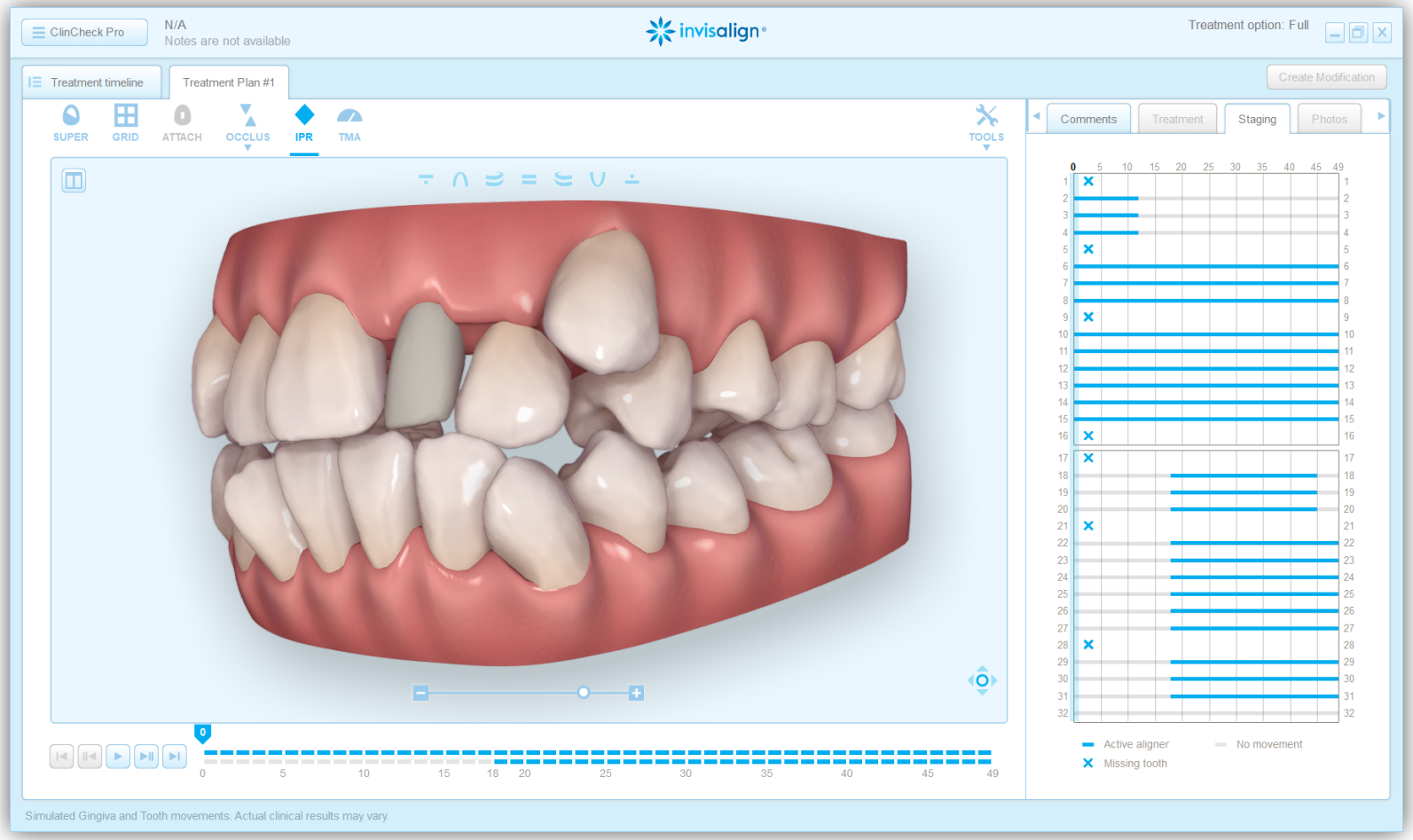
Как мы апгрейдили компилятор и поддерживали кроссплатформенность

Михаил Матросов, Expert Software Engineer, Align Technology

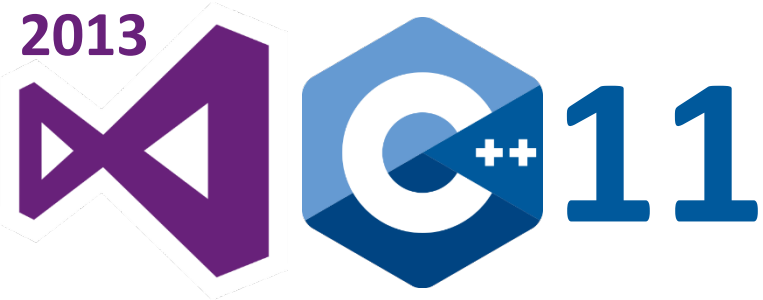
Александр Воронков, Senior Software Developer, Align technology

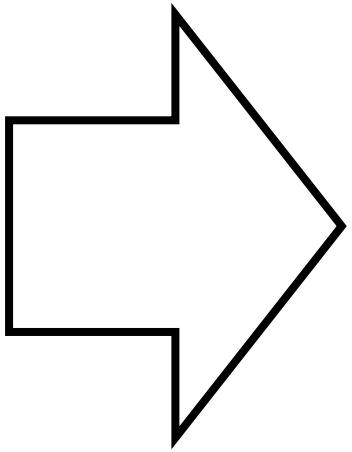
Background

- 20 лет разработки
- 2M строк кода
- 200 разработчиков
- GUI на MFC/Qt
- 80+ сторонних библиотек

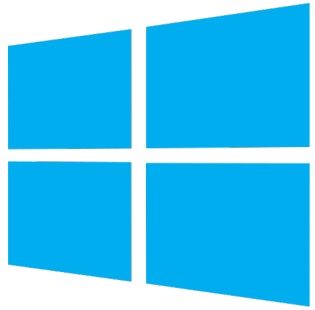


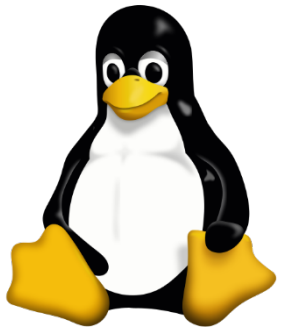
Март 2017

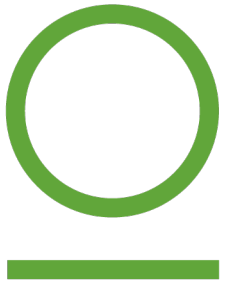




Февраль 2019







JFrog Artifactory

Уточнения – в процессе

Вопросы – в конце

Мотивация

Обоснование работы для руководства

DEVELOPERS

https://www.smbc-comics.com/comic/2010-12-09

Зачем?

- Исправлены баги компилятора, *которые приводили к багам в нашем продукте* (ссылки на багтрекер)
- Добавлены возможности компилятора, *которые позволят избежать костылей в нашем коде* (ссылки на код ревью и багтрекер)
- Уменьшено время сборки, *что сохранит время наших разработчиков и серверов* (ссылки на официальные бенчмарки)
- Улучшена производительность, *что сохранит время наших пользователей* (ссылки на официальные бенчмарки)

Why switch to new version of Visual Studio?

Created by Mikhail Matrosov, last modified just a moment ago

Tasks or issues that we should check after migration to VS2017 should be marked in JIRA with label WaitForVS2017.

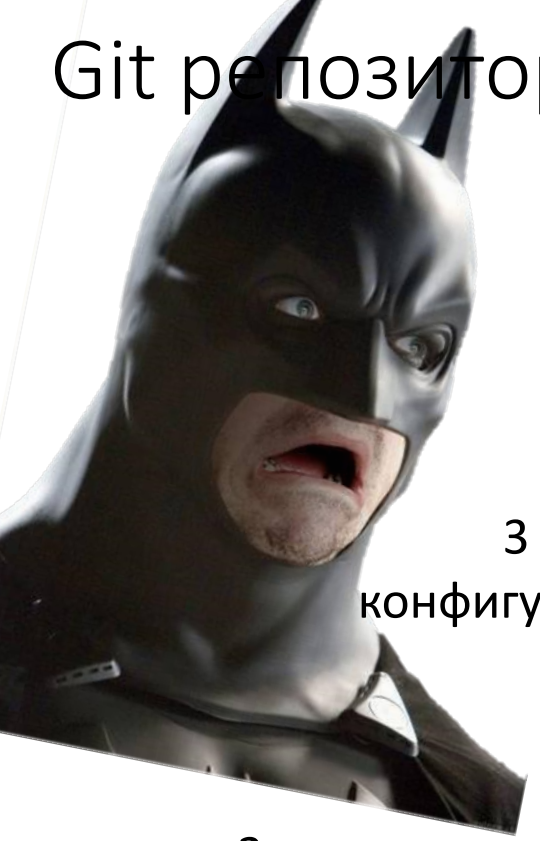
- Compiler bugs
  - std::initializer_list<Thing3D*> used in-place in range-based loop. 4 days wasted. See SDK-646.
  - Default argument as {} for a string. 3 days wasted and a broken build. See Vs 2013 u4 initialization has been fixed.
  - std::call_once global locks
  - Compiler bug for `std::async` with Non-Copyable types(`std::unique_ptr`)
    - Click here to expand...
  - Loop on std::initializer_list<std::pair<int, int>>
    - Click here to expand...
  - Compiler crashes
    - Click here to expand...
- Improved C++ support
  - Default-generated move semantics
  - Magic statics. See TRT-25362, VLB-2032, TRT-27160. Also indirectly TRT-24695.
  - Unicode string literals support. Important for SDK-839, SCANTOADF-62.
- Improved build time
  - See Recommendations to speed C++ builds in Visual Studio.
- Improved runtime performance
  - See MSVC: The best choice for Windows
- Improved solution load time.
  - See Faster C++ solution load with VS 2017.
- More powerful support of NatVis
  - See Visual Studio Debug Visualizers for more details
- Better precompiled headers support
  - In VS 2013 we have to override /Zm option for some projects, since VS 2015 it is not needed

Почему именно сейчас?

- Нам всё равно нужен Линукс

Сторонние библиотеки

Git репозиторий с бинарями



8 новых конфигураций

VS2017
Release|Debug|FastDebug
32-bit|64-bit

GCC 7.3
Release|Debug
64-bit

30Гб?..

