

# Git学习

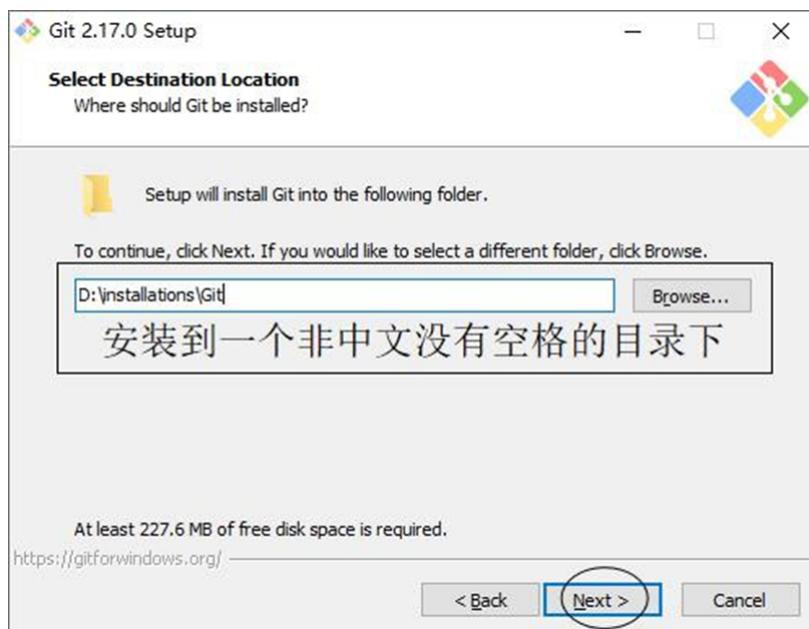
## 第一章、git简介

### 1.1 git安装

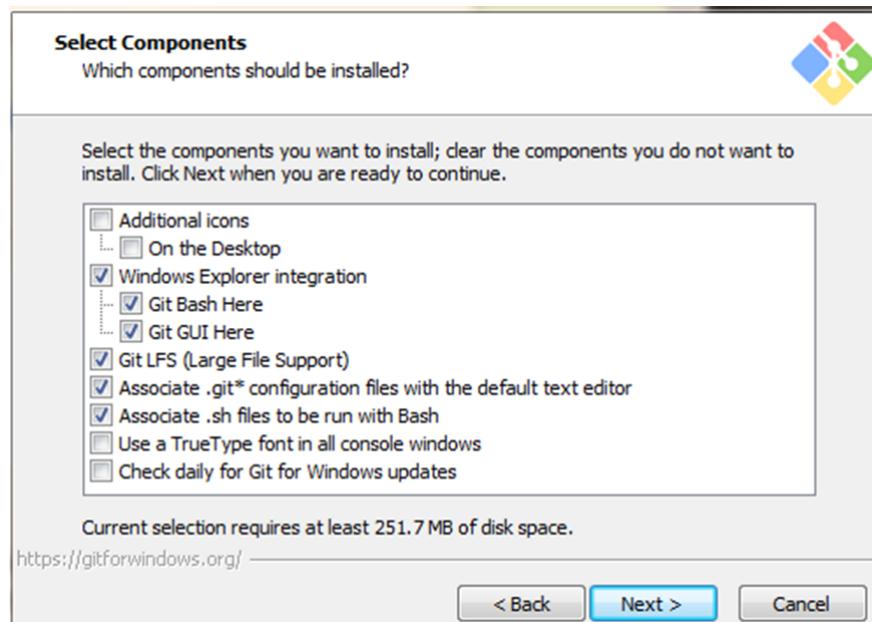
- windows安装

<https://www.cnblogs.com/wlming/p/12213876.html>

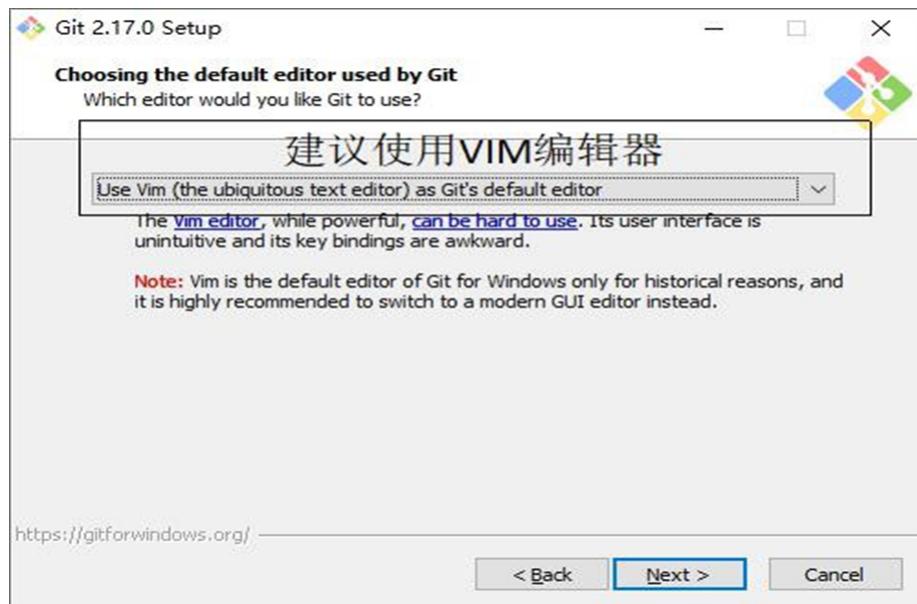
#### 1. 开始安装



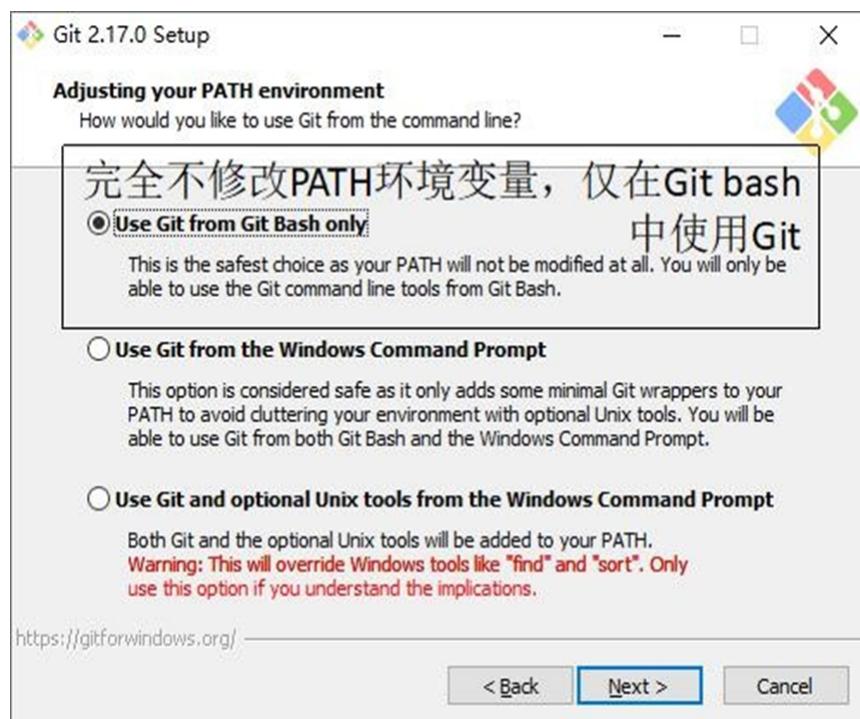
#### 2. 下面默认设置就行:下图(下一步), 这个的下一步也使用默认 直接下一步



