

Môn học Quản lý dự án phần mềm

Tuần 6

Thao tác cơ bản bằng dòng lệnh và hành động nâng cao trong Git:

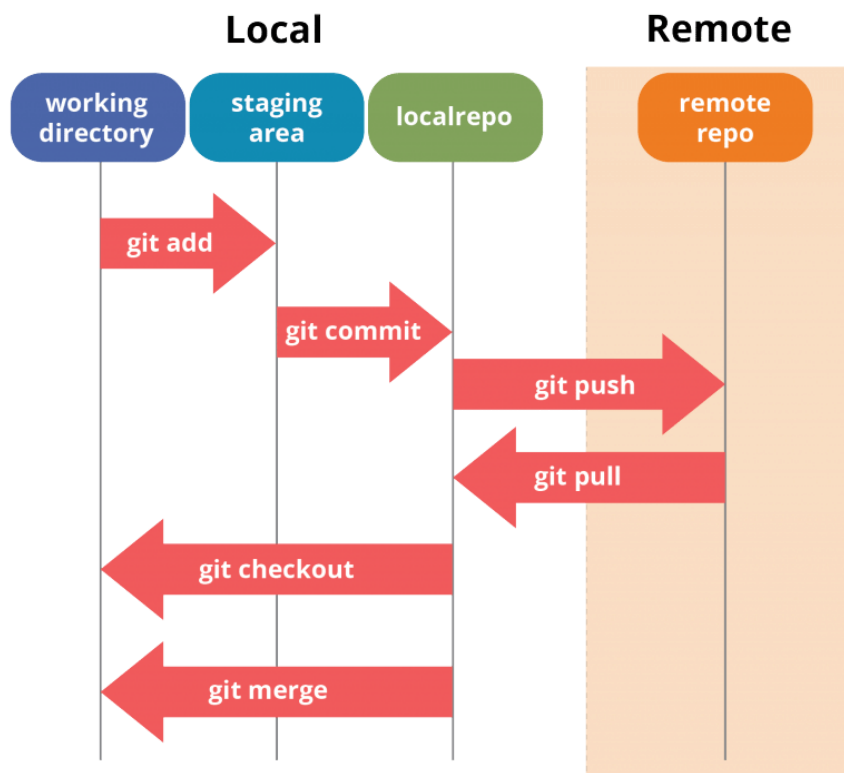
Một số thao tác cơ bản trong Git là:

1. Init
2. Add
3. Commit
4. Pull
5. Push

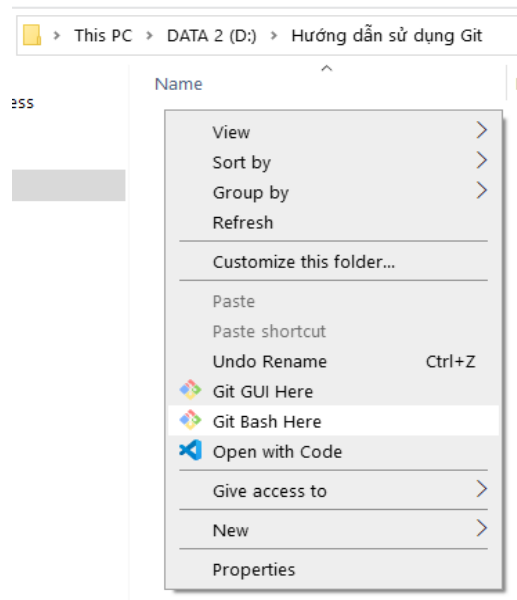
Một số hành động nâng cao trong Git là:

1. Branch
2. Merge
3. Rebase

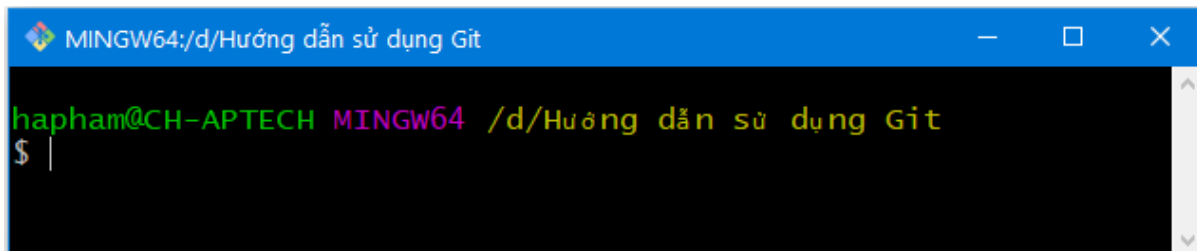
Trước tiên khi đi vào từng nội dung, chúng ta hãy tìm hiểu về kiến trúc và cách thức hoạt động của kho Git. Hãy xem sơ đồ của Git dưới đây:



Trong **Hướng dẫn sử dụng Git** này, để làm quen với các dòng lệnh tôi sẽ chỉ cho bạn làm việc với Git Bash. **Git Bash** là giao diện dòng lệnh (CLI) chỉ có văn bản để sử dụng Git trên Windows, cung cấp các tính năng để chạy các tập lệnh tự động. Sau khi cài đặt Git trong hệ thống Windows của bạn, chỉ cần mở thư mục của bạn nơi mà bạn muốn lưu trữ tất cả các tệp dự án của mình; nhấp chuột phải và chọn '**Git Bash here**'.



Điều này sẽ mở ra một cửa sổ dòng lệnh Git Bash nơi bạn có thể nhập các lệnh bắt đầu từ dấu \$ để thực hiện các hoạt động Git khác nhau.



Bây giờ, nhiệm vụ tiếp theo là khởi tạo kho lưu trữ của bạn.

Lệnh Git Init: git init tạo một kho Git trống hoặc khởi tạo lại một kho hiện có. Về cơ bản, nó tạo ra một thư mục **.git** với các thư mục con và các tệp mẫu. Chạy lệnh **git init** trong kho lưu trữ hiện tại sẽ không ghi đè lên những thứ đã có. Bây giờ để khởi tạo kho lưu trữ, bạn hãy gõ dòng lệnh tại cửa sổ Git bash vừa được mở ở trên:

git init

```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git
$ git init
Initialized empty Git repository in D:/Hướng dẫn sử dụng Git/.git/
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$
```

Khi kết thúc dòng lệnh kho lưu trữ của bạn đã được khởi tạo, việc tiếp theo chúng ta hãy tạo một số tệp trong thư mục / kho lưu trữ. Ví dụ: tôi đã tạo hai tệp văn bản là *aptechbmt-1.txt* và *aptechbmt-2.txt* bằng cách sử dụng lệnh **touch** sau đó liệt kê danh sách các tệp trong thư mục hiện hành bằng lệnh **dir**

```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ touch aptechbmt-1.txt
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ touch aptechbmt-2.txt
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ dir
aptechbmt-1.txt  aptechbmt-2.txt
```

Bây giờ hãy xem liệu các tệp này đã có trong chỉ mục của tôi hay không bằng cách sử dụng lệnh **git status**. Chỉ mục ở đây là gì? Chỉ mục giữ một ảnh chụp nhanh "snapshot" nội dung của cây / thư mục làm việc và ảnh chụp nhanh này được lấy làm nội dung cho phiên bản tiếp theo được thực hiện trong kho lưu trữ cục bộ. Để dễ hình dung, sau khi khởi tạo kho lưu trữ git, phiên bản đầu tiên của chúng ta chưa có gì đúng không? Khi chúng ta thêm 2 tệp tin mới như trên, nghĩa là kho lưu trữ của chúng ta đã có sự thay đổi. Do đó phiên bản thứ 2 sẽ chứa ảnh chụp nội dung thay đổi (thêm 2 file mới) đó. ^^

