上一篇记录了Git仓库的一些基本操作，这篇会说到Git的王牌，往下看：

**一.远程仓库**

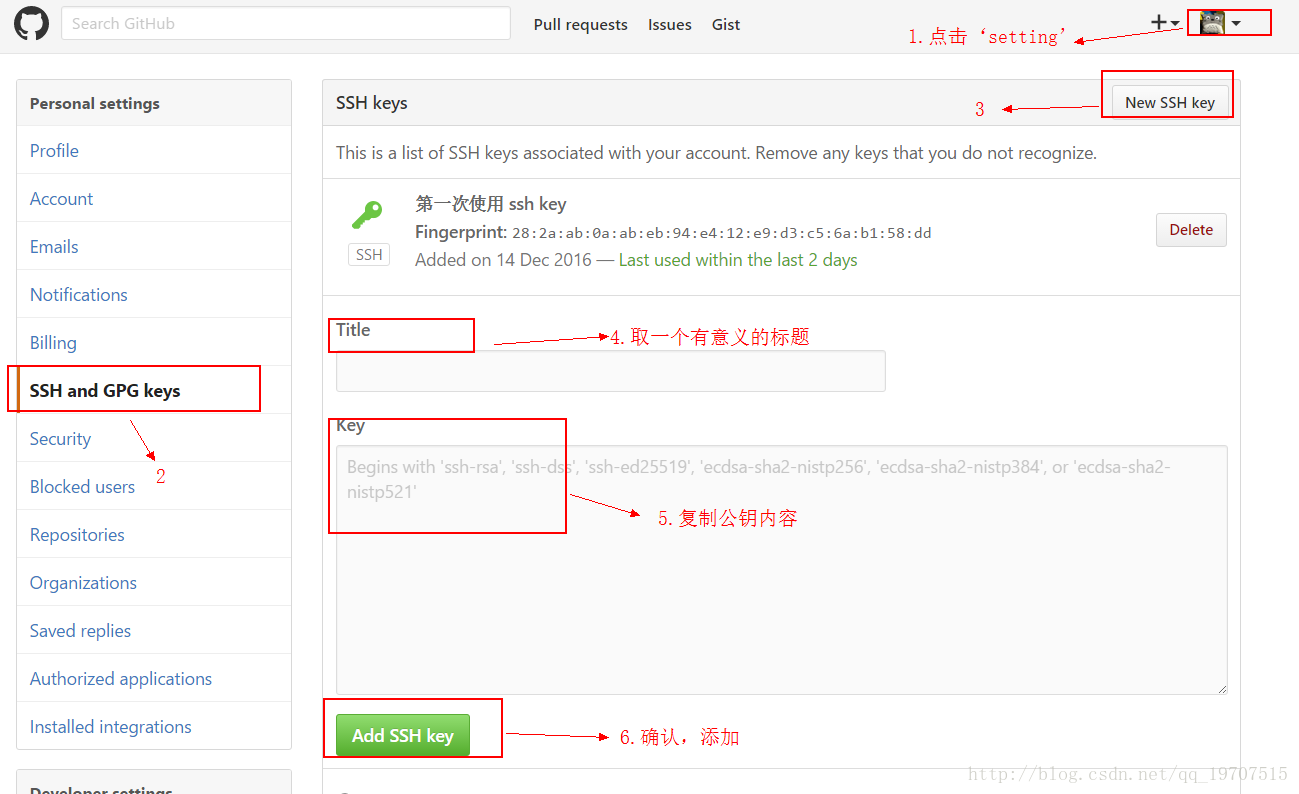
先注册一个github帐号，由于你的本地Git仓库和github仓库之间的传输是通过SSH加密的，所以，先行设置：

