工具篇：vscode 及 其他

Thursday, March 10, 2022 10:10 AM

Table of Contents

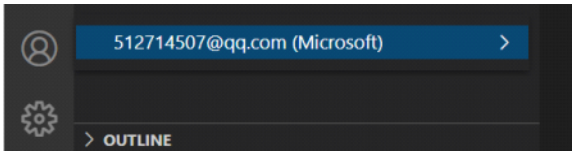
1.关于vscode (Back to Top)

好处：

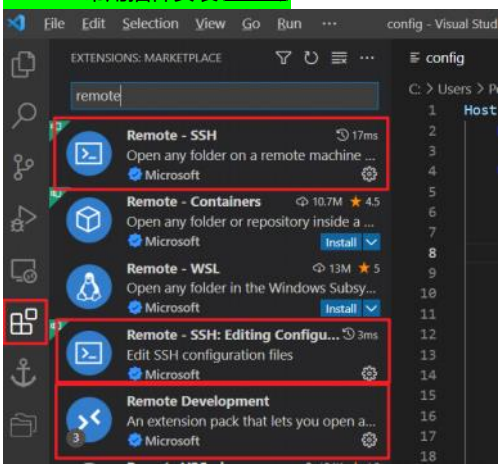
1. 丰富的插件
2. ssh连接服务器很方便

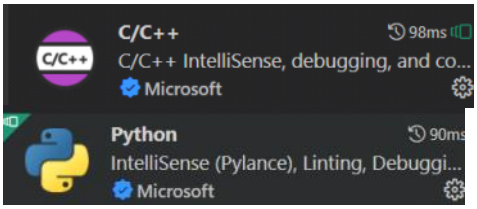
下载地址： https://code.visualstudio.com/download (linux / win)

建议：拥有一个微软账号或者github账号登录vscode来保存和更新自己的配置



2.vscode常用插件安装 (Back to Top)





Markdown 自动目录生成

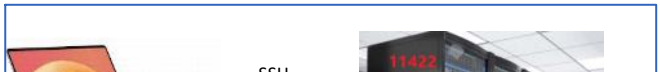
文件夹比较器

等等

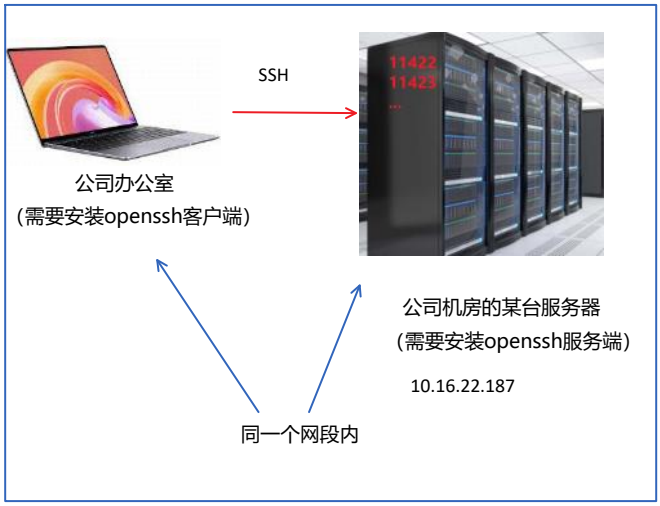
3.关于SSH (Back to Top)

什么是SSH (Back to Top)

