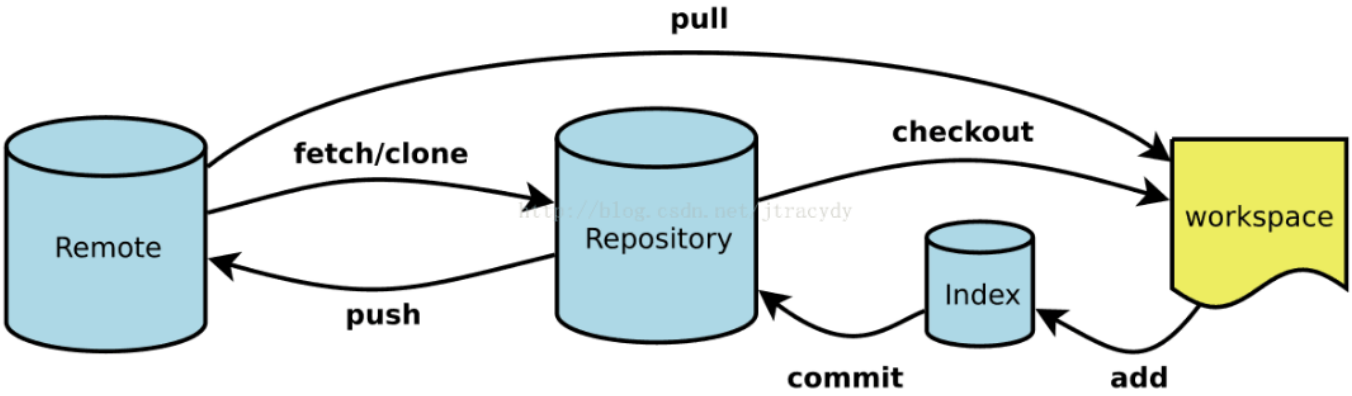
原 git命令-远程仓库拉取、本地仓库更新、工作空间提交等等

2017年04月22日 11:47:37 阅读数：15338

这个可以虽然不是自己写的，但是也不是转载的



Workspace：工作区

Index / Stage：暂存区

Repository：仓库区（或本地仓库）

Remote：远程仓库

一，新建代码库
1，在当前目录下新建一个git代码库
$ git init
2，新建一个目录将其初始化为git代码库
$ git init [project-name]
3，git clone 远程代码库
$ git clone [url]

二，配置
1，Git的设置文件为.gitconfig，它可以在用户主目录下（全局配置），也可以在项目目,录（项目配置）。
2， 显示当前的Git配置
$ git config--list
3，编辑git配置文件
$ git config -e[--global]
4，设置提交代码时的用户信息
$ git config[--global] user.name "[name]"
$ git config[--global] user.email "[email address]"

三、增加/删除文件
1，添加指定文件到暂存区
$ git add [file1][file2] ...
2， 添加指定目录到暂存区，包括子目录,
$ git add [dir]
3， 添加当前目录的所有文件到暂存区
$ git add .
4 添加每个变化前，都会要求确认， 对于同一个文件的多处变化，可以实现分次提交
$ git add -p
5，删除工作区文件，并且将这次删除放入暂存区
$ git rm [file1][file2] ...
6， 停止追踪指定文件，但该文件会保留在工作区
$ git rm --cached[file]

7， 改名文件，并且将这个改名放入暂存区
$ git mv[file-original] [file-renamed]

四、代码提交
1， 提交暂存区到仓库区,
$ git commit -m[message],
2， 提交暂存区的指定文件到仓库区
$ git commit[file1] [file2] ... -m [message],
3，提交工作区自上次commit之后的变化，直接到仓库区
$ git commit -a,
4， 提交时显示所有diff信息
$ git commit -v
5，使用一次新的commit，替代上一次提交，如果代码没有任何新变化，则用来改写上一次commit的提交信息
$ git commit--**amend** -m [message]
6， 重做上一次commit，并包括指定文件的新变化
$ git commit--**amend** [file1] [file2] ...

