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**

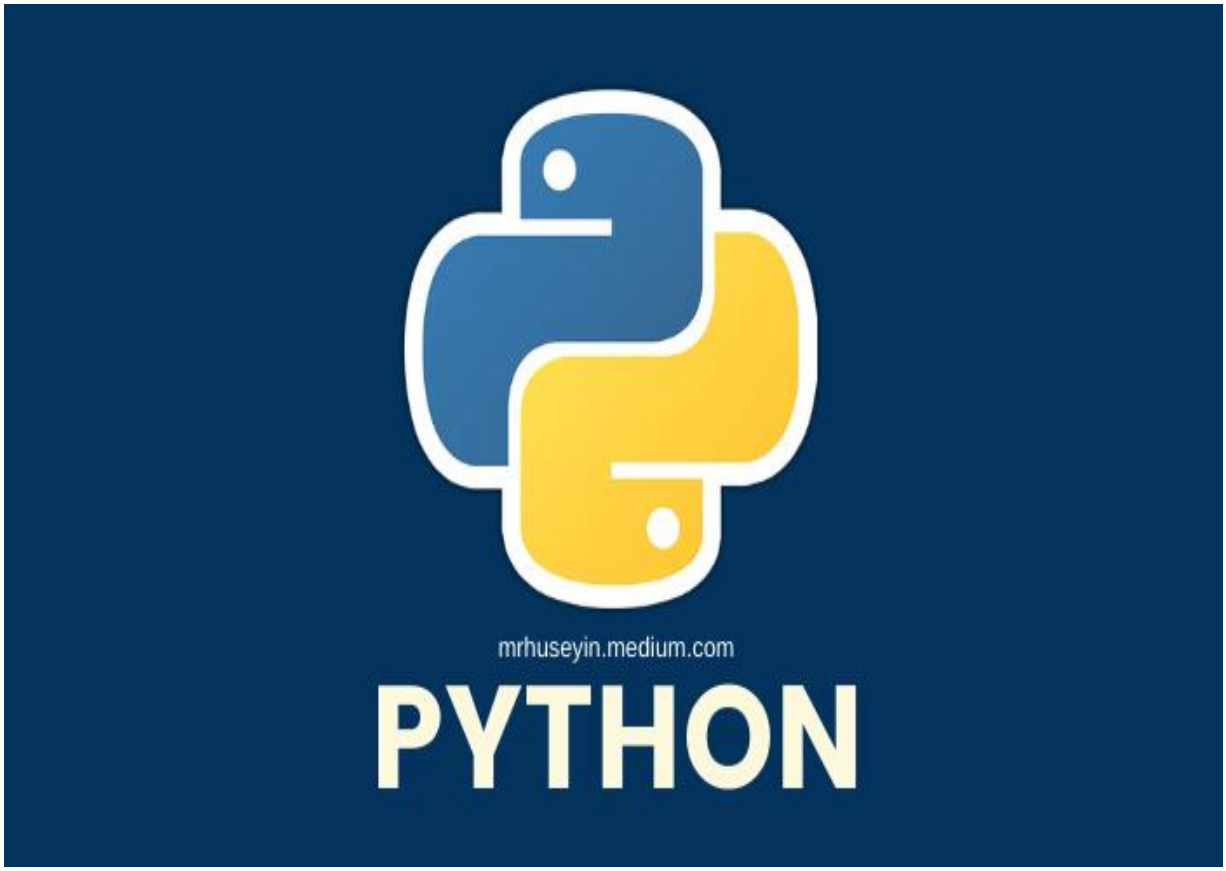
Filename extensions: .**py,.pyw,.pyz,.pyi,.pyc,.pyd**

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.

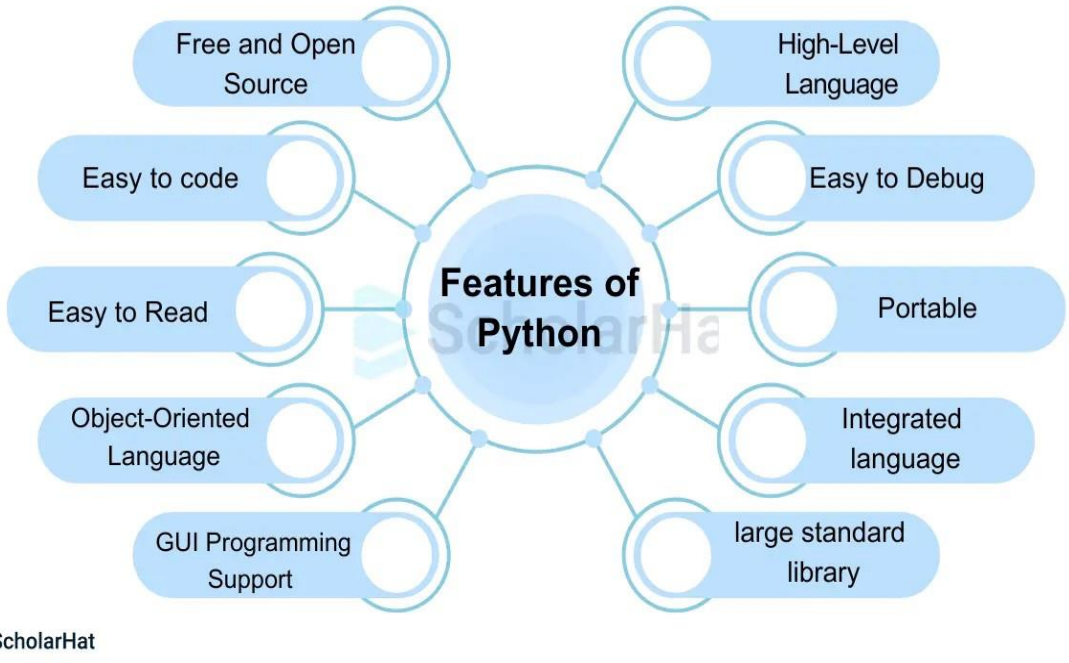
Advantages:

- 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:

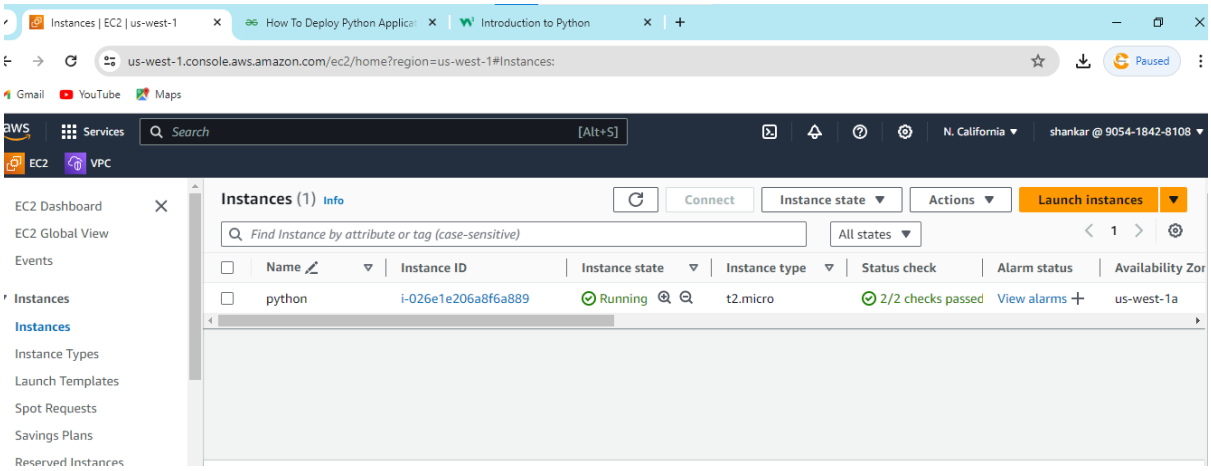
- Python is Slow at Runtime.
- Mobile Application Development.
- Difficulty in Using Other Languages.
- High Memory Consumption.
- Not used in the Enterprise Development Sector.
- Runtime Errors.
- Simplicity.

Features:



1.Build and deploy python applications in manual process?

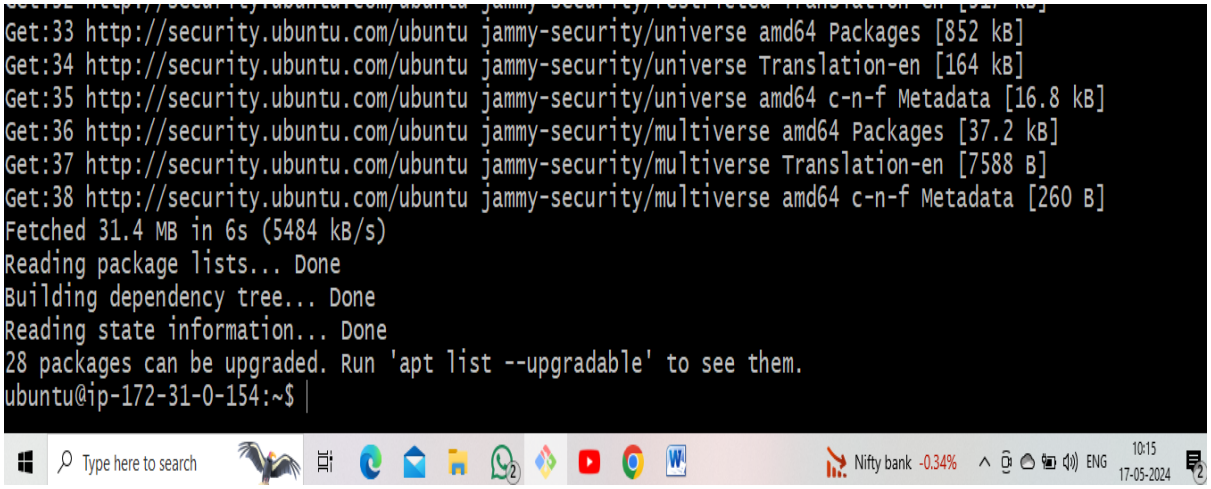
- Go to EC2 instance.
- Select Ubuntu server.
- Edit inbound rules.
- Launch EC2 instance.



Step 1: (Flask-Library-app)

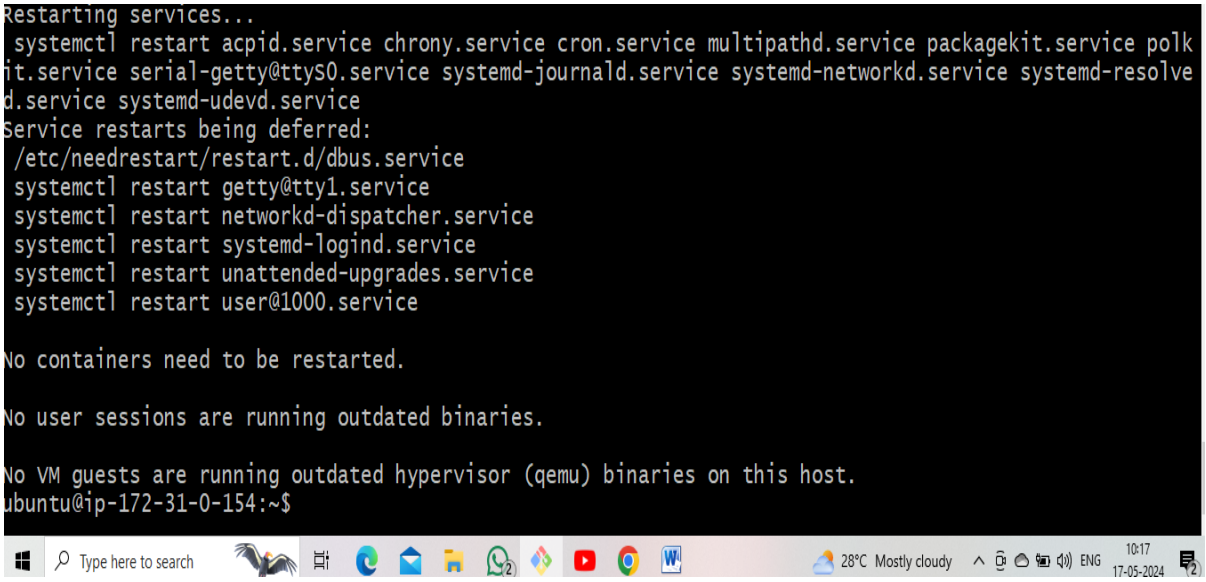
- Connect to Git Bash.
- Go to Git Bash.
- Update Ubuntu machine using these following command as

  < sudo apt update >



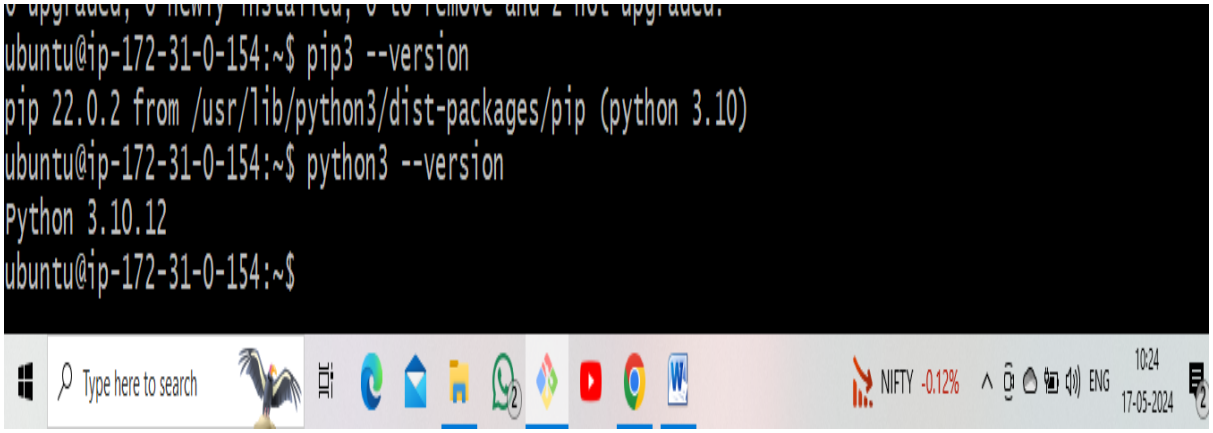
- Full upgrade the machine using these following command as

  < sudo apt-get full-upgrade –y >



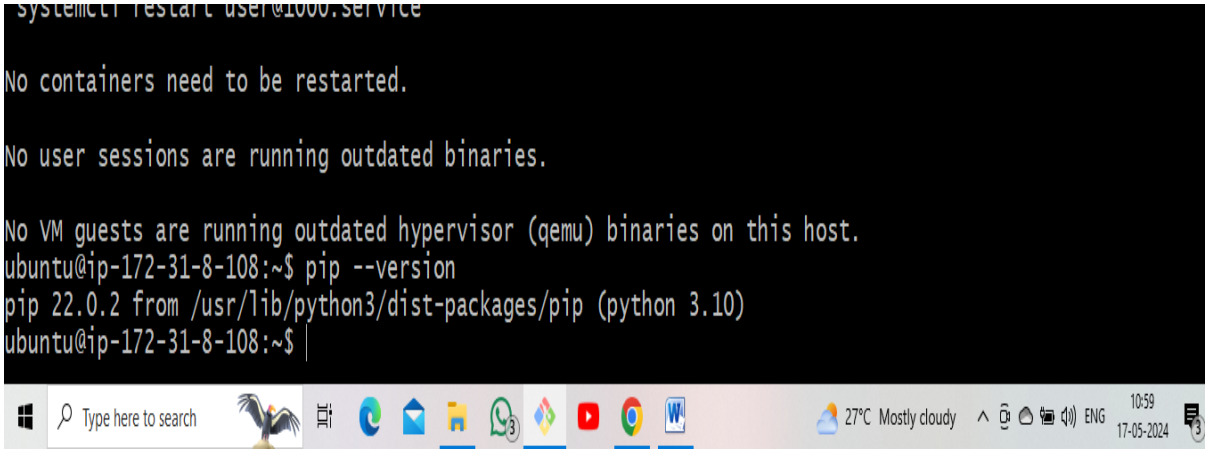
- Install required packages or tools related for deployment project.
- Install python and pip using these command as follows

  < sudo apt-get install python-3 pip >



- Check the status

  < pip3 --version>



- 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 >

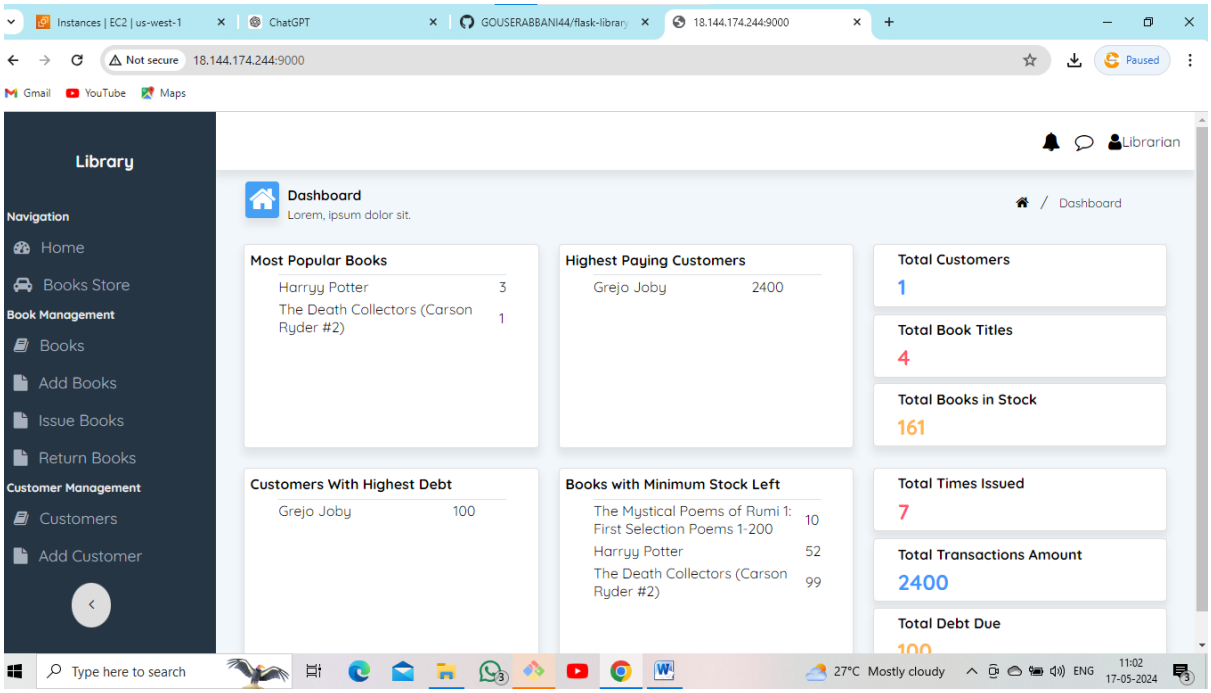
- After running python app.py. It will generate local host IP address. using these command as

