**Git代码管理规范**

**基本原则：master 为保护分支，不直接在 master 上进行代码修改和提交。 release** **为发布分支。**

**开发流程：**

* 开发日常需求或者修复时，从 master 分支 checkout 创建 feat\_\* 功能分支 或 fix\_ 修复分支进行开发。
* 完成开发、自测，合并 master 分支到当前开发分支，提交测试（测试组在集成发布时也会先尝试合并 master 代码到提测分支）。
* 测试通过后，MR 开发分支到 **release** 。
* 发布上线成功后，按照规范更新 CHANGLOG 文档，记录该版本更新内容。
* 使用 **Merge Requests** ，请求将 **release** 分支合并到 master 。
* 由PM审核后，同意合并。
* 基于 master 分支 创建 tag ，在 **发行说明** 中填入 该版本的更新内容。

**分支管理**

* 当前版本分支开发完毕后需要将当前分支合并到release分支
* 合并后进行tags管理，在release分支上建当前版本的tag
* 预发和生产需要通过tag进行构建
* 上线后通过tag后的release分支合并到master

**master分支**

**保护型分支** 。需求迭代完成后将代码合并到该分支，该迭代才算正式完成。业务组负责人负责对该分支的管理。

**开发分支**

日常需求迭代或是线上bug修复，在开发过程中所使用的分支。从 master 分支 创建。

**命名规范：**

* 功能分支： feat\_ 开头。后面可以是功能名也可以是版本号。
* 修复分支： fix\_ 开头。后面跟功能名。

**release分支**

**release** 分支为保护型(一个项目仅有一个)，用于发布的分支。

* 发布时，先将需要发布的功能或修复分支以MR形式合并到 **release** 。再次过程可进行CR操作。
* 再创建 **tag** 进行发布(tag规范见下文)。
* 发布完成后，将 release 分支合并到 master 分支留存。

**TAG**

使用 jekins 发布均需要用 release 分支创建 tag 才可发布。同时，该tag 将作为回滚使用。

**tag命名规范：**

* 正常迭代： feat\_<版本或功能>\_日期[\_发布次数] 如： feat\_v1.0.0\_20200823\_02
* 修复bug： fix\_<bug名>\_日期[\_发布次数] 如： fix\_v1.0.0\_20200823

**Git Commit规范**

提交commit必须添加说明内容不可为空。

* 格式： actionType: 修改内容
* actionType 代表某次提交的类型，比如是增加一个新的feature还是修复一个bug。所有的actionType类型如下：
* feat : 新增feature
* fix : 修复bug
* docs : 仅修改了文档，比如README, CHANGELOG, CONTRIBUTE等等
* style : 仅修改了空格、格式缩进、都好等等，不改变代码逻辑
* refactor : 代码重构，没有加新功能或者修复bug
* perf : 优化相关，比如提升性能、体验
* test : 测试用例，包括单元测试、集成测试等
* chore : 改变构建流程、或者增加依赖库、工具等
* revert : 回滚到上一个版本
* 示例： fix: 修复内容

注意：actionType: 后面必须有且只有英文空格。