Github 教程

《Typora----工具》

如果你是一枚Coder，但是你不知道Github，那么我觉的你就不是一个菜鸟级别的Coder，因为你压根不是真正Coder，你只是一个Code搬运工。

但是你如果已经在读这篇文章了，我觉的你已经知道Github了。

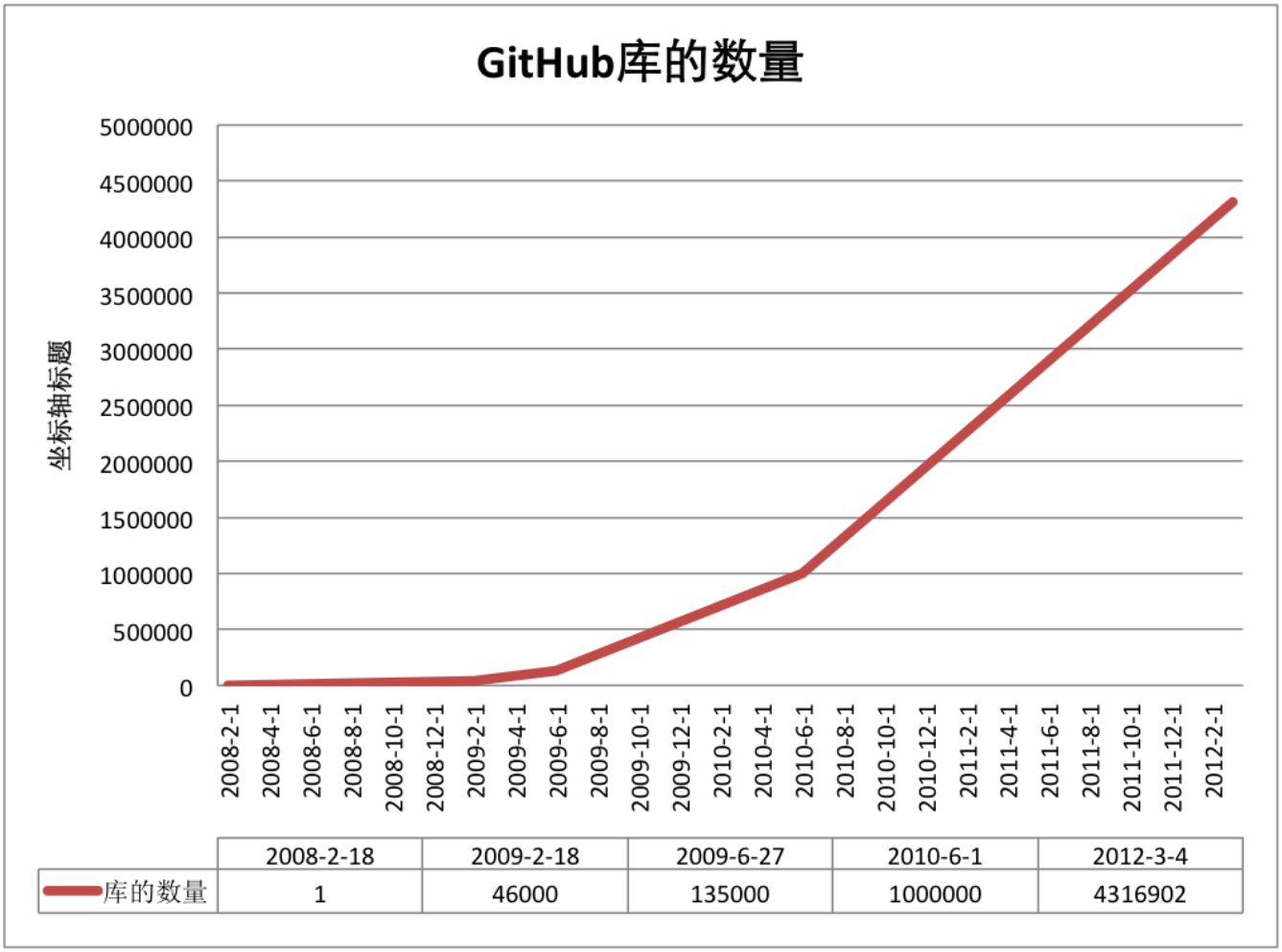
正是Github，让社会化编程成为现实。

Git可以使用四种主要的协议来传输数据： 本地传输、SSH协议、Git协议和HTTP协议 注意，除了HTTP协议外，其他所有协议都要求在服务器端安装并运行Git。 本地协议： 最基本的协议本地协议（Local protocol）,所谓的远程仓库在协议中的表示，就是硬盘上的另一个目录。团队中的每一个成员对共享文件系统（NFS）都有访问权。

什么是 Github?

github是一个基于git的代码托管平台，付费用户可以建私人仓库，我们一般的免费用户只能使用公共仓库，也就是代码要公开。 Github 由Chris Wanstrath, PJ Hyett 与Tom Preston-Werner三位开发者在2008年4月创办。迄今拥有59名全职员工，主要提供基于git的版本托管服务。

目前看来，GitHub这场冒险已经胜出。根据来自维基百科关于GitHub的描述，我们可以形象地看出GitHub的增长速度：



今天，GitHub已是：

- 一个拥有143万开发者的社区。其中不乏Linux发明者Torvalds这样的顶级黑客，以及Rails创始人DHH这样的年轻极客。
- 这个星球上最流行的开源托管服务。目前已托管431万git项目，不仅越来越多知名开源项目迁入GitHub，比如Ruby on Rails、jQuery、Ruby、Erlang/OTP；近三年流行的开源库往往在GitHub首发，例如：TensorFlow BootStrap、Node.js、CoffeScript等。

注册账户以及创建仓库

要想使用github第一步当然是注册github账号了， github官网地址：https://github.com/。 之后就可以登陆--创建仓库了（免费用户只能建公共仓库），Create a New Repository，填好名称后Create，之后会出现一些仓库的配置信息，这也是一个git的简单教程。

Github 安装

