

## GIT : Các Thủ Thuật Cơ Bản

*Gợi ý thì u: GIT hiện đã có sẵn để dùng như u để làm công cụ quản lý mã nguồn và version để thay cho SVN, như t là các để quản lý mã nguồn, vì vậy xin gợi ý thì u sử dụng nó đây.*

Thay vì lao vào các mã lệnh phức tạp của Git, bạn hãy sử dụng các ví dụ cơ bản để bắt đầu. Mặc dù chúng rất đơn giản, nhưng tất cả chúng đều rất hữu dụng. Quyết định là vậy, trong tháng đầu tiên sử dụng Git tôi chưa bao giờ vượt qua những gì nói trong chương này.

### Ghi Nhớ Trạng thái

Bạn muốn nhớ các lệnh mà bạn đã sử dụng gì đó với Git? Trừ khi bạn làm điều đó, các lệnh khác nhau sau trong thế giới của bạn hành động các mã nguồn hay vẫn bạn mà bạn muốn quản lý:

```
$ git init $ git add . $ git commit -m "Bạn sao đầu tiên"
```

Bây giờ nếu bạn các sự cố về vấn đề của bạn không như mong đợi, hãy phục hồi lại bạn cũ:

```
$ git reset --hard # Đột lại trạng thái và đưa lại về commit cuối
```

Sau đó sẽ có nội dung cho đúng ý bạn rồi ghi lại thành một trạng thái mới:

```
$ git commit -a -m "Bạn sao lần khác" Thêm, Xóa, Đặt Tên
```

Liên kết trên cho bạn để biết các tệp tin hiện tại của bạn để thêm vào `git add`. Nếu bạn thêm các tệp tin hay thư mục, thì bạn sẽ phải thông báo với Git:

```
$ git add readme.txt Documentation
```

Tất cả những gì bạn cần, nếu bạn muốn Git bỏ đi các tệp tin nào đó:

```
$ git rm kludge.h obsolete.c $ git rm -r incriminating/evidence/ #gõ bỏ một cách đơn giản
```

Git xóa bỏ những tệp tin nếu bạn chưa làm.

Đặt tên tệp tin thì cũng giống như là việc bạn gõ tên cũ và đặt vào nó cái tên mới. Sử dụng

## Các lệnh GIT cơ bản

Written by Administrator

Saturday, 06 October 2012 07:49 - Last Updated Saturday, 06 October 2012 10:54

---

Đúng lệnh **git mv** có cú pháp như thế nào? Lệnh **mv** của hệ thống Linux. Ví dụ:

```
$ git mv bug.c feature.c
```

### Chức Năng

### Undo/Redo

Đôi khi bạn cần muốn quay trở lại và bỏ đi những thay đổi trong quá khứ thì mất thời gian nào đó bởi vì chúng ta đã sai. Thì làm như:

```
$ git log
```

Để hiển thị cho bạn danh sách các lần commit gần đây cùng với giá trị băm SHA1:

```
commit 766f9881690d240ba334153047649b8b8f11c664 Author: Bob <bob@example.com>  
Date: Tue Mar 14 01:59:26 2000 -0800 Replace printf() with write(). commit  
82f5ea346a2e651544956a8653c0f58dc151275c Author: Alice <alice@example.com> Date:  
Thu Jan 1 00:00:00 1970 +0000 Initial commit.
```

Chỉ vài ký tự của giá trị băm là đủ để chọn ra một commit cụ thể; một cách khác là chép và dán giá trị băm. Gõ:

```
$ git reset --hard 766f
```

Để phục hồi lại trạng thái đã được chọn ra và xóa bỏ tất cả các lần commit mới hơn kể từ đó.

Một lúc nào đó bạn lại muốn nhìn lại một bản cũ hơn. Trong trường hợp này thì gõ:

```
$ git checkout 82f5
```

Nó giúp bạn quay lại đúng thời điểm đó, trong khi vẫn giữ lại những lần commit mới hơn. Tuy nhiên, gõ như thế này có thể mất thời gian trong các bộ phim khoa học viễn tưởng, nếu bạn gõ lệnh sau đó commit, bạn sẽ ở trong một thế giới khác, bởi vì hành động của bạn bây giờ đã khác với khi chúng ta lần đầu tiên ở đây.

