工作流程作业指导书

版本管理

|  |  |  |  |
| --- | --- | --- | --- |
| 审 核: |  | 日 期: |  |
| 批 准: |  | 日 期: |  |

|  |  |
| --- | --- |
| 版 本号 |  |
| 密 级 | 内部 |

修改履历

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 作者 | 修订要点 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 4](#_Toc15729)

[1 分工协作 5](#_Toc13039)

[1.1 角色化分 5](#_Toc29534)

[1.1.1 架构 5](#_Toc15815)

[1.1.2 开发 5](#_Toc12315)

[1.1.3 测试 5](#_Toc18016)

[2 版本和分枝 5](#_Toc22015)

[2.1 固定分枝 5](#_Toc8523)

[2.1.1 主分枝(master) 5](#_Toc15520)

[2.1.2 开发分枝(dev) 6](#_Toc28643)

[2.2 临时分枝 6](#_Toc3975)

[2.2.1 上线计划分枝(release) 6](#_Toc26816)

[2.2.2 个人功能分枝(feature) 6](#_Toc18263)

[2.2.3 bug分枝(bug) 6](#_Toc29886)

[2.2.4 补丁分枝(hotfix) 7](#_Toc14643)

[2.3 语义化版本规范 7](#_Toc19780)

[2.4 版本号定义 8](#_Toc22131)

[2.4.1 新项目开始 8](#_Toc28075)

[2.4.2 项目初次对外上线后 8](#_Toc18153)

[2.4.3 项目改版 8](#_Toc30963)

[2.4.4 功能叠加的开发版本定义为 8](#_Toc6710)

[2.4.5 开发版本bug定义为 9](#_Toc32309)

[2.4.6 生产环境中的bug定义为 9](#_Toc28944)

[3 操作规范 10](#_Toc6502)

[3.1 操作规范流程说明 10](#_Toc17762)

[3.2 操作规范说明 11](#_Toc7013)

[3.2.1 架构 11](#_Toc28837)

[3.2.2 开发 15](#_Toc31609)

[3.2.3 测试 20](#_Toc18929)

[3.3 命名规范 21](#_Toc9630)

[3.3.1 提交内容规范 21](#_Toc16779)

[3.3.2 合并请求规范 22](#_Toc5202)

[3.3.3 tag标签规范 22](#_Toc19973)

# 

# 分工协作

## 角色化分

从平台现有的组织结构，明确出平台的角色及角色职责定义如下:

### 架构

创建基线版本master分枝；

基于master分枝创建临时分枝release(根据版本计划进行创建)；

基于master分枝创建开发分枝dev；

制定版本号和版本计划；

根据版本号和测试上线情况tag版本；

### 开发

基于临时分枝release，以版本号和功能为单位，创建feature个人分枝；

在个人分枝测试完成后，将分枝合并到dev分枝；

将合并新功能后的dev分枝发布到开发环境，提供联调和合并代码测试;

根据版本计划，将个人对应feature分枝提交合并请求到测试分枝release；

### 测试

接收开发提交的合并请求进行测试环境代码合并;

确认当前版本后，将release分枝代码发布到测试环境，进行测试验证；

测试验证通过后，将测试分枝release提交合并请求到master分枝给架构；

# 版本和分枝

## 固定分枝

### 主分枝(master)

用于存放对外稳定的发布版本，代码库有且只有一个主分支，作为代码库的基线版本，其他分枝都来源于此分枝。同时用来发布生产TAG版本。

master分支需要有权限设置，不允许开发者直接对master分支代码进行修改和提交。每次发布需要打上Tag。

### 开发分枝(dev)

该分枝从release分枝检出，用于版本计划的开发，开者从release分枝检出个人分枝检出自己的功能性分枝feature，feature上完成功能开发后，合并到此分枝上，进行联调和开发环境合并后测试。

## 临时分枝

### 上线计划分枝(release)

该分枝从master分出，用于版本上线前的验收，主要用于阶段性版本完时，合并开发人员的功能分枝，合并完成后发布测试环境进行封版验收。验收通过后，提交master合并请求，进行生产发布。

该分枝只允许权限范围内人员审核合并。

### 个人功能分枝(feature)

是开发人员为了开发某种功能，以release分支为基础创建，开发完成后，通过合并请求方式再并入release分支，合并完成删除该分支。

### bug分枝(bug)

在功能开发后，在验收环境(release)中产生的bug，以合并的release分枝为基础创建bug分枝进行bug修补。

修复bug后，开发人员需要将此分枝自行合并到dev分枝；

并且将此bug分枝向测试发起合并请求到release分枝进行回归测试。验收通过后，删除bug分枝。

### 补丁分枝(hotfix)

软件正式发布以后，出现bug，以发布的TAG分支为基础创建hotfix分支，进行bug修补。

修复hotfix后直接发布到测试环境进行验证，验证通过后根据情况如下处理：

1. 如果hotfix紧急，将hotfix分枝直接打出tag标记，并将tag标记发布生产。如果不是紧急的问题，归并hotfix到当前版本计划release分枝中，并将修改点提交测试需求验证。
2. 归并hotfix到当前版本计划，提交hotfix分枝的合并请求到dev和release分枝，然后删除hotfix分枝。

版本号规范

## 语义化版本规范

版本格式：主版本号.次版本号.修订号-yyyymmdd，版本号递增规则如下：

1. 主版本号：当你做了不兼容的 API 修改
2. 次版本号：当你做了向下兼容的功能性新增
3. 修订号：当你做了向下兼容的问题修正
4. 版本计划日期

## 版本号定义

### 新项目开始

主线版本号定义：0.X.X-general-yyyymmdd

* 主版本号为0，表示项目的起步
* 次版本号为由0开始递增，表示功能的叠加
* 修订号由0开始递境，表示功能的性的bug修复
* general为项目归属，例: general表示通用版本， henan表示河南版本
* yyyymmdd版本计划日期

### 项目初次对外上线后

主线版本号定义：1.X.X-general-yyyymmdd

* 主版本号为1，表示项目发布的正式版本计划
* 次版本号为由0开始递增，表示功能的叠加
* 修订号由0开始递境，表示功能的性的bug修复
* general为项目归属，例: general表示通用版本， henan表示河南版本
* yyyymmdd版本计划日期

### 项目改版

主线版本号定义：2.X.X-general-yyyymmdd

* 主版本号由2开始递境，表示项目改版的改版次数
* 次版本号为由0开始递增，表示功能的叠加
* 修订号由0开始递境，表示功能的性的bug修复
* general为项目归属，例: general表示通用版本， henan表示河南版本
* yyyymmdd版本计划日期

### 功能叠加的开发版本定义为

feature-X.X.X-general-yyyymmdd-XXX-X

* feature表示增加新的功能
* X.X.X表示当前主线版本
* general为项目归属，例: general表示通用版本， henan表示河南版本
* 最后的XXX表示开发人员的名字别名缩写
* YYYYMMDD表示版本计划的提交日期
* X表示提交序号，当该版本出现多次提交时修改，从1开始

### 开发版本bug定义为

bug-X.X.X-general-X-yyyymmdd-XXXX

* bug表示验收中的bug
* X.X.X表示当前主线版本
* general为项目归属，例: general表示通用版本， henan表示河南版本
* X表示在禅道中对应的bug的id编号
* YYYYMMDD表示版本计划的提交日期
* XXXX表示开发人员的名字别名缩写

