

CONTAINER ORCHESTRATION AND INFRASTRUCTURE AUTOMATION

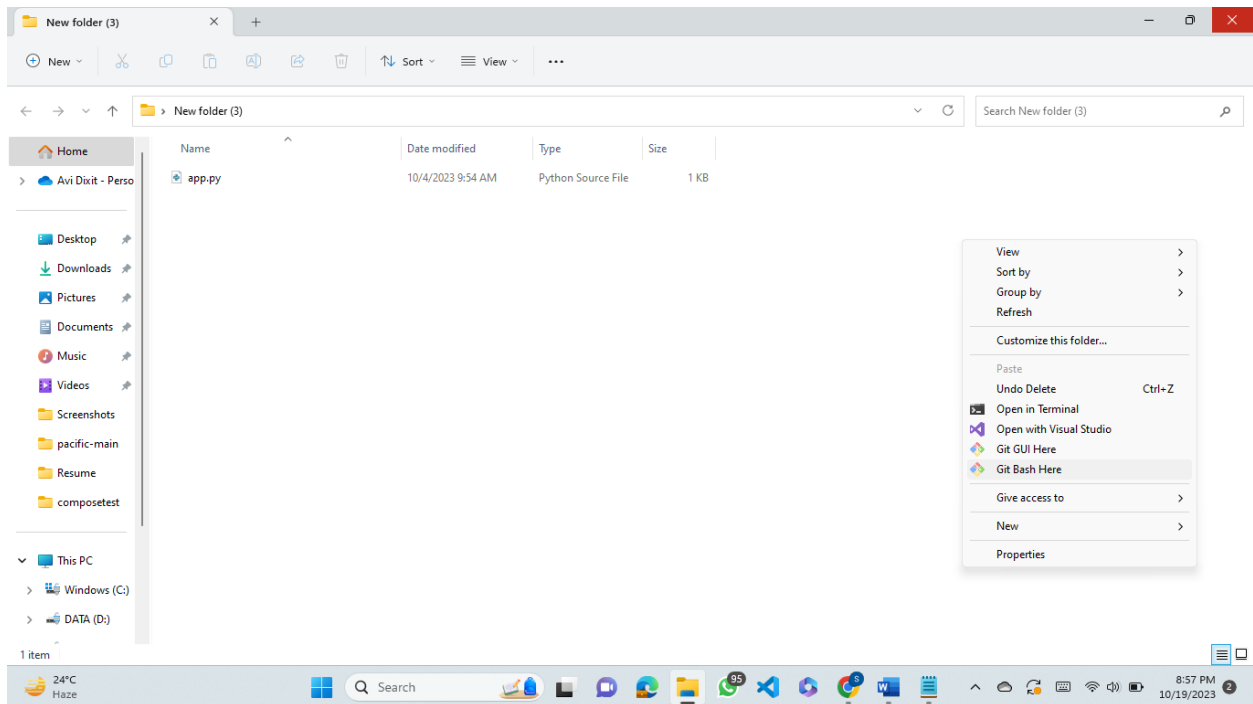
NAME- Shubhi Dixit

SAP ID- 500094571

BATCH- 05

CI/CD Pipeline

Step 1: Go to your project directory and open git bash in the directory.



Step 2: Initialise an empty git repository in the terminal using the command:
`git init`

