这一讲，我就讲解rebase这个东西，为什么放这儿讲解？853769620

这里的话，我说几句闲话

上一次有同学说，看完maven之后，问，能不能讲解git之后，直接讲解一下jenkins

不行啊，因为的授课风格和授课理念，未来应该全部是实战驱动，场景驱动，需求驱动，去学习每一块技术

git，从需求出发，就是说，你需要进行版本控制了，才需要去使用git，然后上来头8讲，都在讲解版本控制这个事儿，包括怎么提交和记录版本，怎么进行版本控制

从分支那块儿开始，都是出于基于git，在不同类型的项目，不同规模的团队的场景下，用什么样的既有分支的协作工作流去进行开发

我没有死板板的给你讲解，branch，一共20个命令

直接就是实际的功能分支工作流场景出发，一步一步学习branch命令，功能分支工作流场景实践完了，你的branch命令就全都掌握了

前8讲，git如何做版本控制这个本质玩儿清楚了，git add，git commit，git status，git log，这些，命令自然而然就清楚是怎么回事了

我不是说，git add，git commit，git status，git branch，git remote

git完事儿了，怎么就直接讲jenkins了呢？？？很突兀，莫名奇妙就开始讲解jenkins了，死记硬背，什么功能

我要直接就是在后面有一个大的阶段，第二个大阶段，微服务架构阶段，jenkins融入到那个阶段里面去讲解，微服务场景下，jenkins是怎么去玩儿的

rebase当做一个很高深的技能，放在最后几讲，git的深入内幕命令，rebase，高深，高手

rebase在实际工作场景中很少用，他的适用场景也有限，但是不是说rebase就没用，也有用，是有其适用的场景的

集中式工作流这个场景结合起来去讲解rebase，一方面是讲解你的这个rebase的原理，一方面是讲解rebase在集中式工作流场景下，使用它的作用是什么呢？

在刚才那种普通的集中式工作流开发流程下，会有一个问题，就是每次如果某个人push了代码到master，然后另外一个哥儿们也要push会失败

此时那个哥儿们就要git pull拉取和合并master分支的代码

坑爹的事情来了，每次都这样合并的话，会导致你最后发现看到这个项目，它的提交历史就是各种分叉，各种合并，提交历史感觉很难看

实际上来说，对于一个普通的集中式工作流开发的小项目，是不需要这样分叉的提交历史的，我们希望的是那种很平滑的，就一条线一样的提交历史，看起来会比较清爽

要优化集中式工作流的提交历史成为一条线，看起来非常的清晰和清爽，此时就要使用rebase命令

git pull --rebase

rebase：变基，就是改变commit之前依赖的基础commit

通过git pull --rebase，执行变基式的合并，改变commit历史，看起来提交历史就是一条直线，张三先提交，李四再提交，李四再提交，张三再提交

集中式工作流的项目，看起来比较清爽

比如说我们有一个master分支，还有另外一个分支比如feature/001

也可以是合并的时候，采取git rebase feature/001来合并

你就全部都采取git pull --rebase来跟其他人对master分支的修改进行合并，就能保持项目的提交历史就是一条直线