**实验1**

**basic**

1. basic
   1. 实验要求

本次实验需要完成MiniOB开发环境的搭建，并对MiniOB的源码组织有初步了解。

MiniOB本身具有的一些基本功能。比如创建表、创建索引、查询数据、查看表结构等。在进行后续实验时，需要留意不要破坏这些基础功能。

* 1. 环境要求

Miniob需要在Linux环境下编译。

CMake：版本 >= 3.13。

gcc：gcc建议8.3以上，编译器需要支持c++20新标准。

flex：2.5+。

bison：3.7+，用于生成词法分析语语法分析部分。

此外，其他所需的依赖环境可以通过执行源码中提供的build.sh进行安装。

下面提供了两种不同的开发环境搭建方法。

* 1. 开发环境搭建（1.5h）

本节提供了使用GitPod开发与linux虚拟机开发两种开发环境的搭建方法。GitPod环境配置比较简单，导入github仓库之后会自动执行环境配置的脚本，所以不需要额外的环境配置了。但gitpod属于浏览器在线开发因此比较卡顿。

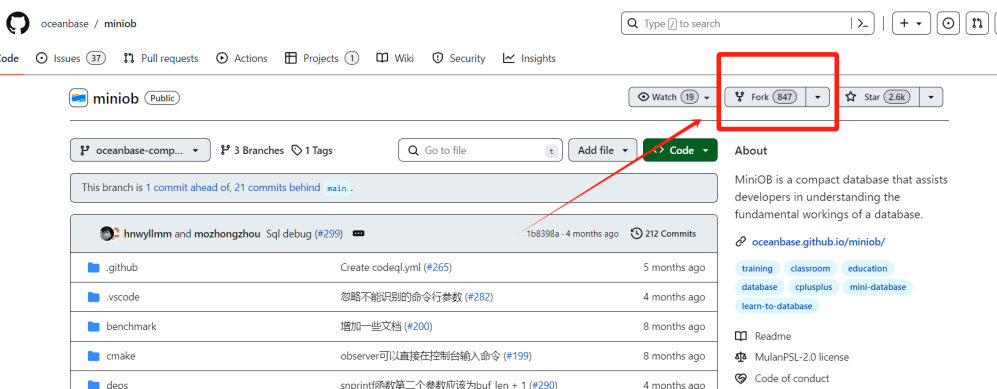
Linux虚拟机开发则不会出现卡顿情况，但是环境配置稍微困难。在今后的学习中同学们不可避免的要使用虚拟机，因此我们更推荐采用Linux虚拟机去完成本实验。

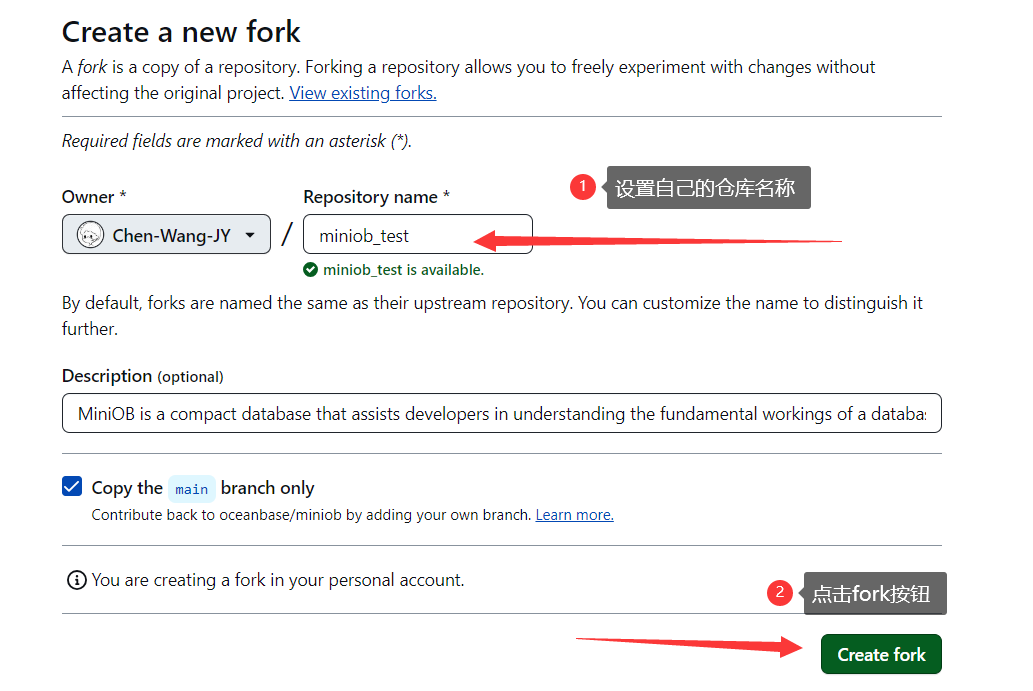
* + 1. 使用GitPod开发

Gitpod 是一个基于云的集成开发环境，专为在浏览器中进行软件开发而设计。它允许开发者在几分钟内启动完全预配置的开发环境，无需在本地安装任何软件或配置开发环境。在开发时，GitPod会提供一个类似虚拟机一样的云开发环境，采用Linux系统。同时，GitPod可以直接打开github项目，支持包括vscode在内的多种编辑器。

* + - * 创建自己的github仓库

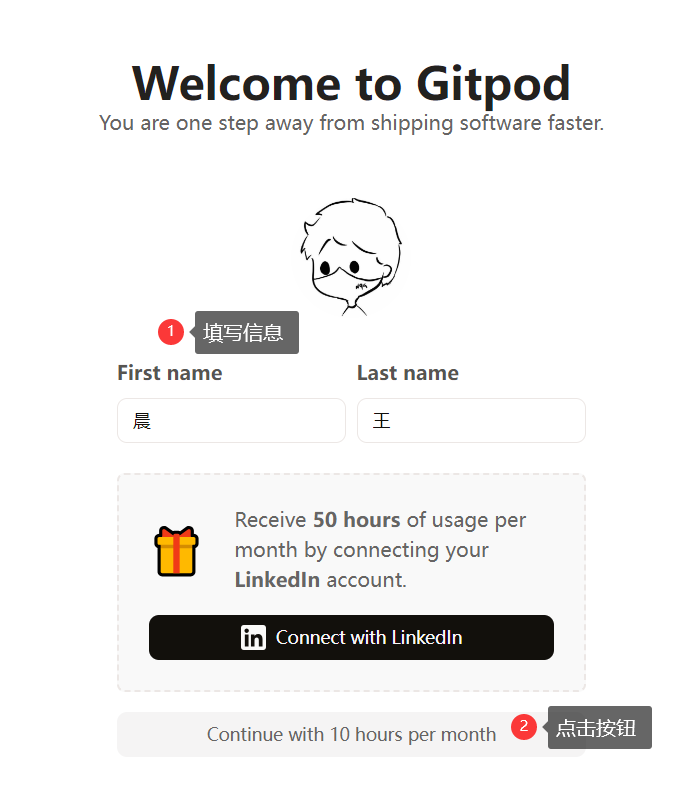
在浏览器中打开[miniob官网](https://github.com/oceanbase/miniob/tree/oceanbase-competition-2023)源码仓库，单击右上方的fork选项，将仓库fork到自己的账户下。之后，在自己的github账号中会出现刚刚创建的miniob源码。

