8 MiniSQL项目框架与环境配置

框架链接：ZJU GitLab链接，请使用内网访问

#0 框架维护日志（更新中）

#1 代码框架介绍

本实验基于CMU-15445 BusTub框架，并做了一些修改和扩展。注意：为了避免代码抄袭，请不要将自己的代码发布到任何公共平台中。

#1.1 编译&开发环境

- `apple clang`: 11.0+ (MacOS)，使用`gcc --version`和`g++ --version`查看
- `gcc`&`g++`: 8.0+ (Linux)，使用`gcc --version`和`g++ --version`查看
- `cmake`: 3.16+ (Both)，使用`cmake --version`查看
- `gdb`: 7.0+ (Optional)，使用`gdb --version`查看
- `flex`&`bison`(暂时不需要安装，但如果需要对SQL编译器的语法进行修改，需要安装)

#1.2 构建

#1.2.1 Windows

目前该代码暂不支持在Windows平台上的编译。但在Win10及以上的系统中，可以通过安装WSL（Windows的Linux子系统）来进行开发和构建。WSL请选择Ubuntu子系统（推荐Ubuntu20及以上）。如果你使用Clion作为IDE，可以在Clion中配置WSL从而进行调试，具体请参考链接Clion with WSL。

#1.2.2 MacOS & Linux & WSL

基本构建命令

```
mkdir build
cd build
cmake ..
make -j
```

若不涉及到`CMakeLists`相关文件的变动且没有新增或删除`.cpp`代码（通俗来说，就是只是对现有代码做了修改）则无需重新执行`cmake..`命令，直接执行`make -j`编译即可。默认以`debug`模式进行编译，如果你需要使用`release`模式进行编译：

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

#1.3 测试

在构建后，默认会在`build/test`目录下生成`minisql_test`的可执行文件，通过`./minisql_test`即可运行所有测试。如果需要运行单个测试，例如，想要运行`lru_replacer_test.cpp`对应的测试文件，可以通过`make lru_replacer_test`命令进行构建。

#1.4 工程目录

- `src`：与MiniSQL工程相关的头文件和源代码。`src/include`中为MiniSQL各个子模块的头文件，`src/buffer`、`src/record`、`src/index`、`src/catalog`等目录为MiniSQL各个子模块的源代码。
- `test`：与测试用例相关的源代码和头文件。
- `thirdparty`：第三方库，包括日志模块`glog`和测试模块`gtest`。

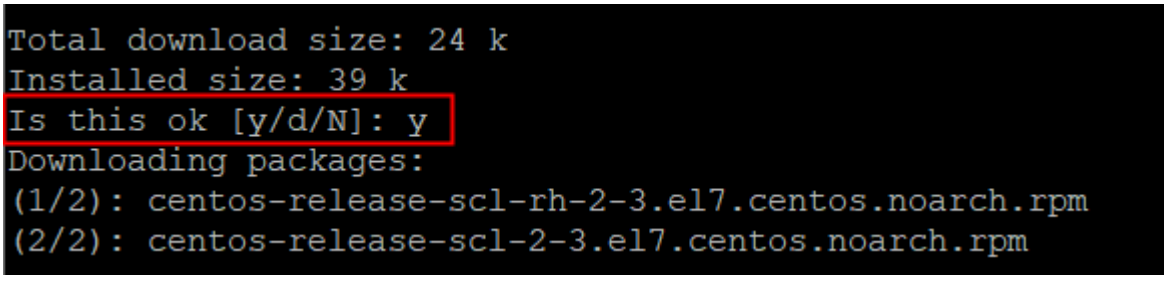
#2 使用WSL-Ubuntu进行开发

Note：Win10系统下，参考Win10系统安装WSL教程安装WSL，选择Ubuntu子系统即可，推荐选用Ubuntu 20.04以上的版本（示例使用的是20.04版本）。

#2.1 配置编译环境

首次安装时，请使用`sudo apt-get update`更新软件源。然后使用命令`sudo apt install gcc g++ cmake gdb`安装编译和调试环境。

Note: 安装时一般都会提示是否需要安装，输入`y`回车即可：



安装完成后，通过`--version`查看是否安装完成，如下图所示：

```
ying@DESKTOP-COSICRG:~$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ying@DESKTOP-COSICRG:~$ g++ --version
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ying@DESKTOP-COSICRG:~$ cmake --version
cmake version 3.16.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
ying@DESKTOP-COSICRG:~$ gdb --version
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

进入一个目录，从远程仓库中克隆代码到该目录下（建议先Fork到自己小组的私有仓库然后再进行克隆），在这里我选择将代码克隆到`/mnt/f`目录下（当然这个目录可以根据你的需要自由选择），`/mnt/f`目录实际上就是我们电脑本地磁盘的`F`盘：

```
cd /mnt/f
git clone https://git.zju.edu.cn/zjucsdb/minisql.git
```

Note：如果在克隆过程中提示`Permission Denied`，请在命令前面加上`sudo`以执行：

```
ying@DESKTOP-COSICRG:/mnt/f$ git clone https://git.zju.edu.cn/zjucsdb/minisql.git
Cloning into 'minisql'...
error: chmod on /mnt/f/minisql/.git/config.lock failed: Operation not permitted
fatal: could not set 'core.filemode' to 'false'
```

然后进入目录，进行构建或测试：

```
# 进入目录
cd /mnt/f/minisql
# 建立并进入build目录
mkdir build
cd build
# 生成Makefile
sudo cmake ..
# 多线程编译生成可执行文件，-j可以指定具体的线程数，如-j4就是使用4线程编译
make -j
```

`cmake`构建成功后如下图所示：

