**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

--🙡⯎🙣--

**KHOA CÔNG NGHỆ THÔNG TIN**



**NỘI DUNG ĐỀ TÀI:**

**TÌM HIỂU VỀ GITHUB**

**⯎Giảng viên: Nguyễn Lê Viết Tuấn**

**⯎Nhóm 12:**

**1) Đoàn Tấn Phát 1351010094 (Nhóm trưởng)**

**2) Mai Lê Nhật Khánh 121c652038**

**3) Nguyễn Thành Nam 1351010082**

**4) Lê Ngọc Tỷ 1351010138**

**Mục lục**

[I. TÌM HIỂU VỀ GITHUB 2](#_Toc445298779)

[1. Tìm hiểu đôi nét về Github 2](#_Toc445298780)

[- Lời nói đầu: 2](#_Toc445298781)

[2.1) Git là gì? 3](#_Toc445298782)

[II. GITHUB VÀ NHỮNG THIẾT LẬP BAN ĐẦU: 6](#_Toc445298783)

[2. Tạo GitHub Repository 14](#_Toc445298784)

[3. Kết nối GitHub và GitHub Desktop 16](#_Toc445298785)

# TÌM HIỂU VỀ GITHUB

## Tìm hiểu đôi nét về Github

### - Lời nói đầu:

-Một câu hỏi mà nhiều người đặt ra khi nhìn thấy serie này đó là Git là cái gì? Nó là ngôn ngữ lập trình gì chăng? Hay một loại thức ăn nào đó? Một kỹ thuật thiết kế website sao? Hoàn toàn không, bạn sẽ hiểu được Git là cái gì sau khi đọc hết từng chữ trong bài viết này.

-Trước khi nói qua về Git, mình có một câu chuyện ngắn muốn kể cho các bạn như sau.

*Có hai anh chàng lập trình viên đang hợp tác với nhau cùng xây dựng một phần mềm cho khách hàng. Dĩ nhiên hai anh chàng này sẽ làm việc với nhau bằng cách mỗi người tự viết code một ít và gửi cho nhau qua email hoặc một phương thức gửi dữ liệu thông thường nào đó. Mỗi người sau khi nhận được code của nhau sẽ tiến hành tự xem và sửa lại, sau đó cùng nhau ráp nối vào phần mềm mà họ cần làm. Tuy nhiên, một hôm anh A bỗng nhiên vô tình viết code sai vào phần mềm dẫn tới sản phẩm bị lỗi mà trước đó cả hai anh đều không hề sao lưu lại do phần mềm quá lớn. Và thế là hai anh lại cùng nhau xây dựng lại tính năng bị lỗi và tiếp tục gửi cho nhau như vậy.*

-Thế bạn có thấy vấn đề gì xảy ra ở câu chuyện trên không? Mình thì mình thấy có hai vấn đề như sau:

1. Việc tự code riêng và gửi cho nhau qua email rất mất thời gian của nhau. Giá như anh A có thể chủ động xem những thay đổi của anh B từ xa và tiến hành gộp trực tiếp những thay đổi của anh B vào sản phẩm.
2. Việc sửa code mà không sao lưu khiến cho họ phải viết lại code từ đầu khi phần mềm bị lỗi.

⇨Và hai vấn đề trên Git có thể sẽ giúp hai anh chàng ấy giải quyết nhanh gọn lẹ và đơn giản hơn rất nhiều. Ngoài ra Git còn làm nhiều việc hơn nữa mà chỉ có thể sử dụng bạn mới hiểu được.

***2.Phân biệt giữa Git và Github:***

### 2.1) 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**. Đại loại là như vậy.



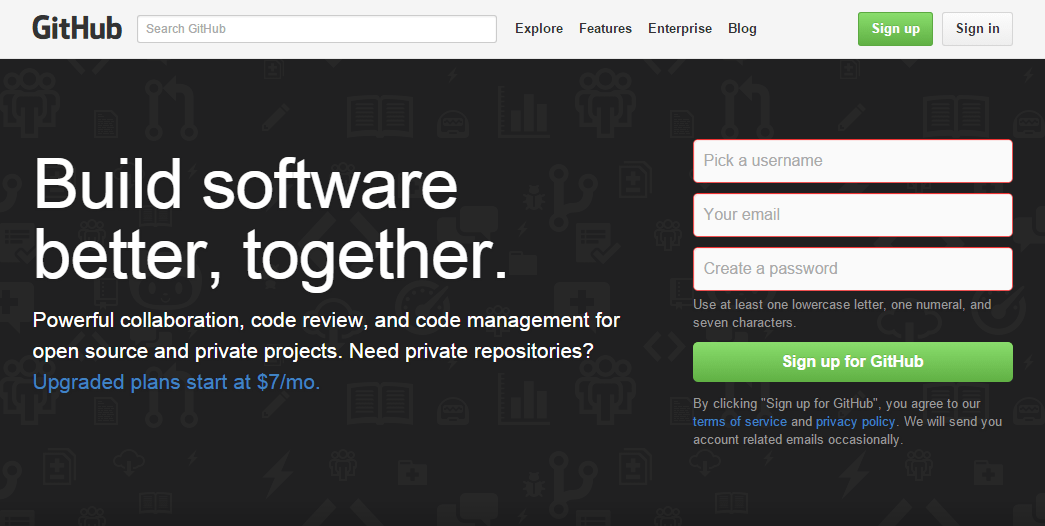
(*Mô hình hoạt động của DVCS*)

-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.

***2.2) Github là gì?***

-Mình biết là có rất nhiều bạn khi nghe nói đến Git sẽ nghĩ ngay đến [Github](https://github.com/) và có thể sẽ có một số hiểu lầm với họ. Cũng xin nhắc lại rằng, Git là tên gọi của một mô hình hệ thống. Như mình đã giải thích ở trên, **các máy tính có thể clone lại mã nguồn từ một repository** và **Github chính là một dịch vụ máy chủ 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.



(*Github là một dịch vụ máy chủ Repository*)

***3. Tại sao nên sử dụng Github:***

-Có rất nhiều lợi thế để bạn nên sử dụng Git trong việc lập trình ngay từ hôm nay, bất kể là lập trình cái gì đi chăng nữa.

* Git 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).
* 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 deployment sản phẩm.
* Và nhiều hơn thế nữa.

## II. GITHUB VÀ NHỮNG THIẾT LẬP BAN ĐẦU:

***1.Nguyên tắc hoạt động:***

-Để làm việc với một Github bạn cần:

1.Đăng ký một tài khoản GitHub và tạo một Repository (GitHub Repository).

