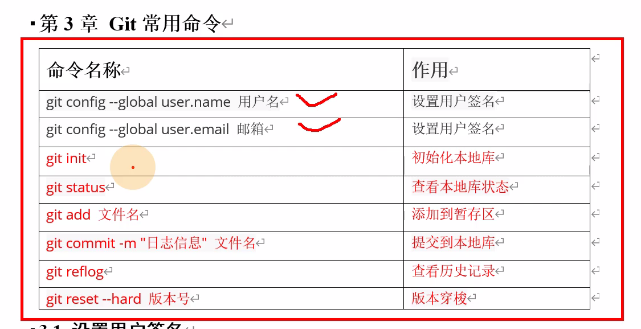
* Git常用命令

红色为常用的。黑色为初始化设置时用，用的较少。

(1)必须设置用户名和邮箱，否则无法提交代码，这里的邮箱并不是登录github时的邮箱，而是相当于一个个人签名似的



用法范例：

(2)

全局设置：

git config --global user.name tom：添加用户名tom

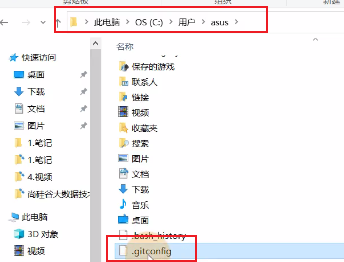
git config --global user.name: 查看用户名

局部设置：

git config user.name tom：添加用户名tom

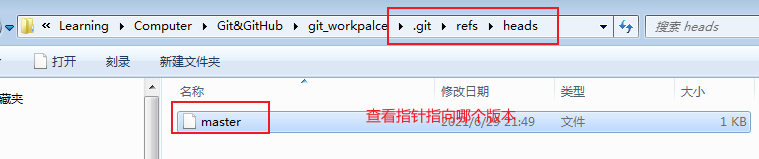
git config user.name: 查看用户名

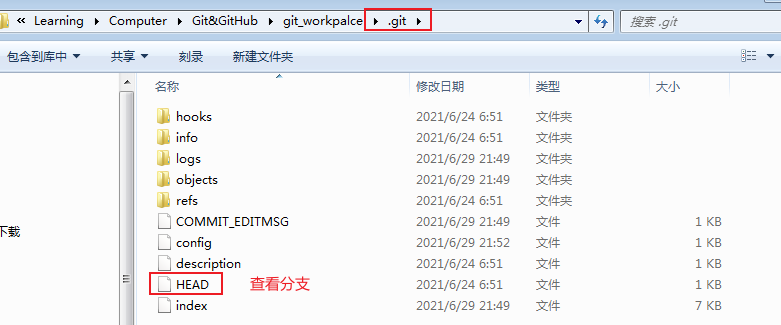
用户名，邮箱签名保存的位置



(3)git init :使用方法，在选定文件夹下打开git，输入此命令即可创建一个目录”.git”默认是隐藏的，就是本地代码的库

* 查看git当前版本信息





* git rm -r - -cached 文件名：删除项目

注意- -cached前面两个横杠, -r是递归删除文件夹内所有文件

删除之后需要提交才可以

git commit –m “删除的文件名”

* 使用git向github.com上传项目步骤

首先把本地的文件添加到本地仓库里面，

一，git add SpringTest001 先添加到缓存区

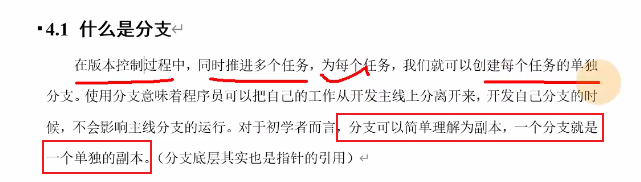
二，git commit -m “first edition” SpringTest001 这步才是添加到本地仓库

三，git remote add 别名 <https://github.com/YZF-R1/nicholas.git> 链接个人远端账户，后面是个人github地址

四，git push –u origin master 执行上传命令

如果出错就多试几次，网络不好

* 分支的概念



分支相关命令

git branch branchName: 创建新的分支

git branch -v :查看处于哪个分支上

git branch -a : 查看所有分支

git checkout ‘branchName’ :转到指定名字的分支上

git merge [branchname] : 把指定名字的分支合并到当前分支



* 创建分支hot\_fix之后，就是创建了一个Test.txt的副本，在这个分支上进行修改，然后也可使用git add Test.txt 和git commit -m “hot\_fix\_v01” Test.txt进行提交。

提交完之后可在文件夹：.git\refs\heads下查看其对应的版本号与git relog命令查到的版本号一致

* git merge hot\_fix(分支名，可自定义): 合并分支到当前分支，例如在master分支操作就合并到master分支下

1. 正常合并，指在master分支下文件的基础上进行了修改， git可自动合并
2. 冲突合并，指hot\_fix修改了文件Test.txt，之后master又在自己原有的基础上进行了修改，两个分支修改的不一样，这样合并时就会冲突，需要人工决定，

