**Explanation of CI Test and Coverage GitHub Actions Workflow**

This GitHub Actions YAML file defines a **CI (Continuous Integration) Test and Coverage** workflow that automatically runs when code is pushed to a specific branch (test\_branch). It includes steps to check out the code, set up Python, run tests, generate a test coverage report, and upload an HTML report. Here’s a breakdown of each section and step:

### 1. ****Trigger on Push to Specific Branch****

on:

push:

branches:

- test\_branch # Trigger on pushes to test\_branch

* This defines the event that will trigger the workflow. The workflow is set to run when code is pushed to the branch named test\_branch.

### 2. ****Define Job:**** test

jobs:

test:

runs-on: ubuntu-latest

* The test job runs on the ubuntu-latest environment. It will contain the steps required to run the tests and generate the coverage report.

### 3. ****Steps:****

Each step under the test job is executed in sequence.

#### a. ****Checkout Code****

steps:

- name: Checkout Code

uses: actions/checkout@v3

* This step checks out the repository’s code to the GitHub runner, allowing the subsequent steps to access the codebase.

#### b. ****Set up Python Environment****

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.9'

* This step sets up Python 3.9 on the runner. It allows you to specify which version of Python to use for running your tests and scripts.

#### c. ****Install Dependencies****

- name: Install Dependencies

run: |

python -m pip install --upgrade pip

pip install -r requirements.txt

pip install pytest pytest-cov pytest-html

* This step installs the Python dependencies needed to run the tests. It first upgrades pip, then installs the dependencies from requirements.txt, followed by the installation of pytest, pytest-cov (for coverage), and pytest-html (for generating HTML reports).

#### d. ****Run Tests and Generate Coverage****

- name: Run Tests and Generate Coverage

run: |

coverage run -m pytest # Run tests and collect coverage

coverage report # Output coverage in the terminal

coverage html # Generate HTML report

* This step runs the test suite using pytest and measures test coverage using coverage. It includes:
  + coverage run -m pytest: Runs the test suite and collects coverage data.
  + coverage report: Outputs the test coverage report in the terminal.
  + coverage html: Generates an HTML version of the coverage report, stored in the htmlcov directory.

#### e. ****Check Coverage Percentage****

- name: Check Coverage

id: coverage-check

run: |

coverage\_percentage=$(coverage report | tail -n 1 | awk '{print $4}' | sed 's/%//')

echo "Coverage percentage: $coverage\_percentage"

echo "::set-output name=coverage\_percentage::$coverage\_percentage" shell: bash

* This step extracts the coverage percentage from the coverage report:
  + coverage report | tail -n 1 | awk '{print $4}' | sed 's/%//': Retrieves the percentage of code covered by the tests from the last line of the coverage report.
  + It prints the coverage percentage and sets it as an output (coverage\_percentage) to be used in later steps or conditions.

#### f. ****Upload HTML Coverage Report****

- name: Upload HTML Coverage Report

uses: actions/upload-artifact@v4

with:

name: coverage-report

path: htmlcov/index.html # Path to the coverage HTML report

* This step uploads the HTML coverage report as an artifact. It takes the index.html file from the htmlcov directory (generated in the previous step) and makes it available for download in the GitHub Actions interface.

### What the Workflow Does

1. **Automatically triggered on code push** to test\_branch.
2. **Runs tests** using pytest.
3. **Generates a coverage report** (both in the terminal and as an HTML file).
4. **Uploads the HTML report** as an artifact, allowing for further inspection.
5. **Extracts the coverage percentage** and sets it as an output that can be used in further conditional steps (though in this case, no further steps use it).