

Python

What is Python?

Python is a very high-level programming language, it is effortless to learn. It was created by **Guido van Rossum, and released in 1991.**

Definition:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

Designed by: **Guido van Rossum**

Developer: **Python Software Foundation**

Filename extensions: **.py,.pyw,.pyz,.pyi,.pyc,.pyd**

First appeared: **20 February 1991; 33 years ago**

Paradigm: Multi-paradigm: **object-oriented, procedural (imperative), functional, structured, reflective**

Stable release: **3.12.3 / 9 April 2024; 41 days ago**

Typing discipline: **duck, dynamic, strong; optional type annotations (since 3.5, but those hints are ignored, except with unofficial tools)**



It is used for?

- Web development (server-side).
- Software development.
- Mathematics.
- System scripting.

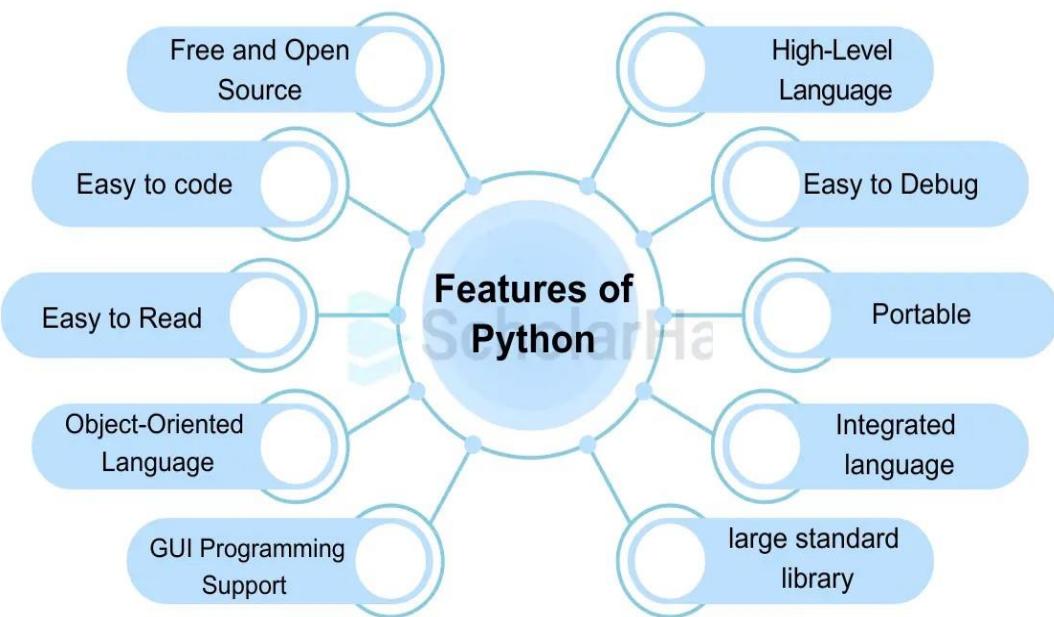
Advantages:

- Easy to Read and Learn.
- Reduces Maintenance Cost.
- Avoid the Harm of Software Bugs.
- Wide Applicability.
- Easy Memory Management.
- Large Community.
- Asynchronous Coding.
- Integration with Other Languages.

Disadvantages:

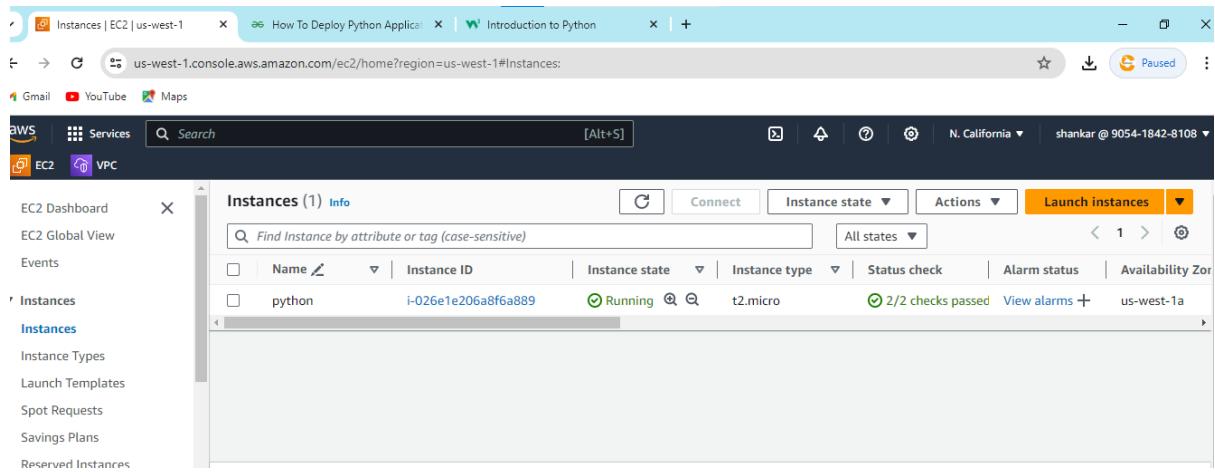
- Python is Slow at Runtime.
- Mobile Application Development.
- Difficulty in Using Other Languages.
- High Memory Consumption.
- Not used in the Enterprise Development Sector.
- Runtime Errors.
- Simplicity.

Features:



1. Build and deploy python applications in manual process?

- + Go to EC2 instance.
- + Select Ubuntu server.
- + Edit inbound rules.
- + Launch EC2 instance.



Step 1: (Flask-Library-app)

- + Connect to Git Bash.
- + Go to Git Bash.
- + Update Ubuntu machine using these following command as
`< sudo apt update >`

```
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [517 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [852 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [164 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.8 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.2 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7588 B]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 31.4 MB in 6s (5484 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
28 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-0-154:~$ |
```



- Full upgrade the machine using these following command as

```
< sudo apt-get full-upgrade -y >
```

```
Restarting services...
systemctl restart acpid.service chrony.service cron.service multipathd.service packagekit.service polkit.service serial-getty@ttys0.service systemd-journald.service systemd-networkd.service systemd-resolved.service systemd-udevd.service
Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-0-154:~$ |
```



- Install required packages or tools related for deployment project.
- Install python and pip using these command as follows

```
< sudo apt-get install python-3 pip >
```

```
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.  
ubuntu@ip-172-31-0-154:~$ pip3 --version  
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)  
ubuntu@ip-172-31-0-154:~$ python3 --version  
Python 3.10.12  
ubuntu@ip-172-31-0-154:~$
```



⊕ Check the status < pip3 --version>

```
systemctl restart user@1000.service  
No containers need to be restarted.  
No user sessions are running outdated binaries.  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-8-108:~$ pip --version  
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)  
ubuntu@ip-172-31-8-108:~$ |
```



⊕ Install git and clone the project source code from Git Hub.

< sudo git clone (repo name) >

⊕ After that using run these following commands as Git Bash.

< cd (repo name) >

⊕ Install requirements packages

< pip3 install –r requirements.txt >

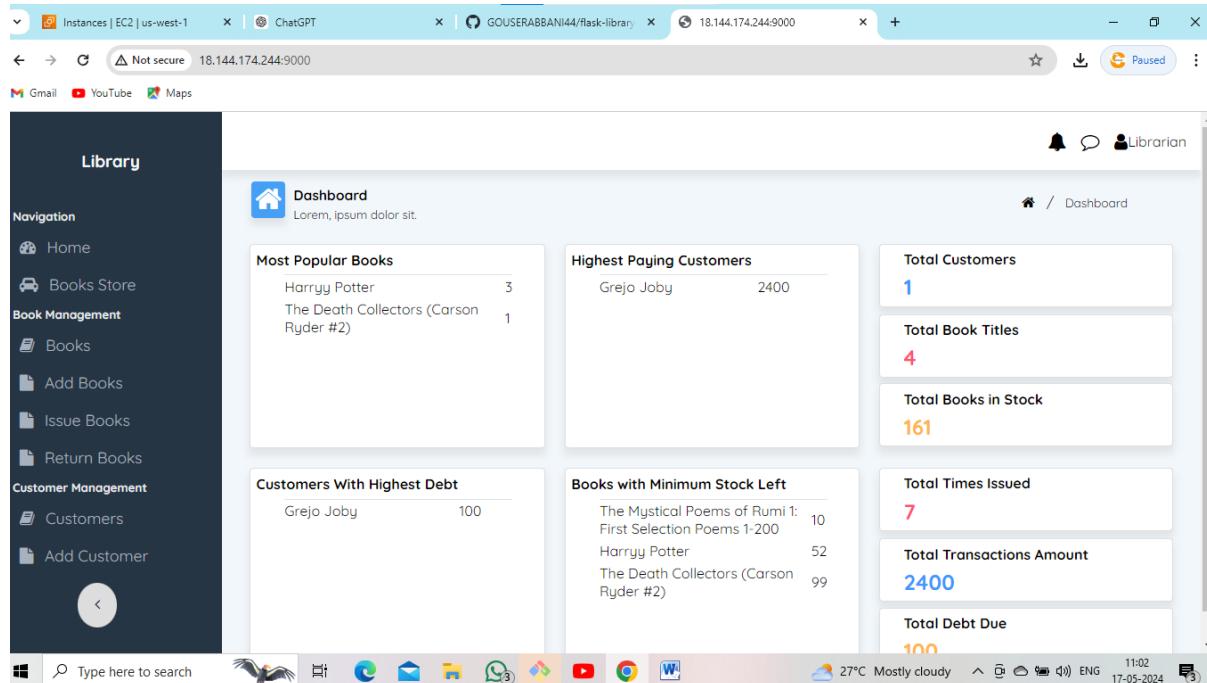
⊕ After running python app.py. It will generate local host IP address. using these command as

< sudo vi app. py >

⊕ After run flask server

- Go to EC2 instance, copy public ip and paste it Google

< copy public ip : port number>



Step 2: (Fish)

- Using these command as follows

< sudo git clone (repo name) >

- Change the name

< cd (repo name) >

- Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

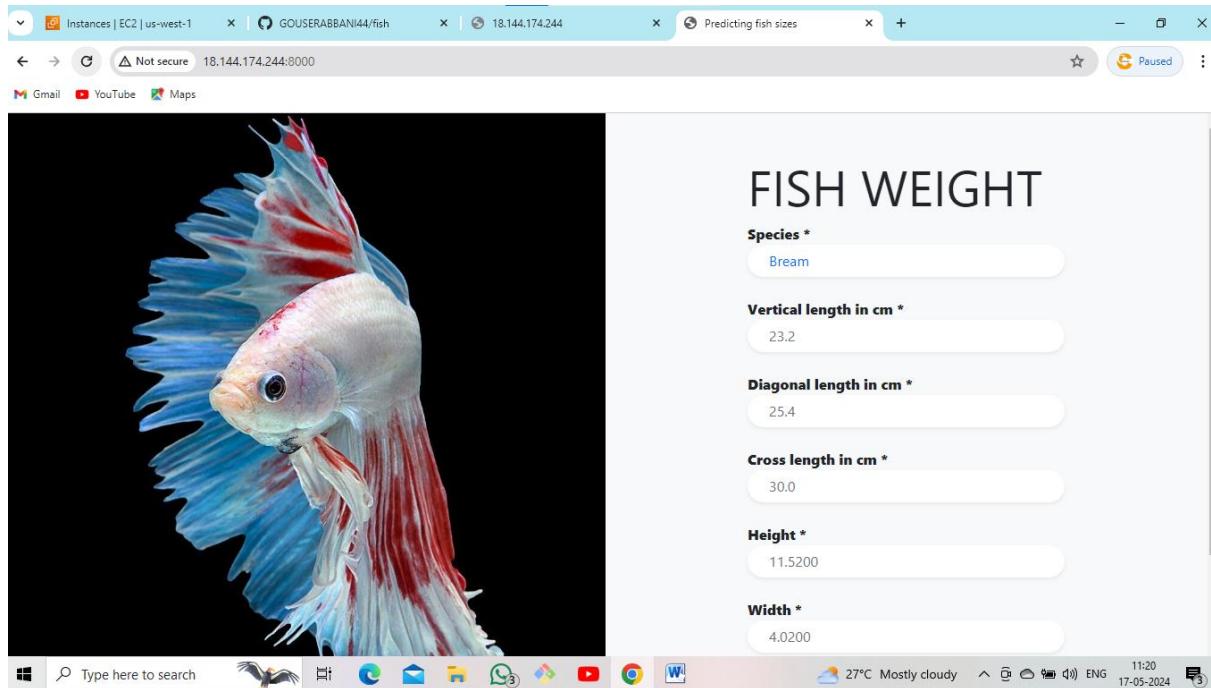
- After that change the port number open the file as

< sudo vi app.py >

- Run the server using these command as

< python3 app.py >

⊕ Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.
<public ip : port number >



Step 3: (Flight - Prediction)

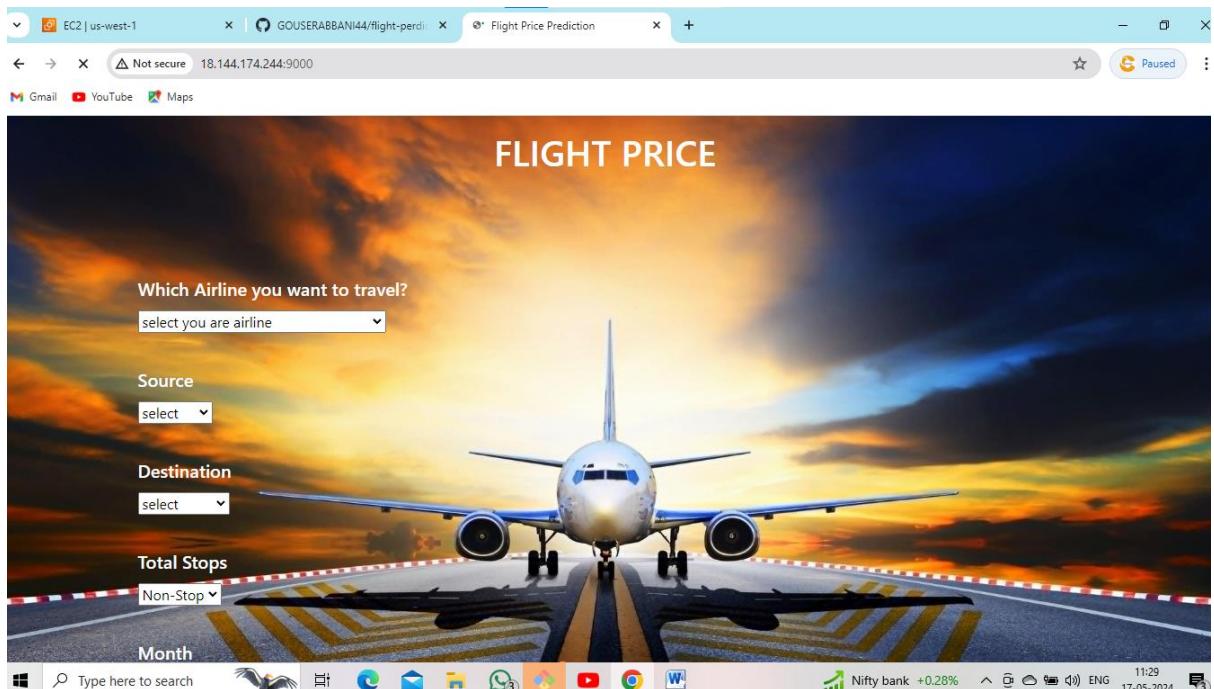
- ⊕ Using these command as follows
< sudo git clone (repo name) >
- ⊕ Change the name
< cd (repo name) >
- ⊕ Install requirements packages using these command as follows
< pip3 install -r requirements.txt >
- ⊕ After that change the port number open the file as
< sudo vi app.py >

Run the server using these command as

< python3 app.py >

Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

<public ip : port number >



Step 4: (Penguin)

Using these command as follows

< sudo git clone (repo name) >

Change the name

< cd (repo name) >

Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

After that change the port number open the file as

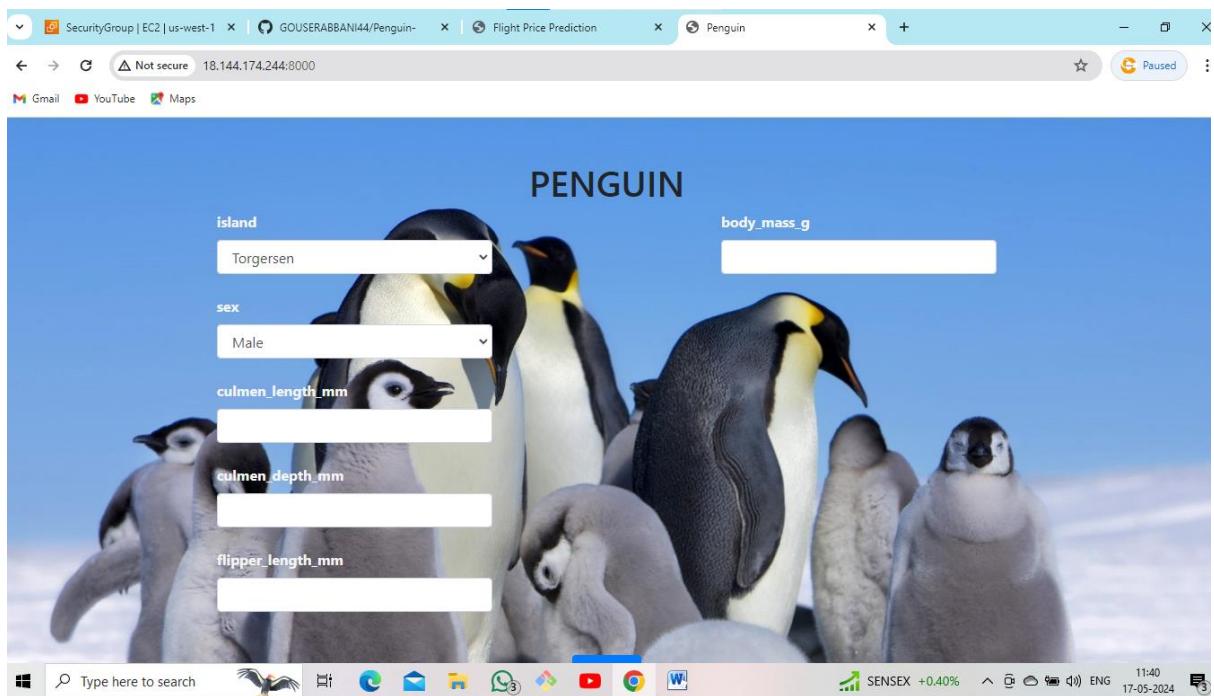
< sudo vi app.py >

Run the server using these command as

< python3 app.py >

Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 5: (Car - Prediction)

