**TÌM HIỂU VỀ GIT**

**1. Git là gì**

Git là một hệ thống quản lý phiên bản phân tán **(Distributed Version Control System – DVCS)**, nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. **Git**cung cấp cho mỗi lập trình viên kho lưu trữ (**repository**) riêng chứa toàn bộ lịch sử thay đổi.

**Version Control System – VCS là gì?**

VCS là viết tắt của **Version Control System** là **hệ thống kiểm soát các phiên bản phân tán mã nguồn mở**. Các VCS sẽ lưu trữ tất cả các file trong toàn bộ dự án và ghi lại toàn bộ lịch sử thay đổi của file. Mỗi sự thay đổi được lưu lại sẽ được vàthành một version (phiên bản).

VCS nghĩa là hệ thống giúp lập trình viên có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (**clone**) từ một kho chứa mã nguồn (**repository**), mỗi thay đổi vào mã nguồn trên local sẽ có thể ủy thác (**commit**) rồi đưa lên server nơi đặt kho chứa chính.

Và một máy tính khác nếu họ có quyền truy cập cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.

Lập trình viên có thể xem lại danh sách các sự thay đổi của file như xem một dòng thời gian của các phiên bản. Mỗi phiên bản bao gồm: nội dung file bị thay đổi, ngày giờ sửa đổi, người thay đổi là ai, lý do thay đổi hay tên phiên bản…

## VCS có tác dụng như thế nào?

1. Lưu lại lịch sử các version của bất kỳ thay đổi nào của dự án. Giúp xem lại các sự thay đổi hoặc khôi phục (revert) lại sau này.
2. Việc chia sẻ code trở nên dễ dàng hơn, lập trình viên có thể để public cho bất kỳ ai, hoặc private chỉ cho một số người có thẩm quyền có thể truy cập và lấy code về.

Vốn là một VCS nên Git cũng ghi nhớ lại toàn bộ lịch sử thay đổi của source code trong dự án. Lập trình sửa file, thêm dòng code tại đâu, xóa dòng code ở hàng nào…đều được Git ghi nhận và lưu trữ lại.

**2. Tác dụng của Git**

Các dự án thực tế thường có nhiều lập trình viên làm việc song song. Vì vậy, một hệ thống kiểm soát phiên bản như Git là cần thiết để đảm bảo không có xung đột code giữa các lập trình viên.

Ngoài ra, các yêu cầu trong các dự án như vậy thay đổi thường xuyên. Vì vậy, một hệ thống kiểm soát phiên bản cho phép các nhà phát triển revert và quay lại phiên bản cũ hơn của code.

Cuối cùng, đôi khi một số dự án đang được chạy song song liên quan đến cùng một cơ sở code. Trong trường hợp như vậy, khái niệm phân nhánh trong Git là rất quan trọng.

* Dễ sử dụng, thao tác nhanh, gọn, lẹ và rất an toàn.
* Sễ dàng kết hợp các phân nhánh (branch), có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều.
* Chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa là bạn có thể làm việc ở mọi lúc mọi nơi.
* Deployment sản phẩm của bạn một cách không thể nào dễ dàng hơn.

3. Terminal/Command line tool

**Command Line** (Dòng lệnh): Là giao diện được thiết kế để người sử dụng có thể ra mệnh lệnh cho máy thực hiện chỉ bằng bàn phím thông qua các câu lệnh ở dạng dấu nhắc lệnh (**prompts**).

**Command Line Tools** là bộ công cụ lệnh bổ sung được tích hợp trong **Xcode** – bộ phát triển phần mềm tích hợp được Apple phát triển chạy trên hệ điều hành Mac để các lập trình viên có thể phát triển phần mềm chạy trên hệ điều hành Mac và iOS được phát hành vào năm 2003.

**4. Các lệnh git**

#### **1) git config**

Tác dụng : Để set user name và email của bạn trong main configuration file.  
Cách xài : Để kiểm tra tên và kiểu email trong cấu hình dùng ***git config -- global user.name*** và ***git config -- global user.email***. Để set email hoặc tên mới ***git config -- global user.name = “Hải Nguyễn”*** và ***git config -- global user.email = “hainguyen@gmail.com”***

#### **2) git init**

Tác dụng : Khởi tạo 1 git repository 1 project mới hoặc đã có.

