|  |  |
| --- | --- |
|  | **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM** |

**ĐỀ TÀI MÔN HỌC**

**CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM**

**TÌM HIỂU VỀ GIT HUB**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **CÔNG NGHỆ PHẦN MỀM**

Giảng viên hướng dẫn: Nguyễn Kim Hưng

Sinh viên thực hiện: TRẦN XUÂN HUY - 1615060021

TP. Hồ Chí Minh, 2019

LỜI CẢM ƠN

Đ

ể hoàn thành môn đề tài môn học này, em đã nhận được sự giúp đỡ và góp ý nhiệt tình của quý thầy và các bạn trường Đại Học HuTech Công Nghệ Thành Phố Hồ Chí Minh.

Trước hết em xin chân thành cảm ơn quý thầy và các bạn trường Đai học Hutech Công Nghệ TP. HCM, đặc biệt là thầy Nguyễn Kim Hưng và các bạn đã dành thời gian quý báu tận tình dạy bảo em trong thời gian làm đề tài môn học này.

Mặc dù đã có nhiều cố gắng để hoàn thành đề tài môn học này, tuy nhiên cũng không thể tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp của quý thầy và các bạn.

TP. HCM, Tháng 04 năm 2019

Sinh viên thực hiện

Trần Xuân Huy

# **CHƯƠNG 1: TỔNG QUAN**

**MỞ ĐẦU**

* 1. **GIỚI THIỆU ĐỀ TÀI**

Ngày nay, khi việc triển khai một dự án phần mềm vừa và lớn thì việc quản lý và kiểm soát source code đối với các lập trình viên khi customize là một việc quan trọng. Nếu việc quản lý và kiểm soát source code không có trình tự và thống nhất giữa các bộ phận sẽ phát sinh ra những khó khăn và rủi ro về sau này.

Chính vì một vài lý do được nêu ở trên và xuất phát từ những yêu cầu thực tế khi triển khai dự án phần mềm. Bên cạnh đó cũng giảm thiểu tối đa những rủi ro sự sai lệch thông tin giữa các source code và từ đó giúp cho người dùng và các bộ phận quản lý được thông tin source code một cách chắn chắn thống nhất hơn. Tìm kiếm giải pháp tốt thiết thực để đáp ứng được nhu cầu về quản lý và kiểm soát ở trên.

* 1. **LÝ DO CHỌN ĐỀ TÀI**

Lý do chọn đề tài là vừa nghiên cứu và tìm hiểu về một giải pháp cho việc quản lý source code phân tán để đáp ứng được nhu cầu kiểm soát và quản lý. Và một giải pháp đáp ứng tốt được nhu cầu này có tên gọi là GIT. Giải pháp này hiện nay đang là xu hướng và được sử dụng phổ biến rộng rãi hiện nay.

* 1. **BẢNG PHÂN CÔNG**

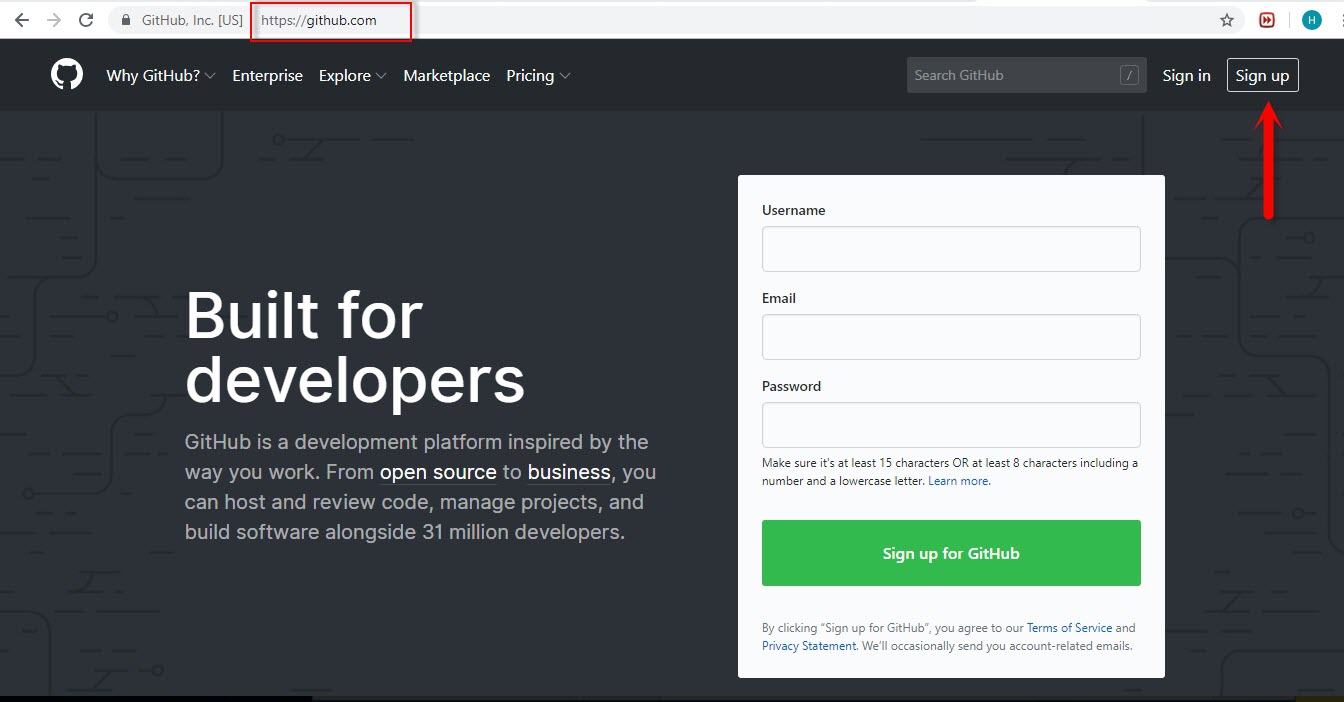
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Họ Tên** | **#** | **#** | **#** | **Viết báo cáo** |
| Trần Xuân Huy |  |  |  | Yes |

# **CHƯƠNG 02: CÀI ĐẶT VÀ CẤU HÌNH CƠ BẢN**

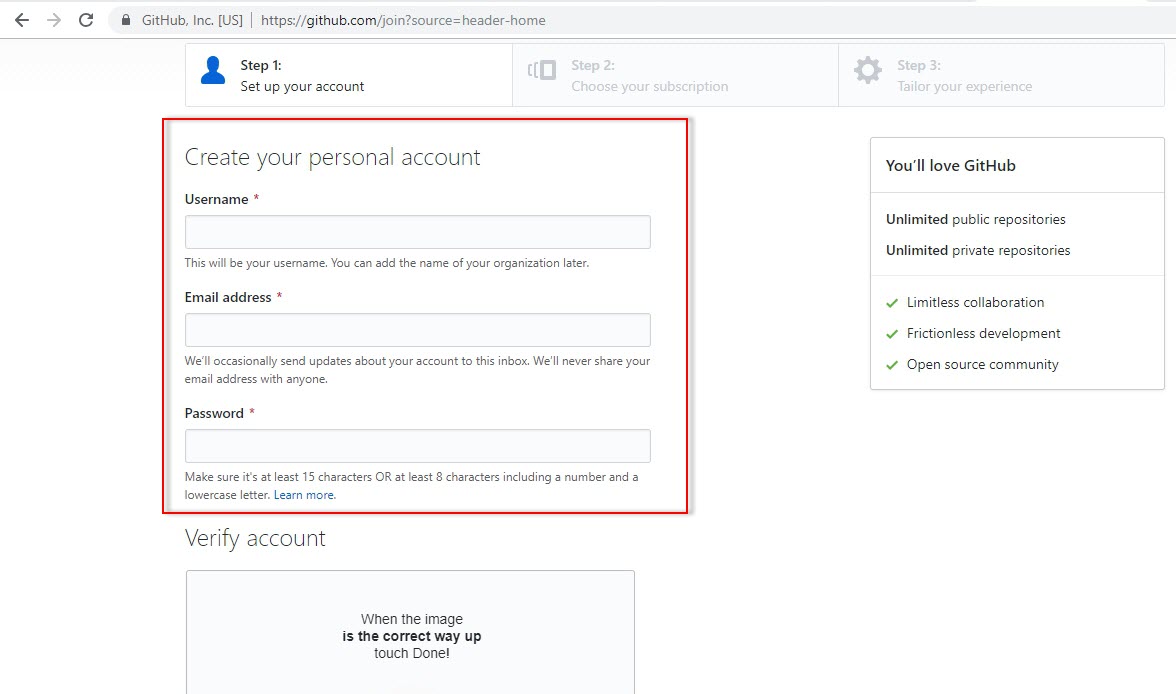
**2.1 Đăng ký tài khoản trên Git Hub**

Truy cập vào trang chủ của Git Hub <https://github.com/> rồi chọn Sign Up để đăng ký tài khoản mới.

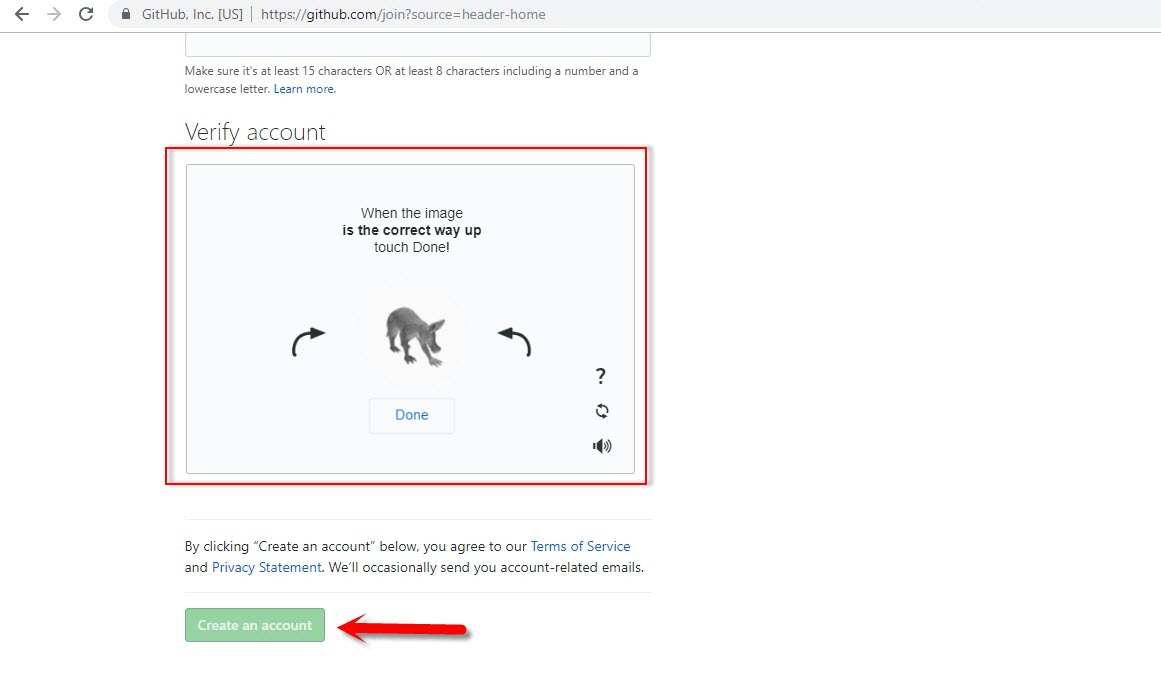
*Lưu ý: Đối với tài khoản Free cho được từ 3-5 Users và Dung lượng từ 30MB -50MB*



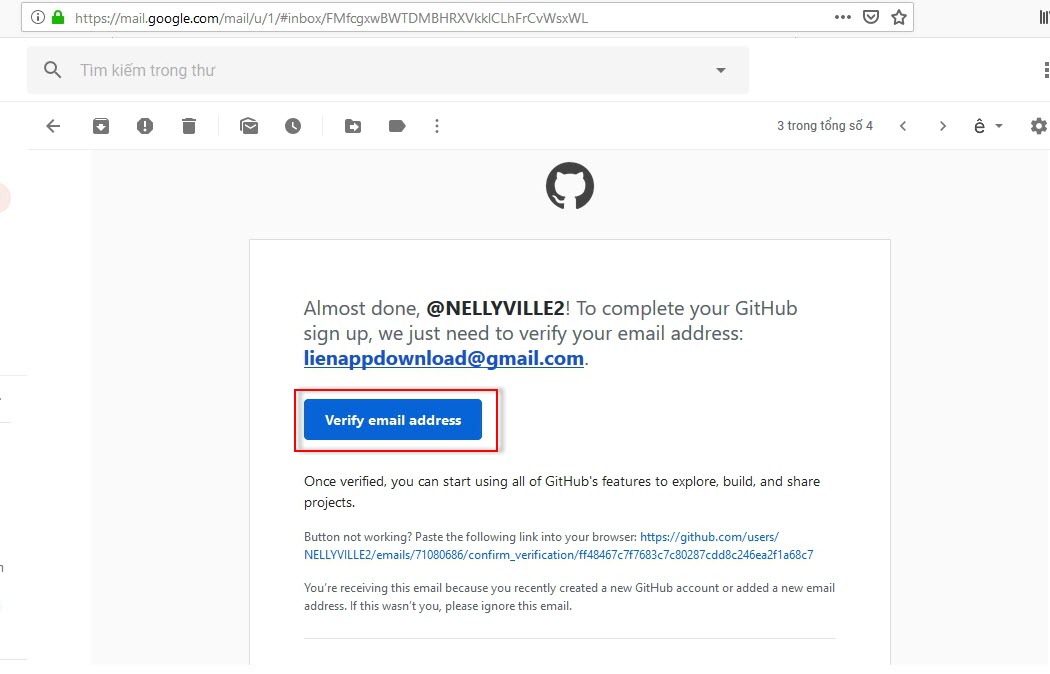
Tiếp theo điền thông tin đầy đủ để đăng ký tài khoản



