

## Day-2

### Step-by-step guide to setting up a simple Python "Hello, Docker!" Flask application using Docker and Docker Compose.

---

#### 1. Install Docker

First, install Docker to get the Docker engine running on your system:

```
sudo apt install -y docker.io
```

- **Explanation:** Installs Docker on your system using the apt package manager. The -y flag auto-confirms any prompts.
- 

#### 2. Start and Enable Docker Service

Start the Docker service and enable it to start automatically at boot time:

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

- **Explanation:** The start command starts the Docker daemon, and enable ensures Docker runs on startup.
- 

#### 3. Verify Docker Installation

Verify that Docker was installed correctly by checking its version:

```
docker --version
```

- **Explanation:** Displays the installed Docker version to confirm the installation.
- 

#### 4. Install Docker Compose

Now, install Docker Compose, a tool to define and manage multi-container Docker applications:

```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

- **Explanation:** The first command downloads the latest Docker Compose binary, and the second command makes it executable.
- 

#### 5. Verify Docker Compose Installation

Check the installed version of Docker Compose:

`docker-compose --version`

- **Explanation:** Displays the installed Docker Compose version to verify the installation.
- 

## 6. Create Project Directory

Create a directory for your project and navigate into it:

```
mkdir ~/docker-python-app
```

```
cd ~/docker-python-app
```

- **Explanation:** Creates a directory for your project and navigates into it.
- 

## 7. Create the app.py file

Create a Python file app.py for the Flask application:

```
nano app.py
```

Paste the following Flask application code:

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def hello_world():
```

```
    return 'Hello, world Running inside the docker!'
```

```
if __name__ == '__main__':
```

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

- **Explanation:** A simple Flask app with one route (/) that returns a greeting message. The Flask server listens on all interfaces (0.0.0.0) and port 5000.
- 

## 8. Create requirements.txt

Create a requirements.txt file to list Python dependencies:

```
nano requirements.txt
```

Add the following content:

flask

- **Explanation:** Lists the Flask library as the required dependency for your project.
- 

## 9. Install pip (if not already installed)

Ensure pip is installed to handle Python package installations:

```
sudo apt update
```

```
sudo apt install python3-pip
```

- **Explanation:** Updates the package list and installs pip to handle Python packages.
- 

## 10. Create Dockerfile

Create a Dockerfile that defines how the Docker image should be built:

nano Dockerfile

Add the following content:

```
# Use the official Python image from Docker Hub
```

```
FROM python:3.9-slim
```

```
# Set the working directory inside the container
```

```
WORKDIR /app
```

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

```
COPY . /app
```

```
# Install any needed packages specified in requirements.txt
```

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

```
# Make port 5000 available to the world outside the container
```

```
EXPOSE 5000
```

```
# Define the environment variable for Flask to run in production mode
```

```
ENV FLASK_ENV=production
```

# Run app.py when the container launches

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

- **Explanation:** This Dockerfile defines the Python environment, installs dependencies, exposes port 5000, and starts the Flask app inside the container.
- 

## 11. Create docker-compose.yml

Create a docker-compose.yml file to manage the application's services:

nano docker-compose.yml

Add the following content:

version: '3.8'

services:

web:

build: .

ports:

- "5000:5000"

environment:

- FLASK\_ENV=development

volumes:

- ./app

restart: always

- **Explanation:** This Compose file:
    - Defines the web service.
    - Builds the image from the current directory.
    - Maps port 5000 from the host to the container.
    - Mounts the current directory (.) into the container to enable live code reloading.
    - Restarts the container if it crashes.
- 

## 12. Add User to Docker Group (if needed)

To avoid using sudo with Docker commands, add your user to the Docker group:

```
sudo usermod -aG docker $USER
```

```
newgrp docker
```

- **Explanation:** The first command adds your user to the Docker group, and the second command applies the changes to your current session.
- 

### 13. Build and Run the Application

Now, you can build and start the Flask app container using Docker Compose:

```
docker-compose up --build
```

