# 易混知识点

## git remote

为了便于管理，Git要求每个远程主机都必须指定一个主机名。

**git remote 命令就用于管理主机名。**

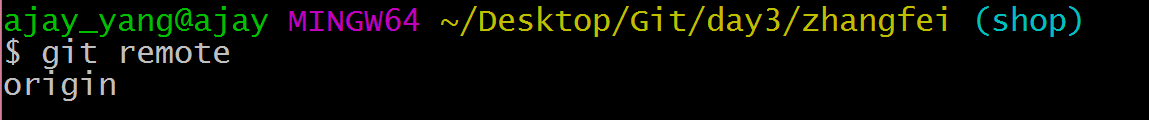
### 1)添加远程主机名

**git remote add <主机名> <网址>**

一般的格式是：git remote add 短名 长网址 //---用短名来代替远程网址

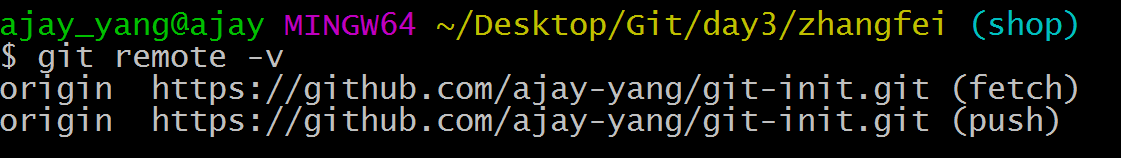
### 2)列出所有远程主机名

**git remote**



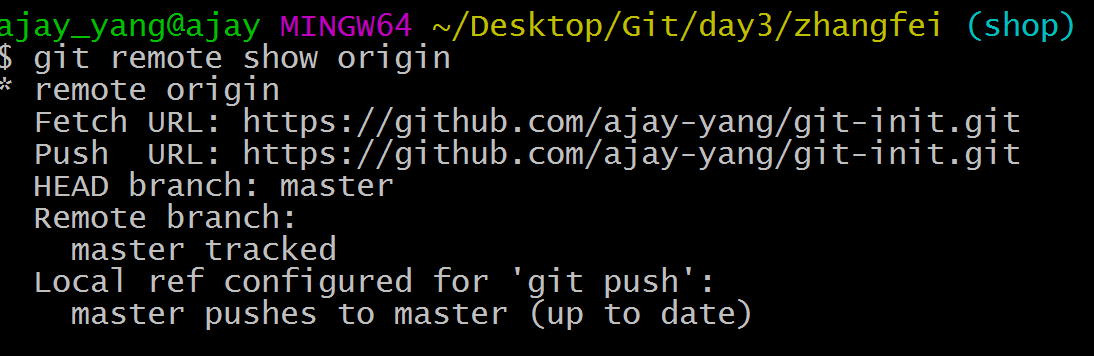
### 3)查看远程主机网址

**git remote -v**



### 4）查看主机详细信息

**git remote show <主机名>**



### 5）删除远程主机名

**git remote rm <主机名>**

### 6）远程主机改名

**git remote rename <主机名>**

## git push

用于将本地分支的更新，推送到远程主机。

### 本地的master分支推送到origin主机的master分支。如果后者不存在，则会被新建。

**git push origin master**

### 将本地的master分支推送到origin主机，同时指定origin为默认主机，后面就可以不加任何参数使用git push了。

**git push -u origin master**

### 删除远程分支

**git push origin :<分支名>**

### 本地新建分支添加到远程

**git push origin 本地分支：远程分支**

**git branch <分支名>**

**git push origin user //---远程添加user分支**

### 本地文件上传到远程指定分支

**git push <远程主机名> <本地分支名>:<远程分支名>**

**git branch user**

**git checkout user**

**git push origin user**

### 删除远程文件及文件夹

删除文件夹

**git rm –r –-cache 文件夹**

**git commit –m ‘删除文件夹’**

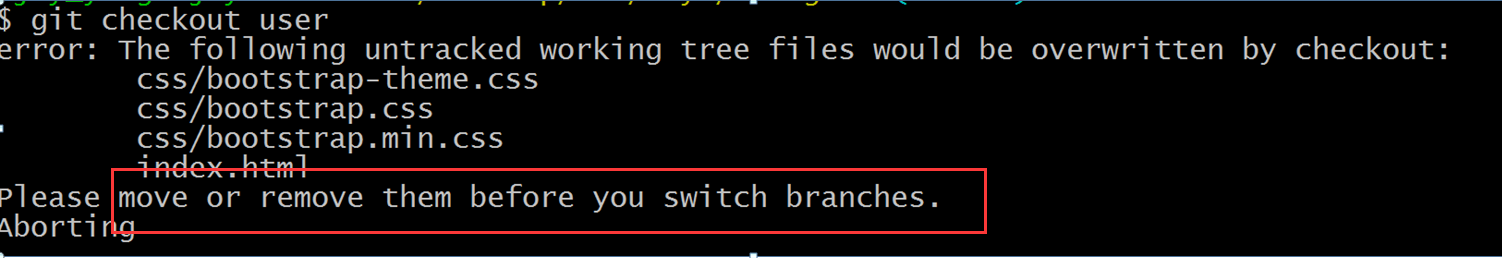
**git push origin <分支名>**

删除文件

**git rm –rf –-cache 文件夹/文件名**

**git commit –m ‘删除文件’**

**git push origin <分支名>**



删除本地未被的跟踪的文件

**git clean -df**

## git pull

取回远程主机某个分支的更新，再与本地的指定分支合并。

