

1. Installer CMake <https://cmake.org/download/>

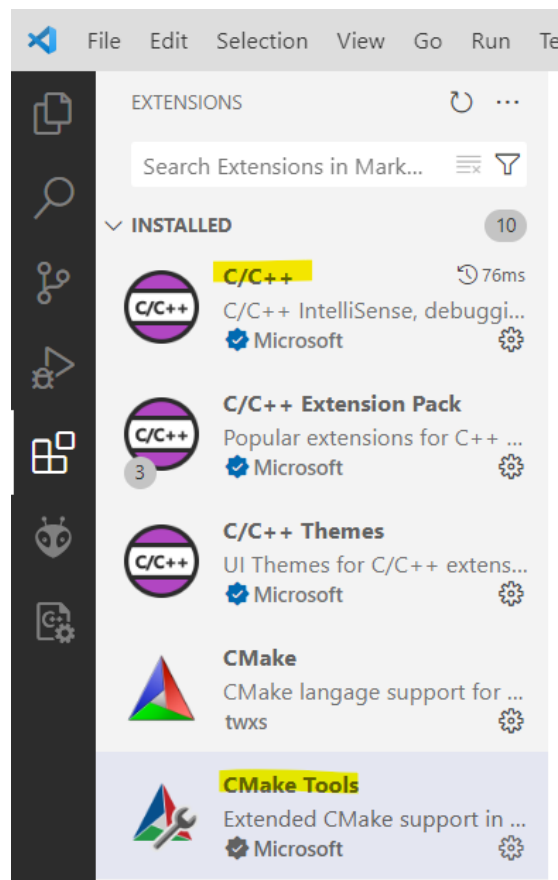
Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.29.0-rc1.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.29.0-rc1.zip

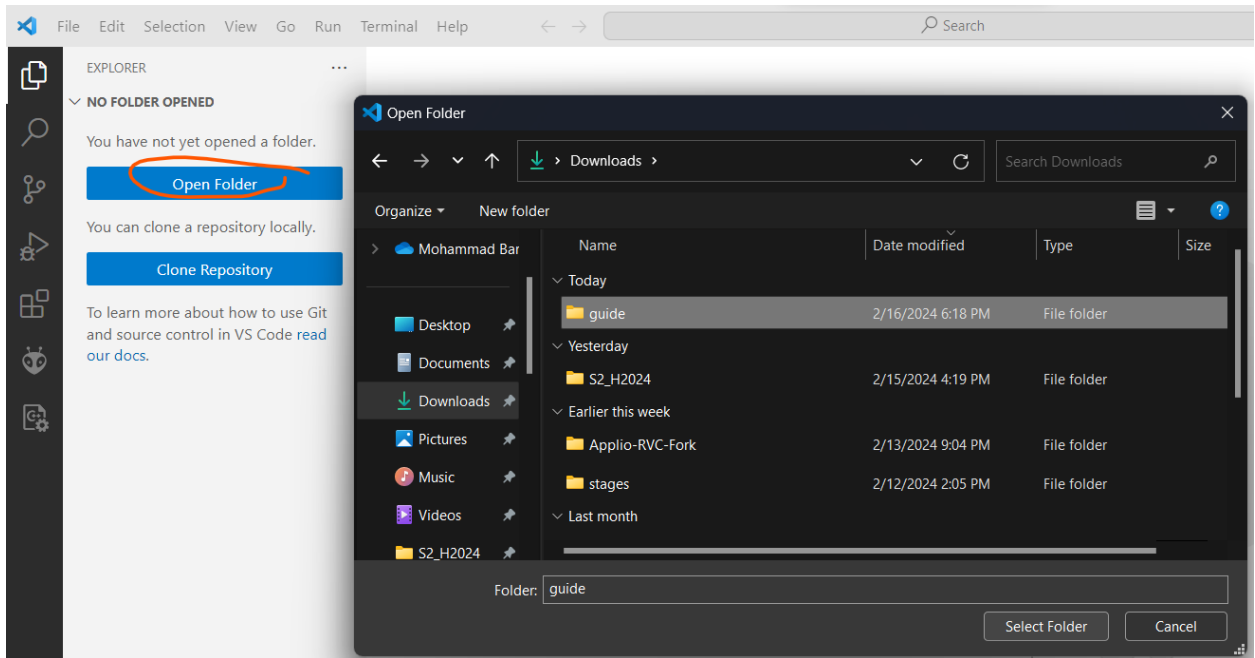
Binary distributions:

Platform	Files
Windows x64 Installer:	cmake-3.29.0-rc1-windows-x86_64.msi

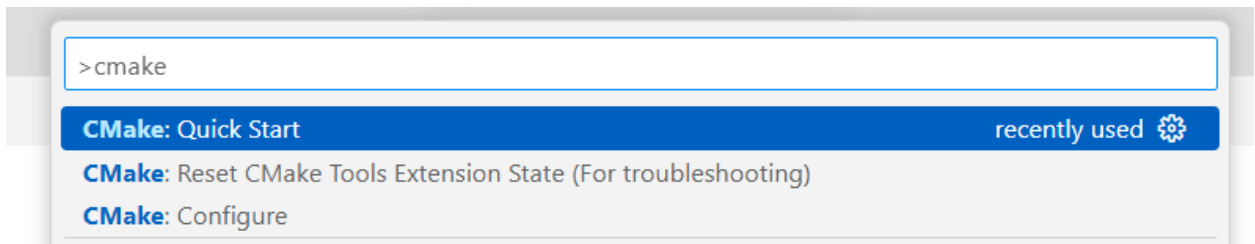
2. Installer les extensions C/C++ et CMake Tools dans VSCode.



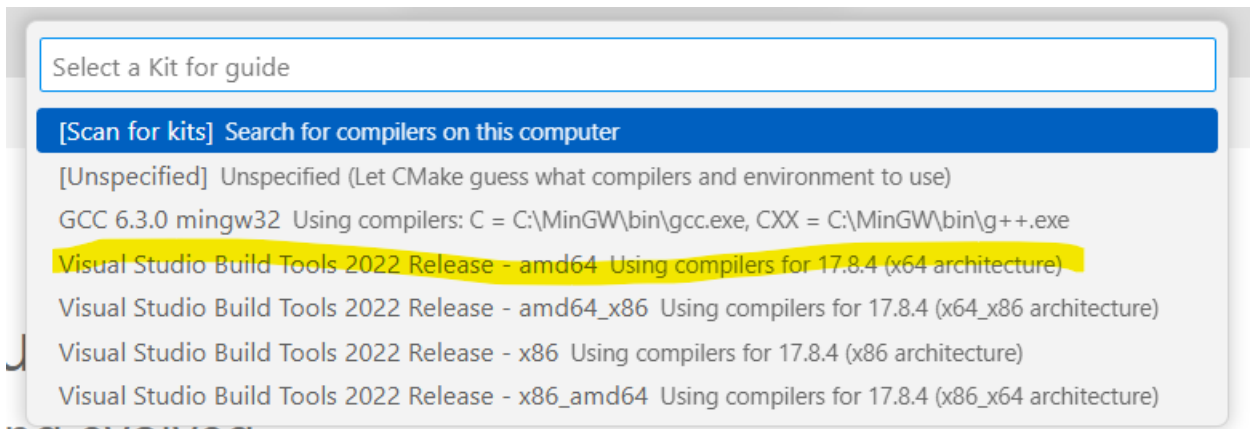
3. Créer un dossier et l'ouvrir dans VSCode.



4. Dans la barre de recherche de VSCode (en haut), taper >cmake et choisir CMake: Quick Start



5. Choisir le compilateur (par exemple, celui pour l'architecture x64).



6. Le fichier CMakeLists.txt n'existe pas. Cliquer ailleurs et ça devrait passer à l'étape suivante.

No CMakeLists.txt was found.

