**Jenkins CI/CD Pipeline for Python & Docker Deployment**

**1. Project Overview**

This project automates the build, test, and deployment process of a Python application using **Jenkins CI/CD** and **Docker**. The pipeline performs:

* **Cloning the repository** from GitHub.
* **Setting up a Python virtual environment**.
* **Building a Python package** using build.
* **Running unit tests** with pytest.
* **Building a Docker image** from the Python package.
* **Deploying the application** inside a container.

**2. Project Structure**

Below is the directory structure of the project:

my\_python\_project/

│-- Dockerfile

│-- Jenkinsfile

│-- pyproject.toml

│-- .gitignore

│-- dist/

│ └── my\_python\_project-0.1.0-py2.py3-none-any.whl

│-- src/

│ └── my\_python\_project/

│ └── main.py

│-- tests/

│ └── test\_main.py

* **src/my\_python\_project/main.py** → Main Python application.
* **tests/test\_main.py** → Unit tests.
* **Dockerfile** → Defines the Docker image for deployment.
* **Jenkinsfile** → Contains the Jenkins pipeline configuration.
* **pyproject.toml** → Defines the Python project metadata.
* **dist/** → Stores the built Python package (wheel file).

**3. File Contents**

**3.1 src/my\_python\_project/main.py**

This is the main Python script that gets executed when the application runs.

import time

def main():

print("Hello from Jenkins CI/CD Pipeline!")

while True:

time.sleep(1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3.2 tests/test\_main.py**

This is the unit test file that validates the output of main.py.

import unittest

from my\_python\_project.main import main

import io

import sys

class TestMain(unittest.TestCase):

def test\_main(self):

captured\_output = io.StringIO()

sys.stdout = captured\_output

main()

sys.stdout = sys.\_\_stdout\_\_

self.assertEqual(captured\_output.getvalue().strip(), "Hello from Jenkins CI/CD Pipeline!")

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**3.3 Dockerfile**

This file defines the Docker image and how it should be built.

FROM python:3.9-slim

WORKDIR /app

COPY dist/\*.whl ./

RUN pip install \*.whl

CMD ["python", "-m", "my\_python\_project.main"]

**3.4 Jenkinsfile**

This file defines the entire Jenkins pipeline.

pipeline {

agent any

environment {

REPO\_URL = "git@github.com:lazor/my\_python\_project.git"

WORKDIR = "${WORKSPACE}/my\_python\_project"

VENV\_PATH = "${WORKSPACE}/venv"

DOCKER\_IMAGE = "my-python-app:latest"

}

stages {

stage('Clone Repository') {

steps {

cleanWs()

sh '''

echo "Current directory: $PWD"

if [ -d "$WORKDIR" ]; then

echo "Directory already exists, removing it"

rm -rf "$WORKDIR"

fi

git clone $REPO\_URL "$WORKDIR"

echo "Repository cloned successfully!"

ls -la "$WORKDIR"

'''

}

}

stage('Setup Python Environment') {

steps {

sh '''

python3 -m venv "$VENV\_PATH"

. "$VENV\_PATH/bin/activate"

pip install --upgrade pip build pytest

'''

}

}

stage('Build Wheel') {

steps {

dir('my\_python\_project') {

sh '''

. "$VENV\_PATH/bin/activate"

python3 -m build --wheel

'''

}

}

}

stage('Test') {

steps {

dir('my\_python\_project') {

sh '''

. "$VENV\_PATH/bin/activate"

export PYTHONPATH=$PYTHONPATH:$(pwd)/src # Add src/ to PYTHONPATH

pytest tests/

'''

}

}

}

stage('Build Docker Image') {

steps {

dir('my\_python\_project') {

sh '''

docker build -t $DOCKER\_IMAGE .

'''

}

}

}

stage('Deploy') {

steps {

sh '''

docker stop my-python-container || true

docker rm my-python-container || true

docker run -d --name my-python-container $DOCKER\_IMAGE

'''

}

}

}

post {

success {

echo '✅ Repository cloned, built, and deployed successfully!'

}

failure {

echo '❌ Pipeline failed. Check the logs for details.'

}

}

}

**3.5 pyproject.toml**

Defines the Python project configuration.

[build-system]

requires = ["hatchling"]

build-backend = "hatchling.build"

[project]

name = "my\_python\_project"

version = "0.1.0"

dependencies = []

[project.scripts]

myapp = "my\_python\_project.main:main"

[tool.hatch.build.targets.wheel]

packages = ["src/my\_python\_project"]

**3.6 .gitignore**

Defines ignored files to avoid committing unnecessary files.

\_\_pycache\_\_/

\*.py[cod]

\*.class

dist/

build/

\*.egg-info/

venv/

**4. Summary of What We Did Today**

1. **Configured Jenkins with SSH for GitHub Authentication**
   * Set up SSH keys for Jenkins.
   * Added public key to GitHub.
   * Switched repository access from HTTPS to SSH.
2. **Fixed Jenkins Pipeline Errors**
   * Set up a **virtual environment** in Jenkins.
   * Resolved **Python module import issues** by updating PYTHONPATH.
   * Ensured **tests ran correctly** using pytest.
3. **Built & Tested Python Package**
   * Used build to create a Python **wheel package**.
   * Ran **unit tests** to verify functionality.
4. **Built & Deployed Docker Container**
   * Created a Docker image with Dockerfile.
   * Used Jenkins to **deploy the container**.
5. **Verified Deployment in Docker**
   * Checked running containers using:
   * docker ps -a
   * Verified logs with:
   * docker logs my-python-container

**5. Outcomes**

* **Jenkins Pipeline completed successfully** ✅
* **Python app deployed inside a Docker container** ✅
* **Automated build, test, and deployment process implemented** ✅