  < 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

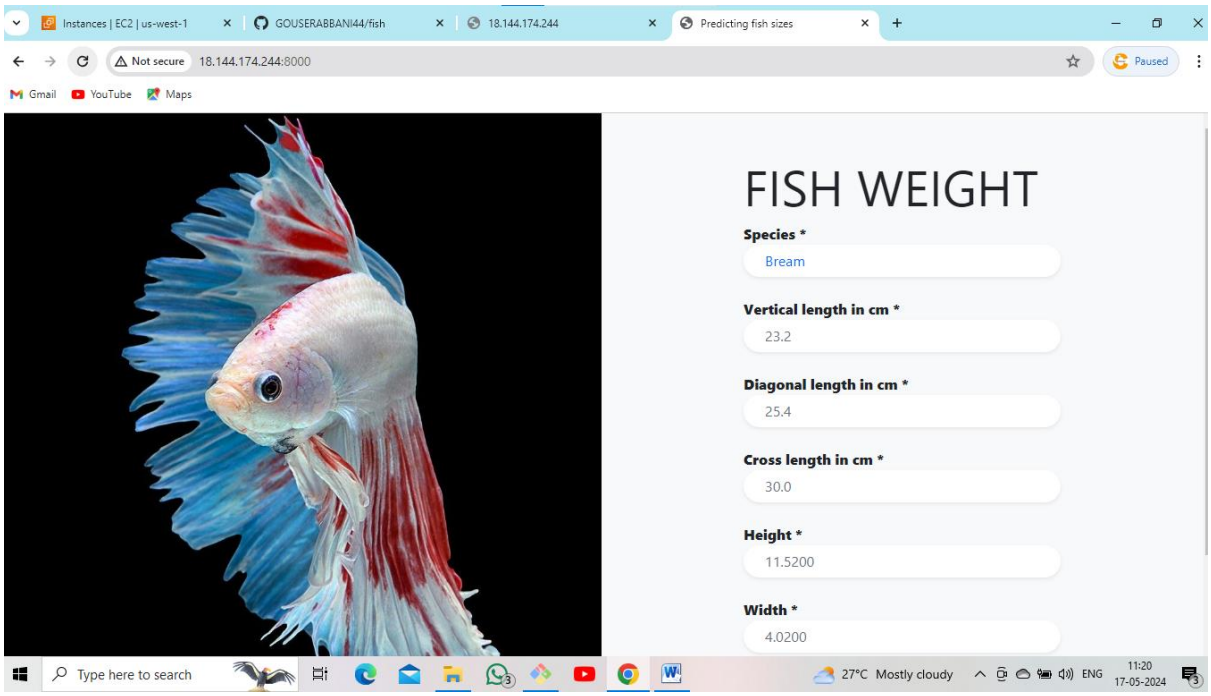
  < sudo vi app.py >

- 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

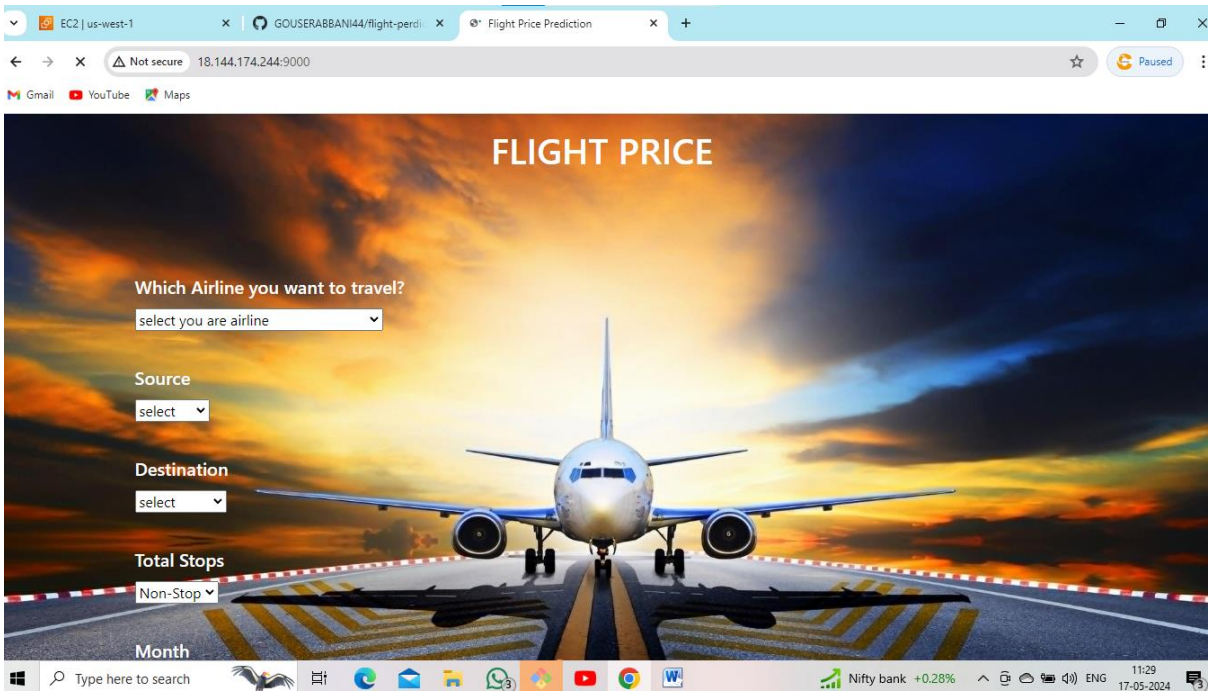
  < sudo vi app.py >

- 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

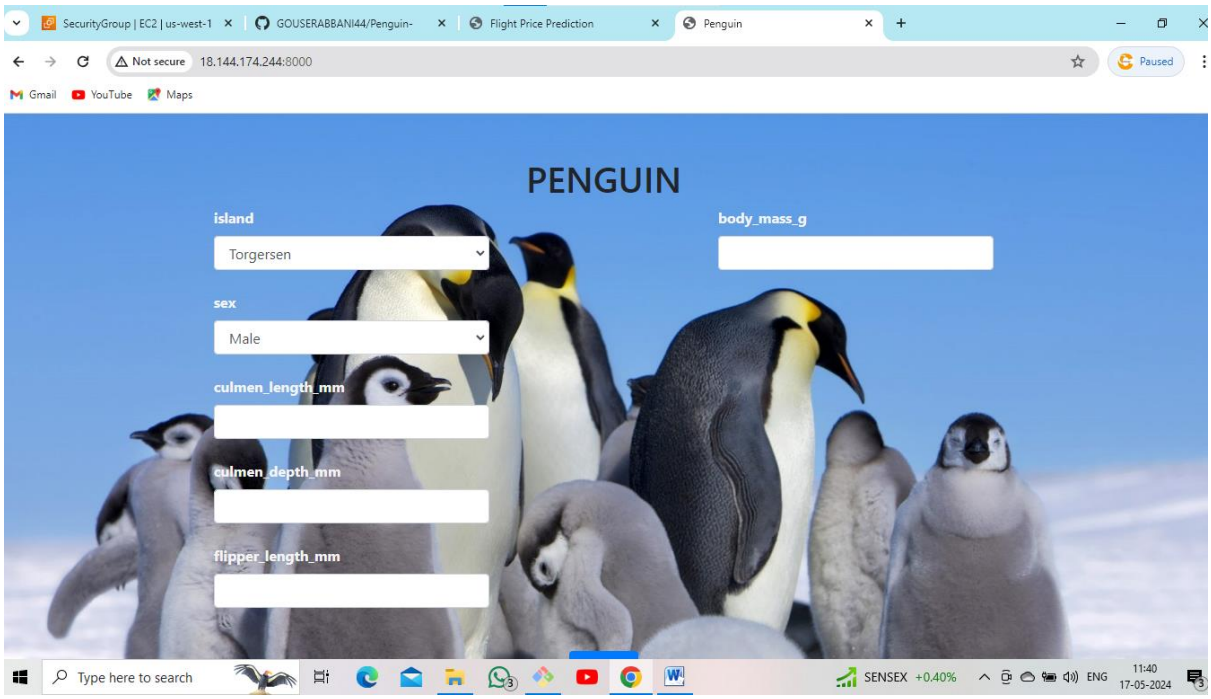
< sudo vi app.py >

- 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

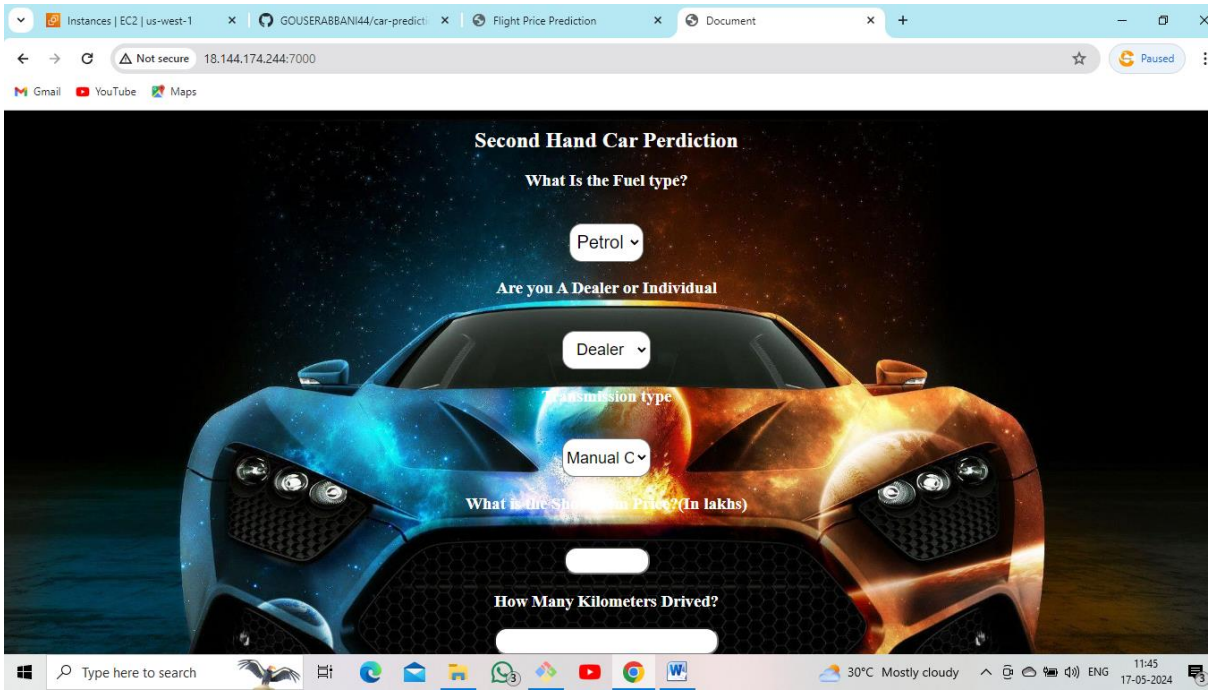
< sudo vi app.py >

- 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

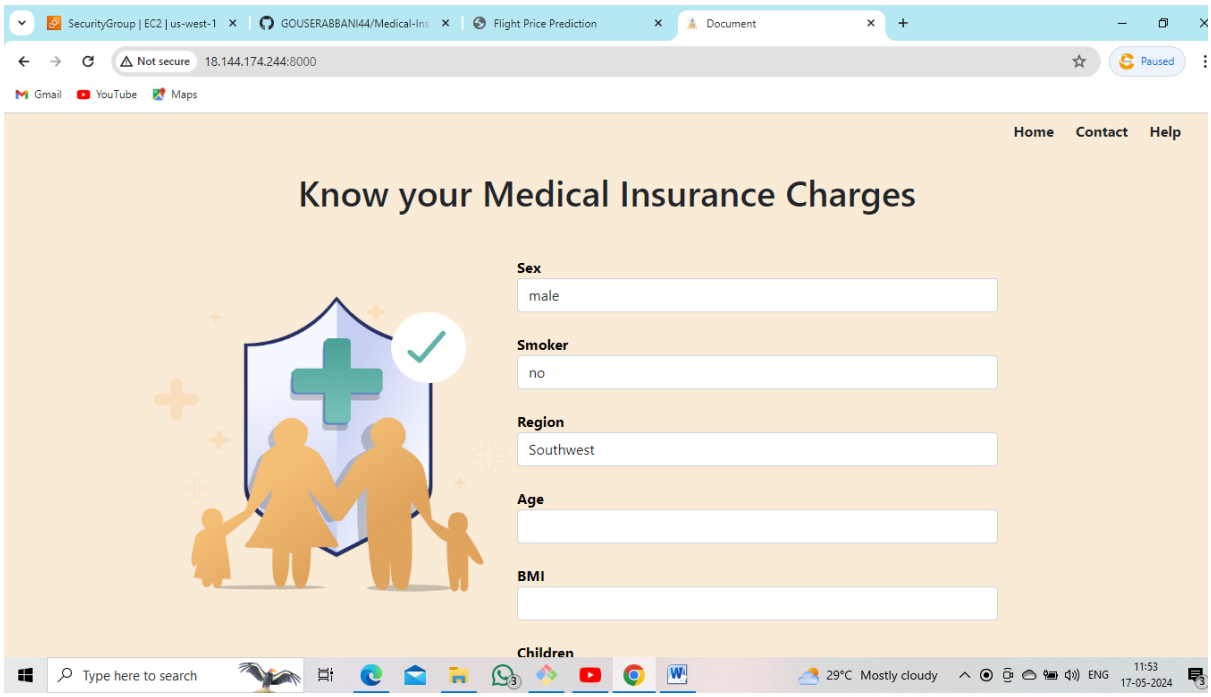
  < sudo vi app.py >

- 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

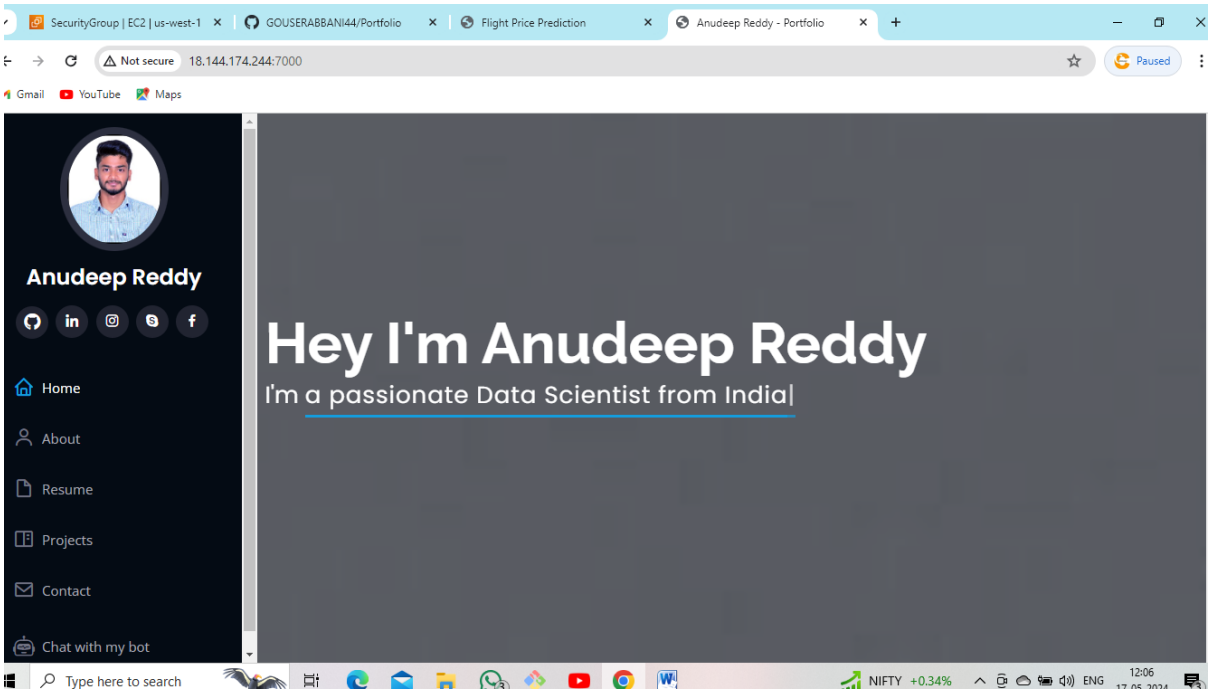
  < sudo vi app.py >

- 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

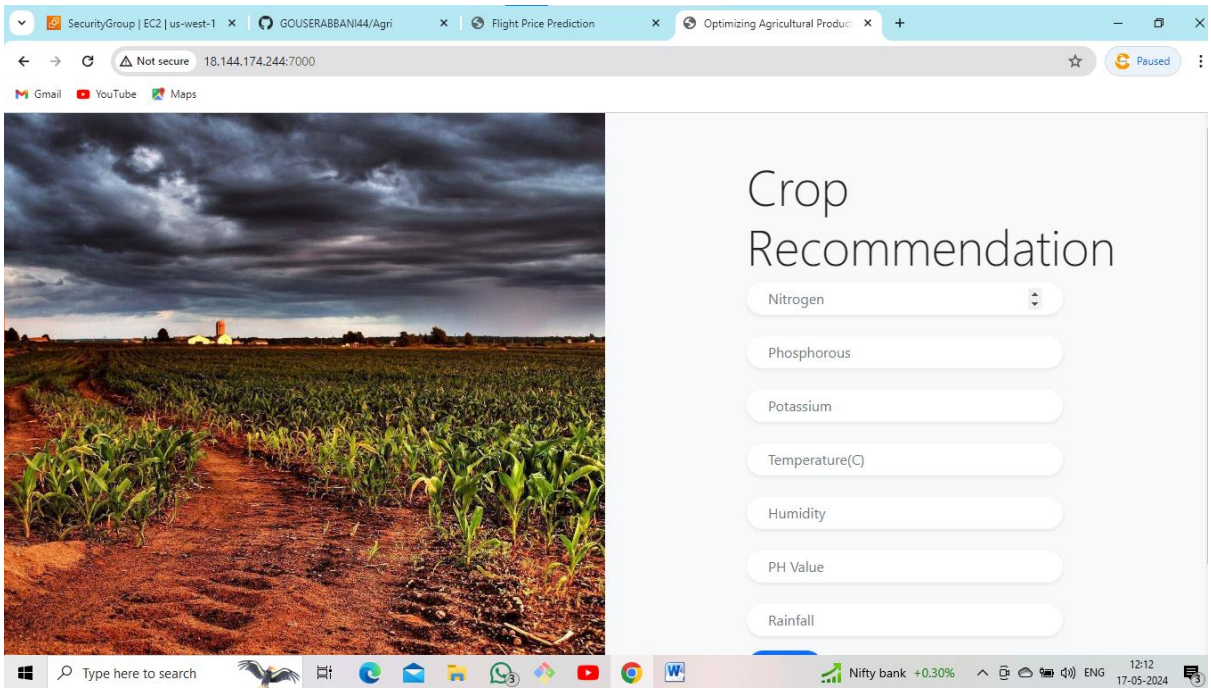
  < sudo vi app.py >

- 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

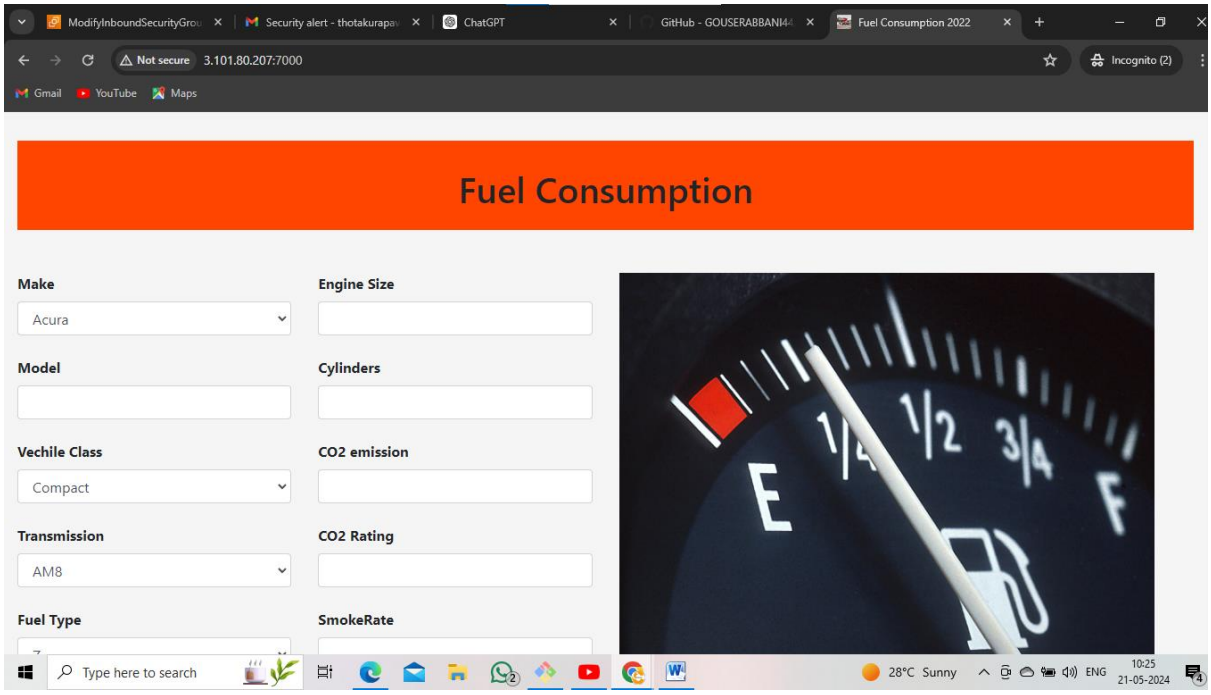
  < sudo vi app.py >

- 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

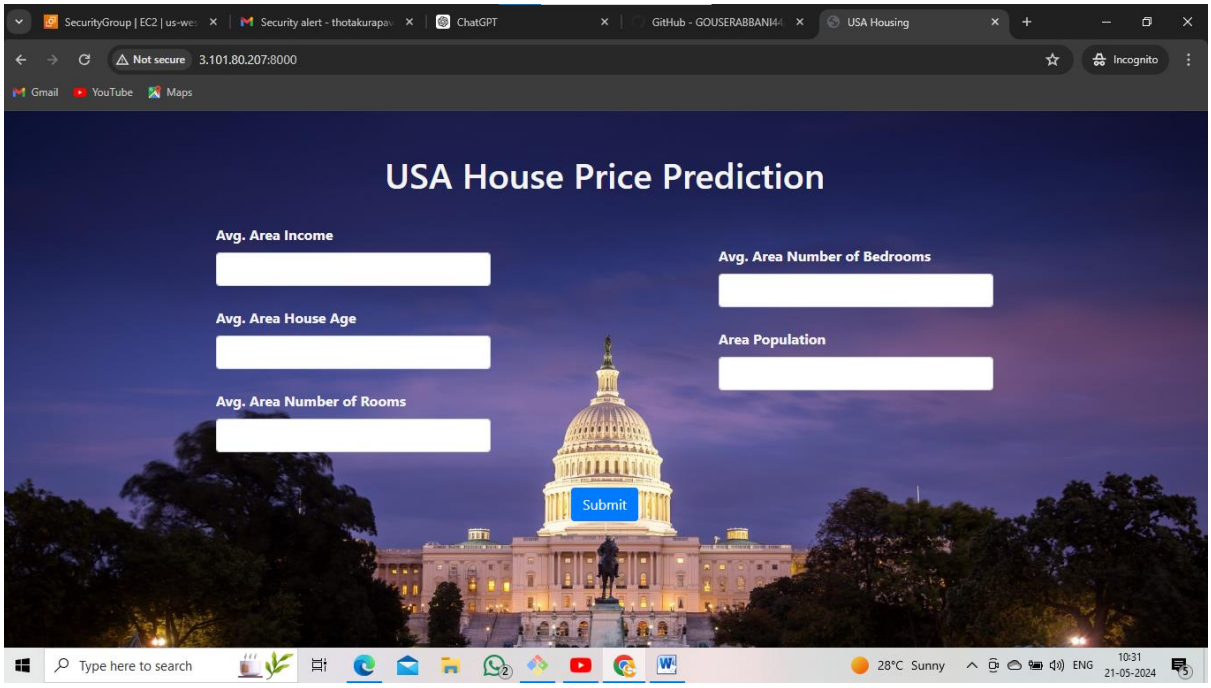
  < cd (repo name) >
- Install requirements packages using these command as follows

