# Git常用命令总结

**--by qiaozq**

[一、 Git安装及配置 3](#_Toc459897309)

[1. 安装 3](#_Toc459897310)

[2. 初始配置 3](#_Toc459897311)

[二、 搭建GIT服务器 5](#_Toc459897312)

[1. 生成SSH公钥 5](#_Toc459897313)

[2. 配置服务器 6](#_Toc459897314)

[三、 远程仓库 7](#_Toc459897315)

[1. 查看远程仓库 7](#_Toc459897316)

[2. 添加远程仓库 8](#_Toc459897317)

[3. 从远程仓库中拉取 8](#_Toc459897318)

[4. 推送到远程仓库 8](#_Toc459897319)

[5. 远程仓库的移除和重命名 8](#_Toc459897320)

[6. 修改远程仓库的地址 8](#_Toc459897321)

[四、 Git版本库基本操作 9](#_Toc459897322)

[1. 添加与提交 9](#_Toc459897323)

[2. 查看提交历史 9](#_Toc459897324)

[3. 查看文件差异 10](#_Toc459897325)

[4. 撤销操作 11](#_Toc459897326)

[5. 版本回退 11](#_Toc459897327)

[五、 Git分支管理 11](#_Toc459897328)

[1. 分支的新建与合并 11](#_Toc459897329)

[2. 解决冲突 12](#_Toc459897330)

[3. 分支管理 12](#_Toc459897331)

[4. 分支开发工作流 12](#_Toc459897332)

[六、 Git标签管理 13](#_Toc459897333)

[1. 创建标签 13](#_Toc459897334)

[2. 删除标签 14](#_Toc459897335)

[3. 推送标签 14](#_Toc459897336)

[七、 自定义Git 14](#_Toc459897337)

[1. 忽略特定文件 14](#_Toc459897338)

[八、 常见问题分析 15](#_Toc459897339)

[1. 换行符转换问题 15](#_Toc459897340)

[2. 拉取远程分支时报错 15](#_Toc459897341)

## Git安装及配置

### 安装

#### Linux下安装

$ sudo yum git

#### **Windows下安装**

<https://git-scm.com/download/win>

### 初始配置

#### 基础配置

$ git config --global user.name “qiaozq”

$ git config --global user.email [qiaozhiqi@163.com](mailto:qiaozhiqi@163.com)

#### 常用配置

$ git config --global core.edit vim //git默认文本编辑器

$ git config --global color.ui true //打开终端着色

$ git config --list //检查配置信息

#### 配置别名

$ git config --global alias.st status

$ git config --global alias.co checkout

$ git config --global alias.ci commit

$ git config --global alias.br branch

$ git config --global alias.unstage ‘reset HEAD’ //撤销暂存区的修改，重新放回工作区，e.g: $ git unstage test.c

$ git config --global alias.last ‘log -1’ //显示最近一次的提交

$ git config --global alias.lg "log --color --graph --abbrev-commit --pretty='%Cred[%h] %Cgreen-%an- <%cd> [%ar] %n %Cblue## %Creset%s%n'" //个性化日志显示格式

$ git config --global alias.difflist 'diff --name-status'

//显示修改文件列表

#### WIN下的配置

##### Windows下配置Byond Compare 4作为比较工具

Windows下配置文件一般在用户数据目录，如C:\Users\qiaozq，名为.gitconfig，配置BC4作为比较工具，配置文件参见：

[user]

name = joe

email = qiaozhiqi@163.com

[diff]

tool = bc3

[difftool "bc3"]

prompt = false

[difftool]

path = C:/Program Files/Beyond Compare 4/BCompare.exe

[merge]

tool = bc3

[mergetool]

prompt = false

[mergetool "bc3"]

path = C:/Program Files/Beyond Compare 4/BCompare.exe

Note: Git versions older than 2.2.0 (git --version) use "bc3" as the keyword for BC4. For Git 2.2.0+, use "bc".

## 搭建GIT服务器

远程git服务器可以用来协作，也可以用来过渡或者作为远程备份。以下是在centos系统中搭建git服务器的步骤：

### 生成SSH公钥

Git服务器使用SSH公钥进行认证，Git服务器中保存可以访问服务器的用户的公钥列表，其他git用户可在将自己的公钥文件（一般是id\_dsa.pub）中的内容加入到服务器的授权文件中（authorized\_keys），该用户即可访问Git服务器，push/pull内容。

$ cd ~/.ssh //查看是否有该目录，如没有可新建

$ ssh-keygen //生成key文件，id\_rsa为私钥文件，id\_rsa.pub为公钥文件

Generating public/private rsa key pair.

Enter file in which to save the key (/home/schacon/.ssh/id\_rsa):

Created directory '/home/schacon/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/schacon/.ssh/id\_rsa.

Your public key has been saved in /home/schacon/.ssh/id\_rsa.pub.