#### 3. 选择默认的文本编辑器



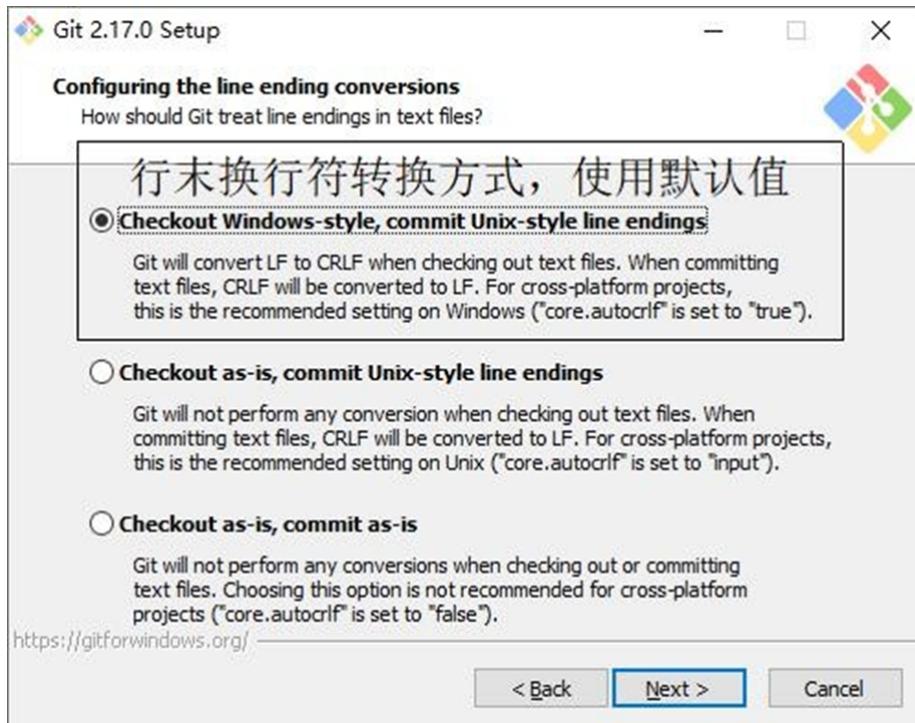
#### 4. 然后修改环境变量(选第一完全不修改)



#### 5. 选择客服端本地库和远程库连接方式(1通用连接2使用Windows连接方式)



6. 选择换行符的方式(1检查文件时LF 转为 CRLF 提交相反)



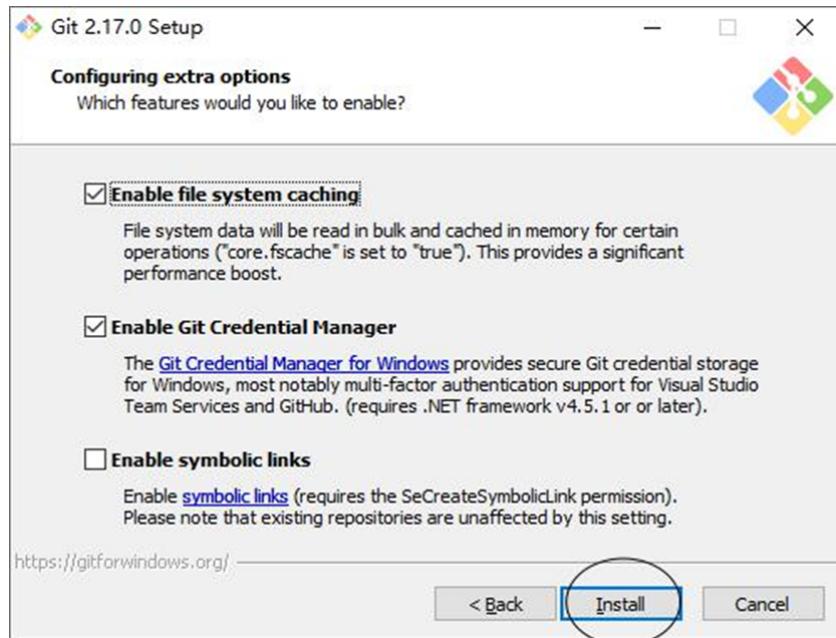
7. 选择终端(1 Git默认终端(是liunx命令)2选择Windows终端(wind命令))



8. 使用默认(选择第二个需要安装.NET framework c4.5.1以上版本)

NET framework安装失败解决方案:

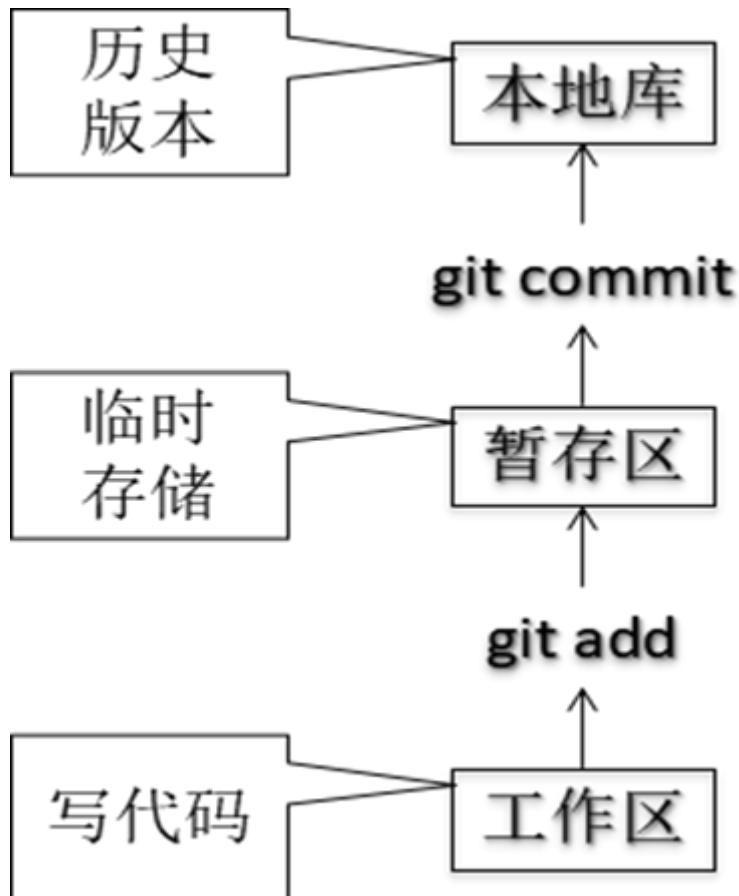
<https://jingyan.baidu.com/article/fb48e8bee50ebf6e632e1464.html>



