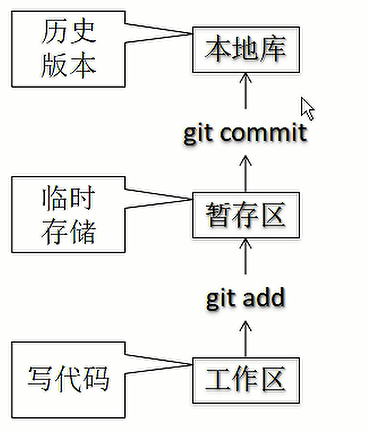
1. **git结构**



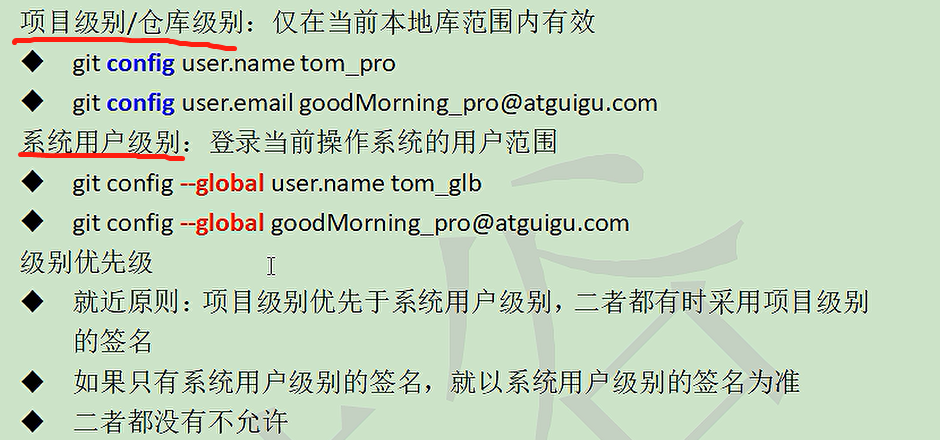
1. **git初始化**

git init 会创建.git文件夹

1. **设置签名**

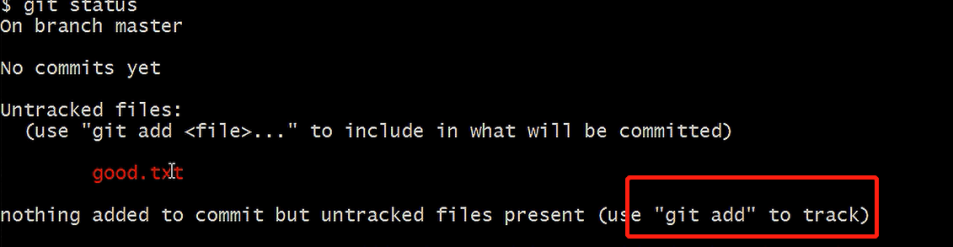
形式: 用户名：tom,EMAIL：aa@a.com作用：只是为了区分不同开发人员的身份，不会给你发邮件的。

级别：

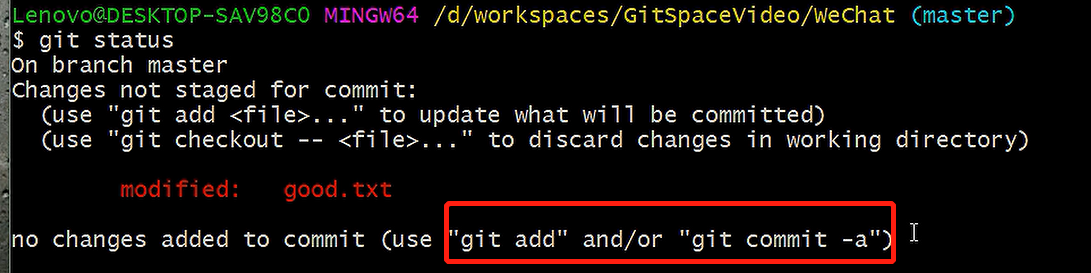


信息保存位置:项目级别保存在.git/config里，系统级别保存在~/.gitconfig里。一般来说只需要设置全局的就可以了。

1. **常规提交命令**
   1. **添加文件git add**

对要增加的文件进行追踪

* 1. **更新文件**



如果你已经追踪了，你可以git add添加到暂存区，也可以用git commit直接提交。但是如果直接提交的话，你就不能够撤销了。

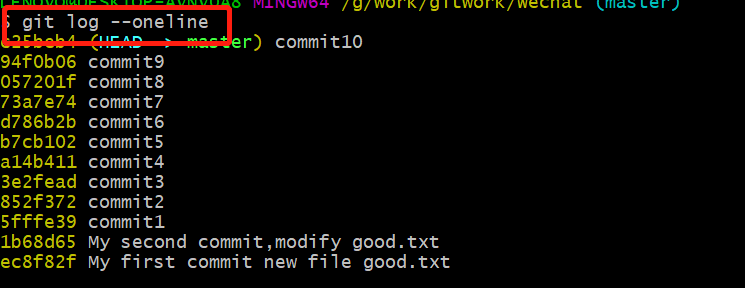
git commit -m ‘提交注释’ 文件名

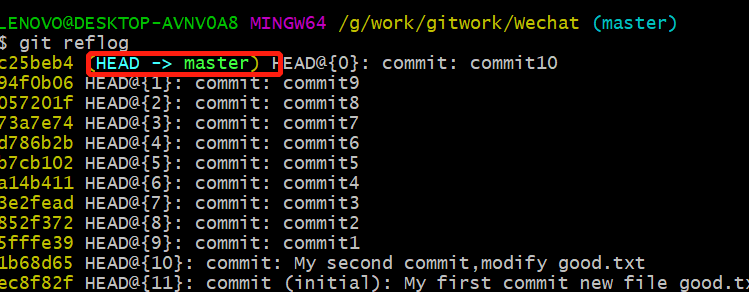
1. **版本穿梭,其实就是移动head的指针**
   1. **git log**



