在MAC上使用VSCode运行调试简单c/c++程序

在MAC上使用VSCode运行调试简单c/c++程序

- 1.下载安装
- 2.将VS Code添加到系统`PATH`
- 3. 使用命令行创建工程，打开VS Code
- 4.配置编译器路径
- 5. 创建一个build 任务
- 6.debug设置
- 7.添加代码文件
- 8.编译运行代码
- 9.简单调试
- 10.插件安装
- 注意点
- 参考

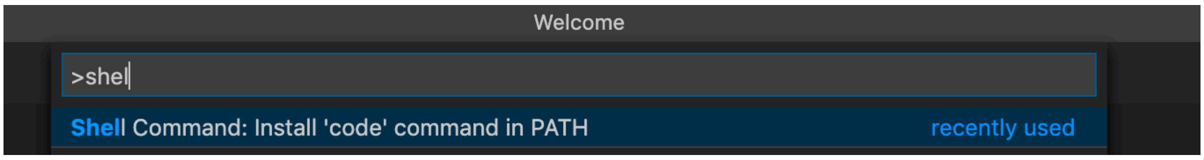
1.下载安装

进入官网(https://code.visualstudio.com/)正常点击下载就好

2.将VS Code添加到系统`PATH`

添加到PATH的好处是可以通过命令行直接在当前目录创建工程目录并在当前目录打开VSCode图形化界面

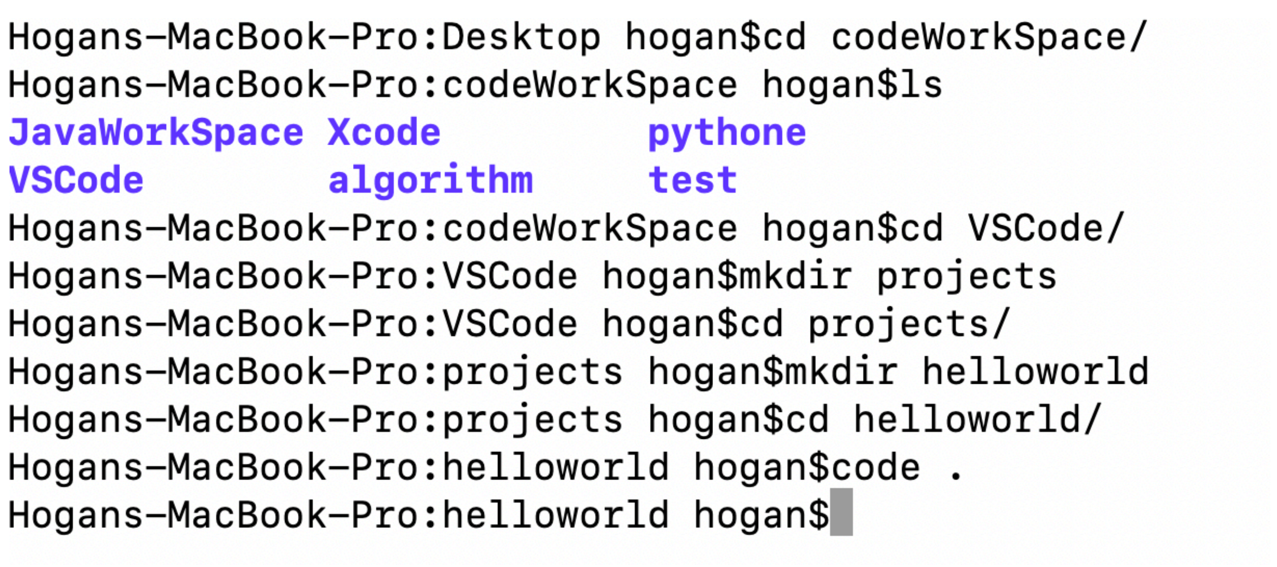
1. 打开`VS Code`
2. 按住`⇧⌘P` 或者选择系统顶部菜单栏 `View` -> `Command pattle`(第一个选项)
3. 输入`shell`，选择第一个选项即可



这样就添加好了

3. 使用命令行创建工程，打开VS Code

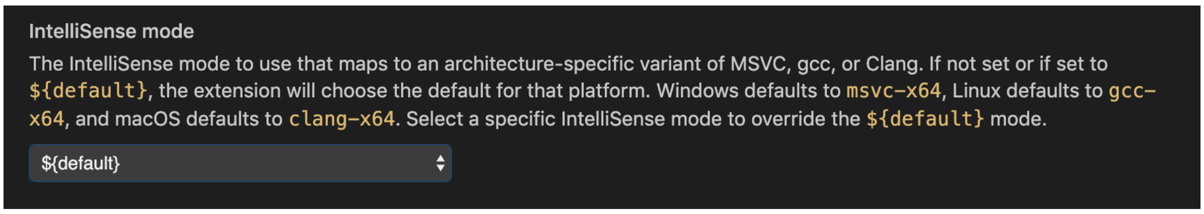
如下图创建helloworld工程并打开vscode



```
Hogans-MacBook-Pro:Desktop hogan$cd codeWorkSpace/
Hogans-MacBook-Pro:codeWorkSpace hogan$ls
JavaWorkSpace Xcode         pythone
VSCode        algorithm     test
Hogans-MacBook-Pro:codeWorkSpace hogan$cd VSCode/
Hogans-MacBook-Pro:VSCode hogan$mkdir projects
Hogans-MacBook-Pro:VSCode hogan$cd projects/
Hogans-MacBook-Pro:projects hogan$mkdir helloworld
Hogans-MacBook-Pro:projects hogan$cd helloworld/
Hogans-MacBook-Pro:helloworld hogan$code .
Hogans-MacBook-Pro:helloworld hogan$
```

4.配置编译器路径

1. 按住`⇧⌘P` 输入 `c/c++`, 选择**Edit Configurations (UI)**
2. 把 **IntelliSense mode** 改成 `${default}`, （原来是 `clang-x64`）.

