

# Git 使用文档

## 一：Git介绍

<https://www.runoob.com/git/git-tutorial.html>

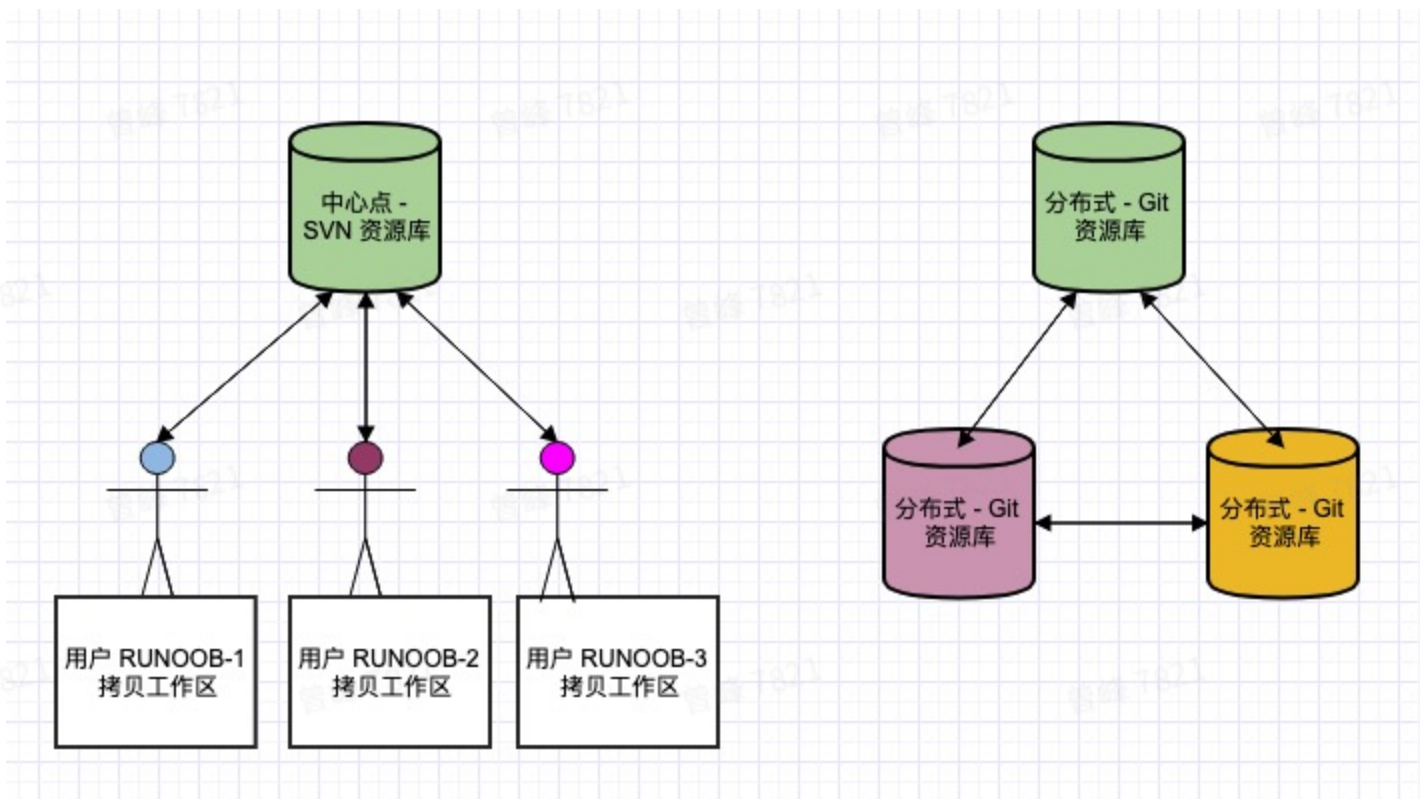
### 1.Git 与 SVN 区别

Git 不仅仅是个版本控制系统，它也是个内容管理系统(CMS)，工作管理系统等。

如果你是一个具有使用 SVN 背景的人，你需要做一定的思想转换，来适应 Git 提供的一些概念和特征。

Git 与 SVN 区别点：

- **1、Git 是分布式的，SVN 不是：**这是 Git 和其它非分布式的版本控制系统，例如 SVN，CVS 等，最核心的区别。
- **2、Git 把内容按元数据方式存储，而 SVN 是按文件：**所有的资源控制系统都是把文件的元信息隐藏在一个类似 .svn、.cvs 等的文件夹里。
- **3、Git 分支和 SVN 的分支不同：**分支在 SVN 中一点都不特别，其实它就是版本库中的另外一个目录。
- **4、Git 没有一个全局的版本号，而 SVN 有：**目前为止这是跟 SVN 相比 Git 缺少的最大的一个特征。
- **5、Git 的内容完整性要优于 SVN：**Git 的内容存储使用的是 SHA-1 哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。



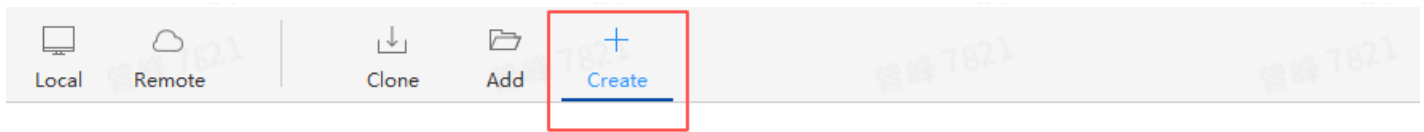
- Git是以仓库为整体clone到本地。不能像SVN以某个目录单独checkout到本地。
- Git是以分支方式提交。SVN是以文件提交。
- Git提交完只是提交到本地仓库，还需要推送到服务器仓库。SVN提交就直接提交到了服务器仓库。

## 2.如何创建仓库。

我们项目用第三种方式，公司内网gitlab。

### 方式一：本地创建仓库

安装过git，就可以在本地创建仓库。这种方式基本不怎么用。



# Create a repository

☐ 在账户中创建仓库:

## 方式二：git服务平台

常用的git服务平台有：

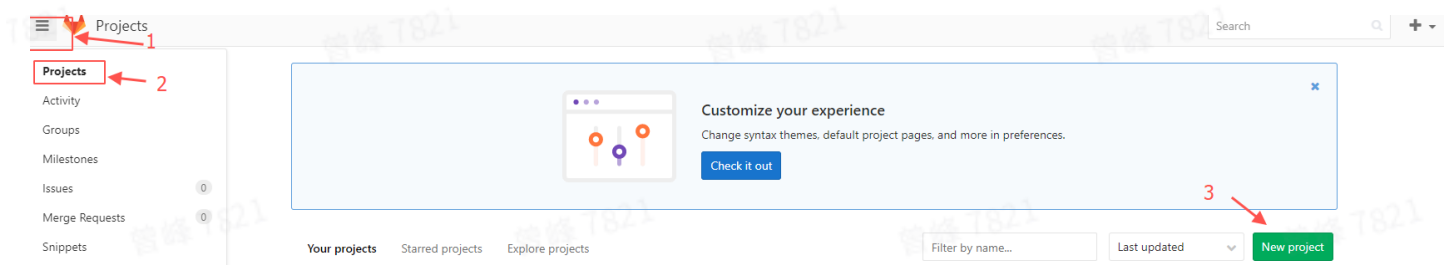
国际 github: <https://github.com/>

国内 码云: <https://gitee.com/>

## 方式三：gitlab

公司一般使用gitlab在公司自己内网搭建服务

<http://git.intra.123u.com/>



Project path:

Project name:  1 必填，仓库名称

Want to house several dependent projects under the same namespace? [Create a group](#)

Project description (optional):  2 仓库介绍

Visibility Level

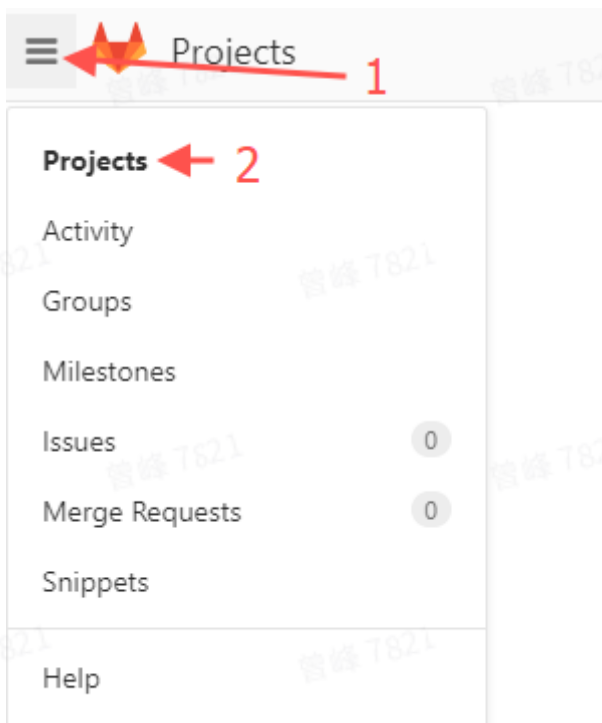
- ☒ Private  
Project access must be granted explicitly to each user.
- ☐ Internal  
The project can be accessed by any logged in user.
- ☐ Public  
The project can be accessed without any authentication.

