# [merge和rebase的区别](http://www.cnblogs.com/xueweihan/p/5743327.html)

## 前言

我从用git就一直用rebase，但是新的公司需要用merge命令，我不是很明白，所以查了一些资料，总结了下面的内容，如果有什么不妥的地方，还望指正，我一定虚心学习。

## merge和rebase

标题上的两个命令：merge和rebase都是用来合并分支的。

**这里不解释rebase命令，以及两个命令的原理**，详细解释参考[这里](http://gitbook.liuhui998.com/4_2.html)。

下面的内容主要说的是两者在实际操作中的区别。

## 什么是分支

分支就是便于多人在同一项目中的协作开发。比方说：每个人开发不同的功能，在各自的分支开发过程中互不影响，完成后都提交到develop分支。极大的提高了开发的效率。

## 合并分支

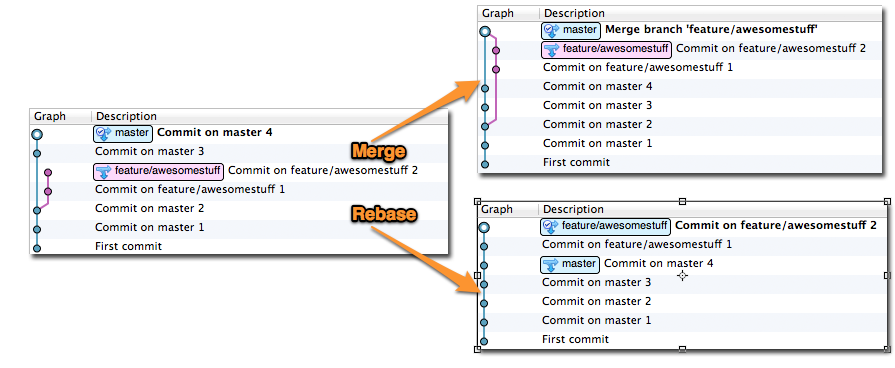
每个人创建一个分支进行开发，当开发完成，需要合并到develop分支的时候，就需要用到**合并**的命令。

## 什么是冲突

合并的时候，有可能会产生冲突。

冲突的产生是因为在合并的时候，不同分支修改了相同的位置。所以在合并的时候git不知道那个到底是你想保留的，所以就提出疑问（冲突提醒）让你自己手动选择想要保留的内容，从而解决冲突。

## merge和rebase的区别

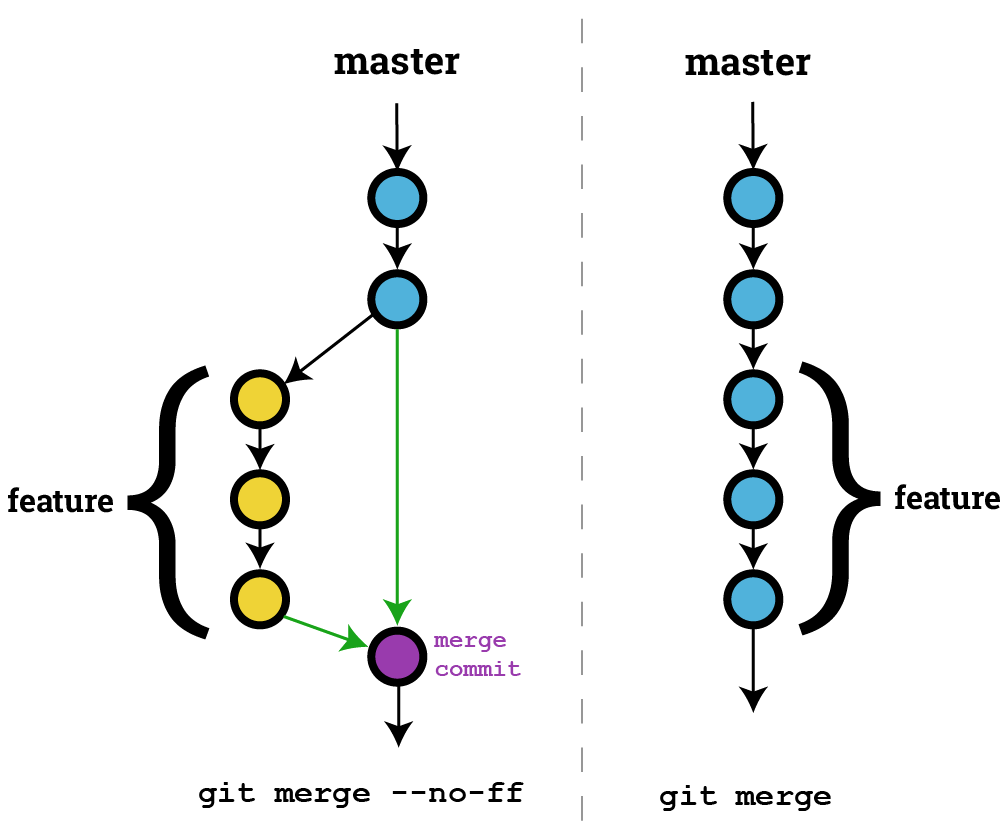
1. 处理冲突的方式：
   * 使用merge命令合并分支，解决完冲突，执行git add .和git commit -m'fix conflict'。这个时候会产生一个commit。
   * 使用rebase命令合并分支，解决完冲突，执行git add .和git rebase --continue，不会产生额外的commit。这样的好处是‘干净’，分支上不会有无意义的解决分支的commit。
2. git pull和git pull --rebase区别：git pull做了两个操作分别是‘获取’和合并。所以加了rebase就是以rebase的方式进行合并分支，默认为merge。
3. 采用merge和rebase后，git log的区别：  
   

