tourmaline
core

Dev контейнеры

Концепция и практическое применение

12 мая 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

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

Локальная установка. Наш опыт

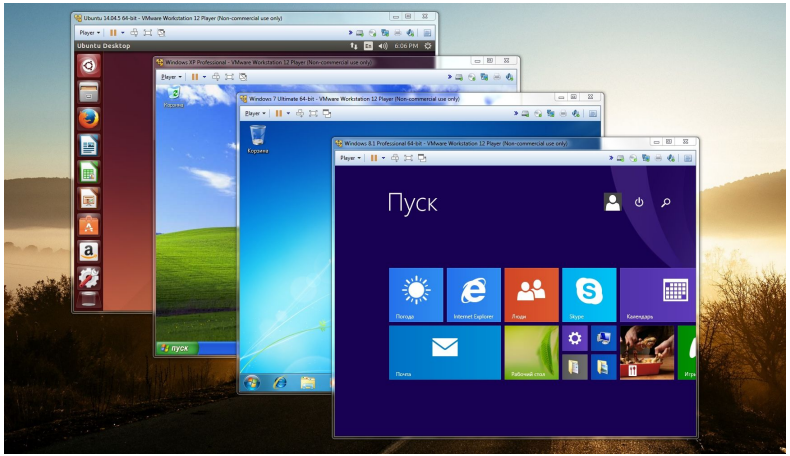
Устанавливаем большое
количество библиотек
и компиляторов

Используем определенный набор
компиляторов и кросс-компилятор
под cygwin

Ретроспектива

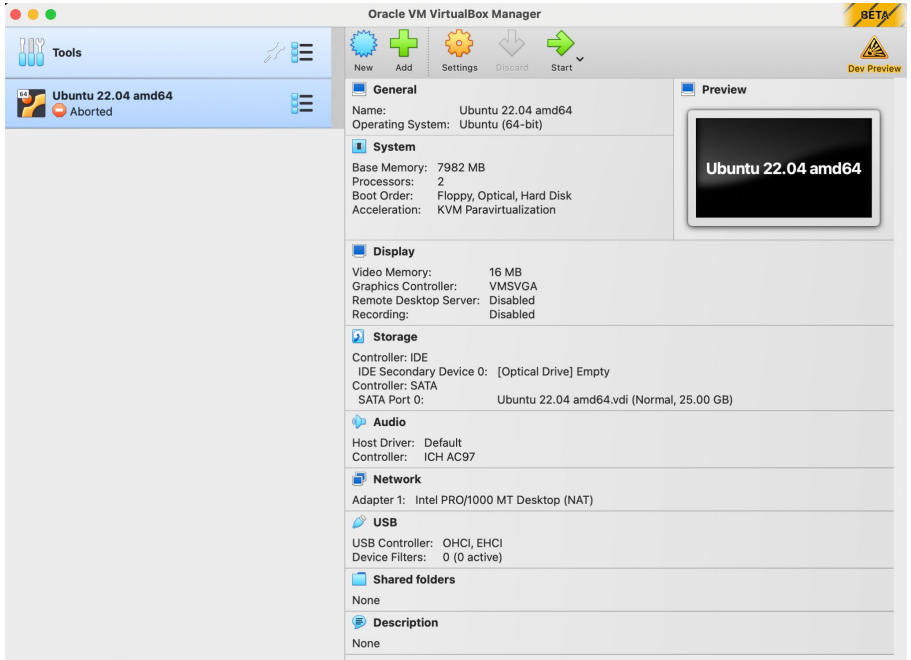
Виртуальные машины

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



Ретроспектива

Виртуальные машины. Наш опыт



Docker контейнеры

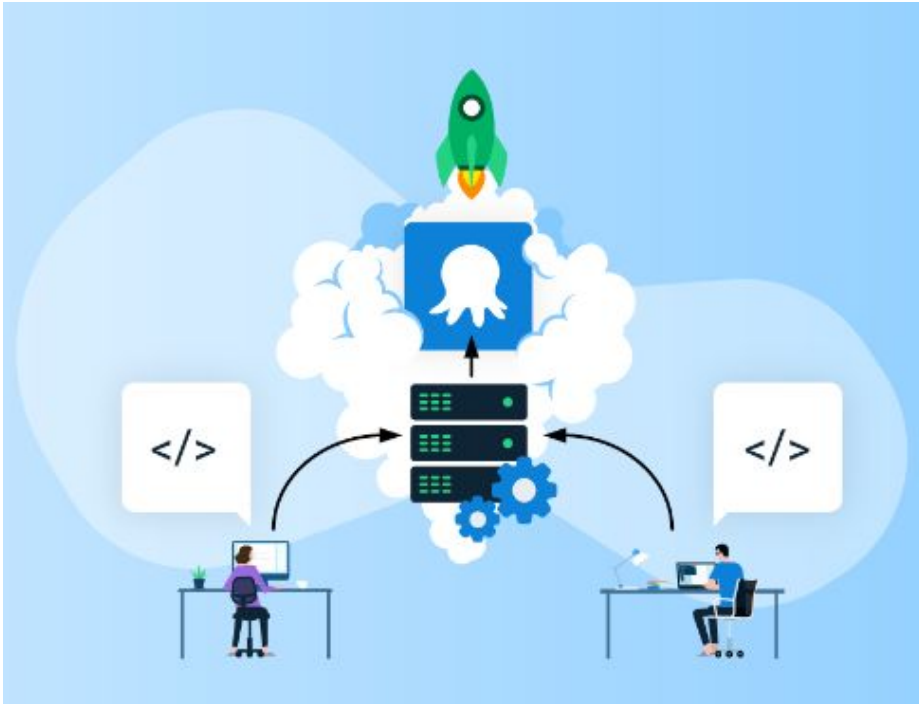
- Легко делиться.
- Легко настраивать.
- Меньше ресурсопотребление, чем у виртуальных машин.