[Browse for CMakeLists.txt] Search for CMakeLists.txt on this computer

7. Choisir un nom pour le projet.

guide

Enter a name for the new project (Press 'Enter' to confirm or 'Escape' to cancel)

8. Choisir le langage.

C++ Create a C++ project

C Create a C project

9. Choisir Executable.

Library Create a library

Executable Create an executable

10. Le fichier CMakeLists.txt a été créé.

```
M CMakeLists.txt 1 X
M CMakeLists.txt
1 cmake_minimum_required(VERSION 3.0.0)
2 project(guide VERSION 0.1.0 LANGUAGES C CXX)
3
4 include(CTest)
5 enable_testing()
6
7 add_executable(guide main.cpp)
8
9 set(CPACK_PROJECT_NAME ${PROJECT_NAME})
10 set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
11 include(CPack)
```

11. Vous pouvez alléger le contenu du fichier ainsi (avec vos noms de fichiers source, pas les .h).

```
cmake_minimum_required(VERSION 3.24)
project(hello_world)
```

```
# Our Project
set(SOURCE_FILES
    main.cpp
    hello.cpp)
```

```
add_executable(${PROJECT_NAME} ${SOURCE_FILES})
```

12. Dans mon cas, j'ai un fichier main.cpp, un fichier hello.cpp et un fichier hello.h. J'appelle une fonction de hello.cpp dans le main.cpp.

The image displays three overlapping screenshots of a code editor, illustrating the structure of a C++ project with modular headers and source files.

Top-left screenshot (main.cpp): Shows the main function in `main.cpp`. It includes `hello.h` and calls the `afficher()` function.

```
main.cpp > ...
#include "hello.h"

int main()
{
    cout << "Hello, from main.cpp!\n";
    afficher();
    return 0;
}
```

