TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TIỂU LUẬN

MÔN HỌC: NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

Đề tài: Nghiên cứu, tìm hiểu công cụ lưu trữ mã nguồn online

Giáo viên hướng dẫn : *Ths.Nguyễn Thái Cường*

Nhóm thực hiện : Nhóm 8

Lớp : 201930503200001

Hà Nội, 2020

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TIỂU LUẬN

MÔN HỌC: NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

Đề tài: Nghiên cứu, tìm hiểu công cụ lưu trữ mã nguồn online

Giáo viên hướng dẫn : *Ths.Nguyễn Thái Cường*

Nhóm thực hiện : Nhóm 8

Lớp : 201930503200001

Sinh viên thực hiện : Đặng Thị Thu Thảo (NT)

Lê Sỹ Đức

Dương Thị Nhung

Nguyễn Phương Thảo

Nguyễn Viết Trí

Hà Nội, 2020

MỤC LỤC

LỜI CẢM ƠN

Để hoàn thành bài tập lớn này, chúng em xin cảm ơn chân thành đến toàn thể thầy cô trong trường Đại học công nghiệp Hà Nội nói chung và các thầy cô trong khoa Công nghệ thông tin nói riêng, những người đã tận tình hướng dẫn, chỉ bảo và trang bị cho chúng em những kiến thức bổ ích trong những năm học vừa qua. Chúng em xin gửi lời cảm ơn đến thầy Nguyễn Thái Cường đã tận tình hướng dẫn, trực tiếp chỉ bảo và tạo điều kiện giúp đỡ chúng em trong suốt quá trình làm bài tập lớn. Thầy không chỉ truyền đạt kiến thức của môn học mà còn truyền đạt những kỹ năng về thuyết trình, kỹ năng làm việc nhóm.

Chúng em xin chân thành cảm ơn !

Sinh viên thực hiện

Nhóm 8

PHẦN I. MỞ ĐẦU

1. Tên đề tài

Nghiên cứu,tìm hiểu công cụ lưu trữ mã nguồn online với công cụ Github

2. Lý do chọn đề tài

Trong quá trình phát triển phần mềm, chúng ta thường sẽ gặp phải nhiều vấn đề như:

- Làm thế nào để quản lý được các phiên bản của quá trình quản lý phần mềm?

- Làm thế nào để quản lý mã nguồn chung cho cả nhóm?

- Phần code này là của ai viết, người nào phải chịu trách nghiệm khi có lỗi xảy ra?

- Khánh hàng muốn sử dụng lại một phiên bản đã phát hình cũ thì làm như thế nào?

...

Để giải quyết được vấn đề đó, chúng ta có thể sử dụng các công cụ quản lý phiên bản và Github là một trong số đó, với công cụ này, toàn bộ phiên bản có thể được lưu trữ trên một thư mục (Repository) và tất cả các thành viên tham gia dự án có thể thực hiện các thao tác đưa thay đổi lên, cập nhật thay đổi về, cập nhật những thay đổi, lấy về những thay đổi do các thành viên khác cập nhật trước đó**,..**

Hơn thế nữa việc quản lý cấu hình tốt sẽ đem lại nhiều lợi ích:

- Giảm thiểu sự nhầm lẫn, tổ chức và quản lý tốt hơn các khoản mục phần mềm

- Tổ chức các hoạt động cần thiết để đảm bảo tính toàn vẹn của nhiều sản phẩm phần mềm

- Đảm bảo tính nguyên vẹn cấu hình hiện tại của sản phẩm

- Tối ưu hóa chi phí phát triển, bảo trì và hỗ trợ sau bán hàng

- Cung cấp môi trường phát triển, bảo trì, thử nghiệm và sản xuất ổn định

- Nâng cao chất lượng và tuân thủ các tiêu chuẩn kỹ thuật phần mềm

- Giảm chi phí làm lại những thành phần đã thực hiện trước đó

...

3. Mục đích đề tài

Tìm hiểu về cách sử dụng các chức năng của Github. Các lệnh của Github được thực hiện như thế nào. Cách đưa một Project mới lên Github, tải một Procject từ Github về máy cá nhân, commit thay đổi. Qua đó trang bị kỹ năng quản lý phiên bản trong một dự án, đặc biệt là khi các thành viên tham gia có những khoảng cách lớn về mặt địa lý.

4. Bố cục đề tài

Nội dung đề tài được trình bày trong 4 chương:

Chương 1. Tổng quan về mã nguồn online

Chương 2. Công cụ quản lý phiên bản Github: Giới thiệu về Git, Github; Ưu, nhược điểm của Github; So sánh Github với công cụ quản lý phiên bản khác.

Chương 3. Ứng dụng

5. Phương pháp

- Tìm kiếm thông tin
- Đọc hiểu tài liệu
- Cài đặt và sử dụng Github

PHẦN II. NỘI DUNG

CHƯƠNG 1. TỔNG QUAN VỀ QUẢN LÝ CẤU HÌNH PHẦN MỀM

1.1 Khái niệm quản lý cấu hình phần mềm

Quản lý cấu hình phần mềm (*configuration management)* là tập các hoạt động để quản lý các thay đổi của phần mềm trong suốt vòng đời của nó. Một loại hoạt động bảo đảm chất lượng phần mềm, được áp dụng cho tất cả các pha của kỹ nghệ. Bao trùm suốt tiến trình phát triển và tiến hóa của phần mềm.

1.2 Nội dung

Nội dung quản lý cấu hình phần mềm bao gồm:

- *Xác định các thay đổi.*
- *Kiểm soát các thay đổi.*
- *Bảo đảm các thay đổi đã được thực hiện.*
- *Báo cáo các thay đổi cho người quan tâm.*

Quản lý cấu hình khác bảo trì phần mềm:

- *Bảo trì phần mềm là các hoạt động kỹ nghệ xuất hiện sau khi phân phát phần mềm và nó đi vào hoạt động.*
- *Quản lý cấu hình phần mềm là các hoạt động theo dõi và kiểm soát , từ bắt đầu dự án phát triển phần mềm và chỉ kết thúc khi phần mềm không HĐ nữa.*

Kết quả của tiến trình kỹ nghệ phần mềm là các thông tin có thể được chia thành 3 loại:

- *Các chương trình máy tính (cả mức nguồn và mức chạy được).*
- *Các tài liệu mô tả chương trình máy tính đó (nhắm đến cả những người thực hành kỹ thuật lẫn những người dùng).*
- *Các cấu trúc dữ liệu (cả bên trong và ngoài chương trình)*

1.3 Cấu hình phần mềm

Các khoản mục cấu thành lên các thành phần của phần mềm được sản ra như là những chế tác của tiến trình kỹ nghệ phần mềm được tập hợp lại trong một cái tên chung gọi là cấu hình phần mềm.

Các chế tác này có nhiều mức khác nhau:

- *Bộ phận - tổng thể (phạm vi)*
- *Chưa hoàn thiện – hoàn thiện (theo tiến trình, chất lượng)*

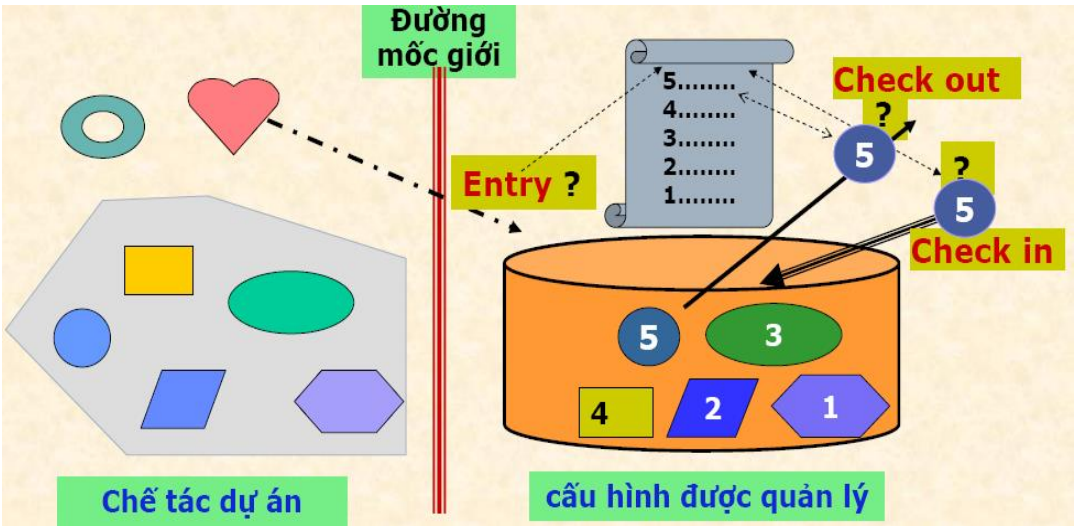
- *Ở các mức tiến hóa khác nhau (các phiên bản)*

1.4 Công cụ quản lý cấu hình

Các đường mốc giới là ranh giới được đặt ra:

- *Trước mốc giới, cấu hình có thể thay đổi nhanh chóng và không chính thức.*
- *Sau mốc giới, cần các thủ tục đặc biệt và chính thức để đánh giá và kiểm soát từng thay đổi cấu hình.*

Đường mốc giới để đánh dấu việc cập nhật hay phân phát một vài khoản mục cấu hình phần mềm. Tại đường mốc các khoản mục cấu hình phần mềm tương ứng được đưa vào cơ sở dữ liệu dự án



Hình 1.1 Đường mốc giới

1.5 Các khoản mục cấu hình phần mềm

Đặc tả hệ thống

Kế hoạch dự án phần mềm.

Đặc tả yêu cầu:

- *Đặc tả yêu cầu phần mềm.*
- *Nguyên mẫu thi hành được hoặc nguyên mẫu "giấy tờ"*

Sổ tay sử dụng sơ cấp

Các đặc tả thiết kế:

- *Dữ liệu*
- *kiến trúc*
- *Môđun (thủ tục)*
- *Giao diện*

- *Đối tượng (nếu dùng kỹ thuật hướng đối tượng)*

Mã nguồn và kiểm thử:

- *Kế hoạch và thủ tục kiểm thử .*
- *Các ca kiểm thử & các kết quả được ghi lại.*
- *Các sổ tay vận hành & sổ tay lắp đặt.*
- *Chương trình thi hành được.*
- *Các môđun & mã thi hành được*
- *Các môđun đã liên kết*

Mô tả cơ sở dữ liệu:

- *Lược đồ & cấu trúc các file*
- *Nội dung hồ sơ ban đầu*

Sổ tay người sử dụng

- *Các tài liệu bảo trì*
- *Các báo cáo những vấn đề phần mềm*
- *Các yêu cầu bảo trì*
- *Đặt thay đổi kỹ nghệ*
- *Các chuẩn & các thủ tục cho kỹ nghệ phần mềm.*

1.6 Sự hình thành quản lý cấu hình

Trách nhiệm nguyên thuỷ của quản lý cấu hình phần mềm – SCM là kiểm soát các thay đổi

Sau này thêm các trách nhiệm:

- *Xác định các khoản mục cấu hình, các version của phần mềm;*
- *Kiểm toán cấu hình phần mềm nhằm bảo đảm phần mềm đã được phát triển đúng và*
- *Báo cáo mọi thay đổi đã được áp dụng cho cấu hình đó.*

1.7 Nhiệm vụ quản lý cấu hình

5 nhiệm vụ cụ thể quản lý cấu hình phần mềm:

- *Xác định cấu hình*
- *Kiểm soát version*
- *Kiểm soát đổi thay*
- *Kiểm toán cấu hình*
- *Báo cáo thay đổi.*

1.8 Câu hỏi cho quản lý cấu hình

Mọi cuộc thảo luận về quản lý cấu hình phần mềm cần đưa ra các câu hỏi:

- *Làm thế nào để tổ chức minh định và quản lý được nhiều version của chương trình sao cho nó có thể thay đổi được để thích nghi một cách hiệu quả?*
- *Làm thế nào để tổ chức kiểm soát được các đổi thay phần mềm trước và sau khi phân phát cho người đặt hàng?*
- *Ai chịu trách nhiệm việc chấp thuận và đặt thứ tự ưu tiên của các đổi thay?*
- *Làm thế nào có thể bảo đảm rằng việc đổi thay đã được thực hiện đúng?*
- *Dùng cơ cấu nào để đánh giá các đổi thay khác?*

1.9 Xác định đối tượng cấu hình phần mềm

Cần đặt tên không trùng cho các khoản mục cấu hình phần mềm, để kiểm soát quản lý và tổ chức lại theo phương cách hướng đối tượng.

Có hai loại đối tượng:

- *Đối tượng cơ bản là một "đơn vị văn bản", được kỹ sư phần mềm tạo ra trong quá trình phân tích thiết kế, lập mã và kiểm thử .*
- *Đối tượng hỗn hợp được cấu thành từ các đối tượng*

Mỗi đối tượng có một bộ các đặc trưng thể hiện của nó là duy nhất: tên, mô tả, danh sách các nguồn lực, sự hiên thực hoá. Mô tả đối tượng bằng một danh sách các khoản mục dữ liệu:

- *Kiểu khoản mục cấu hình phần mềm (tài liệu hay chương trình hay dữ liệu).*
- *Chứng thư dự án (thuộc phần nào trong dự án).*
- *Thông tin đổi thay và/hoặc thông tin version*

Nguồn lực là tất cả các thực thể được cung cấp, xử lý, tham khảo, và các thứ khác được đối tượng cần đến. Mối quan hệ giữa các đối tượng là quan hệ bộ phận – toàn bộ. Ta có đồ thị các đối tượng. Một quan hệ khác là quan hệ liên quan với nhau (<interrelated>). Để kiểm soát đổi thay của đối tượng ta cần đến đồ thị tiến hoá cho từng đối tượng, nó mô tả lịch sử đổi thay của đối tượng đó.

1.10 Kiểm soát phiên bản

Kiểm soát phiên bản bằng tổ hợp các thủ tục & các công cụ để quản lý các phiên bản khác nhau của các đối tượng cấu hình (đã được tạo ra trong tiến trình kỹ nghệ phần mềm). Quản lý cấu hình cho phép người sử dụng đặc tả các cấu hình thay

thế của hệ thống phần mềm bằng lựa chọn các phiên bản thích hợp và gắn kết với các thuộc tính; nhờ đó mà cho phép đặc tả một cấu hình bằng mô tả tập các thuộc tính mong muốn.

Để xây dựng một biến thể thích hợp của một phiên bản của một chương trình, mỗi thành phần của phiên bản được gán một "bộ thuộc tính" - là một danh sách các đặc trưng. Một phiên bản hay biến thể được xây dựng cần xác định thành phần nào được dùng hay cần thay đổi. Một cách khác để hình thành khái niệm về quan hệ giữa các thành phần, các biến thể, các phiên bản là biểu diễn chúng như là một vụng (pool) đối tượng. Mỗi thành phần được cấu tạo bởi một bộ các đối tượng trong cùng một mức xét duyệt. Mỗi biến thể là một bộ các đối tượng trong cùng một mức xét duyệt. Xác định khi các thay đổi mỗi phiên bản chủ yếu đã được thực hiện đối với một vài đối tượng.

CHƯƠNG 2. TỔNG QUAN VỀ QUẢN LÝ PHIÊN BẢN PHẦN MỀM

2.1 Quản lý phiên bản phần mềm

Quản lý mã nguồn chủ yếu liên quan đến việc theo dõi các sửa đổi đối với mã. Các công cụ để quản lý mã nguồn đôi khi được gọi là "Hệ thống quản lý mã nguồn" (SCMS - Source Control Management System), "Hệ thống kiểm soát phiên bản" (VCS – Version Control System), "Hệ thống kiểm soát sửa đổi" (RCS - Revision Control System) hoặc đơn giản là "kho mã" tùy thuộc vào các tính năng mà chúng cung cấp hoặc cách chúng được sử dụng. Một "kho lưu trữ" thường đề cập đến một dự án. Các hệ thống quản lý phiên bản phải bao gồm các tính năng như xác thực truy cập, theo dõi lịch sử sửa đổi, các phiên bản.

Quản lý phiên bản còn được gọi là “Kiểm soát phiên bản” hoặc “Kiểm soát sửa đổi”, là phương tiện để theo dõi và kiểm soát hiệu quả các thay đổi đối với một tập hợp các thực thể có liên quan.

Kiểm soát phiên bản thường được sử dụng nhất để theo dõi và kiểm soát các thay đổi đối với mã nguồn. Nó là một công cụ rất quan trọng trong một chiến lược quản lý vòng đời tổng thể của một phần mềm. Qua nhiều năm, nhiều hệ thống kiểm soát phiên bản khác nhau đã được phát triển, đặc biệt là trong các lĩnh vực quản lý phiên bản và quản lý tài liệu. Các hệ thống này có thể là thương mại hoặc nguồn mở và thường chạy như các ứng dụng độc lập.

Hệ thống quản lý phiên bản có thể hoạt động như kho lưu trữ trung tâm (ví dụ như Git). Đối với những hệ thống này, người dùng cá nhân nhận được một bản sao hoàn chỉnh của một kho lưu trữ (ví dụ, sao chép nó) và lưu trữ nó cục bộ. Những thay đổi đó sau đó được đẩy / kéo vào kho gốc để sẵn sàng cho các thành viên khác trong nhóm. Môi trường cộng tác cũng đã được phát triển (ví dụ GForge, GitHub, GitLab, v.v.) xung quanh các hệ thống điều khiển phiên bản để tạo thuận lợi cho việc tiếp cận theo nhóm dựa trên việc quản lý vòng đời của các ứng dụng.

Trong phát triển phần mềm mã nguồn mở, những người tham gia hầu hết đến từ những vùng địa lý khác nhau, do đó cần có các công cụ để hỗ trợ người tham gia trong việc phát triển mã nguồn. Vào đầu những năm 2000, Hệ thống phiên bản đồng thời (CVS) là một ví dụ nổi bật về một công cụ quản lý phiên bản đang được sử dụng trong các dự án phần mềm mã nguồn mở. CVS giúp quản lý các tập tin và mã số của một dự án khi một số người đang làm việc trên dự án cùng một lúc. CVS cho phép nhiều người cùng làm việc trên cùng một tệp. Điều này được thực hiện bằng cách di chuyển tệp vào thư mục của người dùng và sau đó hợp nhất các tệp khi người dùng hoàn tất. CVS cũng cho phép người dùng dễ dàng truy xuất phiên bản trước của

tệp. Vào giữa những năm 2000, hệ thống điều khiển sửa đổi Subversion (SVN) được tạo ra để thay thế CVS. Nó nhanh chóng đạt được nền tảng như một hệ thống quản lý phiên bản PMNM

2.2 Một số công cụ quản lý phiên bản

Có nhiều công cụ quản lý phiên bản như:

- Github: Là nơi lưu trữ source code nổi tiếng thế giới thế giới, Github có chức năng như một nguồn resource phát triển độc lập. Các developer có thể xây dựng project, chia sẻ với cộng đồng và cộng đồng có thể sửa đổi.
- GitLab nó cũng tương tự như GitHub nhưng GitLab theo hướng kinh doanh- Launchpad :Là nhà của untity và nhiểu tiện ích linux phổ biến.Phần lớn các dự án lưu trữ tại launchpad phát triển và xây dựng cho cộng đồng Linux
- Hệ thống phiên bản đồng thời (CVS). CVS đã xuất hiện từ những năm 80 và đã rất phổ biến với cả các nhà phát triển thương mại và nguồn mở.
- Mercurial bắt đầu gần cùng thời với Git và cũng là một công cụ kiểm soát sửa đổi phân tán.

Ngoài ra còn có các công cụ khác như SVN, VisualSourceSafe.

CHƯƠNG 3. CÔNG CỤ QUẢN LÝ PHIÊN BẢN GITHUB

3.1 Git là gì?

Git là tên gọi của một **Hệ thống quản lý phiên bản phân tán**(*Distributed Version Control System* – ***DVCS***) là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (**clone**) từ một kho chứa mã nguồn (**repository**), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (**commit**) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia. Trong Git, thư mục làm việc trên máy tính gọi là **Working Tree**.

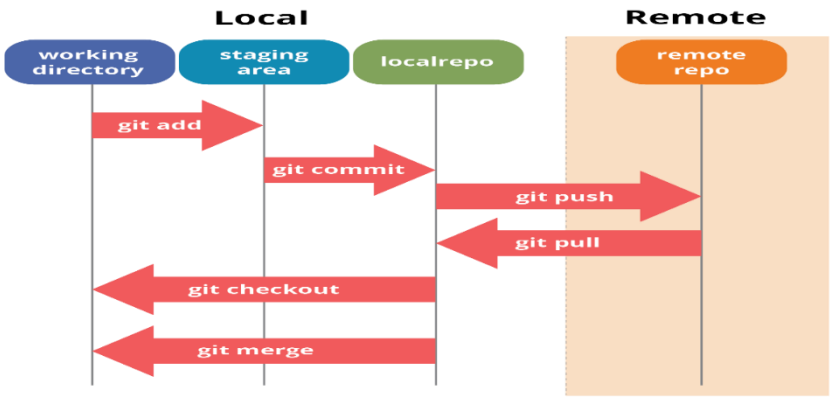
Ngoài ra, có một cách hiểu khác về Git đơn giản hơn đó là nó sẽ giúp bạn lưu lại các phiên bản của những lần thay đổi vào mã nguồn và có thể dễ dàng khôi phục lại dễ dàng mà không cần copy lại mã nguồn rồi cất vào đâu đó. Và một người khác có thể xem các thay đổi của bạn ở từng phiên bản, họ cũng có thể đối chiếu các thay đổi của bạn rồi gộp phiên bản của bạn vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.

Cơ chế lưu trữ phiên bản của Git là nó sẽ tạo ra một "*ảnh chụp*" (*snapshot*) trên mỗi tập tin và thư mục sau khi commit, từ đó nó có thể cho phép bạn tái sử dụng lại một ảnh chụp nào đó mà bạn có thể hiểu đó là một phiên bản. Đây cũng chính là lợi thế của Git so với các DVCS khác khi nó không "lưu cứng" dữ liệu mà sẽ lưu với dạng snapshot.

3.2 Cơ chế check in – check out trong Git

Mỗi tập tin trong Git được quản lý dựa trên ba trạng thái: committed, Modified, và staged. Committed có nghĩa là dữ liệu được lưu trữ một cách an toàn trong cơ sở dữ liệu.Modified có nghĩa là bạn đã thay đổi tập tin nhưng chưa commit vào cơ sở dữ liệu.và Staged là bạn đánh dấu sẽ commit phiên bản hiện tại của một tập tin đã chỉnh sửa trong lần commit sắp tới. Điều này tạo ra ba phần riêng biệt của một dự án sử

dụng Git: thư mục Git, thư mục làm việc ,và khu vực tổ chức(staging area)



Hình 3.1 Cơ chế check in - check out trong Git

3.3 GitHub là gì?

GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn,Git dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án sẽ được cung cấp kho lưu trữ miễn phí. Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.

Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên. Sự phát triển của nền tảng GitHub bắt đầu vào ngày 19 tháng 10 năm 2007. Trang web được đưa ra vào tháng 4 năm 2008 do Tom Preston-Werner, Chris Wanstrath, và PJ Hyett thực hiện sau khi nó đã được hoàn thành một vài tháng trước đó, xem như giai đoạn beta.

Dự án trên Github có thể được truy cập và thao tác sử dụng một giao diện dòng lệnh và làm việc với tất cả các lệnh Git tiêu chuẩn. Github cũng cho phép người dùng đăng ký và không đăng ký để duyệt kho công cộng trên trang web. Github cũng tạo ra nhiều client và plugin cho máy tính để bàn. Trang web cung cấp các chức năng mạng xã hội như feed, theo dõi, wiki (sử dụng phần mềm Gollum Wiki) và đồ thị mạng xã hội để hiển thị cách các nhà phát triển làm việc trên kho lưu trữ. Một người sử dụng phải tạo ra một tài khoản cá nhân để đóng góp nội dung lên Github, nhưng các kho mã nguồn công cộng có thể được duyệt và tải về với bất cứ ai. Với một người dùng đã đăng ký tài khoản, họ có thể thảo luận, quản lý, tạo ra các kho, đóng góp cho kho của người dùng khác, và xem xét thay đổi mã.

GitHub cũng có một dịch vụ khác: một trang web kiểu pastebin gọi là Gist, dùng để lưu trữ các đoạn mã; trong khi Github sẽ được cho lưu trữ các dự án lớn hơn. Một dịch vụ lưu trữ khác được gọi là Speaker Deck. Các phần mềm chạy GitHub được viết bằng Ruby on Rails và Erlang.

3.4 Phạm vi

GitHub chủ yếu được sử dụng để lưu trữ mã nguồn phần mềm, nhưng cũng thường được sử dụng với nhiều loại tập tin như Final Cut hoặc các tài liệu Word. Ngoài mã nguồn, Github hỗ trợ các định dạng và các tính năng sau đây: 3D làm cho các tập tin mà có thể được xem trước bằng cách sử dụng tích hợp trình xem file STL mới hiển thị các tập tin trên một khung 3D. Người xem được hỗ trợ bởi WebGL và Three.js; Nguồn gốc định dạng PSD của Photoshop có thể được xem trước và so với các phiên bản trước của cùng một tập tin; Lồng nhiệm vụ danh sách; Tài liệu và Wiki; Các trang web nhỏ có thể được lưu trữ từ kho công cộng trên Github. Định dạng URL là **http://projectname.github.io**. Và có thể được tạo ra bằng cách bắt đầu một kho lưu trữ được định dạng như projectname.io; Code Snippets (bằng cách sử dụng tên miền phụ Gist); Theo dõi vấn đề và tính năng yêu cầu; Trực quan của dữ liệu không gian địa lý

Github dành cho doanh nghiệp: Github cho doanh nghiệp cũng hoạt động giống Github.com, nhưng hỗ trợ phiên bản trả phí cho các doanh nghiệp muốn bảo vệ mã nguồn của mình, không công khai ra cộng đồng.

Việc làm: Một trong những nguồn thu nhập khác của Github là *GitHub Jobs* nơi sử dụng lao động có thể gửi lời mời làm việc với $450/listing. Nhân viên bán hàng của GitHub không được trả lương trên cơ sở hoa hồng.

Phổ biến: *24 Tháng Hai năm 2009,* trong một cuộc nói chuyện tại Yahoo! trụ sở thành viên trong nhóm GitHub công bố trong một cuộc nói chuyện tại trụ sở trên Yahoo! rằng trong năm đầu tiên GitHub là trực tuyến, nó tích lũy 46.000 kho công cộng, 17.000 trong số họ trong tháng trước đó một mình. Vào thời điểm đó, khoảng 6.200 kho đã được chia hai ít nhất một lần và 4.600 sáp nhập, *05 tháng 7 năm 2009,* một Blog Github bài thông báo họ đạt đến 100.000 người sử dụng nhãn hiệu, 27 tháng 7 năm 2009, Tom Preston-Werner thông báo rằng những con số này đã tăng lên 90.000 kho công cộng duy nhất, 12.000 đã được chia hai ít nhất một lần, với tổng số 135.000 kho. Vào tháng 7 năm 2010, GitHub thông báo rằng nó chứa 1 triệu kho. Vào tháng 4 năm 2011, GitHub thông báo rằng nó được lưu trữ 2 triệu kho. *16 Tháng 1 năm 2013,* GitHub thông báo đã thông qua 3 triệu người sử dụng đánh dấu và sau đó được lưu trữ hơn 5 triệu kho. *Tháng 7 năm 2012,* Peter Levine, đối tác ở

nhà đầu tư GitHub của Andreessen Horowitz, nói rằng GitHub đã được phát triển doanh thu 300% mỗi năm kể từ năm 2008 "có lợi nhuận suốt từ đó đến giờ".

3.5 Độ tin cậy và quản lý bảo mật trên Github

GitHub có hơn 800 dự án chuyên về bảo mật cung cấp cho các nhà quản trị CNTT và các chuyên gia an toàn thông tin đủ loại công cụ để phân tích phần mềm độc hại , kiểm tra tấn công xâm nhập, tầm soát máy tính và mạng, ứng phó sự cố, giám sát mạng, và nhiều việc khác.

Kiểm tra tấn công xâm nhập: Khi nói đến kiểm tra tấn công xâm nhập, không có lựa chọn nào tốt hơn Metasploit Framework của Rapid7. Thư viện các kiểu tấn công phong phú của nó có thể sử dụng để đánh giá mức độ an toàn của ứng dụng hoặc hệ thống mạng trước khi bị tin tặc kẻ tấn công. Metasploit có cấu trúc mô-đun linh hoạt cho từng loại thiết bị, dùng để kiểm tra máy tính, điện thoại di động, thiết bị định tuyến (router), chuyển mạch (switch), hệ thống điều khiển công nghiệp và các thiết bị nhúng. Metasploit có thể chạy trên nhiều nền tảng, bao gồm Windows, Linux, Mac, Android và iOS.

Phòng thủ toàn diện: CFSSL của CloudFlare là "con dao Thụy Sĩ" đa năng cho phép tạo chữ ký số, xác minh và đóng gói chứng chỉ TLS. Vừa là công cụ dòng lệnh vừa là máy chủ HTTP API, CFSSL cho phép nhà quản trị CNTT tạo công cụ TLS/PKI tùy chỉnh và cấp chứng chỉ số (CA) có thể sử dụng nhiều khóa chữ ký. CFSSL còn có tính năng quét TLS kiểm tra cấu hình máy chủ dò tìm lỗ hổng và chuyển gói tin để thiết lập cấu hình hay thu hồi chứng chỉ. Việc vô tình lộ dữ liệu nhạy cảm như các khóa và mật khẩu là vấn đề phổ biến trong phát triển phần mềm. Gitrob giúp các chuyên gia bảo mật quét kho mã nguồn của mình trên GitHub tìm các tập tin nhạy cảm. Tuy GitHub có sẵn chức năng dò tìm những thông tin này, nhưng Gitrob giúp cho công việc đơn giản hơn bằng cách lập danh sách tất cả kho chung và riêng trên GitHub, và dựa trên đó dò tìm các tên tập tin có thể chứa thông tin nhạy cảm. Gitrob lưu kết quả tìm kiếm vào một cơ sở dữ liệu PostgreSQL và hiển thị với một ứng dụng web đơn giản.

Giám sát mạng: Bro Security Network Monitor cho phép các chuyên gia bảo mật giám sát tất cả máy tính trên mạng (có thể can thiệp vào luồng dữ liệu mạng và kiểm tra các gói tin truyền trên mạng) và cho phép các nhà phân tích kiểm tra lớp ứng dụng. Ngôn ngữ kịch bản của Bro có thể dùng để tạo các chính sách giám sát cho website. Theo thông tin trên trang web của dự án (https://github.com/bro/bro), Bro được sử dụng nhiều trong môi trường khoa học như các trường đại học, viện nghiên cứu, và các trung tâm điện toán. OSSEC là hệ thống phát hiện xâm nhập dựa trên máy

chủ có các tính năng theo dõi nhật ký hệ thống (log) và quản lý sự kiện và thông tin bảo mật (SIEM - security information and event management), có thể chạy trên nhiều nền tảng gồm Linux, Mac OS, Solaris, AIX, và Windows. Nó thường được dùng để phân tích log, kiểm tra sự toàn vẹn tập tin, giám sát chính sách, phát hiện rootkit, cảnh báo thời gian thực, … Bằng cách cấu hình OSSEC gửi cảnh báo khi có những thay đổi hệ thống tập tin trái phép hay hành vi độc hại chèn vào các nhật ký phần mềm, các tổ chức và doanh nghiệp có thể đảm bảo việc tuân thủ các chính sách bảo mật .

Ứng phó sự cố và điều tra: Mozilla Defense Platform (MozDef) cung cấp cho các chuyên gia bảo mật nền tảng để giám sát, ứng phó và hợp tác đối phó với những sự cố bảo mật trong thời gian thực, cho phép tự động hóa việc xử lý sự cố. MozDef sử dụng Elasticsearch, Meteor và MongoDB, mở rộng các tính năng SIEM truyền thống với các biểu đồ, hình ảnh trực quan. Đây là nền tảng hiện được sử dụng tại Mozilla. OS X Auditor phân tích các thành phần mở rộng của hệ thống, các thành phần của bên thứ ba, các tập tin tải về và các ứng dụng cài đặt trên hệ thống đang chạy (hoặc bản sao). Công cụ điều tra này trích xuất thông tin người dùng chẳng hạn như lịch sử và cookie trình duyệt, các tập tin tải về, dữ liệu đăng nhập, tài khoản mạng xã hội và email, kết nối Wi-Fi... và xác minh "uy tín" của từng thứ dựa trên nhiều nguồn.

Nghiên cứu và dò tìm lỗ hổng: Công cụ phân tích malware tự động Cuckoo Sandbox có nguồn gốc từ một dự án năm 2010 trong chương trình Google Summer of Code (hỗ trợ phát triển các dự án mã nguồn mở). Cuckoo cho phép mổ xẻ các tập tin nghi ngờ và giám sát các hành vi có thể gây hậu quả trong một môi trường ảo cô lập, nó kết xuất bộ nhớ và phân tích dữ liệu (chẳng hạn như lần vết các lệnh gọi API) để xác định hành vi của một tập tin đáng ngờ thực hiện trên hệ thống. Jupyter (Jupyter Notebook) là ứng dụng web cho phép tạo và chia sẻ tài liệu (sổ tay) có chứa mã nguồn, ký hiệu, hình ảnh trực quan và văn bản chú giải. Dự án này không chuyên về bảo mật nhưng bất kỳ chuyên gia gia bảo mật nào cũng cần phải có. Có nhiều công cụ bổ sung cho nó, trong đó có Jupyterhub, một máy chủ nhiều người dùng.

3.6 Tính năng API của Github

Ngoài những tính năng tuyệt vời của hệ thống quản lý source phân tán GIT nói chung, Github còn hỗ trợ người dùng những tính năng quan trọng thông qua API sau: API to Update The Repository via HTTP: GitHub hỗ trợ người dùng có thể edit file source code từ web browser thông qua HTTP – POST; API to Access Compare Views: Tính năng này hỗ trợ người dùng review và so sánh code của dự án thông qua việc xem các commit, comments, các dòng khác nhau giữa 2 version của file code ...

Tính năng này cũng thông qua HTTP - POST, người dùng có thể thực hiên trên web browser; API to Manage Service Hooks: GitHub hỗ trợ tính năng mở rộng post-receive hooks. Tính năng này cho phép người dùng đăng ký 1 URL của mình (như là một web hook) cho các respository. Bất cứ khi nào có người push source code của họ lên repository, GitHub sẽ thông báo cho bạn biết bằng cách POST thông tin (dạng JSON) về lần push đó đến URL mà bạn đã đăng ký trước đó.

3.7 Ưu, nhược điểm của Github

3.7.1 Ưu điểm

Git dễ cài đặt và sử dụng, an toàn và nhanh chóng. Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc kết hợp các phân nhánh (branch).

Giúp cải thiện kỹ năng lập trình bằng cách theo dõi và sửa đổi thường xuyên: Bạn có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa. Dễ dàng trong việc triển khai sản phẩm. Chứng minh bạn là 1 lập trình viên thực thụ. Giúp học hỏi các kỹ năng mới.

3.7.2 Nhược điểm

Tài khoản github là miễn phí,nhưng kho chứa riêng tư lại bị giới hạn nếu muốn dùng thêm phải trả phí. Phát hiện nhiều điểm yếu trên thuật toán SHA1 của Github việc này có thể dẫn đến bị mã hóa dữ liệu. Sử dụng github trên window hơi cồng kềnh. Hệ thống quản lý phiên bản buộc bạn đánh dấu rõ ràng vào tập tin. Trong khi điều này đặc biệt phiền toái vì nó lại dính líu đến việc phải liên lạc với máy chủ trung tâm

3.8 So sánh SVN với GitHub và ViSual SourceSafe, CVS

3.8.1 SVN

SVN(viết tắt của Subversion là một hệ thống quản lý version control system-VCS). Nó là 1 hệ thống quản lý phiên bản tập trung. SVN Subversion là hệ thống quản lý phiên bản mạnh mẽ, hữu dụng, và linh hoạt. Tích hợp vào Windows explorer, mỗi khi cập nhật phải vào đúng thư mục rồi cập nhật, rất bất tiện (nếu muốn tích hợp vào trong Visual studio thì bạn phải cài đặt thêm Visual SVN, phải tốn tiền mua). Không phép bạn cài đặt nhiều loại "tiến trình công việc" (workflow). SVN có khả năng xử lý các cấu trúc lồng nhau của các dự án và các gói Java

Rất dễ bị xung đột (conflic) nếu sửa file mà không update trước, nếu đã bị conflic thì bạn phải mở chức so sánh 2 phiên bản rồi tiến hành sửa. SVN Subversion quản lý tập tin và thư mục theo thời gian. SVN Subversion giống như một hệ thống file server

mà các client có thể download và upload file một cách bình thường. Điểm đặt biệt của SVN Subversion là nó lưu lại tất cả những gì thay đổi trên hệ thống file: file nào đã bị thay đổi lúc nào, thay đổi như thế nào, và ai đã thay đổi nó. SVN Subversion cũng cho phép recover lại những version cũ một cách chính xác. Các chức năng này giúp cho việc làm việc nhóm trở nên hiệu quả và an toàn hơn rất nhiều.

3.8.2 ViSual SourceSafe

ViSual SourceSafe (viết tắt là VSS) là 1 phần mềm để quản lý mã nguồn bằng cách tạo thư viện ảo cho các tập tin trên máy tính. Được tích hợp sẵn vào Visual studio nên dùng rất thuận tiện. Sử dụng 2 cơ chế là check in 1 người và nhiều người (nếu 1 người check in thì người khác ko thể checkin, do đó sẽ không bị vấn đề conflic)

VSS không có khả năng sử lý các cấu trúc lông nhau của các dự án và các gói Java. Không được phát triển từ lâu. Đối với nhóm phát triển nhỏ Microsoft, VSS là rất dễ dàng để thực hiện giải pháp, trong đó giải pháp thay thế là không thể kiểm soát VERSION.

VSS không có chi phí, không cần máy chủ (trừ một file chia sẻ). Nó thường không làm việc, và một số cửa hàng sử dụng nó. Dễ duy trì phiên bản mới nhất của một tập tin, nhưng lịch sử thường bị hỏng.

3.8.3 DVCS của GitHub

DVCS(viết tắt của Distributed Version Control Systems) là 1 hệ thống quản lý phiên bản phân tán. SVN là 1 kho chứa trung tâm còn DVCS là đa kho

Các máy khách không chỉ "check out" phiên bản mới nhất của các tập tin: chúng sao chép (mirror) toàn bộ kho chứa (repository)vì vậy nếu như một máy chủ nào mà các hệ thống quản lý phiên bản này (mỗi máy khách là một hệ thống riêng biệt) đang cộng tác ngừng hoạt động, thì kho chứa từ bất kỳ máy khách nào cũng có thể dùng để sao chép ngược trở lại máy chủ để khôi phục lại toàn bộ hệ thống

Hệ thống xử lý rất tốt việc quản lý nhiều kho chứa từ xa. Cho phép bạn cài đặt nhiều loại "tiến trình công việc" (workflow).

3.8.4 Hệ thống phiên bản đồng thời (CVS)

CVS đã xuất hiện từ những năm 80 và đã rất phổ biến với cả các nhà phát triển thương mại và nguồn mở. Nó được phát hành theo giấy phép GNU và sử dụng một hệ thống để cho phép người dùng kiểm tra các mã mà họ sẽ làm việc trên và kiểm tra các thay đổi của họ.

Ban đầu, CVS đã xử lý xung đột giữa hai lập trình viên bằng cách chỉ cho phép phiên bản mã mới nhất được xử lý và cập nhật. Như vậy, đây là lần đầu tiên, hệ thống

phục vụ đầu tiên mà người dùng phải xuất bản các thay đổi nhanh chóng để đảm bảo rằng những người dùng khác đã đánh bại họ.

Bây giờ, CVS có thể xử lý các dự án phân nhánh để phần mềm được phát triển có thể phân chia thành các sản phẩm khác nhau với các tính năng độc đáo và sẽ được đối chiếu sau. Máy chủ CVS chạy trên các hệ thống giống Unix với phần mềm máy khách chạy trên nhiều hệ điều hành. Nó được coi là hệ thống kiểm soát phiên bản trưởng thành nhất vì nó đã được phát triển trong một thời gian dài như vậy và không nhận được nhiều yêu cầu cho các tính năng mới tại thời điểm này. Một dự án rẽ nhánh của CVS, CVSNT đã được tạo để chạy CVS trên các máy chủ Windows và hiện đang được tích cực phát triển để tăng chức năng.

Ưu điểm:

- *Đã được sử dụng trong nhiều năm và được coi là công nghệ trưởng thành*

Nhược điểm:

- *Di chuyển hoặc đổi tên tập tin không bao gồm cập nhật phiên bản*
- *Rủi ro bảo mật từ các liên kết tượng trưng đến các tập tin*
- *Không được thiết kế để phân nhánh*

3.9 Ưu điểm của cơ chế làm việc trên nhiều nhánh

Hai nhánh (branch) đi suốt chiều dài phát triển code là Master và Develop. Trong đó:

Nhánh chính là nhánh Master, nhánh này luôn đảm bảo rằng code được lưu trữ trên đó là phiên bản chính thức mới nhất đang được phát hành.

Nhánh Develop là nhánh được cập nhật liên tục các đóng góp của tất cả mọi người ở mọi thời điểm. Nhánh này sẽ tiếp nhận các đóng góp của mọi người gửi đến thông qua việc tiếp nhận Pull Request cũng như tiếp nhận việc nhập các nhánh khác vào (integration branch). Khi nhánh Develop đạt độ chín mùi, nó sẽ được nhập vào nhánh Master đồng thời được dán nhãn phiên bản và quá trình phát hành phiên bản được tiến hành.

Ngoài 2 nhánh trên, kho code còn có các nhánh hỗ trợ phát triển, và nó chỉ tồn tại trong một giai đoạn nào đó của dự án nhằm phục vụ những mục đích nhất định, gồm có các nhánh sau: Feature branches; Các nhánh tính năng Release branches ; Các nhánh hỗ trợ phát hành phiên bản ; Hotfix branches ; Các nhánh hỗ trợ vá lỗi nhanh. Để đánh dấu các mốc quan trọng (ví dụ các phiên bản được phát hành), ta sử dụng các tag. Các tag này sẽ giúp truy cập nhanh đến kho code tại thời điểm diễn ra dấu

mốc sự kiện được đánh dấu; Nhánh cho phép ta thoải mái thử nghiệm các ý tưởng mới với dự án. Với nhánh, ta có thể thay đổi, vọc phá mà không sợ gây ảnh hưởng đến dự án. Sau đó, nếu ưng ý với kết quả, ta có thể hợp nhất (merge) nó với dự án. Còn nếu không, ta có thể bỏ cả nhánh và dự án không hề bị ảnh hưởng. Nếu bây giờ chạy lệnh git status, ta sẽ thấy ở dòng trên cùng cho biết ta đang ở nhánh master . Đây là nhánh mặc định của Git.

Để xem danh sách các nhánh hiện hành, ta dùng lệnh: "-git branch". Một danh sách các nhánh sẽ hiện ra. Dấu sao đánh dấu nhánh mà ta đang ở. Để tạo một nhánh mới, ta thêm tên của nhánh vào sau lệnh git branch: "-git branch branch_name". Sau khi tạo nhánh mới, ta phải chuyển qua nhánh đó mới có thể sử dụng được: "- git checkout branch_name". Giờ đây, ta có thể làm việc trên nhánh mới vừa tạo. Những commit thực hiện ở nhánh mới hoàn toàn không ảnh hưởng gì đến nhánh chính (gọi là master) cho đến khi ta ra lệnh hợp nhất (merge).

