# [SourceTree的基本使用](http://www.cnblogs.com/tian-xie/p/6264104.html)

## **1. SourceTree是什么**

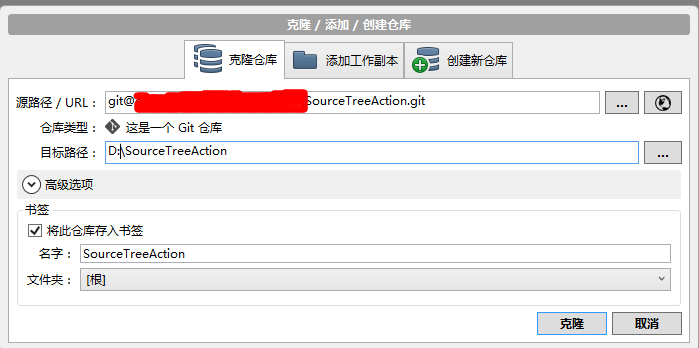
* 拥有可视化界面的项目版本控制软件，适用于git项目管理
* window、mac可用

## **2. 获取项目代码**

1. 点击克隆/新建

IMG_256

2. 在弹出框中输入项目地址，http或者ssh地址都可以

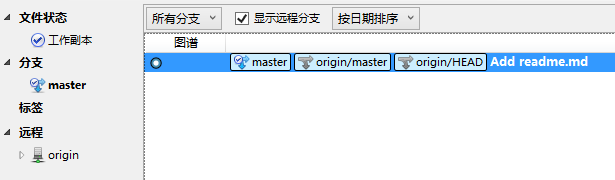


　　如果箭头指向的仓库类型表明“这不是一个标准的Git仓库”，可能是有以下原因

　　　　1) 项目地址获取错误

　　　　2) 没有项目访问权限

3. 点击“克隆”，等待项目克隆完成，完成后，左侧只有一个分支master

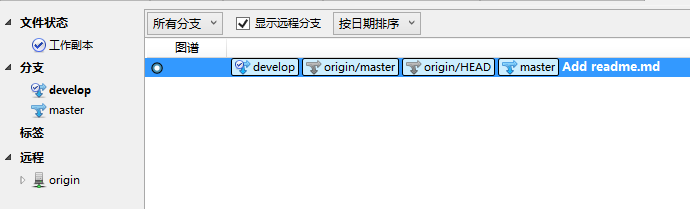


　　克隆完成后，得到的是发布后的master源码，如果想要获取最新的正在开发中的源码，需要对项目流进行初始化，点击“Git工作流”





　　直接点“确定”，获取develop分支源码



　　开发任务都是在develop分支上完成的

4. 分支共有5种类型

　　1) master，最终发布版本，整个项目中有且只有一个

　　2) develop，项目的开发分支，原则上项目中有且只有一个

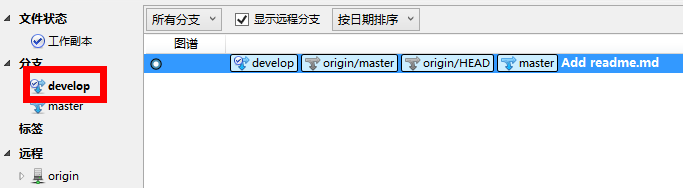
　　3) feature，功能分支，用于开发一个新的功能

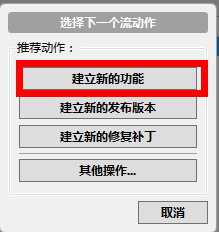
　　4) release，预发布版本，介于develop和master之间的一个版本，主要用于测试