### 生产环境中的bug定义为

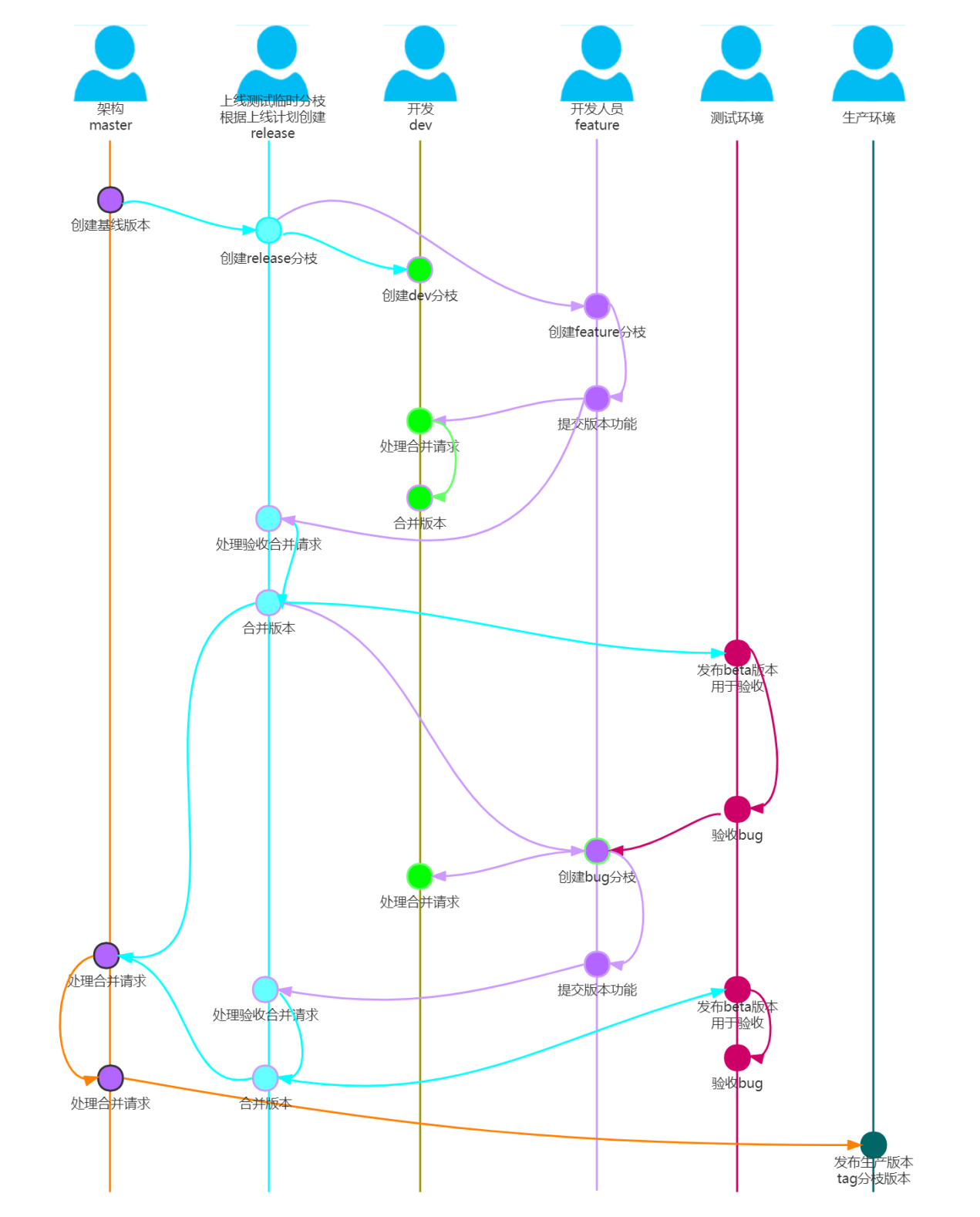
hotfix-X.X.X-general-X-yyyymmdd- XXXX

* hotfix表示生产中的bug
* X.X.X表示**生产的**主线版本
* general为项目归属，例: general表示通用版本， henan表示河南版本
* X表示在禅道中对应的bug的id编号
* YYYYMMDD表示版本计划的提交日期
* XXXX表示开发人员的名字别名缩写

