

GIT DOCUMENT

Git là một hệ thống quản lý phiên bản phân tán (**Version Control System**). Git mã nguồn mở và miễn phí được thiết kế nhằm quản lý và làm việc với các project từ nhỏ đến lớn.

Tham khảo từ git-scm.com

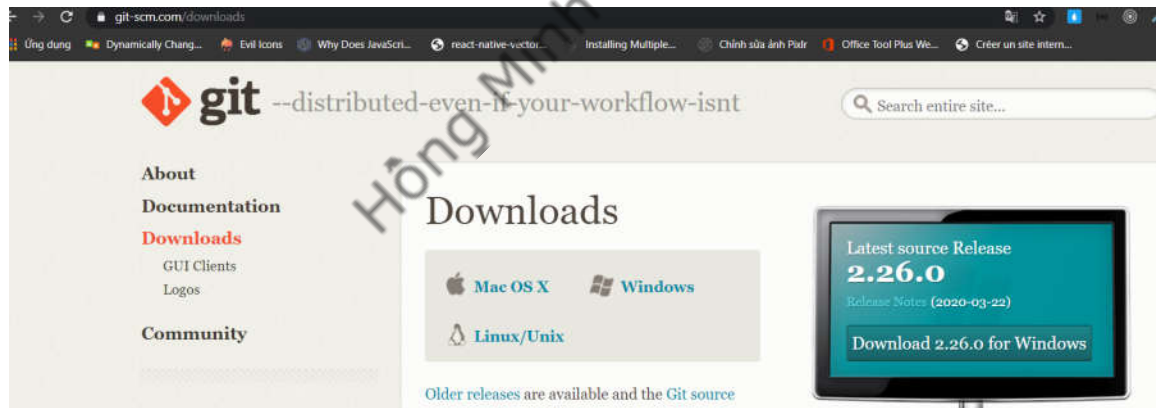
=> **Nói theo kiểu người bình thường học code:** Nó là một công cụ để quản lý và chia sẻ source code.

1. Hướng dẫn cài đặt(Setup Environment):

Đầu tiên các bạn tải phần mềm quản lý của git về nhé tại trang:

<https://git-scm.com/downloads>

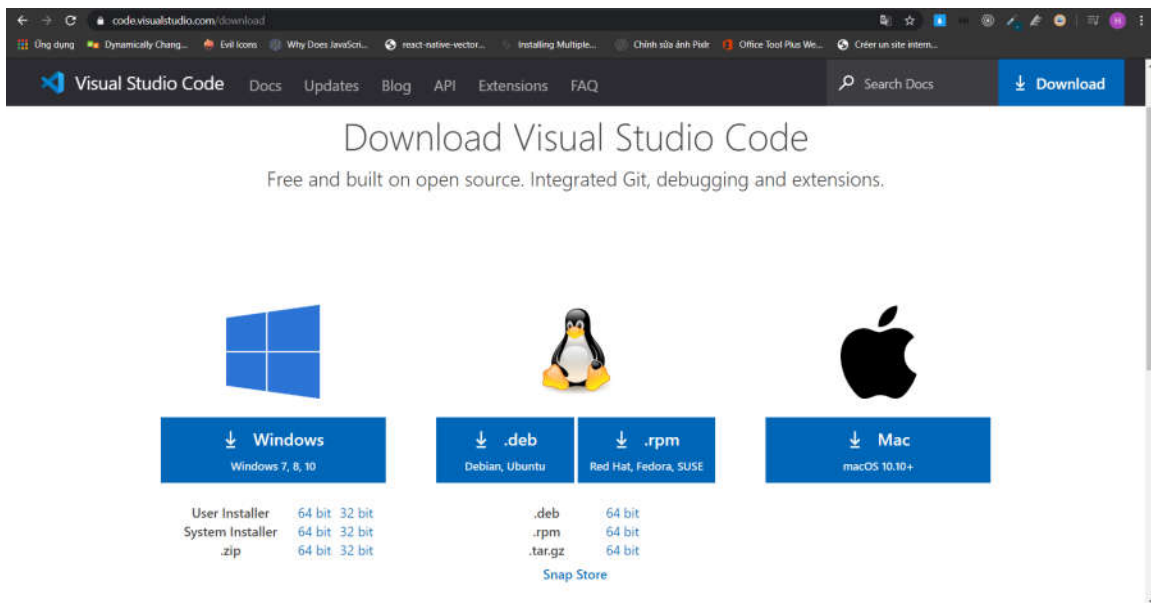
Chọn phiên bản cho hệ điều hành tương ứng.



Bấm vào và đợi tải thôi.

Tiếp theo mình sẽ hướng dẫn trên VSCode nên các bạn nên cài VSCode để mình tiện support nhé.

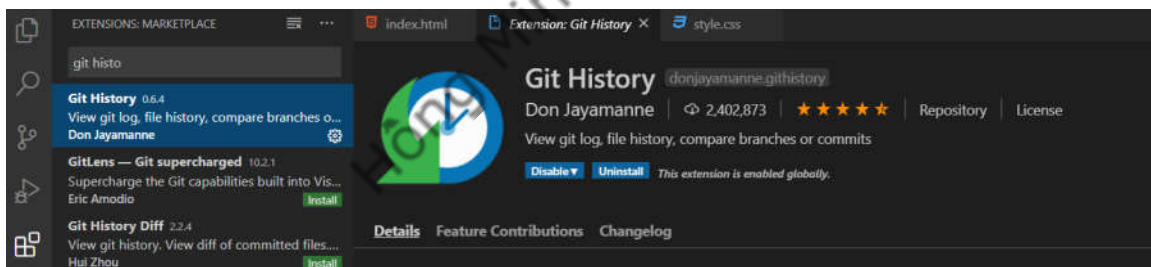
<https://code.visualstudio.com/download>



Chọn phiên bản tương ứng cho hệ điều hành.

Các bạn mở Visual Studio Code cài extension hỗ trợ git này nhé:

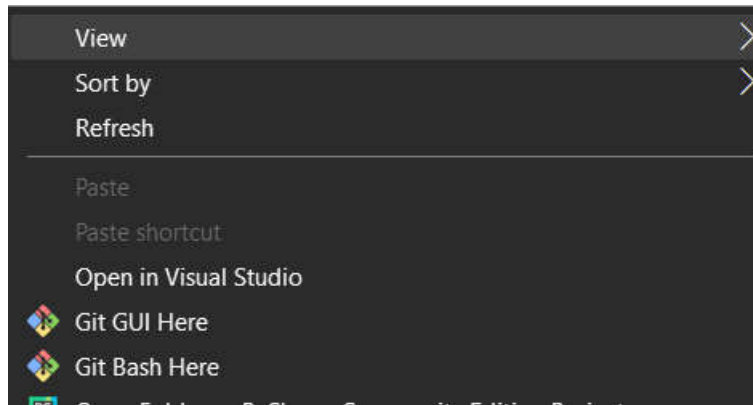
Chọn mục Extensions gõ Git History và install thôi nó sẽ tự động kích hoạt cho các bạn.



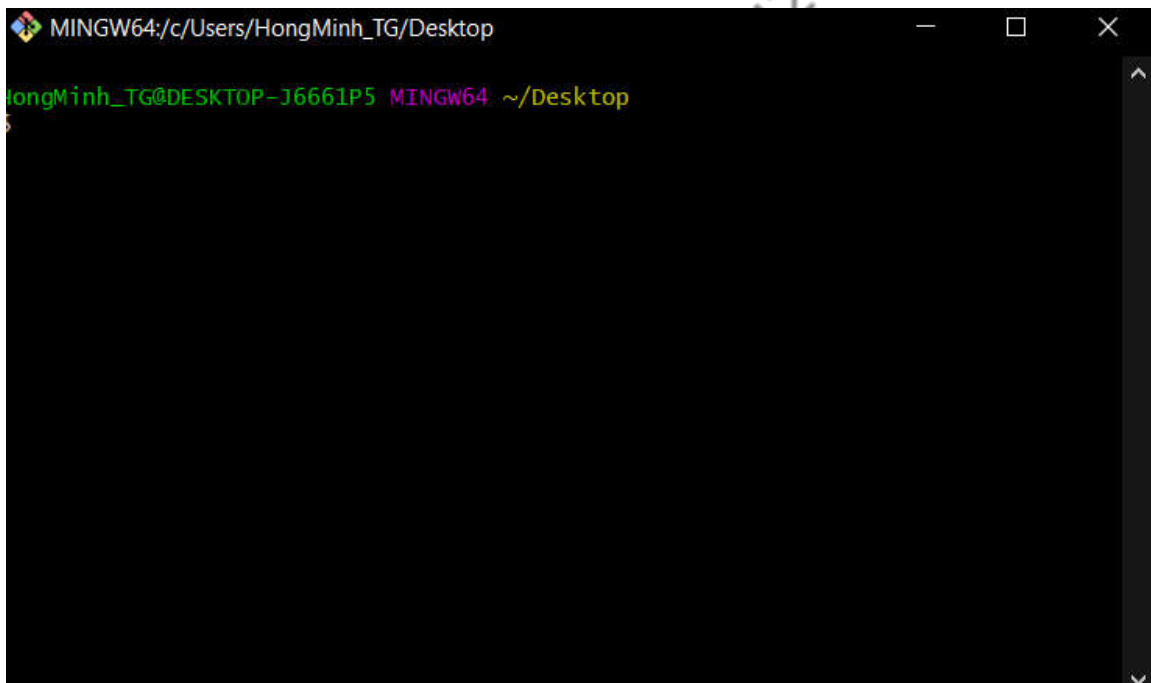
Mình sẽ dùng Server Git của **github** vì nó khá nổi tiếng rồi :D. Các bạn truy cập <https://github.com/> để đăng ký tài khoản và nhớ là phải xác nhận tài khoản qua email nhé (Tức là vào email sẽ thấy mail của github gửi các bạn nhấn vào đường dẫn nó gửi để xác nhận) và đăng nhập vào **github**.

a. Tạo mã SSH trên máy tính:

Lý do chúng ta tạo mã này nhằm ủy quyền và kết nối với **github**.
Sau khi cài phần mềm quản lý của Git khi click chuột phải sẽ thấy:



Chọn **Git Bash Here** nhé mọi người:



Và đây là giao diện của Git Bash:

Ta thực hiện lệnh như hướng dẫn để kiểm tra xem mã SSH đã tồn tại chưa (tức là máy chúng ta đã có mã trước đó chưa):

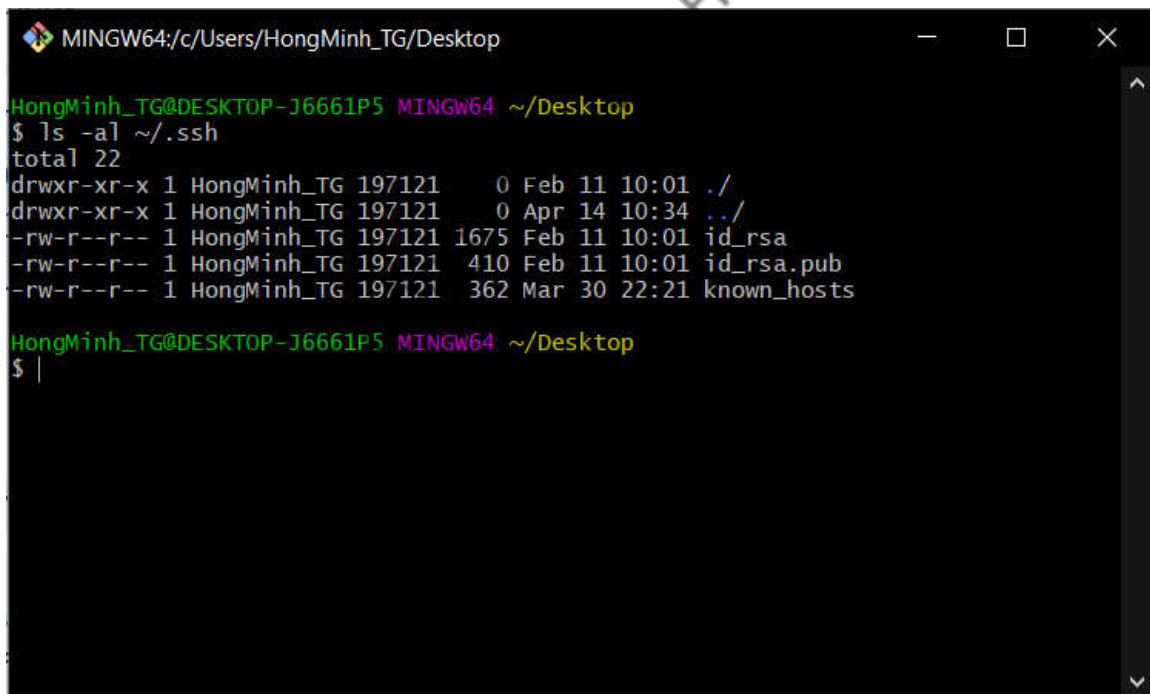
- 1 Open Git Bash.
- 2 Enter `ls -al ~/.ssh` to see if existing SSH keys are present:

```
$ ls -al ~/.ssh
# Lists the files in your .ssh directory, if they exist
```

