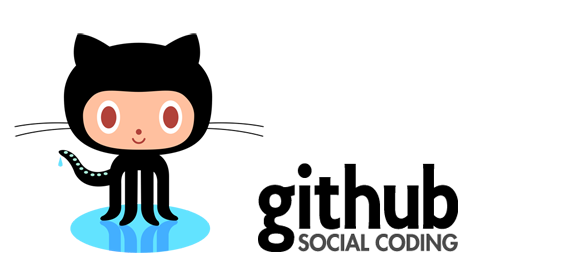
**Git và Github**

# 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.

Git vốn được phát triển nhằm quản lý mã nguồn (source code) của Linux, người tạo ra Git là Linus Torvalds cũng chính là cha đẻ của nhân 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

Trên Git, ta có thể lưu trạng thái của file dưới dạng lịch sử cập nhật. Vì thế, có thể đưa file đã chỉnh sửa một lần về trạng thái cũ hay có thể biết được file đã được chỉnh sửa chỗ nào.



Thêm nữa, khi định ghi đè (overwrite) lên file mới nhất đã chỉnh sửa của người khác bằng file đã chỉnh sửa dựa trên file cũ, thì khi upload lên server sẽ hiện ra cảnh cáo. Vì thế, sẽ không xảy ra lỗi khi ghi đè lên nội dung chỉnh sửa của người khác mà không hề hay biết.

Git sử dụng mô hình phân tán, ngược lại so với SVN hoặc CSV. Mỗi nơi lưu source sẽ đc gọi là repositories, không cần lưu trữ tập trung một nơi, mà mỗi thành viên trong team sẽ có một repository ở máy của riêng mình.

Điều đó có nghĩa là nếu có 3 người A,B,C cùng làm việc trong 1 project. Thì bản thân repo trên máy của người A, người B, và người C có thể kết nối được với nhau.

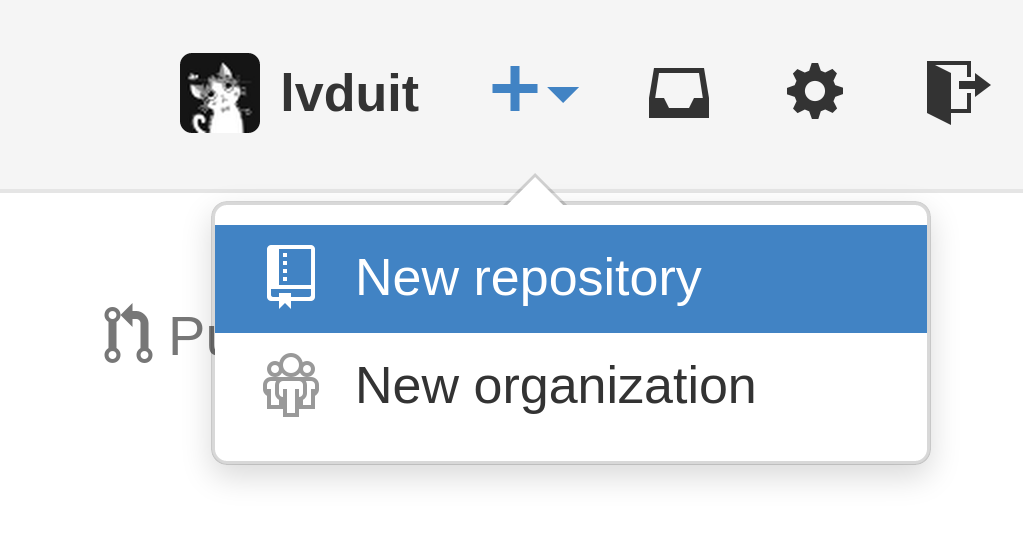
Khi quyết định thay đổi chỗ nào đó lên server ta chỉ cần một thao tác "push" nó lên server. Chúng ta vẫn có thể share thay đổi của chúng ta cho thành viên khác, bằng cách commit hoặc update trực tiếp từ máy của họ mà không phải thông qua repositories gốc trên server (thông qua share ssh cho nhau).

Lợi ích

* An toàn hơn (vì mỗi bản copy của thành viên đều là full copy từ repository gốc, khi server bị down).
* Các thành viên vẫn có thể làm việc offline, họ vẫn có thể commit và update trên local của họ hoặc thậm chí với nhau mà không cần thông qua server.
* Khi server hoạt động trở lại, họ có thể cập nhật tất cả lên lại server.

## Kho Repo (Repository)

Repository hay được gọi tắt là Repo, đơn giản là nơi chứa tất cả những thông tin cần thiết để duy trì và quản lý các sửa đổi và lịch sử của toàn bộ project. Trong Repo có 2 cấu trúc dữ liệu chính là Object Store và Index. Tất cả dữ liệu của Repo đèu được chứa trong thư mục bạn đang làm việc dưới dạng folder ẩn có tên là .git



## Remote repository và local repository

Đầu tiên, repository của Git được phân thành 2 loại là remote repository và local 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.

Do repository phân thành 2 loại là local và remote nên với những công việc bình thường thì có thể sử dụng local repository. Khi muốn public nội dung công việc mà mình đã làm trên local repository, thì ta sẽ upload lên remote repository rồi public. Thêm nữa, thông qua remote repository bạn cũng có thể lấy về nội dung thay đổi của người khác.



## Nhánh (Branch)

Đây là một trong những thế mạnh của git là nhánh. Với git, việc quản lý nhánh rất dễ dàng. Mỗi nhánh trong Git gần giống như một workspace. Việc nhảy vào một nhánh để làm việc trong đó tương tự việc chuyển qua ngữ cảnh làm việc mới, và sau đó có thể nhanh chóng quay lại ngữ cảnh cũ.

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.  
Chú ý:

* Kiểm tra branch hiện đang làm việc trước khi merge
* Phải đẩy tất cả những thay đổi dưới máy local lên Git trước khi merge
* Trước khi merge phải lấy hết những thay đổi mới nhất của các branch khác, hay ít nhất là branch cần merge về máy
* Merge thành công thì nên đẩy source lên lại lên server
* Nên merge bằng GUI tool.

## Xung đột (Conflict)

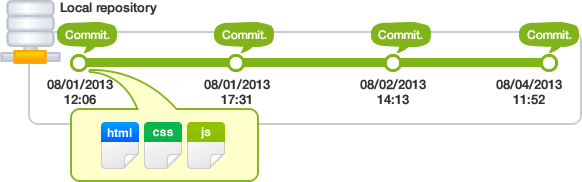
Conflic là trường hợp có 2 sự thay đổi trong một dòng code và máy tính không thể tự quyết định dòng code nào là “đúng”. Đúng ở đây có nghĩa là “ý đồ của lập trình viên”.

Để giải quyết mâu thuẫn bạn phải dùng “tay không” 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.



Các commit này, được đặt tên bởi 40 ký tự alphabet (mã md5 thì phải) không trùng nhau được băm từ thông tin commit. Bằng việc đặt tên cho commit, có thể chỉ định commit nào từ trong repository.

Mỗi commit đều có yêu cầu phải có commit message, để giải thích commit này là bạn đã làm gì trong này.

## Git Remote

Để kết nối được với một repo khác người ta sử dụng một khái niệm gọi là remote.

Trên thực tế khi làm việc với nhau thì không như vậy, vì không phải máy ai cũng cài một “git server” để người khác kết nối được với mình. Thông thường thì chúng ta sẽ sử dụng một repo chung và các máy kết nối vào repo đó.

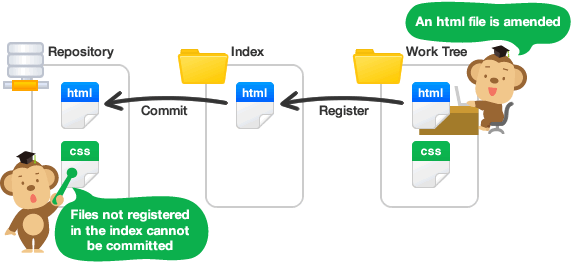
Có 2 “git repo server” được sử dụng nhiều là github.com và bitbucket.org.

Trên thực tế khi có 2 người cùng làm việc với 1 project thì thông thường sẽ tạo một repo trên github hoặc bitbucket và repo trên máy người A sẽ kết nối với repo trên github và máy người B cũng kết nối với repo trên github/bitbucket. Từ đó source code của người A và người B sẽ được đồng bộ với nhau thông qua repo trên github/bitbucket.

## Working Tree và Index

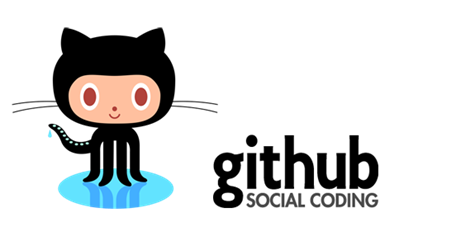
Trên Git, những thư mục được đặt trong sự quản lý của Git mà mọi người đang thực hiện công việc trong thực tế được gọi là working tree.

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.



