

# TASK 2- Automating CI/CD Pipeline with Jenkins

**Name:** Swetha M

**Rollno:** 22CSR217

## STEP 1: Installation of Docker

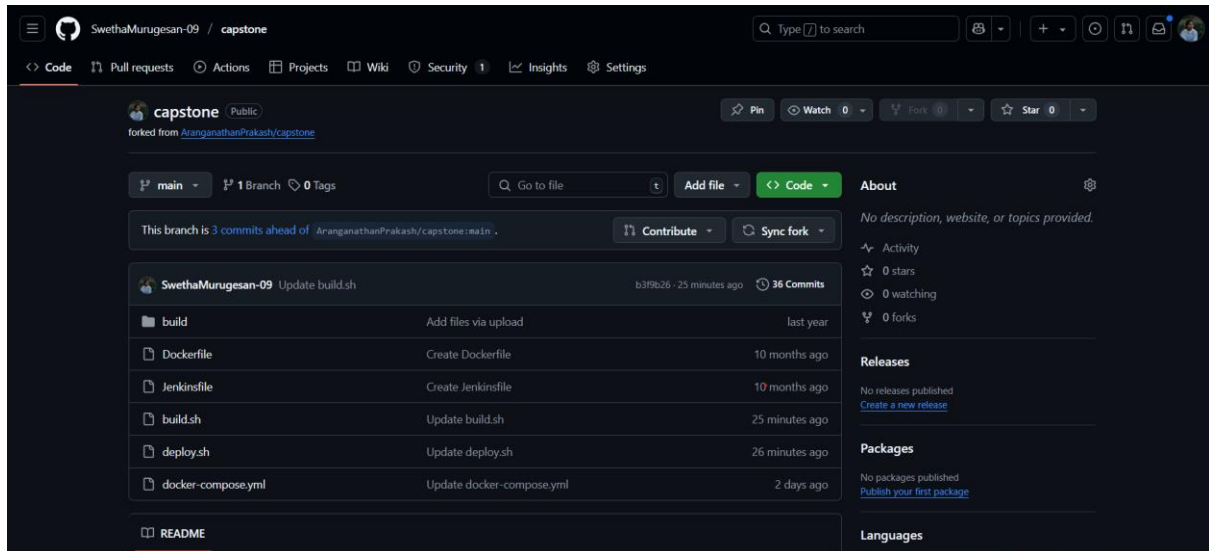
Install the Docker using the following commands:

- `sudo apt install docker.io`
- `docker --version`
- `sudo systemctl start docker`
- `sudo systemctl enable docker`
- `sudo systemctl status docker`

```
root@swetha:~/dock# sudo apt-get update
Ign:1 https://pkg.jenkins.io/debian binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian binary/ Release
Get:3 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Get:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [671 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [130 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [8968 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [6936 B]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [820 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [177 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.0 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [16.9 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [208 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Ign:18 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages
Ign:19 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en
Ign:20 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [13.5 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1040 kB]
Get:23 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]
Get:24 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [25.8 kB]
Get:25 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:26 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [922 kB]
Get:27 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:28 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [20.0 kB]
Get:29 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:30 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [209 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Fetched 5009 kB in 8s (652 kB/s)
Reading package lists... Done
root@swetha:~/dock# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:2 https://pkg.jenkins.io/debian binary/ InRelease
Hit:3 https://pkg.jenkins.io/debian binary/ Release
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease
```


## Step 2: Fork a GitHub repository

Fork a copy of a GitHub repository, which will create a clone of that repository in your own account.



## STEP 3: Create a Pipeline Job in the Jenkins

- In Jenkins, create a job using a pipeline.
- In the 'Definition' section, choose 'Pipeline script from SCM'.
- Under SCM, select 'Git', then paste the GitHub repository link in the 'Repository URL' field, which contains the script files for your job.
- Change the branch to match the one you're using in Git, and verify the filename.
- Finally, click 'OK' to proceed.

 Jenkins

Dashboard

All

New Item

Search

Swetha

log out

Dashboard

All


New Item


### New Item


Enter an item name


dev

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.

 **Folder**

OK

## Step 4: Give the repository url and branch name and script path from github

Dashboard

dev

Configuration

### Configure

General

Triggers

Pipeline

Advanced

Git

Repositories

Repository URL

https://github.com/SwethaMurugesan-09/capstone

Credentials

- none -

+ Add

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Save

Apply

Dashboard

dev

Configuration

### Configure

General

Triggers

Pipeline

Advanced

Branch Specifier (blank for 'any')

