**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----🙠🕮🙢-----**

****

**THỰC TẬP CƠ SỞ**

**Chủ đề: Tìm hiểu về Git**

**Giảng viên: Nguyễn Văn Tiến**

**Nhóm: 28**

**Họ tên: Phạm Việt Quân**

**Mã sinh viên: B21DCCN104**

**Hà Nội 2024**

**MỤC LỤC**

[I. Tài liệu tham khảo 3](#_Toc160324886)

[II. Tìm hiểu về git. 3](#_Toc160324887)

[1. Git là gì ? Cách thức hoạt động của Git. 3](#_Toc160324888)

[2. Tại sao nên sử dụng Git? 4](#_Toc160324889)

[III. Một số thuật ngữ cơ bản về Git. 5](#_Toc160324890)

[1. Repository 5](#_Toc160324891)

[2. Branch 5](#_Toc160324892)

[3. Tags 5](#_Toc160324893)

[4. Commit 5](#_Toc160324894)

[5. Checkout 5](#_Toc160324895)

[6. Fetch 5](#_Toc160324896)

[7. Fork 6](#_Toc160324897)

[8. Head 6](#_Toc160324898)

[9. Index 6](#_Toc160324899)

[10. Merge 6](#_Toc160324900)

[11. Origin 6](#_Toc160324901)

[12. Pull 6](#_Toc160324902)

[13. Push 6](#_Toc160324903)

[14. Rebase 7](#_Toc160324904)

[15. Remote 7](#_Toc160324905)

[16. Stash 7](#_Toc160324906)

[17. Upstream 7](#_Toc160324907)

[IV. Các lệnh Git phổ biến. 7](#_Toc160324908)

[1. git config 7](#_Toc160324909)

[2. git init 8](#_Toc160324910)

[3. git clone 8](#_Toc160324911)

[4. Git status 8](#_Toc160324912)

[5. git add 8](#_Toc160324913)

[6. git commit 8](#_Toc160324914)

[7. git push/git pull 9](#_Toc160324915)

[8. git branch 9](#_Toc160324916)

[9. git checkout 9](#_Toc160324917)

[10. git stash 10](#_Toc160324918)

[11. git merge 10](#_Toc160324919)

[12. git reset 10](#_Toc160324920)

# 

# I. Tài liệu tham khảo

Tài liệu tham khảo : <https://www.w3schools.com/git/git_intro.asp?remote=github>

<https://topdev.vn/blog/git-la-gi/>

<https://www.youtube.com/watch?v=cC7vg8lWneo>

<https://xuanthulab.net/dung-lenh-git-reset-huy-commit-cuoi-hoac-staging.html>

# II. Tìm hiểu về git.

## 1. Git là gì ? Cách thức hoạt động của Git.

- Git, viết tắt của từ Global Information Tracker. Hiểu một cách đơn giản, GIT là hệ thống quản lý các phiên bản dưới dạng phân tán. Đây là hệ thống quản lý phổ biến nhất hiện nay. GIT là một phiên bản của Distributed Version Control System – DVCS hay còn gọi là VCS. Sử dụng GIT sẽ đem tới cho các lập trình viên một kho lưu trữ, trong đó chứa đầy đủ các lịch sử thay đổi của hệ thống.

- GIT là một VCS. Tuy nhiên, so với các VCS thông thường, GIT cũng có một số điểm khác biệt. Trong đó, điểm khác biệt nổi bật nhất của GIT đó là cách đọc và lưu trữ dữ liệu của phần mềm này.

- Cơ chế lưu trữ phiên bản của Git là nó sẽ tạo ra một snapshot (lưu lại nội dung tất cả các tập tin, thư mục tại thời điểm đó) trên mỗi tập tin sau khi commit, từ đó nó có thể cho phép ta tái sử dụng lại một ảnh chụp nào đó mà bạn có thể hiểu đó là một phiên bản. Đây cung chính là lợi thế của Git so với các DVCS khác khi nó không “lưu cứng” dữ liệu mà sẽ lưu với dạng snapshot.

## 2. Tại sao nên sử dụng Git?

- Git sẽ giúp chúng ta :

+ Cập nhật và lưu lại những phiên bản khác nhau của source code.

+ Khôi phục lại source code từ một phiên bản bất kỳ.

+ Dễ dàng so sánh những thay đổi từ các phiên bản khác nhau.

+ Phát hiện ai đã sửa phần nào, đã làm phát sinh lỗi.

+ Khôi phục lại tập tin bị mất.

+ Phối hợp với các đồng nghiệp một cách hiệu quả.

- Lợi ích khi sử dụng Git :

+ Khi không có kết nối internet, chúng ta vẫn có thể làm việc và commit. Và các lập trình viên có thể làm việc cùng nhau từ mọi nơi trên thế giới.

