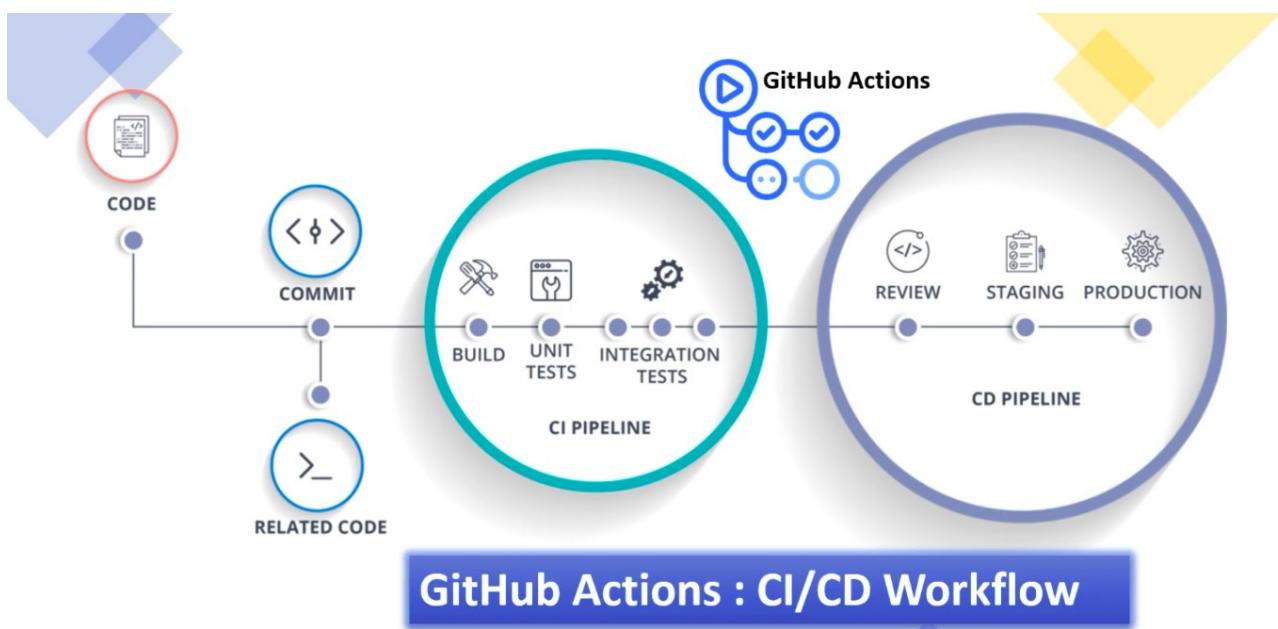


# SpringBoot - Build CI/CD Pipeline Using GitHub Actions

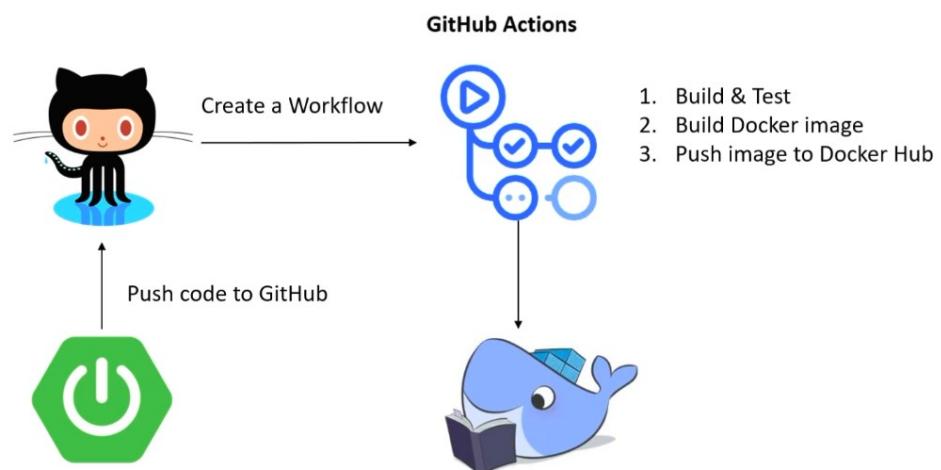
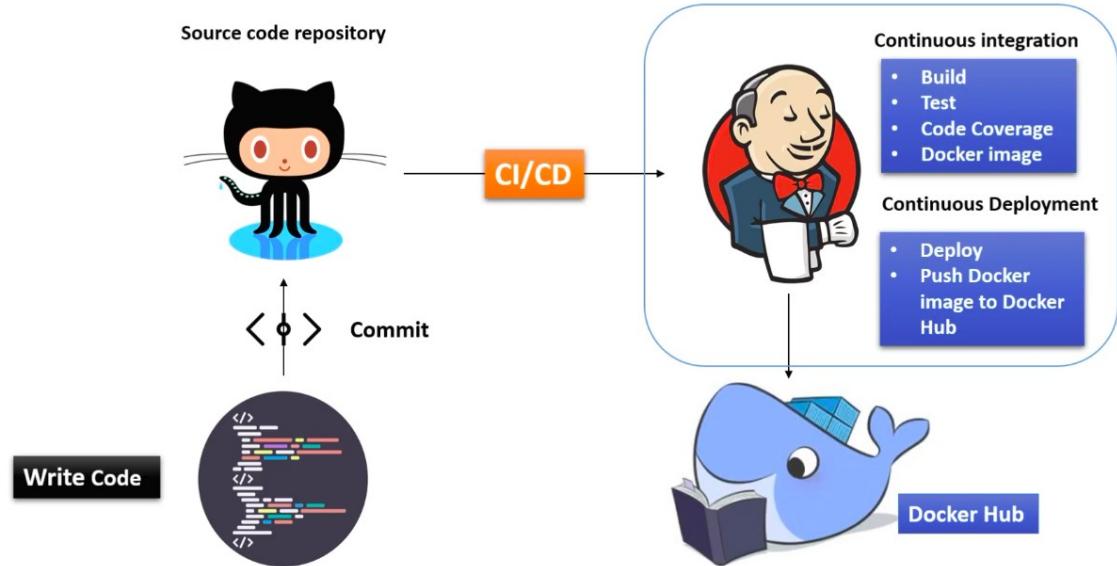
## What is CI/CD?

- **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:

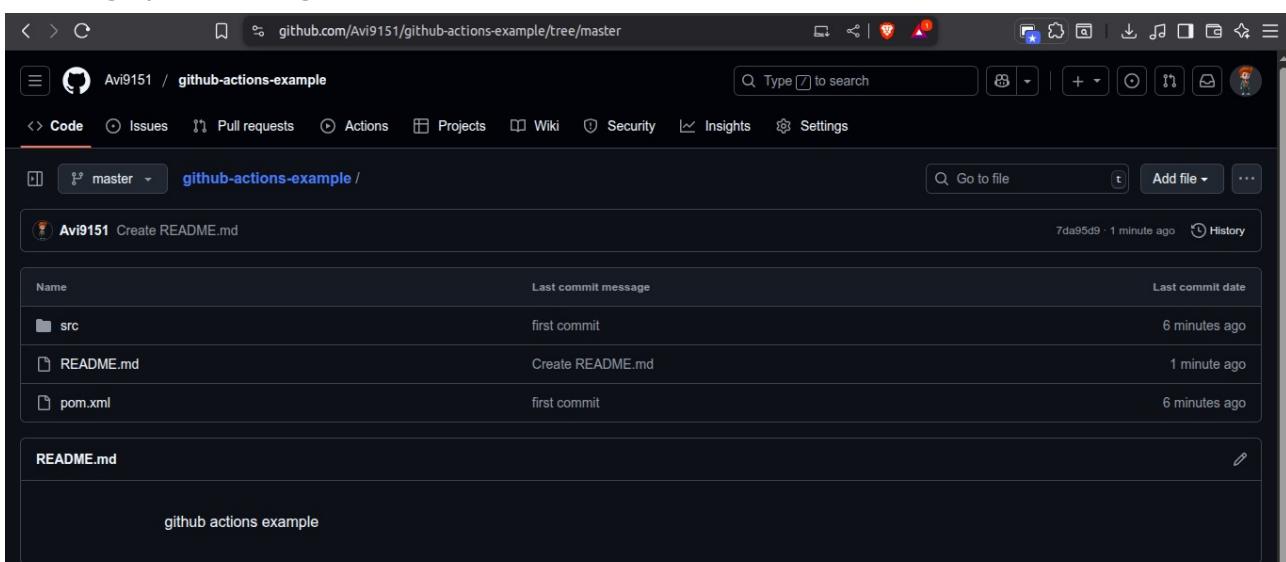
```
git add .
```

- Commit:

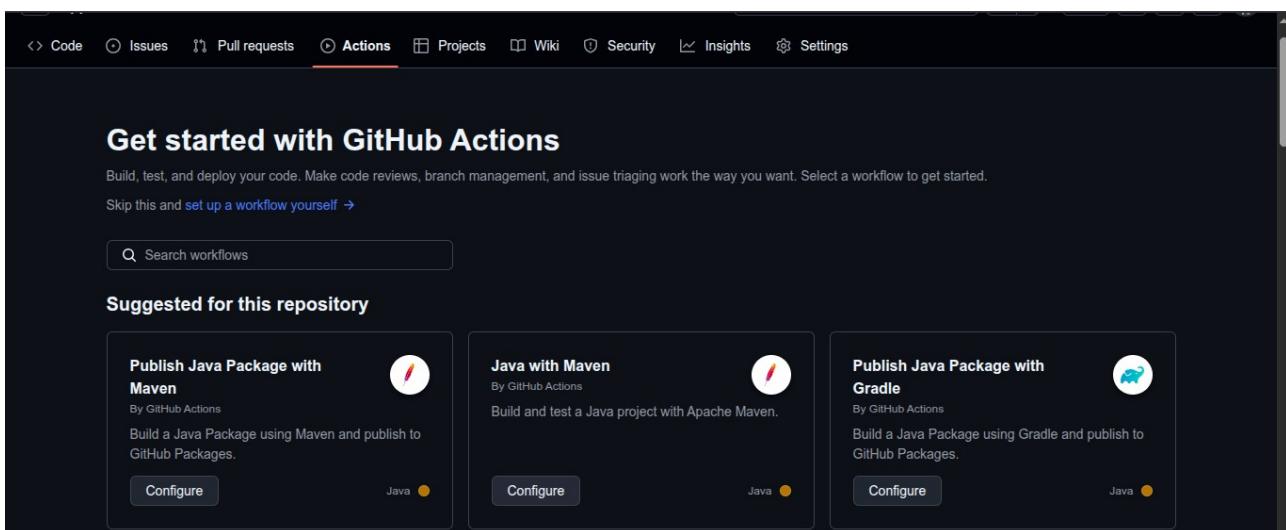
```
git commit -m "Initial commit"
```

- Push to GitHub repository:

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



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



A screenshot of a web browser window titled "Brave Web Browser". The address bar shows the URL [github.com/Avi9151/github-actions-example/new/master?filename=.github%2FWorkflow%2FJenkinsfile](https://github.com/Avi9151/github-actions-example/new/master?filename=.github%2FWorkflow%2FJenkinsfile). The main content area displays a GitHub Actions workflow 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     runs-on: ubuntu-latest
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
```

The right side of the screen shows the GitHub Marketplace sidebar with sections for "Featured Actions" and "Documentation".

A screenshot of a web browser window titled "Brave Web Browser". The address bar shows the URL [github.com/Avi9151/github-actions-example/new/master?filename=.github%2FWorkflow%2FJenkinsfile](https://github.com/Avi9151/github-actions-example/new/master?filename=.github%2FWorkflow%2FJenkinsfile). The main content area displays a GitHub Actions workflow file with additional steps added:

```
16|
17 jobs:
18   build:
19     runs-on: ubuntu-latest
20
21   steps:
22     - uses: actions/checkout@v4
23     - name: Set up JDK 17
24     - uses: actions/setup-java@v4
25     - with:
26       java-version: '17'
27       distribution: 'temurin'
28       cache: maven
29     - name: Build with Maven
30     - run: mvn -B package --file pom.xml
31
32   # Optional: Uploads the full dependency graph to GitHub to improve the quality of Dependabot alerts this repository
33   - name: Update dependency graph
34   - uses: advanced-security/maven-dependency-submission-action@57ie99aab1055c2e71a1e2309b9691de18d6b7d6
35
36
```

The right side of the screen shows the GitHub Marketplace sidebar with sections for "Featured Actions" and "Documentation".

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

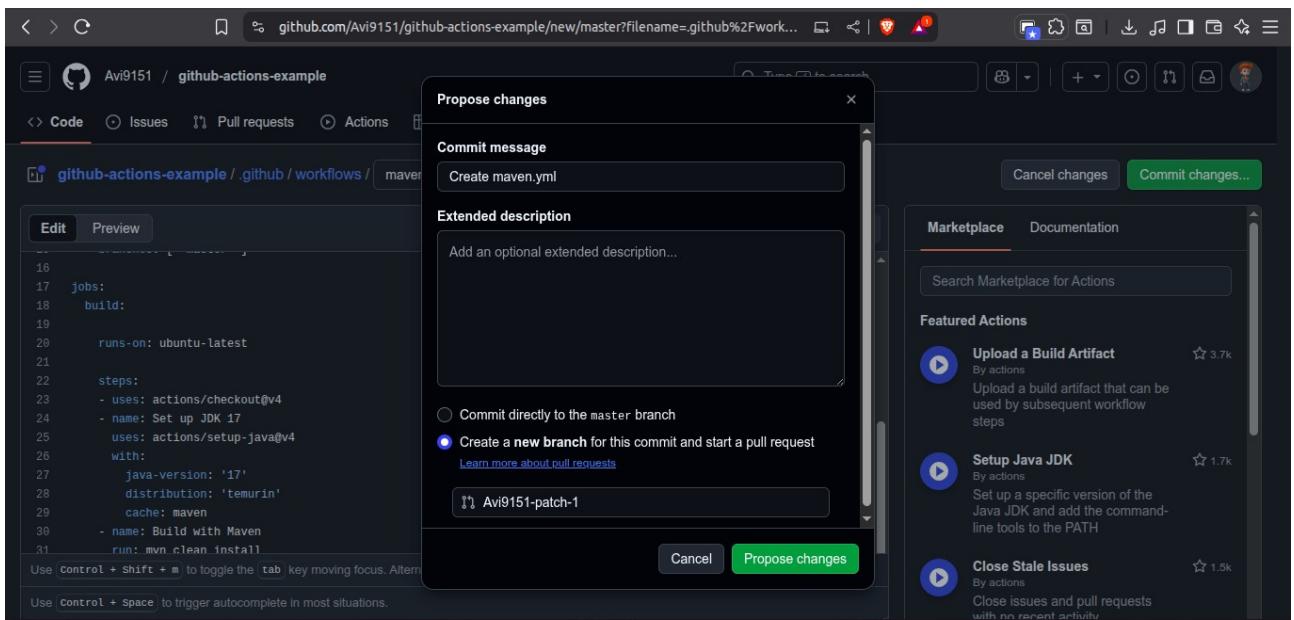
`name`: project Cicd flow

`run`: mvn clean install

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

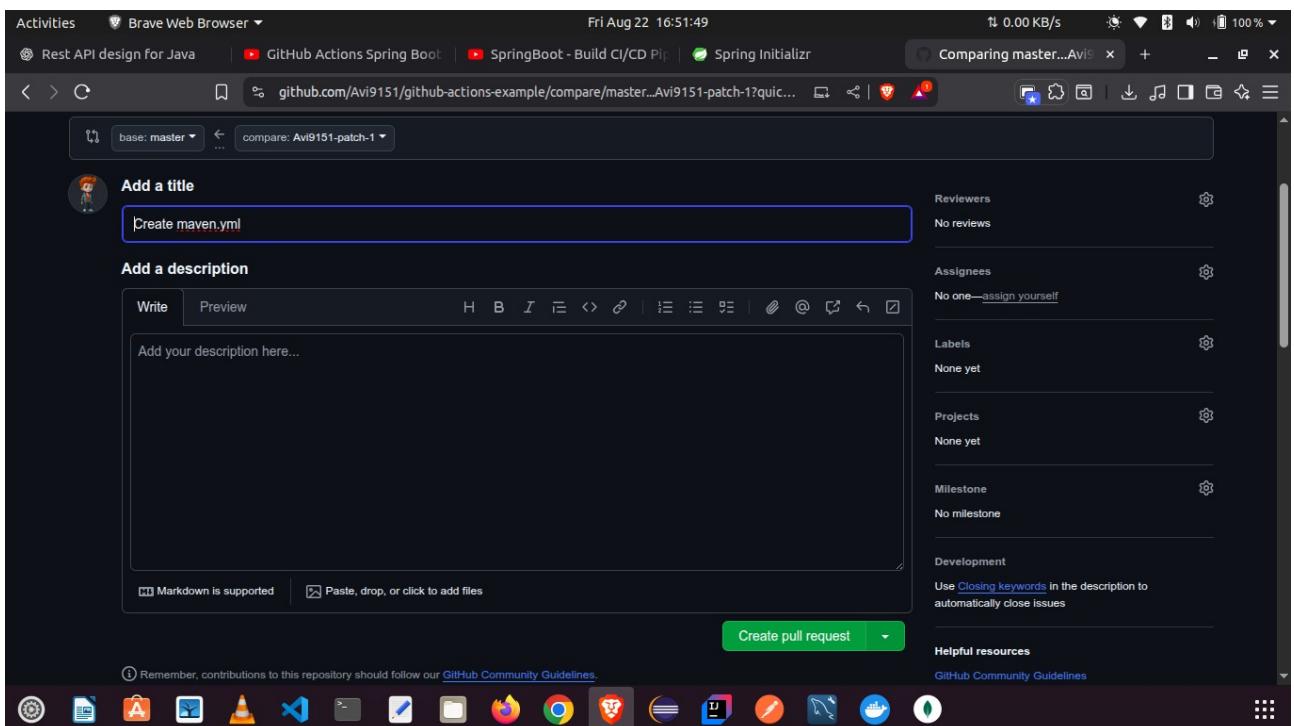
Save changes by clicking **Commit changes**.

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

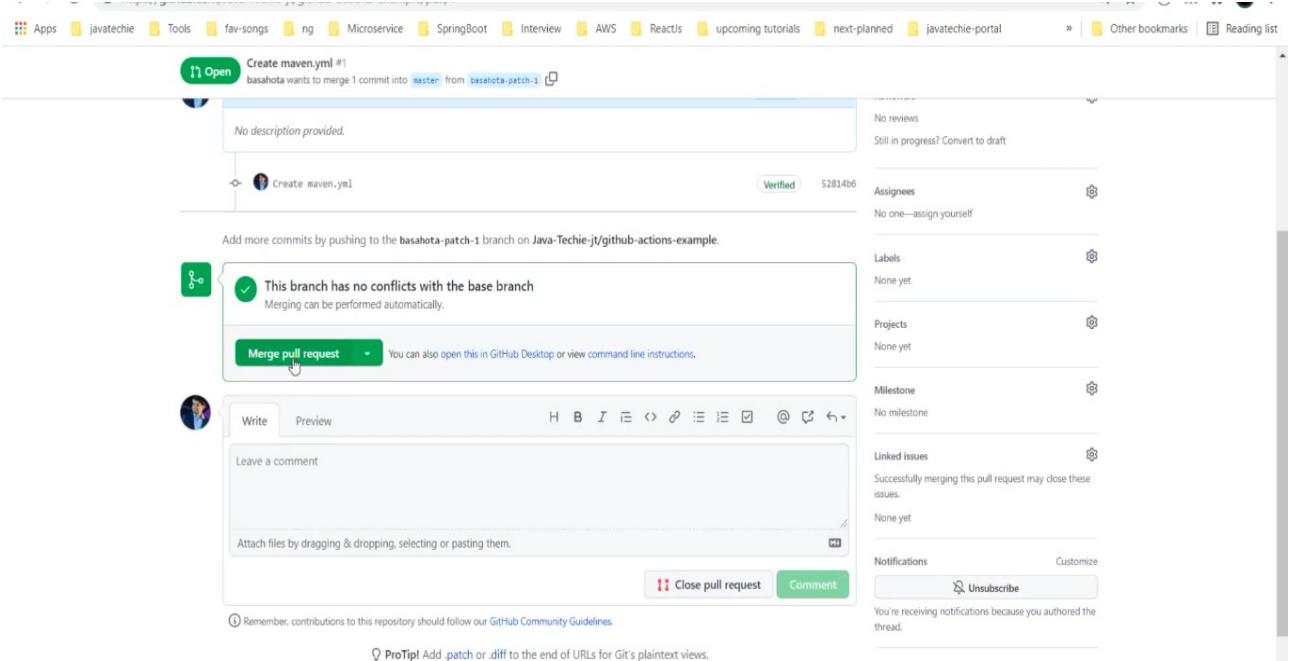


**Step 6:** Click on **Create a new branch for this commit and start a pull request** -> click on **Propose changes** button.

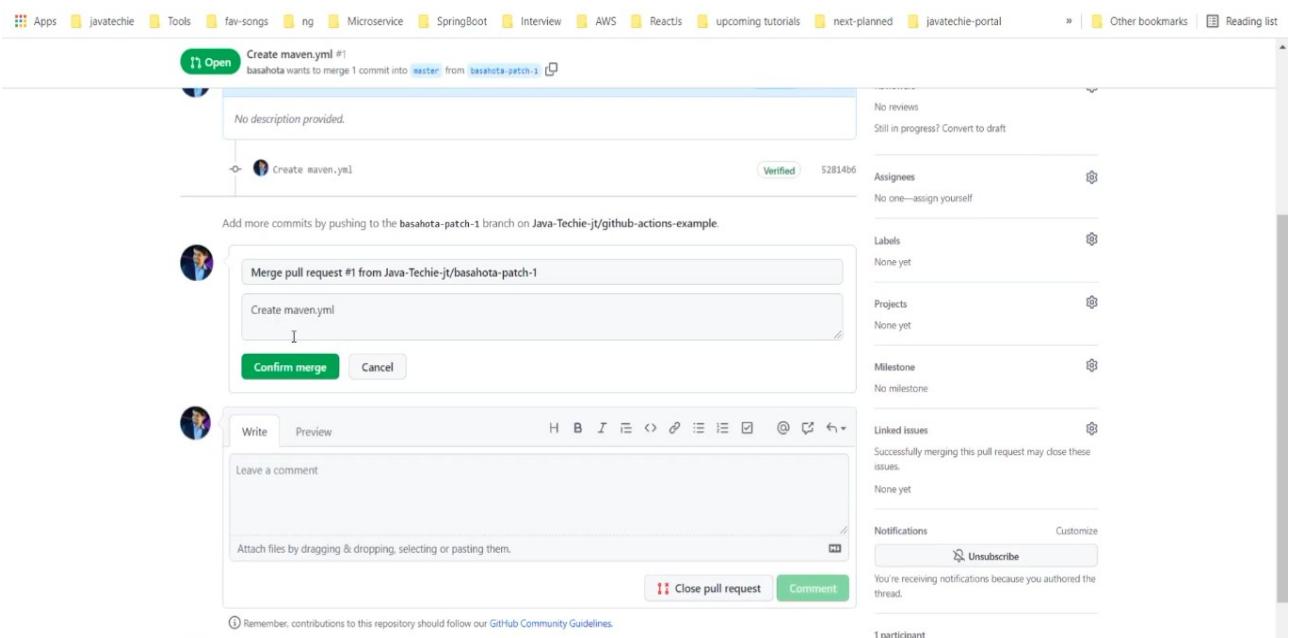
**Step 7:** Click on the **Create pull request** to merge these changes.



## 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 a Java file named `GithubCicdActionsApplication.java` open. A modal dialog titled "Pull to master" is displayed, containing the command `git pull origin master`. The "Pull" button is highlighted in blue.

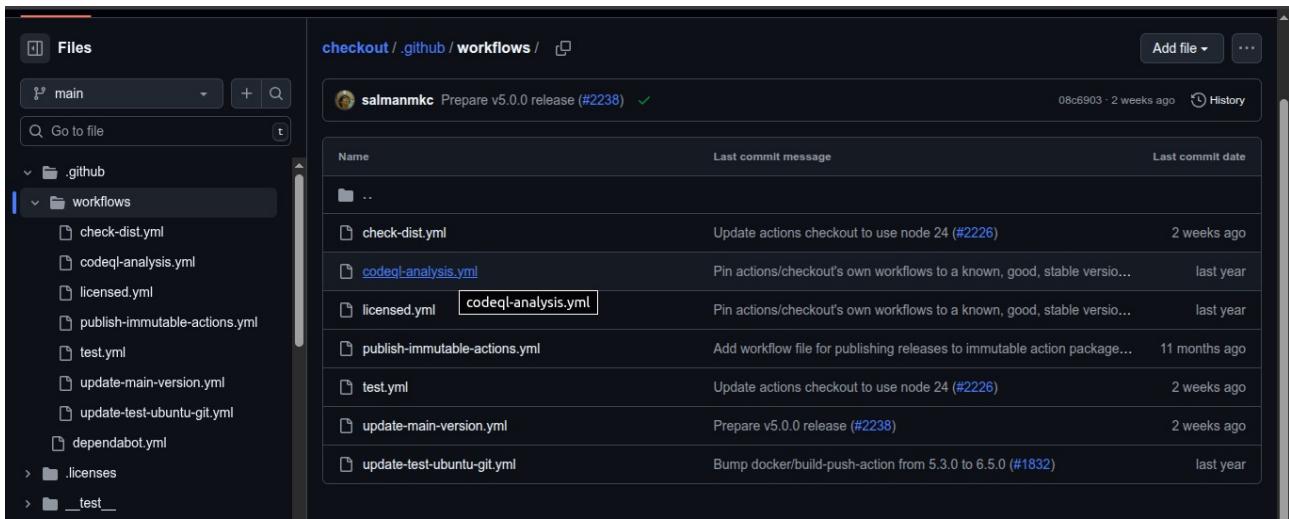
```
1 package com.avisoft.cicd;
2
3 > import ...
4
5 @SpringBootApplication
6 @RestController
7
8 public class GithubCicdActionsApplication {
9
10     @GetMapping("/welcome") no usages
11     public String welcome() {
12         return "Hello, Cicd!";
13     }
14
15     public static void main(String[] args) {
16         SpringApplication.run(GithubCicdActionsApplication.class, args);
17     }
18 }
```