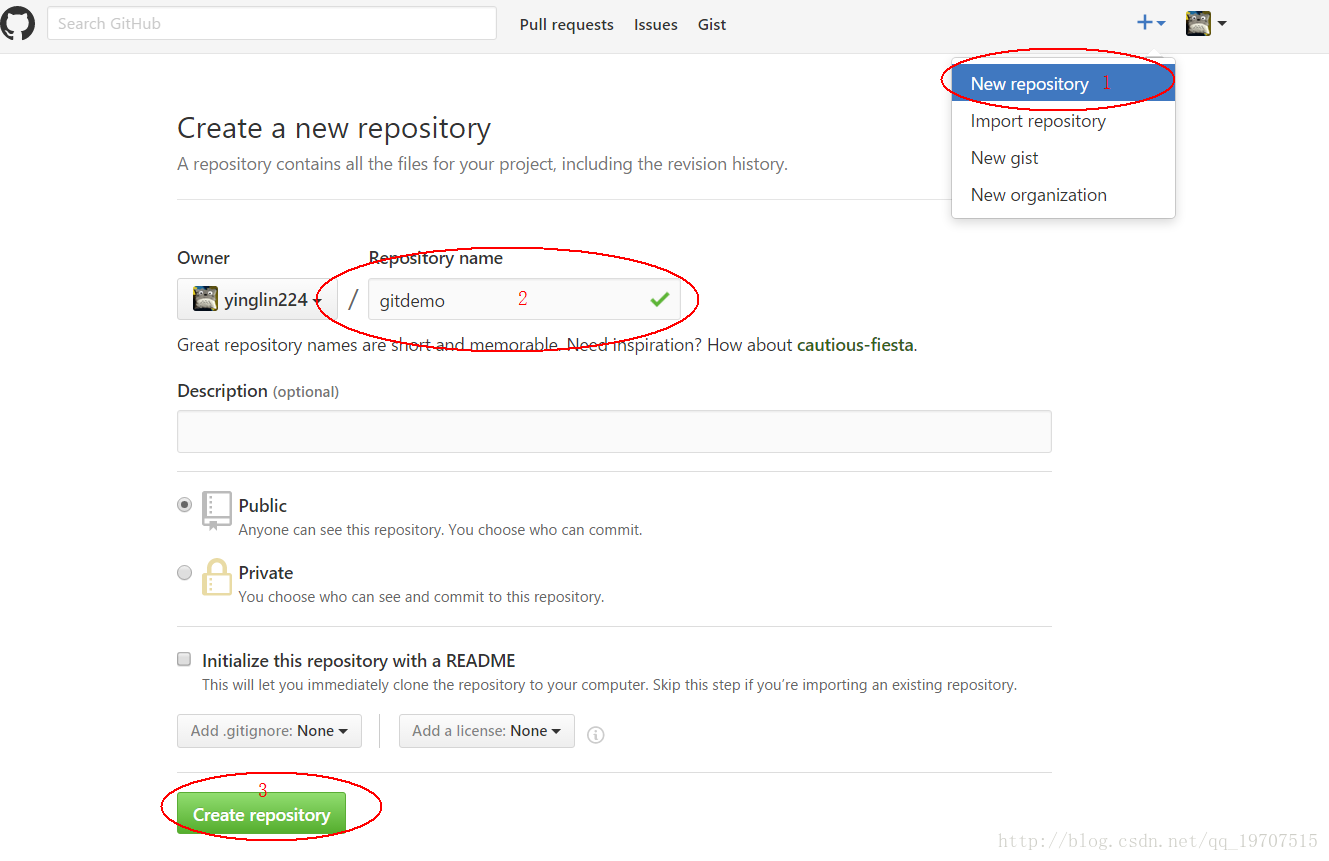
1.创建SHH key。在用户主目录下，查看有没有.ssh目录，如果有，看有没有id\_rsa和id\_rsa.pub,如果有，跳过这步，没有的话打开Git Bash Here ,键入如下命令

ssh-keygen -t rsa -C 'eamil@126.com',如下

* 1

  
id\_rsa是私钥，id\_rsa.pub是公钥（可以公布）

2.登录Github，打开右上角‘settings’,点击‘SSH and GPG keys’,然后点击‘New SSH key ‘，在key中录入公钥，title备注，然后add即可   


