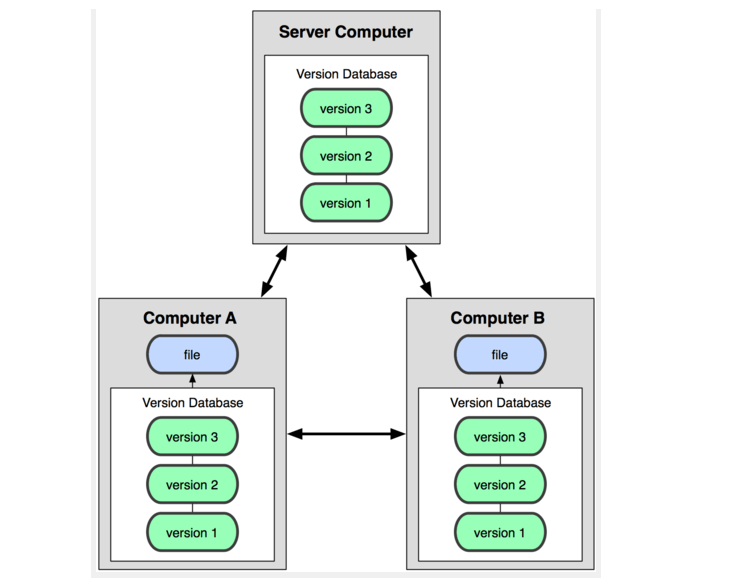
Git : Git là tên gọi của một Hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (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



Mo hinh hoat dong

Ngoài ra, có một cách hiểu khác về Git đơn giản hơn đó là nó sẽ giúp bạn lưu lại các phiên bản của những lần thay đổi vào mã nguồn và có thể dễ dàng khôi phục lại dễ dàng mà không cần copy lại mã nguồn rồi cất vào đâu đó. Và một người khác có thể xem các thay đổi của bạn ở từng phiên bản, họ cũng có thể đối chiếu các thay đổi của bạn rồi gộp phiên bản của bạn vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.

Github: Git là tên gọi của một mô hình hệ thống. Như đã giải thích ở phần git, các máy tính có thể clone lại mã nguồn từ một repository và Github chính là một dịch vụ máy chủ repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc.

Cài đặt Git

Trên linux:

Ubuntu/Fedora : dùng lệnh “ sudo apt-get install git “

Centos: dùng lệnh “ yum install git “

Trên Mac-os: http://git-scm.com/download/mac để cài đặt

Trên window: <http://git-scm.com/download/win>

( Demo)

Kiểm tra git đã được cài đặt chưa : dùng lệnh “ git -v “ ( kiểm tra version của git nêu không hiện version thì git chưa được cài đặt)

(demo)

Cấu hình ban đầu:

Khai báo username và usermail:

git config --global user.name "git"

git config --global user.email [git@github.coml](mailto:git@github.coml)

Xem thông tin vừa tạo : cat ~/.gitconfig

Hoặc dùng lệnh : git config --list

(demo)

Cai dat ssh key : giup moil an push source len server se khong phai danh password

Cau hinh:

Kiem tra ssh key da co chua: *ls -al ~/.ssh*

Tao key ssh: *ssh-keygen -t rsa -b 4096 -C* [your\_email@example.com](mailto:your_email@example.com)

### Adding your SSH key to the ssh-agent

*eval $(ssh-agent -s)*

*ssh-add ~/.ssh/id\_rsa*

*Ssh-agent tạo một socket và kiểm tra các kết nối từ ssh. Tất cả mọi người có thể kết nối với ổ cắm này cũng có quyền truy cập vào ssh-agent. Các điều khoản được thiết lập như trong một hệ thống Linux hoặc Unix thông thường. Khi tác nhân bắt đầu, nó tạo ra một thư mục mới trong / tmp với các điều khoản hạn chế. Ổ cắm nằm trong thư mục.*

