**TRƯỜNG ĐẠI HỌC KỸ THUẬT – CÔNG NGHỆ CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN**

------*------*

****

**BÁO CÁO MÔN HỌC BẢO TRÌ PHẦN MỀM**

**NGHIÊN CỨU CÔNG CỤ QUẢN LÝ PHIÊN BẢN PHÂN TÁN GITHUB**

**NHÓM 2**

**Sinh viên thực hiện:**

**TRẦN HUY TOÀN (MSSV:1500504)**

**BÙI THỊ THỦY DUYÊN (MSSV:1501105)**

**Cần Thơ - 2017**

**TRƯỜNG ĐẠI HỌC KỸ THUẬT – CÔNG NGHỆ CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN**

------*------*

****

**BÁO CÁO MÔN HỌC BẢO TRÌ PHẦN MỀM**

**NGHIÊN CỨU CÔNG CỤ QUẢN LÝ PHIÊN BẢN PHÂN TÁN GITHUB**

|  |  |
| --- | --- |
| **GIÁO VIÊN HƯỚNG DẪN** | **SINH VIÊN THỰC HIỆN** |
| Th.S Nguyễn Xuân Hà Giang | Trần Huy Toàn (MSSV: 1500504  Bùi Thị Thủy Duyên (MSSV:1501105)  Ngành: Kỹ thuật phần mềm |

**Cần Thơ – 2017**

# NHẬN XÉT ĐÁNH GIÁ CỦA GVHD

# MỤC LỤC

[NHẬN XÉT ĐÁNH GIÁ CỦA GVHD 3](#_Toc496378652)

[MỤC LỤC i](#_Toc496378653)

[DANH MỤC VIẾT TẮT iii](#_Toc496378654)

[DANH MỤC HÌNH iv](#_Toc496378655)

[DANH MỤC BẢNG v](#_Toc496378656)

[MỤC TIÊU – PHẠM VI NGHIÊN CỨU ĐỀ TÀI 1](#_Toc496378657)

[PHẦN 1 2](#_Toc496378658)

[CÔNG CỤ QUẢN LÝ PHIÊN BẢN PHÂN TÁN GITHUB 2](#_Toc496378659)

[1.1 Giới thiệu về quản lý phiên bản phân tán 2](#_Toc496378660)

[1.1.1 Quản lý phiên bản 2](#_Toc496378661)

[1.1.2 Hệ thống quản lý phiên bản phân tán 2](#_Toc496378662)

[1.2 sơ lược về github 3](#_Toc496378663)

[1.2.1 Khái niệm về Git 3](#_Toc496378664)

[1.2.2 Khái niệm về github 4](#_Toc496378665)

[1.2.3 Lịch sử hình thành và phát triển của github 4](#_Toc496378666)

[1.2.4 Phạm vi 5](#_Toc496378667)

[1.2.5 Mức độ phổ biến 5](#_Toc496378668)

[1.3 Quy tắc làm việc trên kho mã nguồn github và lợi ích khi sử dụng github 6](#_Toc496378669)

[1.3.1 Quy Tắc 6](#_Toc496378670)

[1.3.2 Lợi ích khi sử dụng Github 8](#_Toc496378671)

[1.4 Các thao tác cơ bản khi sử dụng github 9](#_Toc496378672)

[1.4.1 Tạo Một repository 9](#_Toc496378673)

[1.4.2 Clone 9](#_Toc496378674)

[1.4.3 Add 9](#_Toc496378675)

[1.4.4 Remove 10](#_Toc496378676)

[1.4.5 Commit 10](#_Toc496378677)

[1.4.6 Push 10](#_Toc496378678)

[1.4.7 Pull 10](#_Toc496378679)

[1.4.8 Remote 10](#_Toc496378680)

[14.4.9 Fork 10](#_Toc496378681)

[1.4.10 Star 10](#_Toc496378682)

[1.4.11 Watch 11](#_Toc496378683)

[1.5 CÀI ĐẶT VÀ SỬ DỤNG GITHUB 11](#_Toc496378684)

[1.5.1 Tạo tài khoản 11](#_Toc496378685)

[1.5.2 Tải Github Desktop và cấu hình 11](#_Toc496378686)

[1.5.3 Code Up của mình lên repo 11](#_Toc496378687)

[1.5.4 Khôi phục lại phiên bản cũ của repo 13](#_Toc496378688)

[1.5.5 Lưu repo của người khác 14](#_Toc496378689)

[1.5.6 Gửi code và cùng contribute một dự án 15](#_Toc496378690)

[1.5.7 Cài đặt github desktop 16](#_Toc496378691)

[PHẦN 2 28](#_Toc496378692)

[ĐÁNH GIÁ VỚI CÁC PHẦN MỀM CÙNG CHỨC NĂNG 28](#_Toc496378693)

[2.1 GITHUB 28](#_Toc496378694)

[2.2 GITLAB 28](#_Toc496378695)

[2.3 NHỮNG ĐIỂM KHÁC BIỆT GIỮA 2 CÔNG CỤ 28](#_Toc496378696)

[TỔNG KẾT 30](#_Toc496378697)

[TÀI LIỆU THAM KHẢO 31](#_Toc496378698)

# DANH MỤC VIẾT TẮT

# DANH MỤC HÌNH

No table of figures entries found.

# DANH MỤC BẢNG

**No table of figures entries found.**

# MỤC TIÊU – PHẠM VI NGHIÊN CỨU ĐỀ TÀI

# PHẦN 1

# CÔNG CỤ QUẢN LÝ PHIÊN BẢN PHÂN TÁN GITHUB

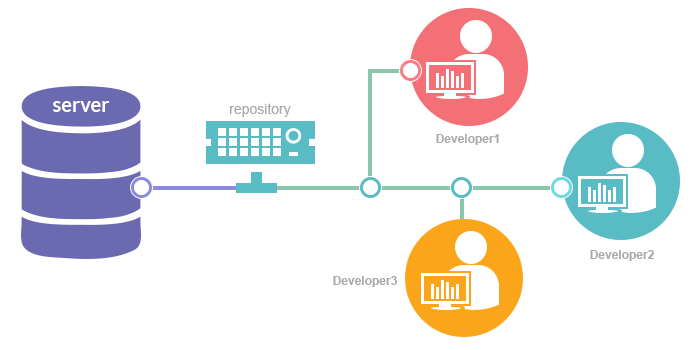
## 1.1 Giới thiệu về quản lý phiên bản phân tán

### 1.1.1 Quản lý phiên bản

Quản lý phiên bản là một hệ thống lưu trữ các thay đổi của một tập tin (file) hoặc tập hợp các tập tin theo thời gian, do đó nó giúp bạn có thể quay lại một phiên bản xác định nào đó sau này.

Hệ thống quản lý cho phép bạn: khôi phục lại phiên bản cũ của các file, khôi phục lại phiên bản cũ của toàn bộ dự án, xem lại các thay đổi đã được thực hiện theo thời gian, xem ai là người thực hiện thay đổi cuối cùng có thể gây ra sự cố, hay xem ai là người đã gây ra sự cố đó và còn nhiều hơn thế nữa. Sử dụng VCS còn đồng nghĩa với việc khi bạn làm rối tung mọi thứ lên hay vô tình xoá mất các file đi, bạn có khôi phục lại chúng một cách dễ dàng. Hơn nữa, tất cả quá trình này có thể được thực hiện rất nhanh chóng và không hề tốn quá nhiều công sức.

### 1.1.2 Hệ thống quản lý phiên bản phân tán

Hệ Thống Quản Lý Phiên Bản Phân Tán - Distributed Version Control Systems (DVCSs). Trong các DVCS (ví dụ như Git, Mercurial, Bazaar hay Darcs), các máy khách không chỉ "check out" (sao chép về máy cục bộ) phiên bản mới nhất của các tập tin: chúng sao chép (mirror) toàn bộ kho chứa (repository). Chính 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. Mỗi checkout thực sự là một bản sao đầy đủ của tất cả dữ liệu.

**Hình 1.1 Hệ thống quản lý phiên bản phân tán**

Phần lớn các hệ thống này xử lý rất tốt việc quản lý nhiều kho chứa từ xa, vì thế bạn có thể cộng tác với nhiều nhóm người khác nhau theo những cách khác nhau trong cùng một dự án. Việc này cho phép bạn cài đặt nhiều loại "tiến trình công việc" (workflow) không thể thực hiện được với các hệ thống tập trung.

## 1.2 sơ lược về github

Đặt vấn đề: có một nhóm gồm 4 người đang hợp tác với nhau cùng xây dựng một phần mềm cho khách hàng. Dĩ nhiên cả 4 người này sẽ làm việc với nhau bằng cách mỗi người tự viết code một ít sau đó nén file lại và gửi cho nhau qua email hoặc tin nhắn facebook. Mỗi người sau khi nhận được code của nhau sẽ tiến hành tự xem và sửa lại, sau đó cùng nhau ghép vào phần mềm mà họ cần làm. Tuy nhiên, một người trong nhóm bỗng nhiên vô tình viết code sai dẫn tới sản phẩm bị lỗi, mà trước đó thì cả nhóm chủ quan không lưu lại bản backup do phần mềm đã quá lớn. Và thế là cả nhóm lại cùng nhau ngồi fix chỗ bị lỗi và tiếp tục gửi cho nhau như vậy.

***Thế vấn đề ở đây là gì?***

* Việc tự code riêng và gửi cho nhau qua email/ tin nhắn rất mất thời gian của nhau. Giá như trong một nhóm,người A có thể chủ động xem những thay đổi/review được code của người B từ xa và tiến hành gộp trực tiếp những thay đổi của người B vào sản phẩm thì đỡ biết bao nhiêu.
* Việc sửa code mà không sao lưu khiến cho cả nhóm lại mất thời gian, thậm chí viết lại cả phần mềm từ đầu.

Và thế là Git ra đời có thể giúp ta khắc phục và giải quyết được vấn đề khi làm việc nhóm.Vậy Git là gì?

### 1.2.1 Khái niệm về Git

**Git** là tên gọi của một **Hệ thống quản lý phiên bản phân tán** (Distributed Version Control System – ***DVCS***). 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 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.