一般加到push之前。或者项目提交之前

**git pull <远程主机名> <远程分支名>:<本地分支名>**

取回origin/next分支，再与当前分支合并

**git pull origin next:master**

提交流程：

**git init**

**git status**

**git add .**

**git commit –m ‘备注信息’**

**git remote add origin http://**

**git pull origin master（有可能有更新，也可能没有）**

**//---合并融合**

**//---出现冲突，自己修改**

**git push origin master（没有更新直接push）**

实质上，这等同于:

### git fetch origin

**git fetch origin //---先从远程取下来**

**git merge origin/next //---再融合合并**

在某些场合，git会自动在本地分支与远程分支之间，建立一种追踪关系（tracking）。

比如，在git clone的时候，所有本地分支默认与远程主机的同名分支，建立追踪关系，也就是说，本地的master分支自动"追踪"origin/master分支,当前分支自动与唯一一个追踪分支进行合并。

### get pull --rebase

**git pull --rebase <远程主机名> <远程分支名>:<本地分支名>**

|  |  |
| --- | --- |
|  |  |

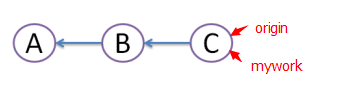
两者的区别可以用下面的关系式来表示：

**git pull = git fetch + git merge**

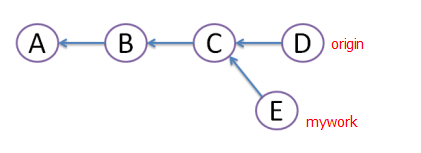
**git pull --rebase = git fetch + git rebase**

git merge和git rebase的区别 :

假设有3次提交A,B,C :

[](http://photo.blog.sina.com.cn/showpic.html#blogid=88243b5e0101j38v&url=http://album.sina.com.cn/pic/002uzKtozy6Hr7sqNqO92)

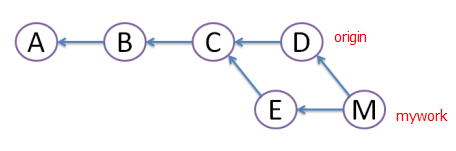
在远程分支origin的基础上创建一个名为"mywork"的分支并提交了，同时有其他人在"origin"上做了一些修改并提交了。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=88243b5e0101j38v&url=http://album.sina.com.cn/pic/002uzKtozy6Hr75R7hA66)

