#### Git相关命令

1. 基础代码：

ls -la 查看隐藏的文件

ls -l|less 分屏查看

mkdir 新建一个项目: mkdir github

pwd 查看路径

cat name 可以查看文件名称

git init// 对本地化进行初始化

cd ~ 回到home目录下

ls -la||less （代表可以查看隐藏目录）

Tail -n 3 filename 显示3行，

rm filename 删除filename

rm \*.log 删除

1. 设置签名：

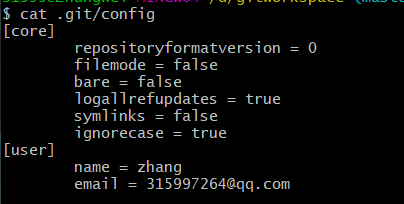
注意：仅在当前的本地库范围有效。主要分为项目级别和系统用户级别，如果两者都存在则优先级为项目级别的用户，两者必须存在一个。

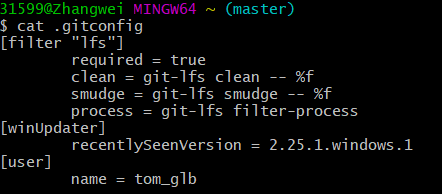
创建一个项目的用户名及邮箱

git config user.name name(project user)

git config user.email email

创建完成之后，信息会存储在以下图示目录下：





与project比较来说，本地只要设置global的账户邮箱即可。（注意回到系统的home目录下）

git config --global user.name name (system user)

git config --global user.email email

cat .git/config

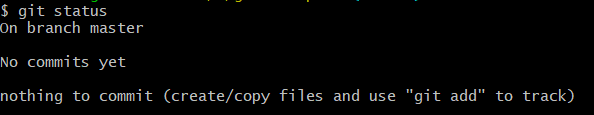
cat ~/.gitconfig(家目录下的gitconfig)

