Почему vcpkg не Conan

ALEXANDER VORONKOV, EXPERT SOFTWARE DEVELOPER
ALIGN TECHNOLOGY INC.

Conan

- Первый кроссплатформенный пакетный менеджер для C++
- Полностью написан на Python
- Язык описания пакетов - Python

vcpkg

- Изначально поддерживал только Windows
- Ядро написано на C++, остальной код – на CMake
- Язык описания пакетов - CMake + JSON

Установка

Установка Conan

- Вариант 1 (требуется Python)
  - pip install conan
- Вариант 2 (PyInstaller)
  - Windows Installer
  - Ubuntu/Debian DEB
- Вариант 3
  - macOS: brew
  - Arch Linux: yay –S conan
- Вариант 4: Собрать из исходников

Установка vcpkg (Windows)

- git clone https://github.com/microsoft/vcpkg
- .\vcpkg\bootstrap-vcpkg.bat
- .\vcpkg\vcpkg integrate install

Установка vcpkg (Linux)

- git clone https://github.com/microsoft/vcpkg
- ./vcpkg/bootstrap-vcpkg.sh
- Установка пакетов
  - apt install curl zip unzip tar pkg-config
  - apk add build-base cmake ninja zip unzip curl pkgconfig
- Alpine: export VCPKG_FORCE_SYSTEM_BINARIES=1
- ./vcpkg/bootstrap-vcpkg.sh

Установка vcpkg (macOS)

- git clone https://github.com/microsoft/vcpkg
- ./vcpkg/bootstrap-vcpkg.sh
- brew install pkg-config

Big Brother is watching you

Отключение телеметрии

- .\vcpkg\bootstrap-vcpkg.bat –disableMetrics
- ./vcpkg/bootstrap-vcpkg.sh –disableMetrics

Использование

Conan базовые понятия

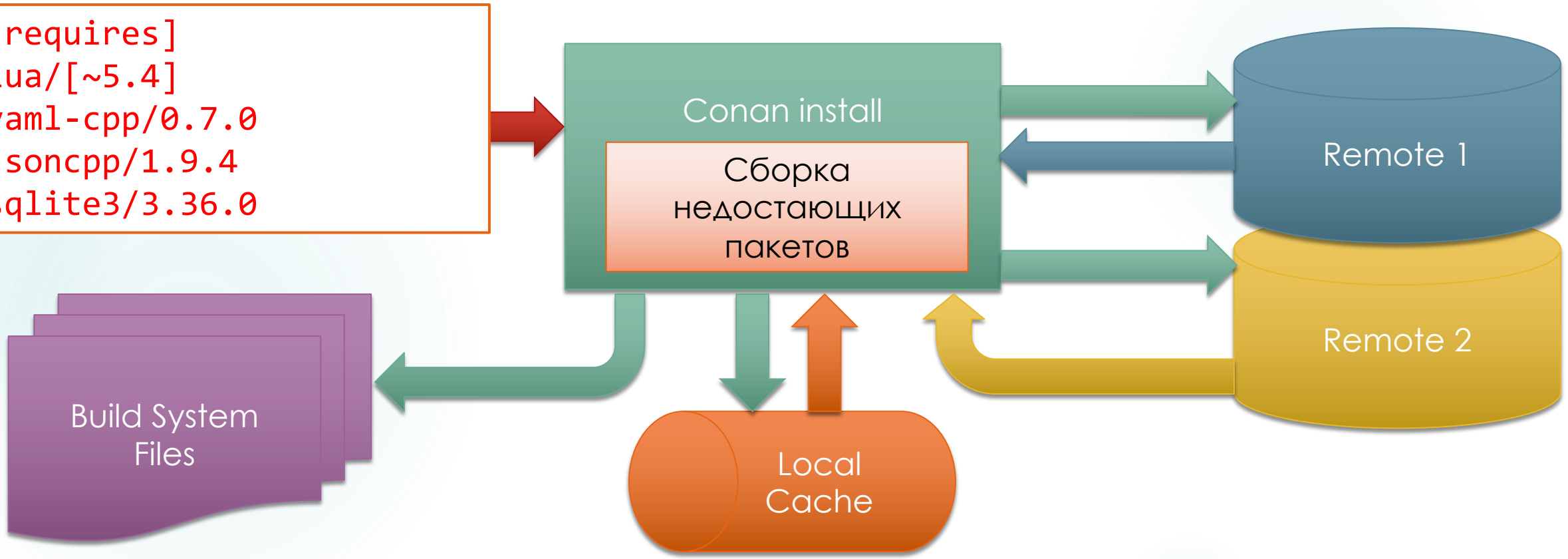
- Recipe (conanfile.py): описывает процесс сборки, формир
  пакетов и их использования
- Reference:
  - boost/1.69.0@conan/stable
  - boost/1.77.0
  - boost/1.77.0@_/_
- Configuration:
  - Settings: Доступный набор описывается Settings.yml
  - Options: Доступный набор описывается conanfile.py
  - Набор конкретных значений формирует ID пакета
- Profile – Файл содержащий конкретные значения Settings и Options

```
[settings]
os=Linux
os_build=Linux
arch=x86_64
arch_build=x86_64
compiler=gcc
compiler.version=7.3
compiler.libcxx=libstdc++
build_type=Release
[options]
[build_requires]
[env]
```

```
conan profile new default --detect --force
conan profile update settings.compiler.libcxx=libstdc++11 default
```

