

# Сборка и непрерывная интеграция

Юрий Литвинов  
y.litvinov@spbu.ru

08.09.2022

# Системы сборки

- ▶ Среда разработки не всегда доступна
  - ▶ Continuous Integration-сервера автоматически выполняют сборку после каждого коммита, там некому открыть IDE и нажать на кнопку “запустить”
- ▶ Воспроизводимость сборки
  - ▶ Если чтобы собрать программу надо открыть проект, скопировать пару десятков файлов, поправить кое-какие пути и делать это в полнолуние, то возможны ошибки
- ▶ Автоматизация сборки
  - ▶ `git clone`
  - ▶ одна консольная команда, которая всё делает за нас
  - ▶ ...
  - ▶ готовое к работе приложение

# Сборка вручную без IDE

- ▶ gcc  
    g++ <имя .c-файла>  
или, например,  
    g++ -Wall -o helloworld helloworld.c
- ▶ Если проект большой, это быстро становится грустно
  - ▶ Десятки тысяч файлов — не редкость

# make

- ▶ Стандарт де-факто по “низкоуровневым” правилам сборки
- ▶ Сама ничего не знает про языки программирования, компиляторы и прочие подобные штуки
- ▶ Знает про цели, зависимости, временные штампы и правила
  - ▶ Смотрит на зависимости цели, если у хоть одной временной штамп свежее цели, запускается правило для цели
  - ▶ В процессе цель может обновить свой временной штамп, что приведёт к исполнению правил для зависящих от неё целей
  - ▶ Цели и зависимости образуют направленный ациклический граф (DAG)
  - ▶ make выполняет топологическую сортировку графа зависимостей
  - ▶ Правила применяются в порядке от листьев к корню
- ▶ Правила сборки описываются в Makefile

# Пример

```
target [target ...]: [component ...]  
  [command 1]  
  .  
  .  
  .  
  [command n]
```

Пример:

```
hello: ; @echo "hello"
```

# Высокоуровневые системы сборки

- ▶ C/C++:
  - ▶ CMake — кроссплатформенная система сборки, очень популярна в C++ open source-сообществе
  - ▶ MSBuild — система сборки Visual Studio
  - ▶ qmake, qbs — системы сборки от фреймворка Qt
- ▶ JVM:
  - ▶ Maven — старая, но популярная и “архитектурно правильная”
  - ▶ Gradle — несколько более “императивна”, нынче более популярна, поддерживает Kotlin
- ▶ .NET — dotnet CLI, FAKE

Написание скриптов сборки для большого проекта — отдельная и довольно трудоёмкая задача

# Continuous Integration

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 в Docker-контейнере

# 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

The screenshot shows a GitHub repository page for 'yurii-litvinov / DocUtils'. The 'Actions' tab is selected, displaying a workflow run titled '- Made ReadTable return rectangular table CI #9'. The workflow status is 'Success'. The 'Summary' section shows the workflow was triggered via a push 8 months ago. The 'Jobs' section lists a single job named 'build' with a status of 'Success' and a duration of '1m 49s'. The workflow file 'ci.yml' is shown as being triggered 'on: push'.

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 <a href="#">release</a>	Success	2m 26s	—

ci.yml  
on: push

✔ build 1m 49s

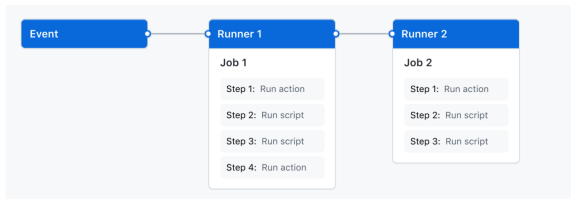
И появятся иконки статуса рядом с коммитами и пуллреквестами

# Типичный Workflow для сборки

## Java

```
name: Build
on: [push, pull_request]
jobs:
  build-Ubuntu:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK
        uses: actions/setup-java@v1
        with:
          java-version: 17
      - name: Build
        run: ./gradlew build
      - name: Run tests
        run: ./gradlew test
  build-Windows:
    runs-on: windows-latest
    steps:
      ...
      - name: Build
        run: .\gradlew.bat build
      - name: Run tests
        run: .\gradlew.bat test
```

# GitHub Actions, Workflow и Job



- ▶ Step — это либо скрипт, либо *Action*
- ▶ Action — произвольный код (по сути, отдельное приложение), выполняющийся как шаг Job-a
  - ▶ 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