


git cherry-pick 详解 —— Git 学习笔记 18

2018年10月28日 12:23:08 [ARM的程序员敲着诗歌的梦](#) 阅读数：225

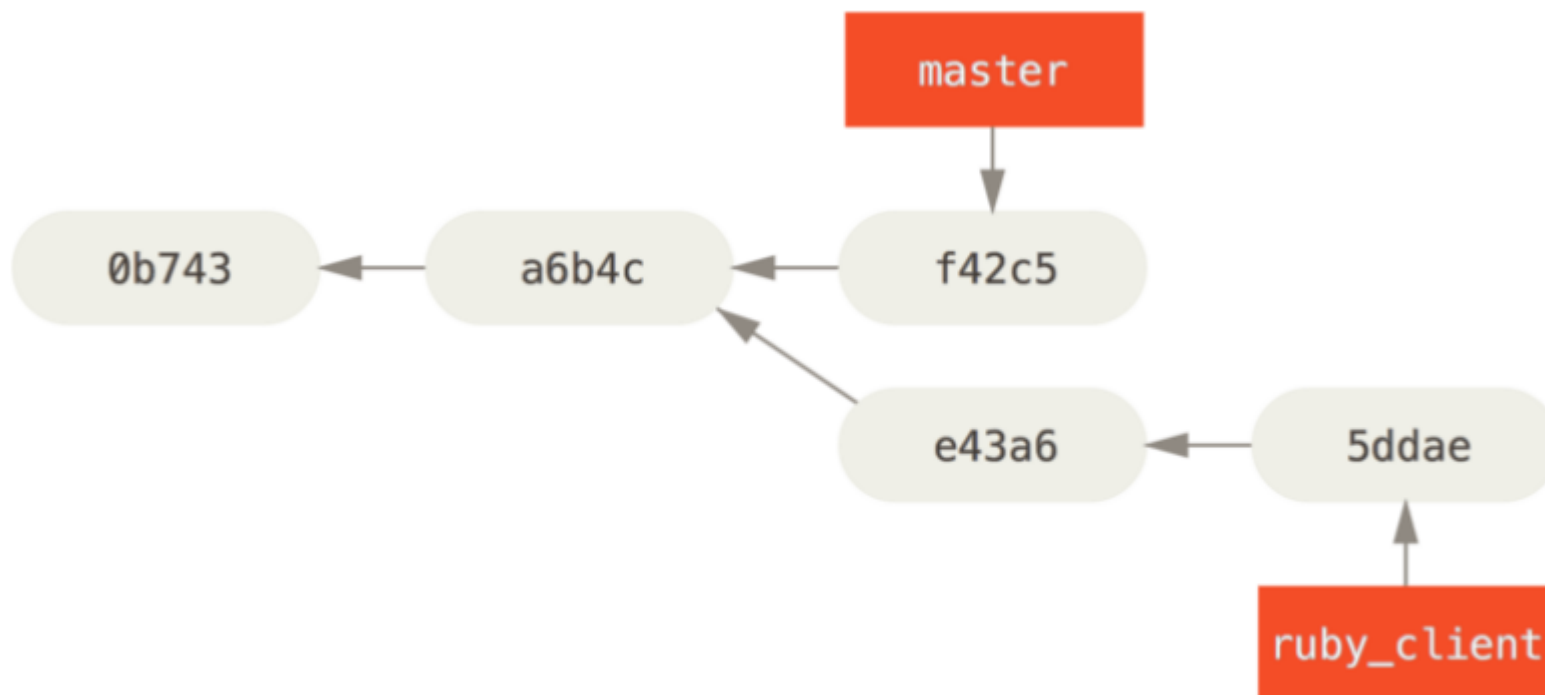
 版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/u013490896/article/details/83473594>

git cherry-pick 详解

初识 git cherry-pick (拣选)

拣选会提取某次提交的补丁，之后尝试将其重新应用到当前分支上。这种方式在你只想引入特性分支中的某个提交时很有用。

假设你的项目提交历史如下：



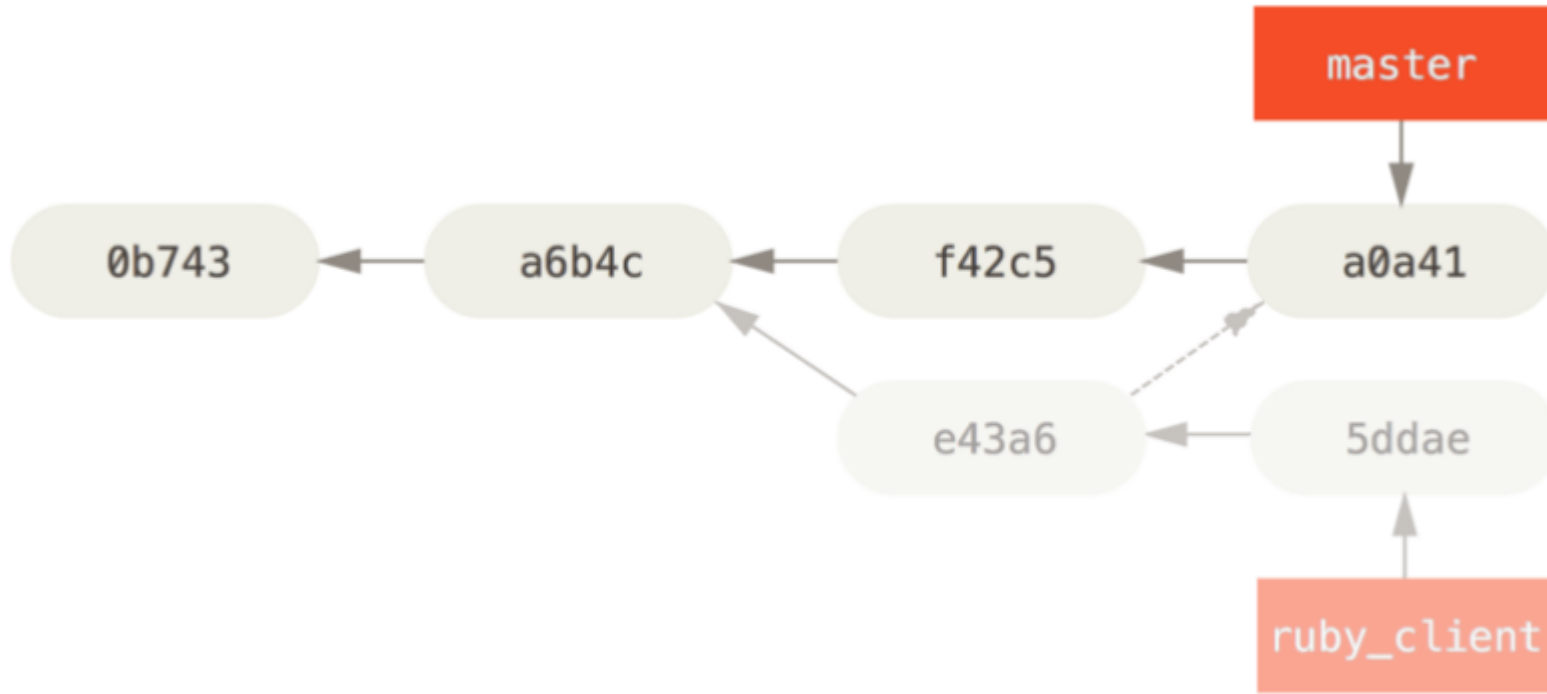
拣选之前的示例历史。

<https://blog.csdn.net/u013490896>

如果你希望将提交 e43a6 拉取到 master 分支，你可以运行：

```
1 # 当前处于 master 分支
2
3 $ git cherry-pick e43a6
4 Finished one cherry-pick.
5 [master]: created a0a41a9: "More friendly message when locking the index
6 fails."
7 3 files changed, 17 insertions(+), 3 deletions(-)
```

