Git 常用指令集合

指令 - 查看状态信息

指令	描述
git --version	查看 git 版本
git status	查看本地仓库状态，比较常用的指令，加参数 `-s` 简洁模式
git [指令] -h	查看指令帮助信息
git [指令] --help	打开指令详细帮助页面
git remote -v	查看所有远程仓库，不带参数 `-v` 只显示名称
git branch	查看分支列表，以及当前分支
git tag	查看标签列表
git log -n20	查看日志 (最近 20 条)，可省略 `n` 为 `-20`；参数 `--graph` 可视化显示分支关系
git log --follow [file]	显示某个文件的版本历史，包括文件改名
git reflog	查看所有可用的历史版本记录（实际是 HEAD 变更记录），包含被回退的记录，常用来撤销回退
git blame [file]	以列表形式查看指定文件的历史修改记录
git config --list	查看配置信息，包括系统（--system）+ 全局（--global）+ 项目（--local）配置
git config --list --system	查看系统配置，全局（--global）、项目（--local）配置配置类似 cat [file] 读取一个文件，展示其文件内容

指令 - 代码 / 仓库管理

指令	描述
git clone [git 地址]	从远程仓库克隆到本地（当前目录）
git init [文件目录]	初始化创建 Git 仓库，如果不指定 [文件目录]，则在当前目录创建。
git add [file1] [file2]	添加文件到暂存区，包括修改的文件、新增的文件
git add [dir]	同上，添加目录到暂存区，包括子目录
git add .	同上，添加所有修改、新增文件（未跟踪）到暂存区
git rm [file]	删除工作区文件，并且将这次删除放入暂存区
git commit -m '说明'	提交变更，参数 `-m` 设置提交的描述信息，应该正确提交，不带该参数会进入说明编辑模式
git commit -a	参数 `-a`，表示直接从工作区提交到版本库，略过了 `git add` 步骤，不包括新增的文件
git commit [file]	提交暂存区的指定文件到仓库区
git commit --amend -m	使用一次新的 `commit`，替代上一次提交，会修改 `commit` 的 `hash` 值（id）
git cherry-pick [commit]	拣选提交，复制一个特定的提交到当前分支，而不管这个提交在哪个分支
git log -n20	查看日志 (最近 20 条)，不带参数 `-n` 则显示所有日志
git log -n20 --oneline	参数 “`--oneline`” 可以让日志输出更简洁（一行）
git log -n20 --graph	参数 “`--graph`” 可视化显示分支关系
git log --follow [file]	显示某个文件的版本历史
git blame [file]	以列表形式显示指定文件的修改记录
git reflog	查看所有可用的历史版本记录（实际是 HEAD 变更记录），包含被回退的记录（重要）

指令 - diff

指令	描述
git diff	查看暂存区和工作区的差异
git diff [file]	同上，指定文件
git diff --cached	查看已暂存的改动，就是暂存区与新版本 `HEAD` 进行比较
git diff --staged	同上
git diff --cached [file]	同上，指定文件
git diff HEAD	查看已暂存的 + 未暂存的所有改动，就是与最新版本 `HEAD` 进行比较
git diff HEAD~	同上，与上一个版本比较。`HEAD~` 表示上一个版本，`HEAD~10` 为最近第 10 个版本
git diff [id] [id]	查看两次提交之间的差异
git diff [branch]	查看工作区和分支直接的差异

指令 - 远程仓库

指令	描述
git clone [git 地址]	从远程仓库克隆到本地（当前目录）
git remote -v	查看所有远程仓库，不带参数 `-v` 只显示名称
git remote show [remote]	显示某个远程仓库的信息
git remote add [name] [url]	增加一个新的远程仓库，并命名
git remote rename [old] [new]	修改远程仓库名称
git pull [remote] [branch]	取回远程仓库的变化，并与本地版本合并
git pull	同上，针对当前分支
git fetch [remote]	获取远程仓库的所有变动到本地仓库，不会自动合并！需要手动合并
git push	推送当前分支到远程仓库
git push [remote] [branch]	推送本地当前分支到远程仓库的指定分支
git push [remote] --force/-f	强行推送当前分支到远程仓库，即使有冲突，⚠很危险！
git push [remote] --all	推送所有分支到远程仓库
git push –u	参数 `–u` 表示与远程分支建立关联，第一次执行的时候用，后面就不需要了
git remote rm [remote-name]	删除远程仓库
git pull --rebase	使用 rebase 的模式进行合并