Using these command as follows

< sudo git clone (repo name) >

Change the name

< cd (repo name) >

Install requirements packages using these command as follows

`< pip3 install -r requirements.txt >`

+ After that change the port number open the file as

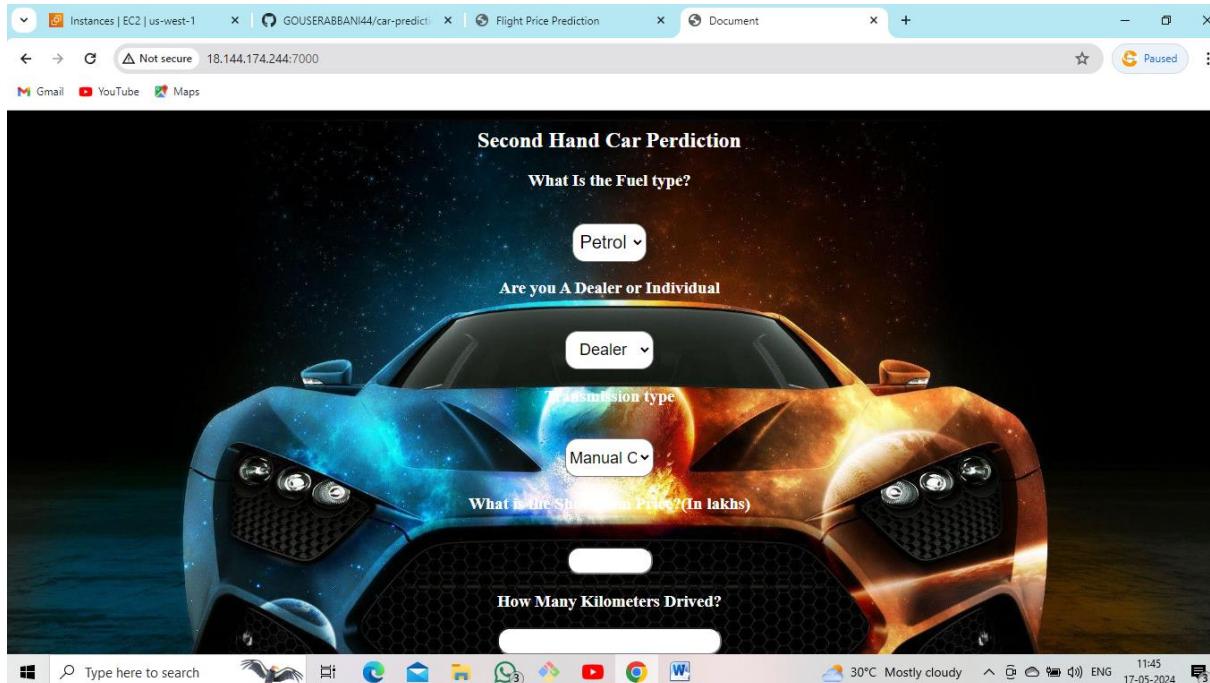
`< sudo vi app.py >`

+ Run the server using these command as

`< python3 app.py >`

+ Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

`< public ip : port number >`



Step 6: (Medical Insurance)

+ Using these command as follows

`< sudo git clone (repo name) >`

+ Change the name

`< cd (repo name) >`

- + Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

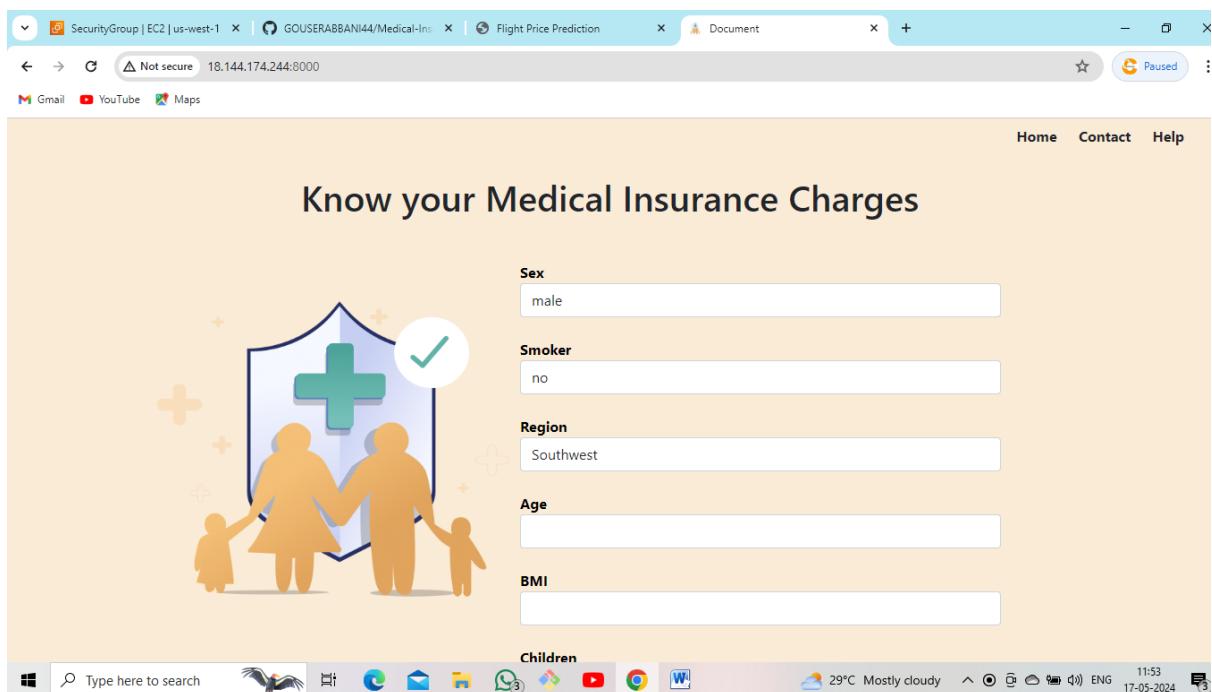
- + After that change the port number open the file as
< sudo vi app.py >

- + Run the server using these command as

< python3 app.py >

- + Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 7: (Portofolio)

- + Using these command as follows
< sudo git clone (repo name) >
- + Change the name

`< cd (repo name) >`

- + Install requirements packages using these command as follows

`< pip3 install -r requirements.txt >`

- + After that change the port number open the file as

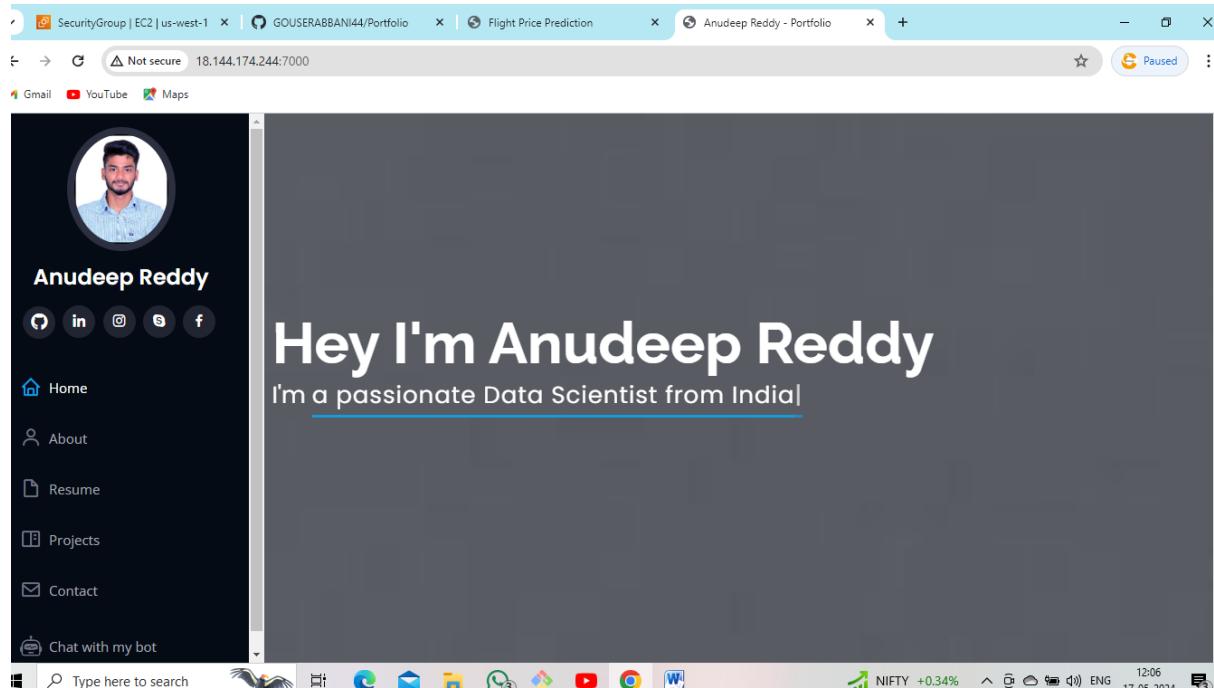
`< sudo vi app.py >`

- + Run the server using these command as

`< python3 app.py >`

- + Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

`< public ip : port number >`



Step 8: (Agri)

- + Using these command as follows

`< sudo git clone (repo name) >`

- + Change the name

`< cd (repo name) >`

- + Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

- + After that change the port number open the file as

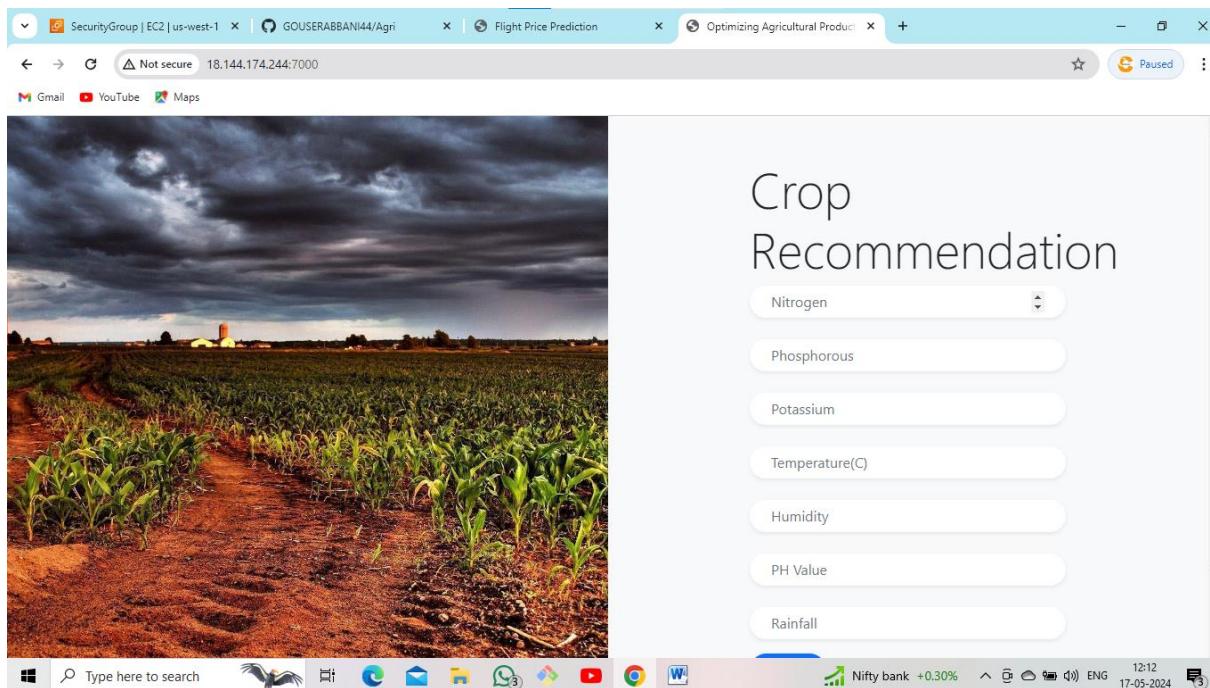
< sudo vi app.py >

- + Run the server using these command as

< python3 app.py >

- + Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 9: (Fuel-consumption-Rating)

- + Using these command as follows

< sudo git clone (repo name) >

- + Change the name

`< cd (repo name) >`

- + Install requirements packages using these command as follows

`< pip3 install -r requirements.txt >`

- + After that change the port number open the file as

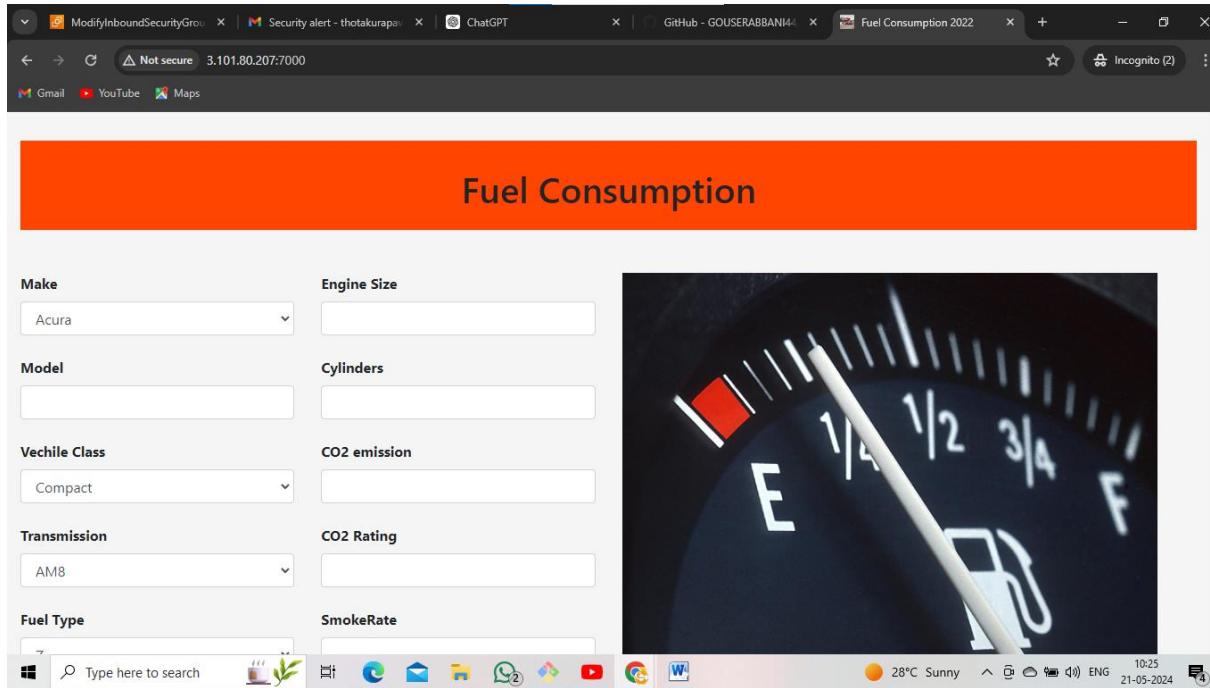
`< sudo vi app.py >`

- + Run the server using these command as

`< python3 app.py >`

- + Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

`< public ip : port number >`



Step 10: (USA-Housing)

- + Using these command as follows

`< sudo git clone (repo name) >`

⊕ Change the name

< cd (repo name) >

⊕ Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

⊕ After that change the port number open the file as

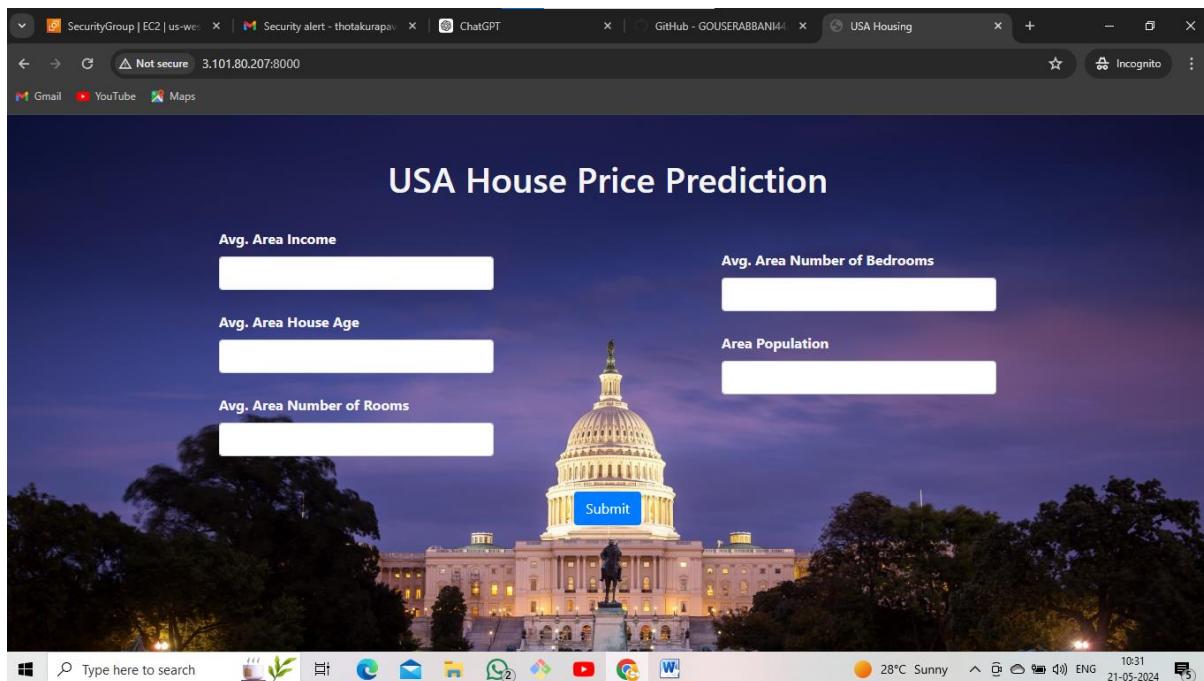
< sudo vi app.py >

⊕ Run the server using these command as

< python3 app.py >

⊕ Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 11: (My Fuel)

⊕ Using these command as follows

< sudo git clone (repo name) >

⊕ Change the name

< cd (repo name) >

⊕ Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

⊕ After that change the port number open the file as

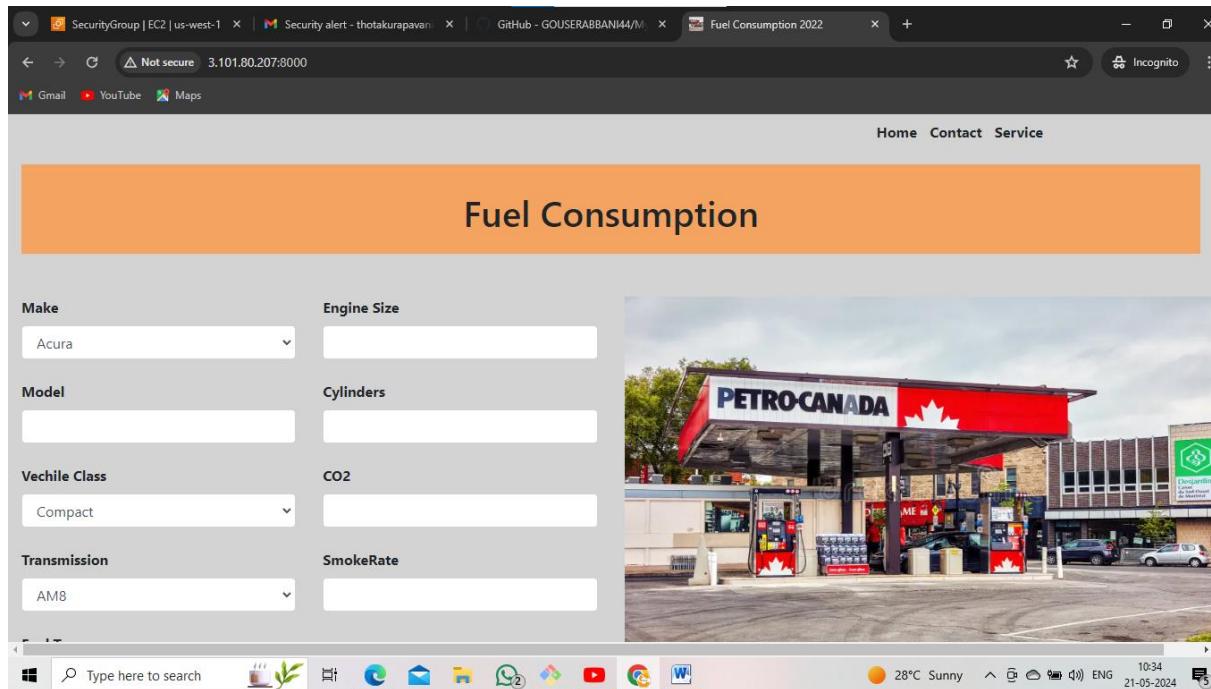
< sudo vi app.py >

⊕ Run the server using these command as

< python3 app.py >

⊕ Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 12: (Indian Liver Patients)

