

Git 命令

开源的分布式版本控制系统 Git 命令集

几 昆虫 制作

关于.....	1	调试.....	2
设置与配置.....	1	邮件.....	2
获取与创建项目.....	1	外部系统.....	2
快照基础.....	1	管理.....	2
分支与合并.....	1	服务器管理.....	2
项目分享和更新.....	1	底层命令.....	2
检查与比较.....	2	参考.....	2
补丁.....	2		

关于

Git 是一个快速、可扩展的分布式版本控制系统。

Git URLs

```
ssh://[user@]host.xz[:port]/path/to/repo.git/
git://host.xz[:port]/path/to/repo.git/
http[s]://host.xz[:port]/path/to/repo.git/
ftp[s]://host.xz[:port]/path/to/repo.git/
rsync://host.xz/path/to/repo.git/
```

指定版本

HEAD	# 最近一个提交
HEAD^ 或 HEAD~1	# HEAD 之前的那个提交
HEAD^^ 或 HEAD~2	# 比 HEAD 早两个的那个提交

设置与配置

git 基本命令

```
git [--version] [--help] [-C <path>] [-c <name>=<value>]
  [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
  [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
  [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
  [--super-prefix=<path>]
  <command> [<args>]
```

查看 Git 版本、执行路径

```
git --version
git --exec-path
```

git-config 获取和设置版本库或全局选项

设置全局的用户名和电子邮箱地址

```
git config --global user.name <name>
git config --global user.email <email address>
```

git-help 显示 Git 帮助信息

了解每一个命令的完整的可选项及标志列表

```
git help <command>
```

获取与创建项目

git-init 创建一个空的 Git 版本库，或重新初始化已有的版本库

在当前或指定目录创建一个新的本地版本库

```
git init [<directory>]
```

git-clone 克隆一个版本库到新目录

克隆一个项目及其的整个版本历史

```
git clone <repository> [<directory>]
```

快照基础

git-add 添加文件内容到索引（暂存区）中

添加目录中所有的文件到暂存区

```
git add .
```

git-status 显示工作树状态

显示工作树及暂存区中不同状态的文件，包含已修改但未暂存，或已暂存但未提交的文件

```
git status
```

git-commit 保存变更到版本库中

提交所有的变更，并带提交说明

```
git commit -m <msg>
```

git-reset 重置当前 HEAD 到指定状态

重置（即：取消息外地输入了 git add * 暂存的文件）

```
git reset
```

git-rm 从工作树和从索引（暂存区）中移除文件

删除文件

```
git rm <file>
```

从版本控制中删除文件，但文件保留在本地

```
git rm --cached <file>
```

git-mv 移动或重命名一个文件、目录或符号链接

修改文件名，并准备提交

```
git mv <source> <destination>
```

分支与合并

git-branch 列出、创建和删除分支

列出当前版本库的所有本地分支

```
git branch
```

创建一个新的分支

```
git branch <branchname>
```

删除指定的分支

```
git branch -d <branchname>
```

git-checkout 检出一个分支或路径到工作树

切换到指定的分支，并更新工作目录

```
git checkout <branch>
```

git-merge 两至多个开发历史并入一起

合并指定分支的历史记录到当前分支

```
git merge <branch>
```

git-mergetool 运行合并冲突解决工具来解决合并冲突

git-stash 隐藏掉修改过的工作目录中的变更

临时存储所有已修改的追踪文件，以便在分支上不需要提交未完成工作就可以清理工作目录

```
git stash
```

恢复最近的隐藏文件

```
git stash pop
```

列出所有隐藏的变更集

```
git stash list
```

舍弃最近隐藏的变更集

```
git stash drop
```

git-tag 创建、删除、列出和验证 GPG 签名的标签对象

列出当前版本库的所有标签对象

```
git tag
```

创建一个新的标签

```
git tag <tagname>
```

删除指定的标签

```
git tag -d <tagname>
```

项目分享和更新

git-fetch 下载其它版本库的对象和 refs

获取远程版本库分支

```
git fetch <repository> [<branchname>]
```

git-pull 获取和集成其它版本库或一个本地分支

拉入远程版本库变更，并合并到本地版本库

```
git pull <repository>
```

git-push 更新远端 refs 及其相关对象

把本地版本库变更推送到远程版本库

```
git push <repository>
```

git-remote 远程版本库管理

查看远程版本库

```
git remote -v
```

添加远程版本库

```
git remote add <name> <url>
```

```
# 删除远程版本库
git remote remove <name>

# 修改远程版本库
git remote set-url [--push] <name> <newurl>
```

```
git-submodule 初始化、更新或检查子模块
# 添加子模块
git submodule add <repository>

# 注册子模块（只在首次检出仓库时运行一次就行）
git submodule init

# 更新子模块（每次更新或切换分支后都需要运行一下）
git submodule update
```

