Quickstart: Building with CMake

This tutorial aims to get you up and running with GoogleTest using CMake. If you're using GoogleTest for the first time or need a refresher, we recommend this tutorial as a starting point. If your project uses Bazel, see the Quickstart for Bazel instead.

Prerequisites

To complete this tutorial, you'll need:

- A compatible operating system (e.g. Linux, macOS, Windows).
- A compatible C++ compiler that supports at least C++14.
- CMake and a compatible build tool for building the project.
    - Compatible build tools include Make, Ninja, and others - see CMake Generators for more information.

See Supported Platforms for more information about platforms compatible with GoogleTest.

If you don't already have CMake installed, see the CMake installation guide.

{: .callout .note}
Note: The terminal commands in this tutorial show a Unix shell prompt, but the commands work on the Windows command line as well.

Set up a project

CMake uses a file named `CMakeLists.txt` to configure the build system for a project. You'll use this file to set up your project and declare a dependency on GoogleTest.

First, create a directory for your project:

```
$ mkdir my_project && cd my_project
```

Next, you'll create the `CMakeLists.txt` file and declare a dependency on GoogleTest. There are many ways to express dependencies in the CMake ecosystem; in this quickstart, you'll use the `FetchContent` CMake module.
To do this, in your project directory (`my_project`), create a file named `CMakeLists.txt` with the following contents:

```
cmake_minimum_required(VERSION 3.14)
```

```
project(my_project)

# GoogleTest requires at least C++14
set(CMAKE_CXX_STANDARD 14)

include(FetchContent)
FetchContent_Declare(
  googletest
  URL
https://github.com/google/googletest/archive/03597a01ee50ed33e9dfd640b249b4be3799d395.zip
)
# For Windows: Prevent overriding the parent project's compiler/linker settings
set(gtest_force_shared_crt ON CACHE BOOL "" FORCE)
FetchContent_MakeAvailable(googletest)
```

The above configuration declares a dependency on GoogleTest which is downloaded from GitHub. In the above example, `03597a01ee50ed33e9dfd640b249b4be3799d395` is the Git commit hash of the GoogleTest version to use; we recommend updating the hash often to point to the latest version.

For more information about how to create `CMakeLists.txt` files, see the CMake Tutorial.

Create and run a binary

With GoogleTest declared as a dependency, you can use GoogleTest code within your own project.

As an example, create a file named `hello_test.cc` in your `my_project` directory with the following contents:

```
#include <gtest/gtest.h>

// Demonstrate some basic assertions.
TEST(HelloTest, BasicAssertions) {
  // Expect two strings not to be equal.
  EXPECT_STRNE("hello", "world");
  // Expect equality.
  EXPECT_EQ(7 * 6, 42);
}
```

GoogleTest provides assertions that you use to test the behavior of your code. The above sample includes the main GoogleTest header file and demonstrates some basic assertions.

To build the code, add the following to the end of your `CMakeLists.txt` file:

```
enable_testing()

add_executable(
  hello_test
  hello_test.cc
)
target_link_libraries(
  hello_test
  GTest::gtest_main
)

include(GoogleTest)
gtest_discover_tests(hello_test)
```

The above configuration enables testing in CMake, declares the C++ test binary you want to build (`hello_test`), and links it to GoogleTest (`gtest_main`). The last two lines enable CMake's test runner to discover the tests included in the binary, using the `GoogleTest` CMake module.

Now you can build and run your test:

```
my_project$ cmake -S . -B build
-- The C compiler identification is GNU 10.2.1
-- The CXX compiler identification is GNU 10.2.1
...
-- Build files have been written to: .../my_project/build
```

my_project$ cmake --build build
Scanning dependencies of target gtest
...
[100%] Built target gmock_main

my_project$ cd build && ctest
Test project .../my_project/build
    Start 1: HelloTest.BasicAssertions
1/1 Test #1: HelloTest.BasicAssertions ....   Passed   0.00 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.01 sec

Congratulations! You've successfully built and run a test binary using GoogleTest.

Next steps

- Check out the Primer to start learning how to write simple tests.
- See the code samples for more examples showing how to use a variety of GoogleTest features.