**Git后悔药 - revert撤销（一）**

**1、Git的三种后悔药**

在Git中后悔药有三种：amend、revert、reset。

* git commit --amend：新的提交覆盖上一次提交的内容。
* git revert：提交一个新的commit，来撤销之前的commit。
* git reset：直接回滚到指定的commit。
* --soft：只回退版本库，工作区和暂存区的内容不回退。
* --mixed（默认）：回退版本库和暂存区，工作区中的内容不回退。
* --hard：工作区，暂存区，版本库中的内容都回退到指定的提交。（危险的）

|  |
| --- |
| 提示：之前我们把git reset命令说完了，git commit --amend命令我们下一篇文章说。这篇文章来说git revert命令。 |

**2、revert命令原理**

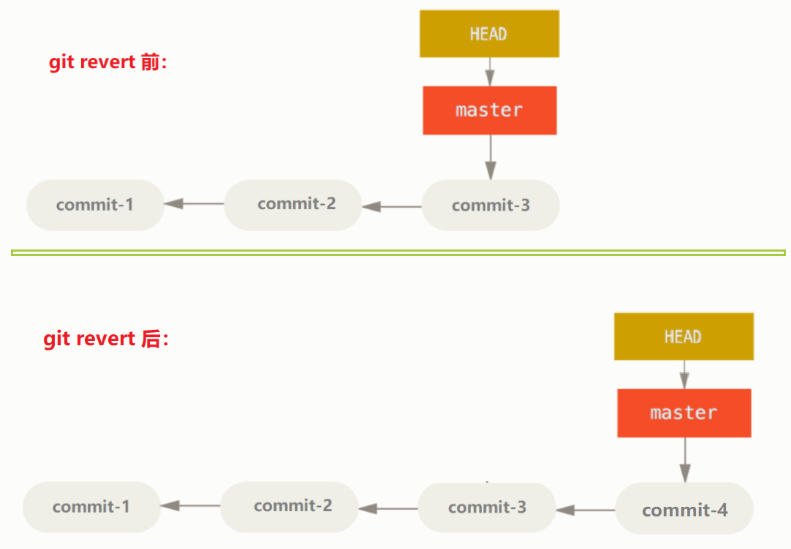
在我们使用Git的操作中，遇到需要回滚代码的情况几乎是难以避免的，而 git revert 命令是一个非常实用的功能，掌握好 git revert 命令的使用是很有必要的。

git revert命令：是用于“反做”某一个版本，以达到撤销该版本的修改的目的。

比如，我们提交了三个版本，突然发现版本二不行（如：有bug），想要撤销版本二，但不想或不需要撤销版本三提交，就可以用 git revert 命令来反做版本二，同时生成新的版本四，这个版本四里会保留版本三的东西，但撤销了版本二的东西。

**但注意：版本二的历史提交记录是不会删除的。**

如下图所示：



|  |
| --- |
| 提示：注意是revert命令是**撤回**某个改动，不是reset命令**撤回到**某个改动。另外，这条命令不会删除任何commit记录，而是会新增一条Revert操作的commit记录（会弹出commit message的编辑窗口）。 |

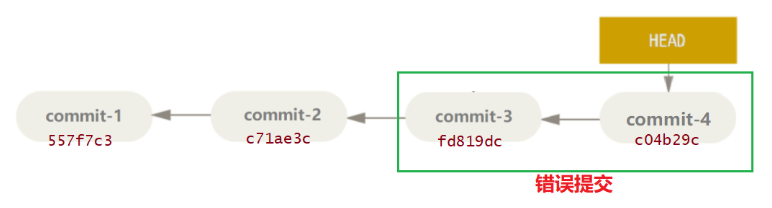
**3、revert命令的使用**

现有一个版本库，其中有4次提交，版本库的历史提交记录如下：

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ git log --oneline c04b29c (HEAD -> master) 第4次提交，新增内容：revert test v4 fd819dc 第3次提交，新增内容：revert test v3 c71ae3c 第2次提交，新增内容：revert test v2 557f7c3 第1次提交，新增readme.txt文件 |

我们发现第三次提交和第四次提交有错误，需要撤销。

如下图：



**（1）移除某次提交的修改**

commit-1和 commit-2 是正常提交，而 commit-3 和 commit-4 是错误提交。现在，我们想把 commit-3 和 commit-4撤销掉。而此时，HEAD 指针指向 commit-4 提交（c04b29c）。我们只需将 HEAD 指针移动到commit-2 提交（c71ae3c），就可以达到目的。

我们一定会想到之前学过的 git reset 命令。执行命令git reset --hard c71ae3c，就可以退回到 commit-2 提交。

采用这种方式回退代码的弊端显而易见，那就是会使 HEAD 指针往回移动，从而会失去之后的提交信息。将来如果突然发现， commit-3 和 commit-4 是多么绝妙的想法，可它们已经早就消失在历史的长河里了（但是通过reflog也能找回来）。

