



# Dev контейнеры

Концепция и практическое применение



18 сентября 2023

# Содержание

- Ретроспектива методов разработки.
- Наш проект и его трудности.
- Решение, которое облегчило нам жизнь.

# С чего все начиналось

## Кто мы?

Оля и Дима, разработчики на C++

## Наша задача?

Разработка библиотеки под ARM на x86

## Цель доклада?

Рассказать про технологию, упрощающую разработку

# О проекте

- VS Code / Clion
- C / C++
- Cmake
- x86 / arm7

# Трудности разработки

## Факт:

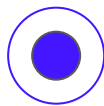
Нам нужно разрабатывать под целевую платформу.

## Проблема:

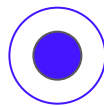
Как и где нам это сделать?

# Ретроспектива

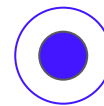
Локальная  
установка



Виртуальные  
машины



Docker  
контейнеры



# Локальная установка

- Рутинная работа.
- Возможность возникновения конфликтов.
- Человеческий фактор.
- Тяжело воспроизвести.
- Быстрое заполнение полезного пространства.



# Локальная установка. Наш опыт

Cygwin



Устанавливаем большое количество библиотек и компиляторов



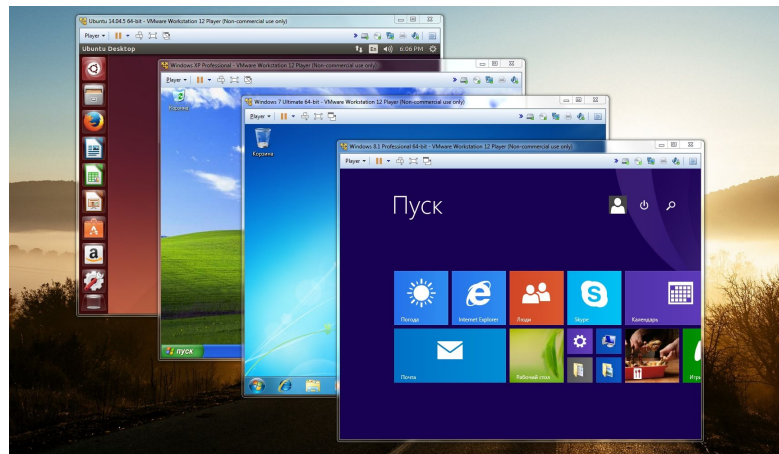
Используем определенный набор компиляторов и кросс-компилятор под cygwin



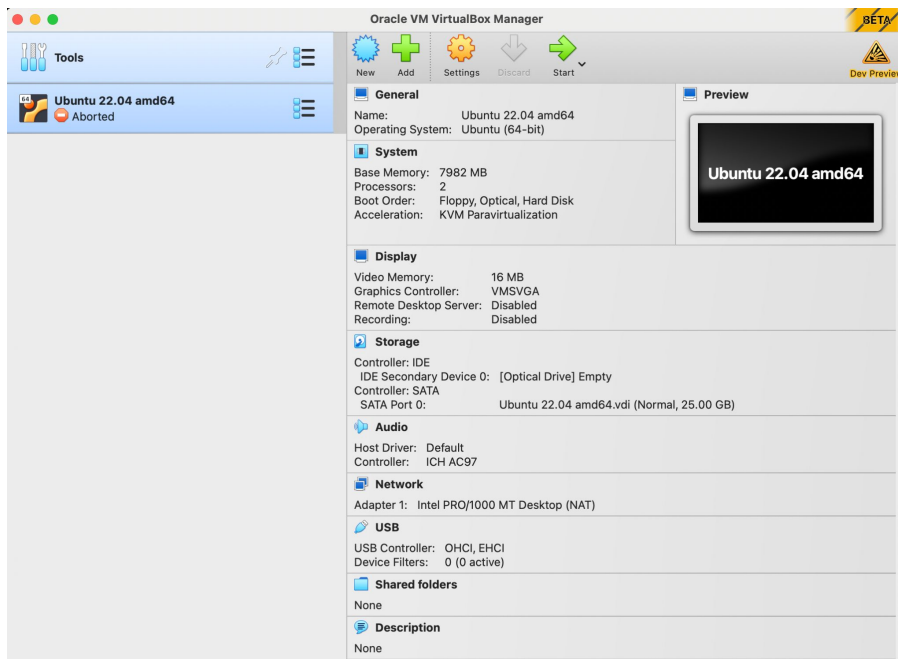
## Ретроспектива

# Виртуальные машины

- Изоляция среды.
- Можно делиться средой разработки.
- Больше ресурсопотребление.



# Виртуальные машины. Наш опыт



# Docker контейнеры

- Легко делиться.
- Легко настраивать.
- Меньше ресурсопотребление, чем у виртуальных машин.

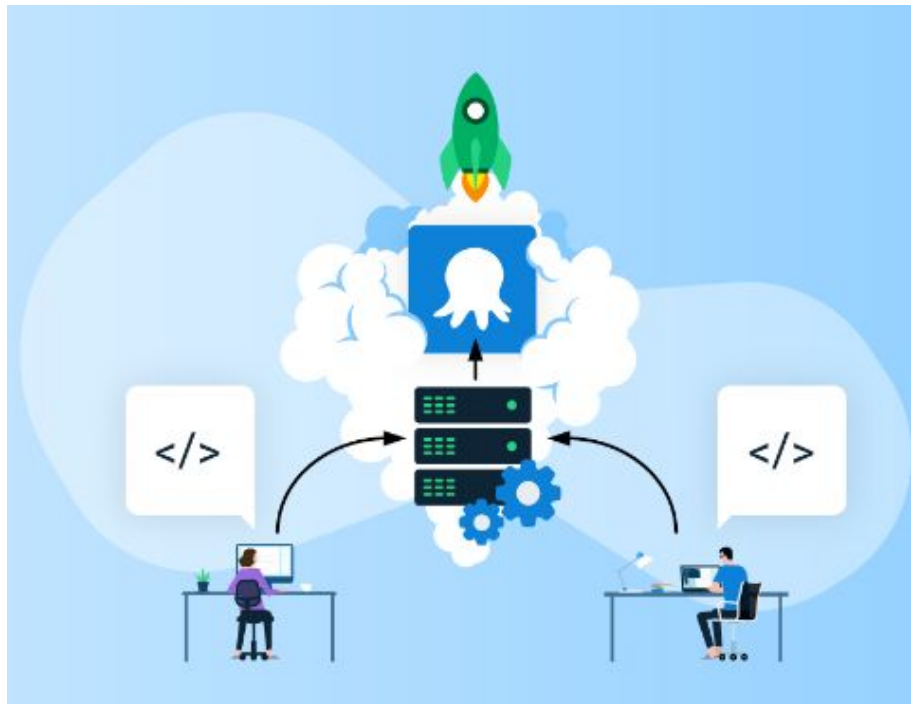


# Build Server

- Главная задача - билд проекта.
- Настроенная среда для билда проекта.
- Подключение через SSH.

