|  |
| --- |
| **Github** |
| *Introduction and Guides for Beginners* |

|  |  |
| --- | --- |
| Người lập văn bản : | Phê duyệt bởi: |
| *JS-Git Team* | *{Reviewer-FullName}* |

Fall Semester

29/11/2016

Contents

[I. Mở đầu 3](#_Toc468223365)

[A. Git là gì? 4](#_Toc468223366)

[B. Github là gì? 6](#_Toc468223367)

[C. Tại sao nên sử dụng Git? 6](#_Toc468223368)

[II. Hướng dẫn tạo tài khoản GitHub 8](#_Toc468223369)

# Mở 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 ngay sau đây.

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:

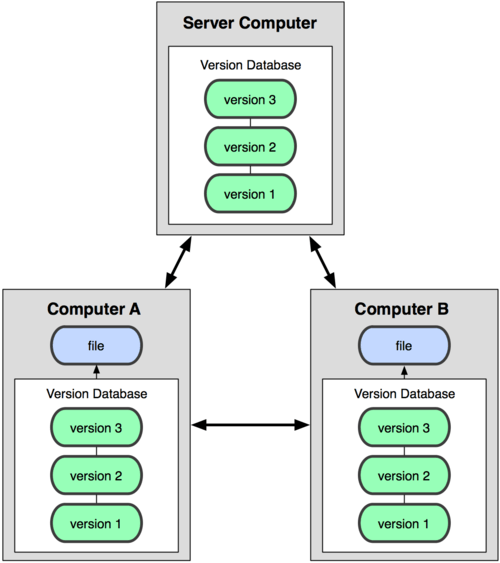
Việc tự code riêng và gửi cho nhau qua email rất mất thời gian của nhau. Giá như anh A có thể chủ động xem những thay đổi của anh B từ xa và tiến hành gộp trực tiếp những thay đổi của anh B vào sản phẩm.

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.

## 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.

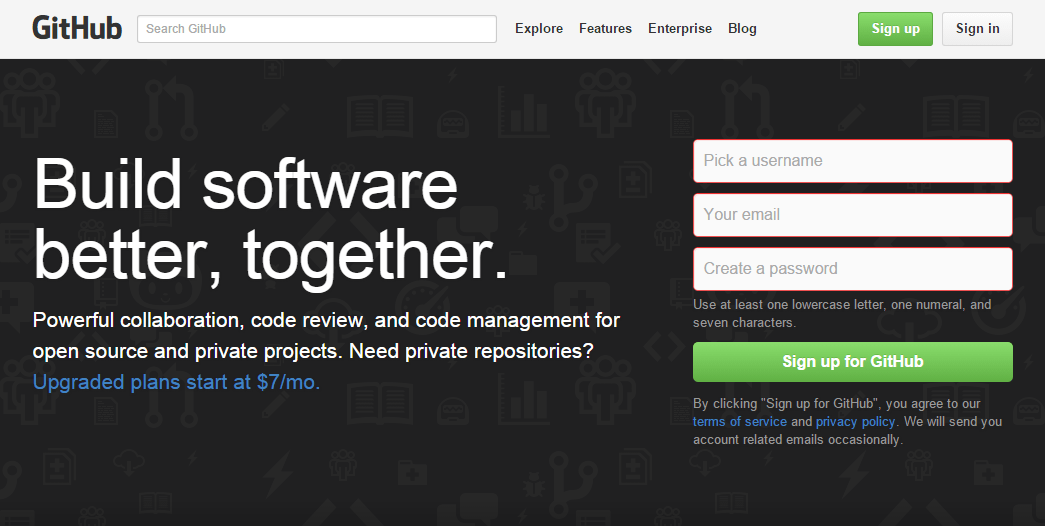


*Mô hình hoạt động của DVCS*

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.

## Github là gì?



*Github là một dịch vụ máy chủ Repository*

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.

Mặc dù Git có thể làm việc với bất kỳ trên máy chủ Linux nào nhưng để dễ hiểu và thực tế hơn, mình sẽ hướng dẫn các bạn sử dụng Git với Github trong suốt serie này.

## Tại sao nên sử dụng Git?

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.

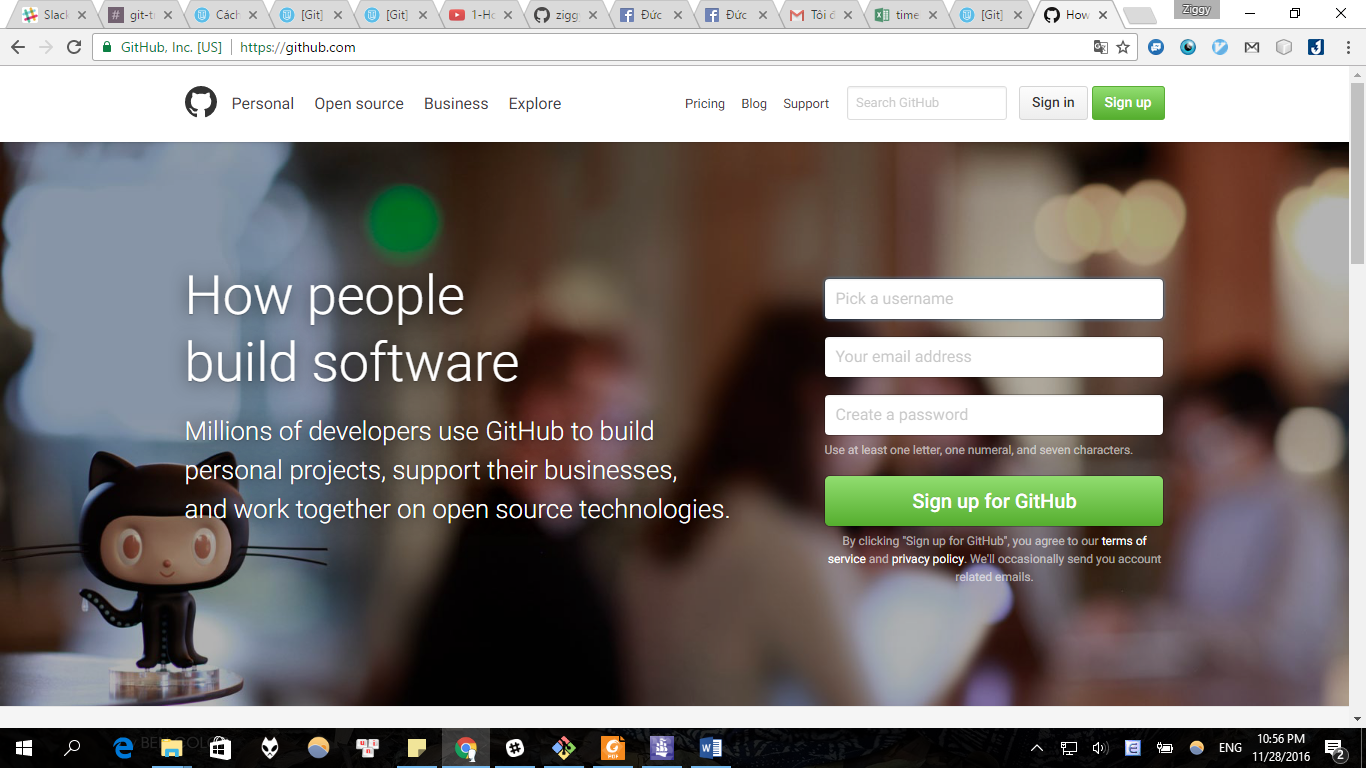
* Git dễ sử dụng, an toàn và nhanh chóng.
* Có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều bằng việc kết hợp các phân nhánh (branch).
* Bạn có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa.
* Dễ dàng trong việc deployment sản phẩm.
* Và nhiều hơn thế nữa.