Tiếp theo xác nhận thông tin tài khoản rồi nhấn **“Create an account”**

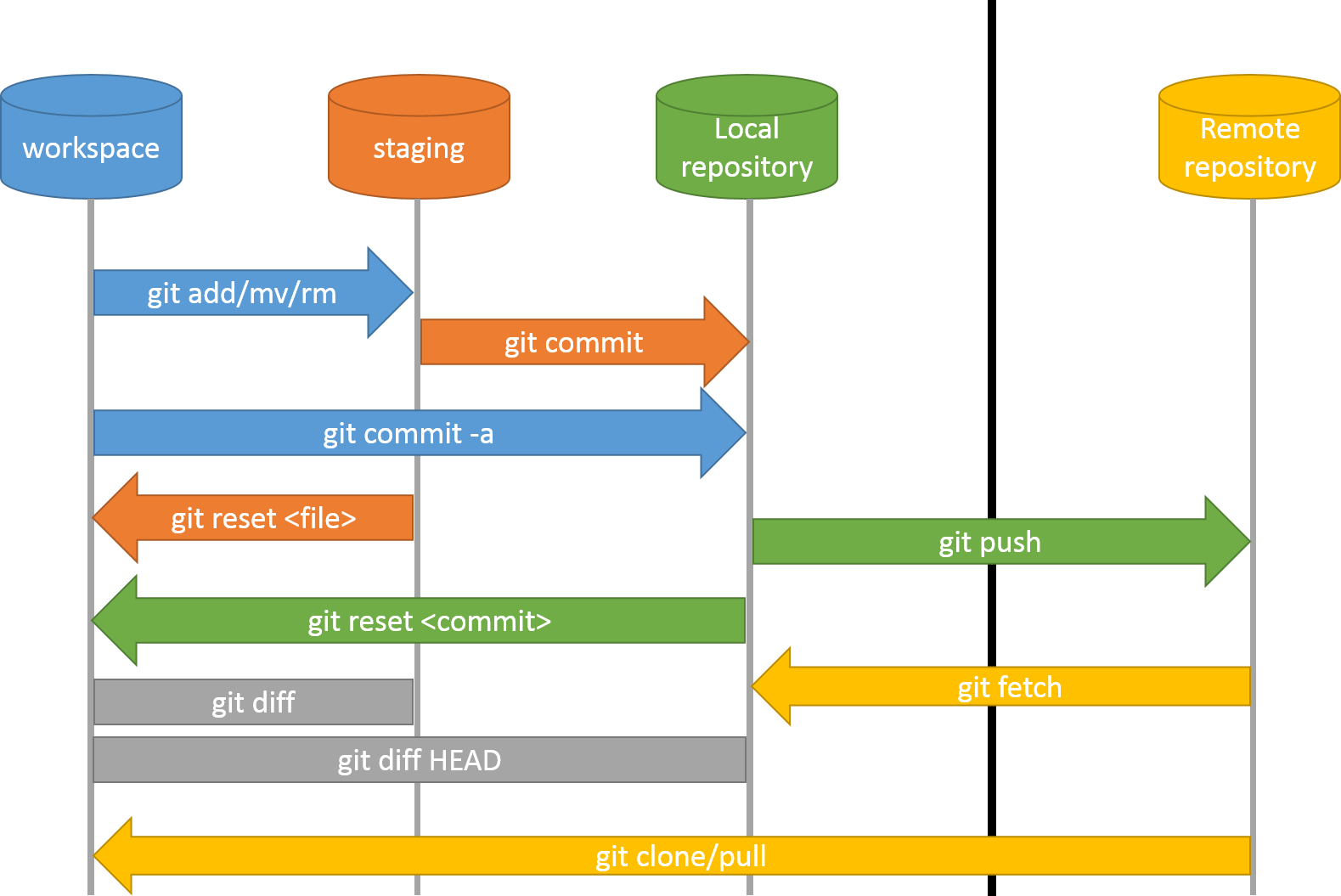


Tiếp theo vào email lúc đăng ký tài khoản để nhấn nút xác thực qua địa chỉ email.

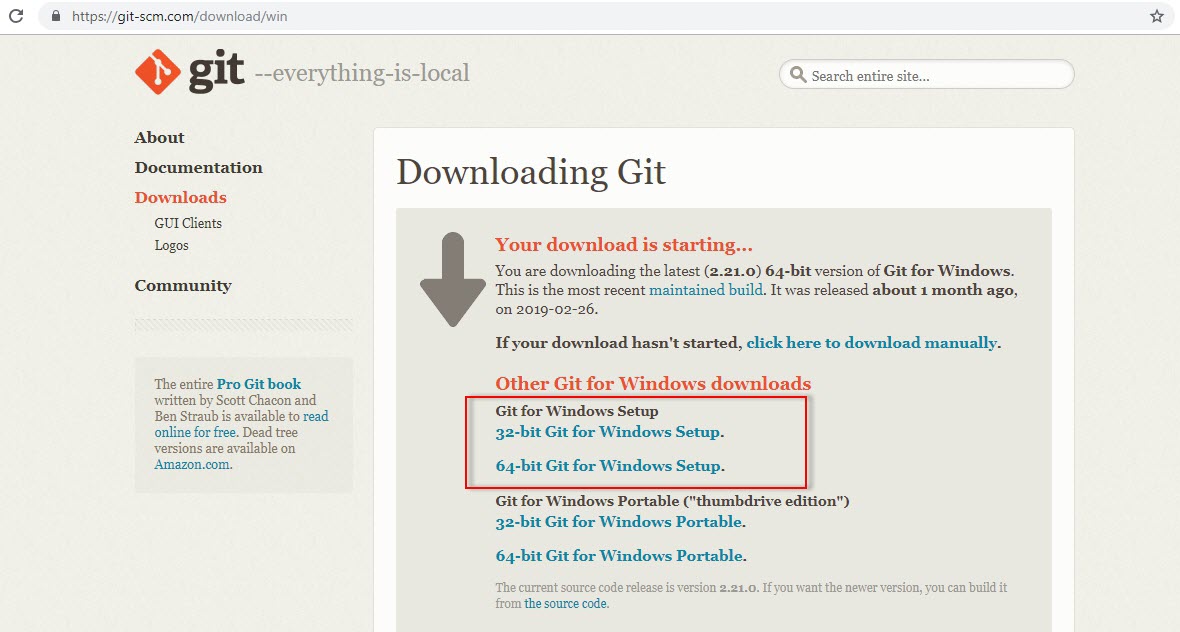


**2.2 Cài đặt GIT và kiểm tra**

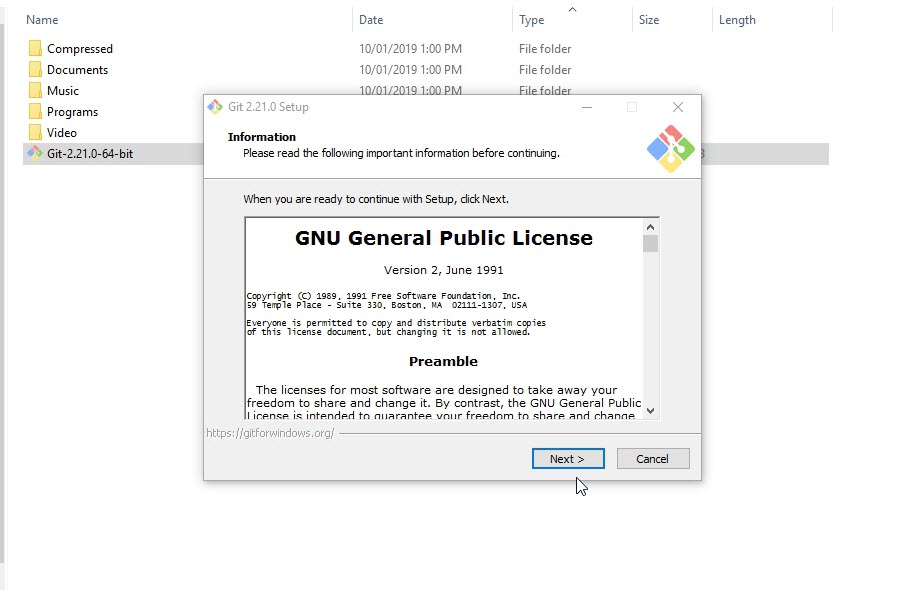
Đây là mô hình mô hình của GIT Hub cơ bản.



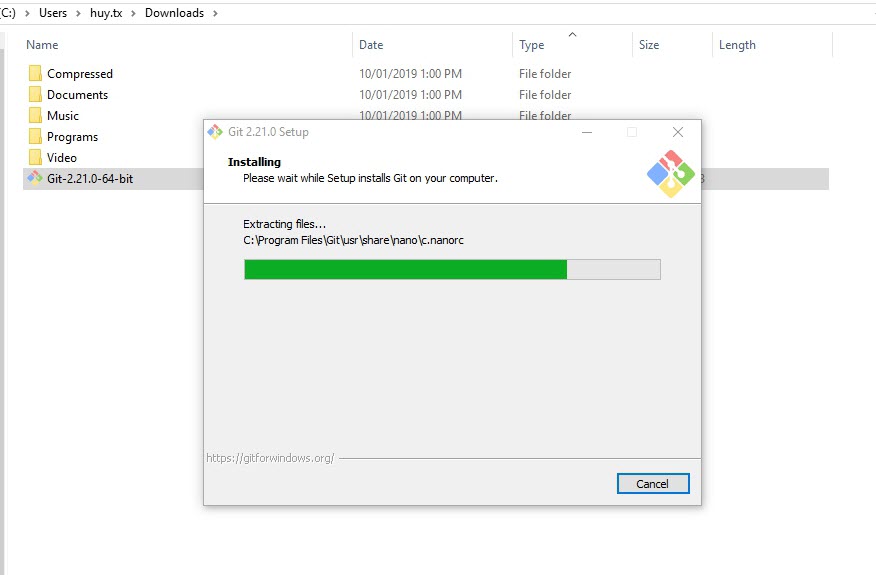
Đầu tiên vào trang web của GIT để tải về. Có thể vào trang <https://git-for-windows.io> hoặc vào Google search từ “git” rồi vào trang đầu tiên : git-scm.com tải về phiên bản phù hợp với Window 32bit hoặc 64bit



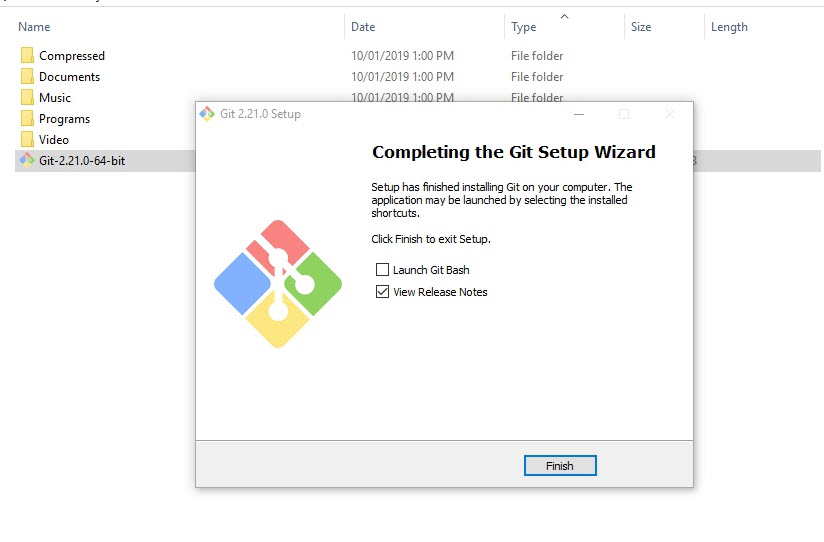
Sau khi tải về bắt đầu tiến hành cài đặt



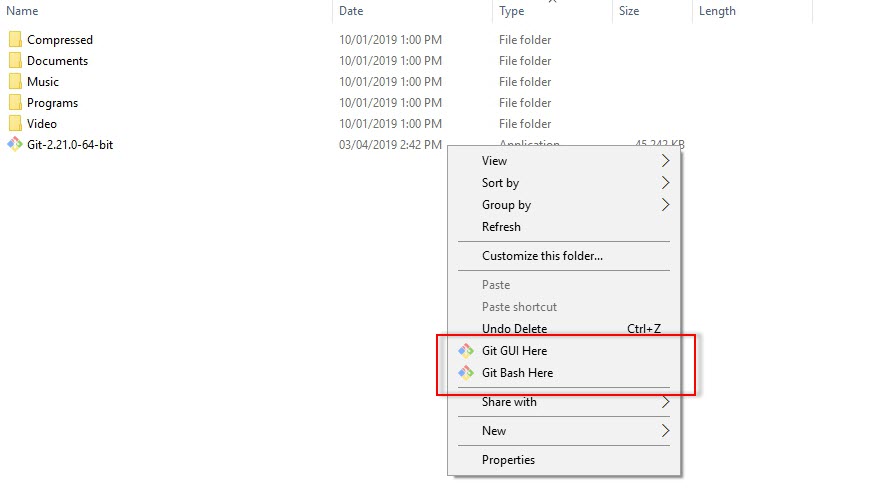
Quá trình cài cũng đơn giản và chọn Next như cài đặt các phần mềm bình thường.