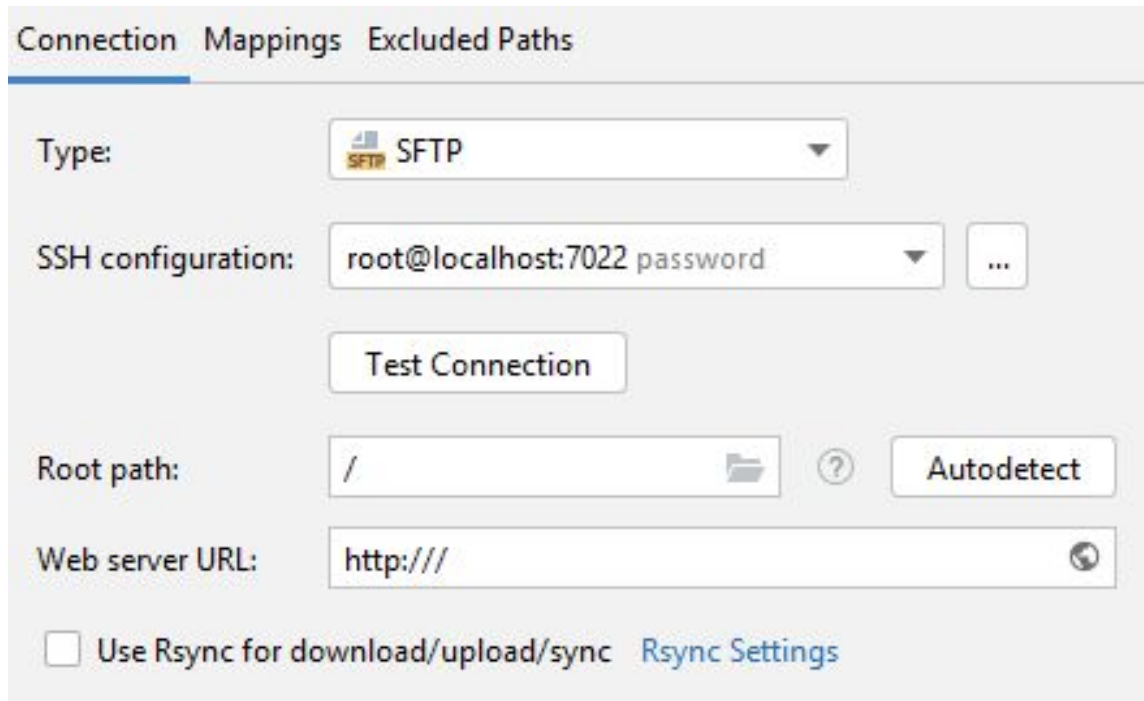
Желательно иметь:

- Большую мощность.
- Хорошее интернет соединение.



# “Локальный Build Server” в Clion

- Docker контейнер, поднятый на локальной системе.
- В контейнере настроен SSH сервер.



The screenshot shows the 'Connection' tab in Clion's settings. The 'Type' is set to 'SFTP'. The 'SSH configuration' is 'root@localhost:7022 password'. There is a 'Test Connection' button. The 'Root path' is '/', with an 'Autodetect' button. The 'Web server URL' is 'http://'. At the bottom, there is a checkbox for 'Use Rsync for download/upload/sync' and a link to 'Rsync Settings'.

Connection Mappings Excluded Paths

Type: SFTP

SSH configuration: root@localhost:7022 password

Test Connection

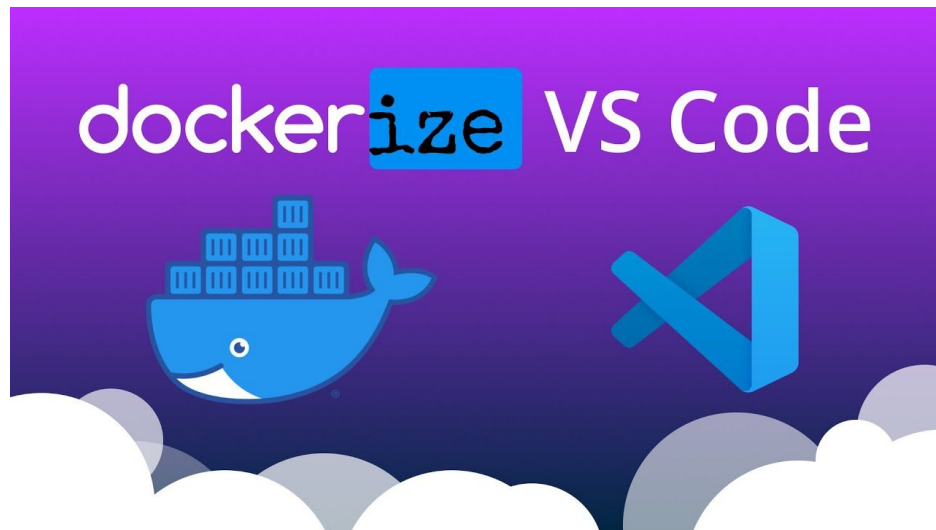
Root path: / Autodetect

Web server URL: http://

Use Rsync for download/upload/sync [Rsync Settings](#)