Nếu bạn là một lập trình viên thì Git là một hệ thống bạn cần phải biết cách sử dụng, ít nhất là ngay từ bây giờ.

Nguồn : <https://thachpham.com/tools/git-git-va-github-la-gi-tai-sao-nen-dung.html>

# Hướng dẫn tạo tài khoản GitHub

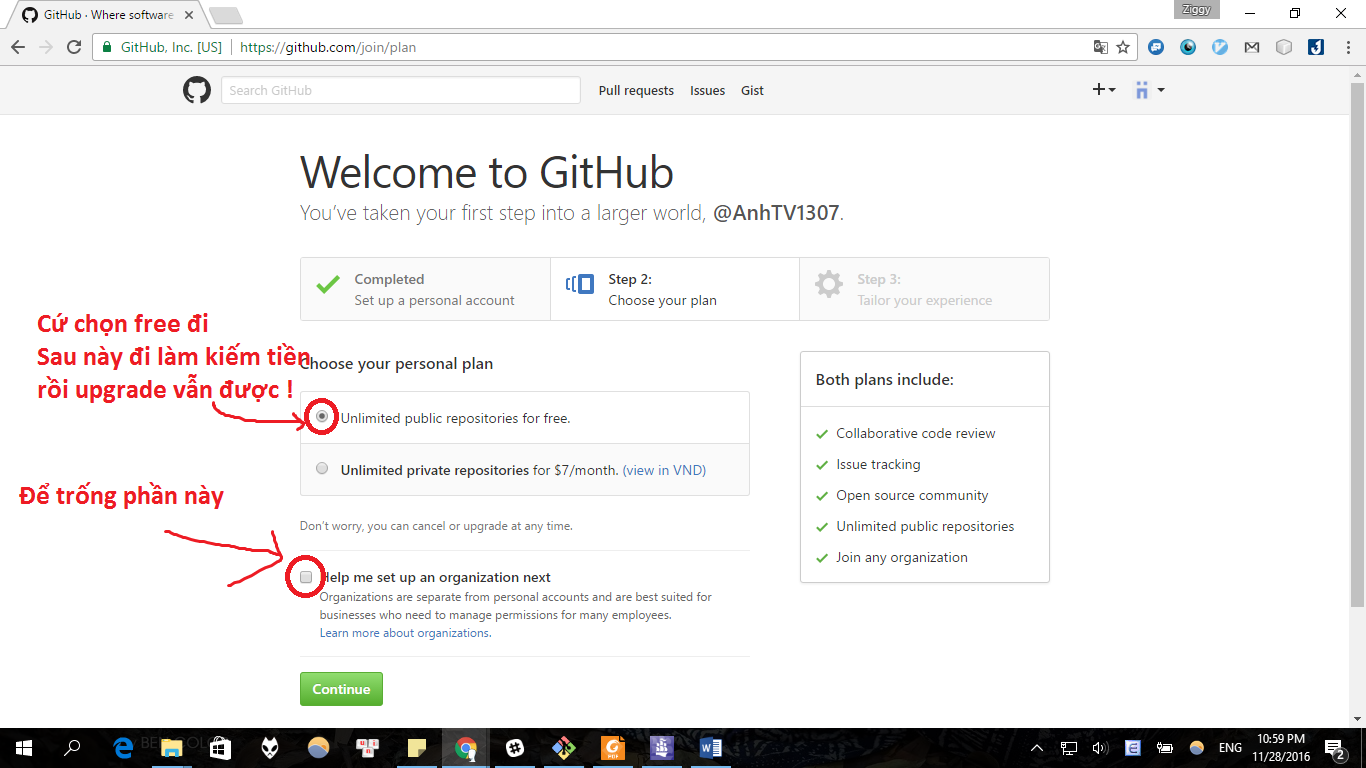
1. **Truy cập** github.com **và điền các thông tin cơ bản**



*Màn hình ban đầu của github.com*

* **Phần Username:** các bạn không cần mất quá nhiều thời gian để chọn tên đâu vì sau này có thể thay đổi được !

1. **Sau** khi khai báo thông tin cá nhân và xác nhận email **xong sẽ đến phần setting kiểu tài khoản**

