**Experiment 3,4**

**Maven Build using GitHub Actions**

**Objective: Set up a GitHub Actions workflow to automatically build a Maven project whenever changes are pushed to a GitHub repository.**

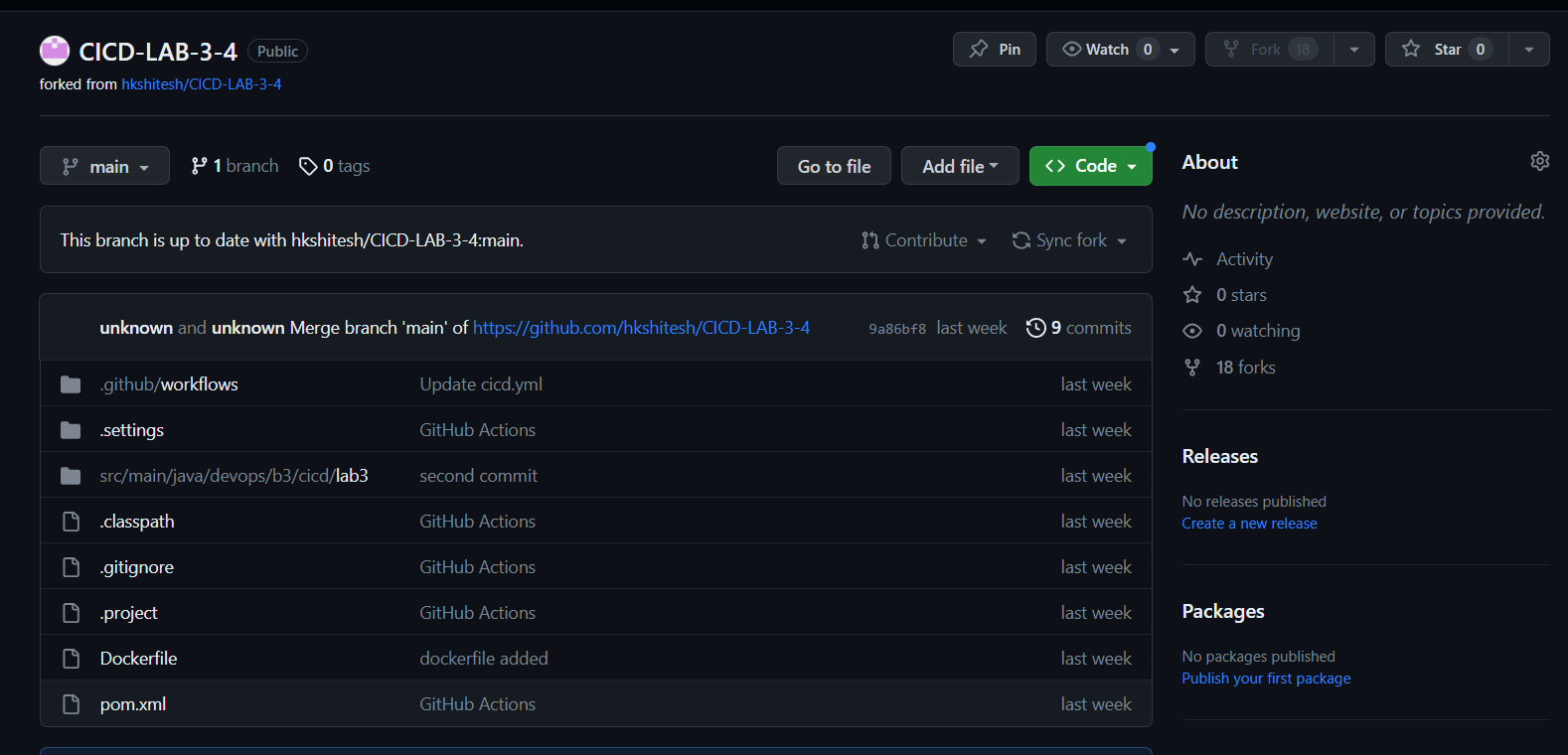
**Prerequisites:**

* GitHub account
* A Maven-based Java project hosted on GitHub

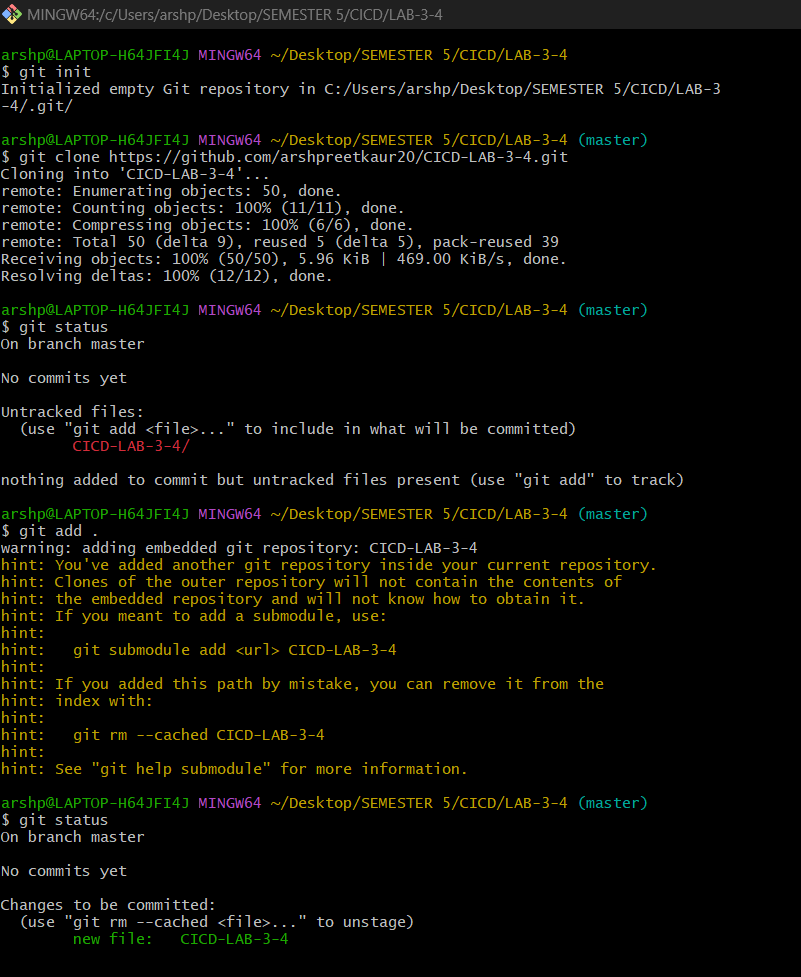
**Exercise Steps:**

**Step 1: Fork and Clone the Repository**

* Fork a sample Maven-based Java project repository on GitHub.



* Clone the forked repository to your local machine.



**Step 2: Create a GitHub Actions Workflow**

* In your cloned repository, create a directory named .github/workflows if it doesn't exist.
* Inside the .github/workflows directory, create a YAML file (e.g., maven-build.yml) to define your GitHub Actions workflow. You can use any text editor to create the file.
* Edit maven-build.yml and add the following content:

name: Maven Build

on:

push:

branches:

- main # Change this to your main branch name

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Java

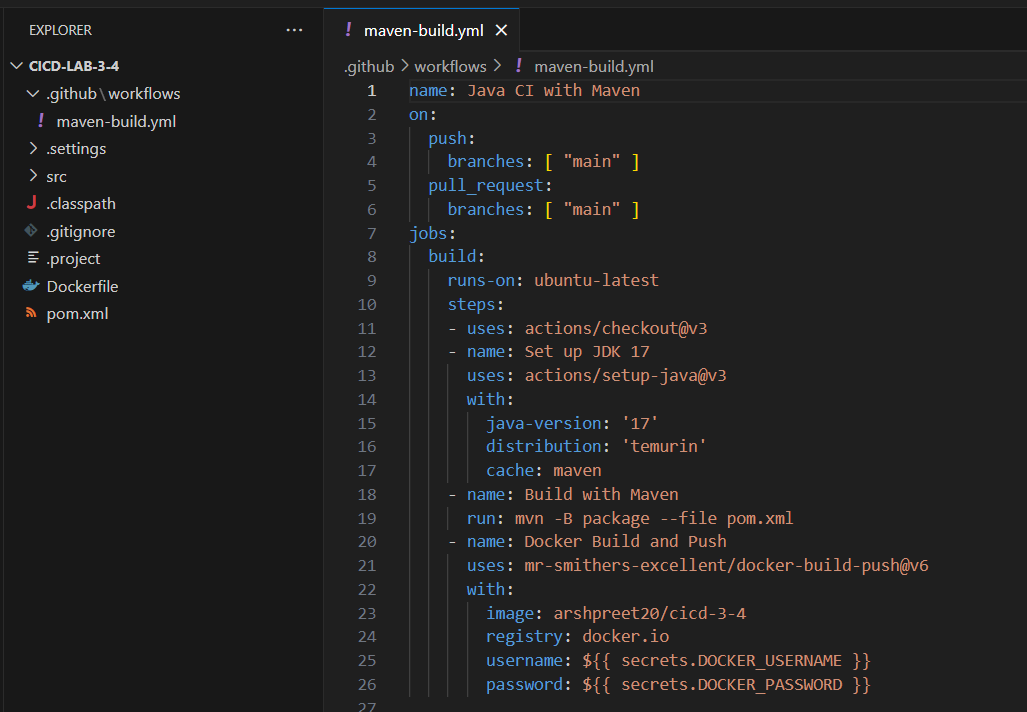
uses: actions/setup-java@v2

with:

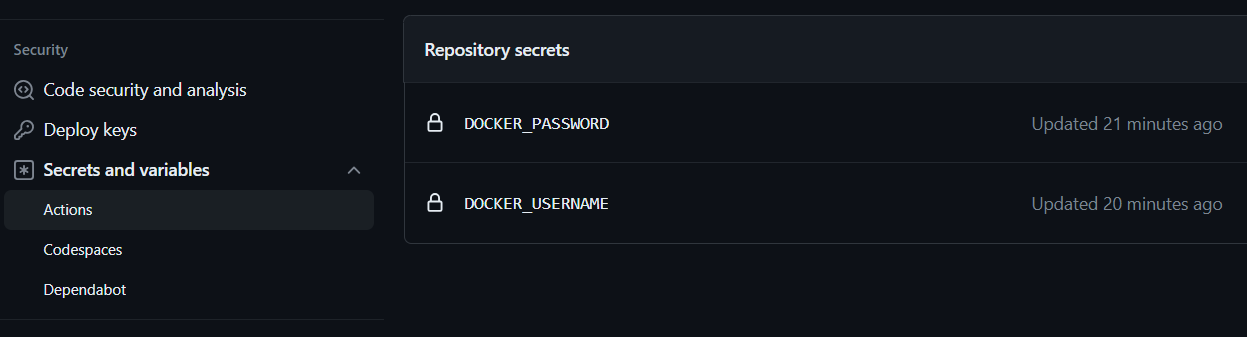
java-version: '11' # Change this to the desired Java version

- name: Build with Maven

run: mvn clean install



* On docker hub make a repo and copy the repo name (image name- see 4th last line of the yml file, that repo name is to be pasted here)
