1-3 GIT使用规范

- Git本地用**户名和邮箱必须和DevOps保持完全一致**,具体可参考GI**T用户配置** 提交commit时,必须<mark>必须关联JIRA工单号</mark>,并给出完整扼要的提交信息,具体可参考commit<mark>规范</mark>。
- 4. 分支的开发过程中,要经常使用rebase命令与上游分支保持同步。
- 5. 开发人员不可直接合并代码到master和dev分支, 需要通过merge request流程进行合并。

GIT用户配置

- 1. 在 Git 中设置用户名和邮箱必须和DevOps帐户配置完全一致
- 可通过git config命令配置本地git用户信息

```
# 配置用户名
```

zhangsan 为DevOps系统中的用户登录名

git config [--global] user. name zhangsan

配置用户邮箱

git config [--global] user.email zhangsan@cmsr.chinamobile.com

检查配置列表

git config --list

Gitlab用户SSH KEY配置

1、生成新密钥,注意邮箱信息必须和DevOps用户邮箱完全一致

```
ssh-keygen -t rsa -C "zhangsan@cmsr.chinamobile.com"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Zhangsan/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Zhangsan/.ssh/id_rsa.
Your public key has been saved in C:\Users\Zhangsan/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:5gmKIvelJ3CVmc7jKAO9AQkhxUq9oZ8l2H40tqh5ft8 "zhangsan@cmsr.chinamobile.com"
The key's randomart image is:
+---[RSA 3072]----+
0+0
0..0
00+ 0 +
+0 + ==
00 B++ S
. +*.+++ .
0.==00..0
1.=+00+0
0++0.. E
+---[SHA256]----+
```

2、登录GitLab, 进入: 用户 → Settings → SSH Keys 页面, 添加SSH Key

User Settings > SSH Keys

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to generate one or use an existing key.

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

9SS5+UFM0HfHzRKGzqvXJAu7Oznyw7OmuVgvyDvMi4QPjM0IUIankV+WL7aZzTMcib5OoqVG1 ZAkWznlw16jIVxOanZRrhHHvLNU5dA6FbCz933AUbpWTdWE+Wdjdxr5nhf4vh4Pes7qWDRbIrFo vgFNxm0x3Px+1CEdj0gdisIuCdeGjMVWM4nc4wcsnxEvPg0v3SNwNgeflaw0mjcrN4qU2oFP1es/ AdeF3qMZ5ch0XkQgm5U= "zhangsan@cmsr.chinamobile.com"

Title

"zhangsan@cmsr.chinamobile.com"

Name your individual key via a title

Add key

分支策略

主分支 - master

主分支只用来发布重大版本。每个代码库有且仅有一个主分支。开发人员不可直接push代码到主分支,只能通过merge-request的方式,将其他分支代码merge到主分支中。**主分支既是最新生发布生产版本**,每次发布后都需要为此次发布打上版本tag,后面会详细介绍版本tag名命规则。

开发分支 - dev

开发分支用于日常迭代开发。开发人员不可用直接推送代码到该分支,必须经够merge request的code review过程,以提升代码质量,降低bug风险。

功能分支 - feature

功能分支用于特定功能开发,该分支属于临时分支。开发成员(小组)开发某功能时,从dev分支分出一个feature-X名命的新分支,其中X为功能名,根据实际情况名命,建议采用JIRA工单号命名。功能开发完成后并回**DEV**分支,该分支即可废弃。

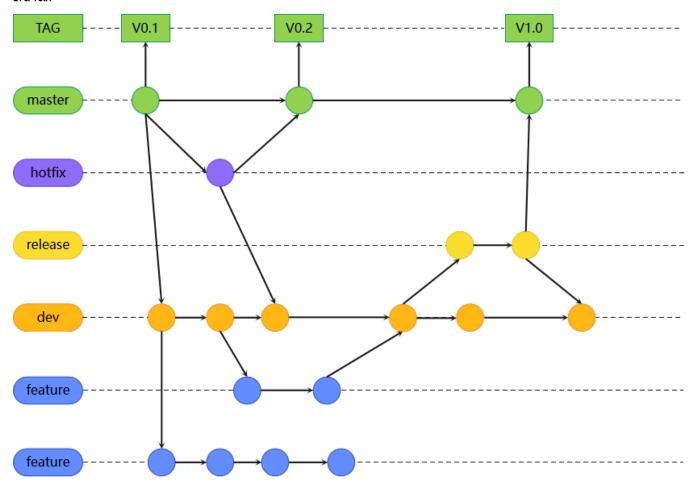
预发布分支 - release

在发布正式版本之前(即合并到master分支之前),我们可能需要有一个预发布的版本进行测试。从devops分支分出**release-X. X**分支,其中X. X为版本号。 当功能预览测试没有问题后,合并入主分支。为主分支合并打上**专属版本tag**,该release分支废弃。

热修补分支 - hotfix

一个版本正式发布以后,难免会出现bug。就需要创建一个分支,进行紧急bug修补。此类分支是从master主分支上分出。修补结束后,再合并进master主分 支和dev开发分支。此类分支以**hotfix-**开头名命。

GitFlow:



commit规范

提交commit时,必须遵循以下规则:

- 1. 提交信息<mark>必须关联JIRA工单号</mark>,这样可以做到提交信息和JIRA任务的关联,方便后续各阶段任务代码跟踪;
- 2. 规范提交信息,必须提供完整扼要的提交信息,不能使用"update"、"问题修复"等无具体意义的提交信息;
- 3. 根据实际提交情况罗列本次提交的具体修改点。

JIRA工单号,必填



不超过50字以内的提要

- 1. 罗列出改动原因、主要变动以及需要注意的问题
- 2. 罗列出改动原因、主要变动以及需要注意的问题
- 3.



变更详细列表,建议填写

名命规则

分支名命规则

- 主分支和开发分支名称不可修改,分别以master和dev名命
- 2. 功能分支,以**feature-**为前缀,后面跟当前功能点名称名命,例如:feature-login
- 3. 预发布分支,以**release-**为前缀,后面根当前形发布的版本号名命。例如:release-2.0.1 4. 热修补分支,以**hotfix-**为前缀,后面跟jira ticket号名命,例如:bugfix-CDPD-125

代码版本规范

版本发布时,一般由产品经理或项目负责人指定里程碑版本号。程碑版本号可以跳版本进行发版,也可以无中间版本号。比如,可以直接发布beta(公测版) 或者GA(正式版),但尽可能遵循里程碑版本顺序进行发版和版本管控。

版本号名命(tag名命)规则

```
release-x.x.x-20200510_beta
                       |- :base|alpha|beta|RC|Release
 - release
```

布正式版本之前(即合并到master分支之前),我们可能需要有一个预发布的版本进行测试。从dev分支分出**release−X. X**分支,其中**X. X**为版本号。当功 览测试没有问题后,合并入主分支,为此次合并打上**专属版本tag**,该分支废弃。