**Git 常用命令速查表(图文+表格)**

转载  2014-09-21   投稿：mdxy-dxy   [[http://www.jb51.net/skin/2018/images/text-message.png](http://www.jb51.net/article/55442.htm#comments) 我要评论](http://www.jb51.net/article/55442.htm#comments)

这篇文章主要介绍了Git 常用命令速查表(图文+表格),需要的朋友可以参考下

[](http://www.php.cn/course.html?ad51)

**一、 Git 常用命令速查**

git branch 查看本地所有分支  
git status 查看当前状态   
git commit 提交   
git branch -a 查看所有的分支  
git branch -r 查看远程所有分支  
git commit -am "init" 提交并且加注释   
git remote add origin git@192.168.1.119:ndshow  
git push origin master 将文件给推到服务器上   
git remote show origin 显示远程库origin里的资源   
git push origin master:develop  
git push origin master:hb-dev 将本地库与服务器上的库进行关联   
git checkout --track origin/dev 切换到远程dev分支  
git branch -D master develop 删除本地库develop  
git checkout -b dev 建立一个新的本地分支dev  
git merge origin/dev 将分支dev与当前分支进行合并  
git checkout dev 切换到本地dev分支  
git remote show 查看远程库  
git add .  
git rm 文件名(包括路径) 从git中删除指定文件  
git clone git://github.com/schacon/grit.git 从服务器上将代码给拉下来  
git config --list 看所有用户  
git ls-files 看已经被提交的  
git rm [file name] 删除一个文件  
git commit -a 提交当前repos的所有的改变  
git add [file name] 添加一个文件到git index  
git commit -v 当你用－v参数的时候可以看commit的差异  
git commit -m "This is the message describing the commit" 添加commit信息  
git commit -a -a是代表add，把所有的change加到git index里然后再commit  
git commit -a -v 一般提交命令  
git log 看你commit的日志  
git diff 查看尚未暂存的更新  
git rm a.a 移除文件(从暂存区和工作区中删除)  
git rm --cached a.a 移除文件(只从暂存区中删除)  
git commit -m "remove" 移除文件(从Git中删除)  
git rm -f a.a 强行移除修改后文件(从暂存区和工作区中删除)  
git diff --cached 或 $ git diff --staged 查看尚未提交的更新  
git stash push 将文件给push到一个临时空间中  
git stash pop 将文件从临时空间pop下来  
---------------------------------------------------------  
git remote add origin git@github.com:username/Hello-World.git  
git push origin master 将本地项目给提交到服务器中  
-----------------------------------------------------------  
git pull 本地与服务器端同步  
-----------------------------------------------------------------  
git push (远程仓库名) (分支名) 将本地分支推送到服务器上去。  
git push origin serverfix:awesomebranch  
------------------------------------------------------------------  
git fetch 相当于是从远程获取最新版本到本地，不会自动merge  
git commit -a -m "log\_message" (-a是提交所有改动，-m是加入log信息) 本地修改同步至服务器端 ：  
git branch branch\_0.1 master 从主分支master创建branch\_0.1分支  
git branch -m branch\_0.1 branch\_1.0 将branch\_0.1重命名为branch\_1.0  
git checkout branch\_1.0/master 切换到branch\_1.0/master分支  
du -hs

git branch 删除远程branch  
git push origin :branch\_remote\_name  
git branch -r -d branch\_remote\_name  
-----------------------------------------------------------

初始化版本库，并提交到远程服务器端  
mkdir WebApp  
cd WebApp  
git init 本地初始化  
touch README  
git add README 添加文件  
git commit -m 'first commit'  
git remote add origin git@github.com:daixu/WebApp.git

增加一个远程服务器端

上面的命令会增加URL地址为'git@github.com:daixu/WebApp.git'，名称为origin的远程服务器库，以后提交代码的时候只需要使用 origin别名即可