以上即为git在本地操作的初始话过程。

#### 基本操作

1. 状态的查看操作

git status(git主命令+status为子命令或者带参数)—查看工作区、暂存区的状态



vim good.txt(创建一个txt文件) 其中:wq退出文件，:set nu(由于是vim编译器)显示行号，i键进行操作

git add name –提交文件

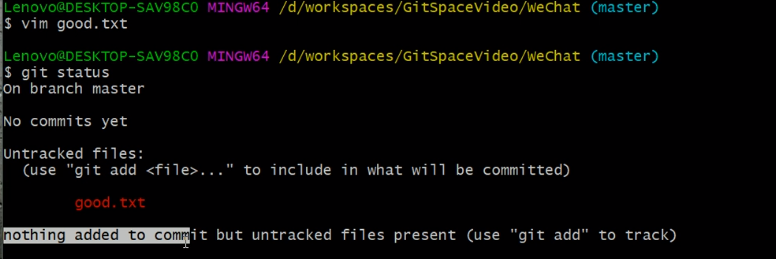
1. 将工作区的“新建/修改”添加到暂存区

git commit good.txt :set nu

或git commit -m “my second commit” good.txt 这样操作方便，提高效率

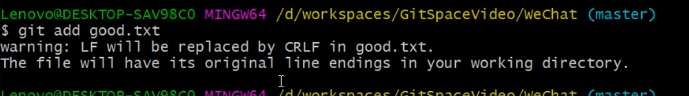
1. 将暂存区的内容提交到本地库

Eg：master (root-commit) 6a34f1a] My fist commit.new file good.txt（数字可以理解为版本，哈希值）

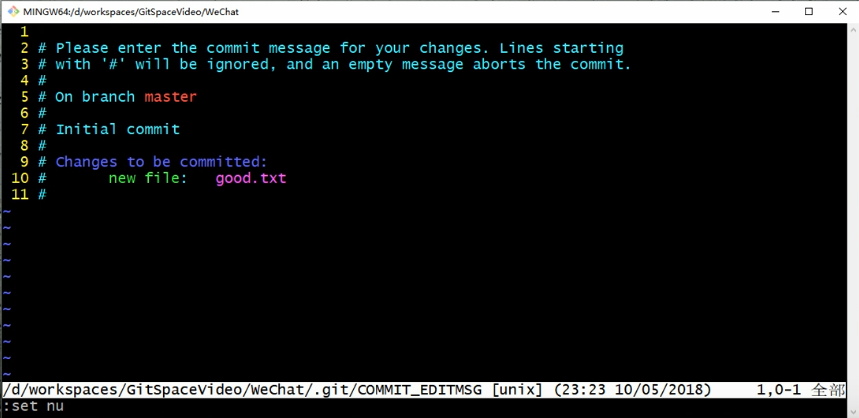


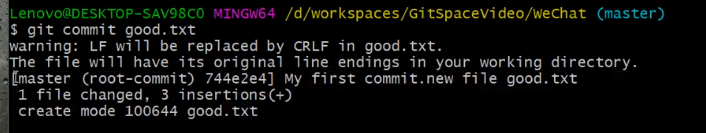
工作区（写代码）

git add—>暂存区（临时存储）git commit—>本地库（历史版本）



在第一行填写修改的地方

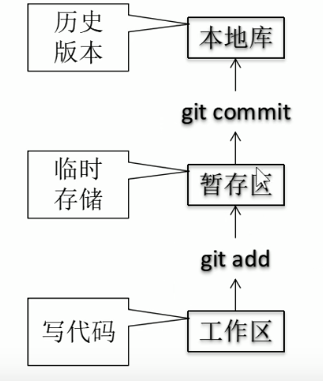




提交的时候git commit -m “commit message” good.txt

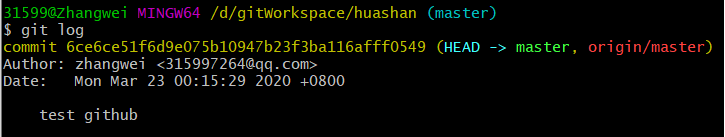


1. 工作区到本地库的工作流程



#### 查看历史记录（处理版本回退的问题）

git log(查看历史提交记录)：多屏控制的方法--空格翻页，b向上翻页，q退出

git log --pretty=oneline

eg: c1dcc4fc649767482230ba6f211e5c793ee4205f (HEAD -> master) My second commit



git log –oneline 查看历史提交日志，单行显示

eg: c1dcc4f (HEAD -> master) My second commit



git reflog：显示了到版本移动的次数HEAD@{移动到当前版本需要多少步}

eg: 6a34f1a HEAD@{1}: commit (initial): My fist commit.new file good.txt

1. 版本回退：

（1）基于索引值进行操作【推荐】

Git reflog

Git reset --hard 3e667d2(索引值)

（2）使用^符号：只能往后退，不能往前

注：一个^表示退后几步

git log --oneline

git reset --hard HEAD^^^^^

（3）使用~符号：只能后退

Git reset - -hard HEAD~n

注：n表示退后几步

Git help 具体命令：可以进行查看

1. reset3个参数对比（soft/mixed/hard）

(1)Soft参数：仅仅移动HEAD指针

(2)Mixed参数：在本地库移动HEAD指针，重置暂存区

(3)Hard参数：在本地库移动HEAD指针，重置暂存区，重置工作区

删除的记录永远不可抹去，创建也是同一个道理，回退某一个历史记录即可。

Git reset –-hard HEAD

1. 删除文件并找回

前提：删除前，文件存在时的状态已经提交到了本地库。

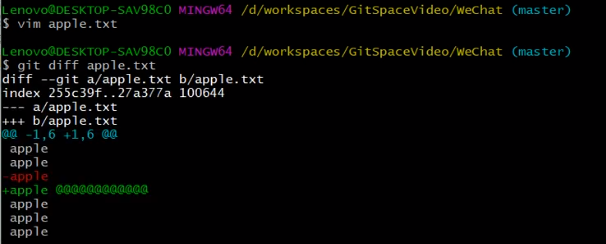
操作：git reset --hard[指针位置]