Đợi cho tới khi quá trình cài đặt hoàn tất

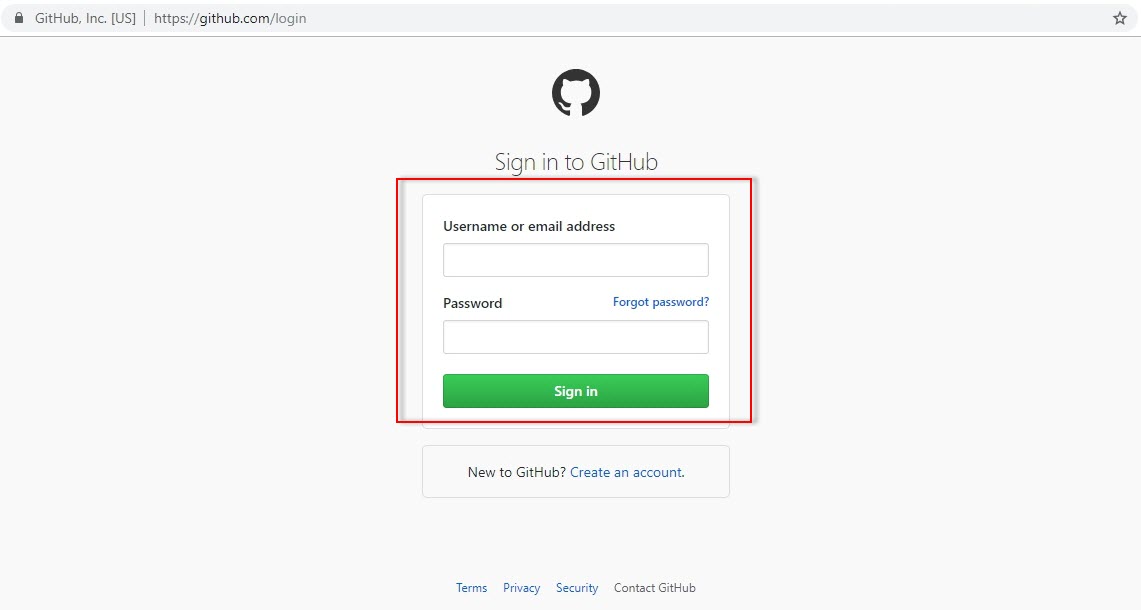


Sau khi cài đặt xong tiến hành kiểm tra đã có GIT chưa bằng cách Click phải chuột xem có 2 mục **GIT GUI Here và GIT Bash Here**



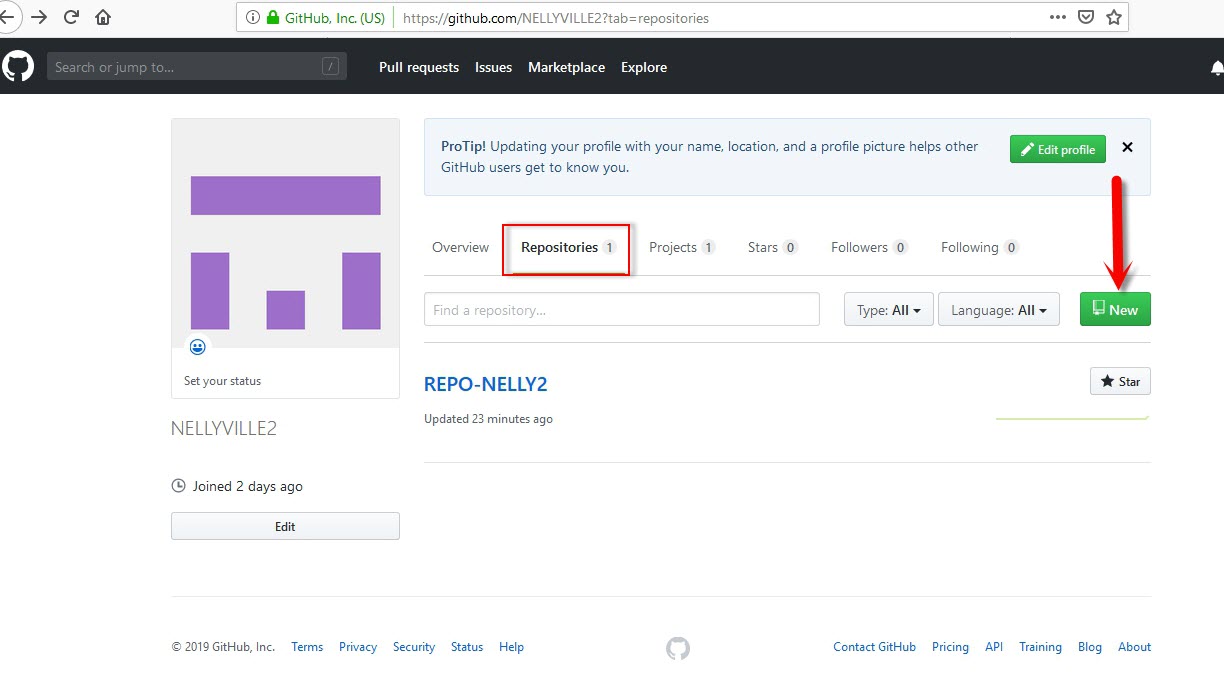
**2.3 Đăng nhập vào GIT và tạo Repository**

Đầu tiên phải Sign in vào Git Hub. Điền thông tin tài khoản đăng ký ban đầu

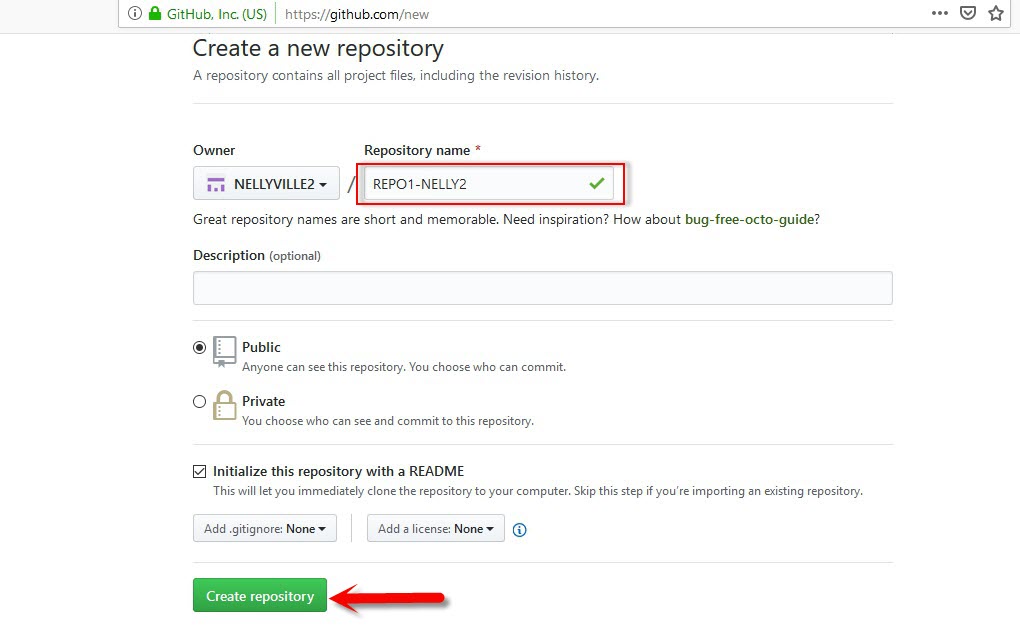


Sau khi đăng nhập vào. Tiếp theo tạo 1 Repository

Repository: Như là 1 kho chứa Source các loại file.



Tiếp theo đặt tên cho Repository chon cơ chế Public hay Private rồi cuối cùng nhấn **Create repository**



**2.4 Cấu hình thông tin người dùng**

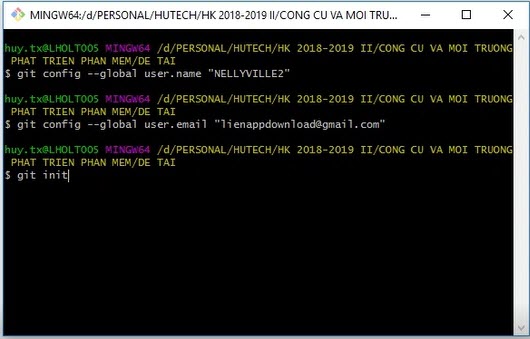
Đầu tiên mở câu GIT Bash Here bằng cách Click phải -> GIT Bash Here

Tiếp theo dùng các câu lệnh để cấu hình thông tin người dùng

**$ git config –global user.name** “Tên tài khoản”

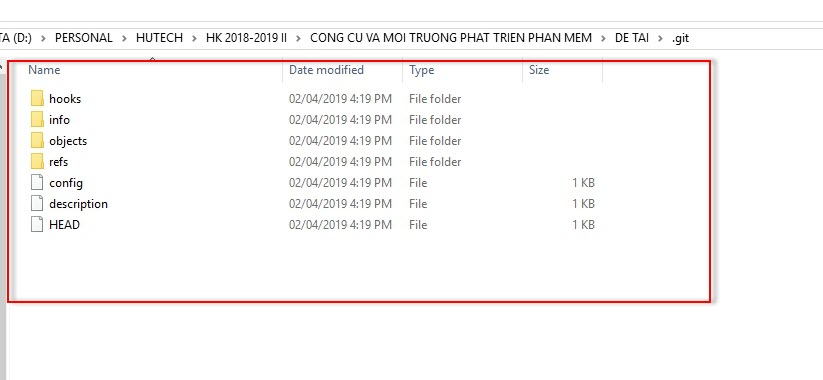
**$ git config –global user.email** “Địa chỉ email dùng để đăng ký tài khoản”

**$ git config init:** Dùng để tạo ra những file cần thiết



Tiếp theo kiểm tra dưới Local đã có các file cần thiết chưa.

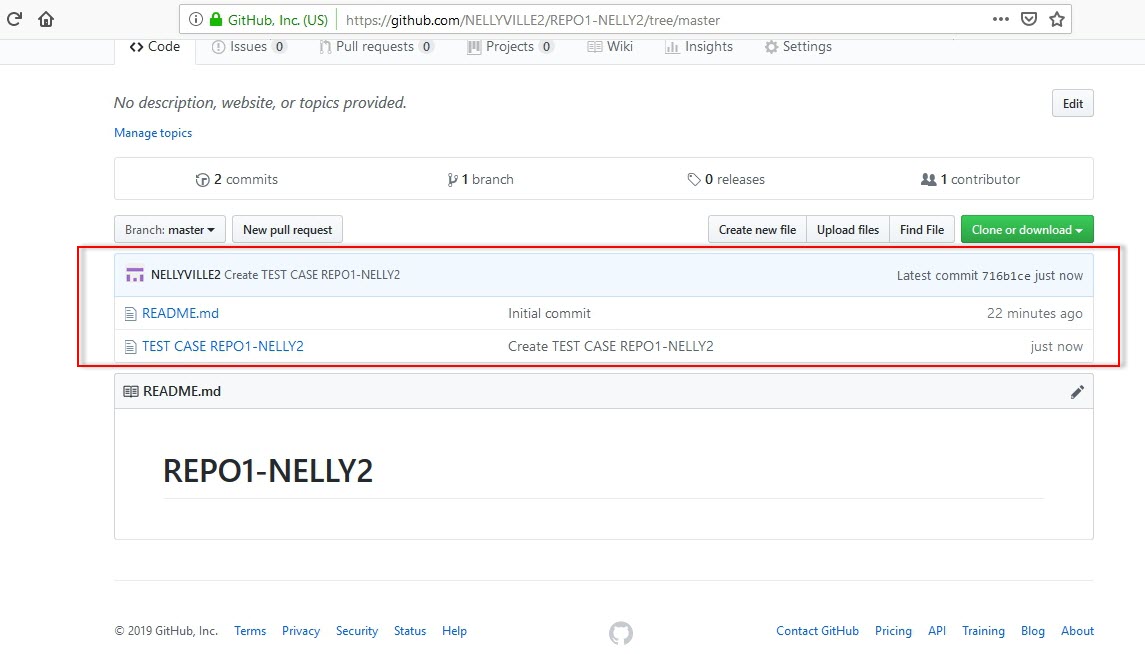
*Lưu ý phải mở chức năng xem file ẩn trong Window mới thấy được các file cấu hình của GIT*



**2.5 Cấu hình Clone Repository về tại Local**

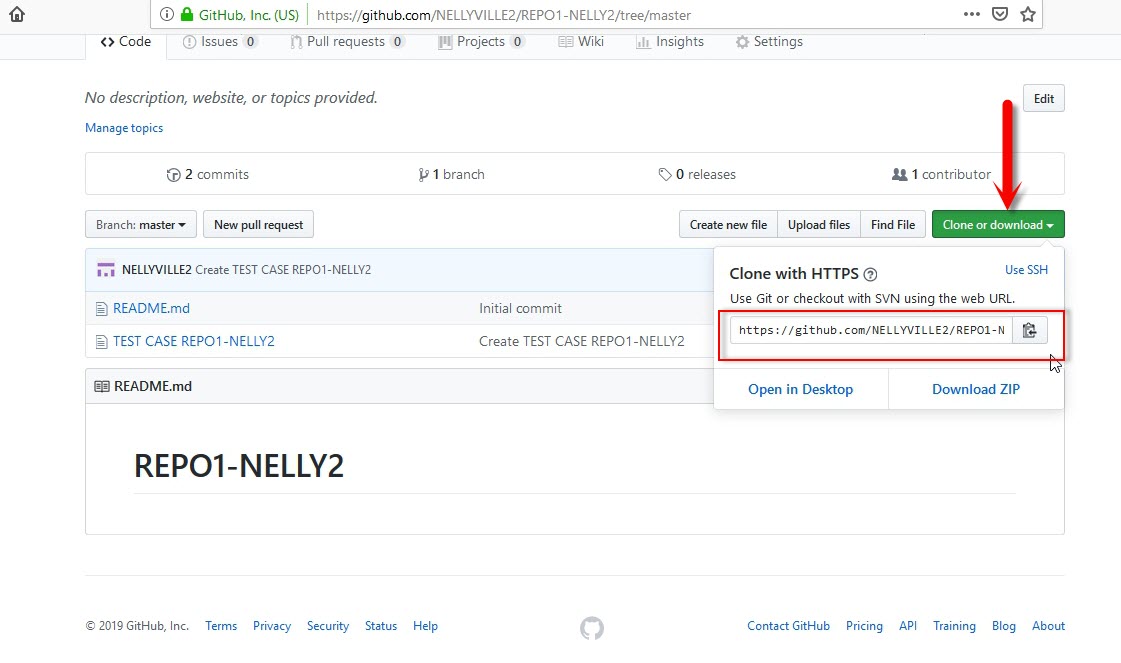
Xem trên Repository hiện tại có các file

Ví dụ: README.md và File Text

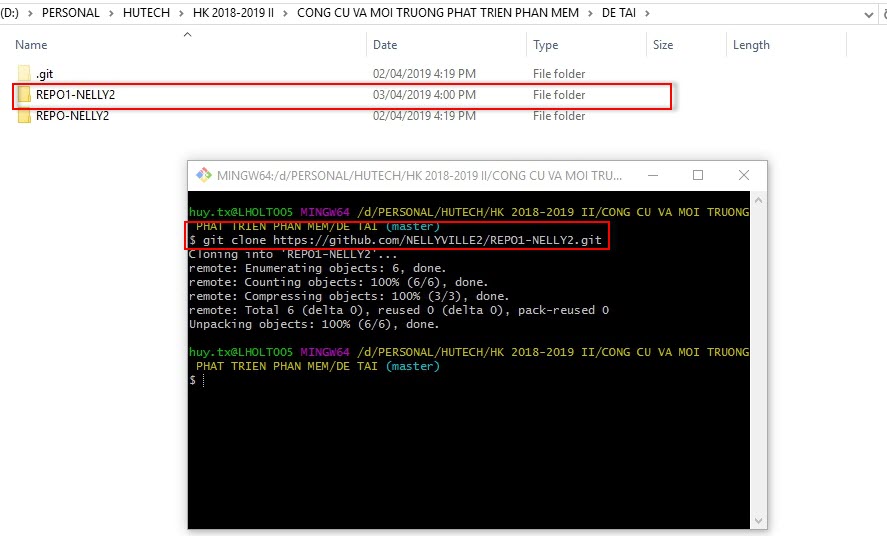


Bắt đầy thực hiện câu lệnh để lấy các File về tại Local

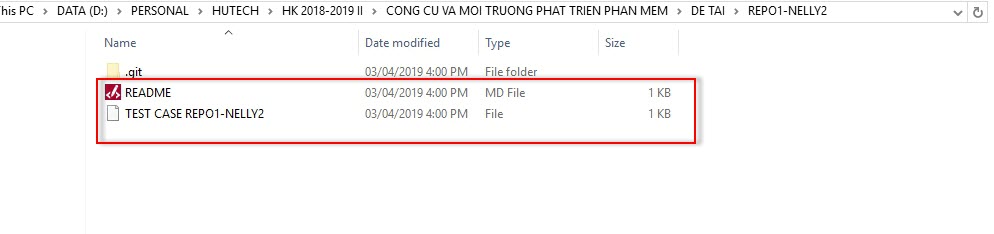
Đầu tiên lấy đường Link của Repository trên Git Hub.



