GitFlow - Git在团队中的最佳实践

Git Flow的分支

- Production 分支

也就是我们经常使用的Master分支，这个分支最近发布到生产环境的代码，最近发布的Release， 这个分支只能从其他分支合并，不能在这个分支直接修改

- Develop 分支

这个分支是我们是我们的主开发分支，包含所有要发布到下一个Release的代码，这个主要合并与其他分支，比如Feature分支

- Feature 分支

这个分支主要是用来开发一个新的功能，一旦开发完成，我们合并回Develop分支进入下一个Release

- Release分支

当你需要一个发布一个新Release的时候，我们基于Develop分支创建一个Release分支，完成Release后，我们合并到Master和Develop分支

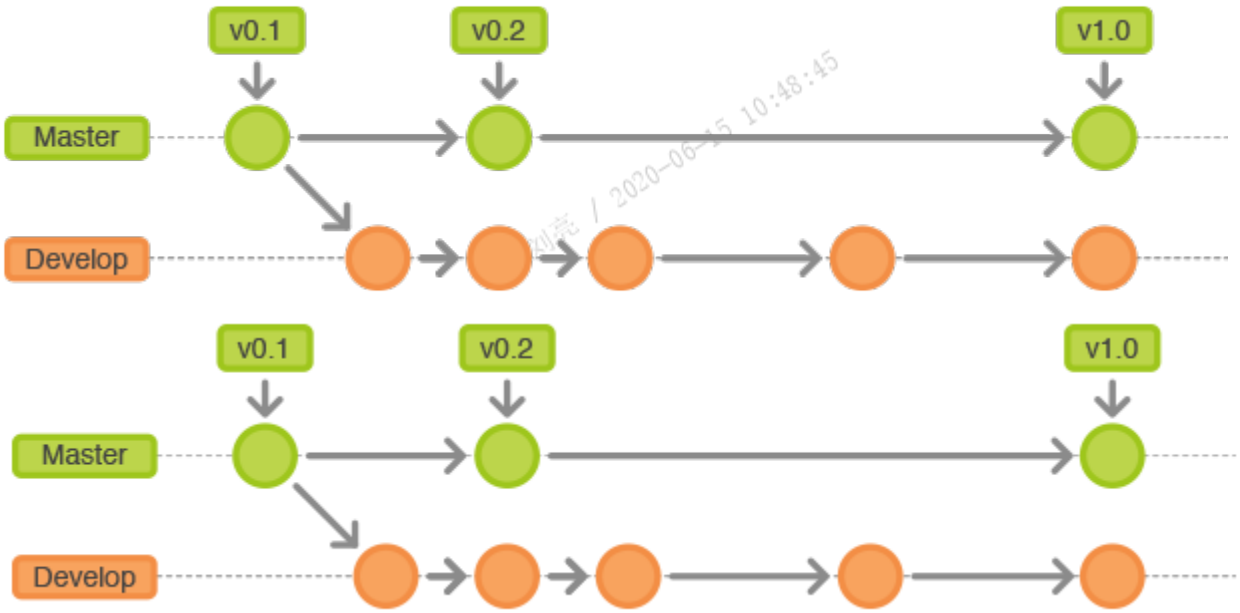
- Hotfix分支

当我们在Production发现新的Bug时候，我们需要创建一个Hotfix，完成Hotfix后，我们合并回Master和Develop分支，所以Hotfix的改动会进入下一个Release