3 权限选择，  
private 私密，其他成员需要加权限才能访问  
internal 公司内部所有人都可以访问  
public 公开，只要能拿到链接的人都可以访问

4 确定

### 3. 查看自己拥有权限的仓库

<http://git.intra.123u.com/dashboard/projects>  
登录公司gitlab后。



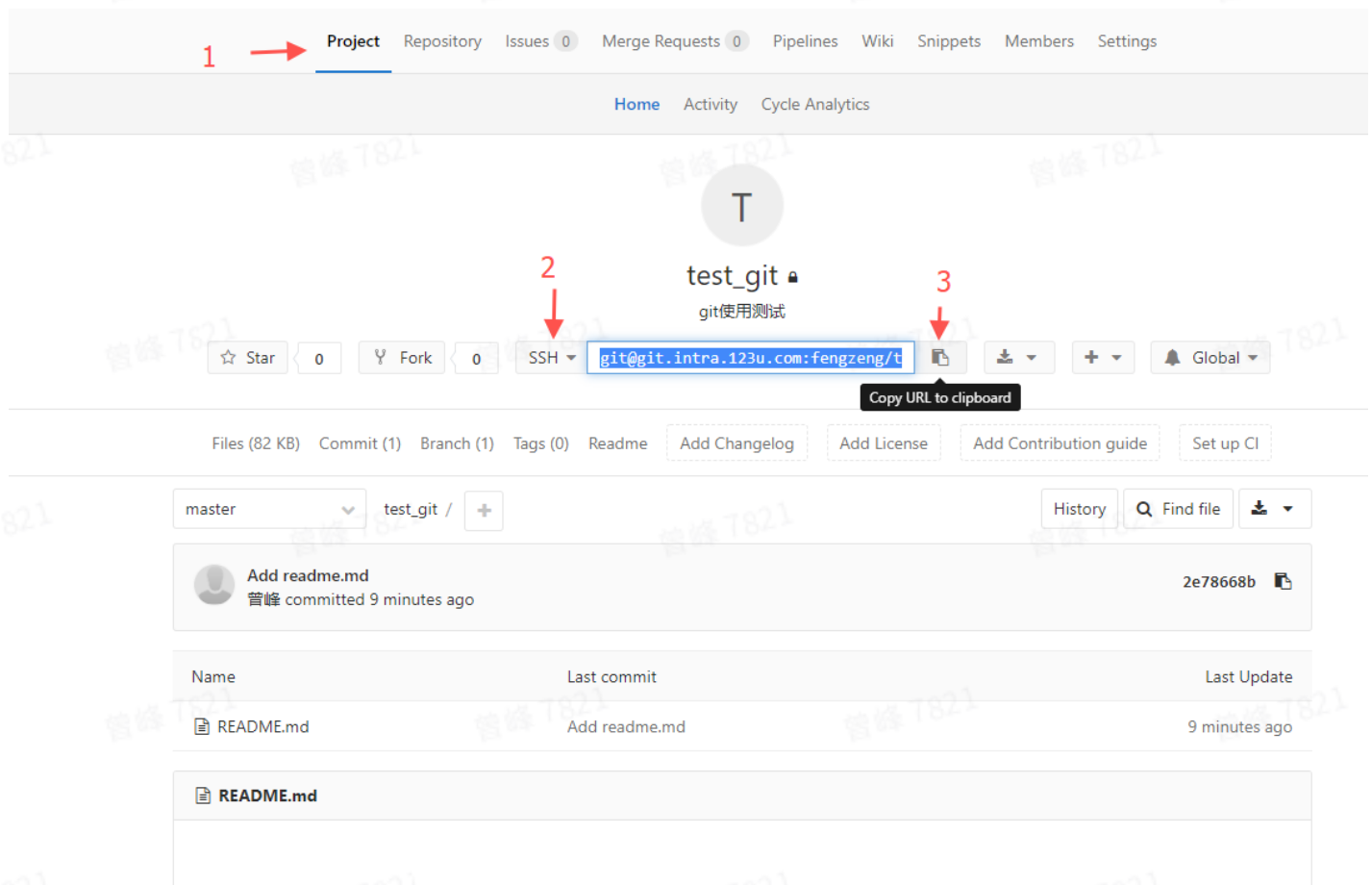


## 4. 查看仓库地址

点击进入仓库



按步骤，点击复制路径按钮就可以复制路径



这里介绍一下2种路径的区别:

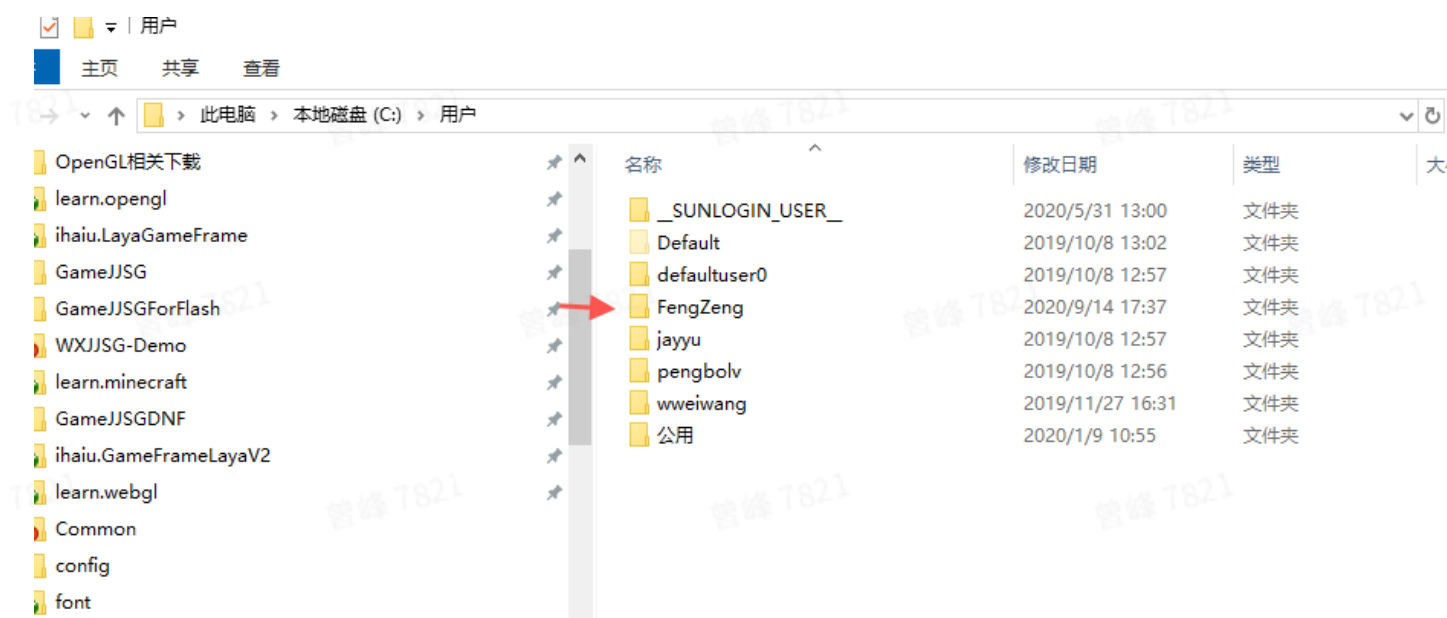
- HTTP: 在本地克隆仓库的时候需要输入账号和密码。
- SSH: 需要配置公钥，以后本地做git操作就不用每次输入账号密码了。



## 5. 公钥配置

### 检测本地是否有sshkey

进入用户文档目录下，C:\Users，再进入到自己用户名目录。



检测是否存在.ssh文件夹。

(C:) > 用户 > FengZeng

名称	修改日期
.LayaAirIDE2.0	2019/3/22 10:11
.LdVirtualBox	2020/9/8 11:32
.local	2020/1/9 10:16
.npminstall_tarball	2020/2/28 14:41
.nuget	2019/3/25 10:42
.ssh	2019/3/23 10:02
.templateengine	2019/6/15 14:05
.vscode	2019/11/14 10:3
3D 对象	2019/12/14 9:53
Adobe Flash Builder 4.7	2019/6/27 21:19
AppData	2020/6/10 9:55
CMakeBuilds	2020/1/30 11:45
Documents - 副本	2019/6/3 19:15
go	2019/10/21 19:1
MicrosoftEdgeBackups	2019/12/20 14:3

如果有这2个文件说明已经创建过秘钥id\_rsa和公钥id\_rsa.pub

ngZeng > .ssh

名称
id_rsa
id_rsa.pub
known_hosts

## 创建sshkey

(1) 打开git bash



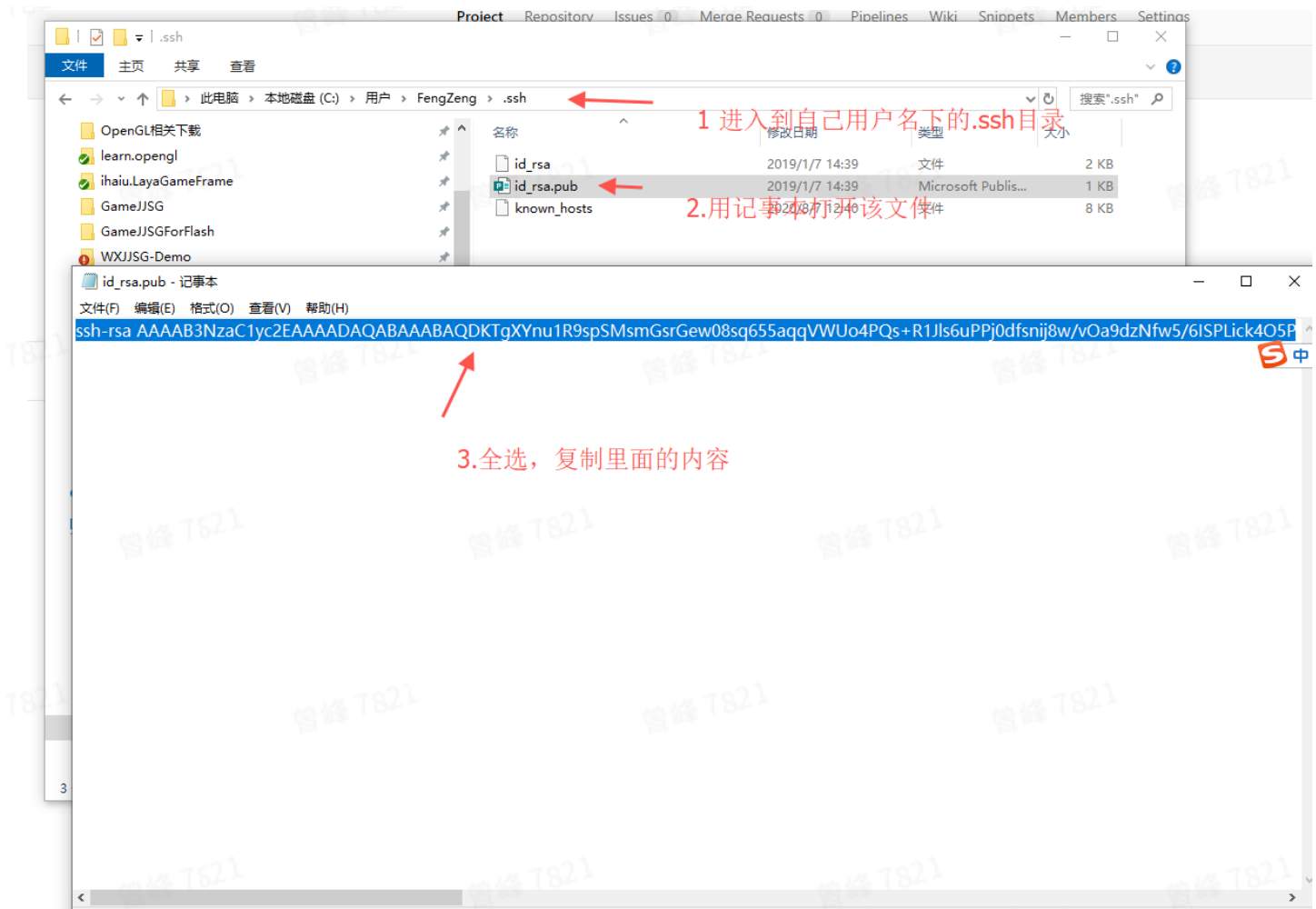
(2) 输入以下命令

- 1 `cd ~`
- 2 `mkdir .ssh`
- 3 `cd .ssh`
- 4 `## 输入以下命令后 连续点几次回车键`
- 5 `ssh-keygen -t rsa -C"这里写自己的用户名"`

