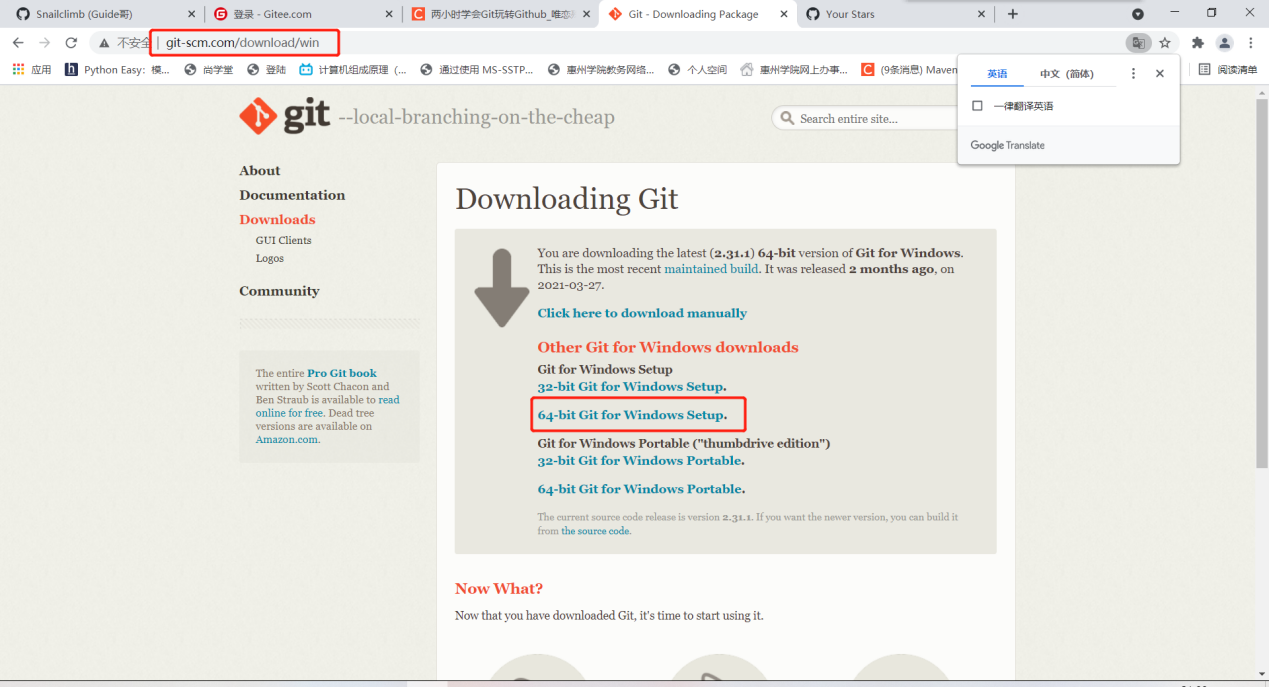
# Git

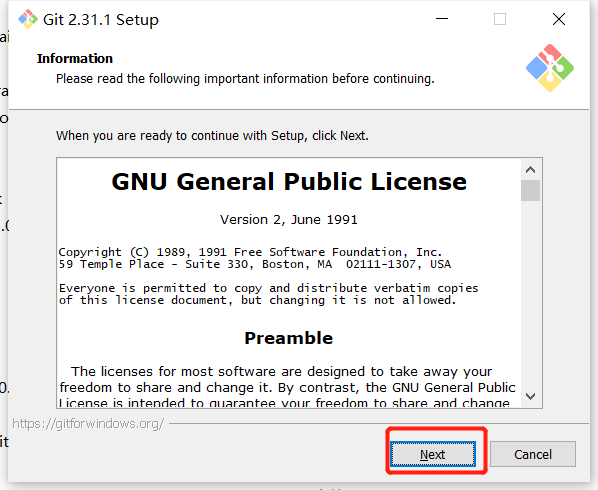
## Git安装

官网下载安装包

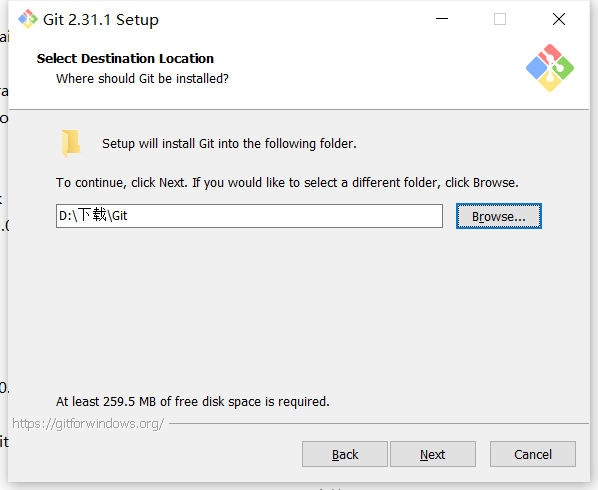
https://git-scm.com/download/win



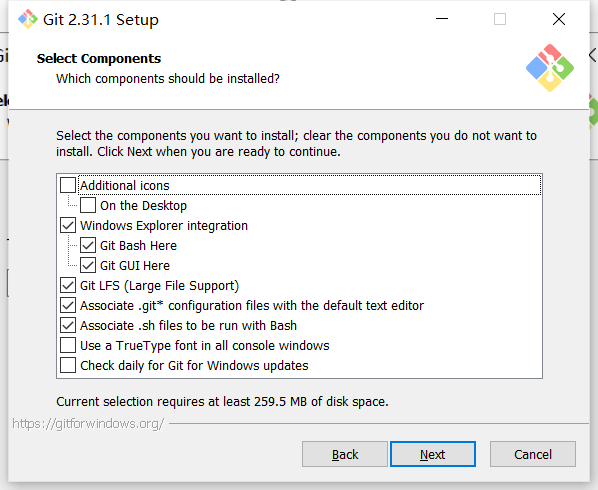
解压下载好的安装包，并打开，点击【Next】

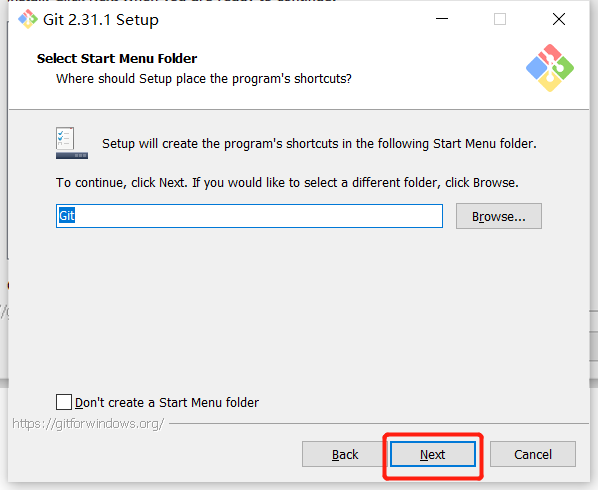


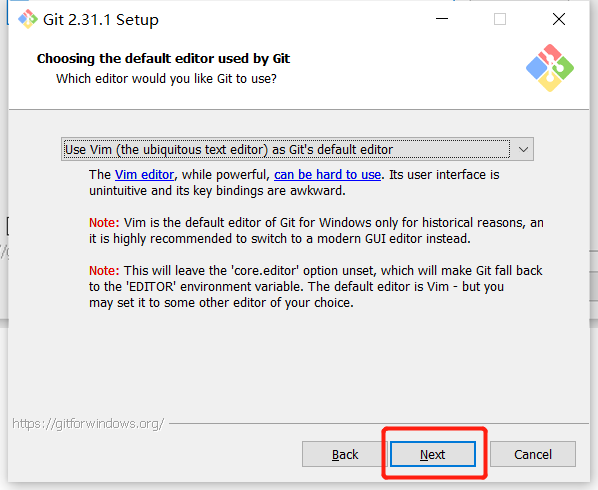
选择要安装的文件目录，最好不要装C盘。

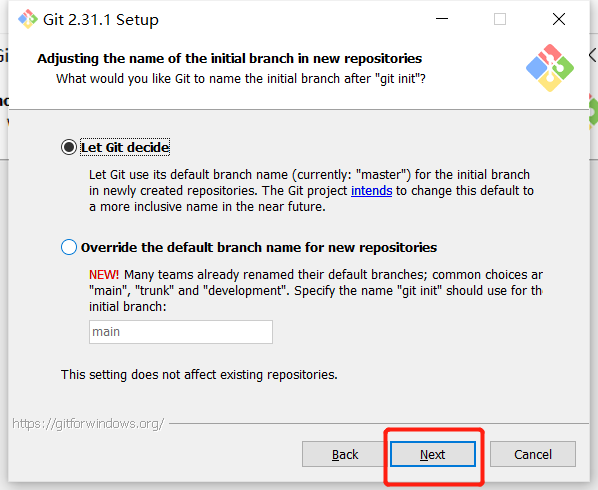


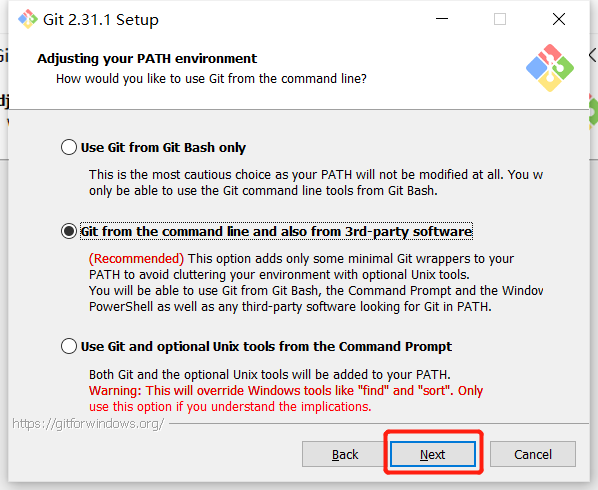
点击【Next】，傻瓜式安装即可。

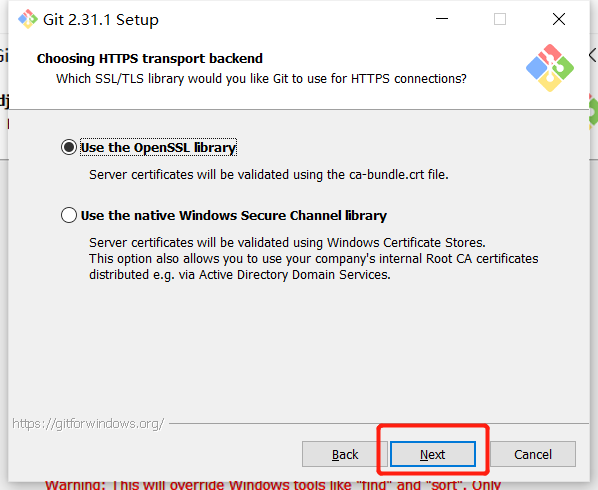


