

Git

Git 是目前世界上最先进的分布式版本控制系统（没有之一）。

一、为什么需要版本控制工具

1. 因为开发中会出现代码的修改，修改前的代码如果没有版本管理，那么是无法恢复。

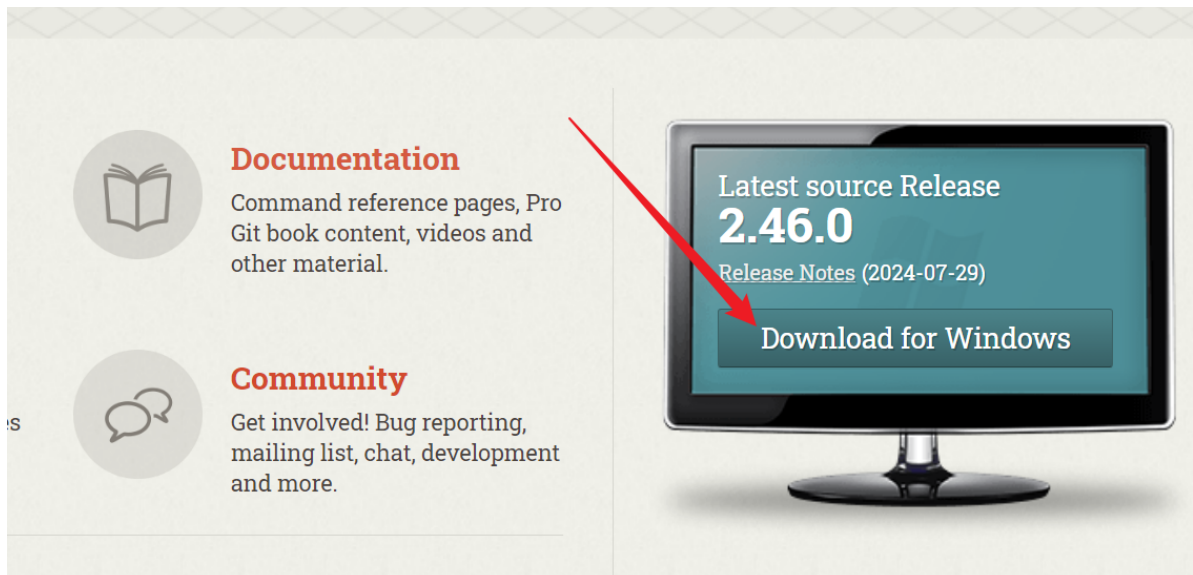
所以使用版本控制工具之后，我们可以给代码加上版本，这样就可以不同版本之间切换。

2. 项目开发不是一个人单打独斗，是一个团队协作完成，团队之间也需要代码的合并，共享等，此时也需要版本控制工具

二、下载和安装

1. 下载地址

<https://git-scm.com/>



Download for Windows

[Click here to download](#) the latest (2.46.0) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 22 days ago, on 2024-07-29.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

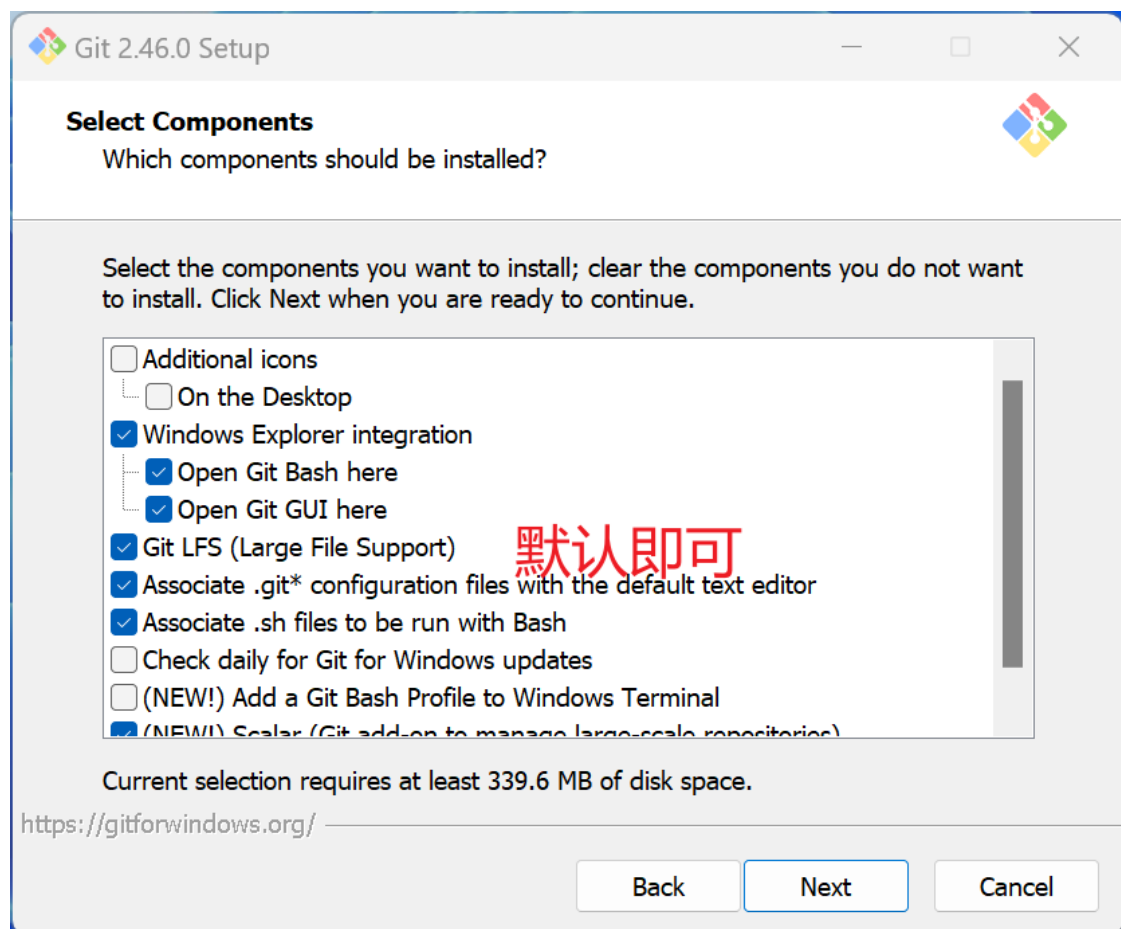
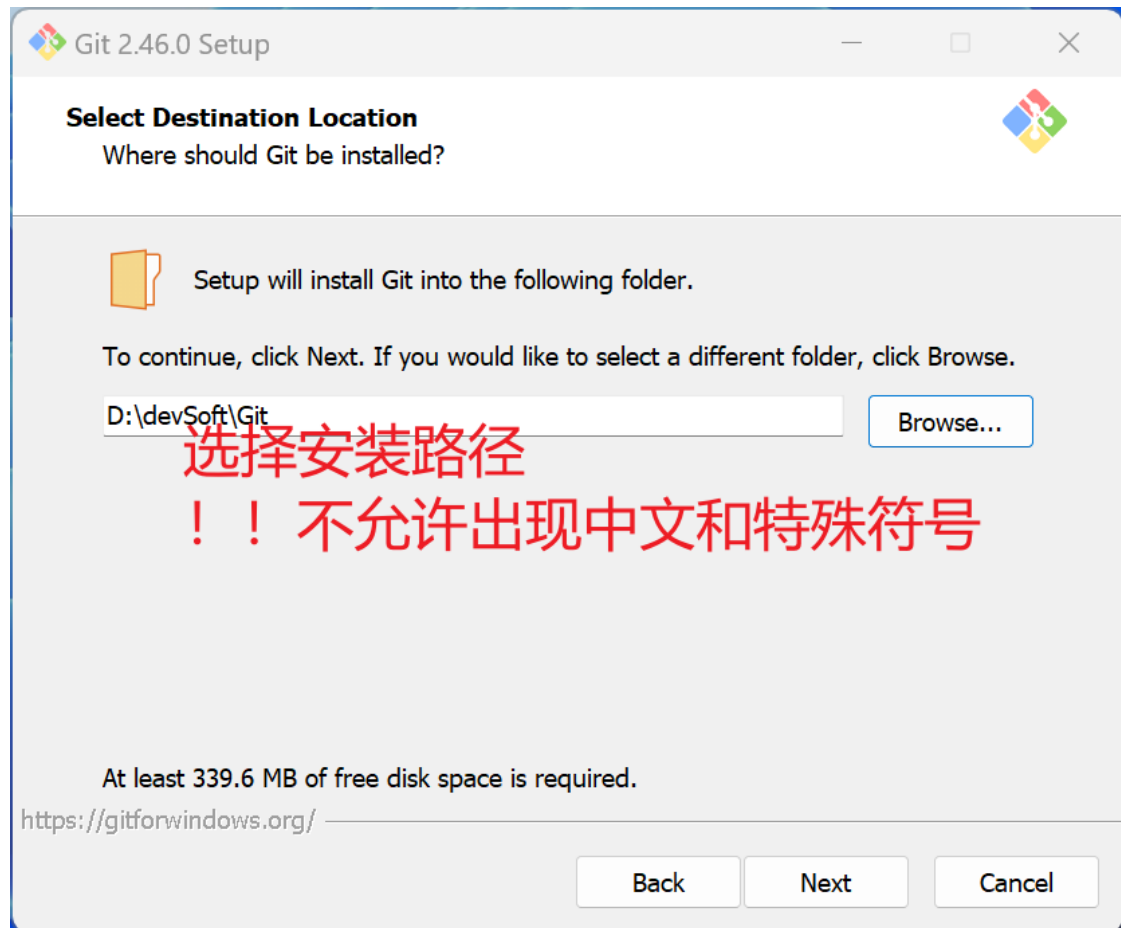
[64-bit Git for Windows Setup.](#) ← 下载这个