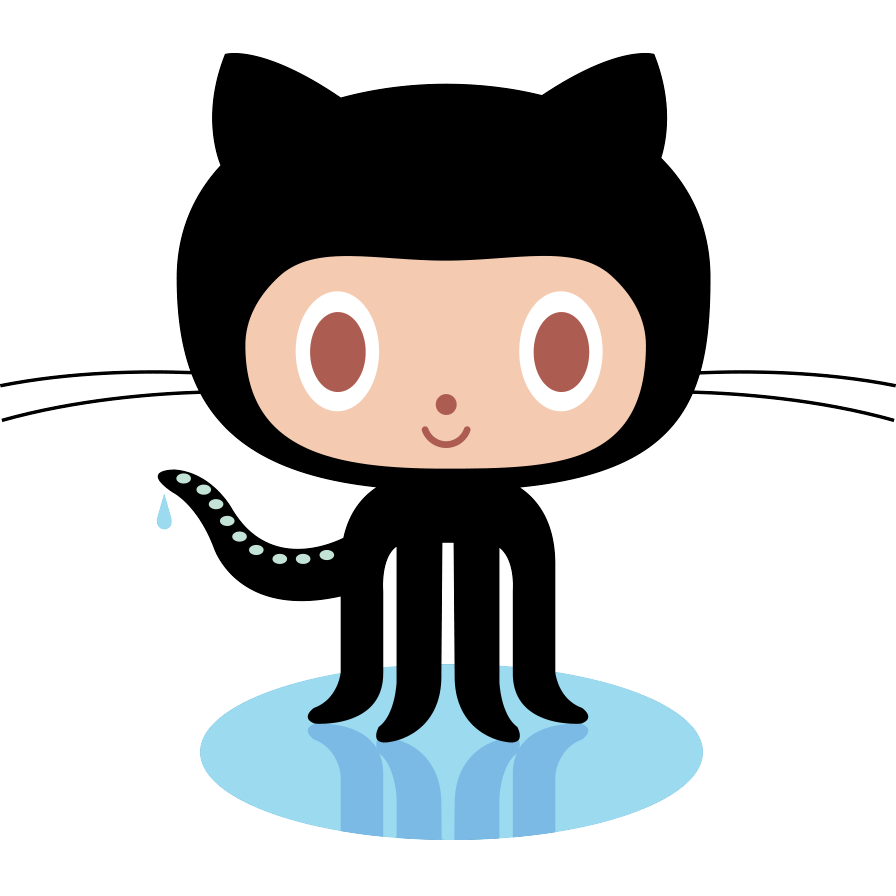
**Hình 1.2 Git**

Hiểu đơn giản là với Git thì nó sẽ giúp ta lưu lại các phiên bản mỗi khi có sự thay đổi mã nguồn, đo đó nếu có làm sai ở đâu đó trong phiên bản mới thì ta vẫn có thể dễ dàng khắc phục bằng cách khôi phục lại phiên bản cũ ổn định hơn.

Trong thực tế thì vẫn còn khá nhiều người nhầm lẫn giữa Git và Github, Git là tên gọi của một mô hình hệ thống, còn Github lại là một dịch vụ máy chủ repository công cộng, mỗi người đều có thể tạo tài khoản và đưa source code của mình lên đây.Để hiểu thêm về github ta sẽ đi sâu vào tìm hiểu nó.

### 1.2.2 Khái niệm về github

GitHub là một dịch vụ cung cấp [kho lưu trữ mã nguồn](https://vi.wikipedia.org/w/index.php?title=Kho_l%C6%B0u_tr%E1%BB%AF_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)) 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 [mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) sẽ được cung cấp kho lưu trữ miễn phí

Biểu tượng của github là chú mèo bạch tuột oputus

**Hình 1.3 Biểu tượng của Github**

### 1.2.3 Lịch sử hình thành và phát triển của github

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.

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…

Các phần mềm chạy GitHub được viết bằng [Ruby on Rails](https://vi.wikipedia.org/w/index.php?title=Ruby_on_Rails&action=edit&redlink=1) và [Erlang](https://vi.wikipedia.org/w/index.php?title=Erlang&action=edit&redlink=1) bởi GitHub, Inc,…

Hệ thống kiểm soát phiên bản (VCS).

### 1.2.4 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
* Các trang web nhỏ có thể được lưu trữ từ kho công cộng trên Github.
* Theo dõi vấn đề và tính năng yêu cầu
* Trực quan của dữ liệu không gian địa lý
* Biểu đồ Gantt

### 1.2.5 Mức độ phổ biến

*Năm 2009,* năm đầu tiên GitHub là trực tuyến, nó tích lũy 46.000 kho công cộng, 17.000 trong số nọ 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.

*Ngày 05 tháng 7 năm 2009,* một Blog Github bài thông báo họ đạt đến 100.000 người sử dụng nhãn hiệu.

Ngày 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.

*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".

*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.

Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn. Chính vì vậy làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.  
Github đã trở thành một yếu 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.

## 1.3 Quy tắc làm việc trên kho mã nguồn github và lợi ích khi sử dụng github

### 1.3.1 Quy Tắc

#### 1.3.1.1 Tài khoản quản lý

Kho code chính của OpenCPS (OpenCPS Repository) được quản lý thông qua tài khoản của tổ chức.

Tài khoản của tổ chức OpenCPS được quản lý bởi Ban quản trị OpenCPS và đội ngũ phát triển code chính.

#### 1.3.1.2 Nguyên tắc phát triển code

***Tự do đóng góp:*** Tất cả mọi người dù là người quản lý kho code, thành viên đội code hay những người đóng góp thuộc các tổ chức, công ty, các cá nhân... đều có thể đóng góp phát triển code một cách tự do.

***Ghi công đóng góp:*** Việc phát triển code phải thực hiện trên kho code riêng, không được code trực tiếp trên kho code chính của OpenCPS (kể cả tài khoản của người quản lý kho code).

***Kiểm duyệt chéo:*** Những người quản lý kho code sẽ chịu trách nhiệm việc kiểm duyệt code được đóng góp vào kho code chính. (Nên hạn chế việc tự duyệt code của mình trên kho code chính).

#### 1.3.1.3 Nguyên tắc phát triển code

Để tham gia phát triển code, chỉ cần 1 tài khoản trên Github. Sau đó truy cập kho code của OpenCPS, ấn nút Fork để copy project có sẵn thành project của mình, rồi sau đó ta có thể tự do chỉnh sửa project đó của mình.

Chỉnh sửa xong thì ấn nút Pull Request để gửi yêu cầu đóng góp lên cho những người quản lý kho code.

Những người quản lý kho code sẽ kiểm tra và phê duyệt các đóng góp này. Nếu đóng góp chưa đạt hoặc cần chỉnh sửa thì những người quản lý có thể viết góp ý ngay tại yêu cầu đóng góp và trả lại để người đóng góp chỉnh sửa theo góp ý và gửi lại sau khi đã sửa xong. Quá trình này diễn ra cho đến khi yêu cầu đóng góp được chấp nhận, đóng góp đó sẽ được trộn (merge) vào các nhánh phù hợp với dự án được đóng góp.

Để tăng hiệu suất quản lý, toàn bộ công việc chỉnh sửa, xử lý xung đột... trước khi gửi đóng góp lên sẽ do người đóng góp xử lý. Người quản lý kho code sẽ chỉ việc kiểm tra và trả lời chứ không phải mất công để sửa lại code được đóng góp.

#### 1.3.1.4 Gợi ý cho các doanh nghiệp / nhóm tham gia phát triển code

Nếu doanh nghiệp của bạn có nhiều nhân viên tham gia phát triển code thì thay vì tham gia dự án dưới tư cách của cá nhân, các bạn có thể tham gia phát triển dự án dưới tư cách của doanh nghiệp (trường hợp doanh nghiệp trả lương cho nhân viên để phát triển code của OpenCPS). Dưới đây là hướng dẫn:

* Đại diện doanh nghiệp đăng ký tài khoản code, tạo tổ chức mang tên doanh nghiệp và add các thành viên vào tổ chức của mình.
* Đại diện doanh nghiệp fork dự án về kho code của doanh nghiệp.
* Các thành viên trong doanh nghiệp fork dự án từ kho code của doanh nghiệp về kho code của cá nhân. Nhân viên doanh nghiệp Pull Request vào kho code của doanh nghiệp sau đó doanh nghiệp Pull Request vào kho code chung của OpenCPS.

Theo hình thức này, một cá nhân vừa có thể tham gia trực tiếp vào dự án của OpenCPS (phát triển ngoài giờ làm việc) và tham gia dưới tư cách là nhân viên của doanh nghiệp (phát triển trong giờ làm việc).

#### 1.3.1.5 Phát hành phiên bản mới (dành cho Release Engineering team)

Việc phát hành OpenCPS được thực hiện bằng cách tagging trực tiếp trên nhánh master [Kho git gốc](https://github.com/VietOpenCPS/opencps) theo qui trình sau:

* Clone kho gốc về: git clone git@github.com:VietOpenCPS/opencps.git
* Hoặc pull bản latest trên nhánh "Release Candidate branch" về (nếu đã clone kho local sẵn): git checkout <qa-rc-xxx>; git pull
* Chuyển về & cập nhật nhánh master: git checkout master; git pull
* Thực hiện merge nhánh RC về nhánh master: git merge <qa-rc-xxx>
* Tagging release: git tag <release-number>, rồi nhập tagging message vào, thường là URL đến Release Notes (cần phải viết & đưa lên trước tại: [Thông tin phát hành](https://github.com/VietOpenCPS/doc/wiki#th%C3%B4ng-tin-ph%C3%A1t-h%C3%A0nh)
* Nên thêm tham số -s để ký vào bản phát hành bằng khóa OpenPGP
* Push lại lên kho gốc trên GitHub: git push
* Optional. Merge nhánh RC về nhánh develop: trong nhiều trường hợp, đặc biệt là giai đoạn đầu của dự án, toàn bộ code trên nhánh RC branch sẽ được sử dụng tiếp trong các phiên bản tiếp theo => cần thực hiện merge tương tự như trên cho nhánh develop.

***Lưu ý:***

* Release Notes cần được viết riêng trước lên một trang wiki mới và link vào trang Wiki Home tại phần [Thông tin phát hành](https://github.com/VietOpenCPS/doc/wiki#th%C3%B4ng-tin-ph%C3%A1t-h%C3%A0nh)
* Sau khi push lên kho git gốc, cần viết thông báo phát hành gửi lên Mailing list. Tham khảo: [Thông báo phát hành OpenCPS 1.0](http://lists.opencps.vn/pipermail/opencps/2016-May/000210.html)

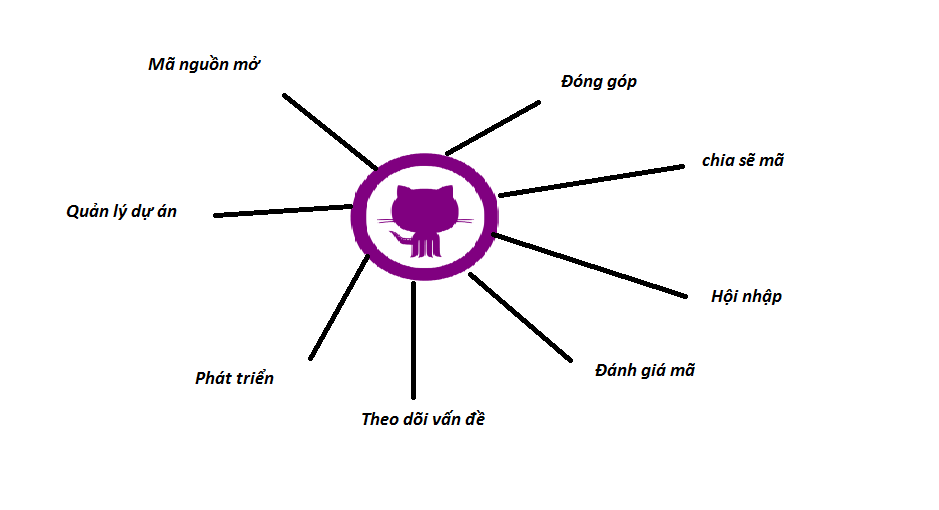
#### 1.3.1.6 Các chú ý

Trong quá trình làm phải commit thường xuyên lên nhánh develop, khi xong xuôi thì merge/rebase vào nhánh master. Không nên chờ đến khi có kết quả cuối cùng mới commit lên kho code chính, vì như thế sẽ gia tăng nguy cơ xung đột code giữa các thành viên trong nhóm và giữa các nhóm với nhau.