Check the status of your current files using the command:
`git status`

```
MINGW64/c:/Users/admin/Desktop/New folder (3)

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git init
Initialized empty Git repository in C:/Users/admin/Desktop/New folder (3)/.git/

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  app.py

nothing added to commit but untracked files present (use "git add" to track)

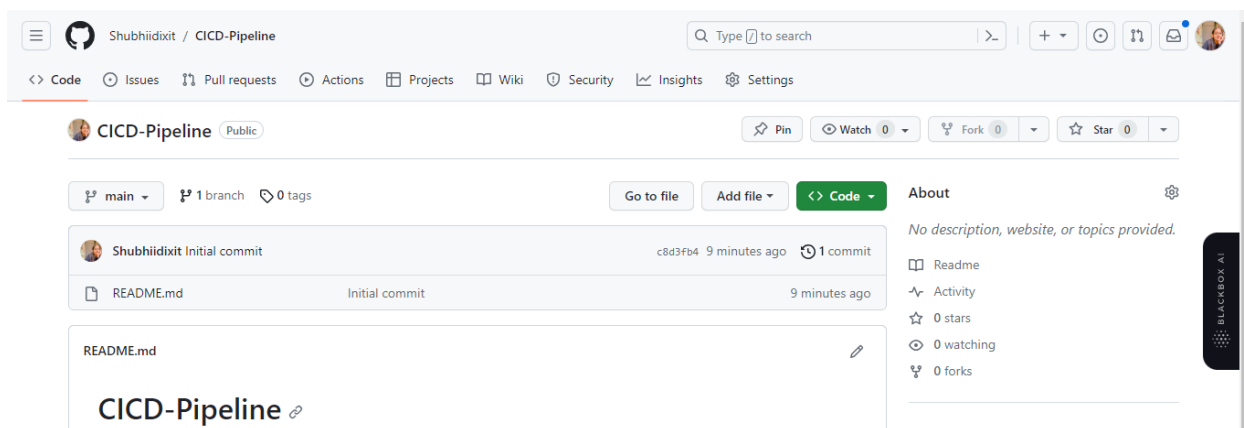
Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git add .

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$
```

Step 3: Now before committing the changes, add the files to the staging area using the command:
`git add .`

Now commit the changes:
`git commit -m "initial commit"`

Step 3: Go to your github account and create a repository for the project.



Step 4: After creating the repository, copy the repository url from the “code” section (highlighted in green) and go back to the git bash terminal.