```
-- Create test suit: buffer_pool_manager_test
-- Create test suit: lru_replacer_test
-- Create test suit: catalog_test
-- Create test suit: b_plus_tree_index_test
-- Create test suit: b_plus_tree_test
-- Create test suit: index_iterator_test
-- Create test suit: index_roots_page_test
-- Create test suit: tuple_test
-- Create test suit: disk_manager_test
-- Create test suit: table_heap_test
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/f/minisql/build
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$ _
```

`make`构建成功后如下图所示：

```
Scanning dependencies of target main
Scanning dependencies of target minisql_test
[ 80%] Building CXX object bin/CMakeFiles/main.dir/main.cpp.o
[ 81%] Building CXX object test/CMakeFiles/minisql_test.dir/buffer/buffer_pool_manager_test.cpp.o
[ 83%] Building CXX object test/CMakeFiles/minisql_test.dir/buffer/lru_replacer_test.cpp.o
[ 85%] Building CXX object test/CMakeFiles/minisql_test.dir/index/b_plus_tree_index_test.cpp.o
[ 85%] Building CXX object test/CMakeFiles/minisql_test.dir/catalog/catalog_test.cpp.o
[ 87%] Building CXX object test/CMakeFiles/minisql_test.dir/page/index_roots_page_test.cpp.o
[ 89%] Building CXX object test/CMakeFiles/minisql_test.dir/index/b_plus_tree_test.cpp.o
[ 89%] Building CXX object test/CMakeFiles/minisql_test.dir/index/index_iterator_test.cpp.o
[ 92%] Building CXX object test/CMakeFiles/minisql_test.dir/storage/disk_manager_test.cpp.o
[ 92%] Building CXX object test/CMakeFiles/minisql_test.dir/record/tuple_test.cpp.o
[ 93%] Building CXX object test/CMakeFiles/minisql_test.dir/storage/table_heap_test.cpp.o
[ 94%] Building CXX object test/CMakeFiles/minisql_test.dir/main_test.cpp.o
[ 96%] Linking CXX static library ../lib/libgtest_main.a
[ 97%] Linking CXX shared library libminisql_test_main.so
[ 98%] Linking CXX executable main
[ 98%] Built target gtest_main
[ 98%] Built target minisql_test_main
[ 98%] Built target main
[100%] Linking CXX executable minisql_test
[100%] Built target minisql_test
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$ _
```

编译生成的可执行文件位于`bin/`和`test/`（测试相关文件）下：

```
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$ ll
total 80
drwxrwxrwx 1 root root  4096 Apr 11 00:20 ./
drwxrwxrwx 1 root root  4096 Apr 11 00:18 ../
-rwxrwxrwx 1 root root 30577 Apr 11 00:19 CMakeCache.txt*
drwxrwxrwx 1 root root  4096 Apr 11 00:21 CMakeFiles/
-rwxrwxrwx 1 root root  3454 Apr 11 00:18 CPackConfig.cmake*
-rwxrwxrwx 1 root root  3882 Apr 11 00:18 CPackSourceConfig.cmake*
-rwxrwxrwx 1 root root   328 Apr 11 00:19 CTestTestfile.cmake*
-rwxrwxrwx 1 root root 35414 Apr 11 00:19 Makefile*
drwxrwxrwx 1 root root  4096 Apr 11 00:21 bin/
-rwxrwxrwx 1 root root  1830 Apr 11 00:19 cmake_install.cmake*
drwxrwxrwx 1 root root  4096 Apr 11 00:21 glog-build/
drwxrwxrwx 1 root root  4096 Apr 11 00:19 googletest-build/
drwxrwxrwx 1 root root  4096 Apr 11 00:21 lib/
drwxrwxrwx 1 root root  4096 Apr 11 00:21 test/
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$
```

最终整个MiniSQL的主程序在这里：