- linux安装

```
yum install -y git
```

## 1.2 git结构——本地库、暂存区、工作区



## 1.3 远程库介绍（代码托管）

代码托管中心的任务：维护远程库

- 局域网环境下

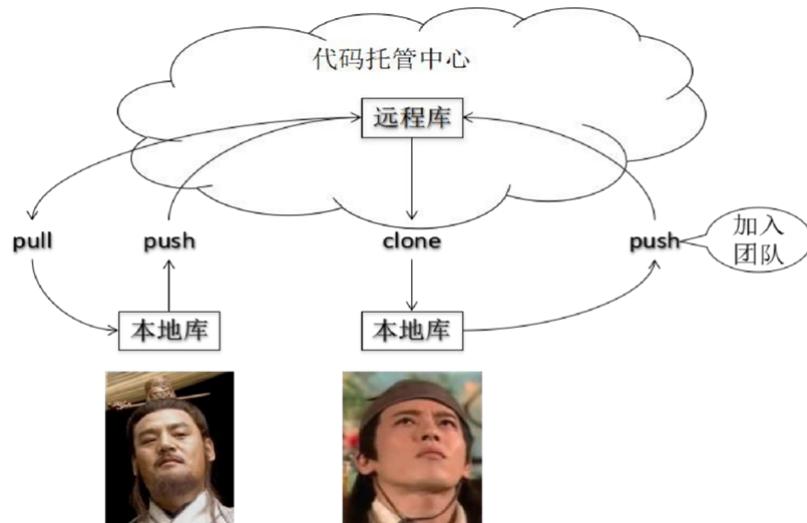
GitLab 服务器

- 外网环境下

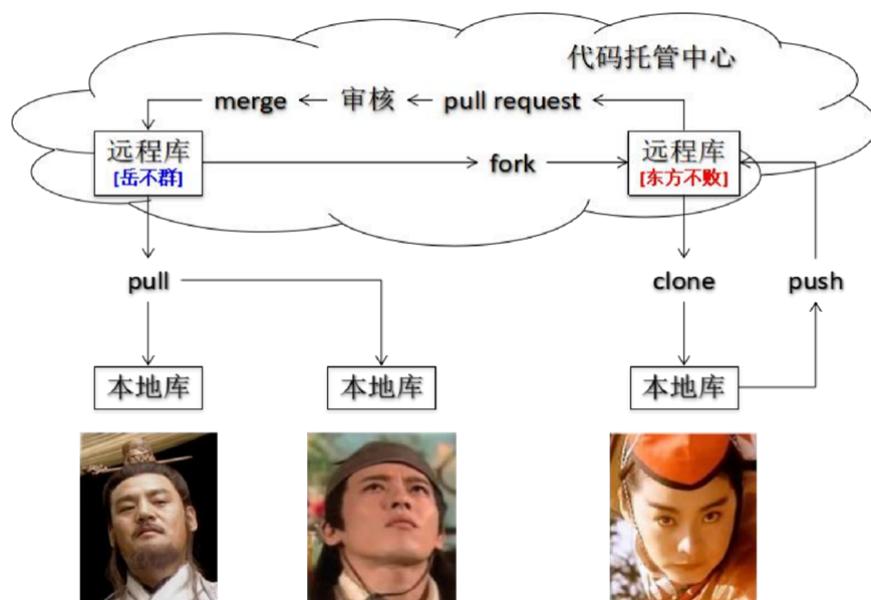
GitHub

码云

- 团队内部协作



- 跨团队协作



## 第二章、git命令行操作

### 2.1 git init命令——本地库初始化

切换到目录>右键Git Bash Here>用linux命令到对应目录下>初始化

```
git init
```

效果: 会在当前目录生成 .git 目录 (隐藏的)

## 2.2 设置签名——本地库初始化后,要执行的形式

用户名: tom

Email 地址: [goodMorning@atguigu.com](mailto:goodMorning@atguigu.com)

作用: 区分不同开发人员的身份

辨析: 这里设置的签名和登录远程库(代码托管中心)的账号、密码没有任何关系

### 命令

**项目级别 / 仓库级别:** (不带参数)-仅在当前本地库范围内有效

```
git config user.name tom_pro          #设置用户名
```

```
git config user.email goodMorning_pro@taku.com      #设置邮箱地址
```

- 信息保存位置: `./.git/config` 文件

**系统用户级别:** 登录当前操作系统的用户范围

```
git config --global user.name tom_pro          #设置用户名
```

```
git config --global user.email goodMorning_pro@taku.com      #设置邮箱地址
```

- 信息保存位置: `~/.gitconfig` 文件

### 级别优先级

- 就近原则: 项目级别优先于系统用户级别,二者都有时采用项目级别的签名
- 如果只有系统用户级别的签名,就以系统用户级别的签名为准
- 二者都没有不允许

## 第三章、基本操作

### 3.1 git status命令——查看工作区、暂存区状态

```
git status          #第1代表是那个分区的,第2是否提交3有没有可提交的文件和提示
```

### 3.2 git add [file name]——工作区提交到暂存区

/\*提交到暂存区,并且转换换行符

```
git add [file name]      #将工作区的“新建/修改”添加到暂存区
```

### 3.3 git commit——提交到本地库

将暂存区的内容提交到本地库

```