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

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

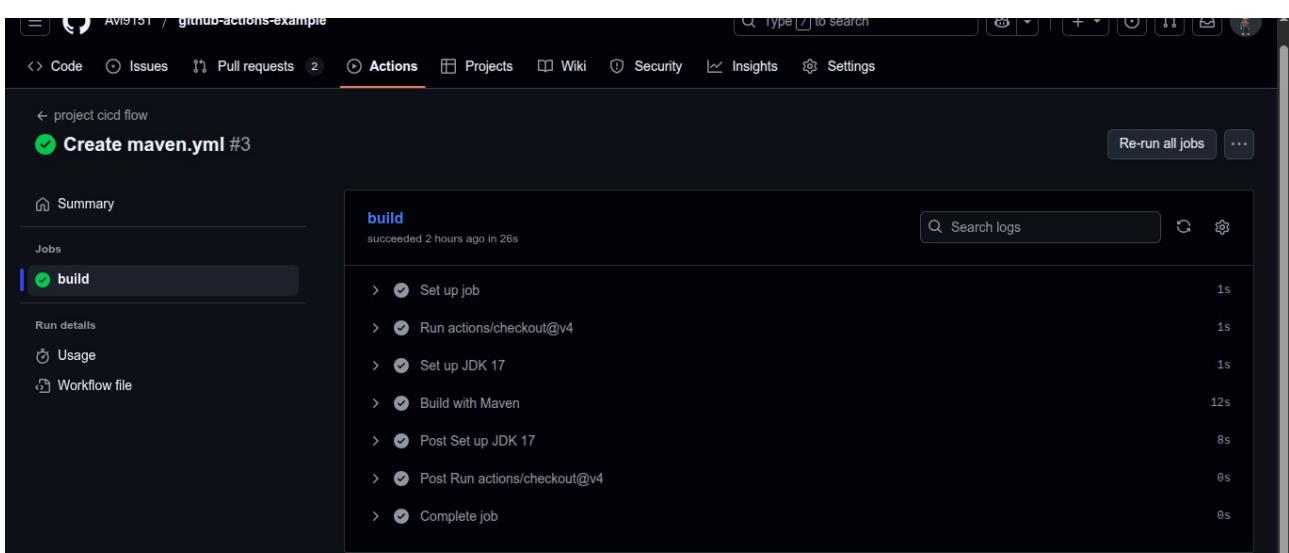
```
1 # This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the build time.
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-javascript#building-a-nodejs-project
3
4 # This workflow uses actions 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
6 # documentation.
7
8 name: project Cicd flow
9
10 on:
11   push:
12     branches: [ "master" ]
13   pull_request:
14     branches: [ "master" ]
15
16 permissions:
17   contents: read
18   dependencies: write
19
20 jobs:
21   build:
22     runs-on: ubuntu-latest
23
24     steps:
```

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



The screenshot shows the GitHub Actions workflow history for the repository `checkout / .github / workflows /`. A specific commit by `salmanmkc` titled "Prepare v5.0.0 release (#2238)" is highlighted. The commit message is "Pin actions/checkout's own workflows to a known, good, stable versio...". The commit was made 2 weeks ago. The workflow file `codeql-analysis.yml` is shown as the source of the changes. Other workflow files listed include `check-dist.yml`, `licensed.yml`, `publish-immutable-actions.yml`, `test.yml`, `update-main-version.yml`, and `update-test-ubuntu-git.yml`.

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



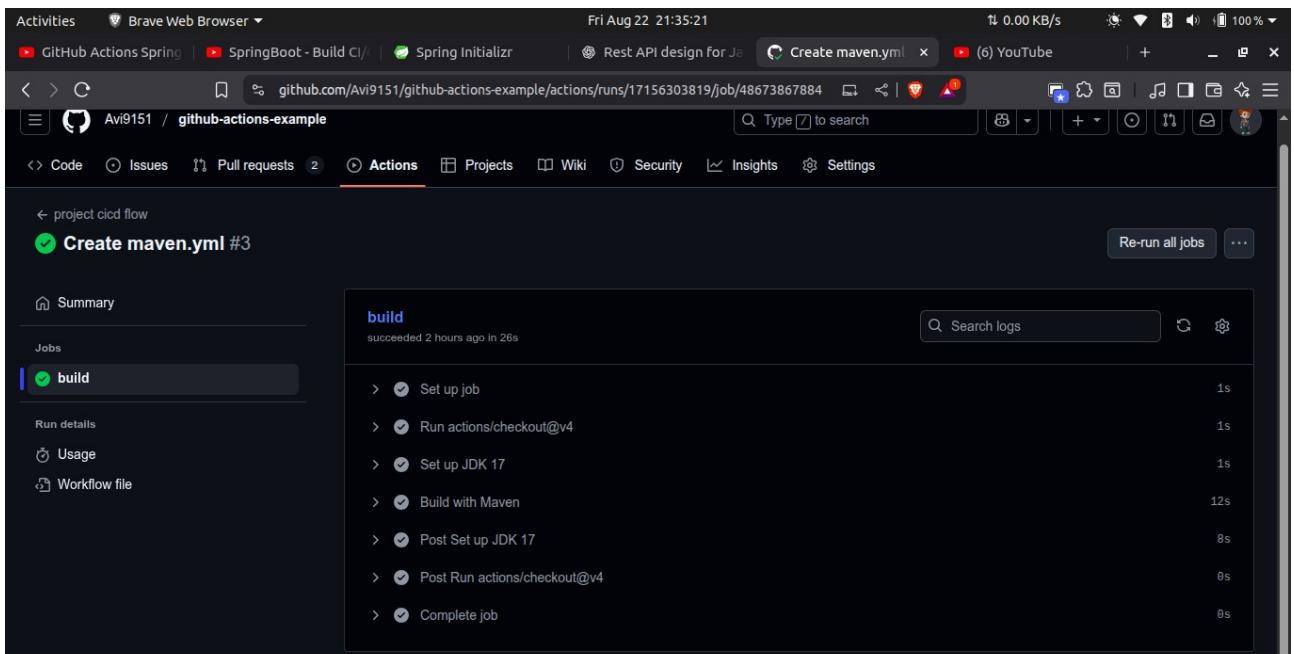
The screenshot shows the GitHub Actions job logs for a workflow named "Create maven.yml #3". The job is labeled "build" and has a status of "succeeded 2 hours ago in 26s". The logs show the execution of several steps: "Set up job" (1s), "Run actions/checkout@v4" (1s), "Set up JDK 17" (1s), "Build with Maven" (12s), "Post Set up JDK 17" (8s), "Post Run actions/checkout@v4" (0s), and finally "Complete job" (0s). A "Re-run all jobs" button is visible at the top right of the logs section.

A screenshot of a web browser displaying the GitHub Actions page for the repository "github-actions-example". The URL is [github.com/Avi9151/github-actions-example/actions](https://github.com/Avi9151/github-actions-example/actions). The page shows a message "Workflow run deleted successfully." and a list of "All workflows". There are two workflow runs listed:

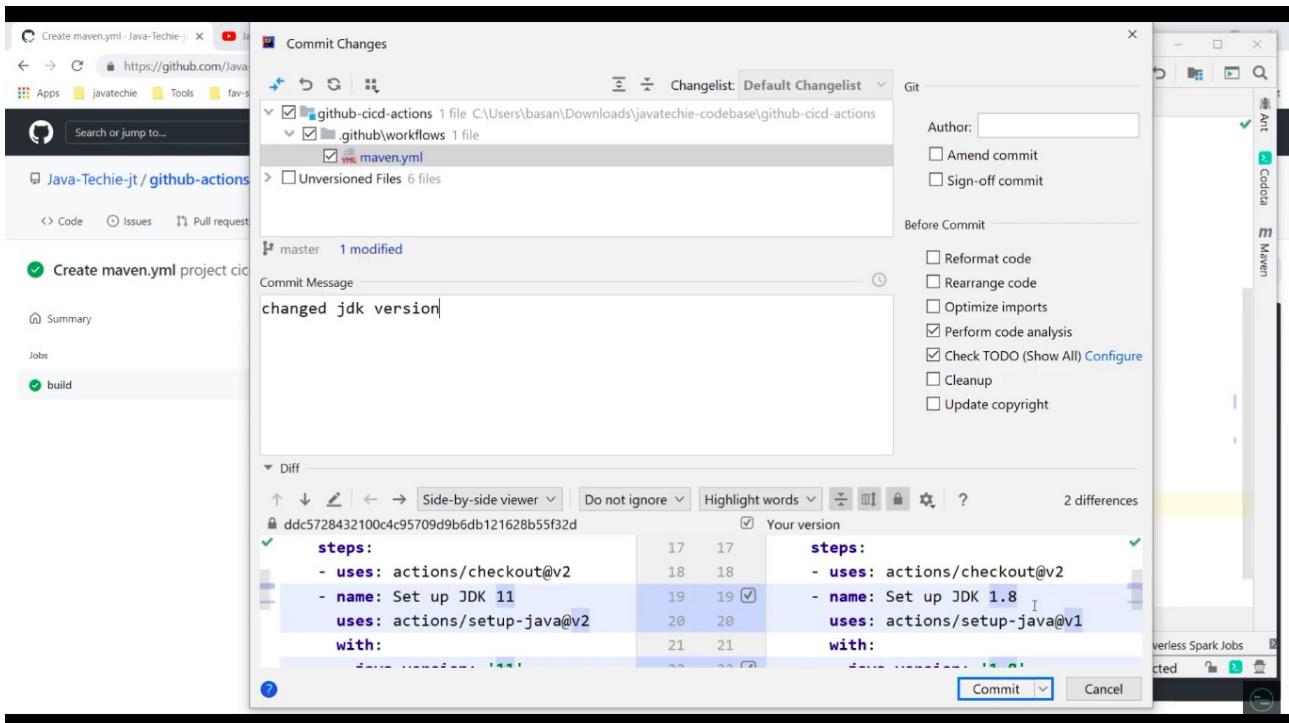
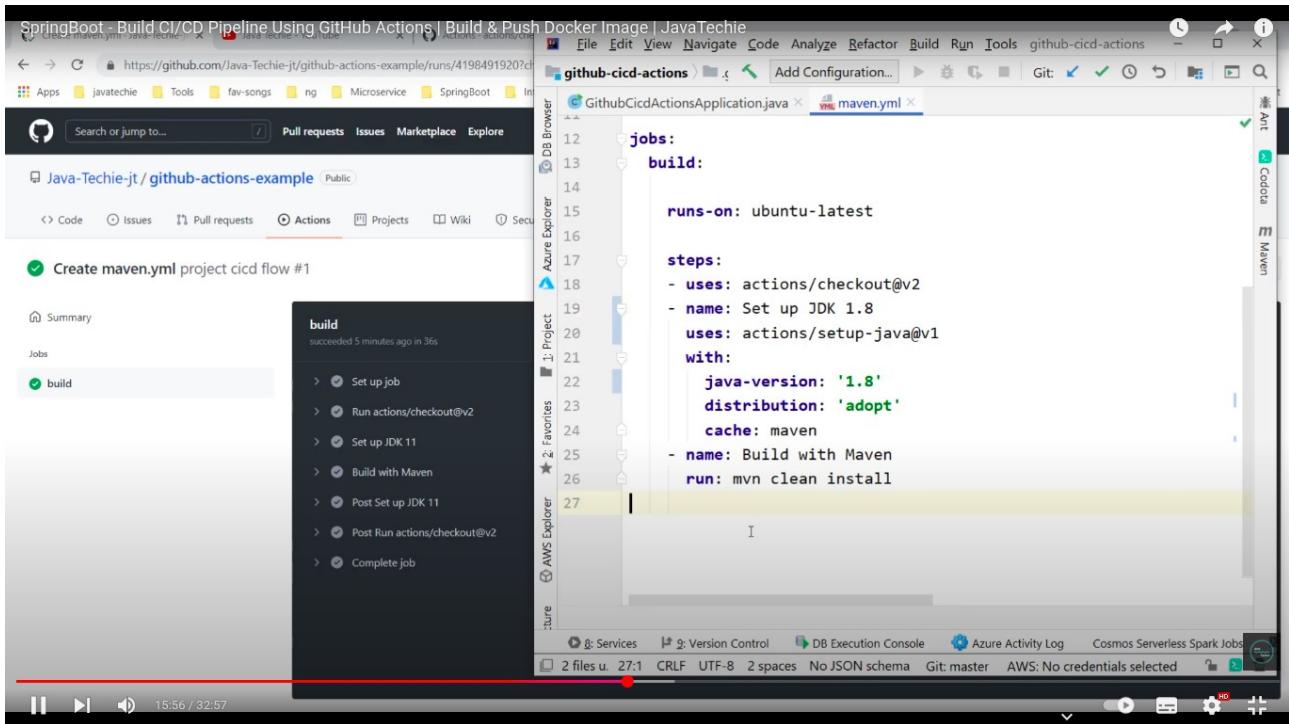
- Merge pull request #3 from Avi9151/Avi9151-patch-3**: Triggered via pull request 2 hours ago, Status: master, Duration: 27s.
- Create maven.yml**: Triggered via pull request #3: Pull request #3 opened by Avi9151, Status: Avi9151-patch-3, Duration: 30s.

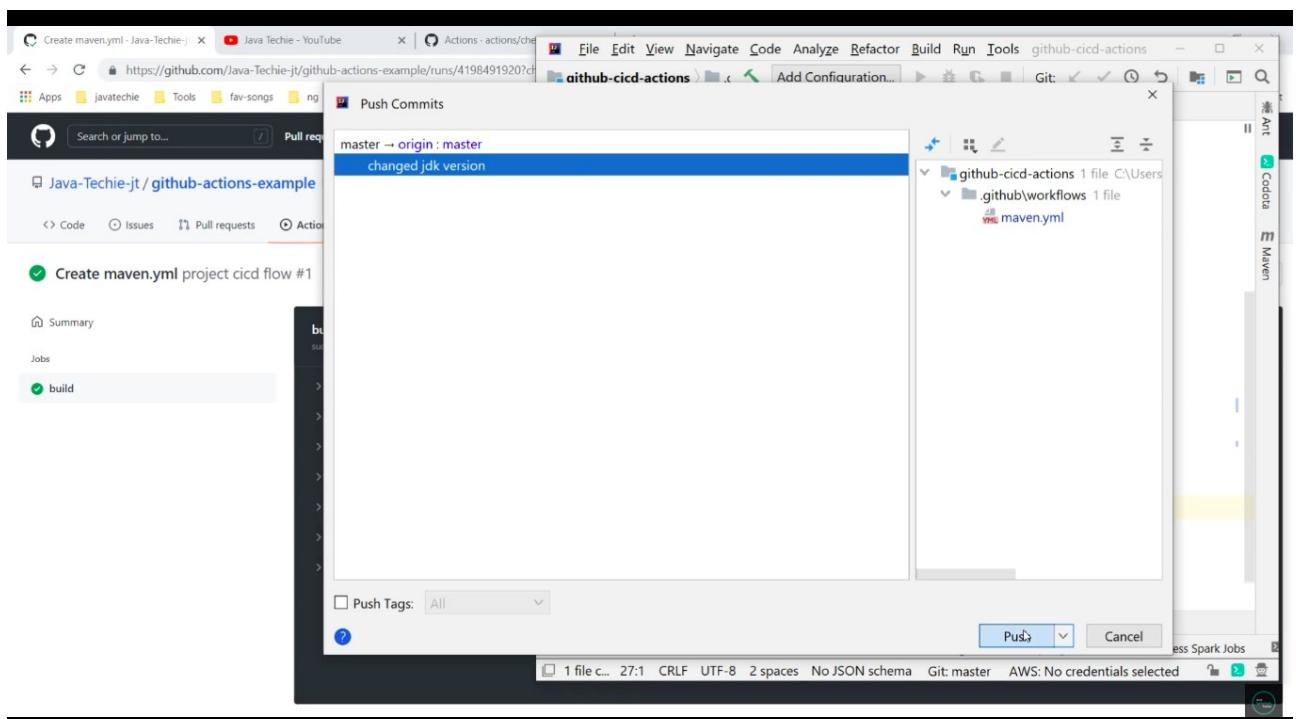
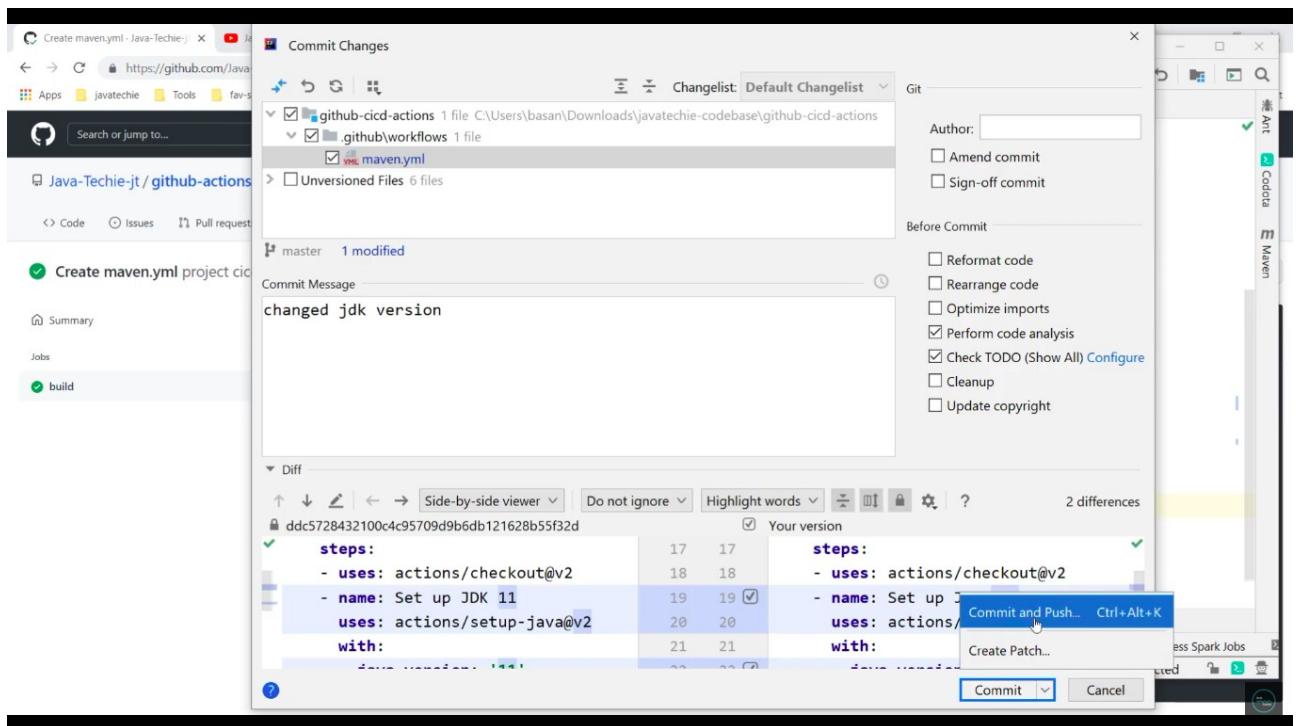
A screenshot of the GitHub Actions page showing the details of a specific workflow run. The URL is [github.com/Avi9151/github-actions-example/actions/runs/3](https://github.com/Avi9151/github-actions-example/actions/runs/3). The run is identified as "#3 Create maven.yml #3". The summary shows the run was triggered via pull request 2 hours ago, status is Success, total duration is 30s, and there are no artifacts. The "maven.yml" job is expanded, showing it was triggered on pull\_request and completed successfully in 