**Teamwork với Github**

**MỤC LỤC**

[1. Commit message 2](#_Toc150522962)

[**Cấu trúc chung của 1 commit message theo conventional:** 2](#_Toc150522963)

[**Type theo conventional commit:** 3](#_Toc150522964)

[2. Pull request 4](#_Toc150522965)

[**Chuẩn bị pull request:** 4](#_Toc150522966)

1. **Commit message**

Thực ra là không có cách viết nào là đúng và cũng không có cách viết nào là sai cả, bởi nó cũng chỉ là 1 một dung text cho việc thể hiện thôi. Do vậy nếu bạn có commit "abc" thì cũng không có sao hiện tại, nhưng tương lại thì có đấy. Tuy nhiên, nếu trong một project mỗi người viết commit message một kiểu thì khi nhìn vào commit history nó cũng khá giống 1 mớ bùi nhùi, tệ =)). Chưa kể khi cần chúng ta cần tìm kiếm và xem lại commit trong đống commit của vài tháng trước đó thông qua các commit message mà không có bất cứ quy tắc nào:

* Đọc commit message mà không phán đoán được commit đấy sinh ra với mục đích gì.
* Không tóm lược các thay đổi trong code.
* Không theo 1 quy tắc nào nên việc tìm kiếm nhanh gần như bất khả thi.

## **Cấu trúc chung của 1 commit message theo conventional:**

<type>[optional scope]: <description>

[optional body]

[optional footer]

Trong đó:

* Các thành phần type, description là bắt buộc cần có trong commit message, optional là tùy chọn có hoặc không có cũng được.
* Type: từ khóa để phân loại commit là feature, fix bug, refactor.
* Scope: cũng được dùng để phân loại commit, vùng ảnh hưởng của commit, trả lời câu hỏi: commit này refactor|fix cái gì? được đặt trong cặp ngoặc đơn ngay sau type. VD: feat(authentication):, fix(parser):
* Description: là mô tả ngắn về những gì sẽ bị sửa đổi trong commit đấy
* Body: là mô tả dài và chi tiết hơn, cần thiết khi description chưa thể nói rõ hết được, có thể thêm phần ghi chú bằng các keyword.
* Footer: một số thông tin mở rộng như số ID của pull request, issue.. được quy định theo conventional.

Một số ví dụ ngắn gọn:

feat: add validate of A feature

fĩx: fix die dashboard page

feat(feature\_a): add validate of A1 feature

Ví dụ đầy đủ:

feat(user-auth): add password reset functionality

Adds a new endpoint for users to request a password reset.

Issue: #123

Task: PROJ-456

## **Type theo conventional commit:**

Một số type phổ biến được khuyên dùng bao gồm:

* feat: thêm một feature
* fix: fix bug cho hệ thống, vá lỗi trong codebase
* refactor: sửa code nhưng không fix bug cũng không thêm feature hoặc đôi khi bug cũng được fix từ việc refactor.
* docs: thêm/thay đổi document
* chore: những sửa đổi nhỏ nhặt không liên quan tới code
* style: những thay đổi không làm thay đổi ý nghĩa của code như thay đổi css/ui chẳng hạn. perf: code cải tiến về mặt hiệu năng xử lý
* vendor: cập nhật version cho các dependencies, packages.

Ngoài ra, với mỗi dự án type này chúng ta có thể custom thêm được theo ý định (vì cũng chỉ để phân loại thôi), vd: Optimize, Bump, Drop, ...

**Note**: Cái này quan trọng, dĩ nhiên phải làm rồi là nội dung của commit phải phù hợp với tương tác và ảnh hưởng của đoạn code được sửa trong commit. Chứ ko phải kiểu fix bug => ghi là feat =)). Rất khó để đọc và hiểu.

1. **Pull request**

Pull request được tạo ra để đưa những file source code của bạn lên 1 host chung nơi mọi người có quyền truy cập sẽ truy cập vào và cùng review, để lại comment trên những file source code đó. Lúc này thời gian review và địa điểm review source code không còn là vấn đề - đó cũng chính là mục đích tạo ra pull request!