五、分支
1， 列出所有本地分支
$ git branch
2， 列出所有远程分支
$ git branch -r
3， 列出所有本地分支和远程分支
$ git branch -a
4， 新建一个分支，但依然停留在当前分支
$ git branch[branch-name]
5， 新建一个分支，并切换到该分支
$ git checkout -b[branch]
6， 新建一个分支，指向指定commit
$ git branch[branch] [commit]
7， 新建一个分支，与指定的远程分支建立追踪关系
$ git branch--track [branch] [remote-branch]
8， 切换到指定分支，并更新工作区
$ git checkout[branch-name],
9， 切换到上一个分支
$ git checkout -,
10， 建立追踪关系，在现有分支与指定的远程分支之间
$ git branch--set-**upstream** [branch] [remote-branch]
11， 合并指定分支到当前分支,
$ git merge[branch]
12， 选择一个commit，合并进当前分支,
$ git cherry-pick[commit]
13， 删除分支,
$ git branch -d[branch-name]
14， 删除远程分支,
$ git push origin--delete [branch-name]
$ git branch -dr[remote/branch]

六、标签,
1， 列出所有tag
$ git tag
2， 新建一个tag在当前commit,
$ git tag [tag]
3， 新建一个tag在指定commit,
$ git tag [tag][commit]
4， 删除本地tag,
$ git tag -d[tag]
5， 删除远程tag,
$ git push origin:refs/tags/[tagName]
6， 查看tag信息,

$ git show [tag]
7， 提交指定tag,
$ git push[remote] [tag]
8， 提交所有tag,
$ git push[remote] --tags
9， 新建一个分支，指向某个tag
$ git checkout -b[branch] [tag]

七、查看信息
1， 显示有变更的文件
$ git status
2， 显示当前分支的版本历史
$ git log
3， 显示commit历史，以及每次commit发生变更的文件
$ git log --stat
4， 搜索提交历史，根据关键词
$ git log -S[keyword]
5， 显示某个commit之后的所有变动，每个commit占据一行
$ git log [tag]HEAD --pretty=format:%s
6， 显示某个commit之后的所有变动，其"提交说明"必须符合搜索条件,
$ git log [tag]HEAD --grep feature
7， 显示某个文件的版本历史，包括文件改名
$ git log --follow[file]
$ git whatchanged[file]
8， 显示指定文件相关的每一次diff
$ git log -p[file]
9， 显示过去5次提交,
$ git log -5--pretty --oneline
10， 显示所有提交过的用户，按提交次数排序
$ git shortlog-sn
11， 显示指定文件是什么人在什么时间修改过,
$ git blame[file]
12， 显示暂存区和工作区的差异
$ git diff
13，显示暂存区和上一个commit的差异
$ git diff--cached [file]
14， 显示工作区与当前分支最新commit之间的差异
$ git diff HEAD
14， 显示两次提交之间的差异,
$ git diff[first-branch]...[second-branch]
15，显示今天你写了多少行代码,
$ git diff--shortstat "@{0 day ago}"
16， 显示某次提交的元数据和内容变化
$ git show[commit]
17， 显示某次提交发生变化的文件,
$ git show--name-only [commit]
18，显示某次提交时，某个文件的内容
$ git show[commit]:[filename]
19， 显示当前分支的最近几次提交
$ git reflog

八、远程同步
1， 下载远程仓库的所有变动
$ git fetch[remote]
2，显示所有远程仓库,
$ git remote -v
3， 显示某个远程仓库的信息,
$ git remote show[remote]
4， 增加一个新的远程仓库，并命名
$ git remote add[shortname] [url]

5，取回远程仓库的变化，并与本地分支合并
$ git pull[remote] [branch]
6， 上传本地指定分支到远程仓库
$ git push[remote] [branch]
7， 强行推送当前分支到远程仓库，即使有冲突
$ git push[remote] --force
8， 推送所有分支到远程仓库
$ git push[remote] --all

九、撤销
1， 恢复暂存区的指定文件到工作区,
$ git checkout[file]
2， 恢复某个commit的指定文件到暂存区和工作区,
$ git checkout[commit] [file]
3，恢复暂存区的所有文件到工作区,
$ git checkout .
4， 重置暂存区的指定文件，与上一次commit保持一致，但工作区不变
$ git reset[file]
5， 重置暂存区与工作区，与上一次commit保持一致
$ git reset--hard
6， 重置当前分支的指针为指定commit，同时重置暂存区，但工作区不变
$ git reset[commit]
7， 重置当前分支的HEAD为指定commit，同时重置暂存区和工作区，与指定commit一致
$ git reset--hard [commit]
8， 重置当前HEAD为指定commit，但保持暂存区和工作区不变
$ git reset--keep [commit]
9， 新建一个commit，用来撤销指定commit， 后者的所有变化都将被前者抵消，并且应用到当前分支
$ git revert[commit]
10， 暂时将未提交的变化移除，稍后再移入
$ git stash
$ git stash pop

十、其他
1， 生成一个可供发布的压缩包
$ git archive