空格向下翻页，b向上翻页 q退出

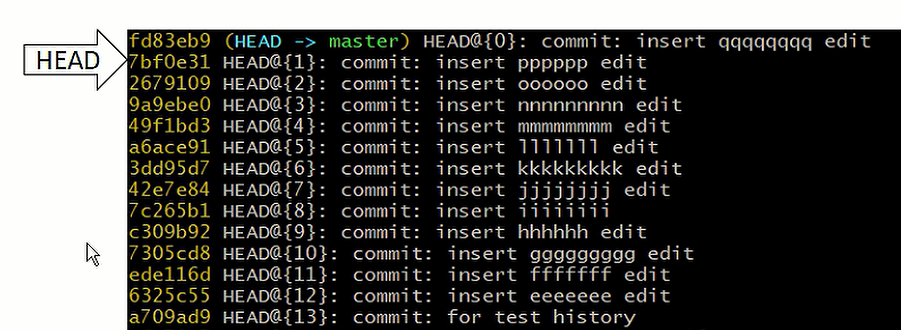






reflog显示的日志对于前进后退，移动指针是非常有参考价值的，head@{移动到当前版本需要的步数}。比如{2}就说明到这里你需要移动2步

* 1. **前进后退历史版本操作的本质**

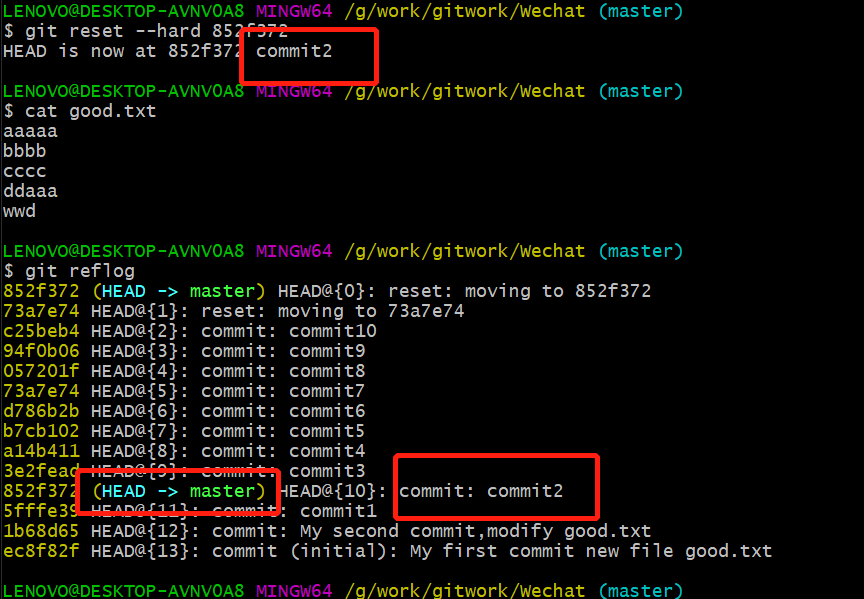


head就是这个指针的名字。这个指针为什么要给他起个名字？因为这样能方便我们去引用。

* 1. **版本前进后退**

有三种方式，一般只用基于索引值来操作。命令都是git reset --hard。这里的--hard说明本地库和暂存区都会被刷新（重置）。

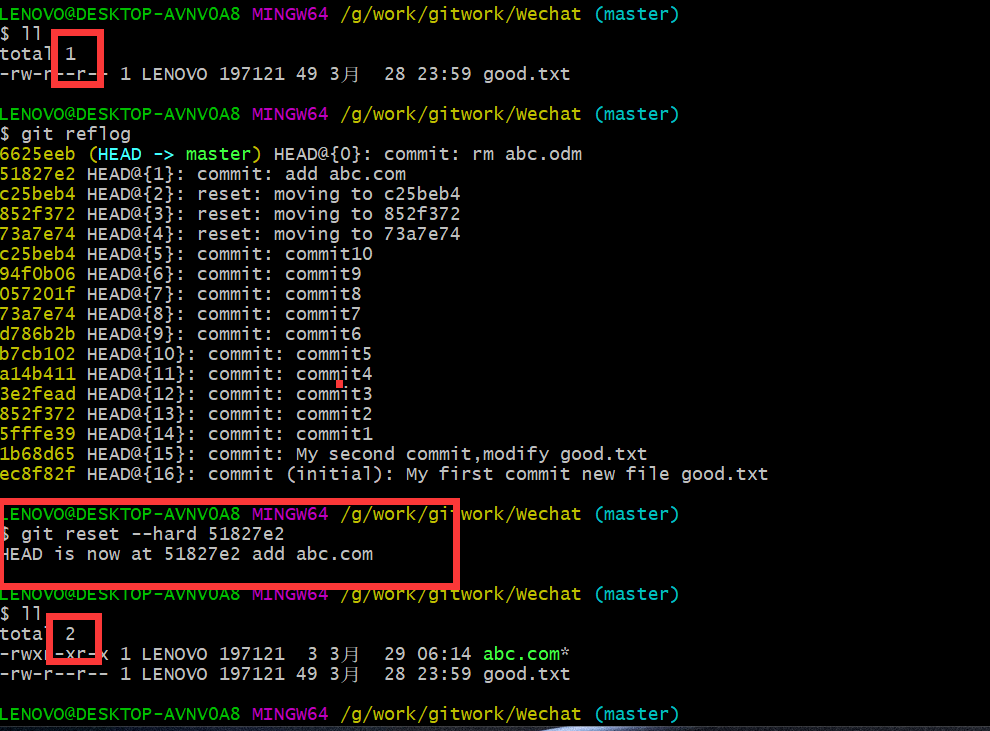




其他两种方式了解即可：使用^或~，只能后退，几个符号就是几步，如git reset --hard HEAD^^^ ==git reset --hard HEAD~3

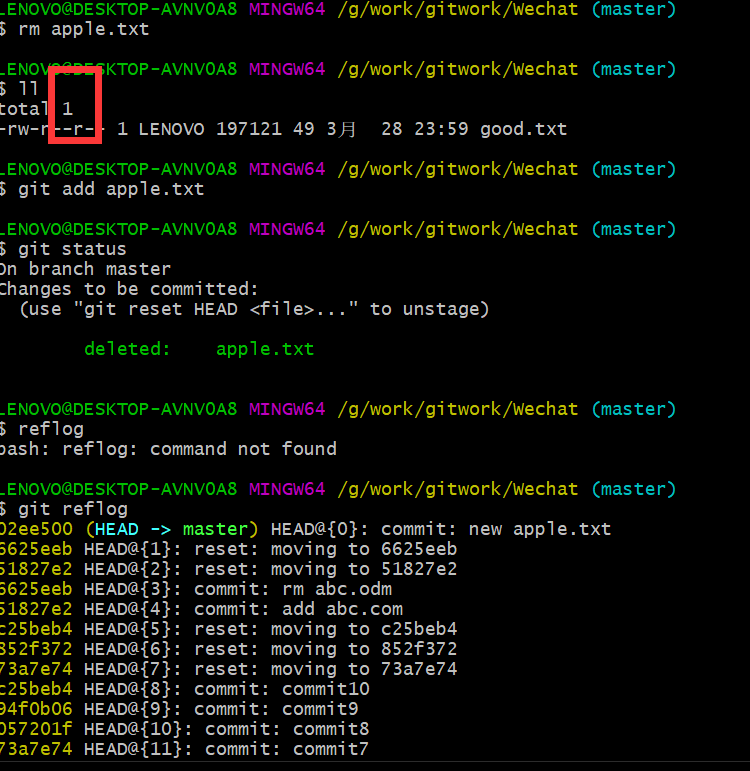
* 1. **删除文件后找回** 
     1. **永久删除文件后找回**

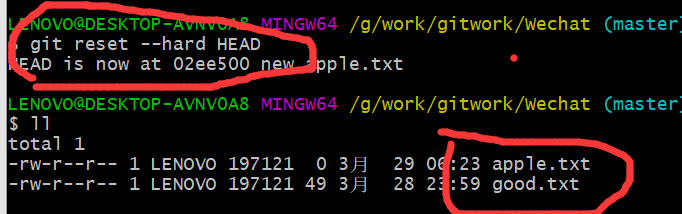
先要保证删除之前文件已经提交到本地库，然后删除文件，删除之后又提交到本地库了，那就直接把指针移动到还没删除之前那个版本就行了。



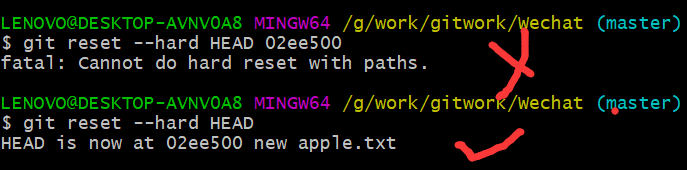
* + 1. **添加到暂存区的删除文件找回**

先要保证删除之前文件已经提交到本地库，然后删除文件，这时候把删除的文件添加到了暂存区，但还没有提交到本地库。



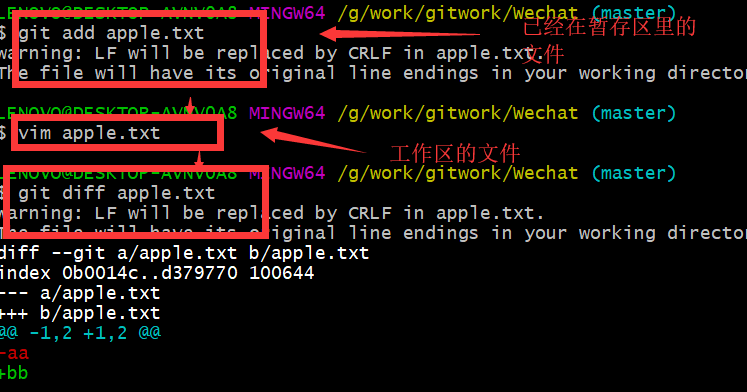


由于已经是最新的了，所以不能（注意是不能而不是不需要）指定版本号，只要打HEAD就可以了，否则会报下面错误



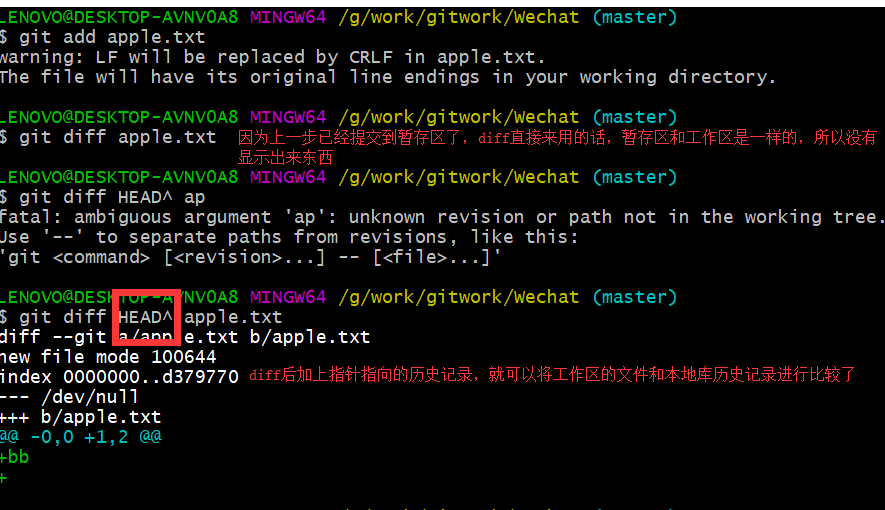
1. **比较文件**
   1. git diff [文件名]

将工作区的文件和暂存区的文件进行比较



* 1. git diff [本地库的历史版本][文件名]

将工作区中的文件和本地库历史记录进行比较



* 1. 不带文件名就会比较多个文件

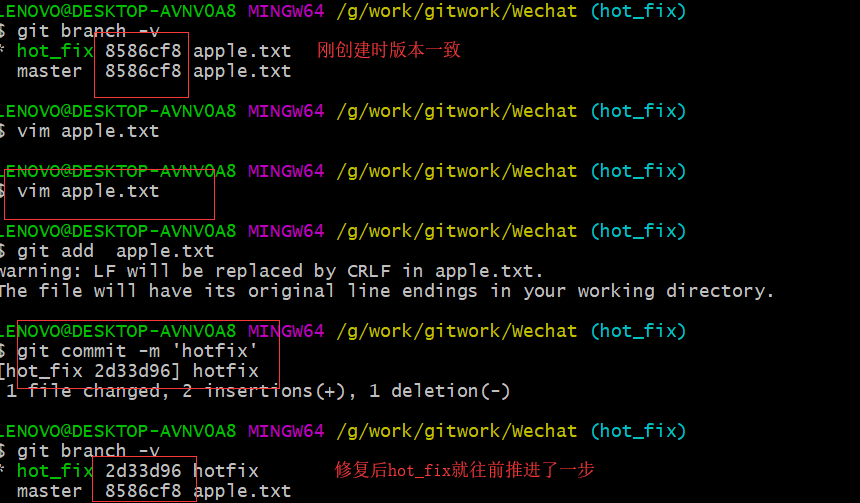
1. **分支**
   1. 初始话好以后就会有master分支，不需要手动创建