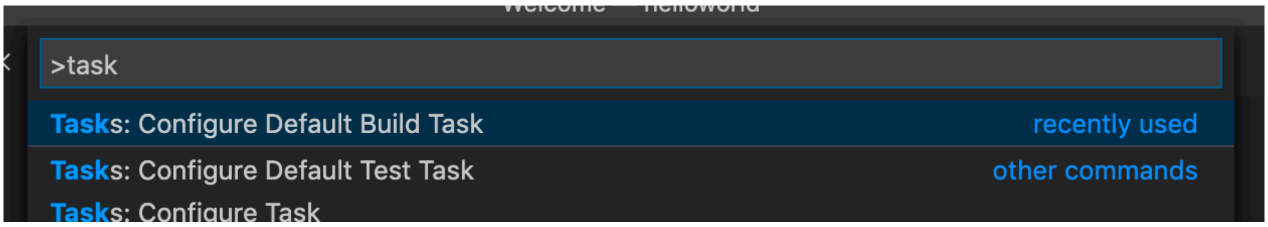


IntelliSense mode

The IntelliSense mode to use that maps to an architecture-specific variant of MSVC, gcc, or Clang. If not set or if set to `${default}`, the extension will choose the default for that platform. Windows defaults to `msvc-x64`, Linux defaults to `gcc-x64`, and macOS defaults to `clang-x64`. Select a specific IntelliSense mode to override the `${default}` mode.

5. 创建一个build 任务

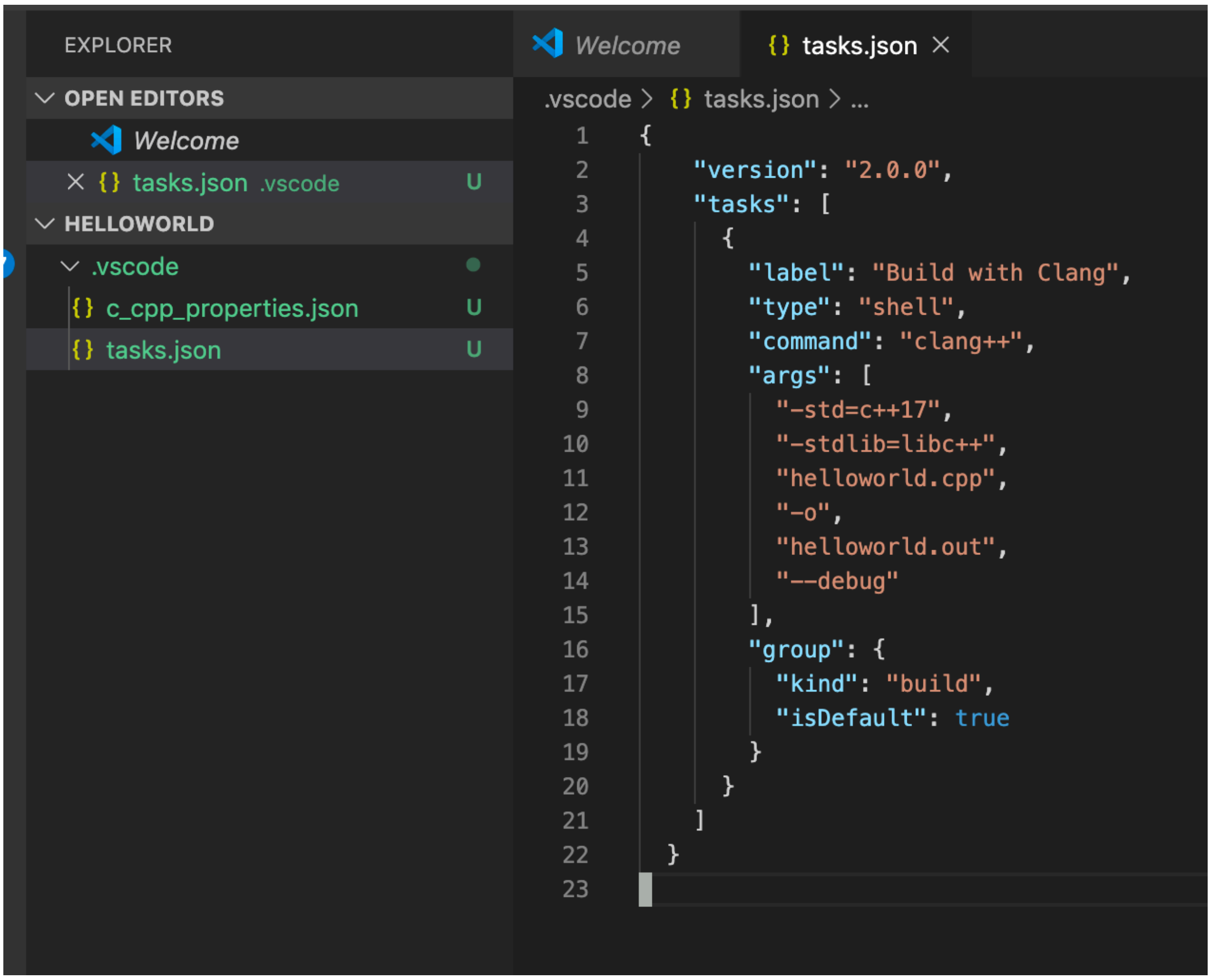
1.打开命令行菜单（`⇧⌘P`），输入`task`, 选择**Configure Default Build Task**. 接着选择 **Create tasks.json file from template**, 然后是 **Others**.