3 новых конфигурации

VS2013
Release|Debug|FastDebug
32-bit|64-bit

17Гб

3 новых конфигурации

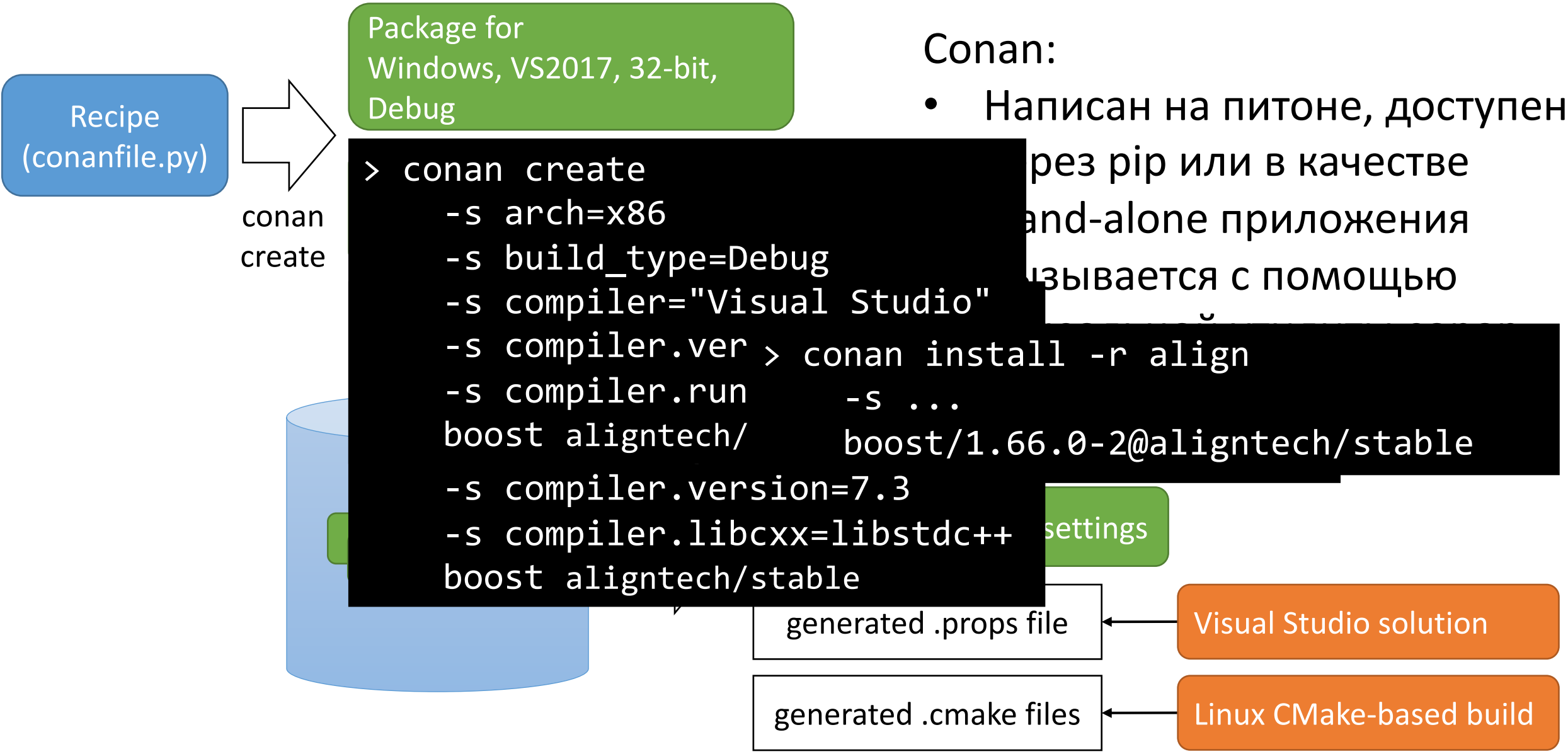
VS2013
Release|Debug|FastDebug
32-bit

VS2010
Release|Debug|FastDebug
32-bit

- Собирать вручную
- Сохранять структуру
- Не забывать пулить

Исчерпывающий список всех кросс-платформенных пакетных менеджеров для C++ по состоянию на март 2017

1. Conan



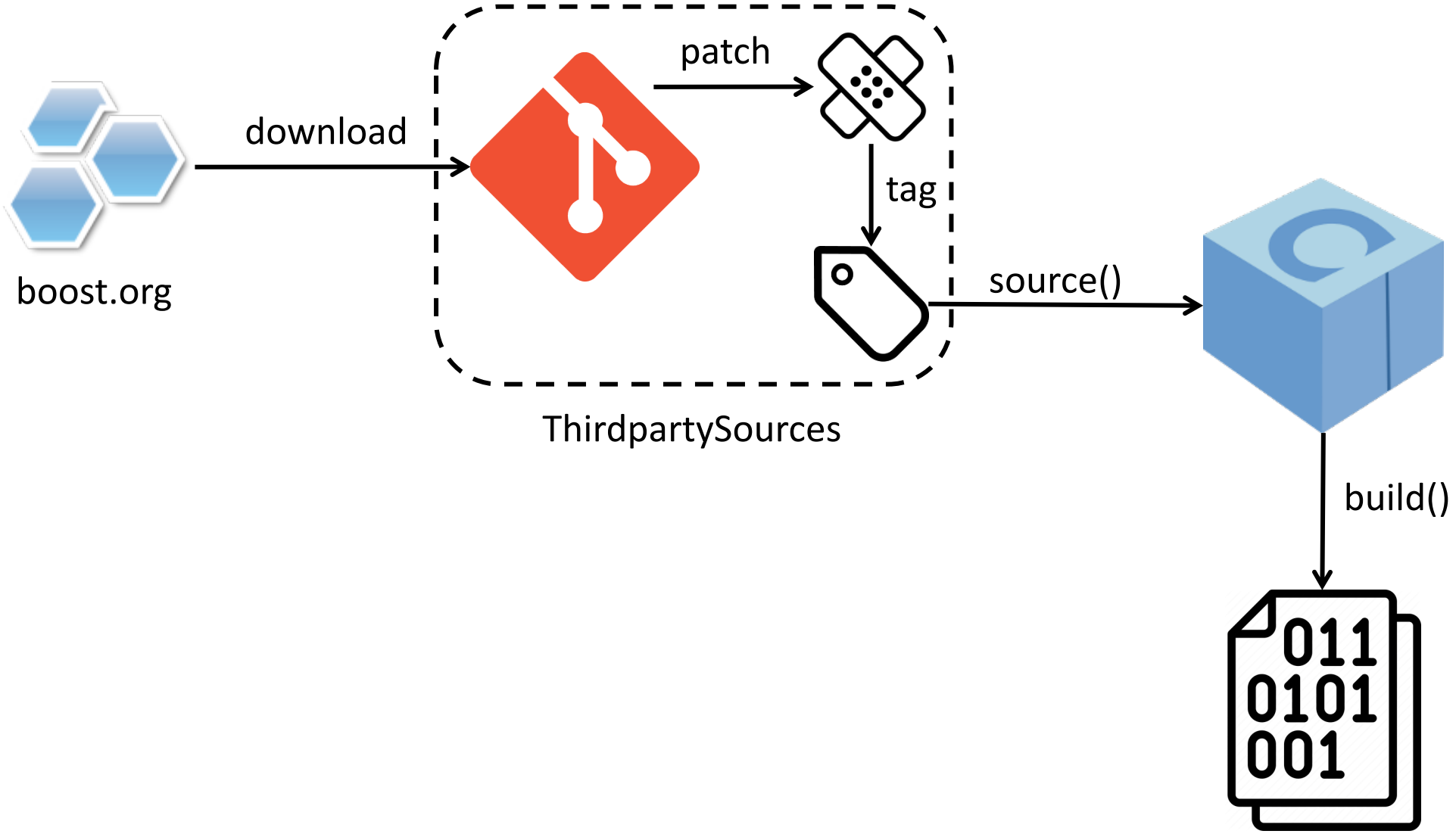
Conan:

- Написан на питоне, доступен
  рез pip или в качестве
  and-alone приложения
  изывается с помощью

```
class PDFLib(ConanFile):
    name = 'pdflib'
    lib_version = '3.03'
    revision = 1
    version = '{}-{}'.format(lib_version, revision)
    settings = 'os', 'compiler', 'build_type', 'arch'
    description = 'C library for generating PDF files'
    homepage = 'https://www.pdflib.com/'
    license = 'PDFlib license'
    build_requires = 'AlignConanBuildTools/[~1]@aligntech/stable'

    def source(self):
        from AlignConanBuildTools import SourceHelpers
        SourceHelpers(self).get_sources()

    def build(self):
        from AlignConanBuildTools import BuildHelpers
        cmake = CMake(self)
        BuildHelpers(self).configure_cmake(cmake)
        cmake.configure()
        cmake.build()
        cmake.test()
        cmake.install()

    def package_info(self):
        self.cpp_info.libs = tools.collect_libs(self)
        self.cpp_info.defines.append('PDFLIB_DLL')
```

Требования к сборке сторонних библиотек

Полностью автоматизирована	По определению
Включает запуск юнит-тестов	Легко делается в функции build()
Не зависит от внешних ресурсов (не требует интернета)	Давайте поговорим об этом
Позволяет делать патчи	


boost.org
download
patch
tag
ThirdpartySources
source()
build()
011
0101
001

Требования к сборке наших продуктов

Автоматически скачивает бинари необходимых библиотек, и только их	По определению
Полностью воспроизводима	Давайте поговорим об этом

Воспроизводимость сборок

Ревизия репозитория ⇔ собранные бинари

Подход по умолчанию: диапазон допустимых версий

```
> conan install -s ... boost/[~1.66]@aligntech/stable
```

Референс рецепта

Проблема: заливаем версию 1.67

Решение: указываем конкретную версию, без диапазонов

```
> conan install -s ... boost/1.66.0@aligntech/stable
```

Проблема: хотим пропатчить исходники

Решение: включаем номер патча в версию

```
> conan install -s ... boost/1.66.0-align1@aligntech/stable
```

Проблема: кто-то залил другую версию рецепта или пакетов для 1.66-align1. Например, с динамической линковкой для bzip2.

Решение: запрещаем перезапись на Artifactory

Проблема: но как всё же изменить рецепт без изменения версии?

Решение: вводим понятие «ревизии» рецепта

```
> conan install -s ... boost/1.66.0-align1-2@aligntech/stable
```

Отлично, но как ассоциировать референс с репозиторием?