  < 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

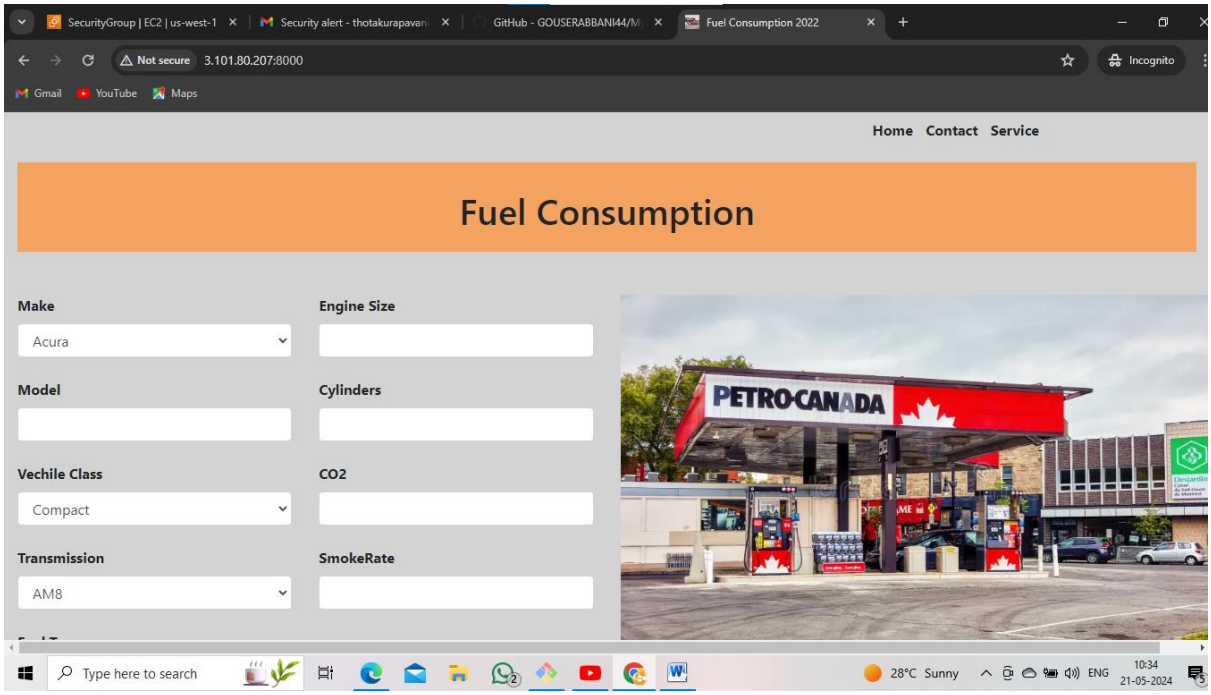
  < sudo vi app.py >

- 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

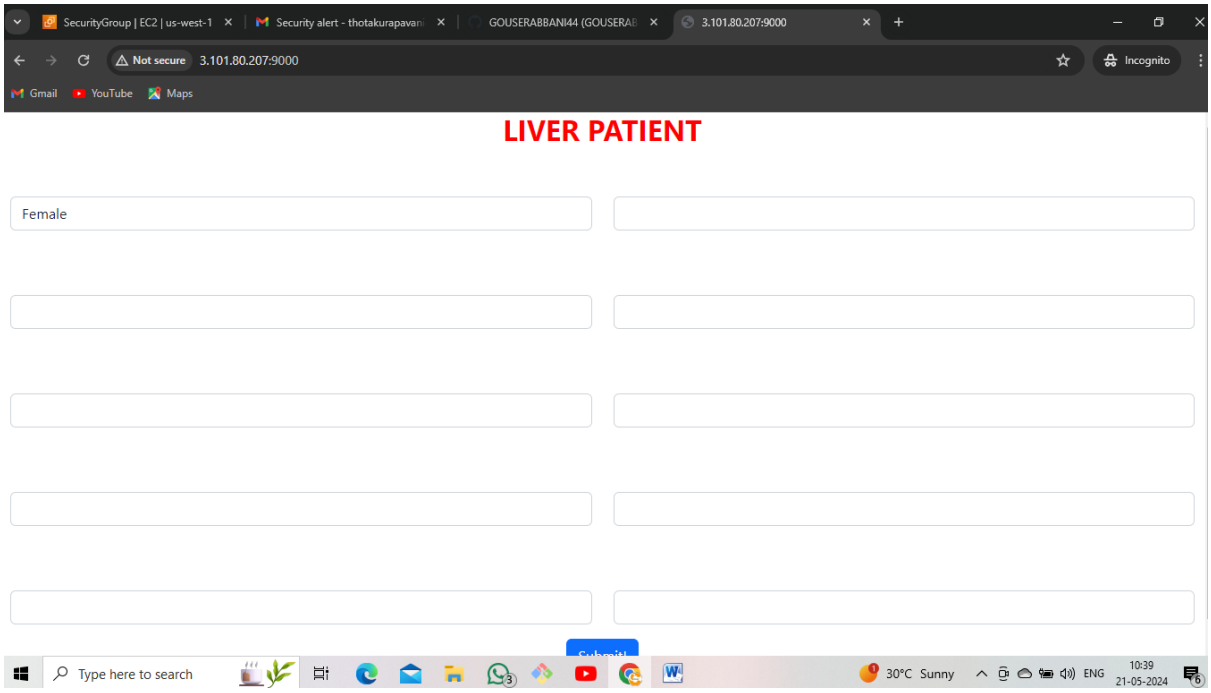
  < cd (repo name) >
- Install requirements packages using these command as follows

  < 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

```
103.213.202.30 - - [21/May/2024 09:51:21] "GET /?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1"
200 -
103.213.202.30 - - [21/May/2024 09:51:21] "GET /?__debugger__=yes&cmd=resource&f=console.png HTTP/1.1"
304 -
^[[A^[[A^Cubuntu@ip-172-31-23-62:~/hearing$ ls
'Logistic Regression.ipynb'   Templates   first          model.pkl          scaler_joblib
 Procfile                     app.py      gitattributes   requirements.txt
ubuntu@ip-172-31-23-62:~/hearing$ sudo mv Templates templates
ubuntu@ip-172-31-23-62:~/hearing$ ls
'Logistic Regression.ipynb'   app.py   gitattributes   requirements.txt   templates
 Procfile                     first    model.pkl       scaler_joblib
ubuntu@ip-172-31-23-62:~/hearing$ sudo vi requirements.txt
ubuntu@ip-172-31-23-62:~/hearing$ sudo vi requirements.txt
ubuntu@ip-172-31-23-62:~/hearing$ sudo vi requirements.txt
ubuntu@ip-172-31-23-62:~/hearing$ pip3 install -r requirements.txt
```

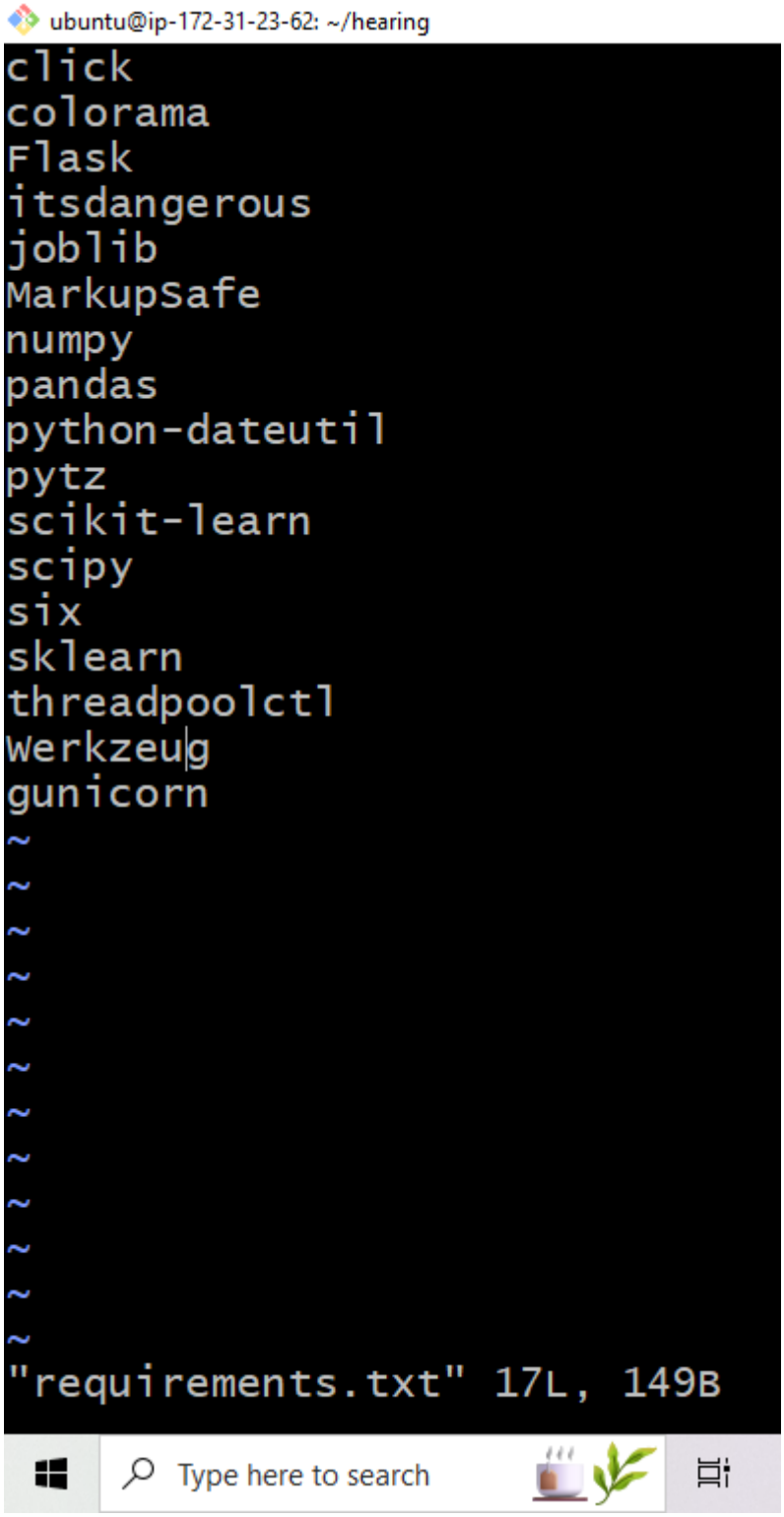
- I am using these command for name changing of Templates to templates.

  < sudo mv Templates templates >

- I am using these command for requirements file.

< sudo vi requirements.txt >

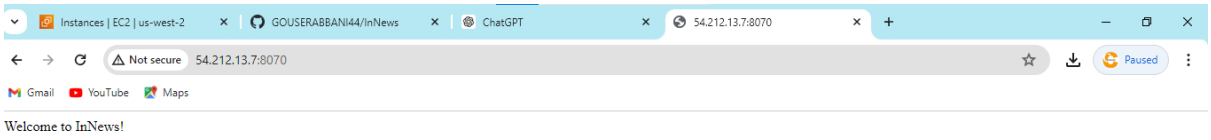
- Remove the requirement versions.



```
            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

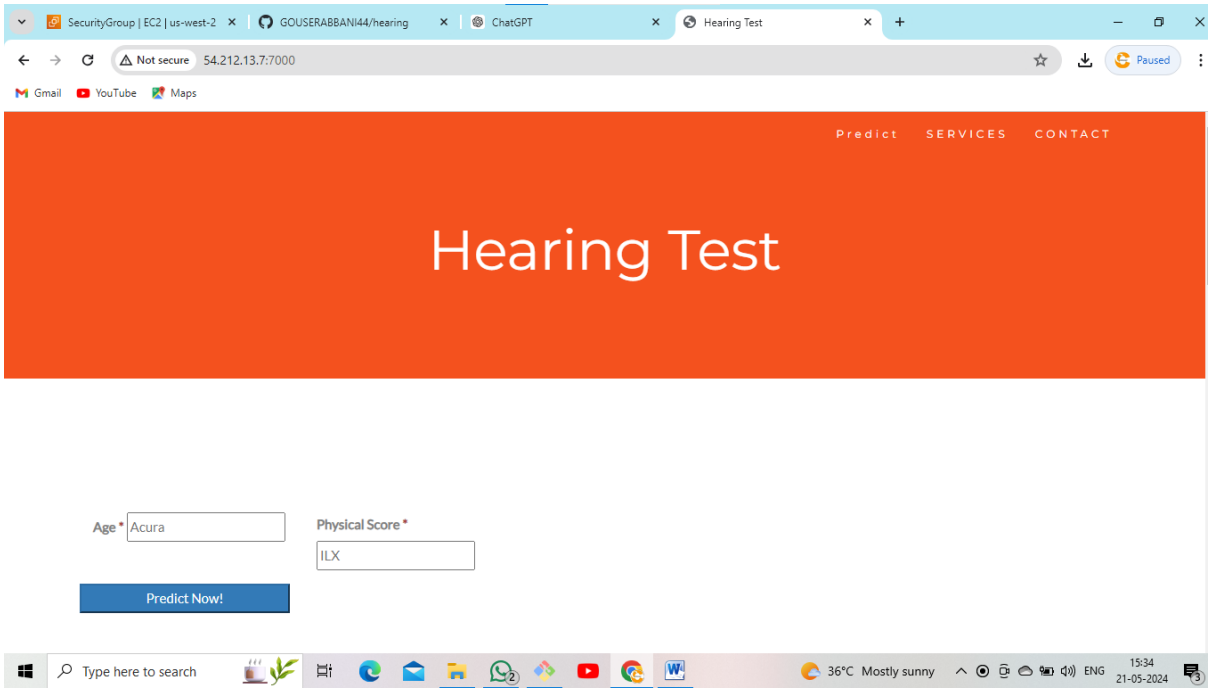
  < cd (repo name) >
- Install requirements packages using these command as follows

  < 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 >



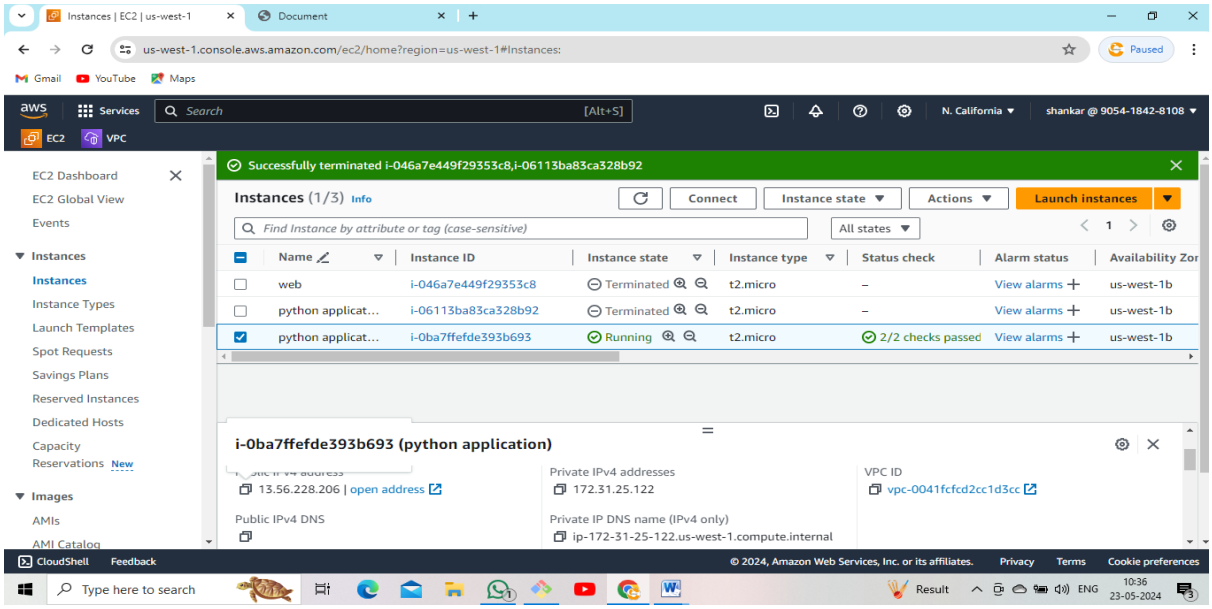
****

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.

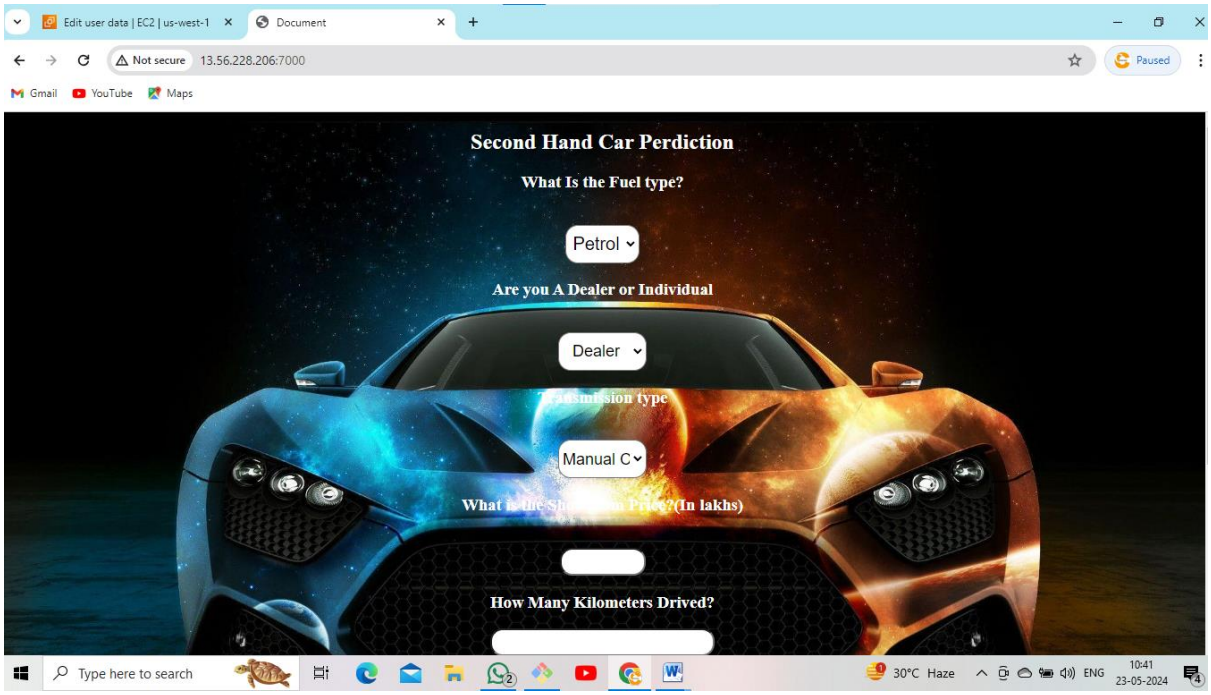
```
#!/bin/bash
sudo apt-get update -y
sudo apt-get install -y python3 python3-pip git
git clone https://github.com/kaasu-pavani/car-prediction.git /home/ubuntu/car-prediction
cd /home/ubuntu/car-prediction
pip3 install -r requirements.txt
```

- Launch EC2 instance.



- Copy public ip and paste it google.

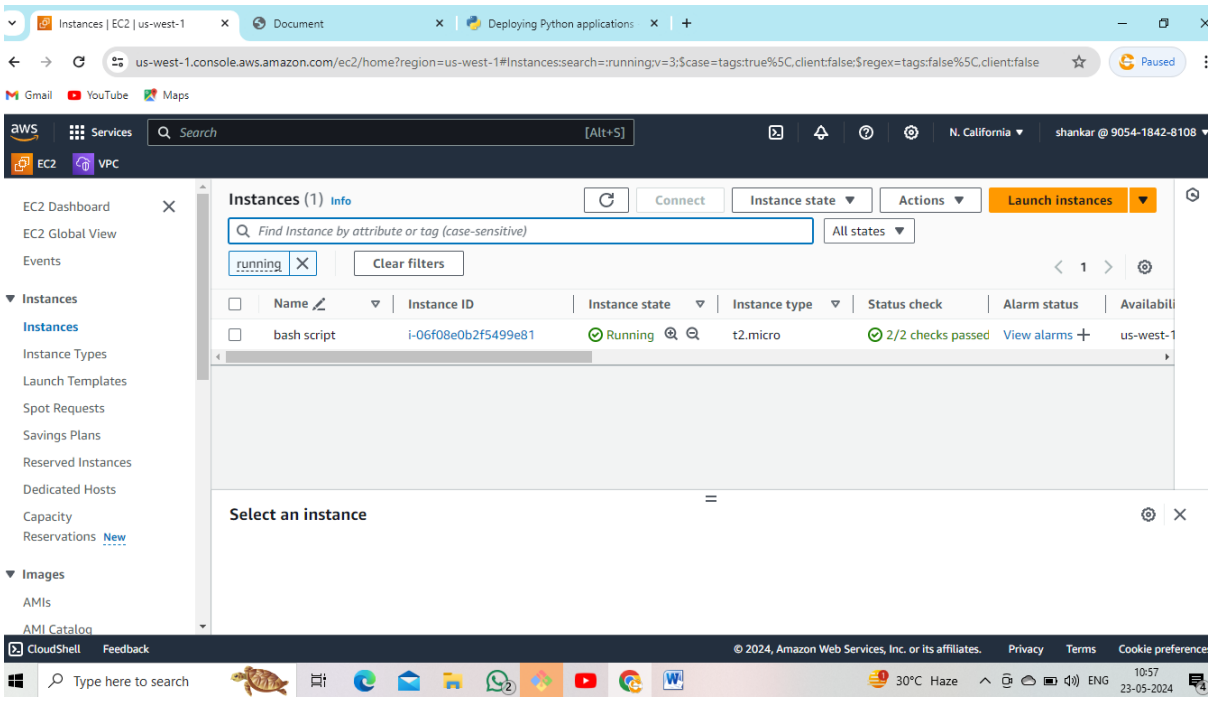
< copy public ip : port number >



****

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-172-31-22-109: ~
#!/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
~
~
~
~
```

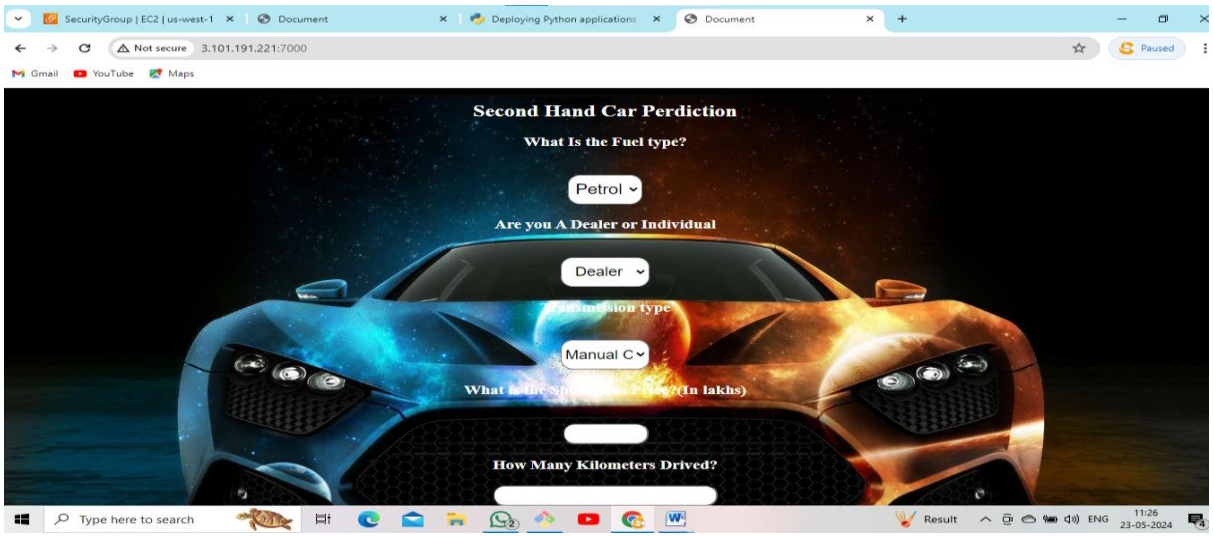
- After that given the permissions.

