飞林沙

公告

2009年微软最有影响力开发者
2010年微软社区精英

昵称：飞林沙
园龄：6年9个月
荣誉：推荐博客
粉丝：370
关注：22
+加关注

随笔分类(89)

C#相关(11)
C语言相关(18)
Javascript相关(19)
PHP相关(9)
Silverlight相关(4)
Web开发相关(2)
WF相关(9)
乱七八糟的生活(2)
软件工程相关(4)
设计模式相关(8)
项目管理相关(3)

随笔档案(148)

2012年12月 (1)
2012年9月 (1)
2012年5月 (2)
2011年12月 (1)
2011年11月 (1)
2011年9月 (1)
2011年7月 (1)

git rebase小计(转)

git rebase，顾名思义，就是重新定义（re）起点（base）的作用，即重新定义分支的版本库状态。要搞清楚这个东西，要先看看版本库状态切换的两种情况：

1. 我们知道，在某个分支上，我们可以通过git reset，实现将当前分支切换到本分支以前的任何一个版本状态，即所谓的“回溯”。即实现了本分支的“后悔药”。也即版本控制系统的初衷。
2. 还有另一种情况，当我们的项目有多个分支的时候。我们除了在本地开发的时候可能会“回溯”外，也常常会将和自己并行开发的别人的分支修改添加到自 己本地来。这种情况下很常见。作为项目管理员，肯定会不断的合并各个子项目的补丁，并将最新版本推送到公共版本库，而作为开发人员之一，提交自己的补丁之 后，往往需要将自己的工作更新到最新的版本库，也就是说把别的分支的工作包含进来。

举个例子来说吧！假设我们的项目初期只有一个master分支，然后分支上作过两次提交。这个时候系统只有一个master分支，他的分支历史如下：

```
master0（初始化后的版本）
||
v
master1（第一次提交后的版本）
||
v
master2（第二次提交后的版本）
```

这个时候，我们可以通过git reset将master分支（工作目录、工作缓存或者是版本库）切换到master1或者master0版本，这就是前面所说的第一种情况。假设我们这里把master分支通过git reset回溯到了master1状态。那么这个时候系统仍然只有一个master分支，分支的历史如下：

```
master0（初始化后的版本）
||
v
master1（第一次提交后的版本）
```

然后，我们在这里以master1为起点，创建了另一个分支test。那么对于test分支来说，他的第一个版本test0就和master1是同一个版本，此时项目的分支历史如下：

2011年6月 (1)
2011年4月 (1)
2011年3月 (9)
2011年1月 (3)
2010年12月 (1)
2010年11月 (4)
2010年10月 (1)
2010年8月 (1)
2010年7月 (6)
2010年5月 (1)
2010年4月 (16)
2010年3月 (4)
2010年2月 (10)
2010年1月 (24)
2009年12月 (5)
2009年10月 (3)
2009年8月 (1)
2009年6月 (1)
2009年5月 (5)
2009年4月 (16)
2009年3月 (7)
2009年2月 (4)
2009年1月 (9)
2008年11月 (2)
2008年9月 (3)
2008年8月 (2)


```
master0（初始化后的版本）
||
v
master1（第一次提交后的版本）===test0（test分支，初始化自master分支master1状态）
```

这个时候，我们分别对master分支、test分支作两次提交，此时版本库应该成了这个样子：

```
master0（初始化后的版本）
||
v
master1===test0==>test1===>test2
||
v
master2===>master3
```

1. 这个时候，通过第一种git reset的方式，可以将master分支的当前状态（master3）回溯到master分支的master0、master1、master2状态。 也可已将test分支当前状态（test2）回溯到test分支的test0、test1状态，以及test分支的父分支master的master0、 master1状态。
2. 那么。如果我要让test分支从test0到test2之间所有的改变都添加到master分支来，使得master分支包含test分支的所有修改。这个时候就要用到git rebase了。

首先，我们切换到master分支，然后运行下面的命令，即可实现我们的要求：

```
1 git rebase test
```

这个时候，git做了些什么呢？

1. 先将test分支的代码checkout出来，作为工作目录
2. 然后将master分支从test分支创建起的所有改变的补丁，依次打上。如果打补丁的过程没问题，rebase就搞定了
3. 如果打补丁的时候出现了问题，就会提示你处理冲突。处理好了，可以运行git rebase –continue继续直到完成
4. 如果你不想处理，你还是有两个选择，一个是放弃rebase过程（运行git rebase –abort），另一个是直接用test分支的取代当前分支的（git rebase –skip）。

此外，rebase还能够让你修订以前提交，这个功能日后再说。

飞林沙
关注 - 22
粉丝 - 370
荣誉：推荐博客
+加关注

4 0

(请您对文章做出评价)

« 上一篇：SQL语句查询 xml 谢谢老大帮我写的例子哈...
» 下一篇：老话重弹——再谈接口与抽象类

posted @ 2010-08-12 11:12 飞林沙 阅读(58031) 评论(6) 编辑 收藏

评论列表

#1楼 2012-11-01 11:03 today4king

请问这个跟merge有什么区别呢？我现在都是在master上merge其它分支。

支持(0) 反对(0)

#2楼 2014-01-06 10:00 juejiang

@今昭
感觉后面两篇内容讲的挺清楚，我在用git merge的时候，merge之后的commit id是按照时间先后排列的；如果用git rebase，应该是将B的修改全都放到A最新的commit的后面。
http://gitbook.liuhui998.com/4_2.html
http://my.oschina.net/MinGKai/blog/142517

支持(0) 反对(1)

#3楼 2014-01-14 17:39 andycnzh

在master分支上进行rebase一般来说应该是不对的。

master分支默认是公共分支，当有多人协同时，master分支在多个地方都有副本。

如果在master分支上执行git rebase test，会把master分支的提交历史进行修改，可以使用git log仔细观察rebase前后，master分支上的commit hash id。

一旦修改了master的commit hash id，而如果其他人已经基于之前的commit对象做了工作，那么当他拉取master的新的对象时，会需要在合并一次，这样反复下去，会把master分支搞得一团乱。

所以你的示例中master分支到提交对象master2、master3，如果已经推送到远端，并有其他人基于master3对象进行了工作，那么后面的结果将会变得非常的乱。

rebase的含义是把当前分支的提交对象在目标分支上重做一遍，并生成了新的提交对象。

所以如果在master分支上需要对test分支进行rebase，你需要的命令是

```
1 git rebase master test
```

这条命令等价于两条命令的合集

```
1 git checkout test
2 git rebase master
```

永远不要在已经发布到公共仓库的提交对象上做rebase操作，而master分支默认就是公共仓库。

支持(3) 反对(0)

#4楼 2014-02-21 15:09 风月飘飘雨

@andycnzh
高见

支持(0) 反对(0)

#5楼 2014-02-28 12:29 gitxzs

推荐一个微信：Git小助手，很不错的

支持(0) 反对(0)

#6楼 2015-03-25 08:34 Gamain

@andycnzh

引用

在master分支上进行rebase一般来说应该是不对的。

master分支默认是公共分支，当有多人协同时，master分支在多个地方都有副本。

如果在master分支上执行git rebase test，会把master分支的提交历史进行修改，可以使用git log仔细观察rebase前后，master分支上的commit hash id。

一旦修改了master的commit hash id，而如果其他人已经基于之前的commit对象做了工作，那么当他拉取master的新的对象时，会需要在合并一次，这样反复下去，会把master分支搞得一团乱。

所以你的示例中master分支到提交...

学习了

支持(0) 反对(0)