

Blockchain Hackathon

2018 正式启动
火热报名中!

· 能源，金融，医疗，慈善等多个领域的创新应用
· 北京 / 广州 / 杭州

首页 问答 专栏 讲堂 标签 发现 ▾ 🔍

立即登录 免费注册

git的reset和checkout的区别

赞 9

收藏 56

git chanjarster 2016年08月05日发布

9.8k 次浏览

在讲git的reset和checkout的区别之前，不得不说说HEAD、Index、Working Directory三个区域。

HEAD、Index、Working Directory

Git里有三个区域很重要

1. HEAD 指向最近一次commit里的所有snapshot
2. Index 缓存区域，只有Index区域里的东西才可以被commit
3. Working Directory 用户操作区域

下图解释了这三个区域的状态的变化过程：



初始状态

当你checkout分支的时候，git做了这么三件事情

1. 将HEAD指向那个分支的最后一次commit
2. 将HEAD指向的commit里所有文件的snapshot替换掉Index区域里原来的内容
3. 将Index区域里的内容填充到Working Directory里

所以你可以发现，HEAD、Index、Working Directory这个时候里的内容都是一模一样的。

注意：一般会误解为，Index中的内容是空的，只有git add后才会有东西。实际上不是，Index里一直是有东西的。

所以，Git的所有操作就是对这三个区域的状态（或内容）的操作。

Changed

如果你在Working Directory里修改了文件，git会发现Working Directory里的内容和Index区域里的内容不一致了。

这个时候git status的结果是：

```
# Changes not staged for commit:
```

Staged

一个文件仅仅changed是不能被commit的，Git要求只能提交Index里的东西。

所以需要git add。这个命令的意思是，把Changed的文件的内容同步到Index区域里。这样Working Directory和Index区域的内容就一致了。这个过程被称之为stage

这个时候git status的结果是：

```
# Changes to be committed:
```

Committed

最后，你就可以提交了

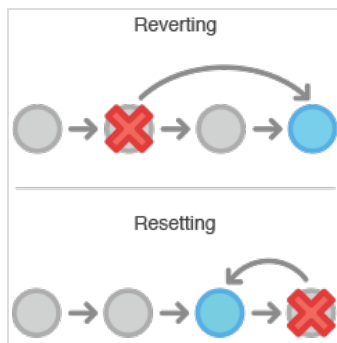
```
git commit
```

这样，就把HEAD的状态和Index以及Working Directory形成一致了。

reset

reset是用来修改提交历史的，想象这种情况，如果你在2天前提交了一个东西，突然发现这次提交是有问题的。

这个时候你有两个选择，要么使用git revert（推荐），要么使用git reset。



上图可以看到git reset是会修改版本历史的，他会丢掉一些版本历史。

而git revert是根据那个commit逆向生成一个新的commit，版本历史是不会被破坏的。

已经push到远程仓库的commit不允许reset

上面已经讲了，git reset是会丢弃掉commit的。

如果commit已经被push到远程仓库上了，也就意味着其他开发人员就可能基于这个commit形成了新的commit，这时你去reset，就会造成其他开发人员的提交历史莫名其妙的丢失，或者其他灾难性的后果。

因此，一旦commit已经被push到远程仓库，那么是坚决不允许去reset它的。

不带文件参数的reset

前面章节已经说道Git有三个区域，Git的所有操作实际上是在操作这三个区域的状态（或内容）。

git reset配合不同的参数，对这三个区域会产生不同的影响。

reset实际上有3个步骤，根据不同的参数可以决定执行到哪个步骤(`--soft` , `--mixed` , `--hard`)。

- 1. 改变HEAD所指向的commit(`--soft`)
- 2. 执行第1步，将Index区域更新为HEAD所指向的commit里包含的内容(`--mixed`)
- 3. 执行第1、2步，将Working Directory区域更新为HEAD所指向的commit里包含的内容(`--hard`)

注意

`--mixed` 是默认参数，也就是说执行reset的时候不给就认为是 `--mixed` 。

下表说明了三种形式的git reset所产生的不同效果。

target代表想要将git指向到哪个commit

working index HEAD target				working index HEAD			
-----				-----			
A	B	C	D	--soft	A	B	D
				--mixed	A	D	D
				--hard	D	D	D
				--merge	(disallowed)		

working index HEAD target				working index HEAD			
-----				-----			
A	B	C	C	--soft	A	B	C
				--mixed	A	C	C
				--hard	C	C	C
				--merge	(disallowed)		

带文件参数的reset

上面讲到的git reset实际上不带参数的，如果带上文件参数，那么效果会是怎样的？

- 1. HEAD不会动
- 2. 将那个commit的snapshot里的那个文件放到Index区域中

需要注意的是带文件参数的git reset没有--hard, --soft这两个参数。只有--mixed参数。

unstage

下面这两个命令是一样的，都是reset到HEAD上。

```
git reset file.txt
git reset --mixed HEAD file.txt
```

这个例子的意义在于，unstage file，仔细想一想是不是这样？当你把一个文件stage到Index区域里后后悔了，那么只需要把Index区域里的这个文件恢复到最近一次commit的状态（也就是HEAD），那就相当于unstage了。

恢复到历史版本

下面这个命令就是将某个文件恢复到历史版本上。

```
reset eb43bf file.txt
```

这个例子的意思在于，把某个文件恢复到Index区域里，然后直接commit，这样就等于把这个文件恢复到历史版本

了，这样依赖你都不需要去改动Working Directory了。

checkout

前面讲到checkout是会修改HEAD的指向，变更Index区域里的内容，修改Working Directory里的内容。

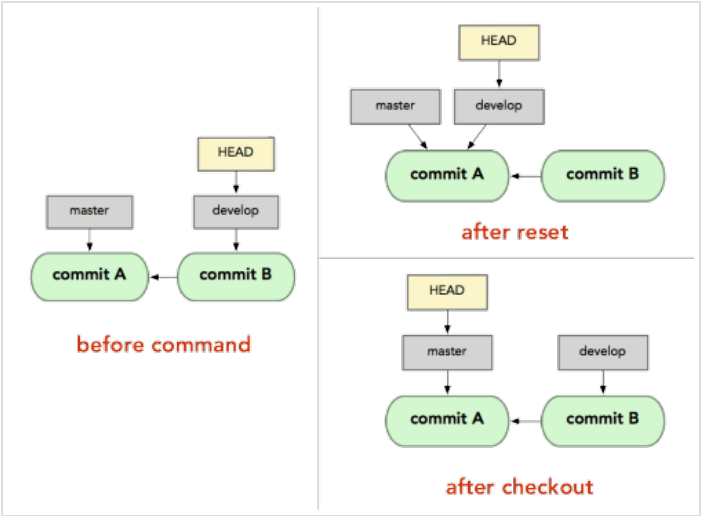
这看上去很像 `reset --hard`，但和 `reset --hard` 相比有两个重要的差别

- 1. reset会把working directory里的所有内容都更新掉
- 2. checkout不会去修改你在Working Directory里修改过的文件
- 3. reset把branch移动到HEAD指向的地方
- 4. checkout则把HEAD移动到另一个分支

第二个区别可能有点难以理解，举例来说：假设你有两个分支master和develop，这两个分支指向不一样的commit，我们现在在develop分支上（HEAD指向的地方）

如果我们 `git reset master`，那么develop就会指向master所指向的那个commit。

如果我们 `git checkout master`，那么develop不会动，只有HEAD会移动。HEAD会指向master。看图：



带文件参数

当执行`git checkout [branch] file`时，checkout干了这件事情：

- 1. 更新了index区域里file文件的内容
- 2. 更新了working directory里file文件的内容

总结reset和checkout

	head	index	work dir	wd safe
Commit Level				
<code>reset --soft [commit]</code>	REF	NO	NO	YES
<code>reset [commit]</code>	REF	YES	NO	YES
<code>reset --hard [commit]</code>	REF	YES	YES	NO
<code>checkout [commit]</code>	HEAD	YES	YES	YES

File Level				
reset (commit) [file]	NO	YES	NO	YES
checkout (commit) [file]	NO	YES	YES	NO

2016年08月05日发布 ...

赞赏支持

赞 | 9

收藏 | 56

如果觉得我的文章对你有用，请随意赞赏



你可能感兴趣的文章

- [windows命令行下 git reset -hard HEAD^无法 正确执行](#) 1.3k 浏览
- [git撤销修改操作](#) 276 浏览
- [git的撤销操作：reset、checkout和revert](#) 2 收藏, 1k 浏览



本作品采用 [署名-非商业性使用-禁止演绎 4.0 国际许可协议](#) 进行许可。

3 条评论

默认排序 时间排序

qq592304796 · 2017年04月01日

可以的。

赞 回复

Object · 2017年06月05日

很棒 写的很不错

赞 回复

billiards · 2017年08月11日


大神辛苦了，恩泽小白

赞 回复



文明社会，理性评论

发布评论



chanjarster

2.9k 声望

发布于专栏


颇忒脱

Java方面的专栏

19 人关注

关注作者

关注专栏

产品	资源	商务	关于	关注	条款
热门问答	每周精选	人才服务	关于我们	产品技术日志	服务条款
热门专栏	用户排行榜	企业培训	加入我们	社区运营日志	内容许可
热门讲堂	徽章	活动策划	联系我们	市场运营日志	
最新活动	帮助中心	广告投放		团队日志	
技术圈	声望与权限	区块链解决方案		社区访谈	
找工作	社区服务中心	合作联系			
移动客户端	开发手册				扫一扫下载 App