**Git常用命令**

[--文件目录操作命令](#1267-1510721048359)

[--git初始化操作](#3341-1510721392683)

[--git 克隆分支](#9621-1510721528100)

[--clone 远程分支](#9832-1510721609181)

[--查看命令](#8899-1510721654506)

[--版本回退](#1680-1510721661009)

[--撤销修改](#0089-1510721665834)

[--分支管理](#9062-1510721691346)

[--tag相关操作](#5249-1510721692097)

[--隐藏的文件](#3544-1510721692097)

[--查看远程库信息(git  remote的用法)](#6332-1510721692097)

[--将远程分支拉取到本地](#4939-1510721692097)

[--git pull操作](#8253-1510721692097)

[--git 设置大小写敏感](#2810-1510721692097)

[--git 设置忽略文件或文件夹权限修改](#1971-1510721692097)

[--创建追踪分支](#1041-1510721692097)

[--切换git 命令提示中文到英文](#3913-1510721692097)

[--git 删除未添加到版本中的文件或者文件夹](#4531-1510721692097)

[--git常用错误](#8590-1510721883016)

**--文件目录操作命令**

1 mkdir \* 创建一个空目录 \*指目录名
2 pwd 显示当前目录的路径。
3 cat \* 查看\*文件内容
4 git rm \* 删除\*\*文件

**--git初始化操作**

1 git init 把当前的目录变成git仓库，生成隐藏.git文件。
2 git remote add origin url 把本地仓库的内容推送到GitHub仓库。
3 git clone git@url/test.git 从远程库克隆
4 git add \* 把x文件添加到暂存区去。
5 git commit –m "\*" 提交文件 –m 后面的是注释。

**--git 克隆分支**

1 git clone xxx.git 最简单直接的命令
2 git clone xxx.git "指定目录" clone到指定目录
3 git clone -b branchname xxx.git clone时创建新的分支替代默认Origin HEAD（master）

**--clone 远程分支**

　　**git clone 命令默认的只会建立master分支**，如果你想clone指定的某一远程分支(如：dev)的话，可以如下：

　　**1. 查看所有分支(包括隐藏的)  git branch -a 显示所有分支**

1 \* master
2 remotes/origin/HEAD -> origin/master
3 remotes/origin/dev
4 remotes/origin/master

　　**2.** **在本地新建同名的("dev")分支，并切换到该分支**

1 git checkout -t origin/dev 该命令等同于：
2 git checkout -b dev origin/dev

**--查看命令**

1 git status 查看仓库状态
2 git diff \* 查看X文件修改了那些内容
3 git log 查看历史记录
4 git reflog 查看历史记录的版本号id（记录你的每一次命令,不论是否提交）
5 git log --pretty=oneline 如果信息量太多可以进行比较好的列表显示

**--版本回退**

1 git reset –hard HEAD^ 回退到上一个版本
2 git reset --hard HEAD~第几个 如果想回退到第3个版本，使用git reset –hard HEAD~3
3 git reset --hard 057d 回退到某一个具体的版本号

**--撤销修改**

1 git checkout file-name 恢复某个已修改的文件（撤销未提交的修改）：
2 git revert HEAD 还原最近一次提交的修改：
3 git revert commit-id 还原指定版本的修改

**--分支管理**

1 git branch 查看本地所有的分支
2 git branch -a 查看远程所有的分支
3 git branch name 创建分支
4 git branch –d dev 删除dev分支
5 git push origin --delete dev 删除远程的dev分支
6 git branch -m dev develop 重命名分支
7 git checkout –b dev 创建dev分支 并切换到dev分支上
8 git merge dev 在当前分支上合并dev分支代
9 git push origin zyf-dev 把当前新疆的zyf-dev分支推送到远程库(远程仓库没有给分支则会新建立该分支)
10 git checkout — \* 把XX文件在工作区的修改全部撤销。
11 git checkout master 切换回master分支
12 git push --set-upstream origin dev 提交修改并创建远程分支dev

**--tag相关操作**

1 git tag 列出所有的tag
2 git tag name 打轻量标签 name
3 git tag -d name 删除本地的tag
4 git push origin --delete tag name 删除远程的tag
5 git show name 查看tag信息
6 git push origin name 将tag提交到远程

**--隐藏的文件**

1 git stash 把当前的工作隐藏起来 等以后恢复现场后继续工作
2 git stash list 查看所有被隐藏的文件列表
3 git stash apply 恢复被隐藏的文件，但是内容不删除
4 git stash drop 删除文件
5 git stash pop 恢复文件的同时 也删除文件

**--查看远程库信息(git  remote的用法)**

1 git remote 查看远程库的信息
2 git remote –v 查看远程库的详细信息
3 git remote add name url 添加远程仓库
4 git remote rename oldname newname 重命名仓库
5 git remote rm 删除仓库

