CI для C++ разработчика

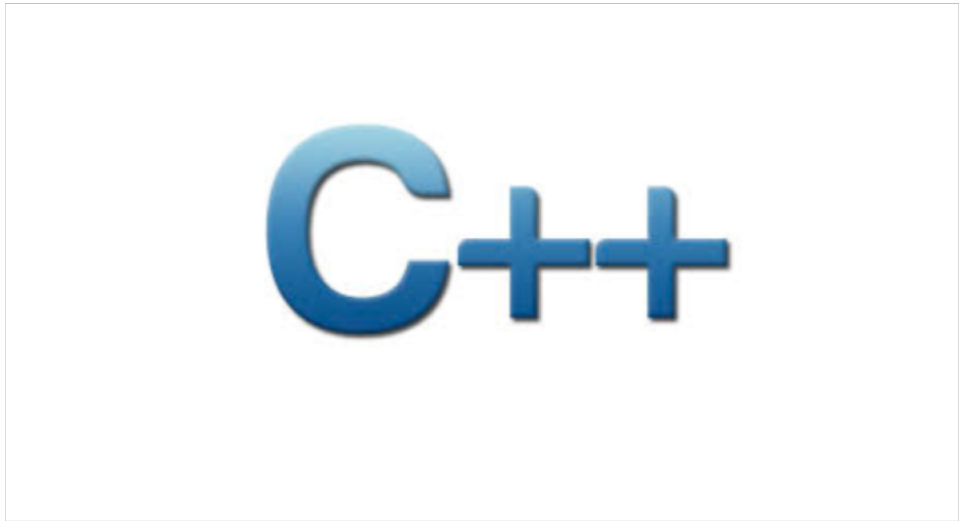
Pavel.Filonov@kaspersky.com

https://t.me/joinchat/D2rdyxHzWdgCEtaqCbe1Zw

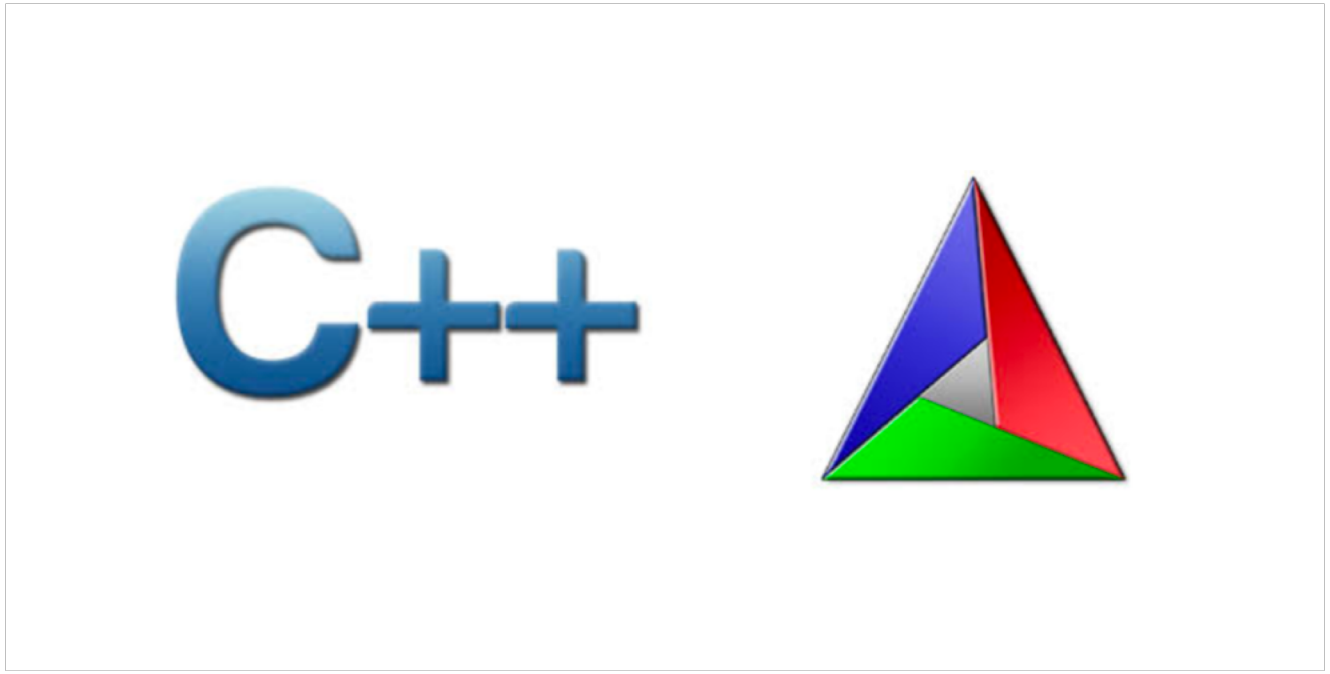
https://github.com/sdukshis/ci-corehard

Зачем мы здесь?

