Create a Hello World app

First, lets get a project set up.

1. Launch a Windows command prompt (Enter **Windows command prompt** in the Windows search bar).
2. Run the following commands. These will create an empty folder called `projects` where you can place all your VS Code projects. There, the next commands will create and navigate to a subfolder called `helloworld`. From there, you will open `helloworld` directly in VS Code.

```
mkdir projects
cd projects
mkdir helloworld
cd helloworld
code .
```

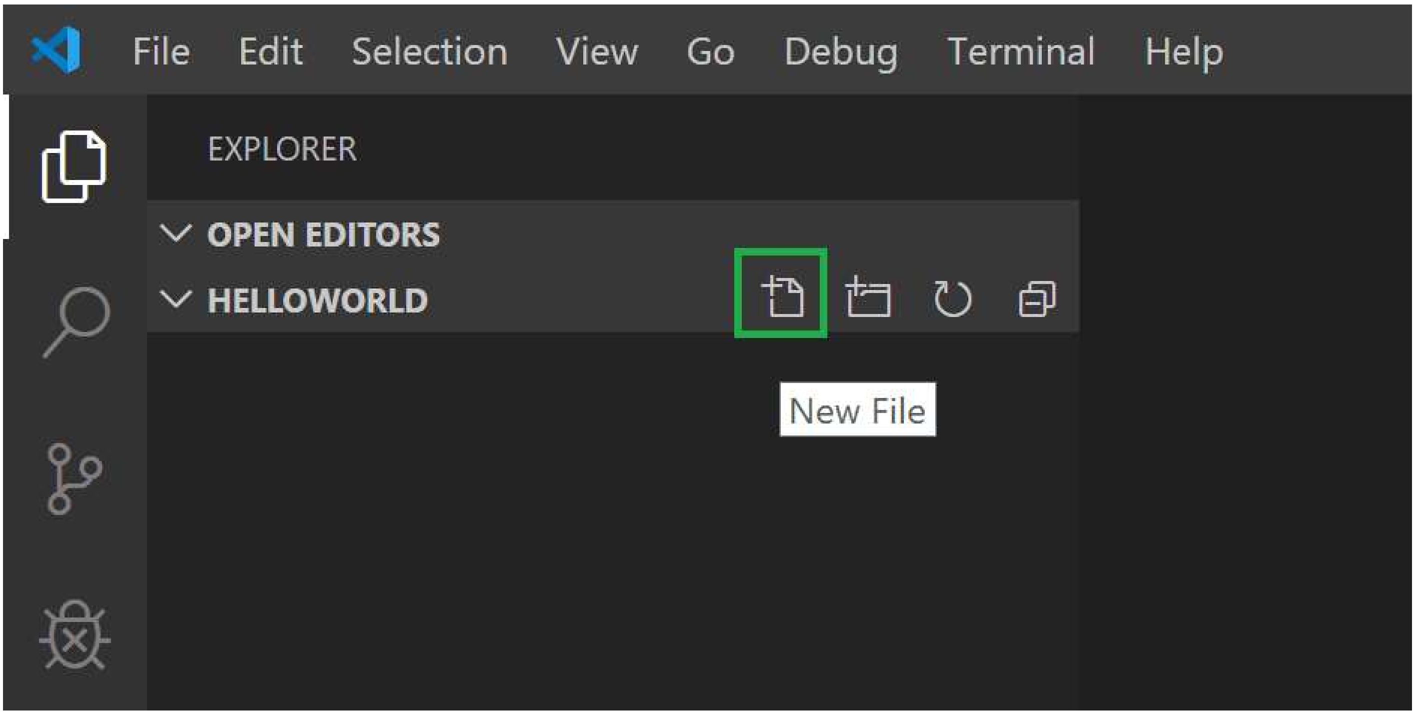
The "code ." command opens VS Code in the current working folder, which becomes your "workspace". Accept the Workspace Trust dialog by selecting **Yes, I trust the authors** since this is a folder you created.

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 Hello World source code file

In the File Explorer title bar, select the **New File** button and name the file `helloworld.cpp`.

