## 工作日志:

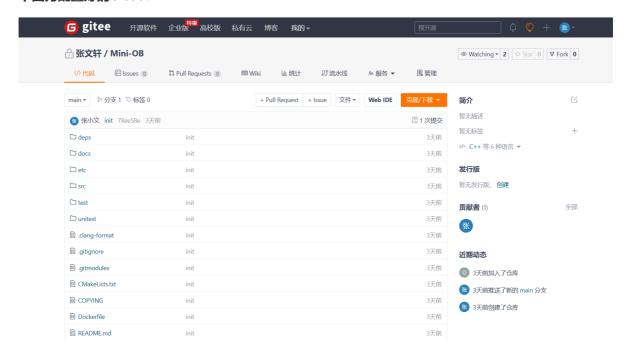
3月2日:讨论任务分工,确定每个人要完成的任务。浏览了MiniOB学习笔记并且注册了MiniOB实训营。

3月3日:完成了任务三的思索与回答。

3月4日: 了解任务四关于皮卡地里电视台广告售卖系统用例图中的include。

3月5日:完成了基于MiniOB的数据库管理系统内核实现实验的实验准备,以及Gitee的配置工作。

#### 下图为配置好的Gitee:



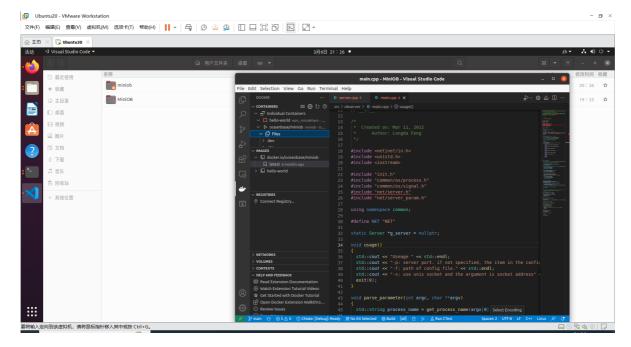
### 3月6日:完成了VSCode+Docker开发MiniOB的环境搭建与代码调试。

首先是搭建Docker环境并挂载本地目录到docker镜像中,第一遍操作时将我自己Gitee中的代码加载到了虚拟机中并且因为复制了下面的命令(含换行符),导致oceanbase/miniob与前面的指令分隔开导致出错。

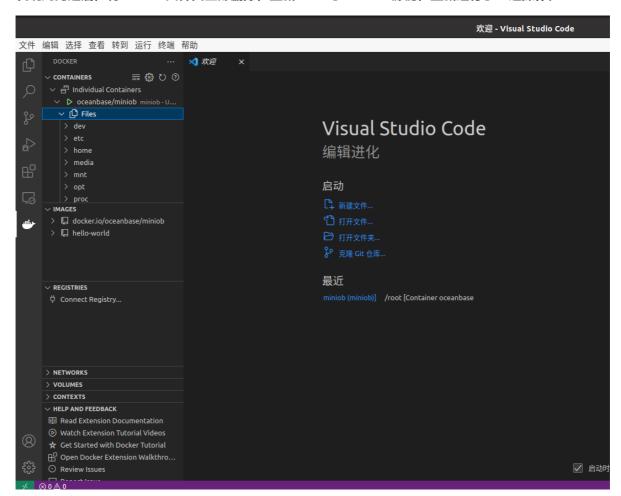
docker run -d --name miniob --privileged -v \$PWD/miniob:/root/miniob
oceanbase/miniob

 $! [image-20230307112832944] (C:\Users\12739\AppData\Roaming\Typora\typora-user-images\image-20230307112832944.png$ 

将上述指令修改,由于clone时出现的错误,导致出现了两个MiniOB文件夹,而miniob中只含有创建的build文件夹,导致vscode中相应的目录为空,如下图所示:



发现此问题后,将miniob文件夹全部删除,重新clone了MiniOB源码,重新进行了上述操作。

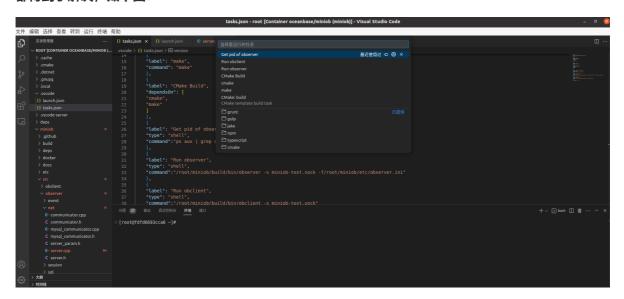


按指导书,创建task.json,选择运行任务后,没有cmake、make以及CMake: Build,如下图:

### 3月6日 23:06 • task.json - root [Container oceanbase/miniob (miniob)] - Visual Studio Code CMake: build 最近使用过 x ⇔ ⊈ CMake template build task ns": { "\${workspaceFolder} 十 配置任务 🛅 grunt 🛅 gulp 🗀 jake 🛅 npm nd": "cmake", typescript 🗎 : [ :UG=ON", cmake cppbuild : 显示所有任务...

