SỔ TAY GIT - GITHUB

# Version Control System (VCS)

* + Định nghĩa VCS: dịch là Hệ thống quản lý phiên bản, là hệ thống lưu trữ và quản lý thay đổi của các tệp trong dự án, cho phép theo dõi và khôi phục các phiên bản trước đó.
  + Phân loại VCS:

+ Local VCS: quản lý phiên bản trên máy cục bộ

+ Centralized VCS (CVCS): hệ thống quản lý phiên bản tập trung

+ Distributed VCS (DVCS): hệ thống quản lý phiên bản phân tán, như Git

# Git

* + Git là một DVCS phổ biến, ghi lại toàn bộ lịch sử thay đổi của mã nguồn trong dự án.
  + Khác với các VCS khác, Git lưu trữ dữ liệu dưới dạng các “snapshot” của toàn bộ tệp tại mỗi thời điểm commit.

# Cài đặt Git

* + Ở đây ta cài Git trên Windows, tải về từ: <https://git-scm.com/downloads>
  + Cài đặt với các tuỳ chọn mặc định (next next…)
  + Kiểm tra cài đặt bằng lệnh:

git --version

* + Thiết lập tên và email (quan trọng để commit):

git config --global user.name “Tên của bạn” git config --global user.email “Email của bạn”

* + Kiểm tra cấu hình:

git config --list

# Sử dụng Git

* + Sau khi cài đặt, Git được sử dụng thông qua cửa sổ dòng lệnh với terminal (command-line). Ngoài ra cũng có một số GUI tool giúp bạn dùng mà không phải dùng lệnh.
  + Ở đây chúng ta sẽ sử dụng Git trên terminal nhé.

# Repository

## Repository là gì?

Repository là kho lưu trữ tất cả những thông tin cần thiết để quản lý các sửa đổi và lịch sử của toàn bộ project. Repository của Git được phân thành 2 loại là:

* + Local Repository: là repository nằm trên chính máy tính của chúng ta, repository này sẽ đồng bộ hoá với remote repository bằng các lệnh của Git. Bạn có thể thêm, sửa, xoá file, tạo commit để lưu lại nhưng chưa thể dùng để chia sẻ tới người khác được. Để tạo local repository, tại folder chứa source bạn thực hiện command line:
    - git init

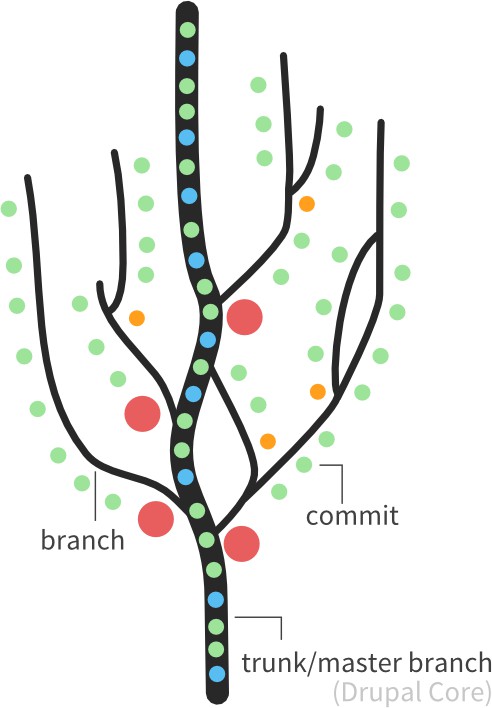
Initialized empty Git repository in /home/ubuntu/my-project/.git/

* + Server Repository (hay Remote Repository): là repository được lưu tại server của các hosting- service sử dụng Git. Một số hosting-service các bạn có thể biết như: Github, Gitlab, Bitbucket,… Repository này có thể dùng để share tới những người khác để họ có quyền truy cập và lấy source code về. Để tạo server repository, bạn truy cập vào trang web của hosting-service bạn dùng và thực hiện vài cú click tuỳ nhà cung cấp dịch vụ.

## Làm việc với Repository

* + Tạo một repository trên máy, ta chỉ cần tạo một folder cho nó, rồi thực hiện câu lệnh:
    - git init
  + Tạo một local repository từ server repository:
    - git clone repository\_url

# Commit



## Commit là gì?

Trên mỗi branch bạn làm việc, sau khi sửa đổi các file source code… Những thay đổi đó cần được lưu lại bằng cách tạo ra một điểm mốc đánh dấu. Điểm đánh dấu các thay đổi này gọi là Commit. Tại mỗi commit, Git chụp lại toàn bộ dữ liệu, tạo ra một snapshot version hoá dữ liệu.

