代码提交

代码提交一般有五个步骤：

1.查看目前代码的修改状态

2.查看代码修改内容

3.暂存需要提交的文件

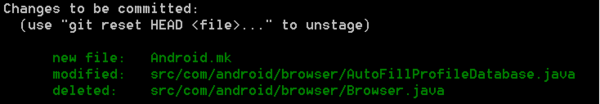
4.提交已暂存的文件

5.同步到服务器

1.     查看目前代码的修改状态

提交代码之前，首先应该检查目前所做的修改，运行git status命令

a)        已暂存 （changes to be committed）

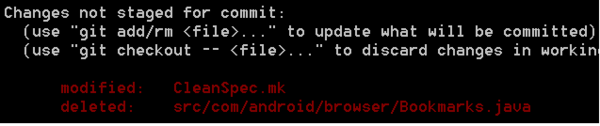
[](https://gss0.baidu.com/-4o3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/fd039245d688d43f0319e235781ed21b0ff43b40.jpg)

new file //表示新建文件

modified //表示修改文件

deleted //表示删除文件

b)       已修改 （changed but not updated）

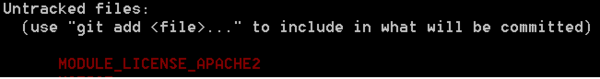
[](https://gss0.baidu.com/-Po3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/622762d0f703918f43074e0d543d269758eec427.jpg)

modified //表示修改文件

deleted //表示删除文件

另外，git 给出了可能需要的操作命令，git add/rm, gitcheckout --

c)        未跟踪 （untracked files）

[](https://gss0.baidu.com/-vo3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/72f082025aafa40feedbbc1aae64034f79f019d3.jpg)

2.     查看代码修改的内容

比较某文件与最近提交节点的差异。

注意：如果该文件已暂存，那么应该使用git diff --cached<file>

 git diff <hashcode> <hashcode>  <file>

比较某文件在提交节点a，节点b的差异。

技巧：如果省略后面一个hashcode，则默认表示与上一提交节点比较。（也可以利用^运算符）

3.     暂存需要提交的文件

如果是新建的文件

则git add  <file>

如果是修改的文件则git add  <file>如果是删除的文件则 git rm  <file>

4.     提交已暂存的文件

git commit

注意注释填写规范。

git commit --amend

修改最近一次提交。有时候如果提交注释书写有误或者漏提文件，可以使用此命令。

5.     同步到服务器

同步到服务器前先需要将服务器代码同步到本地

命令： git pull

如果执行失败，就按照提示还原有冲突的文件，然后再次尝试同步。

命令：git checkout -- <有冲突的文件路径>

同步到服务器

命令： git push origin  <本地分支名>

如果执行失败，一般是没有将服务器代码同步到本地导致的，先执行上面的git pull命令。

我们应该能够看到输出的用户信息。

GIT 命令

新建一个仓库

为了建立一个新仓库，运行如下命令：

$ git init

查找一个仓库

为了查找一个仓库，命令如下：

$ git grep "repository"

与远程仓库连接

为了与远程仓库连接，运行如下命令：

$ git remote add origin remote\_server

然后检查所有配置的远程服务器，运行如下命令：

$ git remote -v

克隆一个仓库

为了从本地服务器克隆一个仓库，运行如下代码：

$ git clone repository\_path

如果我们想克隆远程服务器上的一个仓库，那克隆这个仓库的命令是：

在仓库中列出分支

为了检查所有可用的和当前工作的分支列表，执行：

$ git branch

创建新分支

创建并使用一个新分支，命令是：

$ git checkout -b 'branchname'

删除一个分支

为了删除一个分支，执行：

$ git branch -d 'branchname'

为了删除远程仓库的一个分支，执行：

$ git push origin:'branchname'

切换到另一个分支

从当前分支切换到另一个分支，使用

$ git checkout 'branchname'

添加文件

添加文件到仓库，执行：

$ git add filename

文件状态

检查文件状态 (那些将要提交或者添加的文件)，执行：

$ git status

提交变更

在我们添加一个文件或者对一个文件作出变更之后，我们通过运行下面命令来提交代码：

$ git commit -a

提交变更到 head 但不提交到远程仓库，命令是：

$ git commit -m "message"

推送变更

推送对该仓库 master 分支所做的变更，运行：

$ git push origin master

推送分支到仓库

推送对单一分支做出的变更到远程仓库，运行：

$ git push origin 'branchname'

推送所有分支到远程仓库，运行：

$ git push -all origin

合并两个分支

合并另一个分支到当前活动分支，使用命令：

$ git merge 'branchname'

从远端服务器合并到本地服务器

从远端服务器下载/拉取变更到到本地服务器的工作目录，运行：

$ git pull

检查合并冲突

查看对库文件的合并冲突，运行：

$ git diff -base 'filename'

查看所有冲突，运行：

$ git diff

如果我们在合并之前想预览所有变更，运行：

$ git diff 'source-branch' 'target-branch'

创建标记

创建标记来标志任一重要的变更，运行：

$ git tag 'tag number' 'commit id'

通过运行以下命令，我们可以查找 commit id ：

$ git log

推送标记

推送所有创建的标记到远端服务器，运行：

$ git push -tags origin

回复做出的变更

如果我们想用 head 中最后一次变更来替换对当前工作树的变更，运行：

$ git checkout -'filename'

我们也可以从远端服务器获取最新的历史，并且将它指向本地仓库的 master 分支,而不是丢弃掉所有本地所做所有变更。为了这么做，运行：

$ git fetch origin $ git reset -hard master

如果您从未使用过Git，甚至可能从未听说过它。莫慌张，只需要一步步地跟着这篇入门教程，很快您就会在GitHub上拥有一个全新的 Git 仓库。

在开始之前，让我们先理清一个常见的误解：Git 并不是 GitHub。Git 是一套版本控制系统（或者说是一款软件），能够协助您跟踪计算机程序和文件在任何时间的更改。它同样允许您在程序、代码和文件操作上与同事协作。GitHub 以及类似服务（包括 GitLab 和 BitBucket）都属于部署了 Git 程序的网站，能够托管您的代码。

步骤 1：申请一个 GitHub 账户

在GitHub.com网站上（免费）创建一个账户是最简单的方式。

选择一个用户名（比如说，octocat123），输入您的邮箱地址和密码，然后点击Sign up for GitHub。进入之后，您将看到下方插图的界面：

步骤 2：创建一个新的仓库

一个仓库（ repository），类似于能储存物品的场所或是容器；在这里，我们创建仓库存储代码。在+符号（在插图的右上角，我已经选中它了） 的下拉菜单中选择New Repository。

给您的仓库命名（比如说，Demo）然后点击Create Repository。无需考虑本页面的其他选项。

恭喜！您已经在 GitHub.com 中建立了您的第一个仓库。

步骤 3: 创建文件

当仓库创建完毕后，界面将和下方一致：

不必惊慌，它比看上去简单。跟紧步骤。忽略其他内容，注意截图上的 “...or create a new repository on the command line,”。

在您的计算机中打开终端。

键入git然后回车。如果命令行显示bash: git: command not found，在您的操作系统或发行版安装 Git命令。键入git并回车检查是否成功安装；如果安装成功，您将看见大量关于使用该命令的说明信息。

在终端内输入：

mkdir Demo

这个命令将会创建一个名为 Demo 的目录（文件夹）。

如下命令将会切换终端目录，跳转到 Demo 目录：

cd Demo

然后输入：

echo "#Demo" >> README.md

创建一个名为README.md的文件，并写入#Demo。检查文件是否创建成功，请输入：

cat README.md

这将会为您显示README.md文件的内容，如果文件创建成功，您的终端会有如下显示：

使用 Git 程序告诉您的电脑，Demo 是一个被 Git 管理的目录，请输入：

git init

然后，告诉 Git 程序您关心的文件并且想在此刻起跟踪它的任何改变，请输入：

git add README.md

步骤 4：创建一次提交

目前为止，您已经创建了一个文件，并且已经通知了 Git，现在，是时候创建一次提交commit了。提交可以看作是一个里程碑。每当完成一些工作之时，您都可以创建一次提交，保存文件当前版本，这样一来，您可以返回之前的版本，并且查看那时候的文件内容。无论何时您修改了文件，都可以对文件创建一个上一次的不一样的新版本。

创建一次提交，请输入：

git commit -m "first commit"

就是这样！刚才您创建了包含一条注释为 “first commit” 的 Git 提交。每次提交，您都必须编辑注释信息；它不仅能协助您识别提交，而且能让您理解此时您对文件做了什么修改。这样到了明天，如果您在文件中添加新的代码，您可以写一句提交信息：“添加了新的代码”，然后当您一个月后回来查看提交记录或者 Git 日志（即提交列表），您还能知道当时的您在文件夹里做了什么。

步骤 5: 将您的计算机与 GitHub 仓库相连接

现在，是时候用如下命令将您的计算机连接到 GitHub 仓库了：

git remote add origin链接

让我们一步步的分析这行命令。我们通知 Git 去添加一个叫做origin（起源）的，拥有地址为链接（它也是您的仓库的 GitHub 地址） 的remote（远程仓库）。当您提交代码时，这允许您在 GitHub.com 和 Git 仓库交互时使用origin这个名称而不是完整的 Git 地址。为什么叫做origin？当然，您可以叫点别的，只要您喜欢（惯例而已）。

现在，我们已经将本地 Demo 仓库副本连接到了其在 GitHub.com 远程副本上。您的终端看起来如下：

此刻我们已经连接到远程仓库，可以推送我们的代码 到 GitHub.com（例如上传README.md文件）。

执行完毕后，您的终端会显示如下信息：

然后，如果您访问链接，您会看到截图内显示的情况：

就是这么回事！您已经创建了您的第一个 GitHub 仓库，连接到了您的电脑，并且从你的计算机推送（或者称：上传）一个文件到 GitHub.com 名叫 Demo 的远程仓库上了。下一次，我将编写关于 Git 复制（从 GitHub 上下载文件到你的计算机上）、添加新文件、修改现存文件、推送（上传）文件到 GitHub。

# 1. 浏览托管项目

在GitHub的页面中可以使用键盘快捷键

（1）按下问号（?）会在弹出窗口显示当前页面可用的快捷键。

（2）在项目的代码浏览页按下字母“w”，弹出分支切换菜单。

（3）按下字母“t”，开启目录树中文件查找和过滤。

# 2. 项目托管

## 2.1 创建新项目

有2种办法：（1）在GitHub创建新项目；（2）从已有版本库创建，然后 remote push到GitHub

### 2.1.1 在GitHub创建新项目

（1）在GitHub首页 “New repository”，创建新版本库 HelloWorld

（2）在本地使用Git Bash，将repository clone到本地。

$ git clone <https://github.com/zhchnchn/HelloWorld.git>

（3）在本地HelloWorld目录下创建 README.md 文件。

以扩展名.md，.mkd，.mkdn，.mdown，.markdown等为结尾的文件，均以Markdown标记语言语法进行解析并显示。

（4）添加README.md文件并提交。

$ git add README.md

$ git commit -m "README for this project."

（5）向GitHub推送，完成版本库初始化。

$ git push origin master

（6）然后刷新GitHub上HelloWorld项目的首页，可见版本库包含了一个新的提交。

（7）如何删除创建的版本库？

在HelloWorld项目首页的右方，点击”Settings“->在红色的”Danger Zone“区域，点击”Delete this repository“->输入项目名HelloWorld确认删除。

如果本地clone的版本库不需要了，则手动删除之。

### 2.1.2 从已有版本库创建

（1）使用Git Bash在本地建立一个Git版本库。

$ mkdir HelloWorld  
$ cd HelloWorld  
$ git init

（2）然后在版本库中添加README.md文件  
$ git add README.md  
$ git commit -m "README for this project."

（3）为版本库添加名为origin的远程版本库  
$ git remote add origin [git@github.com:zhchnchn/HelloWorld.git](mailto:git@github.com:zhchnchn/HelloWorld.git)

（4）执行推送命令，完成GitHub版本库的初始化。注意命令行中的-u参数，在推送成功后自动建立本地分支与远程版本库分支的追踪。  
$ git push -u origin master

注：这一步没有成功，显示错误信息。

remote: Repository not found.

fatal: repository 'https://github.com/zhchnchn/HelloWorld.git/' not found

 尝试了很多种方法都没有解决，有可能公司网络防火墙禁止了SSH操作。

2014-05-20，NOTE:

换了另外一种方式成功了：

（1）-（2）步骤与前面相同

（3）为版本库添加名为origin的远程版本库  
$ git remote add origin [https://github.com/zhchnchn/HelloWorld.git](https://github.com/zhchnchn/PushTest.git)

（4）执行推送命令

$ git push origin master

提示错误：

remote: Repository not found.

fatal: repository 'https://github.com/zhchnchn/HelloWorld.git/' not found

（5）在GitHub主页创建HelloWorld仓库，注意不要添加README.md等任何文件。  
（6）创建完成后，再次git push origin master，这次终于成功了。

[IMG_256](http://www.cnblogs.com/zhcncn/p/javascript:void(0);)

$ git push origin master

Counting objects: 12, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (12/12), done.

Writing objects: 100% (12/12), 25.02 KiB | 0 bytes/s, done.

Total 12 (delta 4), reused 0 (delta 0)

To <https://github.com/zhchnchn/HelloWorld.git>

\* [new branch] master -> master

[IMG_257](http://www.cnblogs.com/zhcncn/p/javascript:void(0);)

 2014-05-21，NOTE:

 在 <http://www.cnblogs.com/plinx/archive/2013/04/08/3009159.html>中提到，使用 SSH来git remote add origin时，出现了Repository not found的问题。而在我本机上SSH不可以，HTTPS可以。说明我本机不支持SSH协议。

## 2.2 在不同电脑上git push同个github账户下的repositories

参考【2】。

# 3. 公钥认证管理

开发者向GitHub版本库写入最常用到的协议是SSH协议，因为SSH协议使用公钥认证，可以实现无口令访问，而若使用HTTPS协议每次身份认证时都需要提供口令.

但是，可以通过在文件~/.netrc中写入明文口令实现使用 HTTPS 协议时也能自动完成认证。具体格式参见ftp命令的MAN手册中相关介绍。

具体设置参见：<http://www.cnblogs.com/zhcncn/p/3681209.html> -- 如何配置，在向Github去 git push 时不用输入用户名密码？

# 4. 建立主页

## 4.1 创建个人主页

GitHub 为每一个用户分配了一个二级域名<user-id>.github.io，用户为自己的二级域名创建主页很容易，只要在托管空间下创建一个名为<user-id>.github.io的版本库，向其master分支提交网站静态页面即可，其中网站首页为index.html.

访问网址： <http://gotgithub.github.io/>

要注意访问用户二级域名的主页要使用HTTP协议非HTTPS协议.

## 4.2 创建项目主页

GitHub会为每个账号分配一个二级域名<user-id>.github.io作为用户的首页地址。实际上还可以为每个项目设置主页，项目主页也通过此二级域名进行访问。例如gotgithub用户创建的helloworld项目如果启用了项目主页，则可通过网址<http://gotgithub.github.io/helloworld/>访问.

为项目启用项目主页很简单，只需要在项目版本库中创建一个名为gh-pages的分支，并向其中添加静态网页即可。

# References

【1】GotGitHub （<http://www.worldhello.net/gotgithub/index.html>）

【2】在不同电脑上git push同个github账户下的repositories（http://yulijia.net/cn/%E7%9F%A5%E8%A1%8C%E5%B9%B6%E8%BF%9B/2013/02/06/use-one-github-account-on-two-computers.html）