Python application deployment using multiple methods

Presented By

Amgoth Pravalika

Workflow Overview

1.Manual Build and Deploy–"Manual Deploy"

2.Build with EC2 and Userdata–"EC2 Deploy"

3.Build with Bash Script–"Bash Automate"

4.Git, GitHub, Jenkins with Shell–"Git-Jenkins Deploy"

5.Terraform with GitHub Integration–"Terraform Git Deploy"

6.Git, GitHub, Jenkins, and Terraform–"CI/CD with Terraform"

7.Docker Build and Hub Push–"Docker Build & Push"

8.Docker with Kubernetes –"Docker-K8s Deploy"

9.Git, GitHub, Jenkins, Terraform-bp,pollscm,webhooks

10.Pipeline with Periodic Builds–"Pipeline Automation"

Project Objective

- Automate the Build and Deployment Process
- Establish Continuous Integration and Continuous Deployment (CI/CD) Pipelines
- Infrastructure as Code (IaC) with Terraform
- Version Control with Git and GitHub
- Container Orchestration with Kubernetes
- Containerization of the Python Application with Docker

1. Build and deploy python applications in manual process

1.Set up a Virtual Private Network (VPN):

2.Install Python and Required Tools: sudo yum install python3-pip

3.Clone the Application Repository:

4.Install Dependencies: pip3 install -r requirements.txt

5.Run the Application: python app.py

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

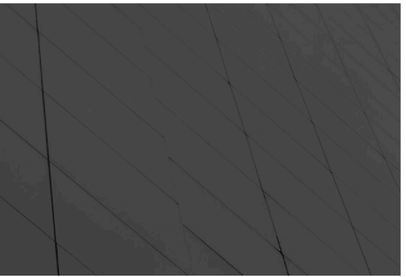
1.Create a Virtual Private Cloud (VPC):

2.Launch an EC2 Instance:

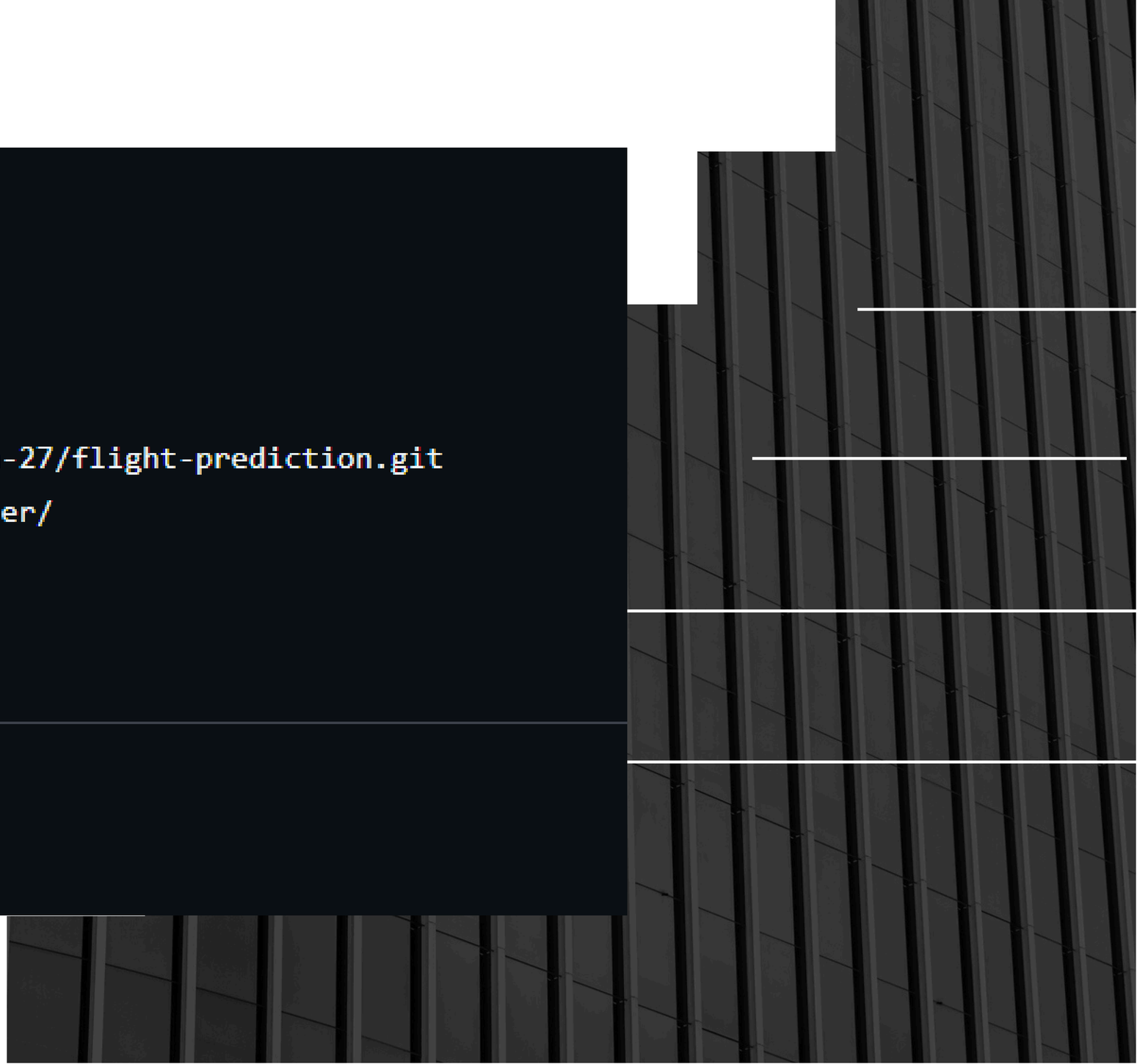
3.Write a User Data Script:

4.Configure the User Data Script:

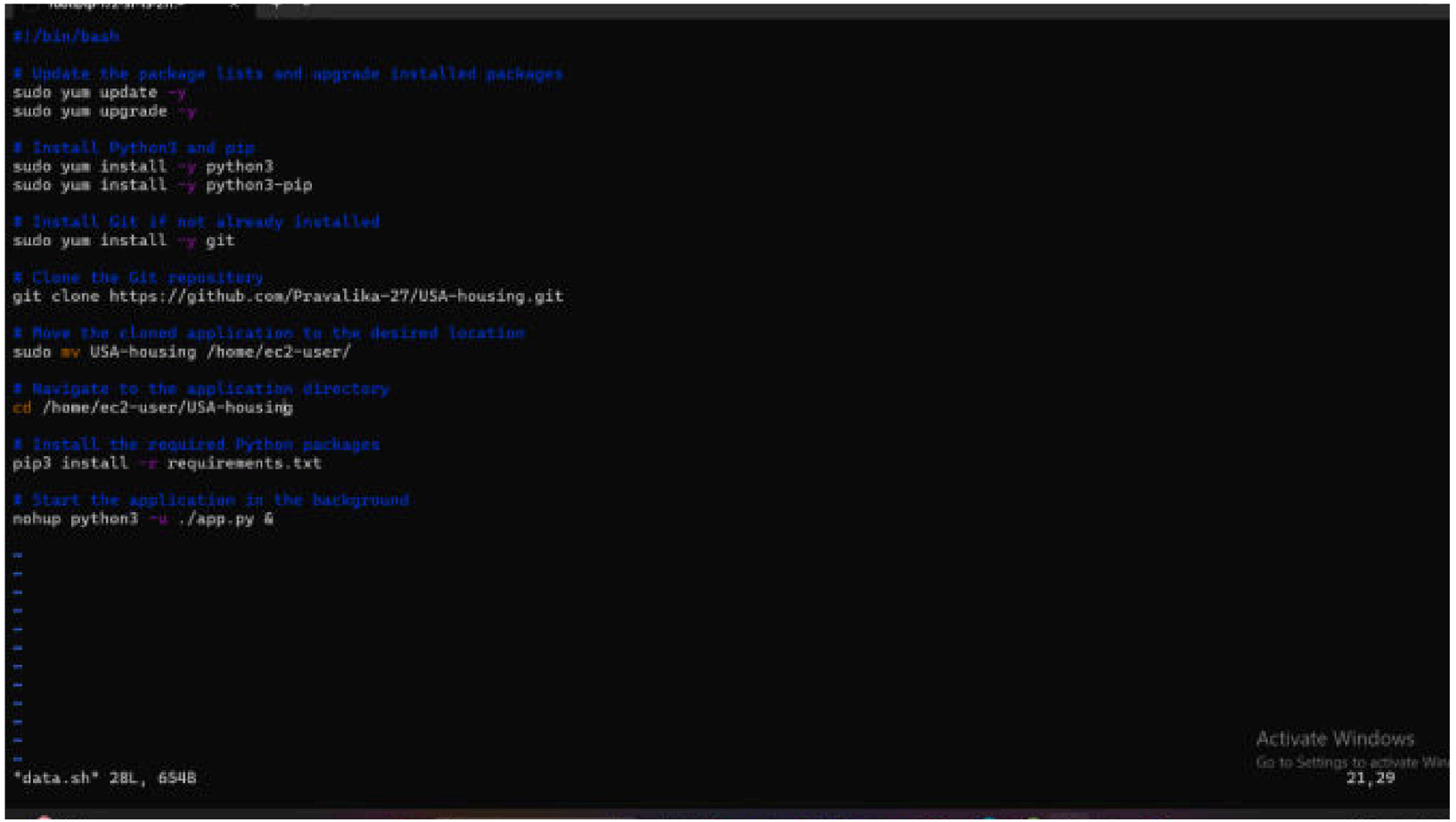
5.access the application using instance id and port



```
sudo yum update -y
sudo yum upgrade -y
sudo yum install -y python3
sudo yum install -y python3-pip
sudo yum install -y git
git clone https://github.com/Pravalika-27/flight-prediction.git
sudo mv flight-prediction /home/ec2-user/
cd /home/ec2-user/flight-prediction
pip3 install -r requirements.txt
nohup python3 -u ./app.py &
```



