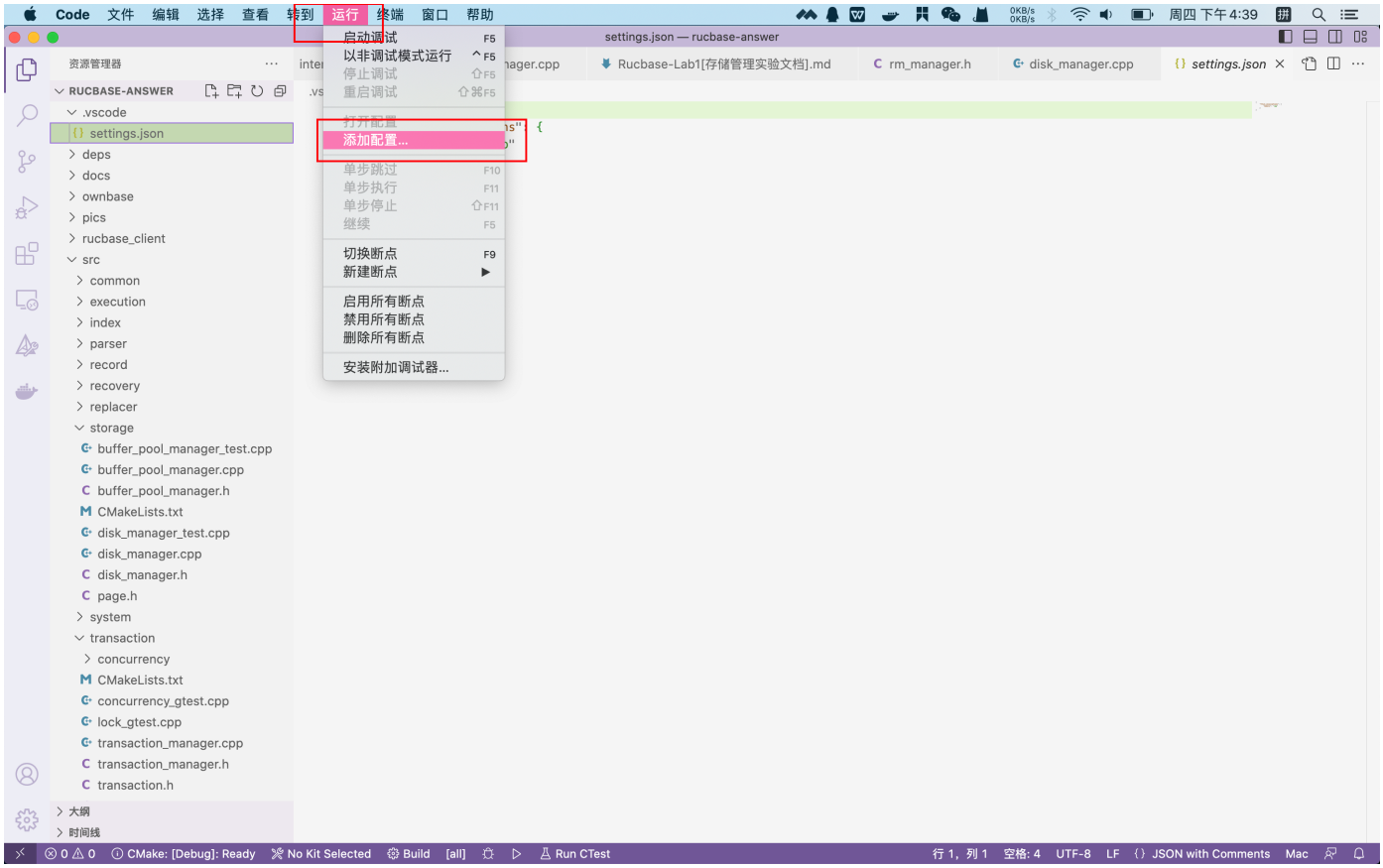
VSCode 配置Debug流程

1、在rucbase-lab/CMakeLists.txt文件夹中添加如下命令：

```
set(CMAKE_CXX_FLAGS "-Wall -O0 -g -ggdb3")
# set(CMAKE_CXX_FLAGS "-Wall -O3")

set(CMAKE_CXX_FLAGS_DEBUG "${CMAKE_CXX_FLAGS_DEBUG} -O0 -g")
set(CMAKE_C_FLAGS_DEBUG "${CMAKE_C_FLAGS_DEBUG} -O0 -g")
```

2、点击“运行（Run）--添加配置（Add Configuration）”添加launch.json文件



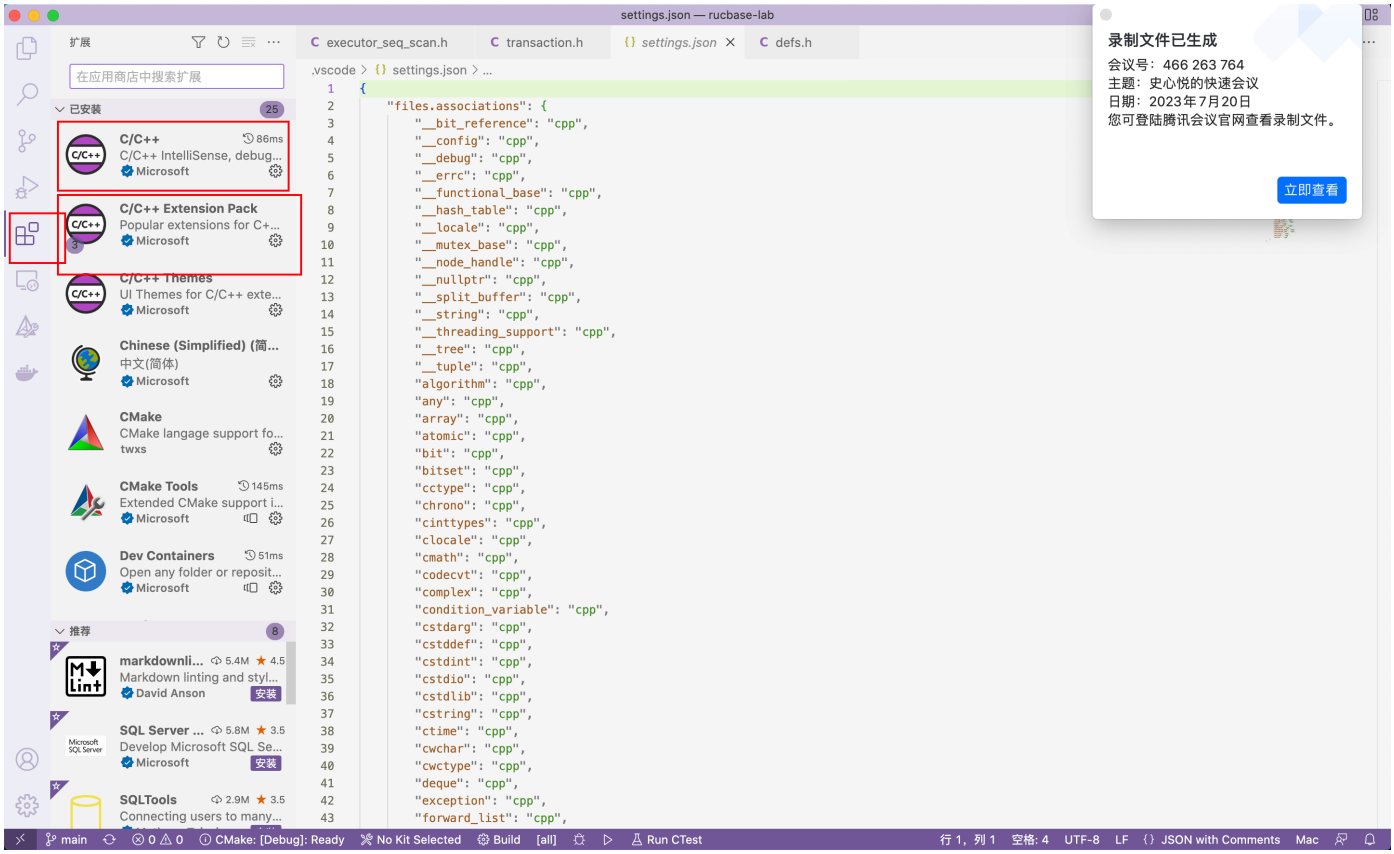
将群文件中的launch.json文件中的内容复制粘贴到新创建的launch.json文件中

