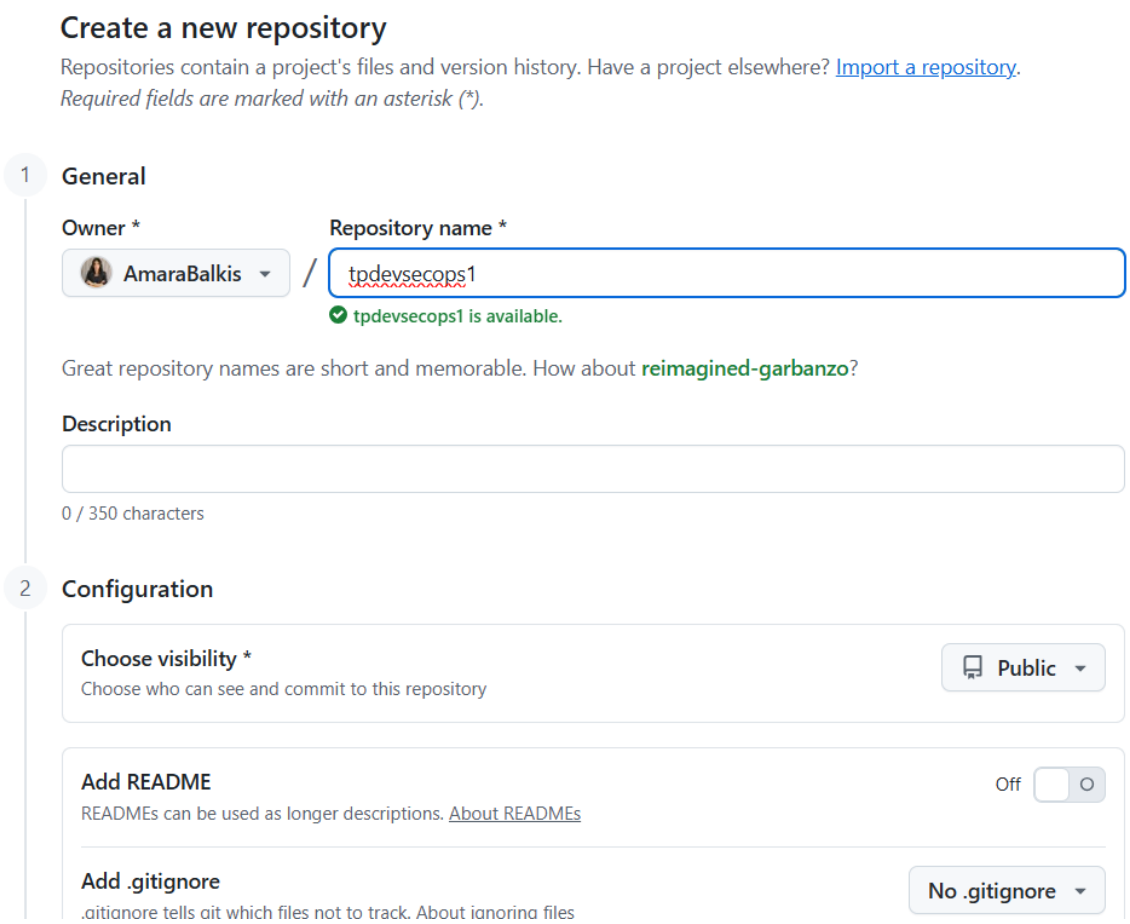
I. Part1 : GIT

1. Create a GitHub Repository

Create a GitHub Repository

- Go to GitHub and Click New Repository
- Name your project: tp-devsecops
- Do not check “Initialize with README”
- Click Create repository



2. Initialize Git

Create a local Git repository for your project

- Make sure that you are in your path of project
- git init

  => creates a Git repository in your project (a hidden .git folder).
- Your project is now tracked by Git.

```
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git init
Initialized empty Git repository in C:/Users/ASUS/Desktop/EPI/5®me ANG/TP/.git/
```

3. Add Files

git add .

```
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git add .
```

- ⇨ The . adds **all files** in the project.
- ⇨ Prepares the files for the **first commit**.