Có cách thực hiện là sử dụng *branch*, và [chúng ta có nhu cầu đi sâu nói về nó sau này](#). Bây giờ, chúng ta cần nhớ là:

```
$ git checkout master
```

Chúng mang chúng ta trở về hiển thị. Ngoài ra, để tránh rủi ro khi sử dụng Git, thì luôn luôn commit hay reset các thay đổi của bạn trước khi chọn lệnh checkout.

Sẽ tiếp tục đang viết game trên máy tính:

- **git reset --hard:** lấy cái cũ đã được lưu lại và xóa tất cả các games mới hiện tại về lại.

- **git checkout:** lấy một cái cũ, nhưng chỉ chỉ về nó, trạng thái của game sẽ tách riêng về phía mới hiện tại mà bạn đã ghi lại lên đầu tiên. Bất kể game nào bạn tạo ra bây giờ sẽ là bạn cũ cùng trong nhánh riêng rồi tiếp tục viết mới các tệp khác mà bạn đã gia nhập vào. [chúng tôi sẽ nói sau](#).

Bạn có thể chọn chỉ nhập các tệp tin hay thậm chí bạn muốn bằng cách thêm vào chúng vào phần sau của câu lệnh:

```
$ git checkout 82f5 some.file another.file
```

Bạn phải cẩn thận khi sử dụng các lệnh, như là lệnh **checkout** có thể âm thầm ghi đè lên các tệp tin. Để ngăn ngừa rủi ro như thế, hãy commit trước khi chuyển lệnh checkout, như là khi mới học sử dụng Git. Tóm lại, bất kể khi nào bạn không chắc chắn về một lệnh nào đó, dù có là lệnh của Git hay không, đầu tiên hãy chuyển lệnh **git commit -a**

Bạn không thích việc commit dần ? Hãy sử dụng:

```
$ git checkout :/"My first b"
```

đơn giản chỉ cần commit mà phần chú thích của nó bạn để về ý chú ý bạn đã cho. Bạn cũng có thể yêu cầu trạng thái 5 tệp cái cuối cùng:

```
$ git checkout master~5 Sẽ quay lại
```

Trong một phiên tòa, mới sử dụng kiểm soát phiên viết mới bạn ghi. Cũng giống thế, bạn có thể chỉ cần lệnh commit để undo.

```
$ git commit -a $ git revert 1b6d
```

Sẽ chỉ undo lệnh commit về giá trị bạn đã chọn ra. Sẽ quay trở lại để ghi nhận như là một lệnh commit mới, bạn có thể xác nhận lại đi này bằng lệnh **git log**.

### Tạo Nhật Ký các thay đổi

Một số dự án yêu cầu có một [changelog](#). Tạo một cái bằng cách gõ:

```
$ git log > ThayĐổi.txt
```

Lấy và cài đặt bản sao của một dự án quản lý bằng Git bằng cách gõ:

```
$ git clone git://server/path/to/files
```

Ví dụ, để lấy tất cả các tệp tin mà tôi đã dùng để tạo ra cho quyển sách này là:

```
$ git clone git://git.or.cz/gitmagic.git
```

Chúng ta sẽ có nhu cầu đi sâu nói về lệnh **clone** sớm thôi.

### Thực Nghiệm

Nếu bạn đã tải và cài đặt bản sao của một dự án bằng lệnh **git clone**, bạn có thể lấy và phiên bản cũ cùng với lệnh:

```
$ git pull
```

### Xuất Bản

Giống bản đã tạo để một kho Git và bạn muốn chia sẻ nó với người khác. Bạn có thể bỏ hờ đi và để máy tính của mình, nhưng nếu bạn làm như thế trong khi bạn đang cài đặt nó hay có những thay đổi mang tính thử nghiệm, bạn có thể gặp rắc rối. Dĩ nhiên, đây là lý do tại sao mà chu kỳ phát hành phần mềm là tốt nhất là không nào. Nhưng người phát triển có thể làm việc thông xuyên trên một dự án, nhưng họ chỉ xuất bản những đoạn mã mà họ cảm thấy nó có thể dùng được để tránh những hỏng hóc của người khác.

