CI/CD Pipeline for Python Flask Application Documentation

# Overview

This documentation provides a step-by-step guide to building a **CI/CD pipeline** for a **Python Flask web application** using **Docker**, **Jenkins**, and **Ansible**. The pipeline automates the deployment of a Flask application from Docker Hub to an EC2 instance.

# Technologies Used

* **Python 3.9**: For the Flask application.
* **Flask**: Python web framework used to create the web application.
* **Docker**: Containerization tool for packaging the application.
* **Jenkins**: For continuous integration and automation.
* **Ansible**: Configuration management tool for deployment.
* **AWS EC2**: Cloud instance to host the application.

# Project Structure

1. **Dockerfile**: Defines the container setup for the Python Flask application.
2. **app.py**: A simple Flask application to demonstrate the functionality.
3. **deploy\_from\_docker\_hub.yml**: Ansible playbook to deploy the application from Docker Hub to EC2.
4. **hosts**: Ansible inventory file defining the EC2 instance configuration.
5. **requirements.txt**: Python dependencies required for the Flask application.
6. **SSH Key**: The SSH key (jenkins-key.pem) for accessing the EC2 instance.

## Step 1: Dockerfile

The Dockerfile sets up a lightweight environment for running the Flask application.

**Dockerfile**:

Dockerfile

Copy code

# Use an official Python runtime as a parent image

FROM python:3.9-slim

# Set the working directory in the container

WORKDIR /app

# Copy the current directory contents into the container at /app

COPY . /app

# Install the required packages

RUN pip install --no-cache-dir -r requirements.txt

RUN pip install --upgrade Flask

# Debugging echo to ensure Docker works as expected

RUN echo "Hello Jenkins"

# Make port 5000 available to the world outside this container

EXPOSE 5000

# Define the command to run the app

CMD ["python", "app.py"]

**Step 2: Flask Application (app.py)**

This is a simple Flask web app that will return "Hello, World!" when accessed.

**app.py**:

python

Copy code

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, World!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

**Step 3: Ansible Playbook (deploy\_from\_docker\_hub.yml)**

The deploy\_from\_docker\_hub.yml Ansible playbook automates the deployment of the Flask application from Docker Hub to the EC2 instance.

**Ansible Playbook**:

yaml

Copy code

---

- name: Deploy Application from Docker Hub to EC2

hosts: EC2

become: yes

vars:

docker\_image: "wasee2mounir/simple-flask-app-img:v1"

container\_name: "flask-app-container"

app\_port: 5000

tasks:

- name: Ensure Docker is installed

apt:

name: docker.io

state: present

update\_cache: yes

- name: Start Docker service

service:

name: docker

state: started

enabled: yes

- name: Pull the Docker image

docker\_image:

name: "{{ docker\_image }}"

source: pull

- name: Run the Docker container

docker\_container:

name: "{{ container\_name }}"

image: "{{ docker\_image }}"

state: started

ports:

- "{{ app\_port }}:{{ app\_port }}"

**Step 4: Ansible Inventory (hosts)**

The hosts file defines the EC2 instance information, including the IP address and SSH key for access.

**hosts**:

ini

Copy code

[EC2]

18.184.6.122 host\_key\_checking=False ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=simple-flask-app/jenkins-key.pem

**Step 5: Python Dependencies (requirements.txt)**

The requirements.txt file lists the Python dependencies for the Flask application.

**requirements.txt**:

makefile

Copy code

Flask==2.0.1

**Step 6: SSH Key**

The SSH key jenkins-key.pem is used to access the EC2 instance via Ansible for deployment.

**CI/CD Pipeline Workflow**

1. **Build and Test**:
   * Jenkins clones the repository from GitHub.
   * Docker builds the image using the Dockerfile.
   * The Flask application is tested locally using Docker.
2. **Push to Docker Hub**:
   * After successful testing, the Docker image is tagged and pushed to Docker Hub for deployment.
3. **Deployment via Ansible**:
   * Ansible connects to the EC2 instance using the hosts file and the SSH key (jenkins-key.pem).
   * Docker is installed and started on the EC2 instance.
   * Ansible pulls the Flask app Docker image from Docker Hub.
   * The Flask app container is run on the EC2 instance, exposing it on port 5000.

**Conclusion**

This pipeline ensures automated building, testing, and deployment of the Python Flask application using Jenkins, Docker, and Ansible. It also enables easy deployment to an EC2 instance with minimal manual intervention.

This document provides all the configuration files and steps required to successfully run the project in a DevOps environment.