

Git 代码管理规范（草稿）

2019-1-24

latest: [from gitlab](#)

Overview

1. 初始配置规范

2. 提交描述规范



3. 代码合并规范

4. Gitlab保护分支和Tag

5. 里程碑和 Issues

初始配置规范

身份配置


-  使用 halliburton 电子邮箱
-  使用 halliburton 用户名

例如：

```
git config --global user.email "Rongbao.WEI@halliburton.com"  
git config --global user.name "Rongbao WEI"
```

 **注意** 区分字母大小写

推送配置

 仅推送当前分支

```
git config --global push.default simple
```

编码配置（仅 Windows）

 配置换行符和 GUI 文本编码

```
# 关闭换行符自动替换，建议跨平台代码请使用LF换行符  
git config --global core.autocrlf false  
# 解决 git gui 命令界面乱码问题（可选）  
git config --global gui.encoding utf-8
```

Overview

1. 初始配置规范
- 2. 提交描述规范**
3. 代码合并规范
4. Gitlab保护分支和Tag
5. 里程碑和 Issues

提交描述规范

在提交代码时，请按要求填写 `Message` 信息。

```
git commit -m "Message"
```

- 英文描述

- 👍 feature: Add translate files for PCA solution

- 👎 feature: 为产变分析加入翻译文件

- 👎 bugfix: fixed 🐛 #13 🙈

- 👎 hotfix: Happy new year :-)

- 有意义的描述

- 👍 feature: Add translate files for PCA solution

- 👎 feature: update

添加前缀

前缀	说明	例子
feature:	功能提交	feature: add translate files for PCA solution
bugfix:	BUG修复	bugfix: fix #13 PCA's translate error
hotfix:	补丁修复	hotfix: fix #31 connect db-server timeout
release:	发布	release: pre-release v1.0.0
WIP:	工作进行中	WIP: feature: add chinese translate text in path/to/file

特殊记号

在提交描述中允许加入下面特殊几号

- 跳过持续集成：`[ci-skip]`
- 关联 Issue ID：`#31`
- 关联 MR ID：`!12`

⚠ 注意：在终端中输入 `!12` 时，需要写成 `\!12`

“ ci：指持续集成（ Continuous Integration ）
MR：指 Gitlab 中的合并请求（ Merge Request ） ”

特殊记号

- 下面的例子是利用关键字关闭 issue

```
bugfix: Awesome commit message
```

```
Fix #20, Fixes #21 and Closes group/otherproject#22.  
This commit is also related to #17 and fixes #18, #19  
and https://gitlab.example.com/group/otherproject/issues/23.
```

Overview

1. 初始配置规范
2. 提交描述规范
- 3. 代码合并规范**
4. Gitlab保护分支和Tag
5. 里程碑和 Issues

代码合并规范

采用 **Gitlab 的 Merge request 方法**，合并代码。

例子：为 demo 项目开发一个名为 foo 的新功能

1. 克隆远程代码到本机

```
git clone https://git.openearth.community/weirongbao/demo.git  
cd demo
```

2. 检出开发分支

```
git pull  
git checkout -b develop origin/develop
```

例子：由我为 demo 项目开发一个名为 foo 的新功能

3. 创建 foo 功能分支

```
# 假设当前分支是 develop, 将 develop 代码检出到 feature/foo  
git checkout -b feature/foo
```

4. 开发并提交代码

```
# 跟踪目录结构变化  
git add -A  
# 提交改变 1  
git commit -am "feature: init foo code"  
...  
# 提交改变 n  
git commit -am "feature: add bar function"  
...
```


