

# GitHub Tutorial



*Author : cristazn*

*Reference : stackoverflow, github.com,...*

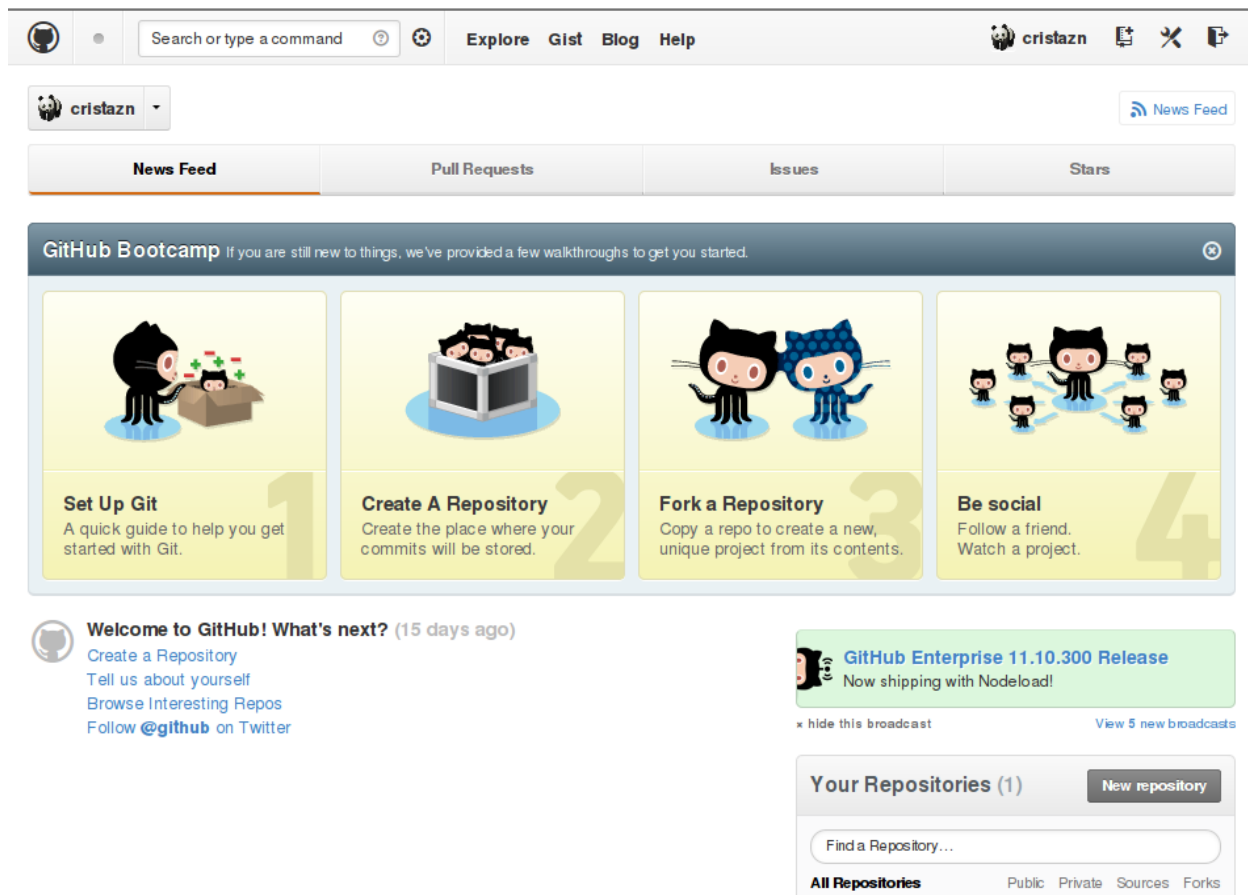
*Create at : 2/2/2013*

## A. GIỚI THIỆU

### 1. Git - GitHub là gì ?

Git là một sản phẩm mã nguồn mở được thiết kế bởi Linus Torvalds dùng để phát triển nhân Kernel của hệ điều hành Linux. Mục đích chính của Git là cung cấp một hệ thống quản lý phiên bản trong quá trình phát triển phần mềm. Git ban đầu được dùng trong môi trường dòng lệnh.

GitHub có hạt nhân là Git, GitHub là một giao diện quản lý trên nền web. Ngoài các chức năng về dòng lệnh tương tự Git thì GitHub còn hỗ trợ thêm các chức năng phục vụ cho việc quản lý dự án như : tạo task , wiki, fork, social code ..



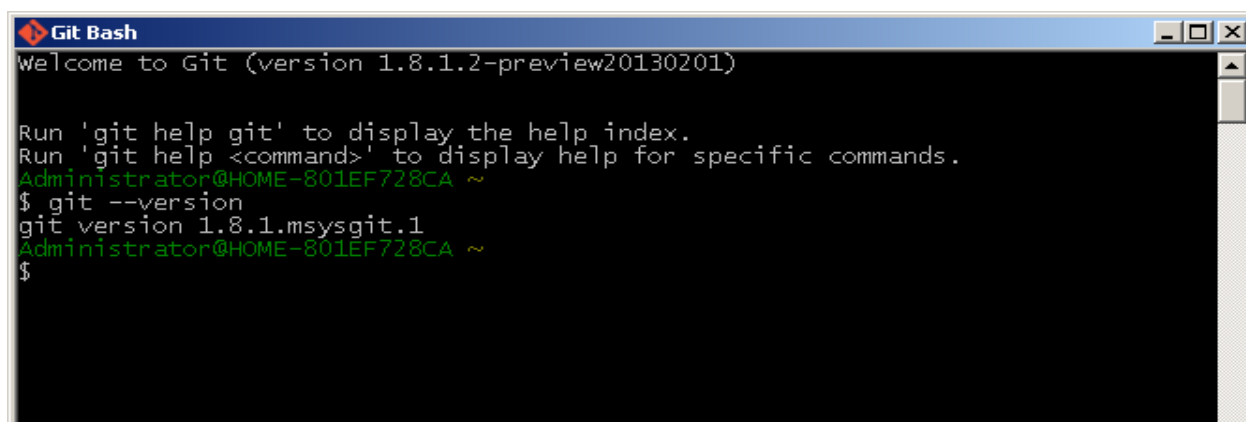
Hình 1 : Giao diện của GitHub

### 2. Công cụ GitHub:

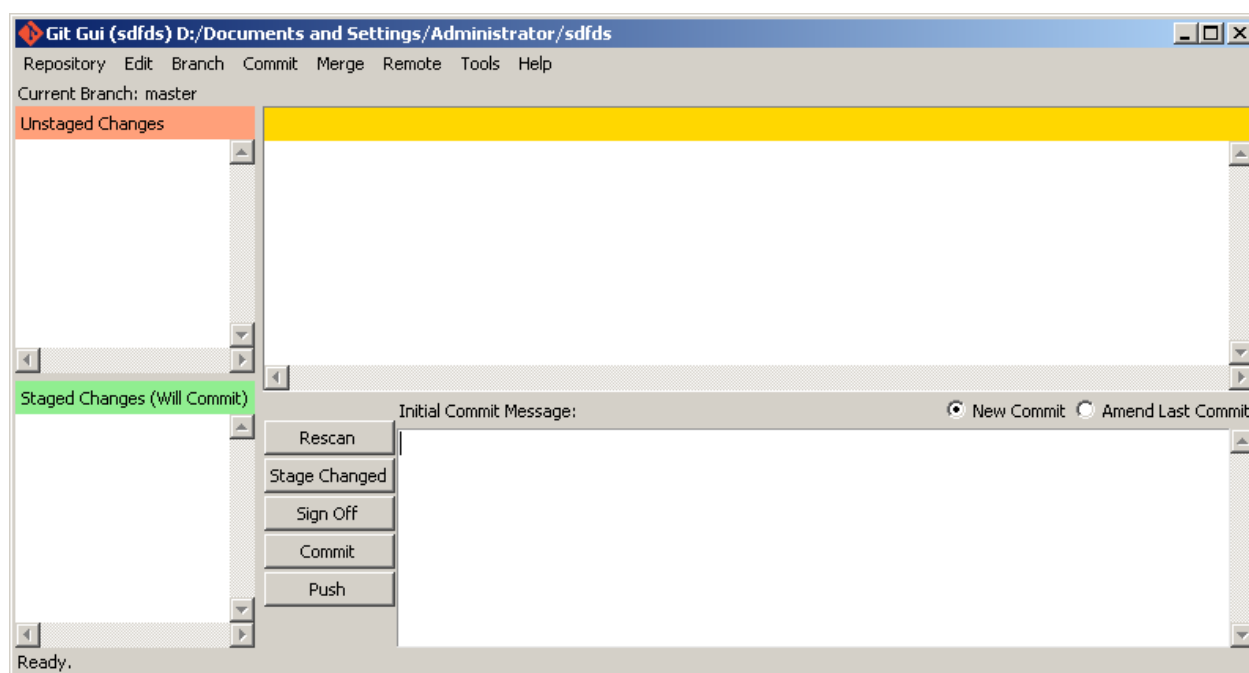
GitHub sử dụng giao diện web để cung cấp cho việc khởi tạo và quản lý code. Ngoài ra để sử dụng ta có thể sử dụng qua command tool, Git Client và các IDE hỗ trợ.

Đầu tiên download gói cài đặt Git  
<http://code.google.com/p/msysgit/downloads/list?can=3>

Sau khi cài đặt bạn sẽ có Git bash và Git gui (basic).