删除操作已经提交到本地库：指针指向历史记录

删除操作已经未提交到本地库：指针则指向HEAD即可（这时候还没有更新之前的地址）

1. 比较文件差异：

git diff [文件名]：将工作区中的文件和暂存区进行比较

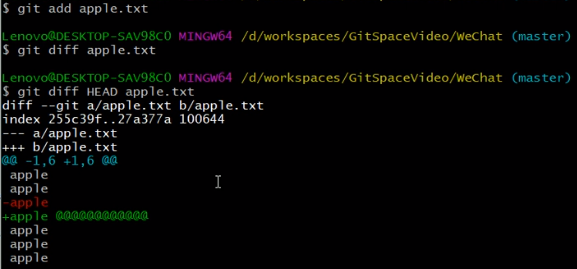


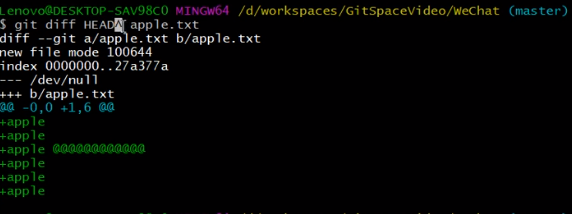
git diff <branch> <filename>

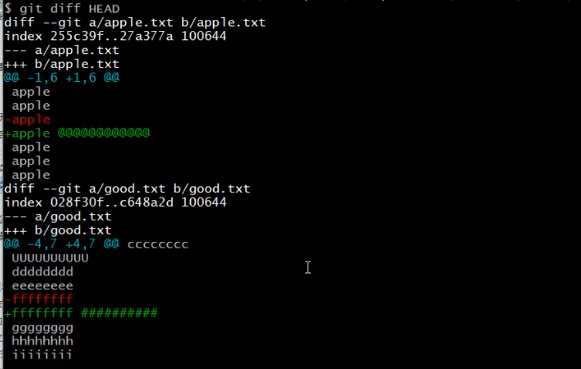
意义：查看文件在工作目录与暂存区的差别。如果还没 add 进暂存区，则查看文件自身修改前后的差别。也可查看和另一分支的区别。

Git diff [本地库中历史版本] [文件名]：将工作区中的文件和本地库历史记录比较

不带文件名比较多个文件(GIT DIFF HEAD)

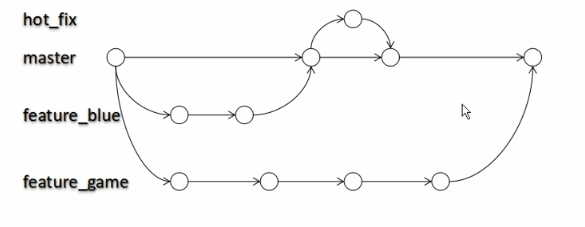






#### 分支管理

1. 什么是分支：在版本控制过程中，使用多条线同时推进多个任务。



1. 分支的好处？

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

1. 分支操作

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]

1. 合并分支：

1.切换到接受修改的分支（被合并，增加新内容）上，

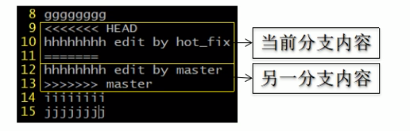
git checkout [被合并的分支名]

2.执行merge命令：

git merge [有新内容的分支名]

