【TIP】在我们的[《Windows 下 Git 配置与使用指南》](http://wangcongming.info/2010/07/31/git-%E7%B3%BB%E5%88%97%E4%B9%8B%E4%B8%89%EF%BC%9Awindows-%E4%B8%8B-git-%E9%85%8D%E7%BD%AE%E4%B8%8E%E4%BD%BF%E7%94%A8%E6%8C%87%E5%8D%97/) 中，有介绍大家使用 $ git go 命令。其实，这并非 Git 的原生命令，它是我们自定义的一个 alias（别名），由 $git add、$git commit、$git push 和 $git pull 四个命令组合而成。待熟悉之后，你可以直接使用这些原生命令，或者自定义更适合自己的 alias。

**add**

添加新文件到 Git 代码仓库的索引中

$ git add filename

**mv**

移动或重命名文件

$ git mv old-filename new-filename

**rm**

从工作目录和 Git 代码索引中删除文件

$ git rm filename

**status**

查看目前工作目录的代码状态，自上次提交以来的添加、修改和删除等

$ git status

**diff**

查看自上次提交以来，本地代码改动的具体情况

$ git diff

**commit**

提交修改的代码（只是提交到本地的代码库，不会推送到服务器）

$ git commit -am '修改说明'

如果觉得刚提交的“修改说明”写得不够好，可输入以下命令调整

$ git commit --amend

**push**

将自上次 push 以来的，本地历次 commit，推送到服务器

结合我们的实际，应该这样写：

$ git push origin master:your-id

其中，master 是本地的分支名；your-id 填你在服务器上的 id，服务器的版本库里会有以你的 id 为名称的分支。

**pull**

将别人推送到服务器的代码，拉到你的机器里

$ git pull

**log**

查看修改记录，含作者、时间、修改说明等

$ git log

**show**

显示具体的代码改动情况

显示最后一次 commit 修改的内容：

$ git show

显示指定 commit 修改的内容：

【TIP】git log 命令中，每条 commit 会有一长长的字符串，此即 commid id，取其前面五六位即可。

$ git show commit-id

**branch**

分支管理

列出所有分支（当前所在分支前会有“\*”号）：

$ git branch

新建分支：

$ git branch 新分支名

删除分支：

$ git branch -d 欲删除的分支名

【注意！】不要把 ‘-d’ 写成了 ‘-D’，危险！

* -d：要求：被删除分支的所有修改，已经合并到当前分支；
* -D：直接删除，未合并的代码，将被丢弃！

**checkout**

恢复某个已修改的文件（撤销未提交的修改）：

$ git checkout file-name

切换到另外的分支，进行开发：

$ git checkout branch-name

【注意！】该命令可能伴随大量的文件增删/修改。Windows 下，改动已被占用的文件可能会被拒绝，导致版本库出现严重问题。如果确实要这样做，安全起见，最好先注销一次。

**merge**

合并指定分支到当前分支：

$ git merge branch-name

**revert**

还原已提交的修改（已经提交过的修改，可以反悔～）

还原最近一次提交的修改：

$ git revert HEAD

还原指定版本的修改：

$ git revert commit-id

**stash**

先将未提交的修改暂存起来，接着清除所有改动，使之与没修改时一样。

若你正在开发功能 A，又需立即去开发功能 B。A 的代码正改到一半，未认真整理，你不想立即提交。此时……请呼叫 stash ～。

它会使你所有未提交的修改瞬间不见了：

$ git stash

它会使刚刚不见了的修改，瞬间又回来了：

$ git stash pop

【TIP】以上命令皆有更多参数，另有一些 Git 命令我们此处没有介绍。但是，这已足令你使用 Git 时游刃有余，你会觉得，Git 简直是一件神器！:-)

【TIP】’$ git help’ 与 ‘$ git help 命令名’ 会在你需要的时候，无私地帮助你。:-)

**附：git push 失败的解决办法**

假设执行操作：

1. 修改代码

2. git commit

3. git push

此时 push 失败（错误提示：! [rejected] master -> master (non-fast-forward) ）

解决办法：

$ git pull

若成功，则：

$ git push origin master:your-id

完事。

若失败（提示：CONFLICT (content): Merge conflict in 文件名），则：

冲突的文件会有类似下面的代码块：

<<<<HEAD

你修改的代码

============

其他人修改的代码

>>>>>commit id of others'

考虑你和他人对代码的修改，更新成合适的内容，并删除 <<<、===、>>> 3行标记符号，保存文件。

$ git commit -am "resolve conflict"

$ git push origin master:your-id

更详细的说明，可以阅读 $git push –help 该文档的 NOTE ABOUT FAST-FORWARDS 一节。