**TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG**

***KHOA ĐIỆN TỬ - TIN HỌC***

***BỘ MÔN TIN HỌC***

**~~~¥~~~**



***Báo Cáo Môn:*** **Công Cụ & Môi Trường Phát Triển Phần Mềm**

***Chủ đề:***  **Tìm Hiểu Công Cụ Quản lí Mã nguồn Github**

***Giảng Viên:*** TÔ LONG PHƯỚC

***Sinh Viên Thực Hiện:***

Nguyễn Văn Khương - 0306171362

Liêu Trần Hiếu Hoài - 0306171345

Nguyễn Công Hậu - 0306171341

Trần Quốc Đạt - 0306171337

Hồ Tấn Đạt - 0306171336

***Lớp:***CDTH17PMA

***Ngày Lập Báo Cáo:***27-09-2019

TÌM HIỂU CÔNG CỤ QUẢN LÍ MÃ NGUỒN GITHUB

MỤC LỤC

[*1. KHÁI NIỆM GIT VÀ CÁCH CÀI ĐẶT 3*](#_Toc20654918)

[2.1 Git là gì? 3](#_Toc20654919)

[1.1 GITHUB là gì 3](#_Toc20654920)

[1.2 Cài Git vào Linux: 4](#_Toc20654921)

[1.3 Cài Git vào Mac OS: 4](#_Toc20654922)

[1.4 Cài Git vào Windows: 4](#_Toc20654923)

[1.5 Download & cài đặt GitHub Desktop 5](#_Toc20654924)

[*2. GIT SỬ DỤNG KHI NÀO 8*](#_Toc20654925)

[2.2 Git được sử dụng ở đâu 8](#_Toc20654926)

[2.3 Khi Tắt Kết Nối Internet có sử dụng được không? 8](#_Toc20654927)

[*3. CÁC MÃ GIT THÔNG DỤNG 9*](#_Toc20654928)

[3.1 Mã git Thiết lập chứng thực cá nhân 9](#_Toc20654929)

[3.2 Tạo một kho chứa Git 9](#_Toc20654930)

[3.3 Sao chép một kho chứa đã tồn tại 9](#_Toc20654931)

[3.4 Nhánh trong git 9](#_Toc20654932)

[3.5 Để tạo mới một branch: 10](#_Toc20654933)

[3.6 Để chuyển và tạo mới: 10](#_Toc20654934)

[3.7 Chuyển nhánh 10](#_Toc20654935)

[3.8 Cập nhật thay đổi 10](#_Toc20654936)

[3.9 Cập nhật lên server 10](#_Toc20654937)

[3.10 Gộp nhánh 11](#_Toc20654938)

[3.11 Xem lại lịch sử commit 11](#_Toc20654939)

[3.12 Xem thay đổi trước khi push 11](#_Toc20654940)

[3.13 Gộp commit 11](#_Toc20654941)

[3.14 Pull từ remote repository 11](#_Toc20654942)

[*4. CÁCH TẠO MỘT REPOSITORY 12*](#_Toc20654943)

[4.1 Git 12](#_Toc20654944)

[4.2 Mô hình biểu diễn Remote Repository: 12](#_Toc20654945)

[4.3 Các bước để tạo repository: 13](#_Toc20654946)

[4.4 Github 14](#_Toc20654947)

[4.5 Push data lên Github 17](#_Toc20654948)

[4.6 Cập nhật, thay đổi mã nguồn 18](#_Toc20654949)

[4.7 Source git khi làm việc với nhiều người thì như thế nào? 19](#_Toc20654950)

[4.8 Lưu lại lịch sử thay đổi source Git 21](#_Toc20654951)

[4.9 Xem lịch sử commit: 22](#_Toc20654952)

[4.10 Clone 1 source về desktop 27](#_Toc20654953)

[*5. TỔNG KẾT 28*](#_Toc20654954)

[*6. TÀI LIỆU THAM KHẢO 28*](#_Toc20654955)

GIỚI THIỆU

Bạn là lập trình viên, và đôi khi bạn muốn đưa về trạng thái trước khi quậy phá của file code nào đó?

