分支

合并提交

pull --rebase冲突不产生merge

暂存不想commit的代码: stash

标签

分支

(1) 新建分支

git branch BranchName 在当前前提下新建分支 git checkout xxxxxxxx哈希值 -b BranchName 在指定的commit的基础上新建分支

(2) 切换分支

git checkout BranchName 切换到某个分支

git switch BranchName

在该分支上可以做各种操作,别忘了git commit -a -m "message"

(3) push到远程仓库

git push origin HEAD:远程新branchName 或者

(4) 将分支合并到主分支

切换到主分支, git merge <name>, 将目标分支合并进来

revert撤销 ("反做"某个commit)

git revert commit "版本号"

git revert是用于"反做"某一个版本,以达到撤销该版本的修改的目的。比如,我们commit了三个版本(版本一、版本二、版本三),突然发现版本二不行(如:有bug),想要撤销版本二,但又不想影响撤销版本三的提交,就可以用 git revert 命令来反做版本二,生成新的版本四,这个版本四里会保留版本三的东西,但撤销了版本二的东西。

reset

git reset --hard 目标版本号 强制回退到目标版本,目标版本之后的版本都不再存在

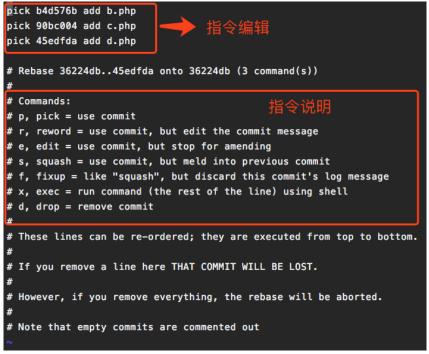
合并提交

(1) 与上一次commit合并,不再产生新的commit记录(hash值更新)
git commit --amend --no-edit
git commit --amend -m "xxx" 顺便修改message
git commit --fixup 也能达到同样的效果(hash值不更新)

- (2) 将若干个本地零碎的commit记录合并整合成一个commit
 - ①git rebase -i -i表示弹出交互式的界面让用户进行合并操作
 - ②指令编辑:

显示当前未提交的所有commit,修改其前面的"pick"即可修改对其的操作通常保留一个提交信息为pick,其余的为s,进行合并

③编辑指令后,Esc退出,然后输入:wq保存执行



④wq保存后进入如下界面,编辑要修改的提交信息,不需要的信息前打#号

```
This is a combination of 3 commits.
 The first commit's message is:
                                     这是合并后的提交注释信息
# This is the 2nd commit message:
# This is the 3rd commit message:
add d.php
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
 Date: 令类 FrigJan 5 11:03:02 2018 +0800
 interactive rebase in progress; onto 36224db
# Last commands done (3 commands done):
    s 90bc004 add c.php
   (s 45edfda add d.php)
 No commands remaining.
# You are currently editing a commit while rebasing branch 'ssss' on '36224db'.
# Changes to be committed:
    new file: b.php
       new file: c.php
new file: d.php
```

可以在this is the 1st commit message中输入合并后的message ⑤操作完可以查看 git log,就可以发现多余的commit记录不见了





pull --rebase冲突不产生merge

参考: https://blog.csdn.net/gican 7/article/details/98870789

git pull = git fetch + git merge 会产生一次merge记录 git pull --rebase = git fetch + git rebase 不会产生额外merge记录

- ①正常commit
- ②git pull --rebase拉取代码
- ③解决冲突 (好像自动会解决,检查一下)
- ④git add.(可能不需要)
- ⑤git rebase --continue告诉git已经解决好了冲突 (不需要再次commit)
- **6**git push

暂存不想commit的代码: stash

切换分支时,会提示有未commit的代码,无法切换。此时不想commit烂代码,又不想它们消失

- ①隐藏修改: git stash (git stash list可以查看被隐藏的代码,且有标号)
- ②做其他操作
- ③重新显示隐藏的stash,并将其从stash list中删除: git stash pop或git stash apply stash@{0?}

标签

当一个代码仓库进过长时间的迭代,针对不同的时期和需求,必定会有不同的版本。而借助 Git 提供的标签功能,可以快捷方便地记录代码版本。无论什么时候,想取回某个版本,不再需要查找冗长的commit id,只需要取出打标签的历史版本即可

- ① (在GitHub上建仓) ,将代码库clone到本地
- ②本地进行标签操作:

git tag <tagname> -m<comment> 标签名及说明 git tag 查看所有标签

③标签推送至远程仓库

git push origin <tagname> 推送指定的标签至远程仓库 git push origin --tags 推送所有标签 此时到GitHub上就可以看到新增的releases

④注意:最好开辟分支来编码,维护一个在线版本master和开发分支dev,完成一个阶段再将dev合并到master