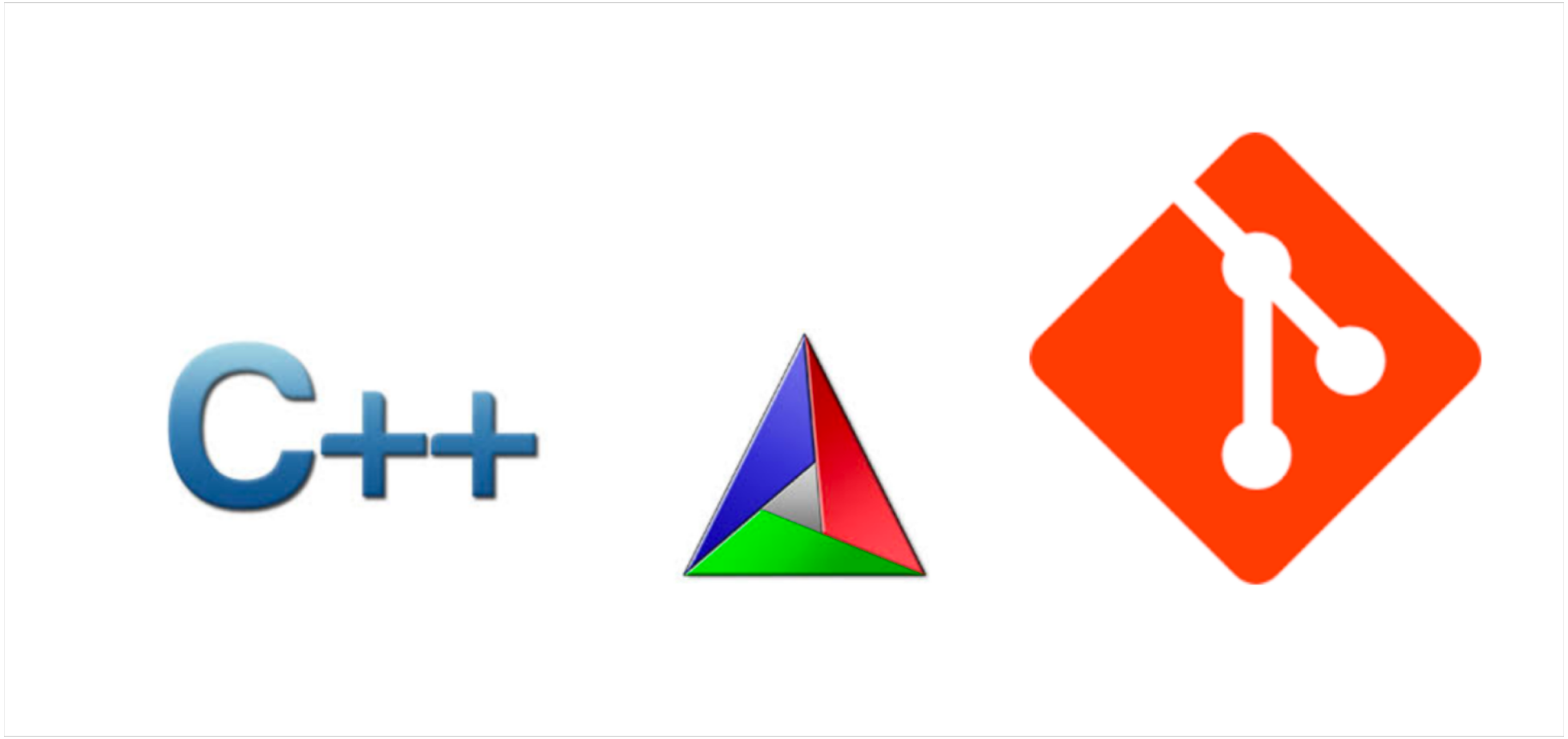
Как все начиналось



Как все начиналось



Как все начиналось



Наши дни

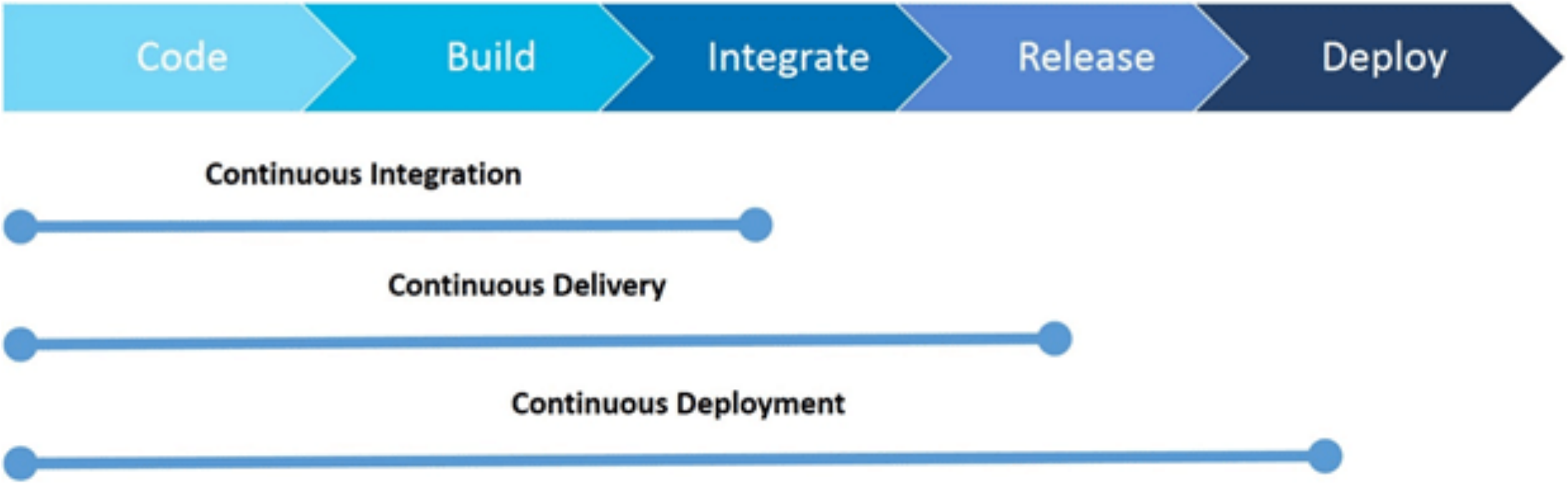
План

1. Основные термины
2. Работаем с GitHub
    1. Hello, world!
    2. Запускаем сборку на TravisCI и Appveyour
    3. Разрешаем зависимости с помощью conan.io
    4. Настраиваем сборку под множество целей
    5. Собираем свои пакеты с conan.io
3. Работаем с GitLab
4. Разворачиваем Gitlab с помощью VPS и Docker
    1. Настраиваем хранение артефактов
    2. Настраиваем Gitlab CI
5. Заключение

