**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Nội dung :**

**Git ->**

**Tiến độ thực hiện công việc**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CHƯƠNG 2: NỘI DUNG CÔNG VIỆC THỰC TẬP TẠI** **CÔNG TY**

* 1. **Tiến độ thực hiện công việc.**

Thời gian thực tập từ 06/02/2020 - 02/05/2020

Thời gian làm việc từ thứ 2 đến thứ 6. Ngoài giờ hành chính hoặc vào những ngày không có mặt tại văn phòng thì vẫn làm online bằng Skypes.

Hình thức làm việc chủ yếu của nhóm e tại đơn vị là hoàng thành những task mà bên đơn vị thực tập giao.

Và được tham gia vào dự án thực tế với vai trò là thực tập sinh học hỏi kinh nghiệm.

Tiến độ thực hiện công việc:

|  |  |
| --- | --- |
| Thời gian thực tập | Nội dung thực tập |
| 06/02/2020 10/02/2020 |  |
| 11/02/2020  18/02/2020 |  |
| 19/02/2020  26/02/2020 |  |
| 27/02/2020  02/03/2020 |  |
| 03/03/2020  15/03/2020 |  |
| 16/03/2020  02/05/2020 |  |

* 1. **Quá trình thực hiện công việc.**
     1. **Tìm hiểu về git và một số lệnh cơ bản.**
* **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.



Hình 1: Tổng quan về git.

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.

* **Github và gitlab**
* **Cài đặt và sử dụng git**

Để có thể sử dụng được Git, bạn phải cài ứng dụng Git vào máy tính để có thể sử dụng các chức năng của Git.

**Cài Git vào Linux**

Nếu bạn đang sử dụng hệ điều hành Ubuntu/Debian thì có thể sử dụng lệnh sau để cài Git.

$ sudo apt-get install git

Hoặc lệnh sau để cài trên CentOS/Fedora/RHEL.

$ yum install git

**Cài Git vào Mac OS**

Đối với Mac, bạn có thể sử dụng file installer tải tại địa chỉ http://git-scm.com/download/mac để cài đặt.

**Cài Git vào Windows**

Nếu bạn dùng Windows thì có thể tải file .exe cài đặt Git tại địa chỉ http://git-scm.com/download/win. Khi cài bạn có thể để nguyên tùy chọn mặc định mà không cần tùy chỉnh gì thêm nếu bạn chưa hiểu về nó.

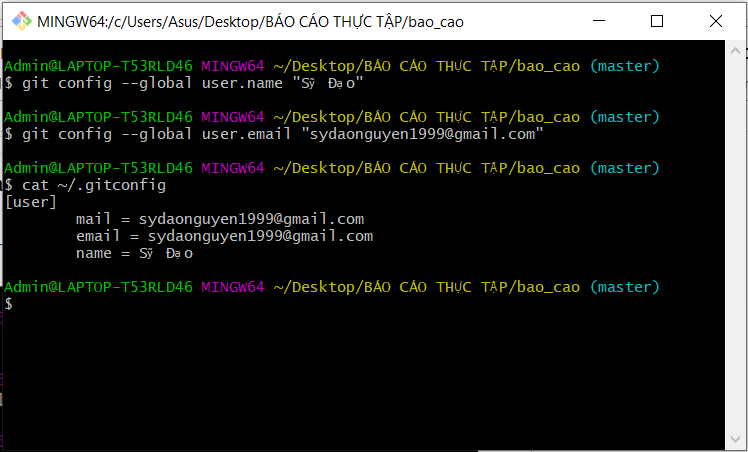
Sau khi cài đặt Git vào Windows, bạn sẽ cần mở ứng dụng Git Bash lên để bắt đầu sử dụng các dòng lệnh của Git.

**Thiết lập chứng thực cá nhân**

Sau khi cài Git xong, việc đầu tiên bạn nên làm là khai báo tên và địa chỉ email vào trong file cấu hình của Git trên máy. Để làm điều này bạn sẽ cần sử dụng hai lệnh sau đây để thiết lập tên và email.

$ git config --global user.name "Tên hiện thị của bạn"

$ git config --global user.email “gmail của bạn”



Hình 3: Thiết lập chứng thực cá nhân.