Hình 2 : Git bash



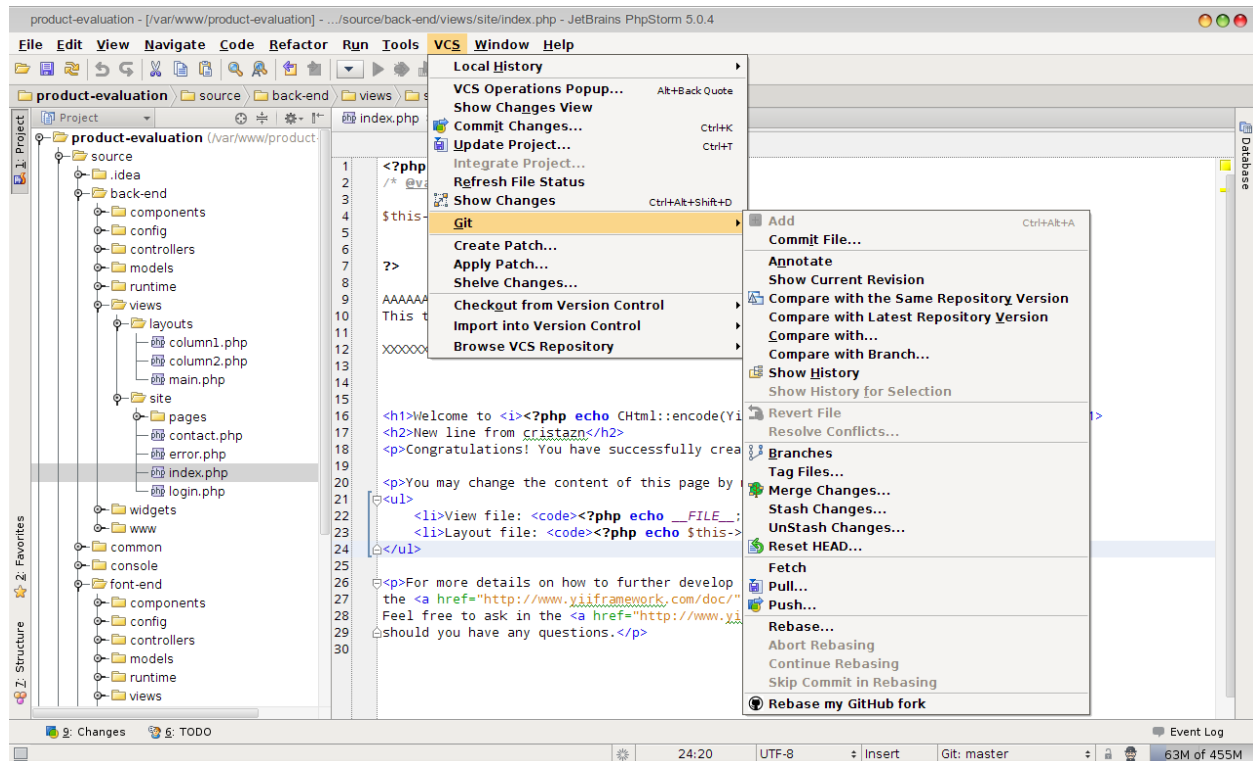
Hình 3 : Git gui

Ngoài ra còn có thể sử dụng GUI của hãng thứ 3 hỗ trợ các công cụ chuyên nghiệp hơn.



Hình 4 : SmartGit/Hg

Bạn cũng có thể dùng GUI của các IDE hỗ trợ.



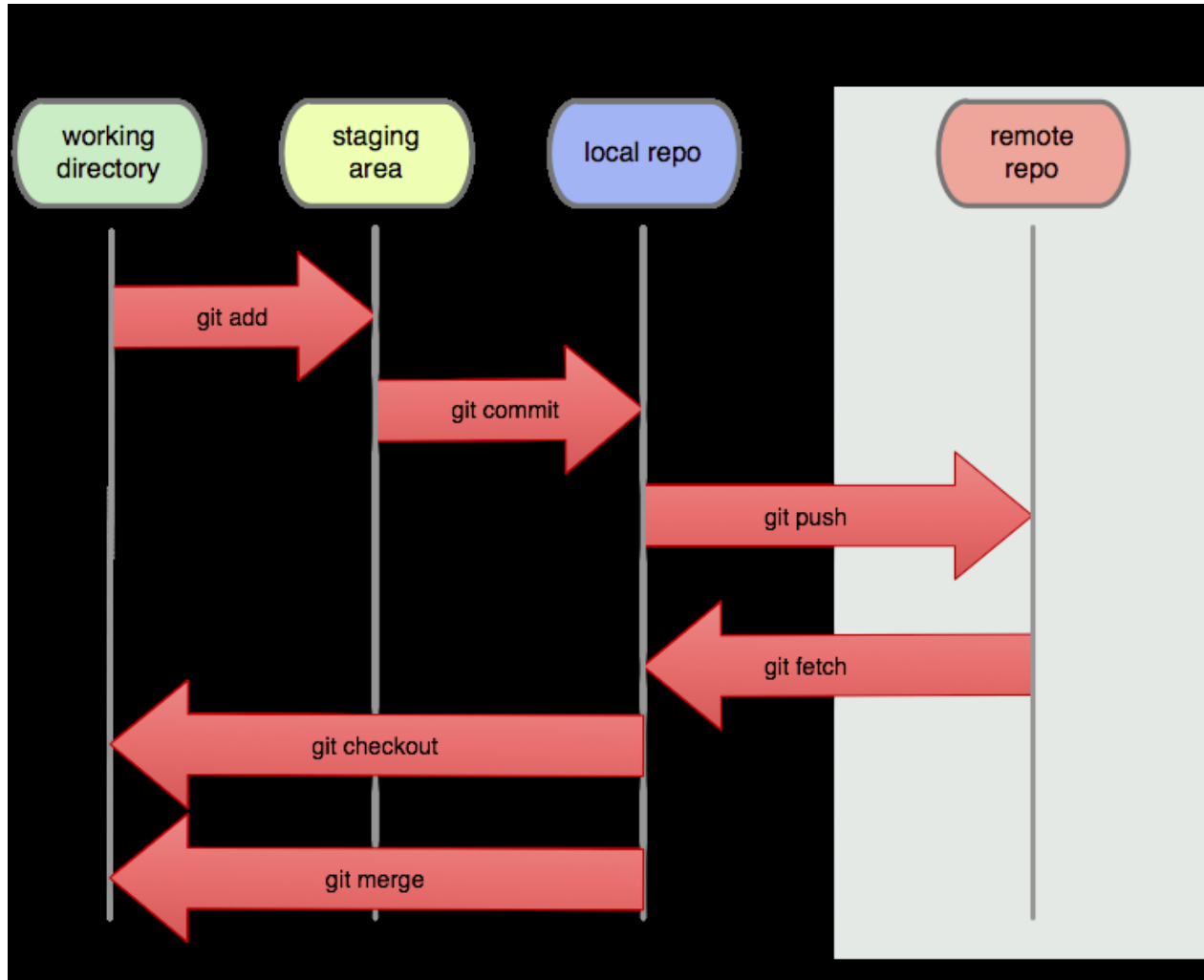
Hình 5 : Git trong PhpStorm 5

\* Ngoài ra ta có thể cài đặt Git trong Terminal của các hệ điều hành hệ \*nix.  
VD : Debian style : apt-get install git

## B. CÁCH SỬ DỤNG GIT – GITHUB

### 1. Các khái niệm cơ bản

Git là mô hình dạng Local – Remote. Trong đó khi ta cấu hình xong thì ta làm việc ở dạng local có nghĩa là ta thao tác trên chính máy chúng ta. Khi nào hoàn thành thì ta sẽ push lên trên Remote (chứa Respository).

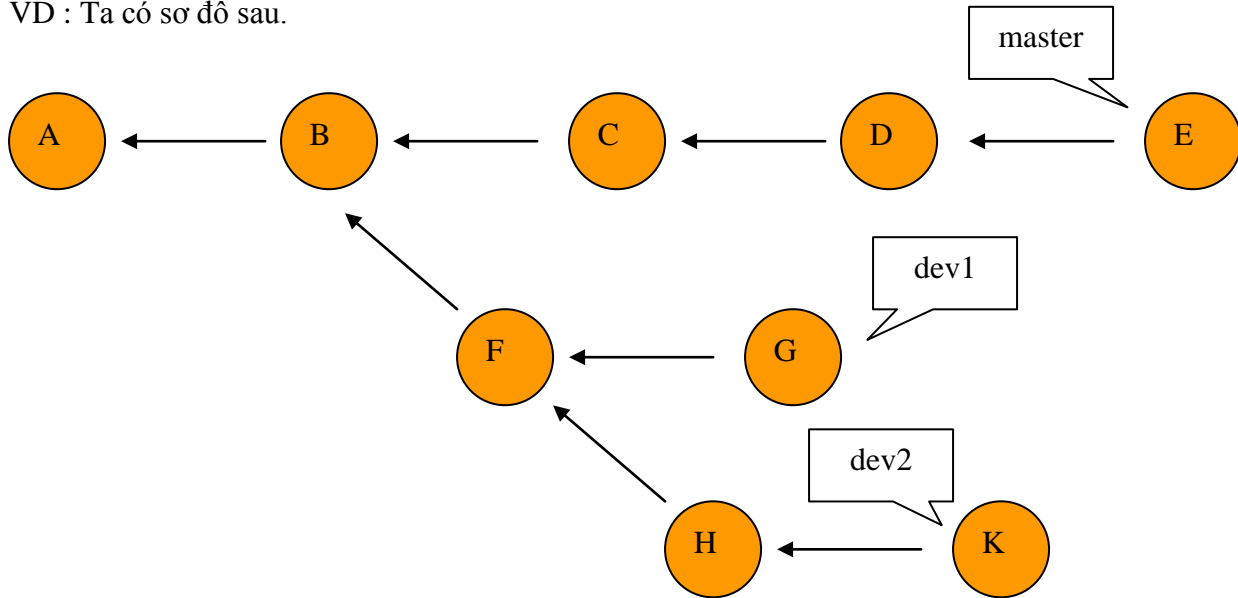


Hình 6 : Model git

Các hoạt động chính là add->commit->push (out way).  
fetch->checkout->merge (in way).

Branch (nhánh) là khái niệm quan trọng trong Git. Trong git sự vận hành, phát triển của project được trừu tượng thành từng nhánh. Khi mới khởi tạo ta có 1 nhánh chính “master” trong lúc phát triển project sẽ có những phát triển ra thành từng nhánh phụ. Từ giờ ta dùng “branch” thay cho nghĩa tiếng việt là “nhánh” để hợp chuẩn quốc tế.

VD : Ta có sơ đồ sau.



Hình 7 : Git Status

Master branch có các trạng thái : A<-B<-C<-D<-E.

Dev1 dùng để phát triển tính năng mới từ phiên bản B của master branch của project.

Dev1 branch : A<-B<-F<-G

Dev2 dùng để phát triển tính năng mới từ phiên bản F của dev1 branch.

