Git là gì?

* là một hệ thống quản lý phiên bản phân tán, nó là một hệ thống quản lý phân tán phổ biến nhất hiện nay, Git cung cấp cho mỗi lập trình viên kho lưu trữ (Repo) riêng chứa toàn bộ lịch sử thay đổi.

Version Control System - VCS là gì ?

* là hệ thống kiếm soát các phiên bản phân tán mã nguồn mở, các VCS sẽ lưu trữ tất cả các file trong toàn bộ dự án và ghi lại toàn bộ lịch sử thay đổi của file. Mỗi sự thay đổi được lưu lại sẽ được và thành một version.
* VCS nghĩa là hệ thống giúp lập trình viên có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn nhân bản (clone) từ một kho chứa mã nguồn (repo), mỗi thay đổi vào mã nguồn trên local sẽ có thể ủy thác commit rồi đưa lên sever 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.
* Lập trình viên có thể xem lại danh sách các sự thay đổi của file như xem một dòng thời gian của các phiên bản. Mỗi phiên bản bao gồm nội dung file bị thay đổi, ngày giờ sửa đổi, người thay đổi là ai, lý do thay đổi hay tên phiên bản…

VCS có tsc dụng như thế nào ?

1. Lưu lại lịch sử các version của bất kỳ thay đổi nào của dự án, giúp xem lại các sự thay đổi.
2. việc chia sẻ code trở nên dễ dàng hơn, lập trình viên có thể public cho bất kỳ ai, hoặc private chỉ cho một số người có thẩm quyển truy cập

Git hoạt động như thế nào?

* Sự khác biệt giữa git và VCS là cách Git nghĩ về dữ liệu của nó
* Về mặt khái niệm, hầu hết các hệ thống khác đều lưu trữ thông tin dưới dạng danh sách các thay đổi dựa trên file. Các hệ thống này coi thông tin chúng lưu trữ dưới dạng một tập hợp các file và những thay đổi được thực hiện đối với mỗi file theo thời gian.
* Git không nghĩ như vậy, Git coi thông tin được lưu trữ là một tập hợp các 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 nó. Để hiệu quả nếu các tệp không thay đổi, Git sẽ không lưu trữ file - chỉ là một liên kết đếm tệp giống file trước đó mà nó lưu trữ

Các thuật ngữ Git quan trọng

1. Branch

* Các Branch 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 phiên bản cũ hơn.

1. Commit

* Một 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 Repo

1. 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 check out master để trở về branch chính

1. Fetch

* Lệnh git fetch tìm nạp bản sao và tải xuống tất cả các tệp branch vào máy tính của bạn. Sử dụng nó để lưu trữ các thay đổi mới nhất vào kho lưu trữ của bạn. Nó có thể tìm nạp nhiều branch cùng một lúc.

1. Fork

* Một fork là một bản sao của một kho lưu trữ. Các lập trình viên tận dụng lợi ích của fork để thử nghiệm các thay đổi mà không ảnh hưởng đến dự án chính

1. Head

* Các commit ở đầu của một branch được gọi là head. Nó đại diện cho commit mới nhất của Repo mà bạn đang làm việc

1. 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 index của bạn.

1. Master

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

1. Merge

* Lệnh git merge kết hợp với các yêu cầu kéo (pull requests) để thêm các thay đổi từ nhánh này sang nhánh khác

1. Origin

* là phiên bản mặc định của repo, đóng vai trò là bí danh hệ thống để liên lạc với nhánh chính
* lệnh git push origin master để đấy các thay đổi cục bộ lên nhánh chính

1. Pull

* Thể hiện các đề xuất thay đổi cho nhánh chính. Nếu bạn làm việc với một nhóm, bạn có thể tạo pull request để yêu cầu người bảo trì kho lưu trữ xem xét các thay đổi và hợp nhất chúng
* lệnh git pull được sử dụng để thêm các thay đổi vào nhánh chính

1. Push

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

1. Rebase

* git rebase cho phép bạn phân tách, di chuyển hoặc thoát khỏi các commit. Nó cũng có thể dùng để kết hợp 2 nhánh khác nhau

1. Remote

* Kho lưu trữ từ xa là một bản sao của 1 nhánh, Giao tiếp ngược dòng với nhánh gốc oirgin main của chúng và các remote khác

1. Repo

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

1. Stash

* Git stash sẽ loại bỏ các thay đổi khỏi chỉ mục của bạn và xóa stashes chúng đi sau.
* Nó có ích nếu bạn muốn tạm dừng những gì bạn đang làm và làm việc khác trong một khoảng thời gian. bạn không thể đặt stash nhiều hơn một bộ thay đổi ở cùng một thời điểm

1. Tags

* cung cấp cho bạn một cách để theo dõi các commit quan trọng. các tags nhẹ chỉ đơn giản đóng vai trò là con trỏ trong khi các tags chú thích được lưu trữ dưới dạng đối tượng đầy đủ

1. Upstream

* Đề cập đến nới bạn push các thay đổi của mình, thường là nhánh chính

Các lệnh git cơ bản

1. git config

* Tác dụng để set user name và email của bạn trong main … 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***](mailto: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.***

#### ***10) git stash***

***Tác dụng: Lưu thay đổi mà bạn không muốn commit ngay lập tức.***

***Cách dùng: git stash trong thư mục làm việc của bạn.***

#### ***11) git merge***

***Tác dụng: Merge 2 branch lại với nahu.***

***Cách dùng: Chuyển tới branch bạn muốn merge rồi dùng git merge <:branch\_ban\_muon\_merge:>***

#### ***12) git reset***

***Tác dụng: Bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo.***

***Cách dùng: git reset HEAD tên\_file***

#### ***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***

***Git (alone)***

***Các cách làm việc với Git***

***1. Tạo một project trên github***

***- tạo 1 repo***

***- kéo về bằng git clone***

***- viết code***

***+ git status (kiểm tra trạng thái files)***

***+ git add . thêm files***

***+ git commit -m "name"***

***+ git push origin main***

***2. Tạo một pj local và đẩy lên github***

***- Tạo 1 repo***

***- git init khởi tạo thư mục mới cho local***

***- Viết code***

***Giống trên***

***thêm dòng git remote add origin ...***

***3. HTTPS và SSH***

***SSH không cần nhập lại password***

***dùng các thuật toán để sinh ra khóa bí mật và khóa công khai***

***4. Các câu lệnh hay dùng của Git***

***1. git config***

***git config user.name***

***git config user.email***

***cài user name và mail***

***2. git init***

***tạo thư mục .git***

***kiểm soát mã nguồn tại local***

***3. git clone***

***kéo từ remote về local***

***4. git status***

***- trạng thái các files trên máy tính***

***untrack : chưa từng làm việc***

***modifier: files tồn tại trước đây***

***5. git add .***

***các file đanh muốn thêm vào, lưu lại lịch sử của file này***

***git add <file1> <file2>----<fileN>***

***6. git commit***

***- hiểu được nội dung công việc là gì***

***- chú thích cho nội dung công việc mình đang làm***

***7. git push***

***- đẩy code từ local lên remote***

***git conflict***

***có 1 file có nhiều người sửa***