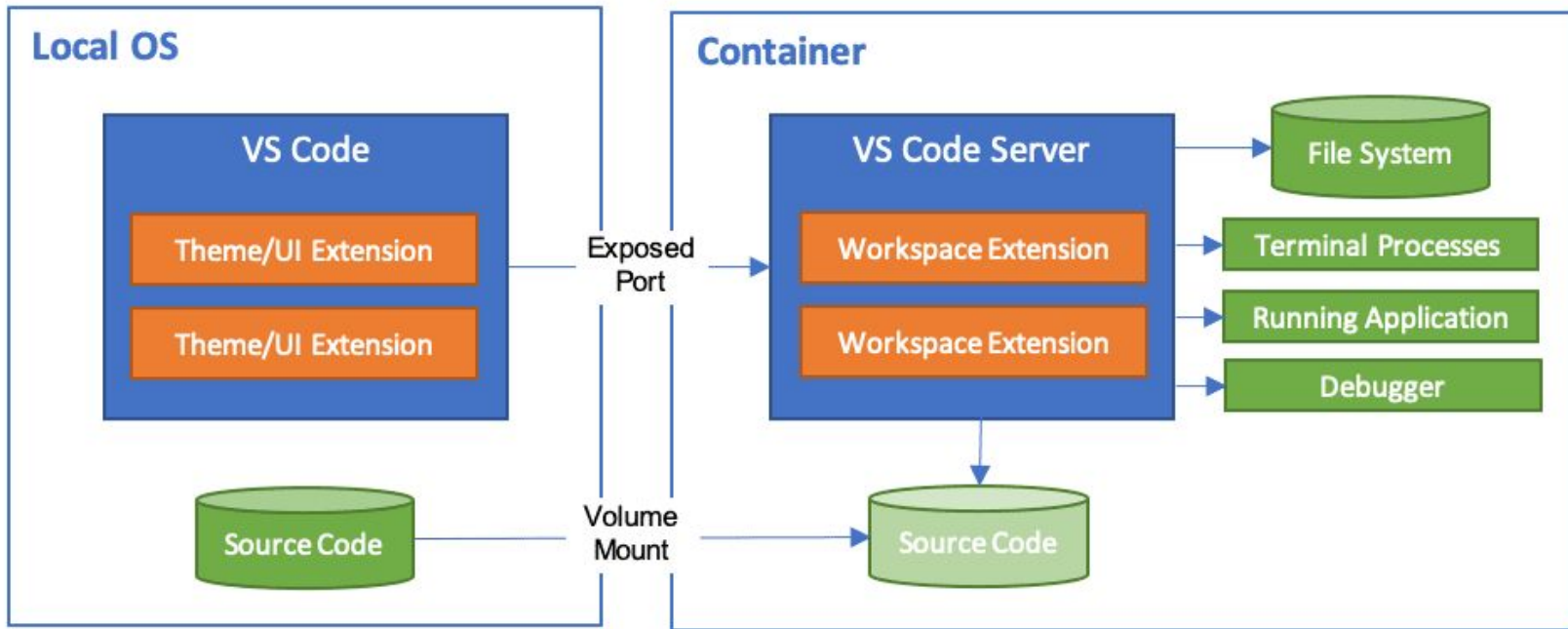
# Dev контейнеры

Visual Studio Code Dev Containers — это расширение, которое позволяет использовать контейнеры в качестве полноценной среды разработки.



**Dev контейнер = Docker контейнер + фичи IDE**

# Архитектура





Dev контейнеры

# Концепция



# Dev контейнеры в разных IDE

## VS Code

Dev Containers

## Clion

Docker toolchain

## Visual Studio

Dev Containers

## Docker desktop

Dev Environments

## GitHub

Codespace

# Development Containers

Открытая спецификация для настройки контейнеров.

<https://containers.dev/>

.devcontainer > {} devcontainer.json > ...

devcontainer.json


```
1  {
2    "name": "Existing Dockerfile",
3
4    "context": "..",
5
6    "dockerFile": "../Dockerfile",
7    "customizations": {
8      "vscode": {
9        "extensions": [
10       "ms-vscode.cpptools",
11       "ms-vscode.cmake-tools",
12       "twxs.cmake"
13     ]
14   }
15 }
16 }
```

# Настройка dev контейнера VS Code



## Dev Containers

Preview

Microsoft  |  16,940,031 installs |  (43) | Free

Open any folder or repository inside a Docker container and take advantage of Visual Studio Code's full feature set.

[Install](#)

[Trouble Installing?](#) 

EXPLORER

OPEN EDITORS

- main.cpp
- Dockerfile
- CMakeLists.txt

DEVCONTAINERAPPLICATION-TEMP

- .devcontainer
  - devcontainer.json
- .idea
- .vscode
- build
- cmake-build-debug
- .gitignore
- CMakeLists.txt
- Dockerfile
- main.cpp
- README.md

```
main.cpp Dockerfile X CMakeLists.txt
Dockerfile
1 FROM amd64/ubuntu:20.04
2
3 # Use non-interactive mode
4 ARG DEBIAN_FRONTEND=noninteractive
5
6 # Set timezone
7 ENV TZ=GMT0
8 RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime \
9     && echo $TZ > /etc/timezone
10
11 # Install all the toolchain dependencies for container
12 RUN apt-get update && apt-get install -y --no-install-recommends \
13     wget build-essential gdb file cmake \
14     && apt-get autoremove \
15     && rm -rf /var/lib/apt/lists/*
16
17 # Install the toolchain for AARCH64 Linux
18 RUN apt-get update && apt-get install -y --no-install-recommends \
```

TERMINAL JUPYTER PROBLEMS OUTPUT ... Tasks

Error: there is no registered task type 'cppbuild'. Did you miss installing an extension that provides a corresponding task provider?



EXPLORER

> OPEN EDITORS

DEVCONTAINERAPPLICATION-TEMP [DEV CONT...]

- > .devcontainer
- > .idea
- > .vscode
- > build
- > cmake-build-debug
- ◆ .gitignore
- M CMakeLists.txt
- 📄 Dockerfile
- 📄 main.cpp
- 📄 README.md

> OUTLINE

> TIMELINE

```
main.cpp CMakeLists.txt Dockerfile X
```

```
Dockerfile
6 # Set timezone
7 ENV TZ=GMT0
8 RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime \
9     && echo $TZ > /etc/timezone
10
11 # Install all the toolchain dependencies for container
12 RUN apt-get update && apt-get install -y --no-install-recommends \
13     wget build-essential gdb file cmake \
14     && apt-get autoremove \
15     && rm -rf /var/lib/apt/lists/*
16
17 # Install the toolchain for AARCH64 Linux
18 RUN apt-get update && apt-get -y install g++-aarch64-linux-gnu
19
```

TERMINAL PORTS PROBLEMS OUTPUT ...