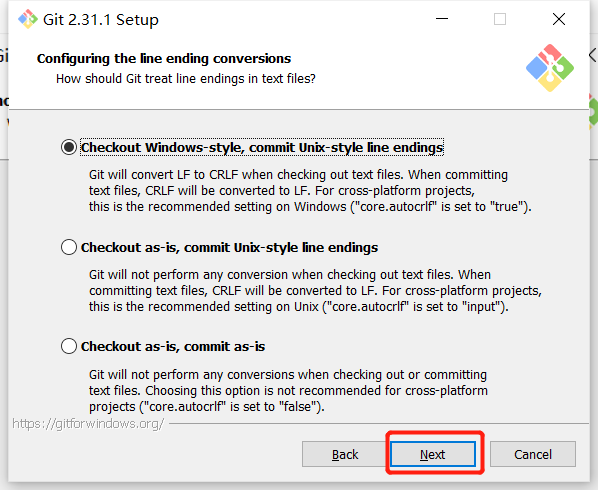


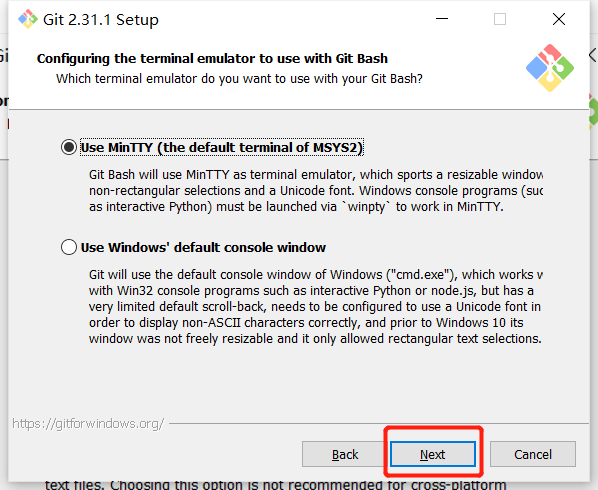


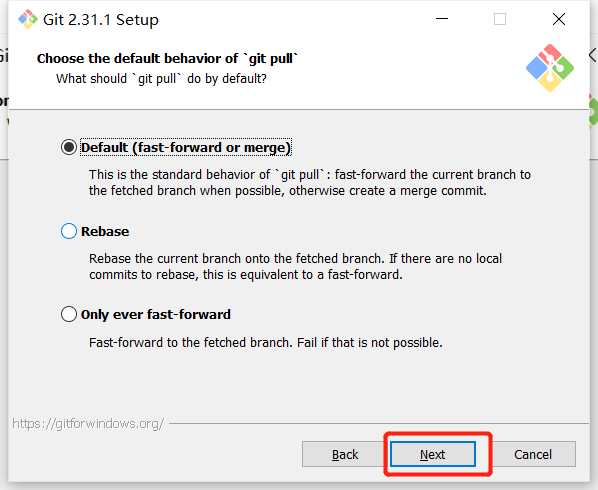


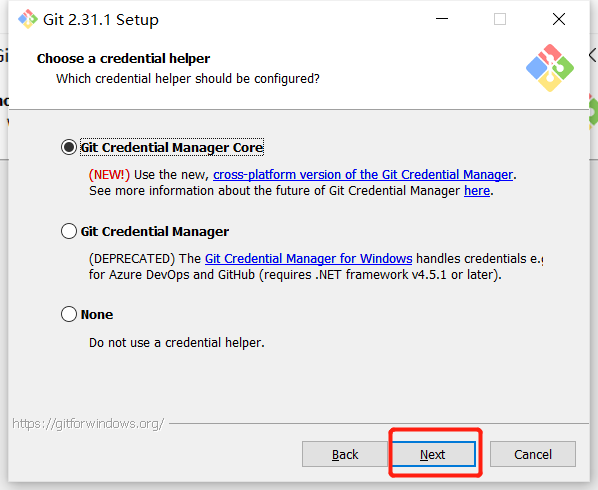


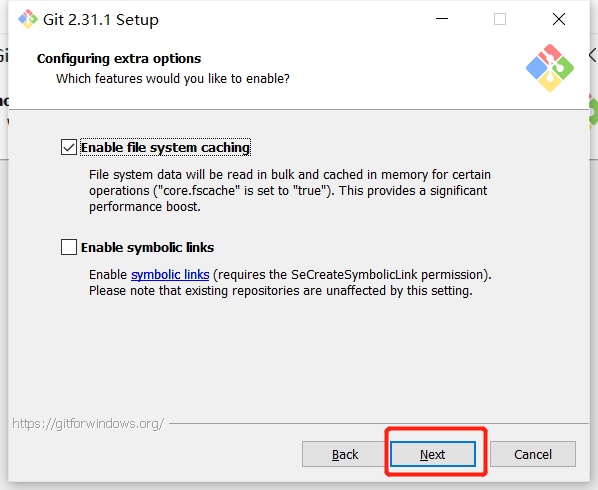


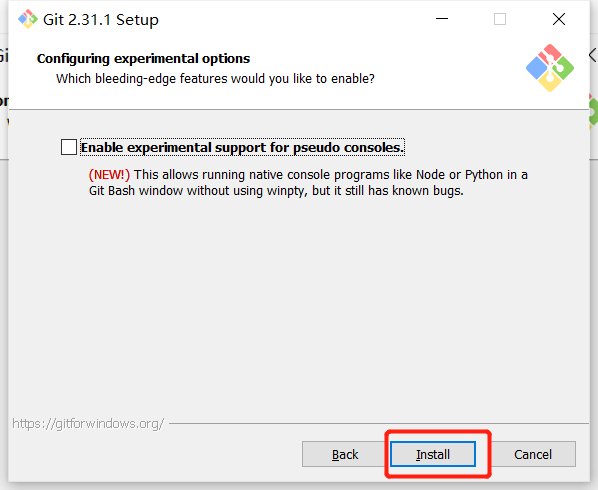


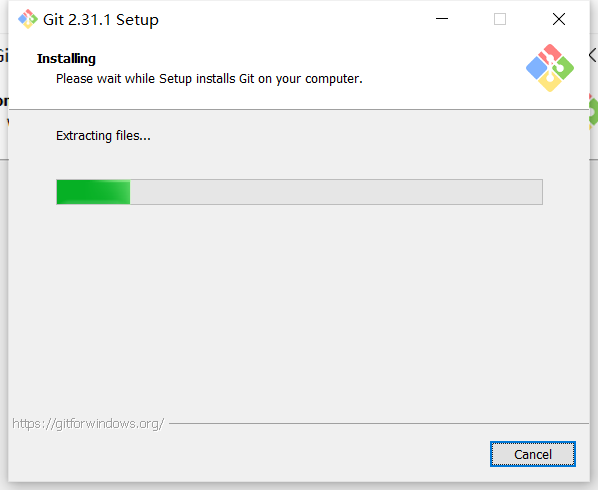


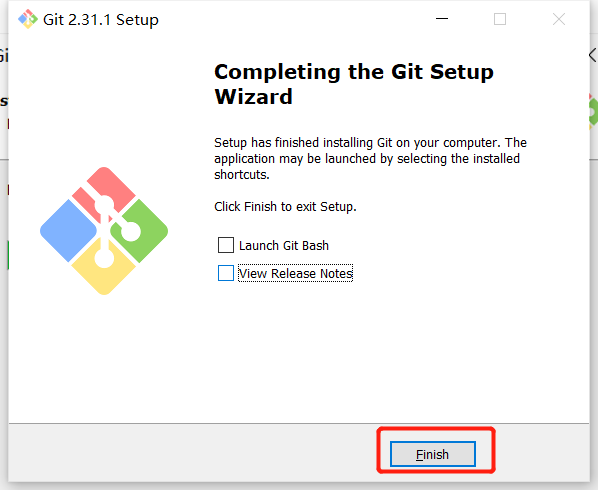












## Git简介

·分布式版本控制工具

### 何为版本控制工具

·版本控制是一种记录文件内容变化，以便将来查阅特定版本修订情况的系统

·可以记录文件修改历史记录

·可查看历史版本

·方便版本切换

### 为什么要用版本控制工具

·更好的团队协作