- 3 Check the directory listing to see if you already have a public SSH key. By default, the filenames of the public keys are one of the following:
 - `id_rsa.pub`
 - `id_ecdsa.pub`
 - `id_ed25519.pub`

Ta dùng lệnh: **ls -al ~/.ssh**

Đây là máy mình chạy lệnh:

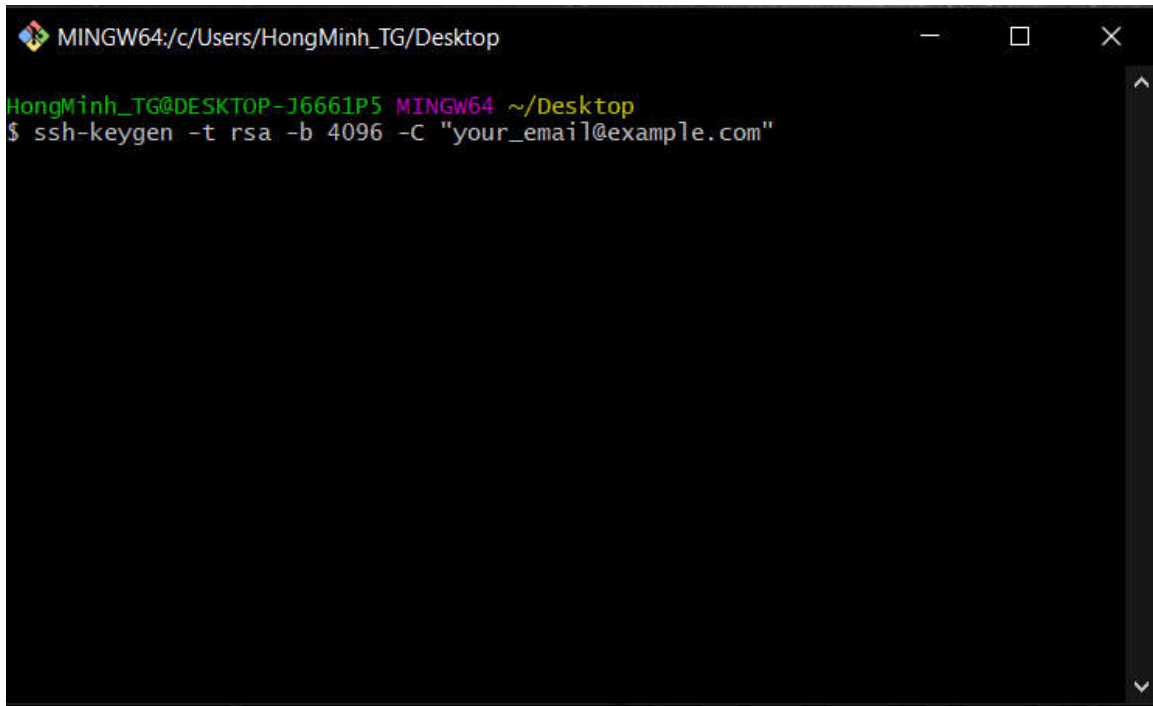


```
MINGW64:/c/Users/HongMinh_TG/Desktop
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ ls -al ~/.ssh
total 22
drwxr-xr-x 1 HongMinh_TG 197121  0 Feb 11 10:01 ./
drwxr-xr-x 1 HongMinh_TG 197121  0 Apr 14 10:34 ../
-rw-r--r-- 1 HongMinh_TG 197121 1675 Feb 11 10:01 id_rsa
-rw-r--r-- 1 HongMinh_TG 197121  410 Feb 11 10:01 id_rsa.pub
-rw-r--r-- 1 HongMinh_TG 197121  362 Mar 30 22:21 known_hosts
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ |
```

Nó hiển thị các file `id_rsa` thì tức là mã SSH đã được tạo trước đó rồi nha mọi người :D.

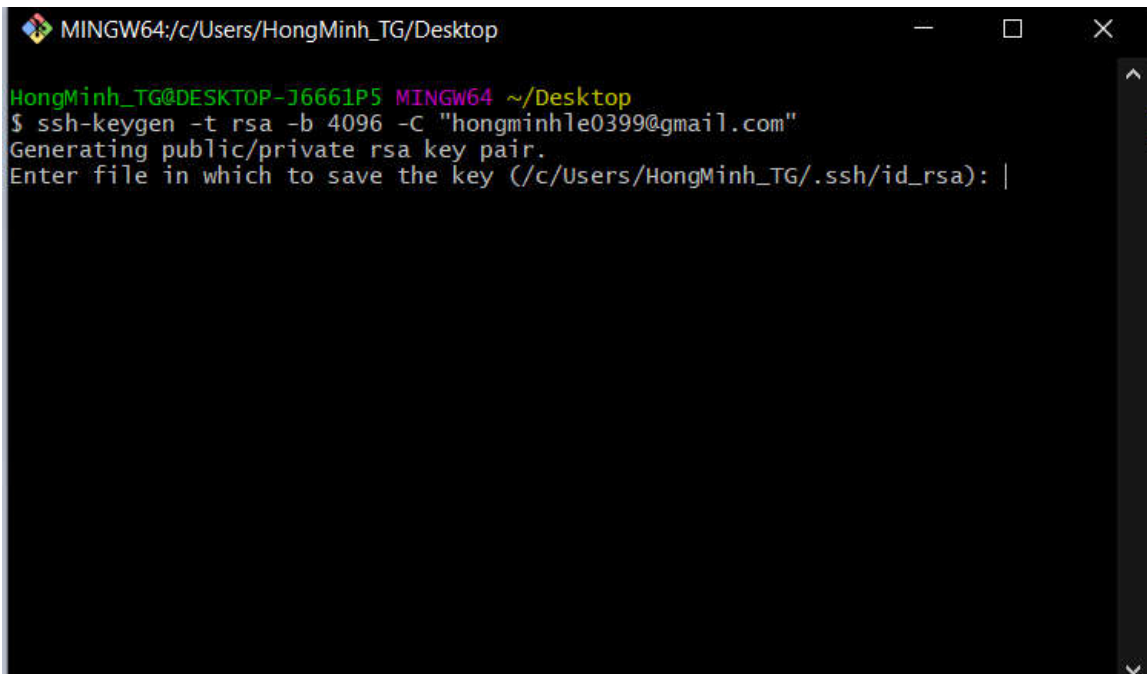
Còn ai chưa tạo thì xem tiếp:

1. Các bạn chạy lệnh này đầu tiên:



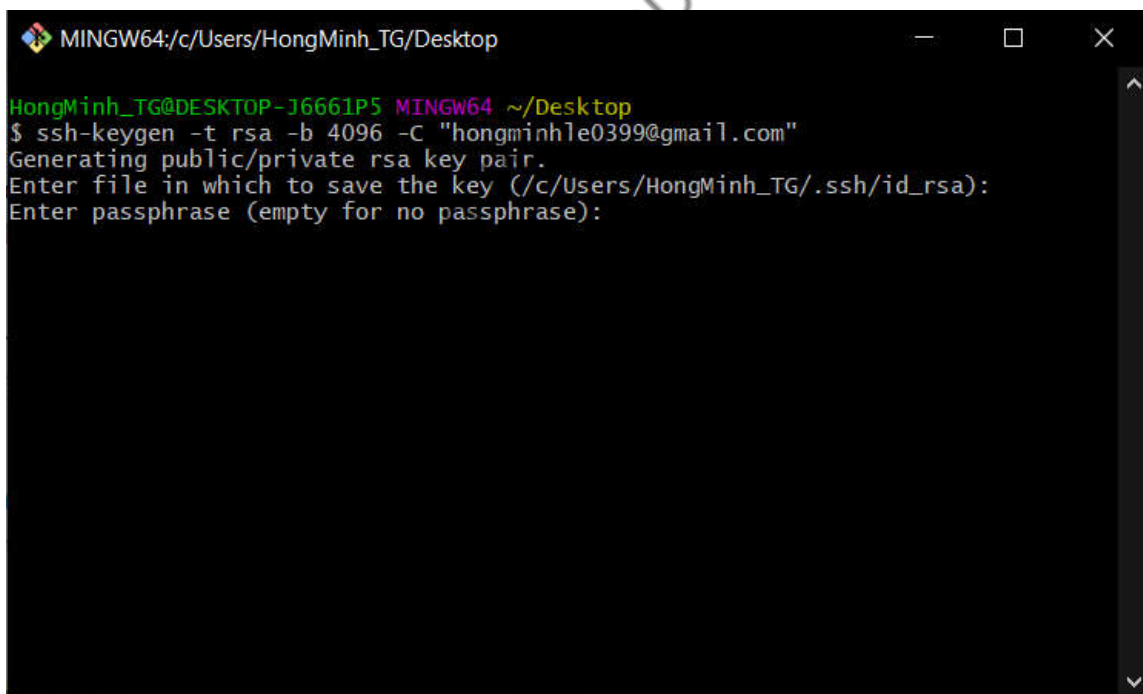
```
MINGW64:/c/Users/HongMinh_TG/Desktop
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Các bạn thay **your email ...** thành email của các bạn đăng ký tại khoản github nhé :D.



```
MINGW64:/c/Users/HongMinh_TG/Desktop
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ ssh-keygen -t rsa -b 4096 -C "hongminhle0399@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/HongMinh_TG/.ssh/id_rsa): |
```

Mình thì không gõ gì Enter luôn để nó lưu key ở nơi mặc định.



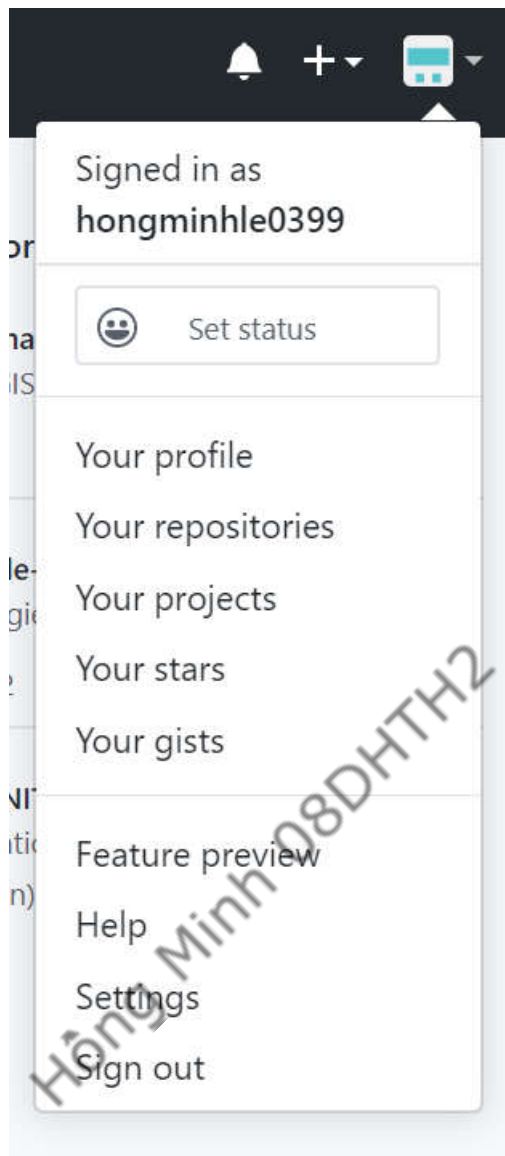
```
MINGW64:/c/Users/HongMinh_TG/Desktop
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ ssh-keygen -t rsa -b 4096 -C "hongminhle0399@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/HongMinh_TG/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
```

Cái này mình để trống nhé nó là cái giúp SSH key bảo mật các bạn có thể tự tìm hiểu thêm nhé. **Enter** tiếp thôi.

```
MINGW64:/c/Users/HongMinh_TG/Desktop
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ ssh-keygen -t rsa -b 4096 -C "hongminhle0399@gmail|.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/HongMinh_TG/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/HongMinh_TG/.ssh/id_rsa.
Your public key has been saved in /c/Users/HongMinh_TG/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:a6o8EUdQwnFaVqbQyJpZUXkItNlKTBBuE0zU6jmbXEk hongminhle0399@gmail.com
The key's randomart image is:
+---[RSA 4096]-----+
|  +**@X*+o          |
| ..B*O=o.          |
|  =**...           |
| .+=E .            |
| . o + S           |
| + + .             |
| . = . o           |
| +.. o             |
|  oo.              |
| +-----[SHA256]-----+
```

Yeah đã tạo xong rồi bây giờ chỉ cần đưa nó vào tài khoản github của tụi mình thôi :D.

Các bạn nhấn vào biểu tượng đầu robot(hay gì)đó góc phải trên cùng nhé.



Chọn **Settings**

Personal settings

Profile

Account

Security

Security log

Emails

Notifications

Billing

SSH and GPG keys

Blocked users

Repositories

Organizations

SSH keys

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.

Các bạn chọn **SSH and GPG Keys**. Rồi chọn New SSH Key:

SSH keys

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

Sẽ ra được kết quả như sau:

SSH keys / Add new

Title

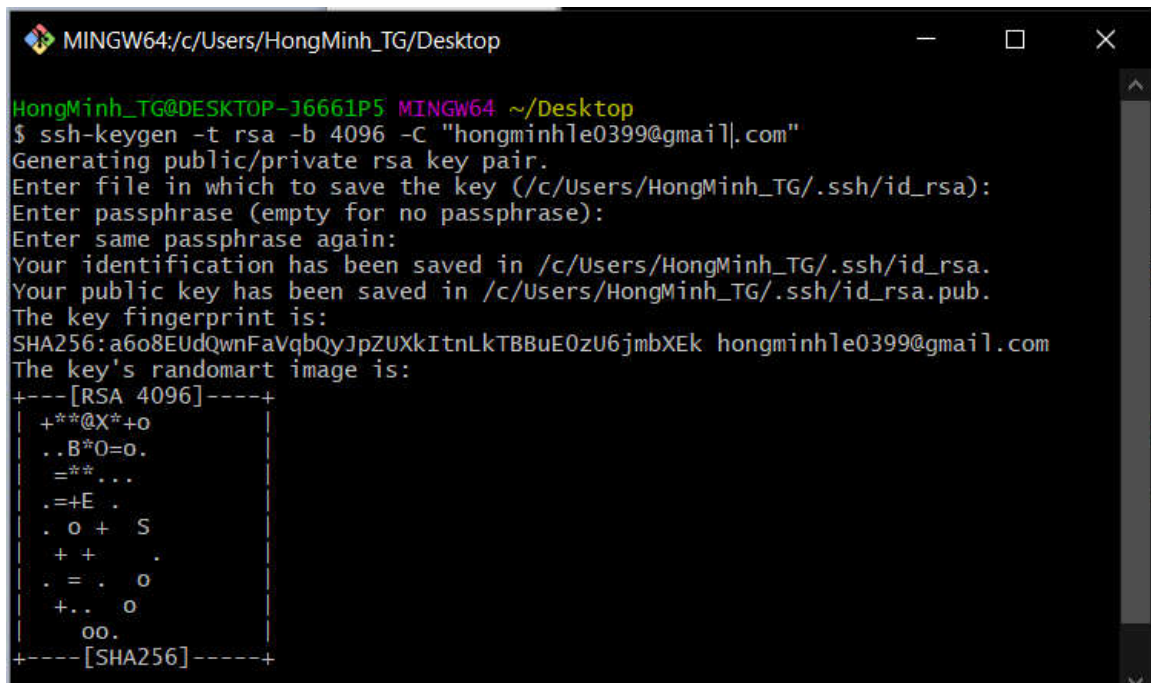
Key

Begins with 'ssh-rsa', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

Các bạn nhập title là gì cũng được nhé mình nhập là **HM SSH Key**

Ở dưới các bạn push code SSH Key vào nhé:



```
MINGW64:/c/Users/HongMinh_TG/Desktop
HongMinh_TG@DESKTOP-J6661P5 MINGW64 ~/Desktop
$ ssh-keygen -t rsa -b 4096 -C "hongminhle0399@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/HongMinh_TG/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/HongMinh_TG/.ssh/id_rsa.
Your public key has been saved in /c/Users/HongMinh_TG/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:a6o8EUdQwnFaVqbQyJpZUXkItNLTBBuE0zU6jmbXEk hongminhle0399@gmail.com
The key's randomart image is:
+---[RSA 4096]-----+
|  +**@X*+o          |
|  ..B*O=o.          |
|  =**..             |
|  .+=E.             |
|  .o+ S              |
|  + + .              |
|  . = . o            |
|  +.. o              |
|    oo.              |
|-----[SHA256]-----+
```

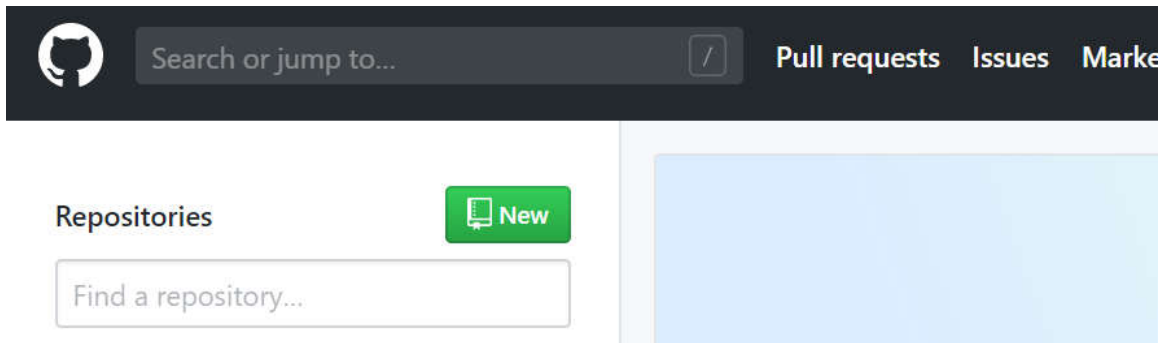
Rồi okay các bạn thấy dòng **Your public key has been saved...** chứ:

Tìm vào thư mục đó mở file đó bằng notepad và copy toàn bộ nội dung file đó bỏ vào ô key thôi :D. Done! Phù.

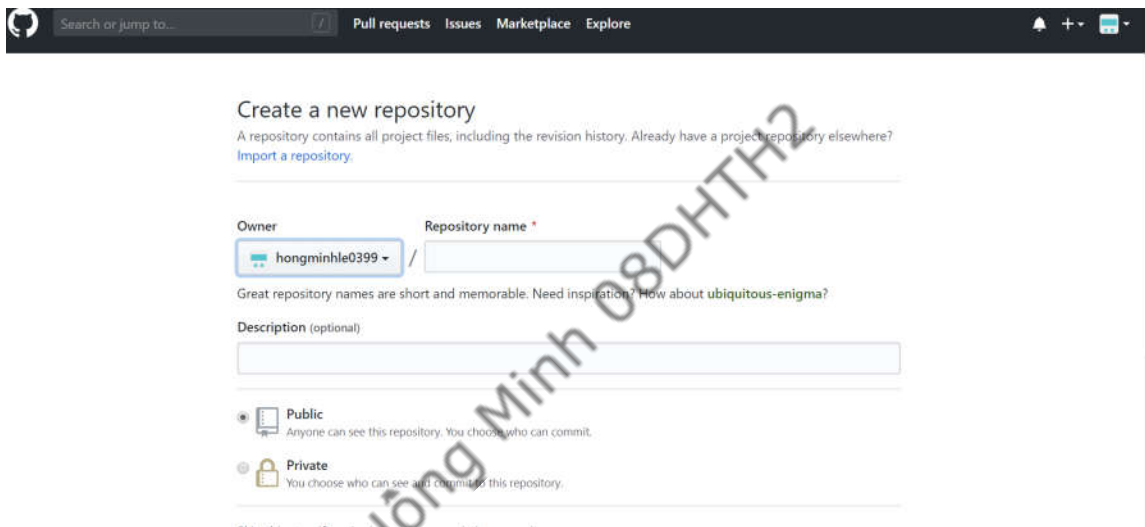
Note: Nếu muốn xóa lệnh SSH Key và tạo SSH Key mới thì các bạn có thể dùng lệnh này để xóa nhé:

rm -f ~/.ssh/id_rsa*

b. Cách tạo một repository trên **github**:

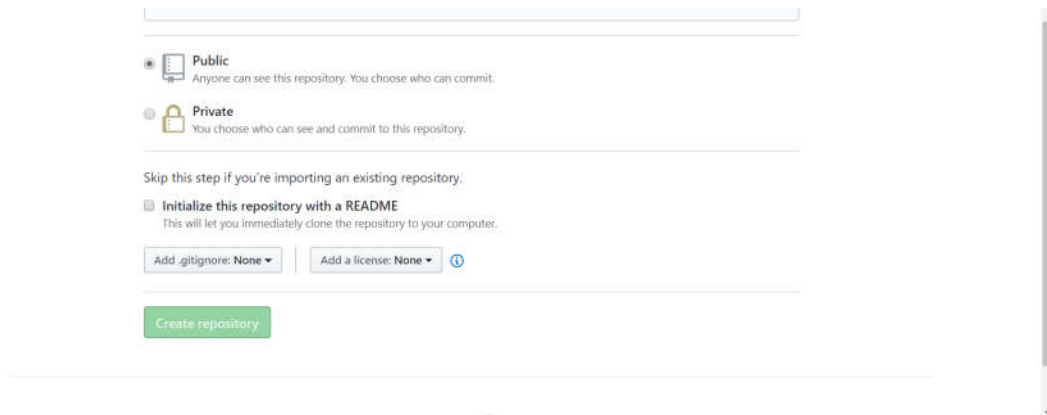


1. Chọn **New** nhé.
2. Nó sẽ hiển thị như thế này:



Các bạn gõ tên project mình vào ô **Repository Name**. Phần **Description** nó là **optional** tức là tùy ý các bạn muốn mô tả dự án hay không cũng được có thể không nhập và bỏ qua.

Ở phần **public/private**: thường thì đây là lựa chọn công khai sourcecode hay không. Thường lựa chọn private khi làm công ty thôi vì source không để lộ ra ngoài được :D. Tụi mình chọn public nhé.



Mấy cái dưới các bạn làm giống mình là được :D. Nhớ là phải nhập tên project(**Repository Name**) nhé không thì nút **Create repository** sẽ bị mờ nhé.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

 hongminhle0399 ▾

Repository name *

GitExample ✓

Great repository names are short and unique. GitExample is available. Inspiration? How about [scaling-octo-tribble](#)?

Description (optional)

☒ **Public**

Anyone can see this repository. You choose who can commit.

☐ **Private**

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

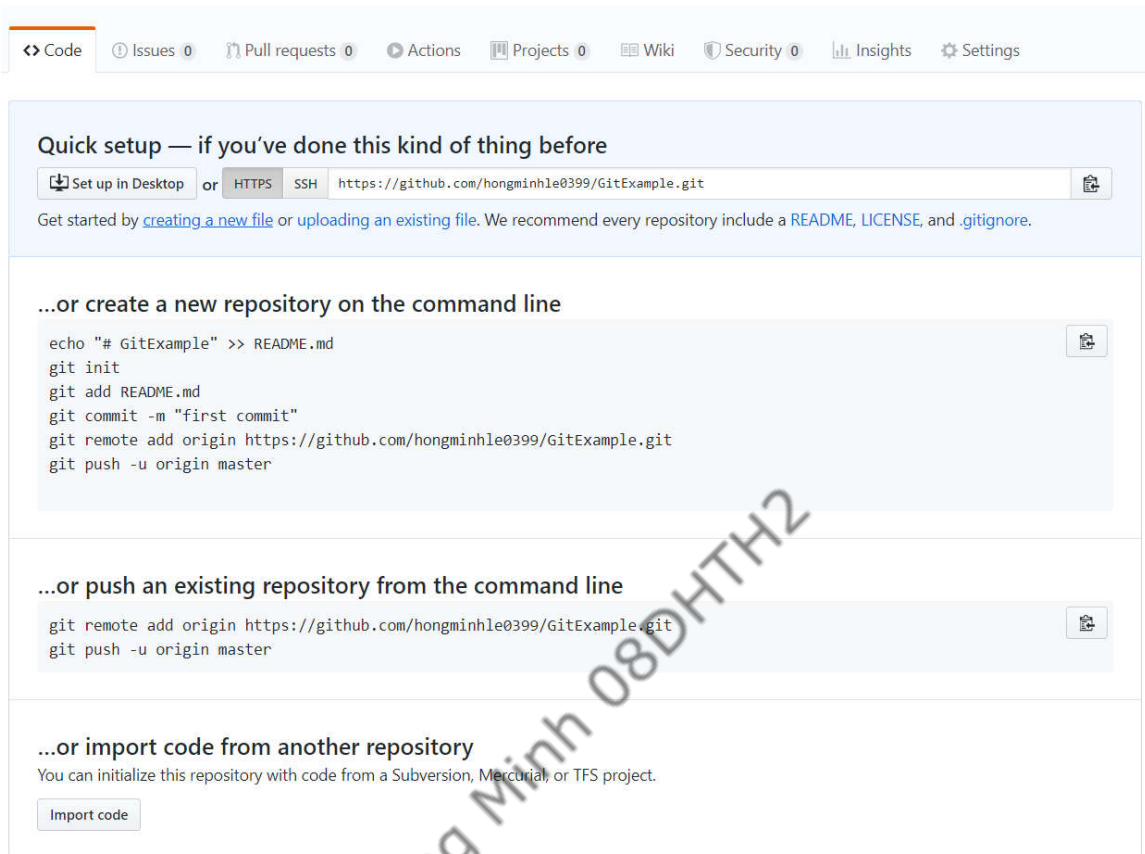
Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

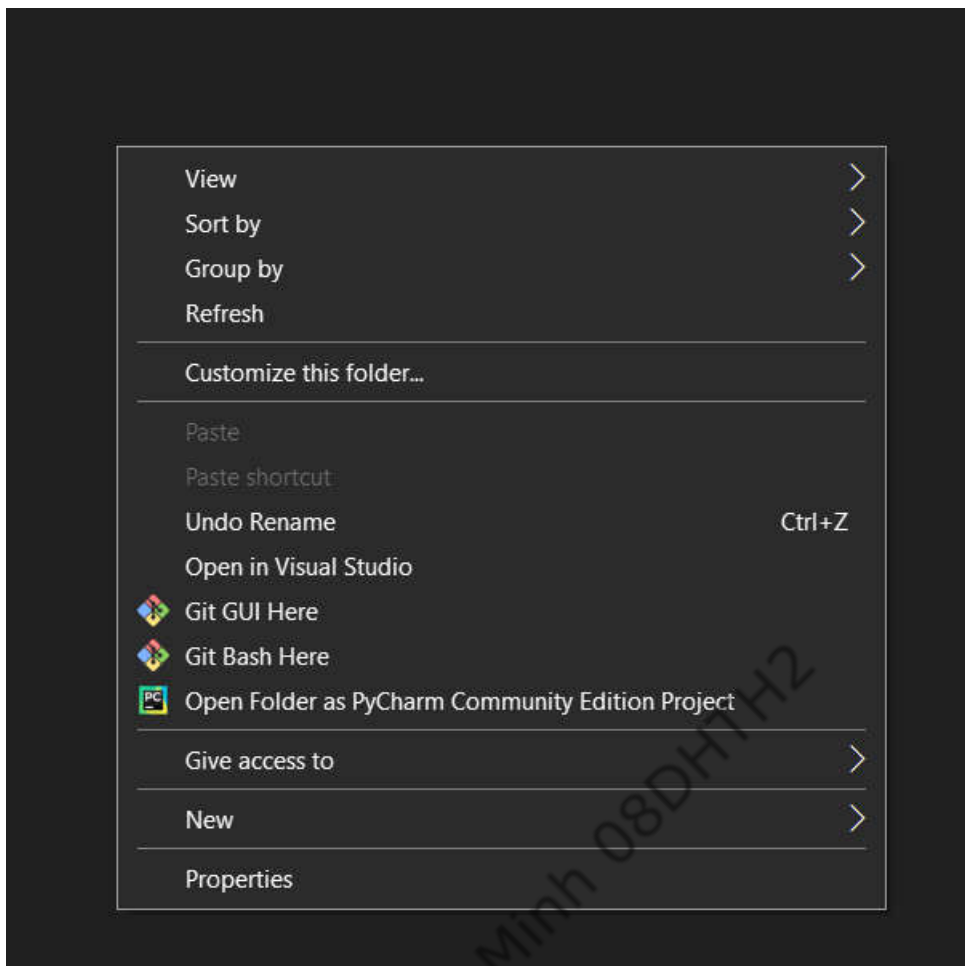
Sau khi tạo xong các bạn sẽ thấy được repository vừa tạo nhé:



Các bạn thực hiện các bước sau:

Tạo một thư mục nhé tên đặt sao cũng được (khuyến khích đặt tên giống với tên của Repo):

Lưu ý: Click chuột phải vào khoảng trống của thư mục.



Nếu bạn muốn khởi tạo mới một project thì chạy lệnh ở Textbox1:
Copy paste rồi **Enter** nhé không lại chạy xót lệnh.

```
HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample
$ egit add README.md
cho "# GitExample" >> README.md
git commit -m "first commit"
git remote add origin https://github.com/hongminhle0399/GitExample.git
git push -u origin master
HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample
$ git init
Initialized empty Git repository in D:/Project/GitExample/.git/

HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$ git commit -m "first commit"
[master (root-commit) 3e9c4fa] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$ git remote add origin https://github.com/hongminhle0399/GitExample.git
```

Kết quả:

```
HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$ git commit -m "first commit"
[master (root-commit) 3e9c4fa] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$ git remote add origin https://github.com/hongminhle0399/GitExample.git

HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 223 bytes | 223.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hongminhle0399/GitExample.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

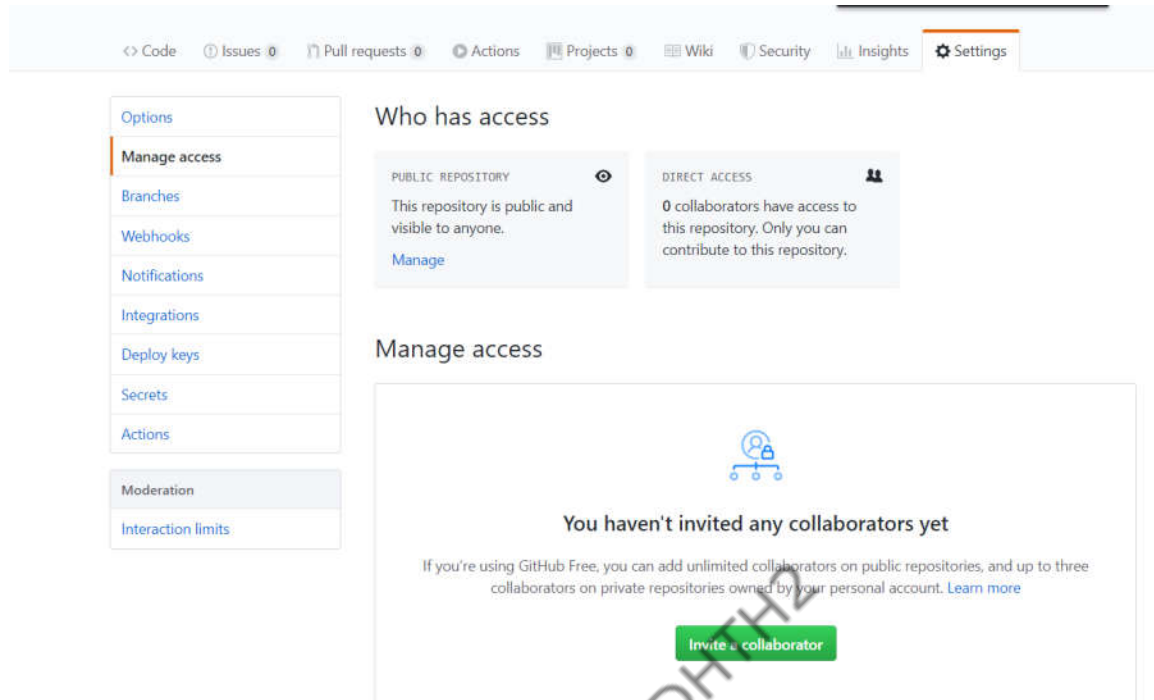
HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/GitExample (master)
$
```

Hoặc nếu muốn đẩy một local repo có sẵn lên remote repo vừa tạo thì ta có thể chạy lệnh ở dưới:

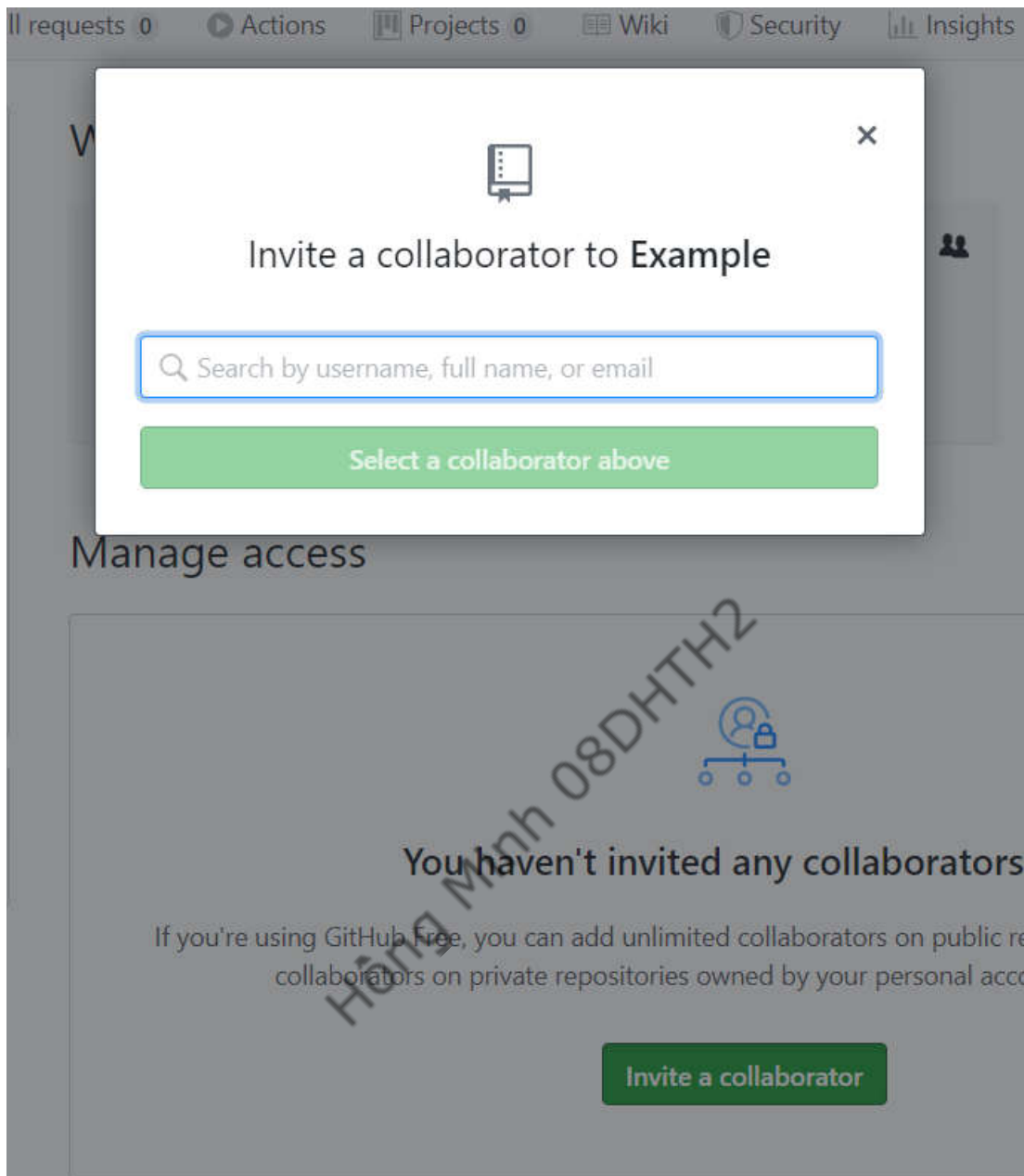
Do mình ví dụ bằng lệnh tạo rồi nên cái này các bạn tự làm nhé :D

Cách mời người khác tham gia dự án chung:

Các bạn chọn **Settings -> Manage access**



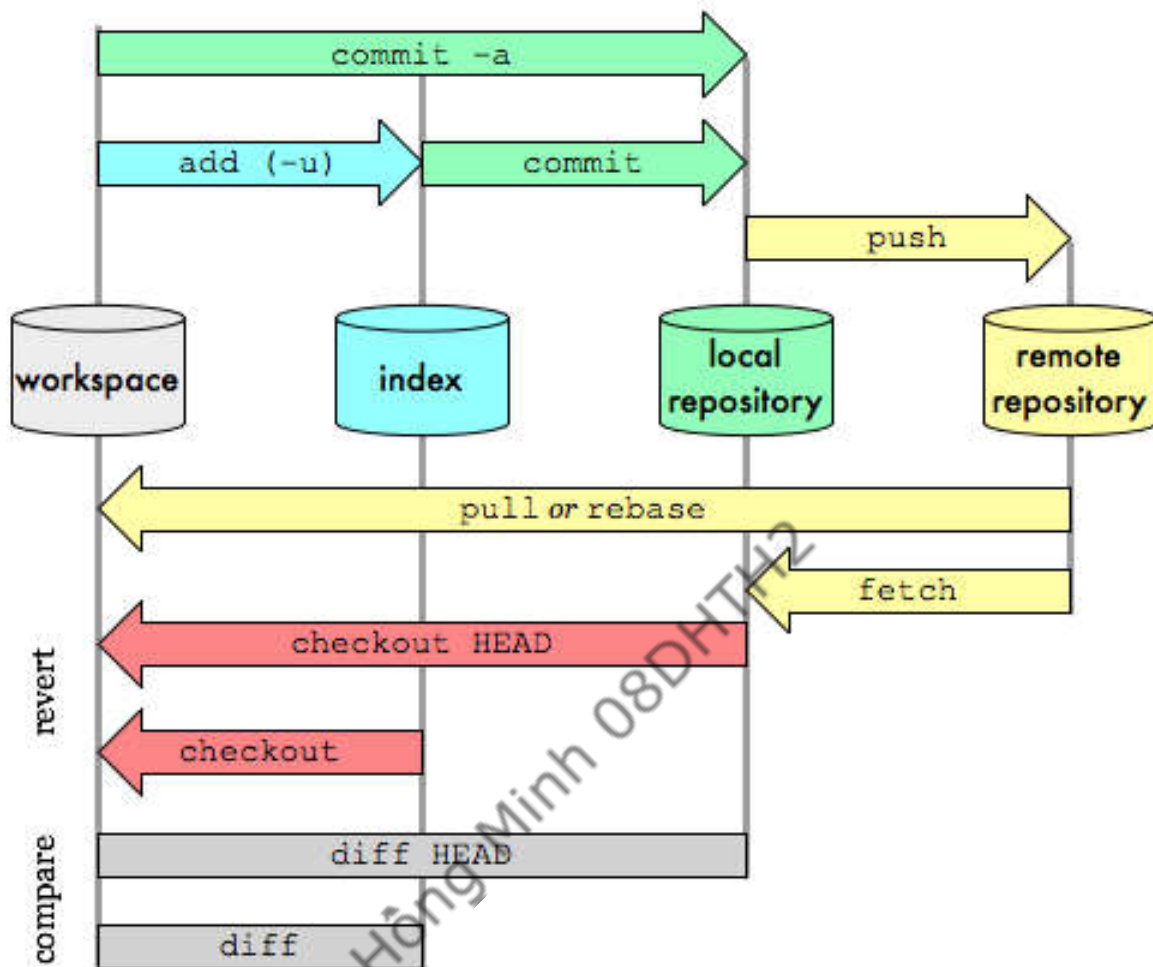
Chọn **Invite a collaborator**. Sau đó gõ mail hoặc tên tài khoản github mà bạn muốn mời tham gia nhé :D.



2. Các định nghĩa:

Git Data Transport Commands

<http://osteele.com>



a. Repository:

- Là nơi lưu trữ project và những thứ cần thiết để quản lý project.
- Có 2 loại repo(gọi tắt):
 - **Local Repository**: là thư mục .git ở máy tính.

- **Remote Repository**: là repo trên git server mà bạn sử dụng.

b. Working Directory:

Gồm 3 lớp:

workspace: là các files trong source của chúng ta.

index(staging area): chứa code được chuyển sang trạng thái staged để sẵn sàng cho đợt commit tiếp theo.

local repo: đây là nơi chứa code được cập nhật và sẵn sàng đẩy lên remote repository.

c. HEAD:

Đây là một con trỏ đặc biệt nó sẽ references đến branch hiện tại mà chúng ta đang làm việc. Thông thường nó sẽ trỏ đến branch hiện tại với commit mới nhất được references bởi branch hiện tại. Một hiện tượng thú vị đó là **detached HEAD** khi bạn checkout vào 1 commit chứ ko phải vào branch.

Note: references = trỏ.

3. Các trạng thái file trong git:

Có 2 trạng thái:

untracked: trạng thái này xuất hiện khi bạn thêm hoặc tạo ra một file mới trong project.

tracked: file ở trong trạng thái này khi bạn thực hiện **git add** . sau khi thêm file.

Khi file ở trạng thái **tracked** thì file chắc chắn sẽ thuộc 1 trong 3

trạng thái sau: **modified, unmodified, staged**.

Như ở trên có đề cập khi thì sau khi **git add** . thì file sẽ chuyển sang trạng thái **tracked -> staged**.

Chú thích:

untracked: chưa được theo dõi bởi git.

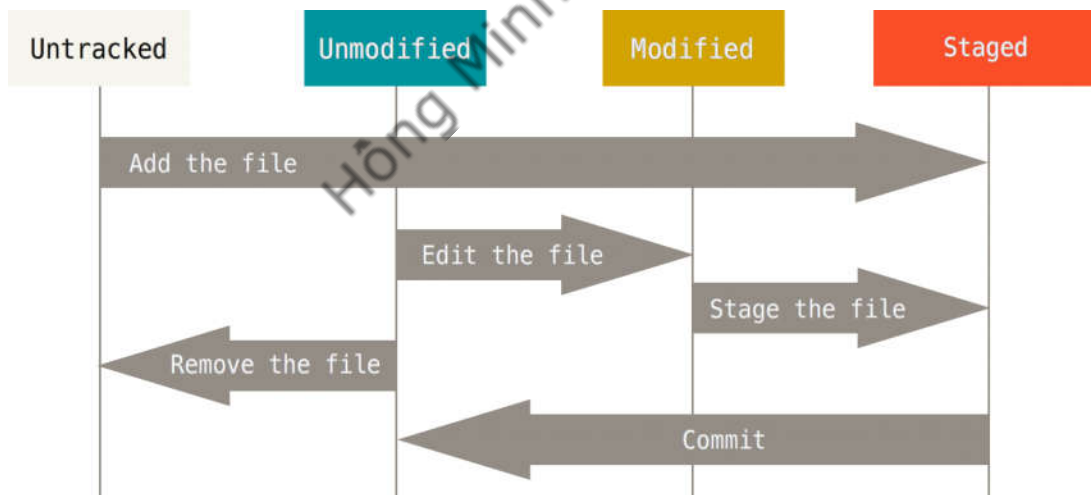
tracked: đang được git theo dõi.

Sơ đồ cho thấy sự thay đổi trạng thái của file trong GIT:

modified: sửa đổi.

unmodified: chưa được sửa đổi.

staged: đang trong giai đoạn sẵn sàng commit.



Như sơ đồ bạn có thể thấy khi bạn thêm file nó ở trạng thái **untracked** và sau khi thực hiện lệnh **git add** . thì nó sẽ chuyển sang ngay **staged**

Khi một file đã được **commit** tức là lệnh **git commit** ở phía trên thì

nó sẽ chuyển về trạng thái **unmodified**.

Khi một file ở trạng thái unmodified mà có một chỉnh sửa gì ở trong file thì hiển nhiên sẽ chuyển sang trạng thái **modified**.

File ở trạng thái sửa đổi chạy **git add** . thì tiếp tục được chuyển ngược lại **staged**.

Một file ở trạng thái unmodified mà bị remove (có lệnh **git rm**) thì sẽ tự động chuyển sang trạng thái **untracked**.

4. Các lệnh git siêu cơ bản:

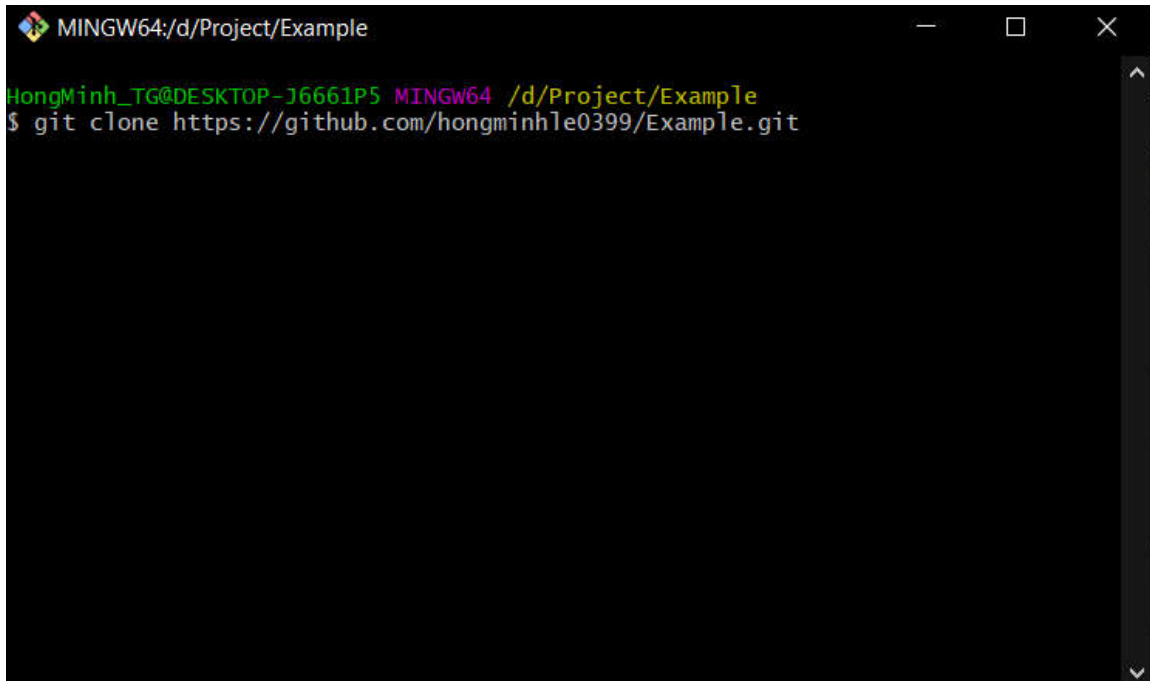
- **git init** : khởi tạo một repo.

Cái này làm ở trên rồi nhé các bạn :D muốn xài thì lên khúc tạo repo coi lại nhé :D

- **git clone** : đúng như cái tên của nó. Nó dùng để clone một project về. Nó hữu dụng khi chúng ta làm việc nhóm khi một người tạo project và share project cho những người cùng nhóm.

Ví dụ:

Các bạn có thể lựa chọn SSH hoặc HTTPS đều được:

A screenshot of a MINGW64 terminal window. The title bar shows 'MINGW64:/d/Project/Example'. The prompt is 'HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/Example'. The command entered is '\$ git clone https://github.com/hongminhle0399/Example.git'.

```
MINGW64:/d/Project/Example
HongMinh_TG@DESKTOP-J6661P5 MINGW64 /d/Project/Example
$ git clone https://github.com/hongminhle0399/Example.git
```

À nhớ mở git bash trong thư mục bạn muốn chứa project nhé :D không thì nó lại nằm tùm lum đó :D Enter và xong :D. Thường thì lần đầu tiên nó sẽ yêu cầu nhập tài khoản và mật khẩu nhe mọi người. Mật khẩu nhập thì sẽ không có chấm gì đâu nha :)) nó để ẩn hết đó nên khi gõ thấy ko hiện gì đừng nghĩ là gõ ko được nha. Bước này gõ cẩn thận nha mọi người hehe :P.

- **git fetch** : lệnh này dùng để cập nhật đồng bộ dữ liệu (metadata) của local repo và remote repo.

Ví dụ:

Có người trong team tạo ra một nhánh mới và đẩy lên remote repo thì tại local repo của chúng ta sẽ không có nhánh đó vì vậy việc của chúng ta làm là phải fetch để đồng bộ dữ liệu.

- **git pull origin <branch-name>** : lệnh này dùng để cập nhật code của một branch nào đó của remote repo cho local branch chúng ta

đang làm việc.

Ví dụ:

Leader team vừa mới update một tính năng mới lên branch cha của branch chúng ta để đảm bảo không bị lỗi **no-fast-forward** thì chúng ta phải pull code về để đảm bảo được history commit cho nhánh của chúng ta. Việc này chúng ta cũng có thể làm khi chuyển branch vì local branch của chúng ko được tự động cập nhật để có commit mới nhất từ remote branch.

git pull = git fetch + git merge

- **git merge <branch-name>** : gộp một nhánh cụ thể vào nhánh chúng ta đang đứng.

Ví dụ:

Trường hợp chúng ta thấy code đã ổn định và muốn cập nhật nó chính thức vào nhánh chính của team chúng ta thì leader sẽ thực hiện lệnh này.

Ta thử merge nhánh develop vào nhánh master nhé:

```
PS D:\Project\GitExample> git checkout master
Your branch is up to date with 'origin/master'.
PS D:\Project\GitExample> git log --oneline
a869474 (HEAD -> master, origin/master, features) them file index.html
3e9c4fa first commit
PS D:\Project\GitExample> git merge develop
Updating a869474..d7f7856
Fast-forward
 style.css | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 style.css
PS D:\Project\GitExample> git log --oneline
d7f7856 (HEAD -> master, origin/develop, develop) add a file: style.css
a869474 (origin/master, features) them file index.html
3e9c4fa first commit
PS D:\Project\GitExample> █
```

Ta có thể thấy bây giờ nhánh master đã bao gồm commit mới nhất của nhánh develop

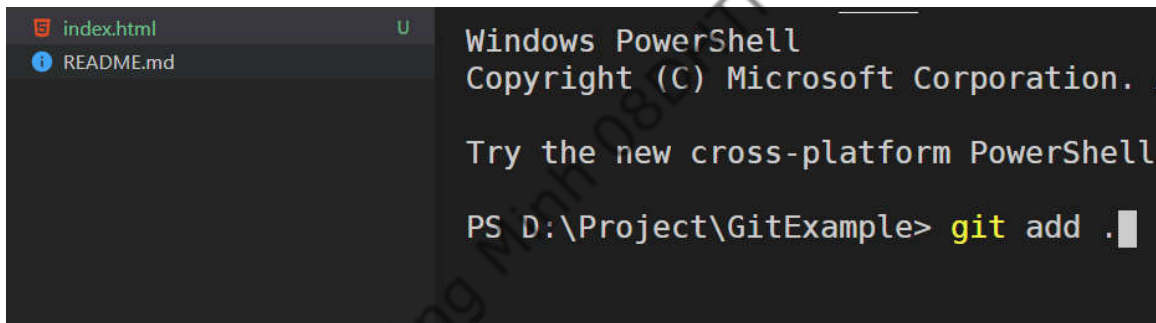
Hoặc có thể làm như sau mà không cần chuyển về nhánh được merge:

git merge <branch-destination-name> <branch-use-to-merge>

Ta sử dụng lại ví dụ trên: **git merge master develop**

- **git add .** : thêm những thay đổi hay chính là đưa các files về trạng thái staged (trạng thái sẵn sàng để commit).

Ví dụ:



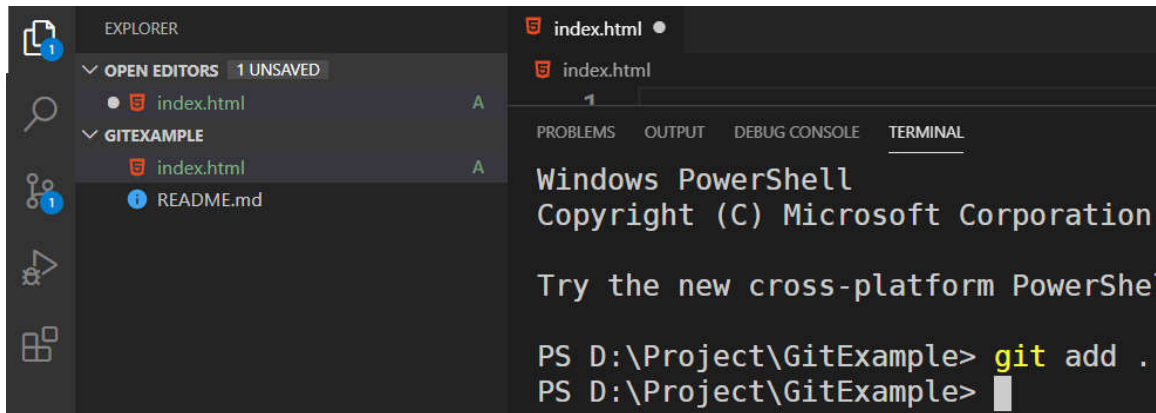
```
Windows PowerShell
Copyright (C) Microsoft Corporation.

Try the new cross-platform PowerShell

PS D:\Project\GitExample> git add .
```

Ở đây mình vừa thêm 1 file nên các bạn có thể thấy chữ U màu xanh bên phải của file đó là trạng thái **untracked**.

Bây giờ thực hiện lệnh **git add .**

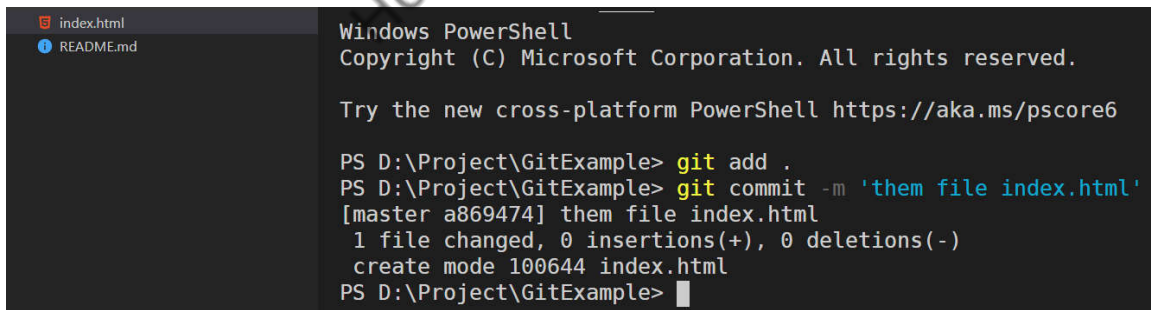


Bây giờ nó là chữ A tức là (Index Added) tức là đã được thêm vào lớp **index** (Không phải file index.html đâu nhé) tức là đã chuyển sang trạng thái **tracked** và sẵn sàng để chuẩn bị commit.

- **git commit -m 'Nội dung commit'** : lệnh này để commit các thay đổi của code vào local repository.

Lưu ý: Những file ở trong index tức là đã ở trạng thái tracked tức là trạng thái sẵn sàng để commit.

Ví dụ:



Mọi người có thể chữ A đã biến mất tức là các file trong lớp **index** đã được đồng bộ với file trong local repo và sẵn sàng để thực hiện lệnh push.

- **git push origin master** : lệnh này đưa các thay đổi lên remote repo của chúng ta. Và từ remote repo chúng ta thể có thể share code.

Ví dụ:

```
PS D:\Project\GitExample> git add .
PS D:\Project\GitExample> git commit -m 'them file index.html'
[master a869474] them file index.html
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html
PS D:\Project\GitExample> git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 280 bytes | 280.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hongminhle0399/GitExample.git
3e9c4fa..a869474 master -> master
PS D:\Project\GitExample>
```

Vậy là code của nhánh repo(branch mà mình đang đứng) đã được đồng bộ với nhánh của remote branch trên remote repo.

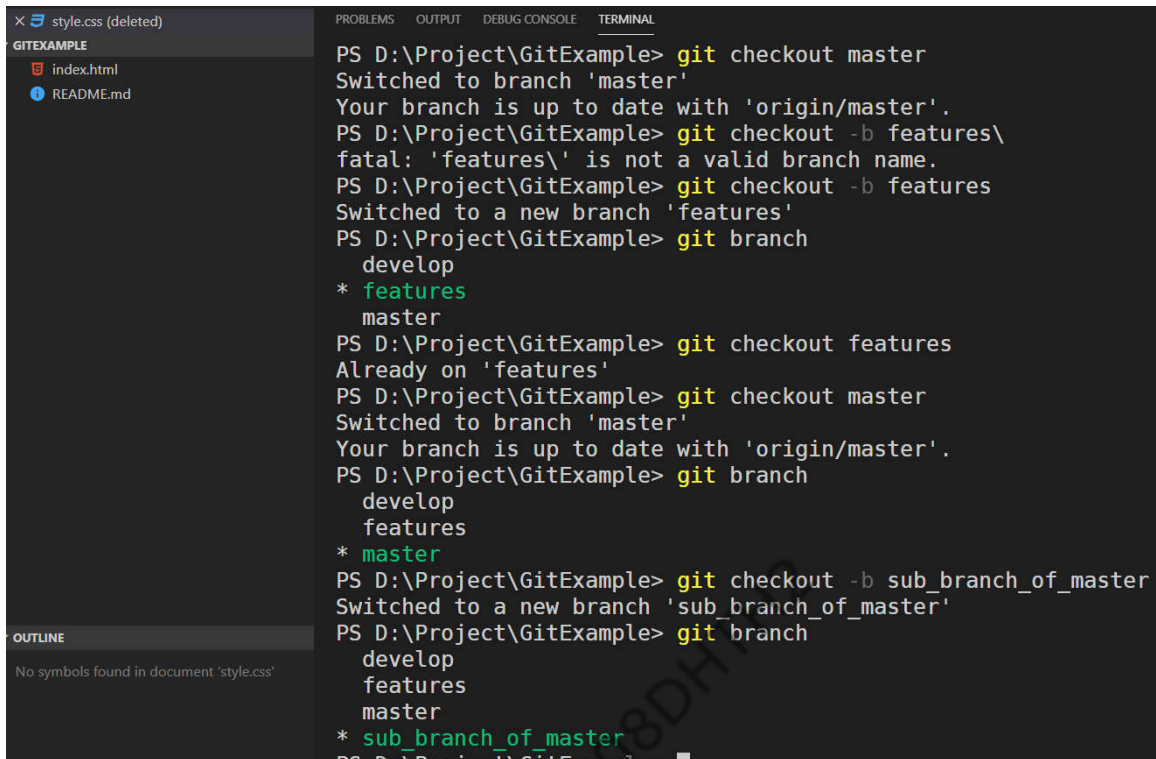
- **git branch** : lệnh này dùng để kiểm tra xem trong local repo của chúng ta có những nhánh nào và hiện tại chúng ta đang đứng ở nhánh nào.

