Git

## 1.1 Git认识：

Git就是一本版本控制工具，学习其固定的操作指令即可。版本管理分为个人版本管理 和 团队版本管理。Git不仅可以管理文件本身，关键他可以管理文件中的内容，知道什么内容被修改操作等等。

版本控制工具应该具备的功能：

1. 协同修改

多人可以并行不悖的修改服务器端的同一个文件。

1. 数据备份  
   不仅可以保留目录和文件的当前状态，也可以保留每一个提交过的历史状态。
2. 权限控制  
   对团队中参与开发的人员进行权限控制。  
   对团队外开发者的贡献进行审核---git独有。
3. 历史记录  
   查看修改人、修改时间、修改内容、日志信息。  
   将本地文件恢复到某一个历史状态。
4. 分支管理  
   允许开发团队在工作过程中采用多条生产线同时推进任务，进一步提高效率。

## 1.2 Git本地结构

3：本地库

2：暂存区

1：工作区

写代码 git add 临时存储 git commit 历史版本

## 1.3 Git和代码托管中心：

代码托管中心任务：维护远程库

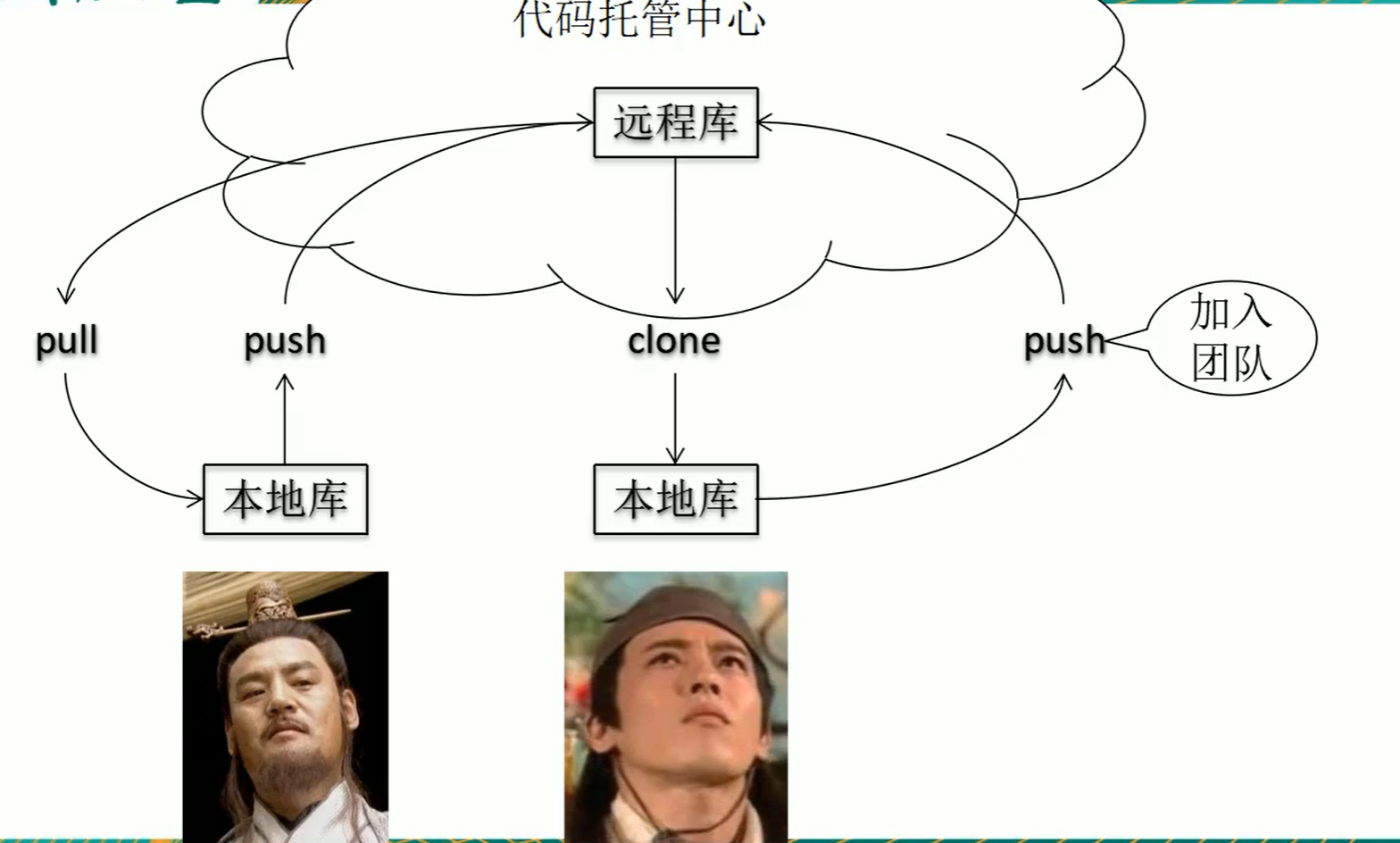
Git和github是完全两个不同的东西，github就是git的一个代托管中心。