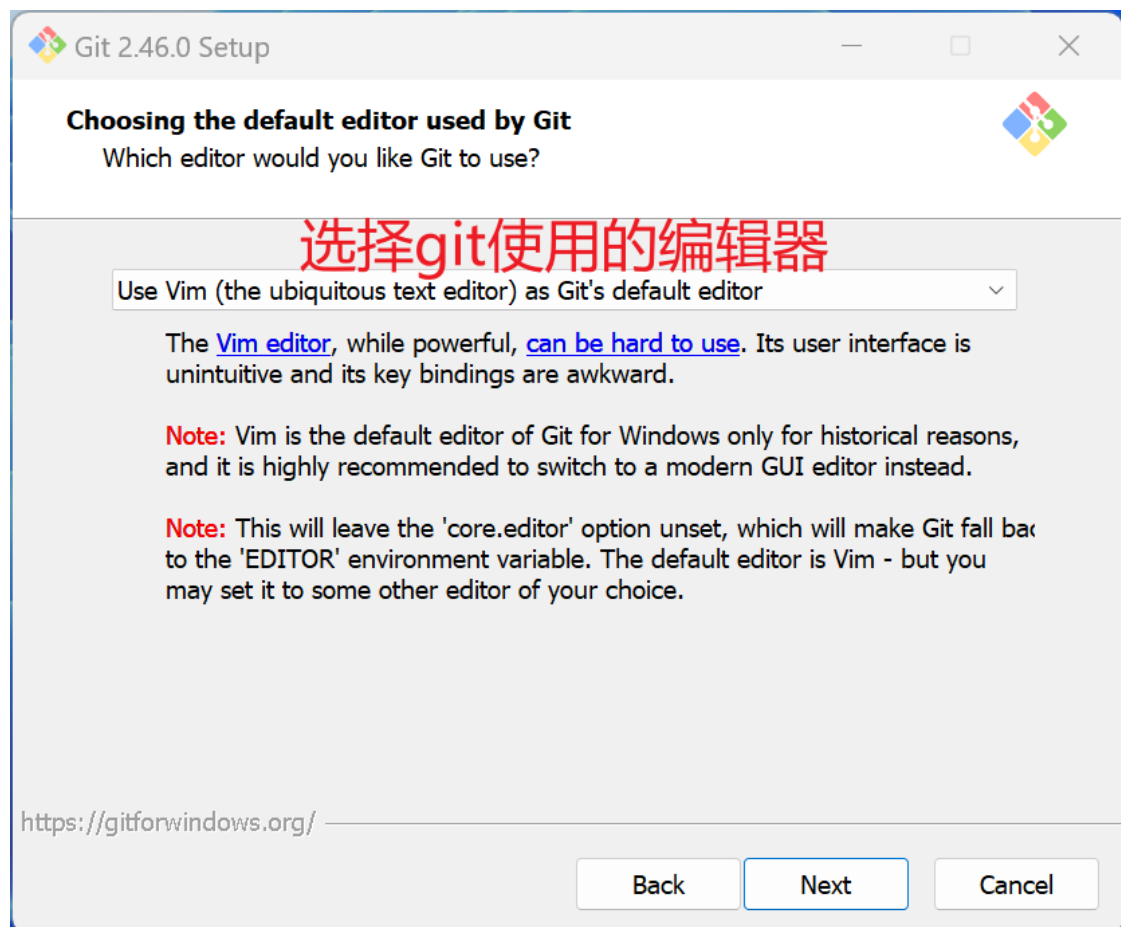
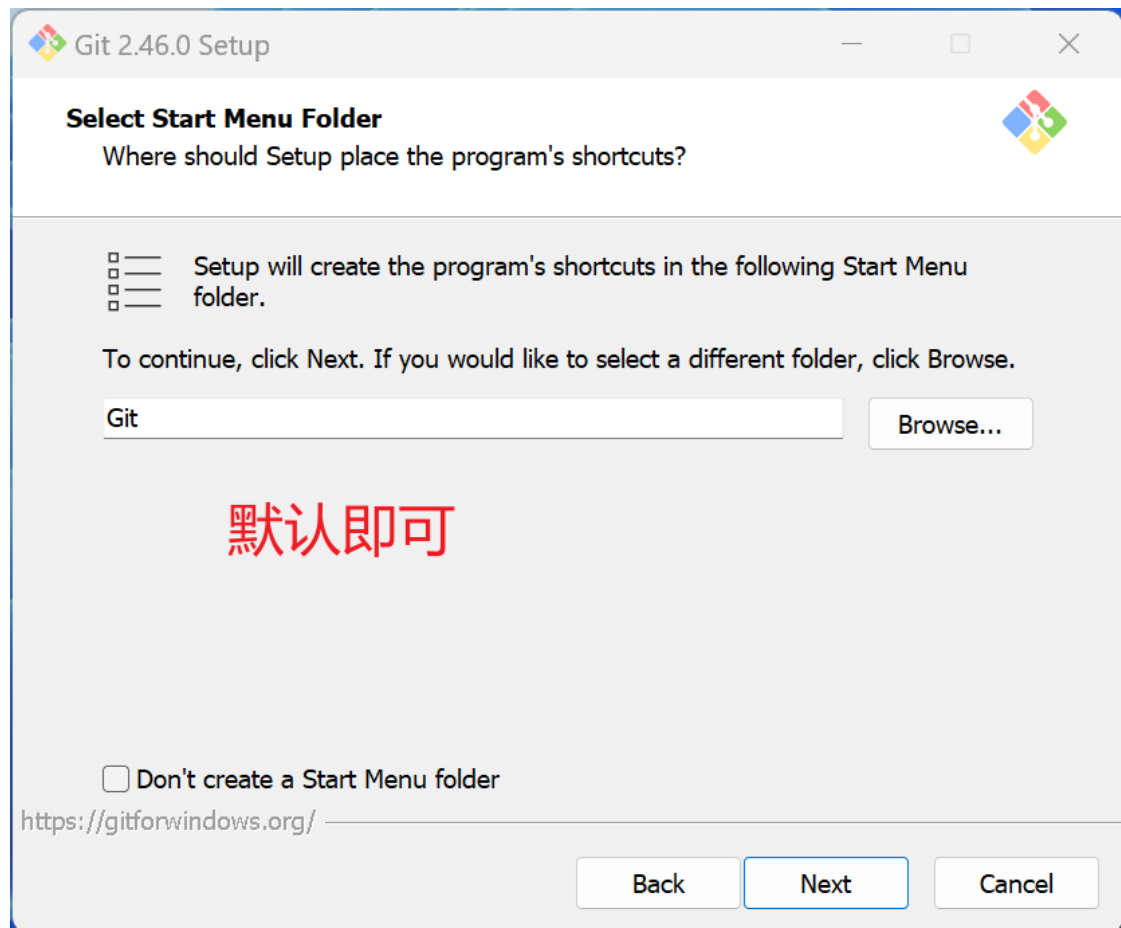
Portable ("thumbdrive edition")

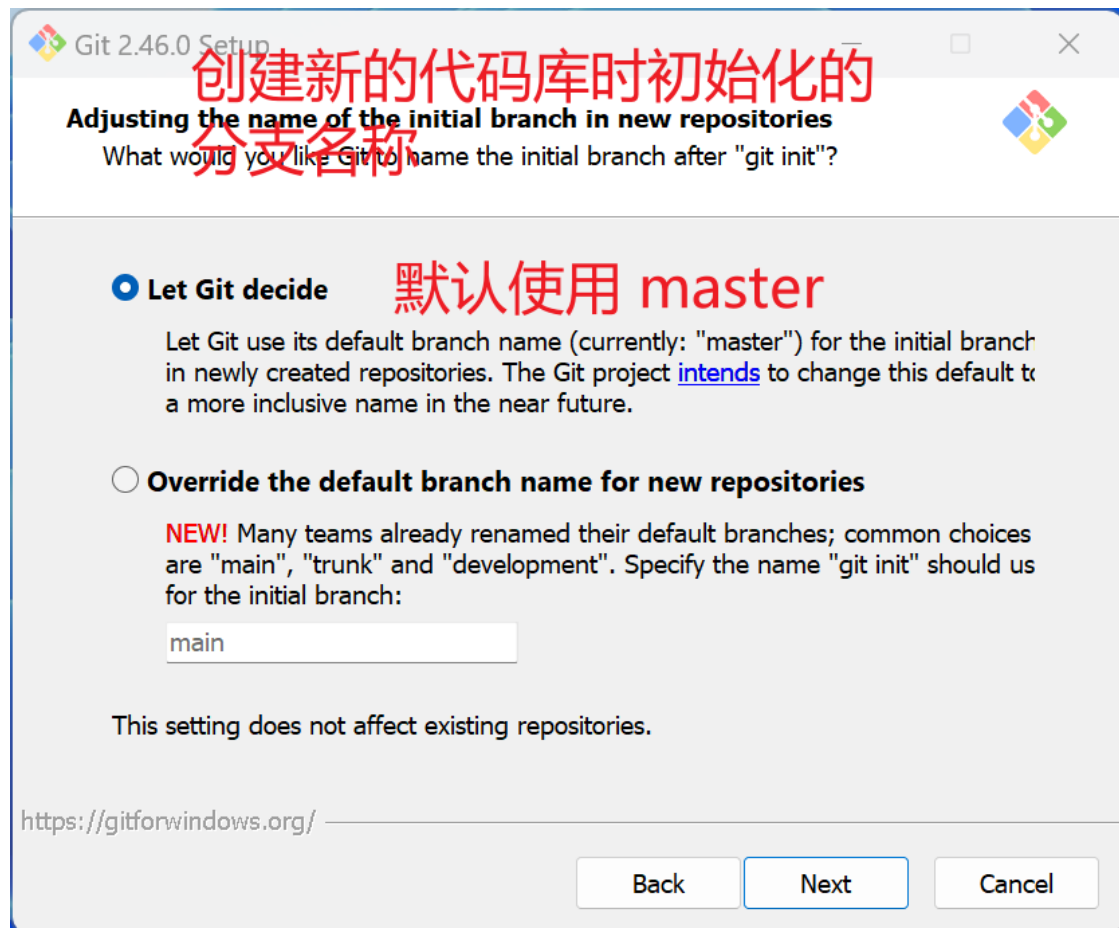
[32-bit Git for Windows Portable.](#)

