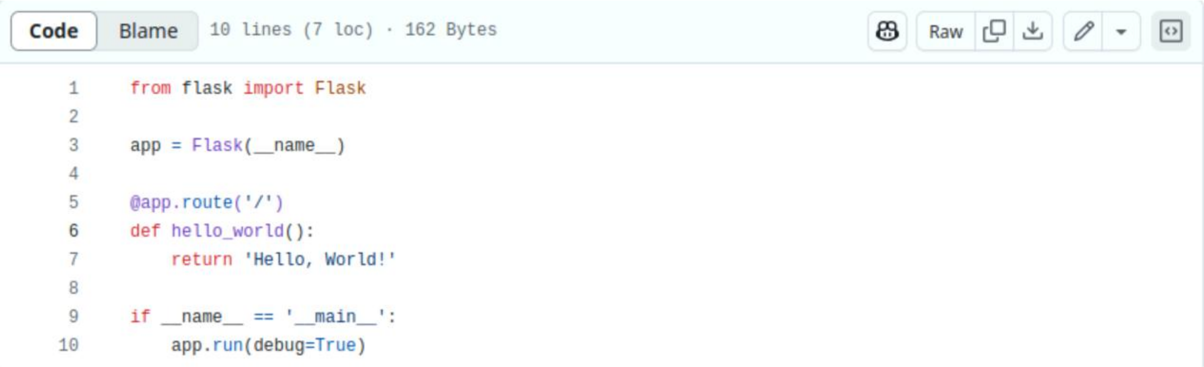
Project 1: Automate Docker built and push using Jenkins file.

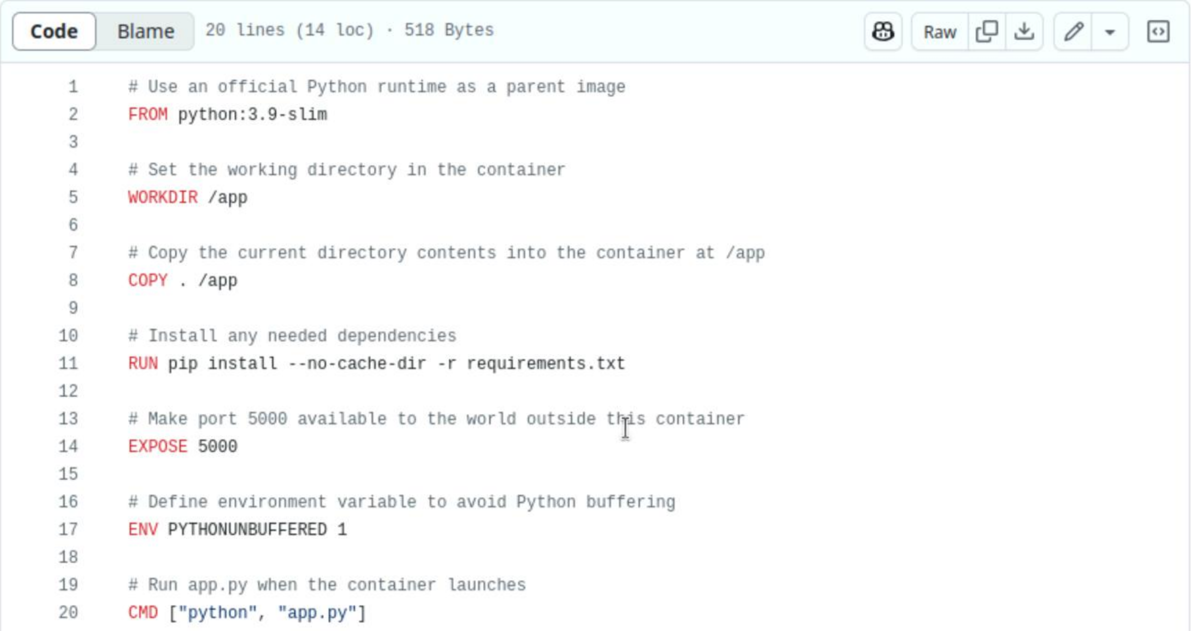
1. Setup a Simple Flask App

Project Structure : app.py has main Flask Application File.



