转眼间加入git的阵营已经快两年了，结识git，缘起github，2年前在寻找代码托管网站，当时还是用svn，起初使用google code，可是google的服务虽好，在天朝你懂得，后来发现了github，多亏了蒋鑫老师的《GotGitHub》将我带入github的大门，如果你是个github新手，那我强烈建议你阅读这篇文章，里面讲了很多东西。

起初的时候我是用github for windows这个客户端，在切换到多分支的时候被，自动转换换行符坑的不浅，后来越来阅读了《[git详解](http://www.open-open.com/lib/view/open1328069609436.html)》系列文章，对git的了解深入了一步，并开始转到命令行上来，如今我在github上开源了60几个库，借助git，可自由在这些项目之间穿梭，同时还维护了github家园的微博和Q群(193091696)，如果你想获取关于git和github的最新消息可以关注微博，如果你有什么疑问或者问题，欢迎加群一起讨论。

这篇文章记录个人常用的一些命令，和记不住的一些命令。

**安装**

在 Windows 上安装 Git 同样轻松，有个叫做 msysGit 的项目提供了安装包：



|  |  |
| --- | --- |
| 1 | http://msysgit.github.io/ |

完成安装之后，就可以使用命令行的 git 工具（已经自带了 ssh 客户端）了，另外还有一个图形界面的 Git 项目管理工具。

**配置**

首先是配置帐号信息



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | git config --global user.name yanhaijing  git config --global user.email yanhaijing@yeah.net    git config --list#查看配置的信息    git help config#获取帮助信息 |

配置自动换行（自动转换坑太大）



|  |  |
| --- | --- |
| 1 | git config --global core.autocrlf input #提交到git是自动将换行符转换为lf |

配置密钥



|  |  |
| --- | --- |
| 1  2  3 | ssh-keygen -t rsa -C yanhaijing@yeah.net #生成密钥    ssh -T git@github.com #测试是否成功 |

**新建仓库**



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | git init#初始化  git status#获取状态  git add file#.或\*代表全部添加  git commit -m &quot;message&quot;#此处注意乱码  git remote add origin git@github.com:yanhaijing/test.git#添加源  git push -u origin master#push同事设置默认跟踪分支 |

**从现有仓库克隆**



|  |  |
| --- | --- |
| 1  2 | git clone git://github.com/yanhaijing/data.js.git  git clone git://github.com/schacon/grit.git mypro#克隆到自定义文件夹 |

**本地**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | git add \*#跟踪新文件    rm \*&amp;git rm \*#移除文件  git rm -f \*#移除文件  git rm --cached \*#取消跟踪  git mv file\_from file\_to#重命名跟踪文件    git log#查看提交记录    git commit#提交更新  git commit -m &#039;message&#039;  git commit -a#跳过使用暂存区域，把所有已经跟踪过的文件暂存起来一并提交  git commit --amend#修改最后一次提交    git reset HEAD \*#取消已经暂存的文件    git checkout -- file#取消对文件的修改（从暂存区去除file）  git checkout branch|tag|commit -- file\_name#从仓库取出file覆盖当前分支  git checkout -- .#从暂存区去除文件覆盖工作区 |

**分支**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | git branch#列出本地分支  git branch -r#列出远端分支  git branch -a#列出所有分支  git branch -v#查看各个分支最后一个提交对象的信息  git branch --merge#查看已经合并到当前分支的分支  git branch --no-merge#查看为合并到当前分支的分支    git branch test#新建test分支  git checkout test#切换到test分支  git checkout -b test#新建+切换到test分支  git checkout -b test dev#基于dev新建test分支，并切换    git branch -d test#删除test分支  git branch -D test#强制删除test分支    git merge test#将test分支合并到当前分支  git rebase master#将master分之上超前的提交，变基到当前分支 |

**远端**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | git fetch originname branchname#拉去远端上指定分支  git merge originname branchname#合并远端上指定分支  git push originname branchname#推送到远端上指定分支  git push originname localbranch:serverbranch#推送到远端上指定分支    git checkout -b test origin/dev#基于远端dev新建test分支    git push origin :server#删除远端分支 |

**源**

git是一个分布式代码管理工具，所以可以支持多个仓库，在git里，服务器上的仓库在本地称之为remote。

个人开发时，多源用的可能不多，但多源其实非常有用。



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | git remote add origin1 git@github.com:yanhaijing/data.js.git    git remote#显示全部源  git remote -v#显示全部源+详细信息    git remote rename origin1 origin2#重命名    git remote rm origin1#删除    git remote show origin1#查看指定源的全部信息 |

**标签**

当开发到一定阶段时，给程序打标签是非常棒的功能。



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | git tag#列出现有标签    git tag v0.1#新建标签  git tag -a v0.1 -m &#039;my version 1.4&#039;#新建带注释标签    git checkout tagname#切换到标签    git push origin v1.5#推送分支到源上  git push origin --tags#一次性推送所有分支    git tag -d v0.1#删除标签  git push origin :refs/tags/v0.1#删除远程标签 |

**总结**

啊哈！终于总结完了，以后不会的时候，再也不用到处去找了。

其实还有两个最有用的命令还未提到。



|  |  |
| --- | --- |
| 1  2 | git help \*#获取命令的帮助信息  git status#获取当前的状态，非常有用，因为git会提示接下来的能做的事情 |