Git Flow如何工作

初始分支

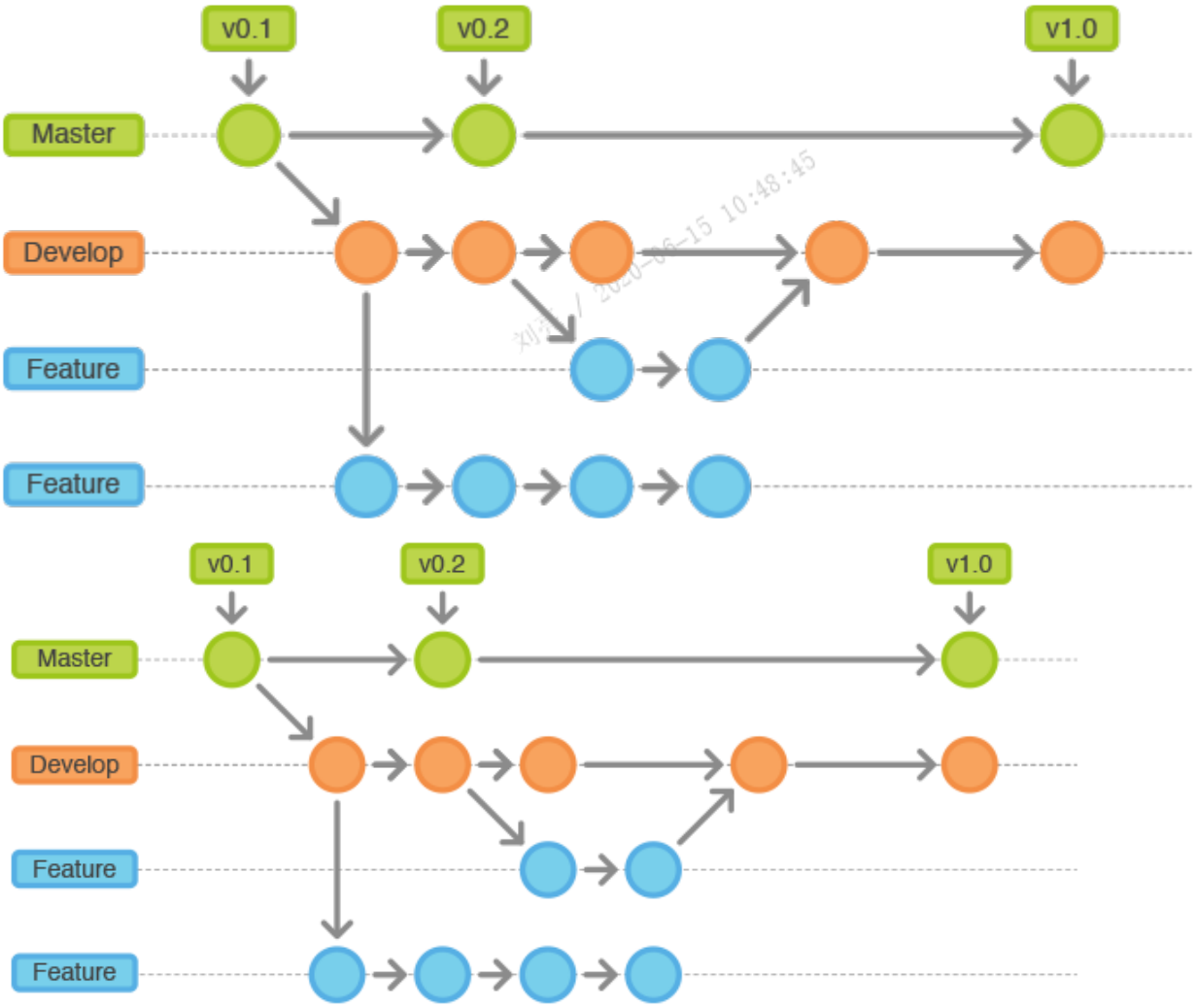
所有在Master分支上的Commit应该Tag



Feature 分支

分支名 feature/*

Feature分支做完后，必须合并回Develop分支，合并完分支后一般会删点这个Feature分支，但是我们也可以保留

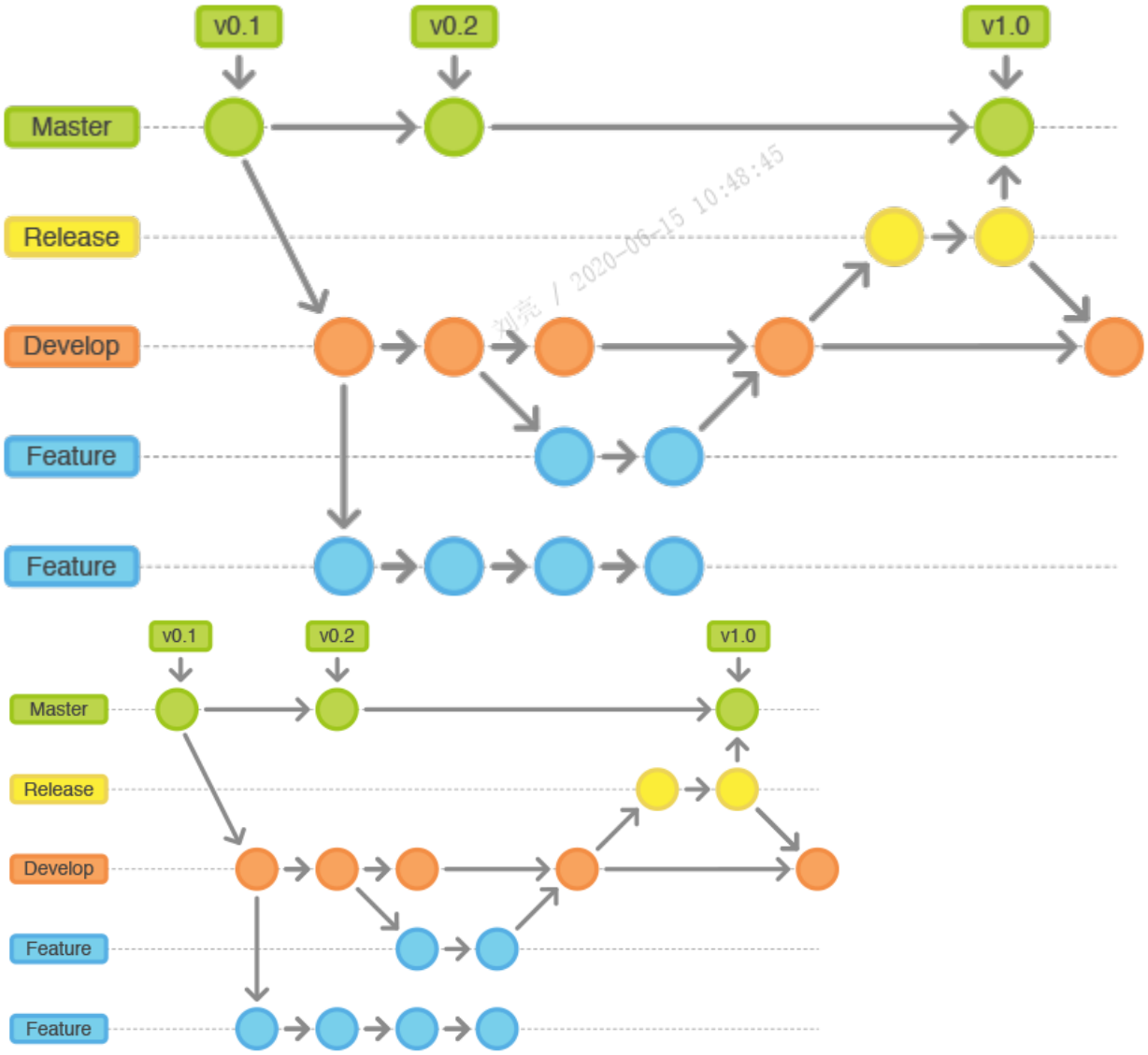


Release分支

分支名 release/*

Release分支基于Develop分支创建，打完Release分之后，我们可以在这个Release分支上测试，修改Bug等。同时，其它开发人员可以基于开发新的Feature