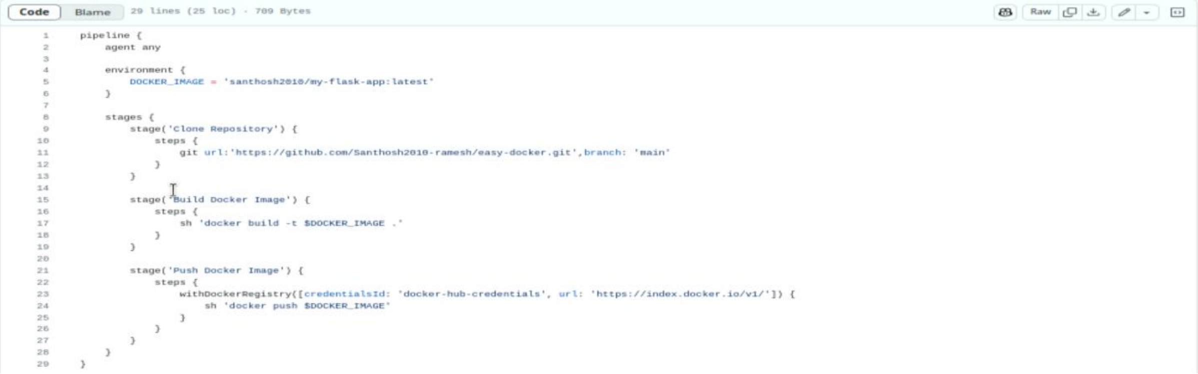
```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True)
```

2. Docker File: Defines the Docker image for the Flask App.



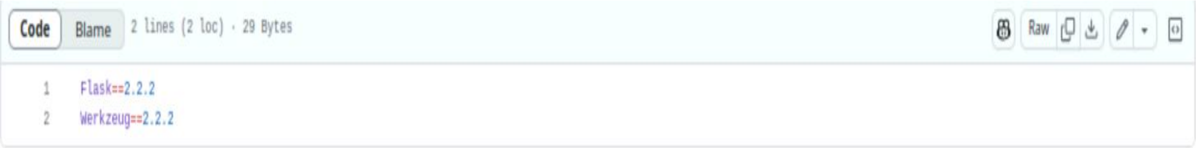
```dockerfile
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

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

# Install any needed dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable to avoid Python buffering
ENV PYTHONUNBUFFERED 1

# Run app.py when the container launches
CMD ["python", "app.py"]
```

3. Jenkinsfile : Contains the Jenkins Pipeline Configuration.



```groovy
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'santhosh2010/my-flask-app:latest'
    }

    stages {
        stage('Clone Repository') {
            steps {
                git url:'https://github.com/Santhosh2010-ramesh/easy-docker.git',branch: 'main'
            }
        }

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

        stage('Push Docker Image') {
            steps {
                withDockerRegistry([credentialsId: 'docker-hub-credentials', url: 'https://index.docker.io/v1/']) {
                    sh 'docker push $DOCKER_IMAGE'
                }
            }
        }
    }
}
```