Mỗi commit được tạo ra gồm một số thông tin chính:

* Tên và email của người tạo commit
* Ngày giờ commit được tạo ra
* Message: mô tả cho các sự sửa đổi
* Một ID là một chuỗi SHA gồm 2 dạng: Full (chuỗi SHA đầy đủ) và Short (7 kí tự đầu tiên được cắt ra từ chuỗi full)

## Làm việc với Commit

* Tạo một commit lưu lại các sửa đổi:

# Cú pháp: git commit -m <message>

* + git commit -m "Message mô tả thay đổi"

# Lưu lại thay đổi nhưng gì đè lên commit trước đó:

* + git commit --amend -m "Message mô tả thay đổi"

# Ghi đè commit trước đó và đổi người tạo commit:

# git commit --amend --author "Your name <your email>" # Chú ý ghi cả dấu < > trong case này

* + git commit --amend --author "Le Gia Khanh <[leogiakhanh1609@gmail.com](mailto:leogiakhanh1609@gmail.com)>"
* Xem lại danh sách n các commit đã tạo gần đây:

# Cú pháp: git log -<n>

* + git log -5

# Hiển thị mỗi commit trên một dòng

* + git log --oneline -5
* So sánh các thay đổi của code trước khi add thay đổi vào commit:
  + git diff
* Tạo commit vào message trên nhiều dòng (mặc định chỉ trên một dòng), sử dụng dấu ‘ thay thế “ khi tạo message:
  + git commit -m '- First line
  + - Second line'
* Gộp nhiều commit làm 1 commit. VD: Gộp 3 commit gần nhất làm 1

+ Thực hiện command:

* + git rebase -i HEAD~3

+ Một file mới hiện ra với danh sách 3 commit gần nhất:

pick fd3fea3 Three pick fd3fea2 Two

pick fd3fea1 One

Dòng commit đầu giữ nguyên, hai dòng thứ 2, 3 sửa pick thành s để gộp. Sửa pick thành d để xoá commit tại dòng đó. Ctrl+O, Enter để lưu lại.

+ Sửa lại commit message, chèn dấu # vào đầu dòng các commit message để loại bỏ message đó. Để lại một message cho commit mới sau khi gộp. Ctrl+O, Enter lưu lại là xong.

* Resolve conflict sau khi tạo merge request. Checkout về branch mà bạn dùng để gửi merge\_request.

+ Fetch mới lại target branch của merge request trên máy bạn:

* + git fetch <remote\_name> <target\_branch\_name>:<target\_branch\_name>

+ Rebase lại target branch:

* + git rebase <target\_branch>

Sau khi chạy lệnh rebase, những file bị conflict sẽ hiển thị với từ khoá conflict. Bạn mở các file đấy lên và xem những phần code bị conflict. Những đoạn code này sẽ được chia ra 2 phần, phần nửa trên là code hiện tại của target\_branch, nửa dưới là code mà bạn đang sửa. Hãy đọc và giữ lại những phần code phù hợp, xoá bỏ các dấu == phân cách.

+ Thêm lại các file sửa đổi vào commit và kết thúc rebase:

* + git add .
  + git rebase --continute

+ Trường hợp bạn chưa biết resolve những đoạn conflict như nào, có thể huỷ bỏ quá trình rebase bằng cách:

* + git rebase –-abort

Trước khi tạo một commit, nếu bạn chưa config thông tin định danh của bạn, Git sẽ yêu cầu bạn làm việc này trước khi cho phép commit. Câu lệnh config:

* + git config user.name "Le Gia Khanh"
  + git config user.email [leogiakhanh1609@gmail.com](mailto:leogiakhanh1609@gmail.com)

# Branch

## Branch là gì?

Trong Git, repository có thể ví von như gốc cây, còn branch là các nhánh mọc ra từ gốc đó. Master (hoặc main) branch là nhánh chính, nơi lưu trữ source code gốc. Nếu chỉ làm việc trên một branch duy nhất, dự án sẽ giống như một cái cây trơ trụi.

Branch giúp quản lý và phát triển nhiều tính năng cùng lúc mà không ảnh hưởng đến nhánh chính. Khi cần, ta có thể chuyển đổi giữa các branch hoặc gộp chúng lại.

## Làm việc với branch

Giả sử bạn đang ở master sau khi tạo mới repository.