>task
Tasks: Configure Default Build Task
recently used
Tasks: Configure Default Test Task
other commands
Tasks: Configure Task

2.直接用下面的代码替换掉出现的文件

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Build with Clang",
      "type": "shell",
      "command": "clang++",
      "args": [
        "-std=c++17",
        "-stdlib=libc++",
        "helloworld.cpp",
        "-o",
        "helloworld.out",
        "--debug"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      }
    }
  ]
}
```



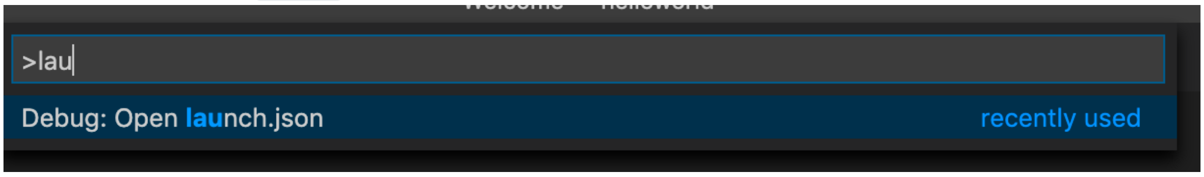
EXPLORER
OPEN EDITORS
Welcome
× {} tasks.json .vscode U
HELLOWORLD
.vscode
{} c_cpp_properties.json U
{} tasks.json U

Welcome | {} tasks.json ×
.vscode > {} tasks.json > ...

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "Build with Clang",
            "type": "shell",
            "command": "clang++",
            "args": [
                "-std=c++17",
                "-stdlib=libc++",
                "helloworld.cpp",
                "-o",
                "helloworld.out",
                "--debug"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ]
}
```

6.debug设置

1. 打开命令行菜单输入 `lauch` 选择**GDB/LLDB**

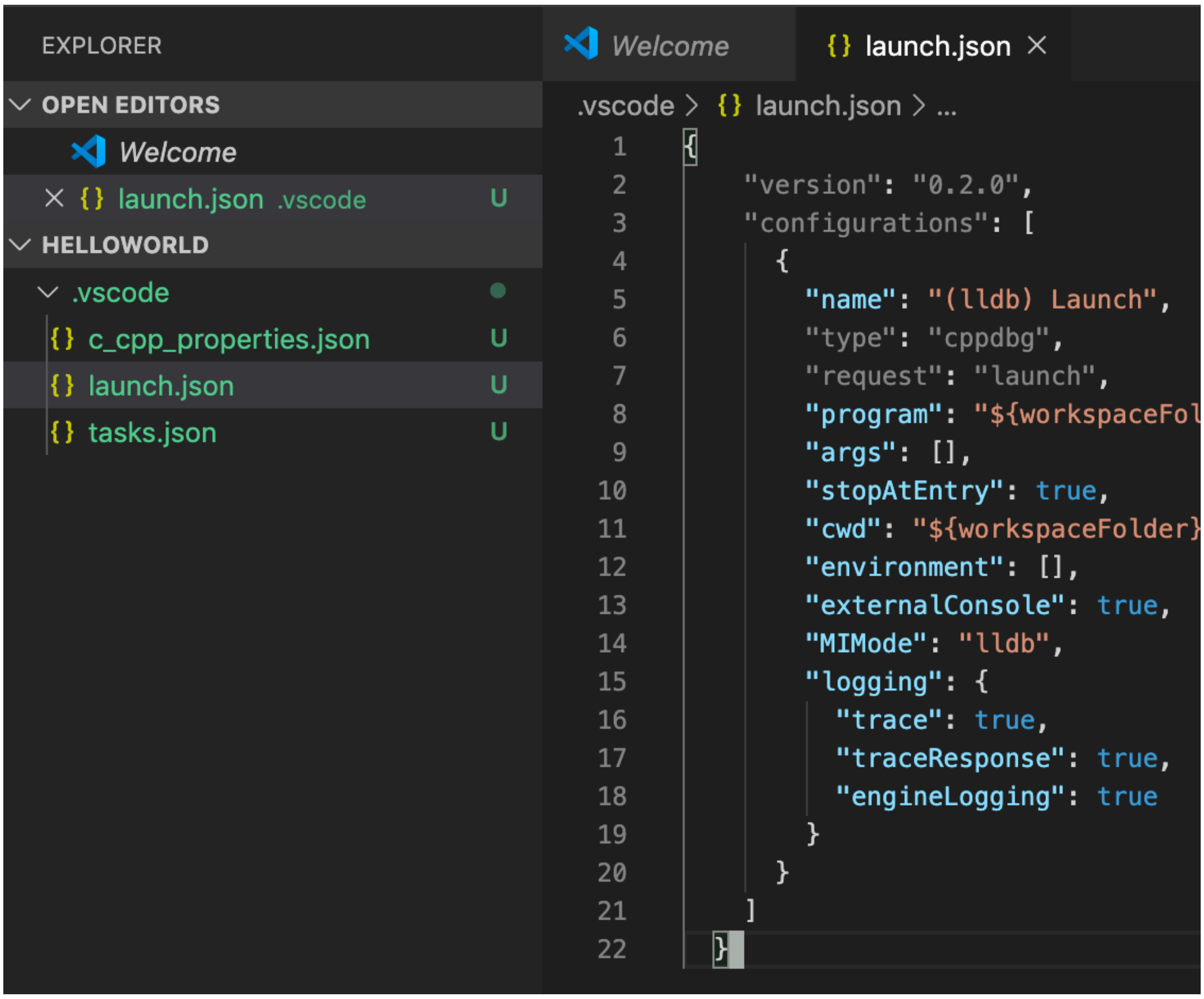


>lau
Debug: Open launch.json — recently used

2. 替换代码，cmd+s保存下

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(lldb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/helloworld.out",
      "args": [],
      "stopAtEntry": true,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": true,
      "MIMode": "lldb",
      "logging": {
        "trace": true,
        "traceResponse": true,
        "engineLogging": true
      }
    }
  ]
}
```

这一步结束，可以注意工程目录文件到变化



