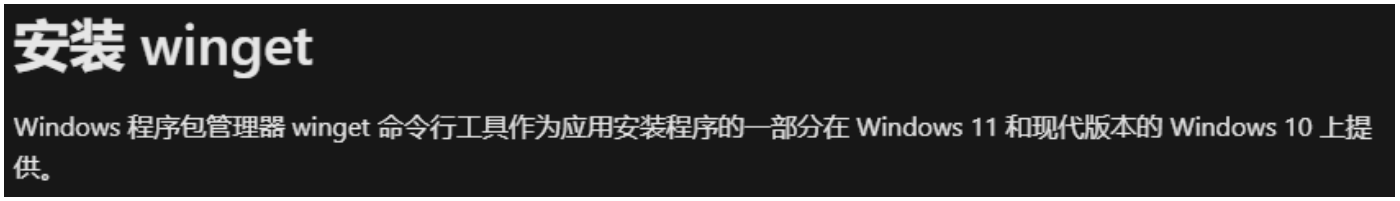
WinGet

在开始之前，先介绍一个win包管理工具，如同node的npm，java的maven，andriod的gradle。



1.换下载源

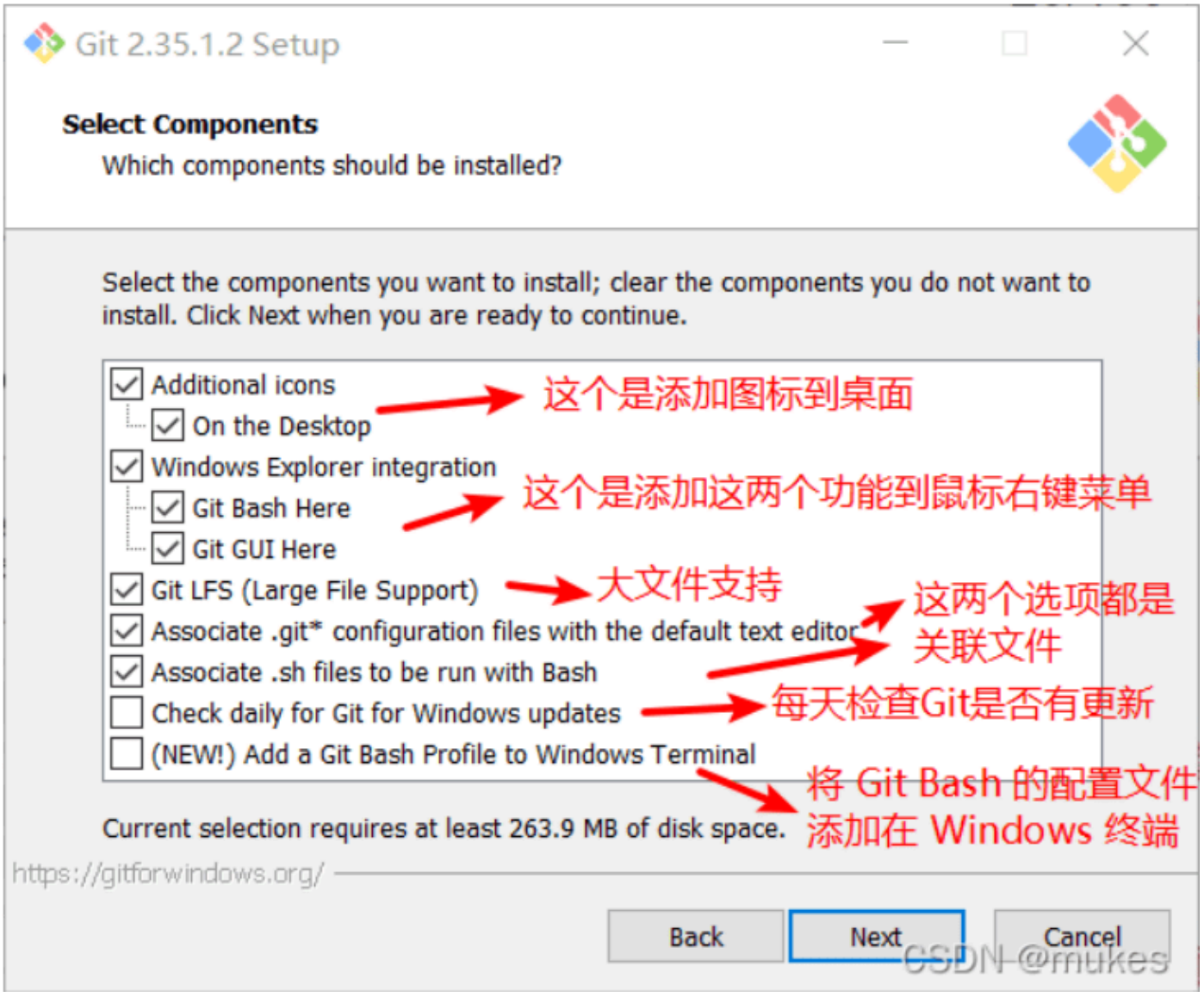
```
winget source list  # 查看下载源:
#二选一
winget source reset #删除现有所有源
winget source remove winget   #移除微软官方源
#
winget source add winget https://mirrors.ustc.edu.cn/winget-source #替换 USTC 镜像:


winget search git  #搜索git的安装包
winget show --id git.git  #显示以id为 git.git的包信息
winget install --id git.git  --rainbow # 安装指定包  --reinbow 安装时的进度条



#其他命令
winget source reset winget  #重置为官方地址:
winget --info # 显示工具的常规信息
```

