**Git常用操作指令**

## 远程仓库上创建新的分支

1. 新建本地分支

例如：git checkout -b git\_note

1. 将本地分支push到远程仓库

例如：git push origin git\_note

## 删除远程分支

git push origin :git\_note

分支名前的冒号代表删除

## 删除本地分支

git branch –d 分支名

## 查看远程分支

git branch –r

## patch生成和打入

### patch生成

当前分支所有超前master的提交：  
git format-patch -M master  
某次提交以后的所有patch:  
git format-patch 4e16 --4e16指的是commit名  
从根到指定提交的所有patch:  
git format-patch --root 4e16  
某两次提交之间的所有patch:  
git format-patch 365a..4e16 --365a和4e16分别对应两次提交的名称  
某次提交（含）之前的几次提交：  
git format-patch –n 07fe --n指patch数，07fe对应提交的名称  
故，单次提交即为：  
git format-patch -1 07fe  
git format-patch生成的补丁文件默认从1开始顺序编号，并使用对应提交信息中的第一行作为文件名。如果使用了-- numbered-files选项，则文件名只有编号，不包含提交信息；如果指定了--stdout选项，可指定输出位置，如当所有patch输出到一个文件；可指定-o <dir>指定patch的存放目录；

### 应用patch

先检查patch文件：git apply --stat newpatch.patch  
检查能否应用成功：git apply --check newpatch.patch  
打补丁：git am --signoff < newpatch.patch

(使用-s或--signoff选项，可以commit信息中加入Signed-off-by信息)

如果应用patch出现问题：

比如，一个典型的git am失败，可能是这样的：

$ git am PATCH

Applying: PACTH DESCRIPTION

error: patch failed: file.c:137

error: file.c: patch does not apply

error: patch failed: Makefile:24

error: libavfilter/Makefile: patch does not apply

Patch failed at 0001 PATCH DESCRIPTION

When you have resolved this problem run "git am --resolved".

If you would prefer to skip this patch, instead run "git am --skip".

To restore the original branch and stop patching run "git am --abort".

正如你所见，如果冲突发生，git只是输出上述信息，然后就停下来。一个小冲突会导致整个patch都不会被集成。

处理这种问题的最简单方法是先使用 git am --abort，然后手动的添加此patch, patch -p1 < PATCH，手动解决掉代码冲突，最后使用 git commit -a 提交代码。但是这样做有个问题就是你会失去PATCH中原本包含的commit信息（比如From，Date，Subject，Signed-off-by等）。应该有一种更聪明的方法。

在 .git/rebase-apply 目录下，存放着相应的补丁文件，名字是“0001” （在更新的git版本中，存放补丁文件的目录名有所改变，这里使用的git版本是 1.7.4.1）。

事实上，你可以使用 git apply 命令打patch（git apply 是git中的patch命令）。如同使用 patch -p1 命令时一样，然后手动解决代码冲突（检视生成的 .rej 文件，与冲突文件比较，修改冲突内容，并最终把文件加入到index中）：

$ git apply PATCH --reject

$ edit edit edit

（译注：根据.rej文件手动解决所有冲突）

$ git add FIXED\_FILES

$ git am --resolved

就这么简单！

想多一些解释，好吧。git am 并不改变index，你需要使用 git apply --reject 打patch（保存在 .git/rebase-apply），手动解决代码冲突，（译注：使用 git status 列出所有涉及文件），把所有文件（不仅仅是引起冲突的文件）添加到（git add）index，最后告诉 git am 你已经解决（--resolved）了问题。这样做的好处是你不需要重新编辑commit信息。而且，如果你正在打的是一系列patch（就是说你在打的是多个patch，比如 git am \*.patch）你不需要使用 git am --abort，然后又 git am。

## Git push 常用命令

git push <远程主机名> <本地分支名>:<远程主机分支名>

这个是push的完整写法，将本地分支上传到远程分支，例如：

git push origin dev:dev

git push <远程主机名> <本地分支名>

如果省略了<远程主机分支名> 即：

git push dev

则git会push到远程分支的同名本地分支，即和

git push dev:dev

等价。如果远程分支dev不存在则会创建dev分支。

git push <远程主机名> :<本地分支名>

如果省略<本地分支名> 即：

git push :dev

则git会删除远程主机上的dev分支，即用一个空分支更新deb分支，相当于删除dev分支，和

git push origin --delete dev

等价。

git push <远程主机名>

如果当前分支和远程分支存在追踪关系，则本地分支和远程分支都可以忽略。

git push

如果当前分支只有一个远程分支，那么远程主机也可以省略，可以使用