Git và GitHub

I) Source Control

- Là chương trình để quản lý source code.
- Nó lưu trữ lịch sử thay dổi của code (code do ai viết, ai thay đổi, thay đổi phần nào,…)
- Hỗ trợ nhiều người làm việc cùng lúc.
- Có thể revert các thay đổi, đưa code về phiên bản cũ mà không lo mất code.

II) VCS

- 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 và thành một version (phiên bản).
- VCS giúp 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.

III) Git

a) Giới thiệu

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.

Một vài khái niệm của Git:

- **Repository**: Kho chứa mã nguồn (source code)
- **Commit**: Mỗi lần sửa code sẽ là 1 lần commit, nó sẽ được lưu vào repository để có thể truy ra ai là người thay đổi code, thay đổi cái gì,…
- **Branch**: Các **Branch** (nhánh) đại diện cho các **phiên bản cụ thể** của một kho lưu trữ tách ra từ project chính của bạn. Branch cho phép bạn theo dõi các thay đổi thử nghiệm bạn thực hiện đối với kho lưu trữ và có thể hoàn nguyên về các phiên bản cũ hơn.

- **Folk**: Folk là thao tác thực hiện sao chép repository của chủ sở hữu khác về git account của mình. sử dụng và đối xử như 1 repository do mình tạo ra.
- **Tag**: Sử dụng để đánh dấu một commit khi bạn có quá nhiều commit tới mức không thể kiểm soát được.
- **Remote**: Sử dụng để điều khiển các nhánh từ một repository trên git server, đối xử với các nhánh trên remote tương tự như đối xử với các nhánh trên local.
- **Diff**: So sánh sự sai khác giữa phiên bản hiện tại với phiên bản muốn so sánh, nó sẽ thể hiện các sự khác nhau
- .**gitignore**: file mặc định của git sử dụng để loại bỏ (ignore) các thư mục, file mà mình không muốn push lên git server

b) Hoạt động

- Về mặt khái niệm, hầu hết các hệ thống khác đều lưu trữ thông tin dưới dạng danh sách các thay đổi dựa trên file. Các hệ thống này (CVS, Subversion, Perforce, Bazaar, v.v.) coi thông tin chúng lưu giữ dưới dạng một tập hợp các file và những thay đổi được thực hiện đối với mỗi file theo thời gian.
- Git không nghĩ đến hoặc lưu trữ dữ liệu của mình theo cách này.
- Mỗi khi bạn "commit", Git sẽ "chụp" và tạo ra một snapshot cùng một tham chiếu tới snapshot đó. Để hiệu quả, nếu các tệp không thay đổi, Git sẽ không lưu trữ lại file — chỉ là một liên kết đến tệp giống file trước đó mà nó đã lưu trữ.

c) Một số lệnh git cơ bản

- **git init**: Khởi tạo 1 git repository 1 project mới hoặc đã có.
- **git clone**: Copy 1 git repository từ remote source.
- **git add**: Thêm thay đổi đến stage/index trong thư mục làm việc.
- **git commit**: 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.
- **git push/ pull**: 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.

d) Thao tác với Git

- **Git Cheet Sheets:** Bạn không thể nào nhớ được hết các lệnh, lúc này bạn nên sử dụng các Git Cheet Sheets để dễ dàng tìm được lệnh Git bạn cần.
- **Nên commit thương xuyên:** Tách nhỏ commit của bạn và commit thường xuyên nhất có thể. Điều này giúp các thành viên trong nhóm dễ dàng tích hợp công việc của họ hơn mà không gặp phải xung đột hợp nhất.
- **Test rồi mới commit:** Không bao giờ commit nếu chưa hoàn tất process. Cần phải test các thay đổi của bạn trước khi chia sẻ chúng với người khác.
- **Viết ghi chú khi commit:** Viết ghi chú khi commit để cho các thành viên khác trong nhóm biết loại thay đổi bạn đã thực hiện. Hãy mô tả càng nhiều càng tốt.
- **Thử nghiệm Branch khác:** Tận dụng lợi thế của các branch để giúp bạn theo dõi các dòng phát triển khác nhau.
- **Theo một Git Workflow:** Bạn nên chọn theo một Git Workflow để đảm bảo cả nhóm của bạn đều cùng thực hiện như nhau.

IV) Github

a) Khái niệm

- 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, …)
- **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.

b) Lợi ích

- Quản lý source code dễ dàng
- Tracking sự thay đổi qua các version
- Bạn có thể chỉnh sửa cách hiển thị của document, format từ như định dạng **in đậm** hay *in nghiêng*, thêm hình và tạo list những thứ bạn có thể làm với Markdown..
- Github giúp cải thiện kỹ năng code, thậm chí là tracking bug.

- Github là một kho tài nguyên tuyệt vời.

c) Làm việc với GitHub

- **Thao tác với repository ở local** với 2 command thường dùng là:
+ **Git add**: add file đã thay đổi vào stage.
+ **Git commit** các file đã add vào stage lên repository ở local Ngoài ra bạn xem một số command khác.
- **Làm việc với repository ở server GitHub** khi có một bản ổn định và hoàn tất ta sẽ quyết định cập nhật nó lên repository server với:
+ **push**: push thay đổi từ repository local lên repository server.
+ **fetch**: cập nhật thay đổi từ repository server về repository local.
+ **pull/rebase**: sao chép source code từ server về local workspace.

d) Hướng dẫn sử dụng GitHub

- Để sử dụng GitHub bạn cần:
+ Đăng ký một tài khoản GitHub và tạo một Repository (GitHub Repository).
+ Cài đặt GitHub Desktop, một công cụ trực quan quản lý Local Repository (Kho chứa dữ liệu địa phương).
+ Cấu hình để có thể đồng bộ hóa dữ liệu bằng GitHub Desktop lên Repository server.

- Bước 1: Bạn cần phải đăng ký miễn phí một tài khoản GitHub tại: https://github.com
- Bước 2: Đăng nhập và tạo một GitHub Repository
- Bước 3: Tải và cài đặt phần mềm GitHub Desktop (https://desktop.github.com/) để quản lý Local Repository trên máy tính cá nhân. Sau khi tải xong thì bạn mở GitHub Desktop lên và đăng nhập theo tài khoản / mật khẩu đã đăng ký ở Bước 1
- Bước 4: Liên kết tài khoản GitHub với phần mềm GitHub Desktop