

Name: Barathkumar J K

Jenkins Assignment

1) Build and Deploy a Simple App using Jenkins

The screenshot displays the Jenkins web interface for a pipeline named 'MyApp-CI-CD'. The 'Stage View' shows a grid of stages and their execution results. The stages are Declarative: Checkout SCM, Clone Repository, Build, Test, and Deploy. The 'Build' stage is currently running, indicated by a green bar. The 'Test' and 'Deploy' stages are marked as failed (red bars). The 'Clone Repository' stage is also marked as failed (red bar). The 'Declarative: Checkout SCM' stage is marked as successful (green bar).

Below the stage view, the 'Permalinks' section lists the last build (#6), last stable build (#6), last successful build (#6), and last failed build (#5).

The bottom part of the image shows the GitHub repository 'jbarathkumar/jenkins' with the 'Jenkinsfile' selected. The Jenkinsfile content is as follows:

```
1 pipeline {
2   agent any
3   stages {
4     stage('Clone Repository') {
5       steps {
6         git url:'https://github.com/jbarathkumar/jenkins.git',branch:'main'
7       }
8     }
9     stage('Build') {
10      steps {
11        sh 'echo "Building the application..."'
12      }
13    }
14    stage('Test') {
15      steps {
16        sh 'echo "Running tests..."'
17      }
18    }
19    stage('Deploy') {
20      steps {
21        sh 'echo "Deploying application..."'
22      }
23    }
24  }
25 }
```

2) Build and Deploy a Simple Python App using Jenkins & docker

Project Structure

```
my-flask-app
├── app.py
├── test.py
├── requirements.txt
├── Dockerfile
└── Jenkinsfile
```

1) Setup a Simple Flask App

app.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello():

    return 'Hello, World!'

if __name__ == '__main__':

    app.run(debug=True, host='0.0.0.0')
```

2) Create a Test File

test.py

```
import unittest

from app import app

class FlaskAppTestCase(unittest.TestCase):

    @classmethod

    def setUpClass(cls):

        cls.client = app.test_client()

        cls.client.testing = True

    def test_homepage(self):

        response = self.client.get('/')

        self.assertEqual(response.status_code, 200)
```

```
self.assertIn(b'Hello, World!', response.data)

if __name__ == '__main__':

    unittest.main()
```

3) Create a requirements.txt File

requirements.txt

Flask==2.2.2

Werkzeug==2.2.2

4) Create a Dockerfile

Dockerfile

FROM python:3.12-slim

WORKDIR /app

COPY . /app

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

EXPOSE 5000

ENV FLASK_APP=app.py

ENV FLASK_RUN_HOST=0.0.0.0

CMD ["flask", "run"]

5) Build & Push Docker Image

1. Build the Docker image:

```
docker build -t your-dockerhub-username/my-flask-app:latest .
```

2. Push the image to DockerHub:

```
docker login
```

```
docker push your-dockerhub-username/my-flask-app:latest
```

6) Connect Jenkins Pipeline to GitHub

1. Open Jenkins Dashboard.
2. Click on **New Item** → Select **Pipeline** → Name it → Click **OK**.

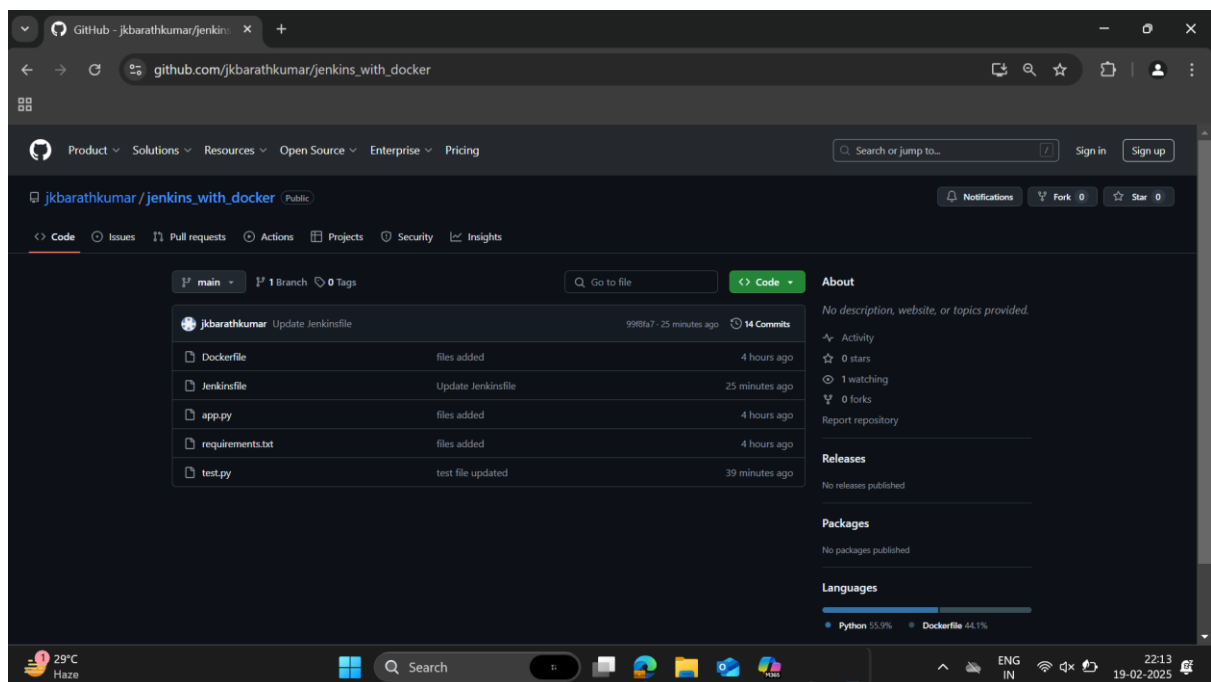
3. In the **Pipeline** section:
 - Select **Pipeline script from SCM**.
 - Choose **Git** as SCM.
 - Enter your **GitHub repository URL**.
 - Set **Branch: main**.
 - Define the **Jenkinsfile path**.
 4. Click **Save**.
-

7) Click "Build Now"

1. Navigate to **Jenkins Dashboard** → Your Pipeline.
2. Click **Build Now**.

Results:

GITHUB



Docker

Creating the docker image using dockerfile

