TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN KIẾN TẬP CÔNG NGHIỆP**

**TÌM HIỂU GITHUB/ AZURE/ JENKINS**

*Người hướng dẫn*: **TS TRẦN THANH PHƯỚC**

*Người thực hiện*: **NGUYỄN THIỆN HƯNG – 51603131**

**TRẦN BẢO DUY – 51603080**

Lớp : **16050301**

**16050303**

Khoá : **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN KIẾN TẬP CÔNG NGHIỆP**

**TÌM HIỂU GITHUB/ AZURE/ JENKINS**

*Người hướng dẫn*: **TS TRẦN THANH PHƯỚC**

*Người thực hiện*: **NGUYỄN THIỆN HƯNG – 51603131**

**TRẦN BẢO DUY – 51603080**

Lớp : **16050301**

**16050303**

Khoá : **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những “lời cảm ơn” đã có.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Trần Thanh Phước;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 14 tháng 6 năm 2019*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Thiện Hưng*

*Trần Bảo Duy*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trong thời đại công nghệ hiện nay , nhu cầu của người dùng ngày càng tăng cao và vì thế nên các ứng dụng , công nghệ luôn được phát triển và triển khai nhanh chóng để đáp ứng và thỏa mãn nhu cầu của người dùng . Tuy nhiên , các quy trình , phương pháp xưa cũ đã góp phần làm giảm tiến độ và hiệu quả trong việc pháp triển phần mềm từ đó làm giảm hiệu suất triển khai phần mềm mới đến người dùng. Phương pháp tích hợp liên tục ra đời để thay đổi việc đó , nó giúp các nhà pháp triển có thể xây dựng , thử nghiệm và triển khai phần mềm một cách liên tục và hiệu quả để đáp ứng nhu cầu của thị trường . Phương pháp tích hợp liên tục thường được triển khai dựa trên các công cụ như Github , Jenkins và một số hệ thống điện toán đám mây như Azure . Kết quả sau cùng của việc tích hợp liên tục sẽ là một quá trình phát triển phần mềm liên tục và xuyên suốt giữa các nhà phát triển cũng như là một môi trường làm việc mang tính hiệu quả .

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc11237177)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc11237178)

[TÓM TẮT iv](#_Toc11237179)

[MỤC LỤC 1](#_Toc11237180)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc11237181)

[CHƯƠNG 1 – GITHUB 5](#_Toc11237182)

[1.1 Tổng quan về GitHub: 5](#_Toc11237183)

[1.1.1. Khái niệm về GitHub: 5](#_Toc11237184)

[1.1.2 Những khái niệm liên quan: 6](#_Toc11237185)

[1.1.1.1 Tiểu mục cấp 3 11](#_Toc11237186)

[1.1.1.2 Tiểu mục cấp 3 tiếp theo. 12](#_Toc11237187)

[1.1.2 Tiểu mục cấp 2 tiếp theo 12](#_Toc11237188)

[1.2 Nội dung của chương này 12](#_Toc11237189)

[2.1 TÍCH HỢP LIÊN TỤC 13](#_Toc11237190)

[2.1.1 Tích hợp liên tục là gì? 13](#_Toc11237191)

[2.1.2 Đặc điểm của việc tích hợp liên tục: 15](#_Toc11237192)

[2.1.3 Lợi ích và khó khăn của tích hợp liên tục, 16](#_Toc11237193)

[2.2 JENKINS 17](#_Toc11237194)

[2.2.1 Jenkins là gì ? 17](#_Toc11237195)

[2.2.2 CI với Jenkins 19](#_Toc11237196)

[2.2.3 Lợi ích và khó khăn của Jenkins 21](#_Toc11237197)

[3.1 Microsoft Azure là gì ? 22](#_Toc11237198)

[3.2 Các dịch vụ, mô hình dịch vụ đám mây và lợi ích của Microsoft Azure: 23](#_Toc11237199)

[3.2.1 Các dịch vụ của Microsoft Azure: 23](#_Toc11237200)

[3.3 Trích dẫn 25](#_Toc11237201)

[25](#_Toc11237202)

[3.2.2 Mô hình dịch vụ đám mây của Microsoft Azure: 26](#_Toc11237203)

[28](#_Toc11237204)

[3.2.3 Lợi ích của Microsoft Azure: 29](#_Toc11237205)

[3.3 Máy ảo trên Microsoft Azure: 31](#_Toc11237206)

[3.4 Lưu trữ trên Microsoft Azure 32](#_Toc11237207)

[3.5 Virtual network trong Microsoft Azure: 37](#_Toc11237208)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

*f Tần số của dòng điện và điện áp (Hz)*

*p Mật độ điện tích khối (C/m3)*

**CÁC CHỮ VIẾT TẮT**

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

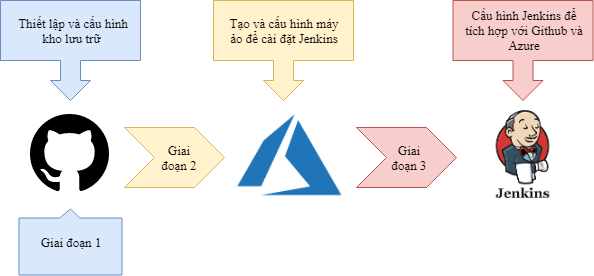
**CHƯƠNG 1 – TỔNG QUAN KHÁI QUÁT VỀ MÔ HÌNH , PHƯƠNG PHÁP TÍCH HỢP LIÊN TỤC VỚI GITHUB / JENKINS / AZURE**

Trong phương pháp tích hợp liên tục , các thành phần quan trọng không chỉ là các nhà phát triển , mà nó còn bao gồm các công cụ hỗ trợ cho nhà phát triển . Việc lựa chọn các công cụ hợp lý và phù hợp để hỗ trợ cho quá trình và môi trường của các nhà phát triển là một trong những việc quan trọng khi bắt đầu một mô hình tích hợp liên tục . Có ba thành phần công cụ chính trong mô hình tích hợp liên tục , đó là :

* **Công cụ quản lý phiên bản/nguồn ( version/source control ) :** Công cụ quản lý phiên bản giúp cho các nhà phát triển có thể kiểm soát được sự thay đổi của cấu trúc chương trình hay các đoạn mã . Nó còn giúp tạo các nhánh ( branch ) khác nhau trong một dự án để các thành viên có thể làm việc độc lập và sau đó công cụ này hỗ trợ để gộp các nhánh khác nhau lại để thành một bản thống nhất. Trong các công cụ quản lý phiên bản thì Github là công cụ được nhiều nhà phát triển tin tưởng và sử dụng để quản lý phiên bản dự án của họ . Github là công ty phi lợi nhuận cung cấp các kho lưu trữ Git dựa trên Cloud thông qua nền tảng web . Github có đầy đủ tính năng của việc quản lý phiên bản như tạo kho lưu trữ , tạo nhánh , push , commit , … . Vì được triển khai trên nền tảng web nên Github có thể dễ dàng truy cập ở bất cứ nơi nào có trình duyệt và mạng internet. Github là hoàn toàn miễn phí nên Github có một cộng đồng người sử dụng đông đảo để có thể hỗ trợ , phản hồi các vấn đề liên quan tới Github hay quản lý phiên bản .
* **Công cụ xây dựng/thử nghiệm/triển khai tự động :** Mặc dù gọi là công cụ nhưng nó thường là một máy chủ đóng vai trò xây dựng/thử nghiệm/triển khai một cách tự động mà không cần nhiều sự tương tác của nhà phát triển . Công cụ nổi tiếng cũng như lâu đời nhất cho việc này là Jenkins. Jenkins luôn được nhắc tới trong phương pháp tích hợp liên tục vì các tính năng của nó . Jenkins cho phép nhà phát triển tạo ra các công việc ( job ) cho các dự án và cấu hình các công việc đó tùy theo nhu cầu của nhà phát triển . Jenkins còn nổi bật hơn các công cụ khác vì Jenkins có các plugin để có thể hỗ trợ nó trong nhiều việc khác nhau , giúp nó đa dạng hơn . Vì Jenkins là một công cụ mã nguồn mở nên nó có một cộng đồng người sử dụng lớn , việc này góp phần giúp việc mở rộng các plugin dễ dàng hơn.
* **Nền tảng để triển khai :** Mỗi dự án hay chương trình sẽ có một nền tảng cụ thể để có thể triển khai chương trình lên nền tảng để người dùng đầu cuối có thể tương tác . Tuy vậy , trong môi trường phần mềm , nền tảng là dùng để tạo nên một môi trường thử nghiệm để có thể thử nghiệm triển khai các chương trình trước khi được đưa lên nền tảng chính thức . Microsoft Azure là một trong nền tảng điện toán đám mây nổi tiếng nhất hiện nay . Nó cung cấp rất nhiều dịch vụ hữu ích cho nhiều nhu cầu hiện nay của các nhà phát triển . Một trong những dịch vụ mà Azure nổi bật nhất đó là việc triển khai và quản lý máy ảo để phục vụ cho việc thử nghiệm việc triển khai của chương trình trước khi được giao cho khách hàng hoặc là tích hợp vào môi trường làm việc chính thức .