Решение: файл CurrentReferences.yaml

```
boost: boost/1.66.0-align1-2@aligntech/stable
bzip2: bzip2/1.0.6-2@aligntech/stable
curl: curl/7.52.1-align4-2@aligntech/stable
Eigen: Eigen/3.2.8-2@aligntech/stable
easyprofiler: easyprofiler/1.2.0-2@aligntech/stable
exiv2: exiv2/0.26-3@aligntech/stable
fmt: fmt/5.3.0-2@aligntech/stable
glew: glew/2.1.0-1@aligntech/stable
glu: glu/9.0.0-1@aligntech/stable
gtest: gtest/1.8.1-align1-1@aligntech/stable
hunspell: hunspell/1.3.2-align2-1@aligntech/stable
inteltbb: inteltbb/2017_U7-align1-3@aligntech/stable
jsoncpp: jsoncpp/1.7.2-2@aligntech/stable
libjpeg: libjpeg/8d-1@aligntech/stable
libpng: libpng/1.6.28-3@aligntech/stable
log4cplus: log4cplus/1.2.0-2@aligntech/stable
minizip: minizip/2.3.1-4@aligntech/stable
minizip-tools: minizip-tools/2.3.1-3@aligntech/stable
opencv: opencv/3.2.0-align1-2@aligntech/stable
openh264: openh264/1.7.0-cisco-1@aligntech/testing
openssl: openssl/1.0.2o-2@aligntech/stable
...
```


На этом всё, что бы с нами ни произошло, вопрос с воспроизводимостью решён

Воспроизводимость

- Ревизия репозитория содержит определённый CurrentReferences.yaml
- CurrentReferences.yaml содержит определённый референс библиотеки
- Референс содержит определённую версию с патчем и ревизию рецепта
- Artifactory содержит определённый пакет для референса и настроек

AlignConanBuildTools

Общий код в рецептах

- Флаги компилятора для каждой конфигурации
- Поддержка нестандартной конфигурации сборки (FastDebug)
- Получение исходников из репозитория
- Конфигурирование переменных CMake
- Копирование DLL/SO в папку пакета

Решение – вспомогательный пакет

- Содержит .py файлы
- Рецепт вспомогательного пакета добавляет его папку в PYTHONPATH
- Добавляем вспомогательный пакет в сборочные зависимости всех наших рецептов:

```
build_requires = 'AlignConanBuildTools/[>=1.4]@aligntech/stable'
```

- И во всех методах вызываем хелперы:

```
def source(self):
    from AlignConanBuildTools import SourceHelpers
    source_helpers = SourceHelpers(self)
    source_helpers.get_sources()
```

С какими настройками нужно собирать пакеты

- Библиотеки бывают header-only, с C-интерфейсом и т.д.
- Количество вариантов настроек неодинаково
- Conan info работает только с рецептом

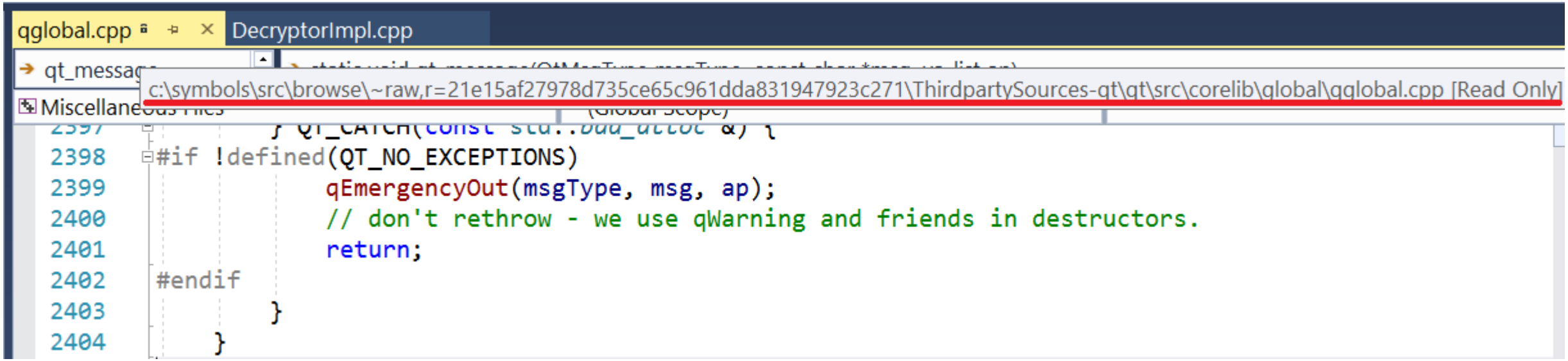
```
[build@ebf8fd22df75 ~]$ conan info zlib/1.2.11-1@aligntech/stable --paths
zlib/1.2.11-1@aligntech/stable
    ID: cc7c6ef50e2fc4146124f5d9a34e6165110c37d1
    BuildID: None
    export_folder: /home/build/.conan/data/zlib/1.2.11-1/aligntech/stable/export
    source_folder: /home/build/.conan/data/zlib/1.2.11-1/aligntech/stable/source
    build_folder: /home/build/.conan/data/zlib/1.2.11-1/aligntech/stable/build/cc7c6
    package_folder: /home/build/.conan/data/zlib/1.2.11-1/aligntech/stable/package/c
```

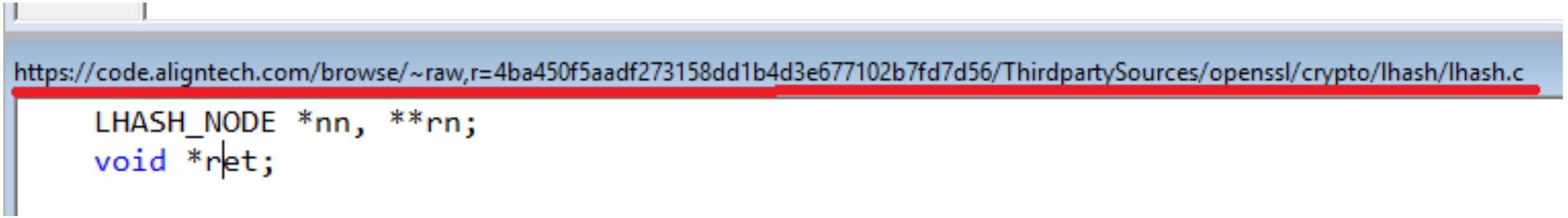
- Для каждой конфигурации вызываем *conan info* , анализируем вывод на предмет поля ID, и путей где искать PDB после сборки

```
D:\dev>conan search zlib/1.2.11-1@aligntech/stable
Existing packages for recipe zlib/1.2.11-1@aligntech/stable:

    Package_ID: 2bb76c9adac7b8cd7c5e3b377ac9f06934aba606
        [settings]
            arch: x86
            build_type: Release
            compiler: Visual Studio
            compiler.runtime: MD
            compiler.version: 15
            os: Windows
        Outdated from recipe: False

    Package_ID: 2f1ddea8e7b02c3635ae6775fdc3efc0b49998b5
        [settings]
            arch: x86_64
            build_type: FastDebug
            compiler: Visual Studio
            compiler.runtime: MDd
            compiler.version: 15
            os: Windows
        Outdated from recipe: False
```

А дальше все просто

Индексация исходников в PDB





[Раздел MSDN посвященный SrcSrv]

CMake+Conan (для сборки наших проектов)

Как предлагают интегрироваться с CMake

- Создать conanfile со всеми зависимостями
- Выполнить conan install до конфигурации CMake
- Подключить модули сгенерированные Conan
- Использовать targets или переменные из модулей

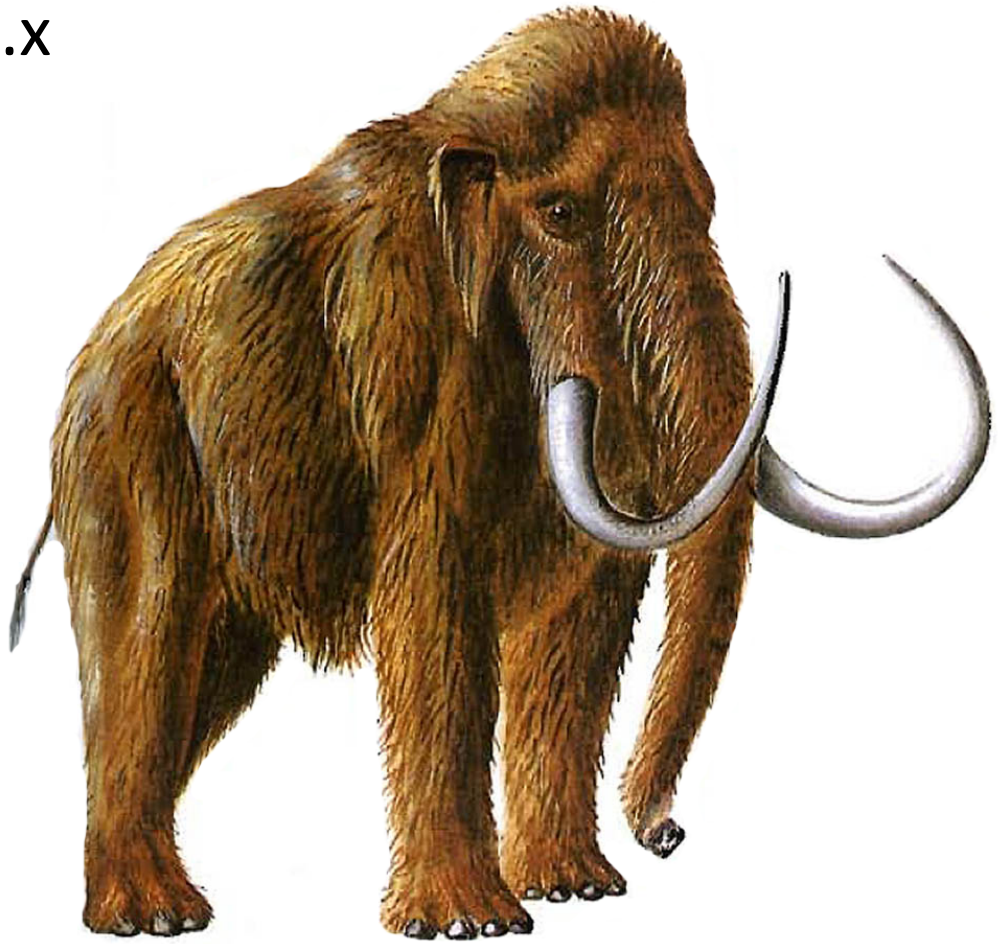

Они предлагают
заранее указать все
нужные пакеты, Карл!
C
BLOCK

Как хотелось нам

```
target_add_conan_packages(AlignSdkModel
    PRIVATE boost_iostreams boost_filesystem boost_system gtest
    PUBLIC log4cplus jsoncpp)
```

Первый вариант

- Мы не были уверены в наличии свежих версий CMake везде
- Использовался стиль CMake 2.x