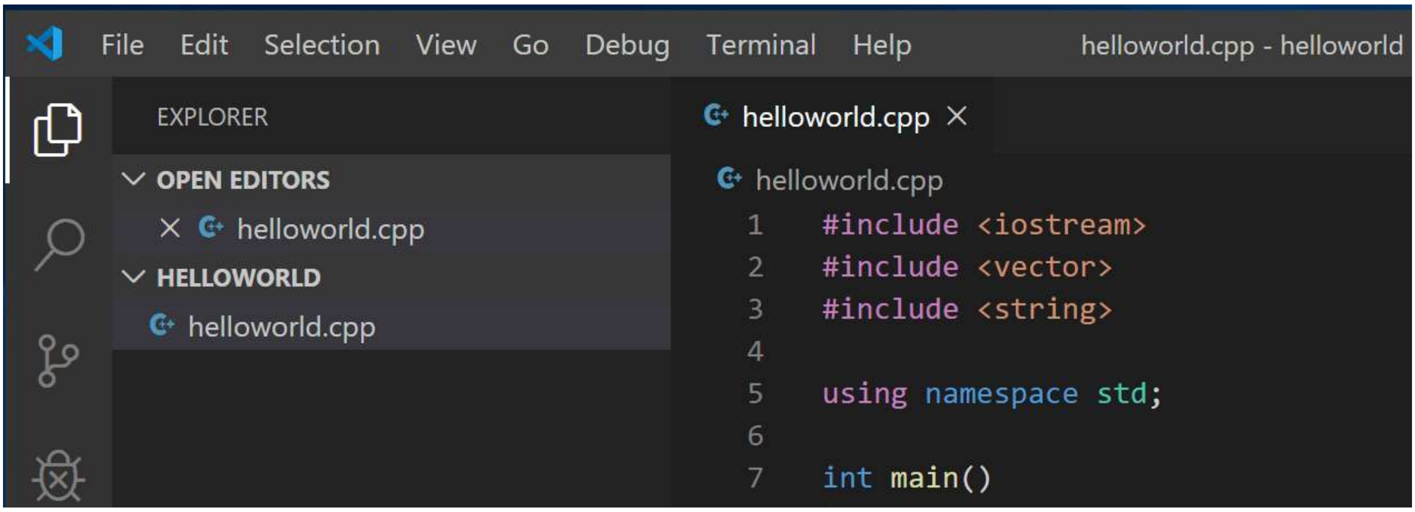


Add hello world source code

Now paste in this source code:

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
{
    vector<string> msg {"Hello", "C++", "World", "from", "VS Code", "and the C++ extension!"};

    for (const string& word : msg)
    {
        cout << word << " ";
    }
    cout << endl;
}
```

Now press `Ctrl+S` to save the file. Notice how the file you just added appears in the **File Explorer** view (`Ctrl+Shift+E`) in the side bar of VS Code:



```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
```

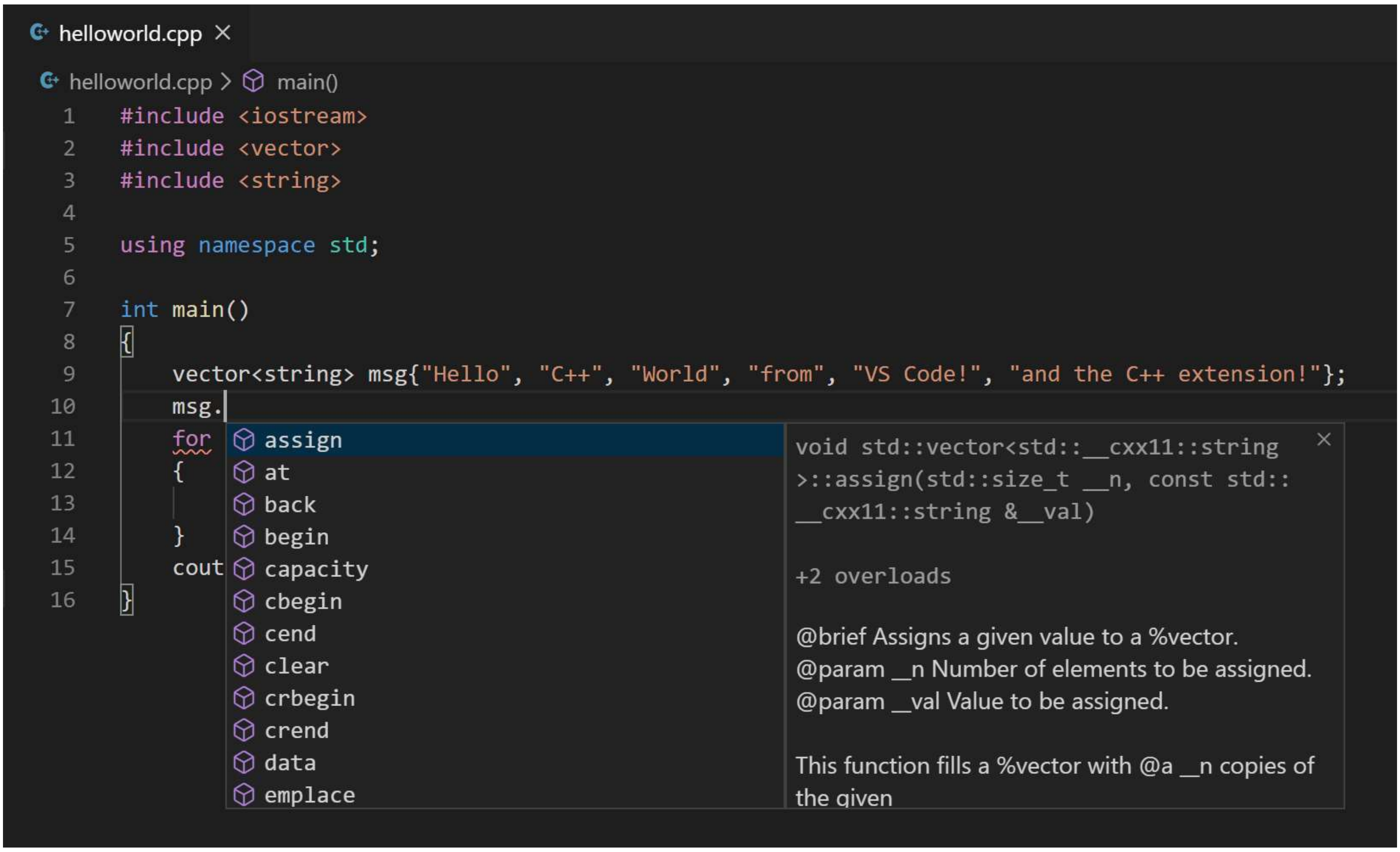
You can also enable Auto Save to automatically save your file changes, by selecting **File** > **Auto Save**. You can find out more about the other views in the VS Code User Interface documentation.

> **Note**: When you save or open a C++ file, you may see a notification from the C/C++ extension about the availability of an Insiders version, which lets you test new features and fixes. You can ignore this notification by selecting the `X` (**Clear Notification**).

Explore IntelliSense

IntelliSense is a tool to help you code faster and more efficiently by adding code editing features such as code completion, parameter info, quick info, and member lists.

To see IntelliSense in action, hover over `vector` or `string` to see their type information. If you type `msg.` in line 10, you can see a completion list of recommended member functions to call, all generated by IntelliSense:



```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
{
    vector<string> msg{"Hello", "C++", "World", "from", "VS Code!", "and the C++ extension!"};
    msg.
    for
    {

    }
    cout
}
```

```
void std::vector<std::__cxx11::string
>::assign(std::size_t __n, const std::
__cxx11::string &__val)