(记住：一旦打了Release分支之后不要从Develop分支上合并新的改动到Release分支)

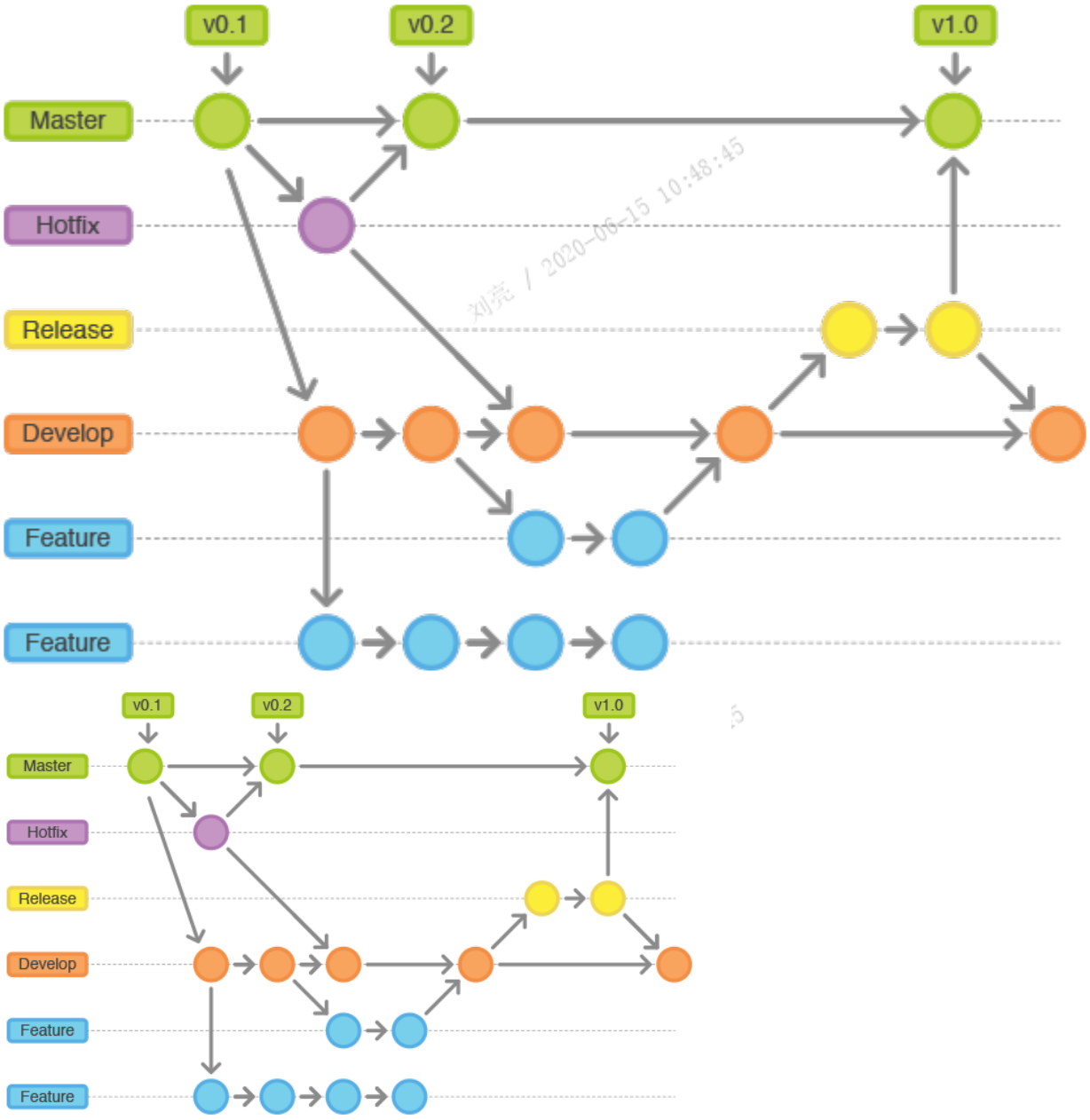
发布Release分支时，合并Release到Master和Develop， 同时在Master分支上打个Tag记住Release版本号，然后可以删除Release分支了。