Как мы перестали волноваться и полюбили modern CMake

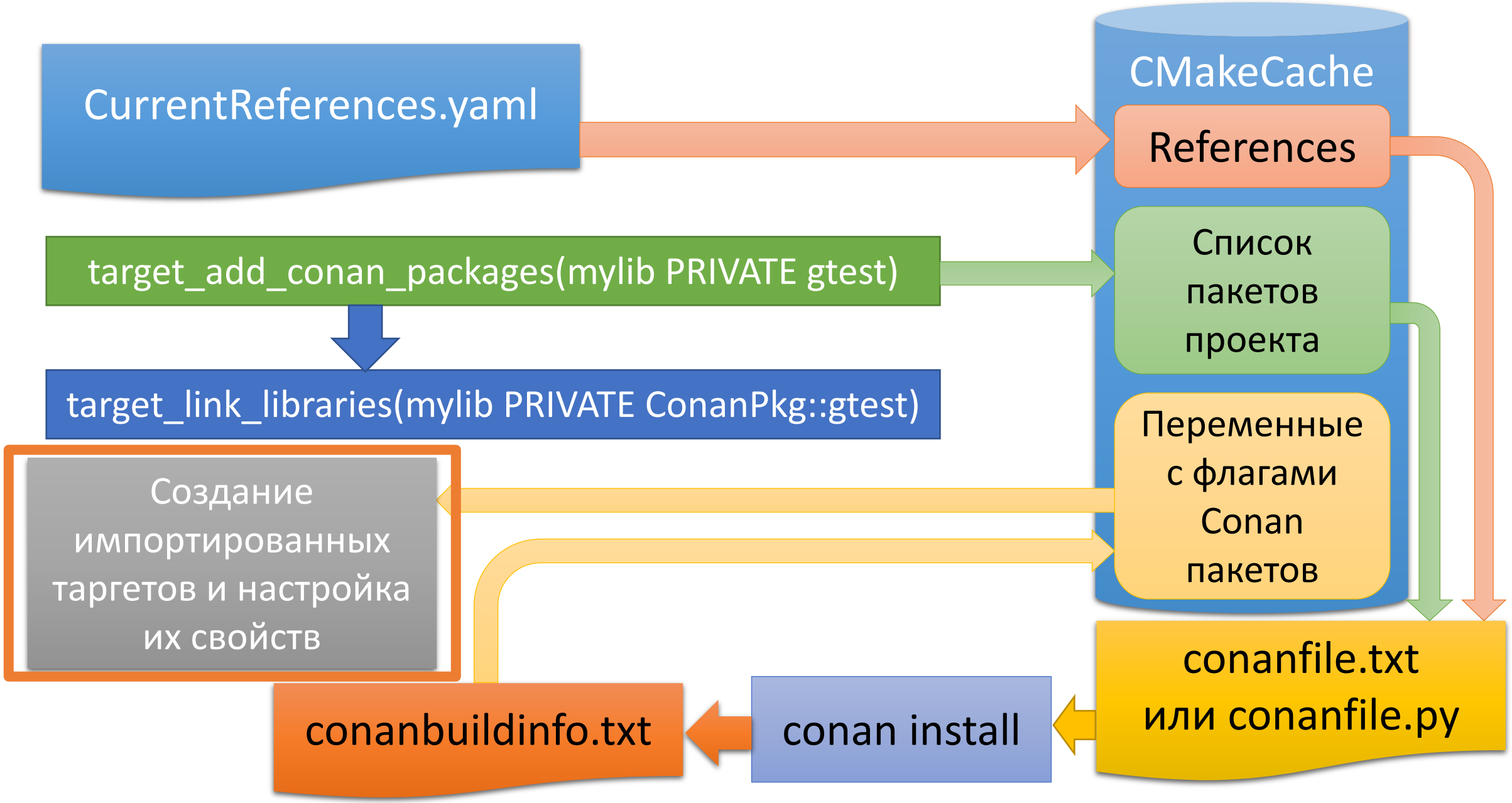
У нас всегда есть python и pip

```
[build@ebf8fd22df75 ~]$ pip install cmake --user
Collecting cmake
  Downloading https://files.pythonhosted.org/packages/45/c4/e69313ade2a3e992e7178744b0e56bdd8f23e79e15066a68cf490504beed/cmake-3.13.3-cp36-cp36m-manylinux1_x86_64.whl (15.9MB)
    100% |████████████████████████████████| 15.9MB 2.1MB/s
Installing collected packages: cmake
Successfully installed cmake-3.13.3
[build@ebf8fd22df75 ~]$
```

Target-based подход

- Появился начиная с CMake 3.1
- В CMake 3.11 появились IMPORTED INTERFACE targets

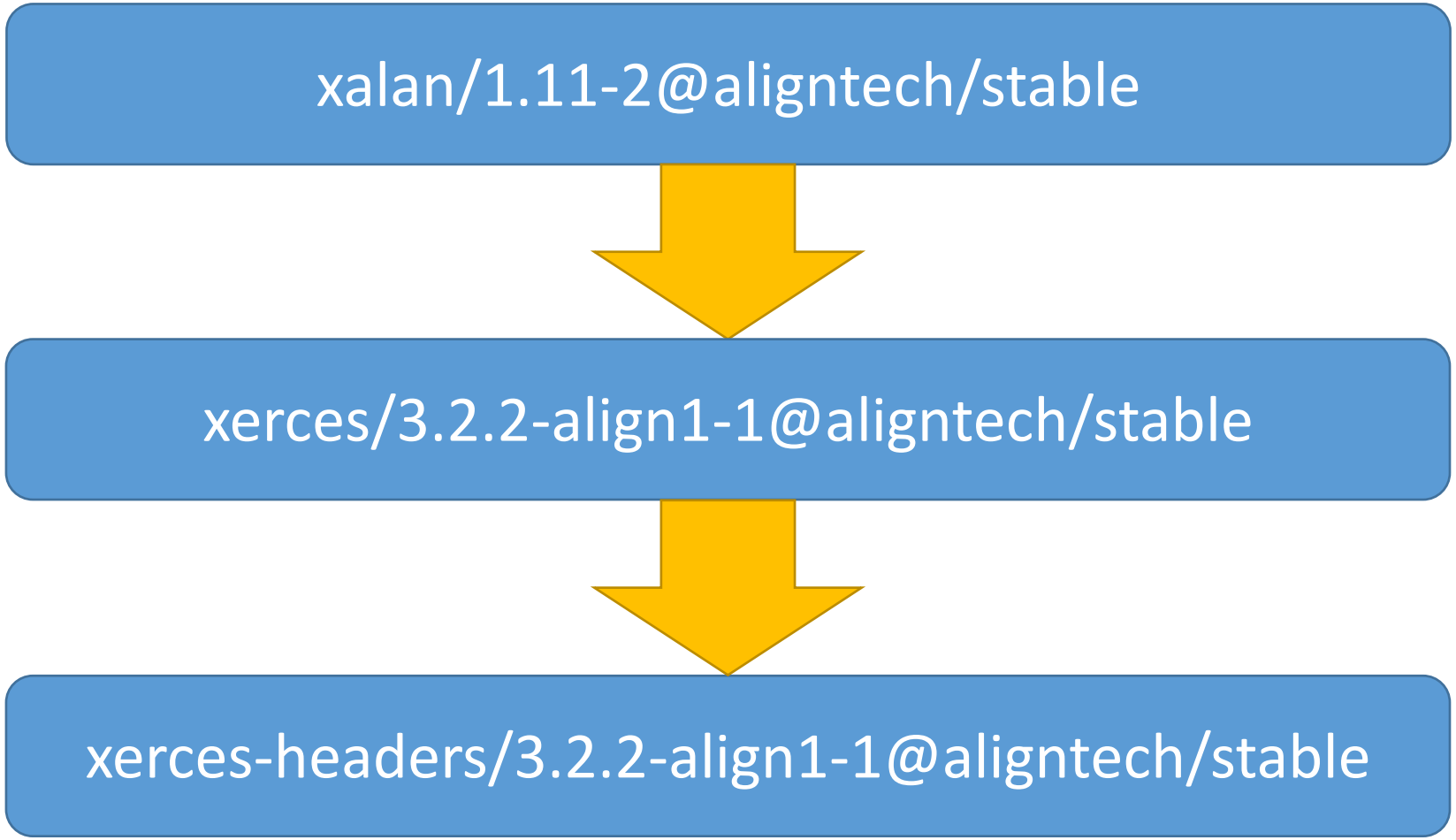
```
if(NOT TARGET ConanPkg::${pkgname})
    add_library(ConanPkg::${pkgname} INTERFACE IMPORTED GLOBAL)
endif()
```

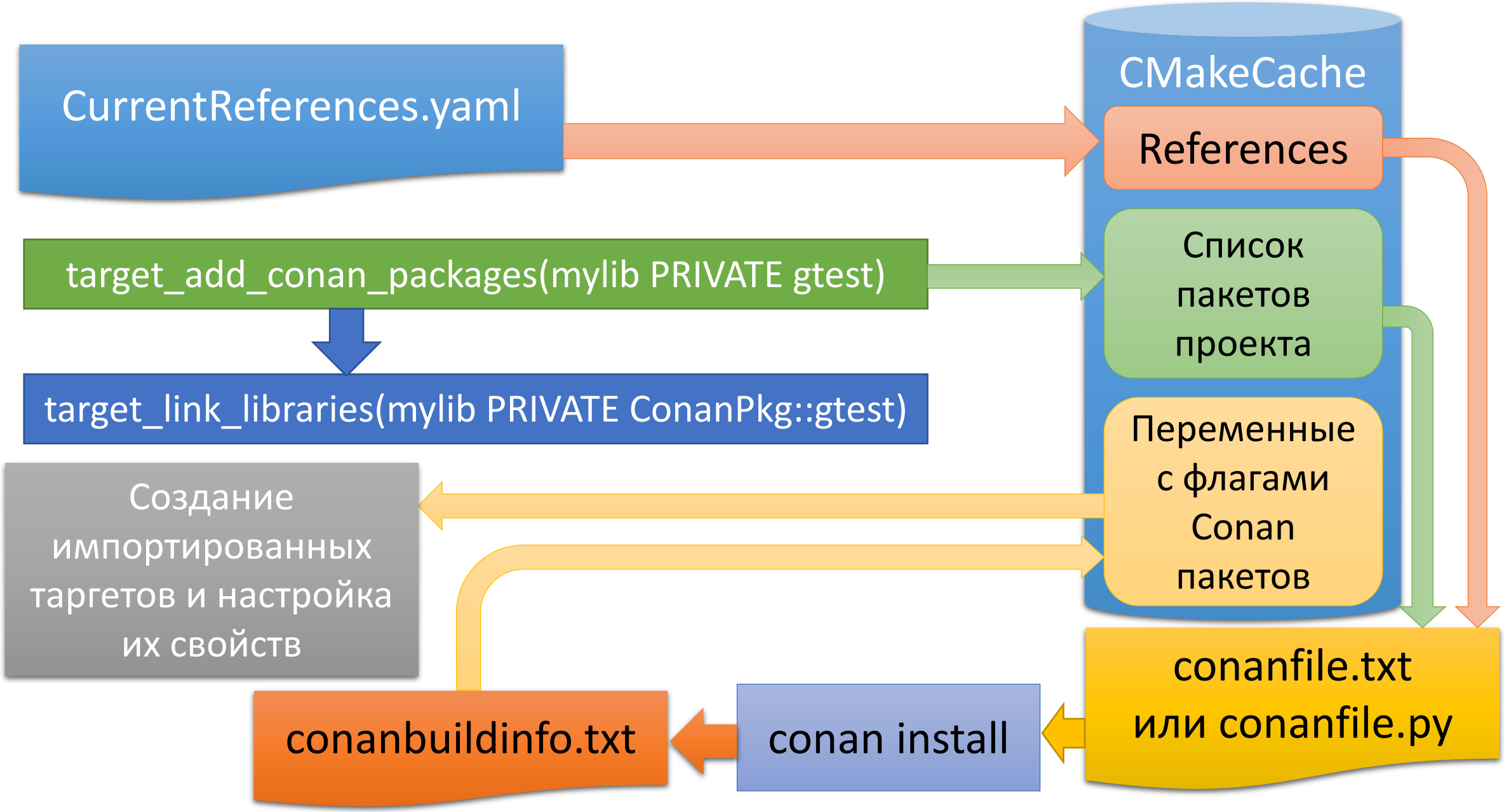

CurrentReferences.yaml
CMakeCache
References
target_add_conan_packages(mylib PRIVATE gtest)
Список пакетов проекта
target_link_libraries(mylib PRIVATE ConanPkg::gtest)
Переменные с флагами Conan пакетов
Создание импортированных таргетов и настройка их свойств
conanfile.txt или conanfile.py
conanbuildinfo.txt
conan install