Tiếp theo thực hiện câu lệnh và dán đường link của Repository vào để lấy các File bên trong của Repository remote về tại Local



Sau khi Clone về thành công. Sẽ kiểm tra các nội dung bên trong Local có giống như Repository trên Server không

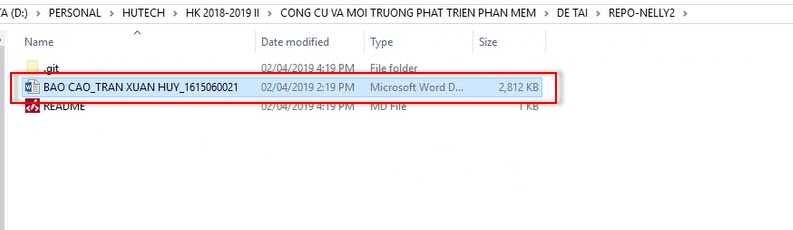


Kết quả nội dung giống nhau giữa Local và trên Server

# **CHƯƠNG 3: THỰC THI CÁC CÂU LỆNH QUẢN LÝ**

**3.1 Add source file lên Repository server**

Trong thư mục Repository vừa được Clone về tại local. Chép một file nào đó vào bên trong thư mục Reposoroty này.



Tiếp theo tiến hành thực hiện đưa lên Server

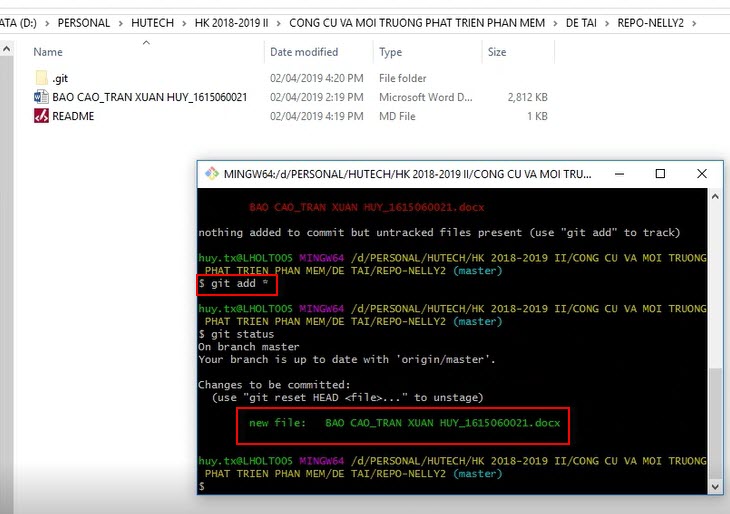
*Lưu ý: Trước khi thực hiện đưa lên Server cần phải kiểm tra đường dẫn thư mục hiện tại đúng chưa. Nếu chưa đúng cần phải chuyển thư mục bằng cách dùng câu lệnh cd\_đường dẫn.*

**

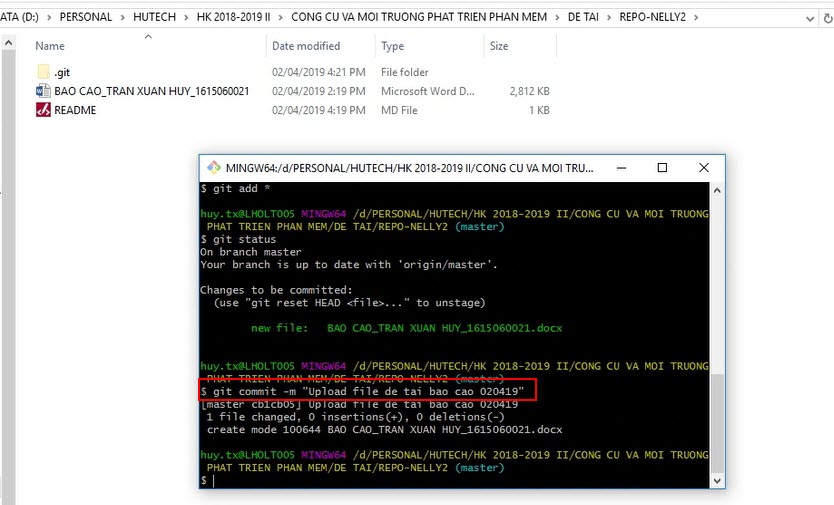
Tiếp theo dùng câu lệnh **$git status** để trạng thái các file bên trong Repository tại Local.

Nếu file hiển thị màu đỏ là chưa được Add file. Nên để Add được file dùng tiếp câu lệnh **$git Add tên file hoặc $git Add \***

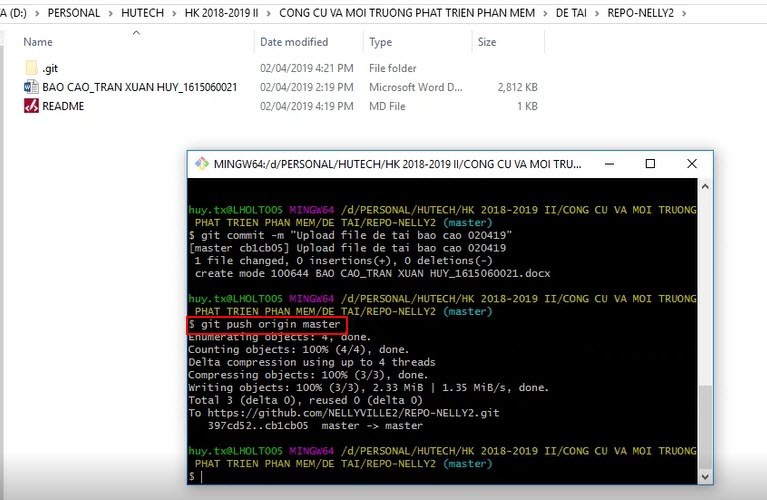
Sau khi thực hiện câu lệnh **$git add**. Các file nếu OK sẽ được chuyển trạng thái màu xanh.



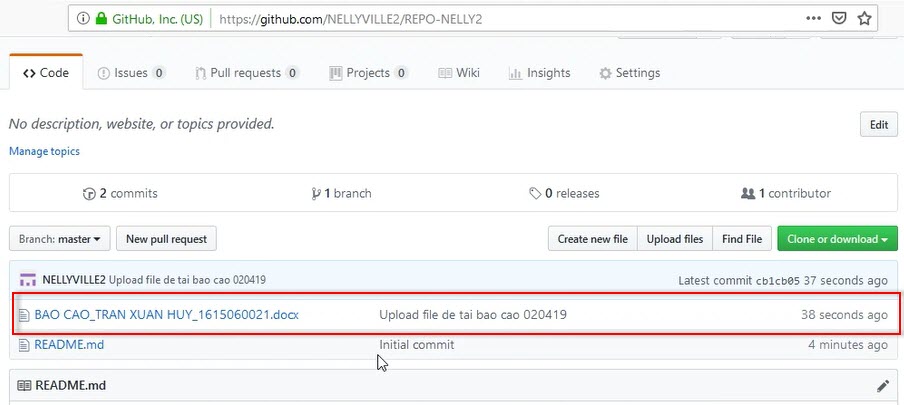
Tiếp theo để đưa các file đã được Add lên trên “Staging Area”. Tiến hành thực hiện câu lệnh **$git commit –m “Nội dung upload”**



Sau khi đưa lên “Staging Area” thành công. Tiến hành thực hiện đẩy lên Server bằng câu lệnh **$git push origin master**



Kiểm tra trong Reposiroty trên server Git Hub xem source file đã có chưa



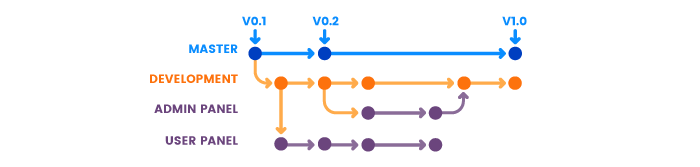
Tới đây như vậy xem như là đã xong phần này.

**3.2 Branch và lưu ý**

Tình huống trong 1 Source Project. Muốn tạo ra một phiên bản thử nghiệm với source đang làm việc trong Working tree hiện tại mà không ảnh hưởng đến Code hiện tại. Vì vậy không thể Clone một Reposotiry khác rồi thực hiện chỉnh sửa source dẫn đến tình trạng mất công và khó đồng bộ.

Để giải quyết tình huống này. Ở đây sẽ sử dụng kỹ thuật phân nhánh trong GIT.

Đây là mô hình hoạt động của Nhánh



Đầu tiên trước khi tạo nhánh. Thực hiện kiểm tra trong Repository đó đã có những nhánh nào với câu lệnh:

**$git branch**



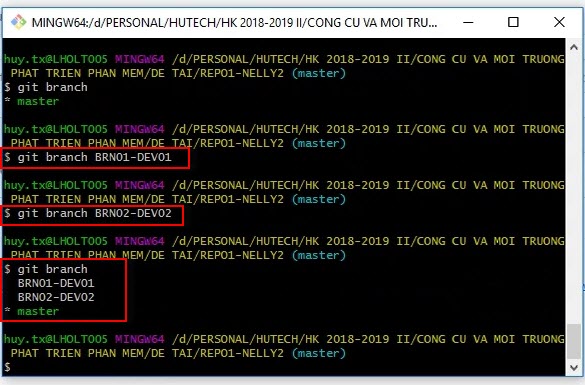
Sau khi thực hiện câu lệnh sẽ trả về kết qua các nhánh hiện đang có. Ở đây chỉ có môt nhánh duy nhất là Master. Nhánh Master này như là một thân cây chính sẽ chứa toàn bộ Source chính

Tiếp theo để tạo nhánh thực hiện câu lệnh $git branch TenNhanh. Ở đây sẽ tạo ra 2 nhánh BRN01-DEV01, BRN02-DEV02

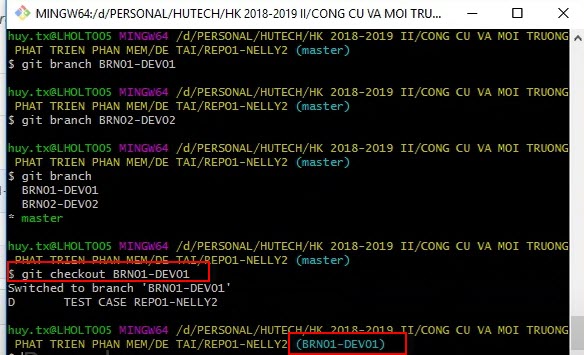
**$git branch BRN01-DEV01**

**$git branch BRN02-DEV02**

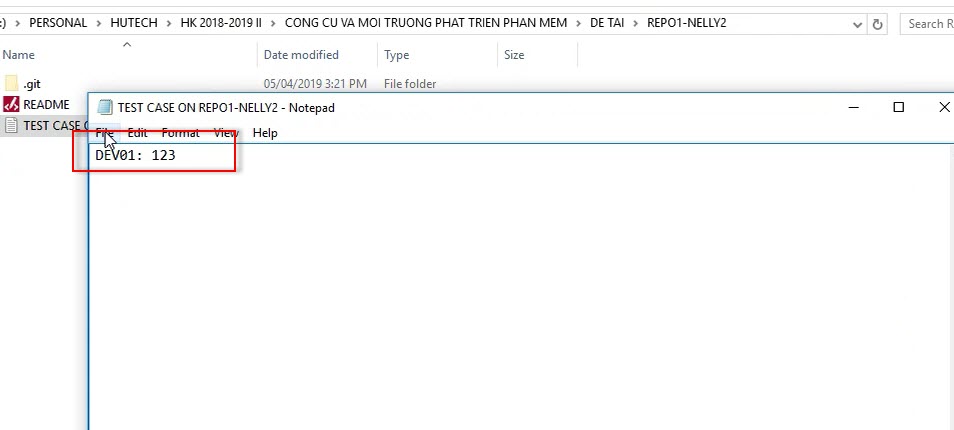
Sau khi thực hiện xong câu lệnh tao nhánh. Thực hiện câu lệnh xem nhánh như dưới. Ở đây khi đang ở trên nhánh nào thì sẽ có màu xanh tại nhánh đó.