4. Requirements.txt List of Dependencies (Flask and other)

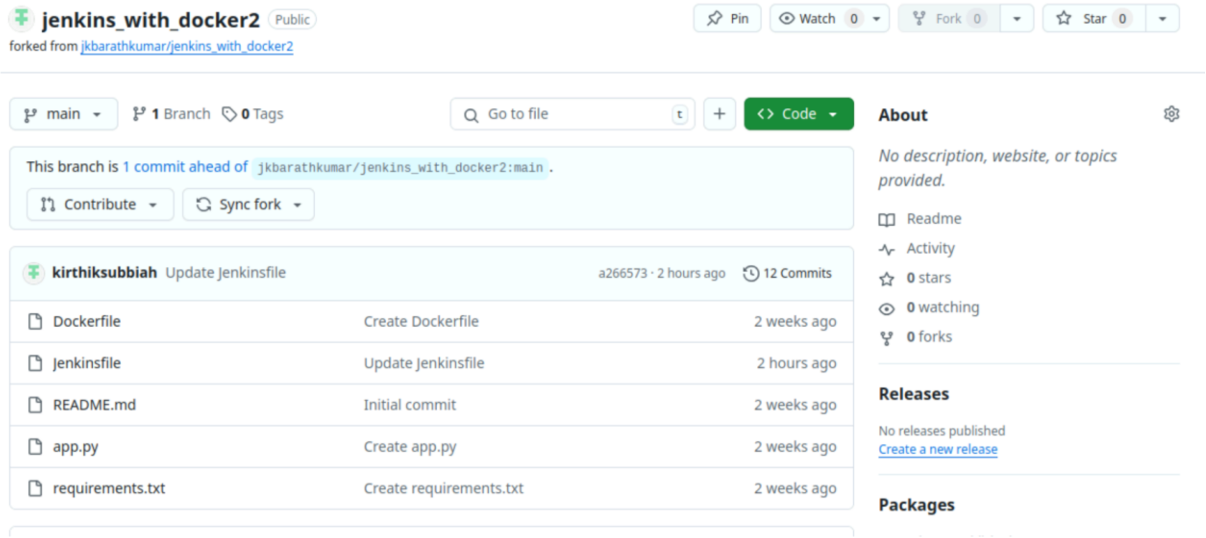


```
Flask==2.2.2
Werkzeug==2.2.2
```

5. Push the code to GitHub

Make sure you have a GitHub repository created for the project.

Push all the files (app.py, requirements.txt, Docker File, Jenkins File) to the GitHub repository.



jenkins_with_docker2 Public

forked from jkbarathkumar/jenkins_with_docker2

This branch is 1 commit ahead of jkbarathkumar/jenkins_with_docker2:main .

kirthiksubbiah Update Jenkinsfile a266573 · 2 hours ago 12 Commits

Dockerfile	Create Dockerfile	2 weeks ago
Jenkinsfile	Update Jenkinsfile	2 hours ago
README.md	Initial commit	2 weeks ago
app.py	Create app.py	2 weeks ago
requirements.txt	Create requirements.txt	2 weeks ago

About

No description, website, or topics provided.

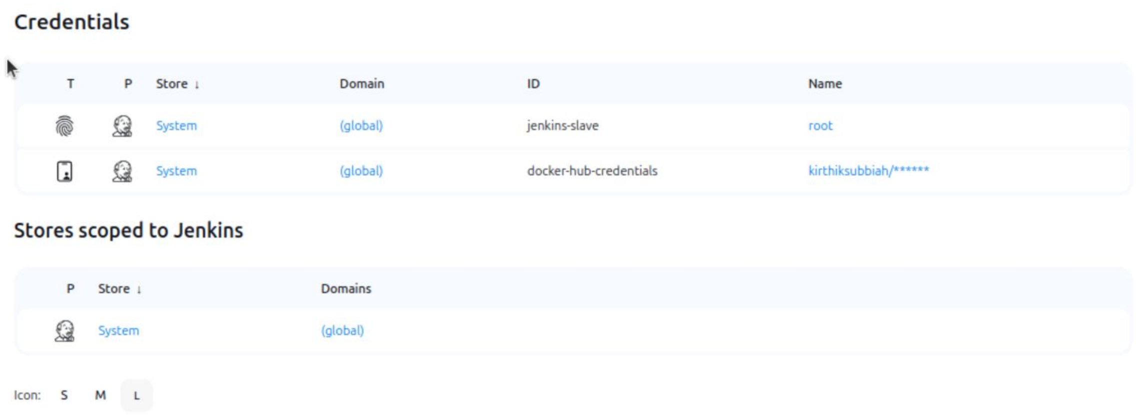
Releases

No releases published
Create a new release

6. Configure Docker Hub Credentials in Jenkins

Goto Jenkins > manage Credentials.