Настройка свойств таргетов

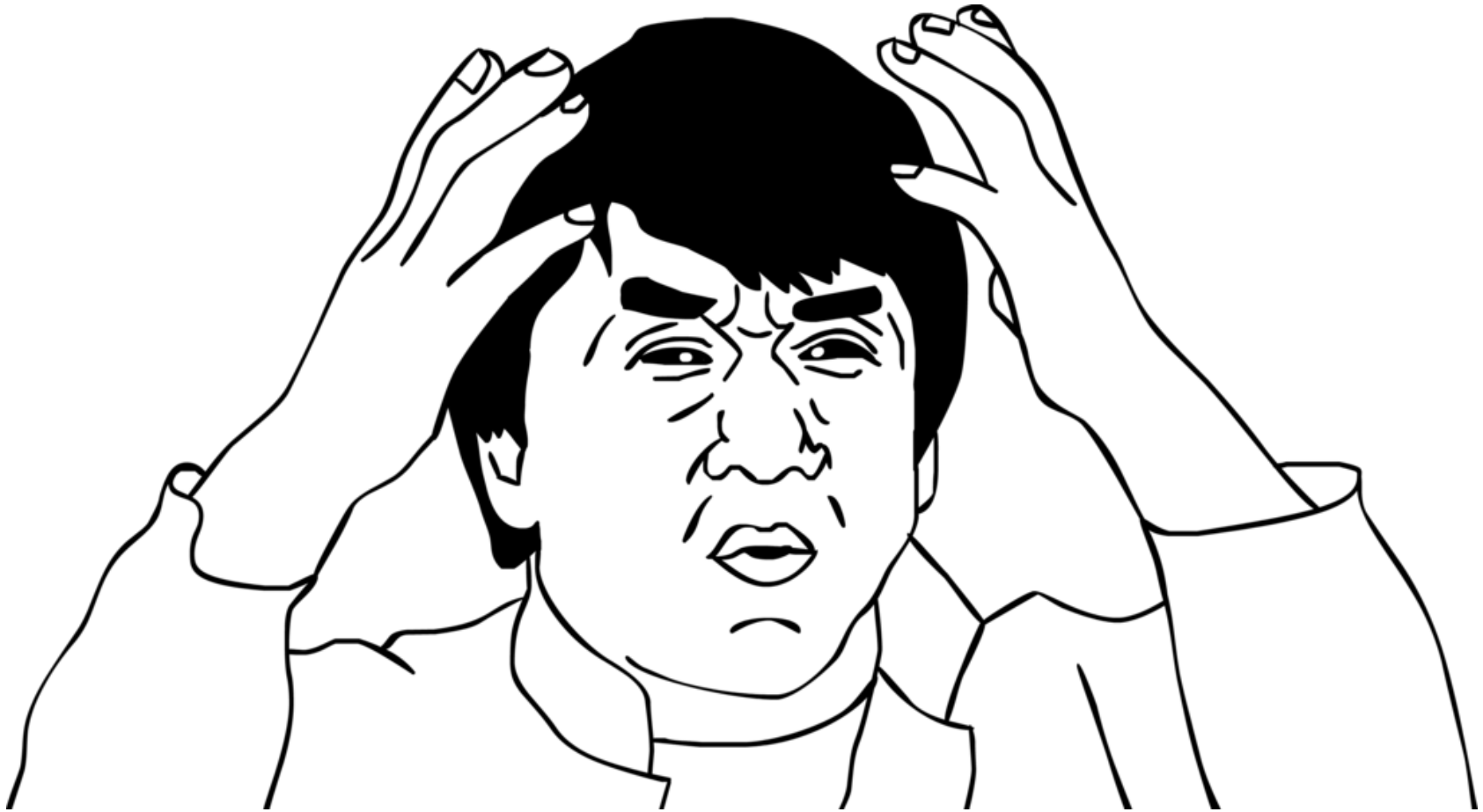
```
if(CONAN_PACKAGE_LIBDIRS_${pkgname}_${config})
    string(REPLACE ";" ";${CMAKE_LIBRARY_PATH_FLAG}" libdirs
        "${CMAKE_LIBRARY_PATH_FLAG}${CONAN_PACKAGE_LIBDIRS_${pkgname}_${config}}")
    target_link_libraries(ConanPkg::${pkgname} INTERFACE
        "$<$<CONFIG:${config}>:${libdirs}>")
endif()
target_link_libraries(ConanPkg::${pkgname} INTERFACE
    "$<$<CONFIG:${config}>:${CONAN_PACKAGE_LIBS_${pkgname}_${config}}>")
```

Зависимости между пакетами




CurrentReferences.yaml
CMakeCache
References
target_add_conan_packages(mylib PRIVATE gtest)
Список пакетов проекта
target_link_libraries(mylib PRIVATE ConanPkg::gtest)
Переменные с флагами Conan пакетов
Создание импортированных таргетов и настройка их свойств
conanfile.txt или conanfile.py
conanbuildinfo.txt
conan install

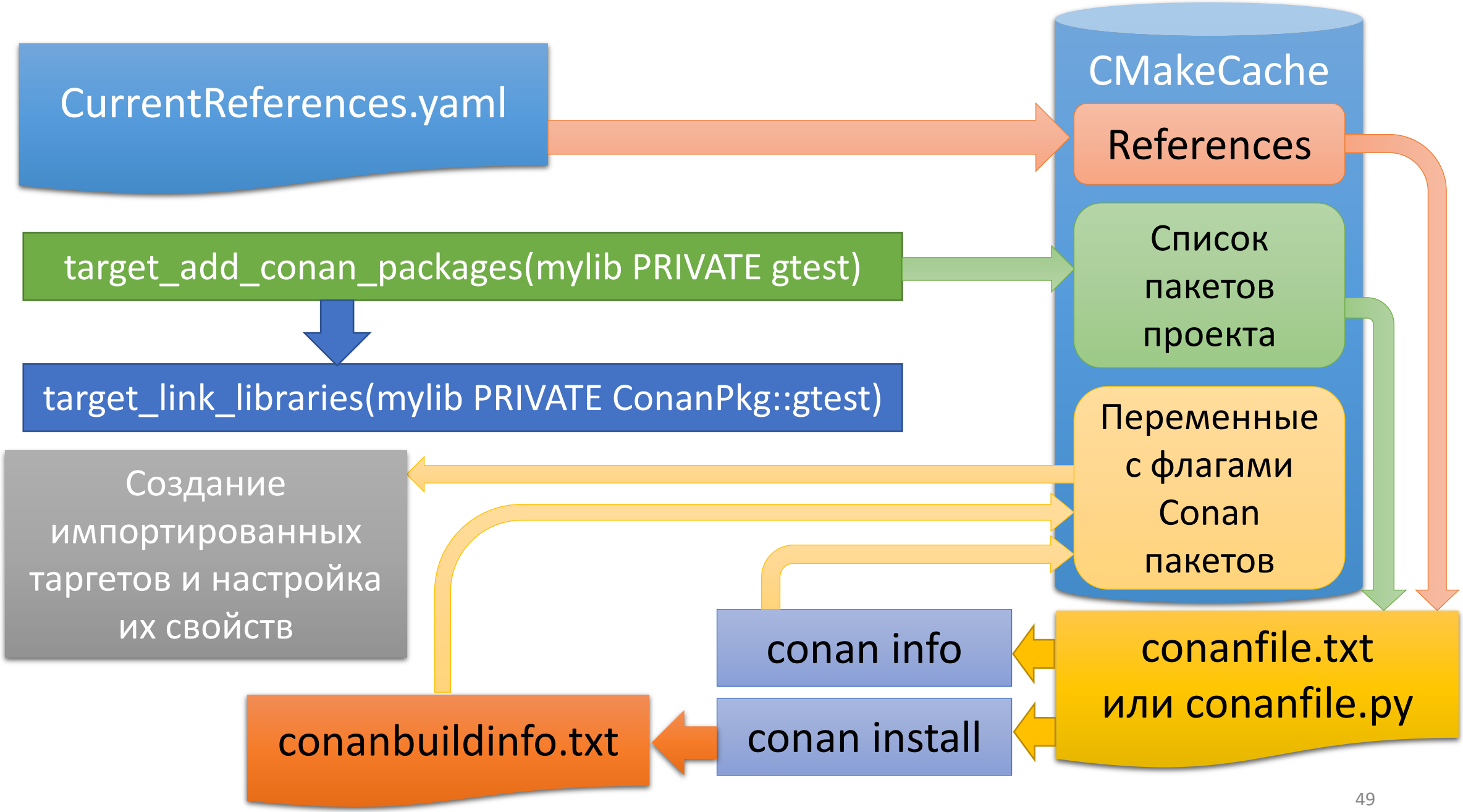
В *conanbuildinfo.txt* нет информации о зависимостях между пакетами



