Git flow

**1.Cài đặt**

* Dowload git tại: <https://git-scm.com/>
* Hướng dẫn cài đặt: [https://www.thegioididong.com/game-app/huong-dan-cach-tai-cai-dat-cau-hinh-git-tren-laptop-may-tinh-1299681#:~:text=c%C3%A0i%20%C4%91%E1%BA%B7t%20Git-,1.,%3E%20Ch%E1%BB%8Dn%20Install%20%3E%20Nh%E1%BA%A5n%20Finish](https://www.thegioididong.com/game-app/huong-dan-cach-tai-cai-dat-cau-hinh-git-tren-laptop-may-tinh-1299681).
* Note: Ngoài chọn vị trí lưu trữ cài đặt ra thì cứ nhấn next đến hết.

**2.Tổng quan về mô hình Git**

Mô hình Git xoay quanh một kho lưu trữ từ xa (remote repository). Các thành viên trỏng team sẽ có các kho lưu trữ cục bộ (local repository).

Khi một thành viên trong team commit và push, mã của họ sẽ được đẩy lên remote repository và những thành viên khác trong team sẽ thấy nhưng thay đổi này và có thể pull những thay đổi này về máy họ.

Để làm việc hiệu quả thì tạo ra những loại branch sau:

Branch chính: master và develop.

Branch phụ: feature, hotfix, release.

1. **Branch chính trong Git.**

*Branch master*:

-Có sẵn sau khi khởi tạo repository, được coi là banch chính chứa phiên bản của mã nguồn của môi trường production.

*Branch develop*:

-Đây là nhánh được tách ra từ nhánh master và luôn đi song son với nhánh master, nó phản ánh trạng thái thay đổi mới nhất trong quá trình phát triển, chuẩn bị cho release tiếp theo.

1. **Branch phụ trong Git.**

Ngoài những branch chính có nhưng branch phụ khác để giúp các thành viên trong team có thể làm việc hiệu quả, dễ dàng theo dõi các tính năng , chuẩn bị cho release hoặc fix nhanh các bug trên môi trường production.

Branch phụ chỉ tồn tại trong một thời gian nắng rồi sẽ bị xoá đi.

*Branch feature:*

Tách ra từ branch develop để code các tính năng mới hoặc cập nhật các tính năng cũ trên nhánh develop.

Quy tắc đặt tên: feature/\*, feature-\*, ...

Branch feature sẽ tồn tại trong quá trình chức năng được phát triển, cuối cùng được merge lại branch develop (khi quyết định lần release tới bao gồm chức năng đó) hoặc bị bỏ đi (khi thấy chắc năng không còn cần thiết).

*Branch release:*

Tách ra từ branch develop để chuẩn bị cho release phiên bản mới lên production.

*Branch hotfix:*

Tách từ branch master.

Merge vào branch: develop và master.

Quy tắc đặt tên: hotfix-\*, hotfix/

**2.Quy tắc quản lý branch.**



Stage area :



Làm chủ git command line.

**git config**

Git config là câu lệnh mà chúng ta phải thực thi đầu tiên cài đặt git lên máy. Câu lệnh này sẽ giúp các bạn thiết lập tên và email cá nhân của bạn, những thông tin này sẽ đính kèm trong mọi commit của bạn, đều này sẽ rất hữu ích khi chúng ta muốn biết đoạn code nào đó đã được ai triển khai để có thể thảo luận trong trường hợp chúng ta không hiểu rõ đoạn code đấy sử dụng cho mục đích gì.

$ git config --global user.name "Your name"

$ git config --global user.email "Your email

### git init

Đây là câu lệnh đầu tiên khi chúng ta bắt đầu một dự án mới, câu lệnh này sẽ giúp chúng ta tạo một repository mới, sau đó nó sẽ được sử dụng để lưu trữ và quản lý mã nguồn trong repository này.

$ git init

// Hoặc bạn có thể đặt tên cho repo với lệnh

$ git init <your repository name>

### git clone

Câu lệnh này sẽ giúp chúng ta download một repository đã tồn tại sẵn trên khô lưu trữu (github, gitlab v.v) về máy.

git clone <your project URL>

### git add

Git add là câu lệnh giúp chúng ta thêm tất cả các file code mới mới hoặc các file code được chỉnh sửa vào repository.

$ git add your\_file\_name - Thêm một file( thêm mới hoặc chỉnh sửa) vào staging area

$ git add \* - Thêm tất cả các file (thêm mới hoặc chỉnh sửa) vào staging area

### Đưa file từ vùng staged về vùng changes

### 2 cách: git restore hoặc reset

### Đưa file từ vùng staged trở về vùng changes

//đưa 1 file hoặc tất cả file .

$ git restore –-staged <filename>||.

//Đưa theo file hoặc tất cả nếu không truyền tên file

$ git reset [<filename>]

### git commit

Đây là câu lệnh được sử dụng phổ biến nhất, câu lệnh này sẽ giúp chúng ta lưu các thay đổi ở các file trong vùng staging area xuống repository.

