# **Git và Git Hub**

## **Git**



* **Hệ thống quản lý phiên bản phân tán** (Distributed Version Control System – ***DVCS***)
* 1 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.

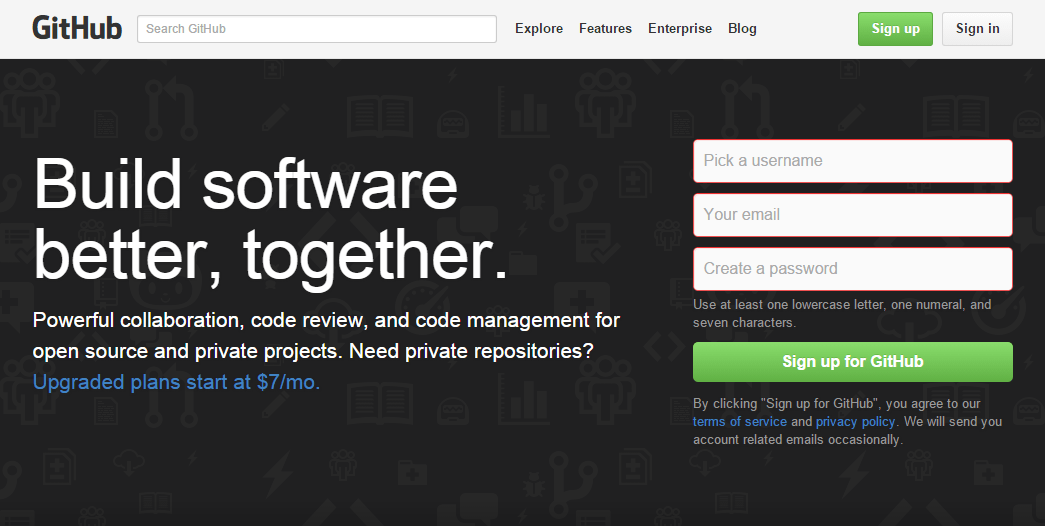
**Ngoài ra,** có một cách hiểu khác về Git đơn giản hơn đó là

1. 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.
2. 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.
3. 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 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**



**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 (Wiki)

**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.

Còn được gọi là social network dành cho developer đi vào hoạt động tháng 2 năm 2008

Là một dịch vụ sử dụng hệ thống quản lý phân tán giúp người dùng lưu trữ source code cho các dự án.

Git là một mô hình quản lý source phân tán, nó có mọi tính năng của một source control như SVN và hơn thế nữa.

Github được viết bằng Ruby on Rails.

GitHub cung cấp dịch vụ thương mại và cả tài khoản miễn phí cho các dự án nguồn mở.

Theo khảo sát của người sử dụng Git vào năm 2009, Github hiện đang là server Git lưu trữ source code phổ biến nhất hiện nay (Ngoài ra, [Gitorious](https://gitorious.org/) cũng là server Git hoạt động giống Github được chú ý đến).

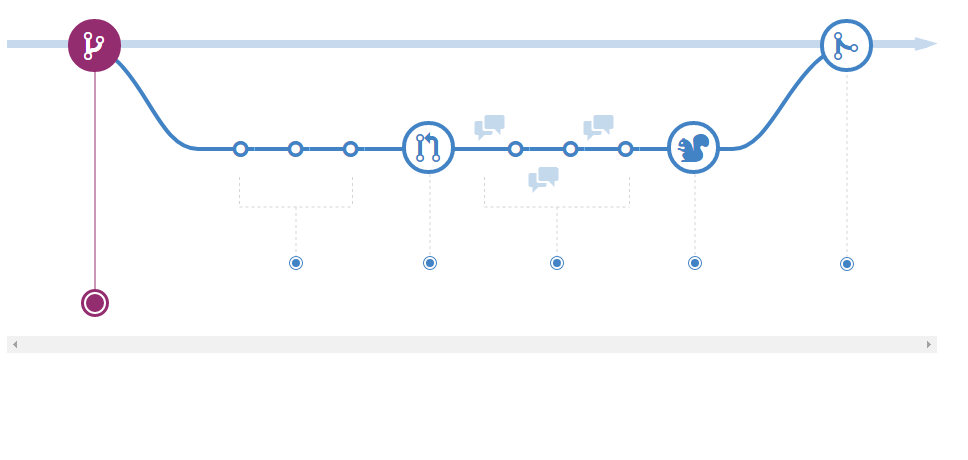
Có hơn 439000 developer tạo hơn 1 triệu 350 ngàn repositories là một con số khá ấn tượng, cùng với một số khách hàng lớn của github như Twitter, Facebook, Yahoo ... cho thấy tính phổ biến của Github, cũng như cộng đồng lập trình thế giới tín nhiệm nó như thế nào.

# **Quy trình làm việc của GitHub**

Làm việc với GitHub nói riêng hay hệ thống GIT nói chung có 2 workflow

* Local workflow
* Server workflow.
* Bạn có thể làm mọi chuyện thay đổi source code ở local
* Sau khi đã thay đổi xong, bạn sẽ commit những thay đổi đó lên server và bản lên server phải là bản hoàn chỉnh một tính năng nào đó, hoặc fix bug xong, test xong hoặc ít nhất bản đó phải chạy được
* Không được commit code dở dang, chưa qua test lên repository server sẽ làm ảnh hưởng đến các thành viên khác
* Ngược lại bạn có thể làm điều đó ở repository local
* Bạn cũng có thể tạo một branch ở server cho việc commit code dở dang hay tính năng chưa hoàn thành như từng làm với SVN, nó sẽ chiếm space ở server cũng như làm mất thời gian của bạn vào việc tương tác kết nối với server
* Vậy tại sao không commit nó lên repository local, vừa nhanh thao tác lại không mất space của server.

### Tạo một branch

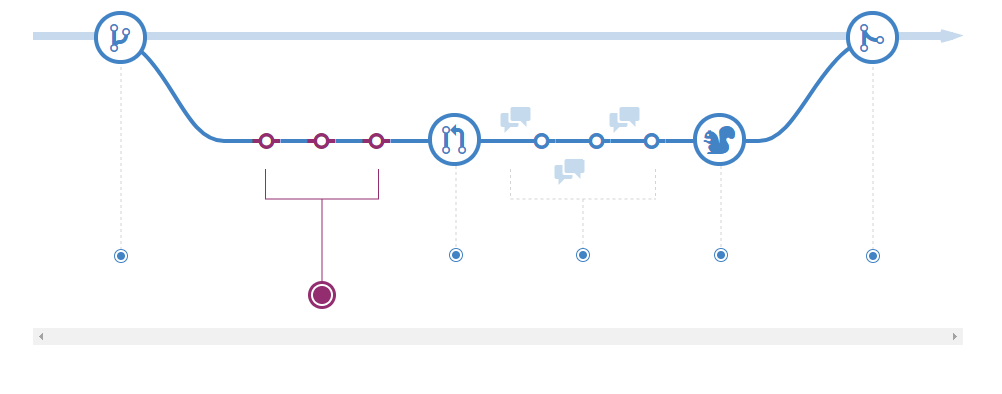


* Khi bạn làm việc trong một dự án, bạn sẽ có nhiều nhóm chức năng hoặc ý tưởng được triển khai trong bất kì khoảng thời gian nào, một số thì đã sẵn sàng nhưng một số khác thì không.
* Các branch tồn tại giúp chúng ta quản lý các quy trình cho công việc này.
* Khi bạn tạo một branch trong dự án của bạn, bạn tạo ra một môi trường ở đó bạn có thể thực hiện những ý tưởng mới.
* Khi bạn thay đổi trên branch đó nó không tác động được tới “master” branch (source code chính), bạn sẽ tự do trong việc thử nghiệm và commit các thay đổi , an toàn khi biết rằng bạn branch của bạn sẽ không bị merge cho đến khi nó sẵn sàng được kiểm tra (review) bởi những thành viên mà bạn đang cộng tác cùng.