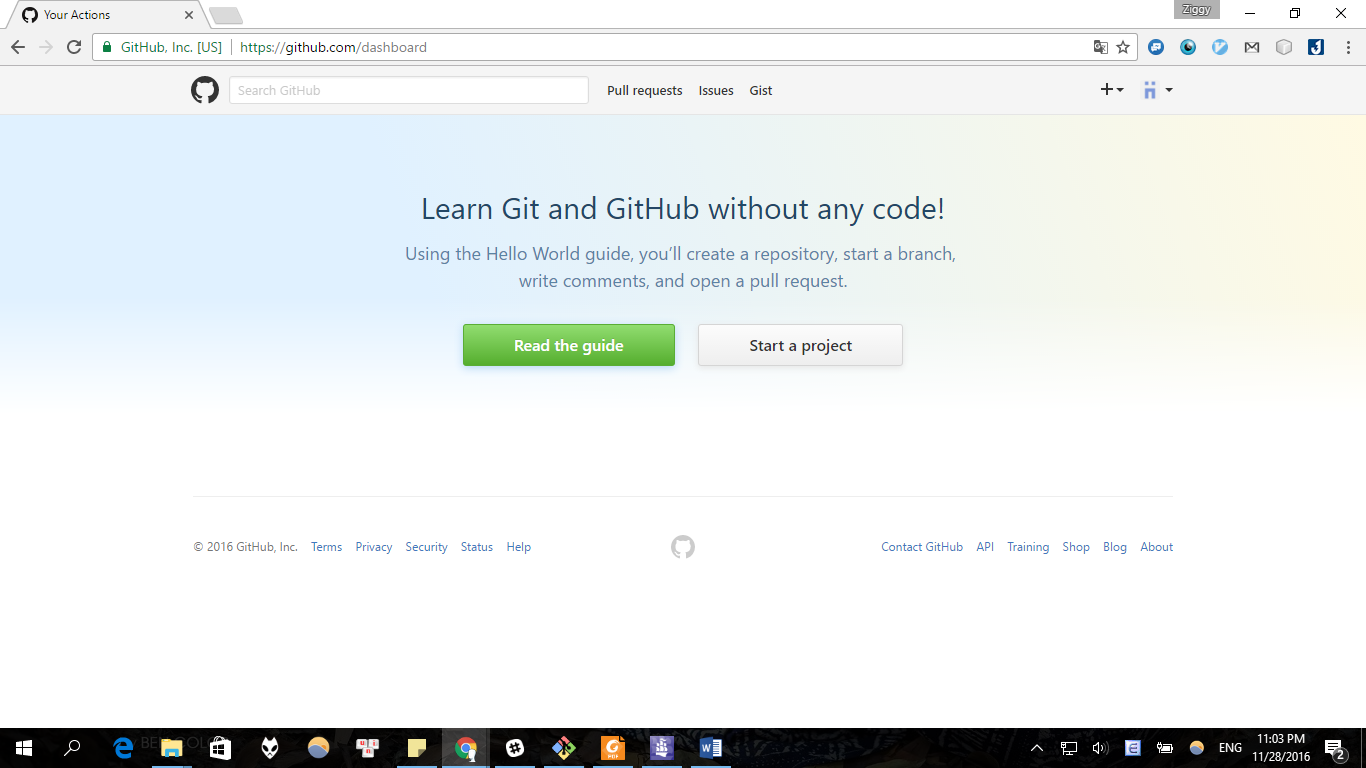


1. **Cho github biết thêm về bản thân bạn ( Skip cũng được)**

**

*Thông tin này sẽ có ích cho phần gọi ý và hướng dẫn riêng của github*

1. **Done !!!**

****

Màn hình chính của GitHub

# Những bước đầu với GitHub Desktop

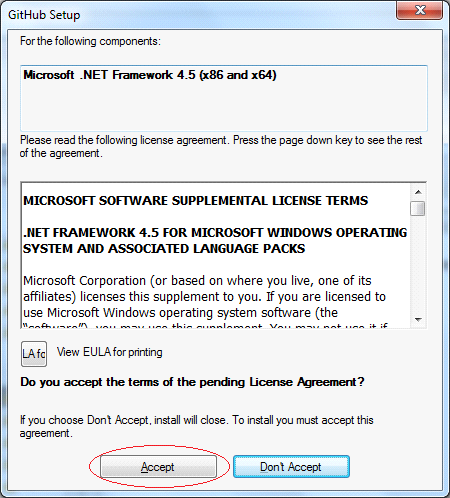
## Download & cài đặt GitHub Desktop

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.

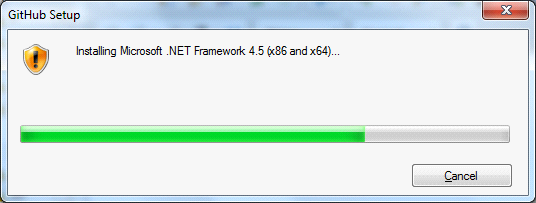
Để download GitHub Desktop bạn vào địa chỉ:

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

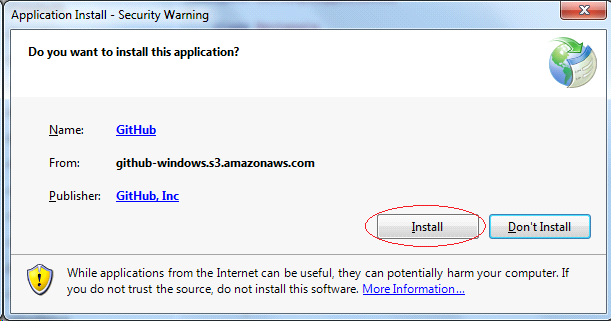
Sau khi download xong, bạn cần cài đặt GitHub Desktop vào máy tính:



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

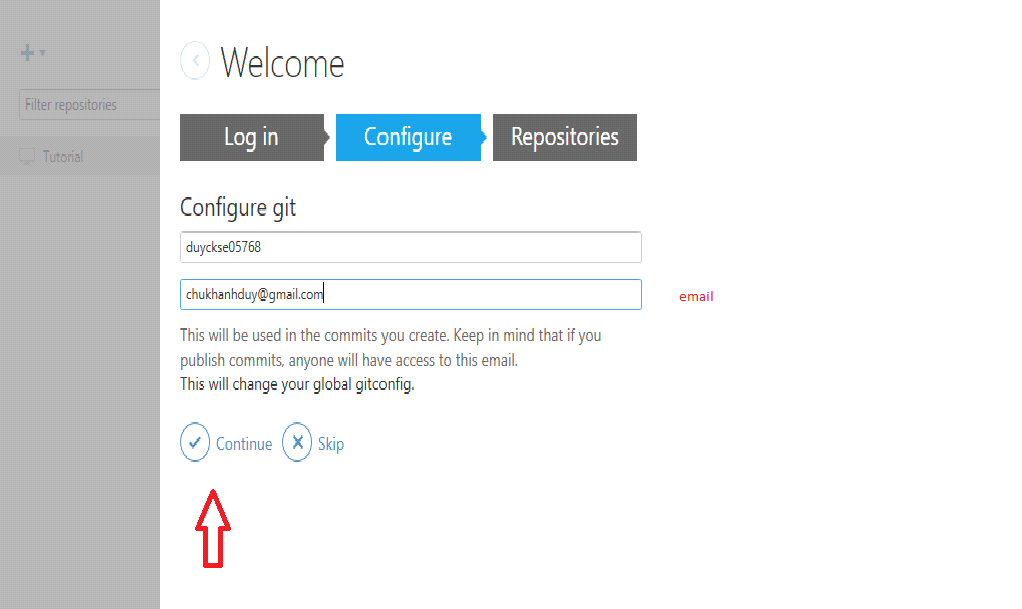
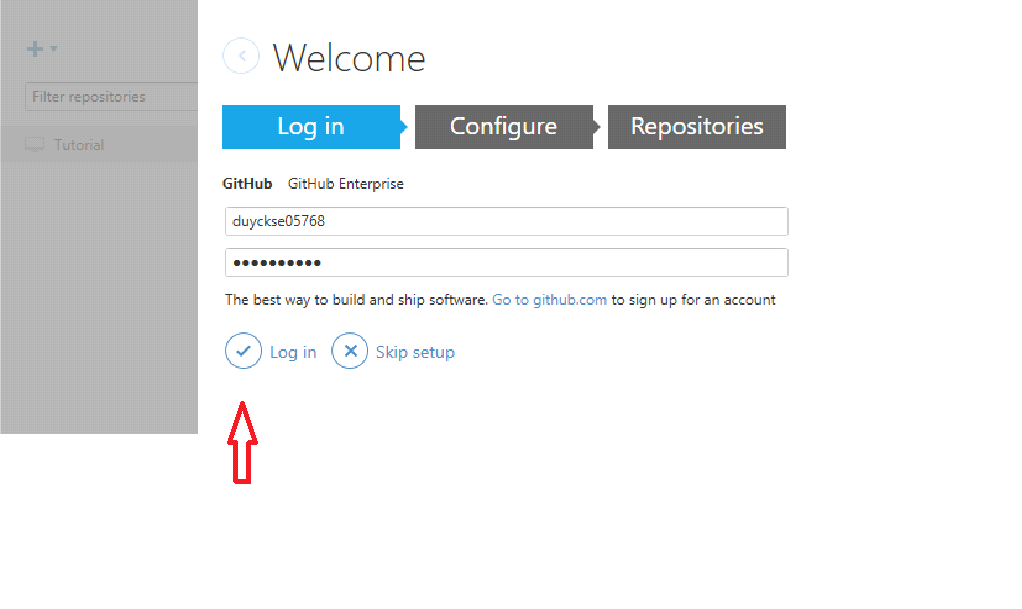


