# git操作

目录

[git操作 1](#_Toc13799)

[查看ssh秘钥 2](#_Toc20650)

[迁移git操作 2](#_Toc9863)

[查看目录（隐藏文件也显示） 2](#_Toc23086)

[创建文件 3](#_Toc26737)

[使用Windows的童鞋要特别注意： 3](#_Toc10034)

[创建版本库 3](#_Toc16066)

[Example 3](#_Toc2886)

[回退版本 4](#_Toc16)

[删除文件 5](#_Toc5863)

[添加远程库 5](#_Toc9775)

[从远程库克隆 5](#_Toc8863)

[创建与合并分支 6](#_Toc32105)

[Bug分支 6](#_Toc20458)

[多人协作的工作模式 7](#_Toc30267)

[创建标签 7](#_Toc5757)

[删除标签 8](#_Toc30215)

[如果想使用其他的远程仓库 8](#_Toc26794)

[忽略特殊文件 9](#_Toc9831)

[gitignore文件本身要放到版本库里，并且可以对.gitignore做版本管理！ 9](#_Toc4082)

[更改命令 9](#_Toc32766)

# 查看ssh秘钥

1. 运行git  
   2.执行命令：ls -al ~/.ssh  查看本机是否有秘钥文件 没有秘钥文件生成秘钥文件  
   3.cd ~/.ssh   
   4.ls  
   5.cat id\_rsa.pub 即可查看本机的秘钥

可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

# 迁移git操作

git clone 老项目地址（下载老项目到本地）

cd 老项目名称

cd .git

Git push --mirror 新项目地址（前提是你已经在新的里面创建了文件，然后复制这个新文件的地址）

或者

Git clone 老项目地址

Cd老项目名称

Cd .git

Git remote rename origin old-origin

Git remote add origin 新项目的SSH地址

Git push -u origin --all

# 查看目录（隐藏文件也显示）

ls -ah

# 创建文件

mkdir filename（ps:为了避免不必要的麻烦，文件命名请不要使用中文名）

进入文件夹以后，pwd显示文件目录

git init 命令把这个目录变成Git可以管理的仓库

# 使用Windows的童鞋要特别注意：

千万不要使用Windows自带的记事本编辑任何文本文件。原因是Microsoft开发记事本的团队在每个文件开头添加了0xefbbbf（十六进制）的字符来保存utf-8文件，你会遇到很多不可思议的问题，比如，网页第一行可能会显示一个“?”，明明正确的程序一编译就报语法错误，等等，都是由记事本的行为带来的。建议你下载[Notepad++](http://notepad-plus-plus.org/" \t "https://www.liaoxuefeng.com/wiki/896043488029600/_blank)代替记事本，不但功能强大，而且免费！记得把Notepad++的默认编码设置为UTF-8 without BOM即可

# 创建版本库

(在目录下)git init 把该目录变成git可以管理的仓库

Git add filename 把文件添加到仓库（反复多次使用，添加多个文件）

Git commit -m ”本次改动内容的说明” 把文件提交到仓库（一次可提交很多文件）

Example:

$ git add file1.txt

$ git add file2.txt file3.txt

$ git commit -m "add 3 files."

git status 可以让我们时刻掌握仓库当前的状态

Git diff filename 查看该文件与上次的区别

要随时掌握工作区的状态，使用git status命令。

如果git status告诉你有文件被修改过，用git diff可以查看修改内容

Git log 查看commit的记录

$ git log --pretty=oneline 记录比较精简(版本号加修改提示信息)

# 回退版本

在Git中，用HEAD表示当前版本，也就是最新的提交，上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

git reset - -hard HEAD^ 回退到上一版本

git reset - -hard 版本号前几位 返回到该版本

git commit的反向命令git reset HEAD，就是把仓库最新版本转移到暂存区。

git reset HEAD <file> 暂存区回到工作区

cat filename 查看文件内容

git reflog 用来记录你的每一次命令

现在总结一下：

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭使用命令

git reset --hard commit\_id

穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。

要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

Git管理的文件分为：工作区，版本库，版本库又分为暂存区stage和暂存区分支master(仓库)

工作区>>>>暂存区>>>>仓库

git add把文件从工作区>>>>暂存区，git commit把文件从暂存区>>>>仓库，

git diff查看工作区和暂存区差异，

git diff --cached查看暂存区和仓库差异，

git diff HEAD 查看工作区和仓库的差异，

git add的反向命令git checkout，撤销工作区修改，即把暂存区最新版本转移到工作区，

每次修改，如果不用git add到暂存区，那就不会加入到commit中

git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。

# 删除文件

Git rm filename (操作的是工作区)