```
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$ ./bin/main
minisql > quit;
bye!
```

一个运行测试用例的例子：

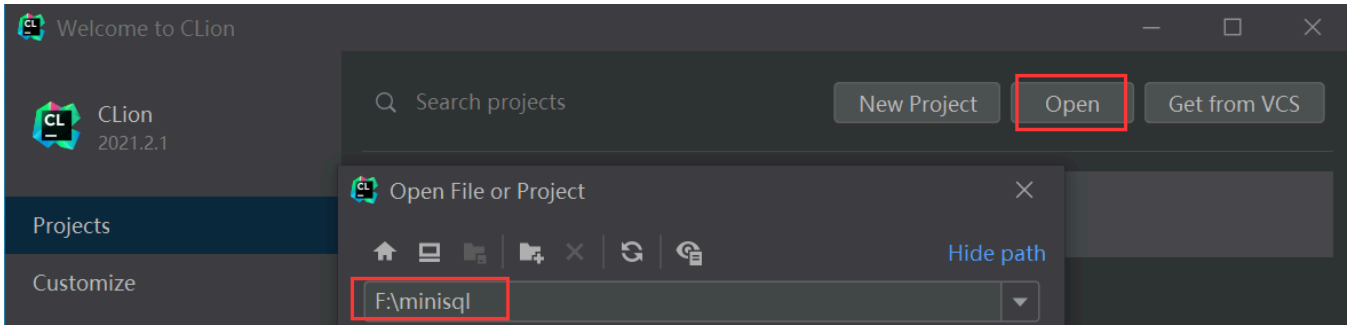
```
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$ make lru_replacer_test
[ 16%] Built target glogbase
[ 19%] Built target glog
[ 23%] Built target gtest
[ 28%] Built target minisql_test_main
[ 95%] Built target minisql_shared
Scanning dependencies of target lru_replacer_test
[ 97%] Building CXX object test/CMakeFiles/lru_replacer_test.dir/buffer/lru_replacer_test.cpp.o
[100%] Linking CXX executable lru_replacer_test
[100%] Built target lru_replacer_test
ying@DESKTOP-COSICRG:/mnt/f/minisql/build$ ./test/lru_replacer_test
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from LRUReplacerTest
[ RUN      ] LRUReplacerTest.SampleTest
/mnt/f/minisql/test/buffer/lru_replacer_test.cpp:15: Failure
Expected equality of these values:
  6
  lru_replacer.Size()
    Which is: 0
/mnt/f/minisql/test/buffer/lru_replacer_test.cpp:20: Failure