Step 5: Run the command:
`git remote add origin "paste the url that you copied"`

Step 6: Push the code files to the github

git push -u origin master

```
MINGW64/c/Users/admin/Desktop/New folder (3)
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  app.py

nothing added to commit but untracked files present (use "git add" to track)

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git add .

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git commit -m "initial commit"
[master (root-commit) e8821bf] initial commit
1 file changed, 22 insertions(+)
create mode 100644 app.py

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git add remote origin https://github.com/Shubhiidixit/CICD-Pipeline.git
fatal: pathspec 'remote' did not match any files

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git remote add origin https://github.com/Shubhiidixit/CICD-Pipeline.git

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$
```

```
MINGW64/c/Users/admin/Desktop/New folder (3)
$ git add remote origin https://github.com/Shubhiidixit/CICD-Pipeline.git
fatal: pathspec 'remote' did not match any files

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git remote add origin https://github.com/Shubhiidixit/CICD-Pipeline.git

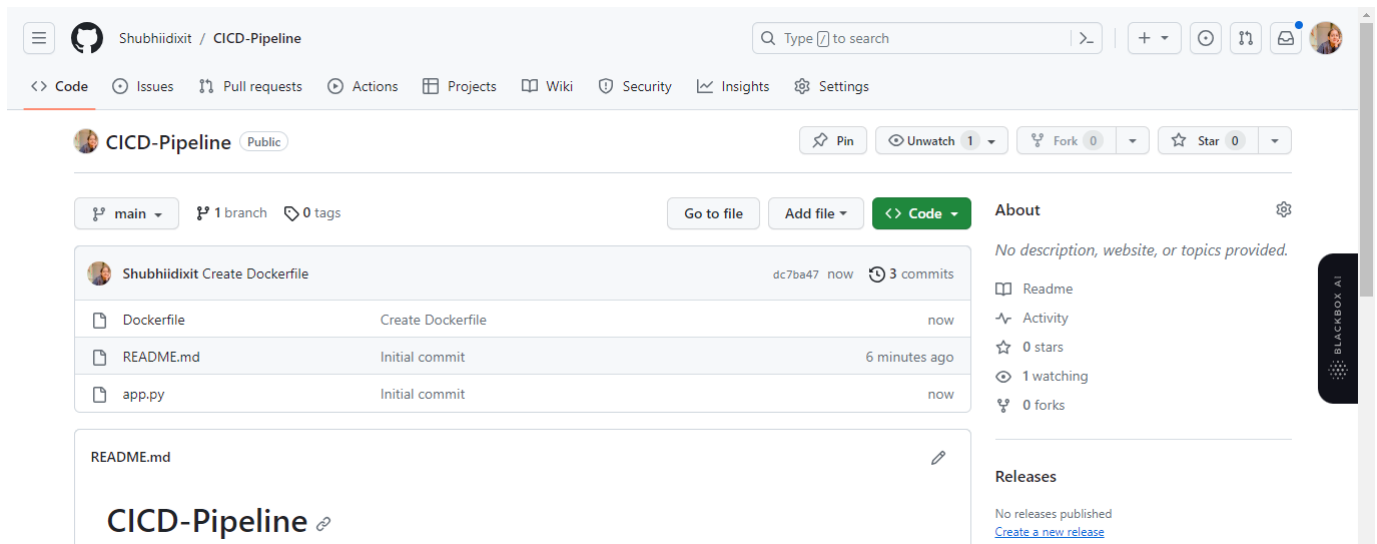
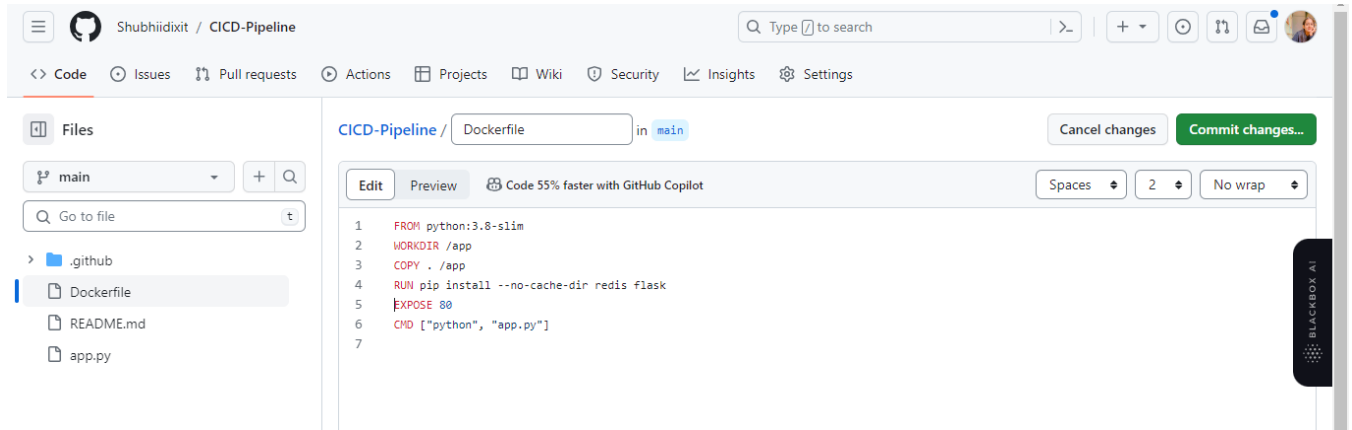
Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 501 bytes | 125.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/Shubhiidixit/CICD-Pipeline/pull/new/master
remote:
To https://github.com/Shubhiidixit/CICD-Pipeline.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Shubhi@Shubhi MINGW64 ~/Desktop/New folder (3) (master)
$ |
```

Go to your repository, refresh the page and check whether the files are uploaded or not.