Quá trình phát triển, kho cá nhân của bạn (fork) trên GitHub có thể cũ so với kho gốc chính của dự án (upstream). Tham khảo các thủ tục bên dưới để đồng bộ kho local với kho upstream rồi push lên kho cá nhân (fork) trên GitHub.

### 1.3.2 Lợi ích khi sử dụng Github

#### 1.3.2.1 Lý do dùng Github



**Hình 1.4 Lợi ích của Github**

#### 1.3.2.2 Lợi ích khi sử dụng Github

* Dễ 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).
* Có chỗ lưu trữ source code miễn phí, phù hợp với các nguồn mở.
* 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 deployment sản phẩm.
* Giúp chứng tỏ bạn là ai thông qua các chương trình, source code mà bạn làm.
* Giúp bạn cải thiện được khả năng code thông qua việc đọc code và code chung với người khác.
* Đây là kho tài nguyên vô tận với vô số các project mã nguồn mở.

#### 1.3.2.3 Giá cả

***Với Cá nhân***

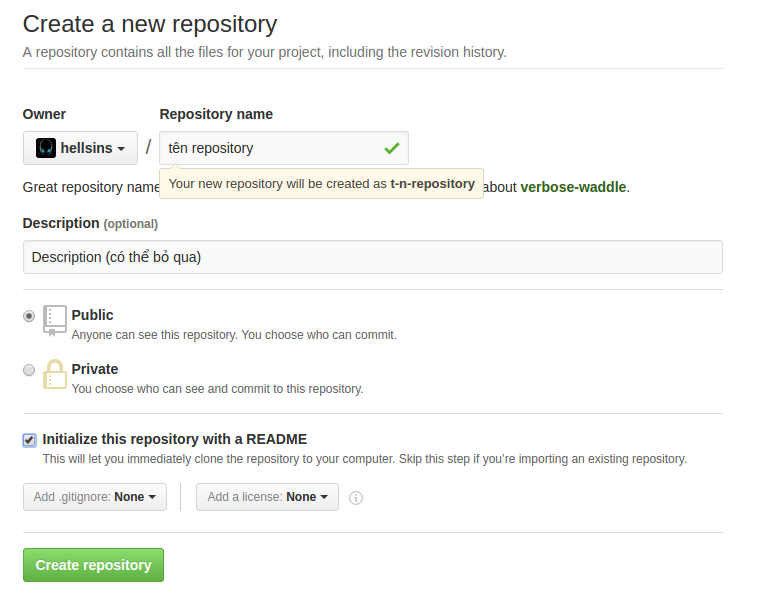
* Miễn phí mọi thứ, không có private repositories
* Ngoài ra còn có các gói Micro, Small, Medium với từng mức giá khác nhau theo tháng (7-50$/ tháng) và giới hạn từ 5-20 private repositories

***Với tổ chức***

* Miễn phí mọi thứ, không có private repositories
* Các gói Bronze, Silver, Gold, Platinum giá đắt hơn (25-200$/ tháng và số lượng private repositories từ 10-125).

## 1.4 Các thao tác cơ bản khi sử dụng github

### 1.4.1 Tạo Một repository

[](https://github.com/hellsins/sysadmin_level1/blob/master/Task04_Git_and_Github/img/git6.png)Bạn vào Github và chọn Create repository và điền thông tin như hình

**Hình 1.5 Tạo reporitory**

### 1.4.2 Clone

Để clone một repo về ta có thể chọn Clone or Download và nhấn Download Zip hoặc copy đường dẫn (bạn có thể chọn clone sử dụng SSH hoặc HTTP) và thực hiện với lệnh sau:

Git clone [đường dẫn vừa copy thư mục chứa repo trên local]

Ví vụ:

* Clone SSH: git clone [git@github.com](mailto:git@github.com):hellsins/hellsins.github.io.git ~/github
* Clone HTTP: git clone <https://github.com/hellsins/hellsins.github.io.git> ~/github

Lưu ý bạn có thể không sử dụng thư mục chứa repo trên local và bạn sẽ clone repo đó về thư mục hiện tại (pwd)

### 1.4.3 Add

Để thực hiện hành động add ta sử dụng lệnh sau:

**Git add [tên\_file]**: dùng để add file chỉ định

**Git add \***: dùng để add tất cả

**Git add --all**: dùng để add tất cả

### 1.4.4 Remove

Để remove một thư mục hay một file nào đó bạn có thể xóa ở máy local sau đó add và commit lại là xong.

Nếu muốn xóa repo bạn vào repo đó trên server và chọn Delete this repository ở phần Setting. Đọc warning và chọn yes...

### 1.4.5 Commit

Để thực hiện hành động commit ta sử dụng lệnh sau:

**Git commit \*:** dùng để commit tất cả

**Git commit [tên\_file] -m [thêm chú thích]:** dùng để thêm chú thích cho commit của mình

### 1.4.6 Push

Sau khi commit thì tất cả đã lưu vào máy cục bộ và giờ thì chúng ta sẽ push lên server với lệnh:

**Git push -u origin master:** Nhập passphrase nếu có

### 1.4.7 Pull

Hành động pull là hành động được thực hiện khi server có những thay đổi mà máy cục bộ vẫn chưa và giờ muốn cập nhật những thay đổi này. Ta dùng lệnh:

**Git pull:** lưu ý vào đúng repo cần pull

### 1.4.8 Remote

Việc remote được hiện khi bạn muốn add một máy chủ từ xa nào đó. Để thực hiện remote bạn làm như sau:

**Git remote add origin [link repo]**

Để liệt kê các remote mà bạn đã add thì có thể dùng lệnh:

**Git remote -v**

### 14.4.9 Fork

Tại một thời điểm ta muốn phân phối hay sử dụng một project hay repo của ai đó để bắt đầu và điều này nghĩa là ta sẽ Fork một repo về. Sau khi Fork về thì repo đó sẽ tồn tại trên github của chúng ta. Chúng ta có thể clone nó về máy local để bắt đầu sử dụng.

Sau khi một repo được được clone, nó sẽ có một remote origin trỏ đến repo mà chúng ta đã Fork về chứ không phải là repo gốc. Để theo dõi repo gốc mà chúng ta đã Fork, chúng ta cần add một remote khác có tên là upstream:

**Git remote add upstream [link repo gốc]**

**Git fetch upstream**

### 1.4.10 Star

Star một repo trong github như thể hiện cho việc repo này được nhiều người quan tâm, theo dõi. Đây cũng là cách để bạn tăng khả năng xuất hiện của repo mình trên github.

Bạn có thể Star một repo bất kỳ và khi đó bạn có thể truy cập nhanh chóng và dể dàng theo dõi repo mà bạn quan tâm. Ngoài ra đây cũng là một sự đánh giá cho chủ nhân repo.

Để thực hiện bạn chỉ cần nhấn vào Star trên repo đã chọn

### 1.4.11 Watch

Để thực hiện bạn chọn Watch trên repo mà bạn muốn và khi đó bạn sẽ nhận được thông báo cho các yêu cầu mới hay vấn đề gì xảy ra với repo đó.

Lệnh này sẽ truy cập vào dự án từ xa nào đó và cập nhật dữ liệu mà bạn chưa có trên repo đó. Sau khi Fetch xong bạn có thể tham chiếu đến toàn bộ các nhánh của dự án đó.

## 1.5 CÀI ĐẶT VÀ SỬ DỤNG GITHUB

Để làm việc được với Github thì cần có:

* Tài khoản và Repository.
* Github Desktop.
* Cấu hình để đồng bộ.

### 1.5.1 Tạo tài khoản

Vào trang <https://github.com/>  sau đó nhập thông tin như tạo tài khoản facebook.

### 1.5.2 Tải Github Desktop và cấu hình

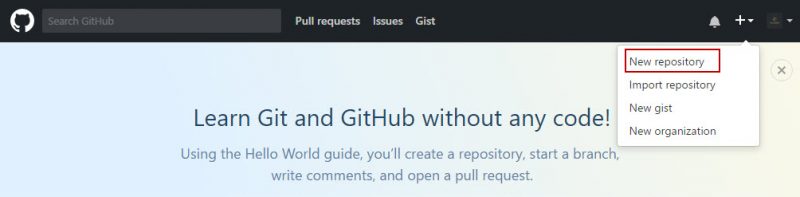
Việc sử dụng Github thông thường dính nhiều tới các lệnh, nếu bạn chưa quen với các hệ điều hành như linux/ ubuntu mà vẫn muốn trung thành với windows hoặc bạn còn bỡ ngỡ ban đầu thì việc sử dụng Github Desktop là một giải pháp khá ổn.

Để tải Github Desktop bạn vào địa chỉ  <https://desktop.github.com/>, tải bản cài về và cài như một chương trình bình thường. Sau khi cài xong sẽ có biểu tượng Github trên desktop, bạn click chuột vào đó để mở phần mềm

Ở màn hình welcome bạn cần làm một số thao tác sau:

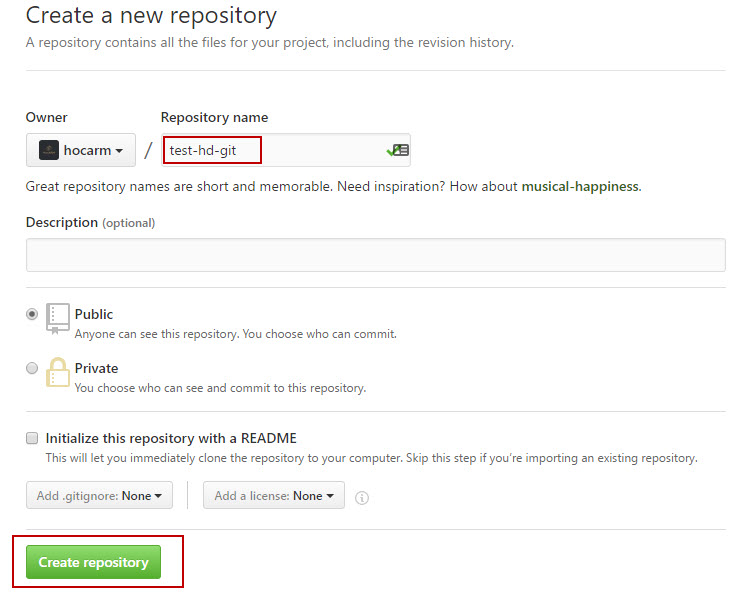
* Mục login nhập id và pass của bạn, sau đó chọn Login
* Mục config bạn chọn tên bất kỳ ở ô đầu tiên, ô thứ 2 là email của bạn
* Mục repositories sẽ kiểm tra xem trên server của bạn có source code nào không ? Do chưa tạo nên bạn chọn skip.

