Непрерывная интеграция, оформление репозитория

Юрий Литвинов
y.litvinov@spbu.ru

09.07.2025

Continuous Integration

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

- Автоматическая сборка
  - 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 сборку
  - Чаще всего каждая сборка запускается на заранее настроенной виртуальной машине или в Docker-контейнере

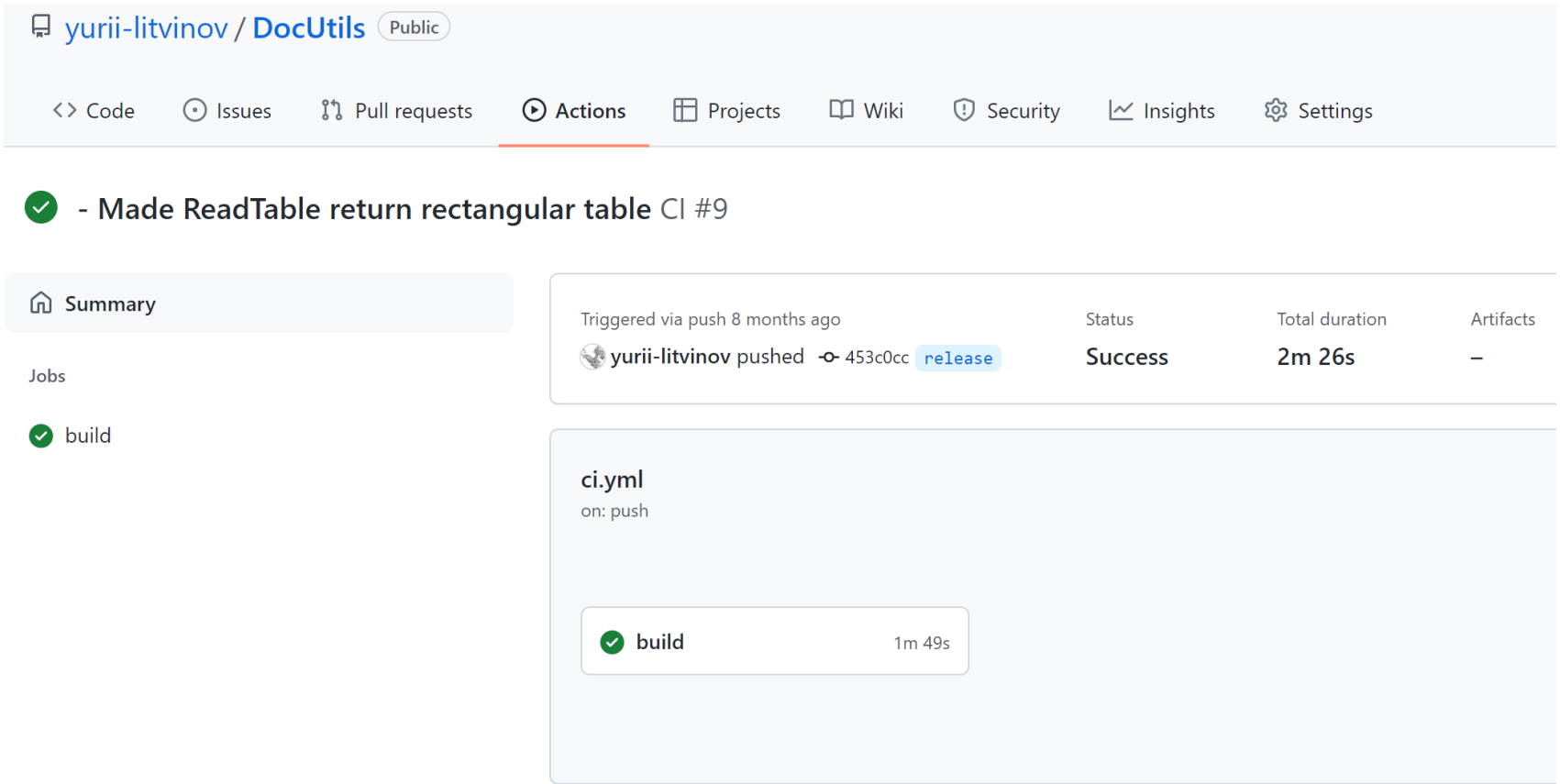
Continuous Integration

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

GitHub Actions

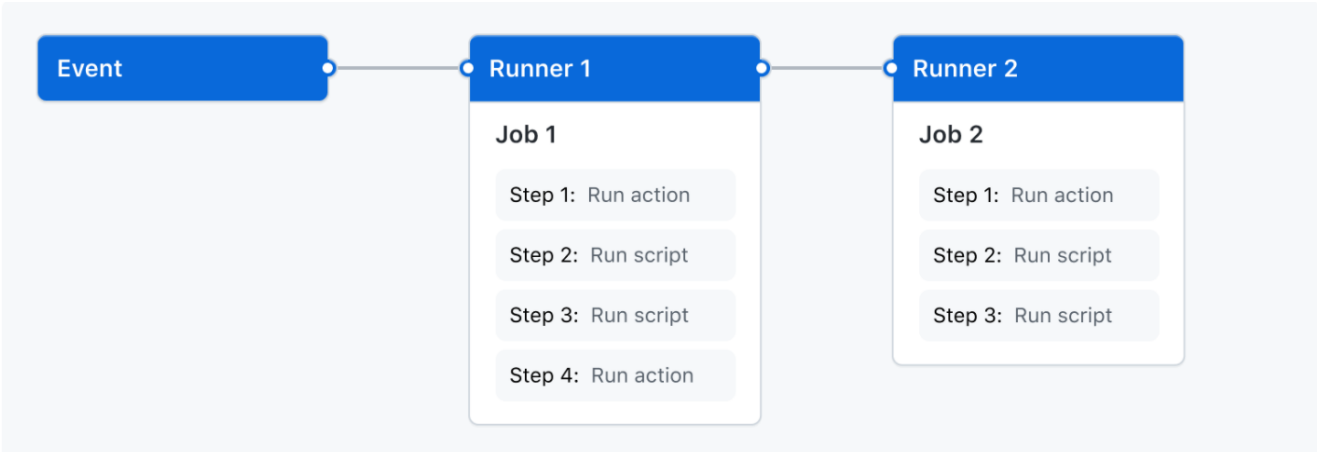
- Бесплатная система облачной сборки для проектов на GitHub
- https://docs.github.com/en/actions
- Как настроить:
  - В репозитории на GitHub Settings -> Actions -> Allow all actions
  - Создаём в корне репозитория папку .github/workflows/
  - В нём создаём файл <имя действия>.yml (например, ci.yml)
  - 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:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-dotnet@v4
        with:
          dotnet-version: '9.x'
      - name: Build and run tests
        run: dotnet test
```

Матрица сборки

```yaml
name: Build
on: [push, pull_request]
jobs:
  build:
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest]
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-dotnet@v4
        with:
          dotnet-version: '9.x'
      - name: Build and run tests
        run: dotnet test
```

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
```

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

Оформление репозитория (1)

- README.md — самая важная часть любого репозитория
  - Плашки CI и анализаторов
  - Общее описание проекта
  - Пример использования (с картинками, если уместно)
  - Как собрать и запустить