- **Explanation:** This command builds the Docker image and starts the container based on the docker-compose.yml configuration. The --build flag forces a rebuild of the Docker image.
- 

### 14. Access the Application

Once the container is running, open your browser and navigate to:

```
http://localhost:5000
```

You should see the message: **"Hello, Docker Python App!"**

---

### Summary of Commands

1. Install Docker:
2. `sudo apt install -y docker.io`
3. Start and enable Docker service:
4. `sudo systemctl start docker`
5. `sudo systemctl enable docker`
6. Install Docker Compose:
7. `sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`
8. `sudo chmod +x /usr/local/bin/docker-compose`
9. Create project directory:
10. `mkdir ~/docker-python-app`
11. `cd ~/docker-python-app`
12. Create app.py with Flask code.
13. Create requirements.txt with flask.

14. Install pip (if needed):
15. `sudo apt update`
16. `sudo apt install python3-pip`
17. Create Dockerfile with the configuration.
18. Create `docker-compose.yml` with service definition.
19. Add your user to the Docker group (if necessary):
20. `sudo usermod -aG docker $USER`
21. `newgrp docker`
22. Build and run the app:
23. `docker-compose up --build`

Now your "Hello, Docker!" Flask app should be running inside a Docker container, accessible at <http://localhost:5000>.

```
pranitha@DESKTOP-SQFEICH:~$ sudo apt update
[sudo] password for pranitha:
Ign:1 https://pkg.jenkins.io/debian binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian binary/ Release
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Get:6 https://packages.adoptium.net/artifactory/deb noble InRelease [7511 B]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Ign:8 https://ppa.launchpadcontent.net/nginx/stable/ubuntu noble InRelease
Err:9 https://ppa.launchpadcontent.net/nginx/stable/ubuntu noble Release
      404 Not Found [IP: 185.125.190.80 443]
Hit:10 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [921 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1040 kB]
Reading package lists... Done
E: The repository 'https://ppa.launchpadcontent.net/nginx/stable/ubuntu noble Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
pranitha@DESKTOP-SQFEICH:~$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (26.1.3-0ubuntu1~24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
pranitha@DESKTOP-SQFEICH:~$ sudo systemctl enable docker
pranitha@DESKTOP-SQFEICH:~$ sudo systemctl start docker
pranitha@DESKTOP-SQFEICH:~$ docker --version
docker: '--version' is not a docker command.
See 'docker --help'
pranitha@DESKTOP-SQFEICH:~$ sudo docker version
Client:
Version:      26.1.3
API version:  1.45
Go version:   go1.22.2
Git commit:   26.1.3-0ubuntu1~24.04.1
```



```
pranitha@DESKTOP-SQFEICH x + v
pranitha@DESKTOP-SQFEICH:~/docker-python09$ nano app.py
pranitha@DESKTOP-SQFEICH:~/docker-python09$ nano requirements.txt
pranitha@DESKTOP-SQFEICH:~/docker-python09$ nano Dockerfile
pranitha@DESKTOP-SQFEICH:~/docker-python09$ nano docker-compose.yml
pranitha@DESKTOP-SQFEICH:~/docker-python09$ sudo docker-compose build
WARN[0000] /home/pranitha/docker-python09/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove
it to avoid potential confusion
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 144.4s (11/11) FINISHED docker:default
=> [web internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 465B 0.0s
=> [web internal] load metadata for docker.io/library/python:3.11 5.5s
=> [web internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [web 1/5] FROM docker.io/library/python:3.11 130.2s
=> => resolve docker.io/library/python:3.11sha 0.0s
=> => sha256:ebfa8696e47a68cffe 9.88kB / 9.88kB 0.0s
=> => sha256:53218a9ae69d30aede 2.33kB / 2.33kB 0.0s