CMake/Build

```
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /workspaces/
DevContainerApplication-temp/build
```

24/53



# Приложение

Идея:

- Разработка на x86
- Тестирование на aarch64

# Dockerfile

Добавим в Dockerfile  
тулчейн целевой  
платформы

```
FROM amd64/ubuntu:20.04

# Use non-interactive mode
ARG DEBIAN_FRONTEND=noninteractive

# Set timezone
ENV TZ=GMT0
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime \
    && echo $TZ > /etc/timezone

# Install all the toolchain dependencies for container
RUN apt-get update && apt-get install -y --no-install-recommends \
    wget build-essential gdb file cmake \
    && apt-get autoremove \
    && rm -rf /var/lib/apt/lists/*

# Install the toolchain for AARCH64 Linux
RUN apt-get update && apt-get -y install g++-aarch64-linux-gnu
```

# Cmake

Добавим в CMakeLists.txt  
описание нового тулчейна  
и настроим кросс-  
КОМПИЛЯЦИЮ

```
cmake_minimum_required(VERSION 3.13)
include(CMakePrintHelpers)

project (DevContainerApplication)

set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR aarch64)
set(CMAKE_CROSSCOMPILING true)

set(PLATFORM_TOOLCHAIN_PATH /usr/bin)
set(HOST_PLATFORM aarch64-linux-gnu)

set(CMAKE_AR ${PLATFORM_TOOLCHAIN_PATH}/${HOST_PLATFORM}-ar)
set(CMAKE_CXX_COMPILER ${PLATFORM_TOOLCHAIN_PATH}/${HOST_PLATFORM}-g++)
set(CMAKE_LINKER ${PLATFORM_TOOLCHAIN_PATH}/${HOST_PLATFORM}-ld)

set(CMAKE_CXX_FLAGS "${CMAKE_C_FLAGS} - static " CACHE INTERNAL "")

cmake_print_variables(CMAKE_CXX_COMPILER)

add_executable(DevContainerApplication-temp main.cpp)
```

# Расширения VS Code

Используем расширения, установленные в dev контейнер, чтобы собрать проект



## CMake Tools v1.13.45

Microsoft | 16 973 420 | ★★★★★ (65)

Extended CMake support in Visual Studio Code

Install



## CMake v0.0.17

twxs | 17 010 947 | ★★★★★☆ (24)

CMake language support for Visual Studio Code

Install



## C/C++ v1.14.3

Microsoft | 43 209 791 | ★★★★★☆ (509)

C/C++ IntelliSense, debugging, and code browsing.

Install

```
"customizations": {  
  "vscode": {  
    "extensions": [  
      "ms-vscode.cpptools",  
      "ms-vscode.cmake-tools",  
      "twxs.cmake"  
    ]  
  }  
}
```

```

1 #include <stdio>
2
3 int main()
4 {
5     printf("Hello, Docker!\n");
6
7     return 0;
8 }
9

```

### Containers [Give Feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

Showing 1 Items

<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)	STARTED	
<input type="checkbox"/>	> cflon 1 container	-	Running (1/1)	-		<a href="#">Open</a>

