Using GitHub Actions - Building the CI pipeline



Estimated time needed: 30 minutes

In this lab, you will build the Continuous Integration pipeline by setting up the steps under job in your workflow. Please ensure you complete setting up the workflow, before starting this lab.

You will add the following steps to the job in this lab:

- Install dependencies Lint code with Flake8
- Run unit tests with nose

Learning Objectives

After completing this lab, you will be able to:

- · Explain components of a job in a GitHub Actions workflow
- Use predefined GitHub actions
- · Run multiline inline code in a ster
- Explore action logs in the GitHub UI

Prerequisites

- It is highly recommended that you complete the lab titled 'Using GitHub Actions Setting up workflow'. If you have not finished the lab, please finish that before starting this lab.
- A basic understanding of YAML
- A GitHub account
- An intermediate-level knowledge of CLIs

Pre-work

- 1. Generate a personal access token on GitHub Settings, if you already don't have one.
- 2. Open a terminal window by using the menu in the editor: Terminal > New Terminal.
- 3. Run the following command to authenticate with GitHub in the terminal with the personal access token.
- 1. gh auth login

Copied! Executed!

You will be taken through a guided experience as shown here:

What account do you want to log into? GitHub.com

What is your preferred protocol for Git operations? HTTPS
Authenticate Git with your GitHub credentials.

You will be logged into GitHub as your account user.

4. Clone your copy of the forked <u>GitHub repository</u>. This creates the wtecc-CICD_PracticeCode subdirectory under /home/project directory.

If you already have cloned this repository under /home/project, you can skip this step

5. If you don't have .github/workflows/workflow.yml in wtecc-CICD_PracticeCode directory, create workflow.yml file in the /home/project/wtecc-CICD_PracticeCode/.github/workflows directory with the following code:

```
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
  1.
2.
3.
4.
5.
6.
7.
8.
9.
                   name: CI workflow
                      push:
branches: [ "main" ]
                      pull_request:
   branches: [ "main" ]
                           runs-on: ubuntu-latest
container: python:3.9-slim
  11.
12.
Copied!
```

Step 1: Add the checkout Step

You will now add your first step to the job. The first task is to check out the code from the repository. You will use the predefined actions/checkout@v3 action to do this. You can learn more about this action on the GitHub

Your Task

- 1. Add the steps: section at the end of the build job in the workflow file
- ▼ Click here for a hint.

```
1. steps:
Copied!
```

1/10 about:blank

2. Add the checkout step as the first step. Give it the name: Check out and call the action actions/checkout@v3 by using the uses: keyword.

▼ Click here for a hint.

```
1. 1
2. 2
3. 3
1. steps:
2. - name: {insert step name here}
3. uses: {insert action name here}
```

Double-check that your work matches the solution below.

Solution

▼ Click here for the answer.

Replace the workflow.yml file with the code snippet below. You can also copy relevant parts of the code. Be sure to indent properly:

```
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15

1. name: CI workflow
2.
3. on:
4. push:
5. branches: ["main"]
6. pull_request:
7. branches: ["main"]
8.
9. jobs:
10. build:
11. runs-on: ubuntu-latest
12. container: python:3.9-slim
13. steps:
14. - name: Checkout
15. uses: actions/checkout@v3
```

Step 2: Install Dependencies

Now that you have checked out the code, the next step is to install the dependencies. This application uses pip, the Python package manager, to install all the dependencies from the PyPI package. The pip command refers to the requirements.txt file for the list of dependencies. The python:3.9-slim container you are using already has the pip package manager installed.

The commands that you will use in this step are:

```
1. 1
2. 2
1. python -m pip install --upgrade pip
2. pip install -r requirements.txt
```

Now, look at each line to understand what it is doing:

1. 1
1. python -m pip install --upgrade pip

This command installs pip if it is not found on the GitHub runner or updates it to the latest version if it is already installed. It is always a good practice to make sure that you are using the latest version of pip to minimize installation problems.

```
    1. 1
    1. pip install -r requirements.txt

Copied!
```

.....

This command instructs pip to install any dependencies listed in the requirements.txt file. It is a best practice to place the names of dependent packages here.

