

Министерство науки и высшего образования РФ

**Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа №4

Github Actions. Linter's subsystem. Интеграция CI/CD.

Выполнил студент группы № М3102

Лопатенко Георгий Валентинович

Подпись:

Проверил:

Приискалов Роман Андреевич

Санкт-Петербург

2021

Требования к выполнению лабораторной работы

На основе лабораторных работ по программированию - с помощью GitHub Actions Произвести настройку CI:CD для вашего проекта - необходимо произвести автоматизацию сборки, прогона тестов и упаковки в exe вашего проекта по событию коммита.

Отчет

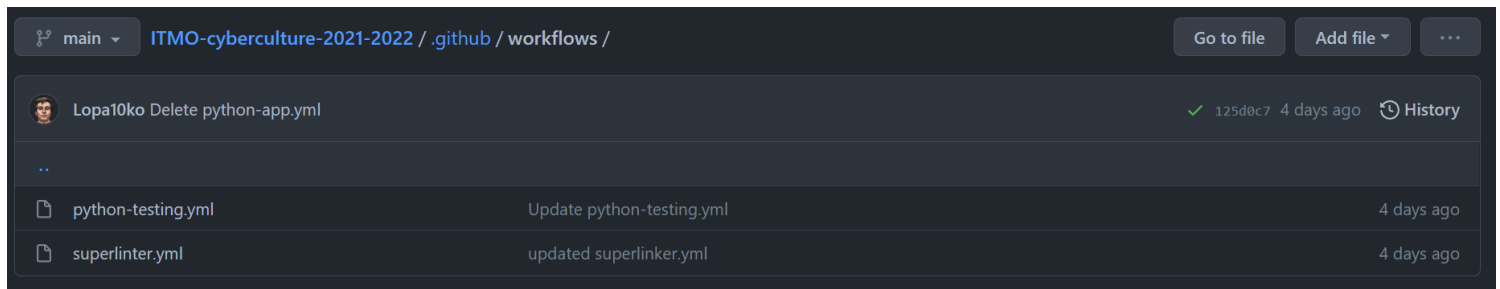
Будем работать с репозиторием из второй лабораторной работы в курсе Цифровая культура. Удаленный репозиторий содержит в себе простой проект на Python и тесты к нему через библиотеку unittest.

Чтобы продемонстрировать работу линтера, необходимо создать файл .yaml (Yet Another Markup Language), но сначала разберемся, что из себя представляет линтер и как он работает на платформе Github:

Линтер – это программа, которая проверяет код на соответствие стандартам в соответствии с определенным набором правил. Правила описывают отступы, названия создаваемых сущностей, скобки, математические операции, длину строк и множество других аспектов. Каждое отдельное правило кажется не очень важным, но соблюдение их всех – основа хорошего кода. (<https://github.com/marketplace/actions/super-linter>)

Главная задача линтера – сделать код единообразным, удобным для восприятия и самим программистом, и другими людьми, которые будут читать код. В разных командах могут использоваться разные линтеры и разные наборы правил для них, но главное – уметь работать с линтером в принципе, а привыкнуть писать по определенным правилам будет несложно. (<https://github.com/github/super-linter>)

1. Создадим директорию в репо “./github/workflows”. (!) Другое название директории не будет триггерить работу линтера.



2. Файл python-testing.yml будет основным для автоматизации работы с тестами. Здесь прописываем сценарий: при каждом git push запускать файлы тестов с unittest библиотекой. Если все тесты прошли, линтер завершает свою проверку. Смысл в том, чтобы всегда после внесения изменений в тело основной программы или в дерево зависимостей проекта оставлять за собой репозиторий с рабочим кодом на известных тестах.

Линтер запускается при каждом обновлении содержимого репозитория, кроме пуша тэгов. Проверка осуществляется по всем веткам. Система отладки и запуска на серверах Github – Ubuntu (последняя доступная версия). В разделе steps перечисляем, что необходимо сделать: в данном случае установить python и все компоненты requirements.txt для работы проекта (docker v4.1.0, Flask v1.1.1, coverage v4.5.4)

3. Умышленно напишем тест, который валит проверку. Как видно, случай обрабатывается правильно.

Update unittesting_console_calculator.py

Run tests on any Push event #10

Summary

Jobs

run_tests

run_tests

failed now in 11s

> Set up job

> Run actions/checkout@master

> Run actions/setup-python@v1

> Install requirements

> Run tests

1

▼ Run coverage run unittesting_console_calculator.py

2

coverage run unittesting_console_calculator.py

3

shell: /usr/bin/bash -e {0}

4

env:

5

pythonLocation: /opt/hostedtoolcache/Python/3.8.12/x64

6

Fs

7

=====

8

FAIL: test_calculator (__main__.Testing)

9

10

Traceback (most recent call last):

11

File "unittesting_console_calculator.py", line 19, in test_calculator

12

self.assertEqual(main(('8 88 8 8 8 8 8 - 7 7 7 7 7 '.replace(' ', '').replace('-', '+').replace('-', '+').split('+'))), 8811111)

13

AssertionError: 88111111 != 8811111

14

15

Ran 2 tests in 0.001s

16

FAILED (failures=1, skipped=1)

17

Error: Process completed with exit code 1.

Tests report

> Post Run actions/checkout@master

> Complete job

4. Посмотрим, какую информацию можно получить во вкладке Github Actions:

Lopa10ko / ITMO-cyberculture-2021-2022 Public

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

✓ Delete python-app.yml Run tests on any Push event #7

Summary

Jobs

✓ run_tests

run_tests

succeeded 4 days ago in 17s

> ✓ Set up job

> ✓ Run actions/checkout@master

> ✓ Run actions/setup-python@v1

> ✓ Install requirements

> ✓ Run tests

> ✓ Tests report

> ✓ Post Run actions/checkout@master

> ✓ Complete job

✓ Set up job

```
1 Current runner version: '2.285.0'
2 ▼ Operating System
3   Ubuntu
4   20.04.3
5   LTS
6 ▼ Virtual Environment
7   Environment: ubuntu-20.04
8   Version: 20211129.1
9   Included Software: https://github.com/actions/virtual-environments/blob/ubuntu20/20211129.1/images/linux/Ubuntu2004-README.md
10  Image Release: https://github.com/actions/virtual-environments/releases/tag/ubuntu20%2F20211129.1
11 ▼ Virtual Environment Provisioner
12   1.0.0.0-master-20211123-1
13 ▼ GITHUB_TOKEN Permissions
14   Actions: write
15   Checks: write
16   Contents: write
17   Deployments: write
18   Discussions: write
19   Issues: write
20   Metadata: read
21   Packages: write
22   Pages: write
23   PullRequests: write
24   RepositoryProjects: write
25   SecurityEvents: write
26   Statuses: write
27 Secret source: Actions
28 Prepare workflow directory
29 Prepare all required actions
30 Getting action download info
31 Download action repository 'actions/checkout@master' (SHA:230611dbd0eb52da1e1f4f7bc8bb0c3a339fc8b7)
32 Download action repository 'actions/setup-python@v1' (SHA:152ba7c4dd6521b8e9c93f72d362ce03bf6c4f20)
```

✓ Tests report

```
1 ▼ Run coverage report
2   coverage report
3   shell: /usr/bin/bash -e {0}
4   env:
5     pythonLocation: /opt/hostedtoolcache/Python/3.8.12/x64
6   Name                               Stmts  Miss  Cover
7   -----
8   console_calculator.py             20     13   35%
9   unittesting_console_calculator.py  18      1   94%
10  -----
11  TOTAL                               38     14   63%
```

mission passed!

RESPECT +99