2.Cài đặt GitHub Desktop, một công cụ trực quan quản lý Local Repository (Kho chứa dữ liệu địa phương).

3.Cấu hình để có thể đồng bộ hóa dữ liệu bằng GitHub Desktop lên Repository server.

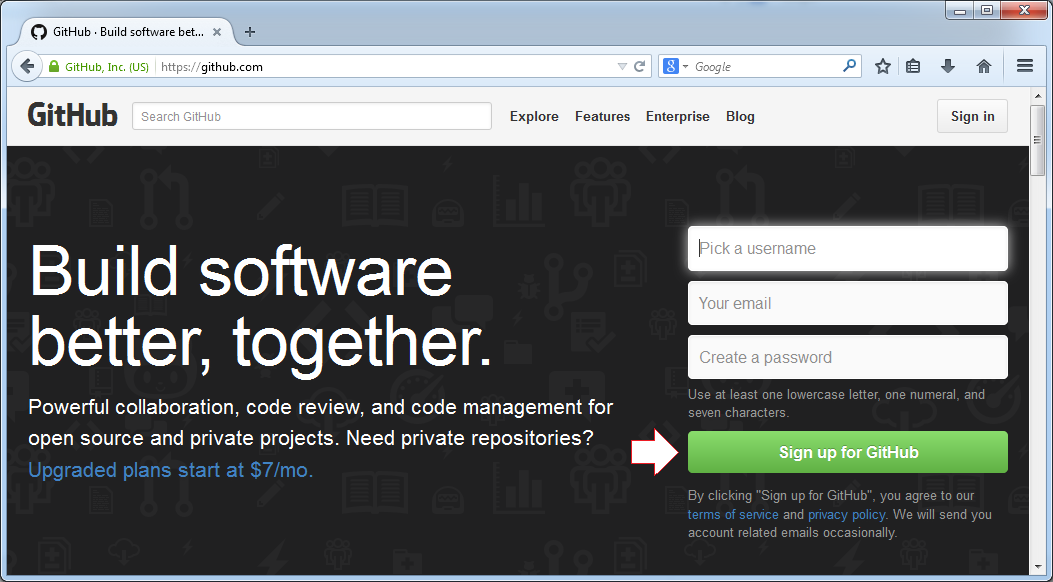
-Hình minh họa dưới đây:



**2. Đăng ký tài khoản Github:**

Trước hết bạn cần phải đăng ký miễn phí một tài khoản GitHub. Bạn có thể vào trang chủ của GitHut tại:

[https://github.com](https://github.com/)



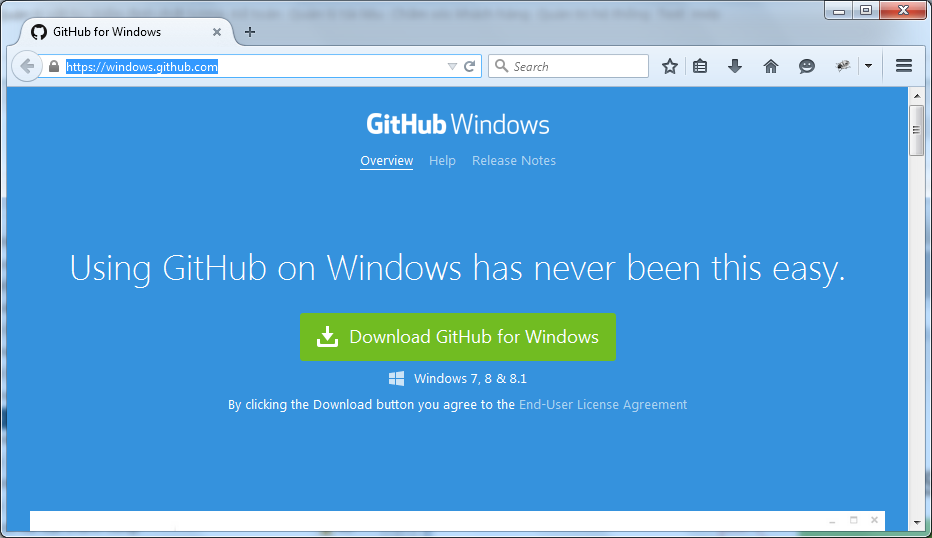
-Việc đăng ký một tài khoản là đơn giản, bạn chỉ cần nhậpusername và password và địa chỉ email. Sau khi đăng ký xong bạn cần vào Email kích hoạt tài khoản. Mọi việc hoàn thành.

**3. Download & cài đặt GitHub Desktop**

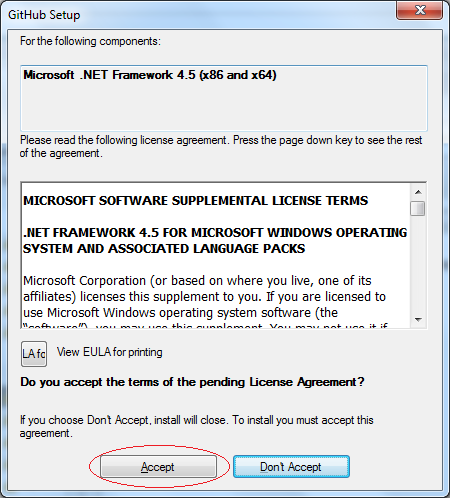
**-GitHub Desktop** về bản chất là một công cụ trực quan cho phép bạn quản lý **Local Repository** (Kho chứa địa phương) trên máy tính của bạn.

-Để download GitHub Desktop bạn vào địa chỉ:

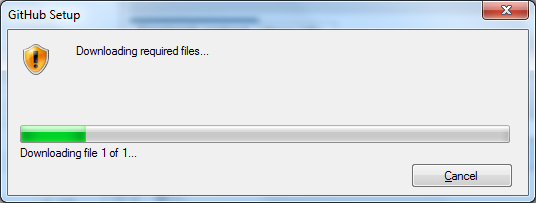
<https://windows.github.com/>



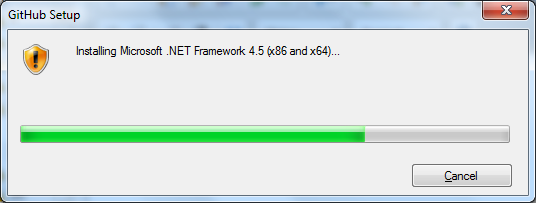
-Sau khi download xong, bạn cần cài đặt **GitHub Desktop** vào máy tính



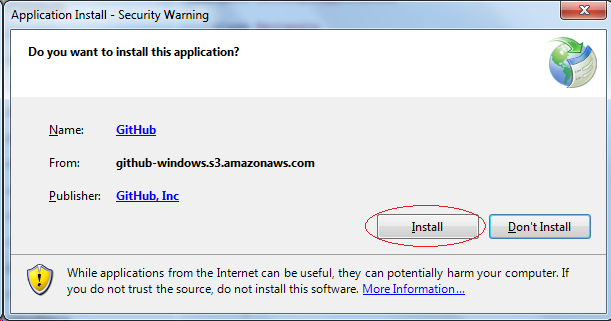
-Bộ cài đặt sẽ download và cài đặt thư viện bắt buộc Microsoft .Net.

