# **范围**

该规范适用于Git工具的使用指导。支撑Git工具的规范、高效运转，帮助各产品提高开发效率和质量。

# **术语和定义**

## Git：是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。

## Git Project：Git版本控制系统上创建的代码仓库。

# **用户组使用及说明**

## 组-Group

## 产品组

#### 产品组名称：产品名称-group；

#### 组成员：产品CMO

## 子组-Subgroup

## 系统组

#### 系统组名称：系统名称-Subgroup；

#### 组成员：对应系统成员；

#### 说明：

#### 1）如系统下按模块或子系统创建Git项目，就需在系统的组里创建相应的子组；

#### 2）子组会自动从父组中继承成员与成员权限，子组的权限只能大于或等于父组权限；

# **Git各角色及所属权限**

|  |  |  |
| --- | --- | --- |
| **Git角色** | **英文简称** | **权限** |
| 系统管理员 | Owner | 组的管理员，他们可以访问组并具有组操作能力。 |
| 项目管理员 | Master | 超级开发人员，他们能够推送保护分支，部署到生产，定义仓库管理规则等。具有该仓库的最高权限，可以Push、Merge该项目中的所有分支。 |
| 开发人员 | Developer | 仓库直接共享者，并且可以访问仓库所有内容（不包含保护分支）。Developer只能Push、Merge除“受保护的”分支之外的分支。 |
| 测试人员 | Reporter | 只具有项目的读取权限，只能clone仓库，无法提交代码。但是可以对issues进行操作。 |
| 访客 | Guest | 访客并非项目的贡献人员，权限局限于查看与评论。 |

# **生命周期管理**

## Git项目创建

#### 新增Git项目时，由项目经理和产品CMO共同评估建库的必要性。评估通过后，由开发人员发起申请流程，由产品CMO进行创建。

#### 项目命名：[系统/模块名称]

#### 说明：项目命名中的系统名称需与IT应用架构发文的名称保持一致。

## Git项目维护

#### 包括Git项目权限变更，master管理员变更等。产品CMO负责Git项目的维护，需保证变更的合理性、准确性，并负责将变更信息及时知会用户。

## Git项目归档

#### Git项目不再需要修改，产品CMO需对项目进行归档，设置项目为只读。

## Git项目下线

#### 满足以下任一条件，且项目信息不再需要查询，产品CMO必须对配置库下线：

#### 应用系统下线一年；

#### 完成新旧项目切换后，旧项目已归档一年且不再需要查询；

## Git项目备份策略

#### 对于Git项目、用户以及组信息采取每天增量备份，每周日全量备份；备份数据保留6个月，备份由运维部门负责。

# **分支策略及使用规则**

## 业界常见的三种分支策略

下表是常见的三种分支策略，项目负责人可以根据自己的项目特性选择合适的分支策略

|  |  |  |  |
| --- | --- | --- | --- |
| **分支策略** | **说明** | **适合什么特性项目** | **案例** |
| **混合分支** | 同时使用5种类型的分支：master、develop、feature、release、hotfix。  [优点]  1流程清晰，分支管理严格。  2.多分支用途明确，减少冲突。  [缺点]  1.大量代码合并，操作比较繁琐。  2.不适合快速迭代。 | 1.开发人数多，解决代码合并冲突的工作量巨大  2.需要严格维护历史版本  3.适用于发布周期比较长的“版本发布”，发布周期可能是几周，几个月，甚至更长时间。 | 操作系统项目  开源基础组件  企业基础组件 |
| **多环境分支** | 每个环境都有对应的发布分支，不同环境可能同时在测不同的feature分支集合。  [优点]  1.方便不同环境的持续部署。  2.方便管理各个特性进入不同环境的时机。 | 1.需要多个环境验证  2.产品迭代前进，不用考虑不同客户企业使用不同的历史版本。 | 企业级Saas应用  企业信息化系统 |
| **主干分支** | 主要使用2种分支：master 、 feature，特殊情况可以多用发布分支release。  [优点]  流程简单，分支简单 | 1.开发人数少, 比较少代码冲突。  2.每天发布一个版本，甚至一天发布多个版本。 | 快速迭代产品  研究验证类项目  演示项目 |

## 三种分支策略的详细工作流

## 混合分支（release分支发布master稳定分支）

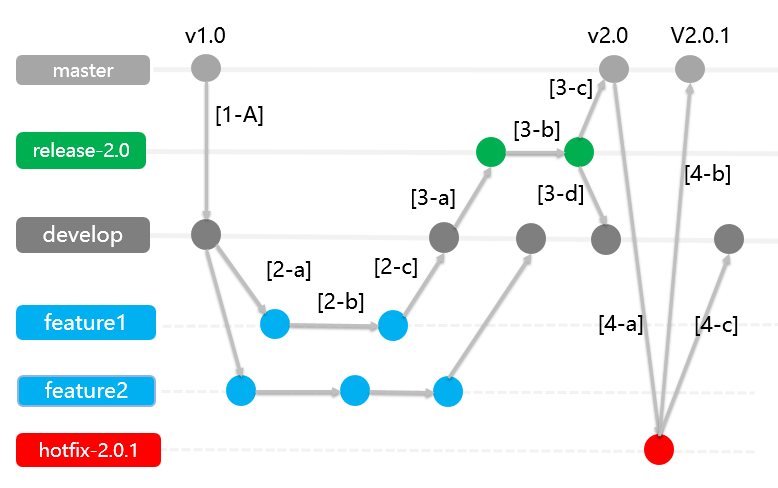


图. 混合分支

[1-a] 组长在master分支基础上创建出develop分支。

[2-a] 开发人员按需求从develop分支基础上创建出feature分支，

[2-b] 开发人员在feature分支上开发代码。

[2-c] 在某个feature分支开发完成且评审通过后,代码评审者将该feature分支合并进develop分支，并删除该feature分支。

[3-a] 在准备版本发布时，组长先通知开发人员暂停提交到develop分支，再从develop分支基础上创建出本次版本发布的分支release-<version> ，组长通知开发人员可以继续提交代码到develop分支。组长触发持续交付流水线对release分支构建打包工件，提测（运维工程师选择本次版本号的工件部署到测试环境，测试工程师在测试环境进行测试）

[3-b] 如果测试发现问题，开发人员从该release分支基础上创建出feature分支作bug修复，bug修复后再合并进该release分支，重新构建打包和部署测试。如果测试通过，部署到生产机