Как работает Conan


```
[requires]
lua/[~5.4]
yaml-cpp/0.7.0
jsoncpp/1.9.4
sqlite3/3.36.0
```



vcpkg базовые понятия

- Portfile (portfile.cmake): описывает процесс сборки пакетов
- Manifest (vcpkg.json):
  - Описывает пакет, его опции и зависимости
  - Описывает необходимые пакеты для проекта и их опции
  - Пришел на смену файлу CONTROL
- Triplet: CMake скрипт, декларирующий название платформы, архитектуру, метод сборки библиотек (static/dynamic)
- Port: папка, содержащая Portfile, Manifest, и все необходимые файлы для сборки пакета (патчи и т.п.), именуется по названию библиотеки, которую собирает
- Registry – папка, содержащая подпапки, являющиеся Port

Как работает vcpkg Manifest mode


```
{
  "$schema":
"https://raw.githubusercontent.com/microsoft/vcpkg/master/scripts/vcpkg.schema.json",
  "name": "my-application",
  "version": "0.1.0",
  "dependencies": [
    "lua",
    "yaml-cpp",
    "jsoncpp",
    "sqlite3"
  ]
}
```

Cache

Основное различие в подходах

Conan может поставлять артефакты не только для C/C++

vcpkg ориентирован на сборку из исходников

Интеграция

Conan интеграция со сборочными системами

- Используется механизм «Генераторов»
- Генераторы могут быть написаны самостоятельно
- Кастомные генераторы можно использовать как пакеты Conan
- Практически для всех сборочных систем требуется ручной запуск conan install
- Для CMake есть модуль, автоматически запускающий Conan

```
if(NOT EXISTS "${CMAKE_BINARY_DIR}/conan.cmake")
   message(STATUS "Downloading conan.cmake from https://github.com/conan-io/cmake-conan")
   file(DOWNLOAD "https://raw.githubusercontent.com/conan-io/cmake-conan/master/conan.cmake"
                 "${CMAKE_BINARY_DIR}/conan.cmake")
endif()
```

Доступные генераторы

- CMake
- MSBuild
- pkg-config
- Xcode
- b2
- qbs
- qmake
- scons
- premake
- make
- virtualbuildenv
- txt
- json
- youcompleteme

Conan интеграция с IDE

- Visual Studio есть плагин
- CLion есть плагин
- Xcode – или CMake, или генератор + запуск conan install
- Любая IDE, использующая CMake, интегрируется модулем https://github.com/conan-io/cmake-conan
- Vim + YCM есть генератор

vcpkg интеграция со сборочными системами

- CMake через Toolchain файл
- MSBuild только на Windows, и Visual Studio новее 2015 Upd. 3

```
C:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\Common7\IDE\VC\VCTargets\Microsoft.Cpp.props
```

```
<PropertyGroup >
    <VCLibPackagePath Condition="'$(VCLibPackagePath)' == ''">$(LOCALAPPDATA)\vcpkg\vcpkg.user</VCLibPackagePath>
</PropertyGroup>

<Import Condition="'$(VCLibPackagePath)' != '' and Exists('$(VCLibPackagePath).props')" Project="$(VCLibPackagePath).props" />
```

vcpkg интеграция с IDE

- Visual Studio интеграция из коробки
- Всё остальное – через CMake

```
cmake -DCMAKE_TOOLCHAIN_FILE=${HOME}/vcpkg/scripts/buildsystems/vcpkg.cmake ~/src

cmake --toolchain ${HOME}/vcpkg/scripts/buildsystems/vcpkg.cmake ~/src
```