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. The top navigation bar includes tabs for Pull requests, Issues, Marketplace, and Explore. Below this, the repository Java-Techie-jt/github-actions-example is selected. The main navigation bar has tabs for Code, Issues, Pull requests, Actions (which is highlighted), Projects, Wiki, Security, Insights, and Settings. The Actions tab is currently active, displaying the 'Workflows' section. A 'New workflow' button is visible. A feedback card asks for suggestions on how GitHub Actions can work better. The 'All workflows' section shows a single workflow named 'project cicd flow'. The 'All workflow runs' table lists three runs:

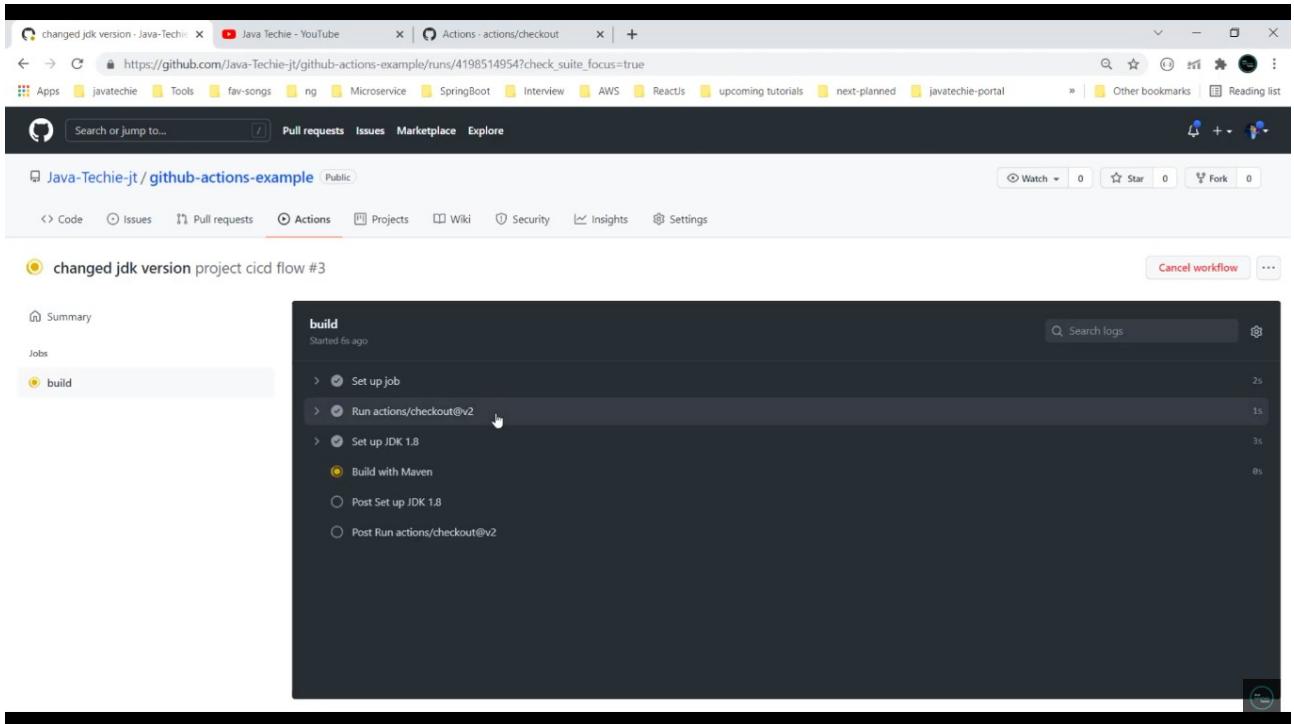
Event	Status	Branch	Actor
changed jdk version	now	master	basahota
Merge pull request #1 from Java-Techie-jt/basahota-patch...	6 minutes ago	master	basahota
Create maven.yml	6 minutes ago	basahota-patch-1	basahota

## Click on build

The screenshot shows the GitHub Actions interface, similar to the previous one but focusing on a specific workflow run. The top navigation bar and repository selection are identical. The main navigation bar is also identical. The 'Actions' tab is active. The 'changed jdk version project cicd flow #3' workflow run is selected. The 'Summary' tab is active, showing the run was triggered via push 10 seconds ago by basahota (commit 14d067b, master branch). The status is 'In progress'. The 'Jobs' section shows a single job named 'maven.yml' which is currently 'build'ing. The build status is shown as '2s'.

Waiting for pipelines.actions.githubusercontent.com...

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 web browser window titled "Brave Web Browser". The address bar shows the URL: "google.com/search?q=build+and+push+docker+image+using+github&oq=build+and+push+...". The search term "build and push docker image using github" is visible in the search bar. Below the search bar, there is a snippet from the GitHub Marketplace listing for the "Docker Build & Push Action". The snippet includes the GitHub logo, the action name, a brief description, and a link to the marketplace page.

A screenshot of a web browser window titled "Brave Web Browser". The address bar shows the URL: "github.com/marketplace/actions/docker-build-push-action". The page displays the "Docker Build & Push Action" listing. It includes sections for "About", "Supported Docker registries", "Features", and "Breaking changes". On the right side, there are sections for "Contributors" (18 contributors shown with their profile pictures) and "Resources" (links to open issues, pull requests, source code, and report abuse). The "About" section highlights that it builds a Docker image and pushes it to a private registry with support for multiple tags.

A screenshot of a web browser window titled "Brave Web Browser". The address bar shows the URL: "github.com/marketplace/actions/docker-build-push-action". The page displays the "Docker Build & Push Action" configuration interface. It shows a code block with a GitHub workflow YAML file snippet, an "Inputs" table, and a large text area for "Workflow description".

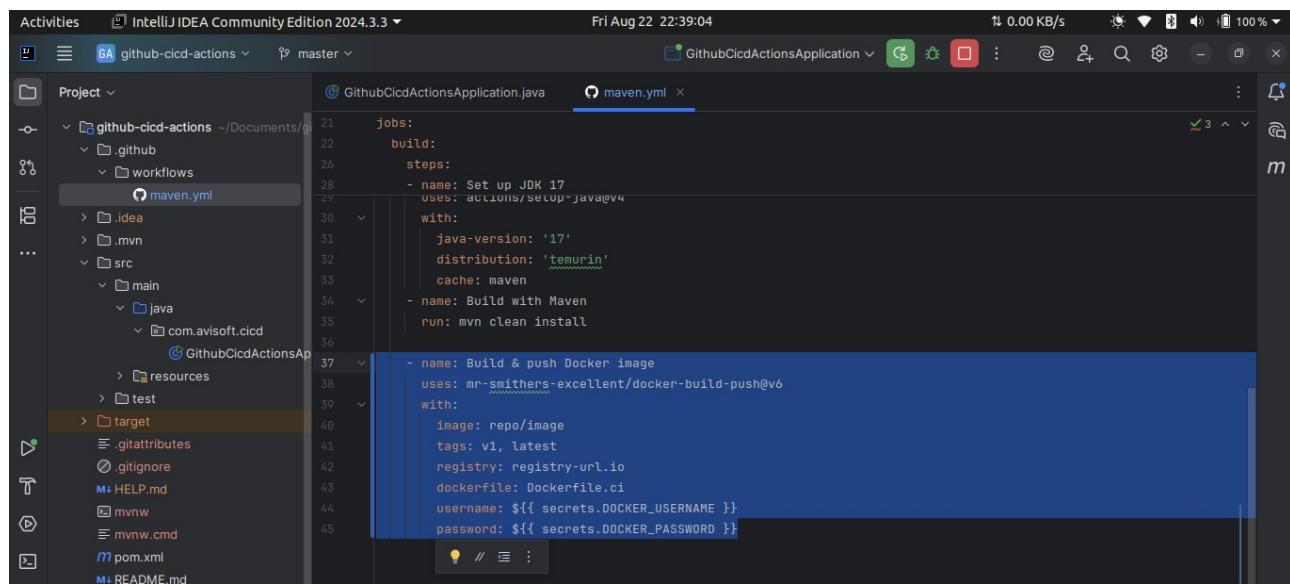
```
steps:
- uses: actions/checkout@v3
  name: Check out code