#### 解决冲突 ：

1. 冲突的表现



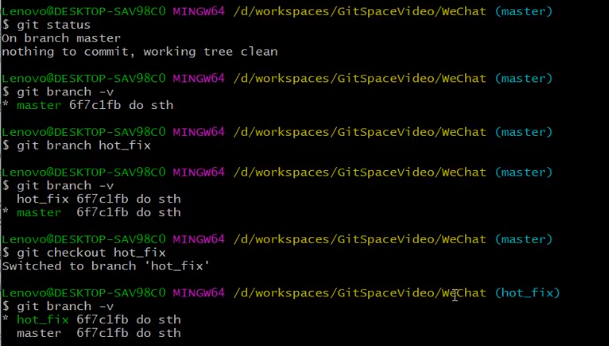
1. 冲突的解决

第一步：编辑文件，删除特殊符号

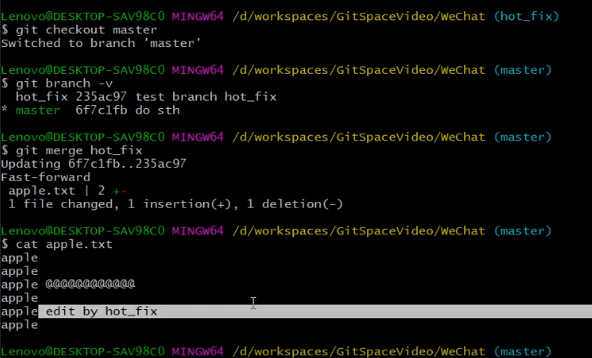
第二步：把文件修改到满意的程度，保存退出

第三步：git add [文件名]