Cách đơn giản nhất đó là sao chép lại file trước khi chỉnh sửa. Trường hợp dùng phương pháp này thì sẽ phải thường xuyên thực hiện việc thêm ngày đã thay đổi vào tên thư mục hay file. Tuy nhiên, việc tự mình sao chép file mỗi lần chỉnh sửa thì sẽ rất vất vả, và cũng dễ xảy ra nhầm lẫn. Và để giải quyết những vấn đề này thì các hệ thống quản lý phiên bản như Git đã được ra đời và với máy chủ lưu trữ là github

~~~¥~~~

1. KHÁI NIỆM GIT VÀ CÁCH CÀI ĐẶT
2. Git là gì?

Git là một trong những Hệ thống Quản lý Phiên bản Phân tán, vốn được phát triển nhằm quản lý mã nguồn (source code) của Linux.

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.

Để có thể sử dụng được Git, bạn phải cài ứng dụng Git vào máy tính để có thể sử dụng các dòng lệnh của Git vì toàn bộ quy trình làm việc với Git đều diễn ra các dòng lệnh.

1. GitHub là gì

**GitHub** là sự kết hợp giữa 2 từ, Git – hệ thống quản lý dự án và phiên bản code và Hub – một mạng xã hội cho lập trình viên.

GitHub được sử dụng chủ yếu cho dự án có nhiều người cùng hợp tác và cần giám sát toàn bộ thay đổi của dự án, cũng như để ngõ khả năng khôi phục code khi cần thiết. Khi sử dụng GitHub, ngoài các công việc chính như tạo Branch, tạo Pull Request và Fork một Repository, bạn có thể theo dõi, tương tác với người khác như một mạng xã hội thông thường.

1. Cài Git vào Linux:

Nếu bạn đang sử dụng hệ điều hành Ubuntu/Debian thì có thể sử dụng lệnh sau để cài Git.

“$ sudo apt-get install git”

Hoặc lệnh sau để cài trên CentOS/Fedora/RHEL.

“ $ yum install git”

1. Cài Git vào Mac OS:

Đối với Mac, bạn có thể sử dụng file installer tải tại địa chỉ:

[http://git-scm.com/download/mac](https://git-scm.com/download/mac)

1. Cài Git vào Windows:

Nếu bạn dùng Windows thì có thể tải file .exe cài đặt Git tại địa chỉ

“[http://git-scm.com/download/win](https://git-scm.com/download/win)”

Khi cài bạn có thể để nguyên tùy chọn mặc định mà không cần tùy chỉnh gì thêm nếu bạn chưa hiểu về nó.

Sau khi cài đặt Git vào Windows, bạn sẽ cần mở ứng dụng Git Bash lên để bắt đầu sử dụng các dòng lệnh của Git.

Sau khi cài Git xong, việc đầu tiên bạn nên làm là khai báo tên và địa chỉ email vào trong file cấu hình của Git trên máy. Để làm điều này bạn sẽ cần sử dụng hai lệnh sau đây để thiết lập tên và email.

“$ git config --global user.name "Hau"”

“ $ git config --global user.email [contact@Hau.com](mailto:contact@Hau.com)”

Sau khi thiết lập xong, bạn có thể kiểm tra thông tin chứng thực trên user của bạn bằng cách xem tập tin *~/.gitconfig* (nhắc lại rằng dấu ~ nghĩa là thư mục gốc của user).

“ $ cat ~/.gitconfig [user] name = Thach Pham email = [contact@thachpham.com](mailto:contact@thachpham.com)”

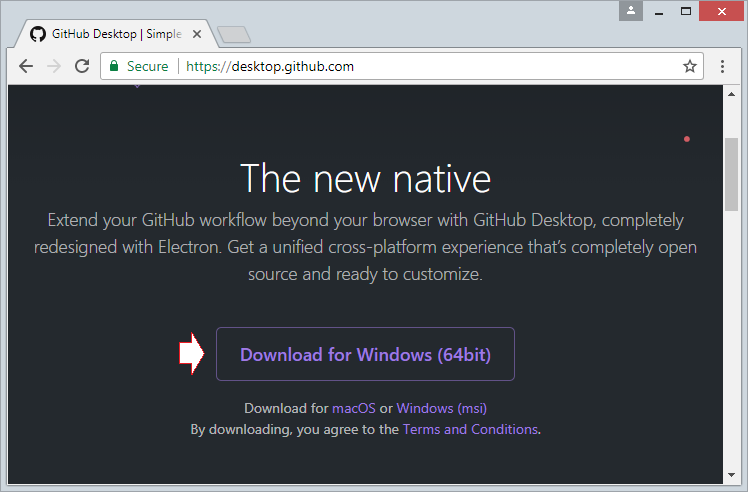
Hoặc bạn cũng có thể dùng lệnh git config --list để ghi danh sách các thiết lập hiện tại mà bạn đã làm.

Như vậy là bạn đã xong bước ban đầu đó là cài đặt Git và thiết lập tên và email của mình vào Git để bắt đầu làm việc.

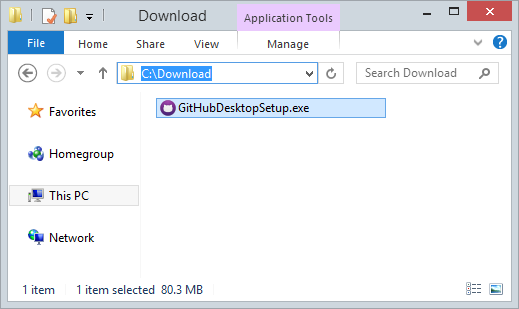
1. Download & cài đặt GitHub Desktop

**GitHub Desktop** về bản chất là một công cụ trực quan cho phép bạn quản lý **Local Repository** (Kho chứa địa phương) trên máy tính của bạn.

<https://desktop.github.com/>

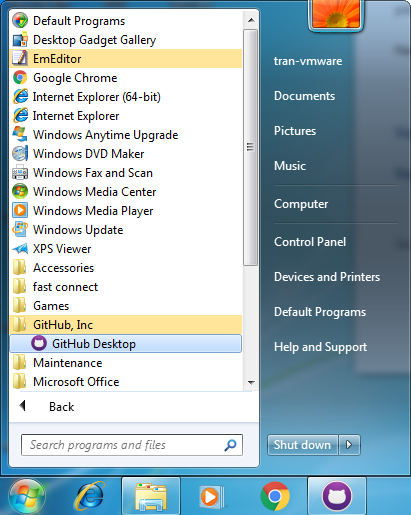


Kết quả download:



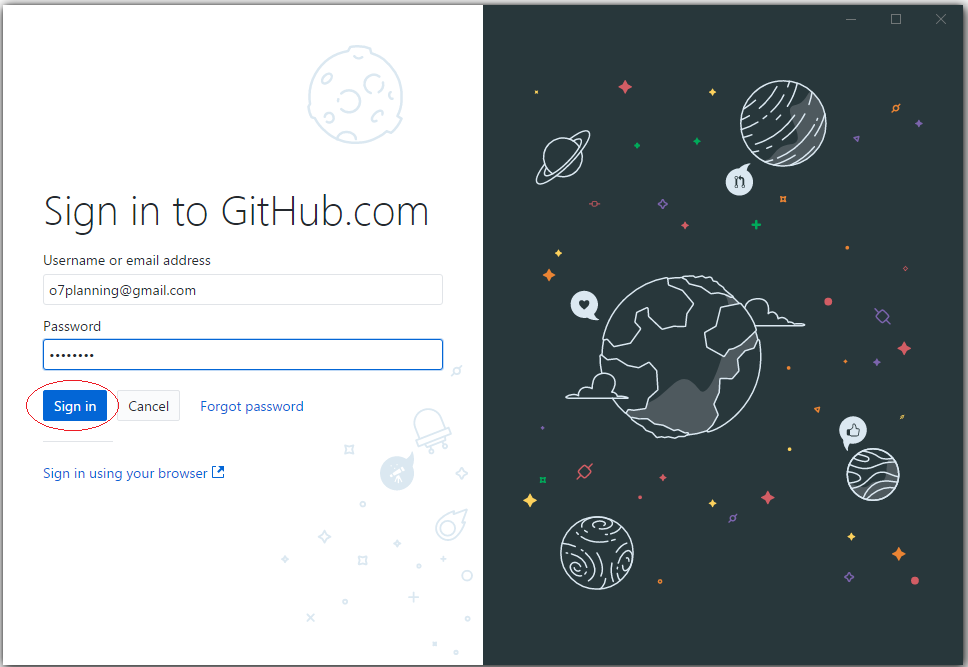
Sau khi download xong, bạn cần cài đặt GitHub Desktop vào máy tính:

GitHub Desktop đã được cài đặt thành công.

****

***Chạy GitHub Desktop***

Đăng nhập trên **GitHub Desktop** để kết nối vào tài khoản GitHub của bạn.



Thành Công Như bên dưới:



~~~¥~~~

1. GIT SỬ DỤNG KHI NÀO
2. Git được sử dụng ở đâu

*Phần lớn thao tác diễn ra ở cục bộ.* Các thao tác/hoạt động trong Git chỉ cần yêu cầu các tập tin hay tài nguyên cục bộ thông thường nó sẽ không cần bất cứ thông tin từ máy tính nào khác trong mạng lưới của bạn.

Nếu như bạn quen với việc sử dụng các hệ thống quản lý phiên bản tập trung nơi mà đa số hoạt động đều chịu sự ảnh hưởng bởi độ trễ của mạng, thì với Git đó lại là một thế mạnh.

Bởi vì toàn bộ dự án hoàn toàn nằm trên ổ cứng của bạn, các thao tác được thực hiện gần nhưngay lập tức.

*Ví dụ:*

Khi bạn muốn xem lịch sử của dự án, Git không cần phải lấy thông tin đó từ một máy chủ khác để hiển thị, mà đơn giản nó được đọc trực tiếp từ chính cơ sở dữ liệu cục bộ của bạn.

Điều này có nghĩa là bạn có thể xem được lịch sử thay đổi của dự án gần như ngay lập tức.

Nếu như bạn muốn so sánh sự thay đổi giữa phiên bản hiện tại của một tập tin với phiên bản của một tháng trước, Git có thể tìm kiếm tập tin cũ đó trên máy cục bộ rồi sau đó so sánh sự khác biệt cho bạn. Thay vì việc phải truy vấn từ xa hoặc "kéo về" (pull) phiên bản cũ của tập tin đó từ máy chủ trung tâm rồi mới thực hiện so sánh cục bộ. Điều này còn đồng nghĩa với có rất ít việc mà bạn không thể làm được khi không có kết nối.

1. Khi Tắt Kết Nối Internet có sử dụng được không?

Internet hoặc VPN bị ngắt. Nếu bạn muốn làm việc ngay cả khi ở trên máy bay hoặc trên tầu, bạn vẫn có thể commit bình thường cho tới khi có kết nối Internet để đồng bộ hoá.

Nếu bạn đang ở nhà mà VPN lại không thể kết nối được, bạn cũng vẫn có thể làm việc bình thường.

Trong rất nhiều hệ thống khác, việc này gần như là không thể hoặc rất khó khăn.