而且，有些公司明令禁止使用 git reset 命令去回退代码，原因与上述一样。所以，我们需要找到一个命令，既可以回退代码，又可以保存错误的提交。这时 git revert 命令就派上用场了。

**命令：git revert <commit>**

演示：

|  |
| --- |
| Bash # 1.撤销第四次提交 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ git revert c04b29c [master d0c8e48] Revert "第5次提交，revert 撤销第4次提交"  1 file changed, 1 deletion(-)  # 之后会弹出一个编辑文本框，来让你写新生成提交的注释，如下图。  # 2.查看版本库历史提交记录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ git log --oneline d0c8e48 (HEAD -> master) Revert "第5次提交，revert 撤销第4次提交" c04b29c 第4次提交，新增内容：revert test v4 fd819dc 第3次提交，新增内容：revert test v3 c71ae3c 第2次提交，新增内容：revert test v2 557f7c3 第1次提交，新增readme.txt文件 # 可以看到第4次提交的commit依然存在。  # 3.查看readme.txt文件内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ cat readme.txt revert test v1 revert test v2 revert test v3 # 可以看到V4 版本内容已经没有了。 |



**这里需要说明一下：**

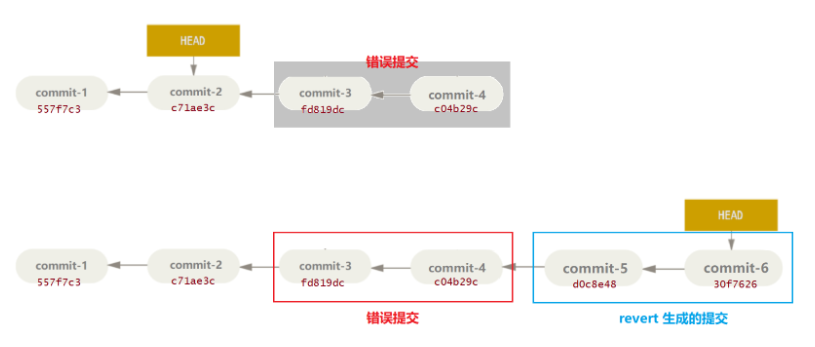
git revert命令的作用通过反做创建一个新的版本，这个版本的内容与我们要回退到的目标版本一样，但是HEAD指针，是指向这个新生成的版本，而不是目标版本。

使用 git revert 命令来实现上述例子的话，我们可以这样做：先 revert commit-4，再 revert commit-3 （**有多个提交需要回退的话需要由新提交到旧提及哦啊进行 revert**）。

我们继续同样的操作步骤，把第三次提交也撤销掉。

|  |
| --- |
| Bash # 1.撤销第3次提交 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ git revert fd819dc [master 30f7626] Revert "第6次提交，revert 撤销第3次提交"  1 file changed, 1 deletion(-)  # 2.查看版本库历史提交记录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ git log --oneline 30f7626 (HEAD -> master) Revert "第6次提交，revert 撤销第3次提交" d0c8e48 Revert "第5次提交，revert 撤销第4次提交" c04b29c 第4次提交，新增内容：revert test v4 fd819dc 第3次提交，新增内容：revert test v3 c71ae3c 第2次提交，新增内容：revert test v2 557f7c3 第1次提交，新增readme.txt文件  # 3.查看readme.txt文件内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/revert\_test (master) $ cat readme.txt revert test v1 revert test v2 |

通过上面练习，我们可以得出，git reset撤销和git revert撤销的区别，如下图所示：



**（2）revert命令说明**

git revert <commit>命令：

1. 这个命令会生成一个新的提交记录，新的提交记录就是把需要移除的提交撤销掉，所生成的新提交。
2. 如果工作区或暂存区还有修改未提交，那么会提示需要先提交或存储起来这些文件。
3. 如果在移除的过程中发生了冲突，可以在解决了冲突后，使用 git add 命令添加修改的冲突文件到暂存区中，在使用 git revert --continue 来继续操作。
4. 如果不想继续操作，当然也可以使用 git revert --abort 命令来停止移除操作，恢复到执行git revert <commit>命令之前的状态。
5. 如果revert移除的过程中出现冲突，需要把这些冲突解决才可以继续操作。我们可以使用 git revert --skip 命令来跳过一个commit的冲突解决。如果后续还有冲突，也同样如此进行跳过，直到全部冲突解决完成。
6. 但是使用git revert --skip 命令跳过的commit，将会在历史提交记录中被删除（reflog命令还是可以看到的），所以git rebase --skip 这个命令慎用。

**拓展思路：**

* git revert HEAD：撤销前一次commit。
* git revert HEAD^：撤销前前一次 commit。

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/0be4e0a70a5a 来源：简书 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。