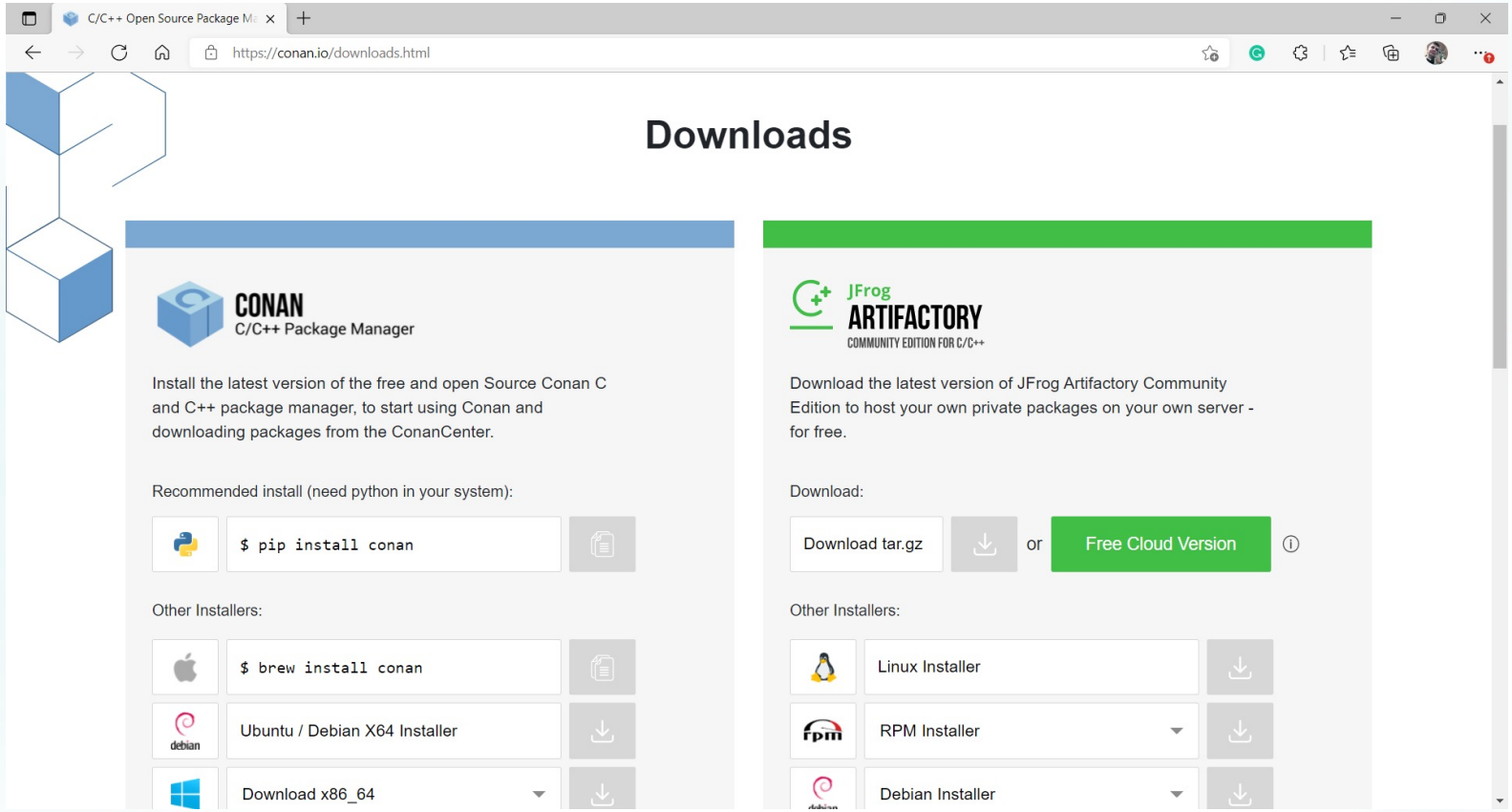
Откуда берутся пакеты

И КУДА ДЕВАЮТСЯ

Conan center

- Официальный дефолтный репозиторий
- Появляется в настройках Conan при первом обращении
- Рецепты разрабатываются в репозитории на GitHub https://github.com/conan-io/conan-center-index
- Пользователь Conan не взаимодействует с Git репозиторием
- Для Conan источником истины являются только рецепты находящиеся в Conan репозитории

