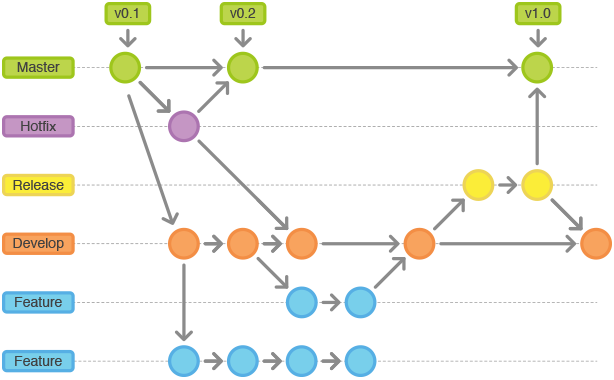
### 开发流程

1. **Git Flow**



* Master分支

生产环境使用分支， 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

开发分支，从Master上初始化，这个分支也不能直接修改，必须从其他分支合并代码。  
develop分支代码后续会进行自动构建（静态检查–单元测试–接口测试等），所以提交到develop分支的代码要求是可以工作的代码，不能导致代码构建失败 。  
测试环境

* 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。  
（功能上线后可删除该分支）

* Release分支

在Develop上测试通过后，基于Develop分支创建一个Release分支，Release分支上的Bug直接修改，完成后合并到Master和Develop分支。  
Release分支创建后develop分支可以合并下一次版本的代码。（功能上线后可删除分支）  
预发环境

* Hotfix分支

当我们在Master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。  
（功能上线后可删除分支）

搜索关键字：git flow  
 参考文章：http://www.open-open.com/lib/view/open1451353135339.html

1. **分支命名规范**

分支名称全部小写，可用符号-分隔单词  
master：固定分支  
develop：固定分支  
feature：feature/功能名称，如feature/demo  
release：release/版本号，如release/1.0.1  
hotfix：hotfix/bug名称，如hotfix/bug-xx

1. **使用角色**

* 开发人员  
  功能开发：在feature上编码，完成后发起合并请求feature->develop  
  测试Bug：继续在feature上修改编码，完成后合并到develop  
  预发环境：在release分支修改代码，完成后合并到develop、master  
  线上Bug：在hotfix分支修改bug，完成后合并到develop、master

注：代码合并到develop和master上代码需使用Gogs上的“合并请求”功能

* 项目负责人  
  功能开发：创建feature分支  
  合并请求：在Gogs上审核合并请求，通过后完成合并，如有冲突需在feature上先解决冲突  
  预发环境：从develop上创建release分支  
  上线发布：合并release到master分支，并设置版本Tag  
  线上Bug：发现线上紧急bug时，创建hotfix分支并协调开发人员
* 管理人员  
  使用统一管理账号创建项目，初始化master和develop分支，设置项目权限