Build Server

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

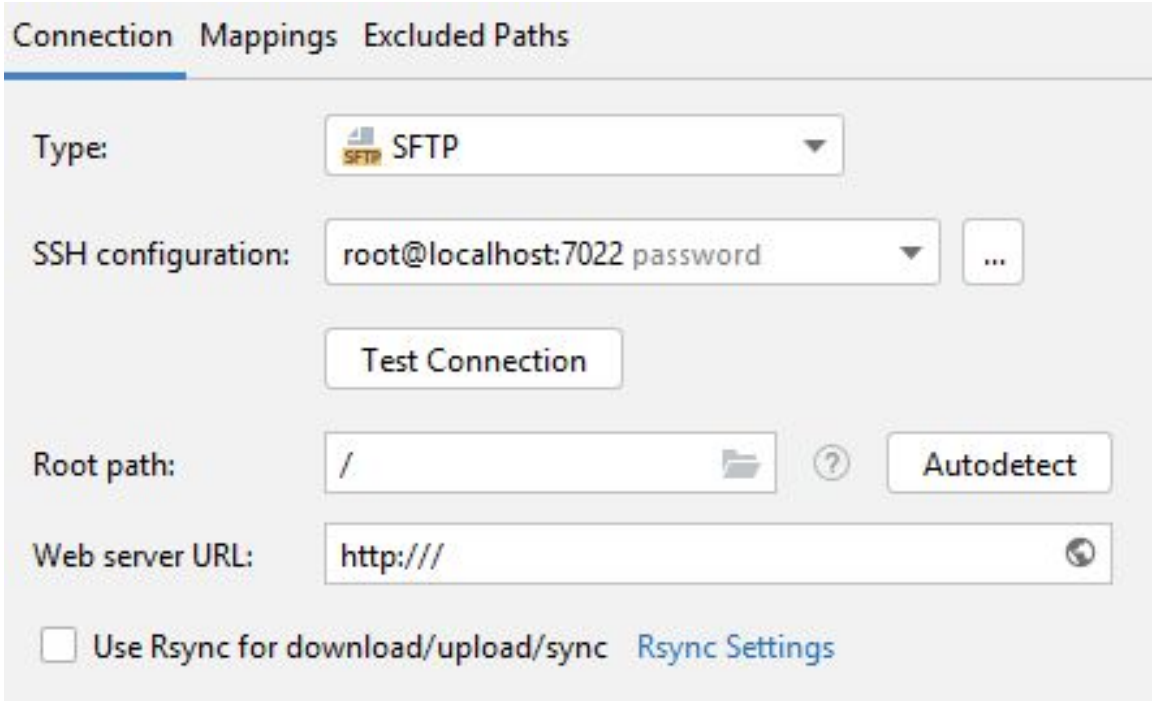
Желательно иметь:

- Большую мощность.
- Хорошее интернет соединение.



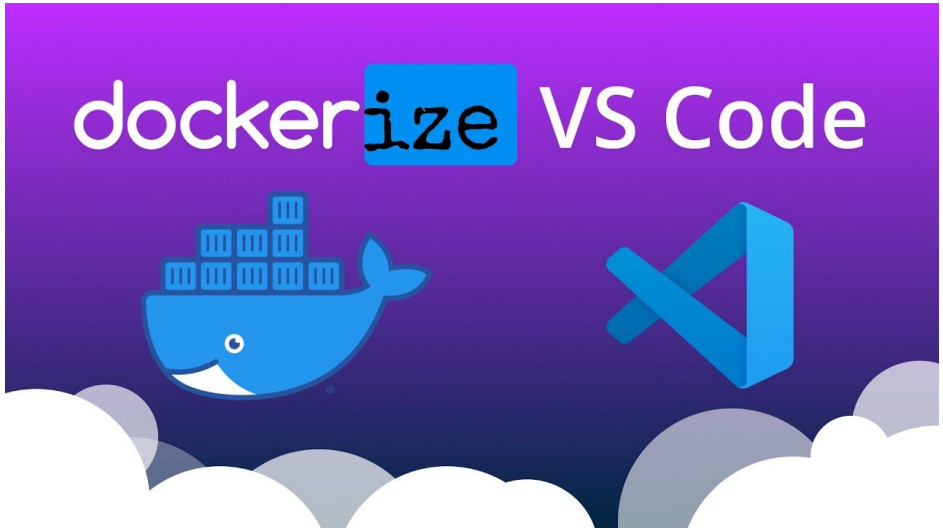
“Локальный Build Server” в Clion

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



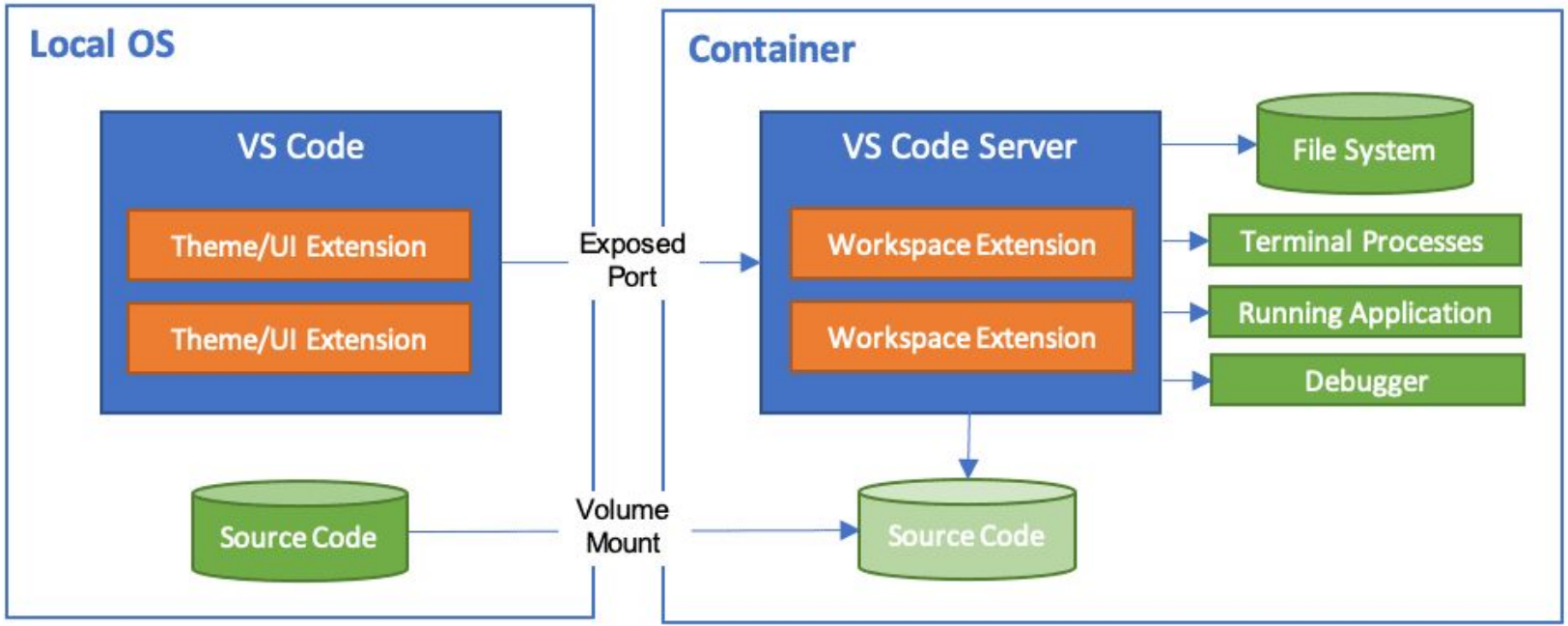
Dev контейнеры

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



Dev контейнер = Docker контейнер + фичи IDE

Архитектура



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.