Dev1 branch : A<-B<-F<-H<-K

Các trạng thái của các branch được riêng biệt với nhau. VD khi ở trạng thái B ở master khi ta tách sang dev1 thì khi ta sửa đổi project thì những thay đổi đó chỉ nằm ở dev1. Chỉ khi nào tính năng ở branch dev1 hoàn chỉnh ta có thể yêu cầu Pull Request để cập nhật vào bản master. Vì vậy lúc nào trong project cũng có 1 phiên bản ổn định.

Mỗi trạng thái được phân biệt với nhau qua SHA1 có dạng như sau .

00bfd8c0d8c9c905b4dcb1e4f698d78359590c54

647254f1f7c1df2103a092332503d351dcf68a05

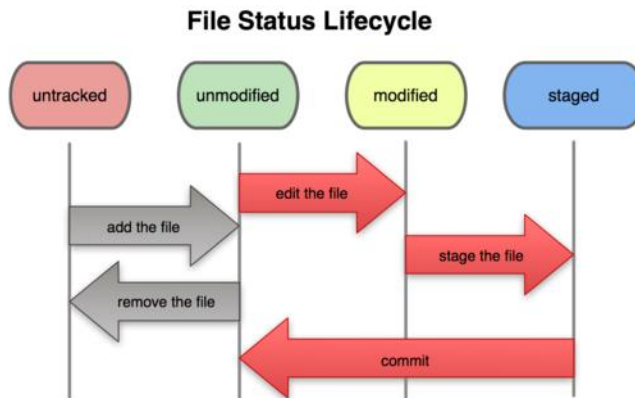
....

Từ trạng thái nào đó trong branch ta có thể lùi về trạng thái trước đó một cách rất dễ dàng bằng cách đưa chuỗi SHA1 vào command để trở về trạng thái được chỉ định.

Một trạng thái được tạo ra khi ta commit trạng thái đó. Ta dùng thuật ngữ staged và unstaged. 1 file đã staged là 1 file đã được vào staging area hoặc có thể nói khác hơn là index file. Ta thực hiện việc này bằng cách sử dụng lệnh **git add <file name>** hoặc **git add .** Dấu "." Ý nghĩa dùng cho tất cả các file. Ta có thể unstaged file bằng cách dùng lệnh : **git reset HEAD <filename>**

HEAD là một con trỏ (pointer) dùng để chỉ vị trí hiện hành của trạng thái mà ta quan tâm.

### Vòng đời của file :

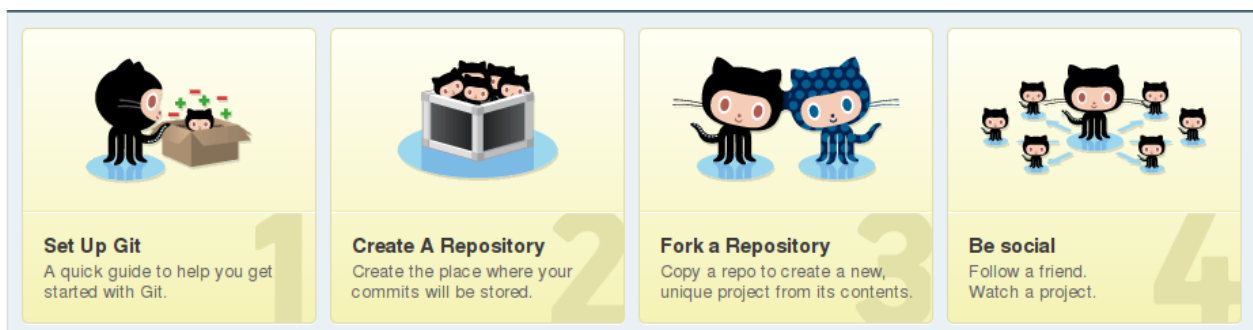


Hình 8 : File life

Các file trong local có 2 trạng thái là tracked và untracked. Khi nói 1 file đã trong trạng thái tracked thì file đó đã được commit (tuy file đó có thể là đã bị sửa đổi (modified) hoặc chưa sửa đổi (unmodified) và staged (file đã commit)). Quá trình đưa file vào trạng thái tracked bằng dòng lệnh nh ***git add <filename>*** đã được trình bày ở trên.

Khi nói 1 file trong trạng thái Untracked thì file đó có thể chưa có trong staging area. Khi 1 project được clone về qua lệnh ***git clone <URL>*** thì các file trong project đó đã staged và unmodified, có nghĩa là các file đã được tracked và chưa có bị thay đổi gì cả. Lợi ích của việc này là để khi ta dùng lệnh ***git status*** ta có thể biết được file nào trong trạng thái Untracked thì ta sẽ đưa nó vào trạng thái Tracked. Lúc này khi ta commit thì file đó mới được cập nhật.

## 2.. Đăng ký sử dụng



Hình 9 : Git Step

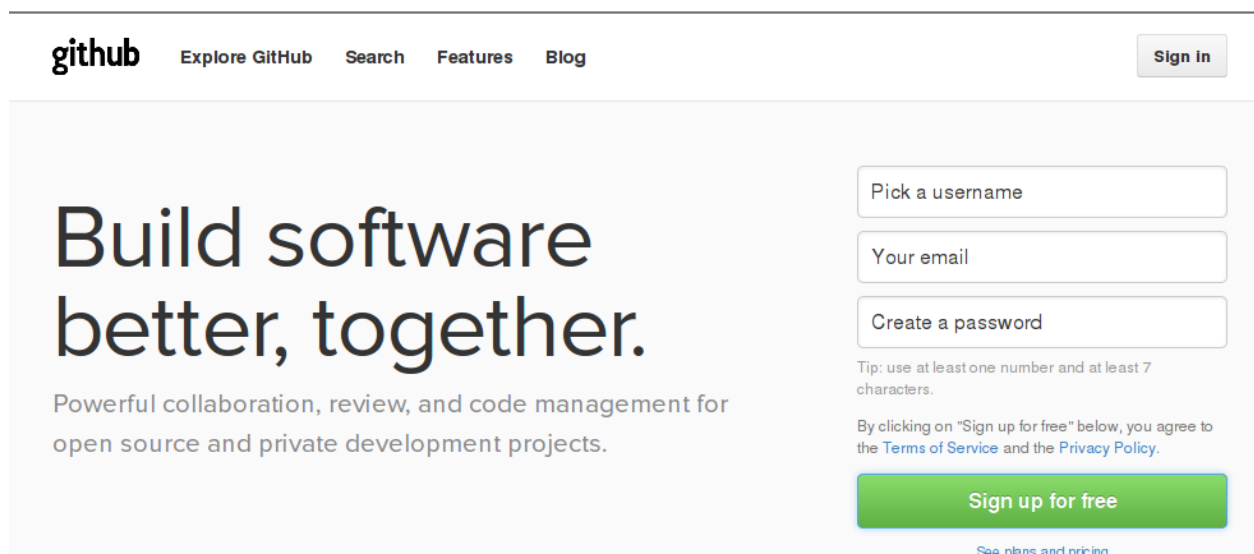
B1 : Đầu tiên đăng ký một tài khoản tại trang chủ GitHub.

B2 : Cấu hình cho tài khoản

B3 : Tạo mới Repository (Kho lưu trữ )

B4 : Fork repository

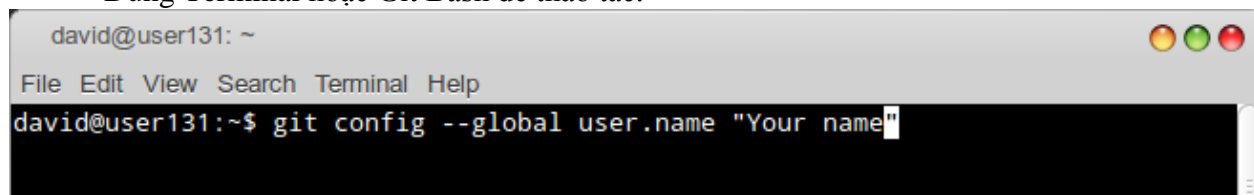
### B1 : Đăng ký



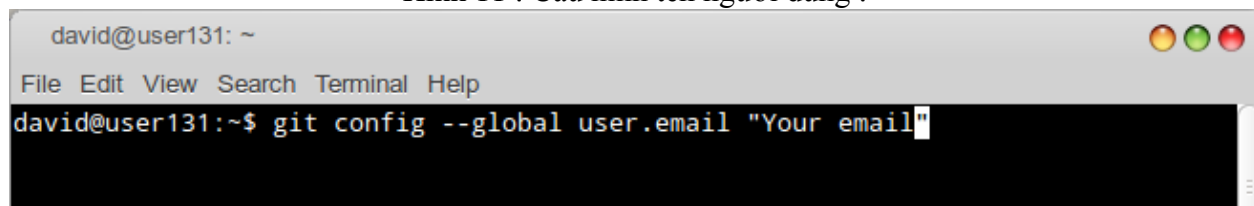
Hình 10 : Giao diện đăng ký trang chủ

## ***B2 : Cấu hình cho tài khoản***

Dùng Terminal hoặc Git Bash để thao tác.

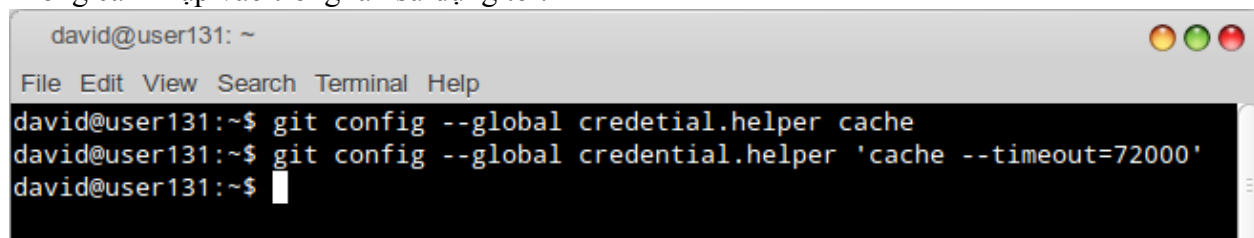


Hình 11 : Cấu hình tên người dùng :



Hình 12 : Cấu hình email

Ngoài ra có thể cấu hình thêm về credential helper cho phép cache username/password để không cần nhập vào trong lần sử dụng tới.



Hình 13: Cấu hình credential

Từ --global ám chỉ là toàn cục có nghĩa là áp dụng cho tất cả các repo trên máy tính đang dùng.