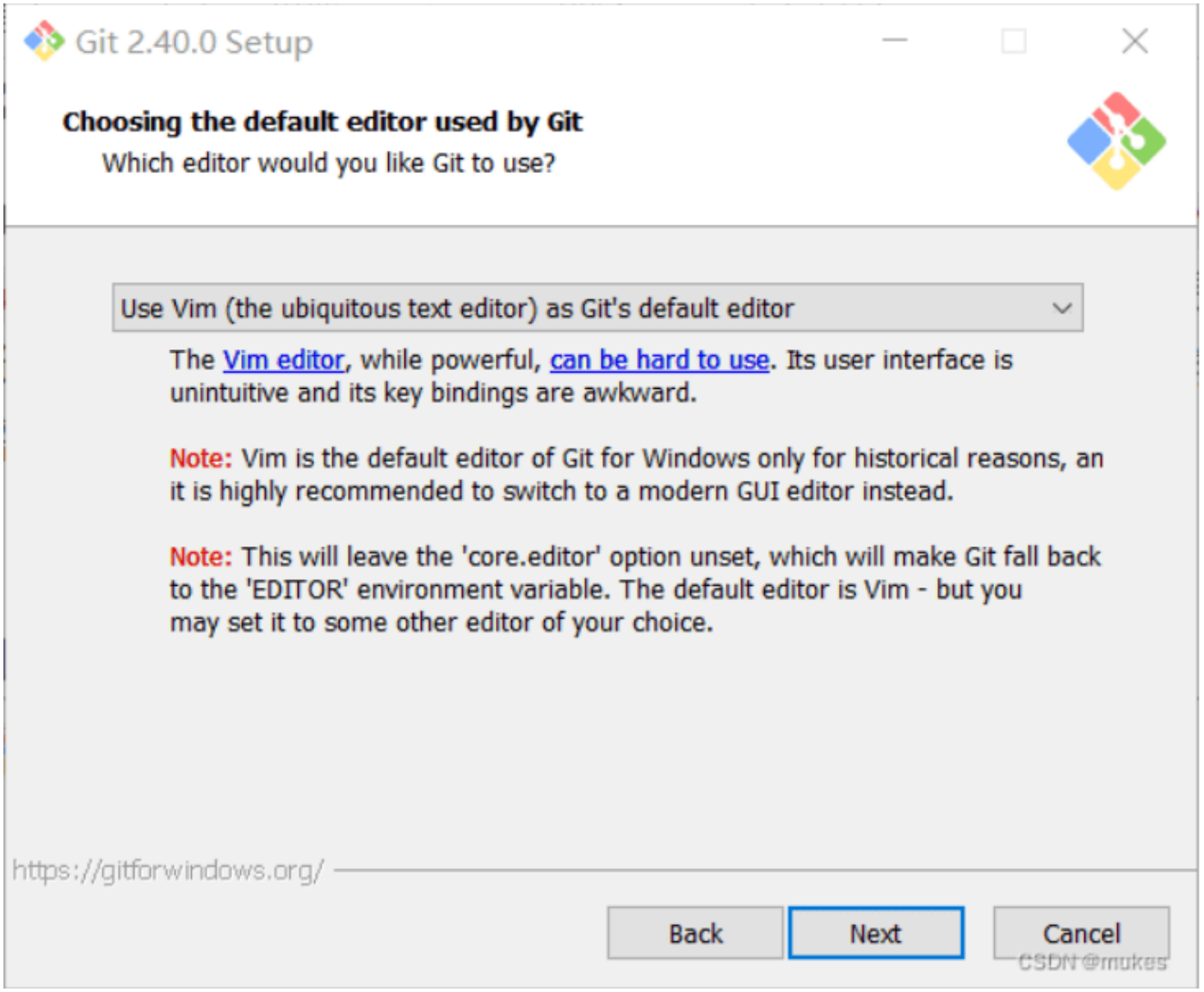
如果没有代理工具，下载git还是会很慢，所以清华源

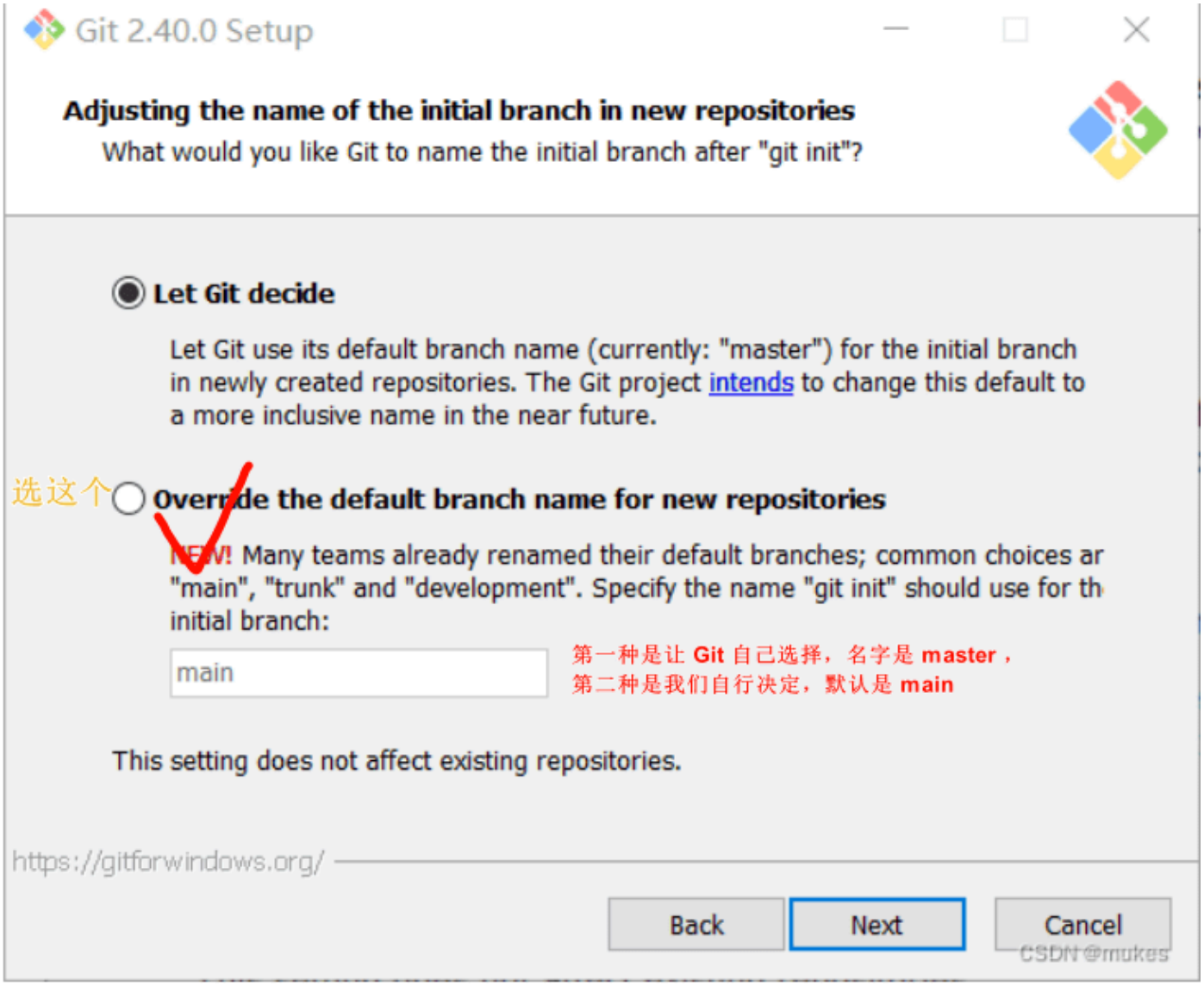
git-2.45.1 的下载地址

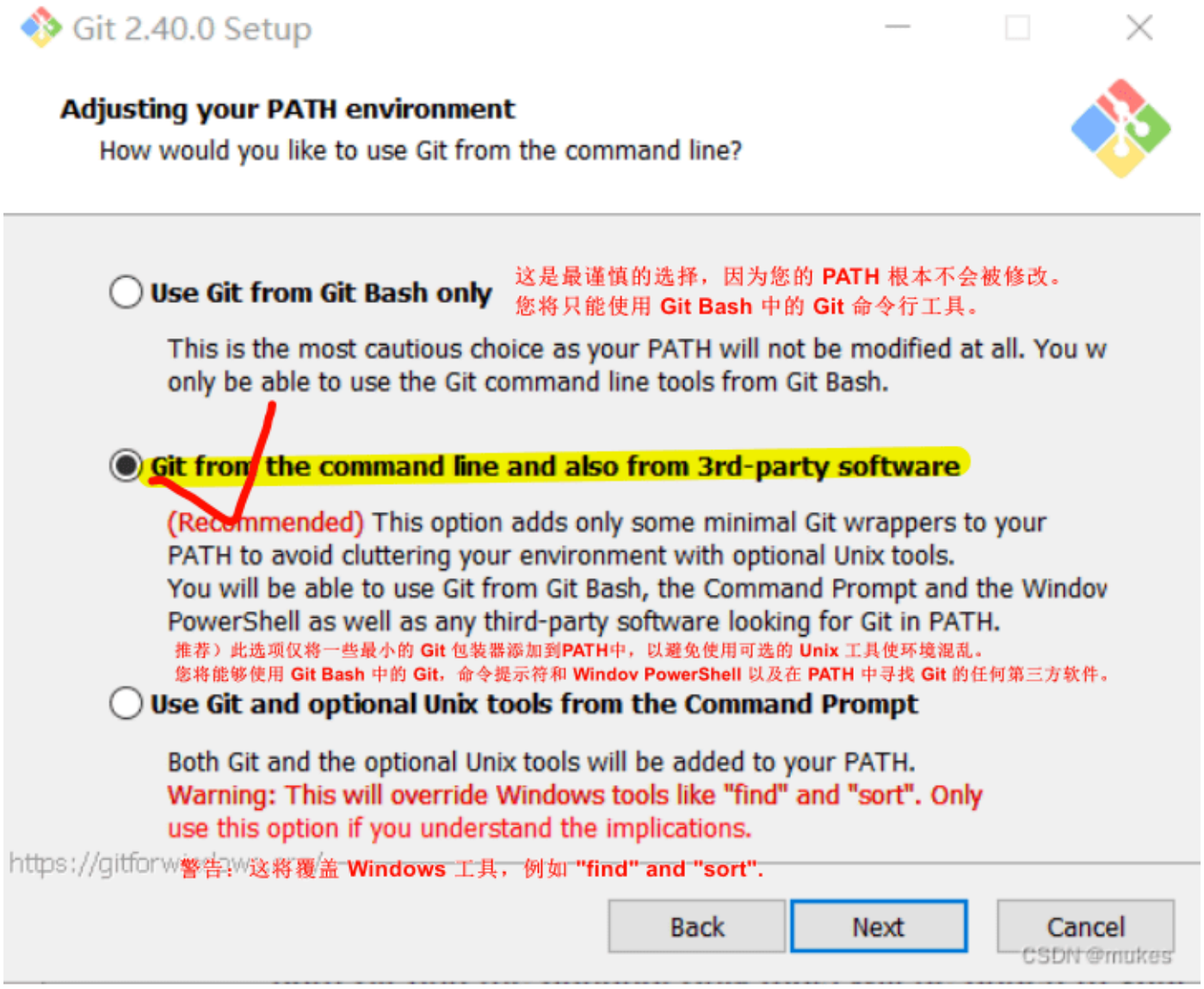
```
https://mirrors.tuna.tsinghua.edu.cn/github-release/git-for-
windows/git/Git%20for%20Windows%20v2.45.1/Git-2.45.1-64-bit.exe
```