Using these command as follows

< sudo git clone (repo name) >

Change the name

< cd (repo name) >

Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

After that change the port number open the file as

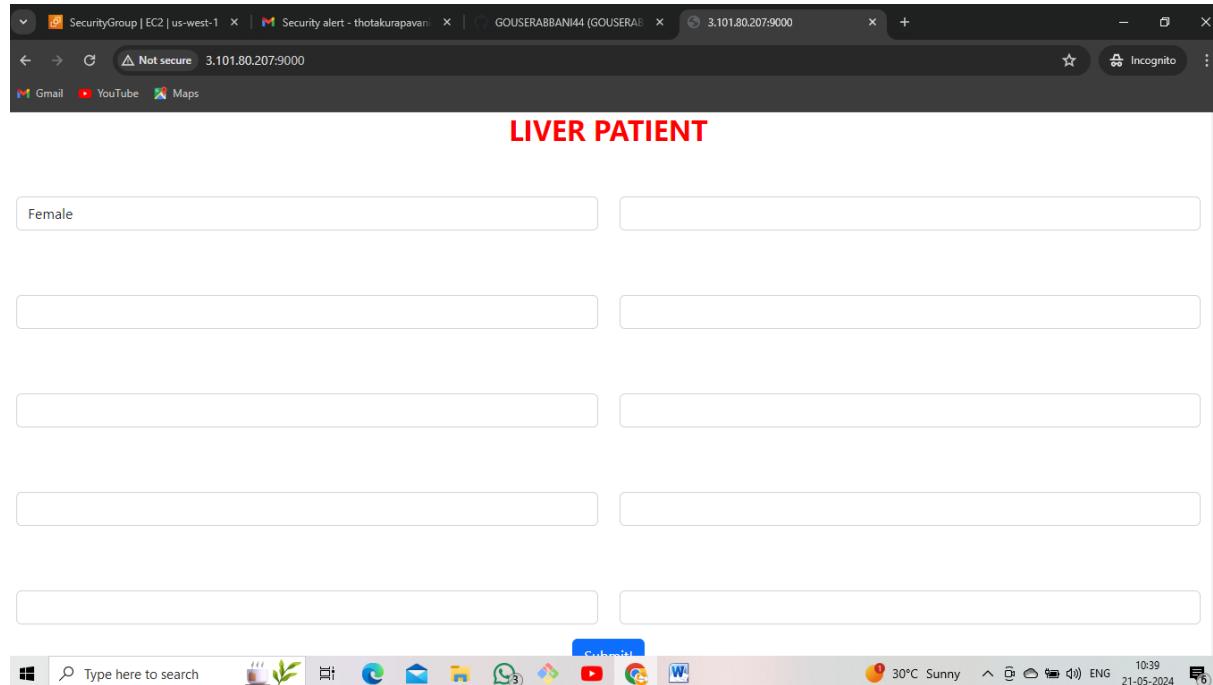
< sudo vi app.py >

Run the server using these command as

< python3 app.py >

Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 13: (InNews)

- + Using these command as follows
`< sudo git clone (repo name) >`
 - + Change the name
`< cd (repo name) >`
 - + Install requirements packages using these command as follows
`< pip3 install -r requirements.txt >`
 - + After that change the port number open the file as
`< sudo vi app.py >`
 - + Run the server using these command as
`< python3 app.py >`
 - + Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.
`< public ip : port number >`
 - + I got error

-  I am using these command for name changing of Templates to templates.

```
< sudo mv Templates templates >
```

-  I am using these command for requirements file.

```
< sudo vi requirements.txt >
```

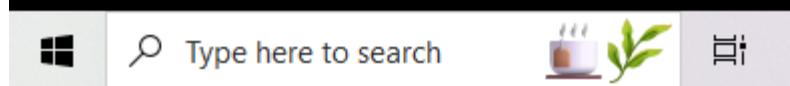
-  Remove the requirement versions.

ubuntu@ip-172-31-23-62: ~/hearing

```
click
colorama
Flask
itsdangerous
joblib
MarkupSafe
numpy
pandas
python-dateutil
pytz
scikit-learn
scipy
six
sklearn
threadpoolctl
werkzeug
unicorn
```

333

"requirements.txt" 17L, 149B



```
        st.warning("Please write Topic Name to search")
from flask import Flask

# Define the Flask application
app = Flask(__name__)

# Define a simple route
@app.route('/')
def home():
    return "welcome to InNews!"

# Define the run function
def run():
    app.run(host='0.0.0.0', port=8070, debug=True)

# Entry point of the script
if __name__ == "__main__":
    run()

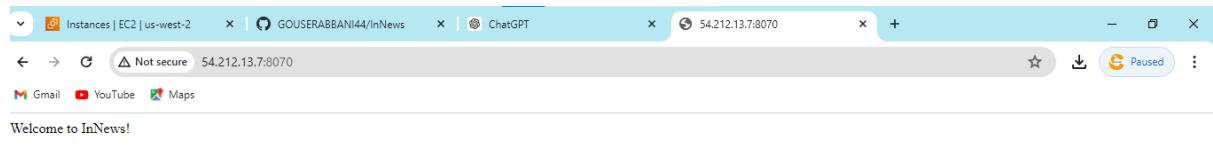
"App.py" 148L, 5191B
```

148 ,0-1 Bot



⊕ Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >



Step 14: (Hearing)

⊕ Using these command as follows
< sudo git clone (repo name) >

⊕ Change the name

< cd (repo name) >

⊕ Install requirements packages using these command as follows

< pip3 install -r requirements.txt >

⊕ After that change the port number open the file as

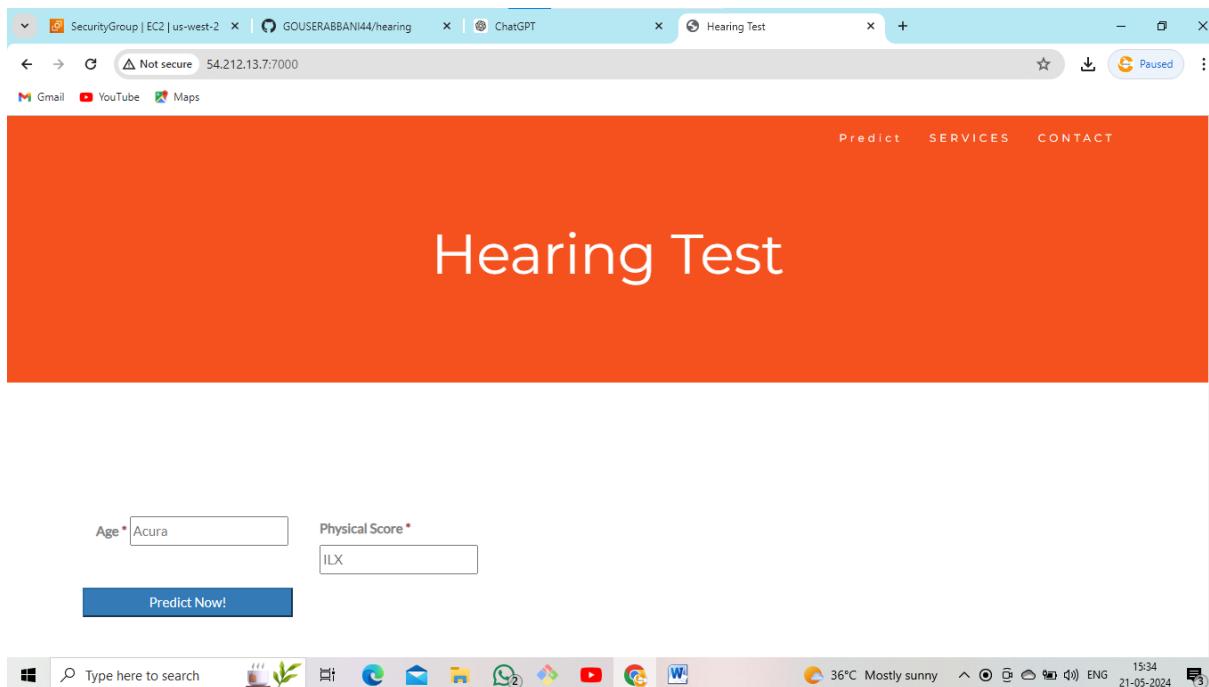
< sudo vi app.py >

⊕ Run the server using these command as

< python3 app.py >

⊕ Go to EC2 instance, Edit inbound rules and save it, copy public ip and paste to Google.

< public ip : port number >

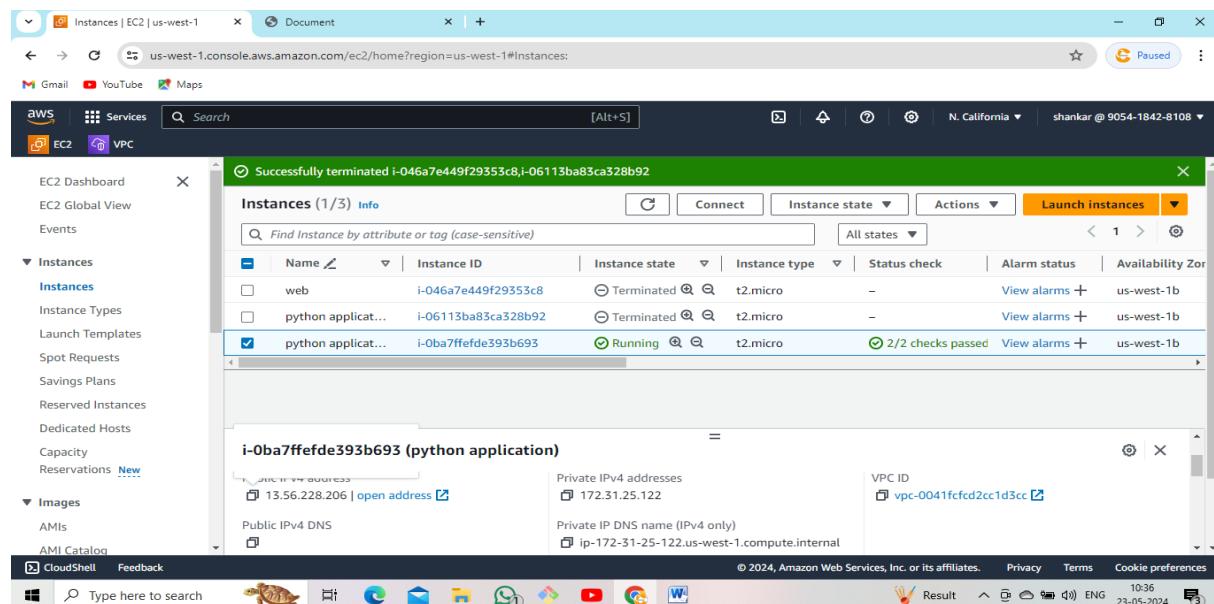


2. Build and deploy python applications along with EC2 instance (user data)?

- + Go to EC2 instance.
- + Edit inbound rules (HTTP, HTTPS, Allow all traffic).
- + Go to advanced details.
- + Opened user data. Upload the script here.

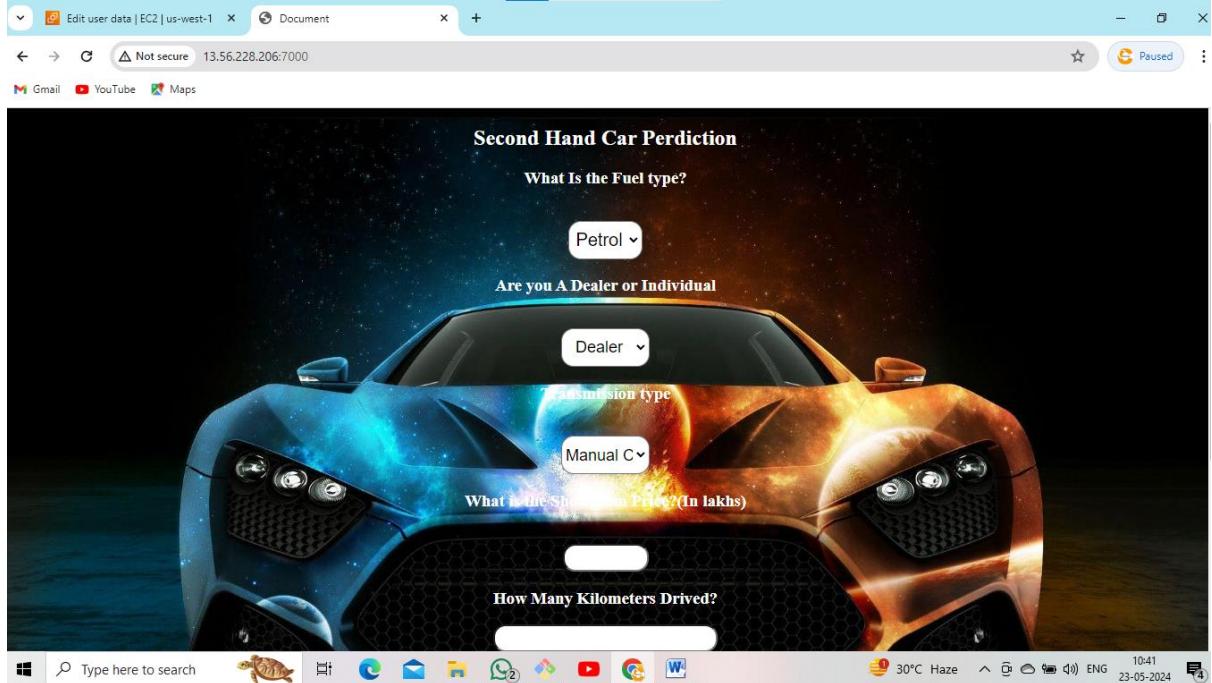
```
#!/bin/bash
sudo apt-get update -y
sudo apt-get install -y python3 python3-pip git
git clone https://github.com/kaasu-pavani/car-prediction.git /home/ubuntu/car-prediction
cd /home/ubuntu/car-prediction
pip3 install -r requirements.txt
```

- + Launch EC2 instance.



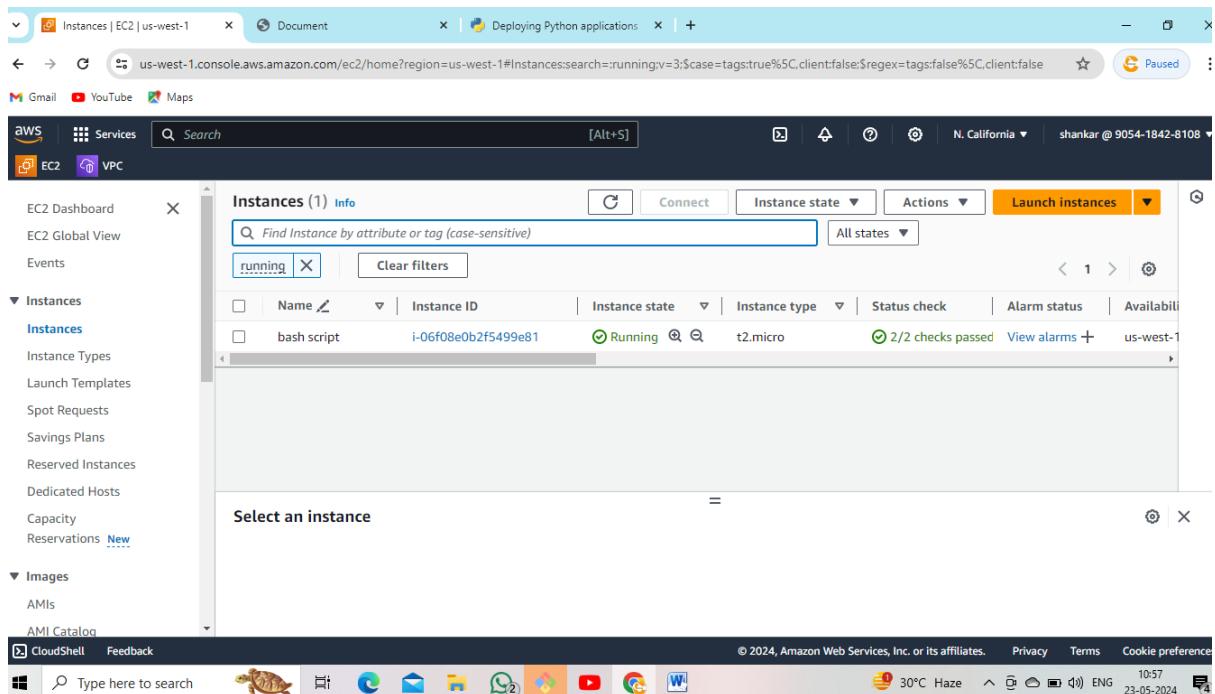
Copy public ip and paste it google.

< copy public ip : port number >



3. Build and deploy python applications in Bash scripting?

- + Go to EC2 instance.
- + Select AMI Ubuntu server.
- + Edit inbound rules (HTTP, HTTPS, Allow all traffic).
- + Launch EC2 instance.



- + Go to Git Bash.
- + Update Ubuntu machine.
`< sudo apt update >`
- + Full upgrade the machine.
`< sudo apt-get full-upgrade -y >`
- + Install required packages.

< sudo apt-get install python3 pip >

➡ Created one file and upload the script.