3.添加远程库   
现在，想让本地库和Github库可以进行同步，操作如下：   


创建新的远程仓库名：gitdemo，新的Git仓库如下：



现在gitdemo仓库是空的，根据Github提示，我们可以将本地库与之关联或者将这个仓库clone出来

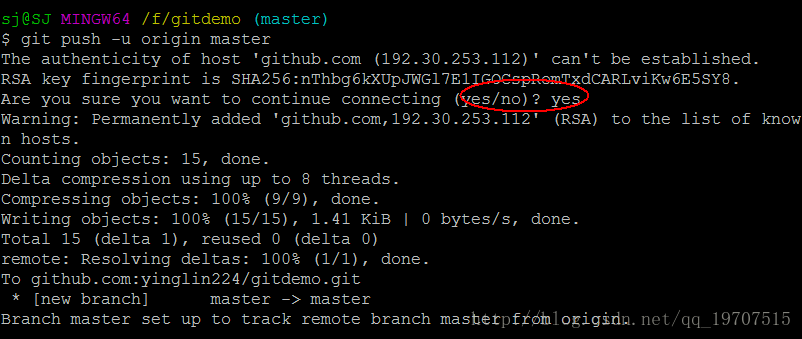
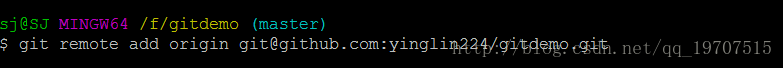
--将本地库关联远程库

git remote add origin git@github.com:yinglin224/gitdemo.git

--将本地库的内容推送到远程库上

git push -u origin master

* 1
* 2
* 3
* 4

由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。   
推送完后，可以看到远程库和本地库内容一致了。   
以后推送使用命令：git push origin master   


同理，可从github上clone到本地，命令：git clone address(协议ssh/https都可)

|  |
| --- |
| 注：Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。 |

**二.创建与合并分支**

--创建并切换

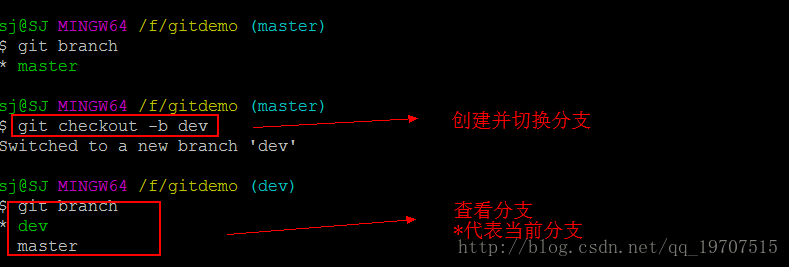
git checkout -b dev

--等同于先创建，再切换

git branch dev

git checkout dev

* 1
* 2
* 3
* 4
* 5

   
下面做具体演示：   
在dev上对readme.txt 添加一行：dev add ,并提交，然后切换分支到master，由于dev的内容没有合并到master，所以看不到文件更改，需要使用命令git merge来同步   


|  |
| --- |
| 注：1 .get merge 用于合并指定分支到当前分支        2.上面有 Fast-forward，指的这次合并是‘快进模式’ |