* Go to github Secrets and variables and add a secret for these tw0-
* DOCKER\_USERNAME AND DOCKER\_PASSWORD



This workflow will trigger a Maven build whenever changes are pushed to the main branch.

**Step 3: Commit and Push Changes**

* Save the maven-build.yml file.
* Commit the changes to your local repository:

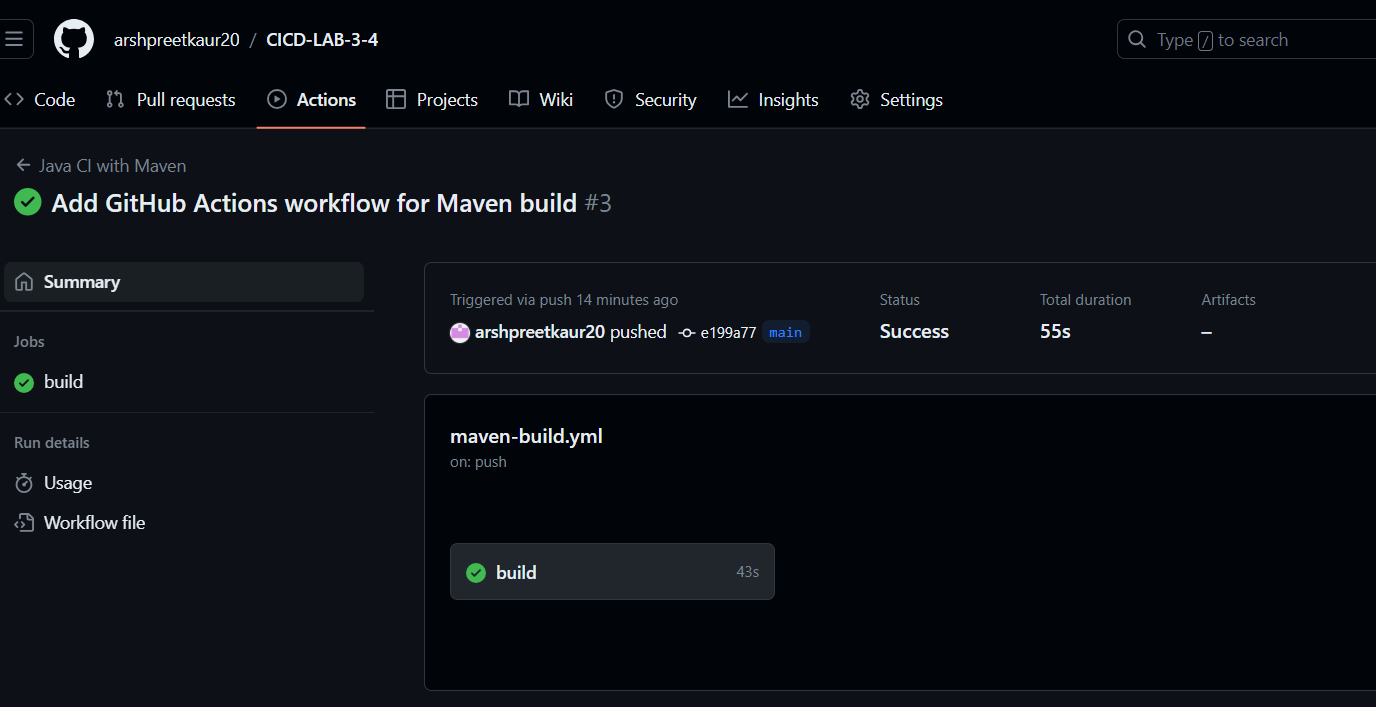
git add .

git commit -m "Add GitHub Actions workflow for Maven build"

git push origin main

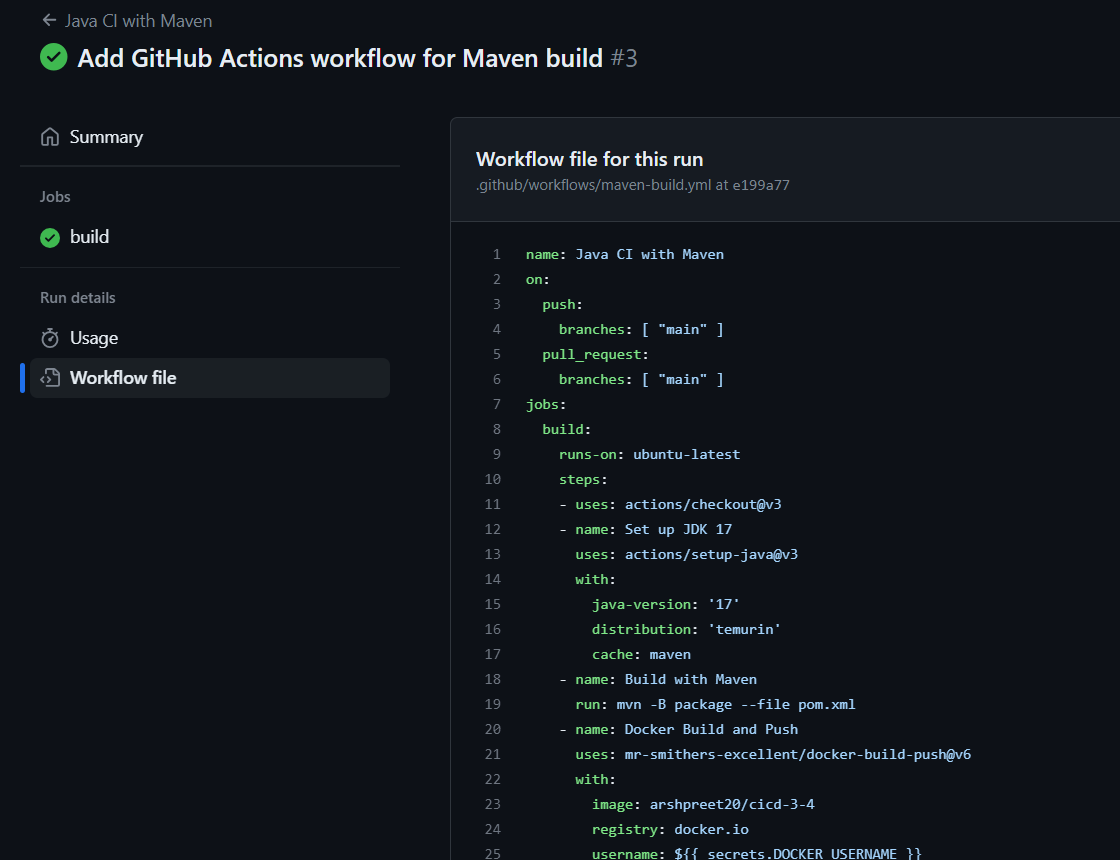
**Step 4: Check the Workflow Status**

* Go to your GitHub repository on the GitHub website.
* Click on the "Actions" tab to see the workflow running. You should see a workflow named "Maven Build" or the name you specified in the YAML file.
* Monitor the workflow's progress, and once it completes successfully, you should see a green checkmark indicating a successful build.



