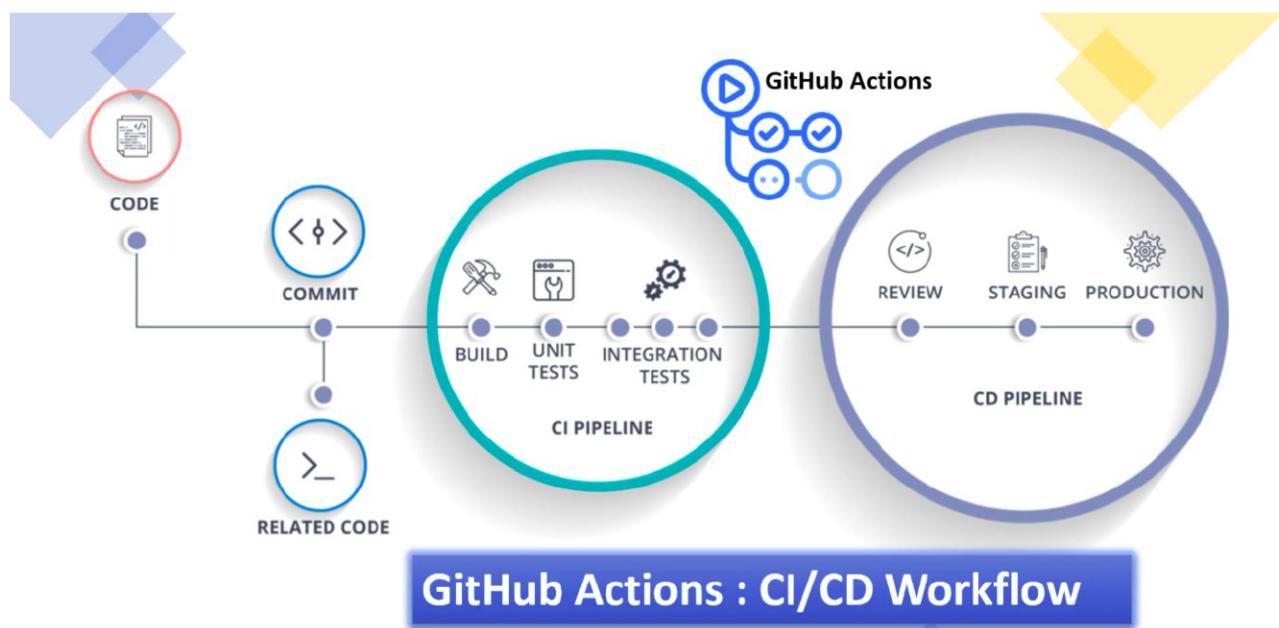


# SpringBoot - Build CI/CD Pipeline Using GitHub Actions

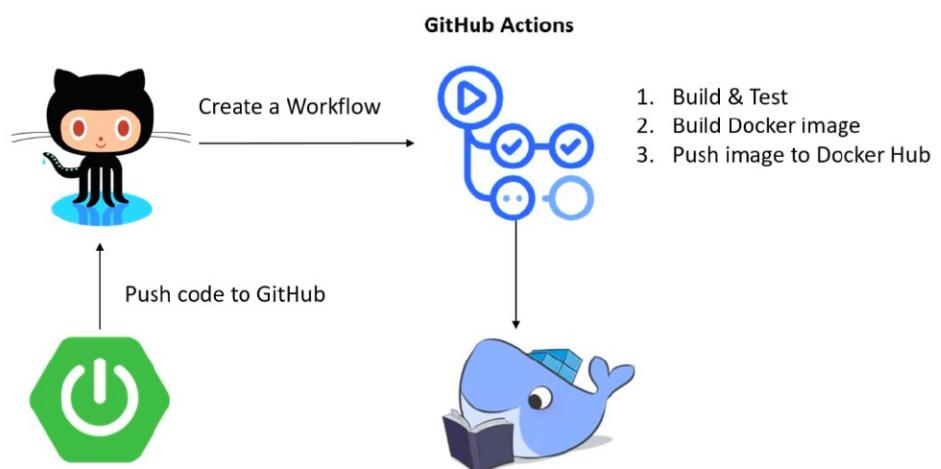
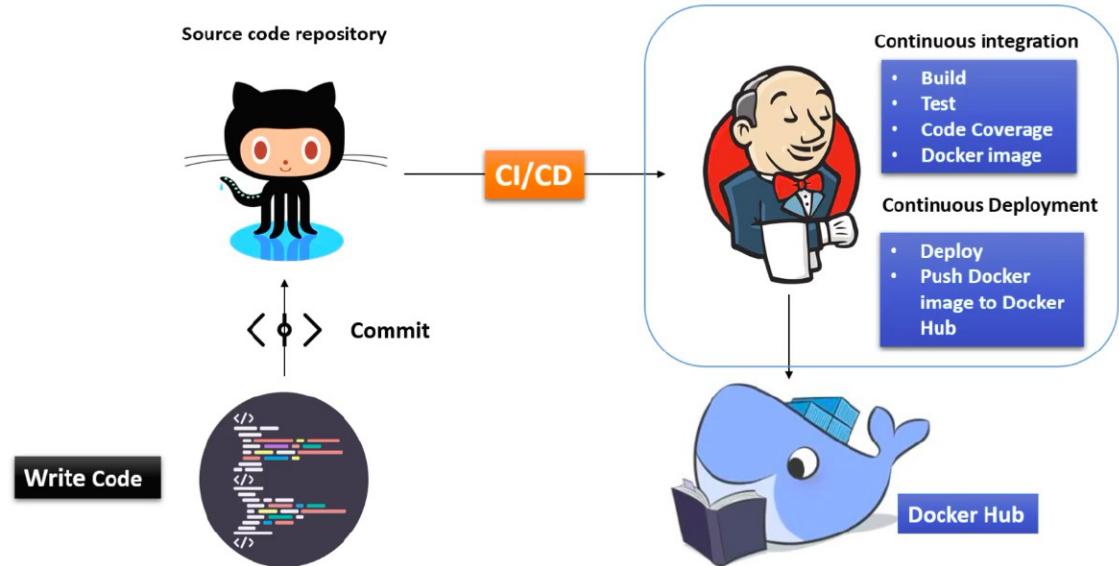
## What is CI/CD?

- **CI (Continuous Integration)**: Automates building & testing code whenever changes are pushed to GitHub.
- **CD (Continuous Delivery/Deployment)**: Automates packaging, creating Docker images, and deploying apps (to Docker Hub, Kubernetes, or Cloud).

We'll build a **Spring Boot project**, configure **GitHub Actions** for automation, and push a **Docker image** to Docker Hub.



## Jenkins as CI/CD



### Step 1: First create a Spring boot sample project

- Use **Spring Initializr** or **IntelliJ** to generate a simple Spring Boot application.
- Add dependencies like **Spring Web** (or others as required).
- Test locally to ensure the project runs.

## Step 2: Push the spring project on Github.

Initialize Git repository:

```
git init
```

- Add files:

```
git add .
```

- Commit:

```
git commit -m "Initial commit"
```

- Push to GitHub repository:

```
git remote add origin <repo-url>
git push -u origin master
```

The screenshot shows a GitHub repository page for 'github-actions-example'. The 'Code' tab is selected, showing the 'master' branch. The 'README.md' file is open, displaying the content: 'github actions example'. Above the file content, there is a commit history table:

Name	Last commit message	Last commit date
src	first commit	6 minutes ago
README.md	Create README.md	1 minute ago
pom.xml	first commit	6 minutes ago

## Step 3: Go to the Actions -> Java with Maven then click on Configure

The screenshot shows the GitHub Actions setup interface for the 'github-actions-example' repository. The 'Actions' tab is selected. A 'Get started with GitHub Actions' section is present, followed by a 'Suggested for this repository' section. Three workflow cards are shown:

- Publish Java Package with Maven** (By GitHub Actions): Build a Java Package using Maven and publish to GitHub Packages. [Configure](#)
- Java with Maven** (By GitHub Actions): Build and test a Java project with Apache Maven. [Configure](#)
- Publish Java Package with Gradle** (By GitHub Actions): Build a Java Package using Gradle and publish to GitHub Packages. [Configure](#)

A screenshot of a GitHub Actions workflow editor in a web browser. The URL is [github.com/Avi9151/github-actions-example/new/master?filename=.github%2Fworkflow.yml](https://github.com/Avi9151/github-actions-example/new/master?filename=.github%2Fworkflow.yml). The code editor shows a YAML configuration file:

```
1  # This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the workflow execution
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-maven
3
4  # This workflow uses actions that are not certified by GitHub.
5  # They are provided by a third-party and are governed by
6  # separate terms of service, privacy policy, and support
7  # documentation.
8
9  name: Java CI with Maven
10
11 on:
12   push:
13     branches: [ "master" ]
14   pull_request:
15     branches: [ "master" ]
16
17 jobs:
18   build:
19
20     runs-on: ubuntu-latest
21
```

The right side of the screen shows the GitHub Marketplace with three featured actions: "Upload a Build Artifact", "Setup Java JDK", and "Close Stale Issues".

A screenshot of a GitHub Actions workflow editor in a web browser, showing the same workflow configuration as the previous screenshot, but with additional steps added. The code editor now includes:

```
16
17 jobs:
18   build:
19
20     runs-on: ubuntu-latest
21
22   steps:
23     - uses: actions/checkout@v4
24     - name: Set up JDK 17
25     - uses: actions/setup-java@v4
26     - with:
27       java-version: '17'
28       distribution: 'temurin'
29       cache: maven
30     - name: Build with Maven
31     - run: mvn -B package --file pom.xml
32
33   # Optional: Uploads the full dependency graph to GitHub to improve the quality of Dependabot alerts this repository.
34   - name: Update dependency graph
35     uses: advanced-security/maven-dependency-submission-action@571e99aah1055c2e71a1e2309b9691de18d6b7d6
36
```

