**Git知识点**

本地库初始化：git init

将工作区添加到缓存区：

git add [文件名] //例如：git add readme.txt

git add **.** 或 git add all //两者都能将工作区中所有未跟踪或者修改的文件添加到暂存区

查看状态：

git status //用于显示工作目录和暂存区的状态。使用此命令能看到那些修改被暂存到了, 哪些没有, 哪些文件没有被Git tracked到。Stage有暂存区的意思。

将缓存区提交到本地库：

git commit [文件名] ，此时会跳到写注释的界面，写完注释后返回可以先按ESC退出编 辑，然后再连按大写的Z两次，即可返回（不推荐）

git commit -m “注释内容” [文件名] （推荐）

git log //最详细的显示，有可能会有多页。多页的控制方式：空格向下翻页，b向上翻页，q退出

git log - -pretty=oneline //较为精简的显示

git log - - oneline //更为精简的显示

git reflog //能够查看索引值； HEAD@{数字} ，数字代表移动到当前版本的需要步数

cat [文件名] //读取文件内容

回退或前进到某个版本

基于索引值操作（推荐）

git reset - -hard [索引值]

使用^符号(只能用于后退)

git reset - -hard HEAD^ //一个^表示后退一步，n个表示后退n步

使用~符号(只能用于后退)

git reset - -hard HEAD~n //把n替换为数字，表示后退n步

reset 命令的三个参数对比（index file指的是暂存区，working tree指的是工作区）

- -soft参数

仅仅在本地库移动HEAD指针

- -mixed参数

在本地库移动HEAD指针、重置暂存区

- -hard参数

在本地库移动HEAD指针、重置暂存区、重置工作区

删除文件并找回

前提：删除前，文件存在时的状态提交到了本地库

操作：git reset - -hard [指针位置]

删除操作已经提交到本地库：指针位置指向文件存在时的历史纪录

删除操作尚未提交到本地库：指针位置使用HEAD

删除文件：rm [文件名] //例如： rm readme.txt

查看该目录下有几个文件：ll //英文字母小写的L

比较文件差异

git diff [文件名] //将工作区的文件和暂存区进行比较，即还没有进行git add命令

git diff [本地库中的历史版本] [文件名] //将工作区的文件和本地库历史记录比较

git diff //不带文件名则比较多个文件

分支

查看分支：git branch -v

创建分支：git branch [分支名]

切换分支：git checkout [分支名]

合并分支：

第一步：切换到接受修改的分支（被合并，增加新内容）

git checkout [被合并分支名]

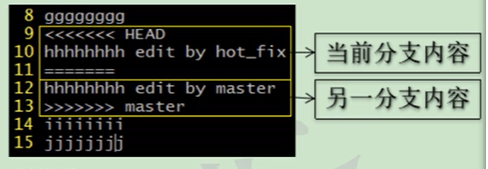
第二步：执行merge命令

git merge [有新内容分支名]

例如：想要把hot\_fix分支合并到master分支上，先切换到master分支上，然后再把hot\_fix分支合并到master上，先git checkout master 后git merge hot\_fix

解决冲突：

冲突的表现



冲突的解决

1. 编辑文件，删除特殊符号
2. 把文件修改到满意的程度，保存退出
3. git add [文件名]
4. git commit -m “日志信息” （此时commit一定不能带具体文件名）

git remote命令的用法

查看关联的远程仓库信息

git remote //查看关联的远程仓库的名称

git remote -v //查看关联的远程仓库的详细信息

添加远程仓库的关联

git remote add origin [url] //origin为远程仓库的别名，也可以设为其他名称

url是指远程仓库的url，可采用http协议或ssh(git)协议

删除远程仓库的关联

git remote remove [name]

推送到远程仓库

git push origin master //origin是之前设置好的别名，然后必须指明分支，此处为master

冲突：

如果不是基于GitHub远程库的最新版所做的修改，则不能推送，必须先拉取；

拉取下来后如果进入冲突状态，则按照“分支冲突解决”操作解决即可。

举例：

甲和乙都从同一个远程仓库进行拉取，甲修改完后进行push，此时远程仓库得到更新，后来乙做完修改想要push就会失败。乙必须在远程库最新版上做修改才行，所以乙应先拉取下来，解决冲突，然后再push

克隆

命令：git clone [url]

效果：

1.完整地把远程库下载到本地

2.创建origin远程地址别名

3.初始化本地库

拉取

pull = fetch + merge

git pull [远程库地址别名][远程分支名] //抓取并合并

git fetch [远程库地址别名][远程分支名] //只是抓取，并不会修改工作区的文件内容

git merge [远程库地址别名/远程分支名] //合并则本地会有新内容

注：抓取之后若想要在不合并的情况下先检查一下抓取到本地的内容，

则可以先git checkout origin/master，再读取文件内容 cat [文件名]。

返回到本地库检查，git checkout master