维护分支 Hotfix

分支名 hotfix/*

hotfix分支基于Master分支创建，开发完后需要合并回Master和Develop分支，同时在Master上打一个tag

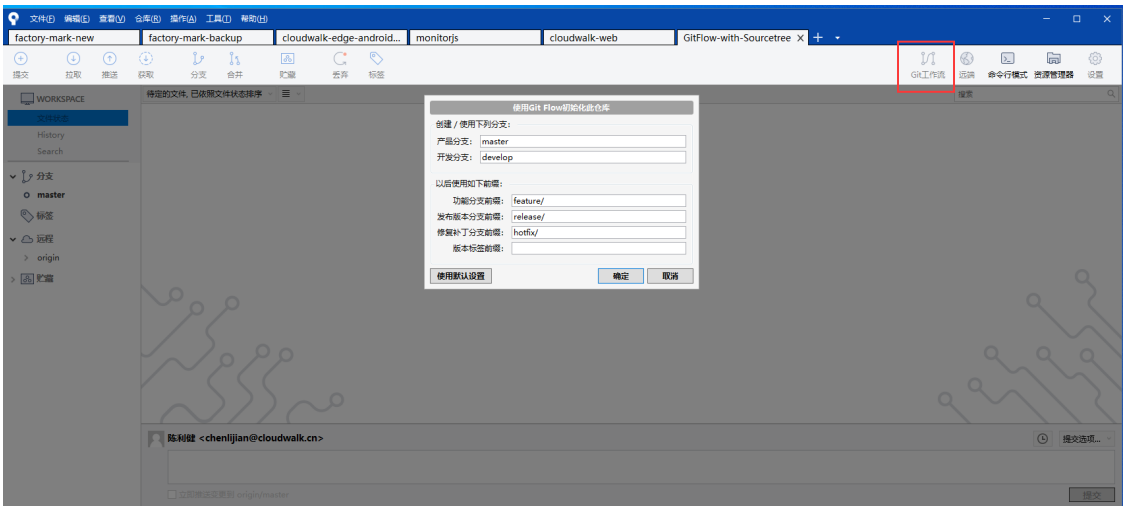


使用Sourcetree工具完成GItFlow

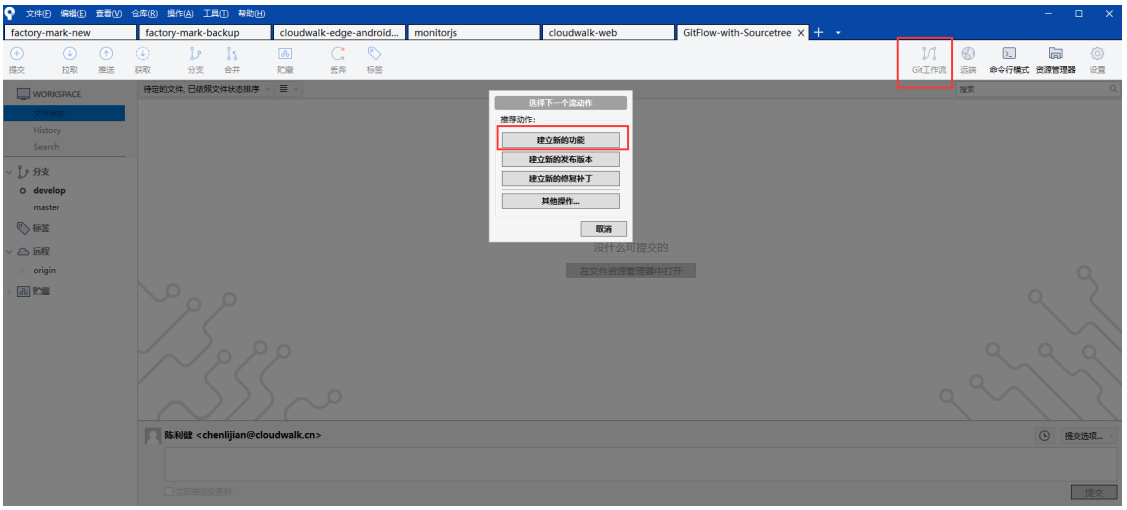
sourcetree是一个可视化的git工具，可以帮助我们

1. 专注与开发流程（开发新功能 - 提测 - bug修复 - 预发及上线 - hotfix等），而不必烦心于分支管理
2. sourcetree会自动帮助团队所有成员进行最佳的统一的分支管理

