# [【Github教程】史上最全github使用方法：github入门到精通](http://blog.csdn.net/hcbbt/article/details/11651229)

原文 <http://www.eoeandroid.com/thread-274556-1-1.html>

**【初识Github】**

首先让我们大家一起喊一句“Hello Github”。YEAH!就是这样。  
  
Git是一个分布式的版本控制系统，最初由Linus Torvalds编写，用作Linux内核代码的管理。在推出后，Git在其它项目中也取得了很大成功，尤其是在Ruby社区中。目前，包括Rubinius和Merb在内的很多知名项目都使用了Git。Git同样可以被诸如Capistrano和Vlad the Deployer这样的部署工具所使用。同样，eoe.cn客户端的源码也托管在github上。

GitHub可以托管各种git库，并提供一个web界面，但与其它像 SourceForge或Google Code这样的服务不同，GitHub的独特卖点在于从另外一个项目进行分支的简易性。为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。已经有人将GitHub称为代码玩家的MySpace。  
  
在GitHub进行分支就像在Myspace（或Facebook…）进行交友一样，在社会关系图的节点中不断的连线。  
  
GitHub 使用 git 分布式版本控制系统，而 git 最初是 Linus Torvalds 为帮助Linux开发而创造的，它针对的是 Linux 平台，因此 git 和 Windows 从来不是最好的朋友，因为它一点也不像 Windows。GitHub 发布了GitHub for Windows，为 Windows 平台开发者提供了一个易于使用的 Git 图形客户端。

**GitHub For Windows**

GitHub for Windows 是一个 Metro 风格应用程序，集成了自包含版本的 Git，bash 命令行 shell，PowerShell 的 posh-git 扩展。GitHub 为 Windows 用户提供了一个基本的图形前端去处理大部分常用版本控制任务，可以创建版本库，向本地版本库递交补丁，在本地和远程版本库之间同步。微软也通过CodePlex向开发者提供 git 版本控制系统，而 GitHub 创造了一个更具有吸引力的 Windows 版本。  
  
GitHub上已自动配置的Mac笔记本电脑，一个工具，可以转换设置Linux或Windows机器。  
  
BOXEN是GitHub的自动化工具，设置和配置的Mac笔记本电脑软件开发[3]或其他类型的工作，正在使用他们的开发人员，律师，设计师，付货人，等。我们的想法是准备系统以自动方式和作为无差错尽可能用最少的干预工作。根据GitHub上，与一个新的开发机器上，他的Mac系统成立，并准备在30分钟内提交代码。  
  
BOXEN的基础上收集了大量的几十个木偶模块，使设置的各种软件，如卡桑德拉，MongoDB中，Java软件中，Python和Ruby开发中，节点，JS，nginx的，Skype公司，甚至MINECRAFT。虽然机器上配备了一个预配置，每个用户都可以调整它的配置应有的作用。  
  
**【如何使用】**  
**1.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。之后就可以创建仓库了（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。  
  
**2.安装客户端msysgit**  
github是服务端，要想在自己电脑上使用git我们还需要一个git客户端，我这里选用msysgit，这个只是提供了git的核心功能，而且是基于命令行的。如果想要图形界面的话只要在msysgit的基础上安装TortoiseGit即可。  
  
装完msysgit后右键鼠标会多出一些选项来，在本地仓库里右键选择Git Init Here，会多出来一个.git文件夹，这就表示本地git创建成功。右键Git Bash进入git命令行，为了把本地的仓库传到github，还需要配置ssh key。  
  
**3.配置Git**

首先在本地创建ssh key；

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. ssh-keygen -t rsa -C "your\_email@youremail.com"

后面的[your\_email@youremail.com](mailto:your_email@youremail.com)改为你的邮箱，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。

回到github，进入Account Settings，左边选择SSH Keys，Add SSH Key,title随便填，粘贴key。为了验证是否成功，在git bash下输入：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. $ 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。

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. $ git config --global user.name "your name"
2. $ git config --global user.email "your\_email@youremail.com"

进入要上传的仓库，右键git bash，添加远程地址：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. $ git remote add origin git@github.com:yourName/yourRepo.git

后面的yourName和yourRepo表示你再github的用户名和刚才新建的仓库，加完之后进入.git，打开config，这里会多出一个remote “origin”内容，这就是刚才添加的远程地址，也可以直接修改config来配置远程地址。  
  
**4.提交、上传**

接下来在本地仓库里添加一些文件，比如README，

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. $ git add README
2. $ git commit -m "first commit"

上传到github：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. $ git push origin master

git push命令会将本地仓库推送到远程服务器。  
git pull命令则相反。  
  
修改完代码后，使用git status可以查看文件的差别，使用git add 添加要commit的文件，也可以用git add -i来智能添加文件。之后git commit提交本次修改，git push上传到github。  
  
**5.gitignore文件**

.gitignore顾名思义就是告诉git需要忽略的文件，这是一个很重要并且很实用的文件。一般我们写完代码后会执行编译、调试等操作，这期间会产生很多中间文件和可执行文件，这些都不是代码文件，是不需要git来管理的。我们在git status的时候会看到很多这样的文件，如果用git add -A来添加的话会把他们都加进去，而手动一个个添加的话也太麻烦了。这时我们就需要.gitignore了。比如一般c#的项目我的.gitignore是这样写的：

**[plain]** [view plaincopyprint?](http://blog.csdn.net/hcbbt/article/details/11651229)

1. bin
2. \*.suo
3. obj

bin和obj是编译目录，里面都不是源代码，忽略；suo文件是vs2010的配置文件，不需要。这样你在git status的时候就只会看到源代码文件了，就可以放心的git add -A了。  
  