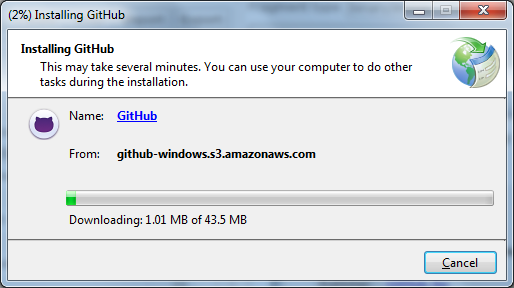


-Chờ cho tới khi tiến trình download hoàn thành. Gói thư viện .Net sẽ được cài đặt vào máy tính của bạn.

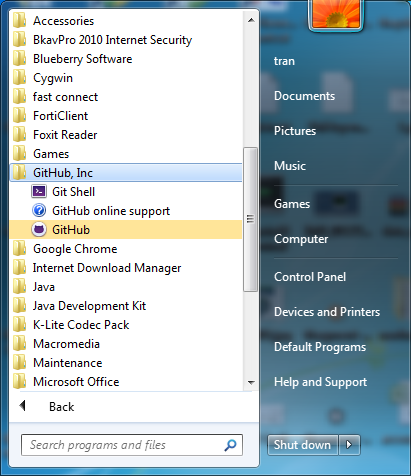


-Bộ cài đặt tiếp tục cài đặt GitHub Desktop:





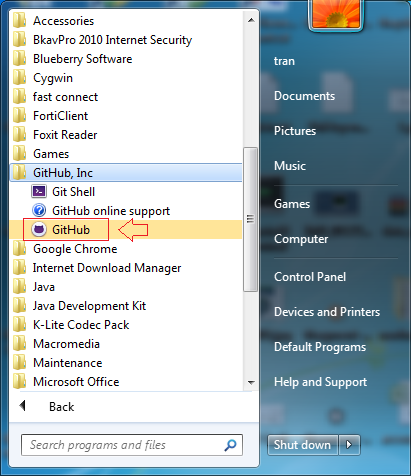
-GitHub đã được cài đặt thành công.