### 代码托管中心

·远程库

- 局域网

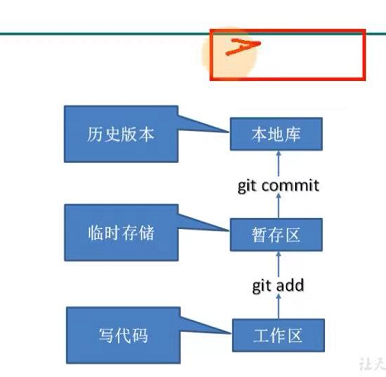
-- GitLab

- 互联网

-- Github

-- Gitee

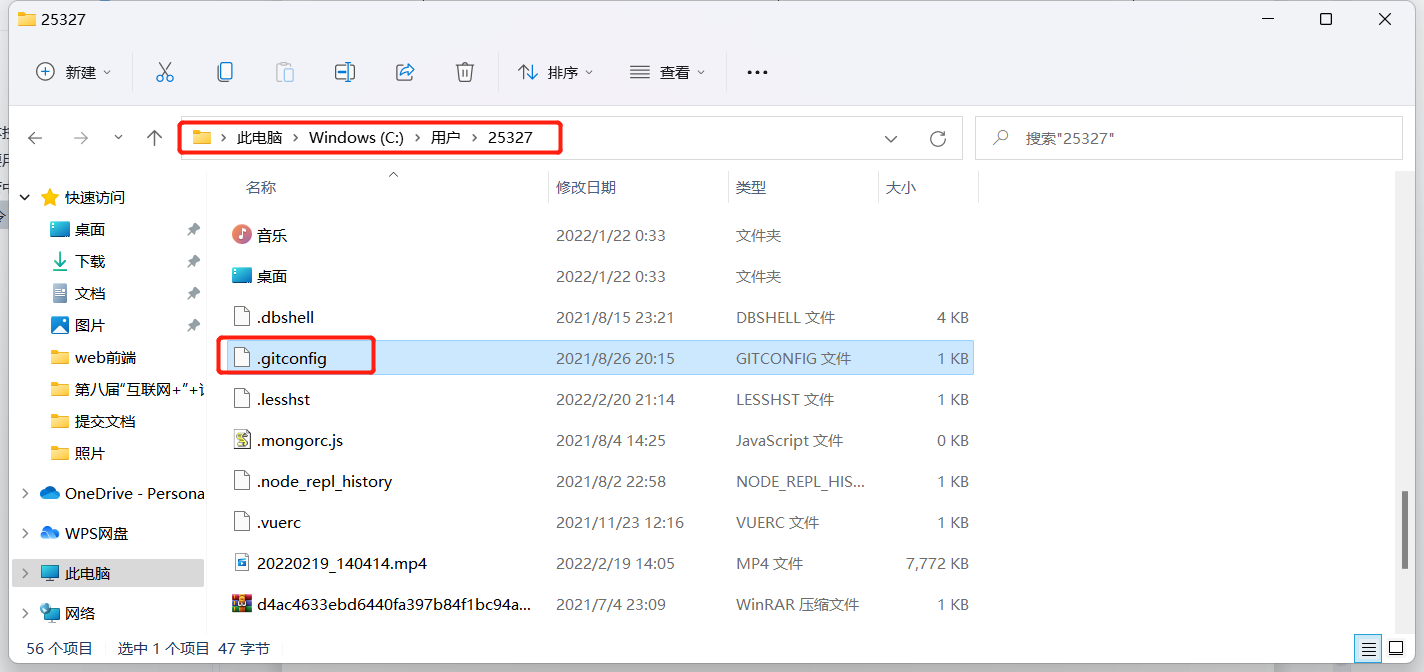
### Git工作机制



## Git常用命令

|  |  |
| --- | --- |
| **命令语句** | **作用** |
| git config --global user.name 用户名 | 设置用户签名 |
| git config --global user.email 邮箱 | 设置用户签名 |
| git init | 初始化本地库 |
| git status | 查看本地库状态 |
| git add 文件名 | 添加到暂存区 |
| git commit -m “日志信息” 文件名 | 提交到本地库 |
| git reflog | 查看历史记录 |
| git reset --hard 版本号 | 版本穿梭 |