## 在gitlab上配置公钥

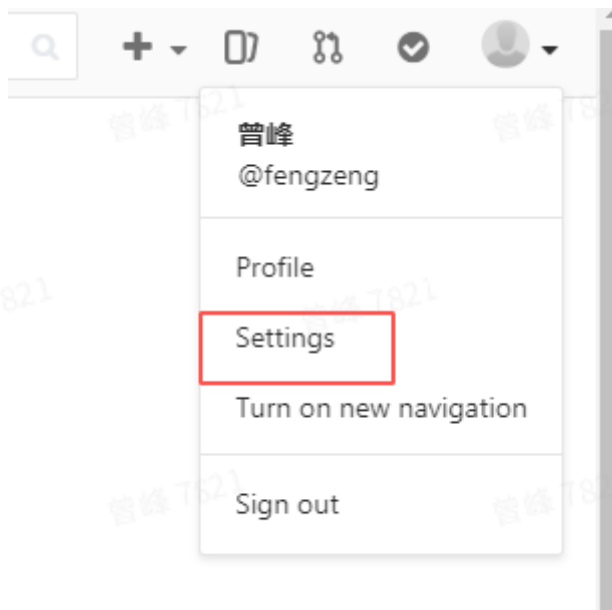
用文本编辑器打开id\_rsa.pub，拷贝里面的内容





进入gitlab添加ssh key

<http://git.intra.123u.com/profile/keys>



Chat Access Tokens Emails Notifications **SSH Keys** GPG Keys Preferences Authentication log

Add an SSH key

Before you can add an SSH key you need to generate it.