< sudo chmod +x data.sh >

< ./data.sh >

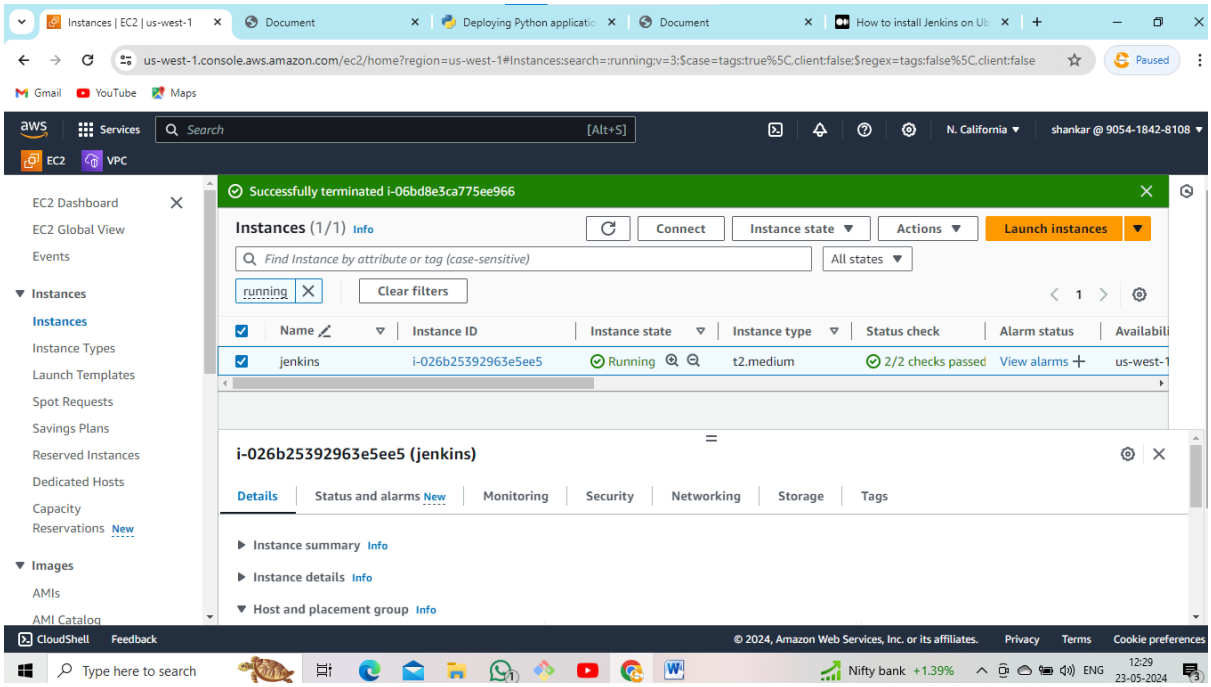
- Go to EC2 instance copy public ip and paste it Google.

< 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.mediam.
- Edit inbound rules.(HTTP,8080,application port number or all traffic)
- Launch EC2 instance.



- Go to Git Bash.
- Update the machine.

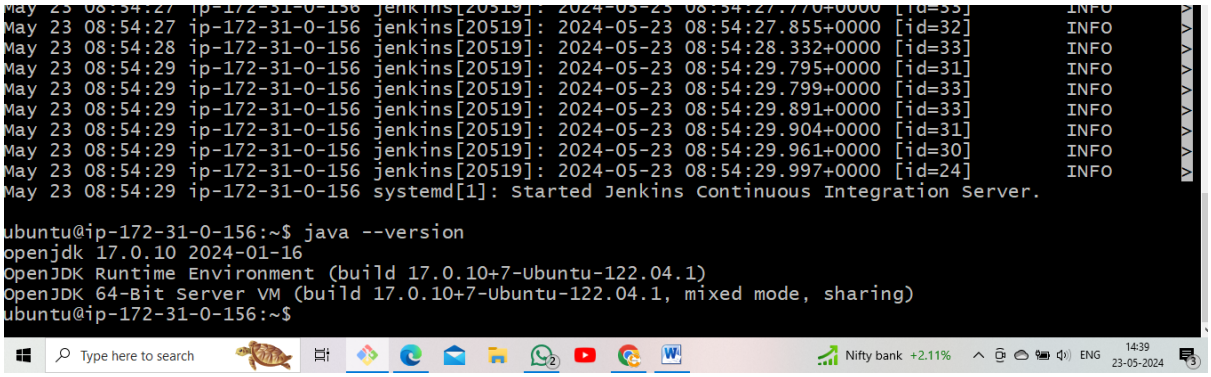
  < sudo apt update >

- After that install git using these command following below.

  < sudo yum install git –y >

- Install java for using these command below

  < >



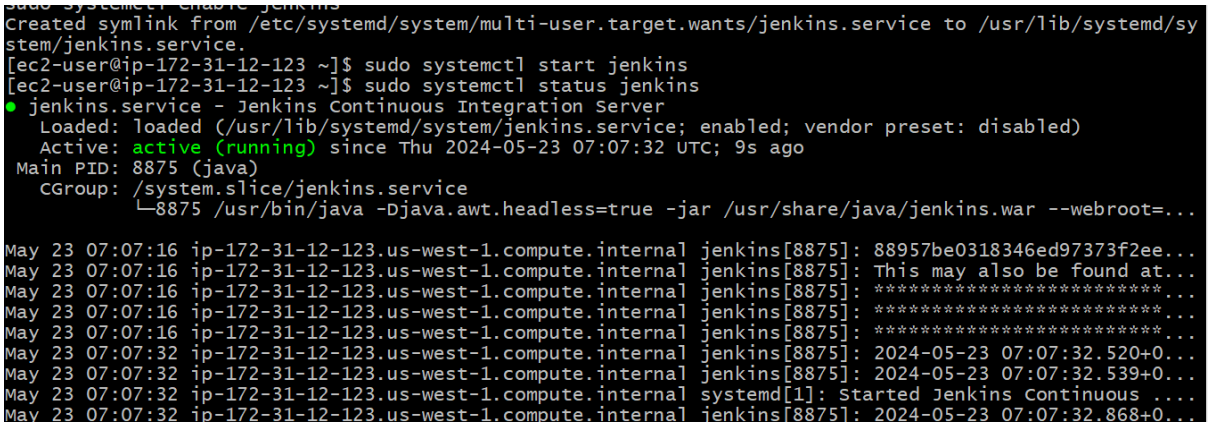
- After that install Jenkins using these commands as below
- Go to Google, search install Jenkins in Ubuntu server.
  - sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
  - echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]"\https://pkg.jenkins.io/debian-stable 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 >

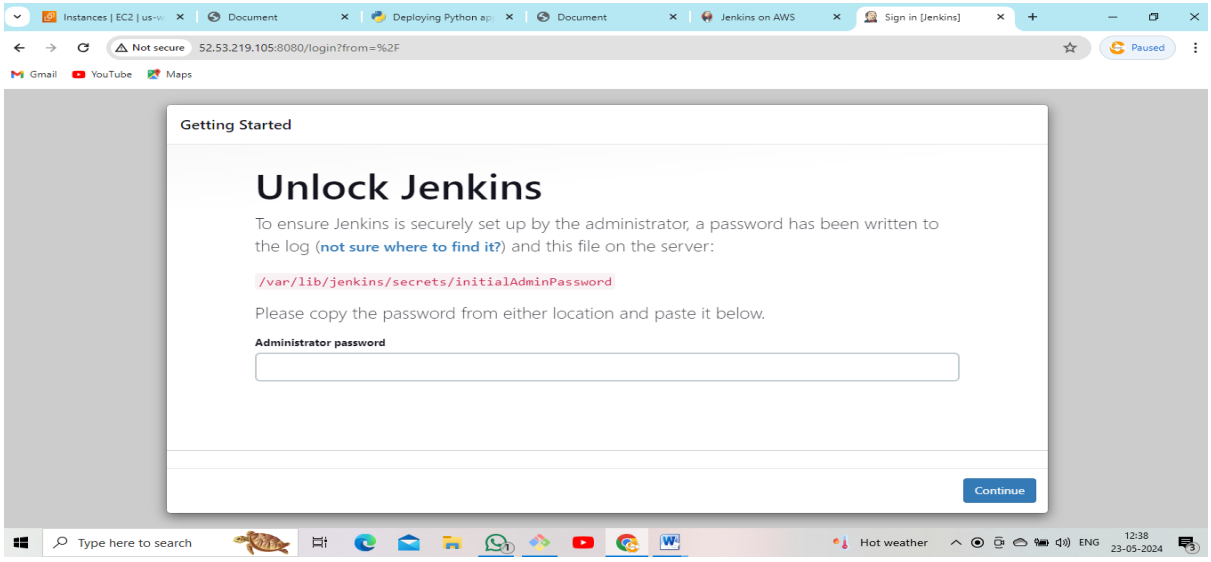
  < sudo systemctl enable jenkins >

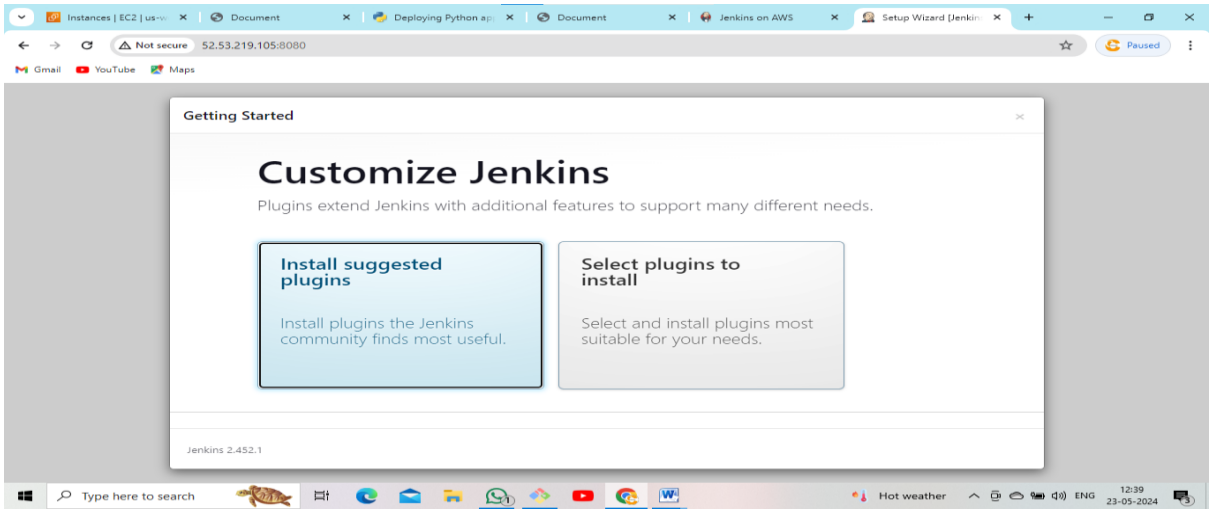
  < sudo systemctl status jenkins >

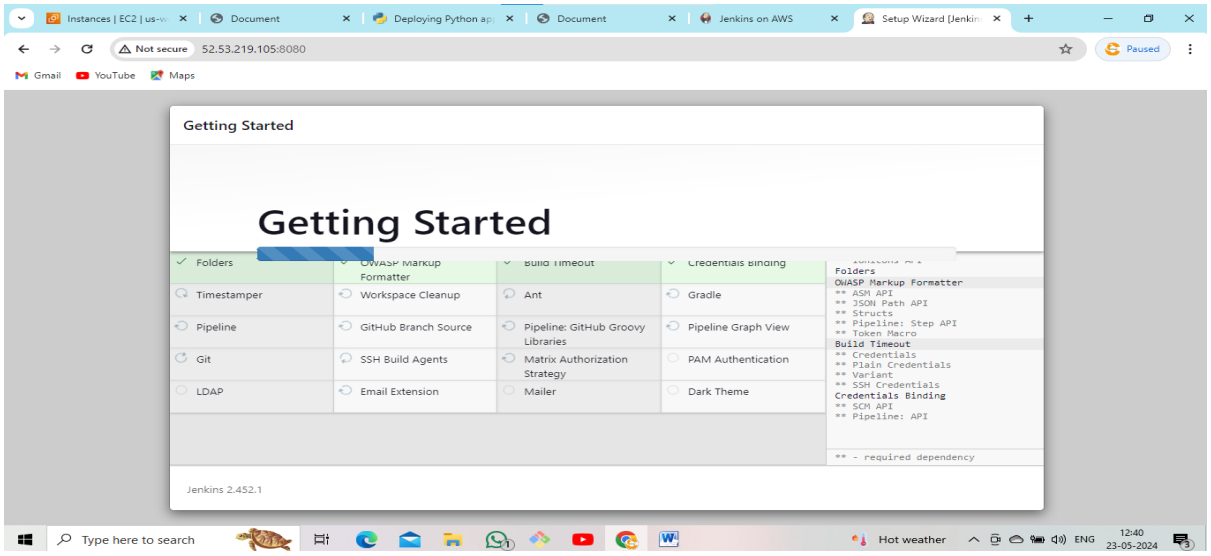


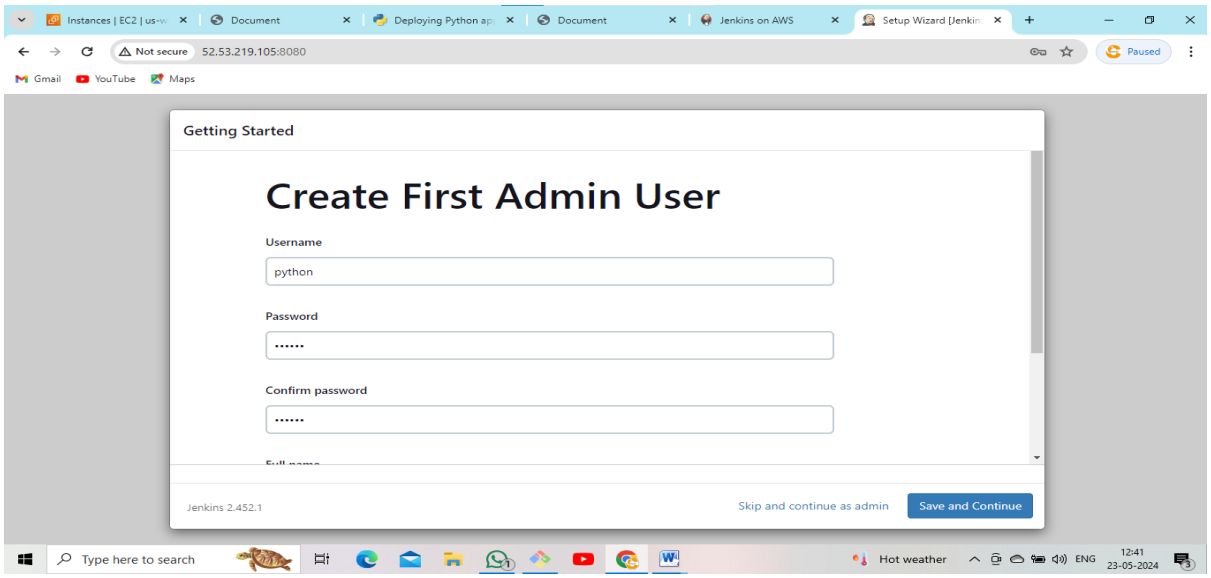
- After that go to EC2 instance copy public ip and jenkins port number paste it Google.

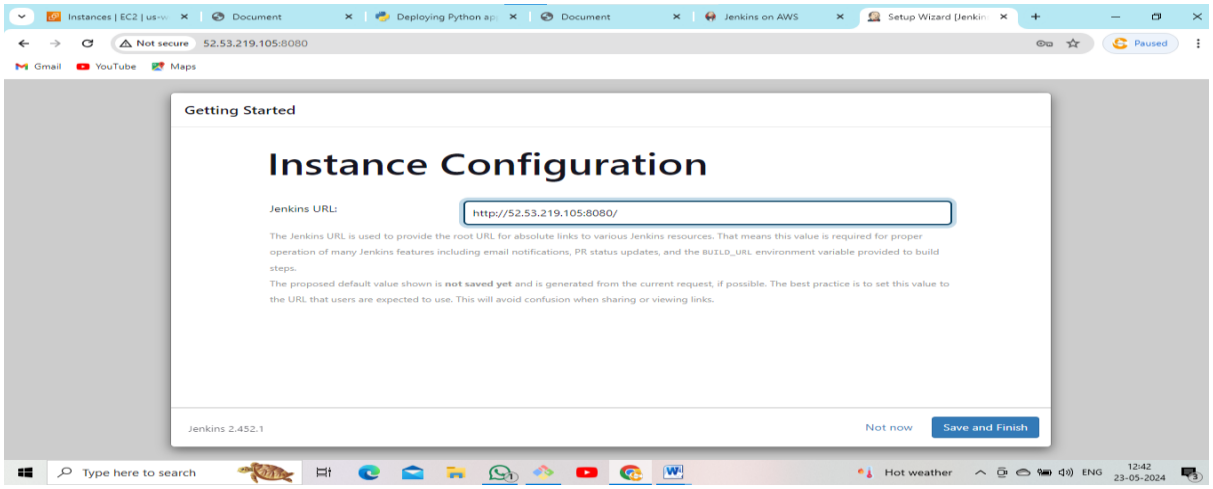
  < copy public ip : 8080 >

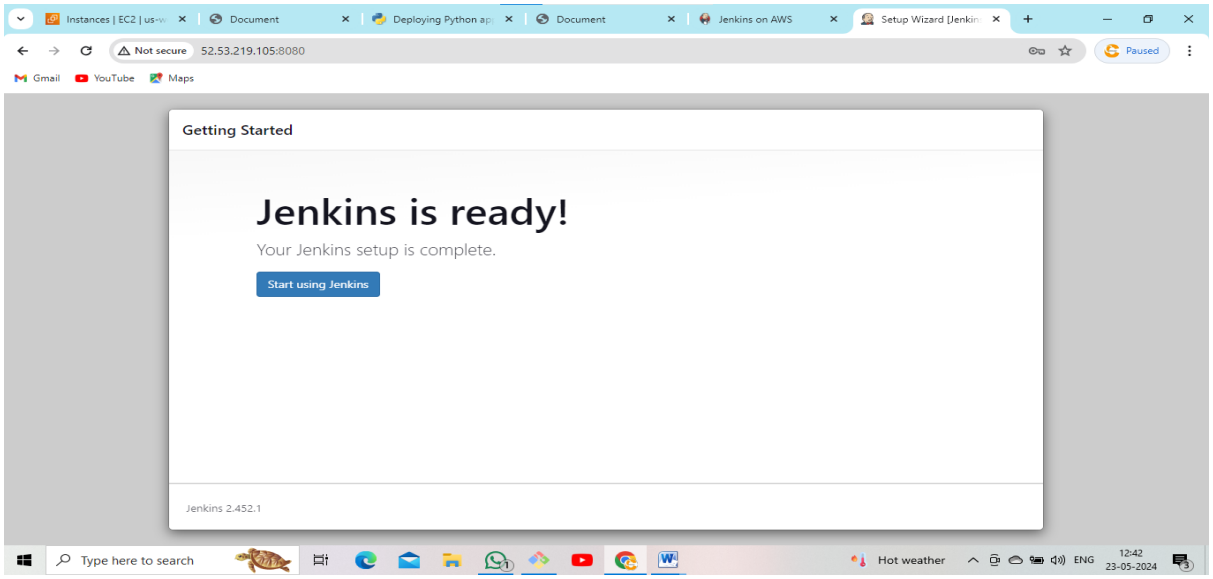


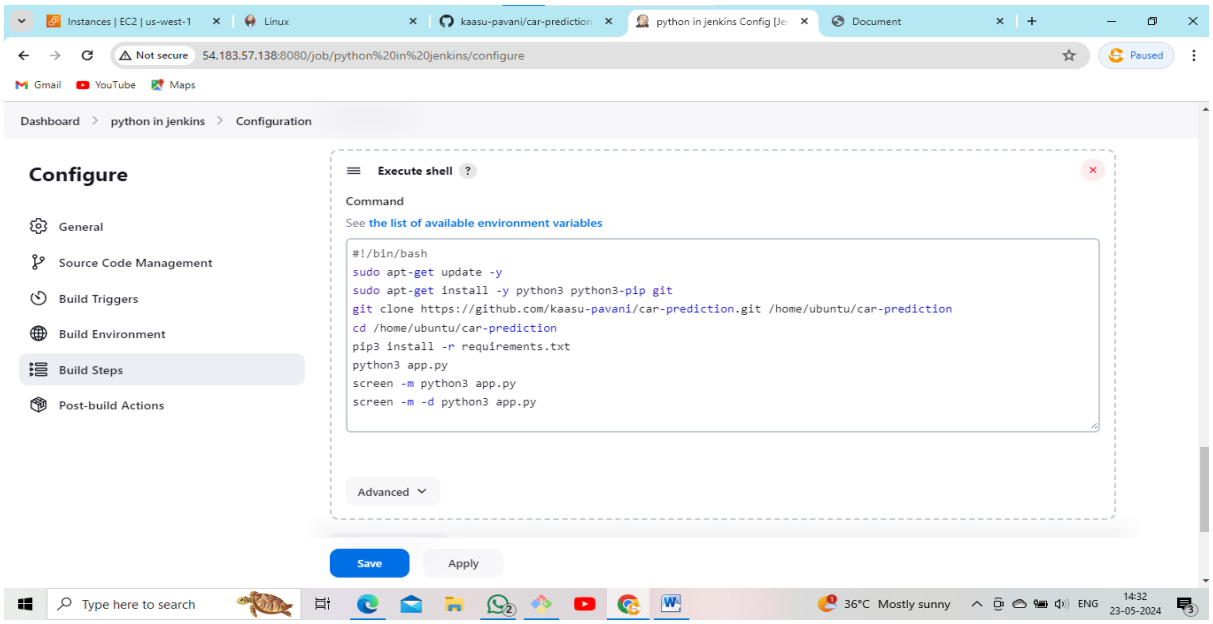

Not secure 52.53.219.105:8080
Gmail YouTube Maps
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.452.1
Type here to search
Hot weather
ENG
12:39
23-05-2024

