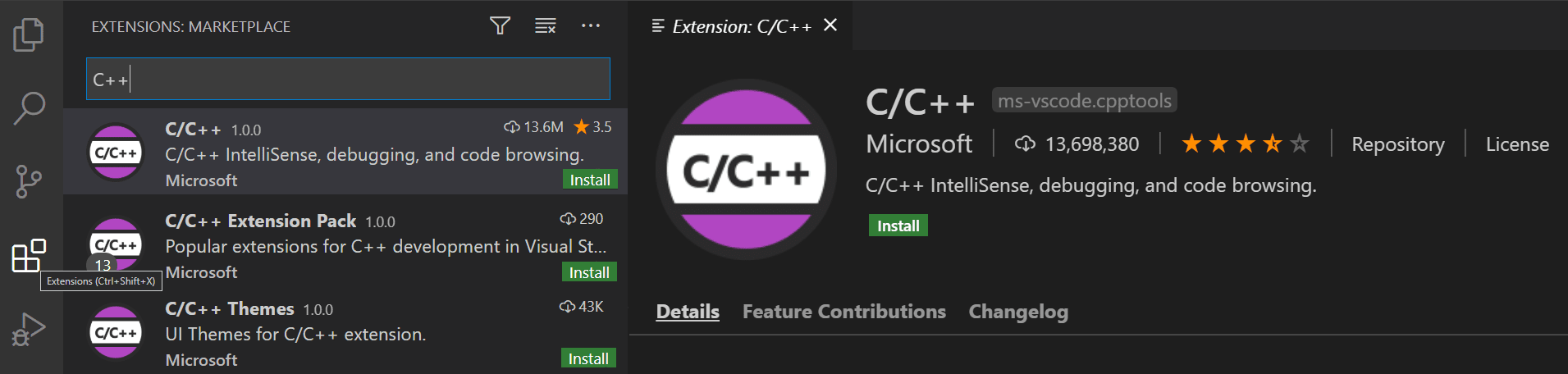
Prerequisites[#](https://code.visualstudio.com/docs/cpp/config-msvc#_prerequisites)

To successfully run this code, you must do the following:

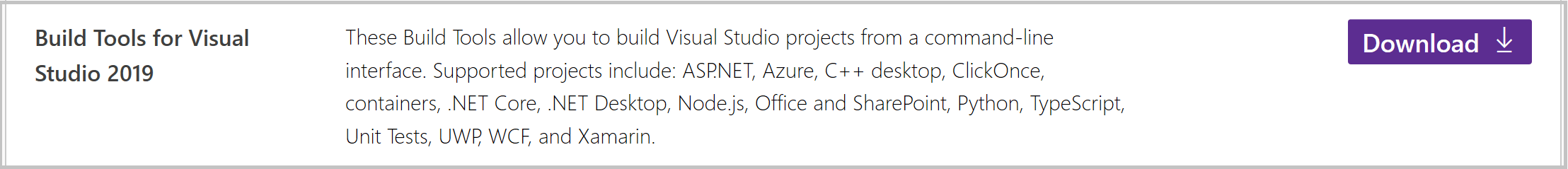
1. Install [Visual Studio Code](https://code.visualstudio.com/download).
2. Install the [C/C++ extension for VS Code](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools). You can install the C/C++ extension by searching for 'c++' in the Extensions view (Ctrl+Shift+X).