*Có một thủ tục có thể ngăn phần mềm độc hại sử dụng ổ cắm ssh-agent. Nếu tùy chọn ssh-add -c được thiết lập khi các phím được nhập khẩu vào ssh-agent thì agent yêu cầu một xác nhận từ người dùng sử dụng chương trình được xác định bởi biến môi trường SSH\_ASKPASS, bất cứ khi nào ssh cố gắng kết nối.*

*Trên hệ thống cục bộ, điều quan trọng là người dùng gốc là đáng tin cậy, bởi vì người sử dụng gốc có thể, trong số những thứ khác, chỉ cần đọc các tập tin quan trọng trực tiếp. Trên hệ thống từ xa, nếu kết nối ssh-agent được chuyển tiếp, điều quan trọng là người dùng gốc là đáng tin cậy, bởi vì họ có thể truy cập vào ổ cắm agent (mặc dù không phải là khoá).*

**Add key ssh len server github**

Coppy key : *clip < ~/.ssh/id\_rsa.pub hoac mo file : cat ~/.ssh/id\_rsa.pub coppy key*

*Sau do add key vao user tren github*

*Kiểm tra :* ssh -T git@github.com

**Repository**

Repository (kho chứa) nghĩa là nơi mà bạn sẽ lưu trữ mã nguồn và một người khác có thể sao chép (clone) lại mã nguồn đó nhằm làm việc. Repository có hai loại là Local Repository (Kho chứa trên máy cá nhân) và Remote Repository (Kho chứa trên một máy chủ từ xa)

( Demo)

Tao tren local su dung lenh : git init “ten thu muc”

Sao chep clone 1 Repository noi bo: git clone /đường-dẫn-đến/repository/

Tao tren github dung tren web

Clone ve may : https://github.com/username/repository

Commit

Staging Area nghĩa là một khu vực mà nó sẽ được chuẩn bị cho quá trình commit. Trước hết, bạn cần phải hiểu rằng trong các hệ thống quản lý phiên bản (Version Control System) thì các dữ liệu sẽ được lưu trữ ở hai nơi, một là thư mục bạn đang làm việc trên máy tính (working tree) và một là kho chứa mã nguồn (repository) sau khi bạn đã thực hiện thay đổi (ví dụ như kho chứa trên Github).

Nhưng với Git thì nó có thêm một lựa chọn nữa đó là có thêm một khu vực trung gian gọi là Staging Area và đây chính là một lợi thế lớn của Git. Staging Area nghĩa là khu vực sẽ lưu trữ những thay đổi của bạn trên tập tin để nó có thể được commit, vì muốn commit tập tin nào thì tập tin đó phải nằm trong Staging Area. Một tập tin khi nằm trong Staging Area sẽ có trạng thái là Stagged

Commit là gì và nó hoạt động ra sao?

Hiểu đơn giản hơn, commit nghĩa là một hành động để Git lưu lại một bản chụp (snapshot) của các sự thay đổi trong thư mục làm việc, và các tập tin và thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của mã nguồn kèm theo tên và địa chỉ email của người commit. Ngoài ra trong Git bạn cũng có thể khôi phục lại tập tin trong lịch sử commit của nó để chia cho một phân nhánh (branch) khác, đây là mấu chốt của việc bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó mà mình có giới thiệu qua ở phần giới thiệu serie này.

Và tất nhiên, lệnh commit trong Git sẽ là *git commit -m "Lời nhắn".*

Và nếu bạn muốn đưa tập tin lên repository thì bạn phải commit nó trước rồi sau đó lệnh *git push origin master* sẽ có nhiệm vụ đưa toàn bộ các tập tin đã được commit lên repository.