git 安装 （提供关键信息选择）

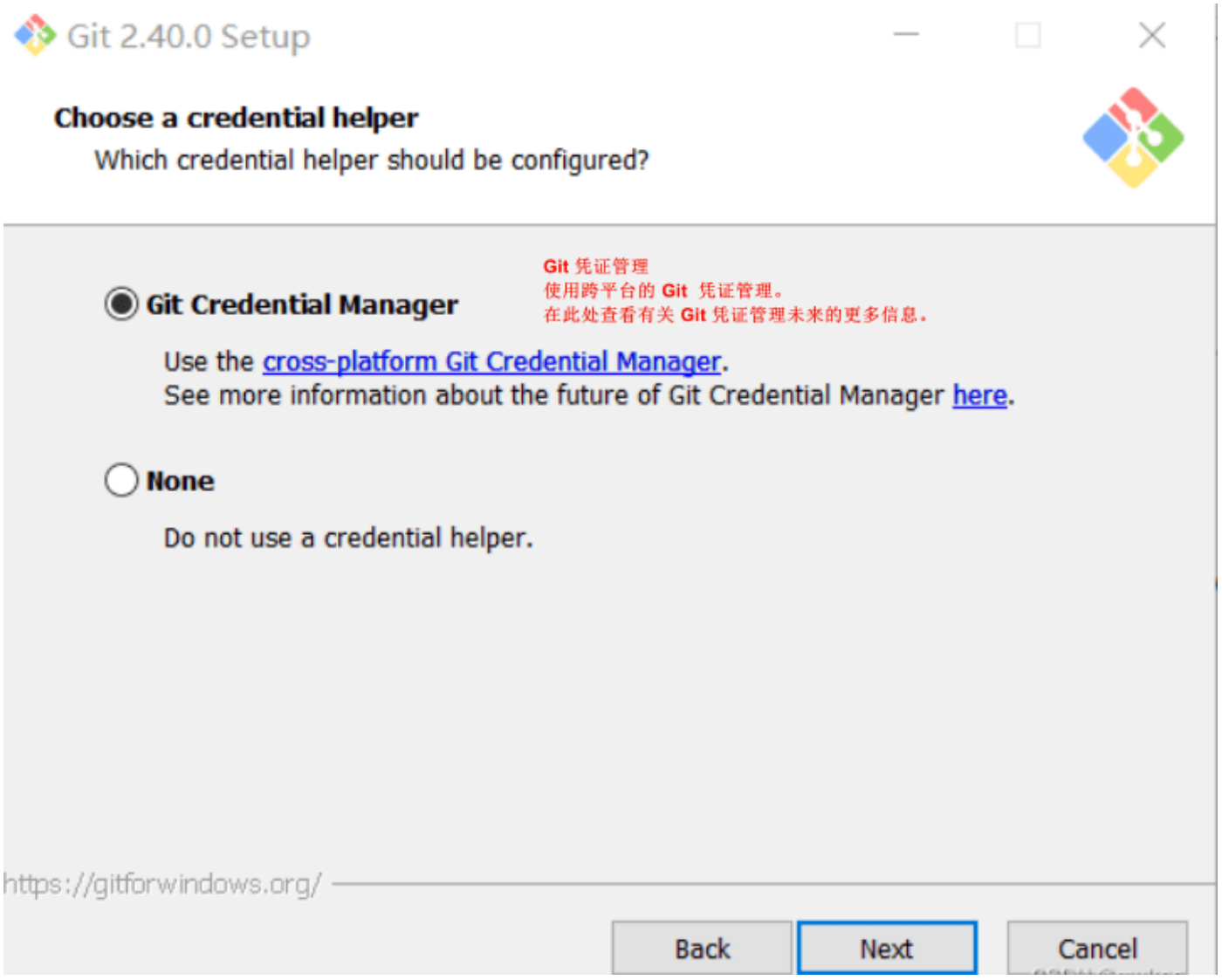

Git 2.35.1.2 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
这个是添加图标到桌面
Windows Explorer integration
Git Bash Here
Git GUI Here
这个是添加这两个功能到鼠标右键菜单
Git LFS (Large File Support)
大文件支持
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
这两个选项都是关联文件
Check daily for Git for Windows updates
每天检查Git是否有更新
(NEW!) Add a Git Bash Profile to Windows Terminal
将 Git Bash 的配置文件添加在 Windows 终端
Current selection requires at least 263.9 MB of disk space.
https://gitforwindows.org/
Back
Next
Cancel
CSDN @mukes


