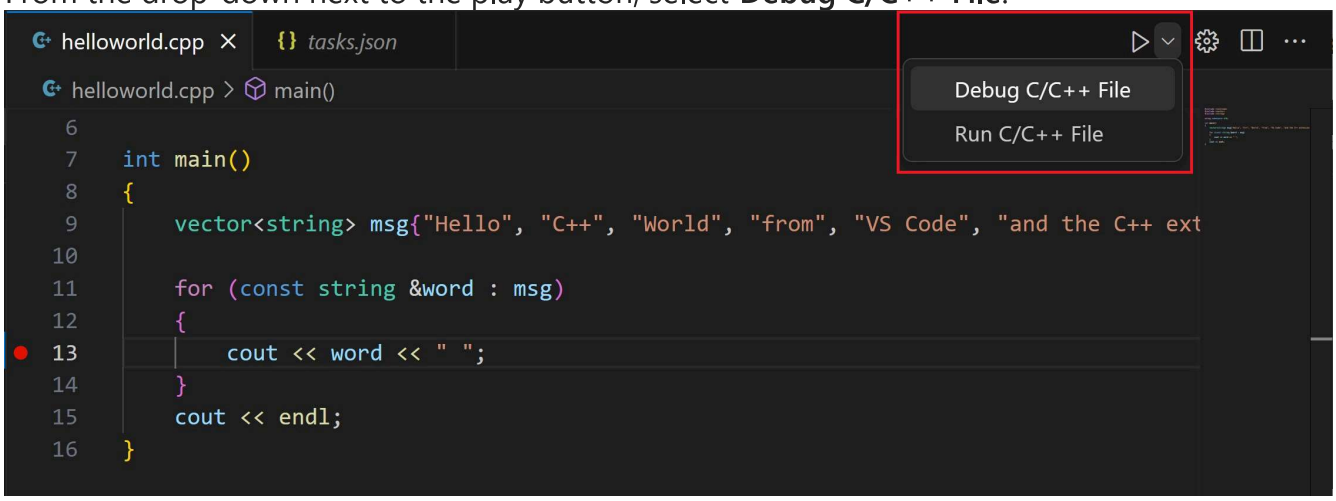
Debug helloworld.cpp

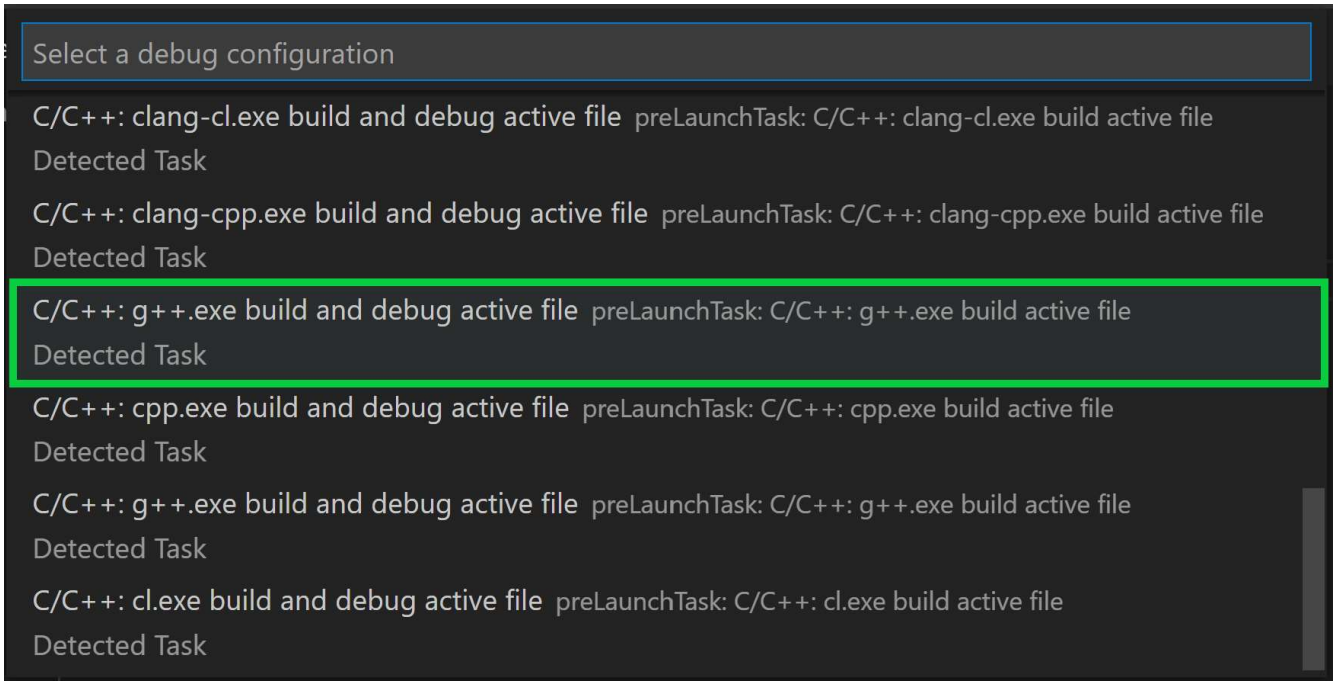
To debug your code,

1. Go back to `helloworld.cpp` so that it is the active file.
2. Set a breakpoint by clicking on the editor margin or using F9 on the current line.