# Как на нас повлияло внедрение dev контейнеров

# Стандарты кодирования

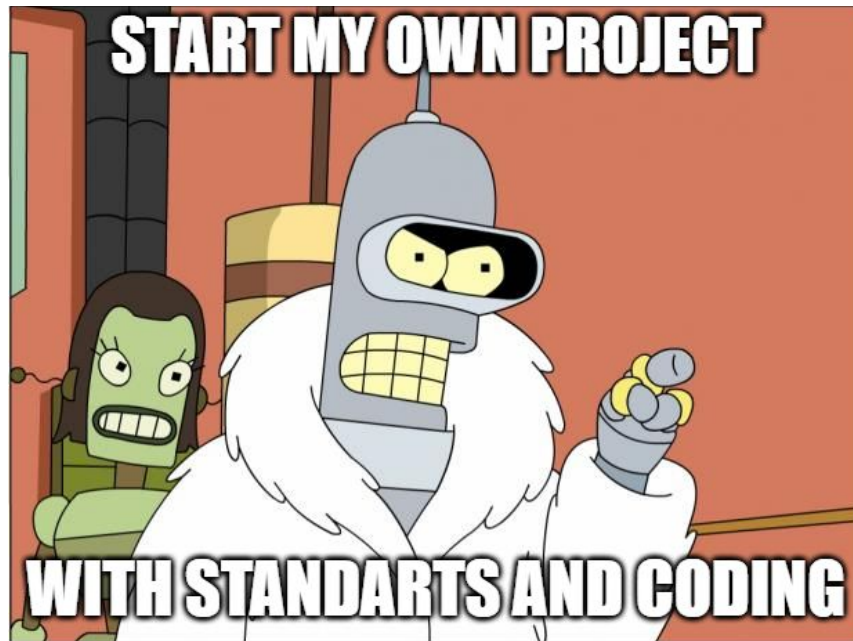
Общий стандарт оформления  
кода



Линтинг одинаково  
настроенный для всех



Читаемый код



# Парное программирование

Одно окружение для всей  
команды



Легко работать вместе



Эффективная разработка



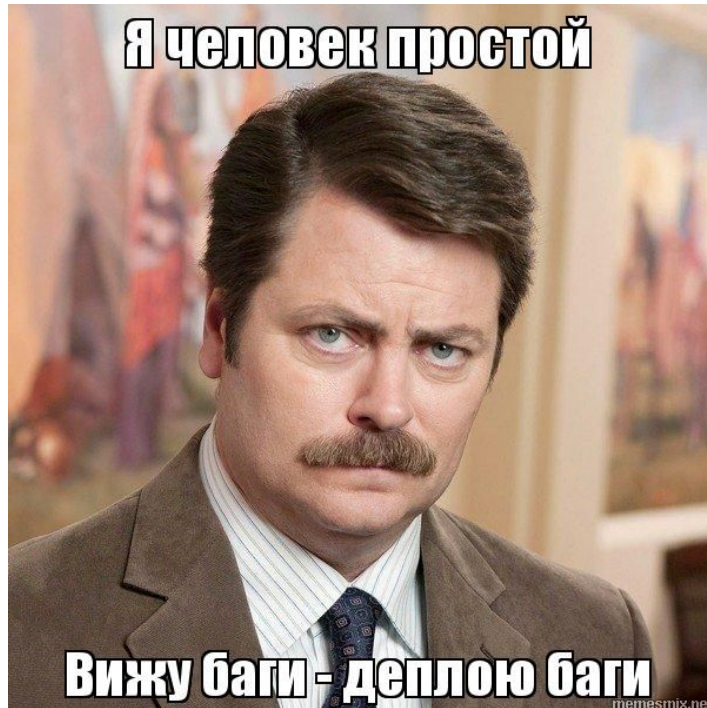


# Локальное интеграционное тестирование

Можно тестировать не только  
по коммиту, но и локально



Быстрый рефакторинг

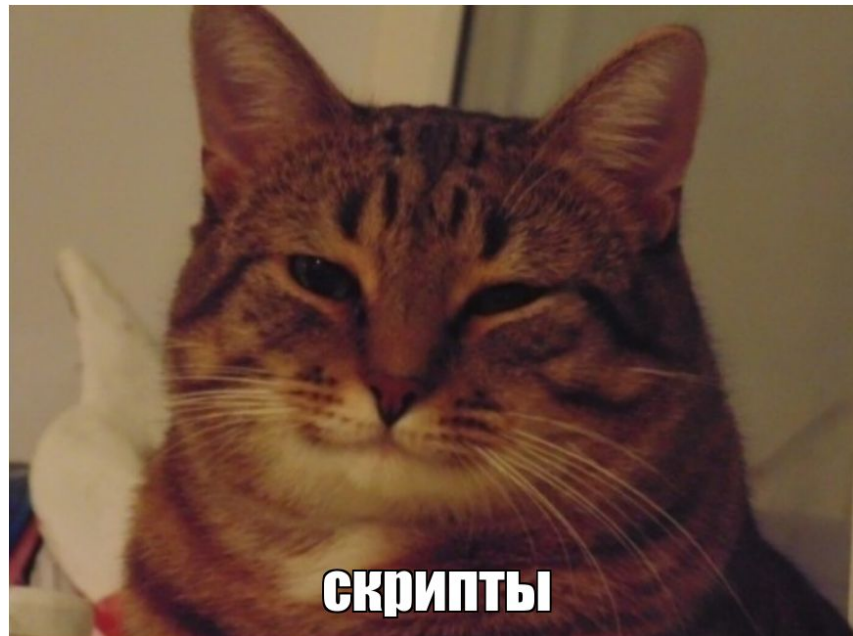


# Запуск скриптов

Все, что нужно в одном  
контейнере



Легкий запуск скриптов  
с зависимостями



# В чем еще польза?

- Удобная мультиплатформенная разработка
- Можно использовать в CI/CD
- Интеграция с Github
- Поддержка разных сред разработки

# CI/CD. А нужно ли?

## Достоинства

- Полная воспроизводимость
- Стандартизация
- Легче поддерживать один контейнер, а не два

## Недостатки

- Скорость
- Размер

# CI/CD. А как?

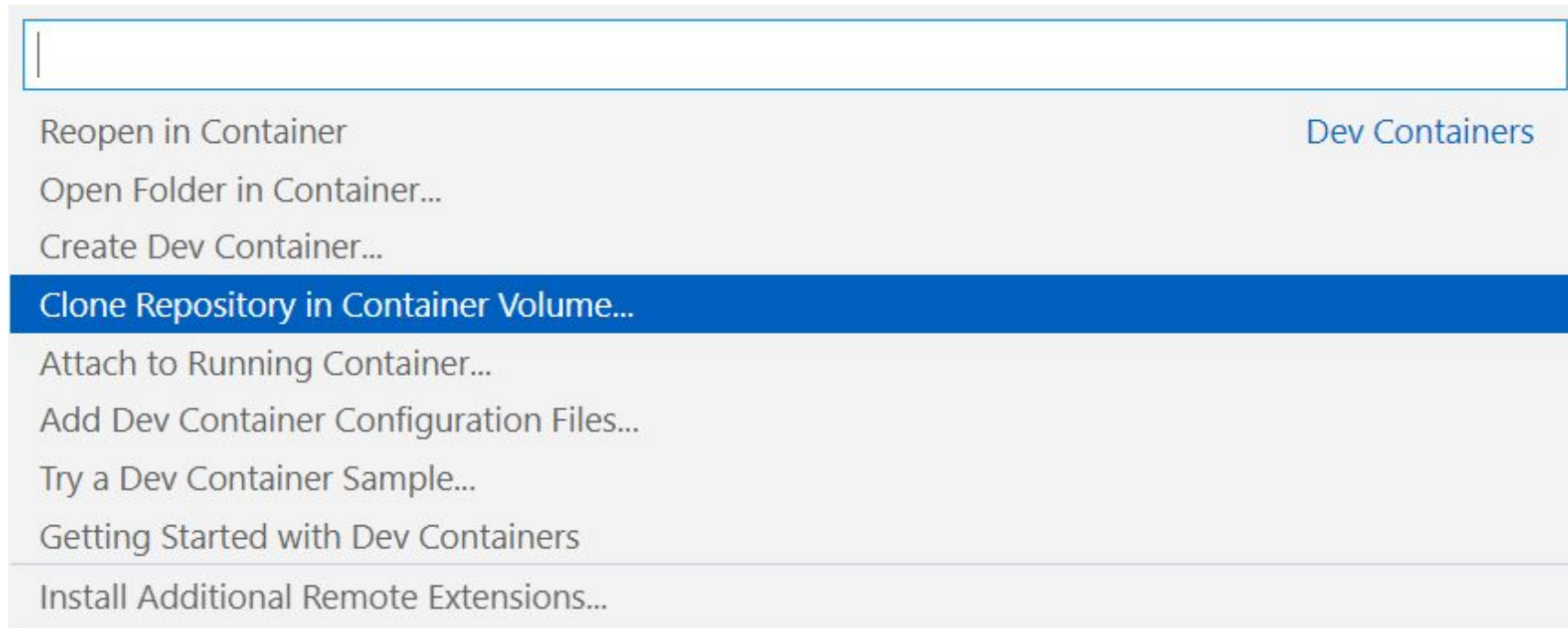
Используем тот же docker образ, что и в dev контейнере

1. Помещаем docker образ dev контейнера в registry
2. Подтягиваем docker образ в workflow.yml

## CI/CD. А что получилось?

- Собираем приложение под различные платформы
- Запускаем тесты для всех нужных платформ
- Можем делать то же самое локально

## Запуск проекта из Github репозитория



Reopen in Container Dev Containers

Open Folder in Container...

Create Dev Container...

**Clone Repository in Container Volume...**

Attach to Running Container...

Add Dev Container Configuration Files...

Try a Dev Container Sample...

Getting Started with Dev Containers

Install Additional Remote Extensions...