```
{
    "version": "0.2.0",
    "configurations": [
      {
        "name": "(lldb) Launch",
        "type": "cppdbg",
        "request": "launch",
        "program": "${workspaceFol
        "args": [],
        "stopAtEntry": true,
        "cwd": "${workspaceFolder}
        "environment": [],
        "externalConsole": true,
        "MIMode": "lldb",
        "logging": {
          "trace": true,
          "traceResponse": true,
          "engineLogging": true
        }
      }
    ]
  }
```

7.添加代码文件

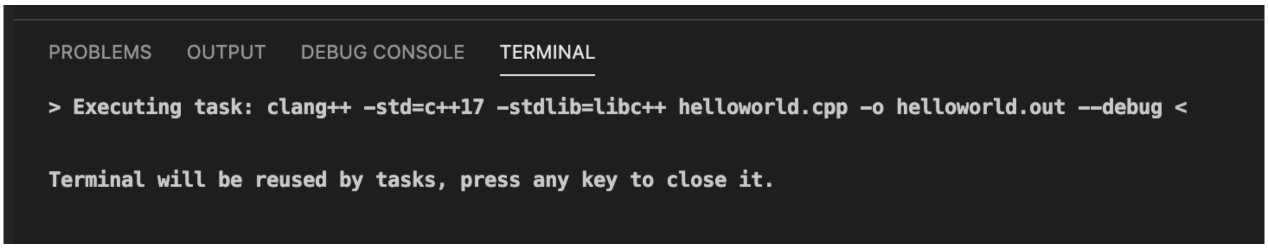
1.在主菜单中选择 **File > New File** 命名为 `helloworld.cpp`

2.粘贴代码，保存下

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

using namespace std;

int main()
{

    vector<string> msg {"Hello", "C++", "World", "from", "VS Code!"};

    for (const string& word : msg)
    {
        cout << word << " ";
    }
    cout << endl;
    return 0;
}
```

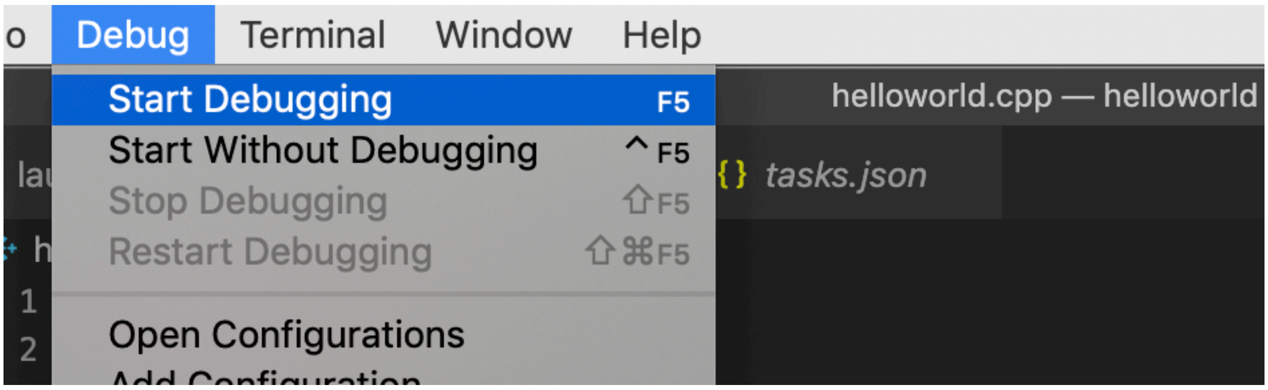
8.编译运行代码

在选中源代码文件的情况下按快捷键 `⇧⌘B`

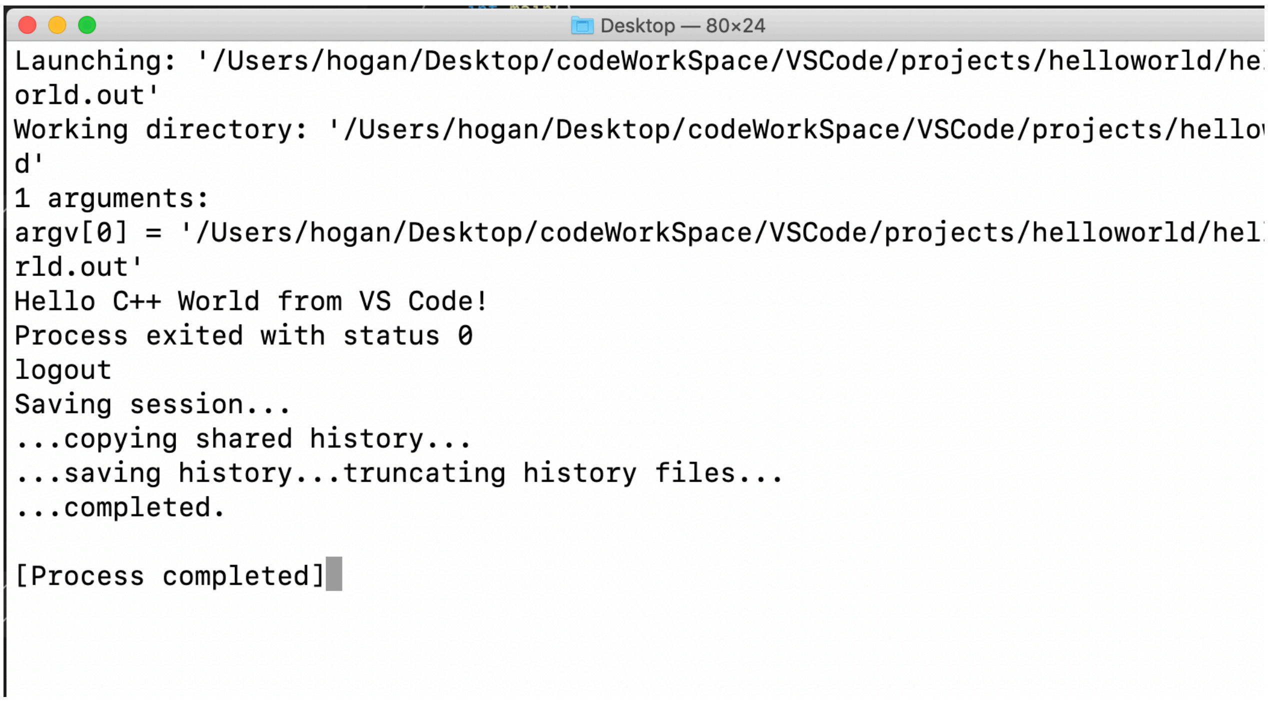


```
> Executing task: clang++ -std=c++17 -stdlib=libc++ helloworld.cpp -o helloworld.out --debug <

Terminal will be reused by tasks, press any key to close it.
```