Step 7: Now go to “**Add file**” and then “**Create new file**” option
Create a dockerfile and include the following instructions:

```
FROM node:latest
WORKDIR /usr/src/app
COPY package.json ./
RUN npm install
COPY . .
```

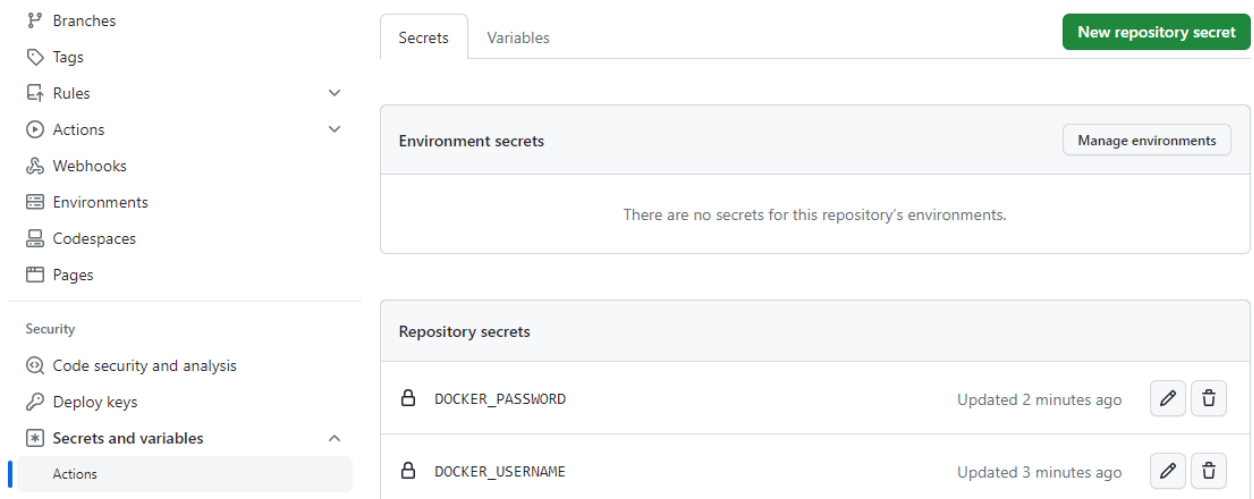


After making the dockerfile commit changes in it and save.

Step 8: Go to “**Settings**” and go to “**Secret and Variables**” in the left panel. Choose the “**Actions**” and then click on “**New Repository Secret**”. Add your docker hub username and passwords in the repository secret with the name:

DOCKER_USERNAME

DOCKER_PASSWORD



Step 9: Go to actions and click on “**set up a workflow yourself**” highlighted in blue at the top.

A **main.yaml** file will be opened. Inside the **main.yaml** file, copy the following code:

name: Publish Docker image

on:

push:

branches: ['master']

jobs:

push_to_registry:

name: Push Docker image to Docker Hub

runs-on: ubuntu-latest

steps:

- name: Check out the repo

uses: actions/checkout@v3

- name: Log in to Docker Hub

uses: docker/login-action@f054a8b539a109f9f41c372932f1ae047eff08c9

with:

username: \${{ secrets.DOCKER_USERNAME }}

password: \${{ secrets.DOCKER_PASSWORD }}

- name: Extract metadata (tags, labels) for Docker

id: meta

uses: docker/metadata-action@98669ae865ea3cfffcbcaa878cf57c20bbf1c6c38

with:

images: shubhid/testpython

- name: Build and push Docker image

uses: docker/build-push-action@ad44023a93711e3deb337508980b4b5e9bc5dc

with:

context: .