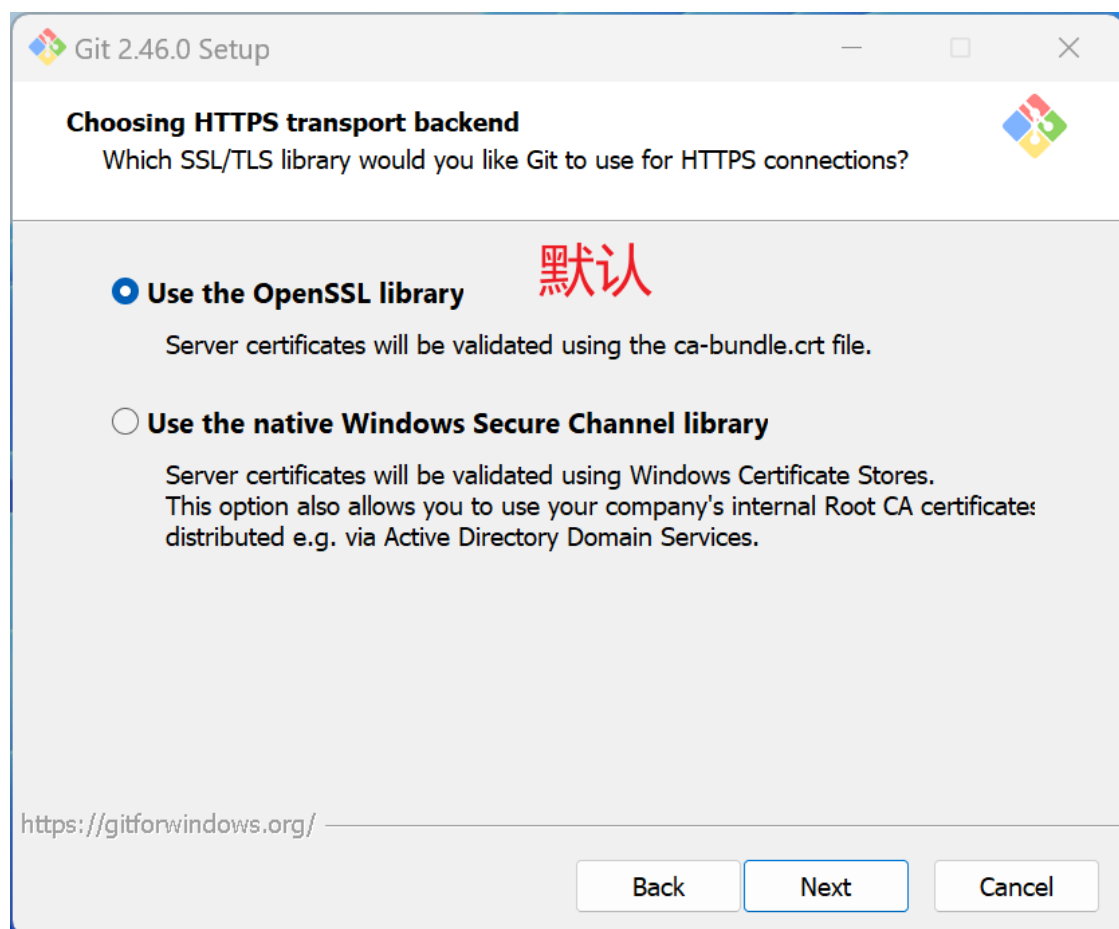
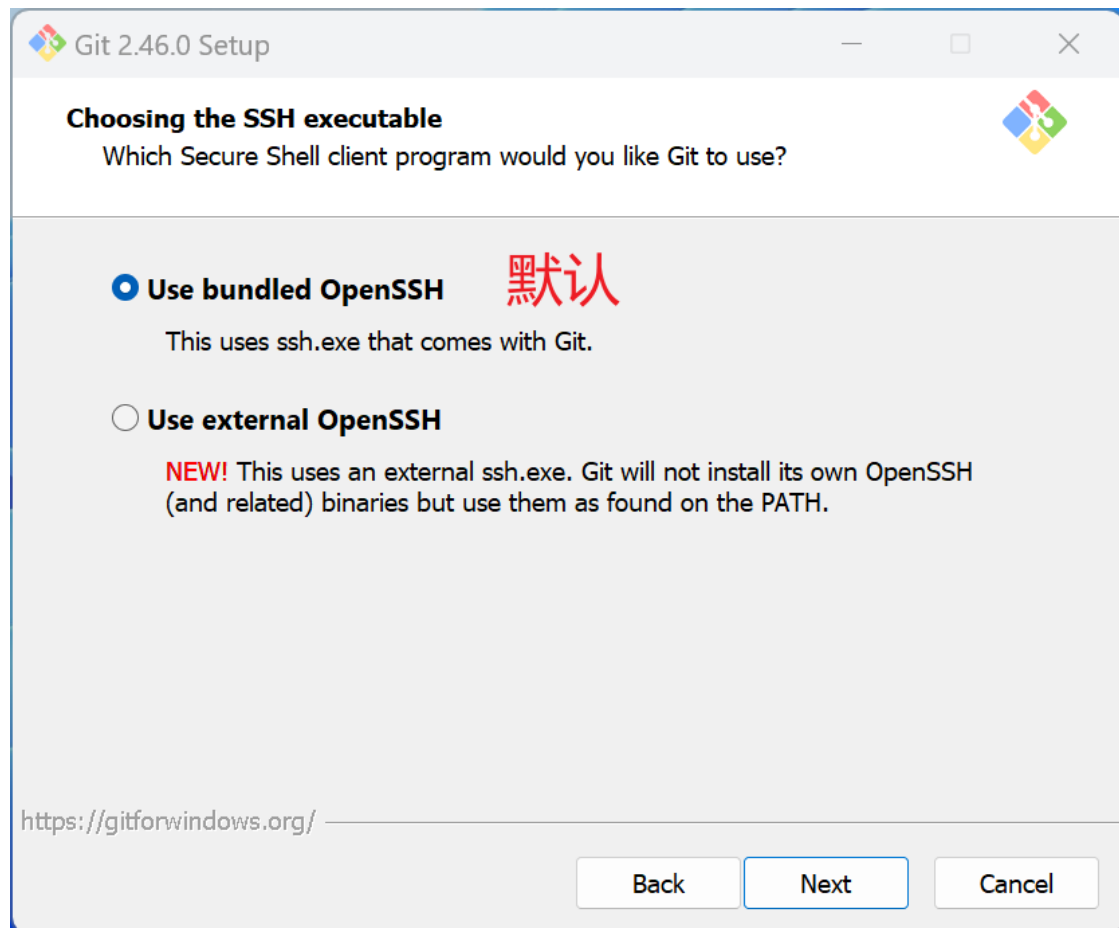
[64-bit Git for Windows Portable.](#) 便携式版本

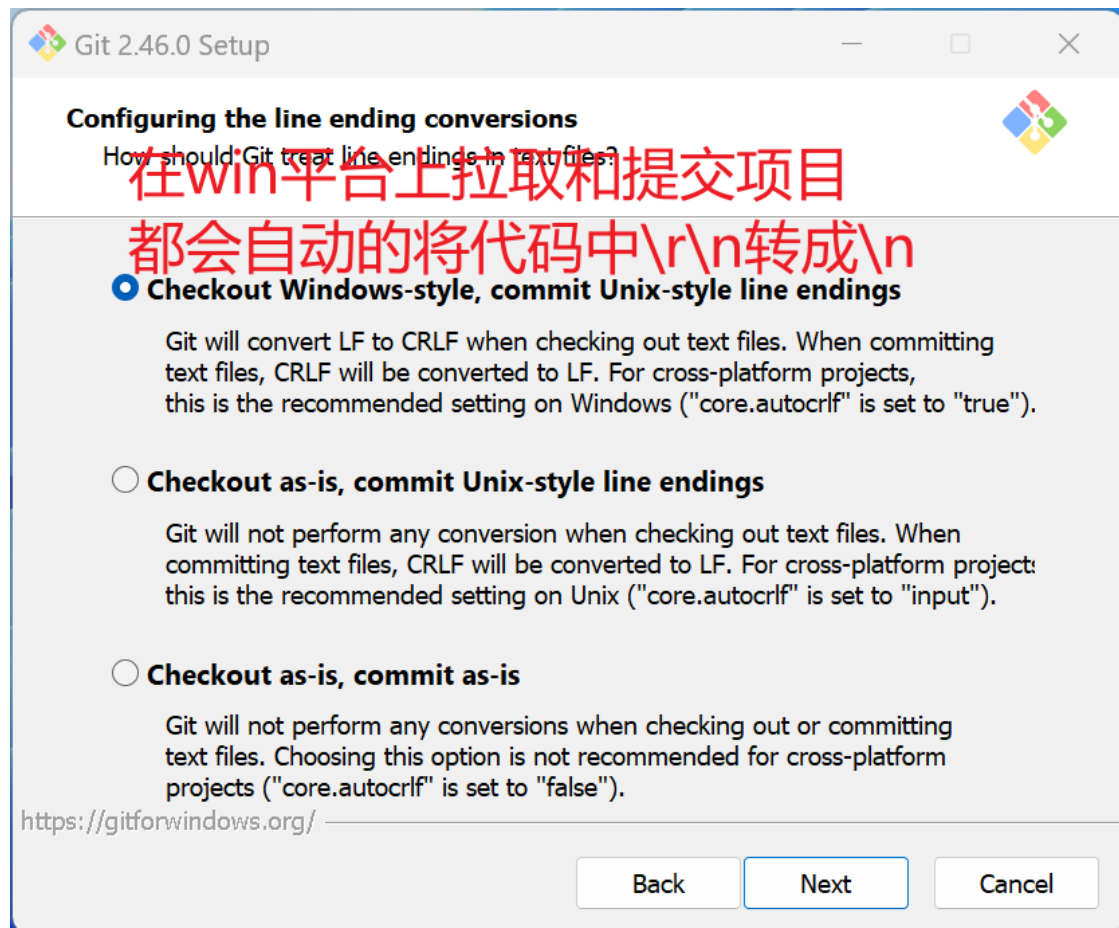
2. 安装

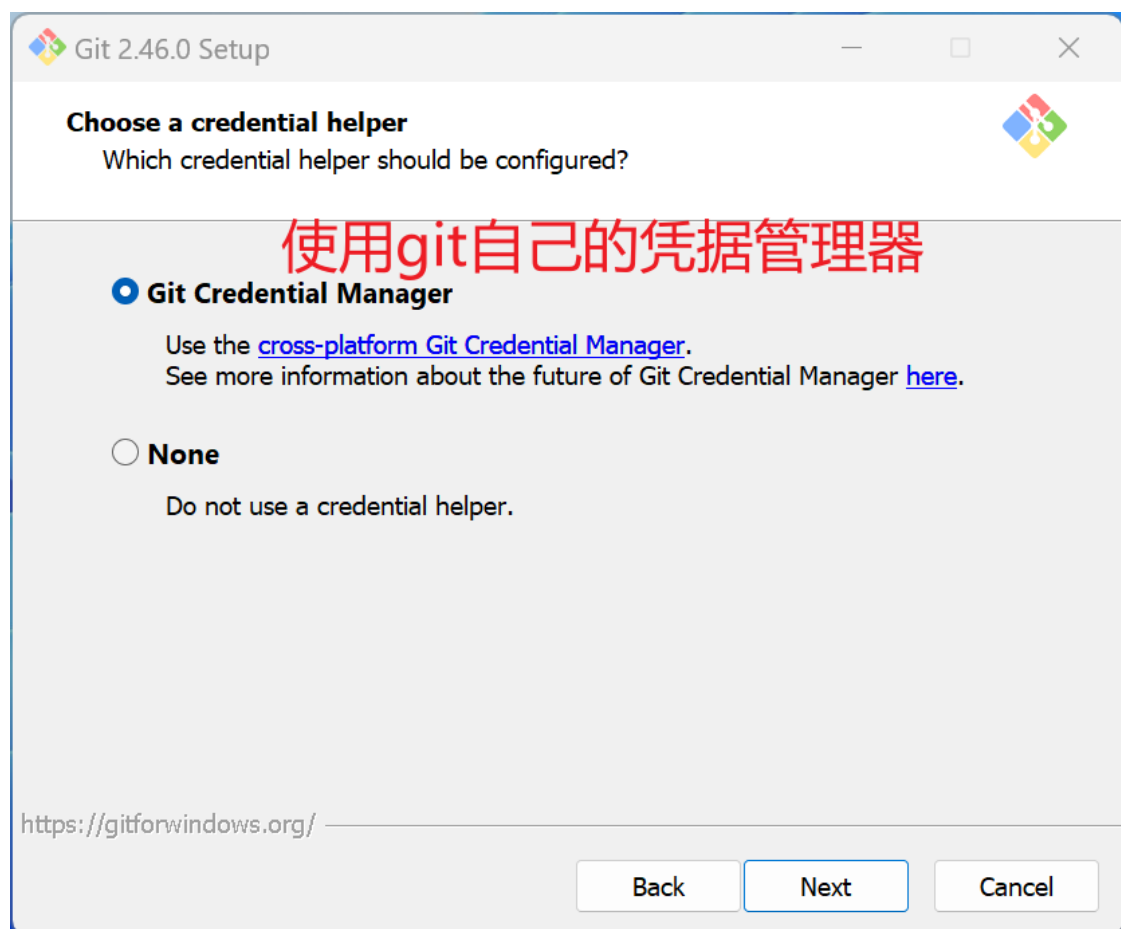
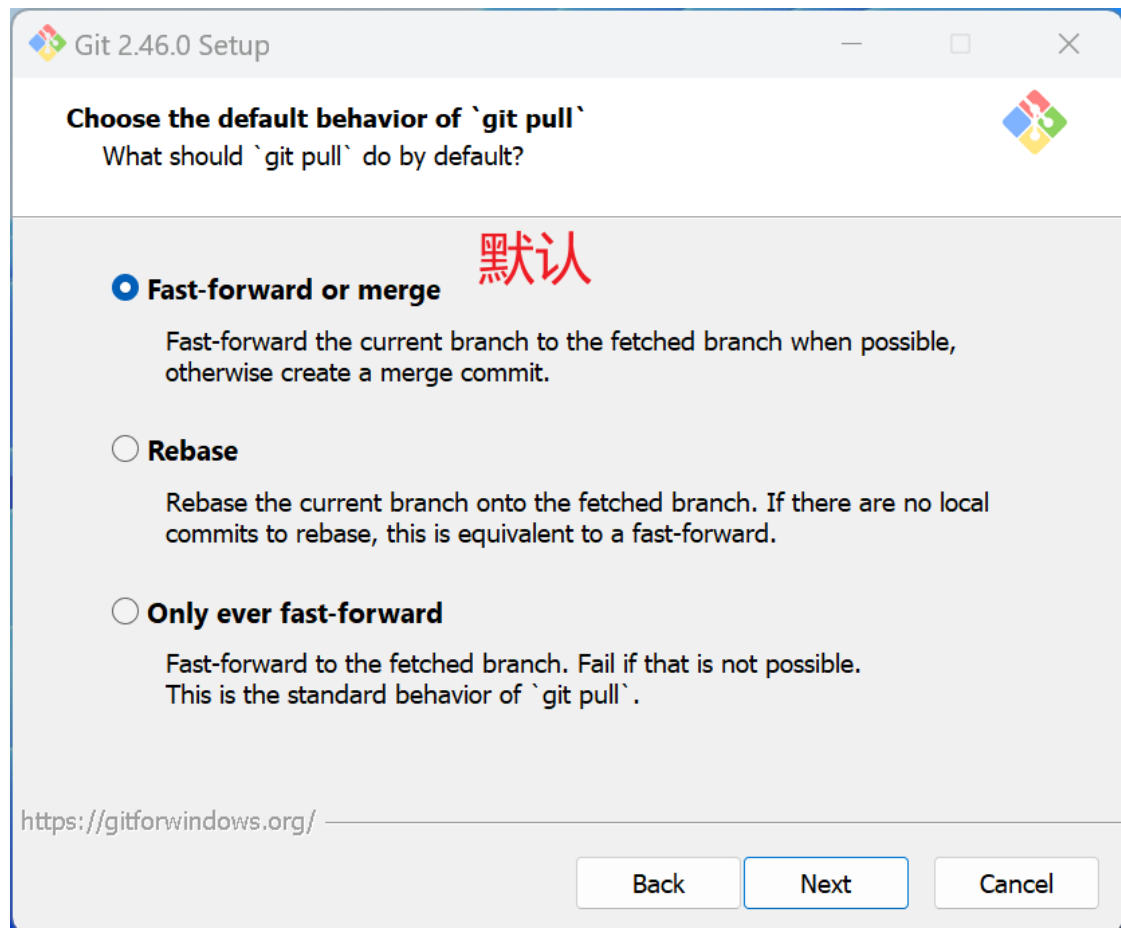