Add new Credentials:
Username & Password: You Docker Hub username and password.
ID: Name it something like dockerhub-creds(the same name used in the Jenkinsfile)



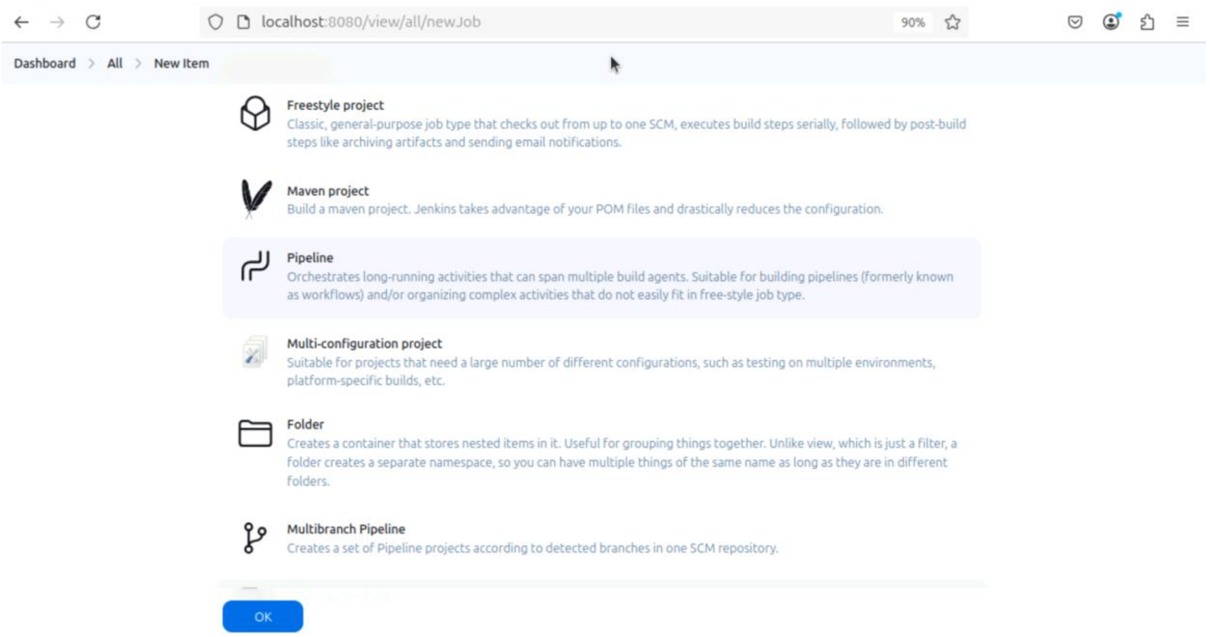
Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	jenkins-slave	root
		System	(global)	docker-hub-credentials	kirthiksubbiah/******

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

7. **Create a New Pipeline in Jenkinsfile**
   In Jenkinsfile, click New Item > Pipeline.
   Enter a new for the Pipeline.
   Under Pipeline definition, Select Pipeline Script from SCM.
   Select Git as the SCm.
   Enter the Github repository URL.
   Set the branch (typically master or main)
   Click Save



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.

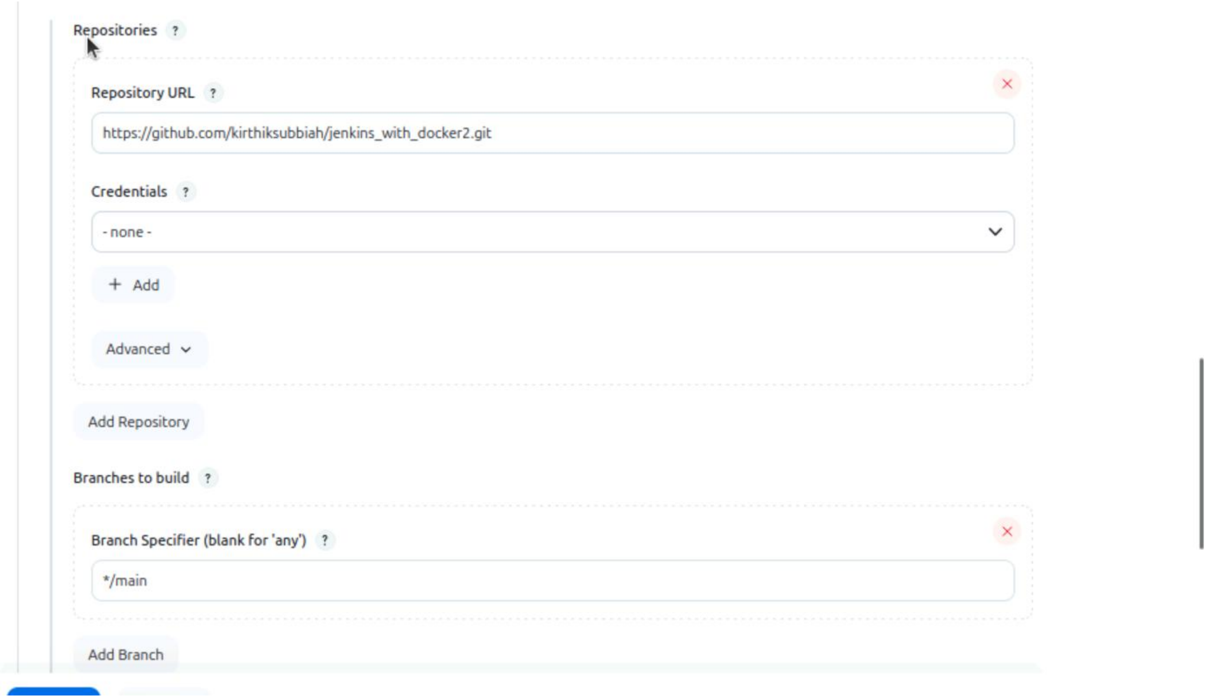
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.