一种安全的网络协议



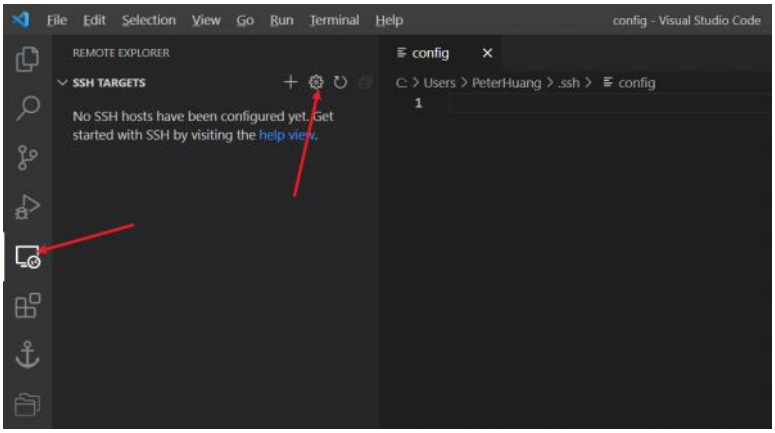
一种安全的网络协议

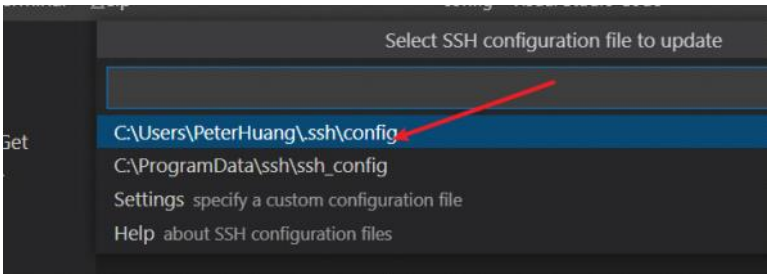


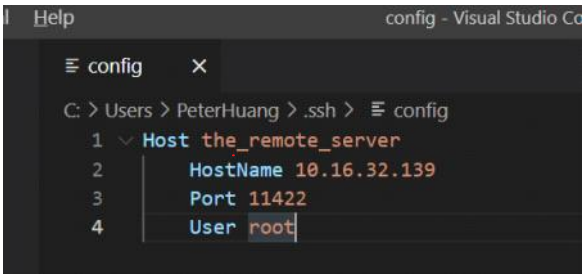
两种连接方式 (Back to Top)

a. 终端连接
b. vscode连接（上文提到的remote插件必须装好）

Vscode 如何进行ssh连接 (Back to Top)







文件传输(关于scp) (Back to Top)

下载:

上传:

```
scp P 9091 C:\Users\peterhuang\Desktop\hello.cpp root@10.16.100.48:/datav/shared/
```

4.高效的快捷键和自定义设置 (Back to Top)

4.1 快捷键 (Back to Top)

Ctrl P	快速查找并打开某个文件
Ctrl F	在当前文件里搜索字符串
Ctrl shift F	在当前工作目录里搜索字符串
Ctrl shift P	快速打开一些配置文件
Code 路径	快速打开某个文件夹并以此为工作空间
Ctrl + `	快速打开终端
Alt + /	快速切换到debug界面（如果在调试状态）
Ctrl + B	快速打开侧边栏

其他参考： https://betterprogramming.pub/15-useful-vscode-shortcuts-to-boost-your-productivity-415de3cb1910

4.2 设置alias (Back to Top)

5.高效调试配置(tasks.json 和 launch.json) (Back to Top)

5.1 配置launch.json (Back to Top)

Python 和 c++

{} launch.json × {} tasks.json

.vscode > {} launch.json > Launch Targets > {} LD_LIBRAYRY_PATH

```
{
    "version": "0.2.0",
    "configurations": [
        // python debug
        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            // "program": "main.py",
            "console": "integratedTerminal",
            "justMyCode": true // false 的话 你可以进入一些库的源码里面进行调试，比如说进入pytorch的一部分源码
        },
        // c++ debug
        {
            "name": "C++ file",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/workspace/pro", // 你要调试的文件，这里指的是cpp最终生成的可执行文件
            "args": [],
                    "environment": [{"name": "LD_LIBRAYRY_PATH", "value": "$(LD_LIBRAYRY_PATH):/mypath/to/lib/"}],
                    // 相当于直接 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mypath/to/lib/
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}/workspace", // c++在运行过程时会在这寻找依赖和其他文件（比如说 图片）
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "/usr/bin/gdb",
            "setupCommands": [
                {
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "build" // 在运行launch之前先运行tasks.json里的东西
        }
    ]
}
```

Add Configurati

详细参考：
https://code.visualstudio.com/docs/editor/debugging

5.2 配置tasks.json (Back to Top)

```
{} launch.json    {} tasks.json ×
.vscode > {} tasks.json > ...
1  {
2      "version": "2.0.0",
3      "tasks": [
4          {
5              "label": "build",
6              "type": "shell",
7              "command": "make pro -j6"
8          }
9      ]
10 }
```