其实这个时候E不应该提交，因为提交后会发生冲突。

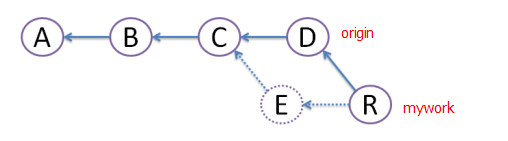
1、git merge

用git pull命令把"origin"分支上的修改pull下来与本地提交合并（merge）成版本M，

[](http://photo.blog.sina.com.cn/showpic.html#blogid=88243b5e0101j38v&url=http://album.sina.com.cn/pic/002uzKtozy6Hr75Sxx486)

2、git rebase

创建一个新的提交R，R的文件内容和上面M的一样，但我们将E提交废除，当它不存在（图中用虚线表示）,保持提交曲线为直线，

[](http://photo.blog.sina.com.cn/showpic.html#blogid=88243b5e0101j38v&url=http://album.sina.com.cn/pic/002uzKtozy6Hr75Vuvif4)

解决完冲突

**git add.**

**git rebase –continue //---继续apply余下的补丁。**

**git rebase –abort //---终止rebase的行动,回到之前的状态**

唐僧，猴子开发项目，远程共享。

唐僧负责master分支，猴子负责shop分支。

### 项目初始化设置

**git init**

**git status**

**git add .**

**git commit –m ‘初始化设置’**

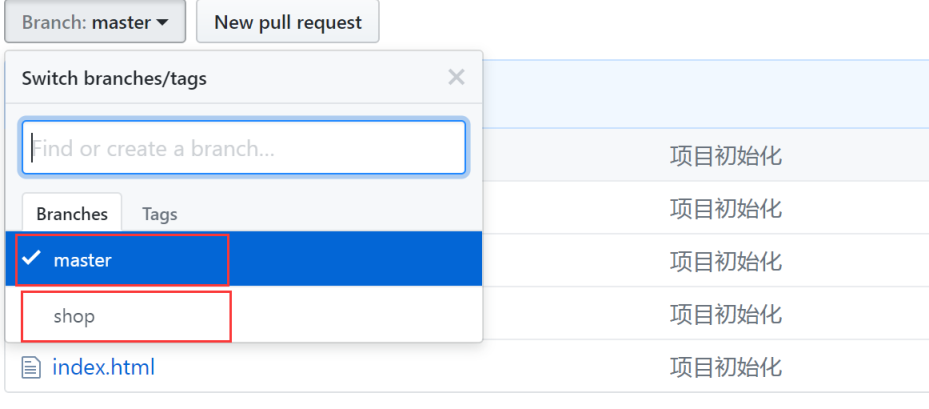
**git remote add origin http://**

**git push origin master**

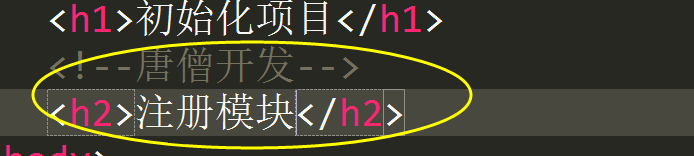
### 新建shop分支,用于后猴子开发。

**git branch shop**

**git push origin shop**



### 唐僧开发master，并上传。



**git status**

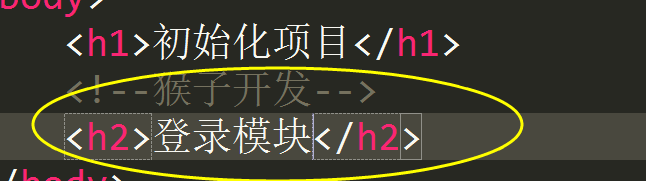
