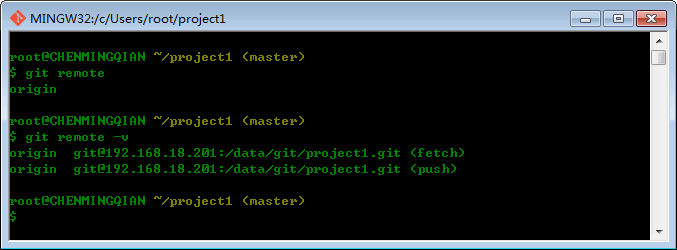
9.团队多人开发协作

在上面的章节中我们讲解了Git的分支管理策略，一般开发团队中有这样几个分支，master、develop、feature、release、 bug、other分支，或者你还有其它分支，那有博友会问了，你讲了那么多分支，都在本地放着我们怎么查看和推送分支到远程服务器上呢？嘿嘿，我们说大 家别急我们在这一章节中就来重点讲解，在团队多人协作中的分支推送与抓取。

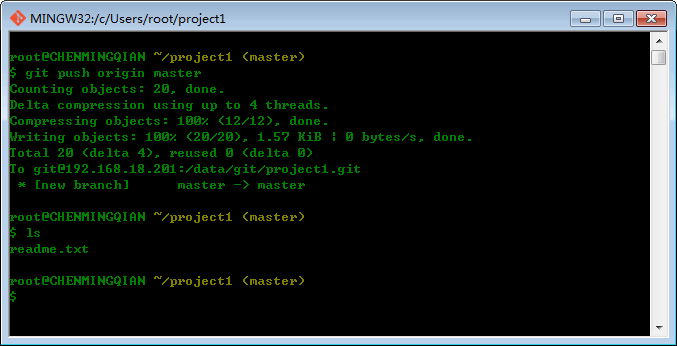
1).查看远程仓库分支

查看远程仓库的信息，如下图：    


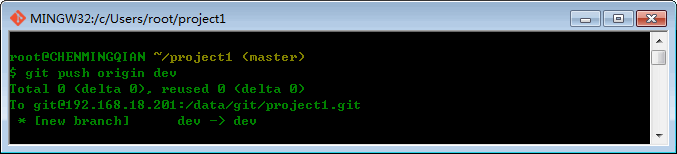
大家可以看到git remote命令可以查看远程仓库，加-v选项可以查看详细信息。上面显示了你抓取和推送的origin（源）信息。 当你从远程仓库克隆时，实际上Git自动把本地的master分支和远程的master分支对应起来了，并且远程仓库的默认名称是origin。

2).分支推送

下面我们来演示一下将本地分支推送到远程的仓库中，如下图：



大家可以看到，我们的本地master分支与远程的master分支已同步。下面我们来演示一下同步dev分支，如下图：



有博友会问了，我本地有很多分支，有哪些分支需要推送到远程仓库中呢？一般是这样的，

* master 分支是主分支，要时刻与远程同步，一般我们发布最新版本就用master分支
* develop 分支是开发分支，团队中所有人都在这个分支上开发，所以也需要与远程同步
* bug 分支一般只在本地使用来修复bug，一般不需推送远程仓库中
* feature 分支是否需要推送到远程，要看是不是有几个人合作开发新功能，如果你是一个开发，那就留在本地吧
* release 分支一般是系统管理，推送或抓取的分支一般与开发人员无关
* other 分支大家按需求分配

3).分支抓取

现在你一同事在电脑上克隆一份仓库，我们来演示一下。

生成公钥：

[root@localhost ~]# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

60:a6:f3:71:d1:1e:0f:5f:3f:0c:6f:a3:61:4c:28:c0 root@localhost.localdomain

The key's randomart image is:

+--[ RSA 2048]----+

|&#160;&#160;&#160;&#160;&#160; ..&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160; |

|&#160;&#160;&#160;&#160;&#160;&#160; E..&#160; .&#160;&#160;&#160; |

|&#160;&#160;&#160;&#160;&#160; + ..+. o.&#160; |

|&#160;&#160;&#160;&#160; + . o.=o.+. |

|&#160;&#160;&#160; o . S . o+ \*.|

|&#160;&#160;&#160;&#160; o o&#160;&#160;&#160; . + o|

|&#160;&#160;&#160;&#160;&#160; .&#160;&#160;&#160;&#160;&#160; .&#160;&#160; |

|&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160; |

|&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160;&#160; |

+-----------------+

[root@localhost ~]#

[root@localhost ~]# cd .ssh/

[root@localhost .ssh]# cat id\_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAx/ms/8BGBrG2FwHNySjFsVdpldArSzS6CkrdFeqJd0YAZfo/e2RwyZ9HkBqp3xcBZxCRdifnPbfSHdACLbVALfw2Dj6s0jJmdg3AHth13hF1qeljlQarXTTkheHNuFhi3OF/MVKZXJh0jRT2xU9UL4GRrtodOmclU6DelBFbqlw7yCj0TFXZSZk3UmHPHG3I2E3il/7jWdqw0qsys85rUL/Ugs50RUcFn53avqUiAYSU+neXiLmHmshCLVpdbDiOxRjfFPVjTgLh88hDLwkcYL4Lrzc5My3glgiEjrSn43WxDdbOiGUyZz1TdJKNngq7OJK+SuA/FkMEA+pXoTIAwQ==

root@localhost.localdomain

将生成的公司增加到git服务器上：

[root@test ~]# cat /data/git/.ssh/authorized\_keys

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAwMU4FKB6NRc3XSoIunWsdvVVuSoncbVb5Al6lB3ciswBVd++YmJFhqwkITNGccrO5sycROs9+Fbjgd6oBSzNuaBtCIbwNNsEyM/henTl2euI3XsnJQ/ITr6c/q0P3WoGl4E2QFQ2kZqs+1eDC0CgHcBrqvFv1Jr414sVYK9lfZwIF+jDdtaBOrSJuq1Agx9pGUFUEB4tQfkXxsWm/MvOmKAVvduKDE1eenUEL9zzyeELPcSXLe3NOoTjZhkX6EEXxQR1ZiZRFywLpfM4qopZ10to2KIUyVtzw6hx6V3cg7kn40lYVW0EAMATw9dVldwcRUI+kJzJSKUTKDVSwY3/+Q== root@CHENMINGQIAN

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAx/ms/8BGBrG2FwHNySjFsVdpldArSzS6CkrdFeqJd0YAZfo/e2RwyZ9HkBqp3xcBZxCRdifnPbfSHdACLbVALfw2Dj6s0jJmdg3AHth13hF1qeljlQarXTTkheHNuFhi3OF/MVKZXJh0jRT2xU9UL4GRrtodOmclU6DelBFbqlw7yCj0TFXZSZk3UmHPHG3I2E3il/7jWdqw0qsys85rUL/Ugs50RUcFn53avqUiAYSU+neXiLmHmshCLVpdbDiOxRjfFPVjTgLh88hDLwkcYL4Lrzc5My3glgiEjrSn43WxDdbOiGUyZz1TdJKNngq7OJK+SuA/FkMEA+pXoTIAwQ== root@localhost.localdomain

克隆远程版本库：

克隆远程版本库：

[root@localhost .ssh]# cd /data/  
[root@localhost data]# ls  
lost+found  
[root@localhost data]# git --version  
git version 1.7.1  
[root@localhost data]# git clone git@192.168.18.201:/data/git/project1.git  
Initialized empty Git repository in /data/project1/.git/  
The authenticity of host '192.168.18.201 (192.168.18.201)' can't be established.  
RSA key fingerprint is 3b:52:6b:ea:4d:50:7c:b2:9e:66:e4:0e:2e:21:98:be.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.18.201' (RSA) to the list of known hosts.  
remote: Counting objects: 20, done.  
remote: Compressing objects: 100% (12/12), done.  
remote: Total 20 (delta 4), reused 0 (delta 0)  
Receiving objects: 100% (20/20), done.  
Resolving deltas: 100% (4/4), done.  
[root@localhost data]# ls  
lost+found&#160; project1  
[root@localhost data]# cd project1/  
[root@localhost project1]# ls  
readme.txt

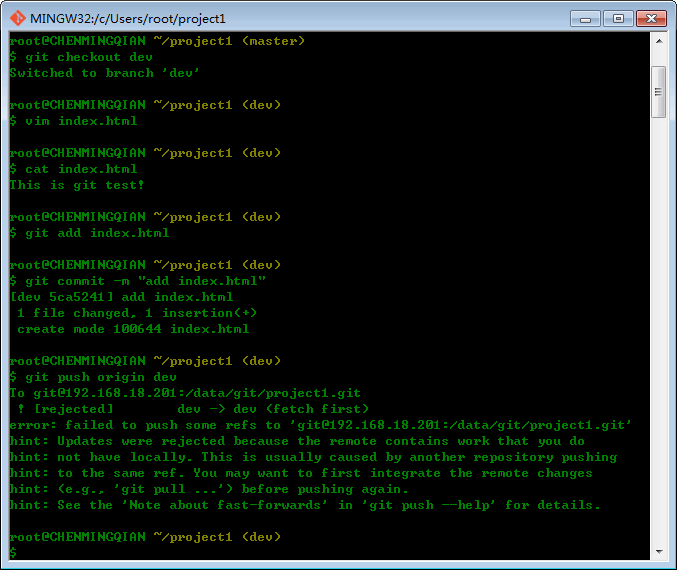
我们现在查看一下分支：

[root@localhost project1]# git branch  
\* master  
你同事要在dev分支上开发，就得创建与远程origin一样的dev分支到本地的仓库上，下面我们来创建一下：  
[root@localhost project1]# git checkout -b dev origin/dev  
Branch dev set up to track remote branch dev from origin.  
Switched to a new branch 'dev'  
[root@localhost project1]# git branch  \* dev  master  
[root@localhost project1]#

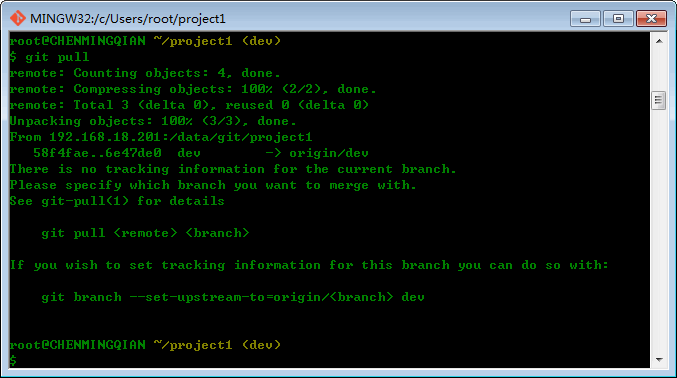
现在你同事就可以在dev分支上开发了，下面我们新建一些文件并提交到远程dev分支：

[root@localhost project1]# touch index.html  
[root@localhost project1]# vim index.html  
This git test index.html!  
[root@localhost project1]# git add index.html  
[root@localhost project1]# git commit -m "add index.html"  
[dev 6e47de0] add index.html  
Committer: root <root@localhost.localdomain>  
Your name and email address were configured automatically based  on your username and hostname.Please check that they are accurate.  
You can suppress this message by setting them explicitly:  
git config --global user.name "Your Name"  
git config --global user.email you@example.com  
If the identity used for this commit is wrong, you can fix it with:  
git commit --amend --author='Your Name <you@example.com>'  
1 files changed, 1 insertions(+), 0 deletions(-)  create mode 100644 index.html  
[root@localhost project1]# git config --global user.name "leo"  
[root@localhost project1]# git config --global user.email "leo@jjhh.com"  
[root@localhost project1]# git commit -m "add index.html"  
# On branch dev  # Your branch is ahead of 'origin/dev' by 1 commit.  
#  nothing to commit (working directory clean)  
[root@localhost project1]# git status  
# On branch dev  # Your branch is ahead of 'origin/dev' by 1 commit.  
#  nothing to commit (working directory clean)  
[root@localhost project1]# git push origin dev  
Counting objects: 4, done.  Delta compression using up to 4 threads.  
Compressing objects: 100% (2/2), done.  Writing objects: 100% (3/3), 305 bytes, done.  
Total 3 (delta 0), reused 0 (delta 0)  To git@192.168.18.201:/data/git/project1.git  
58f4fae..6e47de0; dev –> de

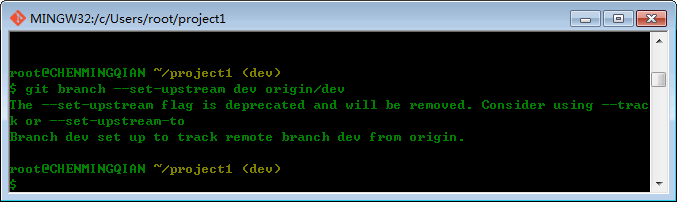
你的同事向origin/dev分支提交了一个index.html页面，现在你也在修改这个文件，并提交：



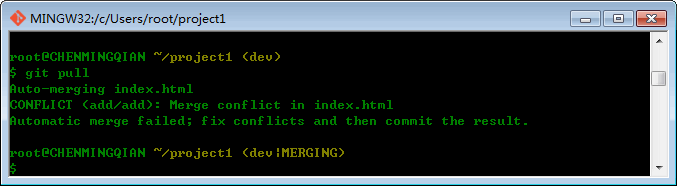
推送失败，因为你同事的最新提交和你推送的提交有冲突，Git提示我们，先用git pull把最新的提交从origin/dev抓下来，然后在本地合并解决冲突，再推送：



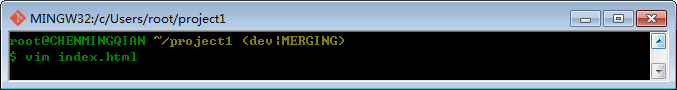
git pull 失败了，原因是没有将dev分支与远程origin/dev分支进行链接，Git提示我们设置dev和origin/dev的链接：

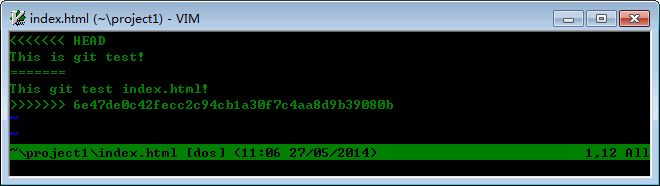


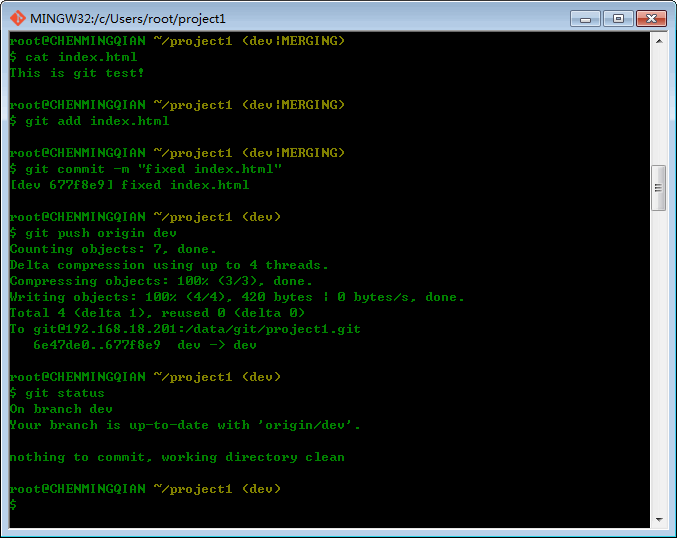
下面我们再来git pull一下试试：



git pull 成功，但是合并有冲突需要手动解决，解决的方法和分支管理中的解决冲突完全一样。解决后提再push：







好了，这样的我们的远程推送与抓取就讲解完成了，下面我们来总结一下。

10.总结

一般在团队中多人开发模式是这样的：

* 首先，可以试图用git push origin branch-name推送自己的修改
* 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
* 如果合并有冲突，则解决冲突，并在本地提交
* 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。

好了，到这里我们的Git分支管理就讲解到这里了。最后，希望大家有所收获^\_^……

**基于Gitolite的Git服务架设** <http://www.linuxidc.com/Linux/2014-02/96991.htm>

**Linux git命令参数及用法详解** <http://www.linuxidc.com/Linux/2012-01/51205.htm>

[Fedora](http://www.linuxidc.com/topicnews.aspx?tid=5)通过Http Proxy下载Git <http://www.linuxidc.com/Linux/2009-12/23170.htm>

在[Ubuntu](http://www.linuxidc.com/topicnews.aspx?tid=2) Server上安装Git <http://www.linuxidc.com/Linux/2009-06/20421.htm>

服务器端Git仓库的创建（Ubuntu） <http://www.linuxidc.com/Linux/2011-02/32542.htm>

Linux下Git简单使用教程（以[Android](http://www.linuxidc.com/topicnews.aspx?tid=11)为例） <http://www.linuxidc.com/Linux/2010-11/29883.htm>

Git权威指南 PDF高清中文版 <http://www.linuxidc.com/Linux/2013-10/91053.htm>

**Git 的详细介绍**：[请点这里](http://www.linuxidc.com/Linux/2013-10/91054.htm)  
**Git 的下载地址**：[请点这里](http://www.linuxidc.com/down.aspx?id=1022)