# Git master

1. Version Control là gì
   1. Version control là một hệ thống lắng nghe sự thay đổi của file hoặc một tập các files mọi lúc mọi nơi vì vậy ta có thể gọi các phiên bản của file. Với Version Control, ta có thể revert file về trạng thái trước đó, revert toàn bộ project về project trước đó, so sánh sự thay đổi mọi lúc, xem xem ai là người chỉnh sửa cuối cùng, ai tạo ra bug và khi nào và lúc nào. Nếu bạn làm hỏng hoặc mất dữ liệu thì có thể khôi phục lại được.
2. Local Version Control System : Nhiều người chọn phương pháp kiểm soát phiên bản của project là copy file sang thư mục khác (nếu thông minh thì thư mục đóng dấu theo thời gian). Cách tiếp cận này rất đơn giản và phổ biến tuy nhiên nó rất dễ xảy ra lỗi. Bạn quên mất bạn đang ở thư mục nào, hoặc là viết, chỉnh sửa sai file hoặc sao chép qua lại nhầm giữa các file. Để giải quyết vấn đề này, các lập trình viên đã phát triển VCS local nó có database đơn giản giữ tất cả các thay đổi của file dưới sự kiểm soát của revision control  
   - Một trong những VCS Tool phổ biến nhất là RCS, nó vẫn được sử dụng cho tới bây giờ
3. Centralized VCS (Quản lý vcs tập trung) (CVCS)  
   - Vấn đề đặt ra bây giờ là ta phải phát triển hệ thống cùng với các developer khác, Centralized Version Control System đã được tạo ra. Hệ thống này là 1 single server chứa tất cả các phiên bản của file. Tất cả các máy tính lấy file từ trung tâm đó.  
     
   - Thiết lập này mang nhiều lợi thế, đặc biệt là trên local VCS. Ở mức độ nhất định, Admin có quyền kiểm soát chi tiết xem ai có thể làm gì, việc quản lý CVCS dễ dàng hơn nhiều so với xử lý cơ sở dữ liệu local trên client.  
   - Tuy nhiên thiết lập này có một vài nhược điểm quan trọng. Điểm nổi bật nhất chính là máy chủ tập trung. Nếu nó bị chết trong vài giờ, không ai có thể lưu các thay đổi của các file dữ liệu mà họ đang làm việc. Hoặc nếu như ổ đĩa của con máy chủ bị hỏng, các bản sao lưu cần thiết không được lưu trữ thì ta mất hoàn toàn tất cả mọi thứ, may mắn thì có người nhanh chân sao lưu lại được dữ liệu trên máy cục bộ của họ.
4. Distributed Version Control System (DVCS)  
   - Đây là hệ thống kiểm soát phiên bản phân tán. Ở trong DVCS (giống như Git, Mercurial, Bazaar, Darcs), client không chỉ xem latest snapshot of the file, họ có thể xem kho lưu trữ và toàn bộ lịch sử của nó. Do đó nếu có bất kỳ server nào bị chết, bất kỳ client repository có thể copy dữ liệu để restore server mới. Mỗi bản sao là 1 bản sao lưu đầy đủ các dữ liệu.  
   
5. Git  
   - Hầu hết các hệ thống lưu thông tin dưới dạng file-based changes. Các hệ thống này nghĩ đến các thông tin chúng lưu trữ dưới dạng một tập hợp tệp và những thay đổi được thực hiện từng tệp theo thời gian  
     
   Git thì không làm như vậy. Trong thực tế, Git nghĩ dữ liệu của nó như một chuỗi các ảnh chụp nhanh của một hệ thống tệp thu nhỏ. Với Git, mỗi khi ta commit, hoặc lưu cái state của dự án, Git sẽ chụp ảnh tất cả các tệp của bạn trông như thế nào tại thời điểm đó và lưu trữ tham chiếu tới ảnh chụp nhanh đó. Để hiệu quả, nếu tệp không thay đổi, Git sẽ không lưu lại tệp, chỉ là một liên kết tới tệp giống hệt tệp trước đó mà nó đã được lưu trữ. Git nghĩ về dữ liệu của nó giống như một luồng các ảnh chụp nhanh  
     