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDKTgXYnu1R9spSMsmGsrGew08sq655aqqVWUo4PQs+
R1Jls6uPPj0dfsnij8w/vOa9dzNfw5/6ISPI ick4O5PqitlzKUHN87qTJl1wgXlgs9SmBq10/u3ljghplSxiiig/
kcd5jLNOqy9RLRgLQWD7H0RUwGf5G0u10PQK6oPvyH45A5mnRbbbei3s8iD8pz8mjz6mrUYhtn
mvAmRmb7zrD/VhG0F2J0H-4FQ0-Gil5M4T7wgyuF2UP2xi592QSDo6nh4I18k/MLtZQ6TTjEiDo
8cJ2cldY1wUft+LQvncIAY2+Jk89ys/HkUvsaOeM4nQabNxBYwF0H0mny-CZengfeng_Joyyou
```

Title

-CZengFeng\_Joyyou

Add key

1. 进入该页签

2. 将刚才复制的公钥粘贴到这里

3. 做完第2步，一般会自己填写。没有的话自己随便写一

4. 点击添加

## 在SourceTree上配置私钥

文件(F) 编辑(E) 查看(V) 仓库(R) 操作(A) 工具(T) 帮助(H)

ihaiu.GPUSki 2 ↑ ihaiu.Laya3DZip ihaiu.Laya3DZip

提交 拉取 推送 获取 分支

WORKSPACE

文件状态

启动SSH助手...  
添加SSH密钥...  
进程查看器  
选项(O)

待定的文件, 已...  
已暂存文件

选项

一般 Updates 比较 Git Mercurial 自定义操作 验证 网络

☒ 允许 Sourcetree 修改您的全局 Git 和 Mercurial 配置文件

☒ Open links on Bitbucket.org with Sourcetree

☒ 在未来提议创建书签

Theme: Light

4. 配置一下自己的用户名，方便区别谁提交的

默认用户信息

全名: zengfeng75

电子邮件地址: zengfeng75@gmail.com

SSH客户端配置

SSH 密钥: C:\Users\FengZeng\.ssh\id\_rsa

SSH 客户端: OpenSSH (仅针对 Git, Mercurial 总是使用 Plink on Windows)

2 3. 做完步骤2自动会填

Repo Settings

项目目录:

语言: 自动 (需要重启) 帮助翻译 Sourcetree!

默认文本编码: utf-8

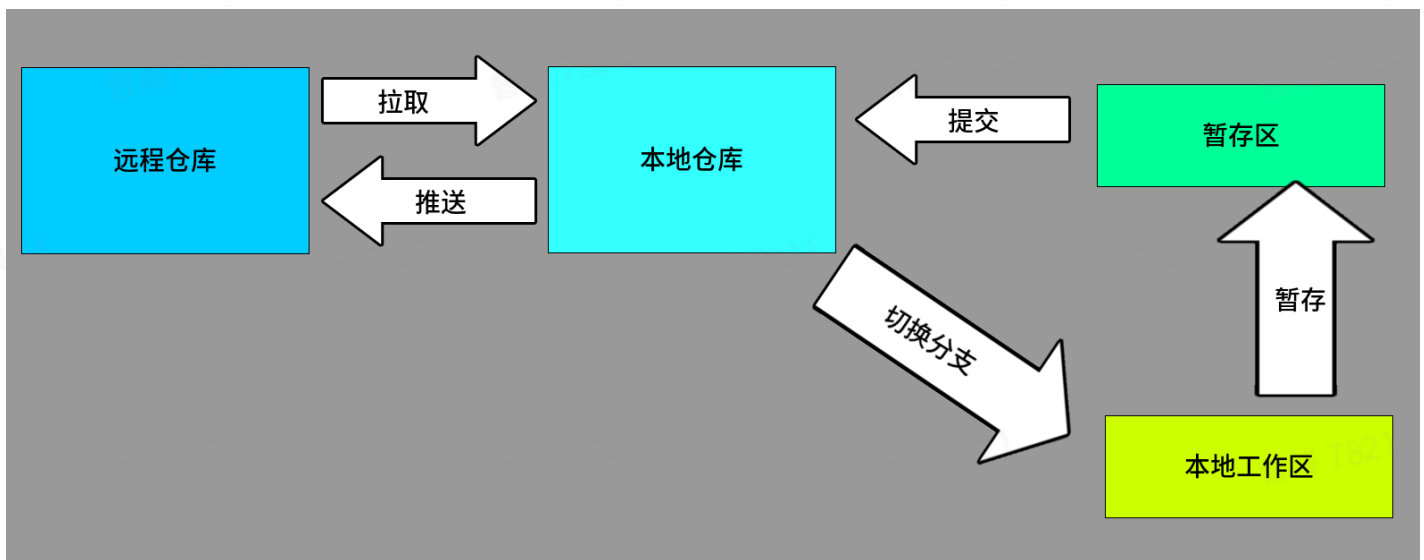
☒ 进行破坏性操作时保留备份

☐ 当文件有改动时自动刷新

5

确定

## 6. Git仓库关系介绍



本地仓库和暂存区都存放在.git目录下。



本地工作区就是除.git外的所有文件。

本地工作区就是除.git的其他所有文件

本地工作区

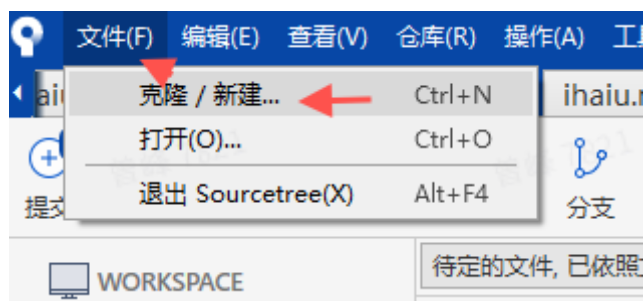
.git  
game  
a.txt  
a3.txt  
b.txt  
image.png  
README.md  
text.txt

## 二：SourceTree 使用

<https://blog.csdn.net/syk123839070/article/details/82534968>

<https://www.cnblogs.com/tian-xie/p/6264104.html>

### (1) 克隆仓库到本地



# Clone

Cloning is even easier if you set up a [remote account](#)

1. 将git仓库服务器地址拷贝到这里

git@git.intra.123u.com:fengzeng/test\_git.git

浏览

仓库类型:  这是一个 Git 仓库

E:\zengfeng\githubs\\_test\test\_git

浏览

test\_git

2. 选择本地目录

Local Folder:

[根]

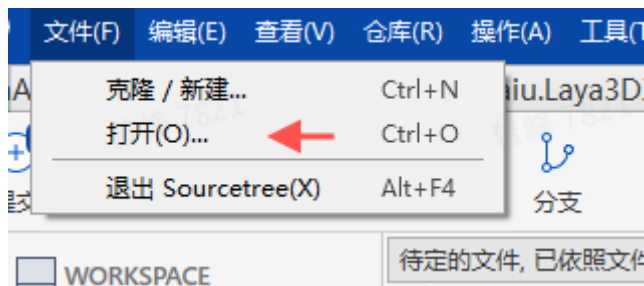
3. 这里是本地仓库名称，尽量和目录文件名相同

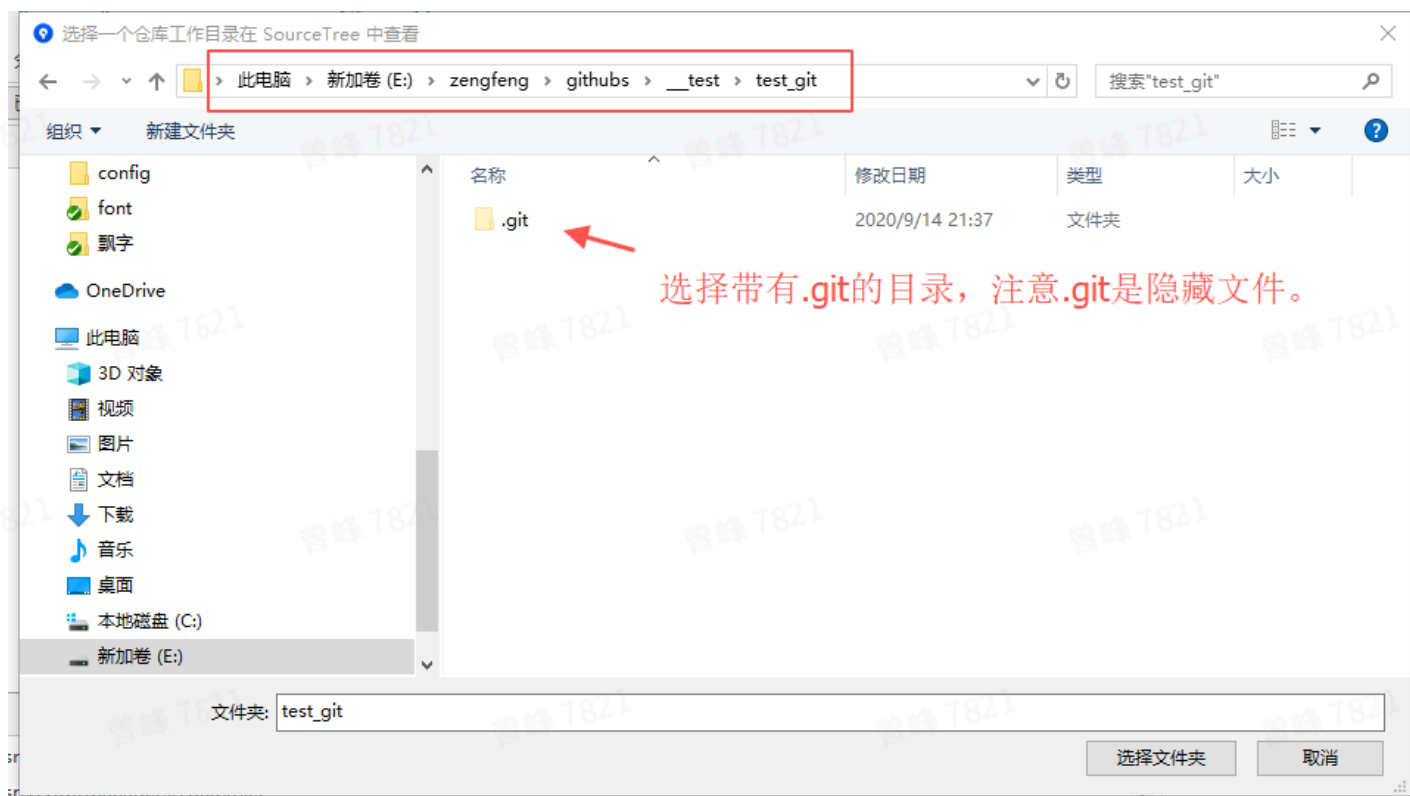
> 高级选项

克隆

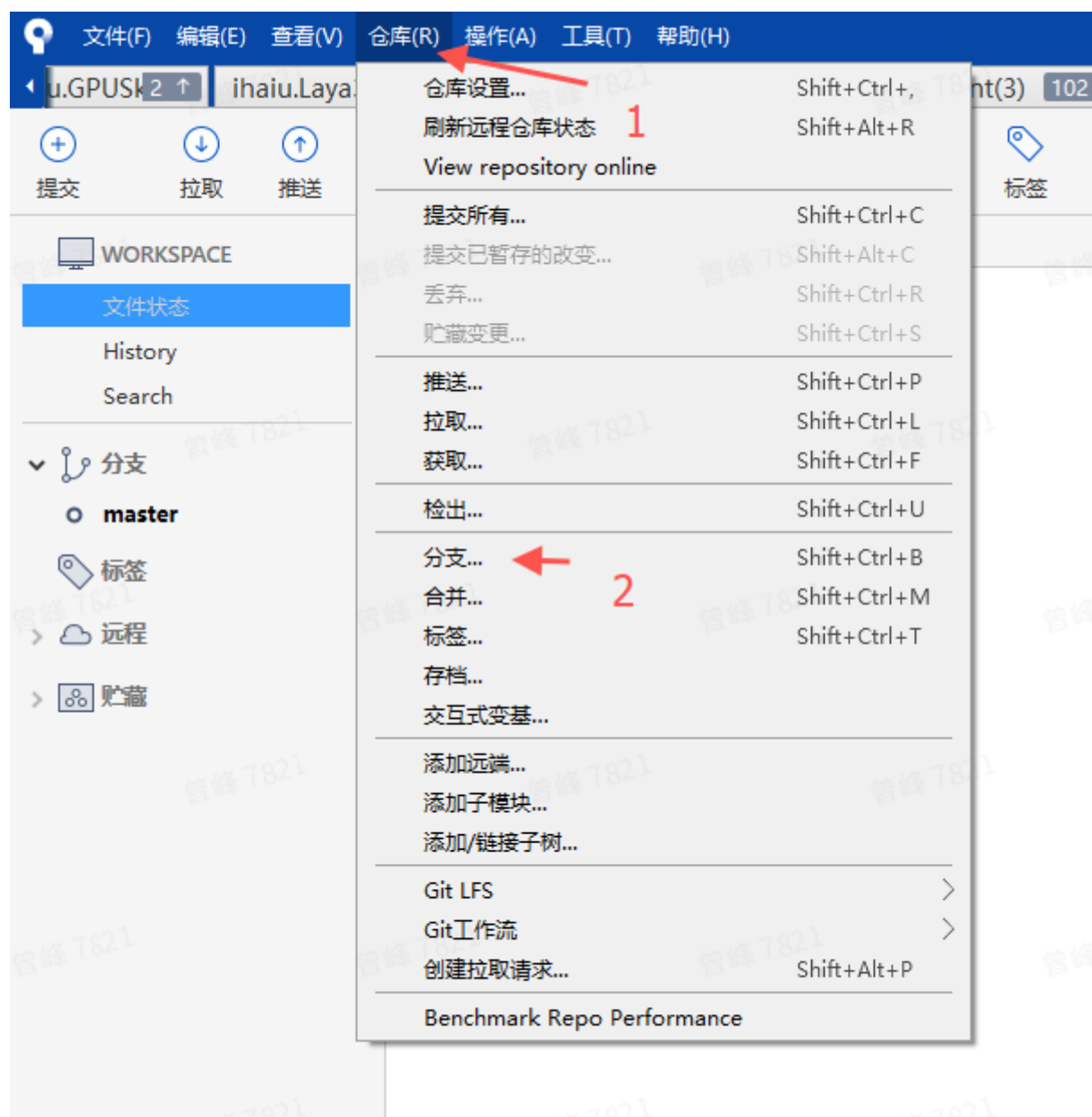
4

## (2) 添加本地已经克隆过的仓库



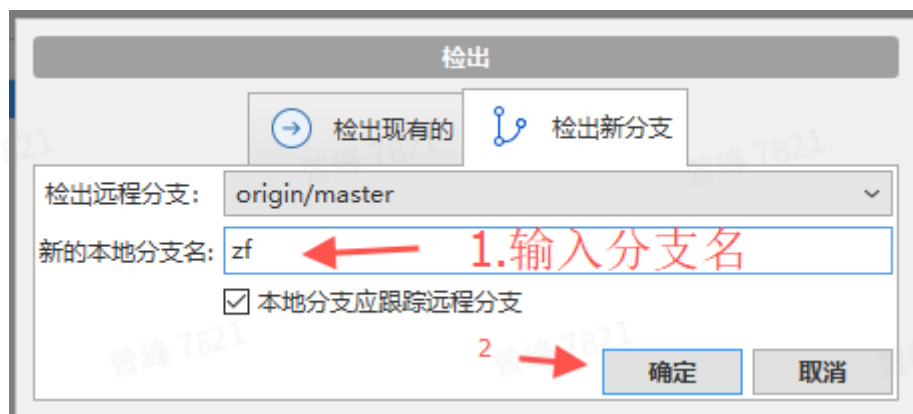


### (3) 创建分支--从当前本地分支



#### (4) 创建分支--从远程分支





## (5) 提交文件

提交 拉取 推送 获取 分支 合并 贮藏 丢弃 标签

1. 选中到文件状态，有的叫工作副本

WORKSPACE

文件状态

History

Search

分支

login

master

标签

远程

origin

贮藏

已暂存文件

取消所有暂存 取消选定暂存

a1.txt

文件内容
a

未暂存文件

暂存所有 暂存所选

b.txt (黄色笔图标, 表示修改了该文件)

c.txt (灰色减号图标, 表示删除文件)

a1.txt (蓝色问号图标, 表示新增文件)

立即推送变更到 origin/login

2. 该区域是即将要提交到分支里的文件

这里按钮操作将文件在这两个区域切换

1. 将这区域里的文件需要保持到分支的选择暂存

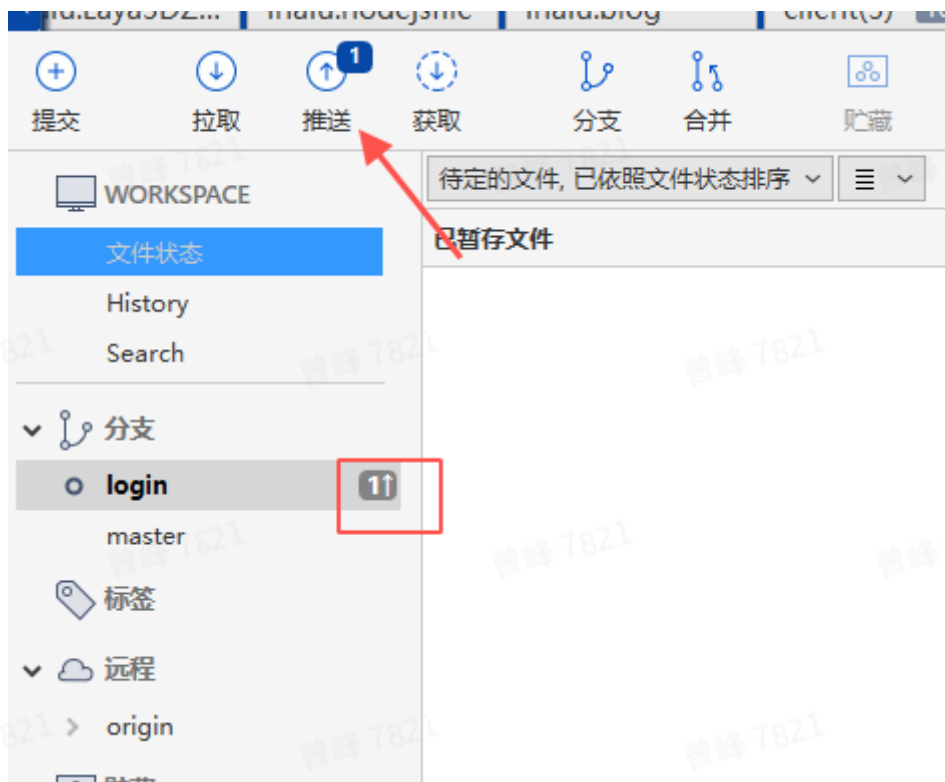
3. 添加提交备注, 必须要填要养成良好习惯

4. 将文件提交到本地分支仓库

提交

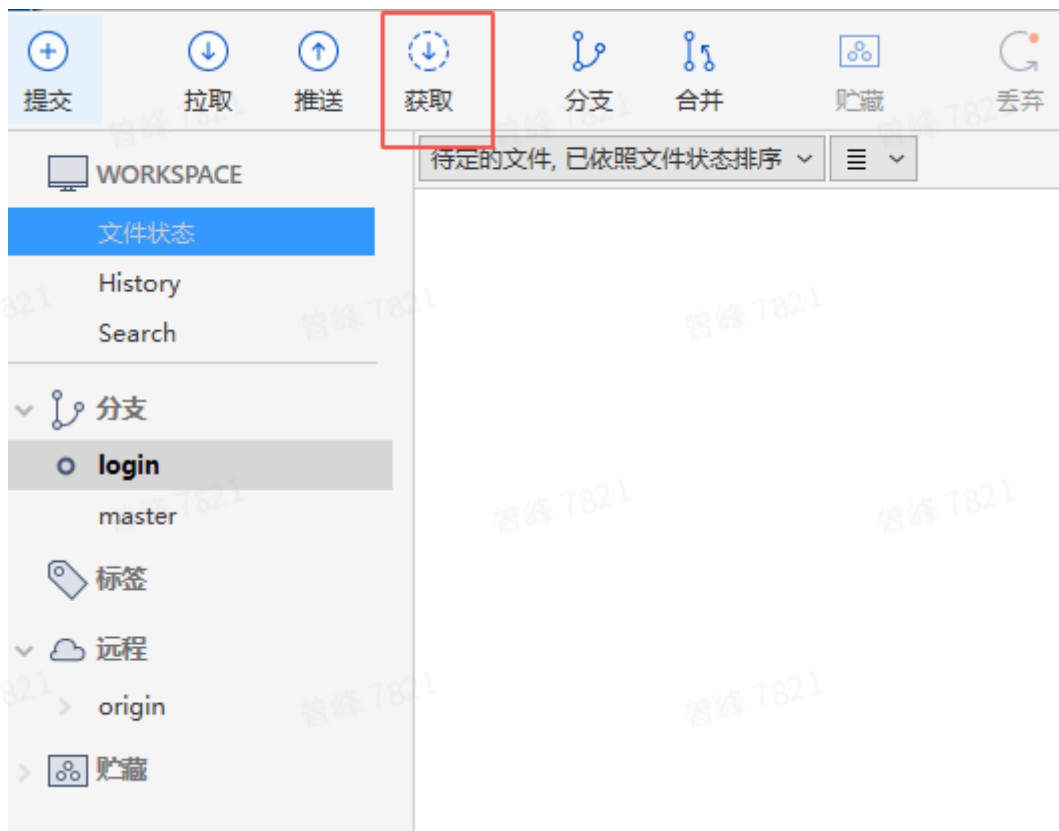
## (5) 推送分支

将本地分支，推送到远程仓库分支



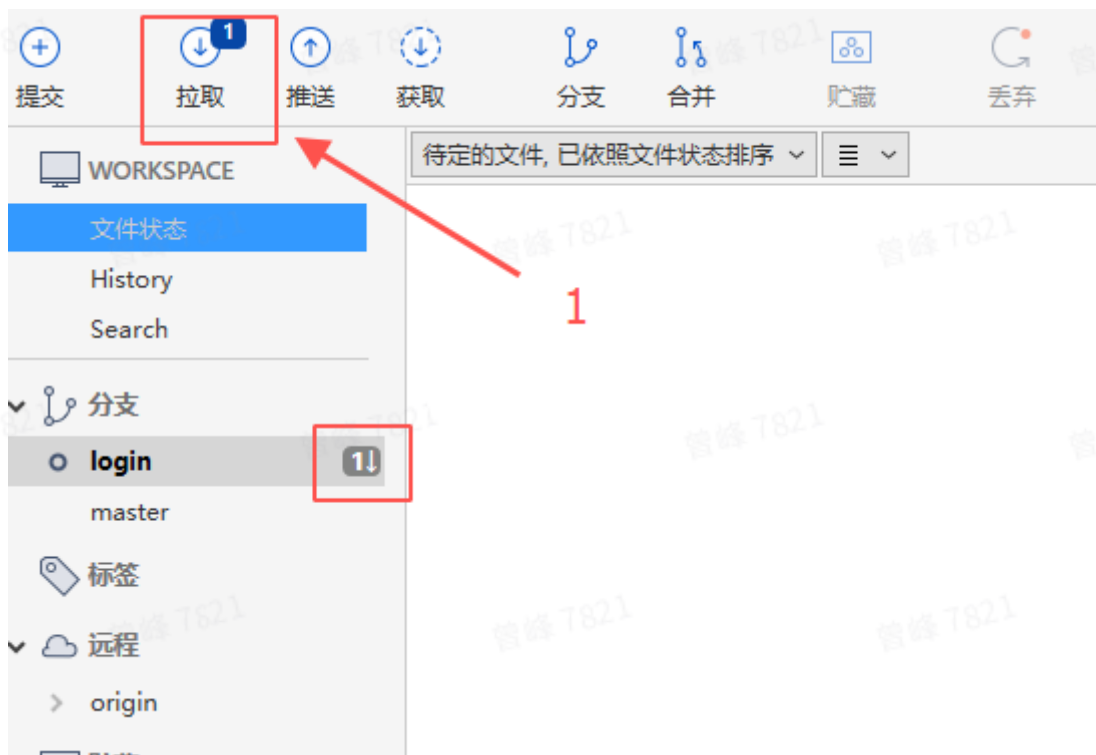
## (6) 获取远程仓库信息

可以获取其他开发人员提交到远程仓库的分支信息。不影响当前本地分支。



## (7) 拉取分支

将远程分支修改同步到本地。



## (8) 查看提交日志

The screenshot shows the Git GUI interface with the following annotations:

- 1. 选择分支 (Select branch): Points to the 'login' branch in the left sidebar.
- 2. 选择过滤条件 (Select filter): Points to the '所有分支' (All branches) dropdown in the top toolbar.
- 3. 显示远程分支 (Show remote branches): Points to the '显示远程分支' checkbox in the top toolbar.
- 4. 选择提交日志 (Select commit log): Points to the commit list table.
- 5. 这里就是步骤4选中提交的文件详情 (Here is the file detail of the commit selected in step 4): Points to the file details pane on the right.

提交	日期	作者	提交
4321d75	2020-09-14 22:10	ihaiu <ihaiu@qq.com>	4321d75
f380949	2020-09-14 21:51	ihaiu <ihaiu@qq.com>	f380949
2e78668	2020-09-14 20:35	管峰 <fengzeng@123u.com>	2e78668

提交: 4321d75  
父级: f380949  
作者: ihaiu <ihaiu@qq.com>  
日期: 2020年9月14日 22:10:46  
提交者: ihaiu  
fix: 修改文件测试

文件: b.txt, c.txt

修改对比 (Diff):