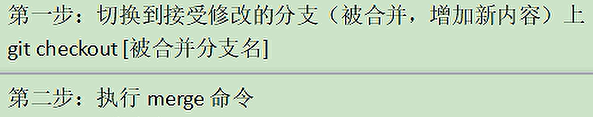
* 1. 创建,查看，切换分支



* 1. 合并分支（非常重要）
     1. 先在分支上修改并提交



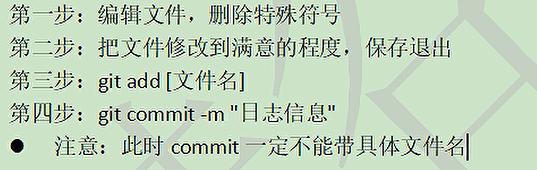
* + 1. 开始合并

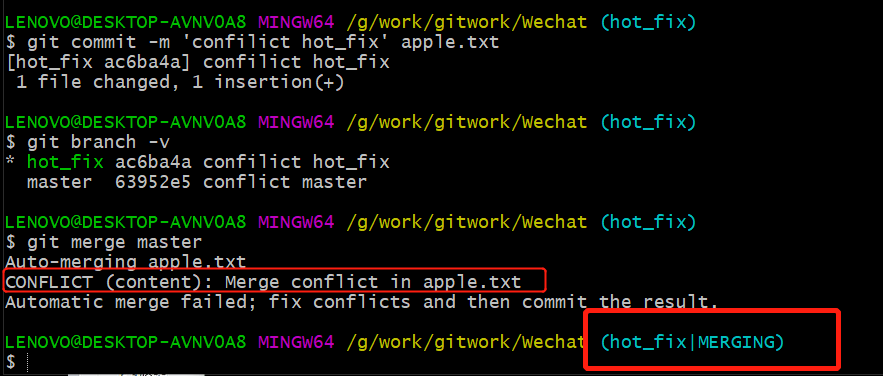


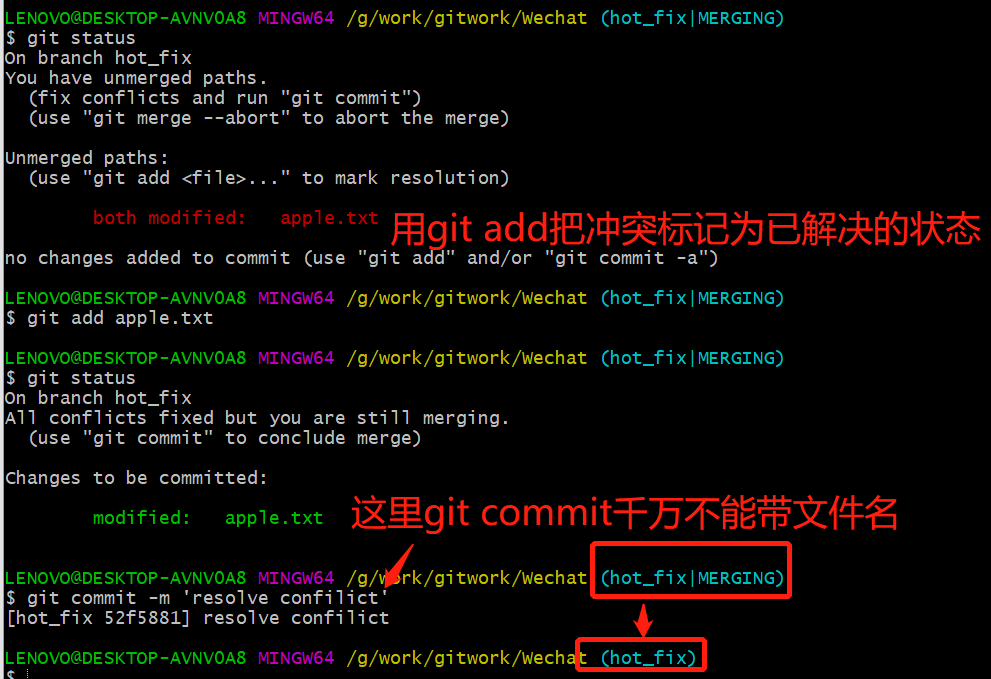


* + 1. 解决冲突

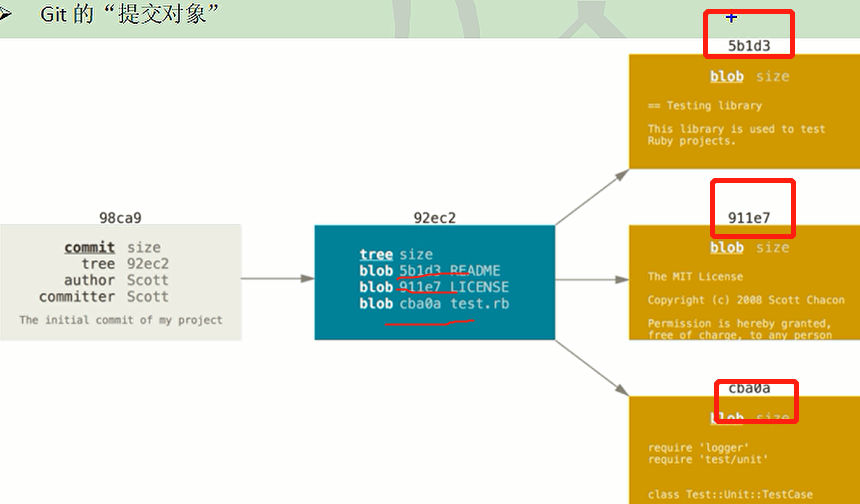
当两版本提交后文件发生冲突，自动合并会失败

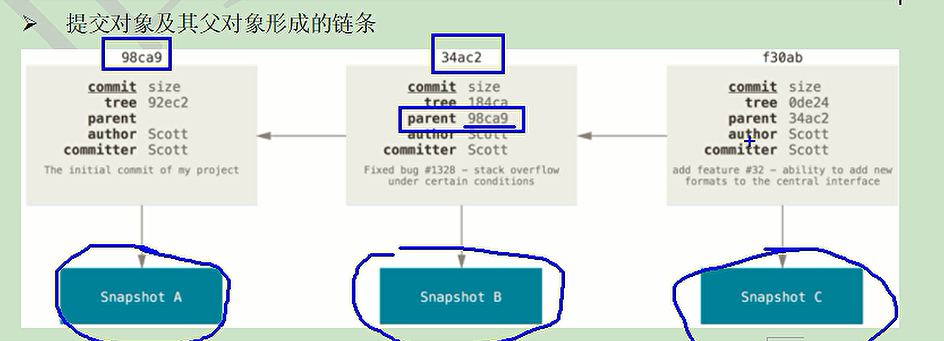




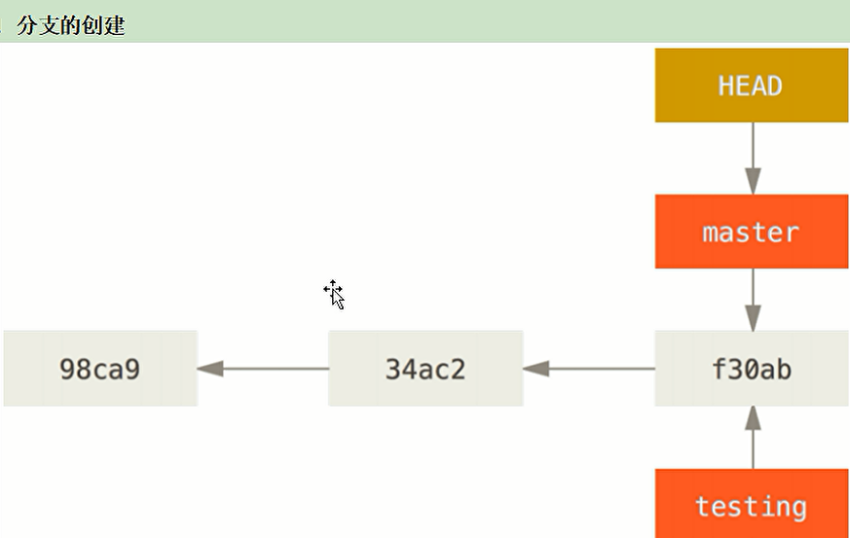


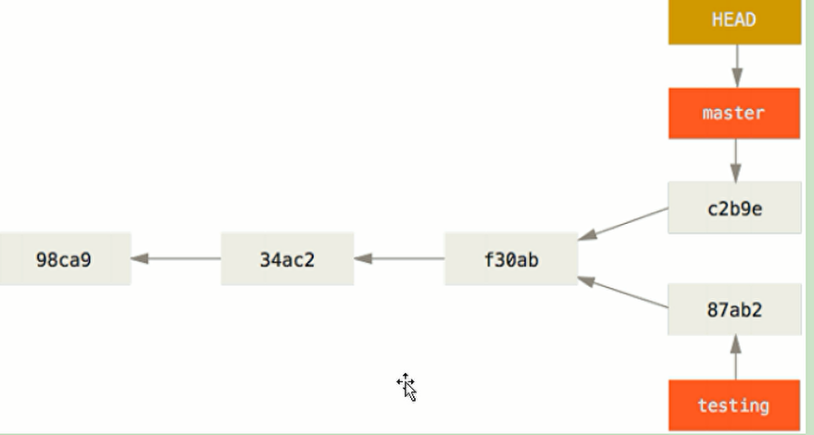
1. **GIT版本管理机制**

****

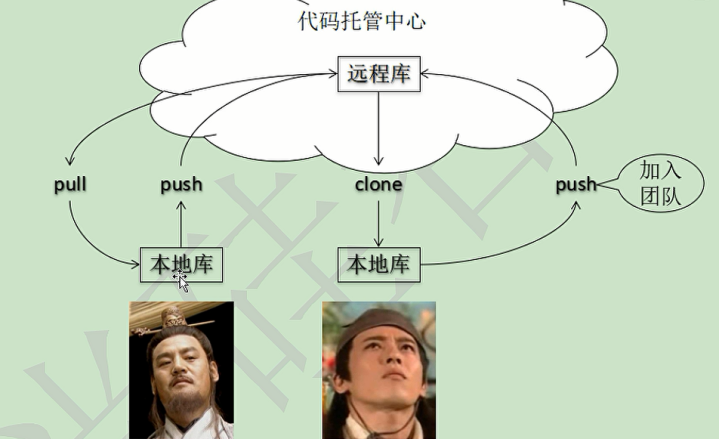


管理分支





1. **远程交互**

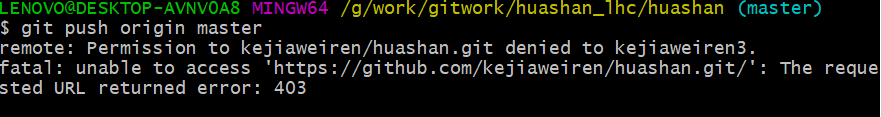