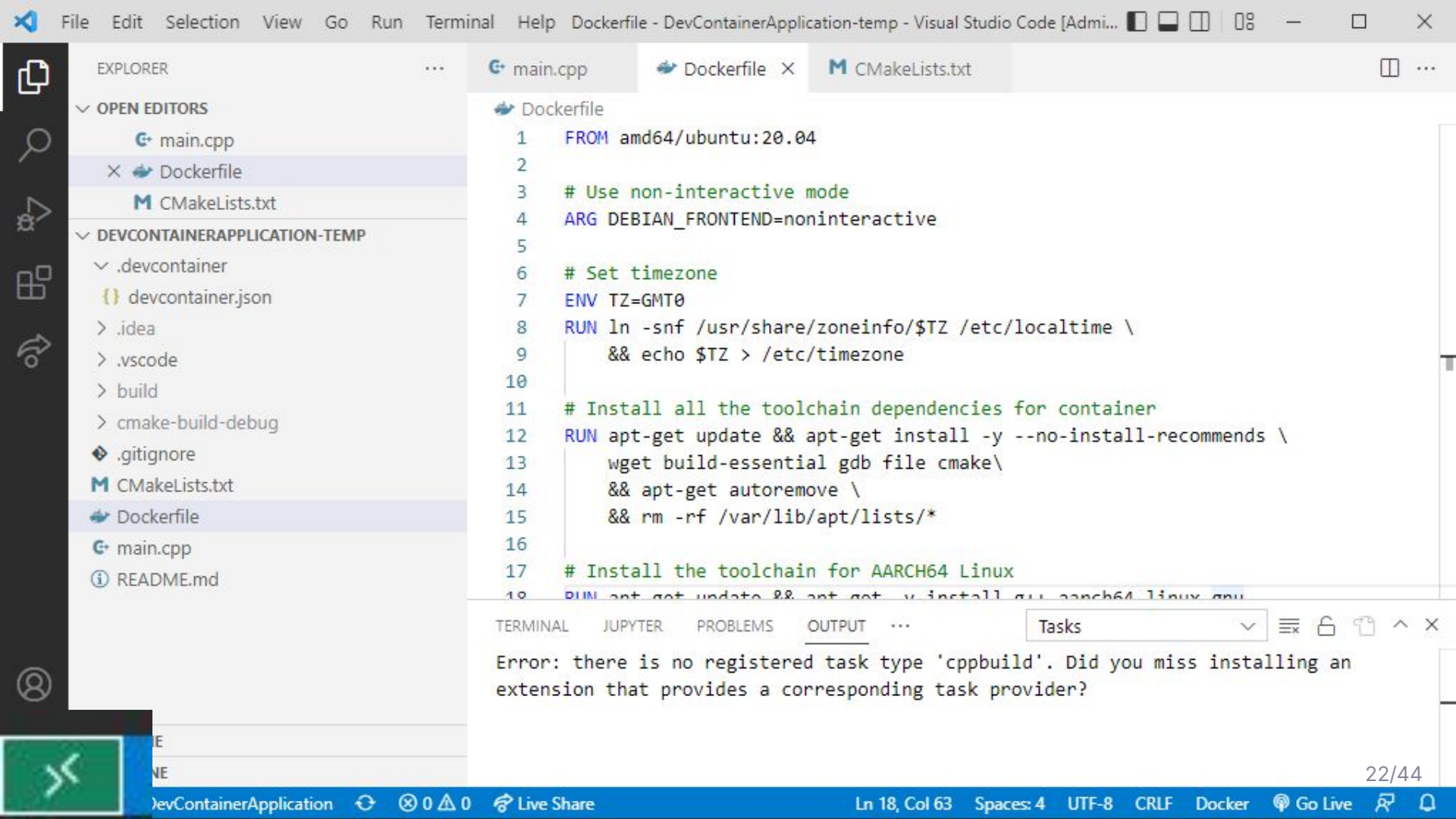
https://containers.dev/

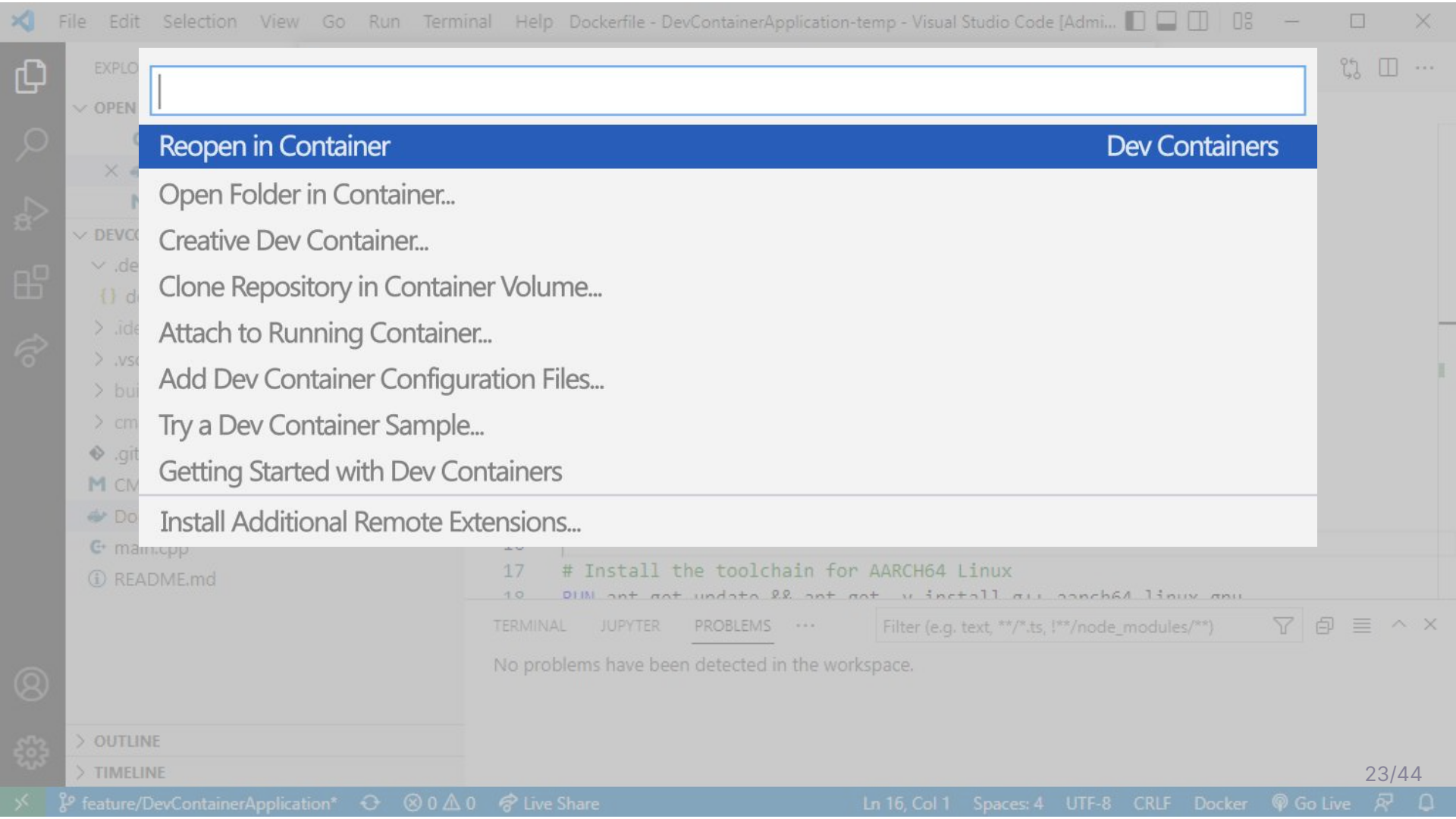
devcontainer.json

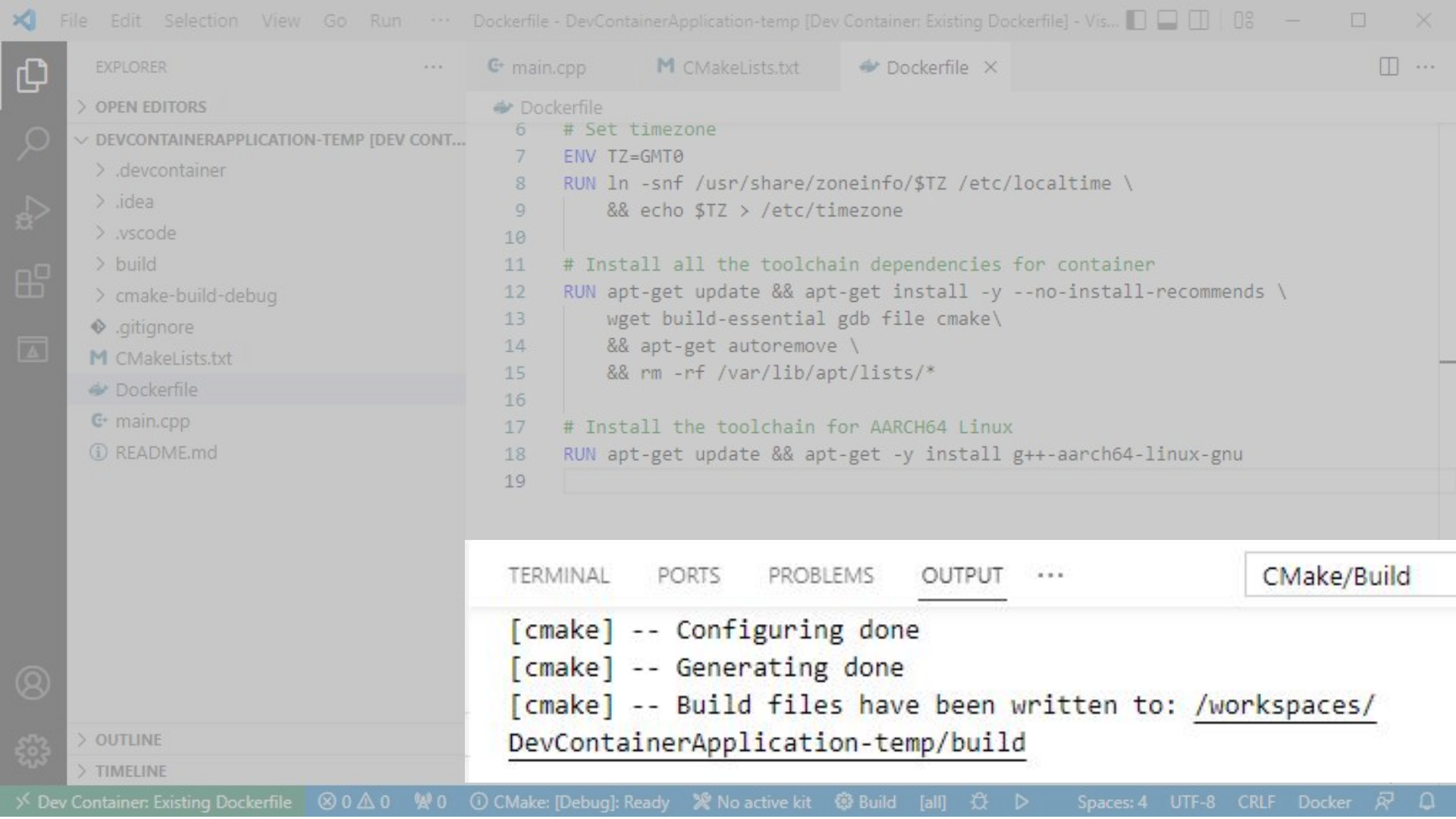
.devcontainer > {} devcontainer.json > ...

```jsonc
{
    "name": "Existing Dockerfile",

    "context": "..",

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

Настройка dev контейнера VS Code


File Edit Selection View Go Run Terminal Help Dockerfile - DevContainerApplication-temp - Visual Studio Code [Admi...
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
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
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?
DevContainerApplication
0 0
Live Share
Ln 18, Col 63 Spaces: 4 UTF-8 CRLF Docker Go Live


Reopen in Container
Dev Containers
Open Folder in Container...
Creative 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...
Install the toolchain for AARCH64 Linux
TERMINAL
JUPYTER
PROBLEMS
Filter (e.g. text, **/*.ts, !**/node_modules/**)
No problems have been detected in the workspace.
OUTLINE
TIMELINE
feature/DevContainerApplication*
Live Share
Ln 16, Col 1
Spaces: 4
UTF-8
CRLF
Docker
Go Live


File Edit Selection View Go Run ··· Dockerfile - DevContainerApplication-temp [Dev Container: Existing Dockerfile] - Vis...
EXPLORER
OPEN EDITORS
DEVCONTAINERAPPLICATION-TEMP [DEV CONT...
.devcontainer
.idea
.vscode
build
cmake-build-debug
.gitignore
CMakeLists.txt
Dockerfile
main.cpp
README.md
OUTLINE
TIMELINE
main.cpp
CMakeLists.txt
Dockerfile
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
Dev Container: Existing Dockerfile
0 0 0
CMake: [Debug]: Ready
No active kit
Build
[all]
Spaces: 4
UTF-8
CRLF
Docker

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
```