=> => sha256:255774e0027b72d 64.40MB / 64.40MB 71.6s
=> => sha256:18c0f2265fd94d4d17 6.18kB / 6.18kB 0.0s
=> => sha256:7cd785773db4440 48.47MB / 48.47MB 24.7s
=> => sha256:091eb8249475f42 24.01MB / 24.01MB 23.0s
=> => sha256:353e14e5cc47 211.35MB / 211.35MB 116.2s
=> => sha256:963091970bc2ac7e8 6.16MB / 6.16MB 28.2s
=> => extracting sha256:7cd785773db44407e20a679 4.0s
=> => sha256:e7235c43f7e31b1 24.31MB / 24.31MB 42.2s
=> => extracting sha256:091eb8249475f42de217265 1.0s
=> => sha256:7f221c50e407e12492524 250B / 250B 45.6s
=> => extracting sha256:255774e0027b72d2327719e 4.9s
=> => extracting sha256:353e14e5cc47664fba714a 11.1s
=> => extracting sha256:963091970bc2ac7e8b231ef 0.6s
=> => extracting sha256:e7235c43f7e31b18eb2e528 1.6s
=> => extracting sha256:7f221c50e407e12492524c3 0.0s
=> [web internal] load build context 0.0s
```

```
=> => sha256:18c0f2265fd94d4d17 6.18kB / 6.18kB 0.0s
=> => sha256:7cd785773db4440 48.47MB / 48.47MB 24.7s
=> => sha256:091eb8249475f42 24.01MB / 24.01MB 23.0s
=> => sha256:353e14e5cc47 211.35MB / 211.35MB 116.2s
=> => sha256:963091970bc2ac7e8 6.16MB / 6.16MB 28.2s
=> => extracting sha256:7cd785773db44407e20a679 4.0s
=> => sha256:e7235c43f7e31b1 24.31MB / 24.31MB 42.2s
=> => extracting sha256:091eb8249475f42de217265 1.0s
=> => sha256:7f221c50e407e12492524 250B / 250B 45.6s
=> => extracting sha256:255774e0027b72d2327719e 4.9s
=> => extracting sha256:353e14e5cc47664fba714a 11.1s
=> => extracting sha256:963091970bc2ac7e8b231ef 0.6s
=> => extracting sha256:e7235c43f7e31b18eb2e528 1.6s
=> => extracting sha256:7f221c50e407e12492524c3 0.0s
=> [web internal] load build context 0.0s
=> => transferring context: 916B 0.0s
=> [web 2/5] WORKDIR /app 0.6s
=> [web 3/5] COPY requirements.txt . 0.1s
=> [web 4/5] RUN pip install --no-cache-dir -r 7.5s
=> [web 5/5] COPY . . 0.1s
=> [web] exporting to image 0.3s
=> => exporting layers 0.2s
=> => writing image sha256:c924612907af41563110 0.0s
=> => naming to docker.io/library/docker-python 0.0s
=> [web] resolving provenance for metadata file 0.0s
[+] Building 1/1
✔web Built 0.0s
pranitha@DESKTOP-SQFEICH:~/docker-python09$ sudo docker-compose up -d
WARN[0000] /home/pranitha/docker-python09/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove
it to avoid potential confusion
[+] Running 2/2
✔Network docker-python09_default Created 0.2s
✔Container docker-python09-web-1 Started 0.5s
pranitha@DESKTOP-SQFEICH:~/docker-python09$ sudo docker ps
CONTAINER ID IMAGE COMMAND
```



```
pranitha@DESKTOP-SQFEICH x + v
=> [web internal] load build context      0.0s
=> => transferring context: 9168         0.0s
=> [web 2/5] WORKDIR /app                0.6s
=> [web 3/5] COPY requirements.txt .     0.1s
=> [web 4/5] RUN pip install --no-cache-dir -r 7.5s
=> [web 5/5] COPY . .                   0.1s
=> [web] exporting to image              0.3s
=> => exporting layers                   0.2s
=> => writing image sha256:c924612907af41563110 0.0s
=> => naming to docker.io/library/docker-python 0.0s
=> [web] resolving provenance for metadata file 0.0s
[+] Building 1/1
✓web Built 0.0s
pranitha@DESKTOP-SQFEICH:~/docker-python09$ sudo docker-compose up -d
WARN[0000] /home/pranitha/docker-python09/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 2/2
✓Network docker-python09_default Created 0.2s
✓Container docker-python09-web-1 Started 0.5s
pranitha@DESKTOP-SQFEICH:~/docker-python09$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND
CREATED       STATUS      PORTS
NAMES
ecbd9c3f61d8   docker-python09-web                 "python app.py"    18 seconds ago   Up 17 seconds    0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   d
pranitha@DESKTOP-SQFEICH:~/docker-python09$
pranitha@DESKTOP-SQFEICH:~/docker-python09$
```