Cách xài: ***git init*** trong thư mục gốc của dự án.

#### **3) git clone**

Tác dụng: Copy 1 git repository từ remote source.

Cách xài: ***git clone <:clone git url:>***

#### **4) git status**

Tác dụng: Để check trạng thái của những file bạn đã thay đổi trong thư mục làm việc. VD: Tất cả các thay đổi cuối cùng từ lần commit cuối cùng.

Cách xài: ***git status*** trong thư mục làm việc.

#### **5) git add**

Tác dụng: Thêm thay đổi đến stage/index trong thư mục làm việc.

Cách xài: ***git add***

#### **6) git commit**

Tác dụng: commit nghĩa là một action để Git lưu lại một snapshot của các sự thay đổi trong thư mục làm việc. Và các tập tin, thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của code kèm theo tên và địa chỉ email của người commit. Ngoài ra trong Git bạn cũng có thể khôi phục lại tập tin trong lịch sử commit của nó để chia cho một branch khác, vì vậy bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó.

Cách dùng: ***git commit -m ”Đây là message, bạn dùng để note những thay đổi để sau này dễ dò lại”***

#### **7) git push/git pull**

Tác dụng: Push hoặc Pull các thay đổi đến remote. Nếu bạn đã added và committed các thay đổi và bạn muốn đẩy nó lên hoặc remote của bạn đã update và bạn apply tất cả thay đổi đó trên code của mình.

Cách dùng: ***git pull <:remote:> <:branch:>*** and ***git push <:remote:> <:branch:>***

#### **8) git branch**

Tác dụng: liệt kê tất cả các branch (nhánh).

Cách dùng: ***git branch*** hoặc ***git branch -a***

#### **9) git checkout**

Tác dụng: Chuyển sang branch khác

Cách dùng: ***git checkout <: branch:>*** hoặc ***\*\* \_ git checkout -b <: branch:>*** nếu bạn muốn tạo và chuyển sang một chi nhánh mới.

#### **10) git stash**

Tác dụng: Lưu thay đổi mà bạn không muốn commit ngay lập tức.

Cách dùng: ***git stash*** trong thư mục làm việc của bạn.

#### **11) git merge**

Tác dụng: Merge 2 branch lại với nahu.

Cách dùng: Chuyển tới branch bạn muốn merge rồi  dùng ***git merge <:branch\_ban\_muon\_merge:>***

#### **12) git reset**

Tác dụng: Bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo.

Cách dùng: ***git reset HEAD tên\_file***

#### **13) git remote**

Tác dụng: Để check remote/source bạn có hoặc add thêm remote

Cách dùng: ***git remote*** để kiểm tra và liệt kê. Và ***git remote add <: remote\_url:>*** để thêm.

#### **14) git add**

Tác dụng: Để đưa một tập tin vào Staging Area

Cách dùng: ***git add tên\_file*** hoặc muốn thêm hết file của thư mục thì ***git add all***

**5. Github là gì**

[GitHub](https://github.com/) là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn [Git](https://topdev.vn/blog/git-la-gi/) cho các dự án phần mềm. **Github có đầy đủ những tính năng của Git**, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

Vài thông tin về GIT:

* Là công cụ giúp **quản lý source code** tổ chức theo **dạng dữ liệu phân tán**.
* Giúp **đồng bộ source code** của team lên 1 server.
* Hỗ trợ các thao tác **kiểm tra source code** trong quá trình làm việc (diff, check modifications, show history, merge source, …)

GitHub có 2 phiên bản: miễn phí và trả phí. Với phiên bản có phí thường được các doanh nghiệp sử dụng để tăng khả năng quản lý team cũng như phân quyền bảo mật dự án.  
Còn lại thì phần lớn chúng ta đều sử dụng Github với tài khoản miễn phí để lưu trữ source code.

Github **cung cấp các tính năng social networking** như feeds, followers, và network graph để các developer học hỏi kinh nghiệm của nhau thông qua lịch sử commit.

Nếu một comment để mô tả và giải thích một đoạn code. Thì với Github, commit message chính là phần mô tả hành động mà bạn thực hiện trên source code.