**6.tag**  
  
我们可以创建一个tag来指向软件开发中的一个关键时期，比如版本号更新的时候可以建一个“v2.0”、“v3.1”之类的标签，这样在以后回顾的时候会比较方便。tag的使用很简单，主要操作有：查看tag、创建tag、验证tag以及共享tag，这些下面的博客中有详细讲解。  
  
  
**【Github的相关使用文章】  
Git介绍,安装,Git+Git flow使用：**[**http://my.eoe.cn/fogs/archive/799.html**](http://my.eoe.cn/fogs/archive/799.html)  
**Git 指令集：**[**http://my.eoe.cn/iceskysl/archive/463.html**](http://my.eoe.cn/iceskysl/archive/463.html)  
**在mac上安装git-flow过程：**[**http://my.eoe.cn/iceskysl/archive/118.html**](http://my.eoe.cn/iceskysl/archive/118.html)  
**git fetch 的简单用法:更新远程代码到本地仓库：**[**http://my.eoe.cn/com360/archive/3533.html**](http://my.eoe.cn/com360/archive/3533.html)  
**git 如何让单个文件回退到指定的版本：**[**http://my.eoe.cn/com360/archive/3351.html**](http://my.eoe.cn/com360/archive/3351.html)  
**如何使用Github上的开源项目：**[**http://my.eoe.cn/fengyiyezi/archive/3427.html**](http://my.eoe.cn/fengyiyezi/archive/3427.html)  
**Window（8）下安装 MSysGit 、gitflow 、GitHub：**[**http://my.eoe.cn/sunxun/archive/158.html**](http://my.eoe.cn/sunxun/archive/158.html)  
**git 打tag：**[**http://my.eoe.cn/xiayang6/archive/446.html**](http://my.eoe.cn/xiayang6/archive/446.html)  
**基于Github参与eoe的开源项目指南：**[**http://my.eoe.cn/iceskysl/archive/3195.html**](http://my.eoe.cn/iceskysl/archive/3195.html)  
**Git stash 使用方法：**[**http://my.eoe.cn/sunxun/archive/190.html**](http://my.eoe.cn/sunxun/archive/190.html)  
**Git tag的使用：**[**http://my.eoe.cn/futurexiong/archive/1943.html**](http://my.eoe.cn/futurexiong/archive/1943.html)

**大白话讲解如何给github上项目贡献代码：**[**http://my.eoe.cn/leigo/archive/3221.html**](http://my.eoe.cn/leigo/archive/3221.html)

最后再推荐几篇社区里介绍知道github使用的帖子：  
**githup的使用**  
<http://www.eoeandroid.com/thread-272837-1-1.html>  
  
**Eclipse上GIT插件EGIT使用手册**  
<http://www.eoeandroid.com/thread-273360-1-1.html>  
  
**ubuntu下git服务器的搭建**  
<http://www.eoeandroid.com/thread-273167-1-1.html>  
  
git ， vim ， ls 全局配置  
<http://www.eoeandroid.com/thread-229638-1-1.html>  
  
以上希望对大家学习起到积极的作用，一个好的程序猿势必要学会github的使用。如果本文大家觉得还不错，就告诉你身边的朋友吧，如果觉得看得过去那么就分享一下吧，如果觉得有待修改，那么请指出不足并且给打赏几个e币。最后感谢百度的无私支持，以及某个人的博客（说真的真的忘了他的地址是啥了），Hello Github。

# 【Github教程】实战命令1 – git方式

1. git init // 建立初始库
2. git add . // 添加所有文件到索引
3. git add ls // 添加ls目录
4. git config --global user.email "mooling@yeah.net"
5. git config --global user.name "mooling"
6. git commit –m “comments” // 提交变更
7. git remote add emacs-profile <https://github.com/mooling/emacs-profile.git>
8. git push -u emacs-profile master //-u 是提示用户名、密码

删除文件用 git rm

改文件名用 git mv

看远端库用 git remote

经常使用 git status 可以看状态，以及建议的命令

# 【Github教程】实战命令1 – ssh方式

1. ssh-keygen -t rsa -C [mooling@yeah.com](mailto:mooling@yeah.com)
2. cp ~/.ssh/id\_rsa.pub to github
3. ssh -T git@github.com // use to verify
4. git config --global user.name "mooling"
5. git config --global user.email "mooling@yeah.net"
6. git remote add ep  [git@github.com:mooling/emacs-profile.git](mailto:git@github.com:mooling/emacs-profile.git)
7. git push ep master

# [Git Submodule使用完整教程](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

http://www.kafeitu.me/git/2012/03/27/git-submodule.html

自从看了蒋鑫的《[**Git权威指南**](http://book.douban.com/subject/6526452/)》之后就开始使用Git Submodule功能，团队也都熟悉了怎么使用，多个子系统（模块）都能及时更新到最新的公共资源，把使用的过程以及经验和容易遇到的问题分享给大家。

Git Submodule功能刚刚开始学习可能觉得有点怪异，所以本教程把每一步的操作的命令和结果都用代码的形式展现给大家，以便更好的理解。

## 1.对于公共资源各种程序员的处理方式

每个公司的系统都会有一套统一的系统风格，或者针对某一个大客户的多个系统风格保持统一，而且如果风格改动后要同步到多个系统中；这样的需求几乎每个开发人员都遇到，下面看看各个层次的程序员怎么处理：