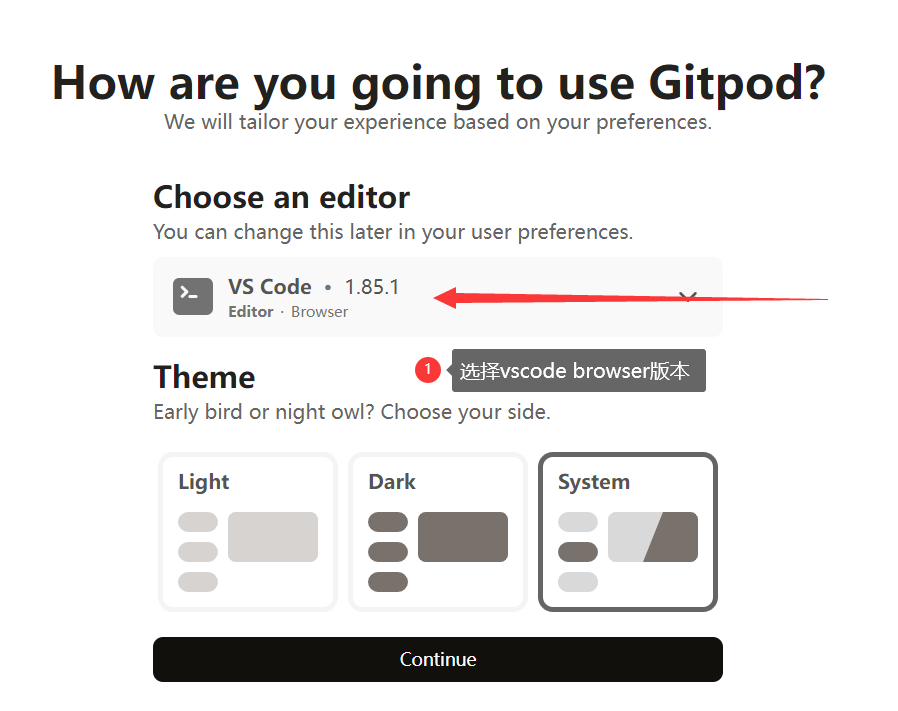




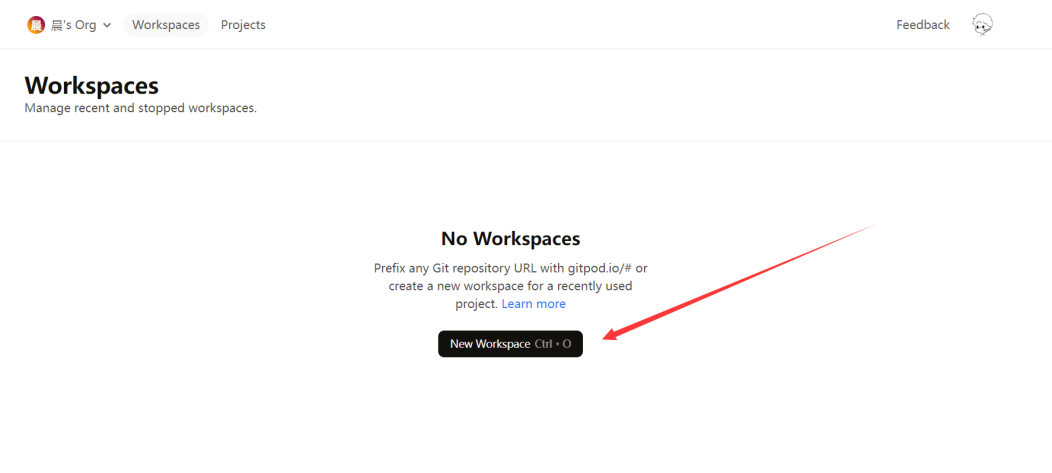
* + - * 使用gitpod打开项目

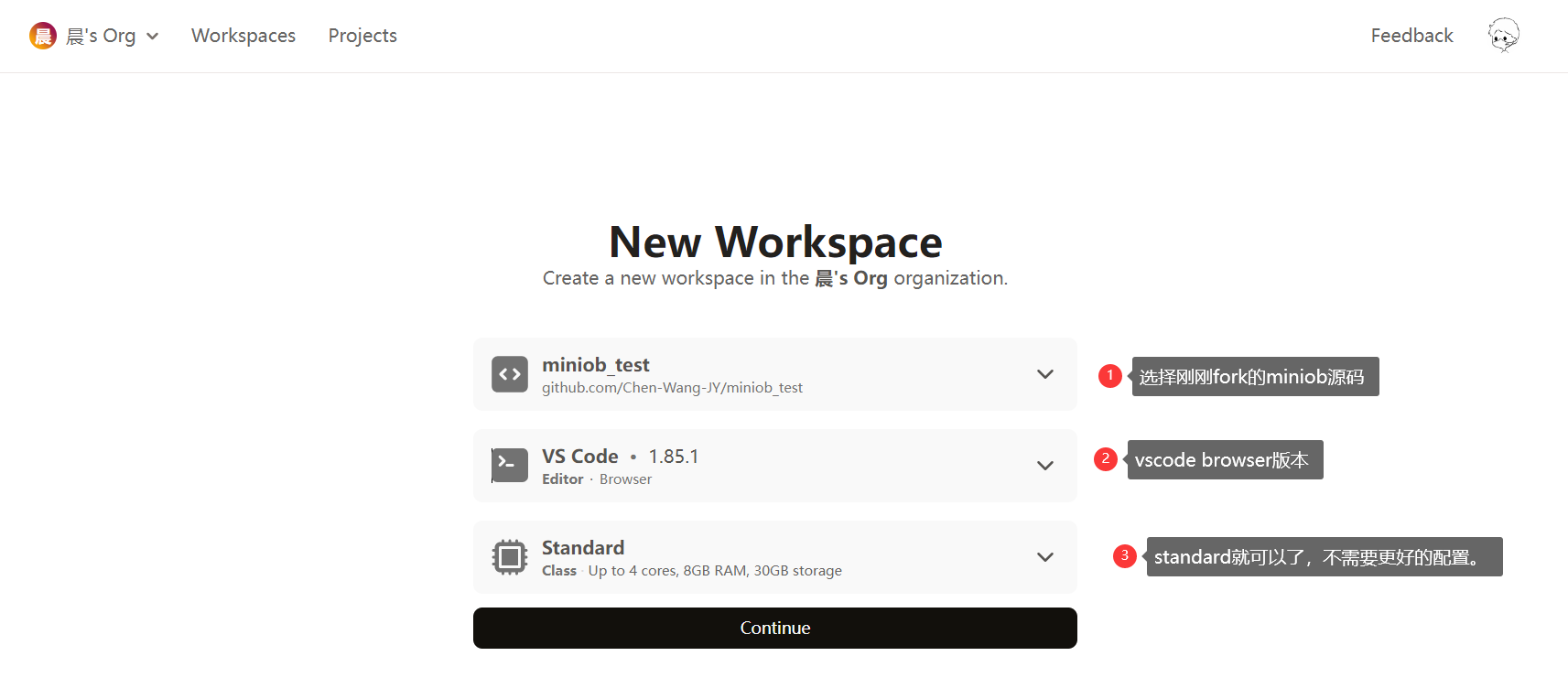
进入[gitpod官网](https://gitpod.io/)，采用自己的github账号登陆gitpod。如果是初次使用，需要填写一些额外信息：