   - Đây là điểm khác biệt quan trọng giữa Git và tất cả các VCS khác. Nó khiến Git xem xét lại hầu hết các khía cạnh của Version Control mà hầu hết các hệ thống copy từ phiên bản trước đó. Điều này làm cho Git giống như một hệ thống tệp nhỏ với một số công cụ cực kỳ mạnh mẽ được xây dựng lên, thay vì chỉ đơn giản là một VCS.
6. Mọi hoạt động với Git gần như đều là Local  
   - Hầu hết các hoạt động trong Git chỉ cần file Local và resources để hoạt động - không cần thông tin cần thiết từ một máy tính khác trên mạng của bạn. Toàn bộ lịch sử của dự án đều lưu trữ trên Local Disk nên nó gần như là tức thời. Ví dụ để mở lịch sử của dự án, Git không cần tới server lấy lại lịch sử và show cho bạn, nó chỉ cần lấy từ local database. Nếu bạn muốn so sánh sự thay đổi của 1 tệp với chính tệp đó ở thời điểm 1 tháng trước, git có thể truy vấn và tính toán sự thay đổi của tệp đó mà không cần phải kết nối đến server.
7. Tính toàn vẹn của Git (Git Integrity)  
   - Mọi thứ ở Git được checksummed trước khi nó được lưu trữ, sau đó được tham chiếu tới checksum đó. Điều đó có nghĩa là không thể thay đổi nội dung của bất kỳ tệp nào mà Git không biết về nó. Chức năng này được tích hợp vào Git ở cấp độ thấp nhất và không thể thiếu trong triết lý của nó. - - Cơ chế mà Git sử dụng để checksumming là SHA-1 hash. Ta có thể thấy hash value này ở khắp nơi trong Git vì nó sử dụng SHA rất nhiều. Trên thực tế, git lưu trữ thông tin file trên database không phải bằng file name mà nó là hash content of file.
8. Git thường chỉ thêm dữ liệu (Git generally only add data)  
   - Khi mà ta có một số hoạt động ở trong Git, gần như tất cả chúng chỉ Add data vào Database. Thật khó để khiến hệ thống làm điều gì mà nó không thể hoàn tác hoặc là xoá mất dữ liệu bằng bất kỳ cách nào. Đối với các VCS khác, bạn có thể mất hoặc xáo trộn các thay đổi mà mình chưa thực hiện. Nhưng nếu tạo 1 snapshot ở trong Git, nó rất khó bị mất đặc biệt nếu ta thường xuyên push database sang repository khác.
9. The three states  
   - Git có 3 trạng thái chính cho tệp là : modified, staged và committed.  
   + Modified được hiểu là ta đã thay đổi cái file (modified) nhưng mà chưa committed ở trong Local database.  
   + Staged được hiêu là ta đã đánh dấu cái file đã được thay đổi (modified) ở current version để đi tới next commit snapshot.  
   + Committed được hiểu là data đã được lưu an toàn ở trong Database  
   - Dự án git có 3 phần chính : Working tree, Staging area, Git directory  
     
   - Working Tree là 1 single checkout duy nhất của project. Các file này được kéo ra từ Compressed database (cơ sở dữ liệu nén ) ở trong Git directory và được đặt trên disk để ta sử dụng và sửa đổi.  
   - Staging area là 1 file, thông thường được chứa trong Git directory. Nó lưu trữ thông tin về việc gì sẽ làm tiếp theo cho lần next commit.  
   - Git directory là cái nơi mà Git lưu trữ metadata và object database trong dự án. Nó là phần quan trọng nhất của Git. Nó là những gì được sao chép khi mà ta clone repository từ một máy tính khác.  
   - Quy trình cơ bản của Git (Git workflow) thường diễn ra ntn:   
   + Bạn thay đổi file ở trong working tree  
   + Bạn chỉ chọn lọc những thay đổi mà bạn muốn nó trở thành một phần của next commit. Nó thêm những thay đổi đó vào Staging Area.  
   + Bạn thực hiện commit, Nó lấy các tệp ở trong Staging Area và lưu vĩnh viễn cái snapshot của file đó ở Git directory.  
   - Nếu như một phiên bản cụ thể của tệp nằm trong Git directory, nó được gọi là committed. Nếu nó được thay đổi và được lưu ở Staging Area, nó gọi là Staged. Và nếu như nó được thay đổi từ khi nó được checked out nhưng mà chưa được staged thì gọi là modified
10. Lần đầu cài đặt Git  
    - [git config] : get và set tất cả các biến cấu hình kiểm soát tất cả các khía cạnh về cách Git look và các hoạt động của nó. Các biến này được lưu ở 3 nơi lưu trữ khác nhau đó là:   
    + [/path]/etc/gitconfig (ở windows là : C:\Program Files\Git\etc\gitconfig): Nó có chứa giá trị config có thể áp dụng cho bất kỳ user nào trong hệ thống và tất cả các repository của họ. Nếu thêm tuỳ chọn --system theo sau git config, nó đọc và ghi từ tệp này. Bởi vì sử dụng System configuration file, bạn cần phải là admin hoặc superuser để có quyền thay đổi nó.  
    + [~/.gitconfig hoặc ~/.gitconfig/git/config] (ở windows là C:\User\{$USER}\.gitconfig) file: Giá trị config chỉ dành riêng cho bạn, cho user đó. Bạn có thể làm Git đọc và ghi vào file đó bằng cách thêm --global option. Config này ảnh hưởng tới tất cả các repository của bạn trên hệ thống.  
    + config file ở trong Git directory (./git/config) : cấu hình này chỉ dành riêng cho 1 repository mà bạnd dang sử dụng, đọc và ghi vào tệp này bằng cách thêm tuỳ chọn --local.   
    - Ta có thể xem tất cả các config bằng cách dùng lệnh : git config --list --show-origin
11. Your Identity  
    - Điều đầu tiên cần làm là set name và email của user của bạn. Vì thông tin này là thông tin của user login, vì vậy ta phải cấu hình ở tuỳ chọn global:   
    + git config --global user.name “John Doe”  
    + git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)
12. Your Editor  
    - git config --global core.editor emacs
13. Your default branch name  
    - Mặc định git tạo 1 branch được gọi là master khi mà tạo mới một repository với lệnh git init, tuy nhiên ta có thể đổi cái initial branch là branch khác.  
    - git config --global init.defaultBranch main
14. Checking your Settings  
    - git config --list  
    Để xem giá trị của config là gì, ta sử dụng lệnh :   
    - git config (--global / --system / --local) user.name . Nếu muốn config đó được cấu hình ở file nào, ta sử dụng thêm tuỳ chọn --show-origin
15. Git Basic