

## CI/CD Pipeline: Building and Pushing Docker Images from GitHub to Docker Hub Using GitHub Actions

### Prerequisites

Before starting, make sure you have:

- A GitHub account
- A Docker Hub account
- Docker installed locally (for testing)
- Your project hosted on GitHub (e.g., a simple Spring Boot App with a Dockerfile)

### Step 1: Create a "Dockerfile" in Your Project

github/workflows	Update docker-image.yml	32 minutes ago
src	Initial	3 months ago
tar_get	Create violetApple-docker.jar	3 months ago
.gitignore	Initial	3 months ago
Dockerfile	Initial	3 months ago
README.md	Initial commit	3 months ago
mvnw	Initial	3 months ago
mvnw.cmd	Initial	3 months ago
pom.xml	Initial	3 months ago

```
1. # 1Use an official Java runtime as the base image
2. FROM openjdk:17-jdk-slim
3.
4. # 2Set the working directory inside the container
5. WORKDIR /app
6.
7. # 3Copy the built JAR file into the container
8. COPY target/violetApple-docker.jar app.jar
9.
10. # 4Expose the application port
11. EXPOSE 8080
12.
13. # 5Run the application
14. ENTRYPOINT ["java", "-jar", "app.jar"]
```

### Step 2: Create a Repository on GitHub

If you haven't already:

1. Push your local project to GitHub
2. Make sure the Dockerfile is in the root or specified directory

VioletApple_SpringBoot_Project Public		Pin	Unwatch 1
main	1 Branch	0 Tags	Go to file
rakeshsharma-ctl Update docker-image.yml ✓		be10c3c · 36 minutes ago	10 Commits
github/workflows	Update docker-image.yml	36 minutes ago	
src	Initial	3 months ago	
tar_get	Create violetApple-docker.jar	3 months ago	
.gitignore	Initial	3 months ago	
Dockerfile	Initial	3 months ago	
README.md	Initial commit	3 months ago	
mvnw	Initial	3 months ago	
mvnw.cmd	Initial	3 months ago	
pom.xml	Initial	3 months ago	

A typical GitHub repository root view

### Step 3: Store Docker Hub Credentials as GitHub Secrets

Go to your GitHub repo → Settings → Secrets and variables → Actions → New repository secret

Add these secrets:

Name	Value
DOCKER_HUB_TOKEN	Your Docker Hub username

Actions secrets / New secret

Name \*

DOCKER\_HUB\_TOKEN

Secret \*

asdfasd\_sampel\_token\_text

Add secret

### Step 4: Create GitHub Actions Workflow

In your repo, create this file:

```
1. .github/workflows/docker-image.yml
```

Sample “docker-image.yml” for a SpringBoot Application

```
1. name: Docker Image CI
2.
3. on:
4.   push:
5.     branches: [ "main" ]
6.   pull_request:
7.     branches: [ "main" ]
8.
9. jobs:
10.  build-and-push:
11.    runs-on: ubuntu-latest
12.
13.    steps:
14.      - name: Checkout Code
15.        uses: actions/checkout@v4
16.
17.      - name: Set up JDK 17
18.        uses: actions/setup-java@v3
19.        with:
20.          java-version: '17'
21.          distribution: 'temurin'
22.
23.      - name: Build with Maven
24.        run: mvn clean package -DskipTests
```

Action workflow gets executed when push or pull request is initiated in “main” branch.

We have used ubuntu latest version as operating system.

```

25.
26.   - name: Log in to Docker Hub using CLI
27.     run: docker login -u sharmarakesh -p ${ secrets.DOCKER_HUB_TOKEN }}
28.
29.   - name: Build Docker Image
30.     run: |
31.       docker build -t sharmarakesh/violetapple-springboot-project:latest .
32.
33.   - name: Push Docker Image to Docker Hub
34.     run: |
35.       docker push sharmarakesh/violetapple-springboot-project:latest

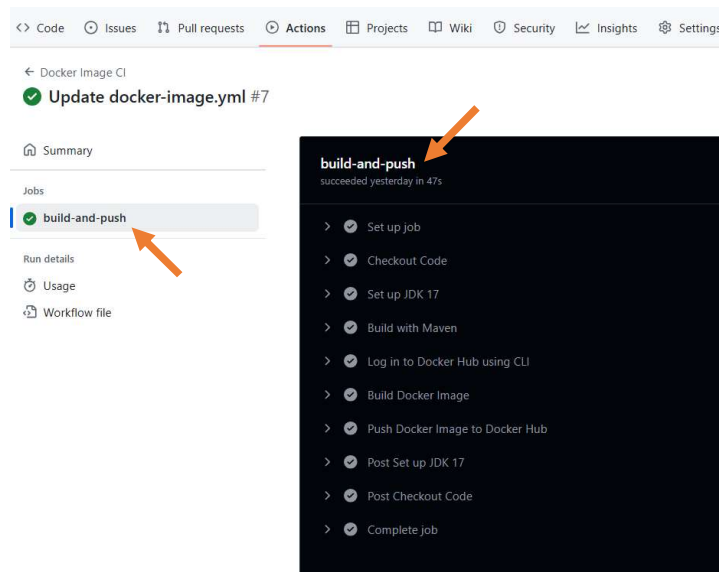
```