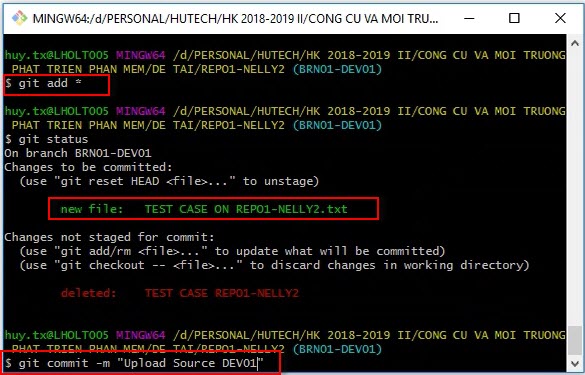
Tiếp theo tiến hành thực hiện công việc trên nhánh mới tạo. Để chuyển qua nhánh khác thực hiện câu lệnh **$git checkout TenNhanh**

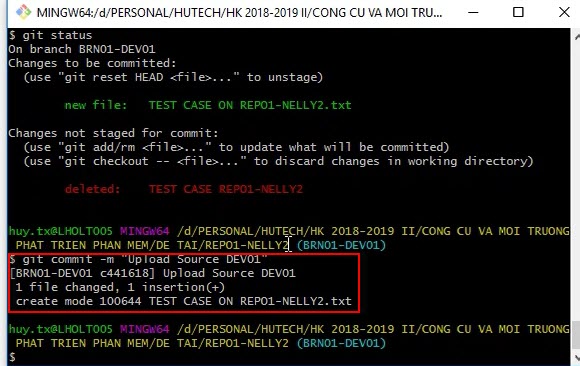


Thực hiện việc chỉnh sửa nội dung trên nhánh BRN01-DEV01



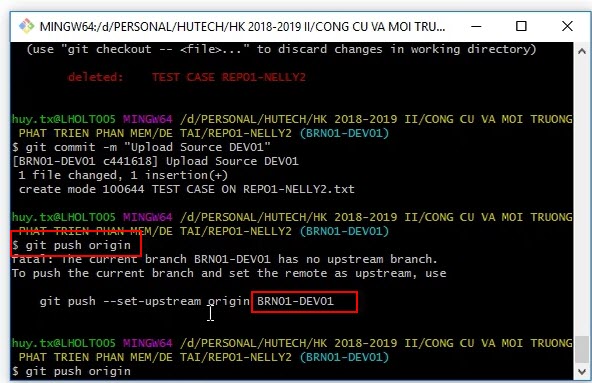
Thực hiện việc Add File, Commit File





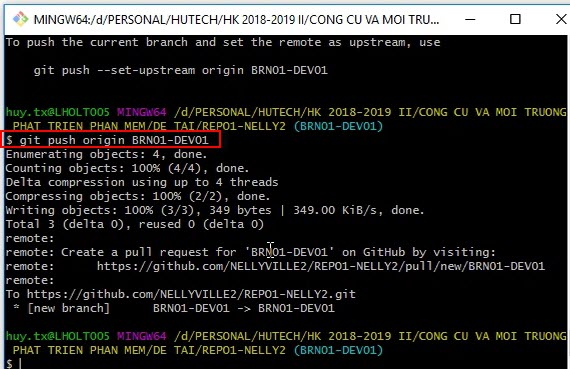
Tiếp theo thực hiện việc Push lên Repository Remote Server. Ở đây lưu ý khi Push lên Reposiroty Remote theo nhánh thì khi thực hiện câu lệnh Push phải thay đổi tên nhánh.

Như trường hợp bên dưới không cho Pust lên vì đang ở nhánh BRN01-DEV01 mà Push lên nhánh Master là không được.

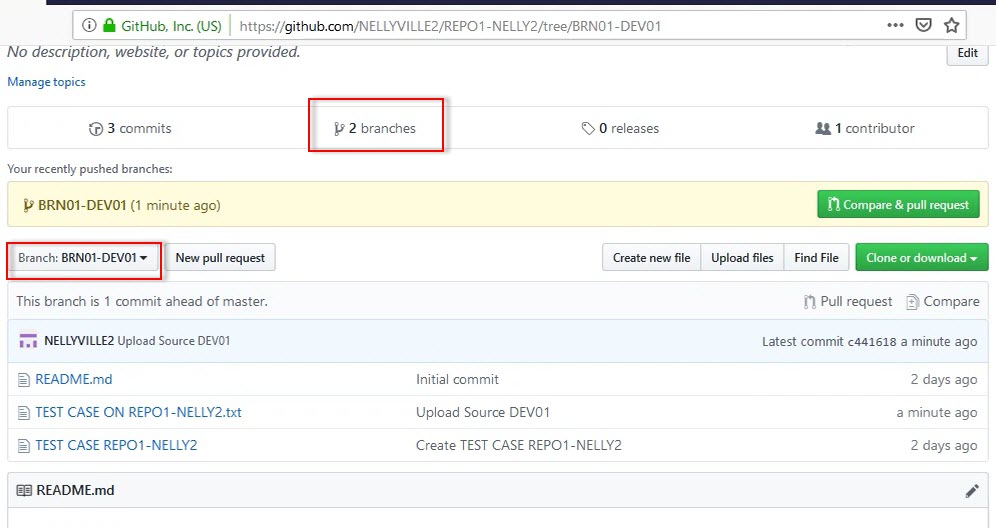


Vì vậy muốn Push lên đúng nhánh phải thực lại câu lệnh đúng nhánh.

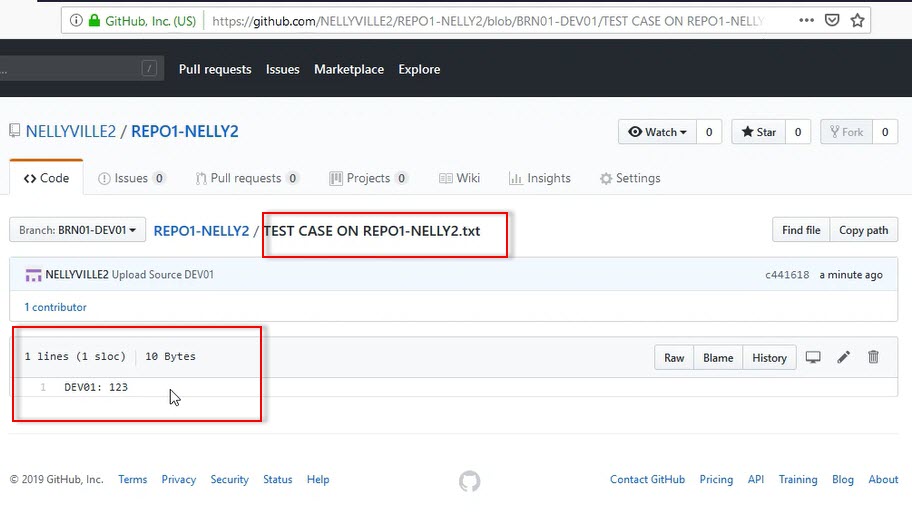
**$git push origin BRN01-DEV01**



Lúc này kiểm tra trên Repository Remote thấy xuấy hiện có nhánh mới vừa mới được Push lên.

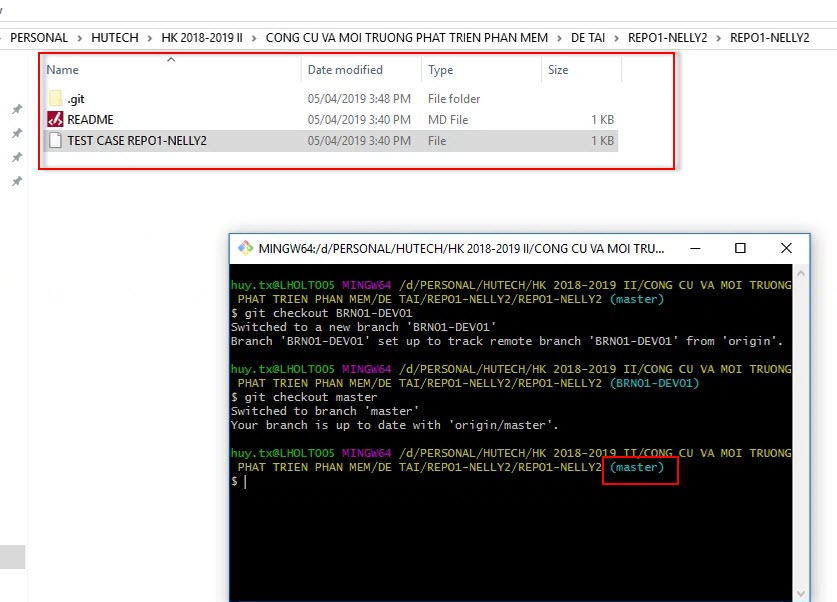


Với nội dung của nhánh BRN01-DEV01

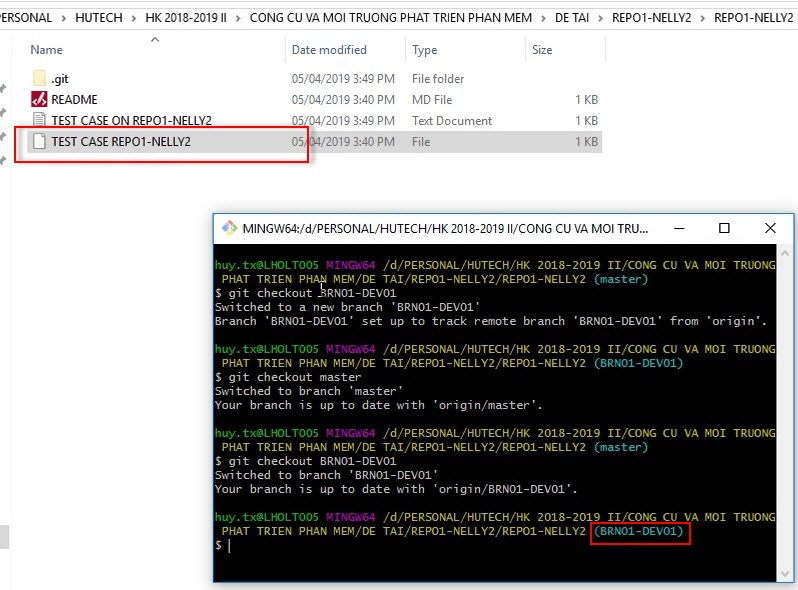


Lưu ý thêm là trong Repository, khi đứng ở nhánh nào thì source nó sẽ đi theo nhánh đó.

Như trường hợp ở đây đang đứng ở source Master.



Chuyển sáng nhánh BRN01-DEV01 thì Source khác. Nên cần phải xem mình đang đứng ở nhánh nào để khi làm việc Working tree không bị nhầm lẫn.



**3.3 Giải quyết xung đột**

Tình huống xung đột xảy ra trên 2 tránh cùng chỉnh sửa trên một source file. Rồi đã thực hiện Add và Commit file.

Đầu tiên trên nhánh BRN01-DEV01. Thêm nội dung vào “Branch:01” vào trong file text.



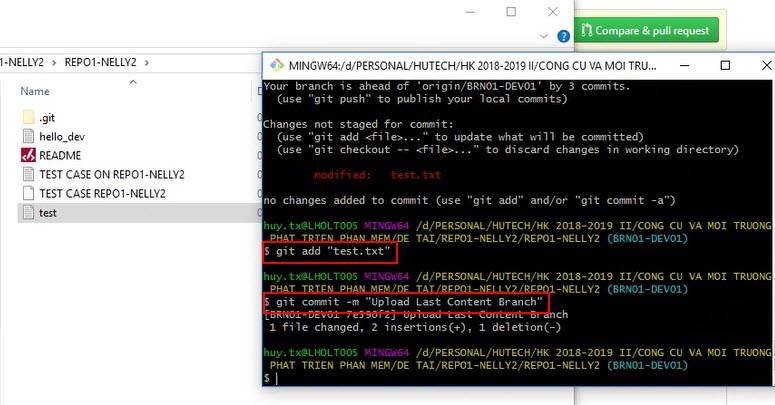
Tiếp theo tiến hành Commit File theo trình tự. Kiểm tra tình trạng file thay đổi với lệnh **$git status**



Tiếp theo thực hiện việc Add file lên “Staging Area” và Commit Local với 2 câu lệnh trên nhánh BRN01-DEV01

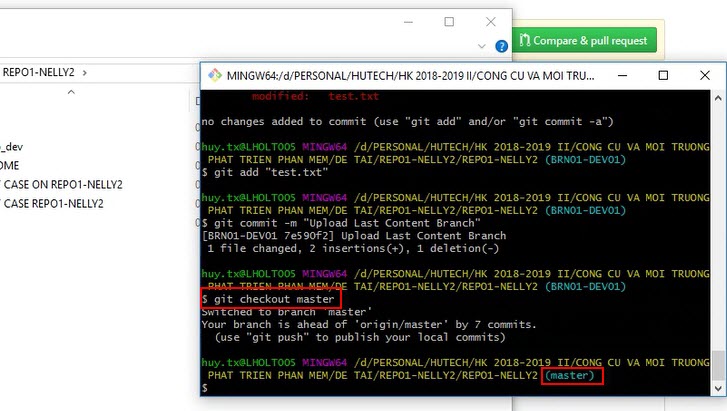
**$git add “test.txt”**

**$git commit –m “Upload Last Content Branch”**



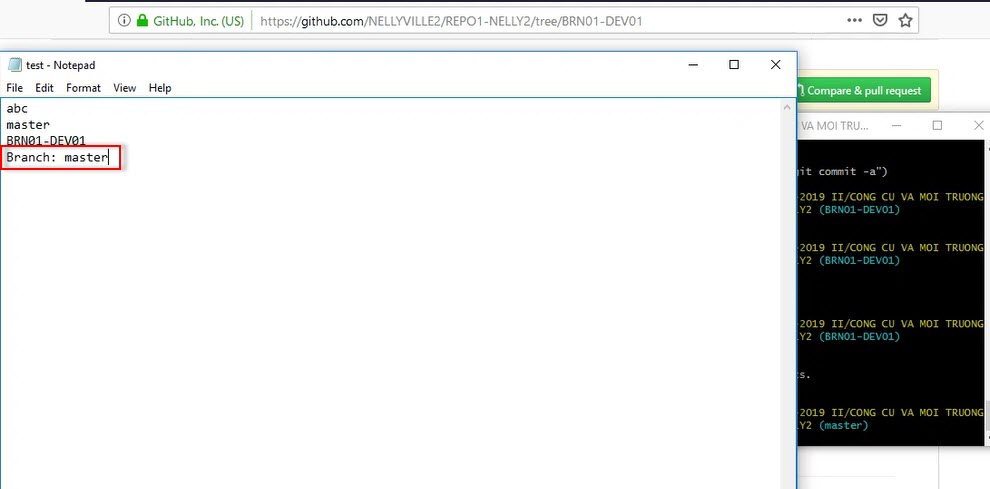
Tiếp theo chuyển qua nhánh Master cũng để thực hiện bổ sung vào file text