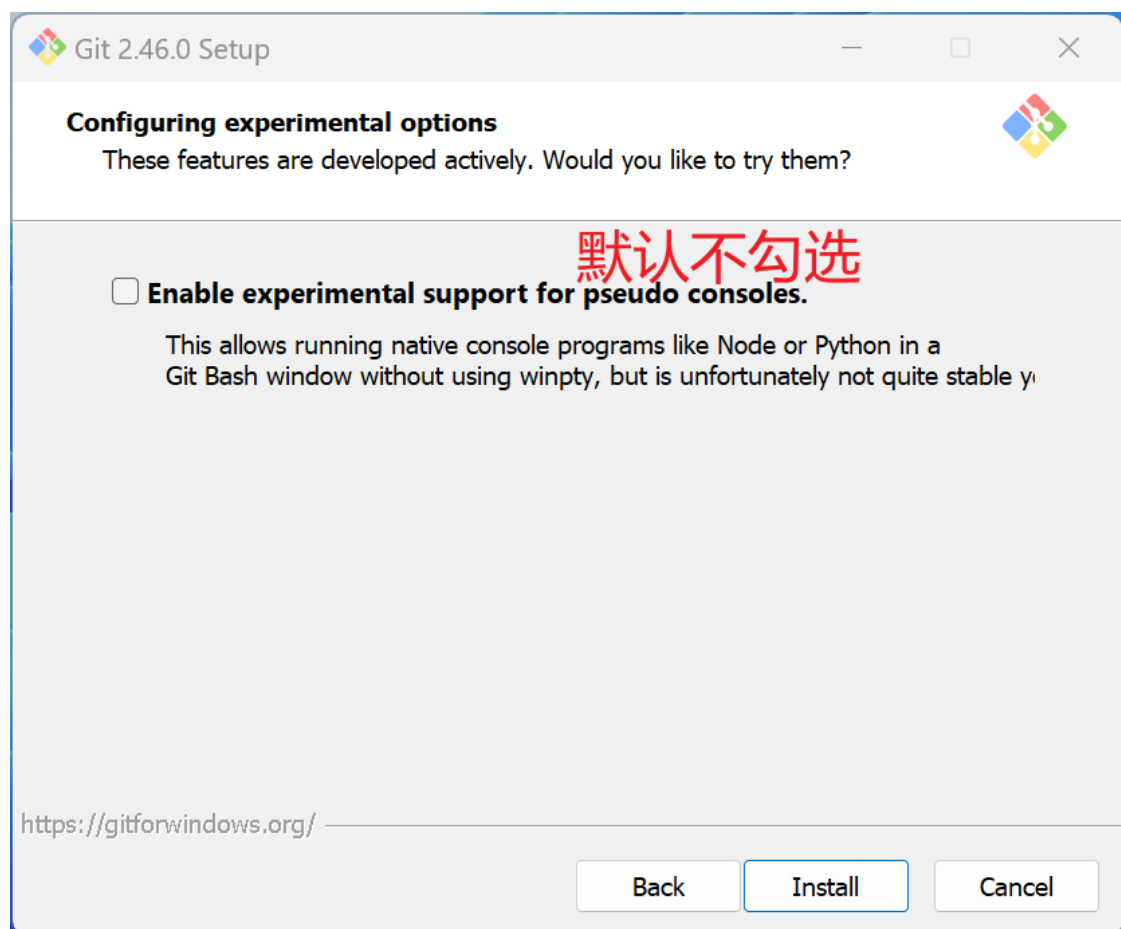
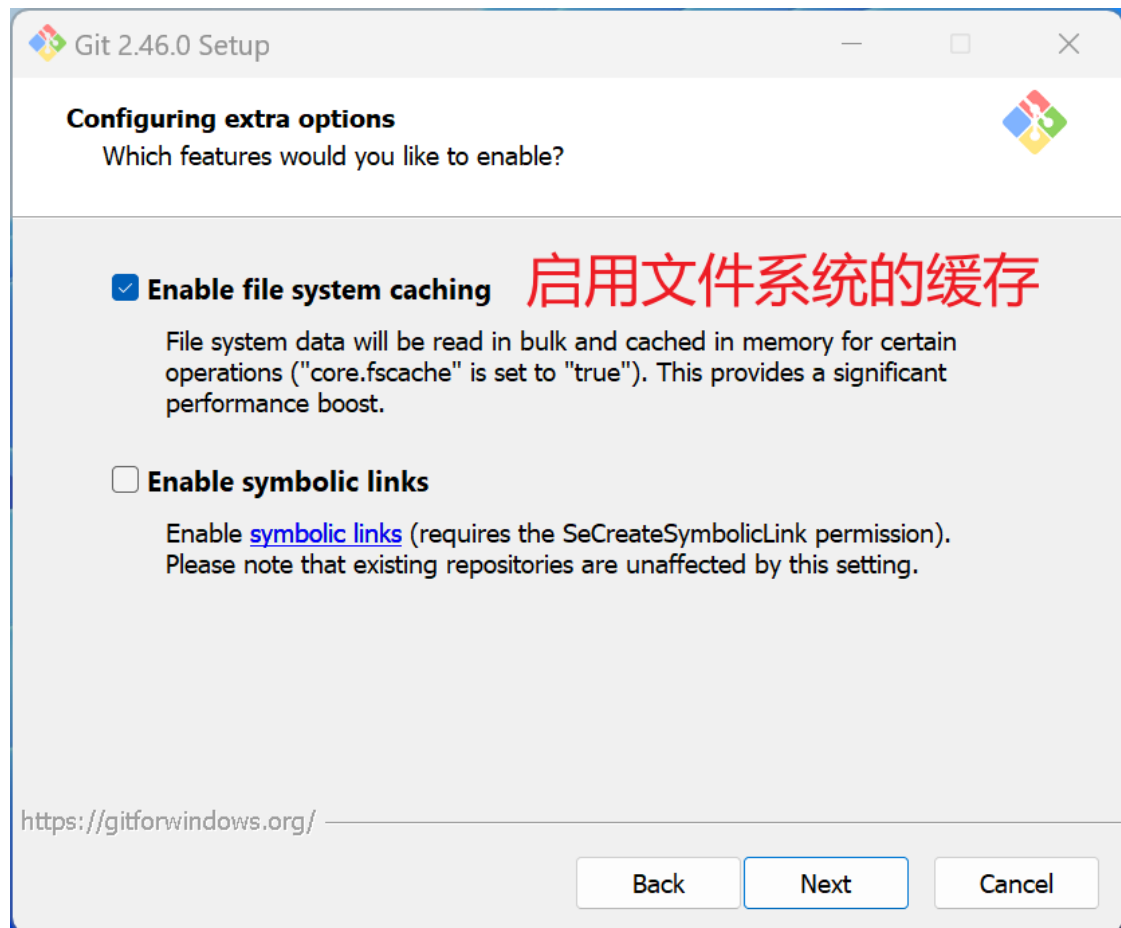