< sudo vi data.sh >

```
ubuntu@ip-172-31-22-105: ~
#!/bin/bash
sudo apt-get update -y
sudo apt-get install -y python3 python3-pip git
git clone https://github.com/kaasu-pavani/car-prediction.git /home/ubuntu/car-prediction
cd /home/ubuntu/car-prediction
pip3 install -r requirements.txt
python3 app.py
screen -m python3 app.py
screen -m -d python3 app.py
~
```

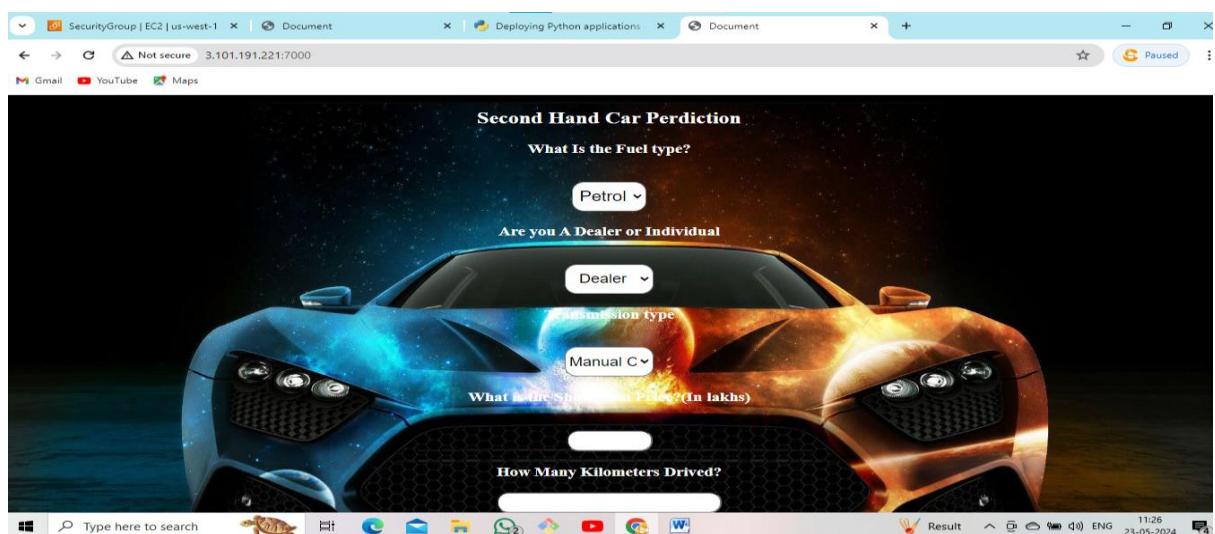
➡ After that given the permissions.

< sudo chmod +x data.sh >

< ./data.sh >

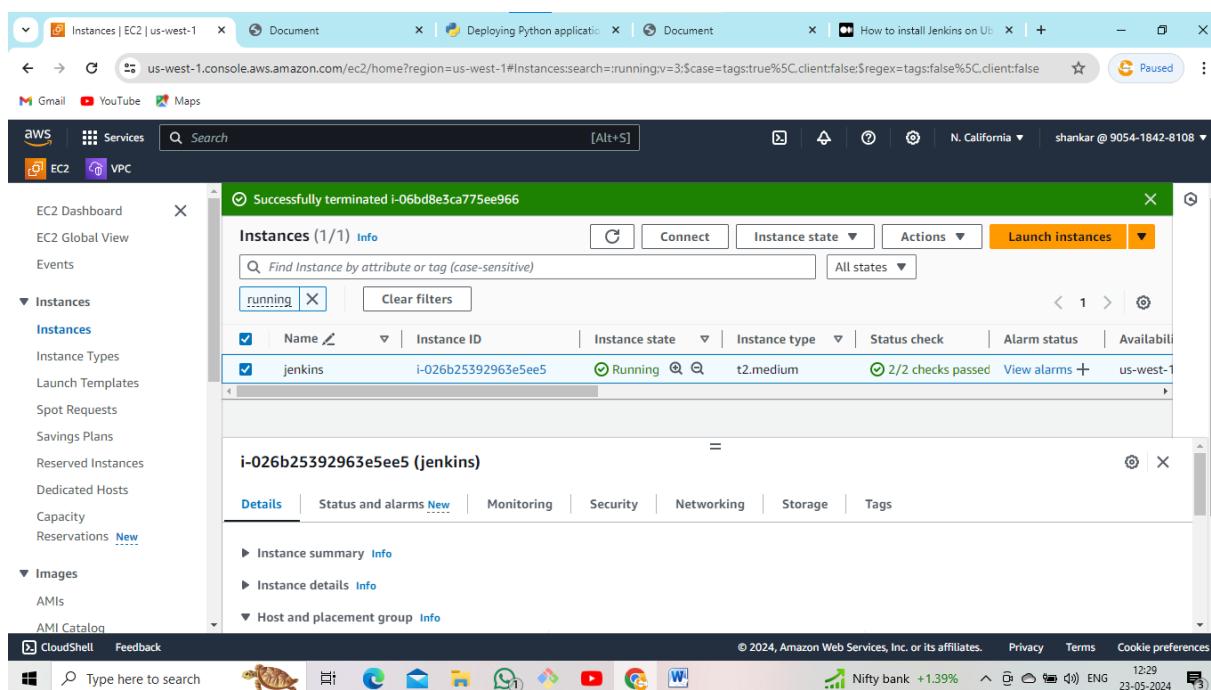
➡ Go to EC2 instance copy public ip and paste it Google.

< copy public ip : port number >



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

- + Go to EC2 instance
- + Select AMI Ubuntu server.
- + Select t3.medium.
- + Edit inbound rules.(HTTP,8080,application port number or all traffic)
- + Launch EC2 instance.



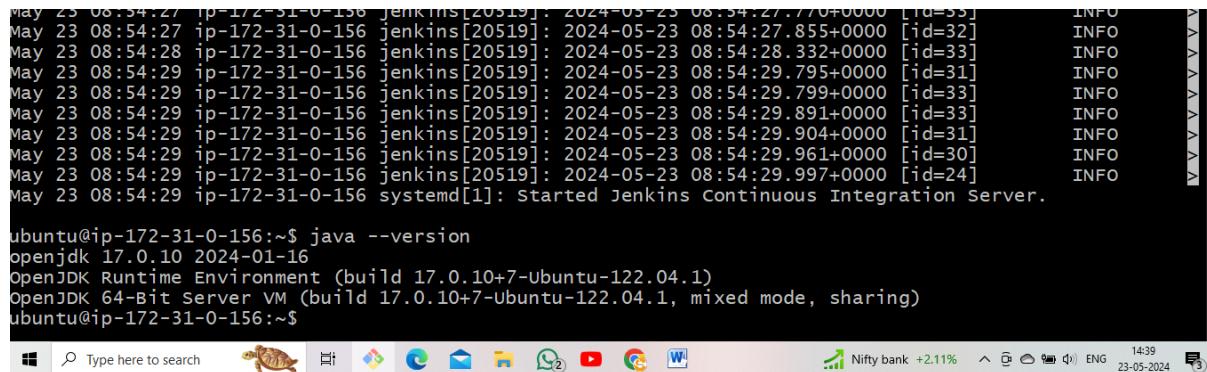
- + Go to Git Bash.
- + Update the machine.
`< sudo apt update >`

- After that install git using these command following below.

```
< sudo yum install git -y >
```

- Install java for using these command below

```
<>
```



A screenshot of a Windows desktop taskbar. The taskbar shows various pinned icons including File Explorer, Edge, Mail, Photos, and others. In the center, there is a terminal window displaying Jenkins logs and Java version information. The logs show Jenkins starting up and connecting to a database. The Java version output shows OpenJDK 17.0.10 running on an Ubuntu 20.04.1 system. The system tray shows battery level at 21.11%, network connection, and system status.

```
May 23 08:54:27 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:27.770+0000 [id=55] INFO
May 23 08:54:27 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:27.855+0000 [id=32] INFO
May 23 08:54:28 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:28.332+0000 [id=33] INFO
May 23 08:54:29 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:29.795+0000 [id=31] INFO
May 23 08:54:29 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:29.799+0000 [id=33] INFO
May 23 08:54:29 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:29.891+0000 [id=33] INFO
May 23 08:54:29 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:29.904+0000 [id=31] INFO
May 23 08:54:29 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:29.961+0000 [id=30] INFO
May 23 08:54:29 ip-172-31-0-156 jenkins[20519]: 2024-05-23 08:54:29.997+0000 [id=24] INFO
May 23 08:54:29 ip-172-31-0-156 systemd[1]: Started Jenkins Continuous Integration Server.

ubuntu@ip-172-31-0-156:~$ java --version
openjdk 17.0.10 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-122.04.1, mixed mode, sharing)
ubuntu@ip-172-31-0-156:~$
```

- After that install Jenkins using these commands as below

- Go to Google, search install Jenkins in Ubuntu server.

- ❖ `sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key`
- ❖ `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`
- ❖ `sudo apt-get update`
- ❖ `sudo apt-get install jenkins`

⊕ after that start jenkins for using these commands as below

< sudo systemctl start jenkins >

< sudo systemctl enable jenkins >

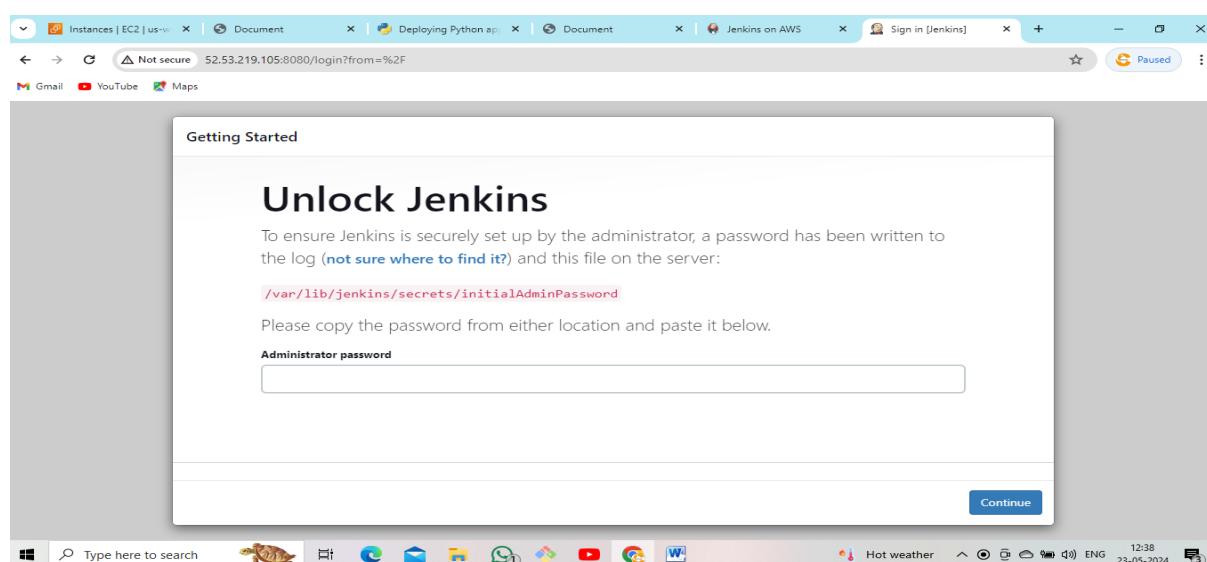
< sudo systemctl status jenkins >

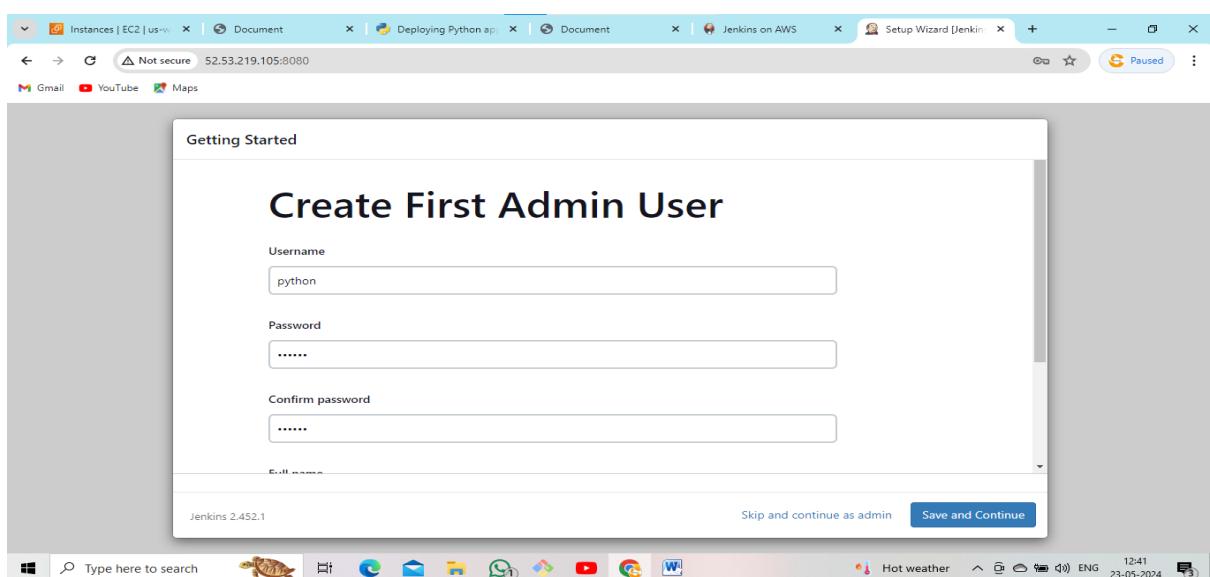
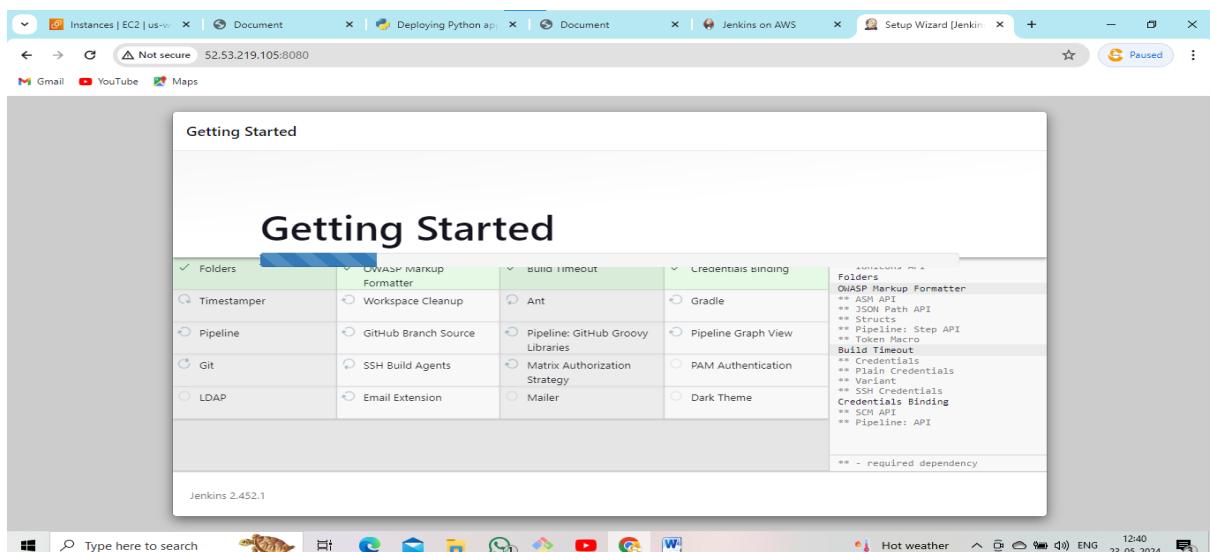
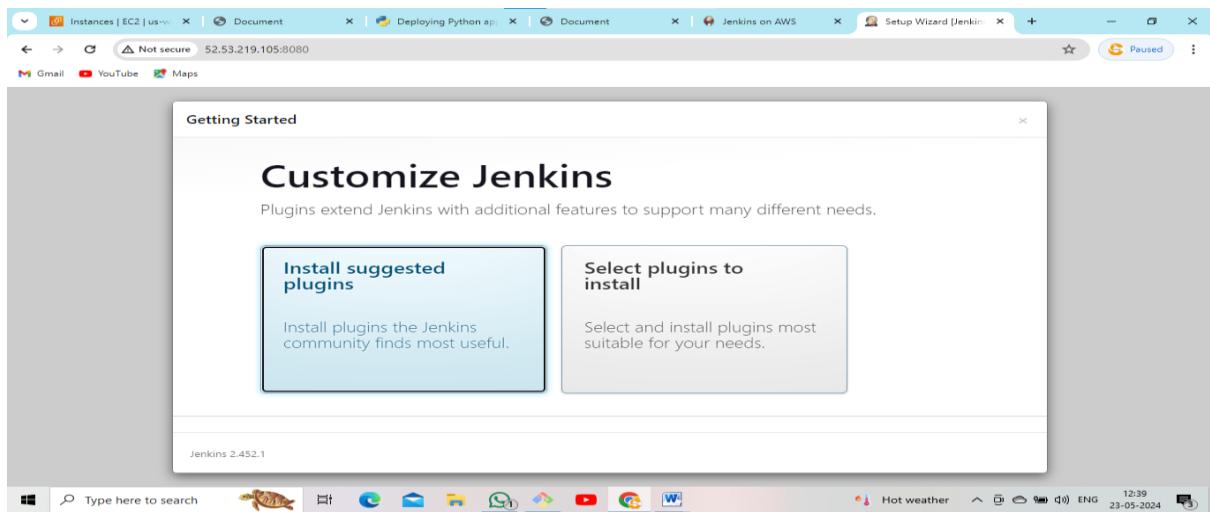
```
sudo systemctl enable jenkins
Created symlink from /etc/systemd/system/multi-user.target.wants/jenkins.service to /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-12-123 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-12-123 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
  Active: active (running) since Thu 2024-05-23 07:07:32 UTC; 9s ago
    Main PID: 8875 (java)
      CGroup: /system.slice/jenkins.service
              └─8875 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=...

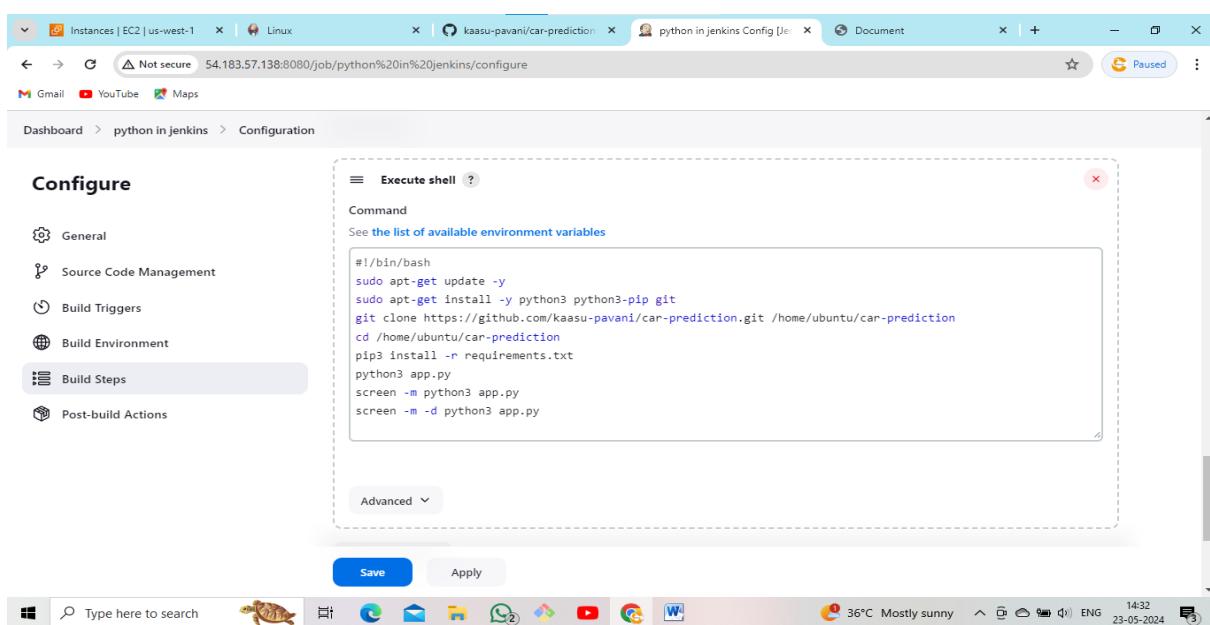
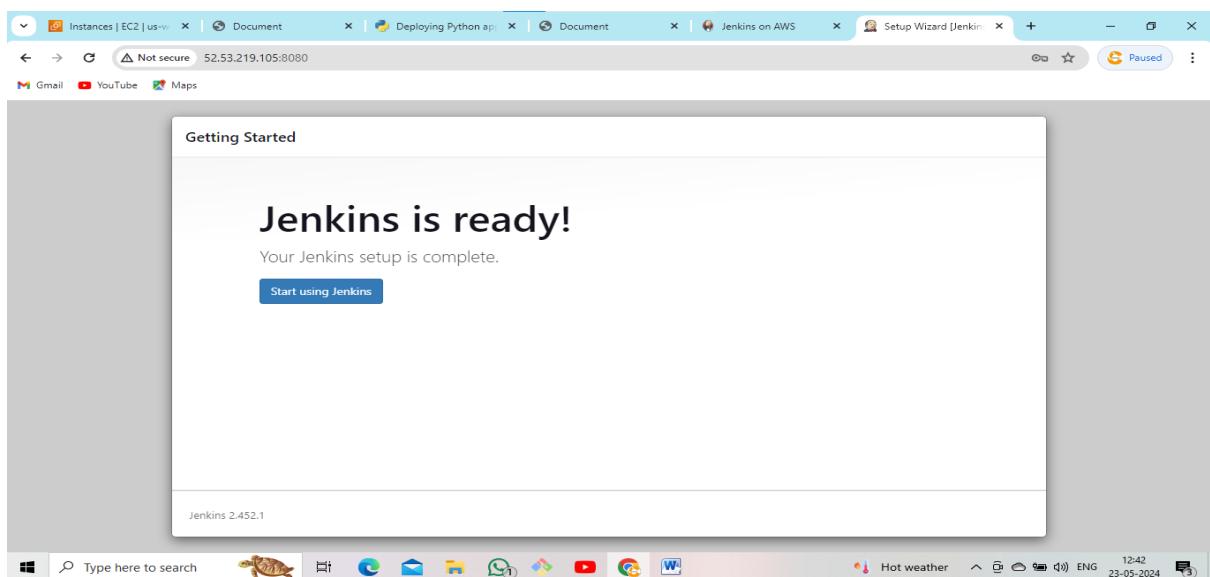
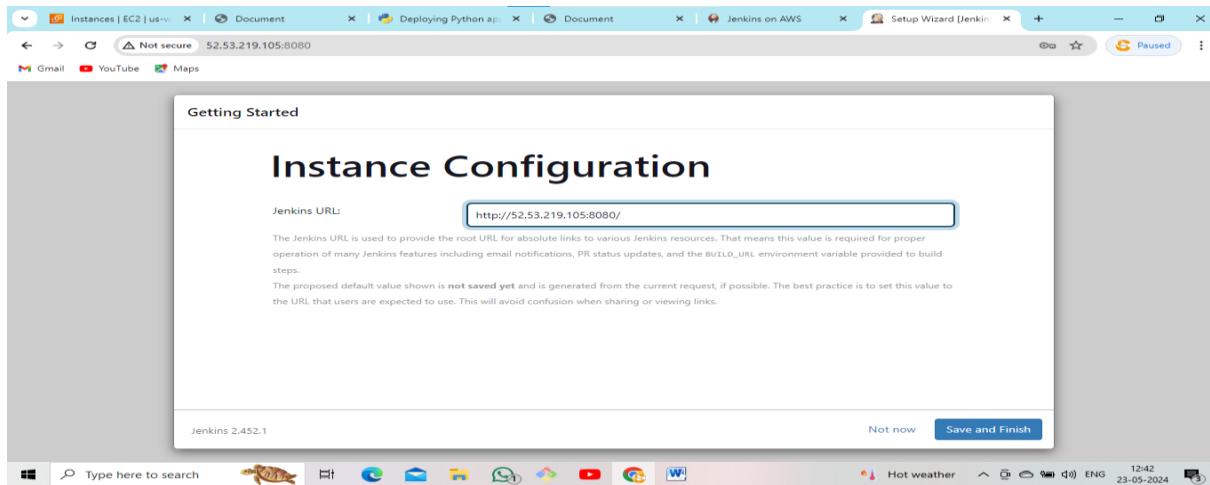
May 23 07:07:16 ip-172-31-12-123.us-west-1.compute.internal jenkins[8875]: 88957be0318346ed97373f2ee...
May 23 07:07:16 ip-172-31-12-123.us-west-1.compute.internal jenkins[8875]: This may also be found at...
May 23 07:07:16 ip-172-31-12-123.us-west-1.compute.internal jenkins[8875]: ****
May 23 07:07:32 ip-172-31-12-123.us-west-1.compute.internal jenkins[8875]: 2024-05-23 07:07:32.520+0...
May 23 07:07:32 ip-172-31-12-123.us-west-1.compute.internal jenkins[8875]: 2024-05-23 07:07:32.539+0...
May 23 07:07:32 ip-172-31-12-123.us-west-1.compute.internal systemd[1]: Started Jenkins continuous ...
May 23 07:07:32 ip-172-31-12-123.us-west-1.compute.internal jenkins[8875]: 2024-05-23 07:07:32.868+0...
```