## Step 4: Rename the `name` and `run`

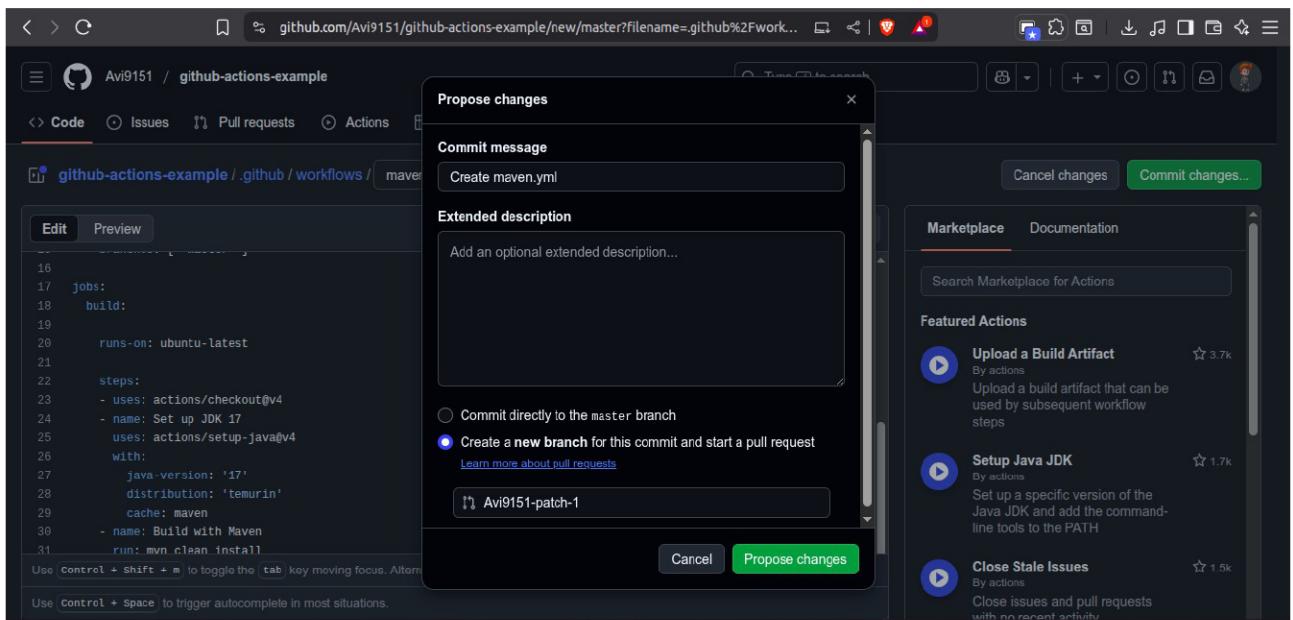
`name`: project Cicd flow

`run`: mvn clean install

## Step 5: Click on the **Commit changes** button.

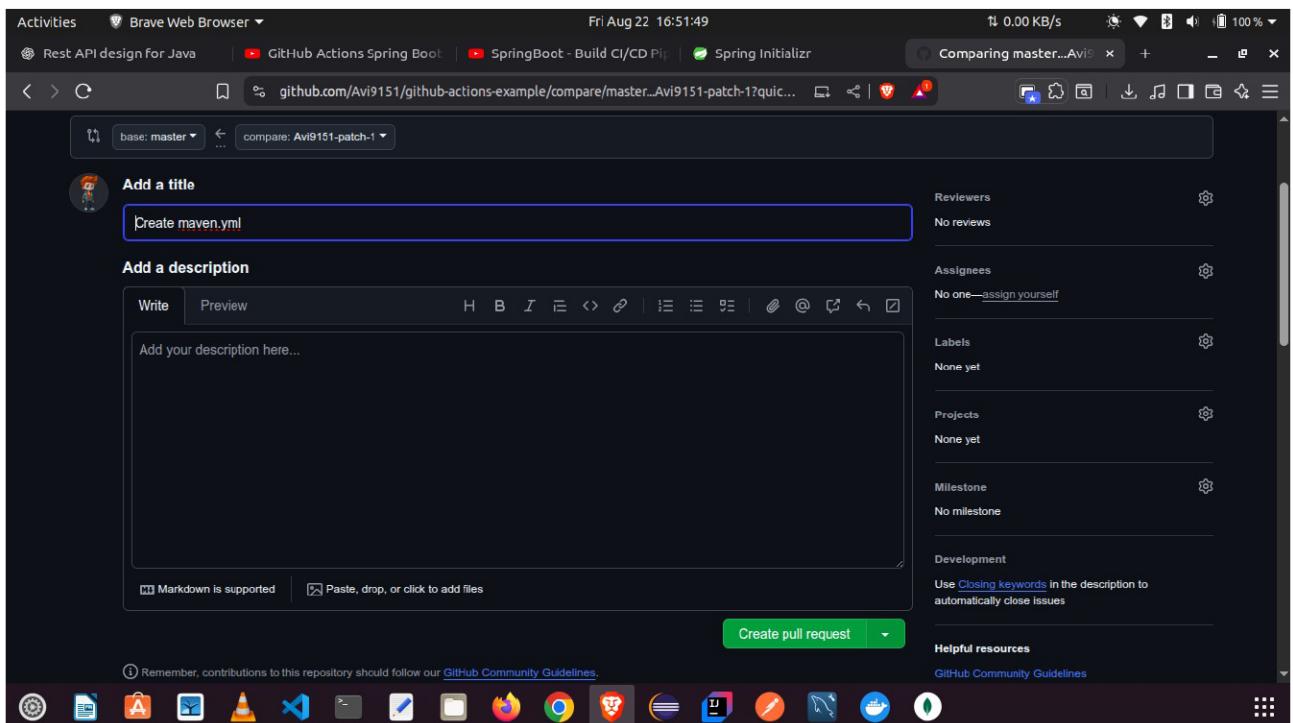
Save changes by clicking **Commit changes**.

- GitHub will suggest creating a new branch → allow it.

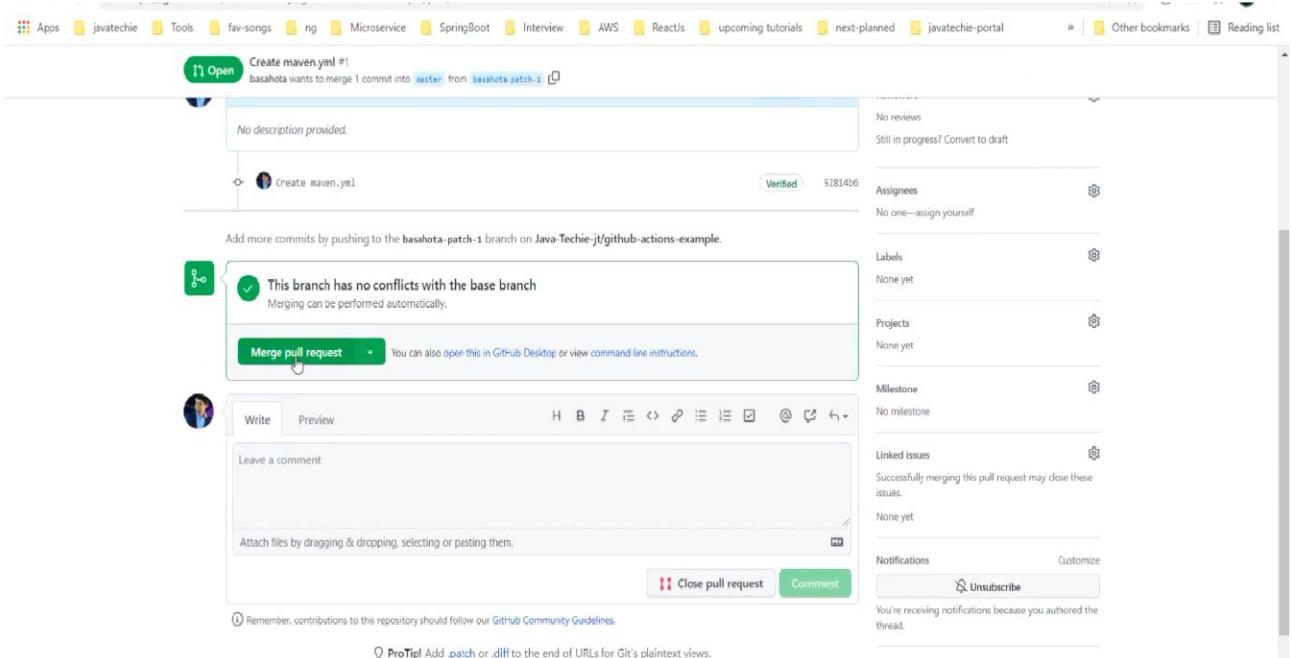


**Step 6:** Click on **Create a new branch for this commit and start a pull request** -> click on **Propose changes** button.

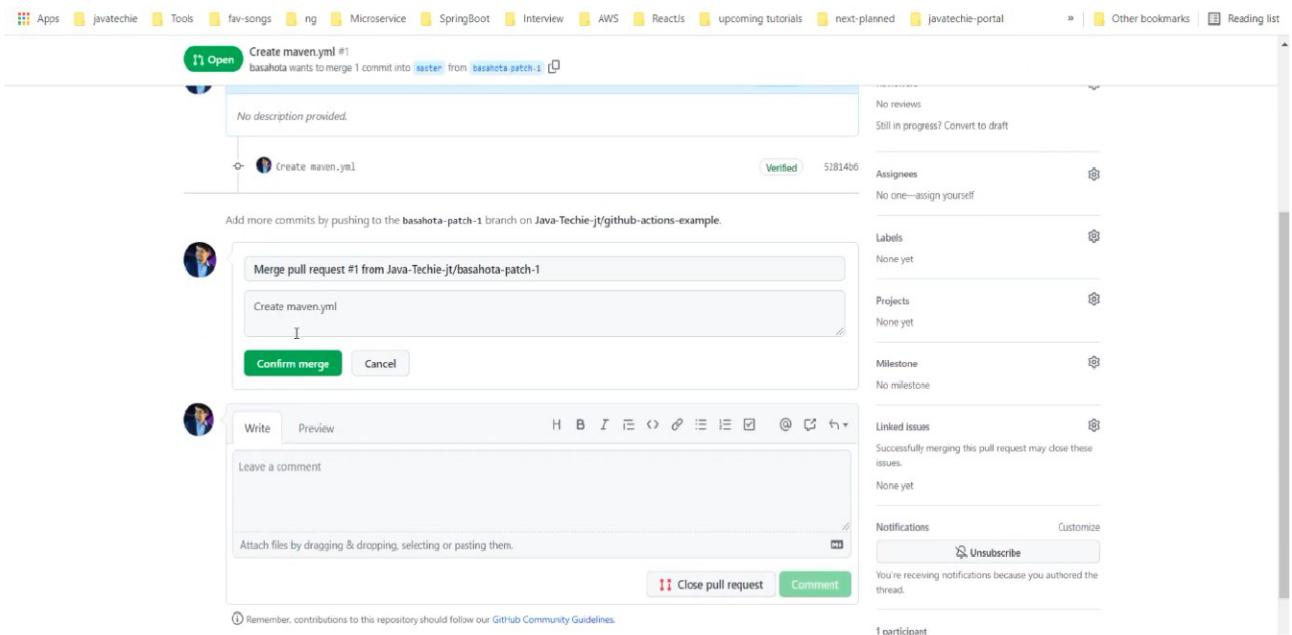
**Step 7:** Click on the **Create pull request** to merge these changes.



## Step 8: Click on Merge pull request



Then Click on the **Confirm merge** button.



**Step 9:** Then Open IntelliJ → git pull → You'll see .github/workflows/maven.yml file added.

The screenshot shows the IntelliJ IDEA interface with the 'GithubCicdActionsApplication.java' file open. A 'Pull to master' dialog is displayed over the code editor. The dialog contains the command 'git pull origin master'. There is a 'Pull' button and a 'Cancel' button.