Thực hiện đi sâu vào Git, trong thời gian làm việc của Git:

```
$ git init $ git add . $ git commit -m "Bản phát hành đầu tiên"
```

Sau đó nói với những người cùng sự nghiệp hãy chờ:

```
$ git clone your.computer:/path/to/script
```

## Các lệnh GIT cơ bản

Written by Administrator

Saturday, 06 October 2012 07:49 - Last Updated Saturday, 06 October 2012 10:54

---

đôi khi để lưu trữ. Giải pháp là truy cập thông qua ssh. Nếu không, hãy **yggit daemon** và nói với người sử dụng là chèn lệnh sau để thay thế:

```
$ git clone git://your.computer/path/to/script
```

Khi thì lúc này, bất cứ khi nào mã nguồn của bạn đã có thể sẽ được đẩy lên, chỉ việc thực hiện:

```
$ git commit -a -m "Bạn phát hành tiếp"
```

và những người sử dụng có thể cập nhật dữ liệu của họ bằng cách chuyển đổi theo mức làm việc tiếp theo và gõ:

```
$ git pull
```

Những người sử dụng sẽ không bao giờ thấy được dữ liệu cũ cùng của bạn mà bạn không muốn họ thấy.

## Tôi Đã Làm Được Gì?

Tìm tất cả các thay đổi kể từ lần bạn commit lần cuối bằng lệnh:

```
$ git diff
```

Hay thì hôm qua:

```
$ git diff "@{yesterday}"
```

Hay giữa một bản nào đó và bản trước đây 2 bản:

```
$ git diff 1b6d "master~2"
```

Trong trường hợp hời hợt, đưa ra là một minh chứng mà nó có thể được sử dụng với **git**

**apply**

Cũng có thể dùng lệnh:

```
$ git whatchanged --since="2 weeks ago"
```

Thường thì, tôi duy trì một số bản [ggit](#) để thay thế cách trên, bởi vì nó có giao diện để **h** **a** **bóng** **b** **y**, hay [tig](#), có giao diện dòng lệnh làm việc rất tốt với các máy có kết nối mạng chậm. Một lựa chọn khác là cài đặt máy chủ **web**, chèn lệnh **git**

**instaweb**

và sẽ được bắt đầu trình duyệt web nào.

## Bài Tập

Coi A, B, C, D là 4 lần commit thành công, nối tiếp nhau mà B ghi đè A ngược lại trong một số tệp tin bị xóa bỏ. Chúng ta muốn thêm các tệp tin đó trở lại D. Chúng ta thực hiện điều này bằng cách nào?

Ở đây chúng ta có ít nhất 3 giải pháp. Giải thích chúng ta đang ở D:

1.

Số khác nhau giữa A và B là việc các tệp tin đã bị ghi đè. Chúng ta có thể tìm kiếm và áp dụng nó:

```
$ git diff B A | git apply
```

2.

Kiểm tra sau khi chúng ta ghi lại các tệp tin từ A trở đi, chúng ta có thể lấy lại:

```
$ git checkout A foo.c bar.h
```

3.

Chúng ta có thể xem số di chuyển từ A tới B ghi đè như là một thay đổi mà chúng ta muốn undo:

```
$ git revert B
```

Lựa chọn nào là tốt nhất? Cách nào bạn thích nhất. Tất nhiên dùng để có được các tệp tin mà bạn muốn với Git, và thông thường là có nhiều hơn một cách để thực hiện được một thao tác bạn muốn.

NGUỒN :

## Các lệnh GIT cơ bản

Written by Administrator

Saturday, 06 October 2012 07:49 - Last Updated Saturday, 06 October 2012 10:54

---

<http://vnwildman.users.sourceforge.net/gitmagic/ch02.html>