Ví dụ:

Trong Perforce, bạn gần như không thể làm gì nếu như không kết nối được tới máy chủ; trong Subversion và CVS, bạn có thể sửa tập tin nhưng bạn không thể commit các thay đổi đó vào cơ sở dữ liệu (vì cơ sở dữ liệu của bạn không được kết nối).

Đây có thể không phải là điều gì đó lớn lao, nhưng bạn sẽ ngạc nhiên về sự thay đổi lớn mà nó có thể làm được.

~~~¥~~~

1. CÁC MÃ GIT THÔNG DỤNG
2. Mã git Thiết lập chứng thực cá nhân

“$ git config --global user.name "User Name"

$ git config --global user.email “[username@gmail.com](mailto:username@gmail.com)” ”

Lưu ý:

--global được sử dụng để áp dụng cho tất cả các projects. Nếu bạn ko sử dụng

--global thì settings sẽ chỉ dùng cho riêng project đó.

1. Tạo một kho chứa Git

Nếu như bạn muốn theo dõi một dự án cũ trong Git, bạn cần ở trong thư mục của dự án đó.

“$ git init”

Lệnh này sẽ tạo một thư mục mới có tên .git, thư mục này chứa tất cả các tập tin cần thiết

cho kho chứa.

1. Sao chép một kho chứa đã tồn tại

“$ git clone <https://github.com/user/repository.git>”

Câu lệnh trên sẽ tạo một thư mục mới có tên giống trên của repo.

1. Nhánh trong git

Khi sử dụng Git, bạn có thể tạo ra nhiều nhánh (branch) khác nhau. Câu lệnh Git này

“$ git branch”

dùng để kiểm tra branch hiện tại:

1. Để tạo mới một branch:

“$ git branch <name\_branch>”

1. Để chuyển và tạo mới:

“$ git branch -b <name\_branch>”

1. Chuyển nhánh

Trước khi muốn thay đổi source code, điều đầu tiên mà bạn cần phải làm là checkout

một nhánh. Để checkout một nhánh, bạn dùng câu lệnh Git sau:

“$ git checkout <name\_branch>”

1. Cập nhật thay đổi

Sau khi bạn thay đổi source code: thêm mới, sửa, xoá files… Bạn cần phải cập nhật lên

Staging Area. Để cập nhật hết các files:

“$ git add”

Sau lệnh add, bạn cần sử dụng câu lệnh Commit để đây thông tin thay đổi lên Local Respository:

“$ git commit -m "Message"”

1. Cập nhật lên server

Sau câu lệnh Commit, thông tin mới chỉ được cập nhật lên Local Repository. Nếu muốn cập nhật lên server thì bạn phải sử dụng câu lệnh push:

“$ git push origin <name\_branch>”

Ngoài ra, nếu chưa tồn tại remote trên server thì bạn cần phải add mới một remote trước rồi mới push:

“$ git remote add origin <remote\_url>

$ git push origin <name\_branch>”

1. Gộp nhánh

Sau một thời gian cập nhật các file và push lên git trên branch mới, bây giờ mình cần ghép

(merge) code lại vào nhánh gốc (master). Trước tiên, cần phải checkout ra khỏi branch hiện

tại cần gộp để vào branch master, sau đó thì dùng lệnh merge để ghép branch mới vào master:

“$ git checkout master

$ git merge <new\_branch>”

1. Xem lại lịch sử commit

“$ git log”

Lệnh git log sẽ cho bạn biết về người commit, ngày giờ, message của những lần commit đó.

1. Xem thay đổi trước khi push

“$ git diff”

Lệnh này giúp bạn biết những gì đã được thay đổi giữa nhánh hiện tại và nhánh trước nó.

1. Gộp commit

“$ git rebase -i HEAD~”

Sau dấu ~ là số commit bạn muốn gộp. Sau khi gõ lệnh này một cửa sổ trình soạn thảo hiện ra. Thay đổi ký tự pick của dòng các dòng sau dòng đầu thành s rồi lưu lại/kết thúc. Khi đó, trình soạn thảo để chỉnh sửa giải thích commit thiết lập cho commit sau khi đã tổng hợp sẽ được hiển thị, nên hãy chỉnh sửa lưu lại/kết thúc.

1. Pull từ remote repository

“$ git pull origin master”

Lệnh trên sẽ gộp những thay đổi mới kéo về từ máy chủ từ xa với nhánh hiện tại trên máy local.

~~~¥~~~

1. CÁCH TẠO MỘT REPOSITORY
2. Git

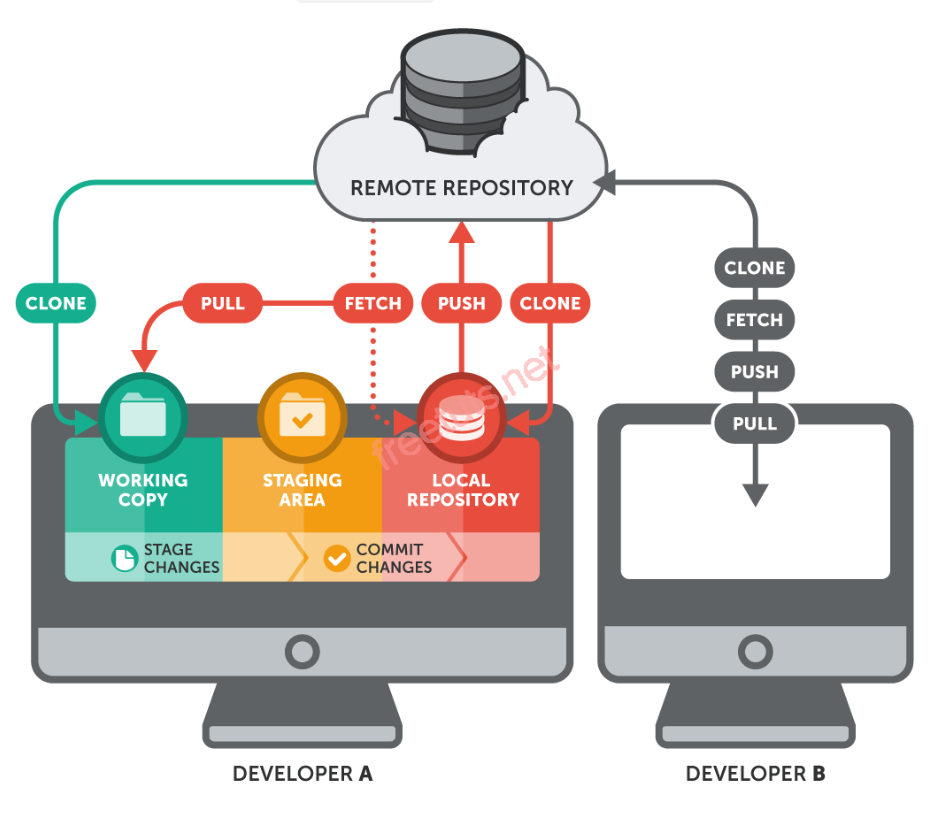
Repository (kho chứa): nơi ta lưu trữ mã nguồn và những người khác có thể sao chép lại mã nguồn đó nhằm phục vụ cho công việc.

Repository có 2 loại: Local Repository (kho chứa trên máy cá nhân) và Remote Repository (kho chứa trên một máy chủ từ xa).

Local Repository: là repository được cài đặt trên máy tính lập trình viên, repository này sẽ đồng bộ hóa với remote repository bằng các lệnh của git.

Remote Repository: là repository được cài đặt server chuyên dụng, điển hình hiện nay là Github.

1. Mô hình biểu diễn Remote Repository:



1. Các bước để tạo repository:

**Bước 1**: Truy cập đến folder của dự án.

**Bước 2**: Sử dụng lệnh “***git init***” để git tự động thực hiện thiết lập repository. Nếu như nhận được thông báo “***Initialized empty Git repository in …*** ” thì đã tạo Repository thành công.