这样会拉取和 e43a6 相同的更改，但是因为应用的日期不同，你会得到一个新的提交 SHA-1 值。现在你的历史会变成这样：



拣选特性分支中的一个提交后的历史。

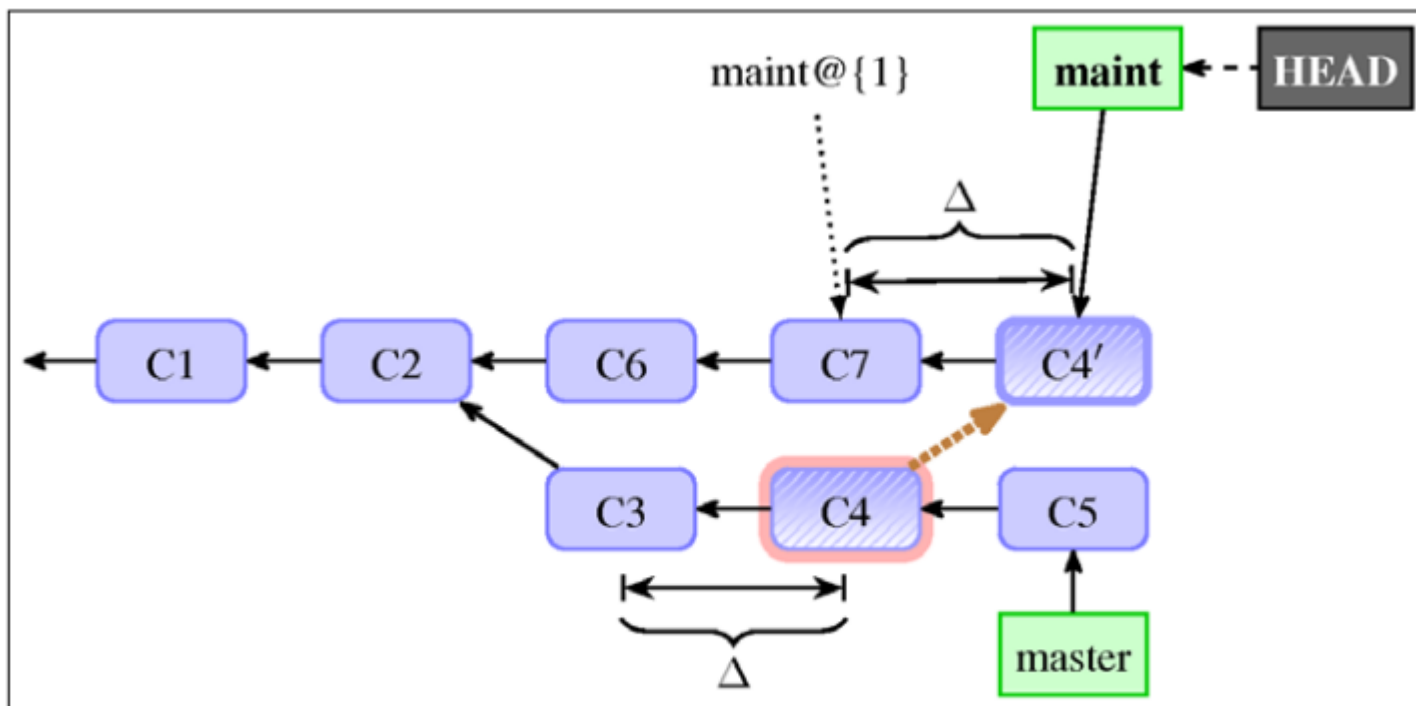
<https://blog.csdn.net/u013490896>

现在你可以删除这个特性分支(ruby_client)，并丢弃不想拉入的提交(5ddae)。

需要说明的是，提取某次提交的“补丁”，这个补丁是基于其父提交的。

下图可以说明：

我们要拣选提交 C4 到 maint 分支（maint 指向 C7），Git 会生成一个补丁($\Delta = C4 - C3$)，然后把 Δ 应用到C7上，也就是说把 C4 对 C3 的变化在 C7 上重放一遍。



Cherry-picking a commit from master to maint. The thick brown dotted line from C4 to C4' denotes copy; it is not a reference.

为何会产生冲突

同 merge 操作一样，拣选操作也可能产生冲突。有人会问：不会吧，打个补丁也能冲突？

当然能。

用 diff 工具生成 patch 时，我们所做的每一处修改都会连同它的“定位信息”（原始文件中的行号、修改处前三行和后三行的原始文本）一并保存到 patch 文件中。patch 被应用时，会在目标文件中寻找“定位信息”，找到后再实施修改。可是，当我们把补丁应用到 C7 上时，有可能找不到那些定位信息了：在 master 分支上，C2 变成了 C3，在 maint 分支上，C2 变成了 C6，又变成了 C7，也许 C3 和 C7 相差越来越远，C3 中的上下文在 C7 中早已面目全非，不见踪迹。于是应用 patch 失败，即发生冲突。

冲突了怎么办

当拣选发生冲突的时候，GIT 会采用三路合并算法。还是以上面的图为例，

当你运行命令 `git cherry-pick C4` 的时候，Local是C7，Remote是C4，Base是C3（即C4的父提交）。总结：

当你运行命令

```
git cherry-pick <commit C>
```

如果冲突了，那么Git会尝试三方合并

- LOCAL: the commit you' re merging on top of (i.e. the HEAD of your branch)
- REMOTE: the commit you' re cherry picking (i.e. commit C)
- BASE: the parent of the commit you' re cherry-picking (i.e. C[^], ie the parent of C)

如果三方合并的时候又冲突了怎么办？那只能靠我们人工解决了。