合并完成后，可以删除dev分支了,常用的几个命令如下：

--查看分支：

git branch

--创建分支：

git branch name

--切换分支：

git checkout name

--创建+切换分支：

git checkout –b name

--合并某分支到当前分支：

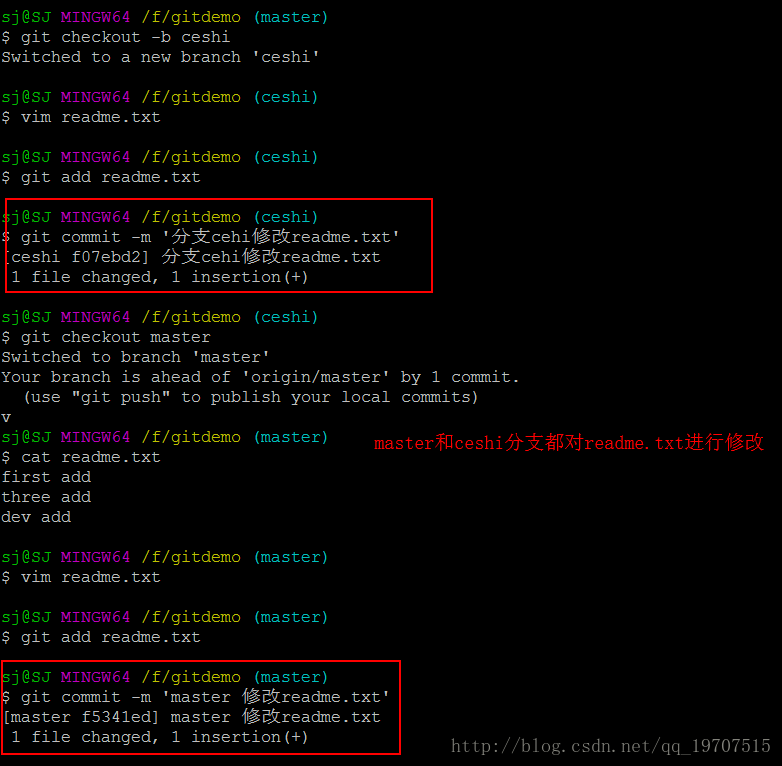
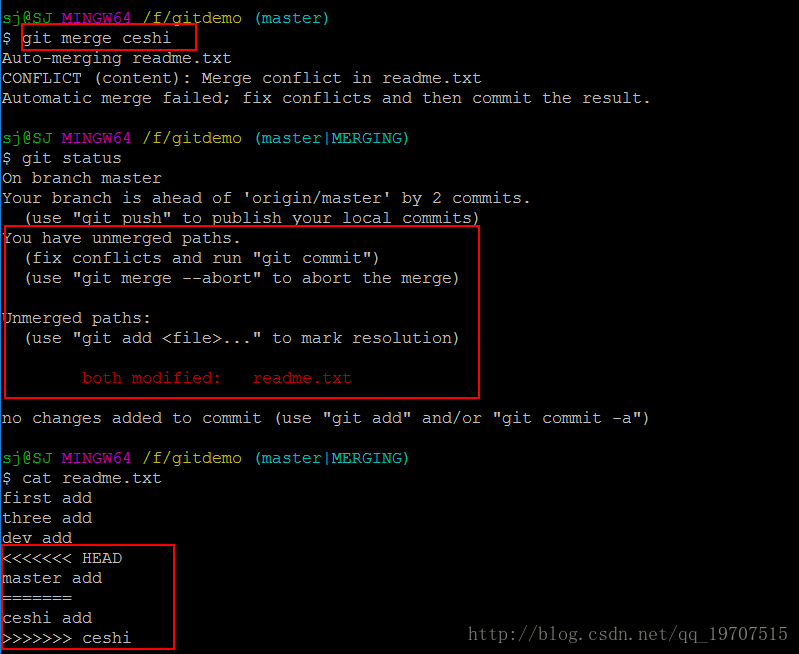
git merge name

