Conan + CMake Hello World Lab (Steps 0 → 7)

# 0) Prereqs (once)

pip install -U conan  
conan profile detect --force  
cmake --version  
g++ --version # or clang/msvc

# 1) Project layout

hello-conan/  
 CMakeLists.txt  
 conanfile.txt  
 src/  
 main.cpp  
  
src/main.cpp  
#include <iostream>  
int main() {  
 std::cout << "Hello, Conan + CMake!\n";  
 return 0;  
}  
  
CMakeLists.txt  
cmake\_minimum\_required(VERSION 3.18)  
project(hello\_conan LANGUAGES CXX)  
set(CMAKE\_CXX\_STANDARD 17)  
add\_executable(hello src/main.cpp)  
  
conanfile.txt  
[generators]  
CMakeDeps  
CMakeToolchain  
[layout]  
cmake\_layout

# 2) First configure & build

rm -rf Release Debug  
conan install . --output-folder=. --build=missing -s build\_type=Release  
cmake -S . -B Release -DCMAKE\_TOOLCHAIN\_FILE=Release/generators/conan\_toolchain.cmake  
cmake --build Release  
./Release/hello

# 3) Add a real dependency (fmt)

conanfile.txt  
[requires]  
fmt/10.2.1  
[generators]  
CMakeDeps  
CMakeToolchain  
[layout]  
cmake\_layout  
  
CMakeLists.txt  
find\_package(fmt REQUIRED)  
add\_executable(hello src/main.cpp)  
target\_link\_libraries(hello PRIVATE fmt::fmt)  
  
src/main.cpp  
#include <fmt/core.h>  
int main() {  
 fmt::print("Hello, {} + {}! {}\n", "Conan", "CMake", "Now with fmt");  
 return 0;  
}  
  
# Commands  
conan install . --output-folder=. --build=missing -s build\_type=Release  
cmake -S . -B Release -DCMAKE\_TOOLCHAIN\_FILE=Release/generators/conan\_toolchain.cmake  
cmake --build Release  
./Release/hello

# 4) Switch build types & options

# Debug build  
conan install . --output-folder=. --build=missing -s build\_type=Debug  
cmake -S . -B Debug -DCMAKE\_TOOLCHAIN\_FILE=Debug/generators/conan\_toolchain.cmake  
cmake --build Debug  
./Debug/hello  
  
# Shared libs (if supported)  
conan install . --output-folder=. --build=missing -s build\_type=Release -o fmt/\*:shared=True  
cmake -S . -B Release -DCMAKE\_TOOLCHAIN\_FILE=Release/generators/conan\_toolchain.cmake  
cmake --build Release

# 5) Add a tiny test (CTest)

tests/smoketest.cpp  
#include <fmt/core.h>  
#include <cassert>  
int main() {  
 auto s = fmt::format("sum={}", 2 + 3);  
 assert(s.find("5") != std::string::npos);  
 return 0;  
}  
  
tests/CMakeLists.txt  
add\_executable(smoketest smoketest.cpp)  
target\_link\_libraries(smoketest PRIVATE fmt::fmt)  
add\_test(NAME smoketest COMMAND smoketest)  
  
Top-level CMakeLists.txt additions:  
enable\_testing()  
find\_package(fmt REQUIRED)  
add\_executable(hello src/main.cpp)  
target\_link\_libraries(hello PRIVATE fmt::fmt)  
add\_subdirectory(tests)  
  
# Commands  
conan install . --output-folder=. --build=missing -s build\_type=Release  
cmake -S . -B Release -DCMAKE\_TOOLCHAIN\_FILE=Release/generators/conan\_toolchain.cmake  
cmake --build Release  
ctest --test-dir Release --output-on-failure

# 6) Package your app with Conan

conanfile.py  
from conan import ConanFile  
from conan.tools.cmake import CMake, cmake\_layout  
  
class HelloAppConan(ConanFile):  
 name = "hello\_app"  
 version = "1.0.0"  
 settings = "os", "arch", "compiler", "build\_type"  
 requires = "fmt/10.2.1"  
 generators = "CMakeDeps", "CMakeToolchain"  
 package\_type = "application"  
  
 def layout(self):  
 cmake\_layout(self)  
 def build(self):  
 cm = CMake(self)  
 cm.configure()  
 cm.build()  
 def package(self):  
 self.copy("hello\*", dst="bin", src=self.build\_folder, keep\_path=False)  
 def package\_info(self):  
 self.cpp\_info.bindirs = ["bin"]  
  
# Build package  
conan create . --name=hello\_app --version=1.0.0 --build=missing -s build\_type=Release

# 7) Clean & iterate

rm -rf Release Debug  
conan cache path