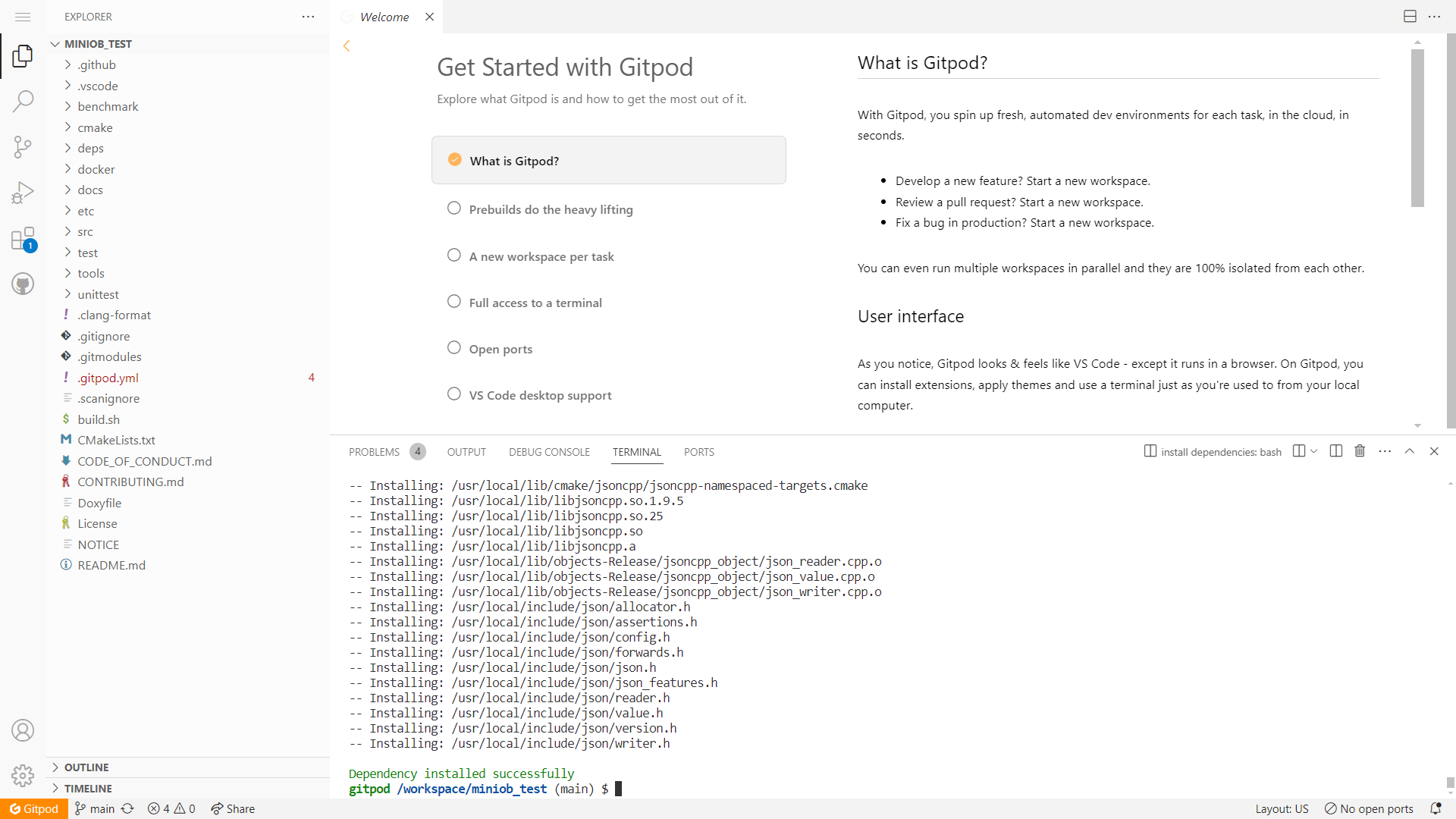


首次登陆后，我们需要新建工作空间，并打开自己fork的miniob源码。

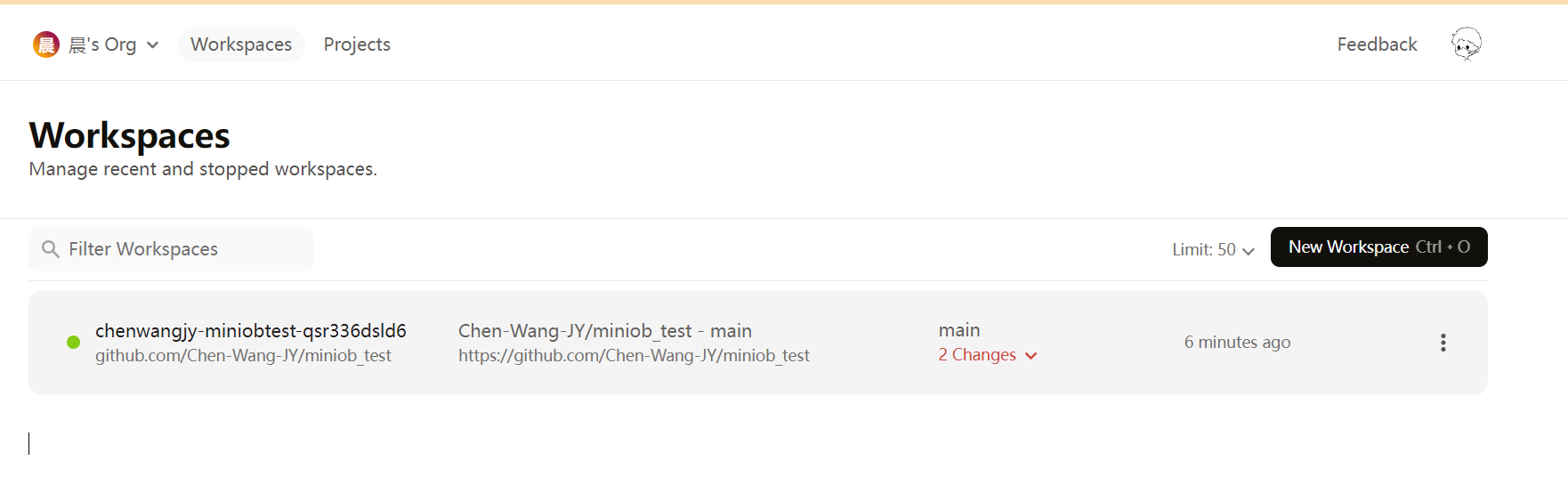




在等待短暂的系统初始化后，便可以进入如下的开发界面：



之后，可以在自己gitpod的工作空间下找到刚刚的项目，点击进入即可。



Gitpod项目启动时，会自动运行初始化，因此，接下来可以直接编译miniob并运行了（第1.4节）。如果没有自动运行初始化程序的话，在命令行中手动执行下面的命令：