·用户签名保存地址



·这里的用户签名和将来登录Githun（或其他代码托管中心）的账号没有任何关系

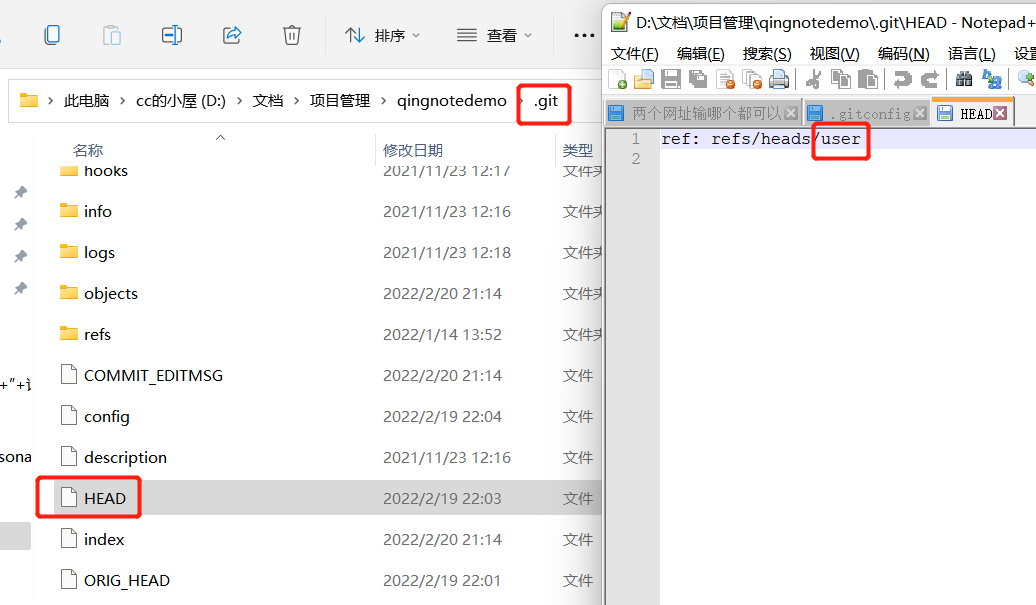
·git rm --cached 文件名

- 将暂存区的某文件删除，工作区还有

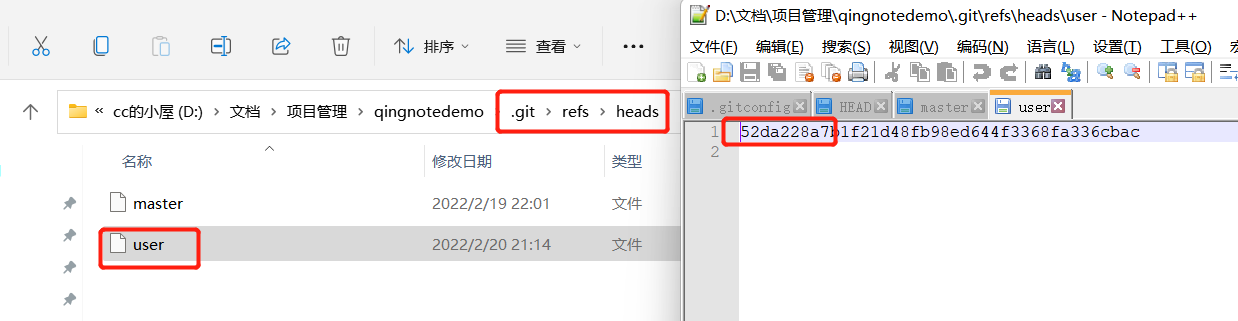
·git log

- 查看详细日志

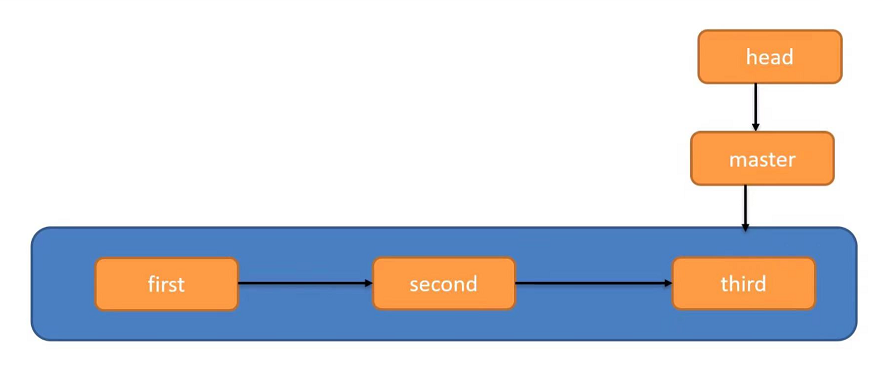
·当前分支存储文件



·查看分支指向的版本号



·git切换版本，底层其实是移动的HEAD指针，原理如下



## Git分支

·版本控制过程中有多个任务，可以创建每个任务的单独分支

·分支会与主线分离，开发分支不会影响主线分支运行

### 什么是分支？

·分支可以理解为项目副本

·分支底层也是指针的引用

### 分支的好处

·同时推进多个功能开发，提高开发效率

·某一分支开发失败，不会对其他分支有任何影响。失败的分支删除重新开始即可

### 分支的操作

|  |  |
| --- | --- |
| **命令语法** | **作用** |
| git branch 分支名 | 创建分支 |
| git branch -v | 查看分支 |
| git chrckout 分支名 | 切换分支 |
| git merge 分支名 | 将指定分支合并到当前分支 |

### 冲突产生的原因

·合并分支时，两个分支在同一个文件的同一个位置有两套完全不同的修改，git无法确定使用哪个，必须人为决定新代码内容

### 解决冲突

·将冲突文件

·修改为

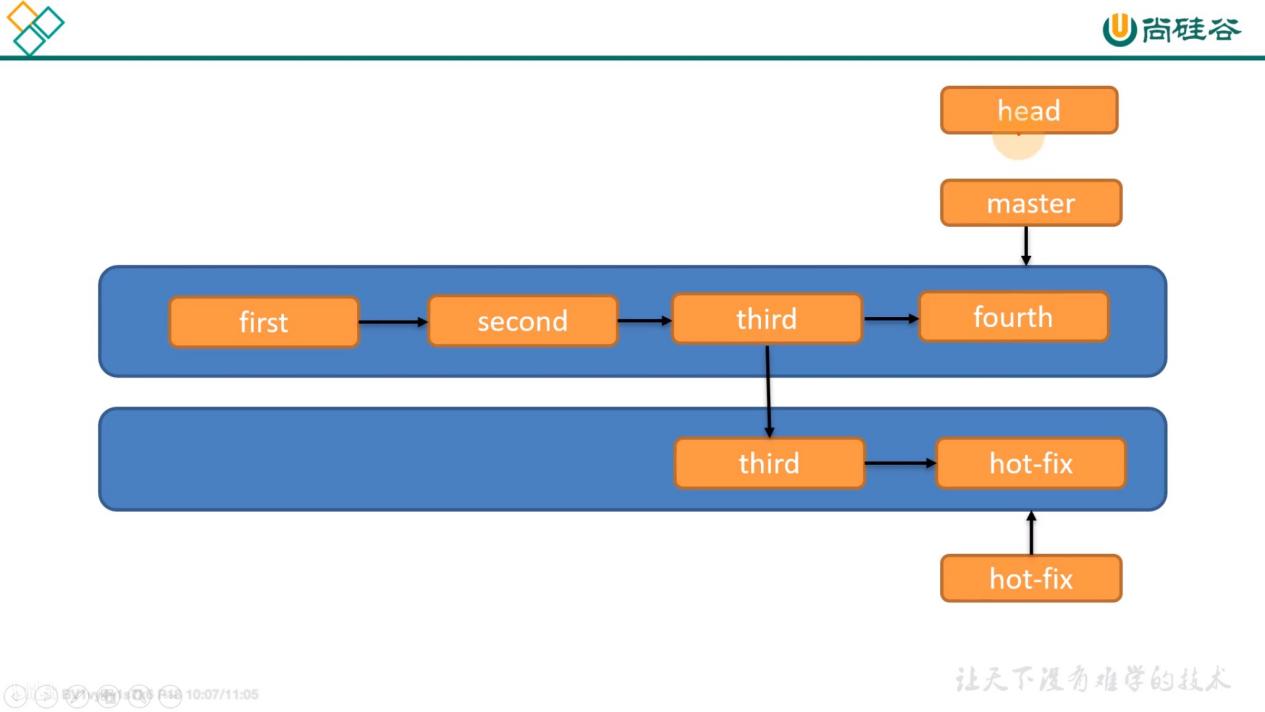
·添加到暂存区

- git add 文件名

·执行提交

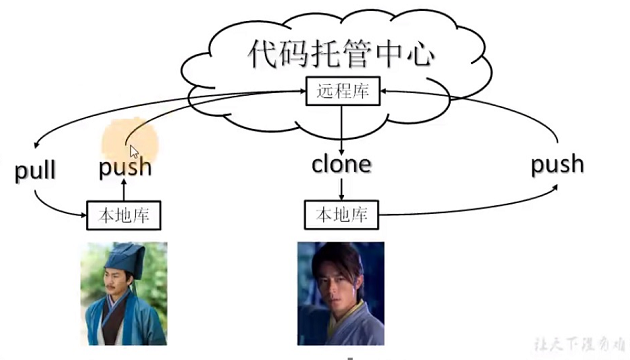
- git commit -m “备注”

