# Centos 安装、卸载 git

yum install git

yum remove git

# 安装完成最后一步设置：

$ git config --global user.name "Your Name"

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



# 创建版本库

首先，选择一个合适的地方，创建一个空目录：

$ mkdir learngit

$ cd learngit

$ pwd

/Users/michael/learngit

第二步，通过git init命令把这个目录变成Git可以管理的仓库：

$ git init

Initialized empty Git repository **in** /Users/michael/learngit/.git/

# 把文件添加到版本库

编写一个readme.txt文件，内容如下：

Git is a version control system.

Git is free software.

第一步，用命令git add告诉Git，把文件添加到仓库：

$ git add readme.txt

第二步，用命令git commit告诉Git，把文件提交到仓库（-m后面输入的是本次提交的说明）：

$ git commit -m "wrote a readme file"

[master (root-commit) cb926e7] wrote a readme file

1 file changed, 2 insertions(+)

create mode 100644 readme.txt

为什么Git添加文件需要add，commit一共两步呢？因为commit可以一次提交很多文件，所以你可以多次add不同的文件，比如：

$ git add file1.txt

$ git add file2.txt file3.txt

$ git commit -m "add 3 files."

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

在Git中查看历史记录，显示从最近到最远的提交日志：

$ git log

$ git log --pretty=oneline

一大串类似3628164...882e1e0的是commit id（版本号）

# 把readme.txt回退到上一个版本

$ git reset --hard HEAD^

用HEAD表示当前版本，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。

或：

$ git reset --hard 3628164（3628164指版本号）

版本号没必要写全，前几位就可以了，Git会自动去找。当然也不能只写前一两位，因为Git可能会找到多个版本号，就无法确定是哪一个了。

Git提供了一个命令git reflog用来记录你的每一次命令：

# 工作区和暂存区

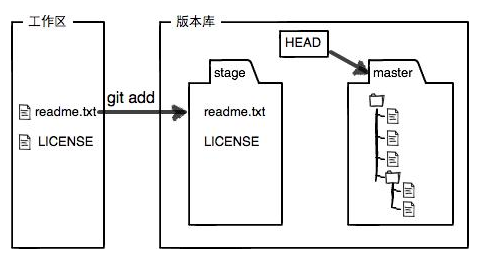
工作区（Working Directory）

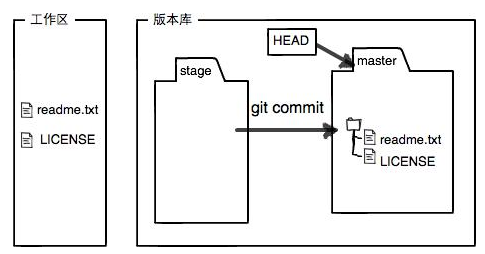
就是你在电脑里能看到的目录，比如我的learngit文件夹就是一个工作区

版本库（Repository）

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。





git diff HEAD -- readme.txt命令可以查看工作区和版本库里面最新版本的区别：

# 撤销修改

丢弃工作区的修改：

$ git checkout -- readme.txt

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”

把暂存区的修改撤销掉（unstage），重新放回工作区：

$ git reset HEAD readme.txt

# 删除文件

$ git rm test.txt

rm 'test.txt'

$ git commit -m "remove test.txt"

[master d17efd8] remove test.txt

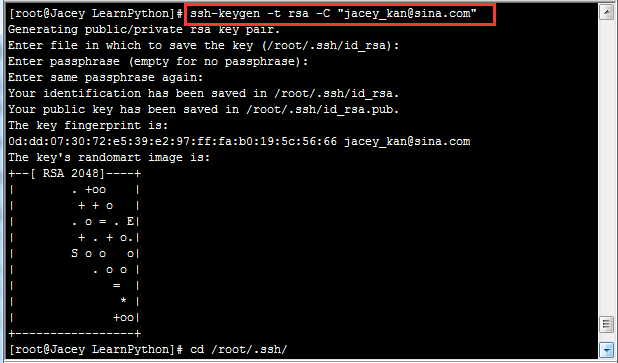
1 file changed, 1 deletion(-)

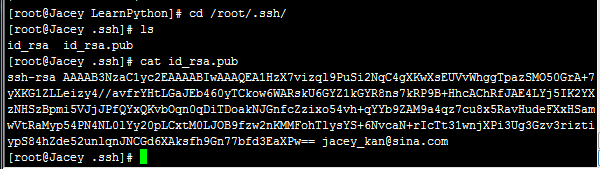
delete mode 100644 test.txt

# 远程仓库

## 第1步：创建SSH Key。

$ ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com)

