

# Gitlab使用方法

余果 2017. 3. 20

# Gitlab

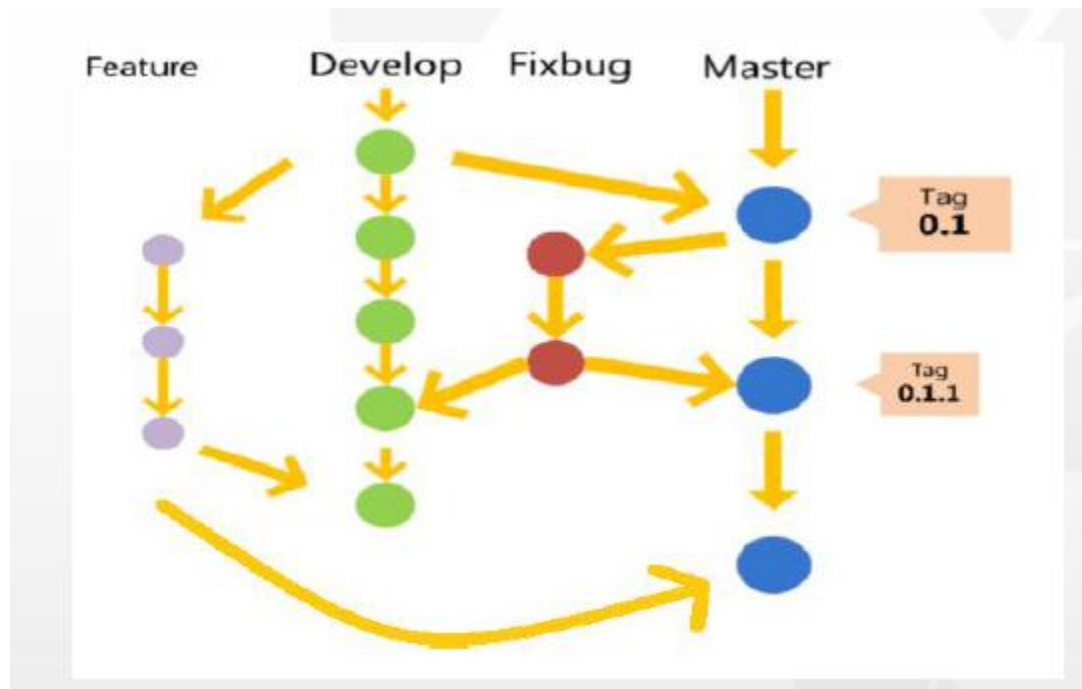
## The platform for modern developers

GitLab unifies issues, code review, CI and CD into a single UI



分支管理、团队协作、bug追踪、需求管理、代码讨论、代码审核、文档共享

# Git flow 分支管理



# Gitlab flow 分支管理

Gitlab flow 的最大原则叫做"上游优先"（upsteam first），即只存在一个主分支 **master**，它是所有其他分支的"上游"。只有上游分支采纳的代码变化，才能应用到其他分支。

