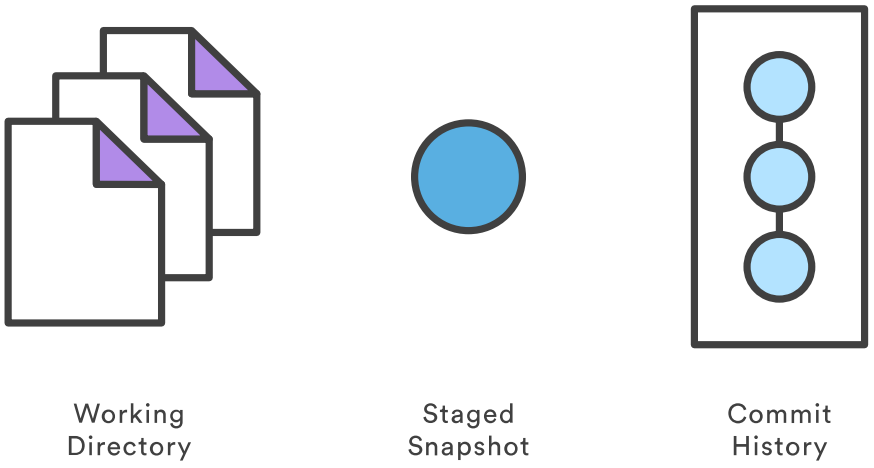
Reset、Checkout和Revert

> BY 童仲毅(geeeeeeeeek@github)
>
> 这是一篇在原文(BY atlassian)基础上演绎的译文。除非另行注明，页面上所有内容采用知识共享-署名(CC BY 2.5 AU)协议共享。

`git reset`、`git checkout`和`git revert`是你的Git工具箱中最有用的命令。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commit，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，stage缓存和提交历史。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。

Commit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commit生效。我们这一节要探讨的就是commit层面的操作。注意`git revert`没有文件层面的操作。

Reset

在commit层面上，reset将一个分支的末端指向另一个commit。这可以用来移除当前分支的一些

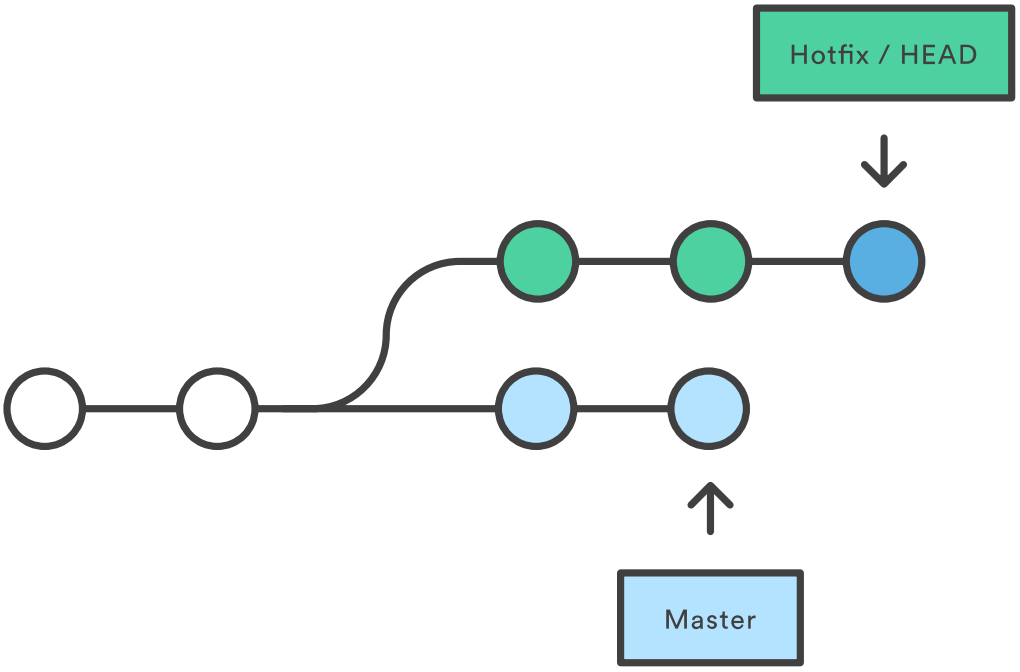
commit。比如，下面这两条命令让hotfix分支向后回退了两个commit。