菜单中选择，或者 直接按F5



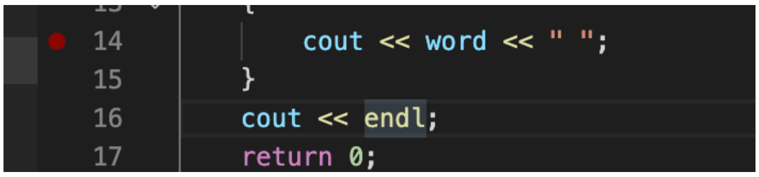
再按一次F5，继续运行，在新弹出的窗口中就可以看到运行结果了：



```
Launching: '/Users/hogan/Desktop/codeWorkSpace/VSCode/projects/helloworld/he
orld.out'
Working directory: '/Users/hogan/Desktop/codeWorkSpace/VSCode/projects/hello
d'
1 arguments:
argv[0] = '/Users/hogan/Desktop/codeWorkSpace/VSCode/projects/helloworld/hel
rld.out'
Hello C++ World from VS Code!
Process exited with status 0
logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

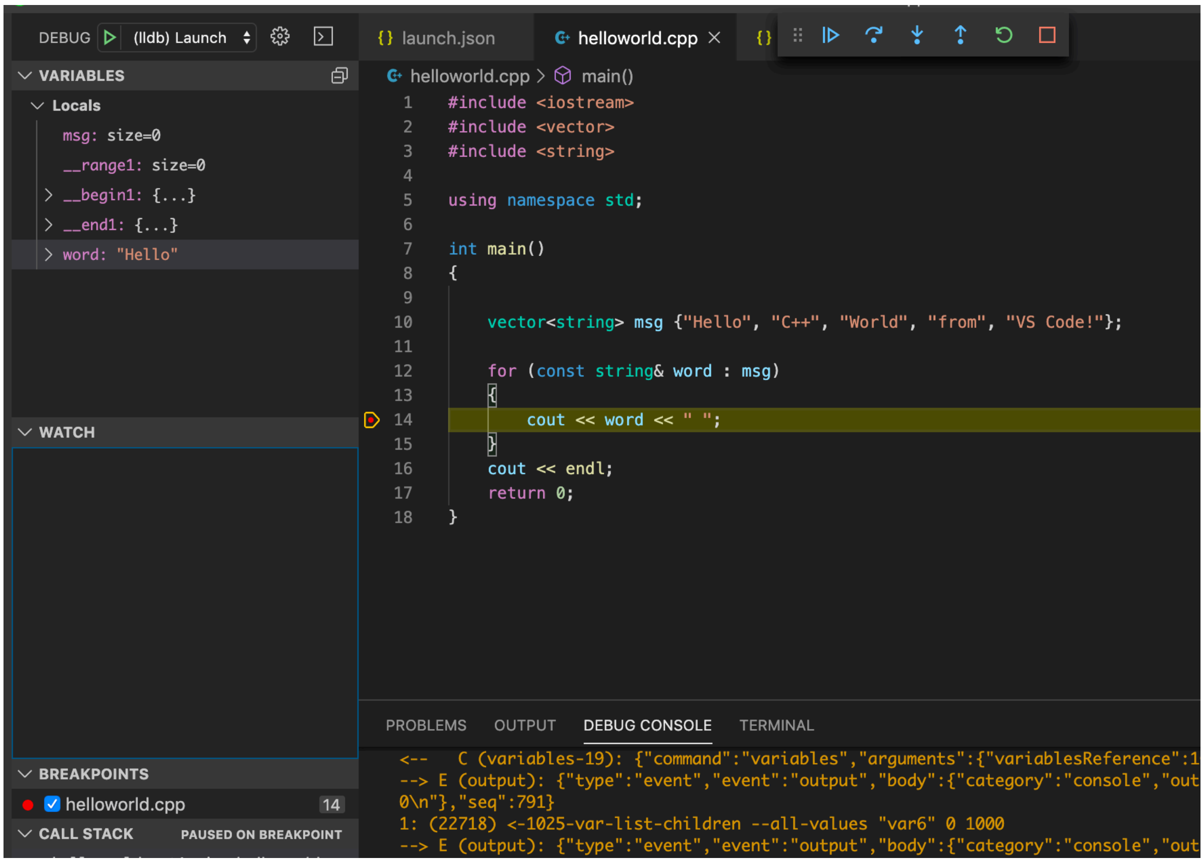
9.简单调试

1. 设置断点，行号前面鼠标单击下就行