4. First Commit

git commit -m "Initial commit"

```
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git commit -m "initial commit"
[master (root-commit) 53b8809] initial commit
 4 files changed, 28 insertions(+)
```

- ⇨ A commit is a **snapshot** of your project.
- ⇨ "Initial commit" → message describing this commit.

5. Rename the Main Branch

git branch

git branch -M main

```
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git branch
* master
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git branch -M main
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git branch
* main
```

- ⇨ M : rename the current branch, force if necessary.

6. Link Local Repository to GitHub

git remote add origin https://github.com/xxxxxxxx/tpdevsecops.git

```
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git remote add origin https://github.com/AmaraBalkis/tpdevsecops
```

- ⇨ origin : standard name for the remote repository.

7. Push Project to GitHub

git push -u origin main

```
PS C:\Users\ASUS\Desktop\EPI\5éme ANG\TP> git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
```

- ⇨ -u : sets main as default branch for future push/pull.

Check your Git repo

II. Part 2:

1. Check tool versions:

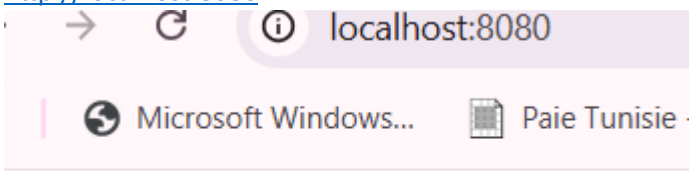
Docker –version
Java –version
Maven -version

2. Create the application:

We will follow the structure shown in the picture below to create the folders and files of the application.

3. Test the app locally:

- mvn clean package => BUILD SUCCESS
- cd target
- This folder is automatically generated by Maven during the build. It contains the compiled .jar file and other temporary build files.)
- java -jar democyber-0.0.1-SNAPSHOT.jar
- http://localhost:8080



Bonjour depuis democyber Spring Boot !

III. Part3 : Docker

1. Create the image:

- Check your Docker username
- Open Docker Desktop

Then, in the terminal:

docker build -t username/democyber:latest .

```
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP> docker build -t amarabalkis/democyber:latest .
 => => transferring dockerfile: 375B
 => [internal] load metadata for docker.io/library/eclipse-temurin:21-jdk-jammy
 => [auth] library/eclipse-temurin:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM docker.io/library/eclipse-temurin:21-jdk-jammy@sha256:adb9b2d15adf1833d9dae0bdc1cff61ef5a804dc58dfbfb34269f32432b2e5d
 => => resolve docker.io/library/eclipse-temurin:21-jdk-jammy@sha256:adb9b2d15adf1833d9dae0bdc1cff61ef5a804dc58dfbfb34269f32432b2e5d
 => => sha256:adb9b2d15adf1833d9dae0bdc1cff61ef5a804dc58dfbfb34269f32432b2e5dc 5.25kB / 5.25kB
 => => sha256:7db733aadd907ea0a5168163d1f0df57e2f611cfa879836878e0872d4e6476dc 1.94kB / 1.94kB
 => => sha256:89730bf19567dbaf708ae2cb66e0c5eaceb60f2f1311b7e5ab97524cec7bba41 5.76kB / 5.76kB
 => => sha256:af6eca94c8104c8e90d3f9efe59c2b3a02b20aad3d985e31c7cd009ea104c447 0B / 29.54MB
```

```
=> => exporting layers
=> => writing image sha256:20bd592d6a6d377bd4b5e721fc33679838969b0c4eaf18223e42be0189e71413
=> => naming to docker.io/amarabalkis/democyber:latest

View build details: docker-desktop://dashboard/build/default/default/kkyzwxdsyuklp1znoimio3rg6

What's Next?
```

Test the image locally:

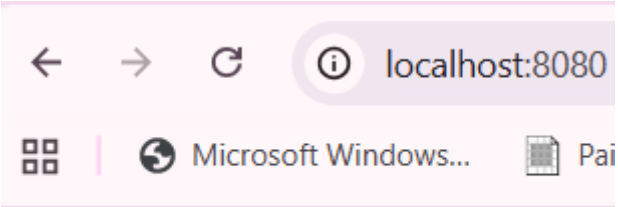
docker run -p 8080:8080 tonusername/democyber:latest

```
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP> docker run -p 8080:8080 amarabalkis/democyber:latest
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.0)
```

Note: -p 8080:8080 → maps port 8080 of the container to your PC

Next, open your browser: http://localhost:8080



Bonjour depuis democyber Spring Boot 123!