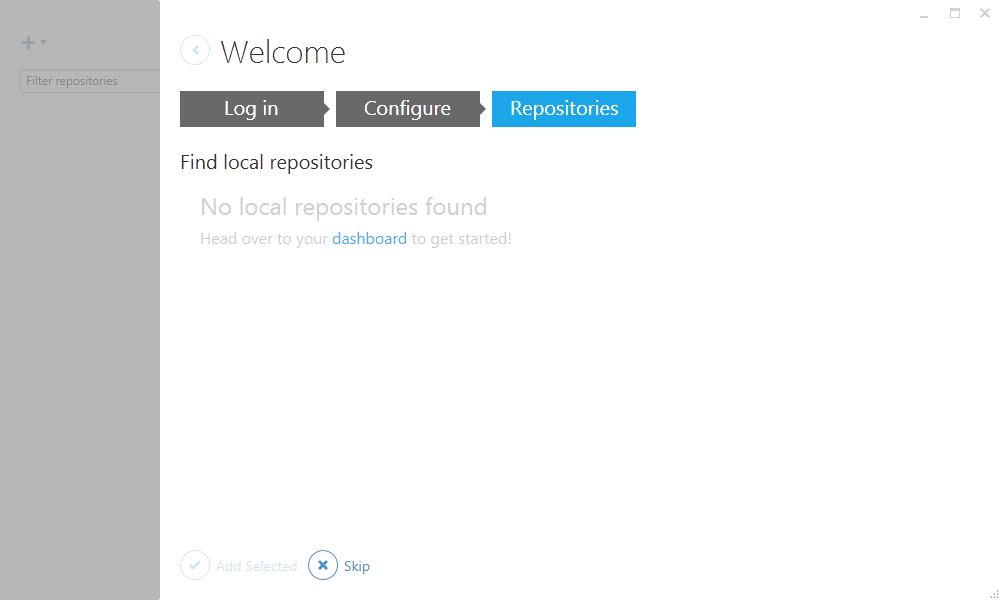
Bộ cài đặt tiếp tục cài đặt GitHub Desktop:



Đợi 1 lát và GitHub đã được cài đặt thành công

Sau đó chạy Github

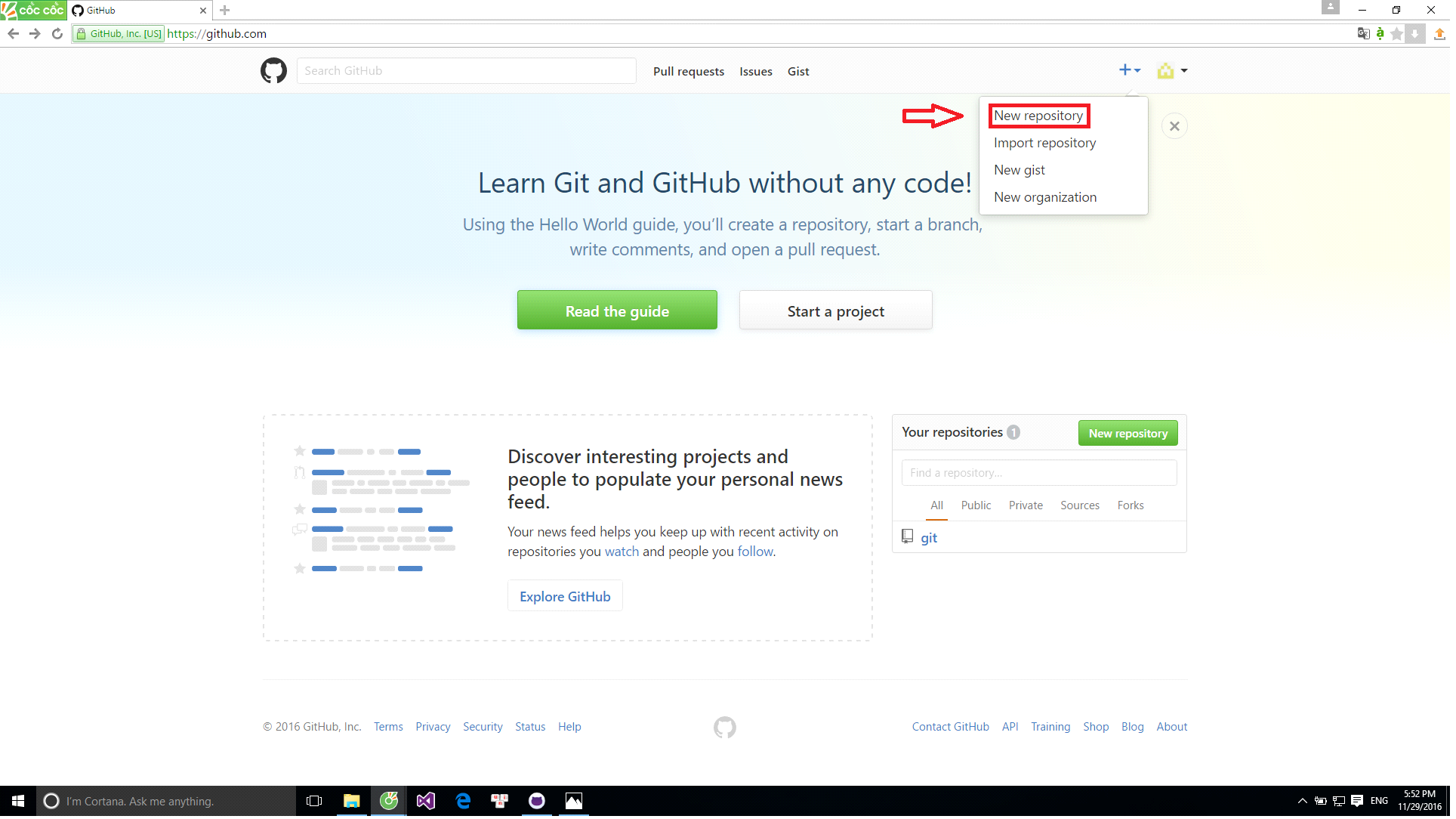
-Đăng nhập trên GitHub Desktop để kết nối vào tài khoản GitHub của bạn.



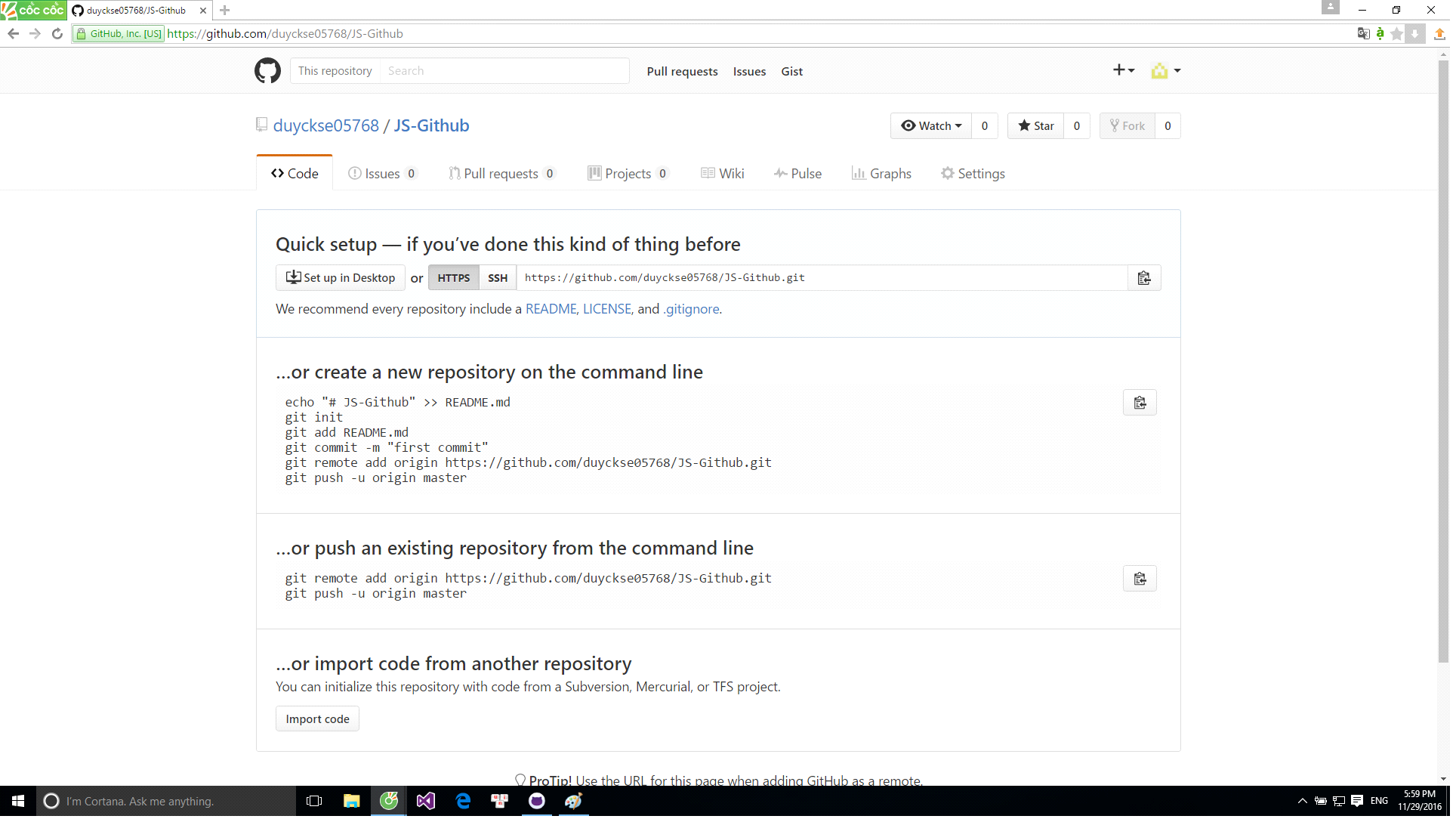
Cho tới lúc này trên máy tính của bạn chưa có một Local Repository nào.