- 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 }}
```

Name	Description	Required	Type

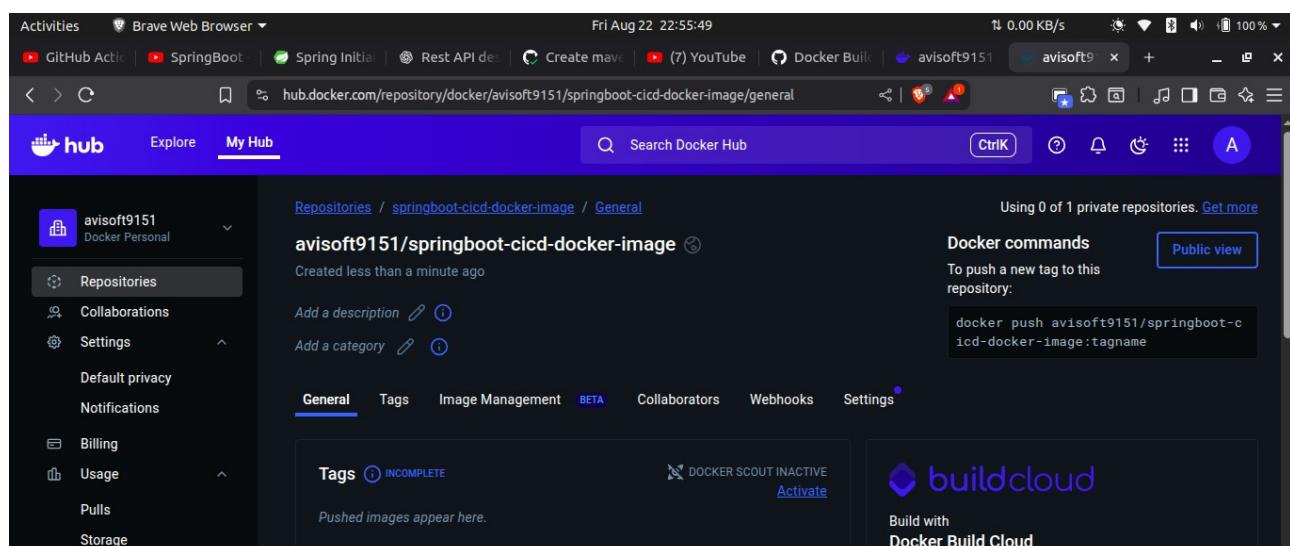
## Step 13: Add this yml code in maven.yml.