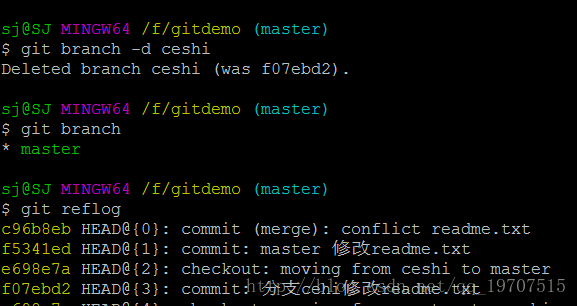
--删除分支：

git branch –d name

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12

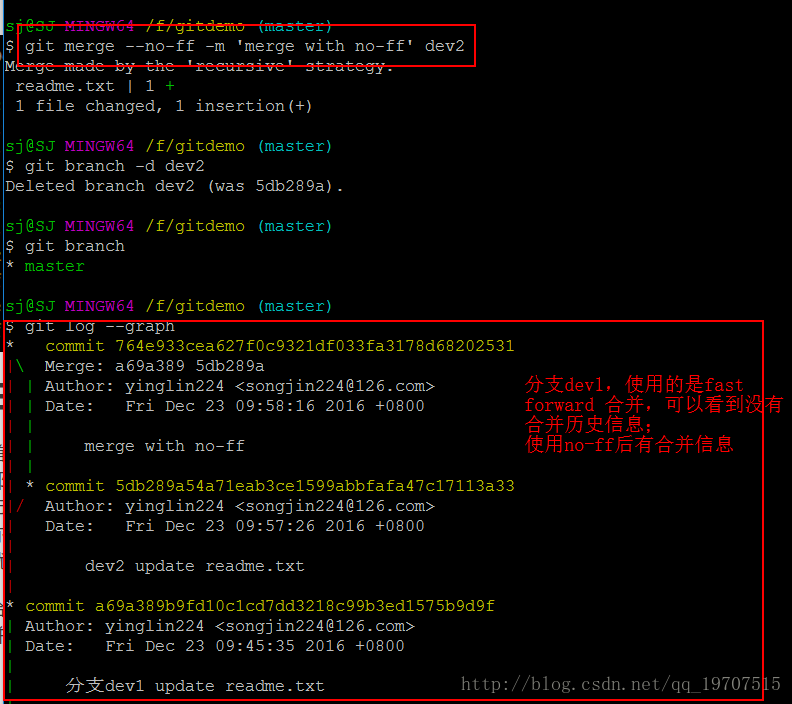
**三.解决冲突**

一般提交的时候，可能会分支和分支之前都会修改相同的文件，造成冲突，下面演示下：   
创建一个分支：ceshi   
   


合并的时候会发现报错，查看文件冲突代码<<<<<，=====， >>>>>，其中HEAD指主分支修改内容，>>>>>ceshi指分支上修改的内容，我们对这个文件手动进行编辑后保存提交即可，然后删除分支ceshi   


**四.分支管理策略**

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；   
那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；   
你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

合并分支时，加上–no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。   


更详细点的，请参考[分支管理策略](http://www.ruanyifeng.com/blog/2012/07/git.html" \t "_blank)