**git add .**

**git commit –m ‘唐僧—注册模块’**

**git push origin master**

### 

### 4）猴子与唐僧一同开发master下的同一个文件



**git init**

**git remote add origin http://**

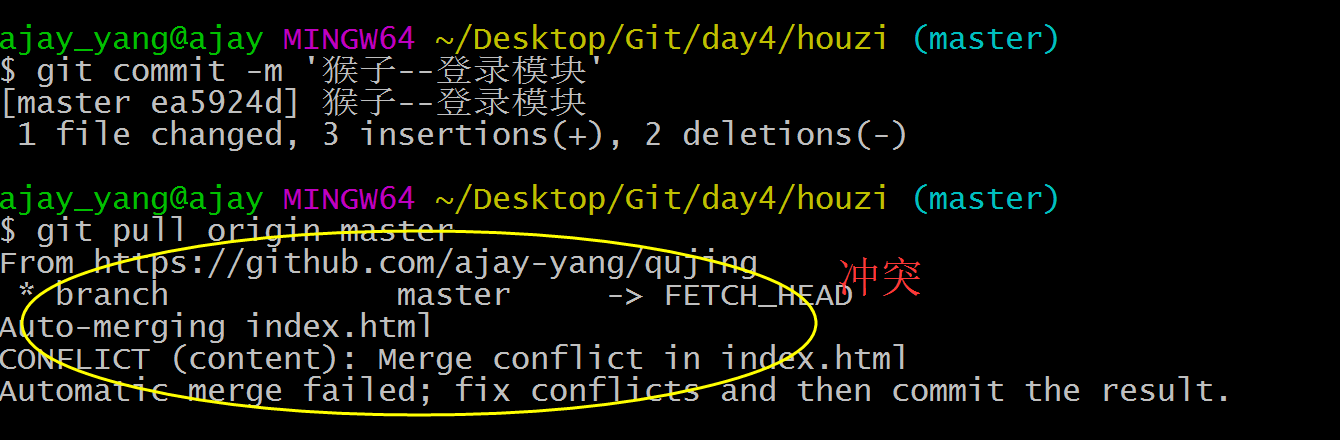
**git status**

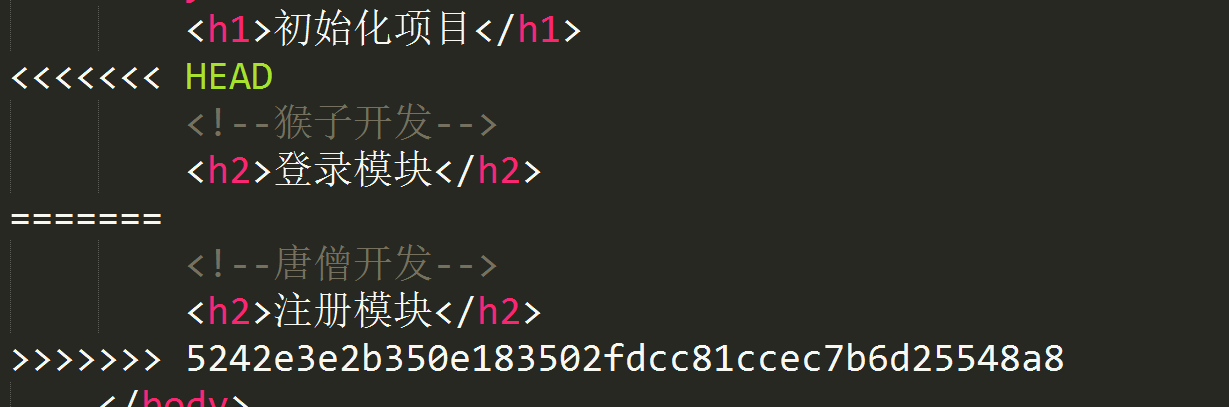
**git add .**

**git commit –m ‘猴子—登录模块’**

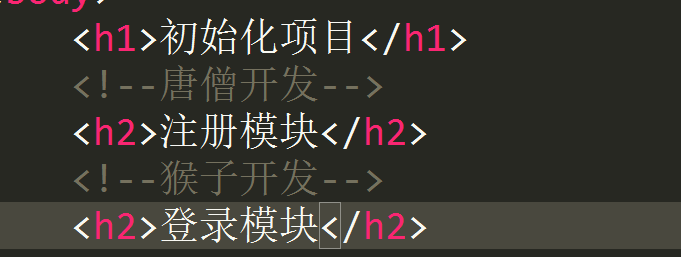
**git pull origin master**

出现冲突。





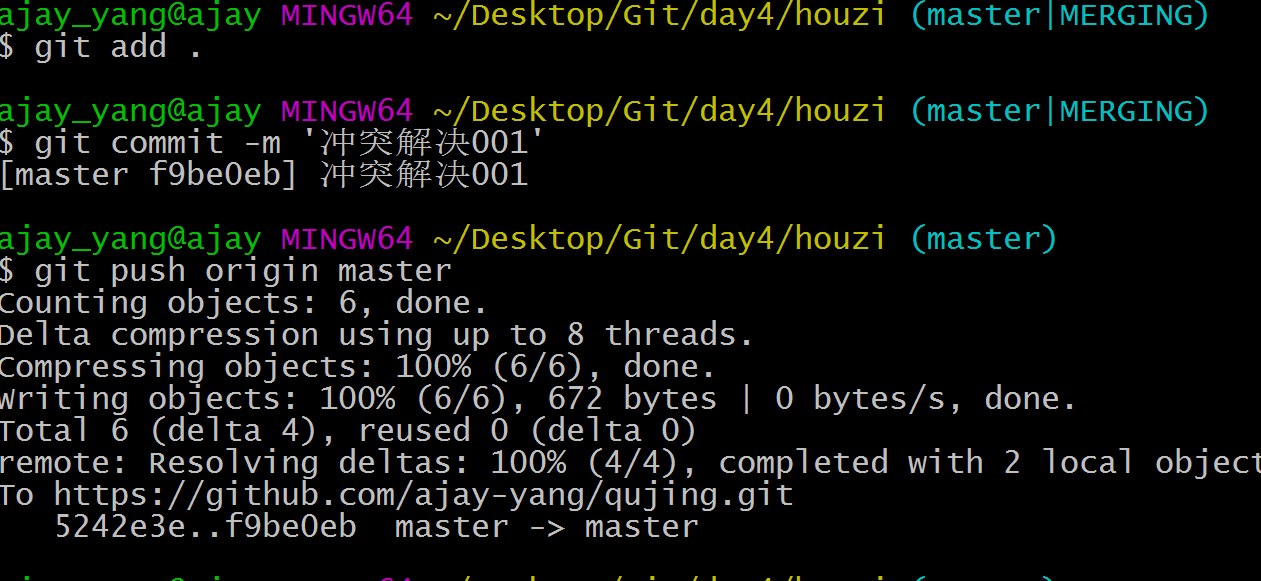
修改冲突：