Để tích hợp cả ba Github/Jenkins/Azure , chúng em chia các thành ba 3 giai đoạn . Mỗi giai đoạn sẽ thiết lập và cấu hình từng công cụ để có thể dần dần tích hợp cả ba . Các giai đoạn đó là :

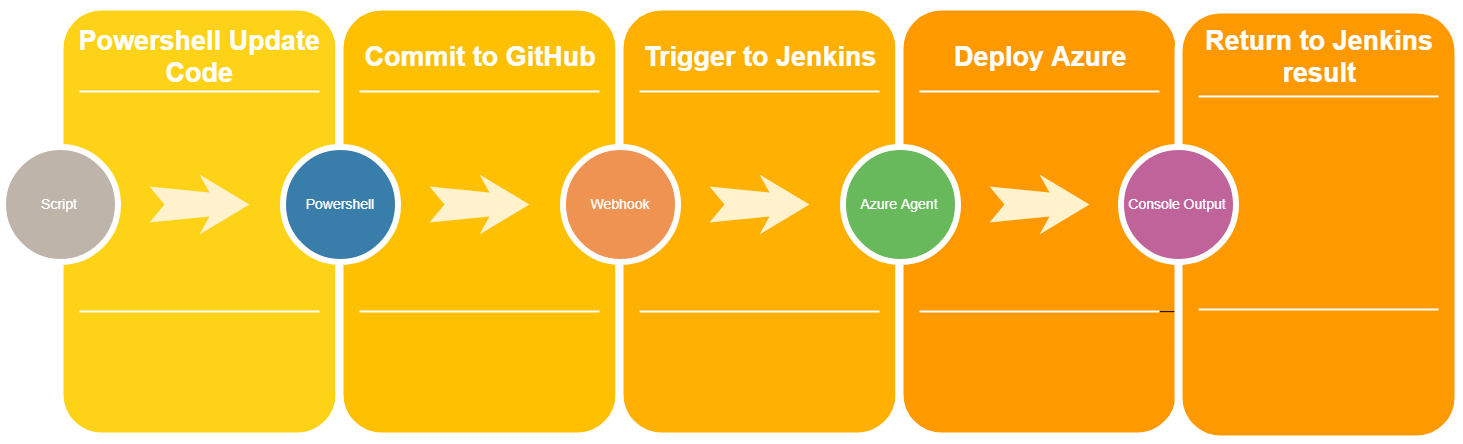
* **Giai đoạn 1 – Cấu hình Github :** Giai đoạn này đặt trọng tâm vào việc tạo nên kho lưu trữ được chia sẻ chung giữa các nhà phát triển. Khi kho được tạo thành công , chúng em sẽ tiếp tục cấu hình Github webhook của kho để có thể gửi thông điệp tới Jenkins khi có hành động push lên kho .
* **Giai đoạn 2 –** **Tạo máy ảo và cài đặt Jenkins trên máy ảo Azure :** Ở giai đoạn thứ 2 sẽ được thực hiện trên nền tảng điện toán đám mây Microsoft Azure . Chúng em sẽ tạo máy ảo và sau đó sẽ tiến hành cài đặt Jenkins trên máy ảo đó để biến Jenkins thành một dịch vụ đám mây .
* **Giai đoạn 3 – Cấu hình Jenkins :** Giai đoạn cuối cùng sẽ là giai đoạn để cấu hình Jenkins , tạo công việc , cũng như thiết lập plugin Azure VM Agent để có thể tự động tạo máy ảo mỗi khi cần xây dựng đoạn mã .

****

Hình 1.1 : Ba giai đoạn tích hợp

Sau khi đã lựa chọn được các công cụ , thành phần phù hợp thì chúng em sẽ tích hợp chúng vào mô hình tích hợp liên tục để tạo thành một luồng công việc hoàn chỉnh . Luồng công việc của chúng em sẽ như sau khi tích hợp :

1. Các nhà phát triển sẽ viết các đoạn mã , chương trình ở dưới máy làm việc của mình ( các đoạn mã mà chúng em sẽ trình bày trong báo cáo này sẽ là một đoạn script được viết bằng Powershell ) .
2. Sau khi hoàn thành chỉnh sửa/viết các đoạn mã , các nhà phát triển sẽ commit và push đoạn mã của mình lên một kho được chia sẻ chung với các nhà phát triển khác .
3. Github sau khi nhận được hành động push lên kho , Github webhook sẽ gửi một thông điệp HTTPs tới Jenkins .
4. Jenkins nhận được thông điệp HTTPs từ Github webhook sẽ bắt đầu lấy các dữ liệu từ trong kho và bắt đầu xây dựng một máy ảo trên Microsoft Azure mà đã được cấu hình bằng plugin Azure VM agent của Jenkins .
5. Jenkins đưa các dữ liệu ở kho trên Github đến máy ảo và bắt đầu quá trình xây dựng và triển khai đoạn mã trên máy ảo .
6. Kết quả của quá trình xây dựng và triển khai sẽ được trả về thông qua cửa sổ Console Output của Jenkins .

