Quản lý nhánh trong Git

Hoạt động nhánh cho phép tạo các tuyến khác nhau của sự phát triển. Chúng ta có thể sử dụng hoạt động này để phân nhánh tiến trình phát triển vào hai hướng khác nhau. Ví dụ, chúng tôi công bố một sản phẩm phiên bản 6 và chúng tôi muốn tạo ra một nhánh để phát triển các tính năng 7.0 mà có thể được giữ cách biệt với sự sửa lỗi trong phiên bản 6.0.

Tạo nhánh

Tom tạo một nhánh mới bằng cách sử dụng lệnh git branch . Chúng ta có thể tạo một nhánh mới từ một nhánh đã tồn tại. Chúng ta có thể sử dụng một commit hoặc một thẻ cụ thể như là điểm bắt đầu. Nếu bất kỳ ID commit cụ thể nào không được cung cấp, thì khi đó nhánh sẽ được tạo ra với HEAD như là điểm bắt đầu.

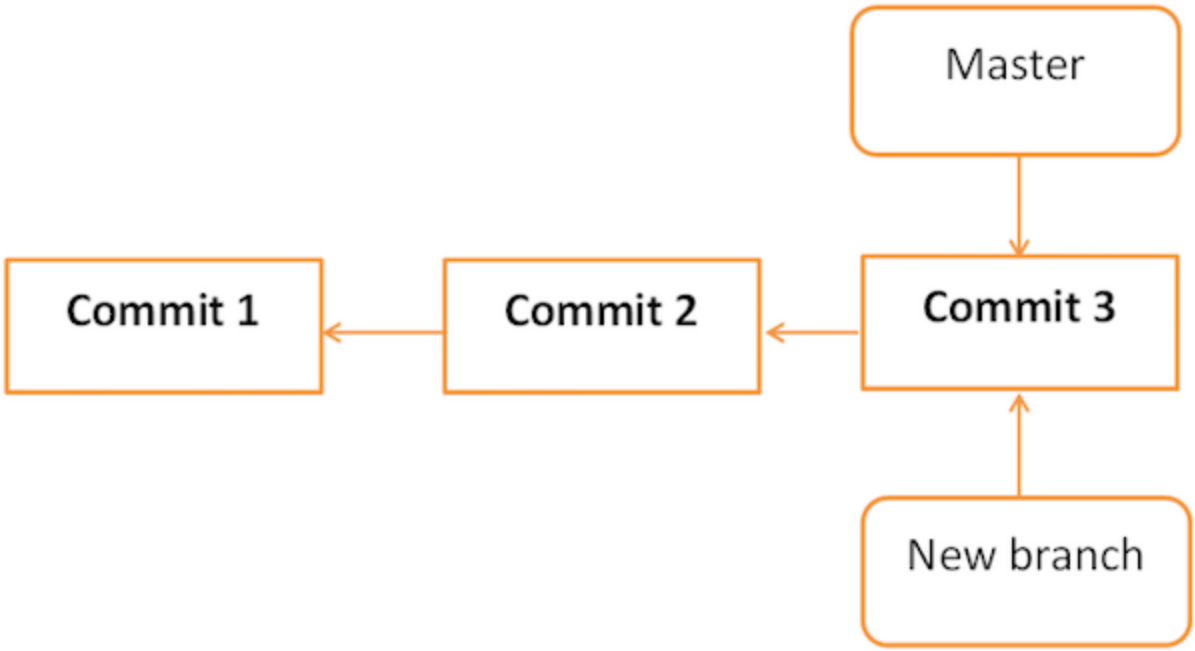
```
[jerry@CentOS src]$ git branch new_branch

[jerry@CentOS src]$ git branch
* master
new_branch
```

Một nhánh mới được tạo ra; Tom sử dụng lệnh git branch để liệt kê các nhánh có sẵn. Git chỉ một dấu hoa thị trước khi kiểm tra nhánh hiện tại.