### 1.5.3 Code Up của mình lên repo

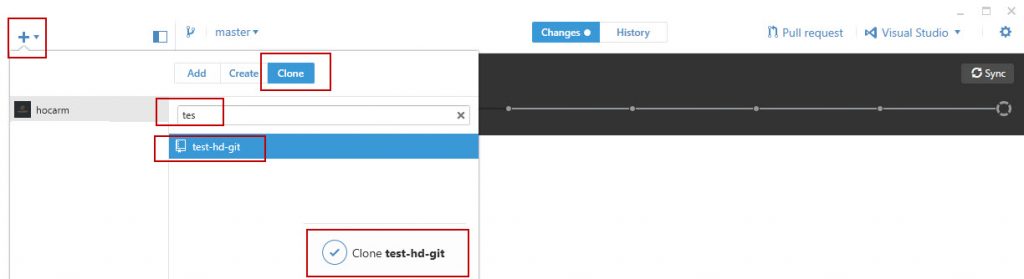
***Bước 1: Vào web github tạo một repo***

**Hình 1.6 Tạo repo**

***Bước 2: Nhập tên***

**Hình 1.7 Nhập tên**

***Bước 3: Mở Github desktop clone repo mình vừa tạo về, nếu github bạn quá nhiều repo thì dùng chức năng tìm kiếm***

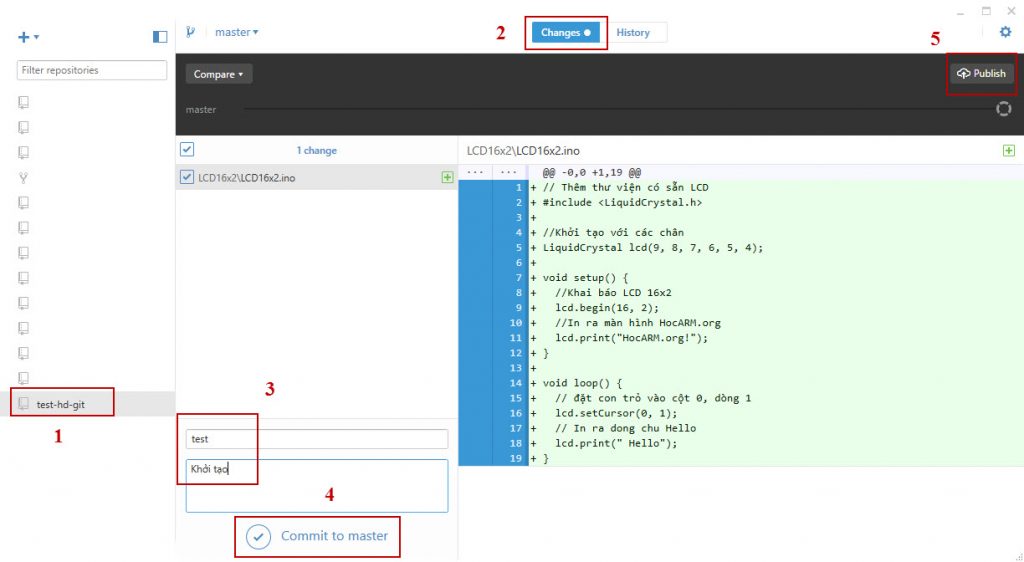


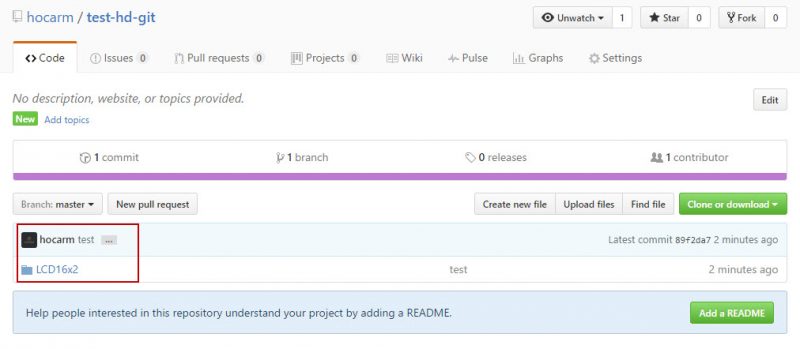
**Hình 1.8 Tìm kiếm repo**

***Bước 4: Sau khi ấn Clone thì sẽ có thông báo lưu trữ repo này, bạn có thể lưu ở bất kỳ vị trí nào bạn muốn.Ví dụ lưu thư mục này vào ổ D, trỏ nó về ổ D, sau đó OK.***

Vào lại ổ D xem thì chỉ có 1 thư mục trống tên test-hd-git.Giờ lấy ví dụ copy chương trình LCD16x2 với Arduino vào thư mục test-hd-git

***Bước 5: Mở Github Desktop chọn vào thư mục test-hd-git thực hiện đẩy code lên github theo các bước từ 1-4***

**Hình 1.9 Đẩy code lên Github**

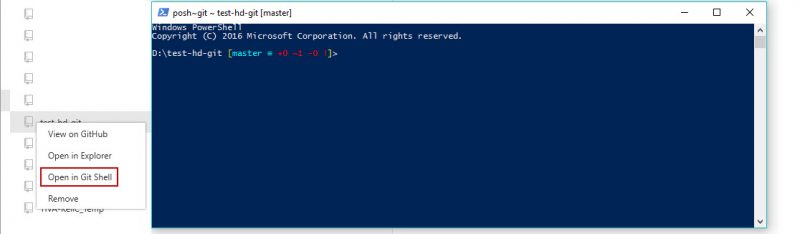
***Bước 6: Xem kết quả trên github, bạn có thể thấy thư mục vừa đẩy lên đã lên github rồi.***

**Hình 1.10 Xem thư mục đẩy lên**

***Bước 7: Chia sẻ repo của mình với mọi người.***

### 1.5.4 Khôi phục lại phiên bản cũ của repo

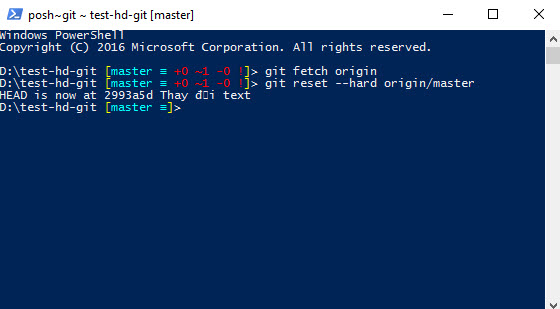
***Bước 1:  Mở gitshell bằng cách chuột phải vào thư mục test-hd-git chọn Open gitshell***

**Hình 1.11 Mở Gitshell**

***Bước 2: Gõ lệnh***

***Git fetch origin***

***Git reset --hard origin/master***

Lệnh này sẽ tự khôi phục source code của bạn về phiên bản được lưu trên github

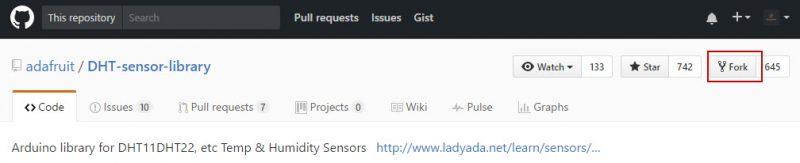
**Hình 1.12 Khôi phục Source code**

### 1.5.5 Lưu repo của người khác

Trong quá trình làm, bạn có thể sẽ thấy rất nhiều nguồn source code khác nhau, để mình có thể sử dụng được repo người khác thì cần phải clone repo của họ về máy của mình, cách thực hiện rất đơn giản, đó là chọn fork

Ví dụ muốn fork thư viện DHT của Adafruit thì:

* Vào trang github <https://github.com/adafruit/DHT-sensor-library>
* Chon fork bên góc phải, github sẽ tự lưu repo này vào trong tài khoản github của chúng ta.

**Hình 1.13 Lưu repo**

### 1.5.6 Gửi code và cùng contribute một dự án

Để đóng góp cùng phát triển source code cho ai đó (lấy ví dụ ở đây với thư viện DHT của Adafruit) thì mình cần thực hiện theo các bước sau:

***Bước 1:*** Fork source code của họ

***Bước 2:*** Clone project đã fork về máy tính của mình thông qua Github Desktop

***Bước 3:*** Cập nhật và đồng bộ thông tin mới nhất từ repo gốc bằng lệnh

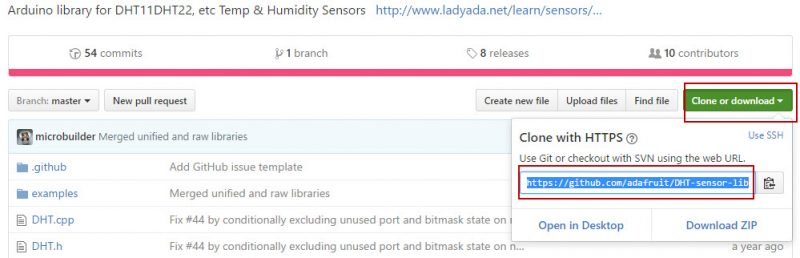
***Git remote add upstream https://github.com/xxxx.git***

***Git fetch upstream***

***Git checkout master***

***Git merge upstream/master***

***Git remote add upstream https://github.com/adafruit/DHT-sensor-library.git***

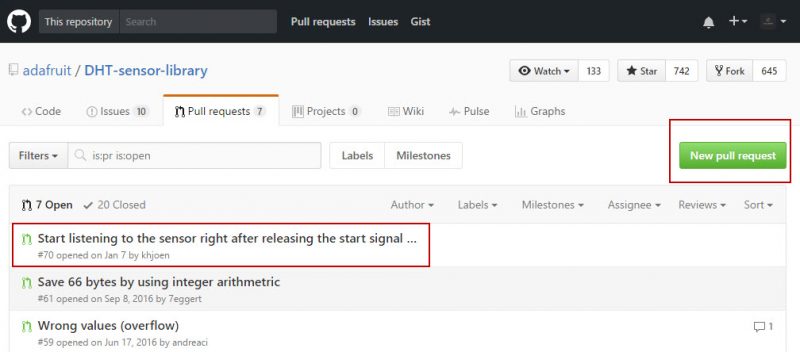
Địa chỉ git bạn có thể copy từ mục Clone or Download

**Hình 1.14 Gửi code**

***Bước 4:*** Chỉnh sửa và tối ưu code, ví dụ như thêm các chức năng vào thư viện

***Bước 5:*** Commit và sync lên github, lúc này những thay đổi chỉ được cập nhật lên tài khoản github của cá nhân, chưa có sự thay đổi trên repo gốc (Adafruit)

***Bước 6:*** Thực hiện pull request lên thư mục gốc, chờ chủ repo thảo luận, sau khi thấy hợp tình hợp lý thì họ sẽ merge code cũng như bổ sung các tính năng mới cập nhật của bạn vào code, đồng thời đưa bạn vào danh sách contributors.