**--将远程分支拉取到本地**

1 方法一：git checkout -b 本地分支名x origin/远程分支名x
2 方式二：git fetch origin 远程分支名x:本地分支名x

**--git pull操作**

1 git pull命令的作用是，取回远程主机某个分支的更新，再与本地的指定分支合并，基本的格式如下。
2 $ git pull <远程主机名> <远程分支名>:<本地分支名>
3
4 取回origin主机的next分支，与本地的master分支合并，需要写成下面这样
5 $ git pull origin next:master
6
7 如果远程分支是与当前分支合并，则冒号后面的部分可以省略。
8 $ git pull origin next
9
10 上面命令表示，取回origin/next分支，再与当前分支合并。实质上，这等同于先做git fetch，再做git merge。
11 $ git fetch origin
12 $ git merge origin/next
13 在某些场合，Git会自动在本地分支与远程分支之间，建立一种追踪关系(tracking)。比如，在git clone的时候，所有本地分支默认与远程主机的同名分支，建立追踪关系，也就是说，本地的master分支自动”追踪”origin/master分支。
14 Git也允许手动建立追踪关系。
15 git branch --set-upstream master origin/next
16
17 上面命令指定master分支追踪origin/next分支。如果当前分支与远程分支存在追踪关系，git pull就可以省略远程分支名。
18 $ git pull origin

**--git 设置大小写敏感**

Windows上的Git默认是大小写不敏感的，这样多平台写作就可能会出现问题。Win上的Git设置为大小写敏感的命令如下

1 git config core.ignorecase false

**--git 设置忽略文件或文件夹权限修改**

1 git config core.filemode false

**--创建追踪分支**

不带任何参数的git push，默认只推送当前分支，这叫做simple方式。此外，还有一种matching方式，会推送所有有对应的远程分支的本地分支。Git 2.0版本之前，默认采用matching方法，现在改为默认采用simple方式。如果要修改这个设置，可以采用git config命令。

$ git config --global push.default matching
# 或者
$ git config --global push.default simple (最好使用这种方式)

  $ git branch --track  master origin/master //在使用 git branch 命令时加上 '--track' 参数, 来手动创建一个追踪分支

**--切换git 命令提示中文到英文**

// ubuntu装的git不知道怎么就出现全中文的提示,不太好,果断切换到中文了,切换方法如下:

// 1:写入
echo "alias git='LANG=en\_GB git'" >> ~/.bashrc
// 2:生效
source ~/.bashrc

**--git 删除未添加到版本中的文件或者文件夹**

git checkout 只能回退在版本中的修改或者删除, 对于新添加的文件是没有作用的, 也就是说, 新建的文件或者文件夹是:Untracked files, 要删除或者清理掉这些文件,需要使用 git clean 命令:

1 // 删除 untracked files
2 git clean -f
3
4 // 连 untracked 的目录也一起删掉
5 git clean -fd
6
7 // 连 gitignore的untrack 文件/目录也一起删掉 （一般这个是用来删掉编译出来的 .o一类的文件）
8 git clean -xfd
9
10 // 在使用清理 git clean之前，建议加上 -n 来先看看会删掉哪些文件，防止重要文件被误删
11 git clean -nxfd
12 git clean -nf
13 git clean -nfd

**--git常用错误**

**1：There is no tracking information for the current branch...**

    则说明本地分?支和远程分?支的链接关系没有创建，用命令：

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

**2: ![rejected] dev -> dev (non-fast-forward)  ... Updates were rejected because the tip of your current branch**

    推送失败，因为远程代码的最新提交和你试图推送的提交有冲突，解决办法也很简单，先用git pull把最新的提交从origin/dev抓下来，然后，在本地合并，解决冲突，再推送

**3：CONFLICT (content): Merge conflict in . . .**

    这回git pull成功，但是合并有冲突，需要手动解决，解决的方法和分支管理中的解决冲突一样。解决后，提交，再push：

**4: You are not currently on a branch, so I cannot use any**

   解决办法：git checkout master

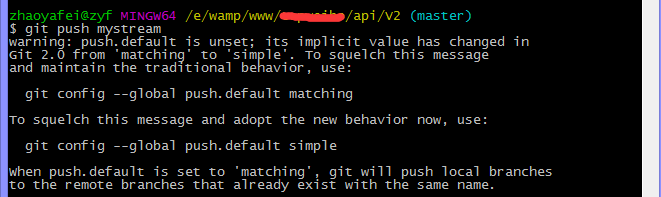
**5：Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.**

   上面的显示的意思是，有一个更新还没有反应到本地来，可能是别人往server上提交了一点东西。

   可以使用git pull命令拿这些更新到本地来。

**6：在执行 git push 时可能会看到如下消息:**

warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:
git config --global push.default matching
To squelch this message and adopt the new behavior now, use:
git config --global push.default simpl



  解决办法：‘matching’ 参数是 Git 1.x 的默认行为，如果你执行 git push 但没有指定分支，它将 push 所有你本地的分支到远程仓库中对应匹配的分支。而 Git 2.x 默认的是 simple，意味着执行 git push 没有指定分支时，只有当前分支会被 push 到你使用 git pull 获取的代码。

根据提示，修改git push:

1 git config --global push.default matching

再次执行git push 就行了。

**7：You asked to pull from the remote 'origin', but did not specify:a branch. Because this is not the default configured remotefor your current branch, you must specify a branch on the command line.**

解决办法：找到：.git/config  修改如下

1 [branch "master"]
2 remote = origin
3 merge = refs/heads/master

**8：ERROR: Permission to user1/test.git denied to user2   fatal: The remote end hung up unexpectedly**

　账户冲突，下面有针对讲解

**9: 添加的ssh不起作用？**

   ssh 的添加一定要在root用户权限在添加，其他的权限不起作用，切记！