  - Если проект большой, то куда писать баги и как поучаствовать в разработке
- Настроена секция About, указаны темы (topics)
- Адекватные сообщения к коммитам
  - См. https://www.conventionalcommits.org/en/v1.0.0/
- Настроена защита веток (это в Settings на GitHub)
- Настроен Dependabot, CodeQL
- Выкладываются релизы (раздел Releases репозитория)

Оформление репозитория (2)

- .gitignore (и .gitattributes, если используете кириллицу и не хотите эльфийские руны в диффах)
  - Никаких результатов сборки в репо быть не должно
- CI обязательно
- Модульные тесты в CI — обязательно
- Линтер — обязательно
- Внешние анализаторы типа Codacy или CodeCov — опционально, но чем больше — тем лучше
- Техническая документация по коду — прямо в README, в вики или в комментариях в коде
  - Настроена автоматическая сборка и деплой документации на GitHub Pages
- Лицензия — обязательно

Оформление пуллреквеста

- Короткое, но понятное и отражающее суть дела название
- Описание пуллреквеста
  - Список предлагаемых изменений — если применимо, со ссылками на закрытые issues, в формате # номер
  - Описание работы предлагаемой функциональности:
    - Если фронтенд, с gif-кой с демо
    - Если оптимизация, то замеры производительности (статистически корректные)
    - Если бэкенд, то описание новых методов API и сценариев использования и т.д.
    - Даже если делаете пуллреквест к лучшим друзьям в репо
  - Техническое описание изменений
    - Можно диаграмму классов UML и текст, можно просто текстом
- Модульные тесты на новый код
- CI проходит, новые тесты проходят
- Проверена лицензионная совместимость
- Стиль кодирования и комментариев к коммитам соответствуют принятым
  - Если надо склеить коммиты в один, склеены в один

Лицензия

- 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.)