Экосистема open source проектов

Полезные инструменты и сервисы

Юрий Литвинов
y.litvinov@spbu.ru

21.03.2024

Основные команды .NET Command-Line Interface

- dotnet new — создать новый проект
  - dotnet new console
- dotnet restore — получить NuGet-пакеты для текущего проекта
- dotnet build — собрать проект в текущей папке
- dotnet run — запустить проект в текущей папке
  - dotnet run -- моиАргументы
- dotnet test — запустить юнит-тесты для проекта в текущей папке

Continuous Integration

Непрерывная интеграция — практика слияния всех изменений по нескольку раз в день, сборки их в известном окружении и запуска юнит-тестов.

- Автоматический билд
  - Всё, что нужно для сборки, есть в репозитории, может быть получено на чистую (ну, практически) машину и собрано одной консольной командой
- Большое количество юнит-тестов, запускаемых автоматически
- Выделенная машина, слушающая репозиторий и выполняющая билд
  - Чаще всего каждый билд запускается на заранее настроенной виртуалке

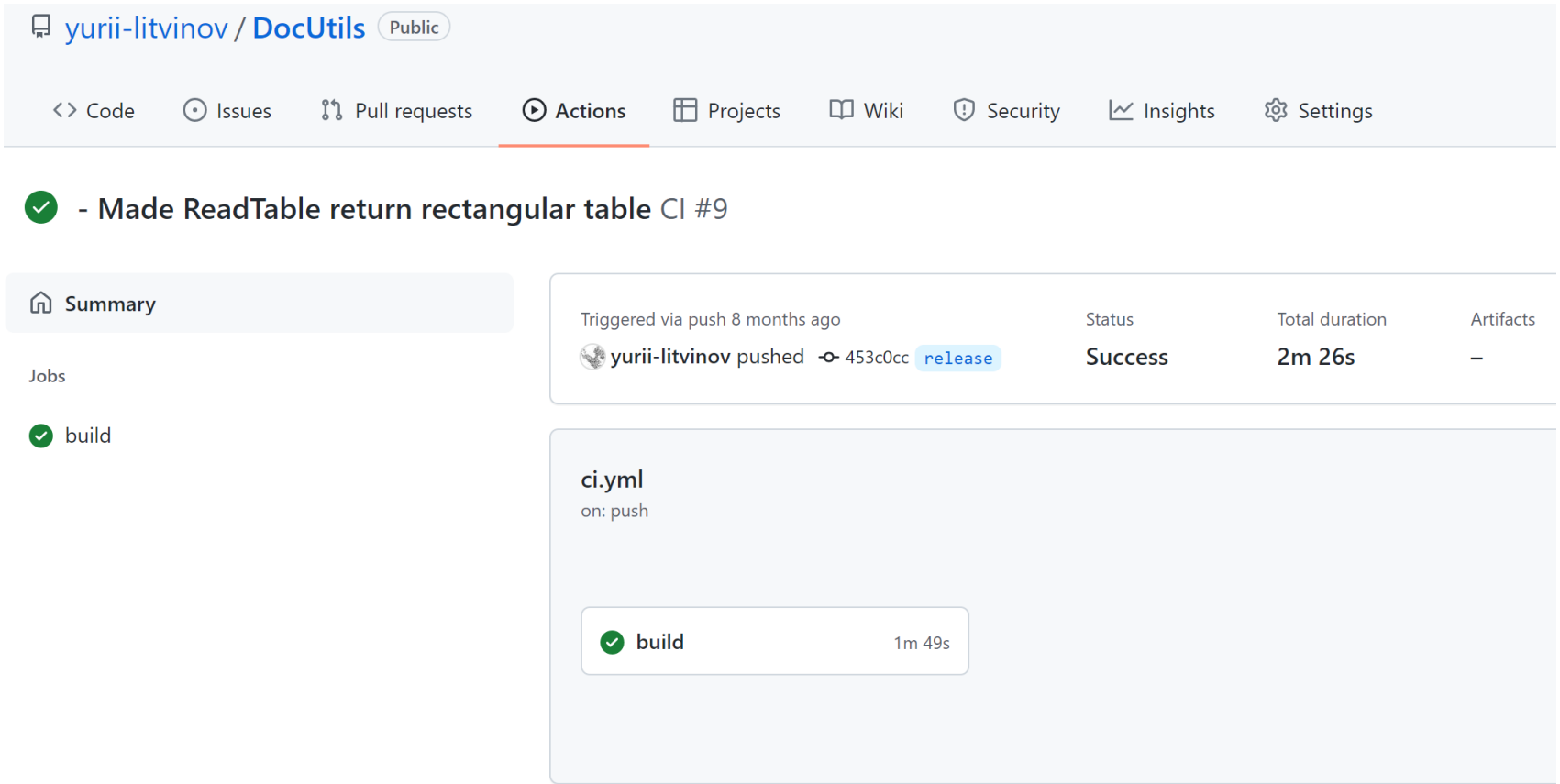
Continuous Integration

- Извещение всех разработчиков о статусе
  - Если билд не прошёл, разработка приостанавливается до его починки
- Автоматическое выкладывание
