

广东党建云

Git 版本管理规范

V1.0

| | | | |
|-------|------|-------|-----------|
| 文档密级： | 内部公开 | 产品名： | 广东党建云 |
| 编写人： | 林江春 | 编写日期： | 2019/7/15 |

内部资料 注意保密

1 分支管理

1.1 主分支

- ① develop: 保存当前迭代版本的最新开发分支，用于发布开发环境
- ② sit: 提测分支，用于发布 SIT 环境
- ③ sit_standby: 非当前迭代版本提测分支，用于发布 SIT 备用环境
- ④ master: 主干分支，随时可供在生产环境中部署的代码，用于归档

1.2 开发辅助分支

主要用于新功能的并行开发、对生产代码的缺陷进行紧急修复工作，开发根据情况自行创建，合并 master 后应该立即删除。

- ① 用于开发新功能时所使用的 feature 分支，基于 develop 创建
- ② 用于对旧模块功能进行重构的 refactor 分支，基于 develop 创建
- ③ 用于修复生产环境缺陷的 fixbug 分支，基于 master 创建（仅针对需紧急修复发布的缺陷）

分支按照开发需求/缺陷创建，原则上每个需求/缺陷创建一个分支。

分支命名规则：分支类型/开发功能_时间_需求或缺陷编号

- ① feature 分支: feat/function_20190701_planID
- ② refactor 分支: ref/function_20190701_planID
- ③ fixbug 分支: fix/function_20190701_bugID

1.3 发布辅助分支

主要用于按照需求迭代计划发布 UAT 进行验证确认，并发布生产的 release 分支，由管理员统一管理，每个发布辅助分支基于上一个版本创建。

分支命名规则：release_版本号_发版日期

2 开发流程

2.1 流程说明

➤ 需求开发

- ① 针对需求基于 develop 创建开发辅助分支并进行开发。
- ② 要发布至开发环境进行联调测试，需要把对应开发辅助分支合并至 develop 分支，由研发组长安排专人负责分支合并请求，合并后自行发布开发环境。
注意，只允许提交合并当前迭代版本范围内的需求对应的开发辅助分支。

- ③ 开发环境自测通过后要发布至 SIT 环境进行提测，需要把对应开发辅助分支合并至 sit 分支，由研发组长安排专人负责分支合并请求，由测试组发布 SIT 环境。
注意，只允许提交合并当前迭代版本范围内的需求对应的开发辅助分支。如非当前迭代版本范围内的需求需要提测，请合并至 sit_standby 分支发布至 SIT 备用环境进行测试。
- ④ SIT 测试通过后，将 sit 分支合并至发布辅助 release 分支，由研发组长安排专人负责分支合并请求，由运维组发布 UAT 环境。
- ⑤ UAT 环境验证通过后，由运维人员基于 UAT 环境发布生产环境。
- ⑥ 由管理员（研发组长）将发布辅助 release 分支合并至 master 分支，release 分支保留 1 个月后删除。
- ⑦ 当前迭代版本发布后，非当前迭代版本的开发辅助分支即可合并至 develop 分支，走正常发布流程。

➤ 缺陷修复

- ① 针对生产环境紧急缺陷基于 master 创建开发辅助分支并进行代码修复。
- ② 基于 master 创建临时发布辅助 release 分支。
- ③ 修复完成后将开发辅助分支合并至临时发布辅助 release 分支，由研发组长安排专人负责分支合并请求，由运维组发布 UAT 环境，由测试组进行验证测试。
- ④ 验证通过后，由运维人员基于 UAT 环境发布生产环境。
- ⑧ 由管理员（研发组长）将发布辅助 release 分支合并至 master 分支，release 分支保留 1 个月后删除。

2.2 GIT 基本操作

2.2.1 Git commit

commit 在什么时候都可以，但是不建议为了保存代码而 commit，每一次 commit 一定是代表代码开发进行到了某一个阶段，可以在后续开发或者合并代码出现错误的时候可以快速回到这个阶段。

每次提交必须要有提交注释，参考 AngularJS 的规范。

格式要求

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

- 消息只占用一行，任何行都不能超过 100 个字符
- 允许使用 GitHub 以及各种 Git 工具阅读消息
- 提交消息由页眉、正文和页脚组成，由空行分隔

➤ 页眉

<type>

必填项，代表某次提交的类型，比如是修复一个 bug 或是增加一个 feature，类型如下：

| 类型 | 描述 |
|----------|---|
| feat | 新增 feature |
| fix | 修复 bug |
| docs | 仅仅修改了文档，比如 README, CHANGELOG, CONTRIBUTE 等等 |
| style | 修改了空格、格式缩进、逗号等，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 |

<scope>

必填项，范围可以是指定提交更改位置的任何内容，如：

- 新增功能，**feat(user)**：在用户模块新增功能
- 或对代码进行重构，**refactor(dept)**：重构部门的缓存

<subject>

必填项，本次提交的标题。

➤ 正文

非必填项，本次修改的主要内容，如果本次提交内容很少，标题就能够说明就不需要填写正文，如果内容较多，需要列举修改内容清单。


➤ 页脚

必填项，指定代码审查人。如 审查人：范峰毓

示例：

refactor(dept): 新增[部门]缓存、消息

- 新增 [部门]缓存、消息
- 新增用户登录类型枚举: ClientTypeEnum
- 删除无用工具类: WxAppLoginUtil、UserUtil、DepartmentDeptUtil、DeptTreeUtil、LoginDeptUtil、OrgDeptInfoUtil、
- 删除拦截器: WebMvcConfig
- 删除无用常量类: Constants

🔗 父级 bbd1a836  ftr_refactor_cache

2.2.2 Git push

代码开发完成，并且自己本地测试 OK 了，再 push 到自己的开发辅助分支。

2.2.3 Git codeReview

- ① 提交 merge Requests 到 develop 分支前需要进行 codeReview
- ② 找有相关开发人员进行代码同行评审
- ③ 按照评审人的建议进行修改，修改后自测

2.2.4 Git merge

- ① 代码需要提交开发环境联调测试，提 merge requests 合并至 develop 分支，用于发布开发环境
- ② merge 之前保证自己的工作区是干净的
- ③ fetch, 更新本地仓库
- ④ 合并 develop，如果出现 merge conflict，找到相关开发人员一起解决，确保操作正确
- ⑤ merge 完成后，验证是否成功
- ⑥ 联调测试过程中如果有问题，把问题修改好后，再 push 自己的开发辅助分支，再提交 merge requests