```
块 1: 行 0-0
- a
\ No newline at end of file
+ bbb
\ No newline at end of file
```

一般我们选择本地当前分支查看。

The screenshot shows the top toolbar of the Git GUI with the following elements:

- 当前分支 (Current branch): A dropdown menu.
- 显示远程分支 (Show remote branches): A checkbox.
- 层级顺序 (Hierarchy order): A dropdown menu.
- 图谱 (Graph): A button.

## (9) 查看文件历史记录

The screenshot shows the Git GUI interface with the following annotations:

- 1. 这里输入文件名搜索要查看的文件 (Enter filename here to search for the file to view): Points to the search input field.
- 2. 这里将会出现搜索的文件列表 (The search results will appear here): Points to the search results table.
- 3. 选择要查看的文件右键 (Right-click the file to view): Points to the context menu.
- 4. 选择查看变更历史 (Select View History): Points to the '选定项目的变更历史...' option in the context menu.

搜索: a

文件内容
a

未暂存文件 (Unstaged files):

- a.txt
- README

上下文菜单 (Context menu):

- 打开 (Open) - Shift+Ctrl+O
- 在资源管理器中打开 (Open in Explorer)
- 复制路径到剪贴板 (Copy path to clipboard)
- 外部差异对比 (External diff) - Ctrl+D
- 添加 (Add) - Ctrl+Shift+Plus
- 跟踪Git LFS的文件类型 (Track Git LFS file type)
- 删除 (Delete) - Ctrl+Del
- 丢弃 (Discard) - Shift+Ctrl+R
- 忽略... (Ignore...)
- 停止跟踪 (Stop tracking)
- 提交... (Commit...) - Shift+Alt+C
- 解决冲突 (Resolve conflict)
- 自定义操作 (Custom operation)
- 选定项目的变更历史... (View history of selected items) - Shift+Alt+L
- 按行审阅选定项目... (Review selected items line by line) - Shift+Alt+B

日志: a.txt

描述	日期	作者	提交
login no message	2020/9/15	ihaiu <ihaiu@...>	597d6ed
测试	2020/9/14	ihaiu <ihaiu@...>	f380949

1. 可以在这里看到提交日志，可以查看提交时间、谁提交的。右边可以看做了什么修改

这里可以看更消息的提交信息。提交时间等

提交: 597d6ed530371ff2b01063e9d70d37f66cce89e1 [597d6ed]  
父级: 4321d75ea1  
作者: ihaiu <ihaiu@qq.com>  
日期: 2020年9月15日 9:55:01  
提交者: ihaiu

no message

如果文件改过名勾选这个可以出现更多日志

☐ 跟踪已重命名文件

a.txt

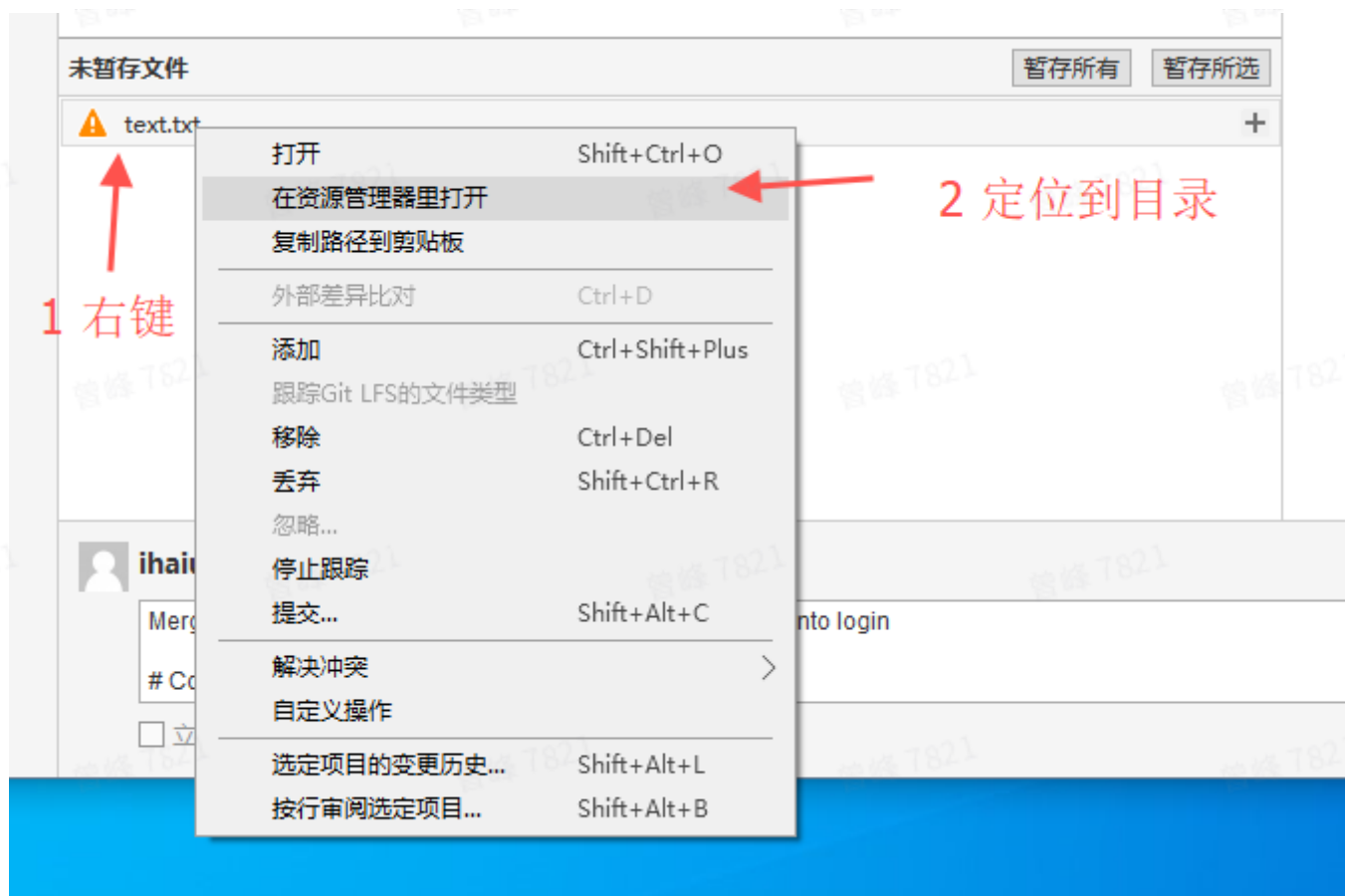
块 1: 行 1-2