Lệnh Git Status : Lệnh **git status** liệt kê tất cả các tệp đã bị sửa đổi, sẵn sàng để thêm vào kho lưu trữ cục bộ. Bạn hãy thử gõ:

```
git status
```

```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    aptechbmt-1.txt
    aptechbmt-2.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Kết quả cho thấy kho lưu trữ của tôi hiện có hai tệp chưa được thêm vào để lập chỉ mục nội dung. Điều này có nghĩa là tôi không thể cam kết "**commit**" các thay đổi với các tệp này.

Lệnh Git Add: Lệnh này giúp chúng ta cập nhật chỉ mục nội dung bằng cách sử dụng nội dung hiện tại được tìm thấy trong thư mục làm việc hiện hành "**working directory**" để lập chỉ mục nội dung mới, và sau đó sẵn sàng chuyển chỉ mục nội dung được lập này sang một khu vực khác được tổ chức cho lần cam kết "commit" tiếp theo gọi là "**staging area**".

Quay trở lại sơ đồ mô tả quy trình làm việc của Git, rõ ràng là trước khi chúng ta thực hiện commit, thì nội dung chỉ mục của sự thay đổi trong kho được lưu trữ và luân chuyển từ khu vực *working directory* sang khu vực *staging area*.

Như vậy, sau khi bạn thực hiện thay đổi bất kỳ đối với thư mục làm việc **working directory** và trước khi chạy lệnh cam kết "**commit**", bạn phải sử dụng lệnh **add** nhằm thêm file và lập chỉ mục nội dung thay đổi cho nó để tiếp tục chuyển sang lưu trữ tại **staging area** chuẩn bị sẵn sàng cho **commit**. Để thực hiện hãy sử dụng các lệnh dưới đây:

`git add <đường dẫn thư mục>`

hoặc là

`git add <đường dẫn file>`

Hãy để tôi chứng minh lệnh **git add** cho bạn để bạn có thể hiểu nó tốt hơn. Tôi sẽ lập chỉ mục nội dung thay đổi của kho cho các tập tin bằng lệnh **git add -A**. Lệnh này sẽ thêm tất cả các tệp vào chỉ mục trong thư mục nhưng chưa được cập nhật trong chỉ mục.

```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git add -A

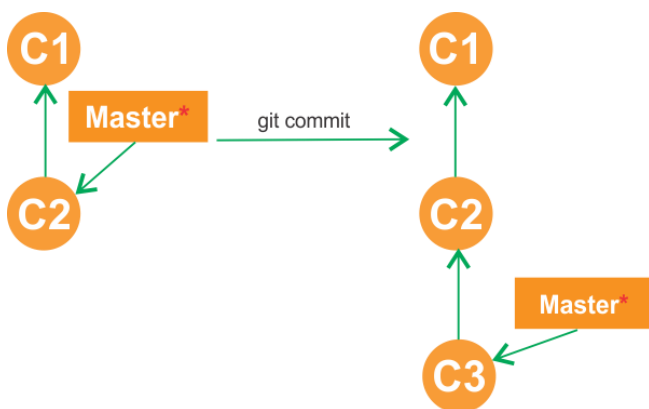
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   aptechbmt-1.txt
        new file:   aptechbmt-2.txt
```

Bây giờ các tệp mới được thêm vào và lập chỉ mục nội dung, bạn đã sẵn sàng để commit chúng.

Lệnh Git Commit: Lệnh này được hiểu như là việc ghi lại các ảnh chụp nhanh "snapshot" của kho lưu trữ tại một thời điểm nhất định. Ảnh chụp nhanh đã được commit sẽ không bao giờ thay đổi trừ khi chúng ta có sự thay đổi trong kho lưu trữ một cách rõ ràng. Hãy để tôi giải thích cách commit hoạt động với sơ đồ dưới đây:



Ở đây, C1 là commit ban đầu, tức là ảnh chụp nhanh của thay đổi đầu tiên mà từ đó một ảnh chụp nhanh khác được tạo với các thay đổi có tên là C2. Lưu ý rằng bản gốc master (Head) sẽ trở đến commit mới nhất.

Bây giờ, khi tôi cam kết một lần nữa, một ảnh chụp nhanh C3 khác được tạo và bây giờ, bản gốc trở đến C3 thay vì C2.

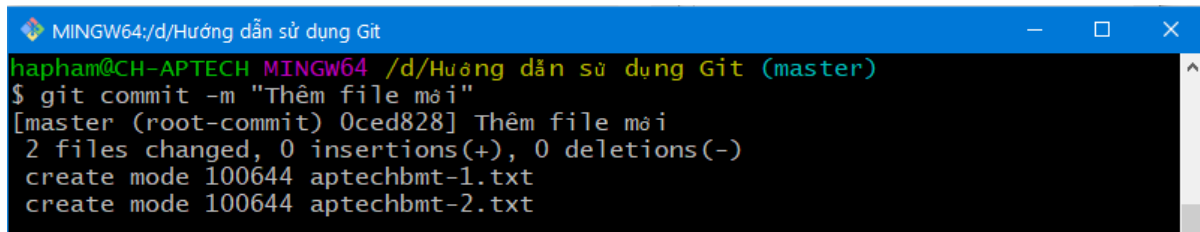
Git nhằm mục đích giữ cho cam kết càng nhẹ càng tốt. Vì vậy, nó không sao chép một cách mù quáng toàn bộ thư mục mỗi khi bạn commit; nó lưu trữ bao gồm các commit như một tập hợp các thay đổi hoặc nó chuyển phiên bản của kho lưu trữ từ phiên bản này sang phiên bản khác mới hơn. Nói một cách dễ hiểu, nó chỉ sao chép những thay đổi được thực hiện trong kho lưu trữ. Bạn có thể commit bằng cách sử dụng lệnh dưới đây:

git commit

Hoặc bạn có thể sử dụng:

git commit -m <nội dung thông điệp cần lưu trữ>

Thử xem sao nhé

A screenshot of a terminal window with a blue title bar that reads "MINGW64:/d/Hướng dẫn sử dụng Git". The terminal shows a user prompt "hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)" followed by the command "\$ git commit -m 'Thêm file mới'". The output shows the commit hash "[master (root-commit) 0ced828] Thêm file mới", followed by "2 files changed, 0 insertions(+), 0 deletions(-)", and then "create mode 100644 aptechbmt-1.txt" and "create mode 100644 aptechbmt-2.txt".

```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git commit -m "Thêm file mới"
[master (root-commit) 0ced828] Thêm file mới
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 aptechbmt-1.txt
create mode 100644 aptechbmt-2.txt
```

Như bạn có thể thấy ở trên, lệnh **git commit** đã cam kết các thay đổi trong hai tệp trong kho lưu trữ cục bộ.

Bây giờ, nếu bạn muốn thực hiện một commit ảnh chụp nhanh tất cả các thay đổi trong thư mục làm việc cùng một lúc, bạn có thể sử dụng lệnh dưới đây:

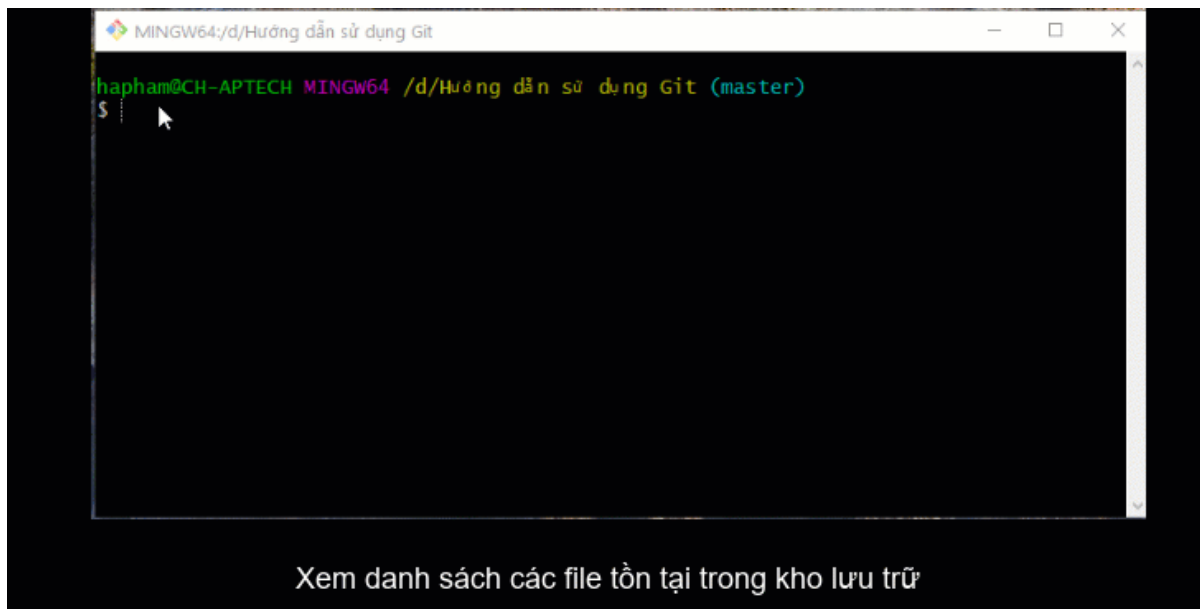
git commit -a

Bây giờ tôi sẽ tạo thêm hai tệp văn bản trong thư mục làm việc của tôi là *aptechbmt-3.txt* và *aptechbmt-4.txt* và chỉnh sửa một chút đối với file *aptechbmt-1.txt* bằng cách thêm nội dung cho nó. Lưu ý: tất cả các file này đều chưa được thêm vào để lập chỉ mục nội dung.

Tôi sẽ thêm file *aptechbmt-3.txt* bằng lệnh:

git add aptechbmt-3.txt

Lúc này tôi chỉ thêm *aptechbmt-3.txt* vào chỉ mục còn *aptechbmt-4.txt* thì không. Bây giờ, tôi muốn commit tất cả các thay đổi trong thư mục cùng một lúc. Tham khảo cách làm bên dưới.



Lệnh này sẽ commit một ảnh chụp nhanh về tất cả các thay đổi trong thư mục làm việc nhưng chỉ bao gồm các sửa đổi đối với các tệp được theo dõi, tức là các tệp đã được thêm bằng git add tại một số điểm trong lịch sử của chúng. Do đó, *aptechbmt-4.txt* không được commit vì nó chưa được thêm vào chỉ mục. Nhưng những thay đổi trong tất cả các tệp trước đó trong kho lưu trữ đã được cam kết. Ví dụ: *aptechbmt-1.txt*, *aptechbmt-2.txt*, *aptechbmt-3.txt*. Bây giờ tôi đã thực hiện các commit mong muốn của mình trong kho lưu trữ cục bộ của tôi.

Lưu ý rằng trước khi bạn tác động đối với kho lưu trữ trung tâm bằng cách thực hiện các thay đổi trong kho lưu trữ, bạn phải luôn luôn nhớ rằng nên thay đổi từ kho lưu trữ trung tâm sang kho lưu trữ cục bộ của mình trước, để kho lưu trữ cục bộ của mình được cập nhật mới nhất với kho lưu trữ cộng tác viên khác đã đóng góp trong kho lưu trữ trung tâm. Để luôn cập nhật kho lưu trữ cục bộ của mình từ kho lưu trữ trung tâm (ở đây là kho lưu trữ Git Hub) tôi sẽ sử dụng lệnh pull .

Lệnh Pull: Lệnh **git pull** tìm nạp các thay đổi từ kho lưu trữ từ xa sang kho lưu trữ cục bộ. Nó hợp nhất các thay đổi ở nguồn trong kho lưu trữ cục bộ của bạn, đây là một nhiệm vụ phổ biến trong việc cộng tác dựa trên Git.

Để làm được điều này, trước tiên bạn cần đặt kho lưu trữ trung tâm của mình làm gốc "origin" hay có thể hiểu là thiết lập kết nối giữa kho lưu trữ từ xa và kho lưu trữ cục bộ bằng lệnh:

git remote add origin <đường dẫn liên kết đến kho lưu trữ trung tâm của bạn>

Trong đó để có đường dẫn liên kết đến kho lưu trữ trung tâm thì trên **GitHub** bạn phải tạo cho mình một kho lưu trữ (Cách tạo xem lại [bài này](#) nhé). Ở đây: tôi sử dụng kho lưu trữ từ xa tại địa chỉ này: <https://github.com/aptechbuonmathuot/tu-hoc-git>

```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git remote -v

hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git remote add origin "https://github.com/aptechbuonmathuot/tu-hoc-git"

hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git remote -v
origin https://github.com/aptechbuonmathuot/tu-hoc-git (fetch)
origin https://github.com/aptechbuonmathuot/tu-hoc-git (push)
```

Đầu tiên tôi sử dụng lệnh **git remote -v** để kiểm tra xem mình đã cấu hình tới các máy chủ từ xa nào, nó sẽ liệt kê tên của mỗi máy chủ từ xa kèm theo đường dẫn. Nếu bạn đã thực hiện sao chép từ một kho chứa có sẵn về kho lưu trữ cục bộ của mình, thì ít nhất bạn sẽ thấy bản gốc (origin) - đây là tên gọi mặc định mà Git đặt cho phiên bản trên máy chủ mà bạn đã sao chép từ đó bạn có thể đặt tên khác cho nó để dễ nhớ hơn. Ở ví dụ này, trước đây tôi chưa từng thực hiện việc kết nối tới một kho lưu trữ từ xa nào, nên để thực hiện thì việc đầu tiên tôi phải thiết lập kết nối với kho lưu trữ từ xa bằng lệnh **git remote add**

Sau khi đã làm xong các bước chuẩn bị, bây giờ chỉ việc sử dụng dòng lệnh sau:

```
git pull origin master
```

Lệnh này sẽ sao chép tất cả các tệp từ nhánh chính master của kho lưu trữ từ xa vào kho lưu trữ cục bộ của bạn.

```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git pull origin master
From https://github.com/aptechbuonmathuot/tu-hoc-git
* branch      master      -> FETCH_HEAD
* [new branch] master      -> origin/master
fatal: refusing to merge unrelated histories
```

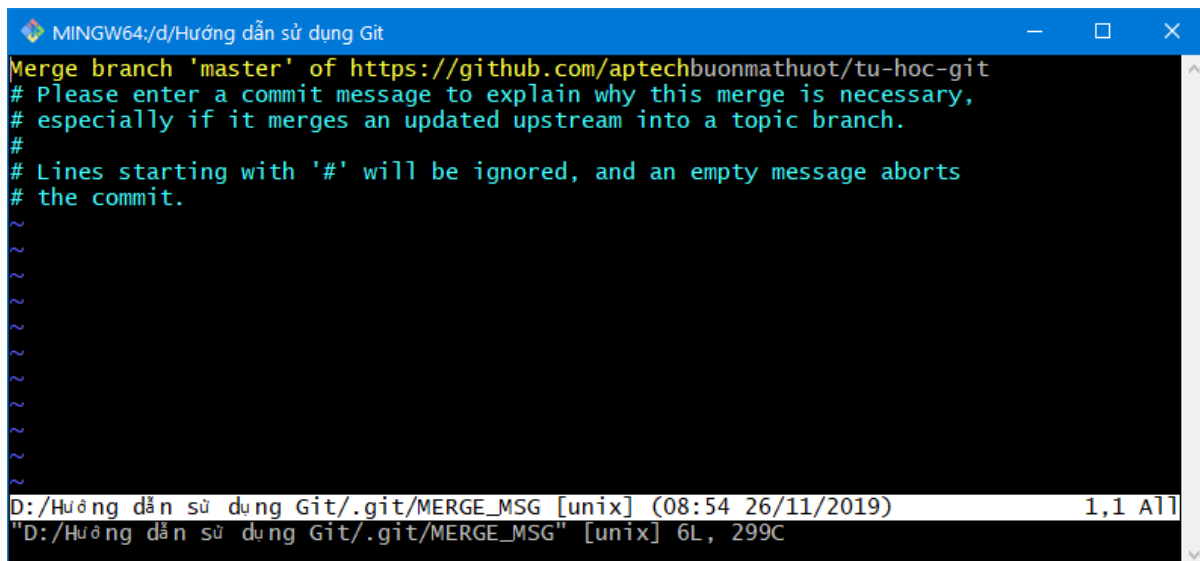
Việc này sẽ hoàn toàn diễn ra suôn sẻ nếu kho lưu trữ cục bộ của bạn và kho lưu trữ từ xa có lịch sử cam kết khớp nhau. Còn trong trường hợp này Git báo lỗi "**fatal: refusing to merge unrelated histories**". Để tôi giải thích lỗi này cho bạn hiểu: Lỗi này thường xảy ra do 2 nguyên nhân:

- Một là, bạn đã sao chép một dự án và bằng cách nào đó, thư mục .git đã bị xóa hoặc bị hỏng. Điều này dẫn đến việc Git không biết về lịch sử commit tại kho cục bộ của bạn và do đó sẽ khiến gây ra lỗi này khi bạn cố gắng đẩy hoặc kéo từ kho lưu trữ từ xa.

- Hai là, bạn đã tạo một kho lưu trữ mới và đã thêm một vài commit (từ đầu bài đến giờ ^^ mình thêm khá khá rồi nhĩ) và bây giờ bạn đang cố gắng lấy từ một kho lưu trữ từ xa đã có một số commit của riêng nó. Git cũng sẽ đưa ra lỗi trong trường hợp này, vì nó không biết hai dự án có liên quan như thế nào. Đây là trường hợp trong ví dụ này. Để khắc phục, rất đơn giản, bạn chỉ cần gõ lại lệnh sau:

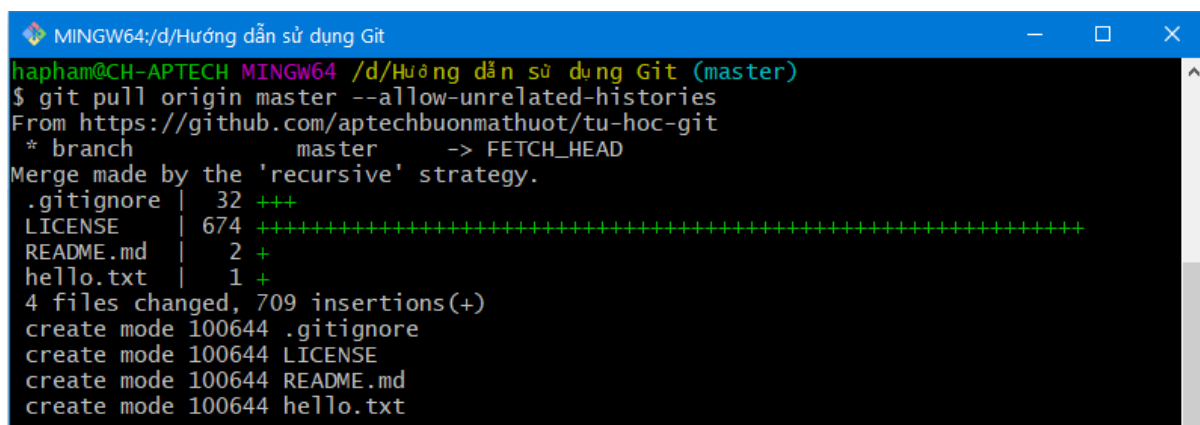
```
git pull origin master --allow-unrelated-histories
```

Lệnh này được hiểu như chúng ta sẽ hợp nhất tất cả các commit trên kho lưu trữ từ xa về kho lưu trữ cục bộ một cách bắt buộc (Nhiều khi ép mới chịu làm ^^) và mặc định Git sẽ tạo ra một commit thông báo về việc hợp nhất này, nên sau khi bạn kết thúc dòng lệnh trên Git sẽ hiển thị cửa sổ cho phép chúng ta chỉnh sửa nội dung commit. Bạn có thể chỉnh sửa chỗ dòng chữ màu vàng nhé.