git commit [file name] [备注信息]  
git commit -m "备注信息" [file name]
```

## 3.4 git rm [--cached -f]——暂存区撤回

- rm 仅删除工作区文件，并没有删除版本库的文件，想要删除版本库文件还要add和commit
- git rm 删除工作区未修改文件，并且将这次删除放入暂存区，想要删除版本库文件只要commit
- git rm -f 删除工作区已修改文件，并且将这次删除放入暂存区，想要删除版本库文件只要commit
- git rm --cached 删除了暂存区和版本库的文件，但保留了工作区的文件，并且将这次删除放入暂存区，想要删除版本库文件只要commit。

### 3.4.1 rm——删除工作区文件

```
rm [file name]
```

rm 命令只是删除工作区的文件，并没有删除版本库的文件，想要删除版本库文件还要执行下面的命令：

```
git add [file name]  
git commit -m "delete [file name]"
```

**结果：**删除了工作区和版本库的文件。

### 3.4.2 git rm——删除工作区文件，并且将这次删除放入暂存区。

**注意：**要删除的文件是没有修改过的，就是说和**当前版本库文件的内容相同**。

```
git rm [file name]
```

查看状态（成功删除了工作区文件，并且将这次删除放入暂存区。）

然后提交：

```
git commit -m "delete [file name]"
```

**结果：**删除了工作区和版本库的文件，因为暂存区不可能有该文件（如果有意味着该文件修改后 git add 到暂存区，那样 git rm 命令会报错，如下面的情况）。

### 3.4.3 git rm -f——删除工作区和暂存区文件，并且将这次删除放入暂存区。

**注意：**要删除的文件已经修改过，就是说和**当前版本库文件的内容不同**。

```
git rm -f [file name]
```

然后提交：

```
git commit -m "delete [file name]"
```

**结果：**删除了工作区、暂存区和版本库的文件。

### 3.4.4 git rm --cached——删除暂存区文件，但保留工作区的文件，并且将这次删除放入暂存区。

删除暂存区文件，但保留工作区的文件，并且将这次删除放入暂存区

```
git rm --cached [file name]
```

然后提交：

```
git commit -m "delete [file name]"
```

**结果：**删除了暂存区和版本库的文件，但保留了工作区的文件。如果文件有修改并 git add 到暂存区，再执行 git rm --cached 和 git commit，那么保留的工作区文件是修改后的文件，同时暂存区的修改文件和版本库的文件也被删了。