3. From the drop-down next to the play button, select **Debug C/C++ File**.



4. Choose **C/C++: g++ build and debug active file** from the list of detected compilers on your system (you'll only be asked to choose a compiler the first time you run or debug `helloworld.cpp`).

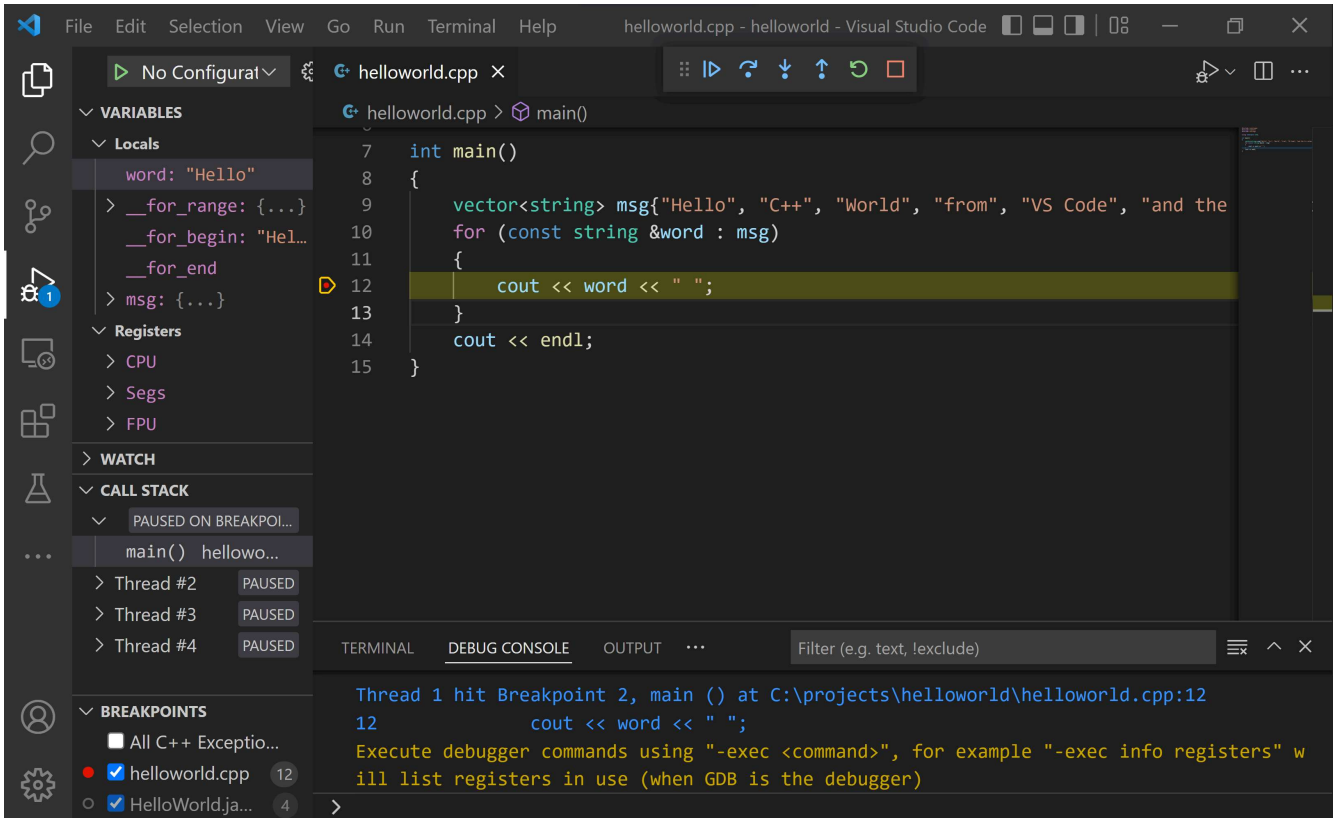


The play button has two modes: **Run C/C++ File** and **Debug C/C++ File**. It will default to the last-used mode. If you see the debug icon in the play button, you can just select the play button to debug, instead of using the drop-down.