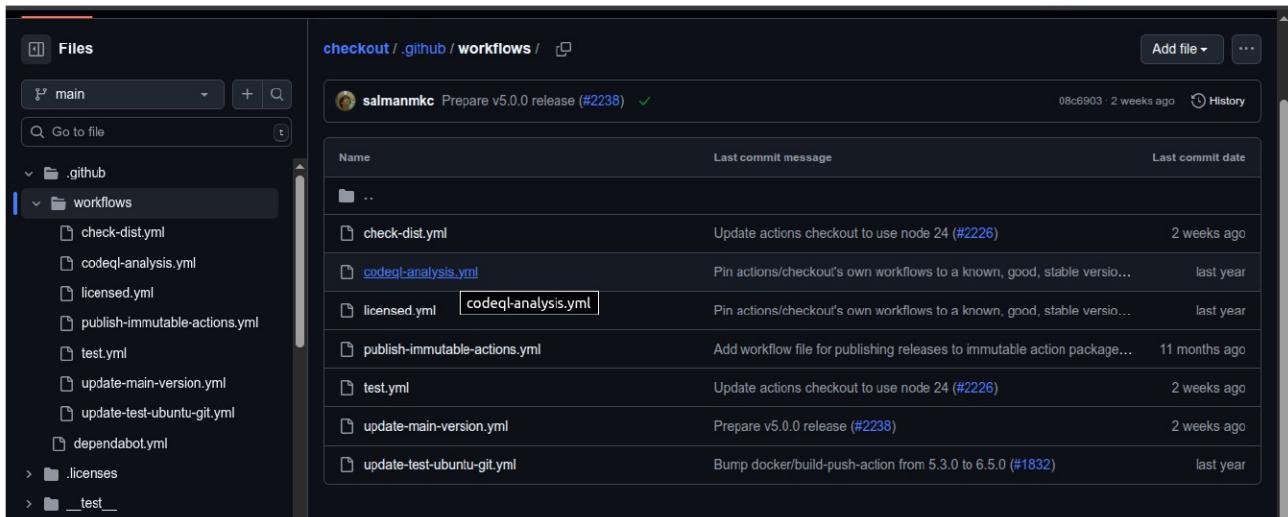
```
1 package com.avisoft.cicd;
2
3 > import ...
4
5 @SpringBootApplication
6 @RestController
7 public class GithubCicdActionsApplication {
8
9     @GetMapping("/welcome")
10    public String welcome() {
11        return "Hello, World!";
12    }
13
14    public static void main(String[] args) {
15        SpringApplication.run(GithubCicdActionsApplication.class, args);
16    }
17}
```

After pull open the **Github folder** -> **workflows** -> **maven.yml**

The screenshot shows the IntelliJ IDEA interface with the 'maven.yml' file open in the 'workflows' directory of the GitHub repository. The code defines a workflow for building a Java project using Maven.

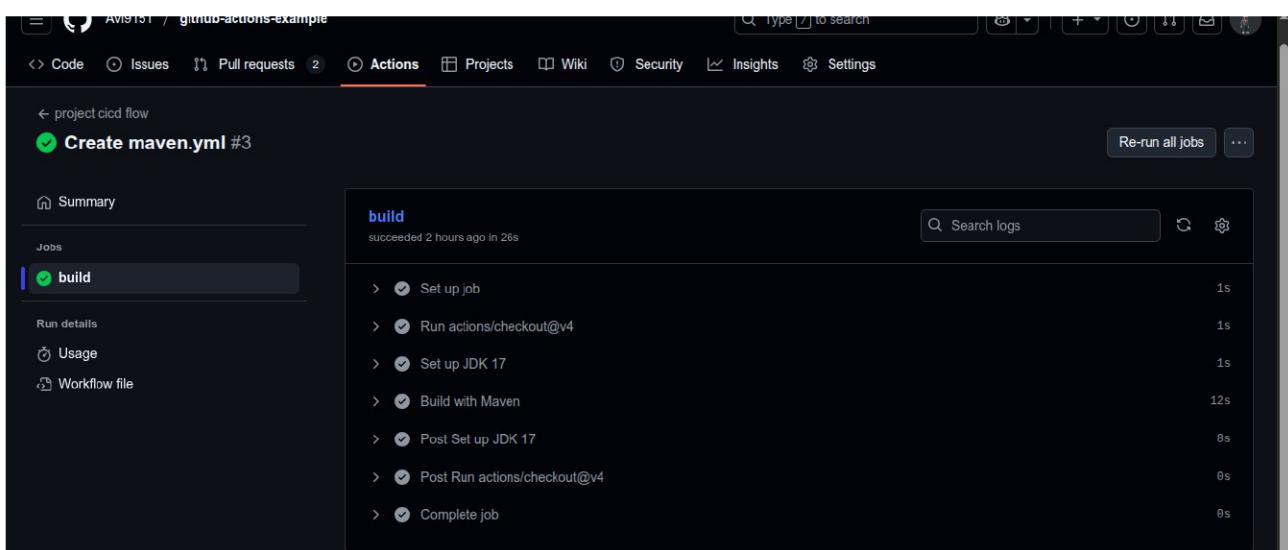
```
1 # This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the build time.
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-javascript#building-a-nodejs-project
3
4 # This workflow uses actions that are not certified by GitHub.
5 # They are provided by a third-party and are governed by separate terms of service, privacy policy, and support documentation.
6
7 name: project Cicd flow
8
9 on:
10   push:
11     branches: [ "master" ]
12   pull_request:
13     branches: [ "master" ]
14
15 permissions:
16   contents: read
17   dependencies: write
18
19 jobs:
20   build:
21     runs-on: ubuntu-latest
22
23     steps:
```

**Step 10:** Lets check the `actions/checkouts` so open url- <https://github.com/actions> and search.



The screenshot shows the GitHub Actions history for the repository. It lists several runs of the workflow file `codeql-analysis.yml`. The most recent run was by user `salmanmkc` for a release, with the commit message "Prepare v5.0.0 release (#2238)". Other runs are listed with their last commit messages and dates, such as "Update actions checkout to use node 24 (#2226)" (2 weeks ago) and "Pin actions/checkout's own workflows to a known, good, stable versio..." (last year).

**Step 11:** Go to the **Actions** and click on the Create **maven.yml**



The screenshot shows the GitHub Actions interface for a workflow named `Create maven.yml`. The `build` job is highlighted, showing its status as "succeeded 2 hours ago in 26s". The job log details the steps taken: "Set up job", "Run actions/checkout@v4", "Set up JDK 17", "Build with Maven", "Post Set up JDK 17", "Post Run actions/checkout@v4", and "Complete job". Each step is accompanied by a timestamp indicating its duration.

Activities Brave Web Browser ▾

Fri Aug 22 21:29:25

GitHub Actions Spring | SpringBoot - Build CI/CI | Spring Initializr | Rest API design for Java | Workflow runs · A | YouTube | + | - | X

github.com/Avi9151/github-actions-example/actions

Avi9151 / github-actions-example

Type to search

Code Issues Pull requests 2 Actions Projects Wiki Security Insights Settings

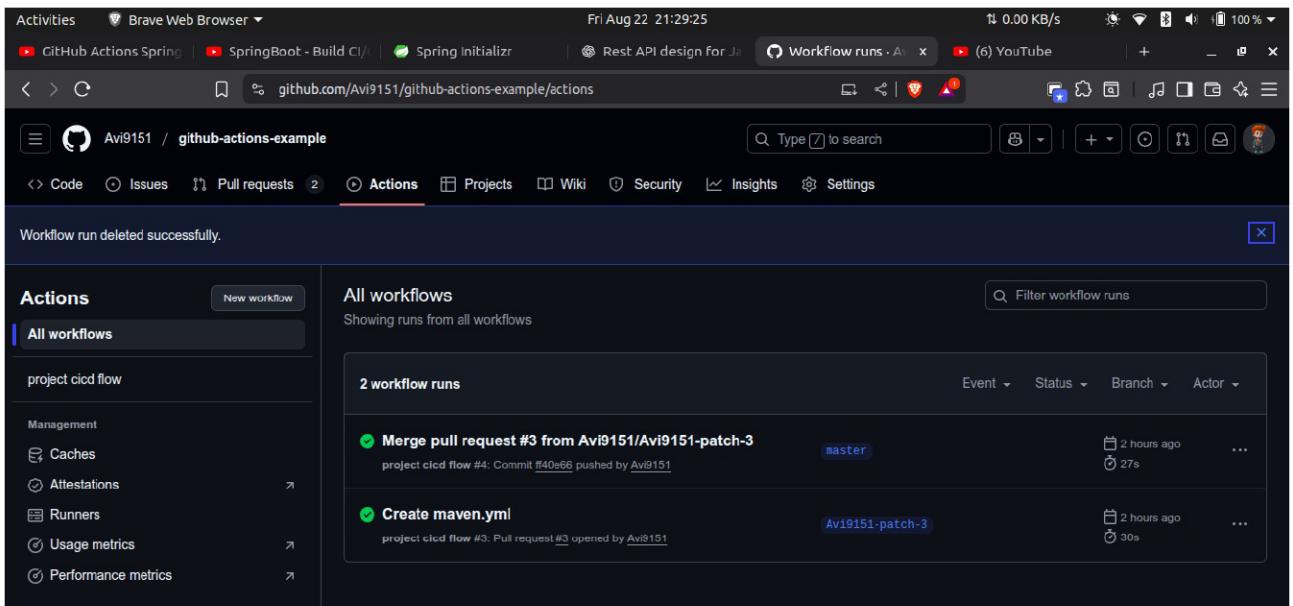
Workflow run deleted successfully.

All workflows Showing runs from all workflows

2 workflow runs

	Event	Status	Branch	Actor
Merge pull request #3 from Avi9151/Avi9151-patch-3	master	Success	2 hours ago	...
Create maven.yml	Avi9151-patch-3	Success	2 hours ago	...

Filter workflow runs



Avi9151 / github-actions-example

Type to search

Code Issues Pull requests 2 Actions Projects Wiki Security Insights Settings

← project cicd flow

>Create maven.yml #3

Re-run all jobs ...