sudo apt-get update

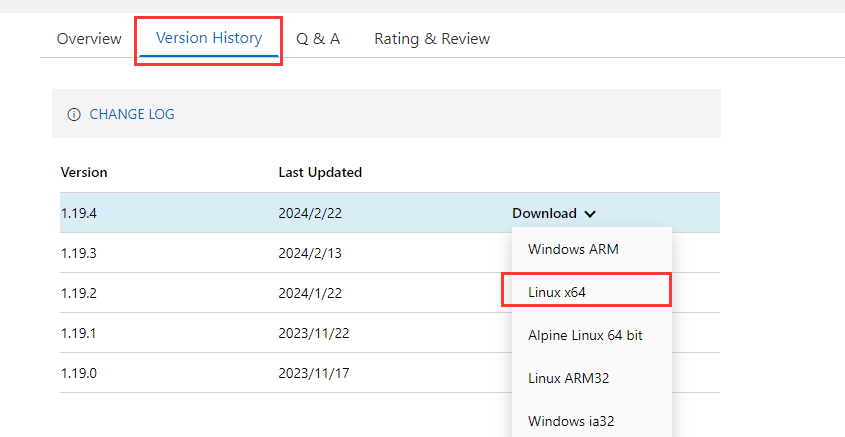
sudo apt install cmake flex bison texinfo libreadline-dev -y

sudo bash build.sh init

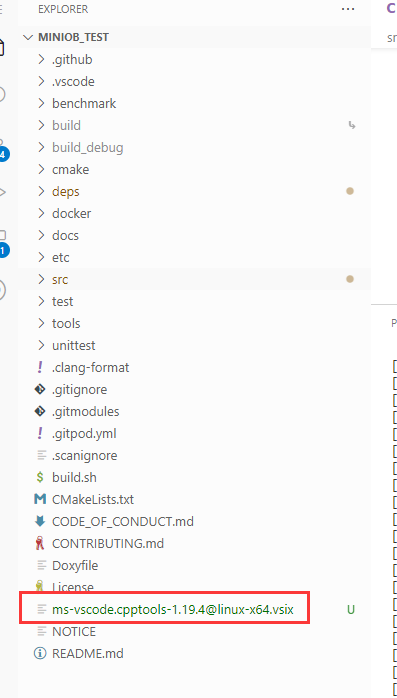
* + - * 必要的插件安装

GitPod 默认是自带 C/C++ 开发插件的，但是功能比较弱，使用起来并不方便。我们常用的Vs Code C/C++Microsoft在gitpod-vscode的插件市场中无法检索到。需要手动安装。

点击[该链接](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools)进行插件下载：



将下载好的插件直接拖拽到gitpod-vscode中



进入插件市场，点击Install from vsix，选择刚刚拖拽进来的 .vsix文件即可安装。

图形用户界面, 文本, 应用程序

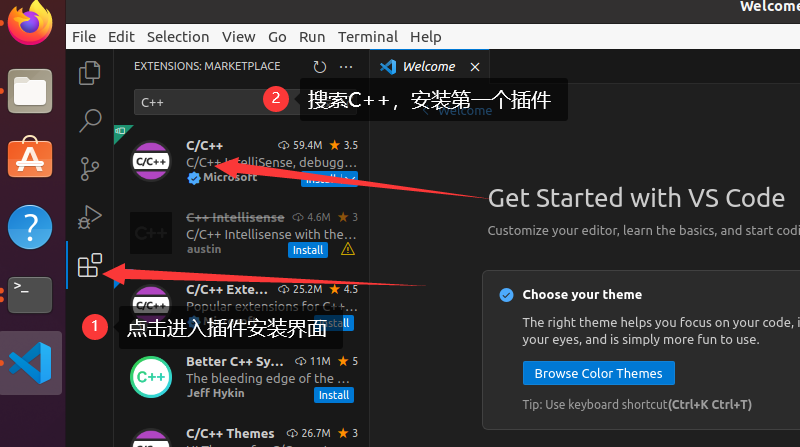
描述已自动生成

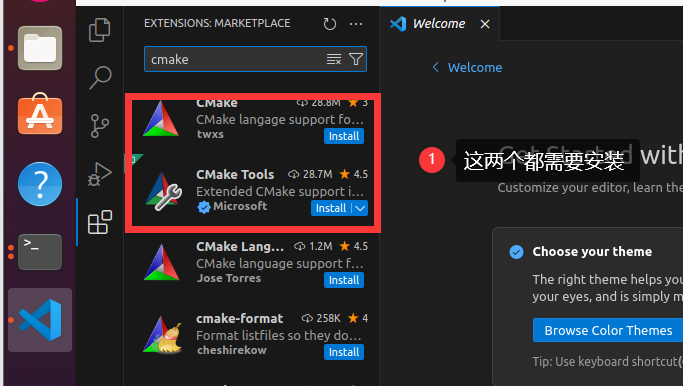
另外，如果本地电脑上已经安装了vscode，我们也可以采用本地的vscode连接到gitpod创建的工作空间进行开发。在创建工作空间时，选择vscode desktop，即可跳转到本地vscode中。

* + 1. 采用linux虚拟机开发

如果对在线的gitpod云环境不习惯的话，我们也可以在本地的linux系统中进行开发。此时，我们需要安装一台linux系统的虚拟机。虚拟机软件与虚拟机的设置教程在《vmware与ubuntu安装教程.doc》中。

在配置好ubuntu并安装了vscode后，我们还需要在vscode中安装必要的插件：





* + - * 代码克隆

git clone https://github.com/oceanbase/miniob.git -b oceanbase-competition-2023

文本

描述已自动生成

有时候可能因为网络问题导致克隆失败，可以考虑使用国内gitee镜像，将git地址替换为https://gitee.com/xiaomayo/miniob.git（后续执行bash build.sh init的时候也需要clone其他仓库，可能同样会出现克隆失败的问题，这个gitee镜像里已经替换了submodule的源地址，为了省事可以直接使用gitee的这个地址）