　　5) hotfix，修复补丁，用于修复master上的bug，直接作用于master

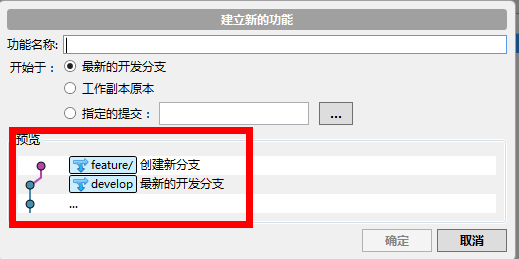
5. master和develop上文中已介绍过，当开发中需要增加一个新的功能时，可新建feature分支，用于增加新功能，并且不影响开发中的develop源码，当新功能增加完成后，完成feature分支，将新功能合并到develop中，更新develop上的代码

    1) 新建feature。首先当前开发分支指向develop，点击“Git工作流”

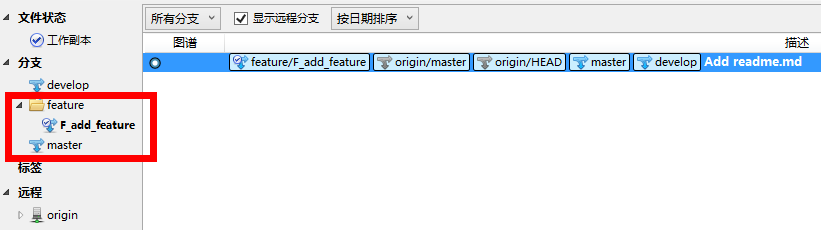




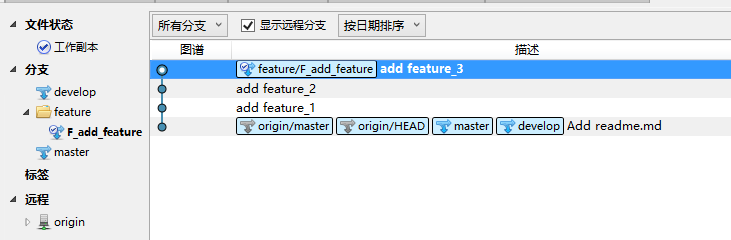
选择“建立新的分支”



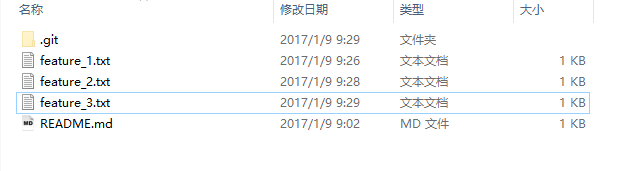
在预览中可看到，feature分支是从develop分出的，输入功能名称，点击确定，项目结构中增加feature分支，并且当前开发分支指向新建的feature分支



　　2) 在F\_add\_feature分支下进行开发任务，并提交



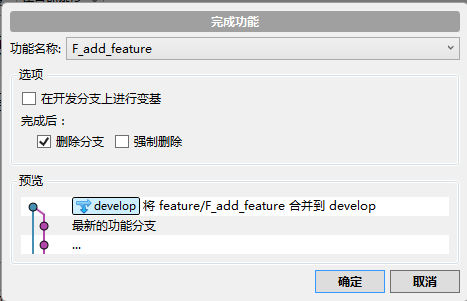
以上操作分别增加了feature\_1、feature\_2、feature\_3文件，共提交3次，现项目文件夹下共三个文件



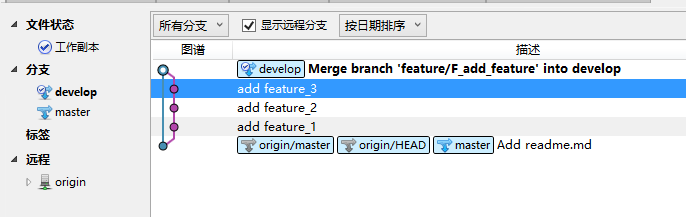
当切换为develop分支后，会发现，在develop下并没有新增的三个文件，说明在feature下进行操作，并不影响develop分支源码