# Codespace

The screenshot shows the VS Code interface for a repository. At the top, there is a breadcrumb 'feature/DevCon...' with a dropdown arrow, '2 branches', and '0 tags'. To the right are buttons for 'Go to file', 'Add file', and a green 'Code' button with a dropdown arrow. Below this, a message states 'This branch is 7 commits ahead of master.' The main area shows a file explorer with a commit message 'Olya Kuzmicheva ref(dev container application): delete wrong executable'. The file list includes folders '.devcontainer' and '.vscode', and files '.gitignore', 'CMakeLists.txt', 'Dockerfile', 'README.md', and 'main.cpp'. A dropdown menu is open over the 'Code' button, showing 'Local' and 'Codespaces' tabs. The 'Codespaces' tab is active, displaying 'Your workspaces in the cloud' with a plus and three dots icon. Below this, it lists workspaces: 'On current branch' with 'fictional happiness' (Active) and 'feature/DevContaine...' (No changes); and 'On other branches' with 'opulent space fortnight' (Active) and 'master' (No changes). At the bottom of the menu, it says 'Codespace usage for this repository is paid for by SoftAvocado'.



## Codespace

✓ Image found.

🔧 Building container...

[Hide logs](#)

```
#7 16.14 Preparing to unpack .../21-g++-9-aarch64-linux-gnu_9.4.0-1ubuntu1~20.04.1cross2_amd64.deb ...
#7 16.18 Unpacking g++-9-aarch64-linux-gnu (9.4.0-1ubuntu1~20.04.1cross2) ...
#7 16.99 Selecting previously unselected package gcc-aarch64-linux-gnu.
#7 16.99 Preparing to unpack .../22-gcc-aarch64-linux-gnu_4%3a9.3.0-1ubuntu2_amd64.deb ...
#7 17.02 Unpacking gcc-aarch64-linux-gnu (4:9.3.0-1ubuntu2) ...
#7 17.18 Selecting previously unselected package g++-aarch64-linux-gnu.
#7 17.19 Preparing to unpack .../23-g++-aarch64-linux-gnu_4%3a9.3.0-1ubuntu2_amd64.deb ...
#7 17.22 Unpacking g++-aarch64-linux-gnu (4:9.3.0-1ubuntu2) ...
#7 17.41 Setting up gcc-9-aarch64-linux-gnu-base:amd64 (9.4.0-1ubuntu1~20.04.1cross2) ...
```

## Возможности dev контейнеров в VS Code

# Codespace

The screenshot displays the VS Code Codespace environment. The browser address bar shows the URL: `softavocado-fictional-happiness-r6rvq46grr42pvpg.github.dev`. The file explorer on the left shows the directory structure:

- PROВОННИК
- DEVCONTAINERAPPLICATION-TEMP [CODESPACES]
  - .devcontainer
  - .vscode
  - build
  - .gitignore
  - CMakeLists.txt
  - Dockerfile
  - main.cpp
  - README.md
- СТРУКТУРА
- ВРЕМЕННАЯ ШКАЛА

The main editor shows the content of `main.cpp`:

```
1 #include <stdio>
2
3 int main()
4 {
5     printf("Hello, Docker!\n");
6
7     return 0;
8 }
9
```

The terminal window at the bottom shows the output of the CMake build process:

```
ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ CMake/Build
[cmake] -- Detecting C compiler ABI info
[cmake] -- Detecting C compiler ABI info - done
[cmake] -- Detecting C compile features
[cmake] -- Detecting C compile features - done
[cmake] -- Check for working CXX compiler: /usr/bin/g++
[cmake] -- Check for working CXX compiler: /usr/bin/g++ -- works
[cmake] -- Detecting CXX compiler ABI info
[cmake] -- Detecting CXX compiler ABI info - done
[cmake] -- Detecting CXX compile features
[cmake] -- Detecting CXX compile features - done
[cmake] -- CMAKE_CXX_COMPILER="/usr/bin/aarch64-linux-gnu-g++"
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /workspaces/DevContainerApplication-temp/build
[cmakefileapi-driver] This version of CMake does not support the "toolchains" object kind. Compiler paths will be determined by reading CMakeCache.txt.
```

The status bar at the bottom indicates the environment: Codespaces, 0 errors, 0 warnings, CMake: [Debug] Ready, GCC 9.4.0 x86\_64-linux-gnu, Build [all], Пробелов: 4, UTF-8, LF, C++, Make: US, Linux.

# Dev контейнер в Visual Studio\*

\* версия 2022 17.5



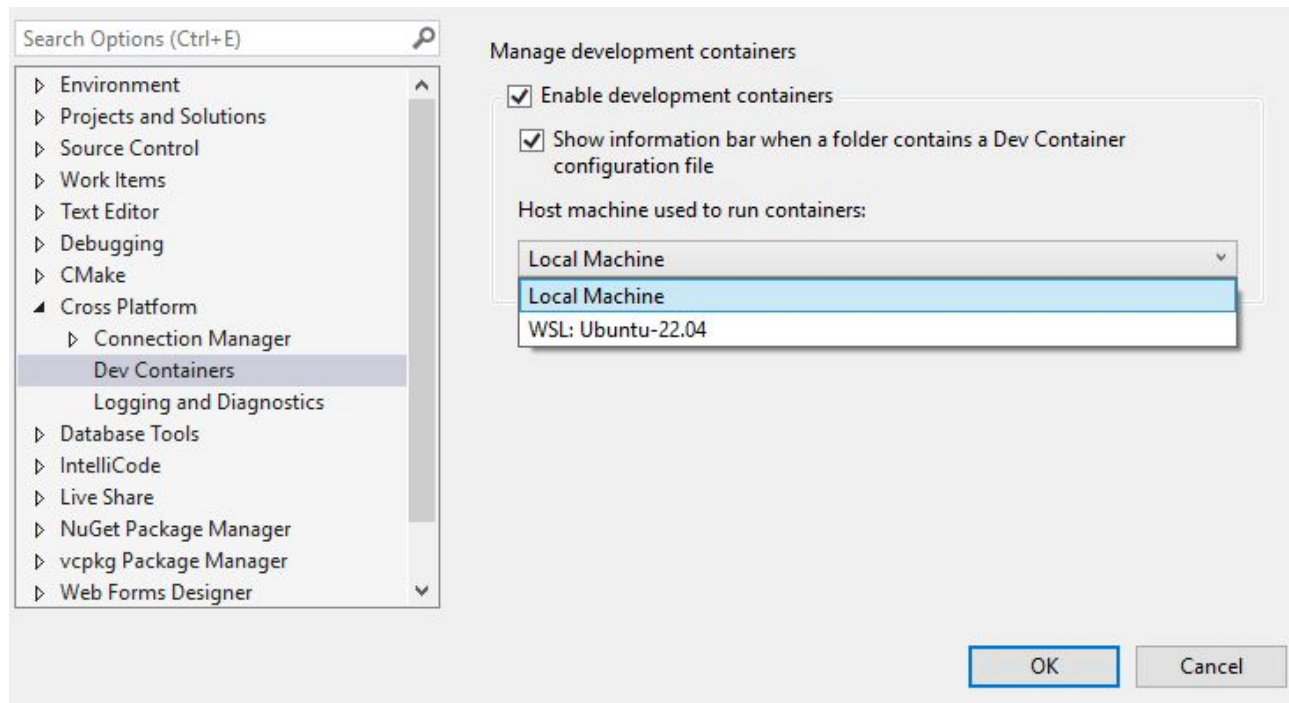
Linux and embedded development with C++  
Create and debug applications running in a Linux environment or on an embedded device.