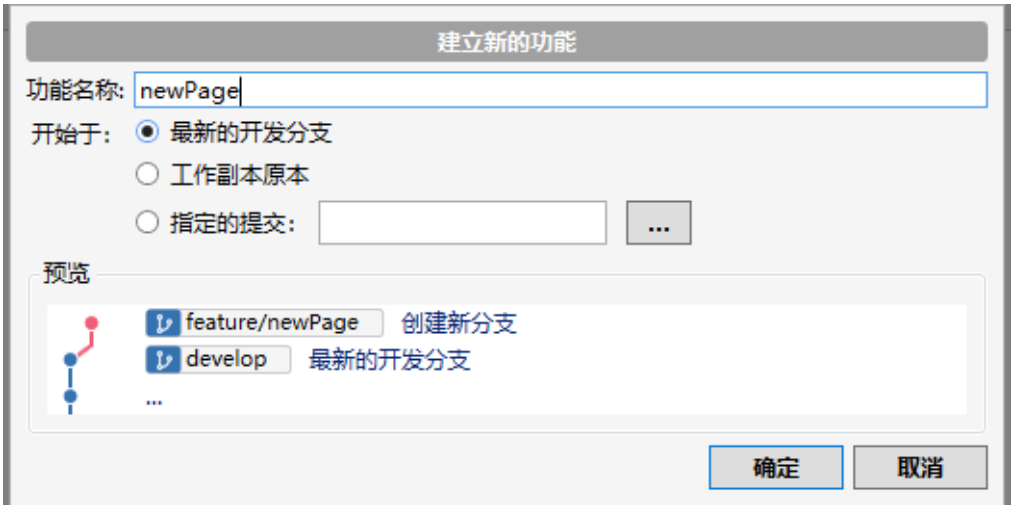
初始化GitFlow



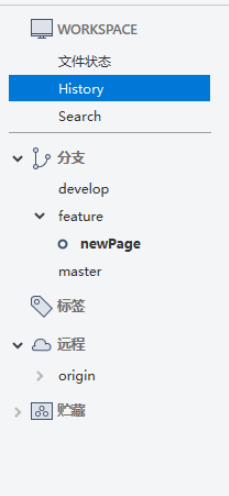
新增Feature分支



输入新功能名称（前缀不需要再加feature-），用分支预览上我们可以看到feature分支会被自动从develop分支上拉取出来

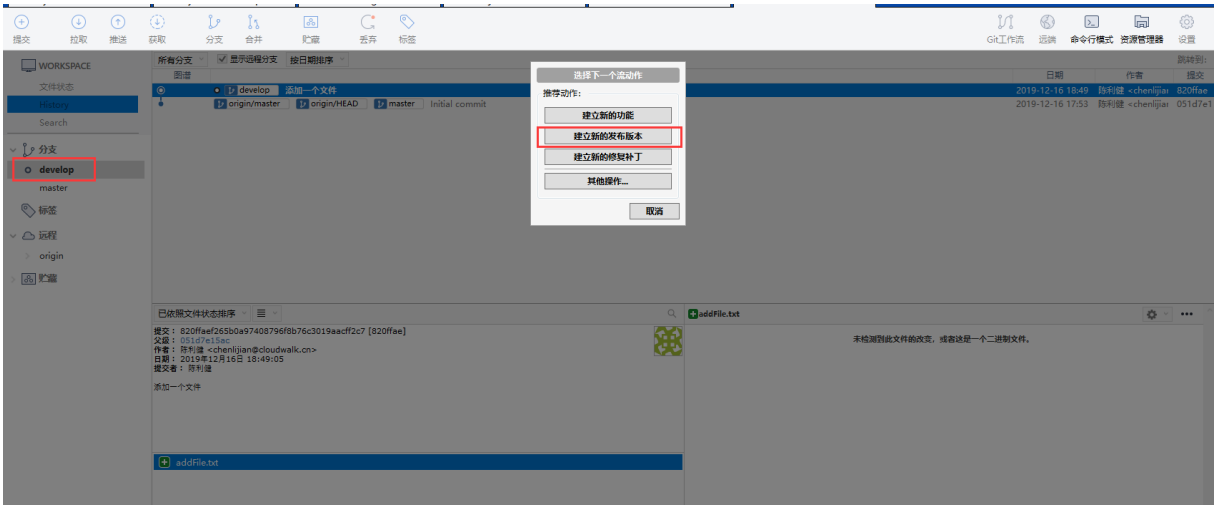


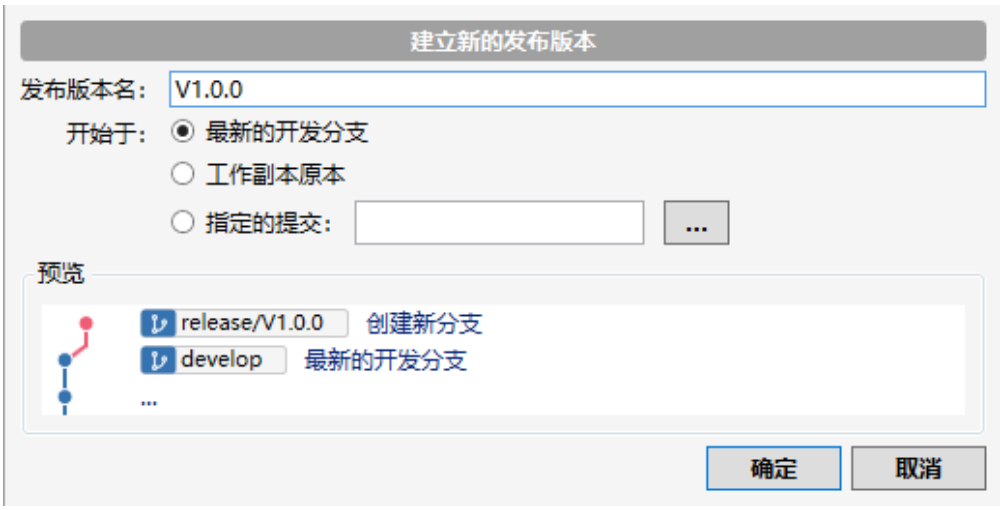
新建完成的新功能分支会自动归档到feature层级下

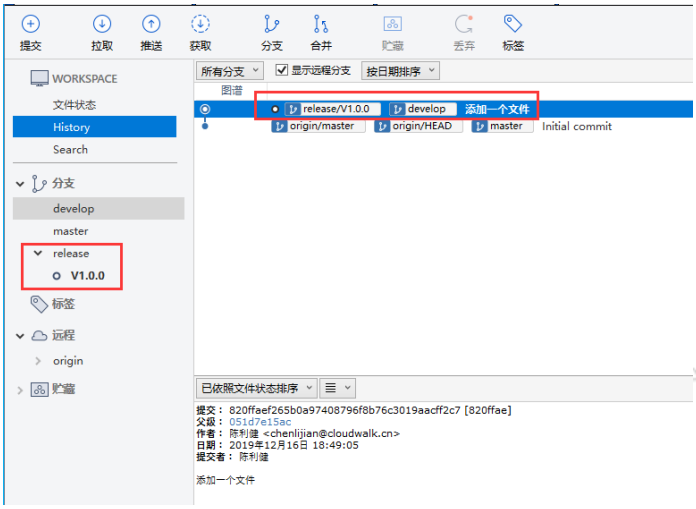


在该feature分支下完成一系列的代码编辑和本地提交后，进行完成操作