## 3.5 git log——查看本地库版本记录

```
git log                                #查看本地库版本记录,空格向下翻页 ,b 向上翻页 ,q 退出(超过了自动多屏)
git log --pretty=oneline                #每个历史只显示一行(hash值和日志),更人性化
git log oneline                         #每个历史只显示一行(hash值和日志),且hash值只显示部分,更人性化
git reflog                             #显示历史只显示一行,并且显示指针(要移动到版本多少步),很强大
```

## 3.6 git reset——前进后退

在git reflog的基础上

```
git reset --hard [局部索引值]          #基于索引值操作
git reset --hard HEAD^^                 #使用^符号, 只能后退到先前版本, 一个^表示后退一步
git reset --hard HEAD~n                  #使用~符号, 只能后退, n表示后退 n 步
```

**git reset三个参数对比：**

- --soft

**仅仅在本地库移动 HEAD 指针 (查看状态时,绿色提示,本地库和暂存区不同步)**

暂存区

工作区

本地库

- --mixed

在本地库移动 HEAD 指针

重置暂存区

## 工作区

本地库

暂存区

- --hard

在本地库移动 HEAD 指针

**重置暂存区**

**重置工作区**

## 3.7 git diff——比较文件的差异

```
git diff [文件名]
```

#比较工作区的文件和暂存区的文件差异

```
git diff [本地库版本索引值] [文件名]
```

#通过版本索引比较本地库和工作区文件差异

```
git diff [HEAD^] [文件名]
```

#通过HEAD指针比较本地库和工作区文件差异

**不带文件名会比较多个文件**

## 3.8 分支操作

同时并行推进多个功能开发，提高开发效率

各个分支在开发过程中，如果某一个分支开发失败，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。

- 创建分支

```
git branch [分支名]
```

- 查看分支

```
git branch -v
```

- 切换分支

```
git checkout [分支名]
```

- 合并分支

```
git merge [有新内容分支名]
```

1. 第一步：切换到接受修改的分支(被合并,增加新内容)上
  2. 第二步：执行 merge 命令 (合并分支指令)
- 解决冲突

冲突原因：2个分支,修改同一文件,同一位置,修改内容不一样时.

```
8 gggggggg
9 <<<<< HEAD
10 hhhhhhhh edit by hot_fix → 当前分支内容
11 =====
12 hhhhhhhh edit by master → 另一分支内容
13 >>>>> master
14 ii iiiiiii
15 jj jjjjjjjj
```

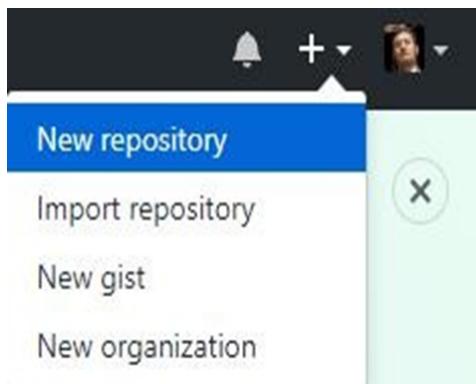
冲突的解决：

1. 第一步：编辑文件，删除特殊符号
2. 第二步：把文件修改到满意的程度，保存退出
3. 第三步：git add [文件名]
4. 第四步：git commit -m "日志信息"

注意：此时 commit 一定不能带具体文件名

## 第四章、远程库操作——github操作

### 4.1 创建远程库



## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: atguigu2018ybuq / Repository name: haushan ✓

Great repository names are short and memorable. Need inspiration? How about fuzzy-octo-lamp.

Description (optional):

Public  
Anyone can see this repository. You choose who can commit.

Private  
You choose who can see and commit to this repository.

Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾ Add a license: None ▾ ⓘ

Create repository 点这里创建即可

## 4.2 创建远程库地址别名

- 远程库地址

Quick setup — if you've done this kind of thing before

Set up in Desktop or  HTTPS  SSH <https://github.com/> [点击复制远程地址](#) 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

- 查看各远程地址别名

```
git remote -v #查看当前所有远程地址别名
```

- 创建别名

```
git remote add [别名] [远程地址]
```

## 4.3 本地库推送到远程库

```
git push [别名] [分支名]  
git push [远程地址] [分支名]
```

(可能需要等待一会儿,弹出对话框 > 输入用户和密码)

## 4.4 克隆——远程库下载到本地

The screenshot shows a GitHub repository page for a project named 'ub'. At the top, there are three status indicators: '0 packages', '0 releases', and '0 contributors'. Below them is a navigation bar with buttons for 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. A red arrow points from the text '推送了后,查询地址,点击这里.' to the 'Clone or download' button. To the right of the button is a 'Clone with HTTPS' section with a 'Use SSH' link and a text input field containing the URL 'https://github.com/'. A red arrow points from the text '然后复制地址' to the URL input field. Below the URL field is a text box containing the command 'git clone [远程地址]'.

克隆效果:

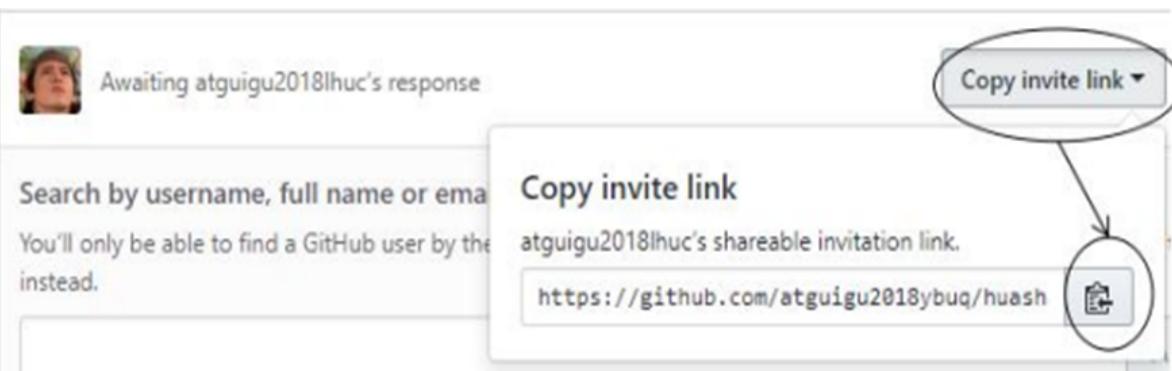
完整的把远程库下载到本地

创建 origin 远程地址别名 (git remote -v 查看远程库别名)

初始化本地库(就是:git init)

## 4.5 团队成员邀请

The screenshot shows the 'Settings' tab of a GitHub repository. On the left, a sidebar lists options like 'Manage access' (marked with a circled '1'), 'Branches', 'Webhooks', 'Notifications', 'Integrations', 'Deploy keys', 'Secrets', 'Actions', 'Moderation', and 'Interaction limits'. The 'Manage access' option is highlighted. A modal window titled 'Who has access' is open, prompting to 'Invite a collaborator to huashan'. It features a search bar for 'Username' (marked with a circled '3') and a green button 'Select a collaborator above' (marked with a circled '4'). Below the modal, a message says 'You haven't invited any collaborators yet'. At the bottom of the modal is a green 'Invite a collaborator' button (marked with a circled '2'). Red boxes with arrows point from the text '输入 邀请人.github账号' to the 'Username' input field and from '邀请一位合作者' to the 'Invite a collaborator' button.



“岳不群”其他方式把邀请链接发送给“令狐冲”，“\*\*令狐冲”登录自己的\*\* GitHub 账号，访问邀请链接。

点击接受 > 然后在执行推送



**atguigu2018ybuq invited you to collaborate**

**Accept invitation**

**Decline**

- 推送了第一次后再次推送不要输入用户名
- git 本身不具备记录功能, Windows 中凭据管理器记录用户名和密码

控制面板 -> 所有控制面板项 -> 凭据管理器 (如果想切换用户: 删除记录)

