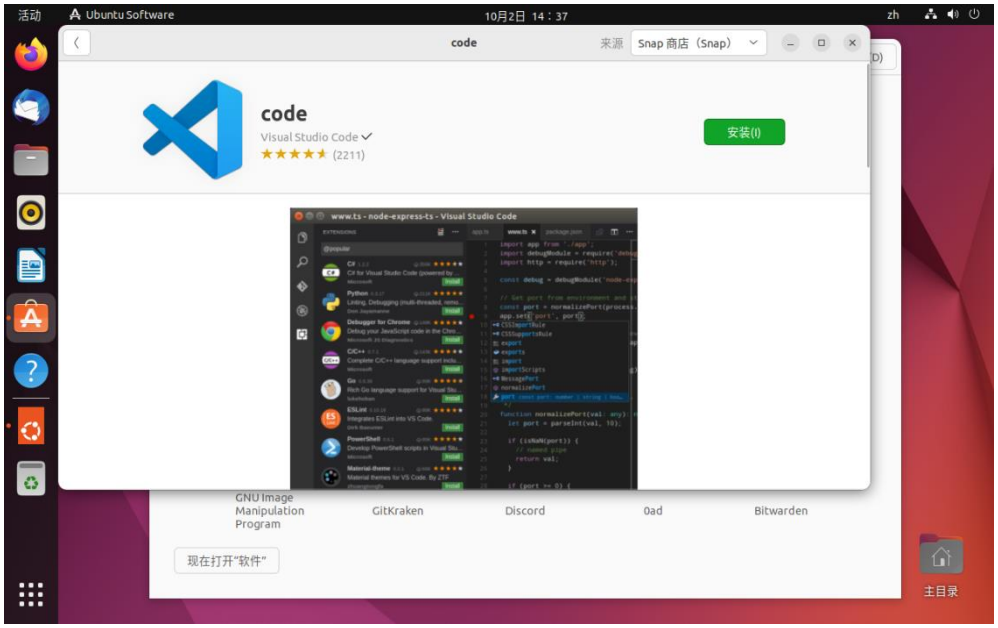
- vscode 和 cpp-tool-chain

vscode 直接在 ubuntu 商店里面安装



然后工具链这里，选择 gcc，g++，gdb（喜欢 clang 的也可以自行安装 clang）

sudo apt upgrade

sudo apt install build-essential

sudo apt install cmake

如果需要安装较高版本的 cmek，可自行从官网下载较高版本的 cmake 进行安装，可以通过下面的命令检查相关的工具是否安装完成

gcc -v

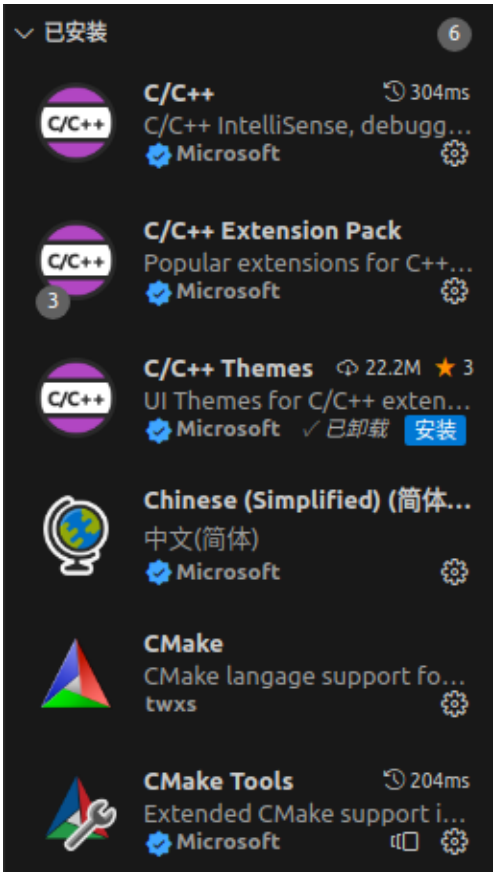
g++ -v

gdb -v

make -v

cmake --version

下面去 vscode 的商城里面安装 cpp 插件，包括如下几个（近期发现 vscode 的市场无法访问，请同学们自行准备代理）



下面是配置 配置文件，在项目目录下面新建.cpp 文件，并运行，这时会提示配置文件不完整，需要我们进行配置，此时，会在项目目录下生成.vscode 文件，内含三个文件

