GIỚI THIỆU VỀ GIT

**/\***

**Version Control system:** là một hệ thống lưu giữ các phiên bản của mã nguồn của sản phẩm phần mềm, giúp các lập trình viên có thể dễ dàng lấy lại phiên bản mong muốn. Có nhiều version control system khác như CSV, SVN, TeamVS,... Git chính là một trong những hệ thống quản lý phiên bản đó. \*/

**Git là gì ?**

* **Git** là [phần mềm quản lý mã nguồn](https://vi.wikipedia.org/w/index.php?title=Ph%E1%BA%A7n_m%E1%BB%81m_qu%E1%BA%A3n_l%C3%BD_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [phân tán](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_t%C3%A1n&action=edit&redlink=1) được phát triển bởi [Linus Torvalds](https://vi.wikipedia.org/wiki/Linus_Torvalds) vào năm [2005](https://vi.wikipedia.org/wiki/2005), ban đầu dành cho việc phát triển [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux). Hiện nay, Git trở thành một trong các phần mềm quản lý mã nguồn phổ biến nhất.
* Git là một trong những hệ thống kiểm soát phiên bản dùng để theo dõi các thay đổi trong tập tin, nó được dùng để thay đổi trong các tập tin bất kỳ và nó được sử dụng chính trong các lĩnh vực phát triển [kỹ thuật phần mềm](https://freelancervietnam.vn/ky-thuat-phan-mem-la-gi-sinh-vien-hoc-ky-thuat-phan-mem-se-lam-gi/).
* Git được hiểu là hệ thống điều khiển về mặt cơ bản nó là một trình theo dõi nội dung vì thế Git có thể sử dụng được để lưu trữ nội dung chủ yếu sử dụng để lưu trữ các mã do tính năng khác mà nó cung cấp.
* Git được coi là một hệ thống điều khiển phân tán có tốc độ xử lý nhanh những vẫn đảm bảo được toàn vẹn dữ liệu và hỗ trợ hiệu quả cao cho các workflow phân tán. Giống như các hệ thống quản lý phiên bản phân tán khác, mọi thư mục Git trên máy tính đều là một kho chứa lịch sử . Gồm có kho lưu trữ từ xa trong máy chủ và một kho lưu trữ cục bộ được lưu trữ trong máy tính của nhà phát triển.

## **Hệ thống kiểm soát Git hoạt động như thế nào ?**

* Trong thực tế làm việc thì các nhà phát triển thường thực hiện song song các dự án. Để tránh sự xung đột giữa các mã giữa các nhà phát triển ta cần một hệ thống kiểm soát. Để làm được điều này hầu hết các nhà phát triển chon Git là hệ thống kiểm soát các mã để tránh sự xung đột các mã của nhà phát triển.
* Bên cạnh đó, các yêu cầu trong các dự án thường xuyên được thay đổi. Có những khi bạn đã xóa một đoạn code nhưng lúc sau phát hiện ra đoạn code đó vẫn cần dùng. Thế nên cần có một kiểm soát để nhà phát triển có thể quay lại phiên bản cũ của mã. Chính vì vậy Git là một trong những lựa chọn để sử dụng. Vì Git giúp người dùng lưu lại những phiên bản của những lần thay đổi mã nguồn.
* Hầu như các hệ thống quản lý phiên bản tập chung tất cả mã nguồn và lịch sử thay đổi ở một chỗ máy chủ, thì với Git nó lại là một hệ thống quản lý phiên bản phân tán. Điều này giúp ích nhiều trong khi các dự án đang được chạy song song liên quan đến cùng một cơ sở mã.

**Các lợi ích lớn nhất trong việc dùng Git là gì ?**

**/\***

* Git dễ sử dụng, an toàn và nhanh chóng. Bạn có thể làm việc ở bất cứ đâu vì 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 \*/
* Sắp xếp công việc tốt hơn. Nghĩa là bạn có thể tập trung giải quyết từng task mà không cần bận tâm lo lắng cho những task liên quan.
* Linh hoạt hơn khi phải làm cùng lúc nhiều task, bởi vì bạn có thể cấu trúc công việc dễ dàng hơn.
* Tự tin thử nghiệm những ý tưởng mới, vì bạn có thể tách biệt việc thử nghiệm với dự án chính. Điều này giúp nâng cao chất lượng code cũng như tính sáng tạo.

