CI/CD with GitHub Actions

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 must ensure the code pushed to the main branch meets the team coding style and is free of syntax errors.

So far, you have created the back end, created the front end, integrated the front end and back end, and pushed all the code to your GitHub repository. Go to your GitHub repository to ensure all your code is present.

The code in your repository should be error-free, follow the standardized coding norms, and adhere to best practices. You may have noticed some suggestions that come up with colored squiggly lines while using the Cloud IDE. These are the IDE's built-in `linters` in action.

Linting automates the code review process by providing feedback on code quality and adherence to coding standards before being merged into the codebase.

Overall, linting is an essential tool for improving code quality, maintaining consistency, and preventing common coding errors and security vulnerabilities in software development projects.

In this lab, you will add linting to your repository using a GitHub workflow containing GitHub Actions. Your linting will automatically check for errors whenever a developer creates a pull request or whenever a branch is merged into the default main branch.

GitHub Workflow

Examine the workflow template provided below.

```
name: 'CI/CD'
on:
  push:
    branches: [master, main]
  pull_request:
    branches: [master, main]
jobs:
  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:
          node-version: 14
      - name: Install JSHint
        run: npm install jshint --global
      - name: Run Linter
        run: |
          # This command finds all JavaScript files recursively and runs JSHint on them
          find ./giftlink-backend -name app.js -exec jshint {} +
          find ./giftlink-backend -name auth.js -exec jshint {} +
          find ./giftlink-backend -name giftRoutes.js -exec jshint {} +
          find ./giftlink-backend -name searchRoutes.js -exec jshint {} +
          echo "Linted all the js files successfully"
  client_build:
      name: Build client
      runs-on: ubuntu-latest
      steps:
      - name: Checkout Repository
        uses: actions/checkout@v3
      - name: Install Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 14
      - name: Run build
        run: |
          cd giftlink-frontend
          npm install
          npm run build
          echo "Client-side build successfully"
```

1. Line 1 contains the name of the workflow.
2. The lines after **on** define when this workflow will run. The workflow should run when developers *push* a change to the main branch or create a *pull_request*. These two ways are captured as follows:

- run on push to the main branch (main or master):

```
push:
  branches: [master, main]
```

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

```
pull_request:
  branches: [master, main]
```

3. Next, the workflow defines 2 jobs:
   - lint_js: Lint specified JavaScript files
   - client_build: Ensure there are no errors when building the React client

GitHub Jobs

Let's look at each of these jobs:

1. lint_js

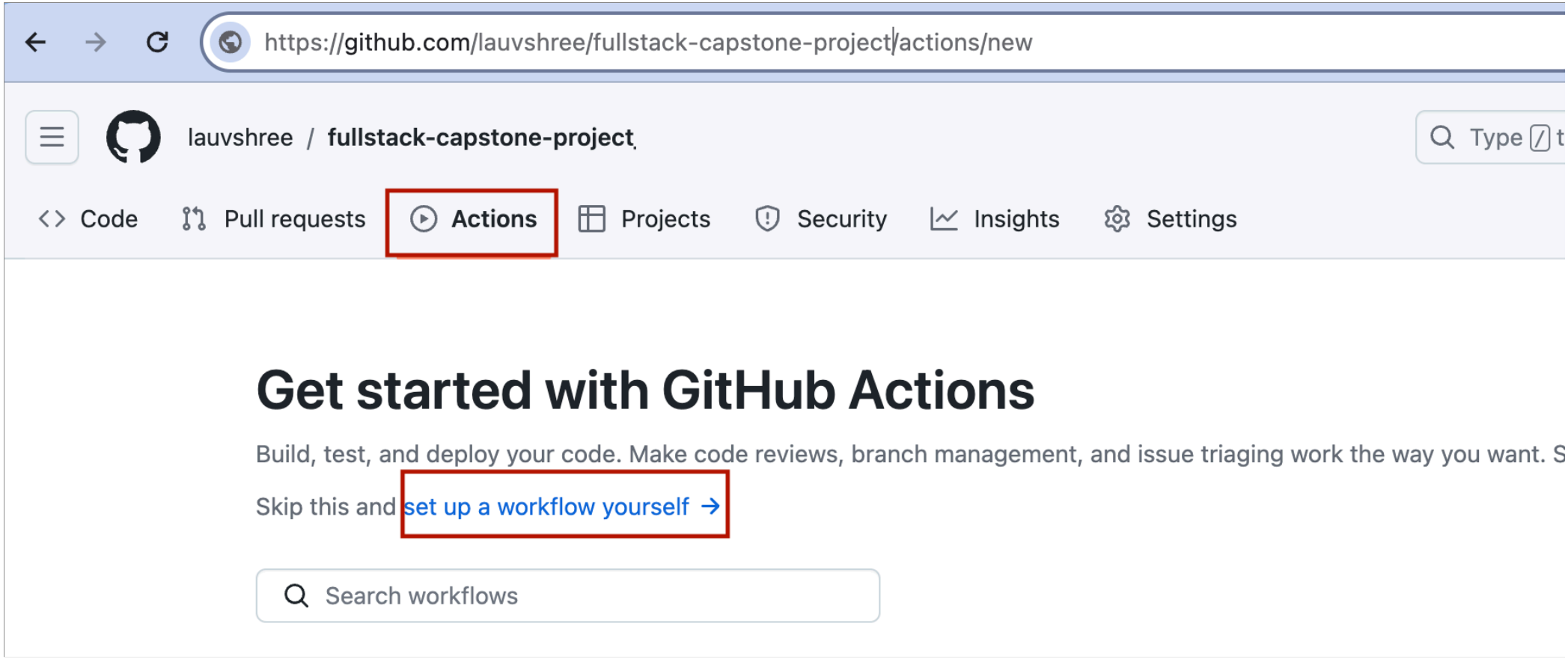
   - Set up the Node.js runtime for the action to run using the `actions/setup-node@v3` action.
   - Install all JSHint linter `npm install jshint`.
   - Run the linting command on the .js files specified.
   - Print a message saying the linting was completed successfully.