Expected equality of these values:
  1
  value
    Which is: 32767
/mnt/f/minisql/test/buffer/lru_replacer_test.cpp:22: Failure
Expected equality of these values:
  2
  value
    Which is: 32767
```

Note：此处运行测试遇到`Failed`是正常现象。

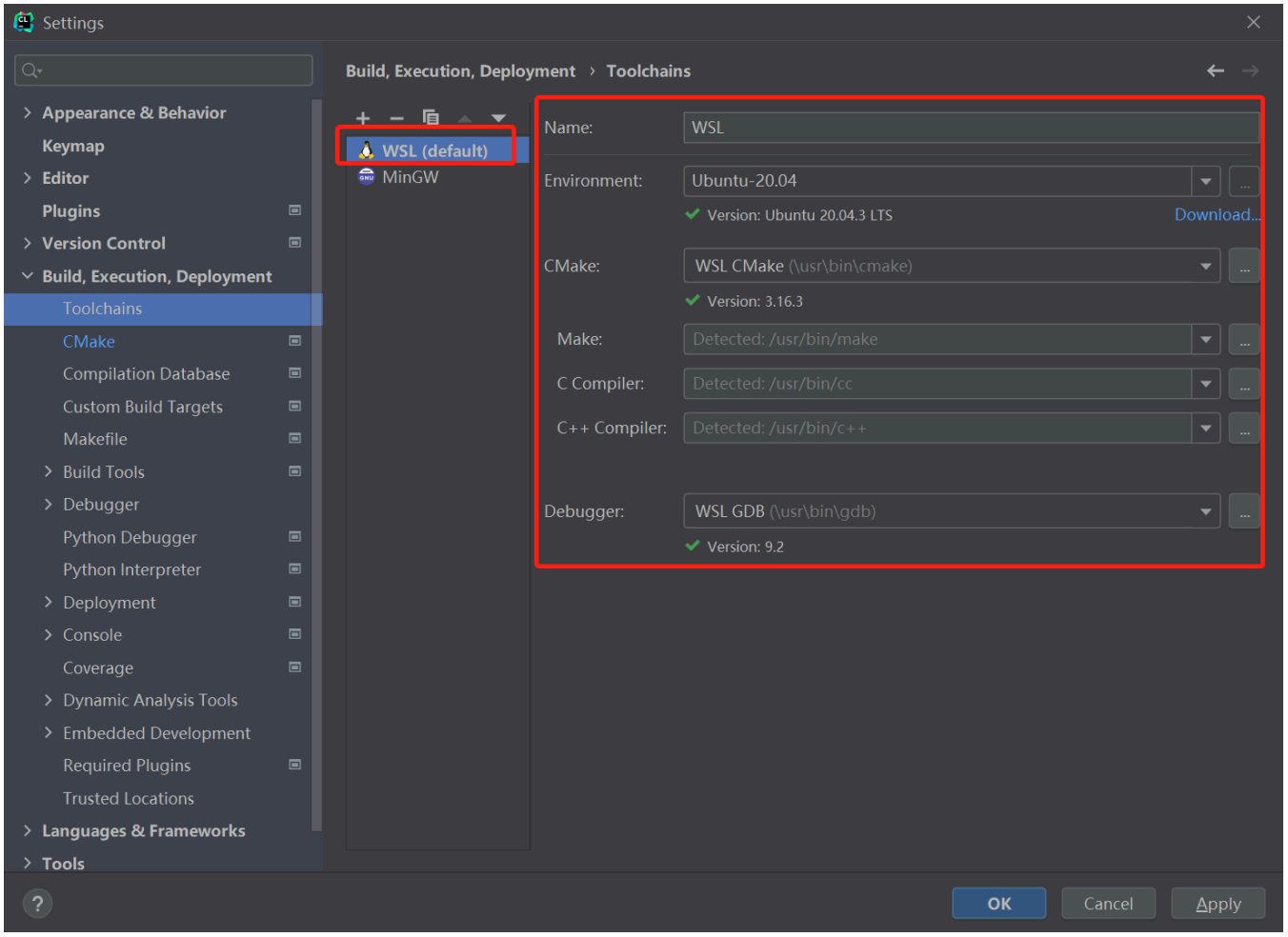
#2.2 使用Clion连接WSL进行开发

Note：[Clion下载网址](), 开始时有30天的免费试用期。在使用ZJU的邮箱进行认证后可以一直免费使用。

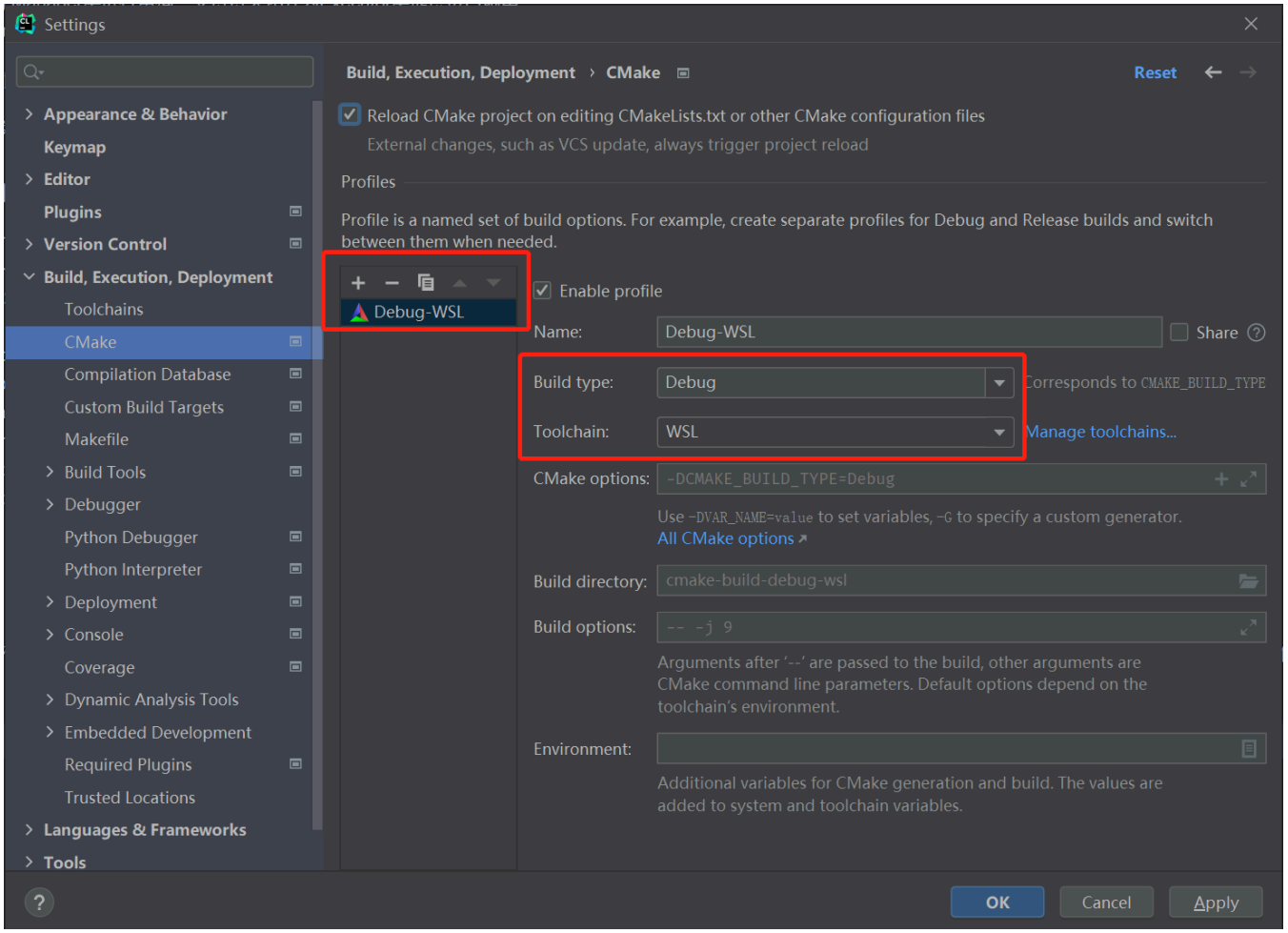
打开Clion后，导入MiniSQL项目：



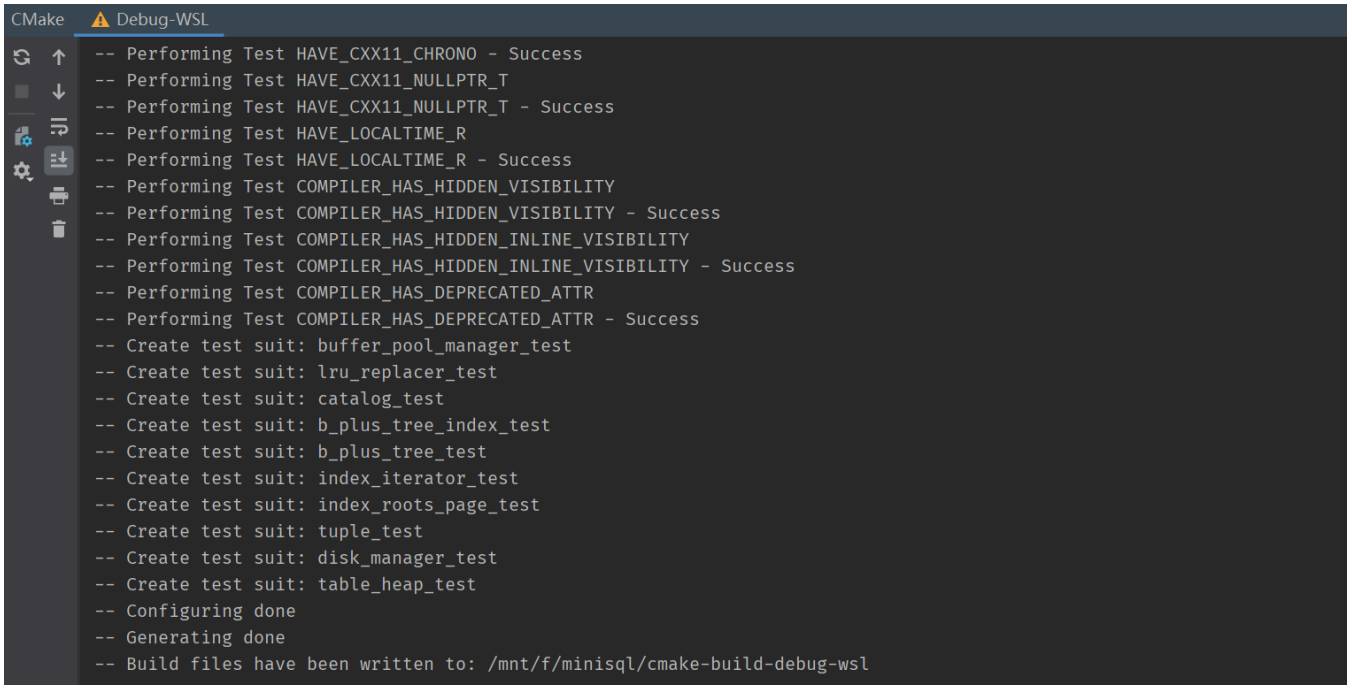
File-->Settings-->Build-->Toolchains 添加WSL相关设置：



File-->Settings-->Build-->CMake中添加CMake相关设置：



保存后会自动运行CMake命令，或是通过下图左边刷新按钮运行。CMake构建成功后如下图所示：



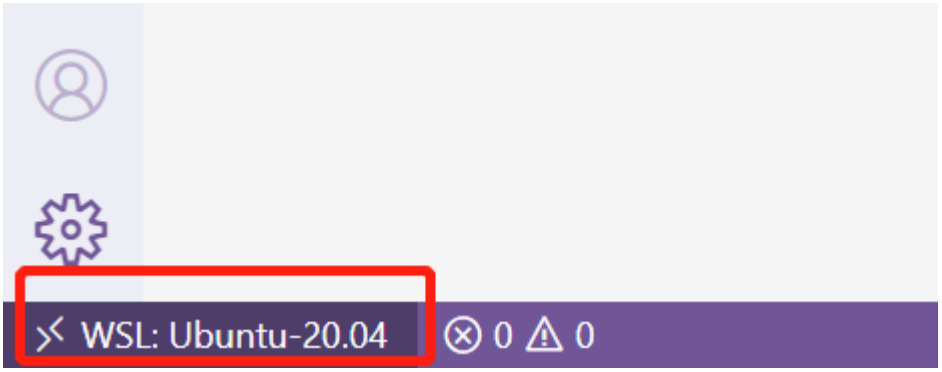
接下来就可以使用Clion进行开发和使用。

#2.3 使用VSCode连接WSL进行开发

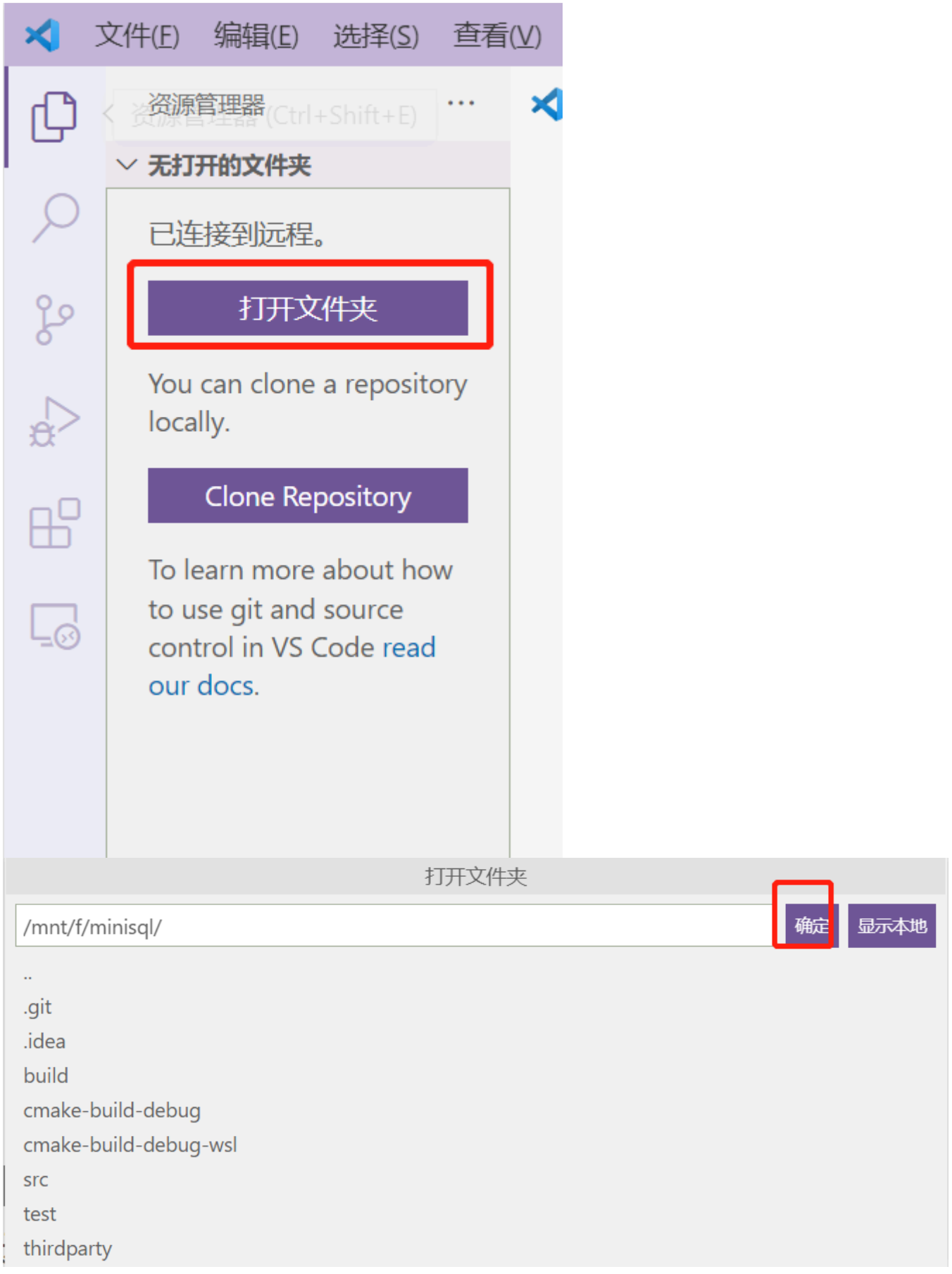
Note：VSCode需要事先在扩展`Ctrl+Shift+X`中安装以下插件：

- Remote-WSL（在本地安装）
- C/C++ （连接上WSL后再安装，安装在WSL）
- CMake Tools（连接上WSL后再安装，安装在WSL）

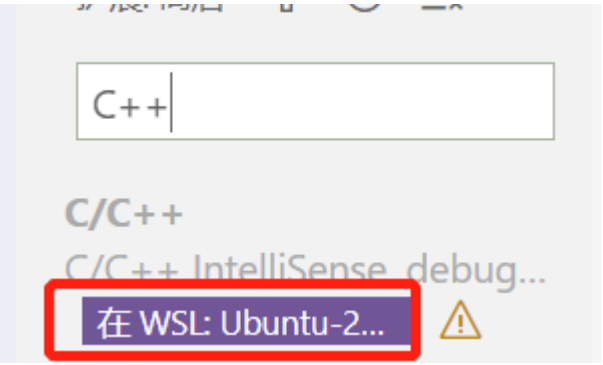
然后使用`Ctrl+Shift+P`打开选项卡输入`WSL`，选择`Remote-WSL:New Window`即可打开WSL。可以看到，左下角已连接的Linux子系统WSL:Ubuntu-20.04。