例子：由我为 demo 项目开发一个名为 foo 的新功能

5. 推送 `feature/foo` 分支到远程仓库

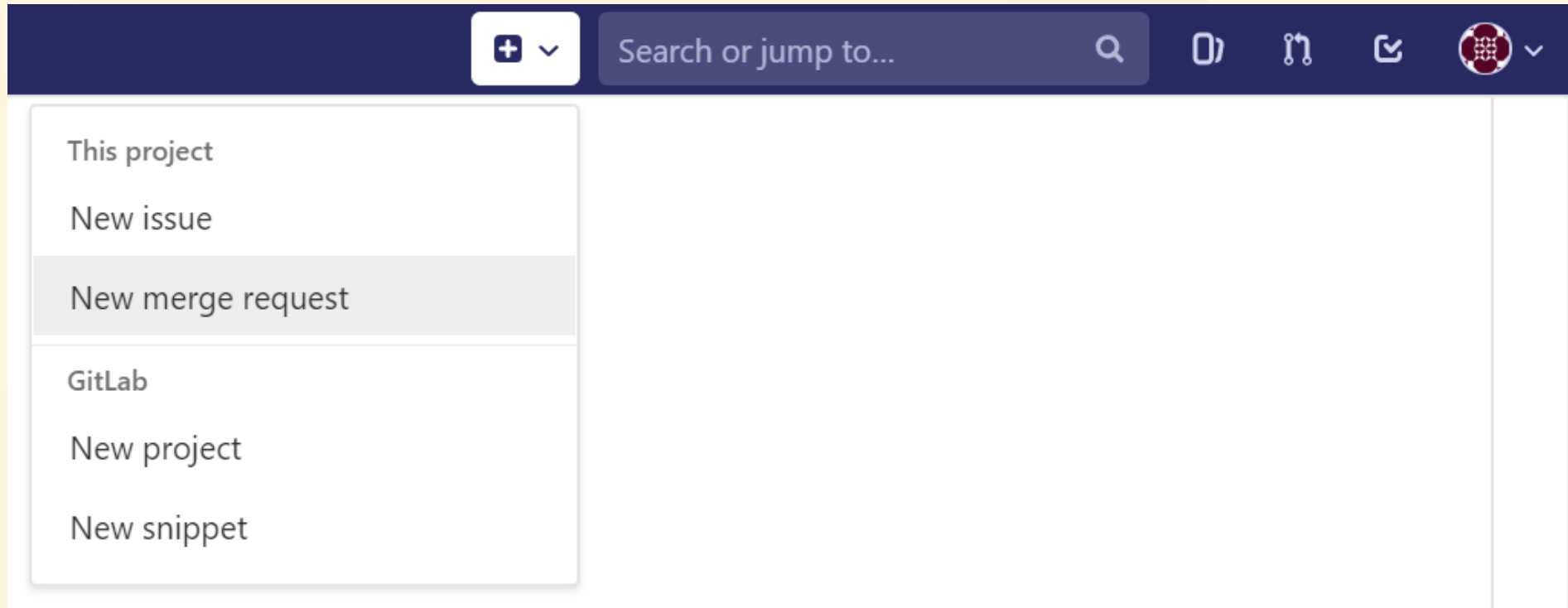
```
# 假设当前分支是 feature/foo，并且已经开发完成  
git push -u origin feature/foo
```

```
# 如果后来再有提交变更，直接在此分支用 git push 命令即可  
git push
```

6. 创建合并请求

用浏览器打开 demo 项目首页，点击右上方的  图标，选择 `New merge request` 菜单项。

例子：由我为 demo 项目开发一个名为 foo 的新功能



例子：由我为 demo 项目开发一个名为 foo 的新功能

左侧选择 `feature/foo` 分支，右侧选择 `develop` 分支


Rongbao WEI > demo > Merge Requests > New

New Merge Request

Source branch


weirongbao/demo

feature/foo

 feature: update README.md
Rongbao WEI authored 4 minutes ago

Verified


3933ea89




Target branch

weirongbao/demo

develop

 feature: init project
Rongbao WEI authored 11 minutes ago

8f77130a



Compare branches and continue

点击 `Compare branches and continue` 按钮

例子：由我为 demo 项目开发一个名为 foo 的新功能

输入适当的说明，Assignee 选择代码审核者

New Merge Request · Rongbao V x Members · Rongbao WEI / demo x +

https://git.openearth.community/weirongbao/demo/merge_requests/new?utf8=✓&merge_request%5Bsource_project_i...

GitLab Projects Groups Activity Milestones Snippets Search or jump to...

New Merge Request

From **feature/foo** into **develop** [Change branches](#)

Title

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

Write Preview

The foo feature completed.

Markdown and [quick actions](#) are supported [Attach a file](#)

Assignee [Assign to me](#)

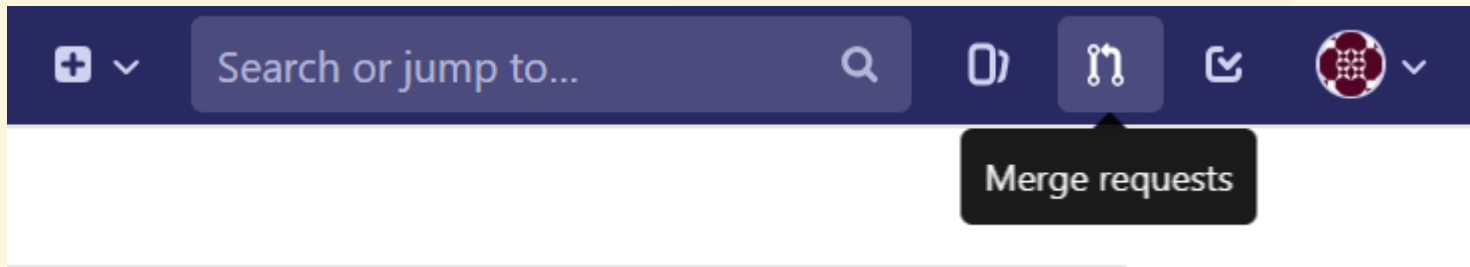
例子：由我为 demo 项目开发一个名为 foo 的新功能

勾选 `Remove source branch when merge request is accepted`
则接受合并后，将删除远程仓库的 `feature/foo` 分支