Repositories ?

Repository URL ?

https://github.com/kirthiksubbiah/jenkins_with_docker2.git

Credentials ?

- none -

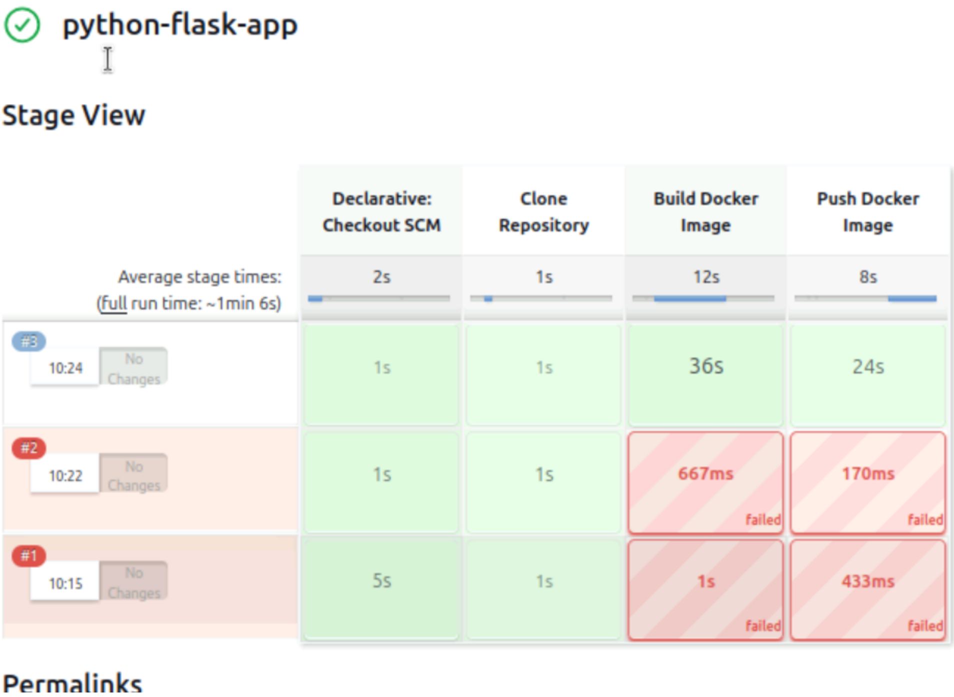
Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

8. Click Build Now

Click Build Now in Jenkins to trigger the build.