\*/main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

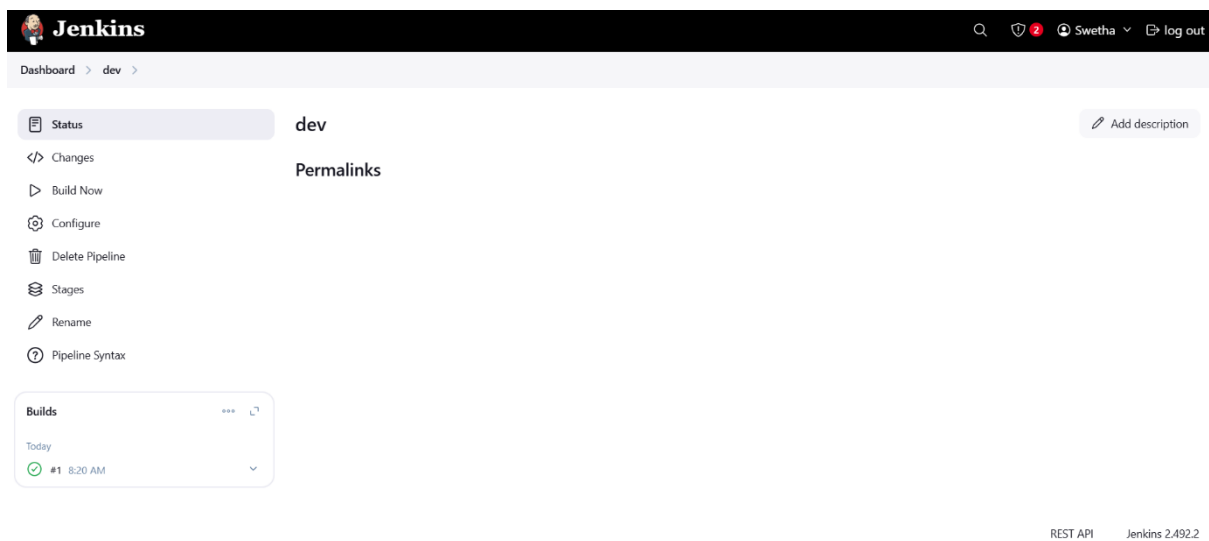
☒ Lightweight checkout

[Pipeline Syntax](#)

Save

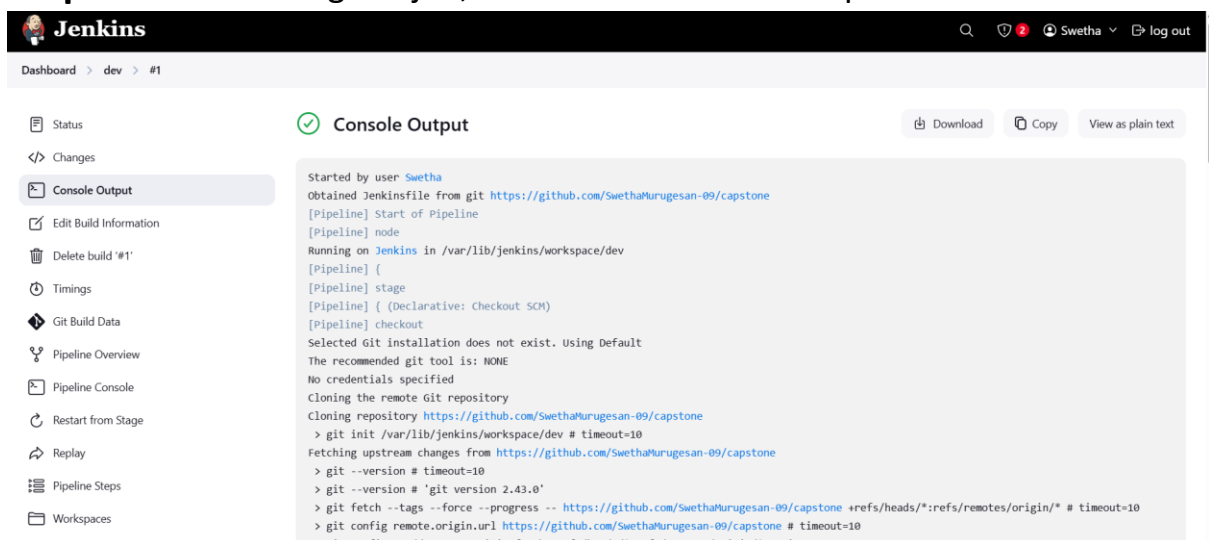
Apply

## Step 5: Give build now in dashboard, it will build the project



The screenshot shows the Jenkins dashboard for a job named 'dev'. The left sidebar contains a list of actions: Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The 'Build Now' button is highlighted. The main area shows the 'dev' job with a 'Permalinks' section. Below this, there is a 'Builds' section showing a single build, '#1', which is green and completed at 8:20 AM. The top right of the dashboard shows the user 'Swetha' and a 'log out' button. The bottom right corner indicates 'REST API' and 'Jenkins 2.492.2'.