* + - * flex、bison、cmake安装

在终端中执行以下命令，以安装相关必要工具。

sudo apt install flex

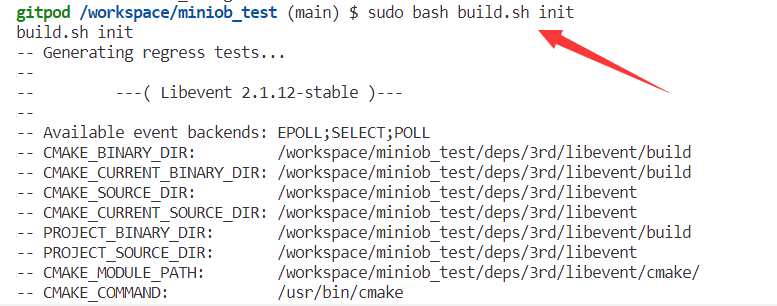
sudo apt install bison

sudo apt install cmake

* + - * 其他依赖库的安装

编译前需要安装一些miniob的依赖库，miniob源码中提供了脚本来自动安装相关依赖库。在miniob源码目录下，执行下面的命令自动安装：

sudo bash ./build.sh init

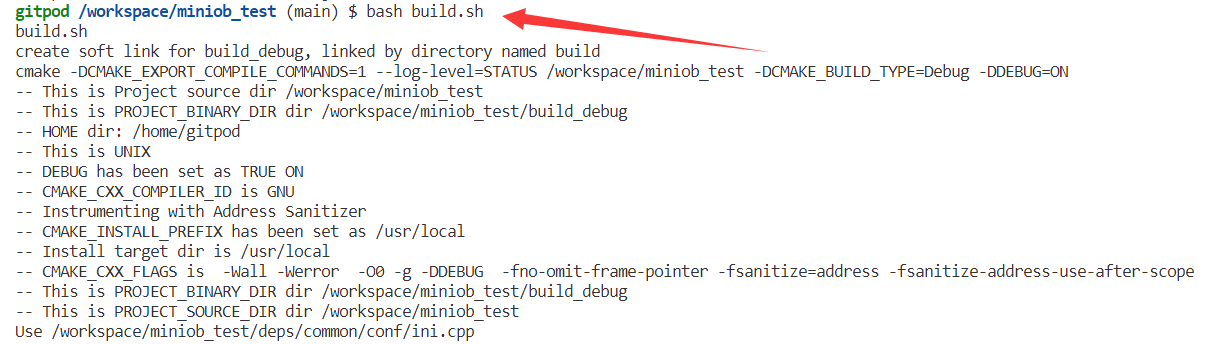


至此，必要的环境已经安装完成，接下来就可以编译miniob并运行了。

* 1. 编译项目（10min）
     1. 在终端编译

在命令行中，执行下面的命令即可完成编译：

bash build.sh



此命令将编译一个DEBUG版本的miniob。需要注意的是，初次编译miniob的速度比较慢，需要耐心等待（一分钟左右）（gitpod环境可能会更慢一些）。

如果希望编译其它版本的miniob，可以参考 bash build.sh -h打印的帮助目录，比如：

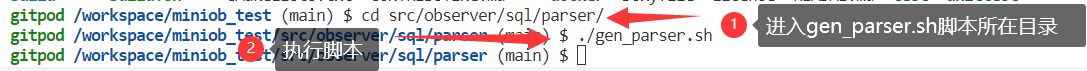
bash build.sh release

此命令将编译release版本的miniob。

另外，miniob依赖于flex、bison提供的词法分析、语法分析功能，如果对相关文件（lex\_sql.h, lex\_sql.l, yacc\_sql.y等）进行改动的话，需要执行gen\_parser.sh脚本，该脚本调用flex、bison工具重新生成词法分析与语法分析的c++代码，随后再编译整个项目。

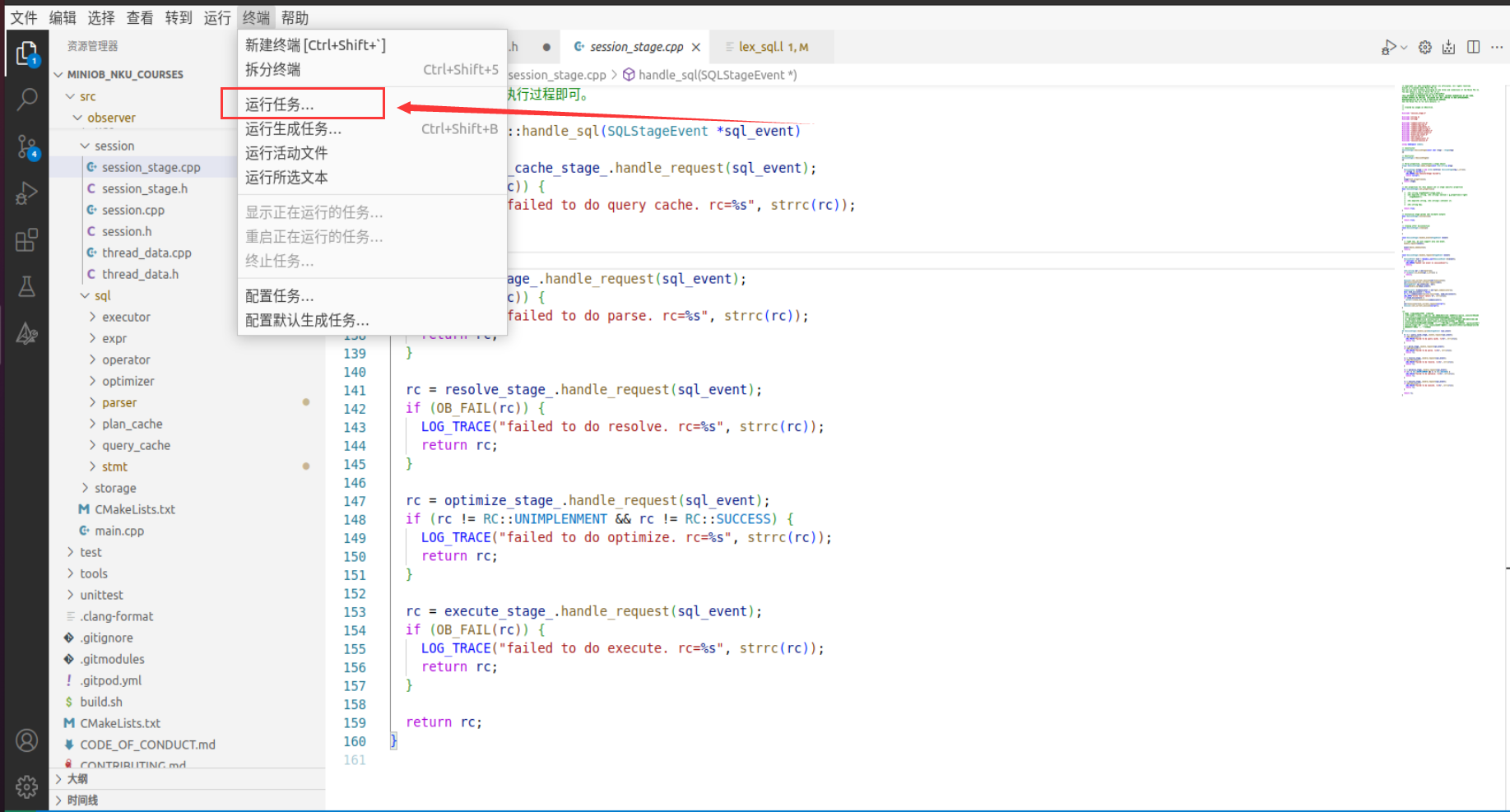
cd src/observer/sql/parser

./gen\_parser.sh



* + 1. 在vscode中编译

点击Terminal—Run Task，并在弹出的对话框中点击build\_debug，即可进行编译；同时，点击gen\_parser即可完成词法分析与语法分析代码的生成。