### 创建分支和切换分支图解

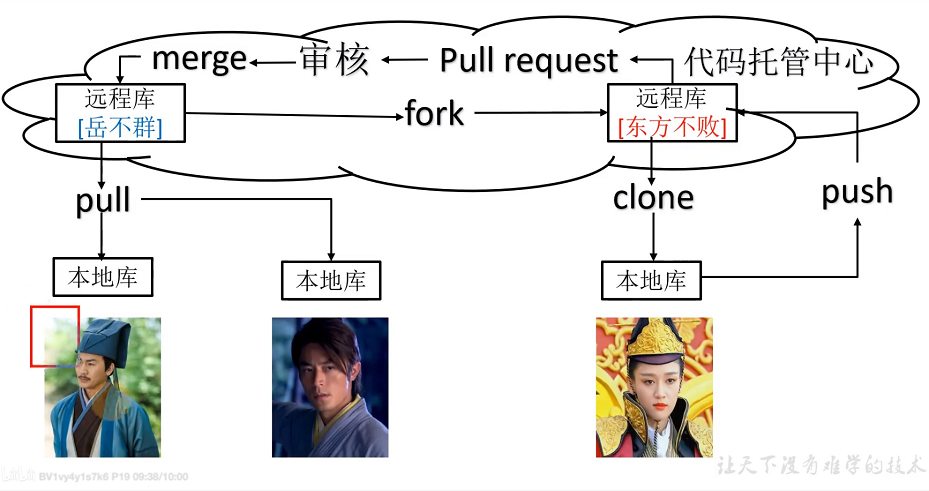


## Git团队协作机制

### 团队内协作



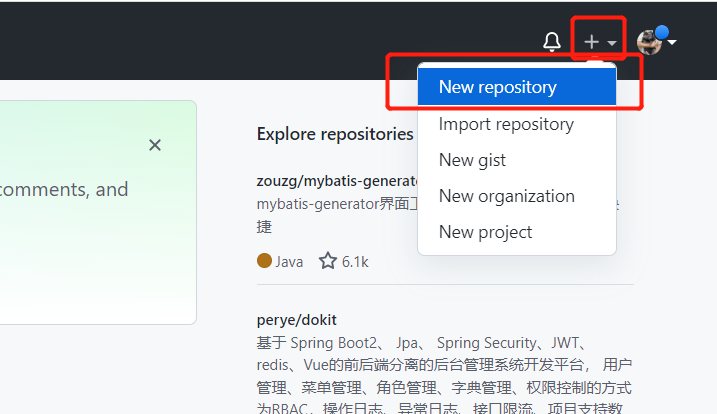
### 跨团队协作

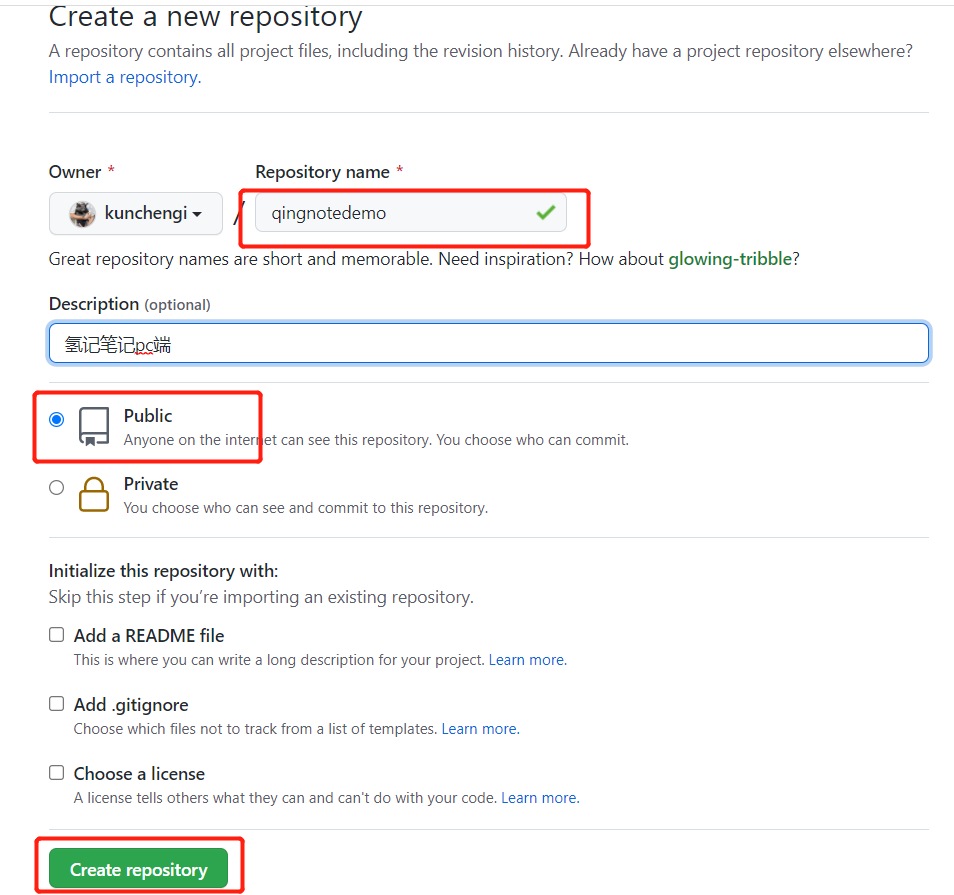


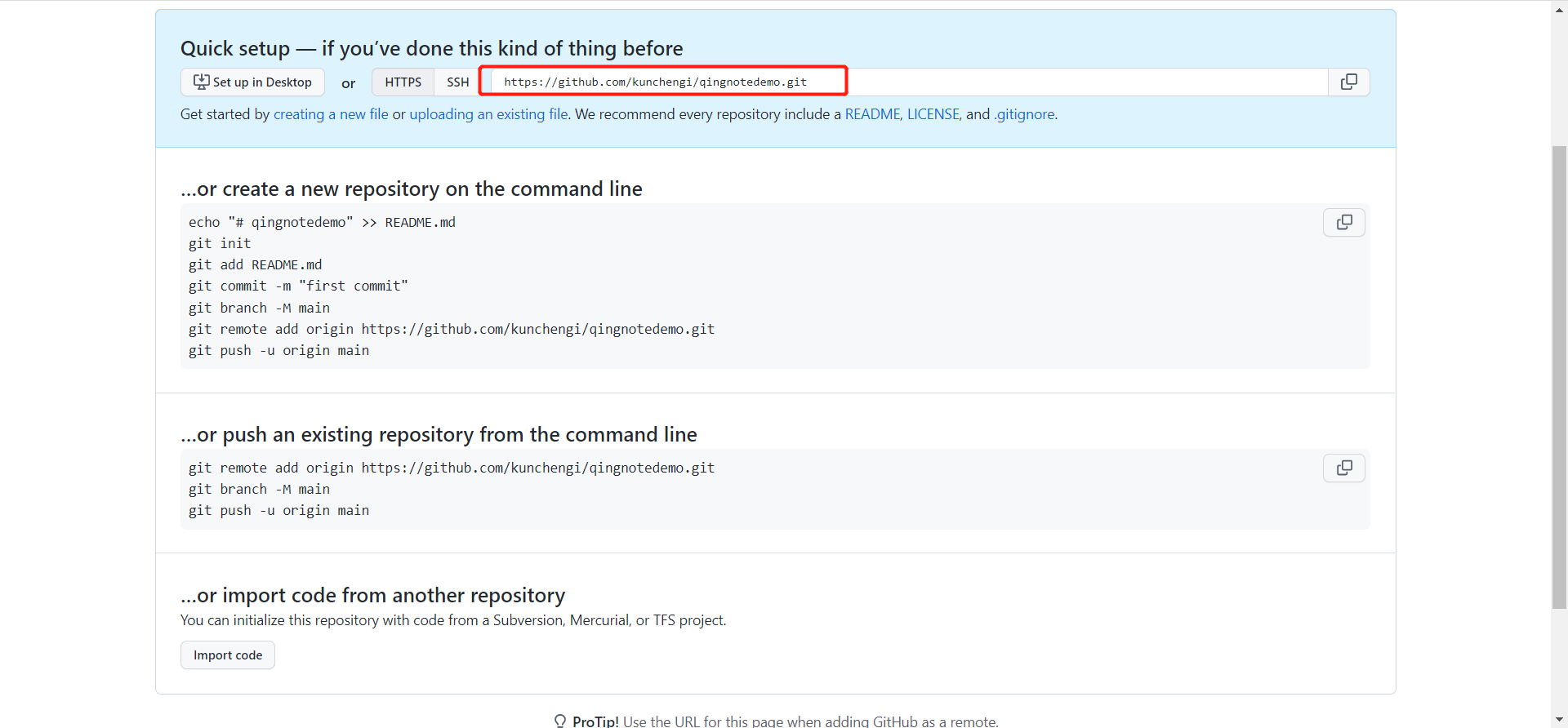
# GitHub

## GitHub操作

### 创建远程库







·git remote -v

- 查看当前所有远程地址别名

·git remote add 别名 远程地址

- 给远程地址取别名

- git remote add qingnote <https://github.com/kunchengi/qingnotedemo.git>

### 推送本地库到远程库

·git push 别名 分支

- 将本地库的某分支推送到远程库