Git知道你删除了文件，因此，工作区和版本库就不一致

现在你有两个选择，一是确实要从版本库中删除该文件，那就用命令git rm filename删掉，并且git commit -m “本次删除的说明”；另一种情况是删错了，因为版本库里还有呢，所以可以很轻松地把误删的文件恢复到最新版本

git checkout -- filename,git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”

# 添加远程库

1.git remote add origin <https://github.com/yuman123/项目名.git>

2.Git push -u origin master

要关联一个远程库，使用命令1；

关联后，使用命令2第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

# 从远程库克隆

要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆。

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快

# 创建与合并分支

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>或者git switch <name>

创建+切换分支：git checkout -b <name>或者git switch -c <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

注意：因为未add的内容不属于任何一个分支， 未commit的内容也不属于任何一个分支。 也就是说，对于所有分支而言， 工作区和暂存区是公共的

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

解决冲突就是把Git合并失败的文件手动编辑为我们希望的内容，再gitadd gitcommit提交。

用git log --graph命令可以看到分支合并图。

合并分支时，加上- -no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

# Bug分支

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场；

在master分支上修复的bug，想要合并到当前dev分支，可以用git cherry-pick 版本号前几位命令，把bug提交的修改“复制”到当前分支，避免重复劳动。

开发一个新功能，最好新建一个分支；

如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。

# 多人协作的工作模式

首先，可以试图用git push origin <branch-name>推送自己的修改；

如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；如果git pull提示no tracking information，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-to <branch-name> origin/<branch-name>。

如果合并有冲突，则解决冲突，并在本地提交；

没有冲突或者解决掉冲突后，再用git push origin <branch-name>推送就能成功！

这就是多人协作的工作模式，一旦熟悉了，就非常简单。

查看远程库信息，使用git remote -v；

本地新建的分支如果不推送到远程，对其他人就是不可见的；

从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；

在本地创建和远程分支对应的分支，使用git switch -c branch-name origin/branch-name，本地和远程分支的名称最好一致；

建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；

从远程抓取分支，使用git pull，如果有冲突，要先处理冲突

# 创建标签

命令git tag <tagname>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

·命令git tag -a <tagname> -m "说明信息..."可以指定标签信息；

·命令git tag可以查看所有标签

注意：标签总是和某个commit挂钩。如果这个commit既出现在master分支，又出现在dev分支，那么在这两个分支上都可以看到这个标签

# 删除标签

Git tag -d tagname

因为创建的标签都只存储在本地，不会自动推送到远程。所以，打错的标签可以在本地安全删除。

如果要推送某个标签到远程，使用命令git push origin <tagname>：

或者，一次性推送全部尚未推送到远程的本地标签git push origin - -tags

命令git tag -d <tagname>可以删除一个本地标签；

命令git push origin :refs/tags/<tagname>可以删除一个远程标签。

# 如果想使用其他的远程仓库

git给远程库起的默认名称是origin，如果有多个远程库，我们需要用不同的名称来标识不同的远程库。

git remote add github [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git)

注意，远程库的名称叫github，不叫origin了。

接着，再关联码云的远程库：

git remote add gitee [git@gitee.com:liaoxuefeng/learngit.git](mailto:git@gitee.com:liaoxuefeng/learngit.git)

我们用git remote -v查看远程库信息，可以看到两个远程库

如果要推送到GitHub，使用命令：

git push github master

如果要推送到码云，使用命令：

git push gitee master

# 忽略特殊文件

在Git工作区的根目录下创建一个特殊的.gitignore文件，然后把要忽略的文件名填进去，Git就会自动忽略这些文件。不需要从头写.gitignore文件，GitHub已经为我们准备了各种配置文件，只需要组合一下就可以使用了。所有配置文件可以直接在线浏览：[https://github.com/github/gitignor](https://github.com/github/gitignore" \t "https://www.liaoxuefeng.com/wiki/896043488029600/_blank)

使用Windows的童鞋注意了，如果你在资源管理器里新建一个.gitignore文件，它会非常弱智地提示你必须输入文件名，但是在文本编辑器里“保存”或者“另存为”就可以把文件保存为.gitignore了

# gitignore文件本身要放到版本库里，并且可以对.gitignore做版本管理！

# 更改命令

如果敲git st就表示git status那就简单多了，当然这种偷懒的办法我们是极力赞成的。

我们只需要敲一行命令，告诉Git，以后st就表示status：

$ git config --global alias.st status

当然还有别的命令可以简写，很多人都用co表示checkout，ci表示commit，br表示branch：

$ git config --global alias.co checkout$ git config --global alias.ci commit$ git config --global alias.br branch