**Git后悔药 - 通过路径进行重置**

**1、通过路径来重置修改说明**

前面讲述了 git reset 命令的基本用法，不过你还可以给它提供一个作用路径（路径+目录/文件）。

若指定了一个路径，git reset命令将会跳过第 1 步，并且将它的作用范围限定为指定的文件或文件集合。

这样做自然有它的道理，因为 HEAD 只是一个指针，你无法让它同时指向两个提交中各自的一部分。

不过索引和工作目录 可以部分更新，所以回退会继续进行第 2、3 步。

现在，假如我们运行 git reset file.txt ，这其实是 git reset --mixed HEAD file.txt 的简写形式。

他会做如下操作：

1. 移动 HEAD 分支的指向 （已跳过）：实际上命令中写的是HEAD，就是当前commit，所以显示的效果是跳过了这一步。（我的理解）
2. 让暂存区中的file.txt文件进行撤销。
3. 而工作区中的file.txt文件的修改，保持不变。

大家想一下这个场景：我在工作区修改完文件，然后添加到了暂存区中，但是我还没有提交到本地版本库中。这个时候我发现有内容写错了，我此时执行 git reset file.txt命令，即git reset --mixed HEAD file.txt 命令。

就会发生表面现象，只是从暂存区中把file.txt文件，撤回到工作区了，且工作区中的修改保持不变，同时暂存区中其它文件不改变。

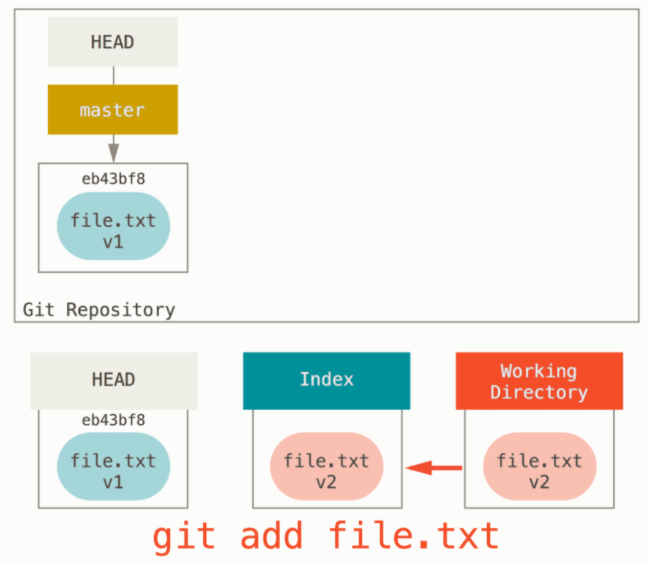
这样的效果，也就相当于是git add filename命令和git reset filename命令，是相互的反操作。