+2 overloads
```

@brief Assigns a given value to a %vector.
@param __n Number of elements to be assigned.
@param __val Value to be assigned.

This function fills a %vector with @a __n copies of the given

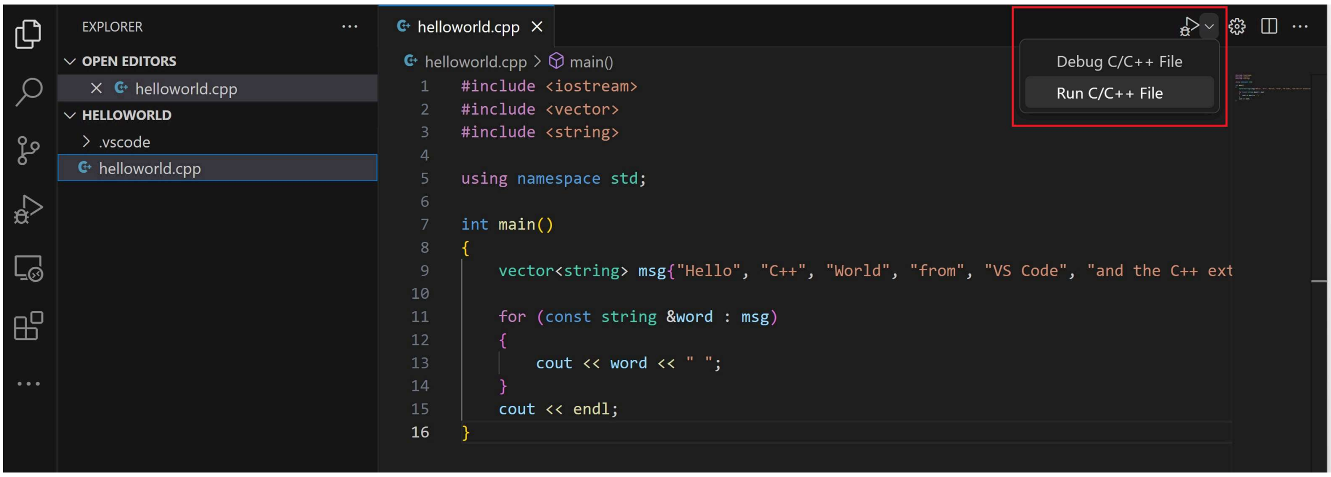
You can press the `Tab` key to insert a selected member. If you then add open parenthesis, IntelliSense will show information on which arguments are required.

If IntelliSense is not already configured, open the Command Palette (`Ctrl+Shift+P`) and enter **Select IntelliSense Configuration**. From the dropdown of compilers, select `Use gcc.exe` to configure. More information can be found in the IntelliSense configuration documentation.

Run helloworld.cpp

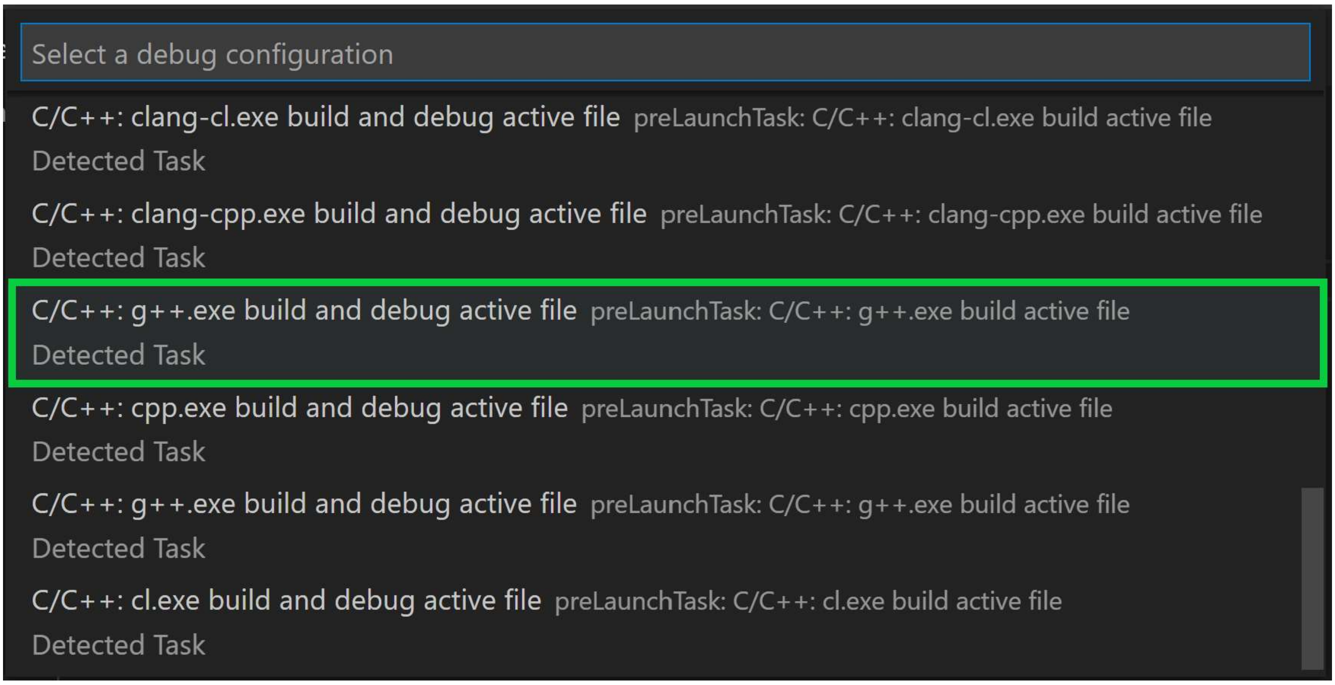
Remember, the C++ extension uses the C++ compiler you have installed on your machine to build your program. Make sure you have completed the "Installing the MinGW-w64 toolchain" step before attempting to run and debug `helloworld.cpp` in VS Code.

1. Open `helloworld.cpp` so that it is the active file.
2. Press the play button in the top right corner of the editor.



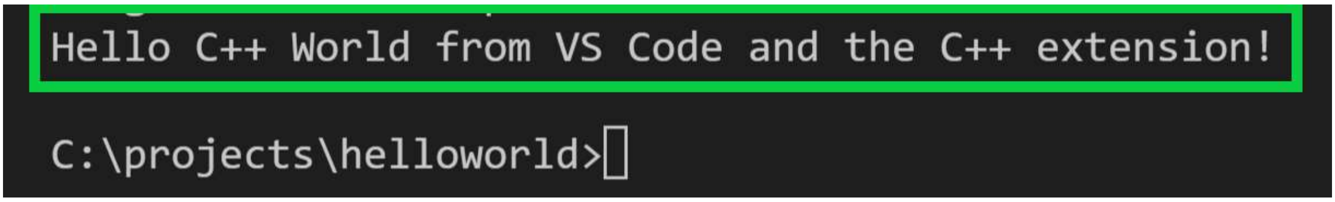
```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
{
    vector<string> msg{"Hello", "C++", "World", "from", "VS Code", "and the C++ ext

    for (const string &word : msg)
    {
        cout << word << " ";
    }
    cout << endl;
}
```

3. Choose **C/C++: g++.exe build and debug active file** from the list of detected compilers on your system.


Select a debug configuration
C/C++: clang-cl.exe build and debug active file preLaunchTask: C/C++: clang-cl.exe build active file
Detected Task
C/C++: clang-cpp.exe build and debug active file preLaunchTask: C/C++: clang-cpp.exe build active file
Detected Task
C/C++: g++.exe build and debug active file preLaunchTask: C/C++: g++.exe build active file
Detected Task
C/C++: cpp.exe build and debug active file preLaunchTask: C/C++: cpp.exe build active file
Detected Task
C/C++: g++.exe build and debug active file preLaunchTask: C/C++: g++.exe build active file
Detected Task
C/C++: cl.exe build and debug active file preLaunchTask: C/C++: cl.exe build active file
Detected Task

You'll only be asked to choose a compiler the first time you run `helloworld.cpp`. This compiler will be set as the "default" compiler in `tasks.json` file.

4. After the build succeeds, your program's output will appear in the integrated **Terminal**.



```
Hello C++ World from VS Code and the C++ extension!

C:\projects\helloworld>
```