```
PS C:\Users\Administrator\Downloads\ust\flask> docker build -t barathkumar29/my-flask-app .
[+] Building 9.4s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 649B
=> [internal] load metadata for docker.io/library/python:3.12-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.12-slim@sha256:34656cd90456349040784165b9decccbcee4de66f3ead0a1168ba893455afd1e
=> => resolve docker.io/library/python:3.12-slim@sha256:34656cd90456349040784165b9decccbcee4de66f3ead0a1168ba893455afd1e
=> [internal] load build context
=> => transferring context: 6.45kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:4468c0d223782a113fb4db114ef2ea36622f2b6385c05f00a79eb3f230790932
=> => exporting config sha256:ce4d8e2f5999c61edda7131fe53571e101c303cb0bd2bc75ec1a67a10af1d581
=> => exporting attestation manifest sha256:c1e99bc6530ef6571ba6410a1f88604f38b10bc160bdf557f2f72587b00680b0
=> => exporting manifest list sha256:78d59f801aef320c64a995e61f0336a3ad5f6c2f91c95d0b138d82a2c684ad90
=> => naming to docker.io/barathkumar29/my-flask-app:latest
=> => unpacking to docker.io/barathkumar29/my-flask-app:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/o793xuqr7v43cz2les8igncx
```

Push the docker image to dockerhub:

```
PS C:\Users\Administrator\Downloads\ust\flask> docker push barathkumar29/my-flask-app:latest
The push refers to repository [docker.io/barathkumar29/my-flask-app]
f7f520a378bd: Pushed
c29f5b76f736: Layer already exists
58c78b362c25: Layer already exists
6ae4ea23f4a: Layer already exists
c9727c30039b: Layer already exists
859caf3a263d: Layer already exists
1ef3011c664f: Pushed
9b97cb8d00d9: Pushed
```

The screenshot shows the Docker Hub web interface in a browser. The address bar displays 'hub.docker.com'. The page header includes the Docker Hub logo, navigation links (Explore, Repositories, Organizations, Usage), a search bar, and a user profile icon. The main content area shows a search for 'barathkumar29' with a dropdown menu. Below the search bar, a table lists the repository 'barathkumar29/my-flask-app', which was pushed 39 minutes ago, contains an image, is public, and has a scout status of 'Inactive'. On the right side, there are links to 'Create an organization' and 'Create and manage users and grant access to your organization'. The bottom of the page shows a taskbar with various application icons and a system clock indicating 22:13 on 19-02-2025.

Jenkins

flask-app #17 Console

localhost:8080/job/flask-app/17/console

Dashboard > flask-app > #17

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Tests on Docker Image)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ docker run --rm -w /app barathkumar29/my-flask-app:latest python -m unittest discover -s /app -p test.py
...
Ran 2 tests in 0.009s

OK
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] sh
+ docker logout
Removing login credentials for https://index.docker.io/v1/
[Pipeline] echo
Tests passed successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.492.1

flask-app [Jenkins]

localhost:8080/job/flask-app/

Jenkins

Dashboard > flask-app >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Favorite

Open Blue Ocean

Stages

Rename

Pipeline Syntax

Builds

Filter

Today

- #17 9:47 PM
- #16 9:39 PM
- #15 9:36 PM

flask-app

Add description

Stage View

Average stage times: (full run time: ~16s)

| Declarative: Checkout SCM | Checkout Code | Pull Latest Docker Image | Debug Docker Container | Run Tests on Docker Image | Declarative: Post Actions |
|---------------------------|---------------|--------------------------|------------------------|---------------------------|---------------------------|
| 1s | 877ms | 7s | 2s | 1s | 427ms |
| 1s | 877ms | 7s | 2s | 1s | 427ms |

Permalinks

- Last build (#17), 24 min ago
- Last stable build (#17), 24 min ago
- Last successful build (#17), 24 min ago
- Last failed build (#16), 32 min ago
- Last unsuccessful build (#16), 32 min ago
- Last completed build (#17), 24 min ago