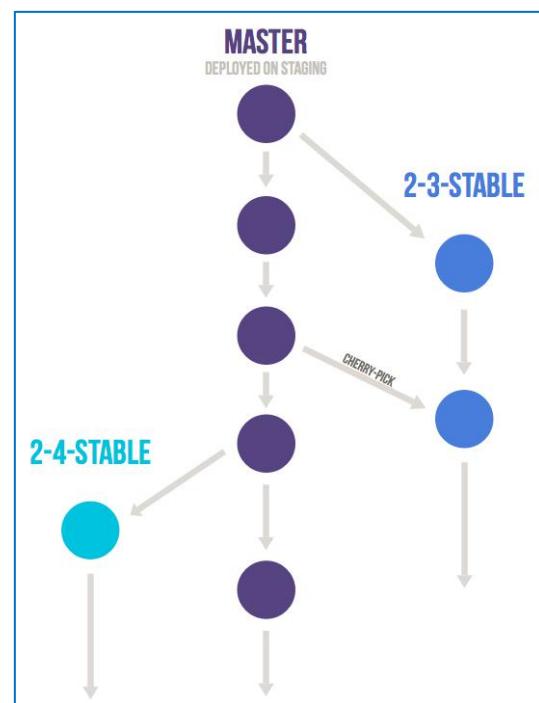
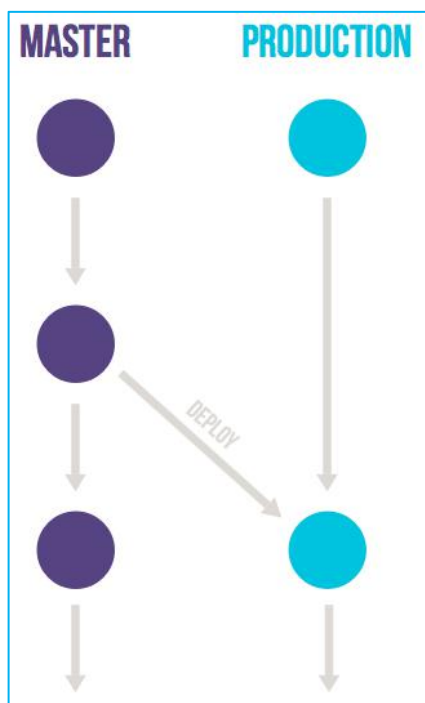
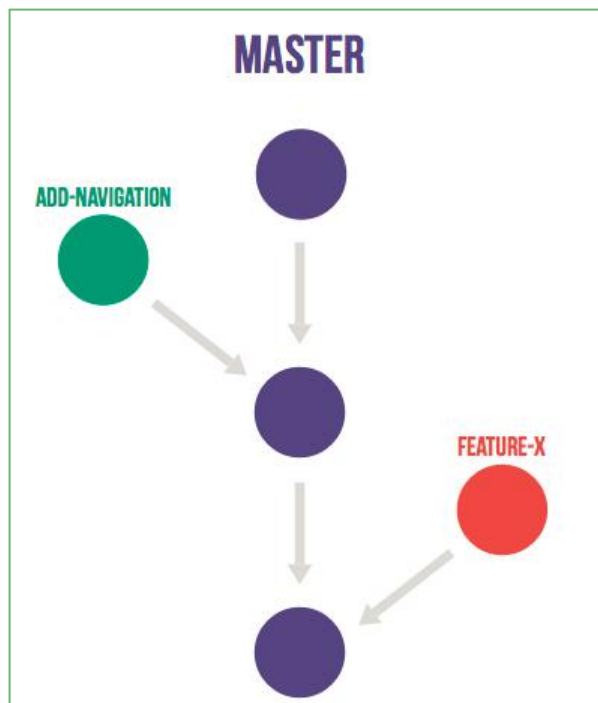
开发环境



线上生产



版本发布



# Code Review 代码讨论

- ◆ 假如你在一个feature分支工作比较长时间，最好像其他人实时的分享你的工作内容，进行merge request而不指向任何人。这意味着你的代码并没有准备好做merge request，但是欢迎大家来提意见。你的团队成员可以评论代理提出意见。

对已有的提交comiits小伙伴们可以经常学习交流下：

- 看不懂的提个疑问、
- 写的精彩的地方赞扬下、
- 写错的帮忙指出来、
- 有更好写法的切磋下

Showing 1 changed file with 9 additions and 10 deletions



```
src/mbio/files/meta/otu/otu_table.py

...      ...      @@ -232,15 +232,7 @@ class OtuTableFile(File):
232      232      }
233      233      tax = re.sub(r'\s', '', tax)
234      234      cla = re.split(';', tax)
235      -
236      -      def get_last_tax(cla, last):
237      -      """
238      -      无 d__ 避免无限无限循环出错
239      -      """
240      -      if re.search(r'^[kpcofgs]__(norank|unclassified|uncultured|incertae_se
241      -      return get_last_tax(cla, last - 1)
242      -      return cla[last]
243      -      last_tax = get_last_tax(cla, len(cla) - 1)
```