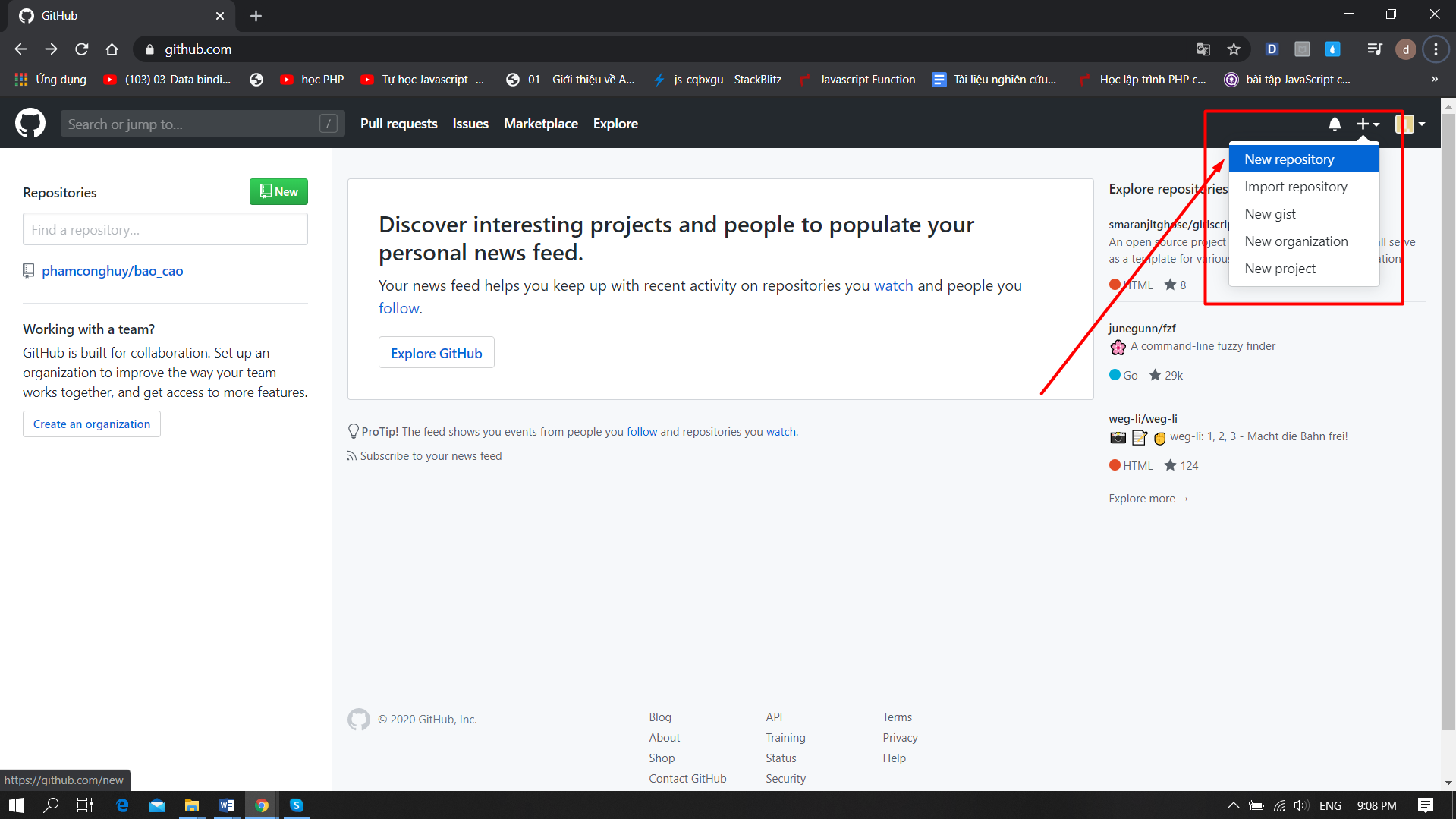
Sau khi thiết lập xong, bạn có thể kiểm tra thông tin chứng thực trên user của bạn bằng cách xem thông tin ~/.gitconfig

Hoặc bạn cũng có thể dùng lệnh git config --list để ghi danh sách các thiết lập hiện tại mà bạn đã làm.

Như vậy là bạn đã xong bước ban đầu đó là cài đặt Git và thiết lập tên và email của mình vào Git để bắt đầu làm việc.