其中，launch.json文件中通过“program”参数来指定Debug的目标执行单元，比如以下界面中，对rucbase可执行单元进行gdb：

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    // /home/zqs/star/bench_tpcc --logtostderr=1 --id=1 --servers="10.77.110.144:10215;10.77.110.145:10216;10.77.110.148:10217"
    "version": "0.2.0",
    "configurations": [{
        "name": "c++ Launch",
        "type": "cppdbg",
        "request": "launch",
        "program": "${workspaceFolder}/build/bin/rucbase", //bench_ycsb", // 需要执行的文件位置 ycsb" test/workload_cluster_test
        "args": [
            "testdb"
        ], // 输入的参数 "-f", "../../etc/observer.ini"
        "stopAtEntry": false,
        "cwd": "${workspaceFolder}/build/", // 当前运行的位置 // "cwd": "${workspaceFolder}",//
        "environment": [],
        // "terminal": "external",
        "externalConsole": false,
        "MIMode": "gdb",
        "setupCommands": [{
            "text": "-enable-pretty-printing",
            "description": "enable pretty printing",
            "ignoreFailures": true
        }],
        //"preLaunchTask": "bench_tpcc", //与tasks.json 的label项目同名
        "targetArchitecture": "x86_64",
        "miDebuggerPath": "/usr/bin/gdb"
    }]
}
```

如果需要对其他可执行单元进行Debug，需要把"program"的值改为对应可执行单元的路径。

3、在左侧的扩展插件中下载C++相关插件

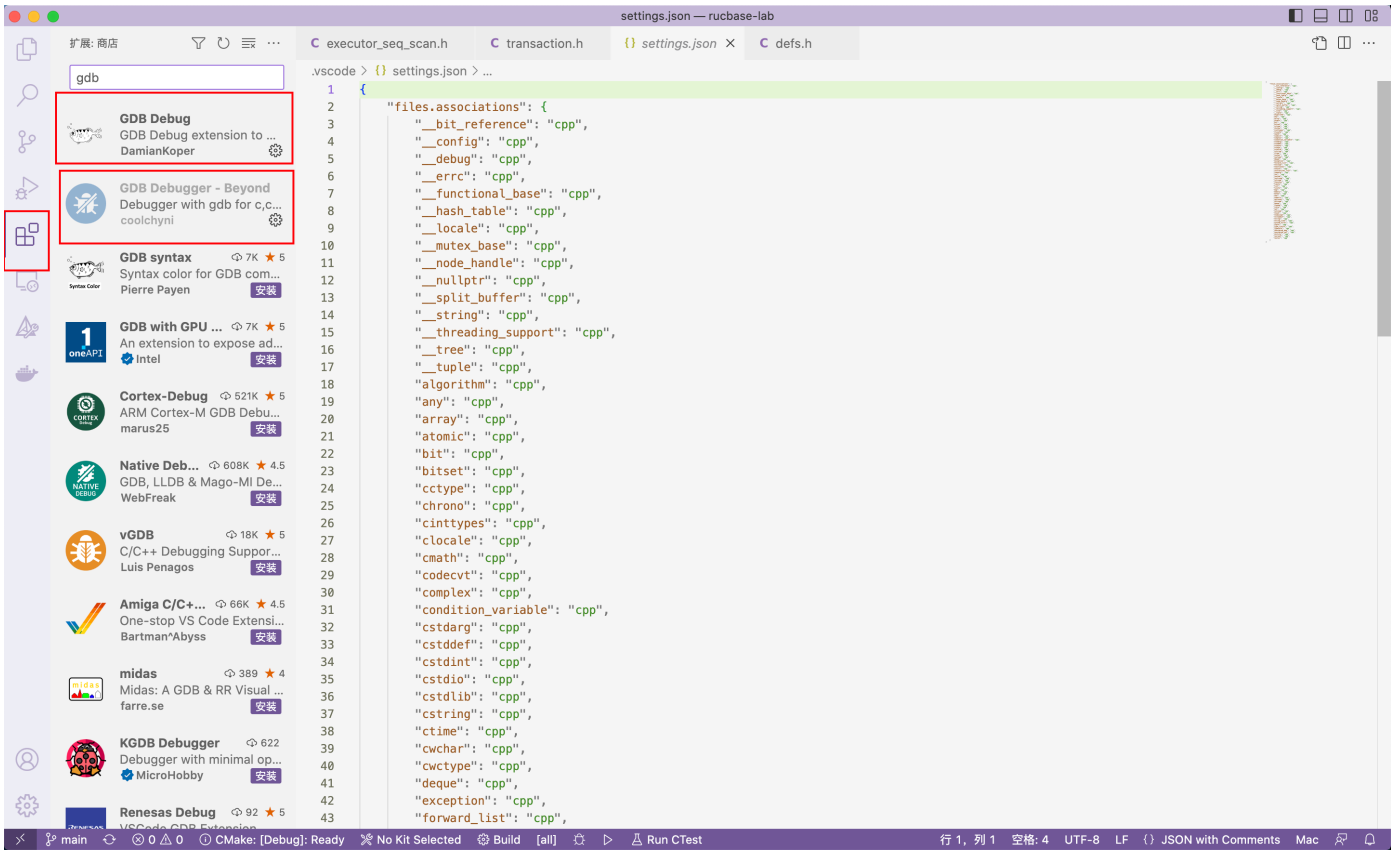


4、在命令行中执行下载安装gdb：

在命令行中执行如下命令：

```
apt-get install gdb
```

5、在左侧的扩展插件中安装gdb相关插件：



6、使用gdb进行Debug