```
- 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" ]

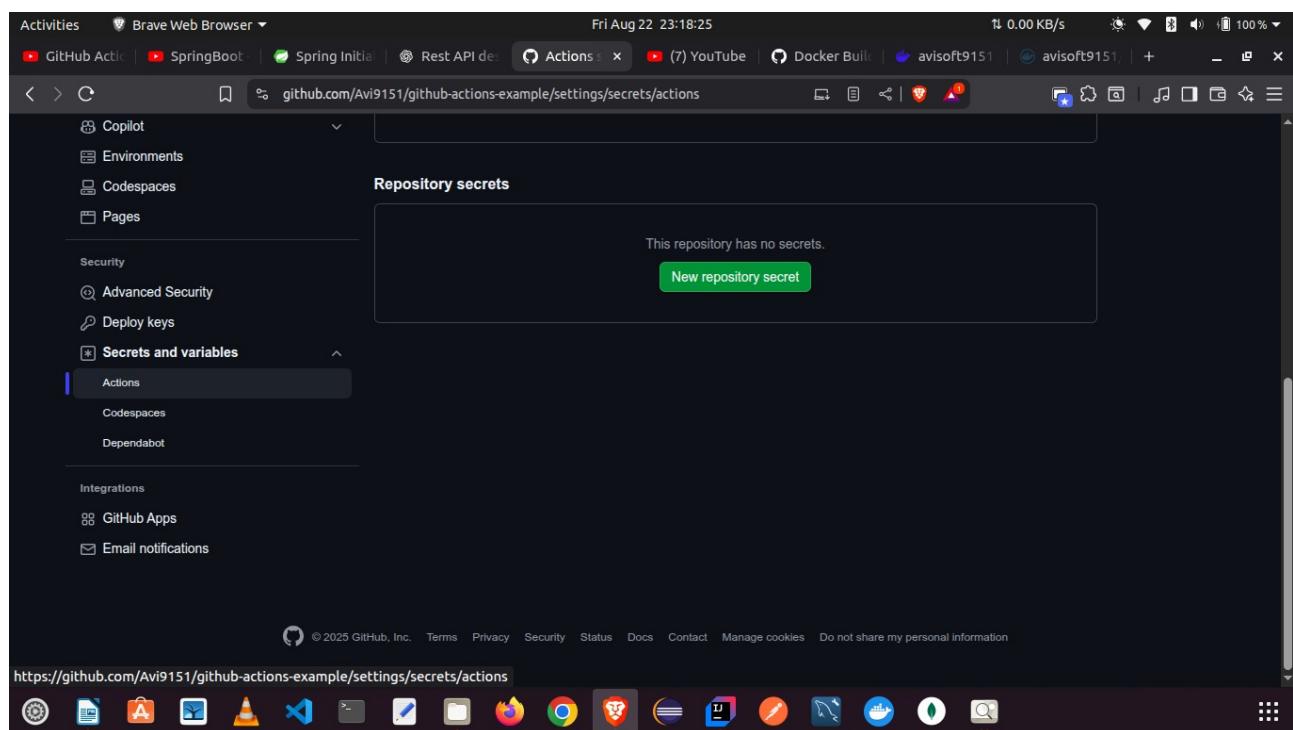
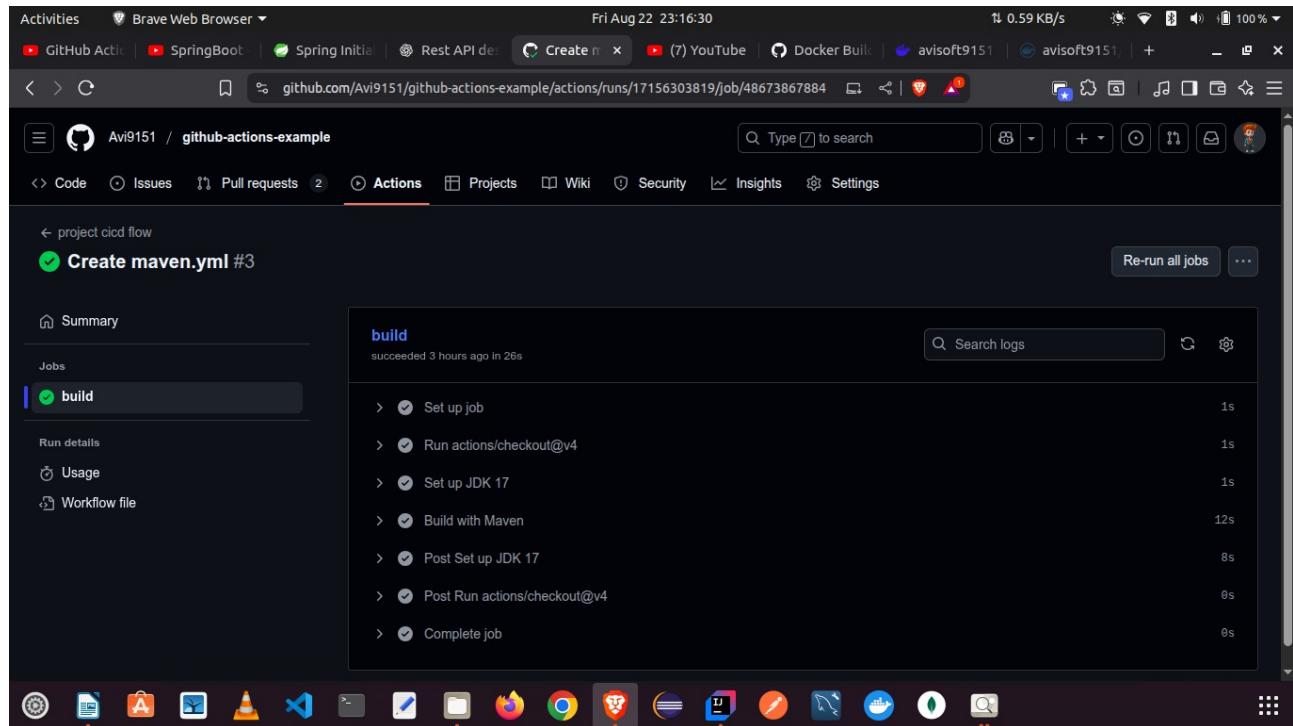
permissions:
  contents: write
  security-events: write

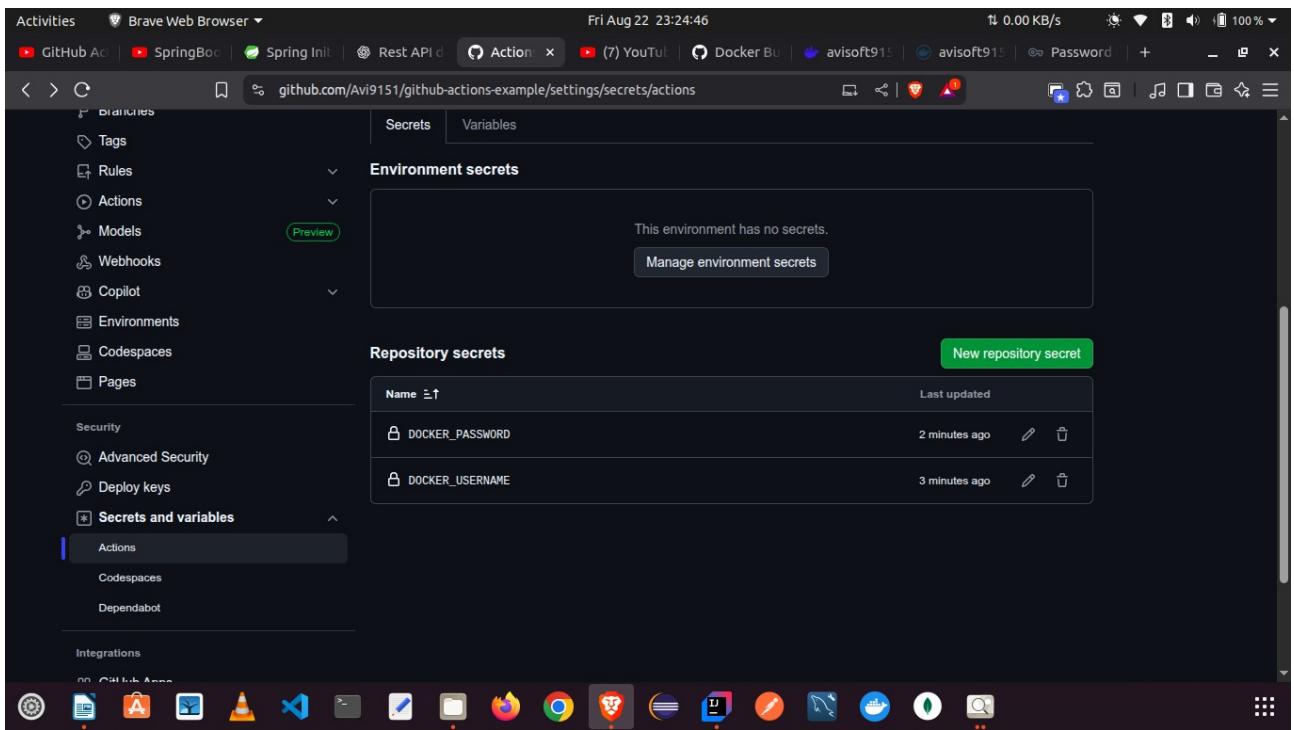
jobs:
  build:
    runs-on: ubuntu-latest

    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.