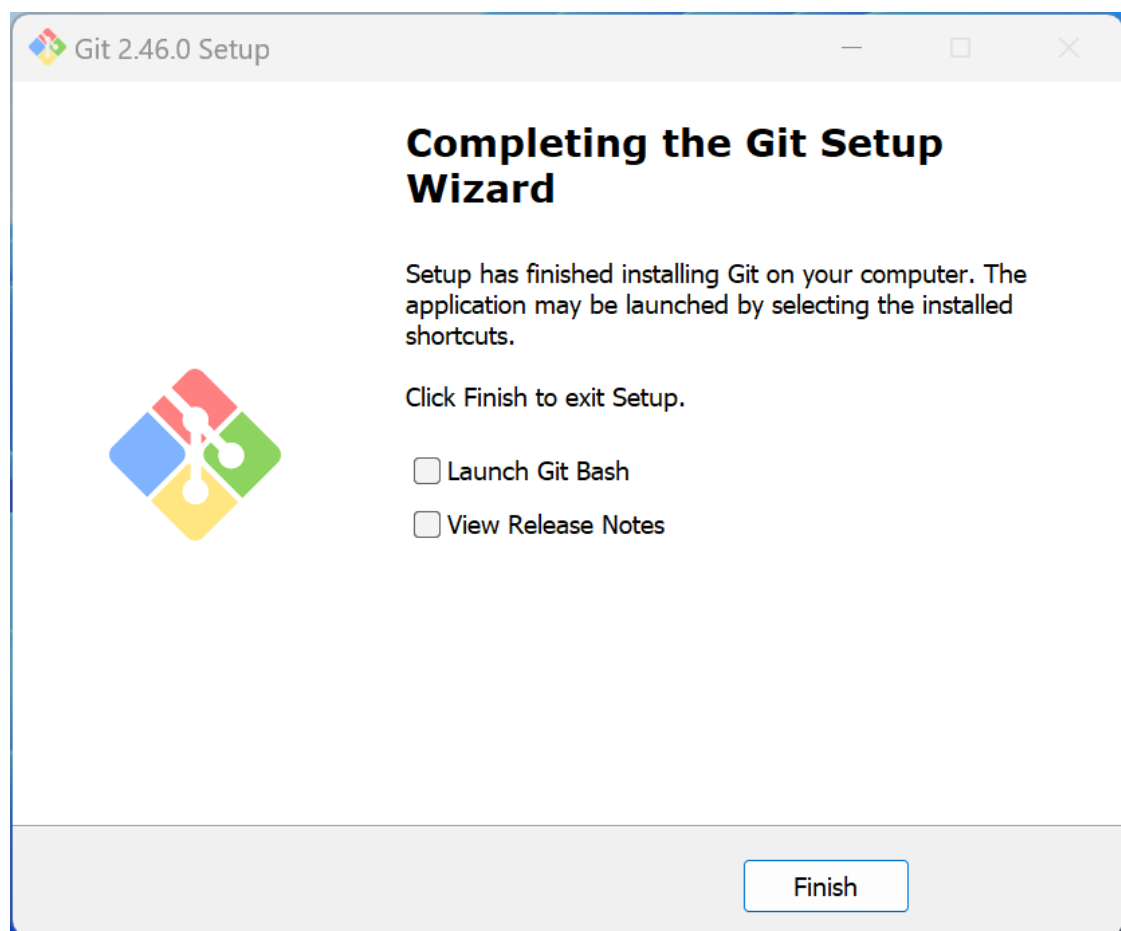
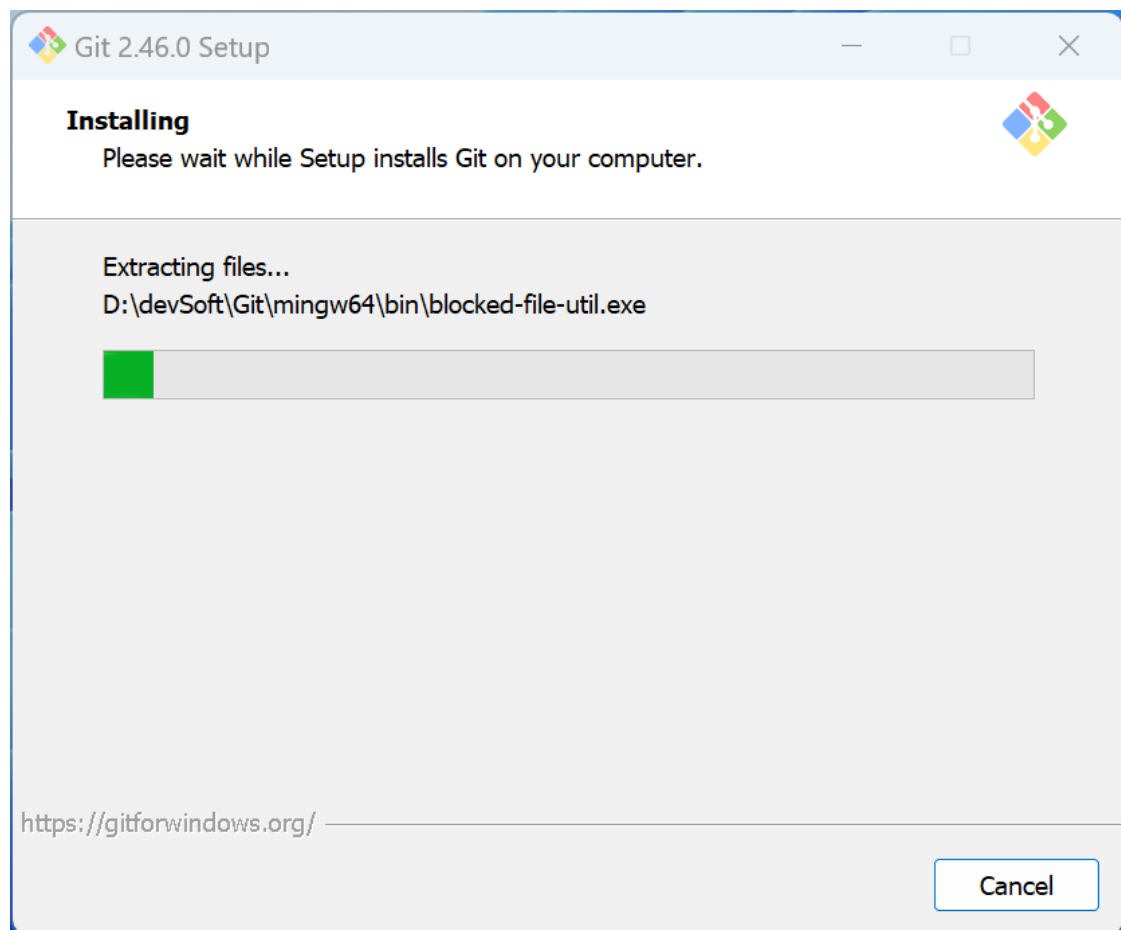




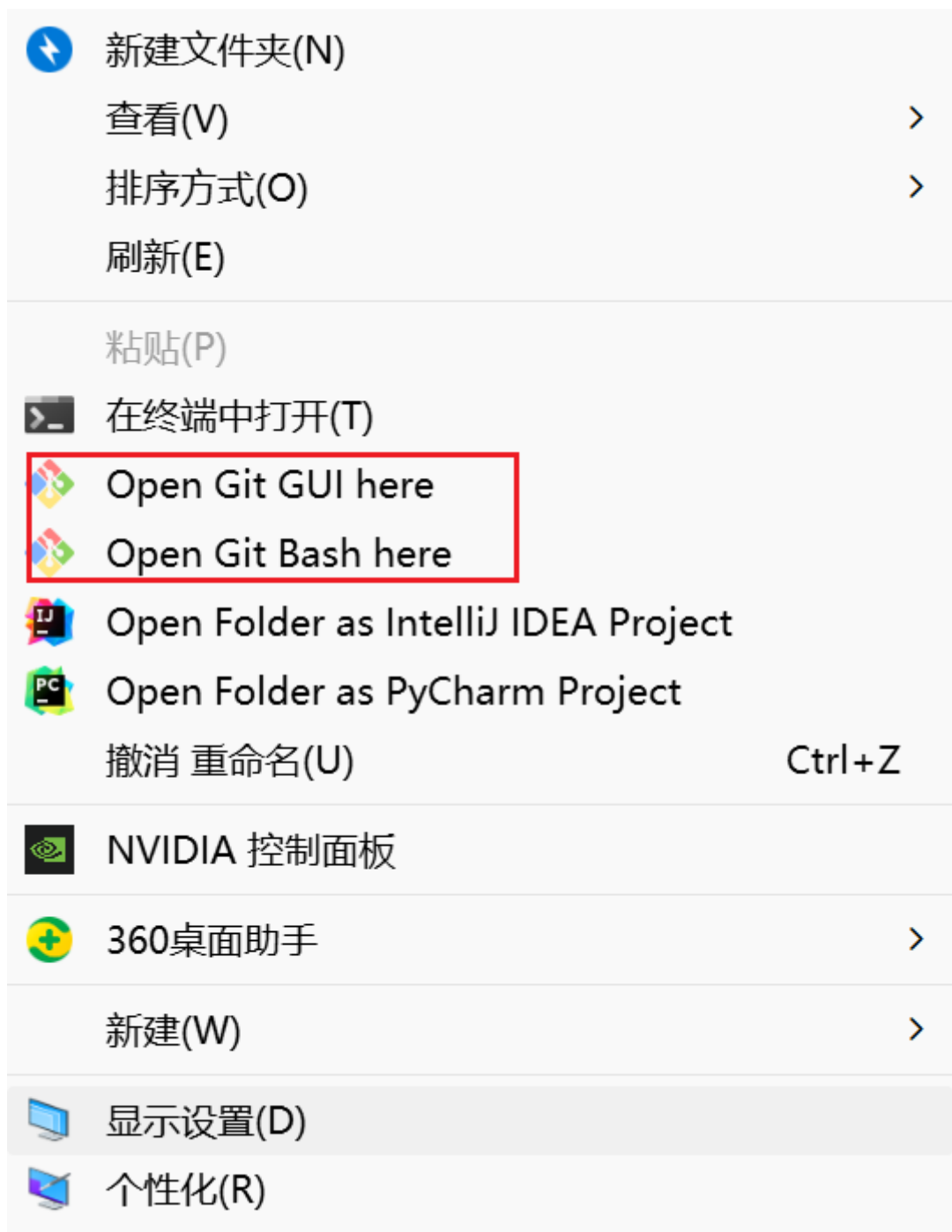








右键菜单:



三、Git中的工作分区

Git分成三个区：

- 工作目录：项目代码存放的目录
- 暂存区（index/stage）：是临时存储代码的区域，是一个虚拟的区域，仅仅是一个文件（index文件）
- 代码/版本库：是一个虚拟的区域，管理我们代码的版本（head指针指向最新的版本）

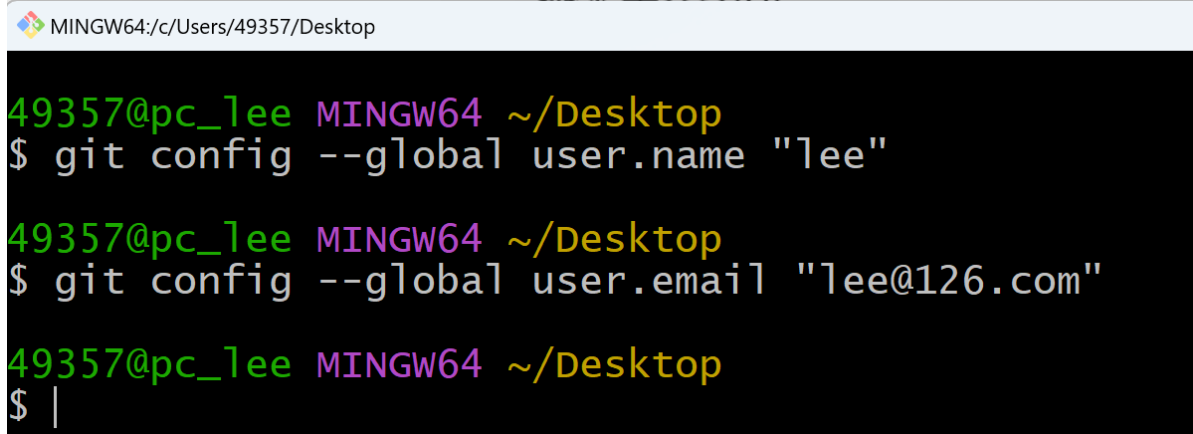
四、Git的使用

查看命名帮助信息：

```
$ git 命令 --help
```

1. 配置git的用户信息

```
git config --global user.name "用户名"  
git config --global user.email "邮箱地址"
```

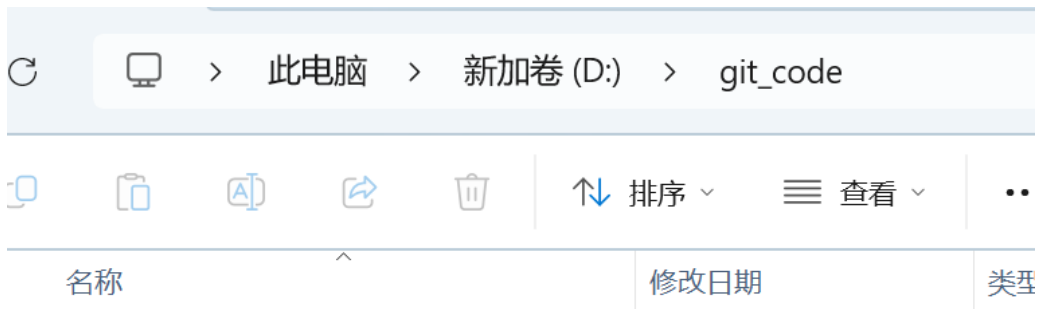


```
MINGW64:/c:/Users/49357/Desktop  
49357@pc_lee MINGW64 ~/Desktop  
$ git config --global user.name "lee"  
49357@pc_lee MINGW64 ~/Desktop  
$ git config --global user.email "lee@126.com"  
49357@pc_lee MINGW64 ~/Desktop  
$ |
```