## Jenkins Pipeline Through Git Token - Setup Procedure

### Step 1: Generate a Git Personal Access Token

Before configuring the Jenkins pipeline, you need to generate a **Personal Access Token (PAT)** from your Git service.

#### GitHub (Example)

1. **Log in to GitHub** and navigate to your profile.
2. Go to **Settings > Developer Settings > Personal Access Tokens**.
3. Click **Generate New Token**.
4. Select the necessary permissions for the token. For example, to clone repositories, select:
  - repo (full control of private repositories)
  - read:org (for organization repository access)
5. Generate the token and **copy it**. This token will act as the password when Jenkins connects to GitHub.

#### GitLab (Example)

1. **Log in to GitLab** and go to **Profile Settings > Access Tokens**.
2. Generate a new token with appropriate scopes (e.g., read\_repository).
3. **Save the token** to use in Jenkins.

#### Bitbucket (Example)

1. **Log in to Bitbucket** and go to **Personal Settings > App Passwords**.
2. Create an app password with necessary permissions (like repository read).
3. **Save the password** to use in Jenkins.

### Step 2: Store Git Token in Jenkins Credentials

Once you've generated the Git token, the next step is to store it securely in Jenkins.

1. **Log in to Jenkins** and navigate to the Jenkins dashboard.
2. In the left menu, click on **Manage Jenkins**.
3. Click on **Manage Credentials**.
4. Select the appropriate **scope** (e.g., (Global)).
5. Click on **Add Credentials**.
6. In the **Kind** dropdown, select **Username with password**.
7. In the **Username** field, enter your Git username (e.g., your-username for GitHub).

8. In the **Password** field, paste the **Git token** you generated.
9. Optionally, give it an ID (e.g., git-token-jenkins).
10. Click **OK** to save the credentials.

### Step 3: Configure Jenkins Pipeline

Now that the Git token is securely stored in Jenkins, you can configure a Jenkins pipeline to use it for Git interactions.

#### Example Pipeline Script (Declarative Pipeline)

You'll now set up a pipeline that uses Git for the source code. Here's an example using a declarative pipeline.