+ Thời gian trễ thấp do hầu hết dữ liệu đều được sao chép ở máy cục bộ.

+ Làm việc dễ dàng hơn với branch.

+ Dễ dàng trong việc deployment sản phẩm

# III. Một số thuật ngữ cơ bản về Git.

## 1. Repository

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

## 2. 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.

## 3. 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.

## 4. Commit

- Được hiểu là để ghi lại những sự thay đổi của file đó vào repository. Khi thực hiện commit, ở repository sẽ tạo ra commit để ghi lại sự khác biệt trong ở trạng thái hiện tại và lần trước. Các commit nối tiếp với nhau, sắp xếp theo thời gian.

## 5. 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).

## 6. 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.

## 7. 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.

## 8. 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.

## 9. 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.

## 10. Merge

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

## 11. Origin

- Origin là phiên bản mặc định của repository. Origin cũng đóng vai trò là bí danh hệ thống để liên lạc với nhánh chính.

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

## 12. 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.

## 13. Push

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

## 14. 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.

## 15. 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.

## 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. Upstream

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

# IV. Các lệnh Git phổ biến.

## 1. git config

-- Là câu lệnh mà chúng ta phải thực thi đầu tiên khi cài đặt Git lên máy

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

VD: git config –global user.name=”name”

git config –global user.email=”email”

## 2. ****git init****

- Là câu lệnh được sử dụng khi chúng ta bắt đầu một project mới.

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

- Cách dùng: nhập “git init” trong cmd tại thư mục làm việc.

VD: git init <repository name>

## 3. ****git clone****

- Một repository ở trên kho lưu trữ của git( github, gitlab) sẽ được tải về máy của user.

- Tác dụng: Copy 1 git repository từ remote source.

-VD: git clone <project URL>

## ****4. Git status****

- Tác dụng: cho phép ta xem tình trạng hiện của mã nguồn có bao nhiêu file được thêm mới hoặc được chỉnh sửa, cũng như cho biết những file nào đang nằm trong staging area hoặc nằm ngoài staging area.

- Cách dùng: nhập “git status” trong cmd tại thư mục làm việc.

## 5. ****git add****

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

- VD: + Thêm một file vào staging area: git add file\_name

+ Thêm tất cả các file vào staging area: git add \*

## 6. ****git commit****

- Tác dụng: Là câu lệnh được sử dụng phổ biến nhất. Khi được thực thi, câu lệnh này sẽ giúp chúng ta lưu những sự thay đổi ở các file trong staging area xuống repository. 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 vậydễ dàng khôi phục lại các thay đổi trước đó.

- VD: git commit -m “secret project”

## 7. ****git push/git pull****

- Tác dụng:

+Push: câu lệnh được sử dụng để đồng bộ, cập nhật những thay đổi trên source code(file) được commit trên local lên một dịch vụ lưu trữ

VD: git push -u <short\_name> <branch\_name>

Lưu ý: Trước khi sử dụng git push chúng ta nên cấu hình origin và upstream

Syntax: git push –set-upstream <short\_name> <branch\_name>

+Pull: Khi gọi đến câu lệnh này, tất cả những nội dung trên dịch vụ lưu trữ từ xa sẽ tải xuống local repository.

VD: git pull <remote\_url>

## 8. ****git branch****

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

- Cách dùng: nhập “git branch” trong cmd tại thư mục làm việc

## 9. ****git checkout****

- Tác dụng: Chuyển sang branch khác

- VD:

+ Hiển thị các branch đang có trong 1 repository nhất định: git branch

+ Tạo một branch mới: git branch

+ Xoá branch: git branch -d <brand\_name>

## 10. ****git stash****

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

- Cách dùng: nhập “git stash” trong cmd tại thư mục làm việc.

VD : + Xem tất cả các stash hiện có: git stash list

+ Áp dụng các thay đổi trong một stash lên branch hiện tại đang sử dụng: git stash apply hoặc git stash pop

## 11. ****git merge****

- Tác dụng: Gộp, liên kết những source code và những thay đổi trên một branch khác lên một branch đang sử dụng hiện tại.

- Cách dùng: Chuyển tới branch bạn muốn merge rồi  dùng

“git merge <:nhánh muốn merge:>”

- VD: git merge <name>

- Trong trường hợp xảy ra xung đột (conflic), người dùng phải xử lý thủ công để merge theo đúng ý của mình.

