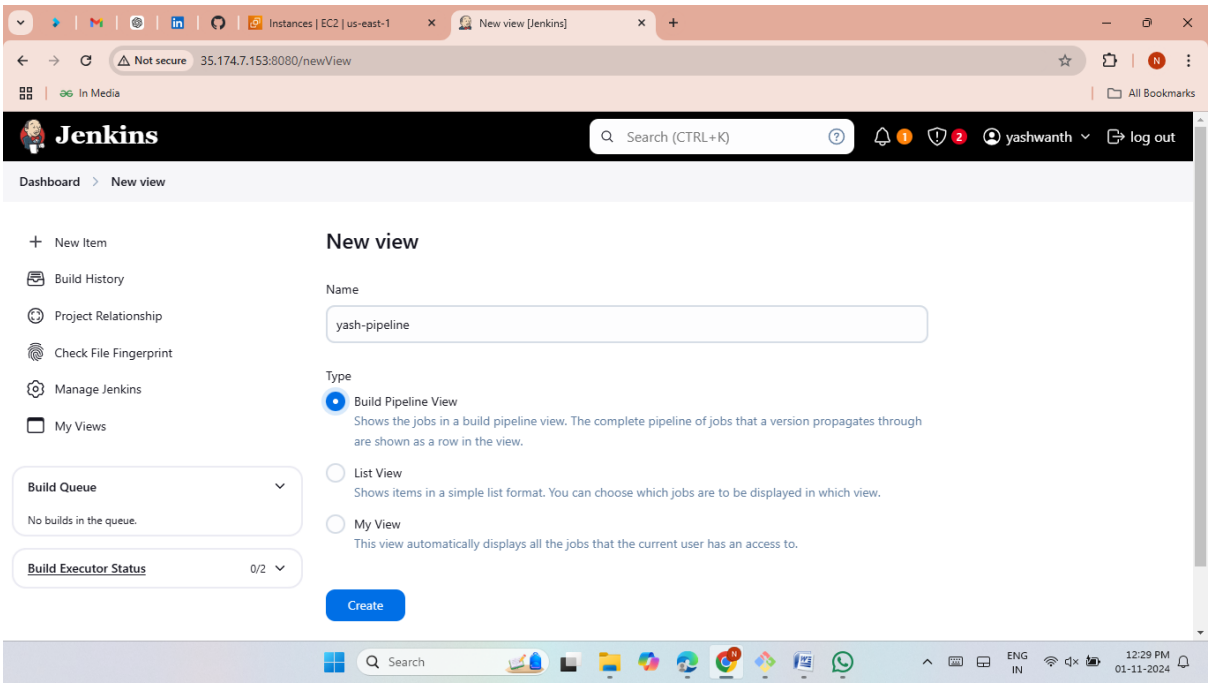

- Here the all fish python applications was launced.



➢ Now install the build pipeline plugin for create a Jenkins pipeline.




➢ Now create a Jenkins pipeline.



➢ Here the build pipeline is successfully created.