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

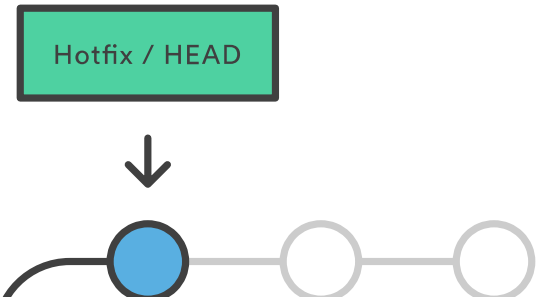
hotfix分支末端的两个commit现在变成了悬挂commit。也就是说，下次Git执行垃圾回收的时候，这两个commit会被删除。换句话说，如果你想扔掉这两个commit，你可以这么做。reset操作如下图所示：

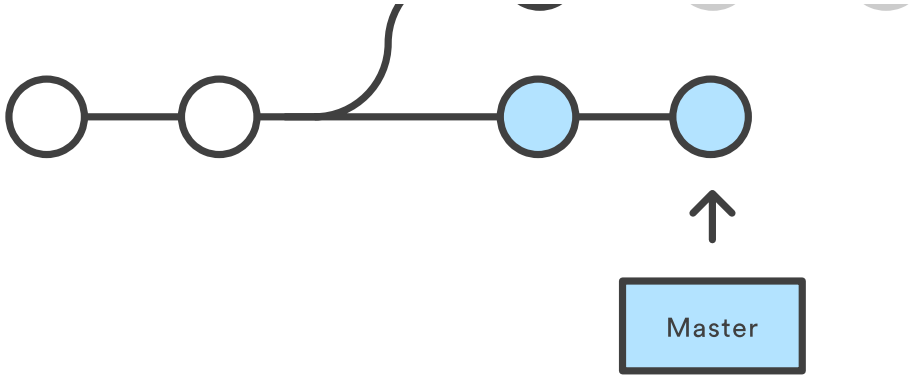
Resetting the hotfix branch to HEAD-2

Before Resetting



After Resetting





Dangling Commits

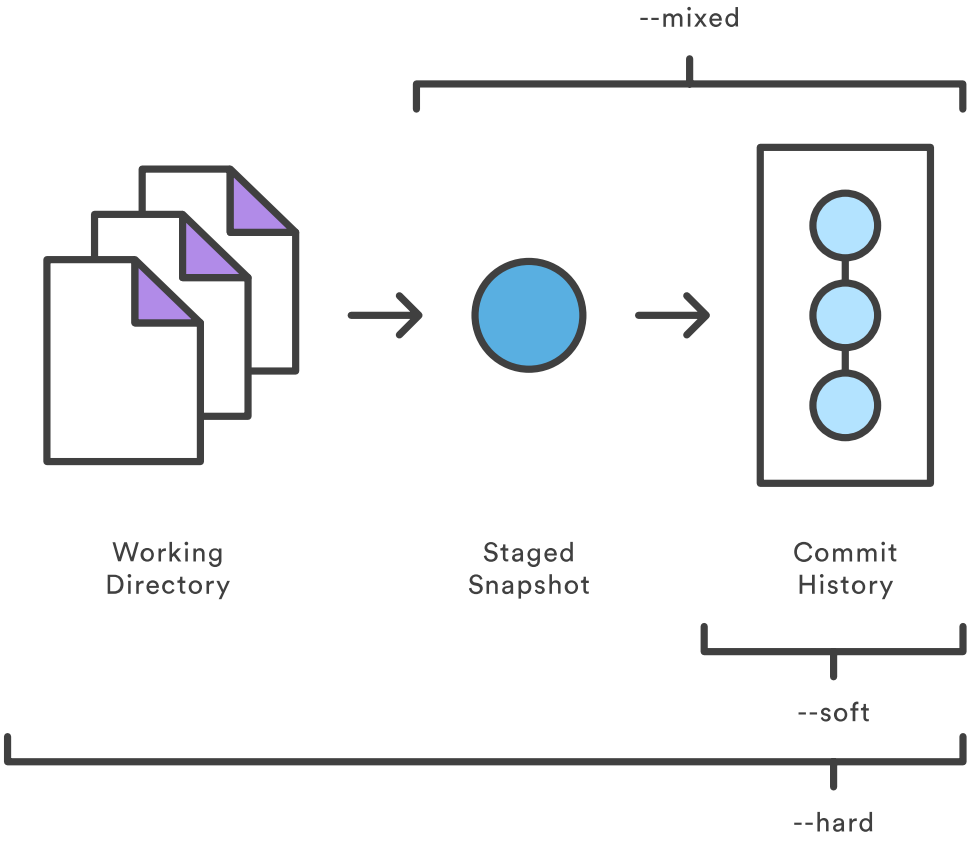
如果你的更改还没有共享给别人，`git reset`是撤销这些更改的简单方法。当你开发一个功能的时候发现“糟糕，我做了什么？我应该重新来过”，这时候reset就像是go-to命令一样。