Git 2.40.0 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, an it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
Back
Next
Cancel
CSDN@mukes


Git 2.40.0 Setup
Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?
Let Git decide
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to change this default to a more inclusive name in the near future.
选这个
Override the default branch name for new repositories
NEW! Many teams already renamed their default branches; common choices ar "main", "trunk" and "development". Specify the name "git init" should use for th initial branch:
main
第一种是让 Git 自己选择，名字是 master，
第二种是我们自行决定，默认是 main
This setting does not affect existing repositories.
https://gitforwindows.org/
Back
Next
Cancel
CSDN @mukes


Git 2.40.0 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
这是最谨慎的选择，因为您的 PATH 根本不会被修改。
您将只能使用 Git Bash 中的 Git 命令行工具。
This is the most cautious choice as your PATH will not be modified at all. You w
only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your
PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov
PowerShell as well as any third-party software looking for Git in PATH.
推荐）此选项仅将一些最小的 Git 包装器添加到PATH中，以避免使用可选的 Unix 工具使环境混乱。
您将能够使用 Git Bash 中的 Git，命令提示符和 Windov PowerShell 以及在 PATH 中寻找 Git 的任何第三方软件。
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only
use this option if you understand the implications.
警告：这将覆盖 Windows 工具，例如 "find" and "sort".
Back
Next
Cancel
CSDN @mukes

之后一路点击Next


Git 2.40.0 Setup
Choose a credential helper
Which credential helper should be configured?
Git 凭证管理
使用跨平台的 Git 凭证管理。
在此处查看有关 Git 凭证管理未来的更多信息。
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Back
Next
Cancel


Windows 安全性
Git Credential Manager for Windows
Enter your credentials for https://github.com/.
用户名
密码
确定
取消

之后一路点击Next。

git初始化

用户信息

当安装完 Git 应该做的第一件事就是设置你的用户名称与邮件地址。 这样做很重要，因为每一个 Git 的提交都会使用这些信息，并且它会写入到你的每一次提交中，不可更改：

```
git config --global user.name "Junk Chen" #“Junk Chen”是你自己设置的名字。
git config --global user.email junkchen@vip.qq.com #junkchen@vip.qq.com是你的邮箱地址
```

git 交作业

1.常用命令

```
git config --list   # 检查配置信息
git init
git clone https://gitee.com/he-knows/task1.git
git status   #查看文件的
git branch -a  #查看所在目录的所有分支
echo xxx > readme.md   #把xxx写入到readme.md文件
git add READEME.md   #把文件添加到暂存区   .代表当前目录所有东西
git commit -m " init:  初始化" # 添加到本地仓库，并加上注释

git remote  #查看远程仓库


git push origin master  #把本地仓库推送到远程仓库origin 的master分支
```

Git 常用命令速查表

```
master :默认开发分支        Head  :默认开发分支
origin :默认远程版本库      Head^ :Head 的父提交
```

创建版本库

```
$ git clone <url>              #克隆远程版本库
$ git init                     #初始化本地版本库
```

修改和提交

```
$ git status                   #查看状态
$ git diff                     #查看变更内容
$ git add .                    #跟踪所有改动过的文件
$ git add <file>               #跟踪指定的文件
$ git mv <old> <new>           #文件改名
$ git rm <file>                #删除文件
$ git rm --cached <file>       #停止跟踪文件但不删除
$ git commit -m "commit message"
                               #提交所有更新过的文件
$ git commit --amend           #修改最后一次提交
```

查看提交历史

```
$ git log                      #查看提交历史
$ git log -p <file>            #查看指定文件的提交历史
$ git blame <file>             #以列表方式查看指定文件
                               的提交历史
```

撤消

```
$ git reset --hard HEAD        #撤消工作目录中所有未提交
                               文件的修改内容
$ git checkout HEAD <file>     #撤消指定的未提交文件的修
                               改内容
$ git revert <commit>          #撤消指定的提交
```

分支与标签

```
$ git branch                   #显示所有本地分支
$ git checkout <branch/tag>    #切换到指定分支或标签
$ git branch <new-branch>      #创建新分支
$ git branch -d <branch>       #删除本地分支
$ git tag                      #列出所有本地标签
$ git tag <tagname>            #基于最新提交创建标签
$ git tag -d <tagname>         #删除标签
```