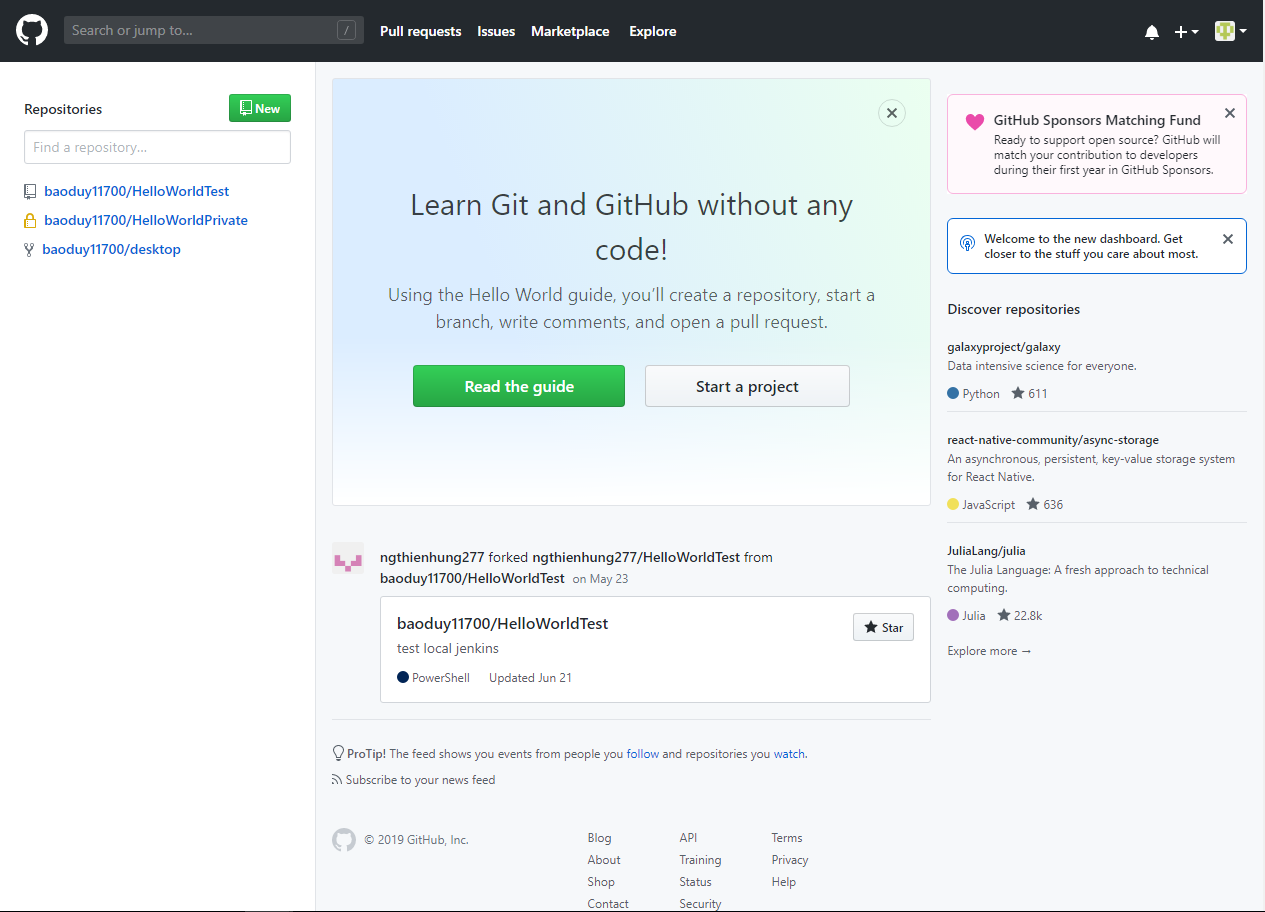


Hình 1.2 : Luồng công việc của mô hình tích hợp Github/Jenkins/Azure

**CHƯƠNG 2 – CÁC GIAI ĐOẠN TÍCH HỢP GITHUB / JENKINS / AZURE**

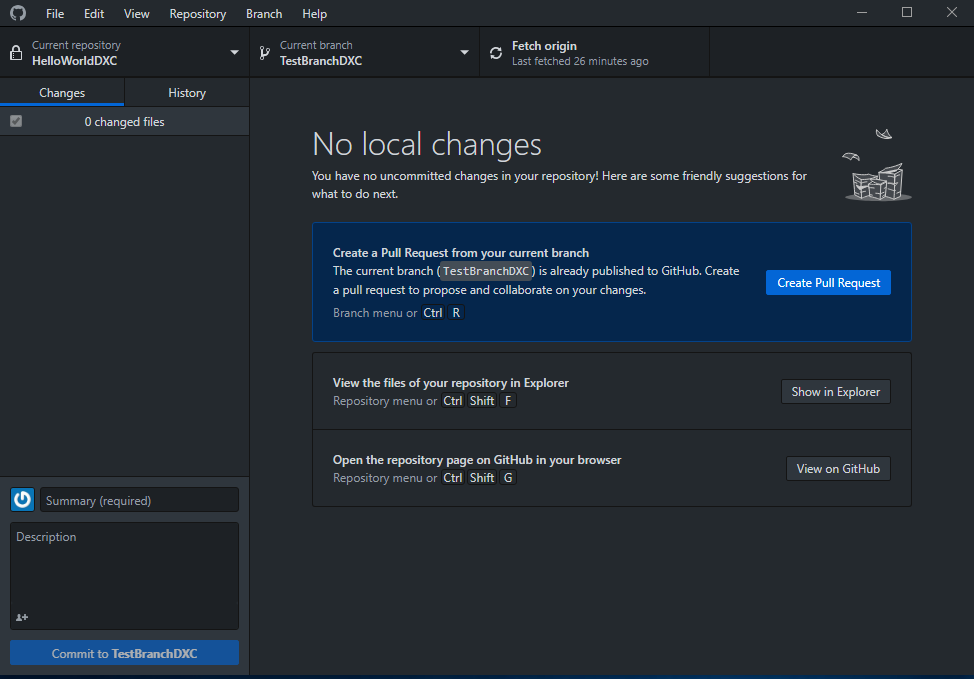
**2.1 Giai đoạn 1 - Cấu hình Github**

Với các nhà phát triển hay các nhà lập trình thì Github không còn gì xa lạ nữa . Github là một trong những dịch vụ kho lưu trữ Git nổi tiếng và thông dụng nhất . Github hoạt động phần lớn thông qua giao diện Web và nền tảng điện toán đám mây . Việc này giúp cho Github có thể dễ dàng truy cập ở bất kì mọi nơi . Với Github , giao diện Web khiến cho người dùng sẽ dễ dàng thao tác với Git hơn thay vì phải làm việc với các dòng lệnh . Thông qua giao diện Web , người dùng có thể sử dụng các tính năng mạnh mẽ của Github như fork , pull request và merge . Nó cũng là nơi mà người dùng sẽ chỉnh sửa thông tin , cài đặt của người dùng cũng như là nơi để quản lý kho lưu trữ và cấu hình kho lưu trữ theo nhu cầu của người dùng .

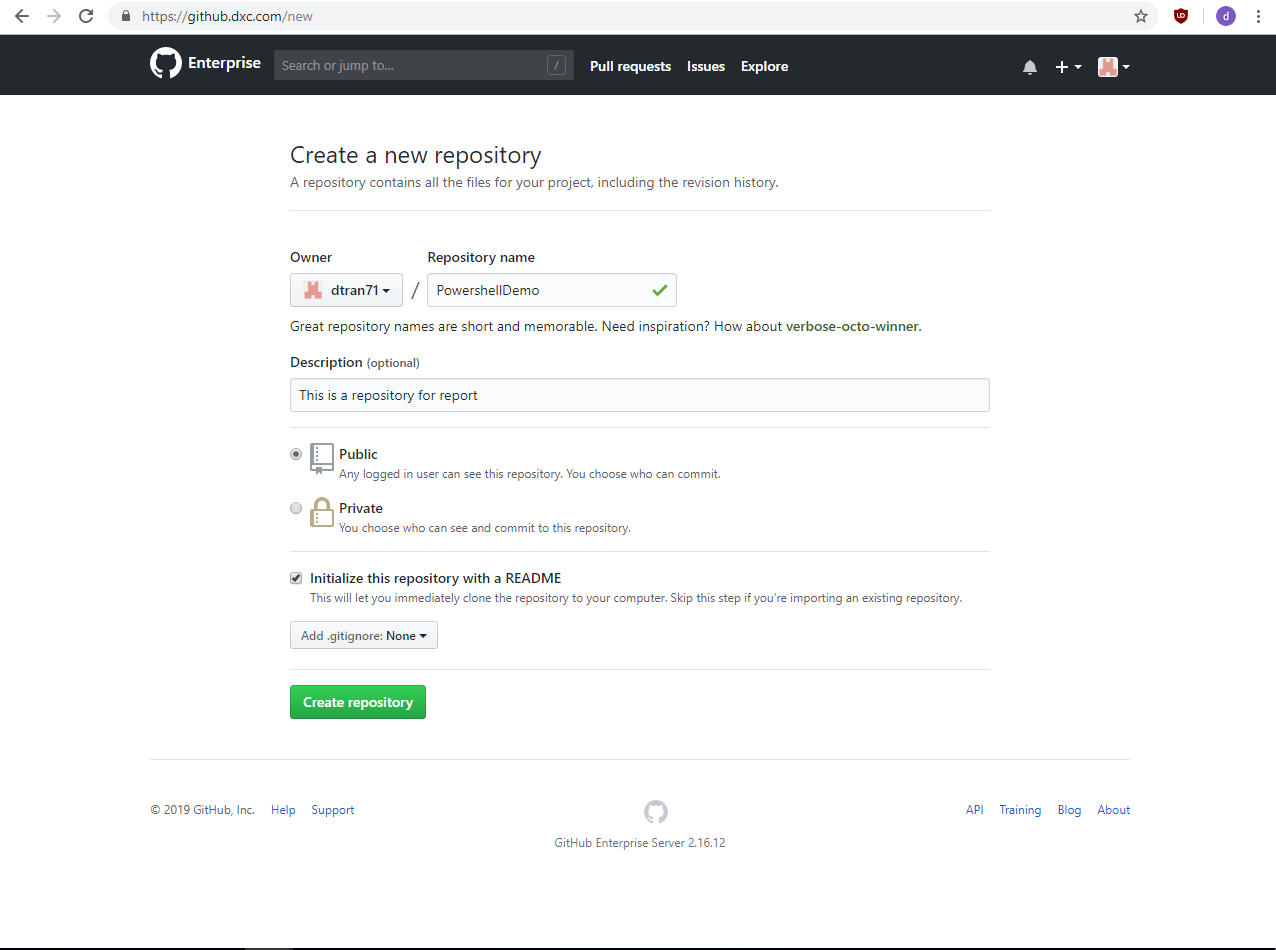


Hình 2.1 : Giao diện web của Github sau khi người dùng đăng nhập

Để làm việc trên Github thì người dùng không chỉ có Github trên giao diện Web mà Github còn hỗ trợ người dùng với một ứng dụng gọi là Github desktop . Github desktop là một ứng dụng đa nền tảng được tạo ra giúp người dùng làm việc dễ dàng hơn với Git bằng cách sử dụng giao diện thay vì phải sử dụng các lệnh dòng lệnh Git , tuy vậy , Github desktop vẫn hỗ trợ các dòng lệnh Git thông qua Git bash nếu người dùng vẫn muốn sử dụng dòng lệnh . Nếu giao diện Web của Github là dùng để quản lý kho lưu trữ ở xa ( Remote Repository ) thì Github sẽ giúp người dùng quản lý kho lưu trữ nội bộ ( Local Repository ) trên máy tính làm việc của người dùng . Nó đơn giản hóa các lệnh cơ bản của Git như commit , push , fetch chỉ bằng các bước thao tác . Không chỉ vậy , Github Desktop còn hỗ trợ người dùng với nhiều tính năng nâng cao khác như so sánh giữa các nhánh với nhau , tạo kho lưu trữ , nhân bản kho lưu trữ , merge các nhánh vào nhánh master , chuyển đổi giữa các kho lưu trữ khác nhau ,… Những điều này đã khiến cho Github Desktop trở thành một công cụ không thể thiếu cho những nhà phát triển hay bất kì ai trong lĩnh vực công nghệ thông tin .