## Step 6: After creating the job, build it. The console output will be



The screenshot shows the Jenkins console output for the 'dev' job, build '#1'. The left sidebar contains a list of actions: Status, Changes, Console Output, Edit Build Information, Delete build '#1', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Restart from Stage, Replay, Pipeline Steps, and Workspaces. The 'Console Output' button is highlighted. The main area shows the console output, which includes the following text:

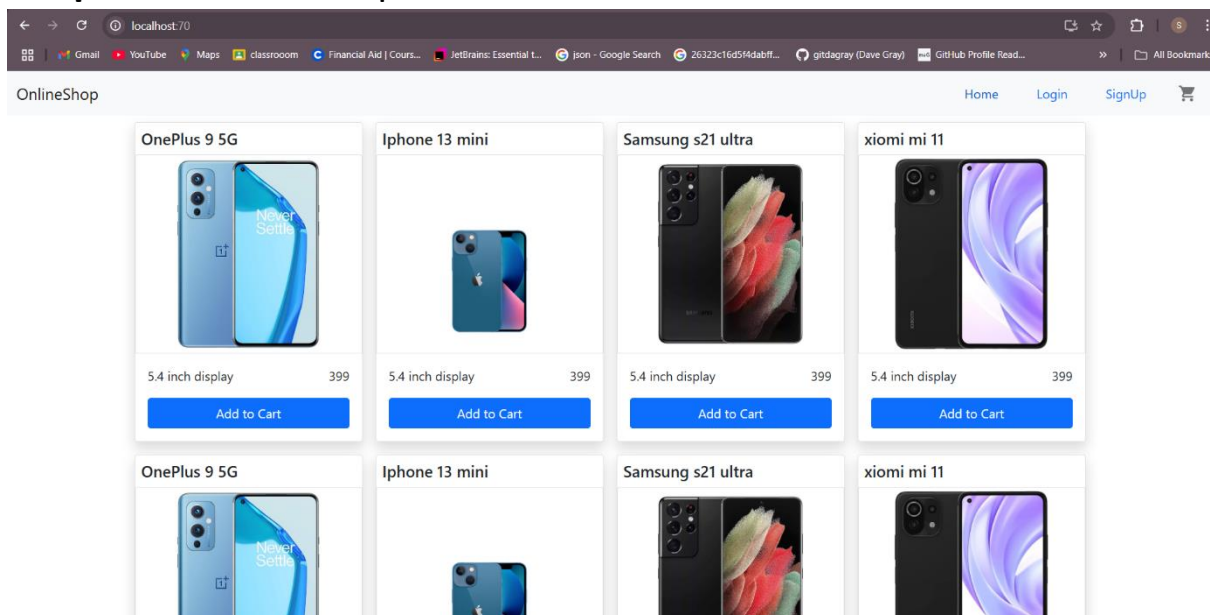
```
Started by user Swetha
Obtained Jenkinsfile from git https://github.com/SwethaMurugesan-09/capstone
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/dev
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/SwethaMurugesan-09/capstone
> git init /var/lib/jenkins/workspace/dev # timeout=10
Fetching upstream changes from https://github.com/SwethaMurugesan-09/capstone
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/SwethaMurugesan-09/capstone +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/SwethaMurugesan-09/capstone # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
```

## Step 7: Run the command to see the output in localhost

```
root@Swetha:~# sudo apt-get install docker-buildx-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-buildx-plugin is already the newest version (0.22.0-1~ubuntu.24.04~noble).
0 upgraded, 0 newly installed, 0 to remove and 131 not upgraded.
root@Swetha:~#
```

```
root@Swetha:~# docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
swethamurugesan/devopsgit  latest     8549df71284f  30 minutes ago  195MB
test1                latest     8549df71284f  30 minutes ago  195MB
test                 latest     69a3483fcf3b  4 hours ago    192MB
swethamurugesan/index.dev  latest     69a3483fcf3b  4 hours ago    192MB
swethamurugesan/nginx      latest     53a18edff809  6 weeks ago    192MB
nginx                 latest     53a18edff809  6 weeks ago    192MB
swethamurugesan/devops.dev  latest     53a18edff809  6 weeks ago    192MB
root@Swetha:~# docker run -itd -p 70:80 test1
064b78012d78d92bb348debb3983f19c03518c00b710d7934ca1d7bb32878204
root@Swetha:~#
```

## Output: View the output in browser



## Setting Up a GitHub Webhook for Automatic Project Builds in Jenkins: STEPS:

- To create a webhook in GitHub, install ngrok using the following command `sudo snap install ngrok`
- Create an account on the official website, and obtain the authentication or configuration command.

- After running the command in the terminal, you will receive the webhook URL for the GitHub repository.
- Add this webhook URL to your GitHub repository for automatic builds of the project.
- Additionally, in the Jenkins configuration page, check the box labeled GitHub hook trigger for GITScm polling.