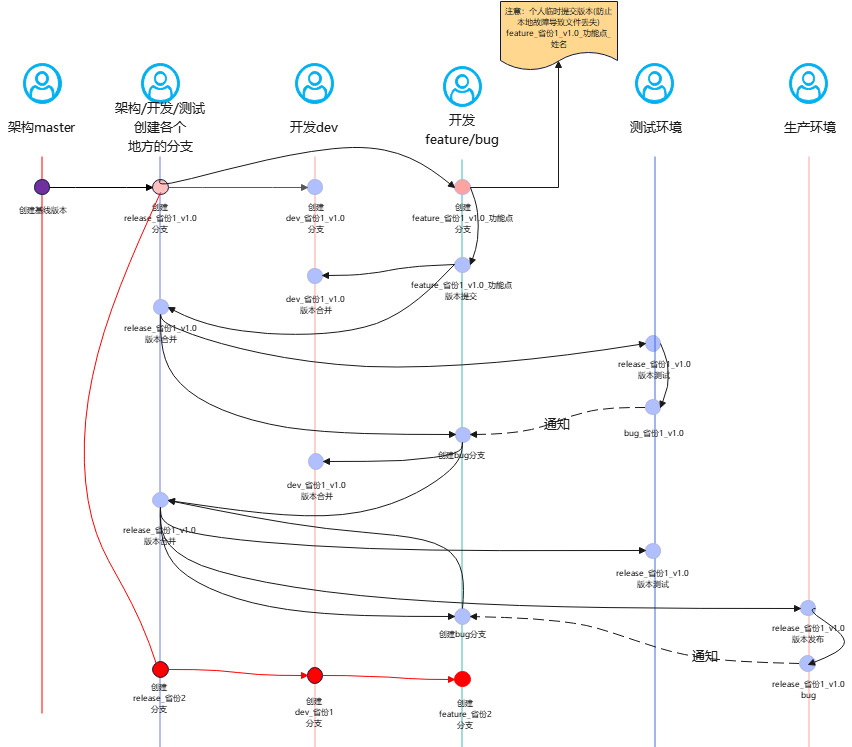
# 操作规范

## 操作规范流程说明

单版本迭代：



多版本并行迭代：

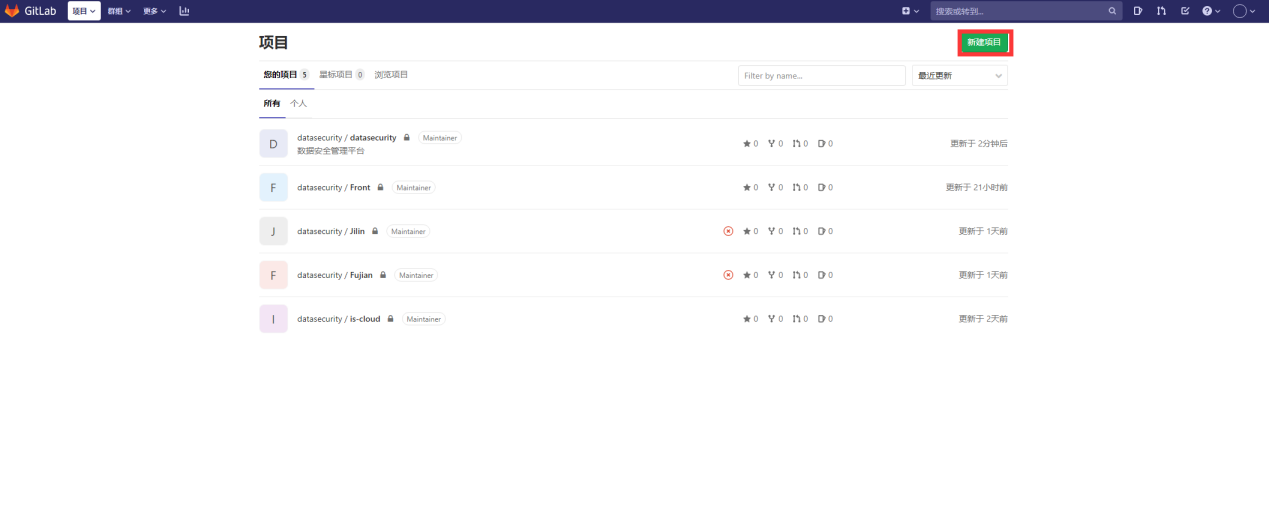


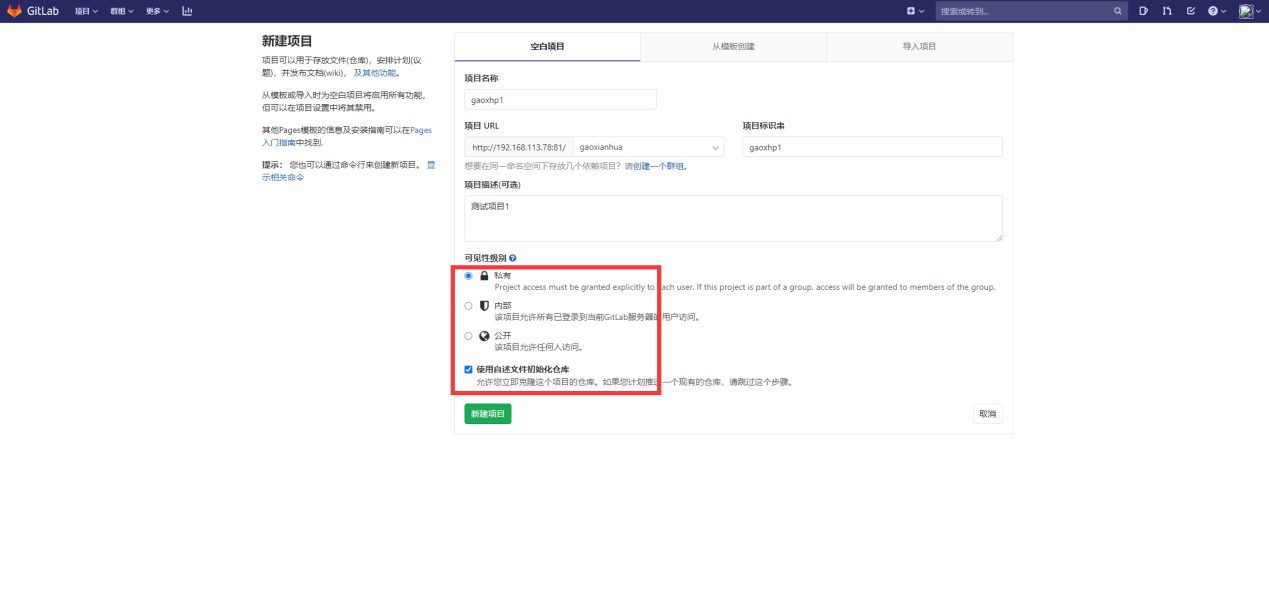
## 操作规范说明

### 架构

#### 创建初始版本

1. 创建工程





1. 新建本地代码，并关连远程分枝

mkdir gaoxhp1

cd gaoxhp1

git init

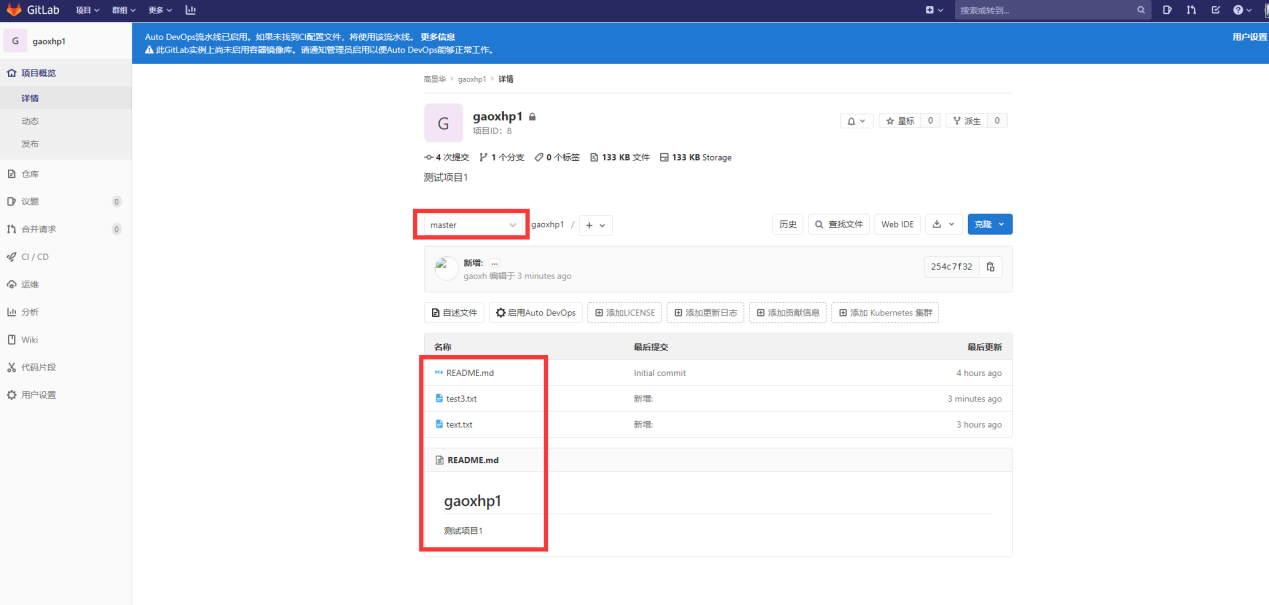
git remote add origin http://192.168.113.78:81/gaoxianhua/gaoxhp1.git

git pull origin master --allow-unrelated-histories

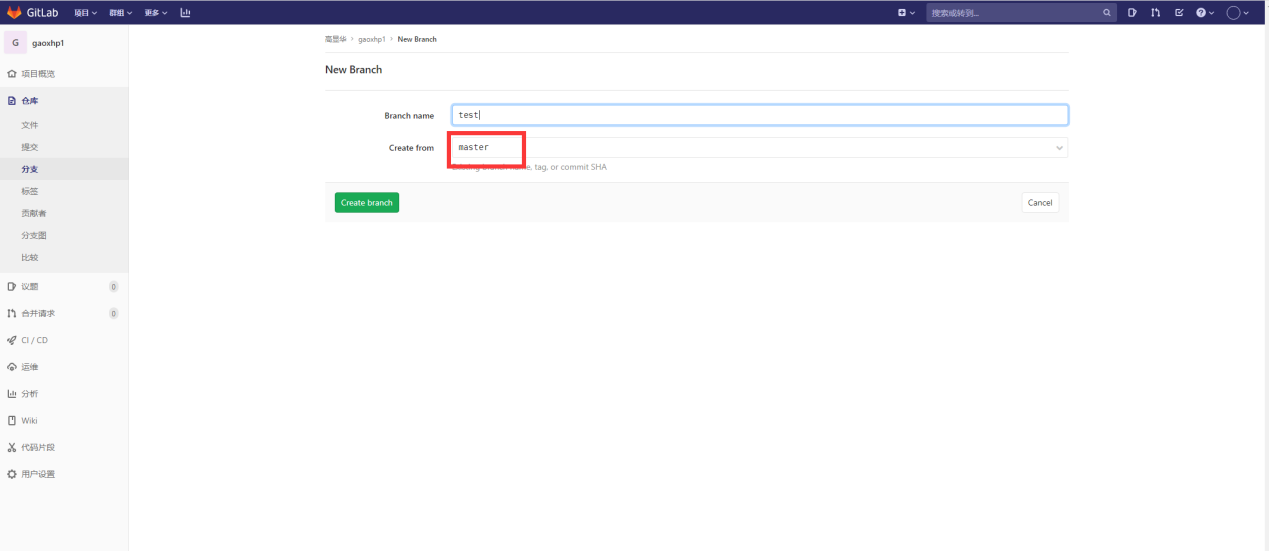
git add .

git commit -m”根据要求格式填”