## Tạo GitHub Repository

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







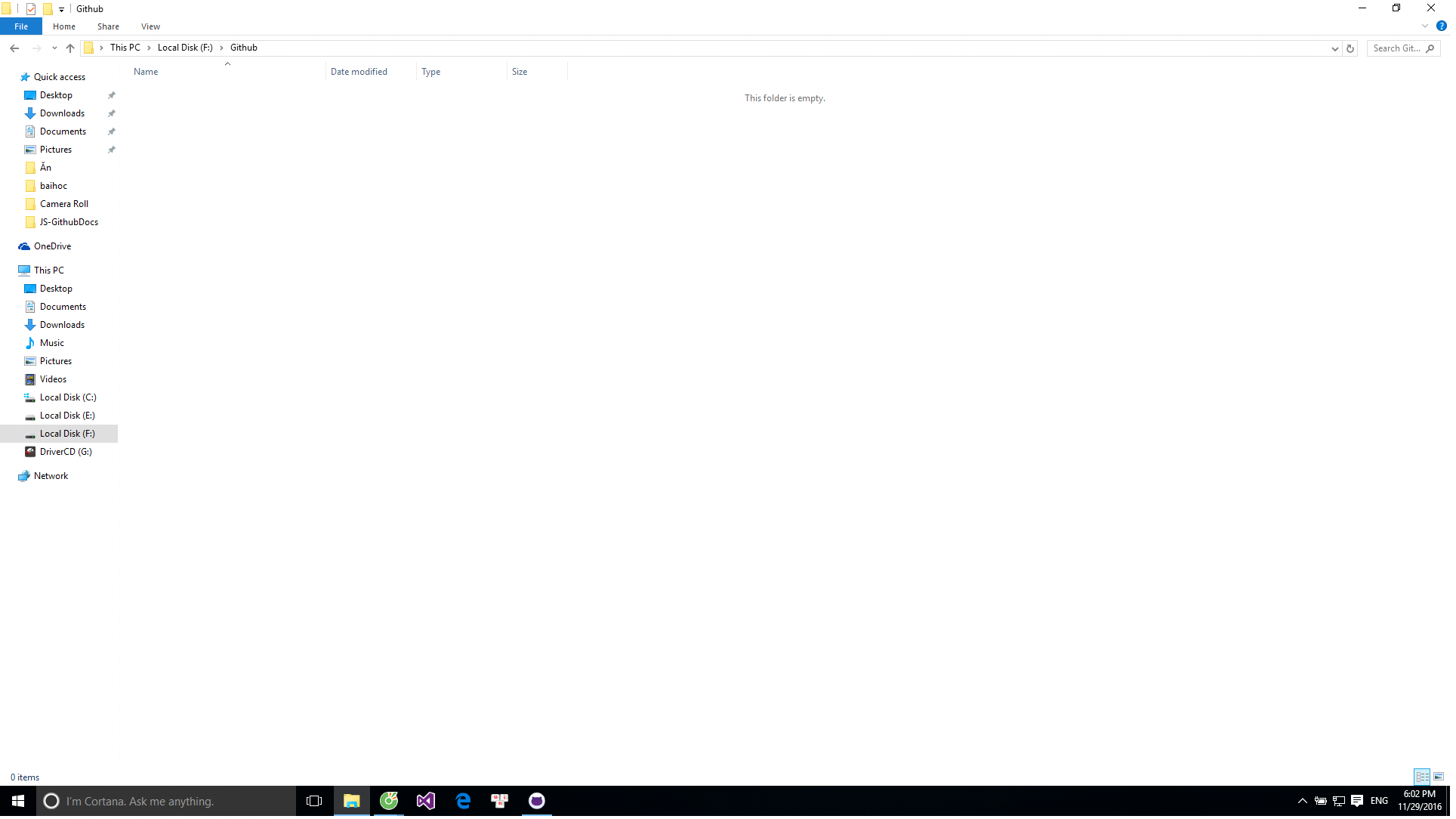
Repository đã được tạo ra.

