【11】当认为目前的代码在一定阶段内稳定，可以提交到测试环境下，git管理员如何打标签。

【12】为什么CI要求remote repository只能由一个master分支。

**作者：yelei**

**email:kursk.ye@gmail.com**

**本文内容：针对各种使用GIT典型场景，描述git操作过程。**

【1】开发者如何生成自己的SSH git账号

(1)客户端产生公私密匙。

**$ ssh-keygen -t rsa -C "git@192.170.100.15 yelei.kursk"**

**Generating public/private rsa key pair.**

**Enter file in which to save the key (/c/Users/kursk/.ssh/id\_rsa):**

**Enter passphrase (empty for no passphrase): #可以不输入密码，直接回车，密码为空，简单但不安全**

**Enter same passphrase again: #与上面一致**

**Your identification has been saved in /c/Users/kursk/.ssh/id\_rsa.**

**Your public key has been saved in /c/Users/kursk/.ssh/id\_rsa.pub.**

**The key fingerprint is:**

**e7:32:dc:8d:e8:f9:b6:0e:2a:7a:71:34:17:40:4c:34 git@192.170.100.15 yelei.kursk**

(2)将客户端id\_rsa.pub里的内容交给**git管理员**，由git管理员负责复制到服务端authorized\_keys文件内，如果有多个直接换行复制。

**$cat /home/git/.ssh/authorized\_keys**

**ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA4BpIh914++YT61aiMMjc6H+31FHB8TJjLhVQ1EKmnB72k+3VtoXa/CCrNqDrlVTJxx9Y003RCzJCMVhLmBBX+100BqSSzDLxjRt+L/WbLZtLDucSJmW1MZywlq/vFAGeLRnuWDP0VHHL2/0jBvY6sajpVdNXRWu3nuaMYPhP6ECYsuZDMBgsSALw/LvkoWA8ZCdy38B+L8FJUZ8J87Vnr+mDB/+b87u21fcE7a/fUiOIkZSAWPMJ78mznk2e2Mz0lmK2OqqJXFq5lk7+h7rAl/rd/aVayESmZ9Xx4S5JKqVwxMeZfAVBcYBNz2xkO+LXnzmX+jnSZkjV+mwN+JoSQQ== git@192.170.100.15**

(3)这样开发者通过客户端就可以访问了git repository

**$ git clone ssh://git@192.170.100.15:/home/git/git.repository/test.git test6**

**Cloning into 'test6'...**

**The authenticity of host '192.170.100.15: (192.170.100.15)' can't be established.**

**RSA key fingerprint is 97:5c:2f:aa:f1:19:0d:81:5d:6a:3e:97:94:18:01:84.**

**Are you sure you want to continue connecting (yes/no)? yes #输入yes,如果之前有密码，这里还要输入密码**

**Warning: Permanently added '192.170.100.15:,192.170.100.15' (RSA) to the list of known hosts.**

**remote: Counting objects: 3, done.**

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

**remote: Total 3 (delta 0), reused 0 (delta 0)**

**Receiving objects: 100% (3/3), done.**

**Checking connectivity... done**

提示：如果开发者忘记了密匙，可以在git gui/帮助/show ssh key可以看到生成的public key

【2】第一次如何下载代码，GIT的checkout与SVN有何不同？

答：git也有checkout命令，但与SVN 不同。SVN的做法是第一次从svn repository 上checkout 到local workcopy,以后每次再update到svn repository。

但git的checkout不是从repository上下载代码，而是切换分支，下面的例子中将讲到分支如何使用。

【3】我是一个GIT初学者，如何尽快上手。

答：首先，理解workspace、 index、 remote Repository的概念，掌握代码在这几个区域切换的命令，如commit、push、pull等，[官方图片说明](http://ndpsoftware.com/git-cheatsheet.html)是一个很好的帮助手册。

其次，多加以练习，局域网内提供了练习用工程，地址是：ssh://git@192.170.100.15:/home/git/git.repository/test.git.

如果可以访问外网，git官方还提供了一个web 版的[git练习](http://try.github.io/levels/1/challenges/1)。

下面简单地说明一些常用设置和命令。

(1)安装git client后，要设置客户端用户名和邮箱，这样才能知道谁又上传了最新的代码，为了方便理解，**要求所有的开发组成员使用"中文姓名全拼音"."个性名字"，例如yelei.kursk,zhengyan.teacher。**

**$ git config --global user.name "yelei.kursk"**

**$ git config --global user.email kursk.ye@gmail.com**

这样当我们上传代码到repository后，从log里可以看到哪些人何时上传或修改了什么代码。

(2)clone一个工程。现在我们已经有一个测试工程，我们需要将它clone到本地，命令是：

**$ git clone ssh://git@192.170.100.15:/home/git/git.repository/test.git test2**

**Cloning into 'test2'...**

**remote: Counting objects: 3, done.**

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

**remote: Total 3 (delta 0), reused 0 (delta 0)**

**Receiving objects: 100% (3/3), done.**

**Checking connectivity... done**

因为公私密匙对的设置上文已经描述，这里不再做说明。

这样在当前目录下就多了一个test2/目录。

(3)修改文件，并上传到local repository.

这里要先了解git中workspace/index(有时也称为staging area)/local repository三个概念。

首先要知道它们三个有个共同点，都在local也就是在你的笔记本电脑上——这是GIT比SVN上传快的原因。

workspace:类似SVN 的workcopy概念，开发者修改文件后会保存在这个区域。

index: 有时也称为staging area,用于存储被修改文件的索引。

local repositor：最终保存修改后并commit的文件的空间。

另外还有一个叫remote repositor,它是所有开发者交互的远程仓库，因为既然local repositor在每个开发者的本地电脑上，那必然要有一个可供所有开发者交换代码的空间，本例中它的地址就是

ssh://git@192.170.100.15:/home/git/git.repository/test.git.

下面以修改一个文件为例说明：

k**ursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ cat test2.txt**

**git is good tool for source code manage**

**kursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ vi test2.txt**

**kursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ cat test2.txt**

**git is good tool for source code manage**

**git is faster than svn #我增加了一行**

**kursk@KURSK-PC /D/gitworkspace/test2 (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: test2.txt**

**#**

**no changes added to commit (use "git add" and/or "git commit -a")**

这时workspace 已经改变，但是index的内容没有改变，通过git status 可以看到“**Changes not staged for commit**”的提示，并说明修改的内容

提示：操作过程中可以随时使用git status了解修改了哪些文件，以及这些文件存于什么状态。

下一步，我们通过git add将修改文件从workspace 推到index

**kursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ git add .** #git add . 注意后面有一个dot，不管修改、新增、删除了多少文件，只要在工程顶级目录执行这个命令，就可以将所有修改内容从workspace 推到index

**kursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ git status**

**# On branch master**

**# Changes to be committed:**

**# (use "git reset HEAD <file>..." to unstage)**

**#**

**# modified: test2.txt**

这时index内容已经改变，但是还没有commit到local repository,通过git status可以看得很清楚。

接下来，我们将修改文件从index commit 到local repository

**kursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ git commit -m "modify test2.txt" #commit命令用于**将修改文件从index commit 到local repository，这里的-m是说明内容，必须

**[master 6349e7f] modify test2.txt**

**1 file changed, 2 insertions(+), 1 deletion(-)**

**kursk@KURSK-PC /D/gitworkspace/test2 (master)**

**$ git status #commit后再查看文件状态，已经没有修改的文件了，这时workspace/index/local repository三个空间的文件状态完全一致了。**

**# On branch master**

**# Your branch is ahead of 'origin/master' by 1 commit.**

**# (use "git push" to publish your local commits)**

**#**

**nothing to commit, working directory clean**

实际的开发工作中的大多数时间，每天的代码提交到local repository就可以了，只是在每天下班前才需要将当天已经解决的问题，在编译不报错的情况，集中一次从local repository提交到remote repository.当然，如果有些人觉得随时提交到remote repository也可以，但是前提是必须保证编译可以通过，因为开发CI环境会自动PULL remote repository代码到开发应用服务器上，如果编译不通过就会导致自动发布失败！

**所以这里特别强调，发布到remote repository上的代码必须保证编译通过，违者重罚！**

注意：使用commit -a -m也可以从workspace一步提交到local repository。

下面继续说明。当local repository里准备好需要修改后的代码后，可以通过push将local repostiroy的内容推到remote repository。虽然这一过程看起来很简单，但实际工作中存在多人交互，所以经常需要先pull再push。

以两个开发者A和B为例，A和B两个开发者使用同一个remote repository,当A 将自己的代码Push 到remote repository后，B再PUSH时就会报错，过程如下

k**ursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ vi A.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git add .**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git commit -a -m "add a.html"**

**[master 31b9fe0] add a.html**

**1 file changed, 1 insertion(+)**

**create mode 100644 A.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git push**

**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)**

**Counting objects: 4, done.**

**Delta compression using up to 4 threads.**

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

**Writing objects: 100% (3/3), 321 bytes | 0 bytes/s, done.**

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

**To ssh://git@192.170.100.15:/home/git/git.repository/test.git**

**f294530..31b9fe0 master -> master**

上面开发者A增加了一个a.html文件，并push到remote repository

k**ursk@KURSK-PC /D/gitworkspace**

**$ cd dev\_B/**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ vi b.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git add .**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git commit -a -m "add file b.html"**

**[master 7262964] add file b.html**

**1 file changed, 1 insertion(+)**

**create mode 100644 b.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git push**

**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)**

**To ssh://git@192.170.100.15:/home/git/git.repository/test.git**

**! [rejected] master -> master (fetch first)**

**error: failed to push some refs to 'ssh://git@192.170.100.15:/home/git/git.repository/test.git'**

**hint: Updates were rejected because the remote contains work that you do**

**hint: not have locally. This is usually caused by another repository pushing**

**hint: to the same ref. You may want to first integrate the remote changes**

**hint: (e.g., 'git pull ...') before pushing again.**

**hint: See the 'Note about fast-forwards' in 'git push --help' for details.**

当开发者B在开发者A后PUSH时就报错了，原因是B目前的local repository的版本与remote repository的版本没有同一个版本节点。

因为开发者A的local repository和remote repository的版本一致，所以我们可以开发者A的机器执行git log查看日志列表，检查当前remote repository的版本历史。

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git log**

**commit 31b9fe04c05ff33c18fe889ca5a12d36bd6dcb71**

**Author: yelei.kursk <kursk.ye@gmail.com>**

**Date: Wed Apr 9 15:59:26 2014 +0800**

**add a.html**

**commit f2945305c9a619b4d5f74754d0783195c489bf93**

**Author: yelei.kursk <kursk.ye@gmail.com>**

**Date: Wed Apr 9 15:02:31 2014 +0800**

**modify test.txt**

**commit 8228bf6902acb5bfd849bc7298d3c38e4d59b735**

**Author: kursk-ye <kursk.ye@gmail.com>**

**Date: Wed Apr 9 14:53:03 2014 +0800**

**test2**

**commit e653668d9ad6516fbd7038fe67fcbe0bd9bcf434**

**Author: kursk-ye <kursk.ye@gmail.com>**

**Date: Wed Apr 9 14:00:36 2014 +0800**

**test**

remote repository当前版本号是：**31b9fe04c05ff33c18fe889ca5a12d36bd6dcb71**

**而开发者B的**local repository呢？**726296473f028b753a87c8a50b0506ef926a3421**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git log**

**commit 726296473f028b753a87c8a50b0506ef926a3421**

**Author: yelei.kursk <kursk.ye@gmail.com>**

**Date: Wed Apr 9 16:02:03 2014 +0800**

**add file b.html**

**commit f2945305c9a619b4d5f74754d0783195c489bf93**

**Author: yelei.kursk <kursk.ye@gmail.com>**

**Date: Wed Apr 9 15:02:31 2014 +0800**

**modify test.txt**

**commit 8228bf6902acb5bfd849bc7298d3c38e4d59b735**

**Author: kursk-ye <kursk.ye@gmail.com>**

**Date: Wed Apr 9 14:53:03 2014 +0800**

**test2**

**commit e653668d9ad6516fbd7038fe67fcbe0bd9bcf434**

**Author: kursk-ye <kursk.ye@gmail.com>**

**Date: Wed Apr 9 14:00:36 2014 +0800**

**test**

很明显，这时已经产生了两个分支，如下图：

**f2945305c9a619b4d5f74754d0783195c489bf93------------>31b9fe04c05ff33c18fe889ca5a12d36bd6dcb71**

**|**

**|**

**|**

**\------------------------------->726296473f028b753a87c8a50b0506ef926a3421**

应该说开发者几乎每次push时都会遇到这种情况，解决方法是先pull后push。

pull的作用是从remote repository推送文件到workspace,如果两个或多个人修改了同一个文件，会产生冲突，需要开发者手动解决，这点与SVN是一样的。

**kursk@KURSK-PC /D/gitworkspace/dev\_B (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 ssh://192.170.100.15:/home/git/git.repository/test**

**f294530..31b9fe0 master -> origin/master**

**Merge made by the 'recursive' strategy.**

**A.html | 1 +**

**1 file changed, 1 insertion(+)**

**create mode 100644 A.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git status**

**# On branch master**

**# Your branch is ahead of 'origin/master' by 2 commits.**

**# (use "git push" to publish your local commits)**

**#**

**nothing to commit, working directory clean**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git push**

**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)**

**Counting objects: 7, done.**

**Delta compression using up to 4 threads.**

**Compressing objects: 100% (4/4), done.**

**Writing objects: 100% (5/5), 684 bytes | 0 bytes/s, done.**

**Total 5 (delta 0), reused 0 (delta 0)**

**To ssh://git@192.170.100.15:/home/git/git.repository/test.git**

**31b9fe0..ef490f5 master -> master**

上面开发者B先PULL，后push。因为开发者A增加a.html,开发者B增加了b.html,没有修改同一个文件，所以没有产生文件冲突，所以可以马上push。下面说明是出现了文件冲突的情况：

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ vi A.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git commit -a -m "dev B modify a.html ,it's would conflict"**

**[master b0cd1fa] dev B modify a.html ,it's would conflict**

**1 file changed, 1 insertion(+)**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git pull**

**remote: Counting objects: 8, done.**

**remote: Compressing objects: 100% (4/4), done.**

**remote: Total 5 (delta 1), reused 0 (delta 0)**

**Unpacking objects: 100% (5/5), done.**

**From ssh://192.170.100.15:/home/git/git.repository/test**

**ef490f5..266d715 master -> origin/master**

**Auto-merging A.html**

**CONFLICT (content): Merge conflict in A.html**

**Automatic merge failed; fix conflicts and then commit the result.**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master|MERGING)**

**$ git status**

**# On branch master**

**# Your branch and 'origin/master' have diverged,**

**# and have 1 and 2 different commits each, respectively.**

**# (use "git pull" to merge the remote branch into yours)**

**#**

**# You have unmerged paths.**

**# (fix conflicts and run "git commit")**

**#**

**# Unmerged paths:**

**# (use "git add <file>..." to mark resolution)**

**#**

**# both modified: A.html**

**#**

**no changes added to commit (use "git add" and/or "git commit -a")**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master|MERGING)**

**$ vi A.html**

**1 it's push by dev A**

**2 <<<<<<< HEAD**

**3 it's modify by dev B**

**4 =======**

**5 it's push by dev A**

**6 >>>>>>> 266d71591fa0dd8976b14116626f8330b43836f0**

这里必须手动解决冲突，需要开发者判断哪些代码需要保留，哪些代码要删除，因为涉及多人的工作成果，修改时要慎重！

修改好文件，在本地调试完毕后，与前面的操作一样，先通过git add提交到index，然后commit

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master|MERGING)**

**$ git status**

**# On branch master**

**# Your branch and 'origin/master' have diverged,**

**# and have 1 and 2 different commits each, respectively.**

**# (use "git pull" to merge the remote branch into yours)**

**#**

**# You have unmerged paths.**

**# (fix conflicts and run "git commit")**

**#**

**# Unmerged paths:**

**# (use "git add <file>..." to mark resolution)**

**#**

**# both modified: A.html**

**#**

**no changes added to commit (use "git add" and/or "git commit -a")**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master|MERGING)**

**$ git add .**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master|MERGING)**

**$ git status**

**# On branch master**

**# Your branch and 'origin/master' have diverged,**

**# and have 1 and 2 different commits each, respectively.**

**# (use "git pull" to merge the remote branch into yours)**

**#**

**# All conflicts fixed but you are still merging.**

**# (use "git commit" to conclude merge)**

**#**

**# Changes to be committed:**

**#**

**# modified: A.html**

**#**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master|MERGING)**

**$ git commit -a -m "resovle conflict"**

**[master 2b50740] resovle conflict**

**kursk@KURSK-PC /D/gitworkspace/dev\_B (master)**

**$ git push**

**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)**

**Counting objects: 8, done.**

**Delta compression using up to 4 threads.**

**Compressing objects: 100% (3/3), done.**

**Writing objects: 100% (4/4), 568 bytes | 0 bytes/s, done.**

**Total 4 (delta 0), reused 0 (delta 0)**

**To ssh://git@192.170.100.15:/home/git/git.repository/test.git**

**266d715..2b50740 master -> master**

**这里特别说明一下，git并不能判断冲突是否被解决，因为只有人才能判断冲突是否被解决，如果开发者这时马虎大意，没有解决有冲突的文件，一样可以成功push到remote repository上，如果其他人又PULL了这些代码，将会对其他开发者也造成影响，所以对于这种不负责任的行为将严惩！请各开发者PUSH前严格检查自己的修改是否正确。**

以上是一般的提交代码过程，相信开发者掌握后日常的代码提交工作已经可以进行了。

【4】当修改了workspace，但还没有提交到index，我想恢复文件原来的内容，怎么处理？

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ vi A.html #我这里删除了一行**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ cat A.html**

**it's push by dev A**

**it's push by dev A**

**it's push by dev B**

**=======**

**it's push by dev A**

**>>>>>>> 7c63b7b02700bac8cb175e9f7e3f7cfb544016cf**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git checkout A.html**

**A.html**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git checkout A.html #在workspace里恢复**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ cat A.html**

**it's push by dev A**

**it's push by dev A**

**<<<<<<< HEAD #删除的一行回来了**

**it's push by dev B**

**=======**

**it's push by dev A**

**>>>>>>> 7c63b7b02700bac8cb175e9f7e3f7cfb544016cf**

【5】当修改了workspace，但已经提交到index，我想恢复文件原来的内容，怎么处理？

不能通过GIT恢复。因为你可能修改了很多文件，假定你只想恢复某一个文件而其他文件不恢复。因为index内容已经被修改，这种情况只能通过local repository取回，但是不能只恢复一个文件。所以这种情况你只能手动把这个文件内容修改，然后add,commit。

如果你发现更好的解决方法请通过邮件告诉我。

【6】文件已经commit到local repository了，但还没有push到remote repository上，如何恢复被修改的文件内容？

既然文件已经commit到local repository了，那么意味着workspace ,index, local repository的内容都已经被修改，而且状态一致，那么只能祈祷remote repository上的内容还没有被修改，通过PULL来恢复。但是如果这时remote repository上已经有人push过内容，并和你local repository内容冲突，那么问题就麻烦了，似乎没有什么好的快速的解决方法。

上述三个问题都是通过GIT恢复文件问题，在解决这类问题时可以通过git diff比较workspace ,index, local repository三个空间的同一个文件，详细内容请参考官方手册。

【7】每天开发者有多个修改的bug，但有些bug很花时间，不能满足每天push以后必须编译成功的时间要求，怎么办？

按照GIT官方推荐的工作方式，一个聪明的开发者在workspaces上应该至少为自己建两个分支，一个是默认的master分支，用于存放已经修改好的、调试成功、绝对保证编译成功或者功能实现、用于向remote repository提交的代码；一个是用于写代码的工作分支(本例称为coding),coding分支调试完一段成功的代码后，立刻合并进入master中，这样既能保证每天有可以提交的工作成果，也不会影响自己的工作效率。

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git checkout -b coding #checkout -b 相当于先创建一个分支，再checkout它**

**Switched to a new branch 'coding' #这里我创建了一个coding分支**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (coding)**

**$ vi gather2.jsp**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (coding)**

**$ cat gather2.jsp #在coding分支上开发新的代码**

**modify gather2.jsp to dev more function**

**this is coding in coding branch**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (coding)**

**$ git commit -a -m "modify in coding" #代码调试完成后，提交代码到local repository**

**[coding ead533f] modify in coding**

**1 file changed, 2 insertions(+), 1 deletion(-)**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (coding)**

**$ git checkout master #切换回master**

**Switched to branch 'master'**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git merge coding #git merge是一个复杂的命令，注意这里是"站"在master分支上，将coding分Updating 789aae1..ead533f #支的内容合并进入master分支，千万不能颠倒。**

**Fast-forward**

**gather2.jsp | 3 ++-**

**1 file changed, 2 insertions(+), 1 deletion(-)**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git status #因为master分支上分离出coding分支后没有新的节点，所以不会有冲突**

**# On branch master**

**# Your branch is ahead of 'origin/master' by 1 commit.**

**# (use "git push" to publish your local commits)**

**#**

**nothing to commit, working directory clean**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ cat gather2.jsp**

**modify gather2.jsp to dev more function**

**this is coding in coding branch**

**kursk@KURSK-PC /D/gitworkspace/dev\_A (master)**

**$ git push #将master分支push到remote repository**

**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)**

**Counting objects: 5, done.**

**Delta compression using up to 4 threads.**

**Compressing objects: 100% (3/3), done.**

**Writing objects: 100% (3/3), 321 bytes | 0 bytes/s, done.**

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

**To d:/git.remote/remote.helloCIServlet/**

**789aae1..ead533f master -> master**

**...............#后续可以切换到coding分支继续调试其他代码，调试完一部分功能就merge,如此反复**

注意git merge注意这里是"站"在master分支上，将coding分支的内容合并进入master分支，千万不能颠倒，否则应出现merge不成功，提示"up to date"。因为coding是从master上分离的，而且分离后master没有做任何修改去产生新的版本节点(实际工作中也是并且应该如此)，所以调试完coding分支后，git merge后没有任何文件冲突，可以很方便地合并。

这种工作方式的优点非常明显，master分支总是一个代码调试成功、功能完善的分支，当每天工作快结束或有需要时就提交到remote repository上，能体现一个开发者每天的工作成果；而coding分支总是处于调试状态，不影响每天要去的提交任务。这样，结合CI的自动build\定时发布能力，每天开发者的成果都能迅捷地通过生产发布系统或测试发布系统体现出来，也方便测试人员和需求人员及时了解情况，与开发者互动。

【8】需求又发生变化了，新的需求和现在的不一样，但是现在的代码以后可能还有用，我该如何保存现在的代码？

开发者要善于使用分支来保存已经开发的代码，减少重复的工作量。

假设我已经开发完一个叫gather2.jsp页面，此页面没有数据填报功能。这时，需求提出因为客户临时要求，目前需要增加填报数据功能，一段时间以后，用户希望该页面还是只需要汇总数据，不需要填报数据的功能，将来这个功能还要取消。我应该如何将没有数据填报的这部分功能代码保留呢？

并且事情还不仅仅是恢复原来的代码这样简单，因为在开发的过程中会解决一些bug，我们需要使用原来的代码，但是又不想完全回到**readonly这个分支的代码状态，因为我们希望修复bug的代码会被保留。**

以下的解决方案可以作为参考，如果发现了更好的解决思路请邮件发给我一起分享。

本例中首先需要创建一个writedata分支用于开发填报功能，同时创建一个readonly分支用于保存原来只汇总不填报功能。在writedata分支上将代码调试完后，在master分支上将writedata分支merge。

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to readonly**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ ls -l**

**total 3**

**drwxr-xr-x 3 kursk Administ 0 Apr 10 16:47 ear**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 ejbs**

**-rw-r--r-- 1 kursk Administ 22 Apr 10 16:47 gather2.jsp**

**-rw-r--r-- 1 kursk Administ 41 Apr 10 16:47 new.html**

**-rw-r--r-- 1 kursk Administ 2609 Apr 10 16:47 pom.xml**

**drwxr-xr-x 1 kursk Administ 0 Apr 10 16:47 primary-source**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 projects**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 servlets**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git branch readonly**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git checkout -b writedata**

**Switched to a new branch 'writedata'**

**kursk@KURSK-PC /D/gitworkspace/hello (writedata)**

**$ vi gather2.jsp #修改jsp功能为可以填报数据**

**kursk@KURSK-PC /D/gitworkspace/hello (writedata)**

**$ touch write.java #增加一个填报数据用的类文件**

**kursk@KURSK-PC /D/gitworkspace/hello (writedata)**

**$ git add .**

**kursk@KURSK-PC /D/gitworkspace/hello (writedata)**

**$ git commit -a -m "write data function"**

**[writedata c4d3992] write data function**

**2 files changed, 1 insertion(+), 1 deletion(-)**

**create mode 100644 write.java**

**kursk@KURSK-PC /D/gitworkspace/hello (writedata)**

**$ git checkout master**

**Switched to branch 'master'**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git merge writedata**

**Updating 655fda3..c4d3992**

**Fast-forward**

**gather2.jsp | 2 +-**

**write.java | 0**

**2 files changed, 1 insertion(+), 1 deletion(-)**

**create mode 100644 write.java**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git status**

**# On branch master**

**# Your branch is ahead of 'origin/master' by 1 commit.**

**# (use "git push" to publish your local commits)**

**#**

**nothing to commit, working directory clean**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to write data**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ ls -l**

**total 3**

**drwxr-xr-x 3 kursk Administ 0 Apr 10 16:47 ear**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 ejbs**

**-rw-r--r-- 1 kursk Administ 25 Apr 10 16:49 gather2.jsp**

**-rw-r--r-- 1 kursk Administ 41 Apr 10 16:47 new.html**

**-rw-r--r-- 1 kursk Administ 2609 Apr 10 16:47 pom.xml**

**drwxr-xr-x 1 kursk Administ 0 Apr 10 16:47 primary-source**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 projects**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 servlets**

**-rw-r--r-- 1 kursk Administ 0 Apr 10 16:49 write.java**

接下来假设测试组告诉你有些bug需要你解决，这时应先从master上创建一个debug分支，然后在debug分支上解决bug，调试完毕后合并进入Master。

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git checkout -b debug**

**Switched to a new branch 'debug'**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ vi gather2.jsp**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ touch debug.java #解决bug时增加一个类文件是**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ ls -l**

**total 3**

**-rw-r--r-- 1 kursk Administ 0 Apr 10 16:50 debug.java**

**drwxr-xr-x 3 kursk Administ 0 Apr 10 16:47 ear**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 ejbs**

**-rw-r--r-- 1 kursk Administ 39 Apr 10 16:50 gather2.jsp**

**-rw-r--r-- 1 kursk Administ 41 Apr 10 16:47 new.html**

**-rw-r--r-- 1 kursk Administ 2609 Apr 10 16:47 pom.xml**

**drwxr-xr-x 1 kursk Administ 0 Apr 10 16:47 primary-source**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 projects**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 servlets**

**-rw-r--r-- 1 kursk Administ 0 Apr 10 16:49 write.java**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ cat gather2.jsp #修改文件解决bug**

**function to write data**

**debug code**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ git add .**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ git commit -m "debug"**

**[debug d6d192e] debug**

**2 files changed, 2 insertions(+)**

**create mode 100644 debug.java**

**kursk@KURSK-PC /D/gitworkspace/hello (debug)**

**$ git checkout master**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git merge debug**

**Updating c4d3992..d6d192e**

**Fast-forward**

**debug.java | 0**

**gather2.jsp | 2 ++**

**2 files changed, 2 insertions(+)**

**create mode 100644 debug.java**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git status**

**# On branch master**

**# Your branch is ahead of 'origin/master' by 2 commits.**

**# (use "git push" to publish your local commits)**

**#**

**nothing to commit, working directory clean**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ ls -l**

**total 3**

**-rw-r--r-- 1 kursk Administ 0 Apr 10 16:52 debug.java**

**drwxr-xr-x 3 kursk Administ 0 Apr 10 16:47 ear**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 ejbs**

**-rw-r--r-- 1 kursk Administ 39 Apr 10 16:52 gather2.jsp**

**-rw-r--r-- 1 kursk Administ 41 Apr 10 16:47 new.html**

**-rw-r--r-- 1 kursk Administ 2609 Apr 10 16:47 pom.xml**

**drwxr-xr-x 1 kursk Administ 0 Apr 10 16:47 primary-source**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 projects**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 servlets**

**-rw-r--r-- 1 kursk Administ 0 Apr 10 16:49 write.java**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to write data**

**debug code**

当用户提出需要回到原来只需要汇总数据，不要填报数据功能时，先checkout到readonly分支，至少commit一次，创建一个新节点——这非常重要，否则直接Merge原来的代码会被覆盖，而不会产生冲突。然后在readonly分支上merge debug分支，解决有冲突的文件。这样做的好处是：

1、在readonly分支和writedata分支上都被修改的文件会提示有冲突，便于开发者找到哪些文件被修改。

2、在debug分支中增加的文件、被删除的文件、文件中被增加的代码内容和被删除的代码内容都会合并进入readonly分支中，减轻了开发者一部分的工作量。

缺点是：

在writedata分支中增加的文件、被删除的文件、文件中被增加的代码内容和被删除的代码内容也都会合并进入readonly分支中，这些代码需要开发者修改。

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git checkout readonly**

**Switched to branch 'readonly'**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly)**

**$ vi gather2.jsp**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly)**

**$ cat gather2.jsp**

**function to readonly**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly)**

**$ git diff**

**diff --git a/gather2.jsp b/gather2.jsp**

**index 4577289..1afd991 100644**

**--- a/gather2.jsp**

**+++ b/gather2.jsp**

**@@ -1 +1,2 @@**

**function to readonly**

**+ #只是增加一行，什么也没改，只为引起文件冲突**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly)**

**$ git commit -m "no change ,but new node"**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly)**

**$ git merge debug**

**Auto-merging gather2.jsp**

**CONFLICT (content): Merge conflict in gather2.jsp**

**Automatic merge failed; fix conflicts and then commit the result.**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ git status**

**# On branch readonly**

**# You have unmerged paths.**

**# (fix conflicts and run "git commit")**

**#**

**# Changes to be committed:**

**#**

**# new file: debug.java**

**# new file: write.java**

**#**

**# Unmerged paths:**

**# (use "git add <file>..." to mark resolution)**

**#**

**# both modified: gather2.jsp**

**#**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ rm write.java #删除write分支创建的文件**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ cat gather2.jsp**

**<<<<<<< HEAD**

**function to readonly**

**=======**

**function to write data**

**debug code**

**>>>>>>> debug**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ vi gather2.jsp #调试有冲突的文件代码**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ cat gather2.jsp**

**function to readonly**

**debug code**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ git add .**

**warning: You ran 'git add' with neither '-A (--all)' or '--ignore-removal',**

**whose behaviour will change in Git 2.0 with respect to paths you removed.**

**Paths like 'write.java' that are**

**removed from your working tree are ignored with this version of Git.**

**\* 'git add --ignore-removal <pathspec>', which is the current default,**

**ignores paths you removed from your working tree.**

**\* 'git add --all <pathspec>' will let you also record the removals.**

**Run 'git status' to check the paths you removed from your working tree.**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly|MERGING)**

