Packaging C/C++ dependencies with Conan
First things first

@theodelrieu

Tanker.io

Conan contribution
First things first

@theodelrieu

Tanker.io

Conan contribution
You said Tanker?

End-to-End encryption SDK

Available in Javascript

Soon™ on Android/iOS
“Starting a new project is awesome!”

“We can have complete control of our stack!”

“Hum... which language shall we use?”

Steve the Intern
Subject of Discord
Decisions, decisions

Our key conditions:

- Write once, run everywhere
- High performance
- Good dependency management
Decisions, decisions

Our key conditions:

- Write once, run everywhere
- High performance
- Good dependency management
Wait, w00t!?

“‘C++’ and ‘good dependency management’ in the same slide...”
Wait, w00t!?

“‘C++’ and ‘good dependency management’ in the same slide...”

Usual C++ dependencies:
Behold Conan!

Decentralized package manager

Open Source

Python

Made by Tanker.io team
Conan 101: Package Management

Using Conan to cross-build for Android
Conan 101: Creating packages

$ conan create

Run recipe

Store package locally
Conan 101: Creating packages

$ conan create

Run recipe

Store package locally

Find local package

Upload to server

$ conan upload
from conans import ConanFile

class ArithmeticConan(ConanFile):
    name = "arithmetic"
    version = "0.1"
from conans import ConanFile

class ArithmeticConan(ConanFile):
    name = "arithmetic"
    version = "0.1"
    settings = "os", "arch", "build_type", "compiler"
from conans import ConanFile

class ArithmeticConan(ConanFile):
    name = "arithmetic"
    version = "0.1"
    settings = "os", "arch", "build_type", "compiler"

    def source(self):
        url = "https://github.com/theodelrieu/FOSDEM2018-arithmetic"
        self.run("git clone %s arithmetic" % url)
from conans import ConanFile, CMake

class ArithmeticConan(ConanFile):
    def build(self):
        cmake = CMake(self)
        cmake.configure(source_dir="arithmetic")
        cmake.build()
        cmake.install()
from conans import ConanFile, CMake

class ArithmeticConan(ConanFile):
    def build(self):
        cmake = CMake(self)
        cmake.configure(source_dir="arithmetic")
        cmake.build()
        cmake.install()

    def package_info(self):
        self.cpp_info.libs = ["arithmetic"]
Conan 101: Consuming packages

$ conan install

- Check requirements
- Fetch packages
- Generate build info
Conan 101: Consuming packages

conanfile.txt

[requires]
arithmetic/0.1@theo/stable
Conan 101: Consuming packages

conanfile.txt

[requires]
arithmetic/0.1@theo/stable

[generators]
cmake
Conan 101: Consuming packages

CMakeLists.txt (before Conan)

cmake_minimum_required(VERSION 3.0)
project(Calculator)

find_package(Arithmetic)
add_executable(calculator src/main.cpp)
target_link_libraries(calculator Arithmetic::Arithmetic)
CMakeLists.txt (after Conan)

cmake_minimum_required(VERSION 3.0)
project(Calculator)
include(${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
conan_basic_setup()
find_package(Arithmetic)
add_executable(calculator src/main.cpp)
target_link_libraries(calculator Arithmetic::Arithmetic)
CMakeLists.txt (after Conan, without find_package)

```
cmake_minimum_required(VERSION 3.0)
project(Calculator)
include(${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
conan_basic_setup(TARGETS)

add_executable(calculator src/main.cpp)
target_link_libraries(calculator CONAN_PKG::arithmetic)
```
DEMO
Conan 101: Change settings

// using older GCC version
$ conan create . theo/stable -s compiler.version=6

// clang 5, new GCC ABI, Debug build. It gets hairy...
$ conan create . theo/stable -s compiler=clang -s build_type=Debug
   -s compiler.version=5.0 -s compiler.libcxx=libstdc++11
Conan 101: Profiles

Profiles are the solution!

```ini
# Generated by default (on my machine)
[settings]
os=Linux
arch=x86_64
compiler=gcc
compiler.version=7
compiler.libcxx=libstdc++  # New ABI: libstdc++11
build_type=Release
```
New profile: clang5-debug

```yaml
[settings]
os = Linux
arch = x86_64
compiler = clang
compiler.version = 5.0
compiler.libcxx = libstdc++11
build_type = Debug
```
Conan 101: Profiles