**2、图解说明**

**1）步骤1：**

现在有一个V1版本的file.txt文件，进行修改后，变成V2版本，添加到暂存区中。

就是如下图所示的状态：

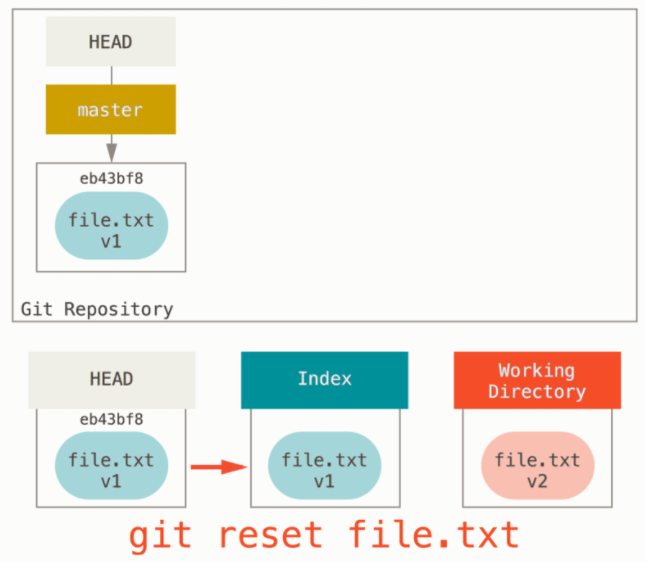


**2）步骤2：**

我发现刚刚修改file.txt文件有错误，需要从暂存去中撤回到工作区中，进行重新修改。

执行命令：git add filename

如下图所示：



说明：

它本质做的是，是将 file.txt 文件从 HEAD 复制到暂存区中，进行单文件的覆盖。

这样就有了"取消暂存文件"的实际效果。 然后再想想 git add 命令所做的事，就会发现它们正好相反。

这就是为什么 git status 命令的输出提示中，会建议运行此命令来取消暂存一个文件。

**3、拓展**

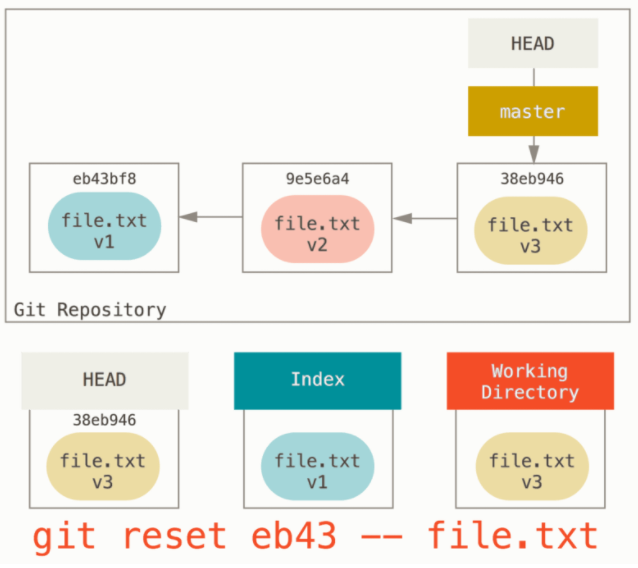
我们可以不让 Git 从 HEAD 拉取数据，而是通过具体指定的某一个提交，来拉取该文件的对应版本。

如下：现在Git的工作目录中，工作区、暂存区和本地版本库中的file.txt文件都是V3版本，我需要把暂存区中的file.txt文件恢复成V1版本。

只需执行命令：git reset eb43bf -- file.txt 即可。

即执行了git reset --mixed eb43bf -- file.txt命令。

如下图所示：



（到这里我也没有，没有具体的底层操作流程是什么。我只知道，只要这样执行命令，工作区和本地版本库中的文件都不会改变，只有暂存区中的文件会回退到指定的版本。）

下面我们通过命令行演示：

|  |
| --- |
| Bash # 1.查看历史提交信息 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git log --oneline e72b30f (HEAD -> master) 第4次提交，新增内容：readme.txt file v4 529ad74 第3次提交，新增内容：readme.txt file v3 1b23cae 第2次提交，新增内容：readme.txt file v2 2612adf 第1次提交，创建readme.txt文件  # 2.查看可回退的历史提交信息 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git reflog e72b30f (HEAD -> master) HEAD@{0}: reset: moving to e72b30f 529ad74 HEAD@{1}: reset: moving to HEAD^ e72b30f (HEAD -> master) HEAD@{2}: commit: 第4次提交，新增内容：readme.txt file v4 529ad74 HEAD@{3}: commit: 第3次提交，新增内容：readme.txt file v3 1b23cae HEAD@{4}: commit: 第2次提交，新增内容：readme.txt file v2 2612adf HEAD@{5}: commit (initial): 第1次提交，创建readme.txt文件  # 3.查看工作目录中文件状态，很干净 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git status On branch master nothing to commit, working tree clean  # 4.查看readme.txt文件内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ cat readme.txt readme.txt file v1 readme.txt file v2 readme.txt file v3 readme.txt file v4  # 5.把暂存区中的readme.txt文件回退到V1版本。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git reset 2612adf -- readme.txt Unstaged changes after reset: M readme.txt  # 6.比较工作区和暂存区中readme.txt文件的差别 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git diff readme.txt diff --git a/readme.txt b/readme.txt index 0d065f4..47b238c 100644 --- a/readme.txt +++ b/readme.txt @@ -1 +1,4 @@  readme.txt file v1 +readme.txt file v2 +readme.txt file v3 +readme.txt file v4  # 7.比较暂存区与本地版本库中readme.txt文件的差别 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git diff --cached readme.txt diff --git a/readme.txt b/readme.txt index 47b238c..0d065f4 100644 --- a/readme.txt +++ b/readme.txt @@ -1,4 +1 @@  readme.txt file v1 -readme.txt file v2 -readme.txt file v3 -readme.txt file v4  # 8.查看暂存区readme.txt文件的内容如下： # 查看暂存区中文件的信息 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git ls-files -s 100644 0d065f480e5ac0200f678ff99a206729d47d808f 0 readme.txt  # 查看tree对象的内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git cat-file -p 0d065f480e5ac0200f678ff99a206729d47d808f readme.txt file v1 |

