Cơ bản về Git

VCS – hệ thống quản lý phiên bản

Version Control System (VCS) là một phần mềm mà giúp các nhà phát triển phần mềm làm việc cùng nhau và duy trì một lịch sử đầy đủ các công việc mà họ đã làm.

Dưới đây là các chức năng của một VCS:

- Cho phép các nhà phát triển phần mềm cùng làm việc với nhau
- Không cho phép ghi đè lên các thay đổi của nhau
- Duy trì một lịch sử của mọi phiên bản.

Dưới đây là các loại VCS:

- Hệ thống kiểm soát phiên bản tập trung (CVCS).
- Hệ thống kiểm soát phiên bản phân phối/phân cấp (DVCS).

Trong chương này chúng ta sẽ chỉ tập trung vào hệ thống quản lý phiên bản phân phối và đặc biệt trên Git.

Hệ thống kiểm soát phiên bản phân phối

Hệ thống kiểm soát phiên bản tập trung (CVCS) sử dụng một máy chủ để lưu giữ tất cả các file và cho phép các team cộng tác với nhau. Nhưng nhược điểm lớn nhất của CVCS cũng là điểm thất bại của nó, tức là, sự thất bại của các máy chủ trung tâm. Thật không may là, nếu máy chủ trung tâm bị hỏng trong một giờ, thì trong suốt quãng thời gian đó không ai có thể cộng tác được với ai cả. Và ngay cả trong trường hợp xấu nhất, nếu đĩa của máy chủ trung tâm bị hỏng và sự sao lưu không được thực hiện, bạn sẽ mất toàn bộ lịch sử của dự án. Tại đây, hệ thống quản lý phiên bản phân phối xuất hiện.

Các client DVCS không chỉ kiểm tra được các ảnh chụp mới nhất của các thư mục mà họ còn quan sát được tất cả repository trữ của dự án. Nếu server bị hỏng, các kho dự trữ của các client có thể sao một bản sao đầy đủ cho server để khôi phục lại nó. Git không phụ thuộc vào server trung tâm và đó là lý do tại sao bạn có thể thực hiện nhiều thao tác khi bạn đang offline. Bạn có thể ủy thác các thay đổi, tạo các nhánh, xem các bản ghi và thực hiện các hoạt động khác khi bạn đang offline. Bạn cần kết nối mạng chỉ để công bố những thay đổi của bạn và đưa những thay đổi mới nhất vào dự án.

Các lợi thế của Git

Nguồn miễn phí và mở

Git được công bố dưới giấy phép nguồn mở của GPL. Nó có sẵn miễn phí trên mạng. Bạn có thể sử dụng Git để quản lý các dự án thích hợp mà không phải trả bất kỳ đồng nào. Như là một nguồn mở, bạn có thể tải mã nguồn của nó và cũng có thể thực hiện các thay đổi theo yêu cầu của bạn.

Tốc độ nhanh và nhỏ gọn

Khi hầu hết các thao tác được thực hiện trong nội bộ, nó mang lại lợi ích rất lớn về tốc độ. Git không phụ thuộc vào server, đó là lý do tại sao mà không cần sự tương tác với server từ xa cho mọi thao tác. Phần cốt lõi của Git được viết bằng C, mà có thể tránh được các chi phí liên quan đến thời gian chạy với ngôn ngữ bậc cao khác. Mặc dù Git phản ánh toàn bộ repository trữ, kích thước của các dữ liệu trên các client là nhỏ. Điều này cho thấy sự hiệu quả của Git trong việc nén và lưu trữ dữ liệu trên các client.

Dự phòng (sao lưu) ẩn

Việc mất dữ liệu là hiếm khi xảy ra khi mà có rất nhiều bản sao của nó. Dữ liệu hiện diện ở bất kỳ client nào, do đó nó có thể được sử dụng trong trường hợp hỏng hoặc ngừng ở server.

An toàn cao

Git sử dụng một hàm băm (hash funtion) mật mã chung được gọi là hàm băm an toàn (SHA1), để đặt tên và xác định các đối tượng trong cơ sở dữ liệu của nó. Mỗi tập tin và commit được kiểm tra tóm tắt và thu được kết quả tại thời gian kiểm tra. Điều này ngụ ý rằng, nó không thể thay đổi tập tin, ngày tháng và thông báo commit và bất kỳ dữ liệu khác từ cơ sở dữ liệu Git mà không hiểu biết về Git.

Không yêu cầu một phần cứng mạnh

Trong trường hợp CVCS, server trung tâm cần đủ mạnh để phục vụ các yêu cầu của toàn team. Đối với những team nhỏ, nó không phải là một vấn đề, nhưng khi kích thước team phát triển, thì những hạn chế của phần cứng server có thể làm hiệu suất của công việc thay đổi theo hướng tiêu cực. Trong trường hợp DVCS, các nhà phát triển không tương tác với server trừ khi họ cần công bố những thay đổi đã thực hiện. Tất cả các việc đều diễn ra trên các client, vì thế phần cứng của server có thể thực sự là vấn đề đơn giản.