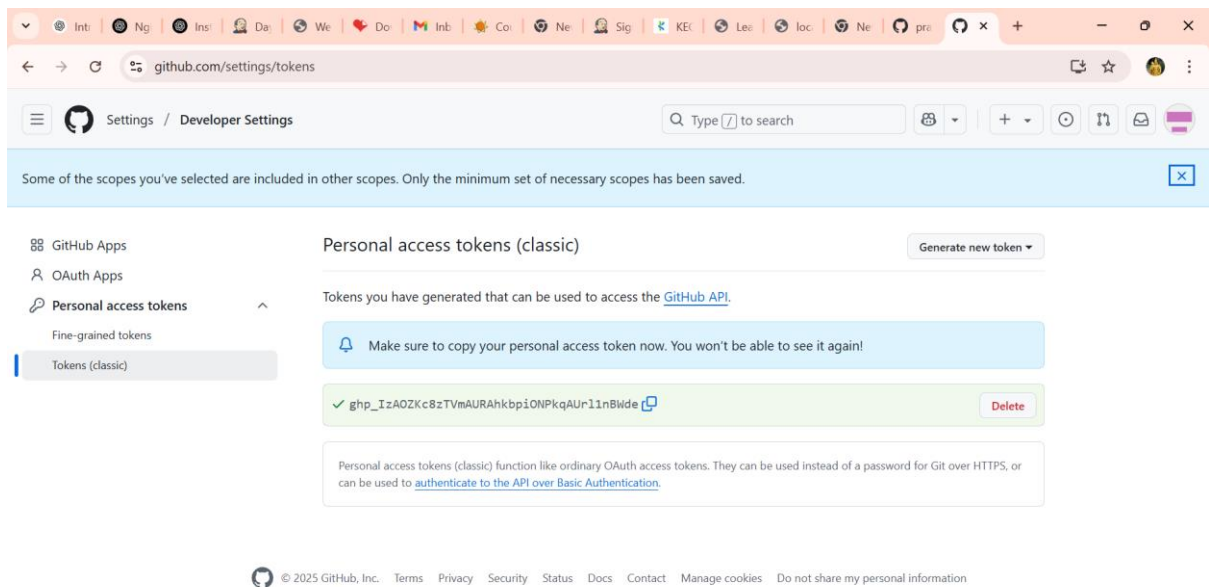
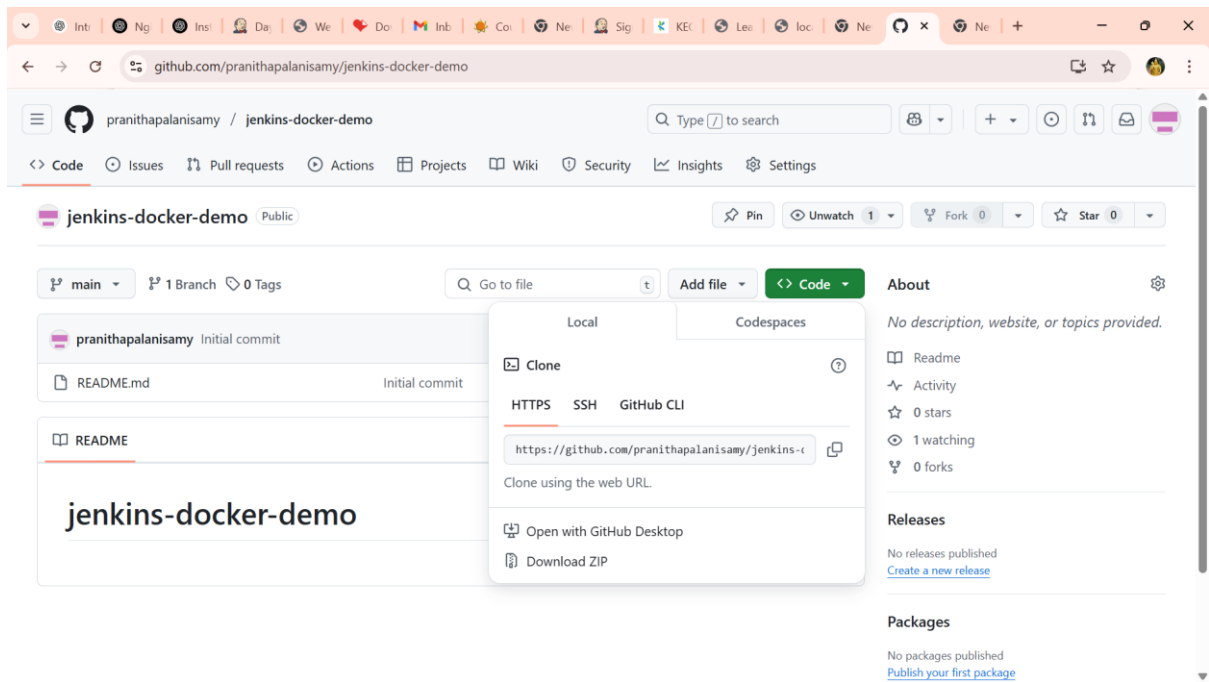
1. **Create a New Pipeline Job:**
  - Go to Jenkins Dashboard.
  - Click **New Item**, select **Pipeline**, and name your pipeline (e.g., Git-Pipeline).
  - Click **OK**.
2. **Configure the Pipeline:**
  - In the pipeline configuration, scroll to the **Pipeline** section.
  - Choose **Pipeline script from SCM**.
  - Set the **SCM** dropdown to **Git**.
  - In the **Repository URL** field, enter your repository URL (e.g., <https://github.com/yourusername/your-repository.git>).
  - Select **Credentials**. Choose the credentials you created earlier (e.g., git-token-jenkins).