⊕ After that go to EC2 instance copy public ip and jenkins port number paste it Google.

< copy public ip : 8080 >







```

requirement already satisfied: pytz==2022.2.1 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 10)) (2.8.2)
Requirement already satisfied: pytz==2022.2.1 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 11)) (2022.2.1)
Requirement already satisfied: scikit-learn==1.1.2 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 12)) (1.1.2)
Requirement already satisfied: scipy==1.9.1 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 13)) (1.9.1)
Requirement already satisfied: six==1.16.0 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 14)) (1.16.0)
Requirement already satisfied: sklearn==0.0 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 15)) (0.0)
Requirement already satisfied: threadpoolctl==3.1.0 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 16)) (3.1.0)
Requirement already satisfied: Werkzeug==2.2.2 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 17)) (2.2.2)
Requirement already satisfied: gunicorn==20.0.4 in /var/lib/jenkins/.local/lib/python3.10/site-packages (from -r requirements.txt (line 18)) (20.0.4)
Requirement already satisfied: setuptools>=3.0 in /usr/lib/python3/dist-packages (from gunicorn==20.0.4->-r requirements.txt (line 18)) (59.6.0)
Finished: SUCCESS

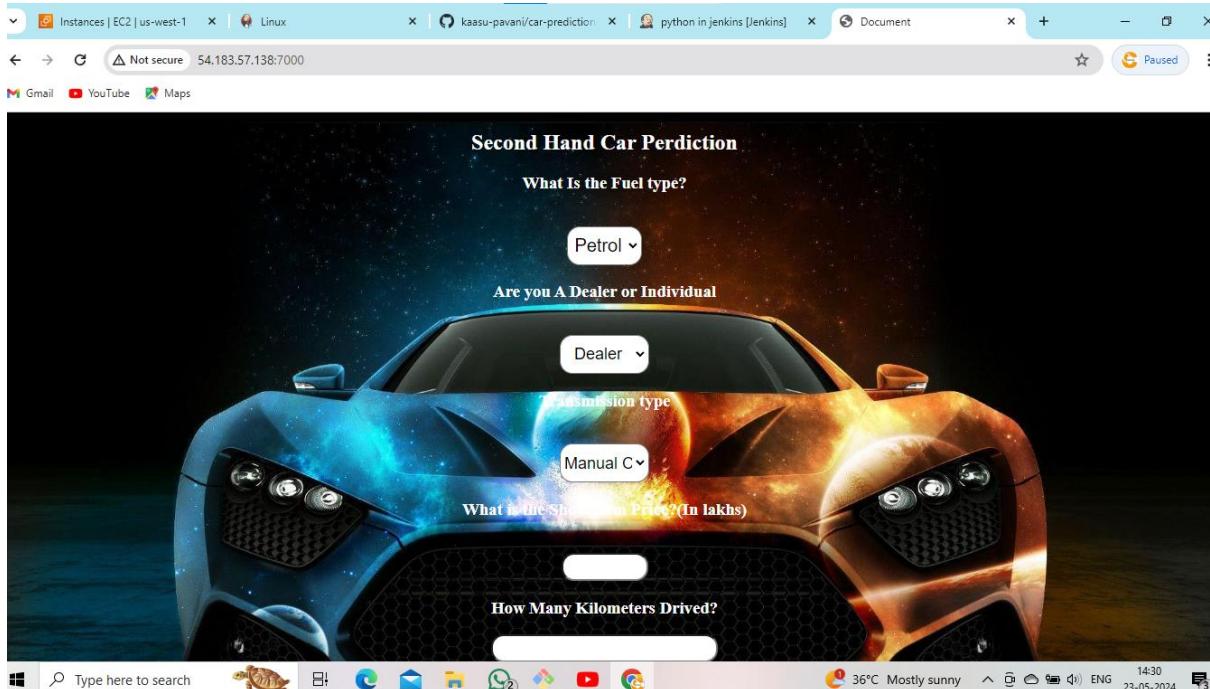
```

REST API Jenkins 2.452.1

Type here to search

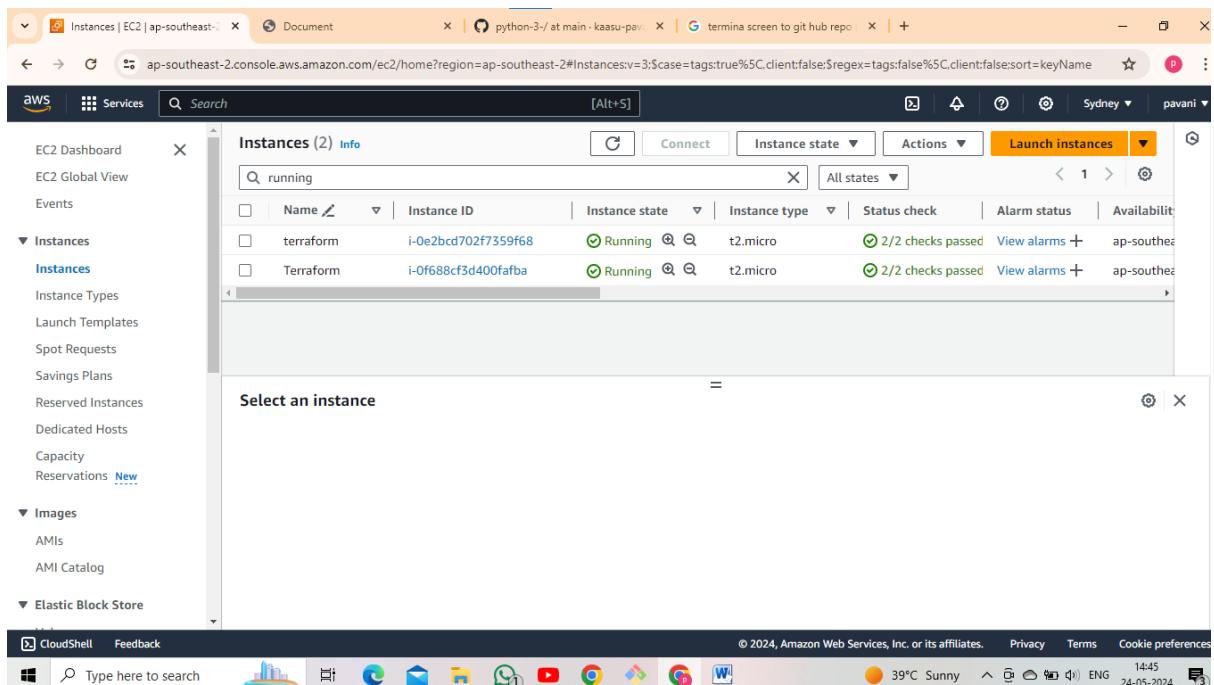
SENSEX +1.49% 14:36 23-05-2024 ENG

Go to EC2 instance copy public ip and application port number paste it Google.



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

- Go to EC2 instance.
- Select AMI Ubuntu server.
- Select t2.micro.
- Edit inbound rules.
- Launch EC2 instance.



- Go to Git Bash.
- Update Ubuntu machine.
`< sudo apt update >`
- Full upgrade the machine.

< sudo apt-get full-upgrade -y >

▪+ Install terraform using these commands as

```
❖ wget -O- https://apt.releases.hashicorp.com/gpg |  
    sudo gpg --dearmor -o/usr/share/keyrings/hashicorp-  
    archive-keyring.gpg  
    echo "deb [signed-by=/usr/share/keyrings/hashicorp-  
    archive-keyring.gpg] https://apt.releases.hashicorp.com  
    ${lsb_release -cs} main" | sudo tee  
    /etc/apt/sources.list.d/hashicorp.list  
    sudo apt update && sudo apt install terraform
```

```
2 packages can be upgraded. Run 'apt list --upgradable' to see them.  
ubuntu@ip-172-31-44-255:~$ sudo apt-get full-upgrade -y  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
The following packages have been kept back:  
  python3-update-manager update-manager-core  
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.  
ubuntu@ip-172-31-44-255:~$ terraform --version  
Terraform v1.8.4  
on linux_amd64  
ubuntu@ip-172-31-44-255:~$
```



▪+ After that create two files. one is data.sh and second one is main.tf.

```
#!/bin/bash  
sudo apt-get update -y  
sudo apt-get install -y python3 python3-pip git  
git clone https://github.com/kaasu-pavani/car-prediction.git /home/ubuntu/car-prediction  
cd /home/ubuntu/car-prediction  
pip3 install -r requirements.txt  
python3 app.py  
screen -d python3 app.py  
screen -d -m python3 app.py  
~  
~  
~  
~  
~  
~
```

- Above screen is data.sh.
- After that create main.tf file.

```
provider "aws" {  
    region = "ap-northeast-3"  
}  
  
# Defining CIDR Block for VPC  
variable "vpc_cidr" {  
    default = "10.0.0.0/16"  
}  
  
# Defining CIDR Block for Subnet  
variable "subnet_cidr" {  
    default = "10.0.1.0/24"  
}  
  
# Creating the VPC  
resource "aws_vpc" "main" {  
    cidr_block = var.vpc_cidr  
    instance_tenancy = "default"  
    tags = {  
        Name = "vpc"  
    }  
}  
  
# Creating the Subnet
```

```
resource "aws_subnet" "main" {
    vpc_id = aws_vpc.main.id
    cidr_block = var.subnet_cidr
    availability_zone = "ap-northeast-3a"
    tags = {
        Name = "subnet"
    }
}

# Creating the Internet Gateway
resource "aws_internet_gateway" "main" {
    vpc_id = aws_vpc.main.id
}

# Creating the Route Table
resource "aws_route_table" "main" {
    vpc_id = aws_vpc.main.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.main.id
    }
    tags = {
        Name = "Route to internet"
    }
}

# Associating Route Table with Subnet
resource "aws_route_table_association" "subnet_assoc" {
```

```
subnet_id = aws_subnet.main.id
route_table_id = aws_route_table.main.id
}

# Creating Security Group
resource "aws_security_group" "py_sg" {
    vpc_id = aws_vpc.main.id

    # Inbound Rules
    ingress {
        from_port  = 80
        to_port   = 80
        protocol  = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port = 7000
        to_port  = 7000
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port = 443
        to_port  = 443
        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"]
}

# Outbound Rules

egress {
    from_port  = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
}

tags = {
    Name = "Web SG"
}

}

}

# Creating EC2 Instance

resource "aws_instance" "Terraform" {
    ami      = "ami-058a260ffc06d25e2"
    instance_type = "t2.micro"
    key_name  = "hm"
    vpc_security_group_id = [aws_security_group.py_sg.id]
    subnet_id = aws_subnet.main.id
```

```
associate_public_ip_address = true
user_data = file ("userdata.sh")
tags = {
    Name = "Terraform"
}
#
# Output the public IP of the instance
output "public_ip" {
    value = aws_instance.Terraform.public_ip
}
#
# Variable for AWS region
variable "region" {
    default = "ap-northeast-3"
}
```

➡ After that apply the terraform commands.

- ❖ Terraform init
- ❖ Terraform fmt
- ❖ Terraform validate
- ❖ Terraform plan
- ❖ Terraform apply

```
Using previously installed hashicorp/aws v3.51.0
```

```
Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.
```

```
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.
```

```
commands will detect it and remind you to do so if necessary.
```

```
ubuntu@ip-172-31-44-255:~$ terraform validate  
Success! The configuration is valid.
```

```
ubuntu@ip-172-31-44-255:~$ terraform plan  
aws_vpc.main: Refreshing state... [id=vpc-0af07b606aba4681b]  
aws_security_group.py_sg: Refreshing state... [id=sg-07f2a48efd5109634]  
aws_internet_gateway.main: Refreshing state... [id=igw-09badb5065732dc59]  
aws_subnet.main: Refreshing state... [id=subnet-0e4046df03afc7fb3]  
aws_route_table.main: Refreshing state... [id=rtb-0adb09b1ce1887687]  
aws_instance.Terraform: Refreshing state... [id=i-0f688cf3d400fafba]  
aws_route_table_association.subnet_assoc: Refreshing state... [id=rtbassoc-084e07b83b49be0c4]
```

```
No changes. Your infrastructure matches the configuration.
```

```
aws_security_group.py_sg: Refreshing state... [id=sg-07f2a48efd5109634]  
aws_route_table.main: Refreshing state... [id=rtb-0adb09b1ce1887687]  
aws_instance.Terraform: Refreshing state... [id=i-0f688cf3d400fafba]  
aws_route_table_association.subnet_assoc: Refreshing state... [id=rtbassoc-084e07b83b49be0c4]
```

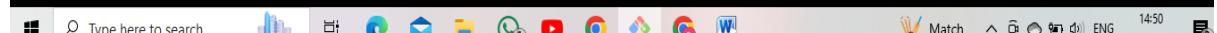
```
No changes. Your infrastructure matches the configuration.
```

```
Terraform has compared your real infrastructure against your configuration and found no differences,  
so no changes are needed.
```

```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

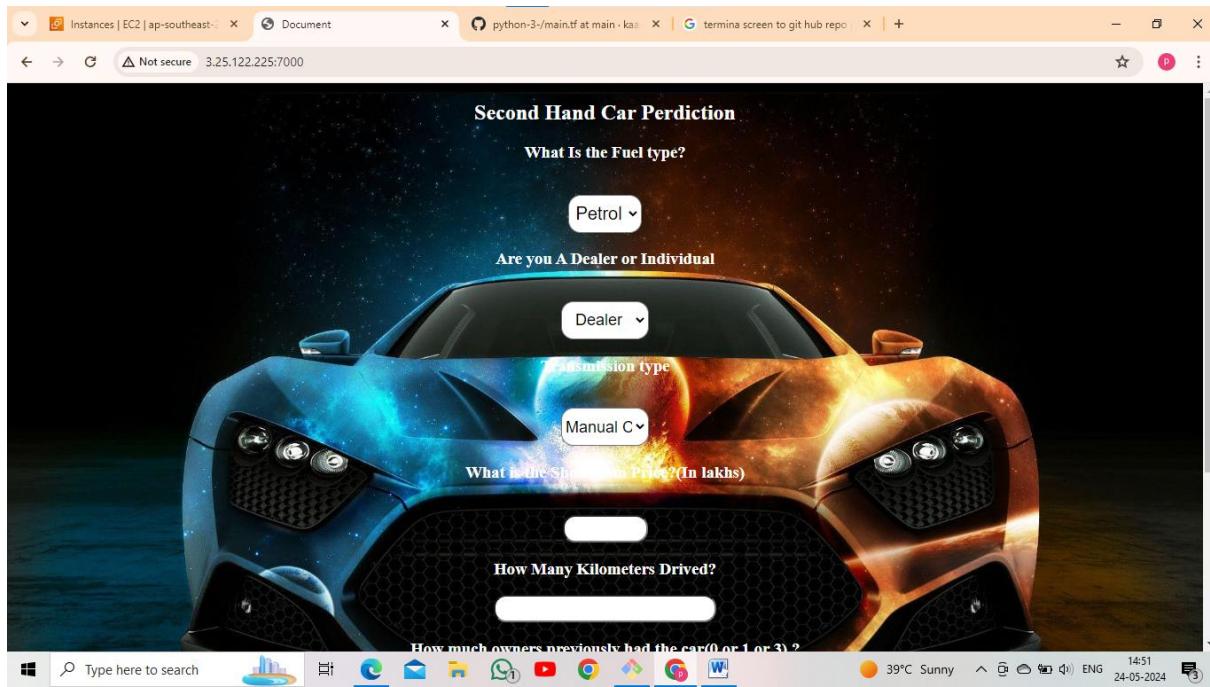
```
Outputs:
```