### B3 : Tạo mới Repository (Kho lưu trữ)

Owner: cristazn / Repository name: abc

Great repository names are short and memorable. Need inspiration? How about **finna-be-octo-bear**.

Description (optional): abc project

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately.

Add .gitignore: **None**

**Create repository**

Hình 14 : Tạo mới repository

### B4 : Fork repository

Public cristazn / product-evaluation

[Pull Request](#) [Unwatch](#) [Star](#) [Fork](#)

**Code** Network Pull Requests 1 Issues 2 Wiki Graphs Settings

**Edit** This is a website support users update hi-tech information and choose product. — [Read more](#)

[ZIP](#) [HTTP](#) [SSH](#) [Git Read-Only](#) <https://github.com/cristazn/product-evaluation.git> [Read+Write access](#)

branch: master Files Commits Branches 2 Tags

**product-evaluation** 34 commits

Merge pull request #4 from cristazn/alpha

cristazn authored 35 minutes ago latest commit 30a4984802

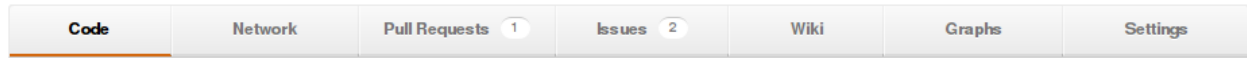
File	Time	Commit
.idea	an hour ago	line from alpha [cristazn]
source	21 hours ago	ddd [cristazn]
.gitignore	2 days ago	Initial commit [cristazn]
Makefile	21 hours ago	dasdsa [cristazn]
Makefile_master	21 hours ago	Added Makefile_test [cristazn]
README.md	2 days ago	Initial commit [cristazn]
TODO	2 days ago	new init [cristazn]
plan	21 hours ago	Add plan file [cristazn]

/cristazn/product-evaluation.git

Hình 15 : Fork repository

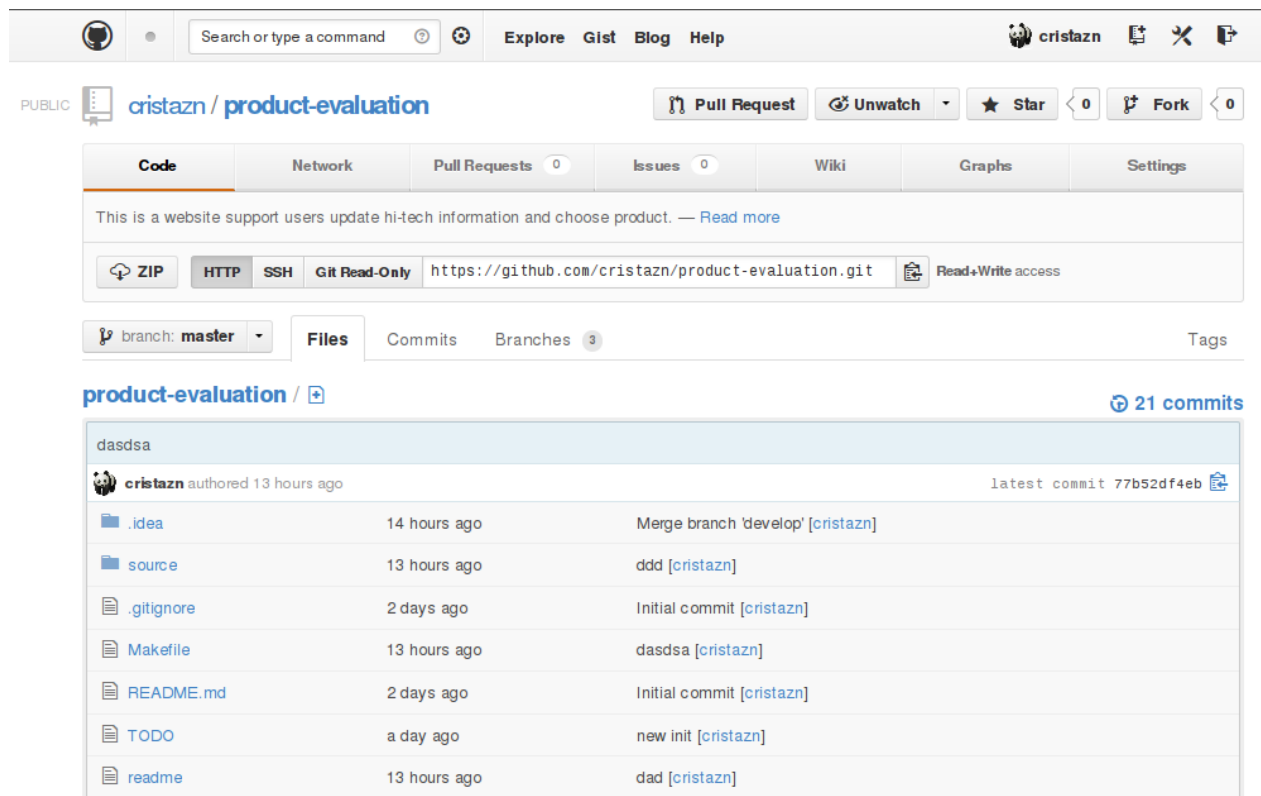
Click chọn vào nút fork. Fork là cho phép nhóm hoặc người lập trình viên khác có thể đóng góp vào project của mình hoặc cho phép người khác có thể nhân bản (clone) project này để sử dụng làm bước khởi đầu cho chính họ.

## 2. GitHub Menu



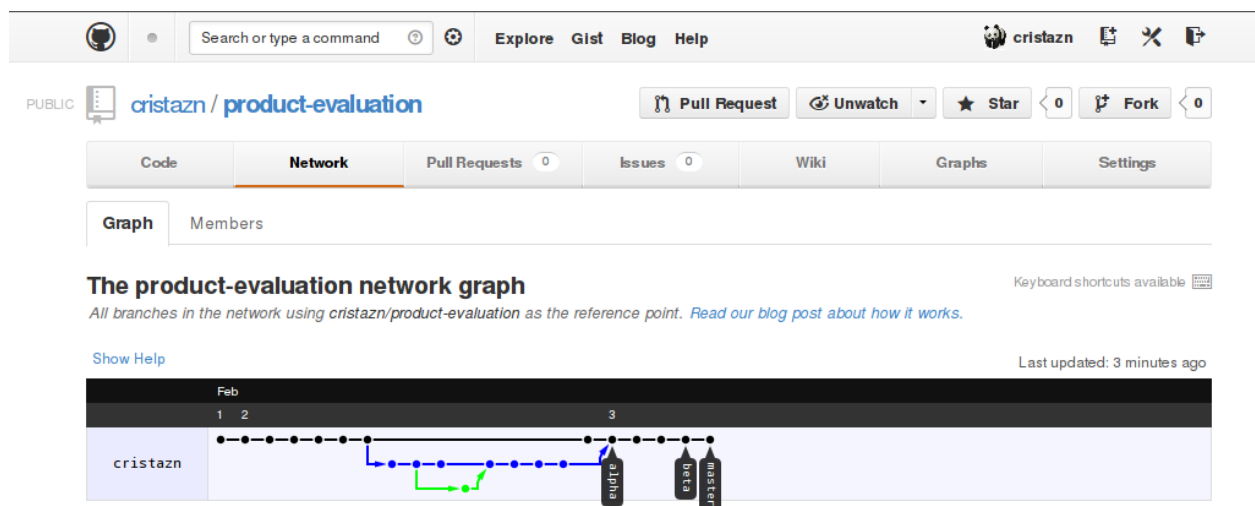
Hình 16 : Menu

- Code Tab : tab hiển thị giao diện thành phần code trong project.



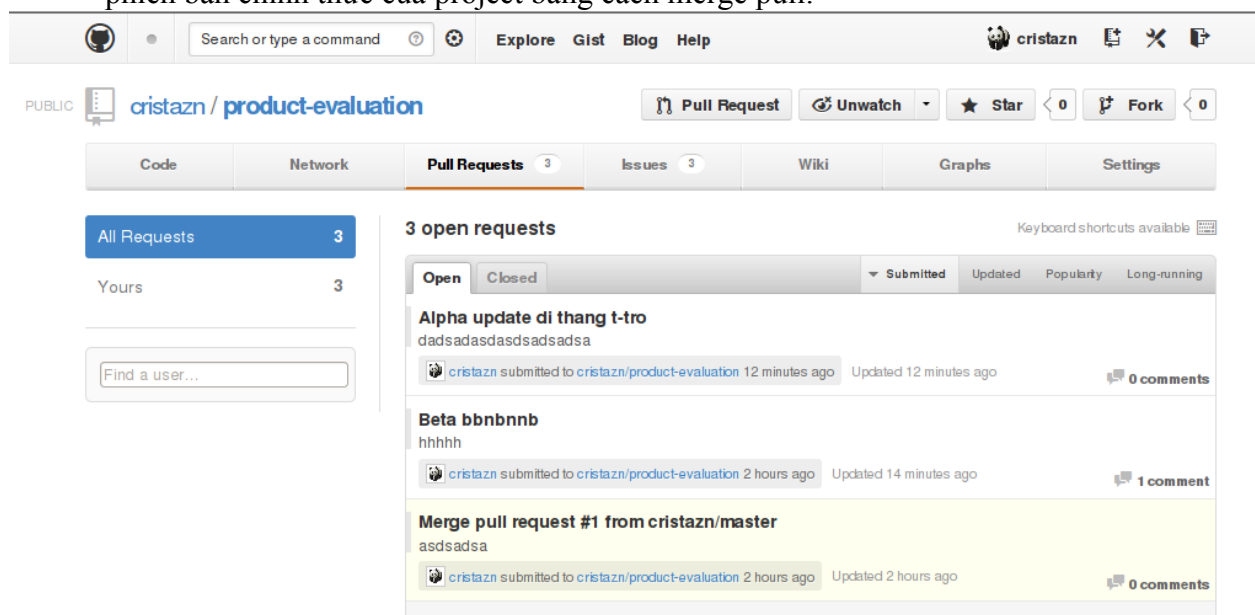
Hình 17 : Giao diện bên trong repository

- Network Tab : tab hiển thị giao diện các nhánh branch và danh sách các thành viên trong project.