**总结**：我的简单理解——merge显性的处理冲突，rebase隐性的处理冲突。

## git merge 和 git merge --no-ff的区别

我自己尝试merge命令后，发现：merge时并没有产生一个commit。不是说merge时会产生一个merge commit吗？

**注意**：只有在冲突的时候，解决完冲突才会自动产生一个commit。

如果想在没有冲突的情况下也自动生成一个commit，记录此次合并就可以用：git merge --no-ff命令，下面用一张图来表示两者的区别：  


## 如何选择合并分支的方式

我的理解：主要是看那个命令用的熟练，能够有效的管理自己的代码；还有就是团队用的是那种方式。

我对于rebase比较熟悉，所以我一般都用rebase，但是现在的公司用的是merge --no-ff命令合并分支。所以，我在工作上就用merge，个人项目就用rebase。

也可以两者结合：

* 获取远程项目中最新代码时：git pull --rebase，这个时隐性的合并远程分支的代码不会产生而外的commit。
* 合并到分支的时候：git merge --no-ff，自动一个merge commit，便于管理（这看管理人员怎么认为了）

## 总结

* 合并时，没有冲突的情况下两者一样。（我认为，如果理解错了还望指正）
* merge**显性**处理冲突
* rebase**隐性**处理冲突

## 参考

* [Git Book](http://gitbook.liuhui998.com/4_2.html)
* [StackoverFlow:difference between merge and merge --no-ff ?](http://stackoverflow.com/questions/9069061/what-is-the-difference-between-git-merge-and-git-merge-no-ff)
* [Source Tree Blog](https://blog.sourcetreeapp.com/2012/08/21/merge-or-rebase/)
* [Yu-Cheng Chuang’s Blog](https://blog.yorkxin.org/2011/07/29/git-rebase)

git merge和rebase

# [git删除目录](http://blog.csdn.net/skykingf/article/details/44078837)

使用**[Git](http://lib.csdn.net/base/28" \t "_blank" \o "Git知识库)** rm -rf dir 命令删除非空目录之后，本地还是会有空的目录存在，这时候空目录已经是untracked状态了。

解决办法是再删除掉untracked状态的目录，命令如下

[git](http://lib.csdn.net/base/28) rm -rf dir #删除本地（非空）目录

git clean –fd #删除掉untracked状态的目录

# [git创建与管理(本地/远程)分支](http://blog.csdn.net/fyh2003/article/details/7528157)

git branch branch1 #创建本地分支

git push origin branch1 #创建远程分支

git branch –a #查看本地和远程分支

git branch –d branch1 #删除本地分支

git branch –r –d branch1 #删除远程分支

**git fetch和pull**

1. git fetch：相当于是从远程获取最新版本到本地，不会自动merge

git fetch origin master:m\_tmp #获取远程origin的master分支的所有数据到本地（新创建的）m\_tmp分支

git fetch origin branch\_1:b\_tmp #获取远程origin的branch\_1分支的所有数据到本地（新创建的）b\_tmp分支

git diff b\_temp #查看当前分支和b\_temp分支的差异

git merge b\_temp #合并b\_temp到当前分支

2. git pull：相当于是从远程获取最新版本并merge到本地

git pull origin master #获取并合并到本地

# git reset和checkout

git reset --hard HEAD #让工作目录回到上次提交时的状态

git checkout -- hello.c #恢复一个文件到上次提交时的状态