Để quay về nhánh chính, ta dùng lệnh: "-git checkout master".Ta cũng có thể kết hợp cả 2 lệnh tạo nhánh và chuyển qua nhánh đó trong cùng một dòng: "-git checkout –b branch_name"

Ngoài ra, để xem toàn bộ commit của tất cả các nhánh thay vì chỉ một nhánh như ta đã biết, ta dùng lệnh: "-git log—all". Để thêm tên nhánh vào trong danh sách kết quả cho dễ xem, ta thêm vào –decorate: "-git log –all-decorate"

3.10 Một số lệnh trên Git

- **git init**

Tạo 1 repository

- **git clone <url>**

Câu lệnh trên sẽ tạo một thư mục mới sao chép từ repository trên Github về local máy tính

- **git add <filename>**

Thêm 1 file vào để Git theo dõi

- **git status**

Kiểm tra trạng thái của tệp tin

- **git commit -m "message"**

Lưu các thay đổi file vào kho cùng với 1 tin nhắn

- **git pull**

Kéo các thay đổi mới nhất từ repository

- **git revert**

Trở lại 1 commit

- **git log**

Gửi lại danh sách lịch sử commits tin nhắn

- **git tag**

Liệt kê các thẻ thao tác Git

- **git tag –a <tagname> -m “message for tag”**

Thêm một thẻ cho commit mới nhất

- **git tag –d <tagname>**

Xóa một thẻ

- **git push origin <tagname>**

Chia sẻ thẻ vào repository

- **git checkout –b <brachname> <tagname>**

Tạo ra một nhánh mới ở một thẻ củ thể

- **git branch**

Liệt kê các nhánh có sẵn trong git

- **git branch <branch_name>**

Tạo ra một nhánh mới

- **git checkout <new_branch_name>**

Chuyển sang 1 nhánh mới

- **git merge <branch_name>**

Hợp nhất nhánh với nhánh hiện tại

- **git clear –f -d**

Xóa các file trước khi xóa

- **git clear –f**

Xóa folder

- **git help clone**

Xem thông tin hỗ trợ cho 1 câu lệnh git

- **git diff**

Xem thay đổi (chưa được thêm)vào file hiện tại

- **git reset**

Reset lại trước thời điểm 1 commit

- **git diff branch-name path/to/file**

Xem thay đổi trong một file, giữa state hiện tại và một branch

- **git fetch orgin**

Lấy tất cả các branches

- **git for-each-ref --sort=-committerdate refs/heads/ | head**

Liệt kê các nhánh theo trình tự sử dụng gần nhất

- **git remote -v**

Liệt kê các remote urls

- **git diff**

Xem thay đổi (chưa đc add) của những file hiện tại

- **git diff –cached**

Xem thay đổi (đã được add chưa commit)

- **git diff origin/master**

Xem thay đổi giữa local mà maste

- **git diff COMMIT1_ID COMMIT2_ID**

Xem thay đổi giữa hai commits

- **git diff --name-only COMMIT1_ID COMMIT2_ID**

Xem những files thay đổi giữa hai commits

- **git diff-tree -no-commit-id --name-only -r COMMIT_ID**
- **git diff --cached origin/master**

Xem thay đổi trước khi push

- **git commit --amend -m "New commit mesage"**

Đổi message của commit cuối

- **git push origin master**

Push local commits sang nhánh remote

CHƯƠNG 4. ỨNG DỤNG

4.1 Bài toán minh họa - Giải quyết vấn đề quản lí mã nguồn

Giả sử trong 1 công ty có 1 dự án , Công việc của mỗi lập trình viên sẽ đảm nhận làm 1 chức năng trong dự án đó , Mỗi người sẽ làm việc với nhau bằng cách mỗi người tự viết code rồi gửi cho nhau thông qua email, facebook hay một phương thức gửi dữ liệu thông thường nào đó. Mỗi người sau khi nhận được code của nhau sẽ tiến hành xem và sửa lại, sau đó cùng nhau lắp ghép để hoàn thành 1 dự án cụ thể. Tuy nhiên , việc vô tình người này sửa sai code của người kia dẫn tới phần mềm bị lỗi và việc không sao lưu do phần mềm quá lớn dẫn tới việc mọi người lại phải cùng nhau giải quyết những vấn đề đó , hơn nữa việc gửi code đi gửi code lại qua e mail hay 1 phương thức gửi dữ liệu thông thường nào đó rất mất thời gian và gửi rất nhiều lần. Vì vậy , việc sử dụng Git sẽ giải quyết vấn đề đơn giản hơn rất nhiều.

Chúng ta sẽ tạo ra kho chứa trên các máy chủ, mỗi máy tính có thể tạo bản sao (clone) lại mã nguồn từ một kho chứa và Github chính là một dịch vụ máy chủ (kho chứa) repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc. Mỗi người có thể làm việc trên 1 nhánh(branch) của riêng mình mà không ảnh hưởng đến code của người khác .Kéo code về máy tính của mình từ kho chứa hoặc các nhánh của kho chứa bất kì lúc nào và ghép code của mình với người khác hoàn toàn dễ dàng.

4.2 Giải quyết bài toán

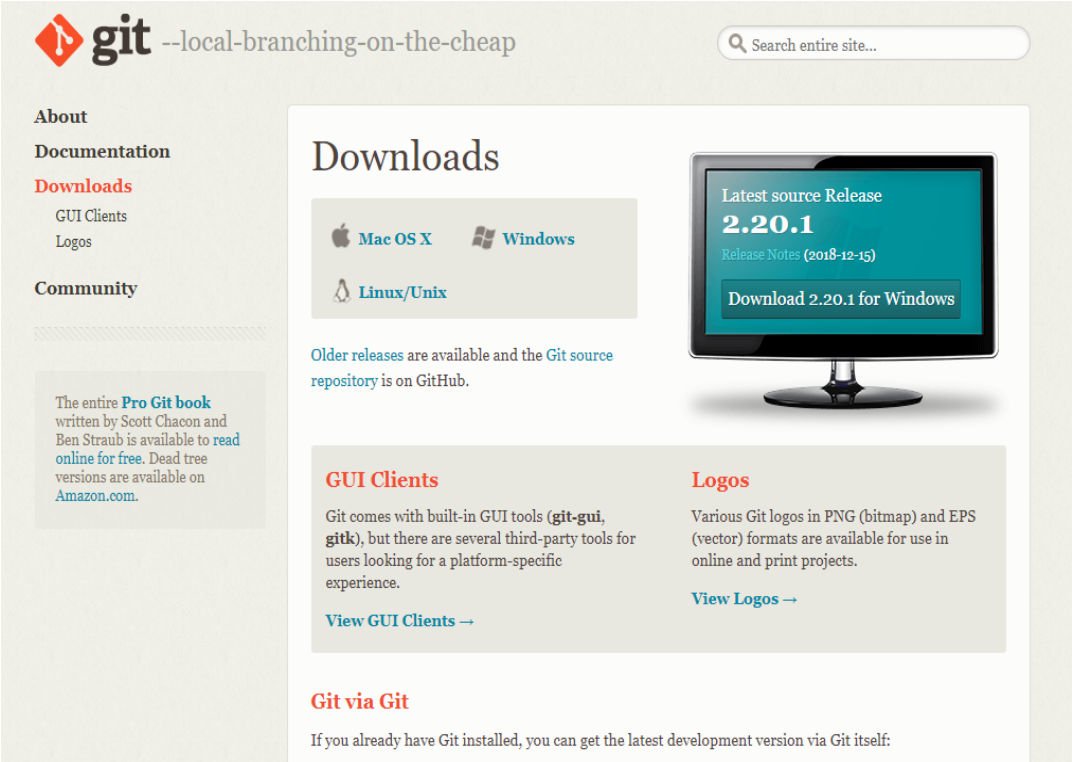
4.2.1 Với Git sử dụng dòng lệnh

4.2.1.1 Hướng dẫn tải và cài đặt Git

Để thao tác với Github trên máy tính chúng ta cần tải 1 công cụ là Git.

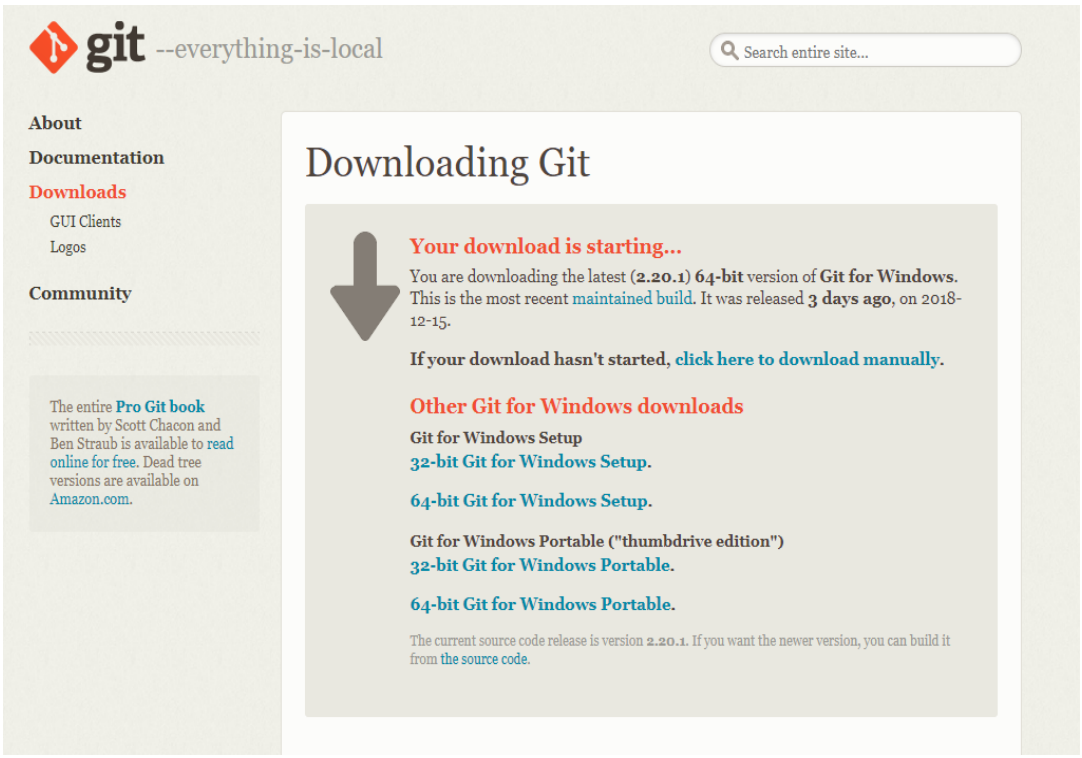
Hướng dẫn tải Git và cài đặt.

Truy cập địa chỉ **https://git-scm.com/downloads** bằng trình duyệt:



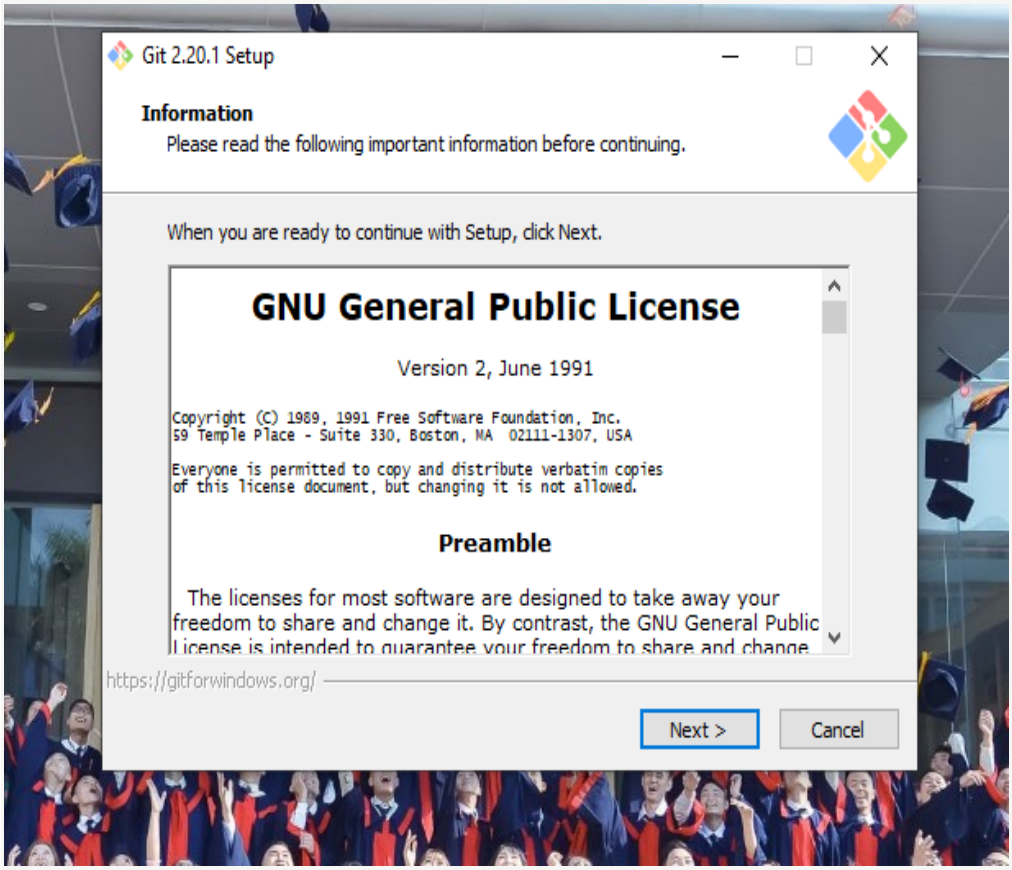
Hình 4.1 Trang tải Git

Chọn phiên bản muốn tải về :

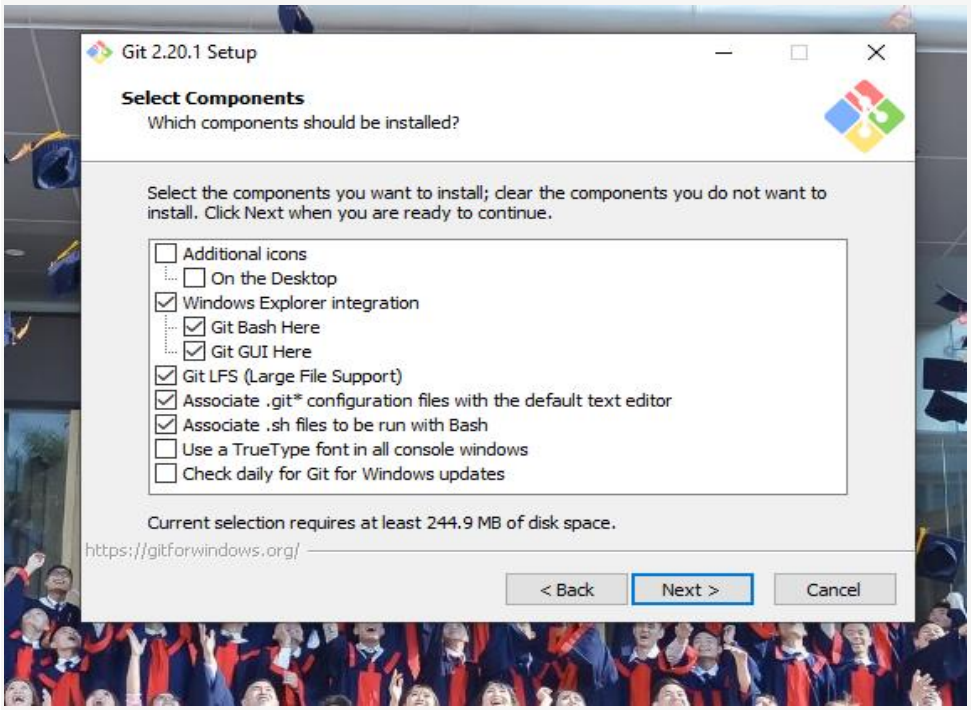


Hình 4.2 Lựa chọn phiên bản

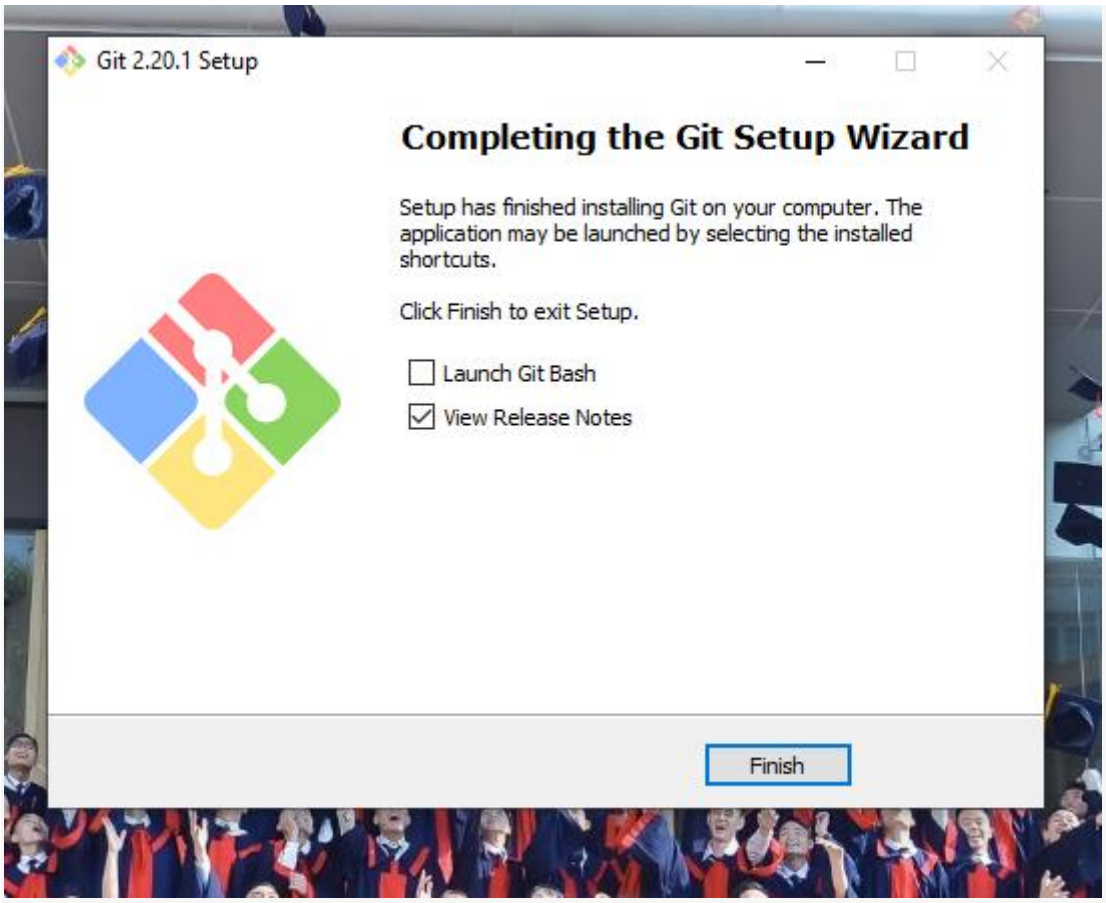
Cài đặt Git trên Windows rất đơn giản, về cơ bản bạn có thể chấp nhận các mặc định và nhấn Next, Next ... cho tới khi hoàn thành.



Hình 4.3 Chấp thuận giấy phép



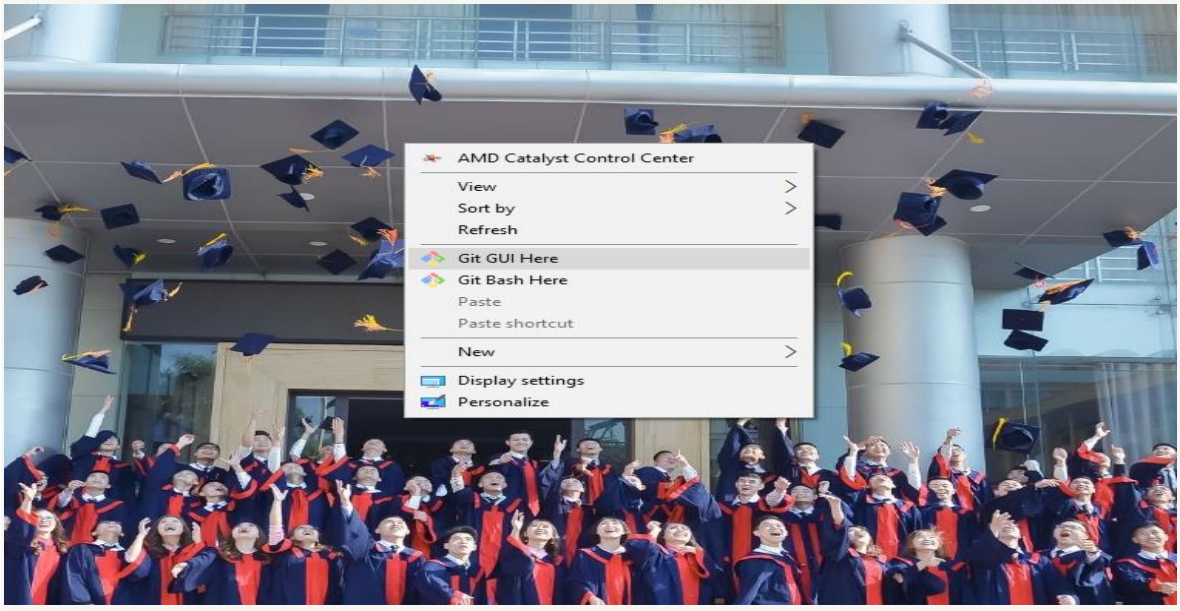
Hình 4.4 Chọn cấu phần muốn cài đặt



Hình 4.5 Cài đặt hoàn thành

OK, Ta vừa cài đặt xong Git.

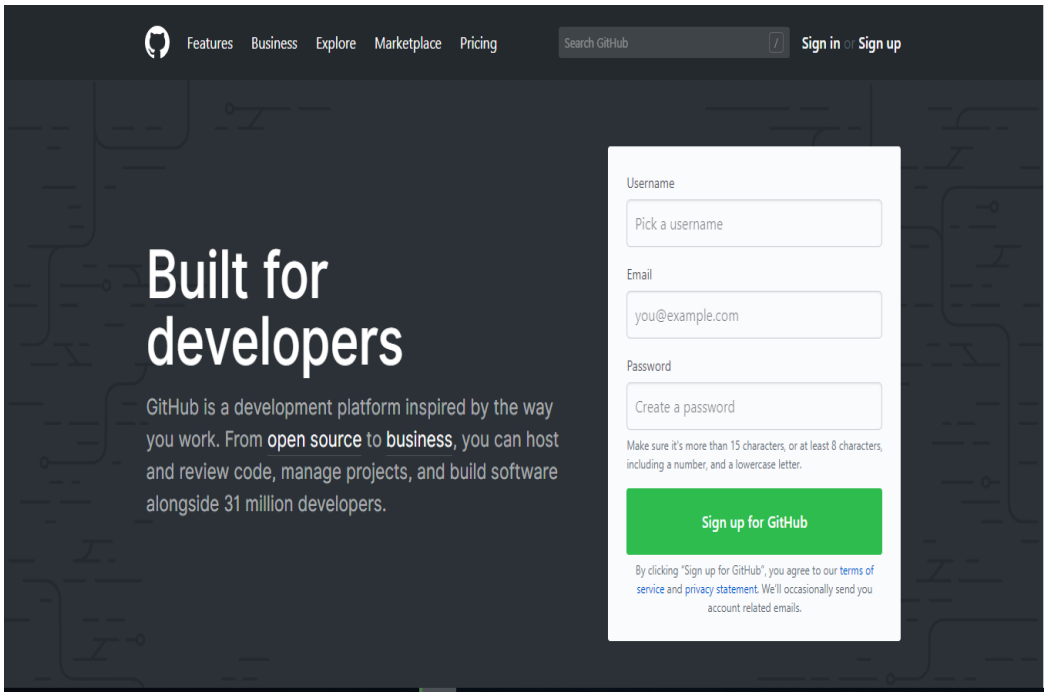
Sau khi cài đặt xong, nhấn phải chuột vào một thư mục bất kỳ, một Context-Menu sẽ hiển thị, bạn có thể nhìn thấy các Menu-Item của Git, điều này chứng tỏ rằng bạn đã cài đặt Git thành công.