```
PS D:\Project\GitExample> git branch
* master
PS D:\Project\GitExample>
```

- **git branch <branch-name>** : lệnh này dùng để tạo ra một branch mới trong local repo. Branch vừa mới tạo sẽ có cùng history commit với branch mà chúng ta chạy lệnh git này.

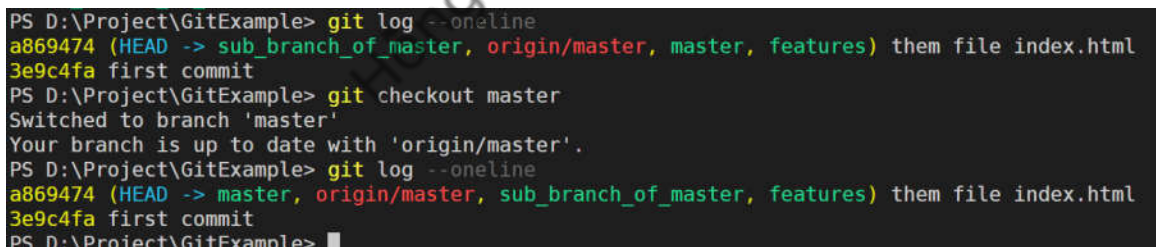
Chứng minh việc branch mới được tạo ra sẽ có cùng commit history

với branch cha nhé:



```
PS D:\Project\GitExample> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
PS D:\Project\GitExample> git checkout -b features
fatal: 'features\' is not a valid branch name.
PS D:\Project\GitExample> git checkout -b features
Switched to a new branch 'features'
PS D:\Project\GitExample> git branch
  develop
* features
  master
PS D:\Project\GitExample> git checkout features
Already on 'features'
PS D:\Project\GitExample> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
PS D:\Project\GitExample> git branch
  develop
  features
* master
PS D:\Project\GitExample> git checkout -b sub_branch_of_master
Switched to a new branch 'sub_branch_of_master'
PS D:\Project\GitExample> git branch
  develop
  features
  master
* sub_branch_of_master
PS D:\Project\GitExample>
```

Giải thích: Mình chuyển về branch master tạo branch **sub_branch_of_master** và các file cũng không đổi



```
PS D:\Project\GitExample> git log --oneline
a869474 (HEAD -> sub_branch_of_master, origin/master, master, features) them file index.html
3e9c4fa first commit
PS D:\Project\GitExample> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
PS D:\Project\GitExample> git log --oneline
a869474 (HEAD -> master, origin/master, sub_branch_of_master, features) them file index.html
3e9c4fa first commit
PS D:\Project\GitExample>
```

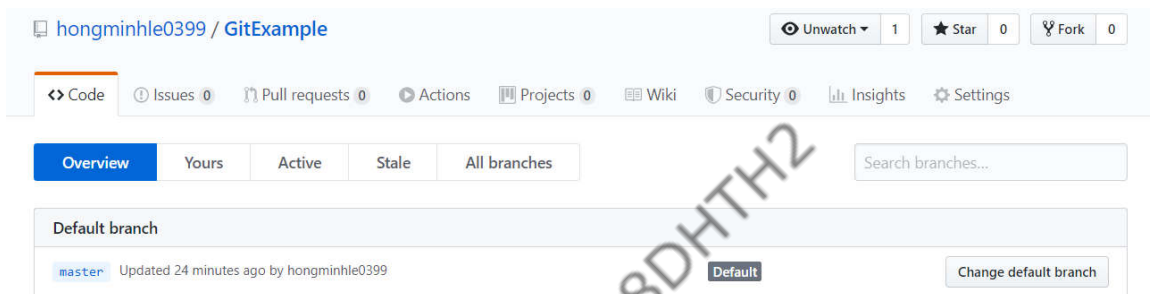
Và rõ ràng nhất là so sánh commit :D. Các bạn có thể thử từ branch develop xem nhé sẽ thấy branch con của nó cũng sẽ thừa hưởng history commit từ nó.

Một vài ví dụ:

```
PS D:\Project\GitExample> git branch develop
PS D:\Project\GitExample> git branch
develop
* master
```

Như các bạn thấy thì branch develop đã được tạo ra và chúng ta dùng lệnh git branch để kiểm tra.

Nhưng lúc này branch của chúng ta chỉ mới ở local repo thôi trên remote vẫn chưa có branch này.

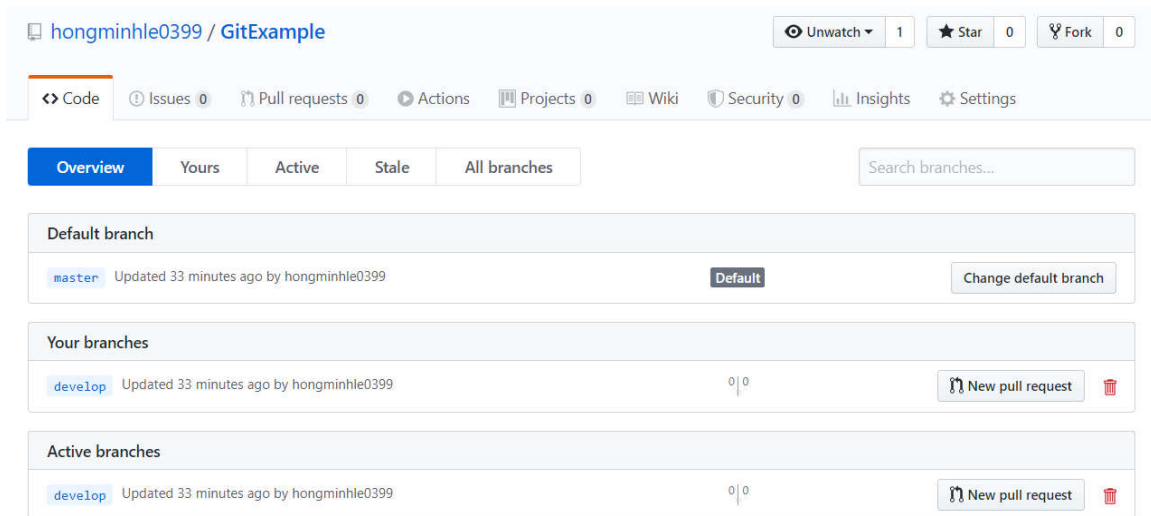


Ta đưa branch lên bằng lệnh **git push origin <branch-name>**. Việc này không bắt buộc vì thường người ta sẽ tạo branch. Chuyển branch rồi làm việc trên đó rồi mới update nhánh mới này lên remote repo.

```
PS D:\Project\GitExample> git push origin develop
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/hongminhle0399/GitExample/pull/new/develop
remote:
To https://github.com/hongminhle0399/GitExample.git
 * [new branch]      develop -> develop
```

Nói thêm: Nếu bạn tạo branch mới và chưa làm gì trên branch đó thì hiển nhiên bạn có thể push để cập nhật branch đó trên remote repo từ branch bạn vừa chạy lệnh **git branch develop** vì nó tương đương **commit history**.

Kiểm tra lại:



- **git checkout <branch-name>** : chuyển qua lại giữa các nhánh đồng thời đưa code ở commit mới nhất của nhánh được chuyển vào workplace.

Ví dụ:

```
PS D:\Project\GitExample> git checkout develop
Switched to branch 'develop'
PS D:\Project\GitExample> git branch
* develop
  master
```

Chứng minh xiu về định nghĩa nha:

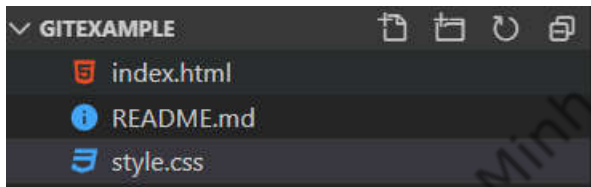
Ta thực hiện thêm file style.css rồi thực hiện các lệnh như trong ảnh.


```

PS D:\Project\GitExample> git add .
PS D:\Project\GitExample> git commit -m 'add a file: style.css'
[develop d7f7856] add a file: style.css
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 style.css
PS D:\Project\GitExample> git push origin develop
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 287 bytes | 287.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To https://github.com/hongminhle0399/GitExample.git
a869474..d7f7856 develop -> develop
PS D:\Project\GitExample> git branch
* develop
  master
PS D:\Project\GitExample>

```

Bây giờ các file trong nhánh develop là:



Khi thực hiện lệnh chuyển branch về master:

```
ITEXAMPLE
index.html
README.md

develop
* master
PS D:\Project\GitExample> git checkout develop
Switched to branch 'develop'
PS D:\Project\GitExample> git branch
* develop
master
PS D:\Project\GitExample> git add .
PS D:\Project\GitExample> git commit -m 'add a
[develop d7f7856] add a file: style.css
1 file changed, 0 insertions(+), 0 deletions(
create mode 100644 style.css
PS D:\Project\GitExample> git push origin deve
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 287 bytes | 287.0
Total 2 (delta 0), reused 0 (delta 0)
To https://github.com/hongminhle0399/GitExamp
a869474..d7f7856 develop -> develop
PS D:\Project\GitExample> git branch
* develop
master
PS D:\Project\GitExample> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'
PS D:\Project\GitExample>
```

Đã mất file **style.css** như định nghĩa nói khi ta chuyển branch thì **những thay đổi ở commit mới nhất** của branch ta chuyển sẽ được hiển thị trong work place của chúng ta.

- **git checkout -b <branch-name>** : lệnh tạo và chuyển nhánh nhanh.

```

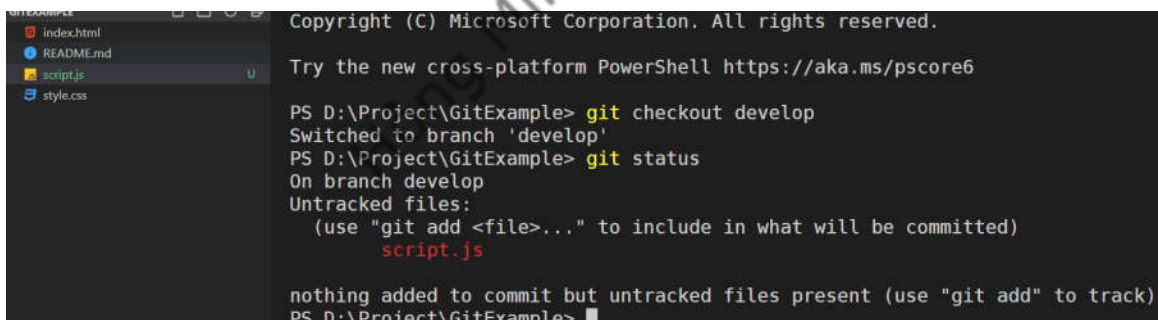
PS D:\Project\GitExample> git checkout -b features
Switched to a new branch 'features'
PS D:\Project\GitExample> git branch
  develop
* features
  master
PS D:\Project\GitExample>

```

Ở ví dụ trên ta đang ở branch master sau khi thực hiện lệnh thì ngay lập tức chuyển sang branch features mà không cần thực hiện hai lệnh đơn tạo branch và chuyển branch như trên kia.

- **git status** : lệnh kiểm tra xem file nào đang được chỉnh sửa và đang được thêm vào hay chính là sự khác nhau giữa 3 lớp của local repo. Ở đây người ta xét các lớp kế nhau. Tóm lại, Nó xét xem local repo đã đồng bộ hoàn toàn chưa.

Ví dụ:



```

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

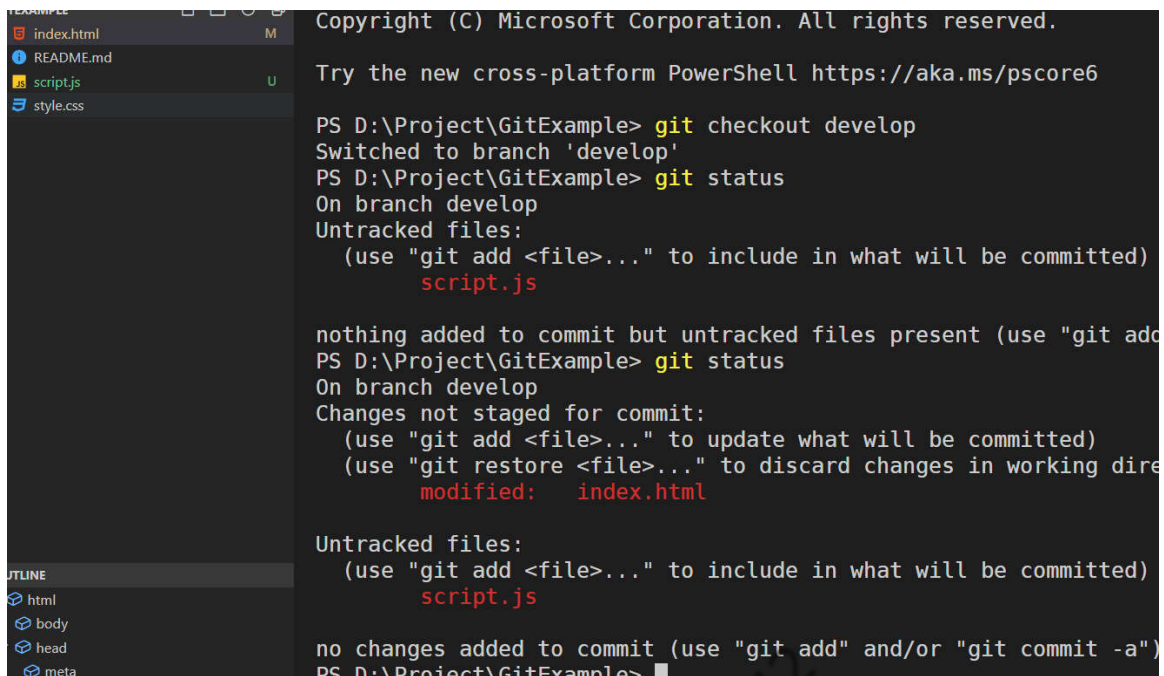
PS D:\Project\GitExample> git checkout develop
Switched to branch 'develop'
PS D:\Project\GitExample> git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        script.js

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Project\GitExample>

```

Ở đây chúng ta thêm file **script.js** và thực hiện lệnh git status thì ra được kết quả như trên.

Tiếp theo t thử thực hiện thay đổi nội dung của file **index.html**



```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Project\GitExample> git checkout develop
Switched to branch 'develop'
PS D:\Project\GitExample> git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        script.js

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Project\GitExample> git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

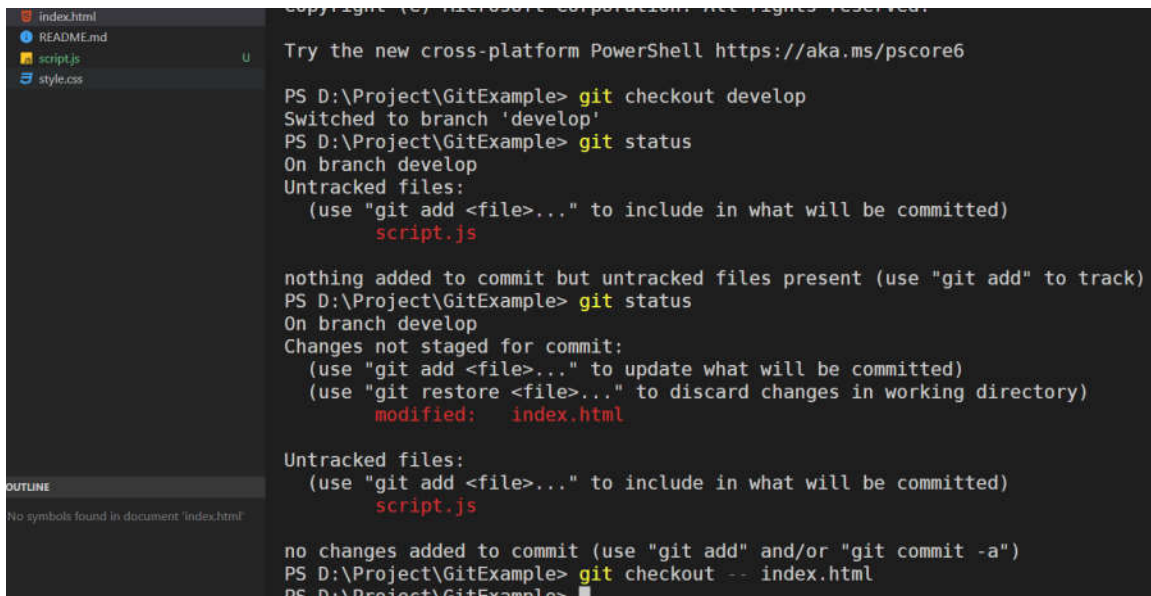
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        script.js

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\Project\GitExample>
```

Và đó chính là chức năng của lệnh git status. Để không còn nhận được các thông báo này các bạn thực hiện lệnh git add . ở đầu nha.

- **git checkout -- <root/file>** : lệnh này dùng để khôi phục file từ trạng thái modified về trạng thái unmodified (tức là trạng thái file đã được commit). Khá hữu dụng khi muốn phục hồi trạng thái file.

Ví dụ:



```
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Project\GitExample> git checkout develop
Switched to branch 'develop'
PS D:\Project\GitExample> git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        script.js

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Project\GitExample> git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        script.js

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\Project\GitExample> git checkout -- index.html
PS D:\Project\GitExample>
```

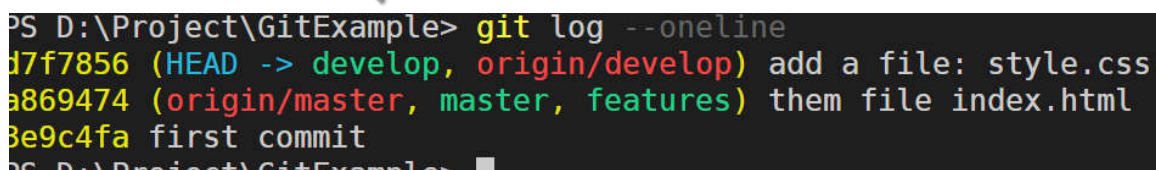
Các bạn có thể thấy file index.html đã trở lại trạng thái unmodified.

- **git checkout -- .** : giống lệnh kia nhưng đưa tất cả các file ở trạng thái modified về trạng thái unmodified.

- **git log --oneline** : lệnh này dùng để xem commit của branch hiện tại của bạn.

Ví dụ:

Đây chính là commit của nhánh develop

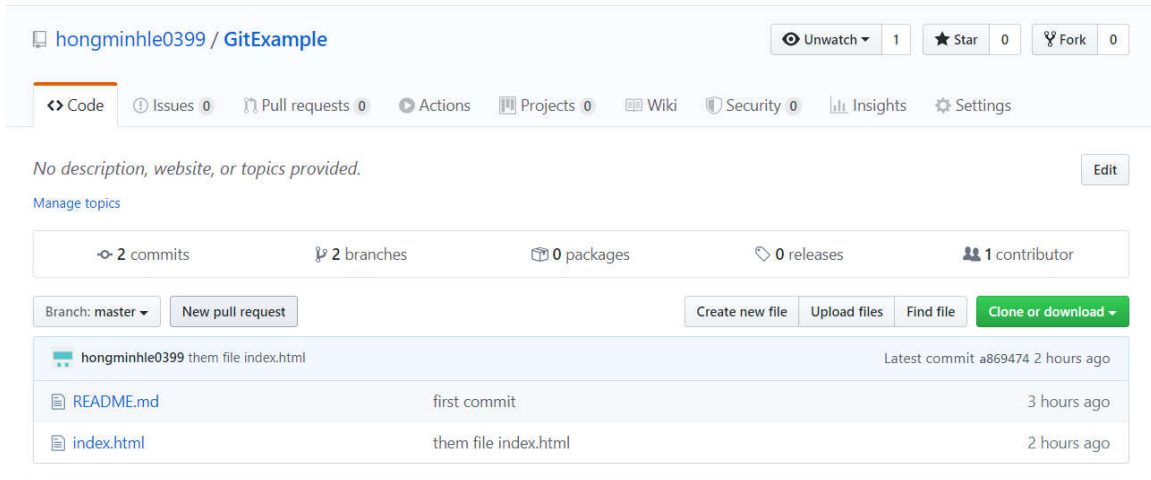


```
PS D:\Project\GitExample> git log --oneline
d7f7856 (HEAD -> develop, origin/develop) add a file: style.css
a869474 (origin/master, master, features) them file index.html
3e9c4fa first commit
PS D:\Project\GitExample>
```

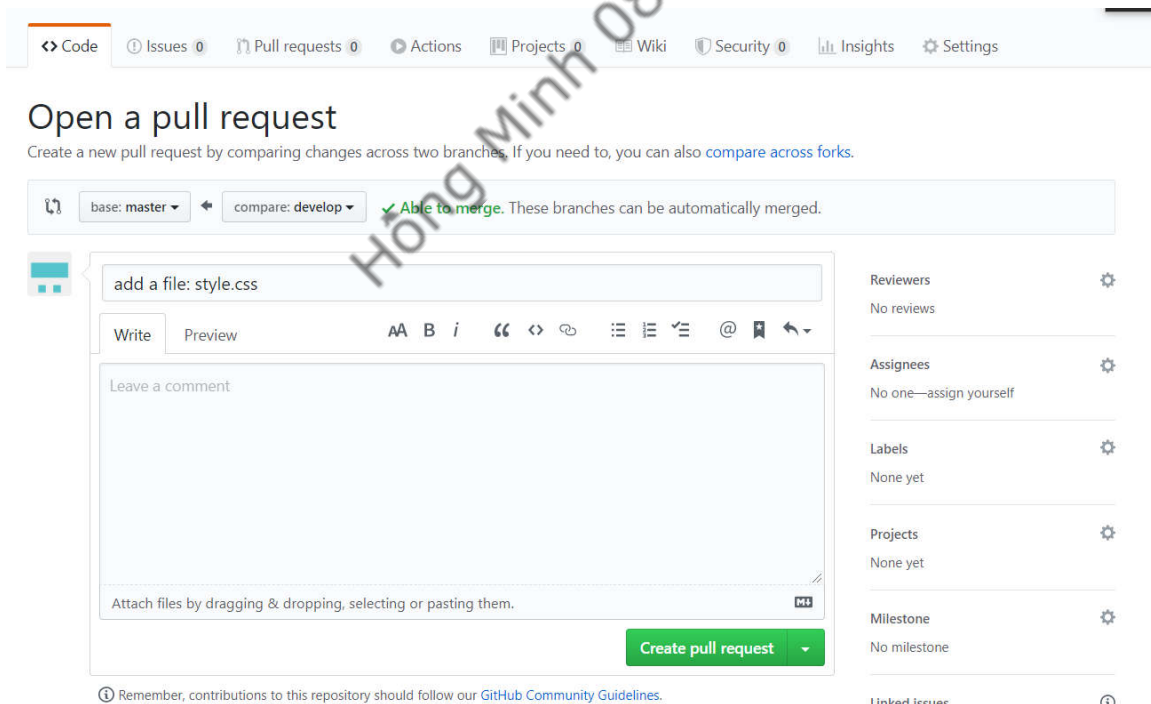
5. Tạo pull request:

Đây là khi chúng ta thấy code chúng ta đã chạy ổn và muốn đưa nó vào nhánh chính của team thì chúng ta sẽ tạo nó (Nó rất cần thiết nhé). Mỗi git hosting sẽ có cách tạo khác nhau ở đây mình sẽ hướng dẫn bên Github.

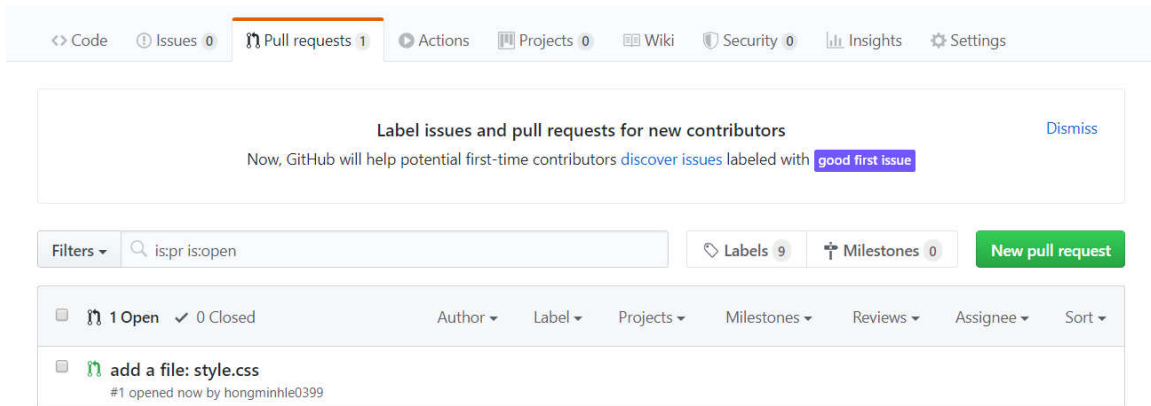
Ta chọn **New pull request**.



Ta thấy như ảnh và điền thông tin cần thiết như tên của pull request và nội dung cũng như tệp đính kèm ... Sau đó chọn create nhé. Lúc này người quản lý remote repo sẽ thực hiện xem xét và merge nó vào nhánh chính nhé. :D



Đây là ảnh bạn quản lý dự án sẽ thấy



Tada. Series hướng dẫn git của mình đã hoàn thành. Và còn rất nhiều lệnh git luôn các bạn có thể tham khảo thêm git-scm, bitbucket, github, git-towel,... Chào các bạn :D

Soạn bởi **LÊ HỒNG MINH CUTE 08DHTH2 :))))**