Conan + Artifactory



Artifactory

- Ваши собственные Conan репозитории для ваших пакетов
- Рецепт и пакеты загружаются командой conan upload
- Полная независимость от Conan Center
- Кеширующий прокси Conan Center
- Есть Cloud версия

Сборка пакетов средствами CI

- https://github.com/conan-io/conan-package-tools
  - Travis CI
  - Appveyor
  - Gitlab
  - Circle CI
  - *Bamboo ???*
- Jenkins CI – поддерживает Conan, и есть Artifactory плагин
- Azure DevOps – JFrog Artifactory Extension

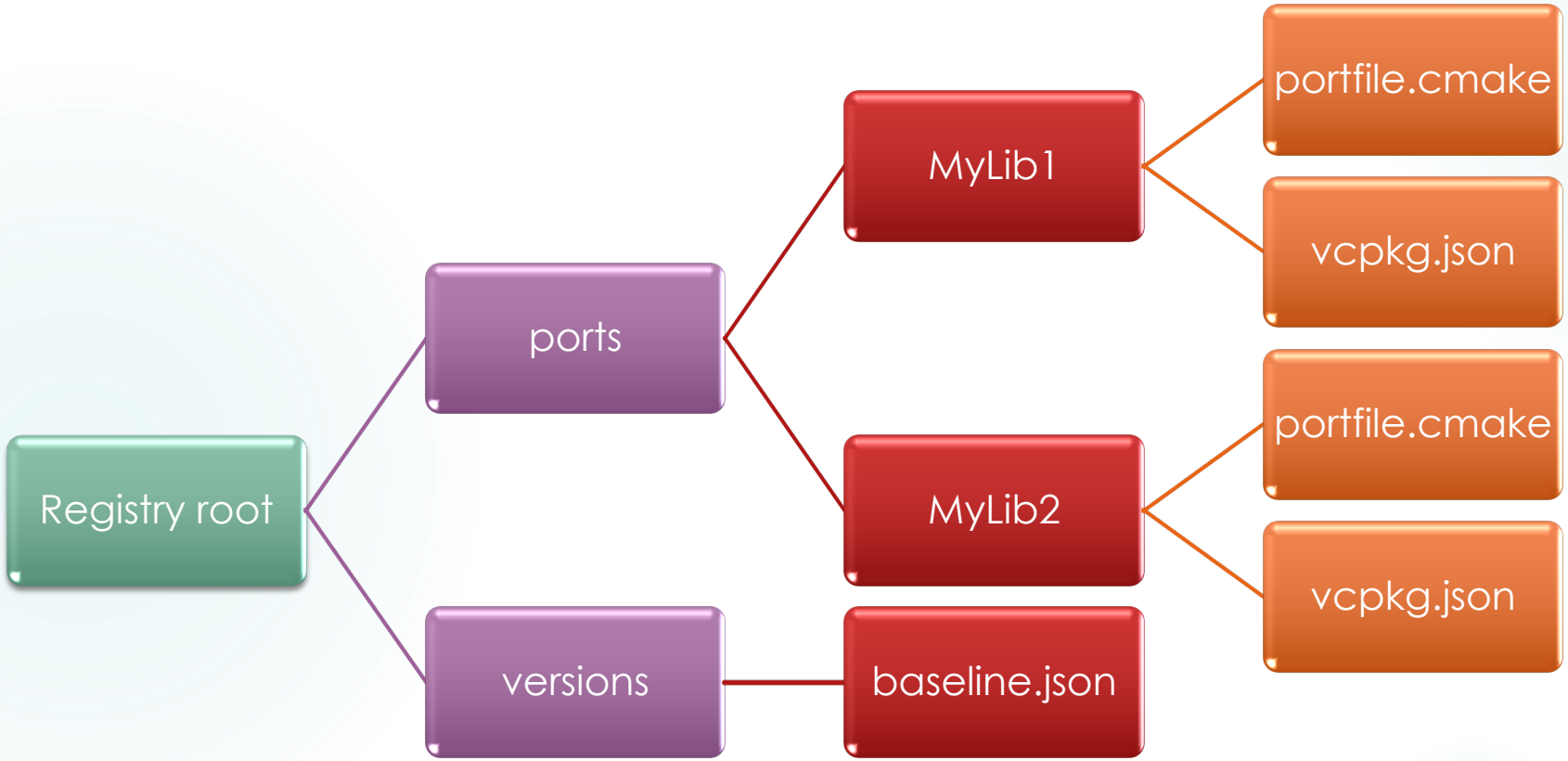
vcpkg default registry

- Реестр портов является частью Git репозитория vcpkg
- Подпапки Ports и Versions

Пользовательские порты vcpkg

- Overlay-ports - папки с портами, требуется указывать при каждом вызове или выставить VCPKG_OVERLAY_PORTS
- Форк репозитория vcpkg