Summary

Jobs build

Run details Usage Workflow file

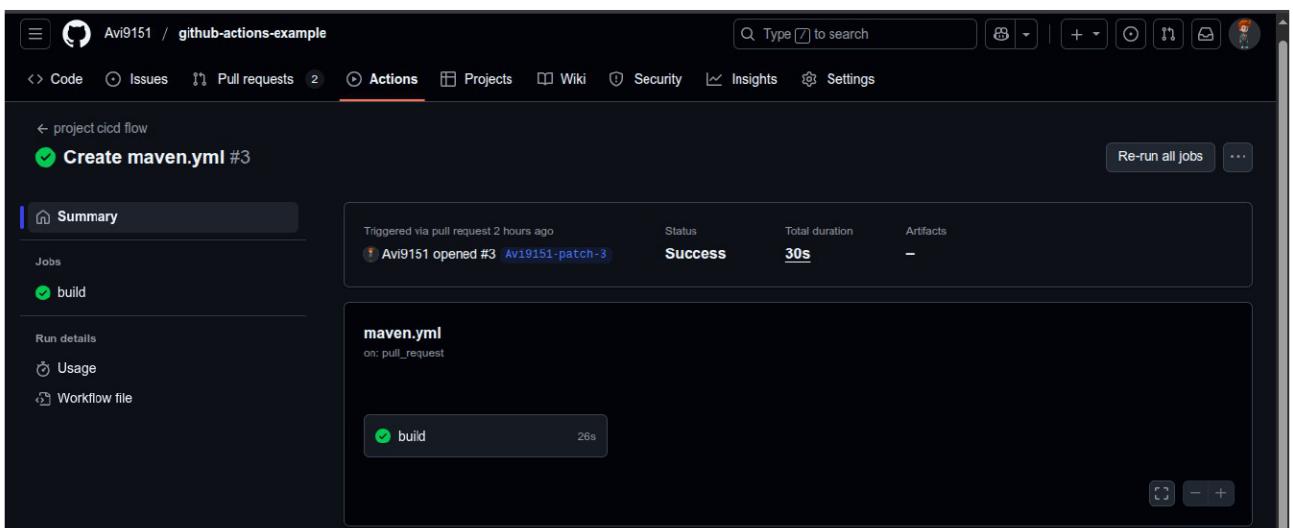
Triggered via pull request 2 hours ago

Avi9151 opened #3 Avi9151-patch-3

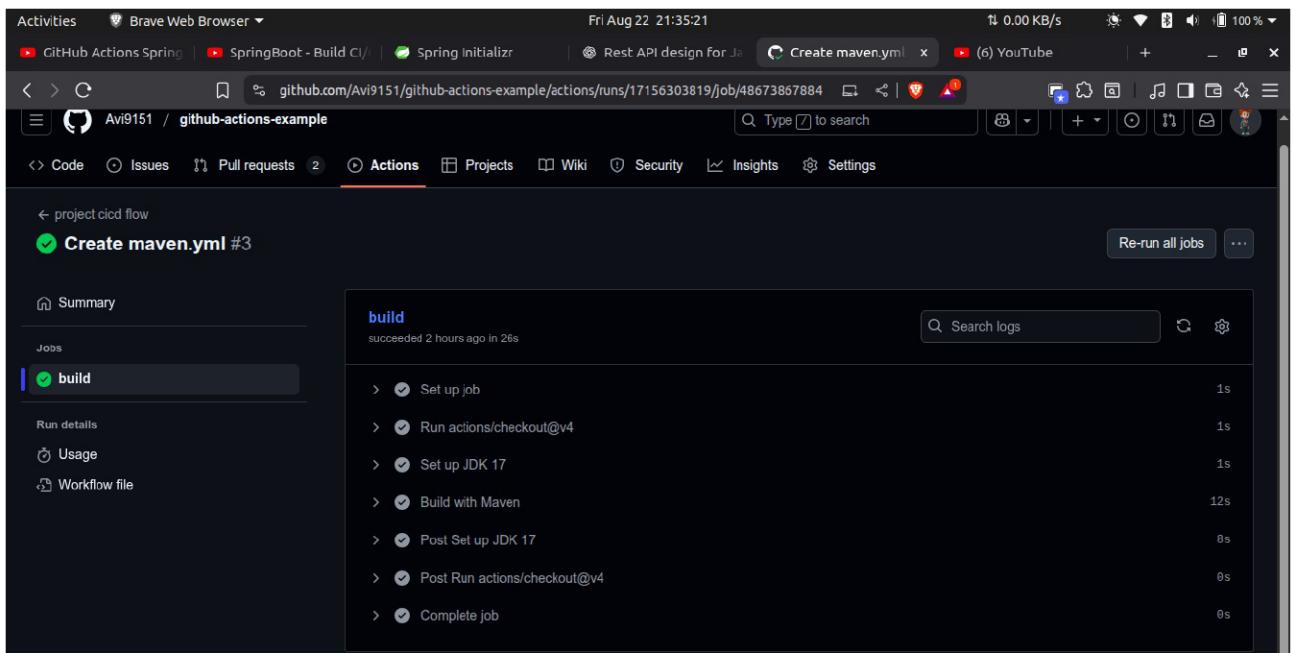
Status Success Total duration 30s Artifacts -

maven.yml on: pull\_request

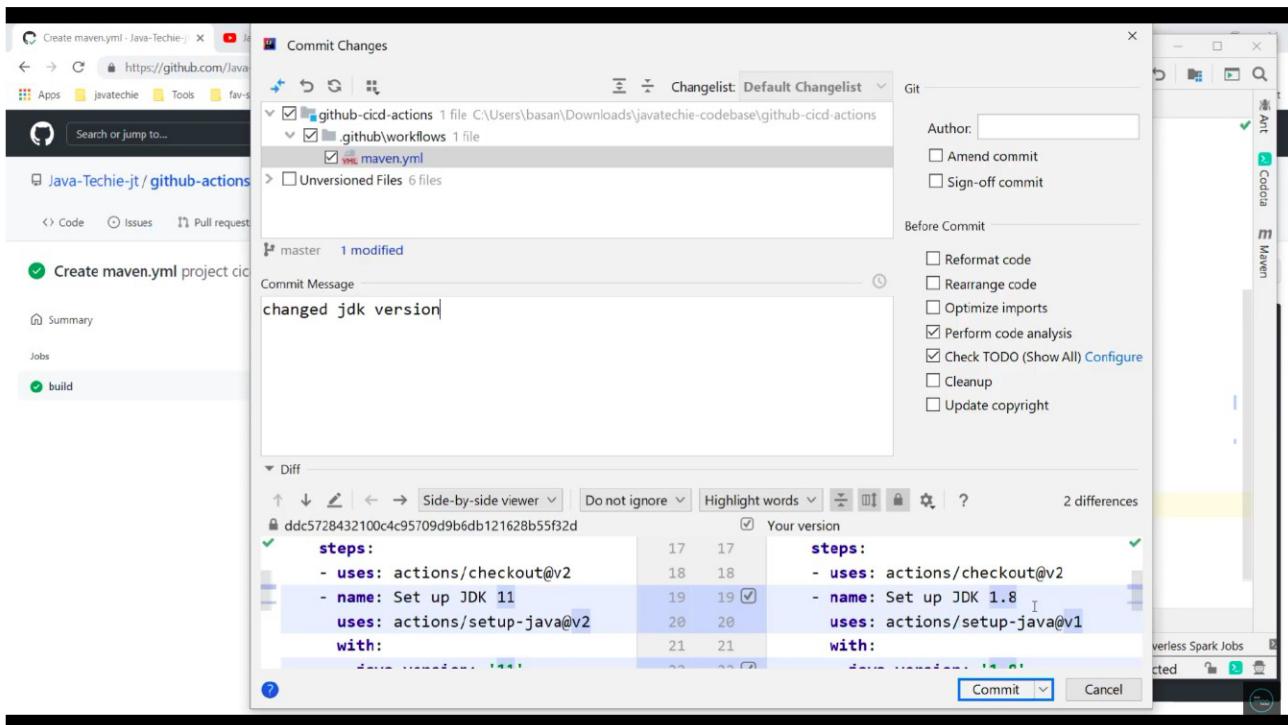
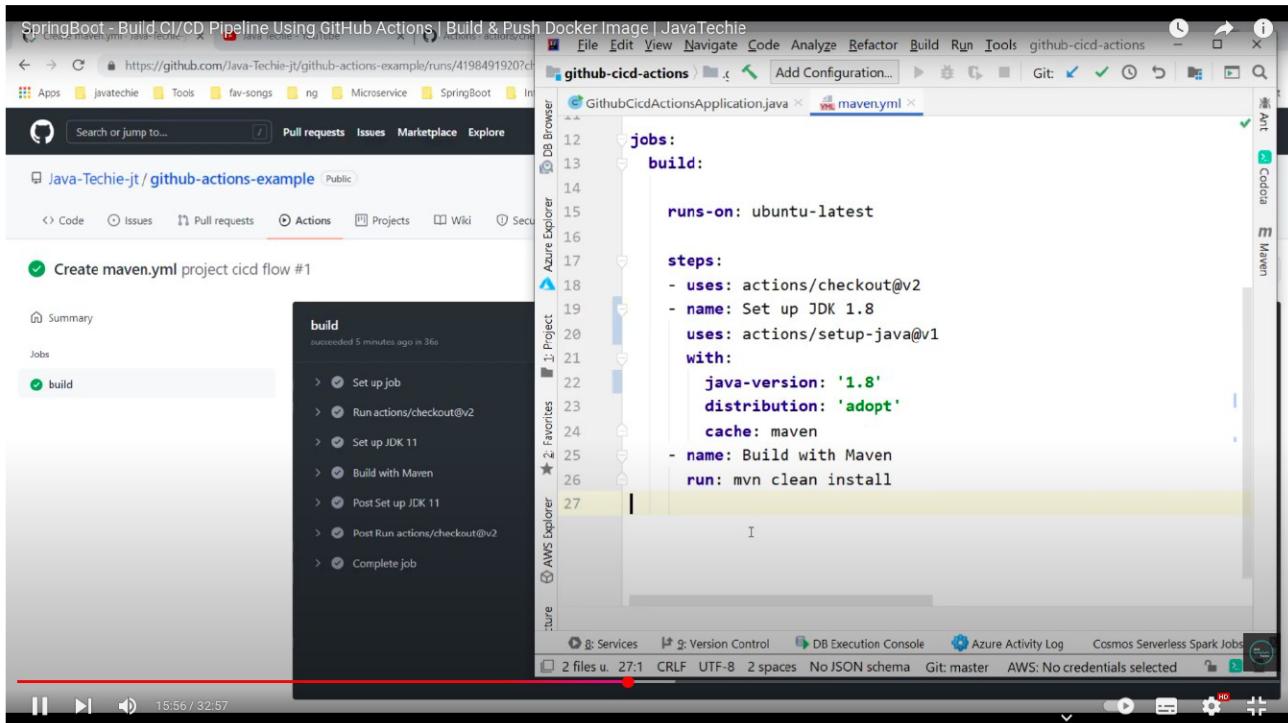
build 26s