**二、 Git 命令速查表**

1、常用的Git命令

| **命令** | **简要说明** |
| --- | --- |
| git add | 添加至暂存区 |
| git add–interactive | 交互式添加 |
| git apply | 应用补丁 |
| git am | 应用邮件格式补丁 |
| git annotate | 同义词，等同于 git blame |
| git archive | 文件归档打包 |
| git bisect | 二分查找 |
| git blame | 文件逐行追溯 |
| git branch | 分支管理 |
| git cat-file | 版本库对象研究工具 |
| git checkout | 检出到工作区、切换或创建分支 |
| git cherry-pick | 提交拣选 |
| git citool | 图形化提交，相当于 git gui 命令 |
| git clean | 清除工作区未跟踪文件 |
| git clone | 克隆版本库 |
| git commit | 提交 |
| git config | 查询和修改配置 |
| git describe | 通过里程碑直观地显示提交ID |
| git diff | 差异比较 |
| git difftool | 调用图形化差异比较工具 |
| git fetch | 获取远程版本库的提交 |
| git format-patch | 创建邮件格式的补丁文件。参见 git am 命令 |
| git grep | 文件内容搜索定位工具 |
| git gui | 基于Tcl/Tk的图形化工具，侧重提交等操作 |
| git help | 帮助 |
| git init | 版本库初始化 |
| git init-db\* | 同义词，等同于 git init |
| git log | 显示提交日志 |
| git merge | 分支合并 |
| git mergetool | 图形化冲突解决 |
| git mv | 重命名 |
| git pull | 拉回远程版本库的提交 |
| git push | 推送至远程版本库 |
| git rebase | 分支变基 |
| git rebase–interactive | 交互式分支变基 |
| git reflog | 分支等引用变更记录管理 |
| git remote | 远程版本库管理 |
| git repo-config\* | 同义词，等同于 git config |
| git reset | 重置改变分支“游标”指向 |
| git rev-parse | 将各种引用表示法转换为哈希值等 |
| git revert | 反转提交 |
| git rm | 删除文件 |
| git show | 显示各种类型的对象 |
| git stage\* | 同义词，等同于 git add |
| git stash | 保存和恢复进度 |
| git status | 显示工作区文件状态 |
| git tag | 里程碑管理 |

2、对象库操作相关命令

| **命令** | **简要说明** |
| --- | --- |
| git commit-tree | 从树对象创建提交 |
| git hash-object | 从标准输入或文件计算哈希值或创建对象 |
| git ls-files | 显示工作区和暂存区文件 |
| git ls-tree | 显示树对象包含的文件 |
| git mktag | 读取标准输入创建一个里程碑对象 |
| git mktree | 读取标准输入创建一个树对象 |
| git read-tree | 读取树对象到暂存区 |
| git update-index | 工作区内容注册到暂存区及暂存区管理 |
| git unpack-file | 创建临时文件包含指定 blob 的内容 |
| git write-tree | 从暂存区创建一个树对象 |

3、引用操作相关命令

| **命令** | **简要说明** |
| --- | --- |
| git check-ref-format | 检查引用名称是否符合规范 |
| git for-each-ref | 引用迭代器，用于shell编程 |
| git ls-remote | 显示远程版本库的引用 |
| git name-rev | 将提交ID显示为友好名称 |
| git peek-remote\* | 过时命令，请使用 git ls-remote |
| git rev-list | 显示版本范围 |
| git show-branch | 显示分支列表及拓扑关系 |
| git show-ref | 显示本地引用 |
| git symbolic-ref | 显示或者设置符号引用 |
| git update-ref | 更新引用的指向 |
| git verify-tag | 校验 GPG 签名的Tag |

4、版本库管理相关命令

| **命令** | **简要说明** |
| --- | --- |
| git count-objects | 显示松散对象的数量和磁盘占用 |
| git filter-branch | 版本库重构 |
| git fsck | 对象库完整性检查 |
| git fsck-objects\* | 同义词，等同于 git fsck |
| git gc | 版本库存储优化 |
| git index-pack | 从打包文件创建对应的索引文件 |
| git lost-found\* | 过时，请使用 git fsck –lost-found 命令 |
| git pack-objects | 从标准输入读入对象ID，打包到文件 |
| git pack-redundant | 查找多余的 pack 文件 |
| git pack-refs | 将引用打包到 .git/packed-refs 文件中 |
| git prune | 从对象库删除过期对象 |
| git prune-packed | 将已经打包的松散对象删除 |
| git relink | 为本地版本库中相同的对象建立硬连接 |
| git repack | 将版本库未打包的松散对象打包 |
| git show-index | 读取包的索引文件，显示打包文件中的内容 |
| git unpack-objects | 从打包文件释放文件 |
| git verify-pack | 校验对象库打包文件 |