Основные термины

- **Непрерывная интеграция** (Continuous integration, CI) — практика, в которой исходный код разработчиков постоянно интегрируется в основную ветку разработки.
- **Непрерывная доставка** (Continuous delivery, CD) — практика, которая обеспечивает возможность выпуска продукта в любое время.
- **Непрерывное развертывание** (Continuous deployment, CD) — практика, в которой версия продукта обновляется автоматически с добавлением в основную ветку нового функционала.

CI/CD workflows



Ref: [4]

Что нам потребуется

- умение программировать на C++ (спасибо, Кэп!)
- laptop с любой ОС
- один из компиляторов:
  - Visual Studio >= 2015
  - g++ >= 5.4
  - clang++ >= 3.9
  - xcode >= 8.0
- cmake >= 3.6 (нужно понимать синтаксис CMakeLists.txt)
- git >= 2.10
- python >= 3.5
- pip3 >= 9.0
- учетная запись на github.com
- учетная запись на travis-ci.org (привязанная к github)
- учетная запись на appveyor.com (привязанная к github)
- учетная запись на bintray.com

Репозиторий с примерами

- git clone https://github.com/sdukshis/ci-corehard.git
- cd ci-corehard
- git tag
  - 01_hello
  - 02_simple_travis
  - 03_conan_travis
  - 04_multi_toolchains
  - 05_osx_travis
  - 06_appveyor
  - 07_conan_package_tools
  - 08_gitlab_ci
  - 09_gitlab_conan_upload
- Переключиться на первый шаг git tag checkout 01_hello

Пишем hello, world

git checkout 01_hello

1. Создаем репозиторий на github.com
2. Клонируем его на машину разработчика
   git clone https://github.com/<username>/<repo>.git
3. Создаем структуру проекта

```
hello/
|-- CMakeLists.txt
|-- LICENSE
|-- README.md
|-- src
        |-- hello.cpp
```

4. Пишем hello.cpp

```cpp
#include <iostream>

int main() {
        std::cout << "Hello, world!" << std::endl;
}
```

Пишем hello, world

- CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.0 FATAL_ERROR)
project(hello CXX)
set(SRCS
    src/hello.cpp
)
add_executable(${PROJECT_NAME} ${SRCS})
set_property(TARGET ${PROJECT_NAME} PROPERTY CXX_STANDARD 14)
```

- Собираем
  - mkdir build && cd build
  - cmake ..
  - cmake --build .
  - ./hello

Подключаем TravisCI

git checkout 02_simple_travis

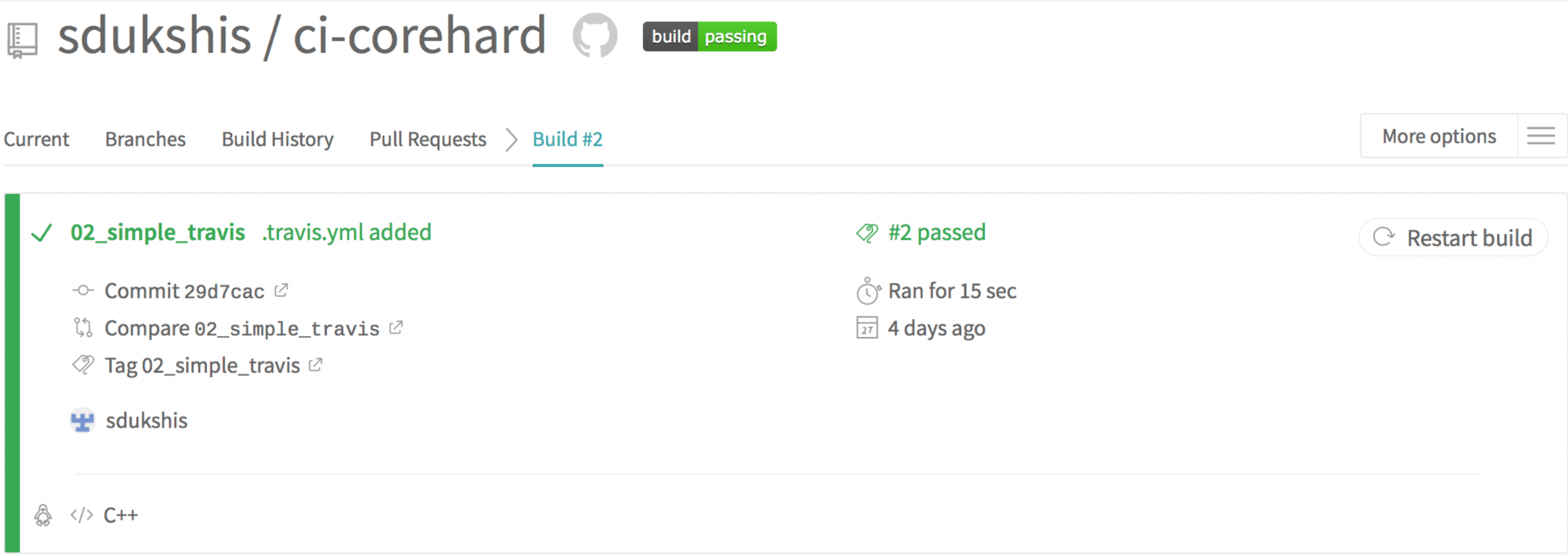
- Регистрируемся на travis-ci.org
- Подключаем к репозиторию сборку на travis
- Добавляем .travis.yml