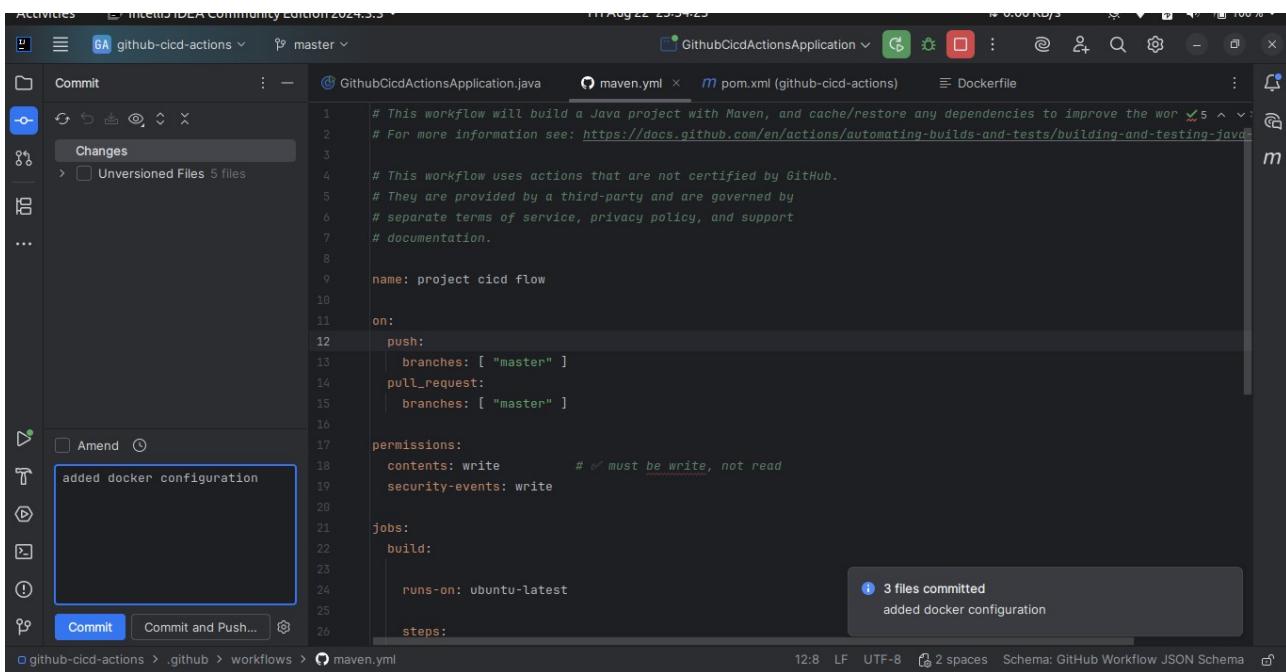
```
FROM openjdk:17
EXPOSE 8080
ADD target/springboot-cicd-docker-image.jar springboot-cicd-docker-image.jar
ENTRYPOINT ["java", "-jar", "/springboot-cicd-docker-image.jar"]
```

**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 project open. The code editor displays the `maven.yml` file, which contains a GitHub workflow configuration. The workflow is triggered on pushes to the `master` branch and runs on an `ubuntu-latest` runner. It includes permissions for writing contents and security events, and a job named `build` with a single step. A commit message `added docker configuration` is visible in the commit dialog, and a notification at the bottom right 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's performance.
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-javascript-with-maven

# 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.

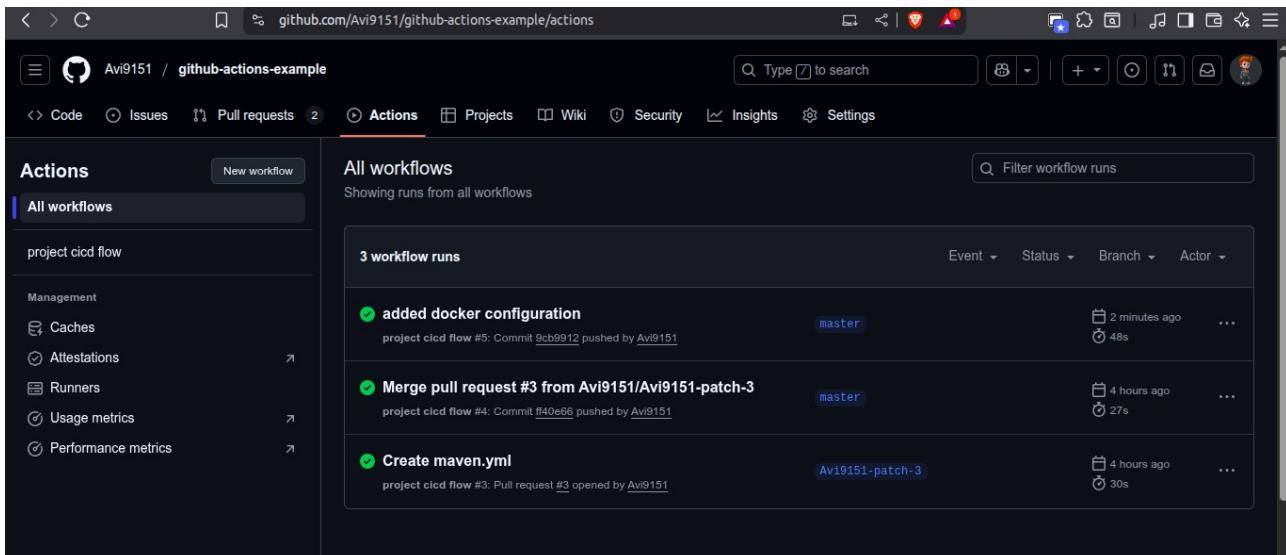
name: project cicd flow

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

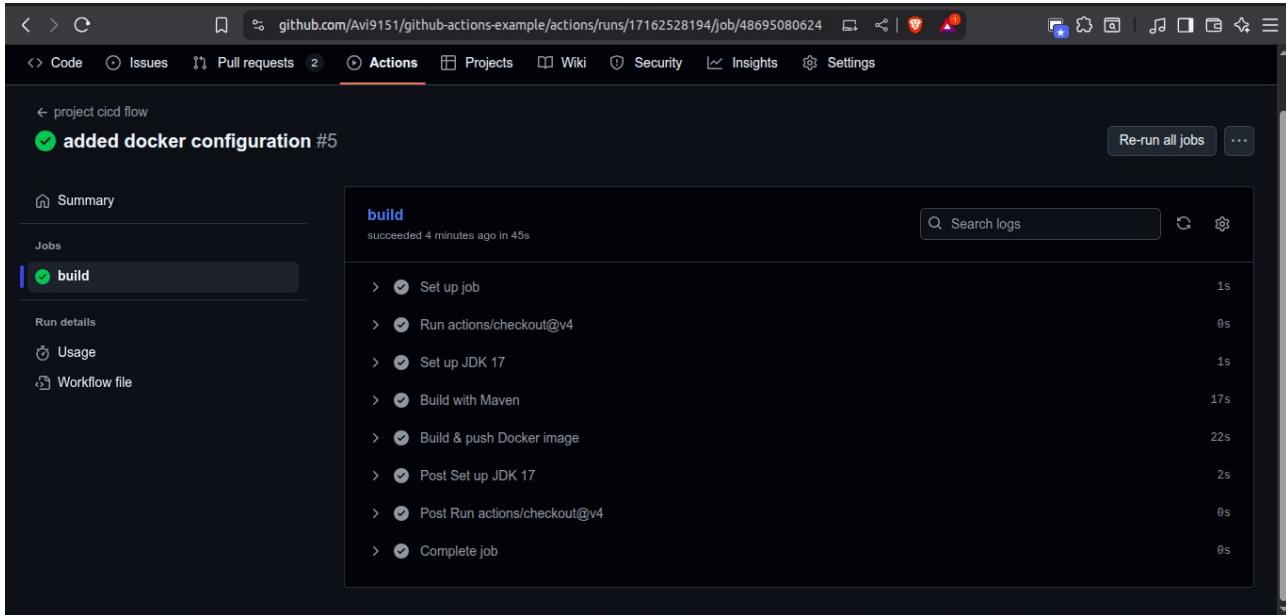
permissions:
  contents: write          # must be write, not read
  security-events: write

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.