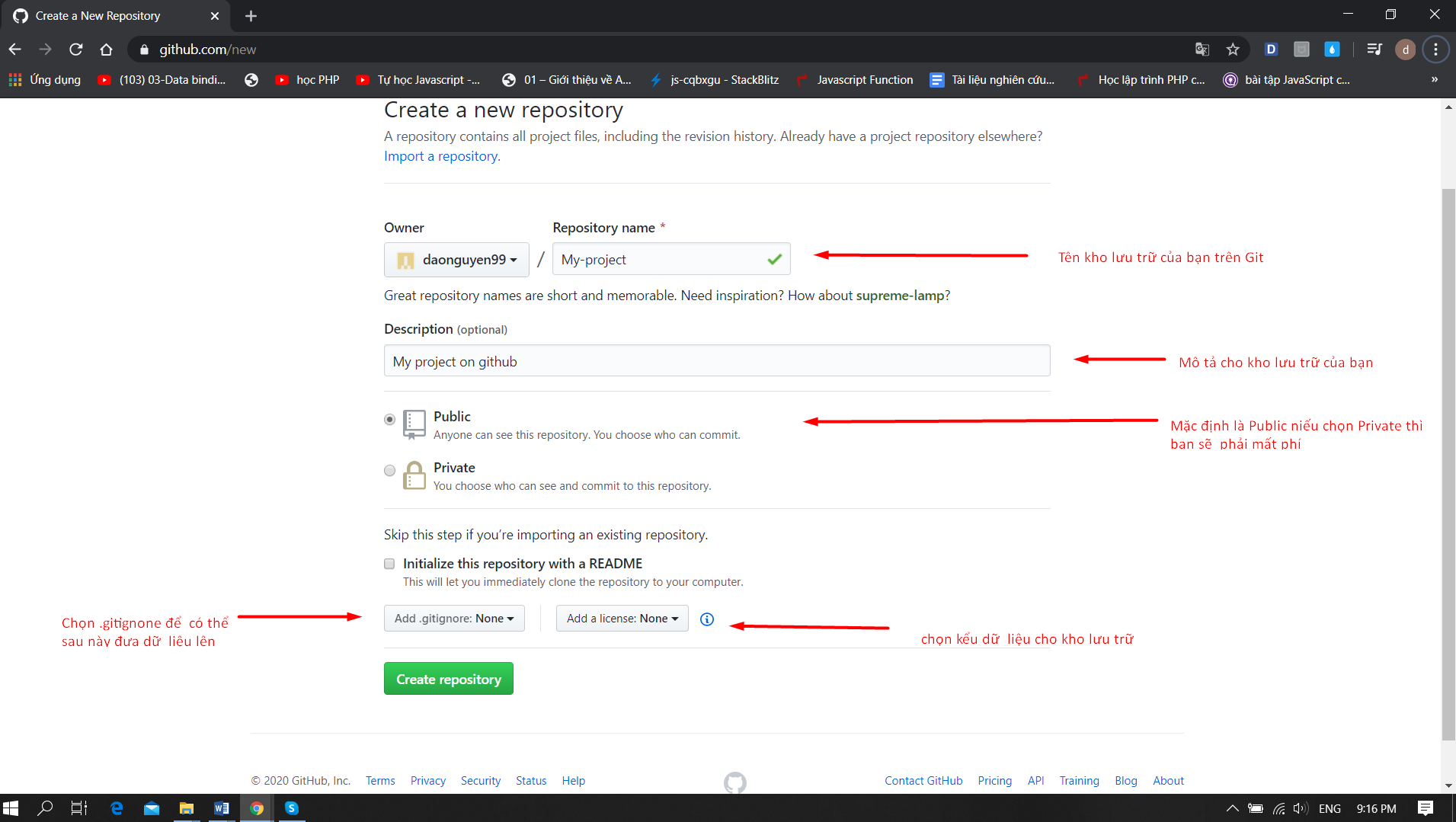
* **Tạo repository trên GitHub và làm việc**

Để tạo một repostory mới trên GitHub thì đầu tiên bạn phải đăng nhập tài khoản và vào trang chủ của GitHub từ địa chỉ <https://github.com/>. sau đó ấn vào dấu + trên menu và chọn