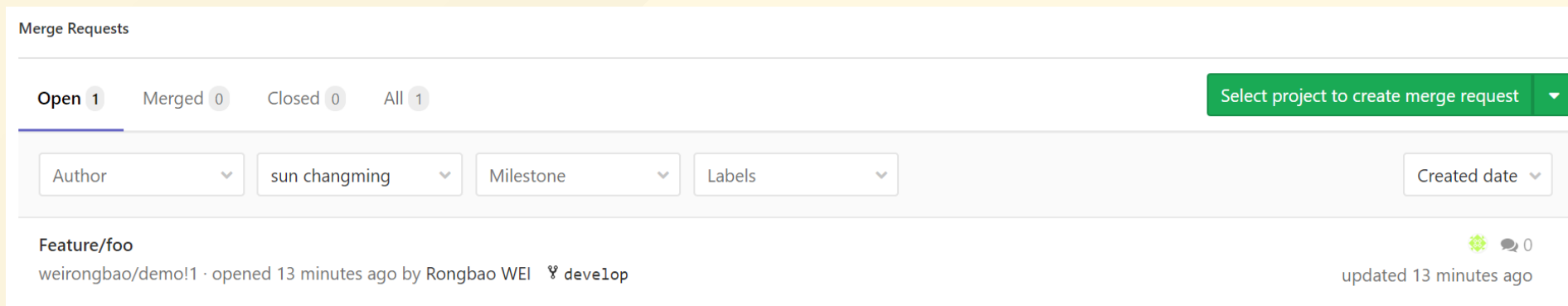
点击页面下面的 `Submit merge request` 按钮提交合并请求

例子：由我为 demo 项目开发一个名为 foo 的新功能

7. 点击项目右上角的 Merge requests 按钮，查看请求



选择筛选器查找刚才的请求



例子：由我为 demo 项目开发一个名为 foo 的新功能

8. 审阅者可以对代码进行批注

The screenshot displays the GitLab web interface for a Merge Request. The browser address bar shows the URL: `https://git.openeearth.community/weirongbao/demo/merge_requests/1/diffs`. The page title is "Feature/foo (!1) · Merge Requests".

The interface includes a sidebar on the left with navigation options: Discussion (0), Commits (2), and Changes (2). Below this is a file filter section showing "Filter files" and a list of files: "README.md" (+5 -0) and "main.c" (+6 -0). It indicates "2 changed files" and "11 additions 0 deletions".

The main content area shows a diff for the file "main.c". The diff highlights changes in the code, including the addition of a main function that prints "Hello World".

A comment box is open over the diff, with the text: "@weirongbao Please Remove `\\n`". The comment box has a "Write" tab, a "Preview" tab, and a rich text editor with various formatting options (bold, italic, quote, code, link, list, table, image, video). Below the text area, it says "Markdown is supported" and "Attach a file". There are "Comment" and "Cancel" buttons at the bottom of the comment box.

On the right side, there is a "Todo" section with an "Add todo" button. Below this are fields for "Assignee" (sun changming, @sunchangming), "Milestone" (None), "Time tracking" (No estimate or time spent), "Labels" (None), "Lock merge request" (Unlocked), "2 participants" (with avatars), and "Notifications" (toggle switch).

例子：由我为 demo 项目开发一个名为 foo 的新功能

9. 审阅者点击 Merge 按钮，合并请求

- Merge 前可以勾选：

- ☒ Remove source branch （删除来源分支）

- ☒ Squash and merge （合并为一个提交）

“当正式发布时，维护人员用同样的方法，合并 develop 分支到 master，并且添加 Tag 标记，如 v1.0.0。”

Overview

1. 初始配置规范
2. 提交描述规范
3. 代码合并规范
- 4. Gitlab保护分支和Tag**
5. 里程碑和 Issues

Gitlab保护分支和Tag

项目维护者应该在创建项目的时候进行下面操作

1. 添加 `README.md` 和 `.gitignore` , 并创建 `develop` 分支
2. 对 `master` 和 `develop` 分支保护
3. 对 `v*` tag 保护

参考 : [Git 实践 - 创建新项目](#)

Overview

1. 初始配置规范
2. 提交描述规范
3. 代码合并规范
4. Gitlab保护分支和Tag
- 5. 里程碑和 Issues**

里程碑和 Issues

- 在开发周期确定后，请维护者在 Gitlab 的 Issues -> Milestones 中，新建 Milestones，设定 Start Date 和 Due Date。
- 里程碑设定后，在 Issues -> List 页面，创建 Issue，每个 feature 或者 bugfix 创建一个 Issue，并且指定 Issue 的 Milestones 为刚刚创建的里程碑。
- 建议创建 Labels 来管理 Issues，比如功能 Issue 的 Label 可以是 **dev/feature**，修复 BUG 的 Issue 可以是 **dev/bugfix**，热修复的 Issue 可以是 **dev/hotfix**。

里程碑和 Issues

- 在某个 Issue 被标记为完成后，Milestones 页面会显示开发进度。一个里程碑开发完成，维护者可以 Close Milestones，合并 develop 分支到 master 分支上，QA进行对 master 进行测试。
- 维护者配合 QA 工作人员创建 Jenkins 项目，利用 jenkins-gitlab 插件，对 master 节点提交测试。在 Gitlab 的 Settings -> Integrations 中对 Push 和（或）Merge request 事件添加 Webhook。

Thank you! 😊