push: true

file: ./Dockerfile

tags: \${{ steps.meta.outputs.tags }}

labels: \${{ steps.meta.outputs.labels }}

CICD-Pipeline / .github / workflows / main.yml in main

Edit

Preview

Code 55% faster with GitHub Copilot

Spaces

2

No wrap

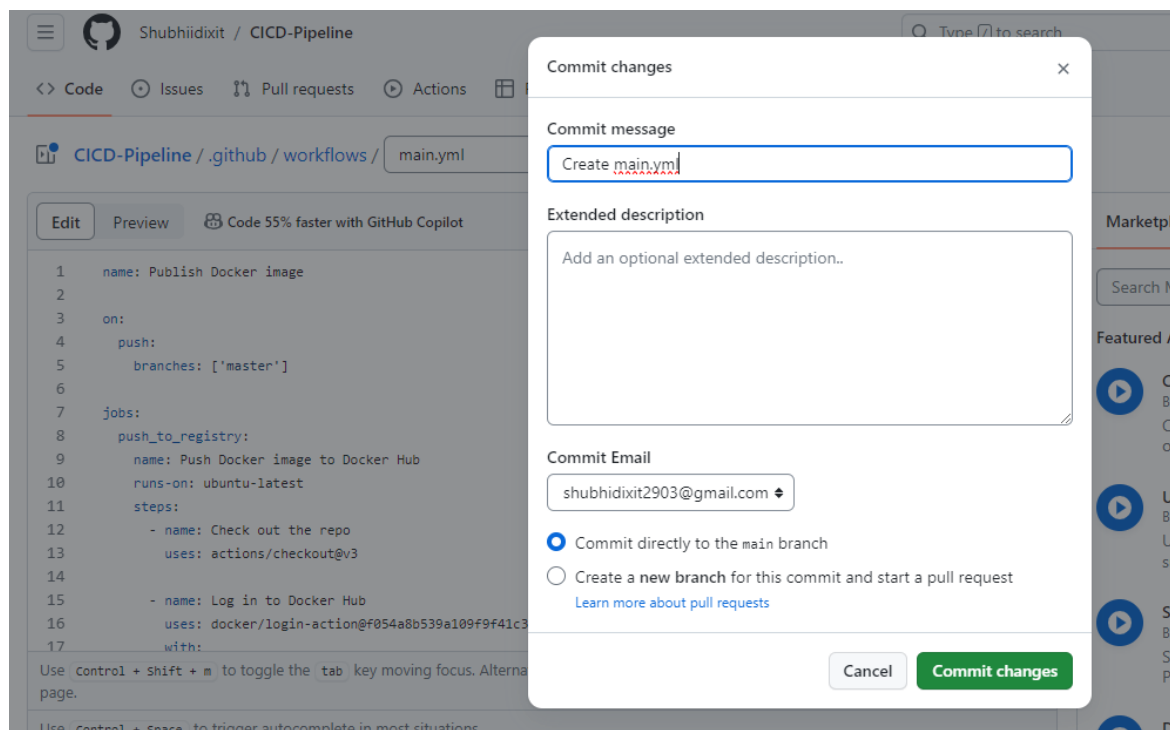
⌂

```
1  name: Publish Docker image
2
3  on:
4    push:
5      branches: ['main']
6
7  jobs:
8    push_to_registry:
9      name: Push Docker image to Docker Hub
10     runs-on: ubuntu-latest
11     steps:
12       - name: Check out the repo
13         uses: actions/checkout@v3
14
15       - name: Log in to Docker Hub
16         uses: docker/login-action@f054a8b539a109f9f41c372932f1ae047eff08c9
17         with:
```