### Step 4: Run the Jenkins Pipeline

- After configuring the pipeline, click **Save** and then **Build Now** to run the pipeline.
- Jenkins will use the credentials you provided to authenticate with Git, clone the repository, and run the pipeline steps.

### Step 5: Monitor and Troubleshoot

- If the pipeline fails, check the Jenkins job's **Console Output** for debugging information. Common issues can be due to incorrect credentials, Git URL, or permission issues.



IntNcIneDeWiDcInlCcNeSltKELeIorNeprPe

localhost:8080/view/all/newJob

Pranitha P log out


Dashboard > All > New Item

## New Item


Enter an item name

Day 2


Select an item type



**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds. etc.

OK

IntNcIneDeWiDcInlCcNeSltKELeIorNeprPe

localhost:8080/job/Day%202/configure

Pranitha P log out

Dashboard > Day 2 > Configuration

Configure

General

Triggers

Pipeline

Advanced

Jenkins Credentials Provider: Jenkins

Global (Jenkins, nodes, items, all child items, etc)

Username ?  
pranithapalanisamy

☐ Treat username as secret ?

Password ?  
.....

ID ?  
github-pranitha

Description ?


SaveApply

IntNeInDeWiDcInlCcNeSlcKELeLoNeprPe

localhost:8080/job/Day%202/

☆

⋮

 **Jenkins**

🔍 🔒 1 👤 Pranita P 🚪 log out

Dashboard > Day 2 >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

📁 Stages

✏️ Rename

❓ Pipeline Syntax

Day 2

✎ Add description

Permalinks

Builds

⋮ 🔗

No builds

IntNeInDeWiDcInlCcNeSlcKELeLoNeprPe

localhost:8080/job/Day%202/configure

☆

⋮

Dashboard > Day 2 > Configuration

Configure

⚙️ General

🕒 Triggers

📁 Pipeline

🔧 Advanced

Add ▾

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

[Pipeline Syntax](#)

Advanced

Advanced ▾

Save Apply

REST API Jenkins 2.501

## Jenkins Pipeline for Dockerized Application Deployment

This document provides a step-by-step guide on how the Jenkins pipeline automates the process of fetching the code from GitHub, building a Docker image, pushing it to a container registry, and deploying the application in a running Docker container.

---

### Pipeline Overview

The pipeline follows these key steps:

1. **Checkout Code** - Fetch the latest code from the GitHub repository.
  2. **Build Docker Image** - Create a Docker image for the application.
  3. **Login to Docker Registry** - Authenticate to the container registry.
  4. **Push to Container Registry** - Upload the built image to a Docker registry.
  5. **Stop & Remove Existing Container** - Stop and remove any existing container with the same name.
  6. **Run Docker Container** - Deploy a new container with the updated image.
  7. **Post Actions** - Handle success or failure messages.
- 

### Step-by-Step Execution

#### 1. Checkout Code

- Uses Jenkins credentials to authenticate and fetch the latest code from GitHub.
- Ensures secure access using stored credentials instead of exposing raw tokens.

#### Implementation:

```
stage('Checkout Code') {  
    steps {  
        withCredentials([usernamePassword(credentialsId: 'github-nisanthg1010',  
usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {  
            git url:  
"https://$GIT_USER:$GIT_TOKEN@github.com/nisanthg1010/Devops_Nisanth.git",  
branch: 'main'  
        }  
    }  
}
```

#### 2. Build Docker Image

- Builds the Docker image using the Dockerfile present in the repository.
- Tags the image with the latest version.

