|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Analytical Model** | **Model Lab** | **University Lab** | **Model Theory** | **University Theory** |
| **17-12-2024** | **18-12-2024** | **19-12-2024** | **20-12-2024** | **21-12-2024** |

1. Create new repo in github and clone in vs code
2. Create 3 files

**Calc.py**

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b == 0:

        raise ValueError("Cannot divide by zero!")

    return a / b

def calculator():

    while True:

        print("\nSimple Calculator")

        print("1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Exit")

        try:

            choice = int(input("Enter choice (1/2/3/4/5): "))

            if choice == 5:

                print("Exiting the calculator. Goodbye!")

                break

            a = float(input("Enter first number: "))

            b = float(input("Enter second number: "))

            if choice == 1:

                print(f"Result: {add(a, b)}")

            elif choice == 2:

                print(f"Result: {subtract(a, b)}")

            elif choice == 3:

                print(f"Result: {multiply(a, b)}")

            elif choice == 4:

                print(f"Result: {divide(a, b)}")

            else:

                print("Invalid choice! Please select a valid option.")

        except ValueError as e:

            print(f"Error: {e}. Please enter valid inputs.")

        except Exception as e:

            print(f"An unexpected error occurred: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    calculator()

**dockerfile**

FROM python:3.8-slim

WORKDIR /docdir

COPY . .

RUN pip install pytest

CMD ["python", "calc.py"]

**Test.py**

import pytest

from unittest.mock import patch

from calc import add, subtract, multiply, divide, calculator

def test\_add():

    assert add(5, 3) == 8

    assert add(-1, -1) == -2

    assert add(0, 0) == 0

def test\_subtract():

    assert subtract(10, 5) == 5

    assert subtract(-1, 1) == -2

    assert subtract(0, 0) == 0

def test\_multiply():

    assert multiply(4, 5) == 20

    assert multiply(-2, 3) == -6

    assert multiply(0, 10) == 0

def test\_divide():

    assert divide(10, 2) == 5

    assert divide(-6, 3) == -2

    with pytest.raises(ValueError, match="Cannot divide by zero!"):

        divide(10, 0)

@patch("builtins.input", side\_effect=["1", "10", "5", "5"])  # Add option

@patch("builtins.print")

def test\_calculator\_add(mock\_print, mock\_input):

    calculator()

    mock\_print.assert\_any\_call("Result: 15.0")

@patch("builtins.input", side\_effect=["2", "15", "5", "5"])  # Subtract option

@patch("builtins.print")

def test\_calculator\_subtract(mock\_print, mock\_input):

    calculator()

    mock\_print.assert\_any\_call("Result: 10.0")

@patch("builtins.input", side\_effect=["3", "3", "4", "5"])  # Multiply option

@patch("builtins.print")

def test\_calculator\_multiply(mock\_print, mock\_input):

    calculator()

    mock\_print.assert\_any\_call("Result: 12.0")

@patch("builtins.input", side\_effect=["4", "20", "4", "5"])  # Divide option

@patch("builtins.print")

def test\_calculator\_divide(mock\_print, mock\_input):

    calculator()

    mock\_print.assert\_any\_call("Result: 5.0")

@patch("builtins.input", side\_effect=["4", "10", "0", "5"])  # Divide by zero

@patch("builtins.print")

def test\_calculator\_divide\_by\_zero(mock\_print, mock\_input):

    calculator()

    mock\_print.assert\_any\_call("Error: Cannot divide by zero!. Please enter valid inputs.")

@patch("builtins.input", side\_effect=["5"])  # Exit option

@patch("builtins.print")

def test\_calculator\_exit(mock\_print, mock\_input):

    calculator()

    mock\_print.assert\_any\_call("Exiting the calculator. Goodbye!")

**in terminal, check test cases by typing below command**

pytest test.py

pytest test.py - -verbosity=2 /// to check what error

now we need to add,commit and push either using vs code or commands in terminal

Now need to yml (template) file.

