Cheatography

Git Cheat Sheet

by ooze via cheatography.com/185867/cs/38824/

基础命令

git init	将当前文件夹初始化为一个 git 仓库，可以在后面指定目录
git clone <repo>	clone 代码仓库到本地
git status	查看本地仓库状态
git log	查看历史记录，有很多参数可以选择
git <command> -h	查看帮助信息，使用 --help 可查看完整的帮助文档

Git 配置

git config --list	查看 Git 配置
git config --global user.name <name>	设置用户名
git config --global user.email <email>	设置用户邮箱
git config --global core.editor <editor>	设置文本编辑器
git config --global alias.<alias-name> <command>	设置 Git 命令别名
git config --global commit.template <file>	设置提交模板
git config --global core.autocrlf [true|false|input]	跨平台换行符配置

--system 系统级，--global 用户级（推荐），--local 仓库级。每一个级别会覆盖前一级别的配置。

Git 分支

git branch	查看本地分支，后面添加分支名即可创建新分支
git checkout <branch>	切换分支

Git 分支的本质是一个指向提交对象的可变指针，有一个名为 HEAD 的指针会指向当前所在的本地分支。

拉取代码

git fetch	从远程仓库取回所有分支的更新，但不会修改本地工作目录的内容
git pull	拉取代码并自动合并，= git fetch + git merge
git pull --rebase	拉取代码并以 rebase 模式合并代码，= git fetch + git rebase

合并代码

git merge <branch>	合并分支，使用 --no-ff 采用非快进合并
git cherry-pick -x <commit>	合并提交，-x 在提交信息末-尾追加来源，-s 追加操作者签名
git cherry-pick <commit1>..<commit2>	合并多个连续的提交，左开右闭区间
git cherry-pick <commit1>^..<commit2>	合并多个连续的提交，左闭右闭区间
git format-patch HEAD~n	生成最近 n 次提交的补丁
git am --keep-cr <patch>	打入补丁，--keep-cr 防止回车符被移除导致打补丁失败，冲突时可使用 --reject 查看冲突

历史记录

git log -S <string>	查看与某个字符串 string 相关的改动
git log -- <path>	查看一个文件或目录的历史记录，可以此查询某个文件是何时被删除的
git reflog	查看本地仓库 HEAD 指针的所有变更，可用于恢复丢失的代码。
git blame	追踪代码改动

查看改动

git diff	查看已修改文件的改动
git diff --staged	查看已暂存文件的改动
git diff HEAD~	查看最近一次已提交文件的改动

Cheatography

Git Cheat Sheet

by ooze via cheatography.com/185867/cs/38824/

提交代码

git add <directory>	暂存指定目录或文件的改动，git add . 会暂存当前目录的所有修改
git commit	提交暂存区的代码，并打开文本编辑器编写提交信息
git commit -m <message>	提交暂存区的代码，将提交信息与命令放在同一行
git commit -a	跳过暂存区直接提交修改的代码，即跳过 git add 的操作
git commit --amend	修改最后一次提交，可用来修改提交信息
git push	推送代码到远程仓库

将已修改（Modified）的文件或未跟踪（Untracked）的文件添加（add）到暂存区（Staged），再将暂存区的文件提交（commit）到本地分支，最后将本地的提交推送（push）到远程仓库。

撤消改动

git clean -f	删除未跟踪的文件，-f 后面可以指定路径，如果不指定路径就是当前目录，-d 可以移除文件夹
git clean -ndf	-n 显示会被删除的文件，并不会执行移除操作，可使用该参数防止误删
git checkout -- <filename>	撤消指定文件的修改
git checkout .	撤消所有文件的修改
git reset HEAD <filename>	将指定文件从暂存区撤消
git reset HEAD .	将所有暂存区的文件撤消，即回退到 git add 之前
git reset --[mixed|soft|hard] <commit>	撤消本地提交，默认参数是 --mixed，使用 --hard 时需小心
git revert <commit>	回滚提交，会创建一个新提交撤消 <commit> 的改动

贮藏改动

git stash	贮藏文件的修改，使用 -u 也会贮藏未跟踪的文件
git stash list	列出所有贮藏的改动
git stash apply	应用最近一次贮藏的改动，后面跟上 stash@{n} 可指定对应的记录
git stash pop	应用最近一次贮藏的改动，并将这条贮藏丢弃
git stash drop	丢弃最近一次贮藏的改动，后面跟上 stash@{n} 可指定对应的记录
git stash clear	丢弃所有的贮藏改动

贮藏的改动使用栈存储

调整历史记录

git rebase -i HEAD~n	以交互方式重新应用最近 n 次的提交，可对提交进行修改

Git 并没有一个修改历史记录的工具，但是却可以利用变基命令来变基一系列提交，从而达到修改历史记录的目的。