* 1. 岳不群在本地创建远程库地址别名git remote add origin 地址
  2. 岳不群推送 git push origin master
  3. 令狐冲clone项目 git clone 远程地址

clone这个命令有三个效果：下载到本地，创建origin远程地址别名，初始化本地库，所以clone命令之前不需要初始化。



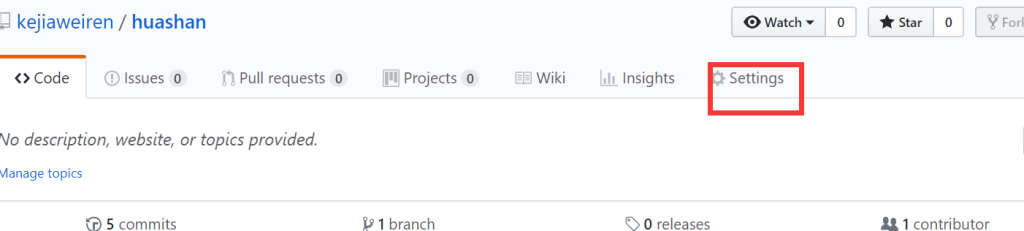
* 1. 令狐冲修改项目，并提交。然后PUSH到github.这时他没有权限



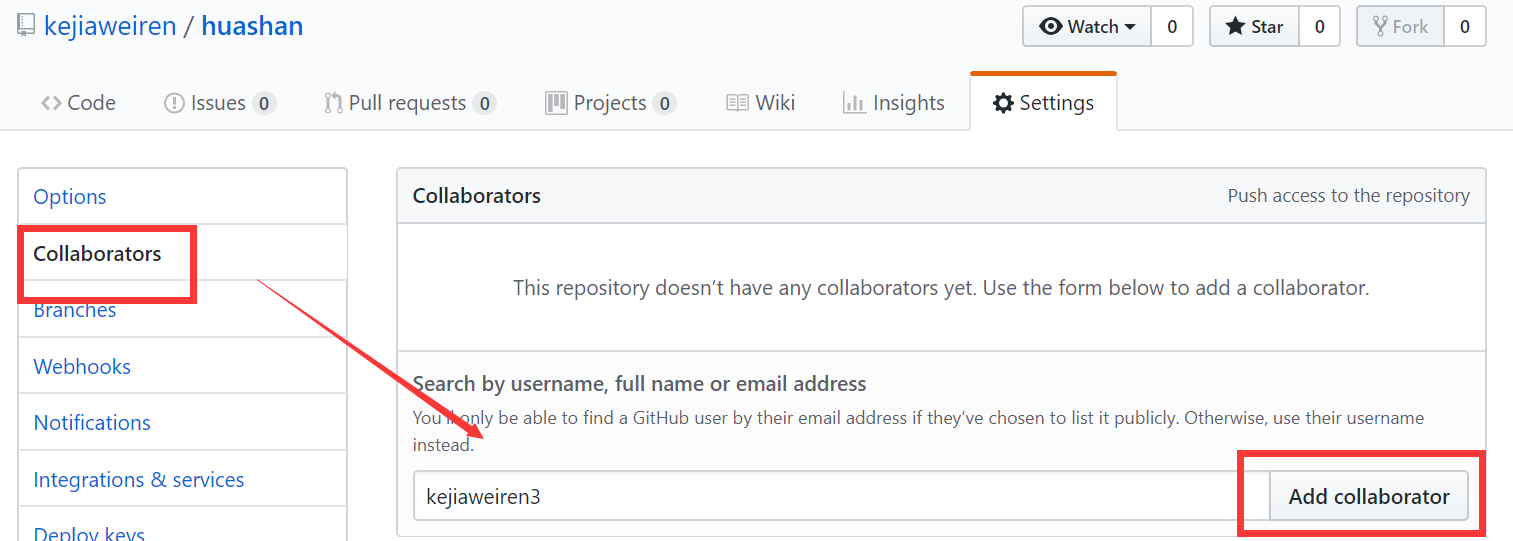
注：测试的时候WIN10会记住密码，需要删除



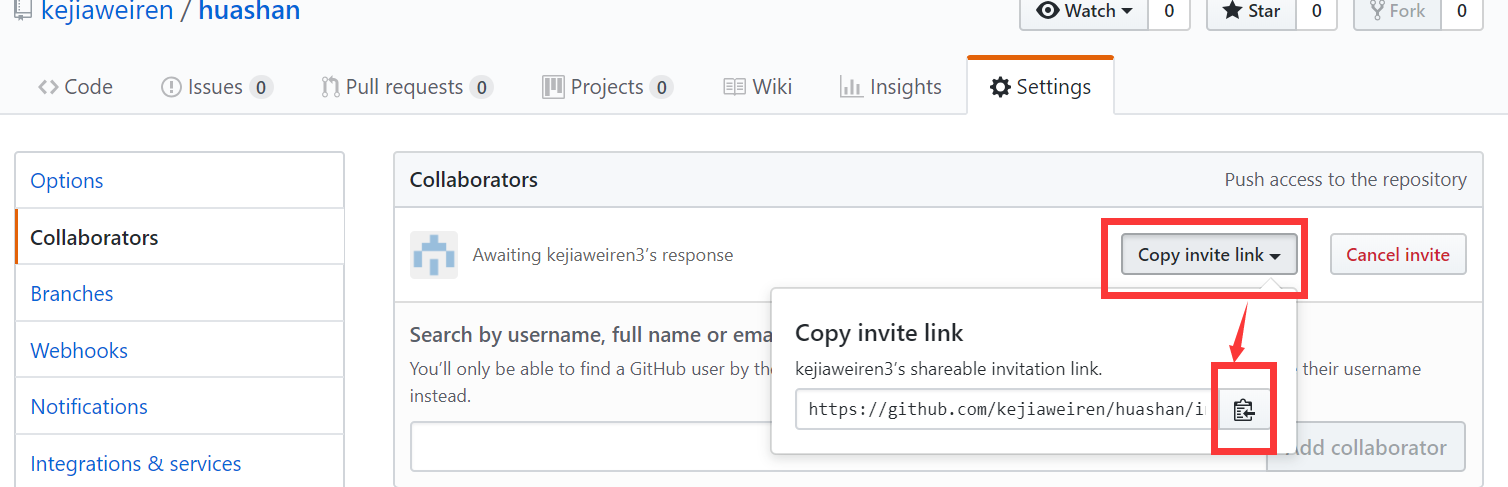
* 1. 这时候需要邀请令狐冲加入团队.在github上操作
     1. 进入仓库，点settings



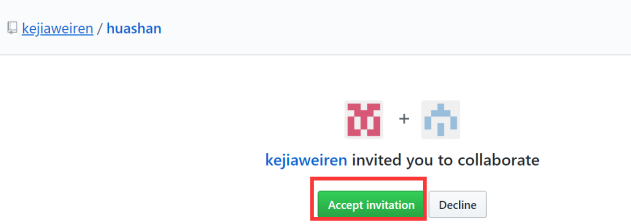
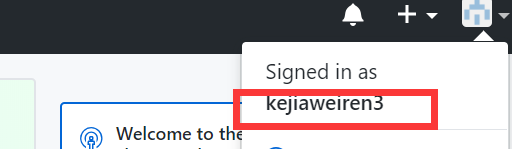
* + 1. 点击collaborators ,输入邀请的用户，点击add collaborator