Dùng câu lệnh **$git checkout master**



Tiếp theo mở file text lên vào thêm nội dung vào bên trong Branch : master

***Lưu ý:*** *Khi chuyển nhánh sẽ thấy nội dung file Text ban đầu khác với nôi dung file Text bên nhánh BRN01-DEV01. Bởi vì khi thực hiện chuyển nhánh sẽ load các source file từ trên “Staging Area” về trước khi Push lên Server. Đây là điểm khác biệt với bên SVN*

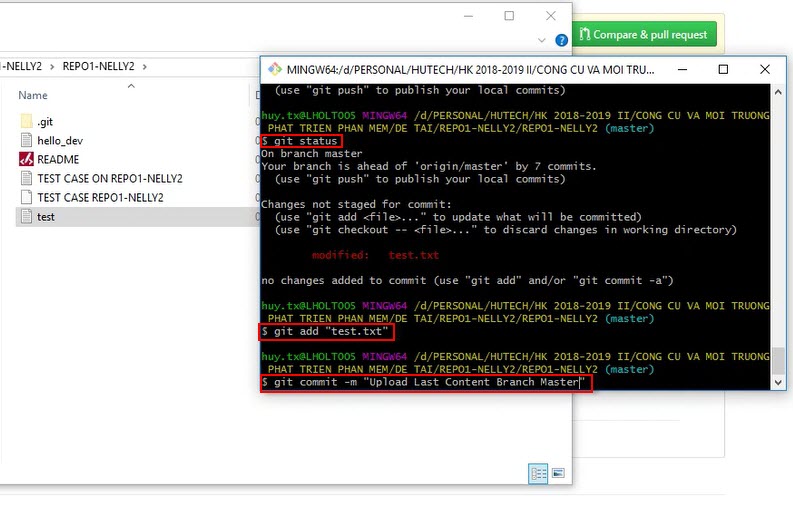


Tiếp theo thực hiện các câu lệnh để Add và Commit Local tại nhánh Master

**$git status**

**$git add “test.txt”**

**$git commit –m “Upload Last Content Branch Master”**



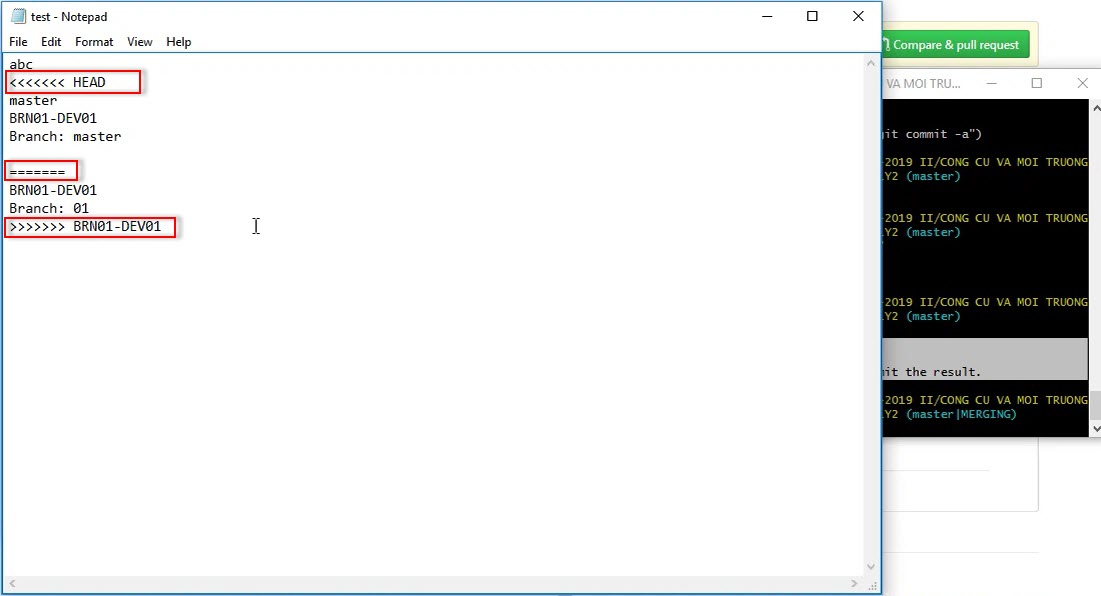
Sau khi thêm nội dung vào file text trên 2 nhánh. Tình huống xảy ra là sẽ Merge từ nhánh BRN01-DEV01 vào nhánh Master. Thực hiện câu lệnh:

**$git merge BRN01-DEV01**

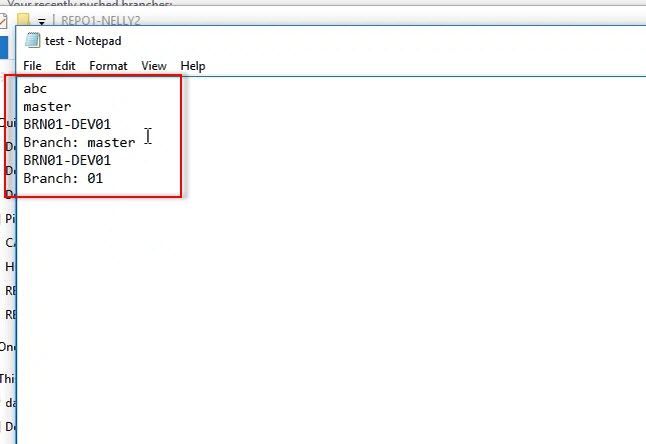
Sau khi thực hiện lệnh Merge. Lúc này xảy ra tình trạng xung đột giữa 2 nhánh khi thực hiện trên cùng 1 source file. Và lúc này trạng trên nhánh Master vẫn đang để trạng thái là đang gọp (…master|MERGING)



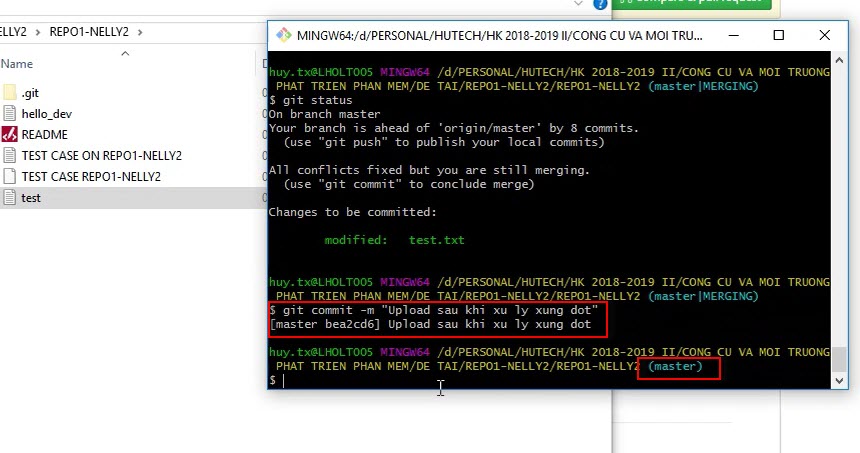
Mở file Text sẽ thấy nội dung 2 nhánh được ngăn cách bởi các ký hiệu như trong hình màu đỏ



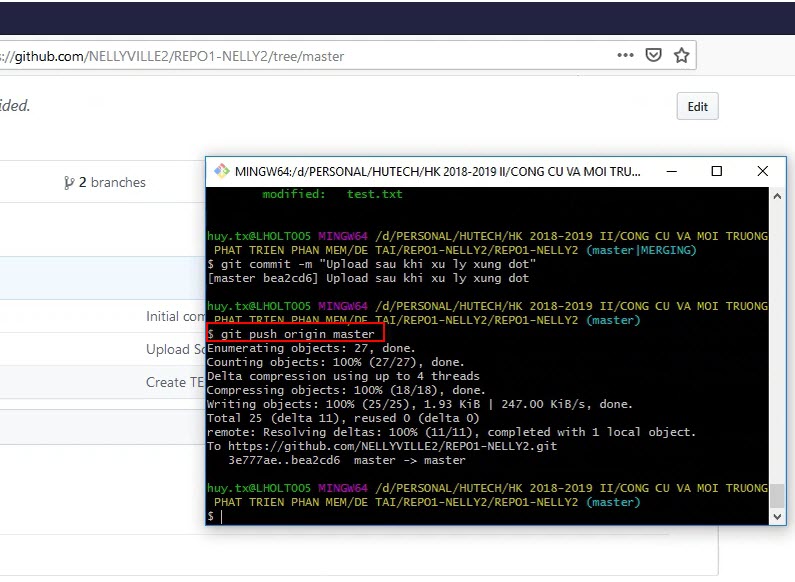
...và để giải quyết tình trạng xung đột này. Thực hiện thủ công bằng cách xóa các dòng màu trong ô màu đỏ và giữ lại nội dung của 2 nhánh lại.



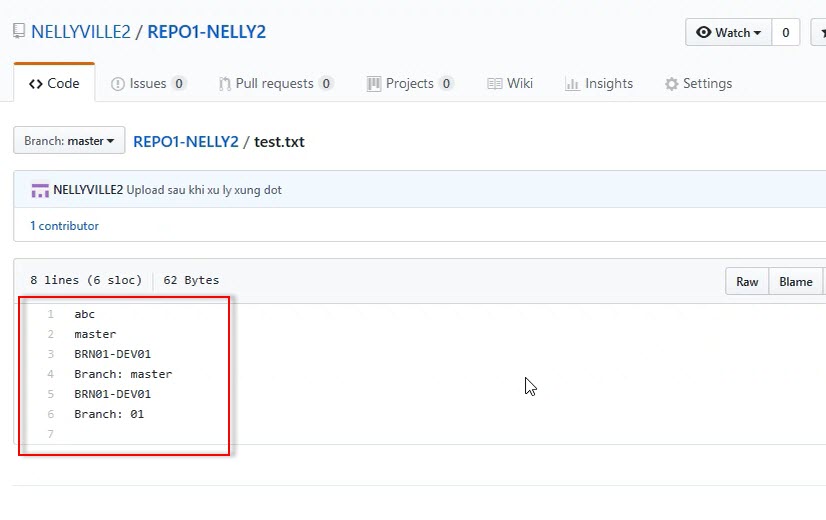
Tiếp theo tiến hành thực hiện các lệnh Add File, Commit lại. Thì lúc này trên nhánh Master không còn hiển thị trạng thái đang Merging như lúc đầu.



Và Push lên Repository Remote.

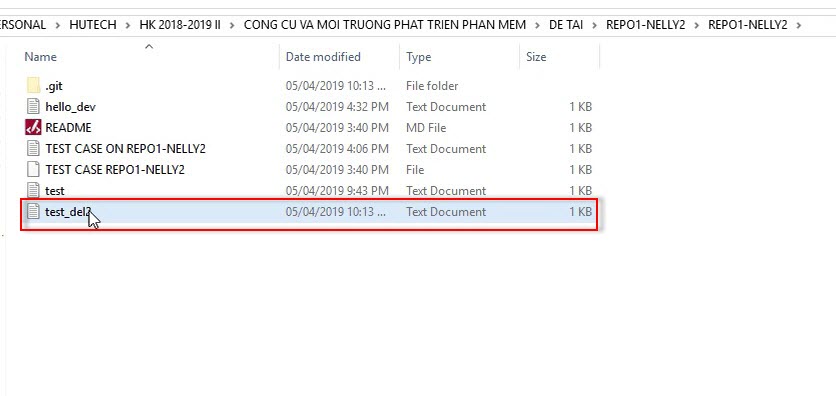


Nội dung file text của 2 nhánh được xử lý xung đột gộp lại hoàn tất.

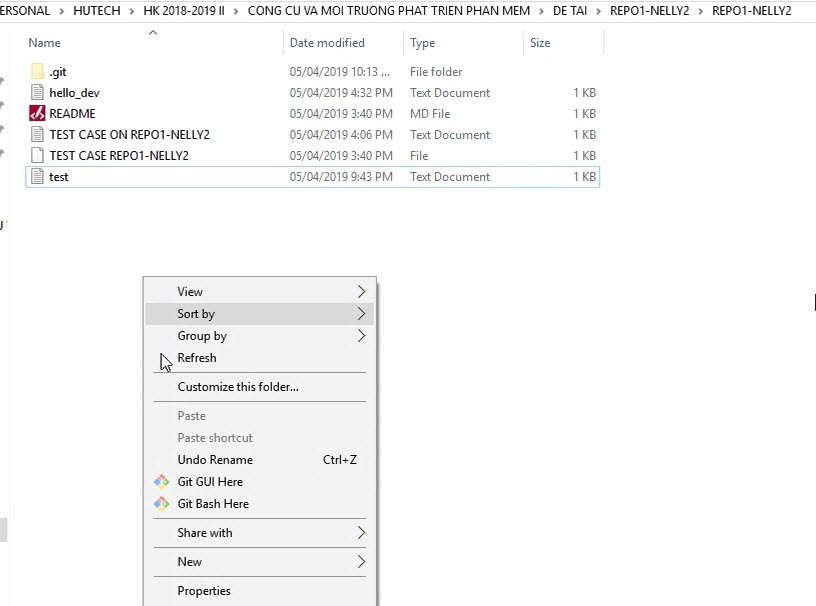


**3.4 Phục hồi source file bị xóa**

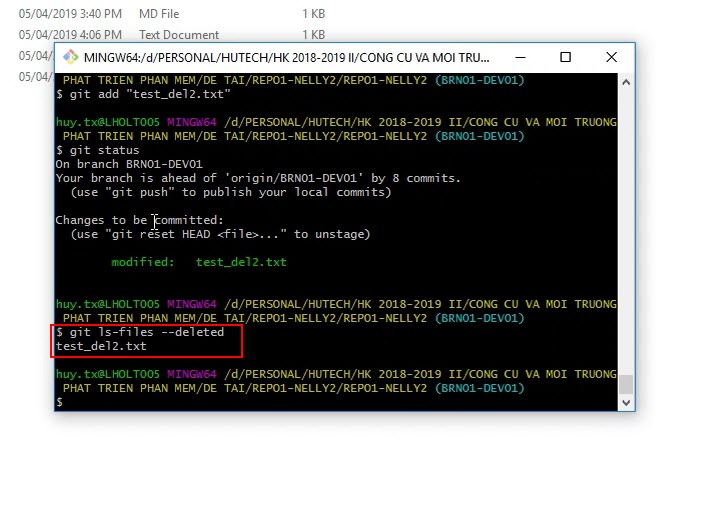
Tình huống source file bị xóa mất khỏi trong Repository tại Local.