图形用户界面, 文本, 应用程序

描述已自动生成

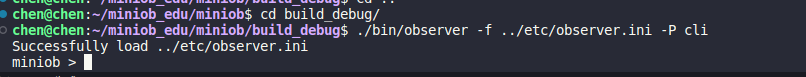
* 1. 运行项目（5min）

编译完成后，可以在build目录（build\_debug或build\_release）下找到bin/observer，该文件是服务端程序，bin/obclient是自带的客户端程序。Miniob的服务端程序既可以以无需客户端的直接执行命令交互的方式启动，还可以以监听TCP端口的方式启动或是监听unix socket的方式启动。

为了简便，推荐采用无需客户端的方式、在命令行直接执行命令交互的方式启动服务端程序。

cd ./build\_debug

./bin/observer -f ../etc/observer.ini -P cli



在这种方式下，可以直接解析SQL命令。



* 1. 源码介绍

Miniob的源码位于./src目录下，目录组织如下：

./src

├── obclient

└── observer

├── common

├── event

├── net

├── session

├── sql

│   ├── executor

│   ├── expr

│   ├── operator

│   ├── optimizer

│   ├── parser

│   ├── plan\_cache

│   ├── query\_cache

│   └── stmt

└── storage

├── buffer

├── clog

├── common

├── db

├── default

├── field

├── index

├── persist

├── record

├── table

└── trx

下面简要介绍部分源码的功能。

* + 1. ./src/observer/net

这部分代码负责与客户端的交互功能，接受发送客户端的请求与回答。

* + 1. ./src/observer/session

每一个客户端的连接都被视作一个会话（session），在miniob中没有做特殊的会话管理，并对会话处理做了简化。在一个会话中，处理一个SQL语句需要经过五个阶段：query cache、parse、resolve、optimize、execute。该部分代码位于./src/observer/session/session\_stage.cpp文件中。另外，Miniob没有实现query cache，只是提供了该阶段的接口。在parse阶段，通过词法分析、语法分析来解析输入的SQL语句；在resolve阶段，将解析后的SQL语句，转换成各种类型的Stmt(Statement)，同时进行错误检查；optimize阶段根据解析后的Stmt生成逻辑和物理查询计划，并进行优化；最后的execute阶段，根据物理查询计划，执行SQL语句。

文本

中度可信度描述已自动生成

* + 1. ./src/observer/sql/parser/

该目录下的代码负责parse阶段，主要负责解析输入的SQL语句。解析后的结果为parse\_defs.h中的某一个结构体类型。

* + 1. ./src/observer/sql/stmt/

该目录下的代码负责resolve阶段，每一种类型的SQL语句都有自身对应的stmt，如SELECT语句对应的SelectStmt、CREATE TABLE语句对应的CreateTableStmt等。

* + 1. ./src/observer/sql/optimizer/与./src/observer/sql/operator/

该目录下的代码负责optimize阶段，包括根据Stmt生成逻辑查询计划、根据逻辑计划生成物理计划两部分。

* + 1. ./src/observer/sql/executor/

该目录下的代码负责execute阶段。对于DDL语句，没有对应的查询计划，直接在executor阶段中执行；而对于一些DML语句，会产生执行计划。

* 1. 测试说明（重要）

本次实验课的内容需要通过oceanbase数据库大赛训练营的测评。比赛官网链接：<https://open.oceanbase.com/competition>。自行注册账号即可。

在官网的最下面，有训练营的入口，我们也可以通过以下超链接进入：<https://open.oceanbase.com/train>。



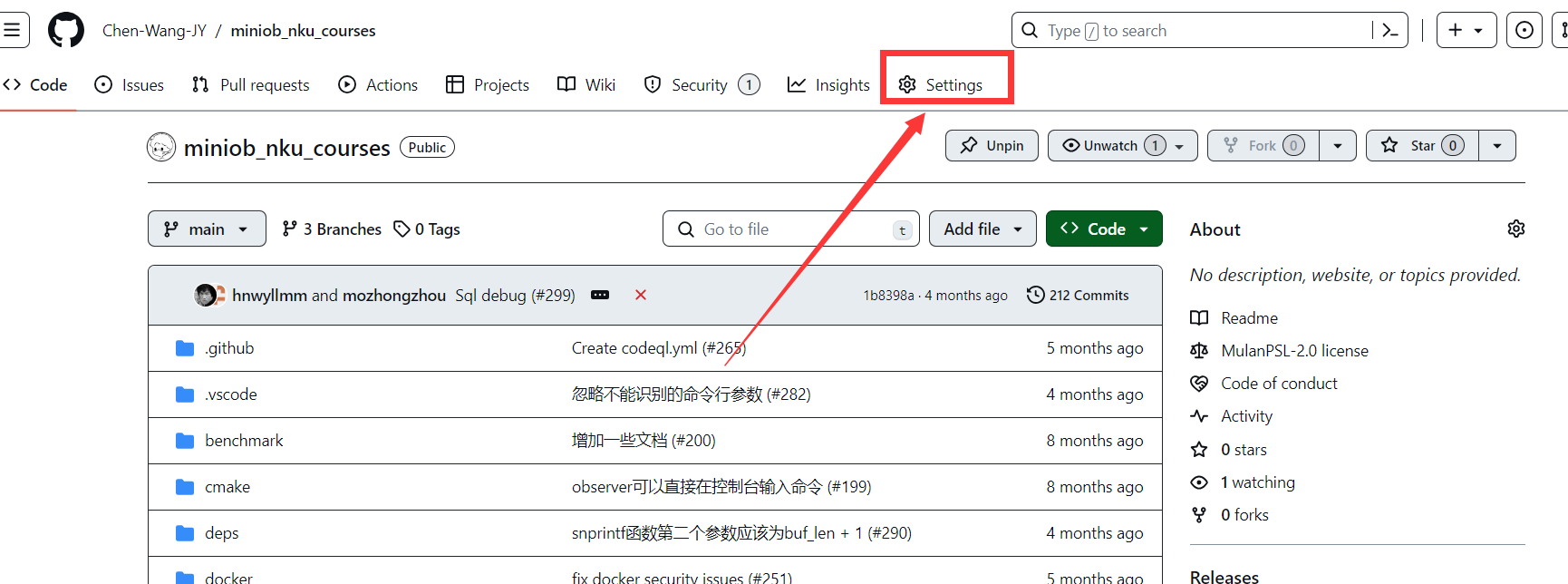
采用oceanbase训练营进行提测时，需要给出github仓库链接、commitid以及branch名称。



