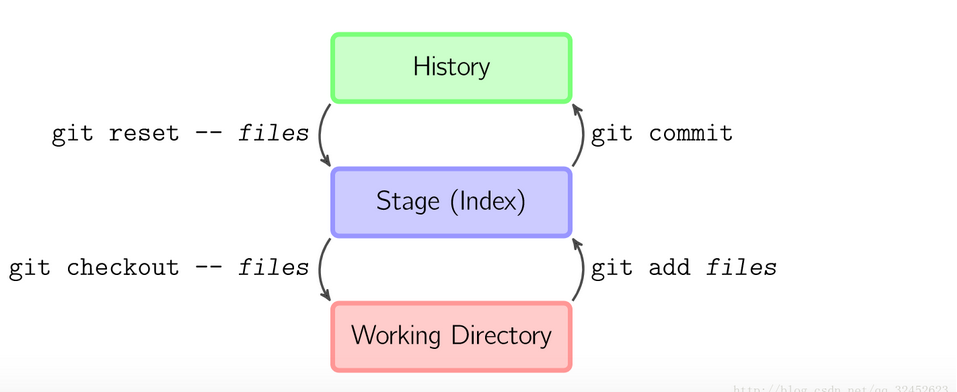
1. 初始化本地git仓库

git init

git的本地版本管理的一些概念：

|  |  |
| --- | --- |
| 名称 | 说明 |
| 工作区（work Directory） | 直接编辑的文件部分 |
| 暂存区（Staged Snapshot） | 文件执行 git add . 后存的地方 |
| 版本库区（Commit History） | 文件执行 git commit . 后存的地方 |



git config - -global user.name ‘zouyun’ ：配置用户名

git config - -gloabal user.email [xxx@163.com](mailto:xxx@163.com) ：配置邮箱

2. 添加文件到暂存区

git add 文件名 ：添加单个文件到暂存区，注意文件名要包含文件的类型，比如我有一个“中文“的txt文件，则上传命令应该写为：git add 中文.txt

git add . ：监控工作区的状态树，把工作时的所有变化提交到暂存区。包括文件内容修改(modified)以及新文件(new)，但不包括被删除的文件。

git add –u ：是git add –update的缩写，仅监控已经被add的文件(即tracked file)，将被修改的文件提交到暂存区。不会提交新文件(untracked file)。

git add –A ：是git add –all的缩写，是git add . 和git add –u 的综合。

总结：git add -A 提交所有变化

git add -u 提交被修改(modified)和被删除(deleted)文件，不包括新文件(new)

git add . 提交新文件(new)和被修改(modified)文件，不包括被删除(deleted)文件

3. 查看暂存区内容

git ls-files ： 查看暂存区中文件信息

一些参数：

- -cached (-c) 查看暂存区中文件，git ls-files 命令默认是此命令

- -midified （-m）查看修改的文件

- -delete （-d）查看删除过的文件

- -other （-o）查看没有被git跟踪的文件

- -stage (-s) 显示mode以及文件对应的Blob对象，进而我们可以获取暂存区中对应文件里面的内容。

这些参数的写法为：

比如git ls-file - -cached 或者git ls-files -c

4. 删除已经add到暂存区，但还未commit到本地版本库的文件

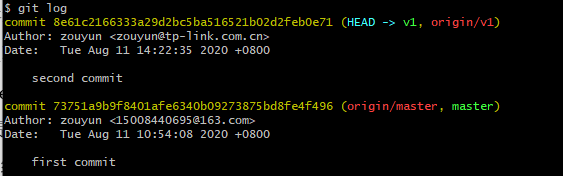
git rm - -cached 文件名（注意文件名要加类型，比如删除暂存区的名为master的txt文件，则此处文件名应该写为master.txt） ： 删除暂存区的master.txt，但不删除工作区的master.txt。