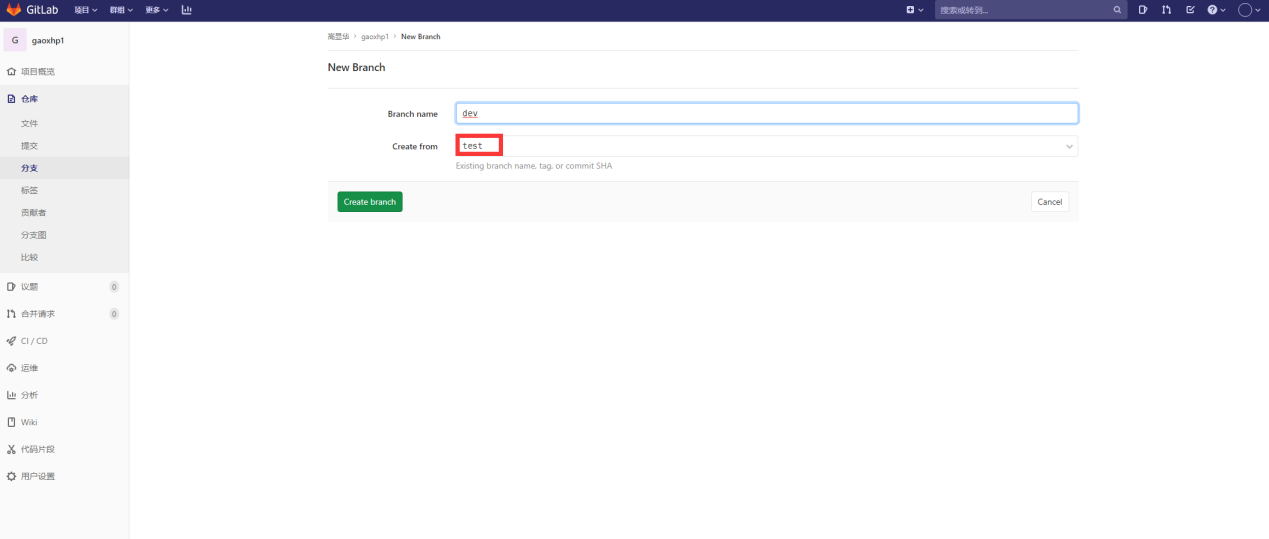
git push -u origin master



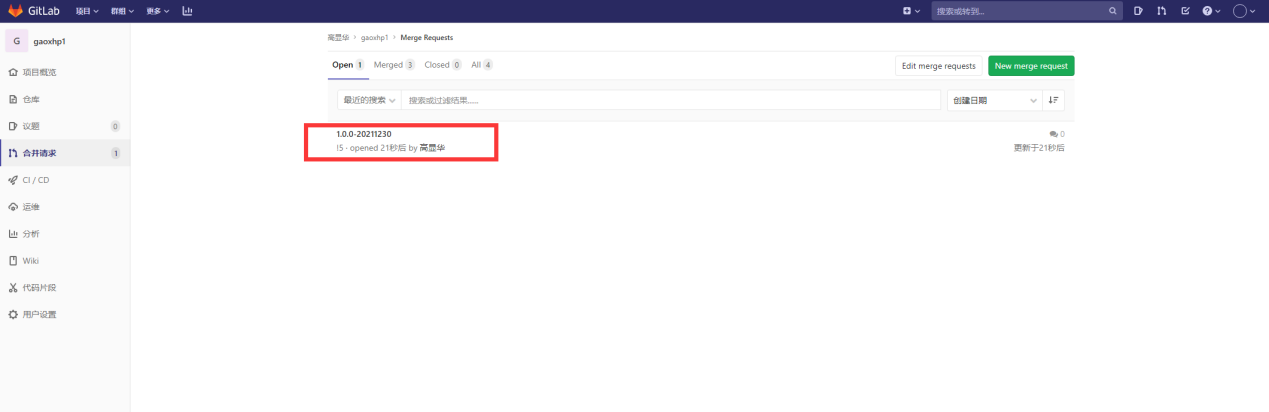
#### 创建分枝release

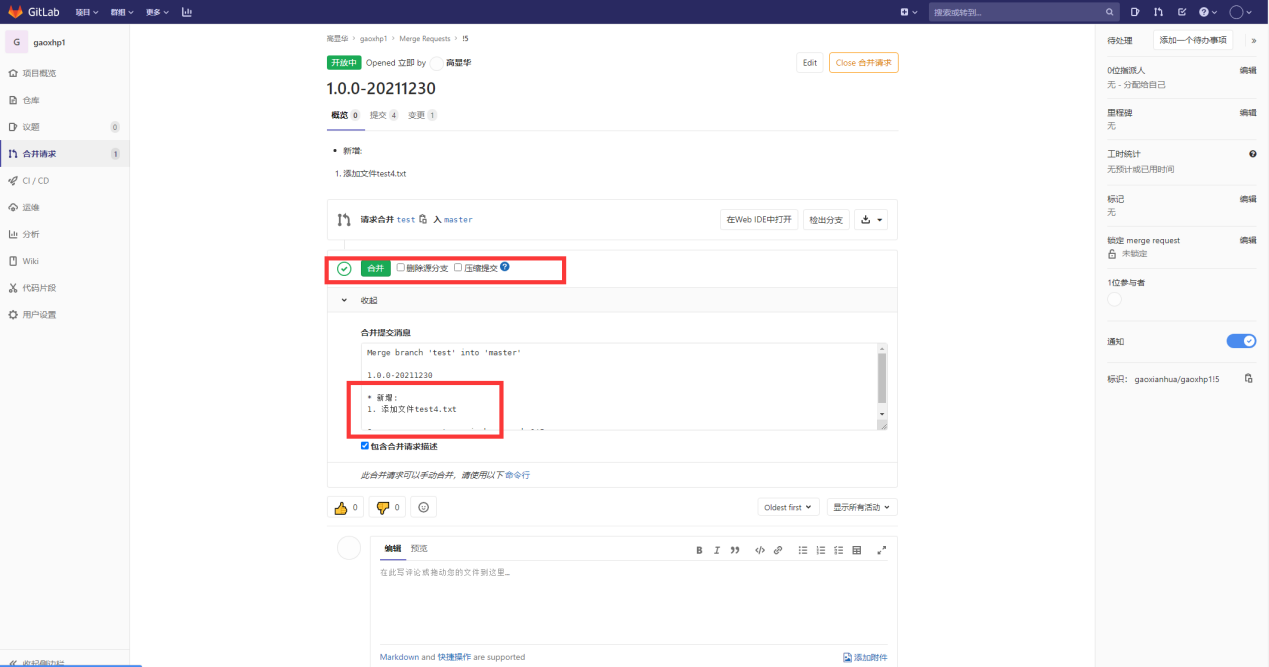


#### 创建dev分枝

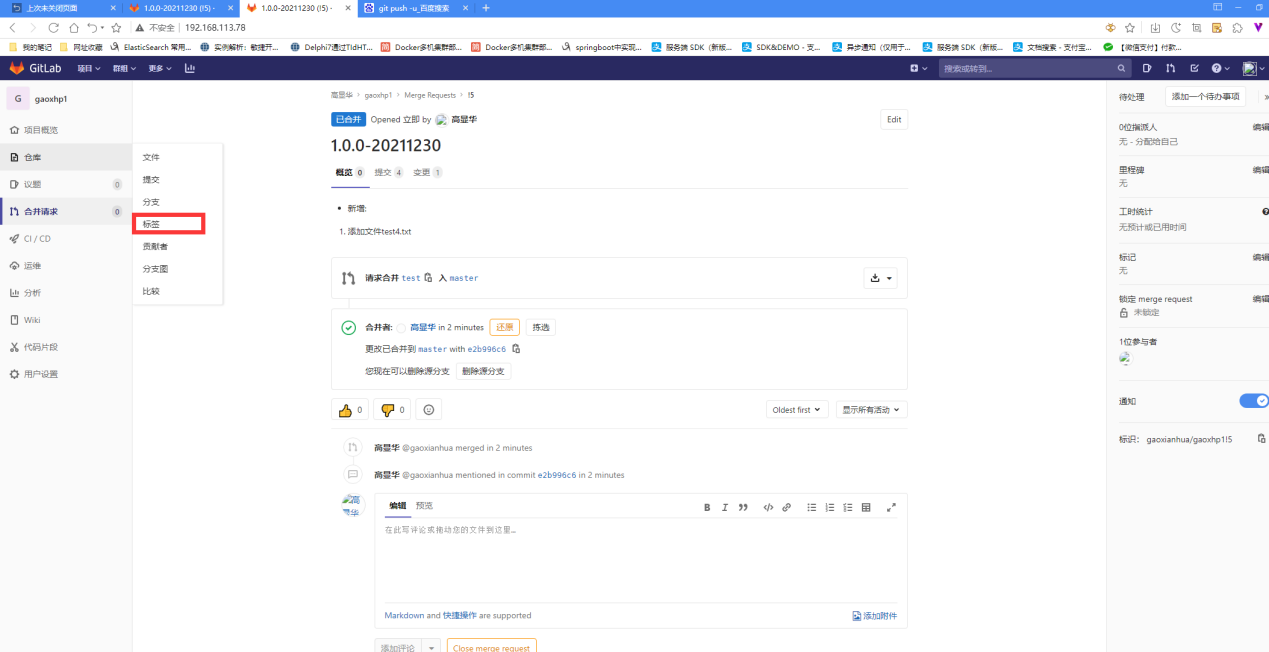


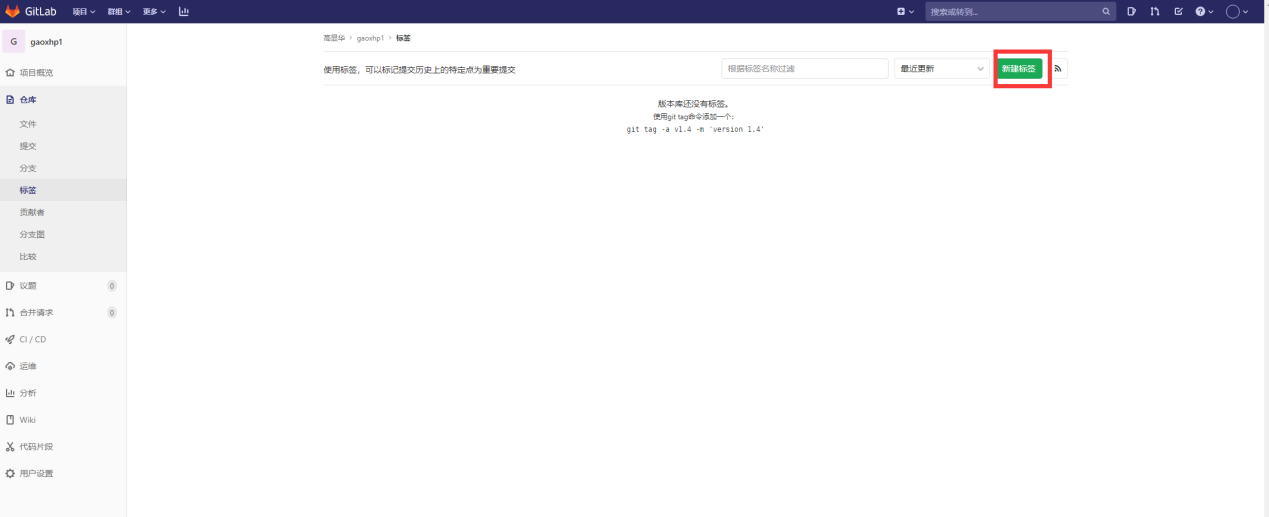
#### 合并master分枝请求

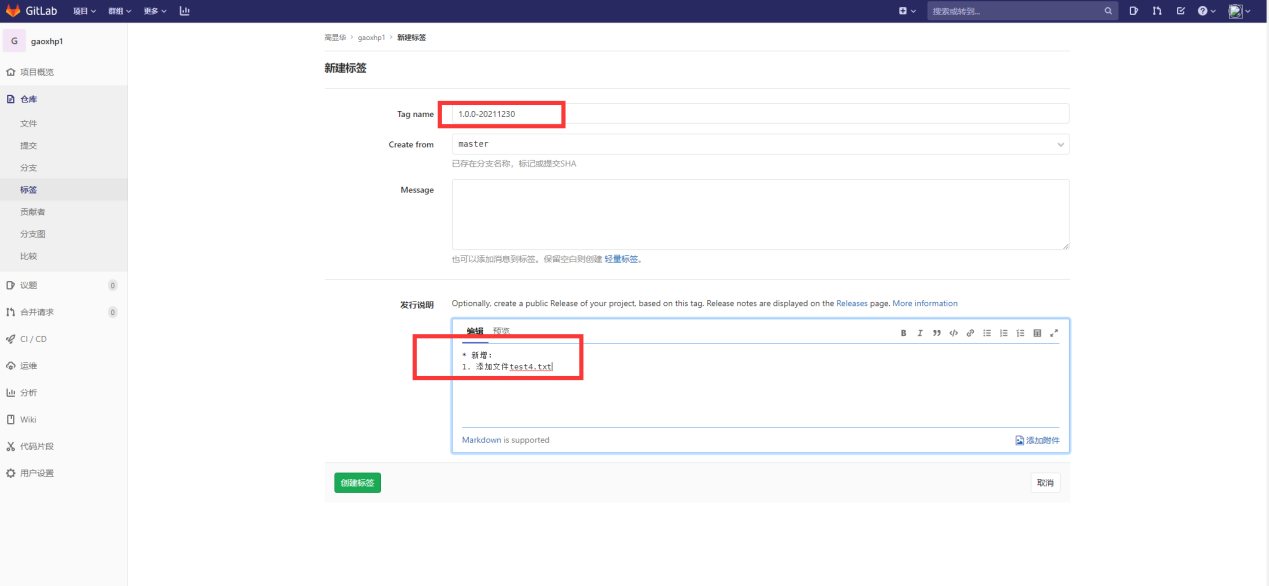


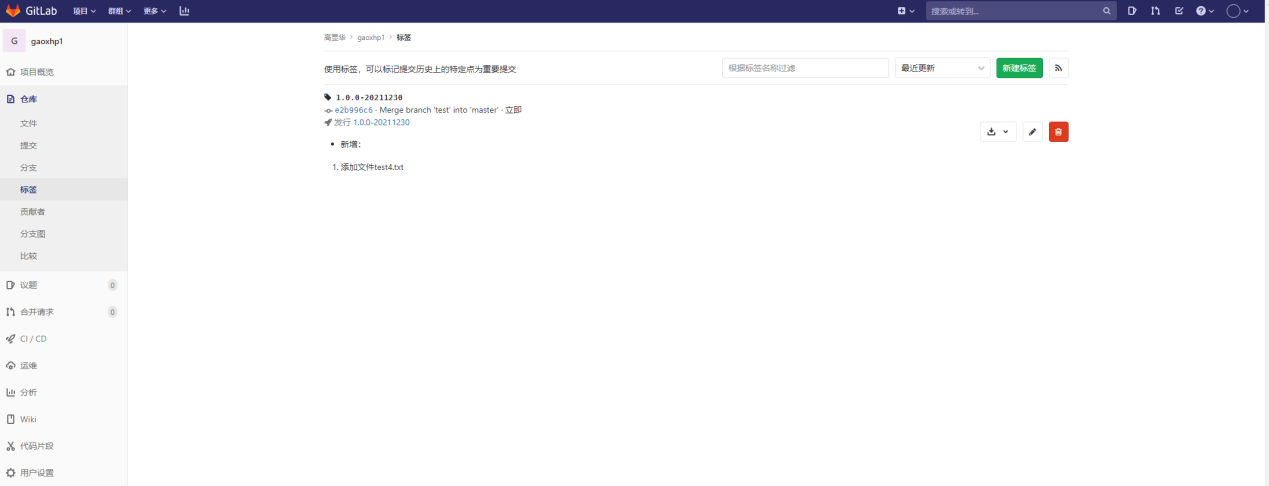


#### tag版本





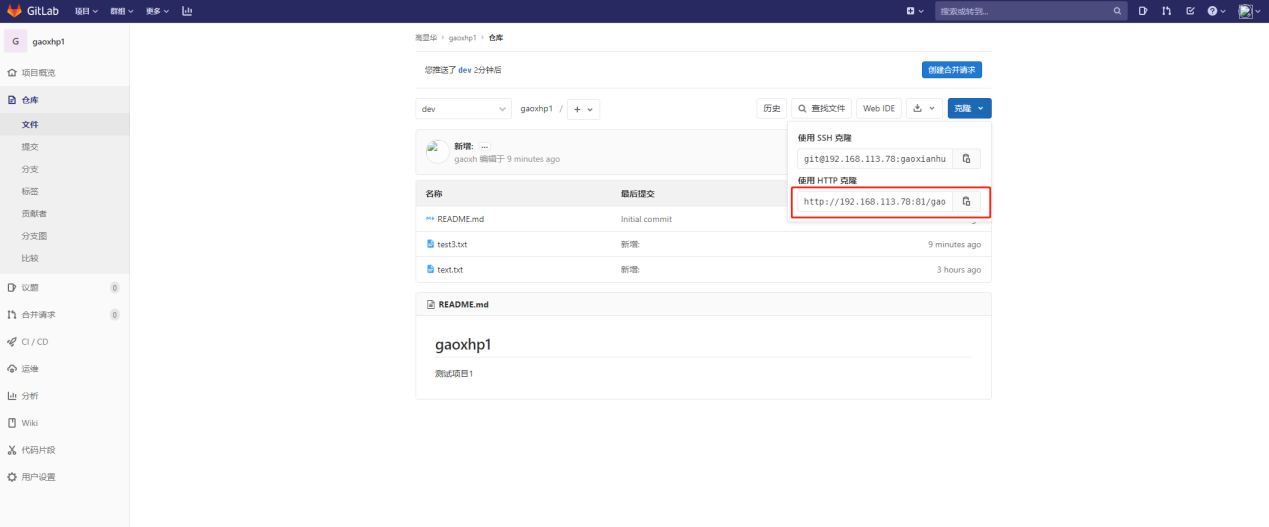




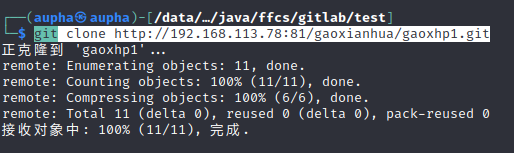