发布新的release分支，release分支和feature分支一样，也总是由develop分支发起的

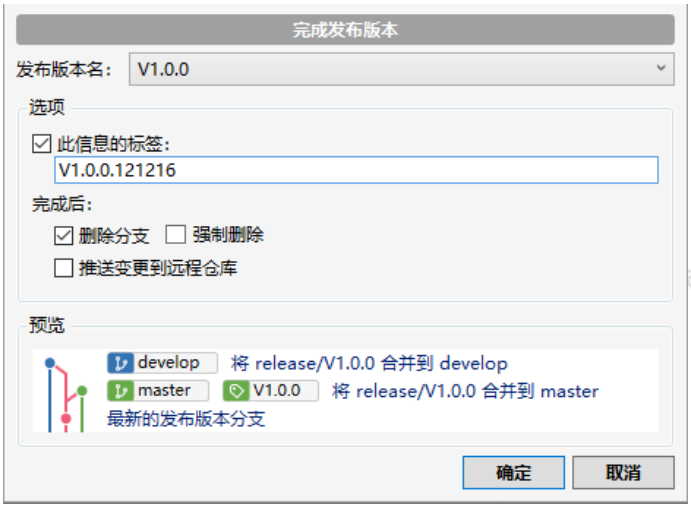






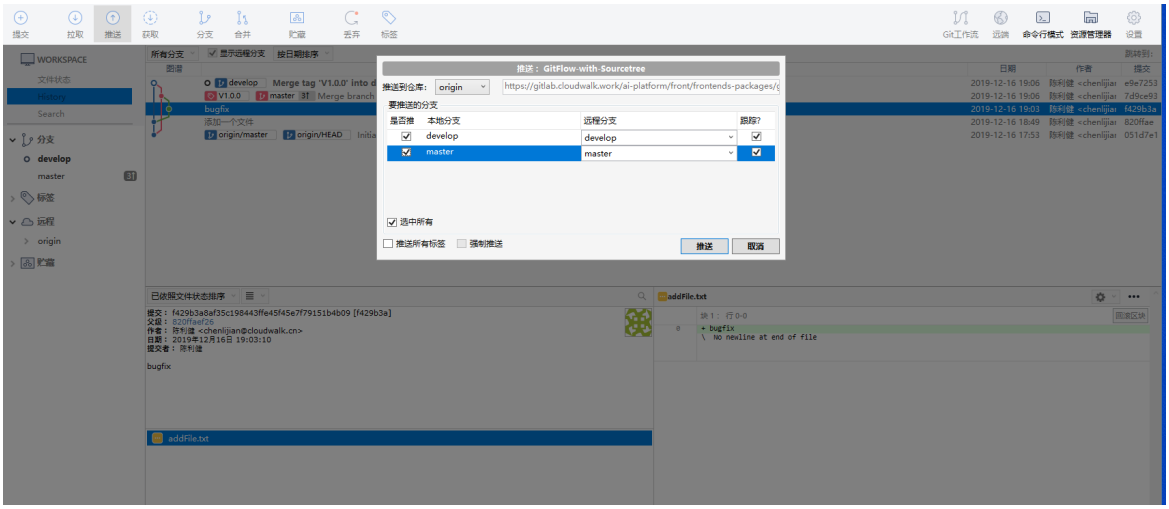
将release分支上的代码编译然后部署到测试环境中供QA测试。QA那边会反馈过来的bug，然后在这个release分支上做bugfix的工作。如果测试完成，可以进行完成发布版本

并打上标签



推送到远程操作，完成真正的发布

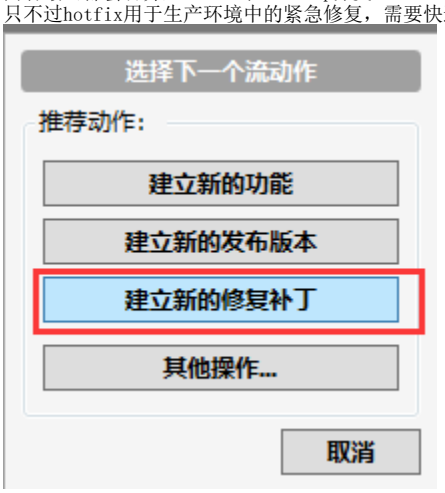
注意： 远程只推送develop和master，release分支只保存在本地，完成测试建议删除也可以暂时保留