```
language: cpp

script:
  - mkdir build && cd build
  - cmake ..
  - cmake --build .
  - ./hello
```

- Добавляем ссылку на бедж в README.md
- Фиксируем изменения
  - git add .
  - git commit –am”Initial commit”
- Отправляем изменения на github
  - git push origin master
- Смотрим результаты сборки на travis-ci.org

Успех!


sdukshis / ci-corehard
build passing
Current
Branches
Build History
Pull Requests
Build #2
More options
02_simple_travis .travis.yml added
#2 passed
Restart build
Commit 29d7cac
Compare 02_simple_travis
Tag 02_simple_travis
Ran for 15 sec
4 days ago
sdukshis
C++

Добавляем зависимости с conan

git checkout 03_conan_travis

1. Устанавливаем conan
   - Windows – C:\Python<…>\Scripts\pip install conan
   - MacOS & Linux – pip3 install conan
2. Пишем conanfile.txt

```
[requires]
gtest/1.8.1@bincrafters/stable

[generators]
cmake
```

3. Подключаем к CMakeLists.txt

```
project(hello CXX)

include(${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
conan_basic_setup()
```

Пишем тесты

- Новая структура каталогов

```
.
├── CMakeLists.txt
├── LICENSE
├── README.md
├── conanfile.txt
├── fmt.sh
├── include
│   └── hello
│       └── hello.h
├── src
│   ├── hello.cpp
│   └── main.cpp
└── tests
    └── test_hello.cpp
```

Пишем тесты

- hello.h

```
#pragma once

#include <ostream>

namespace hello {

std::ostream& greetings(std::ostream&);

}
```

Пишем тесты

- hello.cpp

```
#include <hello/hello.h>


namespace hello {


std::ostream& greetings(std::ostream& stream) {
    return stream << "Hello, corehard";
}


}
```

Пишем тесты

- hello.cpp

```
#include <sstream>


#include <gtest/gtest.h>


#include <hello/hello.h>


TEST(hello, greetings) {
    std::stringstream ss;
    hello::greetings(ss);
    ASSERT_FALSE(ss.str().empty());
}
```

Пишем тесты

- CMakeLists.txt

```
set(LIB_SRCS src/hello.cpp)
add_library(${PROJECT_NAME} ${LIB_SRCS})
set_property(TARGET ${PROJECT_NAME} PROPERTY CXX_STANDARD 14)
target_include_directories(${PROJECT_NAME} PUBLIC ${CMAKE_SOURCE_DIR} include)
set(TEST_SRCS tests/test_hello.cpp)
add_executable(test_${PROJECT_NAME} ${TEST_SRCS})
target_link_libraries(test_${PROJECT_NAME}
    ${PROJECT_NAME}
    ${CONAN_LIBS}
)
```

Пишем тесты

- правим .travis-ci.yml

```
language: cpp


install:
  # Install conan
  - pip install --user -q conan
  # Automatic detection of your arch, compiler, etc.
  - conan user


script:
  - mkdir build && cd build
  - conan install .. --build
  - cmake ..
  - cmake --build .
  - ./bin/test_hello
```

- git push
- Смотрим лог сборки на travisCI

Выбираем версию компилятора

git checkout 04_multi_toolchains

- .travis-ci.yml

```
language: cpp
dist: trusty

before_install:
   - eval "${MATRIX_EVAL}"

install:
  # Install conan
  - pip install --user -q conan
  # Automatic detection of your arch, compiler, etc.
  - conan user

script:
  - mkdir build && cd build
  - conan install .. --build
  - cmake ..
  - cmake --build .
  - ./bin/test_hello
```

Выбираем версию компилятора

- .travis-ci.yml

```
matrix:
 include:
  - os: linux
    addons:
     apt:
      sources:
       - ubuntu-toolchain-r-test
      packages:
       - gcc-5
       - g++-5
    env:
     - MATRIX_EVAL="CC=gcc-5 && CXX=g++-5"
  - os: linux
    addons:
     apt:
      sources:
       - ubuntu-toolchain-r-test
       - llvm-toolchain-trusty-6.0
      packages:
       - clang-6.0
    env:
     - MATRIX_EVAL="CC=clang-6.0 && CXX=clang++-6.0"
```

sdukshis / ci-corehard

build passing

Current Branches Build History Pull Requests > Build #22 More options

✓ **04_multi_toolchains** multi toolchains

#22 passed

Restart build

Commit 6e6e6b4

Compare 04_multi_toolchains

Tag 04_multi_toolchains

Ran for 2 min 27 sec

Total time 9 min 43 sec

4 days ago

sdukshis

Build Jobs

✓	# 22.1	</> C++	MATRIX_EVAL="CC=gcc-5 && CXX=g++-5"	2 min 9 sec
✓	# 22.2	</> C++	MATRIX_EVAL="CC=gcc-6 && CXX=g++-6"	2 min 16 sec
✓	# 22.3	</> C++	MATRIX_EVAL="CC=gcc-7 && CXX=g++-7"	2 min 5 sec
✓	# 22.4	</> C++	MATRIX_EVAL="CC=clang-6.0 && CXX=clang++-6.	1 min 38 sec
✓	# 22.5	</> C++	MATRIX_EVAL="CC=clang-7 && CXX=clang++-7"	1 min 35 sec

Подключаем сборки под MacOS

git checkout 05_osx_travis

- .travis-ci.yml