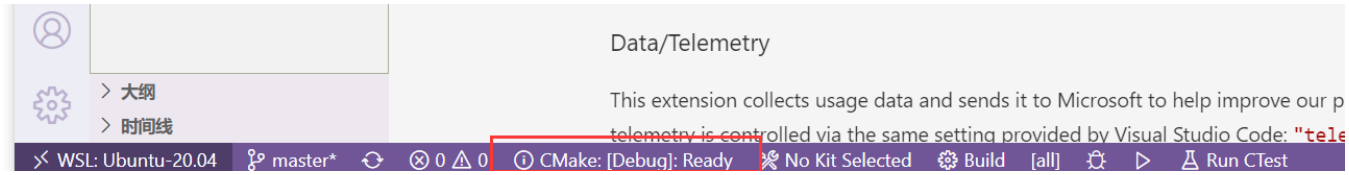
然后打开源代码所在目录：



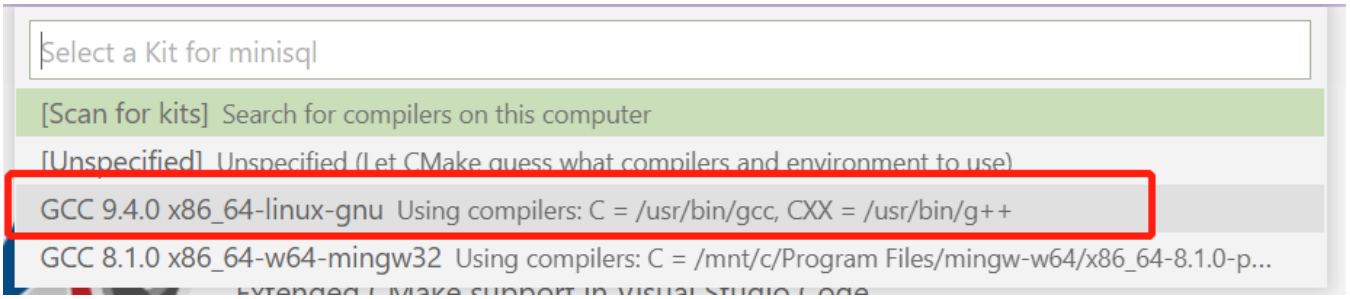
在WSL中安装C/C++和CMake插件：



安装成功后可以看到下面的工具，对CMake进行Configure：



选择WSL中的编译器：



构建完成后：

点击`Build`即可生成可执行文件。

#3 使用MacOS进行开发

MacOS中一般自带Apple Clang，可以使用`g++ --version`和`gcc --version`查看，`cmake`和`gdb`可以通过`brew`命令安装（或是从官网上下载然后添加到环境变量中）。在MacOS中可以使用Clion和VS Code直接在本地进行开发调试，方法与#2中提到的类似。

#4 远程连接服务器进行开发

同学们可以根据自己服务器选择合适的Linux镜像，推荐选用Ubuntu 20.04+或是CentOS 7.2+的镜像。若选用Ubuntu的镜像，请参考#2中的教程进行编译环境的配置，然后参考#4.2连接远程服务器进行开发调试。**在本节的示例中，服务器镜像选用的是CentOS 7.2**。

Note: 由于外网服务器正常情况下无法访问位于学校内网的`ZJU GitLab`，一个可行的解决办法是，先从`ZJU GitLab`上克隆源代码到本地，然后将代码推送到自己小组**私有**的远程`GitHub`仓库中，这样外网服务器就可以通过`GitHub`存储库访问到代码。另外一种可行的方法是，通过`scp`命令将源代码直接上传到远程服务器中，然后在远程服务器中新建`Git`仓库。

#4.1 配置编译环境

本节以**CentOS 7.2镜像**进行示例。对于其它镜像，配置编译环境的方法类似，可以自行网上搜索在该类型的系统镜像中如何安装`GCC`、`G++`、`CMake`和`GDB`。

服务器镜像中如果自带`GCC`、`G++`、`CMake`和`GDB`，但版本较低的（如下图中`GCC`的版本是4.8.5），则需要对相应的软件进行升级（具体升级教程可上网查找）。