Hình dưới đây miêu tả hoạt động tạo nhánh:

Before create branch command



After create branch operation

Chuyển đổi giữa các nhánh

Jerry sử dụng lệnh git checkout để chuyển đổi giữa các nhánh:

```
[jerry@CentOS src]$ git checkout new_branch
Switched to branch 'new_branch'
[jerry@CentOS src]$ git branch
master
* new_branch
```

Cách tắt để tạo nhánh và chuyển đổi giữa các nhánh

Ở ví dụ trên, chúng ta đã sử dụng hai lệnh riêng rẽ để tạo và chuyển đổi giữa các nhánh. Git cung cấp tùy chọn -b với lệnh checkout; hoạt động này tạo một nhánh mới và ngay lập tức chuyển đổi đến nhánh mới.

```
[jerry@CentOS src]$ git checkout -b test_branch
Switched to a new branch 'test_branch'

[jerry@CentOS src]$ git branch
master
new_branch
* test_branch
```

Xóa một nhánh

Một nhánh có thể được xóa bằng cách sử dụng tùy chọn -D với lệnh git branch. Nhưng trước khi xóa một nhánh đang tồn tại, bạn chuyển tới nhánh khác.

Jerry hiện tại đang trên nhánh test_branch và anh ta muốn dỡ bỏ nhánh đó. Vì thế anh ta chuyển sang nhánh khác và xóa nhánh như dưới đây:

```
[jerry@CentOS src]$ git branch
master
new_branch
* test_branch

[jerry@CentOS src]$ git checkout master
Switched to branch 'master'

[jerry@CentOS src]$ git branch -D test_branch
Deleted branch test_branch (was 5776472).
```

Bây giờ, Git sẽ chỉ có hai nhánh.

```
[jerry@CentOS src]$ git branch
* master
new_branch
```

Đặt lại tên cho một nhánh

Jerry quyết định thêm sự hỗ trợ cho các ký tự rộng rãi trong dự án các chuỗi hoạt động của anh ta. Anh ta đã tạo một nhánh mới, nhưng tên nhánh không được cung cấp chính xác. Vì thế anh ta thay đổi tên nhánh bằng cách sử dụng tùy chọn -m theo sau bởi old branch name và new branch name.

```
[jerry@CentOS src]$ git branch
* master
new_branch

[jerry@CentOS src]$ git branch -m new_branch wchar_support
```

Bây giờ, lệnh git branch sẽ chỉ tên nhánh mới.

```
[jerry@CentOS src]$ git branch
* master
wchar_support
```

Sáp nhập hai nhánh

Jerry thực hiện một chức năng để trả lại độ dài chuỗi của một chuỗi ký tự rộng. Một code mới sẽ xuất hiện như dưới đây:

```
[jerry@CentOS src]$ git branch
master
* wchar_support

[jerry@CentOS src]$ pwd
/home/jerry/jerry_repo/project/src

[jerry@CentOS src]$ git diff
```

Lệnh trên sẽ tạo ra kết quả sau:

```
t a/src/string_operations.c b/src/string_operations.c
index 8ab7f42..8fb4b00 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,4 +1,14 @@
#include <stdio.h>
+#include <wchar.h>
```

```
+
+size_t w_strlen(const wchar_t *s)
+
{
   +
   const wchar_t *p = s;
   +
   +
   while (*p)
   + ++p;
   + return (p - s);
   +
}
```

Sau khi kiểm tra, anh ta commit và push những thay đổi của anh ta tới nhánh mới.

```
[jerry@CentOS src]$ git status -s
M string_operations.c
?? string_operations

[jerry@CentOS src]$ git add string_operations.c

[jerry@CentOS src]$ git commit -m 'Added w_strlen function to return string lenght of wchar_t
string'

[wchar_support 64192f9] Added w_strlen function to return string lenght of wchar_t string
1 files changed, 10 insertions(+), 0 deletions(-)
```

Ghi chú rằng Jerry đang push những thay đổi này tới nhánh mới, đó là tại sao anh ta sử dụng nhánh tên là wchar_support thay cho nhánh master.