```
public_ip = "3.25.122.225"  
ubuntu@ip-172-31-44-255:~$
```



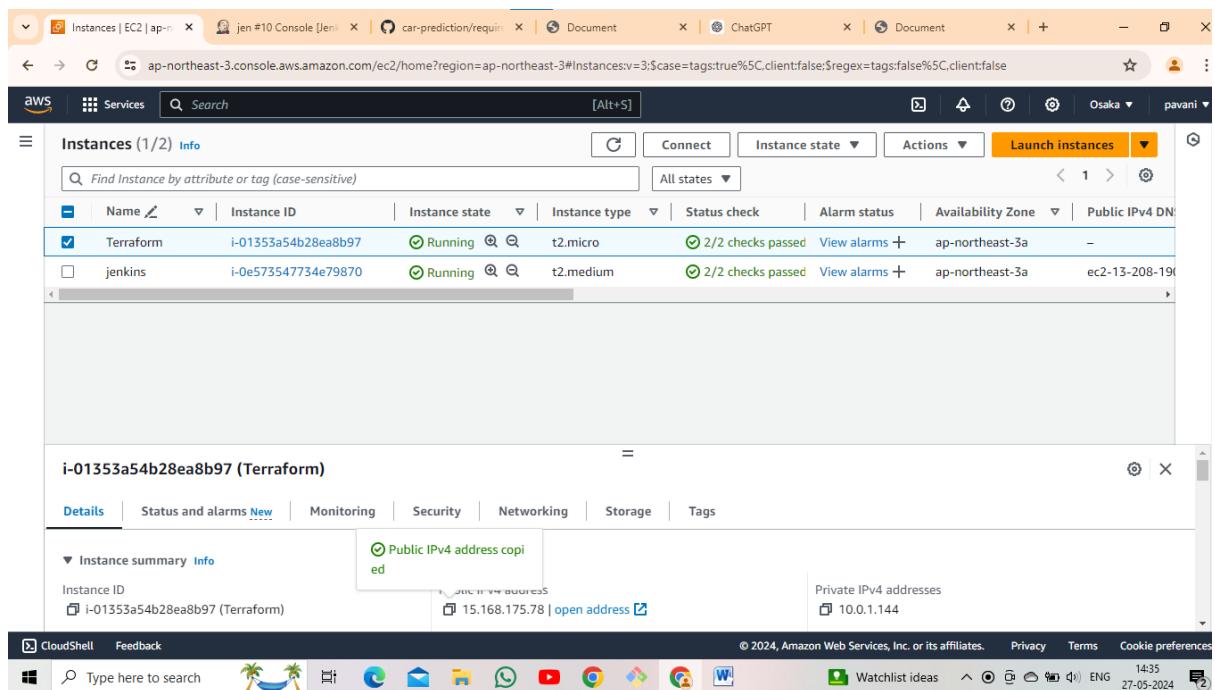
➡ After that copy public ip and application port number
paste it Google.

< copy public ip : port number >



6. Build and deploy python applications with Git, github, Jenkins and Terraform?

- Go to EC2 instance.
- Select AMI Ubuntu server.
- Select t3.medium.
- Edit inbound rules.
- Launch EC2 instance.

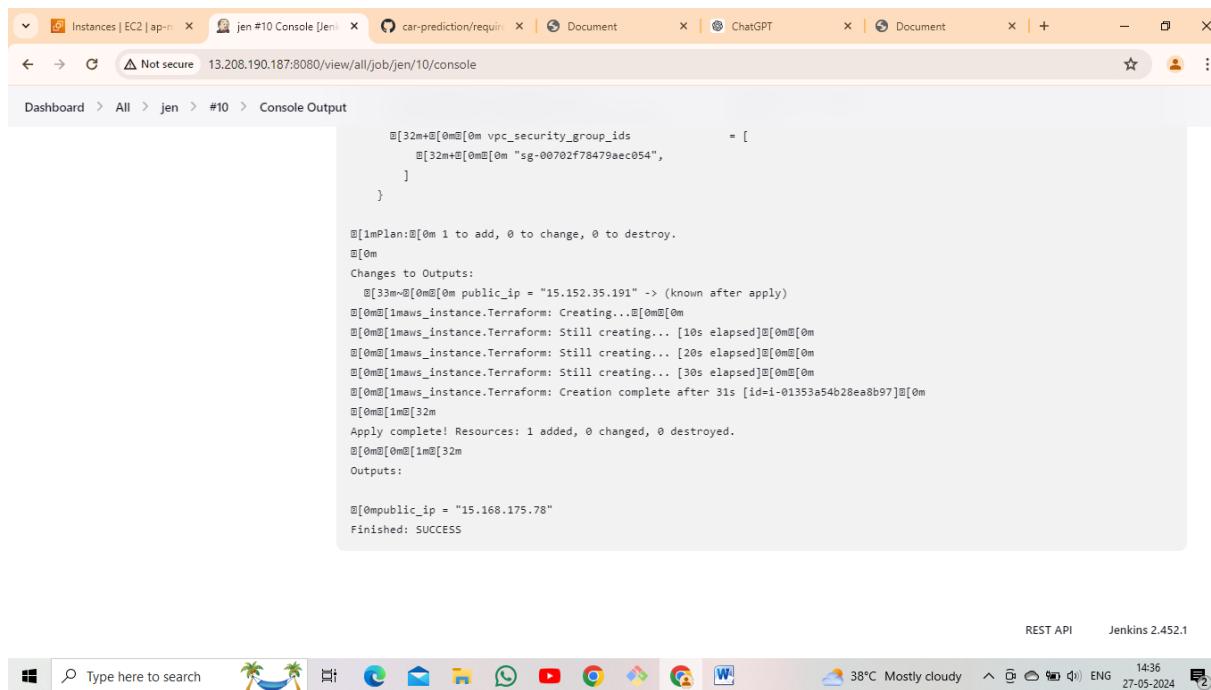


- Go to Git Bash.
- Update the machine.
`< sudo yum update >`
- After that install java.
`< sudo yum -y install java* >`

⊕ Install Jenkins.

NOTE: Jenkins installation is above steps (4th step).

```
#!/bin/bash
# Install necessary packages
sudo yum install -y yum-utils shadow-utils
# Add HashiCorp repository
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
# Install Terraform
sudo yum -y install terraform
# Update system packages
sudo yum update -y
# Set AWS credentials (make sure to replace with your actual credentials)
export AWS_ACCESS_KEY_ID="AKIAXYKJVTWSM2SMR3SC"
export AWS_SECRET_ACCESS_KEY="zGJqshXpDHf+85g5C8sGjwOcRWzfaKXkDfYMGY/4"
# Initialize Terraform in the project directory
terraform init
# Validate Terraform configuration
terraform validate
# Plan Terraform deployment
terraform plan
# Apply Terraform changes
```



The screenshot shows a browser window with multiple tabs. The active tab is 'jen #10 Console [jen]' showing Jenkins job output. The output displays Terraform command-line interface (CLI) logs:

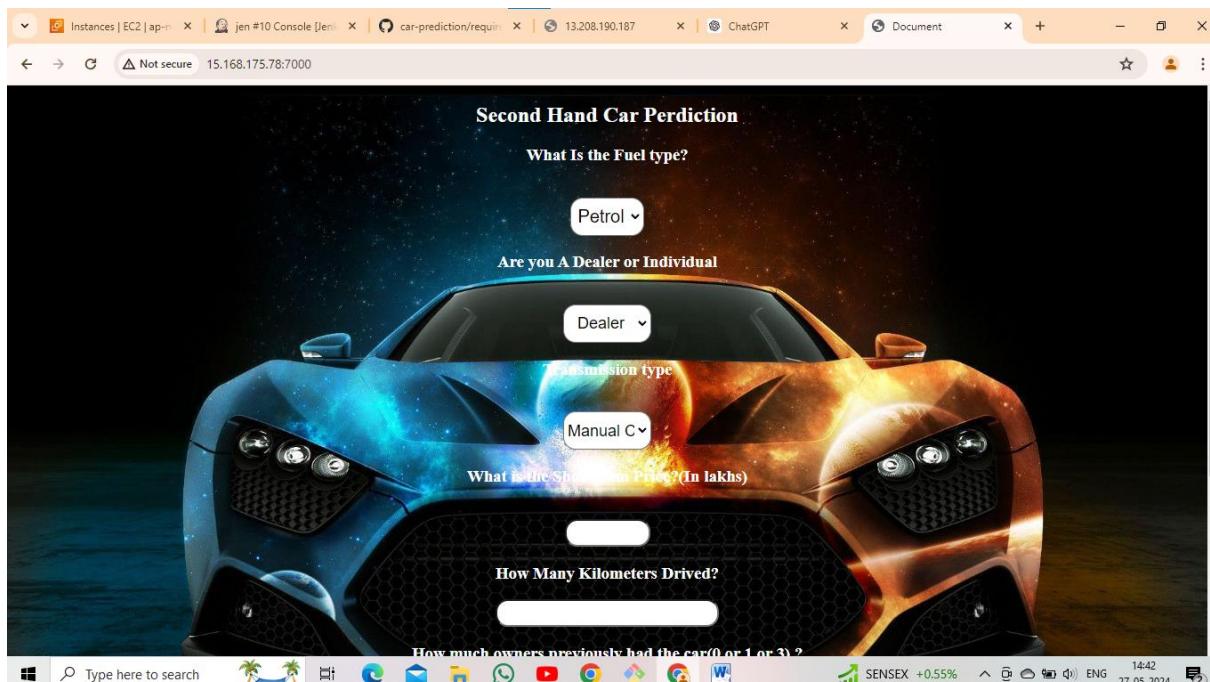
```
Dashboard > All > jen > #10 > Console Output
[32m+@[0m@[0m vpc_security_group_ids = [
  @[32m+@[0m@[0m "sg-00702f78479aec054",
]
}

@[1mPlan:@[0m 1 to add, 0 to change, 0 to destroy.
@[0m
Changes to Outputs:
@[33m~@[0m@[0m public_ip = "15.152.35.191" -> (known after apply)
@[0m@[1maws_instance.Terraform: Creating...@[0m@[0m
@[0m@[1maws_instance.Terraform: Still creating... [10s elapsed]@[0m@[0m
@[0m@[1maws_instance.Terraform: Still creating... [20s elapsed]@[0m@[0m
@[0m@[1maws_instance.Terraform: Still creating... [30s elapsed]@[0m@[0m
@[0m@[1maws_instance.Terraform: Creation complete after 31s [id:i-01353a54b28ea8b97]@[0m
@[0m@[1m@[32m
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
@[0m@[0m@[1m@[32m
Outputs:

@[0mpublic_ip = "15.168.175.78"
Finished: SUCCESS
```

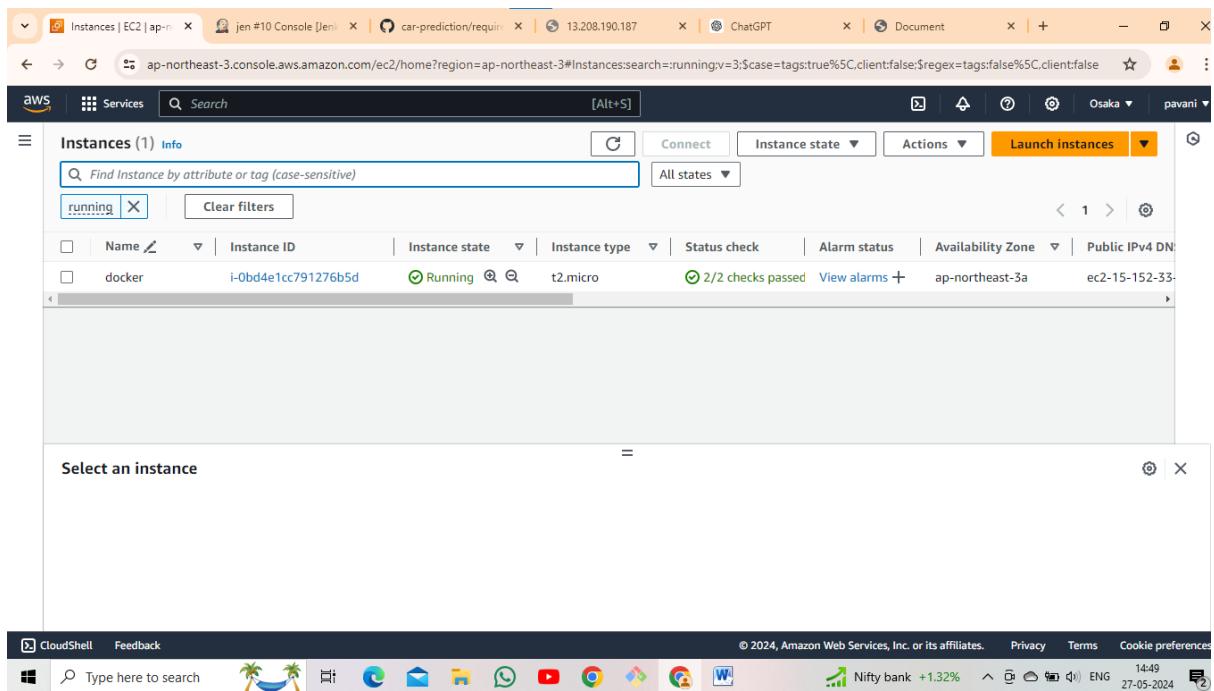
⊕ Go to EC2 instance copy public ip and application port number paste it Google.

< copy public ip : port number >



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?

- + Go to EC2 instance.
- + Select AMI linux.
- + Edit inbound rules
- + Launch EC2 instance.



- + Go to Git Bash.
- + Update the machine using these following command as

< sudo yum update >

 Install docker using these command as

```
< sudo yum -y install docker >
```

verifying : docker-20.10.25-1.amzn2.0.4.x86_64
5/5

Installed:
docker.x86_64 0:20.10.25-1.amzn2.0.4

Dependency Installed:
containerd.x86_64 0:1.7.11-1.amzn2.0.1
pigz.x86_64 0:2.3.4-1.amzn2.0.1
libcgroup.x86_64 0:0.41-21.amzn2
runc.x86_64 0:1.1.11-1.amzn2

Complete!

```
[ec2-user@ip-172-31-44-39 ~]$ docker --version
Docker version 20.10.25, build b82b9f3
[ec2-user@ip-172-31-44-39 ~]$
```

 Docker enable, start, status using these commands as

< sudo systemctl start docker >

```
< sudo systemctl enable docker >
```

```
< sudo systemctl status docker >
```

```
[ec2-user@ip-172-31-44-39 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-05-27 09:21:51 UTC; 18s ago
     Docs: https://docs.docker.com
Main PID: 3470 (dockerd)
   CGroup: /system.slice/docker.service
           └─3470 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-u...
May 27 09:21:50 ip-172-31-44-39.ap-northeast-3.compute.internal dockerd[3470]: time="2024-05-27T09:2...
May 27 09:21:51 ip-172-31-44-39.ap-northeast-3.compute.internal dockerd[3470]: time="2024-05-27T09:2...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-44-39 ~]$
```

■ Install docker-compose file

■ Create file using these command as

< vi Dockerfile >

```
FROM amazonlinux:latest
RUN yum update -y
RUN yum install -y git python3-pip python3-setuptools
RUN git clone https://github.com/kaasu-pavani/car-prediction.git
WORKDIR /car-prediction
RUN pip3 install --no-cache-dir -r requirements.txt
EXPOSE 7000
CMD ["python3","./app.py"]
~
```

■ Give the sudo permissions

< sudo chmod 666 /var/run/docker.sock >

■ Docker build using these command as

< sudo docker build -t kasuu20/ car-
prediction -app . >

■ Docker push using these command as

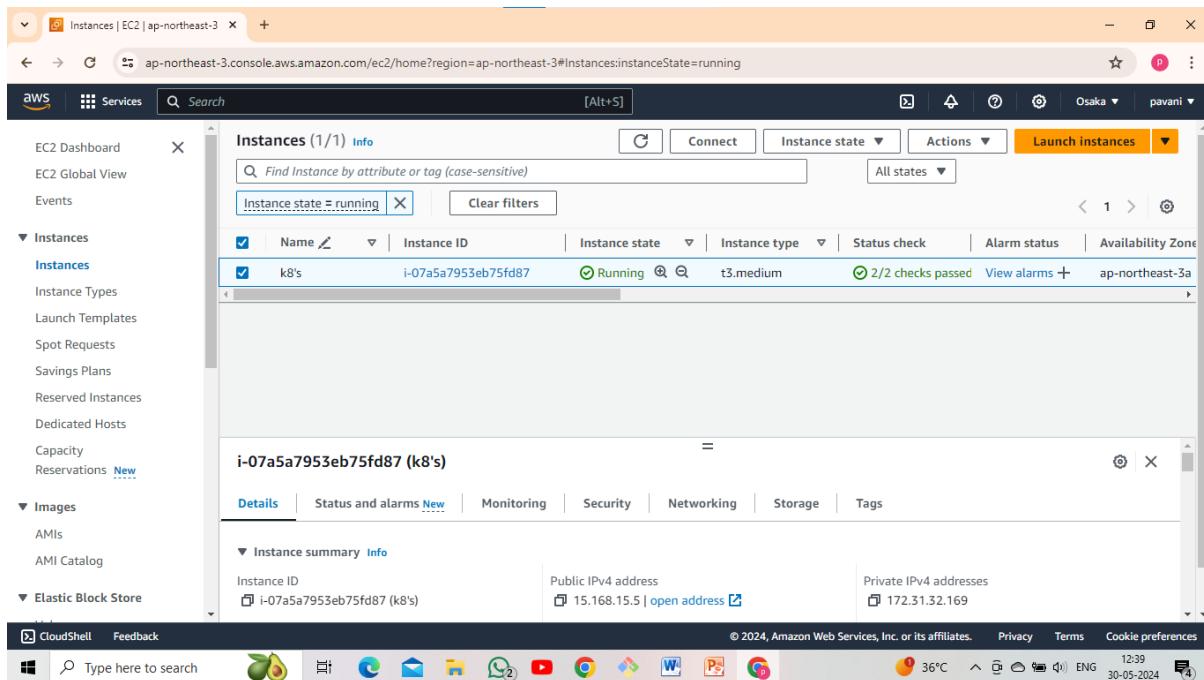
< sudo docker push kasuu20/ car-prediction >

Screenshot of the Docker Hub website showing a repository named "kasuu20 / car-prediction-app". The repository is public, contains one image, and was last pushed about 19 hours ago. A search bar at the top right allows users to search for Docker Hub content.

The interface includes a "Create repository" button and a sidebar on the right with a "Create An Organization" section, which provides instructions for creating and managing users and repositories.

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

- Go to EC2 instance.
- Select AMI linux.
- Select t3.medium.
- Edit inbound rules.
- Launch EC2 instance.



- Go to Git Bash.
- Install kops. (kops commands search for Google)

Install using native package management

Debian-based distributions

Red Hat-based distributions

SUSE-based distributions

1. Add the Kubernetes `yum` repository. If you want to use Kubernetes version different than v1.30, replace v1.30 with the desired minor version in the command below.

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repo/repodata/repomd.xml.
EOF
```

Note:

After that install kubectl using these command as

`< sudo yum install -y kubectl >`

Check the kubectl using these command as

`< kubectl version --client>`

```
Running transaction test
Transaction test succeeded
Running transaction
  Installing : kubectl-1.30.1-150500.1.1.x86_64                               1/1
  Verifying  : kubectl-1.30.1-150500.1.1.x86_64                               1/1

Installed:
  kubectl.x86_64 0:1.30.1-150500.1.1

Complete!
[ec2-user@ip-172-31-47-62 ~]$ snap install kubectl --classic
-bash: snap: command not found
[ec2-user@ip-172-31-47-62 ~]$ kubectl version --client
Client Version: v1.30.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
[ec2-user@ip-172-31-47-62 ~]$
```



⊕ After that install kops.