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stage缓存或工作目录：

- --soft – stage缓存和工作目录不会被改变
- --mixed – 默认选项。stage缓存和你指定的commit同步，但工作目录不受影响
- --hard – stage缓存和工作目录都同步到你指定的commit

把这些标记想成定义`git reset`操作的作用域就容易理解多了。

The scope of git reset's modes



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stage缓存中移除，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commit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commit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。

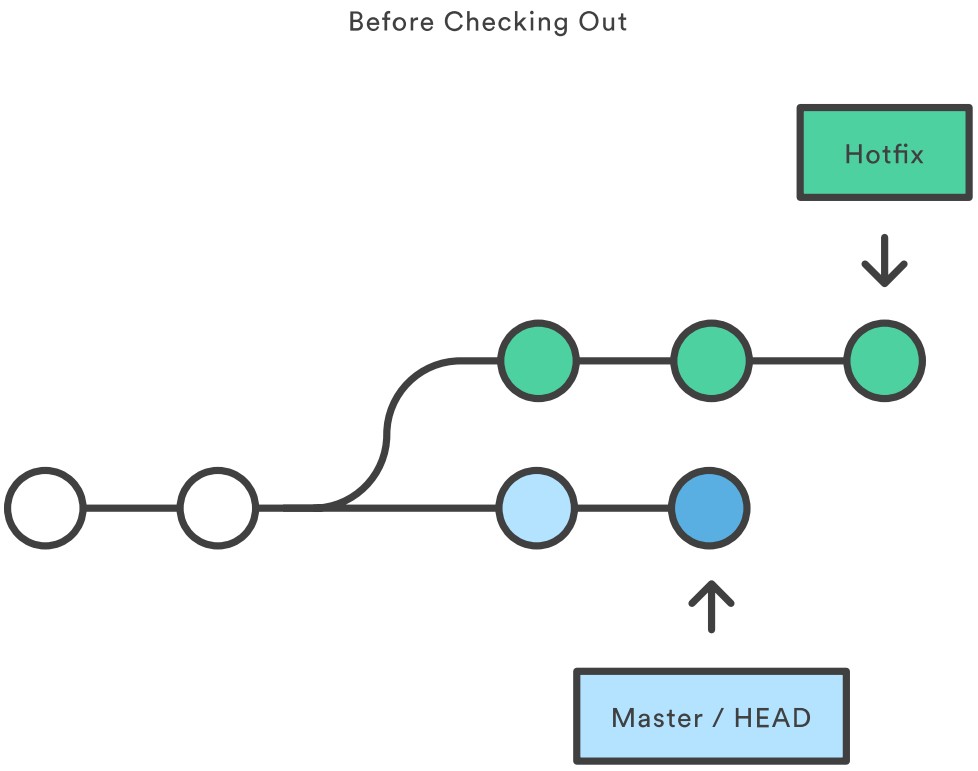
Checkout

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

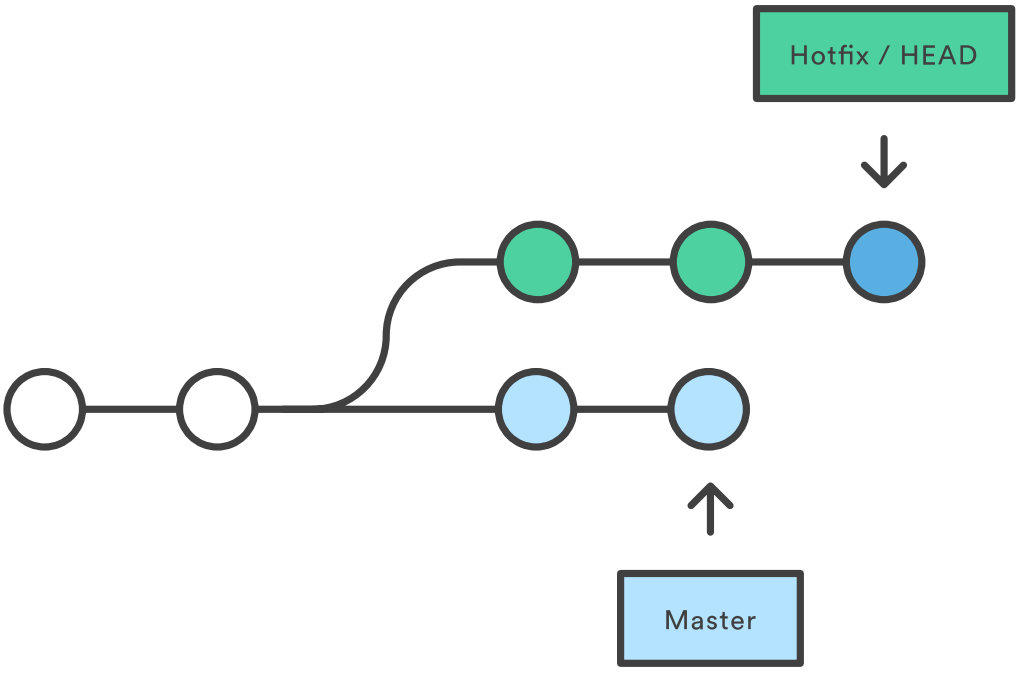
```
git checkout hotfix
```

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commit或者stash工作目录中的所有更改，不如在checkout的时候这些更改都会丢失。不像`git reset`，`git checkout`没有移动这些分支。

Moving HEAD from master to hotfix



After Checking Out

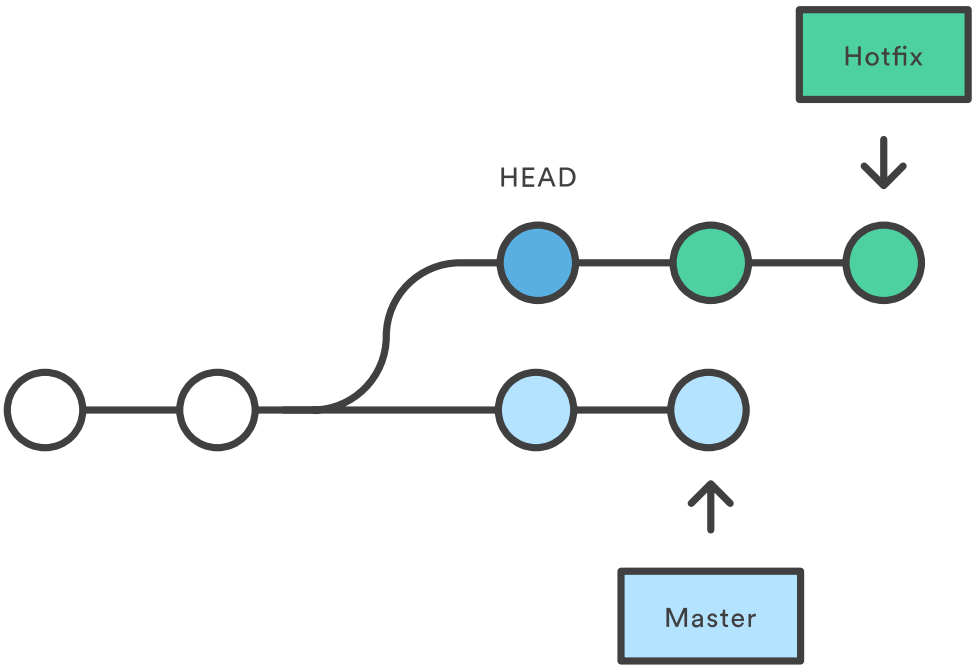


Dangling Commits

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

```
git checkout HEAD~2
```