2. client_build

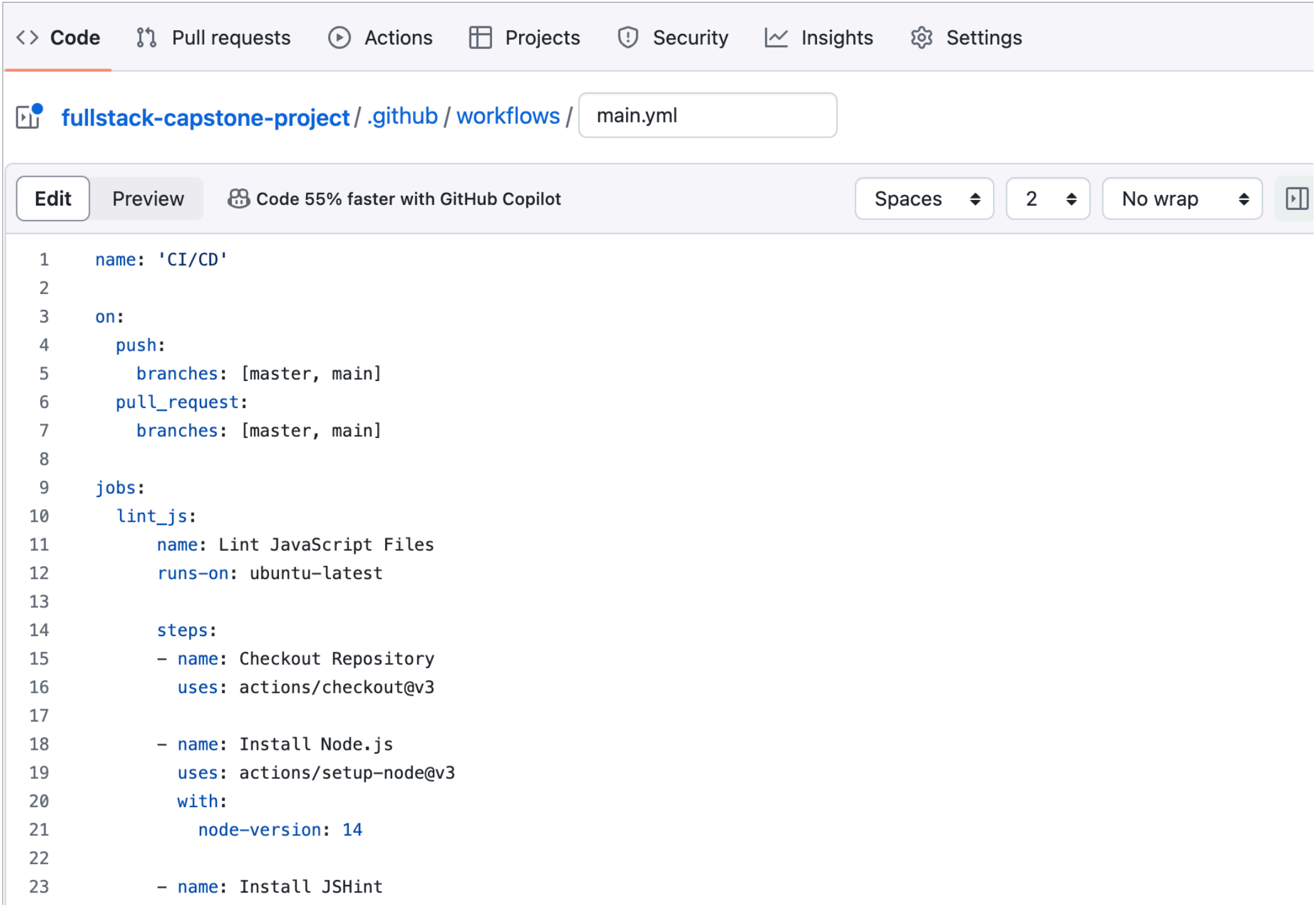
   - Set up the Node.js runtime for the action to run using the `actions/setup-node@v3` action.
   - Change to the front-end directory `giftlink-frontend`
   - Install all the packages.
   - Run the `npm run build` to build the client side and copy it to the server side.
   - Print a message saying the linting was completed successfully.

Enable GitHub Actions

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

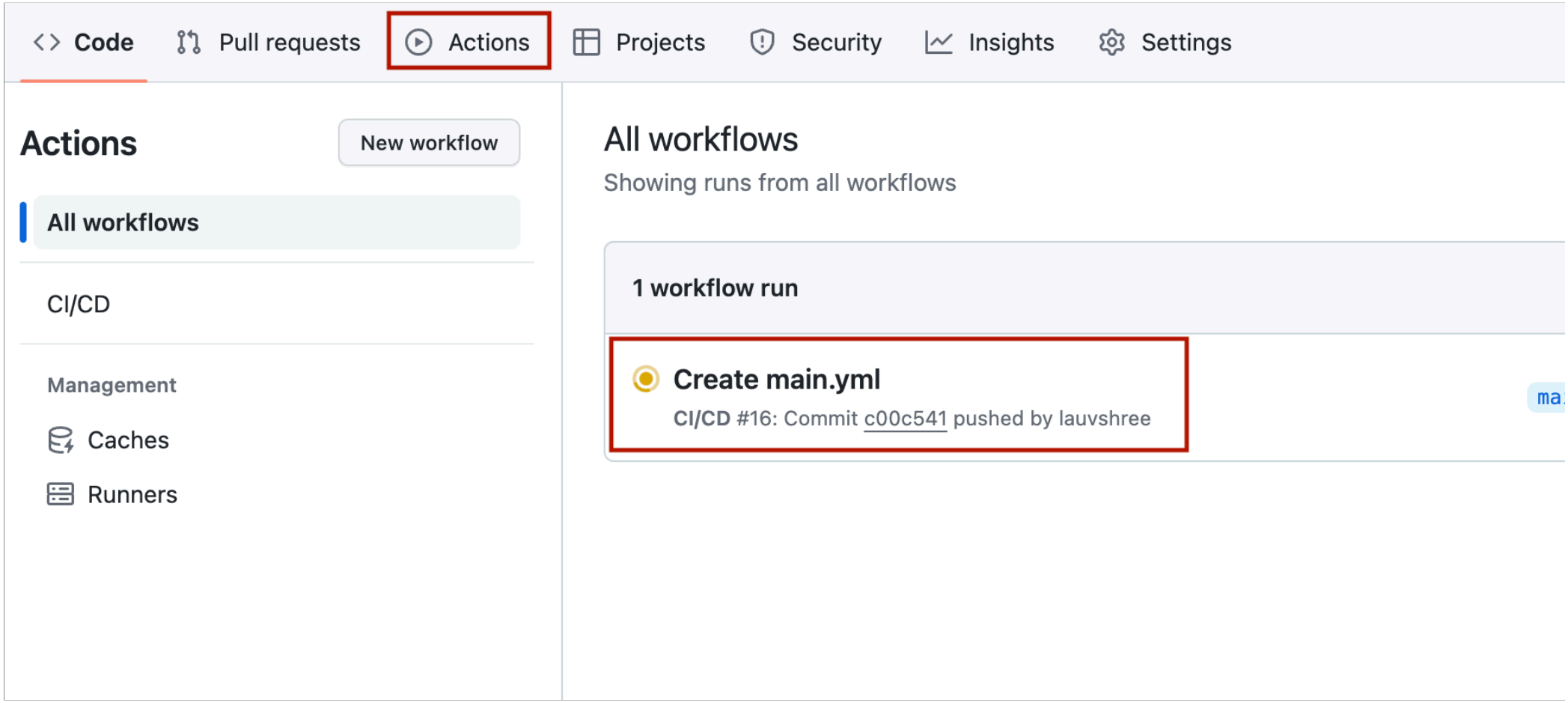


2. Paste the lint code given above inside `main.yml` and commit it.

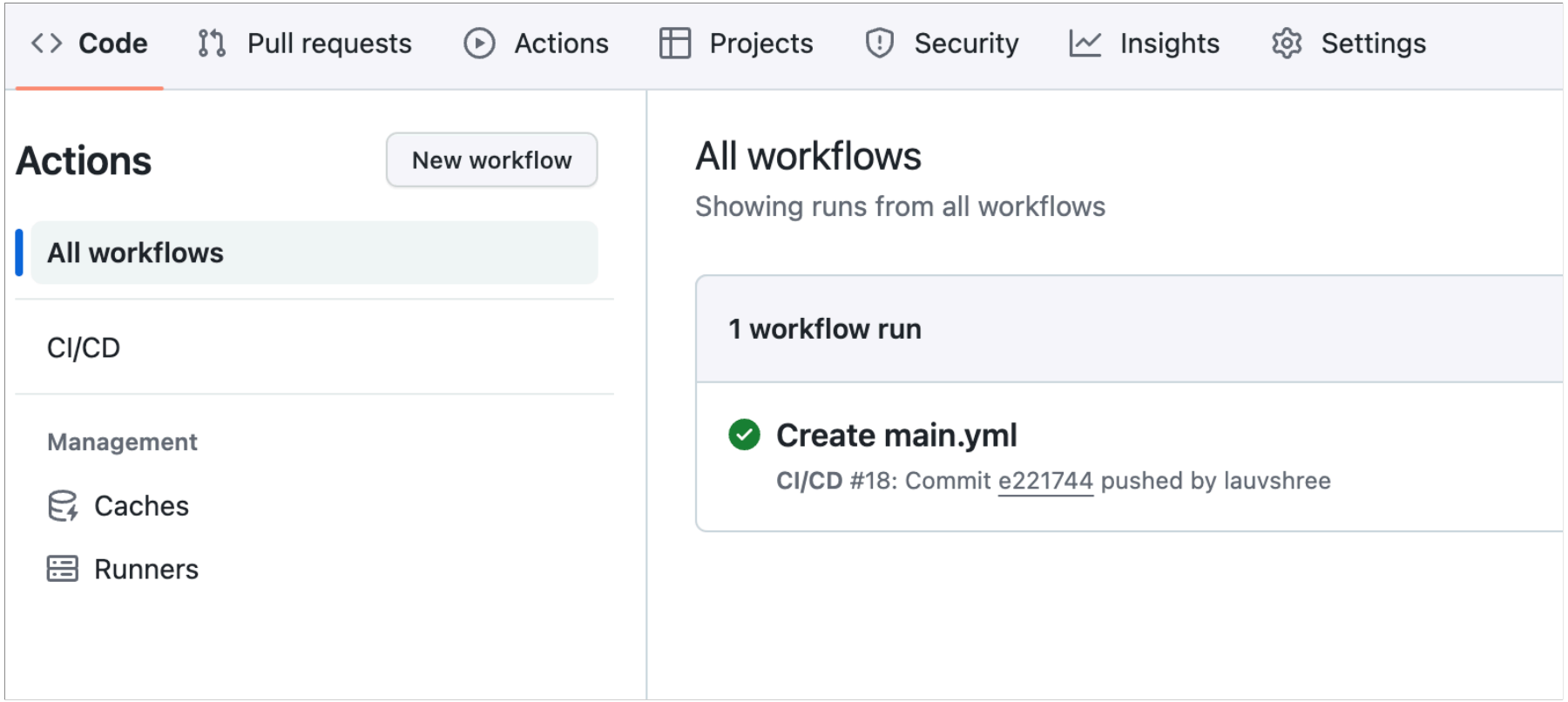

Code
Pull requests
Actions
Projects
Security
Insights
Settings
fullstack-capstone-project / .github / workflows / main.yml
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
No wrap
name: 'CI/CD'
on:
push:
branches: [master, main]
pull_request:
branches: [master, main]
jobs:
lint_js:
name: Lint JavaScript Files
runs-on: ubuntu-latest
steps:
- name: Checkout Repository
uses: actions/checkout@v3
- name: Install Node.js
uses: actions/setup-node@v3
with:
node-version: 14
- name: Install JSHint

```
24            run: npm install jshint --global
25
26          - name: Run Linter
27            run: |
```

3. Open the `Actions` tab again, and you will see that the commit has automatically started the lint workflow.



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 means it found errors in the code while linting.



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

▼ Click here

JS Hint (JavaScript) linting errors:

1. If you receive one or more errors as listed below:

```
'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
'arrow function syntax (=>)' is only available in ES6 (use 'esversion: 6').
'async functions' is only available in ES8 (use 'esversion: 8').
'let' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
'template literal syntax' is only available in ES6 (use 'esversion: 6').
```

Resolution:

Add the line below at the start of the file(s) where this error is reported:

```
/*jshint esversion: 8 */
```

Submission

Take a screenshot of the action workflow succeeding and save it as `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.