- git push qingnote master

- 首次推送会弹出登录框登录github

### 拉取远程库到本地库

·当团队成员提交了新的代码时，需要拉取来更新本地库

·git pull 别名 分支

- 将远程库的某分支拉取到本地库，用来更新本地库

- git pull qingnote mester

### 克隆远程库到本地

·git clone 远程地址

- 将远程库项目克隆到本地

- git clone <https://github.com/asxldb/iceEditor.git>

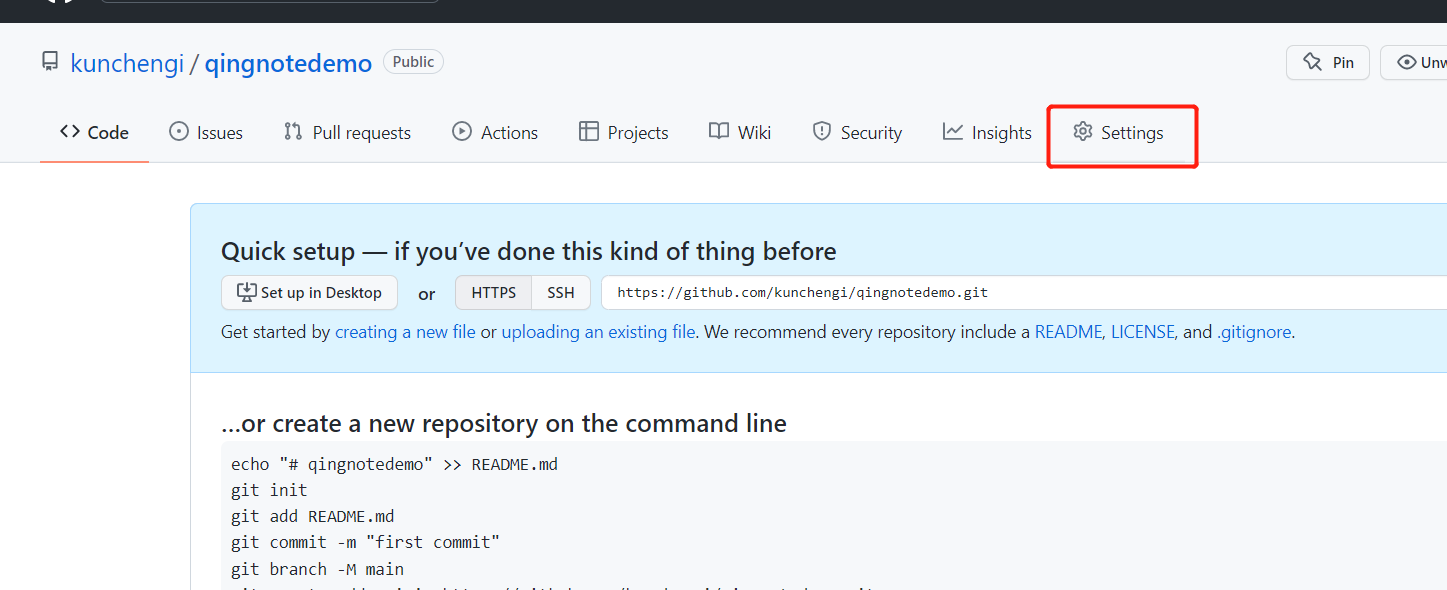
- 克隆代码是不需要登录账号的

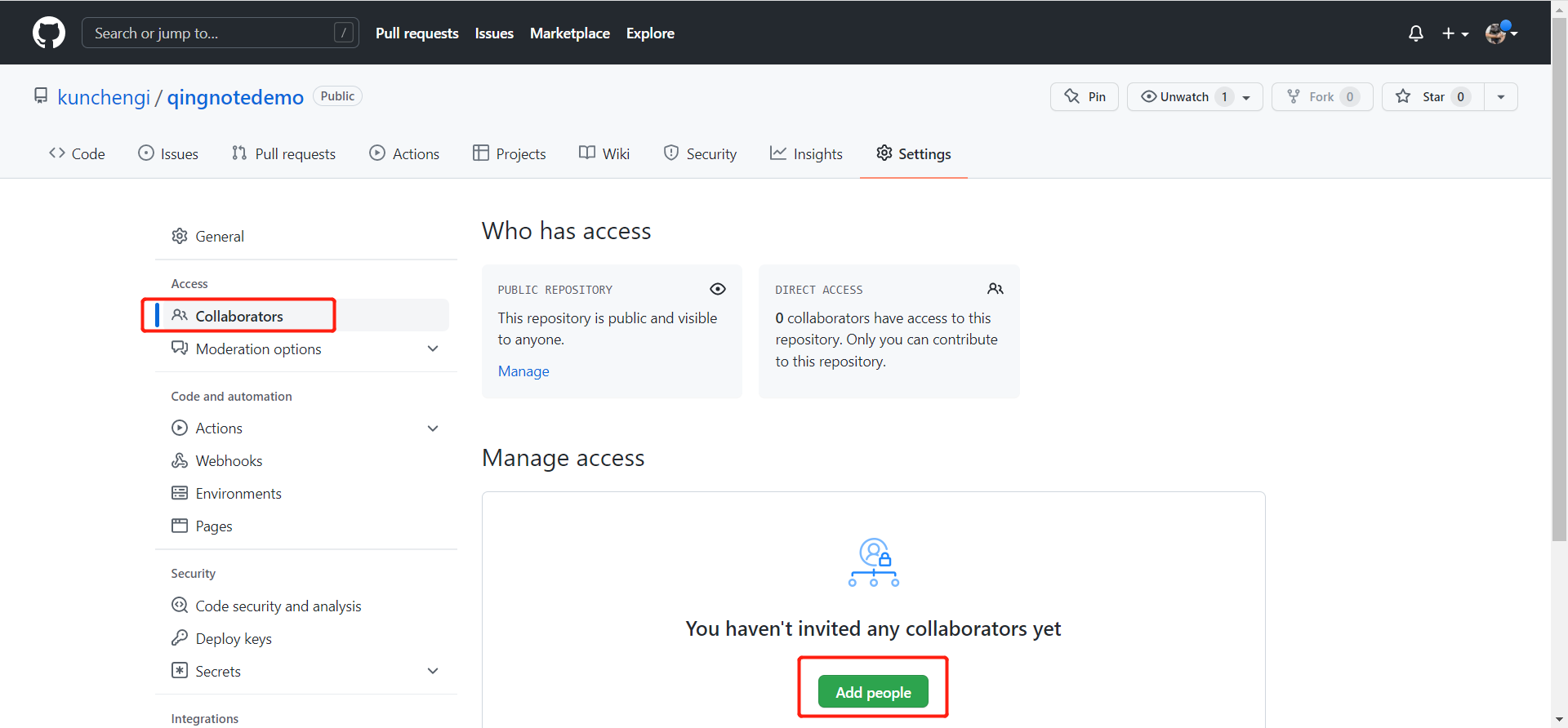
- 会自动初始化git本地库

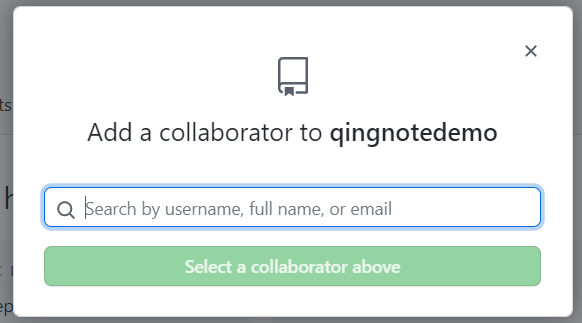
- 会自动创建别名origin

### 团队内协作

·在团队中的成员才有权限推送代码