Trên Git, khi đã thực hiện commit thì trạng thái sẽ không được ghi trực tiếp trong repository từ working tree, mà sẽ ghi trạng thái đã được thiết lập của index được xây dựng ở giữa đó. Vì thế, để ghi lại trạng thái của file bằng commit thì trước hết cần thông báo file trong index.

# Github

GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn Git dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án mã nguồn mở sẽ được cung cấp kho lưu trữ miễn phí. Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.

Github đã trở thành một yếu có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

Linh vật của Github là chú mèo Octocat.

## Dịch vụ

Sự phát triển của GitHub bắt đầu vào ngày 19 tháng 10 năm 2007. Trang web được đưa ra vào tháng tư năm 2008 do Tom Preston-Werner, Chris Wanstrath, và PJ Hyett thực hiện sau khi nó đã được hoàn thành một vài tháng trước đó, xem như giai đoạn beta.

Dự án trên Github có thể được truy cập và thao tác sử dụng một giao diện dòng lệnh và làm việc với tất cả các lệnh Git tiêu chuẩn. Github cũng cho phép người dùng đăng ký và không đăng ký để duyệt kho công cộng trên trang web. Github cũng tạo ra nhiều client và plugin cho máy tính để bàn.

Trang web cung cấp các chức năng mạng xã hội như feed, theo dõi, wiki (sử dụng phần mềm Gollum Wiki) và đồ thị mạng xã hội để hiển thị cách các nhà phát triển làm việc trên kho lưu trữ.

Một người sử dụng phải tạo ra một tài khoản cá nhân để đóng góp nội dung lên Github, nhưng các kho mã nguồn công cộng có thể được duyệt và tải về với bất cứ ai. Với một người dùng đã đăng ký tài khoản, họ có thể thảo luận, quản lý, tạo ra các kho, đóng góp cho kho của người dùng khác, và xem xét thay đổi mã.

GitHub cũng có một dịch vụ khác: một trang web kiểu pastebin gọi là Gist, dùng để lưu trữ các đoạn mã; trong khi Github sẽ được cho lưu trữ các dự án lớn hơn. Một dịch vụ lưu trữ khác được gọi là Speaker Deck.

Các phần mềm chạy GitHub được viết bằng Ruby on Rails và Erlang bởi GitHub Inc, phát triển bởi Chris Wanstrath, PJ Hyett, và Tom Preston-Werner.

Hệ thống kiểm soát phiên bản (VCS- Concurrent Versions System)- ghi lại thay đổi vào một tập tin hoặc thiết lập các tập tin theo thời gian để có thể nhớ lại các phiên bản cụ thể sau.

## Phạm vi

GitHub chủ yếu được sử dụng để lưu trữ mã nguồn phần mềm, nhưng cũng thường được sử dụng với nhiều loại tập tin như Final Cut hoặc các tài liệu Word.

Github cũng có thể được dùng như một pluggin để các thành viên trên diễn đàn có thể trao đổi thông tin với nhau.

## Thao tác lệnh trong Git-Github

Khởi tạo một kho local.

Kho local là nơi git sẽ đồng độ tất cả dữ liệu trên mây với máy tính local, trên windows, ta có thể thực hiện việc xác định kho local bằng thao tác chuột phải trên folder và chọn “Gitgui here” , nếu dùng lệnh, ta chọn “git bash here” nếu ta có kiến thức dùng lệnh linux, câu lệnh sẽ là:

git init

Clone repository về máy bằng lệnh sau:

git clone “link kho chủ”

“link kho chủ” chính là đường dẫn mà người đứng đầu project được cung cấp khi tạo một project, nó có nhiệm vụ liên kết tất cả các client lại với nhau.

Thiết lập thông tin cá nhân:

$ git config –global user.name “your\_name”  
$ git config –global user.email “your\_email@youremail.com”

Thiết lập upstream cho repository bằng lệnh sau:

git remote add upstream “link kho chủ”

Tạo các thay đổi ở local

Để đơn giản bạn có thể tạo 1 file đặt tên là readme.txt với nội dung là testGitHub như sau:

echo “# testGitHub” >> “tên file”

Thêm file mới tạo

git add “tên file”

Kiểm tra sự thay đổi trước khi commit

git status

kết quả như sau:

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

 new file:   “tên file”

commit các thay đổi, log message là bắt buộc, nó chứa lý do một client thay đổi file tại một thời điểm

git commit -m 'log message'

đẩy các thay đổi từ local lên github như sau:

git push ‘origin master’

Sau đó ta vào trang github kiểm tra sẽ thấy file vừa tạo.

Vì github được phát triển dựa trên hđh mã nguồn mở linux nên hầu hết các câu lệnh của linux đều có thể áp dụng cho git.