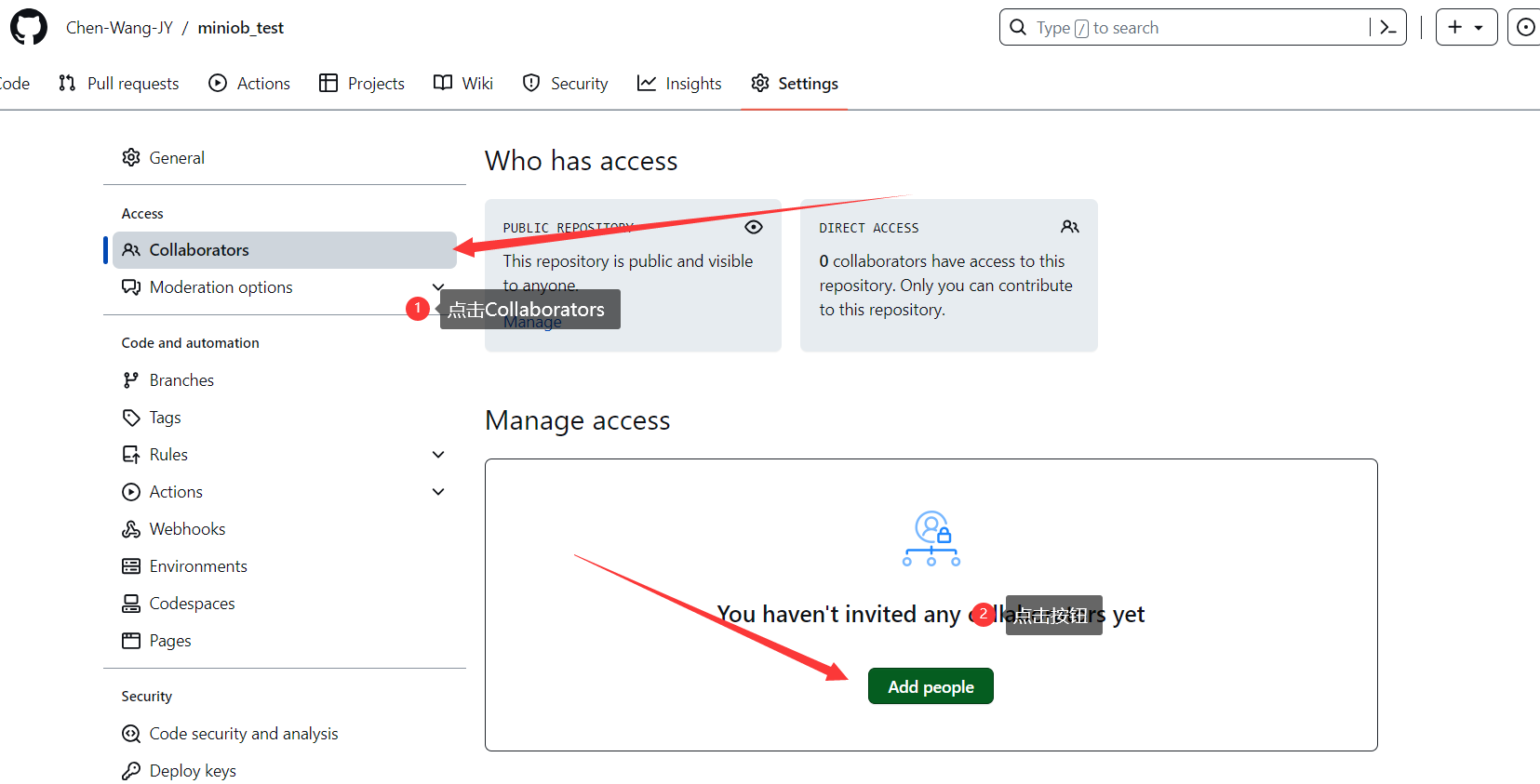
以下给出具体的提测方式。

* + 1. 邀请Oceanbase测试账号
       - Github

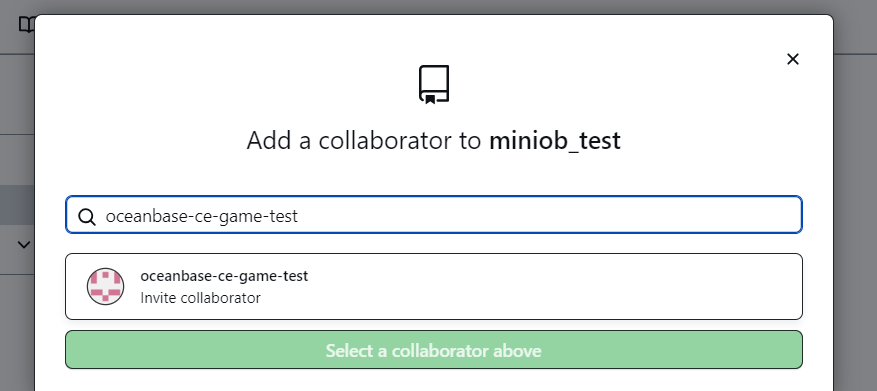
进入刚刚fork的github仓库中，进入设置界面。



点击左侧导航栏的Collaborators，选择Add People以添加合作者。



输入oceanbase-ce-game-test，添加Oceanbase官方测试账号作为合作者。



Github在邀请合作者加入仓库时，会发送邮件通知被邀请人，以征求被邀请人的统一。在对oceanbase官方测试账号发送邀请后，官方测试人员会定期刷新邮件处理邀请。在官方人员接受邀请后，就可以提交测试了。

