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
```