**PYTHON DEVOPS PROJECTS WITH SOLUTIONS**

**1️. Automating Server Provisioning with Ansible & Python**

**Steps:**

1. **Install Ansible** (pip install ansible).
2. **Write an Ansible playbook** to configure servers.
3. **Use Python to trigger the Ansible playbook**.

**Ansible Playbook (server-setup.yml)**

yaml

CopyEdit

- name: Setup Web Server

hosts: all

become: true

tasks:

- name: Install Nginx

apt:

name: nginx

state: present

- name: Start Nginx

service:

name: nginx

state: started

**Python Script to Execute Ansible Playbook (run\_ansible.py)**

python

CopyEdit

import os

def run\_ansible():

os.system("ansible-playbook -i inventory server-setup.yml")

run\_ansible()

**Solution:**

* Automates server provisioning.
* Ensures Nginx is installed and running.

**2️. CI/CD Pipeline with Jenkins & Python**

**Steps:**

1. **Use Python to trigger Jenkins jobs** via API.
2. **Monitor build status** and send alerts.

**Python Script to Trigger Jenkins Job (trigger\_jenkins.py)**

python

CopyEdit

import requests

JENKINS\_URL = "http://your-jenkins-server/job/myjob/build"

JENKINS\_USER = "admin"

JENKINS\_TOKEN = "your-api-token"

response = requests.post(JENKINS\_URL, auth=(JENKINS\_USER, JENKINS\_TOKEN))

if response.status\_code == 201:

print("Jenkins job triggered successfully!")

else:

print("Failed to trigger job.")

**Solution:**

* Automates Jenkins job execution from Python.

**3️. Docker Image Builder & Pusher**

**Steps:**

1. **Build a Docker image from a Python script**.
2. **Push the image to Docker Hub**.

**Python Script (docker\_build\_push.py)**

python

CopyEdit

import os

IMAGE\_NAME = "yourdockerhub/python-app:latest"

# Build Docker image

os.system(f"docker build -t {IMAGE\_NAME} .")

# Login to Docker Hub

os.system("docker login -u yourusername -p yourpassword")

# Push the image

os.system(f"docker push {IMAGE\_NAME}")

print("Docker image pushed successfully!")

**Solution:**

* Automates container image creation and deployment.

**4️. Log Monitoring with Python & ELK Stack**

**Steps:**

1. **Use Python to read logs**.
2. **Send logs to Elasticsearch**.
3. **Visualize logs in Kibana**.

**Python Script (log\_monitor.py)**

python

CopyEdit

import json

import requests

LOG\_FILE = "/var/log/syslog"

ELK\_URL = "http://your-elasticsearch-server:9200/logs/\_doc/"

def send\_log\_to\_elk(log\_data):

headers = {"Content-Type": "application/json"}

response = requests.post(ELK\_URL, data=json.dumps(log\_data), headers=headers)

return response.status\_code

with open(LOG\_FILE) as file:

for line in file:

log\_entry = {"message": line.strip()}

send\_log\_to\_elk(log\_entry)

print("Logs sent to ELK stack!")

**Solution:**

* Automates log collection for real-time monitoring.

**5️. Kubernetes Deployment Manager**

**Steps:**

1. **Use Python to deploy applications to Kubernetes**.
2. **Monitor pod status**.

**Python Script (k8s\_deploy.py)**

python

CopyEdit

from kubernetes import client, config

config.load\_kube\_config()

v1 = client.CoreV1Api()

pod = client.V1Pod(

metadata=client.V1ObjectMeta(name="my-python-app"),

spec=client.V1PodSpec(

containers=[

client.V1Container(

name="my-python-container",

image="yourdockerhub/python-app:latest"

)

]

)

)

v1.create\_namespaced\_pod(namespace="default", body=pod)

print("Pod deployed successfully!")

**Solution:**

* Automates Kubernetes pod deployment.