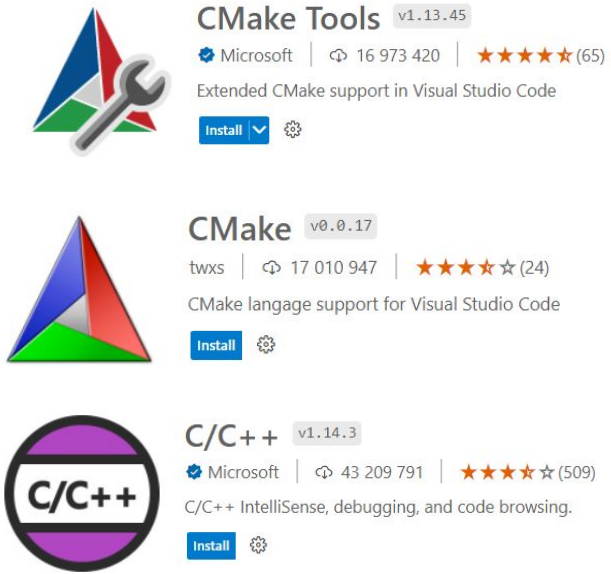
Cmake

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

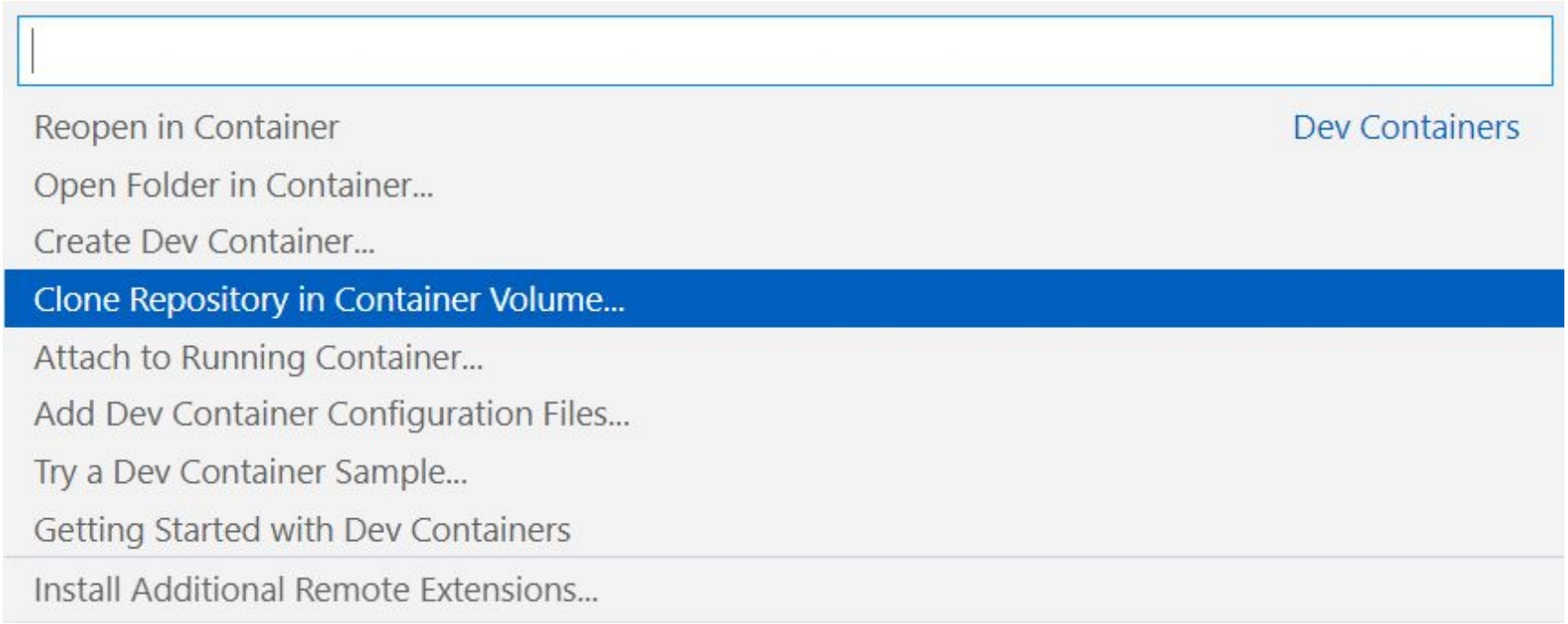
Используем расширения, установленные в dev контейнер, чтобы собрать проект



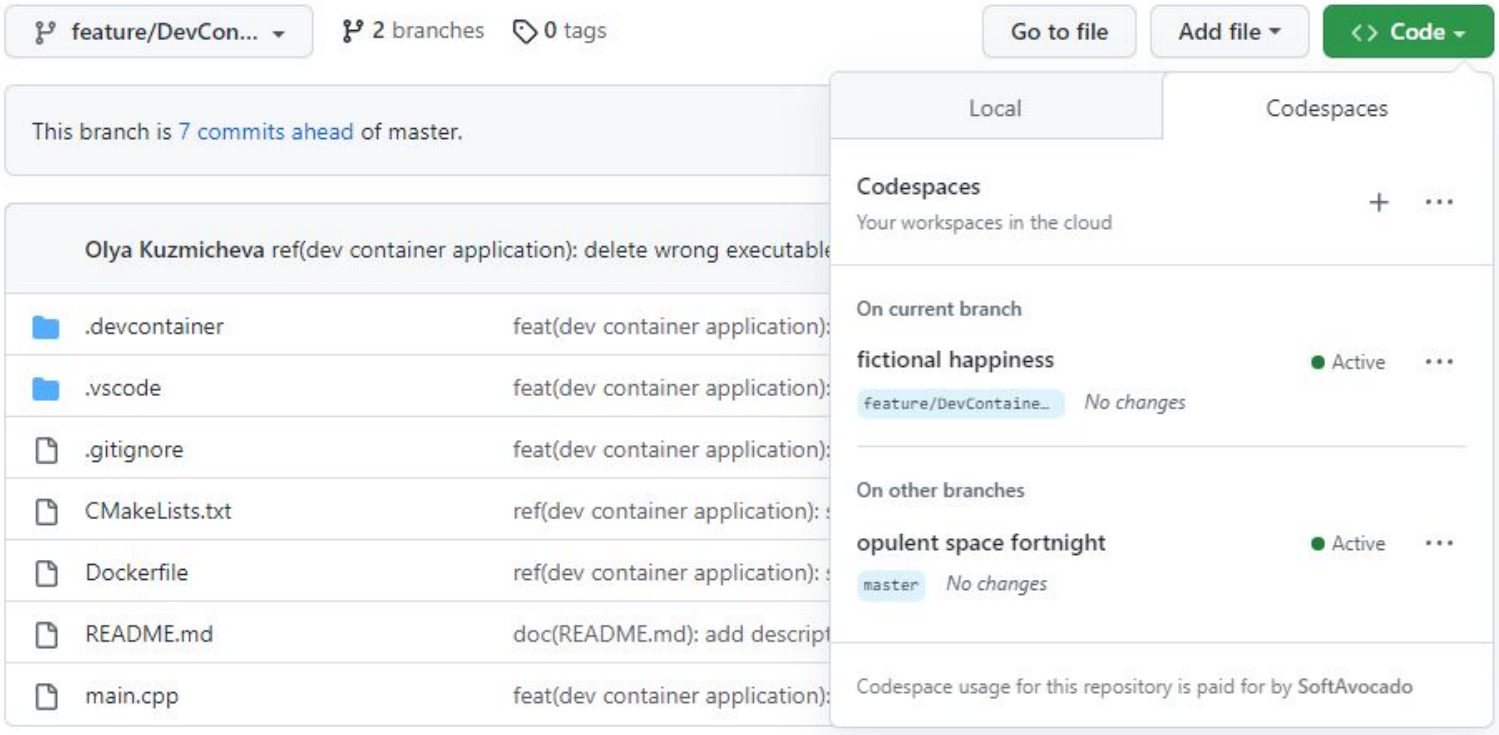

```
"customizations": {
    "vscode": {
        "extensions": [
            "ms-vscode.cpptools",
            "ms-vscode.cmake-tools",
            "twxs.cmake"
        ]
    }
}
```