```
- a  
\ No newline at end of file  
+ a1  
+ a2  
\ No newline at end of file
```

回滚区块

关闭

## (10) 查看文件所在目录



## (11) 文件冲突

**冲突发生的原因：**你和其他成员在这段时间有对同路径名称的文件都有操作。

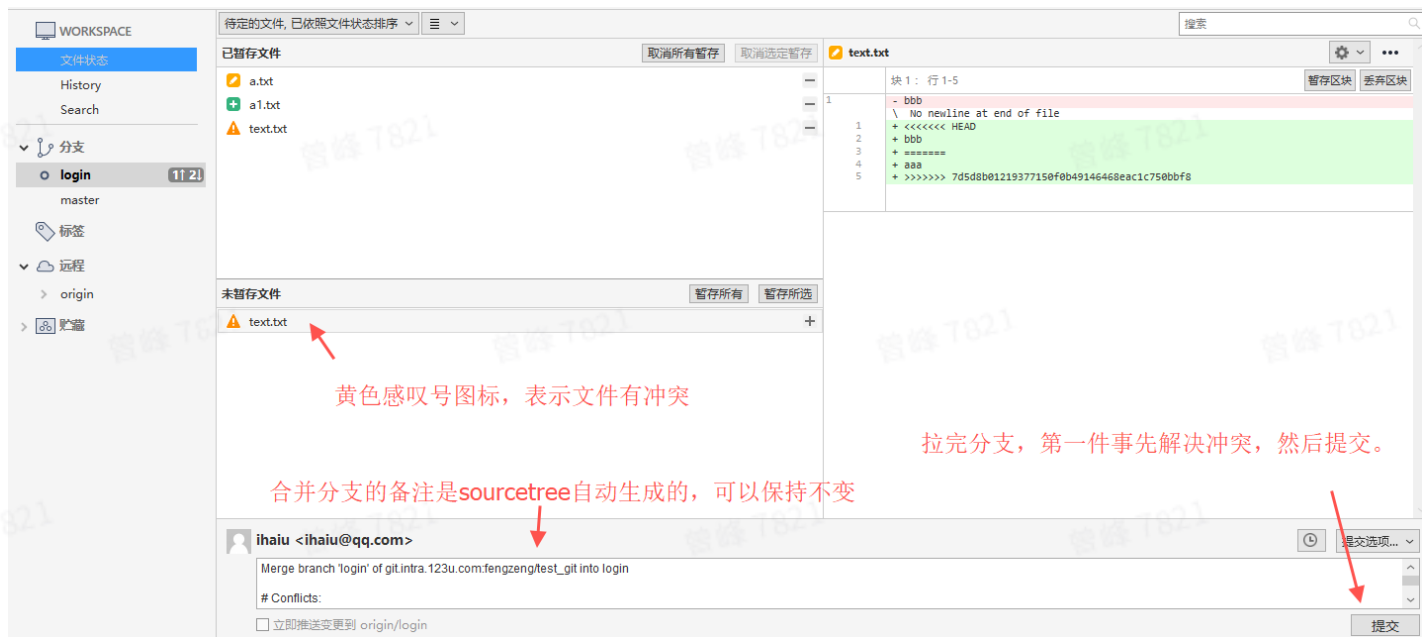
**解决的目的：**让这文件满足你们双方需求

**解决方法：**

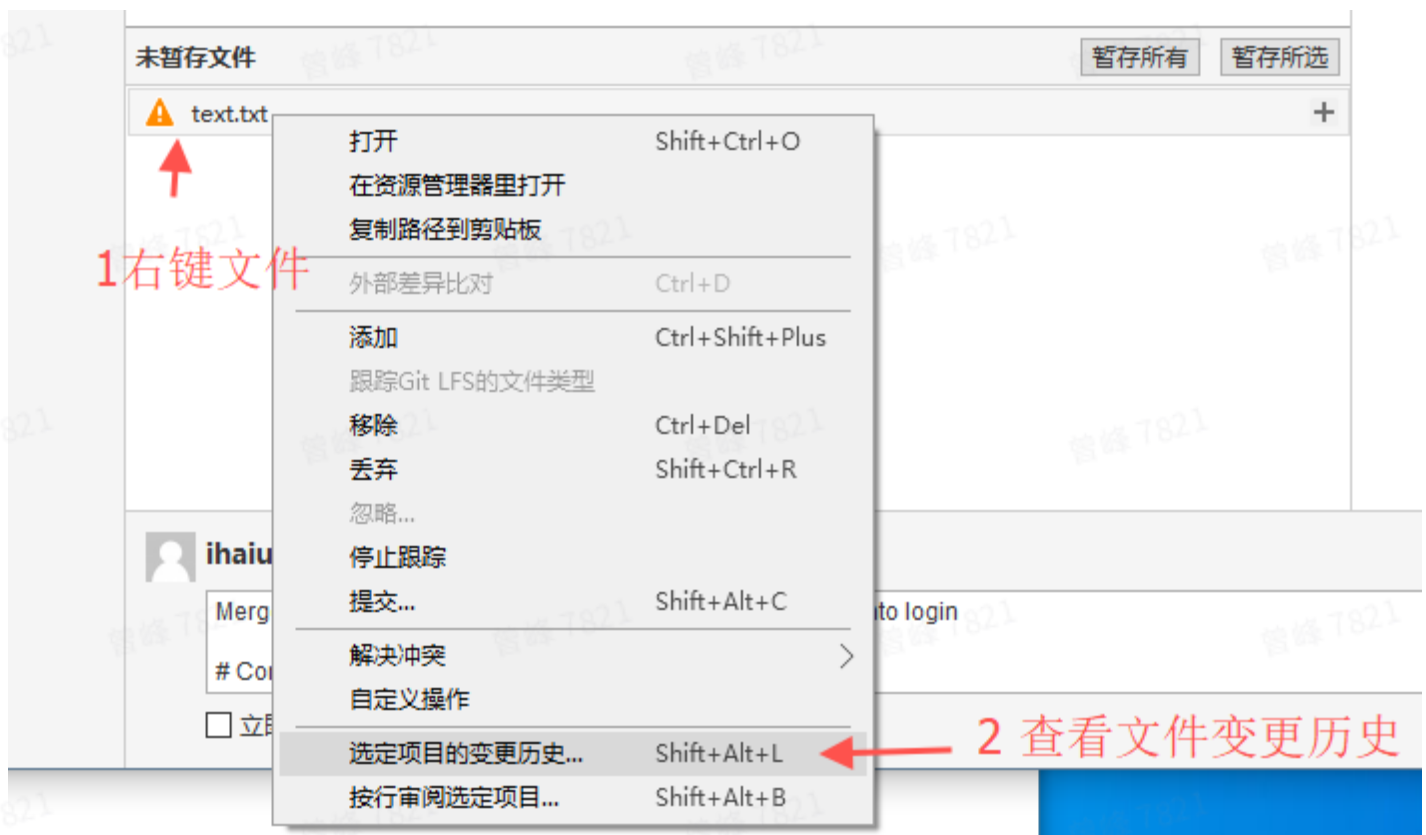
- 1.找最后一次提交该文件的同事询问。
- 2.你们双方协商使用谁的，或者怎么修改。

### 解决冲突流程

拉取分支，如果有冲突，将会出现黄色感叹号图标文件。



查看该文件提交日志, 找到最后一次提交该文件的作者, 协商解决冲突







如果你们协商好了。一般有以下几种方式解决：

### 方式一：选择使用某个人的



### 方式二：先修改文件，然后标记为已解决

文本文件可以直接双击打开，其他文件需要定位到文件目录位置替换该文件。

文本文件：

```
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<<<<<< HEAD
bbb
=====
aaa
>>>>>> 7d5d8b01219377150f0b49146468eac1c750bbf8
```

冲突区块开始标记

2个提交分割线

冲突区块结束标记