```
[root@ecs-0000 ~]# gcc --version
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

CentOS 7.2镜像中自带了`GCC`（但未自带`G++`），在这里简单叙述一下CentOS 7.2升级`GCC`、`G++`的方法，升级方法参考引用链接，命令如下：

```
# 安装centos-release-scl
sudo yum install centos-release-scl
# 安装devtoolset
sudo yum install devtoolset-9-gcc*
sudo yum install devtoolset-9-g++*
# 替换旧的gcc和g++
mv /usr/bin/gcc /usr/bin/gcc-4.8.5
ln -s /opt/rh/devtoolset-9/root/bin/gcc /usr/bin/gcc
mv /usr/bin/g++ /usr/bin/g++-4.8.5  # Note: 如果CentOS中没有自带g++,
                                    # 即g++ --version提示命令不存在,
                                    # 则不需要执行该步命令，只需要执行下面的ln即可。
ln -s /opt/rh/devtoolset-9/root/bin/g++ /usr/bin/g++
```

Note: 安装时一般都会提示是否需要安装，输入`y`回车即可：

```
Total download size: 24 k
Installed size: 39 k
Is this ok [y/d/N]: y
Downloading packages:
(1/2): centos-release-scl-rh-2-3.el7.centos.noarch.rpm
(2/2): centos-release-scl-2-3.el7.centos.noarch.rpm
```