3. Build and deploy python applications in Bash scripting



```
#!/bin/bash

# Update the package lists and upgrade installed packages
sudo yum update -y
sudo yum upgrade -y

# Install Python3 and pip
sudo yum install -y python3
sudo yum install -y python3-pip

# Install Git if not already installed
sudo yum install -y git

# Clone the Git repository
git clone https://github.com/Pravalika-27/USA-housing.git

# Move the cloned application to the desired location
sudo mv USA-housing /home/ec2-user/

# Navigate to the application directory
cd /home/ec2-user/USA-housing

# Install the required Python packages
pip3 install -r requirements.txt

# Start the application in the background
nohup python3 -u ./app.py &
```

"data.sh" 28L, 654B

21,29

3. Build and deploy python applications in Bash scripting

a.Write userdata in data.sh file

b.run that userdata by using -sh data.sh

1.Update System: The script starts by updating the system to ensure it’s up-to-date.

2.Install Python: It installs Python 3 using the command.

3.Install Dependencies: The script installs any libraries or dependencies your Python app needs by reading thefile with.

4.Run the Application: It then starts the Python application using

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

1..Launch an EC2 Instance:

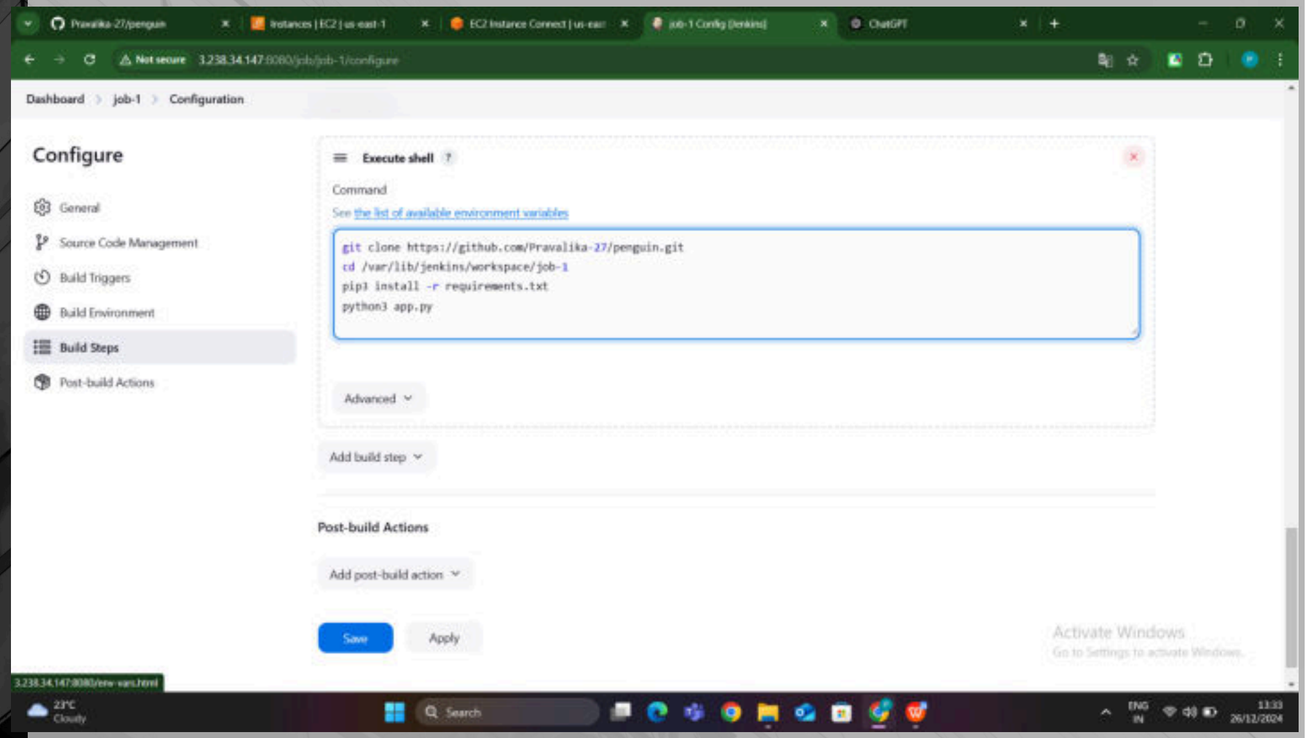
2.Setup Jenkins Environment:

3.Create Jenkins Job:

4..Take GitHub Code into Jenkins:

5.Run Commands in Execute Shell:

6.Build the Job:



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

1.Create an EC2 Instance and Install Jenkins Manually:

2.Create Terraform Files to Automate EC2 Instance, VPC, Route Table, and IGW Creation:

3.Initialize and Apply Terraform:

ubuntu@ip-172-31-29-81: ~

```
resource "aws_vpc" "demovpc" {
cidr_block="10.0.0.0/16"
instance_tenancy="default"
tags={
Name="DEMO VPC"
}
}
resource "aws_subnet" "public_subnet-1" {
vpc_id="${aws_vpc.demovpc.id}"
cidr_block="10.0.1.0/24"
map_public_ip_on_launch=true
availability_zone="us-east-1a"
tags={
Name="Web Subnet 1"
}
}
~
```

```
sudo yum update -y
sudo yum upgrade -y
sudo yum install -y python3
sudo yum install -y python3-pip
sudo yum install -y git
git clone https://github.com/Pravalika-27/flight-prediction.git
sudo mv flight-prediction /home/ec2-user/
cd /home/ec2-user/flight-prediction
pip3 install -r requirements.txt
nohup python3 -u ./app.py &
```

ubuntu@ip-172-31-19-232: ~

```
resource "aws_internet_gateway" "demogateway" {
vpc_id="${aws_vpc.demovpc.id}"
tags={
Name="Internet Gateway"
}
}
~
~
```

Pravalika-27 Update ec2.tf

6bbb736 · 3 days ago History

Code Blame 27 lines (26 loc) · 651 Bytes Code 55% faster with GitHub Copilot

```
resource "aws_instance" "public_subnet-1" {
ami="ami-0ca9fb66e076a6e32"
instance_type="t2.micro"
count=1
key_name="pinku"
vpc_security_group_ids=[aws_security_group.demosg.id]
subnet_id="${aws_subnet.public_subnet-1.id}"
associate_public_ip_address=true
user_data="${file("data.sh")}"
tags={
Name="My public Instance 1 "
}
}

resource "aws_instance" "public_subnet-2" {
ami="ami-012967cc5a8c9f891"
instance_type="t2.micro"
count=1
key_name="pinku"
vpc_security_group_ids=[aws_security_group.demosg.id]
subnet_id="${aws_subnet.public_subnet-2.id}"
associate_public_ip_address=true
user_data="${file("data.sh")}"
tags={
Name="My public Instance 2 "
}
}
```

```
resource "aws_route_table" "route" {
vpc_id="${aws_vpc.demovpc.id}"
route{
cidr_block="0.0.0.0/0"
gateway_id="${aws_internet_gateway.demogateway.id}"
}
tags={
Name="Route to Internet"
}
}

resource "aws_route_table_association" "rt1" {
subnet_id="${aws_subnet.public_subnet-1.id}"
route_table_id="${aws_route_table.route.id}"
}
```

```
resource "aws_internet_gateway" "demogateway" {
vpc_id="${aws_vpc.demovpc.id}"
tags={
Name="Internet Gateway"
}
}
```