指令 - 分支

指令	描述
git branch	列出所有本地分支，加参数 `-v` 显示详细列表，下同
git branch -r	列出所有远程分支
git branch -a	列出所有本地分支和远程分支，用不同颜色区分
git branch [branch-name]	新建一个分支，但依然停留在当前分支
git branch -d dev	删除 `dev` 分支，`-D`（大写）强制删除
git checkout -b dev	从当前分支创建并切换到 `dev` 分支
git checkout -b feature1 dev	从本地 `dev` 分支代码创建一个 `feature1` 分支，并切换到新分支
git branch [branch] [commit]	新建一个分支，指向指定 `commit id`
git branch --track [branch] [remote-branch]	新建一个分支，与指定的远程分支建立关联
git checkout -b hotfix remote hotfix	从远端 `remote` 的 `hotfix` 分支创建本地 `hotfix` 分支
git branch --set-upstream [branch] [remote-branch]	在现有分支与指定的远程分支之间建立跟踪关联: `git branch --set-upstream hotfix remote/hotfix`
git checkout [branch-name]	切换到指定分支，并更新工作区
git checkout .	撤销工作区的（未暂存）修改，把暂存区恢复到工作区。
git checkout HEAD .	撤销工作区、暂存区的修改，用 `HEAD` 指向的当前分支最新版本替换
git merge [branch]	合并指定分支到当前分支
git merge --no-ff dev	合并 `dev` 分支到当前分支，参数 `--no-ff` 禁用快速合并模式
git push origin --delete [branch-name]	删除远程分支
git rebase master	将当前分支变基合并到 `master` 分支
☑switch：新的分支切换指令	切换功能和 `checkout` 一样，`switch` 只单纯的用于切换
git switch master	切换到已有的 `master` 分支
git switch -c dev	创建并切换到新的 `dev` 分支

指令 - 标签管理

指令	描述
git tag	查看标签列表
git tag -l 'a*'	查看名称是 “a” 开头的标签列表，带查询参数
git show [tagname]	查看标签信息
git tag [tagname]	创建一个标签，默认标签是打在最新提交的 commit 上的
git tag [tagname]	新建一个 tag 在指定 commit 上
git tag -a v5.1 -m'v5.1 版本'	创建标签 `v5.1.1039`，`-a` 指定标签名，`-m` 指定说明文字
git tag -d [tagname]	删除本地标签
git checkout v5.1.1039	切换标签，同切换分支
git push [remote] v5.1	推送标签，标签不会默认随代码推送推送到服务端
git push [remote] --tags	提交所有 tag

指令 - 撤销变更

指令	描述
git checkout .	撤销工作区的（未暂存）修改，把暂存区恢复到工作区。不影响暂存区，如果没暂存，则撤销所有工作区修改
git checkout [file]	同上，`file` 指定文件
git checkout HEAD .	撤销工作区、暂存区的修改，用 `HEAD` 指向的当前分支最新版本替换工作区、暂存区
git checkout HEAD [file]	同上，`file` 指定文件
git reset	撤销暂存区状态，同 `git reset HEAD`，不影响工作区
git reset HEAD [file]	同上，指定文件 `file`，`HEAD` 可省略
git reset [commit]	回退到指定版本，清空暂存区，不影响工作区。工作区需要手动 `git checkout` 签出
git reset --soft [commit]	移动分支 master、HEAD 到指定的版本，不影响暂存区、工作区，需手动 `git checkout` 签出更新
git reset --hard HEAD	撤销工作区、暂存区的修改，用当前最新版
git reset --hard HEAD~	回退到上一个版本，并重置工作区、暂存区内容。
git reset --hard [commit]	回退到指定版本，并重置工作区、暂存区内容。
git revert [commit]	撤销一个提交，会用一个新的提交（原提交的逆向操作）来完成撤销操作，如果已 `push` 则重新 `push` 即可

指令 - stash

指令	描述
git stash	把未提交内容隐藏起来，包括未暂存、已暂存。 等以后恢复现场后继续工作
git stash list	查看所有被隐藏的内容列表
git stash pop	恢复被隐藏的内容，同时删除隐藏记录
git stash save "message"	同 `git stash`，可以备注说明 `message`
git stash apply	恢复被隐藏的文件，但是隐藏记录不删除
git stash drop	删除隐藏记录

命令行的退出方式

有些 git 命令的执行并不是一次就完成了，会有连续的后续操作。

- **“git log” 退出：**`git log` 输出历史日志记录，比较多的时候先出现一部分，输入回车持续输出。
  - 退出方式：输入 “q”。
- **“git commit” 退出：**`git commit` 没有带 `-m` 参数时，会进入 vim 编辑模式，等待输入提交的描述信息。
  - 保存并退出：
    - 按 Esc 键退出编辑模式，英文模式下输入 `:wq` ，然后回车。
    - 按 Esc 键退出编辑模式，大写英文模式下输入 `ZZ` ，然后回车。
  - 不保存退出：
    - 按 Esc 键退出编辑模式，英文模式下输入 `:q!` ，然后回车。
    - 按 Esc 键退出编辑模式，英文模式下输入 `:qa!` ，然后回车。

转载

https://www.yuque.com/kanding/ktech/ai3d3ky8f0dgixto