```
        cout << word << " ";
    }
    cout << endl;
    return 0;
```

2. 简单信息查看：

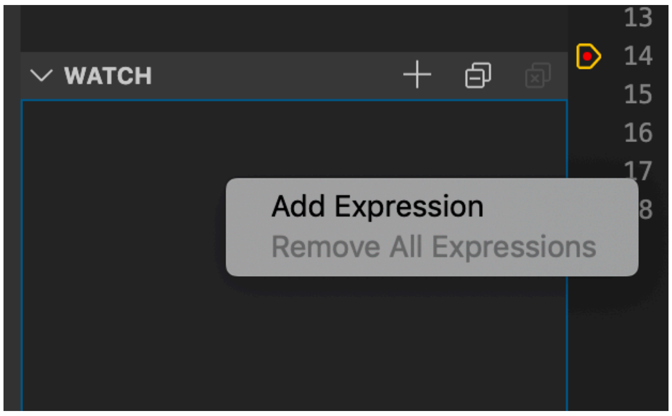


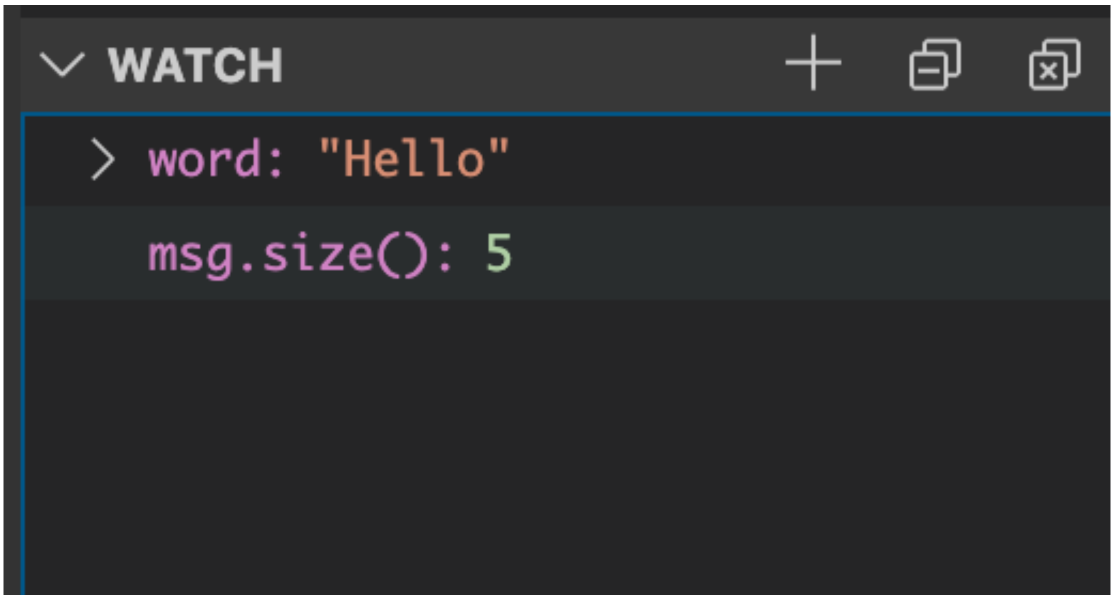
```
msg: size=0
__range1: size=0
> __begin1: {...}
> __end1: {...}
> word: "Hello"
```

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

using namespace std;

int main()
{

    vector<string> msg {"Hello", "C++", "World", "from", "VS Code!"};

    for (const string& word : msg)
    {
        cout << word << " ";
    }
    cout << endl;
    return 0;
}
```

```
<--   C (variables-19): {"command":"variables","arguments":{"variablesReference":1
--> E (output): {"type":"event","event":"output","body":{"category":"console","out
0\n"},"seq":791}
1: (22718) <-1025-var-list-children --all-values "var6" 0 1000
--> E (output): {"type":"event","event":"output","body":{"category":"console","out
```