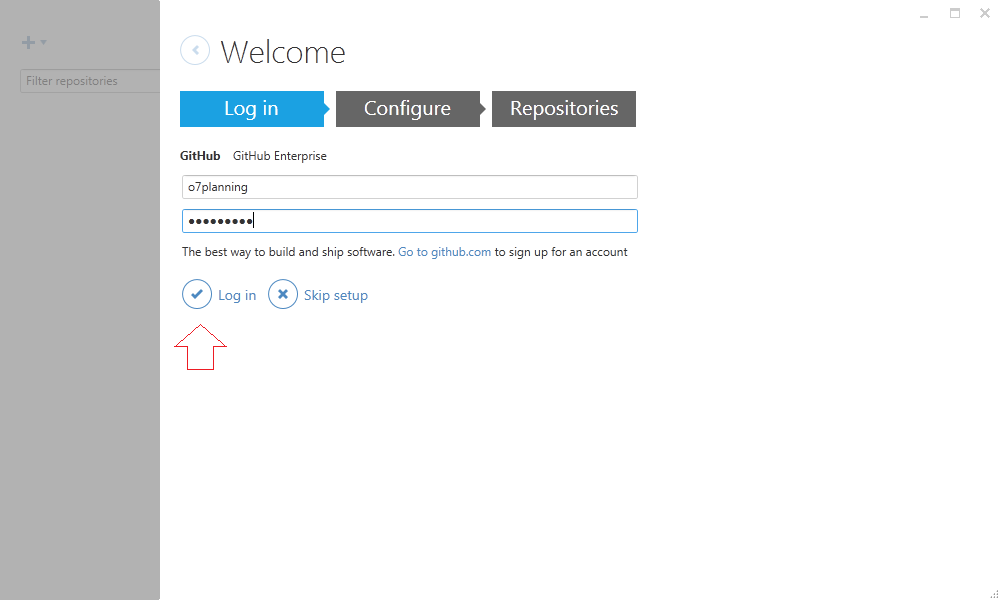
**III.GITHUB DESKTOP:**

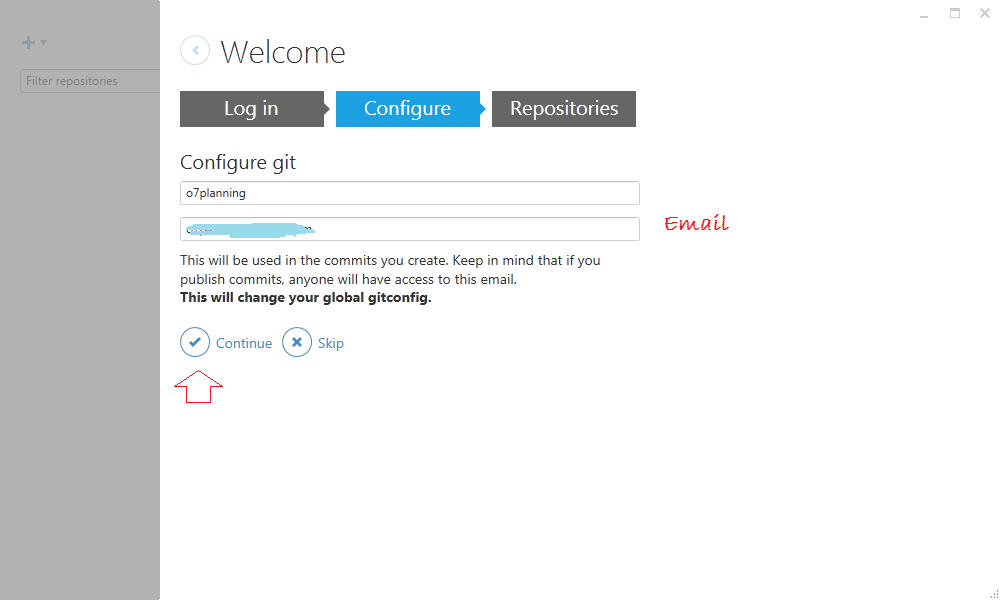
**1.Chạy Github Desktop:**

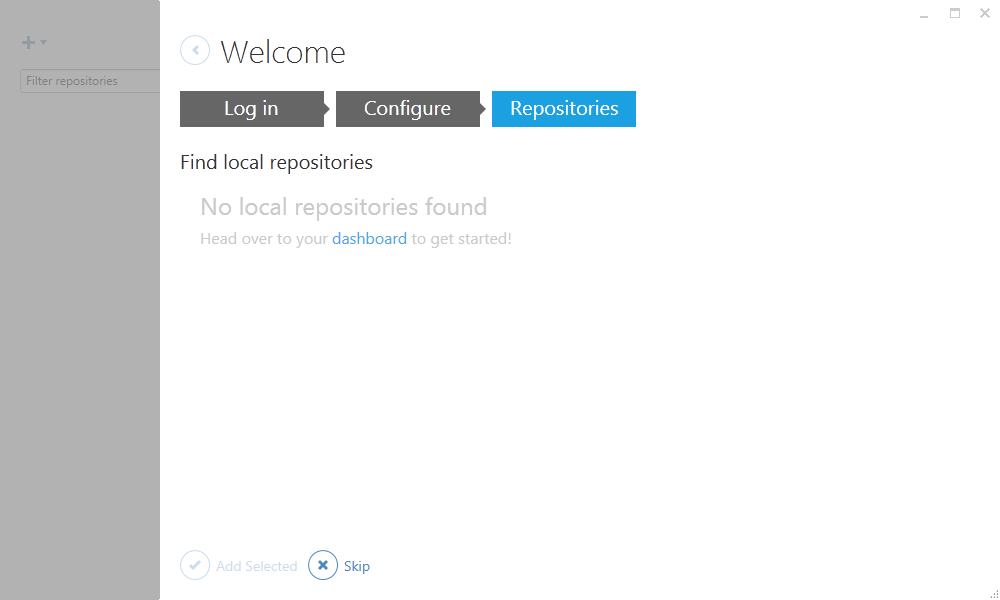
-Sau khi cài đặt thành công:



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



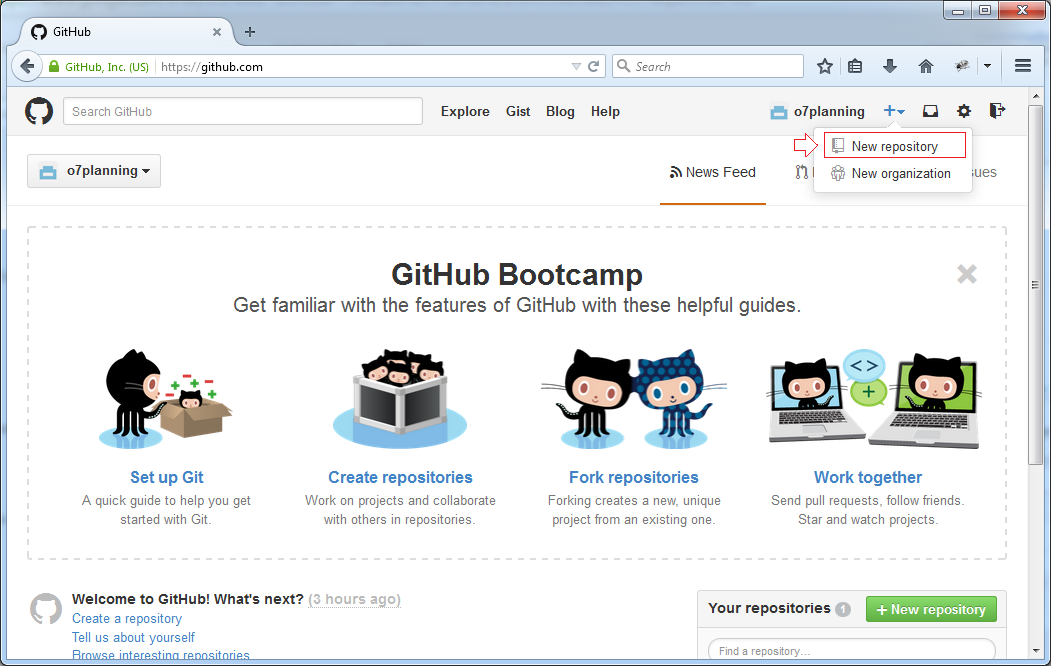


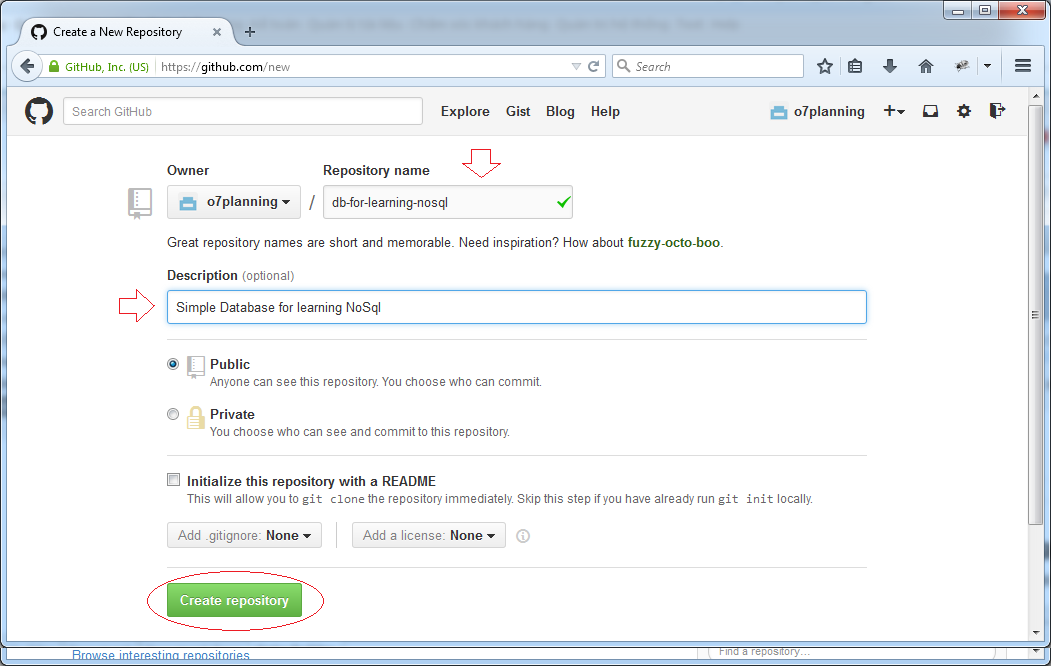


-Cho tới lúc này trên máy tính của bạn chưa có một **Local Repository** nào.