* Tạo một branch mới có tên là feature/a:

# Cách 1, chỉ thực hiện tạo branch:

* git branch feature/a

# Cách 2, tạo mới branch và chuyển luôn sang branch đấy để code:

* git checkout -b feature/a
* Xem danh sách branch hiện có:
  + git branch feature/a

\* master

Branch hiện tại bạn đang working sẽ có dấu \* ở đầu dòng. Khi có quá nhiều branch trong danh sách, Git sẽ hiển thị dạng ScrollView, bạn nhấn nút điều hướng lên/xuống để cuộn lên/xuống.

Muốn thoát khỏi chế độ cuộn, gõ:

* + :q

Rồi nhánh Enter để thoát khỏi danh sách.

* Chuyển đổi qua lại giữa các branch, thực hiện command:
  + git checkout <branch\_name> # Ví dụ:
  + git checkout feature/a

Khi muốn chuyển sang một branch khác, hãy hết sức lưu ý rằng bạn cần phải tạo thói quen commit lại các thay đổi trước khi chuyển bằng lệnh:

* + git commit -m "My message for note"
* Xoá branch:

# Xóa local branch

* + git branch -D branch/name

# Xóa remote branch <xem phần dưới>

* + git push remote\_name branch/name –delete
* Đẩy code lên remote branch:

# Cú pháp: git push <remote\_name> <branch\_name>

* + git push origin master

# Nếu muốn bắt buộc ghi đè lên remote branch, thêm -f (force)

* + git push orgin master -f
* Merge branch, để merge code từ nhánh B vào nhánh A:

+ Checkout về nhánh A: > git checkout A

+ Merge nhánh B vào: > git merge B

+ Nhập message, lưu lại là xong.

* Đổi tên branch của branch hiện tại:

git branch -m new-name

# Remote và Remote Branch

## Remote

* + Remote là tham chiếu đến repository trên server.
  + Chứa các thông tin:

+ Tên remote (thường mặc định là origin sau khi clone)

+ URL của repository (HTTP hoặc SSH)

* + HTTP URL yêu cầu nhập username/password mỗi lần thao tác
  + SSH URL giúp thao tác không cần nhập username/password

## Remote Branch

* + Remote branch là branch trên remote repository.
  + Có tiền tố (prefix) là tên của remote để phân biệt:

+ master: branch local

+ origin/master: branch master trên remote origin

+ upstream/master: branch master trên remote upstream

## Làm việc với Remote

* + Thêm một remote:

# Cú pháp: git remote add <remote\_name> <remote\_url>

* + - git remote add upstream git@github.com/LGKAI/demo-repository.git
  + Xem danh sách remote đang có:
    - git remote -v
  + Thay đổi url của remote:

# Cú pháp: git remote set-url <remote\_name> <remote\_url>

* + - git remote set-url upstream <https://github.com/LGKAI/demo-repository>
  + Thay đổi tên của remote:

# Cú pháp: git remote rename <old\_name> <new\_name>

* + - git remote rename upstream demo
  + Xoá một remote:

# Cú pháp: git remote remove <remote\_name>

* + - git remote remove demo
  + Lấy code của một pull request bất kì trên remote repository về local để làm việc:

# Cú pháp: git fetch <remote\_name> pull/<pull\_id>/head:<local\_branch\_name>

* + - git fetch pull/2210/head:new-feature

# Git Flow

Để làm việc với Git hiệu quả, cả team của dự án cần tuân thủ các quy tắc của Git Flow.

* + Nhánh chính trong repository:

+ master: chứa code hoàn chỉnh, dùng để deploy production.

+ develop: chứa code mới nhất đang phát triển, dùng để deploy testing.

+ Cả 2 nhánh cần được bảo vệ, chỉ người có quyền mới có thể merge/push.

* + Quy trình làm việc của developer:

+ Fork repository gốc về repository cá nhân.

+ Tạo branch mới từ develop, đặt tên theo quy tắc: feature/branch-name (tính năng mới) và

fix/feature-a (sửa lỗi).

+ Phát triển, commit code và tạo pull request (merge request) về repository gốc.

* + Hotfix (sửa lỗi khẩn cấp trên production):

+ Nếu cần fix bug gấp, có thể tạo pull request thẳng vào master, bỏ qua testing.

+ Merge commit hotfix vào develop để đảm bảo code luôn đồng bộ.

* + Flow chính:

Phát triển (feature branch) → Merge vào develop → Testing → Merge develop vào master để deploy production

Lúc này, quá trình làm việc với Git Flow có thể được mô tả qua bức ảnh sau:

