Ex No. **Create GitHub repository for Ci/CD and configure continuous integration with GitHub Actions**

Date:

Aim:

To create GitHub repository for Ci/CD and configure continuous integration with GitHub Actions.

Algorithm:

Step 1:Create a GitHub Repository

Log in to GitHub.

Click on the "+" icon and select New Repository.

Give your repository a name, set visibility (public or private), and create it.

Step 2:Add Your Project Files

Add the necessary project files to your repository (e.g., your code, tests).

Push your files to GitHub.

Step 3: Set Up GitHub Actions

Create a folder called .github/workflows in your repository.

Inside that folder, create a workflow file (usually named with .yml or .yaml extension).

Define the actions to be performed (for example, build the code, run tests, and set up the environment).

Step 4: Commit and Push Changes

Add your changes and push them to GitHub.

The GitHub Actions CI pipeline will automatically trigger on every push to your repository.

Step 5: Monitor the Workflow

Go to the Actions tab in your GitHub repository to see the status of the workflows.

If there are issues, check the logs to identify the problem.

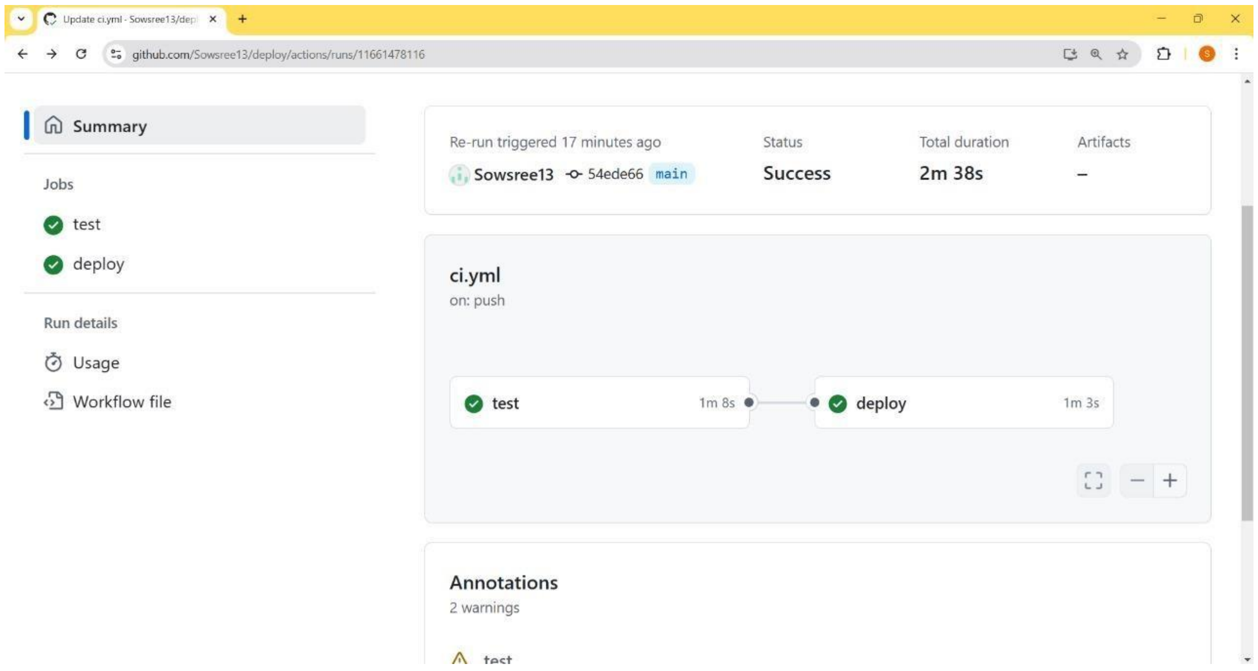
PROGRAM:

```
import unittest from main import
bye, hello class
TestMain(unittest.TestCase):     def
test_hello(self):
        self.assertEqual(hello(), "hi")

    def test_bye(self):
        self.assertEqual(bye(), "bye")



if _name_ == "_main_":
unittest.main()
test_main.py def
hello():     return "hi"



def bye():
return "bye"



print(hello())
```

```
main.py name: CI
Workflow

on:   push:
branches:
- main

jobs:
lint:
   name:   Lint   code   base
runs-on:       ubuntu-latest
steps:
- name: Checkout code        uses:
  actions/checkout@v2

- name: Run Super-Linter        uses:
  github/super-linter@v4        env:
        DEFAULT_BRANCH: main
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

  test:
   name:   Run   Unit   Tests
runs-on:       ubuntu-latest
steps:
- name: Checkout code        uses:
  actions/checkout@v2
```

```
- name: Set up Python        uses:
  actions/setup-python@v2
  with:
        python-version: '3.x'

- name: Run Tests        run: python -
  m unittest discover superlinter.yml
```