```
MINGW64; d:\Hướng dẫn sử dụng Git
Merge branch 'master' of https://github.com/aptechbunmathuot/tu-hoc-git
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
~
~
~
~
~
~
~
~
~
D:\Hướng dẫn sử dụng Git/.git/MERGE_MSG [unix] (08:54 26/11/2019) 1,1 All
"D:\Hướng dẫn sử dụng Git/.git/MERGE_MSG" [unix] 6L, 299C
```

Kết quả:

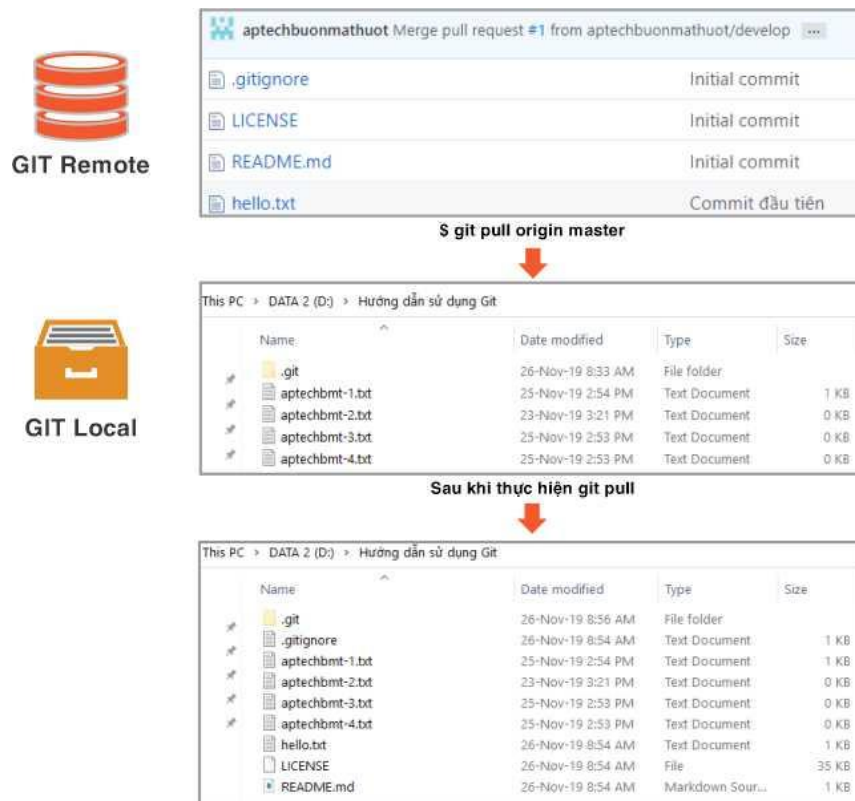


```
MINGW64; d:\Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git pull origin master --allow-unrelated-histories
From https://github.com/aptechbunmathuot/tu-hoc-git
* branch      master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 .gitignore | 32 +++
 LICENSE    | 674 +++++
 README.md  | 2 +
 hello.txt  | 1 +
4 files changed, 709 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
create mode 100644 README.md
create mode 100644 hello.txt
```

Sử dụng lệnh **git log --oneline** để xem nhanh lịch sử commit trên máy tính cục bộ đã thay đổi thế nào

```
MINGW64; d:\Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git log --oneline
e684e80 (HEAD -> master) Merge branch 'master' of https://github.com/aptechbunmathuot/tu-hoc-git
96a4dfb Thêm nhiều file mới
0ced828 Thêm file mới
fc38f37 (origin/master) Merge pull request #1 from aptechbunmathuot/develop
795a628 Commit đầu tiên
2334e00 Initial commit
```

Như bạn thấy, các commit của kho lưu trữ từ xa đều được ghép vào kho lưu trữ cục bộ của mình. Trong đó, có 1 commit trên cùng "Merge branch 'master' ..." được tạo thêm. Ngoài ra, bạn có thể kiểm tra lại ngoài Windows Explorer để thấy các file trên kho lưu trữ từ xa cũng sẽ được tải về lưu trữ tại máy tính nhé.



Lưu ý: Bạn chỉ thực hiện điều này, khi thực sự hiểu rõ về lịch sử commit của 2 kho lưu trữ nhé. Ngoài ra, chúng ta cũng có thể thử kéo các tệp từ một nhánh khác bằng lệnh sau:

git pull origin <tên nhánh cần pull>

Như vậy, chúng ta vừa lấy các nội dung từ kho lưu trữ từ xa về thành công. Vậy làm cách nào để đẩy ngược lại dự án của mình ở cục bộ lên kho lưu trữ từ xa? Hãy tiếp tục bằng cách sử dụng lệnh **Push** dưới đây.

Lệnh push: Lệnh này chuyển commit từ kho lưu trữ cục bộ của bạn sang kho lưu trữ từ xa. Nó là đối nghịch của lệnh **pull**.

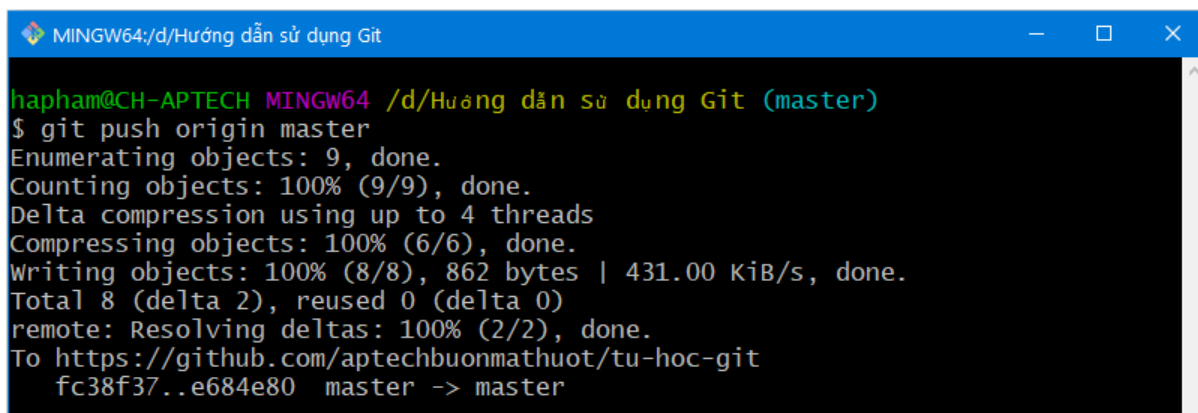
Pull là để kéo các cam kết vào kho lưu trữ cục bộ trong khi **Push** là đẩy các cam kết vào kho lưu trữ từ xa.

Việc sử dụng **git push** là để xuất bản các thay đổi cục bộ của bạn lên một kho lưu trữ từ xa. Sau khi bạn tích lũy được một số cam kết cục bộ và sẵn sàng chia sẻ chúng với các thành viên còn lại trong nhóm, bạn có thể đẩy chúng vào kho lưu trữ từ xa bằng cách sử dụng lệnh sau:

```
git push <tên remote>
```

Lưu ý : Điều khiển từ xa này đề cập đến kho lưu trữ từ xa đã được cấu hình ở trên trước khi sử dụng lệnh **pull**.

Tôi sẽ sử dụng lệnh **git push origin master** để đẩy kho lưu trữ cục bộ này lên nhánh **master** của kho lưu trữ từ xa của tôi. Khi kết thúc dòng lệnh trên, Git sẽ xuất hiện cửa sổ thông báo bạn phải đăng nhập vào tài khoản Git Hub nếu như bạn chưa từng đăng nhập trước đây. Tài khoản này phải là chủ sở hữu kho lưu trữ từ xa hoặc phải được cấp quyền truy cập kho lưu trữ. Sau khi đăng nhập, kết quả sẽ như bên dưới:

A screenshot of a Windows terminal window with a blue title bar that reads "MINGW64:/d/Hướng dẫn sử dụng Git". The terminal shows the command "hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)\$ git push origin master" and its output: "Enumerating objects: 9, done.", "Counting objects: 100% (9/9), done.", "Delta compression using up to 4 threads", "Compressing objects: 100% (6/6), done.", "Writing objects: 100% (8/8), 862 bytes | 431.00 KiB/s, done.", "Total 8 (delta 2), reused 0 (delta 0)", "remote: Resolving deltas: 100% (2/2), done.", and "To https://github.com/aptechbuonmathuot/tu-hoc-git fc38f37..e684e80 master -> master".

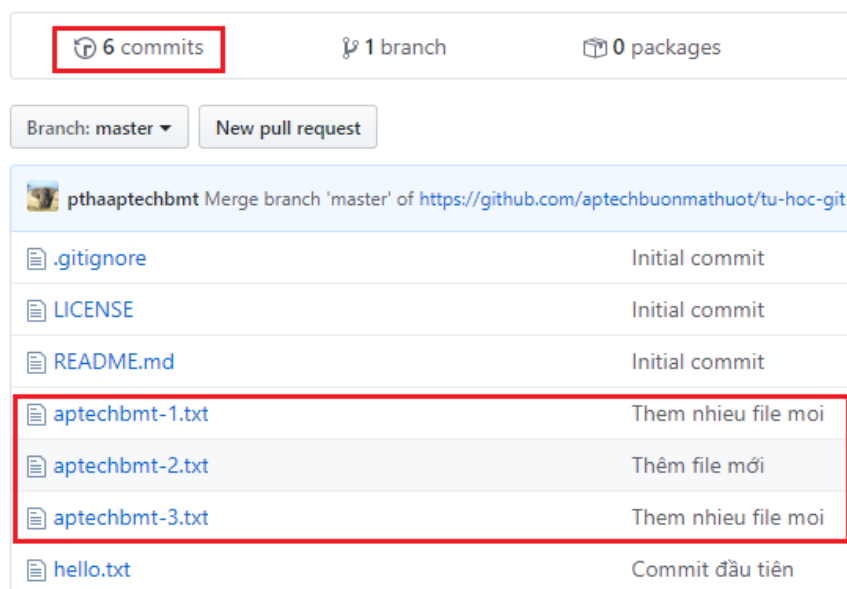
```
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 862 bytes | 431.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/aptechbuonmathuot/tu-hoc-git
fc38f37..e684e80 master -> master
```

Lưu ý: Nếu khi bạn thực hiện mà mắc lỗi như bên dưới.

