新建

$ git init
注解 # 初始化当前项目
$ git init [project-name]
注解 # 新建一个目录，将其初始化为Git代码库
$ git init --bare
注解 # 在指定目录创建一个空的 Git 仓库。运行这个命令会创建一个名为 directory，只包含 .git 子目录的空目录。
$ git clone [url]
注解 # 这个命令就是将一个版本库拷贝到另一个目录中，同时也将分支都拷贝到新的版本库中。这样就可以在新的版本库中提交到远程分支 #下载一个项目和它的整个代码历史

配置

$ git config --list
注解 # 显示当前的Git配置
$ git config -e [--global]
注解 # 编辑Git配置文件 输出、设置基本的全局变量 $ git config --global user.email
注解 # 设置基本的全局当前用户所有提交者的邮箱
$ git config --global user.name
注解 # 设置基本的全局当前用户所有提交者的用户名
$ git config --global user.email "MyEmail@gmail.com"
注解 # 设置基本的全局当前用户所有提交者的用户名以及邮箱
$ git config --global user.name "My Name"
注解 # 修改用户名
$ git config --global alias.
注解 # 定义当前用户所有提交使用的作者邮箱。
$ git config --system core.editor
注解 # 为Git命令创建一个快捷方式（别名）。

状态

$ git status
注解 # 显示分支，未跟踪文件，更改和其他不同
$ git help status
注解 # 查看其他的git status的用法

添加

$ git add test.js
注解 # 添加一个文件
$ git add /path/to/file/test.js
注解 # 添加一个子目录中的文件
$ git add ./*.js
注解 # 支持正则表达式
$ git add [file1] [file2] ...
注解 # 添加指定文件到暂存区
$ git add [dir]
注解 # 添加指定目录到暂存区，包括子目录
$ git add .
注解 # 添加当前目录的所有文件到暂存区
$ git add -p
注解 # 对于同一个文件的多处变化，可以实现分次提交# 添加每个变化前，都会要求确认

提交

$ git commit -m [message]
注解 # 提交暂存区到仓库区附带提交信息
$ git commit [file1] [file2] ... -m [message]
注解 # 提交暂存区的指定文件到仓库区
$ git commit -a
注解 # 提交工作区自上次commit之后的变化，直接到仓库区
$ git commit -v
注解 # 提交时显示所有diff信息
$ git commit --amend -m [message]
注解 # 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# 使用一次新的commit，替代上一次提交
$ git commit --amend [file1] [file2] ...
注解 # 重做上一次commit，并包括指定文件的新变化

拉起

$ git pull origin master
注解 # git pull <远端> <分支># 从远端origin的master分支更新版本库
$ git pull --no-ff

注解 # 抓取远程仓库所有分支更新并合并到本地，不要快进合并
$ git ci
注解 # ci：# 复制代码
$ git ci .
注解
$ git ci -a
注解 # 将git add, git rm和git ci等操作都合并在一起做
$ git ci -am "some comments"
注解
$ git ci --amend
注解 # 修改最后一次提交记录

推送

$ git push [remote] [branch]
注解 # git push 相当于 git push origin master# git push <远端> <分支># 把本地的分支更新到远端origin的master分支上# 上传本地指定分支到远程仓库
$ git push [remote] --force
注解 # 强行推送当前分支到远程仓库，即使有冲突
$ git push [remote] --all
注解 # 推送所有分支到远程仓库

远程管理

$ git fetch [remote]
注解 # 下载远程仓库的所有变动
$ git remote -v
注解 # 显示所有远程仓库
$ git remote show [remote]
注解 # 显示某个远程仓库的信息
$ git remote add [shortname] [url]
注解 # 增加一个新的远程仓库，并命名
$ git remote -v
注解 # 查看远程服务器地址和仓库名称
$ git remote add origin git@ github:xxx/xxx.git
注解 # 添加远程仓库地址
$ git remote set-url origin git@ github.com:xxx/xxx.git
注解 # 设置远程仓库地址(用于修改远程仓库地址)
$ git remote rm
注解 # 删除远程仓库

注解 # 删除远程仓库

信息

$ git log --stat
注解 # 显示commit历史，以及每次commit发生变更的文件
$ git log -S [keyword]
注解 # 搜索提交历史，根据关键词
$ git log [tag] HEAD --pretty=format:%s
注解 # 显示某个commit之后的所有变动，每个commit占据一行
$ git log [tag] HEAD --grep feature
注解 # 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 --follow [file]
注解 # 显示某个文件的版本历史，包括文件改名
$ git whatchanged [file]
注解
$ git log -p [file]
注解 # 显示指定文件相关的每一次diff
$ git log -5 --pretty --oneline
注解 # 显示过去5次提交
$ git shortlog -sn
注解 # 显示所有提交过的用户，按提交次数排序
$ git blame [file]
注解 # 显示指定文件是什么人在什么时间修改过
$ git show [commit]
注解 # 显示某次提交的元数据和内容变化
$ git show --name-only [commit]
注解 # 显示某次提交发生变化的文件
$ git show [commit]:[filename]
注解 # 显示某次提交时，某个文件的内容
$ git reflog
注解 # 显示当前分支的最近几次提交

对比差异

$ git diff
注解 # 显示暂存区和工作区的差异
$ git diff --cached [file]
注解 # 显示暂存区和上一个commit的差异
$ git diff HEAD

注解 # 显示工作区与当前分支最新commit之间的差异
$ git diff [first-branch]...[second-branch]
注解 # 显示两次提交之间的差异
$ git diff --shortstat "@{0 day ago}"
注解 # 显示今天你写了多少行代码
$ git diff --staged
注解 # 比较暂存区和版本库差异
$ git diff --cached
注解 # 比较暂存区和版本库差异
$ git diff --stat
注解 # 仅仅比较统计信息