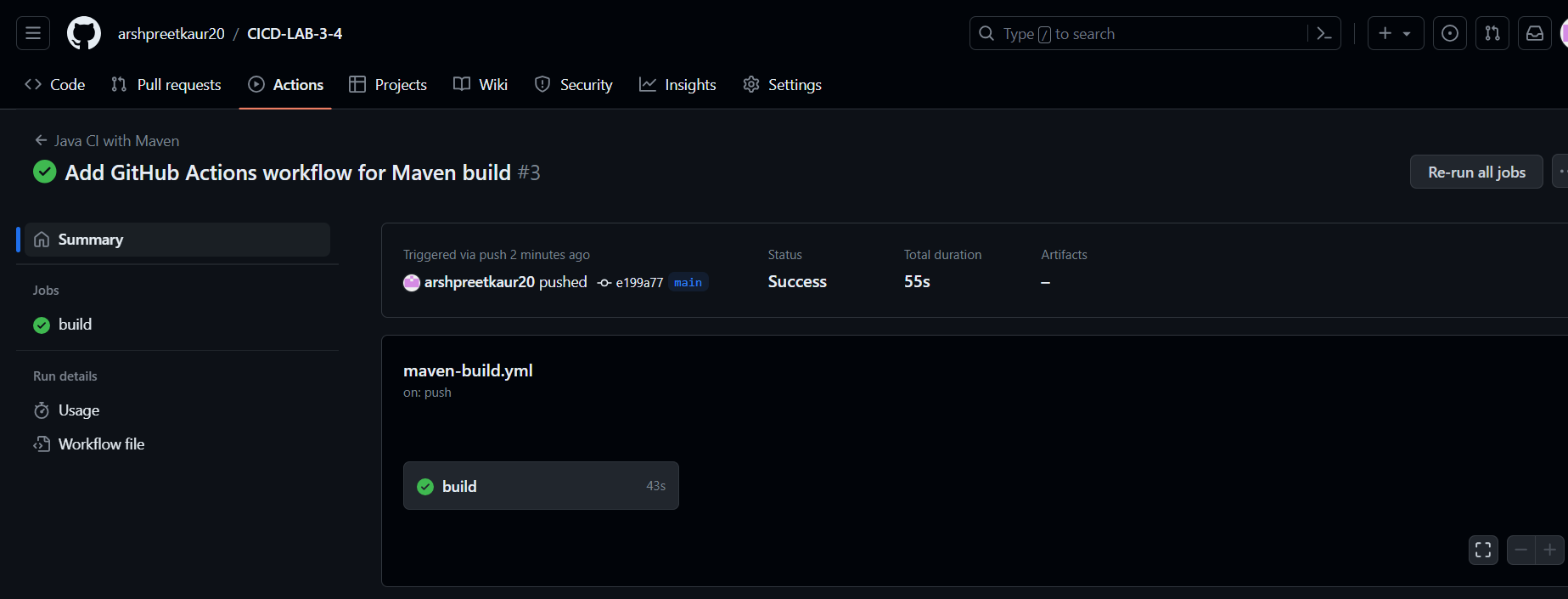
**Step 5: Verify the Build Artifacts**

* If the build was successful, navigate to the "Actions" tab on your GitHub repository, and click on the latest workflow run.
* In the workflow details, you can find the "Artifacts" section. Click on the artifact(s) to download and verify the build artifacts.



**Step 6: Optional - Trigger a Build**

To test the workflow, make a change in your project, commit, and push it to the repository. This should trigger the GitHub Actions workflow automatically.



**Conclusion:**

In this lab exercise, you've set up a GitHub Actions workflow for a Maven-based Java project to automate the build process. Participants should now have a basic understanding of how to integrate CI/CD into their projects using GitHub Actions. You can extend this exercise by adding deployment steps, testing, or other actions to the workflow as needed.