**一、安装svn**

1、先在服务端安装包

yum install -y subversion

2、创建版本库

mkdir -p /data/svnroot/myproject

3、初始化一些文件

svnadmin create /data/svnroot/myproject/

4、主要配置/data/svnroot/myproject/conf 里面的三个文件

5、vim /data/svnroot/myproject/conf/authz

admins=aming,user1 //admins组名；aming，user1

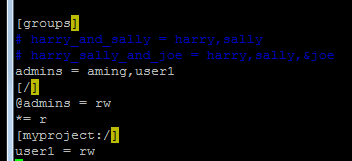
[/] // “/”是目录，并不是系统根目录而是项目的目录/data/svnroot/myproject

@admins = rw // admins组具有读写权限

\*= r //除开admins组的其他组具有可读权限

[myproject:/] //这里是针对有多个项目时，以项目来设置权限

user1 = rw



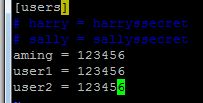
6、vim /data/svnroot/myproject/conf/passwd

aming = 123456

user1 = 123456

user2 = 123456

左边是用户名，右边是密码



7、vim /data/svnroot/myproject/conf/svnserve.conf

anon-access = none //匿名用户无权限

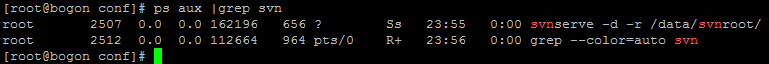
auth-access = write //授权用户可写

password-db = passwd //密码文件

authz-db = authz //权限文件

realm = /data/svnroot/myproject //对哪个项目生效（这里是myproject项目）

svnserve -d -r /data/svnroot/ //启动服务



备注： svnserver.conf里面设定的整个项目的读写权限，而authz是针对个人用户的。 若svnserver.conf中设置为写，但是个人用户却为只读，那最终权限是只读。若svnserver.conf里面是只读，即使authz设置为写，也不能写。

**二、客户端上使用svn(linux)**

1、yum install -y subversion //在客户机上安装包

2、在客户机上创建 mkdir /mnt/svntest目录

3、svn checkout svn://192.168.1.33/myproject --username=aming

在客户机上导出服务端的项目，IP为服务器IP，“aming”为服务端设置的用户名，密码为服务端设置的密码。导出后会在该目录下发现：



LANG=zh\_CN.UTF-8 //设置中文模式。

4、svn add . //添加到版本控制中心

5、svn commit -m “add file” //把文件上传到服务器

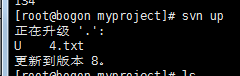
svn add 1.txt

svn commit -m "add 1.txt"

这样文件1.txt就上传到服务端了，如果其他客户端要使用上传的数据，执行第2、3步。

如果1.txt的内容更新了，请重新用第5上传到服务端，如果其他客户端要用就用svn up更新





记录用户名和密码的文件，如果删除，有文件更新执行svn up会提示输入登录当前系统用户的密码（这直接回车跳过），然后输入关联服务端时创建的用户名和密码（我创建的用户是aming，密码是123456）

[root@bogon svn.simple]# cat 8bab05daae0810581c94016d901899cb

K 8

passtype

V 6

simple

K 8

password

V 6

123456 //密码

K 15

svn:realmstring

V 49

<svn://192.168.1.33:3690> /data/svnroot/myproject

K 8

username //用户名

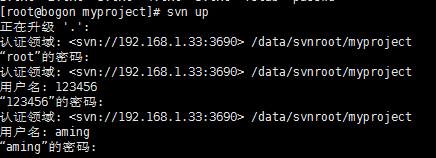
V 5

user1

END

[root@bogon svn.simple]# pwd

/root/.subversion/auth/svn.simple



删除文件,在客户端上操作的

[root@f myproject]# svn delete \*.txt //删除本地

D 1.txt

D 2.txt

D 3.txt

D 4.txt

D 5.txt

[root@f myproject]# svn commit -m "delete \*.txt" //删除服务端数据

删除 1.txt

删除 2.txt

删除 3.txt

删除 4.txt

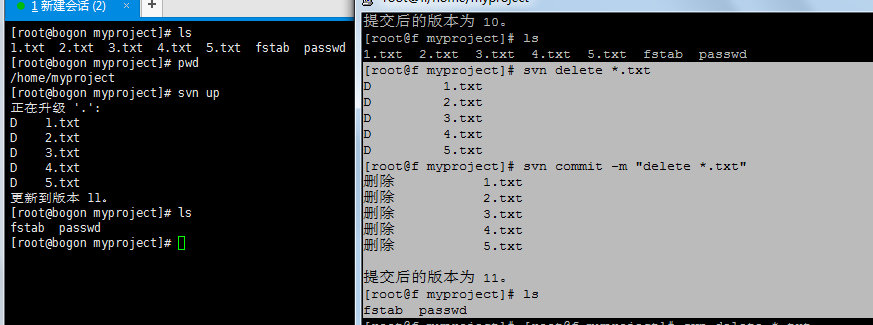
删除 5.txt

提交后的版本为 11。

[root@f myproject]# ls //这是删除后剩下的数据

fstab passwd

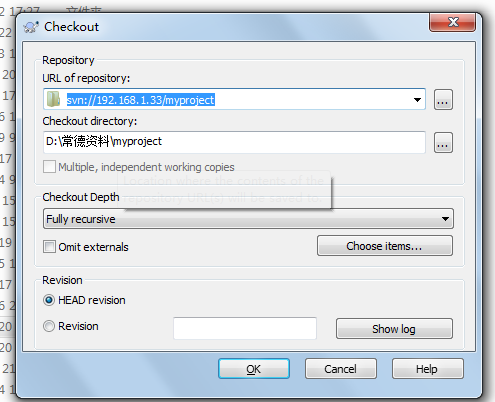
6、同比前后的对比



7、svn log //查看日志

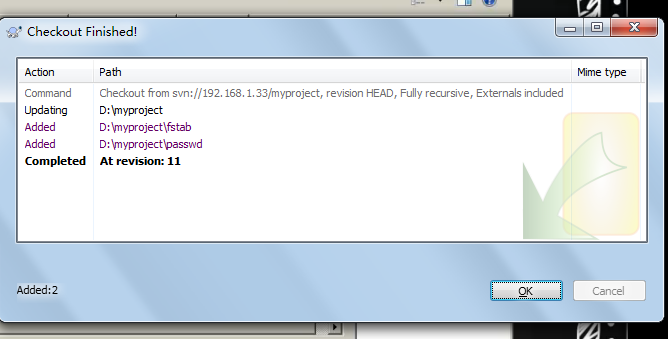
**8、windows 上使用svn**

1、新一个myproject的文件夹，鼠标右键文件夹checkout



2、点击OK，输入用户名和密码

我的是aming ,密码：123456

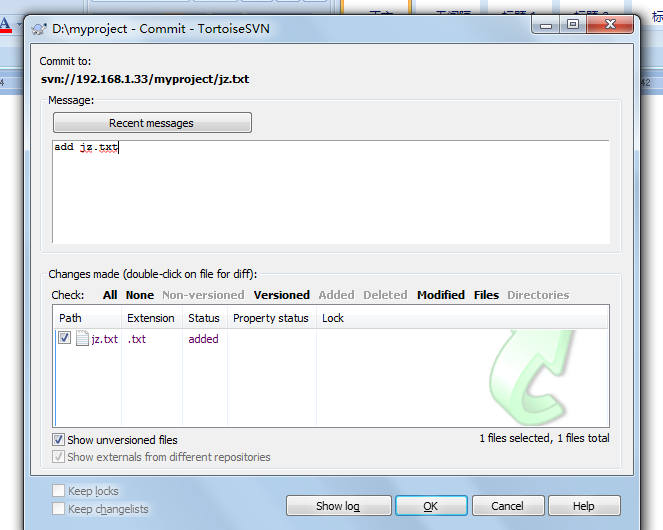


3、在创建的myproject文件夹里新建一个jz.txt,里面自定义一些东西

鼠标右键该文件选择TortoiseSVN- - add

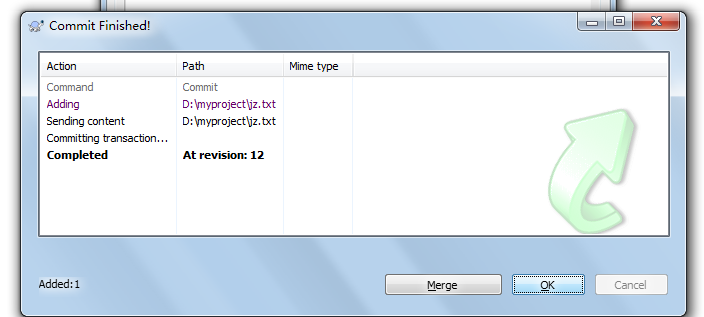
4、鼠标右键该文件选择TortoiseSVN- - commit

add jz.txt //做的描述



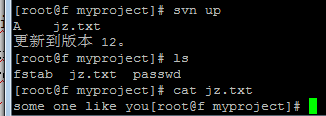
点击OK

最后这里点击OK



5、测试是否成功

在linux客户端上执行过“svn checkout svn://192.168.1.33/myproject --username=aming”这个命令的目录里 执行svn up 看看这个jz.txt有没有同步过来。



在linux客户端上，那个目录里添加一个linux.txt，自定义一些内容，看看windows能不能同步过来。

[root@f myproject]# vim linux.txt

[root@f myproject]# svn add linux.txt

A linux.txt

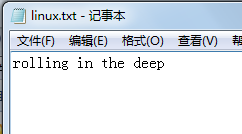
[root@f myproject]# svn commit -m "add linux.txt"

增加 linux.txt

传输文件数据.

提交后的版本为 13。

在windows上鼠标右键myprojiect这个目录，选择update，同步成功。



**三、单机上使用git**

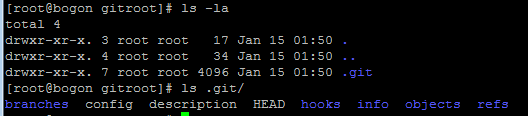
1、yum install -y git //包安装

2、mkdir /data/gitroot //创建目录

3、cd /data/gitroot //进入该目录

4、git init //初始化仓库。

初始化完成后会生成一些东西。



5、自定义编辑一个文件：如 1.txt，里面的内容为 “hello git”

git add 1.txt

git commit -m "add 1.txt"

我在做第二步操作时，报错需要配置用户名和邮件地址

1. git config --global user.name "jz"
2. $ git config --global user.email jz@163.com

6、更改1.txt的内容，再次操作

git add 1.txt

git commit -m "add 1.txt agine"

7、git log --pretty=oneline //前面的版本ID，可用于回滚到指定版本

666869e85ae7ee3cb75c2909f5e1355becc72728 add 1.txt agine

2a92c4f99ec1d322929a04c6db93c6fee156386a add 1.txt agine

9127df622022f6a54a761bbd74c875c376cfe10d add 1.txt //这是最原始版本

8、测试回滚，下面是目前1.txt内容

[root@bogon gitroot]# cat 1.txt

dfadsfads

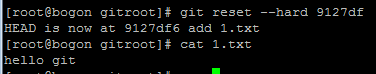
fasdf

fadsf

ashello git

dff3323232

git reset --hard 9127df //回滚到最初的版本



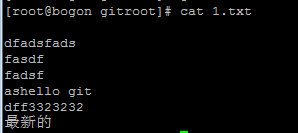
git reset --hard 666869 //还可以滚到最新的版本

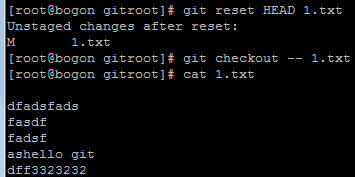
git reflog //这里可以找到所有版本ID

9、测试删除该文件，rm -rf 1.txt

git checkout -- 1.txt //恢复删除的文件

10、如果1.txt文件修改，add后但没有commit，再想回退到上一次提交的状态，可以使用git reset HEAD 1.txt，再执行git checkout -- 1.txt





10、删除文件，这样删除后还是可以用8步完成回滚

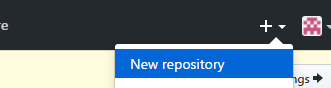
git rm 1.txt

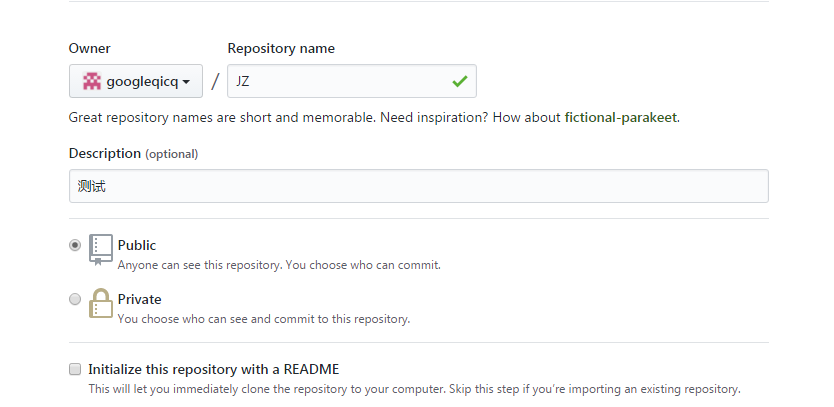
git commit -m "delete 1.txt"

**三、建立远程仓库（把服务器上的文件上传大github.com上去）**

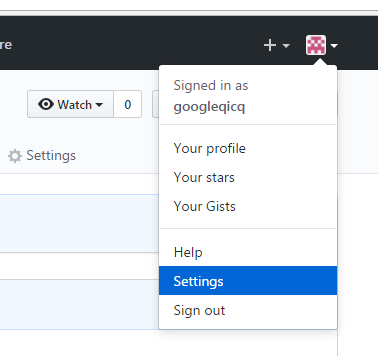
1、现在github.com上注册

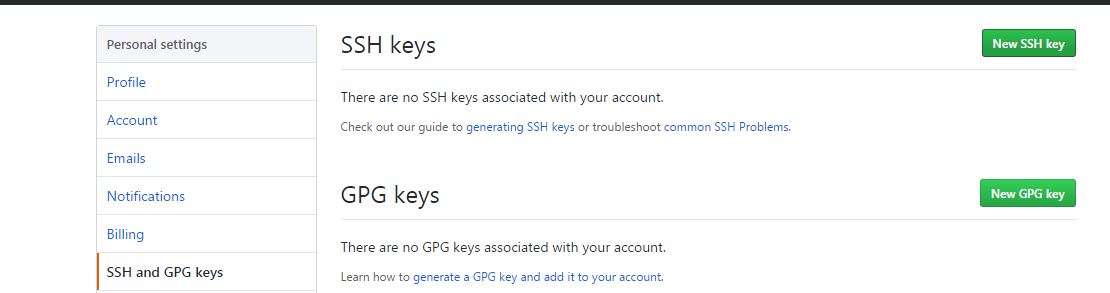
2、在githubweb上新建仓库

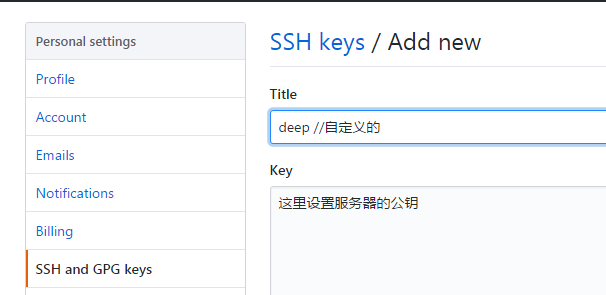




3、设置密钥







ssh-keygen //生成服务器密钥对，将cat .ssh/id\_rsa.pub的内容放入key中。

4、注意：此时选择的是“ssh”



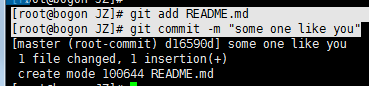
创建目录mkdir /mnt/JZ --- cd /mnt/JZ

echo "# JZ" >> README.md

git init //初始化

[root@bogon JZ]# git add README.md

[root@bogon JZ]# git commit -m "some one like you"



本地仓库的内容推送到远程仓库

git remote add origin git@github.com:googleqicq/JZ.git

git push -u origin master //有3各告警

a、Warning: Permanently added 'github.com,192.30.255.112' (RSA) to the list of known hosts.

解决:vim /ets/hosts



其它2个告警：

warning: push.default is unset; its implicit value is changing in

Git 2.0 from 'matching' to 'simple'. To squelch this message

and maintain the current behavior after the default changes, use:

git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.

(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode

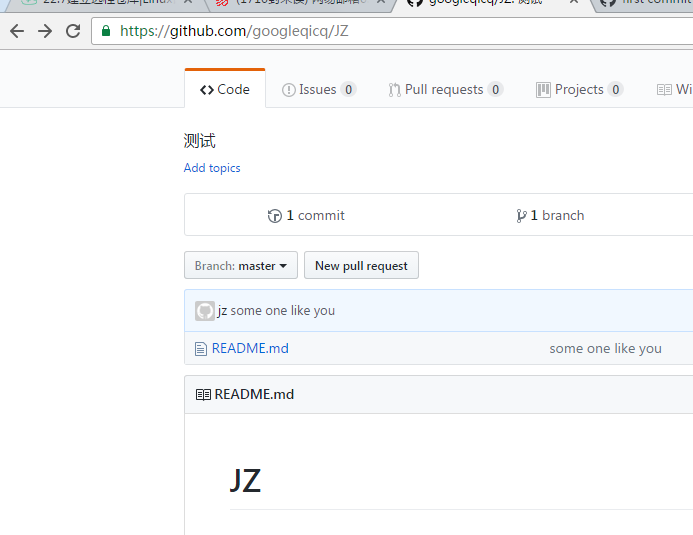
'current' instead of 'simple' if you sometimes use older versions of Git)

解决：

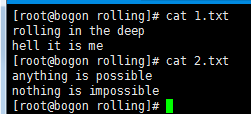
[root@bogon rolling]# git config --global push.default simple

[root@bogon rolling]# git config --global push.default matching

刷新web



5、/mnt/rolling 增加1.txt 和2.txt



[root@bogon rolling]# git add 1.txt

[root@bogon rolling]# git commit -m "add 1.txt"

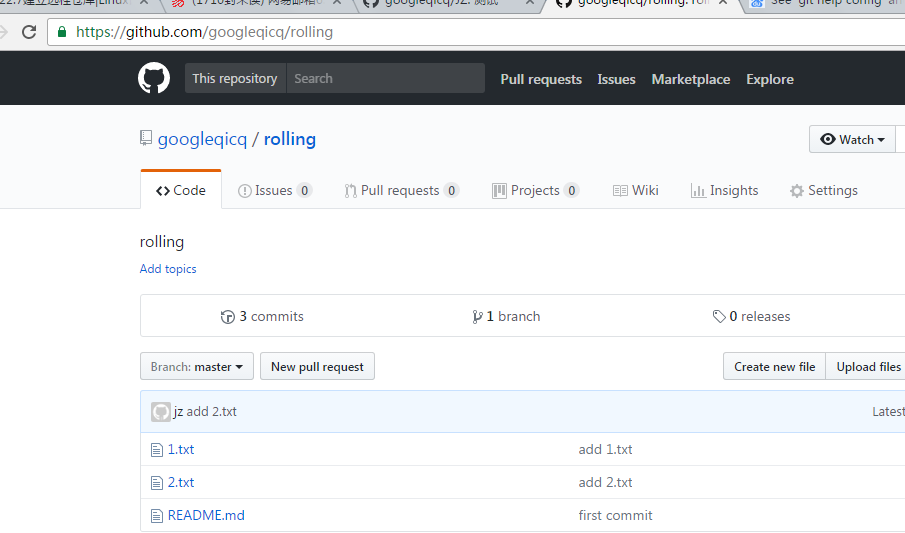
[root@bogon rolling]# git push

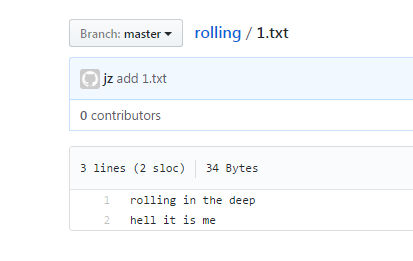
[root@bogon rolling]# git add 2.txt

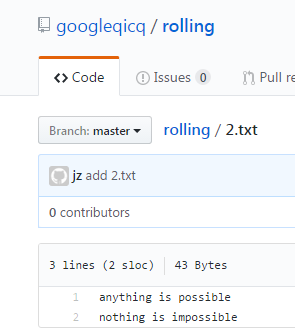
[root@bogon rolling]# git commit -m "add 2.txt"

[root@bogon rolling]# git push

然后刷新对应远程仓库是web界面

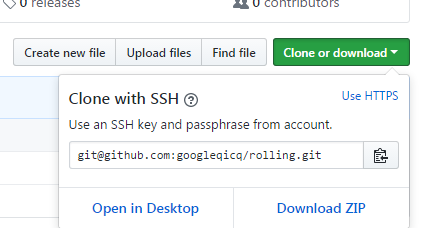




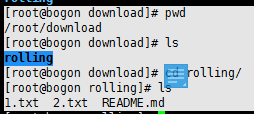


**四、克隆远程仓库（远程仓库下载到本地服务器）**

1、登录到https://github.com/googleqicq/rolling上找到地址，直接克隆



git clone git@github.com:googleqicq/rolling.git



更改1.txt内容后再上传到web上去。

[root@bogon rolling]# vim 1.txt

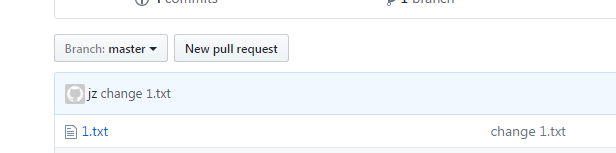
[root@bogon rolling]# git add 1.txt

[root@bogon rolling]# git commit -m "change 1.txt"

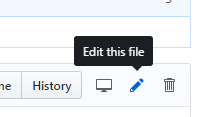
[master a06143b] change 1.txt

1 file changed, 11 insertions(+)

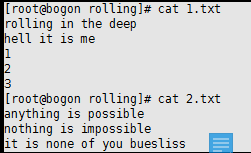
[root@bogon rolling]# git push



在web上编辑更改1.txt和2.txt的内容,



[root@bogon rolling]# git pull //将web上更新的内容和本地服务器的内容同步



**五、分支管理**

1、[root@bogon gitroot]# git branch //查看分支

\* master

2、[root@bogon gitroot]# git branch aming //创建分支

3、[root@bogon gitroot]# git branch //这时候就有2个分支了，‘\*’表示当前你在哪个分支上

aming

\* master

4、[root@bogon gitroot]# git checkout aming //切换到‘aming’分支

Switched to branch 'aming'

[root@bogon gitroot]# git branch

\* aming

master

5、[root@bogon gitroot]# ls

1.txt 2.txt // 1.txt是master分支里面的，2.txt是我在aming分支新建的

[root@bogon gitroot]# git add 2.txt

[root@bogon gitroot]# git commit -m "add 2.txt"

[root@bogon gitroot]# git checkout master //切换到master分支

[root@bogon gitroot]# ls

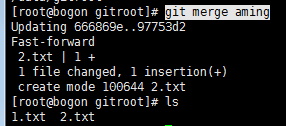
1.txt //并没有发现有aming分支的2.txt，说明分支与分支之间是相互隔离的，但aming分支里面却有1.txt。

如果当前分支是master，你新建一个aming分支，那么aming分支就会有1.txt,我在aming分支里面新建一个2.txt,然后我在aming分支里新建一个rolling分支，那么rolling分支里面会同时存在1.txt和2.txt。

6、将aming分支合并到master分支里面去

Ⅰ、git checkout master //切换到master分支下

Ⅱ、git merge aming //切换后，master分支里面有了2.txt



7、合并时有冲突的处理

Ⅰ、vim 2.txt //改变master分支里面 2.txt的内容

[root@bogon gitroot]# vim 2.txt //新增一些东西

[root@bogon gitroot]# git add 2.txt

[root@bogon gitroot]# git commit -m "ch 2.txt"

Ⅱ、在aming分支上的操作

[root@bogon gitroot]# git checkout aming

Switched to branch 'aming'

[root@bogon gitroot]# vim 2.txt //删除一些东西

[root@bogon gitroot]# git add 2.txt

[root@bogon gitroot]# git commit -m "ch 2.txt"

Ⅲ、在master分支上操作

git checkout master

[root@bogon gitroot]# git merge aming //再次合并就报错了

自动合并 2.txt

冲突（内容）：合并冲突于 2.txt

自动合并失败，修正冲突然后提交修正的结果。

[root@bogon gitroot]# cat 2.txt //这时候master 2.txt的内容

<<<<<<< HEAD

rolling in the deep

some one like you

i wish nothing bu the best wish for you to do not for get me

i beg

======= //这是不同的部分

rolling in //这是相同的部分

>>>>>>> aming

[root@bogon gitroot]# git checkout aming //有冲突时

2.txt: needs merge

error: 您需要先解决当前索引的冲突

Ⅳ、改变master分支2.txt的内容，让其和aming分支2.txt内容一样

[root@bogon gitroot]# vim 2.txt

[root@bogon gitroot]# git commit -a

[master 3493d29] Merge branch 'aming'

[root@bogon gitroot]# git commit -m "ch 2.txt"

git checkout aming //这就可以正常切换过来

8、[root@bogon gitroot]# git branch -d aming //不能在当前分支里删除分支，需要切换出去再删

error: 无法删除您当前所在的分支 'aming'。

[root@bogon gitroot]# git checkout master

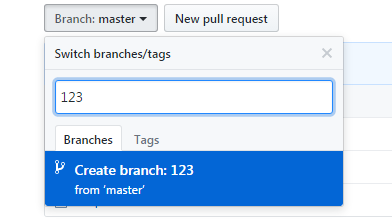
切换到分支 'master'

[root@bogon gitroot]# git branch -d aming

已删除分支 aming（曾为 4fddf0c）。

**六、远程分支管理**

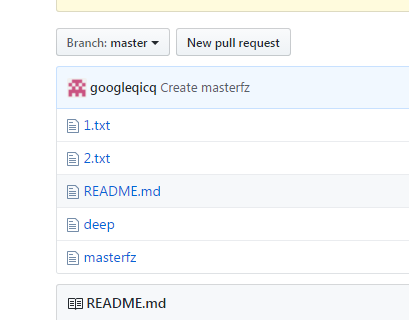
1、创建其他分支



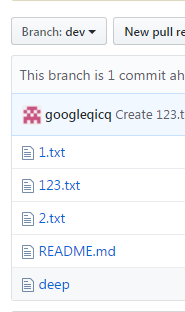
2、将文件克隆到本地设备，实际上是克隆的master分支的文件，但是自己新建的dev分支里面的东西没克隆到本地

git clone git@github.com:googleqicq/rolling.git





dev 分支的文件如下：



3、查看远程分支

[root@bogon rolling]# git ls-remote origin

2337fcea0b97b50c914042b3c958c1ed32e994d5 HEAD

2f6fdac752eb5bdb10aa06ecc7048d09170e64e1 refs/heads/dev

2337fcea0b97b50c914042b3c958c1ed32e994d5 refs/heads/master

git checkout -b dev origin/dev //远程的分支搞到本地



4、在本地的dev分支上新增一个文件，然后上传到远程去（web上）

[root@bogon rolling]# ls

123.txt 1.txt 2.txt deep README.md

[root@bogon rolling]# vim deep123.txt

[root@bogon rolling]# git add deep123.txt

[root@bogon rolling]# git commit -m "ch deep123.txt"

[dev 593cfac] ch deep123.txt

1 file changed, 11 insertions(+)

create mode 100644 deep123.txt

[root@bogon rolling]# git push //默认是更新所有分支的，比如我的master和dev分支里面的内容更新了。如果要指定更新某个分支，git push origin branch-name(就是指定分支名字)

Counting objects: 4, done.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 295 bytes | 0 bytes/s, done.

Total 3 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To git@github.com:googleqicq/rolling.git

2f6fdac..593cfac dev -> dev

5、本地设备新建分支，更新到远程上去（web），本地分支搞到远程去

git branch dev2 //新建分支

git checkout dev2 //切换到该分支

vim dev2.txt //在该分支下新增文件

git add dev2.txt

git commit -m "add dev2.txt"

git push origin dev2 //更新