**1 HEAD**

HEAD 是一个对当前**检出记录**的**符号引用** —— **也就是指向你正在其基础上进行工作的提交记录**。HEAD 总是指向当前分支上最近一次提交记录。大多数修改提交树的 Git 命令都是从改变 HEAD 的指向开始的。HEAD 通常情况下是指向分支名的（如 bugFix）。在你提交时，改变了 bugFix 的状态，这一变化通过 HEAD 变得可见。

如果想看 HEAD 指向，可以通过 cat .git/HEAD 查看， 如果 HEAD 指向的是一个引用，还可以用 git symbolic-ref HEAD 查看它的指向。

分离的HEAD 让其指向某个具体的提交记录而不是分支名：

Eg.

original state: HEAD->master-> C1

Run git checkout **C1** (it’s a specific commit,not a branch)

HEAD ->C1

**2 相对引用**

用 git log 来查查看提交记录的哈希值。哈希值是基于SHA-1,共40位，一般只需要输入可以唯一标识的前几位。

通过哈希值指定提交记录很不方便，所以 Git 引入了**相对引用**。这个就很厉害了!使用相对引用的话，你就可以从一个**易于记忆的地方**（比如 bugFix 分支或 HEAD）开始计算。相对引用非常给力，这里我介绍两个简单的用法：

1. 使用^向上移动一个提交记录；把这个符号加在引用名称的后面，表示让 Git 寻找指定提交记录的父提交。

**所以 master^ 相当于“master 的父节点”。**

**master^^ 是 master 的第二个父节点**

1. 使用~<num>向上移动多个提交记录，如~3.

另外，也可以将 HEAD 作为相对引用的参照。也就是说，可以一直使用git checkout HEAD^实现一直上移。

强制修改分支位置，git branch **-f** master HEAD~3 此命令会将master分支强制指向HEAD的第三级父提交

注意 git branch -f <分支> <提交记录>

git checkout <分支>

**3 撤销变更**

撤销变更由底层部分（暂存区的独立文件或者片段）和上层部分（变更到底是通过哪种方式被撤销的）组成。我们主要关注后者。主要有两种方法用来撤销变更 —— 一是 **git reset**，还有就是 **git revert**。接下来咱们逐个进行讲解。

git reset 通过把分支记录回退几个提交记录来实现撤销改动。你可以将这想象成“改写历史”。git reset 向上移动分支，原来指向的提交记录就跟从来没有提交过一样。

Eg. original state: HEAD->master->C2

Run git reset HEAD~1

New state :HEAD->masetr->C1

注意，这和上面的相对引用不同，这会将HEAD和master同时改变，并且现在我们的本地代码库根本就不知道有 C2 这个提交了，C2 所做的变更还在，但是处于未加入暂存区状态。

**但是这种“改写历史”的方法对大家一起使用的远程分支是无效的**

Eg.original state: HEAD->master->C2

**Run git revert HEAD（注意，这里并没有加~，因为这两种方法背后原理是不一样的）**

New state :HEAD->masetr->**C2‘**

在我们要撤销的提交记录后面居然多了一个新提交！这是因为新提交记录 C2' 引入了更改 —— 这些更改刚好是用来撤销 C2 这个提交的。也就是说 C2' 的状态与 C1 是相同的。revert 之后就可以把你的更改推送到远程仓库与别人分享啦。