经过不断的研究,最终发现指导书的截图中为tasks.json,于是将task.json修改为task.json后,一切都得到了解决,如下图:

nd": "make"



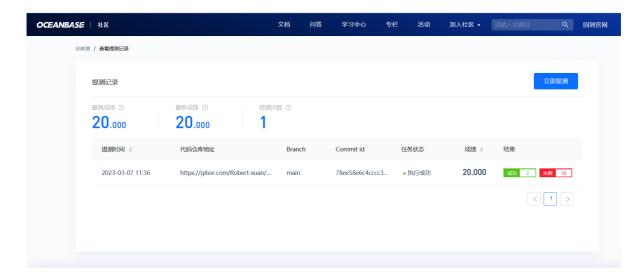
执行完cmake,make以及CMake:Build后,build中的内容如下图上方内容,执行完Run observer与 Run obclient后,build中的内容如下图下方内容。

```
@fdfd6693cca6:~/miniob/build
                                                                                                      Q =
[root@fdfd6693cca6 miniob]# cd build
[root@fdfd6693cca6 build]# ls -l
total 64
-rw-r--r-- 1 root root 15434 Mar 7 02:31 CMakeCache.txt
                                                  7 02:36 CMakeFiles
drwxr-xr-x 5 root root 4096 Mar
-rw-r--r-- 1 root root 15318 Mar
                                                    7 02:36 Makefile
drwxr-xr-x 2 root root 4096 Mar
                                                   7 02:35 bin
-rw-r--r-- 1 root root 2322 Mar 7 02:31 cmake_install.cmake drwxr-xr-x 4 root root 4096 Mar 7 02:36 deps drwxr-xr-x 2 root root 4096 Mar 7 02:34 lib drwxr-xr-x 5 root root 4096 Mar 7 02:36 src
drwxr-xr-x 3 root root 4096 Mar 7 02:31 test
drwxr-xr-x 3 root root 4096 Mar 7 02:36 unit
                                                   7 02:36 unitest
[root@fdfd6693cca6 build]# ls -l
total 96
-rw-r--r-- 1 root root 15434 Mar 7 02:31 CMakeCache.txt
drwxr-xr-x 5 root root 4096 Mar 7 02:43 CMakeFiles
-rw-r--r-- 1 root root 15318 Mar 7 02:43 Makefile
drwxr-xr-x 2 root root 4096 Mar 7 02:35 bin
-rw-r--r-- 1 root root 2322 Mar 7 02:31 cmake_install.cmake drwxr-xr-x 4 root root 4096 Mar 7 02:43 deps drwxr-xr-x 2 root root 4096 Mar 7 02:34 lib
drwxr-xr-x 3 root root 4096 Mar 7 02:43 miniob
srwxr-xr-x 1 root root 0 Mar 7 02:43 miniob-test.sock
-rw-r--r-- 1 root root 26251 Mar 7 02:43 observer.log.20230307 drwxr-xr-x 5 root root 4096 Mar 7 02:43 src
drwxr-xr-x 3 root root 4096 Mar 7 02:43 test
drwxr-xr-x 3 root root 4096 Mar 7 02:43 unitest
[root@fdfd6693cca6 build]#
```

SQL语句测试以及配置Debug环境,如下图:

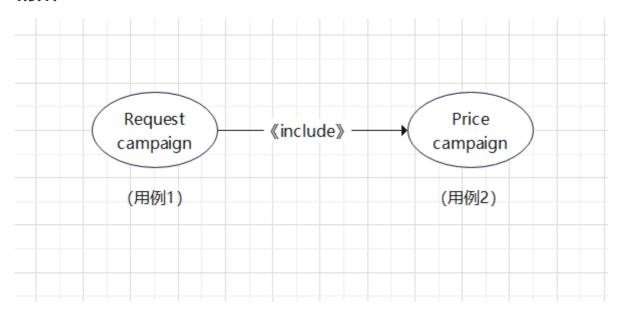
```
| Separate | Communicator | Communic
```

3月7日: 提测代码。



# 任务四:

#### 1.9.1:



《include》:包含关系,指用例可以简单地包含其他用例具有的行为,并把它包含的用例行为作为自身行为的一部分。包含关系代表着基础用例会用到被包含用例,将被包含用例的时间流插入到基础用例的时间流中,其基于分解出两个或更多用例所共有的行为,来实现用例的复用,简化用例图。

箭头指向: 从包含用例指向被包含用例。

用例2对于了解用例1的主要目的并不是必须的,但是用例2必须在执行用例1前先执行,用例2只有其结果才是比较重要的。