[3-c] 生产环境版本发布成功，组长将该release分支的代码合并进master分支，并在master分支打Tag（Tag为版本号）

[3-d] 将该release分支的代码合并回develop分支；然后将该release分支删除。

[4-a] 当生产环境发生严重缺陷需要紧急修复开发人员从master分支上创建出hotfix-<缺陷ID >

[4-b] 生产环境紧急修复后，需要将hotfix分支合并进master分支，并在master分支上打新的Tag。

[4-c] 并将该hotfix分支代码合并回Develop分支，然后将该hotfix分支删除。

## 多环境分支（UAT分支发布master稳定分支）

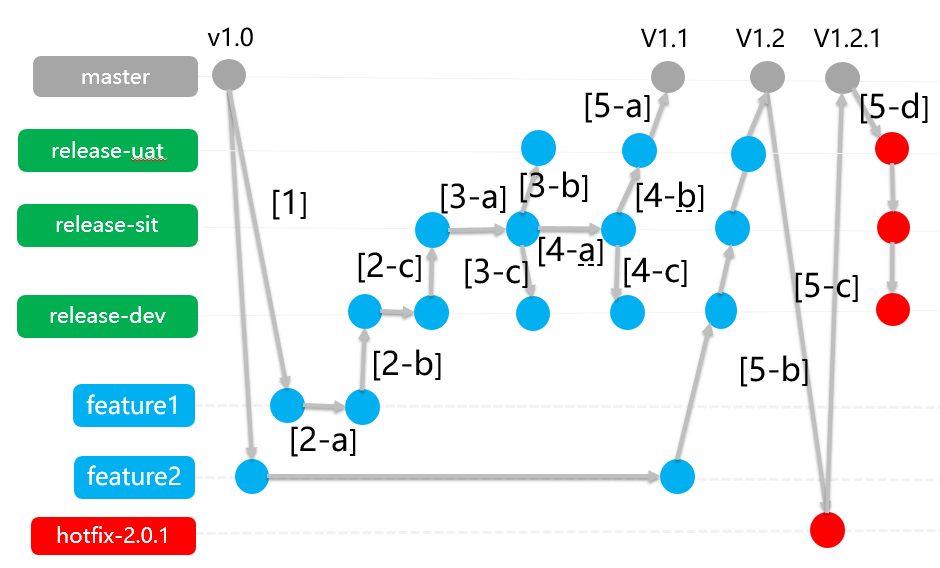


图. 多环境分支

[1] 开发人员从master创建特性分支feature1，进行功能开发。

[2-a] 开发人员feature1 完成后，提交到DEV分支，DEV分支会触发流水线进行构建部署到环境

[2-b] 如果DEV环境测试发现问题 ，修改bug提交DEV分支

[2-c] 在DEV环境测试稳定后，合并特性分支到SIT分支

[3-a] 如果SIT环境测试发现问题 ，开发人员直接在SIT分支上修改bug，提交

[3-b] SIT环境测试没有问题后，从SIT分支merge到UAT分支

[3-c] 同时从将SIT分支的bug修复代码merge到DEV分支

[4-a] 如果UAT环境测试有bug，开发人员直接在SIT分支上修改bug，提交

[4-b] 将SIT分支的bug修复代码merge到UAT分支

[4-c] 将SIT分支的bug修复代码merge到DEV分支

[5-a] 生产环境版本发布成功，组长将该UAT分支的代码合并进master分支，并在master分支打Tag（Tag为版本号）

[5-b] 当生产环境发生严重缺陷需要紧急修复开发人员从master分支上创建出hotfix-<缺陷ID >

[5-c] 生产环境紧急修复后，需要将hotfix分支合并进master分支，并在master分支上打新的Tag。

[5-d] 并将该hotfix分支代码合并回UAT、SIT、Develop分支，然后将该hotfix分支删除。

## 主干分支

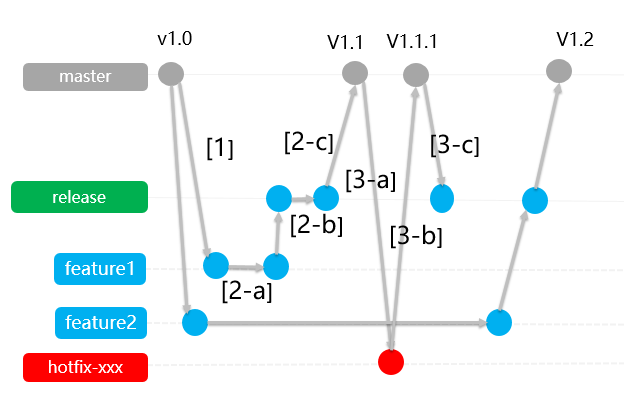


图. 主干分支开发法

[1] 开发人员从master创建特性分支feature1，进行功能开发。

[2-a] 开发人员feature1 完成后，提交到release分支，会触发流水线进行构建部署，提测

[2-b] 如果测试发现问题 ，修改bug提交release分支，重新提测

[2-c] 测试稳定后，正式发布，合并代码分支到master ，打Tag

[3-a] 如果发布后，生产环境测试发现问题 ，从master分支 checkout出hotfix-xxx分支，修改bug

[3-b] 修复bug后，打包发布，对master打Tag

[3-c] master修复代码merge到release分支

## 分支策略管理原则

1.分支策略的选择没有绝对的对与错，关键是要符合规模和发布节奏相匹配。如果开发中出现发布混乱、代码冲突过多，就要检查自己分支工作流是否需要调整

2.发布版本时，master分支一定要保持生产机代码一致，并对master打Tag标记。

3. master 分支中任何 commit 都是可部署的。

# **版本管理**

## 系统版本命名规范

## 系统版本命名规则及跃迁规则，请参见《美的IT开发配置管理办法》第7章节。

## Tag管理

## Tag管理要求：

#### 只允许master制定；

#### 保护规则：Tag一旦制定，不允许随意变动。

## Tag命名规范：

#### 测试阶段：Tag命名规则为：预发布版本号\_Test\_构建的日期和时间；

#### 发布阶段：Tag命名规则为：发布版本号\_Released；