Moving HEAD to an arbitrary commit



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

Revert

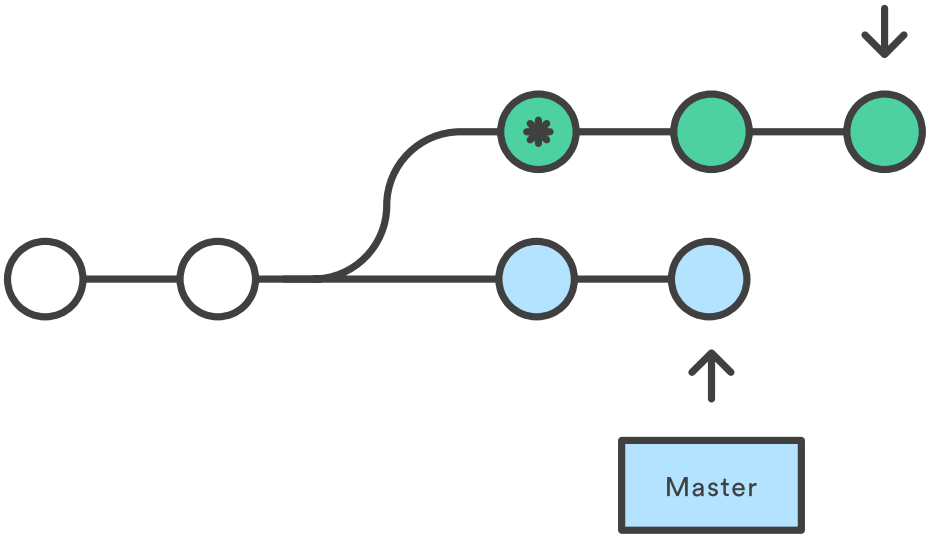
Revert撤销一个commit的同时会创建一个新的commit。这是一个安全的方法，因为它不会重写commit历史。比如，下面的命令会找出倒数第二个commit，然后创建一个新的commit来撤销这些更改，然后把这个commit加入项目中。

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

如下图所示：

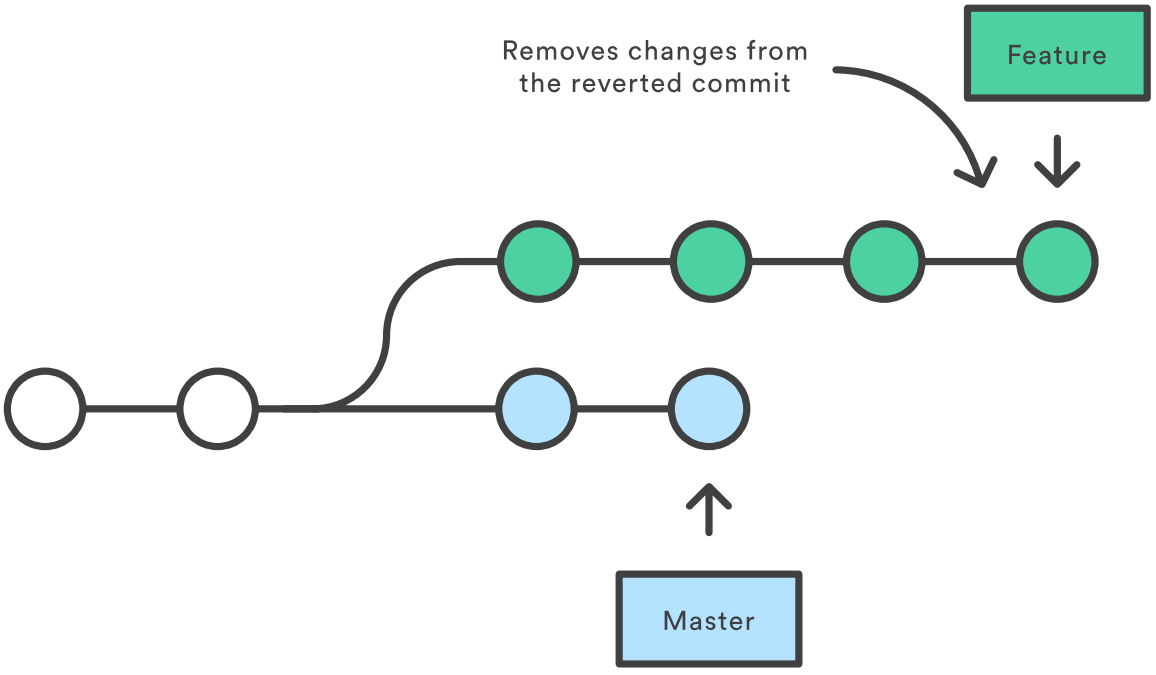
Reverting the 2nd to last commit

Before Reverting



Commit to be reverted

After Reverting



相比`git reset`，它不会改变现在的commit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commit的更改，而`git reset HEAD`用来撤销没有commit

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commit或者stash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commit，参数限制它作用于特定文件。