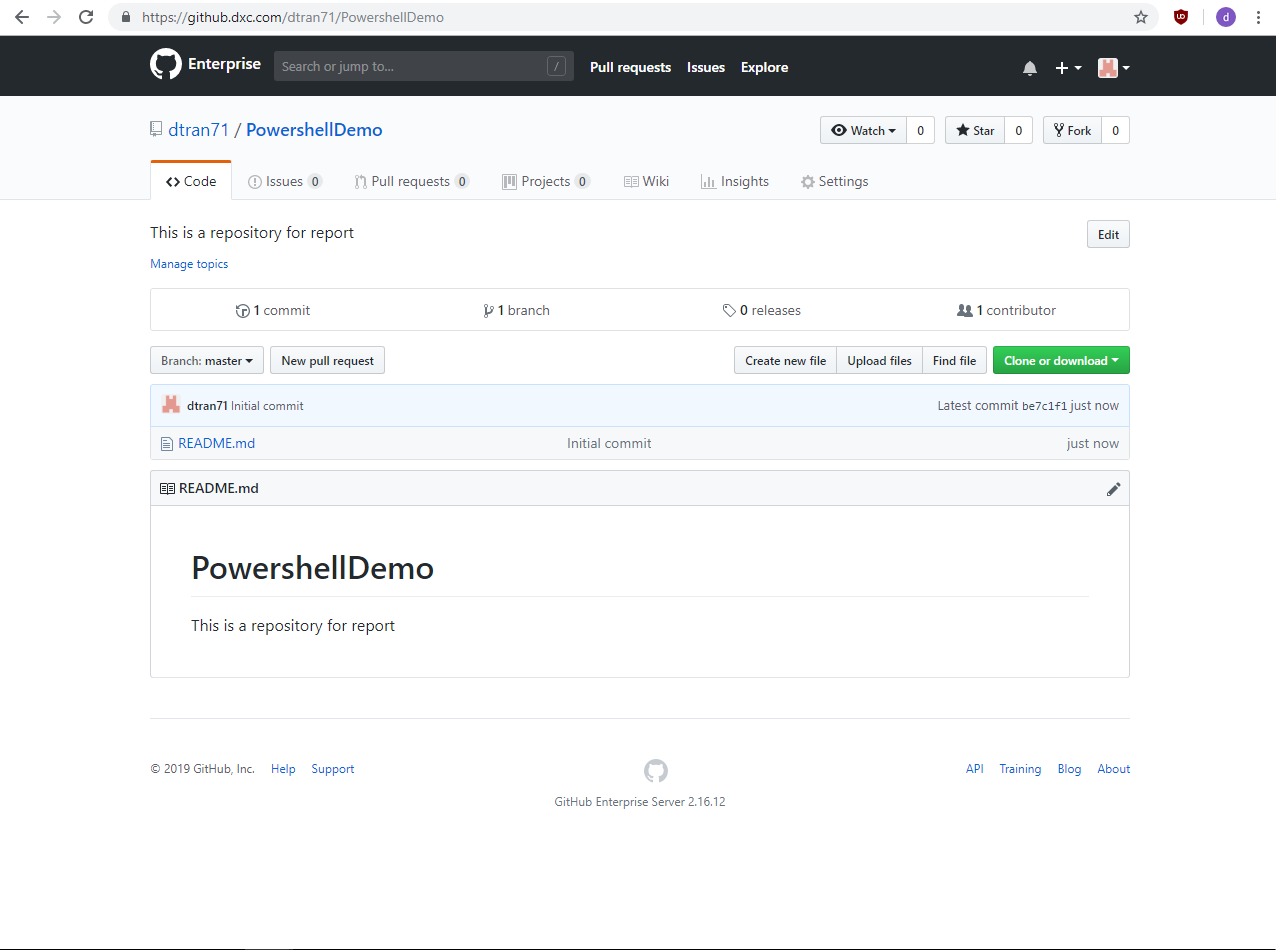
Hình 2.2 Giao diện Github Desktop

Để làm việc với Github , chúng em sẽ sử dụng tài khoản Github thuộc loại Enterprise được cấp bởi công ty DXC technology . Với loại tài khoản này , chúng em sẽ đăng nhập vào Github thông qua tên miền [https://github.dxc.com](https://github.dxc.com/) thay vì <https://github.com> . Trong dự án này , chúng em sẽ tạo một kho lưu trữ riêng với tên gọi “PowershellDemo” trên tài khoản dtran71.

Hình 2.3 Tạo kho lưu trữ mới thông qua giao diện Web

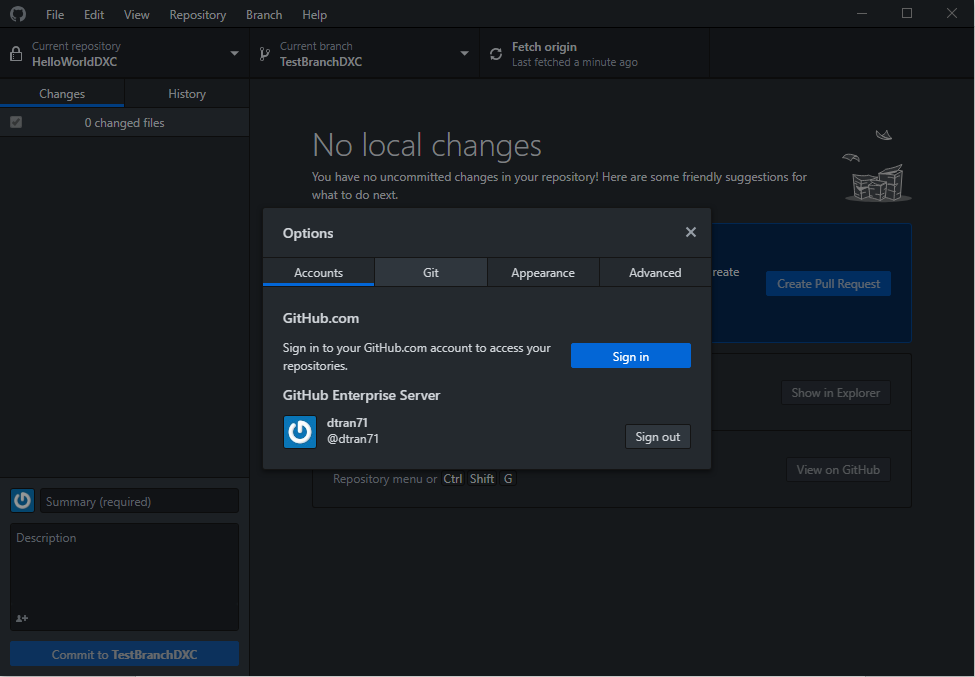
Trên giao diện của Github , việc tạo một kho lưu trữ mới sẽ có một số mục lưu ý như sau :

* **Owner :** Tên người tạo ra kho lưu trữ .
* **Repository name :** Tên kho lưu trữ .
* **Description :** Mô tả của kho lưu trữ . Điều này là không bắt buộc nhưng nếu người dùng cón nhiều kho lưu trữ thì nên mô tả các kho để phân biệt .
* **Public và Private :** Loại kho lưu trữ mà người dùng sẽ tạo . Điểm khác biệt lớn nhất đó là chủ kho lưu trữ sẽ có quyền cho ai thấy được kho và cho ai commit lên kho .
* **Initialize this repository with a README :** khởi tạo kho lưu trữ với một file README sẵn trong kho .
* **Add .gitignore** : .gitignore là một dạng file do Git quản lý có nhiệm vụ bỏ qua , lờ đi các file đã được quy định trong gitignore khi người dùng commit hay push lên kho .



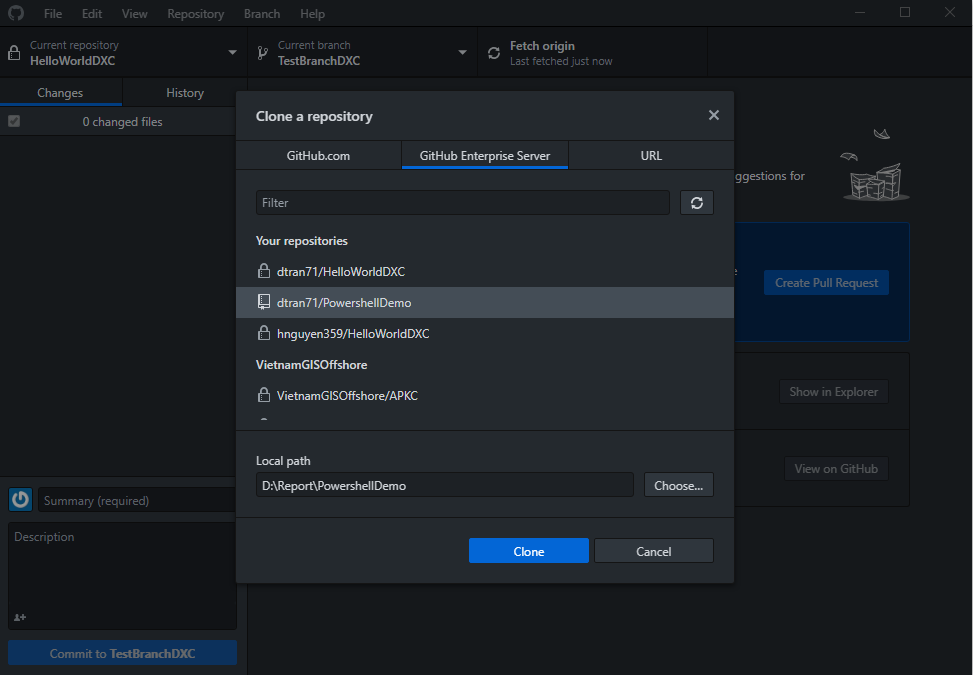
Hình 2.3 Kho lưu trữ được tạo mới

Kho lưu trữ từ xa đã được tạo mới , tuy nhiên , để thực sự làm việc với Github thì cần phải tạo một kho lưu trữ cục bộ ở trên máy tính làm việc. Việc tạo một kho lưu trữ cục bộ từ một kho lưu trữ từ xa được gọi là nhân bản kho lưu trữ ( Clone repository ). Thông thường , người dùng sẽ phải sử dụng câu lệnh trên Git bash để nhân bản kho nhưng với Github desktop việc này có thể làm chỉ với vài thao tác đơn giản. Để bắt đầu sử dụng Github Desktop , thì người dùng trước tiên phải đăng nhập thông qua tài khoản mình sở hữu . Người dùng đăng nhập bằng cách chọn vào File -> Option hoặc tổ hợp phím “Ctrl + ,” . Ở mục Account , chọn loại tài khoản cần đăng nhập , vì chúng em sử dụng tài khoản Enterprise nên sẽ đăng nhập ở mục “Github Enterprise Server” .