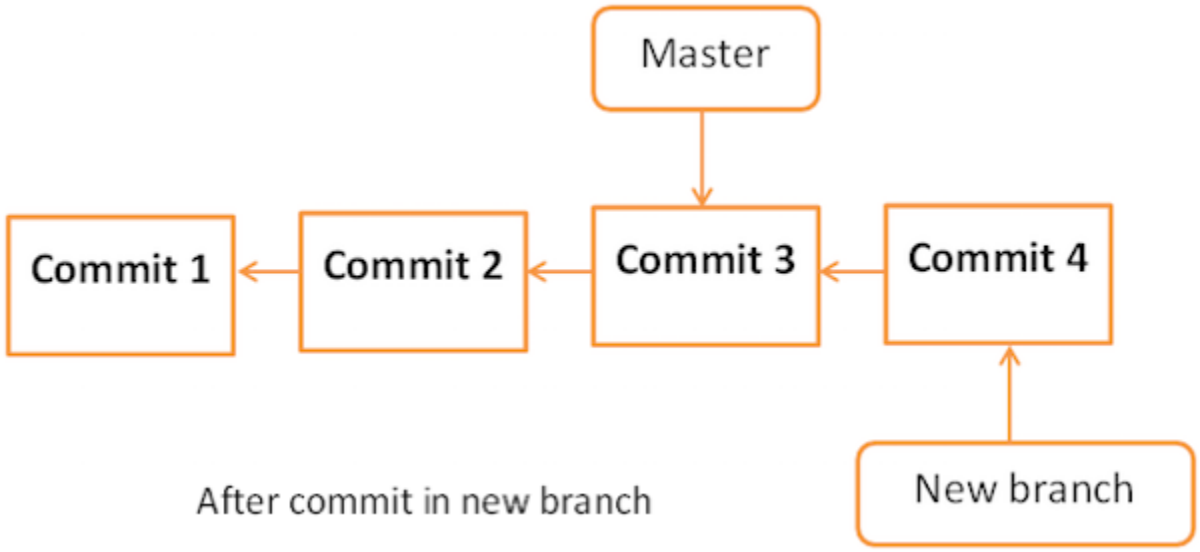
```
[jerry@CentOS src]$ git push origin wchar_support   <--- Observer branch_name
```

Lệnh trên sẽ tạo ra kết quả sau:

```
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 507 bytes, done.
Total 4 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
```

```
* [new branch]
wchar_support -> wchar_support
```

Sau khi commit những thay đổi, nhánh mới sẽ xuất hiện như sau:



Tom tò mò về những gì Jerry đang làm trong nhánh tư nhân của cậu ta và anh ta kiểm tra log từ nhánh wchar_support.

```
[tom@CentOS src]$ pwd
/home/tom/top_repo/project/src


[tom@CentOS src]$ git log origin/wchar_support -2
```

Lệnh trên sẽ tạo ra kết quả sau:

```
commit 64192f91d7cc2bcdf3bf946dd33ece63b74184a3
Author: Jerry Mouse <jerry@tutorialspoint.com>
Date: Wed Sep 11 16:10:06 2013 +0530

Added w_strlen function to return string lenght of wchar_t string


commit 577647211ed44fe2ae479427a0668a4f12ed71a1
Author: Tom Cat <tom@tutorialspoint.com>
Date: Wed Sep 11 10:21:20 2013 +0530
```

```
Removed executable binary
```