　　3) 完成feature开发后，将feature中的源码合并到develop分支。将当前分支指向F\_add\_feature分支，点击“Git工作流”，选择“完成功能”



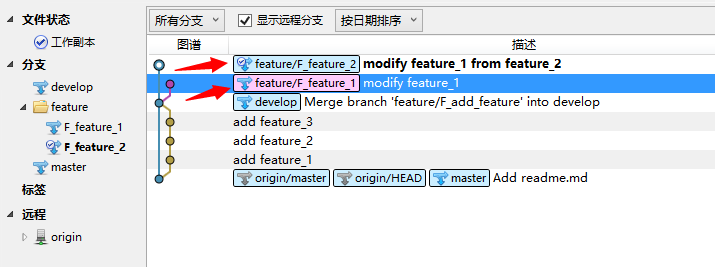


预览中，表明feature分支将合并到develop，点击确定，进行提交合并，合并成功后

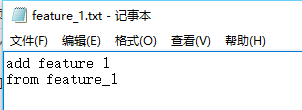


　　4) 需要再增加新的功能时，重复以上操作即可

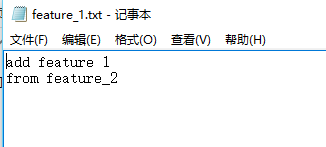
　　5) 当多人协作开发时，可能会出现，不同人员对同一文件进行操作，从而引起合并冲突，对这种情况进行模拟，在当前新建两个feature，分别对feature\_1文件进行修改，然后分别合并



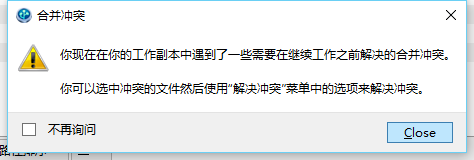
feature\_1在feature\_1.txt下做如下操作



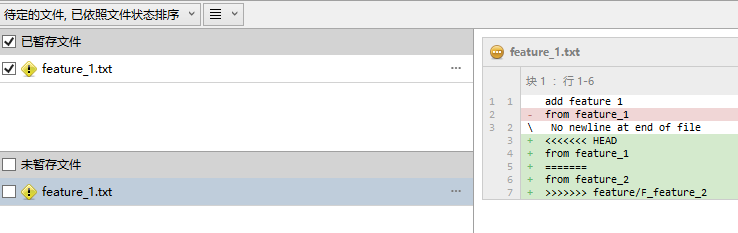
feature\_2在feature\_1.txt下做如下操作



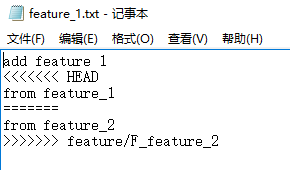
先后合并F\_feature\_1和F\_feature\_2，会出现冲突



点击close，查看未提交的更改，提示feature\_1.txt出现冲突，

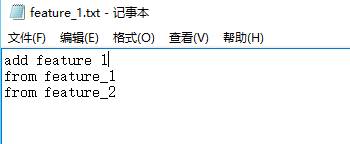


打开feature\_1.txt

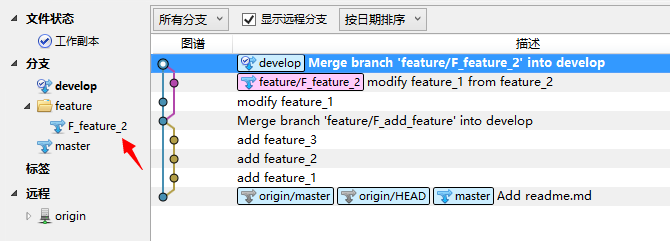


 出现<<<<<<< HEAD、=======、>>>>>>> feature/F\_feature\_2，HEAD和=号之间表示当前分支下的代码，=号和>>>>>>> feature/F\_feature\_2之间表示要合并的分支下的代码，>>>>>>> feature/F\_feature\_2表示了要合并的分支的分支名称，