Note: If the above command gives any error, please execute the below command to install the dependencies

```
    1. 1
    1. pip3 install -r requirements.txt

Copied!
```

Your Task

- 1. Add a new named step after the checkout step. Name this step Install dependencies.
- ▼ Click here for a hint.

```
1. 1
1. - name: {insert step name here}
Copied!
```

- 2. Next, you want to run the commands to update the package manager pip and then install the dependencies. Since there are two commands, you can run these inline using the run: keyword with the pipe | operator.
- ▼ Click here for a hint.

```
1. 1
2. 2
3. 3
1. - name: Install dependencies
2. run: |
{insert code here}
```

about:blank 2/10

Copied!

Double-check that your work matches the solution below.

Solution

▼ Click here for the answer.

Replace the workflow.yml file with the code snippet below. You can also copy relevant parts of the code. Be sure to indent properly:

```
2. 2
3. 3
4. 4
  5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20

    name: CI workflow

   3. on:
4. p
           push:
               branches: [ "main" ]
            pull_request:
   branches: [ "main" ]
    6.
7.
  8.
9. jobs:
10. bui
            build:
  11.
                runs-on: ubuntu-latest
                container: python:3.9-slim
                steps:
- name: Checkout
uses: actions/checkout@v3
  16.
17.
18.
19.
                    - name: Install dependencies
                          un: |
python -m pip install --upgrade pip
pip install -r requirements.txt
Copied!
```

Step 3: Lint with flake8

It is always a good idea to add quality checks to your CI pipeline. This is especially true if you are working on an open source project with many different contributors. This makes sure that everyone who contributes is following the same style guidelines.

The next step is to use flake8 to lint the source code. Linting is checking your code for syntactical and stylistic issues. Some examples are line spacing, using spaces or tabs for indentation, locating uninitialized or undefined variables, and missing parentheses. The flake8 library was installed as a dependency in the requirements.txt file.

The flake8 commands take a few parameters. Now, take a look at the command and the parameters:

```
    1. 1
    1. flake8 service --count --select=E9,F63,F7,F82 --show-source --statistics
    Copied!
```

This command will run flake8 for the service folder of your repository. By adding the following options, you customize the linting process:

- --count: shows the count of all warnings and errors in the result of the command
 --select=E9,F63,F7,F82: limits the checks to syntax errors
 --show-source: adds the actual line number where the error/warning is found
 --statistics: shows the count of each type of error and warning in the result
- 1. flake8 service --count --max-complexity=10 --max-line-length=127 --statistics

Copied!

Unlike the first command that checks for syntax errors only, the second command will run all the available checks on the service folder of your repository.

Your Task

 $1.\,Add\ a\ new\ named\ step\ after\ the\ {\tt Install\ dependencies\ step.}\ Call\ this\ step\ {\tt Lint\ with\ flake8}.$

```
▼ Click here for a hint.

1. 1
2. 2
1. steps:
2. - name: {insert step name here}

Copied!
```

2. Next, you want to run the flake8 commands to lint your code. The commands are explained in detail above.

```
1. 1
2. 2
1. flake8 service --count --select=E9,F63,F7,F82 --show-source --statistics
2. flake8 service --count --max-complexity=10 --max-line-length=127 --statistics
Copied!
```

You can run inline commands using the run keyword with the pipe | operator.

▼ Click here for a hint.

```
1. 1
2. 2
3 . 3
4. 4
5. 5
1. steps:
2. - name: Lint with flake8
3. run: |
```

about:blank 3/10

```
4. {insert first command here}
5. {insert second command here}
Copied!
```

Double-check that your work matches the solution below.

Solution

▼ Click here for the answer.

Replace the workflow.yml file with the code snippet below. You can also copy relevant parts of the code. Be sure to indent properly:

```
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24

    name: CI workflow

   3. on:
           push:
                branches: main
            pull_request:
branches: main
    9. jobs:
  10.
11.
12.
13.
14.
15.
16.
17.
18.
             build:
                runs-on: ubuntu-latest
container: python:3.9-slim
                    - name: Checkout
                       uses: actions/checkout@v3
                    - name: Install dependencies
                       run: |
python -m pip install --upgrade pip
pip install -r requirements.txt
  20.
                     - name: Lint with flake8
                       flake8 service --count --select=E9,F63,F7,F82 --show-source --statistics
flake8 service --count --max-complexity=10 --max-line-length=127 --statistics
Copied!
```

Step 4: Test Code Coverage with nosetests

You will use nose in this step to unit test the source code. Nose is configured via the included setup.cfg file to automatically include the flags –with-spec and –spec-color so that red-green-refactor is meaningful. If you are in a command shell that supports colors, passing tests will be green and failing tests will be red.