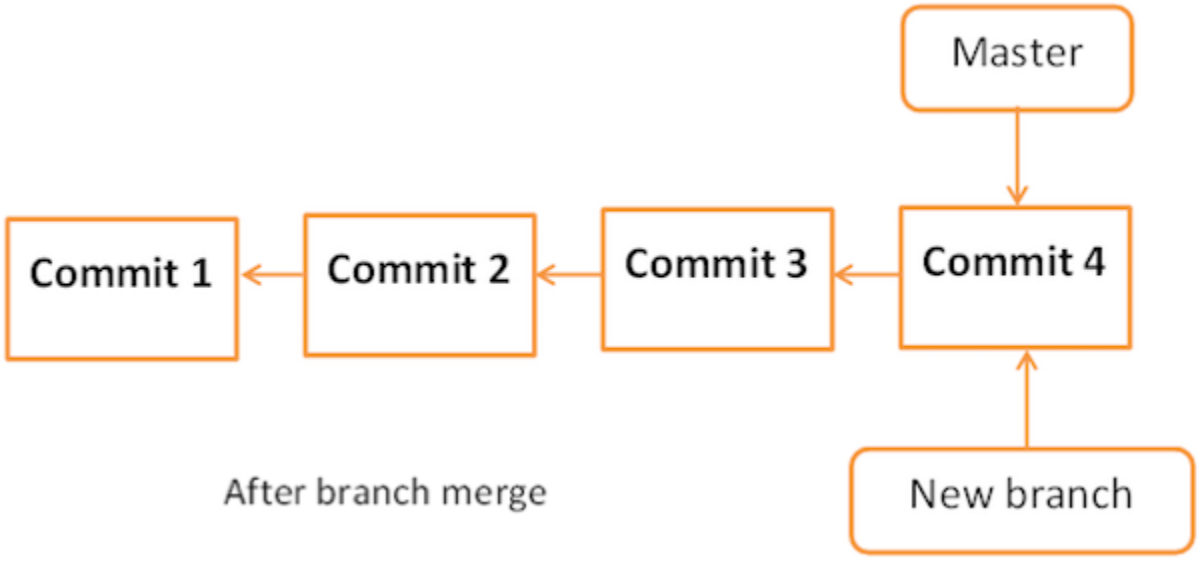
Bằng cách quan sát các thông báo commit, Tom nhận ra rằng Jerry thực hiện chức năng strlen cho ký tự mở rộng và anh ta muốn có chức năng tương tự trong nhánh master. Thay vì thực hiện lại các bước trên, anh ta quyết định lấy code của Jerry bằng cách sáp nhập nhánh của anh ta với nhánh master.

```
[tom@CentOS project]$ git branch
* master

[tom@CentOS project]$ pwd
/home/tom/top_repo/project

[tom@CentOS project]$ git merge origin/wchar_support
Updating 5776472..64192f9
Fast-forward
src/string_operations.c | 10 ++++++++++
1 files changed, 10 insertions(+), 0 deletions(-)
```

Sau hoạt động sáp nhập, nhánh master sẽ xuất hiện như sau:



Bây giờ, nhánh wchar_support đã được nhập với nhánh master. Chúng ta thẩm tra nó bằng cách quan sát thông tin commit hoặc quan sát các chỉnh sửa được thực hiện trong tệp string_operation.c.

```
[tom@CentOS project]$ cd src/
```

```
[tom@CentOS src]$ git log -1

commit 64192f91d7cc2bcdf3bf946dd33ece63b74184a3
Author: Jerry Mouse
Date: Wed Sep 11 16:10:06 2013 +0530

Added w_strlen function to return string lenght of wchar_t string

[tom@CentOS src]$ head -12 string_operations.c
```

Lệnh trên sẽ tạo ra kết quả sau:

```
#include <stdio.h>
#include <wchar.h>
size_t w_strlen(const wchar_t *s)
{
   const wchar_t *p = s;

   while (*p)
      ++p;

   return (p - s);
}
```

Sau khi kiểm tra, anh ta push những thay đổi code của anh ta tới nhánh master.

```
[tom@CentOS src]$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
5776472..64192f9 master -> master
```

Rebase các nhánh

Lệnh git rebase là một lệnh sáp nhập nhánh, nhưng điểm khác biệt ở đây là nó chỉnh sửa thứ tự của các commit.

Lệnh git merge cố gắng để đặt các commit từ nhánh khác lên trên đầu của HEAD của nhánh nội bộ hiện tại. Ví dụ, nhánh nội bộ của bạn có các commit A−>B−>C−>D và nhánh sáp nhập có các commit là A−>B−>X−>Y, thì sau đó lệnh git merge sẽ biến đổi nhánh nội bộ hiện tại thành một nhánh giống như A−>B−>C−>D−>X−>Y

Lệnh git rebase cố gắng để tìm ra gốc giữa nhánh nội bộ hiện tại và nhánh sáp nhập. Nó push các commit tới nhánh nội bộ bằng cách chỉnh sửa thứ tự của các commit trong nhánh nội bộ hiện tại. Ví dụ, hai nhánh có các commit như trên, thì lệnh git rebase sẽ chuyển đổi nhánh nội bộ hiện tại thành một nhánh giống như A>B>X>Y>C>D.

Khi có nhiều nhà lập trình cùng làm việc trên một repository từ xa, bạn không thể chỉnh sửa thứ tự của các commit trong repository này. Trong tình huống này, bạn có thể sử dụng hoạt động rebase để đặt các commit nội bộ của bạn trên phần đầu của các commit ở repository từ xa và bạn có thể push những thay đổi.