假如对于系统的风格需要几个目录：css、images、js。

* **普通**程序员，把最新版本的代码逐个复制到每个项目中，如果有N个项目，那就是要复制**N x 3**次；如果漏掉了某个文件夹没有复制…@（&#@#。
* **文艺**程序员，使用Git Submodule功能，执行：**git submodule update**，然后冲一杯咖啡悠哉的享受着。

引用一段《[**Git权威指南**](http://book.douban.com/subject/6526452/)》的话： 项目的版本库在某些情况虾需要引用其他版本库中的文件，例如公司积累了一套常用的函数库，被多个项目调用，显然这个函数库的代码不能直接放到某个项目的代码中，而是要独立为一个代码库，那么其他项目要调用公共函数库该如何处理呢？分别把公共函数库的文件拷贝到各自的项目中会造成冗余，丢弃了公共函数库的维护历史，这显然不是好的方法。

## 2.开始学习Git Submodule

“工欲善其事，必先利其器”！

既然文艺程序员那么轻松就搞定了，那我们就把过程一一道来。

**说明**：本例采用**两个项目**以及**两个公共类库**演示对submodule的操作。因为在一写资料或者书上的例子都是一个项目对应1～N个lib，但是实际应用往往并不是这么简单。

### 2.1 创建Git Submodule测试项目

#### 2.1.1 准备环境

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3 | ➜ henryyan@hy-hp  ~  pwd  /home/henryyan  mkdir -p submd/repos |

创建需要的本地仓库：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | cd ~/submd/repos  git --git-dir=lib1.git init --bare  git --git-dir=lib2.git init --bare  git --git-dir=project1.git init --bare  git --git-dir=project2.git init --bare |

初始化工作区：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2 | mkdir ~/submd/ws  cd ~/submd/ws |

#### 2.1.2 初始化项目

初始化project1：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | ➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/project1.git  Cloning into project1...  done.  warning: You appear to have cloned an empty repository.    ➜ henryyan@hy-hp  ~/submd/ws  cd project1  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) echo "project1" > project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ ls  project-infos.txt    ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git add project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status  # On branch master  #  # Initial commit  #  # Changes to be committed:  #   (use "git rm --cached <file>..." to unstage)  #  #   new file:   project-infos.txt  #  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git commit -m "init project1"  [master (root-commit) 473a2e2] init project1   1 files changed, 1 insertions(+), 0 deletions(-)   create mode 100644 project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git push origin master  Counting objects: 3, done.  Writing objects: 100% (3/3), 232 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/ws/../repos/project1.git   \* [new branch]      master -> master  </file> |

初始化project2：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | ➜ henryyan@hy-hp  ~/submd/ws/project1 cd ..  ➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/project2.git  Cloning into project2...  done.  warning: You appear to have cloned an empty repository.    ➜ henryyan@hy-hp  ~/submd/ws  cd project2  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) echo "project2" > project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ ls  project-infos.txt    ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git add project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status  # On branch master  #  # Initial commit  #  # Changes to be committed:  #   (use "git rm --cached <file>..." to unstage)  #  #   new file:   project-infos.txt  #  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -m "init project2"  [master (root-commit) 473a2e2] init project2   1 files changed, 1 insertions(+), 0 deletions(-)   create mode 100644 project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git push origin master  Counting objects: 3, done.  Writing objects: 100% (3/3), 232 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/ws/../repos/project2.git   \* [new branch]      master -> master  </file> |

#### 2.1.3 初始化公共类库

初始化公共类库lib1：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | ➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/lib1.git  Cloning into lib1...  done.  warning: You appear to have cloned an empty repository.  ➜ henryyan@hy-hp  ~/submd/ws  cd lib1  ➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) echo "I'm lib1." > lib1-features  ➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) ✗ git add lib1-features  ➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) ✗ git commit -m "init lib1"  [master (root-commit) c22aff8] init lib1   1 files changed, 1 insertions(+), 0 deletions(-)   create mode 100644 lib1-features  ➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) git push origin master  Counting objects: 3, done.  Writing objects: 100% (3/3), 227 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/ws/../repos/lib1.git   \* [new branch]      master -> master |

初始化公共类库lib2：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | ➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) cd ..  ➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/lib2.git  Cloning into lib2...  done.  warning: You appear to have cloned an empty repository.  ➜ henryyan@hy-hp  ~/submd/ws  cd lib2  ➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) echo "I'm lib2." > lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) ✗ git add lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) ✗ git commit -m "init lib2"  [master (root-commit) c22aff8] init lib2   1 files changed, 1 insertions(+), 0 deletions(-)   create mode 100644 lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) git push origin master  Counting objects: 3, done.  Writing objects: 100% (3/3), 227 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/ws/../repos/lib2.git   \* [new branch]      master -> master |

### 2.2 为主项目添加Submodules

#### 2.2.1 为project1添加lib1和lib2

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | ➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) cd ../project1  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ls  project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git submodule add ~/submd/repos/lib1.git libs/lib1  Cloning into libs/lib1...  done.  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git submodule add ~/submd/repos/lib2.git libs/lib2  Cloning into libs/lib2...  done.  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ ls  libs  project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ ls libs  lib1  lib2    ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status  # On branch master  # Changes to be committed:  #   (use "git reset HEAD <file>..." to unstage)  #  #   new file:   .gitmodules  #   new file:   libs/lib1  #   new file:   libs/lib2  #    # 查看一下公共类库的内容    ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat libs/lib1/lib1-features  I'm lib1.  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat libs/lib2/lib2-features  I'm lib2.  </file> |