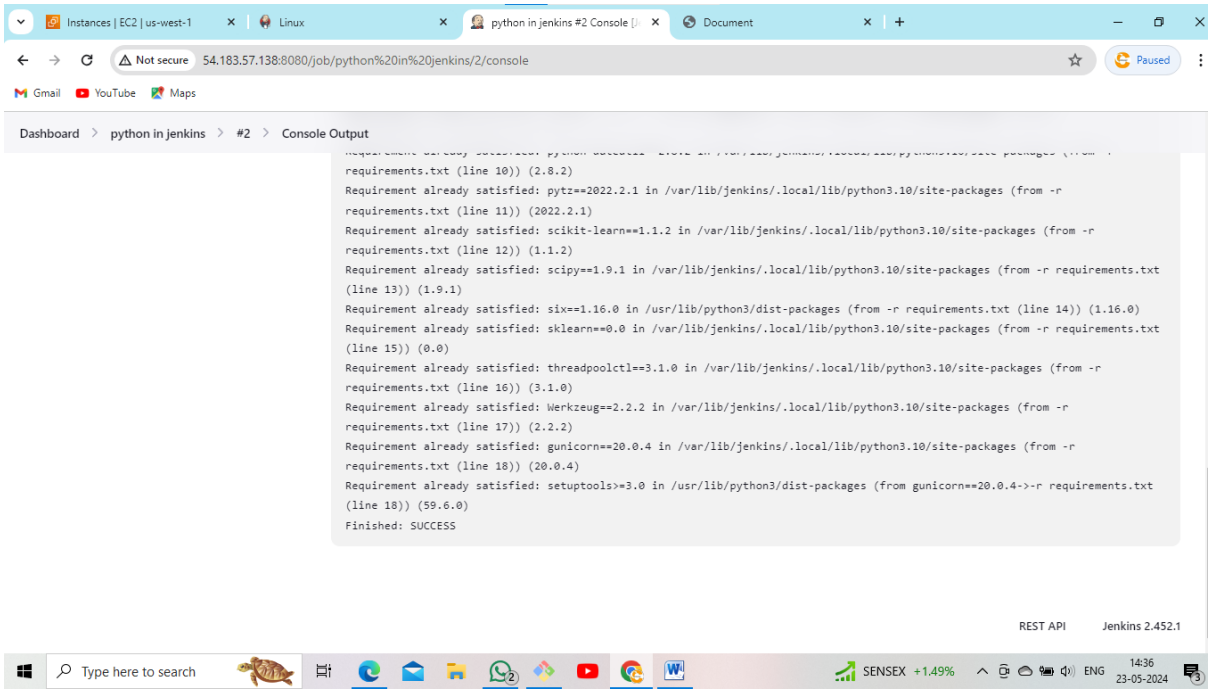

Not secure 52.53.219.105:8080
Gmail YouTube Maps
Getting Started
Getting Started
Folders
OWASP Markup Formatter
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
Jenkins 2.452.1
Type here to search
Hot weather
ENG
12:40
23-05-2024


Not secure 52.53.219.105:8080
Gmail YouTube Maps
Getting Started
Create First Admin User
Username
python
Password
Confirm password
Jenkins 2.452.1
Skip and continue as admin
Save and Continue
Type here to search
Hot weather
ENG
12:41
23-05-2024

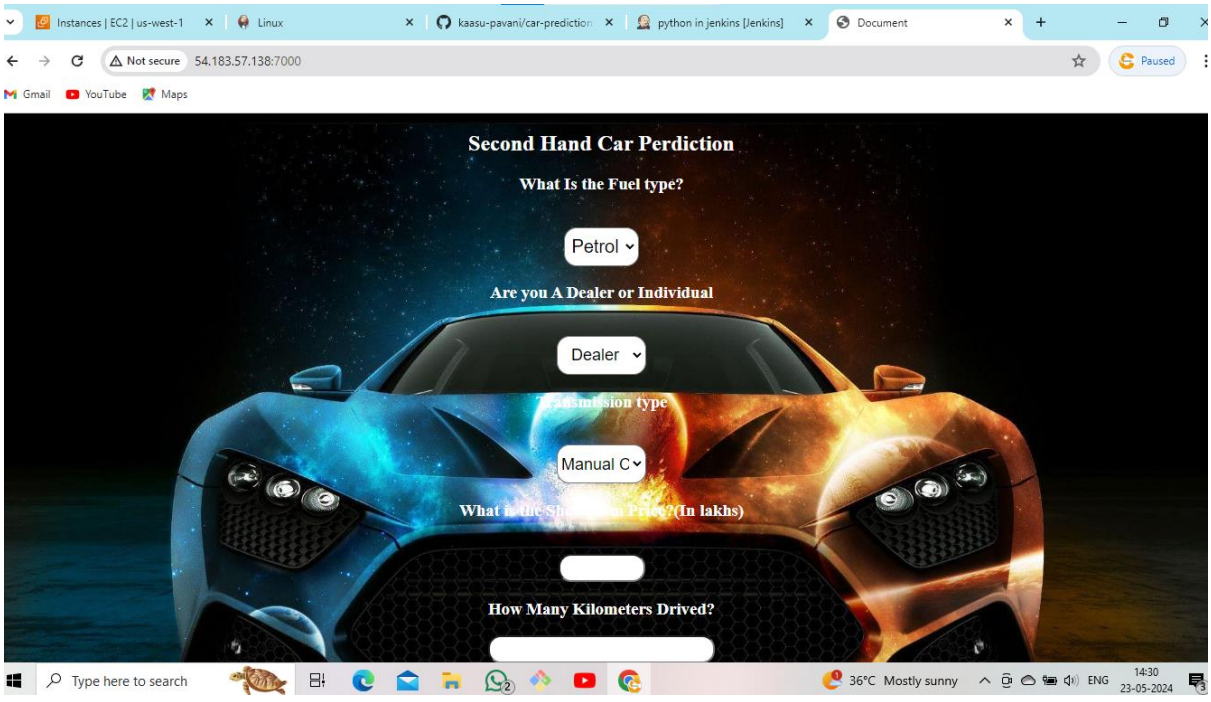

Getting Started
Instance Configuration
Jenkins URL:
http://52.53.219.105:8080/
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.
Jenkins 2.452.1
Not now
Save and Finish


Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.452.1


Dashboard
python in jenkins
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Execute shell
Command
See the list of available environment variables
#!/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
Advanced
Save
Apply



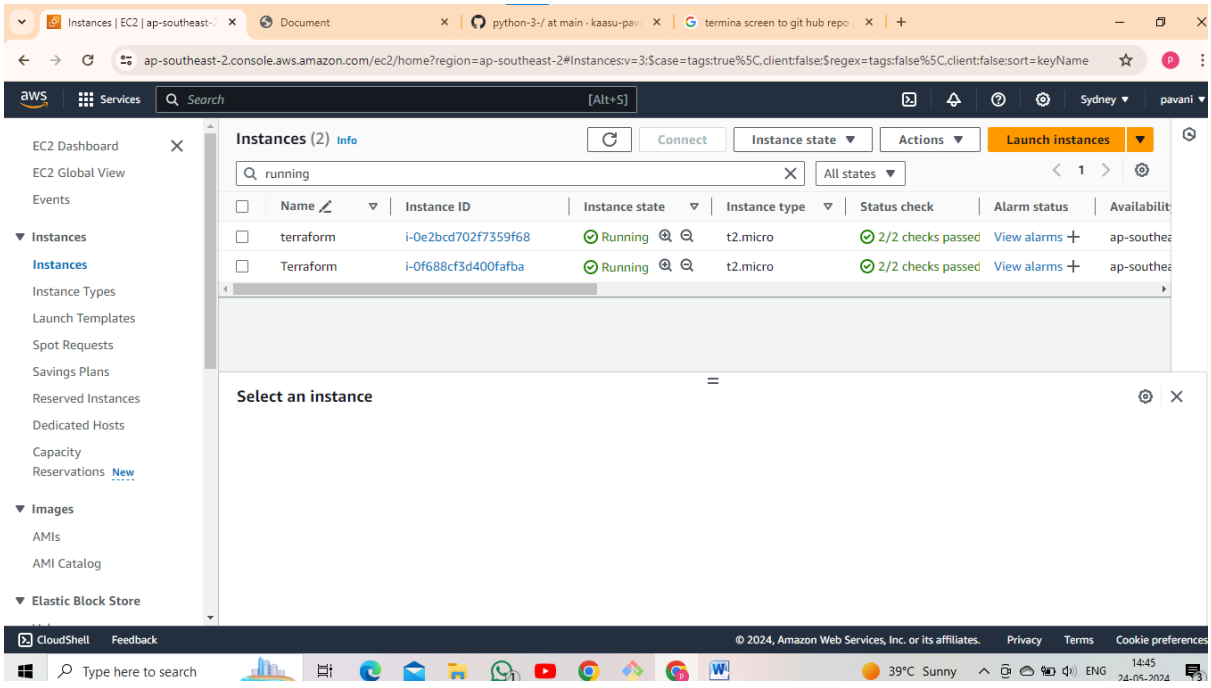
- Go to EC2 instance copy public ip and application port number paste it Google.



****

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.

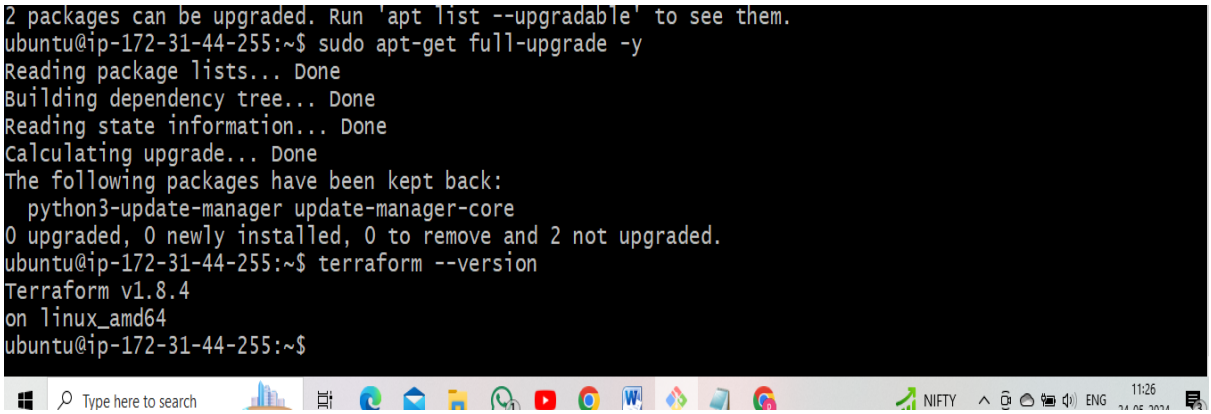
< sudo apt-get full-upgrade –y >

- Install terraform using these commands as

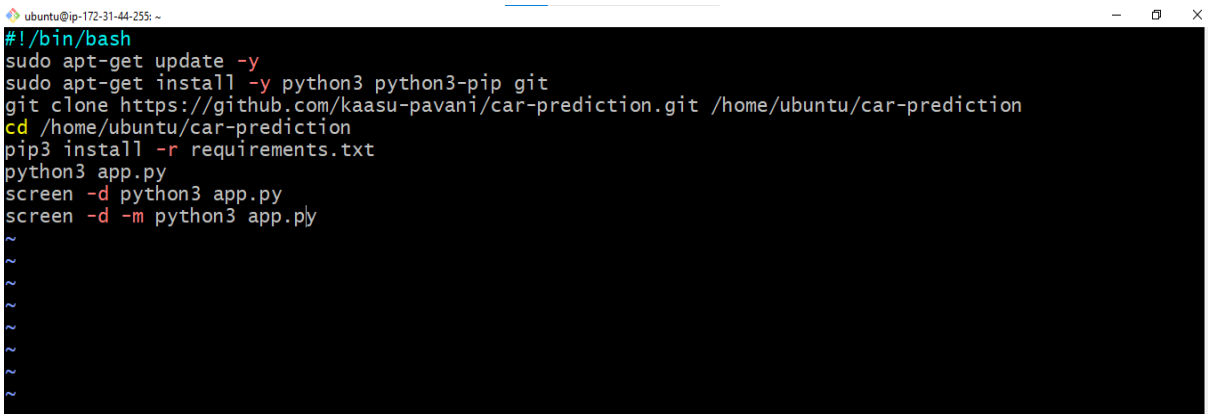
  - wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o/usr/share/keyrings/hashicorp-archive-keyring.gpg

    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



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



- 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

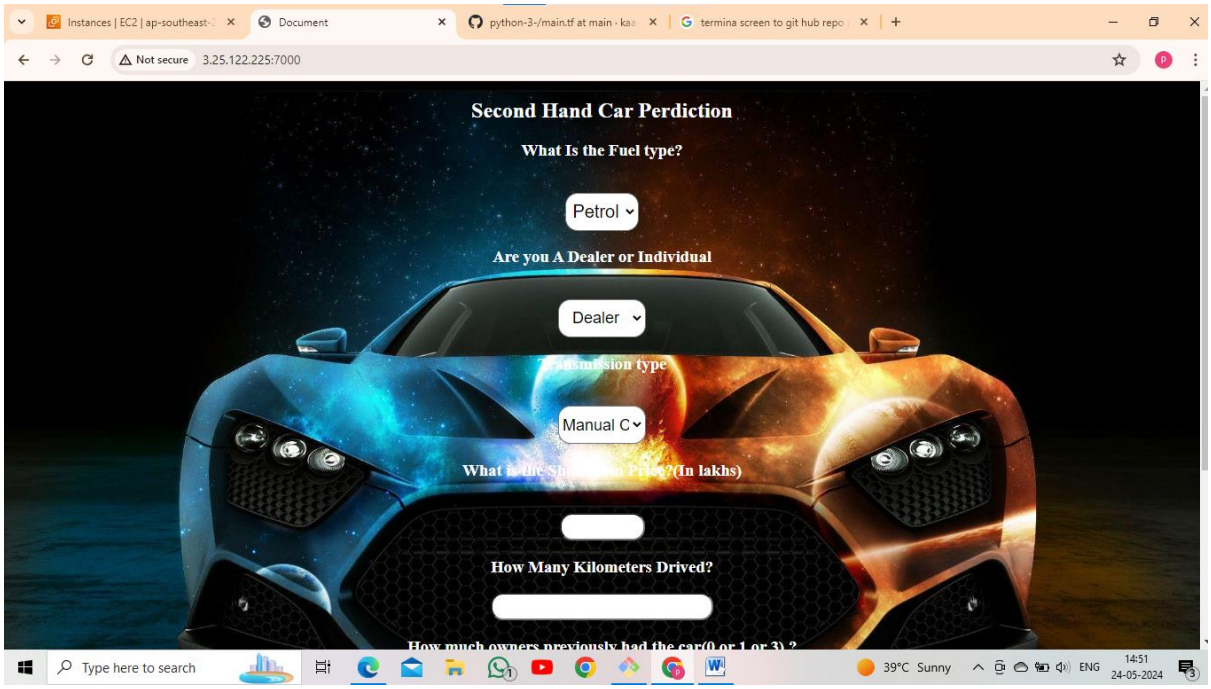
```
Terraform has been successfully initialized!

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.
```

```
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-09badd5065732dc59]
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.

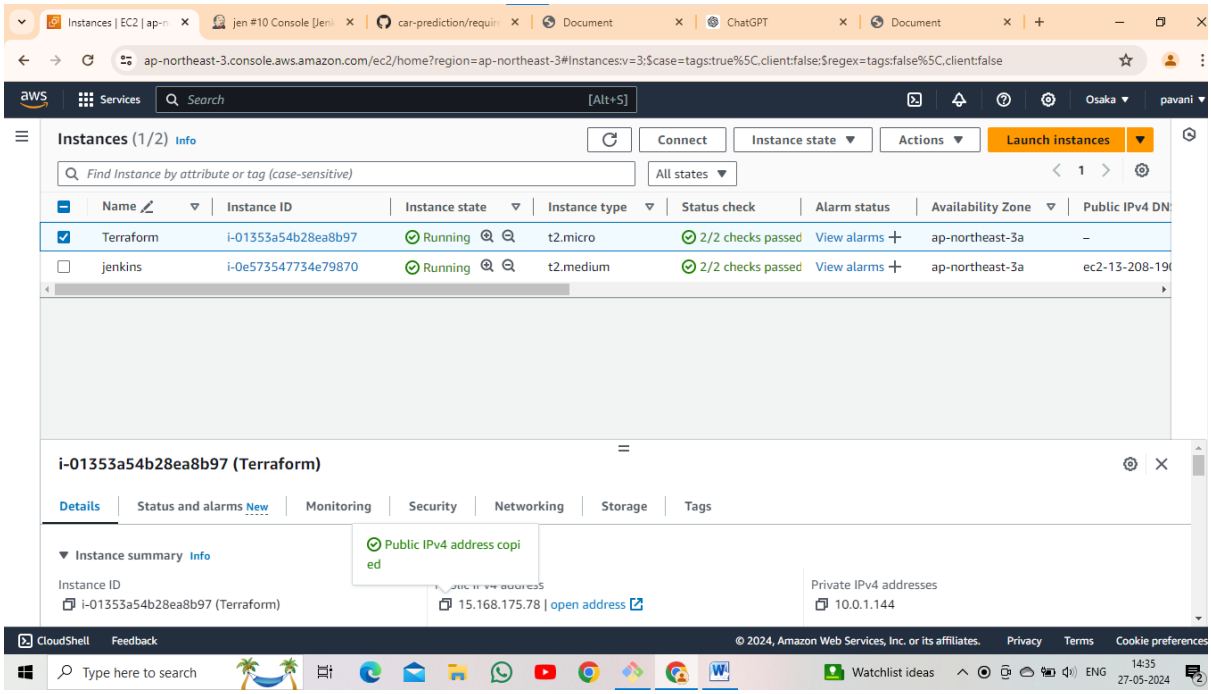
  < copy public ip : port number >


Not secure 3.25.122.225:7000
Second Hand Car Perdiction
What Is the Fuel type?
Petrol
Are you A Dealer or Individual
Dealer
Transmission type
Manual
(In lakhs)
How Many Kilometers Drived?
Type here to search
39°C Sunny
ENG
14:51
24-05-2024

****

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 >

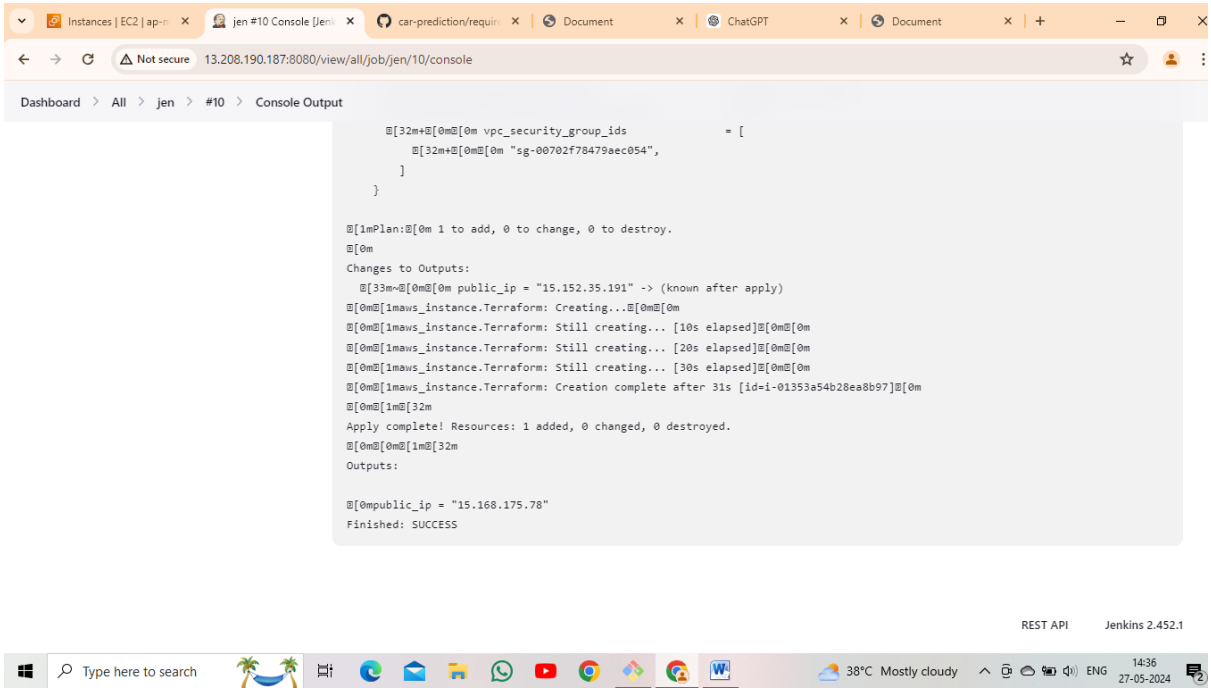
- After that install java.

  < sudo yum –y install java* >

- Install Jenkins.

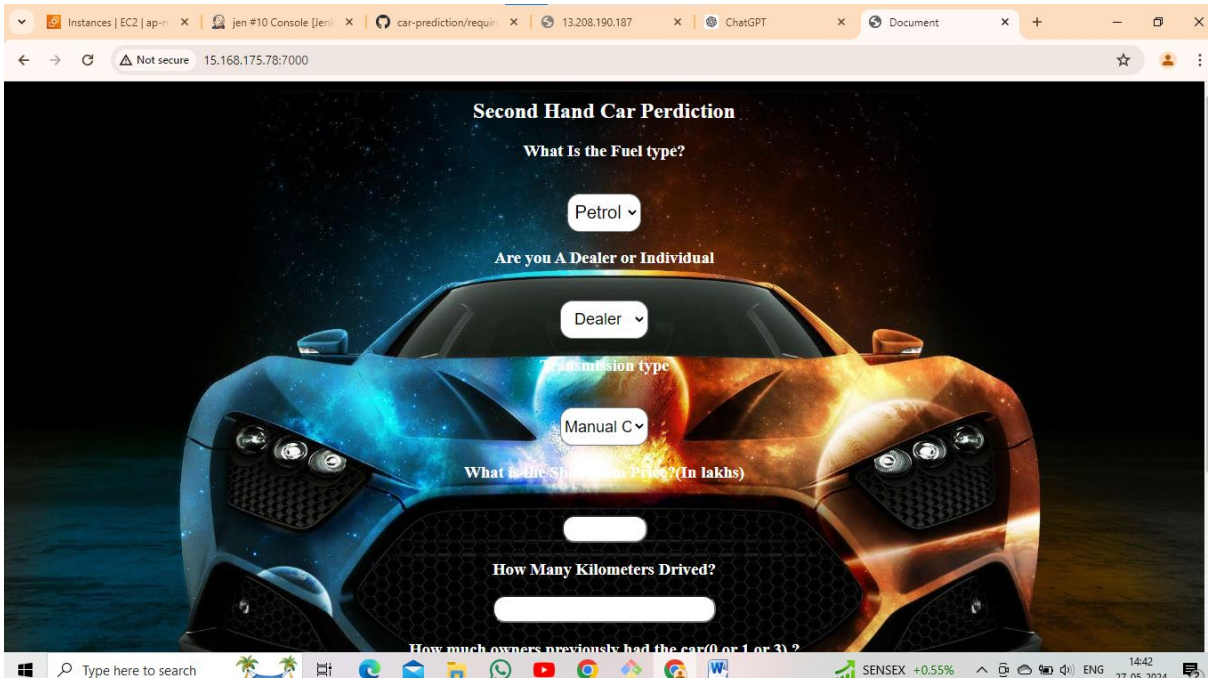
  NOTE: Jenkins installation is above steps (4th step).