```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git push origin master
remote: Permission to aptechbunmathuot/tu-hoc-git.git denied to pthaaptechbmt.
fatal: unable to access 'https://github.com/aptechbunmathuot/tu-hoc-git/': The requested URL
returned error: 403
```

Lỗi này xảy ra khi quyền truy cập kho lưu trữ từ xa của bạn chưa được cấp do tài khoản đang sử dụng không được truy cập vào kho lưu trữ này. Để khắc phục bạn nên đăng nhập lại bằng cách, truy cập vào **Control Panel** **All Control Panel Items** **Credential Manager** tại tab **Windows Credential** tìm tài khoản GitHub đã được lưu trước đây và xóa nó đi. Sau đó chạy lại lệnh `git push`, tiến hành làm theo hướng dẫn bên trên là được.

Sau khi thực hiện lệnh **git push** thì các thay đổi từ kho lưu trữ cục bộ sang kho lưu trữ từ xa cùng với tất cả các commit cần thiết và các đối tượng bên trong. Điều này tạo ra một nhánh cục bộ trong kho đích. Bây giờ hãy kiểm tra trên kho lưu trữ từ xa có gì nhé!



Như vậy, file `aptechbmt-4.txt` ở dưới kho lưu trữ cục bộ sẽ không được đẩy lên kho lưu trữ từ xa vì nó chưa được thêm vào để lập chỉ mục nội dung bằng `git add`. Các file và lịch sử commit đều đã được đẩy lên thành công.

Để ngăn việc ghi đè, Git không cho phép push khi kết quả là hợp nhất chuyển tiếp không nhanh trong kho đích.

Lưu ý : Hợp nhất chuyển tiếp không nhanh có nghĩa là hợp nhất ngược dòng tức là hợp nhất với các nhánh tổ tiên hoặc các nhánh cha từ một nhánh con.

Để kích hoạt hợp nhất như vậy, sử dụng lệnh dưới đây:

`git push <tên remote> -force`

Lệnh trên buộc phải push ngay cả khi nó dẫn đến kết hợp chuyển tiếp không nhanh.

Đến đây, với các lệnh cơ bản của Git chắc các bạn cũng đã hình dung được quy trình làm việc của git và github theo sơ đồ ở đầu bài này. Bây giờ, chúng ta hãy tiếp tục thực hiện các hành động nâng cao hơn.

Hướng dẫn các hoạt động nâng cao với Git

Git branch: Các nhánh trong Git không có gì ngoài con trỏ đến một cam kết cụ thể. Git thường thích giữ các nhánh của nó càng nhẹ càng tốt. Về cơ bản có hai loại là **local branches** và **remote tracking branches**.

Local branch (nhánh cục bộ) chỉ là một nhánh rẽ khác trên cây làm việc của bạn. Mặt khác, các **remote tracking branches** (nhánh theo dõi từ xa) có các mục đích đặc biệt. Một số trong số chúng là:

- Chúng liên kết công việc của bạn từ kho lưu trữ cục bộ với công việc trên kho lưu trữ từ xa.
- Chúng tự động phát hiện các nhánh từ xa để nhận thay đổi, khi bạn sử dụng git pull .

Bạn có thể kiểm tra nhánh hiện tại của bạn bằng cách sử dụng lệnh:

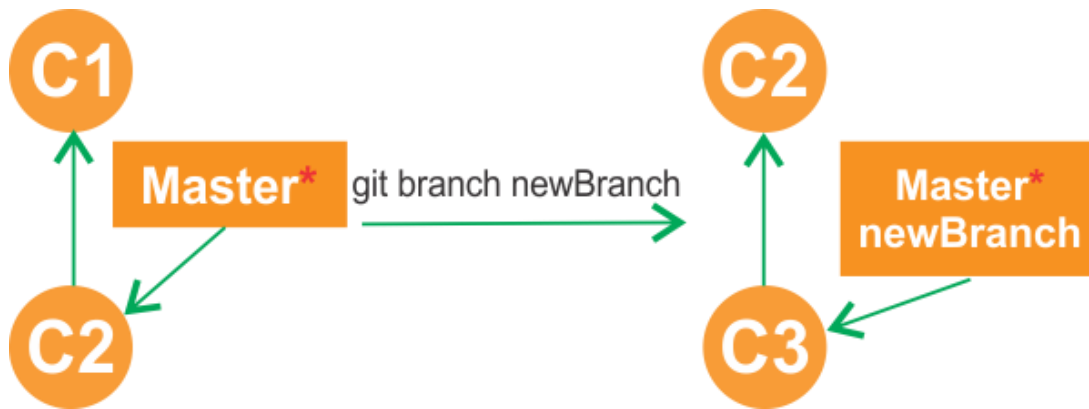
`git branch`

Một câu thần chú mà bạn nên luôn luôn tụng trong khi phân nhánh ^^

Để tạo một nhánh mới, tôi sử dụng lệnh sau:

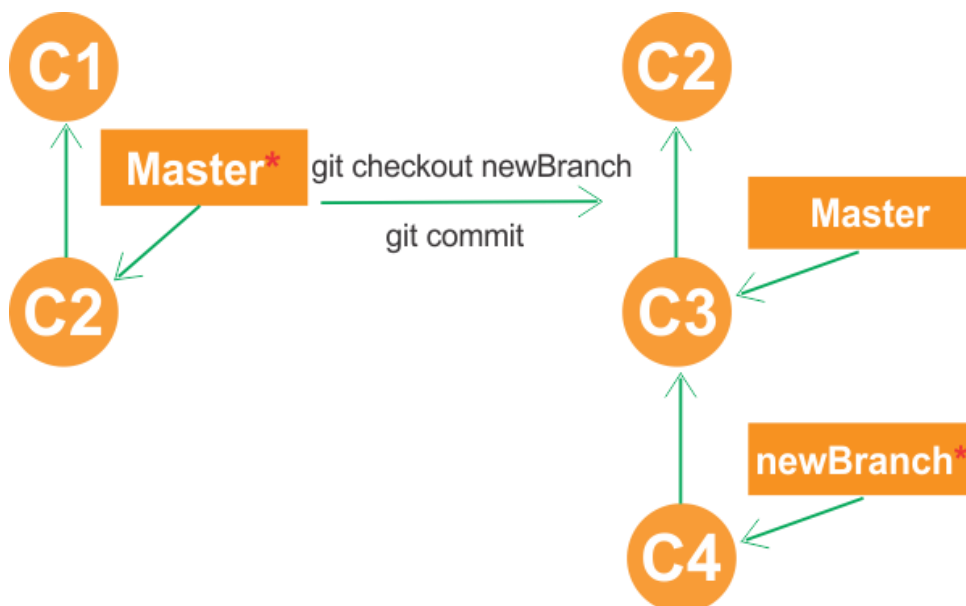
`git branch <tên chi nhánh>`

Sơ đồ dưới đây cho thấy quy trình làm việc khi một nhánh mới được tạo. Khi chúng ta tạo một nhánh mới, nó sẽ bắt nguồn từ nhánh chính. Ghi chú: dấu * để chỉ Head đang trở tới commit tương ứng của nhánh đó, trên sơ đồ Head của nhánh master đang trở tới commit C2, khi tạo nhánh newBranch thì Head của 2 nhánh master và newBranch đều trở đến commit C3



Việc tạo nhánh sẽ giúp bạn phân chia và quản lý công việc được dễ dàng hơn. Thay vì, phải tạo một kho lưu trữ mới để phát triển một tính năng mới cho dự án đã có thì bạn có thể phân nhánh ngay trong kho lưu trữ dự án để thực hiện công việc này mà không ảnh hưởng gì tới nhánh chính master.

Bây giờ, chúng ta hãy xem làm thế nào để commit khi sử dụng các nhánh.



Sự phân nhánh bao gồm một cam kết cụ thể cùng với tất cả các cam kết cha. Như bạn có thể thấy trong sơ đồ trên, newBranch đã tách ra khỏi bản gốc và do đó sẽ tạo ra một đường dẫn khác.

`git checkout <tên nhánh>`

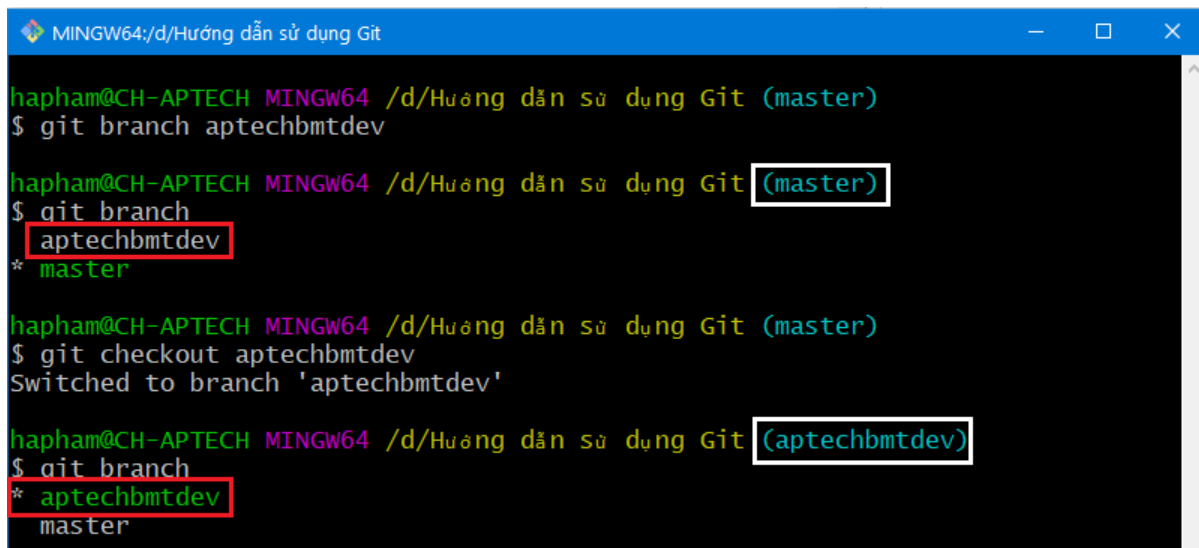
Lệnh này giúp chuyển đổi sang một nhánh khác, mặc định chúng ta đang ở master. Ở đây, tôi đã tạo ra một chi nhánh mới có tên là *aptechbmtdev*, và chuyển sang nhánh mới

bằng cách sử dụng lệnh git checkout.

Một phím tắt cho các lệnh trên là:

git checkout -b <tên nhánh>

Lệnh này sẽ tạo ra một nhánh mới và chuyển sang nhánh đó cùng một lúc. Tuy nhiên tại phiên bản tôi đang sử dụng là git version 2.24.0.windows.2 thì lệnh này không hoạt động nhé T.T. Hình ảnh dưới đây mô tả quá trình tạo branch và checkout branch, hãy để ý các ô màu đỏ và trắng để nhận ra sự khác biệt nhé.



```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git branch aptechbmtdev

hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git branch
  aptechbmtdev
* master

hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git checkout aptechbmtdev
Switched to branch 'aptechbmtdev'

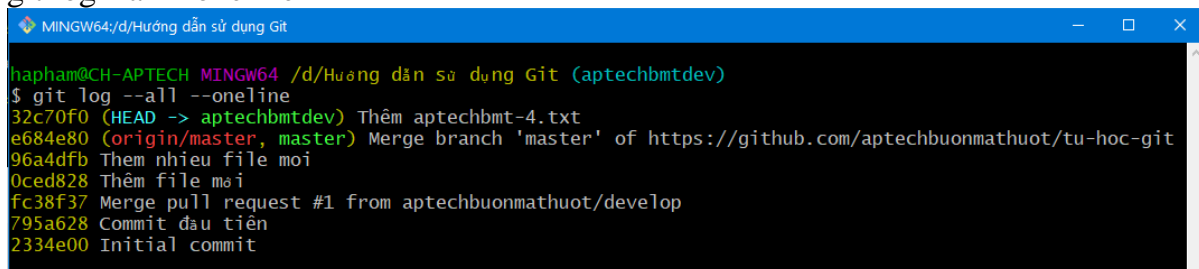
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (aptechbmtdev)
$ git branch
  aptechbmtdev
* master
```

Bây giờ trong khi chúng ta đang ở trong nhánh *aptechbmtdev*, hãy commit tệp văn bản *aptechbmt-4.txt* bằng các lệnh sau:

git add aptechbmt-4.txt
git commit -m "Thêm aptechbmt-4.txt"

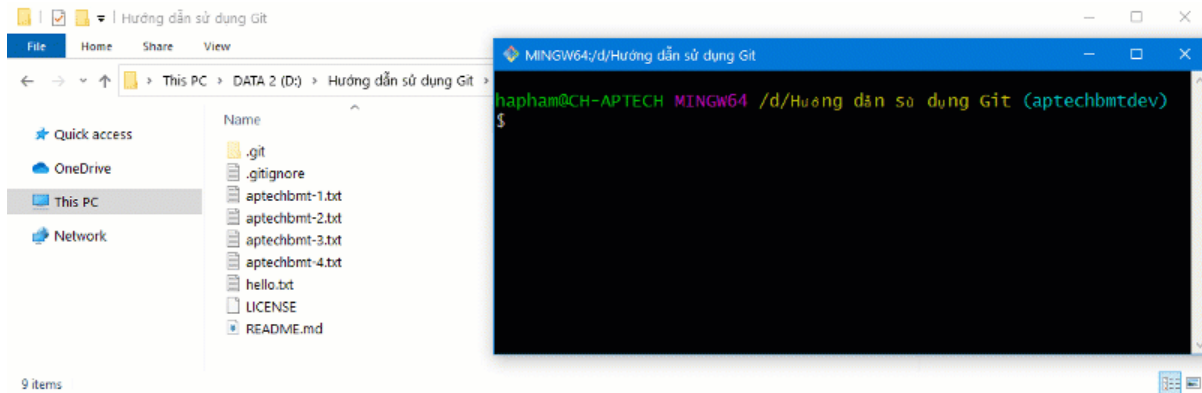
Kiểm tra lịch sử commit bằng lệnh dưới để xem kết quả:

git log --all --oneline