Có thể hiểu git add dùng để thêm thêm các file được thay đổi hoặc thêm mới vào vùng staging area, và chúng sẽ sẵn sàng để commit và sau đó những thay đổi này sẽ được lưu xuống repository.

$ git commit -m “your useful commit message”

Or

//Thêm tham số -a sẽ tự động git add . rồi commit

$ git commit -a -m “your useful commit message”

### Huỷ commit cuối và đưa những thay đổi vào vùng staged.

$ git reset –-soft HEAD^

Or

//xoá trực tiếp commit cuối và không đưa thay đổi vào vùng staged.

$ git reset -–hard HEAD^

### Phục hồi file về 1 commit hoặc quay trở lại trước khi chưa chỉnh sửa.

//Đưa các file về commit chỉ định.

$ git checkout [HASH] <filename>||.

Or

//đưa file hoặc tất cả các file đang được sửa đổi về trước khi sửa.

$ git restore <filename>||.

### 

### git status

Câu lệnh này cho phép bạn xem tình trạng hiện tại của mã nguồn như có bao nhiêu file được thêm mới hoặc chỉnh sửa.  Những file nào đang nằm trong vùng staging area hoặc đang nằm ngoài staging area.

$ git status -s

* ' ' = unmodified (không đổi)
* M = modified (có sửa đổi)
* A = added (file mới thêm)
* D = deleted (file bị xóa)
* R = renamed (đổi tên file)
* C = copied (file copy từ file khác)
* U = updated but unmerged (đã cập nhật, nhưng chưa merge)

### git diff

Nếu các bạn muốn so sánh một file code nào thay đổi những gì trước khi commit thì các bạn có thể sử dụng

$ git diff HEAD <filename>

Để kiểm tra sự khác nhau giữa mã nguồn hiện tại đã được thay đổi so với local repo

$ git diff HEAD <filename>

So sánh 2 branch

### git branch

Trong một Git repository luôn luôn tồn tại nhiều nhánh riêng biệt dùng để triển khai một tính năng nào đó độc lập với các nhánh khác.

Dùng để hiển thị tất cả các branch đang có.

$ git branch

Dùng để tạo một branch mới.

$ git branch <branch\_name>

### git checkout/switch

Để di chuyển qua lại giữa các branch, chúng ta có thể sử dụng git checkout để đạt được điều này.

git checkout <branch\_name>

Ngoài ra các bạn có thể vừa chuyển qua một branch mới và tiện thể khởi tạo nếu chưa tồn tại với câu lệnh.

$ git checkout -b <your\_new\_branch\_name>

$ git switch -c <branch\_name>

### git merge

Git merge cho phép các bạn kết mã nguồn và những thay đổi trên một branch khác vào branch hiện tại.

$ git merge <branch\_name>

### git log

Với câu lệnh git log các bạn có thể xem tất cả những commit trước đó được sắp xếp theo thứ tự commit gần nhất cho đến những commit cũ hơn.

$ git log

### git stash

Git stash cho phép chúng ta lưu trữ các file được chỉnh sửa trong vùng nhớ tạm.

$ git stash

//stash file chỉ định

$ git stash -- <file1> <file2>

Nếu muốn xem tất cả các stash các bạn có thể sử dụng lệnh

$ git stash list

Nếu bạn muốn áp dụng các chỉnh sửa trong một stash nào đó lên branch hiện tại đang sử dụng.

$ git stash apply “<tên stash>”

or

$ git stash pop “<tên stash>”

### Làm việc với remote Repository

### git remote

Repository được các bạn khởi tạo với câu lệnh git init chỉ đang tồn tại trên máy local của các bạn. Nếu muốn lưu trữ repository này lên một dich vụ lưu trữu git từ xa nào đó chẳng hạn như gitlab, github thì các bạn cần phải sử dụng git remote để kết nối giữa chúng.

$ git remote add <shortname> <url>

$ git remote add origin https://dev.azure.com/aCompiler/\_git/DemoProject

### git push

Khi đã kết nối giữa local và dịch vụ lưu trữ git, chúng ta cần sử dụng lệnh git push để đồng bộ những thay đổi được commit trên local lên dich vụ lưu trữ.

$ git push -u <short\_name> <your\_branch\_name>

Ví dụ

$ git push -u origin feature\_branch

Một số tham số hay dùng như:

* --all đẩy tất cả các nhánh lên server
* --tags đẩy tất cả các tag lên server
* --delete xóa một nhánh chỉ ra trên server
* -u đẩy và tạo một upstream (luồng upload tương ứng với nhánh của local), hay sử dụng cho lần đầu đẩy lên server

--force ghi đè toàn bộ một nhánh ở remote

### git fetch

Git được sử dụng để làm việc nhóm, quản lý mã nguồn. Ngoài những commit của bạn thì còn vô số commit khác của các thành viên khác trong team. Sử dụng git fetch sẽ giúp chúng ta cập nhật tất cả những thông tin mới như commit, branch, v.v.

$ git fetch

### git pull

Câu lệnh này sẽ download tất cả những nội dung (không chỉ là metadata như git fetch) từ dịch vụ lưu trữ xuống local repository.

$ git pull <remote\_url>