5. 将暂存区的改动提交到本地版本库

每次使用git commit 命令我们都会在本地版本库生成一个40位的哈希值（commit-id）

commit-id在版本回退的时候是非常有用的，它相当于一个快照,可以在未来的任何时候通过与git reset的组合命令回到这里。

查看commit的具体信息：git log



git commit –m “message” ： 将暂存区改动提交到版本库，message为说明这次提交的简短信息。message较长时，可以使用如下格式：

git commit -m ‘

message1

message2

message3

’

git commit - -amend ：在上一次提交的基础上追加修改提交，注意这不是新的提交，比如先前有一个first commit，修改后用git commit - -amend提交表示追加修改提交，仍然只有一次提交，它的使用如下：

①用git log 查看commit历史



②用git add –A 添加后，输入git commit - -amend 进行追加提交。

这时bash里出现如下内容：



③可用键盘上下键转到描述所在的那一行，然后进行修改：



④修改完成后，按下 Esc键退出编辑模式，在键入 :wq 回车退出并保存修改，完成提交。再git log 看一下提交日志：



注意：git commit - - amend 会改变commit id

6. 分支基本操作

git remote ：列出已经存在的远程仓库

git remote –v ：列出详细信息，在每一个名字后面列出其远程url，此时， -v 选项(译注:此为 –verbose 的简写,取首字母),显示对应的克隆地址

git remote add origin url ：添加一个远程仓库，语法为git remote add [<options>] <name> <url>

git branch ：列出本地分支

git branch 分支名（比如v1） ： 创建名为v1的本地分支

ls ：查看当前本地分支下的所有文件

git checkout 分支名（比如v1） ：切换当前本地分支到本地分支v1

git branch –d 分支名（比如v1） ：删除本地分支v1

注意：删除本地分支时可能出现以下几种问题

①error:The branch ‘testing’ is not fully merged

解决：这是因为删除的分支包含了还未合并的工作。解决办法是强制删除它，用参数“-D”表示。所以删除命令就是git branch -D testing

②error:Cannnot delete branch ‘testing’ checked out at ‘D:/practice/project’

解决：错误原因很有可能是你正处于该分支上，然后尝试删除该分支是不被允许的。和Windows系统下删除打开的文件道理一样。解决办法是只要切换分支到其他任意分支上，然后进行删除即可

③error:Cannot delete the branch ‘testing’ which you are currently on

同②

④error:Branch ‘testing’ not found

出现这种错误的原因是没有找到testing分支，但是你用git branch -a命令，又可以看到它们。从提示的原因看是没有找到testing分支，因此为了准确起见，我们可以从./git/refs/HEADS/文件夹查找，看是否有testing，如果有就把这个文件删除。再看.git/packed-refs文件，搜索testing，如果有，就把那一行删除。

git merge 分支名（比如v1） : 将本地分支v1上的内容合并到本地主分支（master）上。

例如：master分支上内容如下：

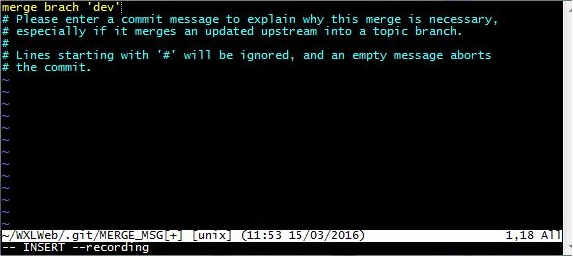


v1分支上内容如下：



执行git merge v1后，master内容如下： 

解决git中退出只读界面的问题



比如这种只读界面

git 在pull或者合并分支的时候有时会遇到这个界面。可以不管(直接下面3,4步)，如果要输入解释的话就需要:

按键盘字母 i 进入insert模式

修改最上面那行黄色合并信息,可以不修改

按键盘左上角"Esc"

