**Họ và tên: Nguyễn Văn Thịnh**

**Nhóm: PHP**

**I. Khái niệm về GIT.**

Git là một trong những Hệ thống Quản lý Phiên bản Phân tán (Distributed Version Control System – DVCS), vốn được phát triển nhằm quản lý mã nguồn (source code) của Linux. DVCS nghĩa là hệ thống giúp mỗi máy tính 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 máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ 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. Trong Git, thư mục làm việc trên máy tính gọi là Working Tree.



Ngoài ra, có một cách hiểu khác về Git đơn giản hơn đó là nó sẽ giúp bạn lưu lại các phiên bản của những lần thay đổi vào mã nguồn và có thể dễ dàng khôi phục lại dễ dàng mà không cần copy lại mã nguồn rồi cất vào đâu đó. Và một người khác có thể xem các thay đổi của bạn ở từng phiên bản, họ cũng có thể đối chiếu các thay đổi của bạn rồi gộp phiên bản của bạn vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.

Cơ chế lưu trữ phiên bản của Git là nó sẽ tạo ra một “ảnh chụp” (snapshot) trên mỗi tập tin và thư mục sau khi commit, từ đó nó có thể cho phép bạn tái sử dụng lại một ảnh chụp nào đó mà bạn có thể hiểu đó là một phiên bản. Đây cũng chính là lợi thế của Git so với các DVCS khác khi nó không “lưu cứng” dữ liệu mà sẽ lưu với dạng snapshot.

Git mang tính toàn vẹn: mọi thứ trong git được “băm” trước khi lưu trữ và tham chiếu tới bằng mã băm đó. Có nghĩa là việc thay đổi nội dung của một tập tin hay một thư mục mà Git không biết tới là điều không thể. Chức năng này được xây dựng trong Git ở tầng thấp nhất về mặt triết học được coi là toàn vẹn. Bạn không thể mất thông tin/dữ liệu trong khi truyền tải hoặc nhận về một tập tin bị hỏng mà Git không phát hiện ra. Cơ chế mà Git sử dụng cho việc băm này được gọi là mã băm SHA-1.

Git chỉ thêm mới dữ liệu: khi bạn thực hiện các hành động trong Git, phần lớn tất cả hành động đó đều được thêm vào cơ sở dữ liệu của Git. Rất khó để yêu cầu hệ thống thực hiện một hành động nào đó mà không thể khôi phục lại được hoặc xoá dữ liệu đi dưới mọi hình thức. Giống như trong các VCS khác, bạn có thể mất hoặc làm rối tung dữ liệu mà bạn chưa commit; nhưng khi bạn đã commit thì rất khó để mất các dữ liệu đó, đặc biệt là nếu bạn thường xuyên đẩy (push) cơ sở dữ liệu sang một kho chứa khác.

Mỗi tập tin trong Git được quản lý dựa trên ba trạng thái: committed, modified và staged. Comitted có nghĩa là dữ liệu đã được lưu trữ một cách an toàn trong cơ sở dữ liệu. Modified có nghĩa là bạn đã thay đổi tập tin nhưng chưa commit vào cơ sở dữ liệu. Và staged là bạn đã đánh dấu sẽ commit phiên bản hiện tại của một tập tin đã chỉnh sửa trong lần commit sắp tới.

Tiến trình công việc (workflow) cơ bản của Git:

* Bạn thay đổi các tập tin trong thư mục làm việc.
* Bạn tổ chức các tập tin, tạo mới ảnh của các tập tin đó vào khu vực tổ chức.
* Bạn commit, ảnh của các tập tin trong khu vực tổ chức sẽ được lưu trữ vĩnh viễn vào thư mục Git.

**II. Các khái niệm**

***Github:***Được gọi là social network dành cho developer đi vào hoạt động tháng 2 năm 2008, là một dịch vụ sử dụng hệ thống quản lý phân tán GIT giúp người dùng lưu trữ source code cho các dự án.

***Repository:*** là nơi chứa tất cả những file, thông tin cần thiết để duy trì và quản lý các sửa đổi và lịch sử của toàn bộ project.

Có 2 loại repository:

* Remote repository: Là repository để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng.
* Local repository: Là repository bố trí trên máy của bản thân mình, dành cho một người dùng sử dụng.

***Nhánh (Branch):*** được dùng để phát triển tính năng mới mà không làm ảnh hưởng đến code hiện tại. Nhánh master là nhánh “mặc định” khi bạn tạo một repository. Nhánh master thông thường là nhánh chính của ứng dụng. Ví dụ bạn thử nghiệm một tính năng mới và muốn không ảnh hưởng đến code chính bạn có thể tạo một nhánh mới và sau khi xong sẽ hợp nhất lại với nhánh master. Việc hợp nhất 2 nhánh lại được gọi là merge.

***Trộn (Merge):*** Trộn source từ một nhánh khác vào nhánh hiện tại. Bạn sẽ hiểu sâu hơn về nó trong các ví dụ ở các bài sau.

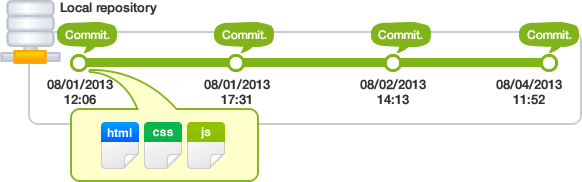
***Xung đột (Conflict):*** Conflic là trường hợp có 2 sự thay đổi trong một dòng code và máy tính không thể tự quyết định dòng code nào là đúng theo yêu cầu.

Để giải quyết mâu thuẫn bạn phải dùng “tay” để sữa các xung đột này. Bạn chỉ việc nhìn vào file bị conflict và tự quyết định dòng code nào giữ lại, dòng nào xóa bỏ.

***Commit:*** Để ghi lại việc thêm/thay đổi file hay thư mục vào repository thì sẽ thực hiện thao tác gọi là Commit.

Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) đã ghi lại sự khác biệt từ trạng thái đã commit lần trước với trạng thái hiện tại.

Commit này đang được chứa tại repository, các commit nối tiếp với nhau theo thứ tự thời gian. Bằng việc lần theo commit này từ trạng thái mới nhất thì có thể biết được lịch sử thay đổi trong quá khứ hoặc nội dung thay đổi đó.



***Working Tree và Index:*** Trên Git, những thư mục được đặt trong sự quản lý của Git mà mọi người đang thực hiện công việc trong thực tế được gọi là working tree.

Và trên Git, giữa repository và working tree tồn tại một nơi gọi là index. Index là nơi để chuẩn bị cho việc commit lên repository.

***III. Cách dùng Github:***

**3.1.Tạo Repository**

Có thể tạo một dự án có sử dụng Git dựa theo 2 phương pháp chính

* Khởi tạo một kho chứa từ thư mục có sẵn ở local:
  + Chỉ cần ở trong thư mục và gõ lệnh: $ git init
  + Lệnh này sẽ tạo một thư mục mới có tên .git, thư mục này chứa tất cả các tập tin cần thiết cho kho chứa.
* Sao chép một kho chứa đã tồn tại
  + Sử dụng lệnh git clone [url]để sao chép một kho chứa. Một thư mục sẽ được tạo, kèm theo thư mục .git và bản sao mới nhất của tất cả dữ liệu của kho chứa bên trong đó.

**3.2.Các lệnh cơ bản trong git**

* git status: kiểm tra trạng thái tệp tin.
* git add: để có thể theo dõi các tập tin mới tạo, hoặc thây đổi.
* git commit -m 'Ghi chú': ghi lại việc thêm/thay đổi file hay thư mục vào repository
* git log: xem lịch sử commit.
* git rm <tên file/thư mục>: Xóa tệp tin hoặc thư mục.
* git checkout -b <tên nhánh>: Tạo nhánh.
* git branch: xem dánh sách nhánh.
* git checkout <Tên nhánh>: chuyển sang nhánh khác.
* git checkout -d <Tên nhánh>: Xóa nhánh ở local.
* git clone
* git pull
* git push
* gt push origin :the\_remote\_branch: xóa nhánh ở remote.

**3.3. Qui trình làm việc git**

* Người quản lý dự án sẽ tạo Repository chứa Source code ban đầu được ở nhánh master.
* Tại nhánh master có thể có nhiều phiên bản cho từ mục đích khác nhau (ơ đây chúng ta tập trung vào phiên bản dành cho lập trình viên.
* Người quản lý dự án sẽ tạo nhánh tương ứng với từng công việc, cho từng lập trình viên hoặc lập trình viên có thể tự tạo cho mình (chưa rõ).
  + Lập trình viên sẽ clone dự án về, và sử dụng nhánh đã được tao. Sau khi làm việc trên nhánh, sẽ làm lần lượt các bướt
  + Kiểm tra trạng thái của các file đã thay đổi (git status).
  + Theo dõi các tệp tin đã thay đổi (git add <tên tệp tin hoặc \* để chọn tất cả>.
  + Ghi lại việc thay đổi (git commit -m'Ghi chú').
  + Đẩy sự dữ liệu thay đổi lên nhánh đang làm việc (git push origin <tên nhánh>).
  + Lên trang chủ của github và chọn dự án, chọn Pull Request để master được biết có sự thay đổi trong nhánh của developer đó.
* Người quản lý sau khi xem Request do developer tạo ra, sẽ review lại code. Nếu đồng ý sẽ merge code đã thay đổi vào master. Nếu không đồng ý, master sẽ yêu cầu developer code lại chỗ không đồng ý đó.

