



Packaging C/C++ dependencies with Conan



First things first



@theodelrieu



Tanker.io



Conan
contributor





First things first



@theodelrieu



Tanker.io



theodelrieu

1 commit 2 ++ 2 --

Conan
contributor

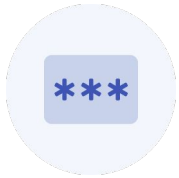




You said Tanker?



Tanker



End-to-End
encryption SDK



Available in
Javascript



Soon(™) on
Android/iOS





A fresh start

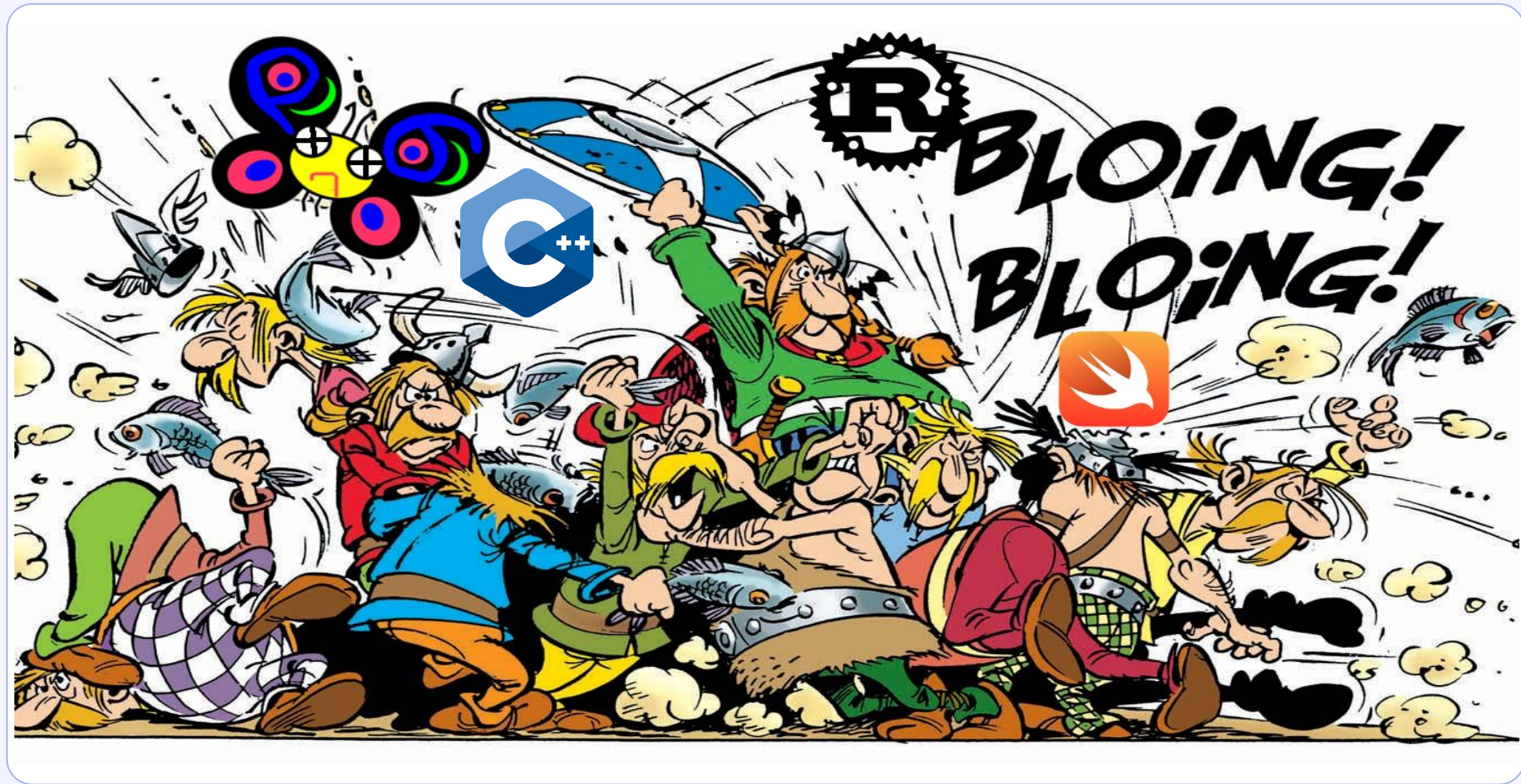
“Starting a new project is awesome!”