Ví dụ:

“Initialized empty Git repository in D:/abc/.git/”.

Ngoài ra ta có thể kết hợp cả 2 bước trên như sau: Sử dụng lệnh:

“git init” + [tên folder]

để nó tự khởi tạo thư mục.

**Chú ý ở ví dụ trên:** “.git/” là nơi nó sẽ chứa các thiệt lập về Git cũng như lưu lại toàn bộ thông tin về kho chứa, chỉ cần không đụng chạm gì vào thư mục “.git/” này.

Trường hợp kho chứa của bạn đã có sẵn mã nguồn thì bạn chỉ cần đưa các tập tin về trạng thái Tracked.

Nhằm có thể làm việc được với Git. Để làm được việc này, ta sử dụng lệnh

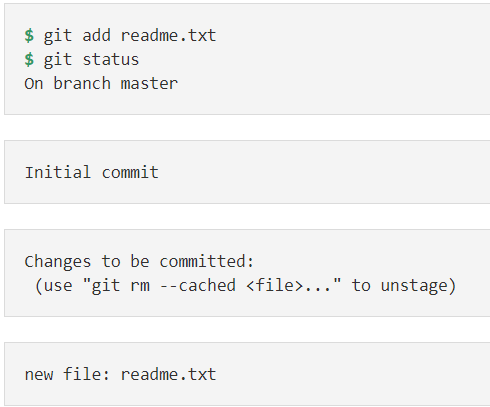
“git add + tên\_file”

có thể sử dụng dấu “ **\*** ” để gom toàn bộ. Sau đó dụng lệnh

“git status”

để xem danh sách các tập tin đã được tracked.

Ví Dụ:

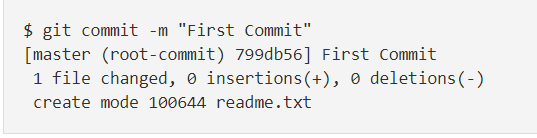


Khi tập tin đã được đưa vào trạng thái tracked và nếu một tập tin đã tracked thì nó phải được đưa vào lại Staging Area (giải thích ở bài sau) cũng bằng lệnh ***“git add”*** thì bạn mới có thể tiến hành ủy thác (**commit**) nhằm lưu lại bản chụp các thay đổi. Lệnh commit sẽ có cấu

“git commit –m “Lời nhắn” ”

lúc này tất cả các tập đang trong trạng thái tracked (file mới) hoặc một tập tin đã được tracked nhưng có một sự thay đổi mới thì sẽ được commit.

Ví Dụ:

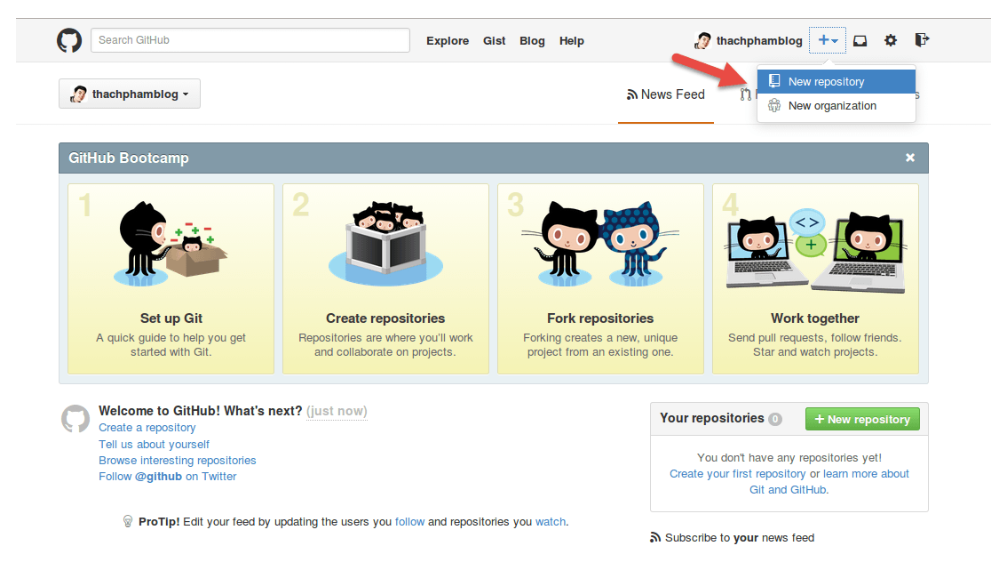


1. Github

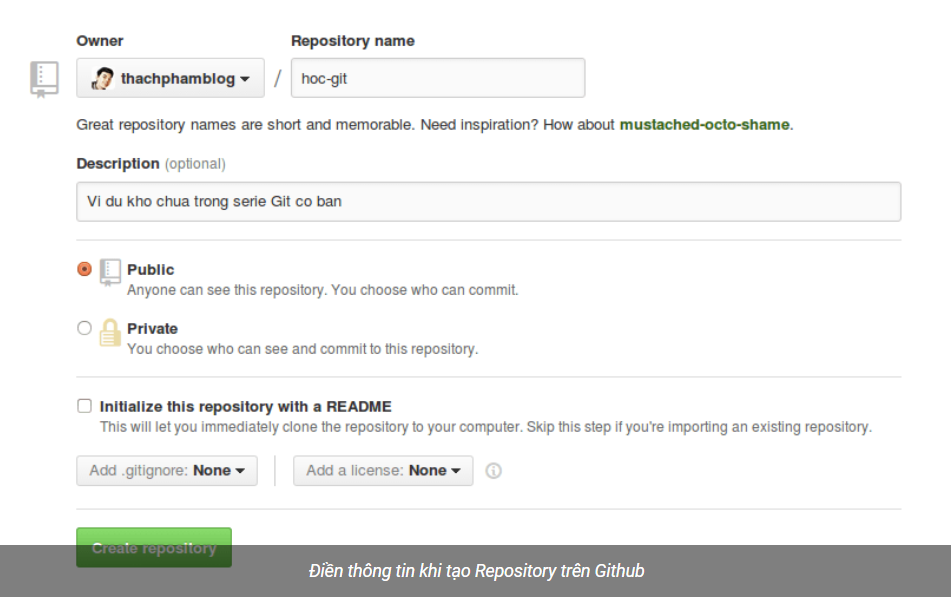
Để tạo một repository trên github trước hết ta cần phải đăng nhập vào trang

“ <https://github.com/> ”

Sau đó ấn chọn dấu “**+**” trên menu và chọn “**New repository**”.

****

Bạn sẽ cần đặt tên cho kho chứa của bạn. Bạn có thể chọn loại kho chứa là **Public** (ai cũng có thể clone) và **Private** (chỉ có những người được cấp quyền mới có thể clone).

****

Khi tạo xong nó sẽ dẫn bạn tới trang hướng dẫn làm việc với kho chứa vừa tạo. Và kho chứa của bạn bây giờ sẽ có địa chỉ là:

<https://github.com/$user-name/$repository>

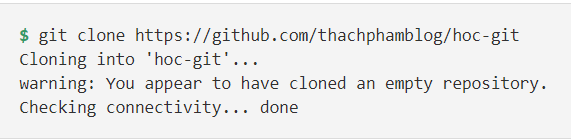
ví dụ:

<https://github.com/vidu/hoc-git.>

Việc của bạn bây giờ là hãy clone cái kho chứa này về máy của mình bằng lệnh:

“git clone địa\_chỉ”

Ví dụ:

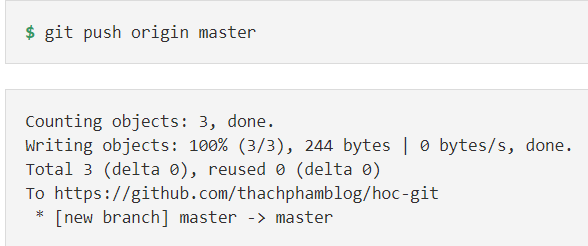


Bây giờ hãy truy cập vào thư mục working tree (thư mục vừa clone repository về) và thử tạo ra một file tên là README.md, sau đó dùng lệnh **git add** để đưa file này vào Staging Area.