Придется спросить у Conan

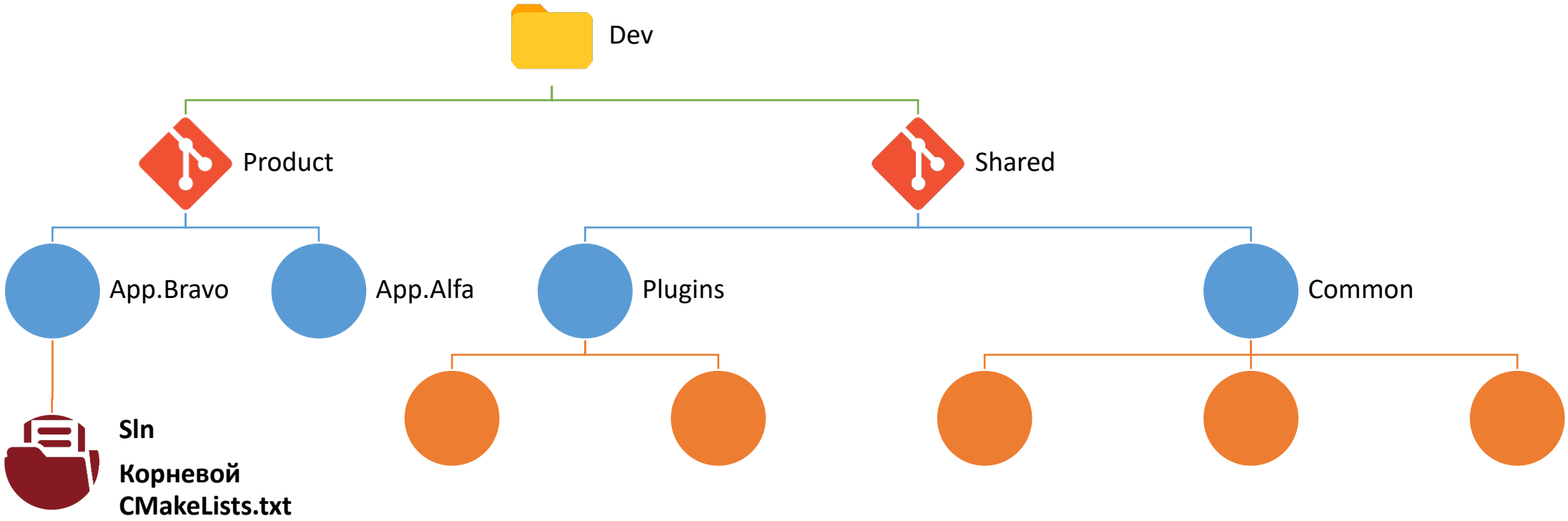
conan info -n requires --package-filter=*@* conanfile.txt

```
valijson/0.1-align1-1@aligntech/stable
xalan/1.11-2@aligntech/stable
    Requires:
        xerces/3.2.2-align1-1@aligntech/stable
xerces/3.2.2-align1-1@aligntech/stable
    Requires:
        xerces-headers/3.2.2-align1-1@aligntech/stable
xerces-headers/3.2.2-align1-1@aligntech/stable
xmlsecurity/2.0.2-align1-2@aligntech/stable
    Requires:
        xerces/3.2.2-align1-1@aligntech/stable
        openssl/1.0.2o-2@aligntech/stable
```

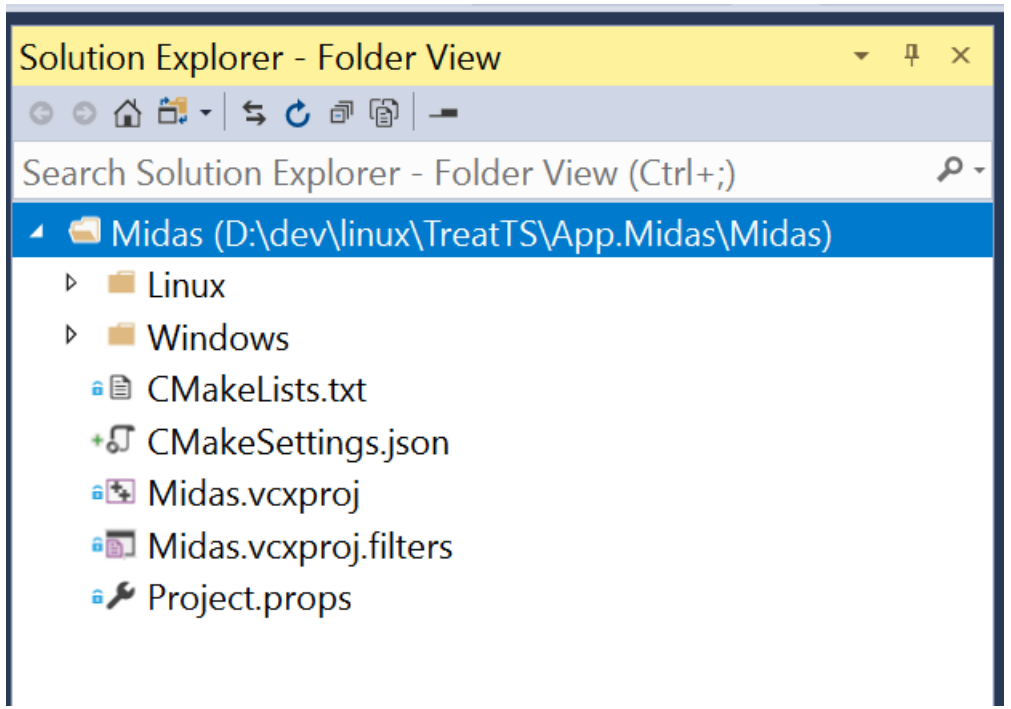

CurrentReferences.yaml
CMakeCache
References
target_add_conan_packages(mylib PRIVATE gtest)
Список пакетов проекта
target_link_libraries(mylib PRIVATE ConanPkg::gtest)
Переменные с флагами Conan пакетов
Создание импортированных таргетов и настройка их свойств
conan info
conanfile.txt или conanfile.py
conanbuildinfo.txt
conan install