git的bash中是可以直接使用linux命令的。

2. 创建一个目录，存放git管理的项目

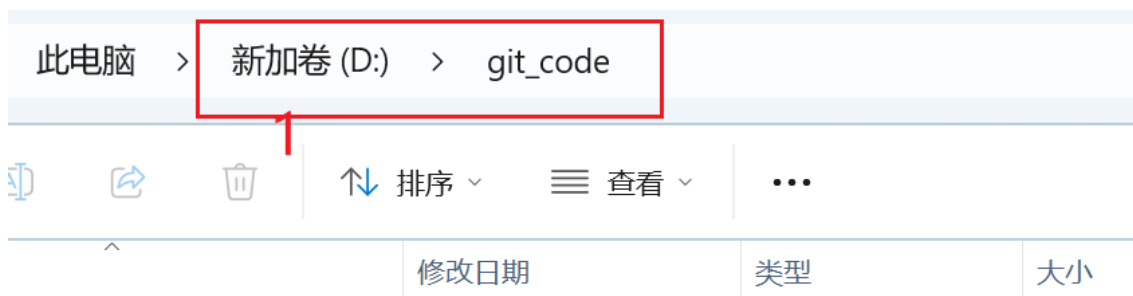
在D盘上创建一个新的文件夹：git_code



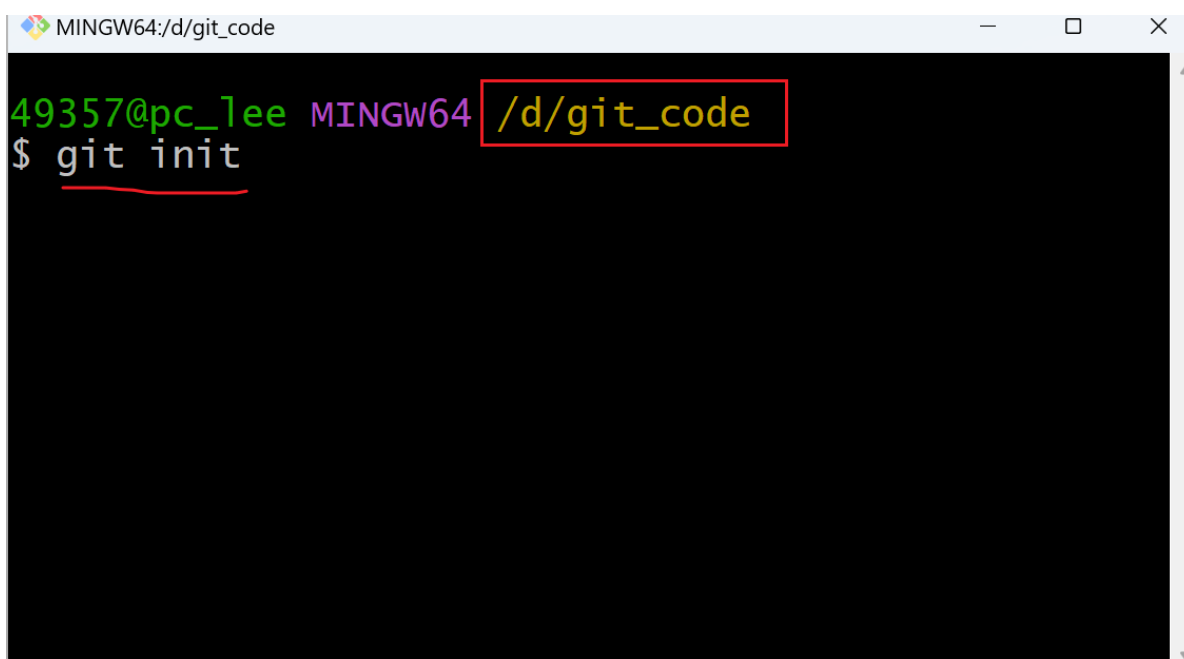
此时文件夹是空的

此时，git_code和git没有任何关系，所以git也不会管理它，所以我们需要让git_code和git产生关系。

通过 git init 命令把git_code目录变成 Git 可以管理的仓库(需要先进入git_code):

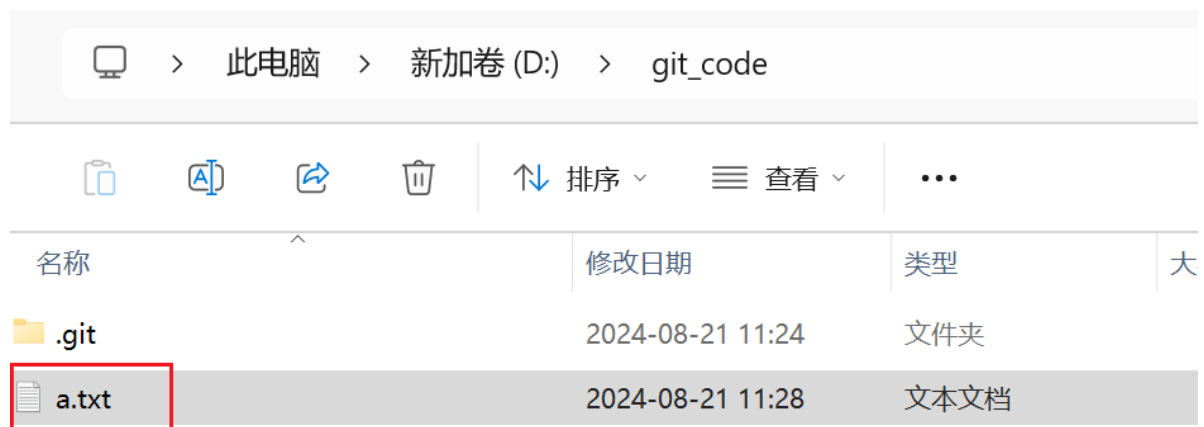


2





3. 在git_code下创建一个文件



此时的a.txt仅仅存储在工作目录下，我们需要将a.txt提交到代码库，才会产生版本，方便版本管理。

注意：工作目录中的内容必须先添加到“暂存区”，然后才能提交到“代码库”

4. 添加工作目录中的内容到暂存区

git add . 表示工作目录中所有的文件到暂存区

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  a.txt

nothing added to commit but untracked files present (use "git add" to track)
```

查看当前git的状态

表示当前文件没有添加到暂存区

添加到暂存区的命令

没有添加到暂存区的内容，默认就是红色

```
$ git add a.txt
49357@pc_lee MINGW64 /d/git_code (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   a.txt

49357@pc_lee MINGW64 /d/git_code (master)
```

添加a.txt到暂存区

再次查看状态

表示有未提交的内容

使用该命令从暂存区移除文件

暂存区未提交的内容默认是绿色

```
$ git rm --cached a.txt
rm 'a.txt'
49357@pc_lee MINGW64 /d/git_code (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  a.txt

nothing added to commit but untracked files present (use "git add" to track)
```

从暂存区移除文件

5. 将暂存区的内容提交到代码库

git commit . -m "提交信息" 提交暂存区中所有的内容到代码库

```

49357@pc_lee MINGW64 /d/git_code (master)
$ git commit a.txt -m "初始化项目"
[master (root-commit) d653fe2] 1 file changed, 1 insertion(+)
create mode 100644 a.txt

49357@pc_lee MINGW64 /d/git_code (master)

49357@pc_lee MINGW64 /d/git_code (master)
$ git reflog
d653fe2 (HEAD -> master) HEAD@{0}: commit (initial): 初始化项目

49357@pc_lee MINGW64 /d/git_code (master)
$ git log
commit d653fe20726346afa39d3442bb3d7582afcade8f8 (HEAD -> master)
Author: lee <lee@126.com>
Date:   Wed Aug 21 11:40:53 2024 +0800

```

提交暂存区的内容到代码块

`-m (message)` 提交时添加的备注信息

版本号简写

查看简单的日志信息

查看完整的日志信息

完整的版本号

初始化项目

提交内容的时候，没有指定 `-m` 参数，就会启动默认的编辑器。

i / o 插入模式

esc 命令模式

:wq 保存并退出

6. 撤销工作目录中增加的内容

撤销工作目录中增加的内容的前提是：没有添加到暂存区

(use "git restore ..." to discard changes in working directory)

```

$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a.txt

no changes added to commit (use "git add" and/or "git commit -a")

49357@pc_lee MINGW64 /d/git_code (master)
$ git restore a.txt

```

撤销工作目录中的修改内容

如果已经添加到暂存区，此时要撤销修改，需要先将内容从暂存区移出来，然后再撤销修改

```

$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   a.txt

```

将内容从暂存区移除

7. 版本回退

当内容提交到代码库，就永久生效了，会产生一个新的版本号。

我们可以使用版本回退的方式回退到想要的版本上。

```
# HEAD^ 表示回退上一个版本
# HEAD^^ 表示回退上上一个版本
git reset --hard HEAD^
# 可以直接使用版本号回退到想要的版本上
git reset --hard commit_id
```

要重返未来，用 `git reflog` 查看命令历史，以便确定要回到未来的哪个版本。

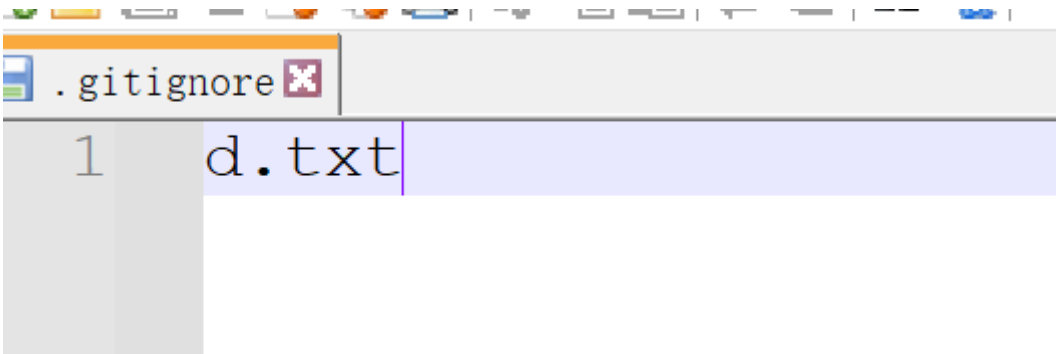
五、.gitignore忽略文件

在工作目录中如果有的内容是我们不需要添加到git代码库中的，此时可以使用 `.gitignore` 文件进行配置。

`.gitignore` 的作用就是用来配置需要 git 忽略的内容

名称	修改日期	类型	大小
.git	2025-01-14 14:49	文件夹	
.gitignore	2025-01-14 14:47	Git Ignore 源文件	1 KB
a.txt	2025-01-14 14:35	文本文档	1 KB
b.txt	2025-01-14 14:35	文本文档	0 KB
c.txt	2025-01-14 14:47	文本文档	0 KB
d.txt	2025-01-14 14:47	文本文档	0 KB

在 `.gitignore` 中添加需要 git 忽略的内容：



六、分支

提交代码，拉取代码都是以分支作为单位进行操作的。


```
$ git branch -v # 查看当前的分支
* master 243d87c 第四次提交，将hello改成了welcome （*所在的位置就是当前的分支）

$ git branch 分支名称 # 创建分支

$ git checkout test # 切换分支
Switched to branch 'test'

49357@pc_lee MINGW64 /d/git_code (master)
$ git merge test # 表示将test分支合并到当前所在的分支(master)
# 如果想将test分支合并到master分支， 必须先将当前分支切换成master
$ git add a.txt
$ git commit -m "内容" # 提交
```

如果多个分支同时修改了相同位置的代码，此时合并就会出现冲突，此时需要手动解决冲突。

合并分支后commit提交内容的时候，git commit 后面不能添加文件名，而是直接提交整个分支

七、远程代码库

github gitee gitlab

1. 查看远程库信息