\*人工决定步骤：

<1>在Test.txt中手动修改，并保存，

<2>git add Test.txt

<3>gti commit -m “merge test”(注意这里后面不用加Test.txt)



* 远程库相关命令

git remote –v ：查看远程库及别名

git remote add nicholas <https://github.com/YZF-R1/nicholas.git> :创建远程库别名为：nicholas

git remote remove origin : 移除远程库链接

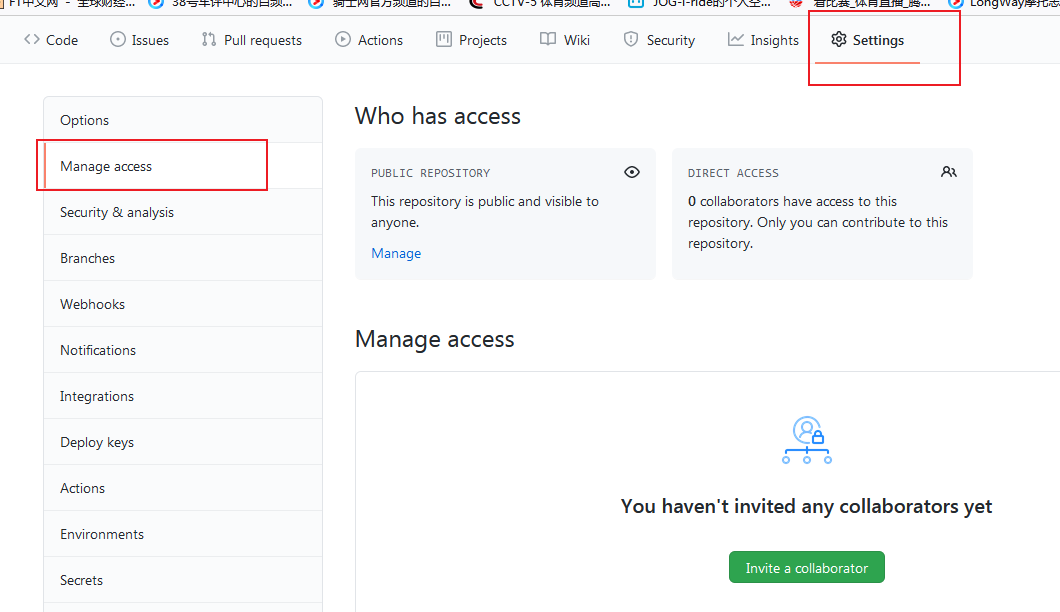
git push -u nicholas master :把本地的master分支推送到nicholas所代表的远程库上。

git push -u 链接地址 master: 也是推送本地master分支到远程库

git pull -u nicholas master : 把远程库拉取到本地，更新本地库代码，此操作自动提交到本地库

git clone <https://github.com/YZF-R1/nicholas.git> :克隆远程库代码到本地，此操作不需登录账户

* 邀请团队内成员协作的设置方法



* 设置ssh秘钥命令

ssh-keygen -t rsa -C [atguiguyueyue@aliyun.com](mailto:atguiguyueyue@aliyun.com)

* git彻底清除项目的步骤

 用git-bash进入project目录（包含.git子目录的目录）；

 rm -rf .git //删除原有'.git'本地仓库，彻底消除所有commit历史!

 git init //重新初始化一个git本地仓库

 git add . //将project目录所有内容作为第一次提交

 git commit -m "Initial commit" //同上

 git remote add origin <github-uri> //加远端GIT仓库(添加多个请直接修改'.git\config'

 git push -u --force origin master //强推并覆盖远端GIT仓库所有历史！

* gitee初步使用方法



* 删除 untracked files命令

git clean -f ：删除未跟踪的文件

git clean -fd: 删除未跟踪的文件夹

* 强制融合，提交，拉取、

git merge master --allow-unrelated-histories

git pull origin master --allow-unrelated-histories

* git取消本地修改

1，取消git add执行之前的修改

git checkout . : 取消本地所有修改，即在，注意有个点“.”

git checkout -- filepathname : 取消单个文件的修改

2，取消git add 之后，在提交commmit之前，加入缓存区后的修改

git reset HEAD filepathname : filepathname是文件的路径

3，已经提交执行commit命令之后

git reset --hard HEAD^ : 回退到上一次commit的状态

git reset --hard  commited : 版本穿梭

* git命令行加密码

git remote add origin [http://{username}:{password}@101.43.18.244:3030/feilukeji/wechat-funeng-server.git](http://%7busername%7d:%7bpassword%7d@101.43.18.244:3030/feilukeji/wechat-funeng-server.git)

注意是复制http链接，不是ssh链接

* GitHub增强脚本

<https://greasyfork.org> 里搜索

具体地址

<https://greasyfork.org/zh-CN/scripts/412245-github-%E5%A2%9E%E5%BC%BA-%E9%AB%98%E9%80%9F%E4%B8%8B%E8%BD%BD>

* idea的gitingore文件不起作用解决办法

直接删了.idea和\*.iml文，然后把.gitignore文件放到目录下重新打开项目