Giới thiệu về Git và một số tính năng cơ bản

Tác giả: Nguyễn Đức Hà

1. Sơ lược về quản lý phiên bản

Khái niệm; Quản lý phiên bản là một hệ thống lưu trữ các thay đổi của một tập tin (file) hoặc tập hợp các tập tin theo thời gian. Quản lý phiên bản không chỉ được áp dụng trong phát triển phần mềm mà còn có thể áp dụng trong nhiều lĩnh vực khác

Quản lý phiên bản cho phép:

* Xem lại lịch sử thay đổi của dự án.
* Xem thông tin của các thay đổi của dự án: tác giả, nội dung.
* Khôi phục lại phiên bản cũ.

Có 3 phương pháp quản lý phiên bản:

* Quản lý phiên bản cục bộ: người dùng sao chép các file và lưu trữ cục bộ đồng thời sử dụng một cơ sở dữ liệu đơn giản để quản lý lịch sử thay đổi của các file này. Đại diện tiêu biểu của phương pháp này là hệ thống quản lý phiên bản rcs.
* Quản lý phiên bản tập trung: hệ thống này gồm một máy chủ lưu trữ tất cả các tập tin đã được phiên bản hóa và danh sách các máy client được quyền sửa đổi các tập tin trên máy chủ. Đại diện của phương pháp này là SVN, Perforce.
* Quản lý phiên bản phân tán: trong phương pháp này, các máy khách không chỉ lấy về phiên bản mới nhất của các file mà còn sao chép toàn bộ repository. Với phương pháp này, dữ liệu ở mỗi máy khách đều có thể được dùng để phục hồi dữ liệu ở máy chủ khi gặp sự cố. Đại diện của phương pháp này là Git và Mercurial.

1. Khái niệm cơ bản về Git
2. Khái niệm

Git là một “Hệ thống quản lý phiên bản phân tán” được viết bởi Linus Torvalds. Git là phần mềm mã nguồn mở, có thể làm việc được trên nhiều hệ điều hành khác nhau: Windows, Linux, Mac OS X,…

1. Đặc điểm của Git

* Cách xử lý dữ liệu:

Git coi dữ liệu của nó là một tập các ảnh (snapshot) của hệ thống tập tin. Điều này có nghĩa là mỗi phiên bản của dự án (có thể hiểu là một commit) sẽ là tập hợp của một số ảnh lưu lại nội dung của các tập tin của phiên bản đó.

Điều này mang đến nhiều tiện lợi cho việc theo dõi lịch sử, phục hồi dữ liệu và phân nhánh.

* Thao tác với dữ liệu:

Hầu hết các thao tác với dữ liệu của Git có thể thực hiện cục bộ. Git thực hiện được việc này vì toàn bộ dữ liệu của dự án đều được lấy về và lưu trữ trên máy tính của người dùng.

Với tính năng này của Git, người dùng có thể làm việc trong nhiều trường hợp mà không nhất thiết phải có kết nối Internet. Điều này mang đến nhiều lợi thế cho Git so với các hệ thống quản lý dữ liệu khác.

* Tính toàn vẹn:

Các thay đổi trong Git được tham chiếu bằng một mã băm sử dụng cơ chế mã hóa SHA-1. Đồng thời, các thay đổi trong Git đều được thêm vào cơ sở dữ liệu do đó rất khó bị mất khi thay đổi và truyền tải dữ liệu. Với Git, người dùng có thể thoải mái thử nghiệm, lưu trữ mà không sợ ảnh hưởng đến dự án.

1. Tổ chức dữ liệu trong Git

Các tập tin trong Git tồn tại ở một trong ba trạng thái: modified, staged, committed.

* Modified: tập tin được sửa nhưng chưa được đánh dấu để commit (nằm trong mục Unstaged files).
* Staged: tập tin được đánh dấu sẽ được commit (nằm trong mục Staged files).
* Committed: tập tin đã được commit và lưu trữ trong cơ sở dữ liệu.