```
sudo: required
install:
  # Install conan
  - sudo -E pip install conan
  # Automatic detection of your arch, compiler, etc.
  - conan user
matrix:
  include:
    - os: osx
      osx_image: xcode9.4
      env:
        - MATRIX_EVAL="CC=clang && CXX=clang++"
```

sdukshis / ci-corehard

build passing

Current Branches Build History Pull Requests > Build #29 More options

✓ **05_osx_travis** multi os

- Commit `c22406b`
- Compare `05_osx_travis`
- Tag 05_osx_travis
- sdukshis

#29 passed

- Ran for 3 min 40 sec
- Total time 5 min 45 sec
- 4 days ago

Restart build

Build Jobs

✓	# 29.1	</> C++	MATRIX_EVAL="CC=gcc-7 && CXX=g++-7"	2 min 51 sec
✓	# 29.2	</> C++	MATRIX_EVAL="CC=clang-7 && CXX=clang++-7"	1 min 41 sec
✓	# 29.3	</> Xcode: xcode9.4 C++	MATRIX_EVAL="CC=clang && CXX=clang++"	1 min 13 sec

Добавляем сборку под Windows

git checkout 06_appveyor

1. создаем учетную запись на appveyor.com
2. подключаем к репозиторию сборку
3. добавляем ссылку на бедж в README.md
4. пишем appveyor.yml

```
build: false
platform: x64
install:
  - set PATH=%PATH%;%PYTHON%/Scripts/
  - pip.exe install conan
  - conan user # Create the conan data directory
build_script:
  - mkdir build && cd build
  - conan install .. --build
  - cmake .. -G "%CMAKE_GENERATOR%"
  - cmake --build . --config Release
test_script:
  - cd bin && test_hello.exe
environment:
  matrix:
    - APPVEYOR_BUILD_WORKER_IMAGE: Visual Studio 2015
      CMAKE_GENERATOR: Visual Studio 14 2015 Win64
    - APPVEYOR_BUILD_WORKER_IMAGE: Visual Studio 2017
      CMAKE_GENERATOR: Visual Studio 15 2017 Win64
```

ci-corehard

Current build | History | Deployments | Events | Settings

NEW BUILD | RE-BUILD COMMIT

appveyor multi toolchain 1.0.9

4 days ago by Pavel Filonov — master — 9ce34c6f — 4 days ago in 6 min 42 sec

Jobs

Job name	Tests	Duration
Environment: APPVEYOR_BUILD_WORKER_IMAGE=Visual Studio 2015, CMAKE_GENERATOR=Visual Studio 14 2015…		2 min 37 sec
Environment: APPVEYOR_BUILD_WORKER_IMAGE=Visual Studio 2017, CMAKE_GENERATOR=Visual Studio 15 2017…		3 min 18 sec

Собираем пакеты для conan

git checkout 07_conan_package_tools

1. conan new -t -cilg -cilc -cio -ciw hello/0.1.0

    File saved: .travis.yml
    File saved: .travis/install.sh
    File saved: .travis/run.sh
    File saved: appveyor.yml
    File saved: build.py
    File saved: conanfile.py
    File saved: test_package/CMakeLists.txt
    File saved: test_package/conanfile.py
    File saved: test_package/example.cpp

conanfile.py

```
Class HelloConan(ConanFile):
    name = "hello"
    version = "0.1.0"
    license = "MIT"
    url = "https://github.com/sdukshis/ci-corehard"
    description = "Demo CI/CD workshop project"
    settings = "os", "compiler", "build_type", "arch"
    generators = "cmake"
    requires = "gtest/1.8.1@bincrafters/stable"
    exports_sources = [
        "include/*",
        "src/*",
        "tests/*",
        "CMakeLists.txt",
    ]

    def build(self):
        cmake = CMake(self)
        cmake.configure()
        cmake.build()
```

Собираем пакеты для conan

1. смотрим
   - .travis-ci.yml
   - appveyor.yml
     - Убрать (из-за gtest)
       - APPVEYOR_BUILD_WORKER_IMAGE: Visual Studio 2015
         CONAN_VISUAL_VERSIONS: 12
   - build.py

   - test_package/CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.12)
project(test_hello CXX)

include(${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
conan_basic_setup()

add_executable(${PROJECT_NAME} ../tests/test_hello.cpp)
target_link_libraries(${PROJECT_NAME} ${CONAN_LIBS})
```

Собираем пакеты для conan

- test_package/conanfile.py

```
class HelloTestConan(ConanFile):
    settings = "os", "compiler", "build_type", "arch"
    generators = "cmake"
    requires = "gtest/1.8.1@bincrafters/stable"

    def build(self):
        cmake = CMake(self)
        # Current dir is "test_package/build/<build_id>" and CMakeLists.txt is
        # in "test_package"
        cmake.configure()
        cmake.build()

    def imports(self):
        self.copy("*.dll", dst="bin", src="bin")
        self.copy("*.dylib*", dst="bin", src="lib")
        self.copy('*.so*', dst='bin', src='lib')

    def test(self):
        os.chdir("bin")
        self.run(".%stest_hello" % os.sep)
```

Build Jobs

✓	# 33.1	Python: 3.6	CONAN_GCC_VERSIONS=4.9 CONAN_DOCKER_I	2 min 13 sec
✓	# 33.2	Python: 3.6	CONAN_GCC_VERSIONS=5 CONAN_DOCKER_IMA	2 min 13 sec
✓	# 33.3	Python: 3.6	CONAN_GCC_VERSIONS=6 CONAN_DOCKER_IMA	2 min 20 sec
✓	# 33.4	Python: 3.6	CONAN_GCC_VERSIONS=7 CONAN_DOCKER_IMA	2 min 37 sec
✓	# 33.5	Python: 3.6	CONAN_CLANG_VERSIONS=3.9 CONAN_DOCKER	2 min 45 sec
✓	# 33.6	Python: 3.6	CONAN_CLANG_VERSIONS=4.0 CONAN_DOCKER	2 min 33 sec
✓	# 33.7	Xcode: xcode7.3	CONAN_APPLE_CLANG_VERSIONS=7.3	7 min 10 sec
✓	# 33.8	Xcode: xcode8.3	CONAN_APPLE_CLANG_VERSIONS=8.1	7 min 7 sec
✓	# 33.9	Xcode: xcode9.2	CONAN_APPLE_CLANG_VERSIONS=9.0	7 min 35 sec
✓	# 33.10	Xcode: xcode9.3	CONAN_APPLE_CLANG_VERSIONS=9.1	8 min 4 sec

Conan package tools

Linux gcc 7

```
 >> Running builds...
Page: 1/1
+--------+--------------+--------------------+------------+
| arch   | build_type   |   compiler.version | compiler   |
|--------+--------------+--------------------+------------|
| x86    | Release      |                  7 | gcc        |
| x86    | Debug        |                  7 | gcc        |
| x86_64 | Release      |                  7 | gcc        |
| x86_64 | Debug        |                  7 | gcc        |
+--------+--------------+--------------------+------------+
```

Conan package tools

Mac os xcode 9.2

```
Page: 1/1
+--------------------+--------+--------------+-------------+
|   compiler.version | arch   | build_type   | compiler    |
|--------------------+--------+--------------+-------------|
|                  9 | x86_64 | Release      | apple-clang |
|                  9 | x86_64 | Debug        | apple-clang |
+--------------------+--------+--------------+-------------+
```

Conan package tools

Visual Studio 2017

```
+--------------------+---------------+--------+--------------+--------------------+
|   compiler.version | compiler      | arch   | build_type   | compiler.runtime   |
|--------------------+---------------+--------+--------------+--------------------|
|                 15 | Visual Studio | x86    | Release      | MT                 |
|                 15 | Visual Studio | x86    | Release      | MD                 |
|                 15 | Visual Studio | x86    | Debug        | MTd                |
|                 15 | Visual Studio | x86    | Debug        | MDd                |
|                 15 | Visual Studio | x86_64 | Release      | MT                 |
|                 15 | Visual Studio | x86_64 | Release      | MD                 |
|                 15 | Visual Studio | x86_64 | Debug        | MTd                |
|                 15 | Visual Studio | x86_64 | Debug        | MDd                |
+--------------------+---------------+--------+--------------+--------------------+
```

Uploading conan packages

- CONAN_UPLOAD – url from bintray
- CONAN_LOGIN_USERNAME – учетная запись на bintray
- CONAN_PASSWORD – не хранить в исходных кодах! Использовать secret variables

- https://docs.conan.io/en/latest/uploading_packages/bintray/uploading_bintray.html

Приватные репозитории github

Developer

$7

per month

Includes:

Personal account
Unlimited public repositories
Unlimited private repositories
Unlimited collaborators

Free for students as part of the Student Developer Pack.

Upgrade your account

Team

$9

per user / month

Includes:

Unlimited public repositories
Unlimited private repositories
Team and user permissions

Starting at $25 / month which includes your first 5 users.

Sign up your team

Business Cloud

$21

per user / month

SAML single sign-on
Access provisioning
24/5 support with 8-hour response time
99.95% uptime SLA