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

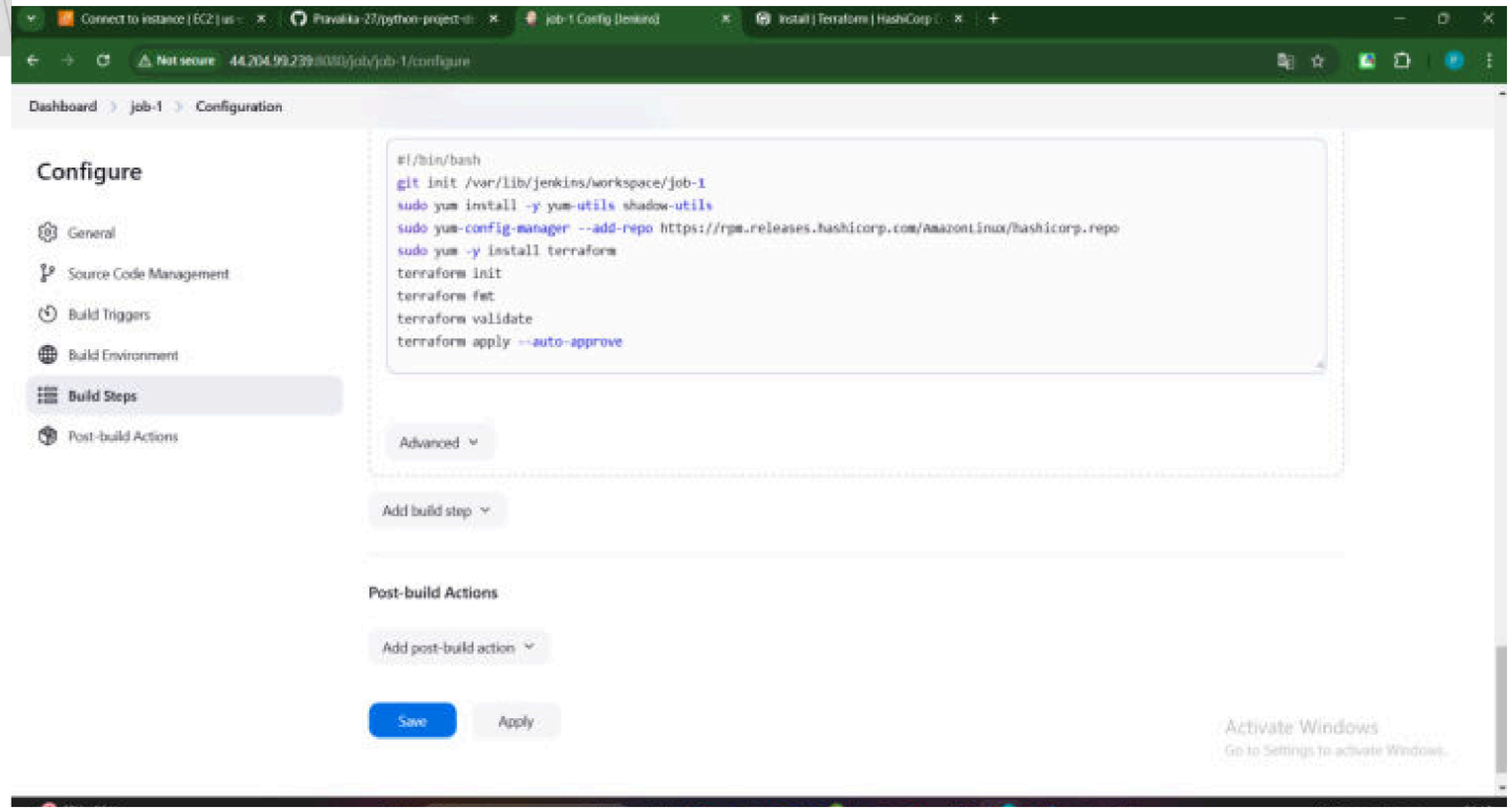
1.Install Jenkins on EC2

2.Create a Jenkins Job

3.Install AWS Credentials Plugin

4.Add Installation Commands to Jenkins Job

5.Run the Job


Connect to instance | EC2 | us
Pravalika-27/python-project-cl
job-1 Config [Jenkins]
Install | Terraform | HashiCorp
Not secure 44.204.99.239:8080/job/job-1/configure
Dashboard
job-1
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
#!/bin/bash
git init /var/lib/jenkins/workspace/job-1
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 fmt
terraform validate
terraform apply --auto-approve
Advanced
Add build step
Post-build Actions
Add post-build action
Save
Apply
Activate Windows
Go to Settings to activate Windows.