The key fingerprint is:

d0:82:24:8e:d7:f1:bb:9b:33:53:96:93:49:da:9b:e3 schacon@mylaptop.local

### 配置服务器

#### 创建用户及目录

$ sudo adduser git //创建用户

$ su git

$ cd

$ mkdir .ssh && chmod 700 .ssh

$ touch .ssh/authorized\_keys && chmod 600 .ssh/authorized\_keys

#### **添加证书**

将用户公钥添加到authorized\_keys文件末尾，一个用户一行。

#### 初始化git仓库

$ sudo git init --bare hsiar\_source.git //裸仓库通常以git结尾

$ sudo chown -R git:git hsiar\_source.git //owner改为git

#### 禁用shell登陆

出于安全考虑，第⼆步创建的git⽤户不允许登录shell，这可以通过编辑/etc/passwd⽂件完成。找到类似下⾯的⼀⾏：

git:x:1001:1001:,,,:/home/git:/bin/bash

改为：

git:x:1001:1001:,,,:/home/git:/usr/bin/git-shell

这样，git⽤户可以正常通过ssh使⽤git，但⽆法登录shell，因为我们为git⽤户指定的gitshell每次⼀登录就⾃动退出。

#### 推送到远程仓库

可以将某仓库的内容推送到远程仓库，以建立联系。例如，在win上的某git目录下，运行git bash：

$ git remote add origin [git@192.168.94.117:/home/git/hsiar.git](mailto:git@192.168.94.117:/home/git/hsiar.git)

$ git push -u origin master //将本地库的master分支推送到远程，由于是第一次崔松，加上-u参数，则将本地master分支和远程master分支关联起来

$ git push origin master //将本地master分支最新修改推送到远程服务器

#### 从远程仓库克隆

$ git clone [git@192.168.94.117:/home/git/hsiar.git](mailto:git@192.168.94.117:/home/git/hsiar.git)

## 远程仓库

### 查看远程仓库

$ git remote -v

$ git remote show origin //查看远程仓库origin的详细信息

$ git branch -r //查看远程仓库有哪些分支

### 添加远程仓库

$ git remote add <shortname> <url>

### 从远程仓库中拉取

$ git fetch [remote-name] //抓取但不合并

$ git pull [remote-name] //抓取并合并

### 推送到远程仓库

$ git push [remote-name] [branch-name] //将当前branch-name分支推送到远程仓库remote-name

$ git push origin master //master分支推送到远程仓库

$ git push origin dev //dev分支推送到远程仓库

$ git push -f origin feature01 //如果feature01使用--amend参数重写过，那么使用-f参数强制覆盖远程版本

### 远程仓库的移除和重命名

$ git remote rename org\_name new\_name //重命名

$ git remote rm name //删除

### **修改远程仓库的地址**

##### 直接重置地址

$ git remote set-url origin [git@new\_ip:/home/git/hsiar\_src.git](mailto:git@new_ip:/home/git/hsiar_src.git)

##### 先删除后添加

$ git remote rm origin

$ git remote add origin [git@new\_ip:/home/git/hsiar\_source.git](mailto:git@new_ip:/home/git/hsiar_source.git)

注意：修改后分支的对应关系可能改变，需要重新建立连接，按照提示操作即可。

## Git版本库基本操作

### 添加与提交

$ git add \* //添加所有文件，加入跟踪

$ git add filename //添加某个文件

$ git commit -m “log msg” //-m表示message，提交暂存的文件

$ git commit -a -m “log msg” //将所有修改的文件暂存并提交

$ git commit -e //提交暂存，打开编辑器编辑说明信息，-e表示edit

$ git status //查看状态

$ git status -s //紧凑格式输出当前状态

$ git rm file //移除文件

$ git mv file\_from file\_to //移动文件

### 查看提交历史

$ git lg filename //查看单个文件的历史

$ git show commit\_id //查看具体某次的修改

$ git log -2 //查看最近两次的提交

$ git log --pretty=oneline --abbrev-commit

//每行显示一条，包含commit-id

$ git log --color --graph --abbrev-commit --pretty="%Cred[%h] %Cgreen-%an- <%cd> [%ar] %n %Cblue## %Creset%s%n" //个性化日志输出格式