CMake&Linux in VS2017

Большой проект

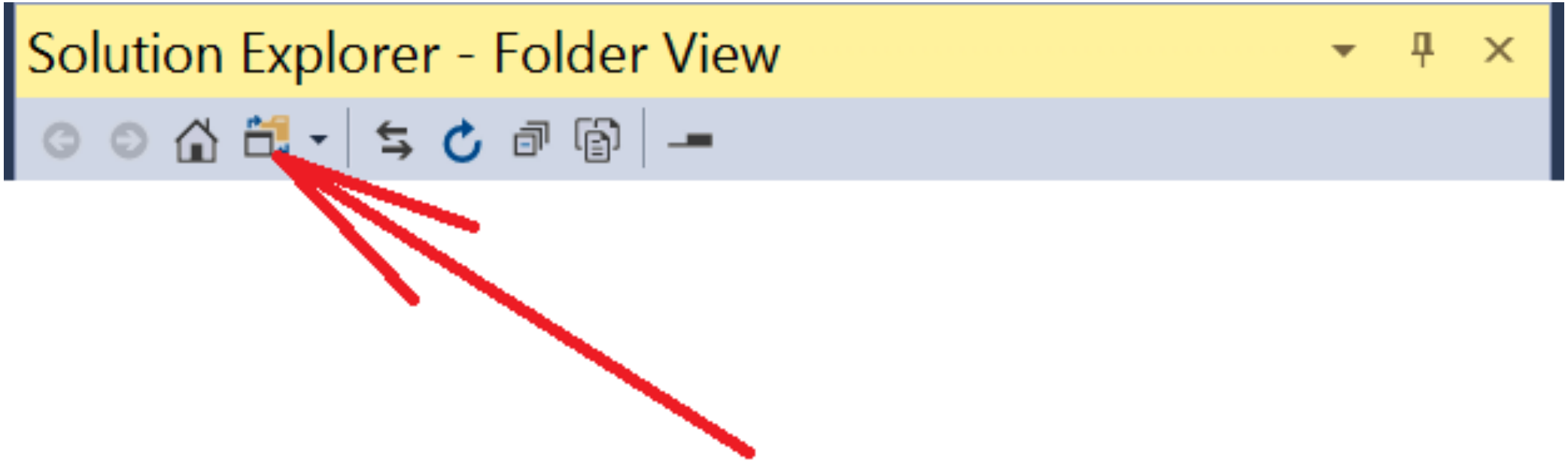


Проблема корневой папки

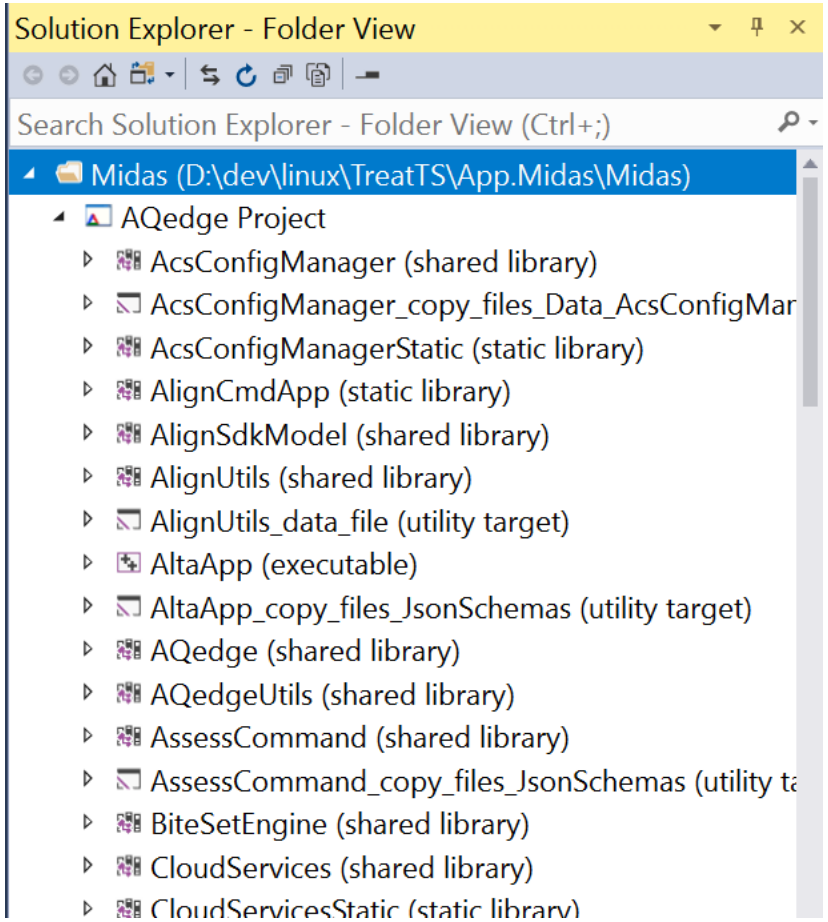


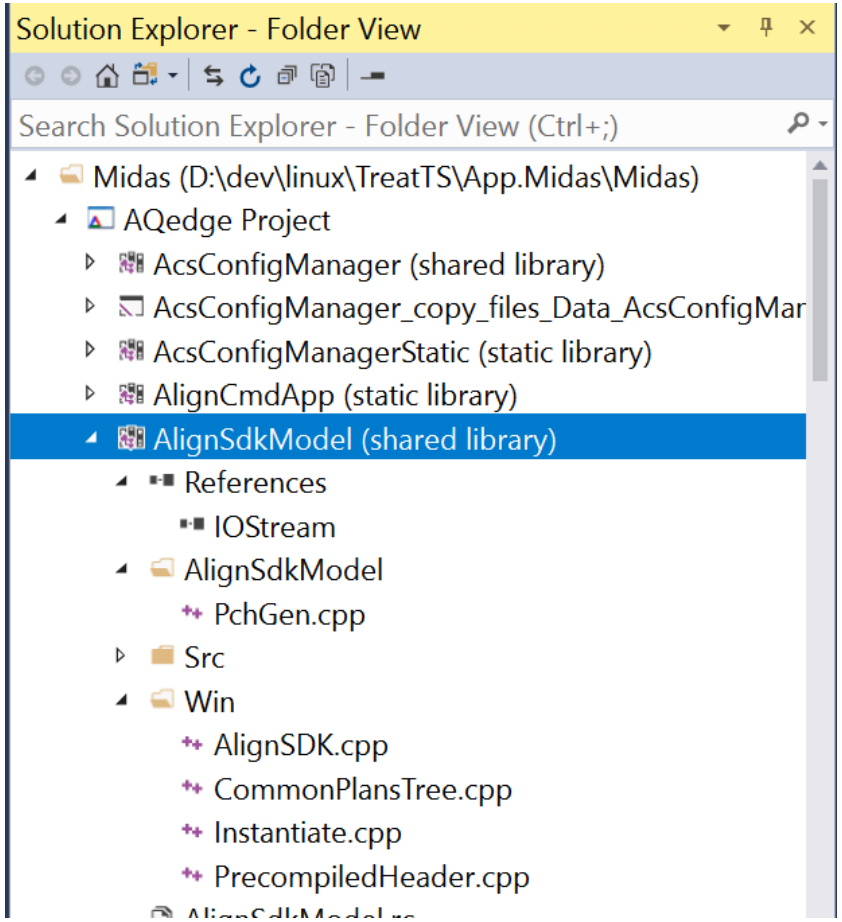
- Visual Studio 2017 Отображает только открытую папку, где и должен находится корневой CMakeList.txt

Magic button



CMake Targets View





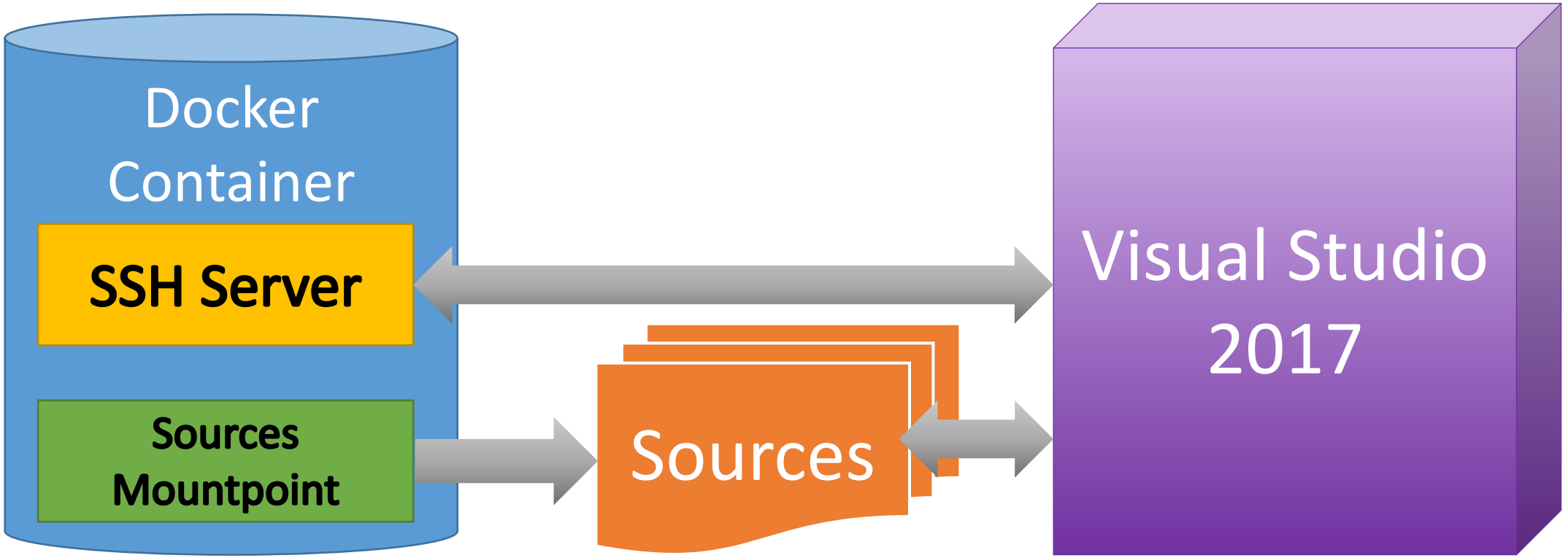
Тогда получается, поддержка CMake в Visual Studio 2017

БЕСПОЛЕЗНА???

Не совсем!

- Мы все равно можем использовать удаленную отладку Linux приложения!!!

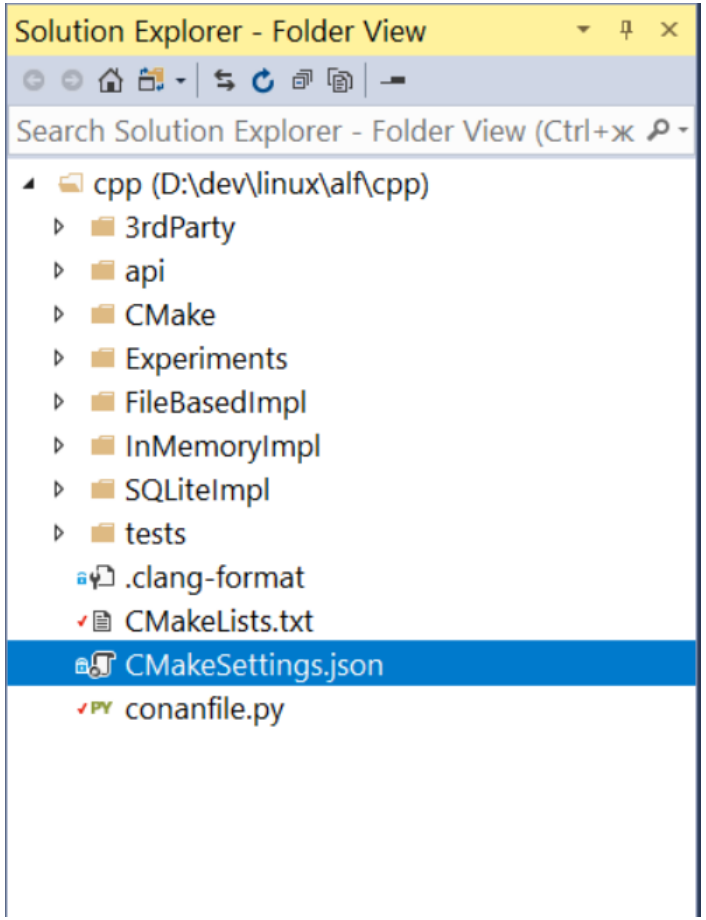
Рабочее место кроссплатформенного разработчика



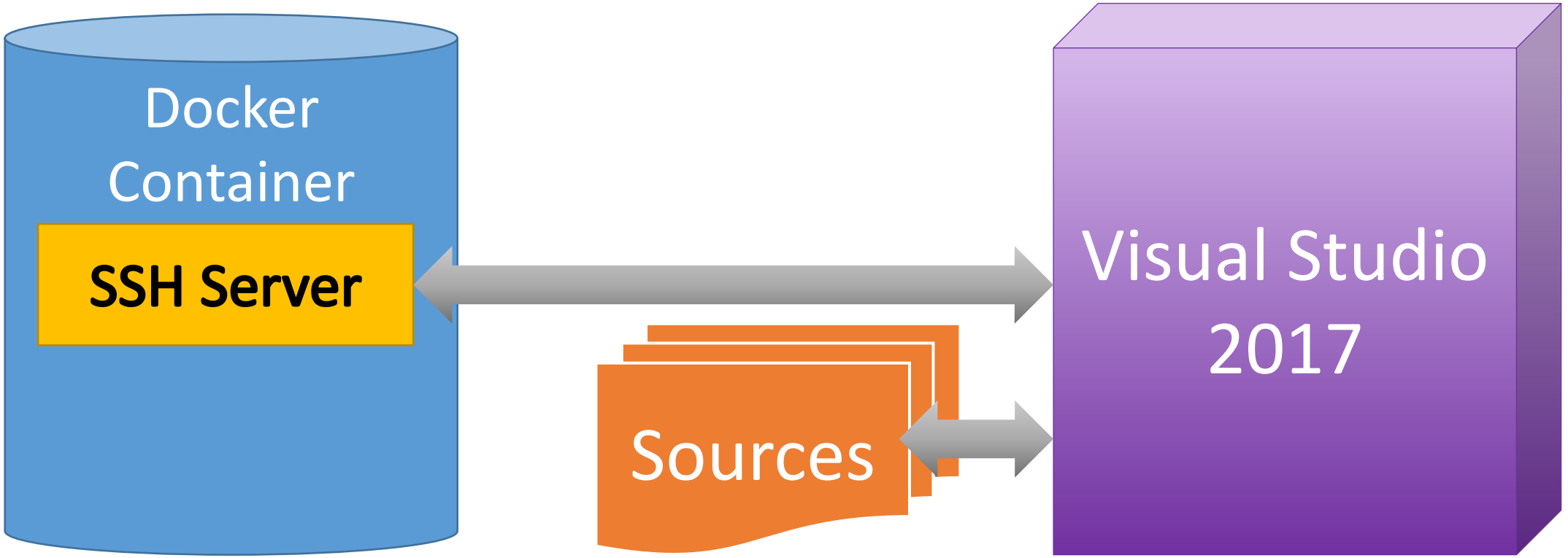
```
{
  "name": "Linux-Release",
  "generator": "Ninja",
  "configurationType": "Release",
  "buildRoot": "${env.USERPROFILE}\\CMakeBuilds\\${workspaceHash}\\build\\${name}",
  "installRoot": "${env.USERPROFILE}\\CMakeBuilds\\${workspaceHash}\\install\\${name}'
  "cmakeExecutable": "/usr/local/bin/cmake",
  "remoteCopySourcesExclusionList": [ ".vs", ".git" ],
  "cmakeCommandArgs": "",
  "buildCommandArgs": "",
  "ctestCommandArgs": "-V",
  "inheritEnvironments": [ "linux_x64" ],
  "intelliSenseMode": "linux-gcc-x64",
  "remoteMachineName": "${defaultRemoteMachineName}",
  "remoteCMakeListsRoot": "/data/TreatTS/App.Midas/Midas",
  "remoteBuildRoot": "/home/build/midas",
  "remoteInstallRoot": "/home/build/install/${name}",
  "remoteCopySources": false,
  "rsyncCommandArgs": "-t --delete --delete-excluded",
  "remoteCopyBuildOutput": false,
  "remoteCopySourcesMethod": "rsync",
  "variables": []
}
```