Hình 4.6 Git

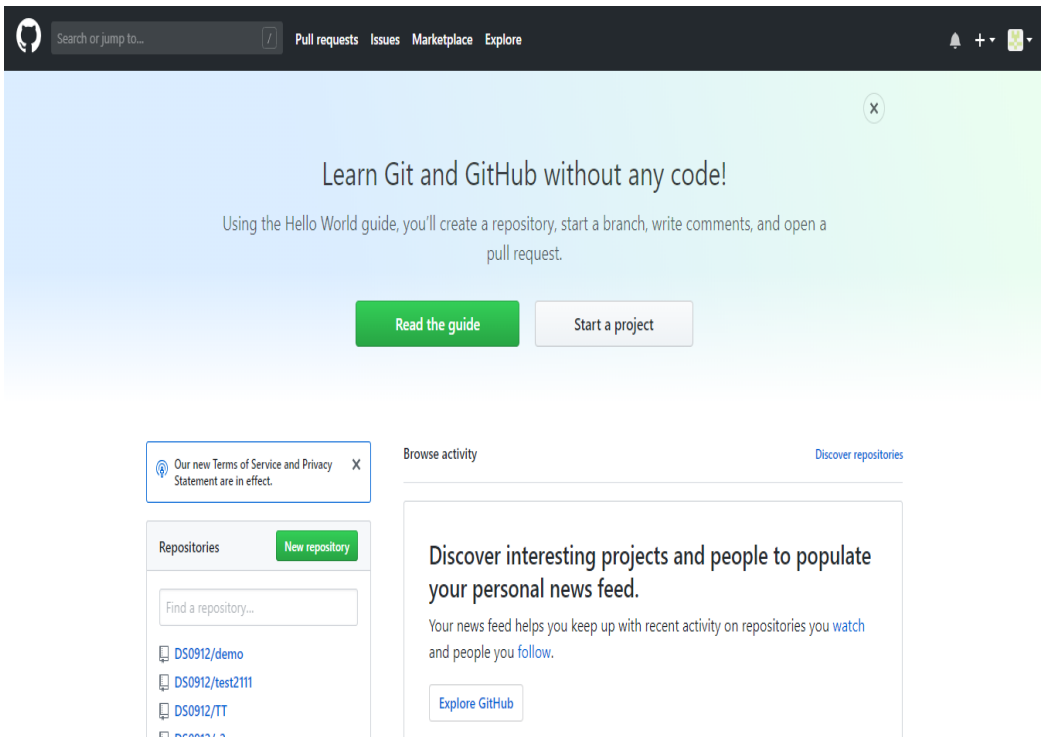
4.2.1.2 Hướng dẫn tạo tài khoản Github

Để sử dụng Github chúng ta truy cập địa chỉ: **https://github.com/**



Hình 4.7 Trang web của Github

Để sử dụng các chức năng của Github chúng ta cần có 1 tài khoản của Github. Nếu bạn chưa có tài khoản Github, bạn cần tạo 1 tài khoản. Đây là giao diện chính của Github.

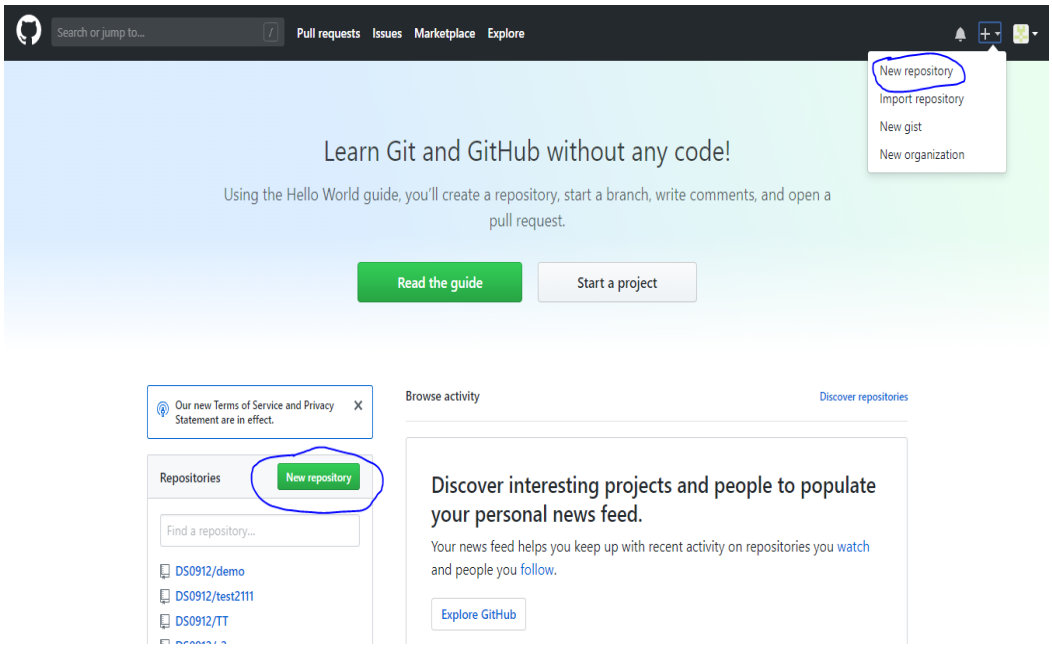


Hình 4.8 Giao diện sau khi đăng nhập

4.2.1.3 Tạo kho chứa

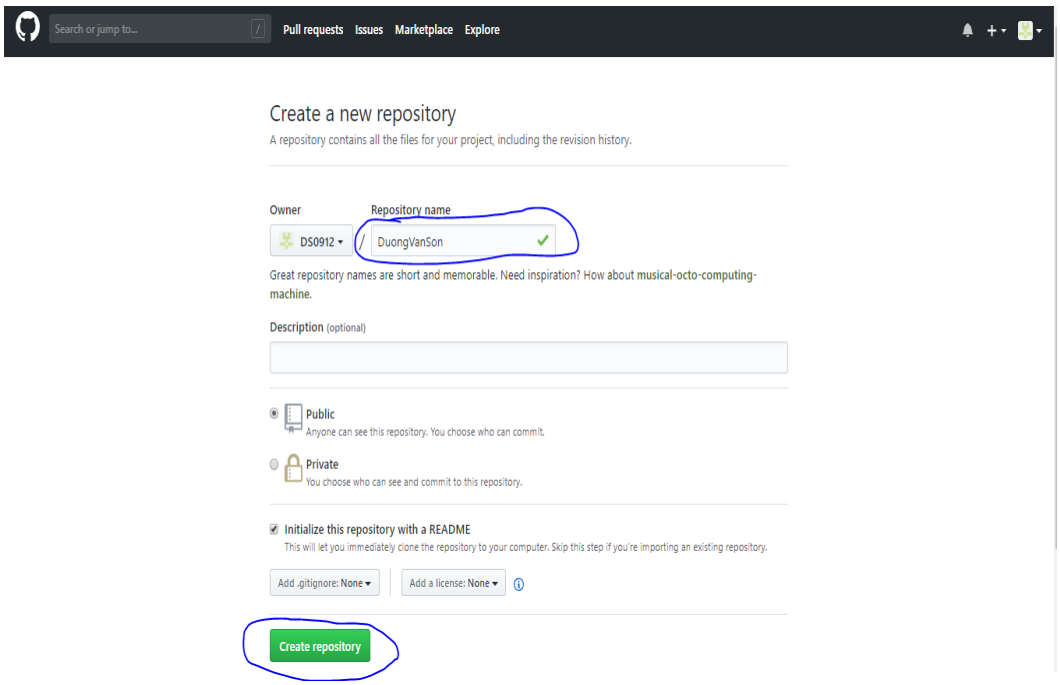
Để tạo 1 kho chứa, chúng ta làm như sau.

Click vào dấu cộng góc trên bên phải màn hình -> New repository hoặc vào nút New repository



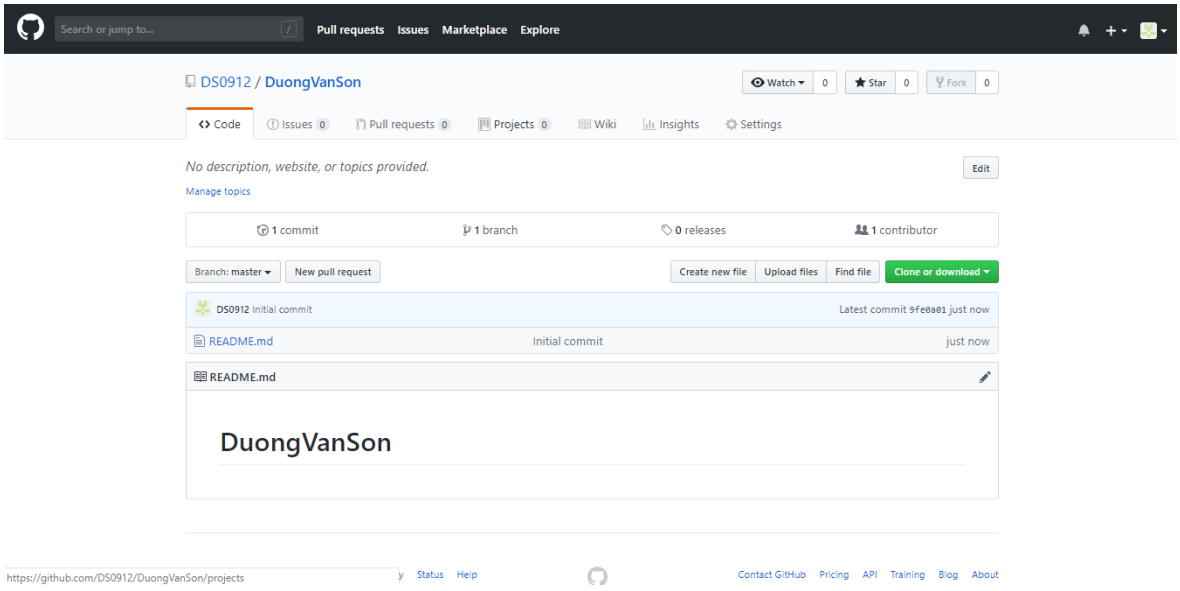
Hình 4.9 Click New repository

Điền tên kho chứa bạn muốn tạo -> Create repository



Hình 4.10 Đặt tên kho chứa và click để tạo

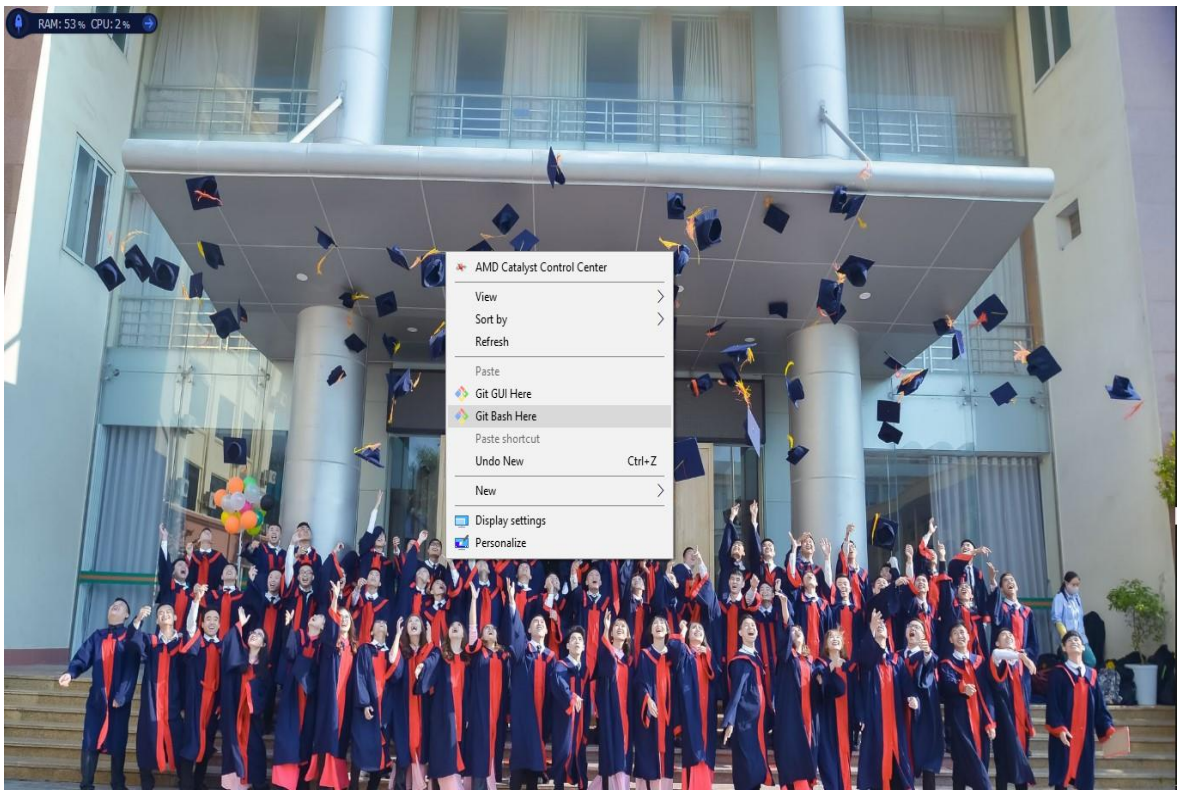
Vậy là chúng ta đã tạo thành công 1 kho chứa có tên DuongVanSon chứa 1 file README.md



Hình 4.11 Kho chứa DuongVanSon

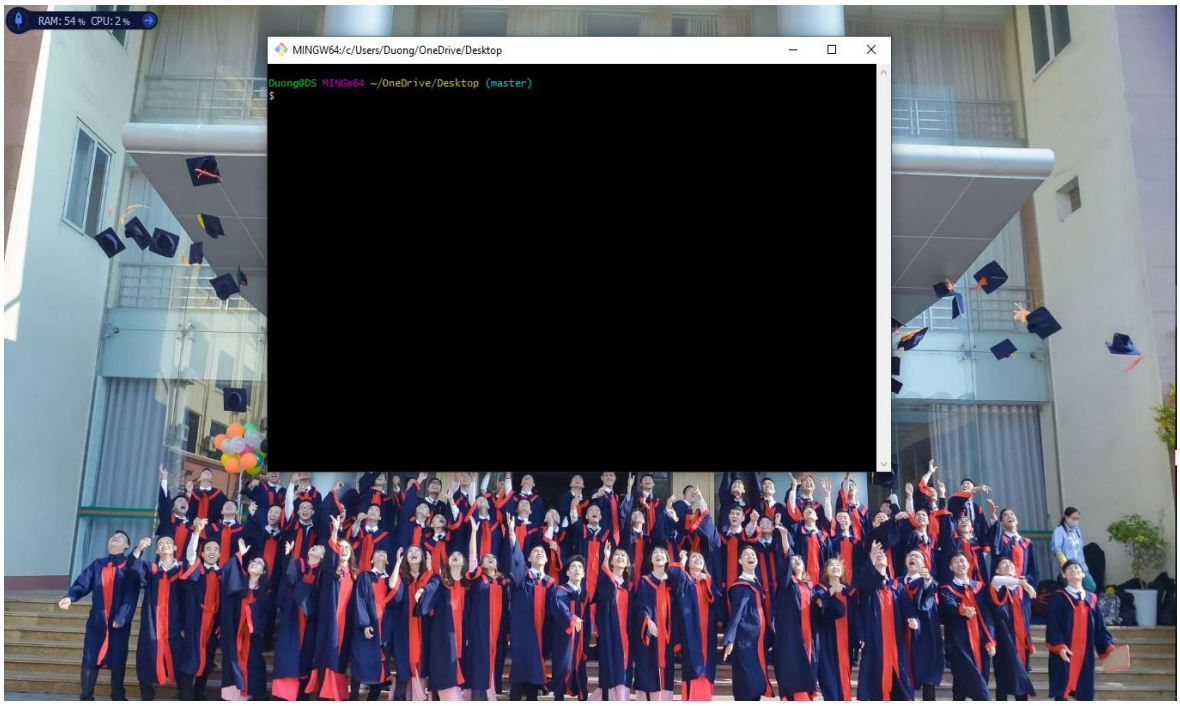
4.2.1.4 Git init

Đầu tiên, bạn cần phải chọn nơi mà bạn muốn sao lưu kho chứa về. Ở đây, em sẽ chọn Desktop. Click chuột phải ở file thao tác, chọn Git Bash Here



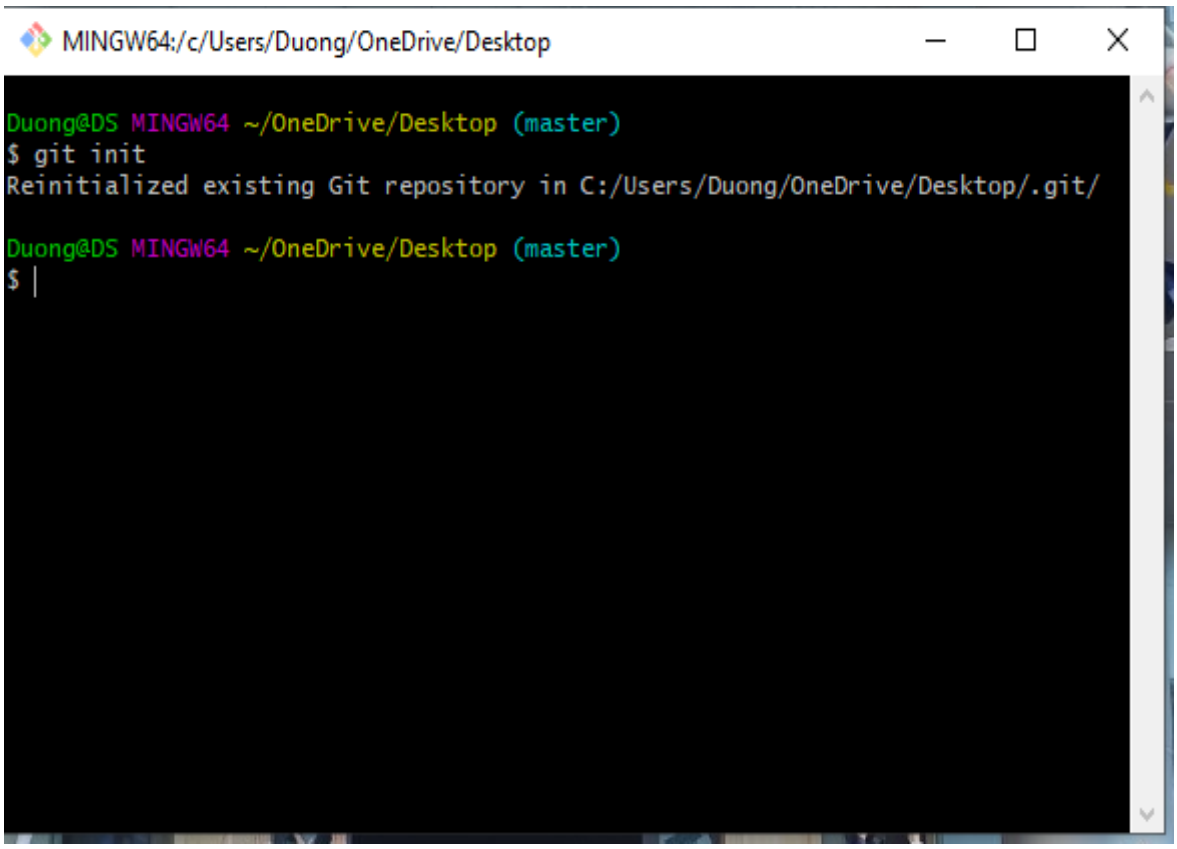
Hình 4.12 Git Bash Here

Chúng ta sẽ có 1 cửa sổ cmd như sau:



Hình 4.13 Cửa sổ dòng lệnh của Git

Gõ câu lệnh git init -> enter để tạo 1 file để chứa kho chứa chúng ta muốn sao lưu về:



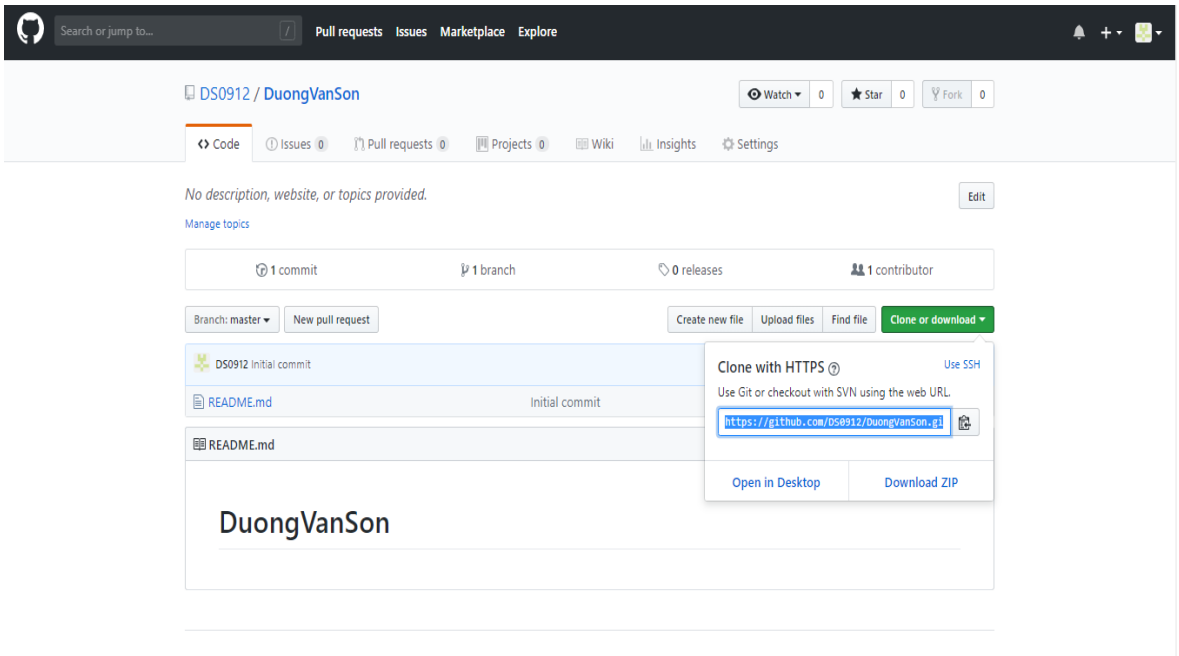
Hình 4.14 Git init

Lúc này trên màn hình Desktop chúng ta đã có 1 file .git

Hình 4.15 File .git

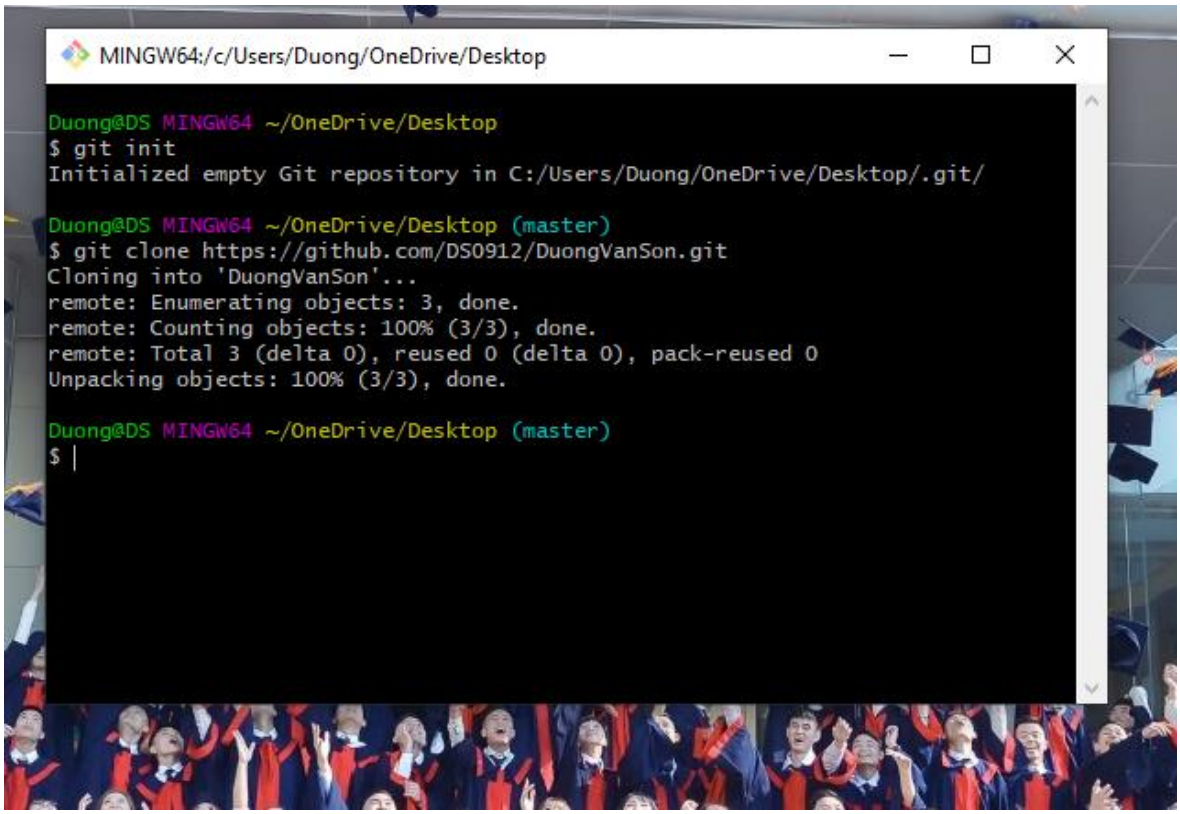
4.2.1.5 Git clone

Trước hết, để sao lưu kho chứa chúng ta cần copy đường dẫn của kho chưa đó trên web bằng cách: Click và nút Clone ỏ download -> click vào biểu tượng như hình dưới.