Phân nhánh dễ dàng hơn

CVCS sử dụng kỹ thuật sao chép rẻ, nếu chúng ta tạo ra một nhánh mới, nó sẽ sao chép tất cả các code tới nhánh mới, vì thế nó tốn thời gian, không hiệu suất. Ngoài ta, việc xóa và sáp nhập của

các nhánh trong CVCS là phức tạp và tốn thời gian. Nhưng quản lý nhánh với Git là rất đơn giản. Nó chỉ mất một vài giây để tạo, xoa, và nhập các nhánh.

Các thuật ngữ của DVCS

Kho commit nội bộ

Mỗi công cụ VCS cung cấp một nơi làm việc riêng như là bản sao làm việc. Các nhà phát triển đã thực hiện những thay đổi trong nơi làm việc riêng và sau khi commit của họ, những thay đổi trở thành một phần của kho. Git có một bước tiến xa hơn bằng cách cung cấp cho họ một bản sao riêng của toàn bộ kho. Người sử dụng có thể thực hiện nhiều thao tác với kho này như thêm, di chuyển, đổi tên file, commit thay đổi và nhiều thao tác khác.

Thư mục làm việc và Staging hoặc Index

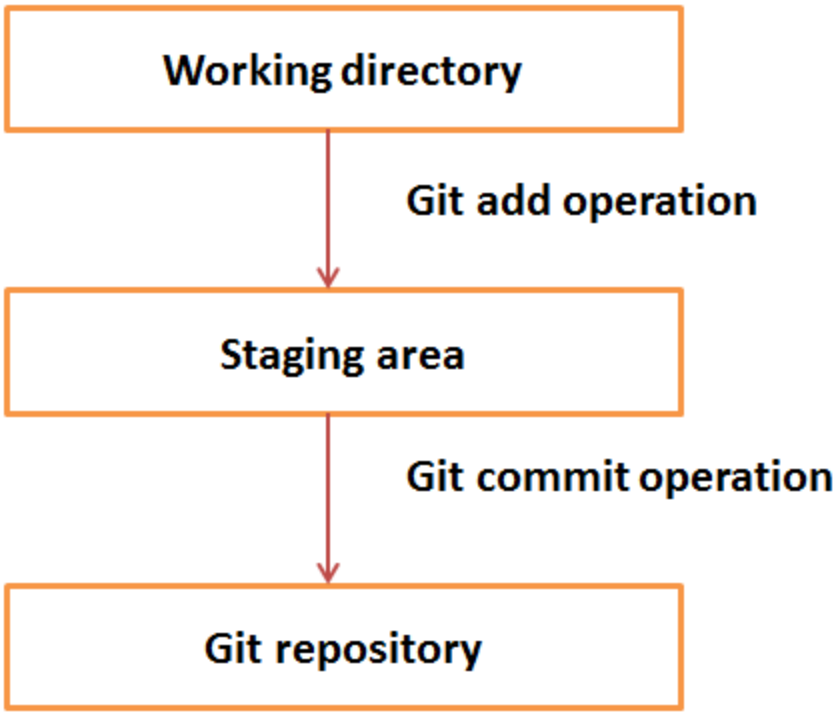
Thư mục làm việc là nơi các file được kiểm tra. Trong CVCS, các nhà phát triển thường tạo các thay đổi và commit các thay đổi của họ một cách trực tiếp tới kho chứa. Nhưng Git sử dụng một phương thức khác. Git không theo dõi từng file sửa đổi. Bất cứ khi nào bạn thự hiện commit một thao tác, Git tìm kiếm file trong khu vực tổ chức (staging area). Chỉ khi nào những file có mặt trong khu vực tổ chức này được xem xét để commit mà không phải tất cả các file sửa đổi.

Sau đây là tiến trình làm việc cơ bản của Git:

Bước 1 : Bạn sửa đổi một file từ thư mục làm việc

Bước 2 : Bạn thêm file đó vào khu vực tổ chức

Bước 3 : Bạn thực hiện các hoạt động commit mà di chuyển các file từ khu vực tổ chức. Sau thao tác đẩy (push), nó lưu các thay đổi cố định tới kho chứa Git.