第四步：git commit -m “日志信息” 注意：此时commit一定不能带有具体的文件名

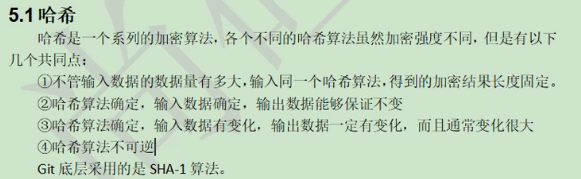


git merge：分支合并到master上



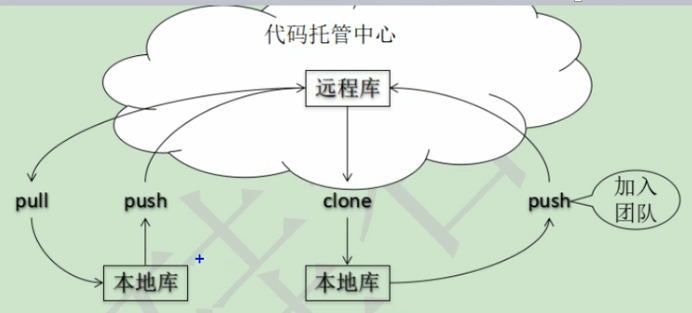
#### Git 原理：



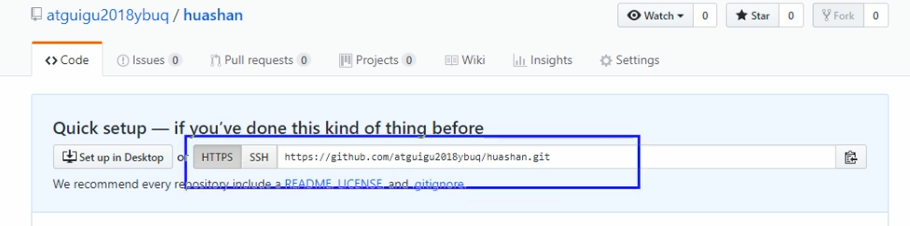


#### 6 GitHub

需要创建远程库和本地库，然后通过push把本地库的文件推送到远程库

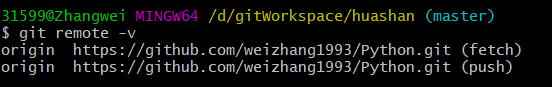


1. GitHub远程库的地址



1. 本地库首先创建一个本地库

git remote -v储存GitHub上的地址

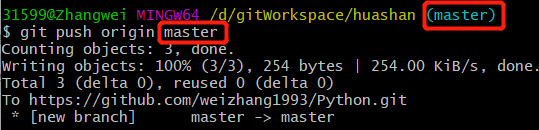


1. 添加地址信息：

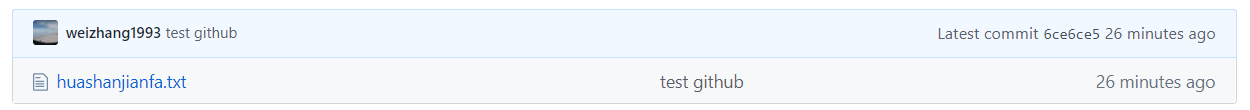
git remote add origin https://地址



git push origin 分支（master/branch）//推送到远程库



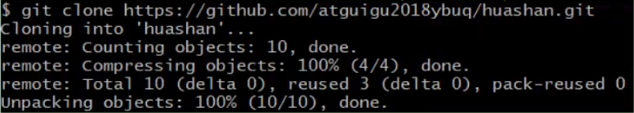
结果：



#### 克隆

1. 步骤

Git clone [远程地址]