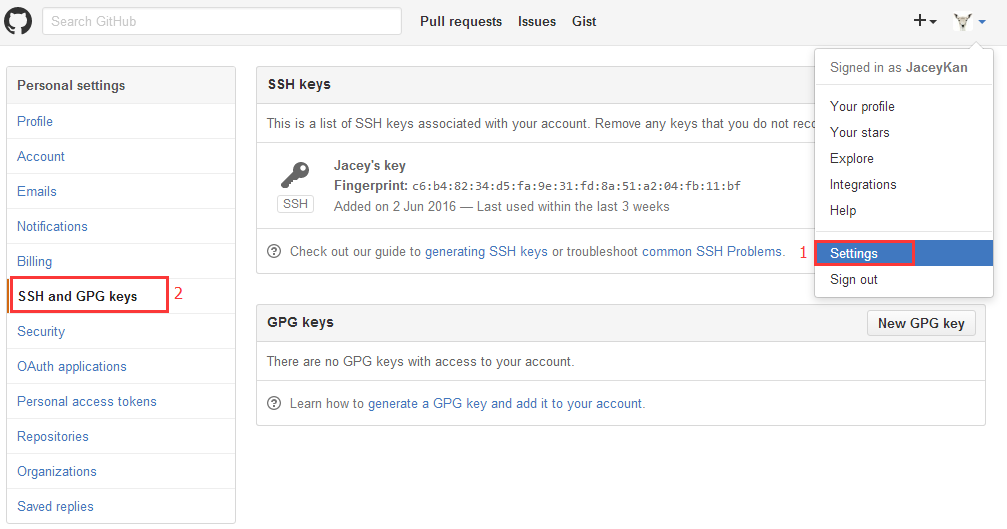


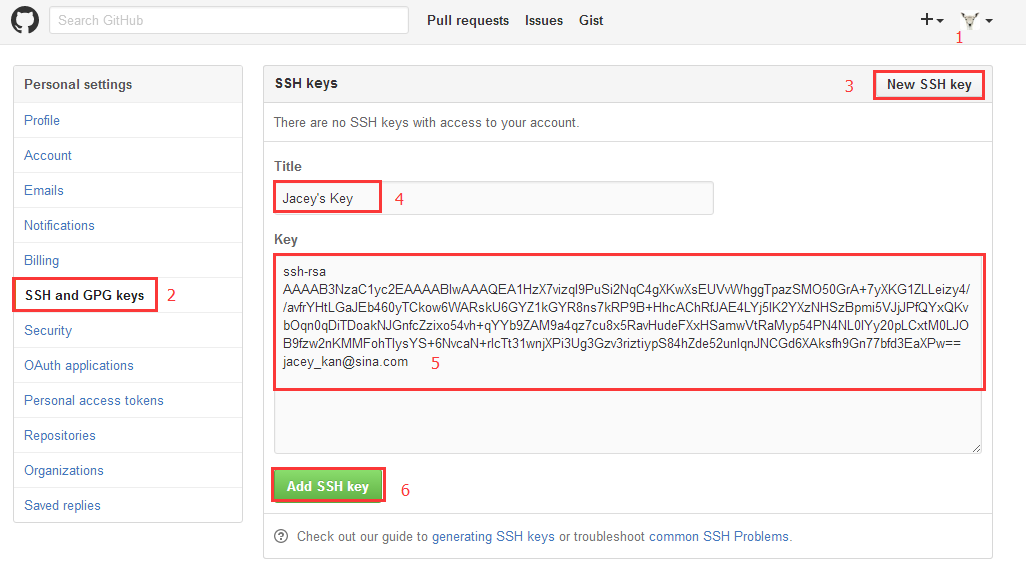


## 第2步：登陆GitHub

打开“Account settings”，“SSH Keys”页面：

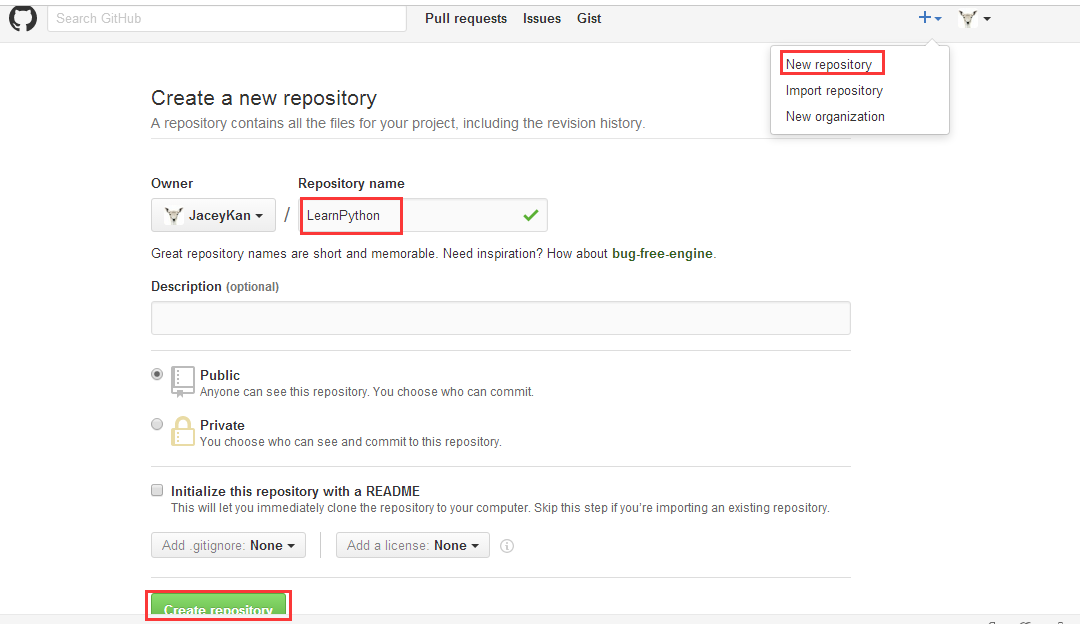
然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容





