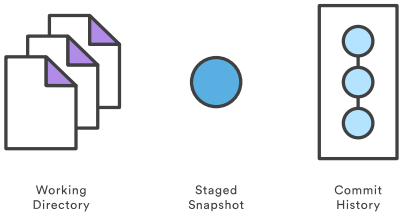
代码回滚：git reset、git checkout和git revert区别和联系

2016-09-20 22:32 by houpy, 13750 阅读, 3 评论, 收藏, 编辑

`git reset`、`git checkout`和`git revert`是你的Git工具箱中最有用的一些命令。它们都用来撤销代码仓库中的某些更改，而前两个命令不仅可以作用于提交，还可以作用于特定文件。

因为它们非常相似，所以我们经常会搞混，不知道什么场景下该用哪个命令。在这篇文章中，我们会比较`git reset`、`git checkout`和`git revert`最常见的用法。希望你在看完后能游刃有余地使用这些命令来管理你的仓库。

The main components of a Git repository



Git仓库有三个主要组成——工作目录，缓存区和提交历史。这张图有助于理解每个命令到底产生了哪些影响。当你阅读的时候，牢记这张图。

提交层面的操作

你传给`git reset`和`git checkout`的参数决定了它们的作用域。如果你没有包含文件路径，这些操作对所有提交生效。我们这一节要探讨的就是提交层面的操作。注意，`git revert`没有文件层面的操作。

Reset

在提交层面上，reset将一个分支的末端指向另一个提交。这可以用来移除当前分支的一些提交。比如，下面这两条命令让hotfix分支向后回退了两个提交。

```
git checkout hotfix
git reset HEAD~2
```

hotfix分支末端的两个提交现在变成了悬挂提交。也就是说，下次Git执行垃圾回收的时候，这两个提交会被删除。换句话说，如果你想扔掉这两个提交，你可以这么做。reset操作如下图所示：

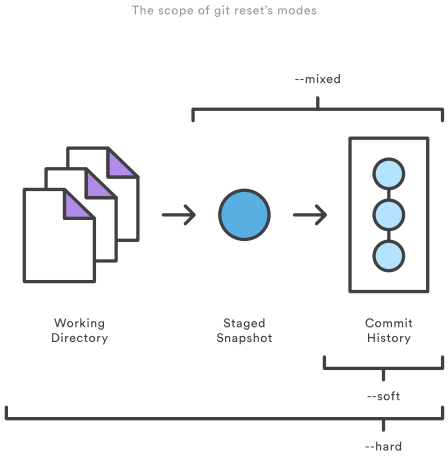
如果你的更改还没有共享给别人，`git reset`是撤销这些更改的简单方法。当你开发一个功能的时候发现『糟糕，我做了什么？我应该重新来过！』时，reset就像是go-to命令一样。

除了在当前分支上操作，你还可以通过传入这些标记来修改你的缓存区或工作目录：

- --soft – 缓存区和工作目录都不会被改变
- --mixed – 默认选项。缓存区和你指定的提交同步，但工作目录不受影响

- --hard – 缓存区和工作目录都同步到你指定的提交

把这些标记想成定义`git reset`操作的作用域就容易理解多了。



这些标记往往和HEAD作为参数一起使用。比如，`git reset --mixed HEAD` 将你当前的改动从缓存区中移除，但是这些改动还留在工作目录中。另一方面，如果你想完全舍弃你没有提交的改动，你可以使用`git reset --hard HEAD`。这是`git reset`最常用的两种用法。