<https://devblogs.microsoft.com/cppblog/dev-containers-for-c-in-visual-studio>

# Dev контейнер в Visual Studio\*

\* версия 2022 17.5



<https://devblogs.microsoft.com/cppblog/dev-containers-for-c-in-visual-studio>

# Dev контейнер в Visual Studio\*

\* версия 2022 17.5

```
Folder contains a Dev Container configuration file. Reopen folder in container Settings Learn more

Output
Show output from: Dev Containers
[10 ms] @microsoft/vscode-dev-containers-cli 0.60.1.
[393 ms] Start: Run: docker build -f c:\source\visualstudio-devcontainer-cpp\.devcontainer\Dc

[+] Building 0.0s (0/1)

[+] Building 0.2s (2/3)
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 649B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for mcr.microsoft.com/vscode/devcontainers/c 0.1s

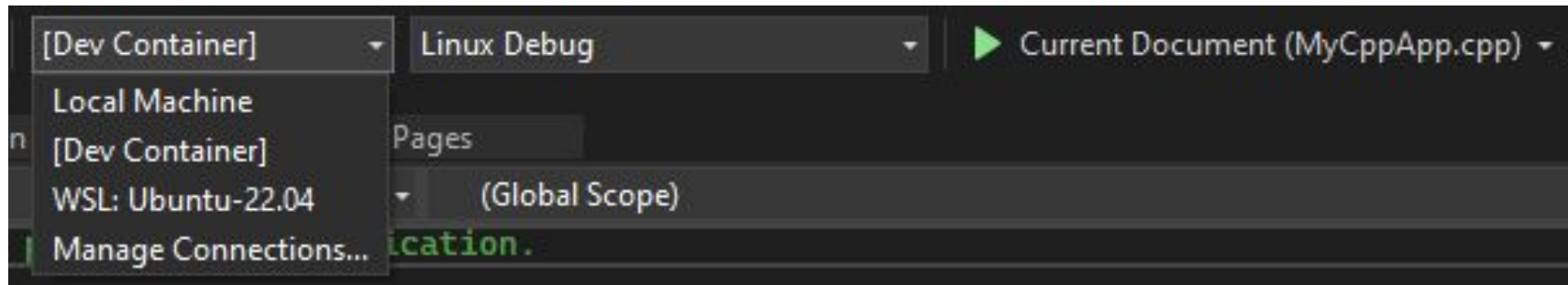
[+] Building 0.3s (2/3)
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 649B 0.0s
```



<https://devblogs.microsoft.com/cppblog/dev-containers-for-c-in-visual-studio>

# Dev контейнер в Visual Studio\*

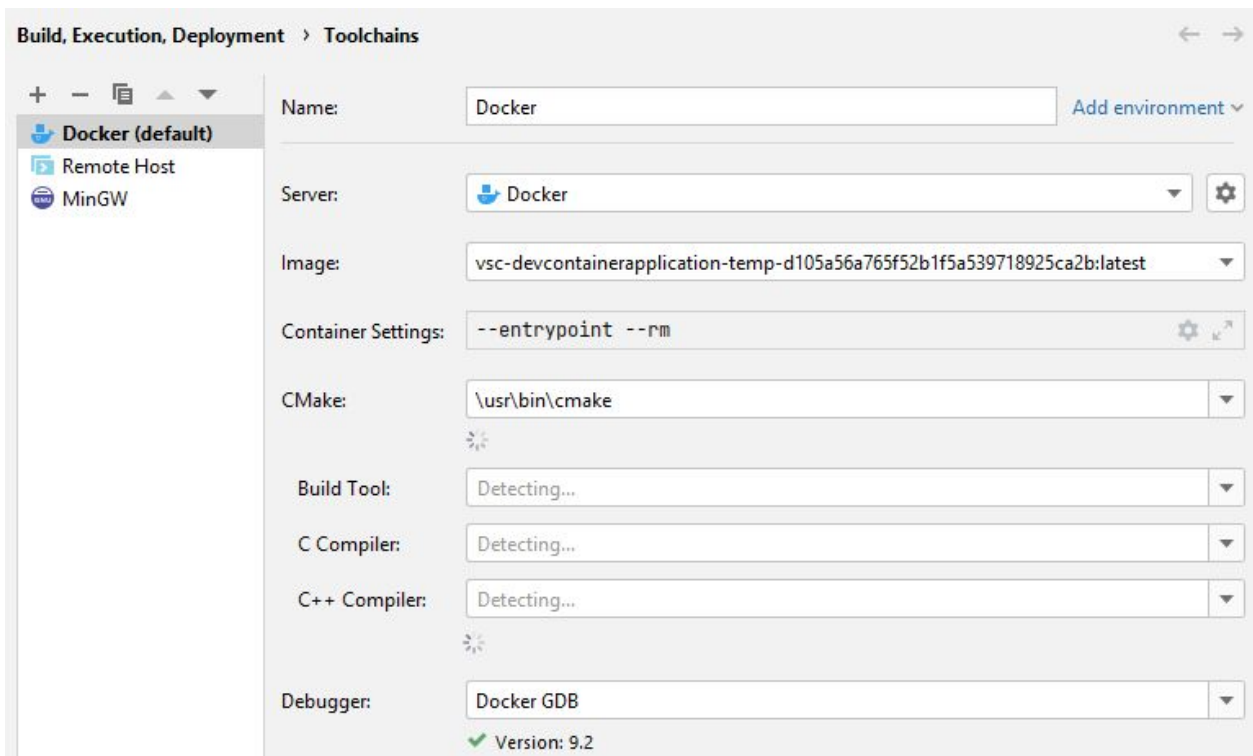
\* версия 2022 17.5



<https://devblogs.microsoft.com/cppblog/dev-containers-for-c-in-visual-studio>

# Dev контейнер в Clion\*

\* версия 2022.2.3



The screenshot shows the 'Toolchains' configuration window in Clion IDE. The window title is 'Build, Execution, Deployment > Toolchains'. On the left, there is a sidebar with three toolchain options: 'Docker (default)', 'Remote Host', and 'MinGW'. The 'Docker (default)' toolchain is selected. The main configuration area contains the following fields:

- Name:** Docker
- Server:** Docker
- Image:** vsc-devcontainerapplication-temp-d105a56a765f52b1f5a539718925ca2b:latest
- Container Settings:** --entrypoint --rm
- CMake:** \usr\bin\cmake
- Build Tool:** Detecting...
- C Compiler:** Detecting...
- C++ Compiler:** Detecting...
- Debugger:** Docker GDB

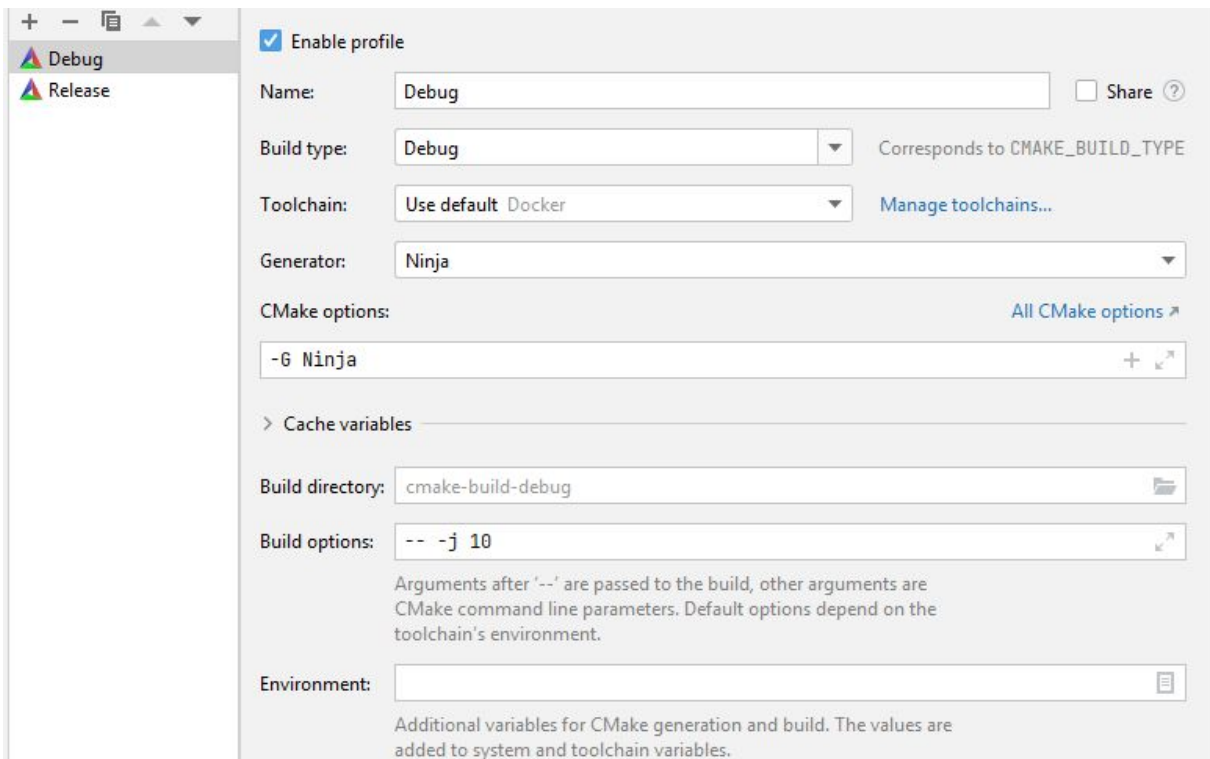
At the bottom, there is a green checkmark and the text 'Version: 9.2'.



<https://www.jetbrains.com/help/clion/clion-toolchains-in-docker.html>

# Dev контейнер в Clion\*

\* версия 2022.2.3



Enable profile

Name:   Share ?

Build type:  Corresponds to CMAKE\_BUILD\_TYPE

Toolchain:  [Manage toolchains...](#)

Generator:

CMake options: [All CMake options ↗](#)

> Cache variables

Build directory:

Build options:

Arguments after '--' are passed to the build, other arguments are CMake command line parameters. Default options depend on the toolchain's environment.

Environment:

Additional variables for CMake generation and build. The values are added to system and toolchain variables.



<https://www.jetbrains.com/help/clion/clion-toolchains-in-docker.html>



# Dev контейнер vs Локальный Build Server

## Dev контейнер:

- + Автоматизация
- + Мобильность
- + Синхронизация расширений VS Code
- + Развитие

## Локальный Build Server:

- Ручное управление
- Сложность настройки

# Сравнение производительности

## Dev контейнер:

Windows: 4 min

Linux: 20 sec

MacOS: 1 min 5 sec

## Локальный Build Server:

Windows: 40 sec

Linux: 19 sec

MacOS: 1 min 3 sec

CPU = 0.8 Memory = 1Gb

\*Windows - AMD Ryzen 5 2400G with Radeon Vega Graphics 4-core, 32Gb RAM

\*Linux - AMD Ryzen 7 3700× 8-core, 16 Gb RAM

\*MacOS - Intel Core i7 4-core, 2.6ghz, 16 Gb RAM

# Итоги по Dev контейнерам

- Изолированная разработка, сборка и дебаг.
- Использование жестко закрепленных компиляторов и других утилит для сборки.
- Синхронизация настроек среды разработки между командой (в том числе на разных платформах).
- Разработка и тестирование на отличных от используемой платформах.

**Будущее за вами!**

Разработчики C++

# Кузьмичева Ольга & Стародубцев Дмитрий

- [okuzmicheva@tourmalinecore.com](mailto:okuzmicheva@tourmalinecore.com)
- [dstarodubtcev@tourmalinecore.com](mailto:dstarodubtcev@tourmalinecore.com)
- <https://github.com/TourmalineCore/DevContainerApplication>

