1. GIT FLOW

1.1. Cấu trúc project

- Các môi trường chính:

  - Môi trường development: Môi trường làm việc của Dev, Dev sẽ code cũng như test ở môi trường này
  - Môi trường staging: Môi trường làm việc của PM với khách hàng, demo cho khác hàng xem sản phẩm cũng trên môi trường này
  - Môi trường production: Môi trường cuối cùng khi sản phẩm hoàn tất

- Một project sẽ có các nhánh:

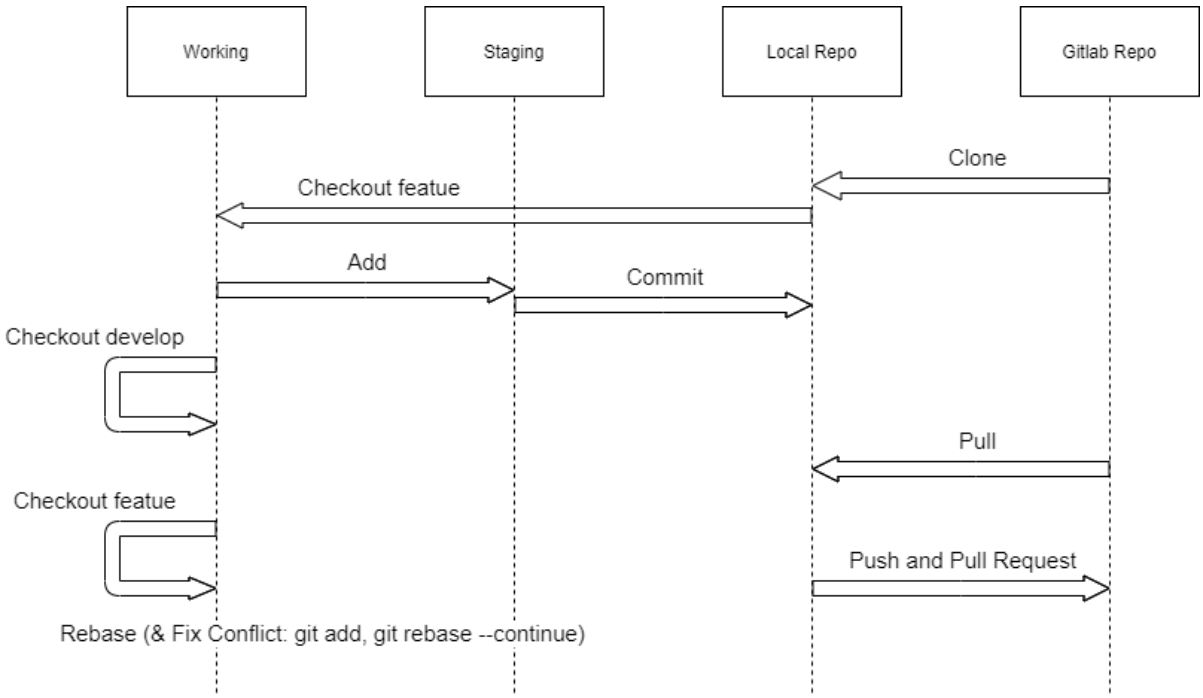
  - Master: Là nhánh chứa code cuối cùng của dự án trước khi được auto test, deploy lên hai môi trường staging và production
  - Developer: Là nhánh mặc định, nhánh làm việc của Dev

- Phân quyền:

  - PM: Người quản lý dư án, có quyền merge các pull request vào nhánh Develop, merge code từ develop vào master, có tất cả các quyền của Dev
  - Dev: Lập trình viên làm việc với nhánh develop, tạo ra các nhánh riêng để làm việc, có quyền tạo pull request

1.2. Quy trình git flow

- Quy trình git flow được mô tả trong hình vẽ sau:



- Tóm tắt quy trình:

- Clone code về repository ở máy local
- Cd đến thư mục chứa code
- Checkout sang một nhánh mới để làm việc (lưu ý: mỗi một tính năng là một nhánh mới)
- Sau khi thay đổi, code xong ta sẽ add code và tạo commit
- Checkout sang nhánh develop để pull code mới nhất về
- Checkout ngược lại sang nhánh ta đang làm việc
- Đứng ở đây rebase với nhánh develop để đồng bộ code, nếu xảy ra conflict sẽ xử lý ở bước này
- Push code lên và tạo pull request

- Các câu lệnh tương ứng:

- $ git clone
- $ git checkout -b feature
- $ git add *
- $ git commit -m “fix api get all user”
- $ git checkout develop
- $ git pull origin develop
- $ git checkout feature
- $ git rebase develop (rebase với nhánh develop để đồng nhất code mới nhất)
- Nếu xảy conflict thực hiện hai lệnh sau:
  - $ git add
  - $ git rebase –continue
- $ git push origin feature
- Tạo pull request trên remote gitlab

1.3. Quy trình xử lý conflict

- Conflict xảy ra khi hai hoặc nhiều người cùng chỉnh sửa một file, git không biết ta sẽ chọn đoạn code nào

- Người xảy ra lỗi conflict là người code sau, cần chủ động bàn bạc với code của người được merge trước đó

- Sau khi bàn bạc thì thống nhất code, phải được đồng ý bởi hai người, nếu không PM sẽ là người ra quyết định

- Sử dụng **$ git add** để thêm thay đổi

- Sử dụng **$ git rebase –continue** để tiếp tục rebase so sánh với code mới nhất

- Sau đó push code lên nhánh của mình

2. QUY TẮC

1.1. Quy tắc đặt tên nhánh

- Mỗi chức năng phải checkout sang một nhánh mới để làm việc

- Quy tắc đặt tên:

STT	Mục đích	Cách đặt tên nhánh
1	Tạo một chức năng mới	Feature/tên_chức_năng
2	Fix bug	Fix/tên_chức_năng_cần_fix
3	Refactoring	Refactoring/tên_chức_năng_cần_refactoring
4	Task mới	Task/Tên_task_mới
5	Hot fix	Hotfix/tên_chức_năng

- Nhánh Hotfix được tạo ra chỉ khi ta muốn tạo pull request trực tiếp vào nhánh master trong trường hợp khẩn cấp

1.2. Quy tắc đặt tên commit

- Quy tắc đặt tên commit:

- Commit phải tóm tắt được rõ ràng chức năng, task của mình đang làm. Ví dụ: **fix api get all categories v1**
- Commit phải viết toàn bộ bằng chữ thường
- Commit phải viết bằng tiếng anh
- Commit nên bắt đầu bằng động từ như: **add, fix, update, refactor,…**
- Không nên dùng từ viết tắt cho commit
- Comit không được kết thúc bằng dấu chấm câu như: **., !, ?, …**

- Mỗi pull request chỉ nên tối đa 1 commit, tối đa 3 file chỉnh sửa (trừ initial project)

- Két thúc một ngày đều phải commit lên gitlab, kể cả công việc đang làm dở