#### fork仓库

### 开发

#### 拉取代码



git clone <http://192.168.113.78:81/gaoxianhua/gaoxhp1.git>



git branch -a

#### 检出release分枝

git checkout release-xxx



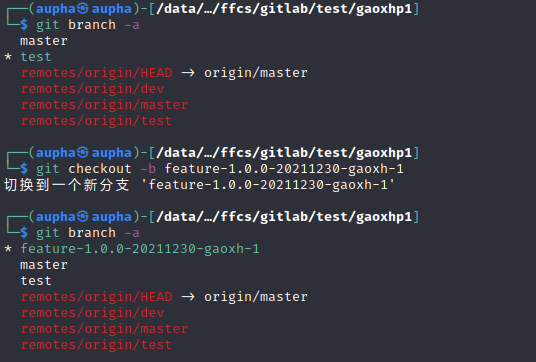
#### 根据release分枝检出功能个人分枝

个人分枝以功能点单位进行检出;

假设当前的版本计划为1.0.0-general-20211230;

检出分枝命令如下(当前为release分枝)：

git checkout -b feature-1.0.0-general- 20211230- gaoxh-1

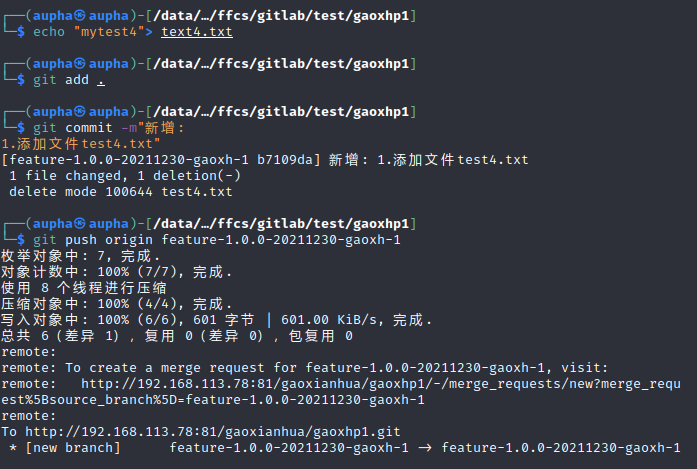


#### 提交个人分枝功能

git add .

git commit -m”新增:\r\n1.添加文件test4.txt”