2. Push image :

- Log in to Docker Hub

docker login

```
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP> docker login
Authenticating with existing credentials...
Login Succeeded
```

docker push tonusername/democyber:latest

```
7dbbd2a3ab4a: Pushed
767e56ba346a: Pushed
latest: digest: sha256:e7eb7015f8940de2b3d8ae3033a6f6c56b5965d
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP>
```

- Test the image from Docker Hub:

docker run --pull always -p 8080:8080 tonusername/democyber:latest

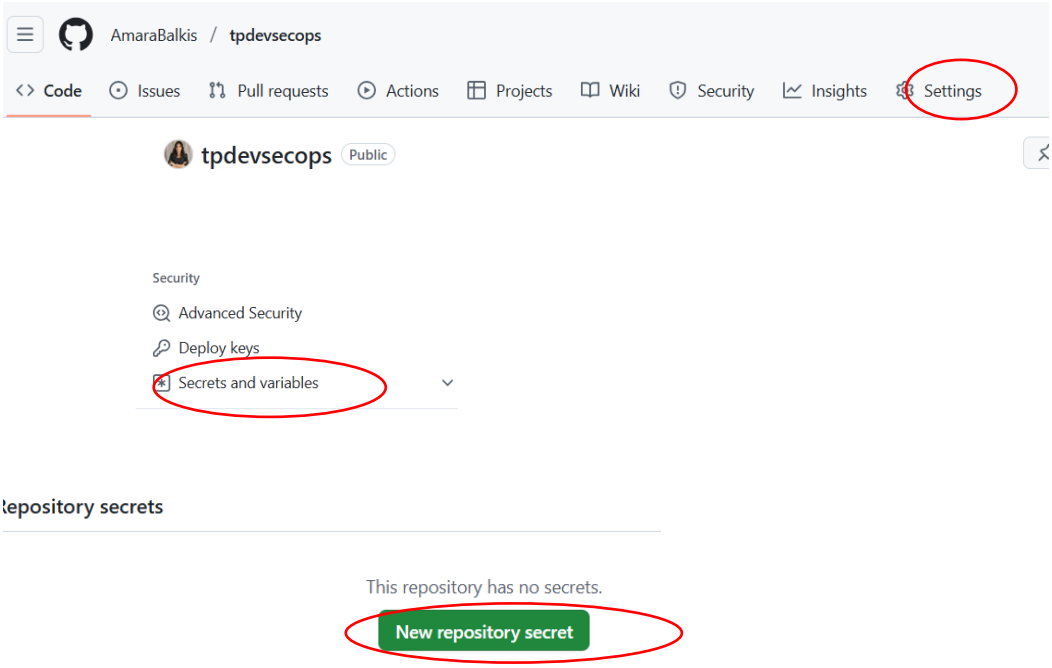
Note: This forces Docker to **download the latest version pushed to the Hub** and run it, even if you already have a local version.

```
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP> docker run --pull always -p 8080:8080
latest: Pulling from balkisamara/democyber
Digest: sha256:e7eb7015f8940de2b3d8ae3033a6f6c56b5965da333d5b3a758bee8325a84741
Status: Image is up to date for balkisamara/democyber:latest
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
```

IV. Basic CI/CD Pipeline

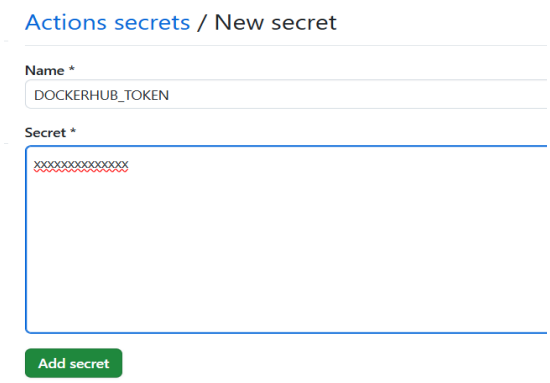
1. Add the token in GitHub Actions

On GitHub → your repository → **Settings** → **Secrets and variables** → **Actions** → **New repository secret**



Add the two secrets:

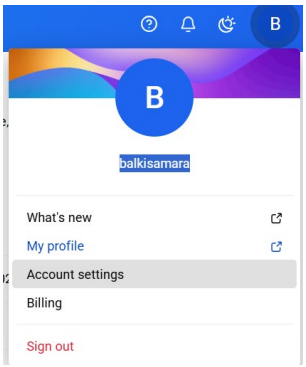
- DOCKERHUB_USERNAME → your Docker Hub username
- DOCKERHUB_TOKEN → your Docker Hub personal access token



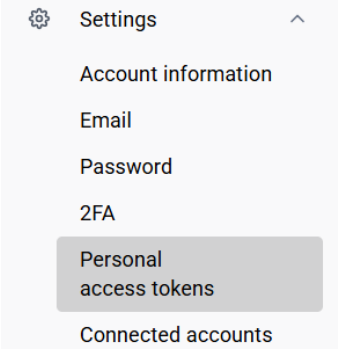
This is a special password that allows GitHub Actions to connect to Docker Hub **without using your real password**.

Steps to create it:

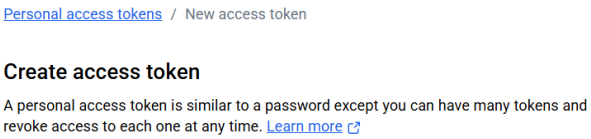
1. Log in to Docker Hub



2. Go to **Personal Access Tokens**

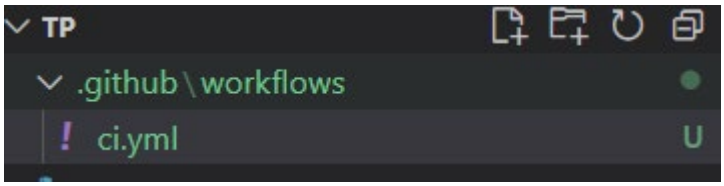


3. **Click New Access Token**



4. Give your token a name (ex : `github-actions`)
5. Click **Generate**
6. Copy the generated token immediately (you won't be able to see it again)

