**git版本控制工具命令总结**

分布式系统架构 2018-10-20 08:01:00

大家好，我是瓜哥。当前很多公司都用git对项目进行版本控制管理。相比svn,git功能更强大。GIT不仅仅是个版本控制系统，它也是个内容管理系统(CMS),工作管理系统等。如果你是一个具有使用SVN背景的人，你需要做一定的思想转换，来适应GIT提供的一些概念和特征。下面本人就git常用命令进行总结，供大家参考。

**1、git中分支直接代码差异比较**

**Git** diff branch1 branch2 --stat //显示出所有有差异的文件列表

**Git** diff branch1 branch2 文件名(带路径) //显示指定文件的详细差异

**Git** diff branch1 branch2 //显示出所有有差异的文件的详细差异

git diff DEV\_20170308 DEV\_20170422 \*.js //比较有有差异的文件后缀名为js的文件

**Git** diff // 可以产看当前没有add 的内容修改

**Git** diff --cached // 查看已经add 没有commit 的改动 使用

**2、git命令**

--从远程分支同步代码到本地仓库（commit代码之前养成先同步远程分支代码到本地的习惯）

git pull origin DEV20170308

--本地仓库提交至远程分支

git push origin HEAD:refs/for/DEV\_20170308

--同步远程分支代码（不会覆盖本地代码，保险起见做好备份）

git reset origin/DEV\_20170308

--同步远程分支代码（慎用：会覆盖掉本地已经修改的代码）

git reset --hard origin/DEV\_20170308

**3、合并DEV\_20170422分支代码到当前分支**

git checkout DEV\_20170422 /d/svn-project/gerrit-project-workspace/cubc-web/src/main/java/com/deppon/cubc/web/common/util

**4、举例说明，需要将bill 0308代码合并到0513上**

1 当下bill代码，然后git checkout DEV\_20170513

2. git merge origin/DEV\_20170308 如果没有冲突，

3. git commit –amend

4. git push origin HEAD:refs/for/DEV\_20170513 如果有冲突 3.解决冲突 4 git add . (走正常提交代码的步骤)

**5、查看到未传送到远程代码库的提交描述/说明**

git cherry -v

**6、--同步远程分支代码（不会覆盖本地代码，保险起见做好备份）**

git reset origin/DEV\_20170308

--同步远程分支代码（慎用：会覆盖掉本地已经修改的代码）

git reset --hard origin/DEV\_20170308

**7、撤销本地所有修改并还原到修改前状态(部分文件)：**

git checkout DEV\_20170513 pom.xml test.txt

**8、删除文件夹**

\* cd /usr/local/bin 切换路径

\* sudo rm -rf /usr/local/bin/tnpm 删除tnpm文件夹

**9、创建并切换到本地分支**

\* git checkout -b 【本地分支名】 origin/【远程分支名】



**10、删除.git 重新初始化git（delete the old .git history and initialize new history）**

$ rm -rf .git

$ git init

**11、检出命令**

\* git clone -b 【本地分支名】【远程分支名】

**12、合并分支：**

\* git branch 查看本地分支

\* git branch -a 查看分支详情（本地分支及远程分支）

\* git checkout 目的分支（如master）

\* git merge 待合并分支（如branch1）【即将branch1分支合并到master】

\* git push origin master 将合并后的代码推送到远程分支

**13、回滚版本**

\* git log 查看提交版本

\* git revert 版本号 将本地库回滚到该版本号之前的版本（即取消该版本及之后的提交）

\* 解决冲突

\* git add . && git commit -m “” && git push 重新提交版本之前的代码

**14、删除分支**

\* git branch -D br 删除本地分支

\* git push origin :br (origin 后面有空格) 删除远程分支

**15、rebase的冲突解决**

rebase的冲突解决过程，就是解决每个应用补丁冲突的过程。

解决完一个补丁应用的冲突后，执行下面命令标记冲突已解决（也就是把修改内容加入缓存）：

git add -u

**注：-u 表示把所有已track的文件的新的修改加入缓存，但不加入新的文件。**

然后执行下面命令继续rebase：

git rebase --**continue**

有冲突继续解决，重复这这些步骤，直到rebase完成。

如果中间遇到某个补丁不需要应用，可以用下面命令忽略：

git rebase --skip

如果想回到rebase执行之前的状态，可以执行：

git rebase --abort

注：rebase之后，不需要执行commit，也不存在新的修改需要提交，都是git自动完成。

**16、git工作流程**