```
#!/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
```



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

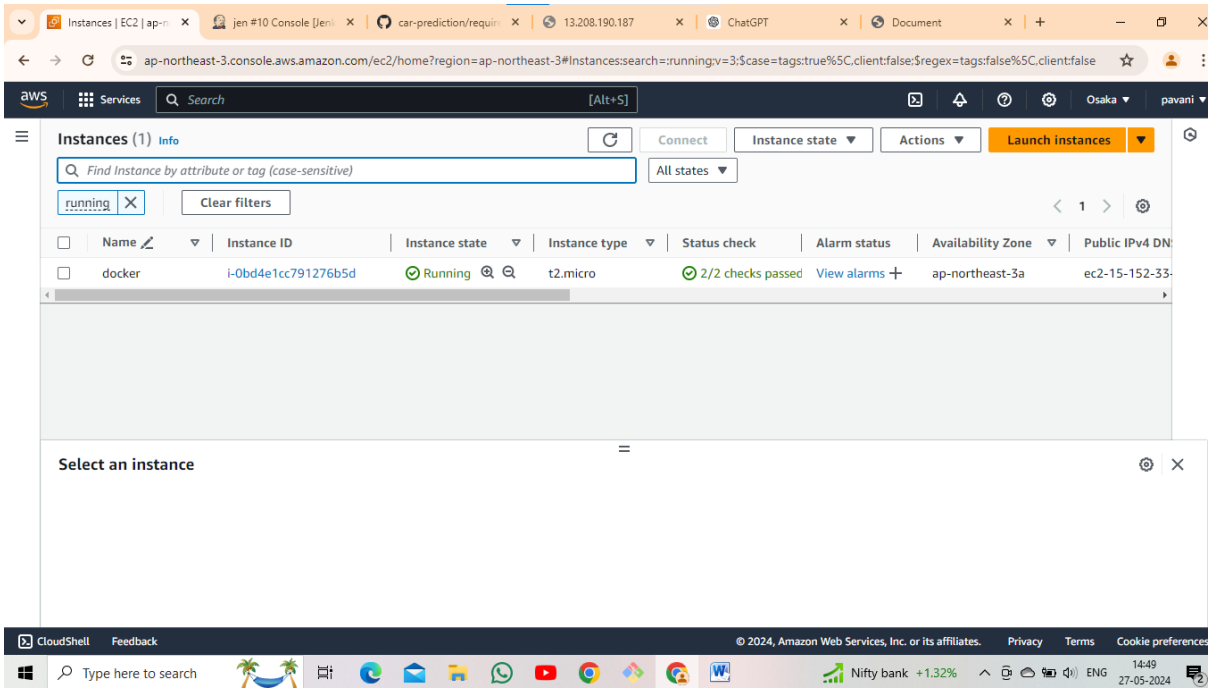
  < copy public ip : port number >


Instances | EC2 | ap-
jen #10 Console [Jen
car-prediction/requir
13.208.190.187
ChatGPT
Document
Not secure
15.168.175.78:7000
Second Hand Car Perdiction
What Is the Fuel type?
Petrol
Are you A Dealer or Individual
Dealer
Transmission type
Manual C
What is the Sh
Price?(In lakhs)
How Many Kilometers Drived?
How much owners previously had the car(0 or 1 or 3) ?
Type here to search
SENSEX +0.55%
ENG
14:42

****

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.

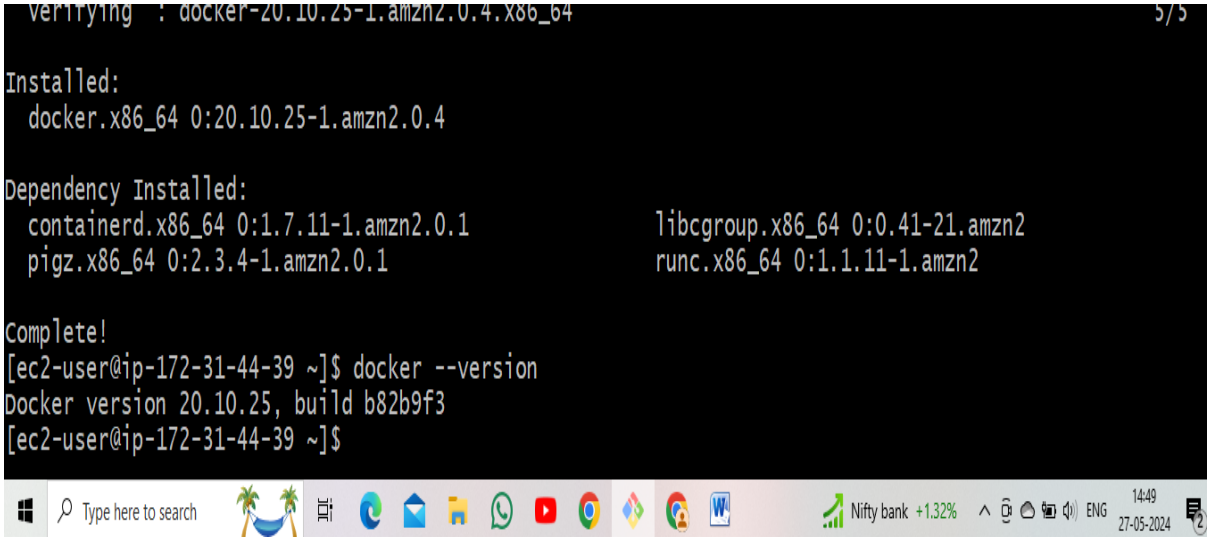


- Go to Git Bash.
- Update the machine using these following command as

  < sudo yum update >

- Install docker using these command as

  < sudo yum –y install docker >

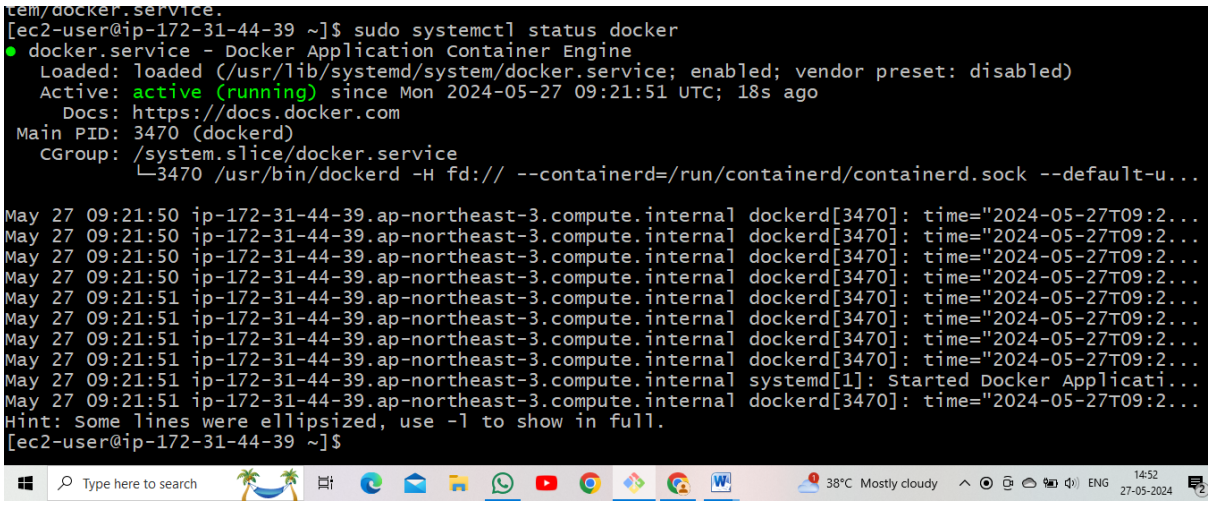


- Docker enable, start, status using these commands as

  < sudo systemctl start docker >

  < sudo systemctl enable docker >

  < sudo systemctl status docker >



- 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

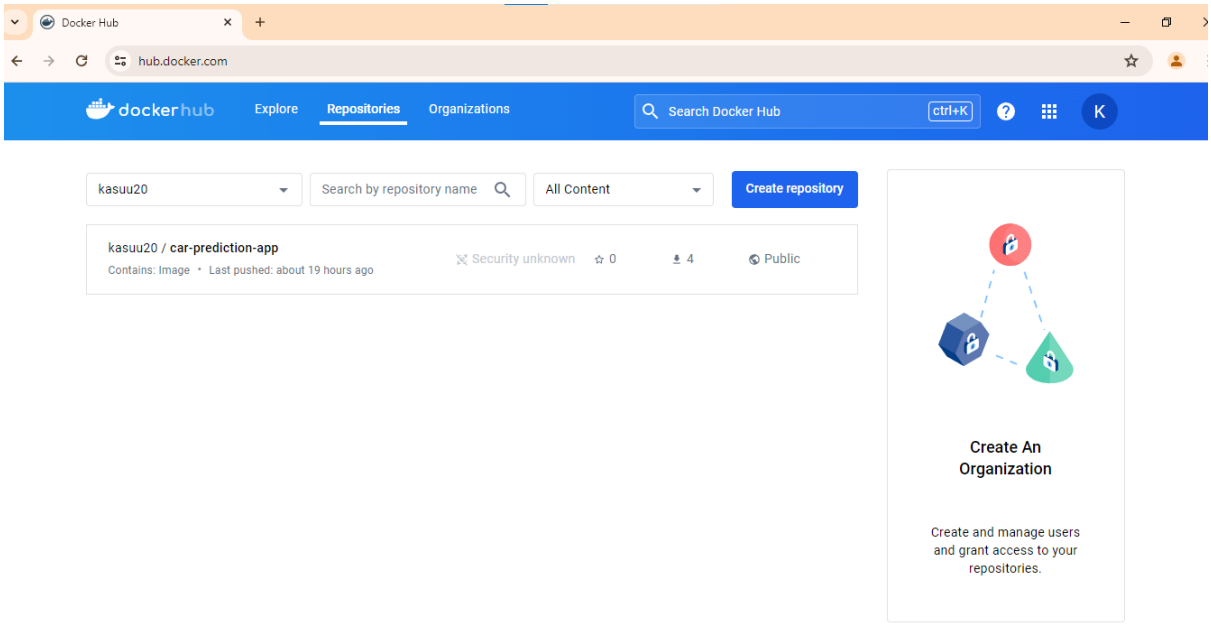
  < sudo chmod 666 /var/run/docker.sock >

- Docker build using these command as

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

- Docker push using these command as

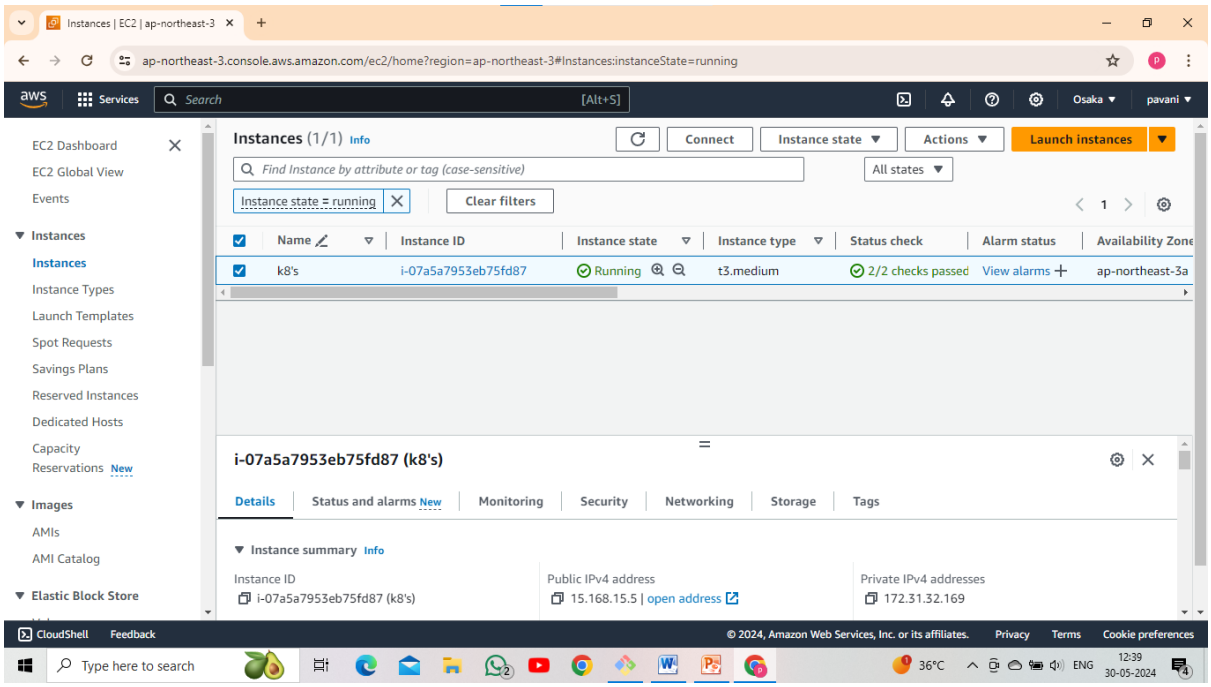
  < sudo docker push kasuu20/ car-prediction >


Docker Hub
hub.docker.com
dockerhub
Explore
Repositories
Organizations
Search Docker Hub
ctrl+K
K
kasuu20
Search by repository name
All Content
Create repository
kasuu20 / car-prediction-app
Contains: Image • Last pushed: about 19 hours ago
Security unknown
0
4
Public
Create An Organization
Create and manage users and grant access to your 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/kubern
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.
EOF
```

Note:

- After that install kubectl using these command as

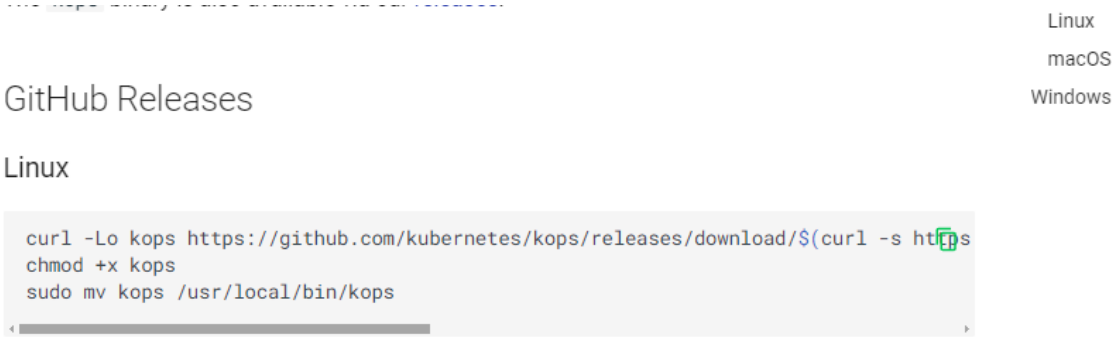
  < sudo yum install –y kubectl >

- 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.



- 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

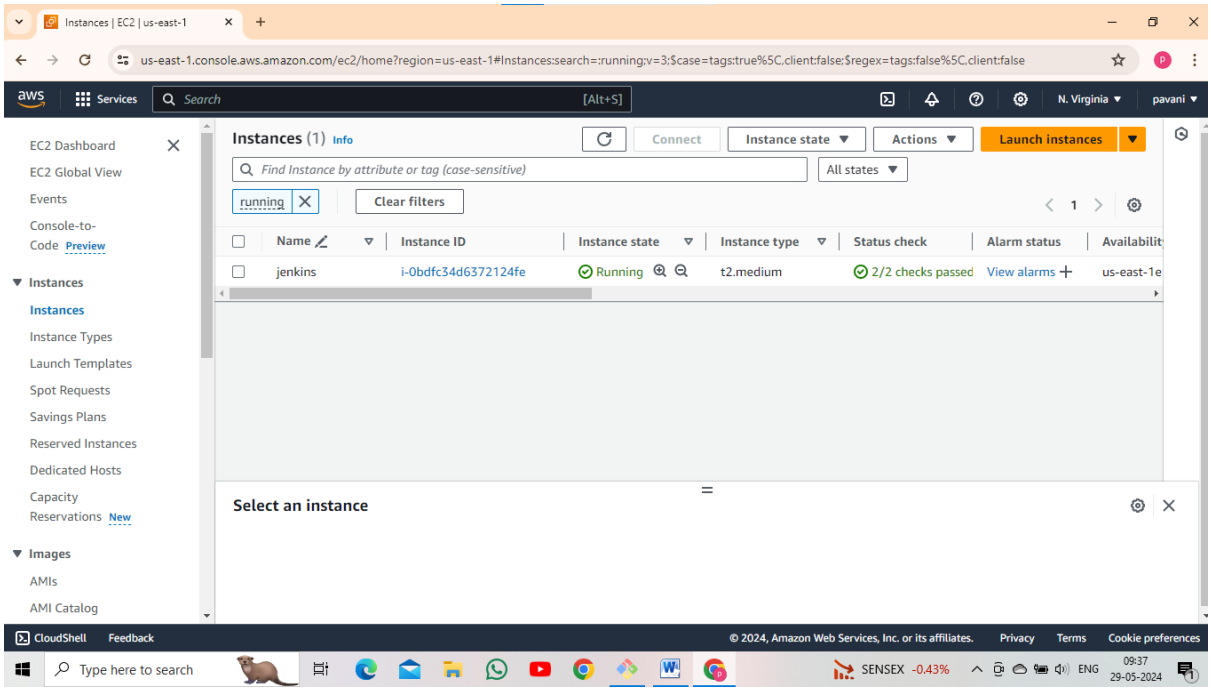
  < kubectl apply –f deployment.yaml >

  < 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.