将 launch.json 配置如下

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "C/C++",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}/${fileBasenameNoExtension}",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
```

```
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "preLaunchTask": "compile",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```

task.json 配置如下

```
{
    "version": "2.0.0",
    "tasks": [{
            "label": "compile",
            "command": "g++",
            "args": [
                "-g",
                "${file}",
                "-o",
                "${fileDirname}/${fileBasenameNoExtension}"
            ],
            "problemMatcher": {
                "owner": "cpp",
                "fileLocation": [
                    "relative",
                    "${workspaceRoot}"
                ],
                "pattern": {
                    "regexp": "^(.*):(\\d+):(\\d+):\\s+(warning|erro
r):\\s+(.*)$",
                    "file": 1,
                    "line": 2,
                    "column": 3,
                    "severity": 4,
                    "message": 5
                }
            },
            "group": {
```

```
            "kind": "build",
            "isDefault": true
        }
    }
]
}
```

有兴趣的同学可以研究下里面各个配置项的含义，这里不再赘述，此时，我们就可以运行并调试 cpp 程序了。

p.s. 请不要使用 code runner 插件，此插件无法调试 cpp 程序

- 使用远程仓库和 git（同学们可以直接使用课程提供的 baseline 无需再使用 git，git 仅仅用于个人的项目管理）

在终端中输入

sudo apt-get install git

然后配置用户名和邮箱

git config --global user.name "your name"

git config --global user.email "your email"