7. Build and deploy python applications with Docker

1.Launch an EC2 Instance

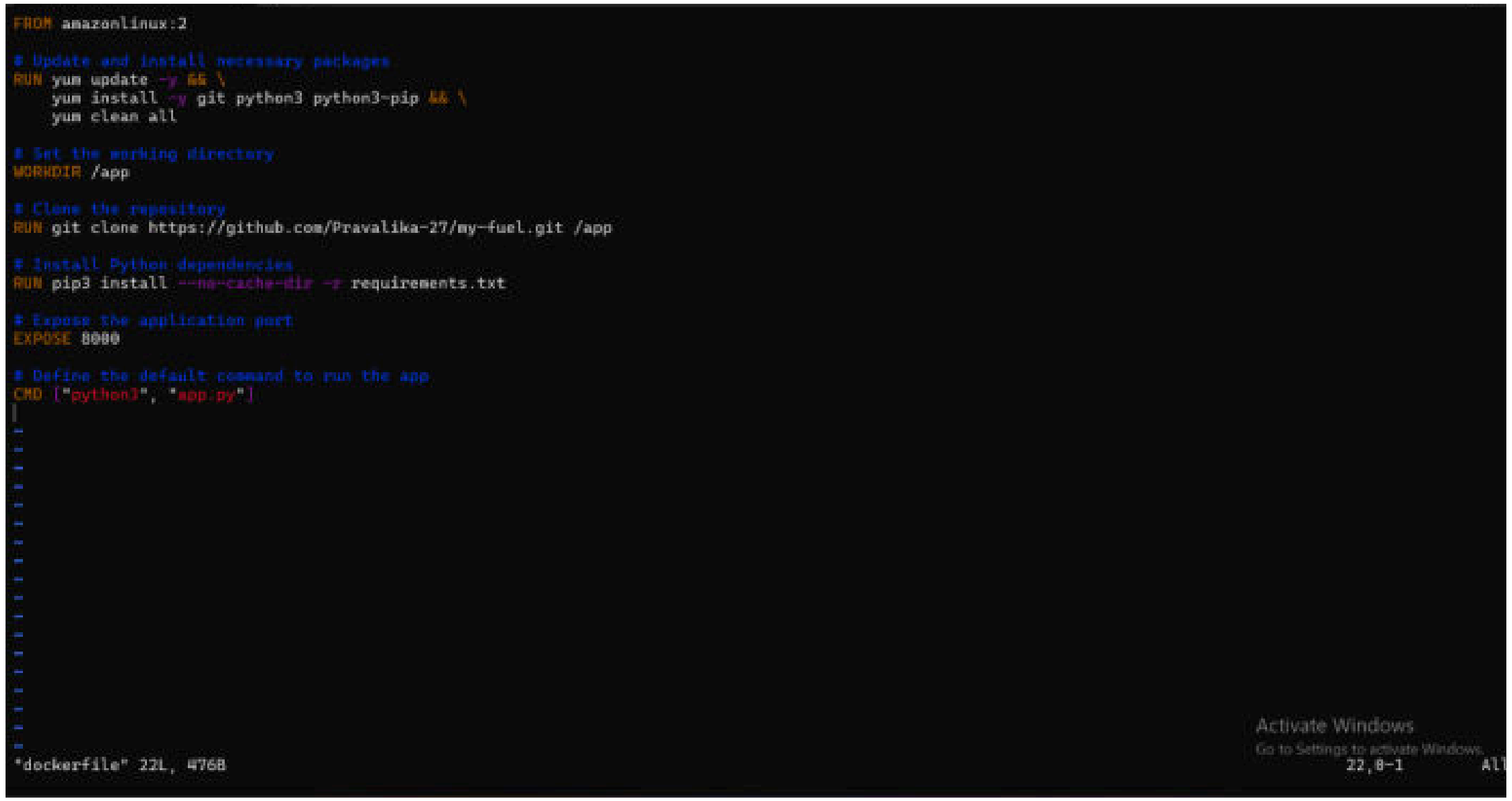
2.Install Docker on EC2

3.Create a Dockerfile

4.Build the Docker Image : docker build -t my-python-app .