输入":wq",注意是冒号+wq,按回车键即可

git push origin - -delete 远程分支名（比如v1） ：删除远程分支v1 （origin是你的远程仓库在你本地的别名）。

git push origin 本地分支名（比如v1）:远程分支名（比如v2） ：将本地分支v1的内容提交到远程分支v2中。

新建远程分支：

①新建一个本地分支

git branch v1

②将新建的本地分支push到远程仓库

git push origin v1:v1

7. 解决gerrit提交代码出现的missing-ChangeId的错误

原因：项目仓库.git/hooks目录下，commit-msg文件缺失。

解决方法：出现该错误时，bash会有如下提示。



在bash中依次输入：

# 对应于你的bash提示中remote后跟的部分

gitdir=$(git rev-parse --git-dir);

scp -p -P 29418 xxxxx@xxxx:hooks/commit-msg ${gitdir}/hooks/

# 再提交一次即可生成change-id

git commit –amend（注意一定要用追加提交，如果使用git commit –m 的形式，仍然会显示miss changid）

8. git clone url ：拷贝一个远程git仓库到本地，url为git仓库的url地址

git status ：用以查看上次提交过后是否有修改（git status –s获得简短信息）

git diff ：显示已写入缓存和已修改但尚未写入缓存的改动的区别（对比工作区和暂存区文件的区别）

git diff ： 查看尚未缓存的改动

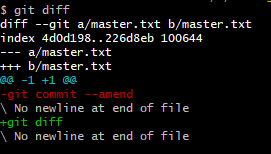
git diff - -cached ：查看已缓存的改动。

git diff HEAD ：查看已缓存和未缓存的所有改动

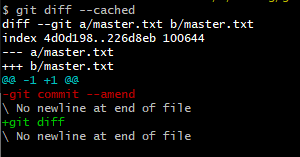
git diff - -stat ：显示摘要而非整个diff

git reset HEAD ：取消已缓存的改动。

比如有改动如下：

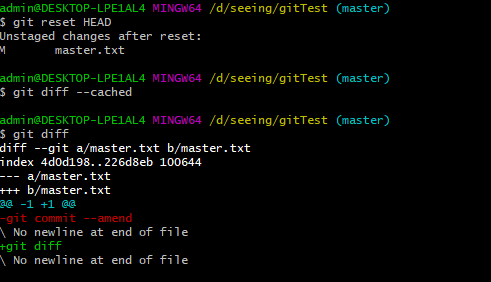


提交缓存区后，用git diff - -cached有下图：





说明改动存入缓存区，现在用git reset HEAD，有



说明被提交到缓存区的改动被取消了。

git reset HEAD 文件名 ：将某文件提交到缓存区的改动取消了。

9. 查看提交历史

git log ：查看历史提交记录

git log - -oneline ：查看历史提交记录的简洁版本

git log - -graph ：查看历史记录中合时进行了分支、合并。

git log - -reverse ：逆向显示日志。

10. git标签

git tag –a v1.0 ：创建带注解的标签（v1.0为注解）

git tag –a v1.0 85fc7e7 ：给commit id为85fc7e7的提交打上含注解的标签

11. git ignore的使用

①建立 .gitignore文件

有两种方式：

a、在windows下打开命令行：echo>.gitignore

b、在文件夹下git bash，输入touch .gitignore

②忽略规则

将.gitignore打开编辑，忽略规则如下

注释：#该行是注释

文件或文件夹filename：忽略掉所有名称为filename的文件和目录，filename.html忽略掉该文件

文件后缀\*.html：忽略所有后缀为html的文件，\*为通配符，?代表一个字符，[abc]代表可选文字范围

例外文件!except.html：忽略掉except.html这个文件带反斜杠/的名称all/ 表示忽略全部

注意事项：

.gitignore无效的情况

问题：在添加文件时，对其忽略但是在git status查看仍然存在

原因：在git库中已经存在过此文件（以前push过），而.gitignore还没有对加入到版本管理的文件起作用

解决办法：在git库中删除该文件，并提交

12. 回退到某次提交

①先用git log 查看自己的提交历史

②用git reset –hard commitid ：-hard为可选参数表示强制提交，commitid为你想要回退到的commit的id值。

注意：gerrit中遇到的删除上次文件，提交新文件后，gerrit上看到仍然有以前提交的文件的问题，可以用回退到最初的提交来解决。