

1.安装Git

Git可安装在所有操作系统上运行，但其安装方法因操作系统而异。

在Linux系统中安装Git

终端执行如下命令：

ubuntu: `sudo apt-get install git`

centos: `yum install git`

在Mac 系统中安装Git

你的mac系统可能已经安装了Git，因此请尝试执行命令 `git --version`

如果你在输出上到了具体的版本号，说明你的系统已经安装了Git；如果你看到一条信息，提示你安装或者升级Git，只需安装屏幕上的说明做即可。

你也可以访问<https://git-scm.com>，点击Downloads，再单击合适你所用系统的安装程序

在Windows系统中安装Git

要在windows系统中安装Git 请访问<http://msysgit.github.io/>，并点击Download

配置Git

Git跟踪谁修改了项目，哪怕参与项目开发的人只有一个。为此，Git需要知道你的用户名和电子邮箱地址。你必须提供用户名，但可以使用虚拟邮箱地址：

```
git config --global user.name "username"
```

```
git config --global user.email "user@xx.com"
```

Git的基本使用

创造项目

我们来创建一个 要进行版本控制的项目。在你的系统中创建一个文件夹，并将其命名为 git_practice。在这个文件夹中，创建一个简单的Python程序：

```
hello_world.py  
print('Hello Git world')
```

我们将使用这个程序来探索Git的基本功能。

忽略文件

扩展名为.pyc的文件是根据.py文件自动生成的，因此我们无需让Git跟踪他们。这些文件存储在目录__pycache__中。为了让Git忽略这个目录，创建一个名为 `.gittignore` 的特殊文件(这个文件名以句点打开，且没有扩展名)，并且在其中添加下面一行内容：

```
__pycache__/
```

这让Git忽略目录__pycache__中的所有文件。使用文件 `.gitignore` 可避免项目混乱，开发起来更容易。

初始化仓库

你创建了一个目录，其中包含了一个Python文件和一个 `.gitignore` 文件，可以初始化一个Git仓库了，为此，在终端窗口，切换文件夹git_practice，并执行如下命令：

```
git init
```

检查状态

```
git status
```

将文件加入到仓库中

```
git add .  
git status
```

执行提交

```
git commit -m "Starte project"
```

查看提历史

```
git log
```

第二次提交

```
git commit -am 'ceshi'
```

撤销修改

```
git checkout .
```

检查出以前的提交

你可以检出提交历史中的任何提交，而不仅仅是最后一次提交，为此可在命令 `git checkout` 末尾指定该提交的引用ID的前6个字符

```
git log --pretty=oneline
```

```
git checkout be017b
```

删除仓库

```
rm -rf .git
```

使用Git上传项目到github



提起GitHub大家并不陌生，GitHub也算为“开源”做出突出的贡献了~ Git这种良好支持分

支管理的分布式的SCM真正解决了一个问题问题：每个工程师在自己本地分支上开发，完成功能以后往master分支合并。

关于Windows下GitHub的一些使用，随处可见。本文着重介绍如何在Mac OSX下上传本地的项目源代码至GitHub。

安装Git工具。

- 1、下载Git installer，地址：<http://git-scm.com/downloads>
- 2、pkg包下载完成，双击安装。
- 3、打开终端，使用git --version命令查看安装版本，能查看到就是安装成功了(eg:git version 2.5.4 (Apple Git-61))。

创建SSH

打开终端，输入以下命令来查看 `.ssh` 是否存在：

```
$ cd ~/.ssh
```

若提示 `-bash: cd: ~/.ssh: No such file or directory`

那就说明.ssh文件夹不存在，那么则进行下一个步骤，否则就是将原来的.ssh文件夹备份以下，亦或是切换到其他路径下建立ssh。输入以下命令来创建ssh：

```
$ ssh-keygen -t rsa -C xxx@xx.com
```

`xxx@xx.com` 为你注册GitHub时的邮箱账号，命令执行成功后，会有以下提示

```
1  +---[ RSA 2048]-----+
2  |           .           |
3  |           +           |
4  |       E . +   +   |
5  |           o * o + +   |
6  |           S + = =   |
7  |           . o + 0    |
8  |           . * .      |
9  |           .          |
10 |                       |
11 +-----+