GitHub Releases

Linux macOS Windows

Linux

```
curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest)kops
chmod +x kops
sudo mv kops /usr/local/bin/kops
```

macOS

```
curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest)kops
chmod +x kops
sudo mv kops /usr/local/bin/kops
```

⊕ Create two files.

1.Deployment.yaml

2.service.yaml

```
apiVersion: app/v1
kind: Deployment
metadata:
  name: car-prediction-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: car-prediction
  template:
    metadata:
      labels:
        app: car-prediction
    spec:
      containers:
        - name: car-prediction-container
          image: kasuu20/car-prediction-app: latest
          ports:
            - containerPort: 7000
~
```

```
apiVersion: v1
kind: Service
metadata:
  name: car-prediction-service
spec:
  selector:
    app: car-prediction
  ports:
    - protocol: TCP
      port: 80
      targetPort: 7000
  type: LoadBalancer
~
```

■ Create s3 bucket

■ Add IAM role in to EC2 instance.

■ Create cluster using these command as

```
< kops create cluster pavani.local –state  
s3://lucky2024 –zones us-northeast-1a,northeast-1b  
–node-count 2 --yes >
```

■ validate cluster using these command as

```
< kops validate cluster >
```

■ kubectl apply the command as

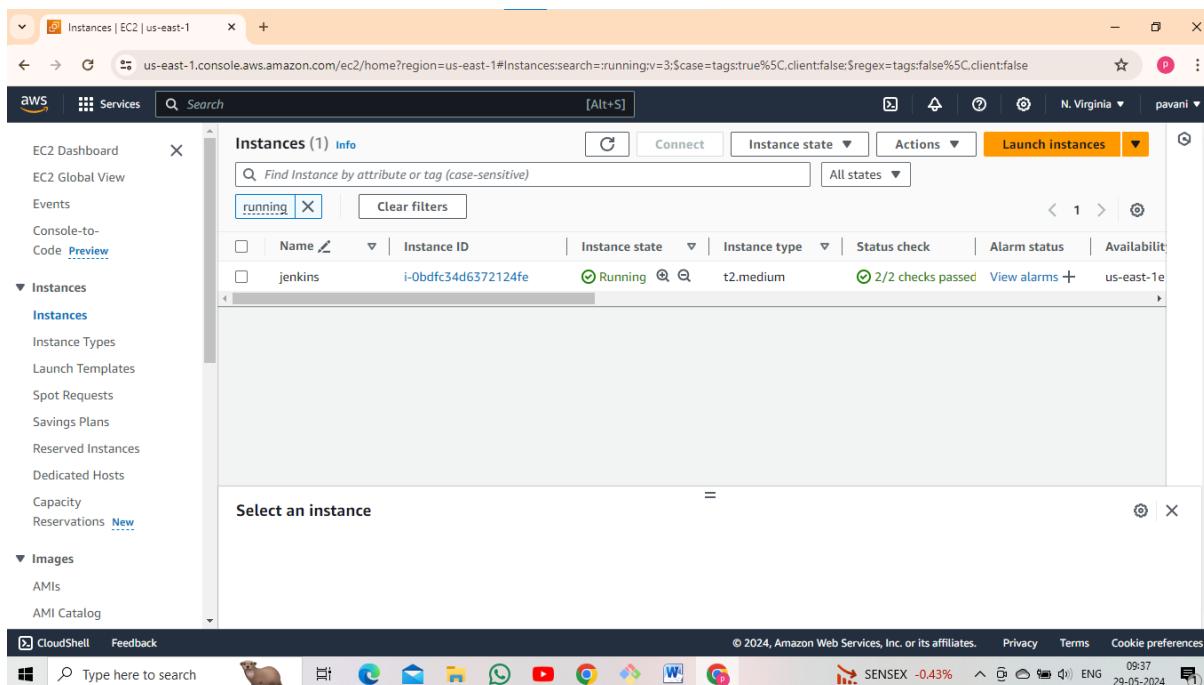
```
< kubectl apply –f deployment.yaml >  
< kubectl apply –f service.yaml >
```

■ After that kubectl delete deployment.yaml,
service.yaml & cluster.

```
*****
```

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

- Go to EC2 instance.
- Select AMI linux.
- Edit inbound rules.
- Launch EC2 instance.



- Add the IAM role into the created EC2 instance.
- Go to Git Bash.
- Update the machine using these following command

```
< sudo yum update >
```
- Install git.

< sudo yum -y install git >

```
xpp3.noarch 0:1.1.3.8-11.amzn2
Complete!
[ec2-user@ip-172-31-53-113 ~]$ java --version
java --version
openjdk 17.0.11 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-17.0.11.9.1 (build 17.0.11+9-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.11.9.1 (build 17.0.11+9-LTS, mixed mode, sharing)
[ec2-user@ip-172-31-53-113 ~]$ git --version
git version 2.40.1
[ec2-user@ip-172-31-53-113 ~]$
```



Install java.

< sudo yum -y install java* >

```
xml-commons-apis.noarch 0:1.4.01-16.amzn2
xml-commons-resolver.noarch 0:1.2-15.amzn2
xmlgraphics-commons.noarch 0:1.5-3.amzn2.0.2
xorg-x11-font-utils.x86_64 1:7.5-21.amzn2
xorg-x11-fonnts-Type1.noarch 0:7.5-9.amzn2
xorg-x11-utils.x86_64 0:7.5-23.amzn2
xpp3.noarch 0:1.1.3.8-11.amzn2

Complete!
[ec2-user@ip-172-31-53-113 ~]$ java --version
java --version
openjdk 17.0.11 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-17.0.11.9.1 (build 17.0.11+9-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.11.9.1 (build 17.0.11+9-LTS, mixed mode, sharing)
[ec2-user@ip-172-31-53-113 ~]$
```



Install Jenkins.

NOTE: Jenkins installation is above steps (4th step)

```
[ec2-user@ip-172-31-53-113 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-53-113 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
    Active: active (running) since Wed 2024-05-29 04:18:42 UTC; 9s ago
      Main PID: 6324 (java)
        CGroup: /system.slice/jenkins.service
                 └─6324 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=...

May 29 04:18:25 ip-172-31-53-113.ec2.internal jenkins[6324]: dc4be6f4a47a415982b20f8c47baa745
May 29 04:18:25 ip-172-31-53-113.ec2.internal jenkins[6324]: This may also be found at: /var/lib/j...rd
May 29 04:18:25 ip-172-31-53-113.ec2.internal jenkins[6324]: ****
May 29 04:18:25 ip-172-31-53-113.ec2.internal jenkins[6324]: ****
May 29 04:18:25 ip-172-31-53-113.ec2.internal jenkins[6324]: ****
May 29 04:18:42 ip-172-31-53-113.ec2.internal jenkins[6324]: 2024-05-29 04:18:42.019+0000 [id=32] ...on
May 29 04:18:42 ip-172-31-53-113.ec2.internal jenkins[6324]: 2024-05-29 04:18:42.036+0000 [id=24] ...ng
May 29 04:18:42 ip-172-31-53-113.ec2.internal systemd[1]: Started Jenkins Continuous Integration S...r.
May 29 04:18:42 ip-172-31-53-113.ec2.internal jenkins[6324]: 2024-05-29 04:18:42.057+0000 [id=50] ...er
May 29 04:18:42 ip-172-31-53-113.ec2.internal jenkins[6324]: 2024-05-29 04:18:42.057+0000 [id=50] ...#1
Hint: some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-53-113 ~]$
```

Windows Type here to search 🐧 🌐 📎 📲 📺 📹 📺 📺 Nifty bank -0.52% ⚡ ENG 09:48 29-05-2024

⊕ Give sudo permissions for the Jenkins.

< sudo visudo >

```
## Allow root to run any commands anywhere
root    ALL=(ALL)      ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)      ALL

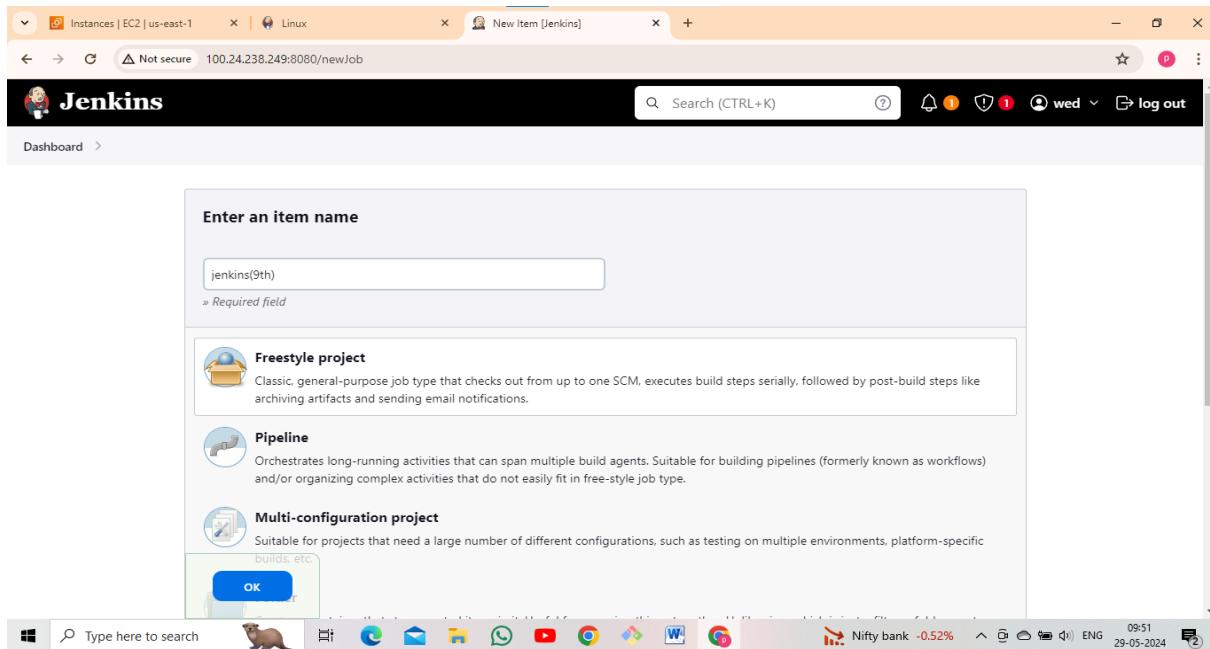
## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
jenkins  ALL=(ALL)      NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users     ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

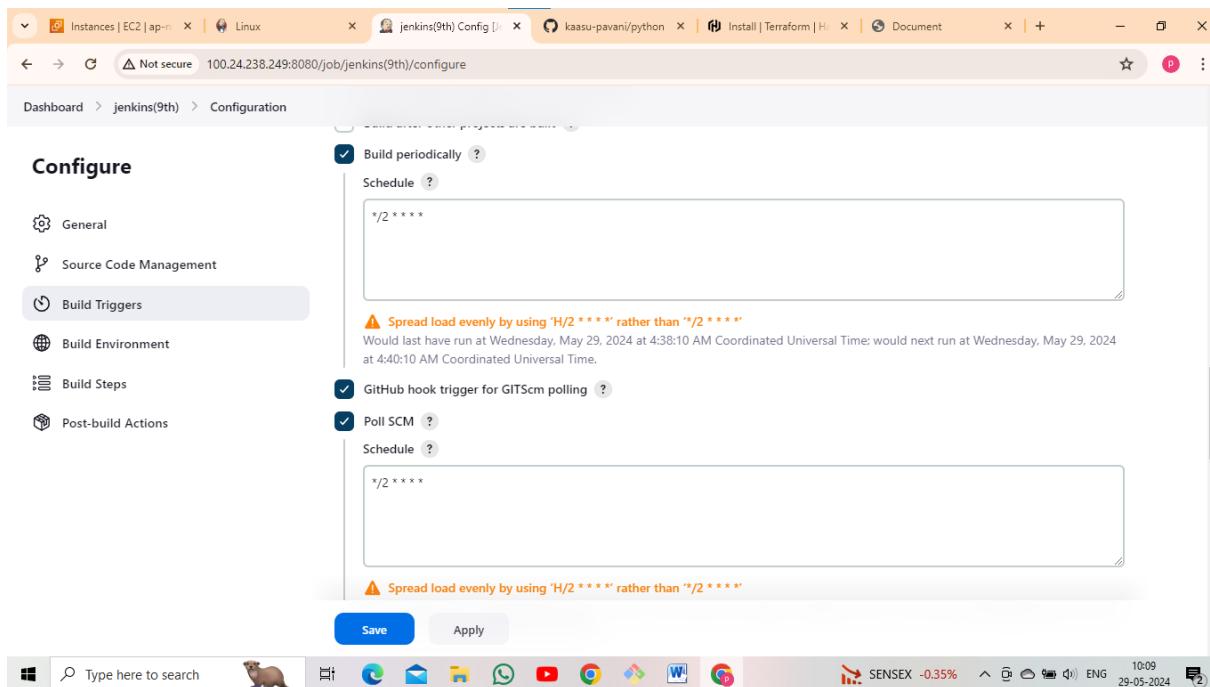
## Allows members of the users group to shutdown this system
# %users    localhost=/sbin/shutdown -h now
```

⊕ Go to Jenkins page.

⊕ Create job.



- Give the Git Hub URL.
- After that select Build periodically and poll SCM. give the */2 * * * *(crown syntax).



- After that select execute shell and give the terraform installation and terraform execution commands.

The screenshot shows the Jenkins configuration interface for a job named 'jenkins(9th)'. The 'Build Steps' section is selected in the sidebar. A single step is defined: 'Execute shell' with the command:

```
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  
terraform init  
terraform validate  
terraform plan  
terraform apply --auto-approve
```

Below the command, there is an 'Advanced' dropdown and a 'Save' button.

After that click the apply and save and Build now.

The screenshot shows the Jenkins console output for a build step. The output shows the execution of Terraform's 'apply' command:

```
user_data_replace_on_change = false
vpc_security_group_ids = [
  "sg-00702f78479aec054"
]

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  public_ip = "15.152.35.191" -> (known after apply)

aws_instance.Terraform: Creating...
aws_instance.Terraform: Still creating... [10s elapsed]
aws_instance.Terraform: Still creating... [20s elapsed]
aws_instance.Terraform: Creation complete after 24s [id=i-0a0c3490165016d6e]

Outputs:
  public_ip = "15.168.60.73"
Finished: SUCCESS
```

At the bottom right, it says 'REST API Jenkins 2.452.1'.

Now go to EC2 instance. check the instance, terraform instance is created.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table of instances. There are two rows, both labeled 'Terraform'. The first row has an instance ID of i-0a8c3490165036d6e and the second has i-03724ef22ab8e654e. Both are in the 'Running' state, t2.micro type, with 2/2 checks passed. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. A search bar at the top allows filtering by attribute or tag. A 'Launch instances' button is visible in the top right.

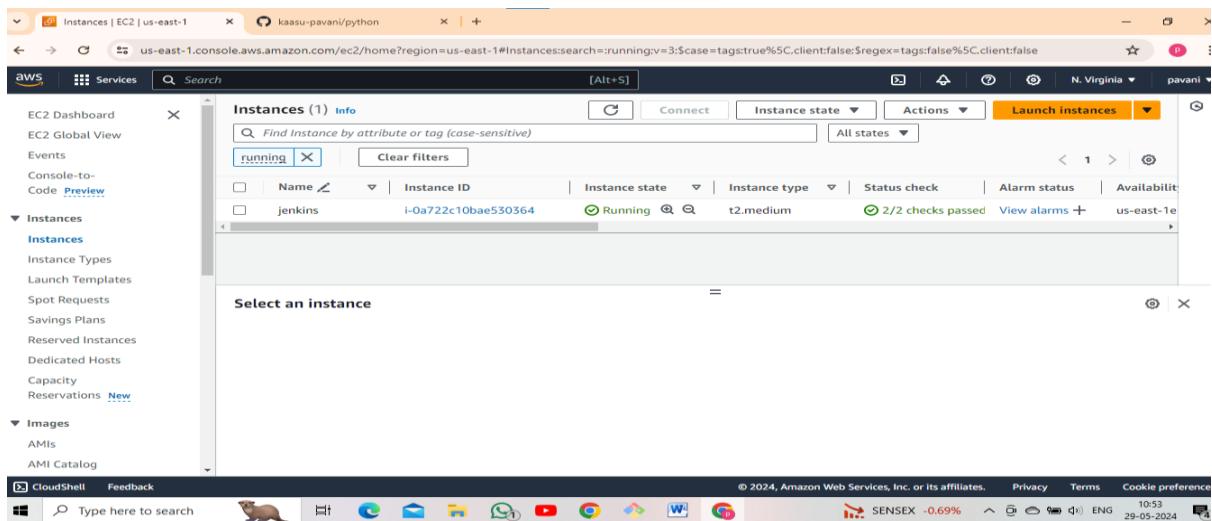
➡ After that copy public ip and paste it Google.

< copy public ip : application port number >

The screenshot shows a web application titled 'Second Hand Car Prediction'. The background features a large image of a sports car. The form contains several dropdown menus and input fields. The first question is 'What Is the Fuel type?' with 'Petrol' selected. The next question is 'Are you A Dealer or Individual' with 'Dealer' selected. Below that is a dropdown for 'Transmission type' with 'Manual C' selected. The next question is 'What is the Starting Price?(In lakhs)' with a blank input field. The final question is 'How Many Kilometers Drived?' with another blank input field. At the bottom, there is a note: 'How much owners previously had the car(0 or 1 or 3) ?' with a dropdown menu showing '0'.