5、数据传输相关命令

| **命令** | **简要说明** |
| --- | --- |
| git fetch-pack | 执行 git fetch 或 git pull 命令时在本地执行此命令，用于从其他版本库获取缺失的对象 |
| git receive-pack | 执行 git push 命令时在远程执行的命令，用于接受推送的数据 |
| git send-pack | 执行 git push 命令时在本地执行的命令，用于向其他版本库推送数据 |
| git upload-archive | 执行 git archive –remote 命令基于远程版本库创建归档时，远程版本库执行此命令传送归档 |
| git upload-pack | 执行 git fetch 或 git pull 命令时在远程执行此命令，将对象打包、上传 |

6、邮件相关命令

| **命令** | **简要说明** |
| --- | --- |
| git imap-send | 将补丁通过 IMAP 发送 |
| git mailinfo | 从邮件导出提交说明和补丁 |
| git mailsplit | 将 mbox 或 Maildir 格式邮箱中邮件逐一提取为文件 |
| git request-pull | 创建包含提交间差异和执行PULL操作地址的信息 |
| git send-email | 发送邮件 |

7、协议相关命令

| **命令** | **简要说明** |
| --- | --- |
| git daemon | 实现Git协议 |
| git http-backend | 实现HTTP协议的CGI程序，支持智能HTTP协议 |
| git instaweb | 即时启动浏览器通过 gitweb 浏览当前版本库 |
| git shell | 受限制的shell，提供仅执行Git命令的SSH访问 |
| git update-server-info | 更新哑协议需要的辅助文件 |
| git http-fetch | 通过HTTP协议获取版本库 |
| git http-push | 通过HTTP/DAV协议推送 |
| git remote-ext | 由Git命令调用，通过外部命令提供扩展协议支持 |
| git remote-fd | 由Git命令调用，使用文件描述符作为协议接口 |
| git remote-ftp | 由Git命令调用，提供对FTP协议的支持 |
| git remote-ftps | 由Git命令调用，提供对FTPS协议的支持 |
| git remote-http | 由Git命令调用，提供对HTTP协议的支持 |
| git remote-https | 由Git命令调用，提供对HTTPS协议的支持 |
| git remote-testgit | 协议扩展示例脚本 |

8、版本库转换和交互相关命令

| **命令** | **简要说明** |
| --- | --- |
| git archimport | 导入Arch版本库到Git |
| git bundle | 提交打包和解包，以便在不同版本库间传递 |
| git cvsexportcommit | 将Git的一个提交作为一个CVS检出 |
| git cvsimport | 导入CVS版本库到Git。或者使用 cvs2git |
| git cvsserver | Git的CVS协议模拟器，可供CVS命令访问Git版本库 |
| git fast-export | 将提交导出为 git-fast-import 格式 |
| git fast-import | 其他版本库迁移至Git的通用工具 |
| git svn | Git 作为前端操作 Subversion |

9、合并相关的辅助命令

| **命令** | **简要说明** |
| --- | --- |
| git merge-base | 供其他脚本调用，找到两个或多个提交最近的共同祖先 |
| git merge-file | 针对文件的两个不同版本执行三向文件合并 |
| git merge-index | 对index中的冲突文件调用指定的冲突解决工具 |
| git merge-octopus | 合并两个以上分支。参见 git merge 的octopus合并策略 |
| git merge-one-file | 由 git merge-index 调用的标准辅助程序 |
| git merge-ours | 合并使用本地版本，抛弃他人版本。参见 git merge 的ours合并策略 |
| git merge-recursive | 针对两个分支的三向合并。参见 git merge 的recursive合并策略 |
| git merge-resolve | 针对两个分支的三向合并。参见 git merge 的resolve合并策略 |
| git merge-subtree | 子树合并。参见 git merge 的 subtree 合并策略 |
| git merge-tree | 显式三向合并结果，不改变暂存区 |
| git fmt-merge-msg | 供执行合并操作的脚本调用，用于创建一个合并提交说明 |
| git rerere | 重用所记录的冲突解决方案 |

