Git学习笔记

Git-简明教程

1. 创建新仓库：创建新文件夹，打开，然后执行**git init** 以创建新的git仓库。
2. 检出仓库：执行如下命令创建一个本地仓库的克隆版本**git clone /path/**。如果是远程服务器上的仓库，你的命令会是这个样子**git clone username@host:/path/**。
3. 工作流：你的本地仓库由git维护的三棵“树”组成。第一个是你的**工作目录**，它持有实际文件；第二个是**暂存区（Index）**，它像个缓存区域，临时保存你的改动；最后是**HEAD**，它指向你最后一次提交的结果。
4. 添加和提交：你可以提出更改（把它们添加到暂存区），使用如下命令**git add <filename>**或者 **git add ***，这是git基本工作流程的第一步；使用如下命令以实际提交改动**git commit -m "代码提交信息"**。现在，你的改动已经提交到了**HEAD**，但是还没到你的远程仓库。
5. 推送改动：你的改动现在已经在本地仓库的**HEAD**中了。执行如下命令将这些改动提交到远端仓库**git push origin master**。 可以把 *master*换成你想要推送的任何分支。如果你还没有克隆现有仓库，并欲将你的仓库连接到某个远程服务器，你可以使用如下命令添加**git remote add origin <server>**，如此你就能够将你的改动推送到所添加的服务器上去了。
6. 分支：分支是用来将特性开发绝缘开来。在你创建仓库的时候，**master**是“默认的分支”。在其他分支上进行开发，完成后再将它们合并到主分支上。创建一个叫做“feature_x”的分支，并切换过去**git checkout -b feature_x，切换回主分支git checkout master**，再把新建的分支删除(*删除分支不能是现在所在分支，即必须切换到其他分支之后，再删除该分支*） **git branch -d feature_x** 。 除非你将分支推送到远端仓库，不然该分支就是*不为他人所见的*： **git push origin <branch>**。
7. 更新与合并：要更新你的本地仓库至最新改动，执行**git pull**。以在你的工作目录中 *获取（fetch)* 并 *合并（merge)* 远端的改动。要合并其他分支到你的当前分支（例如master)，执行 **git merge <branch>** 。在这两种情况下，git都会尝试去自动合并改动。遗憾的是，这可能并非每次都成功，并可能出现*冲突（conflicts)*。这时候就需要你修改这些文件来手动合并这些*冲突（conflicts)*。改完之后，你需要执行如下命令来将它们标记为合并成功 **git add <filename>**。在合并改动之前，你可以使用如下命令预览差异 **git diff <source_branch> <target_branch>**。
8. 标签：为软件发布创建标签是推荐的。这个概念早已存在，在**SVN**中也有。你可以执行如下命令创建一个叫做1.0.0的标签 **git tag 1.00 1b2e1d63ff**。 *1b2e1d63ff*是你想要标记的提交ID的前10位字符。可以使用下列命令获取提交ID **git log** 。你也可以使用少一点的提交ID前几位，只要它的指向具有唯一性。
9. 替换本地改动：加入你操作失误（当然，这最好永远不要发生），你可以使用如下命令替换掉本地改动 **git checkout -- <filename>** 。**此命令会使用HEAD**中的最新内容替换掉你的工作目录中的文件。已添加到暂存区的改动以及新文件都不会受到影响。假如你想丢弃你在本地的所有改动与提交，可以到服务器上获取最新的版本历史，并将你本地主分支指向它 **git fetch origin** and **git reset --hard origin/master**。
10. 实用小贴士：内建的图形化git **gitk** 。彩色的git输出 **git config color.ui true** 。显示历史记录时，每个提交的信息只显示一行 **git config format.pretty oneline**。交互式添加文件到暂存区 **git add -i**。
11. 图形化客户端：
    1. GitX(L)(OSX,开源软件）
    2. Tower(OSX)
    3. Source Tree(OSX，免费）
    4. GitHub for Mac(OSX，免费）
    5. GitBox(OSX，App Store)
12. git config工具：专门用来配置或读取相应的工作环境变量。这些环境变量，决定了Git在各个环节的具体工作方式和行为。这些变量可以存放在一下三个不同的地方：
    1. /etc/gitconfig文件：系统中对所有用户都普遍使用的配置。若使用git config时用--system选项，读写的就是这个文件。
    2. ~/.gitconfig文件：用户目录下的配置文件只适用于该用户。若使用git config时用--global选项，读写的就是这个文件。
    3. 当前项目的Git目录中的配置文件（也就是工作目录中的.git/config文件）：这里的配置仅仅针对当前项目有效。每一个级别的配置都会覆盖上层的相同配置，所以.git/config里的配置会覆盖/etc/gitconfig中的同名变量。

代码：

```
git config -- global user.name "yang" %配置该用户的用户名%

git config --global user.email "yang@163.com" %配置该用户的电子邮箱地址%

git config --global core.editor emacs %设置Git默认的文本编辑器偏好

git config --global merge.tool vimdiff %设置Git默认的差异分析工具%

git config --list %查看配置信息%

git config user.name %查看某个变量的配置%

git --version %查看Git的版本号%

git init %以当前目录作为仓库进行初始化
```

`git init \<newrepo\> %制定目录作为仓库进行初始化`

`git add \<filename/filenames\> %将指定的文件加入到index区，即z暂存区`

`git commit -m "提交信息" %提交代码到仓库中，并注明提交的信息`

`git commit -a %省略了将代码加入暂存区的步骤，直接提交至仓库`

`git clone \<repo\> %从Git仓库拷贝项目`

`git clone \<repo\> \<directory\> %从Git仓库拷贝项目，并指定本地目录`

`git clone \[url\] %从指定的URL拷贝项目`

`git status %当前项目的状态`

`git diff` ： 执行git diff 来查看执行git status的结果的详细信息

`git diff %尚未缓存的改动`

`git diff --cached %查看已缓存的改动`

`git diff HEAD %查看已缓存的与未缓存的所有改动`

`git diff --stat %显示摘要而非整个diff`

`git reset HEAD %取消已缓存的内容`

`git rm file %从缓存区中移除，注意与上边的区别`

`git mv oldfilename newfilename %类似于git rm file，重命名`

`git branch \<branchname\> %创建分支`

`git checkout \<branchname\> %切换分支`

`git branch %列出分支`

`git merge \<branchname\> %合并分支`

`git log %查看历史提交记录`

`git log oneline %一行显示一条记录`

`git log --graph %以拓扑图的形式`

`git log --reverse %逆向显示`

`git log --author=yang --oneline -5 %查找指定用户的提交日志`

git log 命令还可以加入参数--since和--before,或者用--util和--after，来选择指定日期，还可以使用--no-merges选项隐藏合并提交

`git tag -a v1.0` 给最近一次提交打上'v1.0'标签，其中-a选项为创建一个带注解的标签

`git tag %查看所有的标签`

`git tag -a \<tagname\> -m "标签信息"`

`git tag -s \<tagname\> -m "PGP标签信息"`

`git remote add [shorname] [url] %添加远程仓库`

`ssh-keygen -t rsa -C "youremail@example.com" %产生SSH key`

`ssh -T git@github.com %验证key是否设置成功`

`git remote %查看远程仓库`

`git remote %显示远程仓库的实际链接地址`

`git fetch %提取远程仓库`

`git pull %从远程仓库提取数据并尝试合并到当前分支`

`git push [alias] [branch] %推送到远程仓库`

`git remote rm [别名] %删除远程仓库`