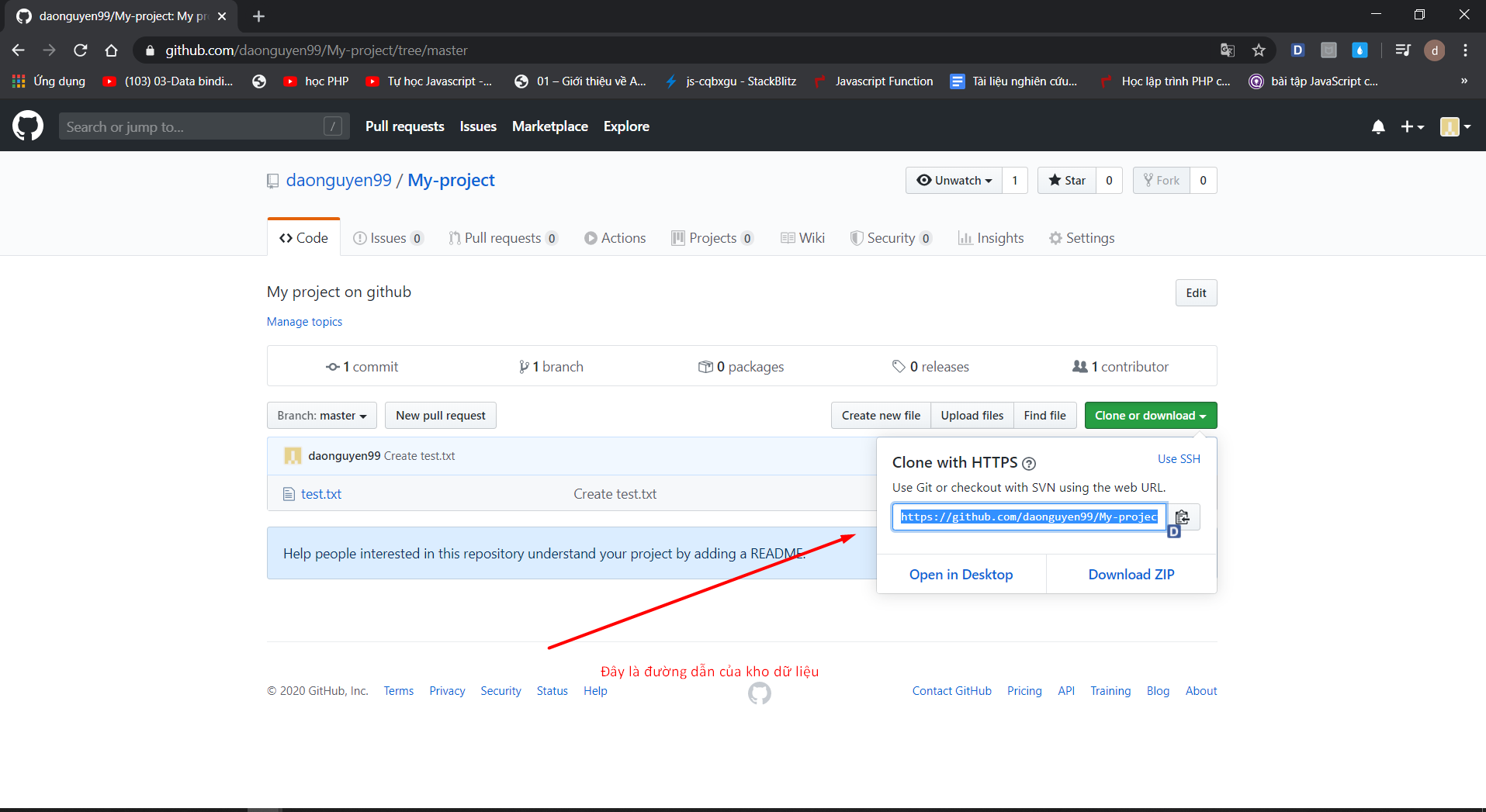


Hình 2.2.1.2 Tạo repostory trên GitHub

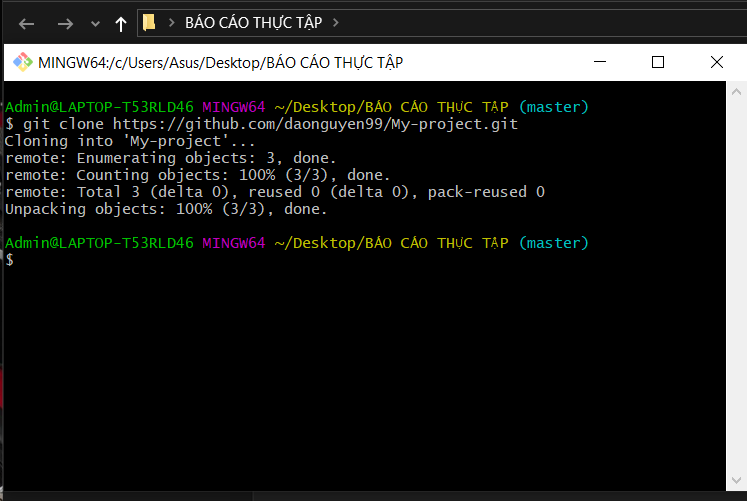
Một số cài đặt ban đầu cho kho lưu trữ của bạn.