Демонстрация

Запуск проекта из Github репозитория



Codespace

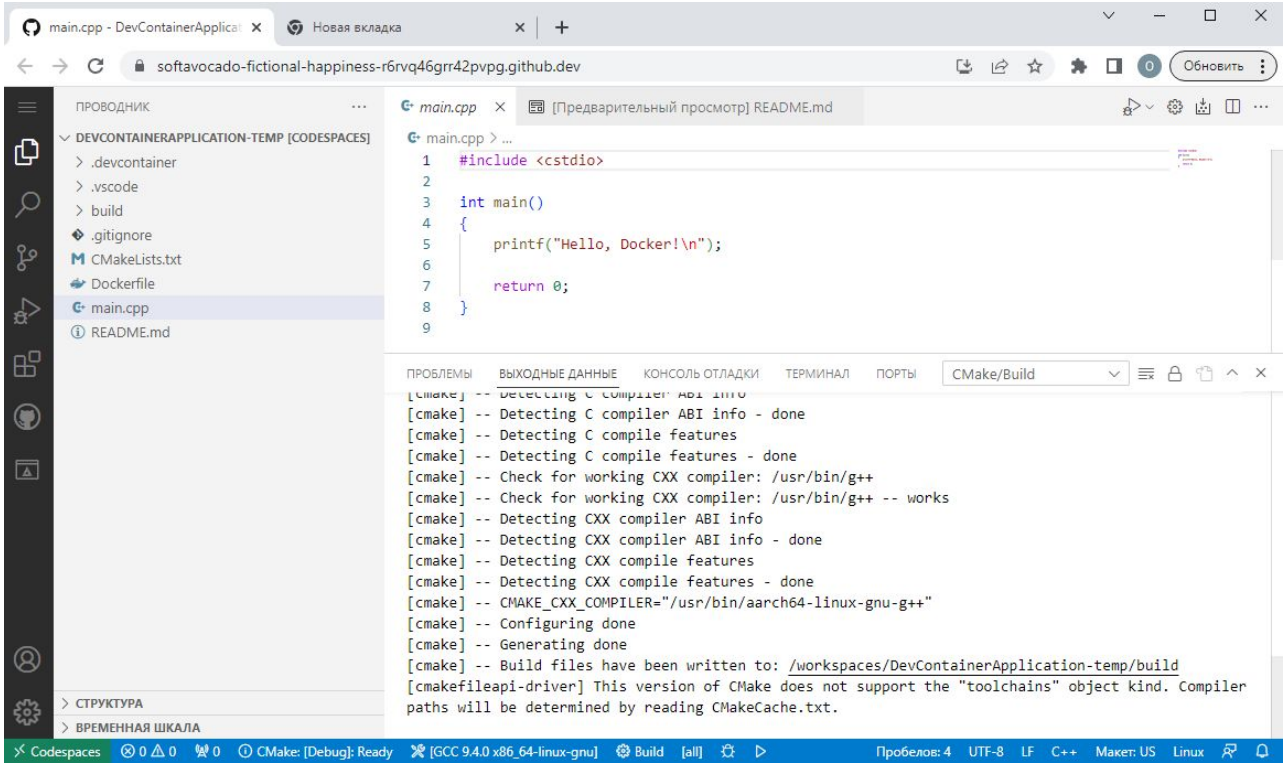

feature/DevCon...
2 branches
0 tags
Go to file
Add file
Code
This branch is 7 commits ahead of master.
Olya Kuzmicheva ref(dev container application): delete wrong executable
.devcontainer
feat(dev container application):
.vscode
feat(dev container application):
.gitignore
feat(dev container application):
CMakeLists.txt
ref(dev container application):
Dockerfile
ref(dev container application):
README.md
doc(README.md): add descript
main.cpp
feat(dev container application):
Local
Codespaces
Codespaces
Your workspaces in the cloud
On current branch
fictional happiness
Active
feature/DevContaine...
No changes