OUTPUT:


Update ci.yml - Sowsree13/dep
github.com/Sowsree13/deploy/actions/runs/11661478116
Summary
Jobs
test
deploy
Run details
Usage
Workflow file

Re-run triggered 17 minutes ago — Sowsree13 54ede66 main

Status: Success

Total duration: 2m 38s

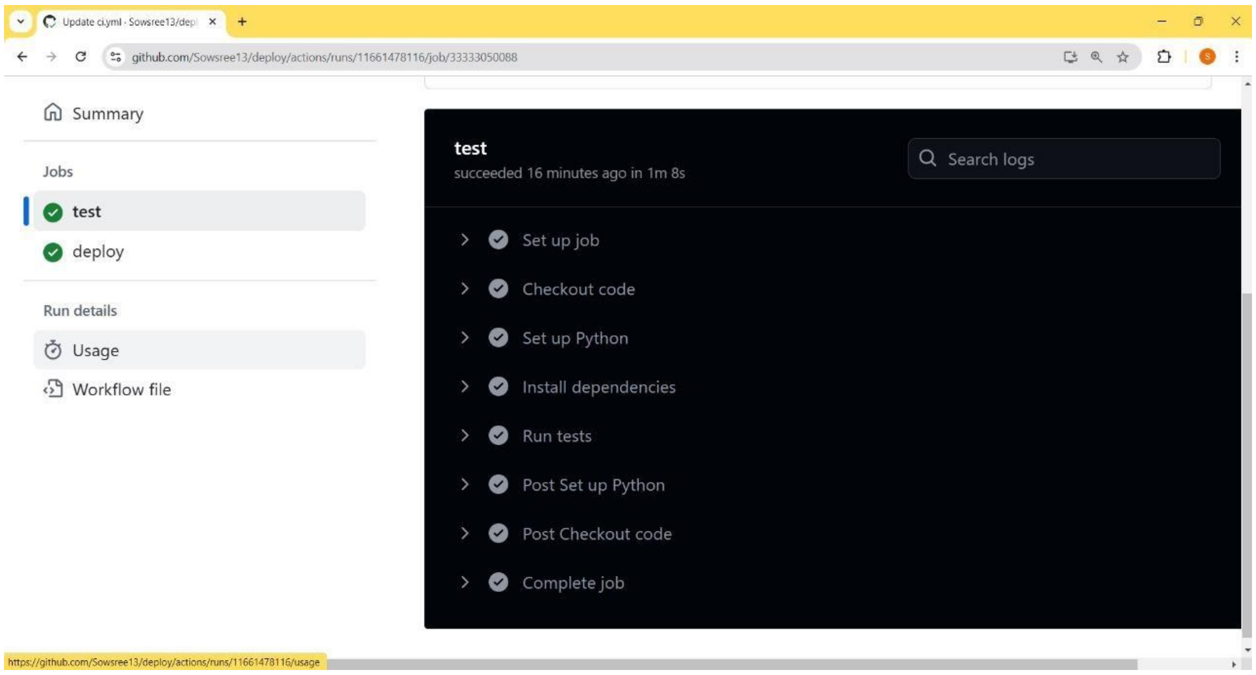
Artifacts: –

ci.yml
on: push

test 1m 8s → deploy 1m 3s

Annotations
2 warnings

test



Update ci.yml · Sowsree13/dep

github.com/Sowsree13/deploy/actions/runs/11661478116/job/33333050088

Summary

Jobs

test

deploy

Run details

Usage

Workflow file

test

succeeded 16 minutes ago in 1m 8s

Search logs

Set up job

Checkout code

Set up Python

Install dependencies

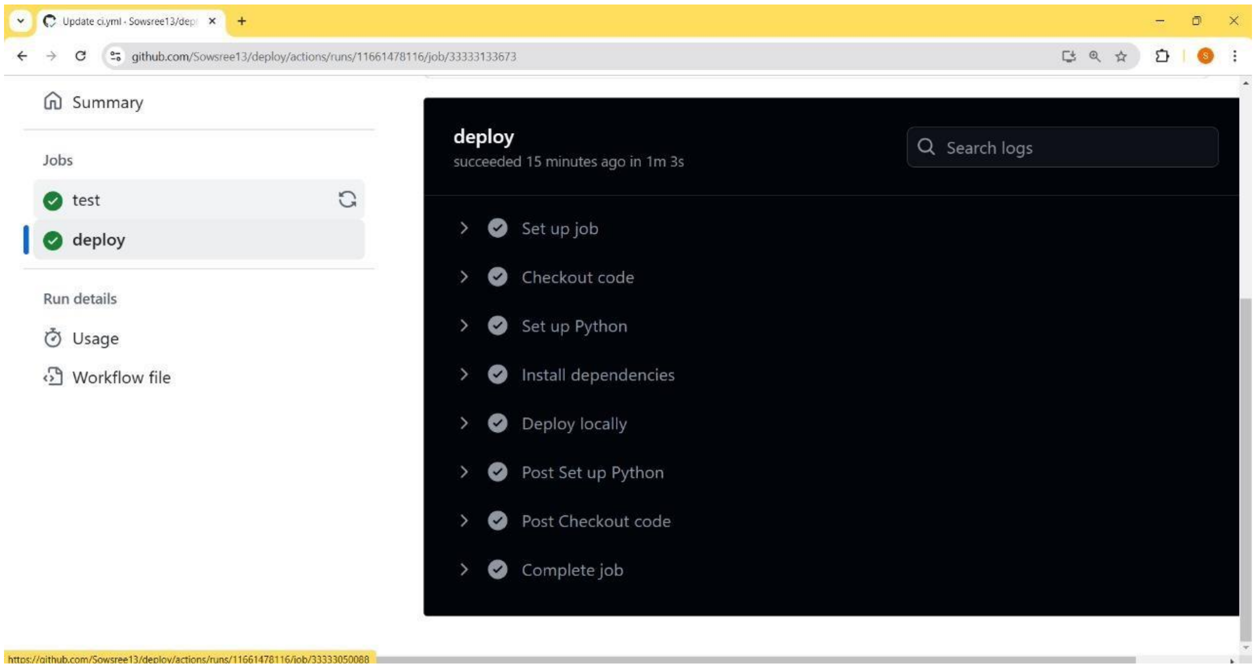
Run tests

Post Set up Python

Post Checkout code

Complete job

https://github.com/Sowsree13/deploy/actions/runs/11661478116/usage



Update ci.yml · Sowsree13/dep

github.com/Sowsree13/deploy/actions/runs/11661478116/job/33333133673

Summary

Jobs

test

deploy

Run details

Usage

Workflow file

deploy

succeeded 15 minutes ago in 1m 3s

Search logs

Set up job

Checkout code

Set up Python

Install dependencies

Deploy locally

Post Set up Python

Post Checkout code

Complete job

https://github.com/Sowsree13/deploy/actions/runs/11661478116/job/33333050088

RESULT:

The experiment to Create GitHub repository for Ci/CD and configure continuous integration with GitHub Actions is implemented successsfully.