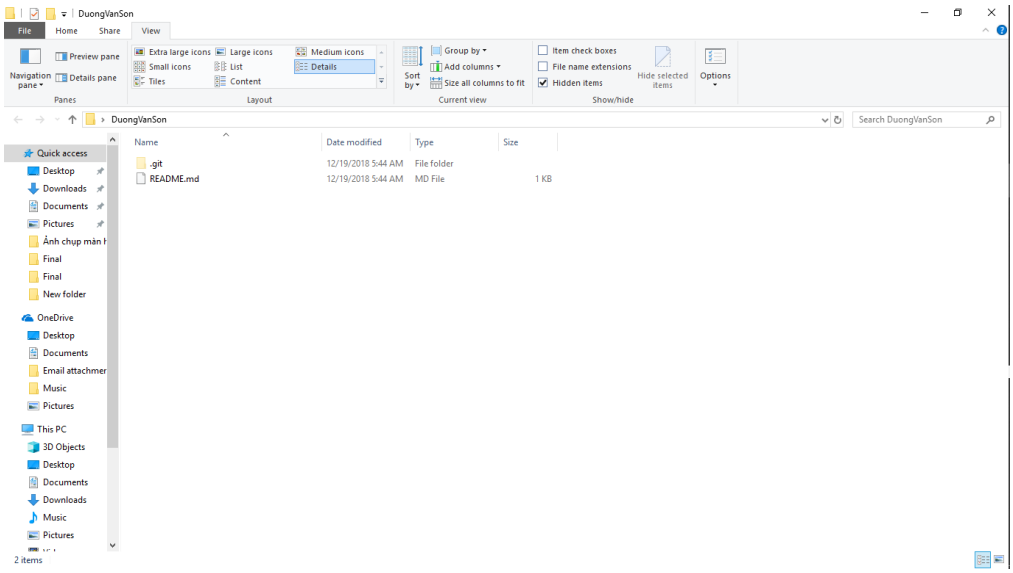
Hình 4.16 Sao chép đường dẫn

Sau đó, ở cửa sổ cmd ta gõ lệnh: git clone https://github.com/DS0912/DuongVanSon.git (là đường dẫn vừa copy ở bước trên) -> enter.



Hình 4.17 Git clone

Nếu cửa sổ cmd như trên thì bạn đã sao lưu kho chứa thành công về máy. Khi đó ta có thư mục



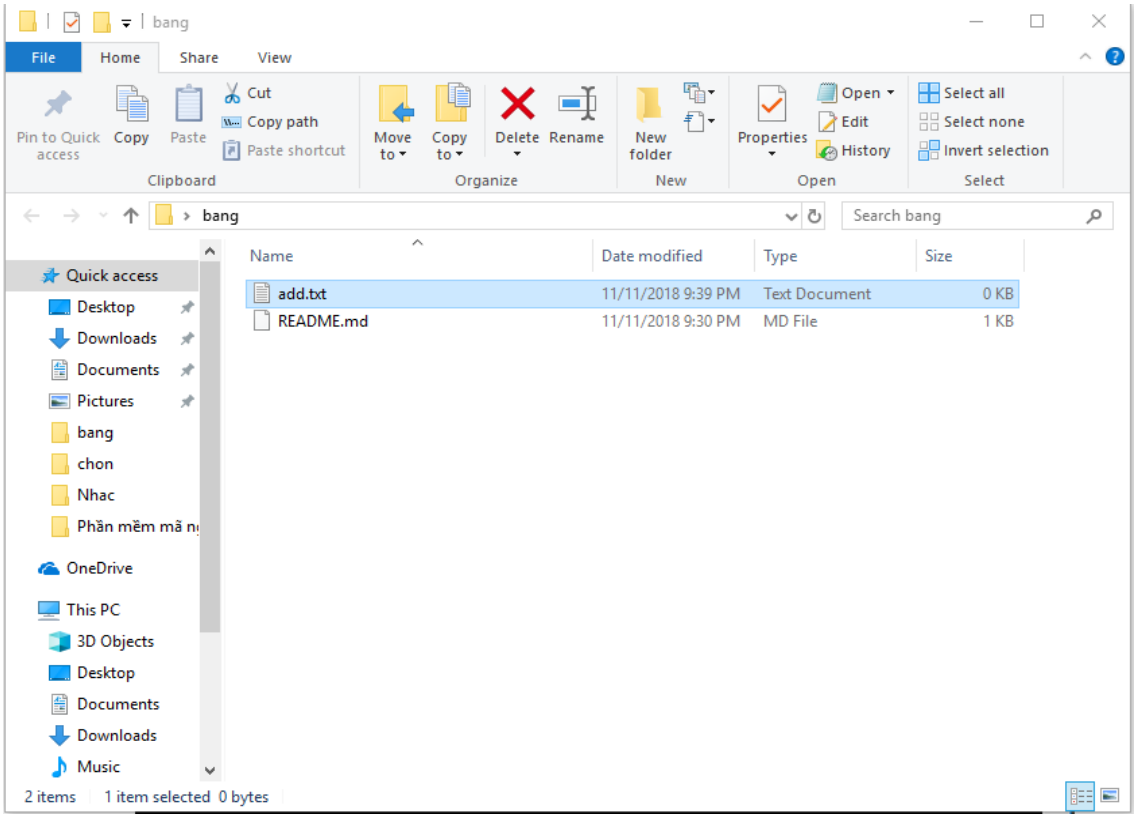
Hình 4.18 Kho chứa được clone về máy

4.2.1.6 Git add

git add <filename>

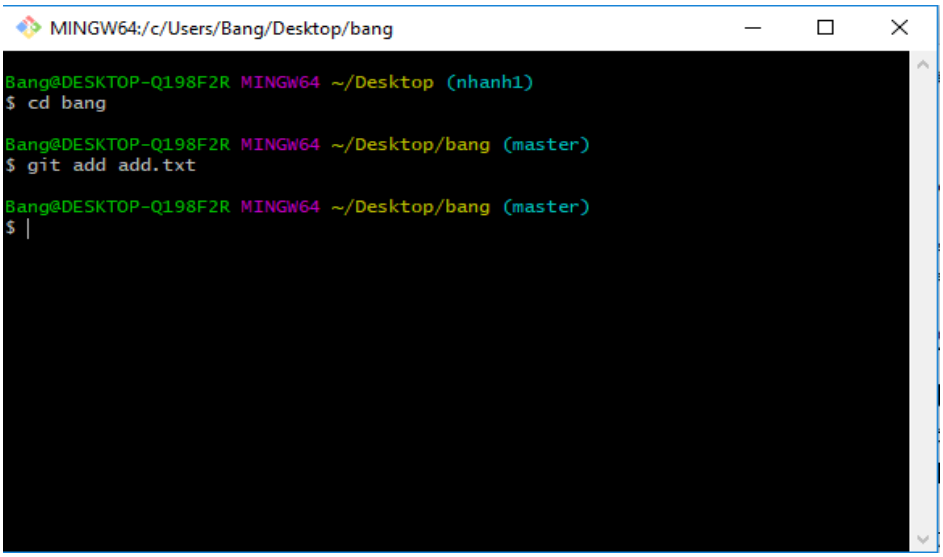
Thêm 1 file vào để Git theo dõi

Đầu tiên chúng ta sẽ tạo 1 file add.txt để add



Hình 4.19 File add.txt

Muốn add file chúng ta sẽ dùng lệnh **git add add.txt** hoặc nếu muốn add tất cả các file trong thư mục chúng ta có thể dùng lệnh **git add .**

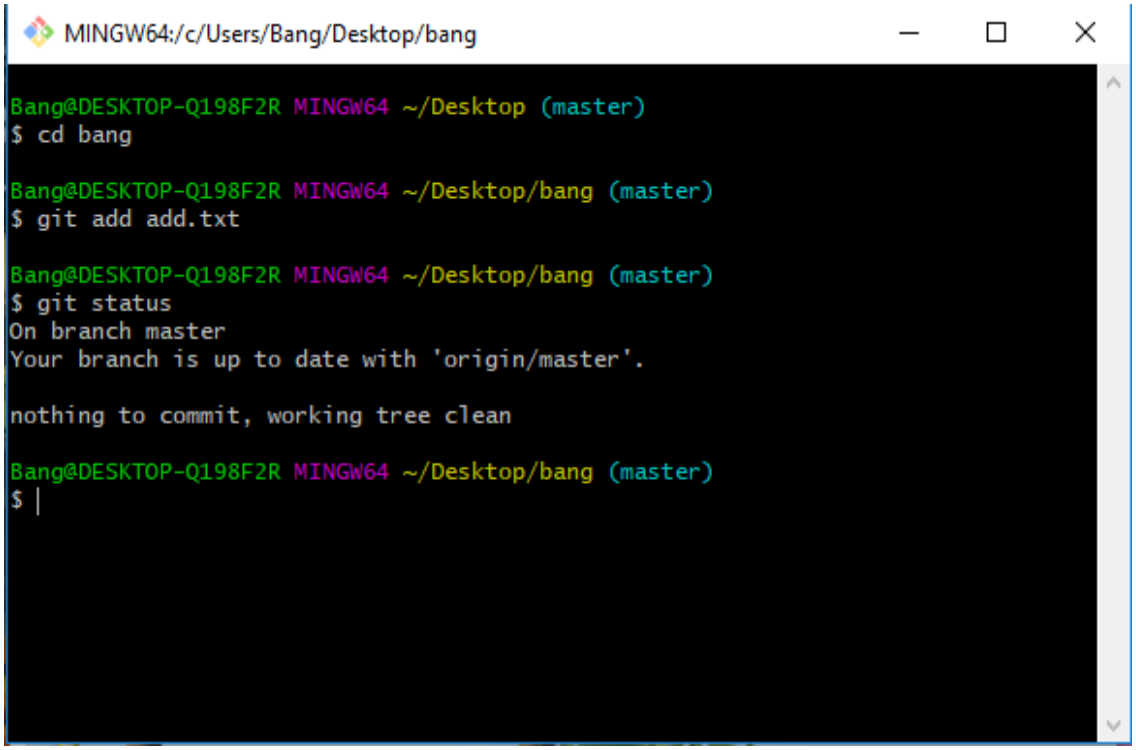


Hình 4.20 Git add

4.2.1.7 Kiểm tra trạng thái

git status

Kiểm tra trạng thái của tệp tin

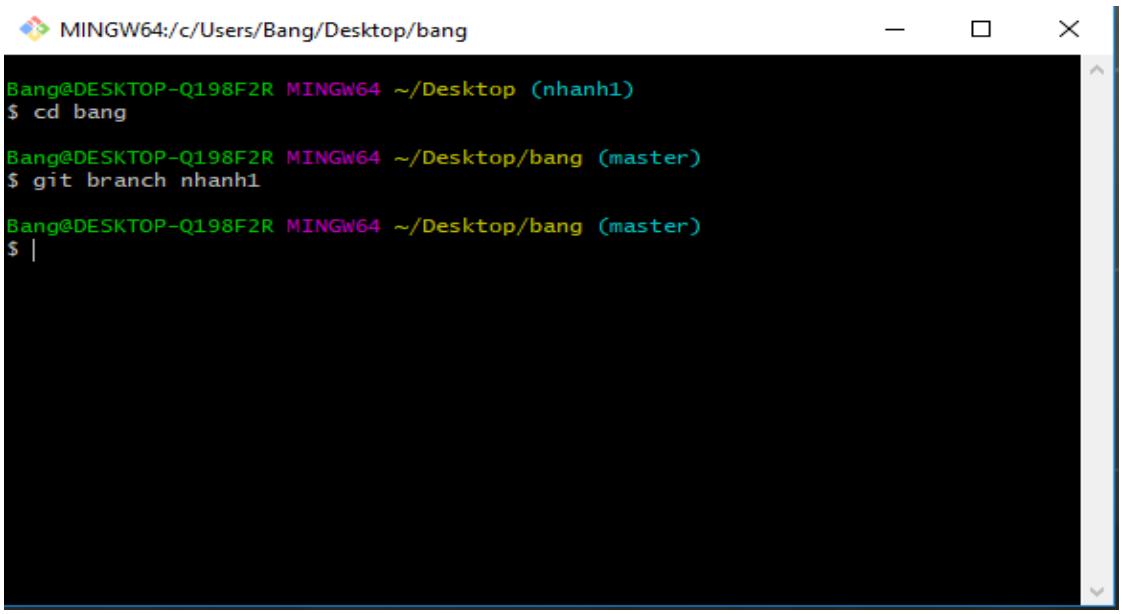


Hình 4.21 Kiểm tra trạng thái

4.2.1.8 Tạo nhánh

git branch <branch_name>

tạo ra một nhánh mới

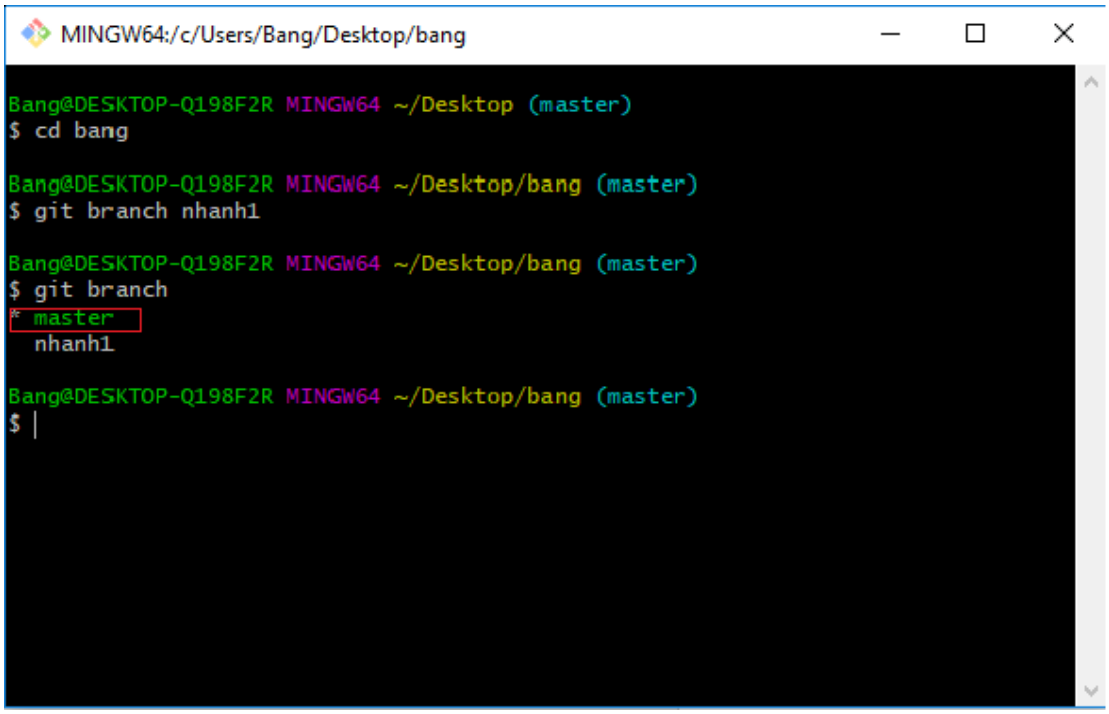


Hình 4.22 Tạo nhánh

git branch

Liệt kê các nhánh có sẵn trong git

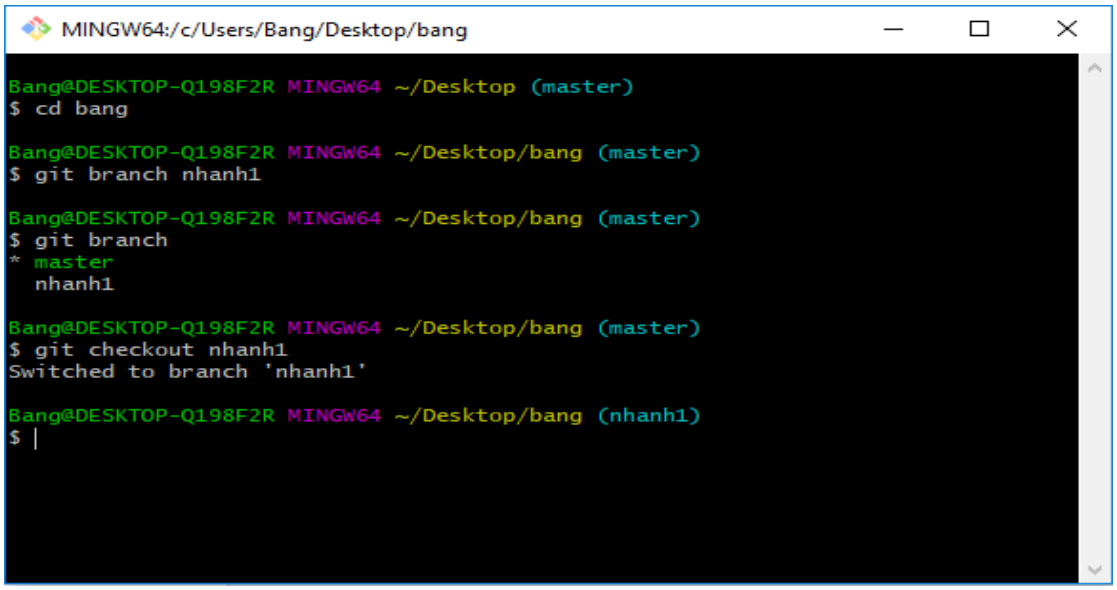
Nhánh có đánh dấu sao tức là biểu thị cho việc đang làm việc ở nhánh đó



Hình 4.23 Liệt kê các nhánh

git checkout <new_branch_name>

chuyển sang 1 nhánh mới

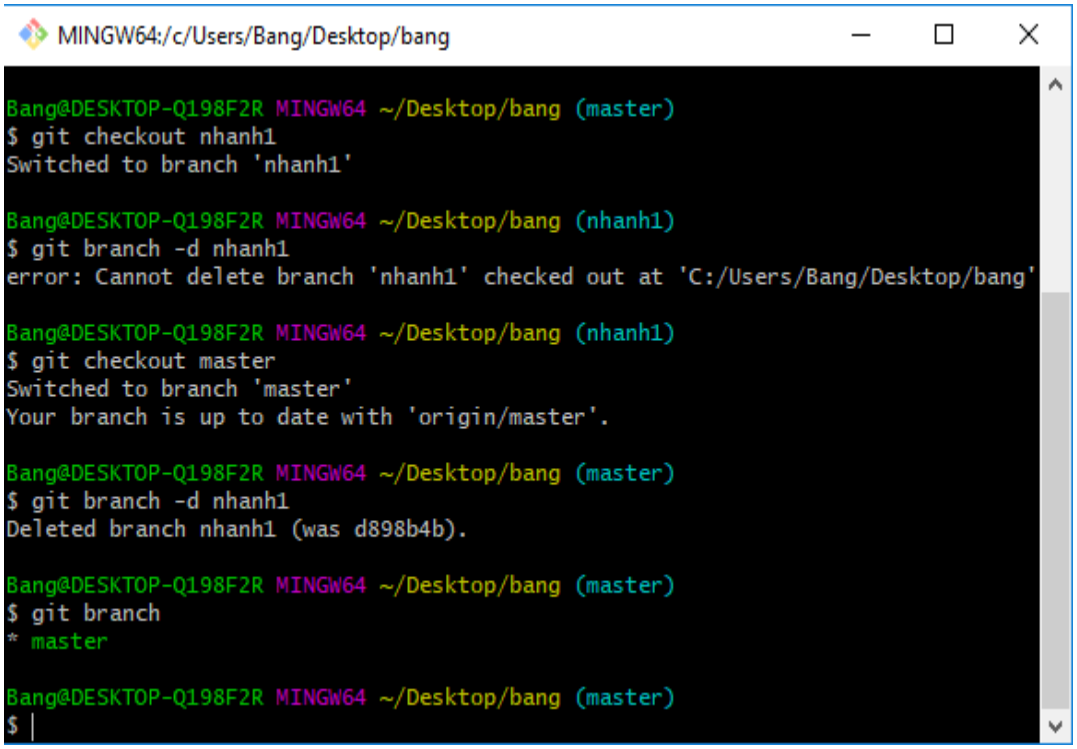


Hình 4.24 Chuyển sang nhánh mới

git branch -d <branch_name>

Lệnh xóa 1 nhánh

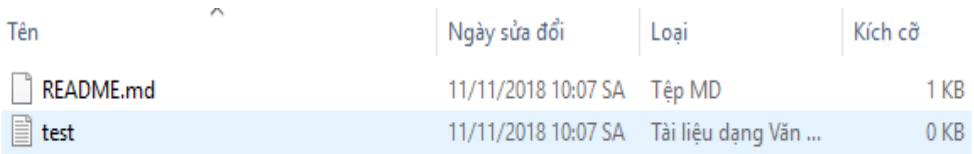
Khi xóa 1 nhánh thì chúng ta lưu ý phải dùng lệnh checkout để chuyển sang 1 nhánh khác thì mới có thể xóa được , và sau khi xóa chúng ta dùng lệnh git branch để kiểm tra. Nếu nhánh đó đang trong quá trình gộp thì ko thể xóa bằng lệnh trên mà chúng ta có thể sử dụng lệnh **git branch -D <branch_name>**