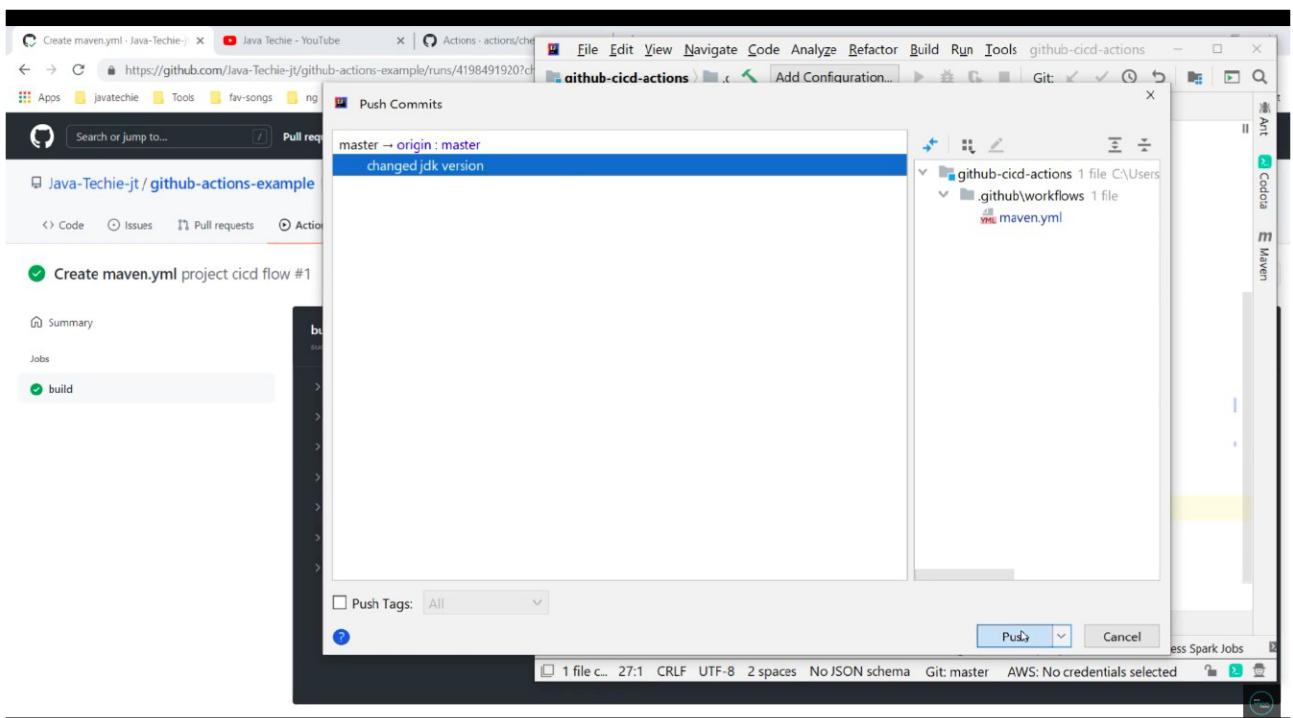
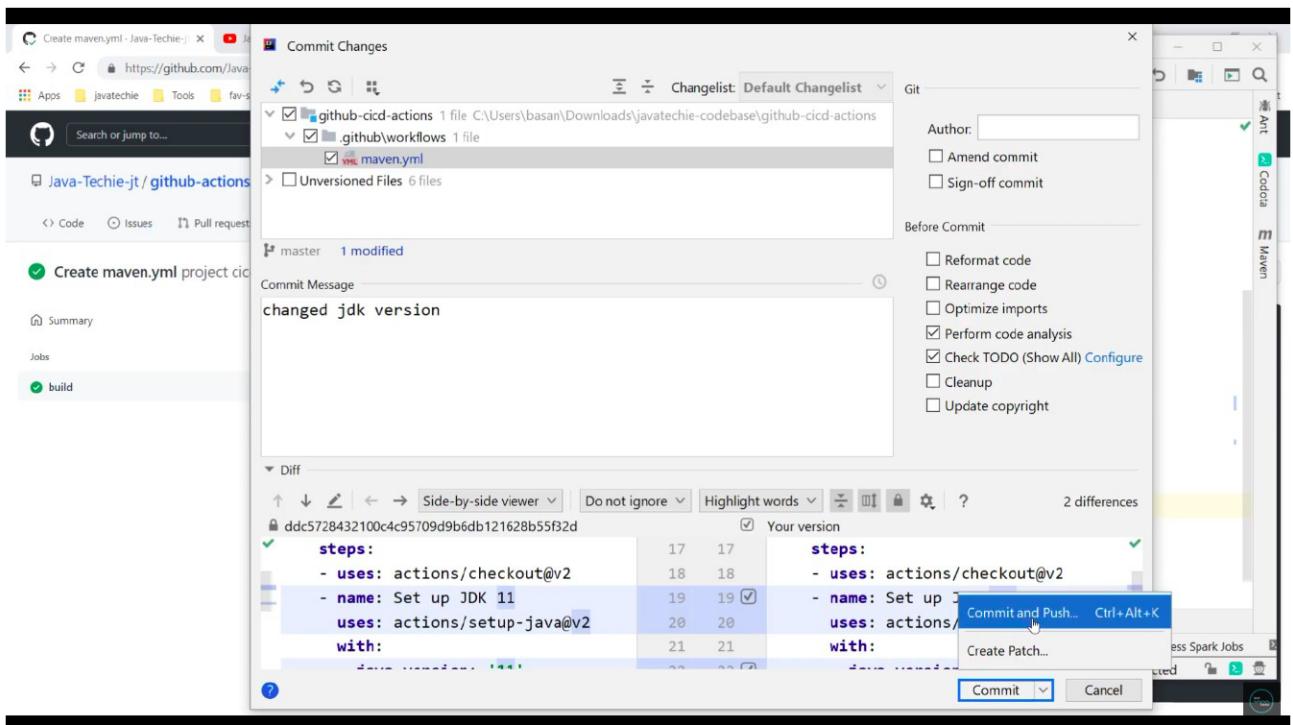


Click on **build**



**Note :** If you want to change jdk version and checkout@v(version).





## Click on changed jdk version

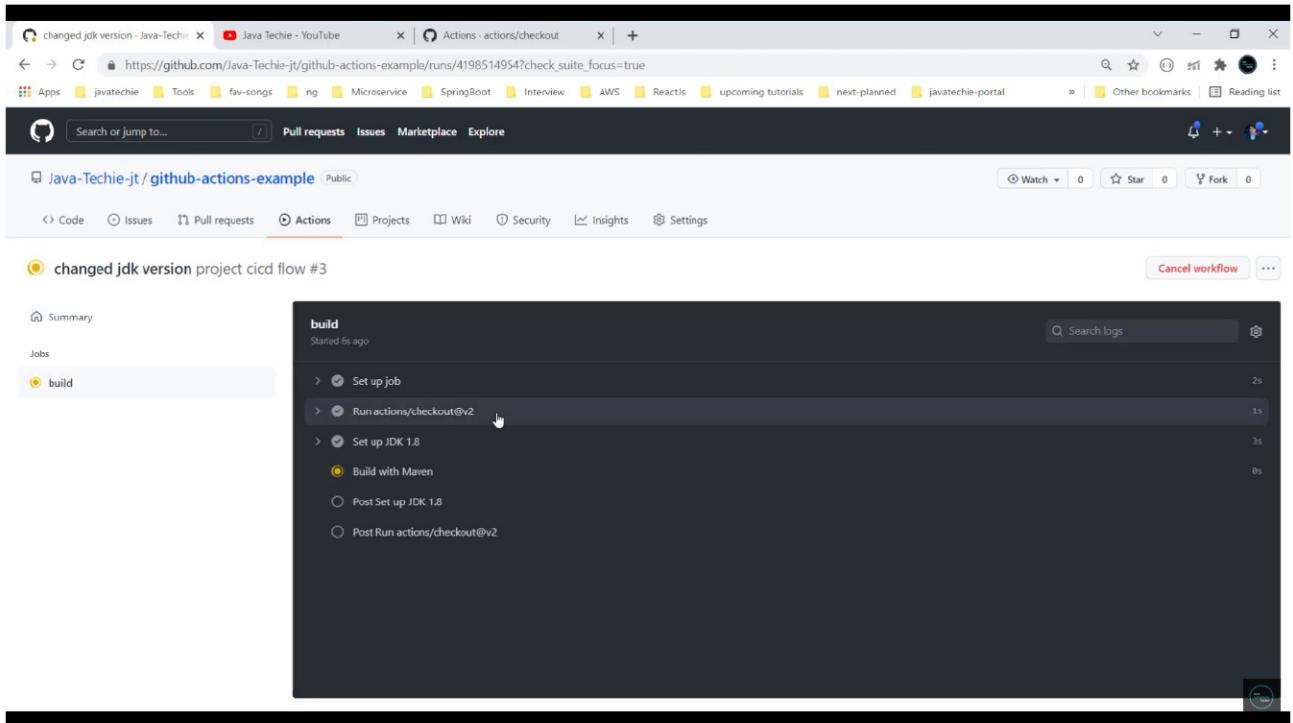
The screenshot shows the GitHub Actions interface. In the top navigation bar, the 'Actions' tab is selected. Below it, the 'Workflows' section is visible, with 'All workflows' selected. A modal window titled 'Tell us how to make GitHub Actions work better for you with three quick questions.' has a 'Give feedback' button. The main area displays 'All workflows' with the heading 'Showing runs from all workflows'. A search bar labeled 'Filter workflow runs' is present. Three workflow runs are listed:

- changed jdk version**: project cicd flow #3: Commit 14d0b7b pushed by basahota. Status: now (Queued). Started at now.
- Merge pull request #1 from Java-Techie-jt/basahota-pat...**: project cicd flow #2: Commit ddc5728 pushed by basahota. Status: 6 minutes ago (40s). Started at 6 minutes ago.
- Create maven.yaml**: project cicd flow #1: Pull request #1 opened by basahota. Status: 6 minutes ago (49s). Started at 6 minutes ago.

## Click on build

The screenshot shows a detailed view of a GitHub Actions workflow run. The URL in the address bar is [https://github.com/java-Techie-jt/github-actions-example/runs/4198514954?check\\_suite\\_focus=true](https://github.com/java-Techie-jt/github-actions-example/runs/4198514954?check_suite_focus=true). The workflow name is 'changed jdk version project cicd flow #3'. The 'Summary' tab is selected, showing a summary card for the 'basahota' job, which was triggered via push 10 seconds ago and is currently 'In progress'. The 'Jobs' tab is also visible. The main pane displays the 'build' job, which is associated with the 'maven.yaml' configuration file and triggered on a push event. The status of the 'build' step is 'In progress' with a duration of 2s.

You can see your all **jdk version** change, etc(if you want to change) it is doing setup.



## Now we tell to the Actions to build docker image

1. Login to your docker hub
2. Build a docker image
3. Push that image to docker hub

**Note :** “Github provide readymade Actions which can create docker image and push automatically without manually configuration.”

**Step 12:** Now search this - “build and push docker image using github” on browser.

A screenshot of a Brave Web Browser window. The address bar shows a Google search query: "build and push docker image using github". The search results page is displayed, with the top result being a link to the GitHub Marketplace for the "Docker Build & Push Action". The action's description states: "Builds a Docker image and pushes it to the private registry of your choosing. Supported Docker registries."

A screenshot of a Brave Web Browser window showing the GitHub Marketplace page for the "Docker Build & Push Action". The page includes sections for "About", "Supported Docker registries" (listing Docker Hub, GCR, ECR, and GitHub Docker Registry), "Features" (Auto-tagging with GitOps, BuildKit support, Multi-platform builds), and "Breaking changes". On the right side, there are sections for "Contributors" (18 contributors shown with small profile pictures) and "Resources" (links to open issues, pull requests, source code, and report abuse).

A screenshot of a Brave Web Browser window showing the GitHub Marketplace page for the "Docker Build & Push Action". It displays the workflow configuration for the action, including the following steps in a YAML file:

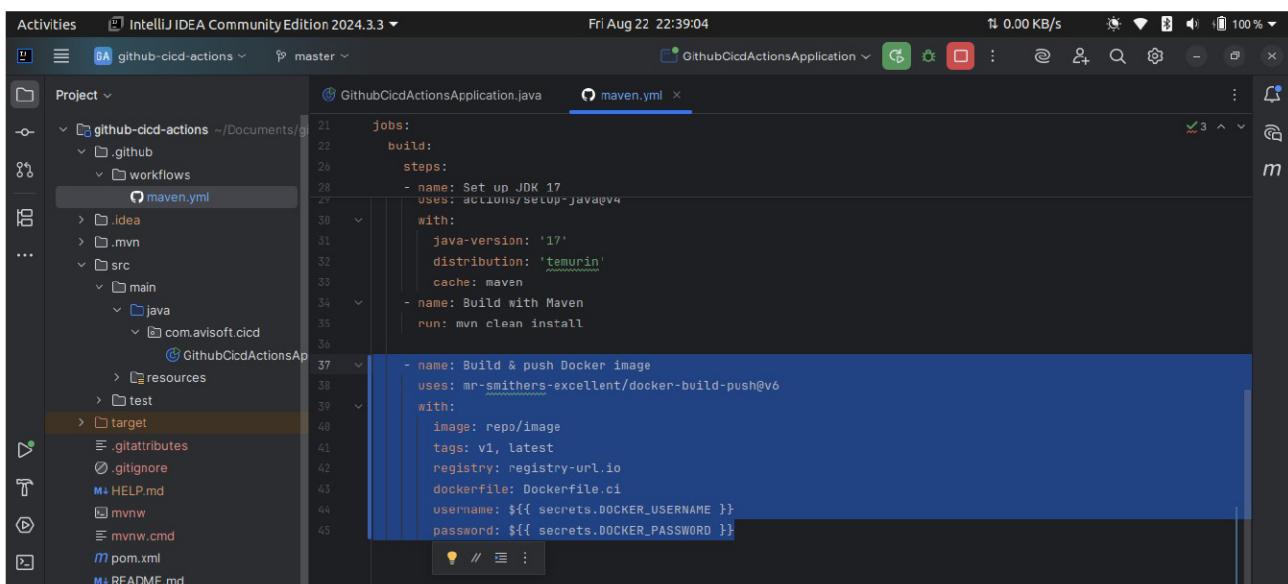
```
steps:
- uses: actions/checkout@v3
  name: Check out code

- uses: mr-smithers-excellent/docker-build-push@v6
  name: Build & push Docker image
  with:
    image: repo/image
    tags: v1, latest
    registry: registry-url.io
    dockerfile: Dockerfile.ci
    username: ${{ secrets.DOCKER_USERNAME }}
    password: ${{ secrets.DOCKER_PASSWORD }}
```

Below the steps, there is a table titled "Inputs" with columns for Name, Description, Required, and Type.