**Hình 1.15 Pull request**

### 1.5.7 Cài đặt github desktop

#### 1.5.7.1 Đăng ký tài khoản Github

A screenshot of a cell phone

Description generated with very high confidenceĐiều đầu tiên trước khi thao tác với github thì phải tạo tài khoản thông qua trang web của Github: [www.github.com](http://www.github.com)

**Hình 1.16 đăng ký tài khoản Github**

Việc đăng ký một tài khoản là đơn giản, bạn chỉ cần nhập **username/password** và địa chỉ **email**. Sau khi đăng ký xong bạn cần vào Email kích hoạt tài khoản.

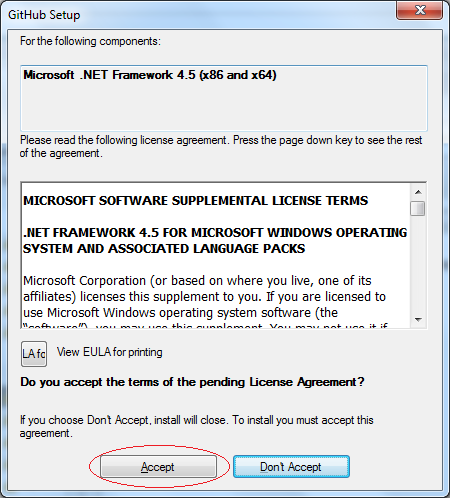
#### 1.5.7.2 Dowload và cài đặt Github desktop

GitHub Desktop về bản chất là một công cụ trực quan cho phép bạn quản lý Local Repository (Kho chứa địa phương) trên máy tính của bạn.

A screenshot of a cell phone

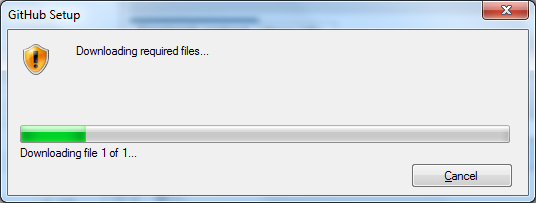
Description generated with very high confidenceĐể download GitHub Desktop bạn vào địa chỉ: https://windows.github.com

**Hình 1.17 Download Github windows**

****Sau khi download xong, bạn cần cài đặt **GitHub Desktop** vào máy tính:

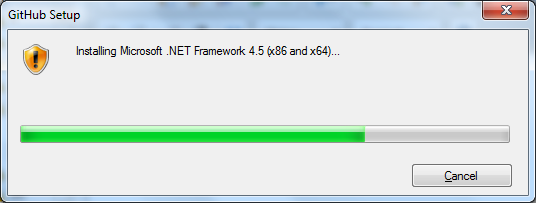
**Hình 1.18 Cài đặt Github**

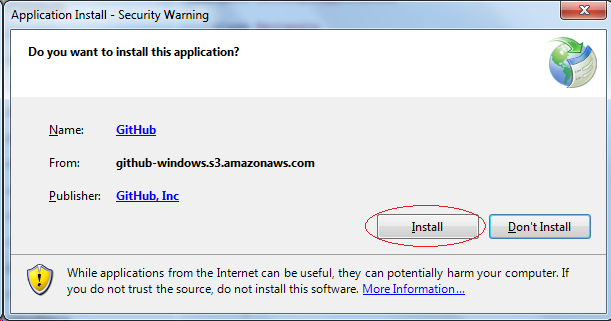
Bộ cài đặt sẽ download và cài đặt thư viện bắt buộc **Microsoft.Net**



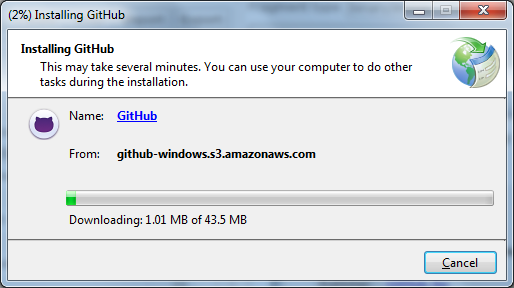
**Hình 1.19 Cài đặt Github**

Chờ cho tới khi tiến trình download hoàn thành. Gói thư viện **.Net** sẽ được cài đặt vào máy tính của bạn.

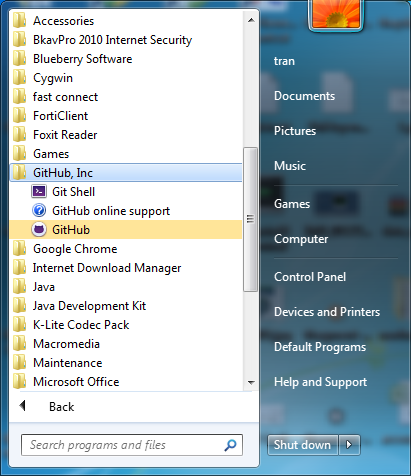
**Hình 1.20 Cài đặt Github**

Bộ cài đặt tiếp tục cài đặt **GitHub Desktop**:

**Hình 1.21 Bộ cài đặt Github**

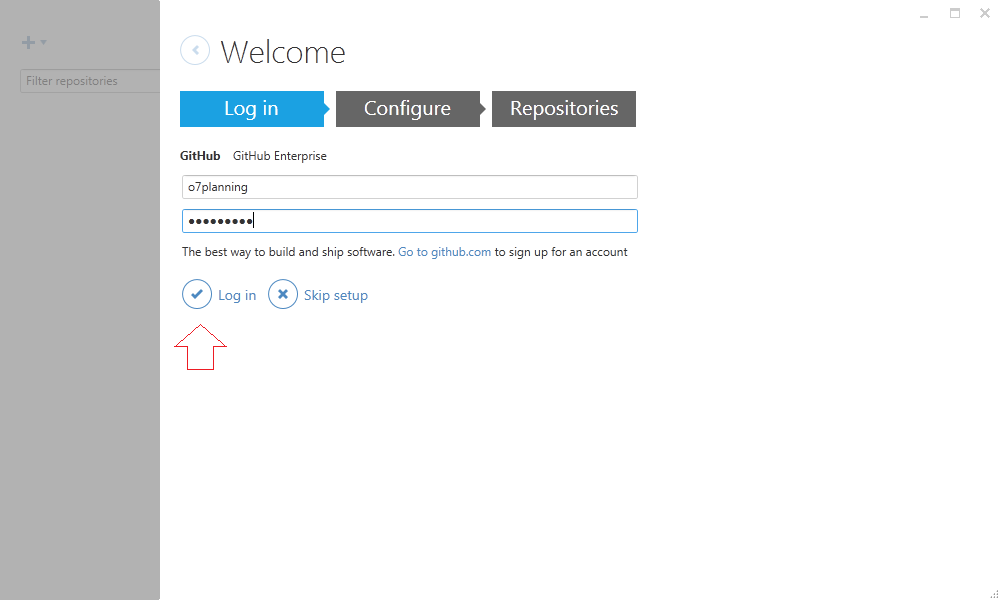
**Hình 1.22 Bộ cài đặt Github**

GitHub Desktop đã được cài đặt thành công.

