**本地环境配置github**

## 将你的代码托管到github

github是程序猿的代码托管平台，也是基于git的开源分布式版本控制系统。然而，当你登陆github官网时，它并没有为你准备一个很好的代码上传的系统，这是因为它是基于git的分布式版本管理系。那么，如何更快更有效的将本地代码上传到github呢？首先，我们需要在本地安装git，这样才能在本地环境中使用git命令行，（例如：$ git add index.html）其次是要连接到你的github账户上，这样才能把你的代码文件上传上去，而每一次的更改都会形成一个版本记录，这样对团队协作是很有帮助的。

## 安装 Git

这里主要讲的是mac系统，windows也是一样的，只不过mac是在终端下进行，而windows是在cmd下进行的。安装git，这里推荐安装Apple公司的Xcode，Xcode集成了Git，最新版的Xcode已经默认安装好了git。完成安装之后，就可以使用 git 的命令行工具。

当然，首先你需要注册一个github账户。

### 配置帐号信息

在cmd或者终端下输入以下命令行：

git config --global user.name trigkit4

git config --global user.email [345823102@qq.com](mailto:345823102@qq.com)

当然，这是我的账户信息，你需要将他们换成你自己的。

## 创建本地ssh

这是一种传输代码的方法，速度快又安全。SSH 是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议。

在终端或cmd输入以下命令行：

ssh-keygen -t rsa -C [345823102@qq.com](mailto:345823102@qq.com)

**路径选择 :** 使用该命令之后, 会出现提示选择ssh-key生成路径, 这里直接点回车默认即可, 生成的ssh-key在默认路径中;  
**密码确认 :** 这里我们不使用密码进行登录, 用密码太麻烦;就一路回车下去

### 将ssh配置到GitHub中

在mac os X 下前往文件夹，/Users/自己电脑用户名/.ssh。

windows应该是（C:\Documents and Settings\Administrator\.ssh （或者 C:\Users\自己电脑用户名\.ssh）中）。

然后用记事本打开id\_rsa.pub，将里面的全部代码复制到github的SSH中。

id\_rsa.pub 文件内容 :

登陆github网站，点击Settings——SSH keys——点击右侧的Add SSH key ，接下去你懂得。

**验证是否配置成功 :**  
复制如下代码:

ssh -T [git@github.com](mailto:git@github.com)

验证时可能让你输入YES，当出现以上信息时，说明配置成功，可以连接上GitHub;

## 创建版本库

第一步，在本地创建一个版本库

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

这里的.git目录是Git用来跟踪管理版本库的，默认是隐藏的。

第三部，接着，在github上创建一个你自己的new repository，然后下一步，

然后会输出以下信息：

Initialized empty Git repository in /Users/trigkit4/banner/.git/

这里的.git目录是Git用来跟踪管理版本库的，默认是隐藏的。

第三部，接着，在github上创建一个你自己的new repository，然后下一步，

mkdir test

cd test

git init

# initialize your git repository

touch README

# create a file named README

git add README

# add README to cache

git commit -m 'first commit'

# commit your files to local repository

然后我们将本地的文件传送至github中，使用如下命令

git remote add origin https://github.com/yourname/test.git

git push -u origin master

## 从现有仓库克隆

git clone git://github.com/yourname/test.git

git clone git://github.com/yourname/grit.git mypro#克隆到自定义文件夹

### 本地

git add \*#跟踪新文件

rm \*&git rm \*#移除文件

git rm -f \*#移除文件

git rm --cached \*#取消跟踪

git mv file\_from file\_to#重命名跟踪文件

git log#查看提交记录

git commit#提交更新

git commit -m 'message'

git commit -a#跳过使用暂存区域，把所有已经跟踪过的文件暂存起来一并提交

git commit --amend#修改最后一次提交

git reset HEAD \*#取消已经暂存的文件

git checkout -- file#取消对文件的修改（从暂存区去除file）

git checkout branch|tag|commit -- file\_name#从仓库取出file覆盖当前分支

git checkout -- .#从暂存区去除文件覆盖工作区

git branch#列出本地分支

git branch -r#列出远端分支

git branch -a#列出所有分支

git branch -v#查看各个分支最后一个提交对象的信息

git branch --merge#查看已经合并到当前分支的分支

git branch --no-merge#查看为合并到当前分支的分支

git branch test#新建test分支

git checkout test#切换到test分支

git checkout -b test#新建+切换到test分支

git checkout -b test dev#基于dev新建test分支，并切换

git branch -d test#删除test分支

git branch -D test#强制删除test分支

git merge test#将test分支合并到当前分支

git rebase master#将master分之上超前的提交，变基到当前分支

## Git 命令详解

现在我们有了本地和远程的版本库，让我们来试着用用Git的基本命令：

git pull：从其他的版本库（既可以是远程的也可以是本地的）将代码更新到本地，例如：'git pull origin master'就是将origin这个版本库的代码更新到本地的master主枝，该功能类似于SVN的update

git add：是 将当前更改或者新增的文件加入到Git的索引中，加入到Git的索引中就表示记入了版本历史中，这也是提交之前所需要执行的一步，例如'git add app/model/user.rb'就会增加app/model/user.rb文件到Git的索引中，该功能类似于SVN的add

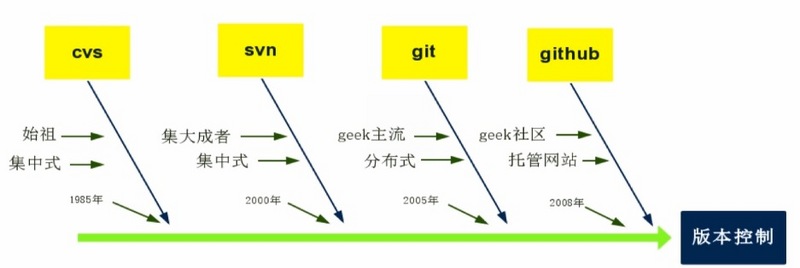
git rm：从当前的工作空间中和索引中删除文件，例如'git rm app/model/user.rb'，该功能类似于SVN的rm、del

git commit：提交当前工作空间的修改内容，类似于SVN的commit命令，例如'git commit -m story #3, add user model'，提交的时候必须用-m来输入一条提交信息，该功能类似于SVN的commit

git push：将本地commit的代码更新到远程版本库中，例如'git push origin'就会将本地的代码更新到名为orgin的远程版本库中

git log：查看历史日志，该功能类似于SVN的log

git revert：还原一个版本的修改，必须提供一个具体的Git版本号，例如'git revert bbaf6fb5060b4875b18ff9ff637ce118256d6f20'，Git的版本号都是生成的一个哈希值



## git思维导图