On other branches
opulent space fortnight
Active
master
No changes
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_9.4.0-1ubuntu1~20.04.1cr
oss2_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.d
eb ...
#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.d
eb ...
#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
) ...
```

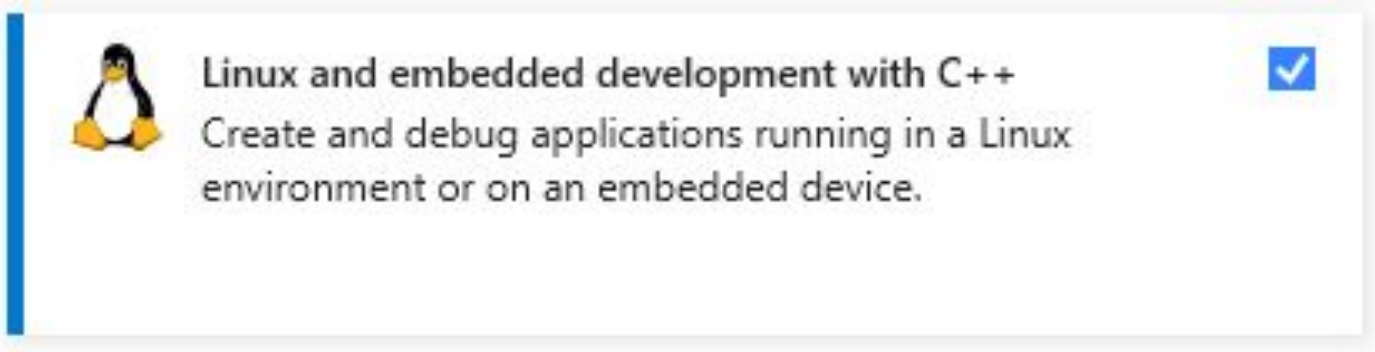
Возможности dev контейнеров в VS Code

Codespace



Dev контейнер в Visual Studio*

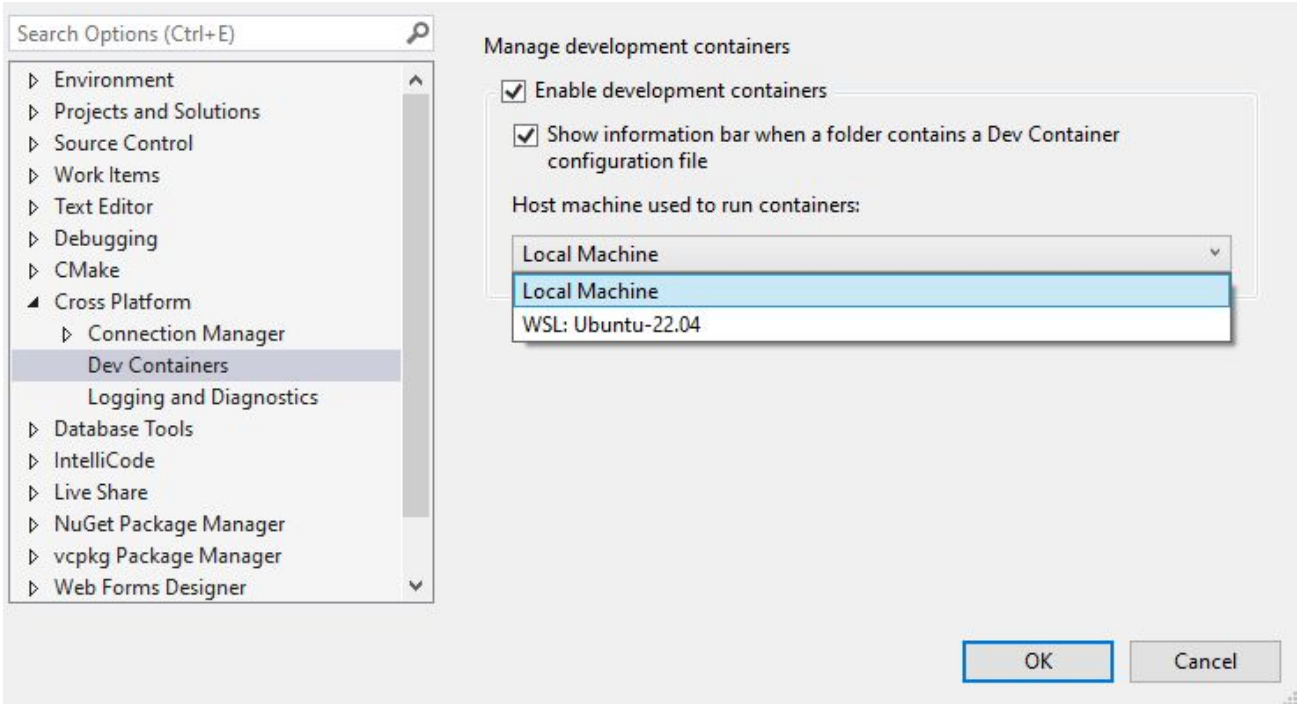
* версия 2022 17.5



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

Dev контейнер в Visual Studio*

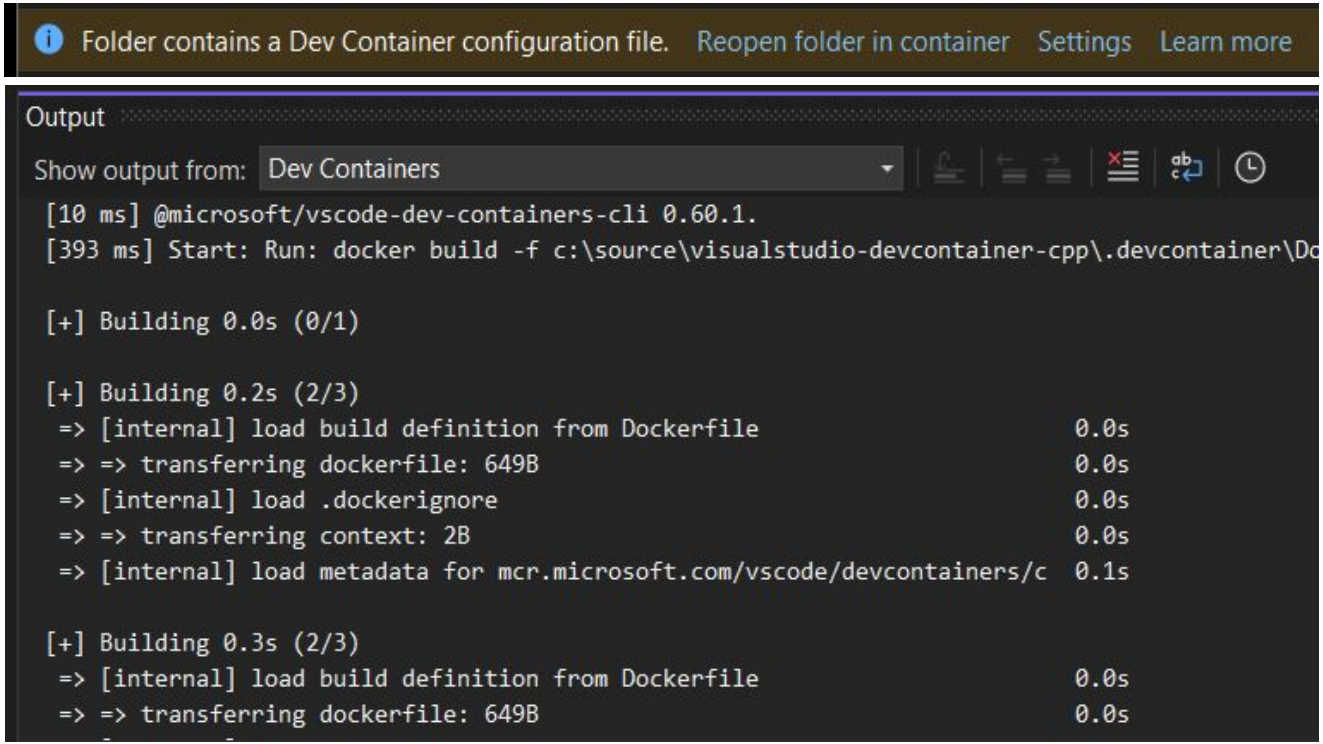
* версия 2022 17.5



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