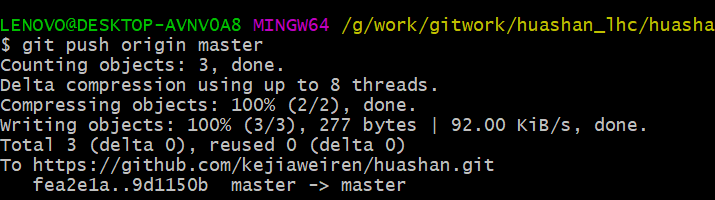
* + 1. 复制连接，发送给邀请者。这时肯定会有相关邮件



* + 1. 邀请者登录，访问刚才的连接，点接受。接受以后令狐冲就变成了团队成员。



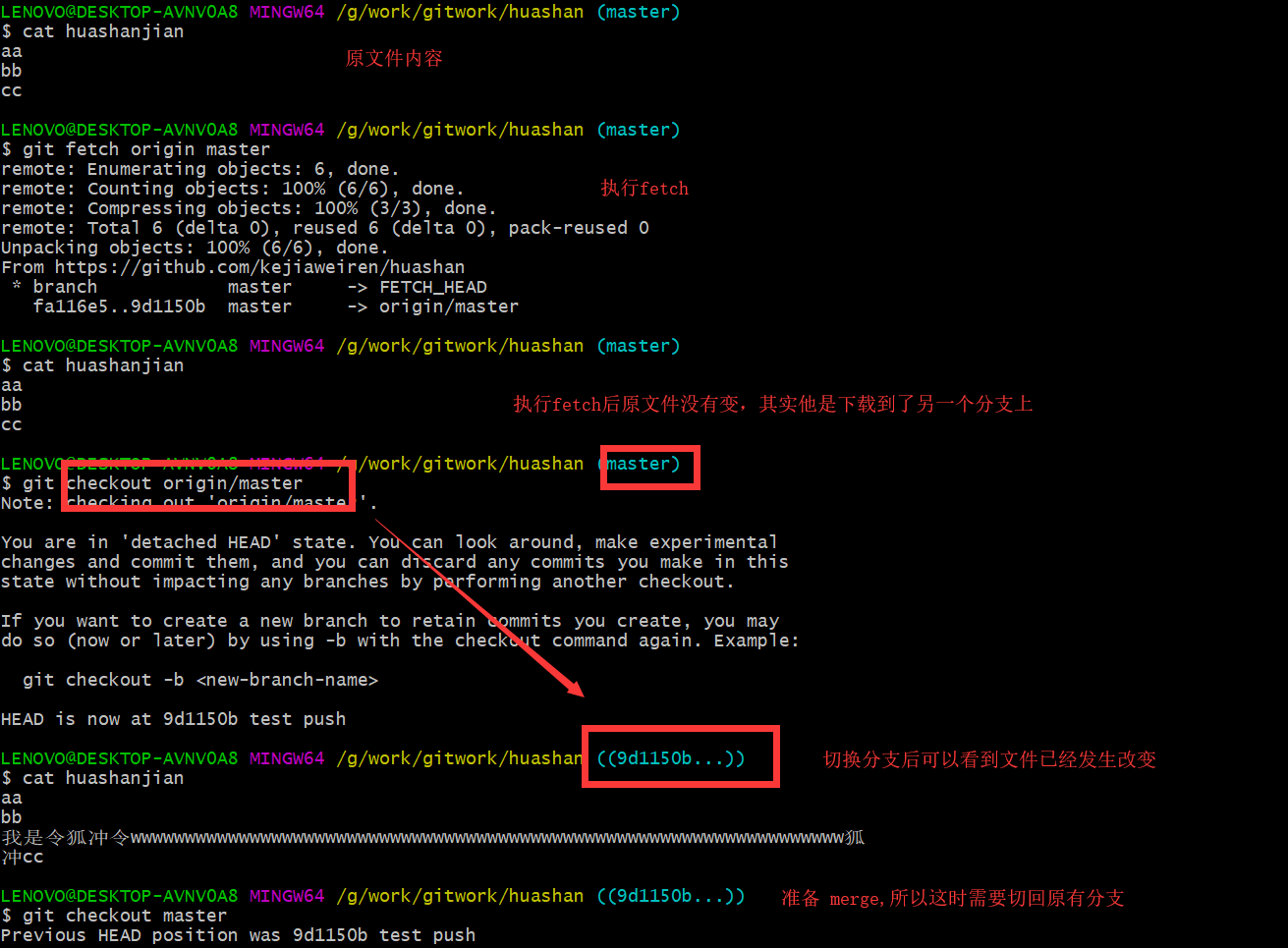
* + 1. 这时令狐冲再次提交，成功

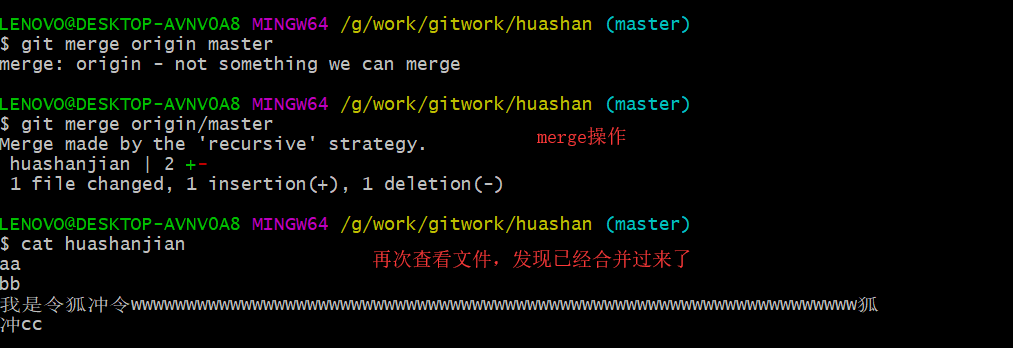


* 1. 岳不群pull修改完的项目。pull=fetch+merge
     1. pull和fetch使用时机(pull和fetch都是读操作，所以不需要登录。)