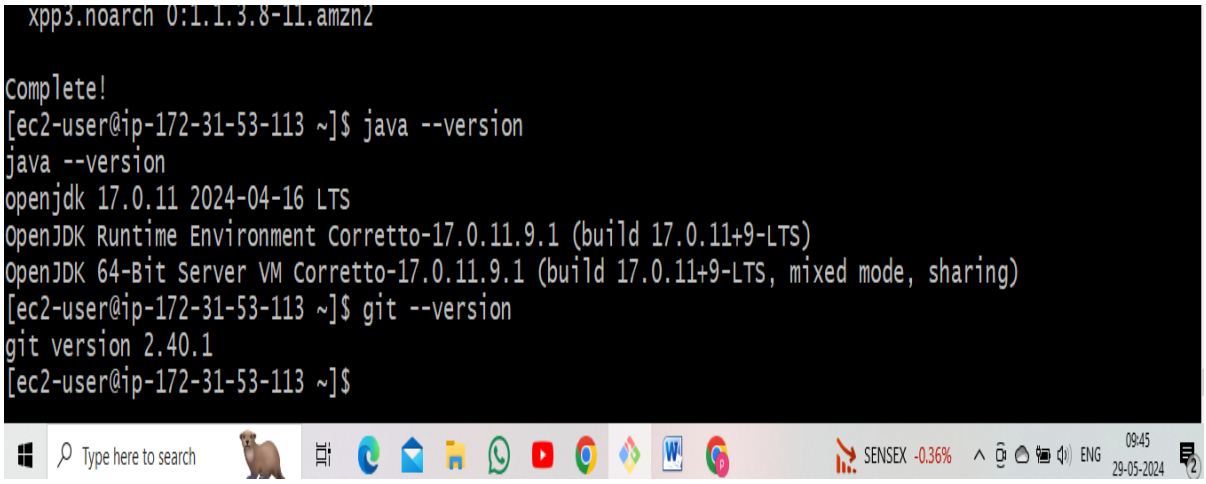


- Add the IAM role into the created EC2 instance.
- Go to Git Bash.
- Update the machine using these following command

  < sudo yum update >

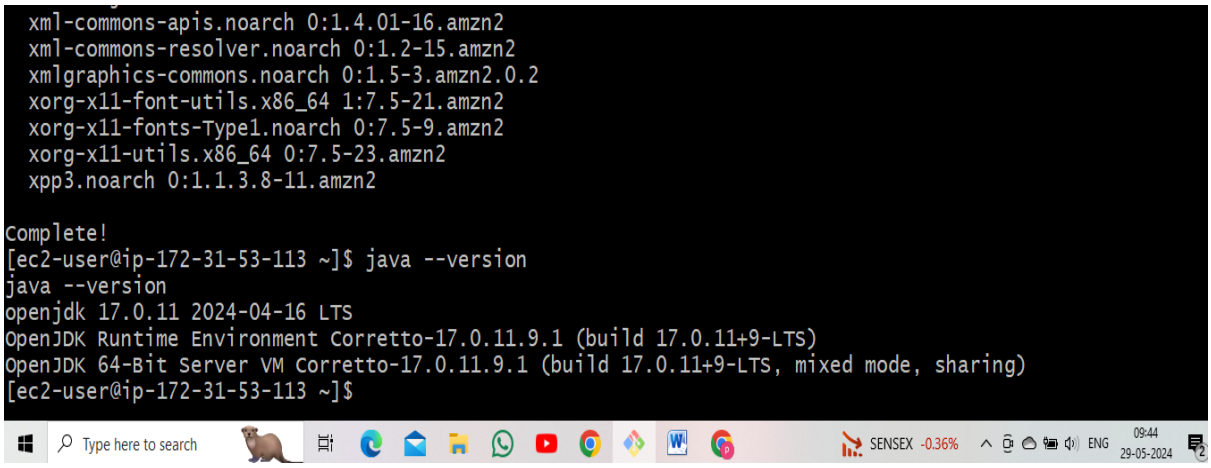
- Install git.

< sudo yum –y install git >



- Install java.

< sudo yum –y install java* >



- Install Jenkins.

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

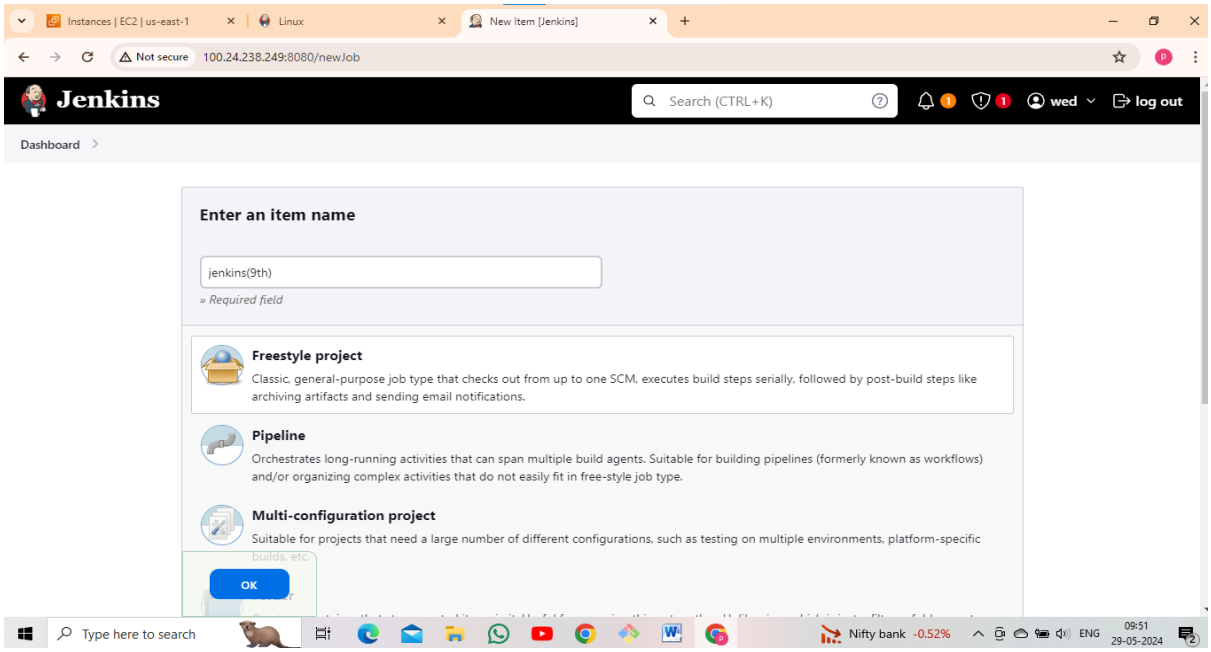
```
[ec2-user@ip-172-31-53-113 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-53-113 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
   Active: active (running) 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 ~]$
```

- Give sudo permissions for the Jenkins.

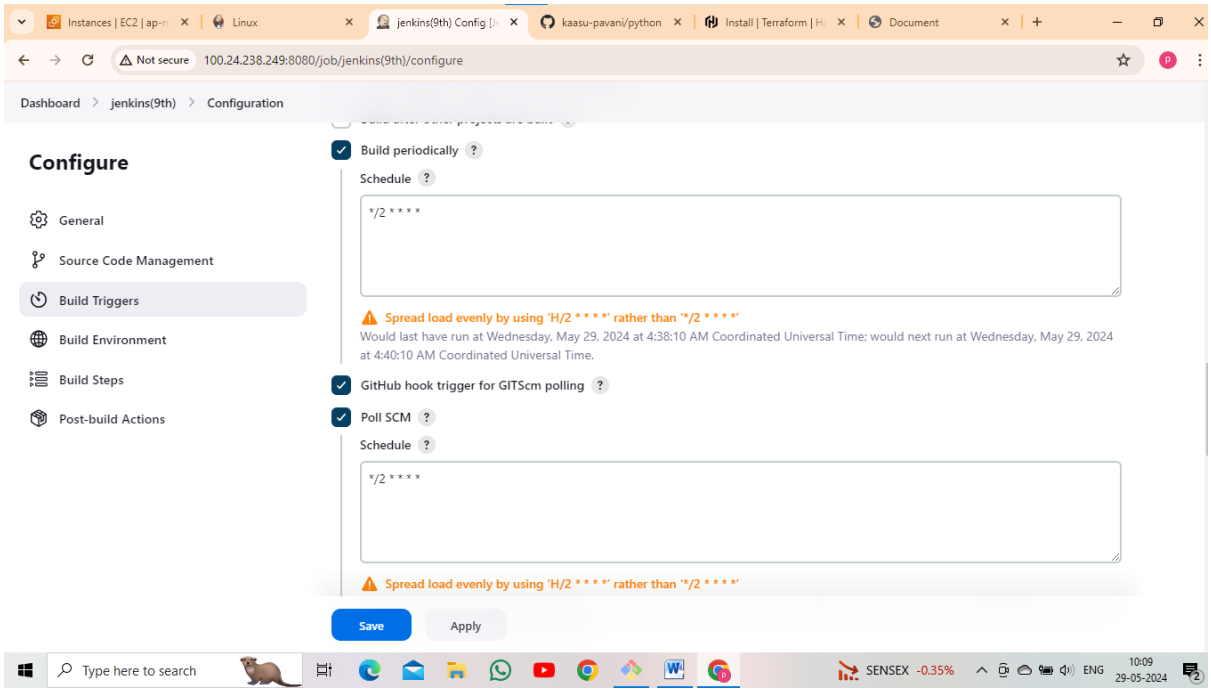
  < sudo visudo >

```
## 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
```

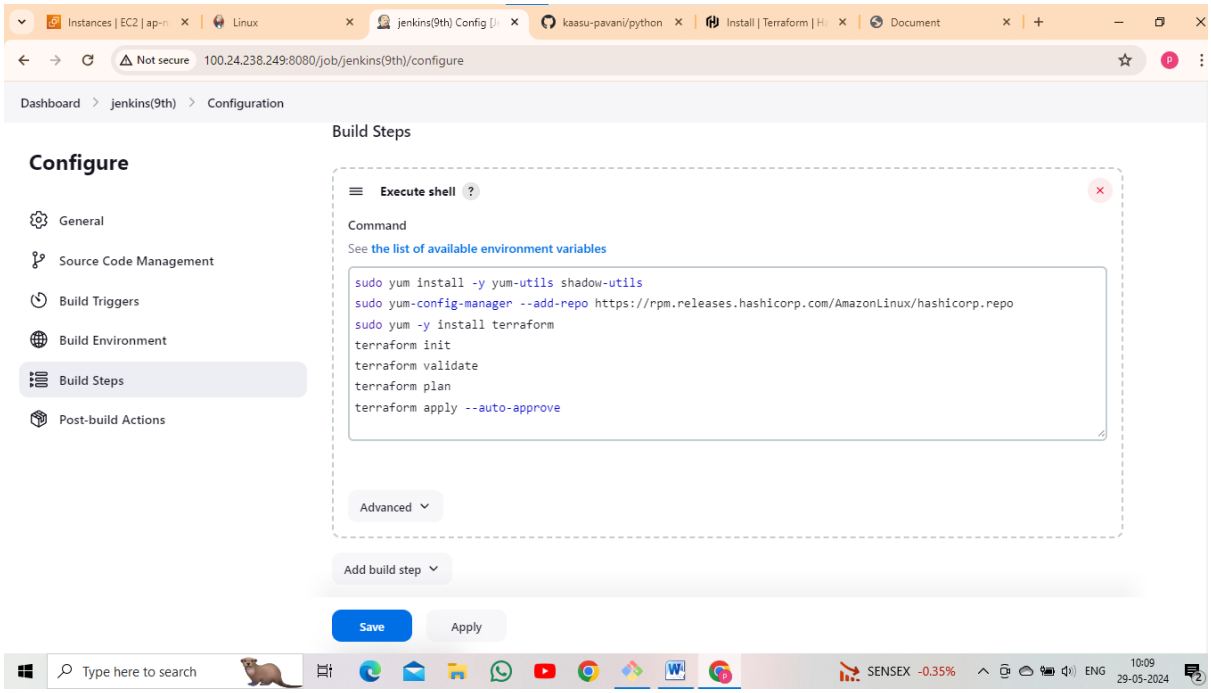
- Go to Jenkins page.
- Create job.



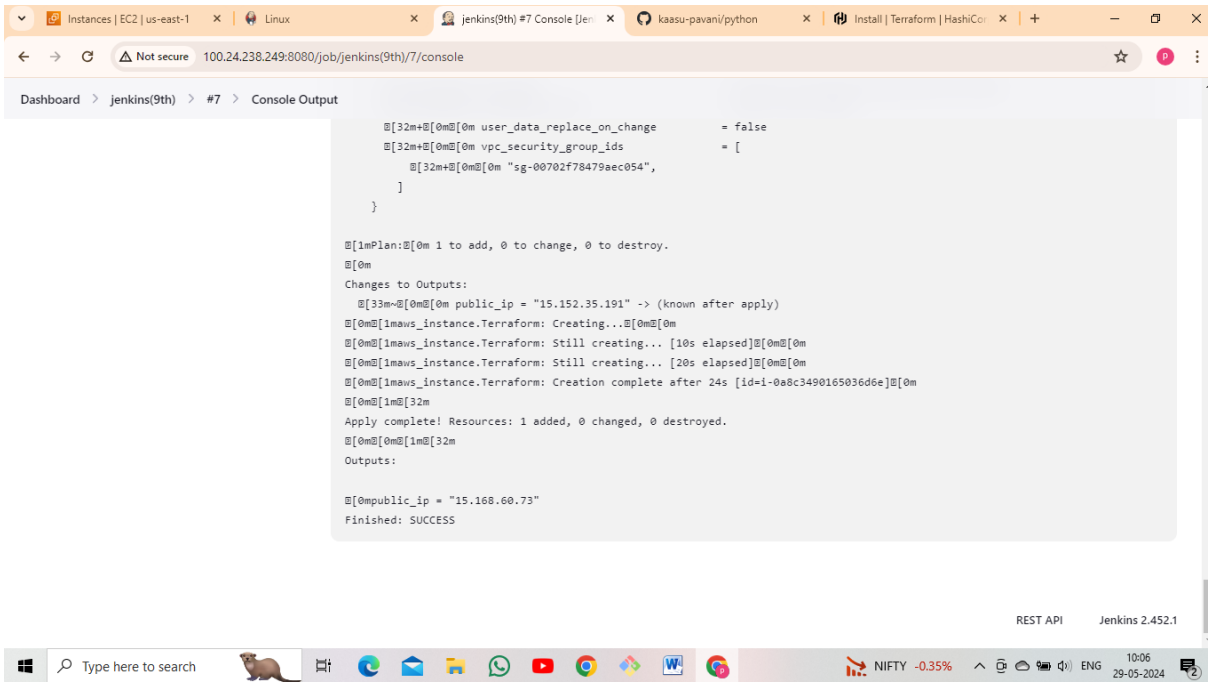
- Give the Git Hub URL.
- After that select Build periodically and poll SCM. give the */2 * * * *(crown syntax).



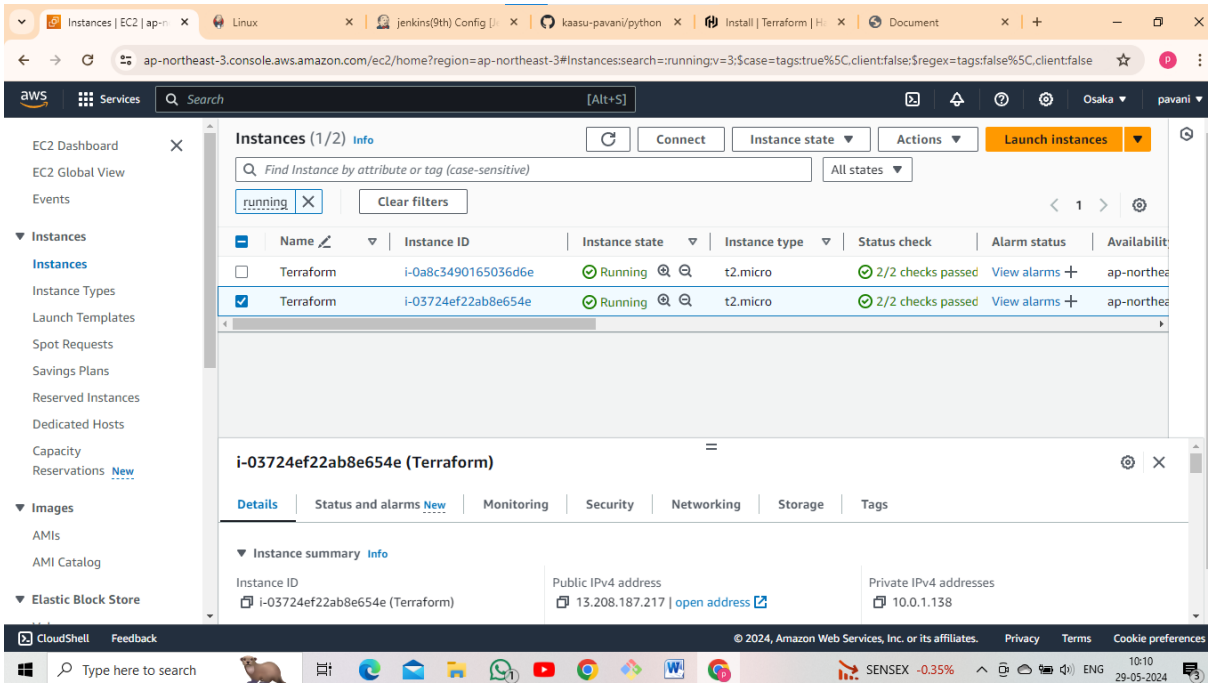
- After that select execute shell and give the terraform installation and terraform execution commands.



After that click the apply and save and Build now.

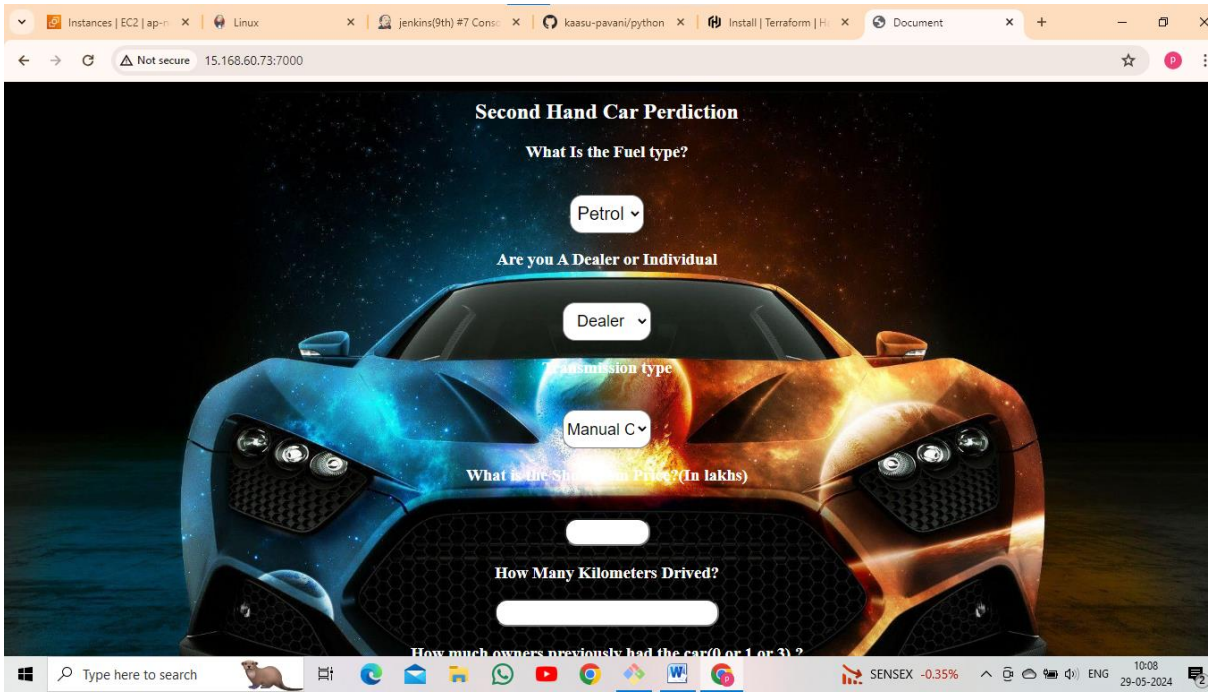


Now go to EC2 instance. check the instance, terraform instance is created.



After that copy public ip and paste it Google.