**Implementation:**

```
stage('Build Docker Image') {
    steps {
        sh 'docker build -t $DOCKER_IMAGE .'
    }
}
```

### 3. Login to Docker Registry

- Uses stored Jenkins credentials to log in securely to the Docker registry.
- Prevents exposing login credentials in the script.

**Implementation:**

```
stage('Login to Docker Registry') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'docker_nisanth', usernameVariable:
'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
            sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
    }
}
```

### 4. Push to Container Registry

- Pushes the newly built Docker image to the specified container registry.
- Ensures the latest version of the application is stored and accessible.

**Implementation:**

```
stage('Push to Container Registry') {
    steps {
        sh 'docker push $DOCKER_IMAGE'
    }
}
```

### 5. Stop & Remove Existing Container

- Stops and removes the running container if it exists.



- Prevents conflicts when deploying the new version.

### **Implementation:**

```
stage('Stop & Remove Existing Container') {
  steps {
    script {
      sh '''
      if [ "$(docker ps -aq -f name=$CONTAINER_NAME)" ]; then
        docker stop $CONTAINER_NAME || true
        docker rm $CONTAINER_NAME || true
      fi
      '''
    }
  }
}
```

## **6. Run Docker Container**

- Starts a new Docker container with the updated image.
- Maps the internal application port 5000 to 5001 on the host machine.

### **Implementation:**

```
stage('Run Docker Container') {
  steps {
    sh 'docker run -d -p 5001:5000 --name $CONTAINER_NAME $DOCKER_IMAGE'
  }
}
```

## **7. Post Actions**

- If successful, displays a success message.
- If failed, displays an error message.

### **Implementation:**

```
post {
  success {
    echo "Build, push, and container execution successful!"
  }
}
```

```

    }

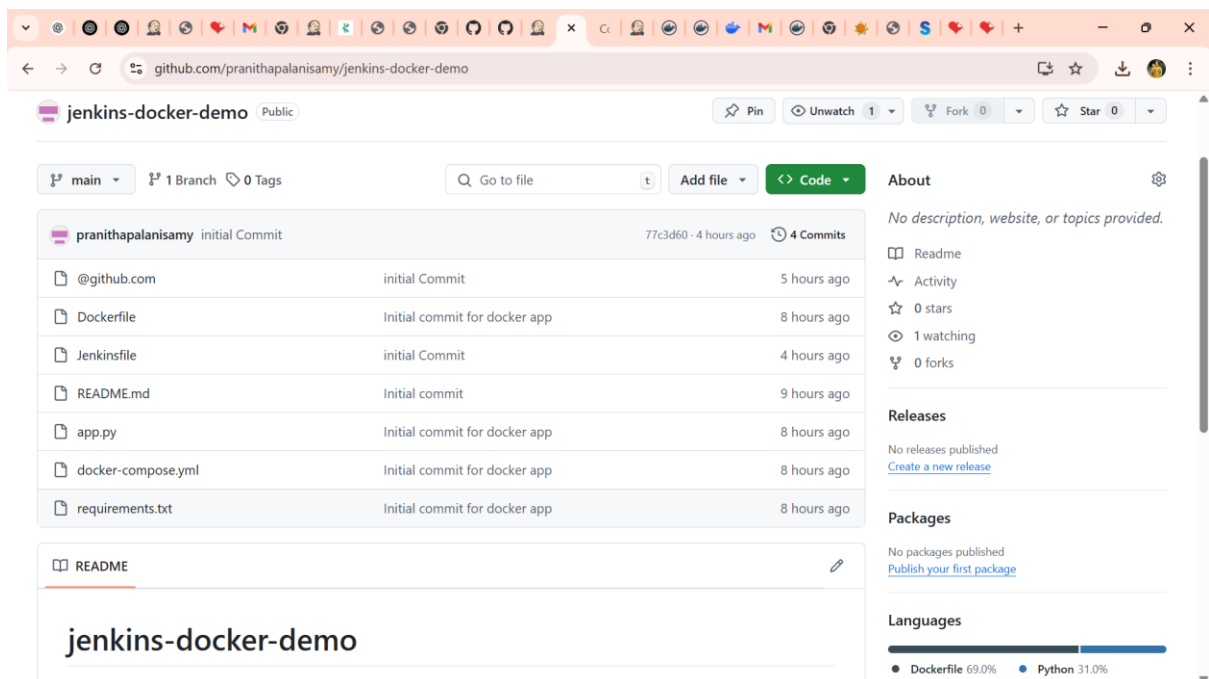
    failure {
        echo "Build or container execution failed."
    }
}