**五.bug分支**

当我们在开发过程中，难免有突然的bug需要去解决，咋办呢，手头上的活没干完，又必须得去解决上产的紧急bug；演示下：

--手头上有活没干完，去处理其他bug时，使用命令可以将工作现场储存

git stash

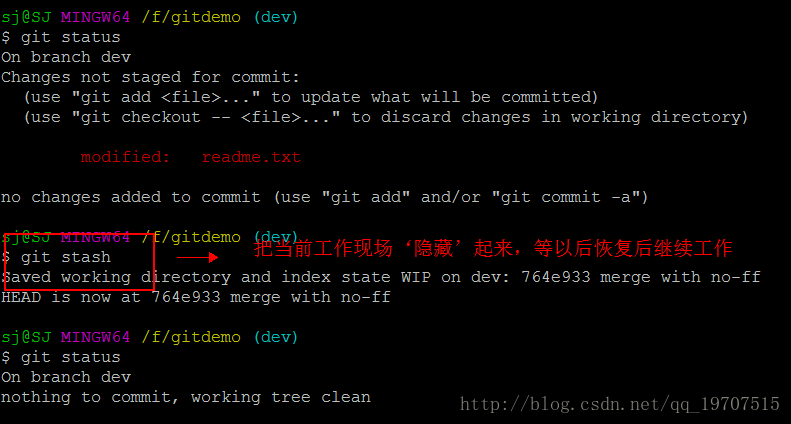
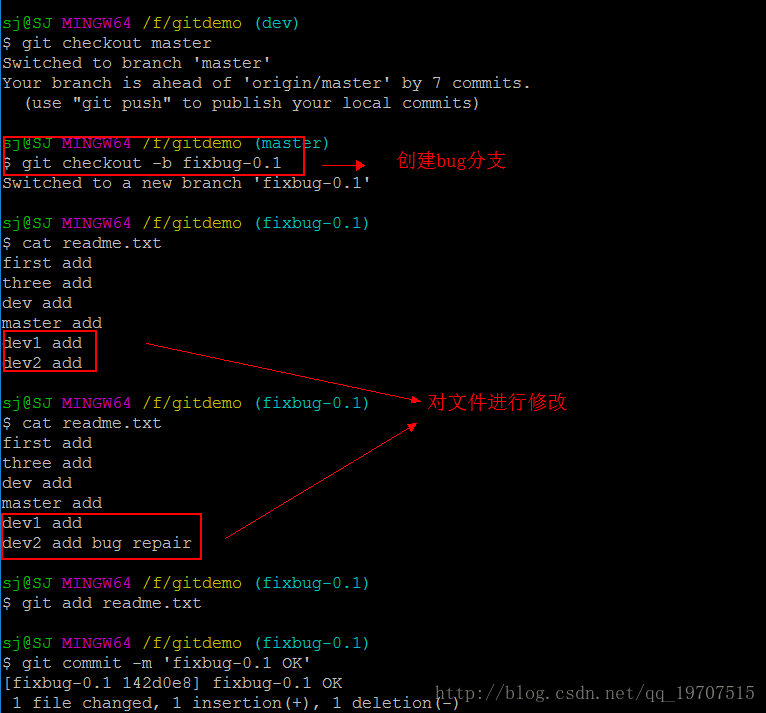
--查看stash集