Lenh commit : git commit -m “ tin nhan”

Hoac git commit -a -m “ tin nhan “ ( tu dong cap nhat cac tap tin da chinh sua va xoa va khong anh huong den nhung file ma git khong de cap den)

**Sua Commit**

**Nếu bạn cần xóa bỏ lần commit trước và cần undo để commit lại thì có thể sử dụng tham số --amend trong lệnh git commit.**

**$ git log --pretty="%s"**

**Hihi**

**First commit on Github**

**$ git commit --amend -m "Hehe"**

**[master 3682e56] Hehe**

**Date: Tue Apr 21 17:16:37 2015 -0700**

**2 files changed, 1 insertion(+), 1 deletion(-)**

**create mode 100644 faq.html**

**$ git log --pretty="%s"**

**Hehe**

**First commit on Github**

# Remote Respository

Github là một máy chủ repository từ xa nên mình sẽ gọi nó là Remote Repository, nghĩa là repository này không nằm trên máy tính của bạn.

Và ở phần đó, bạn có thấy mình kêu các bạn gửi dữ liệu lên repository bằng cách dùng lệnh *git push origin master* sau khi commit không? Cái master là tên branch mà mình sẽ không nói ở bài này, nhưng cái origin trong đoạn đó chính là tên remote repository. Mặc định khi clone một repository thì nó tự đặt tên là origin.

Để kiểm tra tên remote, bạn có thể gõ lệnh git remote -v.

*git remote -v*

*origin https://github.com/thachphamblog/hoc-git.git (fetch)*

*origin https://github.com/thachphamblog/hoc-git.git (push)*

Doi ten Origin

*git remote rename tên\_cũ tên\_mới*

*git remote rename origin thach*

*$ git remote -v*

*thach https://github.com/thachphamblog/hoc-git.git (fetch)*

*thach https://github.com/thachphamblog/hoc-git.git (push)*

Thêm một remote

*git remote add unuit https://github.com/thachpham92/inuit.css-web-template*

*git remote -v*

Trường hợp bạn cần thêm một cái remote để lấy dữ liệu khi cần thì có thể sử dụng lệnh git remote add tên\_remote URL

Các loại giao thức của Remote Repository

Chúng ta không chỉ kết nối với một remote repository qua giao thức HTTP hay HTTPS mà còn có thể chọn nhiều giao thức khác, dưới đây là một vài giao thức remote repository.

Local Repository

Giao thức này nghĩa là bạn kết nối tới một repository nào đó trên chính máy tính của bạn và URL của giao thức sẽ có dạng /path/repository/.

HTTP Repository

Giao thức thông dụng nhất cũng như dễ hiểu nhất, thường được sử dụng nếu bạn dùng các dịch vụ remote repository như Github hay Assembla, nó sẽ bao gồm định dạng http://domain.com/repository.git hoặc https://domain.com/repository.git.

SSH Repository

Giao thức này thường được dùng trên các nhu cầu tạo một server repository riêng và kết nối thông qua giao thức SSH. Đường dẫn của giao thức này sẽ có dạng là user@server:/path/repository.git. Ở phần cuối serie mình sẽ hướng dẫn bạn cách tạo một repository server riêng và nó là lựa chọn rất tốt nếu bạn làm việc nhóm mà không cần phụ thuộc vào các dịch vụ như Github hay Assembla.

Ngoai github con co cac dich vu repository mien phi nhu: Bitbucket, Assembla

### Branch trong Git là gì?

Khi bắt đầu khởi tạo một repository hoặc clone một repository, bạn sẽ có một nhánh (branch) chính tên là master . Đây là branch mà sẽ chứa toàn bộ các mã nguồn chính trong repository.

Đó là lý do tại sao, ở các phần trước khi chúng ta push hoặc pull hay làm một số việc khác thì lại có tham số master trong câu lệnh, đó nghĩa là chúng ta đang thực hiện thao tác trên branch master.

Bây giờ nếu bạn muốn tạo một sự thay đổi nào đó mà có thể trực tiếp sử dụng trên working tree hoặc commit, push lên repository mà không ảnh hưởng đến branch master thì sẽ cần tạo ra một branch mới với tên là develop chẳng hạn. Và từ đó mỗi khi bạn thực hiện lệnh checkout vào branch nào đó thì toàn bộ mã nguồn trên working tree của bạn sẽ được đổi sang môi trường dành cho branch đang checkout.