### ProTip

* Rẽ nhánh (Branching) là khái niệm cốt lõi của git, toàn bộ github flow dựa trên hoạt động của nó. Chỉ có duy nhấtmột quy định: mọi thứ trên “master ” branch luôn luôn có thể triển khai (deploy) được.
* Bởi vì điều này, điều cực kì quan trọng là branch mới được tạo ra từ “master” khi bạn đang tạo ra một chức năng (feature)hoặc sửa chữa(fix). Tên branch của bạn nên có tính mô tả (Ví dụ: refactor-authentication, user-content-cache-key, make-retina-avatars) vì vậy những người khác có thể thấy những gì đang được xây dựng

### Thêm một commits

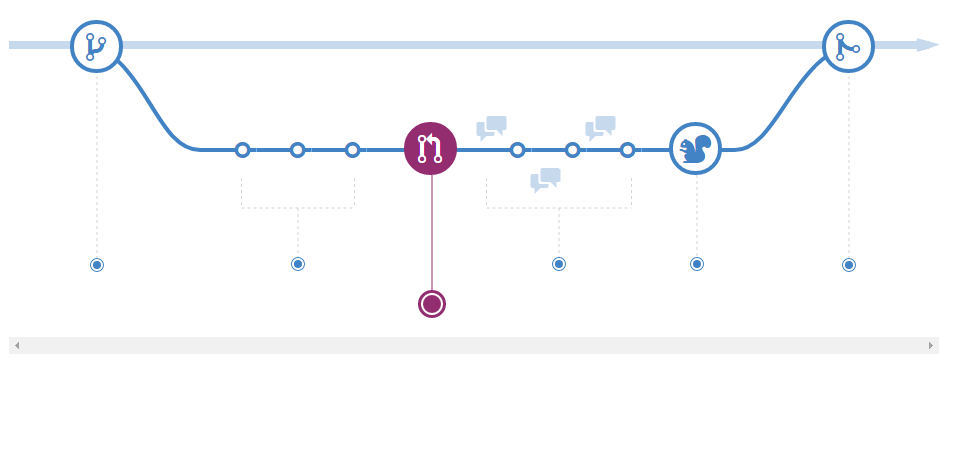


* Một branch của bạn khi tạo mới, đây là lúc bắt đầu tạo ra những thay đổi.
* Bất cứ khi nào bạn thêm, sửa hoặc xoá một file, bạn đang tạo mới một commit, và thêm nó vào branch của bạn
* Quy trình của việc thêm commit này theo dõi các thay đổi khi bạn làm việc trong một feature branch.
* Commits cũng tạo ra một lịch sử rất minh bạch cho, công việc của bạn giúp cho các thành viên khác có thể theo dõi để hiểu được bạn đã hoàn thành công việc gì và tại sao làm thế. Mỗi một commit có một tin nhắn bổ trợ thông tin (commit message) , nó mô tả tại sao lại có các thay đổi được tạo ra.
* Ngoài ra, mỗi commits là một đơn vị tách rời của thay đổi. Điều này giúp bạn bạn lật lại các thay đổi về trước (rollback) nếu bug được tìm thấy, hay nếu bạn muốn sửa đổi theo một hướng khác.

### ProTip

* Commit mesages rất quan trọng, nhất là từ khi git theo dõi được các thay đổi của bạn và hiển thị mọi sự thay đổi khi các commit được push lên sever.
* Bằng việc viết commit mesages một cách rõ ràng, bạn có thể giúp cho những thành viên khác có thể theo dõi và đưa cho bạn những phản hồi (feedback) một cách dễ dàng.

### Mở một pull request

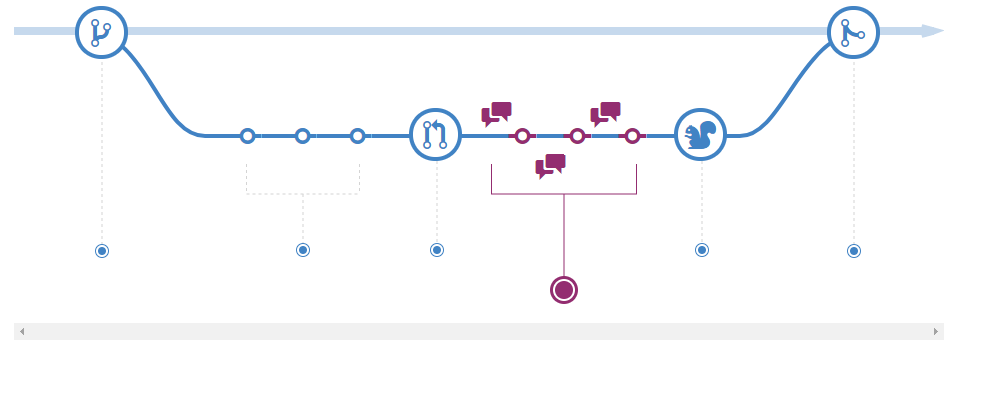


* Pull requests là bắt đầu thảo luận về commits của bạn
* Bởi vì chúng được lấy và phân tích chặt chẽ với git repository bên dưới, bất kì ai cũng có thể nhìn thấy chính xác cái gì đang thay đổi sẽ được merged nếu họ đồng ý yêu cầu của bạn.
* Bạn có thể mở pull request tại mọi điểm trong suối quá trình quy trình phát triển. Khi bạn có ít hoặc không có code nhưng bạn muốn chia sẻ ảnh hoặc ý tưởng chung,khi bạn bị kẹt và cần giúp đỡ hoặc lời khuyên, hoặc khi bạn sẵn sàng với review công việc của mọi người.
* Việc sử dụng Github được hệ thống nhắc đến trên pull requeest mesage của bạn, bạn có thể yêu cầu trả lời từ những người cụ thể hoặc các nhóm, cho dù bất cứ khi họ ở đâu và vào lúc nào cũng ok.

### ProTip

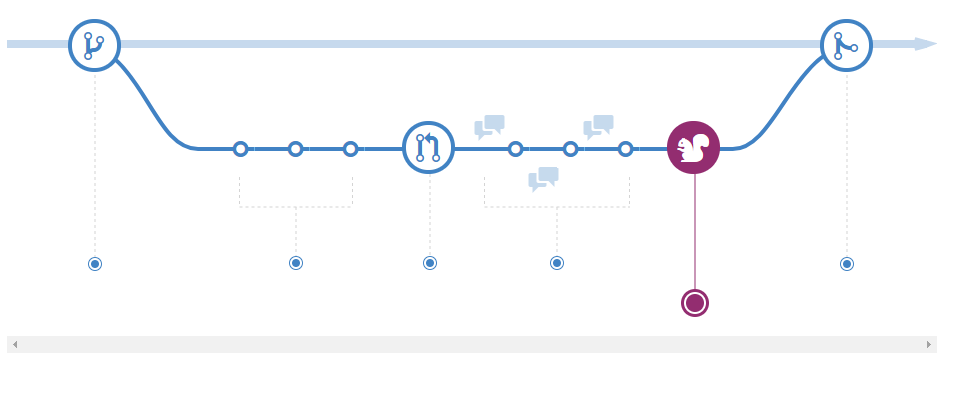
* Pull request là đóng góp có ích để mở nguồn dự án và mở thay đổi quản lý shared repostiories.
* Nếu bạn sử dụng mô hình fork and pull, pull request cung cấp một cách thông báo cho người quản lý dự án nếu bạn muốn họ xem xét thay đổi bạn đưa ra
* Nếu bạn sử dụng chia sẻ pepository, việc pull requests giúp bắt đầu xem lại code và cuộc hội thoại được dự kiến thay đổi trước khi chúng được kết hợp với nhánh master “ code chinh dự án”.

### Discuss and review your code



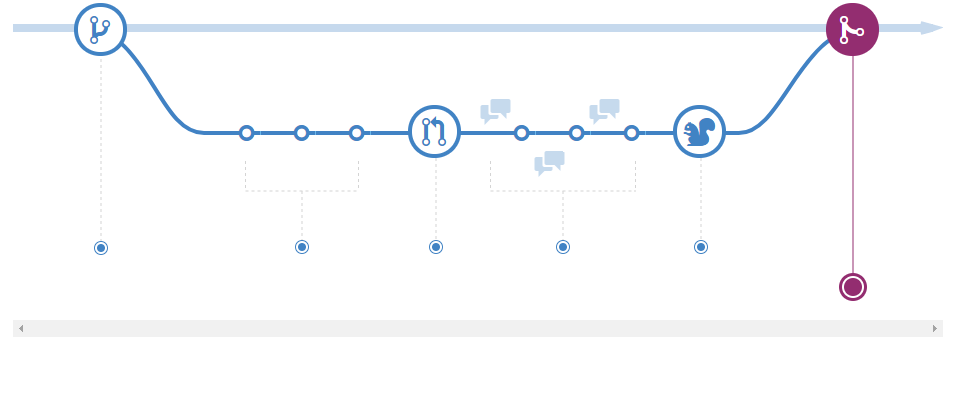
* Trong một pull request, mọi người trong nhóm của bạn có thể xem lại bạn đã thay đổi code như thế nào
* Họ có thể có những câu hỏi hay lưu ý cho bạn. Có thể có những code style không phù hợp với nguyên tắc dự án, những phần đó sẽ được thay đổi kiểm tra sao cho phù hơp mới tổng thể của dự án.
* Pull requests là thiết kế để khuyến khích và lắm bắt sự trao đổi trong nhóm. Bạn cũng có thể tiếp tục push vào branch code của bạn các thảo luận và trả lời những thực hiện commits của bạn
* Nếu ai đó nhận xét những thứ bạn quên không làm hoặc họ đưa ra lỗi trong code của bạn, bạn có thể sửa nó trong code của mình và thay đổi chúng
* GIthub sẽ hiện thị những thự hiện mới của bạn và mọi phản hồi mà bạn có thể nhận được thống nhất trên pull request view.
* ProTip
* Pull request nhận xét là viết trên Markdown
* Bạn có thể gửi ảnh và biểu tượng cảm xúc, sử dụng định dạng sẵn, và định dạng nhẹ nhàng khác.

### Deploy



* Bạn đưa lên một pull request đã được review và branch vượt qua các kiểm tra của bạn, bạn có thể triển khai dự án, bạn có thể thay đổi để kiểm tra và đưa ra các trình bày về chúng( code chương trình).
* Nếu phần branch của bạn có lỗi, bạn có thể lật ngược lại triển khai lại chúng trình bày trên master (có nghĩa là xem lại master nó chạy ổn định đến đâu lỗi chỗ nào sau đó kiểm tra lại phần đó)

### Merge



* Bây giờ, mọi thứ bạn thay đổi đã kiểm tra trong quá trình chạy, nó có thể được hợp nhất với code chính của bạn
* Sau khi hợp nhất, pull requests đưa lên được bảo quản trên dòng lịch sử thay đổi code của bạn. Bỏi vì chúng có thể được tìm kiếm lại, pull request cho phép quay trở lại thời gian, giúp người trong nhóm hiểu tại sao và làm như thế nào bạn quyết định làm vậy.
* ProTip
* Với những sát nhập từ khóa vào pull request của bạn, ban có thể liên kết các lỗi với code.
* Khi bạn pull request ghép code với master , các vấn đề liên quan sẽ được đóng

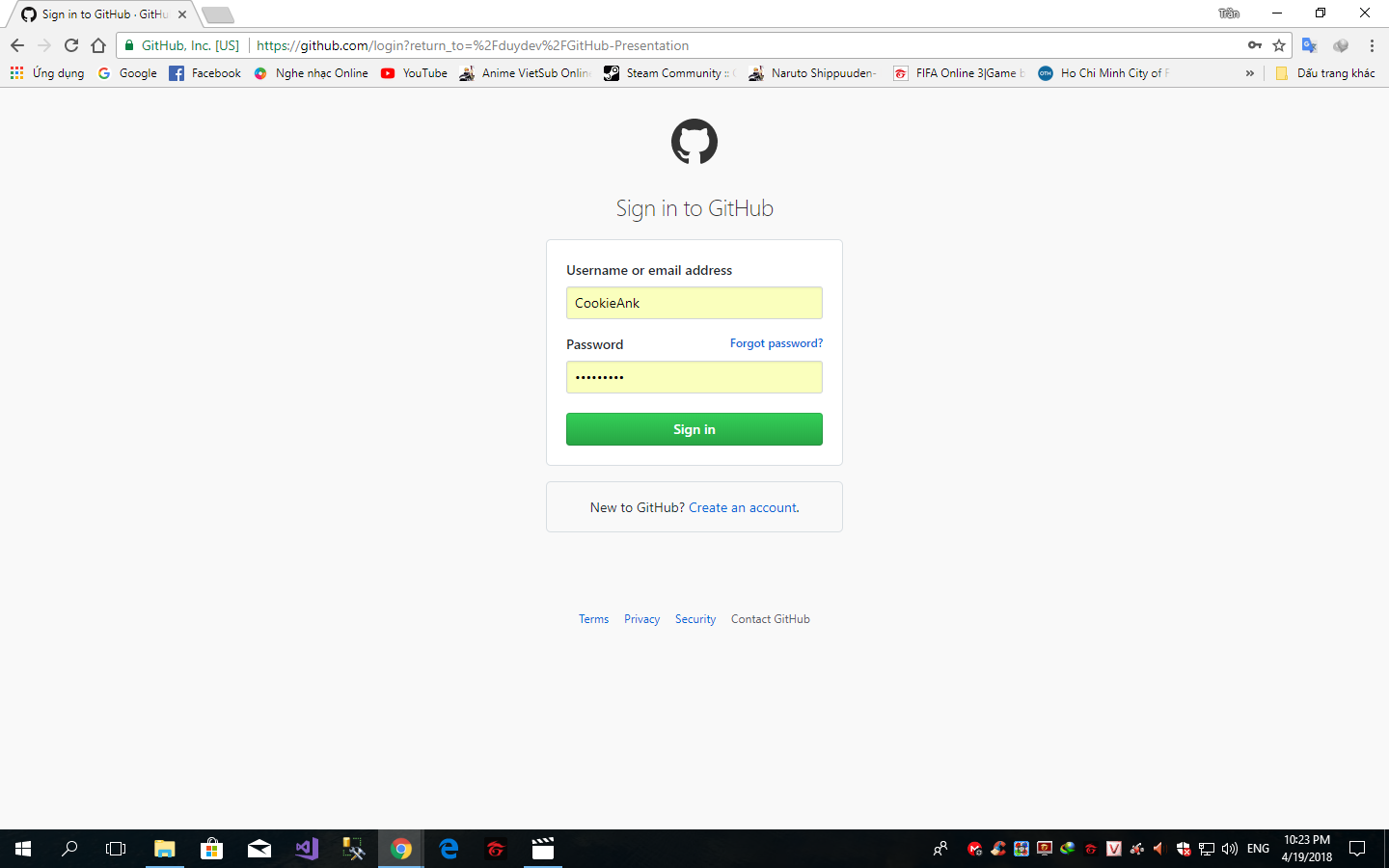
VD : nhập closees #32 nó sẽ đóng lỗi có số 32 trên repository.