Get started

Enterprise

Contact Sales

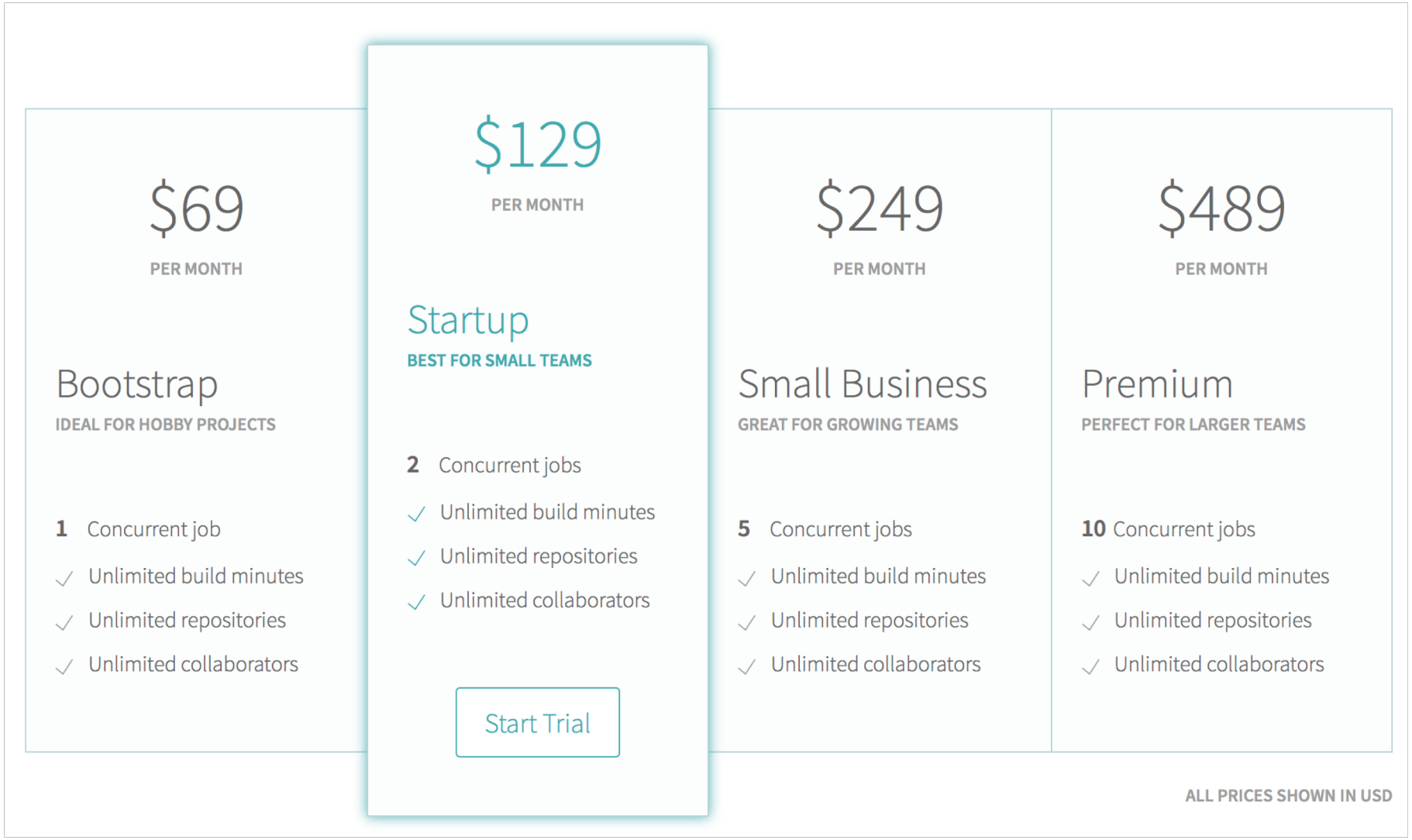
for pricing*

Self-hosted
SAML, LDAP, and CAS
Access provisioning
24/7 support for urgent issues
Advanced auditing

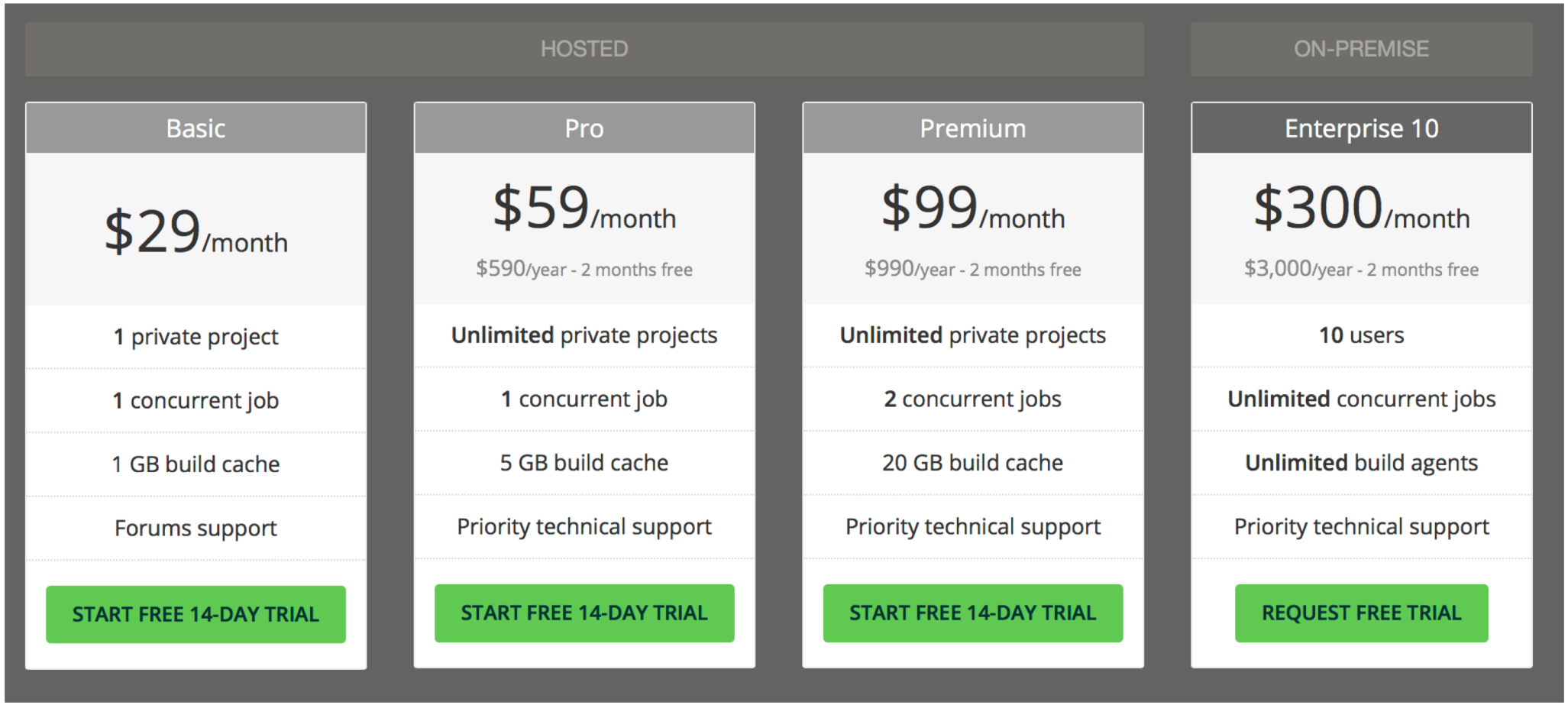
Contact us to learn more

Start a free trial

Приватные сборки travis



Приватные сборки appveyor



Gitlab

- легко развернуть самостоятельно
- удобный CI
- тикеты
- code review
- wiki
- etc.

