空谷幽兰

小隐于野，大隐于市，神隐于网络

git常用命令整理

2017-07-10 | 2019-06-03 | 常用整理， 工具

记录一些git的常用操作

术语

三个基本区域

`Staged Snapshot：index，暂存区，索引`

`Working Directory：工作区`

`Commit History：commit 历史，版本库`

初始化/克隆

- `git init` 初始化仓库
- `git clone <git仓库地址/路径>` 克隆

修改和提交

查看文件状态/比较差异/加入暂存区

- `git status` 显示仓库状态
- `git status -s` 简要显示仓库的状态
- `git diff` 比较变动 默认比较工作区的变动
    - `git diff <branch1> <branch2>` 比较两个分支
    - `git diff <hash1> <hash2>` 比较两个指定hash
- `git rm <file>` 删除文件,并添加到暂存区
    - `-r <folder>` 删除文件夹,并添加到暂存区
    - `--cached` 不真正删除文件，只将删除操作添加到暂存区
- `git add` 将变动添加到暂存区
    - `git add <path>` 添加指定路径的所有变动到暂存区, 在仓库根路径下执行 `git add . == git add --all`
    - `git add -A` 添加所有变动到暂存区
    - `git add -u` 添加修改(modified)和删除(deleted)文件,不包括新(new)的文件
    - `git add -i` 交互式操作暂存区

提交

- `git commit -m 'log'` 提交暂存区的变动

- `git commit -am 'log'` 添加已经在版本库控制的文件变动到暂存区,并提交
- `git commit --amend` 修改最后一次commit (不会增加新的commit信息)

分支

分支基本操作

- `git branch` 查看本地分支
  - `-r` 查看远程分支
  - `-a` 查看所有分支
  - `-m <新分支名>` 修改分支名
  - `-d/-D <分支名>` 删除/强制删除 本地分支
- `git branch <分支名>` 创建新分支
- `git checkout <分支名|Tag名>` 切换到分支|Tag
  - `-b <新分支名>` 创建并切换到新分支

合并分支

- `git merge <分支名>` //把分支合并到本分支
  - `--squash` 不自动提交,用把分支的多个提交合成一个并添加到暂存区,需要手动commit。
  - `--on-ff` 可以快速合并的情况也添加合并日志,要配合 `-m` 使用
- `git rebase <分支名>` 衍和分支(分支衍合不会保留合并的日志，不留痕迹),相当于本地修改

提取commit

- `git cherry-pick <commit hash>` 将别的分支的一个commit拉到本分支

远程操作

源操作

- `git remote -v` 显示更详细的源信息
- `git remote add <origin name> <GIT URL>` 增加源
- `git remote set-url <origin name> <GIT URL>` 修改源
- `git remote remove <origin name>` 删除源

关联 推送

关联分支后 `pull` , `push` 等命令操作当前分支可省略 `<origin>/<brnach>` .

- `git branch --set-upstream-to <brnach> <origin>/<brnach>` 关联 本地与远程 分支
- `git push -u <origin> <brnach>` 推送并关联
- `git push -f` 强制用本地分支覆盖远程分支, 慎用

删除远程分支

- `git push origin :<brnach>` 删除本地分支后, 把空分支推送到远程分支，以删除远程端分支
- `git push origin --delete <brnach>` 删除远程分支git 1.7版本后支持

更新/同步

- `git fetch` 更新更程所有分支的信息,但不影响本地代码
  - `git fetch -all` 同步所有分支信息
  - `git fetch -p` 同步分支信息,分同步远程仓库分支,tag删除情况
- `git pull` 更新分支 `git pull == git fetch + git merge`
- `git pull --rebase` rebase方式合并

Tag

- `git tag <tagname>` 新建tag
- `git tag -a <tagname> -m '说明'` 新建带说明的完整tag
- `git tag -d <tagname>` 删除tag
- `git push --tags` 推送所有本地tag到远程仓库
- `git push origin --delete tag <tagname>` 删除远程端的tag 1.7
- `git tag --sort="version:refname"` 按版本号排序显示tag
- `git tag --sort="version:refname" -l '*.*.*'` 增加筛选

常规提交流程

- `git status`
- `git diff` 比较差异
- `git add 1.txt` 把文件添加到git的仓库管理系统中
- `git add <文件夹>` 把文件夹添加到git的仓库管理系统中
- `git add --all` 提交所有修改到暂存
- `git rm 1.txt` 把文件从git的仓库管理系统中移除
- `git rm -r myFolder` 把文件夹从git的仓库管理系统中移除
- `git commit -m '说明'` 提交到仓库(本地)
- `git push` 推送到仓库
- `git push <主机名> <分支名>` 推送到指定远程分支
- `git push -u <主机名> <分支名>` 推送并关联远程分支

简单提交流程

1. `git pull` 同步远程仓库到本地(非必须,如果远程有更新)
2. `git add --all && git commit -m '提交说明'` 添加所有修改并提交,
3. `git push` 推送到远程

日志

- `git log`
  - `-p` 提交的祥细内容
  - `-p <file>` 指定文件的提交日志
  - `-n` 查看最后n次提交的信息
  - `--oneline` 简要列出commit记录

- `git blame <file>` 以列表方式查看提交历史

撤消

- `git reset <commit id>` 清理暂存区，移动当分支的游标(HEAD)
- `git reset HEAD <file>...` 把暂存区的文件移出
- `git checkout <path>` 重新检出文件 (d)
- `git clean` 删除新增,还未add的文件
  - `-n` 预览要删除的文件和目录
  - `-f` 删除 文件，
  - `-df` 删除 文件 和 目录
- `git reset --hard` 还原所有文件修改到HEAD
- `git reset --hard <哈希值>` 回退到指定版本
- `git reset --hard HEAD^` 回退到上一个commit
- `git reset --hard HEAD~2` 放弃本地所有修改，回退到上上个commit
- `git reset --hard HEAD@{1}` 放弃本地所有修改，回退到HEAD前一次指向的commit
  - `git reflog` 显示 HEAD 移动路径
- `git reset --hard ORIG_HEAD` 放弃本地所有修改，回退到上一次(合并等)操作前的版本
- `git revert <commit id>` 用一个新commit撤销commit操作，此次操作之前的commit都会被保留

本地跟踪忽略

- `git update-index --assume-unchanged <path>` 忽略跟踪
- `git update-index --no-assume-unchanged <path>` 恢复跟踪

列出文件状态

- `git ls-files` 列出所有已缓存的文件
  - `-v` 用小写显示被 `assume-unchanged` 的文件

嵌套仓库

- `git submodule add <仓库地址> <路径>` 添加嵌套仓库
- `git submodule init` 初始化嵌套仓库
- `git submodule update` 更新嵌套仓库

保存当前状态

- `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。 可多次git stash
- `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。
- `git stash pop stash@{num}` 如果有多个工作现场，num是工作现场的编号。使用pop命令恢复的工作现场，其对应的stash 在队列中删除。

- `git stash apply stash@{num}` 除了不在stash队列中删除外其他和 `git stash pop` 完全一样。
- `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。

取消版本控制

- `git rm -r -n --cached <path>`

git设置

- `git config --global push.default current`
- `git config --global pull.default current`

git

mysql常用命令 fs-extra模块简要使用说明