Một tập tin có thể vừa ở trạng thái modified vừa ở trạng thái staged hoặc committed. Điều này xảy ra khi người dùng chỉ staged một số dòng trong tập tin.

Ba trạng thái này tạo ra ba phần riêng biệt của dự án:

* Working directory: là bản sao của một phiên bản của dự án. Người dùng sẽ làm việc với các tập tin ở khu vực này. Mọi thay đổi của các tập tin sẽ được hiển thị ở đây.
* Staging area: là một tập tin trong thư mục Git. Tập tin này chứa thông tin về những thay đổi sẽ được commit.
* Git directory: đây là nơi Git lưu trữ các siêu dữ liệu (metadata) và cơ sở dữ liệu của toàn bộ dự án. Đây cũng là phần quan trọng nhất của Git, nó là một bản sao của dự án được sao chép (clone) từ repository trên server Git.

1. Làm việc với Git
2. Khởi tạo

* Khởi tạo repository từ thư mục cũ:

Để khởi tạo một repository từ thư mục cũ, cần thêm thư mục “.git” vào thư mục đó bằng cách chạy lệnh:

git init

* Sao chép một repository đã tồn tại:

Để sao chép từ một repository đã tồn tại cần sử dụng lệnh:

git clone [url] [folder name]

Trong đó:

url là đường dẫn đến repository. Đường dẫn này tùy thuộc vào giao thức truyền tải mà server cung cấp. Một số loại giao thức thường dùng:

* + Giao thức git://
  + Giao thức http hoặc https
  + Giao thức SSH: user@server:/path.git

folder name là tên folder ở máy cục bộ mà repository sẽ được sao chép về. Tham số này là không bắt buộc, nếu không dùng tham số này thì repository sẽ được sao chép vào thư mục mặc định có tên như tên repository.

1. Lưu trữ

Trước khi tìm hiểu cách lưu trữ các tập tin trong thư mục Git, chúng ta cần hiểu về vòng đời của các tập tin này. Vòng đời của một tập tin gồm có 4 trạng thái: untracked, unmodified, modified, staged. Sơ đồ vòng đời của một tập tin được mô tả trong hình dưới đây.

* untracked: là trạng thái của tập tin mà không có trong bất kỳ ảnh nào trước đó.
* unmodified: là trạng thái của tập tin chưa được sửa gì so với ảnh cuối cùng.
* modified: là trạng thái của tập tin đã được sửa so với ảnh cuối cùng.
* staged: là trạng thái của tập tin đã được đánh dấu và chuyển vào staging area.

Bỏ qua các tập tin: việc này khá cần thiết trong dự án khi có một số loại tập tin mà người dùng không muốn theo dõi. Những tập tin này có thể là các tập tin được tạo ra tự động, các tập tin tạm thời hay các tập tin được tạo ra khi biên dịch. Để bỏ qua các tập tin này có thể tạo ra một tập tin có tên là .gitignore và liệt kê tên các tập tin hoặc các mẫu tên tập tin muốn bỏ qua. Các mẫu này có dạng như các regular expression được dùng khi kiểm tra các xâu trong code.

Ví dụ:

\*.[abc] : bỏ qua các tập tin có đuôi là .a hoặc .b hoặc .c

\*.~ : bỏ qua các tập tin có đuôi là ~

\*.[0-9] : bỏ qua các tập tin có đuôi nằm trong khoảng từ 0 đến 9

1. Phục hồi
2. Git branch
3. Khái niệm nhánh
4. Quản lý nhánh (tạo, xóa, chuyển nhánh)
5. Quy trình làm việc với nhánh: nhánh lâu đời (Long-Running Branches) và nhánh chủ đề (Topic Branches)
6. Tích hợp nhánh: Merge và Rebase
7. Nhánh từ xa (Remote branches)
8. Các câu lệnh thường dùng