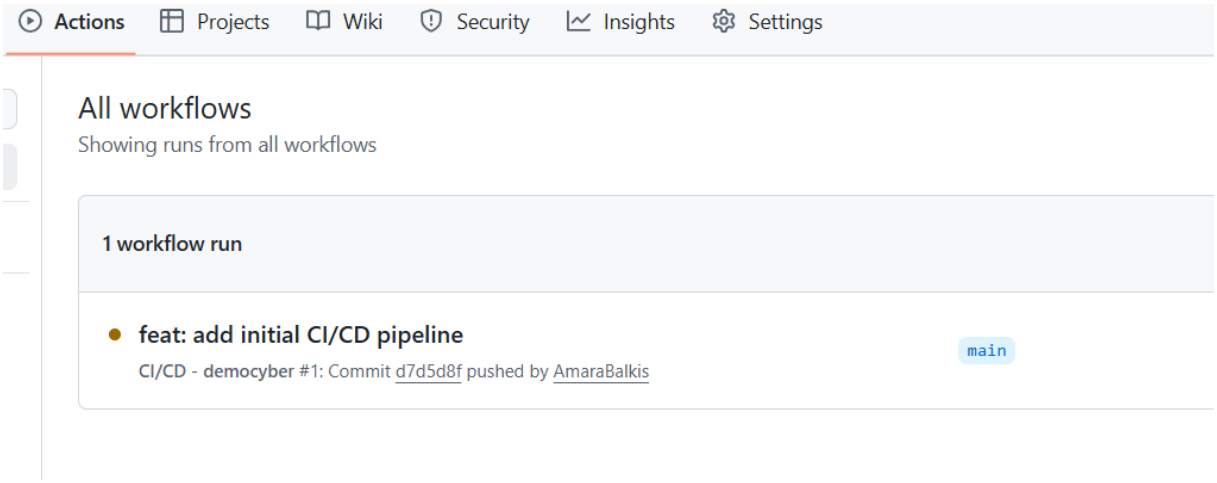
2. Create the GitHub Actions workflow

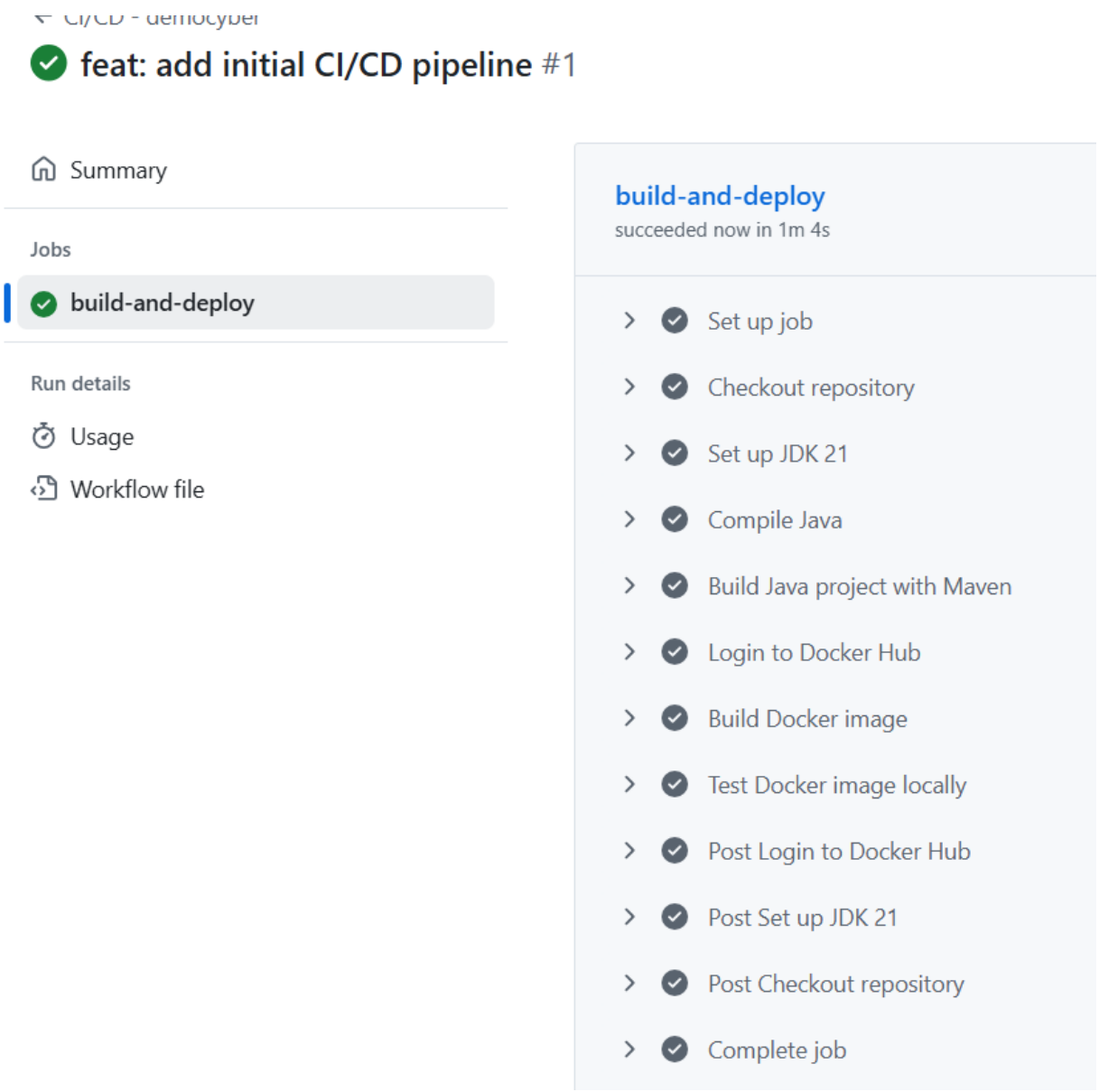


In your local project, create the following folder and file:

.github/ workflows/ ci.yml

```
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP> git add .
PS C:\Users\ASUS\Desktop\EPI\5emecyberA\TP> git commit -m "feat: add initial CI/CD pipeline"
```





V. Add gitleaks :