- Пока билд не прошёл, задача не считается сделанной
  - Короткие билды (<10 мин.)
  - deployment pipeline
    - Отдельная машина для сборки, для коротких тестов, для длинных тестов, для выкладывания

GitHub Actions

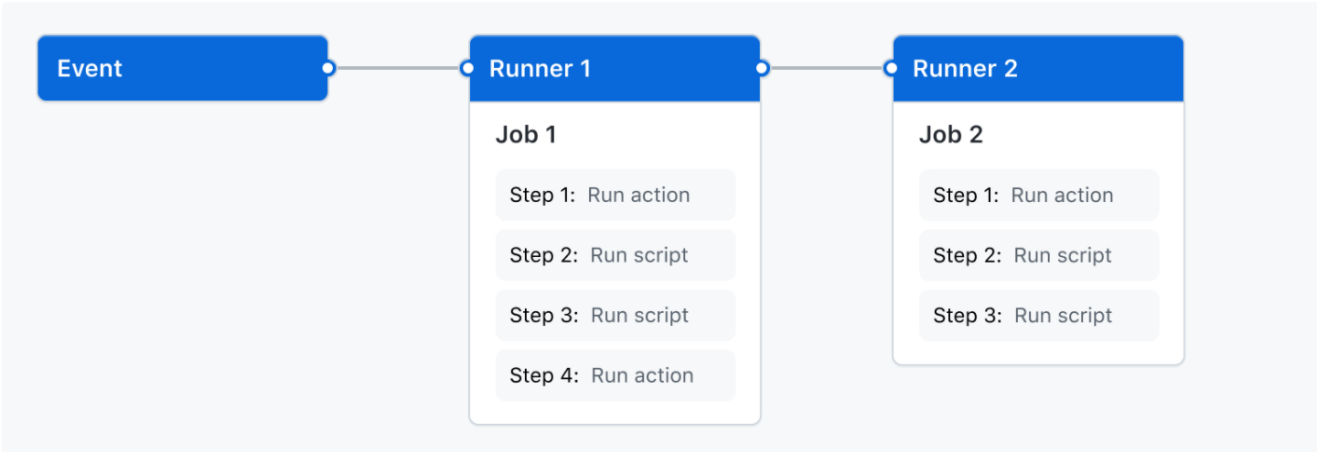
- Бесплатная система облачной сборки для проектов на GitHub
- https://docs.github.com/en/actions
- Как настроить:
  - В репозитории на GitHub Settings -> Actions -> Allow all actions
  - Создаём в корне репозитория папку .github/workflows/
  - В нём создаём файл <имя действия>.yml (например, ci.yml)
  - Описываем процесс сборки согласно https://docs.github.com/en/actions/learn-github-actions/workflow-syntax-for-github-actions
    - Пример и описание линуксовой сборки: https://www.incredibuild.com/blog/using-github-actions-with-your-c-project
  - Коммитим-пушим
  - Смотрим статус коммита и пуллреквеста

Что получится



И появятся иконки статуса рядом с коммитами и пуллреквестами

GitHub Actions, Workflow и Job



- Step — это либо скрипт, либо Action
- Action — произвольный код (по сути, отдельное приложение), выполняющийся как шаг Job-а
  - Переиспользуемый строительный блок
  - Можно переиспользовать Workflow-ы