当文件比较简单，不可能产生冲突时，直接fetch即可。当你需要看一看文件内容，再进行合并，使用fetch

* + 1. git fetch+merge

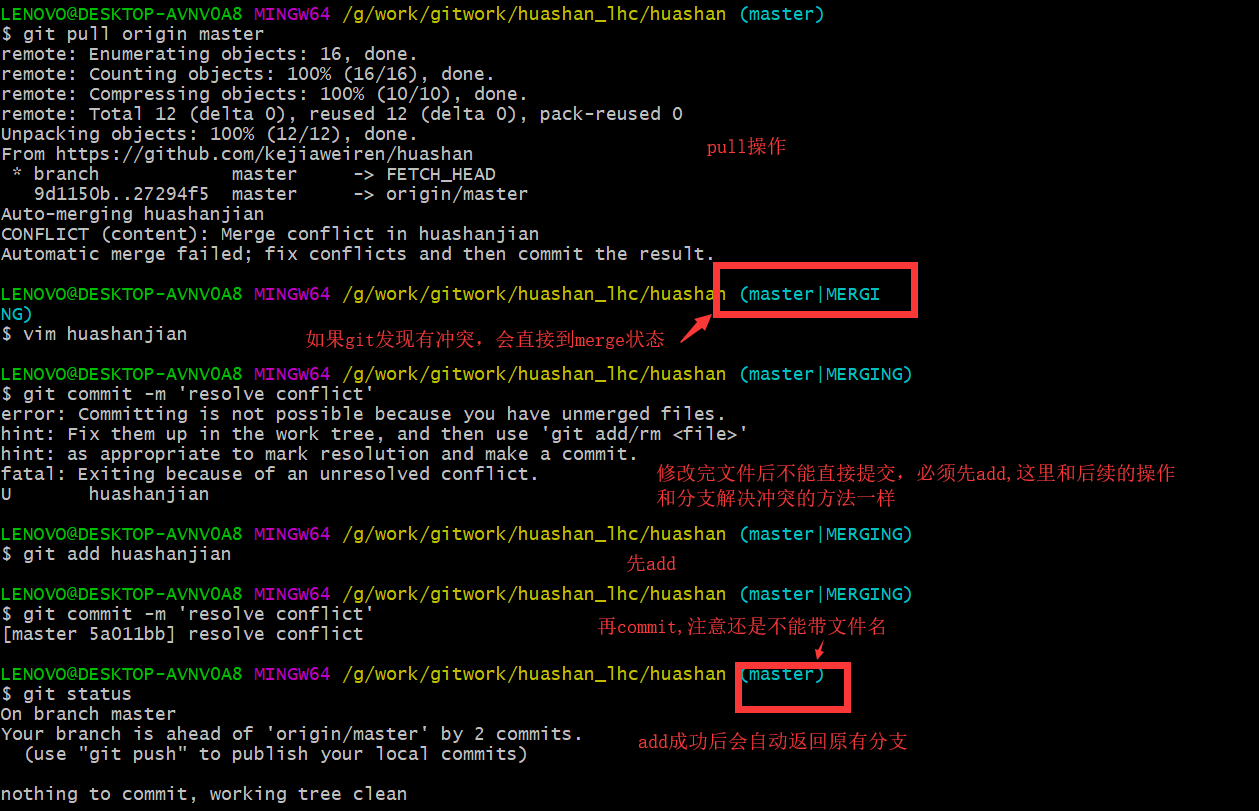




* 1. 协同开发时冲突的解决
     1. 岳不群先提交了，然后令狐冲准备再提交，这时令狐冲会报错，提示必须先pull最新的



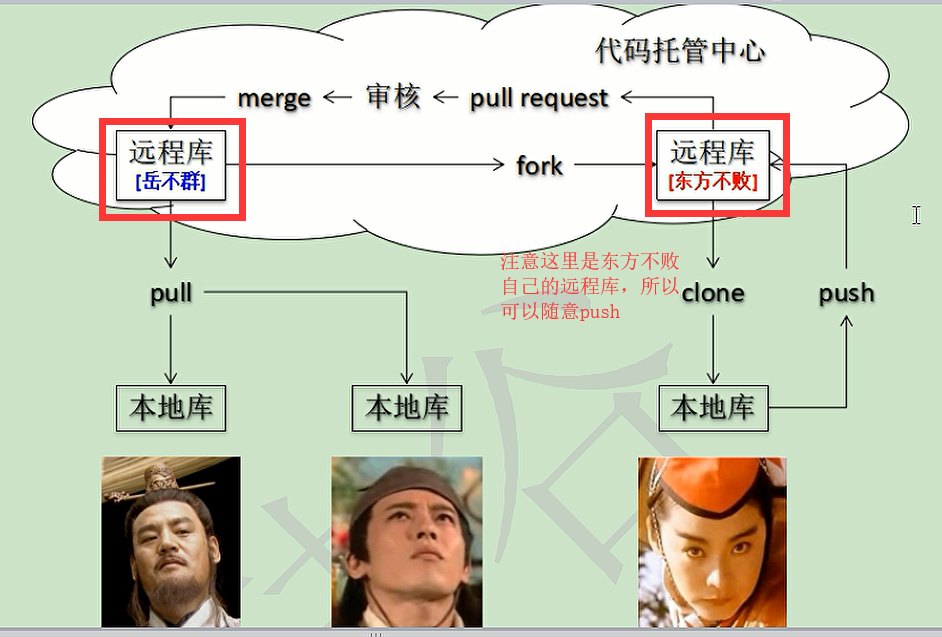
* + 1. 解决冲突



gitadd是为了把冲突标记为已解决的状态。

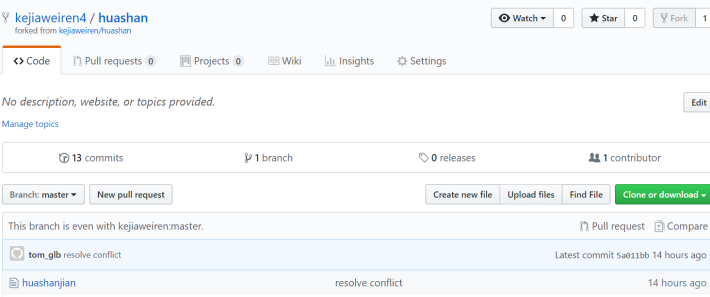
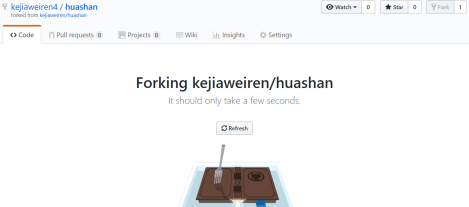
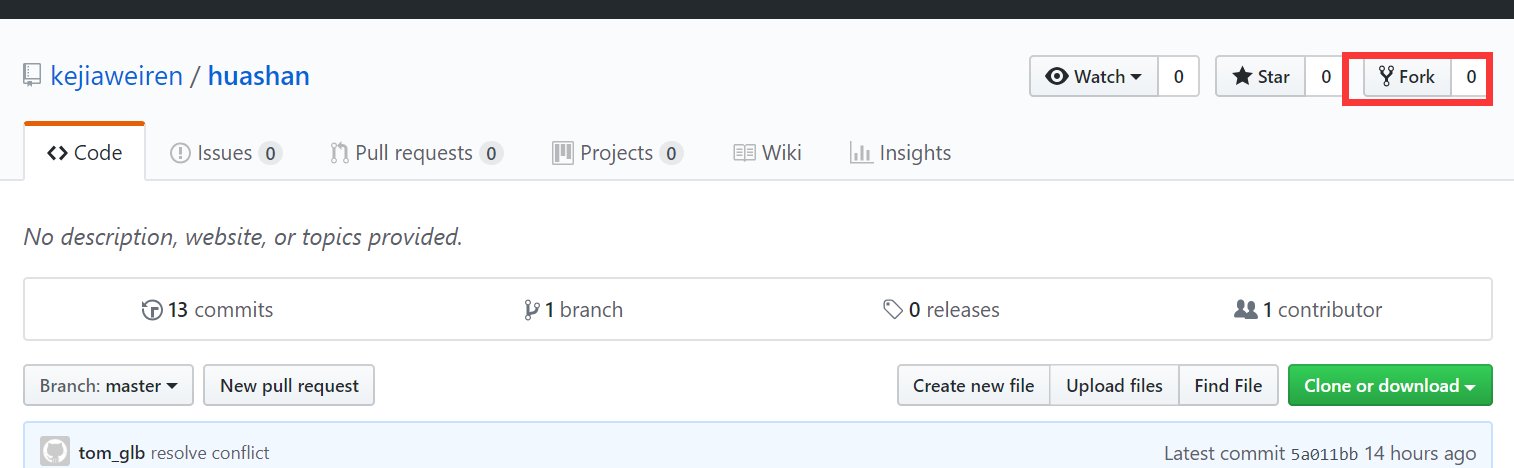