升级完成后，查看`GCC`和`G++`是否正确安装：

```
[root@ecs-0000 ~]# gcc --version
gcc (GCC) 9.3.1 20200408 (Red Hat 9.3.1-2)
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[root@ecs-0000 ~]# g++ --version
g++ (GCC) 9.3.1 20200408 (Red Hat 9.3.1-2)
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

由于CentOS 7.2中通过`yum`源安装的`CMake`版本较老（2.X版本），因此需要从官网下载，下载链接：[CMake Download](), 根据不同的CPU架构，选择不同的链接下载，上面的是X86架构，下面的是ARM架构：

Linux x86_64	cmake-3.23.0-linux-x86_64.sh
	cmake-3.23.0-linux-x86_64.tar.gz
Linux aarch64	cmake-3.23.0-linux-aarch64.sh
	cmake-3.23.0-linux-aarch64.tar.gz

然后通过`wget`命令进行下载：

```
# X86架构
wget https://github.com/Kitware/CMake/releases/download/v3.23.0/cmake-3.23.0-linux-x86_64.tar.gz
# ARM架构
wget https://github.com/Kitware/CMake/releases/download/v3.23.0/cmake-3.23.0-linux-aarch64.tar.gz
```

下载完成后：

```
# 解压压缩包
tar -xzvf cmake-3.23.0-linux-aarch64.tar.gz #ARM架构下使用该命令
tar -xzvf cmake-3.23.0-rc2-linux-x86_64.tar.gz #X86架构下使用该命令
# 重命名
mv cmake-3.23.0-linux-aarch64 cmake #ARM架构下使用该命令
mv cmake-3.23.0-rc2-linux-x86_64 cmake #X86架构下使用该命令
# 移动 & 链接
mv cmake /usr/local/
ln -s /usr/local/cmake/bin/cmake /usr/bin/cmake
```

使用`cmake --version`即可查看`CMake`是否安装成功：