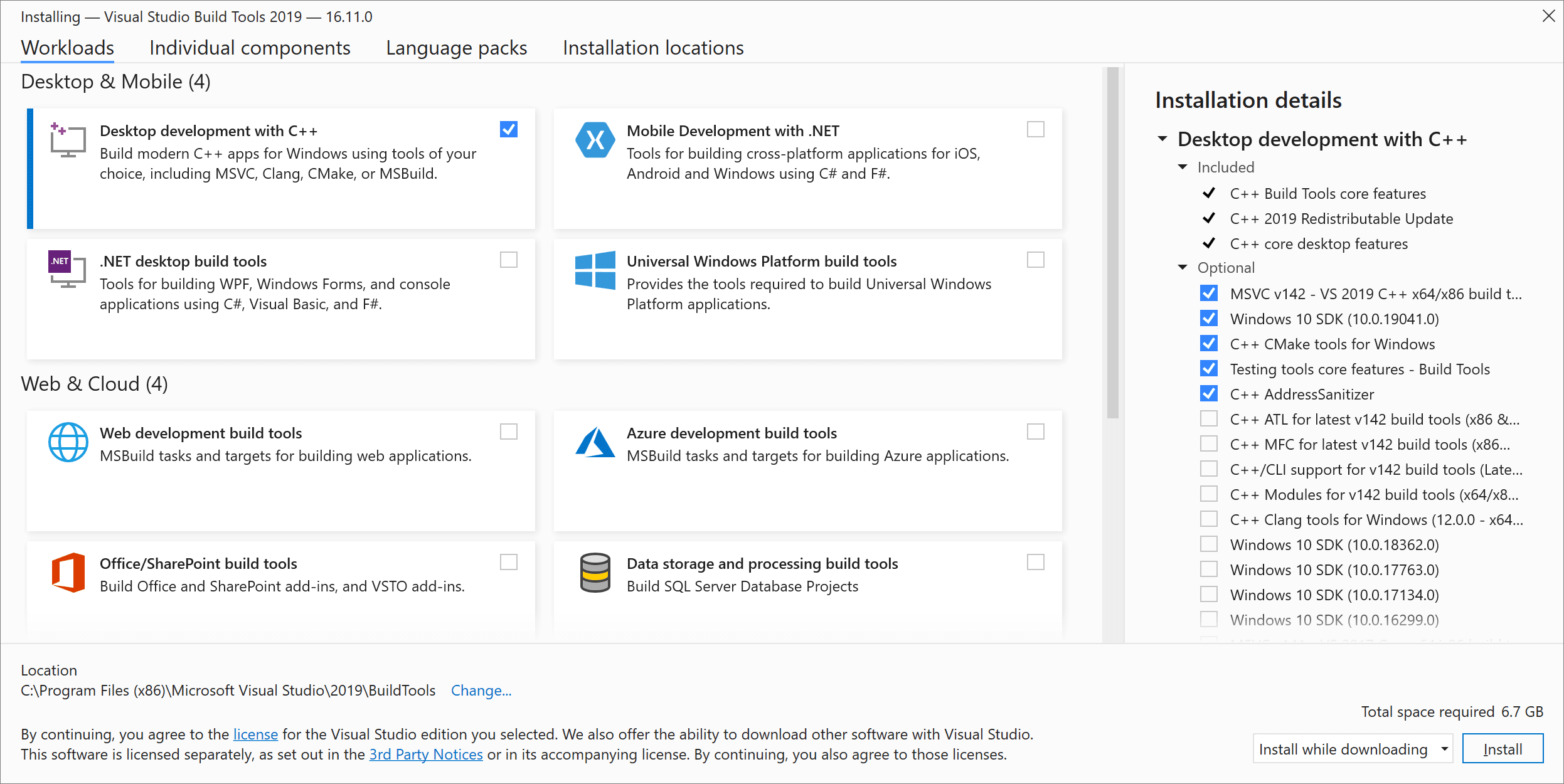
1. Install the Microsoft Visual C++ (MSVC) compiler toolset.

If you have a recent version of Visual Studio, open the Visual Studio Installer from the Windows Start menu and verify that the C++ workload is checked. If it's not installed, then check the box and select the **Modify** button in the installer.

You can also install the **Desktop development with C++** workload without a full Visual Studio IDE installation. From the Visual Studio [Downloads](https://visualstudio.microsoft.com/downloads#other) page, scroll down until you see **Tools for Visual Studio 2022** under the **All Downloads** section and select the download for **Build Tools for Visual Studio 2022**.



This will launch the Visual Studio Installer, which will bring up a dialog showing the available Visual Studio Build Tools workloads. Check the **Desktop development with C++** workload and select **Install**.

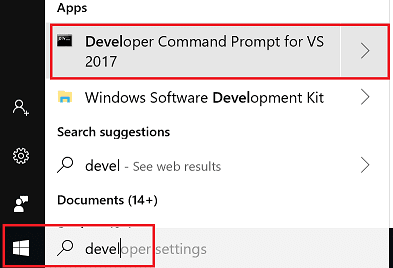


**Note**: You can use the C++ toolset from Visual Studio Build Tools along with Visual Studio Code to compile, build, and verify any C++ codebase as long as you also have a valid Visual Studio license (either Community, Pro, or Enterprise) that you are actively using to develop that C++ codebase.

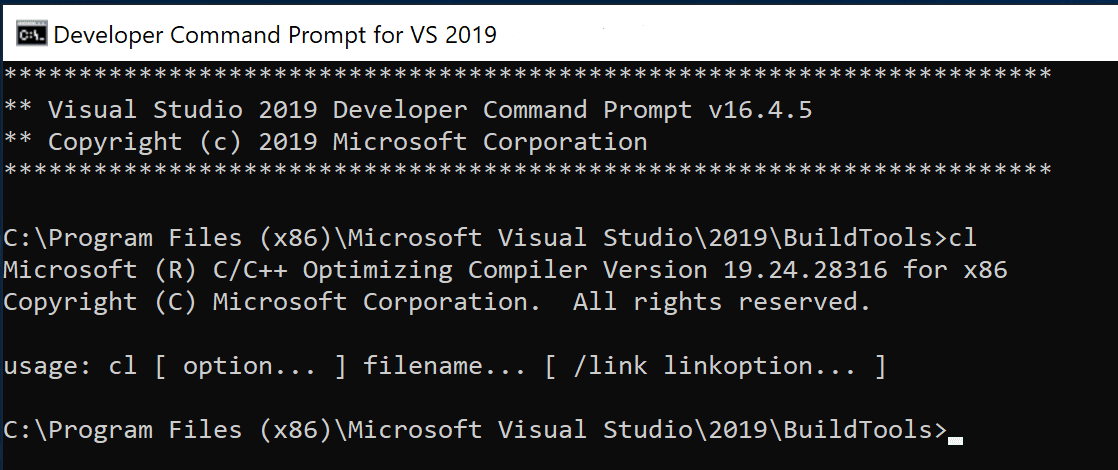
Check your Microsoft Visual C++ installation[#](https://code.visualstudio.com/docs/cpp/config-msvc#_check-your-microsoft-visual-c-installation)

To use MSVC from a command line or VS Code, you must run from a **Developer Command Prompt for Visual Studio**. An ordinary shell such as PowerShell, Bash, or the Windows command prompt does not have the necessary path environment variables set.

To open the Developer Command Prompt for VS, start typing 'developer' in the Windows Start menu, and you should see it appear in the list of suggestions. The exact name depends on which version of Visual Studio or the Visual Studio Build Tools you have installed. Select the item to open the prompt.



You can test that you have the C++ compiler, cl.exe, installed correctly by typing 'cl' and you should see a copyright message with the version and basic usage description.



If the Developer Command Prompt is using the BuildTools location as the starting directory (you wouldn't want to put projects there), navigate to your user folder (C:\users\{your username}\) before you start creating new projects.

**Note**: If for some reason you can't run VS Code from a **Developer Command Prompt**, you can find a workaround for building C++ projects with VS Code in [Run VS Code outside a Developer Command Prompt](https://code.visualstudio.com/docs/cpp/config-msvc#_run-vs-code-outside-the-developer-command-prompt).