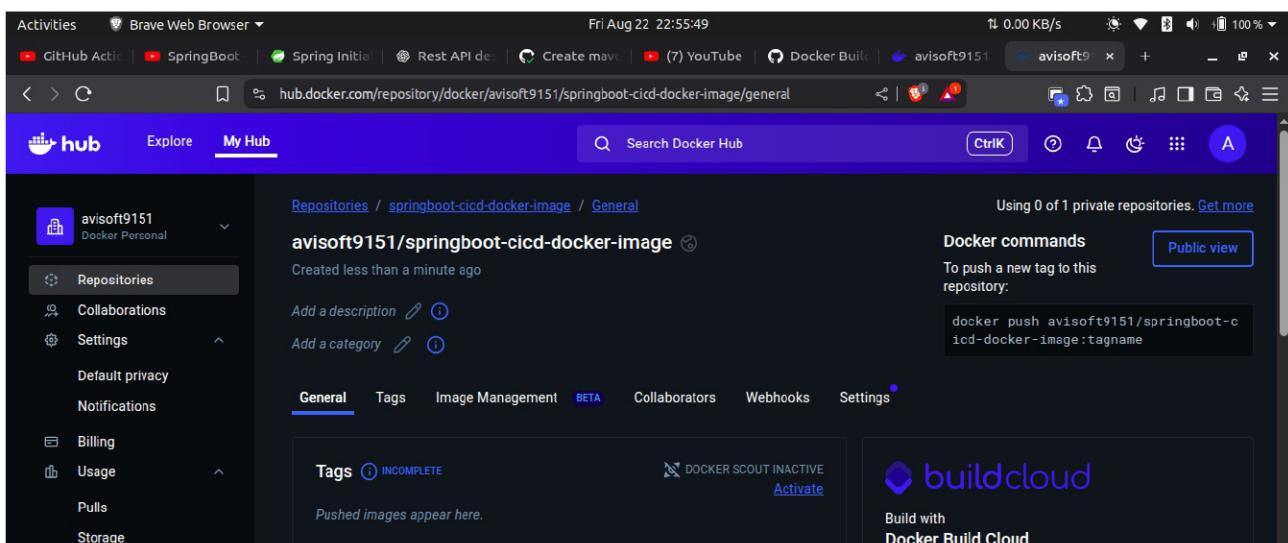
## Step 13: Add this yml code in maven.yml.

```
- name: Build & push Docker image
uses: mr-smithers-excellent/docker-build-push@v6
with:
  image: repo/image
  tags: v1, latest
  registry: registry-url.io
  dockerfile: Dockerfile
  username: ${{ secrets.DOCKER_USERNAME }}
  password: ${{ secrets.DOCKER_PASSWORD }}
```



## Step 14: Open **Docker hub** on browser and login.

## Step 15: Create new Repository.



## Step 16: Now update maven.yml code.

```
name: project cicd flow

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

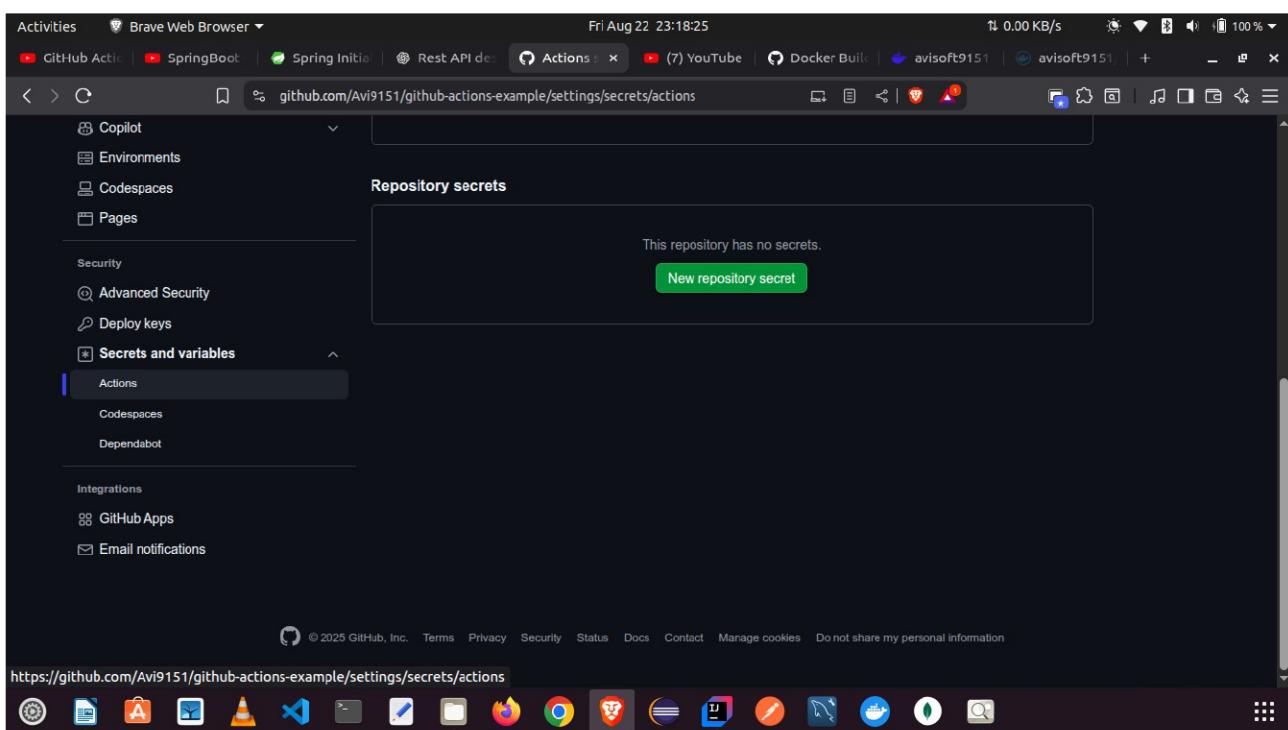
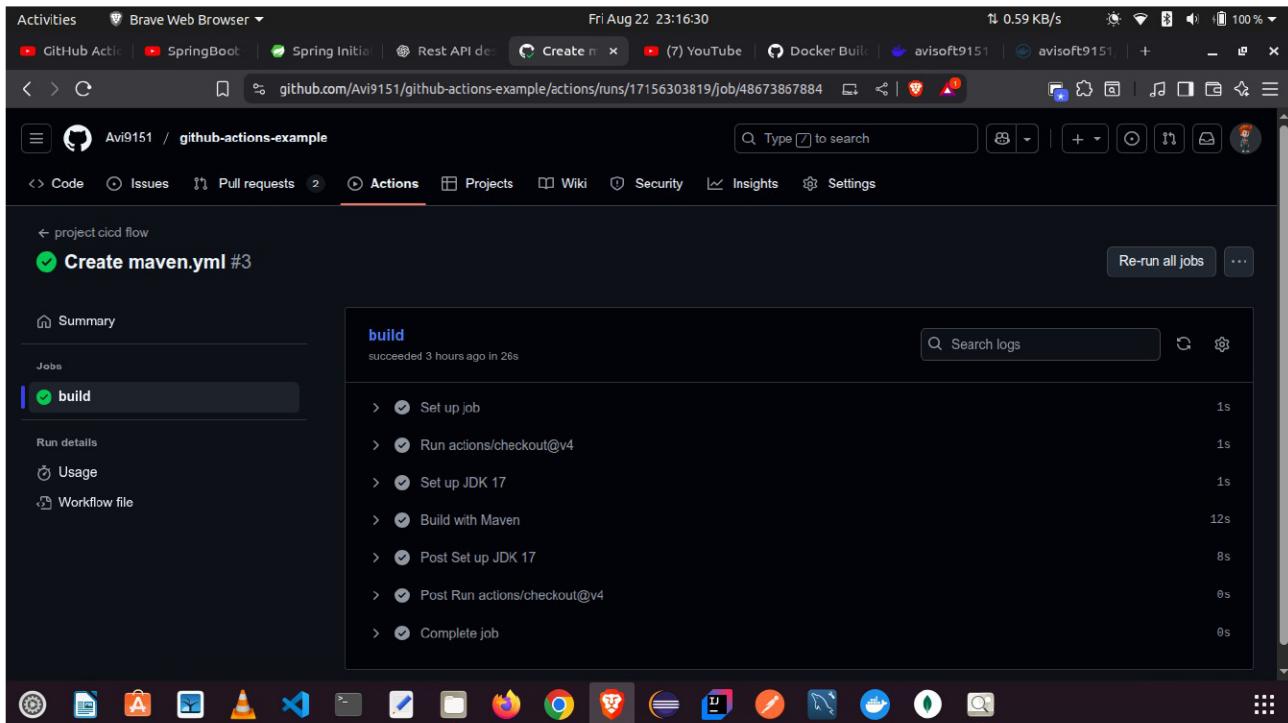
permissions:
  contents: write
  security-events: write

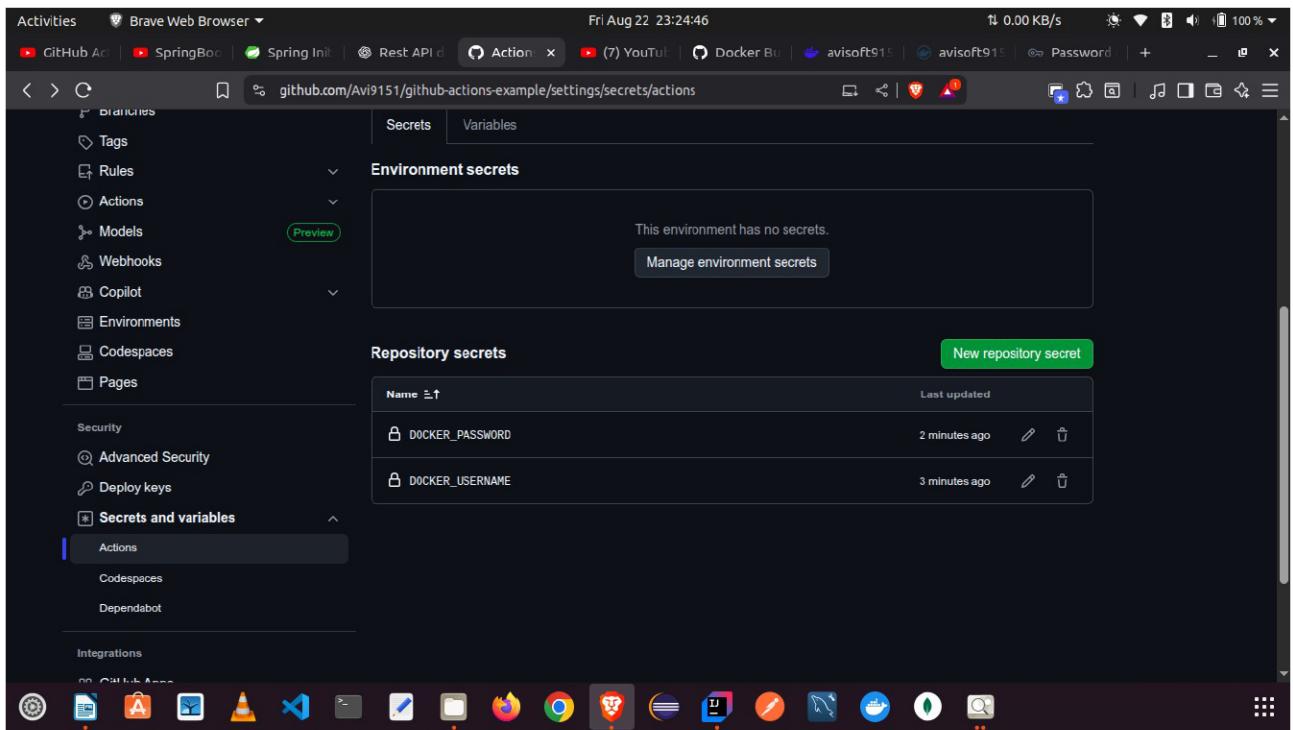
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4
      - name: Set up JDK 17
        uses: actions/setup-java@v4
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: maven
      - name: Build with Maven
        run: mvn clean install

# Update from here
- name: Build & push Docker image
  uses: mr-smithers-excellent/docker-build-push@v6
  with:
    image: avisoft9151/springboot-cicd-docker-image
    tags: latest
    registry: docker.io
    dockerfile: Dockerfile
    username: ${ secrets.DOCKER_USERNAME }
    password: ${ secrets.DOCKER_PASSWORD }
```