好了，到目前为止我们已经使用**git submodule add**命令为**project1**成功添加了两个公共类库（lib1、lib2），查看了当前的状态发现添加了一个新文件(**.gitmodules**)和两个文件夹(libs/lib1、libs/lib2)；那么新增的**.gitmodules**文件是做什么用的呢？我们查看一下文件内容便知晓了：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | n@hy-hp  ~/submd/ws/project1 git:(master) ✗ cat .gitmodules  [submodule "libs/lib1"]      path = libs/lib1      url = /home/henryyan/submd/repos/lib1.git  [submodule "libs/lib2"]      path = libs/lib2      url = /home/henryyan/submd/repos/lib2.git |

原来如此，**.gitmodules**记录了每个submodule的引用信息，知道在当前项目的位置以及仓库的所在。

好的，我们现在把更改提交到仓库。

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git commit -a -m "add submodules[lib1,lib2] to project1"  [master 7157977] add submodules[lib1,lib2] to project1   3 files changed, 8 insertions(+), 0 deletions(-)   create mode 100644 .gitmodules   create mode 160000 libs/lib1   create mode 160000 libs/lib2    ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (4/4), done.  Writing objects: 100% (4/4), 491 bytes, done.  Total 4 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (4/4), done.  To /home/henryyan/submd/ws/../repos/project1.git     45cbbcb..7157977  master -> master |

假如你是第一次引入公共类库的开发人员，那么项目组的其他成员怎么Clone带有Submodule的项目呢，下面我们再clone一个项目讲解如何操作。

### 2.3 Clone带有Submodule的仓库

模拟开发人员B……

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cd ~/submd/ws  ➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/project1.git project1-b  Cloning into project1-b...  done.  ➜ henryyan@hy-hp  ~/submd/ws  cd project1-b  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git submodule  -c22aff85be91eca442734dcb07115ffe526b13a1 libs/lib1  -7290dce0062bd77df1d83b27dd3fa3f25a836b54 libs/lib2 |

看到submodules的状态是hash码和文件目录，但是注意前面有一个减号：**-**，含义是该子模块还没有检出。

OK，检出project1-b的submodules……

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git submodule init  Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path 'libs/lib1'  Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path 'libs/lib2'  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git submodule update  Cloning into libs/lib1...  done.  Submodule path 'libs/lib1': checked out 'c22aff85be91eca442734dcb07115ffe526b13a1'  Cloning into libs/lib2...  done.  Submodule path 'libs/lib2': checked out '7290dce0062bd77df1d83b27dd3fa3f25a836b54' |

读者可以查看：.git/config文件的内容，最下面有submodule的注册信息！

验证一下类库的文件是否存在：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) cat libs/lib1/lib1-features libs/lib2/lib2-features  I'm lib1.  I'm lib2. |

上面的两个命令(git submodule init & update)其实可以简化，后面会讲到！

### 2.3 修改Submodule

我们在开发人员B的项目上修改Submodule的内容。

先看一下当前Submodule的状态：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) cd libs/lib1  ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  git status  # Not currently on any branch.  nothing to commit (working directory clean) |

为什么是**Not currently on any branch**呢？不是应该默认在**master**分支吗？别急，一一解答！

Git对于Submodule有特殊的处理方式，在一个主项目中引入了Submodule其实Git做了3件事情：

* 记录引用的仓库
* 记录主项目中Submodules的目录位置
* 记录引用Submodule的**commit id**

在**project1**中push之后其实就是更新了引用的commit id，然后project1-b在clone的时候获取到了submodule的commit id，然后当执行**git submodule update**的时候git就根据**gitlink**获取submodule的commit id，最后获取submodule的文件，所以clone之后不在任何分支上；但是master分支的commit id和HEAD保持一致。

查看~/submd/ws/project1-b/libs/lib1的引用信息：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  cat .git/HEAD  c22aff85be91eca442734dcb07115ffe526b13a1  ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  cat .git/refs/heads/master  c22aff85be91eca442734dcb07115ffe526b13a1 |

现在我们要修改lib1的文件需要先切换到**master**分支：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  git checkout master  Switched to branch 'master'  ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) echo "add by developer B" >> lib1-features  ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) ✗ git commit -a -m "update lib1-features by developer B"  [master 36ad12d] update lib1-features by developer B   1 files changed, 1 insertions(+), 0 deletions(-) |

在主项目中修改Submodule提交到仓库稍微繁琐一点，在**git push**之前我们先看看**project1-b**状态：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  </file></file> |

**libs/lib1 (new commits)**状态表示**libs/lib1**有新的提交，这个比较特殊，看看**project1-b**的状态：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git diff  diff --git a/libs/lib1 b/libs/lib1  index c22aff8..36ad12d 160000  --- a/libs/lib1  +++ b/libs/lib1  @@ -1 +1 @@  -Subproject commit c22aff85be91eca442734dcb07115ffe526b13a1  +Subproject commit 36ad12d40d8a41a4a88a64add27bd57cf56c9de2 |

从状态中可以看出**libs/lib1**的commit id由原来的**c22aff85be91eca442734dcb07115ffe526b13a1**更改为**36ad12d40d8a41a4a88a64add27bd57cf56c9de2**

注意：如果现在执行了git submodule update操作那么libs/lib1的commit id又会还原到c22aff85be91eca442734dcb07115ffe526b13a1，

这样的话刚刚的修改是不是就丢死了呢？不会，因为修改已经提交到了master分支，只要再git checkout master就可以了。

现在可以把**libs/lib1**的修改提交到仓库了：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ cd libs/lib1  ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) git push  Counting objects: 5, done.  Writing objects: 100% (3/3), 300 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/repos/lib1.git     c22aff8..36ad12d  master -> master |

现在仅仅只完成了一步，下一步要提交**project1-b**引用submodule的commit id：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) cd ~/submd/ws/project1-b  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git add -u  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git commit -m "update libs/lib1 to lastest commit id"  [master c96838a] update libs/lib1 to lastest commit id   1 files changed, 1 insertions(+), 1 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (3/3), done.  Writing objects: 100% (3/3), 395 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/ws/../repos/project1.git     7157977..c96838a  master -> master |

OK，大功高成，我们完成了Submodule的修改并把**libs/lib1**的最新commit id提交到了仓库。

接下来要看看**project1**怎么获取submodule了。

### 2.4 更新主项目的Submodules

好的，让我们先进入**project1**目录同步仓库：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) cd ../project1  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git pull  remote: Counting objects: 5, done.  remote: Compressing objects: 100% (3/3), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/ws/../repos/project1     7157977..c96838a  master     -> origin/master  Updating 7157977..c96838a  Fast-forward   libs/lib1 |    2 +-   1 files changed, 1 insertions(+), 1 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  </file></file> |

我们运行了**git pull**命令和**git status**获取了最新的仓库源码，然后看到了状态时**modified**，这是为什么呢？

我们用**git diff**比较一下不同：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git diff  diff --git a/libs/lib1 b/libs/lib1  index 36ad12d..c22aff8 160000  --- a/libs/lib1  +++ b/libs/lib1  @@ -1 +1 @@  -Subproject commit 36ad12d40d8a41a4a88a64add27bd57cf56c9de2  +Subproject commit c22aff85be91eca442734dcb07115ffe526b13a1 |

从diff的结果分析出来时因为submodule的commit id更改了，我们前面刚刚讲了要在主项目更新submodule的内容首先要提交submdoule的内容，然后再更新主项目中引用的submodulecommit id；现在我们看到的不同就是因为刚刚更改了project1-b的submodule commit id；好的，我来学习一下怎么更新**project1**的公共类库。

follow me……

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git submodule update  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  </file></file> |

泥马，为什么没有更新？**git submodule update**命令不是更新子模块仓库的吗？

别急，先听我解释；因为子模块是在**project1**中引入的，**git submodule add ~/submd/repos/lib1.git libs/lib1**命令的结果，操作之后git只是把lib1的内容clone到了**project1**中，但是没有在仓库注册，证据如下：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | ➜ henryyan@hy-hp  ~/submd2/ws/project1 git:(master) ✗ cat .git/config  [core]      repositoryformatversion = 0      filemode = true      bare = false      logallrefupdates = true  [remote "origin"]      fetch = +refs/heads/\*:refs/remotes/origin/\*      url = /home/henryyan/submd/ws/../repos/project1.git  [branch "master"]      remote = origin      merge = refs/heads/master |

我们说过**git submodule init**就是在**.git/config**中注册子模块的信息，下面我们试试注册之后再更新子模块：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git submodule init  Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path 'libs/lib1'  Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path 'libs/lib2'  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git submodule update  remote: Counting objects: 5, done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/repos/lib1     c22aff8..36ad12d  master     -> origin/master  Submodule path 'libs/lib1': checked out '36ad12d40d8a41a4a88a64add27bd57cf56c9de2'    ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat .git/config  [core]      repositoryformatversion = 0      filemode = true      bare = false      logallrefupdates = true  [remote "origin"]      fetch = +refs/heads/\*:refs/remotes/origin/\*      url = /home/henryyan/submd/ws/../repos/project1.git  [branch "master"]      remote = origin      merge = refs/heads/master  [submodule "libs/lib1"]      url = /home/henryyan/submd/repos/lib1.git  [submodule "libs/lib2"]      url = /home/henryyan/submd/repos/lib2.git    ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat libs/lib1/lib1-features  I'm lib1.  add by developer B |

上面的结果足以证明刚刚的推断，所以记得当需要更新子模块的内容时请先确保已经运行过**git submodule init**。

### 2.5 为project2添加lib1和lib2

这个操作对于读到这里的你来说应该是**轻车熟路**了，action：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cd ~/submd/ws/project2  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git submodule add ~/submd/repos/lib1.git libs/lib1  Cloning into libs/lib1...  done.  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git submodule add ~/submd/repos/lib2.git libs/lib2  zsh: correct 'libs/lib2' to 'libs/lib1' [nyae]? n  Cloning into libs/lib2...  done.  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ ls  libs  project-infos.txt  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git submodule init  Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path 'libs/lib1'  Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path 'libs/lib2'    ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status  # On branch master  # Changes to be committed:  #   (use "git reset HEAD <file>..." to unstage)  #  #   new file:   .gitmodules  #   new file:   libs/lib1  #   new file:   libs/lib2  #  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -a -m "add lib1 and lib2"  [master 8dc697f] add lib1 and lib2   3 files changed, 8 insertions(+), 0 deletions(-)   create mode 100644 .gitmodules   create mode 160000 libs/lib1   create mode 160000 libs/lib2  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (4/4), done.  Writing objects: 100% (4/4), 471 bytes, done.  Total 4 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (4/4), done.  To /home/henryyan/submd/ws/../repos/project2.git     6e15c68..8dc697f  master -> master    </file> |

我们依次执行了添加submodule并commit和push到仓库，此阶段任务完成。

### 2.6 修改lib1和lib2并同步到project1和project2

假如开发人员C同时负责**project1**和**project2**，有可能在修改**project1**的某个功能的时候发现lib1或者lib2的某个组件有bug需要修复，这个需求多模块和大型系统中经常遇到，我们应该怎么解决呢？

假如我的需求如下：

* 在lib1中添加一个文件：README，用来描述lib1的功能
* 在lib2中的lib2-features文件中添加一写文字：学习Git submodule的修改并同步功能

#### 2.6.1 在lib1中添加一个文件：README

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) cd libs/lib1  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib1 git:(master) echo "lib1 readme contents" > README  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib1 git:(master) ✗ git add README  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib1 git:(master) ✗ git commit -m "add file README"  [master 8c666d8] add file README   1 files changed, 1 insertions(+), 0 deletions(-)   create mode 100644 README  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib1 git:(master) git push  Counting objects: 4, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (2/2), done.  Writing objects: 100% (3/3), 310 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/repos/lib1.git     36ad12d..8c666d8  master -> master |

前面提到过现在仅仅只完成了一部分，我们需要在**project2**中再更新lib1的commit id：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git add libs/lib1  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -m "update lib1 to lastest commit id"  [master ce1f3ba] update lib1 to lastest commit id   1 files changed, 1 insertions(+), 1 deletions(-)  </file></file> |

我们暂时不push到仓库，等待和lib2的修改一起push。

#### 2.6.2 在lib2中的lib2-features文件添加文字

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) cd libs/lib2  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) echo "学习Git submodule的修改并同步功能" >> lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) ✗ git add lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) ✗ git commit -m "添加文字：学习Git submodule的修改并同步功能"  [master e372b21] 添加文字：学习Git submodule的修改并同步功能   1 files changed, 1 insertions(+), 0 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (2/2), done.  Writing objects: 100% (3/3), 376 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/repos/lib2.git     7290dce..e372b21  master -> master    ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) echo "学习Git submodule的修改并同步功能" >> lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) ✗ git add lib2-features  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) ✗ git commit -m "添加文字：学习Git submodule的修改并同步功能"  [master e372b21] 添加文字：学习Git submodule的修改并同步功能   1 files changed, 1 insertions(+), 0 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (2/2), done.  Writing objects: 100% (3/3), 376 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/repos/lib2.git     7290dce..e372b21  master -> master  ➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) cd -  ~/submd/ws/project2  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status  # On branch master  # Your branch is ahead of 'origin/master' by 1 commit.  #  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib2 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git add libs/lib2  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -m "update lib2 to lastest commit id"  [master df344c5] update lib2 to lastest commit id   1 files changed, 1 insertions(+), 1 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git status  # On branch master  # Your branch is ahead of 'origin/master' by 2 commits.  #  nothing to commit (working directory clean)  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git push  Counting objects: 8, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (6/6), done.  Writing objects: 100% (6/6), 776 bytes, done.  Total 6 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (6/6), done.  To /home/henryyan/submd/ws/../repos/project2.git     8dc697f..df344c5  master -> master  </file></file> |

### 2.7 同步project2的lib1和lib2的修改到project1

现在project2已经享受到了最新的代码带来的**快乐**，那么既然project1和project2属于同一个风格，或者调用同一个功能，要让这两个(可能几十个)项目保持一致。

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3 | ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) cd ../project1  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git pull  Already up-to-date. |

看看上面的结果对吗？为什么lib1和lib2更新了但是没有显示**new commits**呢？说到这里我记得刚刚开始学习的时候真得要晕死了，Git跟我玩捉迷藏游戏，为什么我明明提交了但是从**project1**更新不到任何改动呢？

帮大家分析一下问题，不过在分析之前先看看当前(project1和project2)的submodule状态：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # project2 的状态，也就是我们刚刚修改后的状态  ➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git submodule   8c666d86531513dd1aebdf235f142adbac72c035 libs/lib1 (heads/master)   e372b21dffa611802c282278ec916b5418acebc2 libs/lib2 (heads/master)    # project1 的状态，等待更新submodules  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git submodule   36ad12d40d8a41a4a88a64add27bd57cf56c9de2 libs/lib1 (remotes/origin/HEAD)   7290dce0062bd77df1d83b27dd3fa3f25a836b54 libs/lib2 (heads/master) |

两个项目有两个区别：

* commit id各不相同
* **libs/lib1**所处的分支不同

#### 2.7.1 更新project1的lib1和lib2改动

我们还记得刚刚在**project2**中修改的时候把lib1和lib2都切换到了master分支，目前project1中的lib1不在任何分支，我们先切换到master分支：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cd libs/lib1  ➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1  git checkout master  Previous HEAD position was 36ad12d... update lib1-features by developer B  Switched to branch 'master'  Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.  ➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1 git:(master) git pull  remote: Counting objects: 4, done.  remote: Compressing objects: 100% (2/2), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/repos/lib1     36ad12d..8c666d8  master     -> origin/master  Updating c22aff8..8c666d8  Fast-forward   README        |    1 +   lib1-features |    1 +   2 files changed, 2 insertions(+), 0 deletions(-)   create mode 100644 README  ➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1 git:(master) |

果不其然，我们看到了刚刚在project2中修改的内容，同步到了project1中，当然现在更新了**project1**的**lib1**，commit id也会随之变动：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | ➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1 git:(master) cd ../../  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git diff  diff --git a/libs/lib1 b/libs/lib1  index 36ad12d..8c666d8 160000  --- a/libs/lib1  +++ b/libs/lib1  @@ -1 +1 @@  -Subproject commit 36ad12d40d8a41a4a88a64add27bd57cf56c9de2  +Subproject commit 8c666d86531513dd1aebdf235f142adbac72c035  </file></file> |

现在最新的commit id和project2目前的状态一致，说明真的同步了；好的，现在可以使用相同的办法更新**lib2**了：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ cd libs/lib2  ➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib2 git:(master) git pull  remote: Counting objects: 5, done.  remote: Compressing objects: 100% (2/2), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/repos/lib2     7290dce..e372b21  master     -> origin/master  Updating 7290dce..e372b21  Fast-forward   lib2-features |    1 +   1 files changed, 1 insertions(+), 0 deletions(-) |

#### 2.7.2 更新project1的submodule引用

在**2.7.1**中我们更新了**project1**的**lib1**和**lib2**的最新版本，现在要把最新的commit id保存到**project1**中以保持最新的引用。

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #   modified:   libs/lib2 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git commit -a -m "update lib1 and lib2 commit id to new version"  [master 8fcca50] update lib1 and lib2 commit id to new version   2 files changed, 2 insertions(+), 2 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (3/3), done.  Writing objects: 100% (3/3), 397 bytes, done.  Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  To /home/henryyan/submd/ws/../repos/project1.git     c96838a..8fcca50  master -> master  </file></file> |

### 2.8 更新project1-b项目的子模块(使用脚本)

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git pull  remote: Counting objects: 5, done.  remote: Compressing objects: 100% (3/3), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/ws/../repos/project1     c96838a..8fcca50  master     -> origin/master  Updating c96838a..8fcca50  Fast-forward   libs/lib1 |    2 +-   libs/lib2 |    2 +-   2 files changed, 2 insertions(+), 2 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git status  # On branch master  # Changes not staged for commit:  #   (use "git add <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   modified:   libs/lib1 (new commits)  #   modified:   libs/lib2 (new commits)  #  no changes added to commit (use "git add" and/or "git commit -a")  </file></file> |

Git提示lib1和lib2有更新内容，这个判断的依据来源于submodule commit id的引用。

现在怎么更新呢？难道还是像project1中那样进入子模块的目录然后**git checkout master**，接着**git pull**？

而且现在仅仅才两个子模块、两个项目，如果在真实的项目中使用的话可能几个到几十个不等，再加上N个submodule，自己算一下要怎么更新多少个submodules？

例如笔者现在做的一个项目有**5**个web模块，每个web模块引用公共的css、js、images、jsp资源，这样就有20个submodule需要更新！！！

**工欲善其事，必先利其器**，写一个脚本代替手动任务。

#### 2.8.1 牛刀小试

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ grep path .gitmodules | awk '{ print $3 }' > /tmp/study-git-submodule-dirs  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ cat /tmp/study-git-submodule-dirs   libs/lib1   libs/lib2 |

我们通过分析**.gitmodules**文件得出子模块的路径，然后就可以根据这些路径进行更新。

#### 2.8.2 上路

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ mkdir bin  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ vi bin/update-submodules.sh |

把下面的脚本复制到**bin/update-submodules.sh**中：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #!/bin/bash  grep path .gitmodules | awk '{ print $3 }' > /tmp/study-git-submodule-dirs    # read  while read LINE  do      echo $LINE      (cd ./$LINE && git checkout master && git pull)  done < /tmp/study-git-submodule-dirs |

稍微解释一下上面的脚本执行过程：

* 首先把子模块的路径写入到文件**/tmp/study-git-submodule-dirs**中；
* 然后读取文件中的子模块路径，依次切换到master分支(修改都是在master分支上进行的)，最后更新最近改动。

#### 2.8.3 2013-01-19更新

网友**@紫煌**给出了更好的办法，一个命令就可以代替上面的**bin/update-submodules.sh**的功能：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1 | git submodule foreach git pull |

此命令也脚本一样，循环进入（enter）每个子模块的目录，然后执行**foreach**后面的命令。

该后面的命令可以任意的，例如 git submodule foreach ls -l 可以列出每个子模块的文件列表

#### 2.8.3 体验工具带来的便利

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git submodule  +36ad12d40d8a41a4a88a64add27bd57cf56c9de2 libs/lib1 (heads/master)  +7290dce0062bd77df1d83b27dd3fa3f25a836b54 libs/lib2 (heads/master)    # 添加执行权限  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ chmod +x ./bin/update-submodules.sh  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ ./bin/update-submodules.sh  libs/lib1  Already on 'master'  remote: Counting objects: 4, done.  remote: Compressing objects: 100% (2/2), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/repos/lib1     36ad12d..8c666d8  master     -> origin/master  Updating 36ad12d..8c666d8  Fast-forward   README |    1 +   1 files changed, 1 insertions(+), 0 deletions(-)   create mode 100644 README  libs/lib2  Switched to branch 'master'  remote: Counting objects: 5, done.  remote: Compressing objects: 100% (2/2), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From /home/henryyan/submd/repos/lib2     7290dce..e372b21  master     -> origin/master  Updating 7290dce..e372b21  Fast-forward   lib2-features |    1 +   1 files changed, 1 insertions(+), 0 deletions(-)  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git submodule   8c666d86531513dd1aebdf235f142adbac72c035 libs/lib1 (heads/master)   e372b21dffa611802c282278ec916b5418acebc2 libs/lib2 (heads/master)     ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git status  # On branch master  # Untracked files:  #   (use "git add <file>..." to include in what will be committed)  #  #   bin/  nothing added to commit but untracked files present (use "git add" to track)  </file> |

更新之后的两个变化：

* git submodule的结果和project2的submodule commit id保持一致；
* **project1-b**不再提示**new commits**了。

现在可以把工具添加到仓库了，当然你可以很骄傲的分享给其他项目组的同事。

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git add bin/update-submodules.sh  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git commit -m "添加自动更新submodule的快捷脚本^\_^"  [master 756e788] 添加自动更新submodule的快捷脚本^\_^   1 files changed, 9 insertions(+), 0 deletions(-)   create mode 100755 bin/update-submodules.sh  ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git push  Counting objects: 5, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (3/3), done.  Writing objects: 100% (4/4), 625 bytes, done.  Total 4 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (4/4), done.  To /home/henryyan/submd/ws/../repos/project1.git     8fcca50..756e788  master -> master |

### 2.9 新进员工加入团队，一次性Clone项目和Submodules

一般人使用的时候都是使用如下命令：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3 | git clone /path/to/repos/foo.git  git submodule init  git submodule update |

新员工不耐烦了，嘴上不说但是心里想：怎么那么麻烦？

上面的命令简直弱暴了，直接一行命令搞定：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1 | git clone --recursive /path/to/repos/foo.git |

–**recursive**参数的含义：可以在clone项目时同时clone关联的submodules。

git help 对其解释：

--recursive, --recurse-submodules

After the clone is created, initialize all submodules within, using their default settings. This is equivalent to running git

submodule update --init --recursive immediately after the clone is finished. This option is ignored if the cloned repository

does not have a worktree/checkout (i.e. if any of --no-checkout/-n, --bare, or --mirror is given)

#### 2.9.1 使用一键方式克隆project2

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ➜ henryyan@hy-hp  ~/submd/ws  git clone --recursive ../repos/project2.git project2-auto-clone-submodules  Cloning into project2-auto-clone-submodules...  done.  Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path 'libs/lib1'  Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path 'libs/lib2'  Cloning into libs/lib1...  done.  Submodule path 'libs/lib1': checked out '8c666d86531513dd1aebdf235f142adbac72c035'  Cloning into libs/lib2...  done.  Submodule path 'libs/lib2': checked out 'e372b21dffa611802c282278ec916b5418acebc2' |

舒服……

## 3.移除Submodule

**牢骚**：搞不明白为什么git不设计一个类似：git submodule remove的命令呢？

我们从project1.git克隆一个项目用来练习移除submodule：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ➜ henryyan@hy-hp  ~/submd/ws  git clone --recursive ../repos/project1.git project1-remove-submodules  Cloning into project1-remove-submodules...  done.  Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path 'libs/lib1'  Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path 'libs/lib2'  Cloning into libs/lib1...  done.  Submodule path 'libs/lib1': checked out '8c666d86531513dd1aebdf235f142adbac72c035'  Cloning into libs/lib2...  done.  Submodule path 'libs/lib2': checked out 'e372b21dffa611802c282278ec916b5418acebc2'  ➜ henryyan@hy-hp  ~/submd/ws  cd !$  ➜ henryyan@hy-hp  ~/submd/ws  cd project1-remove-submodules |

### 3.1 Step by

1、删除git cache和物理文件夹

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4 | ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) git rm -r --cached libs/  rm 'libs/lib1'  rm 'libs/lib2'  ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ rm -rf libs |

2、删除.gitmodules的内容（或者整个文件） 因为本例只有两个子模块，直接删除文件：

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1 | ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ rm .gitmodules |

如果仅仅删除某一个submodule那么打开.gitmodules文件编辑，删除对应submodule配置即可。

3、删除.git/config的submodule配置 源文件：

[core]

repositoryformatversion = 0

filemode = true

bare = false

logallrefupdates = true

[remote "origin"]

fetch = +refs/heads/\*:refs/remotes/origin/\*

url = /home/henryyan/submd/ws/../repos/project1.git

[branch "master"]

remote = origin

merge = refs/heads/master

[submodule "libs/lib1"]

url = /home/henryyan/submd/repos/lib1.git

[submodule "libs/lib2"]

url = /home/henryyan/submd/repos/lib2.git

删除后：

[core]

repositoryformatversion = 0

filemode = true

bare = false

logallrefupdates = true

[remote "origin"]

fetch = +refs/heads/\*:refs/remotes/origin/\*

url = /home/henryyan/submd/ws/../repos/project1.git

[branch "master"]

remote = origin

merge = refs/heads/master

4、提交更改

[?](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ git status  # On branch master  # Changes to be committed:  #   (use "git reset HEAD <file>..." to unstage)  #  #   deleted:    libs/lib1  #   deleted:    libs/lib2  #  # Changes not staged for commit:  #   (use "git add/rm <file>..." to update what will be committed)  #   (use "git checkout -- <file>..." to discard changes in working directory)  #  #   deleted:    .gitmodules  #  ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ git add .gitmodules  ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ git commit -m "删除子模块lib1和lib2"  [master 5e2ee71] 删除子模块lib1和lib2   3 files changed, 0 insertions(+), 8 deletions(-)   delete mode 100644 .gitmodules   delete mode 160000 libs/lib1   delete mode 160000 libs/lib2  ➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) git push  Counting objects: 3, done.  Delta compression using up to 2 threads.  Compressing objects: 100% (2/2), done.  Writing objects: 100% (2/2), 302 bytes, done.  Total 2 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (2/2), done.  To /home/henryyan/submd/ws/../repos/project1.git     756e788..5e2ee71  master -> master  </file></file></file> |

## 4.结束语

对于Git Submodule来说在刚刚接触的时候多少会有点凌乱的赶紧，尤其是没有接触过**svn:externals**。

本文从开始创建项目到使用git submodule的每个步骤以及细节、原理逐一讲解，希望到此读者能驾驭它。

学会了Git submdoule你的项目中再也不会出现大量重复的资源文件、公共类库，更不会出现多个版本，甚至一个客户的多个项目风格存在各种差异。

本文的写作来源于工作中的实践，如果您对于某个做法有更好的办法还请赐教，希望能留下您宝贵的意见。

原创文章，转载请注明出处！ [***Git Submoudle使用完整教程--咖啡兔***](http://www.kafeitu.me/git/2012/03/27/git-submodule.html)