Create Hello World[#](https://code.visualstudio.com/docs/cpp/config-msvc#_create-hello-world)

From the Developer Command Prompt, create an empty folder called "projects" where you can store all your VS Code projects, then create a subfolder called "helloworld", navigate into it, and open VS Code (code) in that folder (.) by entering the following commands:

mkdir projects

cd projects

mkdir helloworld

cd helloworld

code .

The "code ." command opens VS Code in the current working folder, which becomes your "workspace". As you go through the tutorial, you will see three files created in a .vscode folder in the workspace:

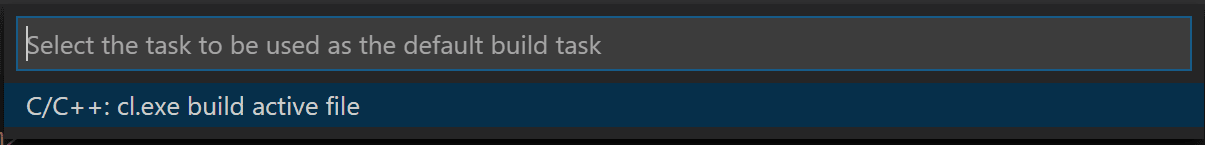
* tasks.json (build instructions)
* launch.json (debugger settings)
* c\_cpp\_properties.json (compiler path and IntelliSense settings)

Add a source code file[#](https://code.visualstudio.com/docs/cpp/config-msvc#_add-a-source-code-file)

Build graphs.cpp[#](https://code.visualstudio.com/docs/cpp/config-msvc#_build-helloworldcpp)

Next, you will create a tasks.json file to tell VS Code how to build (compile) the program. This task will invoke the Microsoft C++ compiler to create an executable file based on the source code.

From the main menu, choose **Terminal** > **Configure Default Build Task**. In the dropdown, which will display a tasks dropdown listing various predefined build tasks for C++ compilers. Choose **cl.exe build active file**, which will build the file that is currently displayed (active) in the editor.



This will create a tasks.json file in a .vscode folder and open it in the editor.

Your new tasks.json file should look similar to the JSON below:

{

"version": "2.0.0",

"tasks": [

{

"type": "shell",

"label": "cl.exe build active file",

"command": "cl.exe",

"args": [

"/Zi",

"/EHsc",

"/Fe:",

"${fileDirname}\\${fileBasenameNoExtension}.exe",

"${file}"

],

"problemMatcher": ["$msCompile"],

"group": {

"kind": "build",

"isDefault": true

}

}

]

}

The command setting specifies the program to run; in this case that is "cl.exe". The args array specifies the command-line arguments that will be passed to cl.exe. These arguments must be specified in the order expected by the compiler. This task tells the C++ compiler to take the active file (${file}), compile it, and create an executable file (/Fe: switch) in the current directory (${fileDirname}) with the same name as the active file but with the .exe extension (${fileBasenameNoExtension}.exe), resulting in helloworld.exe for our example.

**Note**: You can learn more about tasks.json variables in the [variables reference](https://code.visualstudio.com/docs/editor/variables-reference).

The label value is what you will see in the tasks list; you can name this whatever you like.

The problemMatcher value selects the output parser to use for finding errors and warnings in the compiler output. For cl.exe, you'll get the best results if you use the $msCompile problem matcher.

The "isDefault": true value in the group object specifies that this task will be run when you press Ctrl+Shift+B. This property is for convenience only; if you set it to false, you can still run it from the Terminal menu with **Tasks: Run Build Task**.

Running the build[#](https://code.visualstudio.com/docs/cpp/config-msvc#_running-the-build)

1. Go back to graphs.cpp. Your task builds the active file and you want to build graphs.cpp.
2. To run the build task defined in tasks.json, press Ctrl+Shift+B or from the **Terminal** main menu choose **Tasks: Run Build Task**.
3. When the task starts, you should see the Integrated Terminal panel appear below the source code editor. After the task completes, the terminal shows output from the compiler that indicates whether the build succeeded or failed. For a successful C++ build, the output looks something like this:

