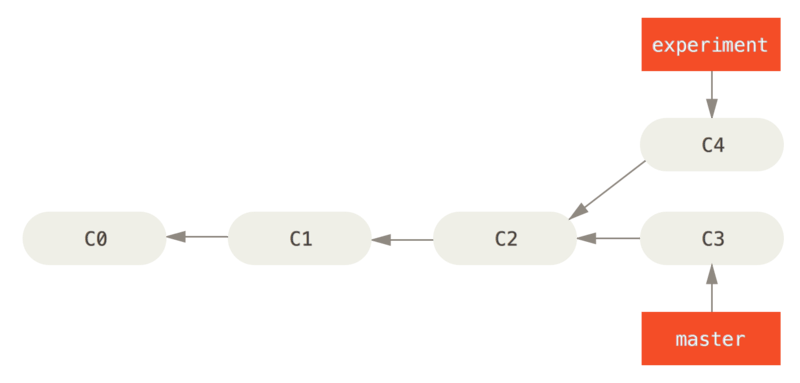
Git实用操作（进阶篇）

1. git rebase vs git merge

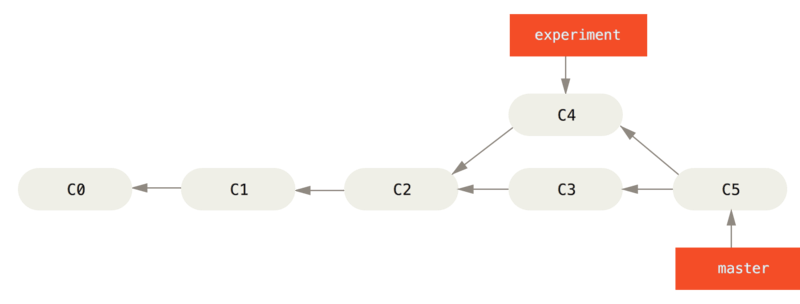
rebase



可以通过git merge 执行合并，变成下图

git checkout master

git merge experiment



小缺点是：提交历史分支分叉太多，不美观。作为一个开源项目，总希望历史是清白干净的。

另一种处理方法是变基 git rebase。

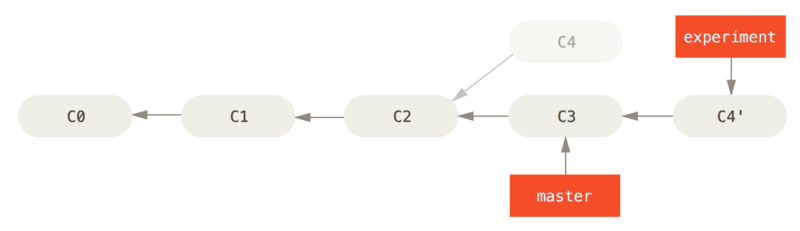
**变基**

**执行如下命令**

**git checkout experiment**

**git rebase master**

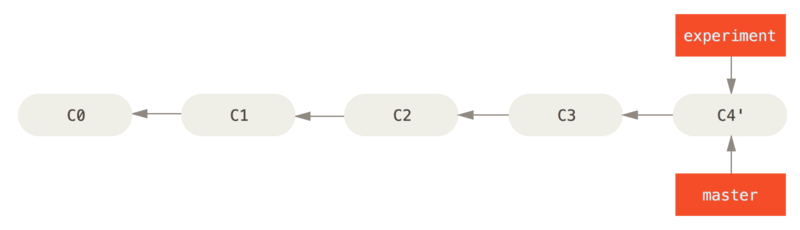
**原理是首先找到这两个分支的最近共同祖先 C2，然后对比当前分支相对于该祖先的历次提交，提取相应的修改并存为临时文件，然后将当前分支指向目标基底 C3, 最后以此将之前另存为临时文件的修改依序应用**