```
[root@ecs-0000 ~]# cmake --version
cmake version 3.23.0-rc2

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[root@ecs-0000 ~]#
```

调试工具`GDB`使用`sudo yum install gdb`命令直接安装即可，然后使用`gdb --version`查看是否安装成功：

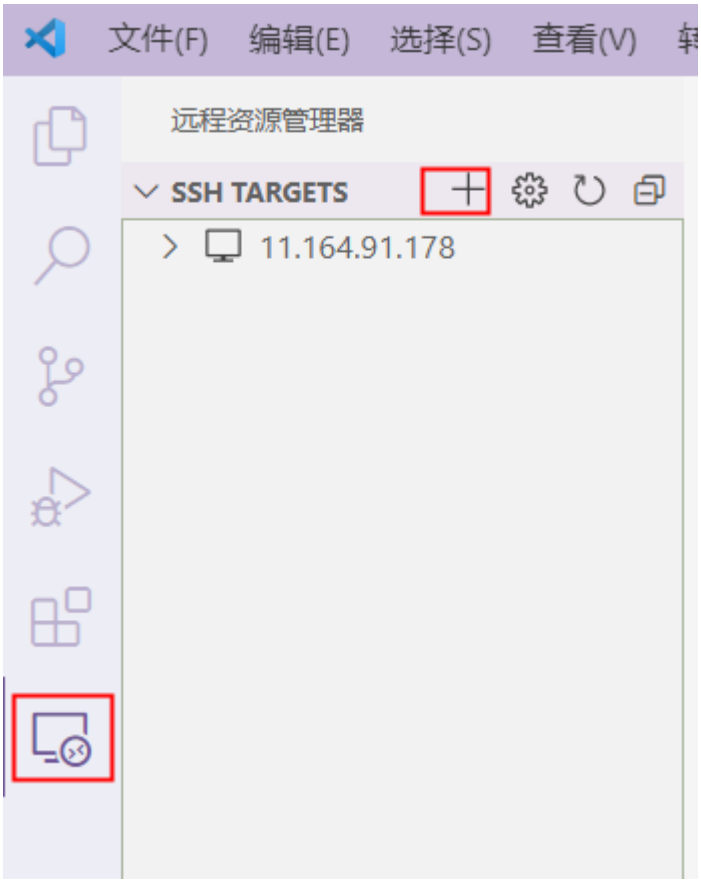
```
[root@ecs-0000 ~]# gdb --version
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
[root@ecs-0000 ~]#
```

#4.2 使用VSCode连接进行开发

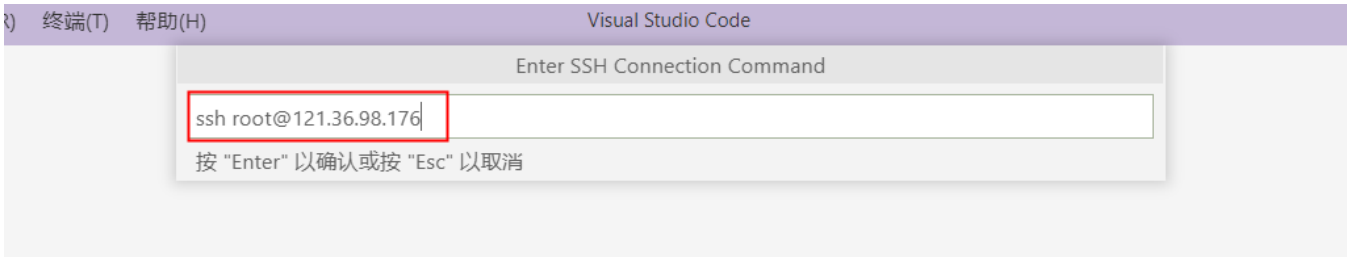
Note：VSCode需要事先在扩展`Ctrl+Shift+X`中安装以下插件：

- Remote-SSH（在本地安装）
- C/C++ （连接服务器后再安装，安装在服务器）
- CMake Tools（连接服务器后再安装，安装在服务器）

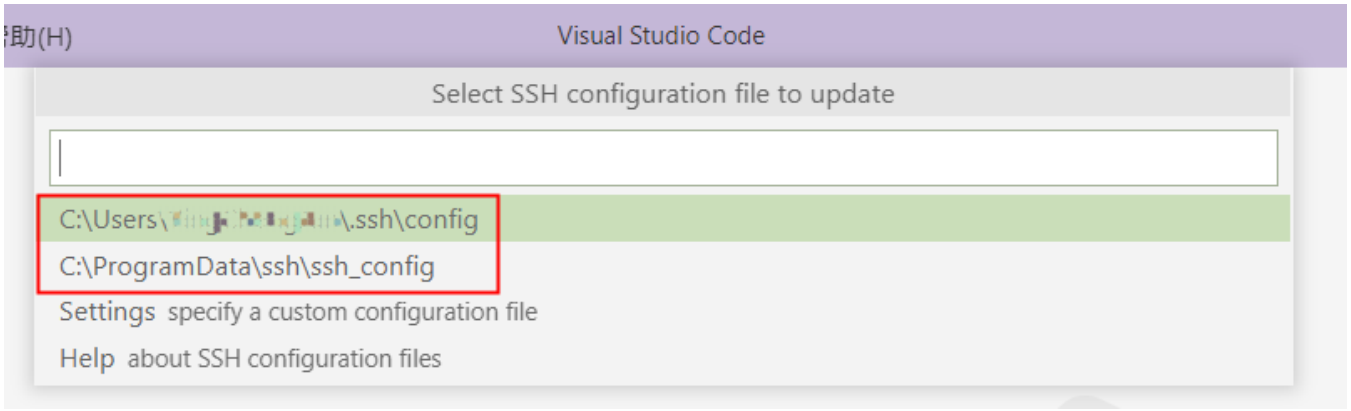
安装Remote-SSH扩展后，点击“+”号新建连接：



在弹框中输入`ssh <USERNAME>@<IP ADDRESS>`

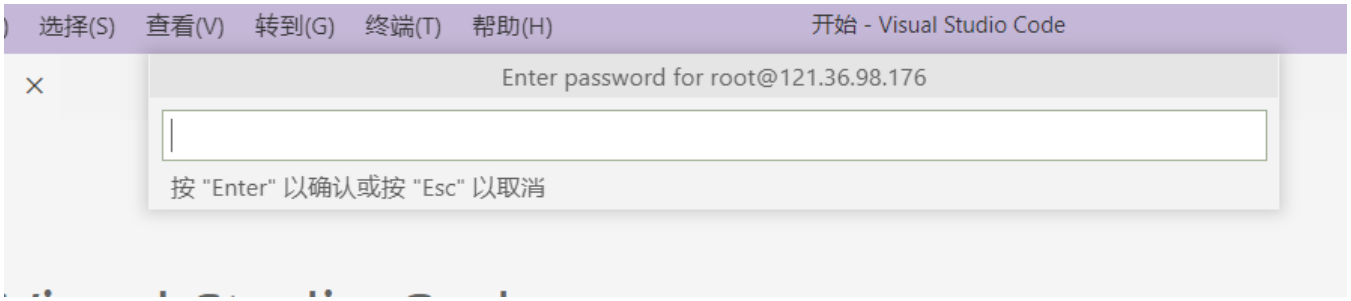


选择任意一个配置文件保存，通常是选上面那个，保存到用户下的配置文件：



在SSH TARGETS中选择刚刚添加的服务器，连接：

输入密码后回车：



打开你存放MiniSQL代码的文件夹：

后续操作方法与#2.3中类似，