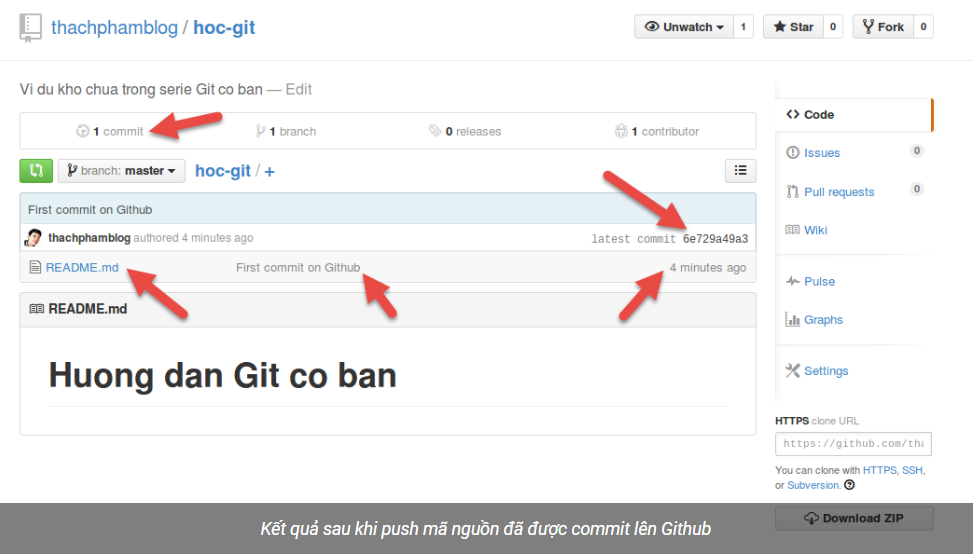
****

Tuy nhiên sau khi commit xong, tập tin đã được commit sẽ vẫn không thể xuất hiện trong kho chứa trên Github mà bạn phải làm thêm một việc nữa đó là dùng lệnh **git push** để đẩy các tập tin đã được commit lên Github.

Lưu ý rằng **bạn sẽ cần nhập tài khoản và mật khẩu Github**.

****

**Origin** nghĩa là tên remote và **master** là tên branch, hai cái này mình sẽ giải thích kỹ hơn ở bài riêng của nó. Bây giờ bạn có thể kiểm tra kho chứa của bạn trên Github rồi đó.



Có thể bạn sẽ thấy mỗi khi push lên Github thì nó hiển thị ra thông báo quá dài dòng, bạn có thể cho ẩn các thông báo đi mà chỉ hiển thị dòng khai báo username và password trên Github thì có thể thiết lập với lệnh này:

“git config --global push.default simple”.

Tuy nhiên nếu bạn không thích bị hỏi mật khẩu nữa thì có thể sử dụng SSH với Github.

1. Push data lên Github

Có 2 trường hợp khi ta Push data lên github:

1/ Trường hợp branch mới tạo:

Nếu bạn mới tạo một branch tại local thì lúc này trên github sẽ không tồn tại branch đó, vì vậy thao tác đầu tiên thay vì sử dụng lênh push thì bạn sẽ sử dụng thêm một tham số kèm theo đó là:

“--set-upstream”.

Ví Dụ:

$ git push --set-upstream abc congacon

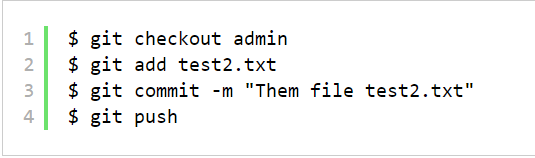
(ở đây **abc**: là *username*, **congacon**: là tên *branch*)

*2/ Trường hợp branch cũ:*

Trường hợp này bạn chỉ cần chạy lệnh:

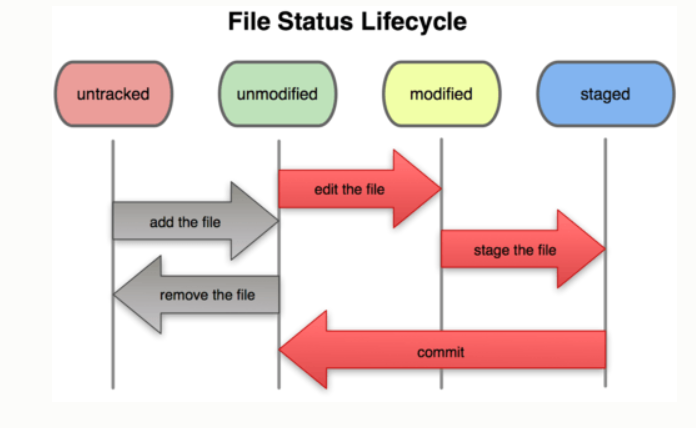
“git push”

Sau đó nhập thông tin đăng nhập nếu bị yêu cầu.



1. Cập nhật, thay đổi mã nguồn

Khi bạn chỉnh sửa các tập tin, Git coi là chúng đã bị thay đổi so với lần commit trước đó. Bạn **stage** các tập tin bị thay đổi này và sau đó commit tất cả các thay đổi đã được staged (tổ chức) đó, và quá trình này cứ thế lặp đi lặp lại như được miêu tả trong Hình



Để cập nhật repository cục bộ của bạn và commit mới nhất, thực thi:

“git pull”

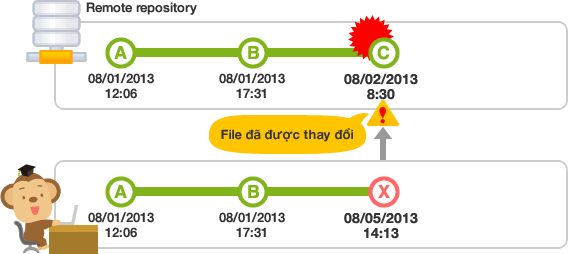
Trong thự mục đang làm việc để lấy về (*fetch*) và trộn (*merge*) các thay đổi ở remote. Để trộn một nhánh khác vào nhánh đang hoạt động (vd: master), sử dụng

“git merge <nhánh>”

Trong cả hai trường hợp, git cố gắng trộn tự động (auto-merge) các thay đổi.

Không may, điều này không phải lúc nào cũng làm được và thường dẫn đến xung đột. Trách nhiệm của bạn là trộn các xung đột đó thủ công bằng cách chỉnh sửa các tập tin được hiển thị bởi git. Sau khi thay đổi, bạn phải đánh dấu chúng là đã được trộn (merged) với lệnh:

“git add <tên-tập-tin>”



Trước khi trộn các thay đổi, bạn có thể xem trước chúng bằng các

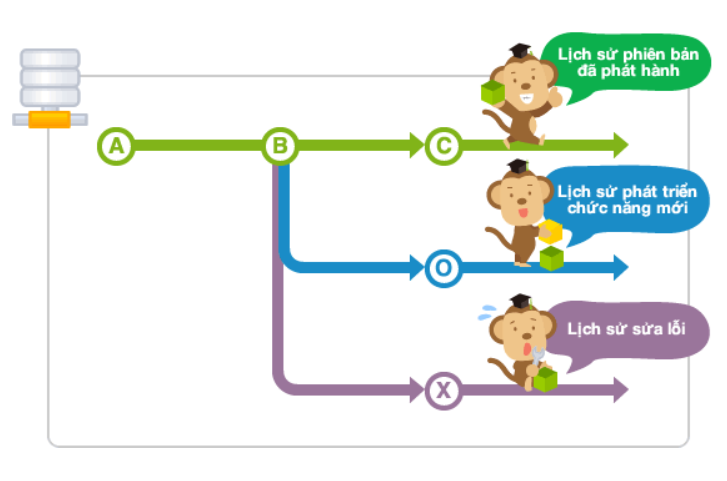
“git diff <nhánh\_nguồn> <nhánh\_mục\_tiêu>”.