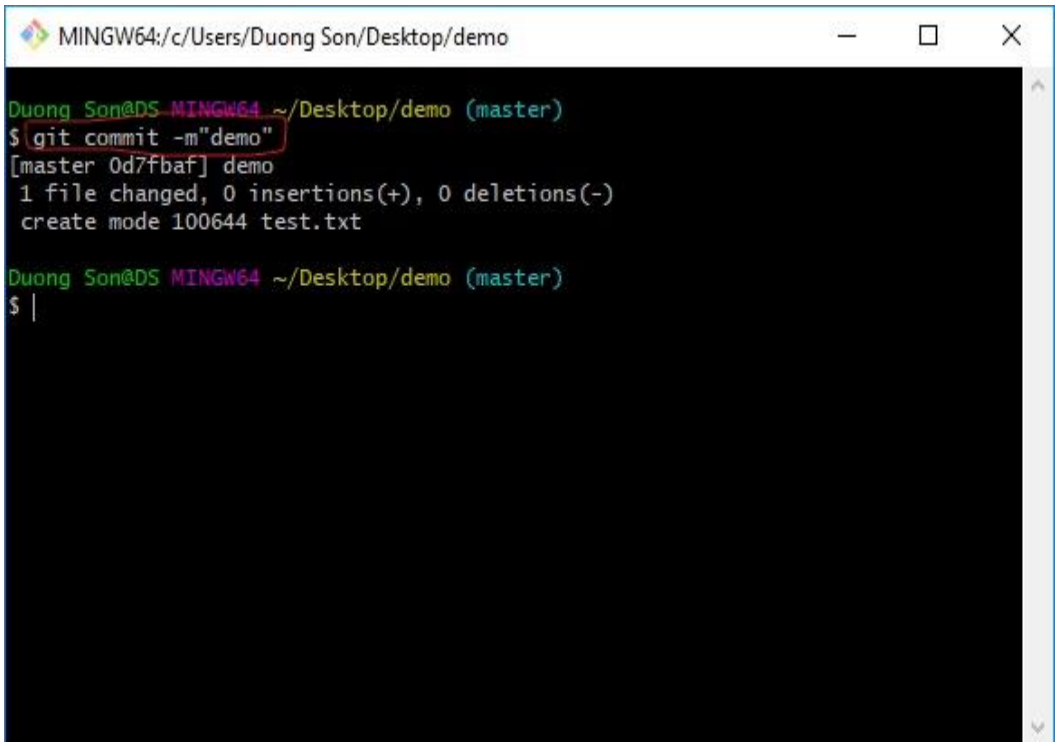
Hình 4.25 Xóa 1 nhánh

4.2.1.9 Git commit

Để thực hiện "commit" file test.txt ta gõ lệnh **git commit –m"message"**



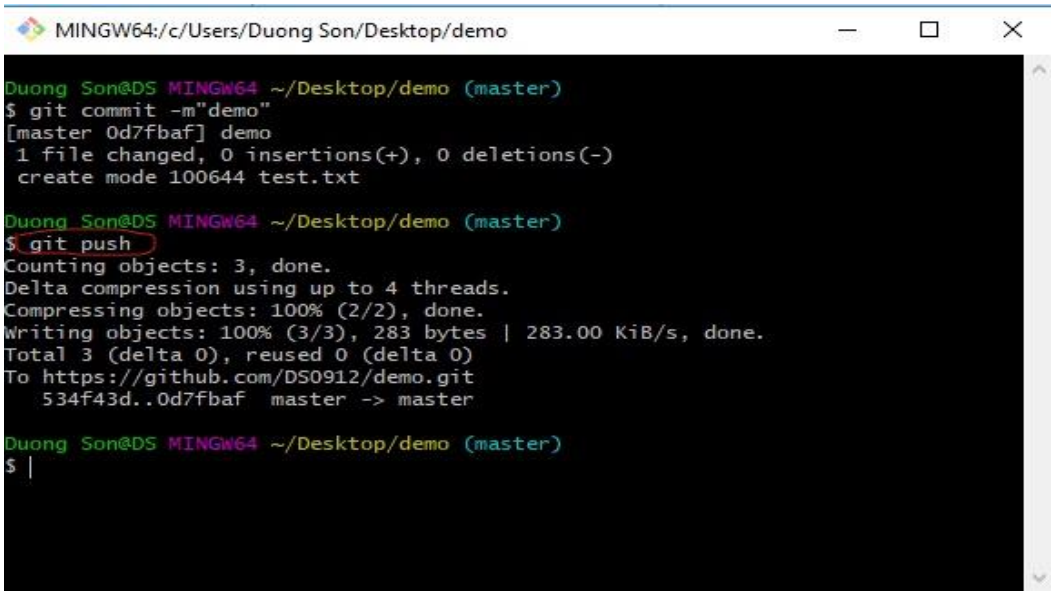
Hình 4.26 File test.txt



Hình 4.27 Lệnh Git commit

4.2.1.10 Đẩy thay đổi lên Github

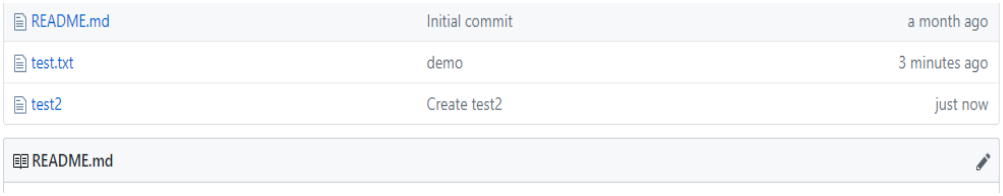
Để đẩy thay đổi lên Github ta dùng lệnh **git push**



Hình 4.28 Lệnh Git push

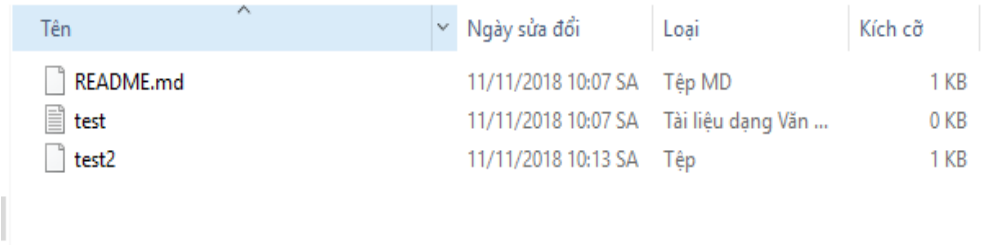
4.2.1.11 Lấy thay đổi trên Github về local

Để lấy thay đổi từ trên Github về local ta dùng lệnh **git pull,** ở đây **test2** là file mới cần lấy về



Hình 4.29 File test2

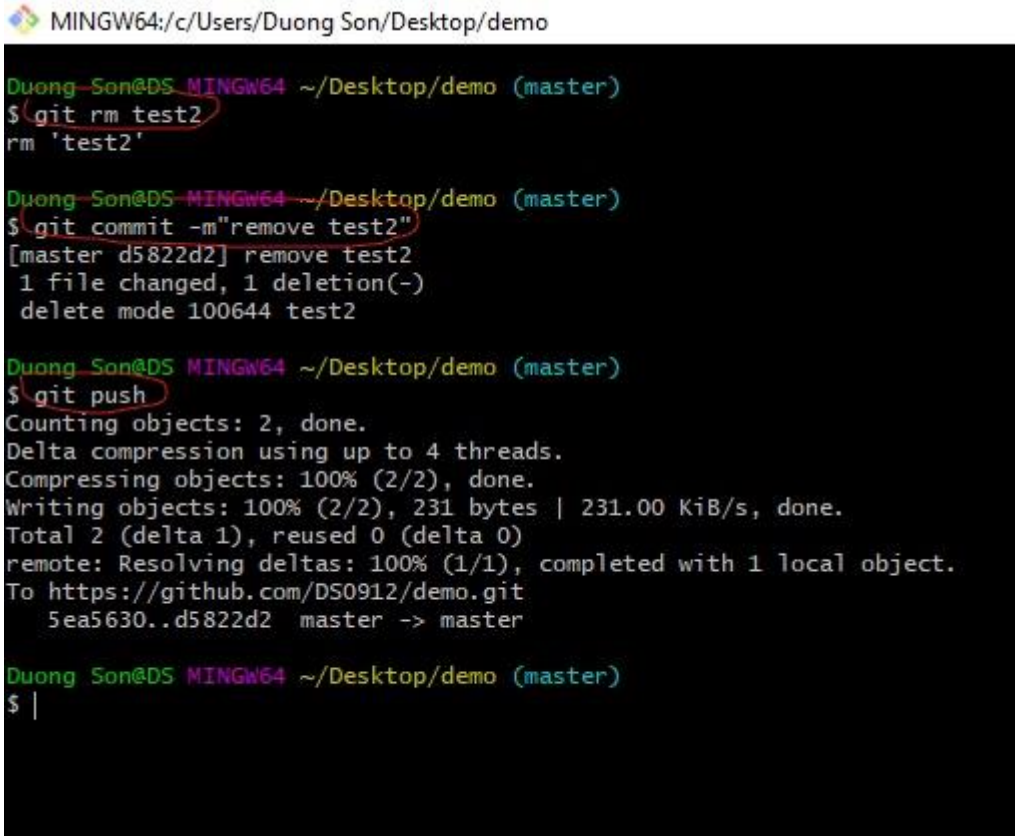
Hình 4.30 Git pull



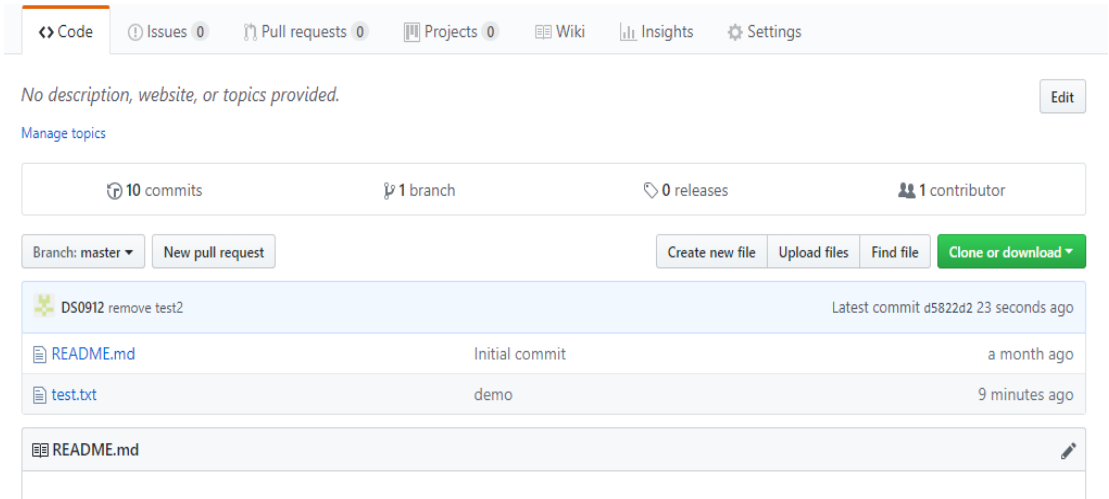
Hình 4.31 File đã được lấy về máy

4.2.1.12 Xóa file trên Github

Để xóa 1 file trên Github ta dùng lệnh **git rm <tên file>,** sau đó tiếp tùng dùng lệnh **git commit –m"message"** và **git push** để thực hiện thay đổi trên Github



Hình 4.32 Git rm

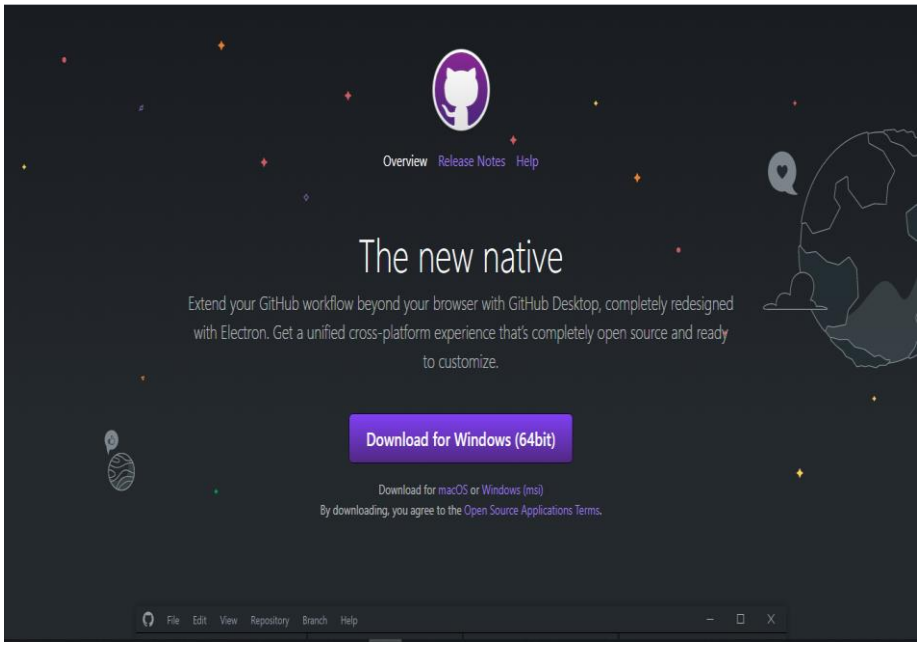


Hình 4.33 File test2 đã được xóa

4.2.2 Với Github Desktop

4.2.2.1 Tải và cài đặt Git Desktop

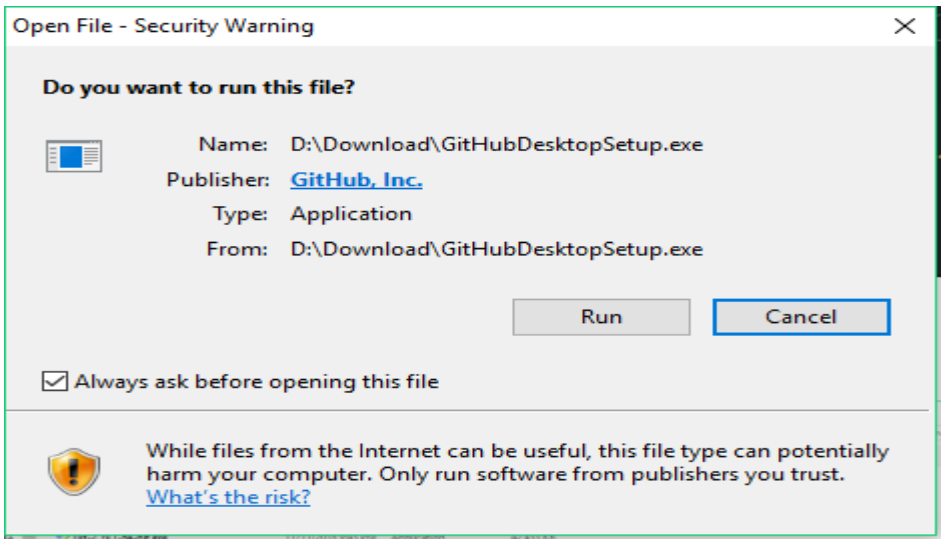
Đầu tiên để tải Git Desktop chúng ta truy cập địa chỉ: https://desktop.github.com/



Hình 4.34 Trang tải Github desktop

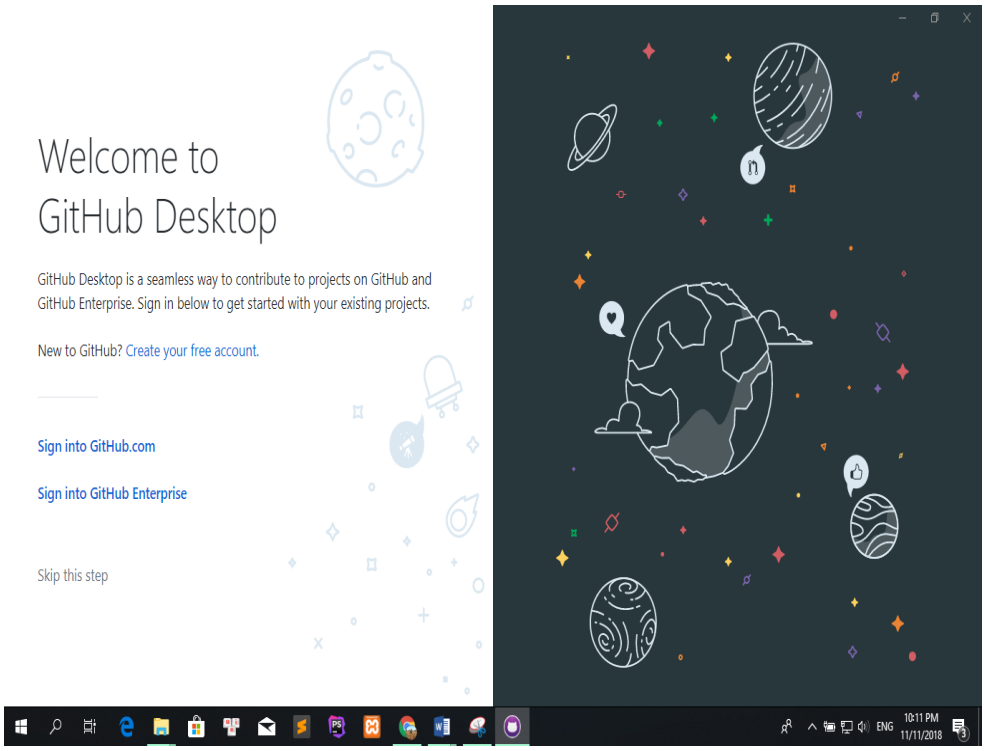
Click Download for Windows (64bit) hoặc hệ điều hành tương ứng của máy. Tiến hành download và cài đặt

Chúng ta chỉ cần chọn Run.



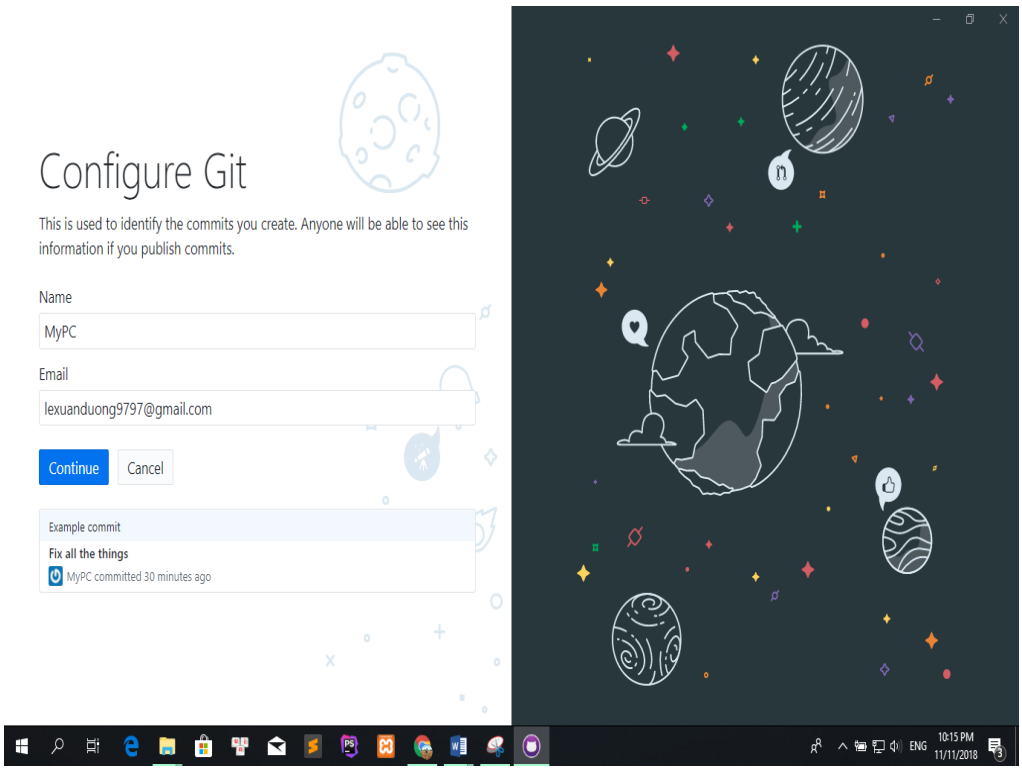
Hình 4.35 Click Run để cài đặt

Màn hình làm việc của Git Desktop:



Hình 4.36 Cài đặt xong

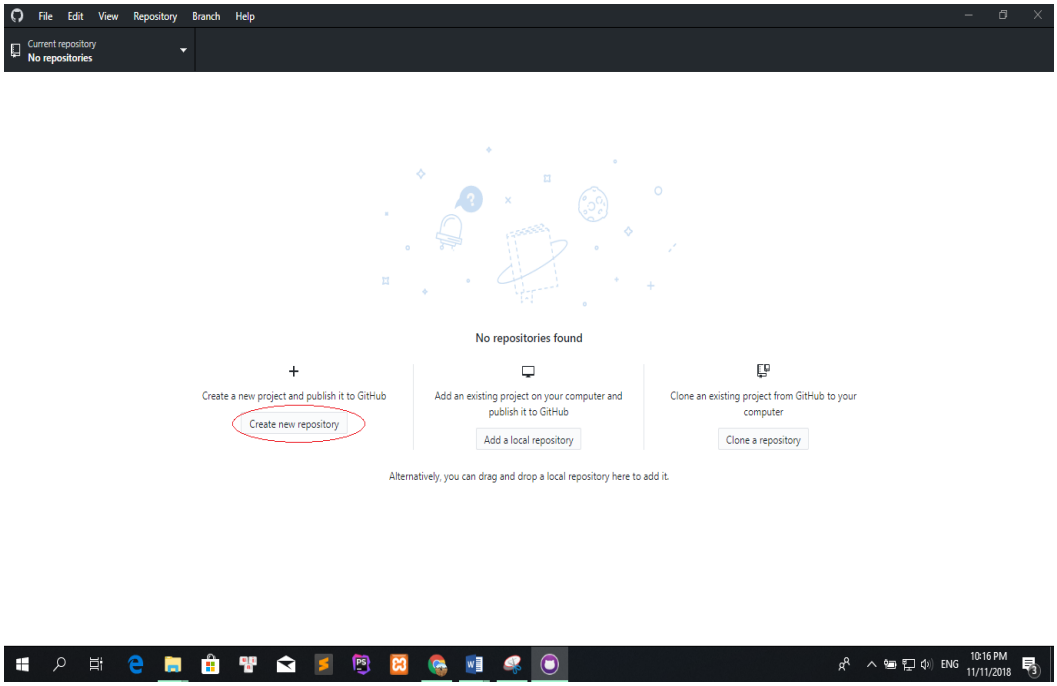
Nếu chưa có tài khoản chúng ta cần tạo 1 tài khoản (Tài khoản được tạo trên web https://github.com/). Chọn Continue.