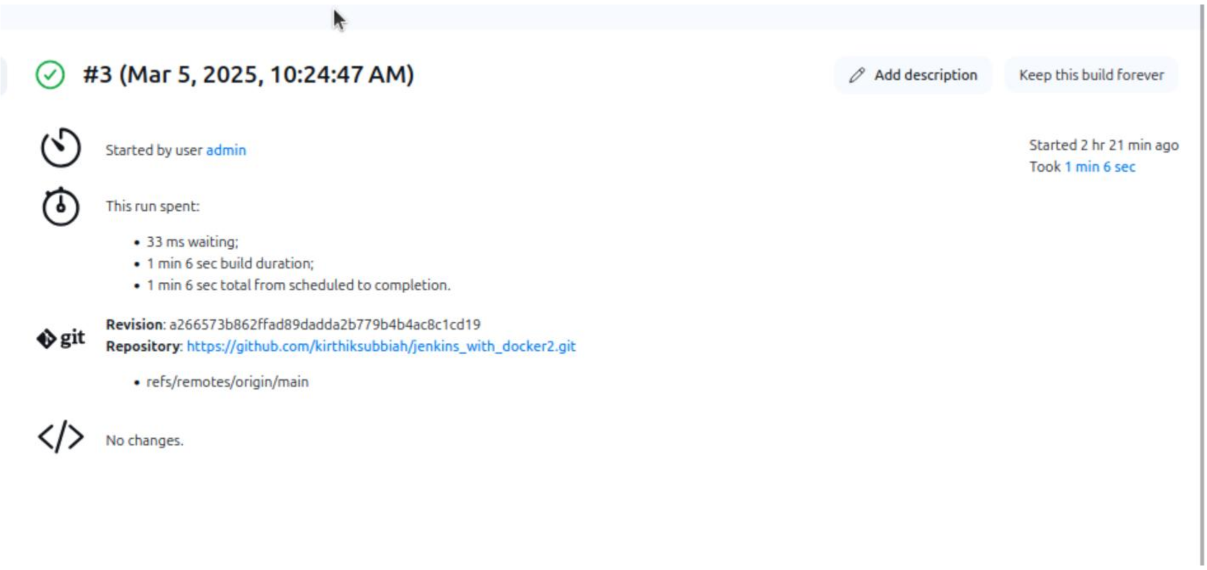


python-flask-app

Stage View

	Declarative: Checkout SCM	Clone Repository	Build Docker Image	Push Docker Image
Average stage times: (full run time: ~1min 6s)	2s	1s	12s	8s
#3 10:24 No Changes	1s	1s	36s	24s
#2 10:22 No Changes	1s	1s	667ms failed	170ms failed
#1 10:15 No Changes	5s	1s	1s failed	433ms failed

Permalinks



#3 (Mar 5, 2025, 10:24:47 AM)

Started by user admin

Started 2 hr 21 min ago
Took 1 min 6 sec

This run spent:

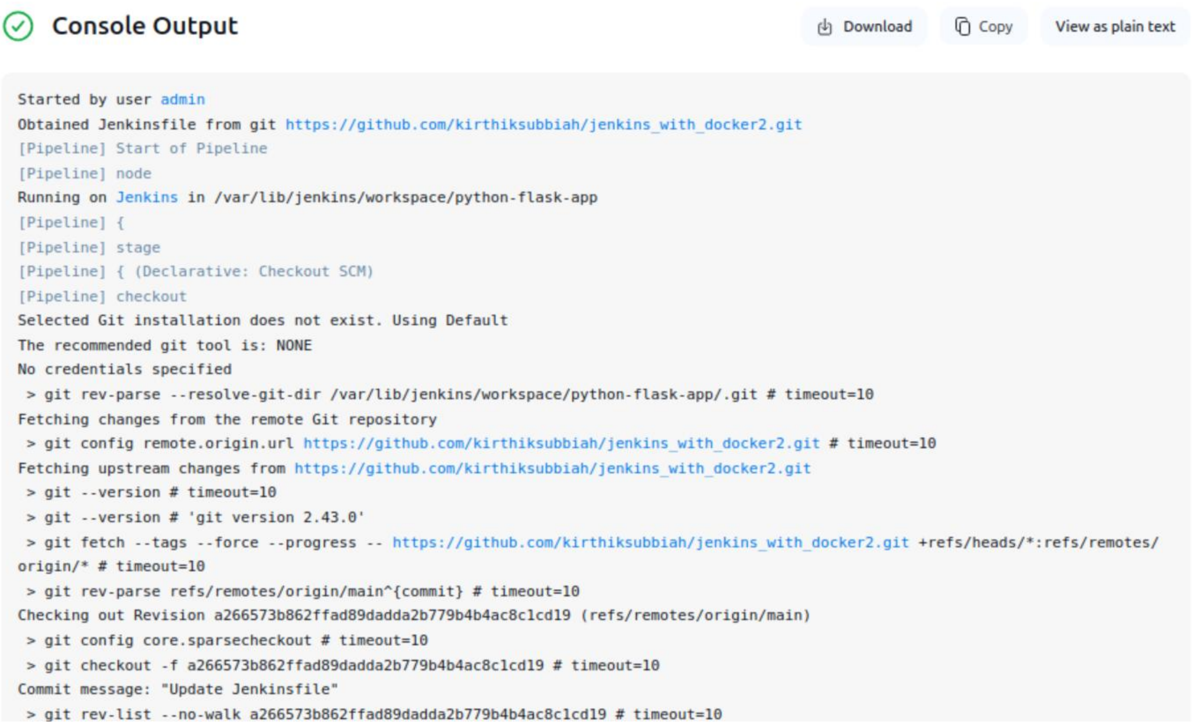
- 33 ms waiting;
- 1 min 6 sec build duration;
- 1 min 6 sec total from scheduled to completion.