## 添加远程库--先有本地库，后有远程库的时候，如何关联远程库

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：



在本地的learngit仓库下运行命令：

$ git remote add origin git@github.com:*JaceyKan/LearnPython.git*

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

下一步，就可以把本地库的所有内容推送到远程库上：

$ git push -u origin master

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。

从现在起，只要本地作了提交，就可以通过命令：

$ git push origin master

## 从远程库克隆---先创建远程库，然后，从远程库克隆

用命令git clone克隆一个本地库：

$ git clone [git@github.com:*JaceyKan/gitskills.git*](mailto:git@github.com:JaceyKan/gitskills.git)

$ git clone <https://github.com/plusjade/jekyll-bootstrap.git> test

# 问题1：git push -u origin master出错

ssh: Could not resolve hostname github.com: Name or service not known

fatal: The remote end hung up unexpectedly



原因：网络不通，无法ping通github



设置好网络就可以了

# 创建与合并分支

首先，我们创建dev分支，然后切换到dev分支：

$ git checkout -b dev

Switched to a new branch 'dev'

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

$ git branch dev

$ git checkout dev

Switched to branch 'dev'

然后，用git branch命令查看当前分支：

$ git branch

\* dev

master

切换回master分支：

$ git checkout master

Switched to branch 'master'

合并指定分支到当前分支，把dev分支的工作成果合并到master分支上：

$ git merge dev

Updating d17efd8..fec145a

Fast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

# 删除dev分支：

$ git branch -d dev

Deleted branch dev (was fec145a).

# 小结：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

用带参数的git log也可以看到分支的合并情况：

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

\* 59bc1cb conflict fixed

|\

| \* 75a857c AND simple

\* | 400b400 & simple

|/

\* fec145a branch test

...

准备合并dev分支，请注意--no-ff参数，表示禁用Fast forward：

$ git merge --no-ff -m "merge with no-ff" dev

Merge made by the 'recursive' strategy.

readme.txt | 1 +

1 file changed, 1 insertion(+)

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。

把当前工作现场“储藏”起来，等以后恢复现场后继续工作：

$ git stash

Saved working directory and index state WIP on dev: 6224937 add merge

HEAD is now at 6224937 add merge

查看被储藏起来的工作现场：

$ git stash list

stash@{0}: WIP on dev: 6224937 add merge

工作现场还在，Git把stash内容存在某个地方了，但是需要恢复一下，有两个办法：

一是用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；

另一种方式是用git stash pop，恢复的同时把stash内容也删了：

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：

$ git stash apply stash@{0}

强行删除分支：

$ git branch -D feature-vulcan

# 创建GitHub展示页

创建gh-pages分支

$ git branch gh-pages //创建分支

$ git checkout gh-pages //切换到gh-pages分支

$ git merge master //合并master分支到gh-pages

$ git add \*

$ git commit -m "update"

$ git push -u origin gh-pages //将当前分支推送到远程gh-pages分支

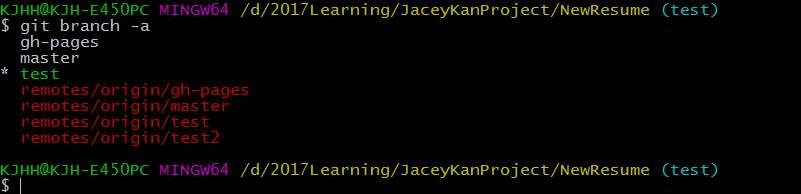
查看展示页：

<https://jaceykan.github.io/NewResume/>

默认展示index.html页面

# 远程分支的相关操作

$ git branch –a //查看所有的分支（包括远程分支）



星号(\*)表示当前所在分支。

$ git push origin test:test2 //将本地的test分支推送到test2分支，如果远程test2分支不存在，创建test2

删除远程分支

1.我比较喜欢的简单方式，推送一个空分支到远程分支，其实就相当于删除远程分支：

$ git push origin :test2 //删除远程的test2分支

2.也可以使用：

$ git push origin --delete test2

这两种方式都可以删除指定的远程分支

# 多人协作

查看远程库的信息：

$ git remote

origin

或者显示更详细的信息：

$ git remote -v

origin git@github.com:michaelliao/learngit.git (fetch)

origin git@github.com:michaelliao/learngit.git (push)

上面显示了可以抓取和推送的origin的地址。如果没有推送权限，就看不到push的地址。

推送分支，就是把该分支上的所有本地提交推送到远程库。

推送时，要指定本地分支：

$ git push origin master

多人协作的工作模式

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

2.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；

3.如果合并有冲突，则解决冲突，并在本地提交；

4.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！

5.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