Hình 4.37 Nhập thông tin và nhấn continue

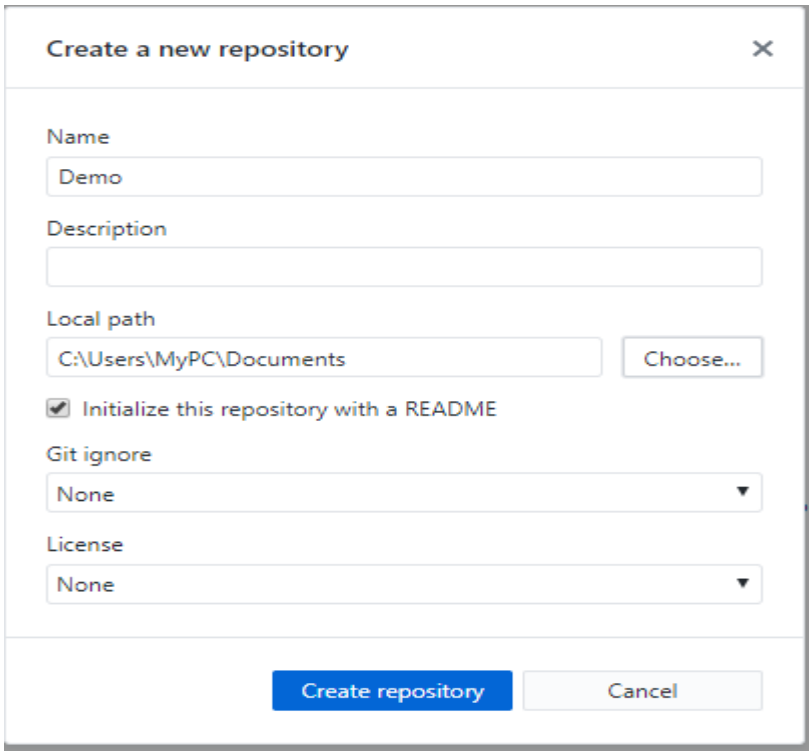
4.2.2.2 Tạo kho chứa bằng Git Desktop

Đăng nhập thành công, sẽ có màn hình làm việc sau. Để tạo 1 khô chứa, click vào Create new repository



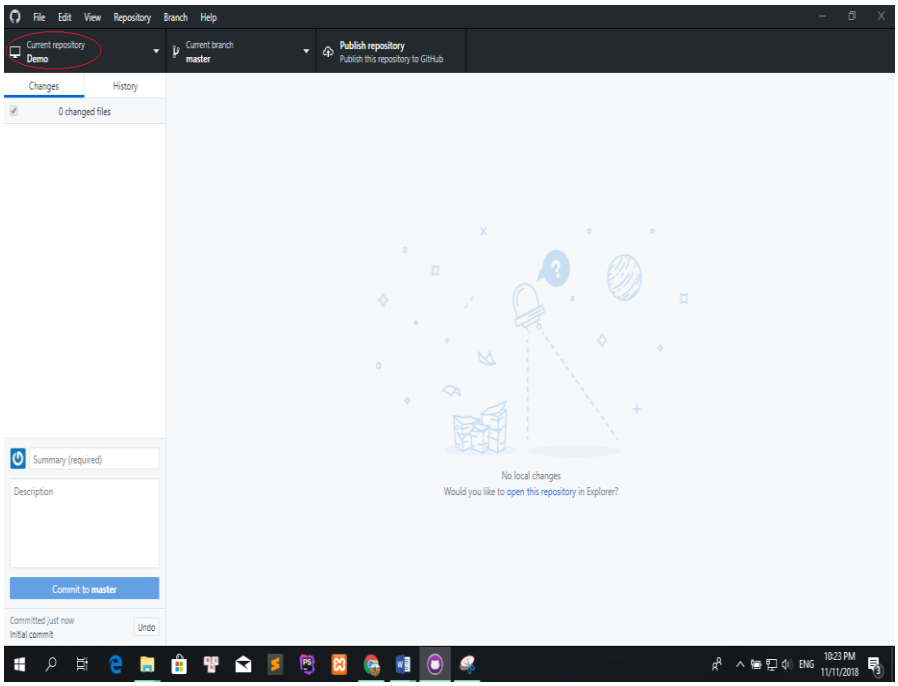
Hình 4.38 Create new repository

Điền tên kho chứa bạn muốn tạo, chọn nơi lưu trữ sau đó click create repository



Hình 4.39 Nhập thông tin

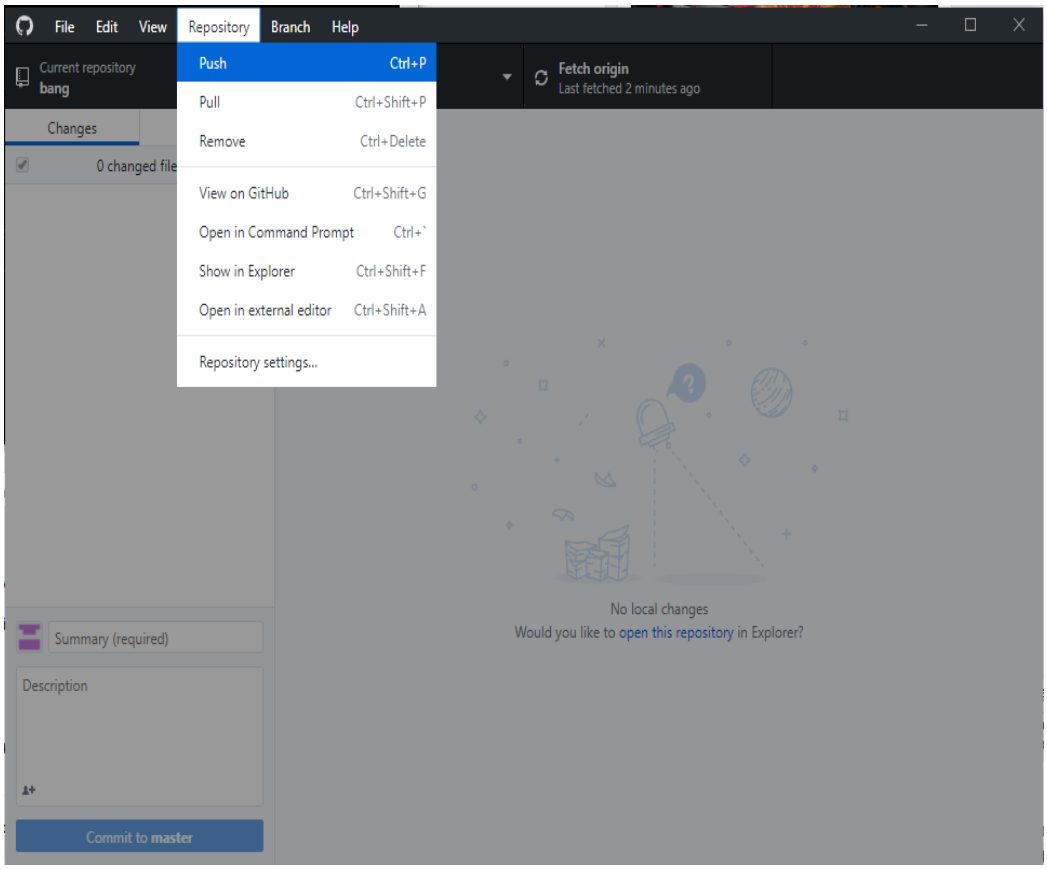
Vậy là chúng ta đã tạo được kho chứa có tên Demo



Hình 4.40 Tạo kho chứa thành công

4.2.2.3 Lấy thay đổi trên kho chứa bằng Git Desktop

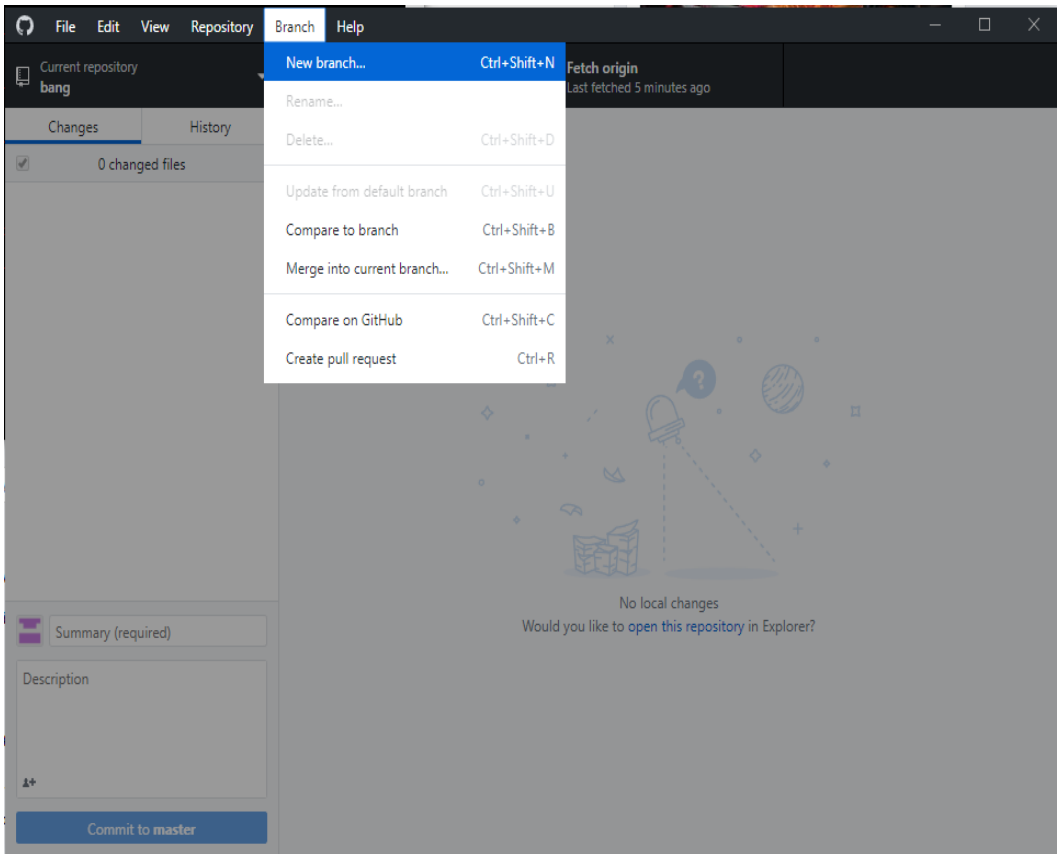
Kéo file trên kho chứa về ta dùng Pull



Hình 4.41 Chọn pull để lấy thay đổi về

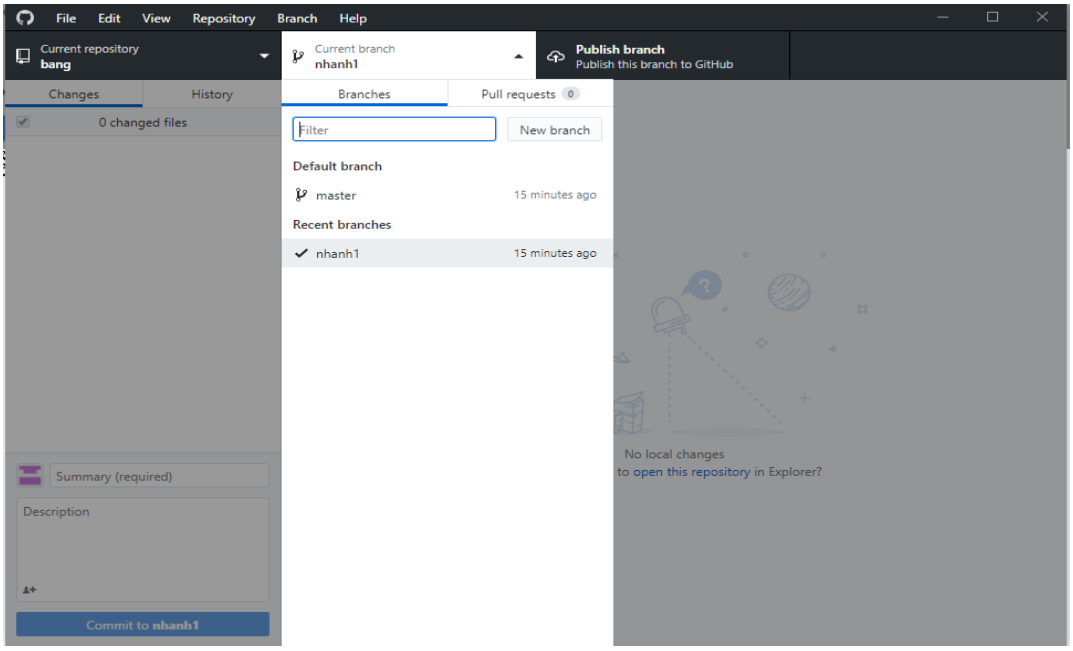
4.2.2.4 Tạo nhánh bằng Git Desktop

Tạo 1 nhánh mới



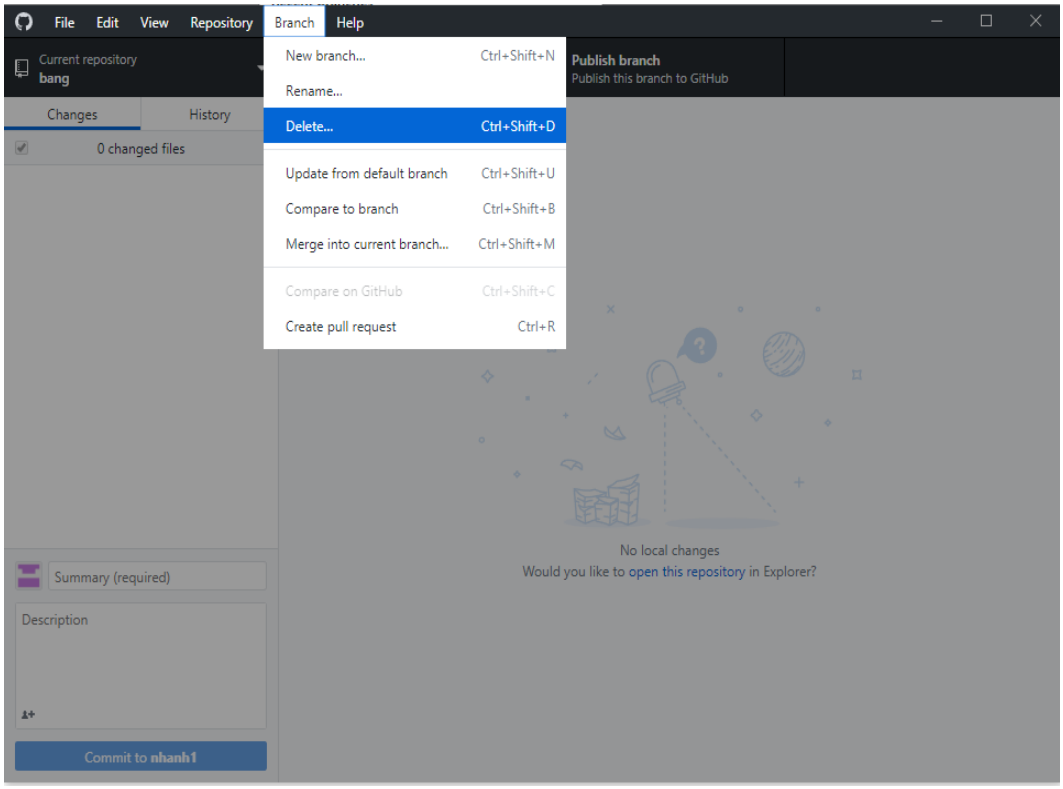
Hình 4.42 Tạo nhánh mới

Để chuyển giữa các nhánh ta click vào tab Curent branch và chọn nhánh muốn chuyển



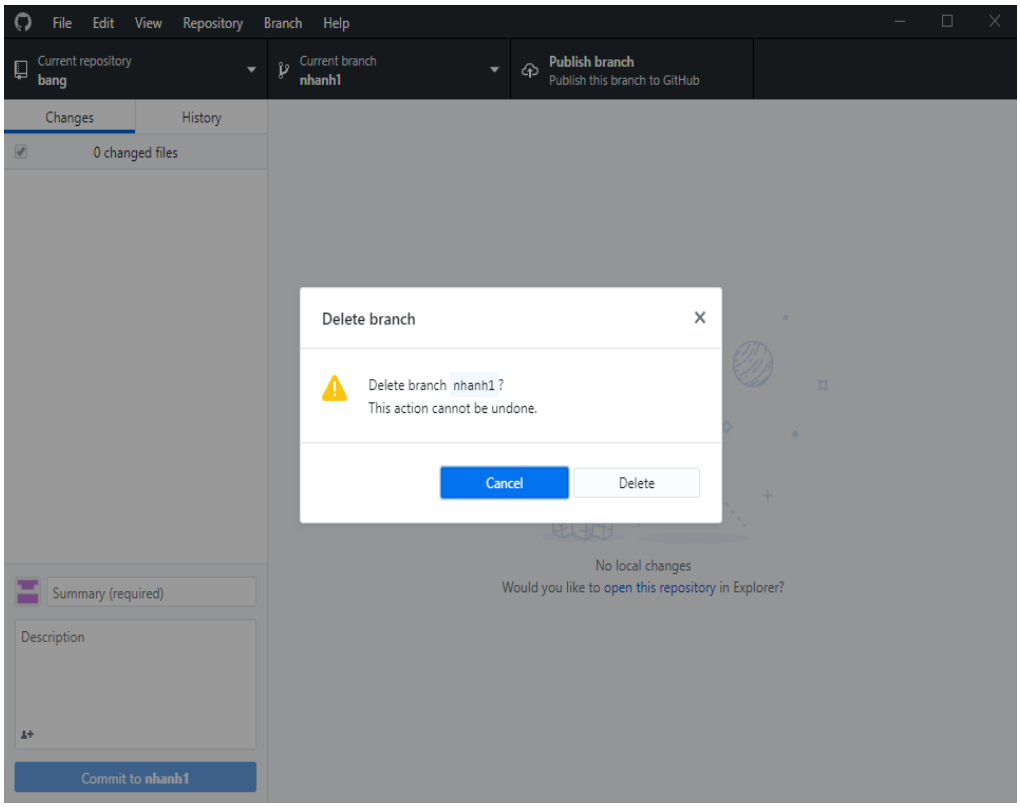
Hình 4.43 Chuyển nhánh làm việc

Để xóa 1 nhánh ta click vào tab branch và chọn Delete…



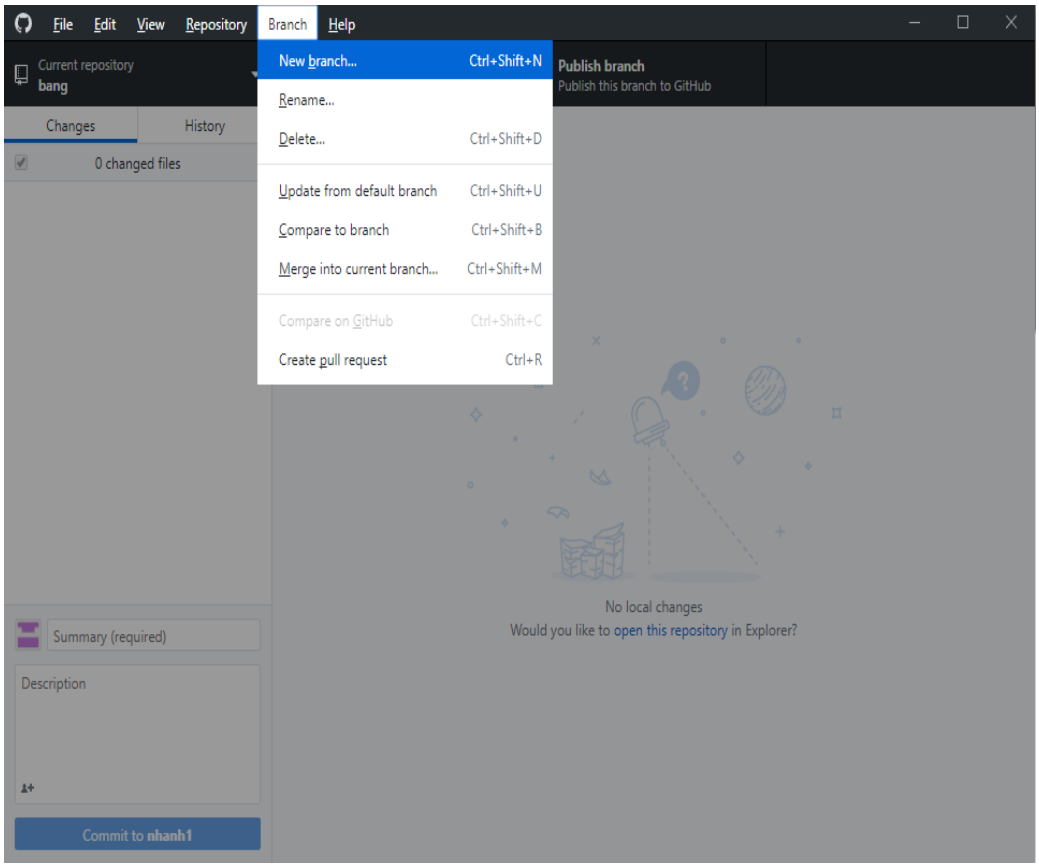
Hình 4.44 Chọn delete

Sau đó nhấn Delete để xóa



Hình 4.45 Chọn nhánh muốn xóa và delete

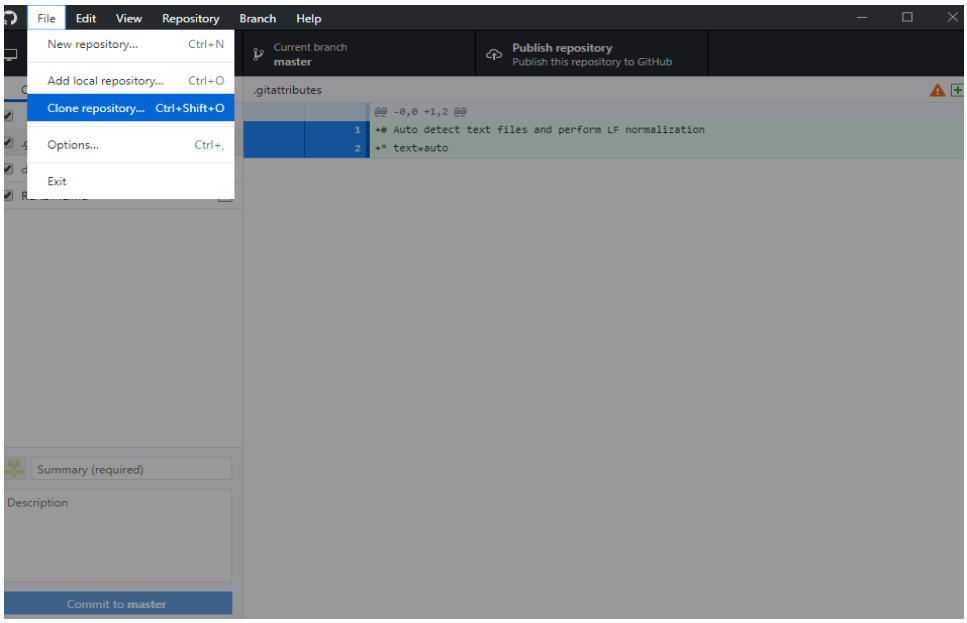
Để đổi tên nhánh ta click vào tab Branch và chọn Rename