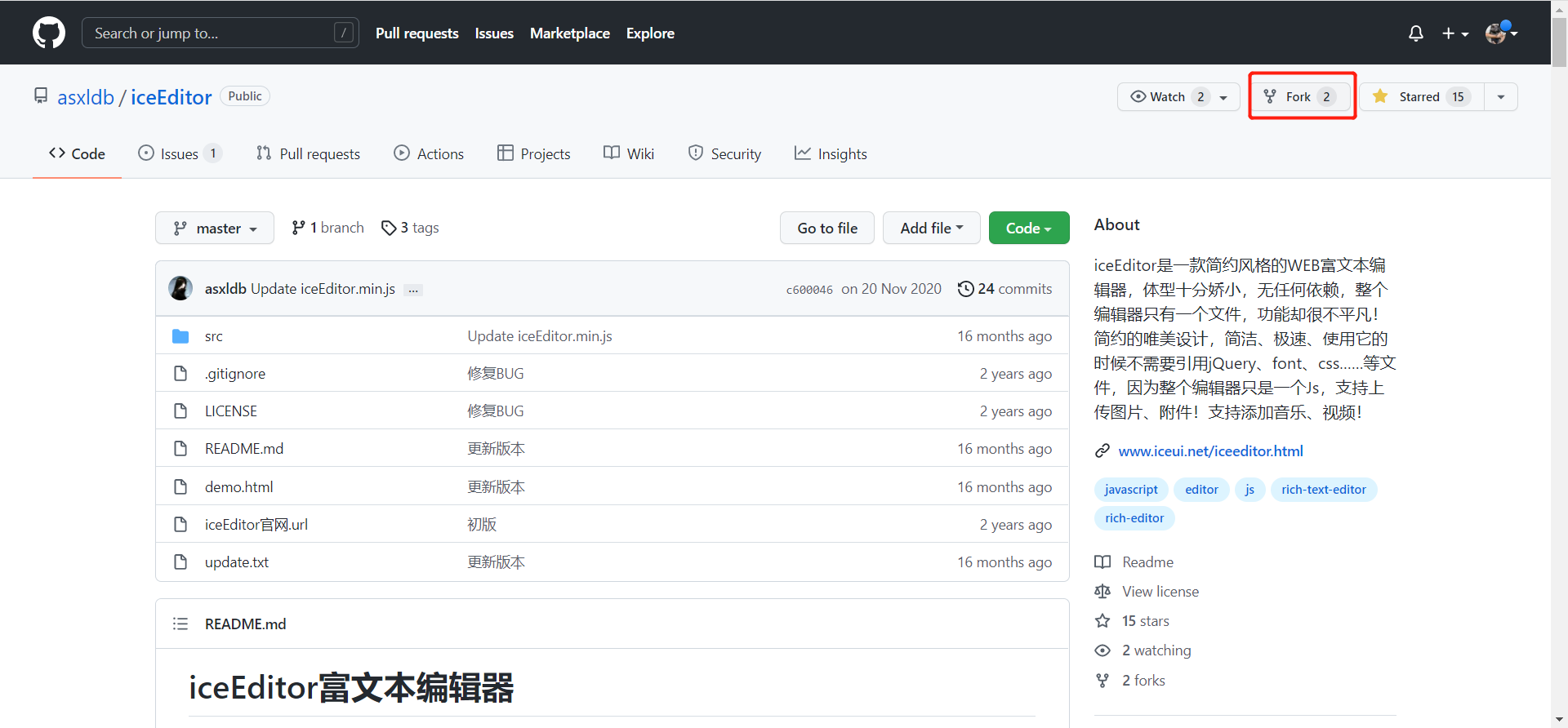






### 跨团队协作

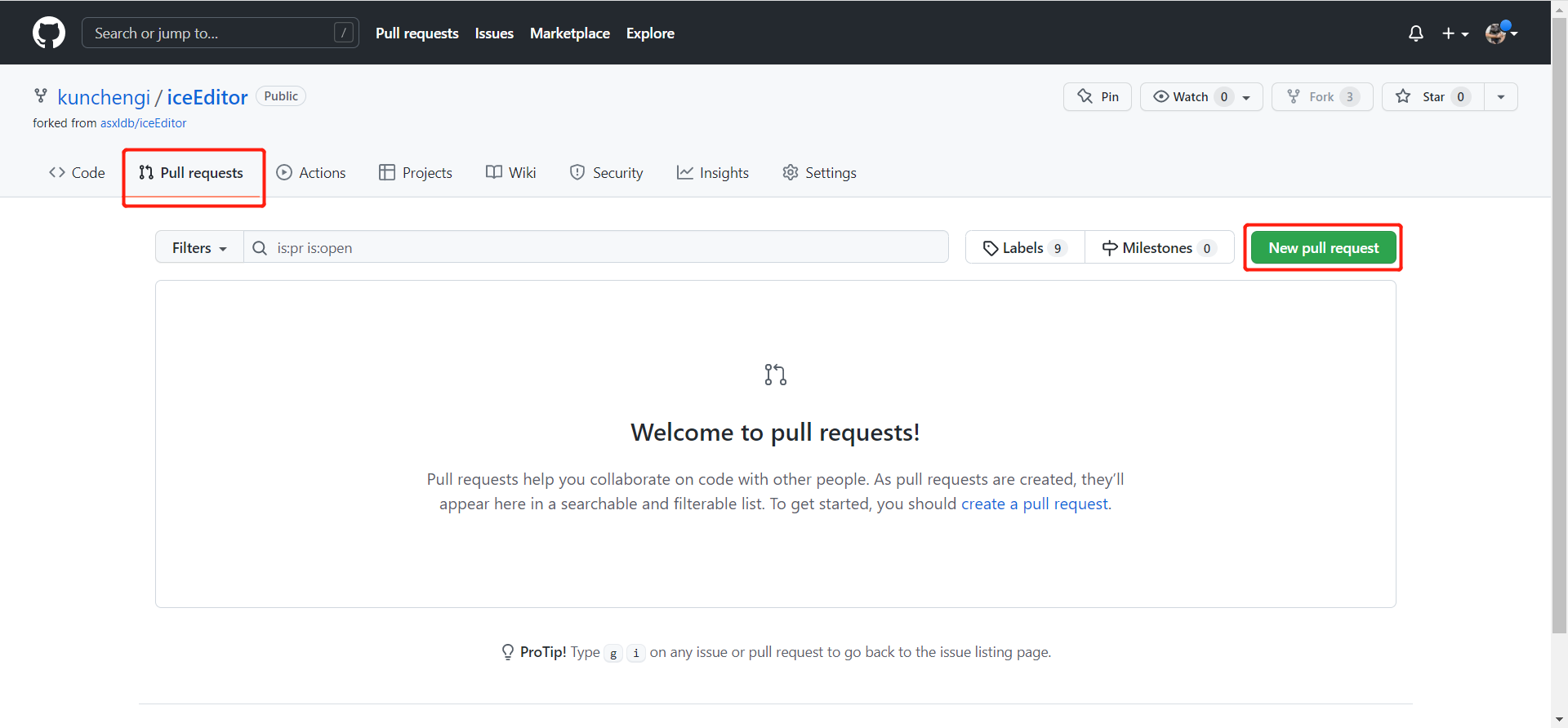
·非团队成员访问远程仓库后，点击fork





·这时就可以对代码进行修改

·克隆到本地修改完代码之后推送到远程仓库，然后点击拉取请求



·点击Create pull request

·输入信息后点击Create pull request

·仓库所有着可在pull requests 中查看别人的拉取请求

·可以在该请求中沟通

·点击Merge pull request合并代码

·点击Confirm merge确定合并

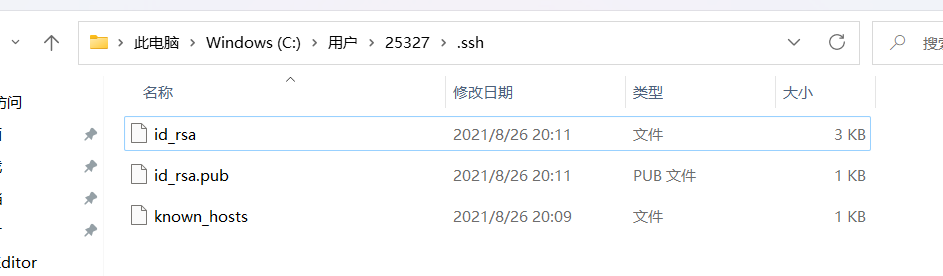
### SSH免密登录

·命令提示符中输入ssh -keygen -t rsa -C [city252@qq.com](mailto:city252@qq.com)