我们可以打开~/.gitconfig 文件查看设置的结果

然后设置 ssh 密钥

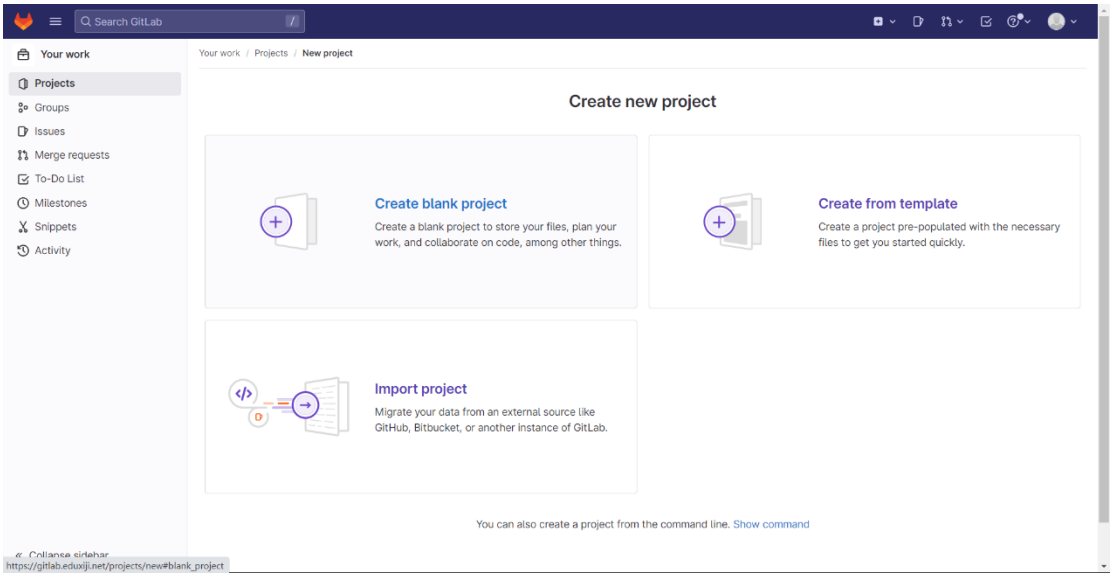
ssh-keygen -t rsa

连续点三次回车，之后可以到/home/username/.ssh/中查看

通过编译原理课程分配的账号和密码登录希冀平台（course.educg.net），下面是我使用课程分配的账号登录 github 的过程

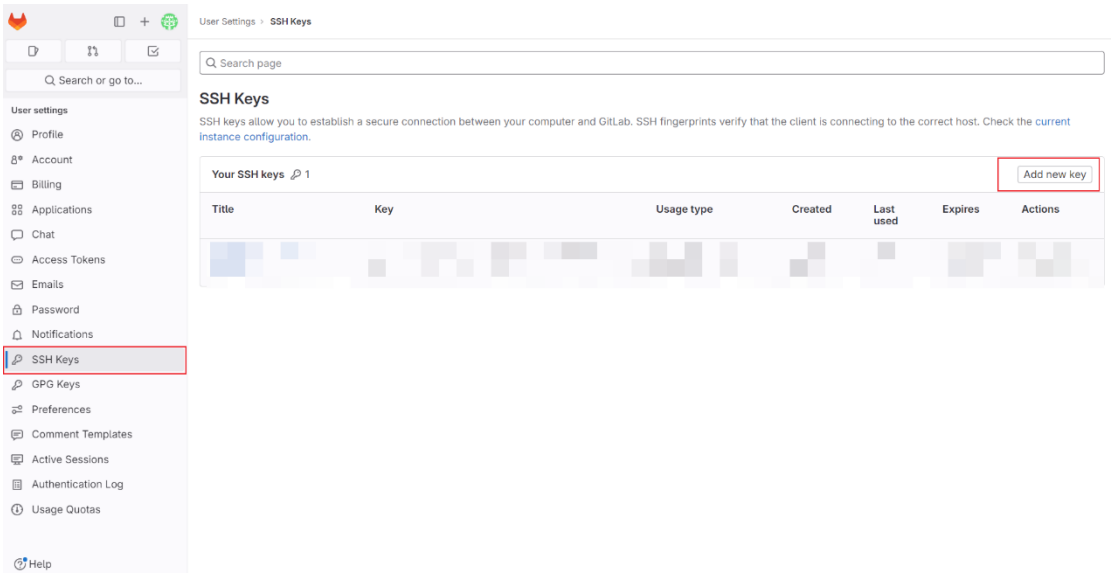
登录之后的界面如下所示：

下面我们点击右上角的更多栏目，选择 gitlab 即可登录，gitlab 的账号和密码与希冀平台相同。

