Git 分支管理规范

（1.0.1）

北京时代亿信科技股份有限公司

2021 年 08 月

文档编号	V1.0.1	文档名称	Git 分支管理规范
编写人	王志鹏	编写日期	2021-08-16
审核人		审核日期	
批准人		批准日期	
修改人	修改日期	修改内容	
王志鹏	2021-08-16	编写规范初稿	
王志鹏	2021-08-24	按项目规模采用不同分支策略	

目录

第一章 引言

1.1. 目的

为了规范代码库分支管理和版本管理，使代码分支及版本结构清晰，方便维护，并避免由于维护造成的错误的版本发布等问题。

1.2. 使用范围

技术经理

基础平台研发工程师

产品研发工程师

项目研发工程师

质量保证人员

1.3. 相关术语

术语	解释
版本管理	为满足不同需求，对同一产品或系统进行局部的改进和改型所产生的产品或系统系列的变更情况进行记录、跟踪、维护和控制的过程
SVN	全称 Subversion，是比较流行的开放源代码的集中式版本控制系统，其特点是使用简单、权限控制灵活等
Git	一个开源的分布式版本控制系统，可以有效、高速的处理各种规模的项目版本管理，其特点是适合分布式开发、离线工作、速度快等
分支(branch)	
标签(tag)	

第二章 Gitflow 工作流

产品研发和实施过程中常用的环境主要有：

简称	全称	用途
DEV	开发环境	用于开发者调试使用
FAT	功能验收测试环境	用于测试环境下的测试者测试使用
UAT	用户验收测试环境	用于生产环境下的测试者测试使用
PRO	生产环境	

2.1. 分支命名

1. master 分支

master 分支是产品主分支，主要跟踪产品正式发布的代码历史。要确保 master 分支的稳定性，一般由 release 分支或 hotfix 分支合并，任何情况下都不允许直接在 master 分支上修改代码。

2. develop 分支

develop 分支是产品开发分支，主要跟踪产品研发的代码提交历史，始终保持最新完成以及 bug 修复后的代码。feature 分支和 release 分支都是基于 develop 分支来创建，并且将各自修改的代码合并回 develop 分支。

3. feature 分支

feature 分支是产品功能开发分支，其命名是以“feature/”开头的分支，命名规则为“feature/<特性名称>”，例如：feature/auth_module。

在功能开发阶段，首先以 develop 分支为基础来创建 feature 分支，待开发完成和自测通过后要合并到 develop 分支，然后 feature 分支可以自由决定继续保留还是删除。

4. release 分支

release 分支是产品预上线分支，用于部署到预上线环境（UAT），其命名是以“release/”开头的分支，命名规则为“release/<版本号>”，例如：release/4.1.0。

在发布提测阶段，首先将一组 feature 功能开发完成并合并到 develop 分支，然后以 develop 分支为基础创建 release 分支，做为 UAT 测试的代码基准。

在 UAT 测试阶段，如果（肯定）有 Bug 需要修复，则开发者直接在 release 分支上修复并提交代码。当测试通过后，合并 release 分支到 master 分支和 develop 分支，并在 master 分支打上相应的标签（例如：v4.0.1）。

5. hotfix 分支

hotfix 分支是紧急修复分支，其命名是以“hotfix/”开头的分支，命名规则为“hotfix/<线上缺陷名>”，例如：hotfix/login_invalid_parameter。

当线上系统出现紧急问题需要立即修复时，首先基于 release 或 master 分支创建 hotfix 分支，开发者在 hotfix 分支上修复问题和提交代码。待线上紧急问题修复完成并上线后，将 hotfix 分支代码合并到 release 分支或 master 分支并打上新标签，如有必要也可以合并到 develop 分支，然后就可以删除该 hotfix 分支。

6. 小型项目分支

小型项目是指在产品封版代码上进行少量定制化修改和第三方系统适配（用户源、信任体系和单点登录等）以及时间周期较短的项目。其项目分支主要跟踪项目定制的代码提交历史，其命名是以“project/”开发的分支，命名规则为“project/<项目名称>”。