Nose is also configured to automatically run the coverage tool, and you should see a percentage of coverage report at the end of your tests.

Your Task

- 1. Add a new named step after the Lint with flake8 step. Call this step Run unit tests with nose.
- ▼ Click here for a hint

```
1. 1
2. 2
1. steps:
2. - name: {insert step name here}
Copied!
```

2. Next, you want to run the nosetests command to test your code and report on code coverage.

Since you are running a single command, you do not have to use the pipe | operator with run.

▼ Click here for a hint.

```
1. 1
2. 2
1. - name: Run unit tests with nose
2. run: {insert command here}
Copied!
```

Double-check that your work matches the solution below.

Solution

▼ Click here for the answer.

Replace the workflow.yml file with the code snippet below. You can also copy relevant parts of the code. Be sure to indent properly:

1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8

about:blank 4/10

```
12. 12
13. 13
14. 14
15. 15
  16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
  23. 23
24. 24
25. 25
  26. 26
27. 27
28. 28
   1. name: CI workflow
   2.
3. on:
4. p
            push:
             branches: [ "main" ]
pull_request:
branches: [ "main" ]
    9. jobs:
  10.
              build:
                 runs-on: ubuntu-latest
container: python:3.9-slim
steps:
- name: Checkout
  11.
12.
13.
14.
15.
                        uses: actions/checkout@v3
                     - name: Install dependencies run: |
                            python -m pip install --upgrade pip
pip install -r requirements.txt
  19.
20.
21.
                     - name: Lint with flake8
run: |
flake8 service --count --select=E9,F63,F7,F82 --show-source --statistics
flake8 service --count --max-complexity=10 --max-line-length=127 --statistics
  25.
26.
  27.
                      - name: Run unit tests with nose
run: nosetests -v --with-spec --spec-color --with-coverage --cover-package=app
Copied!
```

Step 5: Push Code to GitHub

To test the workflow and the CI pipeline, you need to commit the changes and push your branch back to the GitHub repository. As described earlier, each new push to the main branch should trigger the workflow.

Your Task

1. Configure the Git account with your email and name using the git config --global user.email and git config --global user.name commands.

▼ Click here for a hint.

```
Open the terminal and configure your email:
```

Open the terminal and configure your user name

```
1. 1
1. git config --global user.name "Your Name"
Copied!
```

- 2. The next step is to stage all the changes you made in the previous exercises and push them to your forked repo on GitHub.
- ▼ Click here for a hint.

You can use the following commands to commit your changes to staging and then push to your forked repository:

```
1. 1
2. 2
3. 3
1. git add -A
2. git commit -m "COMMIT MESSAGE"
3. git push

Copied! Executed!
```

Your output should look similar to the image below:

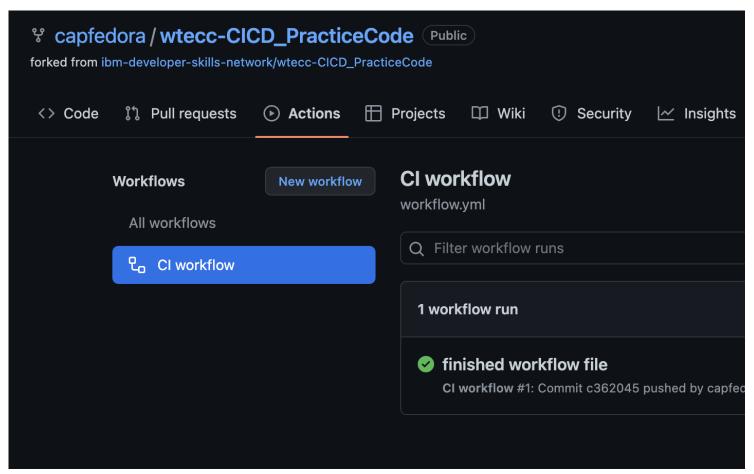
Solution

```
theia@theia-captainfedol:/home/project/wtecc-CICD_PracticeCode$ git config --global user.email "
theia@theia-captainfedol:/home/project/wtecc-CICD_PracticeCode$ git config --global user.name "
theia@theia-captainfedol:/home/project/wtecc-CICD_PracticeCode$ git add -A
theia@theia-captainfedol:/home/project/wtecc-CICD_PracticeCode$ git commit -m "finished workflow file"
[main c362045] finished workflow file
1 file changed, 24 insertions(+)
create mode 100644 .github/workflows/workflow.yml
theia@theia-captainfedol:/home/project/wtecc-CICD_PracticeCode$ git push
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 746 bytes | 373.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/capfedora/wtecc-CICD_PracticeCode.git
03977e9..c362045 main -> main
```