git

Revision: a266573b862ffad89dadda2b779b4b4ac8c1cd19
Repository: https://github.com/kirthiksubbiah/jenkins_with_docker2.git

- refs/remotes/origin/main

No changes.



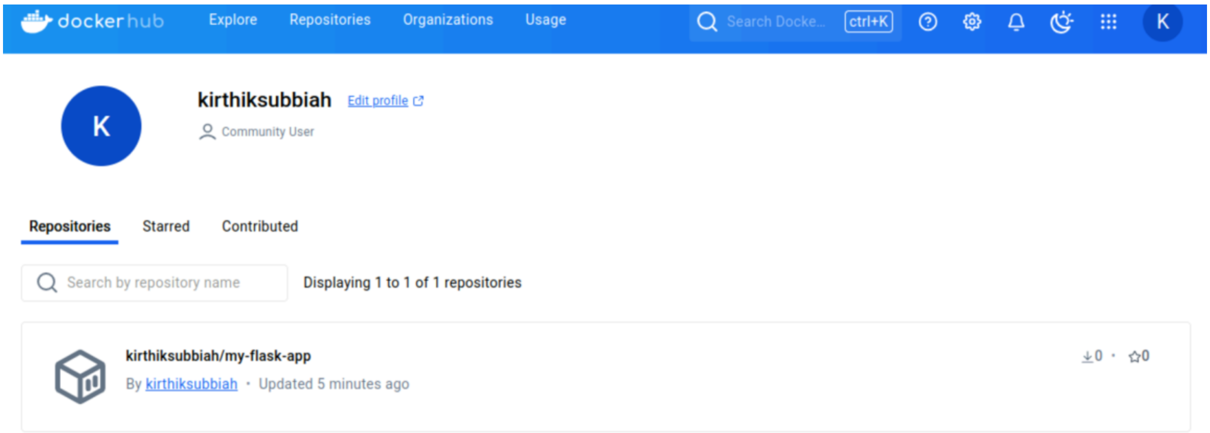
Console Output

```
Started by user admin
Obtained Jenkinsfile from git https://github.com/kirthiksubbiah/jenkins_with_docker2.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/python-flask-app
[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/python-flask-app/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/kirthiksubbiah/jenkins_with_docker2.git # timeout=10
Fetching upstream changes from https://github.com/kirthiksubbiah/jenkins_with_docker2.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress -- https://github.com/kirthiksubbiah/jenkins_with_docker2.git +refs/heads/*:refs/remotes/
origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision a266573b862ffad89dadda2b779b4b4ac8c1cd19 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f a266573b862ffad89dadda2b779b4b4ac8c1cd19 # timeout=10
Commit message: "Update Jenkinsfile"
 > git rev-list --no-walk a266573b862ffad89dadda2b779b4b4ac8c1cd19 # timeout=10
```

9. Verify Docker Image on Docker Hub.

After the Build Finishes, log into your Docker Hub Account.

You should see the my_flask_app image under Repositories with the latest tag.



kirthiksubbiah

Displaying 1 to 1 of 1 repositories

kirthiksubbiah/my-flask-app

By kirthiksubbiah · Updated 5 minutes ago