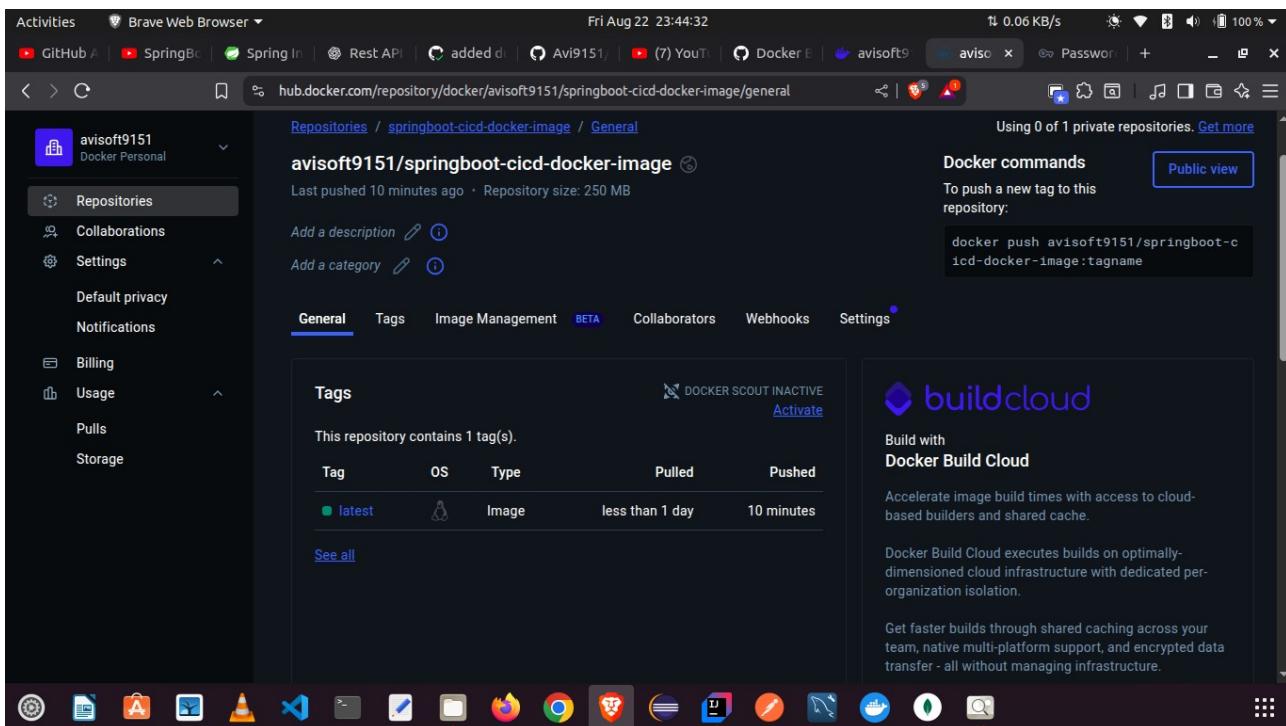


**Step 22:** Click on the added docker configuration -> build and you can see start build with maven



**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.

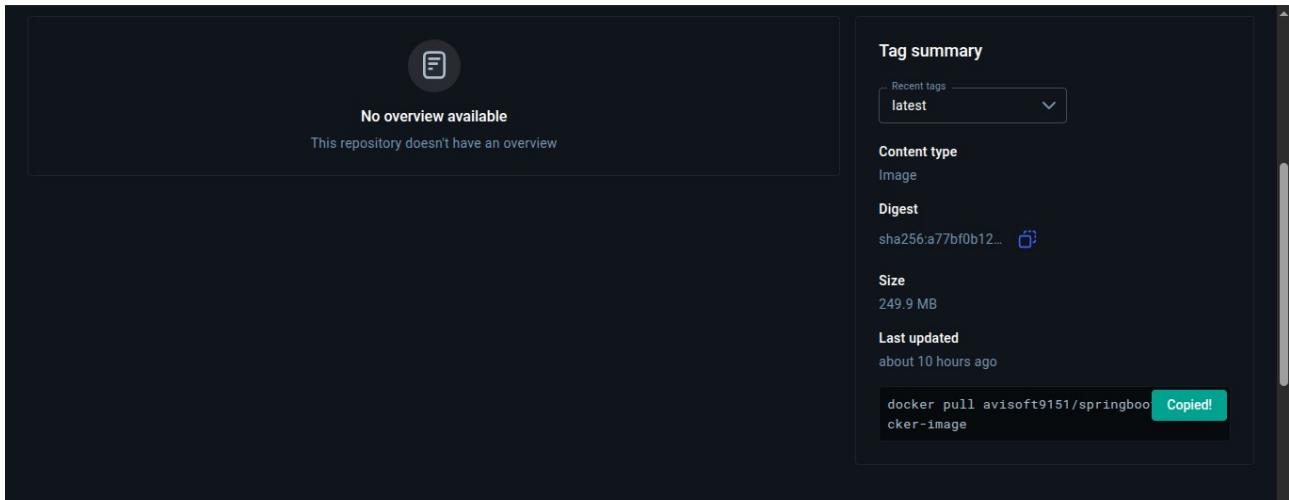
And you can click on **latest** tag and see.

The screenshot shows the Docker Hub interface for the repository `avisoft9151/springboot-cicd-docker-image`. The `latest` tag is selected. The page displays the manifest digest, OS/ARCH, compressed size, last pushed time, type, and manifest digest. Below this, the `Image Layers` section lists the layers of the image, showing commands like ADD file, CMD, ENV, and RUN. A command editor is visible on the right side of the layer list.

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

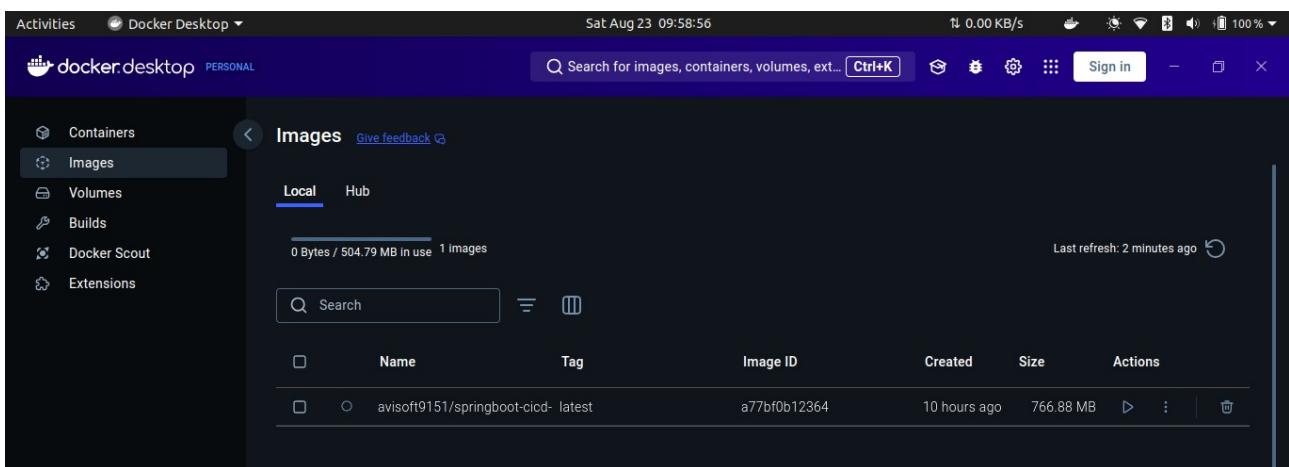
The screenshot shows the Docker Hub interface for the repository `avisoft9151/springboot-cicd-docker-image` on the `General` tab. It shows the last push time, repository size, and Docker commands for pushing a new tag. On the right, there is a promotional section for `buildcloud`, which offers accelerated image build times with cloud-based builders and shared cache. It highlights Docker Build Cloud's ability to execute builds on optimally dimensioned cloud infrastructure with dedicated per-organization isolation.

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



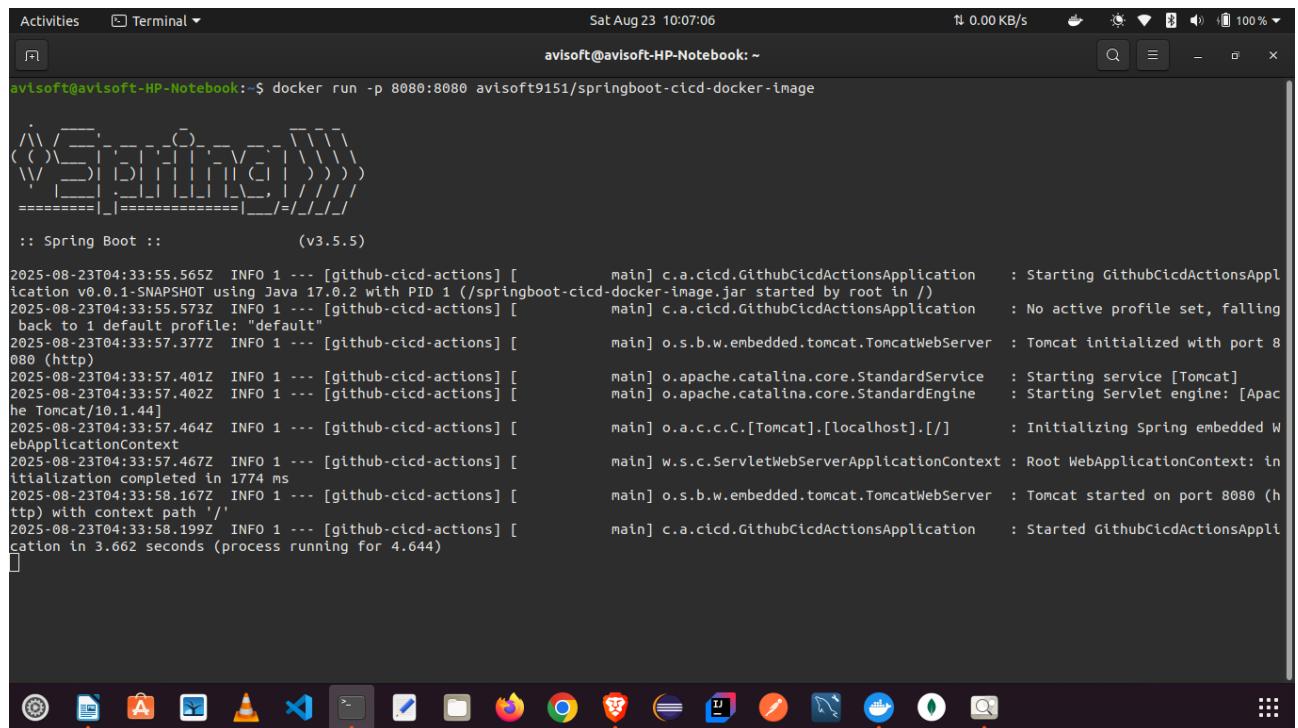
```
Activities Terminal ▾ Sat Aug 23 09:21:39 avisoft@avisoft-HP-Notebook: ~
avisoft@avisoft-HP-Notebook: $ docker pull avisoft9151/springboot-cicd-docker-image
Using default tag: latest
latest: Pulling from avisoft9151/springboot-cicd-docker-image
38a980f2cc8a: Pull complete
de849fc1cbe6: Pull complete
a7203ca35e75: Pull complete
8fd1cdd58553: Pull complete
Digest: sha256:a77bf0b12364462d81fe397855f6caadb5f82d26de26b976ffc8437c825baa3
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](#)



```
Activities Terminal ▾ Sat Aug 23 10:07:06 avisoft@avisoft-HP-Notebook:~  
avisoft@avisoft-HP-Notebook:~$ docker run -p 8080:8080 avisoft9151/springboot-cicd-docker-image  
:: Spring Boot ::          (v3.5.5)  
2025-08-23T04:33:55.565Z INFO 1 --- [github-cicd-actions] [           main] c.a.cicd.GithubCicdActionsApplication : Starting GithubCicdActionsAppl  
ication v0.0.1-SNAPSHOT using Java 17.0.2 with PID 1 (/springboot-cicd-docker-image.jar started by root in /)  
2025-08-23T04:33:55.573Z INFO 1 --- [github-cicd-actions] [           main] c.a.cicd.GithubCicdActionsApplication : No active profile set, falling  
back to 1 default profile: "default"  
2025-08-23T04:33:57.377Z INFO 1 --- [github-cicd-actions] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8  
080 (http)  
2025-08-23T04:33:57.401Z INFO 1 --- [github-cicd-actions] [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]  
2025-08-23T04:33:57.402Z INFO 1 --- [github-cicd-actions] [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apac  
he Tomcat/10.1.44]  
2025-08-23T04:33:57.464Z INFO 1 --- [github-cicd-actions] [           main] o.a.c.c.c.[Tomcat].[localhost].[/] : Initializing Spring embedded W  
ebApplicationContext  
2025-08-23T04:33:57.467Z INFO 1 --- [github-cicd-actions] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: in  
itialization completed in 1774 ms  
2025-08-23T04:33:58.167Z INFO 1 --- [github-cicd-actions] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (h  
ttp) with context path ''  
2025-08-23T04:33:58.199Z INFO 1 --- [github-cicd-actions] [           main] c.a.cicd.GithubCicdActionsApplication : Started GithubCicdActionsAppl  
ication in 3.662 seconds (process running for 4.644)  
[]
```

- 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).