普通凭据 添加普通凭据

git:https://github.com	修改时间: 今天
Internet 地址或网络地址: git:https://github.com	
用户名: PersonalAccessToken	
密码: .....	
永久性: 本地计算机	
<a href="#">编辑</a> <a href="#">从保管库中删除</a>	

## 4.6 pull——拉取

pull=fetch+merge

```
git pull [远程库地址别名] [远程分支名]
```

等同于下面步骤

```

git fetch [远程库地址别名origin] [远程分支名master]    #抓取下来
git checkout origin/master                           #切换到链接地址(别名)的master(可查看
抓取下来内容
git checkout master                                #切换回
git merge [远程库地址别名origin/master远程分支名]   #合并

```

## 4.7 远程库解决冲突

- 如果不是基于 GitHub 远程库的最新版所做的修改，不能推送，必须先拉取。
- 拉取下来后如果进入冲突状态，则按照**分支冲突解决**操作解决即可。

## 4.8 跨团队协作

dfbb表示开发人员，ybq表示审核人员

- (先复制当前库地址,发式给dfbb,然后有dfbb登录访问这个地址)>然后Fork

fork过来的仓库说明回多下面一行(forked from at...)说明fork来源



- dfbb("东方不败")本地修改，然后推送到远程 git push origin master
- dfbb在远程库中选择Pull Request

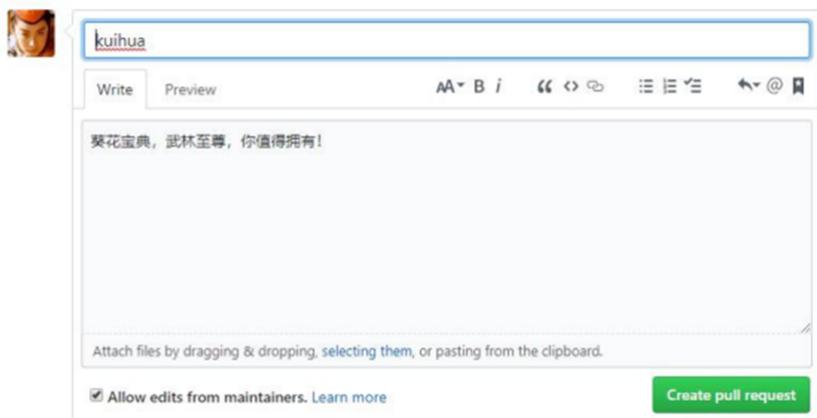
1. 然后点击里面的New pull request

New pull request

2. 然后点击 Create pull request

Create pull request

3. 然后发送消息给, fork的库(ybq(岳不群))



- ybq操作 (审核)

atguigu2018ybuq / huashan

Code

Issues 0

Pull requests 1

“岳不群”点这里

1 Open 0 Closed

老岳，你拜托我的事情办好啦！

#1 opened 2 minutes ago by atguigu2018east

“岳不群”点这里

- 可对话



atguigu2018east commented 4 minutes ago

葵花宝典，武林至尊，你值得拥有！



kuihua



atguigu2018ybuq commented 12 seconds ago

老东，你这代码靠谱吗？练了会不会有什么危险？



atguigu2018east commented just now

你放心吧，我都亲自练过啦！

- 审核代码

Conversation 2

Commits 1

Checks 0

Files changed 1

这是查看对话

这是看做了哪些提交

点击这里查看代码

Changes from all commits ▾ Jump to... ▾ +1 -0

1 huashanjianfa.txt



@@ -2,3 +2,4 @@

2 2 我是令狐冲，我比岳不群还厉害！edit by lhuc

3 3 我是令狐冲，我比岳不群还厉害！edit by ybuq

4 4 我会独孤九剑，哈哈，你不会！

5 +小葵花课堂开讲啦！孩子葵花宝典老练不好怎么办？多半是装的，打一顿就好啦！

- 合并代码

合并代码 (回到对话Conversation>>合并操作如图)

 Continuous integration has not been set up  
Several apps are available to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch  
Merging can be performed automatically.

**Merge pull request** You can also open this in GitHub Desktop or view command line instructions.

点这里合并代码

Add more commits by pushing to the `master` branch on `atguigu2018east/huashan`.

 Merge pull request #1 from `atguigu2018east/master`

既然老东都说了没问题，那我就练一下试试吧！  
.....  
“啊.....啊.....！！！”  
“东方不败！你骗我！”

**Confirm merge** **Cancel**