- 下载 git Windows 版
- 下载 git Linux 版(推荐)

  在linux下在线下载安装git sudo apt-get install git

配置Git

首先在本地创建`ssh key；`

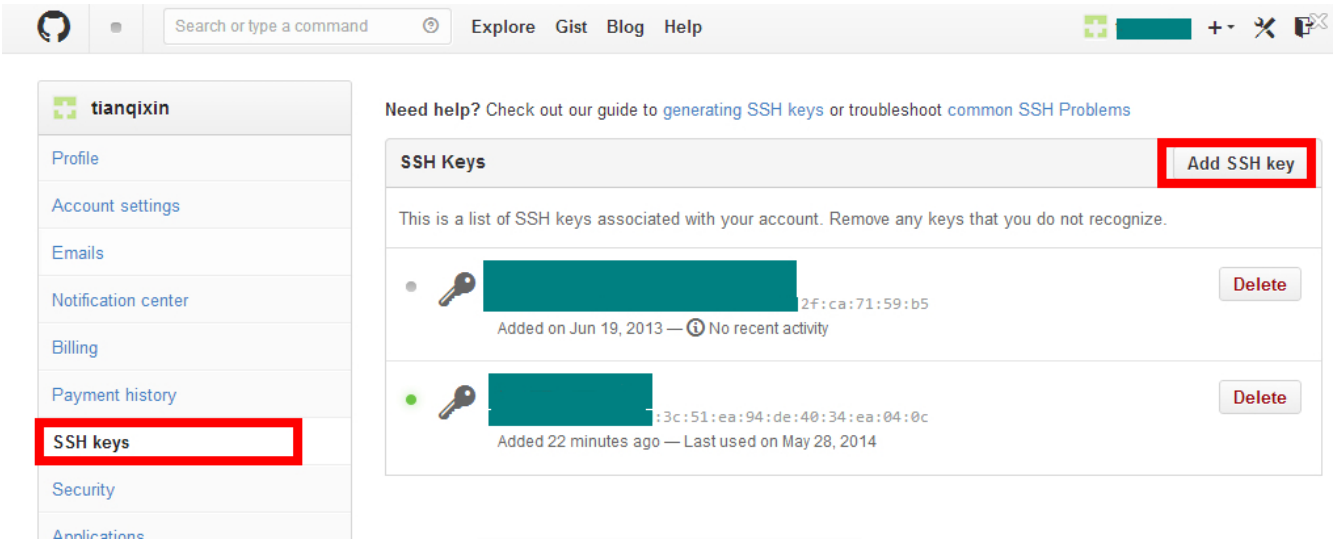
```
$ ssh-keygen -t rsa -C "your@email.com"
实例：
    linux@ubuntu:~/Github$  ssh-keygen -t rsa -C "99393534x@qq.com"
    Generating public/private rsa key pair.
    Enter file in which to save the key (/home/linux/.ssh/id_rsa):
    Created directory '/home/linux/.ssh'.
    Enter passphrase (empty for no passphrase):
    Enter same passphrase again:
    Your identification has been saved in /home/linux/.ssh/id_rsa.
    Your public key has been saved in /home/linux/.ssh/id_rsa.pub.
    The key fingerprint is:
    SHA256:vEhpn9dSwsD9aTij4pAcHtebvfNIWr4+tzJkEQyrG+o 993935340@qq.com
    The key's randomart image is:
    +---[RSA 2048]----+
    |         .o      |
    |       . oo      |
    |        + ..     |
    |       = o.o .   |
    |    o B S *.=    |
    |   o O = BoB     |
    |    * + Bo* .    |
    |   . o . *=+.    |
    |    E . ..*Oo.   |
    +----[SHA256]-----+
```