1. Source git khi làm việc với nhiều người thì như thế nào?

Trong việc phát triển phần mềm, thì ứng với một phần mềm có nhiều thành viên đồng thời tiến hành thêm chức năng hay là tiến hành chỉnh sửa lỗi cùng một lúc. Và ở tình trạng tồn tại của nhiều phiên bản đã phát hành thì cũng phải lưu giữ từng phiên bản. Vì vậy để hỗ trợ quản lý phiên bản hay thêm nhiều chức năng được tiến hành song song, một chức năng được trang bị thêm được gọi là **branch** ở Git.

*Vậy* ***Branch*** *là gì?*

Branch là cái dùng để phân nhánh và ghi lại luồng của lịch sử. Branch đã phân nhánh sẽ không ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng 1 repository.

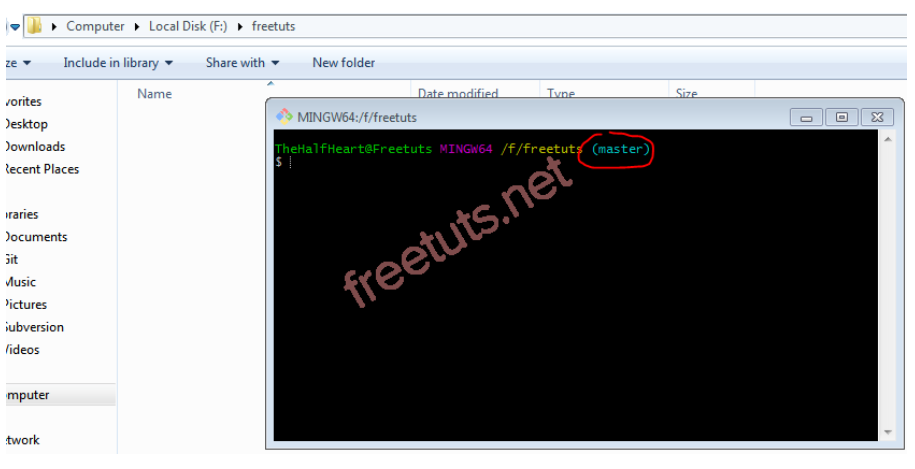
****

Các thành viên của nhóm sẽ tạo branch dùng riêng cho công việc của mình từ branch chính để không ảnh hưởng đến công việc của các thành viên khác. Sau đó, những thành viên đã hoàn thành công việc của mình sẽ thực hiện đưa thay đổi của mình vào branch chính. Theo cách như vậy, sẽ không bị ảnh hưởng từ công việc của các thành viên khác, và bản thân mình có thể thực hiện công việc của mình.

Hơn nữa, bằng việc để lại lịch sử theo đơn vị công việc, trong trường hợp có phát sinh vấn đề thì việc điều tra nguyên nhân ở những vị trí thay đổi cũng như việc tiến hành đối sách khắc phục sẽ trở nên dễ dàng hơn.

*Vậy làm thế nào để dùng* ***Branch****?*

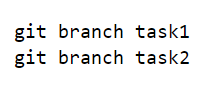
Mặc định khi bạn mở Git bash lần đầu tiên thì branch master được chọn như hình sau:



Lúc này ta không nên làm việc trên branch master mà hãy tạo thêm 2 branch bằng cú pháp sau:

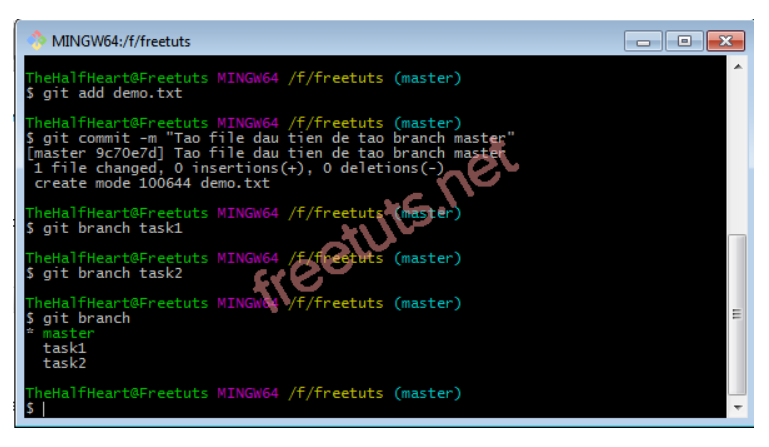


Dựa vào cú pháp này mình sẽ tạo 2 branch cho 2 task như sau:



Sau đó bạn nhập lệnh sau để xem danh sách branch, branch nào có dấu “ **\*** ” đằng trước là branch hiện tại.

Nếu như hình sau thì tức là bạn đã tạo thành công.



1. Lưu lại lịch sử thay đổi source Git

Để lưu lại lịch sử thay đổi mỗi khi ta chuyển sang Branch khác thì bạn nên lưu lại tất cả những gì đã làm bằng lệnh **commit**. Nếu bạn không commit thì sau khi chuyển sang branch khác những thay đổi của bạn sẽ không được lưu lại trong history và sau này bạn không thể rollback lại được.

Để lưu lại những gì đã làm vào branch ta dùng cú pháp:

“git commit -m "message" ”

Message là tin nhắn bạn muốn lưu cho lần commit đó:

VD:

git commit -m "Sua phan dang nhap"

Trước khi commit thì bạn phải sử dụng lệnh

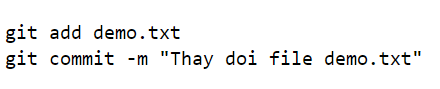
“git add <file\_name>”

để khai báo là sẽ commit file này, trường hợp bạn muốn commit tất cả các file có thay đổi thì sử dụng lệnh

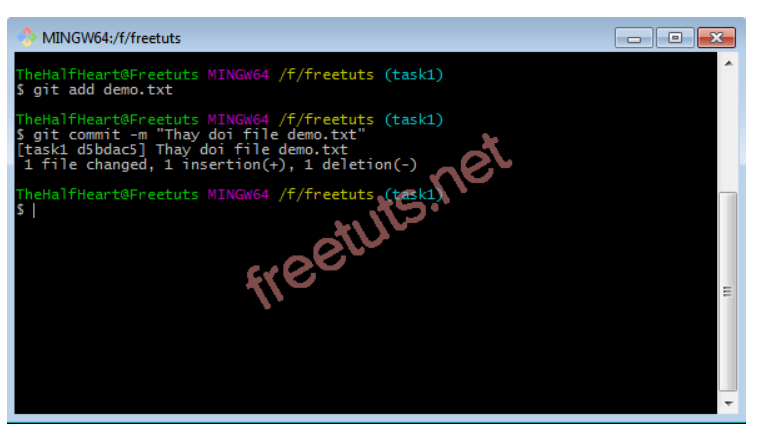
“git commit” .