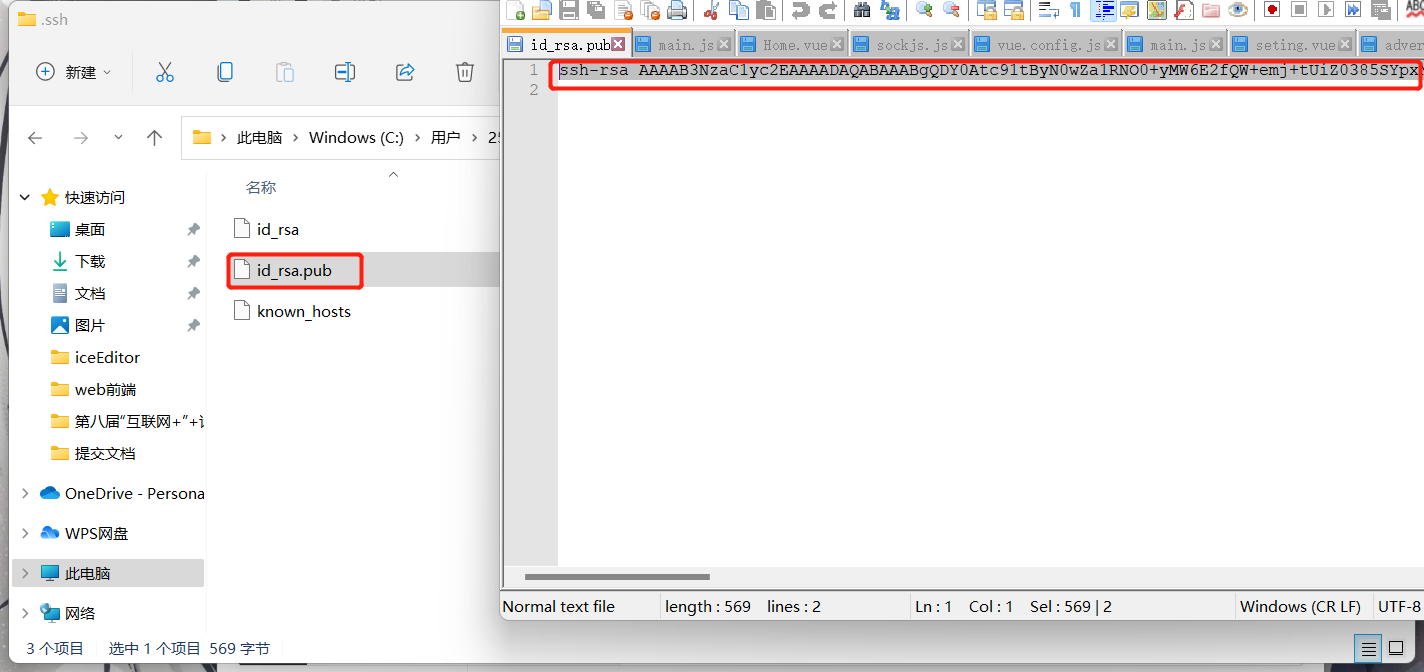
- 敲三次回车

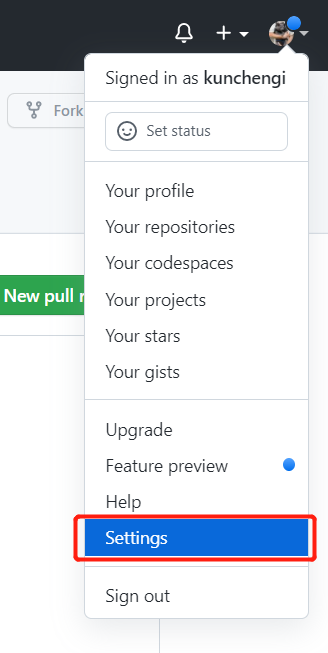
- 在本机生成shh协议

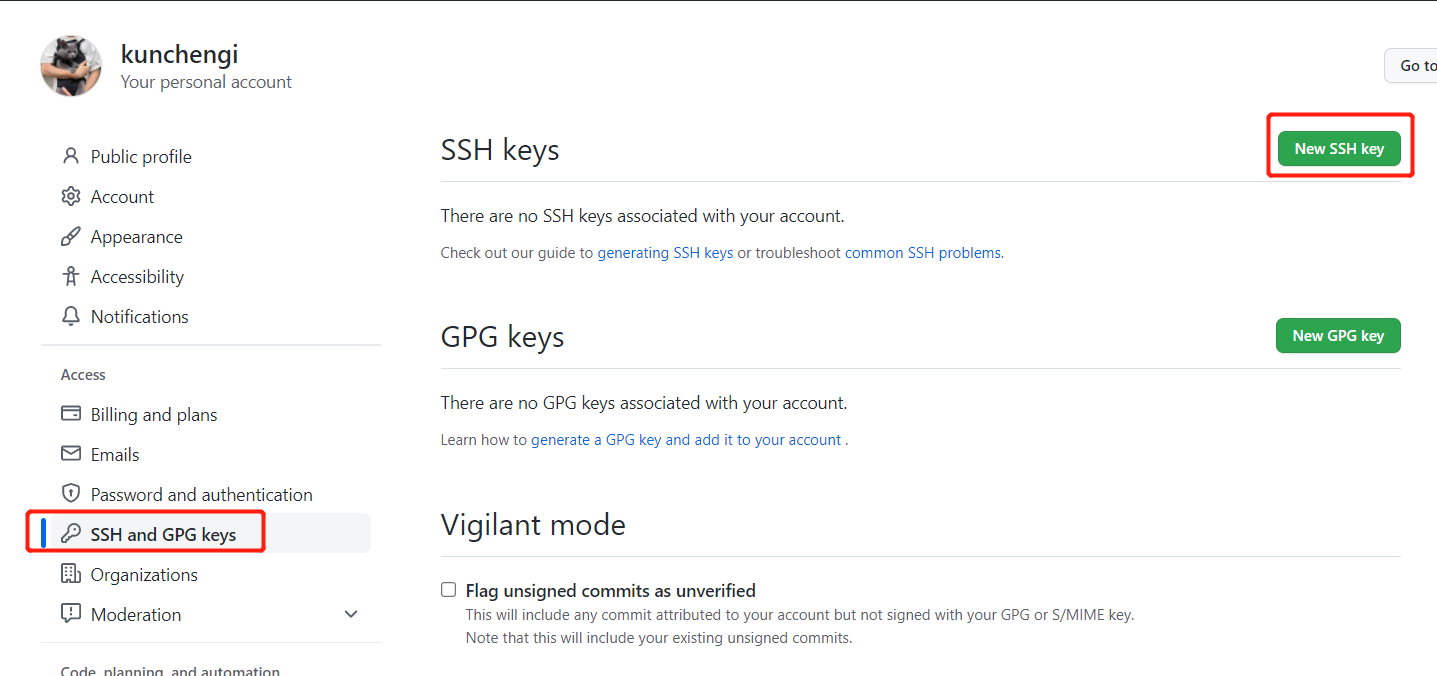
·在此路径查看shh协议文本



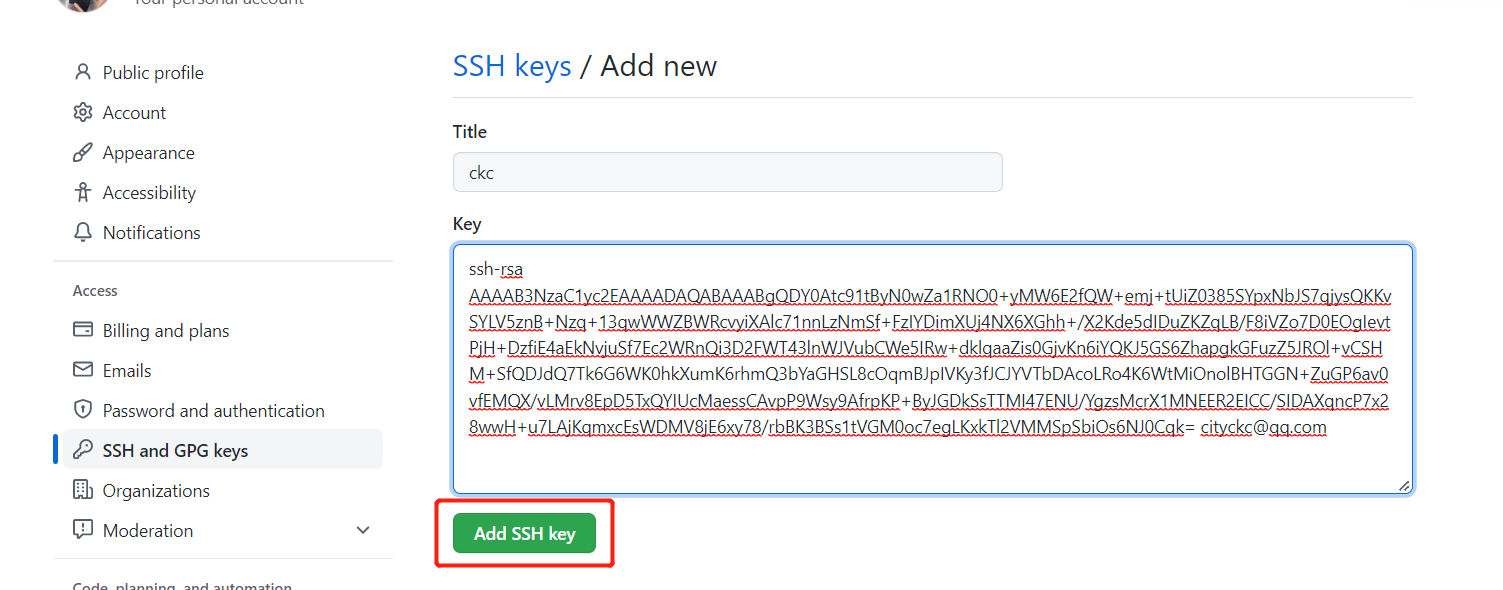
·将文本内容全选复制







将刚刚复制的代码粘贴到此处

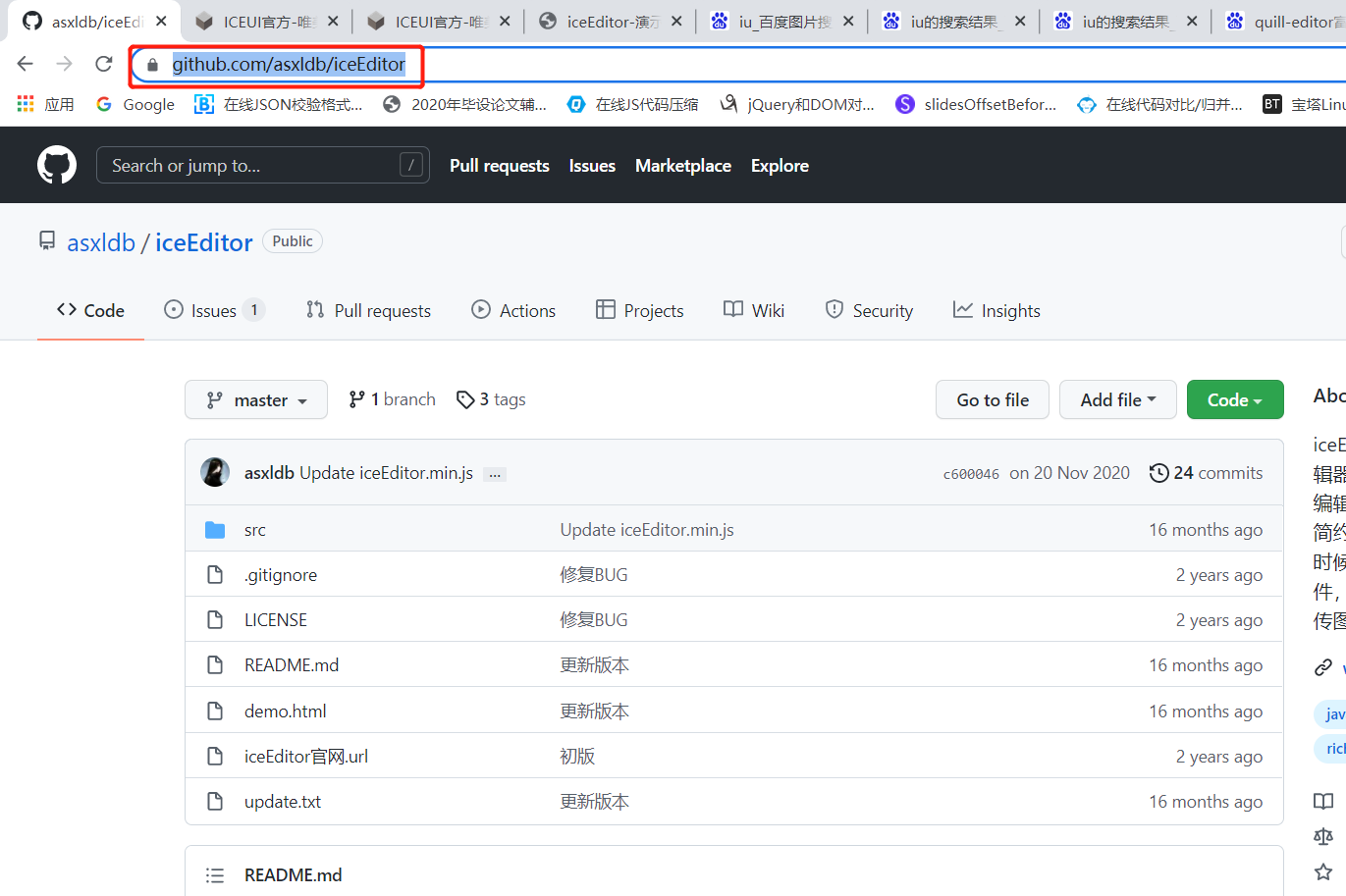


# Gitee

## Gitee操作

### 导入github项目







### 更新github修改的内容



# GitLab

## 安装

·暂时不学