3. 可以在watch中右键添加关注的变量,也可以输入函数表达式

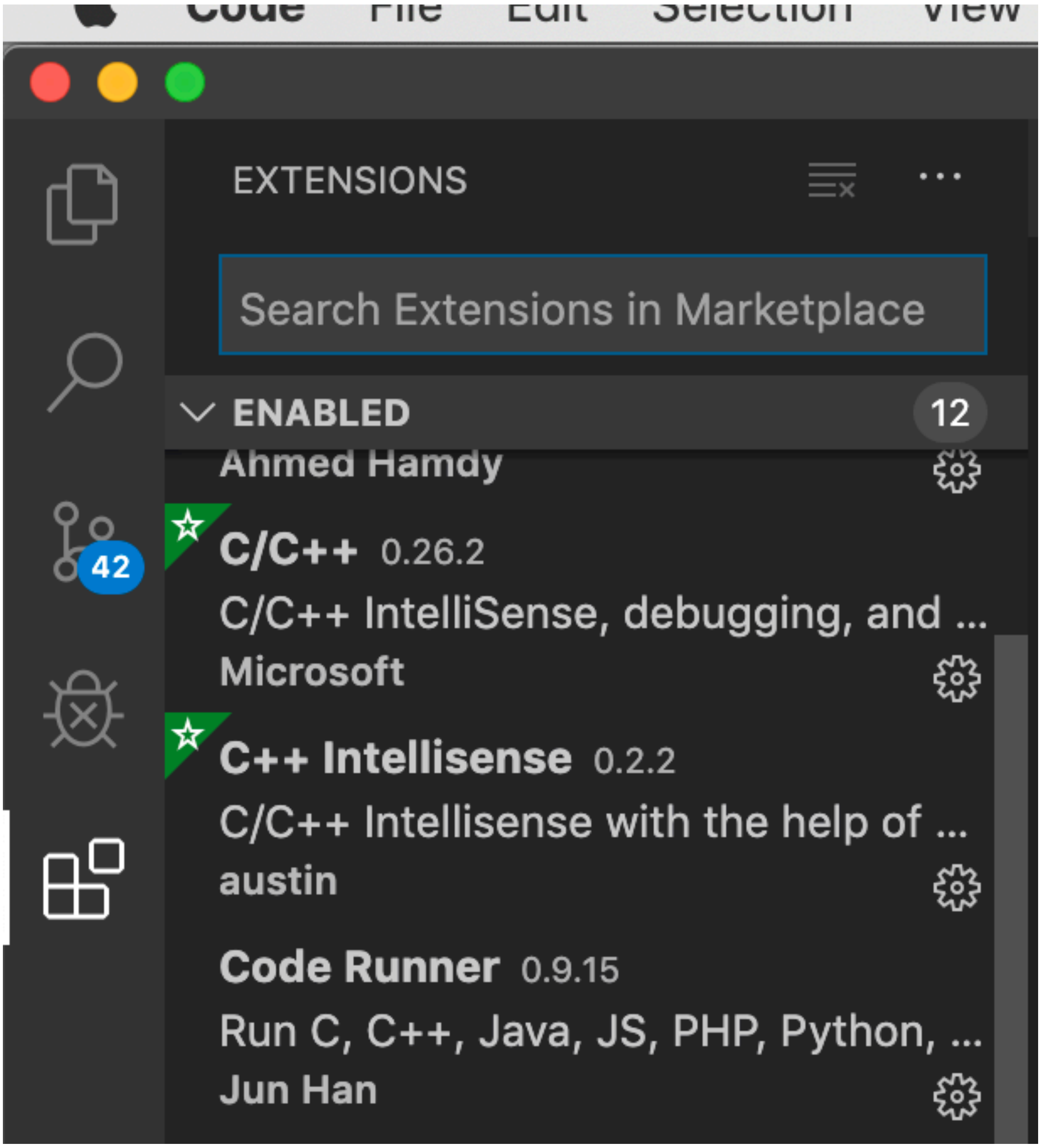




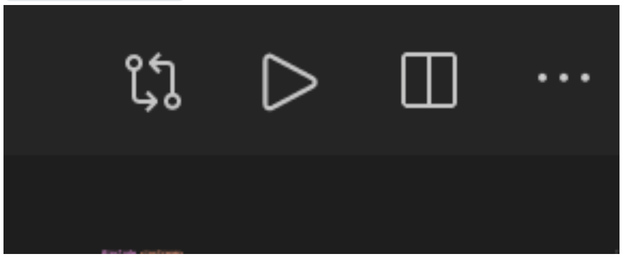
```
word: "Hello"
msg.size(): 5
```

10.插件安装

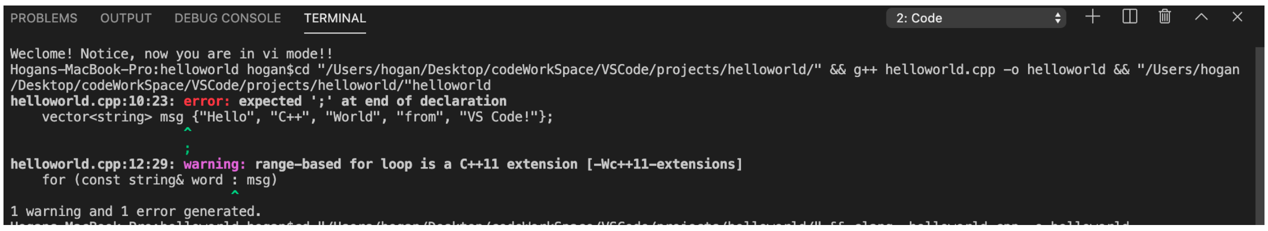
下面这几个插件装好之后，写代码就能有提示补全之类的功能了，


Code
File
Edit
Selection
View
EXTENSIONS
Search Extensions in Marketplace
ENABLED
12
42
Ahmed Hamdy
C/C++ 0.26.2
C/C++ IntelliSense, debugging, and ...
Microsoft
C++ Intellisense 0.2.2
C/C++ Intellisense with the help of ...
austin
Code Runner 0.9.15
Run C, C++, Java, JS, PHP, Python, ...
Jun Han

`Code Runner`装好之后，就可以直接写代码单击右上角的三角形符号

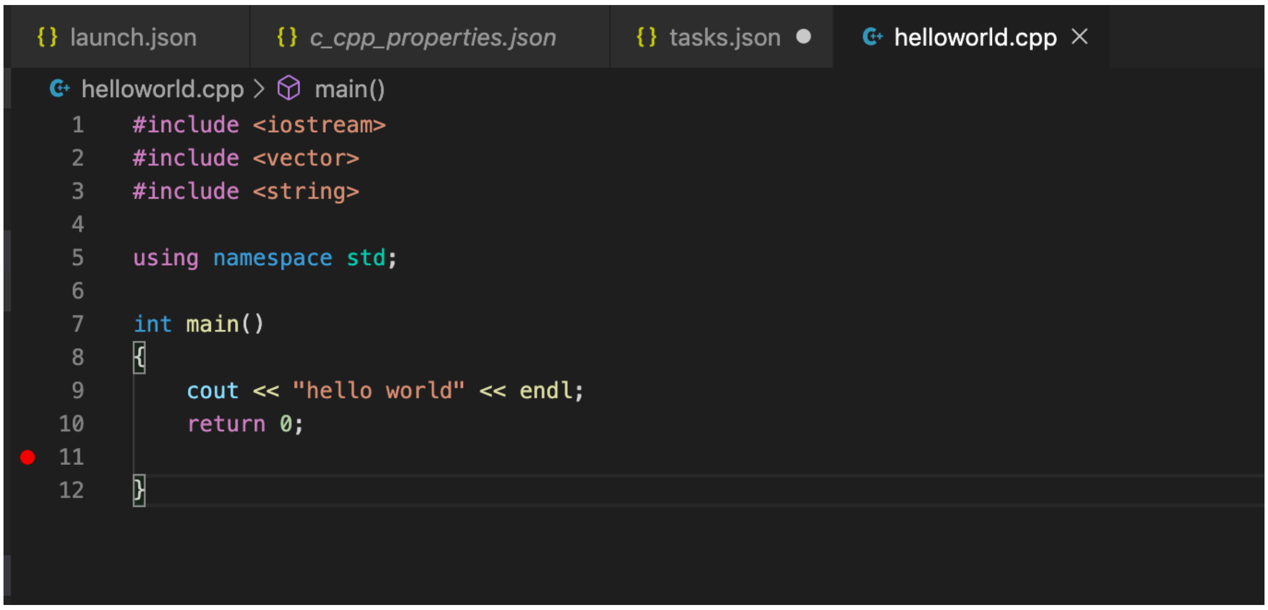


编译运行了,但是 code Runner 编译选项不能配置，所以就不能使用c++17 了



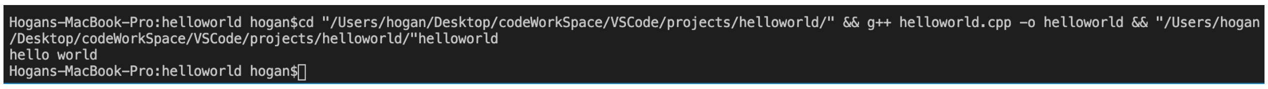
```
Weclome! Notice, now you are in vi mode!!
Hogans-MacBook-Pro:helloworld hogan$cd "/Users/hogan/Desktop/codeWorkSpace/VSCode/projects/helloworld/" && g++ helloworld.cpp -o helloworld && "/Users/hogan
/Desktop/codeWorkSpace/VSCode/projects/helloworld/"helloworld
helloworld.cpp:10:23: error: expected ';' at end of declaration
    vector<string> msg {"Hello", "C++", "World", "from", "VS Code!"};
                      ^
                      ;
helloworld.cpp:12:29: warning: range-based for loop is a C++11 extension [-Wc++11-extensions]
    for (const string& word : msg)
                            ^

1 warning and 1 error generated.
```