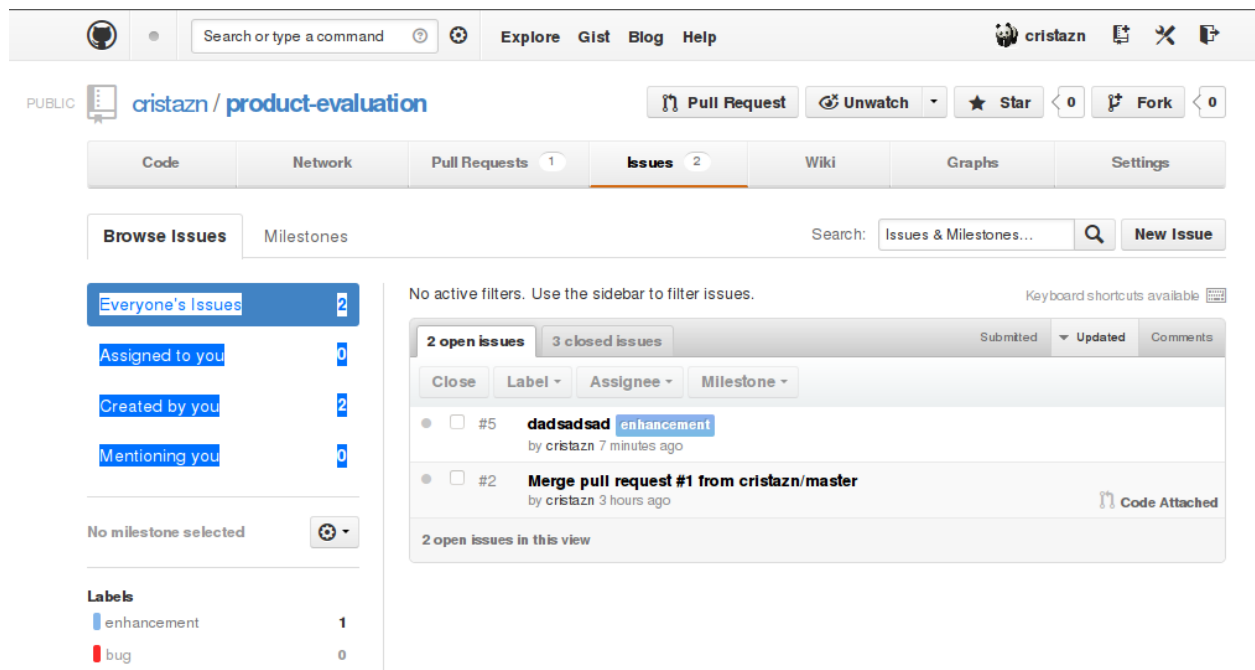
Hình 18 : Network tab

- Pull Request : tab hiển thị và báo hiệu cho người dùng ở các branch khác rằng dữ liệu ở branch người pull request đã được đưa lên do đó yêu cầu họ xem xét và thay đổi trong phiên bản chính. Ngoài ra còn cho phép bình luận các thông tin. Khi 1 bản tin pull được chấp thuận bởi tác giả hay các thành viên trong project thì nó có thể được ứng dụng vào phiên bản chính thức của project bằng cách merge pull.



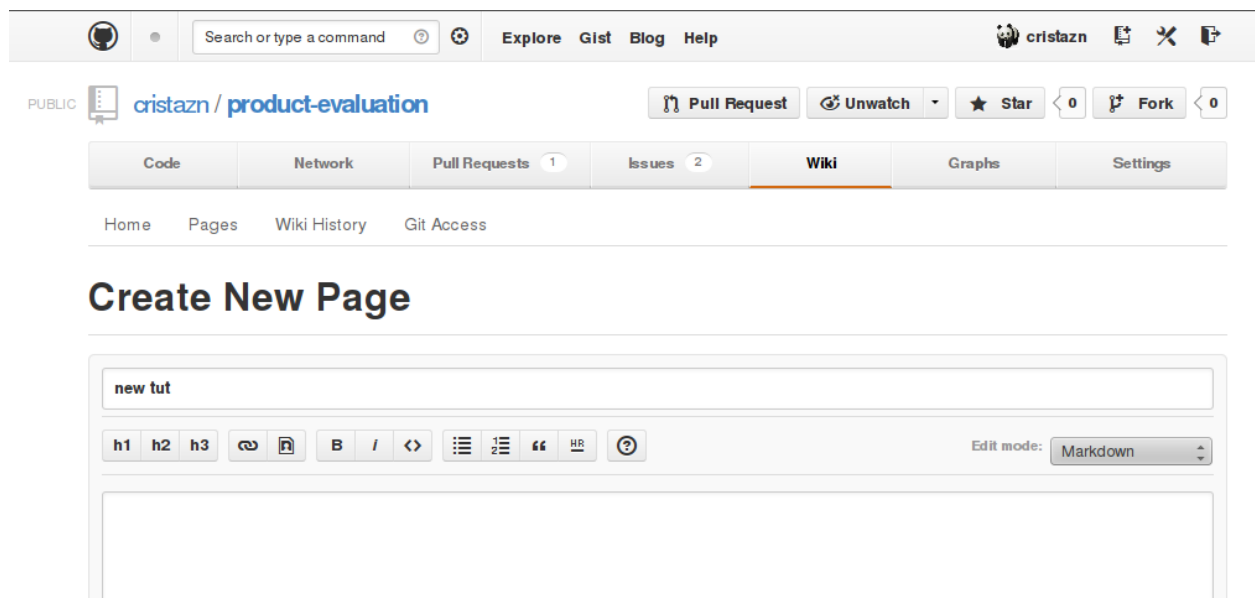
Hình 19 : Pull Request tab

- Issue Tab : tab cho phép người dùng hoặc người quan tâm đến project có thể post các thông tin thông báo về các lỗi (bug), cải tiến (enhanced) đến cộng đồng. Vì vậy nó hỗ trợ cho lập trình viên trong project có thể tìm thấy và sửa lỗi nhanh hơn.



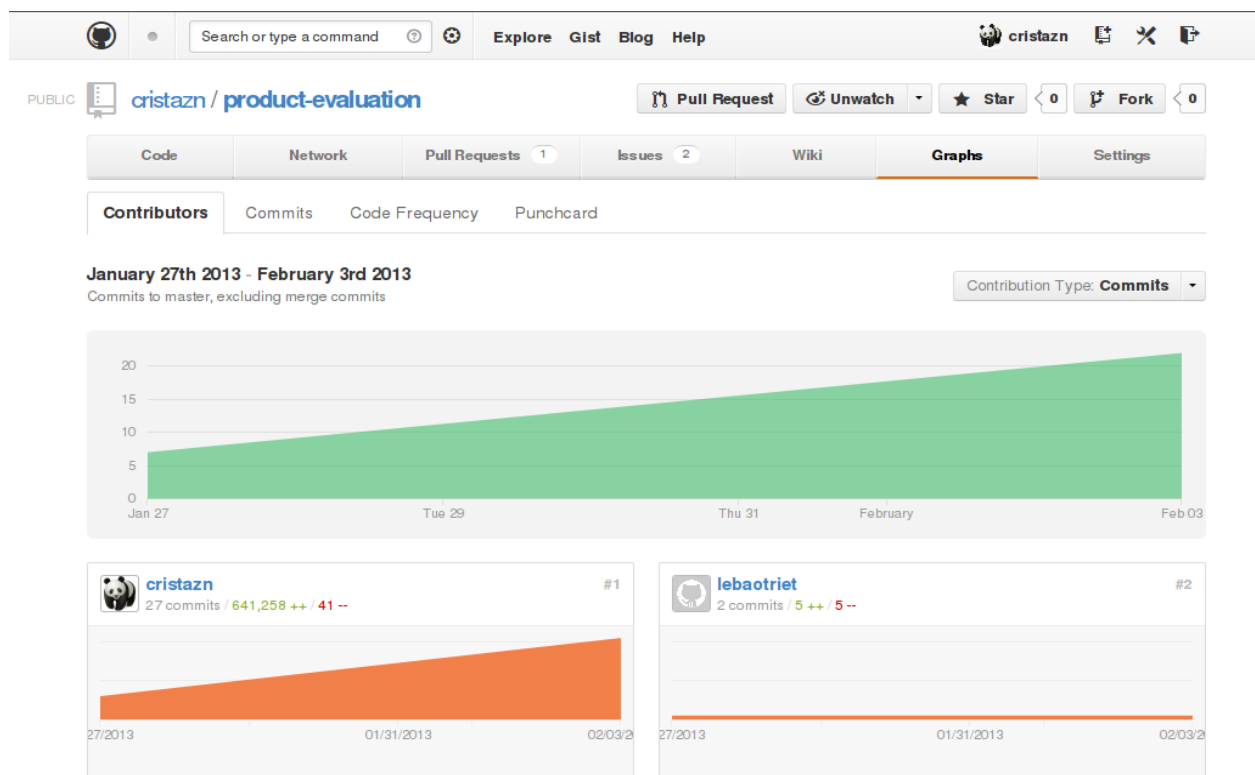
Hình 20 : Issues tab

- Wiki tab : tab cho phép nhóm lập trình có thể tạo ra các hướng dẫn cho người dùng cuối hoặc các các hỗ trợ, api cho các lập trình viên sử dụng project của mình làm điểm xuất phát.



Hình 21 : Wiki tab

- Graph tab : biểu đồ thể hiện mức độ hoạt động, đóng góp của thành viên trong project.



Hình 22 : Graph tab

- Setting tab : tab cho phép điều chỉnh các thành phần của project tên, default branch ...

