**<https://nordiccoder.com/blog/git-va-github-la-gi-cach-su-dung-git-cap-nhat/>**

[**https://codelearn.io/sharing/git-github-tu-co-ban-den-nang-cao-p1**](https://codelearn.io/sharing/git-github-tu-co-ban-den-nang-cao-p1)

[**https://thachpham.com/series/git-co-ban**](https://thachpham.com/series/git-co-ban)

[**https://topdev.vn/blog/git-la-gi/**](https://topdev.vn/blog/git-la-gi/)

[**https://blog.duyet.net/2015/04/git-va-cac-khai-niem-co-ban.html**](https://blog.duyet.net/2015/04/git-va-cac-khai-niem-co-ban.html)

[**https://viblo.asia/p/tim-hieu-mot-so-khai-niem-va-ki-thuat-thuong-dung-trong-git-phan-1-naQZR4pP5vx**](https://viblo.asia/p/tim-hieu-mot-so-khai-niem-va-ki-thuat-thuong-dung-trong-git-phan-1-naQZR4pP5vx)

1. **Git là gì ? Git hoạt động như thế nào ? Git có lợi ích gì ?**

## ****Tìm hiểu cách sử dụng Git & GitHub****

### **Làm việc với các dự án GitHub**

1. **Các thuật ngữ Git quan trọng**

**5. Các lệnh git cơ bản**

**6. Lời khuyên khi thao tác thường xuyên với Git trong công việc**

**Git là gì?**

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System). Hiểu nôm na rằng Git là 1 hệ thống giúp cho việc quản lý tài liệu, source code... của 1 nhóm các developer cùng làm chung dự án. Git sẽ ghi nhớ lại toàn bộ lịch sử thay đổi của source code trong dự án. Bạn sửa file nào, thêm dòng code nào, xóa dòng code nào, bỏ thừa dấu ở đâu... tất cả các hành động đều được Git ghi lại. Qua đó giúp dự án có thể điều tra nguyên nhân gây lỗi hệ thống, tổng hợp code trở nên dễ dàng hơn.

## Cách Git hoạt động

Git coi thông tin được lưu trữ là một tập hợp các snapshot – ảnh chụp toàn bộ nội dung tất cả các file tại thời điểm.

Mỗi khi bạn “commit”, Git sẽ “chụp” và tạo ra một snapshot cùng một tham chiếu tới snapshot đó. Để hiệu quả, nếu các tệp không thay đổi, Git sẽ không lưu trữ lại file — chỉ là một liên kết đến tệp giống file trước đó mà nó đã lưu trữ

Git luôn xem xét lại hầu hết mọi khía cạnh của kiểm soát phiên bản mà hầu hết các hệ thống khác đã sao chép từ thế hệ trước. Điều này làm cho Git giống như một hệ thống tệp nhỏ với một số công cụ cực kỳ mạnh mẽ được xây dựng trên nó

Git có lợi ích gì

Các dự án thực tế thường có nhiều lập trình viên làm việc song song. Vì vậy, một hệ thống kiểm soát phiên bản như Git là cần thiết để đảm bảo không có xung đột code giữa các lập trình viên.

Dễ sử dụng, thao tác nhanh, gọn, lẹ và rất an toàn.

Sễ dàng kết hợp các phân nhánh (branch), có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều.

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 là bạn có thể làm việc ở mọi lúc mọi nơi.

Deployment sản phẩm của bạn một cách không thể nào dễ dàng hơn.

## 

## ****Trước khi sử dụng Git, chúng ta nên biết lý do tại sao chúng ta cần nó****

### **Git giúp dễ dàng đóng góp cho các dự án nguồn mở**

Gần như mọi dự án nguồn mở đều sử dụng GitHub để quản lý các dự án của họ. Sử dụng GitHub là miễn phí nếu dự án của bạn là nguồn mở và nó bao gồm wiki và trình theo dõi vấn đề giúp dễ dàng đưa vào tài liệu chuyên sâu hơn và nhận phản hồi về dự án của bạn.