```
git remote -v #查看远程库信息
```

2. 添加远程库

```
git remote add 别名 远程库地址
```

3. 删除远程库

```
git remote rm 别名
```

4. 将本地代码推送到远程库

注意：代码推送是以分支为单位的，也就是多分支的时候需要指定推送的分支

```
$ git push 远程库的别名 本地分支名
```

5. 拉取远程库中的代码

```
$ git pull 远程库的别名 远程分支名
远程分支名要求是本地分支关联的远程跟踪分支
```

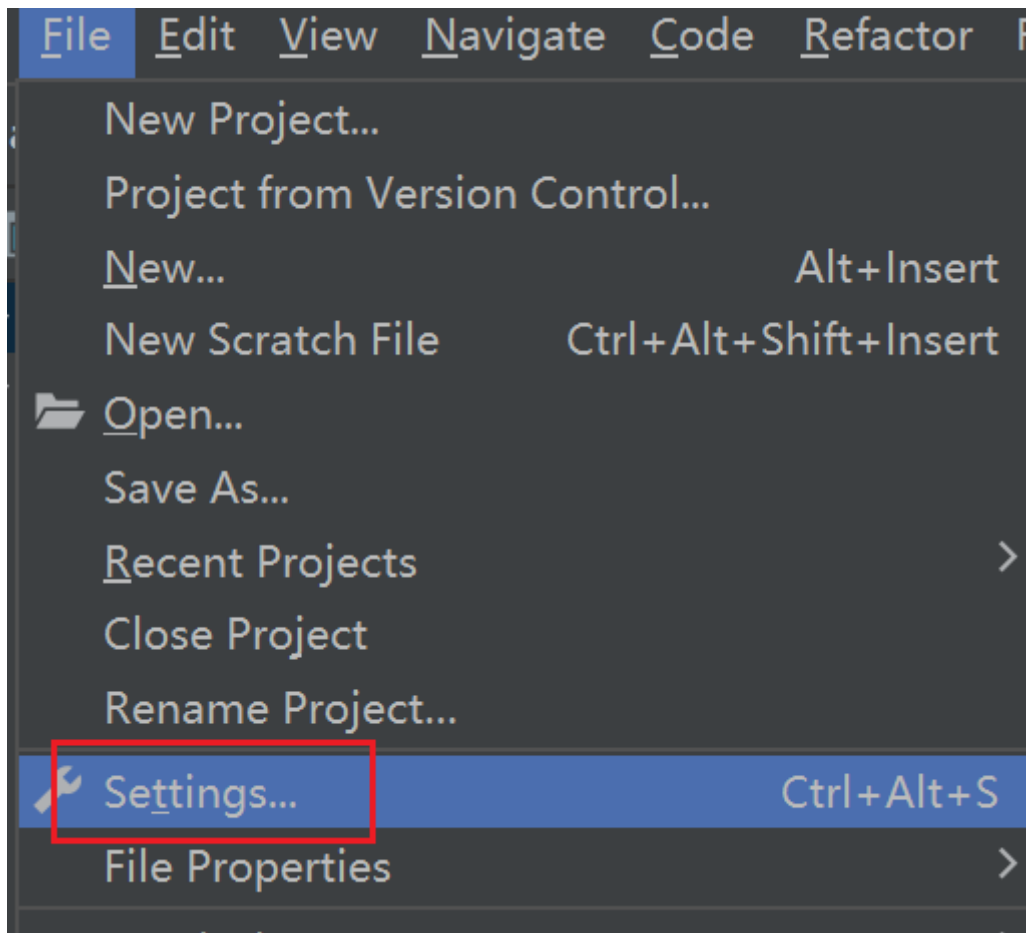
注意：在推送代码之前先拉取代码

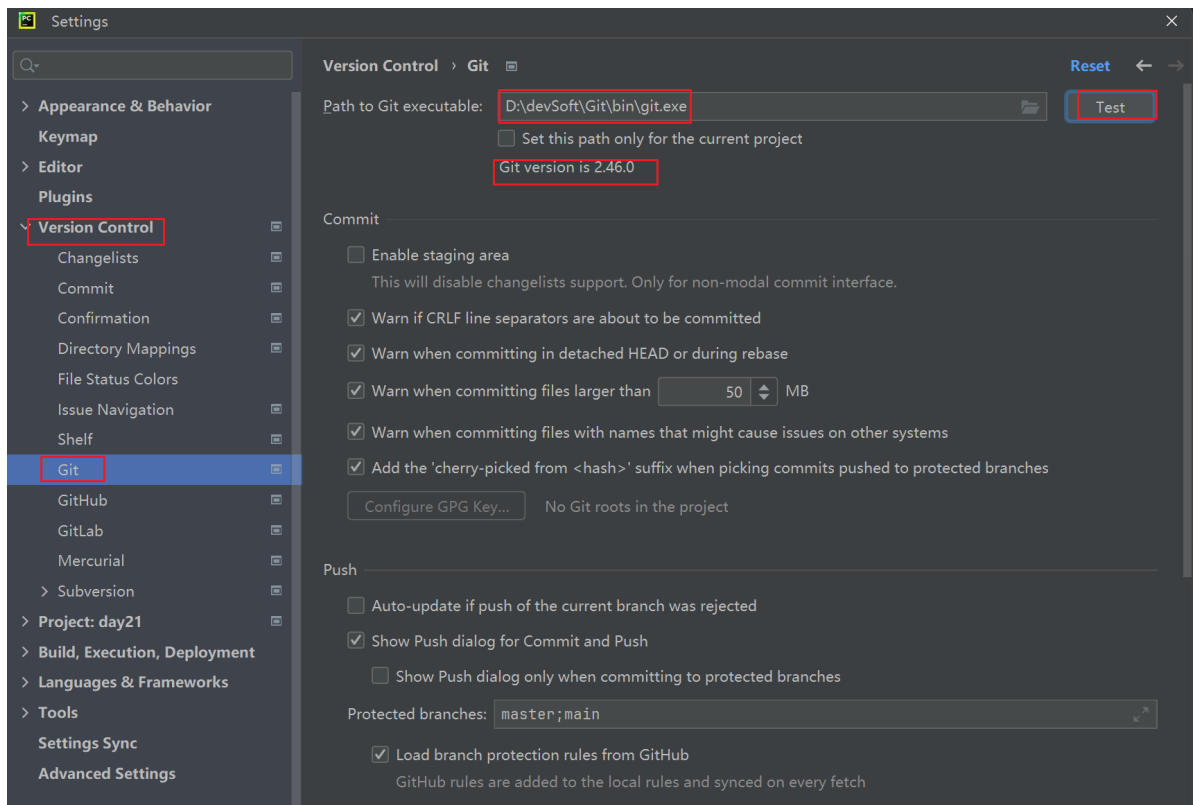
6. 克隆项目