- Пользовательский registry - имеет структуру аналогичную дефолтному
  - Для использования пользовательского registry в проекте требуется указать его в *vcpkg-configuration.json*
  - Дефолтный также должен быть указан в *vcpkg-configuration.json*

Структура Registry



Собранные бинари в vcpkg

- Складываются в Binary Cache
- Nuget репозиторий может быть подключен как бэкенд кеша
- Экспериментально поддерживается Azure Blob и Google Cloud
- Переменная VCPKG_BINARY_SOURCES управляет кешами
- Также естьvcpkg export

Сборка vcpkg пакетов CI

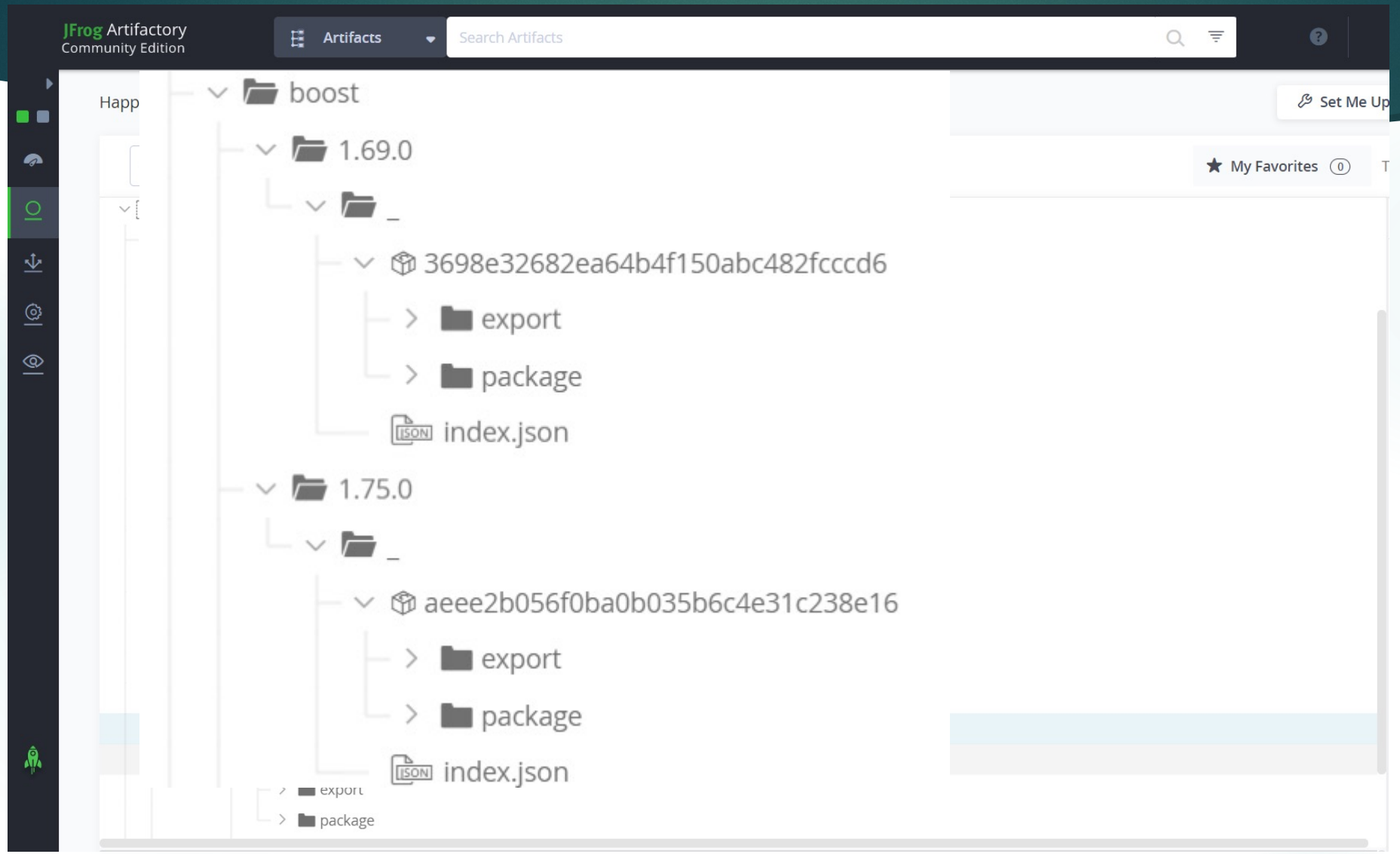
- Настраивается Nuget репозиторий как хранилище binary cache
- Доступ к Nuget репозиторию настраивается на CI агентах
- Для Linux/macOS необходим Mono
- CI собирает все нужные триплеты и они кешируются в Nuget
- PROFIT