**再执行**

**git checkout master**

**git merge experiment**



1. **修改或者合并提交历史**

**2.1、修改最后一次提交**

**git commit --amend**

**上述命令可以直接修改最后一次提交的内容。**

**另外一种场景：用户新增文件a，做了一次提交，提交消息为：“add file a”。**

**之后用户新增文件b。此时用户可以再次做一次提交，比如提交消息为“add file b”, 但这样做就有两条类似的消息，过于冗余可简化。**

**用户在第二次提交时执行 git commit --amend ，此刻能修改上次提交的消息，保存后两次提交会合并为一次提交。**

**2.2、修改多个提交**

**$ git rebase -i HEAD~3**

**运行这个命令会在文本编辑器上给你一个提交的列表，看起来像下面这样：**

**pick f7f3f6d changed my name a bit**

**pick 310154e updated README formatting and added blame**

**pick a5f4a0d added cat-file**

**# Rebase 710f0f8..a5f4a0d onto 710f0f8**

**#**

**# Commands:**

**# p, pick = use commit**

**# r, reword = use commit, but edit the commit message**

**# e, edit = use commit, but stop for amending**

**# s, squash = use commit, but meld into previous commit**

**# f, fixup = like "squash", but discard this commit's log message**

**# x, exec = run command (the rest of the line) using shell**

**#**

**# These lines can be re-ordered; they are executed from top to bottom.**

**#**

**# If you remove a line here THAT COMMIT WILL BE LOST.**

**#**

**# However, if you remove everything, the rebase will be aborted.**

**#**

**# Note that empty commits are commented out**

**你需要修改脚本来让它停留在你想修改的变更上。 要达到这个目的，你只要将你想修改的每一次提交前面的 ‘pick’ 改为 ‘edit’。 例如，只想修改第三次提交信息，可以像下面这样修改文件：**

**edit f7f3f6d changed my name a bit**

**pick 310154e updated README formatting and added blame**

**pick a5f4a0d added cat-file**

**当保存并退出编辑器时，Git 将你带回到列表中的最后一次提交，把你送回命令行并提示以下信息：**

**$ git rebase -i HEAD~3**

**Stopped at f7f3f6d... changed my name a bit**

**You can amend the commit now, with**

**git commit --amend**

**Once you’re satisfied with your changes, run**

**git rebase --continue**

**这些指令准确地告诉你该做什么。 输入**

**$ git commit --amend**

**修改提交信息，然后退出编辑器。 然后，运行**

**$ git rebase --continue**

**这个命令将会自动地应用另外两个提交，然后就完成了。 如果需要将不止一处的 pick 改为 edit，需要在每一个修改为 edit 的提交上重复这些步骤。 每一次，Git 将会停止，让你修正提交，然后继续直到完成**

**2.3、合并提交**

**通过交互式变基工具，也可以将一连串提交压缩成一个单独的提交。 在变基信息中脚本给出了有用的指令：**

**#**

**# Commands:**

**# p, pick = use commit**

**# r, reword = use commit, but edit the commit message**

**# e, edit = use commit, but stop for amending**

**# s, squash = use commit, but meld into previous commit**

**# f, fixup = like "squash", but discard this commit's log message**

**# x, exec = run command (the rest of the line) using shell**

**#**

**# These lines can be re-ordered; they are executed from top to bottom.**

**#**

**# If you remove a line here THAT COMMIT WILL BE LOST.**

**#**

**# However, if you remove everything, the rebase will be aborted.**

**#**

**# Note that empty commits are commented out**

**如果，指定 “squash” 而不是 “pick” 或 “edit”，Git 将应用两者的修改并合并提交信息在一起。 所以，如果想要这三次提交变为一个提交，可以这样修改脚本：**

**pick f7f3f6d changed my name a bit**

**squash 310154e updated README formatting and added blame**

**squash a5f4a0d added cat-file**

**当保存并退出编辑器时，Git 应用所有的三次修改然后将你放到编辑器中来合并三次提交信息：**

**# This is a combination of 3 commits.**

**# The first commit's message is:**

**changed my name a bit**

**# This is the 2nd commit message:**

**updated README formatting and added blame**

**# This is the 3rd commit message:**

**added cat-file**

**当你保存之后，你就拥有了一个包含前三次提交的全部变更的提交。**

1. **撤销对比 git reset 、git revert、 git checkout**

**3.1、git reset**

**场景一，从暂存移除变更**

**# 创建文件**

**touch a.md**

**# 变更加入暂存**

**git add a.md**

**# 从暂存移除变更**

**git reset**

**git status**

**场景2，取消提交，变更还在暂存区待提交**

**touch b.md**

**git add b.md**

**git commit -m "add b"**

**# 移除最近一次提交，变更退回到暂存区，文件本身不变**

**git reset --soft HEAD~1**

**场景3，取消提交，重暂存移除变更、文件b.md从文件夹删除。一切像未曾发生一样**

**touch b.md**

**git add b.md**

**git commit -m "add b"**

**git reset --hard HEAD~1**

**3.2、git revert**

**通过创建新的提交来“撤销提交”，换句话说并不是真的撤销，而是增加相反变更。**

**git revert HEAD~1**

**git revert 59f39a1**

**git revert 更安全。**

**3.3、git checkout**

**临时切换到老的提交上，只是看看不做事。**

**git checkout HEAD~1**

**git checkout 59f39a1**