Hình 2.2.1.3 Tạo repostory trên GitHub



Hình 2.2.1.4 Tạo repostory trên GitHub



Hình 2.2.1.5 Tạo repostory trên GitHub

Giờ thì bạn có thể clone kho dữ liệu về máy từ đường dẫn bằng lệnh “ git clone + đường dẫn ”

* **Những lệnh git cơ bản thường dùng.**

Nói đến những lệnh git thì mình nghĩ ngay đến git thần chú và git 6 bước mà mình được học từ tiền bối trong team.

Git thần chú sẽ là “ git pull origin develop ”

Git 6 bước lần lượt là :

1. “ thần chú ”.
2. Code task được giao sau khi hoàn thành thì tiếp bước 3.
3. Lại 1 lần nữa dùng “ thần chú ”.
4. git add .
5. git commit –a –m “( đây là message ) nội dung task mình làm ”.
6. Git push origin Dao ( Dao là tên nhánh mình đang làm).

Và đây cũng là những lệnh git cơ bản mà mình được học trong quá trình thực tập giờ chúng ta sẽ cùng nhau tìm hiểu cụ thể những lệnh này.

1. “ **git pull origin develop** ” ( develop là tên nhánh mà bạn cần pull ).

Lệnh này sẽ giúp chúng ta gộp những thay đổi mới và kéo về từ máy chủ từ xa với nhánh hiện tại trên máy local của bạn.

Để dễ hình dung thì bạn có thể xem lại git 6 bước ở trên và nó giúp chúng ta lấy kéo được những thay đổi mới nhất từ những thành viên trong team về máy local và tránh được những xung đột trong quá trình làm.

1. “ **git add tên\_files**”

Lệnh này được dùng để cập nhật những file có sự thay đổi như vào trạng thái commit và để cập nhật hết tất cả các files thì bạn thay tên\_files thành dấu “ . ” .

1. “ **git commit -a –m “** *message này bạn có thể dùng để note lại những thay đổi bạn đã làm để sau này dễ tìm lại* **”**

Commit nghĩa là một action để Git lưu lại một snapshot của các sự thay đổi trong thư mục làm việc. Và các tập tin, thư mục được thay đổi đã phải nằm trong Staging Area. Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của code kèm theo thông tin người commit. Ngoài ra trong Git bạn cũng có thể khôi phục lại tập tin trong lịch sử commit của nó để chia cho một branch khác, vì vậy bạn sẽ dễ dàng khôi phục lại các thay đổi trước đó. ”

1. **“ git push origin Dao (***Dao là tên nhánh mình đang làm* **) ”.**

Sau khi đã commit, thông tin mới chỉ được cập nhật lên kho lưu trữ cục bộ. Nếu muốn cập nhật lên server thì bạn phải sử dụng câu đến push.

1. **“ git branch”**

Khi làm việc với git thì bạn có thể tạo ra nhiều branch **(***nhánh***)** khác nhau câu lệnh trên dùng để kiểm tra branch hiện tại.

Để tạo mới 1 branch thì **“ git branch <name\_branch> ”**

Checkout một nhánh thì ta dùng **“ git checkout<name\_branch> ”**

Để tạo mới và checkout luôn nhánh đó thì bạn dùng lệnh **“ git checkout –b <name\_branch>”.**

Mặc định ban đầu của bạn sẽ là nhánh master trước khi làm việc thì bạn có cần checkout sang nhánh con của mình. Khi đã kết thúc hoặc muốn quay lại branch master thì bạn dùng câu lệnh **“ git checkout master”.**

* + 1. **Cấu hình project Laravel trên local.**
    2. **Tìm hiểu laravel cơ bản và bài test.**
    3. **Tìm hiểu về Laravel Lumen và tham gia dự án thực tế.**
    4. **Tham gia thực tế ( dự án update giao diện đơn giản).**
  1. **Công tác bảo trì sản phẩm . . .**