Giờ mình lấy một ví dụ đơn giản, bên branch master bạn tạo một tập tin master.html rồi commit lên, lúc này khi bạn qua cái branch develop (đã được tạo trước đó) thì cái tập tin master.html kia sẽ không có vì nó đã được commit bởi branchmaster, không liên quan gì tớidevelop. Tương tự, các thay đổi của bạn bên develop cũng sẽ không ảnh hưởng gì tới bên master cả. Điều này có một cái thú vị là nó tương tác trực tiếp trên máy tính của bạn, ví dụ tập tin index.html của master có nội dung khác và index.html bên develop thì khi bạn dùng lệnh checkout là nội dung trên máy tính nó tự đổi tương ứng, đó là lý do bạn có thể test nhiều phiên bản trên máy tính mà không cần đổi thư mục, chỉ cần checkout cái branch.

Tao: git branch < ten nhanh >

Xoa: git branch -d <ten nhanh>

Chuyen: git checkout < ten nhanh >

Tao va chuyen: git branch -b <ten nhanh>

Các lệnh GIT cơ bản

git init Tạo một repository mới

git clone Sao chép (clone) một repository

Để clone 1 repository có sẵn ở trên máy cục bộ, bạn hãy sử dụng dòng lệnh sau:

git clone /đường-dẫn-đến/repository/

Nếu repository đó ở máy chủ khác thì bạn hãy gõ dòng lệnh sau:

git clone tênusername@địachỉmáychủ:/đường-dẫn-đến/repository

(vi du: git clone <https://github.com/NghiaJava/Seminar>)

Git remote -v kiem tra remote repository

git branch Kiểm tra nhánh hiện tại

git branch <Tên nhánh> Tao nhanh moi

git branch -d <Tên nhánh> Xoa nhanh

git checkout Chuyen nhanh

cu phap : git checkout < ten nhanh >

git checkout -b < nhanh moi > tao nhanh moi va chuyen luon sang nhanh moi do

git branch -d < ten nhanh > Xoa di 1 nhanh

git commit Xac nhan va luu lai nhung thay doi truoc khi up len server

cu phap : git commit -a -m “ ghi chu “

git push Cập nhật những thay đổi của project và gửi lên server

cu phap : git push origin < ten nhanh >

( neu dang lam viec tren nhanh thi chi can “git push”)

Nếu bạn chưa clone một repository hiện có và muốn kết nối repository của bạn đến máy chủ remote, bạn phải thêm nó với : git remote add origin <máy-chủ>

git fetch tk 1 dang o nhanh master, tk2 tao nhanh develop. Khi tk 1 thuc hien lenh “git branch” thi tren tk 1 se khong thay dc nhanh develop. Tk1 Muon thay dc nhanh develop thi phai thuc hien lenh git “ fetch origin”

git pull Lấy source mới nhất trên server về local

cu phap : git pull origin <ten nhanh > hoac git pull

git merge trộn một nhánh khác vào nhánh đang hoạt động

cu phap : git merge < ten nhanh > chi merge duoc mot lan lan sau muon merge nua thi phai dung lai lenh git merge

git log xem lai tat c noi dung da commit

git log -p hien thi chi tiet moi lan commit

git status kiem tra xem tinh trang chinh sua cac file

Ưu điển khi dùng git/github

Git dễ sử dụng, an toàn và nhanh chóng.

Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc kết hợp các phân nhánh (branch).

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.

Dễ dàng trong việc deployment sản phẩm

Nhược điểm:

Bản miễn phí bạn chỉ được dùng với quyền public nên project của bản luôn luôn sẵn có ở tren mạng. nếu muốn an toàn hơn thì bạn phải mua bản tính phí.