```

GitHub上创建SSH

首先登陆你的GitHub， 右上角用户头像下选择Settings， 在SSH Keys 选项里面添加ssh， 如下所示。

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Blocked users

Repositories

Organizations

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

- Title： 名字随意
- Key： 打开你生成的id_rsa.pub文件， 将其中内容拷贝过来。

添加成功如图所示：

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security


Blocked users

Repositories

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



dell

Fingerprint: 82:a6:f9:c6:95:1c:67:fa:51:54:ec:3e:c6:c0:b7:ae

Added on 12 Jul 2018

Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

值得一提的是， 在Mac 下， 可以通过以下命令 `open .` 来打开当前文件夹， 当然， 熟悉Linux命令行操作的人， 可以直接用vim命令打开， 或者cat直接显示。

```
niaQ/E0bcTS5Q5J4TLGDKprLX'  
ab8AvfrqBEA95j7sUES51bl8t  
E0cHTu1v13DMwz+cZDtLx1Q6z
```

```
centing$ ls
```

```
centing$ ls -ah
```

```
centing$ ls -a
```

```
centing$ ls
```

```
centing$ cat id_rsa.pub
```

```
QABAAABAQDGketFLkTW/7n02P
```

```
MvrEAN1C8MtvGcGI5/NCYE/tu
```

```
QzmlihN96clRTTTk3G/y90Kd/
```

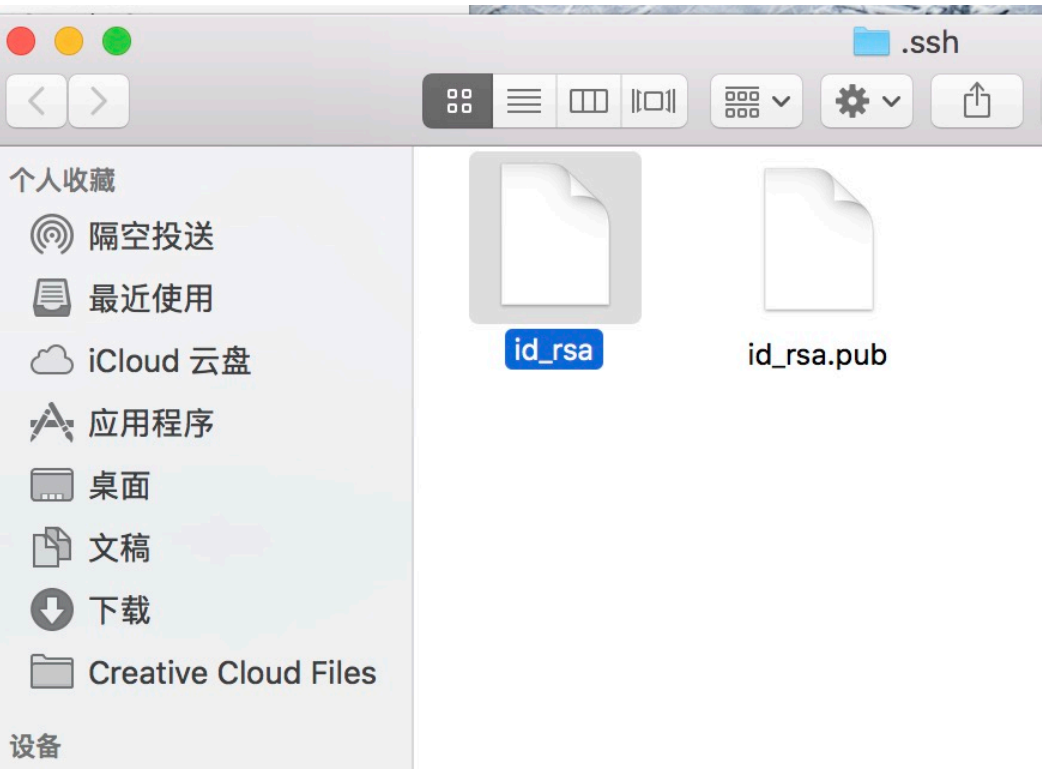
```
hc0ElXN/bkkI1bz46uFu4jIwj
```

```
146Bxtv092S+le7+0c0gBfl b
```

```
centing$ open .
```

```
centing$
```

```
_rsa.pub文件，将
```



GitHub上创建版本库

在GitHub首页上，点击“Create a New Repository”，如下所示（为了便于后面演示，创建README.md这步暂不勾选）：

创建完成后跳转到代码仓库界面，如下：

Owner

Repository name

 binyoucai ▾ / c++ ✓

Great repository names Your new repository will be created as c- on? How about **probable-journey**.