# **Cách thức hoạt động của GitHub**

<https://www.youtube.com/watch?v=w3jLJU7DT5E&t=104s>

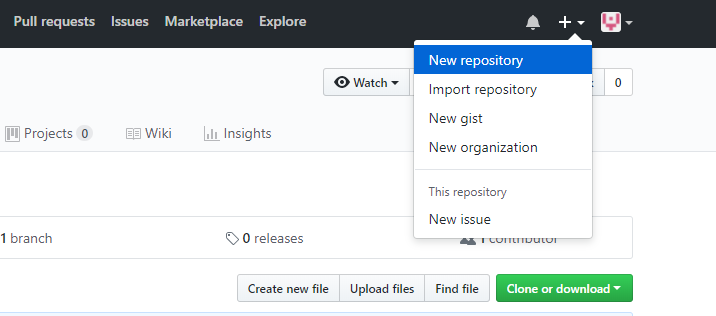
# **Cách sử dụng GitHub**

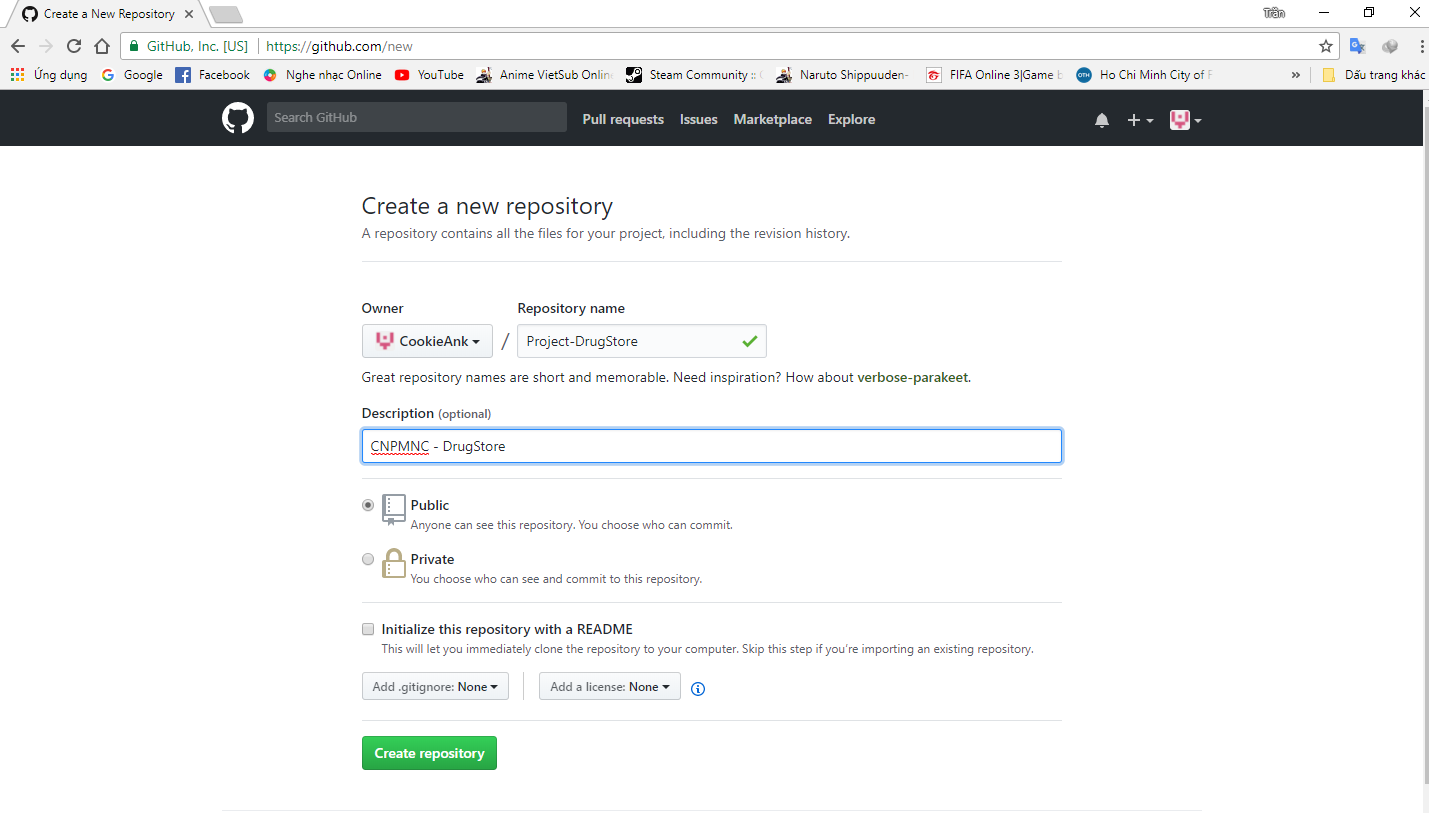
* **Đăng nhập**



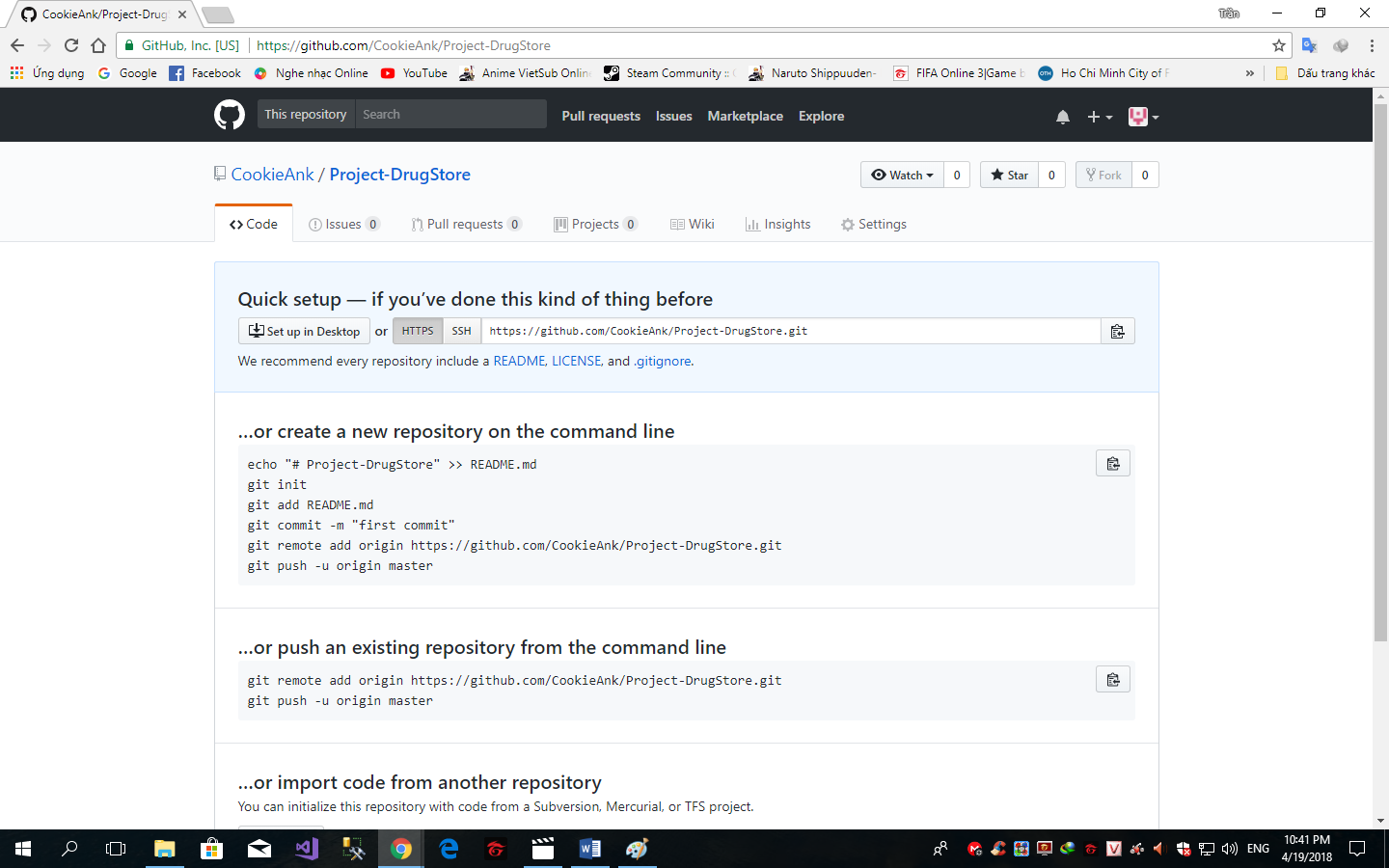
* **Tạo GitHub Repository**

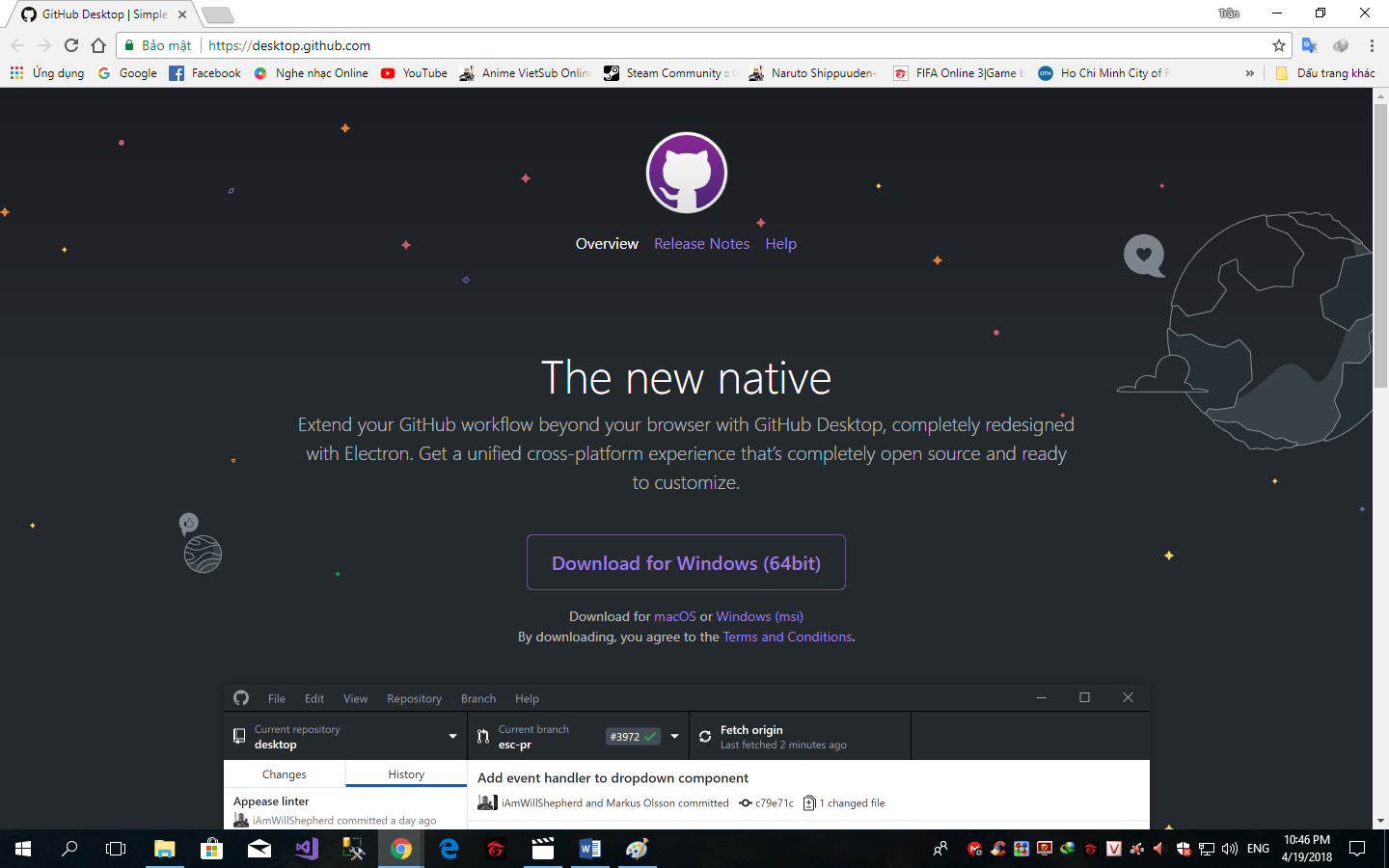
Sau khi đăng ký xong tài khoản **GitHub**, và đăng nhập vào. Bạn có thể tạo một **GitHub Repository**.