局域网环境：GitLab服务器

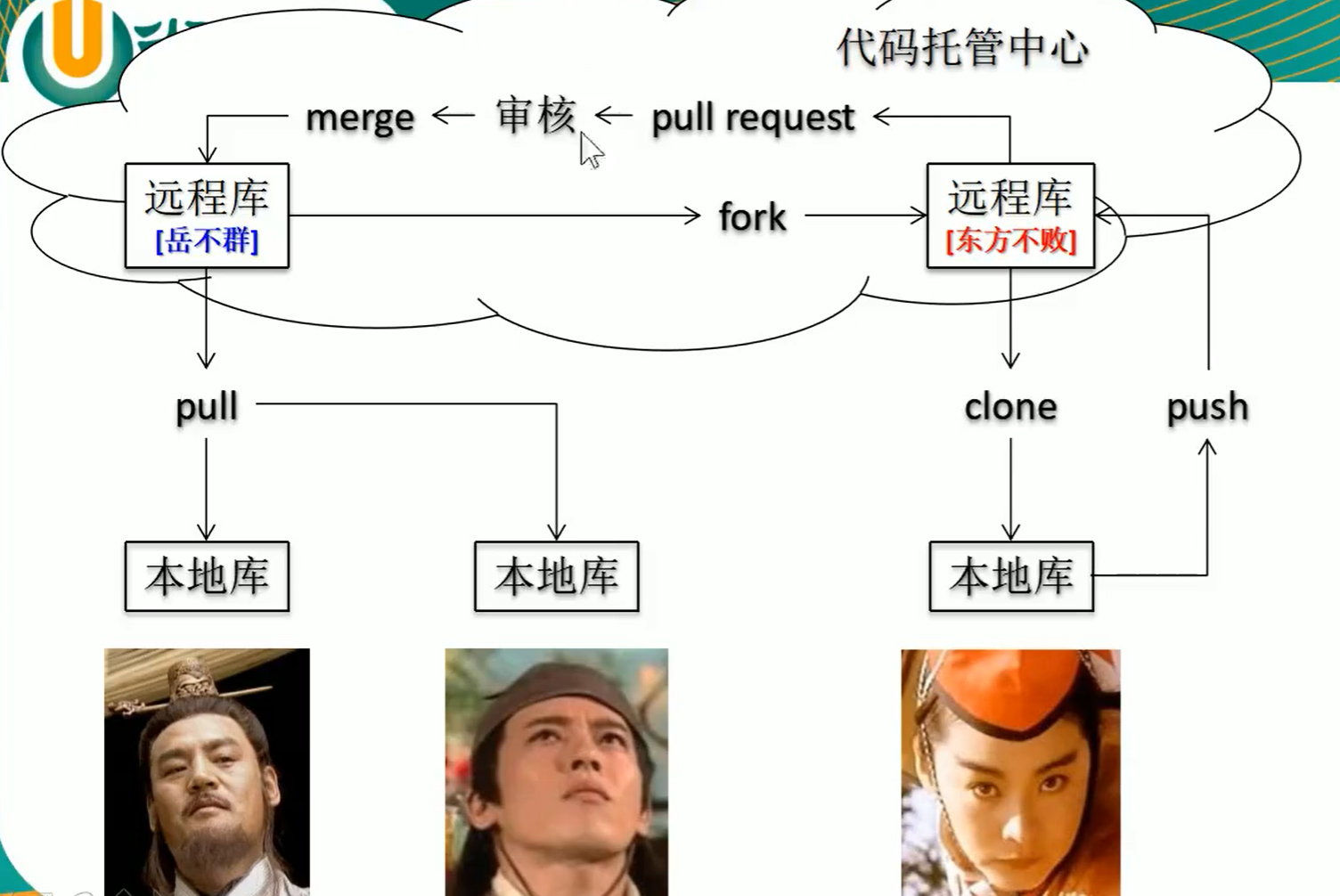
外网环境： Github 和 码云（国内快）

## 1.4本地库和远程库交互方式

1：团队内部协作



2：跨团队之间协作



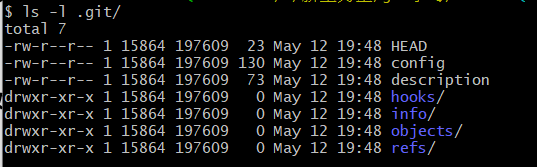
## 1.5 Git命令操作

### 1.5.1本地库初始化：

Linux中 以 . 开头的目录代表隐藏资源 ls – lA查看隐藏目录 ls 显示大致信息 ，ls –l 等价于 ll 显示详细信息。

命令：git init

效果：生成一个 .git配置文件



注意：.git目录中存放的是本地库相关的目录和文件，不要删除 也不要胡乱修改。

1. 设置签名
2. 形式：

用户名：tom

Email地址：[15864626291@163.com](mailto:15864626291@163.com)

1. 作用：区别不同开发人员的身份
2. 辨析：这里设置的签名和登录远程库（代码托管中心）的账号、密码无任何关系。
3. 命令：

项目级别/仓库级别：仅在当前本地库范围内有效

git config user.name tom\_pro  
git config user.email [15864626291@163.com](mailto:15864626291@163.com)

设置完毕后 就在git中的config下可以看到 cat .git/config

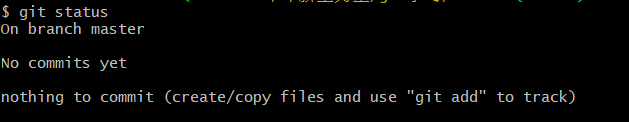
系统用户级别：登录当前操系统的用户范围

git config --global user.name tom  
git config --global user.name [15864626291@163.com](mailto:15864626291@163.com)