## Merge request 提交合并 代码检查

当你觉得合适了，需要合并到主要受保护分支，就把merge request指定给熟悉你的正在做的事的人，提醒他来查看代码给你反馈，负责review的小伙伴可以对代码进行评论，在accept之前，该分支中再次push的commit都归属于这次merge request。accept之后，分支自动合并到目标分支中

New Merge Request

From toolapps into master

Change branches

Title

Toolapps

Start the title with `WIP:` to prevent a **Work In Progress** merge request from being merged before it's ready.

Add description templates to help your contributors communicate effectively!

Description

WritePreview

B I " <> ☰ ☷ ☑ ✕

Write a comment or drag your files here...

Styling with Markdown and slash commands are supported

📎 Attach a file

Assignee

Assignee

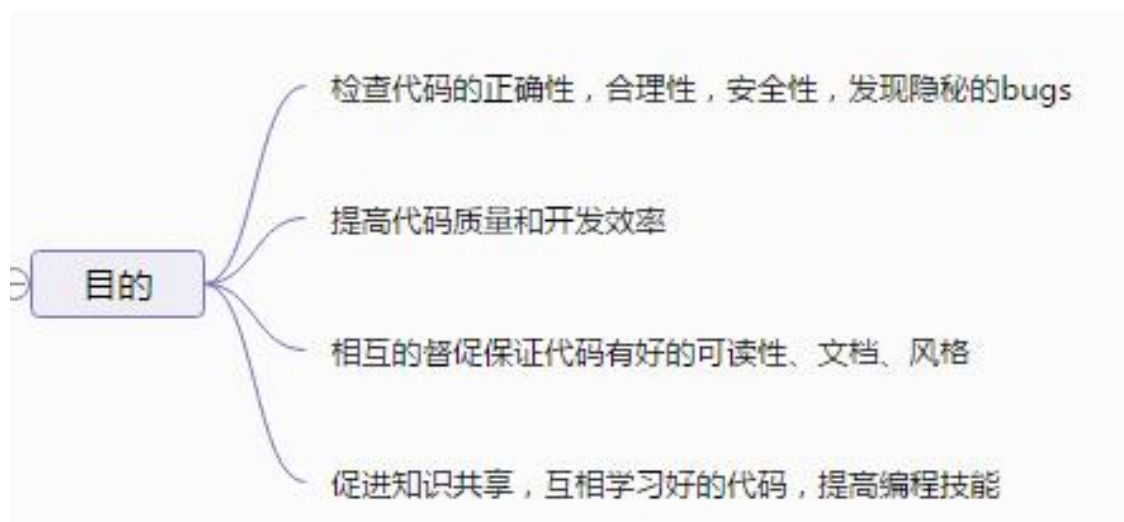
Showing 2 changed files with 2 additions and 1 deletions

InlineSide-by-side

▼ 📄 scripts/dataexchange/upload.py 🗑

... @@ -6,6 +6,7 @@ import os  
6 from upload\_task import UploadTask  
7  
8  
9 parser = argparse.ArgumentParser(description="根据验证码，上传一个文件夹至某一个项目下")  
10 parser.add\_argument("-c", "--identity\_code", help="验证码", required=True)  
11 parser.add\_argument("-l", "--file\_list", help="文件列表，用于描述文件的信息，以及是否允许让客户直接下载", required=True)  
...  
... @@ -6,6 +6,7 @@ import os  
6 from upload\_task import UploadTask  
7  
8  
9 +  
10 parser = argparse.ArgumentParser(description="根据验证码，上传一个文件夹至某一个项目下")  
11 parser.add\_argument("-c", "--identity\_code", help="验证码", required=True)  
12 parser.add\_argument("-l", "--file\_list", help="文件列表，用于描述文件的信息，以及是否允许让客户直接下载", required=True)  
...

# Code Review 代码审查





# Code Review 代码审查

review方法

- 每次 review 包含 200 行左右代码是比较理想的，最多不要超过 400 行
- 自己先看一遍，每次提交前整体把自己的代码过一遍，尤其是看看有没有犯低级错误。
- 把简单的检查自动化，有很多检查是可以自动化的，比如一些风格规范（缩进、空行、行尾空格、命名等）
- 该写清楚作者希望 reviewer 重点关注的问题
- Code review 应该全员平等参与，每个人都会有所收获
- 关注设计方面的问题和客观的规范，避免在主观意见上争执
- 多问问题。多问“这块儿是怎么工作的？”“如果有XXX case，你这个怎么处理？”
- 开放的心态，虚心接受大家的Review Comments
- 花足够的时间进行适当缓慢的评审，但是不要超过 60-90 分钟



# Code Review 代码审查

## 代码整洁

代码大部分时候是用来维护的，而不是用来实现功能的

恰到好处的注释。但更多我看到比较差质量的工程的一个特点是缺少注释。

优秀的代码大部分是可以自描述的，好于文档和注释

单一职责原则：要定义的东西，如果不能用一句话描述清楚职责，就把它拆掉。

行为是否统一：同一逻辑/同一行为 有没有走同一Code Path，比如缓存是否统一，错误处理是否统一，错误提示是否统一

清晰的命名，思考一个方法命名的时间，比写一段代码的时间还长

避免过长参数、避免过长方法和类

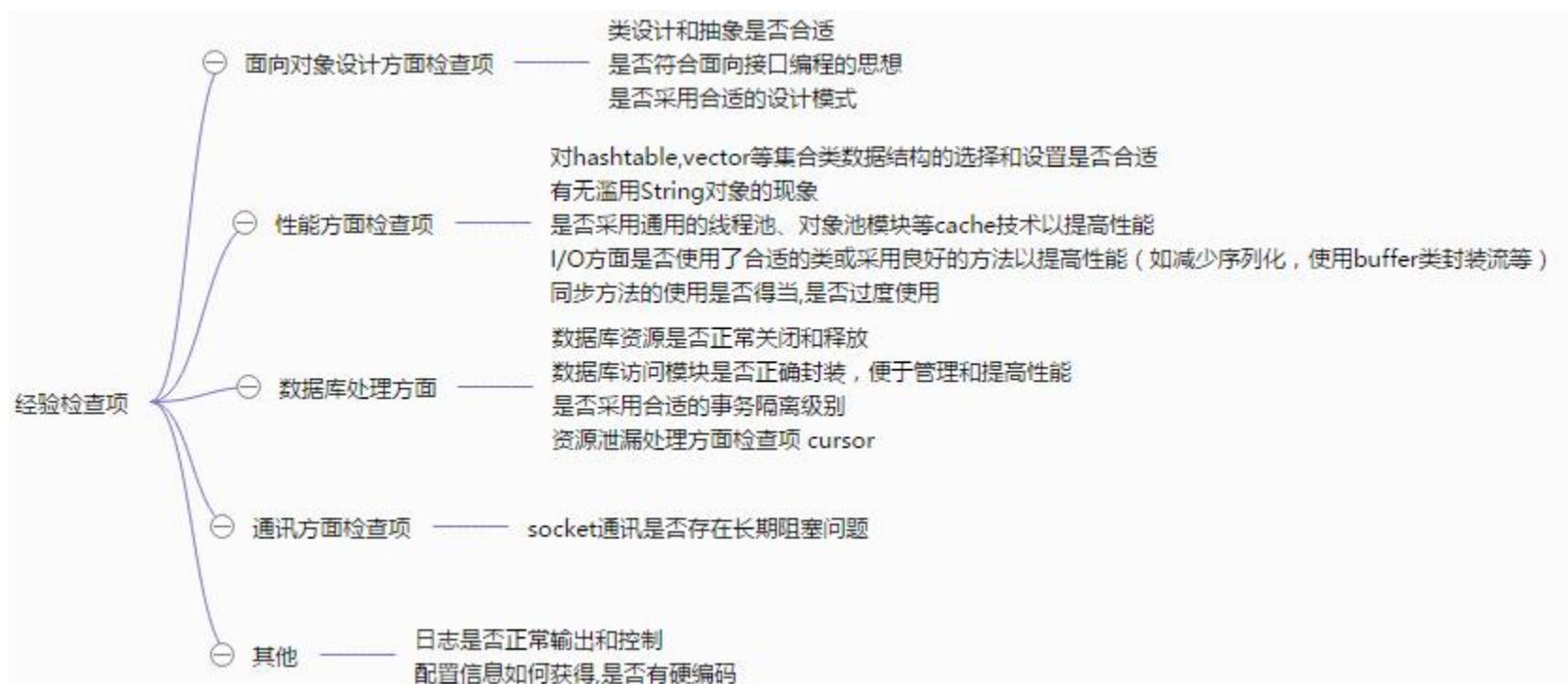
重复代码：主要看有没有把公用组件，可复用的代码，函数抽取出来

错误处理：有没有很好的Error Handling？比如网络出错，IO出错。

# Code Review 代码审查



# Code Review 代码审查



## Issue 需求问题追踪

代码的任何修改都应该开始于一个目标明确的issue，根据每个重要的功能点或者问题来创建issue。

The Issue Tracker is the place to add things that need to be improved or solved in a project

Issues can be bugs, tasks or ideas to be discussed. Also, issues are searchable and filterable.

New issue

当你准备写代码时，首先为issue创建一个新分支，这个分支的名字应该以issue number开始，例如“15-require-a-password-to-change-it”

# Issue 需求问题追踪

1. 填写详细描述的时候可以@某个人，或者@某个组得成员
2. 填写详细描述的时候 可以用 #issues编号引用某个issue的问题

New Issue

Title

Add description templates to help your contributors communicate effectively!