在项目定制开发阶段，首先基于 release 或 master 分支创建 project 分支，开发者在 project 分支上开发定制功能并提交代码。

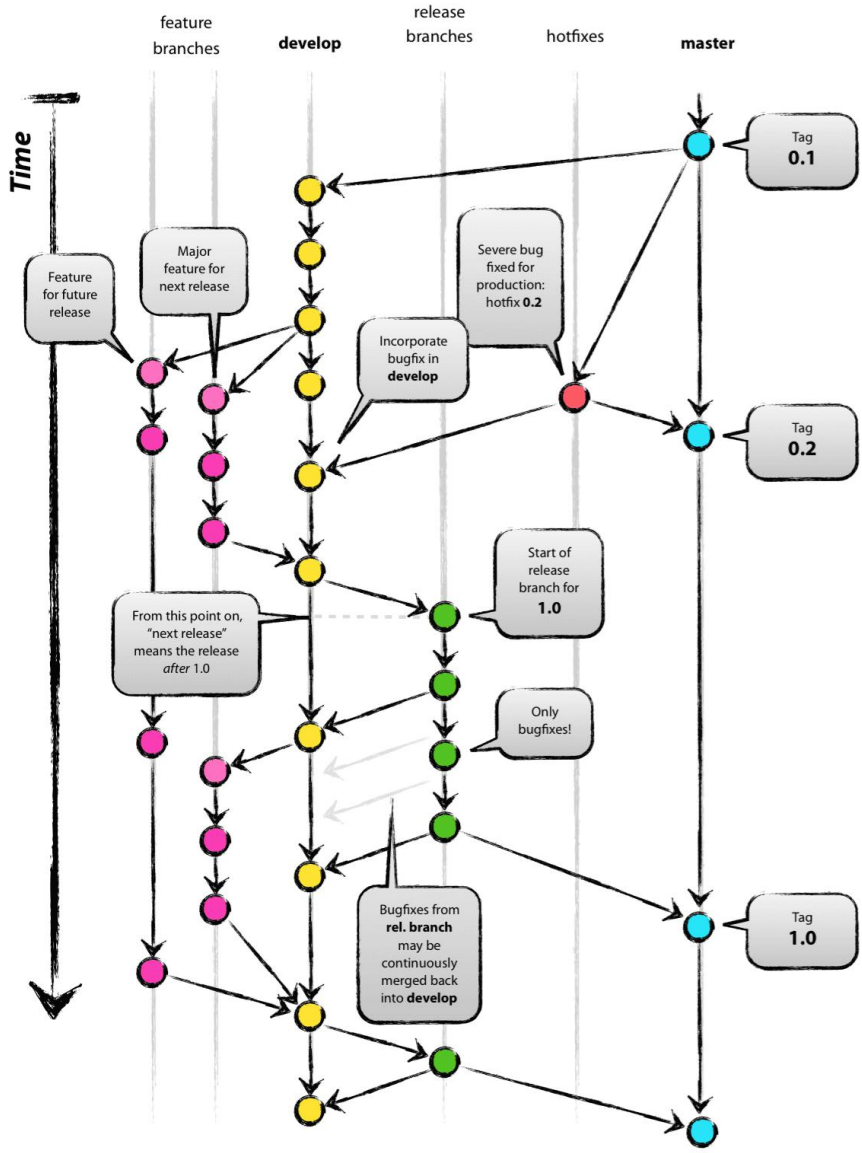
在项目 UAT 测试阶段，开发者在 project 分支打上相应的标签来标记不同的发布版本，例如：“v4.0.1-20210816”。

7. 大型项目分支

大型项目是指在产品封版代码上要完成定制化修改和第三方系统适配，还要按照客户需求定制开发许多新功能，产品研发周期长，需要持续演进的项目。其项目分支建议采用和产品研发相同的分支策略，针对不同的场景建立不同的代码分支。

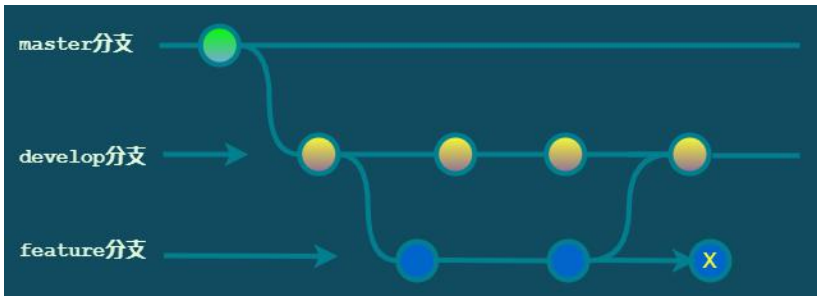
大型项目的代码分支，其分支命名都以“project/<项目名称>”为前缀，然后创建 master 分支、develop 分支、feature 分支、release 分支和 hotfix 分支等，各分支的使用流程和产品分支相同。

2.2. 使用流程

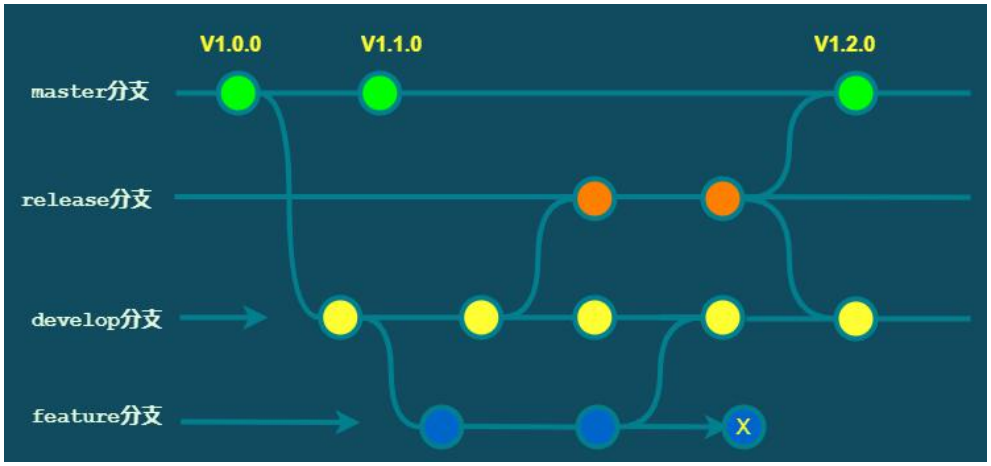


1. 新建代码仓库

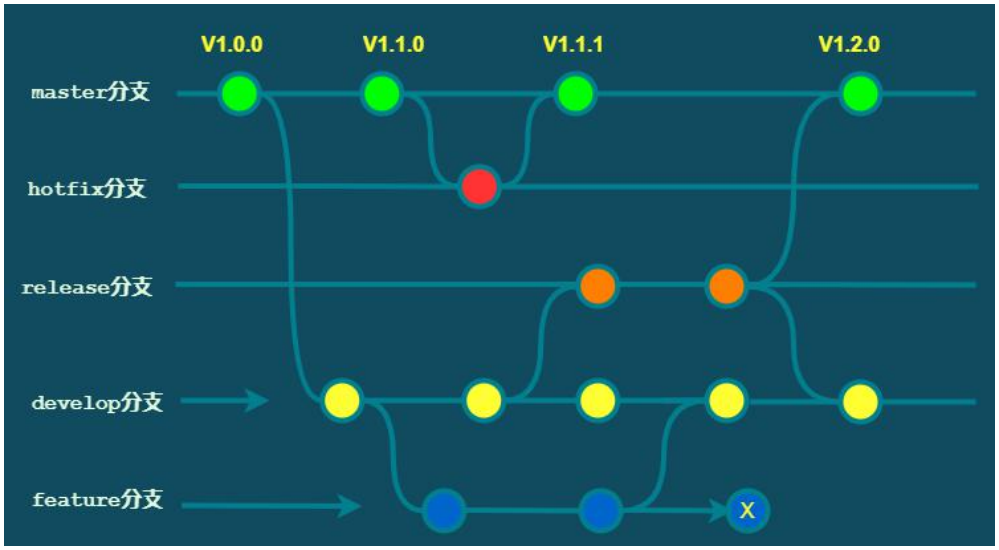
2. 开发新功能



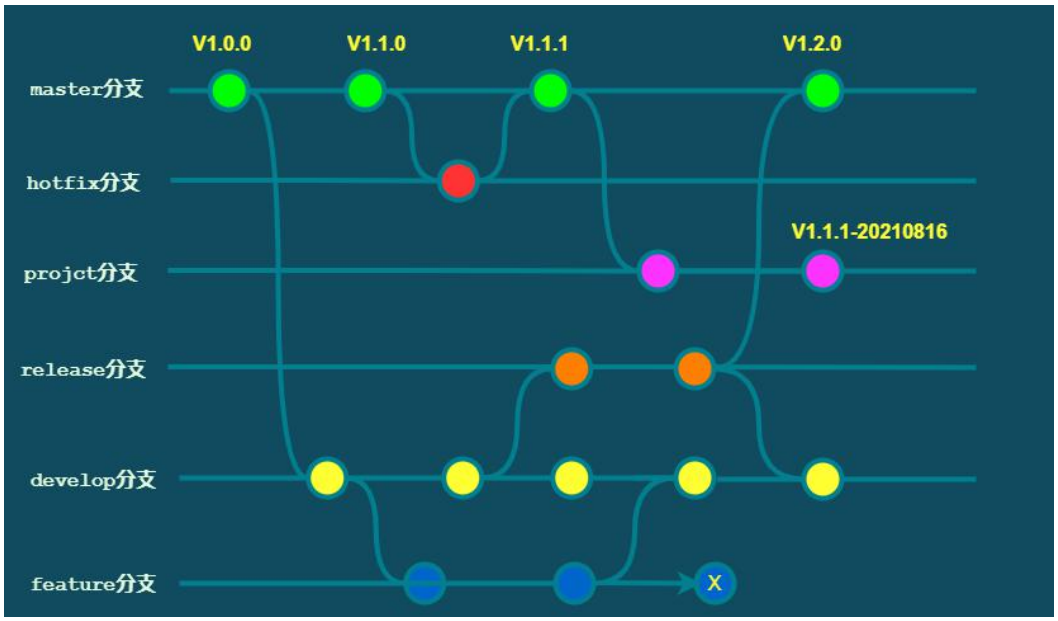
3. 发布测试



4. 线上修复



5. 项目定制


V1.0.0
V1.1.0
V1.1.1
V1.2.0
master分支
hotfix分支
V1.1.1-20210816
projct分支
release分支
develop分支
feature分支
X

第三章 Git Commit 格式

在一个团队协作研发的产品或项目中，开发者需要经常提交代码去实现新功能或修复缺陷等，而 Commit 日志就是每次提交解决什么问题的记录。

编写良好的 Commit 日志可以达到 3 个重要目的：

- 统一团队 Commit 标准，便于后续代码 Review、版本发布、自动化生成发布日志（Changes Log）
- 提供更多更有效的代码变更历史信息，方便快速预览以及快速合并代码
- 团队成员更容易了解代码变更的原因和内容

所有产品/项目的 Commit 日志的格式必须使用统一规范，增加代码变更内容的可读性，快速查看代码变更历史，提高团队协作开发的整体效率。

3.1. 约定定提交规范

现在比较流行的 Git Commit 日志方案是约定式提交规范（Conventional Commits），以 Angular 提交准则做为依据。约定式提交规范是一种基于提交消息的轻量式约定，提供了一组用于创建清晰的提交历史的简单准则，并且与 SemVer 相结合，在提交信息中描述新特性、Bug 修复和破坏性变更等。

约定式提交规范的日志格式如下：

<类型>(<作用域>):<小写空格><描述>

<空行>
[正文]
<空行>
[脚注]

