TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

🙟🕮🙝

A picture containing text, clipart

Description automatically generated

MÔN:

BÁO CÁO BÀI TẬP NHÓM 6

ĐỀ TÀI: TÌM HIỂU VỀ GIT

Thành viên nhóm : Nguyễn Đức Nam

Đặng Tiến Toàn

Nguyễn Trường An

Lớp: 70DCTT21

HÀ NỘI – 08/2022

PHỤ LỤC

[CHƯƠNG 1: GIỚI THIỆU VỀ GIT và VCS 3](#_Toc111975182)

[1.1: Khái niệm về git 3](#_Toc111975183)

[1.2: khái niệm về VCS 3](#_Toc111975184)

[1.3: Tác dụng của VCS 3](#_Toc111975185)

[CHƯƠNG 2: NGUYÊN LÝ HOẠT ĐỘNG 5](#_Toc111975186)

[2.1: Quản lý phiên bản 5](#_Toc111975187)

[2.2: Luồng công việc phân tán 5](#_Toc111975188)

[CHƯƠNG 3: LỢI ÍCH KHI SỬ DỤNG GIT 7](#_Toc111975189)

[3.1 Khả năng phân nhánh (branch) 7](#_Toc111975190)

[3.2 Khả năng phát triển phân tán 7](#_Toc111975191)

[3.3 Cộng đồng tích cực, nhân lực dồi dào 7](#_Toc111975192)

[3.4 Chu kỳ phát triển nhanh hơn 7](#_Toc111975193)

[3.5 Quy trình xử lý công việc trên Git 7](#_Toc111975194)

[CHƯƠNG 4: CÁC THUẬT NGỮ VÀ LỆNH GIT CƠ BẢN 8](#_Toc111975195)

[4.1 Các thuật ngữ: 8](#_Toc111975196)

[4.2 các lệnh git cơ bản 11](#_Toc111975197)

CHƯƠNG 1: GIỚI THIỆU VỀ GIT và VCS

1.1: Khái niệm về git

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository) riêng chứa toàn bộ lịch sử thay đổi.

1.2: khái niệm về VCS

VCS là viết tắt của **Version Control System** là **hệ thống kiểm 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 vàthành một version (phiên bản).

VCS nghĩa là hệ thống giúp lập trình viên 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 local sẽ có thể ủy thác (**commit**) rồi đưa lên server 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…

1.3: Tác dụng của VCS

- Lưu lại lịch sử các version của bất kỳ thay đổi nào của dự án. Giúp xem lại các sự thay đổi hoặc khôi phục (revert) lại sau này.

- Việc chia sẻ code trở nên dễ dàng hơn, lập trình viên có thể để public cho bất kỳ ai, hoặc private chỉ cho một số người có thẩm quyền có thể truy cập và lấy code về.

Vốn là một VCS nên Git cũng ghi nhớ lại toàn bộ lịch sử thay đổi của source code trong dự án. Lập trình sửa file, thêm dòng code tại đâu, xóa dòng code ở hàng nào…đều được Git ghi nhận và lưu trữ lại.

CHƯƠNG 2: NGUYÊN LÝ HOẠT ĐỘNG

2.1: Quản lý phiên bản

Giống như các VCS khác, Git hoạt động bằng cách ghi nhận lại mọi thay đổi trong hệ thống file theo thời gian. Sự khác biệt của Git là mỗi khi một thay đổi được người dùng commit (xác nhận), Git sẽ chụp lại trạng thái của toàn bộ hệ thống file tại thời điểm đó (snapshot), nghĩa là lưu lại phiên bản đã thay đổi của các file.

Như có thể thấy trong biểu đồ trên, mỗi Version của hệ thống file đều chứa đầy đủ các file kể cả có thay đổi hay không. Ví dụ tại Version 2 có sự thay đổi tại file A và file C so với Version 1, các file đã thay đổi là A1 và C1 sẽ được lưu vào chỗ tương ứng. Git cũng rất thông minh khi không copy lại file B không bị thay đổi mà chỉ tham chiếu đến vị trí của file B trong version trước đó, giúp giảm lãng phí bộ nhớ do trùng lặp.