## **Chuẩn bị pull request:**

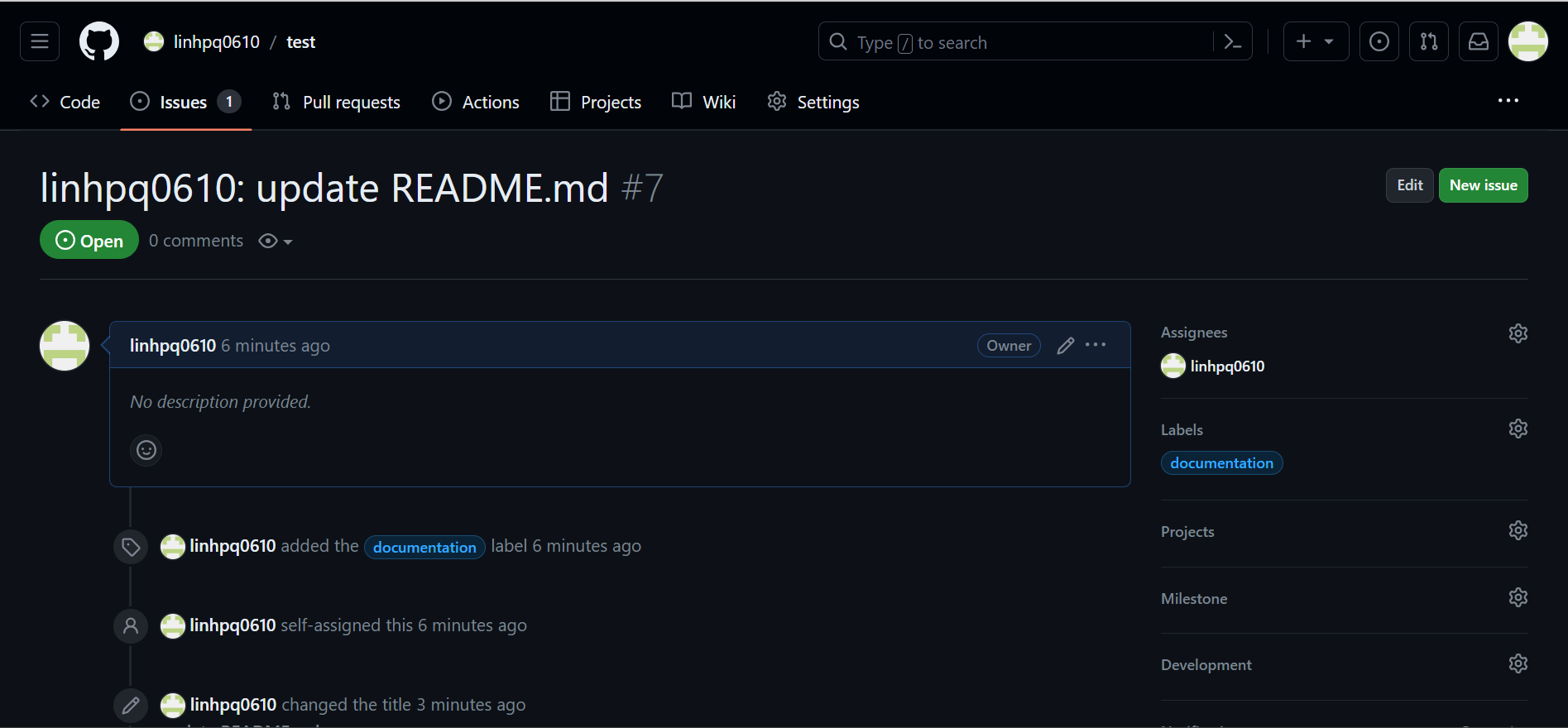
Thứ nhất, chắc chắn là naming. Tên branch phải thể hiện mục đích của việc update. Cái tên nói lên tất cả, nó sẽ giúp cho reviewer có thể nắm bắt nhanh chóng bạn đang làm gì. Thông thường mình sẽ tạo tên format sau:

<type>\_<issue id>\_<tên issue>

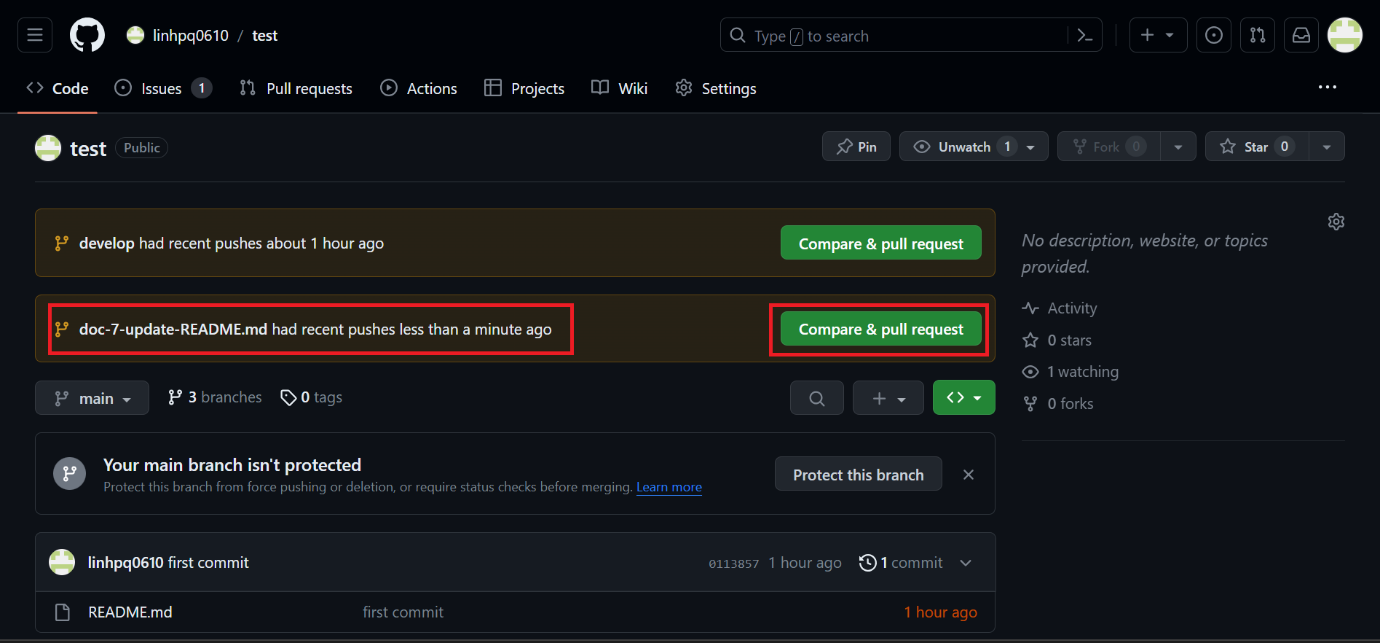
Trong đó:

* Tên tài khoản github của người thực hiện
* Tên issue: mô tả ngắn gọn mục đích của issue. Vì nhìn vào issue ID không thể biết ngay bạn đang làm gì nhưng chỉ cần nhìn thêm tên issue hẳn ai cũng nắm được cơ bản công việc của bạn.

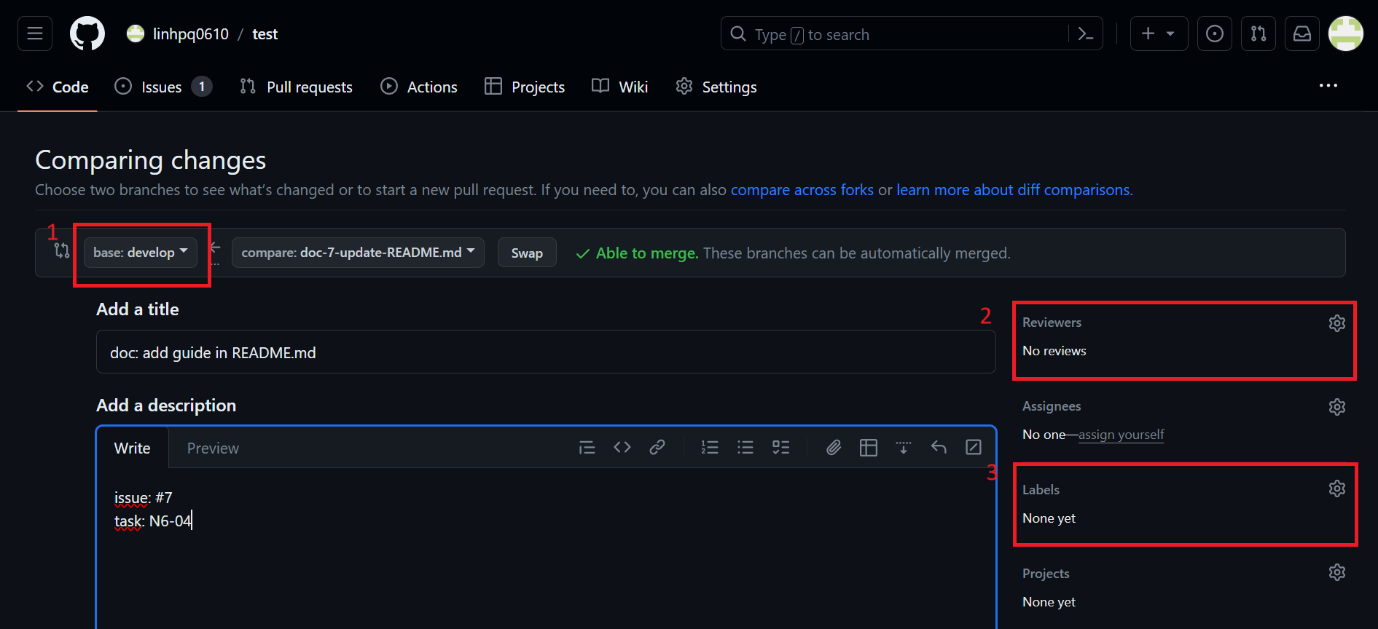
Ví dụ như có issue như này:



doc-7-update-README.md

Sau khi hoàn thành task và push code thì ở giao diện github.com sẽ như này:  


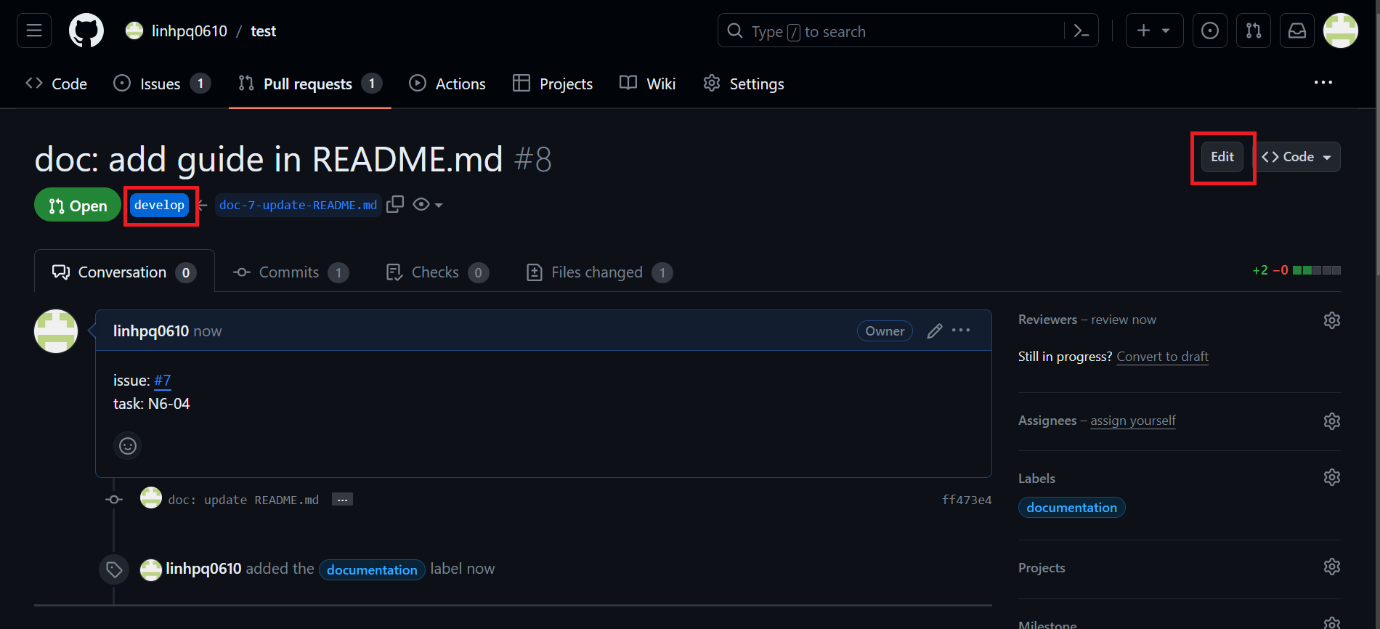
Nhấn vào ‘Compare & pull request’



1: phải luôn là develop

2: thêm người review

3: chọn label giống với issue



Nếu sau khi tạo xong mà nhánh được merge không phải là develop thì có thể sửa lại bằng nút ‘Edit’