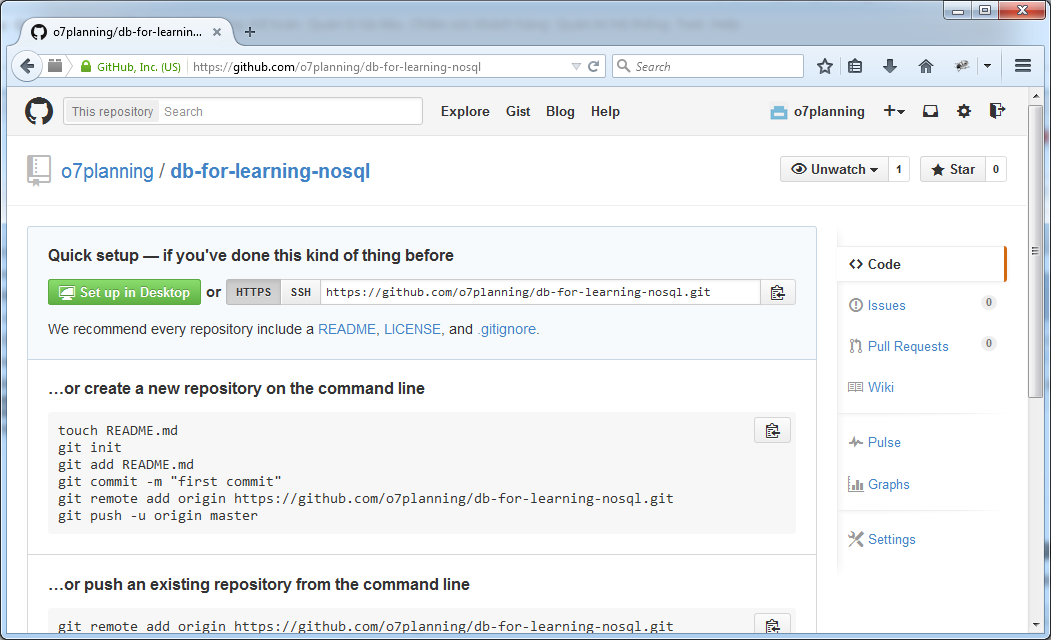
# 2. Tạo GitHub Repository

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





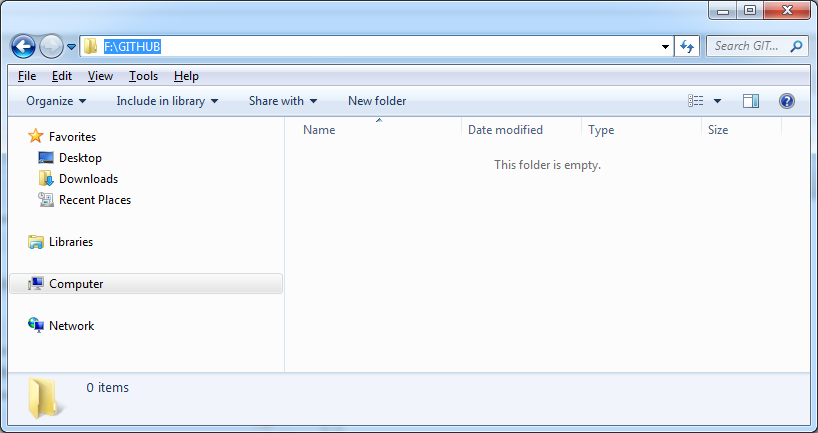
-Repository đã được tạo ra.



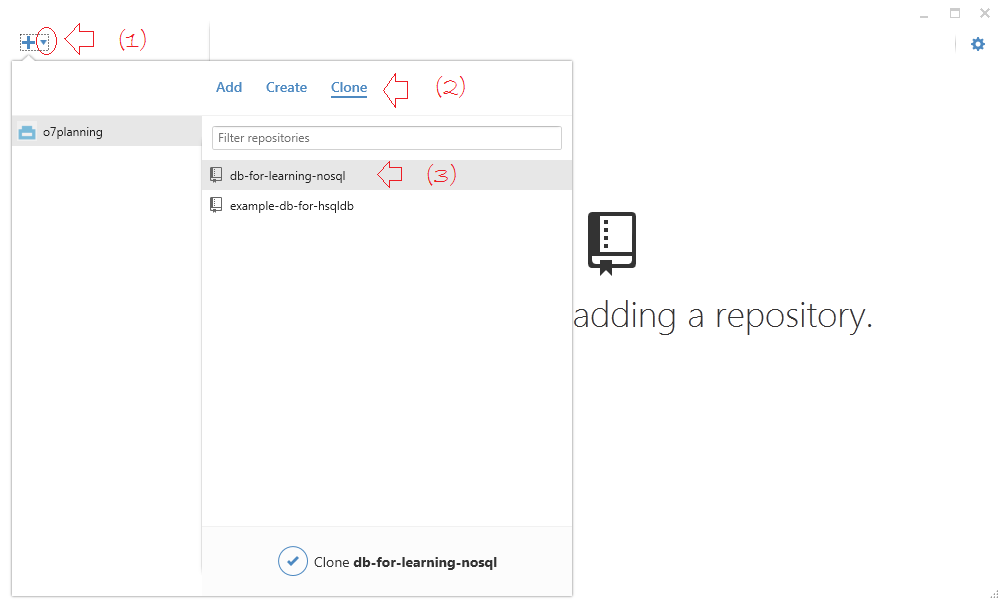
# 3. Kết nối GitHub và GitHub Desktop

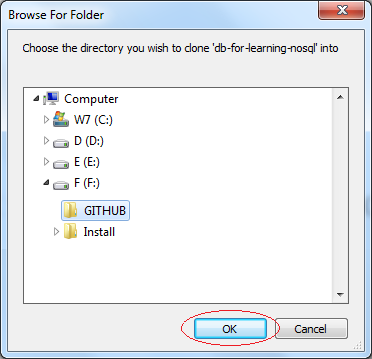
-Trước hết cần chọn một thư mục rỗng để làm vị trí chứa dữ liệu địa phương. Chẳng hạn:

F:/GITHUB

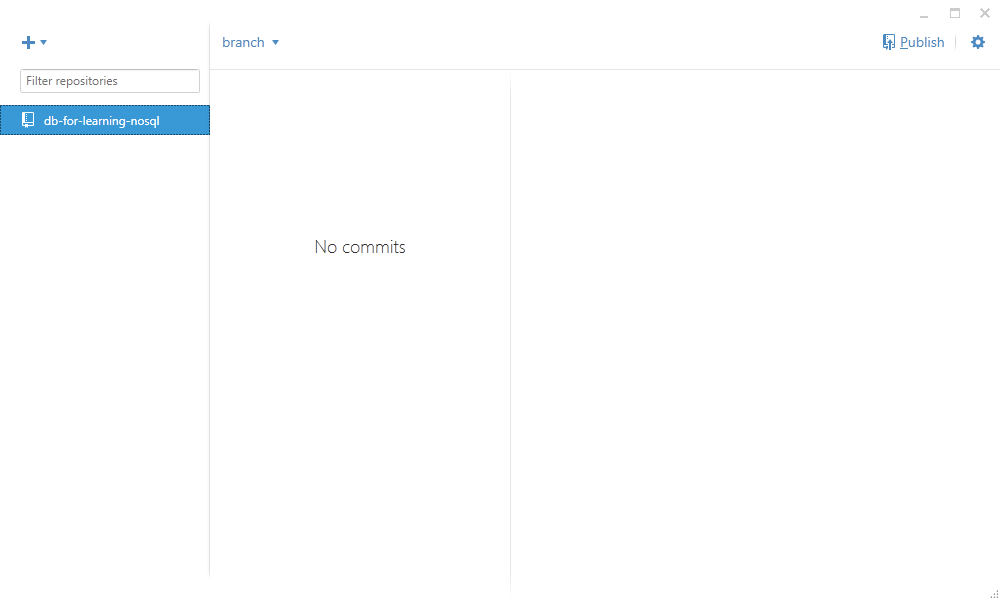


-Trên GitHub Desktop, lựa chọn một Repository bạn đã tạo trên GitHub để Clone thành một bản ở máy tính địa phương của bạn.

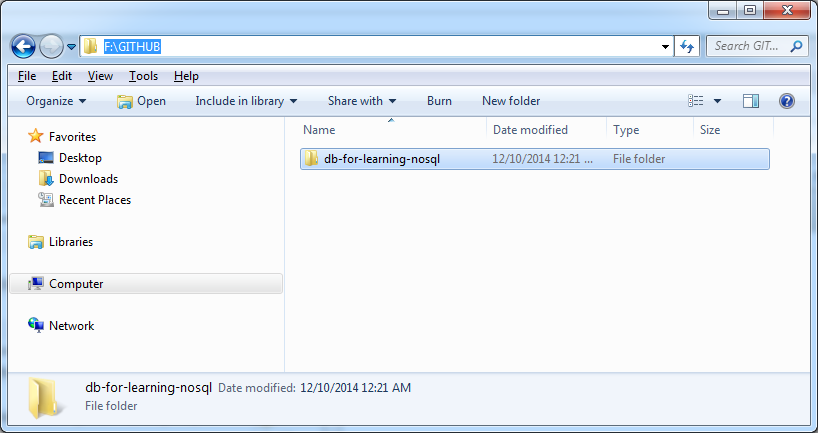




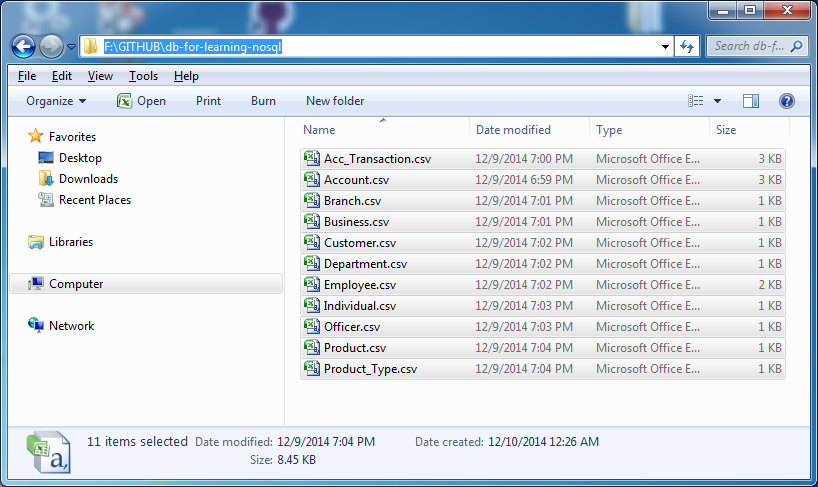
-Lúc này trên **GitHub Desktop** bạn sẽ thấy một **Local Repository** đã được tạo ra.



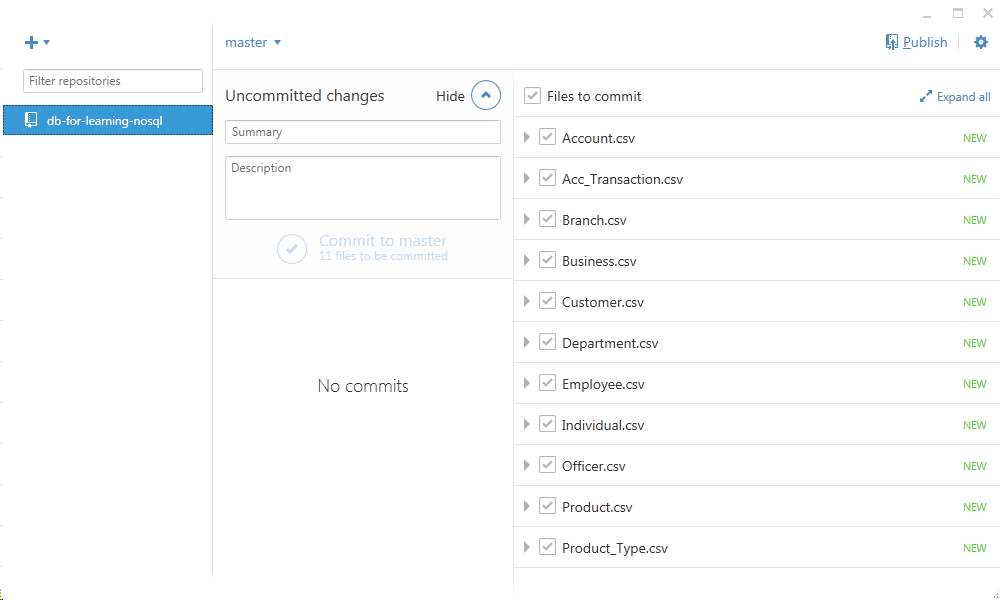
-Và trên thư mục tại ổ cứng, một thư mục con đã được tạo ra:



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

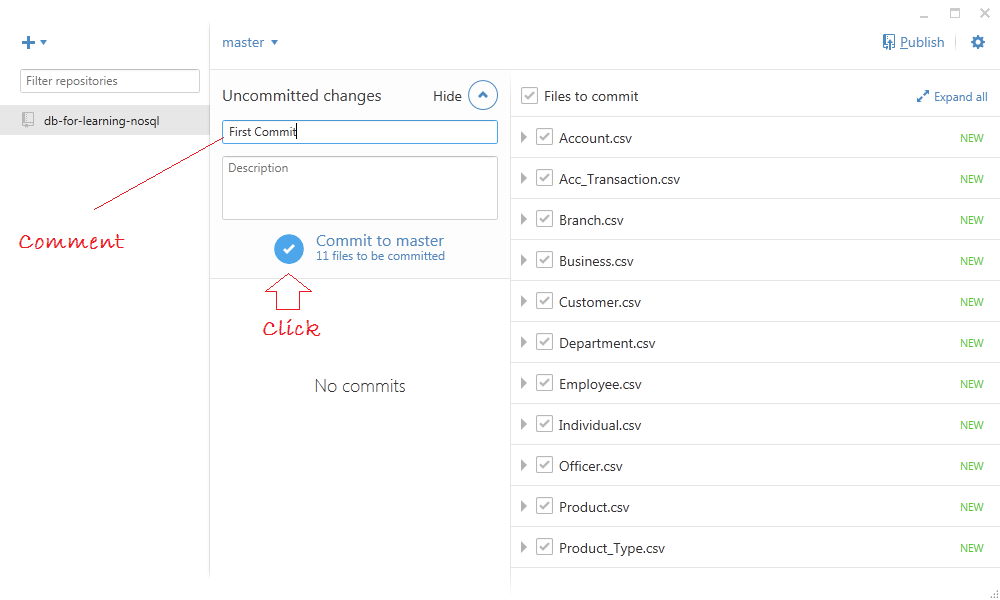


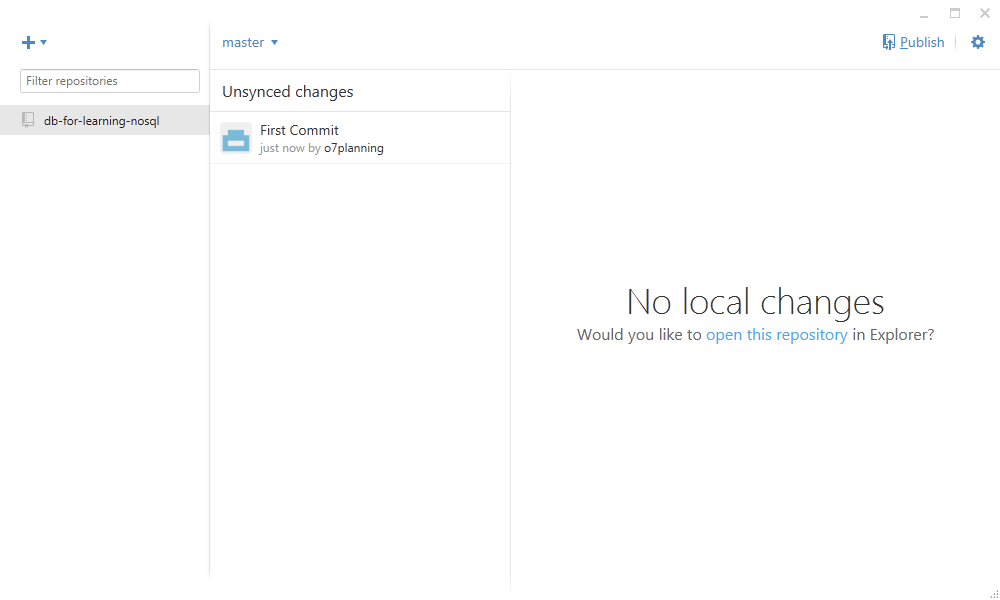
-**GitHub Desktop** ngay lập tức nhận biết được các thay đổi tại **Local Repository**.