Ví dụ:

Mình sửa file *demo.txt* và muốn commit nó lên thì thực hiện các bước như sau:



Kết quả:



1. Xem lịch sử commit:

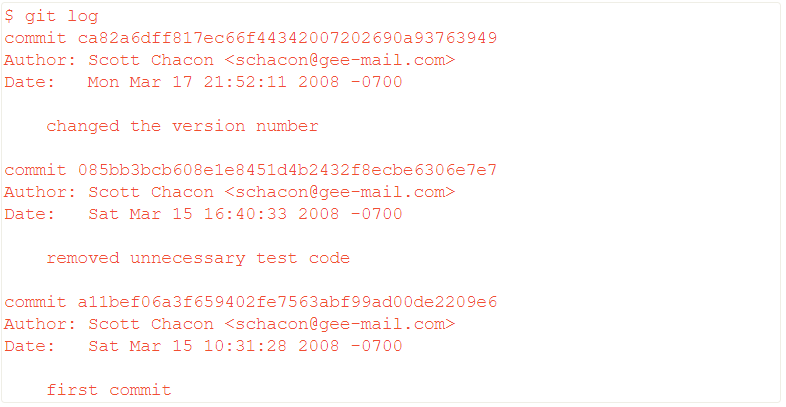
Sau khi bạn đã thực hiện rất nhiều commit, hoặc bạn đã sao chép một kho chứa với các commit có sẵn, chắc chắn bạn sẽ muốn xem lại những gì đã xảy ra. Cách đơn giản và có liệu lực tốt nhất là sử dụng lệnh

“git log”

Ví dụ sau đây sử dụng một dự án rất đơn giản là “simplegit” tôi thường sử dụng làm ví dụ minh hoạ. Để tải dự án này, bạn hãy chạy lệnh:



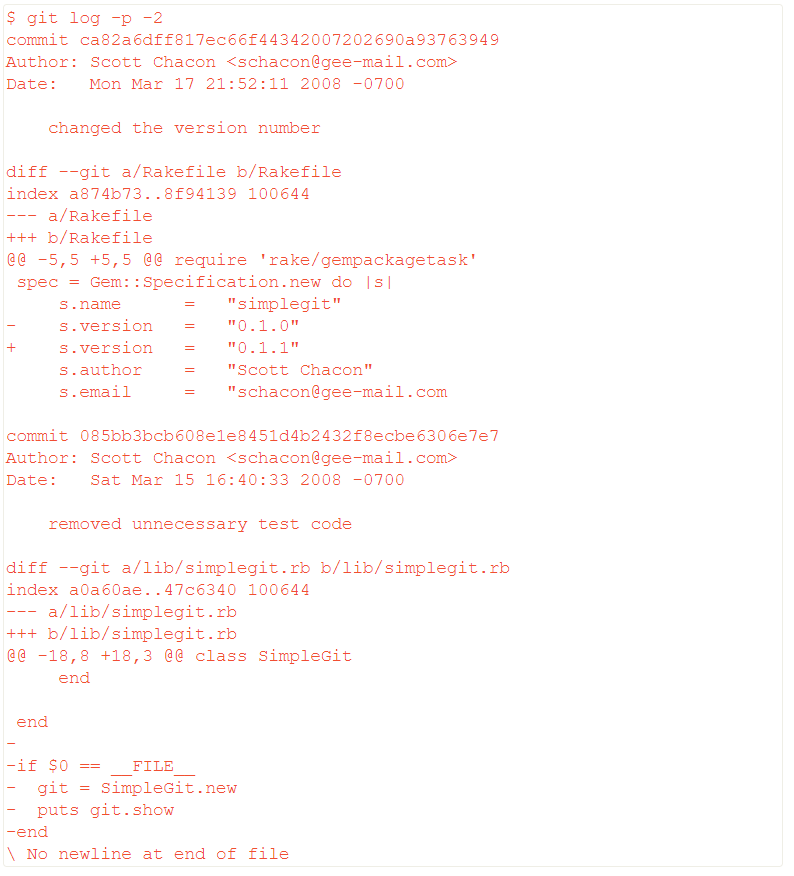
Khi bạn chạy “**git log**” trên dự án này, bạn sẽ thấy tương tự như sau:



Mặc định, không sử dụng tham số nào, “**git log**” liệt kê các commit được thực hiện trong kho chứa đó theo thứ tự thời gian. Đó là, commit mới nhất được hiển thị đầu tiên. Như bạn có thể thấy, lệnh này liệt kê từng commit với mã băm SHA-1, tên người commit, địa chỉ email, ngày lưu, và thông điệp của chúng.

Có rất nhiều tuỳ chọn (tham biến/số) khác nhau cho lệnh git log giúp bạn tìm chỉ hiện thị thứ mà bạn thực sự muốn.

Một trong các tuỳ chọn hữu ích nhất là -p, nó hiện thị diff của từng commit.



Lựa chọn **“--word-diff**” được cung cấp trong Git, bạn có thể thêm nó vào sau lệnh

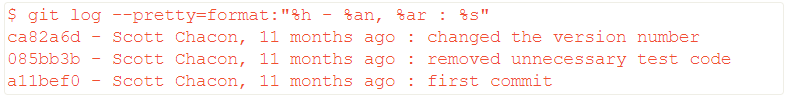
” git log –p”

để xem diff một cách tổng quát thay vì xem từng dòng theo cách thông thường. Xem diff tổng quát dường như là vô dụng khi sử dụng với mã nguồn.

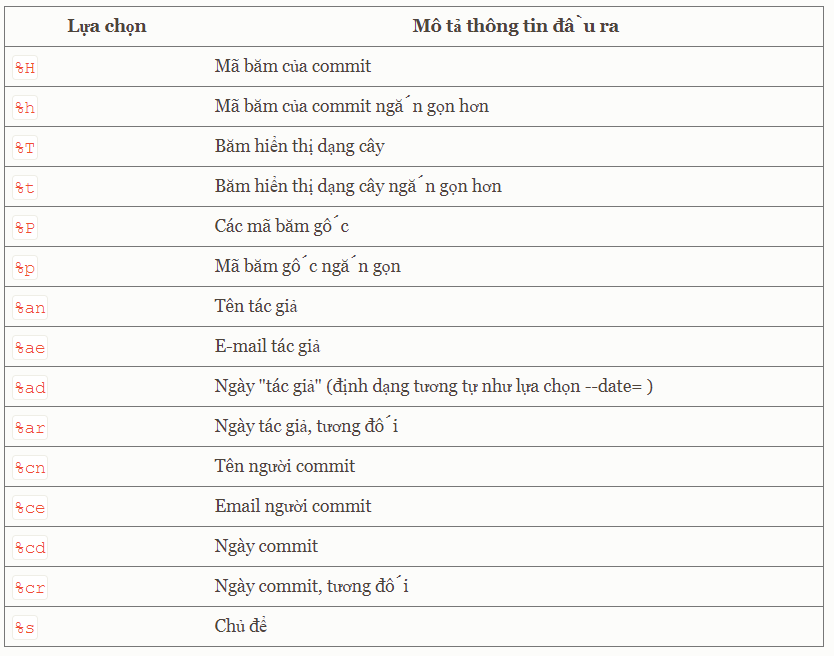


Lựa chọn thú vị nhất là “Format”, cho phép bạn chỉ định định dạng riêng của phần hiện thị. Nó đặc biệt hữu ích khi bạn đang xuất ra cho các máy phân tích thông tin (machine parsing).

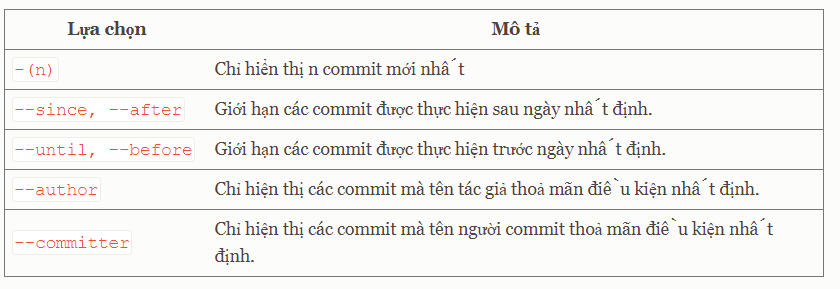
Vì bạn là người chỉ rõ định dạng, nên bạn sẽ biết được nó không bị thay đổi cùng với các cập nhật sau này của Git.



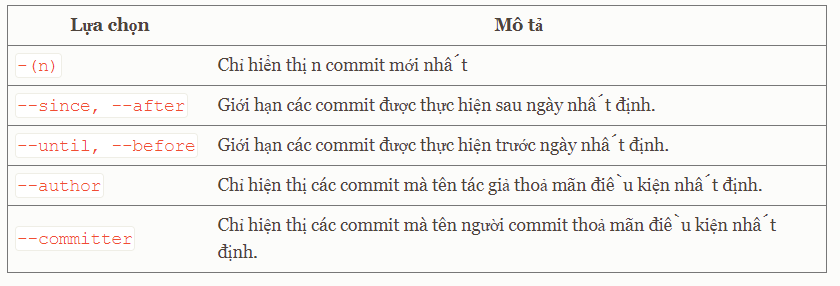
Bảng liệt kê một vài lựa chọn mà “Format” sử dụng:



Bảng liệt kê các lựa chọn chúng ta đã đề cập qua và một số định dạng cơ bản khác có thể hữu ích, cùng với mô tả đầu ra của lệnh “log”.



 Bảng liệt kê các lựa chọn trên và một số lựa chọn phổ biến khác:



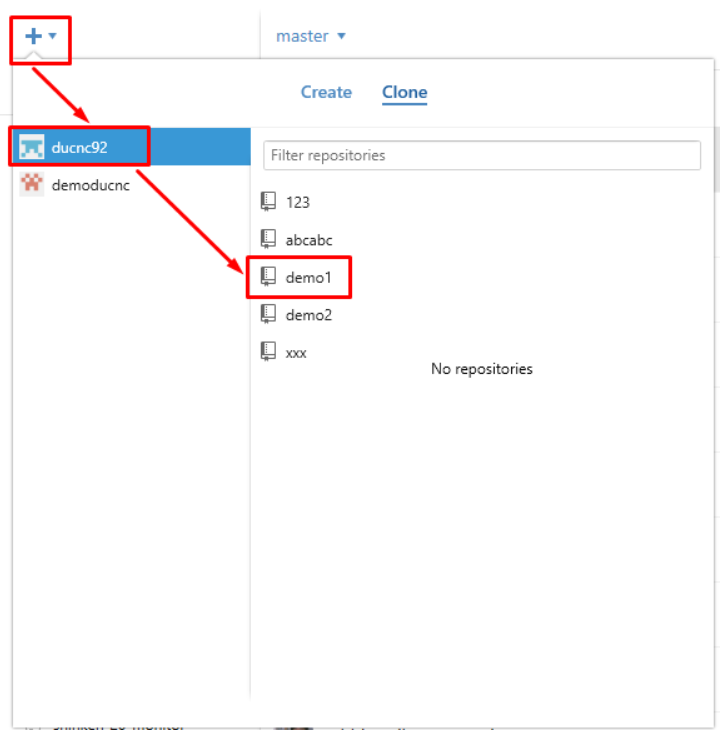
1. Clone 1 source về desktop

Để sao chép remote repository, sẽ thực hiện thao tác gọi là "clone".

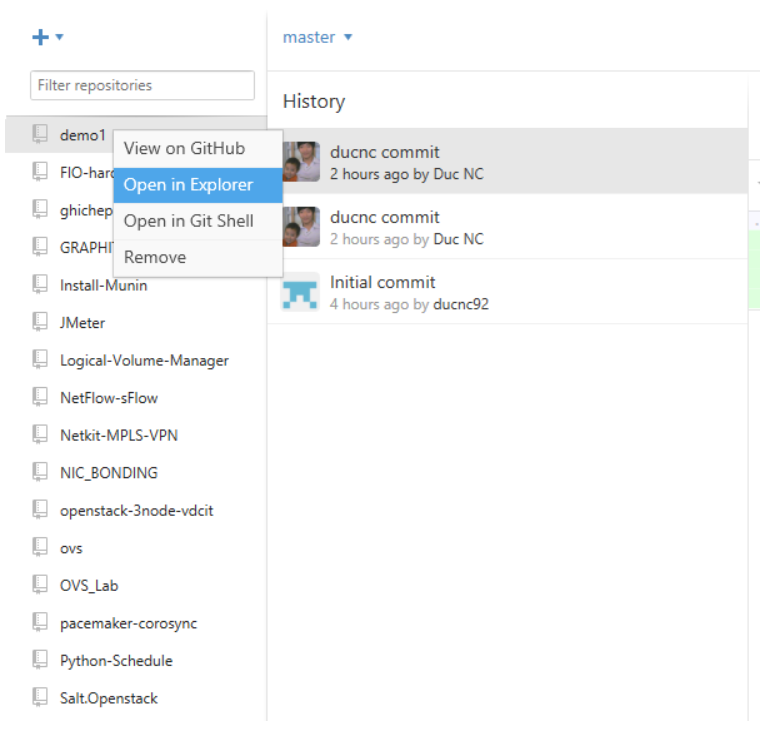
Khi thực hiện Clone, sẽ tải về toàn bộ nội dung của remote repository, và có thể tạo thành local repository ở máy khác.

*Các bước gồm:*

B1: Click vào dấu cộng, chọn tab Clone, lựa chọn tổ chức mong muốn và chọn repo cần clone



B2: Để chỉnh sửa nội dung của repo này ta chọn chuột phải vào nó và chọn Open in Explorer



Lúc đó chương trình Windows Explorer sẽ mở ra thư mục chứa repo của github, bạn có thể chỉnh sửa các file trong này, tạo xóa thư mục,... một cách bình thường.

~~~¥~~~

1. TỔNG KẾT

Qua đây các bạn có thể hiểu thêm về cách cài đặt, quản lí team, lịch sử cập nhật ,quản lí mã nguồn source code sao cho hiệu quả với công việc.

1. TÀI LIỆU THAM KHẢO

* <https://thachpham.com/tools/cach-tao-repository-cho-git.html>
* <https://freetuts.net/push-data-len-github-1112.html>
* <https://backlog.com/git-tutorial/vn/stepup/stepup1_1.html>
* <https://freetuts.net/git-lenh-tao-branch-1074.html?fbclid=IwAR1l1yvPk8NoswEcl5fikxuXON0j0UEHpsNB_TyBWv06uvEYxXfsiE8jxhA>
* <https://freetuts.net/git-commit-can-ban-1078.html>
* <https://git-scm.com/book/vi/v1/C%C6%A1-B%E1%BA%A3n-V%E1%BB%81-Git-Xem-L%E1%BB%8Bch-S%E1%BB%AD-Commit>
* <https://backlog.com/git-tutorial/vn/intro/intro3_2.html>
* <https://forum.xdavn.com/huong-dn-s-dung-github-toan-tp-t-a-z.t342/>
* <https://rogerdudler.github.io/git-guide/index.vi.html>
* <https://git-scm.com/book/vi/v1/C%C6%A1-B%E1%BA%A3n-V%E1%BB%81-Git-Ghi-L%E1%BA%A1i-Thay-%C4%90%E1%BB%95i-v%C3%A0o-Kho-Ch%E1%BB%A9a>

***HẾT***