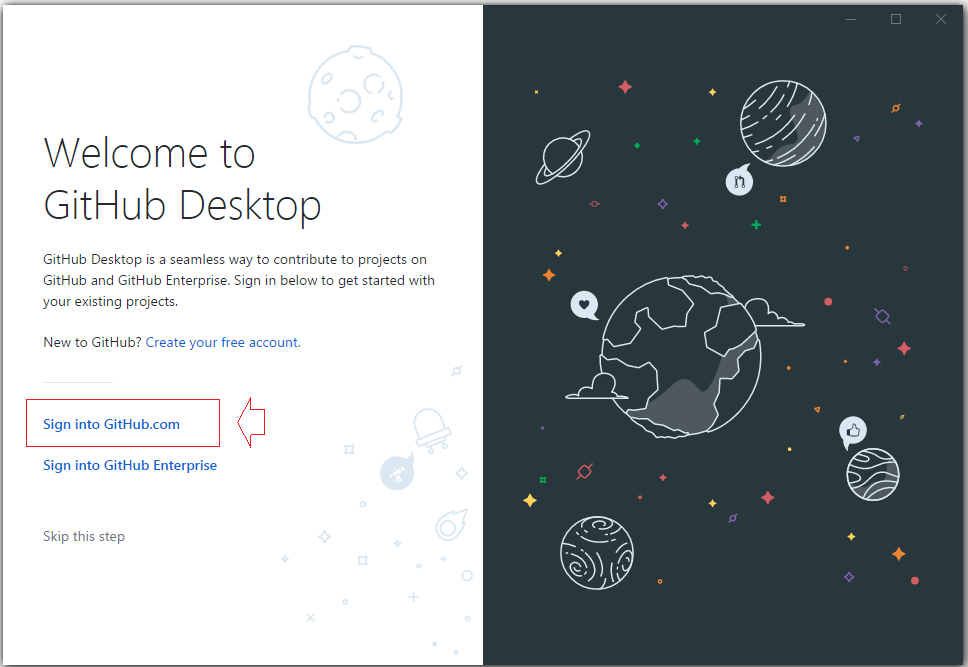




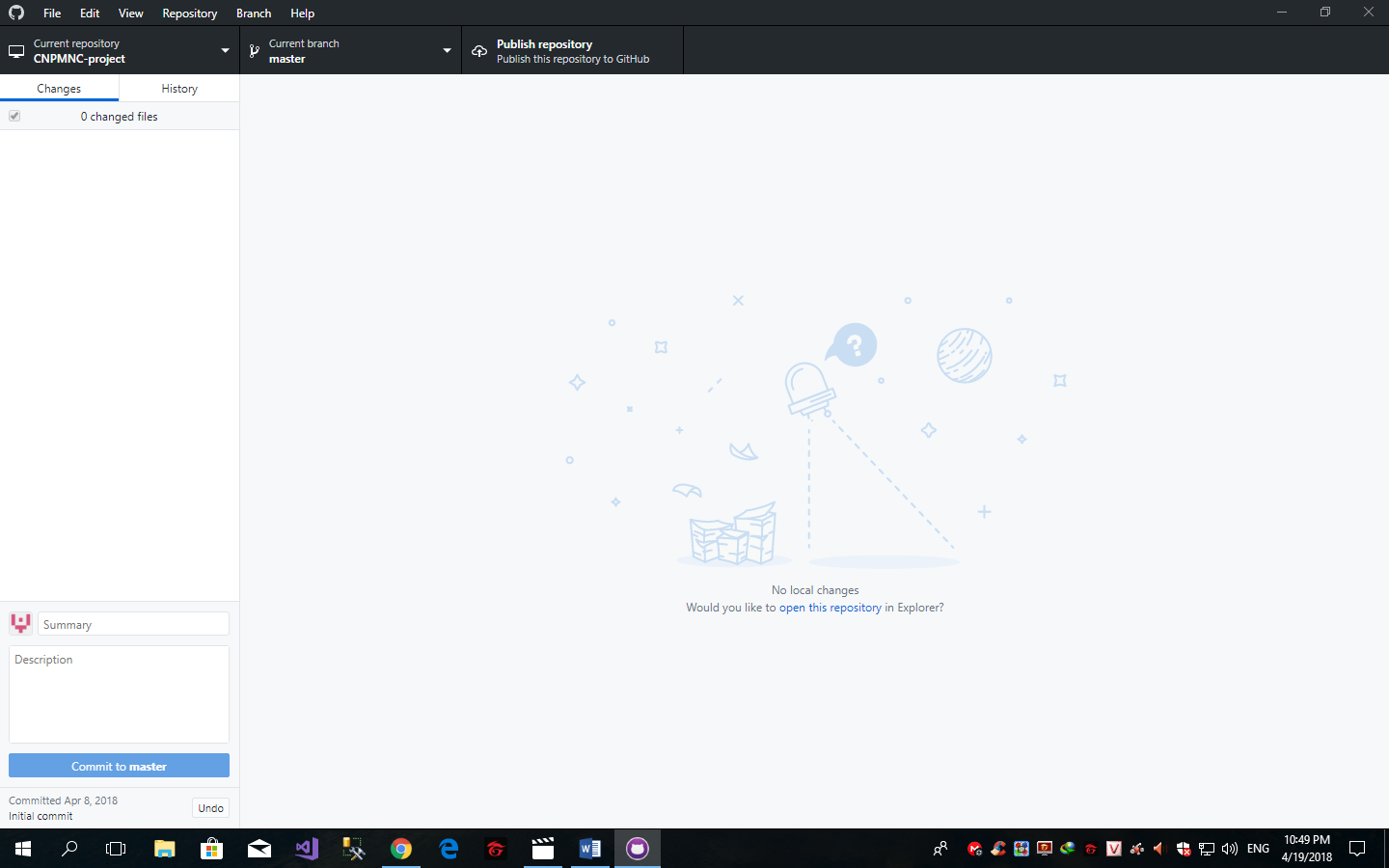
* **Repository**đã được tạo ra



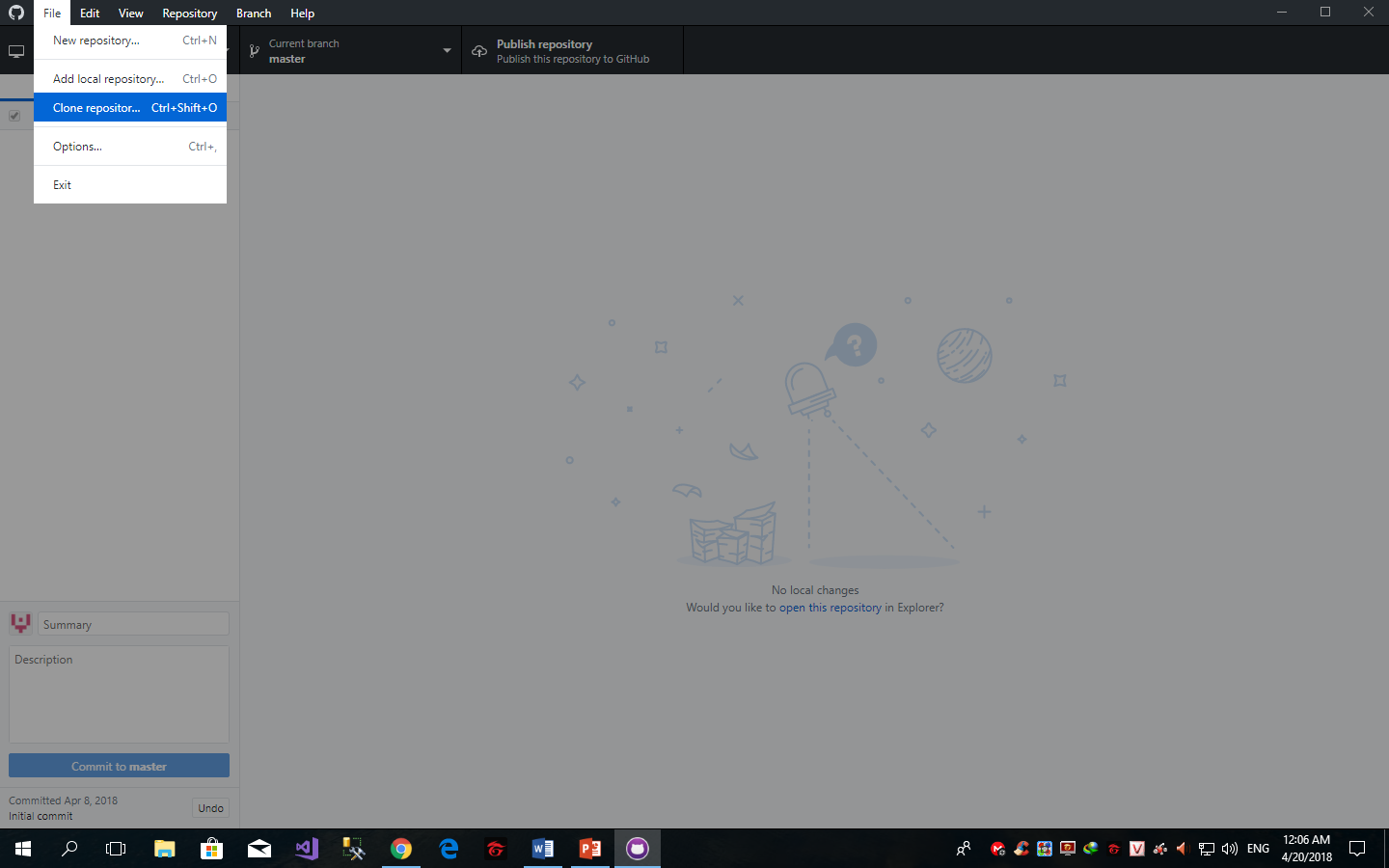
* **GitHub Desktop** công cụ trực quan cho phép bạn quản