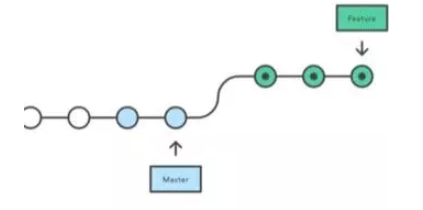
Git rebase操作：

rebase需要基于一个分支来设置你当前的分支的基线，这基线就是当前分支的开始时间轴向后移动到最新的跟踪分支的最后面，这样你的当前分支就是最新的跟踪分支。**rebase**会把你当前分支的 commit 放到公共分支的最后面, 所以叫变基。就好像你从公共分支又重新拉出来这个分支一样。

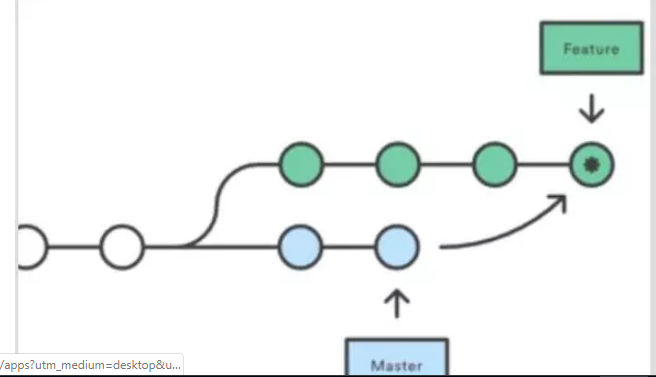
举例:如果你从 master 拉了个feature分支出来,然后你提交了几个 commit,这个时候刚好有人把他开发的东西合并到 master 了,这个时候 master 就比你拉分支的时候多了几个 commit,如果这个时候在feature分支上，执行git rebase master的话，就会把你当前的几个 commit，放到那个人 commit 的后面。



Rebase

可以看到rebase以后， git会把master后来的提交放在feature分支的提交之前，重新建立基础。

**merge** 会把公共分支和你当前的commit 合并在一起，形成一个新的 commit 提交



Merge

我们可以看到当merge以后，git会将master的最新提交放到feature分支前面，并且再生成一次merge commit.

举例2：

 1-2-3 是现在的分支状态

 这个时候从原来的master ,checkout出来一个prod分支

 然后master提交了4.5，prod提交了6.7

 这个时候master分支状态就是1-2-3-4-5，prod状态变成1-2-3-6-7

 如果在prod上用rebase master ,prod分支状态就成了1-2-3-4-5-6-7

 如果是merge  
1-2-3-6-7-8  
........ |*4-5*|

 会出来一个8，这个8的提交就是把4-5合进来的提交

------不要再公共分支使用rebase

公共分支就是大家已经从这个分支建立其他分支的基本分支，例如master公共分支。一般不在master分支上进行rebase操作，因为master分支默认是公共分支，当多人协作时，master分支在多个地方都有副本；若在master分支上作rebase，***会改变master分支上的提交历史，而其他人已经基于以前的commit对象工作，当其拉取master的新对象时，需要合并，***这样反复下去，会把master分支搞乱！

------分支经过rebase以后再次push

我们有时候会遇到这样一个情况，我们从master上拉出feature分支，然后master改变以后，我们在本地rebase了feature分支，然后本地feature分支的提交将会被重写。但是远端的feature分支的提交历史没有被重写，然后我们继续在本地修改feature分支以后，向远端push的时候默认会被拒绝。因为提交历史不一致，我们可以使用git push origin feature --force.