```
git clone https://gitee.com/hehziwei/student-system.git
```

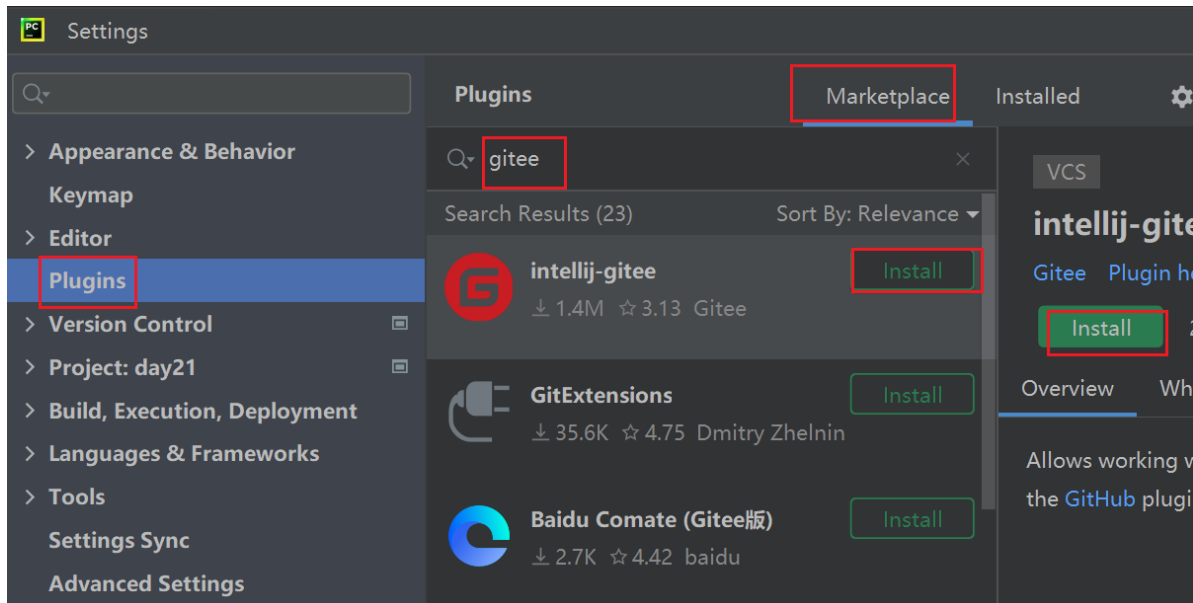
八、PyCharm中使用git

1. PyCharm中集成git

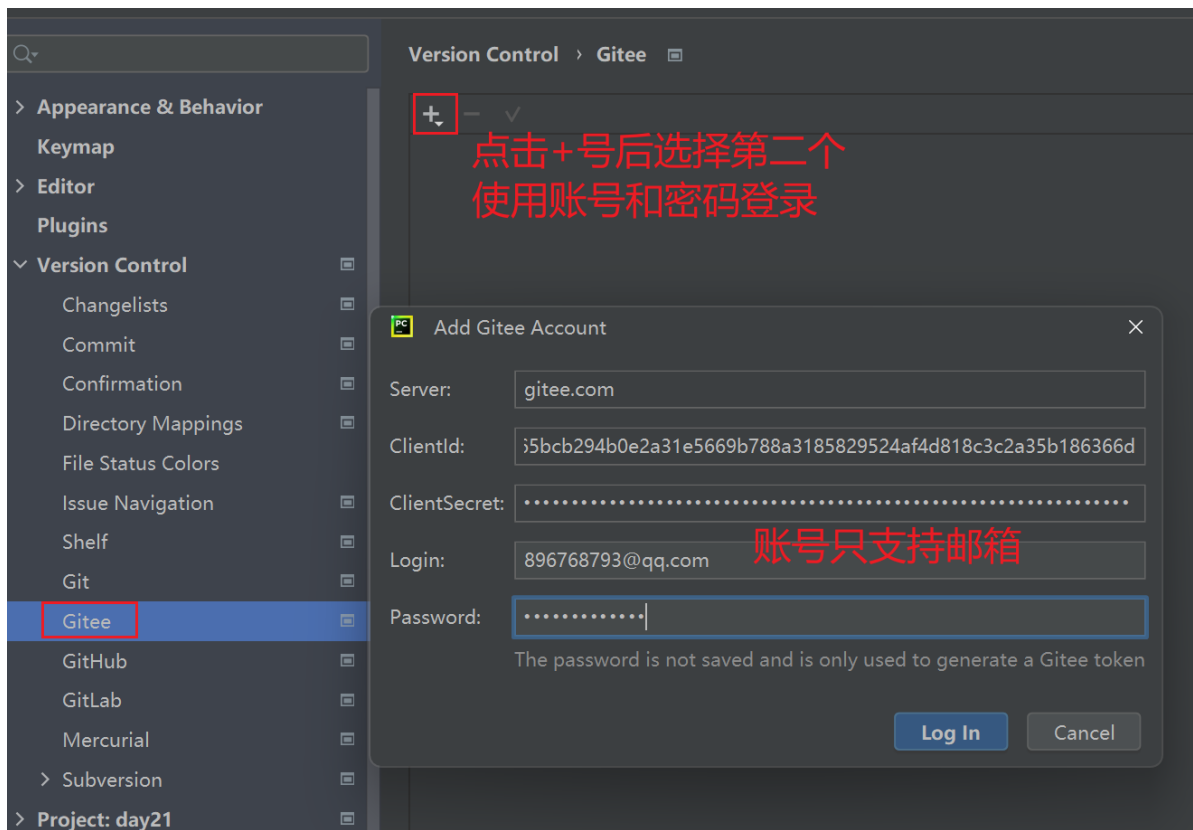




2. PyCharm中安装gitee插件



3. 添加gitee账号信息





4. 使用token进行登录


先将刚才添加的账户信息删除




安全设置

 SSH公钥

 GPG公钥 Beta

 私人令牌

 登录历史

+生成新令牌

私人令牌

使用私人令牌，可以通过 [Gitee Open API](#) 访问你授权的数据，请注意数据安全！

私人令牌描述

test 自己去一个名字

请选择将要生成的私人令牌所拥有的权限

- | | |
|---|-------------------------|
| <input checked="" type="checkbox"/> 全选 | |
| <input checked="" type="checkbox"/> user_info | 访问你的个人信息、最新动态等 |
| <input checked="" type="checkbox"/> projects | 查看、创建、更新你的项目 |
| <input checked="" type="checkbox"/> pull_requests | 查看、发布、更新你的 Pull Request |
| <input checked="" type="checkbox"/> issues | 查看、发布、更新你的 Issue |
| <input checked="" type="checkbox"/> notes | 查看、发布、管理你在项目、代码片段中的评论 |
| <input checked="" type="checkbox"/> keys | 查看、部署、删除你的公钥 |
| <input checked="" type="checkbox"/> hook | 查看、部署、更新你的 Webhook |
| <input checked="" type="checkbox"/> groups | 查看、管理你的组织以及成员 |
| <input checked="" type="checkbox"/> gists | 查看、删除、更新你的代码片段 |
| <input checked="" type="checkbox"/> enterprises | 查看、管理你的企业以及成员 |
| <input checked="" type="checkbox"/> emails | 查看你的个人邮箱信息 |

提交 取消

私人令牌生成提示



你的私人令牌test已生成
此界面只有一次
将token复制下来

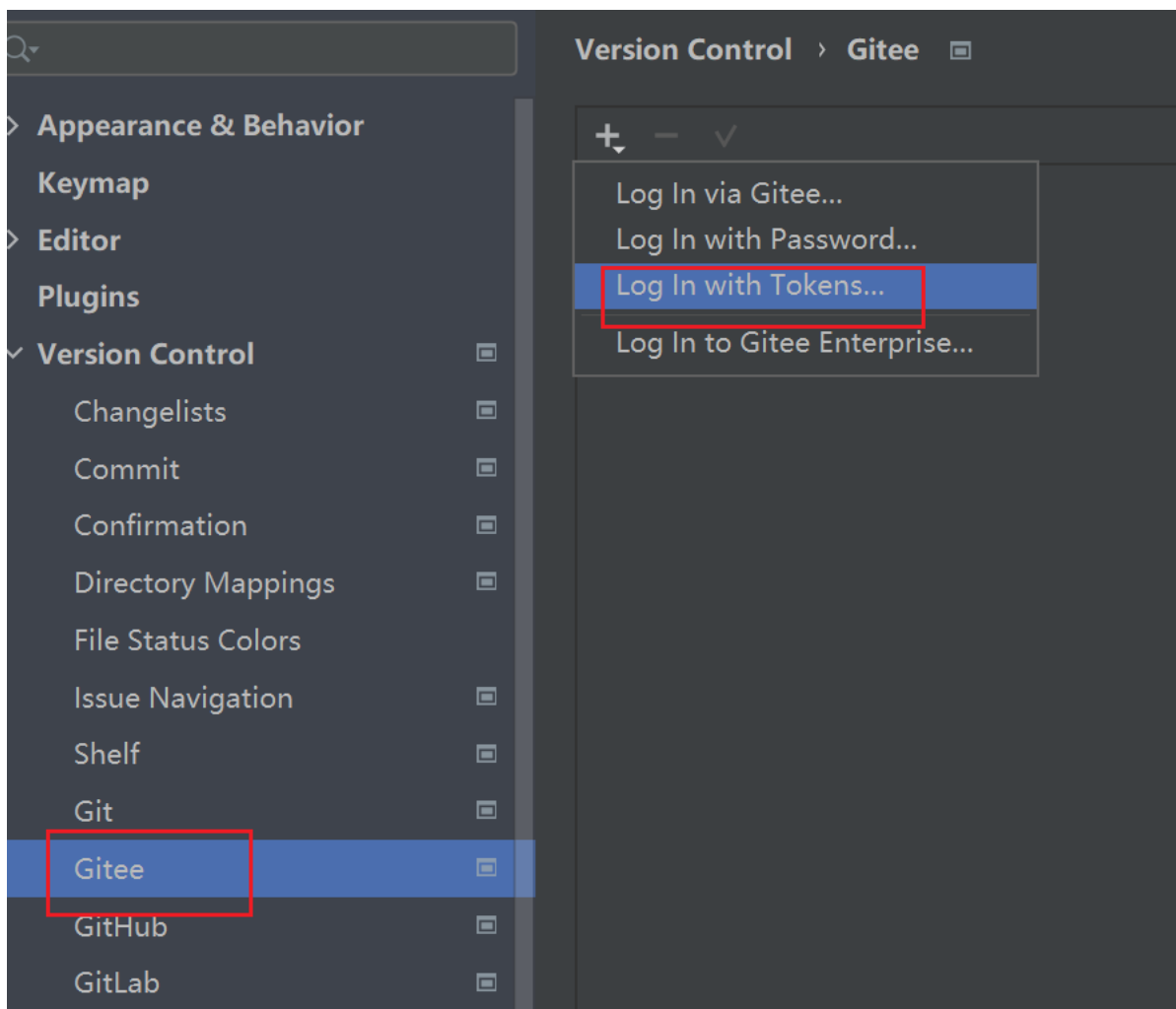
599f03c2a3e88a3195c131ff359b7b2a

复制

本页面关闭后，平台将不再显示私人令牌，请妥善保存。

☐ 我已经了解私人令牌不再明文显示在平台上，并且已经复制保存好该令牌。

确认并关闭



PC Add Gitee Account

Server:

Access Token:

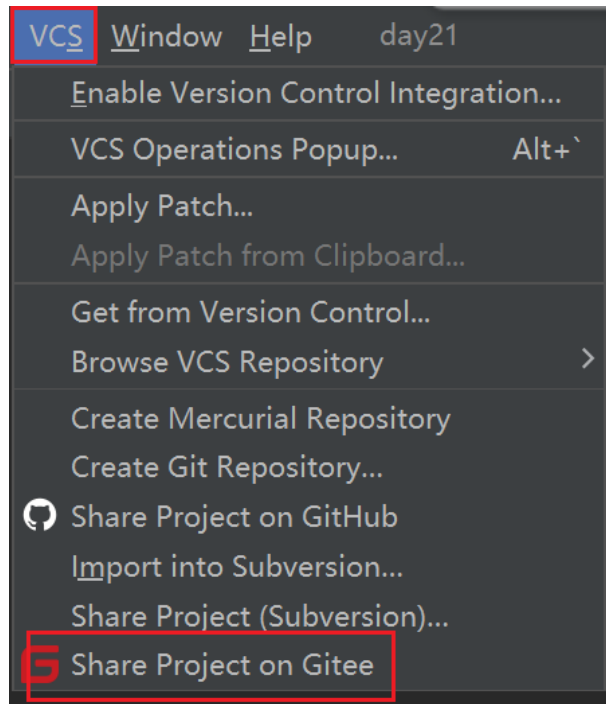
The following scopes must be granted to the access token: user_info projects pull_requests gists issues notes groups

Refresh Token:

☐ is private token

Add Account Cancel

5. 将项目分享到gitee上



PC Share Project On Gitee

Repository name: ☐ Private

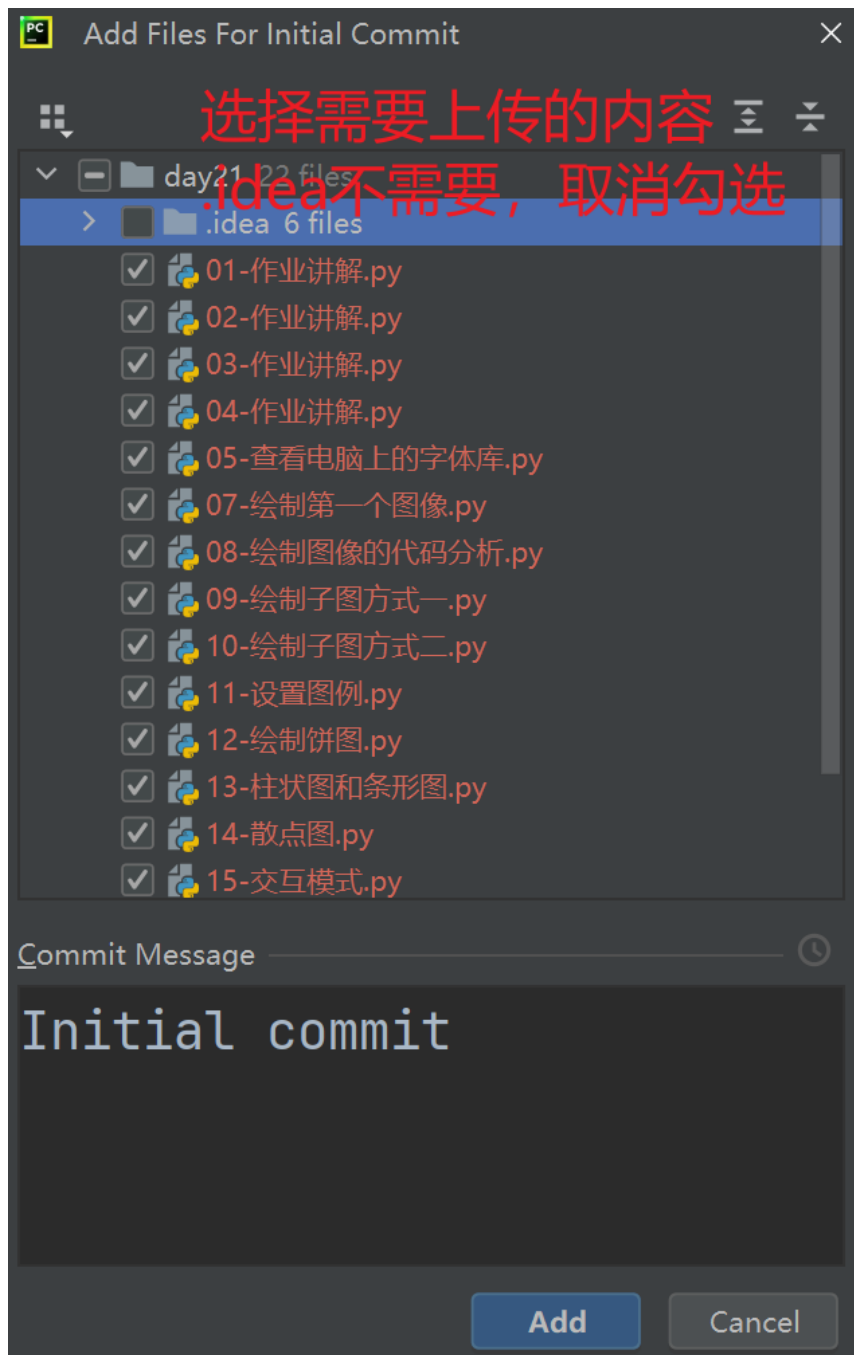
Remote:

Description:

Description:

? Share Cancel

仓库名字
远程库的别名，默认是origin
可以自行修改，我将其修改成
项目名字



6. 在PyCharm中使用git本地管理代码

