



链滴

git 命令大全

作者: [zdlgv5](#)

原文链接: <https://ld246.com/article/1472282056447>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

<p>查看、添加、提交、删除、找回，重置修改文件</p>

<p>git help <command> # 显示command的help</p>

<p>git show # 显示某次提交的内容 git show \$id</p>

<p>git co -- <file> # 抛弃工作区修改</p>

<p>git co . # 抛弃工作区修改</p>

<p>git add <file> # 将工作文件修改提交到本地暂存区</p>

<p>git add . # 将所有修改过的工作文件提交暂存区</p>

<p>git rm <file> # 从版本库中删除文件</p>

<p>git rm <file> --cached # 从版本库中删除文件，但不删除文件</p>

<p>git reset <file> # 从暂存区恢复到工作文件</p>

<p>git reset -- . # 从暂存区恢复到工作文件</p>

<p>git reset --hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改</p>

<p>git ci <file> git ci . git ci -a # 将git add, git rm和git ci等操作都合并在一起做</p>

<p>git ci --amend # 修改最后一次提交记录</p>

<p>git revert <\$id> # 恢复某次提交的状态，恢复动作本身也创建次提交对象</p>

<p>git revert HEAD # 恢复最后一次提交的状态</p>

<p>查看文件diff</p>

<p>git diff <file> # 比较当前文件和暂存区文件差异 git diff</p>

<p>git diff <span id="MathJax-Element-1-Frame" class="MathJax" data-mathml="<math xmlns="http://www.w3.org/1998/Math/MathML" ><mi>i</mi><mi>d</mi></mi></mi><mn>1</mn></mn><mo>&gt;</mo></math" >id1&id1 id2 # 比较两次提交之间的差异</p>

<p>git diff <branch1>..<branch2> # 在两个分支之间比较</p>

<p>git diff --staged # 比较暂存区和版本库差异</p>

<p>git diff --cached # 比较暂存区和版本库差异</p>

<p>git diff --stat # 仅仅比较统计信息</p>

<p>查看提交记录</p>

<p>git log git log <file> # 查看该文件每次提交记录</p>

<p>git log -p <file> # 查看每次详细修改内容的diff</p>

<p>git log -p -2 # 查看最近两次详细修改内容的diff</p>

<p>git log --stat #查看提交统计信息</p>

<p>tig</p>

<p>Mac上可以使用tig代替diff和log, <code>brew install tig</code></p>

<p>Git 本地分支管理</p>

<p>查看、切换、创建和删除分支</p>

<p>git br -r # 查看远程分支</p>

<p>git br <new_branch> # 创建新的分支</p>

<p>git br -v # 查看各个分支最后提交信息</p>

<p>git br --merged # 查看已经被合并到当前分支的分支</p>

<p>git br --no-merged # 查看尚未被合并到当前分支的分支</p>

<p>git co <branch> # 切换到某个分支</p>

<p>git co -b <new_branch> # 创建新的分支，并且切换过去</p>

<p>git co -b <new_branch> <branch> # 基于branch创建新的new_branch</p>

<p>git co \$id # 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</p>

<p>git co \$id -b <new_branch> # 把某次历史提交记录checkout出来，创建成一个分支</p>

<p>git br -d <branch> # 删除某个分支</p>

<p>git br -D <branch> # 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)</p>

<p> 分支合并和rebase</p>

<p>git merge <branch>; # 将branch分支合并到当前分支</p>
<p>git merge origin/master --no-ff # 不要Fast-Foward合并, 这样可以生成merge提交</p>
<p>git rebase master <branch>; # 将master rebase到branch, 相当于: git co <branc > && git rebase master && git co master && git merge <ranch>;</p>
<p> Git补丁管理(方便在多台机器上开发同步时用)</p>
<p>git diff >; ../sync.patch # 生成补丁</p>
<p>git apply ../sync.patch # 打补丁</p>
<p>git apply --check ../sync.patch #测试补丁能否成功</p>
<p> Git暂存管理</p>
<p>git stash # 暂存</p>
<p>git stash list # 列所有stash</p>
<p>git stash apply # 恢复暂存的内容</p>
<p>git stash drop # 删除暂存区</p>
<p>Git远程分支管理</p>
<p>git pull # 抓取远程仓库所有分支更新并合并到本地</p>
<p>git pull --no-ff # 抓取远程仓库所有分支更新并合并到本地, 不要快进合并</p>
<p>git fetch origin # 抓取远程仓库更新</p>
<p>git merge origin/master # 将远程主分支合并到本地当前分支</p>
<p>git co --track origin/branch # 跟踪某个远程分支创建相应的本地分支</p>
<p>git co -b <local_branch>; origin/<remote_branch>; # 基于远程分支创建本地分支功能同上</p>
<p>git push # push所有分支</p>
<p>git push origin master # 将本地主分支推到远程主分支</p>
<p>git push -u origin master # 将本地主分支推到远程(如无远程主分支则创建, 用于初始化远程库)</p>
<p>git push origin <local_branch>; # 创建远程分支, origin是远程仓库名</p>
<p>git push origin <local_branch>;<remote_branch>; # 创建远程分支</p>
<p>git push origin :<remote_branch>; #先删除本地分支(git br -d <branch>;), 然后push删除远程分支</p>
<p>Git远程仓库管理</p>
<p>GitHub</p>
<p>git remote -v # 查看远程服务器地址和仓库名称</p>
<p>git remote show origin # 查看远程服务器仓库状态</p>
<p>git remote add origin git@ github:robbin/robbin_site.git # 添加远程仓库地址</p>
<p>git remote set-url origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址(用于改远程仓库地址) git remote rm <repository>; # 删除远程仓库</p>
<p>创建远程仓库</p>
<p>git clone --bare robbin_site robbin_site.git # 用带版本的项目创建纯版本仓库</p>
<p>scp -r my_project.git git@ git.csdn.net:~ # 将纯仓库上传到服务器上</p>
<p>mkdir robbin_site.git && cd robbin_site.git && git --bare init # 在服务器创建纯仓库</p>
<p>git remote add origin git@ github.com:robbin/robbin_site.git # 设置远程仓库地址</p>
<p>git push -u origin master # 客户端首次提交</p>
<p>git push -u origin develop # 首次将本地develop分支提交到远程develop分支, 并且track</p>
>
<p>git remote set-head origin master # 设置远程仓库的HEAD指向master分支</p>
<p>也可以命令设置跟踪远程库和本地库</p>
<p>git branch --set-upstream master origin/master</p>
<p>git branch --set-upstream develop origin/develop</p>