**Step 17:** Go to the github repo and click on [settings](#) -> [Secrets and Variables](#) -> [Actions](#) -> [New Repository Secrets](#) -> [Add Secrets](#) for [DOCKER\\_USERNAME](#) and [DOCKER\\_PASSWORD](#)





**Step 18:** Create docker file, click on [github-cicd-actions](#) -> create file **Dockerfile** and write code.

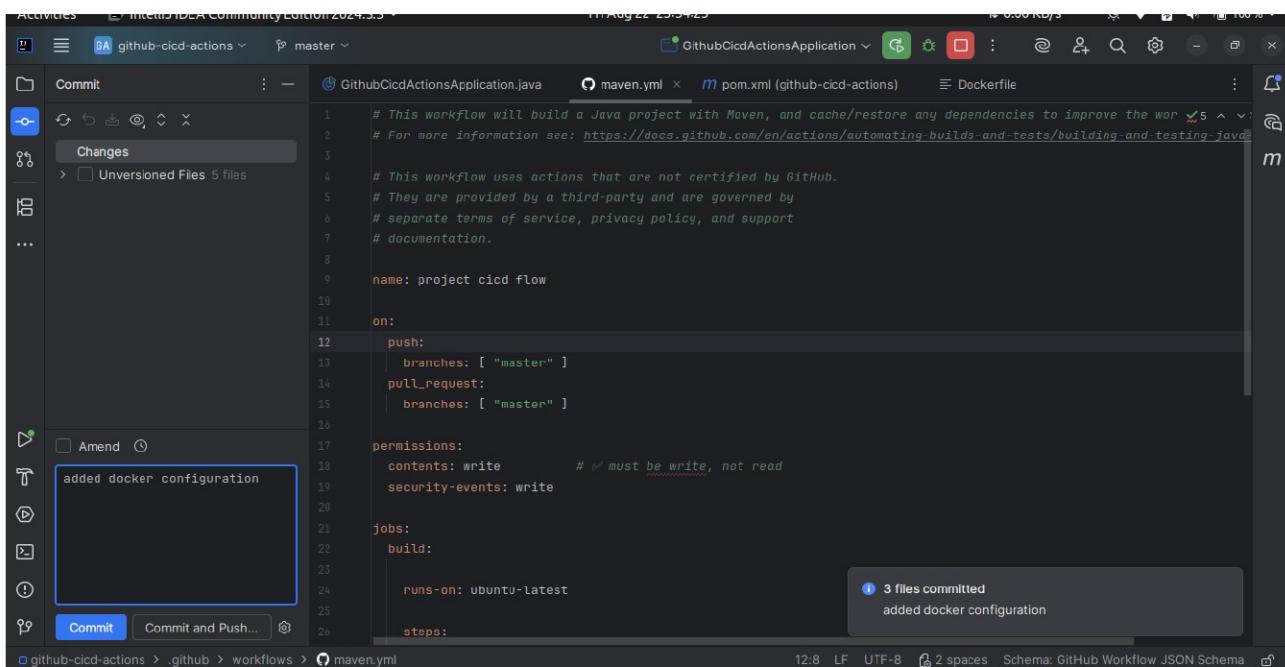
```
FROM openjdk:17
EXPOSE 8080
ADD target/springboot-cicd-docker-image.jar springboot-cicd-docker-image.jar
ENTRYPOINT ["java", "-jar", "/springboot-cicd-docker-image.jar"]
```

**Step 19:** Update **pom.xml** code and add this **<finalName>**.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
  <finalName>springboot-cicd-docker-image</finalName>
</build>
```

**Note :** It will create docker image and push to the docker hub.

**Step 20:** Now open IntelliJ Idea and git commit and push.



The screenshot shows the IntelliJ IDEA interface with the GitHub CICD Actions application open. The code editor displays a `maven.yml` file containing a GitHub workflow configuration. The workflow is named `project cicd flow` and triggers on pushes to the `master` branch. It includes permissions for writing contents and security events, and a job named `build` that runs on `ubuntu-latest`. A commit message `added docker configuration` is shown in the commit dialog, and the status bar indicates 3 files committed with the same message.

```
# This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the workflow. For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-javaworkflows

# This workflow uses actions that are not certified by GitHub. They are provided by a third-party and are governed by separate terms of service, privacy policy, and support documentation.

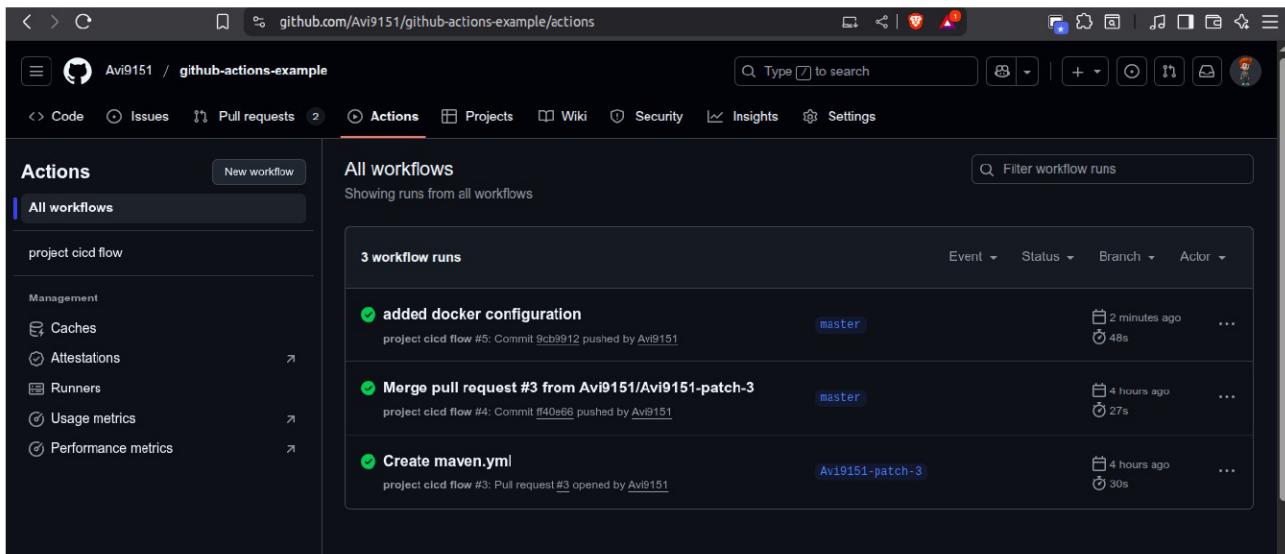
name: project cicd flow

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

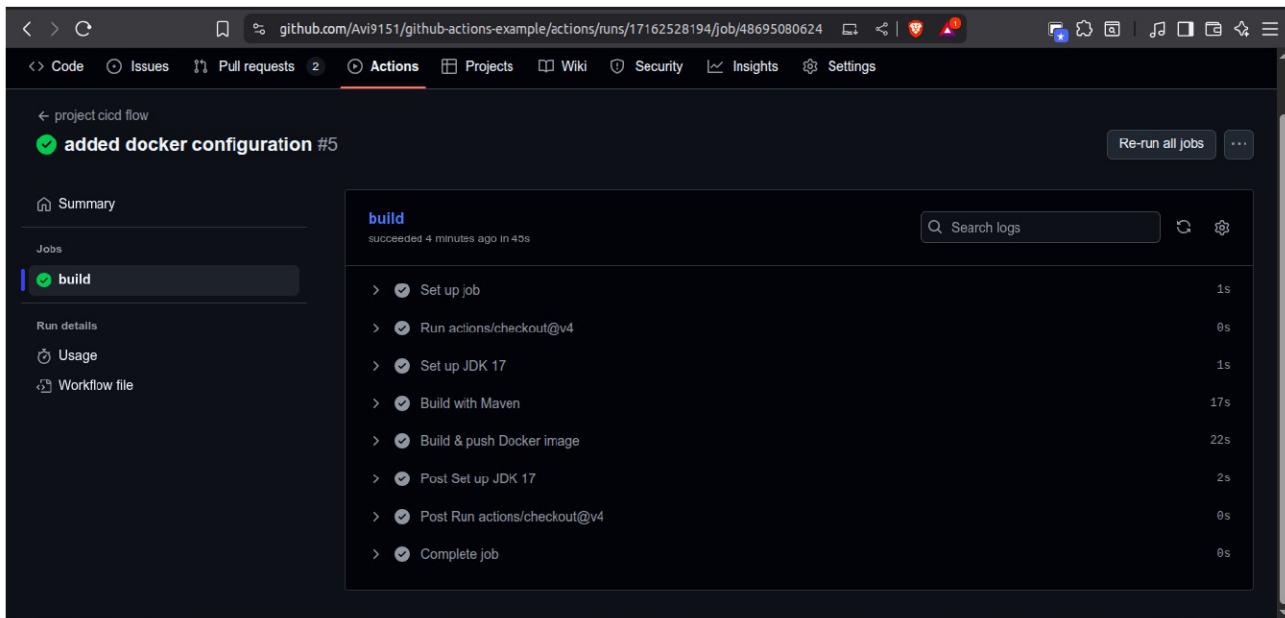
permissions:
  contents: write          # must be write, not read
  security-events: write