Description

Write Preview

B I " " </> [ ] [ ] [ ] [ ]

Write a comment or drag your files here...

Styling with Markdown and slash commands are supported

Attach a file

☐ This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee

Assignee

Due date

Select due date

Milestone

Milestone

Labels

Labels

Submit issue

Cancel

## Issue 需求问题追踪

- ◆ 从commit的message中或者是merge request的信息中可以关联相关的issue，具体语法是：**fixes #14, closes #67**。在gitlab中，这样的操作会在issue中创建一个评论，并且merge request会显示相关联的issue。假如这个merge request被接受，这个issue会自动被关闭。
- ◆ 假如你仅仅想关联这个问题，但是不关闭这个问题，你可以这样写：  
**Ducktyping is preferred. #12**
- ◆ 假如你有一个issue关联着好几处的修改，最好的办法是为每一个修改都创建一个issue，最后把这些issue关联到一个issue。



## Wiki 文档共享

存放项目的文档，使用指南等跟项目相关得信息，当然还可以存放  
**Issues**讨论中得精华部分

Project Activity Repository Pipelines Graphs Issues 0 Merge Requests 0 Wiki Snippets

Wiki was successfully updated.

Home

Last edited by 余果 5 minutes ago

New Page Page History Edit

Sanger Biocluster

Welcome to a new world !



# GitLab CI持续集成 自动代码检查和运行测试

## Pipeline

一次 Pipeline 其实相当于一次构建任务，里面可以包含多个流程，如安装依赖、运行测试、编译、部署测试服务器、部署生产服务器等流程。

任何提交或者 Merge Request 的合并都可以触发 Pipeline，如下图所示：



# 感谢您的欣赏



**美吉生物**  
**Majorbio**

地址/Add: 上海市浦东新区国际医学园区康新公路3399号3号楼

电话/Tel: 021-51875086

服务热线: 400-660-1216

网址/Web: [www.majorbio.com](http://www.majorbio.com)

传真/Fax: 021-51875086-8002