Explore the debugger

Before you start stepping through the code, let's take a moment to notice several changes in the user interface:

- The Integrated Terminal appears at the bottom of the source code editor. In the **Debug Console** tab, you see output that indicates the debugger is up and running.
- The editor highlights the line where you set a breakpoint before starting the debugger:



- The **Run and Debug** view on the left shows debugging information. You'll see an example later in the tutorial.
- At the top of the code editor, a debugging control panel appears. You can move this around the screen by grabbing the dots on the left side.

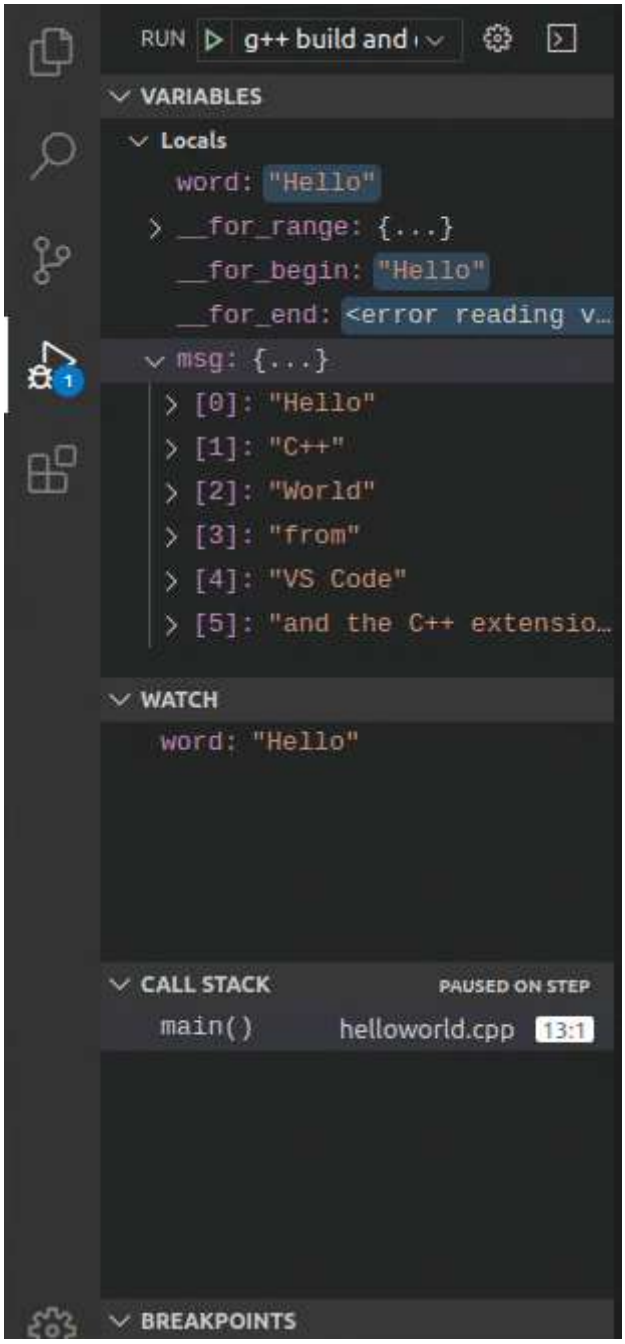
Step through the code

Now you're ready to start stepping through the code.

1. Select the **Step over** icon in the debugging control panel.

   This will advance program execution to the first line of the for loop, and skip over all the internal function calls within the `vector` and `string` classes that are invoked when the `msg` variable is created and initialized. Notice the change in the **Variables** window on the left.



In this case, the errors are expected because, although the variable names for the loop are now visible to the debugger, the statement has not executed yet, so there is nothing to read at this point. The contents of `msg` are visible, however, because that statement has completed.

2. Press **Step over** again to advance to the next statement in this program (skipping over all the internal code that is executed to initialize the loop). Now, the **Variables** window shows information about the loop variables.

3. Press **Step over** again to execute the `cout` statement. (Note that the C++ extension does not print any output to the **Debug Console** until the loop exits.)

4. If you like, you can keep pressing **Step over** until all the words in the vector have been printed to the console. But if you are curious, try pressing the **Step Into** button to step through source code in the C++ standard library!

   To return to your own code, one way is to keep pressing **Step over**. Another way is to set a breakpoint in your code by switching to the `helloworld.cpp` tab in the code editor, putting the insertion point somewhere on the `cout` statement inside the loop, and pressing `F9`. A

red dot appears in the gutter on the left to indicate that a breakpoint has been set on this line.

```
12    for (const string& word : msg)
13    {
14        cout << word << " ";
15    }
16 }
```

Then press `F5` to start execution from the current line in the standard library header. Execution will break on `cout`. If you like, you can press `F9` again to toggle off the breakpoint.

When the loop has completed, you can see the output in the Integrated Terminal, along with some other diagnostic information that is output by GDB.