**git add .**

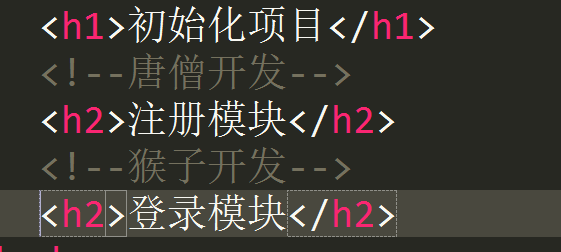
**git commit –m ‘冲突解决’**

**git push origin master**



如果唐僧再次想开发，先从远程仓拉下代码，在继续开发。

**git pull origin master**

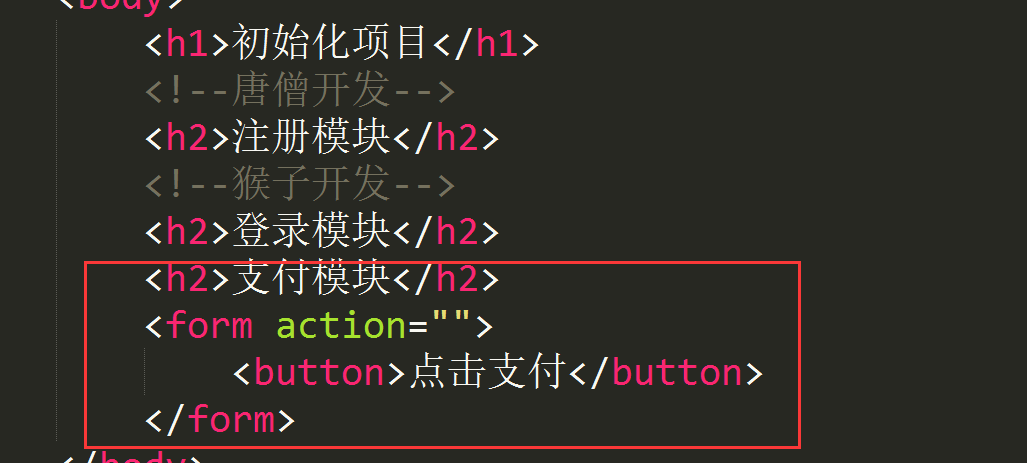


### 猴子去shop分支支付功能。

**git branch shop**

**git push origin shop**

**git checkout shop**



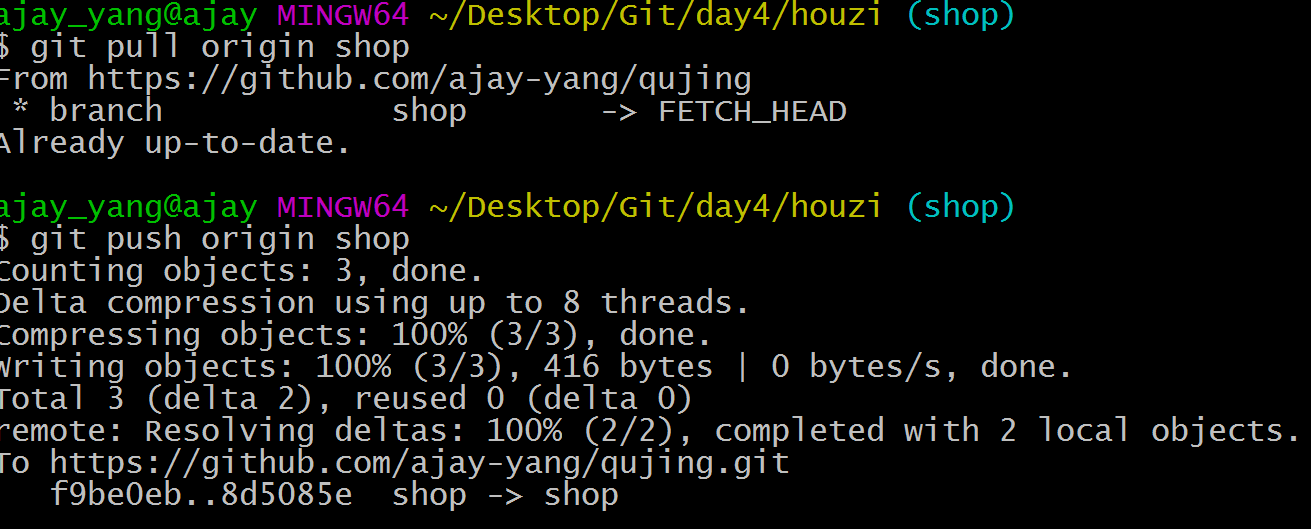
**git status**

**git add .**

**git commit –m ‘支付功能—猴子’**

**git pull origin shop**

**git push origin shop**



### 项目结束要合并分支，并删除远程分支