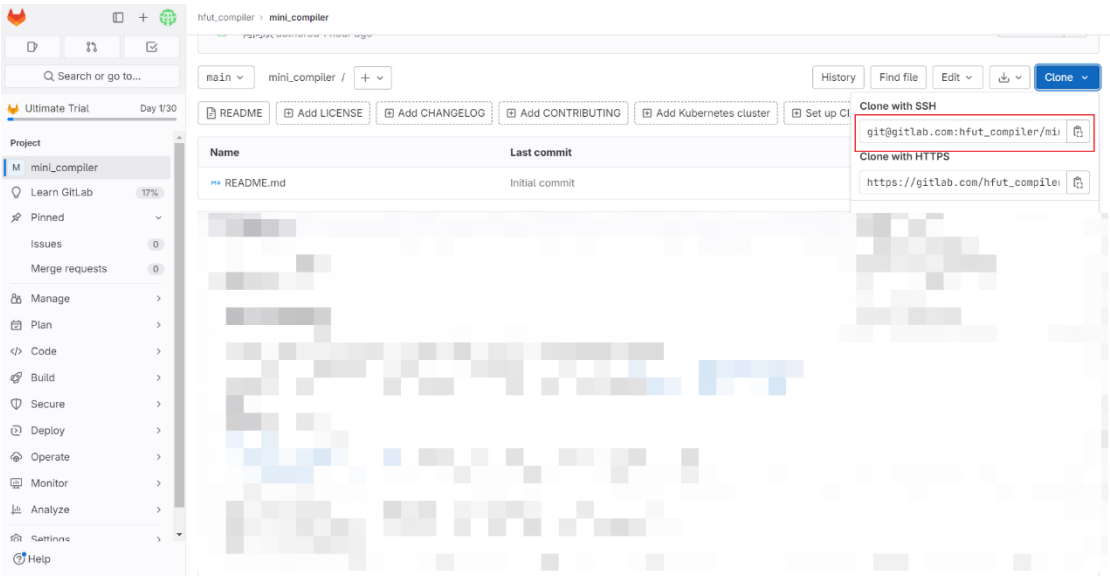


点击创建空项目，根据自己的需要进行设置

在 gitlab 中找到公钥设置（可以在上面搜索框中直接搜索 ssh），将 id_rsa.pub 中的内容粘贴进去，标题请随意

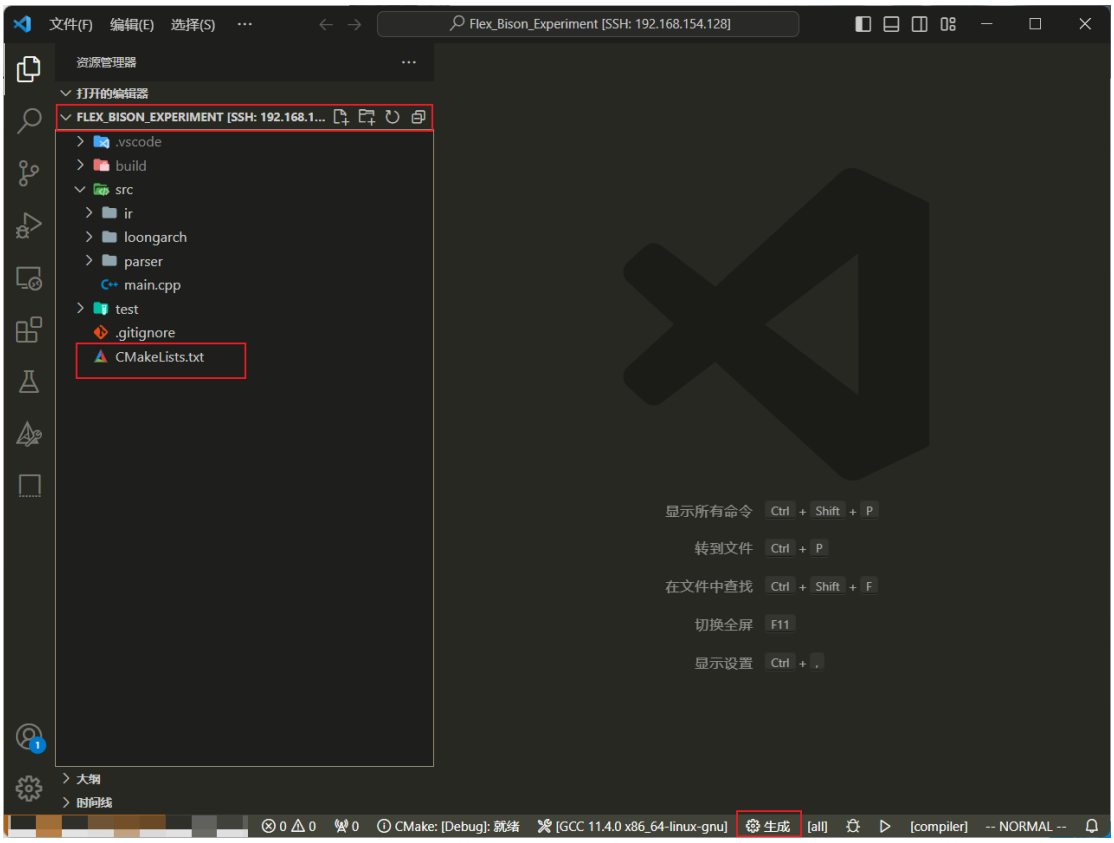


然后将自己项目 clone 到本地即可

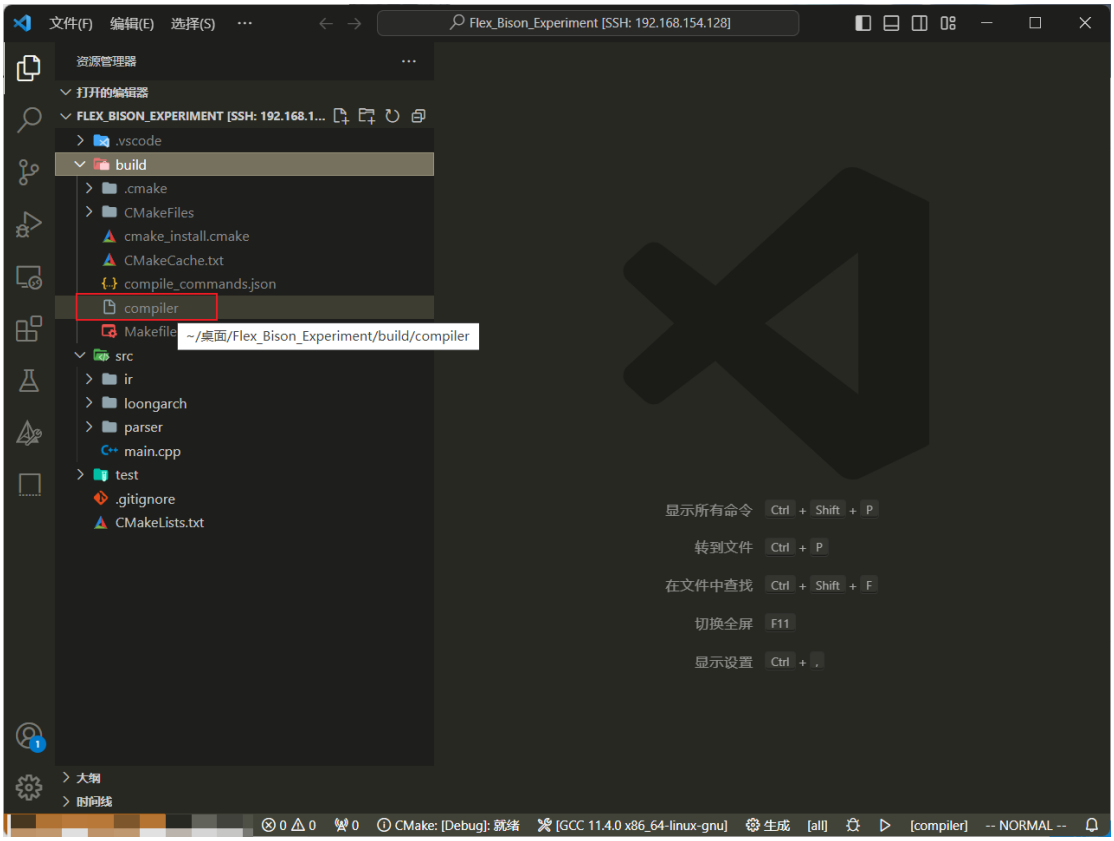


git clone url

在 vscode 里面打开项目，确保 CMakeList.txt 文件在项目的根目录下，此时我们会发现底部多出了一个生成按钮，这样我们可以避免每次构建项目时重新运行 cmake 和 make 指令，具体如下图所示：



在生成时，我们需要注意终端中的报错信息，并及时修改代码或项目配置中的问题，确保项目成功构建。



若构建成功，我们会在 build 文件夹里面发现可执行文件，如果我们生成之后发现 build 文件夹里面没有这个文件，那么说明我们的项目在构建的过程中出现了问题，此时我们需要检查错误，修改之后尝试重新 build。（在出现一些令人不明所以的问题时，我们可以尝试将整个 build 文件夹删除，然后重新生成）