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
```

**Step 21:** Now go to the github and click on **Actions** and you can see started the build.

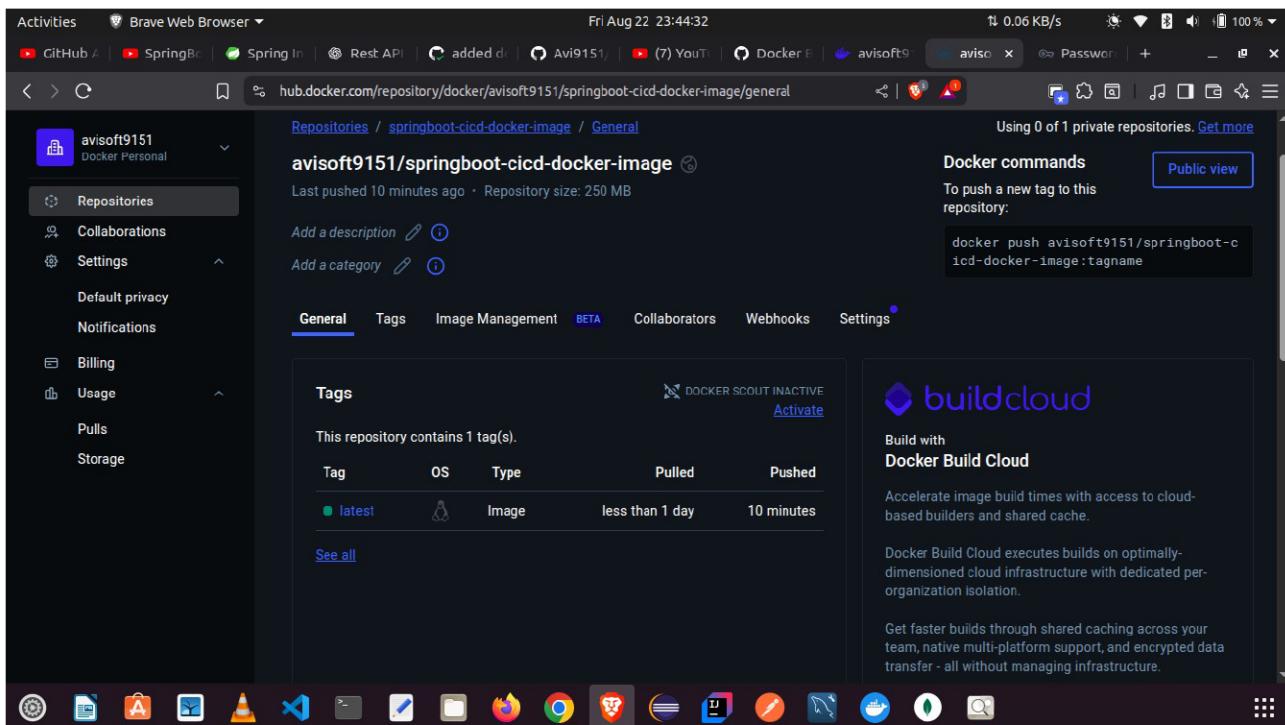


**Step 22:** Click on the added docker configuration -> build and you can see start build with maven



**Note :** Now you can see all build process completed and it should particular image to docker hub

**Step 23:** Now open the Docker hub on browser and refresh the page.



**Note :** Now you can see image got pushed on the docker hub.

And you can click on **latest** tag and see.

The screenshot shows the Docker Hub interface for the repository `avisoft9151/springboot-ci-cd-docker-image`. The 'Tags' section is selected, showing the `latest` tag. The tag details are as follows:

- MANIFEST DIGEST: `sha256:a77bf0b12364462d81fe397855f6caadbb5f82d26de26b976ff8437e825bea3`
- OS/ARCH: `linux/amd64`
- COMPRESSED SIZE: `249.95 MB`
- LAST PUSHED: `about 10 hours ago by avisoft9151`
- TYPE: `Image`
- MANIFEST DIGEST: `sha256:a77bf0b1...`

The 'Image Layers' tab is active, displaying the following layer details:

Layer	Command	Size
1 ADD file ... in /	<code>ADD file:9893213a9ea238f53ac68d87a3cf2f05d86763688392e5dd6a2c</code>	40.16 MB
2 CMD ["/bin/bash"]		0 B
3 /bin/sh -c set -eux; microdnf		12.9 MB
4 ENV JAVA_HOME=/usr/java/openjdk-17		0 B
5 ENV PATH=/usr/java/openjdk-17/bin:/usr/local/sbin:/...		0 B

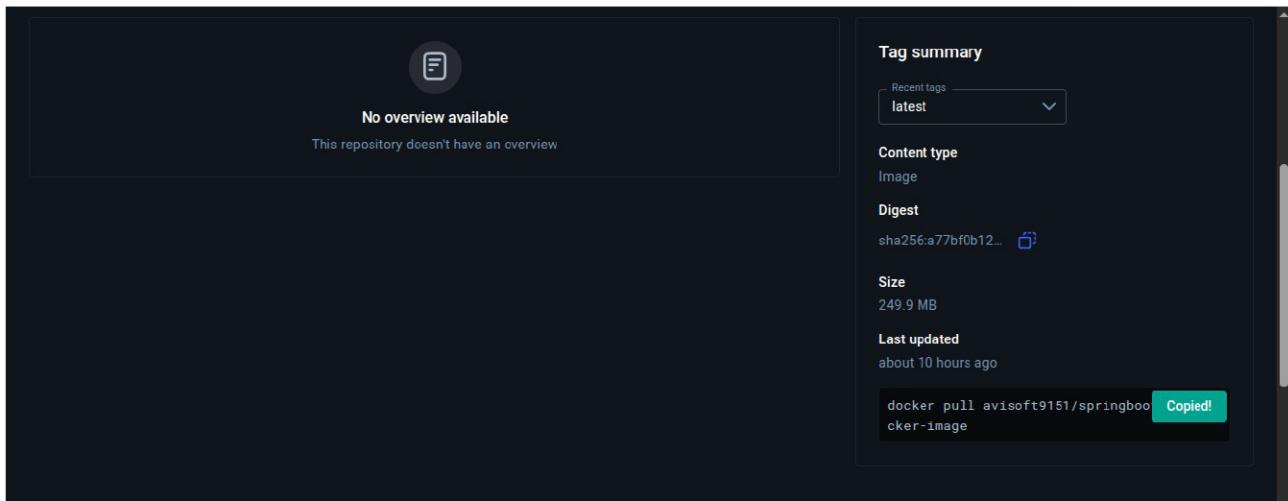
**Step 24:** We can pull this image and run on locally then click on **Public view**.

The screenshot shows the Docker Hub interface for the repository `avisoft9151/springboot-ci-cd-docker-image` on the 'General' tab. The tag `latest` is selected. The page displays the following information:

- Last pushed: 10 minutes ago
- Repository size: 250 MB
- Docker commands:
  - `docker push avisoft9151/springboot-ci-cd-docker-image:tagname`
- Public view button
- Tags table:

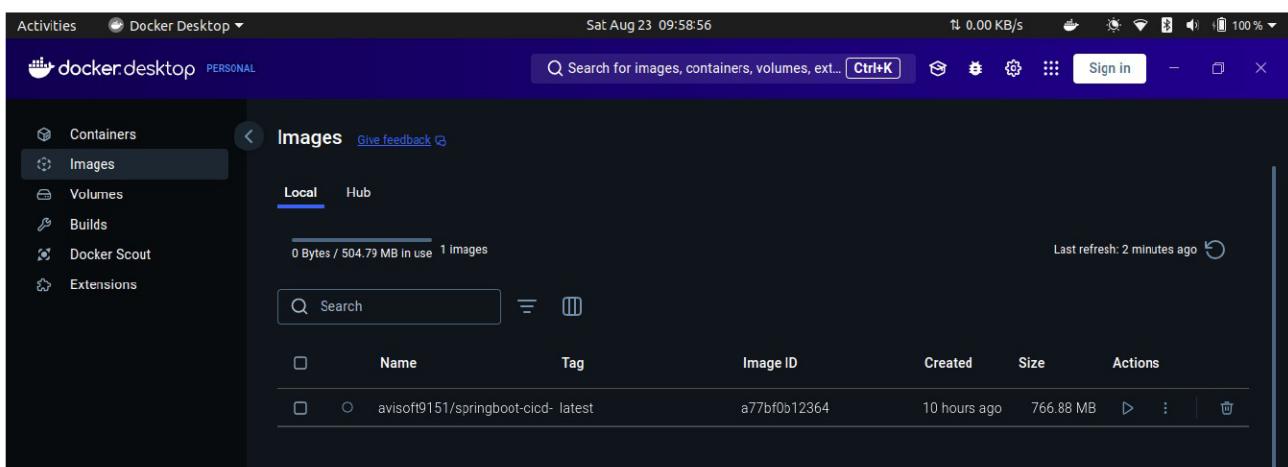
Tag	OS	Type	Pulled	Pushed
latest	Linux	Image	less than 1 day	10 minutes
- Buildcloud integration section:
  - Build with Docker Build Cloud
  - Accelerate image build times with access to cloud-based builders and shared cache.
  - Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.
  - Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

Copy this command and run it on terminal for local run.



```
Activities Terminal ▾ Sat Aug 23 09:21:39 avisoft@avisoft-HP-Notebook:~  
docker pull avisoft9151/springboot-cicd-docker-image  
Using default tag: latest  
latest: Pulling from avisoft9151/springboot-cicd-docker-image  
3ba980f2cc8a: Pull complete  
de849fcfb0e: Pull complete  
a7203ca35e75: Pull complete  
8fd1cdd58553: Pull complete  
Digest: sha256:a77bf0b12364462d81fe397855f6caadb5f82d26de26b976ff8437c825baa3  
Status: Downloaded newer image for avisoft9151/springboot-cicd-docker-image:latest  
docker.io/avisoft9151/springboot-cicd-docker-image:latest  
avisoft@avisoft-HP-Notebook:~$ docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
avisoft9151/springboot-cicd-docker-image latest a0ebce86e18c 10 hours ago 492MB  
avisoft@avisoft-HP-Notebook:~$
```

- You can see downloaded the docker image.
- And you observe the docker image add after run - [docker images](#) command run on terminal.
- Now open the Docker Desktop and you can see this image added.



Now we run this image manually through command and run this command on terminal.

**Command -** [docker run -p 8080:8080 avisoft9151/springboot-cicd-docker-image](#)

- We can see Application started at port number 8080 and docker container.
  - Now go to the browser and hit this endpoints -  
<http://localhost:8080/welcome>



## Final Outcome

- **Fully automated CI/CD pipeline:**
  - Code push → GitHub Actions → Build → Docker Image → Push to Docker Hub.
- No manual build required.
- Can deploy this image anywhere (local, Kubernetes, cloud).