根据情况区分要保留的代码，要删除的代码，最后再删除<<<<<<< HEAD、=======、和>>>>>>> feature/F\_feature\_2

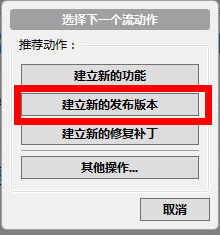
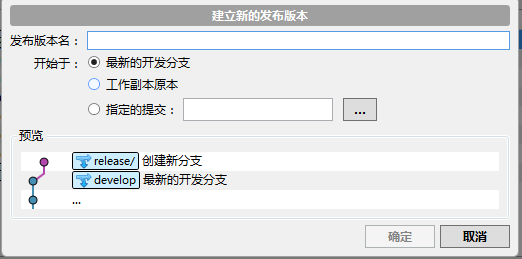


将修改的代码再进行一次提交

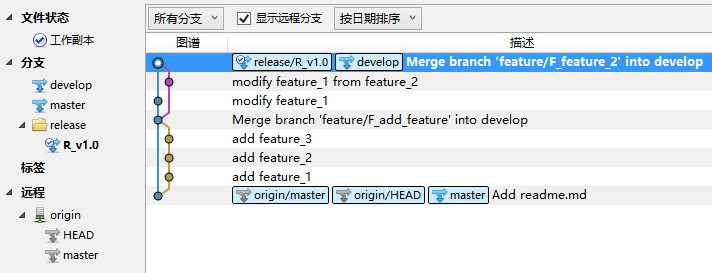


一旦出现feature合并冲突，要合并的feature分支不会被删除，如F\_feature\_2，确保合并没有问题后，可手动删除F\_feature\_2

6. 当开发到一定阶段，可以发布测试版本时，可以从develop分支，建立release分支，进入预发布测试阶段。点击“Git工作流”，选择“建立新的发布版本”

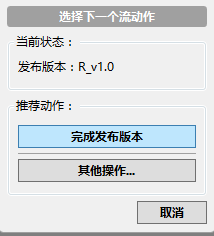
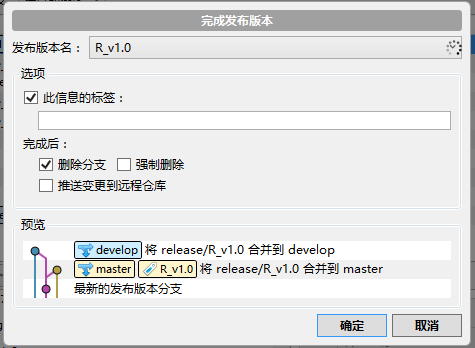
　　

预览中可以看到，release是从develop分出的，输入发布版本名‘R\_v1.0’，点击确定

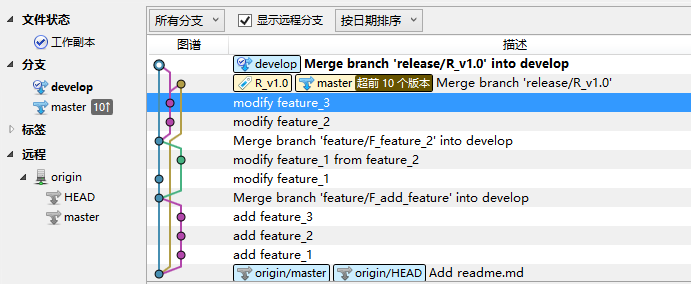


R\_v1.0为阶段性发布版本，主要用于发布前进行测试，后续的开发工作仍旧在develop上进行，如果在测试过程中发现问题，直接在release上进行修改，修改完成后进行提交

7. 对release分支R\_v1.0进行两次修改后，测试完成，可以进行正式发布，在当前分支指向R\_v1.0分支下，点击“Git工作流”，选择“完成发布版本”