Description (optional)

thi is a cpp



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



Creating repository...

 binyoucai / c-

 Watch ▾

0

 Star

0

 Fork

0

 Code

 Issues 0

 Pull requests 0



 Projects 0

 Wiki

 Insights

 Settings

Quick setup — if you've done this kind of thing before

 Set up in Desktop or ☐ HTTPS ☒ SSH `git@github.com:binyoucai/c-.git` 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


...or create a new repository on the command line

```
echo "# c-" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:binyoucai/c-.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin git@github.com:binyoucai/c-.git
git push -u origin master
```



当然了，在没有上传代码之前，列表是空的。

紧接着按照以下步骤进行本地仓库的创建及代码上传。打开终端，输入以下命令：

```

1 $ touch README.md //新建一个R
2 EADME文档，若上一步勾选了创建README.md，提交时导致冲突
3 $ git init //初始化本
4 地仓库
5 $ git add README.md //添加刚刚
6 创建的README文档
$ git commit -m "你的注释...." //提交到本地
仓库，并写一些注释
$ git remote add origin git@github.com:yourname/xxxx.git //连接远程
仓库并建了一个名叫：origin的别名，当然可以为其他名字，但是origin一看就知道是别
名，yourname记得替换成你的用户名
$ git push -u origin master //将本地仓库
的文件提交到别名为origin的地址的master分支下，-u为第一次提交，需要创建master
分支，下次就不需要了

```

初始化完成之后，我们可以把我们项目的源代码提交上去，使用git add命令，如下：

```

1 $ git add 系统签名/ // 添加需要提
2 交的文件夹，使用git add . 则添加全部
3 $ git add assets/
4 $ git add project.properties
5 $ git add res/
6 $ git add src/
7 $ git commit -m "上传项目源代码" // 提交到本地仓
库
$ git push origin master // 将本地仓库合
并到别名为origin地址的master分支下

```

显示结果如下，则代码上传成功。

```

1 Counting objects: 63, done.
2 Delta compression using up to 8 threads.
3 Compressing objects: 100% (53/53), done.
4 Writing objects: 100% (63/63), 1.41 MiB | 217.00 KiB/s, done.
5 Total 63 (delta 16), reused 0 (delta 0)
6 To git@github.com:smuyyh/autoinstall.git
7 000a667..61357d8 master -> master

```

刷新一下GitHub，则显示刚刚提交的项目源代码。

GitHub项目下载到本地

`git clone git_url` 命令将存储库克隆到新目录中

相关总结

1、要关联一个远程库，使用命令`git remote add origin git@server-name:path/repo-name.git`；关联后，使用命令`git push -u origin master`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；

2、切记上传文件时，一定要先commit到本地仓库，才能进行push提交，否则会显示Everything up-to-date（意思就是目前的远程仓库的内容跟本地仓库对比后，没有做修改，是最新的）；

3、初用Mac的童鞋可能还不知道终端在哪里，点击Finder旁边的Launchpad，在“其他”文件夹下。

4、在设置别名的时候，出现“fatal: remote origin already exists.”错误，说明该别名已经存在，可以另外建一个别名，或者使用“`git remote rm origin`”命令删除原来的别名，然后重新执行“`git remote add origin git@github.com:yourname/xxxx.git`”；

5、在提交的时候，出现“error: failed to push some refs to '[git@github.com](https://github.com):xxx/xxx.git'
hint: Updates were rejected because the remote contains work that you do not have locally...”的错误，说明有冲突，远程仓库的版本比本地仓库的要新，所以要先进行更新，才能提交。使用“`git pull git@github.com:xxx/xxx.git`”命令进行更新，地址自己相应替换掉。