5.Run the Docker Container : docker run -p 8000:8000 my-python-app

6.Push Docker Image to Docker Hub


FROM amazonlinux:2
Update and install necessary packages
RUN yum update -y && \
yum install -y git python3 python3-pip && \
yum clean all
Set the working directory
WORKDIR /app
Clone the repository
RUN git clone https://github.com/Pravalika-27/my-fuel.git /app
Install Python dependencies
RUN pip3 install --no-cache-dir -r requirements.txt
Expose the application port
EXPOSE 8080
Define the default command to run the app
CMD ["python3", "app.py"]
"dockerfile" 22L, 476B
22,8-1
Activate Windows
Go to Settings to activate Windows.

8. Build and deploy python applications with Docker and k8's

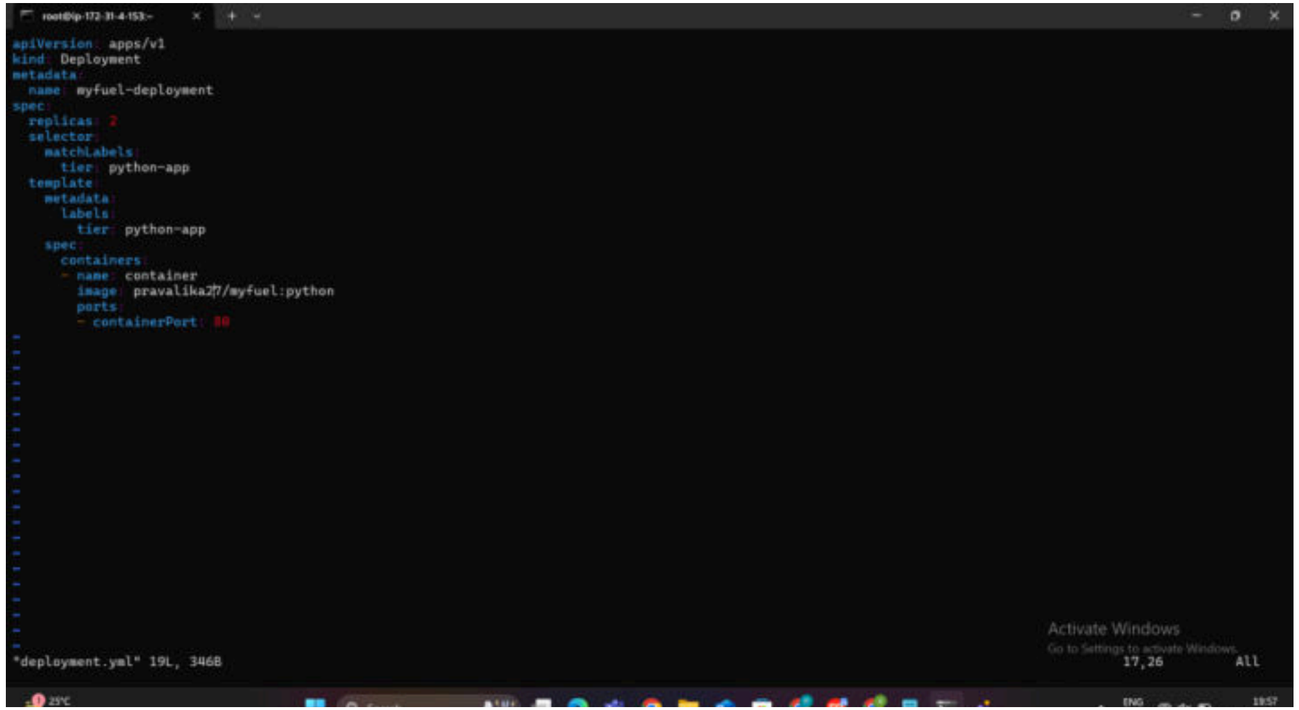
1.Set up Kops on EC2 Instance

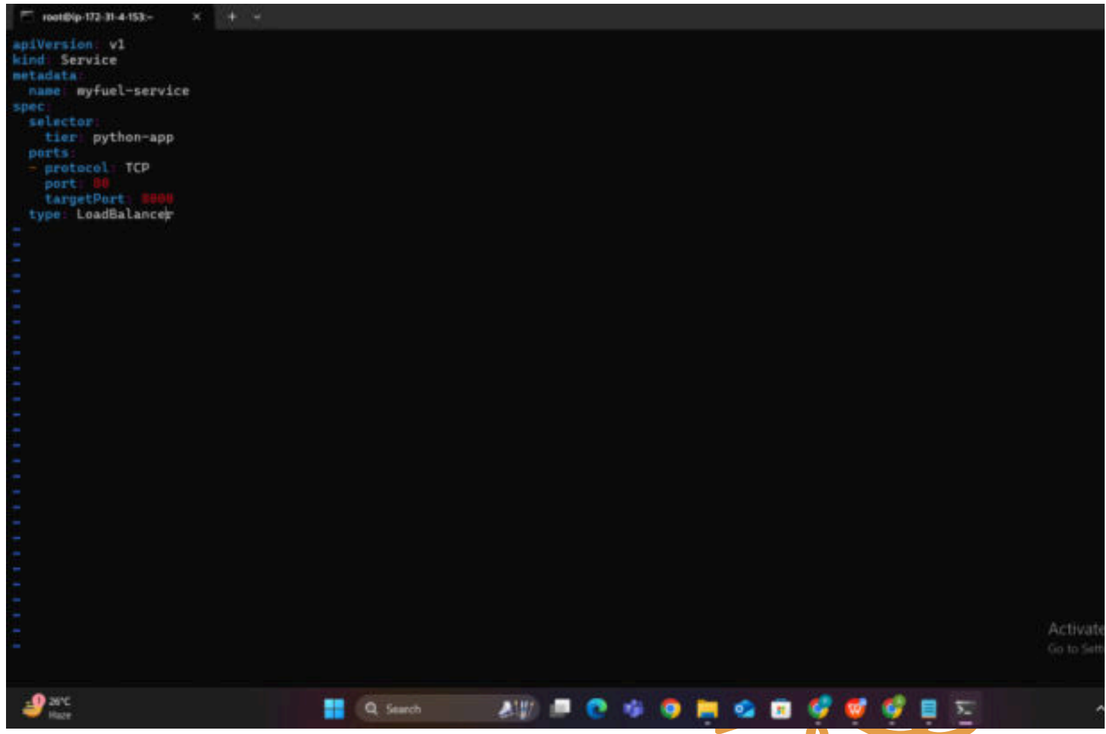
2.Set up the Kops Cluster

3.Write the Deployment and Service Files

4.Deploy the Application on Kubernetes

5.Access the Application


root@ip-172-31-4-153:~
apiVersion: apps/v1
kind: Deployment
metadata:
name: myfuel-deployment
spec:
replicas: 2
selector:
matchLabels:
tier: python-app
template:
metadata:
labels:
tier: python-app
spec:
containers:
- name: container
image: pravalika27/myfuel:python
ports:
- containerPort: 80
"deployment.yml" 19L, 346B
17,26
All
Activate Windows
Go to Settings to activate Windows.


root@ip-172-31-4-153:~
apiVersion: v1
kind: Service
metadata:
name: myfuel-service
spec:
selector:
tier: python-app
ports:
- protocol: TCP
port: 80
targetPort: 8000
type: LoadBalancer
Search

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

1.Set Up Jenkins

2.Create a Jenkins Job

3.Configure Build Triggers

4.Add Build Steps

5.Save the Job and Run Builds

6.Check Builds

a.Write execute shell and select build periodically triggering method.

b.PollSCM: Trigger builds based on code changes.

c.Webhooks: Real-time triggers from GitHub.

d.Enable webhooks in github repository.

Build and deploy python applications with pipeline method with Build periodically, pollscm and webhooks

1.Set Up Jenkins Pipeline Job

2.Create a pipeline with three stages “code”,”dependency”,”deploy”.

3.Select triggering methods by using syntax and enabling webhooks in github and check with each method.

Dashboard > 2 > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Definition

Pipeline script

Script ?

try sample Pipeline...

```
pipeline {
    agent any

    stages {
        stage('code') {
            steps {
                git 'https://github.com/Pravalika-27/india-liver-patient.git'
            }
        }
        stage('dependency') {
            steps {
                sh '''
                sudo yum install python3 python3-pip -y
                pip3 install -r requirements.txt
                '''
            }
        }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply


THANK
YOU!