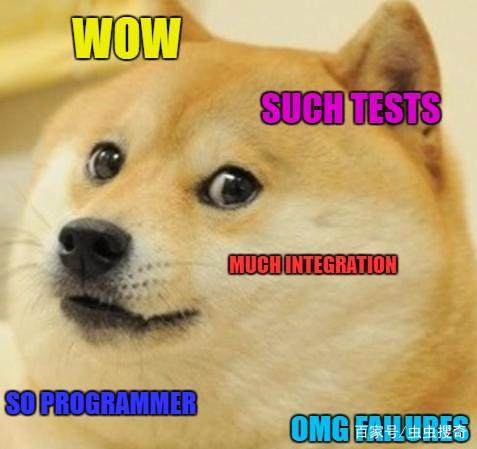
如果你处于一个组织良好的开发环境中，这个命令对你来说几乎是无用的。但现实中的场景往是，你是宅在家里一个人做小型项目，或者你的公司还没要建立完善的CI/CD(持续的集成和发布)流程，更大的可能是你们的测试被狗吃了（不是@了狗的测试，虽然也差不多），并出现以下情况：



试想，当你在愉快的开发着你想象中的炫酷功能的时候，突然销售妹纸（想得美，是个老爷们大胖子）给你打了一个紧急电话说你发布版本崩溃了，没法工作。你试着看了看代码和错误日志，但没有任何头绪，在qq群询问老司机后，老司机建议你找之前正常的版本，然后对比下。因此你现在问题是要找到引入错误的那次commite。该怎么做？什么，不会了吧。

git bisect是一个很有用的命令，用来查找哪一次代码提交引入了错误。

它的原理很简单，就是将代码提交的历史，按照两分法不断缩小定位。所谓"两分法"，就是将代码历史一分为二，确定问题出在前半部分，还是后半部分，不断执行这个过程，直到范围缩小到某一次代码提交。

我们通过一个实例，解释如何使用这个命令。

下面是一个代码库，请将它克隆到本地。

$ git clone [git@github](mailto:git@github).com:bradleyboy/bisectercise.git

$ cd bisectercise

这个库是一个网页，在浏览器打开这个网页

网页上有一个计数器，有两个按钮，点击+号按钮可以看到计数器没有递增，反而递减，这说明代码有问题

现在，就要来查找，到底是哪一次的代码提交引入了错误。首先，检查一下代码提交历史

$ git log --pretty=oneline

可以看到，这个库一共有101次提交，最早的一次提交的哈希是4d83cf

$ git bisect start [终点] [起点]

“起点”是更久以前的提交，“终点”是最近的提交。他们之间的这段历史。就是查错的范围。

$ git bisect start HEAD 4d83cf

执行上面的命令之后，代码库就会切换到这段范围正文中的那一次提交，也就是第51次提交。

现在刷新浏览器，点击+按钮，发现可以正常递增。使用 git bisect good命令，标识本次提交没有问题。

$ git bisect good

也就是说1-51次提交没有问题，那也就意味着错误是在后半段引入的。执行上边的命令之后，Git就自动切换到后半段的中点（第76次提交）

现在刷新浏览器，点击+按钮，发现不能正常递增。使用 git bisect bad命令，标识本次提交（第76）有问题

$ git bisect bad

执行上边的命令之后，Git就自动切换到51-76的中点（第63次提交）

接下来，不断重复这个过程，直到成功找到问题的那一次提交为止。这时，Git会给出如下的提示

b47892 is the first bad commit

既然找到那个有问题的提交，就可以检查代码，确定具体是什么错误

然后使用 git bisect reset 命令，退出差错，回到最近一次的代码提交

$ git bisect reset