检查与比较

```
git-show 显示各种类型的对象
# 显示一或多个对象（blobs, trees, tags 和 commits）
git show <object>...
```

```
git-log 显示提交日志
# 列出当前分支的版本历史
git log

# 列出一个文件（包括重命名）的版本历史
git log --follow <file>
```

```
git-diff 显示多次提交、单次提交和工作树等之间的变更
# 工作树与暂存区之间的差异
git diff

# 工作树与版本库之间的差异
git diff HEAD

# 暂存区与版本库之间的差异（即：已暂存的将要添加到下次提交里的内容）
git diff --cached

# 暂存区与最后提交之间的差异
git diff --staged

# 两个提交记录之间的差异
git diff master branchB
```

```
git-shortlog 概述 git log 输出
# 简单命令
git shortlog
```

```
git-describe 使用从它可获得的最近标签来描述一次提交
# 简单命令
git describe
```

补丁

```
git-apply 给文件和（或）索引打补丁
# 生成和使用补丁
git diff > patch
git apply patch
```

```
git-cherry-pick 提交拣选
# 获得在单个提交中引入的变更，然后尝试将其作为一个新的提交引入到你当前分支上
git cherry-pick <commit>...
```

```
git-rebase 分支变基
# 计算出一系列的提交，然后再以它们在其他地方以同样的顺序一个个的 cherry-picks 出它们
git rebase master
```

```
git-revert 撤销先前的提交
# 撤销某次操作（此次操作之前的提交都会被保留）
git revert <commit>...
```

调试

```
git-bisect 使用二分查找来找出 BUG 相关的提交
git-blame 显示一个文件的每一行修订了什么，以及最后修改的作者
git-grep 打印匹配一个模式的所有行
```

邮件

```
git-am 应用来自邮箱的系列补丁
git-format-patch 准备电子邮件提交的补丁（以 mbox 的格式）
git-send-email 用电子邮件发送补丁集
git-request-pull 生成即将变更的摘要
```

外部系统

```
git-svn Subversion 版本库和 Git 之间的双向操作
git-fast-import 快速 Git 数据导入器后端
```

管理

```
git-clean 移除工作树中未跟踪的文件
git-gc 清理不必要的文件和优化本地版本库
git-fsck 验证数据库中对象的连通性和有效性
git-reflog 管理 reflog 信息
git-filter-branch 重写分支
git-instaweb 立即启用 gitweb 浏览你的工作版本库
git-archive 创建指定工作树的文件归档
git-bundle 通过归档来移动对象和引用
```

服务器管理

```
git-daemon 一个非常简单的 Git 版本库服务器
git-update-server-info 更新辅助信息文件帮助哑服务器
```

底层命令

```
git-cat-file 提供版本库对象的内容或类型和尺寸
git-check-ignore 调试 gitignore 或排除文件
git-commit-tree 创建新的提交对象
git-count-objects 计算打开的对象数和磁盘消耗量
git-diff-index 同工作树和索引比较一颗树
git-for-each-ref 输出每个引用的信息
git-hash-object 计算一个文件的对象 ID，创建 blob（可选）
git-ls-files 显示索引和工作树中文件的信息
git-merge-base 为合并寻找尽可能好的共同祖先
git-read-tree 读取树信息至索引中
git-rev-list 列出按时间顺序倒序提交的对象
git-rev-parse 挑选和揉捏参数（探测工具）
git-show-ref 列出本地版本库中的引用
git-symbolic-ref 读取、修改和删除符号引用
git-update-index 将工作树中的文件内容注册到索引中
git-update-ref 安全地更新存储在一个引用中的对象名称
git-verify-pack 验证打包的 Git 归档文件
git-write-tree 从当前索引创建树对象
```

参考



Git Reference

来源


<http://git-scm.com/docs/>

作者

Git 官网

```
推荐图书
# ISBN: 9787111349679 《Git 权威指南》 2011-06;
```

版权所有 © 2017 参考集 refs.cn 。保留所有权利。
没有参考集的授权，你不得以任何形式发布或出售本文档。



意见反馈 / 赞助机会

cards@refs.cn