CICD-Pipeline / .github / workflows / main.yml in main

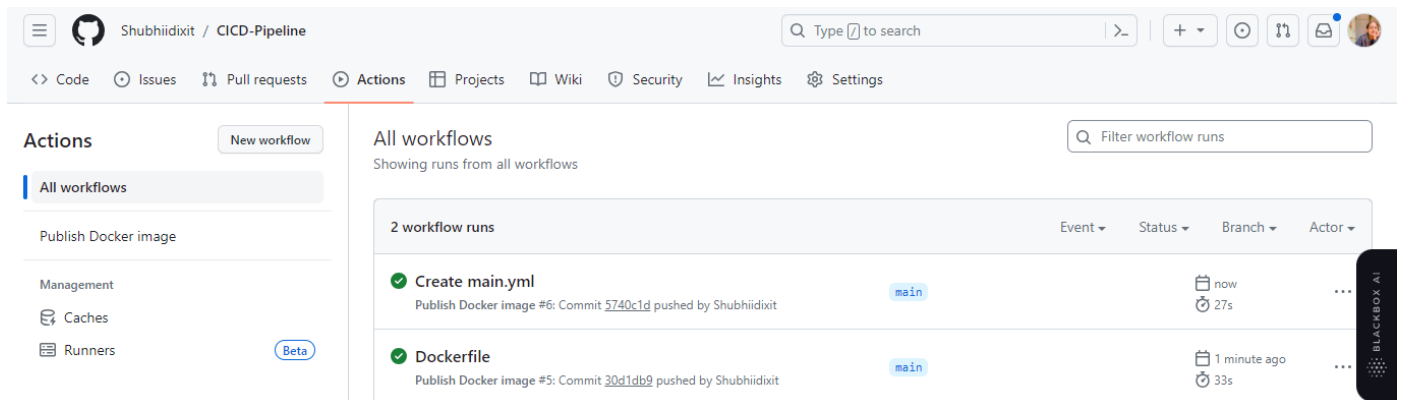
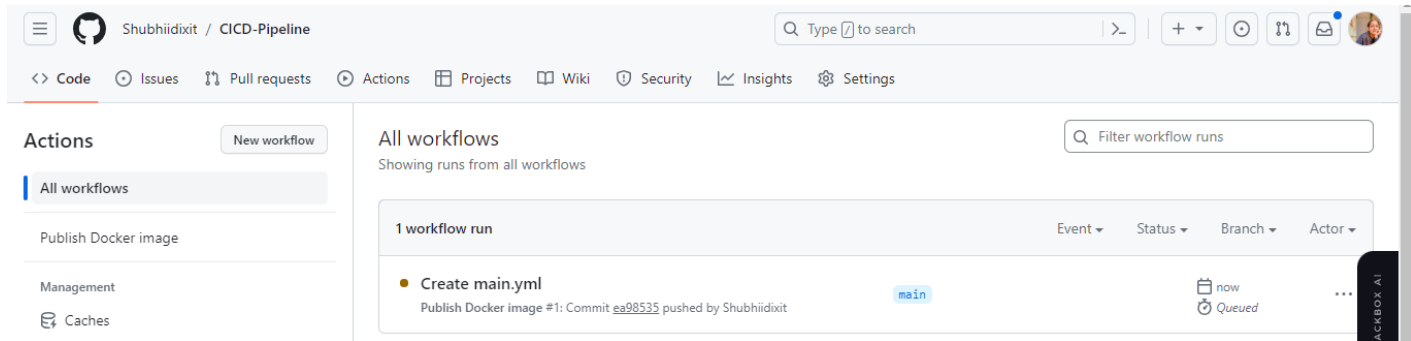
Edit Preview Code 55% faster with GitHub Copilot Spaces 2 No wrap

```
13     uses: actions/checkout@v3
14
15     - name: Log in to Docker Hub
16       uses: docker/login-action@f054a8b539a109f9f41c372932f1ae047eff08c9
17       with:
18         username: ${ secrets.DOCKER_USERNAME }
19         password: ${ secrets.DOCKER_PASSWORD }
20
21     - name: Extract metadata (tags, labels) for Docker
22       id: meta
23       uses: docker/metadata-action@98669ae865ea3cfffcbcaa878cf57c20bbf1c6c38
24       with:
25         images: ShubhiD/testpython
26
27     - name: Build and push Docker image
28       uses: docker/build-push-action@ad44023a93711e3deb337508980b4b5e9bdc5dc
29       with:
30         context: .
31         push: true
32         file: ./Dockerfile
33         tags: ${ steps.meta.outputs.tags }
34         labels: ${ steps.meta.outputs.labels }
```



Step 10: After adding the code, commit the changes in the main.yml file.

Step 11: Go to “**Actions**” and see whether the file is being created or not. If yes, then the instructions that are included in the yaml file will be executed and the image will be pushed to the docker hub.



Step 12: Login to your docker hub and check whether the image is pushed or not.

