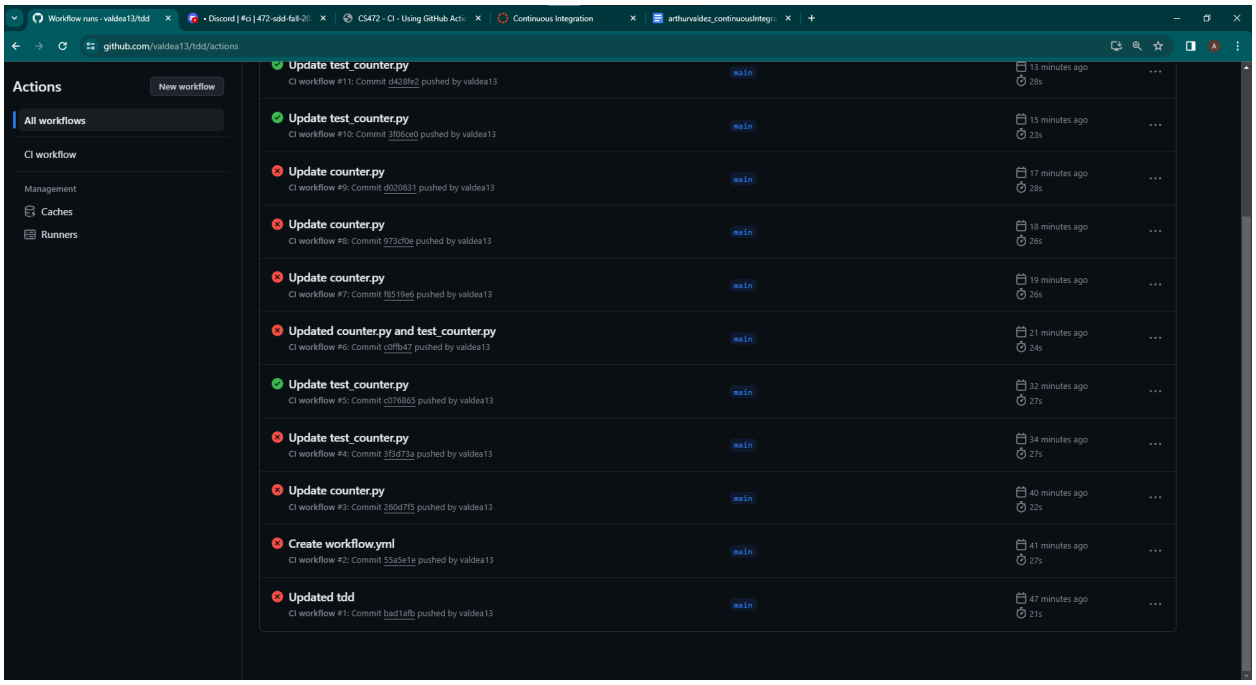
Continuous Integration Report

Task 1



.github > workflows > workflow.yml

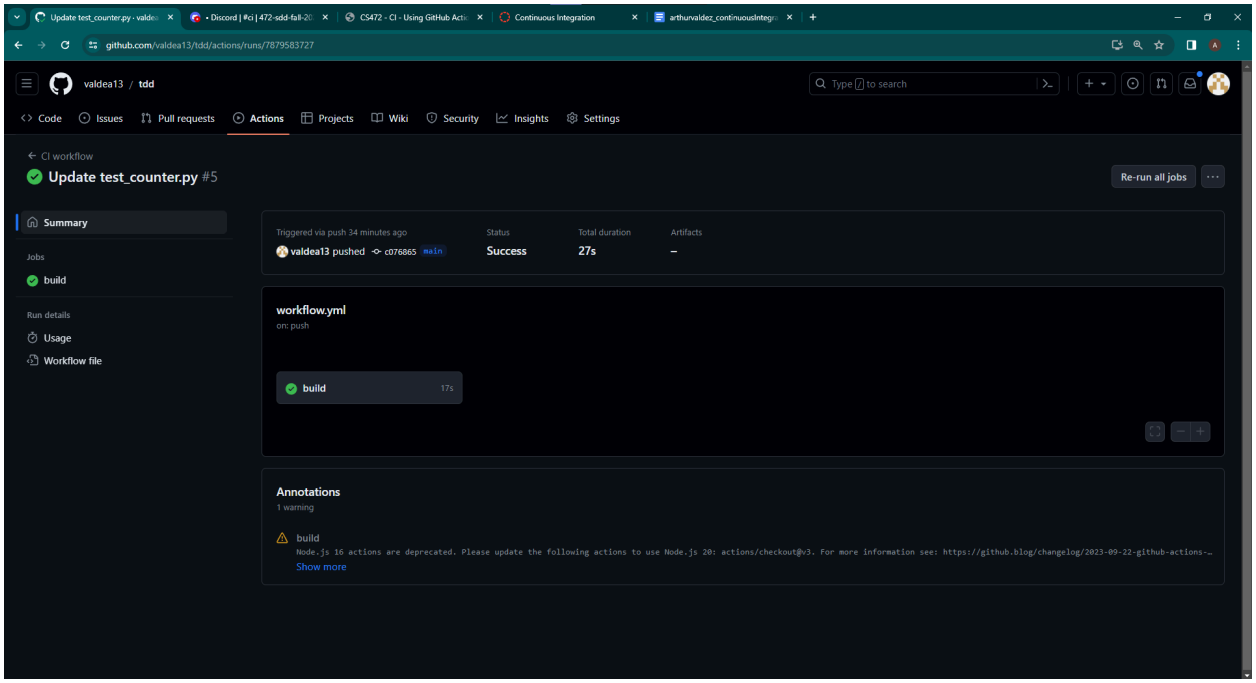
```yaml
name: CI workflow

on:
    push:
        branches:
            - "main"
    pull_request:
        branches:
            - "main"

jobs:
  build:
    runs-on: ubuntu-latest

    container: python:3.9-slim
```

Task 2



.github > workflows > ! workflow.yml

```yaml
name: CI workflow

on:
    push:
        branches:
            - "main"
    pull_request:
        branches:
            - "main"

jobs:
  build:
    runs-on: ubuntu-latest

    container: python:3.9-slim

    steps:
        - name: Checkout
          uses: actions/checkout@v3
        - name: Install dependencies
          run: |
            python -m pip install --upgrade pip
            pip install -r requirements.txt
        - name: Lint with flake8
          run: |
            flake8 src --count --select=E9,F63,F7,F82 --show-source --statistics
            flake8 src --count --max-complexity=10 --max-line-length=127 --statistics
        - name: Run unit tests with nose
          run: |
            nosetests -v --with-spec --spec-color --with-coverage --cover-package=app
```

```python
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 assetEqual() 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().

```python
@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