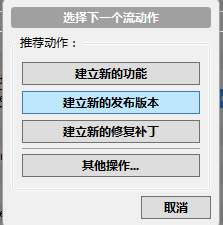
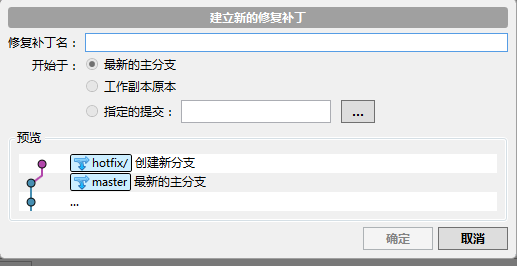
 

在预览中可以看到，R\_v1.0向develop和master分别合并，点击确定，完成正式发布。

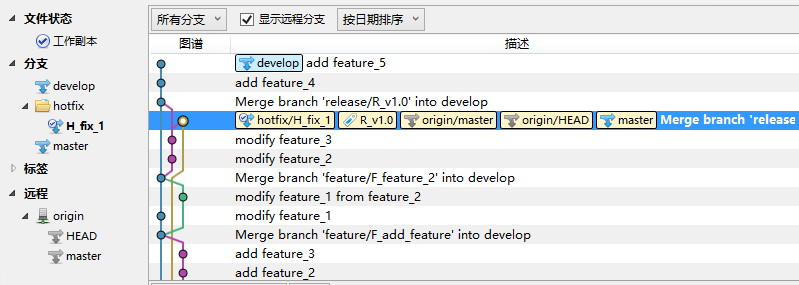


完成合并后，默认指向develop为当前分支，master增加多个版本更新，将master分支推送到origin，完成线上发布

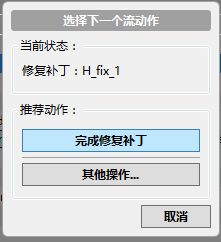
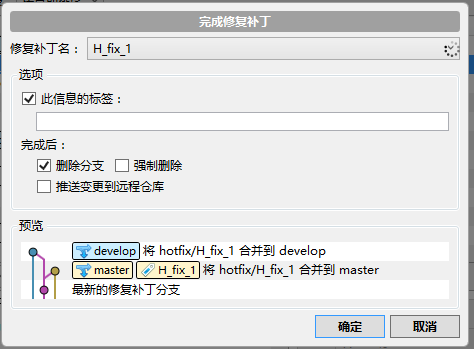
8. 正式版本发布后，develop可继续进行后续开发，当正式版本出现问题时，需要进行问题的修改，可以在master分支建立修改补丁hotfix。将当前分支切换到master，点击“Git工作流”，选择“建立新的修复补丁”

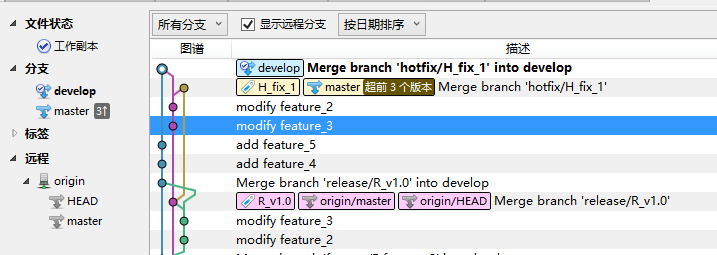
预览中hotfix分支是从master拉去出来的，输入修复补丁名，点确定



在该分支下进行master的问题修改，修改完成后进行提交。当所有补丁问题修改完成后，点击“Git工作流”，选择“完成修复补丁”

预览中，H\_fix\_1向master和develop分别合并，点击确定，完成分支合并。



合并完成后，默认当前分支为develop，master分支有版本需要更新，当前分支切换为master，进行推送，完成补丁修复。

9. 在完成发布版本和完成修复补丁时，如果遇到冲突，可仿照上述5进行冲突修改，再进行后续操作