效果：

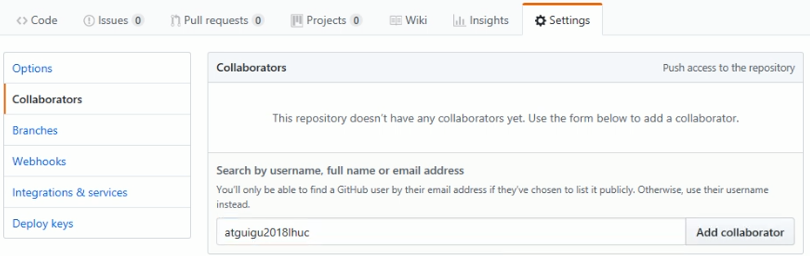
完整的把远程库下载到本地

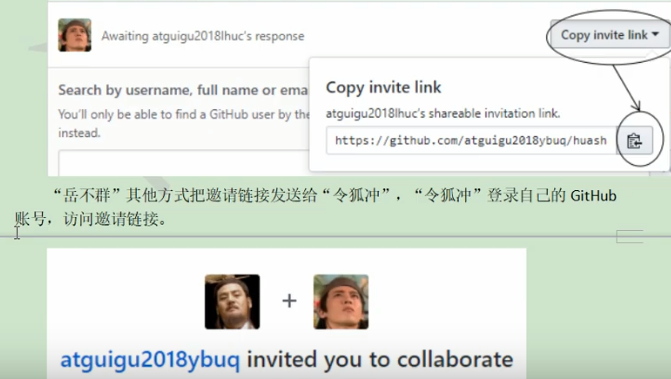
创建origin远程地址别名

初始化本地库

2、如果有团队的加入一起开发的话，流程如下：

首先用git clone去下载项目到本地库，接下来修改文件，然后上传到暂存区（这里需要master给权限，将其加入到项目组中），然后上传到远程库



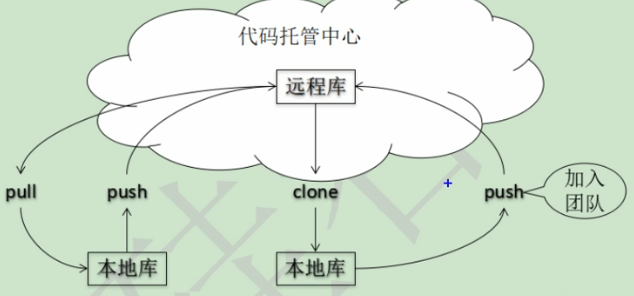


3、控制面板中的凭据管理器中保存有密码：

控制面板\用户帐户\凭据管理器

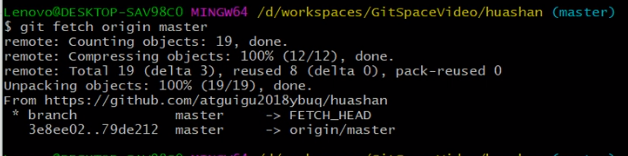


#### 拉取

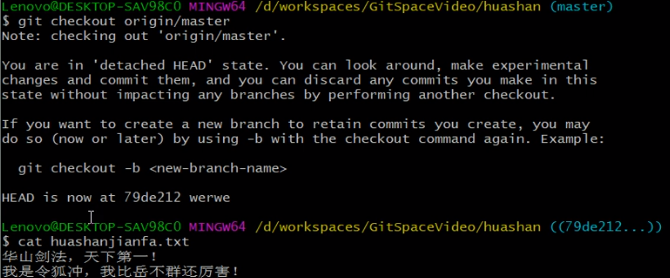


Pull = fetch+merge

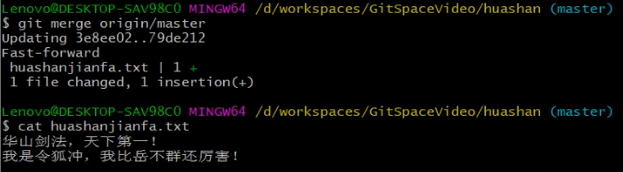
git fetch [远程库地址别名][远程分支名]



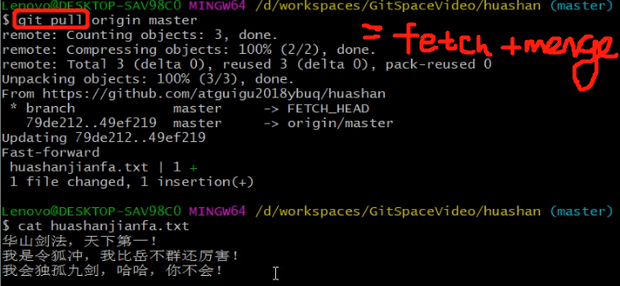
只是下载下来，并没有修改



git mergre [远程库地址别名/远程分支名]