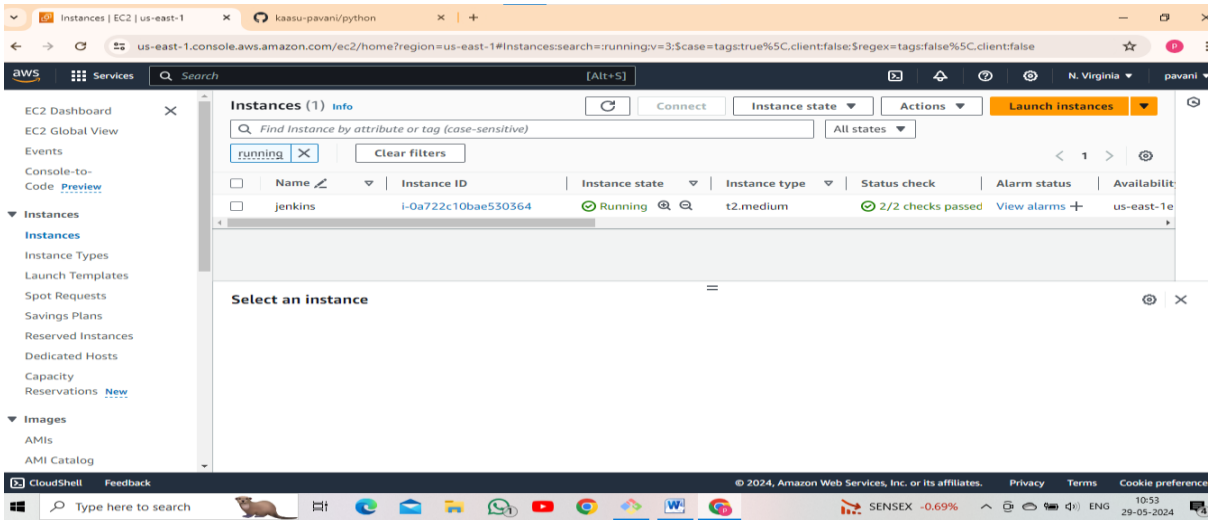
< copy public ip : application port number >



****

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

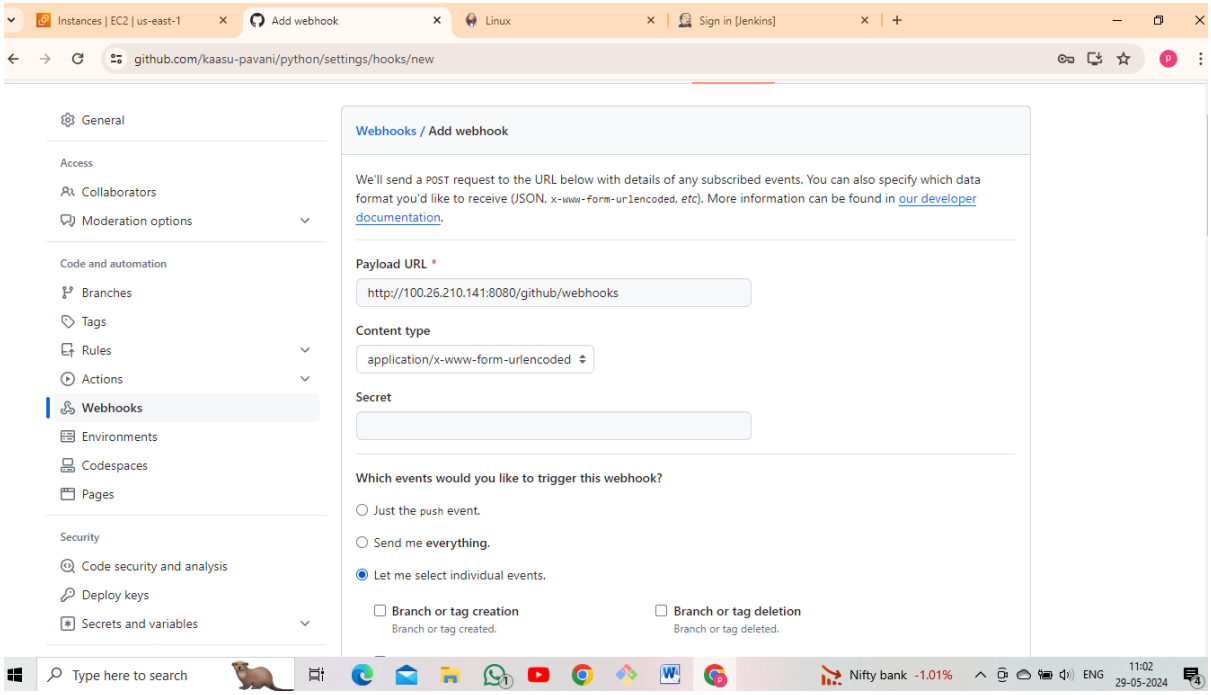
```
  xerces-j2.noarch 0:2.11.0-17.amzn2
  xml-commons-apis.noarch 0:1.4.01-16.amzn2
  xml-commons-resolver.noarch 0:1.2-15.amzn2
  xmlgraphics-commons.noarch 0:1.5-3.amzn2.0.2
  xorg-x11-font-utils.x86_64 1:7.5-21.amzn2
  xorg-x11-fonts-Type1.noarch 0:7.5-9.amzn2
  xorg-x11-utils.x86_64 0:7.5-23.amzn2
  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.

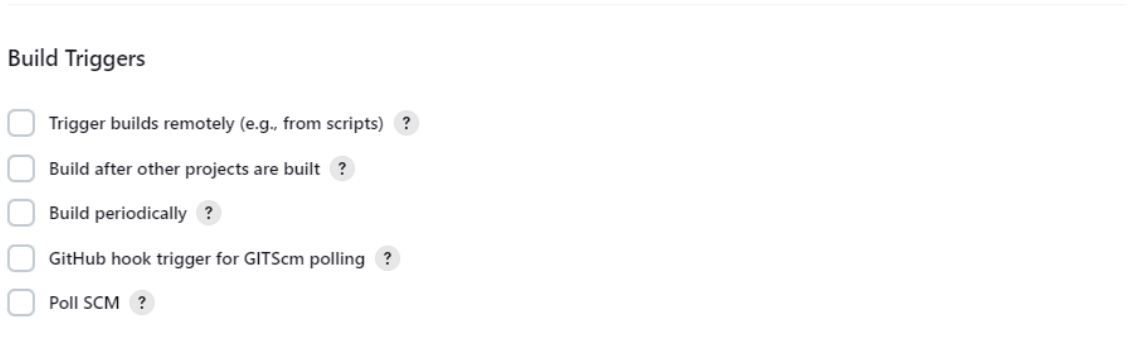
  NOTE: Jenkins installation process is above steps (4th step).

```
[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 05:29:21 ip-172-31-49-74.ec2.internal jenkins[5404]: ************************************...**
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 ~]$
```

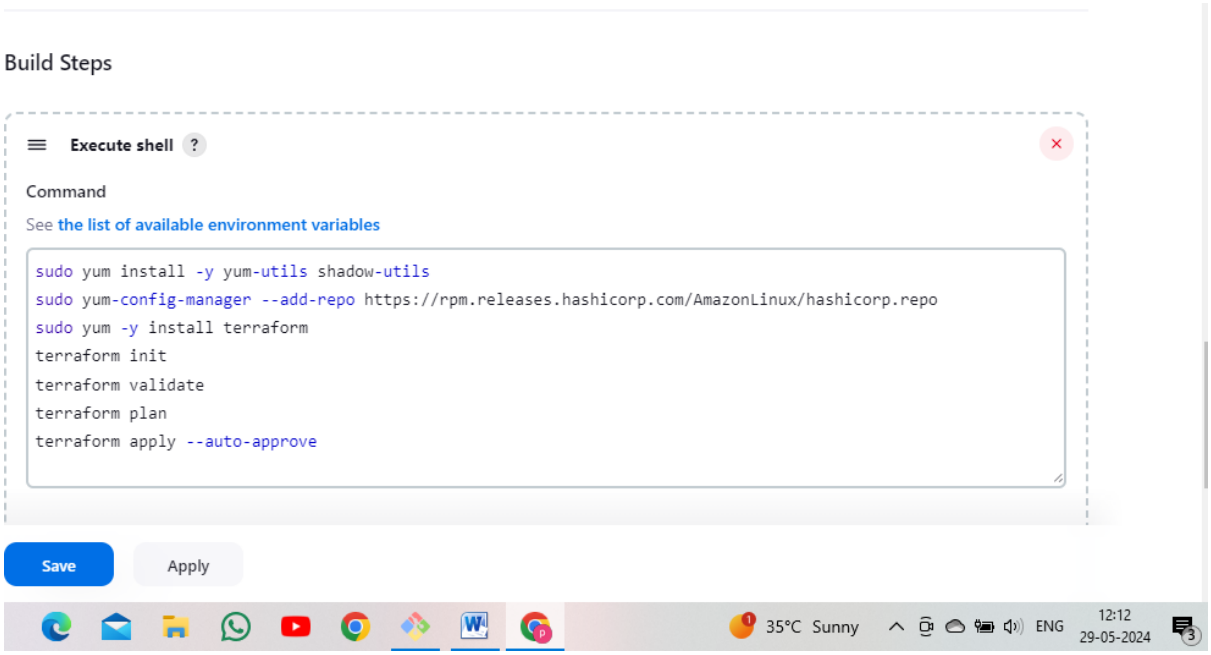
- Go to Git Hub repo.
- Create webhooks.



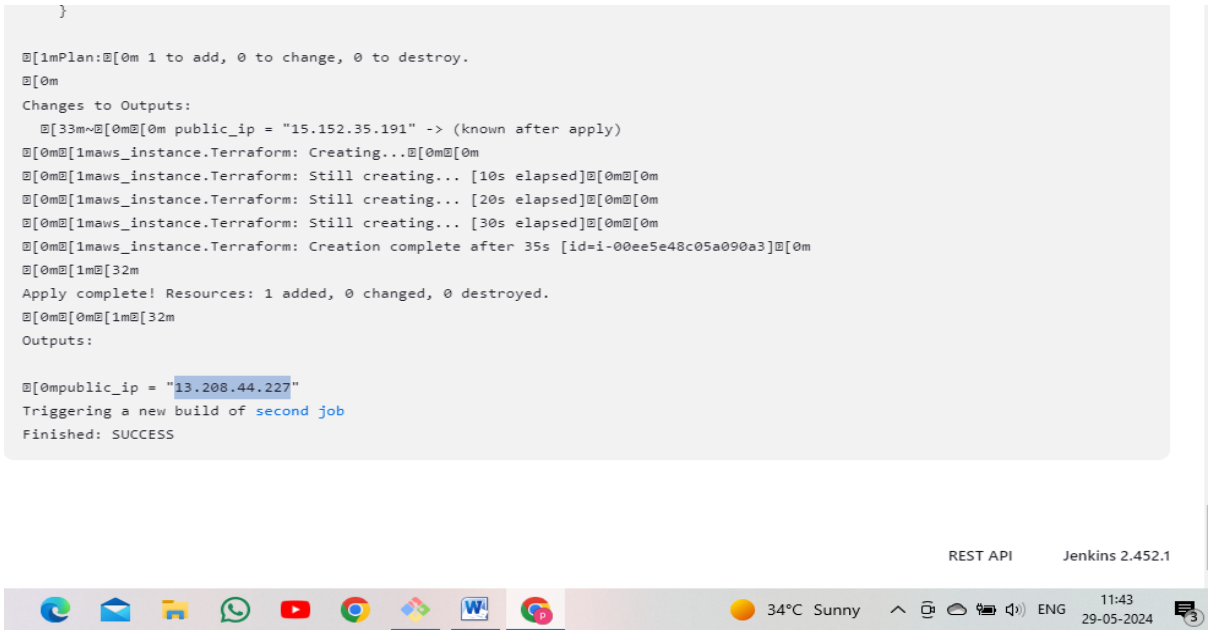
- Go to Jenkins page.
- Create 3 jobs (using pipeline creation).
- After that creating first job.
- Give the Git Hub URL.
- Select Build after other projects are built.



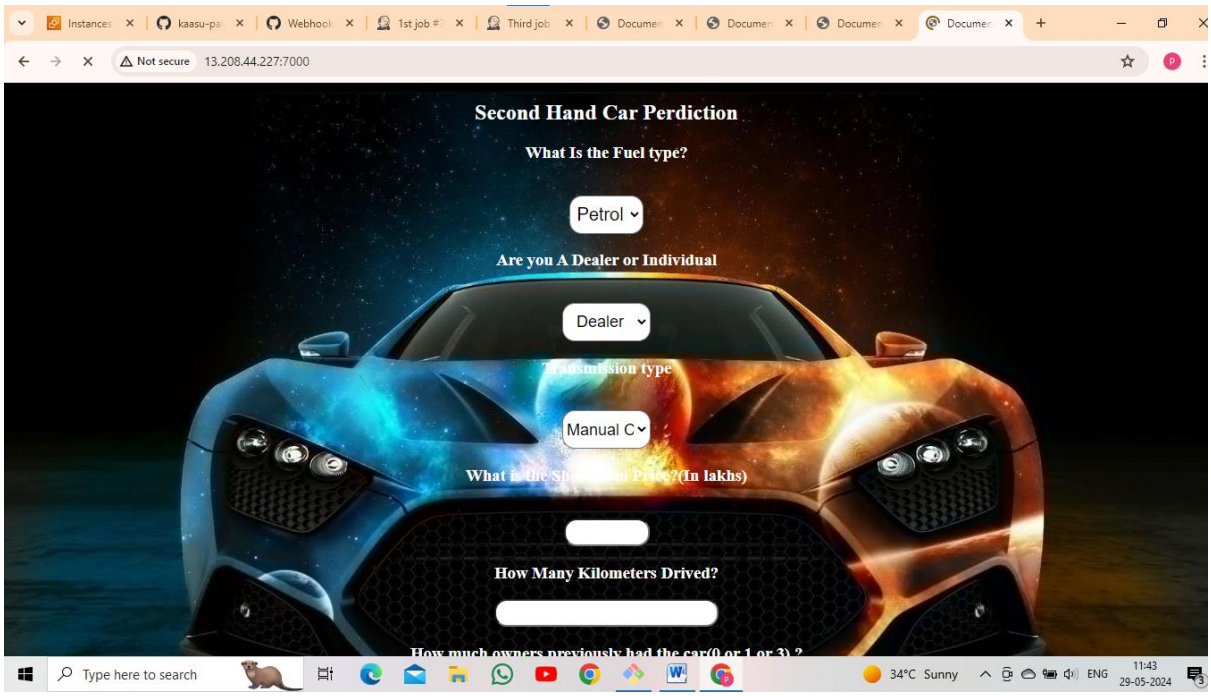
- After that execute shell give the installation of terraform commands and execution of terraform commands.



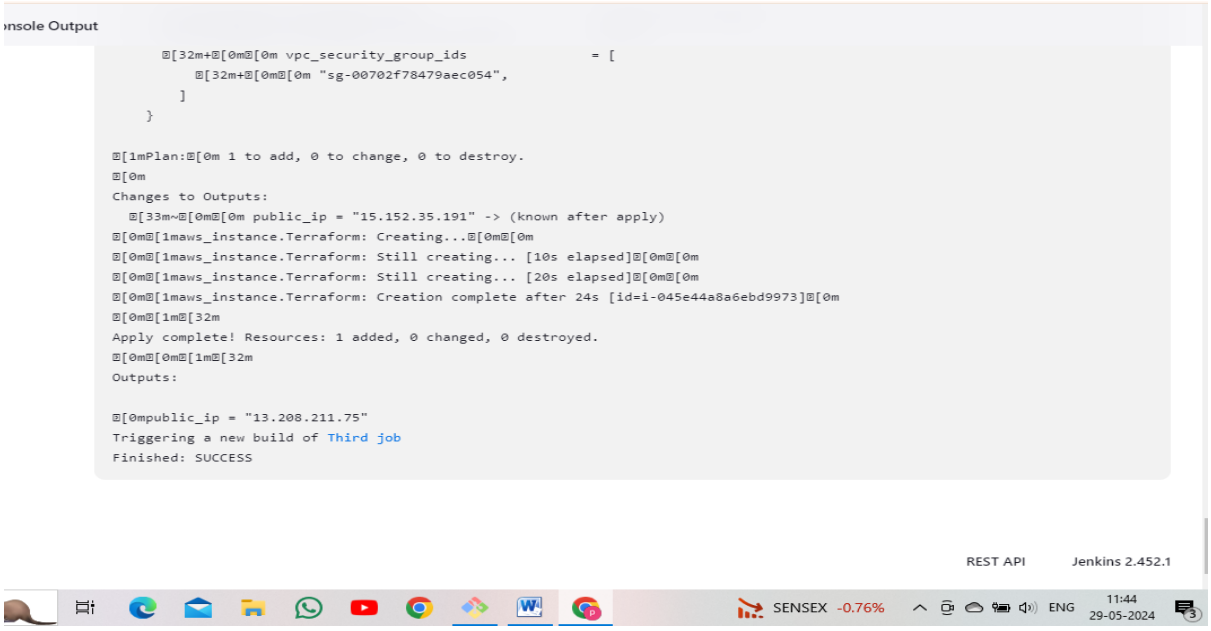
- After that click apply and save and Build now.
- The first job console output as



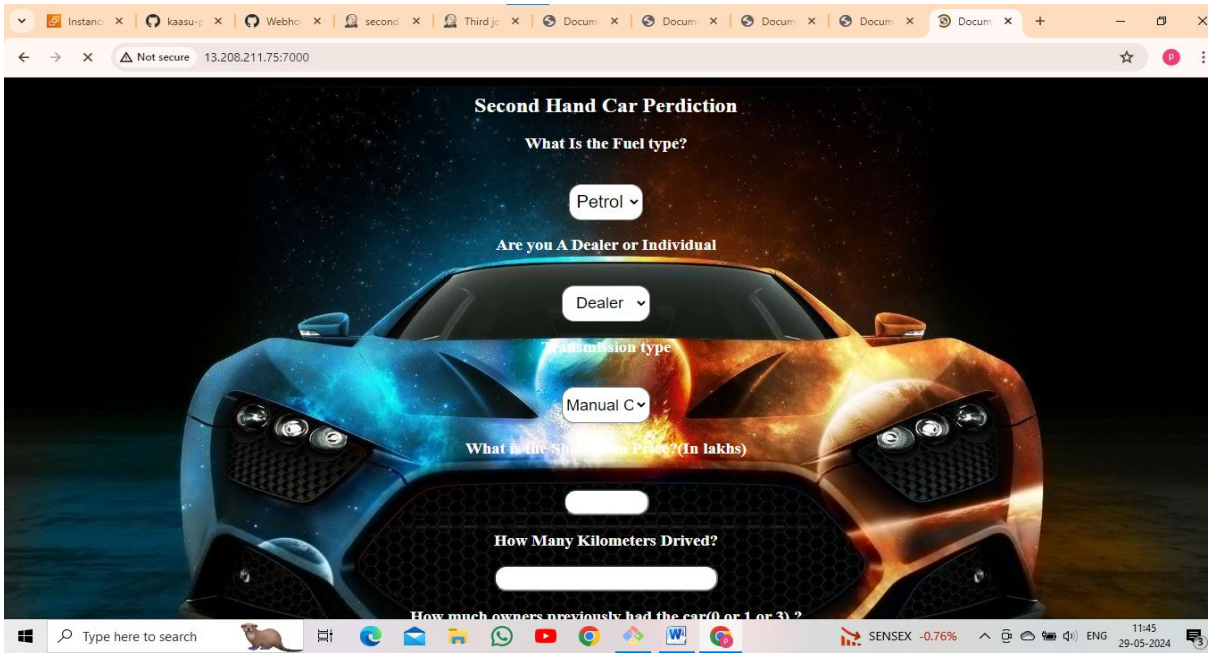
- Console output public copy ip and paste it Google.



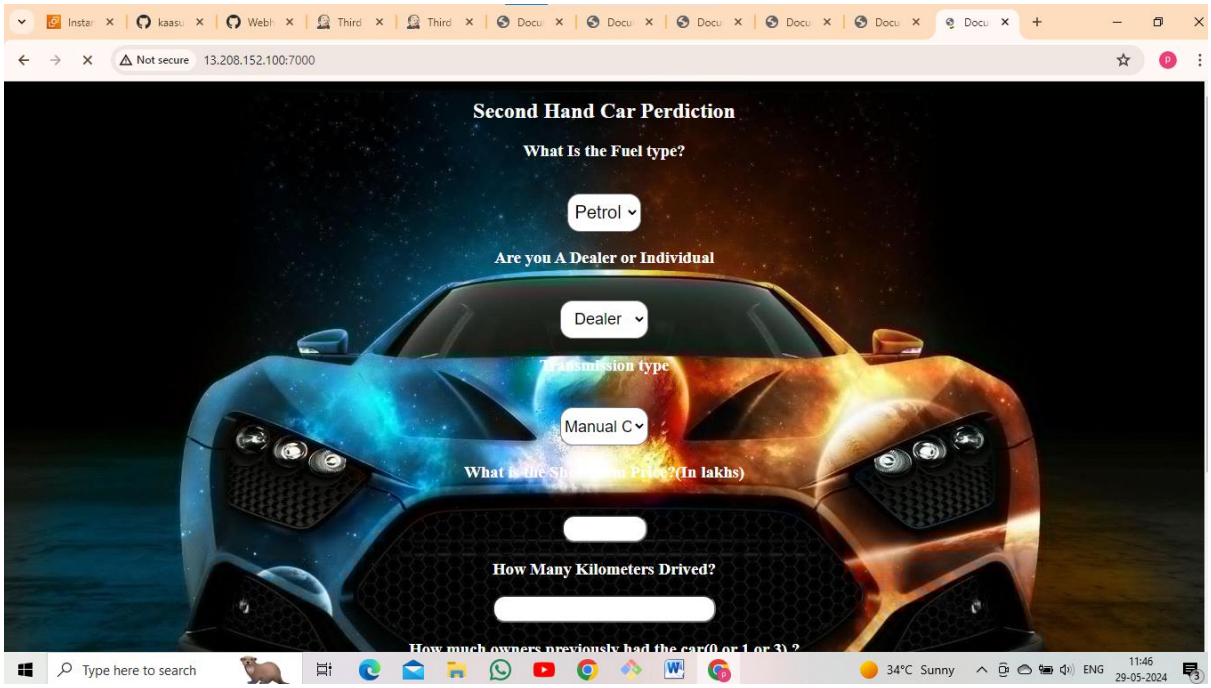
- After that go to Jenkins dashboard create new item.
- Select Build after other projects are built.
- Next step is apply and save and Build now.



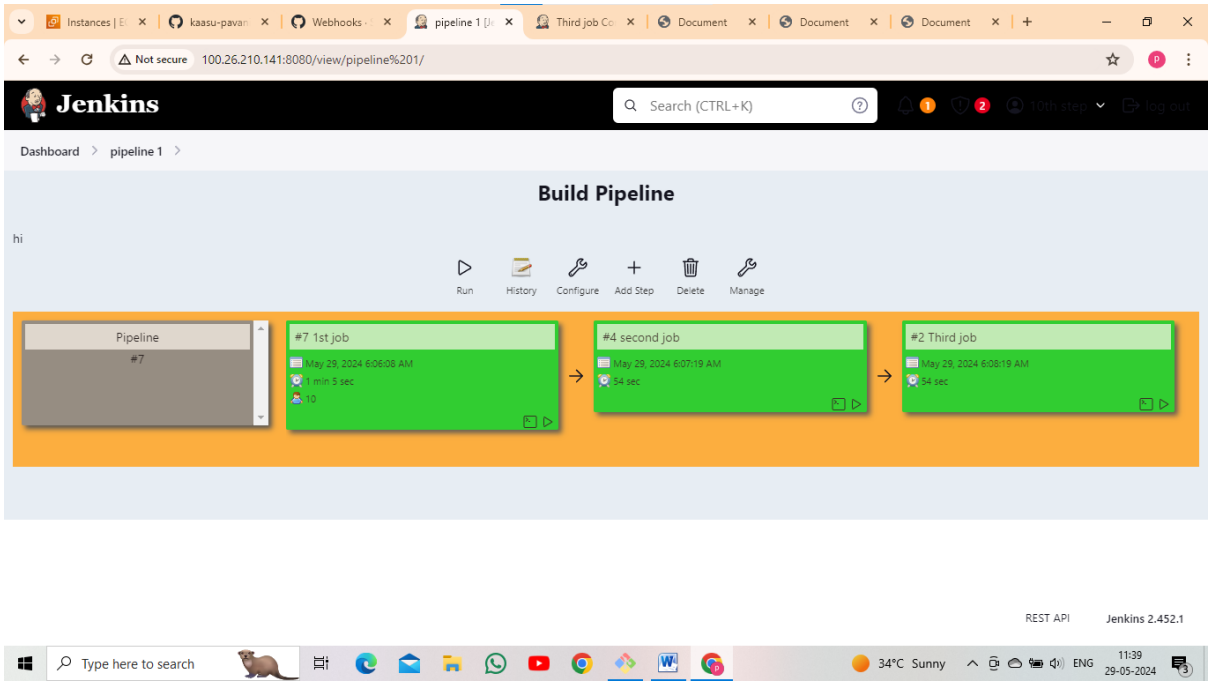
- Console output public ip and paste it Google.



- 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**********