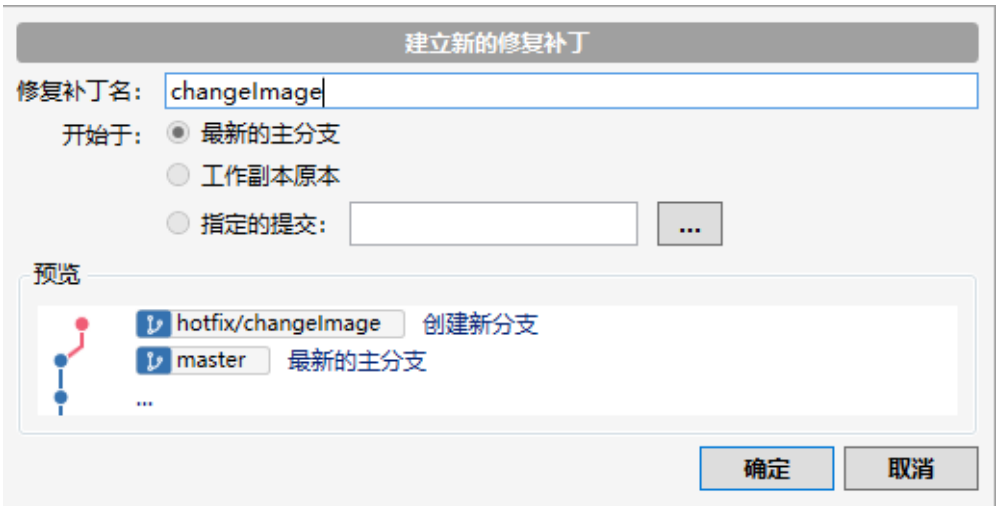


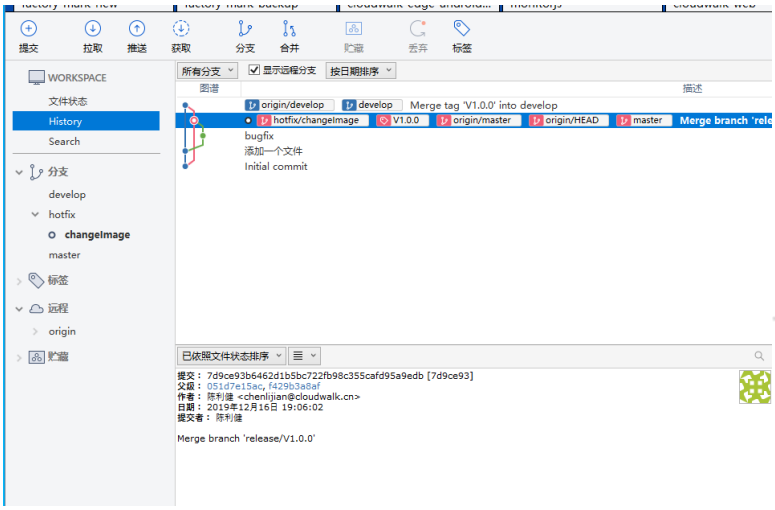
hotfix 分支（紧急修复线上版本bug）

git-flow假定当前产品线只需要维护一个版本，所以Git-Flow的hotfix总是基于最新的master分支里的版本来开辟的。
如果需要同时维护多个版本，那么就不应该用master分支了，可以多建几个分支比如1.x分支，2.x分支,3.x分支，
这样就可以同时维护3个版本线的产品，但是所带来的维护量就变的很大了。所以建议大家只维护一个版本来做CI（持续集成）

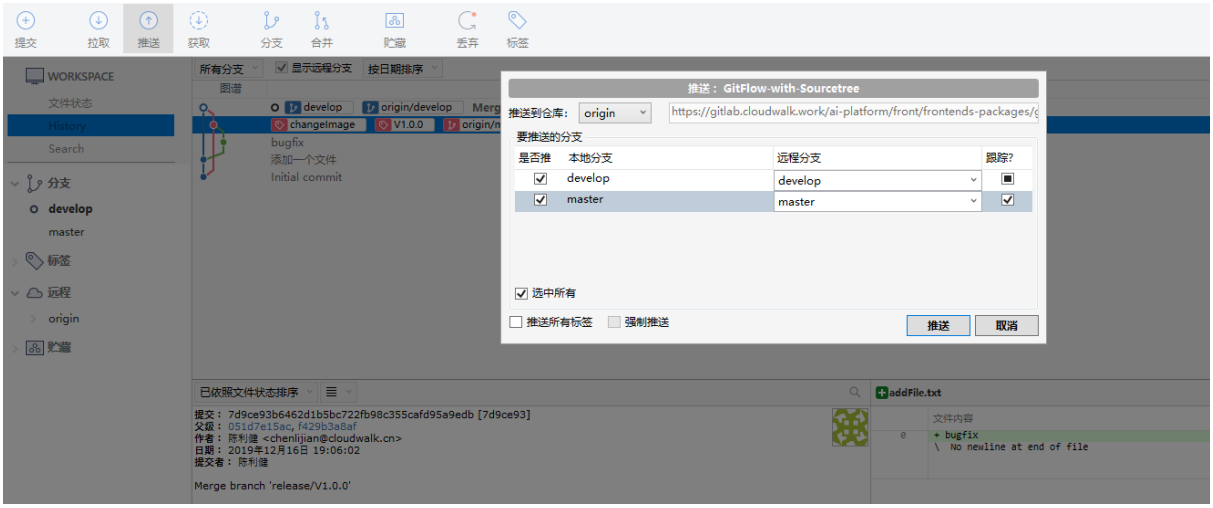
Git-Flow的hotfix分支和release分支有点像，区别在于release分支是由develop分支拉取出来的新分支，而hotfix分支是由master分支拉取出来的新分支，
两者最终都会合并入master和develop分支。
只不过hotfix用于生产环境中的紧急修复，需要快速响应和修复，减少Code ReView和QA环节的时间。







在这个hotfix上做bug修复



总结：

- 远程仓库仅仅应该存在两个分支，一个是master分支，存放线上（生产环境）版本，这个分支的代码总是可靠可用的；另一个是develop分支，这个分支用于日常开发。
- master分支上的内容不应直接提交，master分支总是应该由develop分支发布到release分支，经过QA测试确认可以上线后（期间可以在release分支上进行小幅bug修复提交更改），
  再完成发布新版本功能然后合并入master分支（发布成功后release分支会被删除，在release分支上所做的更改会自动合并到master和develop分支上）。
- feature 分支 下可以有多个feature同时在开发，并不影响。feature最终是提交到develop分支上的，release是从develop分支上拉取的，release分支是提交到master分支上的（develop分支不能直接提交到master上），他们几个并不冲突。允许在仍然有feature在开发的情况下从develop分支拉取到release分支。