about:blank 5/10

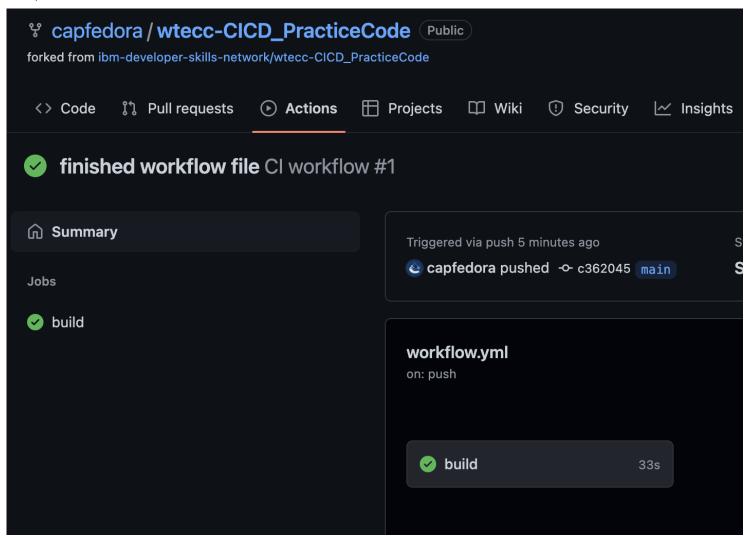
Step 6: Run the Workflow

You can run the workflow now. Open the Actions tab in the forked repository in your GitHub account. Click the CI workflow action. Pushing changes in the previous step should have triggered a build action and it should be visible in the Actions tab.



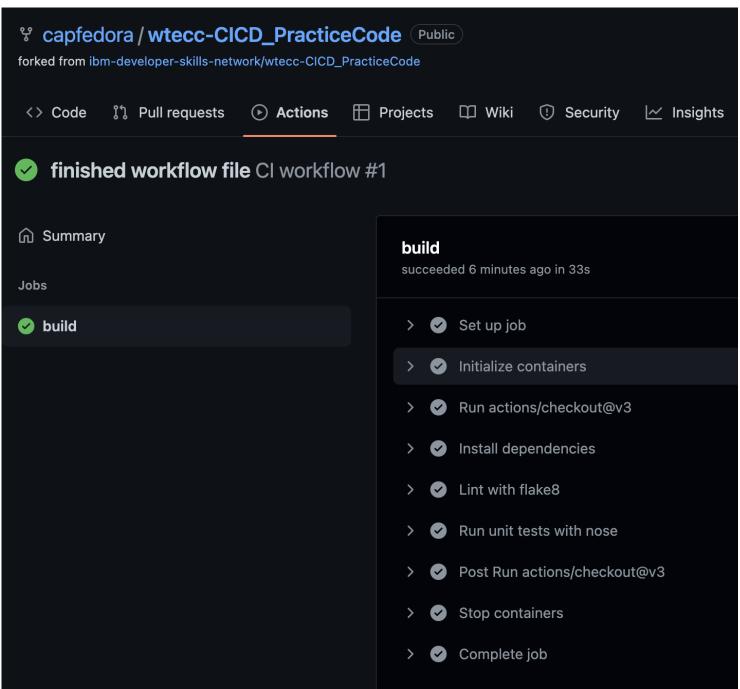
Click the most recent workflow run to open the details page. You should see your workflow completed successfully:

about:blank 6/10



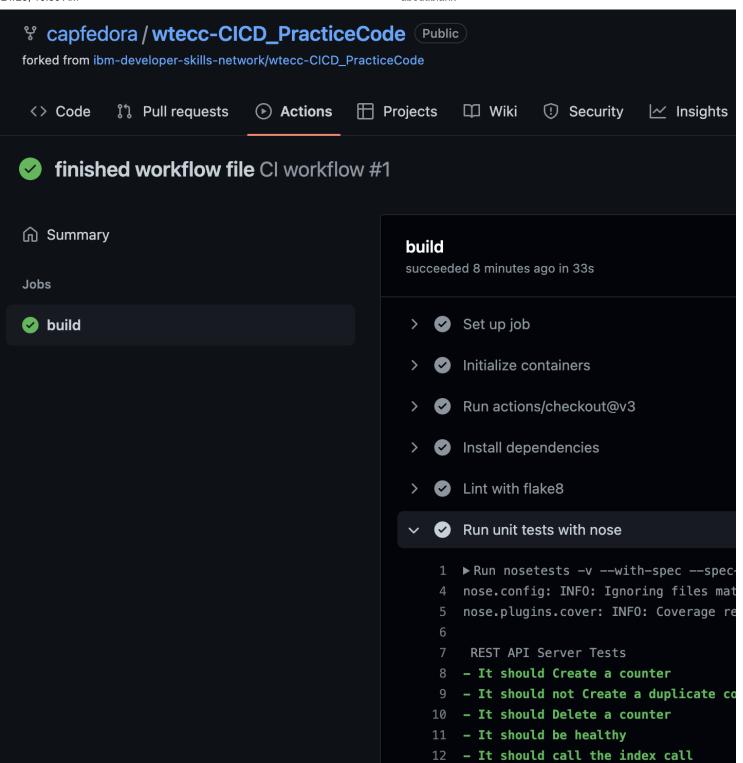
You can now click the build job to see the logs for each step:

about:blank 7/10



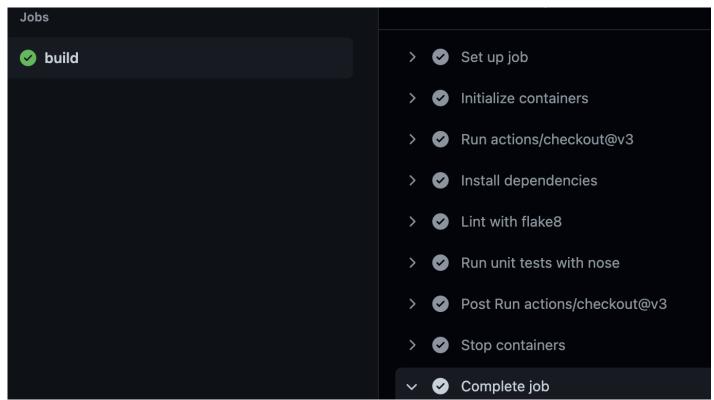
Spend some time here to expand each section:

about:blank 8/10



If the workflow completes, you will see the $Complete\ job\ step\ at$ the end of the build job logs.

about:blank 9/10



Conclusion

Congratulations! You are now able to use GitHub Actions to create workflows for the CI pipeline.

In this lab, you added the steps to your Continuous Integration pipeline for your application. As a result, your application is automatically built and tested when you commit your code to the GitHub repository.

Next Steps

Try to set up a GitHub action for your own project that checks out the code, runs unit tests, and performs linting on every push request to the default branch.

Author(s)

Tapas Mandal

Other Contributor(s)

Captain Fedora John Rofrano

Change Log

Date	Version	Changed by	Change Description
			e .
2022-06-19		Tapas Mandal	Created new instructions for CI/CD project
2022-06-2	1 0.2	Tapas Mandal	Adding few improvements as per review comments
2022-08-03	5 0.3	Beth Larsen	Initial QA pass
2022-08-11	1 0.4	Captain Fedora	Revised version with new instructions and fixes
2022-08-12	2 0.5	Steve Ryan	ID Review
2022-08-12	2 0.4	Captain Fedora	Fixes from ID Review
2022-08-13	5 0.6	Beth Larsen	Second QA pass
2022-08-13	5 0.7	Steve Ryan	Fixed all 'workflow.yml' instances
2022-08-13	5 0.8	Steve Ryan	Fixed all missing periods
2022-08-13	5 0.9	John Rofrano	Added minor additions for clarity
2022-08-13	5 1.0	Steve Ryan	Final minor typo fixes
2022-08-13	5 1.1	Captain Fedora	Renamed images
2022-08-13	5 1.2	Captain Fedora	Split the lab into part 1 and part 2
2022-08-22	2 1.2	Captain Fedora	Added next steps
2022-08-23	3 1.3	Beth Larsen	Part 2 QA pass
2023-02-2	1 1.4	Lavanya	Changed the wording to make this lab more independent
2023-03-13	5 1.5	Lavanya Rajalingam	Updated SN Logo
2023-04-20	5 1.6	Lavanya Rajalingam	Added note for using pip3

$\hbox{@}$ IBM Corporation 2023. All rights reserved.

about:blank 10/10