每次运行launch之前都会运行tasks （这里指的是都会编译一遍）

5.3配置settings.json (Back to Top)

在.vscode/settings.json中配置"*.cu": "cuda-cpp"可以实现对cuda的语法解析

```
{} c_cpp_properties.json    {} launch.json    {} settings.json ×
1.1-hello-tensorrt > .vscode > {} settings.json > {} files.associations
1  {
2      "files.associations": {
3          "*.cpp": "cpp",
4          "*.cu": "cuda-cpp",
5          "deque": "cpp",
6          "string": "cpp",
7          "vector": "cpp",
```

5.4 配置c_cpp_properties.json (Back to Top)

```
{} c_cpp_properties.json ×    {} launch.json    {} settings.json
1.1-hello-tensorrt > .vscode > {} c_cpp_properties.json > [ ] configurations > {} 0 > intelliSenseMode
1  {
2      "configurations": [
3          {
4              "name": "Linux",
5              "includePath": [
6                  "${workspaceFolder}/**",
7                  "/datav/software/anaconda3/lib/python3.9/site-packages/trtpy/trt8cuda112cudnn8/include/**",
8                  "/datav/software/anaconda3/lib/python3.9/site-packages/trtpy/cpp-packages/opencv4.2/include/**"
9              ], // 让 intelliSense engine (语法提示, 自动补全) 能够找到所有需要的头文件
10             "compilerPath": "/usr/bin/gcc",
11             "cStandard": "gnu11",
12             "cppStandard": "gnu++11",          ** 指的是递归查询，除了当前目录，多少级目录都往下找
13             "intelliSenseMode": "linux-gcc-x64"
14         }
15     ],
16     "version": 4
17 }
```

ref:
https://code.visualstudio.com/docs/cpp/c-cpp-properties-schema-reference#:~:text=includePath%20An%20include%20path%20is%20a%20folder%20that%20contains%20header%20files

5.4 C++ Python 并行调试 (Back to Top)

在launch中配置好python 和c++的debug配置如下

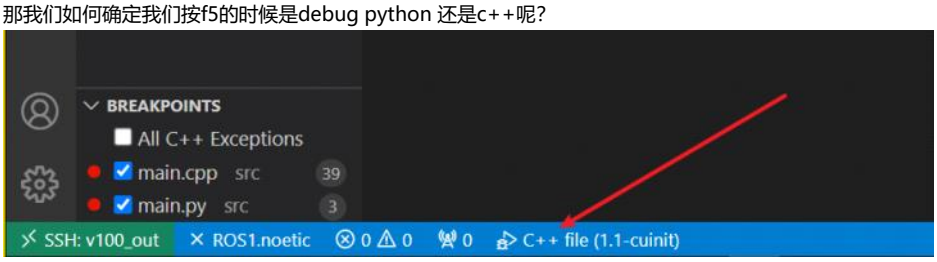
main.cpp main.py launch.json

.vscode > launch.json > Launch Targets > {} Python file

```
"configurations": [
    // python debug
    {
        "name": "Python file",
        "type": "python",
        "request": "launch",
        // "program": "${file}",
        "program": "src/main.py",
        "console": "integratedTerminal",
        "justMyCode": true // false 的话 你可以进入一些库的源码里面进行调试，比如说进入pytorch的一部分源码
    },
    // c++ debug
    {
        "name": "C++ file",
        "type": "cppdbg",
        "request": "launch",
        "program": "${workspaceFolder}/workspace/pro", // 你要调试的文件，这里指的是cpp最终生成的可执行文件
        "args": [],
                "environment": [{"name": "LD_LIBRAYRY_PATH", "value": "$(LD_LIBRAYRY_PATH):/mypath/to/lib/"}]
                // 相当于直接 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mypath/to/lib/
        "stopAtEntry": false,
        "cwd": "${workspaceFolder}/workspace", // c++在运行过程时会在这寻找依赖和其他文件（比如说 图片）
        "externalConsole": false,
        "MIMode": "gdb",
        "miDebuggerPath": "/usr/bin/gdb",
        "setupCommands": [
            {
                "text": "-enable-pretty-printing",
                "ignoreFailures": true
            }
        ],
```

Add Configu

那我们如何确定我们按f5的时候是debug python 还是c++呢?



可以通过这个来选择debug python 还是c++ file