/\*

**Các khái niệm cơ bản trong git:**

**epository (repo):** Nghĩa gốc là kho lưu trữ. Hiểu đơn giản, repo là các dự án bạn tạo ra, chứa mã nguồn phần mềm của bạn. Có 2 dạng: **Remote repo** (là repo chung được lưu trên server), và **Local repo** (là repo lưu trên máy của người dùng). Ta có thể làm việc và commit trên local reposity ngay cả trong điều kiện offline, khi có mạng chỉ việc đồng bộ lên remote repo để chia sẻ cho người dùng khác.  
  
**Commit:** là hành động xác nhận sự thay đổi của repo, lưu lại một trạng thái của repo. Khi thực hiện commit, trong repo sẽ tạo ra commit (hoặc revision) ghi lại sự khác biệt của tất cả các file trong repo từ trạng thái đã commit lần trước đó đến trạng thái hiện tại. Bằng việc xem các commit bạn có thể biết được file đã được sửa đổi như thế nào. Khi code một tính năng phải được hoàn thiện cơ bản và chạy được thìmớinêncommit.  
  
**Branch:** là các nhánh phát triển của repo. Ví dụ: bạn muốn thêm một tính năng mới cho phần mềm, bạn có thể tạo một branch khác để phát triển tiếp. Nếu sau này có muốn hủy bỏ tính năng đó thì cũng sẽ được thực hiện một cách dễ dàng mà không ảnh hưởng đến branch chính. Mỗi repo sẽ có một branch chính là master, các thành viên trong nhóm sẽ tạo các branch khác nhau để làm việc. Có nhiều cách chia branch khác nhau tùy mục đích và bạn cần phải học cách chia branch. Mỗi branch giống như một ngữ cảnh khác nhau, branch có thể được chia tách cũng nhưsátnhậpdễdàng.  
  
**Check out:** là hành động chuyển sang làm việc trên một branch khác. Trước khi chuyển branch thì bạn phải lưu lại trạng thái của branch hiện tại bằng cách commit.  
  
**Push:** là hành động upload một commit hoặc branch lên remote repo. Sau khi upload lên thì các thành viên của team có thể thấy và đồng bộ code xuống máy.  
  
**Pull:** là hành động download các thay đổi xuống local repo. Ví dụ: trong khi bạn đang code trên một file thì một người bạn trong nhóm của bạn cũng code trên một file khác cùng branch, người bạn đó hoàn thành công việc, commit và push lên remote repo. Lúc này bạn muốn lấy những thay đổi mà người bạn của bạn đã thực hiện thì bạn sẽ thực hiện hành động Pull xuống.  
  
**Clone:** là hành động tạo bản sao của remote repo từ máy chủ về máy mình để có thể lập trình và phát triển. Local repo được clone về cũng có lịch sử thay đổi giống hệt như remote repo. Vì vậy bạn có thể commit lên local repo này.  
  
**Merge:** là hành động hợp nhất một nhánh phát triển vào nhánh khác hoặc hợp nhất lịch sử thay đổi khi pull hoặc push. Ví dụ trường hợp merge branch: bạn phát triển xong 1 tính năng, đã test/ kiểm thử các kiểu và thấy nó hoàn chỉnh, có thể tích hợp vào phần mềm thì bạn sẽ tiến hành merge code. Sau khi merge có thể giữ lại 1 trong 2 branch hoặc cả 2. Yêu cầu trước khi merge: phải push hết các commit lên branch.  
  
**Conflict:** là trường hợp có nhiều sự thay đổi trong cùng 1 dòng code khi merge và máy không thể tự quyết định cái nào là đúng. Lúc này bạn phải tự quyết định giữ lại dòng code nào. Công việc xử lý conflict nên được thực hiện bằng GUI thay vì command-line.  
  
**Fork:** (khái niệm này trên GitHub) là hành động một người dùng khác copy một bản sao của repo về kho của họ. Trước khi tham gia vào một dự án của người khác thì bạn sẽ fork repo của họ và kho của mình nếu như người khác chưa cho phép bạn trở thành thành viên.  
  
**Pull request hay Merge request:** Khi người dùng khác tham gia phát triển phần mềm và đã phát triển xong một tính năng, họ muốn merge tính năng của họ vào phần mềm của bạn thì lúc này họ sẽ gửi một Pull request/Merge request để bạn chọn chấp nhận hay không. \*/