合并与衍合

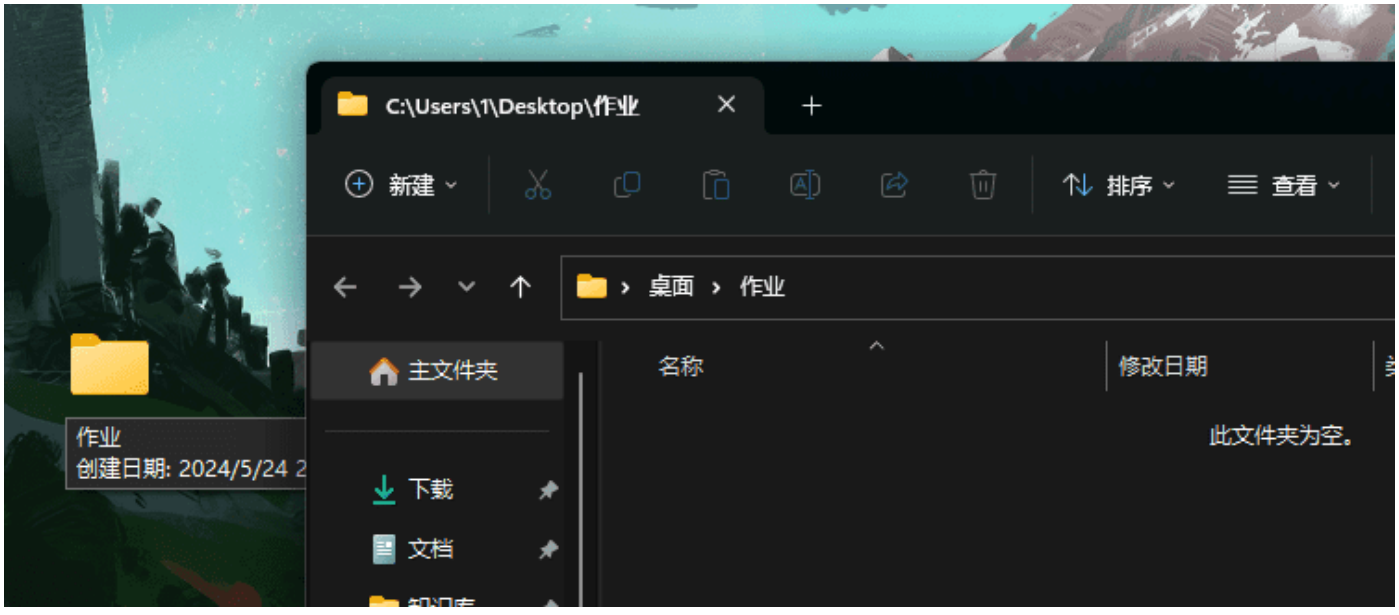
```
$ git merge <branch>           #合并指定分支到当前分支
$ git rebase <branch>          #衍合指定分支到当前分支
```

远程操作

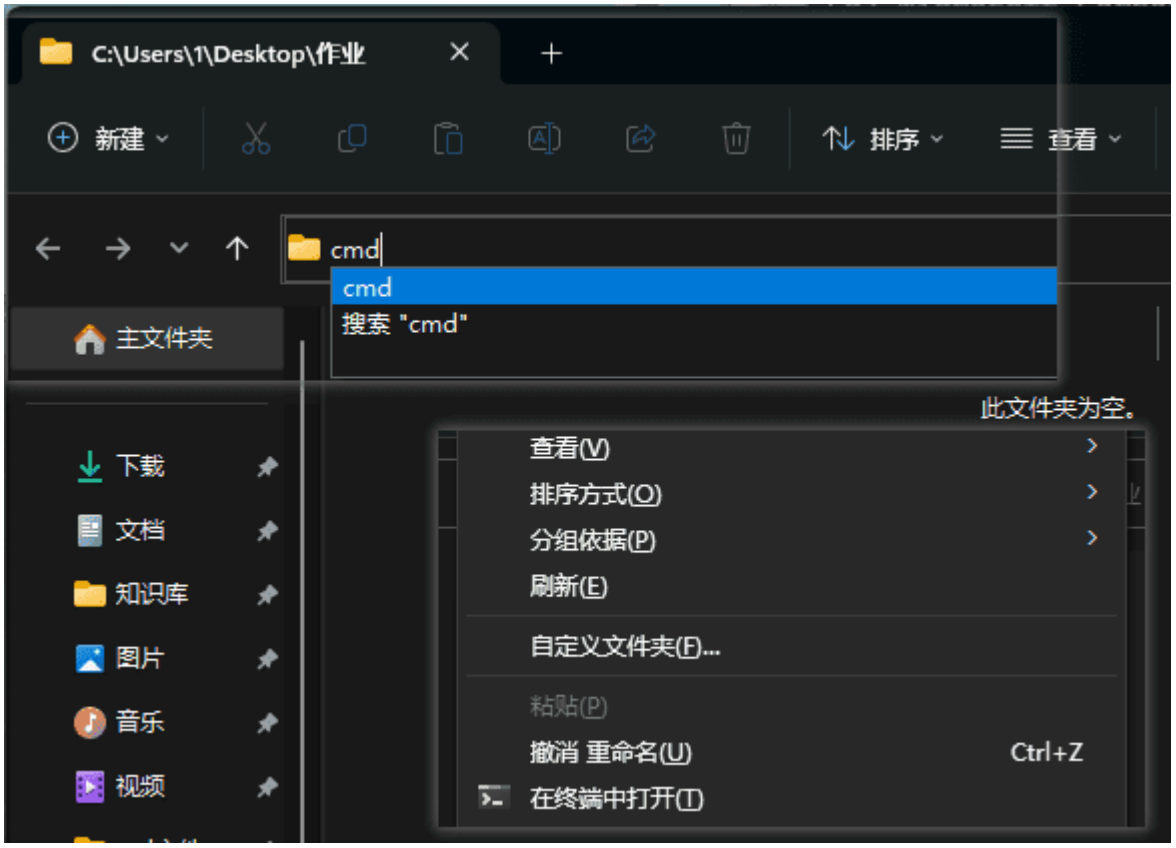
```
$ git remote -v                #查看远程版本库信息
$ git remote show <remote>     #查看指定远程版本库信息
$ git remote add <remote> <url>
                               #添加远程版本库
$ git fetch <remote>           #从远程库获取代码
$ git pull <remote> <branch>   #下载代码及快速合并
$ git push <remote> <branch>   #上传代码及快速合并
$ git push <remote> :<branch/tag-name>
                               #删除远程分支或标签
$ git push --tags              #上传所有标签
```

```
# Git Cheat Sheet <CN> (Version 0.1)     # 2012/10/26  -- by @riku  < riku@gitcafe.com / http://riku.wowubuntu.com >
```


2.在桌面新建文件夹

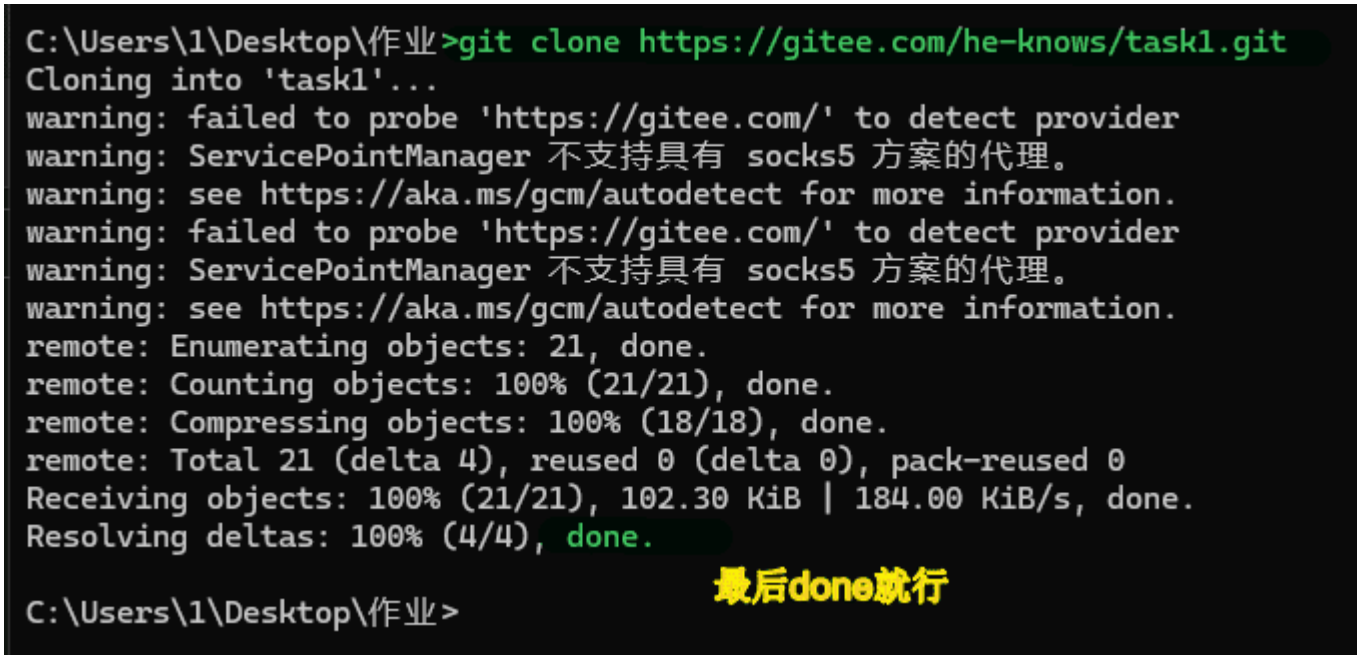


右键，选择>>在终端中打开，或者在地址栏输入cmd



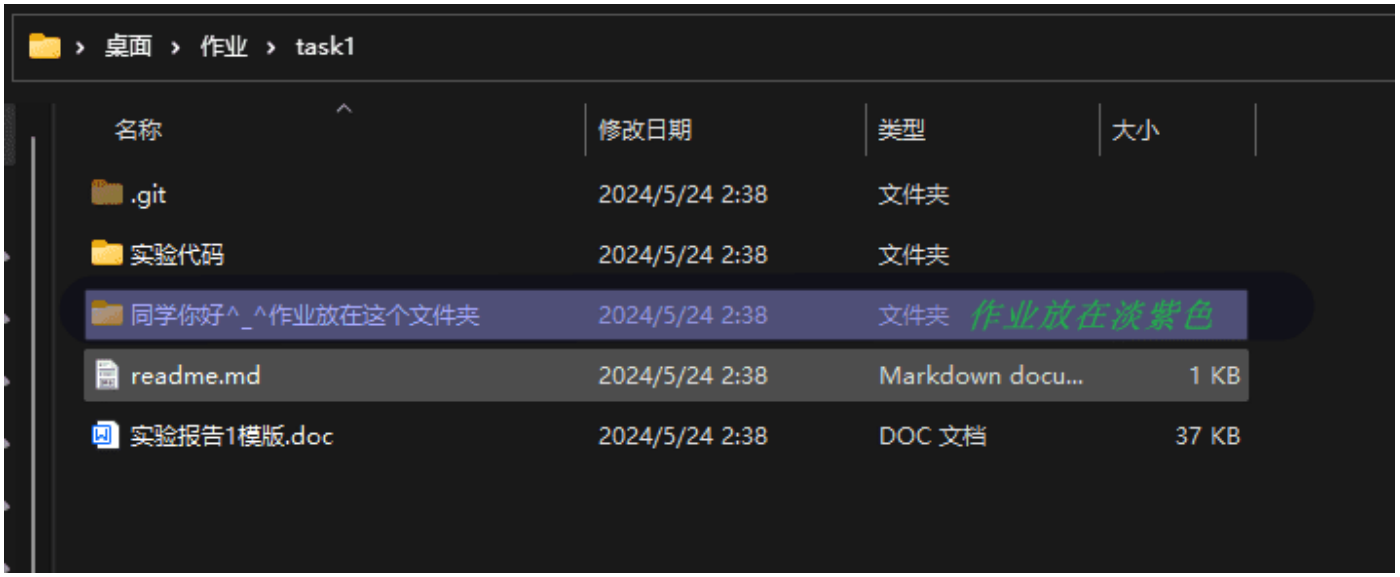
3.cmd中输入

```
git clone https://gitee.com/he-knows/task1.git
```

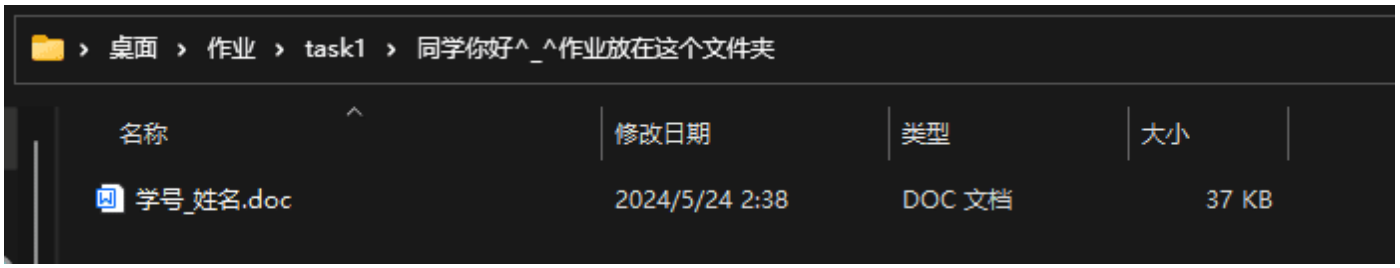


4.进入

```
cd task1  # tab键 快速弹出目录 cd t
```



5.放入你的word文档



6. shell中输入

```
git add .
git status
git conmit -m " 你的姓名 完成作业1"

git push origin master  #推到远程仓库
```

7.成功

添加本地ssh秘钥到远程仓库平台

1. 生成了**SSH密钥对**,运行以下命令来生成一个RSA密钥对（默认情况下会保存在`~/.ssh/`目录下，`id_rsa`是私钥文件，`id_rsa.pub`是公钥文件）：

```
ssh-keygen -t rsa
```

- **-t**: 指定密钥的类型。在这里，`rsa` 表示我们将生成一个RSA类型的密钥对。

- **rsa**: 随 `-t` 之后，指定实际的密钥类型名称。RSA是目前较为常用的一种密钥类型，尽管Ed25519因为其更强的安全性而逐渐变得流行。

2. **查找公钥**: SSH公钥通常存储在`.ssh/id_rsa.pub`（如果你使用默认设置和RSA密钥类型）或者相应私钥的`.pub`文件中。你可以用文本编辑器打开这个文件查看公钥，或者使用如下命令打印公钥到终端：

```
cat  ~/.ssh/id_rsa.pub
```

3. **添加公钥到GitHub账户**: 登录到GitHub，点击右上角的头像，选择“Settings” > “SSH and GPG keys” > “New SSH key”。在“Title”框中输入一个便于记忆的标题（例如，“Home Laptop”），在“Key”框中粘贴你的公钥（即你在上一步中复制的内容）。完成后点击“Add SSH key”。

4. 登录到Gitee，点击右上角的头像，选择“账号设置” > “安全设置” > “SSH公钥”。在“标题”框中输入一个便于记忆的标题（例如，“xxx的电脑”），在“公钥”框中粘贴你的公钥（即你在上一步中复制的内容）。完成后点击“确定”。

5. **测试连接**: 可以通过运行以下命令来测试你的电脑是否能成功通过SSH连接到GitHub：

```
ssh -T git@github.com
ssh -T git@gitee.com
```

配置正确，你会看到类似于“Hi username! You've successfully authenticated...”的消息

一个图形化的Git客户端 Tortoisegit

自行了解

TortoiseGit 使用教程 - 啊，那一个人 - 博客园

Git 高级进阶

1. [git-简明指南](git - the simple guide - no deep shit!) [图形化模式，简单易懂]

2. 图解Git [一样是图形化教程]

3. .Git的奇技淫巧 [GitHub 14.9k stars]