Нам нужен VPS

Создаем VPS и ставим docker

1. Создаем учетку на sprintbox.ru с промокодом
2. Для Windows ставим PuTTy
3. В письме находим адрес сервера и пароль root
4. Логинимся на сервер
   - ssh root@<server ip>
5. Ставим docker
   - apt update && apt install docker.io

Запускаем Gitlab

git checkout 08_gitlab_ci

```
GITLAB_HOST="185.185.69.25"

docker run \
  --detach \
  --hostname $GITLAB_HOST \
  --env GITLAB_OMNIBUS_CONFIG="external_url 'http://$GITLAB_HOST/'; gitlab_rails['gitlab_shell_ssh_port'] = 2222;" \
  --publish 443:443 --publish 80:80 --publish 2222:22 --publish 5005:5005 \
  --name gitlab \
  --restart always \
  --volume /srv/gitlab/config:/etc/gitlab \
  --volume /srv/gitlab/logs:/var/log/gitlab \
  --volume /srv/gitlab/data:/var/opt/gitlab  \
  gitlab/gitlab-ce:latest
```

- Ждем ~ 5 минут. Можно посмотреть docker stats и дождаться когда закончится потребление CPU
- !!! при настоящем развертывании необходимо добавить HTTPS с использованием сертификатов. Например Let’s Encrypt

Создаем репозиторий

- Заходим на http://<server ip>
- Меняем пароль для root
- Создаем проект
- Подключаем в локальном репозитории новый remote
- git push

Запускаем gitlab CI

```
docker run \
 --detach \
 --name gitlab-runner \
 --restart always \
 --volume /var/run/docker.sock:/var/run/docker.sock \
 --volume /srv/gitlab-runner/config:/etc/gitlab-runner \
 --privileged \
 gitlab/gitlab-runner
```

```
docker exec -ti gitlab-runner gitlab-runner register --docker-privileged
```

- Используем url и token из настроек проекта Project->Settings->CI/CD->Runners->Expand

Запускаем gitlab CI

- добавляем .gitlab-ci.yml

```
variables:
  CONAN_USERNAME: "admin"
  CONAN_REFERENCE: "hello/0.1.0"
  CONAN_CHANNEL: "testing"
  CONAN_LOGIN_USERNAME: "admin"

.build-template: &build-template
  before_script:
    - sudo pip install --upgrade conan_package_tools
    - conan user
  script:
    - python build.py


gcc-6:
  image: lasote/conangcc6
  variables:
    CONAN_GCC_VERSIONS: "6"
  <<: *build-template
```

6 jobs from `master` in 8 minutes and 41 seconds (queued for 38 seconds)

`13481a60`

Pipeline Jobs 6

Test

- clang-3.9
- clang-4.0
- gcc-4.9
- gcc-5
- gcc-6
- gcc-7

Запускаем artifactory

git checkout 09_gitlab_conan_upload

https://jfrog.com/open-source/#conan

```
docker run \
    --detach \
    --name artifactory \
    --restart always \
    --publish 8081:8081 \
    --volume artifactory-data:/var/opt/jfrog/artifactory \
    docker.bintray.io/jfrog/artifactory-cpp-ce
```

- проверяем http://<server ip>:8081
- меняем пароль
- создаем conan репозиторий

Заливаем conan на artifactory

- .gitlab-ci.yml

  CONAN_UPLOAD: http://<server_ip>:8081/artifactory/api/conan/conan-local

  CONAN_LOGIN_USERNAME: "admin”

- пароль задаем через gitlab->project->settings->CI/CD->Variables

  CONAN_PASSWORD

- git push

Artifact Repository Browser

Set Me Up Deploy

Tree Simple

- conan-local
  - admin/hello/0.1.0/stable
    - export
    - package
      - 0abfb5c7087c97923a26001e416471810ba91662
      - 2bb0719db63a6b78771b0dfa1e0909fa692b8a85
      - 2e9f8ed219bde1a194c20f74f9da53d19d9853d1
      - 9b9cc7f6d20c2e415eef11336f9cc5c625df609b
      - 89b82d879dd3e27c641fee17f5547049c53a6d3f
      - 097b0c9c27b6aa920964fa78d6b1d959bfc9f0a1
      - 132d8fd2a49c6ad4f1e0def10ebfee33bdb80261
      - 177f49616759ed025c15cc4a4b05d05383d777d3
      - 189d4cec3cf2bc7ecf579e51077c076634a315b6

0abfb5c7087c97923a26001e416471810ba91662 Actions

General Conan Package Info Properties

Settings

OS:	Linux
Architecture:	x86_64
Build Type:	Debug
Compiler:	gcc
Compiler Version:	7
compiler.libcxx:	libstdc++

Requires

gtest/1.8.1@bincrafters/stable:bd125d7a12e434c681065417c3f1249858085f43

Чему мы научились сегодня

- Как настроить CI с помощью GitHub, TravisCI и Appveyor
- Как управлять зависимостями с помощью conan
- Как настроить CD с помощью conan
- Как развернуть собственный GitLab
- Как настроить Gitlab CI
- Как настроить собственный Artifactory для conan

Источники

1. Grady Booch. Object Oriented Design: with applications — Book
2. Martin Fowler. Continuous Integration (original version) — Blogpost
3. Martin Fowler. Continuous Integration — Blogpost
4. Continuous Integration Vs Continuous Delivery Vs Continuous Deployment
5. Kent Beck. Extreme Programming Explained
6. docs.conan.io
7. Nick Sarten — Travis CI and Modern C++
8. Travis CI - The OS X Build Environment
9. Вы еще не авторизуетесь по ключам? Тогда мы идем к вам
10. Get Docker for Ubuntu