或者直接用pull，但是要注意此种情况适用于冲突比较少的情况



#### 解决冲突

1. 要点：

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

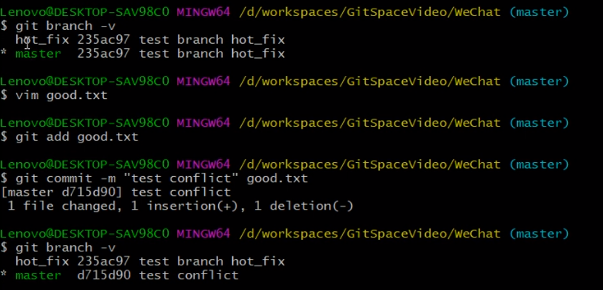
拉取下来如果进入冲突状态，则按照“分支冲突解决”操作解决即可。

1. 类比：

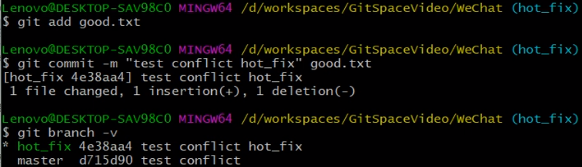
债权人：老王

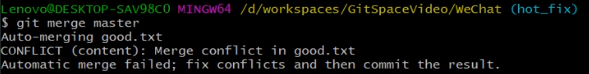
债务人：小刘

老王说：10天之后归还，小刘接受，双方达成一致；老王媳妇说：5天后归还，小刘不能接受，老王媳妇需要找老王确认后再执行。

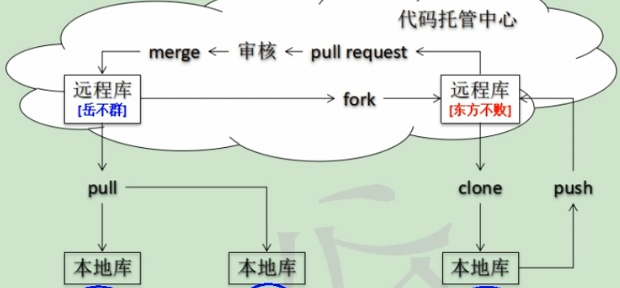




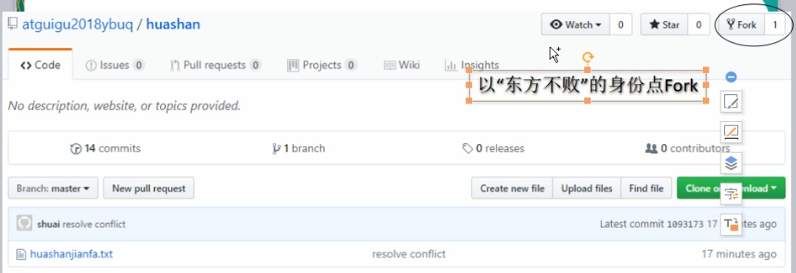




#### 跨团队协作



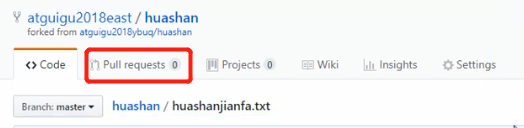
1. Fork操作





1. 本地修改，后推送到远程

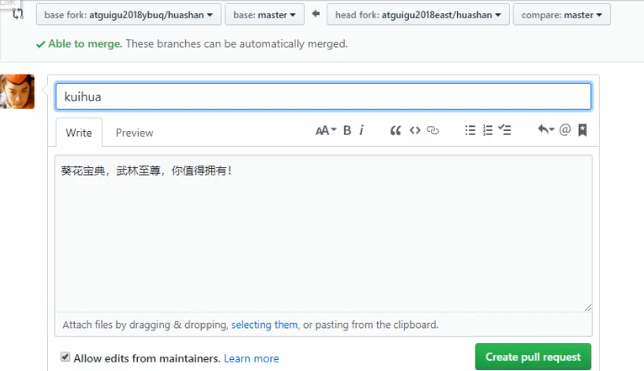
Pull requests



1. 接下来点击这两个操作：

New pull request

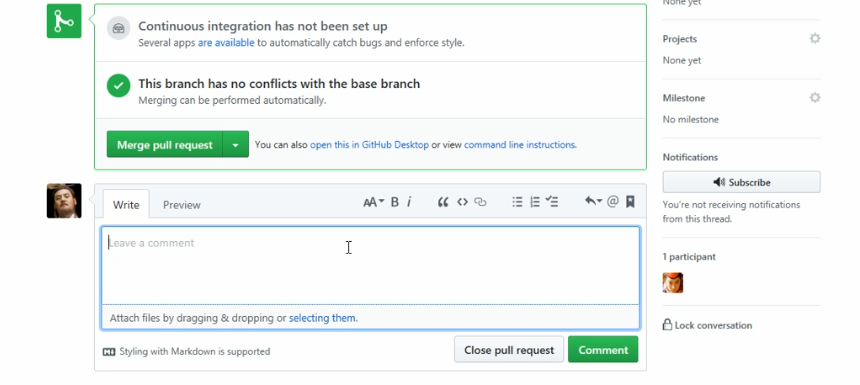
Create pull request



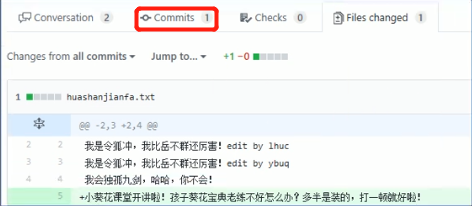
1. 岳不群点击这里



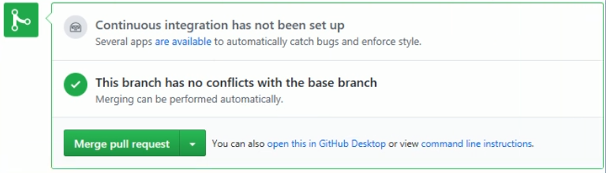
1. 可以进行对话，继续询问一些相关知识。



1. 审核代码：



1. 进行代码合并



1. 填写日志，从而进行代码的合并



1. 将远程库修改，拉取到本地

#### SSH登陆(只能为一个用户开发)

1. 进入当前用户的家目录

cd~

1. 删除.ssh目录

rm -rvf .shh

运行命令生成.shh密钥目录

ssh-keygen -t rsa -C github的账户（1111@qq.com）

[注意：这里的-C这个参数是大写的C]

