Quy ước sử dụng Git

Branch cố định

Mỗi dự án có 3 branch cố định như sau:

1. Branch master: default branch, là branch phát triển
2. Branch staging: branch trung gian để test trước khi triển khai lên production
3. Branch production: branch cho bản production

Tham khảo: https://about.gitlab.com/2014/09/29/gitlab-flow[1i]

Branch feature

Là branch chức năng, được tạo ra để thực hiện task, sau khi task kết thúc, thì remove đi.

Quy ước tên branch feature

```
{mã-định-danh}-{tên-branch-phân-cách-bằng-dấu-gạch-ngang}
```

Trong đó:

- Mã định danh: có thể là User Story ID hoặc Issue ID, viết thường
- Tên branch: viết thường, không dấu, là tên tóm tắt của User Story hoặc Issue. Khuyến khích viết tiếng Anh

Ví dụ:

```
us1-user-can-access-via-browser-ipad
```

```
78-fix-bug-user-cannot-login
```

Các thao tác với Git trong quá trình thực hiện task

Bắt đầu thực hiện task

Checkout branch feature từ branch master

```
git checkout master           # Chuyển sang branch master
```

```
git pull origin master        # Update branch master ở local
```

```
git checkout -b {tên-branch}  # Checkout branch feature từ master và chuyển
tới branch vừa tạo. Tên branch theo quy ước tên branch feature
```

Trong khi thực hiện task

Trường hợp cần commit tạm để lưu

```
git add           # Để add file, nếu add toàn bộ thì thêm dấu .
```

1

```
git commit        # Theo quy ước commit message. Trường hợp không theo quy ước commit message thì khi hoàn thành task sẽ phải sửa commit

git push origin {tên-branch}
```

Trường hợp cần update code từ branch master về

```
git pull origin master –rebase

# Lệnh rebase giúp đẩy các commit ở branch hiện tại ra sau cùng, các commit của branch master sẽ được đây lên trước
```

Khi hoàn thành task

```
git commit        # Theo quy ước commit message

git push origin {tên-branch}        # Để backup branch lên remote

# Gộp commit nếu cần

git pull origin master --rebase     # Update và rebase lại code từ master

git push -f origin {tên-branch}     # Force push branch lên remote

# Tạo merge request (MR)

# Khi push lên branch remote bất kỳ ngoại trừ master, thì lệnh git push sẽ trả về link để tạo MR từ {tên-branch} sang master, hoặc link của MR trong trường hợp đã tạo MR trước đó. Ctrl + click để truy cập link này
```

Quy ước 1 commit

Mỗi task, mỗi MR chỉ có 1 commit.

Lý do:

- Dễ review, dễ trace, dễ revert
- Mỗi commit đều có ý nghĩa, không có commit rác

Các trường hợp ngoại lệ:

1. Trường hợp task có cài thư viện, thêm rất nhiều file/folder vào source code: nên tách riêng commit cài thư viện. Như vậy MR có thể có 2 commit: 1 commit cài thư viện và 1 commit code cho task
2. Trường hợp task lớn, nhiều code và có thể chia ra nhiều bước xử lý, thì có thể chia ra nhiều commit và cần đảm bảo mỗi commit đều có ý nghĩa. Trường hợp này sẽ không gặp nhiều, vì nếu vậy, task đã được chia nhỏ.

Quy ước commit message

Ngôn ngữ và nội dung

1. Khuyến khích viết tiếng Anh
2. Nội dung có ý nghĩa, tránh chung chung như: “Fix bug”, “Update code”…

3. Nội dung bắt buộc có mã issue. Xem phần Mã Issue

Trường hợp commit message ngắn gọn (< 50 ký tự)

```
git commit -m ‘Nội dung message’    # Chữ cái đầu của nội dung message viết hoa đúng chính tả
```

Trường hợp commit message dài

```
git commit        # Trình editor để viết nội dung commit message. Mặc định trên Ubuntu là vi. Cần học cách sử dụng 1 số lệnh cơ bản của vi: i, :w, :wq, :q!
```

Quy ước về cách viết nội dung: *Tham khảo: https://chris.beams.io/posts/git-commit/*

Tóm tắt, 7 quy tắc:

1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line
6. Wrap the body at 72 characters
7. Use the body to explain *what* and *why* vs. *how*

Mã issue

Nội dung commit message bắt buộc có Từ khóa + Mã issue. Trong đó:

- Từ khóa: chỉ quan hệ giữa Commit và Isssue, có thể là những từ khóa sau:
  - Resolves: commit sẽ resolves issue
  - Closes: commit sẽ close issue
  - Relates: commit có liên quan đến issue
- Mã issue: mã issue gitlab hoặc Redmine, Jira… bắt đầu bằng dấu #

Ví dụ:

Cho trường hợp commit message ngắn gọn 1 dòng

```
git commit -m ‘Fix bug user cannot login. Resolves #35’
```

```
git commit -m ‘Admin can block user. Resolves #33, #36’
```

```
Cho trường hợp commit message dài
```

```
git commit
```

```
Fix bug user cannot login

Causes: config wrong guard
Solution: config guard using model users
Resolves #35
Relates #37
```

Gộp commit

Cách 1: reset + commit

Cách này sử dụng để gộp thành 1 commit khi kết thúc task, trước khi tạo MR.

Tại branch feature:

```
git log --oneline -10         # Xem 10 commit gần nhất, xem có bao nhiêu commit tạm. Giả sử có 3 commit tạm

git reset HEAD~3              # Reset 3 commit gần đây, các file đã thay đổi trong 3 commit gần đây vẫn được giữ nguyên và chuyển về trạng thái unstage

git add                       # Add các file cần commit

git commit                    # Theo quy ước commit message. Nhập vào message commit hoàn thành

git log --online -10          # Xem 10 commit gần nhất, để kiểm tra lại việc gộp commit

git push -f origin {tên-branch}     # Force push lên branch feature remote
```

Cách 2: rebase commit

Tham khảo: https://help.github.com/en/articles/about-git-rebase

---

i