Và không Undo lại được.

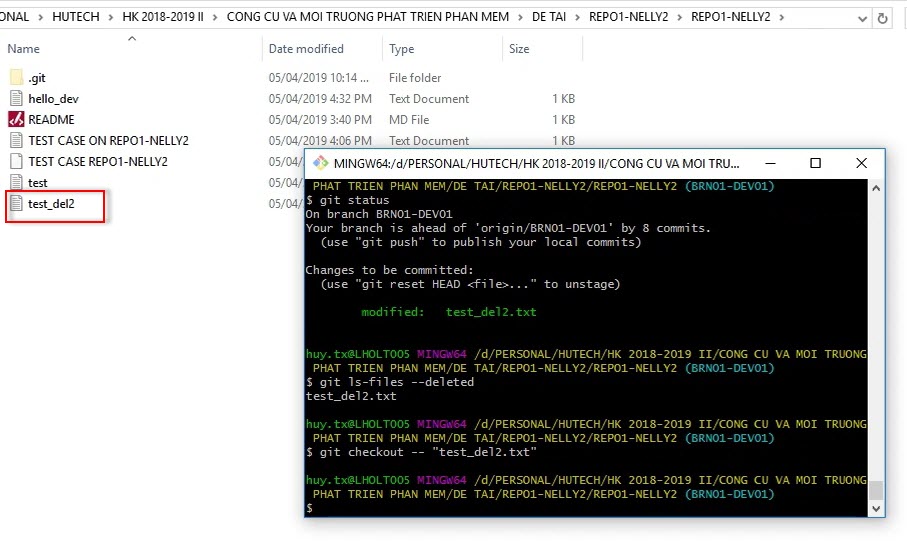


Để phục hồi lại các file nào đã lỡ xóa. Sử dụng câu lệnh để liệt kê các file nào đã bị xóa trong Repository **$git ls-files --deleted .** Lúc này sẽ hiển thị những file nào đã bị xóa.



Sau khi xác định được file nào cần được phục hồi lại. Sử dụng câu lệnh để phục hồi lại file

**$git checkout –“test\_del2.txt”**

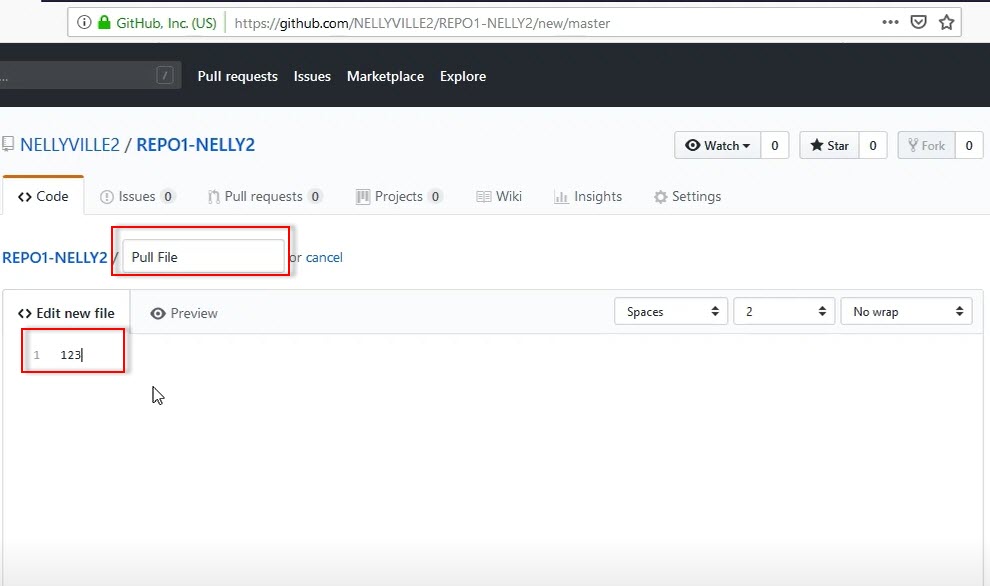


Và như vậy. File test\_del2 đã được phục hồi lại trong Repository ở nhánh BRN01-DEV01

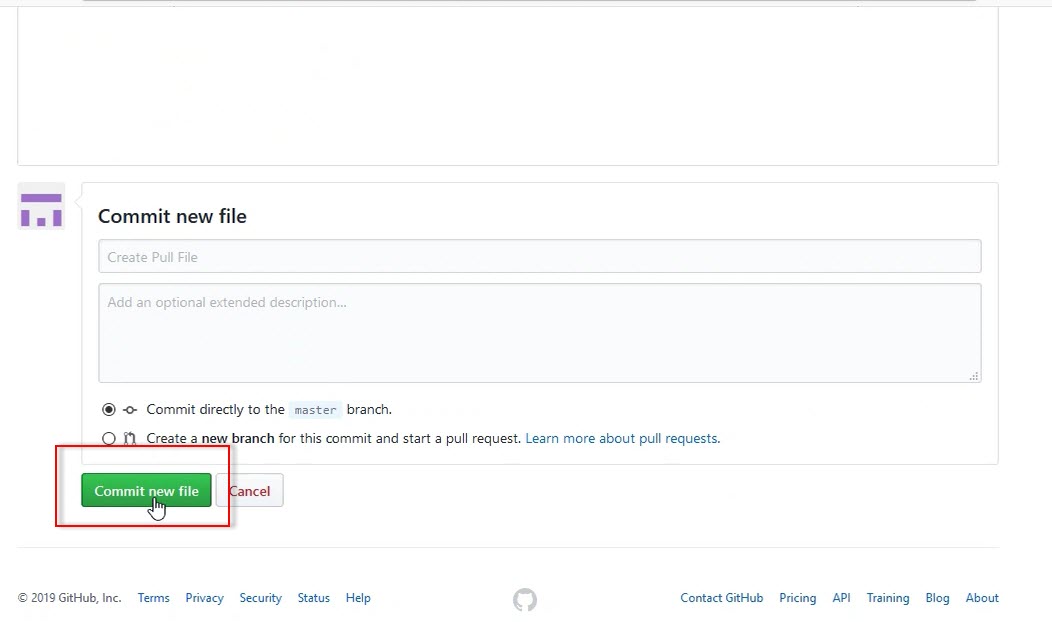
**3.5 Pull files**

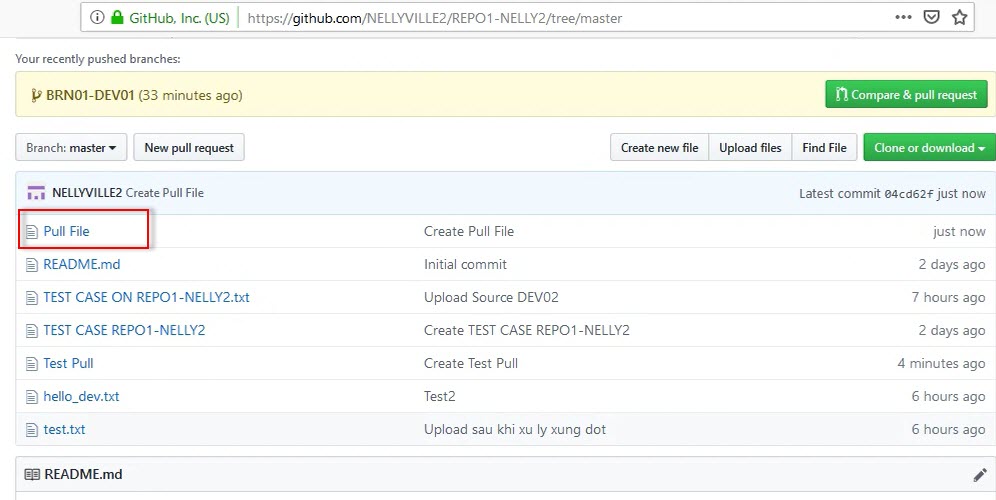
Tình huống muốn tải về những nội dung file mới nhất có trên Reposotory Remote về tại Working tree local

Đầu tiên tạo 1 file mới trên Reposooty Remote



Sau đó Commit

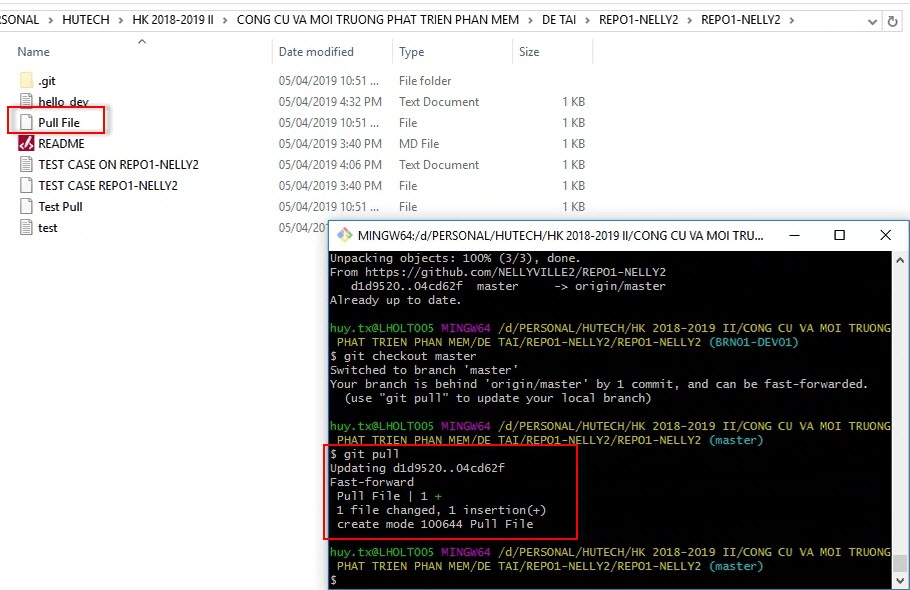




Bên dưới Repositpry local hiện chưa có tải file mới về.



Tiếp theo thực hiện câu lệnh **$git pull** để tải file mới nhất về.

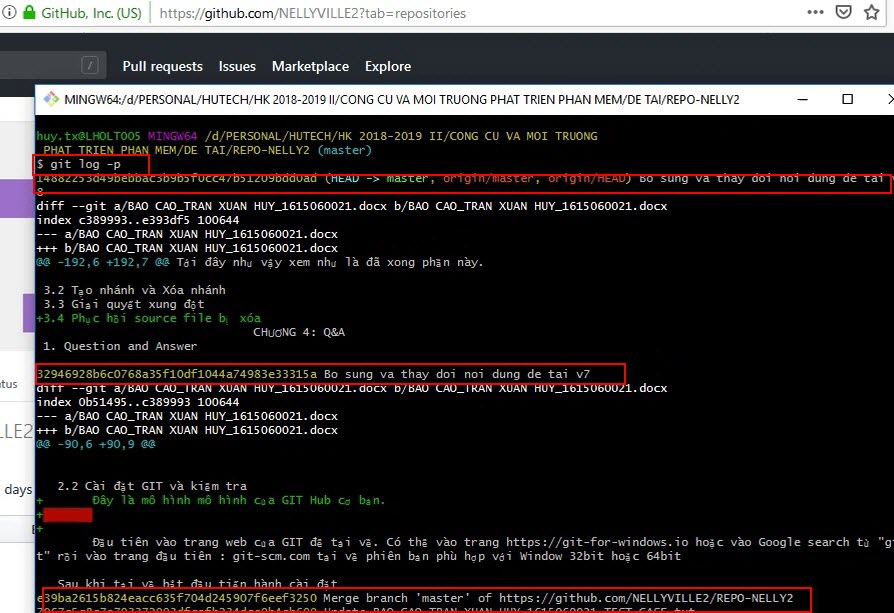


Vậy là trong Repository Local ở nhánh Master xuất hiện file “Pull file” đã được tải về trong Working tree local

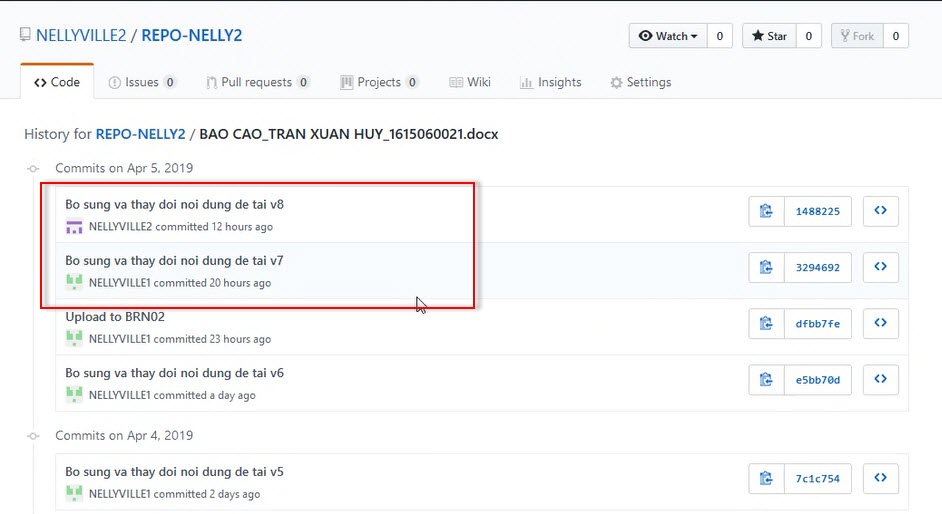
**3.6 Logs**

Tình huống muốn xem lại thông tin tất cả nội dung thông tin bên trong Repository khi Commit nhiều lần như thế nào thì ở đây thực hiện câu lệnh

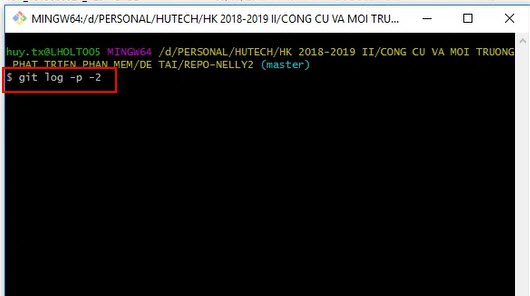
**$git log -p**



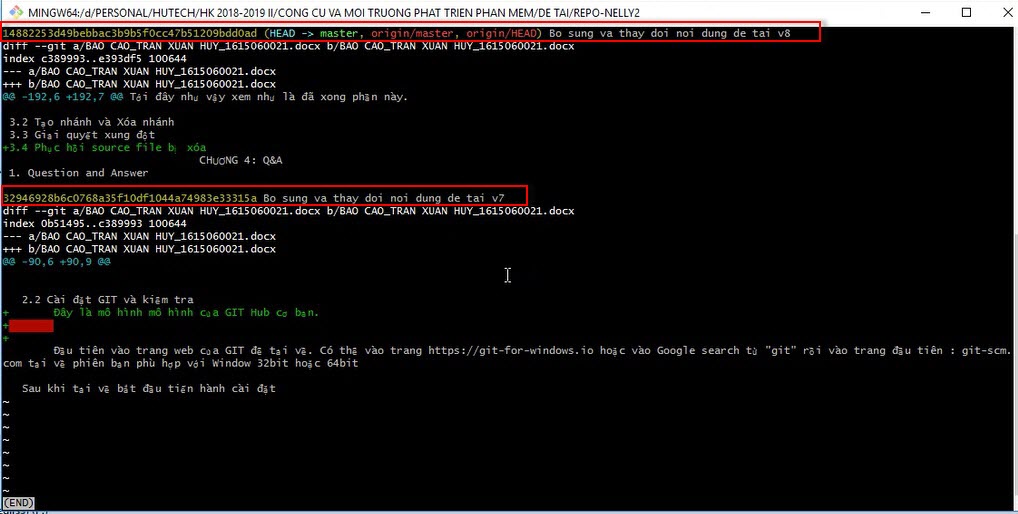
Sau khi thực hiện câu lệnh sẽ hiển thị ra kết quả tất cả các Commit. Tuy nhiên tình huống muốn xem lại các Commit nào gần nhất. Ở đây muốn xem lại 2 lần Commit cuối cùng như thế nào.



Thì sẽ thực hiện câu lệnh: **$git log –p -2**



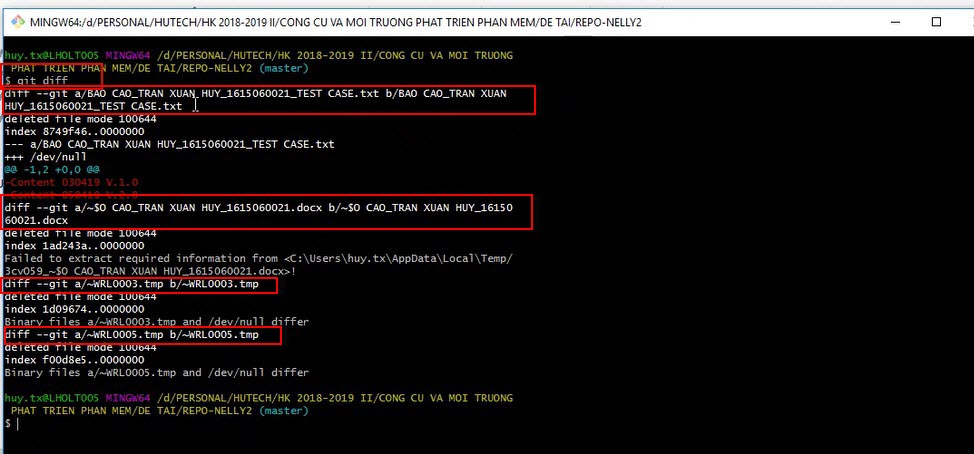
Kết quả trả về 2 Commit gần nhất



**3.7 Git Diff xem thông tin các Source file đã thay đổi nhưng chưa Commit**

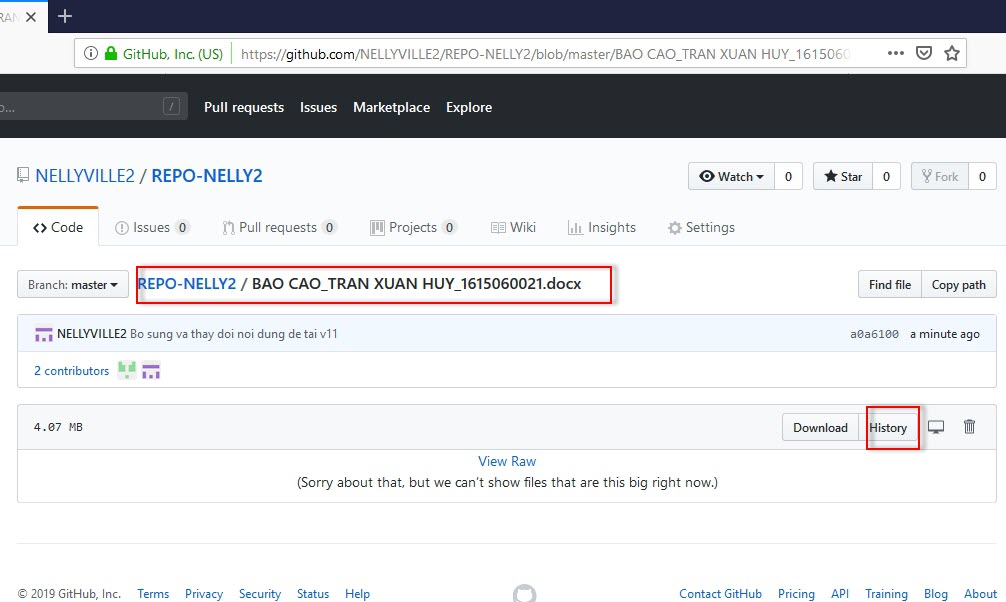
Tình huống muốn biết thông tin các Source file nào đang làm việc nhưng chưa được Commit thì thực hiện bằng câu lệnh

**$git diff**

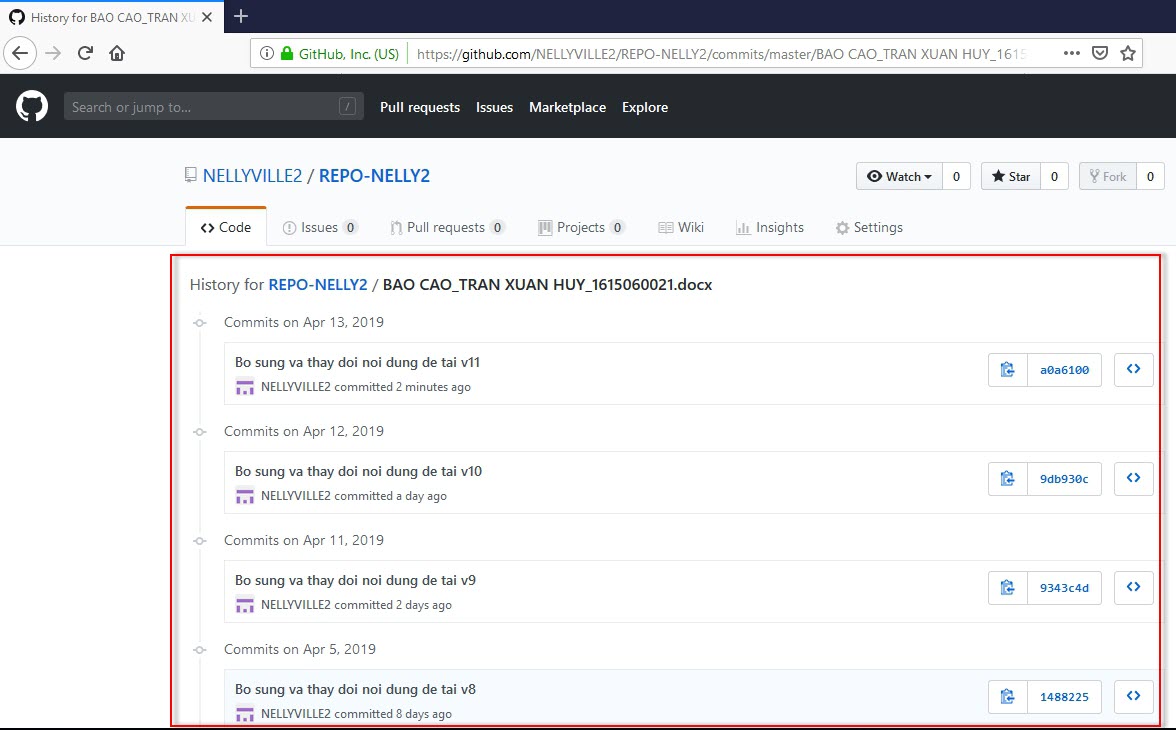


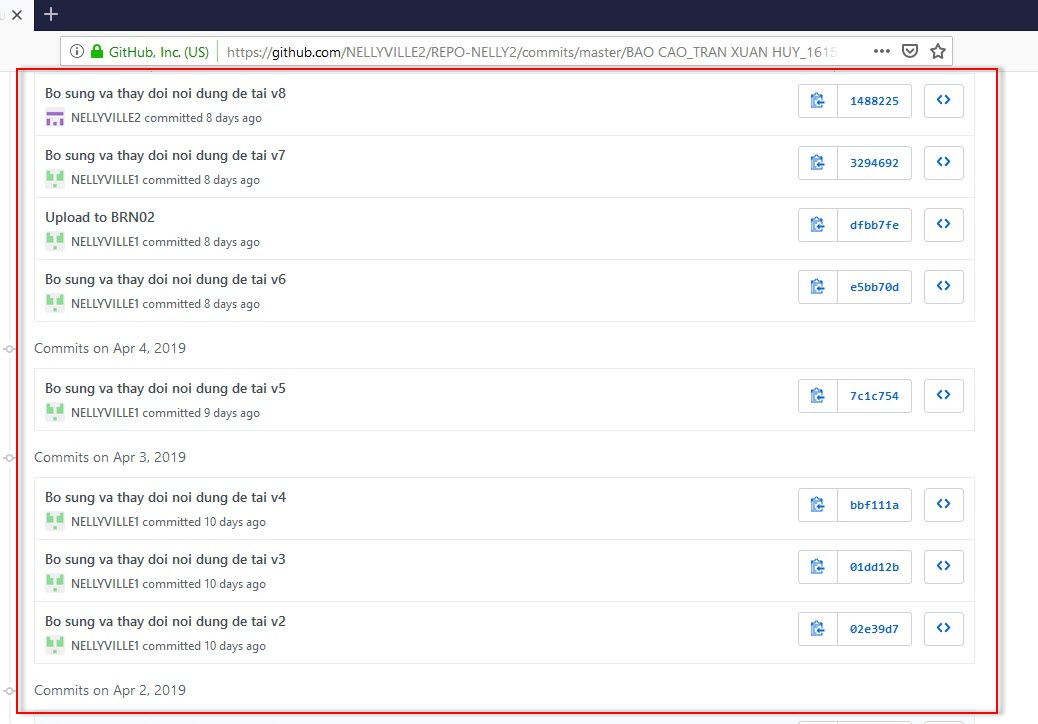
# **CHƯƠNG 4: BÁO CÁO TIẾN ĐỘ LÀM VIỆC**

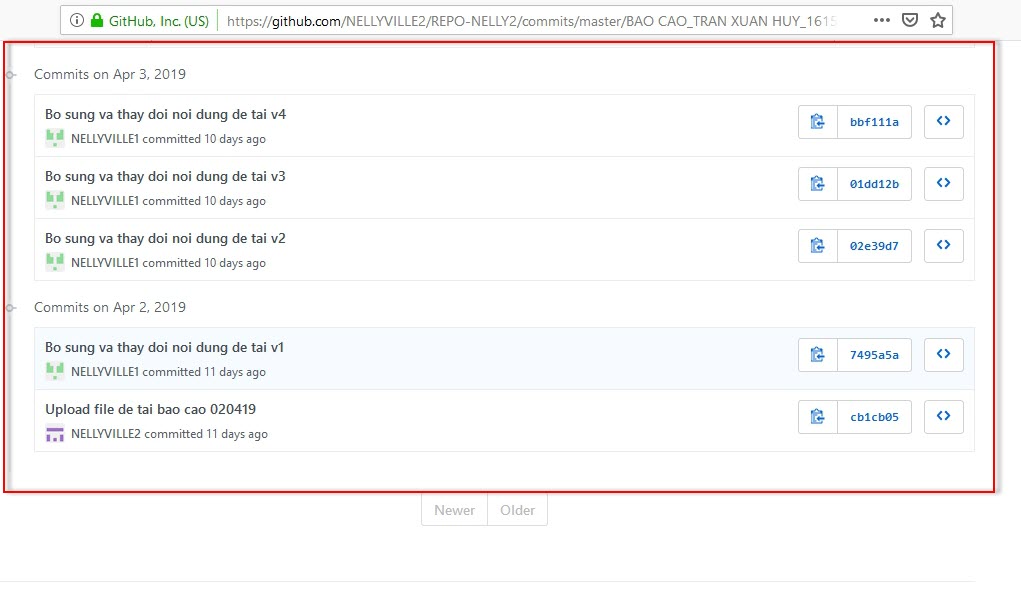
Đầu tiên chọn file cần xem lịch sử rồi nhấn History

****

Danh sách kết quả làm file báo cáo đề tài

****

****

****

# **CHƯƠNG 5: Q&A – KẾT LUẬN**

**Question:**

|  |  |
| --- | --- |
| Sự khác nhau giữa Git Clone, Git Fetch, Git Pull ?  **Answer:** | |
|  | Git Clone: Cơ chế gần như giống Git Pull, chỉ dùng cho lần đầu tiên tải về Local Working space |
|  | Git Pull: Cơ chế gần như giống Git Clone, chỉ khác tải những gì thay đổi mới nhất về Local Working space |
|  | Git Fetch: Khác Git Clone và Git Pull ở chỗ tải về nằm tại Repository Local |

**Kết luận:** Qua phần báo cáo ở trên có thể thấy Git hiện là một công cụ giải pháp cho việc quản lý source code phân tán để đáp ứng được nhu cầu kiểm soát và quản lý rất tốt hiện nay khi triển khai một dự án phần mềm nào đó. Bên dưới là thông tin về Liscense khi sử dụng Git Hub