Hình 4.46 Đổi tên nhánh

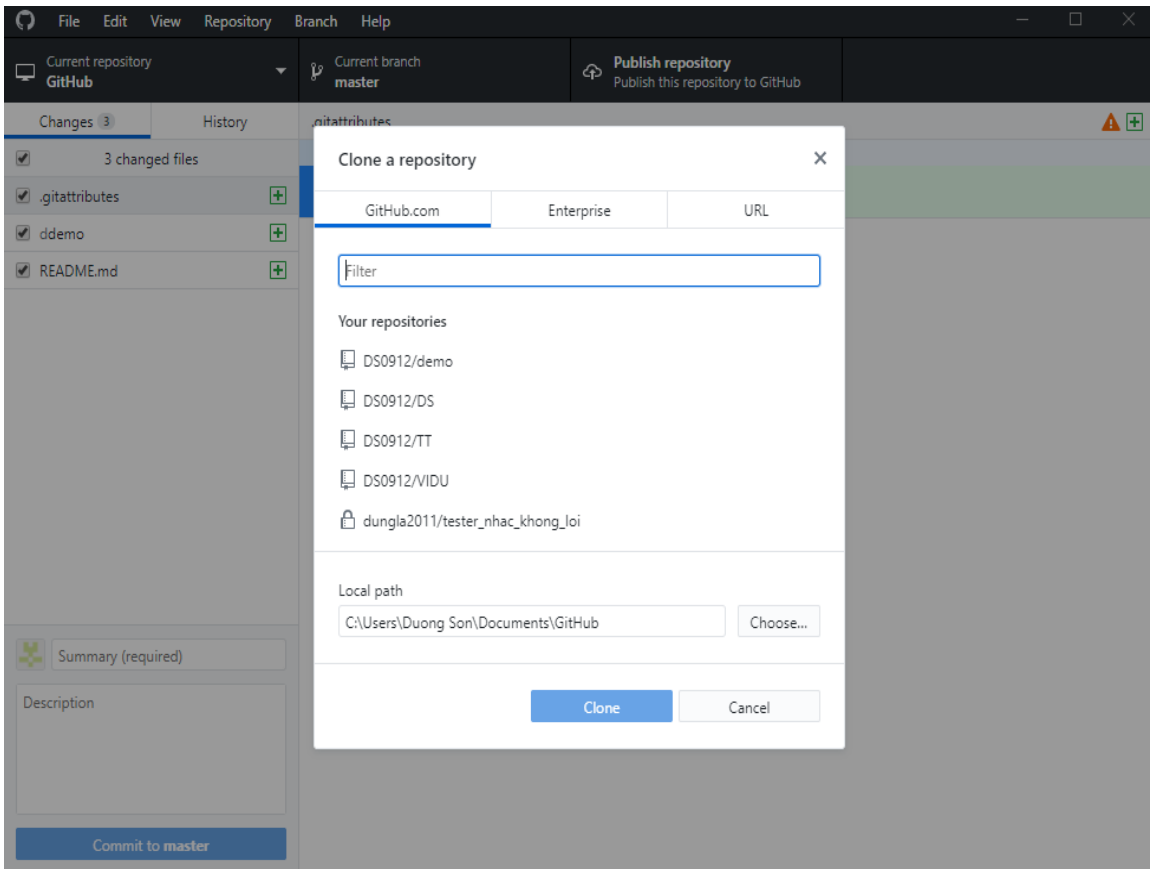
4.2.2.5 Lấy kho chứa trên Github về local bằng Git Desktop

Để lấy kho chứa từ trên Github về ta làm như sau: Chọn **File > Clone repository** hoặc nhấn **ctr+shift+O**



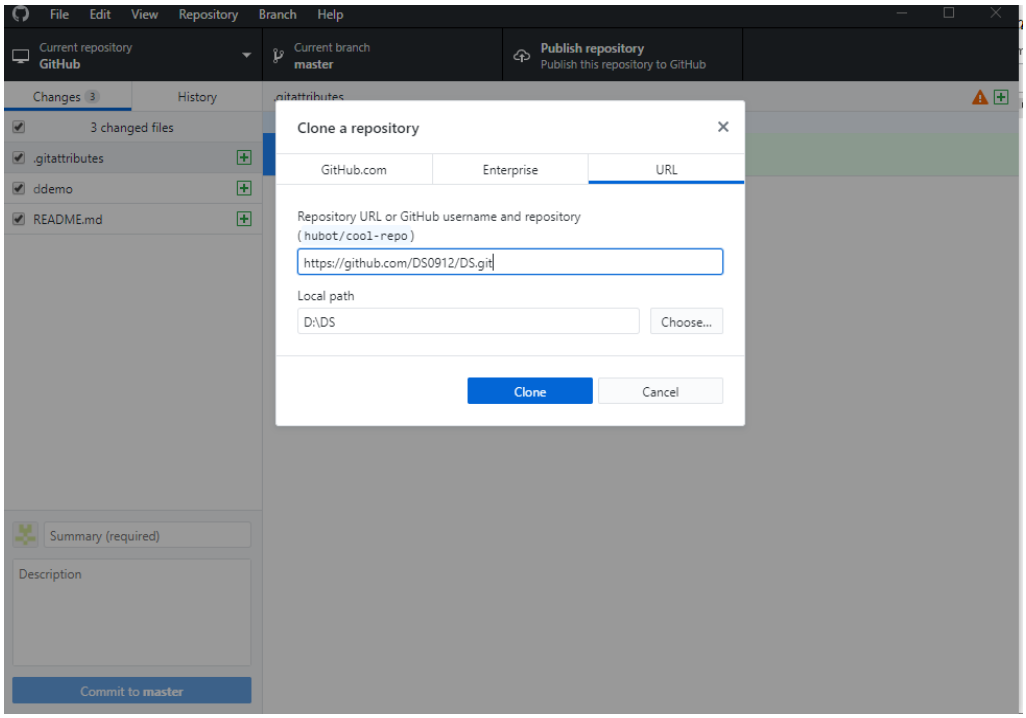
Hình 4.47 Clone kho chứa

Ta có thể chọn từ danh sách



Hình 4.48 Chọn Repository

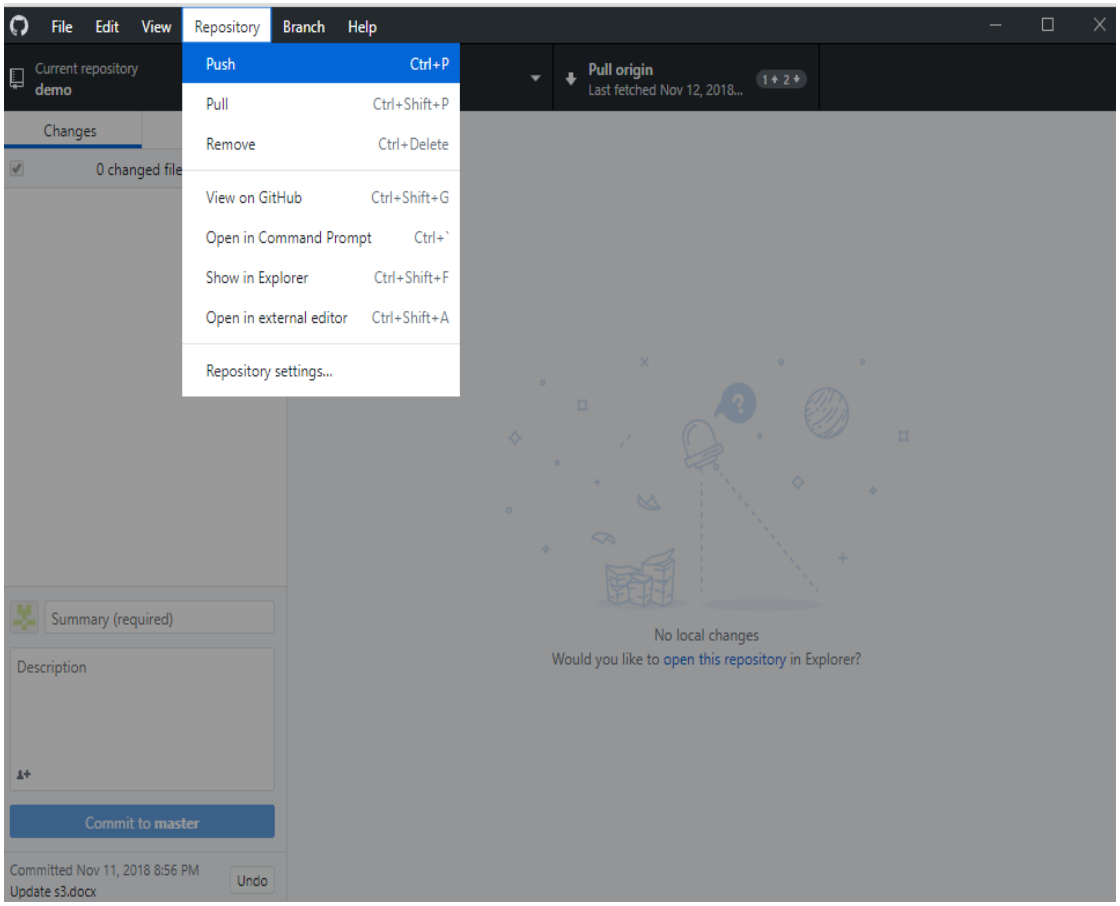
Hoặc gõ link trực tiếp



Hình 4.49 Điền link trực tiếp

4.2.2.6 Đẩy thay đổi lên Github

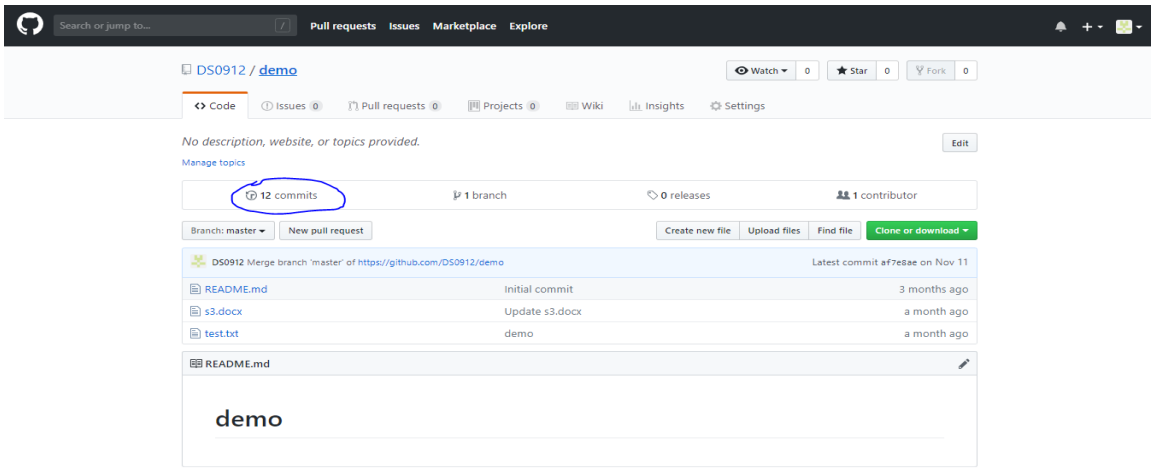
Để đẩy thay đổi lên ta chọn **Repository > Push** hoặc nhấn **ctr+p**



Hình 4.50 Đẩy thay đổi lên

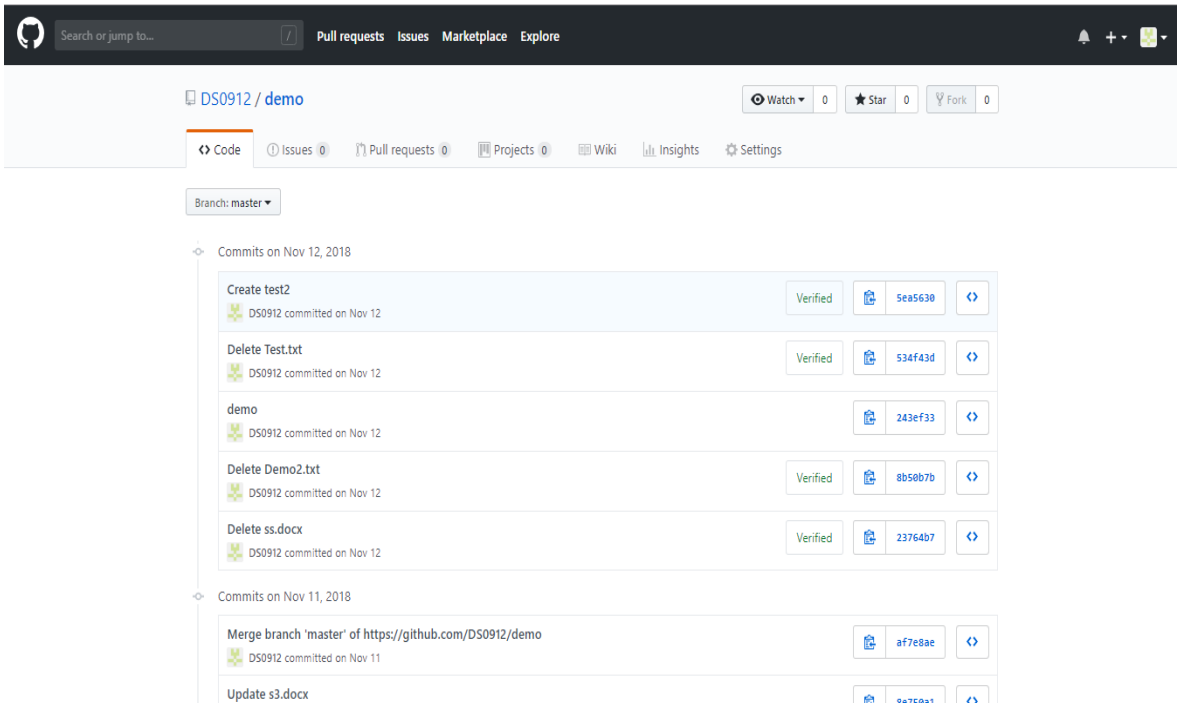
4.2.3 Quản lý phiên bản

Để xem lại những phiên bản cũ, trong repository, ta chọn commits như hình dưới



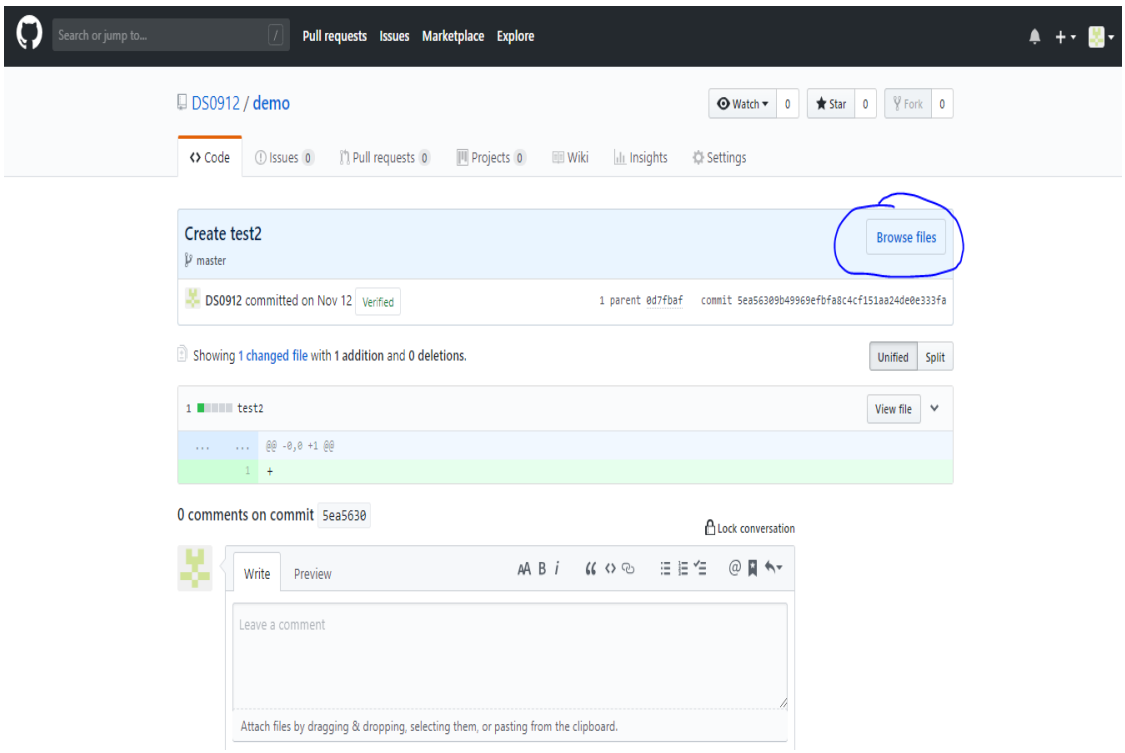
Hình 4.51 Click để xem các phiên bản cũ hơn

Tại đây ta sẽ thấy danh sách những lần cập nhật, ta click vào phần mà ta muốn xem lại



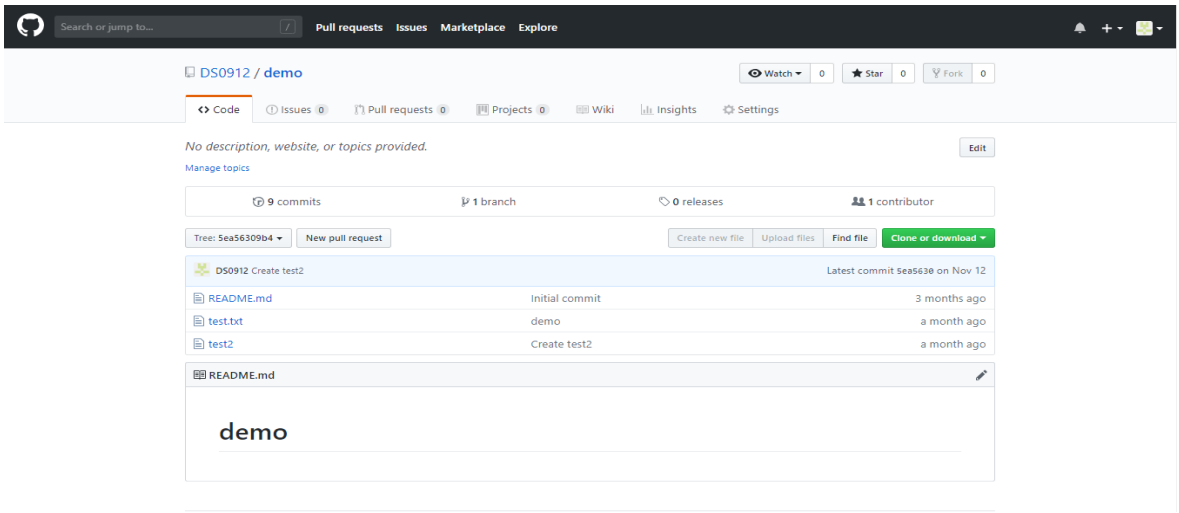
Hình 4.52 Danh sách các thay đổi

Sau đó chọn Browse files để xem lại các file



Hình 4.52 Chọn Browse files

Tại đây, ta có thể nén và tải các file về máy



Hình 4.53 Ta có thể tải file về

PHẦN III. KẾT LUẬN

1. Kết quả đạt được

Tìm hiểu tổng quan về quản lý cấu hình phần mềm như nội dung, khoản mục, nhiệm vụ của quản lý cấu hình phần mềm. Tìm hiểu công cụ Github trong việc quản lý cấu hình phần mềm, mà ở đây là quản lý phiên bản (version). So sánh Github với một vài công cụ quản lý phiên bản khác. Nắm bắt những lệnh hay dùng trên Github sử dụng dòng lệnh cũng như trên Github Desktop.

2. Hạn chế của đề tài

Github là một công cụ để quản lý phiên bản – một phần của quản lý cấu hình phần mềm, không mang tính bao quát toàn bộ nội dung của quản lý cấu hình phần mềm

Quy trình phát triển phần mềm trong thực tế được thực hiện dưới sự tham gia của nhiều người với những vai trò khác nhau và quản lý cấu hình phần mềm cũng vậy. Do điều kiện chưa được tiếp xúc nhiều với môi trường phát triển phần mềm thực tế nên không thể làm rõ hết được vai trò quyền hạn của từng người trong quá trình thực hiện quản lý.

Github còn hỗ trợ nhiều công cụ khác trong quá trình phát triển phần mềm và kết hợp với việc quản lý cấu hình phần mềm để tạo ra quy trình phát triển phần mềm chuyên nghiệp như: Quản lý nhóm, quản lý tiến độ công việc,..

3. Hướng phát triển

Ứng dụng Github để đưa vào quản lý phiên bản phần mềm cho những môn học khác cũng như những dự án mà mình tham gia.

Tìm hiểu những công cụ đi kèm, hỗ trợ khác trong quá trình phát triển phần mềm để góp phần tạo ra sản phẩm đạt chất lượng cao nhất

Tìm hiểu, thực tập tại các môi trường phát triển phần mềm thực tế để làm rõ quy trình, vai trò, nhiệm vụ của từng thành viên trong dự án

TÀI LIỆU THAM KHẢO

[1] https://www.wikipedia.org/

[2] https://docs.openstack.org/developer/glance/#user-guide

[3] https://docs.openstack.org/cli-reference/glance.html

[4] http://docs.openstack.org/user-guide/common/cli_manage_images.html

[5] https://docs.openstack.org/developer/python-openstackclient/

[6] http://docs.openstack.org/user-guide/cli_manage_images_curl.html

[7] https://developer.openstack.org/api-ref/image/index.html

[8] https://backlog.com/git-tutorial/vn/reference/