```
<<<<<< HEAD
bbb
=====
aaa
>>>>>> 7d5d8b01219377150f0b49146468eac1c750bbf8
```

HEAD表示自己分支内容

其他分支内容

根据情况修改文本。然后标记已修改。

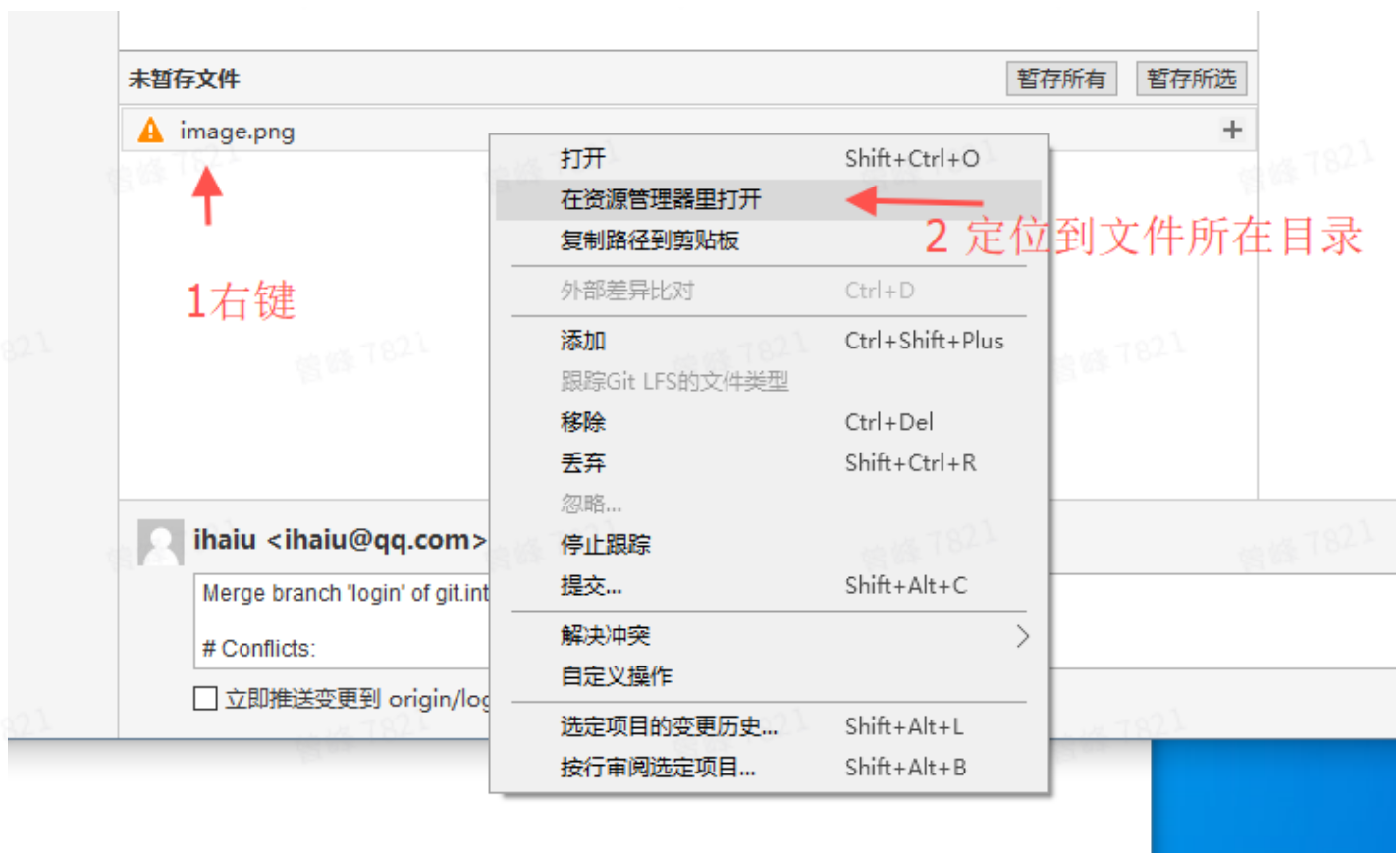
```
*text.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