1. 跨团队协作演示

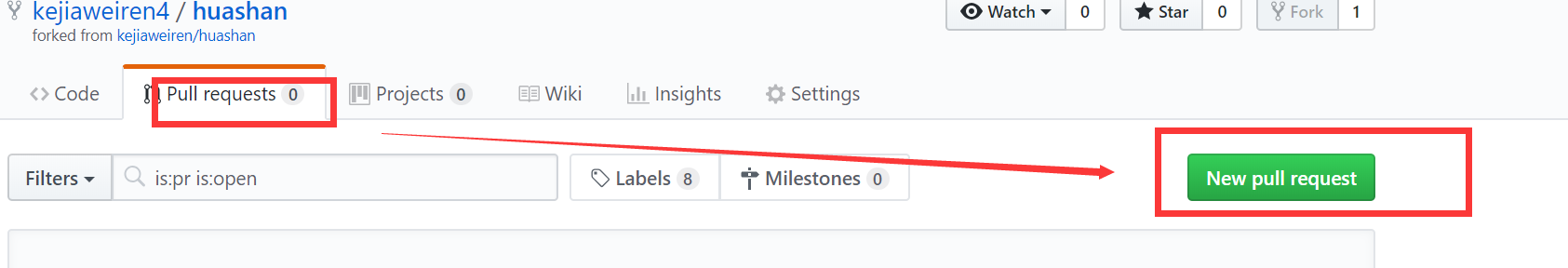


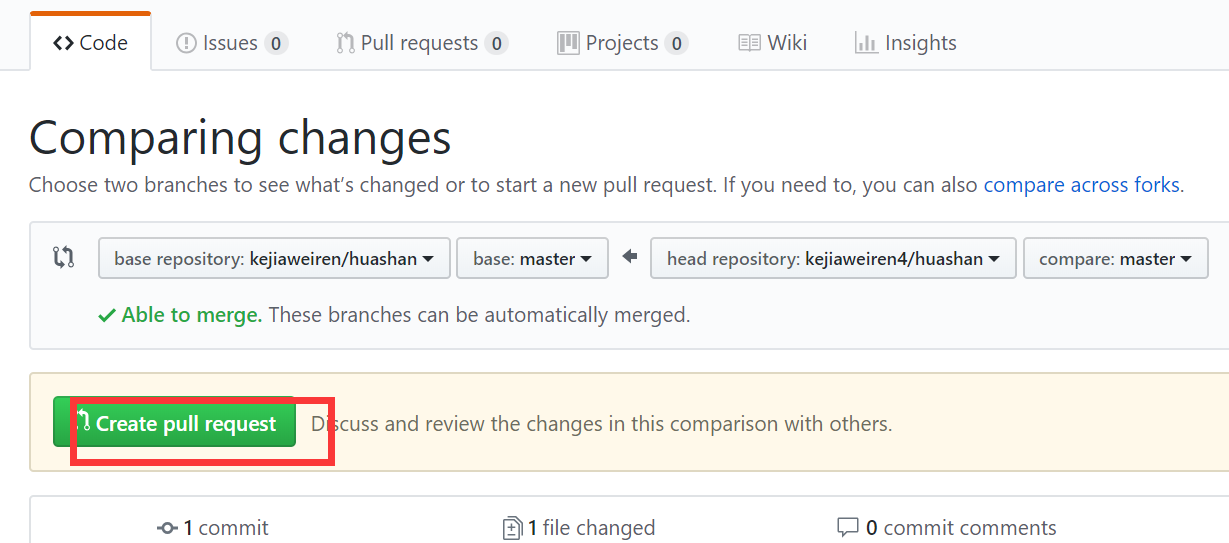
这里是岳不群和东方不败两个团队

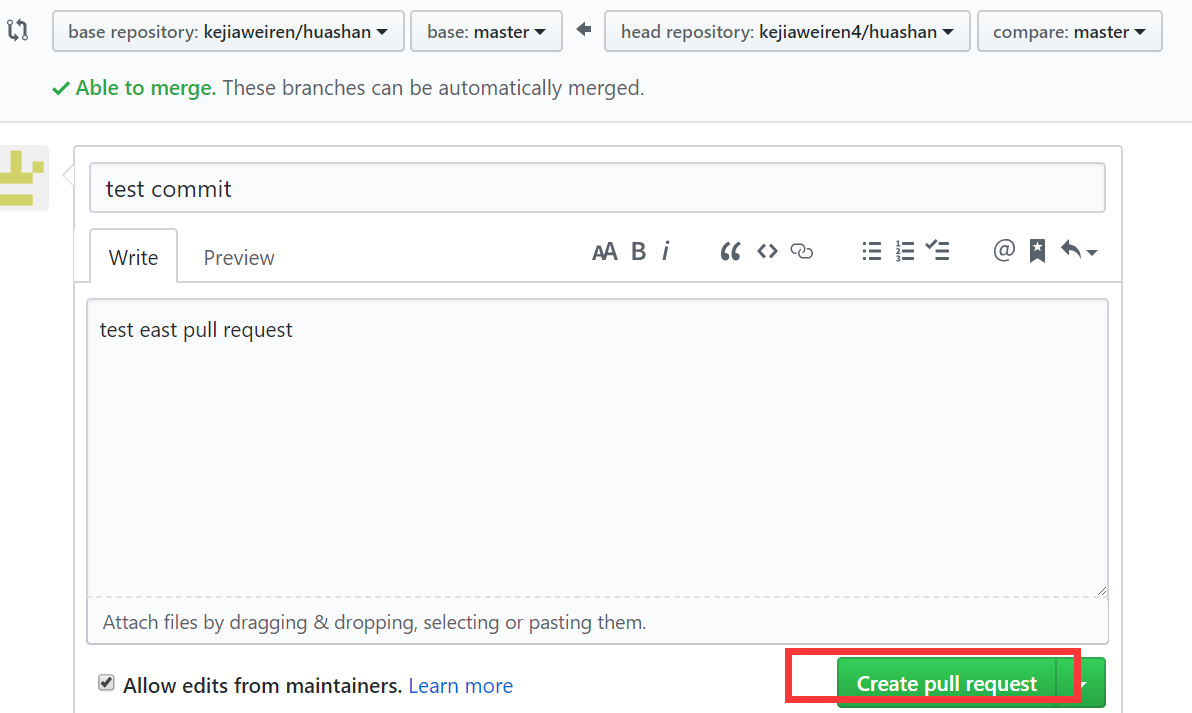
* 1. 东方不败fork代码到他自己的GITHUB中

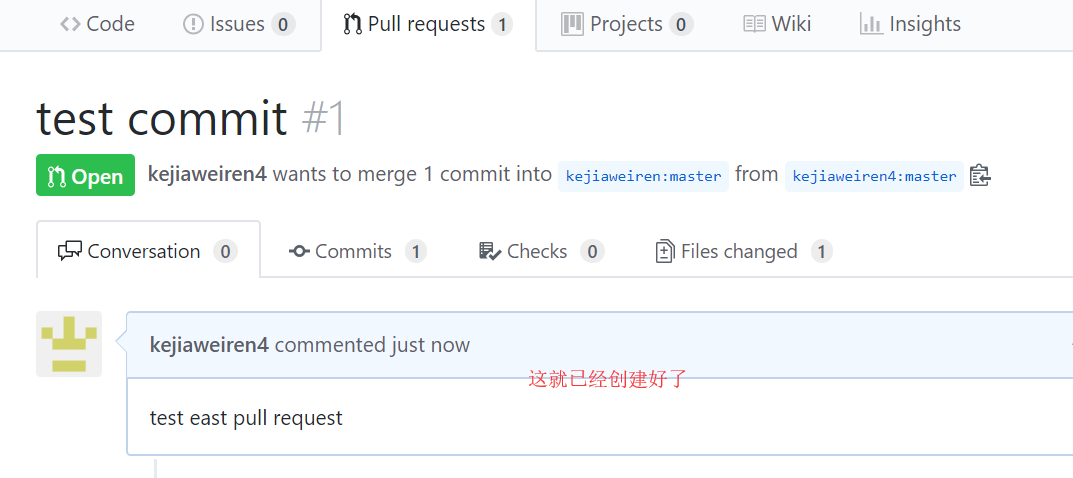


* 1. 东方不败clone自己的github库到自己的本地，提交，并push到自己的github库
  2. 东方不败pull request

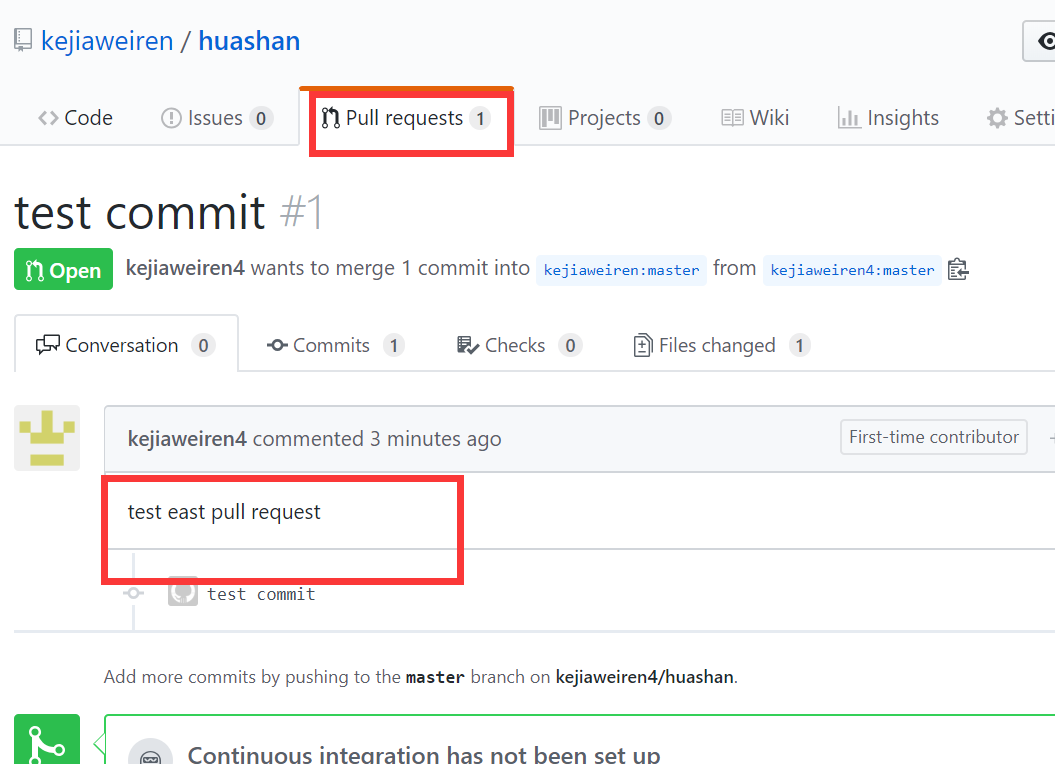


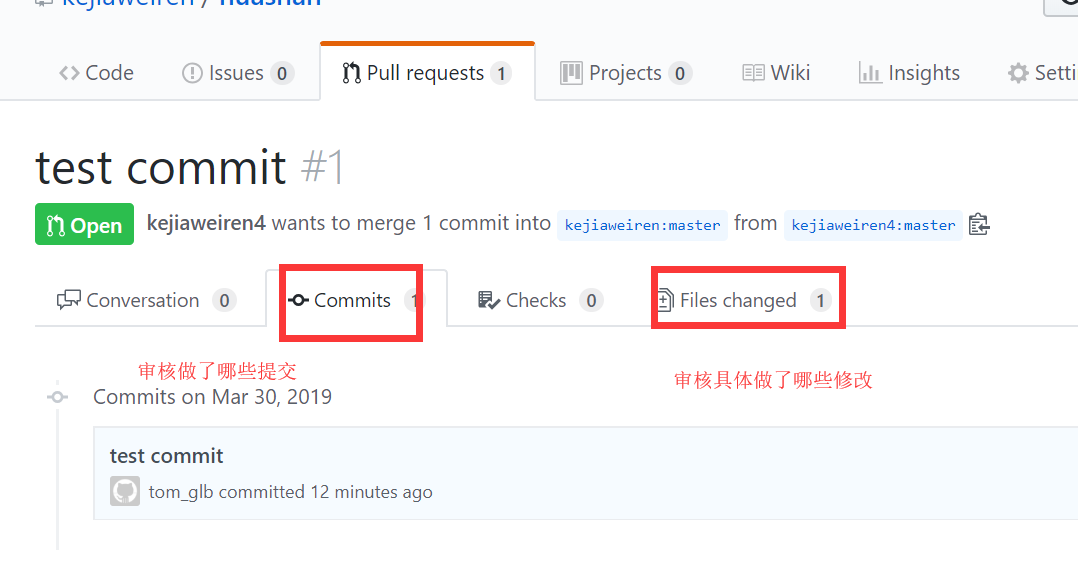




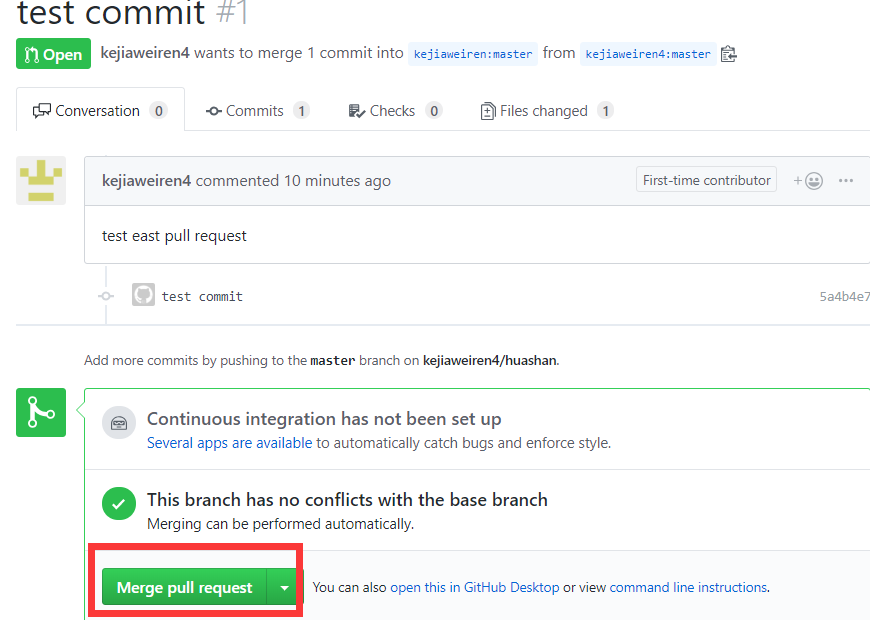


* 1. 主团队登录,审核代码



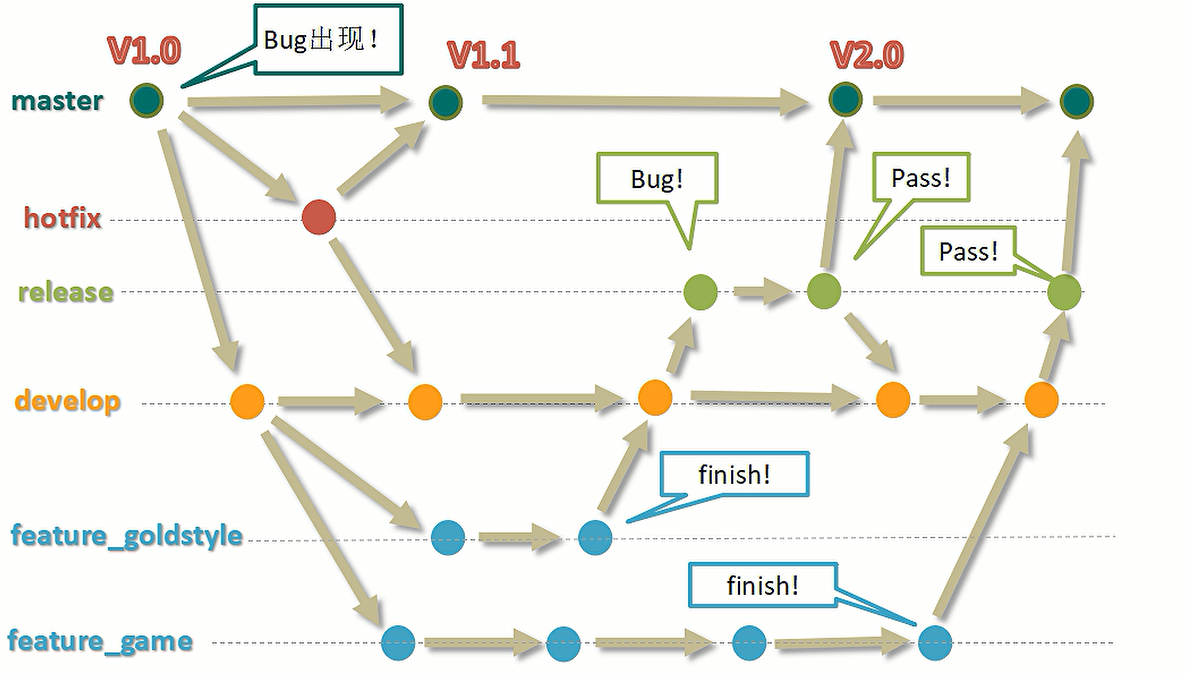


* 1. 主团队合并代码，后面会有需要填写commit信息

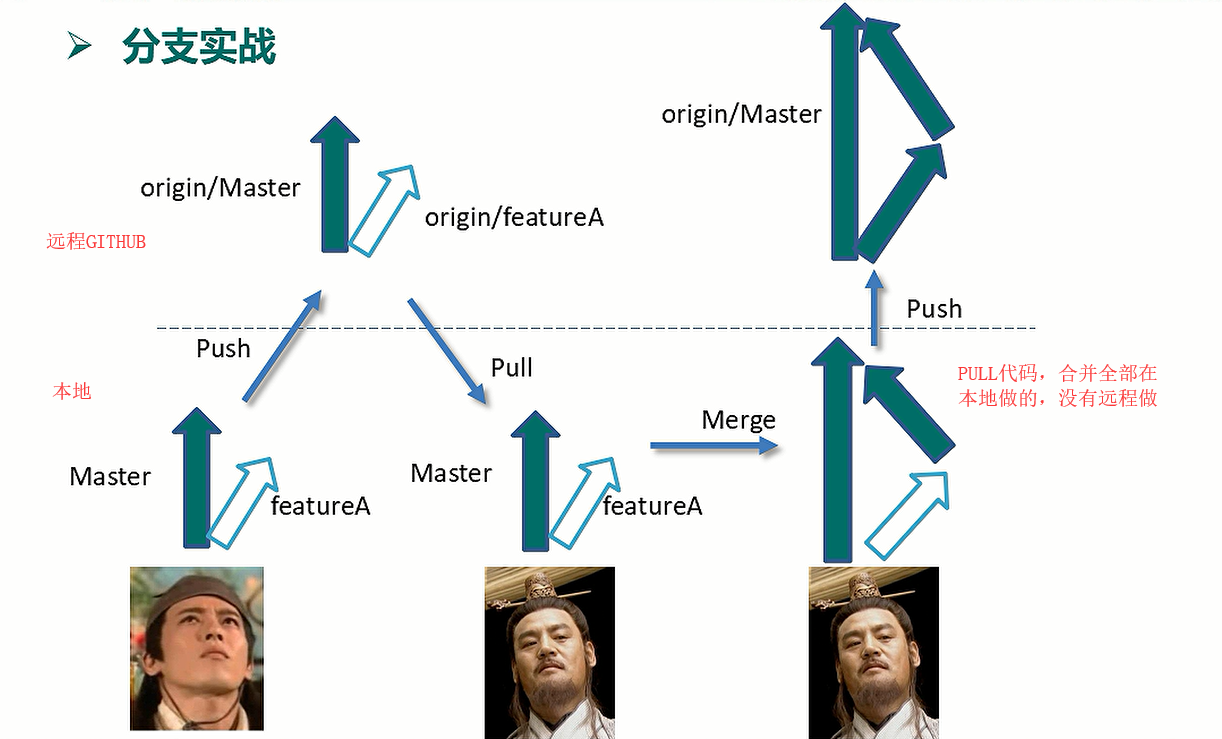


* 1. 主团队pull orgin master

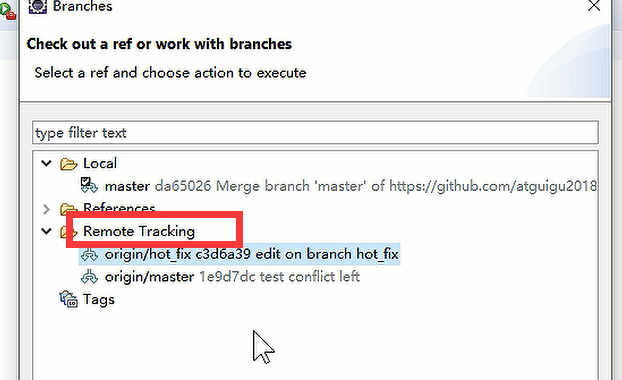
1. git工作流
   1. 集中式（GIT很多特性用不到，相当于SVN），gitflow工作流（用的最多），forking工作流（虚拟团队或者请团队外的成员帮忙解决问题时会用到）



* 1. 分支实战说明



* 1. 分支实战操作
     1. 令狐冲新建分支，修改代码，提交到远程库
     2. 岳不群pull分支，pull时不需要指定哪个分支，pull时候就能检测到服务器新建的分支。选中需要的分支即可。
     3. switch分支，由于分支在远程，所以选中远程的对应分支，然后检出到本地



* + 1. 审核代码，没问题之后，准备合并。合并之前切换分支到master
    2. 进行本地merge 。
    3. 把master推送到远程

1. GITLAB
   1. gitlab安装
      1. 先下载rpm包，再进行安装
      2. 安装命令

sudo rpm -ivh /opt/gitlab-ce-10.8.2-ce.0.el7.x86\_64.rpm

sudo yum install -y curl policycoreutils-python openssh-server cronie

sudo lokkit -s http -s ssh

sudo yum install postfix

sudo service postfix start

sudo chkconfig postfix on

curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash

sudo EXTERNAL\_URL="http://gitlab.example.com" yum -y install gitlab-ce

* + 1. 可以新建一个脚本文件XXX.sh，把这么多命令放在里面一起执行。chmod 755 XXX.sh
    2. 执行之前可以在虚拟机里先拍摄快照
  1. 安装完后重启，开始配置
     1. 初始化配置 gitlab,action run时会执行挺长时间的。千万不要在安装的时候去动他。

gitlab-ctl reconfigure

* + 1. 启动 gitlab 服务

gitlab-ctl start

* + 1. 停止 gitlab 服务

gitlab-ctl stop