git stash list

--当bug修复后，再使用如下命令，回到工作现场

git stash pop

* 1
* 2
* 3
* 4
* 5
* 6

**六.多人协作**

--查看远程库

git remote

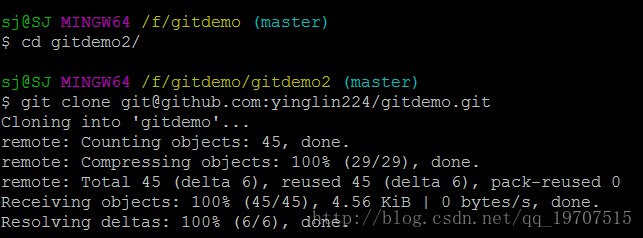
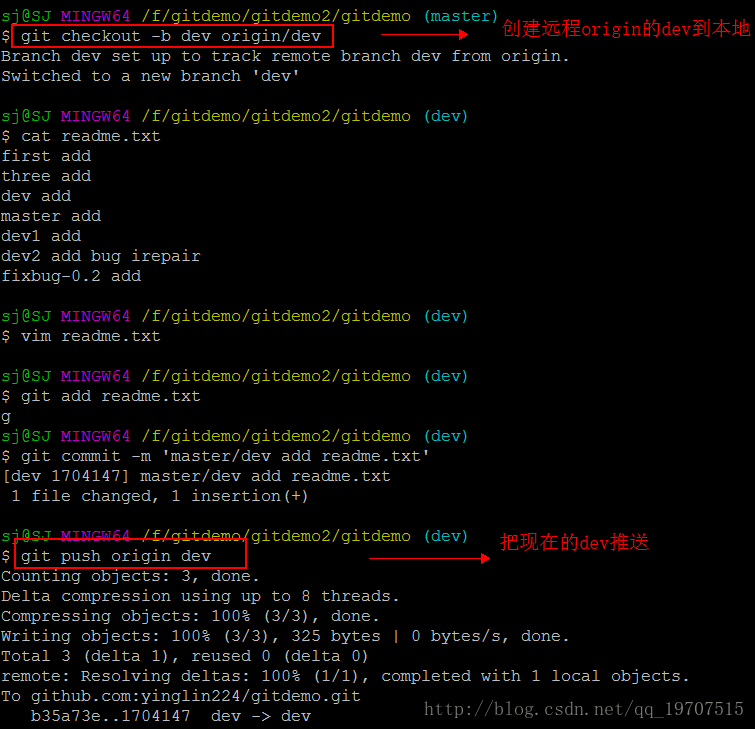
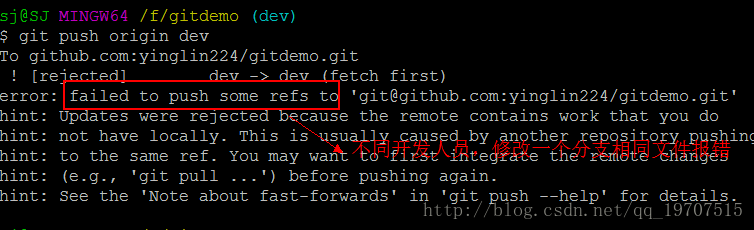
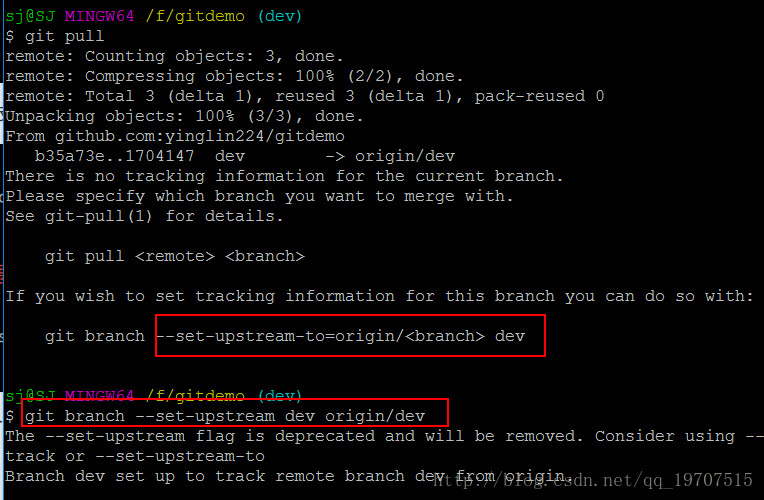
--查看远程详细信息

git remote -v

* 1
* 2
* 3
* 4

**（一）.推送分支**

我们在开发过程中，一般情况下，哪些分支需要推送   
1. master是主分支，需要时刻与远程库同步   
2. dev是开发分支，开发人员都在上面工作，也需要与远程同步   
3. 其他临时分支（feature/release/fixbug），一般合并到dev,使用完删除

**（二）.抓取分支**   
模拟，本地新建一个文件夹gitdemo2   
将dev推送到远程   
   
进入gitdemo2目录，将远程库clone到本地   
   
现在甲需要在dev分支上做开发，就必须把origin的dev分支弄到本地来，甲完成dev上面的开发后，推送   
   
同步的，乙也对dev分支上的相同文件进行修改，推送失败   
   
根据提示，说明更改有冲突，则使用命令git pull 从origin/dev获取最新的提交，解决冲突，再推送   


|  |
| --- |
| 多人协作模式：  1.首先，可以试图用git push origin branch-name推送自己的修改；   2.如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；   3.如果合并有冲突，则解决冲突，并在本地提交；   4.没有冲突或者解决掉冲突后，再用git push origin branch-name推送就能成功！   如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch –set-upstream branch-name origin/branch-name |