10、 杂项

| **命令** | **简要说明** |
| --- | --- |
| git bisect–helper | 由 git bisect 命令调用，确认二分查找进度 |
| git check-attr | 显示某个文件是否设置了某个属性 |
| git checkout-index | 从暂存区拷贝文件至工作区 |
| git cherry | 查找没有合并到上游的提交 |
| git diff-files | 比较暂存区和工作区，相当于 git diff –raw |
| git diff-index | 比较暂存区和版本库，相当于 git diff –cached –raw |
| git diff-tree | 比较两个树对象，相当于 git diff –raw A B |
| git difftool–helper | 由 git difftool 命令调用，默认要使用的差异比较工具 |
| git get-tar-commit-id | 从 git archive 创建的 tar 包中提取提交ID |
| git gui–askpass | 命令 git gui 的获取用户口令输入界面 |
| git notes | 提交评论管理 |
| git patch-id | 补丁过滤行号和空白字符后生成补丁唯一ID |
| git quiltimport | 将Quilt补丁列表应用到当前分支 |
| git replace | 提交替换 |
| git shortlog | 对 git log 的汇总输出，适合于产品发布说明 |
| git stripspace | 删除空行，供其他脚本调用 |
| git submodule | 子模组管理 |
| git tar-tree | 过时命令，请使用 git archive |
| git var | 显示 Git 环境变量 |
| git web–browse | 启动浏览器以查看目录或文件 |
| git whatchanged | 显示提交历史及每次提交的改动 |
| git-mergetool–lib | 包含于其他脚本中，提供合并/差异比较工具的选择和执行 |
| git-parse-remote | 包含于其他脚本中，提供操作远程版本库的函数 |
| git-sh-setup | 包含于其他脚本中，提供 shell 编程的函数库 |

下面脚本之家小编特为大家分享一个图片版的

Git 常用命令速查表。点击查看大图。

[](http://files.jb51.net/file_images/article/201409/git_big_jb51.jpg)

Git命令参考手册(文本版)

