## VCS VÀ GIT

### VCS - Hệ thống quản lý phiên bản

Hệ thống quản lý phiên bản - Version Control System (VCS) là một hệ thống ghi nhận và lưu lại sự thay đổi của các file theo thời gian, từ hệ thống đó một file có thể phục hồi quay về trạng thái (phiên bản) ở một thời điểm trước đó. Ngoài ra bạn có thể theo dõi sự thay đổi của một file theo thời gian, ai đã thay đổi, thay đổi vào lúc nào .... Có nhiều hệ thống VCS mà bạn có thể chọn sử dụng như: Concurrent Versions System, Subversion, Git, Mercurial

Ở đây chỉ giới thiệu hệ thống **Git** (và dịch vụ GitHub)

### Phân loại VCS

Khi bạn chưa biết đến VCS thì chắc là bạn đã từng làm việc sau: *Giả sử bạn có một thư mục chứa một số file (code, media ...), trước khi bạn sửa đổi các file đó có thể bạn sẽ copy lại thư mục và lưu lại với một cái tên đánh dấu để nhỡ cần phục hồi lại các file gốc thì có thể copy trở lại. Việc làm của bạn chính là bạn vừa vận hành một hệ thống quản lý phiên bản.*. Cách làm này đơn giản tuy nhiên gây ra nhiều lỗi như bạn có thể quyên mất thư mục ban đầu lưu trữ nhất là khó mà nhớ thời điểm lưu lại khi copy nhiều lần thư mục.

Để giúp bạn thực hiện các điều trên dễ dàng hơn, bạn nên sử dụng một hệ quản trị phiên bản (VCS) chạy trên máy, lúc này VCS sẽ tạo ra một **Cơ sở dữ liệu** lưu giữ những thay đổi của các file và giúp bạn phục hồi về một phiên bản cũ một cách dễ dàng. Hệ thống đó bạn có thể gọi là **VCS cục bộ**, mô hình hóa hoạt động như hình.

Diagram

Description automatically generated

Hình 1.1: VCS cục bộ

Khi làm việc nhóm các developer thiết lập một hệ **thống quản lý phiên bản trung tâm (CVCS)** ở một máy chủ, ở đó chứa các phiên bản của file, các lập trình viên kết nối và lấy về sửa đổi rồi lại cập nhật lại ... Tuy nhiên CVCS vẫn có nhược điểm như Server bị dừng thì không thể kết nối để các thành viên làm việc, Database ở Server bị lỗi (ví dụ do đĩa) thì khó phục hồi file đã mất ...

Diagram

Description automatically generated

Hình 1.2: VCS trung tâm

Khắc phục nhược điểm của CVCS thì một loại VCS nữa được đề xuất đó là DVCS (Distributed Version Control Systems ) - **Hệ thống quản lý phiên bản phân tán**, nó giống với CVCS là có một máy chủ ở đó có Database lưu giữ các phiên bản của file, tuy nhiên khác biệt đó là các máy khách (các developer) kết nối vào thỉ nó không chỉ lấy file mà nó lấy luôn cả hệ thống Database. Điều này có nghĩa là khi Server bị ngắt, các máy khách vẫn làm việc bình thường trên Database ở máy trạm, sau đó commit (chuyển) lên Server sau, hoặc Database ở Server bị lỗi thì máy khách bất kỳ đều cỏ thể phục hồi lại cho Server.

Diagram

Description automatically generated

Hình 1.3: VCS phân tán

### Giới thiệu hệ thống VCS Git

**Git** chính là hệ thống quản lý phiên bản phân tán (DVCS), với các ưu điểm: tốc độ, đơn giản, phân tán, phù hợp với dự án lớn nhỏ.

### Cách lưu dữ liệu của Git

**Git** lưu dữ liệu dưới dạng một loạt "ảnh chụp" (snapshot) của một tập hợp các file, có nghĩa là mỗi khi bạn commit (lưu lại) thì Git tiến hành chụp lại hệ thống các file thời điểm đó và lưu giữ một tham chiếu đến ảnh chụp dó, nhớ rằng các file không có thay đổi thì Git sẽ không lưu lại file đó lần nữa mà chỉ có một liên kết đến file đã lưu ở lần trước.

### Hầu hết mọi thao tác với Git diễn ra ở Local

Vì mỗi máy trạm có một Database Git nên hầu hết thao tác với Git như thêm mới, xem lại lịch sử ... đều không cần đến Server (trừ cần commit lên Server, lấy về một file do người khác đã cập nhật)

### Dữ liệu lưu trữ trong Git đảm bảo tính toàn vẹn

Mọi thứ trước khi được lưu trữ vào Git đều đước kiểm tra bởi mã băm (hash, checksum), có nghĩa là không thể thay đổi nội dung của file mà Git không biết về sự thay đổi đó. Chức năng này giúp cho bạn không thể mất thông tin khi trao đổi dữ liệu hay file lỗi mà không thể nhận ra được. Git sử dũng mã hash SHA-1, mỗi chuỗi hash SHA-1 sinh ra căn cứ theo nội dung của file dài 40 ký tự (tạo ra từ các ký tự trong khoảng thập lục phân : 0-9, a-f) có dạng:

62FC2DBFB0CB299DD8548286FE1BB1D2B2041379

Mã hash kiểu này bạn sẽ gặp thường xuyên khi làm việc với Git vì mọi thứ lưu trong dữ liệu đều căn cứ theo mã hash này.

### Với Git cơ bản là thêm vào nó các dữ liệu

Các hành động trong Git hầu hết là các thao tác là thêm dữ liệu vào hệ thống Git. Rất khó để yêu cầu hệ thông thực hiện việc gì đó mà không phục hồi lại được, cũng như khó mà xóa hoàn toàn dữ liệu khỏi hệ thống.

### Ba trạng thái với Git

Với Git, các file của bạn có thể nằm ở một trong ba trạng thái: committed, modified, staged

**committed** có nghĩa là dữ liệu đã lưu trữ an toàn trong Database (local)

**modified** có nghĩa là dữ liệu (file) có thay đổi nhưng chưa lưu trong Database (local)

**staged** có nghĩa đánh dấu các file sửa đổi modified chuẩn bị được commit tiếp theo

Các khái niệm trên dẫn tới Git tổ chức một dự án ra ba khu vực:

**1** Thư mục làm việc (Working tree) - Chứa bản sao phiên bản cụ thể của dự án

**2** Khu vực sắp xếp (staging) - là một file, nằm trong thư mục Git chứa thông tin sẽ commit

**3** Thư mục Git (Git directory) - nơi Git lưu Database các file Metadata

Timeline

Description automatically generated

Từ hình trên, khi làm việc với Git bạn theo tiến trình như sau:

* Bạn code, sửa đổi các file trong **1** Thư mục làm việc (Working tree)
* Bạn tổ chức những sự thay đổi nào muốn commit tiếp theo, cơ bản là việc đánh dấu sự thay đổi vào khu vực **2** Khu vực sắp xếp (staging)
* Bạn thực hiện commit để các file đánh dấu trong staged lưu vào **3** Git (database) như nhưng snapshot (chụp file)

## Cài đặt Git

### Trên Linux

Chạy lệnh ở terminal với Ubuntu

sudo apt install git-all

Với CentOS

sudo yum install git

Với macOS

brew install git

### Trên Windows

Tải Git tại [Git Download](https://git-scm.com/download/win), tiến hành chạy file tải về để cài đặt.

## Lần đầu chạy Git - Thiết lập tham số

Mặc định bạn sẽ làm việc với Git ở giao diện dòng lệnh cmd, đối với Git trên Windows mà bạn cài như trên bạn cũng có thể sử dụng Git Bash - git-bash.exe trong thư mục C:\Program Files\Git, để mở giao diện dòng lệnh tiện ích hơn là cmd mặc định của Windows. Nó có giao diện như dưới

Shape, rectangle

Description automatically generated

Bạn có thể tùy biến hiện thị của **Git Bash** bằng cách sửa file cấu hình của nó ở C:\Program Files\Git\etc\profile.d\git-prompt.sh

Các tác vụ với Git thực hiện bằng lệnh **git** với tham số thích hợp.

Lệnh đầu tiên là hãy kiểm tra Version của Git đang dùng:

git --version

Hiện thị các lệnh Git bằng cách gõ

git --help

### Cấu hình Git

Các tham số chung để bạn làm việc với Git được thiết lập với lệnh git config với vài cách khác nhau, nó thiết lập các biến tham số, các biến này có thể lưu trữ ở một số nơi tùy cách bạn thiết lập như:

* /etc/gitconfig : tham số khai báo trong file này có hiệu lực cho mọi user, dự án. Lệnh tương tác thiết lập với tham số --system
* ~/.gitconfig hoặc ~/.config/git/config (Windows: C:\Users\$USER\.gitconfig) - thiết lập có hiệu lực với User hiện tại của hệ thống, thực hiện với tham số --global
* .git/config cấu hình lưu ở chính thư mục .Git của dự án (repository), chỉ có tác dụng cho Repository này.

Áp dụng bạn có thể thiết lập các tham số chung ban đầu như sau:

**Tên trên hệ thống Git**

git config --global user.name "Tên củ**a** **B**ạn"

**Email của bạn trên Git**

git config --global user.email emailcuaban**@domain**.com

Nhớ là để xác định được danh tính của bạn cần thiết lập cả tên và email

**Hiện thị các tham số thiết lập Git**

git config --list

### Tóm tắt các lệnh đã sử dụng

# Kiểm tra phiên bản GIT

git --version

# Danh sách các thiết lập cấu hình

git config --list

# Thiết lập tên

git config --global user.name "Tên của Bạn"

# Thiết lập email

git config --global user.email emailcuaban@domain.com

Mục lục bài viết

[Mục lục Git - GitHub](https://xuanthulab.net/git-va-github/)

[ĐĂNG KÝ KÊNH, XEM CÁC VIDEO TRÊN XUANTHULAB](https://www.youtube.com/xuanthulab?sub_confirmation=1)