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

- Go to EC2 instance.
- Select AMI linux.
- Select t3.medium.
- Edit inbound rules.
- Launch EC2 instance.



- Go to Git Bash.
- Update the machine using these following command.

```
< sudo yum update >
```

- Install git.

```
< sudo yum -y install git >
```

```

Verifying : git-2.40.1-1.amzn2.0.2.x86_64 3/6
Verifying : git-core-2.40.1-1.amzn2.0.2.x86_64 4/6
Verifying : perl-Git-2.40.1-1.amzn2.0.2.noarch 5/6
Verifying : 1:perl-Error-0.17020-2.amzn2.noarch 6/6

Installed:
git.x86_64 0:2.40.1-1.amzn2.0.2

Dependency Installed:
git-core.x86_64 0:2.40.1-1.amzn2.0.2
perl-Error.noarch 1:0.17020-2.amzn2
perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2
git-core-doc.noarch 0:2.40.1-1.amzn2.0.2
perl-Git.noarch 0:2.40.1-1.amzn2.0.2

Complete!
[ec2-user@ip-172-31-49-74 ~]$ git --version
git version 2.40.1
[ec2-user@ip-172-31-49-74 ~]$
```




Install java.

< sudo yum -y install java* >

```

[xec2-user@ip-172-31-49-74 ~]$ xerces-j2.noarch 0:2.11.0-17.amzn2
xml-commons-apis.noarch 0:1.4.01-16.amzn2
xml-commons-resolver.noarch 0:1.2-15.amzn2
xmlgraphics-commons.noarch 0:1.5-3.amzn2.0.2
xorg-x11-font-utils.x86_64 1:7.5-21.amzn2
xorg-x11-fonnts-Type1.noarch 0:7.5-9.amzn2
xorg-x11-utils.x86_64 0:7.5-23.amzn2
xpp3.noarch 0:1.1.3-8-11.amzn2

Complete!
[ec2-user@ip-172-31-49-74 ~]$ java --version

openjdk 17.0.11 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-17.0.11.9.1 (build 17.0.11+9-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.11.9.1 (build 17.0.11+9-LTS, mixed mode, sharing)
[ec2-user@ip-172-31-49-74 ~]$
```




Install Jenkins.

NOTE: Jenkins installation process is above steps (4th step).

```

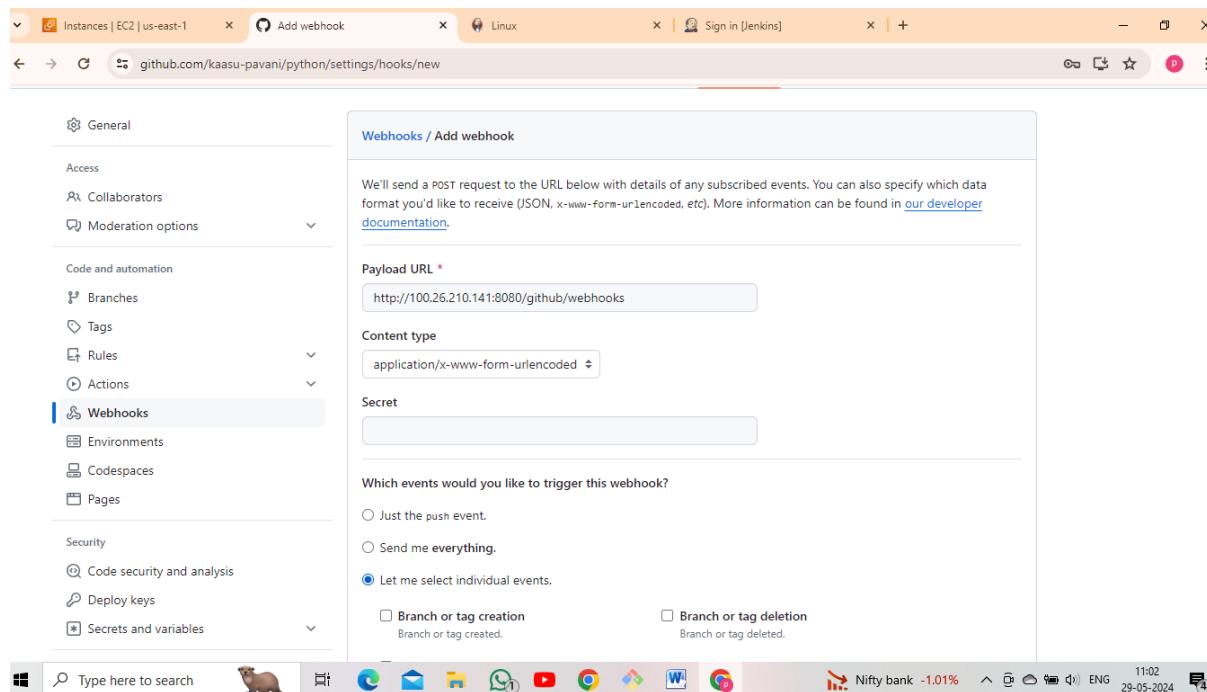
[ec2-user@ip-172-31-49-74 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-49-74 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
  Active: active (running) since Wed 2024-05-29 05:29:36 UTC; 9s ago
    Main PID: 5404 (java)
      CGroup: /system.slice/jenkins.service
              └─5404 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=...

May 29 05:29:21 ip-172-31-49-74.ec2.internal jenkins[5404]: c4fc0a2a2318432eb1a18914b97ada41
May 29 05:29:21 ip-172-31-49-74.ec2.internal jenkins[5404]: This may also be found at: /var/lib/je...rd
May 29 05:29:21 ip-172-31-49-74.ec2.internal jenkins[5404]: ****
May 29 05:29:36 ip-172-31-49-74.ec2.internal jenkins[5404]: 2024-05-29 05:29:36.304+0000 [id=34] ...on
May 29 05:29:36 ip-172-31-49-74.ec2.internal jenkins[5404]: 2024-05-29 05:29:36.333+0000 [id=25] ...ng
May 29 05:29:36 ip-172-31-49-74.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
May 29 05:29:36 ip-172-31-49-74.ec2.internal jenkins[5404]: 2024-05-29 05:29:36.408+0000 [id=50] ...er
May 29 05:29:36 ip-172-31-49-74.ec2.internal jenkins[5404]: 2024-05-29 05:29:36.408+0000 [id=50] ...#1
Hint: some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-49-74 ~]$
```




➡ Go to Git Hub repo.

➡ Create webhooks.



➡ Go to Jenkins page.

➡ Create 3 jobs (using pipeline creation).

➡ After that creating first job.

➡ Give the Git Hub URL.

➡ Select Build after other projects are built.

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

After that execute shell give the installation of terraform commands and execution of terraform commands.

Build Steps

The screenshot shows the Jenkins build step configuration. It is titled "Execute shell". The "Command" field contains the following Jenkinsfile code:

```
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
terraform init
terraform validate
terraform plan
terraform apply --auto-approve
```

Below the command field are two buttons: "Save" and "Apply". At the bottom of the screen, there is a system tray with icons for various applications and a weather widget showing 35°C Sunny.

After that click apply and save and Build now.

The first job console output as

```
}
```

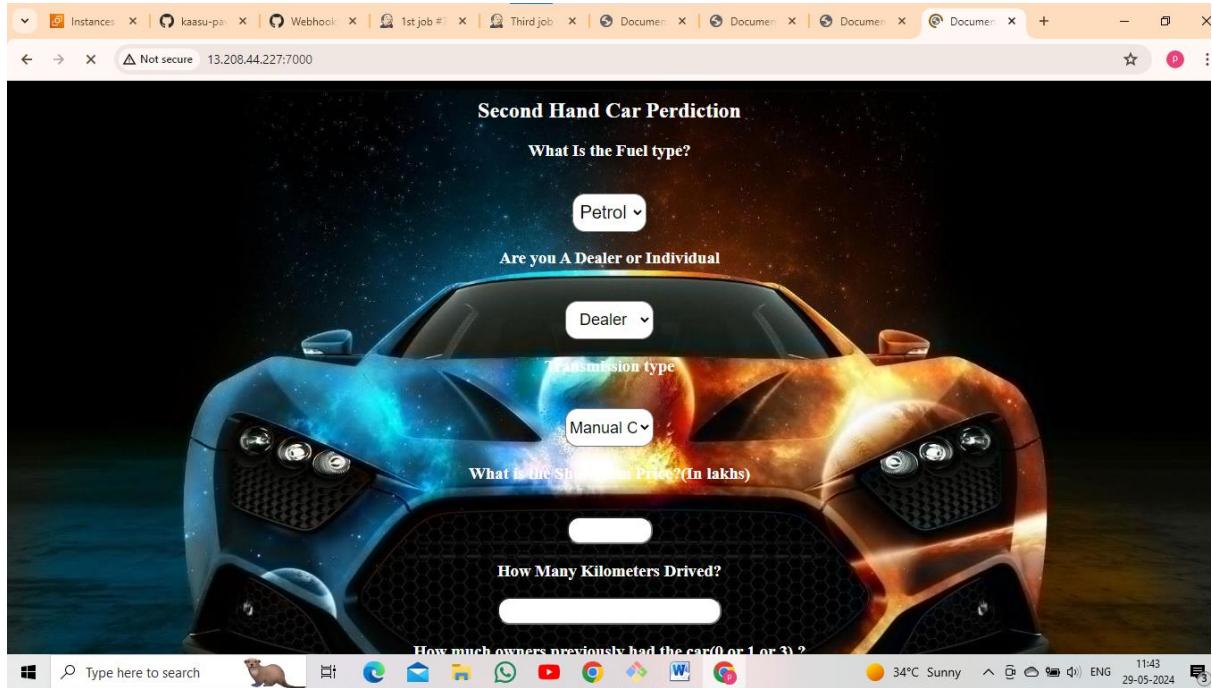
```
 1mPlan: 1 to add, 0 to change, 0 to destroy.
 0m
Changes to Outputs:
 33m~[0m~[0m public_ip = "15.152.35.191" -> (known after apply)
 0m~[1maws_instance.Terraform: Creating... 0m~[0m
 0m~[1maws_instance.Terraform: Still creating... [10s elapsed] 0m~[0m
 0m~[1maws_instance.Terraform: Still creating... [20s elapsed] 0m~[0m
 0m~[1maws_instance.Terraform: Still creating... [30s elapsed] 0m~[0m
 0m~[1maws_instance.Terraform: Creation complete after 35s [id=i-00ee5e48c05a090a3] 0m
 0m~[1m~[32m
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
 0m~[0m~[1m~[32m
Outputs:

 0mpublic_ip = "13.208.44.227"
Triggering a new build of second job
Finished: SUCCESS
```

REST API Jenkins 2.452.1



➡ Console output public copy ip and paste it Google.



➡ After that go to Jenkins dashboard create new item.

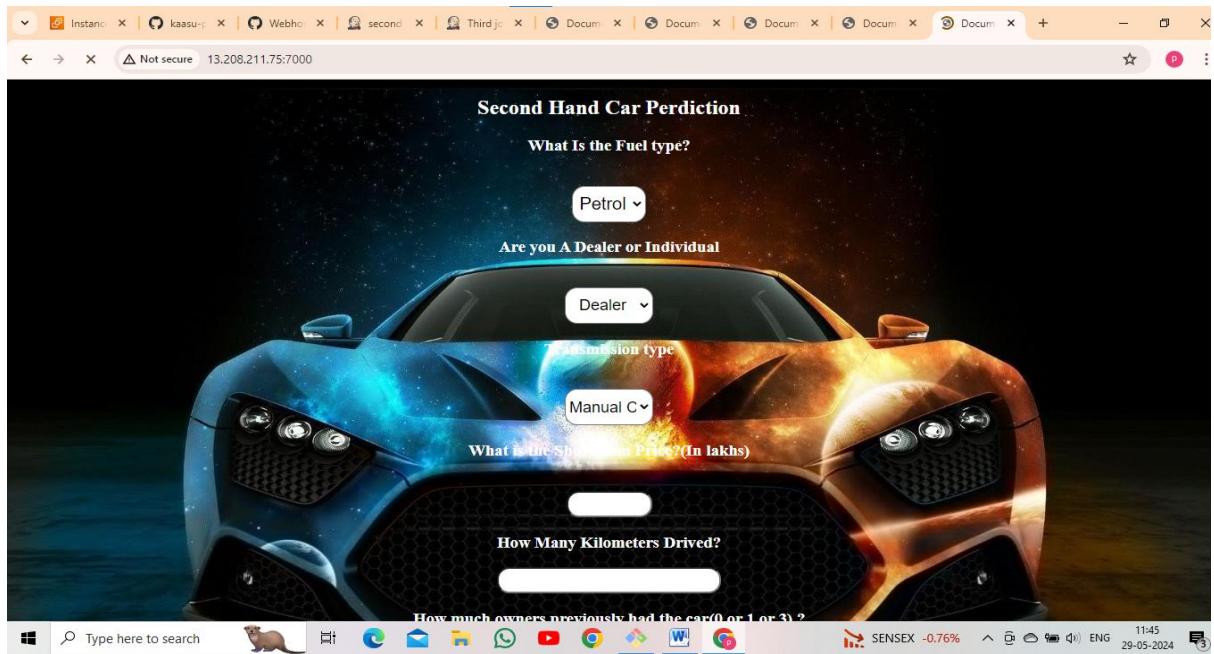
- ➡ Select Build after other projects are built.
- ➡ Next step is apply and save and Build now.

```
Console Output
 32m+ 0m@0m vpc_security_group_ids      =
 32m+ 0m "sg-00702f78479aec054",
}

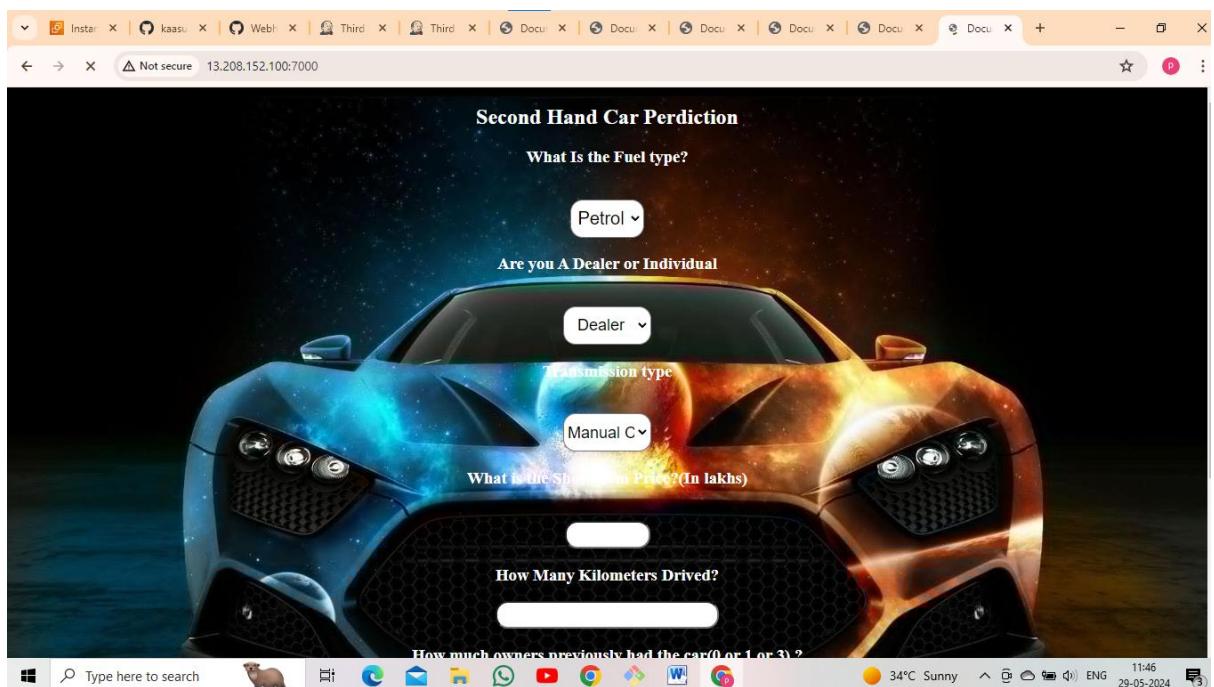
[imPlan:@0m 1 to add, 0 to change, 0 to destroy.
@0m
Changes to Outputs:
@33m-@0m@0m public_ip = "15.152.35.191" -> (known after apply)
@0m@1maws_instance.Terraform: Creating...@0m@0m
@0m@1maws_instance.Terraform: Still creating... [10s elapsed]@0m@0m
@0m@1maws_instance.Terraform: Still creating... [20s elapsed]@0m@0m
@0m@1maws_instance.Terraform: Creation complete after 24s [id=i-045e44a8a6ebd9973]@0m
@0m@1m@32m
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
@0m@0m@1m@32m
Outputs:
@0mpublic_ip = "13.208.211.75"
Triggering a new build of Third job
Finished: SUCCESS

REST API Jenkins 2.452.1
SENSEX -0.76% 11:44
ENG 29-05-2024 3
```

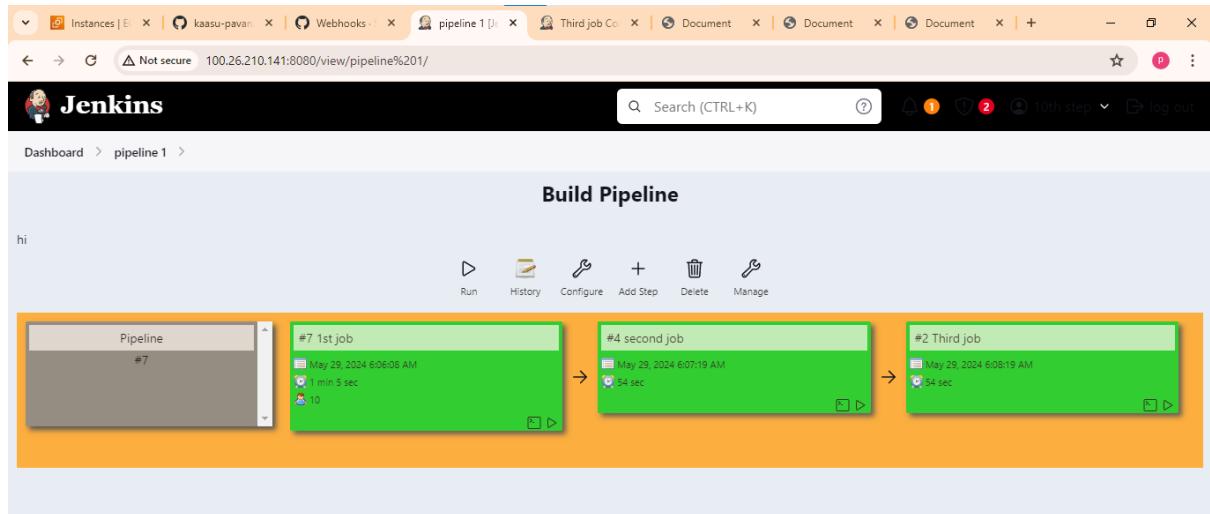
➡ Console output public ip and paste it Google.



- + Same process are repeated in third job creation.
- + Go to Jenkins dashboard.
- + Create job.
- + Select Build after other projects are built.
- + After that click apply and save and Build now.



- Next step is creating pipeline.
- Jenkins Dashboard->manage Jenkins->available plugins->search for Build pipeline and install.



*****THE END*****