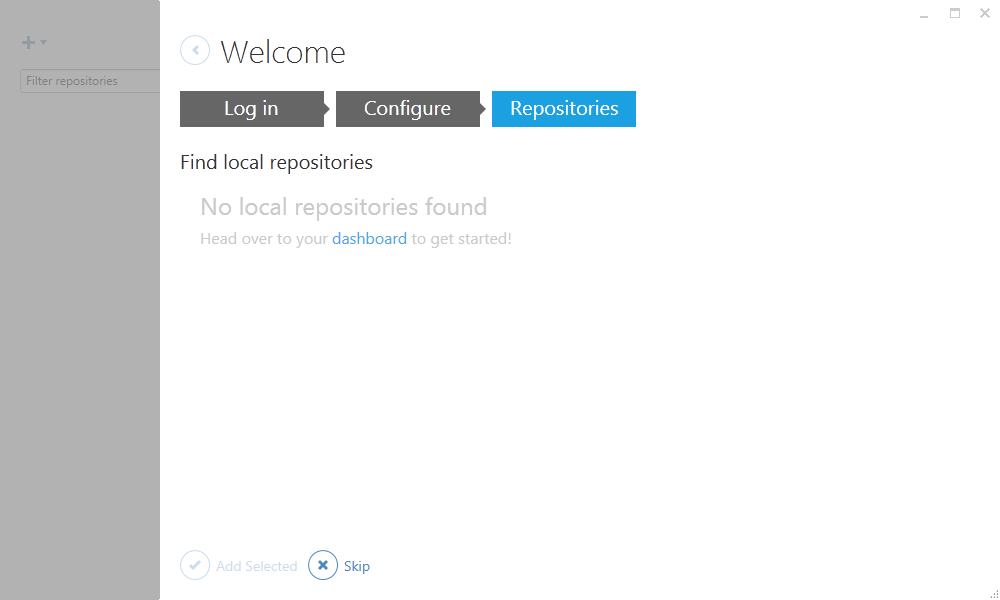


#### http://o7planning.org/vi/10283/cache/images/i/70469.png1.5.7.3 Chạy Github desktop

**Hình 1.23 Khởi động Github**

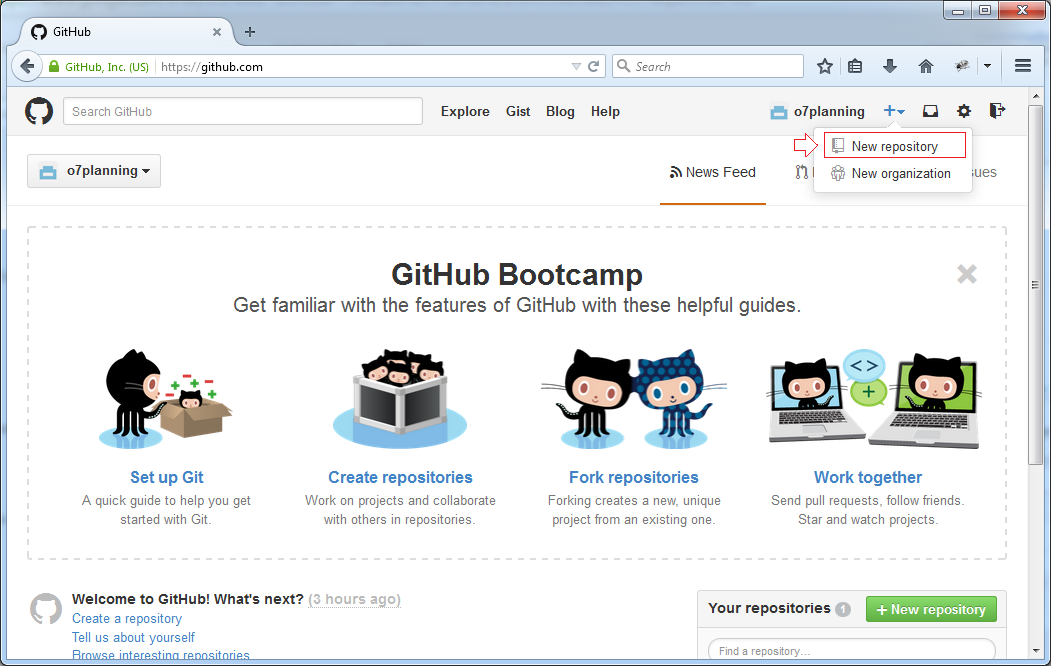
Đăng nhập trên **GitHub Desktop** để kết nối vào tài khoản **GitHub**của bạn

**Hình 1.24 Đăng nhập Github**

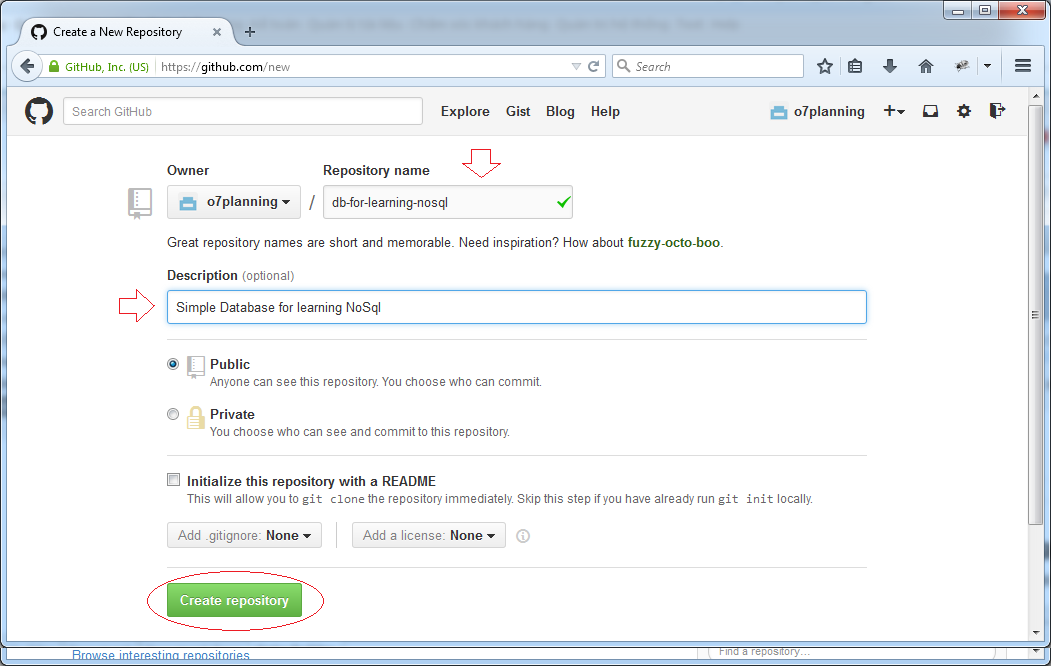
****Cho tới lúc này trên máy tính của bạn chưa có một **Local Repository** nào.

**Hình 1.25 Hộp thoại Find Repositories**

#### 1.5.7.4 Tạo GitHub Repository

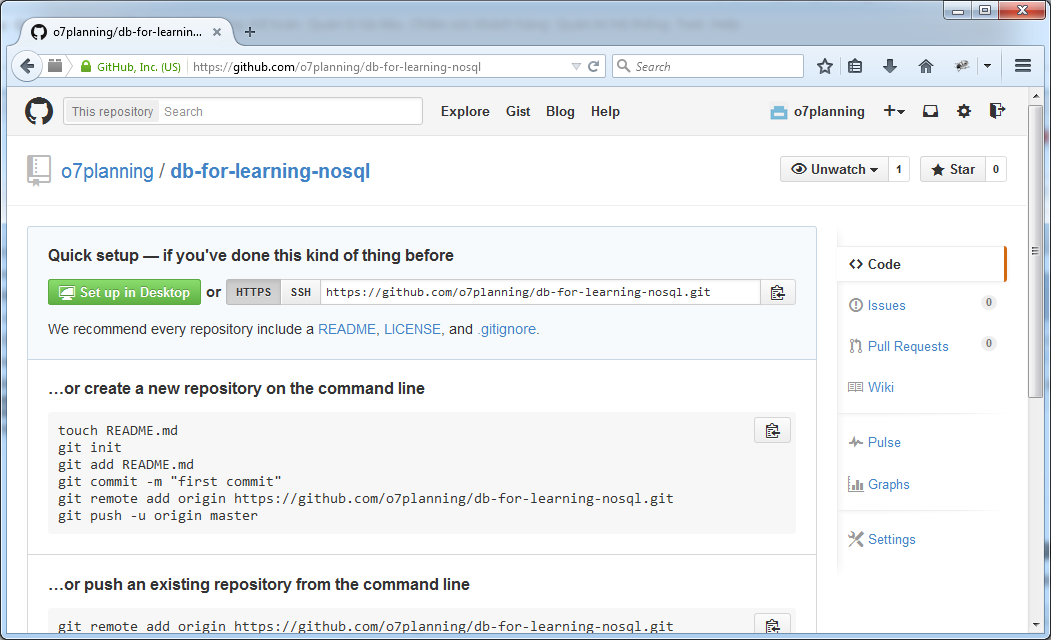
****Sau khi đăng ký xong tài khoản **GitHub**, và đăng nhập vào. Bạn có thể tạo một **GitHub Repository**.

**Hình 1.26 Tạo Github Repository**



**Hình 1.27 Tạo Github Repository**

**Repository**đã được tạo ra.

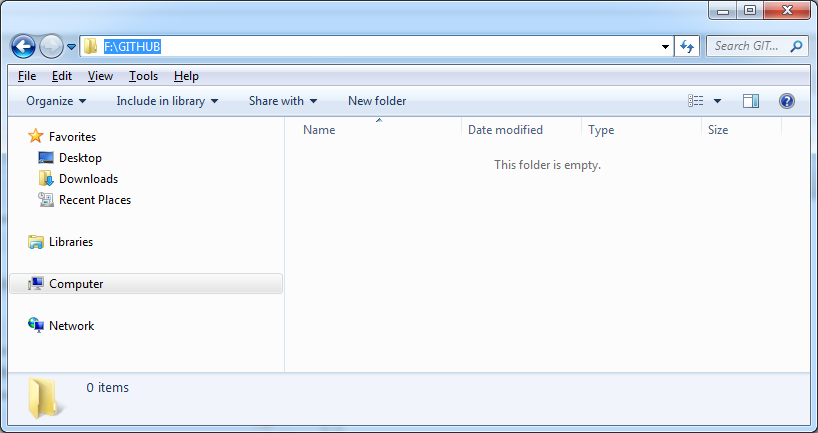


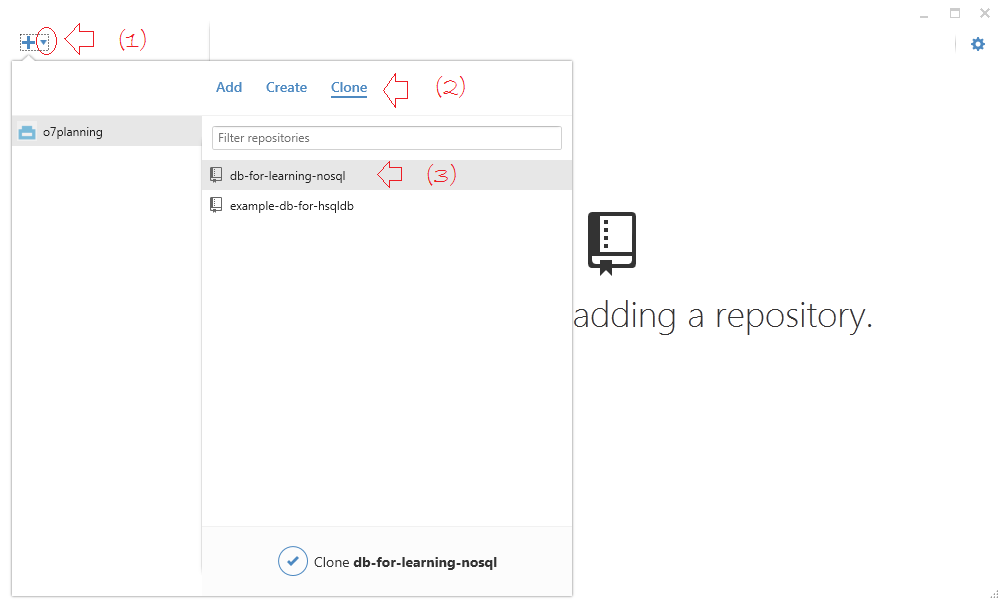
**Hình 1.28 Github Repository**

#### 1.5.7.5 Kết nối GitHub và GitHub Desktop

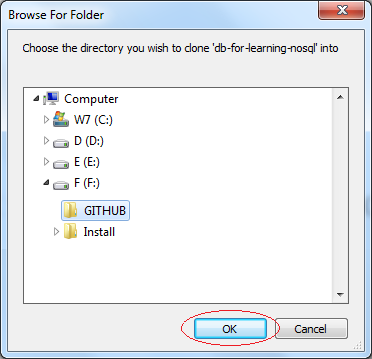
Trước hết cần chọn một thư mục rỗng để làm vị trí chứa dữ liệu địa phương.