bbb
```

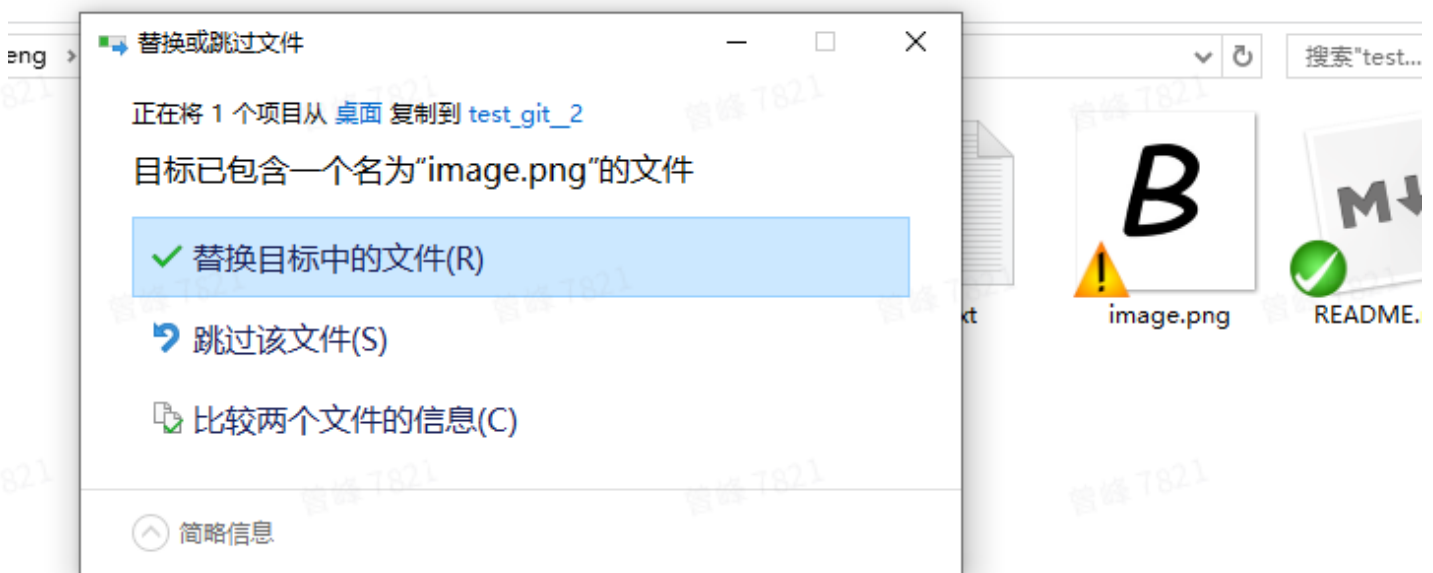


二进制文件，比如图片

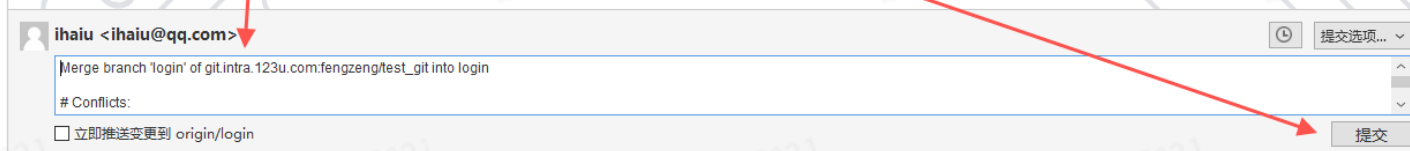
先定位到图片所在目录



将自己在其他目录对应的文件复制粘贴，覆盖该文件。然后标记为已解决。



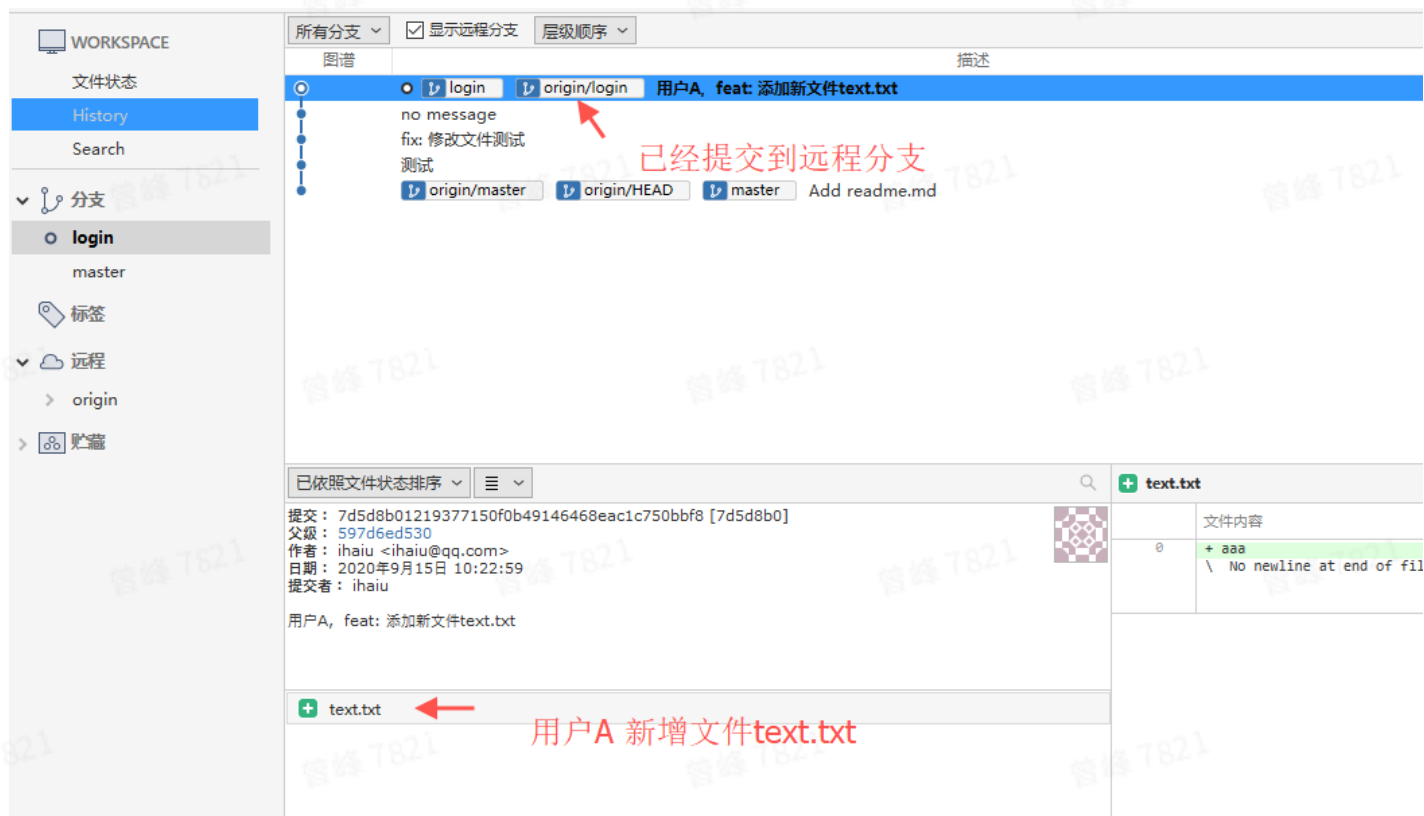
再次强调：合并分支，解决完冲突。一定要先提交，再去做其他事

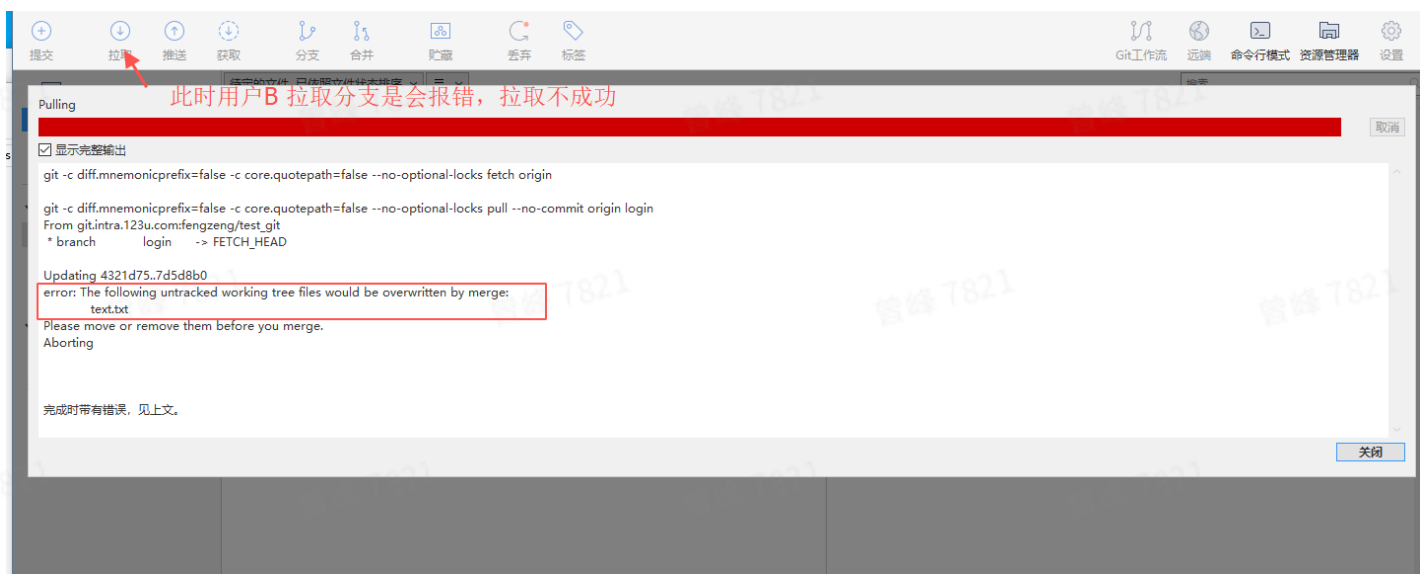


## 冲突情况一：用户同时新增相同路径名称文件

用户A：新增文件，text.txt，提交到远程分支。

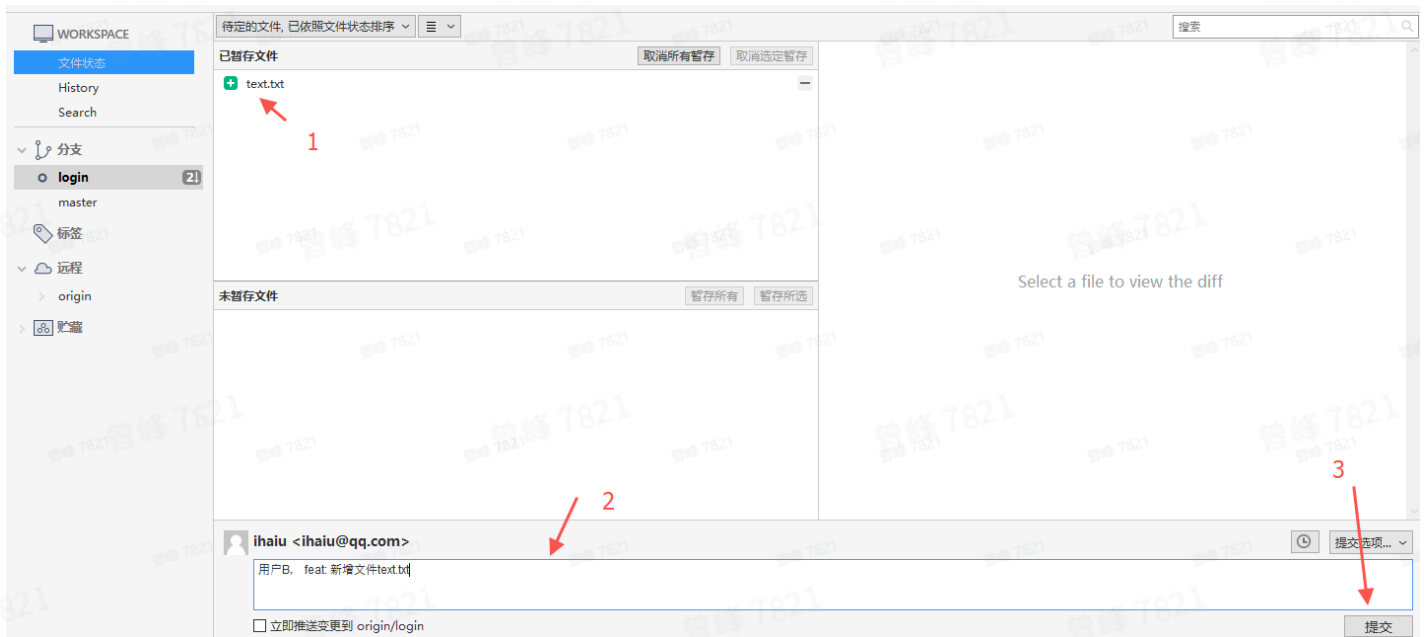
用户B：新增文件，text.txt，拉取远程分支。



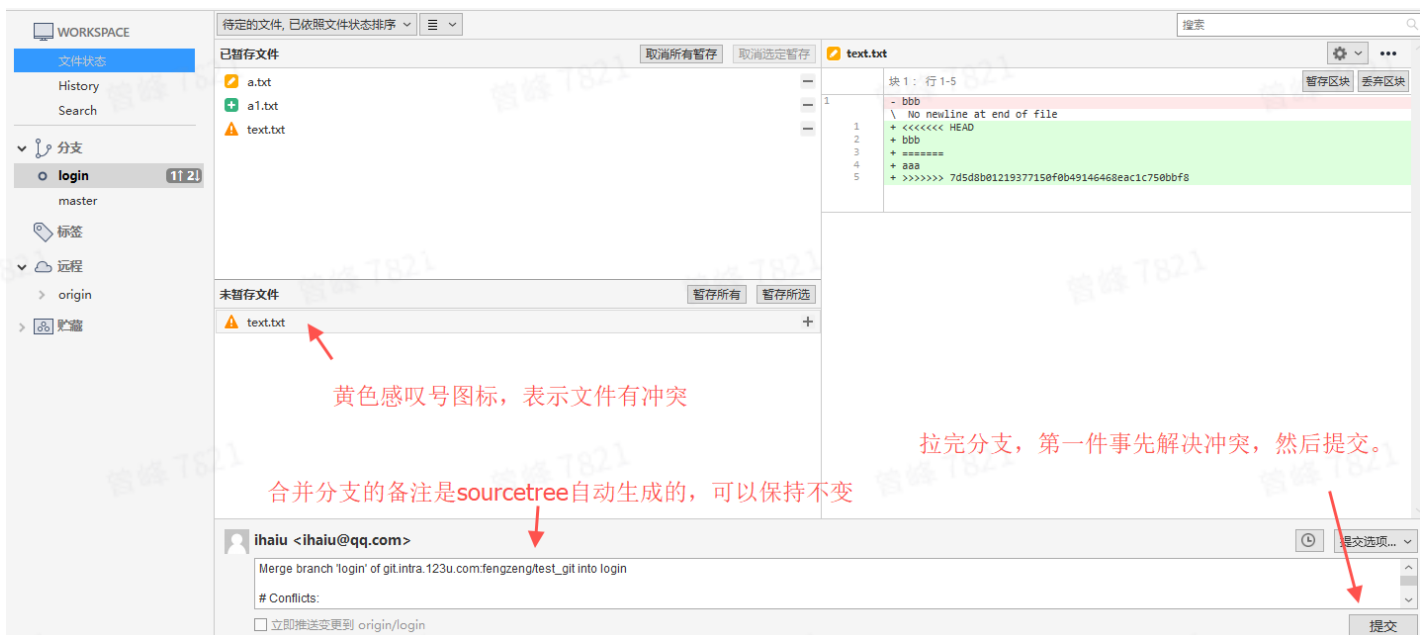


解决方法：

用户B, 先将text.txt 提交本地分支



此时再拉取，可以成功把远程的拉到本地，但是将会出现冲突。



然后走”解决冲突流程“

## 冲突情况二：用户同时修改相同路径名称文件

避免方法：如果修改的不是自己文件，先和对方说一下，让对方先提交，你拉取完成再去修改。

## (12) 忽略文件

忽略文件只存放在本地文件夹中，不会进git仓库，更不会传到远程仓库。

原则：

1. 合并分支时，不准"忽略"和"停止跟踪"别人的文件。
2. 只能忽略自己文件，并且是确保分支中没有依赖该文件。

### (13) 停止跟踪文件

停止跟踪是将仓库中已经存在的文件删除。

原则：

1. 合并分支时，不准"忽略"和"停止跟踪"别人的文件。
2. 不允许美术使用

### (14) 丢弃文件

丢弃文件是将还未暂存的文件，放弃操作（修改、删除）。

原则：

1. 只允许对 黄色笔图标 修改的文件使用

### (14) 移除文件

移除文件是删除本地文件。如果是“黄色笔图标的修改文件”将会改变为“灰色减号图标的删除文件”，如果将该文件继续提交将会删除仓库文件，再推送将会删除远程仓库文件。

原则：

1. 不允许乱移除其他人的文件。不是自己的文件需要和对方协商。
2. 美术只能移除问号图标新增文件

## 三：Git 使用规范

1. 拉取完分支：先解决冲突 -> 提交 -> 再去做自己的事。否则自己的修改将会和合并分支日志混杂在一起，会很混乱。



2. 如果遇到自己不会解决的冲突，先查看日志，找到最后提交该文件的人协商。双方都不会解决找程序去帮忙。
3. 如果要动别人的文件，一定要事先和对方协商。否则不准动。
4. 避免冲突的方法：不要去动别人的文件。如果要动，先叫对方提交->你拉取->然后再修改
5. 不要乱忽略文件，只有确保是自己的文件，且提交的文件中没有依赖该文件才能添加忽略。
6. 不允许"忽略"、"停止跟踪"、“移除”别人文件，否则将会使别人的提交丢失。

## git commit 提交描述规范

<https://www.jianshu.com/p/b584152d2a16>

简单规范：[提交人] type: <描述>

type 是提交的类型，一般常用的如下：

- feat：新功能
- fix：修正
- refactor：重构
- chore：维护
- style：修正代码格式，无代码功能上的变化
- docs：文档的变更
- test：测试