Mặt khác, các VCS khác như CVS, Subversion, Perforce,… đều giữ lại các file ban đầu như một phiên bản gốc và chỉ lưu những phần bị thay đổi (delta) tương ứng với từng file mỗi lần cập nhật thay đổi.

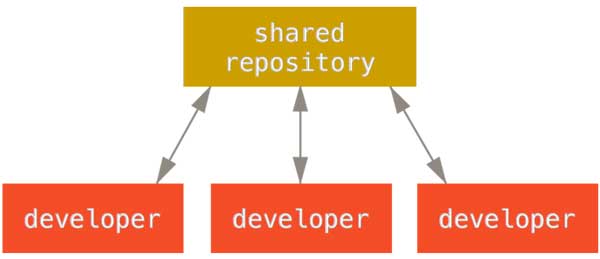
2.2: Luồng công việc phân tán

Ta đã nói về chức năng của git như một VCS thông thường, nhưng định nghĩa về git là DVCS, nghĩa là còn một chữ D quan trọng chưa được nhắc tới, đó là tính phân tán – Distributed.

Ngược lại với hệ thống kiểm soát phiên bản tập trung (CVCS), bản chất phân tán của Git cho phép bạn linh hoạt hơn nhiều trong cách các nhà phát triển cộng tác trong các dự án. Trong các hệ thống tập trung, mỗi nhà phát triển là một nút hoạt động ngang bằng với với trung tâm.

Tuy nhiên, trong Git, mọi nhà phát triển đều có khả năng vừa là nút vừa là trung tâm; nghĩa là, mọi nhà phát triển đều có thể đóng góp mã nguồn cho các kho lưu trữ (repository) khác và duy trì một kho lưu trữ công khai mà những người khác có thể làm cơ sở cho công việc của họ và họ có thể đóng góp vào đó. Điều này đưa ra nhiều lựa chọn về quy trình làm việc cho dự án, sau đây là một ví dụ.

Các workflow có tính tập trung nghĩa là mọi developer trong dự án đều làm việc trên cùng một kho chứa mã nguồn trung tâm (shared repository). Mỗi người khi làm việc cần đồng bộ mã nguồn trên máy mình với trung tâm trước, sau đó mới thực hiện thay đổi để đảm bảo không gây xung đột.



*Git và các DVCS trong khi đó cho phép các nhà phát triển tạo một bản sao của repository gốc*

Nếu có nhiều developers cùng thực hiện thay đổi, người đầu tiên đẩy các thay đổi của mình vào shared repository sẽ không gặp vấn đề gì. Tuy nhiên những người sau nếu không đồng bộ với phiên bản bị sửa đổi bởi người đầu tiên trước khi thực hiện sửa đổi của riêng mình thì có thể sẽ gây tình trạng xung đột, tức là sửa đổi lẽ ra có mục tiêu là phiên bản gốc nay lại áp dụng cho mã nguồn đã bị thay đổi.

Git và các DVCS trong khi đó cho phép các nhà phát triển tạo một bản sao của repository gốc (branch) trên máy mình và thực hiện mọi thứ thay đổi trên đó một cách độc lập với repository gốc, không lo xung đột với người khác, commit tức thì do không cần kết nối đến repository gốc.

CHƯƠNG 3: LỢI ÍCH KHI SỬ DỤNG GIT

3.1 Khả năng phân nhánh (branch)

Một trong những ưu điểm lớn nhất của Git là khả năng phân nhánh của nó. Không giống như các hệ thống kiểm soát phiên bản tập trung, các branch trong Git rất dễ tạo và dễ hợp nhất.

Các branch cung cấp một môi trường riêng biệt cho mọi thay đổi đối với mã nguồn. Khi một nhà phát triển muốn thay đổi hay thêm tính năng, họ tạo một nhánh mới, làm việc trên đó và kiểm thử trước khi đưa các thay đổi vào nhánh chính. Điều này đảm bảo rằng nhánh chính luôn chứa mã nguồn chuẩn và tối ưu nhất.

3.2 Khả năng phát triển phân tán

Git là một hệ thống kiểm soát phiên bản phân tán, mỗi developer có thể có kho lưu trữ cục bộ của riêng họ, với đầy đủ lịch sử thay đổi. Do đó các thành viên có thể làm việc độc lập với nhau và sau đó hợp nhất khi mọi thứ hoàn chỉnh, không cần duy trì kết nối liên tục đến nhánh gốc.