如上已确认，工作区和本地版本库中都是V4版本，而暂存区回退到V1版本。（这个示例比图片中多一个提交）

那我们查看一下，当前工作目录中文件的状态，还有历史提交信息。

|  |
| --- |
| Bash # 1.查看当前工作目录中文件的状态 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git status On branch master Changes to be committed:  (use "git restore --staged <file>..." to unstage)  modified: readme.txt  Changes not staged for commit:  (use "git add <file>..." to update what will be committed)  (use "git restore <file>..." to discard changes in working directory)  modified: readme.txt  # 2.再次查看历史提交版本 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git log --oneline e72b30f (HEAD -> master) 第4次提交，新增内容：readme.txt file v4 529ad74 第3次提交，新增内容：readme.txt file v3 1b23cae 第2次提交，新增内容：readme.txt file v2 2612adf 第1次提交，创建readme.txt文件  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/learngit (master) $ git reflog e72b30f (HEAD -> master) HEAD@{0}: reset: moving to e72b30f 529ad74 HEAD@{1}: reset: moving to HEAD^ e72b30f (HEAD -> master) HEAD@{2}: commit: 第4次提交，新增内容：readme.txt file v4 529ad74 HEAD@{3}: commit: 第3次提交，新增内容：readme.txt file v3 1b23cae HEAD@{4}: commit: 第2次提交，新增内容：readme.txt file v2 2612adf HEAD@{5}: commit (initial): 第1次提交，创建readme.txt文件 # 发现日志信息和最开始一样，没有变动。 |

我们可以看到，暂存区中有一个已修改状态的readme.txt文件，工作区中有一个未被追踪的readme.txt文件。

同是也还发现了一个现象，就是这样操作不会生成新的commit。不像之前使用git reset命令后，会自动生成一个commit提交。

看到这里我就大概明白了，git reset --mixed eb43bf -- readme.txt命令，它其实做了同样的事情：

1. 因为--mixed后边加了路径（包括文件和目录），所以会跳过第1步，也就是HEAD指针不进行移动。
2. 然后继续执行第2步，把暂存区中的readme.txt文件回退到V1版本。
3. 然后工作区中的readme.txt文件，在使用--mixed参数进行回退的时候，内容不会变动。

所以工作区和暂存区中的readme.txt文件内容不一样，才会出现暂存区中有已修改状态的readme.txt文件，和工作区中有一个未被追踪的readme.txt文件。

这时如果我们直接执行git commit命令进行提交，暂存区中V1版本的readme.txt文件就会被提交到本地版本库中，会新生成一个commit提交，它就会记录一条“将该文件恢复到 **v1** 版本”的更改。

|  |
| --- |
| 提示：在这种情况下，如果把工作区的readme.txt文件，先添加到暂存区中，然后在提交，当前版本库的历史提交信息，是不会有任何变化的。（你可以试试）  分析原因：  我之前一直以为只要有git commit提交操作，就应该有一个commit提交生成。我是这样想的，当我把工作区的readme.txt文件，添加到暂存区后。这个时候工作区，暂存区和本地版本库中readme.txt文件都是一样的，等于我没有做任何的修改。然后我直接提交新的commit，应该这个操作会被Git忽略把。Git也给你提示nothing to commit, working tree clean：没有可提交的内容，暂存区中的树对象没改变。  说明：还有一点同 git add 一样，就是 reset 命令也可以接受一个 --patch 选项，来一块一块地取消暂存的内容。 这样你就可以根据选择，来取消暂存或恢复内容了。  参考：[https://git-scm.com/book/zh/v2/](https://links.jianshu.com/go?to=https%3A%2F%2Fgit-scm.com%2Fbook%2Fzh%2Fv2%2F) |

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/f340de60d6aa 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。