项目源代码管理规范

2018/9/10

# 概述

## 代码管理工具

项目源代码采用Git进行管理，源代码库位于<https://github.com/dfjxps2/portal.dev>。

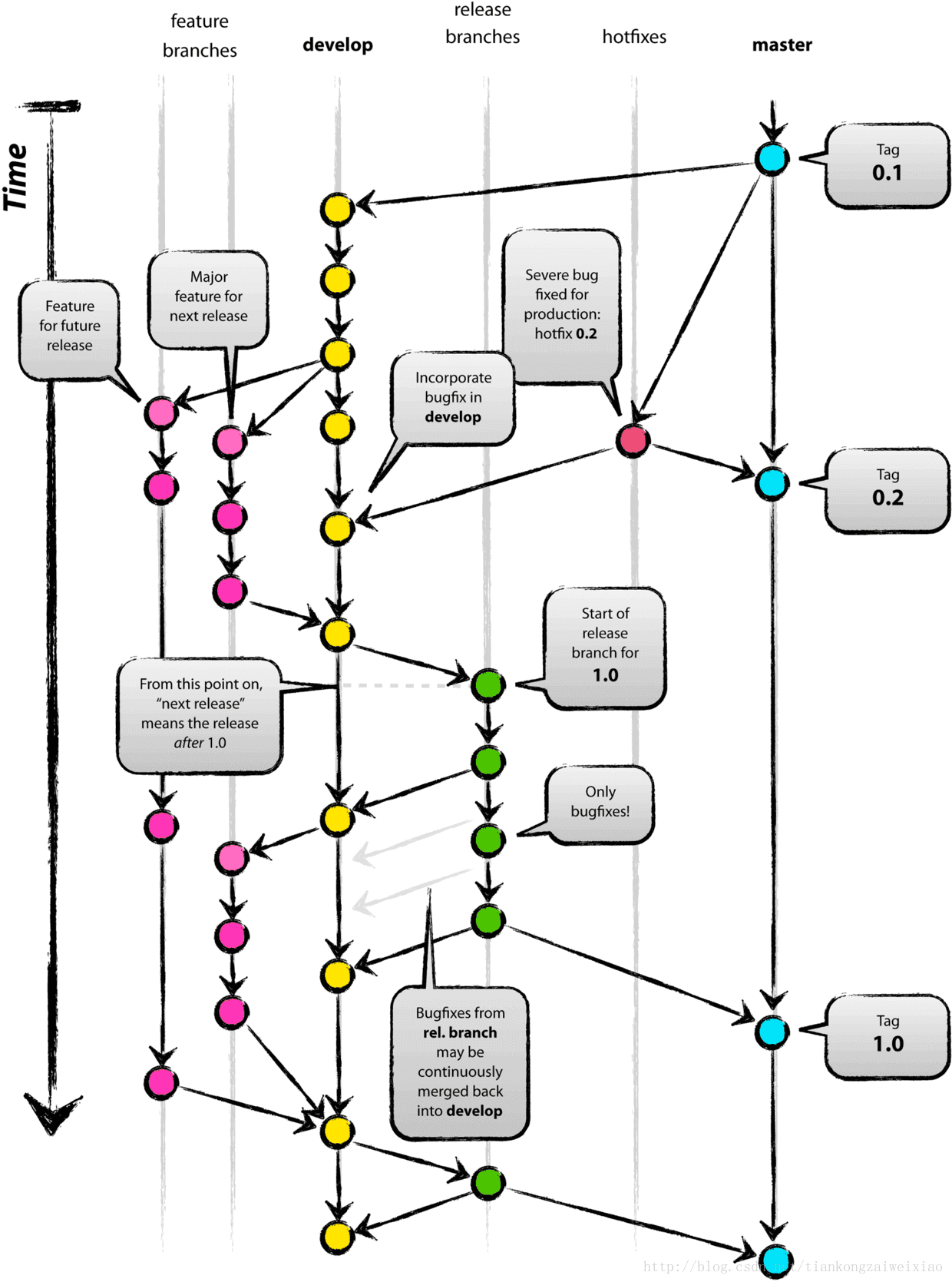
## 代码管理角色

参与项目源代码管理的角色包括：

1. 开发人员
   1. 负责创建新功能实现分支或问题修复分支，并在这些分支上修改代码。
   2. 将修改后的代码推送到代码库的相同分支上。
   3. 代码合并到开发分支后，删除自己的新功能分支或问题修复分支。
2. 代码审核员
   1. 负责审查开发人员推送到代码库的代码。
   2. 如果审查通过，负责通知发布管理员可以进行合并的分支。
3. 发布管理员。
   1. 负责将开发人员提交到新功能分支或问题修复分支上的代码合并到开发分支上。
   2. 负责创建发布分支。
   3. 负责将发布分支的代码修改合并到开发分支上。
   4. 负责将开发分支的代码修改合并到主分支上。

# 源代码管理过程

## 代码分支管理



### master分支

* master分支是项目代码管理的核心分支，所有新开发的功能和问题修复代码最终都将合并到master分支。
* master分支由发布管理员进行维护，其他人不可以对master分支进行修改。

### develop分支

* develop分支是项目的主要开发分支，代表项目的开发主线。
* 所有新功能开发分支和问题修复分支的代码最后都将合并到开发分支。
* 开发分支由发布管理员进行维护，其他人不可以直接将代码提交到开发分支。

### release分支

* release分支可以有多个。
* 当准备发布一个新的系统版本时，将会从开发分支创建一个release分支，这个分支上将包含本次发布的全部功能。
* 在release分支上进行系统发布之前的集成测试。
* 在测试中或发布后发现的问题，需要通过从release分支上创建的新的bug fix分支进行修改，在审核通过后，将代码合并到release分支。
* 合并到release分支的代码需要合并到开发分支。
* release分支由发布管理员进行维护，其他人不可以将代码直接提交到release分支。
* release分支采用‘r<版本号>’的方式进行命名。

### feature分支

* feature分支在实现新功能时创建。
* feature分支的代码在审核通过后合并到开发分支。
* feature分支上的修改可以分阶段多次合并到开发分支。
* feature分支上的代码全部合并到开发分支之后，feature分支应被删除。
* feature分支由负责该feature的开发人员创建并维护。
* feature分支采用‘T<开发任务编号>’的形式进行命名。

### bug fix分支

* bug fix分支用于修复在测试过程中或已发布的版本上发现的问题，每个问题对饮一个bug fix分支。
* 问题修复完成并通过审核后，代码将被合并到develop分支或release分支。
* 在代码被合并到develop或release分支后，原有的bug fix分支应被删除。
* bug fix分支由负责问题修复的开发人员创建并维护。
* bug fix分支采用‘T<开发任务编号>’的形式进行命名。

## 功能开发流程



## 问题修复流程



## 版本发布流程



# Git参考

## Git命令行工具

<https://git-scm.com/>

## Git常用命令

### 检查代码库状态

git status

### 查看提交日志

git log

### 克隆服务器代码库

git clone <https://github.com/dfjxps2/portal.dev>

### 从服务器拉取分支

git fetch origin <remote-branch-name>

### 切换本地分支

git checkout <branch-name>

### 从当前分支创建新分支

git checkout -b <new-branch-name>

### 提交代码到本地分支

git add <file-path>

git commit -m “commit message”

### 将本地分支推送到服务器

git push origin <branch-name>