1. 先级：

就近原则：项目级别优于用户级别，二者都有采用项目级别的签名。

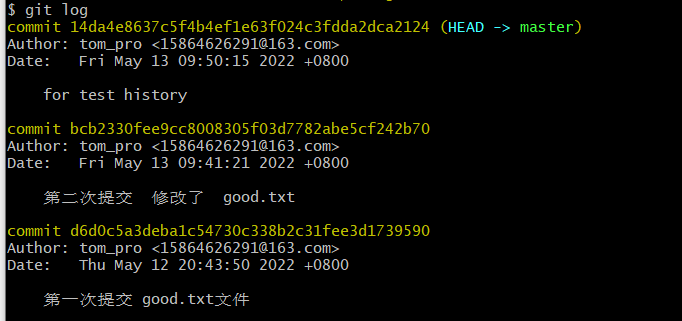
1. 查看状态 git status



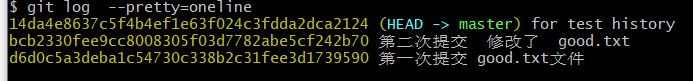
三行分别代表了 主分支 第二行表示没有任何提交 （代表本地库什么也没有）

第三行 没什么可提交（达标暂存区也没东西）

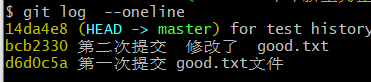
### 1.5.2基本操作

1. 状态查看操作   
   git status 用来查看工作区、暂存区的状态
2. 添加操作  
   git add [file name] 将工作区的新建/修改添加到暂存区
3. 提交操作  
   git commit –m “commit message” [file name]
4. 查看历史操作  
   git log  
   

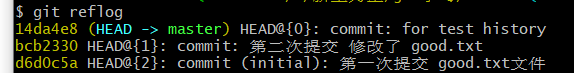
git log –pretty=oneline(简洁版)



git log oneline （哈希值显示的少了）



git reflog (HEAD{num}其中 num代表了从当前版本退回到相应的版本 指针HEAD移动的步数)

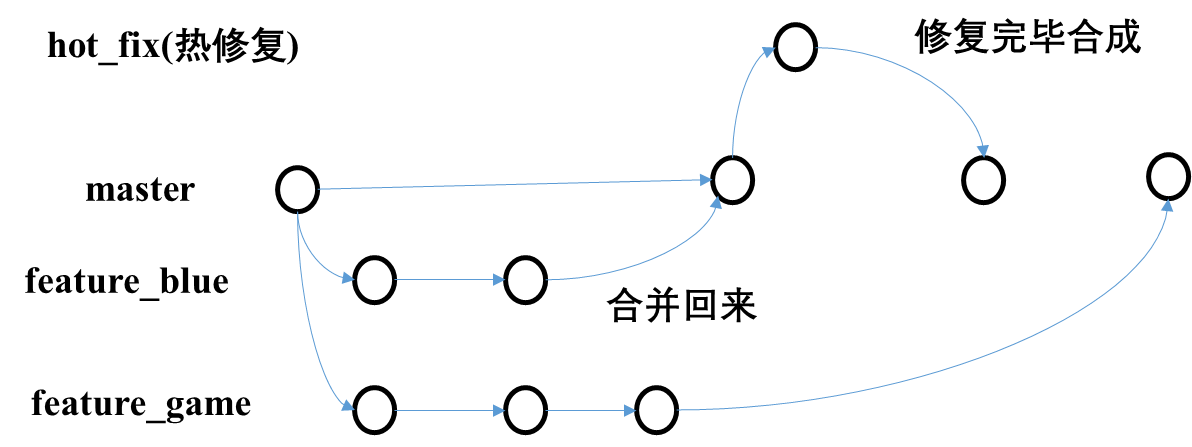


1. 版本前进后退操作  
   操作指针（HEAD）的一个过程就是。  
   1：基于索引值的操作（推荐）  
    后退：git reset –hard [索引值]  
    前进：git reset –hard[索引值]  
   --soft：只动本地库  
     
