# xu19921225 xy6999976

# Git简介

http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000

### git 看看系统有没有安装git

### $ git config --global user.name "Your Name" 配置全局用户名

### $ git config --global user.email ["email@example.com"](mailto:\"email@example.com\") 配置邮箱

### $ mkdir learngit 创建learngit空白目录

### $ cd learngit 到 learngit 目录下

### $ pwd 显示当前目录

### git init 把当前目录变成git管理的仓库

### git add readme.txt 把readme.txt文件添加到仓库

### git commit -m ”wrote a readme file” 把文件提交到仓库

### git status 查看当前仓库状态

### git diff readme.txt 查看readme.txt具体细节

### Git log 查看详细提交日志

### Git log --pretty=oneline 简介的显示提交的日志

### Git reset --hard HEAD^ 返回上一个版本

### Git reset --hard 3455433 返回到3455433这个版本

### Cat readme.txt 查看readme.txt文件的信息

### Git reflog 查看以前使用过的命令

### Git diff HEAD -- readme.txt 查看工作区和版本库里最新版本的区别

### git checkout -- readme.txt 把readme.txt文件在工作区的修改全部撤销

### rm test.txt 删除test.txt 文件（工作区的文件）

### git rm test.txt 删除test.txt文件（版本库的文件） 删除之后要提交git commit

### git checkout -- test.txt恢复删除的文件（恢复工作区的文件）

### ssh-keygen -t rsa -C ["youremail@example.com"](mailto:\"youremail@example.com\") 创建SSH Key

看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文 件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。

1. 第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容

1. 登陆GitHub，然后，在右上角找到“Create a new repo”按钮，创建一个新的仓库：
2. 在Repository name填入learngit，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：
3. git remote add origin [git@github.com:michaelliao/learngit.git](mailto:git@github.com:michaelliao/learngit.git) 关联仓库

添加后，远程库的名字就是origin，这是Git默认的叫法，也可以改成别的，但是origin 这个名字一看就知道是远程库。

1. git push -u origin master 把master分支推送到远程上去

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地 的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的 master分支关联起来，在以后的推送或者拉取时就可以简化命令。

1. git push origin master master分支提交推送远程
2. git clone [git@github.com:michaelliao/gitskills.git](mailto:git@github.com:michaelliao/gitskills.git) 克隆一个本地仓库
3. git checkout -b dev 创建dev分支，然后切换到dev分支
4. $ git branch dev 创建dev分支
5. $ git checkout dev 切换到dev分支
6. git branch命令查看当前分支：
7. git merge dev 合并dev分支
8. git branch -d dev 删除分支dev
9. Git发生了冲突

Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容，我们要手动修改并提交才行

1. git log --graph --pretty=oneline --abbrev-commit 查看分支合并图的情况
2. git merge --no-ff -m "merge with no-ff" dev 表示禁用Fast forward合并

因为本次合并要创建一个新的commit，所以加上-m参数，把commit描述写进去。

1. git stash 当前工作现场“储藏”起来，等以后恢复现场后继续工作
2. git stash list 命令看看工作现场
3. git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除
4. git stash pop，恢复的同时把stash内容也删了
5. 可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令：
6. $ git stash apply stash@{0}
7. git branch -d feature-vulcan 删除分支
8. git branch -D feature-vulcan 强行删除
9. git remote 查看远程仓库的信息
10. git remote -v 查看更加详细的远程仓库信息
11. git push origin master 推送本地分支到远程仓库上
12. git checkout -b dev origin/dev 创建远程origin的dev分支到本地

现在，他就可以在dev上继续修改，然后，时不时地把dev分支push到远程： git push origin dev

1. 推送失败

因为你的小伙伴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，Git已经 提示我们，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： git pull

1. git pull失败了

原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和 origin/dev的链接：git branch --set-upstream dev origin/dev在 git pul

1. 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完全一样。解决 后，提交，再push git commit -m "merge & fix hello.py" git push origin dev

1. 多人协作的工作模式通常是这样：

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！

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 branch-name origin/branch-name。

1. git tag v1.0 打上v1.0的标签
2. git tag查看所有标签
3. git tag v0.9 6224937 不知6224937打上v0.9的标签
4. git show <tagname>查看标签信息
5. git tag -a v0.1 -m "version 0.1 released" 3628164
6. git tag -s v0.2 -m "signed version 0.2 released" fec145a 通过-s用私钥签名一个标签
7. git tag -d v0.1 标签打错了，也可以删除

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。

1. git push origin v1.0 推送标签到远程
2. git push origin --tags 一次性推送全部尚未推送到远程的本地标签
3. 如果标签已经推送到远程，要删除远程标签就麻烦一点，先从本地删除

git tag -d v0.9 然后，从远程删除。删除命令也是push git push origin :refs/tags/v0.9

1. git config --global color.ui true Git会适当地显示不同的颜色
2. git config --global alias.st status 告诉Git，以后st就表示status
3. git config --global alias.co checkout
4. git config --global alias.ci commit
5. git config --global alias.br branch
6. cat .git/config 查看配置文件

别名就在[alias]后面，要删除别名，直接把对应的行删掉即可。配置别名也可以直接修改这个文件，如果改错了，可以删掉文件重新通过命令配置。