Giả sử bạn sửa đổi 2 file, tên là sort.c và search.c và bạn muốn thực hiện hai commit khác nhau cho mỗi hoạt động. Bạn có thể thêm một file vào khu vực tổ chức và thực hiện commit. Sau hoạt động commit đầu tiên, làm lại theo phương thức tương tự cho file còn lại.

```
# First commit
[bash]$ git add sort.c

# adds file to the staging area
[bash]$ git commit –m “Added sort operation”

# Second commit
[bash]$ git add search.c

# adds file to the staging area
[bash]$ git commit –m “Added search operation”
```

Blobs

Blob là viết tắt của **B**inary **L**arge **Ob**ject. Mỗi phiên bản của một file được đại diện bởi blob. Một blob chứa dữ liệu file nhưng không chứa bất kỳ siêu dữ liệu nào về file. Nó là một tập tin nhị phân,

và trong cơ sở dữ liệu Git, nó được đặt trên là SHA1 hash của file đó. Trong Git, các file không được đặt bằng tên. Tất cả mọi thứ được đặt địa chỉ theo nội dung.

Cây - Trees

Cây (Tree) là một đối tượng, mà biểu diễn một thư mục. Nó giữ các blob cũng như các thư mục phụ khác. Một cây là một file nhị phân mà giữ các thứ liên quan đến blob và các cây cũng được đặt tên là SHA1 hash của đối tượng cây.

Ký thác - Commits

Hoạt động commit giữ trạng thái hiện tại của repository. Một commit cũng được đặt tên là SHA1 hash. Bạn có thể xem xét một đối tượng commit như là một nút của danh sách liên kết. Mỗi đối tượng commit có một điểm con trỏ tới đối tượng commit gốc. Từ một commit đã cho, bạn có thể truy xét trở lại bằng cách nhìn vào điểm con trỏ gốc để xem lịch sử của commit đó. Nếu một commit có nhiều commit gốc, thì khi đó các commit cụ thể sẽ được tạo bởi cách sáp nhập hai nhánh.

Các nhánh - Branches

Các nhánh được sử dụng để tạo ra các tuyến khác của sự phát triển. Theo mặc định, Git có một nhánh chủ, mà tương tự như thân (trunk) trong Subversion. Thông thường, một nhánh được tạo để làm việc về một điểm mới. Một khi điểm này được hoàn thành, nó được sáp nhập lại với nhánh chủ và chúng ta xóa nhánh đó đi. Mỗi nhánh được ám chỉ bởi HEAD, mà các điểm con trỏ tới commit mới nhất trong nhánh. Bất cứ khi nào bạn thực hiện một commit, HEAD được cập nhật bởi các commit mới nhất đó.

Thẻ - Tags

Các thẻ chỉ một tên có nghĩa với một phiên bản xác định trong kho chứa. Các thể là tương tự như các nhánh, nhưng sự khác nhau là các thẻ không thay đổi được. Nó có nghĩa là, thẻ là một nhánh, mà không ai có ý định sửa chúng. Một khi một thẻ được tạo ra cho các commit cụ thể, ngay cả khi bạn tạo một commit mới, nó sẽ không được cập nhật. Thông thường, các nhà phát triển tạo các thẻ cho công bố sản phẩm.

Mô phỏng - Clone

Hoạt động mô phỏng tạo bản sao của repository. Hoạt động này không chỉ kiểm tra việc sao chép, mà còn phản ánh kho toàn bộ repository. Người sử dụng có thể thực hiện rất nhiều thao tác với repository nội bộ. Các mạng thời gian chỉ được tham gia khi các repository instance đang được đồng bộ.

Pull

Hoạt động pull sao chép những thay đổi từ một repository instance xa tới kho nội bộ. Hoạt động này được sử dụng để đồng bộ giữa hai repository instance. Điều này tương tự như hoạt động cập nhật trong Subversion.

Push

Thao tác đẩy (push) sao chép các thay đổi từ repository nội bộ tới một kho xa. Nó được sử dụng để lưu các thay đổi vĩnh viễn trong repository Git. Nó tương tự như hoạt động commit trong Subversion.

HEAD

HEAD là một điểm con trỏ, mà thường trỏ vào commit mới nhất trong nhánh. Bất cứ khi nào bạn thực hiện một commit, HEAD được cập nhật với commit mới nhất đó. Đầu của các nhánh được lưu trong **.git/refs/heads/**directory.

```
[CentOS]$ ls -1 .git/refs/heads/
master

[CentOS]$ cat .git/refs/heads/master
570837e7d58fa4bccd86cb575d884502188b0c49
```

Revision

Revision đại diện cho phiên bản của một mã nguồn. Revision trong Git được đại diện bởi các commit. Những commit này được xác định bởi SHA1 secure hash.

URL

URL đại diện cho vị trí của repository Git. Git URL được giữ trong một tệp config.

```
[tom@CentOS tom_repo]$ pwd
/home/tom/tom_repo

[tom@CentOS tom_repo]$ cat .git/config
[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
[remote "origin"]
```

```
url = gituser@git.server.com:project.git
fetch = +refs/heads/*:refs/remotes/origin/*
```