后面的`your@email.com`改为你在github上注册的邮箱，之后会要求确认路径和输入密码，我们这使用默认的一路回车就行。成功的话会在`~/`下生成`.ssh`文件夹，进去，打开`id_rsa.pub`，复制里面的`key`。

```
    查看生成文件：
    linux@ubuntu:~/Github$ cd ~/.ssh/
    linux@ubuntu:~/.ssh$ ls
    id_rsa  id_rsa.pub

    vi   id_rsa.pub
    拷贝哈
```

回到github上，进入 Account Settings（账户配置），左边选择SSH Keys，Add SSH Key,title随便填，粘贴在你电脑上生成的key。



为了验证是否成功，在git bash下输入：

```
$ ssh -T git@github.com
```

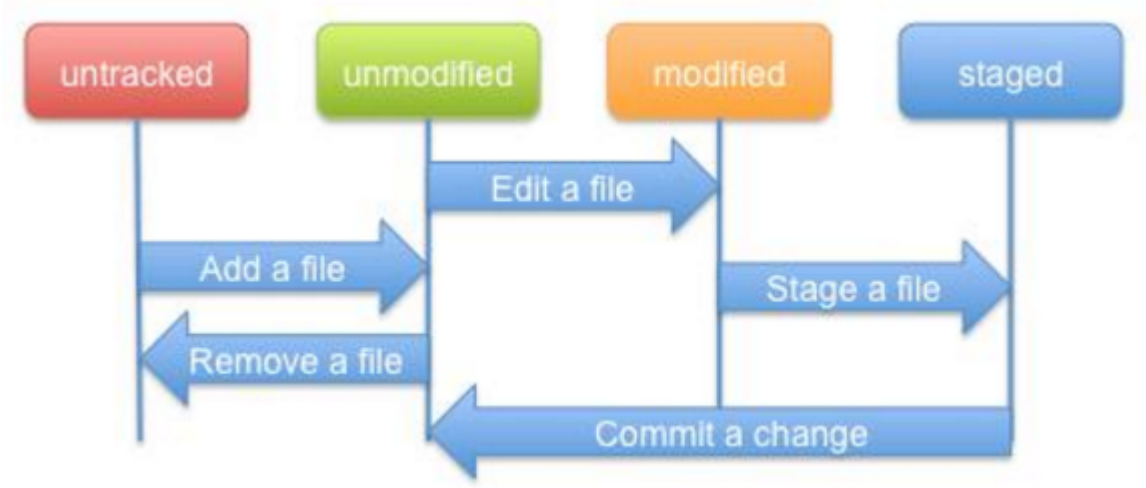
如果是第一次的会提示是否continue，输入yes就会看到：You've successfully authenticated, but GitHub does not provide shell access 。这就表示已成功连上github。

接下来我们要做的就是把本地仓库传到github上去，在此之前还需要设置username和email，因为github每次commit都会记录他们。

```
$ git config --global user.name "your name"
$ git config --global user.email "your_email@youremail.com"
```

Git 内部只有四种状态:

对于任何一个文件,在 Git 内部只有四种状态: 未跟踪untracked ，已提交 (committed),已修改(modified)和已暂存(staged) （1）已修改(modified)：表示修改了某个文件,但还没有提交给暂存区。 （2）已暂存(staged)：表示把已修改的文件放在下次提交时要保存的清单中。 （3）已提交 (committed)：表示该文件已经被安全地保存在仓库中了。 （4）未跟踪（untracked）：对于没有加入Git控制的文件。



初始化本地仓库

创建新文件夹，打开，然后执行`git init`以创建新的 git 仓库。

```
git init
```

克隆远程仓库

执行如下命令以创建一个本地仓库的克隆版本：

```
git clone  git@github.com:fengjunhuii/staff_manage.git
```

查看仓库状态 ---查看整体修改信息状态

执行如下命令以创建一个本地仓库的克隆版本：

```
git status

Changes not staged for commit:
    git发现已经有修改但还未git add 的内容
如果提示“Changes to be committed”:
    说明git发现已经git add但还未git commit的内容
如果提示“Untracked files”:
    说明你增加了新文件或者在某个子目录下增加了新文件
```

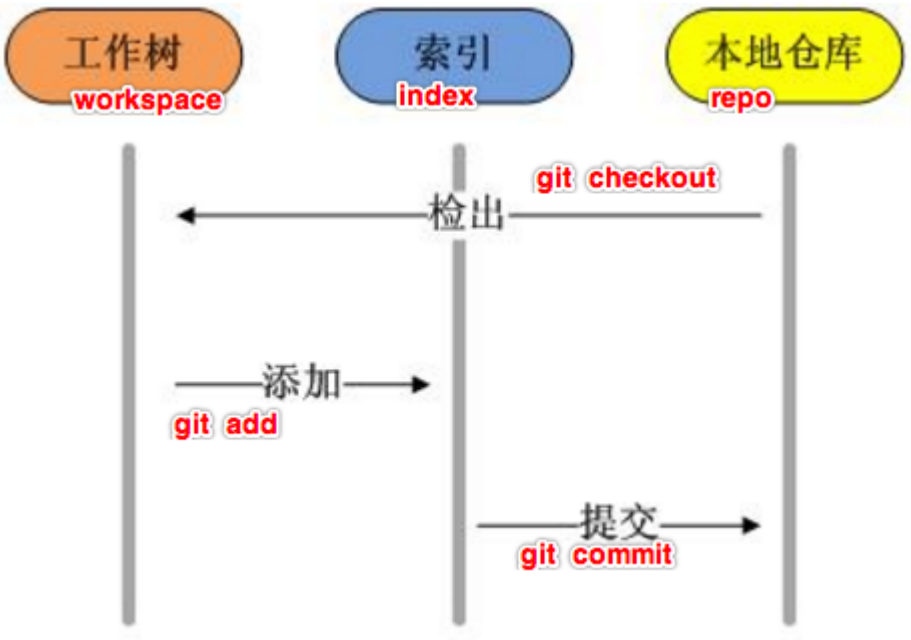
工作流与本地文件提交

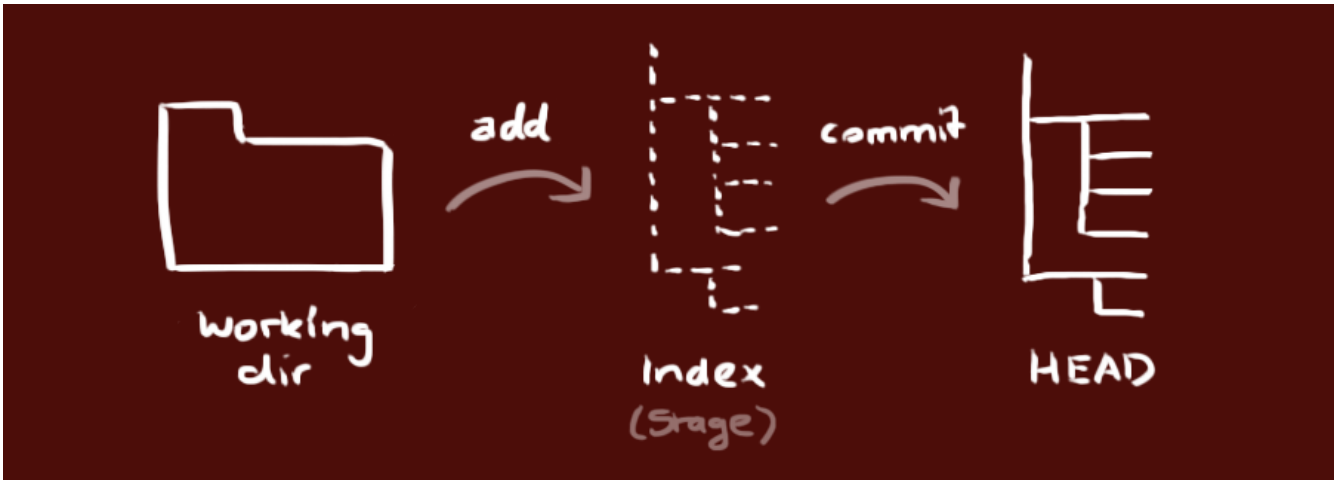
你的本地仓库由 git 维护的三棵"树"组成。第一个是你的`工作目录`，它持有实际文件；第二个是`暂存区（Index）`，它像个缓存区域，临时保存你的改动；最后是`HEAD`，它指向你最后一次提交的结果。

你可以提出更改（把它们添加到暂存区），使用如下命令：

```
git add <filename>           //添加某个指定的文件到index
git add  .                  //添加当前目录下的所有文件到index
```

`这是 git 基本工作流程的第一步；使用如下命令以实际提交改动：` git commit -m "代码提交信息"` 现在，你的改动已经提交到了 **HEAD**，但是还没到你的远端仓库。





推送改动到远程仓库

推上去----你的改动现在已经在本地仓库的 **HEAD** 中了。执行如下命令以将这些改动提交到远端仓库：

```
$ git push origin master
```

可以把 *master* 换成你想要推送的任何分支。

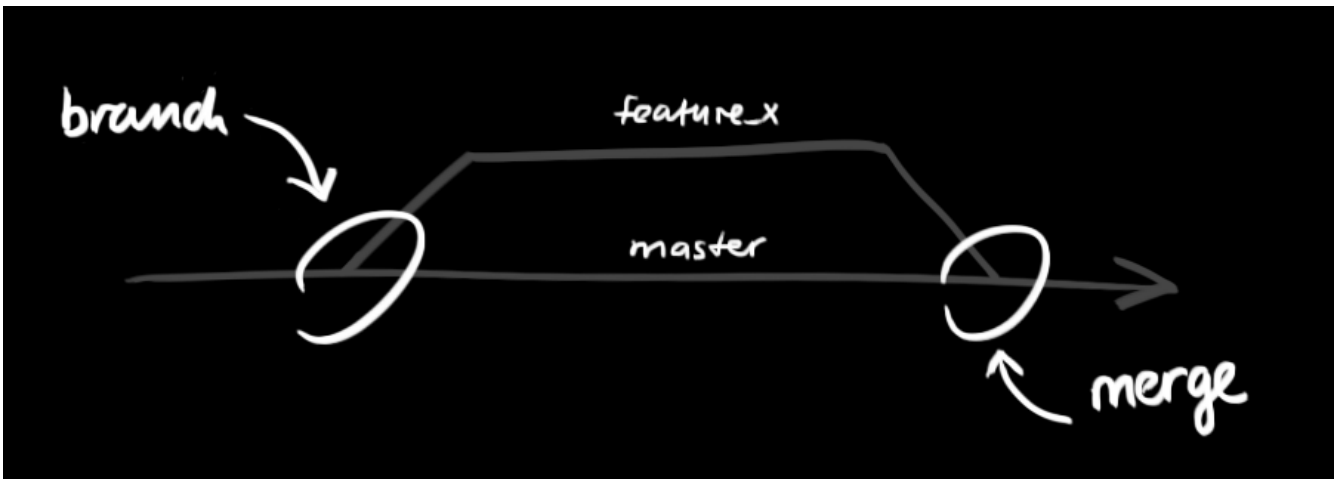
注：**如果你还没有克隆现有仓库**，并欲将你的仓库连接到某个远程服务器，你可以使用如下命令添加：

```
$ git remote add origin <server>
```

如此你就能够将你的改动推送到所添加的服务器上去了。

分支与分支切换

分支是用来将特性开发绝缘开来的。在你创建仓库的时候，*master* 是"默认的"分支。在其他分支上进行开发，完成后再将它们合并到主分支上。



创建一个叫做"feature_x"的分支，并切换过去： 创建分支：

```
    git branch  feature_x
```

查询分支：

```
    git branch
```

切换分支：

```
$ git checkout  feature_x
```

查看分支状态

```
git status
```

添加到暂存区：

```
git add <filename>
git add .
```

提交改动：

```
git commit -m "代码提交信息"`
```

推送到远程分支

```
$ git push origin feature_x
```

备用命令

切换回主分支：

```
$ git checkout master
linux@ubuntu:~/github/staff_manage$ git branch
  develop
* master
```

删除新建分支：

```
$ git branch -d feature_x
```

除非你将分支推送到远端仓库，不然该分支就是 *不为他人所见的*：

注:不同分支内的内容是不一样的

更新与合并

拉回来-----要更新你的本地仓库至最新改动，执行(操作的时候注意备份哈)：

```
$ git pull
```

以在你的工作目录中 *获取（fetch）* 并 *合并（merge）* 远端的改动。 要合并其他分支到你的当前分支（例如 master），执行：

```
$ git merge <branch>
```

在这两种情况下，git 都会尝试去自动合并改动。遗憾的是，这可能并非每次都成功，并可能出现*冲突（conflicts）*。这时候就需要你修改这些文件来手动合并这些*冲突（conflicts）*。改完之后，你需要执行如下命令以将它们标记为合并成功：

```
$ git add <filename>
```

在合并改动之前，你可以使用如下命令预览差异：

```
$ git diff <source_branch> <target_branch>
```

快速Git使用 - windows

...或在命令行上创建一个新的存储库

```
echo "Github usage." > README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/fengjunhuii/makeru.git
git push -u origin master
```

命令行推送现有库

```
从命令行推送到存储库
git remote add origin https://github.com/fengjunhuii/makeru.git
git push -u origin master
```

```
或从另一个存储库导入代码
您可以使用Subversion，Mercurial或TFS项目中的代码初始化此存储库。
```

链接与资源

图形化客户端

- GitX (L) (OSX, 开源软件)
- Tower (OSX)
- Source Tree (OSX, 免费)
- GitHub for Mac (OSX, 免费)
- GitBox (OSX, App Store)

指南和手册

- Git 社区参考书
- 专业 Git
- 像 git 那样思考
- GitHub 帮助
- 图解 Git

相关文章

- Github 简明指南：http://rogerdudler.github.io/git-guide/index.zh.html
- 如何高效利用GitHub:http://www.yangzhiping.com/tech/github.html