   --mixed：动本地库和暂存区  
     
   --hard：全变  
     
   2：使用^符号   
    只能后退  
   3：使用~符号
2. 永久删除文件后找回  
   没太学会这一部分

### **1.5.3分支管理**

1：什么是分支？

在本部控制的过程中，使用多条线同时推进多个任务。版本库初始化之后本身就是master这个分支。叫master分支或者主干分支。



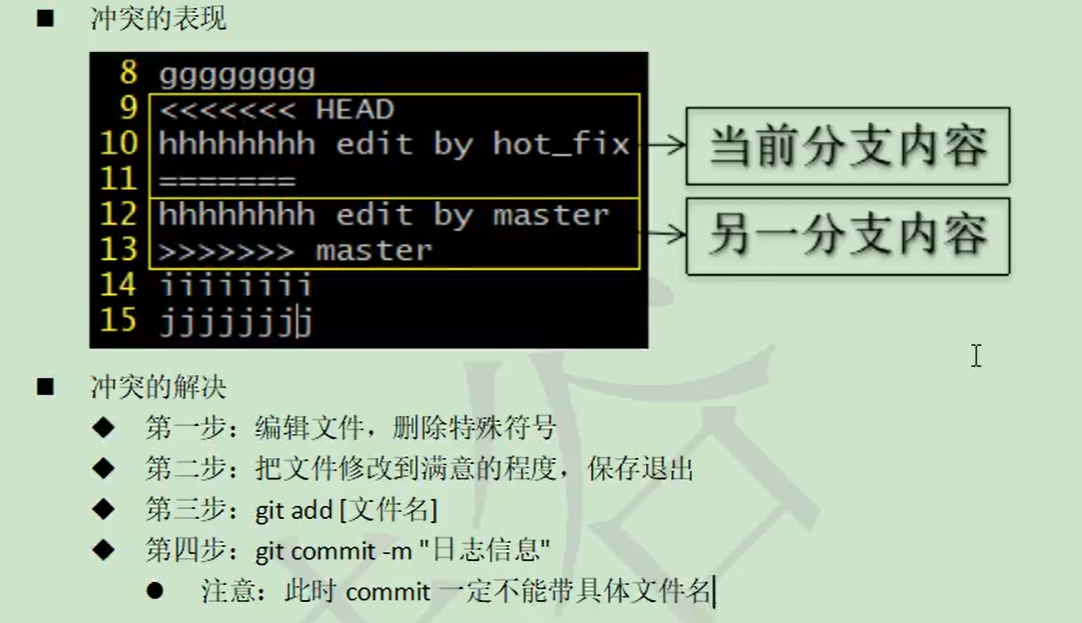
2：分支的好处？

同时推进多个功能的开发，提高开发效率。

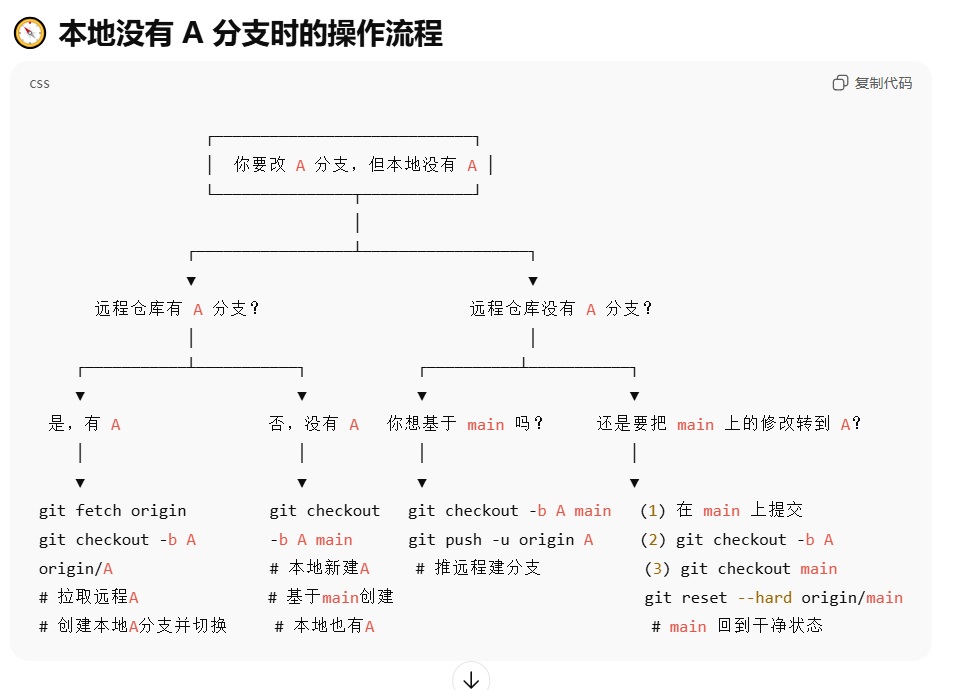
各个分支在开发的过程中，如果一个分支开发失败，不会对其他分支有任何影响。失败的分支删除重新开始即可。