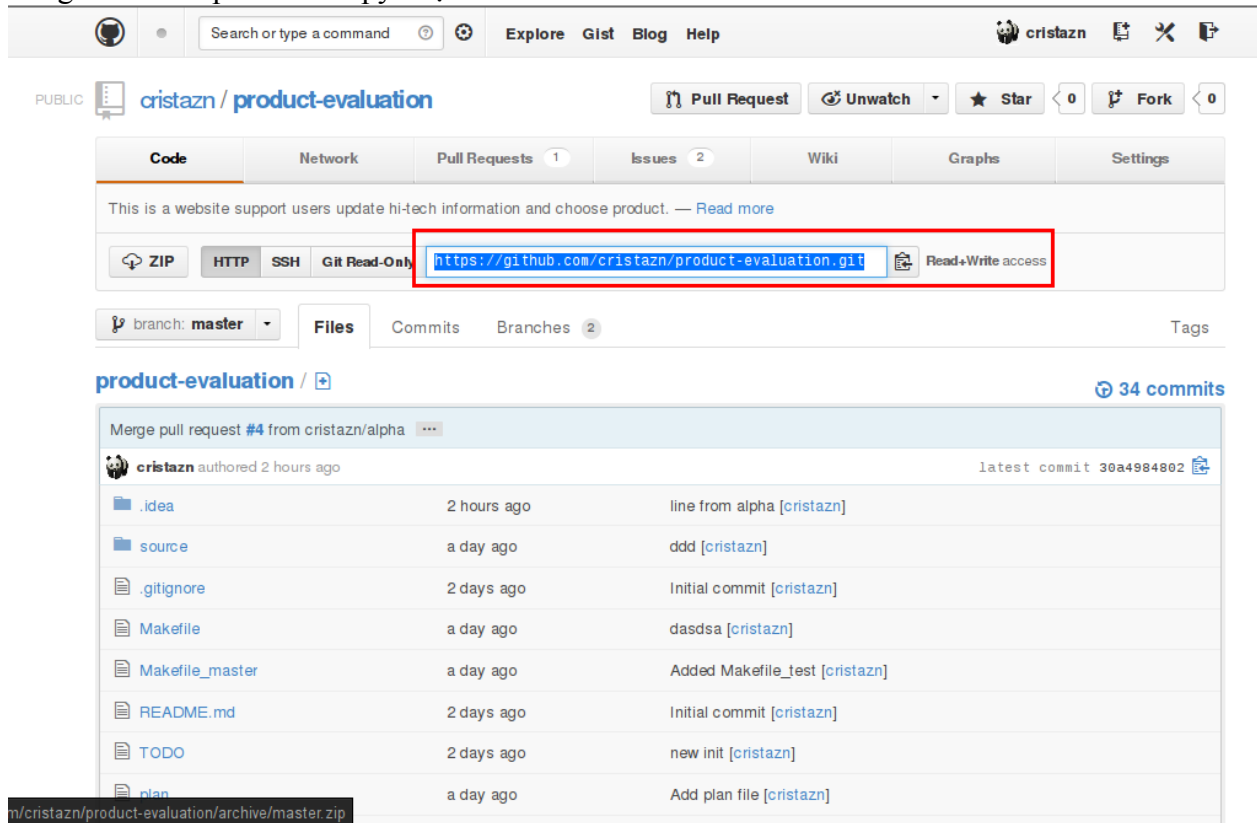
The screenshot displays the GitHub 'Repository Settings' page for the repository `cristazn/product-evaluation`. The page is divided into several sections:

- Options**: A sidebar menu with links to [Collaborators](#), [Service Hooks](#), and [Deploy Keys](#).
- Settings**: The main content area, containing:
  - Repository Name**: A text input field with the value `product-evaluation` and a **Rename** button.
  - Default Branch**: A dropdown menu with the value `master`.
- Features**: A section with checkboxes for enabling or disabling features:
  - ☒ **Wikis**: GitHub Wikis are the simplest way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support or anything you wish.
  - ☐ **Restrict edits to Collaborators only**: Public Wikis will still be readable by everyone.
  - ☒ **Issues**: GitHub Issues adds lightweight issue tracking tightly integrated with your repository. Add issues to milestones, label issues, and close & reference issues from commit messages.

Hình 23 : Setting tab

### 3. Hiện thực

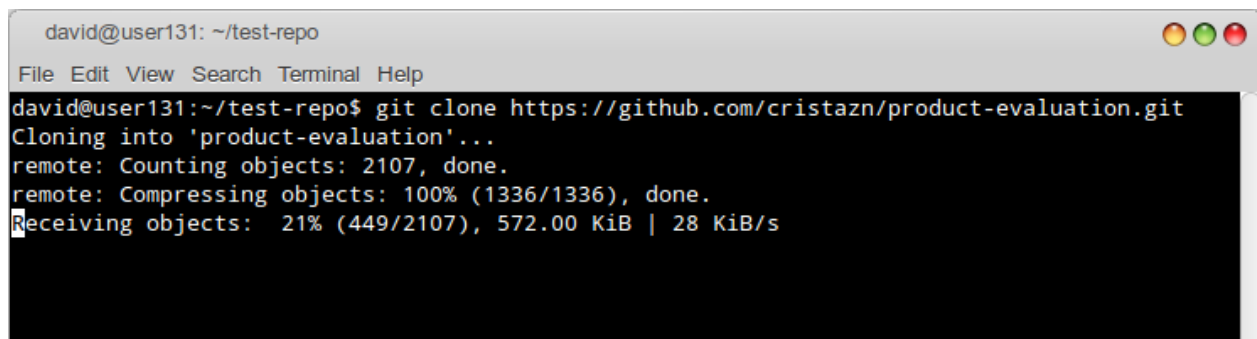
Giả sử ta đã có 1 repo trên GitHub công việc tiến hành đầu tiên của chúng ta là tìm đến trang chủ của repo đó và copy đoạn code URL của nó :



Hình 24 :URL Github

Dùng giao diện command gõ dòng lệnh dạng :

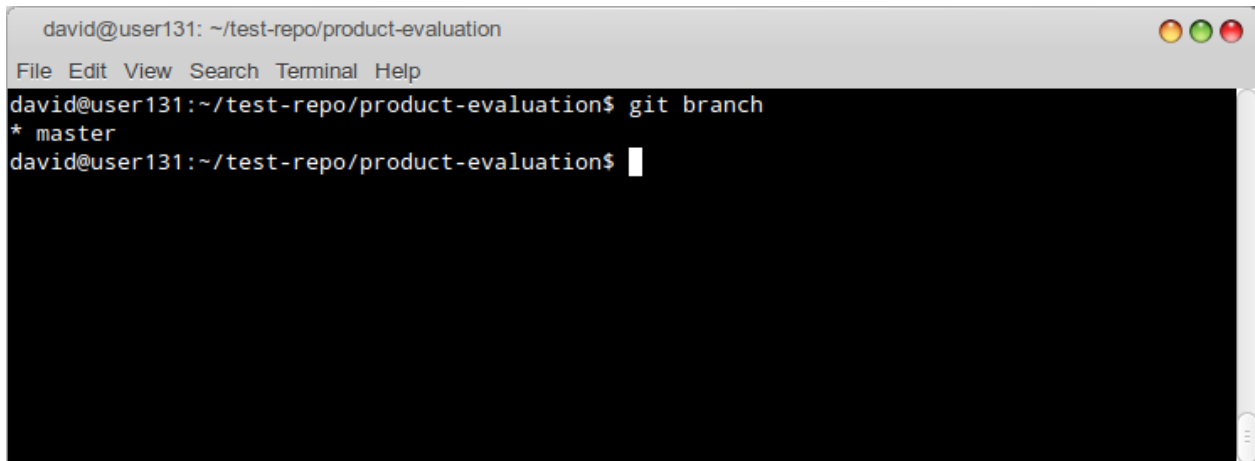
***git clone <URL> : dòng lệnh này cho phép ta nhân bản 1 repo để vào thư mục hiện hành để sử dụng.***



Hình 25 : git clone

Việc làm này ta sẽ sao chép tất cả các repo về máy local.

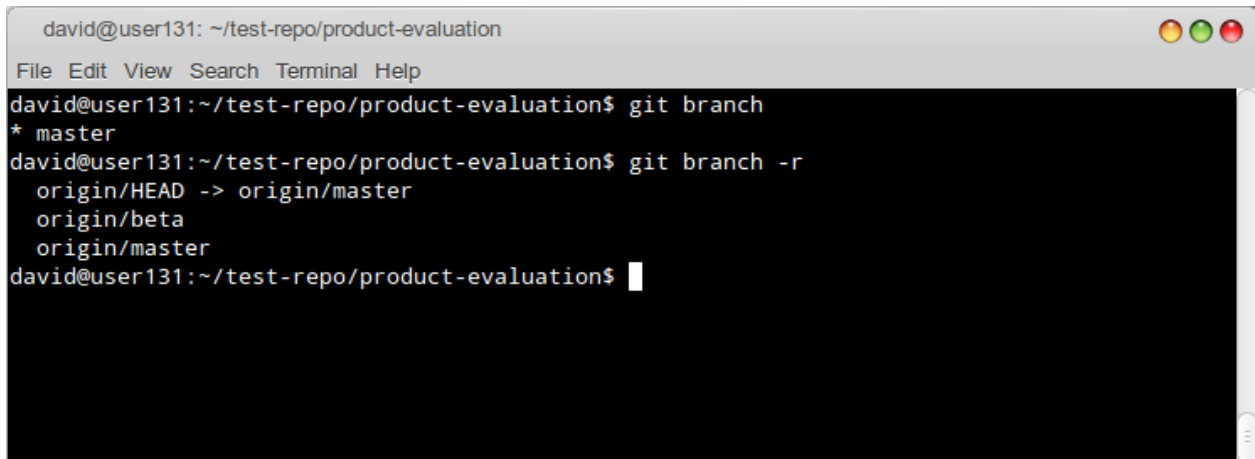
**git branch [<branch name>]** : xem các branch trên máy local. Nếu để tham số thì lệnh này có ý nghĩa là tạo branch mới nếu branch chưa tồn tại hoặc chuyển sang branch có tên

A terminal window titled 'david@user131: ~/test-repo/product-evaluation' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'git branch' has been executed, resulting in the output '\* master'.