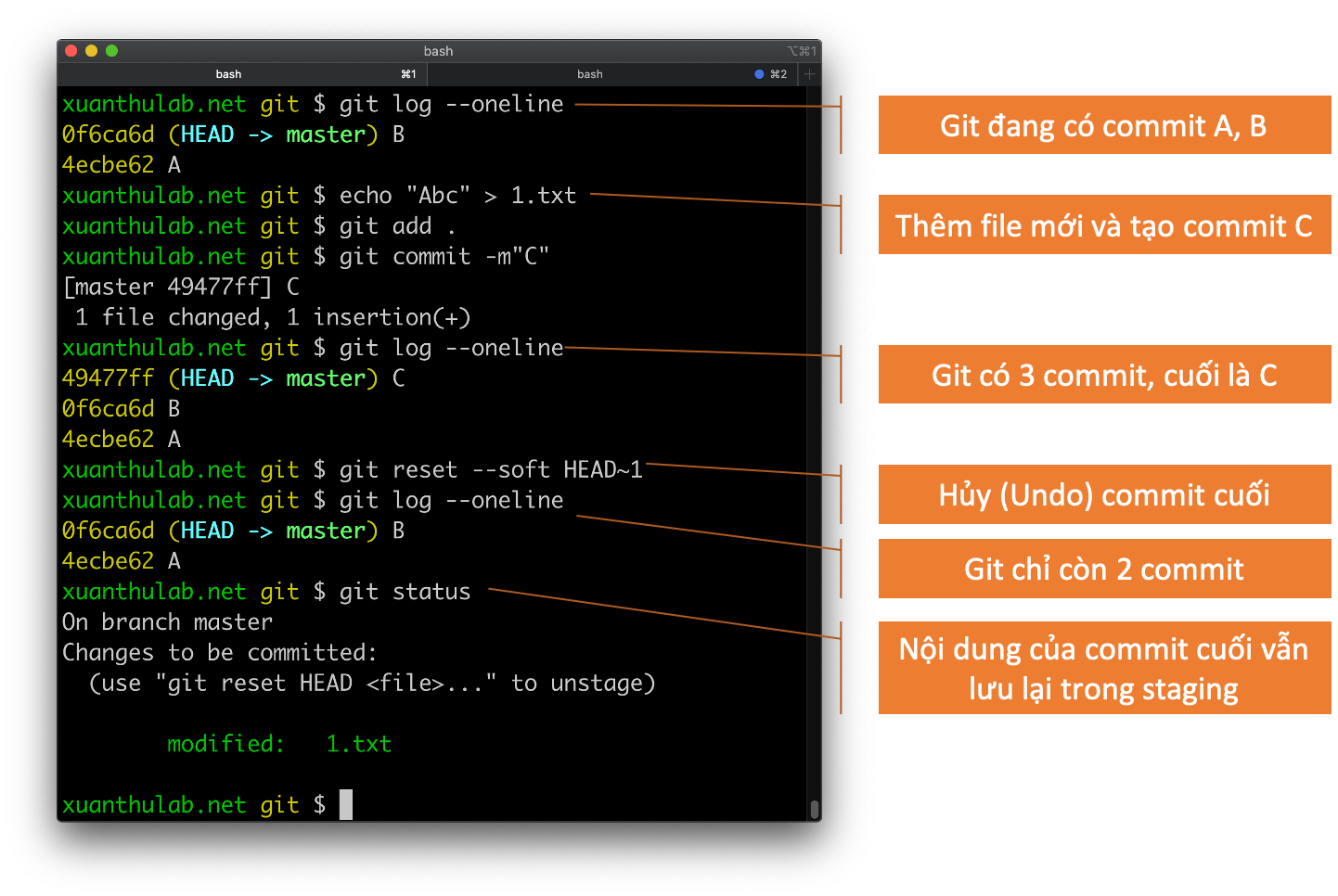
## 12. ****git reset****

- Tác dụng: Git reset được dùng để quay về một điểm commit nào đó, đồng thời xóa lịch sử của các commit trước nó.

- git reset với tham số --soft:

Trường hợp này sẽ hủy commit cuối, con trỏ HEAD sẽ chuyển về commit cha. Đồng thời những thay đổi của commit cuối được chuyển vào vùng staging nhằm để có cơ hội commit lại hoặc sửa đổi, cú pháp lệnh như sau:

git reset --soft HEAD~1



- git reset với tham số --hard”

Khi dùng tham số --hard thì kết quả giống với dùng tham số --soft, chỉ có một khác biết là nội dung thay đổi của commit cuối không đưa đưa vào staging mà bị hủy luôn. Trường hợp này dùng khi bạn quyết định hủy hoàn toàn commit cuối:

git reset --hard HEAD~1



-Một vài trường hợp dùng git reset khác:

+Hủy git add

Nếu đã dùng lệnh git add để cập nhật thay đổi vào vùng staging, có thể hủy thao tác này bằng cách thực hiện lệnh:

git reset

+Hủy đưa một file vào staging

Nếu muốn hủy một file nào đó trong vùng staging chứ không phải toàn bộ thì dùng lệnh:

git reset -- filename