3：分支操作命令

1. 查看当前所有的分支： git branch –v
2. 创建分支 git branch +[分支名字]
3. 切换分支 git checkout +[分支名字]
4. 合并分支   
   第一步：切换接受修改的分支上（被合并，增加新内容）。  
   第二步：执行命令 git merge 分支1 分支2（要增加内容的分支）
5. 解决冲突  
   有两个分支，我们都可以去修改分支，如果修改了同一个文件的同一个部分，合并的时候就会产生冲突。



4. 本地没有A分支 想修改A分支的东西怎么办。



首先：git checkout -b git\_gjd master 。 建立了本地分支git\_gjd，但是这个分支仅仅是在本地，你在远程端还看不到，需要将这个分支推送到远程。

git push -u origin git\_gjd。



并且这里需要注意的是：你切换到那个分支，对应的本地文件夹就会显示对应分支的内容。

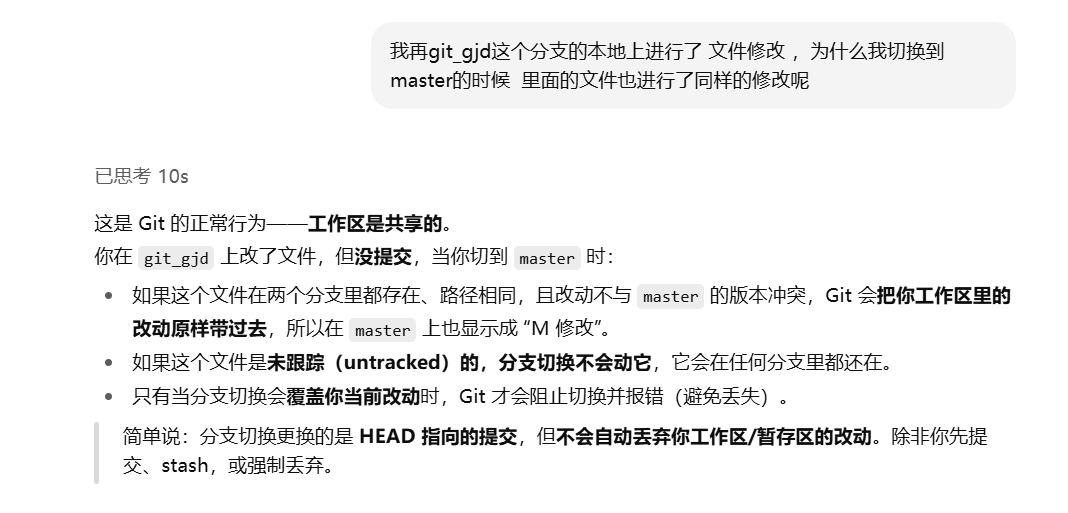
5.合并分支



#测试修改工程名字后 是否可以正常上传 合并 分支等。

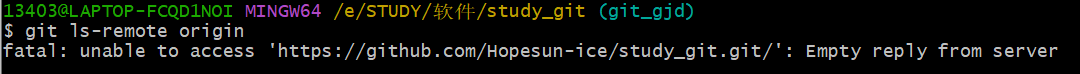
6.远程覆盖本地的做法



7. 我再git\_gjd这个分支的本地上进行了 文件修改 ，为什么我切换到 master的时候 里面的文件也进行了同样的修改呢 

## 1.6 Git常见错误

(1):无法访问github，我一看网页上github也上不去了，所以肯定也无法远程了。所以下次无法访问的时候，先确定远程的网页是否可以访问。



(2):显示什么端口啥的超时，根据gpt提示进行远程地址规范化问题得以解决。



##上课的还上课

##1111