lý **Local Repository** (Kho chứa địa phương) trên máy tính của bạn
* **Đăng nhập GitHub Desktop**



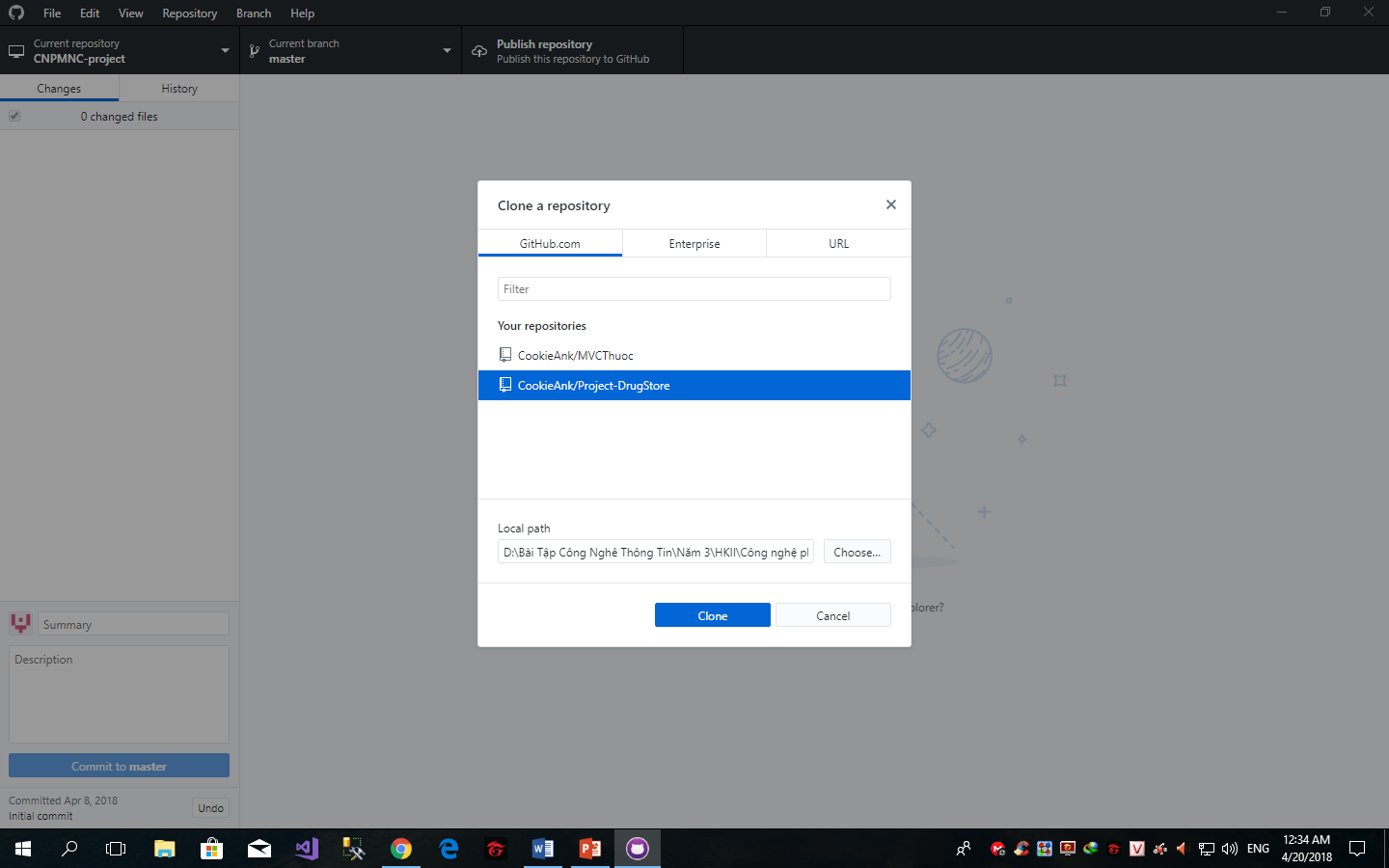
* Cho tới lúc này trên máy tính chưa có một **Local Repository** nào