Если у нас маленький проект

- Правильная структура проекта позволяет работать с ним также удобно как и с нативным VisualStudio Solution
- Кроме GUI для настроек проекта
- Все нужно описывать в CMake
- Нужно знать флаги компилятора



Использование поддержки CMake и Linux в маленьком проекте



Внимание! RHEL и производные

- Проще всего CMake из pip
- Если используете EPEL версию CMake, нужен враппер

```
#!/bin/bash
if [[ "$1" =~ ^-?-version$ ]] ; then
  cmake3 --version | sed 's/^cmake3/cmake/'
else
  cmake3 "$@"
fi
```

- Не забудьте настроить правильный путь к gdb из используемого devtoolset в launch.vs.json

Воспроизводимость сборок в Visual Studio 2017

Visual Studio 2017 обновляется каждые две недели

Visual Studio 2017 version 15.9 Releases

- April 02, 2019 -- Visual Studio 2017 version 15.9.11 Servicing Update **New**
- March 25, 2019 -- Visual Studio 2017 version 15.9.10 Servicing Update
- March 12, 2019 -- Visual Studio 2017 version 15.9.9 Servicing Update
- March 05, 2019 -- Visual Studio 2017 version 15.9.8 Servicing Update
- February 12, 2019 -- Visual Studio 2017 version 15.9.7 Servicing Update
- January 24, 2019 -- Visual Studio 2017 version 15.9.6 Servicing Update
- January 08, 2019 -- Visual Studio 2017 version 15.9.5 Servicing Update
- December 11, 2018 -- Visual Studio 2017 version 15.9.4 Servicing Update
- November 28, 2018 -- Visual Studio 2017 version 15.9.3 Servicing Update
- November 19, 2018 -- Visual Studio 2017 version 15.9.2 Servicing Update
- November 15, 2018 -- Visual Studio 2017 version 15.9.1 Servicing Update
- November 13, 2018 -- Visual Studio 2017 version 15.9 Minor Release

- Каждое обновление может включать изменения в компиляторе C++
- Гарантируется полная совместимость по ABI
- Гарантируется обратная совместимость линкера

Хотим воспроизводимости!

VS2017 настойчиво предлагает обновляться

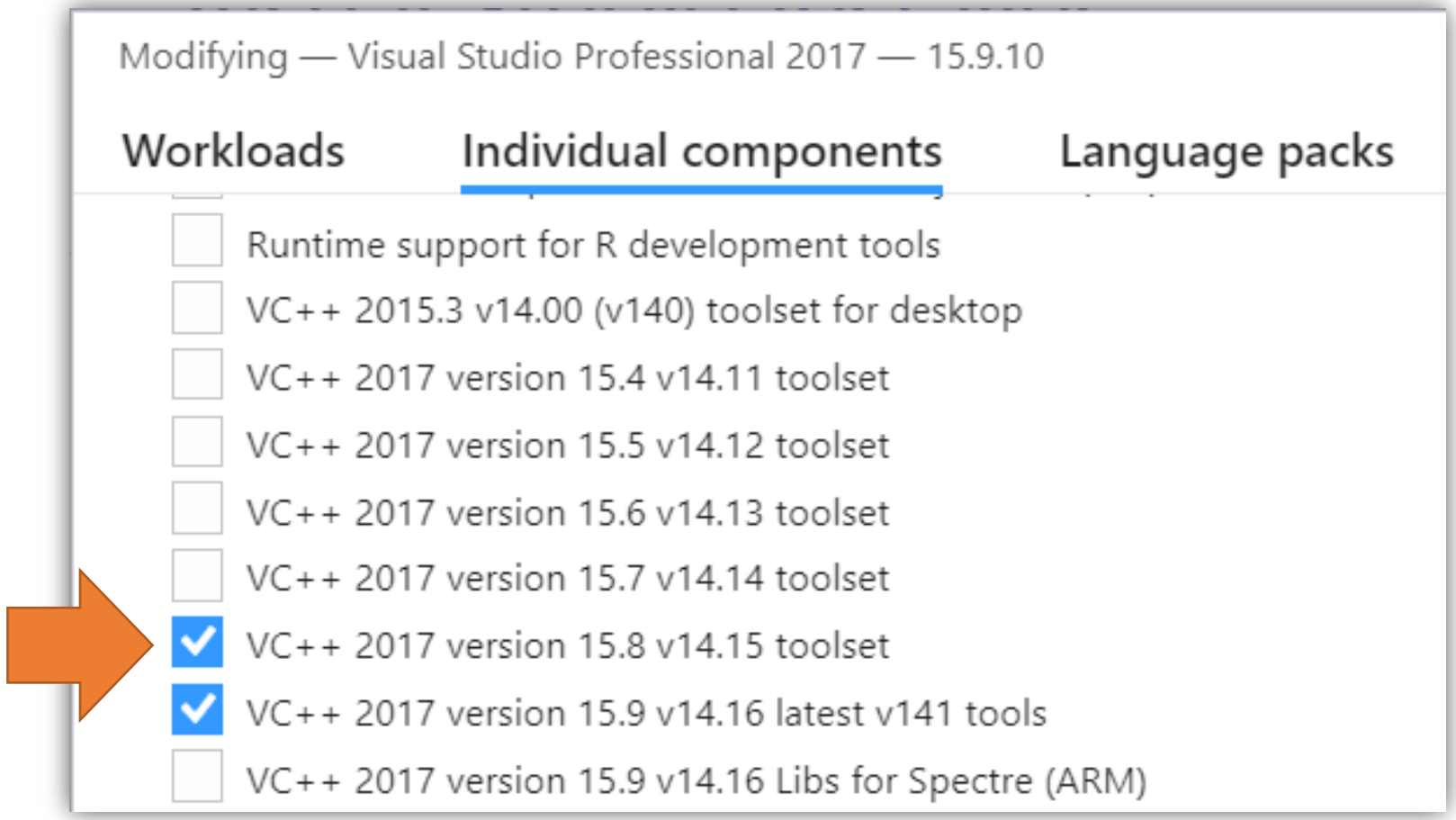
https://starecat.com/unattended-laptops-will-be-upgraded-to-windows-10/

Как делать не надо

- Create an offline installation of Visual Studio (опция --layout)
- Проблема: нельзя обновить установленную веб версию до этого спец. дистрибутива. Следовательно, потеряются настройки VS.

А вот так в самый раз

[Side-by-side minor version MSVC toolsets in Visual Studio 2017]



Задаём тулсет:

```
<PropertyGroup>
  <!-- Toolset version for VS2017 v15.8.9 -->
  <VCToolsVersion Condition = "'$(VCToolsVersion)' == '" >
    14.15.26726
  </VCToolsVersion>
  <VCToolsRedistVersion Condition = "'$(VCToolsRedistVersion)' == '" >
    14.15.26706
  </VCToolsRedistVersion>
</PropertyGroup>
```

Проверяем тулсет:

```
<Target Name="CheckToolsetVersionSpecified" BeforeTargets="Build">
  <Exec Command="cl.exe -Bv" ConsoleToMsBuild="true" IgnoreExitCode="true"
        IgnoreStandardErrorWarningFormat="true" StandardErrorImportance="low">
    <Output TaskParameter="ConsoleOutput" PropertyName="ClExeBvOutput" />
  </Exec>
  <Error Condition="!$(ClExeBvOutput.Contains('19.15.26731.0'))"
         Text="The solution must use cl.exe version 19.15.26731.0!
               See https://wiki.aligntech.com/x/WezaBg" />
</Target>
```

Проблемы в C++ коде

Шаблоны в Visual Studio

```
template<typename T>
struct Base {
  void f() { std::cout << "Base" << std::endl; }
};

void f() { std::cout << "Global" << std::endl; }

template<typename T>
struct Derived : Base<T> {
  void g() { f(); }
};

int main() {
  Derived<int>().g();  // "Global" in GCC, "Base" in Visual Studio
  // Also "Global" in Visual Studio if using /permissive-
}
```

Специфика Windows и Visual Studio 2013

- Бэкслеши “\” и НеВеРнЫй РеГиСтР файлов в `#include` директивах
- Нестандартные преобразования (`char* str = “literal”;`)
- Ожидание `wchar_t` от `boost::filesystem::path::native()`
- Пропущенные `typename`
- Операторы `and`, `or`, и т.д.

Возможности, убранные в C++17

- register keyword
  - Просто убрать
- std::auto_ptr
  - Использовался только в сторонних бибилотеках, помогло обновить версию
- std::unary_function, std::binary_function
  - Либо переписать, либо в Windows-only коде добавить _HAS_AUTO_PTR_ETC

Что не было поймано на этапе компиляции

- `noexcept` деструкторы
- Размер `long`
  - Убрать `long` где возможно
  - Пометить перегрузки используемых извне функций как `=delete`
- Размер `wchar_t`
  - Убрать `wstring` где возможно
  - Форсировать сериализацию `wstring` как UTF-16

Всевозможные случаи UB

```
struct RigidTransform {
  Vector3F d_translation;
  Quaternion d_rotation;
};

RigidTransform Tooth::getBasisTransform() const;

static void restoreZMovement(Mold_NS::Tooth& i_tooth)
{
  const Quaternion& basisRotation = i_tooth.getBasisTransform().d_rotation;
  // Use basisRotation
}
```

```
std::lock_guard<std::mutex> locker(mutex);
threads.push_back(std::move(thread));
threadHandles.push_back(thread.native_handle());
```

std::type_info

```
std::cout << typeid(int).name() << std::endl;
std::cout << typeid(std::vector<int>).name() << std::endl;
std::cout << typeid(Mold_NS::Tooth).name() << std::endl;
```

Visual Studio:

```
int
class std::vector<int,class std::allocator<int> >
class Mold_NS::Tooth
```

GCC:

```
i
St6vectorIiSaIiEE
N7Mold_NS5ToothE
```

clang:

```
i
NSt3__16vectorIiNS_9allocatorIiEEEE
N7Mold_NS5ToothE
```

Решение: использовать boost::core::demangle()

Спасибо!

Михаил Матросов, Expert Software Engineer, Align Technology

Александр Воронков, Senior Software Developer, Align technology

Ссылки на исходники в PDB

- AlignConanBuildTools формирует JSON метаданных: список исходников, адрес репозитория, ревизия.
- Сборочный скрипт:
  - Загружает JSON с метаданными
  - Вызывает srctool и получает список исходников в PDB
  - Готовит файл соответствия списка исходников PDB и http адресов для загрузки
  - Добавляет файл соответствия в PDB с помощью pdbstr

Основные цели

- Воспроизводимость сборки наших продуктов
- Упрощение работы со сторонними библиотеками

О чём мы не рассказали

- Магический вызов “conan install” в процессе сборки солюшна
- Автоматическое определение списка сторонних библиотек, использованных в солюшне
- VisualStudioSplitGen
- Скрытие символов в .SO