Annotations:

- Line 26: **Log in to Docker Hub using CLI**
- Line 29: **Build Docker Image**
- Line 33: **Push Docker Image to Docker Hub**
- Line 27: **Docker token stored in Action secrets**
- Line 30: **Github username**
- Line 31: **Name image always as latest.**

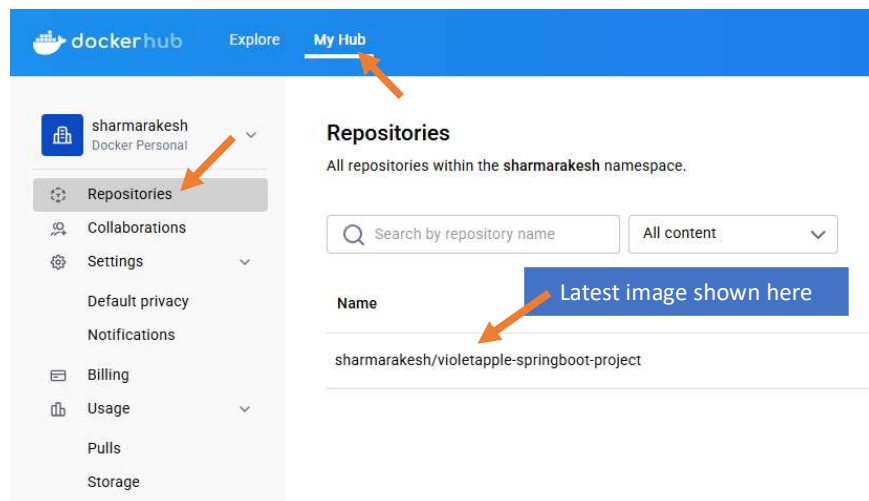
## Step 5: Commit and Push the Workflow

Once the workflow YAML is committed and pushed to the main branch, it will automatically trigger the pipeline.



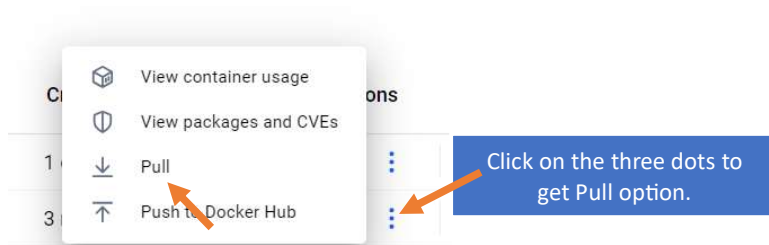
## Step 6: Confirm on Docker Hub

Go to Docker Hub → Check your repository → The latest image should be pushed.

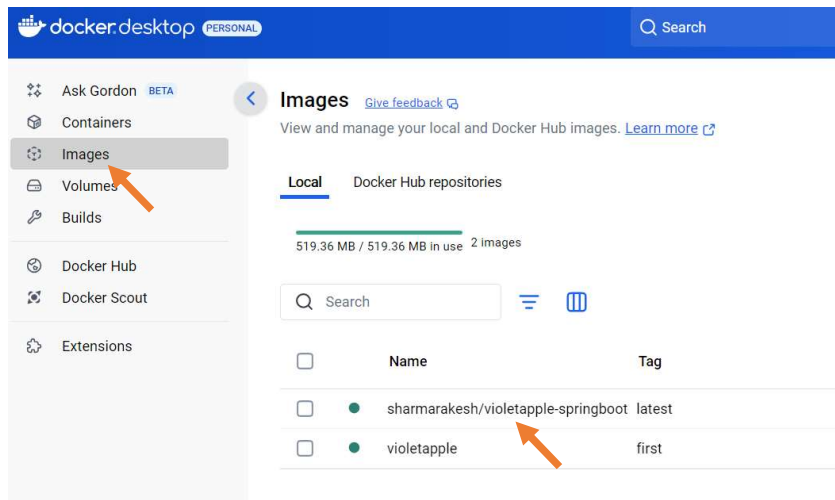


## Step 6: Test on Docker Desktop

1. Once the image is pushed to Docker Hub, you can test it locally using Docker Desktop.
2. Pull the image from Docker Hub.



3. Run a container from the image.



4. Open Docker Desktop to confirm the container is running.
5. Visit <http://localhost:8080> (or your chosen port) in a browser to test the application.