tag系列

- git tag 查看所有标签
- git show v1.0 查看具体标签信息
- git tag -a v1.4 -m "my version 1.4" 附注标签
- git tag v1.4-l 轻量标签
- git tag -a v1.2 9fceb02 后期打标签
- git push origin v1.5 提交到git(共享标签)
- git push origin --tags 一次提交多个标签
- git tag -d v1.4-l 删除标签（本地）
- git push origin :refs/tags/v1.4-l （远程）

提交日志

- git log --pretty=oneline 查看提交历史
- git rebase -i [startpoint] [endpoint] 合并提交

branch

其他

- git add -i 打开一个交互式界面按需求添加文件
- git add -p 对同一个文件的多处变化分次提交
- git add -u 将本地的(修改, 删除, 没有新增)文件添加到暂存区
- git add .git add -A 添加当前目录的所有文件到暂存区
- git add [DirName] 添加指定目录到暂存区
- git add [FileName] ... 添加指定文件到暂存区
- git archive 生成一个可供发布的压缩包
- git blame [FileName] 显示指定文件是什么人在什么时间修改的信息
- git branch --set-upstream [Branch] [RemoteBranch] 建立追踪关系, 在现有分支与指定的远程分支之间
- git branch --track [Branch] [RemoteBranch] 新建一个分支, 与指定的远程分支建立追踪关系
- git branch -a 列出所有分支
- git branch -d [Branch] 强制删除分支
- git branch -m [OldName] [NewName] 本地分支重命名
- git branch -r 列出所有远程分支
- git branch [Branch] [Commit] 新建一个分支, 指向指定commit
- git branch [Branch] 新建一个分支, 但不切换
  git branch
  查看本地所有分支
  git checkout -
  切换到上一个分支
  git checkout --track origin/dev
  切换到远程dev分支
  git checkout -b [Branch] [tag]
  新建一个分支, 指向某个tag
  git checkout -b [Branch]
  新建一个分支, 并切换到该分支
  git checkout .
  恢复暂存区的所有文件到工作区
  git checkout [Branch]
  切换到指定分支, 并更新工作区
  git checkout [Commit] [File]
  恢复某个commit的指定文件到暂存区和工作区

git checkout [FileName]
恢复暂存区的指定文件到工作区

git checkout [BranchName]
切换分支

git cherry-pick [CommitID]
用于把另一个本地分支的commit修改应用到当前分支

git clone [URL]
克隆对应项目URL的代码到本地

git commit --amend -m [message]
代码没有任何新变化, 则改写上一次commit的提交信息

git commit --amend [file1] [file2] ...
重改上一次commit, 并包括指定文件的新变化

git commit -a -v
一般提交命令

git commit -a
提交当前repos的所有的改变

git commit -am [message]
提交工作区变化的文件直接到仓库区, 并添加提交描述信息

git commit -m [message]
提交暂存区到仓库区, 并添加提交描述信息

git commit -v
提交时显示所有diff信息

git commit [file1] [file2] ... -m [message]
提交暂存区的指定文件到仓库区, 并添加提交描述信息

git commit
提交

git config --global alias.ci commitgit config --global alias.co checkout
设置git命令的别名

git config --list
显示Git配置

git config -e [--global]
编辑Git配置文件

git config -l
查看当前配置

git config [--global] user.name "[userName]"git config [--global] user.email "[email]"
设置用户信息

git diff --cached [file]
显示暂存区和上一个commit的差异

git diff --cachedgit diff --staged
查看尚未提交的更新

git diff --shortstat "@{0 day ago}"
显示今天你写了多少行代码

git diff --staged
查看暂存起来的文件(stage)与并未提交(commit)的差别

git diff --stat
查看显示简略结果(文件列表)

git diff HEAD
显示工作区与当前分支最新commit之间的差异

git diff [first-branch]...[second-branch]
显示两次提交之间的差异

git diff
显示暂存区和工作区的差异

git fetch [remote]
下载远程仓库的所有变动

git fetch
从远程获取最新版本到本地, 不会自动merge

git init [dirName]
新建一个目录并初始化为Git本地仓库

git init
在当前目录创建一个Git本地仓库

git log --follow [file]git whatchanged [file]
显示某个文件的版本历史, 包括文件改名

git log --graph
显示何时出现了分支和合并等信息

git log --oneline
一行显示一条log

git log --stat
显示commit历史, 以及每次commit发生变更的文件

git log -3
查看前3次修改

git log -N --pretty --oneline
显示过去N次提交

git log -S [keyword]
搜索提交历史, 根据关键词

git log -p [file]
显示指定文件相关的每一次diff

git log -p
查看详细修改内容

git log [tag] HEAD --grep feature
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 --pretty=format:%s
显示某个commit之后的所有变动, 每个commit占据一行

git log
显示当前分支的版本历史

git ls-files
看已经被提交的

git merge --abort
尽量回退到merge前的状态(可能会失败)