**$ git commit -a -m "revert readonly and debug"**

**[readonly 878def2] revert readonly and debug**

**kursk@KURSK-PC /D/gitworkspace/hello (readonly)**

**$ git checkout master**

**Switched to branch 'master'**

**Your branch is ahead of 'origin/master' by 2 commits.**

**(use "git push" to publish your local commits)**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git merge readonly #master分支再合并readonly**

**Updating d6d192e..878def2**

**Fast-forward**

**gather2.jsp | 2 +-**

**write.java | 0**

**2 files changed, 1 insertion(+), 1 deletion(-)**

**delete mode 100644 write.java**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git status**

**# On branch master**

**# Your branch is ahead of 'origin/master' by 4 commits.**

**# (use "git push" to publish your local commits)**

**#**

**nothing to commit, working directory clean**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ ls -l**

**total 3**

**-rw-r--r-- 1 kursk Administ 0 Apr 10 16:58 debug.java**

**drwxr-xr-x 3 kursk Administ 0 Apr 10 16:47 ear**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 ejbs**

**-rw-r--r-- 1 kursk Administ 36 Apr 10 17:00 gather2.jsp**

**-rw-r--r-- 1 kursk Administ 41 Apr 10 16:47 new.html**

**-rw-r--r-- 1 kursk Administ 2609 Apr 10 16:47 pom.xml**

**drwxr-xr-x 1 kursk Administ 0 Apr 10 16:47 primary-source**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 projects**

**drwxr-xr-x 4 kursk Administ 0 Apr 10 16:47 servlets**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to readonly**

**debug code**

如大家所见，上述做法也只能部分减轻开发者的工作量，开发者在merge后还是必须小心翼翼地检查被修改的文件内容。但是如果checkout readonly分支后，**对debug这部分的内容不合并，则原先已经解决的bug会重新产生，这是一种很不负责的行为，在开发过程中应该杜绝！**

想了解哪些文件涉及修改、新增或删除，可以使用图形化gitk工具。

【9】如何查看日志？如何看版本号？

**GIT投入使用后，每个开发者在RDMS关闭bug时必须说明版本号，以便测试人员了解当前测试的系统是否已经解决该bug，便于测试人员有针对性地做回归测试。**

每次commit时，git都会返回一串字符，即版本号。通过gitk工具也可以查看到历次的版本号，每次版本涉及修改哪些文件和注释。

为了方便查看日志，对开发者做几点要求：

1、多commit，频繁commit。尽量避免一次commit涉及太多内容，这样写注释也需要太多文字。

2、注释简单扼要，说明修改了什么功能，而不仅仅是修改了什么文件。

【10】当GIT remote repository被开发者提交了错误的代码，git管理员如何恢复到原来的版本

如果问题发现得早，在错误代码被push到remote repository后没其他的push，则可以使用revert命令；如果还有其他push，则建议由开发者修改为正确代码后重新push。

注意，在remote repository上的操作都会通过pull操作影响所有人，所以能不做就不做，要做一定要慎重！

以下是revert过程

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to readonly**

**debug code**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ vi gather2.jsp**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git commit -a -m "push error code"**

**[master a3be677] push error code**

**1 file changed, 2 insertions(+)**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git push**

**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)**

**Counting objects: 5, done.**

**Delta compression using up to 4 threads.**

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

**Writing objects: 100% (3/3), 301 bytes | 0 bytes/s, done.**

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

**To d:/git.remote/remote.helloCIServlet/**

**878def2..a3be677 master -> master**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to readonly**

**debug code**

**erro code**

上面的操作中我们push了错误的代码。我们先要用git log找到错误提交的版本号。

**commit a3be6772d6a28af834e776dae901ecb8c2e9004f #根据提交时的注释找到版本号，只需要前6个字符**

**Author: yelei.kursk <kursk.ye@gmail.com>**

**Date: Thu Apr 10 18:02:35 2014 +0800**

**push error code**

**commit 878def23183a6d4e721506cb6b28748dae44ec36**

**Merge: f9ba08b d6d192e**

**Author: yelei.kursk <kursk.ye@gmail.com>**

**Date: Thu Apr 10 17:00:03 2014 +0800**

**revert readonly and debug**

**..................................................................**

**..................................................................**

**..................................................................**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git revert a3be67**

**[master 72b5c7f] Revert "push error code"**

**1 file changed, 2 deletions(-)**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ cat gather2.jsp**

**function to readonly**

**debug code**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**

**$ git push**

**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)**

**Counting objects: 5, done.**

**Delta compression using up to 4 threads.**

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

**Writing objects: 100% (3/3), 318 bytes | 0 bytes/s, done.**

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

**To d:/git.remote/remote.helloCIServlet/**

**a3be677..72b5c7f master -> master**

**kursk@KURSK-PC /D/gitworkspace/hello (master)**