Go to github  
in newly created repo – select “Action tab”

Search workflow – search docker

Select dockerimage and click configure

**.yml file**

name: Docker Image CI/CD

on:

push:

branches: [ "main" ]

jobs:

build-and-push:

runs-on: ubuntu-latest

steps:

# Step 1: Check out the repository

- name: Check out the code

uses: actions/checkout@v4

# Step 2: Set up Python

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.9' # Adjust to your Python version

# Step 3: Install dependencies

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install pytest

# Step 4: Run the tests using pytest

- name: Run tests with pytest

run: pytest test.py

# Step 5: Log in to Docker Hub

- name: Log in to Docker Hub

uses: docker/login-action@v2

with:

username: "dharanijashu"

password: "Journal@25"

# Step 6: Build the Docker image

- name: Build the Docker image

run: docker build . --file Dockerfile --tag dharanijashu/myimages:latest

# Step 7: Push the Docker image to Docker Hub

- name: Push the Docker image

run: docker push dharanijashu/myimages:latest

**commit changes by click the button**

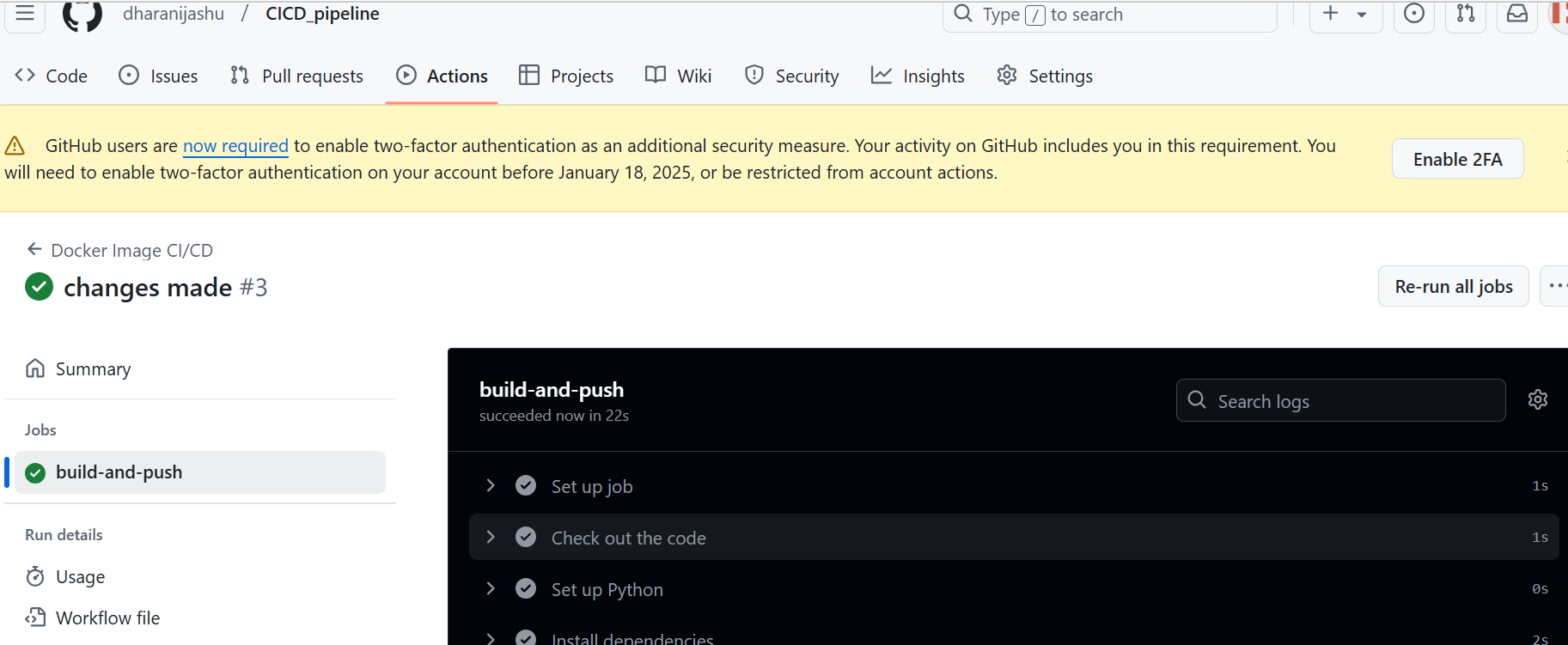
Now we need pull yml file in vs code

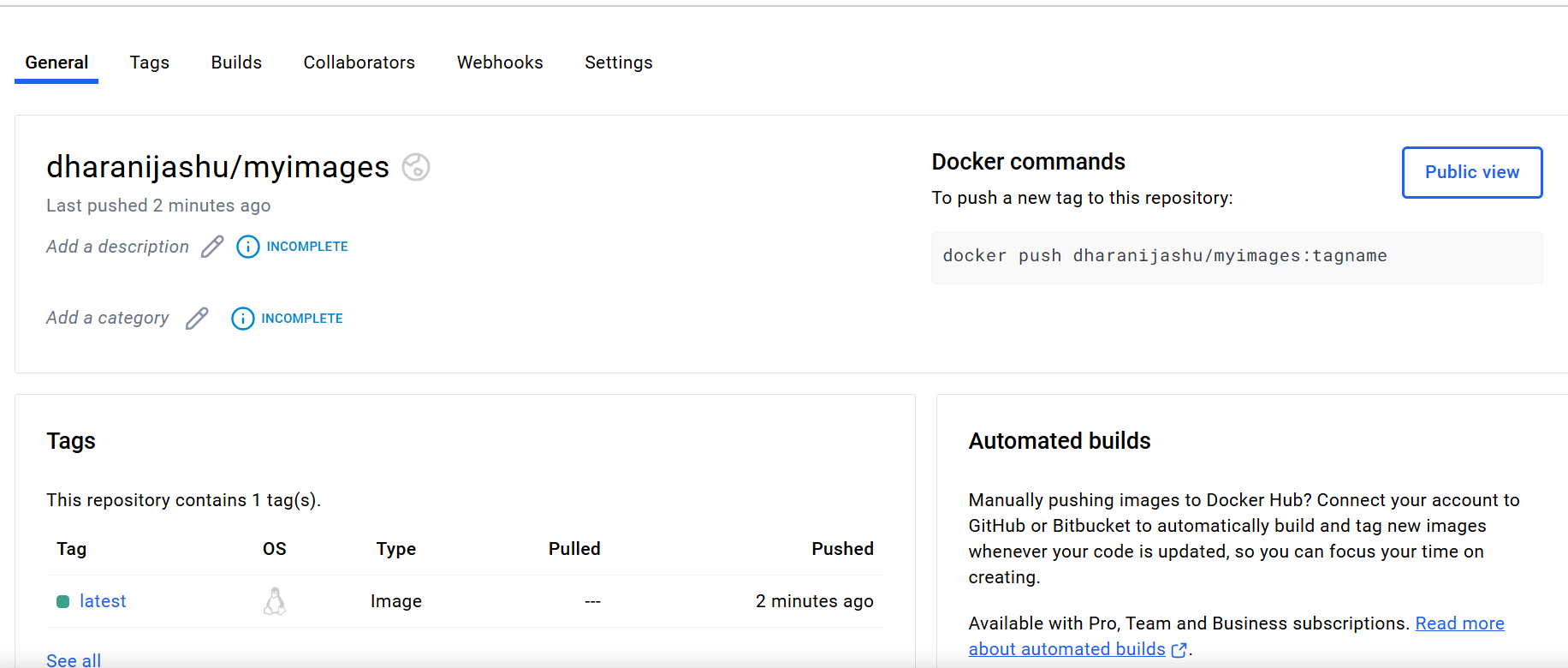
>Git pull

//you can see in vs cod .git branch

Make any changes in the cde

Then perform add, commit , push





docker pull dharanijashy/myimages:latest

docker run -it dharanijashu/myimages:latest