Chẳng hạn:F:/GITHUB

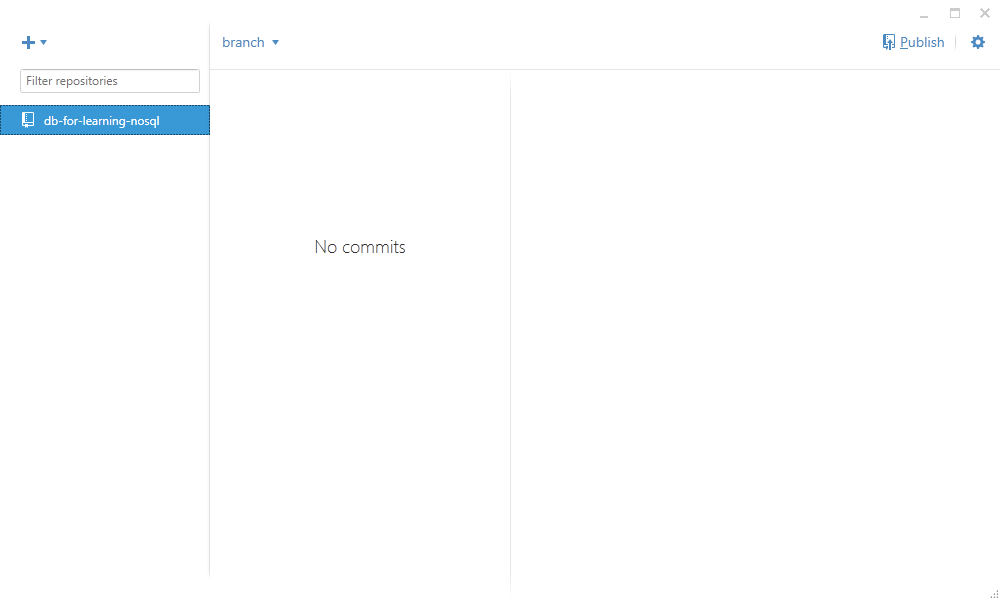
**Hình 1.29 Chọn vị trí chứa dữ liệu**

Trên **GitHub Desktop**, lựa chọn một **Repository**bạn đã tạo trên **GitHub** để **Clone**(Tạo bản sao chép) thành một bản ở máy tính địa phương của bạn.

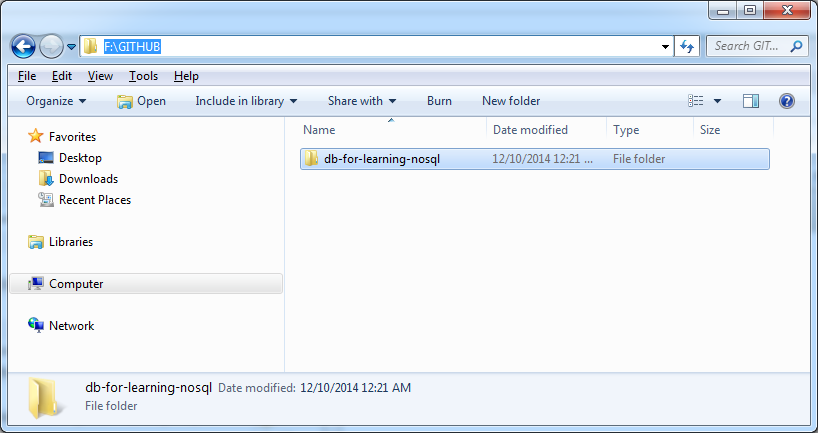
**Hình 1.30 Tạo bản sao chép trên máy tính từ Clone**

Lúc này trên **GitHub Desktop** bạn sẽ thấy một **Local Repository** đã được tạo ra.

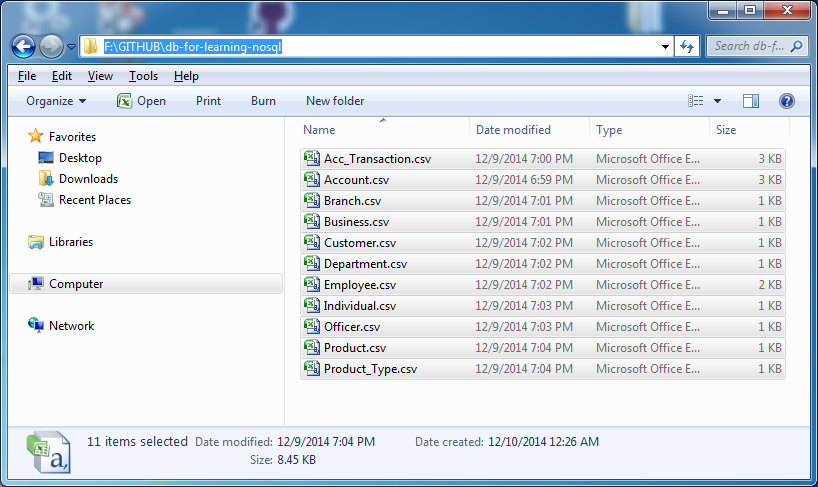
**Hình 1.31 Local Repository**

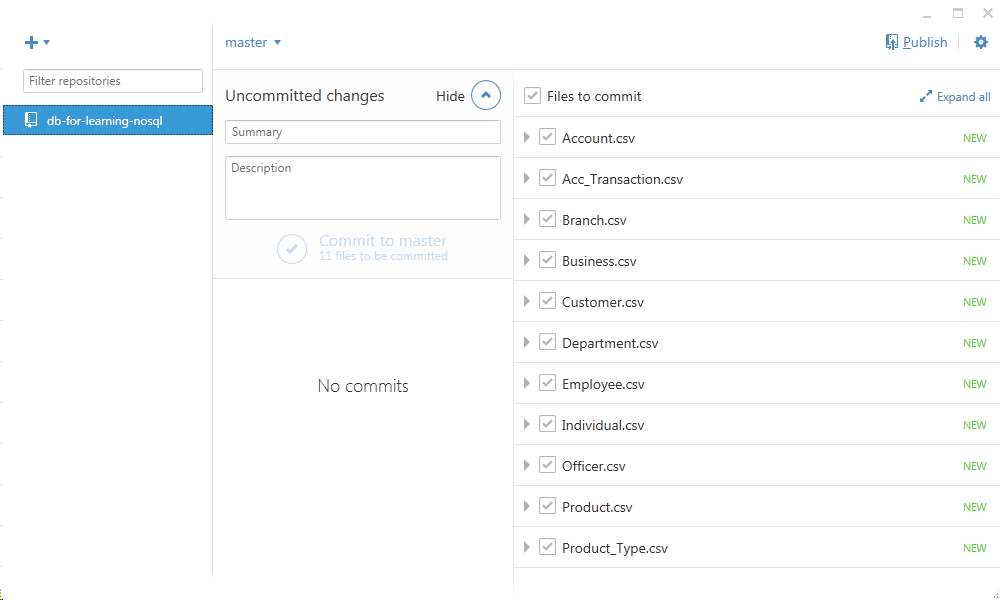
****Và trên thư mục tại ổ cứng, một thư mục con đã được tạo ra

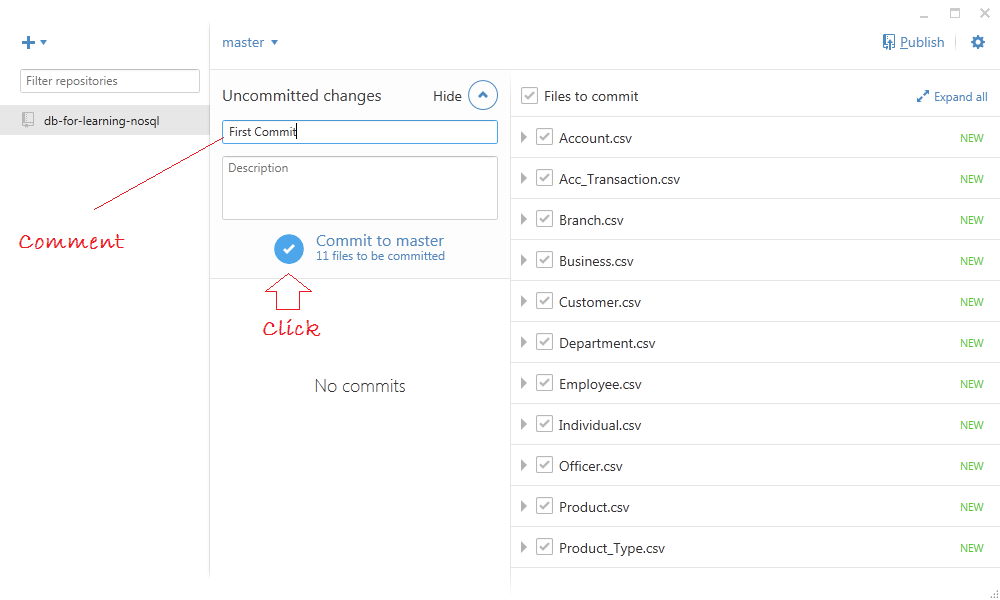
**Hình 1.32 Thư mục con được tạo**

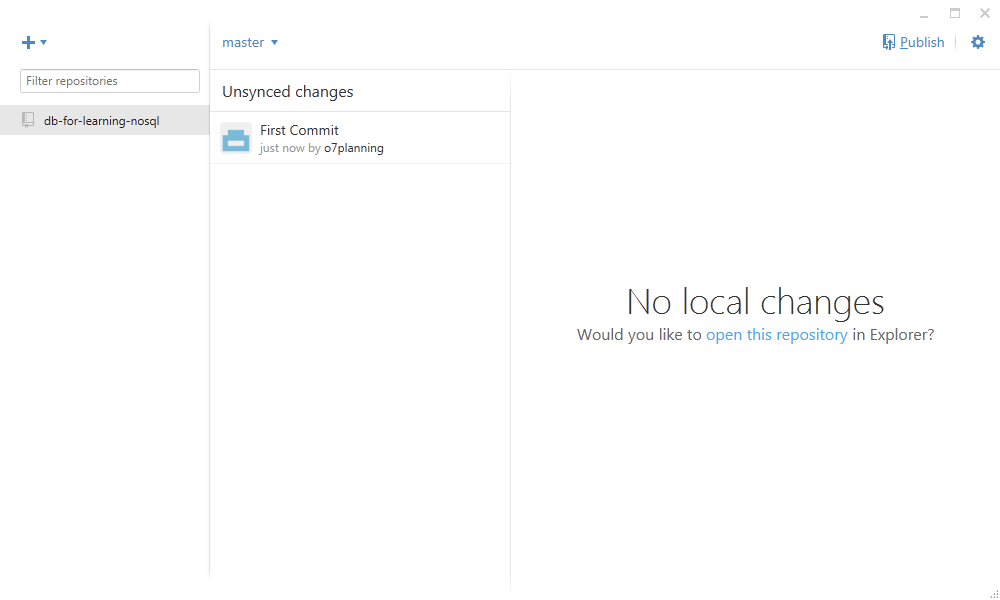
Copy một vài file dữ liệu của bạn vào **Local Repository**:

**Hình 1.33 Copy dữ liệu vào** **Local Repository**

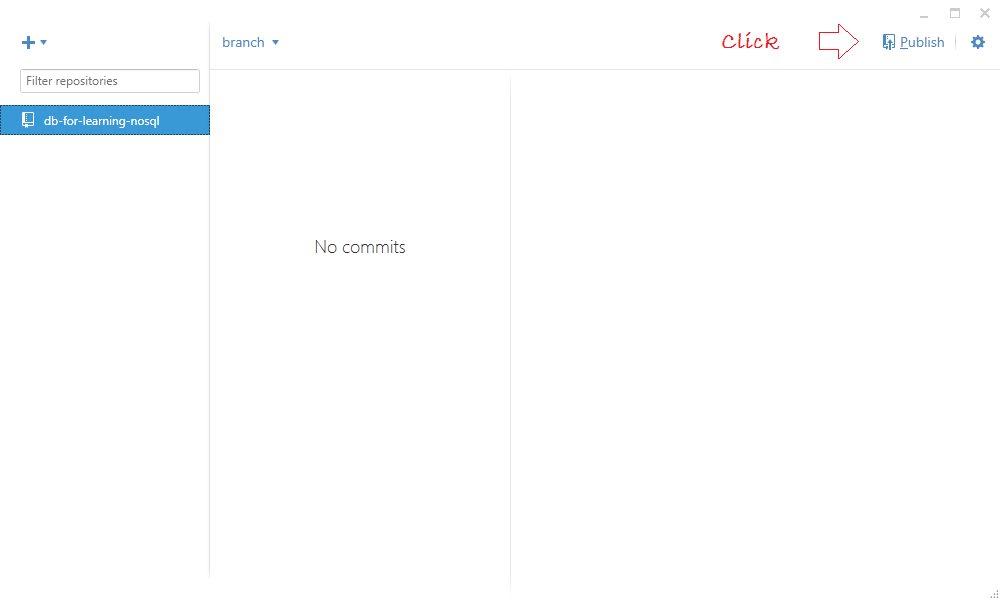
**GitHub Desktop** ngay lập tức nhận biết được các thay đổi tại **Local Repository**.

**Hình 1.34 Các thay đổi tại** **Local Repository**

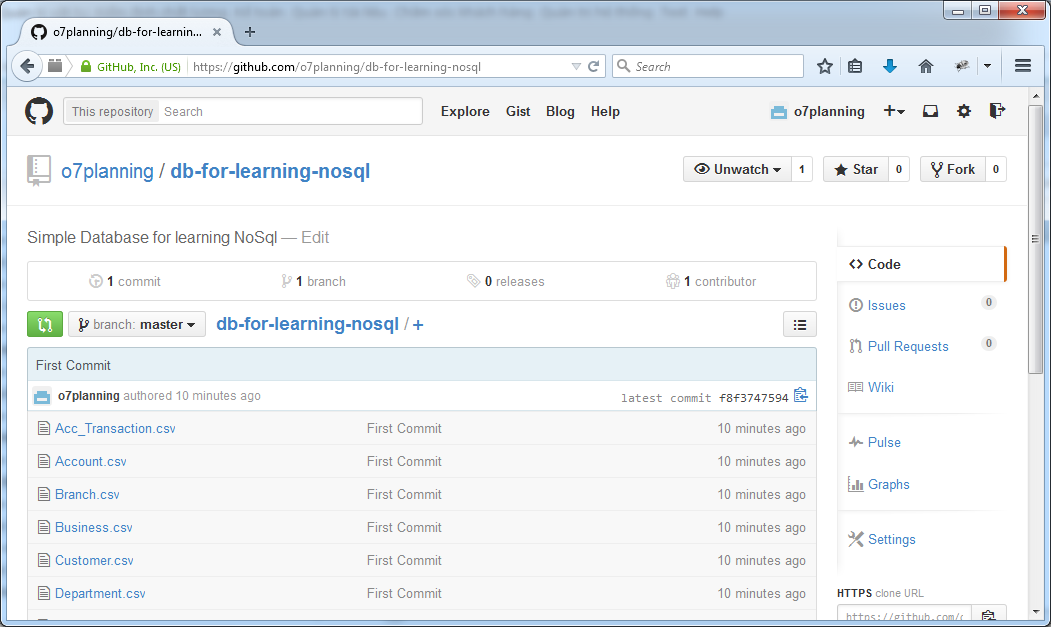
Nhập thông tin ghi chú (Comment) và nhấn **Commit**dữ liệu.

**Hình 1.35 Nhập thông tin ghi chú**

**Hình 1.36 Nhập thông tin ghi chú**

Cuối cùng là công khai cho mọi người có thể xem:

**Hình 1.37 Công khai dữ liệu**

Các file dữ liệu bạn có thể nhìn thấy trên **Server**

**Hình 1.38 Dữ liệu trên Server**

# PHẦN 2

# ĐÁNH GIÁ VỚI CÁC PHẦN MỀM CÙNG CHỨC NĂNG

GitLab và GitHub là kho lưu trữ Git dựa trên web. Mục đích là để quản lý các dự án phát triển web và các tập tin. Khi chúng thay đổi theo thời gian.

## 2.1 GITHUB

Các dự án GitHub có thể được công khai và mọi mã được chia sẻ công khai đều tự do được mở cho mọi người. Bạn cũng có thể có các dự án riêng tư, tuy nhiên, họ cần một kế hoạch GitHub đã thanh toán.

Các kho lưu trữ công cộng trên GitHub thường được sử dụng để chia sẻ phần mềm nguồn mở. Tôi không đào sâu vào các vấn đề cơ bản của GitHub ở đây nhưng GitHub cũng cung cấp các tính năng như theo dõi vấn đề và trang GitHub.

## 2.2 GITLAB

Tương tự như GitHub, GitLab là một quản lý kho lưu trữ cho phép các nhóm cộng tác trên mã. Viết bằng Ruby, GitLab cung cấp một số tính năng tương tự để theo dõi vấn đề và quản lý dự án như GitHub.

## 2.3 NHỮNG ĐIỂM KHÁC BIỆT GIỮA 2 CÔNG CỤ

***Mức xác thực***

Với GitLab bạn có thể thiết lập và sửa đổi quyền của người dùng theo vai trò của họ Trong GitHub, bạn có thể quyết định xem ai đó có quyền đọc hoặc ghi vào kho lưu trữ không. Với GitLab bạn có thể cung cấp truy cập vào bộ theo dõi vấn đề mà không cần cho phép mã nguồn.

***File dính kèm***

Trong GitLab bạn có thể đính kèm bất kỳ tệp tin nào vào bất kỳ vấn đề nào. Bạn không thể làm điều đó bên trong GitHub.

***Theo dõi vấn đề***

Nếu bạn đang sử dụng các sự cố GitHub, bạn có thể nghĩ rằng nó thiếu một số chức năng. GitLab cung cấp một Tracker vấn đề mạnh mẽ cho phép bạn thay đổi tình trạng và chuyển nhượng cho nhiều vấn đề cùng một lúc.

***Tình trạng trong công việc tiến trình***

Với GitLab, các nhà phát triển có thể gắn nhãn một dự án với "Work in progress" cho cộng tác viên một dấu hiệu rõ ràng rằng mã này vẫn chưa kết thúc. Điều này có thể là một điều nhỏ, mặc dù nó có thể được thực sự hữu ích. Tính năng này ngăn chặn mã được hợp nhất với các mã khác trước khi nó thực sự kết thúc.

***Tích* *hợp***

Cả GitLab và GitHub đều cung cấp một loạt các tích hợp bên thứ ba. Tích hợp hệ thống kiểm soát phiên bản của bạn với các ứng dụng khác làm phong phú thêm luồng công việc của bạn và có thể tăng năng suất cho các nhà phát triển của bạn.

***Doanh nghiệp gitlab và github***

Ở cấp độ doanh nghiệp, có nhiều yếu tố cần được xem xét khi đưa ra quyết định có thông tin về việc nên sử dụng GitLab hay GitHub hay không

GitHub rất phổ biến trong giới phát triển.

Tuy nhiên, có vẻ như nó đã không phát triển đầy đủ như một phần mềm doanh nghiệp. Do đó, rất nhiều doanh nghiệp vẫn còn có một số lo ngại về việc sử dụng [GitHub](https://translate.googleusercontent.com/translate_c?depth=1&hl=vi&prev=search&rurl=translate.google.com.vn&sl=en&sp=nmt4&u=https://usersnap.com/github&usg=ALkJrhiyALSmNEa7NPacrpPokKJyyYq15Q) để sử dụng chuyên nghiệp

**Bảng 2.1 So sánh GitHub và GitLab**

| **Tính năng**  **đặc điểm** | **GitLab** | **GitHub** |
| --- | --- | --- |
| Phát hành | Tháng 9 năm 2011 | Tháng 4 năm 2008 |
| Giá | Kho chứa công cộng và tư nhân không giới hạn / cộng tác công cộng và tư nhân không giới hạn | Miễn phí cho các kho lưu trữ công cộng / Gói thanh toán cho các kho riêng |
| Tính năng xem xét mã | Có | Có |
| Theo dõi lỗi và vấn đề | Có | Có |
| Chi nhánh riêng | Có | Có (với kế hoạch trả tiền) |
| Xây dựng hệ thống | Có | Có (với dịch vụ của bên thứ ba) |
| Tự lưu trữ | Có | Có (với kế hoạch doanh nghiệp) |

***Về phía người dùng***

GitLab vẫn đứng sau GitHub và có một chặng đường dài để đi nếu muốn cạnh tranh với GitHub trên mức người dùng. Tuy nhiên, GitLab là một lựa chọn thay thế tốt nhất (và có thể là một trong những lựa chọn tốt nhất) cho GitHub.

# TỔNG KẾT

Trong bài này đã phân tích rõ những gì mà Github đã làm được, cách cài đặt, tạo tài khoản, thiêt lập cấu hình để sử dụng Github. Với sự phát triển của công nghệ hiện nay thì Github không còn xa lạ với chúng ta, Github kết nối mọi lập trình viên trên thế giới, tạo sự thuận tiện tốt nhất trong công việc, ngoài ra còn giúp đảm bảo an toàn cho phần mềm đang phát triển bằng việc có thể phục hồi lại các phiên bản trước nếu phiên bản phát triển hiện tại gặp sự cố ngoài ý muốn. Vậy nên muốn trở thành một developer chuyên nghiệp thì nên tìm hiểu về Github.

# TÀI LIỆU THAM KHẢO

[1]. Ngọc Lục, 2016. *Hướng dẫn sử dụng Git và GitHub cơ bản.* <https://lucngoc.com>, 18/10/2017.

[2]. Thành Phạm, 2017. *Hướng dẫn dung GitHub trên Windows cơ bản*. <http://daynhauhoc.com>, 18/4/2017.