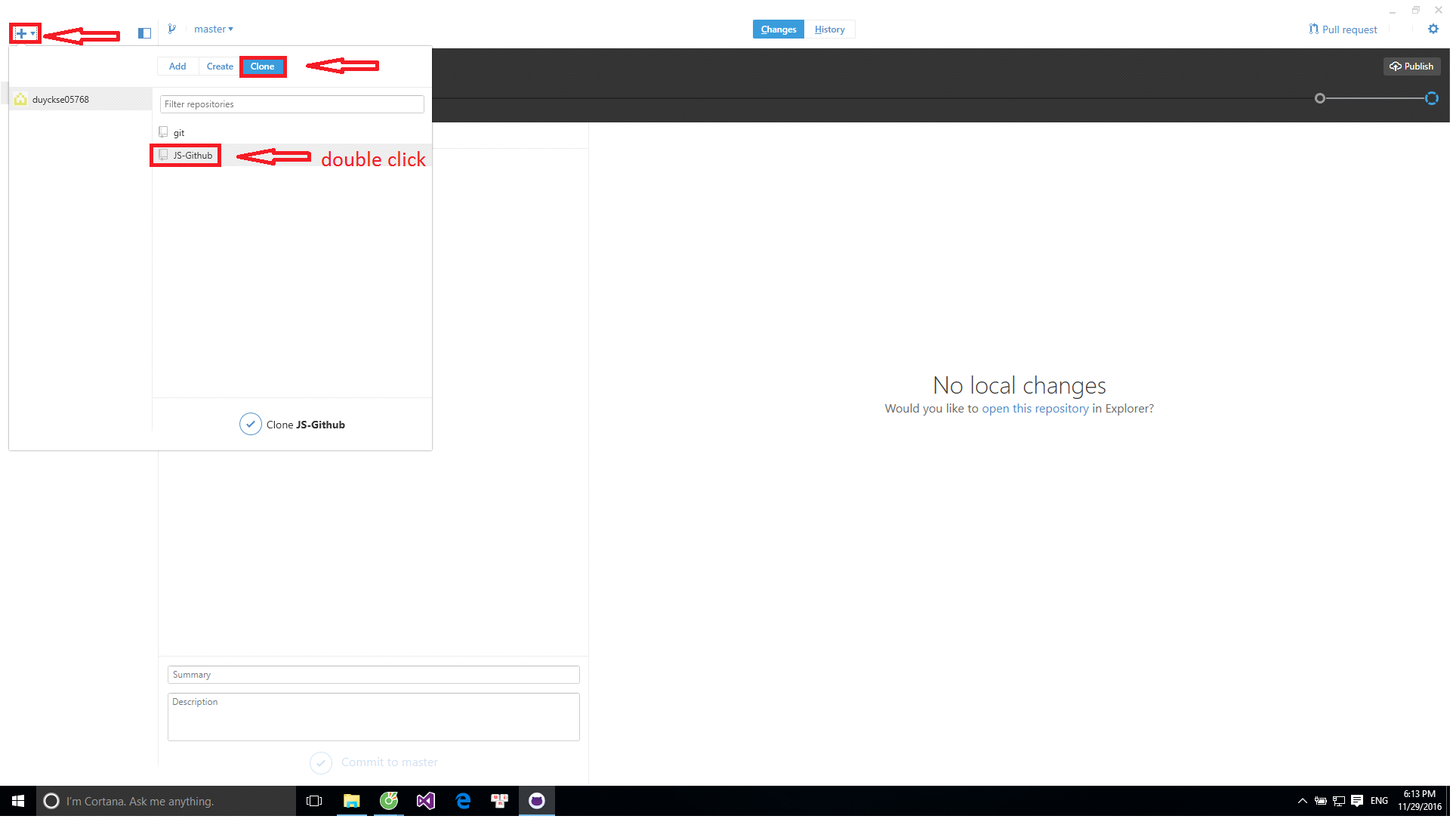
## Kết nối GitHub và GitHub Desktop

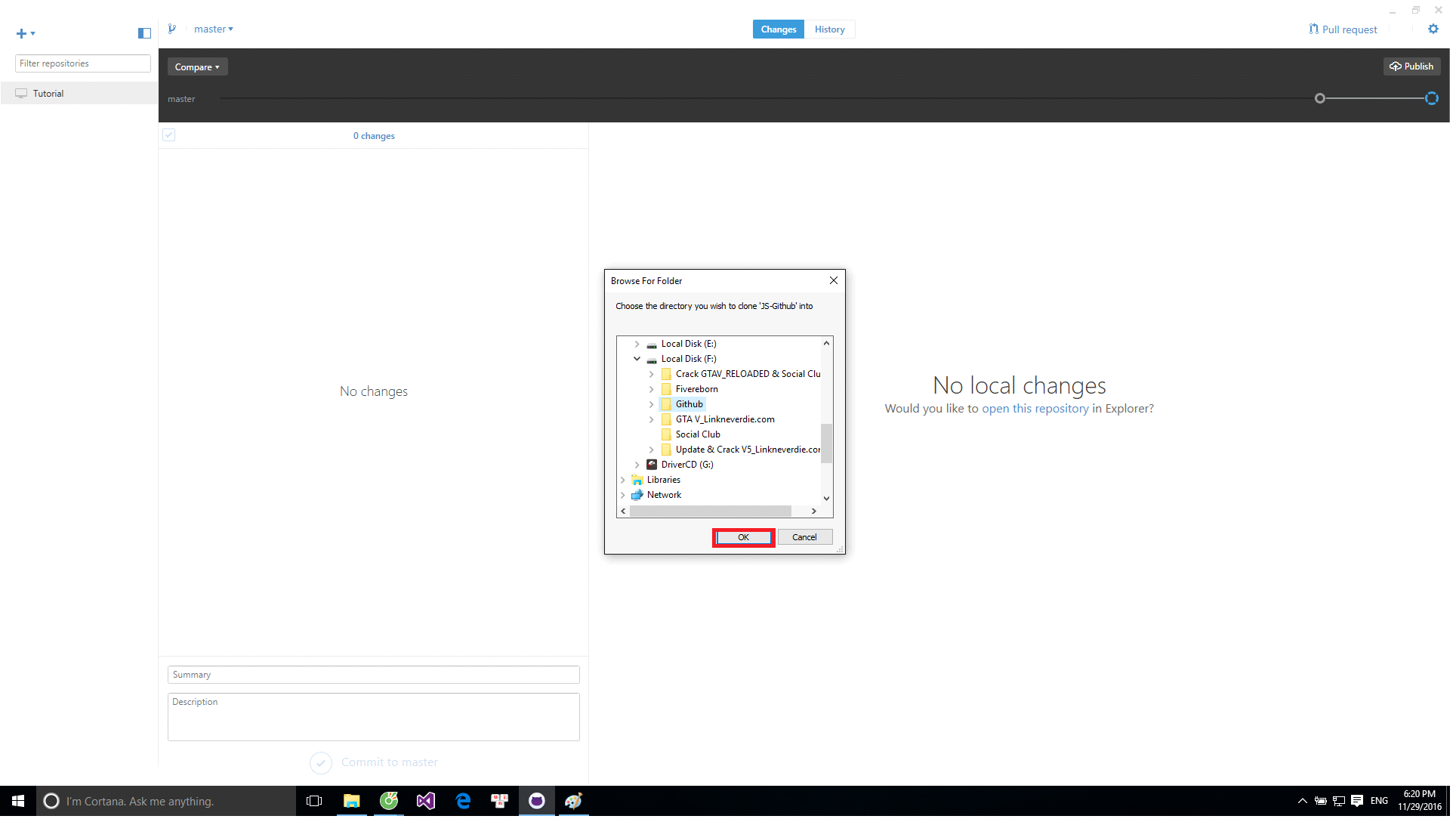
Trước hết cần chọn một thư mục rỗng để làm vị trí chứa dữ liệu địa phương.

Chẳng hạn:

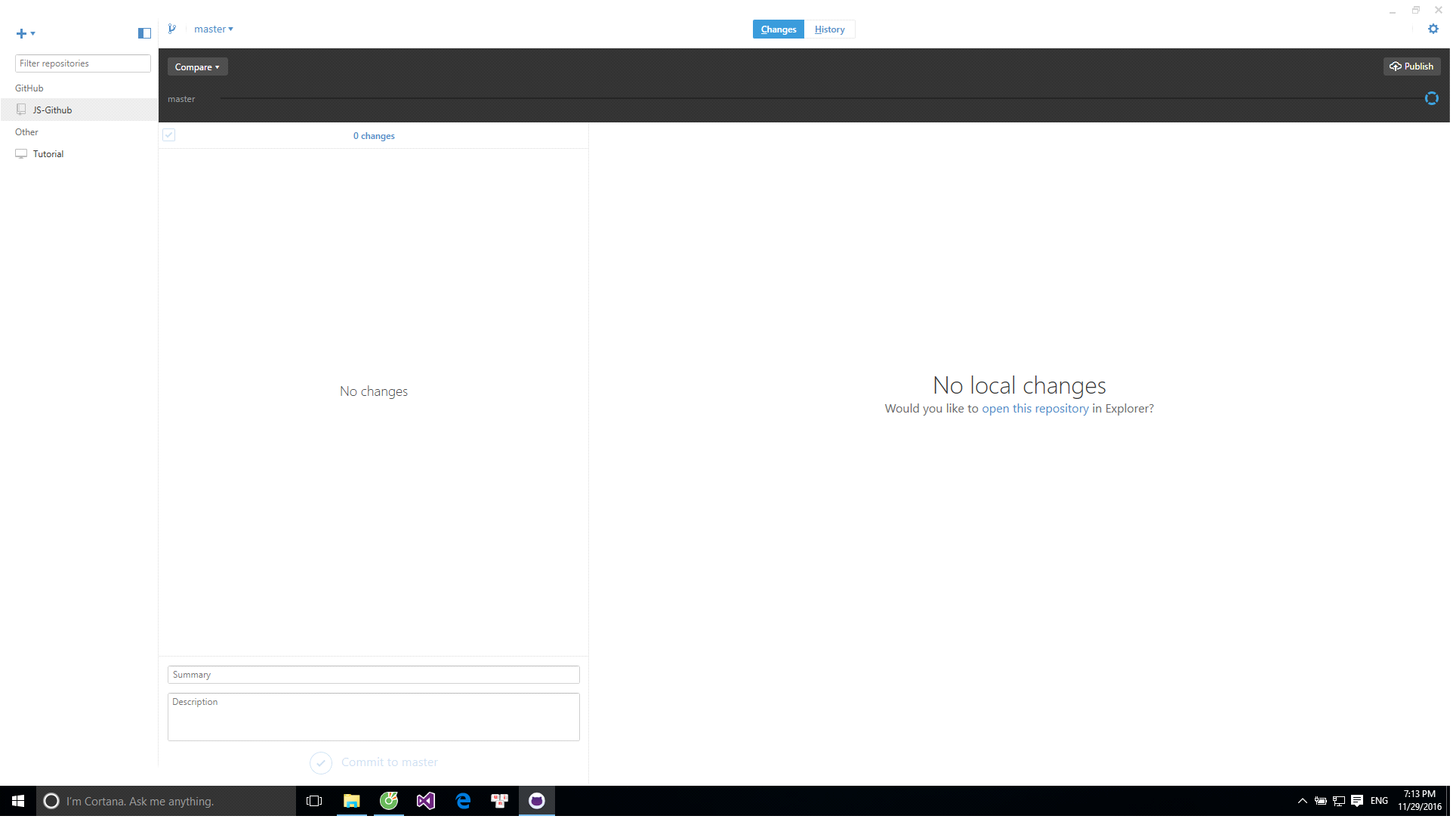
F:/GITHUB



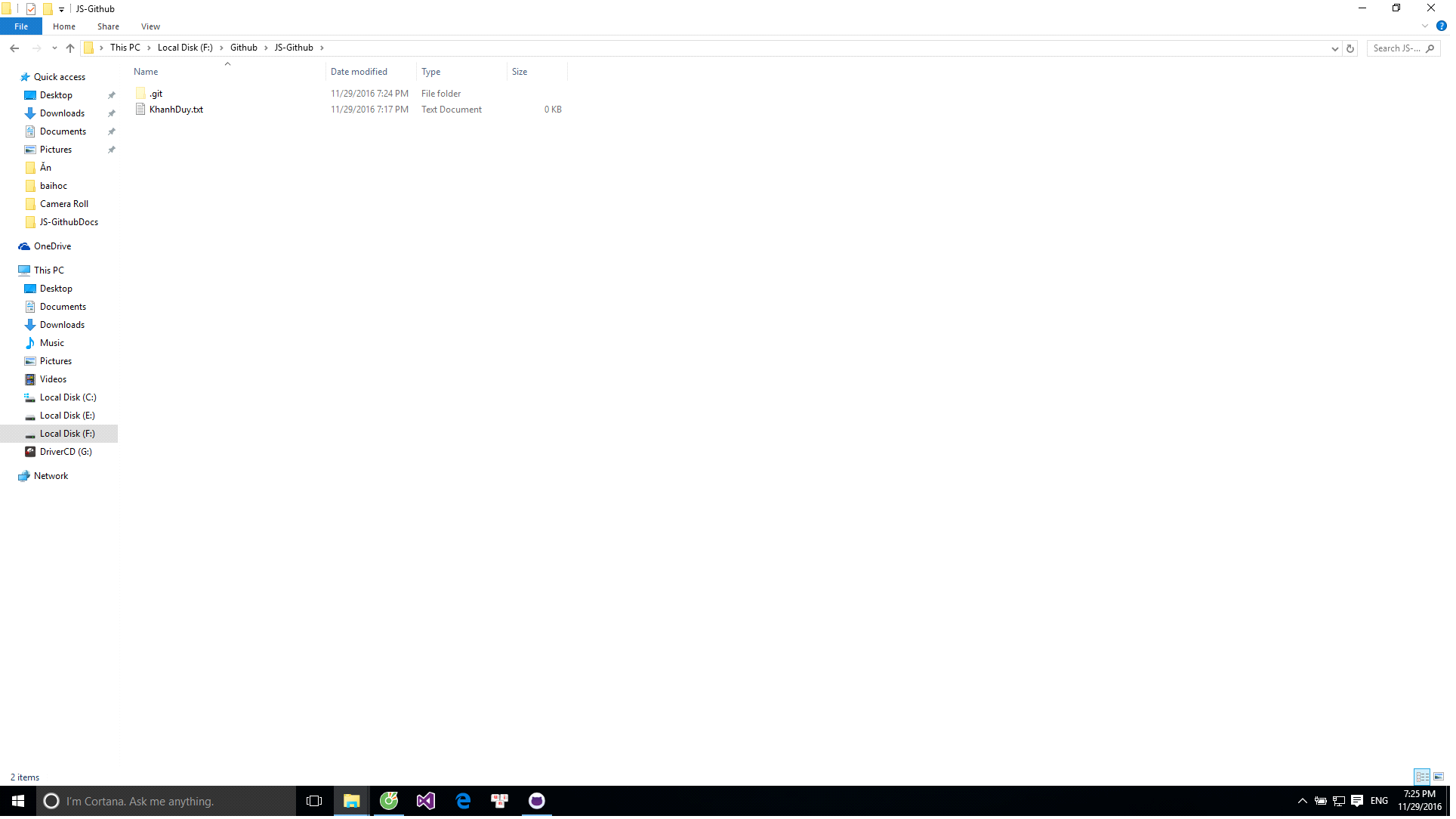
Trên GitHub Desktop, lựa chọn một Repository bạn đã tạo trên GitHub để Clone thành một bản ở máy tính địa phương của bạn.