当你传入HEAD以外的其他提交的时候要格外小心，因为reset操作会重写当前分支的历史。正如Rebase黄金法则所说的，在公共分支上这样做可能会引起严重的后果。

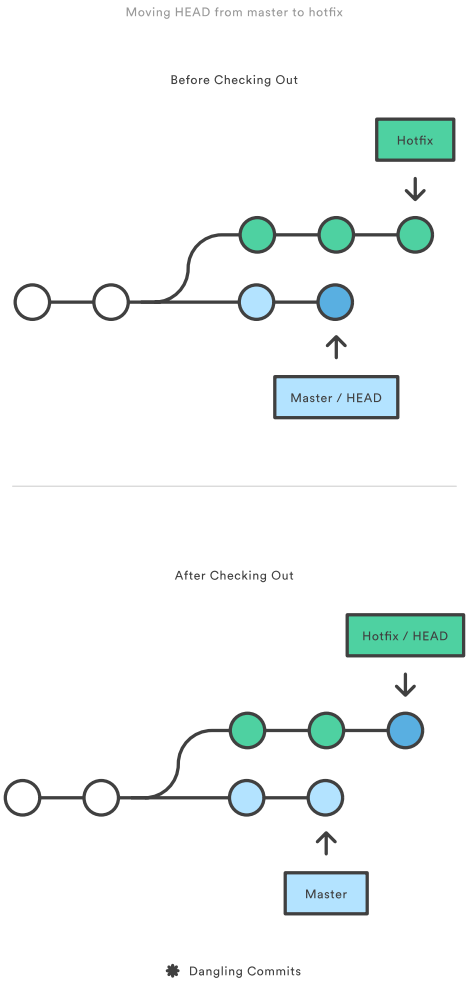
Checkout

你应该已经非常熟悉提交层面的`git checkout`。当传入分支名时，可以切换到那个分支。

```
git checkout hotfix
```

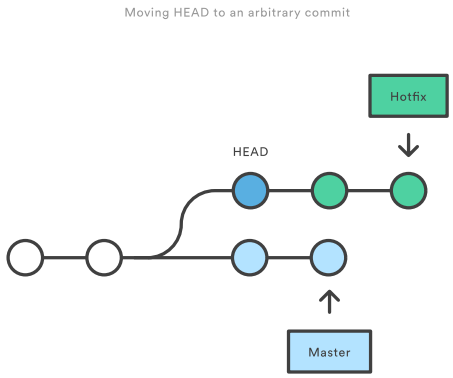
上面这个命令做的不过是将HEAD移到一个新的分支，然后更新工作目录。因为这可能会覆盖本地的修改，Git强制你提交或者缓存工作目录中的所有更改，不然在checkout的时候这些更改都会丢失。和`git reset`不一样的是，`git checkout`没有移动这些分支。

revert区别和联系(13750)
2. sh脚本异常：/bin/sh^M:bad interpreter: No such file or directory(6922)
3. 精确获取函数运行时间，精确到微秒(1490)
4. 在github分支上上传空文件夹(476)
5. Linux -- objdump二进制文件比较(438)



除了分支之外，你还可以传入提交的引用来checkout到任意的提交。这和checkout到另一个分支是完全一样的：把HEAD移动到特定的提交。比如，下面这个命令会checkout到当前提交的祖父提交。

```
git checkout HEAD~2
```



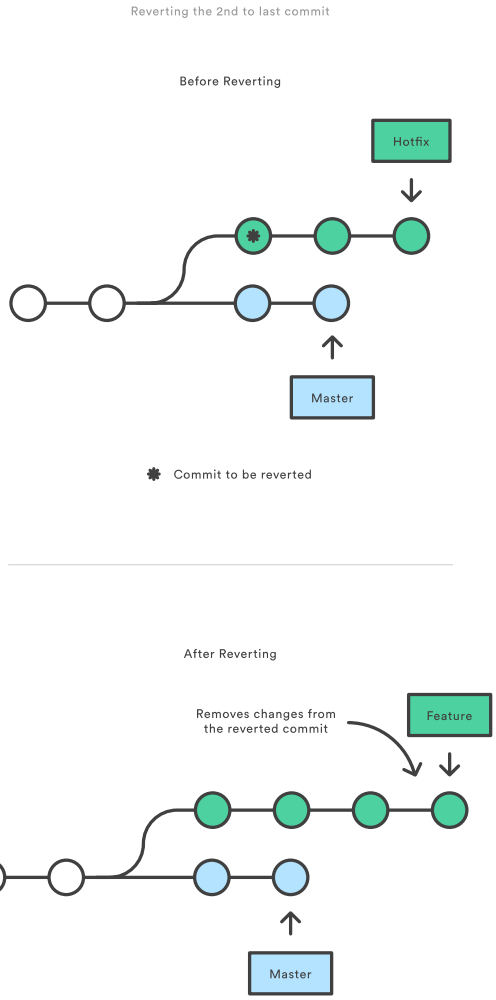
这对于快速查看项目旧版本来说非常有用。但如果你当前的HEAD没有任何分支引用，那么这会造成HEAD分离。这是非常危险的，如果你接着添加新的提交，然后切换到别的分支之后就没办法回到之前添加的这些提交。因此，在为分离的HEAD添加新的提交的时候你应该创建一个新的分支。

Revert

Revert撤销一个提交的同时会创建一个新的提交。这是一个安全的方法，因为它不会重写提交历史。比如，下面的命令会找出倒数第二个提交，然后创建一个新的提交来撤销这些更改，然后把这个提交加入项目中。

```
git checkout hotfix
git revert HEAD~2
```

如下图所示：



相比`git reset`，它不会改变现在的提交历史。因此，`git revert`可以用在公共分支上，`git reset`应该用在私有分支上。

你也可以把`git revert`当作撤销已经提交的更改，而`git reset HEAD`用来撤销没有提交的更改。

就像`git checkout` 一样，`git revert` 也有可能会重写文件。所以，Git会在你执行revert之前要求你提交或者缓存你工作目录中的更改。