“We can have complete control of our stack!”

“Hum... which language shall we use?”

Steve the Intern

⚙️ A subject of discord





Decisions, decision

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?!

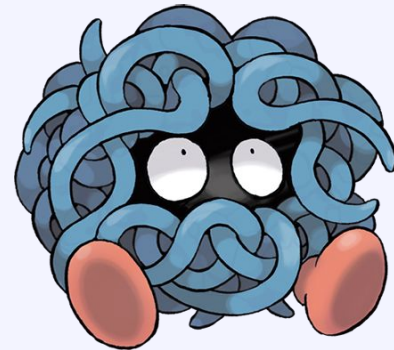
'C++' and 'good dependency management'
in the same slide...



Wait, w00t?!

'C++' and 'good dependency management'
in the same slide...

Usual C++ dependencies:





Behold, Conan!



Decentralized
package manager



Open Source



Python



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

\$ conan upload



Find local
package



Upload to
server



Conan 101: First recipe

```
from conans import ConanFile

class ArithmeticConan(ConanFile):
    name = "arithmetic"
    version = "0.1"
```



Conan 101: First recipe

```
from conans import ConanFile

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




Conan 101: First recipe

```
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)
```



Conan 101: First recipe

```
from conans import ConanFile

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



Conan 101: First recipe

```
from conans import ConanFile

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
```



Run recipe



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)
```

Conan 101: Consuming packages

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)
```


Conan 101: Consuming packages

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 a solution

```
# 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
```



Conan 101: Profiles

New profile: clang5-debug

```
[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 ...
```



Conan 101: Package Management

Using Conan to cross-build for Android



Prerequisites:

Android NDK

Standalone
Android Toolchain

New
Conan Profile



Conan & Android: Build requirements



PackageA

RECIPE

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



Conan & Android: Build requirements

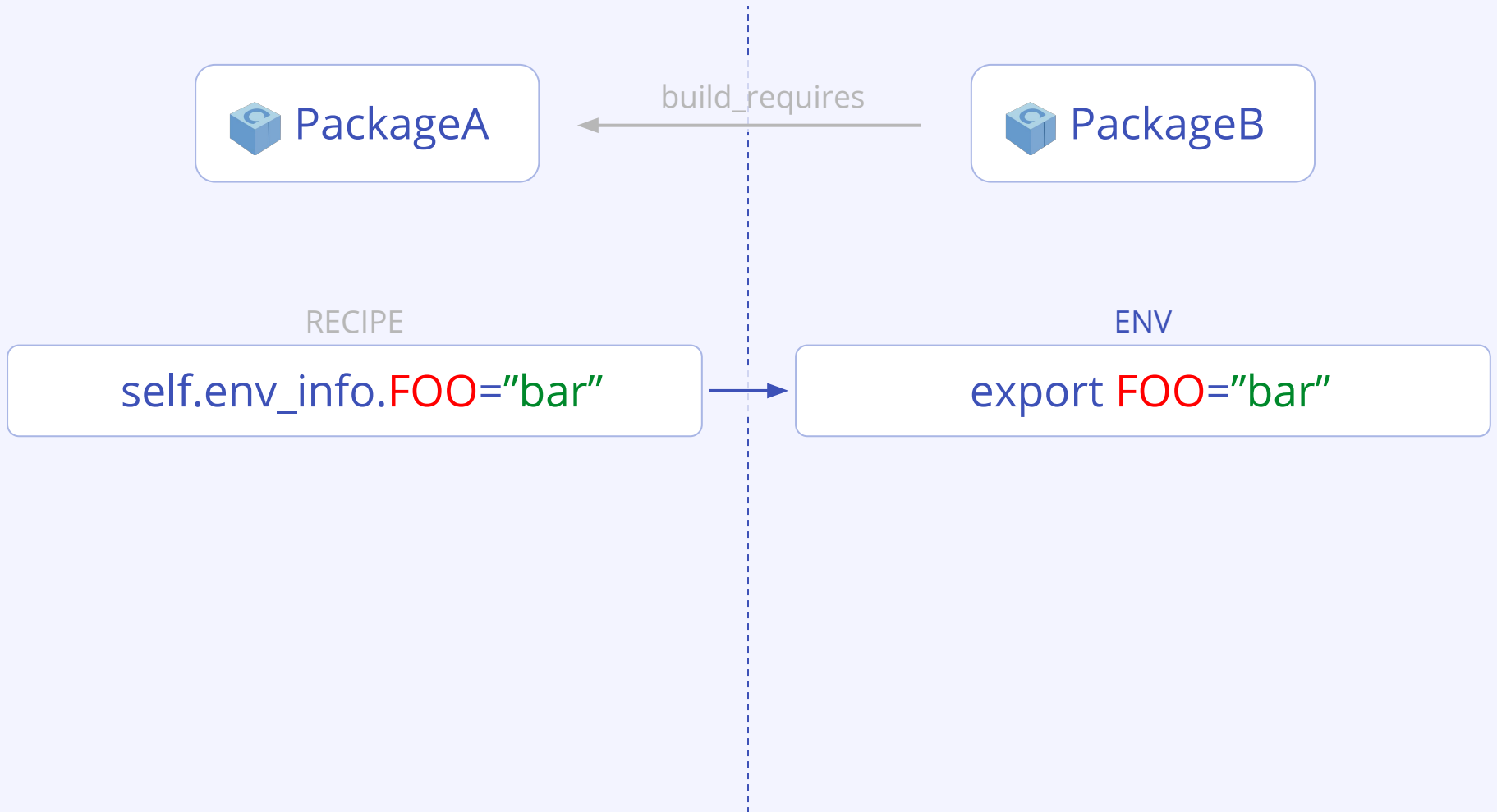


RECIPE

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

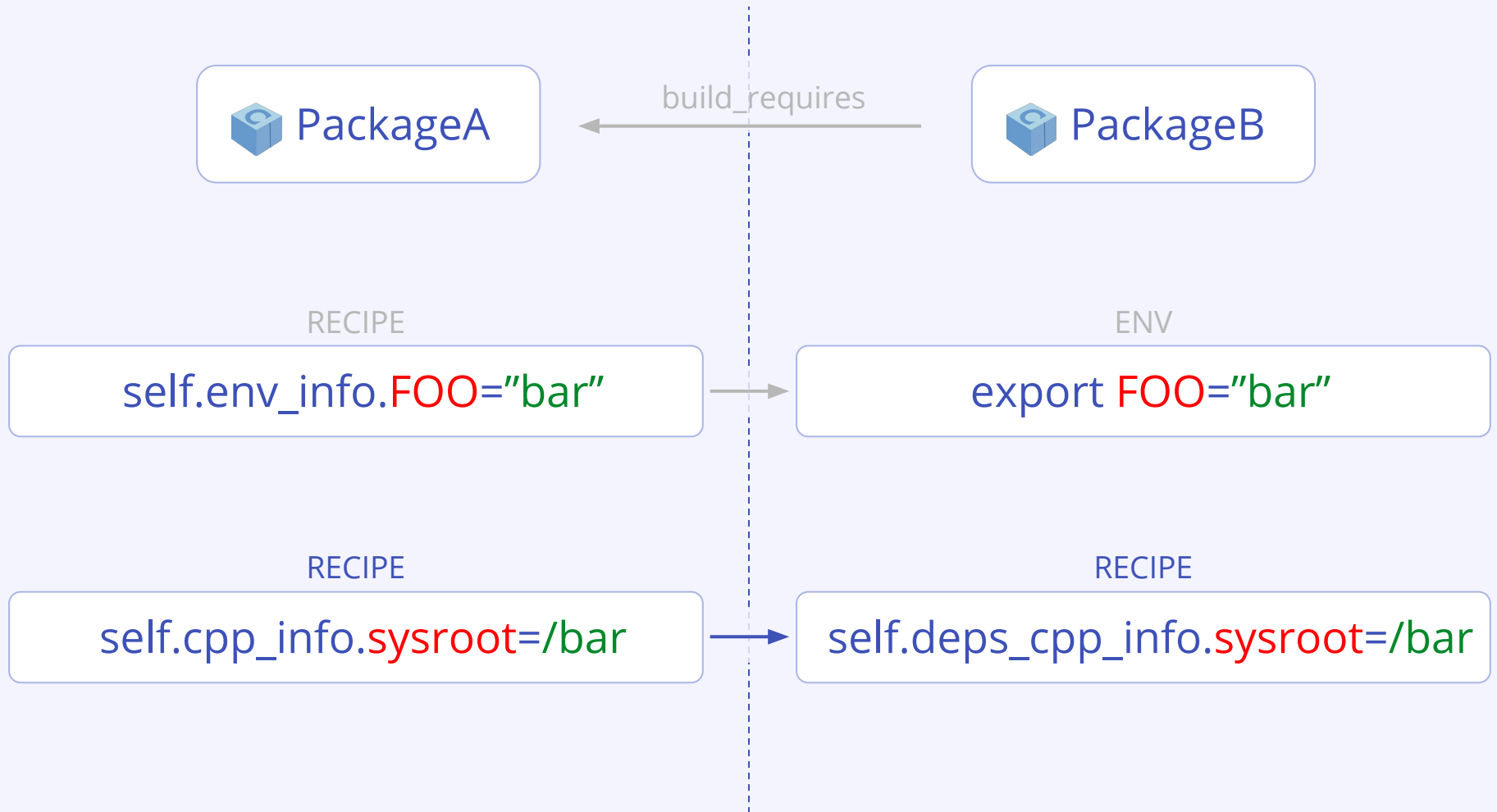


Conan & Android: Build requirements





Conan & Android: Build requirements



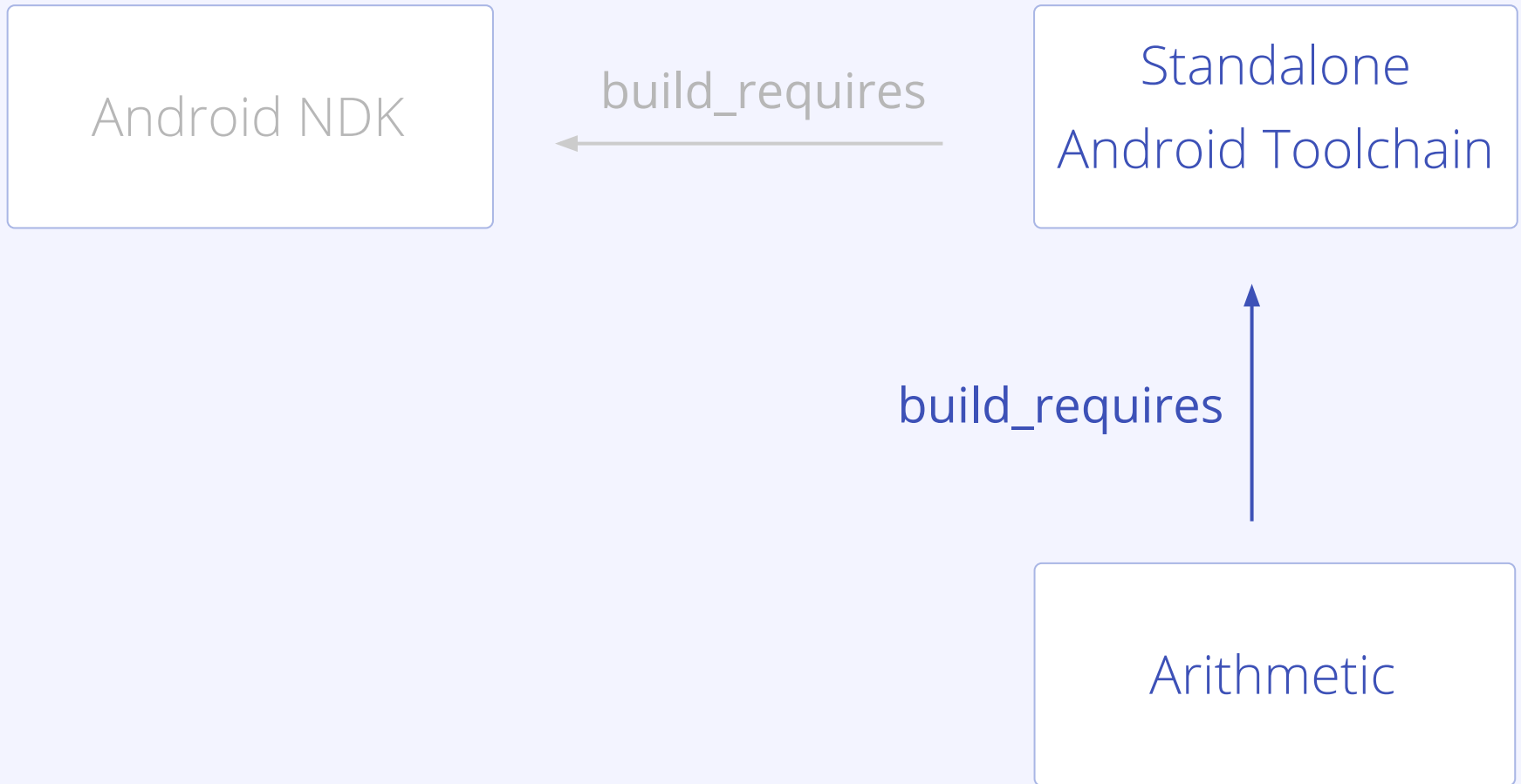


Conan & Android: Setting up the Toolchain





Conan & Android: Setting up the Toolchain





Android profile

```
[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
```




Conan & Android: NDK recipe

```
from conans import ConanFile, tools
from os import path, unlink

class AndroidNDKConan(ConanFile):
    name = "android-ndk"
    version = "r16"
```



Conan & Android: NDK recipe

```
from conans import ConanFile, tools
from os import path, unlink

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



Conan & Android: NDK recipe

```
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")
```



Conan & Android: NDK recipe

```
from conans import ConanFile, tools
from os import path, unlink

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



Conan & Android: NDK recipe

```
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)
```



Conan & Android: NDK recipe

```
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    name = "android-toolchain"
    version = "r16"
```



Conan & Android: NDK recipe

```
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    name = "android-toolchain"
    version = "r16"
    settings = "os_build", "arch_build"
```



Conan & Android: NDK recipe

```
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"
```




Conan & Android: NDK recipe

```
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)
```



Conan & Android: Standalone Toolchain recipe

```
from conans import ConanFile, tools
from os import path

class AndroidToolchainConan(ConanFile):
    def package_info(self):
        sys = path.join(self.package_folder, "sysroot")
        self.cpp_info.sysroot = sys
```



Android profile

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

[build_requires]
android-toolchain/r16@theo/stable
```



FINAL DEMO



tanker.io



Conan.io



tanker.io/docs



github.com/conan-io



github.com/supertanker/



[#conan](https://CPPlang.com)

Thank you!