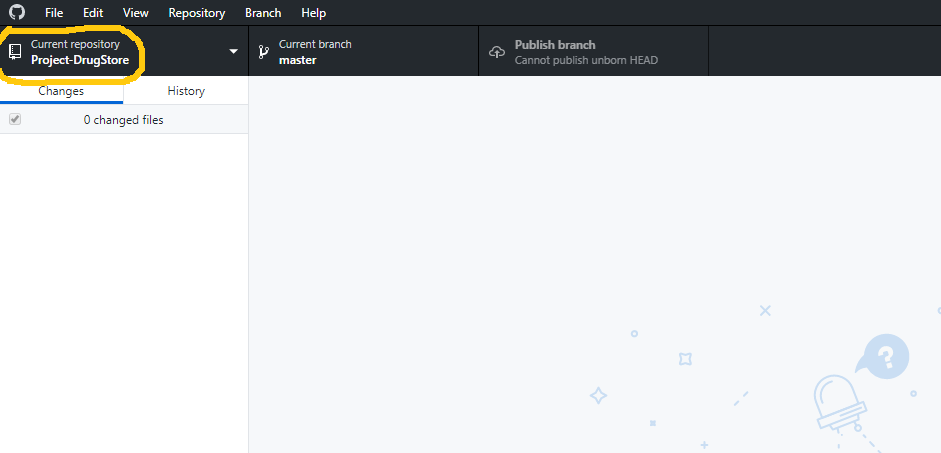
## **Kết nối GitHub và GitHub Desktop**



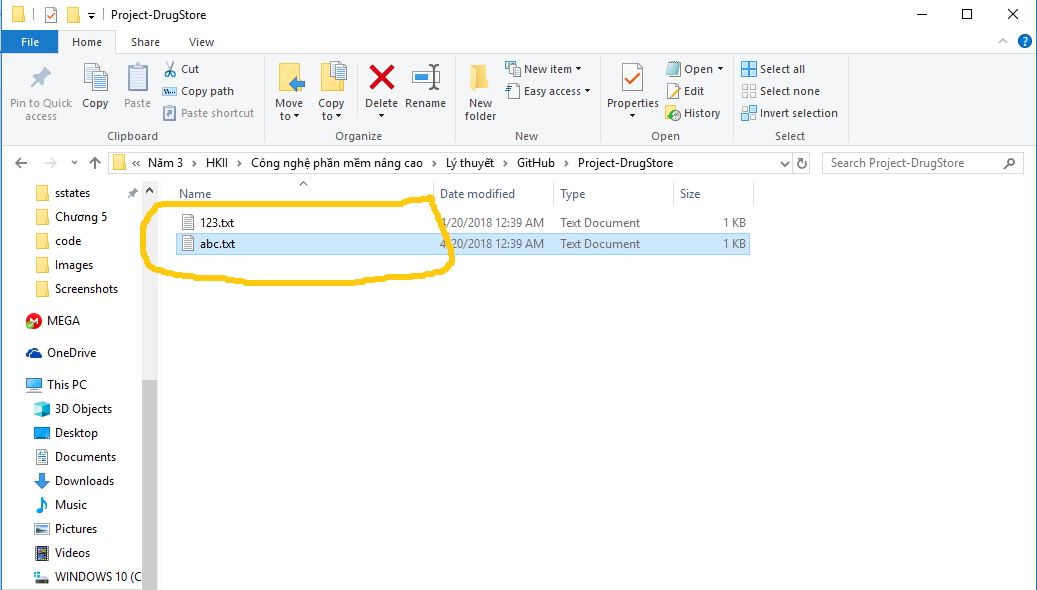
* Trên **GitHub Desktop**, lựa chọn một **Repository**bạn đã tạo trên **GitHub** để **clone**(Tạo bản sao chép) thành một bản ở máy tính địa phương của bạn.



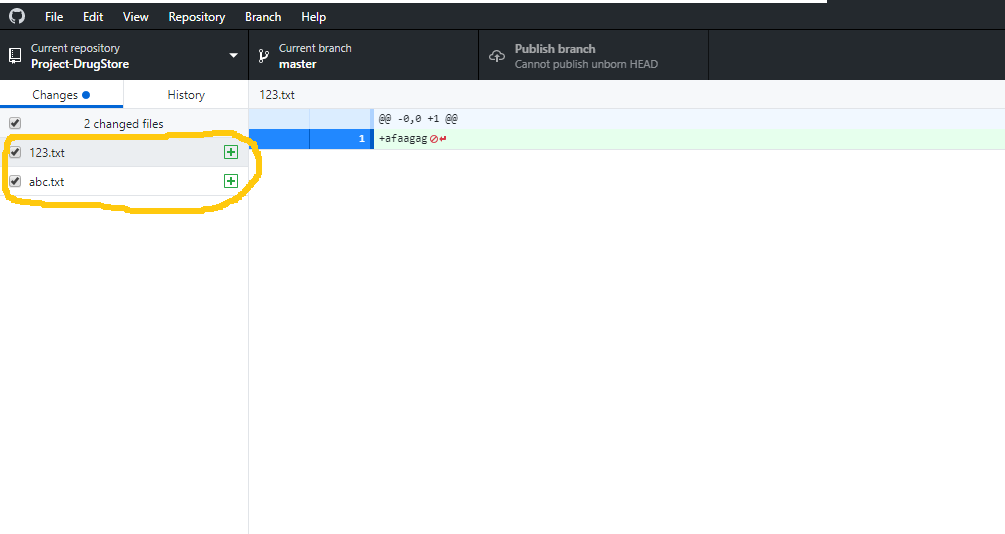
* Lúc này trên **GitHub Desktop** bạn sẽ thấy một **Local Repository** đã được tạo ra.



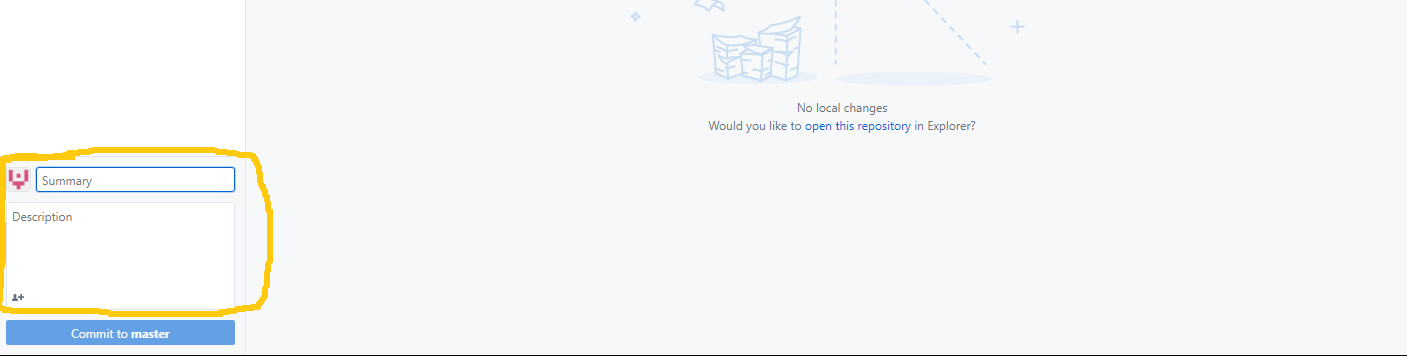
* Copy một vài file dữ liệu của bạn vào **Local Repository**



