**1) 远程仓库相关命令**

检出仓库：$ git clone git://github.com/jquery/jquery.git  
查看远程仓库：$ git remote -v  
添加远程仓库：$ git remote add [name] [url]  
删除远程仓库：$ git remote rm [name]  
修改远程仓库：$ git remote set-url --push[name][newUrl]  
拉取远程仓库：$ git pull [remoteName] [localBranchName]  
推送远程仓库：$ git push [remoteName] [localBranchName]

**2）分支(branch)操作相关命令**

查看本地分支：$ git branch  
查看远程分支：$ git branch -r  
创建本地分支：$ git branch [name] ----注意新分支创建后不会自动切换为当前分支  
切换分支：$ git checkout [name]  
创建新分支并立即切换到新分支：$ git checkout -b [name]  
删除分支：$ git branch -d [name] ---- -d选项只能删除已经参与了合并的分支，对于未有合并的分支是无法删除的。如果想强制删除一个分支，可以使用-D选项  
合并分支：$ git merge [name] ----将名称为[name]的分支与当前分支合并  
创建远程分支(本地分支push到远程)：$ git push origin [name]  
删除远程分支：$ git push origin :heads/[name]  
我从master分支创建了一个issue5560分支，做了一些修改后，使用git push origin master提交，但是显示的结果却是'Everything up-to-date'，发生问题的原因是git push origin master 在没有track远程分支的本地分支中默认提交的master分支，因为master分支默认指向了origin master 分支，这里要使用git push origin issue5560：master 就可以把issue5560推送到远程的master分支了。  
如果想把本地的某个分支test提交到远程仓库，并作为远程仓库的master分支，或者作为另外一个名叫test的分支，那么可以这么做。

$ git push origin test:master // 提交本地test分支作为远程的master分支 //好像只写这一句，远程的github就会自动创建一个test分支  
$ git push origin test:test // 提交本地test分支作为远程的test分支

如果想删除远程的分支呢？类似于上面，如果:左边的分支为空，那么将删除:右边的远程的分支。

$ git push origin :test // 刚提交到远程的test将被删除，但是本地还会保存的，不用担心

**3）版本(tag)操作相关命令**

查看版本：$ git tag  
创建版本：$ git tag [name]  
删除版本：$ git tag -d [name]  
查看远程版本：$ git tag -r  
创建远程版本(本地版本push到远程)：$ git push origin [name]  
删除远程版本：$ git push origin :refs/tags/[name]  
推送所有tag: git push origin --tags

**tag迁出branch：**  
git checkout tagname  
git checkout -b new\_branch\_name

**4）git stash暂存相关命令**

git stash: 备份当前的工作区的内容，从最近的一次提交中读取相关内容，让工作区保证和上次提交的内容一致。同时，将当前的工作区内容保存到Git栈中。  
git stash pop: 从Git栈中读取最近一次保存的内容，恢复工作区的相关内容。由于可能存在多个Stash的内容，所以用栈来管理，pop会从最近的一个stash中读取内容并恢复。  
git stash list: 显示Git栈内的所有备份，可以利用这个列表来决定从那个地方恢复。  
git stash clear: 清空Git栈。此时使用gitg等图形化工具会发现，原来stash的哪些节点都消失了。

**5）git rebase 使用**

当前分支问题develop，从master拉取最新代码  
git pull --rebase origin master 或者 git rebase develop  
先将develop分支的代码checkout出来，作为工作目录  
然后将master分支从develop分支创建起的所有改变的补丁，依次打上。如果打补丁的过程没问题，rebase就搞定了  
如果打补丁的时候出现了问题，就会提示你处理冲突。处理好了，可以运行git rebase –continue继续直到完成  
如果你不想处理，你还是有两个选择，一个是放弃rebase过程（运行git rebase –abort），另一个是直接用test分支的取代当前分支的（git rebase –skip）。

**6） 工具**

  SourceTree & git bash