目录

[1. 版本管理 4](#_Toc505789843)

[1.1. 发布1.0.0版本及标签 4](#_Toc505789844)

[1.2. 继续在master主干上开发新功能 5](#_Toc505789845)

[1.3. 创建临时分支master\_hotfix进行 BUG修改 6](#_Toc505789846)

[1.4. 修改BUG 7](#_Toc505789847)

[1.5. 打包升级至线上 8](#_Toc505789848)

[1.6. 在master\_hofix分支上打标签master\_hofix\_1.0.0.1 9](#_Toc505789849)

[1.7. 合并master\_hofix分支代码至master 10](#_Toc505789850)

[1.8. 继续开发… 11](#_Toc505789851)

[1.9. 发布1.0.1版本及标签 12](#_Toc505789852)

[1.10. 删除hotfix分支 13](#_Toc505789853)

[2. 其它命令 14](#_Toc505789854)

# 版本管理

## 发布1.0.0版本及标签

…

使用 git tag 来添加新标签

git tag -a version -m "note"

$ git tag -a master\_1.0.0 -m '这是master 1.0.0'

使用 git push 来将标签推送到远程仓库

git push origin refs/tags/源标签名:refs/tags/目的标签名

$ git push origin master\_1.0.0:master\_1.0.0



## 继续在master主干上开发新功能

暂时不考虑单独开develop分支进行代码开发

$ git add .

$ git commit -m 修改test1及删除test2

$ git push

……

$ git add .

$ git commit -m 添加test3

$ git push



## 创建临时分支master\_hotfix进行 BUG修改

线上发现BUG需要修改，找到线上版本标签

$ git tag

$ git checkout master\_1.0.0（tag名）

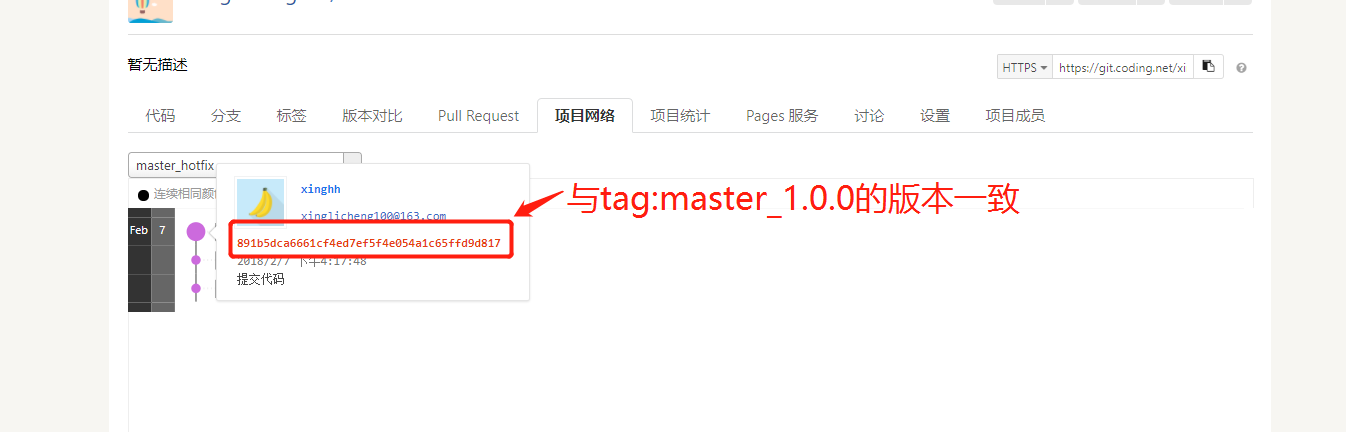
因为 tag 相当于是一个快照，无法修改代码，如果要在 tag 代码的基础上做修改，需要创建分支：

$ git checkout -b master\_hotfix master\_1.0.0

推送BUG修改分支至coding

$ git push origin master\_hotfix:master\_hotfix





## 修改BUG

在master\_hofix分支上进行BUG修改

$ git add .

$ git commit -m 修改BUG

$ git push

……

$ git add .

$ git commit -m 修改BUG2

$ git push



## 打包升级至线上

…

## 在master\_hofix分支上打标签master\_hofix\_1.0.0.1

$ git tag -a master\_hofix\_1.0.0.1 -m 'master hofix\_1.0.0.1'

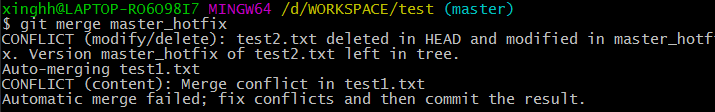
$ git push origin master\_hofix\_1.0.0.1:master\_hofix\_1.0.0.1

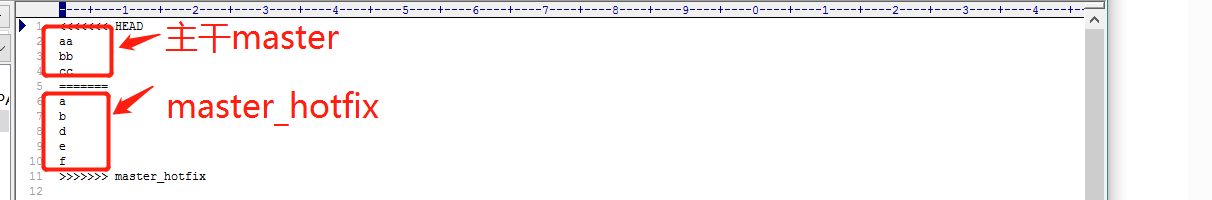


## 合并master\_hofix分支代码至master

$ git checkout master

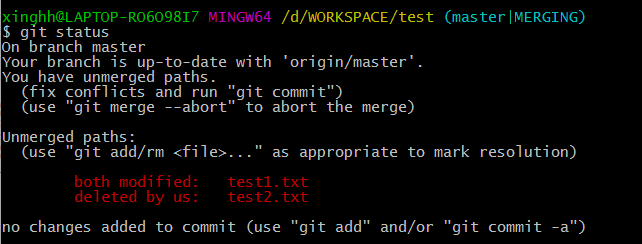
$ git merge master\_hotfix





查看合并后的本地库状态

$ git status



进行手动合并

…

$ git add .

$ git commit -m 合并代码

$ git push

## 继续开发…

## 发布1.0.1版本及标签

$ git tag -a master\_1.0.1 -m 'master 1.0.1'

$ git push origin master\_1.0.1:master\_1.0.1



## 删除hotfix分支

$ git branch

$ git branch -d master\_hotfix

$ git push origin :master\_hotfix



删除master\_hotfix被删除，则相应的tag标签依然有效

如果master\_1.0.1线上发现问题，则按1.3按需往下执行即可

# 其它命令

查看git log

$ git log --pretty=oneline --abbrev-commit

回退版本

$ git reset --hard HEAD^/HEAD~100(^表示当前版本前一位/HEAD~100表示当前版本前100位)

$ git reset --hard d87a1(至少四位)