Add Continuous Integration and Continuous Deployment

Estimated time needed: 60 minutes

Your team is growing! Management has decided to hire front-end and back-end engineers to ensure features on the roadmap are developed in time for future releases. However, this means that multiple engineers will need to work in parallel on the repository. You are tasked with ensuring the code being pushed to the main branch meets the team coding style and is free of syntax errors.

In this lab, you will add linting to your repository that automatically checks for such errors whenever a developer creates a pull request or whenever a branch is merged into the default main branch. Before we dive into the lab, here is a primer on GitHub Actions.

GitHub Actions

GitHub actions provide an event-driven way to automate tasks in your project. There are several kinds of events you can listen to. Here are a few examples:

- push: Runs tasks when someone pushes to a repository branch.
 pull_request: Runs tasks when someone creates a pull request (PR). You can also start tasks when certain activities happen, such as:
- pull_request: Runs tasks when someone creates a pull requ

 PR opened
 PR closed
 PR reopened

 create: Run tasks when someone creates a branch or a tag, delete: Run tasks when someone deletes a branch or a tag.
 manually: Jobs are kicked off manually.

GitHub Action Components

You will use one or more of the following components in this lab:

- Workflows: A collection of jobs you can add to your repository.
 Events: An activity that launches a workflow.
 Jobs: A sequence of one or more steps. Jobs are run in parallel by default.
 Steps: Individual tasks that can run in a job. A step may be an action or a command.
 Actions: The smallest block of a workflow.

GitHub Workflow

You are provided with a workflow template below. Let's examine it.

```
1. name: 'Lint Code
2.
                               branches: [master, main]
pull_request:
branches: [master, main]
8.
9. jobs:
10. lint_mython:
110. lint_mython:
111. lint_mython:
112. runs-on: ubuntu-latest
113. steps:
114. steps:
115. - name: Checkout Raposit
116. usses: actions/setcout
118. - name: Set up Python
119. - name: Set up Python
120. usses: actions/setcup-py
121. with:
121. python-uses: actions/setup-py
122. python-uses: actions/setup-py
123. python-uses: install dependen
125. run: |
126. python - up pi install
127. ppthon-use pi install flakes
128. - name: Install flakes
129. - name: Print uorking di
120. - name: Print uorking di
121. lint_js:
122. - name: Print uorking di
123. run: |
124. - name: Install lates
125. - name: Install the
126. python - uses in tinter
127. - name: Install lates
128. - name: Install lates
129. - name: Install lode.
129. - name: Install lode.
129. - name: Install Shint
120. - name: Install Shint
120. - name: Install Shint
120. - name: Install Shint
121. - name: Install Shint
122. - name: Run Liter
123. - name: Install Shint
124. - name: Install Shint
125. - name: Run Liter
126. - name: Install Shint
127. - name: Install Shint
128. - name: Run Liter
129. - name: Install Shint
129. - name: Run Liter
129. 
                                                       - name: Set up Python
uses: actions/setup-python@v4
with:
                                                                                                 :
thon-version: 3.12
                                                       - name: Install dependencies
run: |
python -m pip install --upgrade pip
pip install flake8
                                                       - name: Print working directory run: pwd
                                                                               pwd
# This command finds all Python files recursively and runs flake8 on them
find .-name "*.py" -exec flake8 {} +
echo "Linted all the python files successfully"
                                       lint_js:
    name: Lint JavaScript Files
    runs-on: ubuntu-latest
                                                                steps:
- name: Checkout Repository
uses: actions/checkout@v3
                                                                           - name: Install Node.js
uses: actions/setup-node@v3
with:
                                                                               name: Install JSHint
run: npm install jshint --global
                                                                                           un: | # This command finds all JavaScript files recursively and runs JSHint on them find ./server/database -name "*.js" -exec jshint \{\} + echo "Linted all the js files successfully"
Copied!
```

1. The first line names the workflow.
2. The next line defines when this workflow will run. The workflow should run when developers push a change to the main branch or create a PR. These two ways are captured as follows:
o run on push to the main branch (main or master):

```
1. 1
2. 2
1. push:
2. branches: [master, main]
Copied!
```

• run when PR is created on main branches (main or master):

Copied!

u will then define all the jobs. There are two jobs in this workflow:
u will then define all the jobs. There are two jobs in this workflow:
u lint_ys: Linting Python function

GitHub Jobs

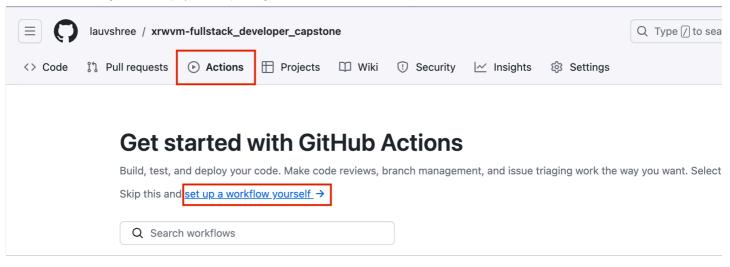
Let's look at each of these jobs:

- 1. lint_python

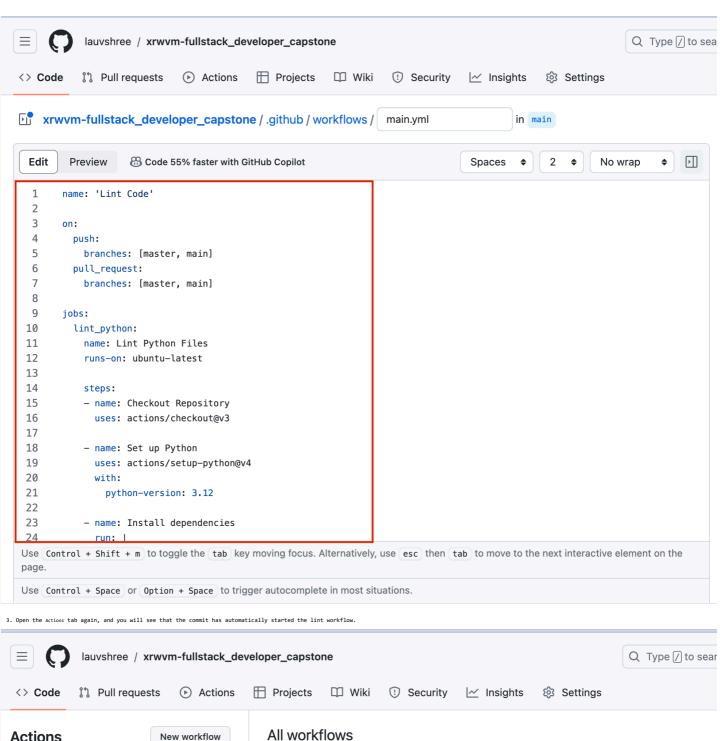
 - Set up the Python runtime for the action to run using the actions/setup-python@v4 action.
 Install all dependencies using pip install.
 Run the linting command flaxes *.p.p in all files in server directory recursively.
 Print a message saying the linting was completed successfully.
- 2. lint function is
 - o Set up the Node.js runtime for the action to run using the actions/setup-node@v3 action.
 o Install all JSHint linter npm install jshint.
 o Run the linting command on all the .js files in the database directory recursively.
 o Print a message saying the linting was completed successfully.

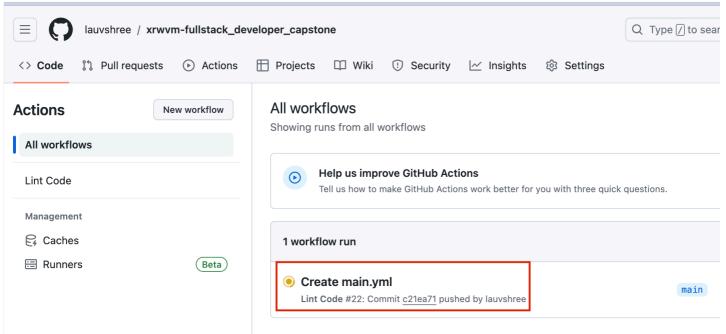
Enable GitHub Actions

1. To enable GitHub action, log into GitHub and open your forked repo. Next, go to the Actions tab and click Set up a workflow yourself.

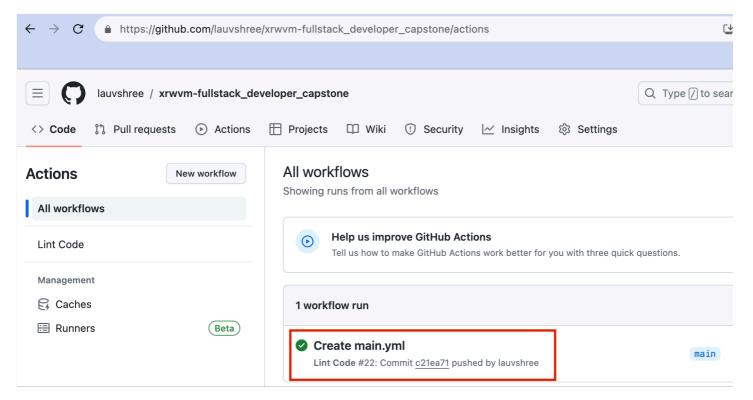


2. Paste the lint code given above inside main.yml and commit it.





^{4.} You can click the workflow run to see the individual jobs and the logs for each job. When the workflow successfully completes, you will see the green tick indicating it went well. A red cross would mean there were errors found in the code while linting.



5. Check these hints to resolve usual Linting errors you could come across.

▶ Click here

Submission

Take a screenshot of the action workflow succeeding and save it as ${\tt CICD.png.}$

Summary

In this lab, you added a linting service to your application. As a result, all new code will automatically get checked for syntax errors, and this will ensure all developers are following the team coding guidelines.

Author(s)

Upkar Lidder

Lavanya T S

Other Contributor(s)

Yan Luo

Priya