Nếu bạn muốn đóng góp, bạn chỉ cần rẽ nhánh (lấy một bản sao) của một dự án, thực hiện các thay đổi của bạn và sau đó gửi pull request dự án bằng giao diện web của GitHub. Pull request này là cách bạn nói với dự án là bạn đã sẵn sàng để họ xem xét các thay đổi của bạn.

### **Tài liệu**

Bằng cách sử dụng GitHub, bạn sẽ dễ dàng có được tài liệu tuyệt vời. Phần trợ giúp và hướng dẫn của họ có bài viết cho gần như bất kỳ chủ đề nào liên quan đến Git mà bạn có thể nghĩ đến.

### **Tùy chọn tích hợp**

GitHub có thể tích hợp với các nền tảng phổ biến như Amazon và Google Cloud, với các dịch vụ như Code Climate để theo dõi phản hồi của bạn và có thể đánh dấu cú pháp trong hơn 200 ngôn ngữ lập trình khác nhau.

### **Theo dõi các thay đổi trong code của bạn qua các phiên bản**

Khi nhiều người cùng hợp tác trong một dự án, thật khó để theo dõi các sửa đổi – ai đã thay đổi những gì, khi nào và nơi các tệp đó được lưu trữ.

GitHub xử lý vấn đề này bằng cách theo dõi tất cả các thay đổi đã được đẩy vào kho lưu trữ.

Giống như sử dụng Microsoft Word hoặc Google Drive, bạn có thể có lịch sử phiên bản code của mình để các phiên bản trước không bị mất. Thật dễ dàng để trở lại phiên bản trước và đóng góp công việc của bạn.

### **Trưng bày công việc của bạn**

Bạn có phải là nhà phát triển muốn thu hút nhà tuyển dụng? GitHub là công cụ tốt nhất bạn có thể dựa vào cho việc này.

Ngày nay, khi tìm kiếm tân binh cho các dự án của họ, hầu hết các công ty đều nhìn vào hồ sơ GitHub. Nếu hồ sơ của bạn có sẵn, bạn sẽ có cơ hội được tuyển dụng cao hơn ngay cả khi bạn không đến từ một trường đại học hoặc cao đẳng lớn.

## ****Tìm hiểu cách sử dụng Git & GitHub****

Trước khi đi vào sử dụng git ta cần hiểu một số khái niệm liên quan đến Git như sau:

**Repository**: Repository hiểu đơn giản nó chính là cái kho lưu trữ tất cả những thông tin cần thiết để quản lý các sửa đổi và lịch sử của toàn bộ project. Repository của Git được phân thành 2 loại là **remote repository** và **local repository**.

* **Local Repository:** là repository nằm trên chính máy tính của chúng ta, repository này sẽ đồng bộ hóa với remote repository bằng các lệnh của git.
* **Remote Repository:** là repository được cài đặt trên server chuyên dụng. Ví dụ: GitHub, GitLab, Bitbucket,...

=> **GitHub** chính là 1 Remote Repository lưu trữ tất cả những thông tin cần thiết để quản lý các sửa đổi và lịch sử của toàn bộ project.

## ****Lệnh: git init****

## ****Tác dụng : Khởi tạo 1 git repository 1 project mới hoặc đã có.****

## ****Cách dùng: Tạo 1 folder mới => vào trong folder đó => click chuột phải chọn Git Bash Here****

### **Tạo tài khoản GitHub**

### **Cài đặt Git**

### **Làm việc với các dự án GitHub**

Chúng ta sẽ làm việc với các dự án GitHub theo hai cách.

#### **LOẠI 1: TẠO KHO LƯU TRỮ, SAO CHÉP NÓ VÀO PC CỦA BẠN VÀ LÀM VIỆC VỚI NÓ. (KHUYẾN KHÍCH)**

#### **LOẠI 2: LÀM VIỆC TRÊN DỰ ÁN CỦA BẠN CỤC BỘ SAU ĐÓ TẠO KHO LƯU TRỮ TRÊN GITHUB VÀ ĐẨY NÓ VÀO REMOTE**

1. Branch

Các Branch (nhánh) đại diện cho các phiên bản cụ thể của một kho lưu trữ tách ra từ project chính của bạn.

Branch cho phép bạn theo dõi các thay đổi thử nghiệm bạn thực hiện đối với kho lưu trữ và có thể hoàn nguyên về các phiên bản cũ hơn.

2. Commit

Commit đại diện cho một thời điểm cụ thể trong lịch sử dự án của bạn. Sử dụng lệnh commit kết hợp với lệnh git add để cho git biết những thay đổi bạn muốn lưu vào local repository.

*Index*

*Bất cứ khi nào bạn thêm, xóa hoặc thay đổi một file, nó vẫn nằm trong chỉ mục cho đến khi bạn sẵn sàng commit các thay đổi. Nó như là khu vực tổ chức (stagging area) cho Git. Sử dụng lệnh git status để xem nội dung của index của bạn.*

Checkout

Sử dụng lệnh git checkout để chuyển giữa các branch. Chỉ cần nhập git checkout theo sau là tên của branch bạn muốn chuyển đến hoặc nhập git checkout master để trở về branch chính (master branch).

## ****Master****

## ****Master là nhánh chính của tất cả các repository của bạn. Nó nên bao gồm những thay đổi và commit gần đây nhất.****

## Push

## Lệnh git push được sử dụng để cập nhật các nhánh từ xa với những thay đổi mới nhất mà bạn đã commit.

Remote

Một Remote (kho lưu trữ từ xa) là một bản sao của một chi nhánh. Remote giao tiếp ngược dòng với nhánh gốc (origin branch) của chúng và các Remote khác trong kho lưu trữ.

Repository

Kho lưu trữ Git chứa tất cả các tệp dự án của bạn bao gồm các branch, tags và commit.

1) git config

Tác dụng : Để set user name và email của bạn trong main configuration file.

Cách xài : Để kiểm tra tên và kiểu email trong cấu hình dùng git config -- global user.name và git config -- global user.email. Để set email hoặc tên mới git config -- global user.name = “Hải Nguyễn” và git config -- global user.email = “hainguyen@gmail.com”

2) git init

Tác dụng : Khởi tạo 1 git repository 1 project mới hoặc đã có.

Cách xài: git init trong thư mục gốc của dự án.

3) git clone

Tác dụng: Copy 1 git repository từ remote source.

Cách xài: git clone <:clone git url:>

4) git status

Tác dụng: Để check trạng thái của những file bạn đã thay đổi trong thư mục làm việc. VD: Tất cả các thay đổi cuối cùng từ lần commit cuối cùng.

Cách xài: git status trong thư mục làm việc.

5) git add

Tác dụng: Thêm thay đổi đến stage/index trong thư mục làm việc.

Cách xài: git add

6) git commit

Tác dụng: commit nghĩa là một action để Git lưu lại một snapshot của các sự thay đổi trong thư mục làm việc. Và các tập tin, thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của code 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 branch khác, vì vậy bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó.

Cách dùng: git commit -m ”Đây là message, bạn dùng để note những thay đổi để sau này dễ dò lại”

7) git push/git pull

Tác dụng: Push hoặc Pull các thay đổi đến remote. Nếu bạn đã added và committed các thay đổi và bạn muốn đẩy nó lên hoặc remote của bạn đã update và bạn apply tất cả thay đổi đó trên code của mình.

Cách dùng: git pull <:remote:> <:branch:> and git push <:remote:> <:branch:>

8) git branch

Tác dụng: liệt kê tất cả các branch (nhánh).

Cách dùng: git branch hoặc git branch -a

9) git checkout

Tác dụng: Chuyển sang branch khác

Cách dùng: git checkout <: branch:> hoặc \*\* \_ git checkout -b <: branch:> nếu bạn muốn tạo và chuyển sang một chi nhánh mới.

13) git remote

Tác dụng: Để check remote/source bạn có hoặc add thêm remote

Cách dùng: git remote để kiểm tra và liệt kê. Và git remote add <: remote\_url:> để thêm.

14) git add

Tác dụng: Để đưa một tập tin vào Staging Area

Cách dùng: git add tên\_file hoặc muốn thêm hết file của thư mục thì git add all