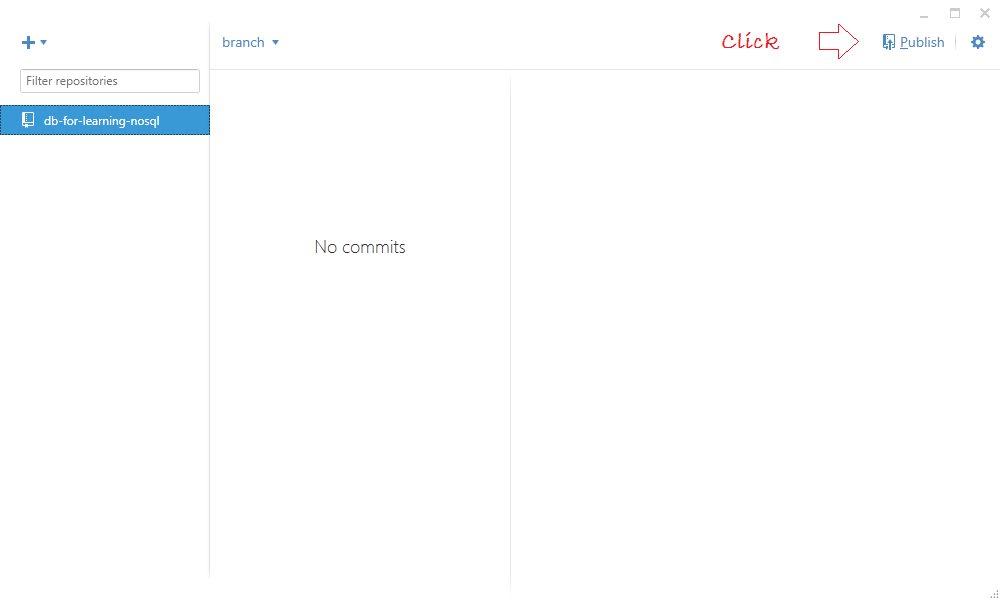
-

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

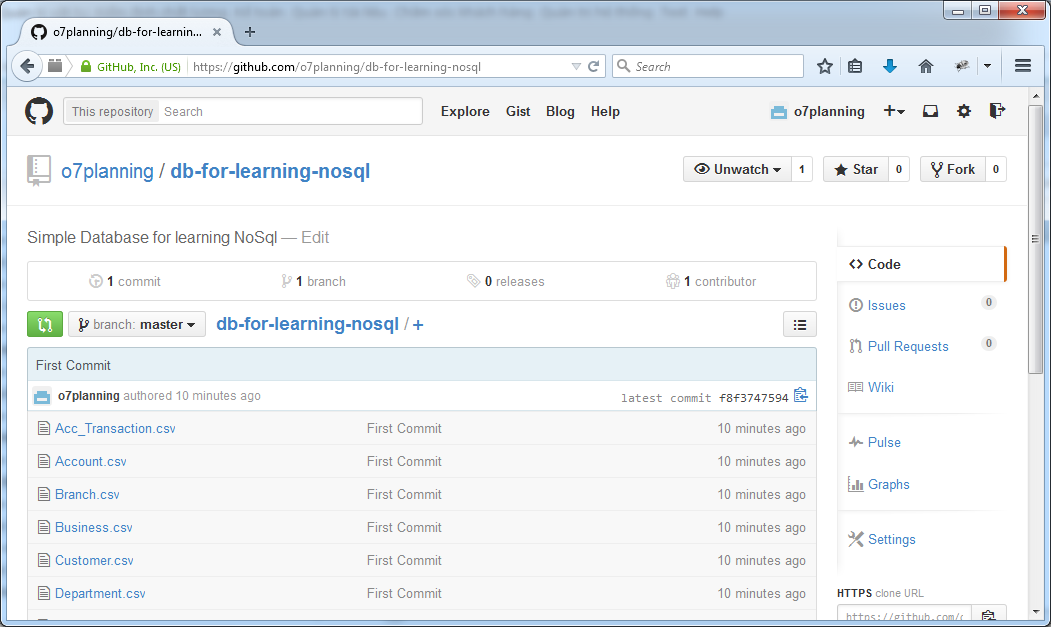




-Cuối cùng là công khai cho mọi người có thể xem:



-Các file dữ liệu bạn có thể nhìn thấy trên Server.



BỔ SUNG

**How to use GitHub (dịch từ trang chủ)**

Github là một nền tản lưu trữ chung nhằm điều khiển và hợp tác. Cho phép bạn và những người khác cùng làm việc trên một dự án (chủ yếu là dự án phần mềm) từ bất kể nơi nào.

Tạo kho lưu trữ

Kho lưu trữ có thể chứa các folders, files, videos, các bộ dữ liệu và bảng tính - bất cú thứ gì mà dự án của bạn cần.

B1: Trên góc phải ở trên ấn vào dấu '+' sau do ấn New repository

B2: Đặt tên cho kho lưu trữ của bạn ở Repository name

B3: Thêm 1 số thông tin miêu tả kho dữ liệu trong phần Description

B4: Chọn giữa tạo kho lưu trữ ở mức public hay private

- Public: ai cũng có thể xem đc, dễ dàng nhận đc sự giúp đỡ của cộng đồng

- Private: Chỉ cho phép bạn và những người bạn chọn để hợp tác có thể truy cập

B5: Tích vào dấu chọn "Initialize this repository with a README." để tạo 1 file README chứa các thông tin về kho lưu trữ này

B6: Ấn Create repository

-----------------------------------

Tạo các nhánh.

Các nhánh là cách làm việc với nhiều phiên bản của một kho lưu trữ tại cùng 1 thời điểm.

Mặc định, kho lưu trữ của bạn có 1 nhánh master, đc xem là nhánh chính. Chúng ta sẽ dùng các nhánh khác để thử nghiệm các thay đổi đối với dự án trước khi hợp nhất nó với nhánh master.

Khi bạn tạo 1 nhánh mới từ nhánh master, bạn đã tạo ra 1 bản copy của nhánh master tạo thời điểm đó. Nếu như đồng nghiệp cập nhật những thay đổi lên nhánh master khi bạn đang làm việc trên một nhánh riêng, bạn có thẻ cập nhật những thay đổi đó về nhánh mình đang làm việc.

B1: truy cập vào kho lưu trữ hiện tại

B2: Click vào branch: master

B3: Tại phần branch name, gõ tên nhánh mới vào

B4: Ấn enter hay chọn dòng Create branch

----------------------------------

Cập nhật thay đổi trên github.

Trên github, mỗi lần cập nhật thay đổi thường đi kèm với một mẫu tin miêu tả lý do những thay đổi này đc cập nhật. Mẫu tin cập nhật có lưu lại lịch sử cập nhật của bạn để những người khác có thể thấy và hiểu những thứ bạn làm.

B1: Mở file README.md

B2: Click icon bút ở bên phải góc trên để cập nhật thay đổi

B3: Trong phan editor, viết các thay đổi

B4: Viết mấu tin cập nhật miêu tả các thay đổi mình vừa cập nhật

B5: Click Commit changes

P/s: Các thay đổi này chỉ nằm ở nhánh readme-edit, nên nhánh này sẽ có nội dung khác với nhánh master

-----------------------------------

Open a pull request

Pull request là cốt lõi của Github. Khi bạn open a pull request, bạn đề xuất những thay đổi và yêu cầu một người nào đó xem và pull in (đéo biêt dịch thế nào cho đúng) sự đóng góp và gộp vào nhánh của họ. Pull request cho thấy sự khác biệt giữa 2 nhánh. Sự khác biệt đc thể hiện qua màu đỏ(thêm) và xanh(thiếu).

Ngay khi bạn cập nhật, bạn có thể mở một pull request và bắt đầu thảo luận, kể cả trước khi hoàn thành code.

Bằng cách sử dụng GitHub’s @mention system trong mẫu tin pull request, bạn có thể yêu cầu phản hồi từ một người hoặc  một team xác định, ngay cả khi họ cách bạn 10m hay nữa vòng trái đất.

Bạn còn có thể open the pull request tại kho lưu trữ cảu chính mình và tự gộp chúng vào dự án của mình. Đây là một cách hay để học cách hoạt động của Github trước khi làm ciệc trên một dự án lớn.

B1: Click Pull Request tab, sau đó click new pull request.

B2: Chọn nhánh mà bạn tạo ra, so sánh với nhánh master.

B3: Xem xét sự khác nhau ở trang Compare, đảm bảo nội dung bạn muốn submit là chính xác.

B4: Click Create Pull request.

B5: Ghi tiêu đề và miêu tả những thay đổi mà bạn submit.

Tip: Bạn có thể them biểu tượng cảm xúc hoặ hình ảnh vào phần comment hay Pull Request, khiến Pull request cảu bạn thân thiện vào chi tiết hơn.

---------------------------------------

Gộp Pull request

B1: Click nút Merge pull request để gộp các thay đổi vào nhánh master.

B2: CLick Confirm mẻge

B3: Xóa nhánh chứa các thay đổi mà bạn làm vì h đây các thay đổi đó đã có trong nhánh master. Click nút Delete branch.