```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (aptechbmtdev)
$ git log --all --oneline
32c70f0 (HEAD -> aptechbmtdev) Thêm aptechbmt-4.txt
e684e80 (origin/master, master) Merge branch 'master' of https://github.com/aptechbunmathuot/tu-hoc-git
96a4dfb Thêm nhiều file mới
0ced828 Thêm file mới
fc38f37 Merge pull request #1 from aptechbunmathuot/develop
795a628 Commit đầu tiên
2334e00 Initial commit
```

Nhìn ảnh trên, bạn sẽ thấy mình đang ở branch *aptechbmtdev* để kiểm tra lịch sử commit, con trỏ Head của nhánh này đang trỏ tới commit cuối cùng của nhánh.



Hình ảnh trên mô tả quá trình **checkout** qua lại giữa các **branch**, tương ứng phần hiển thị bên tay trái là thư mục làm việc cũng có sự thay đổi về số lượng file.

Công việc tiếp theo của chúng ta là Push thay đổi vừa rồi lên kho lưu trữ từ xa. Đơn giản bằng cách thực hiện lệnh

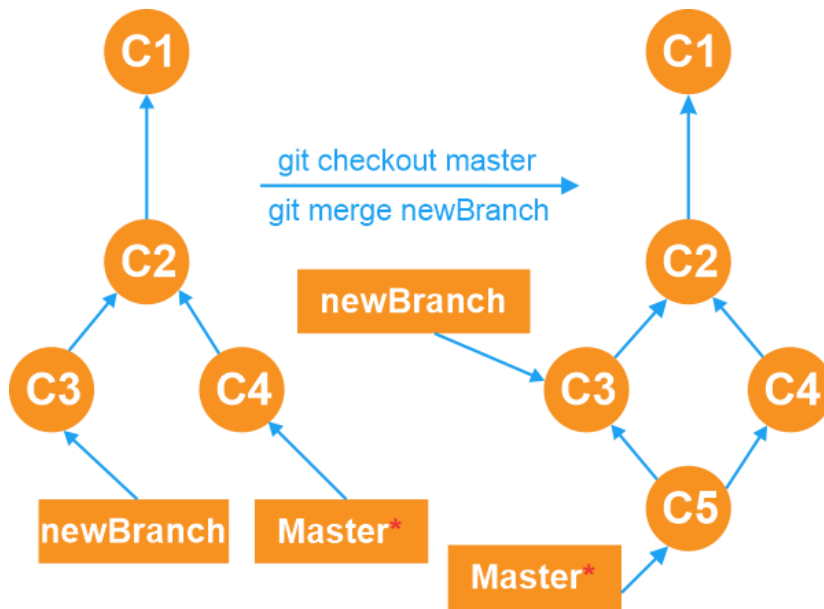
`git push origin aptechbmtdev`

Lệnh này sẽ đẩy toàn bộ nhánh *aptechbmtdev* (bao gồm các file và commit của nhánh) lên kho lưu trữ từ xa mà chúng ta đã thực hiện bên trên.

```
MINGW64/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (aptechbmtdev)
$ git push origin aptechbmtdev
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 256 bytes | 256.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'aptechbmtdev' on GitHub by visiting:
remote:   https://github.com/aptechbunmathuot/tu-hoc-git/pull/new/aptechbmtdev
remote:
To https://github.com/aptechbunmathuot/tu-hoc-git
 * [new branch]      aptechbmtdev -> aptechbmtdev
```

Bạn hãy kiểm tra lại trên kho lưu trữ của mình đã có thêm file *aptechbmt-4.txt* và commit của nó không nhé!

Git merge: Hợp nhất "Merge" là cách kết hợp công việc của các nhánh khác nhau lại với nhau. Điều này sẽ cho phép chúng ta phân nhánh, phát triển một tính năng mới và sau đó kết hợp lại với nhánh gốc.



Biểu đồ trên cho chúng ta thấy hai nhánh khác nhau giữa newBranch và master. Bây giờ, khi chúng ta hợp nhất công việc của nhánh newBranch (C3) sang nhánh master (C1,C2,C4) nó sao chép tất cả công việc (commit) của newBranch vào master (C1,C2,C3,C4) và tạo ra một commit mới (C5) trên cùng cây. Lúc này, con trỏ Head Master sẽ trở đến C5. Câu lệnh hợp nhất 2 nhánh như sau:

`git merge <tên nhánh cần hợp nhất vào>`

Lưu ý: Bạn phải checkout nhánh đích (là nhánh mà bạn muốn hợp nhất vào) như sơ đồ trên khi hợp nhất newBranch vào master bạn phải checkout master.

Bây giờ, chúng ta hãy hợp nhất tất cả các công việc của nhánh *aptechbmtdev* vào nhánh *master*. Vì vậy, trước tiên tôi sẽ kiểm tra nhánh master bằng lệnh **git checkout master** và hợp nhất với *aptechbmtdev* với lệnh **git merge aptechbmtdev**

`git checkout master`
`git merge aptechbmtdev`

```

MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (aptechbmtdev)
$ git checkout master
Switched to branch 'master'

hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git merge aptechbmtdev
Updating e684e80..32c70f0
Fast-forward
 aptechbmt-4.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 aptechbmt-4.txt

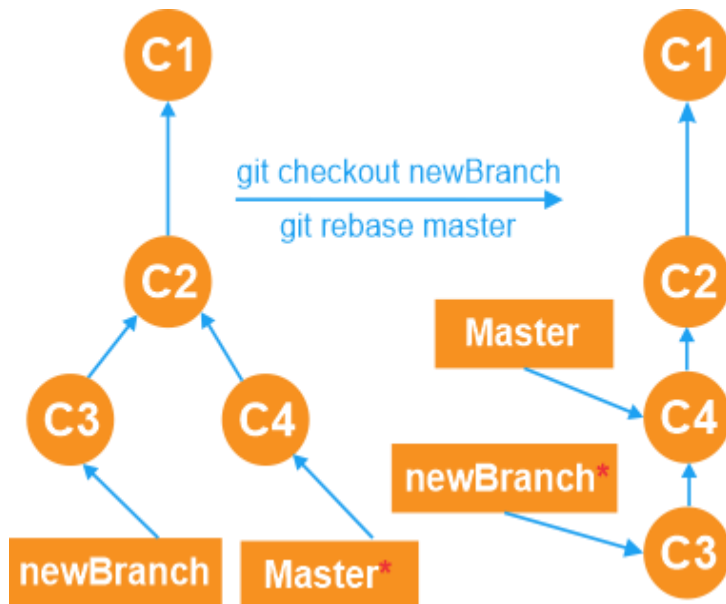
```

Sử dụng lệnh **git log --oneline --all** để kiểm tra lịch sử commit

```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git log --oneline --all
32c70f0 (HEAD -> master, origin/aptechbmtdev, aptechbmtdev) Thêm aptechbmt-4.txt
e684e80 (origin/master) Merge branch 'master' of https://github.com/aptechbunmathuot/tu-hoc-git
96a4dfb Thêm nhiều file mới
0ced828 Thêm file mới
fc38f37 Merge pull request #1 from aptechbunmathuot/develop
795a628 Commit đầu tiên
2334e00 Initial commit
```

Như bạn có thể thấy ở trên, tất cả dữ liệu từ nhánh aptechbmtdev được hợp nhất với nhánh master. Bây giờ, tệp văn bản *aptechbmt-4.txt* đã được thêm vào nhánh master.

Lệnh Rebase: Đây cũng là một cách kết hợp công việc giữa các nhánh khác nhau. Rebasing tạo một tập hợp các cam kết, sao chép chúng và lưu trữ chúng vào phía sau commit mới nhất trên nhánh đích như biểu đồ bên dưới. Ưu điểm của rebasing là nó có thể được sử dụng để tạo các chuỗi commit có trình tự.



Bây giờ, các commit từ newBranch được đặt ngay sau nhánh chính và chúng ta có một chuỗi các cam kết tuyến tính nhìn đẹp hơn ^^ là merge.

Bây giờ, để **rebase master**, hãy gõ lệnh bên dưới trong Git Bash của bạn:

```
git rebase master
```

Lệnh này sẽ chuyển tất cả commit từ nhánh hiện tại sang nhánh master. Chúng trông như thể được phát triển một cách tuần tự, nhưng thực tế chúng lại được phát triển song song.

Ví dụ: Trên dự án sẵn có của chúng ta, tôi tạo thêm một nhánh *aptechbmthotfix* và thực hiện một vài **commit** cho nó.

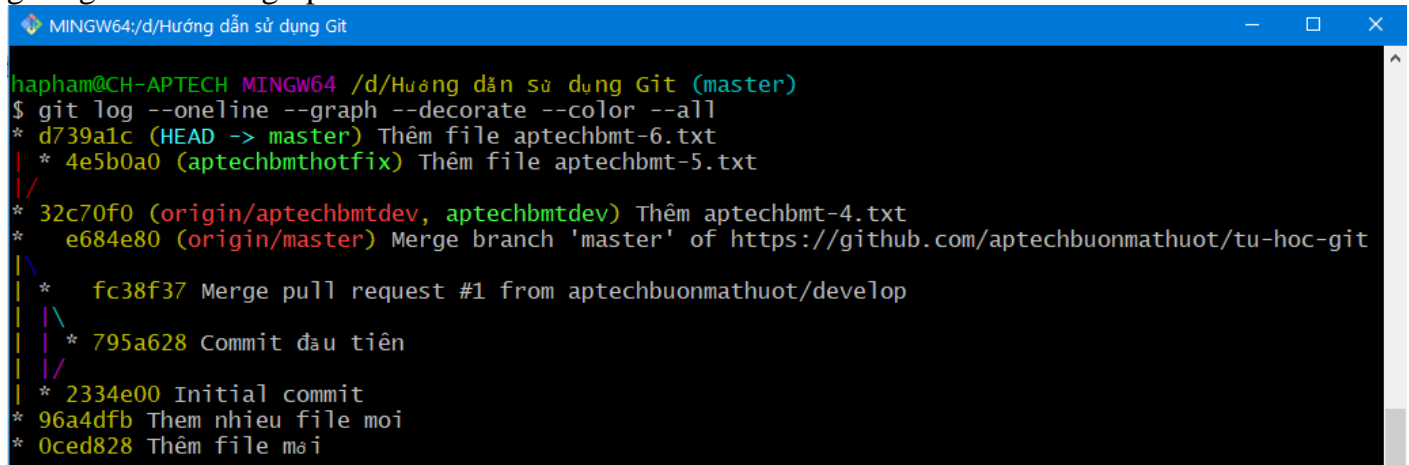
```
git branch aptechbmthotfix
git checkout aptechbmthotfix
touch aptechbmt-5.txt
git add aptechbmt-5.txt
git commit -m "Thêm file aptechbmt-5.txt"
```

Tiếp tục tôi lại **checkout master** và tạo mới 1 commit cho master. Mục đích là để trên nhánh *master* này có một **commit** mới hơn trên nhánh *aptechbmthotfix*

```
git checkout master
touch aptechbmt-6.txt
git add aptechbmt-6.txt
git commit -m "Thêm file aptechbmt-6.txt"
```

Sau khi thực hiện các lệnh trên tôi sử dụng lệnh **git log** để kiểm tra lịch sử commit (dạng cây) bao gồm tất cả các branch từ đầu đến giờ. Lưu ý: Head sẽ trở đến commit mới nhất của nhánh hiện tại bạn đang checkout.

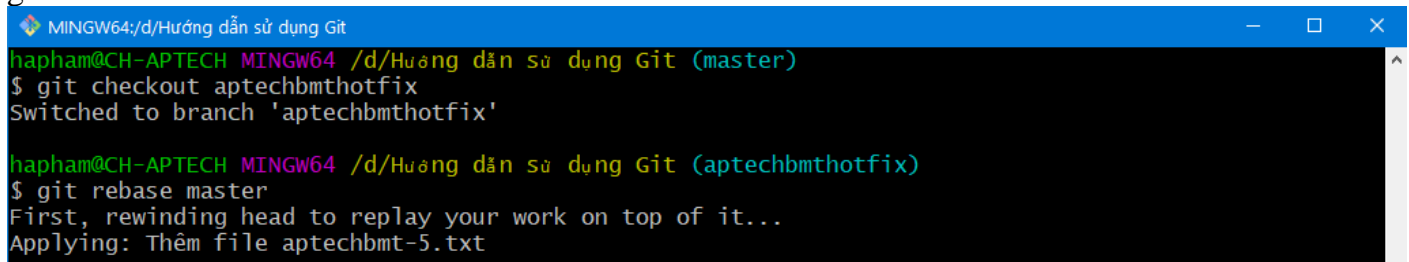
```
git log --oneline --graph --color --decorate --all
```



```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git log --oneline --graph --color --decorate --all
* d739a1c (HEAD -> master) Thêm file aptechbmt-6.txt
| * 4e5b0a0 (aptechbmthotfix) Thêm file aptechbmt-5.txt
|/
* 32c70f0 (origin/aptechbmtdev, aptechbmtdev) Thêm aptechbmt-4.txt
* e684e80 (origin/master) Merge branch 'master' of https://github.com/aptechbunmathuot/tu-hoc-git
| \
|  * fc38f37 Merge pull request #1 from aptechbunmathuot/develop
|  | \
|  |  * 795a628 Commit đầu tiên
|  |  | \
|  |  |  * 2334e00 Initial commit
|  |  |  * 96a4dfb Thêm nhiều file mới
|  |  |  * 0ced828 Thêm file mới
```

Bạn để ý hiện tại commit của nhánh master có số hiệu d739a1c đang đứng trên cùng và commit của nhánh *aptechbmthotfix* có số hiệu 4e5b0a0 đang ở ngay bên dưới. Bây giờ, tôi sẽ **checkout aptechbmthotfix** và thực hiện **rebase master** bằng lệnh:

```
git rebase master
```



```
MINGW64:/d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (master)
$ git checkout aptechbmthotfix
Switched to branch 'aptechbmthotfix'

hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (aptechbmthotfix)
$ git rebase master
First, rewinding head to replay your work on top of it...
Applying: Thêm file aptechbmt-5.txt
```

Kết quả sau khi rebase

```
MINGW64; d/Hướng dẫn sử dụng Git
hapham@CH-APTECH MINGW64 /d/Hướng dẫn sử dụng Git (aptechbmthotfix)
$ git log --oneline --graph --decorate --color --all
* e5dec9e (HEAD -> aptechbmthotfix) Thêm file aptechbmt-5.txt
* d739a1c (master) Thêm file aptechbmt-6.txt
* 32c70f0 (origin/aptechbmtdev, aptechbmtdev) Thêm aptechbmt-4.txt
* e684e80 (origin/master) Merge branch 'master' of https://github.com/aptechbunmathuot/tu-hoc-git
| \
|  * fc38f37 Merge pull request #1 from aptechbunmathuot/develop
|  |
|  | * 795a628 Commit đầu tiên
|  | /
|  |
|  | * 2334e00 Initial commit
|  |
|  * 96a4dfb Thêm nhiều file mới
|  * 0ced828 Thêm file mới
```

Bạn sẽ thấy rằng sau khi rebase commit của nhánh aptechbmthotfix có số hiệu **4e5b0a0** lúc đầu đã được git đổi sang số hiệu khác là **e5dec9e** và đặt nó ngay trên commit của nhánh master có số hiệu **d739a1c**.

Bài tập:

1. Tạo sự thay đổi trên Repo Github và thực hiện đồng bộ
 - Upload tất cả các file quản lý dự án phần mềm đã thực hiện trước đây
2. Tạo Branch cho tất cả các thành viên trong nhóm
 - Yêu cầu tất cả thành viên trong nhóm thực hiện commit trên Branch đã được phân chia
3. Hợp nhất các nhánh công việc đã được phân chia trước đây.
4. Hãy tạo ra trường hợp có xung đột trên Git và hãy giải quyết các vấn đề này