```

---

## Conclusion

This Jenkins pipeline automates the entire process of fetching the code, building a Docker image, pushing it to a registry, and deploying the container. It ensures a seamless CI/CD workflow, making application updates smooth and efficient. 🚀



localhost:8080/job/Day%202/5/console

# Jenkins

Dashboard > Day 2 > #5

- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#5'
- Timings
- Git Build Data
- Git Build Data
- Pipeline Overview
- Pipeline Console
- Restart from Stage
- Replay
- Pipeline Steps

## Console Output

Download Copy View as plain text

```
Started by user Pranitha P
Obtained Jenkinsfile from git https://github.com/pranithapalanisamy/jenkins-docker-demo.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Day 2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Day 2/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/pranithapalanisamy/jenkins-docker-demo.git # timeout=10
Fetching upstream changes from https://github.com/pranithapalanisamy/jenkins-docker-demo.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
```

localhost:5001

Hello, World! Running inside Docker!

hub.docker.com/repositories/pranitha025

NewIntroducing our new CEO Don Johnson - Read More →

docker hub

ExploreMy Hub

Search Docker Hub

ctrl+K

P

pranitha025

Docker Personal

Repositories

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories

All repositories within the pranitha025 namespace.

Search by repository name

All content

Create a repository

Name	Last Pushed ↑	Contains	Visibility	Scout
pranitha025/docker-app	about 3 hours ago	IMAGE	Public	Inactive

1-1 of 1

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

Cookies Settings

Reject All

Accept All Cookies

X

hub.docker.com/repository/docker/pranitha025/docker-app/general

Using 0 of 1 private repositories. [Get more](#)

docker hub

ExploreMy Hub

Search Docker Hub

ctrl+K

P

pranitha025

Docker Personal

Repositories

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

[Repositories](#) / [docker-app](#) / [General](#)

pranitha025/docker-app

Last pushed about 3 hours ago

Add a description

Add a category

Docker commands

To push a new tag to this repository.

`docker push pranitha025/docker-app:tagname`

Public view

General

Tags

Image Management BETA

Collaborators

Webhooks

Settings

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	about 3 hours

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

Cookies Settings

Reject All

Accept All Cookies

X

https://hub.docker.com/repository/docker/pranitha025/docker-app/general

pranitha025  
Docker Personal

Repositories

Settings

Default privacy

Notifications

Billing


Usage

Pulls

Storage

hub.docker.com/repository/docker/pranitha025/docker-app/tags/latest/sha256-b72735f017b9219d5cef2852acff01a2973f914c98c29d6215e8d1047ed37016

Repositories / docker-app / Tags / latest



pranitha025/docker-app:latest

MANIFEST DIGEST sha256:b72735f017b9219d5cef2852acff01a2973f914c98c29d6215e8d1047ed37016

Delete Tag

OS/ARCH  
linux/amd64

COMPRESSED SIZE  
366.57 MB

LAST PUSHED  
8 minutes by pranitha025

TYPE  
Image

MANIFEST DIGEST  
sha256:b72735f0...

Image Layers

Vulnerabilities

Image Layers

1 # debian.sh --arch 'amd64' out/ 46.22 MB

2 RUN /bin/sh -c set -eux; 22.9 MB

3 RUN /bin/sh -c set -eux; 61.41 MB

4 RUN /bin/sh -c set -ex; 201.56 MB

5 ENV PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/games/bin

Command

# debian.sh --arch 'amd64' out/ 'bookworm' '01742169600'

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

Cookies Settings

Reject All

Accept All Cookies