Git：分布式版本控制软件

Git是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。Linux系统的创始大神就是Linus Torvalds ，他花了1周的时间，写出来出版的Git,真心牛。

程序员写出的软件，随着时间和用户需求，会产生各钟各样的版本，怎么办？最蠢笨的方法是，为每个版本建个文件夹，然后赋值源码到各个文件夹，单实际的操作中，增删改查，会增加很多工作量，把你累的不如一条狗。

我估算90%的软件公司会使用到Git(或其它工具),来做软件的管理开发。除了软件工程师可以用到Git,搞硬件的，

搞电子，单片机的都可用到。只要你写的程序，画的PCB板，要分类型，分型号，你都会需要用到Git.

Git 是分布式的？ 啥意思？ 就是 你做的工程项目文件，可以保存在自己的电脑上，同时还可以上传到Git的服务器Github 上(在全球任何地方都可以下载)。这样做，就不用担心，硬盘坏了。。（话说固态SSD坏了，数据不可以恢复，不知是真是假）

Git 学习资源：

菜鸟教程：<http://www.runoob.com/git/git-tutorial.html>

网易云课堂：<https://ke.qq.com/course/170986>

以下是我个人的笔记，可能不适合初学者看。

常用命令：

Git add: 将本地文件增加到暂存区

Git commit: 将暂存区的内容提交到本地仓库(分支1,2,3,4) 默认分支master

Git push: 将本地仓库的内容推送到 远程仓库（远程分支）

Git pull: 将远程长裤(远程分支)的内容拉取到本地仓库

**Git 的安装：**去git官网下载软件，默认下一步安装。

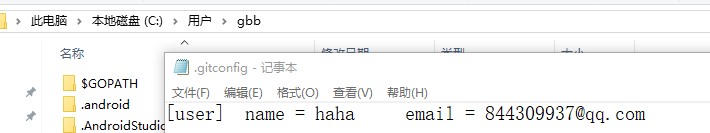
安装完成后需要注意配置下环境变量。（新版的git工具，可能会自动添加PATH路径）

将 C:\Program Files\Git\bin 添加到PATH

**配置Git:** 配置本地的用户名和邮箱(用户名，邮箱名，写自己的，也可以随便写)



命令执行完成后，会发现

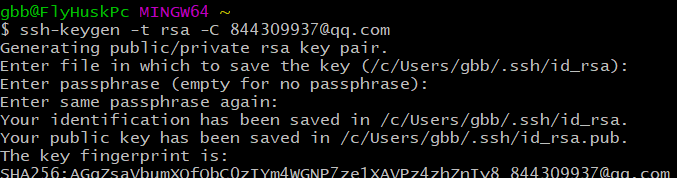


Git 统一的托管网站 Github, 全球最大的同性交友网站，哈哈。

我的GitHub 账号 username: FlyHusky Password: \*\*\*\*\*\*\* 2018luofang@gmail.com

配置免密钥登录到github, 用ssh key. 先在本地配置，让后发送给远程。

输入： ssh-keygen -t rsa -C [844309937@qq.com](mailto:844309937@qq.com) 然后一路回车



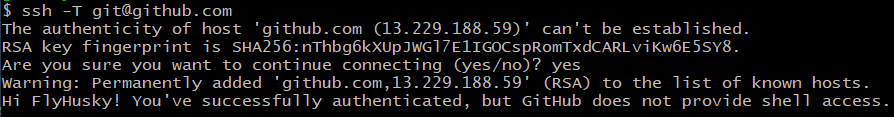
文件已经保存到： id\_rsa 是私钥， id\_ras.pub 是公钥。

key has been saved in /c/Users/gbb/.ssh/id\_rsa.pub.

打开Github setting , 点左侧 “SSH and GPG keys”, 在点“New SSH key”

Title 起个响亮个名字， key: 打开id\_ras.pub ，将其内容拷贝到 文本框内。 然后 Add ssh

**测试连接下： 固定命令：ssh -T git@github.com**

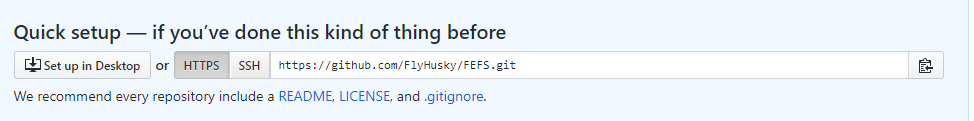


项目测试： 本地建个项目，推送到远程github

我有个项目叫 FEFS 在：E:\go\_gopath\src\github.com\goinaction\code\FEFS

cd到FEFS目录下，输入命令：git init 然后会发现FEFS目录下多了一个 .git 隐藏文件夹。

到github 网上创建一个 res 仓库



仓库项目标识符 HTTPS： <https://github.com/FlyHusky/FEFS.git>

SSH: [git@github.com:FlyHusky/FEFS.git](mailto:git@github.com:FlyHusky/FEFS.git)

本地项目和远程项目关联： git remote add origin 网址 。 如下图



项目交互：

第一次发布项目： （本地-远程）

git add . 将当前本地文件增加到暂存区。 . 表示当前目录

git commit -m “注释内容” 将暂存区的内容提交到本地仓库

git push -u origin master 将本地仓库的内容推送到 远程仓库, 因为是第一次，所以要写 origin master

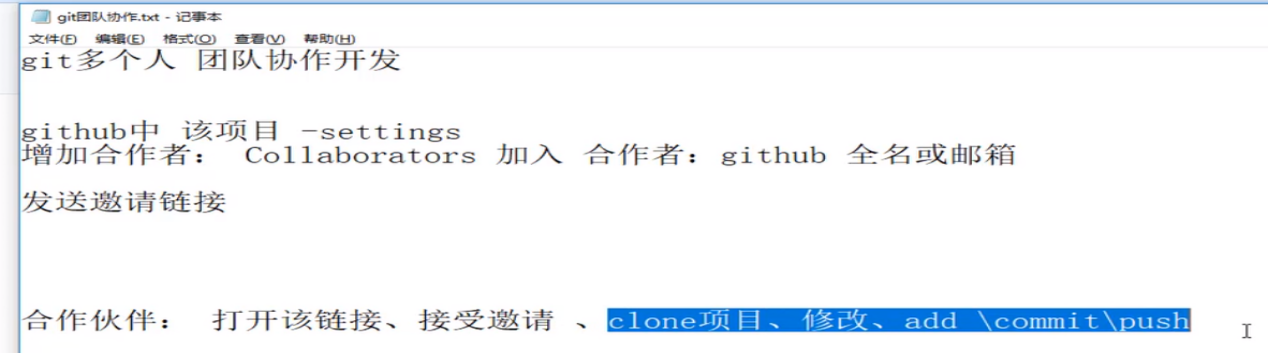
执行完上述命令，就可以去github 看到我发布的项目了。。。

第一次下载项目： git clone [git@github.com:FlyHusky/FEFS.git](mailto:git@github.com:FlyHusky/FEFS.git) 简单。

提交 本地-远程： git add. 和 git commit -m “提交到分支了” 和 git push origin master

下载 远程-本地： 将github 上的代码项目，pull 到本地 git pull 既可。

团队协作：



GitHub 上 Branch 分支

仓库中默认的分支只要一个即master. 当我们在开发过程中接到一个新的功能需求，我们可以创建一个新的branch,或者更多的branch, 等到开发完成后，在将分支合并到master上。

几个人合作用开发项目时，代码保存到GitHub上，我们不可能在原有代码上直接修改调试，这时就要创建一个新的分支，在分支上改自己的代码，修改完成后，把分支上修改的代码合并到主分支master上就好了。这个过程需要经过以下几个步骤：

**A ：可以在github 完整上创建一个分支，很简单，如下图：**

|  |  |
| --- | --- |
|  | 点击 Branch ，出现一个框，  查找或创建一个branch.  若要创建，只需要输入一个新的branch  名字，回车，就创建了一个新的分支。 |

**B：在本地 bash shell 中 用命令创建分支，并提交分支。**

**B.1 git branch test1 //创建了一个分支叫 test1**

**B.2 git checkout test1 //切换到分支test1**

**然后就修改写项目内容**

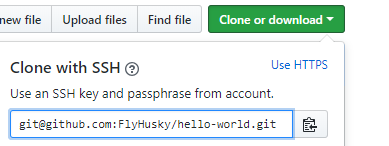
**B.3 git add . B.4 git commit -m “分支 test1 测试”**

**B.5 git push --set-upstream origin test1 //第一次要用 --set-upstream origin test1**

**//以后提交 就写 git add . --> git commit -m “提交信息” -> git push 就好了。**

**C：演示 从 github 上clone 一个项目， 然后修改，在提交。 当时是我的github, 先前创建了一个hello-world 项目。**

**C.1 选用ssh clone**



**Bash shell 输入： git clone [git@github.com:FlyHusky/hello-world.git](mailto:git@github.com:FlyHusky/hello-world.git)**

**然后你就发现当前的工作目录，多了一个hello-world 项目文件夹。**

**可以进去修改或添加一些文件**

**C.2： bash shell 里 cd 到 hello-world**

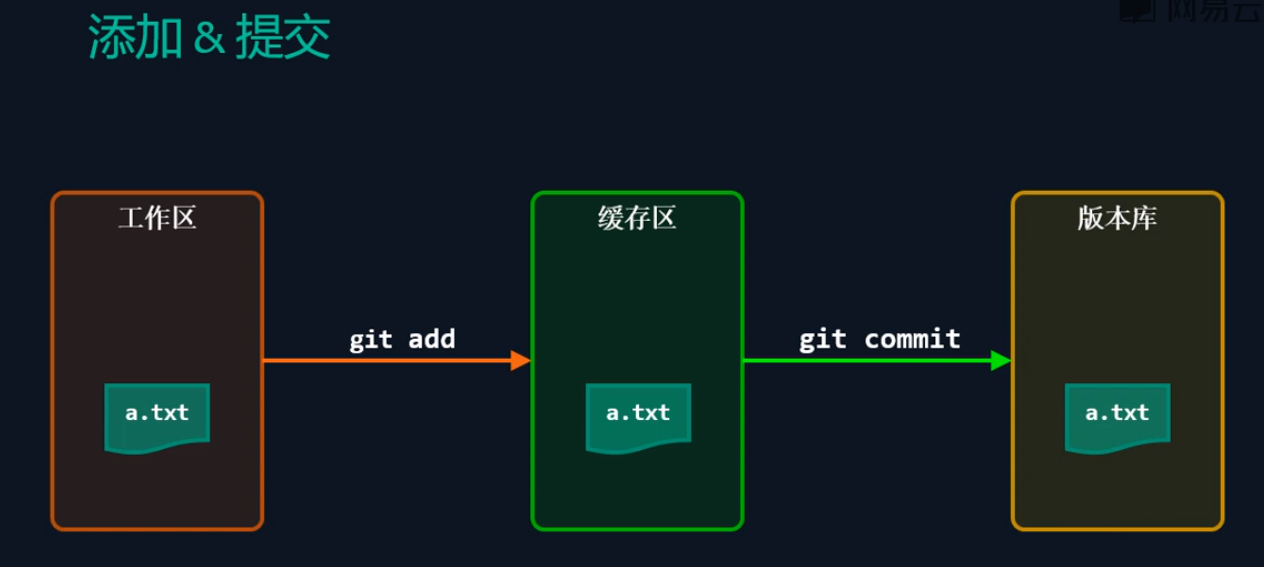
**C.3： git add .**

**C.4： git commit -m “测试提交” c.3 和c.4 也可以只用一个命令： git commit -ad -m “测提交”**

**C.5： git push //over done**

**D: 分支的 合并**

**Git 的版本控制系统说明：**



**Git add file.name 将file.name添加到缓存区，同时git也会追踪file.name是否变化。**

**3：git init 在项目的文件夹内，反键，选择”Git bash here”,就可以打开git的命令终端。然后输入**

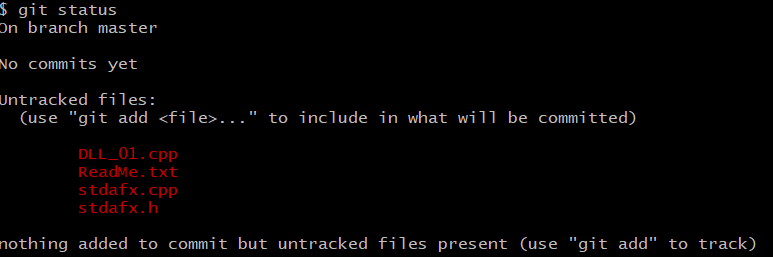
**git init 就可以给文件夹内的所有文件添加了版本控制，同时你会发现在当前文件夹内会多出一个隐藏的文件夹叫”.git”。**

**4：git add filename将文件添加到”缓冲区”，git会追踪缓冲区内的文件是否变化。**

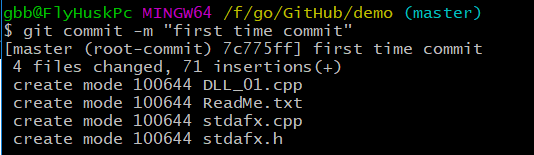
**4.1 git add filename 将单个文件添加到缓冲**

**4.2 git add . 将当前整个文件夹内的文件添加到缓冲区。**

**5：git status 常看文件状态。比如下面提示，当前是在“master”版本上，没有“commit”,还列出了没有被追踪的文件untracked files。**



**6:git commit** 将文件从缓冲区添加到版本库。git commit -m “first time commit”



**7:git diff --cached 比较缓冲区和版本库， git diff HEAD 比较工作区和版本库。**

**8:git checkout HEAD filename . 从版本库中取出filename到工作区。** 比如我将工作区内的文件修改后，后悔了，想恢复到以前提交到版本库的样子。

**git checkout HEAD hello.cpp 将版本库的hello.cpp 恢复到工作区。**

**git checkout HEAD . 将版本库中的所有文件恢复到工作区。**

**9：git checkout -- . 将暂存区内的所有文件恢复到工作区**，如果暂存区没有，则到版本库中去取文件。

**10：分支概念 branch. git branch branchname 创建一个分支。** 为什么会有分支？ 比如我们搞了个项目前期开发完成了某个模块，然后客户要测试用，那么这时就可以创建一个分支1，将分支1交付给客户，然后研发部继续在master主区，继续开发。 或者是发布一个新的版本软件时，生成一个分支，起名叫V几点几。然后直接把分支发布出去，然后在一直开发master. 这就是版本控制系统。为什么说git是分布式的版本控制系统，因为当我们使用git hub后，可以联合其他人一起开发项目，每个人电脑上都有一条版本，所以是分布式的。

**git branch v0.1 :创建一个名叫v0.1 的版本库。**Git在创建v0.1时，会将master版本库赋值拷贝给v0.1.举例比如master库中三个文件1.txt, 2.txt, 3.txt，创建v0.1版本库后，V0.1内也会有有1.txt,2.txt,3.txt。

**Git checkout v0.1** 切换到v0.1 版本库，我们在工作空间内修改文件，比如这里增加一个文件4.txt.

然后git add . 在 git commit -m “add 4.txt” 。

Git checkout master 切换到master, 这个时候，会发现工作目录空间少了4.txt. 腻害把，git在切换工作空间后，会把对于版本库中的文件恢复到工作空间。这就是git的功能，很强大把。哈哈哈哈哈！！！

2018年8月12日21:39:03， 愿所有付出，终将会有回报。Git 所有coder 都必须会的工具。

**11： git checkout -b branch-name 创建分支branch-name 同时切换到branch-name.**

**12: 分支的合并。**如上面，项目需要测试一个功能，于是创建了一个叫 v-test的版本库，在v-test库内修改创建了一些文件，当测试完成后，没问题了，想要将v-test修改的东西合并到master,则么做呢？

比如当前在master下，要合并v-test 用， git merge v-test 既可。

**13：前文已经说过了，在github上托管自己的远程仓库。也可以在自己的服务器上管理远程仓库。**

**比如在我的阿里云服务器上，用ssh远程登录我的ecs, 然后建立一个目录/root/git/my-c 在my-c目录下输入命令：git init --bare 既可建立一个空的仓库。**

**然后在本地的工作目录空间输入命令**

git push --set-upstream ssh://root@47.96.0.254/root/git/my-c master

按提示输入远程服务主机密码，既可push到远程主机。

**以后再push时的命令是：**

git push ssh://root@47.96.0.254/root/git/my-c master

Clone 下载命令：git clone ssh://root@47.96.0.254/root/git/my-c

其实一般是将my-c文件夹改为 my-c.git 这样的名字。

当然了，也可以用如下命令来**关联远程的仓库。**

git remote add origin ssh://root@47.96.0.254/root/git/my-c master

后面的push 就可以直接写 git push 了。

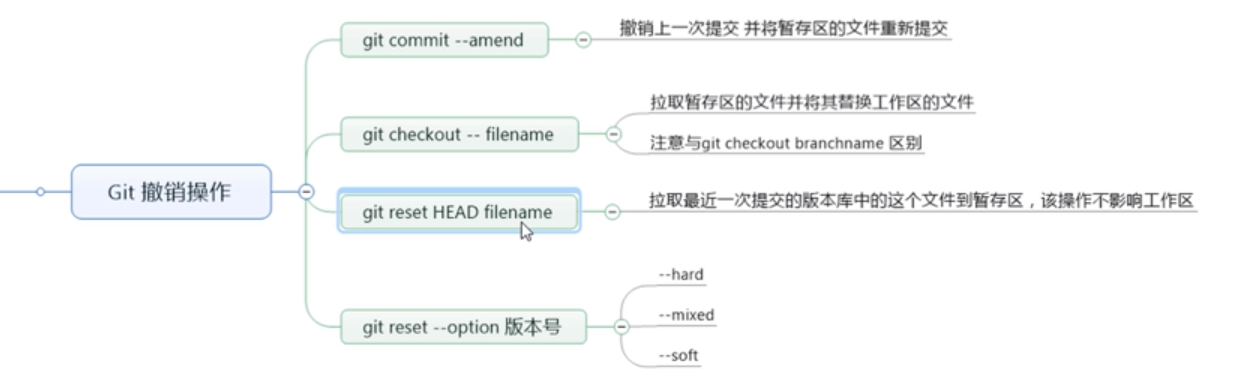
**13.1 如何避免每次都要输入密码**

在本地输入命令：ssh-copy-id [root@47.96.0.254](mailto:root@47.96.0.254) ，git会现在本地创建密匙，然后将公匙上传到远程主机，以后，都不用输入远程主机密码了。

**14 如何设置某些文件不被追踪，不被提交到版本库。 在工作目录创建文件 .gitignore**

在文件内可以写入要忽略的文件名或者文件夹的名字，比如有些编译生成的中间文件，缓成文件等。

如果要忽略.o 文件， 在文件内添加 \*.o



2018年11月29日

在github 上创建好一个仓库后，会有详细的操作提示，如下图示例。