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git branch
* master
david@user131:~/test-repo/product-evaluation$
```

Hình 26 : git branch

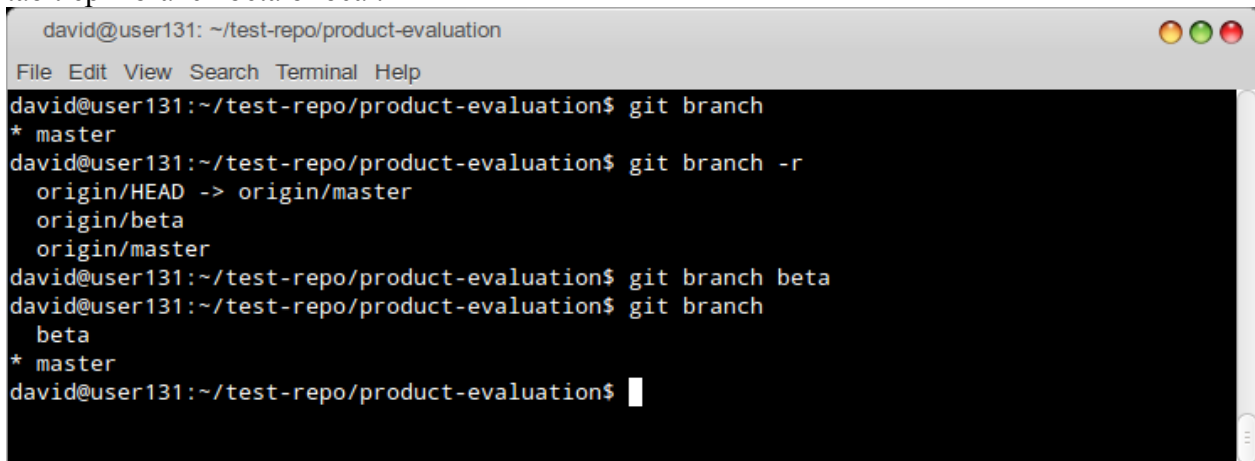
Ta dùng thêm tham số -r để tìm các branch remote đang tồn tại trong máy local.

A terminal window titled 'david@user131: ~/test-repo/product-evaluation' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'git branch' is executed first, showing '\* master'. Then, the command 'git branch -r' is executed, showing 'origin/HEAD -> origin/master', 'origin/beta', and 'origin/master'.

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git branch
* master
david@user131:~/test-repo/product-evaluation$ git branch -r
  origin/HEAD -> origin/master
  origin/beta
  origin/master
david@user131:~/test-repo/product-evaluation$
```

Hình 27 : git branch -r

Ở đây chúng ta có 2 branch master, beta ở remote và 1 branch master ở local. Ta có thể tạo tiếp 1 branch beta ở local.

A terminal window titled 'david@user131: ~/test-repo/product-evaluation' with a menu bar (File, Edit, View, Search, Terminal, Help). The sequence of commands and outputs is: 'git branch' (output: '\* master'), 'git branch -r' (output: 'origin/HEAD -> origin/master', 'origin/beta', 'origin/master'), 'git branch beta' (no output), and a second 'git branch' (output: 'beta', '\* master').

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git branch
* master
david@user131:~/test-repo/product-evaluation$ git branch -r
  origin/HEAD -> origin/master
  origin/beta
  origin/master
david@user131:~/test-repo/product-evaluation$ git branch beta
david@user131:~/test-repo/product-evaluation$ git branch
  beta
* master
david@user131:~/test-repo/product-evaluation$
```

Hình 28 : Tạo branch beta.

Ta xem danh sách file trong branch master

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ ls
Makefile Makefile_master plan readme README.md source TODO
david@user131:~/test-repo/product-evaluation$
```

Ta chuyển sang branch beta bằng dòng lệnh sau :  
*git checkout beta*

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git branch
* master
david@user131:~/test-repo/product-evaluation$ git branch -r
origin/HEAD -> origin/master
origin/beta
origin/master
david@user131:~/test-repo/product-evaluation$ git branch beta
david@user131:~/test-repo/product-evaluation$ git branch
beta
* master
david@user131:~/test-repo/product-evaluation$ git checkout beta
Switched to branch 'beta'
david@user131:~/test-repo/product-evaluation$
```

Xem danh sách file trong branch beta

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git checkout beta
Switched to branch 'beta'
david@user131:~/test-repo/product-evaluation$ ls
Makefile Makefile_master plan readme README.md source TODO
david@user131:~/test-repo/product-evaluation$
```

Tạo mới 1 file tên là beta\_test



```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
Switched to branch 'beta'
david@user131:~/test-repo/product-evaluation$ ls
Makefile Makefile_master plan readme README.md source TODO
david@user131:~/test-repo/product-evaluation$ gedit beta_test

(gedit:2556): Gtk-WARNING **: Theme parsing error: gtk-widgets.css:1947:11: Not using units
is deprecated. Assuming 'px'.

(gedit:2556): Gtk-WARNING **: Failed to parse /usr/share/themes/mac-os-lion-theme/gtk-3.0/s
ettings.ini: Key file contains line '/* ' which is not a key-value pair, group, or comment
david@user131:~/test-repo/product-evaluation$ ls
beta_test Makefile Makefile_master plan readme README.md source TODO
david@user131:~/test-repo/product-evaluation$
```

Thực hiện lệnh *git add .* và *git commit -m "< Message >"* để cập nhật trạng thái mới của branch beta. Ta quay trở lại branch master và xem danh sách file ta thấy không xuất hiện file beta\_test như vậy chứng tỏ sự độc lập của các branch. Các branch cho phép ta thay đổi nhưng không làm ảnh hưởng đến các phần đã ổn định trong project.

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git add .
david@user131:~/test-repo/product-evaluation$ git commit -m 'add new file'
[beta abffed5] add new file
0 files changed
create mode 100644 beta_test
david@user131:~/test-repo/product-evaluation$ git checkout master
Switched to branch 'master'
david@user131:~/test-repo/product-evaluation$ ls
Makefile Makefile_master plan readme README.md source TODO
david@user131:~/test-repo/product-evaluation$
```

Quay trở lại branch beta. Ta thêm dòng "This is line from beta branch" vào đầu file.

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git checkout beta
Switched to branch 'beta'
david@user131:~/test-repo/product-evaluation$ gedit beta_test

(gedit:2815): Gtk-WARNING **: Theme parsing error: gtk-widgets.css:1947:11: Not using units
is deprecated. Assuming 'px'.

(gedit:2815): Gtk-WARNING **: Failed to parse /usr/share/themes/mac-os-lion-theme/gtk-3.0/s
ettings.ini: Key file contains line '/* ' which is not a key-value pair, group, or comment
david@user131:~/test-repo/product-evaluation$ cat beta_test
This is line from beta branch
david@user131:~/test-repo/product-evaluation$ git add .
david@user131:~/test-repo/product-evaluation$ git commit -m 'add new line'
[beta 699f2a6] add new line
1 file changed, 1 insertion(+)
david@user131:~/test-repo/product-evaluation$
```

Để xem được sự thay đổi trên branch master thì ngay trên branch master ta phải thực hiện lệnh *git merge <branch name>* cho phép ta hợp nhất 2 trạng thái thành 1

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git checkout master
Switched to branch 'master'
david@user131:~/test-repo/product-evaluation$ git merge beta
Updating 30a4984..699f2a6
Fast-forward
 beta_test | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 beta_test
david@user131:~/test-repo/product-evaluation$ ls
beta_test Makefile Makefile_master plan readme README.md source TODO
david@user131:~/test-repo/product-evaluation$ cat beta_test
This is line from beta branch
david@user131:~/test-repo/product-evaluation$
```

Ta thấy danh sách file đã thay đổi và file đã có nội dung ở branch master. Ta thay đổi dòng nội dung “This is a line from master branch”.

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ gedit beta_test

(gedit:2941): Gtk-WARNING **: Theme parsing error: gtk-widgets.css:1947:11: Not using units
is deprecated. Assuming 'px'.

(gedit:2941): Gtk-WARNING **: Failed to parse /usr/share/themes/mac-os-lion-theme/gtk-3.0/s
ettings.ini: Key file contains line '/* ' which is not a key-value pair, group, or comment
david@user131:~/test-repo/product-evaluation$ git add .
david@user131:~/test-repo/product-evaluation$ git commit -m 'add text from master'
[master 89f9250] add text from master
 1 file changed, 1 insertion(+), 1 deletion(-)
david@user131:~/test-repo/product-evaluation$ cat beta_test
This is a line from master branch
david@user131:~/test-repo/product-evaluation$
```


Sự thay đổi này chỉ đang ở dạng cục bộ local. So sánh trên remote thì ta thấy trên repo chưa có sự thay đổi (không có file `beta_test`).

The screenshot shows the GitHub interface for the repository 'cristazn / product-evaluation'. At the top, there are buttons for 'Pull Request', 'Unwatch', 'Star' (0), and 'Fork' (0). Below these are tabs for 'Code', 'Network', 'Pull Requests' (1), 'Issues' (2), 'Wiki', 'Graphs', and 'Settings'. The 'Code' tab is selected, showing a description: 'This is a website support users update hi-tech information and choose product. — Read more'. Below the description are buttons for 'ZIP', 'HTTP', 'SSH', 'Git Read-Only', and a text input field containing the repository URL: 'https://github.com/cristazn/product-evaluation.git'. To the right of the URL is a 'Read+Write access' button. Below the URL bar, there's a 'branch: master' dropdown and tabs for 'Files', 'Commits', 'Branches' (2), and 'Tags'. The 'Files' tab is selected, showing a list of files and their commit history. A red box highlights the file list on the left: '.idea', 'source', '.gitignore', 'Makefile', 'Makefile\_master', 'README.md', 'TODO', 'plan', and 'readme'. The commit history for each file is shown on the right, with the latest commit being '30a4984802'.

Để có sự thay đổi ta phải push dữ liệu lên trên repo. Ta gõ lệnh *git push <repo name> <branch>* lệnh này sẽ đưa dữ liệu từ local với branch xxx lên repo (mặc định là repo origin).