* + - * Gitee

将仓库https://gitee.com/xiaomayo/miniob进行fork后，按照下图所示邀请官方的gitee账号。



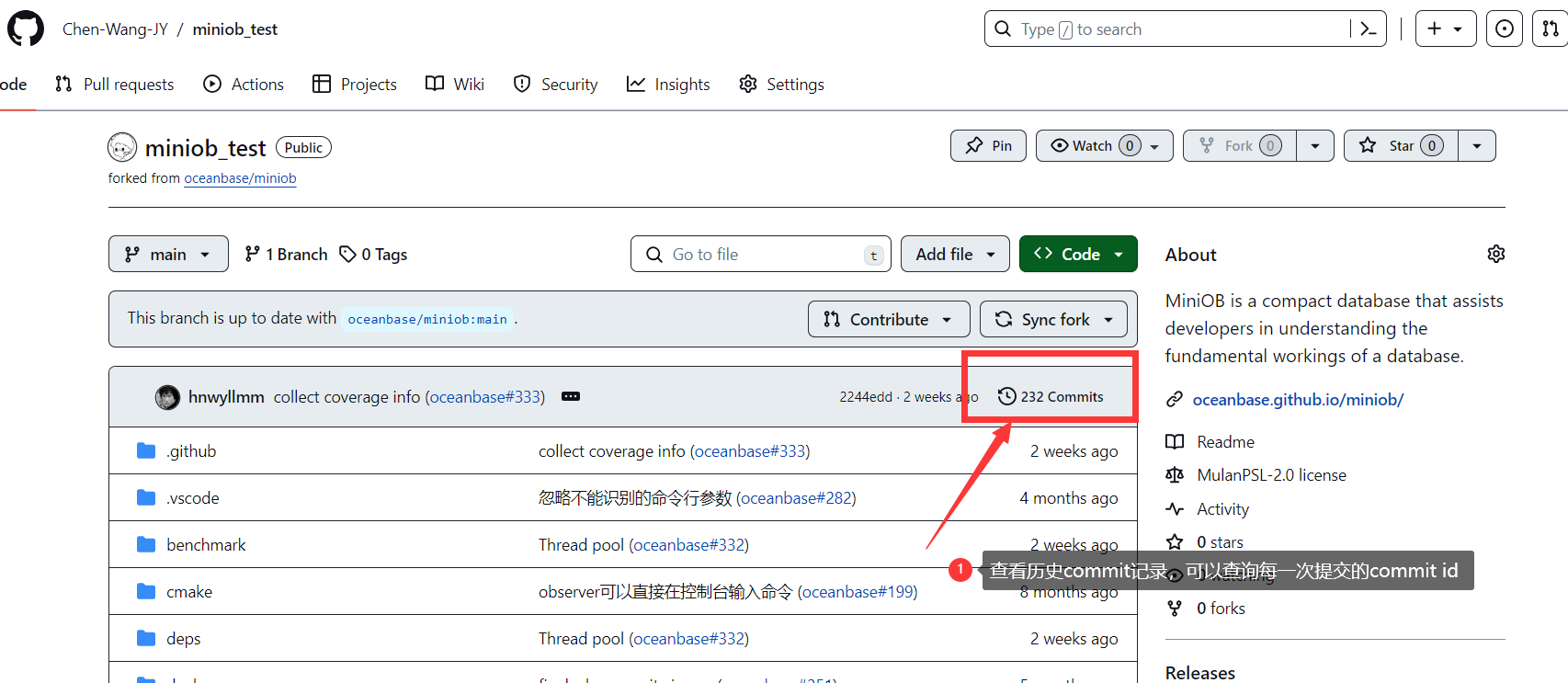
选择直接添加，权限选择“观察者”，输入官方的gitee账号oceanbase-ce-game-test，选择第一个进行添加，如下图所示。

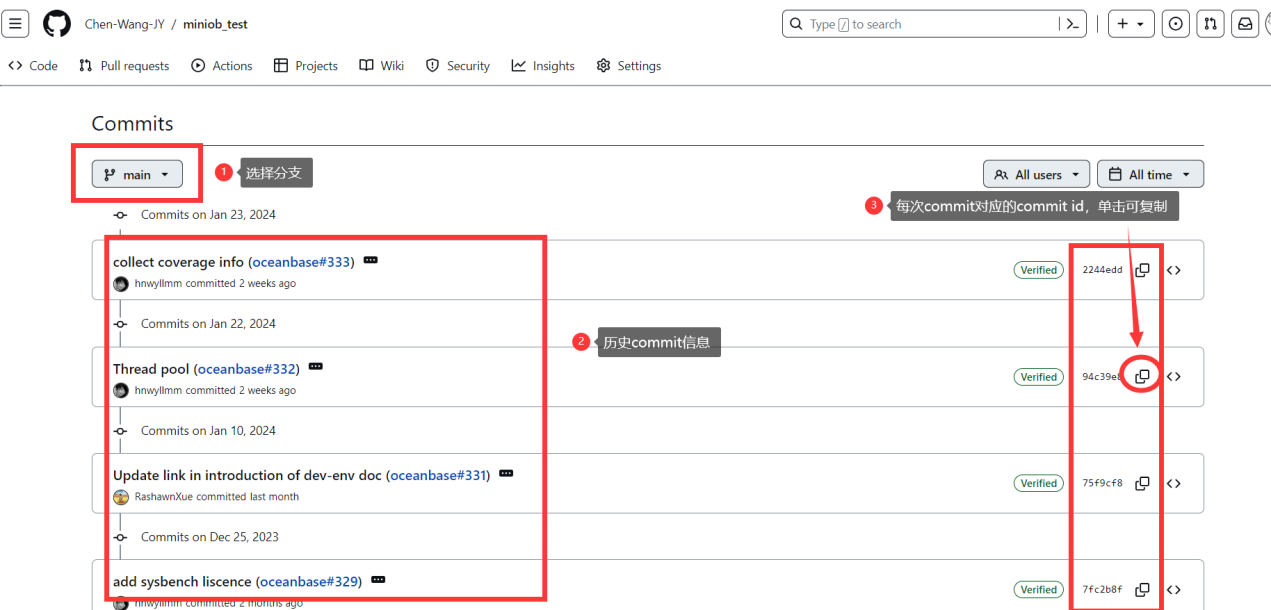


点击添加后，官方的gitee账号会直接通过，此时仓库成员出现了oceanbase-ce-game-test则表示添加成功。

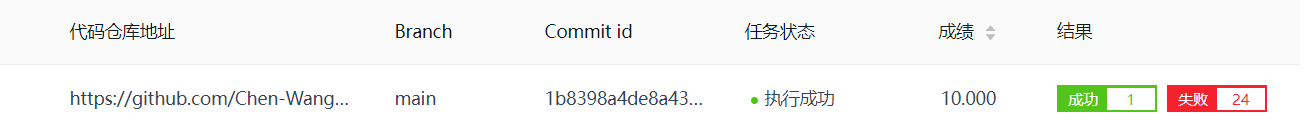
* + 1. 实验1 basic的提交流程

Oceanbase训练营提测时需要给出仓库地址、commit id以及分支名。我们fork的仓库已经由多个贡献者进行代码提交与合并，我们点击commits按钮，便可以查询分支名、历史的commit记录以及每一次提交的commit id





可以看出，当前分支名为main，我们复制最近一次commit记录的commit id，提交到训练营中，oceanbase官方会进行测评并返回结果。



在后续实验过程中，同学们需要自行学习git的使用，尝试将本地修改好的代码提交至github对应仓库中。例如，对于实验2 date题目，我们创建了新的分支exp1\_date，并进行了两次commit。