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

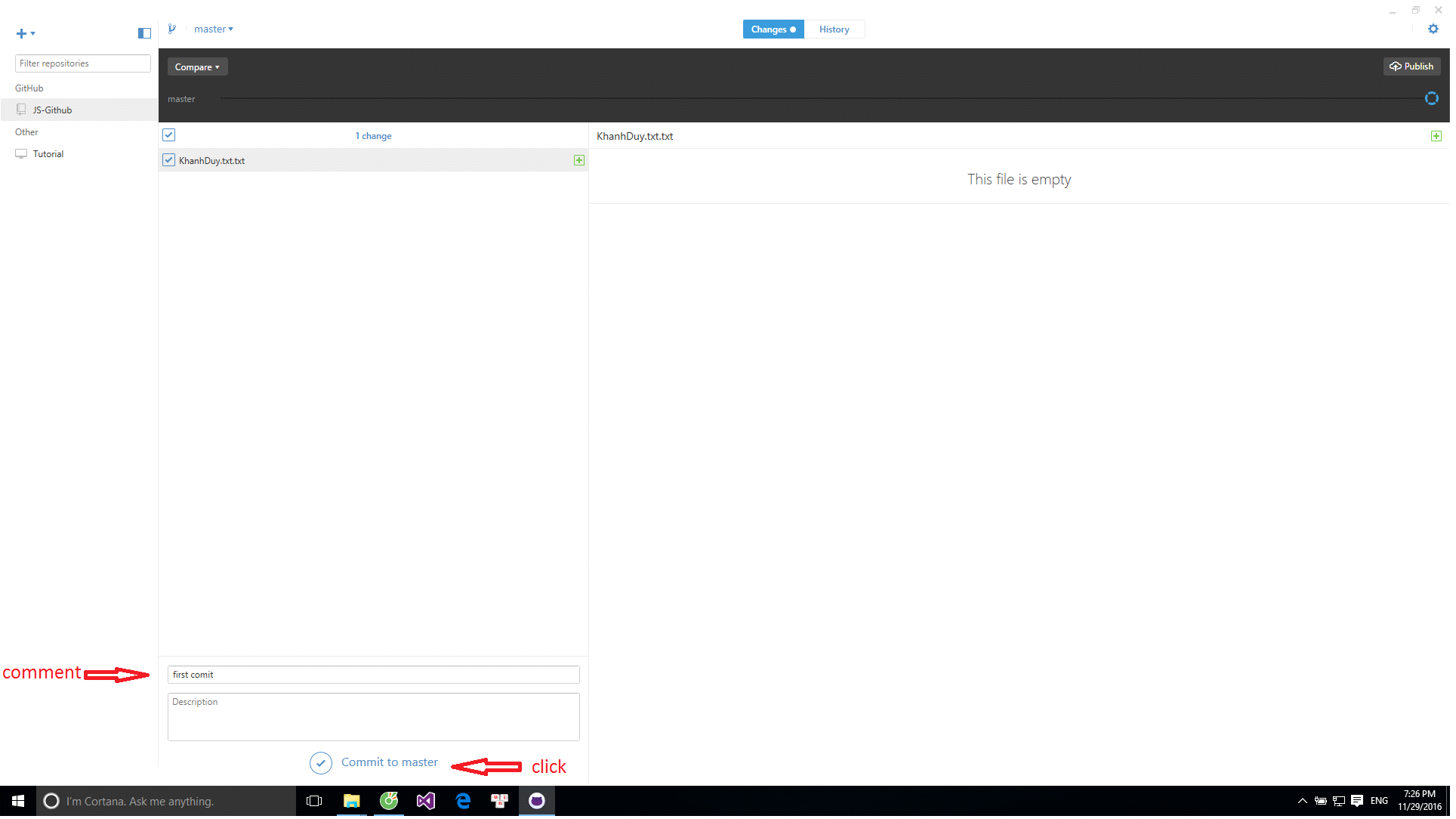


Và trên thư mục tại ổ cứng, một thư mục con đã được tạo ra:

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

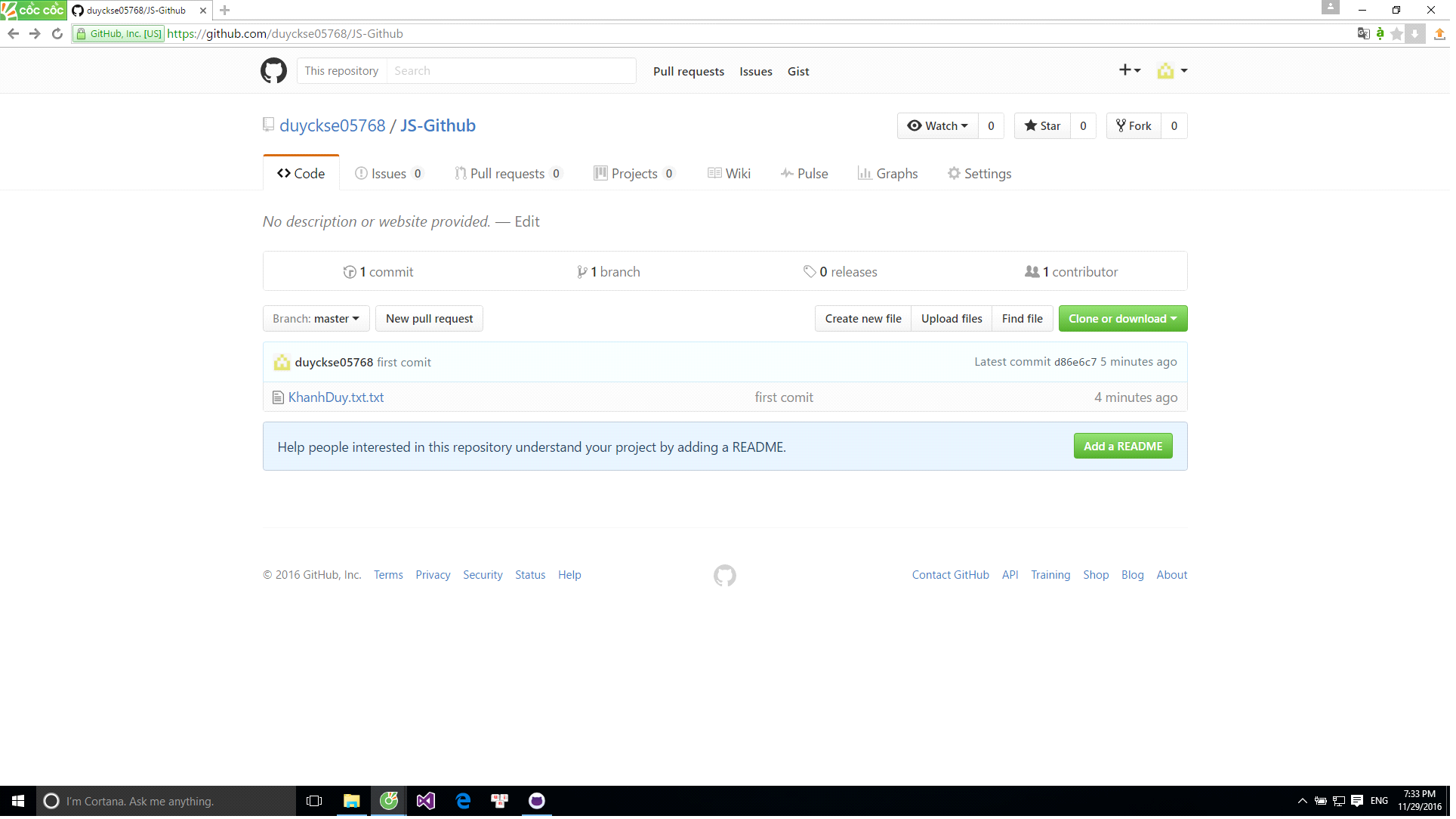
GitHub Desktop ngay lập tức nhận biết được các thay đổi tại Local Repository.

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



Cuối cùng là công khai cho mọi người có thể xem: bạn click vào Publish góc trên bên phải màn hình

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



# **Sử dụng các chức năng cơ bản của Github Desktop**

## Cách tạo và clone một Repository

**Repository** (kho chứa) nghĩa là nơi mà bạn sẽ lưu trữ mã nguồn và một người khác có thể sao chép (clone) lại mã nguồn đó nhằm làm việc. Repository có hai loại là Local Repository (Kho chứa trên máy cá nhân) và Remote Repository (Kho chứa trên một máy chủ từ xa).

Trong bài này, mình sẽ hướng dẫn bạn cách tạo local repository và remote repository (sử dụng Github) và làm việc với nó.

### Tạo local repository