Hình 2.4 đăng nhập thành công vào Github Desktop

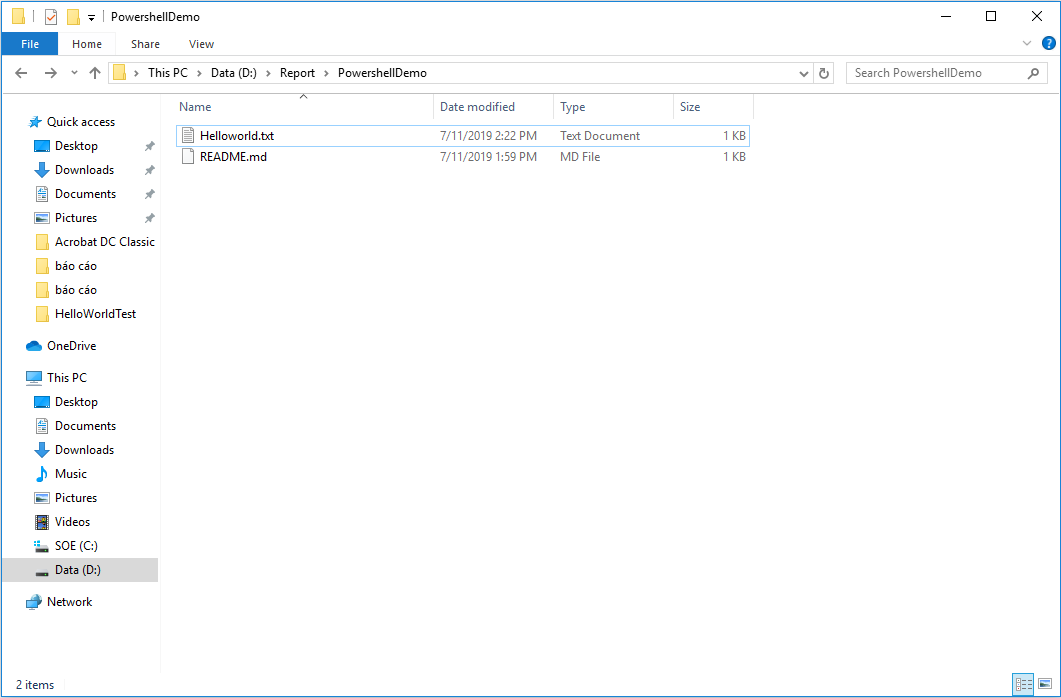
Sau khi đăng nhập thì người dùng đã có thể toàn quyền sử dụng Github Desktop để làm việc với kho lưu trữ cục bộ. Để bắt đầu nhân bản kho lưu trữ thì người dùng vào mục File -> Clone Repository hoặc tổ hợp phím “Ctrl + Shift + O” . Như đăng nhập , khi nhân bản một kho lưu trữ người dùng có thể chọn nhân bản kho từ loại tài khoản nào . Vì chúng em sử dụng tài khoản Enterprise nên sẽ nhân bản kho ở mục “Github Enterprise Server” . Github Desktop không chỉ hỗ trợ nhân bản kho từ tài khoản cá nhân , mà còn có thể nhân bản kho của các người dùng khác thông qua mục URL với cú pháp tên-người-dùng/tên-kho .



Hình 2.5 GUI để nhân bản kho lưu trữ

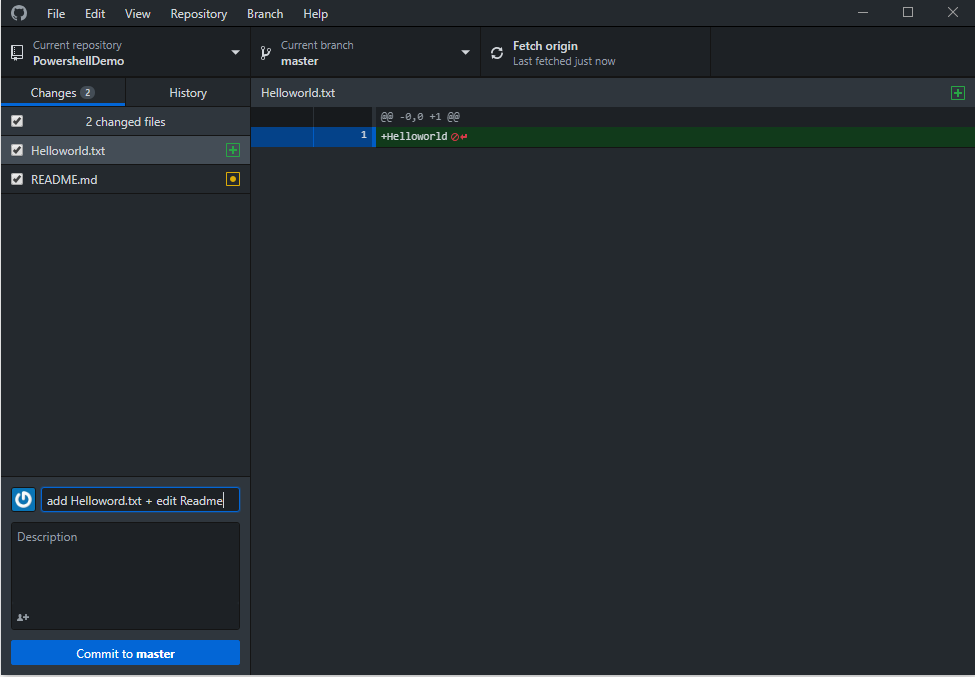
Trong đó , Local path là nơi mà Github Desktop sẽ nhân bản kho xuống dưới thư mục lưu trữ dưới máy tính làm việc. Tùy vào người dùng , mỗi máy tính nhân bản kho lưu trữ có thể sẽ có Local path khác nhau. Trên máy hiện tại em đặt Local path cho kho lưu trữ cục bộ em ở đường dẫn “D:\Report\PowershellDemo” . Sau khi chọn đường dẫn và nhấn Clone , tất cả những gì trong kho lưu trữ từ xa sẽ được Github Desktop đưa xuống kho lưu trữ cục bộ .

Với kho lưu trữ cục bộ được tạo ở máy tính làm việc , người dùng có thể sử dụng Github Desktop để làm việc với kho lưu trữ cục bộ và commit lên kho lưu trữ từ xa. Ở kho lưu trữ cục bộ hiện tại sẽ chỉ có file Readme đã được tạo trước từ lúc khởi tạo kho. Để thử nghiệm việc commit và push lên kho lưu trữ từ xa từ kho lưu trữ cục bộ , em sẽ tạo ra 1 file text tên Helloworld.txt đặt ở trong kho lưu trữ cục bộ và commit nó lên kho lưu trữ từ xa thông qua Github Desktop .



Hình 2.6 Helloworld.txt được tạo ra ở kho lưu trữ cục bộ

Sau khi thêm file Helloworld.txt , em sẽ sử dụng Github Desktop để commit và push lên kho lưu trữ từ xa . Khi có bất kỳ thay đổi gì ở kho lưu trữ cục bộ , Github Desktop sẽ tự động nhận thấy những thay đổi này và đánh dấu chúng để chúng sẵn sàng có thể được commit lên kho lưu trữ từ xa. Ngoài việc thêm file Helloworld.txt , em còn thay đổi nội dung file README để cho thấy sự khác biệt giữa thêm mới một file và chỉnh sửa một file khi chúng được hiện lên trên Github Desktop.

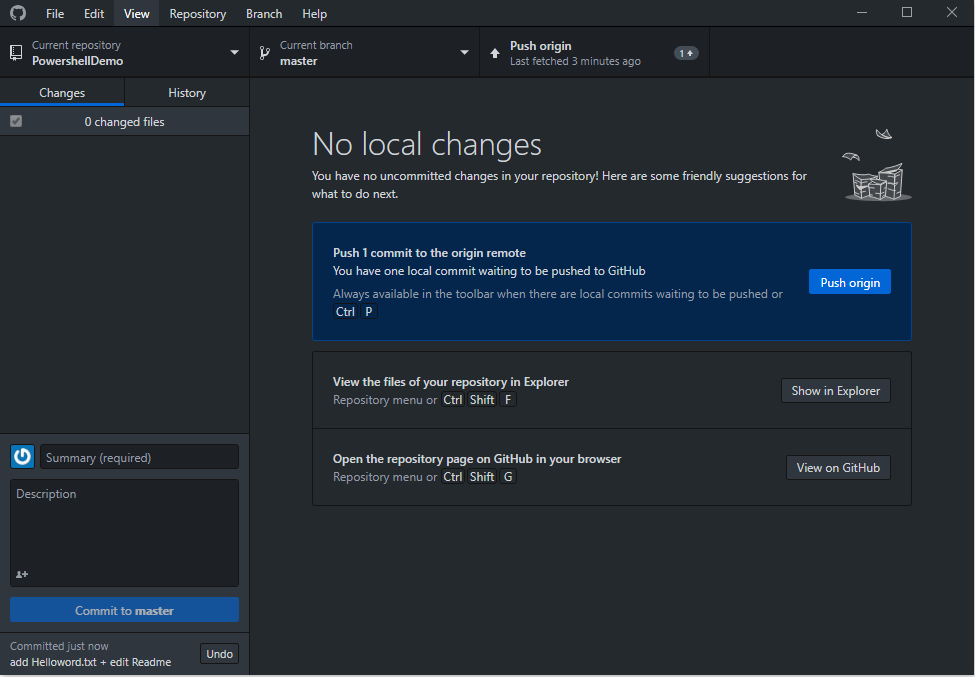


Hình 2.7 Github Desktop hiện lên những thay đổi ở kho lưu trữ cục bộ

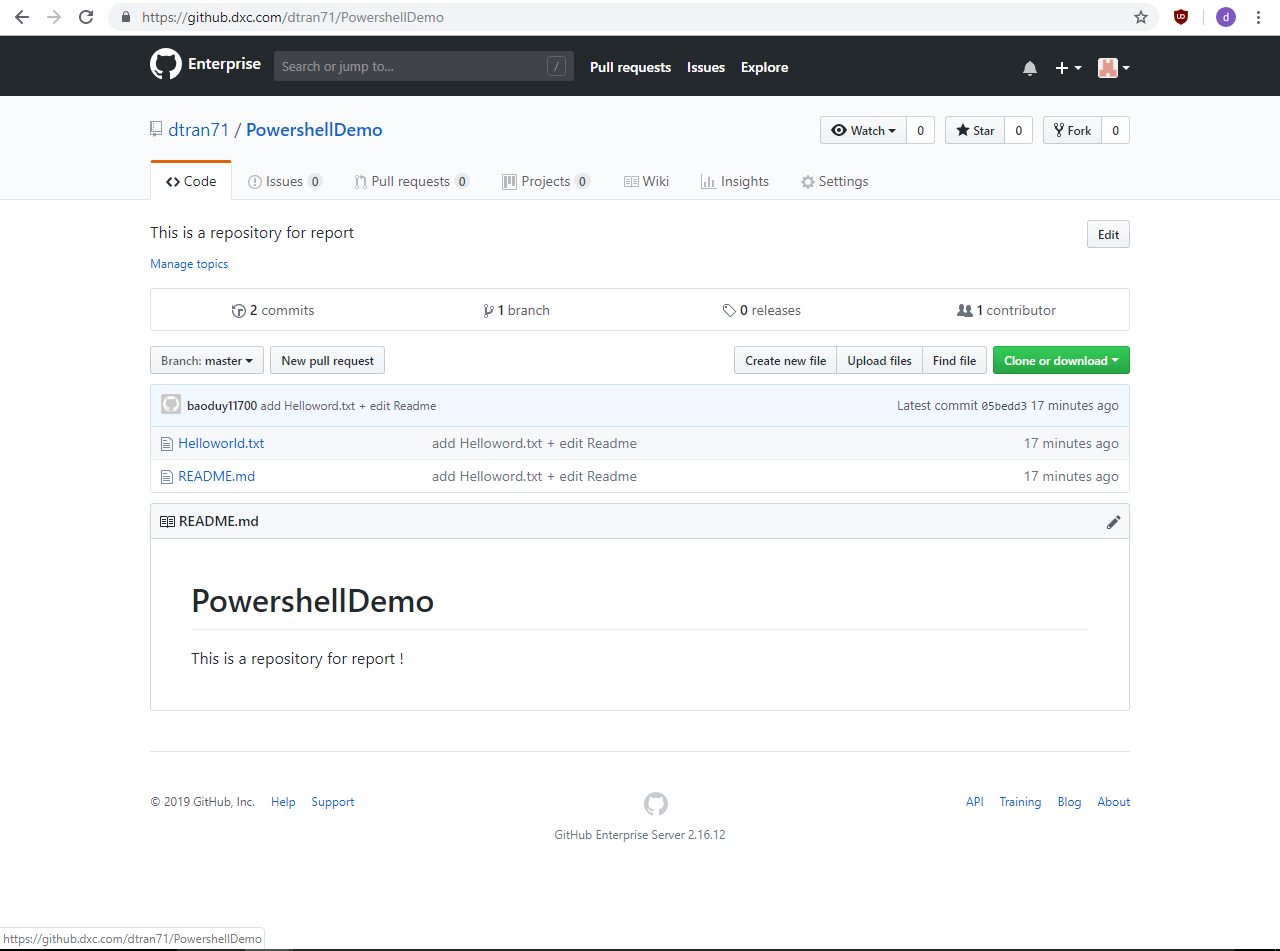
Trên hình , Github Desktop thể hiện người dùng các thông tin trước khi người dùng commit trên kho lưu trữ từ xa :

* **Current Repository :** Đây là kho lưu trữ mà người dùng đang làm việc . Nếu người dùng có nhiều kho thì có thể chuyển đổi kho tùy ý .
* **Current Branch :** Nhánh mà hiện tại người dùng đang làm việc . Khi commit các file thì sẽ được push lên các nhánh này , nhánh hiện tại em đang làm việc là nhánh chính của Kho lưu trữ ( nhánh Master ) . Nếu như có nhiều nhánh trong một kho thì người dùng có thể thay đổi nhánh tùy ý .
* **Change :** Mục này sẽ cho người dùng thấy họ đã thay đổi gì trong kho lưu trữ cục bộ . Nếu như họ đã chỉnh sửa một file thì nó sẽ được đánh dấu là vòng tròn cam , thêm một file thì sẽ là dấu thập màu xanh lá và khi xóa sẽ là dấu trừ màu đỏ. Khi trỏ vào từng file , các chỉnh sửa/xóa/sửa sẽ được hiện lên ở bảng bên cạnh .
* **History :** Sẽ hiện lên lịch sử các lần commit của người dùng lên kho lưu trữ từ xa.
* **Summary :** Tóm tắt những gì người đã làm trong lần commit này . Mục này là bắt buộc phải có khi commit lên kho lưu trữ nội bộ .
* **Description :** Người dùng có thể thêm một số chi tiết , thông tin về lần commit này . Mục này là không bắt buộc phải viết khi commit .
* **Commit to master :** Sau khi xem xét các thay đổi , nhấn vào mục commit to master để đưa các chỉnh sửa người dùng đã làm ở thư mục làm việc đến kho lưu trữ cục bộ .

Sau khi Commit to master , em chỉ đưa các thay đổi lên kho lưu trữ cục bộ chứ các thay đổi vẫn chưa được đưa lên kho lưu trữ từ xa . Để đưa lên kho lưu trữ từ xa , chọn vào mục “Push origin” , Push origin sẽ đẩy các thay đổi ở kho lưu trữ cục bộ lên trên kho lưu trữ từ xa. .

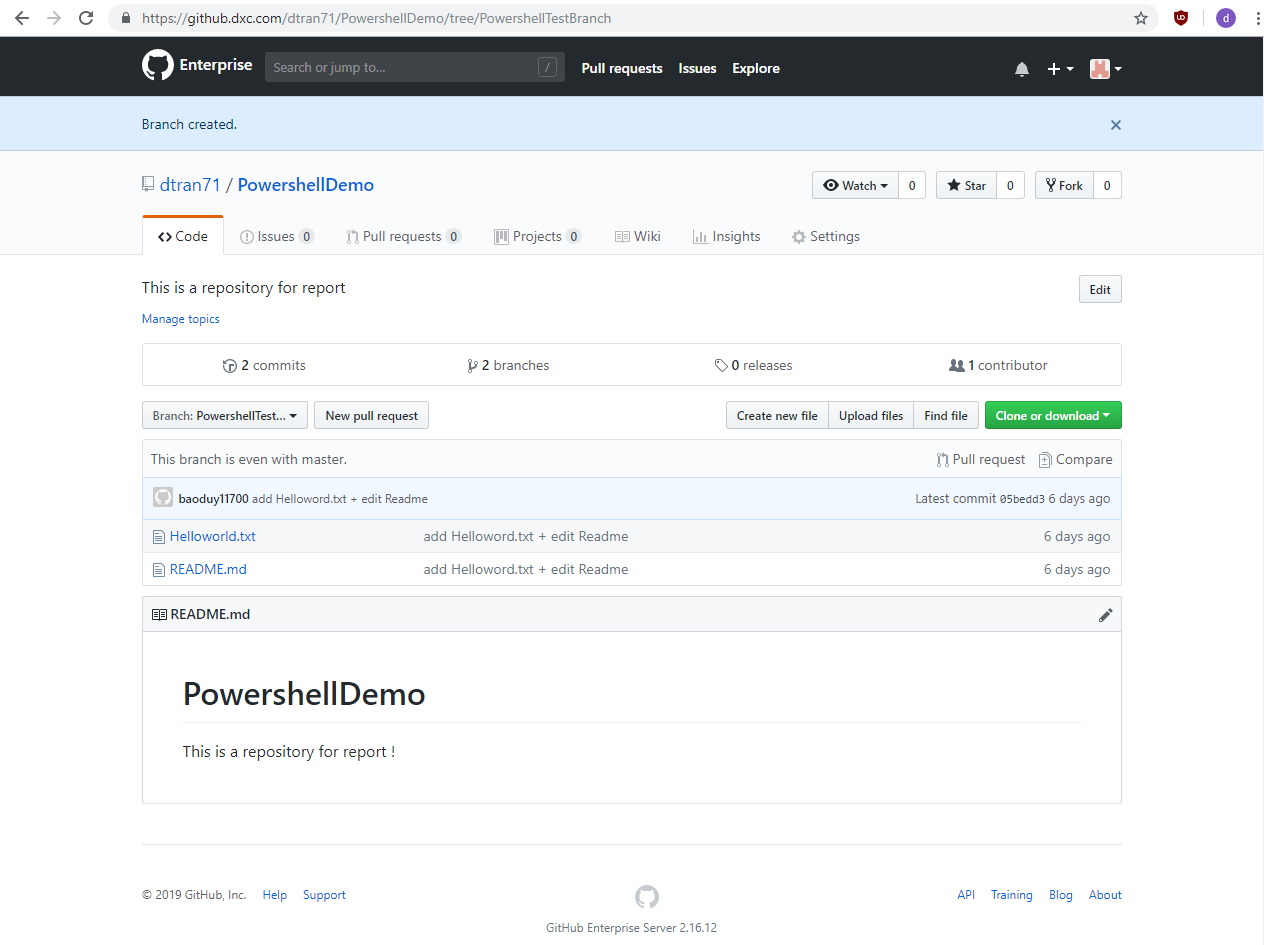


Hình 2.8 Giao diện sau khi commit những thay đổi .



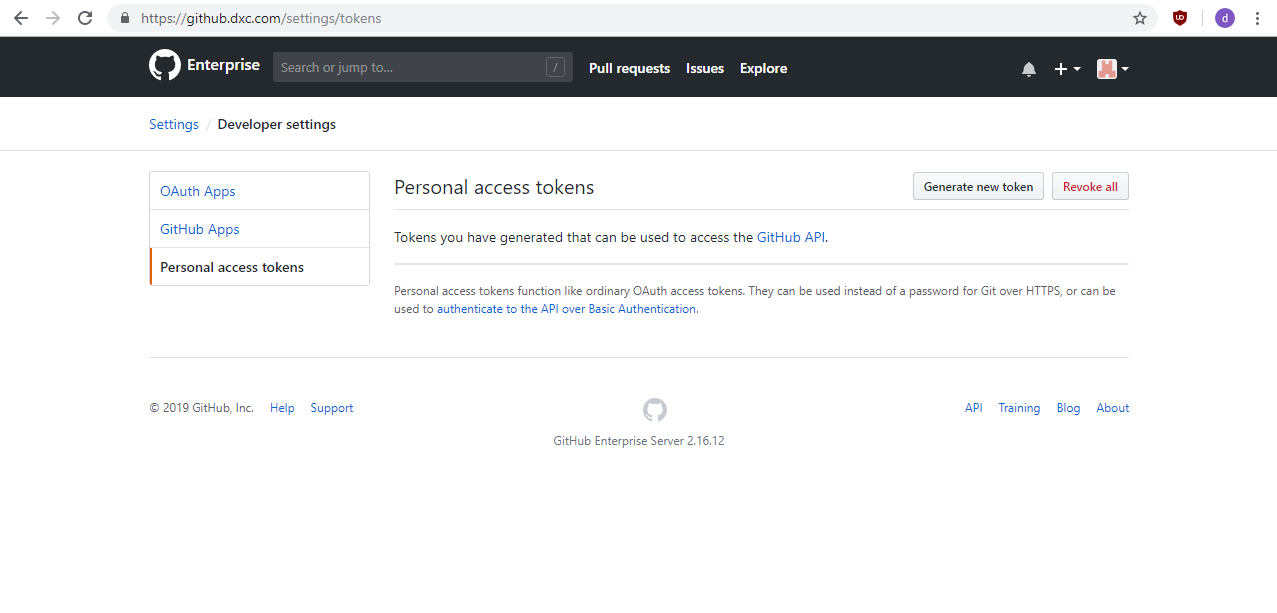
Hình 2.9 Các thay đổi đã được đẩy lên thành công

Dựa vào hình trên , em đã thành công trong việc push các thay đổi từ kho lưu trữ cục bộ đến kho lưu trữ từ xa lên trên nhánh master. Nhánh master là nhánh làm việc chính của một dự án hoặc của một nhóm nhà phát triển nên việc push các thay đổi liên tục lên nhánh master là điều không nên vì nếu như nhiều nhà phát triển thường xuyên push lên nhánh master sẽ khiến nhánh trở nên khó kiểm soát , hỗn độn dễ bị xung đột . Vì vậy , thay vì làm việc trực tiếp trên nhánh master , các nhà phát triển sẽ tạo ra các nhánh riêng để làm việc rồi sau đó sẽ gộp vào trong nhánh master.Nhánh riêng làm việc cũng như với nhánh master , các thay đổi sẽ được commit , push lên kho lưu trữ từ xa từ kho lưu trữ cục bộ . Các nhánh riêng này độc lập với nhánh master , thay đổi trên các nhánh riêng sẽ không ảnh hưởng tới nhánh chính. Việc này cho phép các nhà phát triển kiểm soát các phiên bản làm việc của mình dễ dàng hơn cũng như việc thay đổi nhánh theo ý muốn mà không sợ bị xung đột với các thay đổi của các nhà phát triển khác. Để tạo nhánh riêng , ta có thể sử dụng cả Github desktop và Github web để tạo chỉ với một số thao tác đơn giản. Khi một nhánh được tạo mới , Github sẽ nhân bản các files ở nhánh master cho nhánh mới được khởi tạo để đảm bảo sự trùng khớp về mặt dữ liệu khi tạo nhánh. Sau đây, em sẽ tạo một nhánh riêng có tên “PowershellTestBranch” , nhánh này sẽ là nhánh làm việc riêng của em trước khi gộp lên nhánh master.



Hình 2.10 : Nhánh PowershellTestBranch đã được tạo

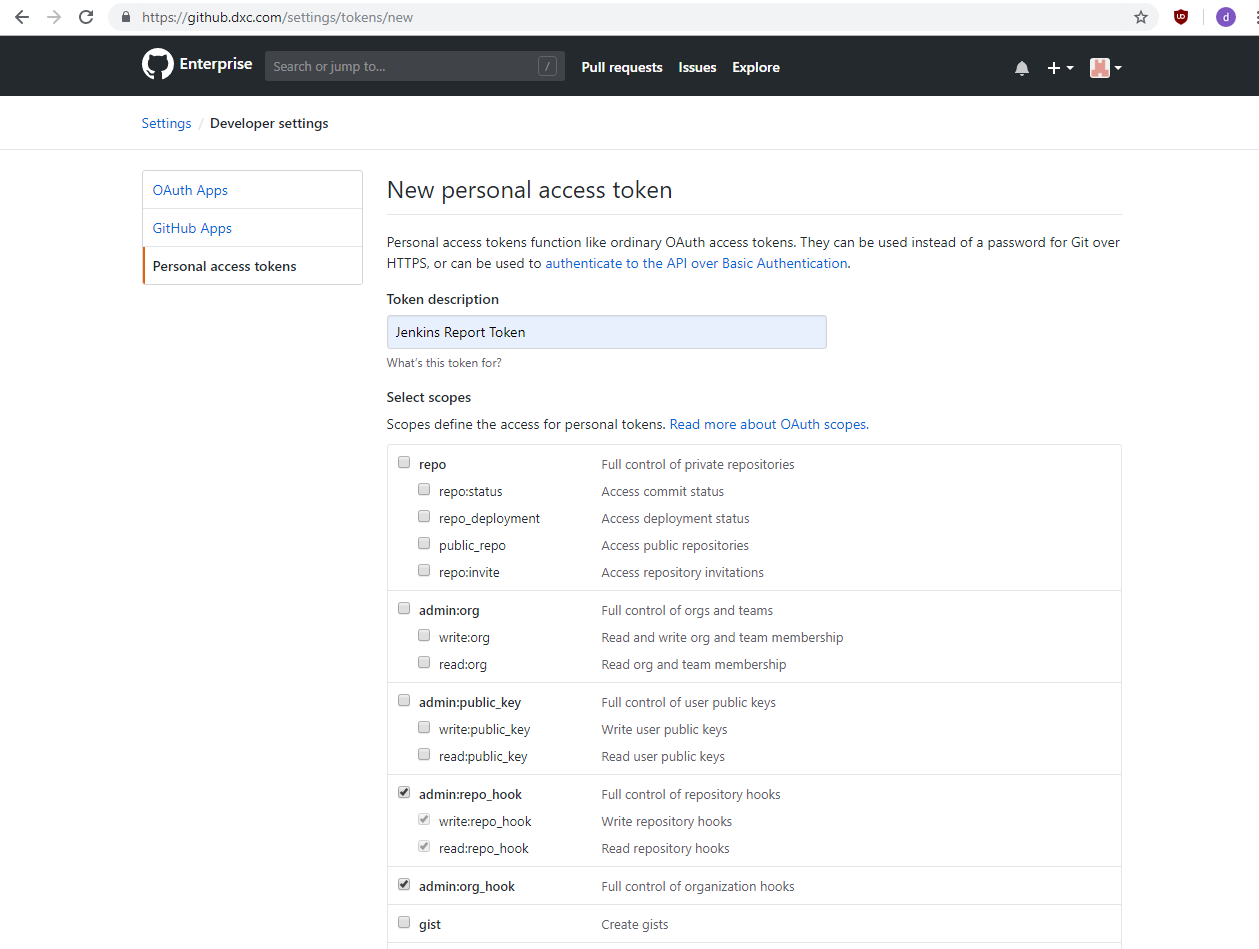
Các kho lưu trữ và nhánh đã được khởi tạo và thiết lập để sử dụng , tuy nhiên các thao tác ở trên đều hướng tới người chủ sở hữu kho lưu trữ. Các phần mềm hay các bên thứ ba sẽ bị gặp hạn chế khi truy cập tới kho lưu trữ vì chúng không thể dễ dàng đăng nhập vào Github như chủ sở hữu. Thay vì đăng nhập theo kiểu truyền thống , các phần mềm thứ ba có thể được cấp quyền truy cập tới kho lưu trữ bằng một tính năng của Github gọi là Token truy cập cá nhân ( Personal Access Token ) . Token truy cập cá nhân hoạt động như các Oauth token khác , nó có thể được dùng thay cho mật khẩu đăng nhập vào Git thông qua Https hoặc xác thực tới các API . Không chỉ vậy , người chủ sở hữu kho lưu trữ nhân còn có thể cấu hình Token truy cập cá nhân để giới hạn một số quyền nhất định cho bên thứ ba để làm việc với kho lưu trữ. Để tạo ra Token truy cập cá nhân , trước tiên cần phải vào mục Setting của tài khoản , đây là nơi mà người dùng có thể xem và quản lý các mục như Emails , SSH keys , Notifications ,… Sau đó vào mục Developer Setting , Token truy cập cá nhân sẽ ở đó .



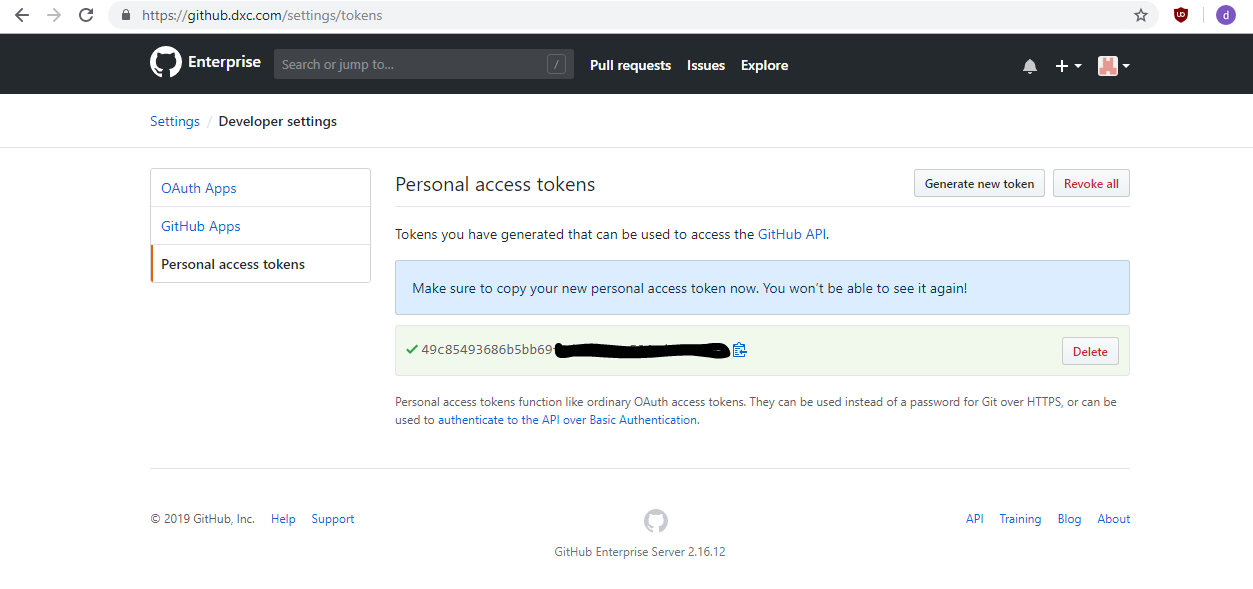
Hình 2.11 : Giao diện Developer settings

Chọn vào Generate new token sẽ đưa người dùng tới trang tạo Token. Trang tạo Token được chia làm 2 phần chính :

* **Token description :** Đây là miêu tả của Token. Nếu như người dùng có sử dụng nhiều token thì nên viết miêu tả khác nhau để có thể biết được từng Token sử dụng cho API nào. Token của chúng em sẽ được đặt miêu tả là “Jenkins Report Token”. Token này sẽ được sử dụng trong việc truy cập và xác thực kho lưu trữ với Jenkins .
* **Scropes :** Đây là những quyền nhất định để có thể chỉ định cho Token. Tùy vào mục đích sử dụng của Token mà lựa chọn quyền thích hợp cho Token. Chúng em chỉ chọn hai quyền cho Token đó là “**admin:repo\_hook**”và “**admin:org\_hook”.**



Hình 2.12 : Giao diện tạo Token truy cập cá nhân

 Sau khi chọn vào Generate Token , Github sẽ tạo ra một đoạn mã Token và đưa người dùng tới trang chứa đoạn mã đó để người dùng có thể lưu lại đoạn mã. Với đoạn mã này , người dùng có thể sử dụng nó để xác thực vào Github API , sử dụng nó như một loại tài khoản , ….

Hình 2.13 Đoạn mã Token được Github tạo ra

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. Nguyen Hong Son , Continuous Integration with Jenkins - Bài 1: Giới thiệu về CI và Jenkins , 5/5/2017 , <https://viblo.asia/p/continuous-integration-with-jenkins-bai-1-gioi-thieu-ve-ci-va-jenkins-OeVKBggEZkW>
2. Tran Van Luan , Giới thiệu về Jenkins: Một Open Source Continuous Integration Server , 26/1/2016 , <https://viblo.asia/p/gioi-thieu-ve-jenkins-mot-open-source-continuous-integration-server-zb7vDV0OMjKd>
3. Chaupm , Jenkins là gì? Mô hình kiểm tra code tự động nổi tiếng , 22/12/2018 , <https://tech.vccloud.vn/jenkins-la-gi-mo-hinh-kiem-tra-code-tu-dong-noi-tieng-2018122210502845.htm>
4. Phung The Tai , Tìm hiểu về Microsoft Azure: Azure Storage phần 1 , 23/8/2018 , <https://viblo.asia/p/tim-hieu-ve-microsoft-azure-azure-storage-phan-1-maGK7j0M5j2>

**Tiếng Anh**

* 1. Martin Heller , What is Jenkins? The CI server explained , 5/12/2017, <https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html>
  2. Bugfender , What Is Jenkins and Why Should You Be Using It? , 21/11/2017 , <https://dev.to/bugfenderapp/what-is-jenkins-and-why-should-you-be-using-it-2pe>
  3. Saurabh , Jenkins Tutorial | Continuous Integration Using Jenkins | Edureka , 22/5/2019 , <https://www.edureka.co/blog/jenkins-tutorial/>
  4. Saurabh , What is Jenkins? | Jenkins For Continuous Integration | Edureka , 22/5/2019 , <https://www.edureka.co/blog/what-is-jenkins/>
  5. Margaret Rouse , Microsoft Azure (Windows Azure) , 8/2018 , <https://searchcloudcomputing.techtarget.com/definition/Windows-Azure>
  6. Nicole Shortslef , Microsoft Azure Explained: What It Is and Why It Matters , <https://ccbtechnology.com/what-microsoft-azure-is-and-why-it-matters/>
  7. Sumo Logic , What is Microsoft Azure and Why Use It ? , 16/9/2016 , <https://www.sumologic.com/brief/what-is-azure/>
  8. Ryan Oistacher , Azure Cloud Services - A Complete Guide , 27/11/2018 , <https://blog.5nine.com/what-is-microsoft-azure>
  9. Hemant Sharma , What is Azure? – An Introduction To Microsoft Azure Cloud , 22/5/2019 , <https://www.edureka.co/blog/what-is-azure/>
  10. Nicola Wright , Microsoft Azure: virtual machines for the confused , <https://www.nigelfrank.com/blog/microsoft-azure-virtual-machines-for-the-confused/>
  11. Duan van der Westhuizen , Explaining Microsoft Azure Virtual Machines to a Windows SysAdmin , 24/5/2017 , <https://blog.rackspace.com/explaining-microsoft-azure-virtual-machines-to-a-windows-sysadmin>
  12. Hemant Sharma , Microsoft Azure Tutorial – Cloud Computing With Azure , 28/5/2019 , <https://www.edureka.co/blog/microsoft-azure-tutorial>
  13. Hemant Sharma , Azure Storage Tutorial – Tables, Blobs, Queues & File Storage in Microsoft Azure , 22/5/2019 , <https://www.edureka.co/blog/azure-storage-tutorial/>
  14. Russell Smith , Understanding Azure Storage: Storage Types and Redundancy , 17/3/2017 , <https://www.petri.com/understanding-azure-storage-storage-types-redundancy>
  15. Chandan Patra , Azure Storage: Service Overview and Introduction , 11/11/2015 , <https://cloudacademy.com/blog/azure-storage-service-overview/>
  16. Denis G , Microsoft Azure Storage Types Explained , 3/10/2017 , <https://www.cloudberrylab.com/resources/blog/microsoft-azure-storage-types-explained/>
  17. Minette Steynberg , Different Azure Storage types (File, Blob, Queue and Table) , 3/5/2017 , <https://www.sqlshack.com/different-azure-storage-types-file-blob-queue-table/>
  18. Hemant Sharma , Azure Virtual Network For Beginners – Securing Your Applications Using VPC , 22/5/2019 , <https://www.edureka.co/blog/azure-virtual-network-tutorial/>
  19. Kathleen Casey , Words to go: Microsoft Azure networking services , 9-2017 , <https://searchcloudcomputing.techtarget.com/feature/Words-to-go-Microsoft-Azure-networking-services>
  20. Edward Smith , Securing Azure Virtual Network , 28/2/2019 , <https://blog.cloudpassage.com/2019/02/28/securing-azure-virtual-network/>
  21. <https://www.guru99.com/jenkin-continuous-integration.html>
  22. <https://en.wikipedia.org/wiki/Jenkins_(software)>
  23. <https://jenkins.io/doc/pipeline/tour/getting-started/>
  24. <https://www.quora.com/What-is-Jenkins-When-and-why-is-it-used>
  25. <https://www.quora.com/What-is-Microsoft-Azure-and-its-uses>
  26. <https://en.wikipedia.org/wiki/Microsoft_Azure>
  27. <https://www.tutorialspoint.com/microsoft_azure/microsoft_azure_storage.htm>