Dev контейнер в Visual Studio*

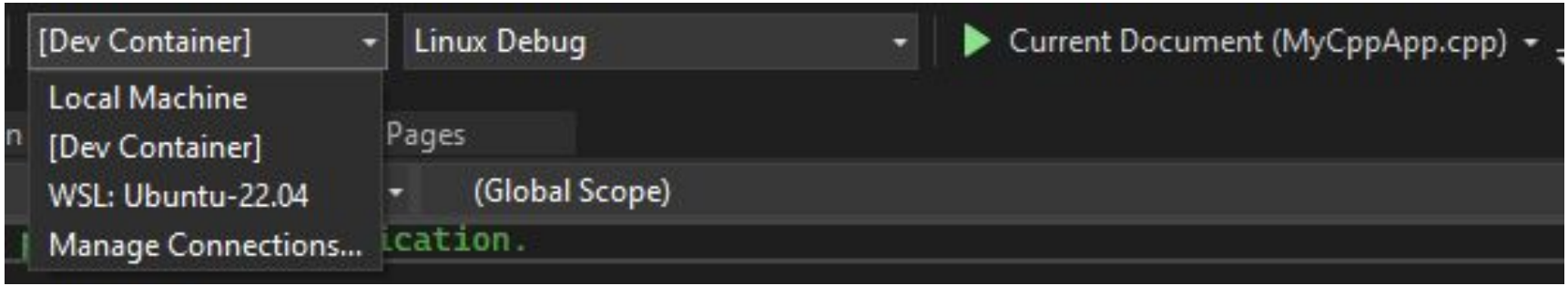
* версия 2022 17.5



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

Dev контейнер в Visual Studio*

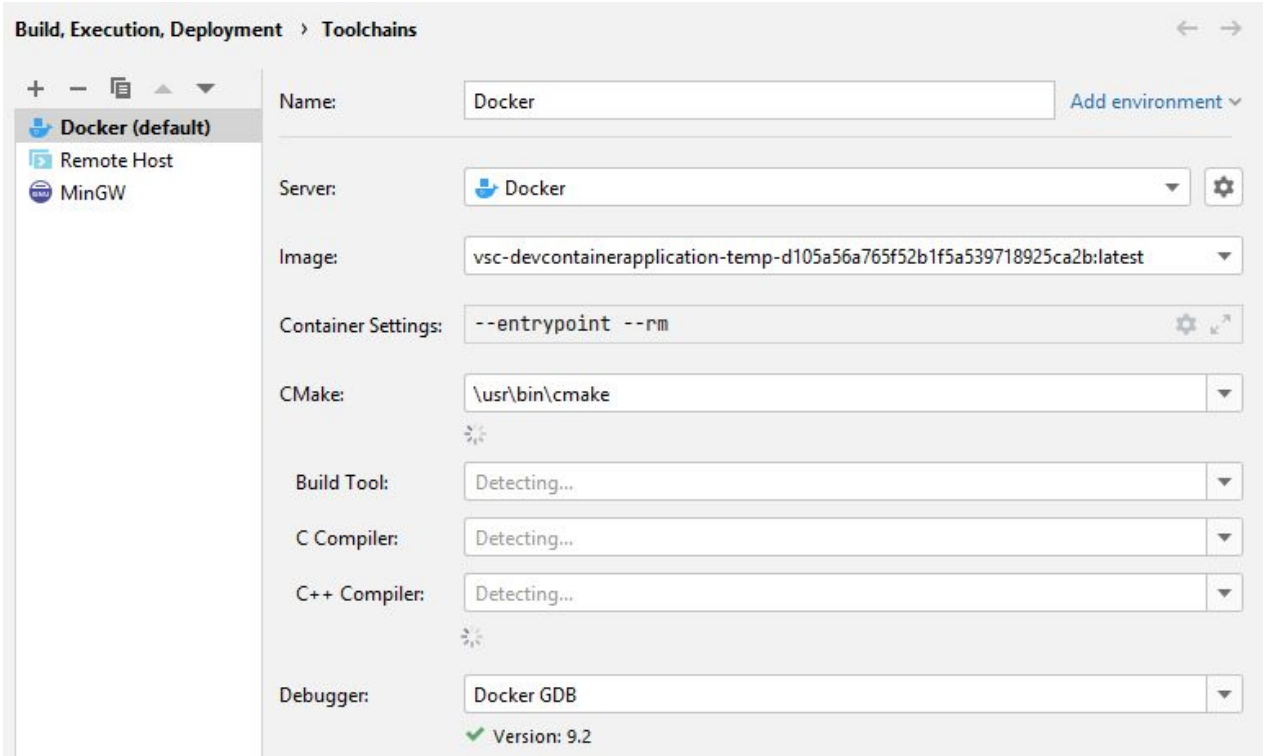
* версия 2022 17.5



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

Dev контейнер в Clion*

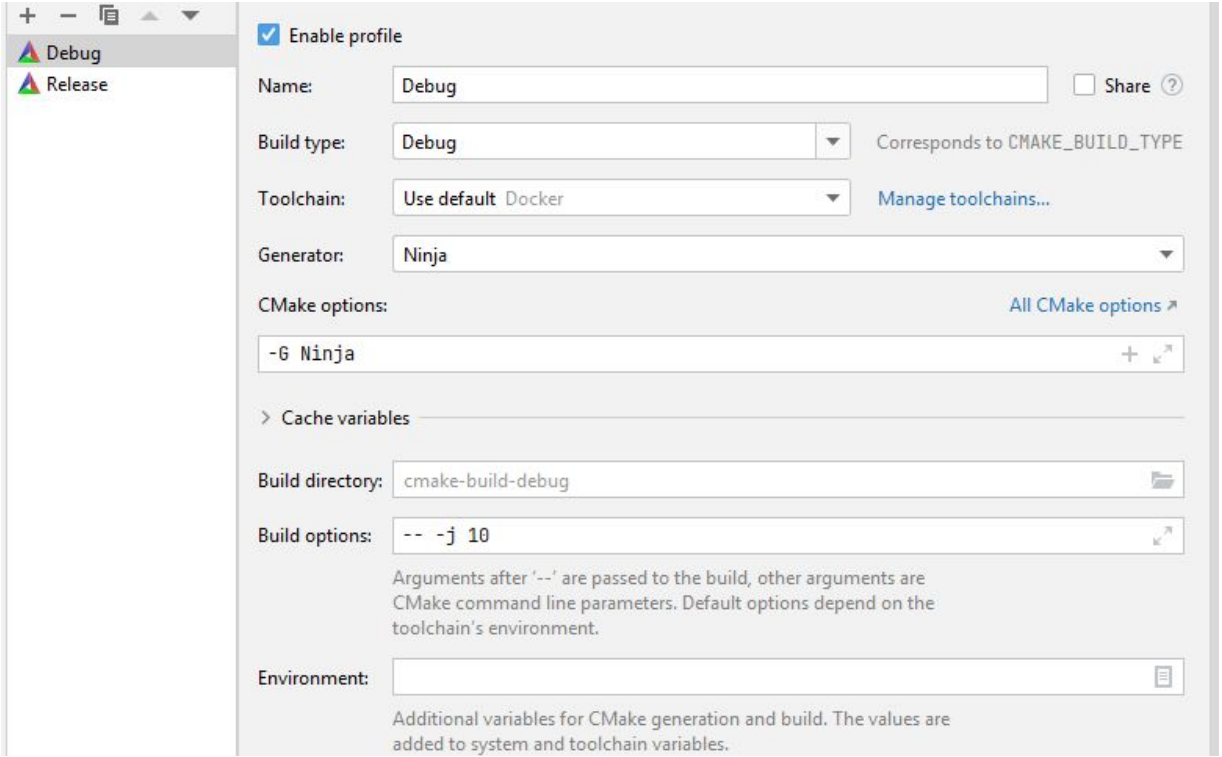
* версия 2022.2.3



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

Dev контейнер в Clion*

* версия 2022.2.3



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

- Изолированные 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
- dstarodubtcev@tourmalinecore.com

- https://github.com/TourmalineCore/DevContainerApplication

tourmaline
core