分支

$ git branch -a
注解 # 查看所有的分支和远程分支
$ git branch [branch-name]
注解 # 创建一个新的分支
$ git branch -m [branch-name] [new-branch-name]
注解 # git branch -m <旧名称> <新名称># 重命名分支
$ git branch [branch-name] --edit-description
注解 # 编辑分支的介绍
$ git branch
注解 # 列出所有本地分支
$ git branch -r
注解 # 列出所有远程分支
$ git branch [branch-name]
注解 # 新建一个分支，但依然停留在当前分支
$ git checkout -b [branch]
注解 # 新建一个分支，并切换到该分支
$ git branch [branch] [commit]
注解 # 新建一个分支，指向指定commit
$ git branch --track [branch] [remote-branch]
注解 # 新建一个分支，与指定的远程分支建立追踪关系
$ git checkout [branch-name]
注解 # 切换到指定分支，并更新工作区
$ git checkout -
注解 # 切换到上一个分支
$ git branch --set-upstream [branch] [remote-branch]
注解 # 建立追踪关系，在现有分支与指定的远程分支之间
$ git merge [branch]

$ git merge [branch]
注解 # 合并指定分支到当前分支
$ git cherry-pick [commit]
注解 # 选择一个commit，合并进当前分支
$ git branch -d [branch-name]
注解 # 删除分支
$ git push origin --delete [branch-name]
注解 # 删除远程分支
$ git branch -dr [remote/branch]
注解 # 删除远程分支
$ git co
注解 # 切换到某个分支
$ git co -b
注解 # 创建新的分支，并且切换过去
$ git co -b
注解 # 基于branch创建新的new_branch
$gitco$id
注解 # 把某次历史提交记录checkout出来，但无分支信息，切换到其他分支会自动删除
$gitco$id -b
注解 # 把某次历史提交记录checkout出来，创建成一个分支
$ git br -d
注解 # 删除某个分支
$ git br -D
注解 # 强制删除某个分支 (未被合并的分支被删除的时候需要强制)
$ git br -v
注解 # 查看各个分支最后提交信息
$ git br --merged
注解 # 查看已经被合并到当前分支的分支
$ git br --no-merged
注解 # 查看尚未被合并到当前分支的分支

删除

$ git rm HelloWorld.js
注解 # 移除 HelloWorld.js
$ git rm /pather/to/the/file/HelloWorld.js
注解 # 移除子目录中的文件
$ git rm [file1] [file2] ...
注解 # 删除工作区文件，并且将这次删除放入暂存区
$ git rm --cached [file]
注解 # 停止追踪指定文件，但该文件会保留在工作区

检出

$ git checkout
注解 # 检出一个版本库，默认将更新到master分支
$ git checkout branchName
注解 # 检出到一个特定的分支
$ git checkout -b newBranch
注解 # 新建一个分支，并且切换过去，相当于"git branch <名字>; git checkout <名字>"

撤销

$ git checkout [file]
注解 # 恢复暂存区的指定文件到工作区
$ git checkout [commit] [file]
注解 # 恢复某个commit的指定文件到暂存区和工作区
$ git checkout .
注解 # 恢复暂存区的所有文件到工作区
$ git reset [file]
注解 # 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 --hard
注解 # 重置暂存区与工作区，与上一次commit保持一致
$ git reset [commit]
注解 # 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 --hard [commit]
注解 # 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 --keep [commit]
注解 # 重置当前HEAD为指定commit，但保持暂存区和工作区不变
$ git revert [commit]
注解 # 后者的所有变化都将被前者抵消，并且应用到当前分支# 新建一个commit，用来撤销指定commit
$ git revert HEAD
注解 # 恢复最后一次提交的状态
$ git stash
注解 # 暂时将未提交的变化移除，稍后再移入
$ git stash pop
注解
$ git stash list
注解 # 列所有stash
$ git stash apply

$ git stash apply
注解 # 恢复暂存的内容
$ git stash drop
注解 # 删除暂存区

查找

$ git config --global grep.lineNumber true
注解 # 在搜索结果中显示行号
$ git config --global alias.g "grep --break --heading --line-number"
注解 # 是搜索结果可读性更好
$ git grep 'variableName' -- '*.java'
注解 # 在所有的java中查找variableName
$ git grep -e 'arrayListName' --and (-e add -e remove)
注解 # 搜索包含 "arrayListName" 和, "add" 或 "remove" 的所有行

合并

$ git merge branchName
注解 # 将其他分支合并到当前分支
$ git merge --no-ff branchName
注解 # 不要 Fast-Foward 合并，这样可以生成 merge 提交# 在合并时创建一个新的合并后的提交
$ git mv test.js test2.js
注解 # 重命名# mv： 重命名或移动一个文件# 复制代码
$ git mv test.js ./new/path/test.js
注解 # 移动
$ git mv [file-original] [file-renamed]
注解 # 改名文件，并且将这个改名放入暂存区
$ git mv -f myFile existingFile
注解 # 这个文件已经存在，将要覆盖掉# 强制重命名或移动

标签

$ git tag
注解 # 列出所有tag
$ git tag [tag]
注解 # 新建一个tag在当前commit
$ git tag [tag] [commit]

注解 # 新建一个tag在指定commit
$ git tag -d [tag]
注解 # 删除本地tag
$ git push origin :refs/tags/[tagName]
注解 # 删除远程tag
$ git show [tag]
注解 # 查看tag信息
$ git push [remote] [tag]
注解 # 提交指定tag
$ git push [remote] --tags
注解 # 提交所有tag
$ git checkout -b [branch] [tag]
注解 # 新建一个分支，指向某个tag