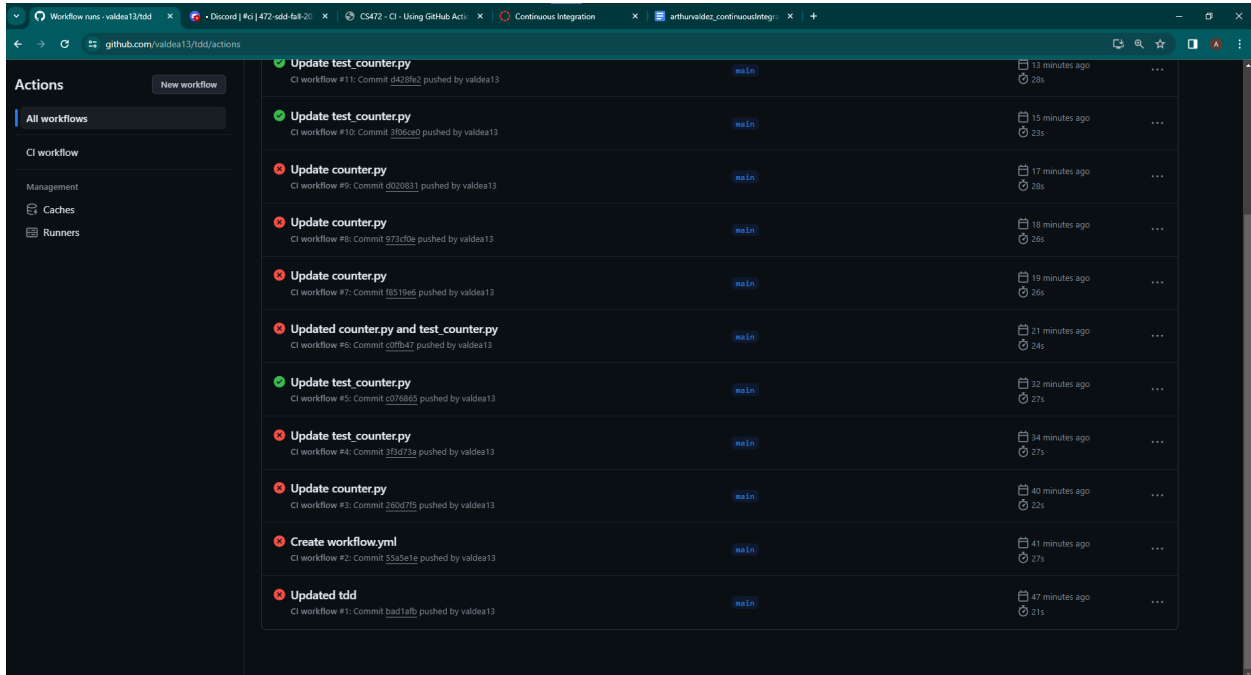


# Continuous Integration Report

## Task 1



The screenshot shows the GitHub Actions interface for the repository valdea13/tdd. The left sidebar contains the 'Actions' tab with a 'New workflow' button and a list of workflow categories: 'All workflows', 'CI workflow', 'Management', 'Caches', and 'Runners'. The main area displays a list of workflow runs. Each run includes a status icon (green for success, red for failure), a title, a commit hash, the branch name (main), and the time taken to complete. The runs are as follows:

Run ID	Commit	Branch	Status	Time
CI workflow #11	d428fe2	main	Success	13 minutes ago
CI workflow #10	3f06ca0	main	Success	15 minutes ago
CI workflow #9	d020831	main	Failure	17 minutes ago
CI workflow #8	973cfe	main	Failure	18 minutes ago
CI workflow #7	f8519e6	main	Failure	18 minutes ago
CI workflow #6	cc0b87	main	Failure	21 minutes ago
CI workflow #5	c070865	main	Success	32 minutes ago
CI workflow #4	3f3d73a	main	Failure	34 minutes ago
CI workflow #3	260d7f5	main	Failure	40 minutes ago
CI workflow #2	55a5e1e	main	Failure	41 minutes ago
CI workflow #1	bad1af6	main	Failure	47 minutes ago

```
.github > workflows > ! workflow.yml
1  name: CI workflow
2
3  on:
4    push:
5      branches:
6        - "main"
7    pull_request:
8      branches:
9        - "main"
10
11  jobs:
12    build:
13      runs-on: ubuntu-latest
14
15      container: python:3.9-slim
```

# Task 2

The screenshot shows a GitHub Actions workflow run for the repository 'valdea13 / tdd'. The workflow is named 'Update test\_counter.py #5' and is currently in a 'Success' state. It was triggered by a push to the 'main' branch by user 'valdea13' 34 minutes ago. The total duration of the run is 27 seconds. The workflow file is 'workflow.yml' and it runs on 'ubuntu-latest'. The workflow consists of a single job named 'build' which has a duration of 17 seconds. The job steps are: Checkout, Install dependencies, Lint with flake8, and Run unit tests with nose. The workflow file is located at '.github/workflows/workflow.yml'. There is a warning annotation stating that Node.js 16 actions are deprecated and should be updated to Node.js 20.

Summary

Jobs

- build

Run details

- Usage
- Workflow file

workflow.yml

on: push

build 17s

Annotations

1 warning

build

Node.js 16 actions are deprecated. Please update the following actions to use Node.js 20: actions/checkout@v3. For more information see: <https://github.blog/changelog/2023-09-22-github-actions-moved-to-action/>

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

```
def test_delete_a_counter(self):
    """It deletes a counter"""
    post_result = self.client.post('/counters/tar')
    self.assertEqual(post_result.status_code, status.HTTP_201_CREATED)

    delete_result = self.client.delete('/counters/tar')
    self.assertEqual(delete_result.status_code, status.HTTP_204_NO_CONTENT)

    get_result = self.client.get('/counters/tar')
    self.assertEqual(get_result.status_code, status.HTTP_404_NOT_FOUND)
```

With this unit test, I'm in the red phase. I haven't yet implemented the `.delete()` function but this test should cover most of what is asked. This unit test creates a counter called `tar` using the `.post()` function and its return value is stored in `post_result`. I then use an `assertEqual()` to verify that the creation of this counter was successful. Then I continue to call the `.delete()` function to delete the newly created `tar` counter. Like earlier, I use an `assertEqual()` to verify that the `.delete()` function was successful. Finally, I call the `get()` function to ensure that the counter `tar` does not exist by using an `assertEqual()`.

```
@app.route('/counters/<name>', methods=['DELETE'])
def delete_counter(name):
    """Deletes a counter"""
    app.logger.info(f"Request to update counter: {name}")

    if name not in COUNTERS:
        return {"Message": f"Counter {name} doesn't exist"}, status.HTTP_404_NOT_FOUND
    del COUNTERS[name]
    return {"Message": f"Counter {name} was successfully deleted."}, status.HTTP_204_NO_CONTENT
```

The `delete_counter()` function is similar to the `get_counter()` function. The only difference is that the counter that was specified is deleted if it exists within the `COUNTERS` dict. The function simply returns a message that the counter was successfully deleted. After implementing this function, the unit test finally worked and I'm in the green phase.

Arthur Valdez

Github Repository: <https://github.com/valdea13/tdd>