git push origin feature-1.0.0-general-20211230-gaoxh-1

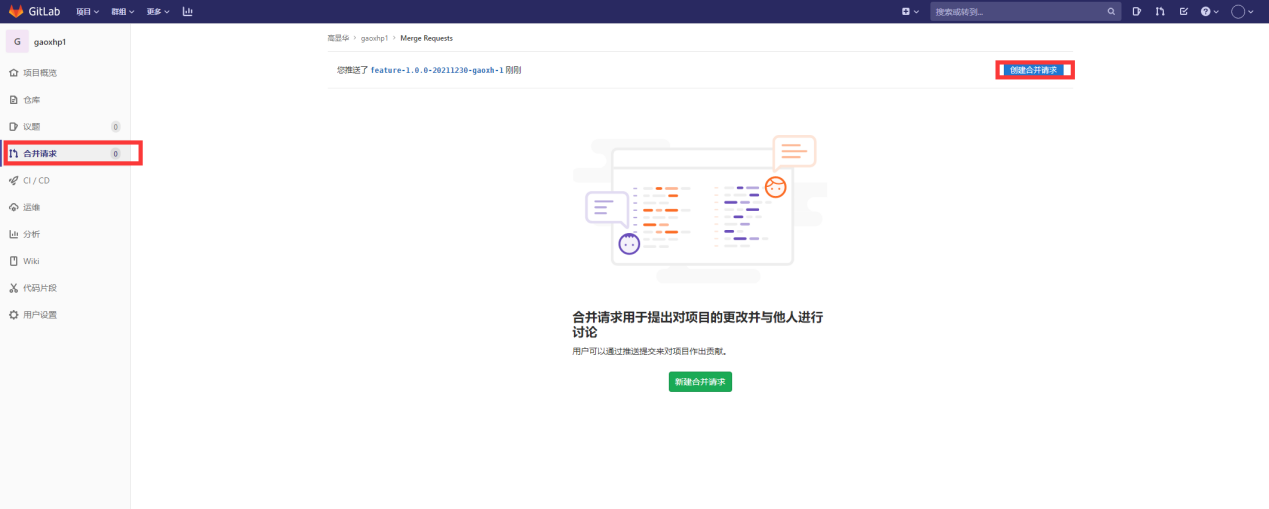


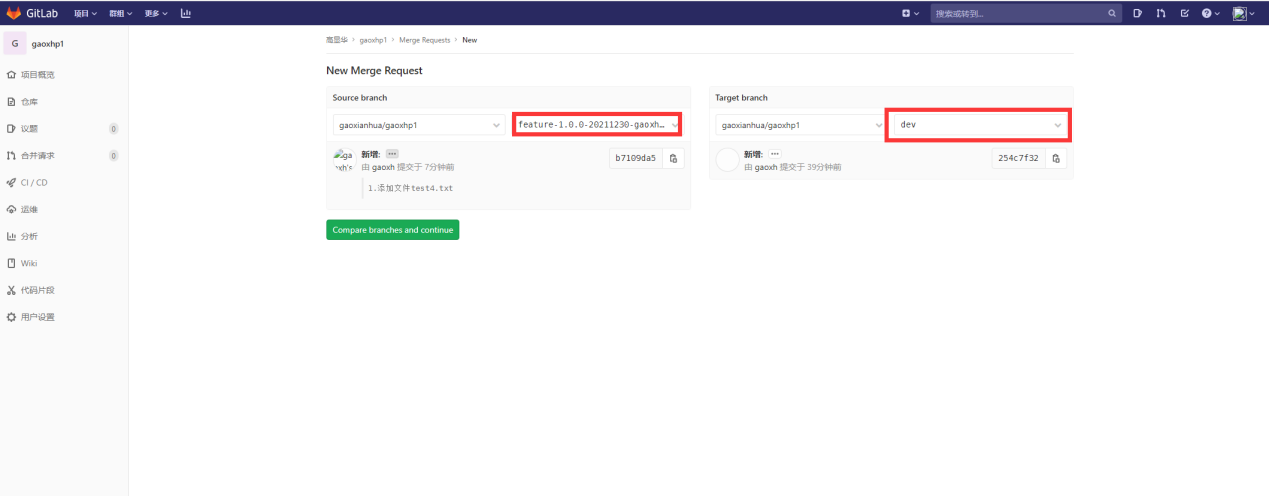
#### 合并个人分枝到开发分枝dev

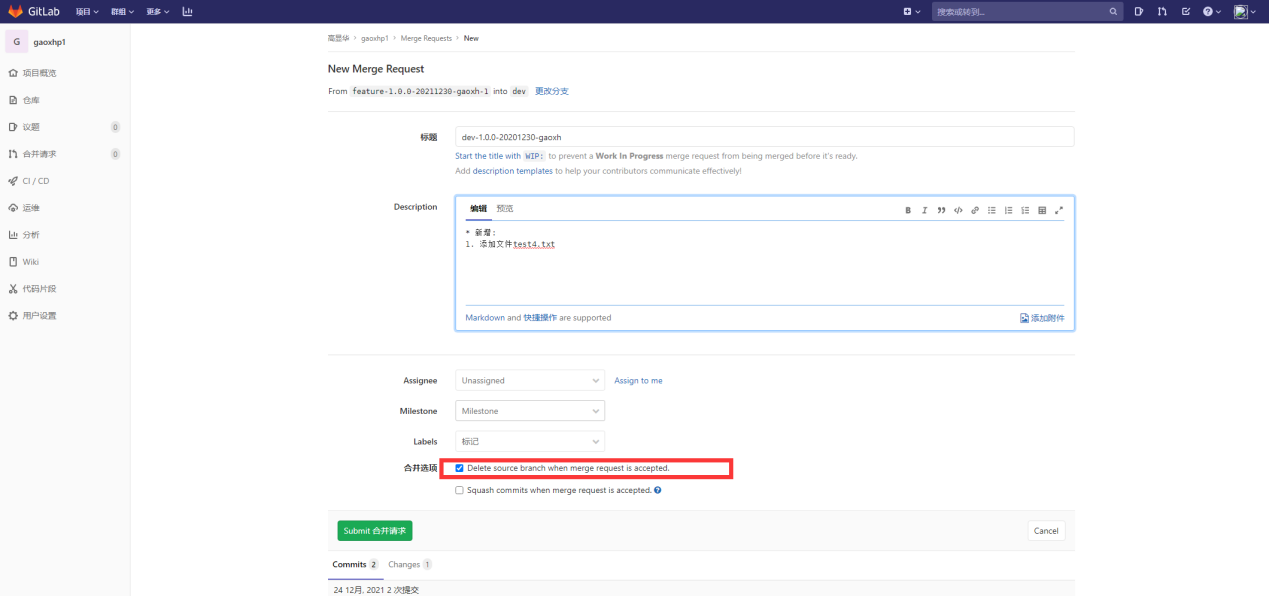
此操作包括合并请求和合并操作

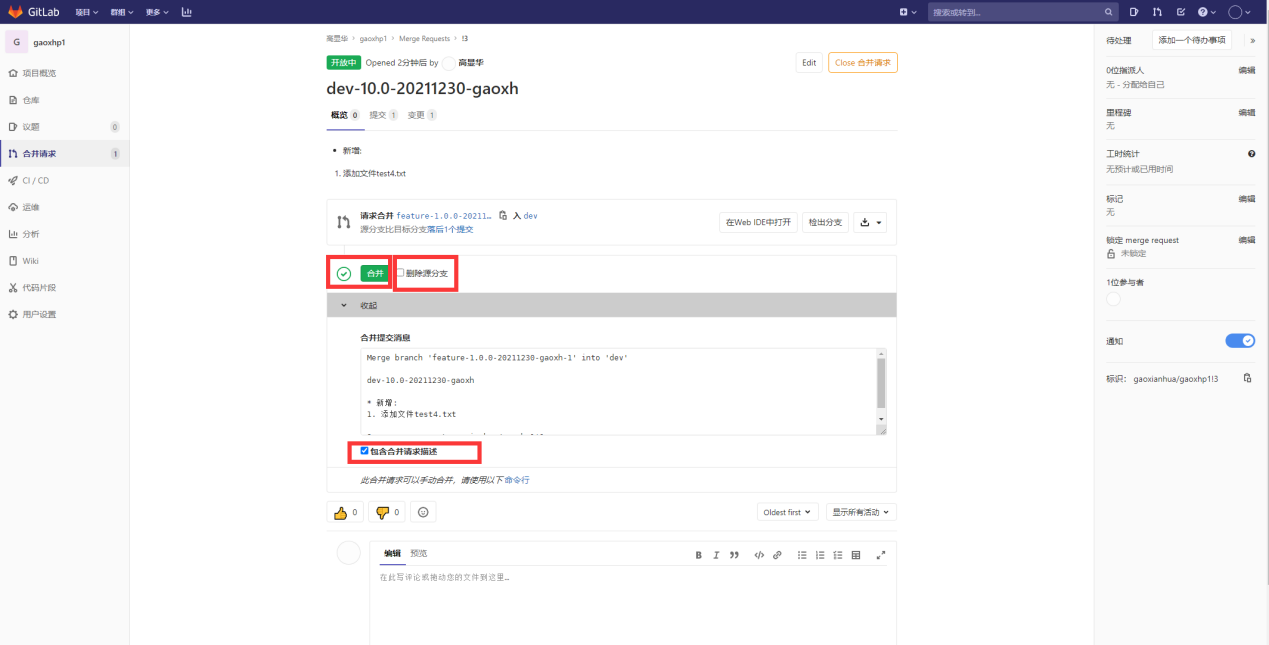
注意:

1. 当前合并请求需要在确认个人分枝功能在个人分枝上验证通过后;
2. 提交代码合并到dev上后，必须保证dev分枝代码是可运行状态；

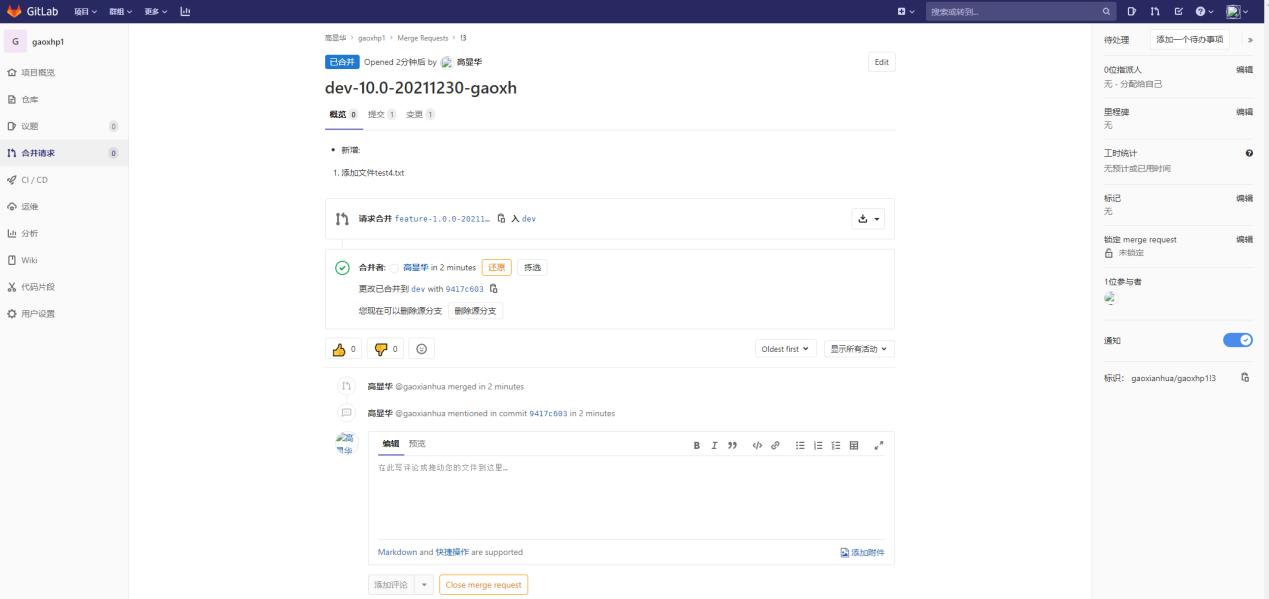






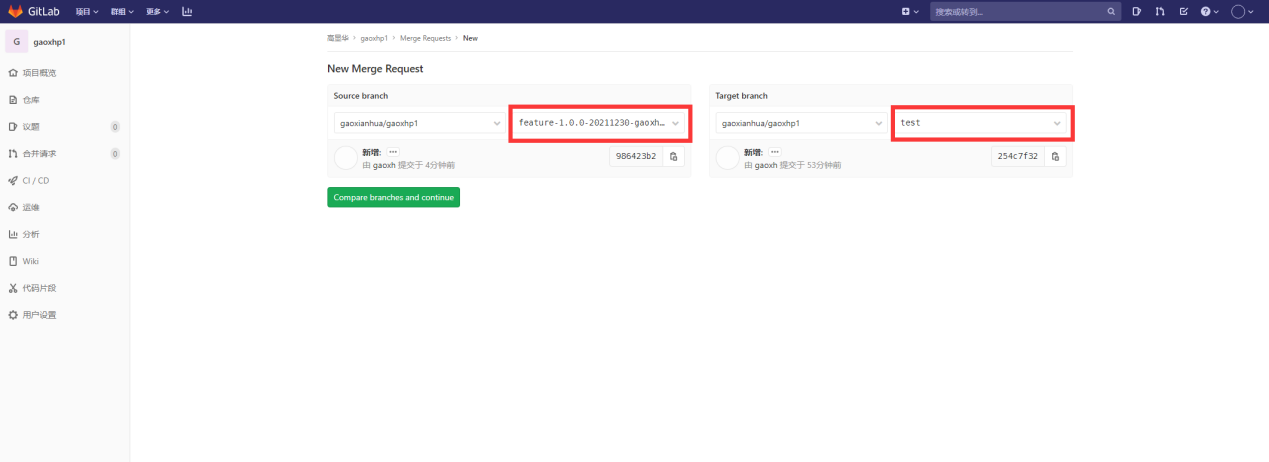


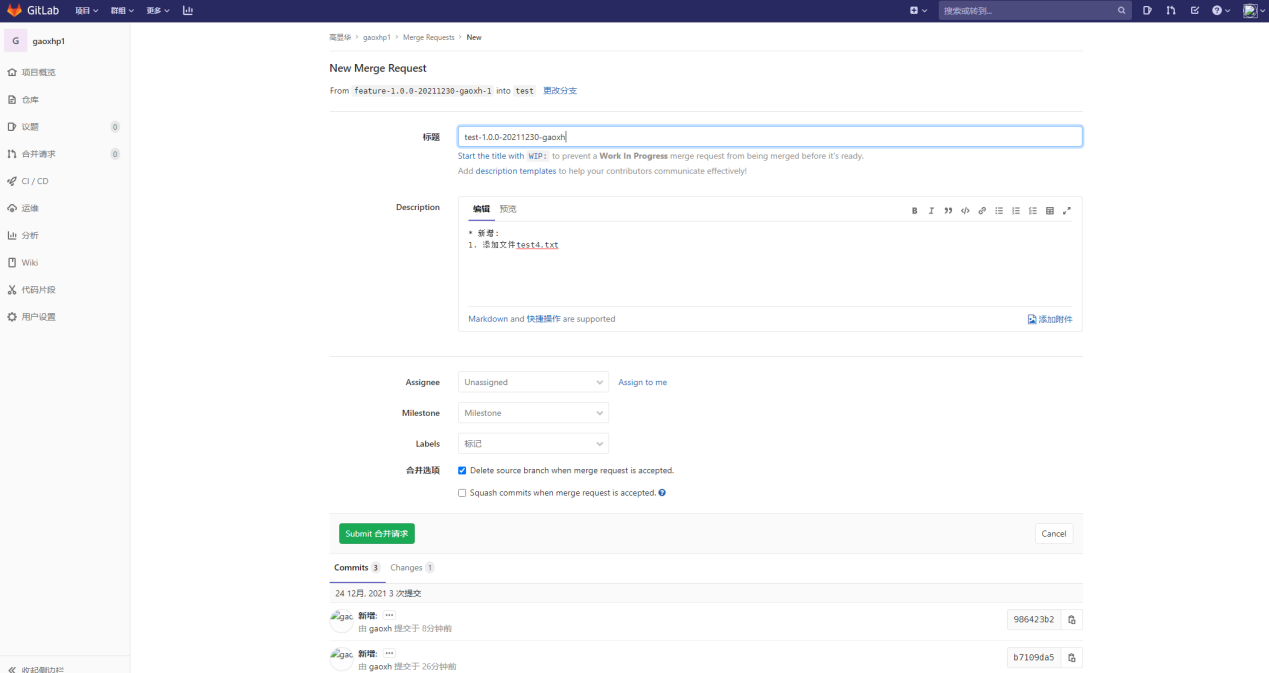
建议上图的勾选先不要勾选，在版本上线到测试环境后再手动删除个人分枝



#### 提交测试分枝release合并请求(MR)

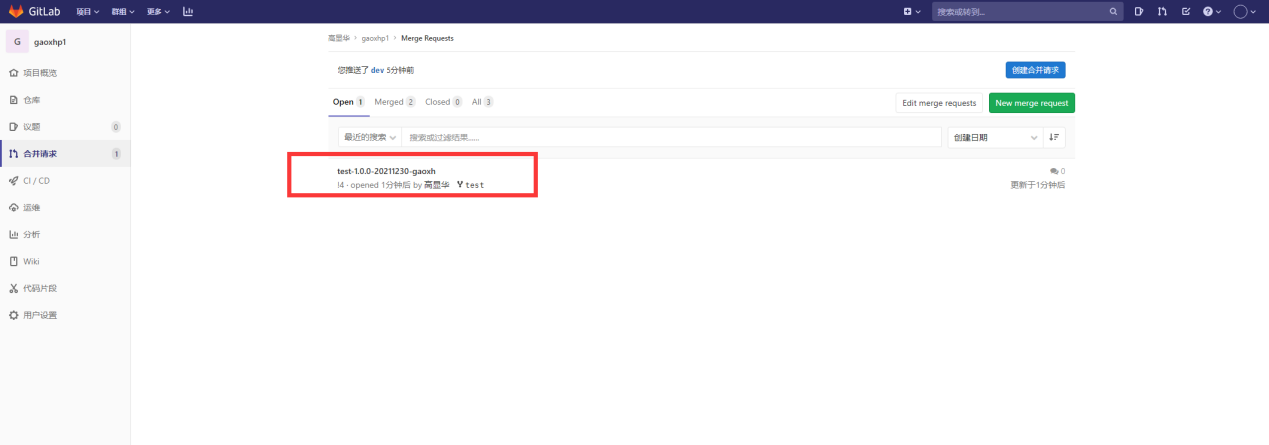
这个操作是衔接开发与测试的流程

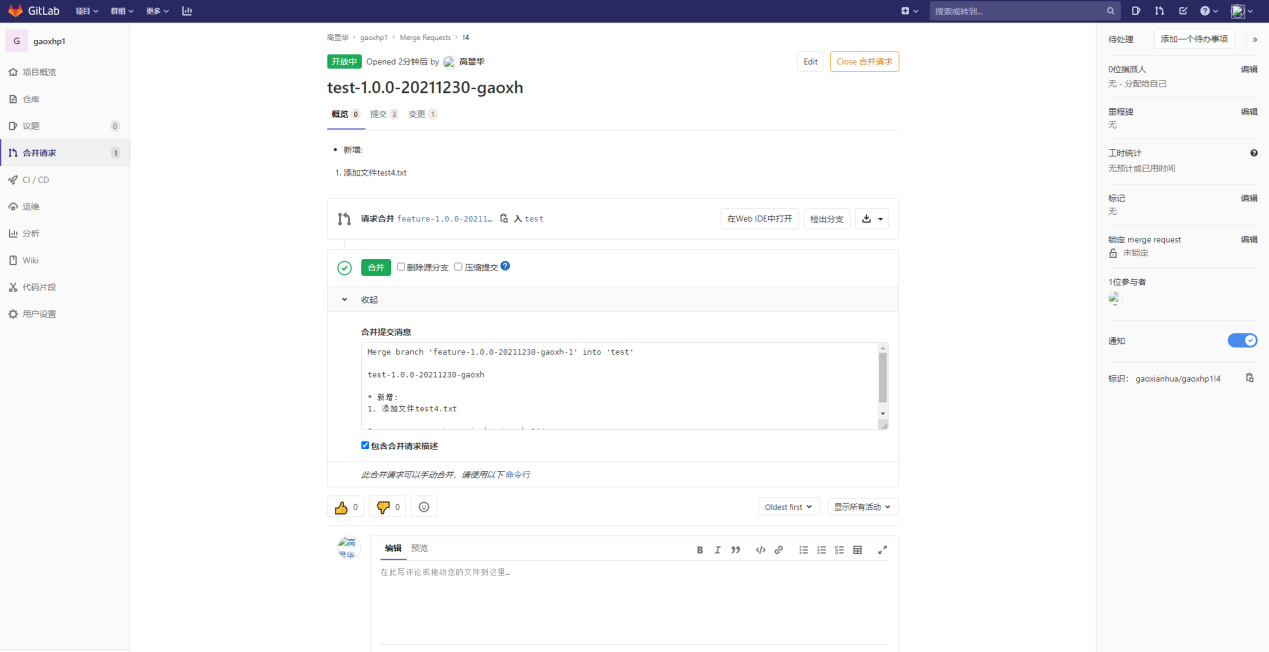


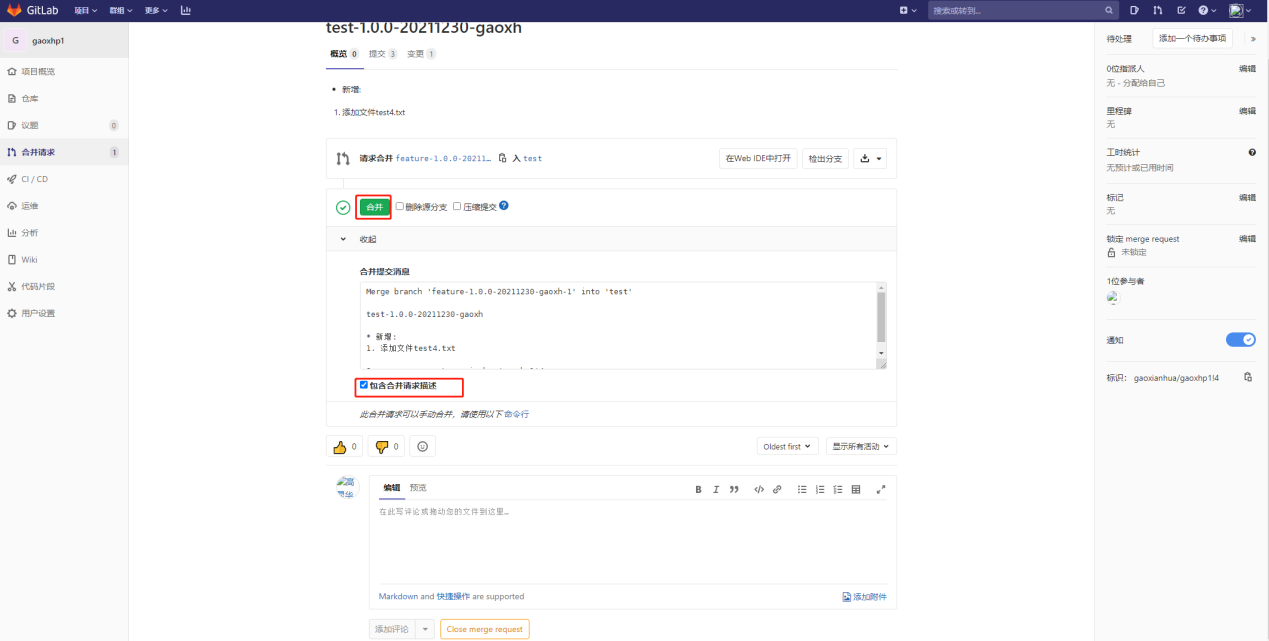


### 测试

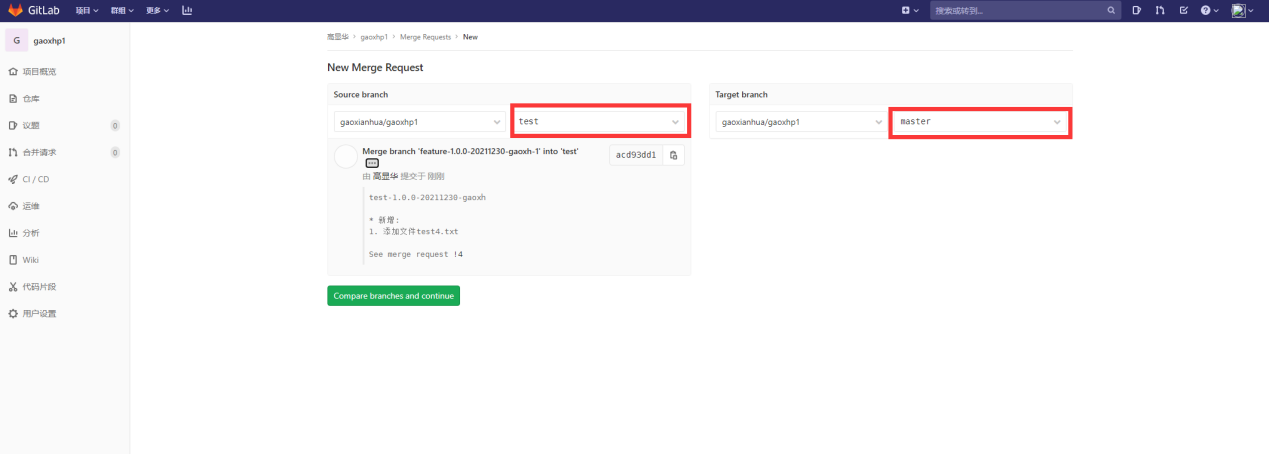
#### 审核版本并合并请求(MR)

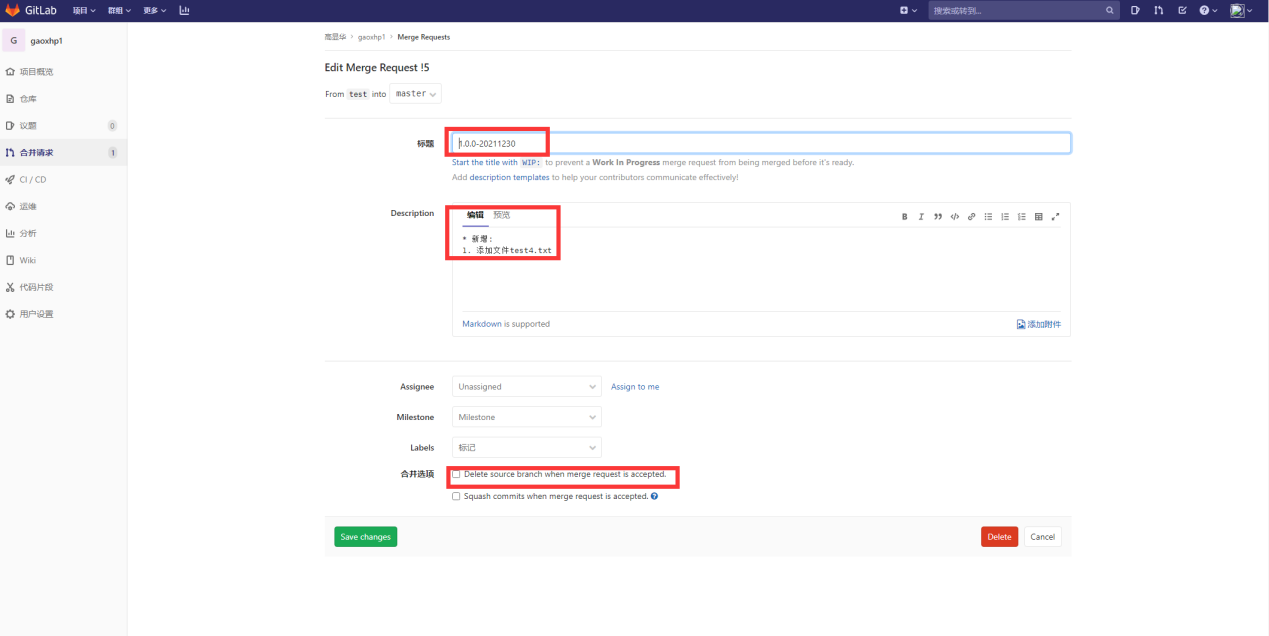






#### 封版后提交master分枝合并请求(MR)





## 命名规范

### 提交内容规范

提交时根据功能性质填写规范为如下:

|  |
| --- |
| 新增:   1. 新功能XXX   修改：   1. 修改功能XXX(需求变更)   bug:   1. 修复XXX   hotfix:  1.解决生产bugxxx |

### 合并请求规范

1. 标题以目标分枝名开头后带上版本和个人信息，如下:

dev-1.0.0-general-20211230-gaoxh

1. 内容参照提交内容规范（3.3.1）

### tag标签规范

1. 标题根据版本计划命名:1.0.0-general-20211230
2. 内容参照提交内容规范（3.3.1）

# 其他

## 合并代码冲突后如何解决

1. 在本地通过dev分支新建一个dev\_x.x.x\_merege\_姓名分支。
2. 将冲突的feture分支合并到dev\_x.x.x\_merege\_姓名分支，解决完冲突后合并到dev分支