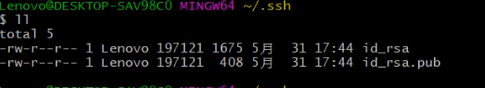


1. 进入.shh目录查看文件列表

·cd .shh

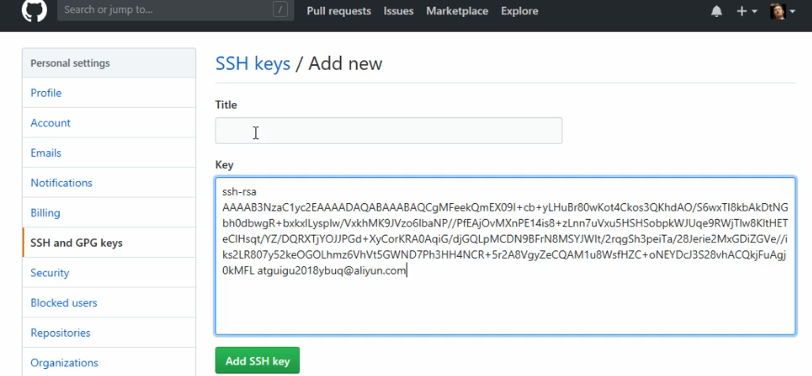
·ls -LF

1. 查看id\_rsa.pub文件内容



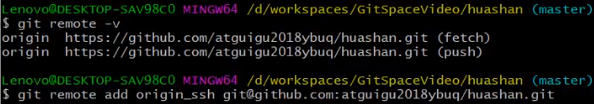
·cat id\_rsa.pub

1. 复制id\_rsa.pub文件内容，登陆GitHub，点击用户头像—settings—SSH and GPG keys
2. New SSH Key
3. 输入复制的密钥信息

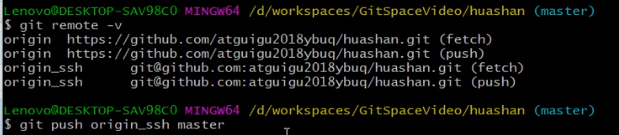


1. 回到git bash 创建远程地址别名

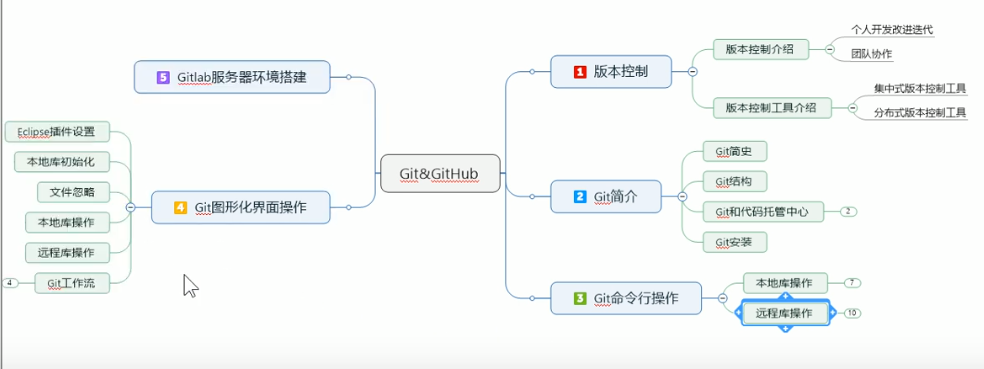
git remote add origin [ssh.git@github.com.....（origin](mailto:ssh.git@github.com.....（origin)随便起，是别名）



1. 进行推送，从而不需要进行输入密码，要回到工作空间进行测试



#### 总结：



Git与eclipse的应用，请查看：<https://www.youtube.com/watch?v=RFUUyufwHsE&list=PLmOn9nNkQxJFISyrvKUk8OWg-Pzs1donX&index=54>