**GIT项目管理规范**

编写人：谭超鹏 2021.9.4

# 缘由



1. 规范项目版本管理
2. 可快速追溯项目可用版本
3. 防止正在开发中的代码提交到生产环境造成生产环境出现问题

# 分支模型



**master**——最为稳定功能最为完整的随时可发布的代码hotfix——修复线上代码的 bug

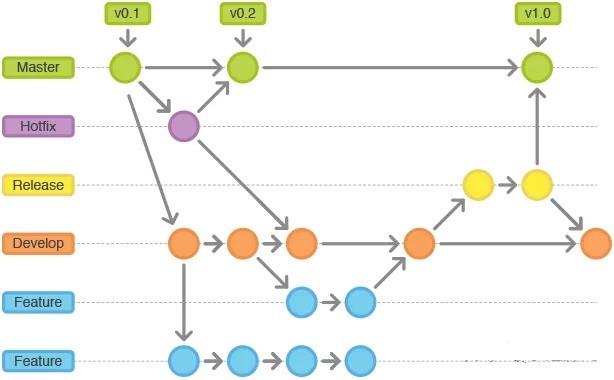
**develop**——永远是功能最新最全的分支feature——某个功能点正在开发阶段

release——发布定期要上线的功能

# 开发流程



**流程图示：**



1. 为每个开发人员 从 develop 创建一个 feature 分支；
2. feature分支开发完成后合并进develop分支
3. 从develop 创建一个 release 分支部署到测试环境进行测试；

# 测试功能



1. 从develop 创建一个 release 分支部署到测试环境进行测试；
2. 若发现了 bug，相应的开发人员就在 release 分支上或者基于 release 分支创建一个分支进行修复。
3. 修复bug后合并到release 分支，继续进行测试。

# 修复问题



当发现线上环境的代码有小问题或者做些文案修改时，遵循以下流程

1. 从master 创建一个 hotfix 分支进行修改
2. 修复后，部署hotfix分支到测试环境进行测试
3. 测试通过后，合并进 master 和 develop 并打上 tag，然后打包发布到线上环境。

# 发布上线



当确保某次发布的功能可以发布时，负责发布的人将 release 分支合并进 master 和 develop 并打上 tag，然后打包发布到线上环境。 建议打 tag 时在信息中详细描述这次发布的内容，如：添加了哪些功能，修复了什么问题。

# 分支命名



除了主要分支的名字是固定的之外，派生分支是需要自己命名的，这里就要有个命名规范了。强烈推荐用如下形式：

feature——按照功能点（而不是需求）命名；例：feature/anesRecord release——项目编号+发布时间命名；例：release/AIMS\_20210904

hotfix——采用项目编号+三段版本号，尾版本号累加命名 例：AIMS\_0.2.1 tag—— 采用项目编号+三段版本号命名 例：AIMS\_0.2.0



# 发布日期

固定功能发布时间：**周一，周四**晚上

功能阻断性bug发布：如果有备用解决方案，可安排在问题修复当天晚上进行发布。否则即时进行发布