$ git log --until=1.day.ago //一分钟之前的所有log

$ git log --since=2.days.ago //两天之内的所有log

$ git log --since=1.week.ago //一周之内的所有log

$ git log --since=1.month.ago --until=2.weeks.ago

//一个月之前到半个月之前的所有log

$ git log --since=2016-06-01 --until=2016-06-06 //某个时间段的log

### 查看文件差异

$ git difftool //比较工作目录与暂存区的差别

$ git difftool filename //查看尚未暂存的某文件的修改

$ git difftool --cached //查看暂存的文件和上次提交的差异

$ git difftool --cached filename //查看暂存的某文件和上次提交的差异

$ git difftool commit\_id1 commit\_id2//查看两个历史版本的修改

$ git difftool commit\_id1 commit\_id2 -- filename //查看某特定文件的两个历史版本的修改

$ git difftool commit\_id //比较某个历史版本和当前工作区的差异

$ git difftool HEAD //比较工作区和上次提交的差异

$ git diff --name-status commit\_id1 commit\_id2 //查看两个版本修改的文件列表及文件的增删改状态

### 撤销操作

$ git commit --amend //如果忘了暂存某些修改，该命令可将该文件添加到上一次的提交中，如果没有修改，则会启动编辑器编辑上一次的提交说明

$ git reset HEAD <file> //取消暂存

$ git checkout -- [file] //撤销还未提交的修改

$ git ls-files -d | xargs -i git checkout {} //恢复误删除的文件

### 版本回退

$ git reset --hard HEAD^ //回退到上一个版本

$ git reset --hard <commit\_id> //回退到某个特定的版本，commit\_id可通过git reflog查看，即使回退到了某一个版本，在log中无法查新于该版本的commit id，也可以通过reflog查看

$ git reset --soft HEAD^ //将最近一次的提交回退到暂存区

## Git分支管理

### 分支的新建与合并

$ git branch //查看分支

$ git branch -v //查看分支并显示最后一次提交信息

$ git branch -r //查看远程分支

$ git branch name //新建name分支

$ git checkout name //切换到name分支

$ git checkout -b name //创建并切换到name分支

$ git merge name //合并name分支到当前分支

$ git merge --no-ff -m “commit msg” name //禁止快速合并模式，将name分支合并到当前分支，可查看到合并记录

$ git branch -d name //删除name分支

$ git branch -D name //强制删除分支

$git co --patch branch plugins/xxx/file.c //merge单个文件

//将branch分支中的file.c文件单独合并到当前分支，选择’a’确认，然后提交

### 解决冲突

$ git mergetool //如果有冲突，该命令会调用指定的merge tool，合并完成后add并提交

### 分支管理

$ git branch --merge //查看哪些分支已合并到当前分支

$ git branch --no-merged //查看所有包含未合并工作的分支

### 分支开发工作流

#### Bug分支，暂存工作状态

如果在某个分支上开发到一半，需要紧急修复bug，新建分支后修改的内容还在，所以需要先储藏而使得工作区干净，此时可以使用stash命令。

$ git stash //将当前工作状态暂存起来

$ git stash pop //出去最近一次的暂存并丢弃记录

$ git stash list //查看暂存的状态

$ git stash apply //取出最近的一次暂存

$ git stash apply stash@{1} //取出特定的暂存

$ git stash drop stash@{2} //丢弃某次暂存

#### 特性分支

按照正常的分支开发，合并即可

#### 远程分支

$ git checkout -b branch-name origin/branch-name

//在本地创建和远程分支对应的分支

$ git branch --set-upstream branch-name origin/branch-name

//建立本地分支和远程分支的关联

$git push origin --delete branch-name

//删除远程分支

## Git标签管理

### 创建标签

$ git tag v1.0 //在最新的commit上打标签

$ git tag //查看所有标签

$ git log --pretty=oneline --abbrev-commit //显示commit-id

$ git tag v0.9 6a7893e //对特定提交打标签

$ git show v0.9 //显示标签信息

$ git tag -a v0.1 -m “tag msg” 8s783a9

//创建带有说明的标签，-a指定标签名，-m指定说明文字

### 删除标签

$ git tag -d v1.0 //删除标签

### 推送标签

$ git push origin v1.0 //推送标签v1.0到远程

$ git push origin --tags //一次性推送尚未推送的标签到远程

$ git push origin :refs/tags/tagname //删除远程标签

## 自定义Git

### 忽略特定文件

在.gitignore文件中指定不需要跟踪的文件列表。

官方提供的列表目录：[gitignore文件列表](https://github.com/github/gitignore)。

## 常见问题分析

### 换行符转换问题

Windows平台以CRLF为行结尾符，Linux/Unix平台以LF为行结尾符，在跨平台开发时可能会导致问题。

$ git config --global core.autocrlf true/input/false

该配置用于CRLF/LF的自动转换，设置为true时会在签出时将LF转换为CRLF，提交时将CRLF转换为LF。input表示提交时将CRLF转换成LF，签出时不转换。false表示保留原来的结束符。

以目前的开发方式，建议在Windows上设置为false，然后尽量使用LF作为结束符开发。包括sourceinsight上设置结束符为LF。

### 拉取远程分支时报错

使用命令：

$ git checkout -b local-name origin/remote-name

时，报错：

fatal: git checkout: updating paths is incompatible with switching branches.  
Did you intend to checkout 'origin/remote-name' which can not be resolved as commit?

原因：该远程分支在本地仓库中未跟踪，可使用以下命令查看：

$ git remote show origin

如果该远程分支在”new”那一行，例如：

dev tracked

feature\_01\_signature new (next fetch will store in remotes/origin)

则先运行如下命令：

$ git remote update

$ git fetch

### **Win下bash速度变慢**

解决方案：关闭360安全卫士。