Top-right screenshot (hello.h): Shows the header file `hello.h`. It includes `<iostream>`, uses the `std` namespace, and declares the `afficher()` function.

```
hello.h > ...
#include <iostream>
using namespace std;

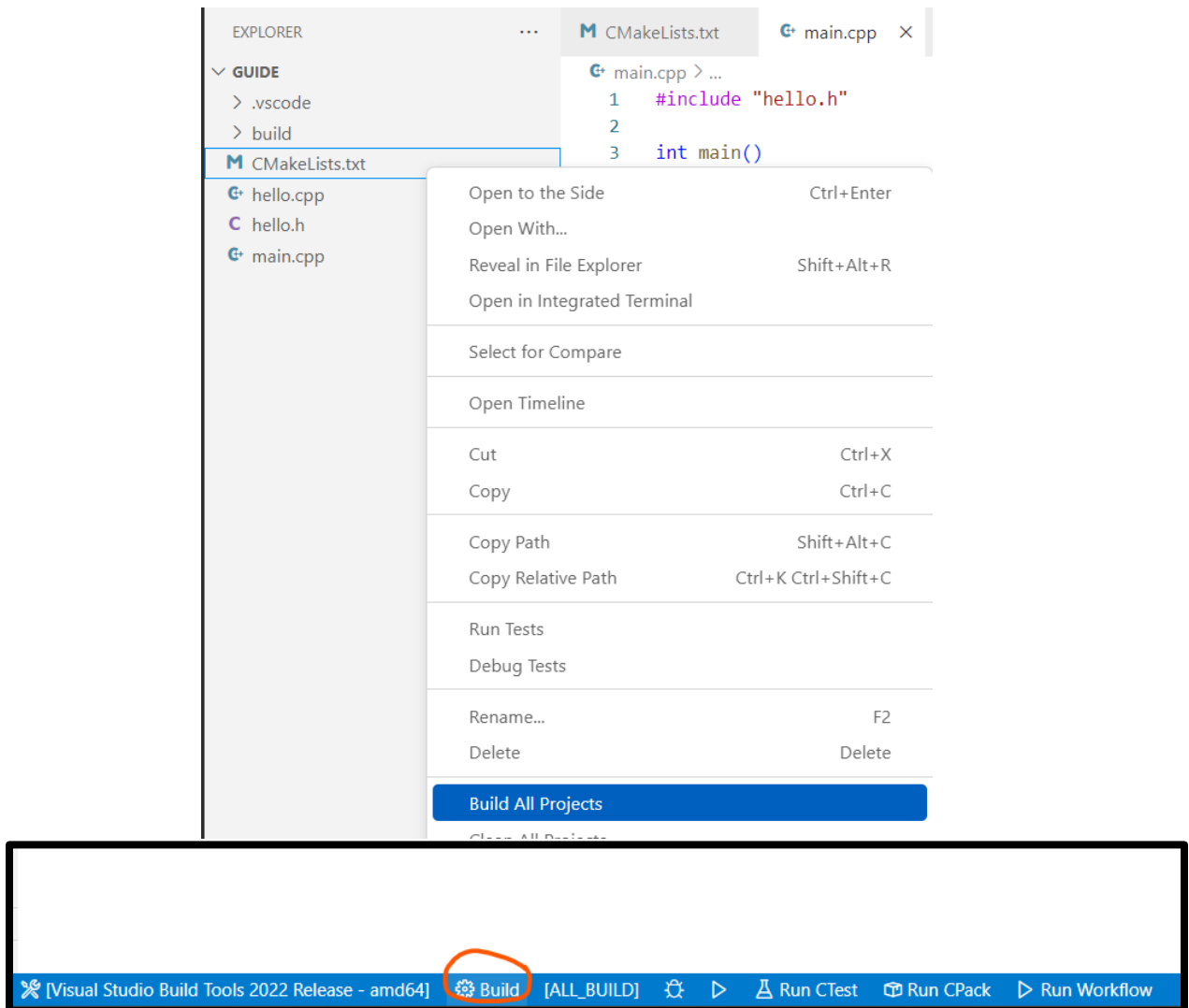
void afficher();
```

Bottom screenshot (hello.cpp): Shows the implementation file `hello.cpp`. It includes `hello.h` and implements the `afficher()` function.

```
hello.cpp > ...
#include "hello.h"

void afficher()
{
    cout << "Hello, from hello.cpp!\n";
}
```

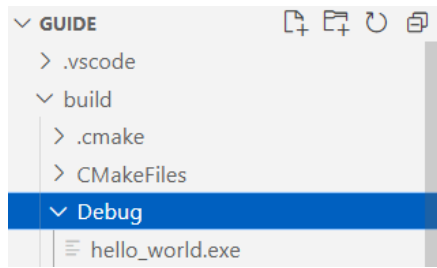
13. Pour compiler, vous pouvez soit faire un clic droit sur le fichier CMakeLists.txt et choisir Build All Projects ou cliquer sur Build en bas dans VSCode.



14. Si tout se passe bien, vous aurez un message de succès.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
[main] Building folder: guide
[build] Starting build
[proc] Executing command: "C:\Program Files\CMake\bin\cmake.EXE" --build c:/Users/barin/Down
[build] MSBuild version 17.8.3+195e7f5a3 for .NET Framework
[build]
[build]   hello.cpp
[build]   main.cpp
[build]   Generating Code...
[build]   hello_world.vcxproj -> C:\Users\barin\Downloads\guide\build\Debug\hello_world.exe
[driver] Build completed: 00:00:01.406
[build] Build finished with exit code 0
```

15. Pour exécuter le code, il suffit de trouver le fichier exécutable. Dans mon cas, il se trouvait dans le dossier Debug du dossier build.



```
PS C:\Users\barin\Downloads\guide> .\build\Debug\hello_world.exe
Hello, from main.cpp!
Hello, from hello.cpp!
PS C:\Users\barin\Downloads\guide> █
```