

Системы непрерывной интеграции CI

Юрий Литвинов
y.litvinov@spbu.ru

Continuous Integration (1)

Непрерывная интеграция — практика слияния всех изменений по несколько раз в день, сборки их в известном окружении и запуска юнит-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 виртуалке или контейнере

Continuous Integration (2)

- ▶ 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 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

📁 yurii-litvinov / DocUtils Public

<> Code
🗨 Issues
🔗 Pull requests
🔧 Actions
📁 Projects
📖 Wiki
🛡 Security
📊 Insights
⚙ Settings

✅ - Made ReadTable return rectangular table CI #9

🏠 Summary

Jobs

✅ build

| | | | |
|--|---------|----------------|-----------|
| Triggered via push 8 months ago | Status | Total duration | Artifacts |
| yurii-litvinov pushed 453c0cc release | Success | 2m 26s | — |

ci.yml
on: push

✅ build 1m 49s

И появятся иконки статуса рядом с коммитами и пуллреквестами

Типичный Workflow для сборки

name: Build

on: [push, pull_request]

jobs:

build-Ubuntu:

runs-on: ubuntu-latest

steps:

- **uses:** actions/checkout@v2
- **uses:** actions/setup-dotnet@v1

with:

dotnet-version: '6.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:

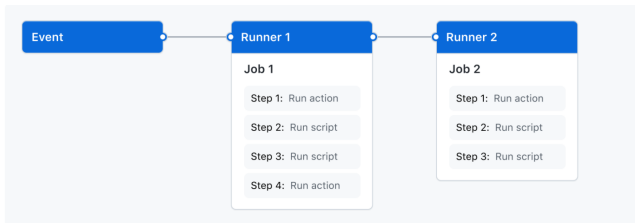
runs-on: windows-latest

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

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

Переменные окружения

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

```
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

AppVeyor

- ▶ <https://www.appveyor.com/> — отдельная облачная CI-система, тоже довольно неплоха и проще в настройке
- ▶ Виртуальная машина с ОС Windows и настроенными инструментами сборки .NET-приложений
 - ▶ Windows Server 2019 + VS 2022 или более старые
 - ▶ Умеет Linux Ubuntu 20.04 и macOS 12.2.1
- ▶ Интегрируется с GitHub-ом, Slack-ом, умеет деплоить
- ▶ Собирает по умолчанию системой сборки MSBuild
 - ▶ Можно переубедить и собирать что угодно
- ▶ Окружение настраивается конфигурационным файлом или «вручную» из скрипта сборки

AppVeyor, настройка сборки

- ▶ Зайти на <https://www.appveyor.com/> по GitHub-аккаунту
- ▶ Добавить проект (разрешив AppVeyor просматривать список репозитория на гитхабе)
- ▶ Положить в корень репозитория файл `appveyor.yml` с конфигурацией сборки
 - ▶ Пустой тоже ок, это конфигурация по умолчанию, ищет `.sln` в корне репозитория и пытается его собрать
- ▶ Закоммитить и запустить, это инициирует процесс сборки
- ▶ Результаты будут видны прямо на GitHub, у каждого коммита и в пуллреквесте:



All checks have passed

2 successful checks

[Hide all checks](#)



continuous-integration/appveyor/pr — AppVeyor build succeeded

[Details](#)



continuous-integration/travis-ci/pr — The Travis CI build passed

[Details](#)

AppVeyor, пример файла конфигурации

image: Visual Studio 2022

before_build:

- nuget restore myCoolHomework/Homework.sln

build:

project: myCoolHomework/Homework.sln

test_script:

- dotnet test myCoolHomework/Homework.sln