Версионирование

Версионирование в Conan

- Версия содержится в Recipe и является частью Reference
- Каждый компонент Reference является отдельным элементом пути в репозитории
- В локальном кеше, - аналогично
- Никаких ограничений на используемую версию Recipe нет, за исключением зависимостей
- При различии версий в зависимостях разных Reference вычисляется совместимая версия по semantic versioning

Структура версий в репозитории



Версионирование в vcpkg

- Baseline – набор версий портов на определенный момент
- baseline.json – список версий
- Файлы в подпапках папки versions – соответствия версий и Git ревизий репозитория с реестром портов
- Версионирование несовместимо с overlay ports

Версионирование в vcpkg


```
…
"abseil": {
      "baseline": "2021-03-24",
      "port-version": 1
    }, {
…         "versions": [
"dependencies": [
    { "name": "asio", "version>=": "1.20.0"},                    982e81ef131a6d1d",
    "abseil",
    "jsoncpp"
],
"builtin-baseline": "153fcbcbc2e90f097eb4336669f0083320cfd504",
"overrides": [
  { "name": "jsoncpp", "version-string": "1.8.1"}                 e41f59aace59c88",
]
…             "port-version": 0
            },
        …
```

Ньюансы

Статические и динамические библиотеки

- Для Conan нет никаких настроек, но есть соглашение об опции рецепта shared
- Для vcpkg это один из параметров триплета
  - Для Windows по умолчанию используется триплет с динамическими библиотеками
  - Для Linux/macOS используются статические библиотеки
  - Для Linux из коробки нет триплета, поддерживающего динамическую сборку.
    Его создание оставлено в качестве упражнения для пользователя.
    https://vcpkg.io/en/docs/examples/overlay-triplets-linux-dynamic.html

Conan и нестандартные Linux дистрибутивы

- MUSL-based (Alpine)
- libstdc++ системная динамическая + статическая из devtoolset (RHEL, AmazonLinux)
- Решается вводом дополнительного sub-setting в Settings.yml

```
os:
    Windows:
        subsystem: [None, cygwin, msys, msys2, wsl]
    Linux:
        distro: [None, RHEL6, RHEL7, Amazon, Amazon2, Alpine]
```

Выводы

- vcpkg имеет меньший порог вхождения но меньшие возможности, хорошо интегрируется с Visual Studio. Нет бинарных репозиториев
- Conan имеет более широкие возможности, но требует изучения. Имеет бинарный репозиторий.

Спасибо!

ALEXANDER VORONKOV, EXPERT SOFTWARE DEVELOPER
ALIGN TECHNOLOGY INC.

align | invisalign® iTero®