git init                                                  # 初始化本地git仓库（创建新仓库）   
git config --global user.name "xxx"                       # 配置用户名   
git config --global user.email "xxx@xxx.com"              # 配置邮件   
git config --global color.ui true                         # git status等命令自动着色   
git config --global color.status auto   
git config --global color.diff auto   
git config --global color.branch auto   
git config --global color.interactive auto   
git clone git+ssh://git@192.168.53.168/VT.git             # clone远程仓库   
git status                                                # 查看当前版本状态（是否修改）   
git add xyz                                               # 添加xyz文件至index   
git add .                                                 # 增加当前子目录下所有更改过的文件至index   
git commit -m 'xxx'                                       # 提交   
git commit --amend -m 'xxx'                               # 合并上一次提交（用于反复修改）   
git commit -am 'xxx'                                      # 将add和commit合为一步   
git rm xxx                                                # 删除index中的文件   
git rm -r \*                                               # 递归删除   
git log                                                   # 显示提交日志   
git log -1                                                # 显示1行日志 -n为n行   
git log -5  
git log --stat                                            # 显示提交日志及相关变动文件   
git log -p -m   
git show dfb02e6e4f2f7b573337763e5c0013802e392818         # 显示某个提交的详细内容   
git show dfb02                                            # 可只用commitid的前几位   
git show HEAD                                             # 显示HEAD提交日志   
git show HEAD^                                            # 显示HEAD的父（上一个版本）的提交日志 ^^为上两个版本 ^5为上5个版本   
git tag                                                   # 显示已存在的tag   
git tag -a v2.0 -m 'xxx'                                  # 增加v2.0的tag   
git show v2.0                                             # 显示v2.0的日志及详细内容   
git log v2.0                                              # 显示v2.0的日志   
git diff                                                  # 显示所有未添加至index的变更   
git diff --cached                                         # 显示所有已添加index但还未commit的变更   
git diff HEAD^                                            # 比较与上一个版本的差异   
git diff HEAD -- ./lib                                    # 比较与HEAD版本lib目录的差异   
git diff origin/master..master                            # 比较远程分支master上有本地分支master上没有的   
git diff origin/master..master --stat                     # 只显示差异的文件，不显示具体内容   
git remote add origin git+ssh://git@192.168.53.168/VT.git # 增加远程定义（用于push/pull/fetch）   
git branch                                                # 显示本地分支   
git branch --contains 50089                               # 显示包含提交50089的分支   
git branch -a                                             # 显示所有分支   
git branch -r                                             # 显示所有原创分支   
git branch --merged                                       # 显示所有已合并到当前分支的分支   
git branch --no-merged                                    # 显示所有未合并到当前分支的分支   
git branch -m master master\_copy                          # 本地分支改名   
git checkout -b master\_copy                               # 从当前分支创建新分支master\_copy并检出   
git checkout -b master master\_copy                        # 上面的完整版   
git checkout features/performance                         # 检出已存在的features/performance分支   
git checkout --track hotfixes/BJVEP933                    # 检出远程分支hotfixes/BJVEP933并创建本地跟踪分支   
git checkout v2.0                                         # 检出版本v2.0  
git checkout -b devel origin/develop                      # 从远程分支develop创建新本地分支devel并检出   
git checkout -- README                                    # 检出head版本的README文件（可用于修改错误回退）   
git merge origin/master                                   # 合并远程master分支至当前分支   
git cherry-pick ff44785404a8e                             # 合并提交ff44785404a8e的修改   
git push origin master                                    # 将当前分支push到远程master分支   
git push origin :hotfixes/BJVEP933                        # 删除远程仓库的hotfixes/BJVEP933分支   
git push --tags                                           # 把所有tag推送到远程仓库   
git fetch                                                 # 获取所有远程分支（不更新本地分支，另需merge）   
git fetch --prune                                         # 获取所有原创分支并清除服务器上已删掉的分支   
git pull origin master                                    # 获取远程分支master并merge到当前分支   
git mv README README2                                     # 重命名文件README为README2   
git reset --hard HEAD                                     # 将当前版本重置为HEAD（通常用于merge失败回退）   
git rebase   
git branch -d hotfixes/BJVEP933                           # 删除分支hotfixes/BJVEP933（本分支修改已合并到其他分支）   
git branch -D hotfixes/BJVEP933                           # 强制删除分支hotfixes/BJVEP933   
git ls-files                                              # 列出git index包含的文件   
git show-branch                                           # 图示当前分支历史   
git show-branch --all                                     # 图示所有分支历史   
git whatchanged                                           # 显示提交历史对应的文件修改   
git revert dfb02e6e4f2f7b573337763e5c0013802e392818       # 撤销提交dfb02e6e4f2f7b573337763e5c0013802e392818   
git ls-tree HEAD                                          # 内部命令：显示某个git对象   
git rev-parse v2.0                                        # 内部命令：显示某个ref对于的SHA1 HASH   
git reflog                                                # 显示所有提交，包括孤立节点   
git show HEAD@{5}   
git show master@{yesterday}                               # 显示master分支昨天的状态   
git log --pretty=format:'%h %s' --graph                   # 图示提交日志   
git show HEAD~3  
git show -s --pretty=raw 2be7fcb476   
git stash                                                 # 暂存当前修改，将所有至为HEAD状态   
git stash list                                            # 查看所有暂存   
git stash show -p stash@{0}                               # 参考第一次暂存   
git stash apply stash@{0}                                 # 应用第一次暂存   
git grep "delete from"                                    # 文件中搜索文本“delete from”   
git grep -e '#define' --and -e SORT\_DIRENT   
git gc   
git fsck