$ conan create . theo/stable --profile clang5-debug

Manually specifying settings is still possible

$ conan create . theo/stable -pr clang5-debug -s ... -s ...
Conan 101: Package Management

Using Conan to cross-build for Android
Conan & Android

Prerequisites:

- Android NDK
- Standalone Android Toolchain
- New Conan Profile
Conan & Android: Build requirements

RECIPE

self.env_info.FOO="bar"
Conan & Android: Build requirements

**RECIPE**

```
self.env_info.FOO="bar"
```
Conan & Android: Build requirements

**RECIPE**

```python
self.env_info.FOO = "bar"
```

**ENV**

```bash
export FOO = "bar"
```

---

Made by Tanker.io team
Conan & Android: Build requirements

**RECIPE**

```
self.env_info.FOO="bar"
```

**ENV**

```
export FOO="bar"
```

**RECIPE**

```
self.cpp_info.sysroot=/bar
```

**RECIPE**

```
self.deps_cpp_info.sysroot=/bar
```
Conan & Android: Setting up the Toolchain

- Android NDK
- build_requires
- Standalone Android Toolchain
Conan & Android: Setting up the Toolchain

- Android NDK
- build.requires
- Standalone Android Toolchain
- build.requires
- Arithmetic
[settings]
os=Android
arch=armv8
os.api_level=21
compiler=clang
compiler.version=5
compiler.libcxx=libc++
build_type=Release
Android profile

[settings]
os=Android
arch=armv8
os.api_level=21
compiler=clang
compiler.version=5
compiler.libcxx=libc++
build_type=Release
os_build=Linux
arch_build=x86_64
```python
from conans import ConanFile, tools
from os import path, unlink

class AndroidNDKConan(ConanFile):
    name = "android-ndk"
    version = "r16"
```
```python
from conans import ConanFile, tools
from os import path, unlink

class AndroidNDKConan(ConanFile):
    name = "android-ndk"
    version = "r16"
    settings = "os_build", "arch_build"
```
from conans import ConanFile, tools
from os import path, unlink

class AndroidNDKConan(ConanFile):
    name = "android-ndk"
    version = "r16"
    settings = "os_build", "arch_build"

    def source(self):
        tools.download(url, NDK_URL)
        tools.unzip("ndk.zip", keep_permissions=True)
        os.unlink("ndk.zip")
from conans import ConanFile, tools
from os import path, unlink

class AndroidNDKConan(ConanFile):
    def package(self):
        self.copy("*", src="android-ndk-r16")
from conans import ConanFile, tools
from os import path, unlink

class AndroidNDKConan(ConanFile):
    def package(self):
        self.copy("*", src="android-ndk-r16")

    def package_info(self):
        tools_folder = path.join(self.package_folder, "build/tools")
        self.env_info.PATH.append(tools_folder)
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    name = "android-toolchain"
    version = "r16"
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    name = "android-toolchain"
    version = "r16"
    settings = "os_build", "arch_build"
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    name = "android-toolchain"
    version = "r16"
    settings = "os_build", "arch_build"
    build_requires = "android-ndk/r16@theo/stable"
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    name = "android-toolchain"
    version = "r16"
    settings = "os_build", "arch_build"
    build_requires = "android-ndk/r16@theo/stable"

def build(self):
    command = "make-standalone-toolchain.sh %s"
    self.run(command % MAKE_TOOLCHAIN_ARGS)
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    def package_info(self):
        sysroot_folder = path.join(self.package_folder, "sysroot")
        self.cpp_info.sysroot = sysroot_folder
Android profile

[settings]
os=Android
arch=armv8
# etc, etc...

[build_requires]
android-toolchain/r16@theo/stable
Conan & Android

FINAL DEMO
Thank you!

tanker.io

tanker.io/docs

github.com/supertanker/

c Conan.io

github.com/conan-io

CPPlang #conan

© 2018 Kontrol SAS