git merge --no-ff <branch_name>
采用no fast forward的合并方式, 这种方式在合并的同时会生成一个新的commit

git merge --squash <branch_name>
将目标分支合并过来但不携带commit信息, 执行后最后需要提交一个commit

git merge [branch]
合并指定分支到当前分支

git mv [file-original] [file-renamed]
修改文件名, 并且将这个改名放入暂存区

git pull --rebase
暂存本地变更, 合并远程最新改动, 合并刚刚暂存的本地变更(不产生无用的merge的同步)

git pull [remoteName] [localBranchName]
拉取远程仓库

git pull origin --tags
合并远程仓库的tag到本地

git pull
从远程获取最新版本并merge到本地

git push --all origin
将所有本地分支都推送到origin主

git push -f origin
-f强推, 在远程主机产生一个"非直进式"的合并(non-fast-forward merge)

git push -u origin master
-u指定origin为默认主机, 后面就可以不加任何参数使用git push了

git push [remoteName] --delete [branch]git branch -dr [remote/branch]
删除远程分支

git push [remoteName] :refs/tags/[tagName]
删除远程tag

git push [remoteName] [localBranchName]
推送远程仓库

git push [remote] --all
推送所有分支到远程仓库

git push [remote] --force
强行推送当前分支到远程仓库, 即使有冲突

git push [remote] --tags
提交所有tag

git push [remote] [branch]
上传本地指定分支到远程仓库

git push [remote] [tag]
提交指定tag

git push origin --tags
上传本地tag到远程仓库

git push origin :heads/[name]git push origin :[name]
删除远程分支

git push origin :refs/tags/[name]
删除远程版本

git push origin master:hb-dev
将本地库与服务器上的库进行关联

git rebase --abort
终止rebase, 分支会回到rebase开始前的状态

git rebase --continue
执行rebase出现冲突解决后, 执行该命令会继续应用(apply)余下的补丁

git rebase --skip
跳过当前提交

git reflog
显示当前分支的最近几次提交

git remote -v
显示所有远程仓库

git remote add [shortname] [url]
增加一个新的远程仓库, 并命名

git remote rm [name]
删除远程仓库

git remote set-url --push [name] [newUrl]
修改远程仓库

git remote show [remote]
显示某个远程仓库的信息

git remote show
查看远程库

git reset --hard [commit]
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
彻底回退到某个版本, 替换暂存区和工作区, 本地的源码也会变为上一个版本的内容

git reset --keep [commit]
重置当前HEAD为指定commit, 但保持暂存区和工作区不变

git reset --mixed
同不带任何参数的git reset一样, 重置暂存区, 但不改变工作区

git reset --soft HEAD~3
引用回退三次(工作区不变, 暂存区不变)

git reset --soft
回退到某个版本, 不改变暂存区和工作区(如果还要提交, 直接commit即可

git reset HEAD
HEAD 效果同上, 因为引用重置到HEAD相当与没有重置

git reset HEAD^
引用回退一次(工作区不变, 暂存区回退)

git reset [commit]
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 [file]
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
将之前用git add命令添加到暂存区的内容撤出暂存区(相当于git add -A 的反向操作)

git revert -n HEAD
撤销上一次但不commit

git revert -no-edit HEAD
撤销上一次并直接使用默认注释

git revert HEAD
撤销上一次commit

git revert [commit]
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 rm --cached [file]
停止追踪指定文件, 但该文件会继续保留在工作区

git rm -r [floder]
删除文件夹

git rm [file1] [file2] ...
删除工作区文件, 并且将这次删除操作放入暂存区

git shortlog -sn
显示所有提交过的用户, 按提交次数排序

git show --name-only [commit]
显示某次提交发生变化的文件

git show [commit]
显示某次提交的元数据和内容变化

git show [commit]:[filename]
显示某次提交时, 某个文件的内容

git show [tag]
查看tag信息

git stash apply
恢复最新保存工作进度, 但不删除

git stash clear
删除所有

git stash drop
删除一个进度, 默认删除最新的

git stash list
显示进度列表

git stash pop
恢复最新保存的工作进度, 并将恢复的工作进度从存储的列表中删除

git stash push
将文件给push到一个临时空间中

git stash save "message"
保存进度加说明

git stash
保存当前的工作进度

git status --ignored
显示被忽略的文件

git status -s
将结果以简短的形式输出

git status
查看当前状态

git submodule add [url] [path]
添加子模块

git submodule init
初始化子模块

git submodule update
更新子模块

git tag -a [name] -m 'yourMessage'
创建带注释的tag

git tag -d [tag]
删除本地tag

git tag -r
查看远程版本

git tag [name]
创建版本

git tag [tag] [commit]
新建一个tag在指定commit

git tag [tag]
新建一个tag在当前commit

git tag
列出所有tag

git version
获取git版本

转自 (https://juejin.im/post/5dc4b88af265da4d3f44bed3)