Типичный Workflow для сборки

```yaml
name: Build
on: [push, pull_request]
jobs:
  build-Ubuntu:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-dotnet@v4
        with:
          dotnet-version: '8.x'
      - name: Build
        run: for f in $(find . -name "*.sln"); do dotnet build $f; done
      - name: Run tests
        run: for f in $(find . -name "*.sln"); do dotnet test $f; done
  build-Windows:
    runs-on: windows-latest
    steps:
      ...
      - name: Build
        run: For /R %%I in (*.sln) do dotnet build %%I
      - name: Run tests
        run: For /R %%I in (*.sln) do dotnet test %%I
```

Переменные окружения

```yaml
env:
  DAY_OF_WEEK: Monday

jobs:
  greeting_job:
    runs-on: ubuntu-latest
    env:
      Greeting: Hello
    steps:
      - name: "Say Hello Mona it's Monday"
        if: ${{ env.DAY_OF_WEEK == 'Monday' }}
        run: echo "$Greeting $First_Name. Today is $DAY_OF_WEEK!"
        env:
          First_Name: Mona
```

Матрица сборки

```yaml
runs-on: ${{ matrix.os }}
strategy:
  matrix:
    os: [ubuntu-18.04, ubuntu-20.04]
    node: [10, 12, 14]
steps:
  - uses: actions/setup-node@v2
    with:
      node-version: ${{ matrix.node }}
```

Что ещё?

- Секреты
  - **super_secret**: ${{ secrets.SUPERSECRET }}
- Кеширование промежуточных результатов
- Автоматическое развёртывание
  - В том числе, автодеплой документации на github-pages
- Проверка стиля кодирования, статический анализ кода и т.п.
  - Может быть интересно для Python-разработчиков
- Можно иметь несколько Workflow-ов в одном репозитории

Анализ тестового покрытия, CodeCov

- https://codecov.io/
- Визуализатор для функциональности компиляторов или специальных инструментов по слежению за исполнявшимися строчками
- Чем больше операторов было исполнено во время тестового прогона, тем меньше вероятность пропустить баг
  - 100% покрытие не гарантирует работоспособность программы
- Интегрируется с GitHub (комментит пуллреквесты информацией о тестовом покрытии)
- Пример конфигурации для .NET с AppVeyor:
  - https://github.com/codecov/example-csharp

Статический анализ, Codacy

- https://www.codacy.com/
- Ищет типичные ошибки: потенциальные баги, стайлгайд, мёртвый код, производительность и т.д.
- Поддерживает много языков (в том числе C#, C++, Java, Kotlin, Python, Scala)
- Не требует дополнительных манипуляций с репозиторием
- Очень настраиваема

GitHub: Issues, Projects, Wiki, Pages

- GitHub сам многое умеет
- Issues — довольно удобный багтрекер
  - Майлстоуны, дедлайны, метки на багах, возможность закрывать баги автоматически (если в сообщении коммита есть “close” или “fix” и #<номер бага>)
  - Пуллреквест тоже считается Issue
- Projects — представляет Issues в виде набора списков, между которыми их можно перетаскивать в духе Trello
- Wiki — викистраницы, куда можно выкладывать полезную информацию о проекте
  - Тоже git-репозиторий
- Pages — хостинг для статических сайтов <имя проекта>.github.io

Авторское право

- Open source-кодом можно пользоваться, только если автор явно это разрешил, так что просто код на GitHub — не совсем open source
- Бывают исключительные и личные неимущественные права
  - Личные неимущественные права неотчуждаемы
  - Исключительные права можно передать
  - Права появляются в момент создания произведения и принадлежат автору
    - Если произведение создано по служебному заданию — работодателю
    - Знак копирайта служит только для информирования, регистрация прав не требуется
  - Соавторы владеют произведением в равной степени
- Идея не охраняется, охраняется её физическое выражение

Open source-лицензии

- Лицензия — способ передачи части прав на произведение
- Пример — “Do what the **** you want to public license”
  - “Want to” может включать в себя патентование произведения и подачу в суд на автора за нарушение патента, поэтому обычно лицензии более длинны и унылы
  - В России и Европе программы не патентуют, в США — да
- Каждый нормальный open source-проект должен иметь лицензию

Open source-лицензии

- Часто используемые open source-лицензии:
  - GPL, LGPL (GPL вирусная, поэтому использовать её, внезапно, плохая практика)
  - MIT License
  - Apache License 2.0 (может применяться пофайлово)
  - BSD License (в разных вариантах)
  - The Unlicense — явная передача произведения в Public Domain
  - Семейство лицензий Creative Commons — не для софта, но хорошо подходит для ресурсов (картинок, текстов и т.д.)