文件层面的操作

`git reset`和`git checkout` 命令也接受文件路径作为参数。这时它的行为就大为不同了。它不会作用于整份提交，参数将它限制于特定文件。

Reset

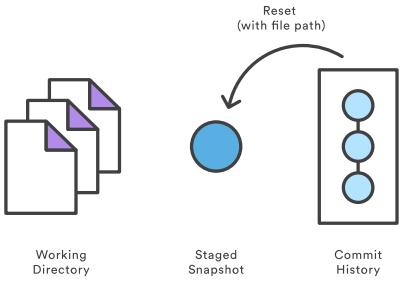
当检测到文件路径时，`git reset` 将缓存区同步到你指定的那个提交。比如

，下面这个命令会将倒数第二个提交中的foo.py加入到缓存区中，供下一个提交使用。

```
git reset HEAD~2 foo.py
```

和提交层面的`git reset`一样，通常我们使用HEAD而不是某个特定的提交。运行`git reset HEAD foo.py` 会将当前的foo.py从缓存区中移除出去，而不会影响工作目录中对foo.py的更改。

Moving a file from the commit history into the staged snapshot

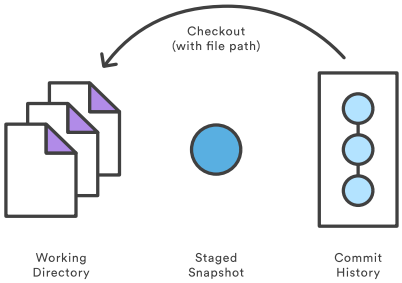


--soft、--mixed和--hard对文件层面的`git reset`毫无作用，因为缓存区中的文件一定会变化，而工作目录中的文件一定不变。

Checkout

Checkout一个文件和带文件路径`git reset` 非常像，除了它更改的是工作目录而不是缓存区。不像提交层面的checkout命令，它不会移动HEAD引用，也就是你不会切换到别的分支上去。

Moving a file from the commit history into the working directory



比如，下面这个命令将工作目录中的foo.py同步到了倒数第二个提交中的foo.py。

```
git checkout HEAD~2 foo.py
```

和提交层面相同的是，它可以用来检查项目的旧版本，但作用域被限制到了特定文件。

如果你缓存并且提交了checkout的文件，它具备将某个文件回撤到之前版本的效果。注意它撤销了这个文件后面所有的更改，而`git revert` 命令只撤销某个特定提交的更改。

和`git reset` 一样，这个命令通常和HEAD一起使用。比如`git checkout HEAD foo.py`等同于舍弃foo.py没有缓存的更改。这个行为和`git reset HEAD --hard`很像，但只影响特定文件。

总结

你现在已经掌握了Git仓库中撤销更改的所有工具。`git reset`、`git checkout`、和`git revert`命令比较容易混淆，但当你想起它们对工作目录、缓存区和提交历史的不同影响，就会容易判断现在应该用哪个命令。

下面这个表格总结了这些命令最常用的使用场景。记得经常对照这个表格，因为你使用Git时一定会经常用到。

命令	作用域	常用情景
git reset	提交层面	在私有分支上舍弃一些没有提交的更改
git reset	文件层面	将文件从缓存区中移除
git checkout	提交层面	切换分支或查看旧版本
git checkout	文件层面	舍弃工作目录中的更改
git revert	提交层面	在公共分支上回滚更改
git revert	文件层面	（然而并没有）

« 上一篇：精确获取函数运行时间，精确到微秒
» 下一篇：VS2010 LINK1123:failure during conversion to COFF:file invalid or corrupt

标签: git

#1楼 猫大东
2017-03-30 10:51
很清晰，赞~!
支持(0) 反对(0)

#2楼 simple0812
2017-10-25 09:47
很清晰，赞~!
支持(0) 反对(0)

#3楼 白火羽
2017-12-20 17:27
关于git如何使用的文章越多，说明git越不好用。。。
支持(1) 反对(0)

最新IT新闻:
· 内忧外患，现在是滴滴上市的好时机吗?
· 光速打脸？Facebook免费上网服务在缅甸等地停止运营
· Google Clips限时优惠 现只需要199美元
· 人工智能为“歌唱北京”投稿 微软小冰填词
· WhatsApp创始人潇洒出局：哥去玩保时捷啦
» 更多新闻...

最新知识库文章:
· 如何识别人的技术能力和水平?
· 写给自学者的入门指南
· 和程序员谈恋爱
· 学会学习
· 优秀技术人的管理陷阱
» 更多知识库文章...