注意：冒号后面必须有一个小写空格，类型和作用域可为多个，中间用逗号分隔。

1．类型（必填项）

类型必须是英文（小写），用于说明 Commit 的类别，必须为下列的一个或多个：

类型	说明
feat	新功能（feature）
fix	修复缺陷
docs	文档相关的变更
style	代码格式的变更，例如去掉空格、改变缩进、增删分号
build	改变构建流程，新增依赖库、工具等（例如 webpack 修改）
refactor	代码重构，未新增任何功能或修复任何缺陷
revert	回滚到上一个版本
merge	代码合并
sync	同步主线或分支的
test	增加测试或修改现有测试
perf	改善性能和体验的变更
ci	自动化流程配置变更
chore	不修改 src 或 test 的其余修改，例如构建过程或辅助工具的变动

2．作用域（必填项）

作用域是必填项，英文（小写），用于描述变更的模块范围，格式为“产品名/模块名”，例如：secmail/auth。

如果某个 Commit 修改多个模块，建议拆分成多个 Commit，以便更好地跟踪和维护。

3．描述（必填项）

描述是 Commit 目的的简短描述，不超过 100 个字符，结尾不要有句号，建议使用中文。

4．正文（可选项）

正文填写详细描述，主要描述改动之前的情况及修改动机。对于小的修改可不做要求，但重大需求、更新等必须添加正文来做说明。

5．脚注

脚注部分只用于两种情况：

(1) 不兼容变更

如果当前代码与上一个版本不兼容，则脚注部分以“不兼容变更”开头，后面是对于变更的描述、以及变更理由和迁移方法等。

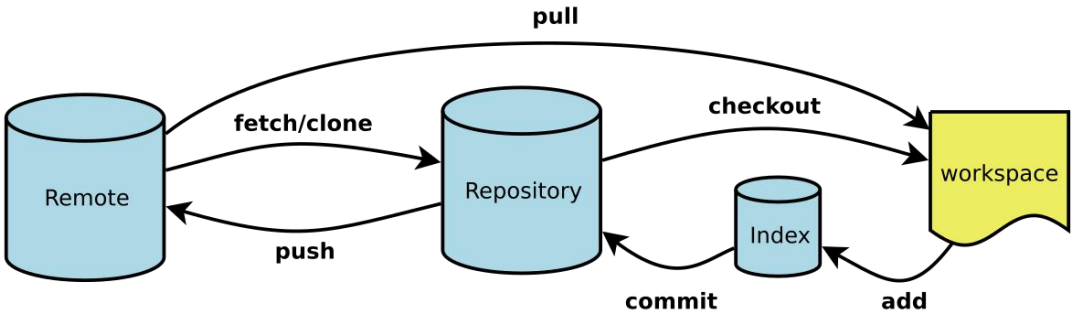
(2) 关闭 Issue

如果当前 Commit 针对某个 issue，则在脚注部分关闭这个 issue。

第四章 Git 使用

4.1. 基本概念

Git 是目前世界上最先进的分布式版本控制系统。



上图中的几个专用名词的含义：

- Workspace：工作区
- Index / Stage：暂存区
- Repository：仓库区（或本地仓库）
- Remote：远程仓库

4.2. 常用 Git 命令

参考“Git 常用命令清单”

第五章 参考资料

(1) Angular Git Commit Guidelines

https://github.com/angular/angular.js/blob/master/DEVELOPERS.md#-git-commit-guidelines

(2) A successful Git branching model

https://nvie.com/posts/a-successful-git-branching-model/

(3) 代码分支及版本管理规范

http://www.uml.org.cn/codeNorms/201502275.asp

(4) Git emoji 规范

https://gitmoji.dev/

https://github.com/carloscuesta/gitmoji

(5) IDEA 插件 -- Git Commit Message Helper

https://plugins.jetbrains.com/plugin/13477-git-commit-message-helper

(6) 阮一峰 -- Git 常用命令清单

http://www.ruanyifeng.com/blog/2015/12/git-cheat-sheet.html

(7)