**BÁO CÁO TÌM HIỂU GITHUB VÀ CÁC LỆNH GIT**

**1. Khái niệm Git**

- Git là tên gọi của một **Hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS)** là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. 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**.

- Git coi dữ liệu của nó giống như một tập hợp các “ảnh” (snapshot) của một hệ thống tập tin nhỏ. Mỗi lần bạn “commit”, hoặc lưu lại trạng thái hiện tại của dự án trong Git, về cơ bản Git “chụp một bức ảnh” ghi lại nội dung của tất cả các tập tin tại thời điểm đó và tạo ra một tham chiếu tới “ảnh” đó. Để hiệu quả hơn, nếu như tập tin không có sự thay đổi nào, Git không lưu trữ tập tin đó lại một lần nữa mà chỉ tạo một liên kết tới tập tin gốc đã tồn tại trước đó. Git thao tác với dữ liệu giống như hình sau:



- 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 toán 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.

**2. Khái niệm Github**

- Github là một dịch vụ máy chủ repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa riêng của mình để có thể làm việc.

**3. Sử dụng git**

**3.1 Tạo một kho chứa git**

- Có thể tạo một dự án có sử dụng Git dựa theo 2 phương pháp chính. Thứ nhất là dùng một dự án hay một thư mục đã có sẵn để nhập vào Git. Thứ hai là tạo bản sao của một kho chứa Git đang hoạt động trên một máy chủ khác.

**3.1.1 Khởi tạo một kho chứa từ thư mục cũ**

- Nếu muốn theo dõi một dự án cũ trong Git, chỉ cần ở trong thư mục của dự án đó 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.

**3.1.2 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.

- Ví dụ: *$ git clone git://github.com/schacon/grit.git mygrit*

**3.2 Kiểm tra trạng thái của tập tin**

- Công cụ chính để phát hiện trạng thái của tập tin là lệnh *git status*.

**3.3 Theo dõi các tập tin mới**

- Để có thể theo dõi các tập tin mới tạo, sử dụng lệnh *git add*.

- Ví dụ, để bắt đầu theo dõi tập tin README có thể chạy lệnh sau:

*$ git add README*

**3.4 Commit thay đổi**

*-* Sau khi đã tổ chức các tập tin theo ý muốn, có thể commit chúng.

- Có thể gõ trực tiếp thông điệp cùng với lệnh *commit* bằng cách thêm vào sau cờ *-m*, như sau:

*$ git commit -m “Đây là source code”*

- Commit lưu lại ảnh các tập tin mà mình chỉ định trong khu vực tổ chức. Bất kỳ tập tin nào không ở trong đó sẽ vẫn giữ nguyên trạng thái là đã sửa (modified); bạn có thể thực hiện một commit khác để thêm chúng vào lịch sử. Mỗi lần thực hiện commit là đang ghi lại ảnh của dự án mà bạn có thể dựa vào đó để so sánh hoặc khôi phục về sau này.

**3.5 Xóa tập tin**

- Dùng lệnh *git rm* để thực hiện việc xóa tập tin khỏi thư mục làm việc.

**3.6 Di chuyển tập tin**

- Dùng lệnh *git mv* *file\_from file\_to*

**3.7 Xem lịch sử commit**

- Sử dụng lệnh *git log*

**3.8 Phân nhánh trong git**

- Tạo một nhánh mới: *git checkout -b <tên nhánh>*

- Kiếm tra nhánh: *git branch*

- Chuyển từ nhánh này sang nhánh khác: *git checkout <tên nhánh>*

- Xóa nhánh: *git checkout -d <tên nhánh>*

**4. Quy trình làm việc git**

- Source code ban đầu và ổn định được lưu trữ ở nhánh master.

- Tại nhánh master, chia làm nhiều phiên bản khác nhau tùy theo mục đích của người sử dụng. Có phiên bản dành cho khách hàng, kiểm thử viên và lập trình viên. Ở đây ta tập trung vào việc sử dụng Git dành cho developer.

- Project Manager là người tạo ra master với source code ổn định ban đầu. Người này có trách nhiệm tạo mới và xóa các nhánh trong repository đó và merge lại các code vào trong master. Chịu trách nhiệm lớn nhất trong repository này.

- Các developer khi thực hiện công việc trên repository sẽ lần lượt thực hiện các công việc sau đây:

+ Sử dụng lệnh *git clone -b <tên nhánh> <đường dẫn để tải repository>* để tải repository này về máy cục bộ của lập trình viên.

+ Developer thực hiện việc code trên máy cục bộ của mình.

+ Sau khi code xong, developer sử dụng lệnh *git status* để kiểm tra trạng thái của các tập tin được thêm mới hay chỉnh sửa.

+ Sử dụng lệnh *git add <tên file tạo mới hay thay đổi>* để theo dõi các tập tin mới tạo hay thay đổi.

+ Sử dụng lệnh *git commit -m <comment/chú thích>*

*+* Sử dụng lệnh *git pull origin master* để lấy những thay đổi mới nhất từ nhánh master

+ Sử dụng lệnh *git push origin <tên nhánh>* để đẩy các tập tin mới nhất vào cơ sở dữ liệu của git.

+ Developer lên trang chủ của github và chọn **Pull Request** để master được biết có sự thay đổi trong nhánh của developer đó.

- Master 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.

- Ngoài ra, developer có thể lấy lại những thay đổi từ những lần commit trước đó.