Phát triển phân tán cũng tạo ra một môi trường đáng tin cậy hơn. Ngay cả khi một nhà phát triển lỡ xóa kho lưu trữ của riêng họ, họ chỉ cần tạo lại một bản sao từ nơi khác và tiếp tục công việc.

3.3 Cộng đồng tích cực, nhân lực dồi dào

Là hệ thống quản lý phiên bản nguồn mở phổ biến bậc nhất, Git có một cộng đồng đông đảo và luôn sẵn sàng hỗ trợ. Nếu dự án của bạn đang sử dụng Git, khả năng cao bạn sẽ không phải đào tạo nhân viên mới về quy trình làm việc của mình, vì họ đã quen với phát triển phân tán trên git.

3.4 Chu kỳ phát triển nhanh hơn

Kết quả cuối cùng của các ưu điểm trên chu kỳ phát triển nhanh hơn cho dự án của bạn, giúp tiết kiệm chi phí và sản phẩm cuối cùng cũng tốt hơn.

3.5 Quy trình xử lý công việc trên Git

Git đánh dấu 3 trạng thái chính của một tệp: modified (đã sửa đổi), staged (đã đánh dấu thay đổi) và commit (ghi lại thay đổi).

* Modified có nghĩa là đã có thay đổi trong tệp nhưng chưa cam kết nó với cơ sở dữ liệu.
* Staged có nghĩa là đánh dấu một tệp đã sửa đổi so với phiên bản hiện tại của nó để đưa vào lần snapshot tiếp theo.
* Commit có nghĩa là dữ liệu đã được lưu trữ an toàn trong cơ sở dữ liệu cục bộ.

Quy trình công việc Git cơ bản diễn ra như sau:

* Sửa đổi các tệp trong working directory.
* Chọn lọc những thay đổi muốn commit và thêm những thay đổi đó vào staging area.
* Thực hiện một commit, lấy các tệp từ staging area, lưu snapshot phiên bản vào Git.

Nếu một phiên bản cụ thể của tệp nằm trong thư mục Git, thì tệp đó được coi là đã commit. Nếu nó đã được sửa đổi và được thêm vào staging area, đó là trạng thái staged. Và nếu nó đã được thay đổi mà chưa đưa vào staging area thì gọi là modified.

CHƯƠNG 4: CÁC THUẬT NGỮ VÀ LỆNH GIT CƠ BẢN

4.1 Các thuật ngữ:

1. Branch

Các Branch (nhánh) là các repository tách ra từ repository của project chính.

Branch cho phép ghi nhận mọi thay đổi với repo và có thể hoàn nguyên về các phiên bản cũ hơn.

2. 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 repository.

3. 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).

4. Fetch

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

5. Fork

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

6. Head

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

7. 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.

8. Master

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

9. Merge

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

10. Origin

Origin là phiên bản mặc định của repository. Origin cũng đóng vai trò là cú pháp để truy cập nhánh chính.

Lệnh git push origin master để đẩy các thay đổi cục bộ đến nhánh chính.

11. Pull

Pull requests thể hiện các đề xuất thay đổi cho nhánh chính. Nếu bạn làm việc với một nhóm, bạn có thể tạo các 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.

12. Push

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

13. Rebase

Lệnh git rebase cho phép bạn phân tách, di chuyển hoặc thoát khỏi các commit. Nó cũng có thể được sử dụng để kết hợp hai nhánh khác nhau.

14. 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.

15. Repository

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

16. Stash

Lệnh 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.

17. Tags

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 các đối tượng đầy đủ.

18. Upstream

Trong ngữ cảnh của Git, upstream đề cập đến nơi bạn push các thay đổi của mình, thường là nhánh chính (master branch).

Hi vọng qua bài viết này các bạn đã có được cái nhì tổng quát về Git, cách Git vận hành và tầm quan trọng của nó trong các dự án phát triển phần mềm. Hãy tiếp tục theo dõi bizflycloud để cùng chúng tôi cập nhật những xu hướng mới nhất về công nghệ nói chung và công nghệ thông tin nói riêng mỗi ngày.

4.2 các lệnh git cơ bản

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 gitconfig -- 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.

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