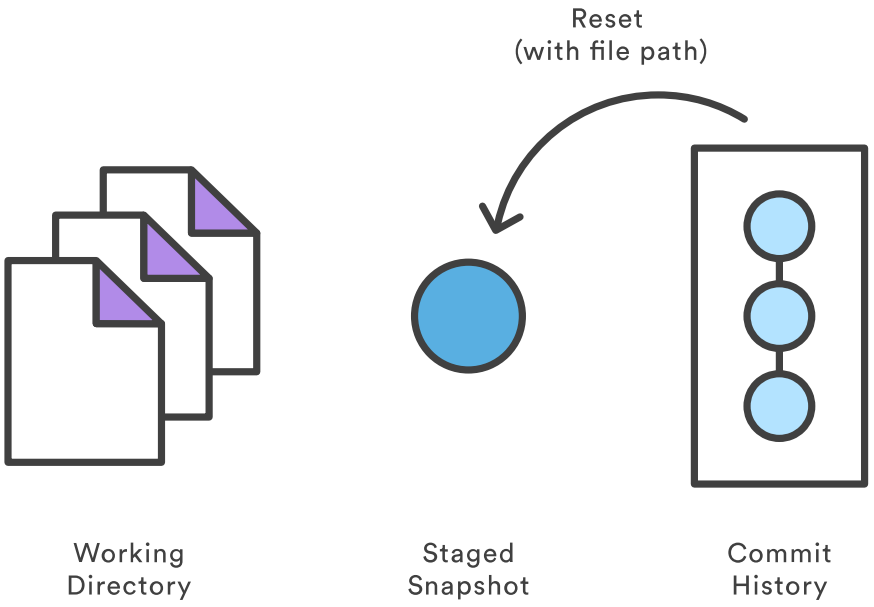
Reset

当检测到文件路径时，`git reset`将stage缓存同步到你指定的那个commit。比如，下面这个命令会将倒数第二个commit中的foo.py加入到stage缓存中，供下一个commit使用。

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

和commit层面的`git reset`一样，通常我们使用HEAD而不是某个特定的commit。运行`git reset HEAD foo.py`会将当前的foo.py从stage缓存中移除出去，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

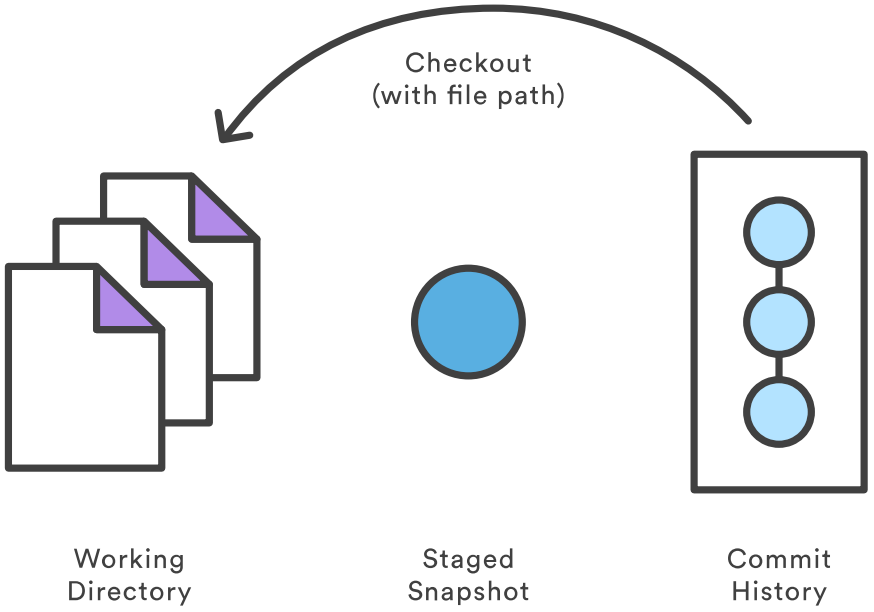


--soft、--mixed和--hard对文件层面的`git reset`毫无作用，因为stage缓存中的文件一定会变化，而工作目录中的文件一定不变。

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stage缓存。不像commit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commit中的foo.py。

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

和commit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stage并且commit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commit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`的作用等同于舍弃foo.py没有stage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，但当你想起它们工作目录、stage缓存和commit历史分别的影响，就会容易判断现在的情况下应该用那个命令。

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，你使用Git时一定会经常用到。

命令	作用域	常用情景
git reset	Commit层面	在私有分支上舍弃一些没有commit的更改
git reset	文件层面	将文件从stage中移除
git checkout	Commit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	Commit层面	在公共分支上回撤更改
git revert	文件层面	（没有）