但是普通的程序还是可以直接运行的：



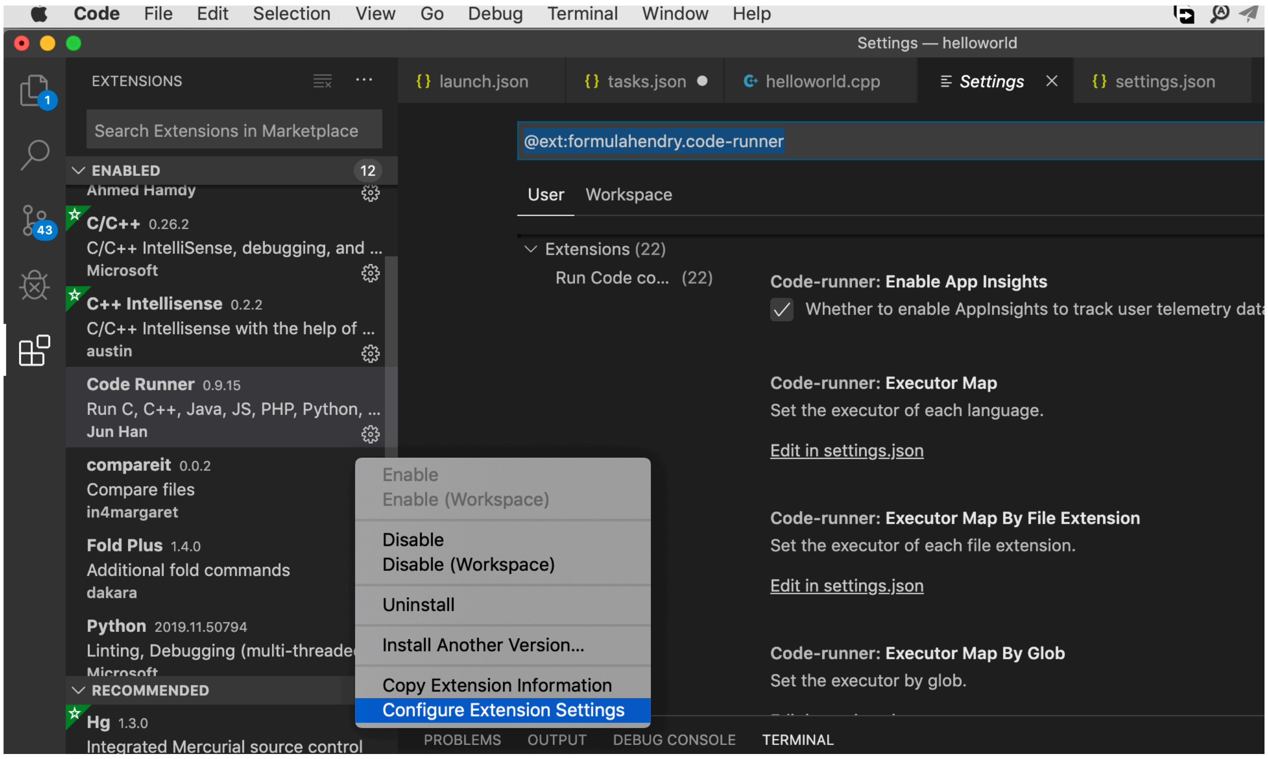
```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
{
    cout << "hello world" << endl;
    return 0;

}
```

运行结果：



```
Hogans-MacBook-Pro:helloworld hogan$cd "/Users/hogan/Desktop/codeWorkSpace/VSCode/projects/helloworld/" && g++ helloworld.cpp -o helloworld && "/Users/hogan
/Desktop/codeWorkSpace/VSCode/projects/helloworld/"helloworld
hello world
Hogans-MacBook-Pro:helloworld hogan$
```

codeRunner 要设置下集成到terminal中



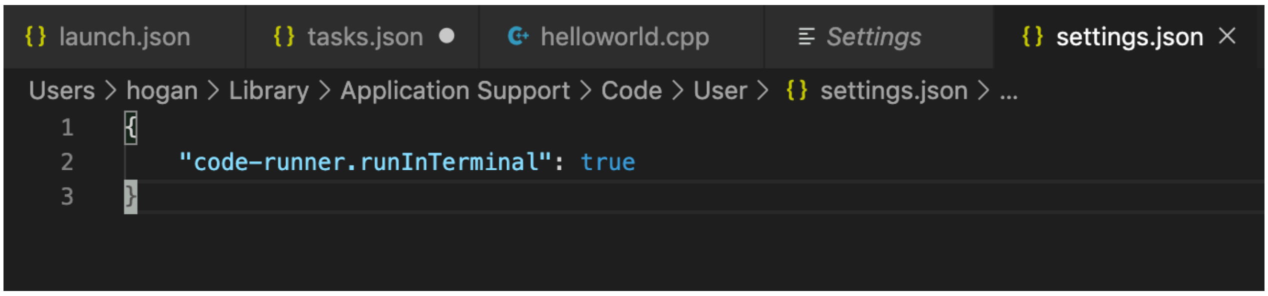
@ext:formulahendry.code-runner

Code-runner: Enable App Insights
✓ Whether to enable AppInsights to track user telemetry data

Code-runner: Executor Map
Set the executor of each language.

Code-runner: Executor Map By File Extension
Set the executor of each file extension.

Code-runner: Executor Map By Glob
Set the executor by glob.


launch.json
tasks.json
helloworld.cpp
Settings
settings.json
Users > hogan > Library > Application Support > Code > User > settings.json > ...

```json
{
    "code-runner.runInTerminal": true
}
```

注意点

mac 系统是 Mojave 10.14 之前用catalina10.15 打断点无法调试，各种google后来好像是bug不支持。。。

参考

> https://code.visualstudio.com/docs/cpp/config-clang-mac