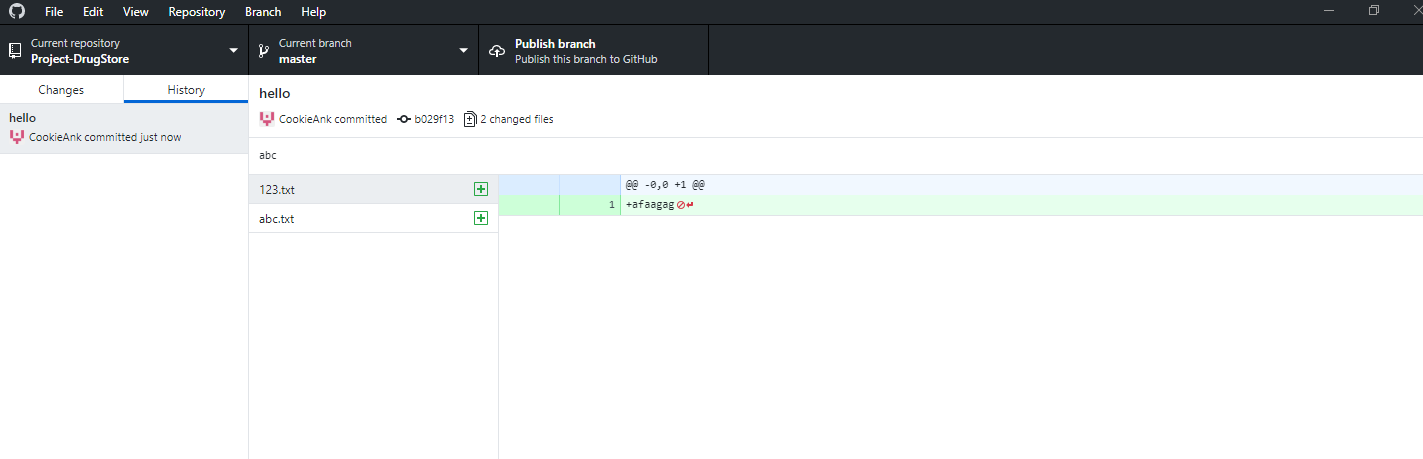
* **GitHub Desktop** ngay lập tức nhận biết được các thay đổi tại **Local Repository**



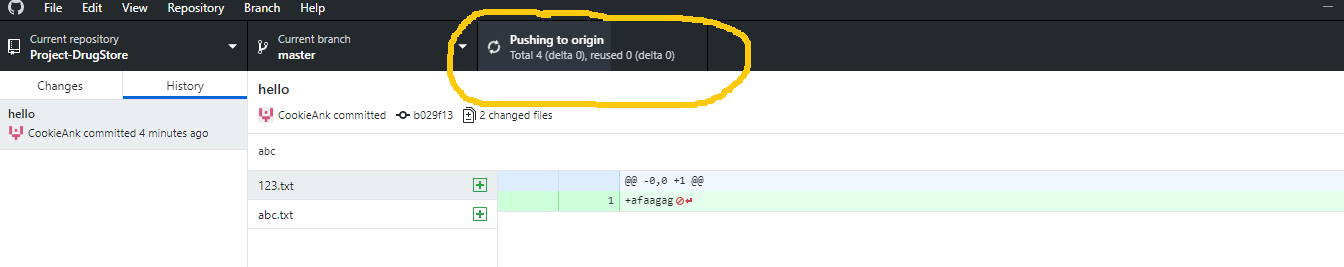
* Nhập thông tin ghi chú (Comment) và nhấn **Commit**dữ liệu



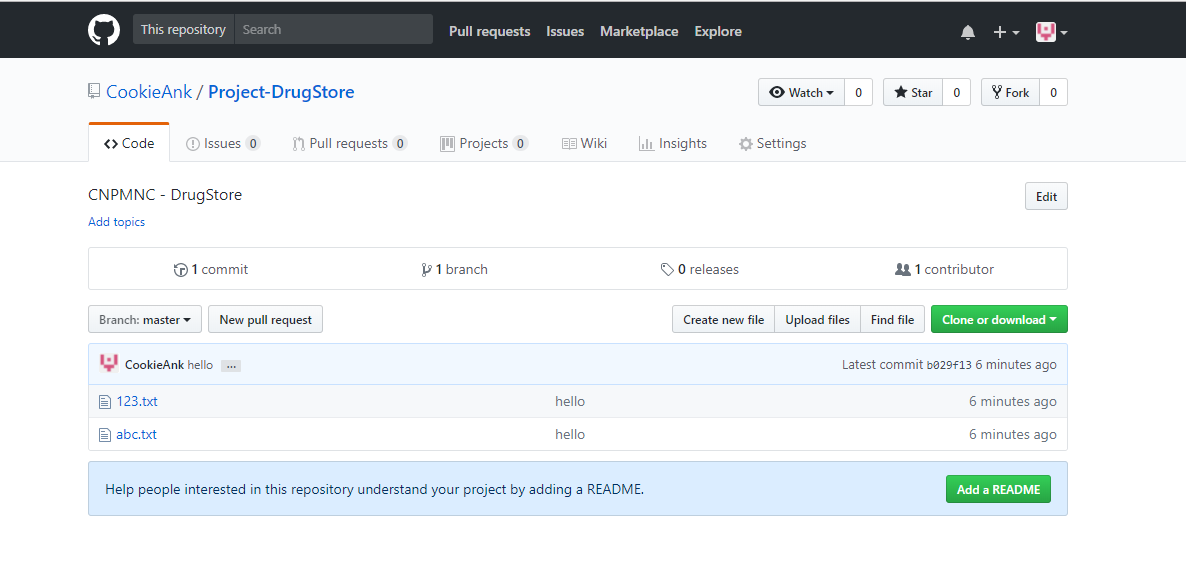
* Lịch sử **commit**



* **Push** dữ liệu lên **Branch**



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

****

# **Tại sao nên sử dụng Git?**

* 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……

# **Tài liệu tham khảo**

[https://try.github.io](https://l.facebook.com/l.php?u=https%3A%2F%2Ftry.github.io%2F&h=ATP_KnKk64gdRg7_gjD18cQRZlSw1pi9KHQQFVgGRbq_Jh6uboHNAI55OfETiaqh3Wvz7hMKsiFri5Q3YMET8zheRv6qfZId-YYlAz3mabhniDf8YCqdMQ) learn git step by step  
[https://git-scm.com/book/en/v2](https://l.facebook.com/l.php?u=https%3A%2F%2Fgit-scm.com%2Fbook%2Fen%2Fv2&h=ATP_KnKk64gdRg7_gjD18cQRZlSw1pi9KHQQFVgGRbq_Jh6uboHNAI55OfETiaqh3Wvz7hMKsiFri5Q3YMET8zheRv6qfZId-YYlAz3mabhniDf8YCqdMQ) - wiki git  
[https://githowto.com/](https://l.facebook.com/l.php?u=https%3A%2F%2Fgithowto.com%2F&h=ATP_KnKk64gdRg7_gjD18cQRZlSw1pi9KHQQFVgGRbq_Jh6uboHNAI55OfETiaqh3Wvz7hMKsiFri5Q3YMET8zheRv6qfZId-YYlAz3mabhniDf8YCqdMQ) - learn git step by step  
[https://www.codecademy.com/learn/learn-git](https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.codecademy.com%2Flearn%2Flearn-git&h=ATP_KnKk64gdRg7_gjD18cQRZlSw1pi9KHQQFVgGRbq_Jh6uboHNAI55OfETiaqh3Wvz7hMKsiFri5Q3YMET8zheRv6qfZId-YYlAz3mabhniDf8YCqdMQ)