

Learn Git & GitHub x (9) Licenses & x Entry on C++ E x [Entry] C++: E x C++ | Codecademy x Kornawin/do x +

codecademy.com/courses/learn-git/articles/github-actions-tutorial-on-automated-testing

Apps ติดตั้ง JDK (Ja... Backtesting A... MATLAB for P... Tableau Server Coursera-Stan... Guide Ragnar... Reading list

 My Home Course Menu Get Unstuck Tools 

GitHub Actions & Tutorial On Automated Testing

In this article, you'll learn about how GitHub Actions streamline the development workflow. You will also follow a tutorial that adds automated testing to a repository.

Introduction

Have you ever developed code that worked perfectly on your local computer but did not behave well in production? Or have you ever worked in a team where everyone promised the final version would work fine but ended up erroring? This is called the "works on my machine" syndrome in software development.

Now imagine we could configure your GitHub repository to automatically run tests to verify the functionality of the codebase after each code change. Well... we can, using GitHub Actions!

GitHub Actions

GitHub Actions is a powerful, advanced capability of GitHub that enables users to define custom and automated workflows triggered on various types of events such as code push or pull request creation. The workflows execute inside a temporary container running in GitHub infrastructure. You can read up on the growing list of supported types of events as well as [more technical details about GitHub Actions](#).

Tutorial: Add Automated Testing To A Repository

In this tutorial project, you will use GitHub Actions to integrate automatic unit tests to a repository. You will fork a public repository that contains a sample bank application and add a configuration to trigger (already written) unit tests in the repository. Try your best to follow along!



Back

Next

Learn Git & Git x (9) Licenses & C x Entry on C++ E x [Entry] C++: E x C++ | Codecademy x Kornkwin/do x + - □ x

codecademy.com/courses/learn-git/articles/github-actions-tutorial-on-automated-testing

Apps ติดตั้ง JDK (Ja... Backtesting A... MATLAB for P... Tableau Server Coursera-Stan... Guide Ragnar... » Reading list

 My Home Course Menu Get Unstuck Tools 

Bank Account Application

The [Bank Account repository](#) contains a simple Python Flask application code to manage the balance of an imaginary checking account. To understand this tutorial, you don't need extensive Python knowledge or experience with the Flask framework. If you're curious though, the [README.md](#) file includes more details about the code.

Run Tests on Code Push

Now that we know what GitHub actions are and how they enhance development workflows, let's continue by adding an action to a repository. Start by forking and then cloning [Codecademy's Bank Account repository](#). Once you clone your forked copy of the repository from your GitHub account onto your local computer, create a new branch:

```
git checkout -b "add-auto-tests"
```

In your new branch, create a new directory and name it `.github`. Note that the dot in the beginning of the directory name is important. This is a keyword known to GitHub.

Then create another directory inside the `.github` directory and name it `workflows`. GitHub looks for the definitions of GitHub Actions inside this directory.

Create a new `.yaml` file inside the directory. Let's name it `unittests.yaml` and paste the following content inside the file. Note that the indentation and spacing are important.

```
name: Continuous Integration
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: 3.10.0
          architecture: x64
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run Tests
        run: python -m pytest
```

Back

Next

Learn Git & Git x (9) Licenses & x Entry on C++ E x [Entry] C++ : E x C++ | Codecademy x Kornkavin/do x + - x

codecademy.com/courses/learn-git/articles/github-actions-tutorial-on-automated-testing

Apps ติดตั้ง JDK (Ja... Backtesting A... MATLAB for P... Tableau Server Coursera-Stan... Guide Ragnar... Reading list

My Home Course Menu

Get Unstuck Tools Tools

The file introduces a new GitHub Action named **Continuous Integration** that is triggered on **push**, meaning that everytime a developer pushes a code to a branch where this file exists. The action then runs the steps in the order of their definition on an **ubuntu-latest** container. The steps are: Checking out to the current Git branch Set up Python on the container Install the Python dependencies of the Bank Account app defined in requirements.txt Run the unit tests using Pytest

Add and commit your changes, then push the branch out to your remote repository on your GitHub account:

```
git add .
git commit -a -m "added a GitHub action to run unit tests automatically on code push"
git push --set-upstream "add-auto-tests"
```

Now open your repository in a browser and navigate under the Actions tab. You should see a new workflow started a few seconds ago:

y / add-automated-tests-off-platform-project Public

Watch 9 Fork 9 Star

Issues Pull requests Actions Projects Security Insights

workflows

All workflows

Showing runs from all workflows

Filter workflow runs

2 workflow runs

added automated unit tests to run in github actions

Continuous Integration #3: Commit artifacts pushed by subgenerator

16 seconds ago

In progress

Click on the workflow to show the details. The logs for every individual step are available:

build

succeeded 16 seconds ago in 12s

Search logs

> Set up job

3s

> Run actions/checkout@v2

1s

> Set up Python

0s

> Install dependencies

6s

> Run Tests

1s

> Post Set up Python

0s

> Post Run actions/checkout@v2

1s

> Complete job

0s

Click through the steps to read the logs. Once the workflow finishes, you will see

Back

Next

Learn Git & Git x(9) Licenses & xEntry on C++ E x[Entry] C++: Er xC++ | Codecademy xKornkawin/do x+ - x xcodecademy.com/courses/learn-git/articles/github-actions-tutorial-on-automated-testingAppsติดตั้ง JDK (Ja...Backtesting A...MATLAB for P...Tableau ServerCoursera-Stan...Guide Ragnar...Reading list

c

My Home

Course Menu

Get UnstuckTools🕒👤

> Complete job0s

Click through the steps to read the logs. Once the workflow finishes, you will see a green checkmark. If any of the unit tests fail, the workflow fails and you will see an email notification. You can try that by intentionally breaking one of the tests and pushing your code to your branch.


Run Tests on Pull Request Creation

Now let's make this GitHub Action workflow also run the tests when creating a new pull request.

Open `unittests.yaml` file and update the array of triggers to add `pull_request`:

```
on: [push, pull_request]
```

Commit and push your change to the branch. Notice that an instance of a container to run the tests will begin under the Actions tab just like before. But now, create a pull request from the `add-auto-tests` branch to your `main` branch (on your own repository). The GitHub action will then run as an automated check and ensure the unit tests pass. If the action fails, the pull request cannot be merged.



✓ All checks have passed2 successful checksHide all checks

✓

Continuous Integration / build (pull_request)Successful in 8sDetails

✓

Continuous Integration / build (push)Successful in 13sDetails

✓ This branch has no conflicts with the base branchMerging can be performed automatically.

Merge pull request

Conclusion

GitHub Actions enable automated workflows for developers. In this tutorial, we were able to add automated testing to a repository, which ensures quality control over the code. We encourage you to explore more GitHub Actions that can be configured to run other types of automation: docker builds, project compilation, tagging, releasing, and more!

BackNext