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git push origin master
Username for 'https://github.com': cristazn
Password for 'https://cristazn@github.com':
To https://github.com/cristazn/product-evaluation.git
 30a4984..89f9250  master -> master
david@user131:~/test-repo/product-evaluation$
```

Khi refresh lại trình duyệt ta thấy.

PUBLIC  cristazn / product-evaluation

[Pull Request](#) [Unwatch](#) [Star](#) [0](#) [Fork](#) [0](#)

[Code](#) [Network](#) [Pull Requests 1](#) [Issues 2](#) [Wiki](#) [Graphs](#) [Settings](#)


This is a website support users update hi-tech information and choose product. — [Read more](#)











[ZIP](#) [HTTP](#) [SSH](#) [Git Read-Only](#) <https://github.com/cristazn/product-evaluation.git> [Read-Write access](#)

branch: **master** [Files](#) [Commits](#) [Branches 2](#) [Tags](#)

**product-evaluation** / [+](#) [37 commits](#)

add text from master

 cristazn authored 10 minutes ago latest commit 89f9250134 [+](#)

 .idea	4 hours ago	line from alpha [cristazn]
 source	a day ago	ddd [cristazn]
 .gitignore	3 days ago	Initial commit [cristazn]
 Makefile	a day ago	dasdsa [cristazn]
 Makefile_master	a day ago	Added Makefile_test [cristazn]
 README.md	3 days ago	Initial commit [cristazn]
 TODO	2 days ago	new init [cristazn]
 beta_test	10 minutes ago	add text from master [cristazn]
 plan	a day ago	Add plan file [cristazn]
 readme	4 hours ago	line from alpha [cristazn]

Đã có tồn tại file beta\_test điều này có nghĩa là phiên bản ổn định master đã được cập nhật thành công. Không chỉ ở đó khi ta xem nội dung file ta còn có thể điều nghiên được có bao nhiêu thay đổi đã áp dụng lên file này (trong mục commit).

add text from master

cristazn authored 14 minutes ago 1 parent 699f2a6 commit 89f9250134719f565b18132a0dbdbca865030cb

Showing 1 changed file with 1 addition and 1 deletion

beta\_test

@@ -1 +1 @@

-This is line from beta branch

+This is a line from master branch

0 notes on commit 89f9250

Write Preview

Comments are parsed with GitHub Flavored Markdown

Leave a comment

Dấu “+” thể hiện cho nội dung mới hiện hữu, dấu “-” thể hiện cho nội dung đã bị thay đổi trước đó.

Như đã nói lúc đầu git cho phép ta nhảy đến bất cứ vị trí nào. Điều ta cần là mã SHA1 để có thể thực hiện lệnh nhảy. Quay trở lại giao diện dòng lệnh ta thực hiện thay đổi trên file beta\_test. Thêm 1 dòng “sample text at end of file”.

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git add .
david@user131:~/test-repo/product-evaluation$ git commit -m 'add text end of file from master'
[master 0308937] add text end of file from master
1 file changed, 5 insertions(+)
david@user131:~/test-repo/product-evaluation$ cat beta_test
This is a line from master branch

sample text at end of file
david@user131:~/test-repo/product-evaluation$
```

Bây giờ xem lại log để xác định trạng thái ta gõ lệnh git log

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
commit 030893756428474f8debd1a44b89925019f30066
Author: Le Bao Triet <crist.azn@gmail.com>
Date: Mon Feb 4 22:30:31 2013 +0700

add text end of file from master

commit 89f9250134719f565b18132a0dbddbca865030cb
Author: Le Bao Triet <crist.azn@gmail.com>
Date: Mon Feb 4 22:09:12 2013 +0700

add text from master

:
```

Phần màu đỏ là git SHA1 và phần màu xanh là các message khi ta commit bằng lệnh git commit -m "message". Ta dùng các message này để gọi nhớ thông tin. Bây giờ ta quay lại trạng thái trước tức là trạng thái ta mới có 1 dòng "This is a line from master branch" trong file beta\_test. Ta dùng lệnh *git reset [--hard] [--soft] [--mixed] [--merge] <SHA1> [HEAD]*

*--hard* : sẽ reset index trong stage của file đang thay đổi và cách này sẽ reset luôn cả file.

*--mixed* : không reset nội dung file (nó không xóa working tree), nó chỉ reset index trong stage của file và lúc này file ở trạng thái là unstaged và phải được thực hiện lệnh git add để file được staged.

*--soft* : tương tự với mixed nhưng nó sẽ làm dùm mình là staged file.

*--merge* : tương tự với mixed nhưng khi thực hiện ở các branch chứa các nhánh bị merge rồi thì nó sẽ cắt các liên kết merge với các file đó.

Ở đây ta thực hiện lệnh *git reset --hard <SHA1>* để đưa file về trạng thái cũ nghĩa là trạng thái lúc mà ta commit file lúc 22:09:12s .

Nội dung file hiện thời ở master branch

```
beta_test (~/test-repo/product-evaluation) - gedit
File Edit View Search Tools Documents Help

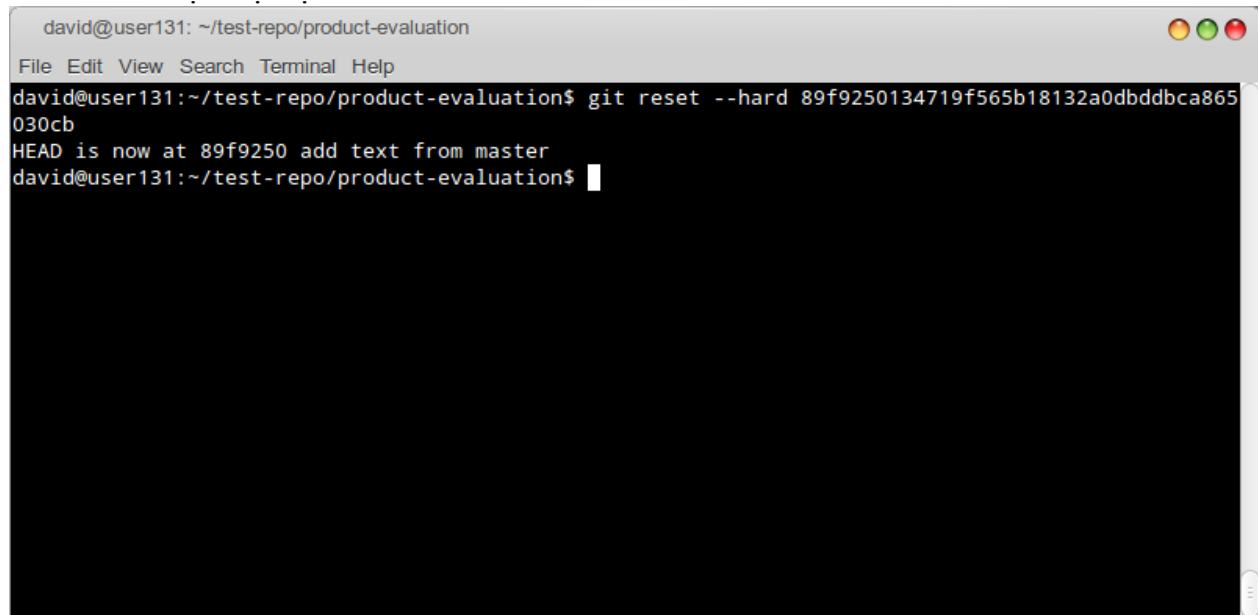
Open Save Undo

beta_test x
This is a line from master branch

sample text at end of file|

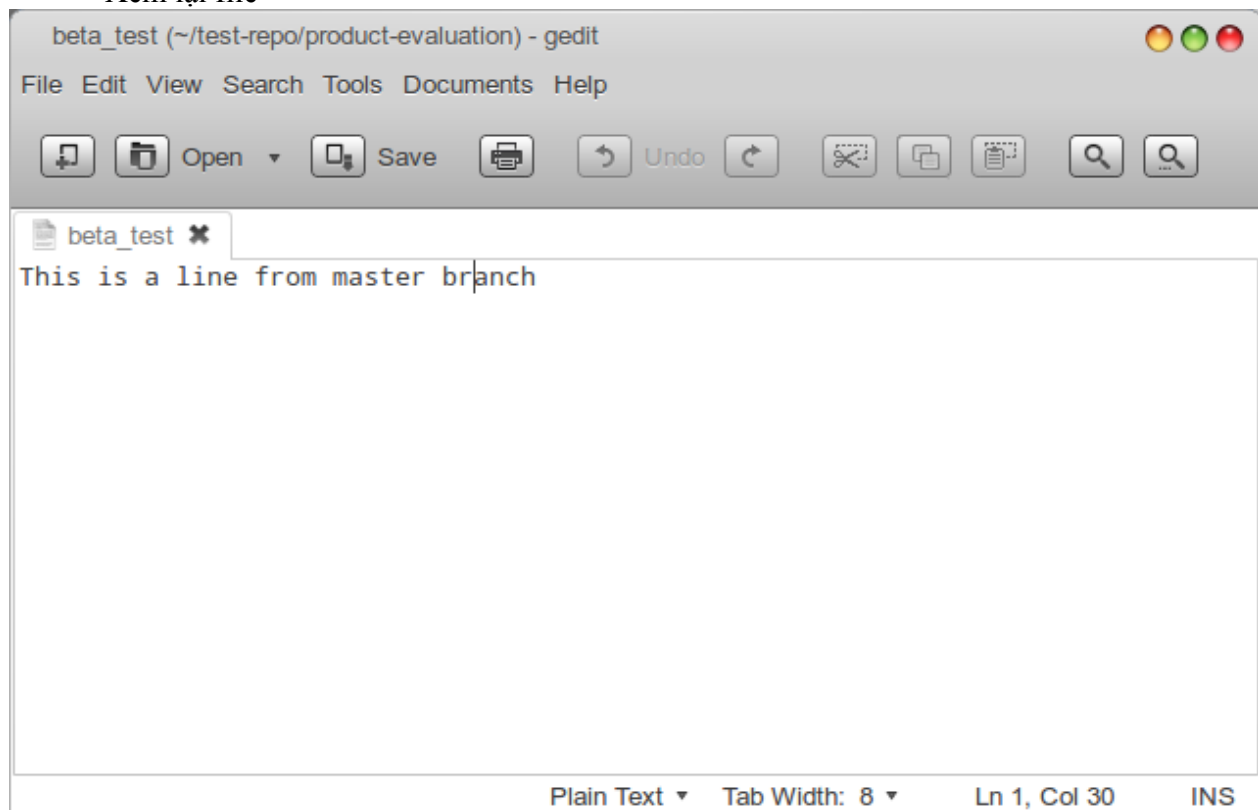
Plain Text Tab Width: 8 Ln 6, Col 27 INS
```

### Khi thực hiện lệnh



```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git reset --hard 89f9250134719f565b18132a0dbddbca865030cb
HEAD is now at 89f9250 add text from master
david@user131:~/test-repo/product-evaluation$
```

### Xem lại file



```
beta_test (~/.test-repo/product-evaluation) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
beta_test x
This is a line from master branch
Plain Text Tab Width: 8 Ln 1, Col 30 INS
```

Ta thấy nội dung file đã lùi về trước. Ta xem lại log

```
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
commit 89f9250134719f565b18132a0dbddbca865030cb
Author: Le Bao Triet <cris.azn@gmail.com>
Date:   Mon Feb 4 22:09:12 2013 +0700

    add text from master

commit 699f2a694e7086726f611298449416d72790860f
Author: Le Bao Triet <cris.azn@gmail.com>
Date:   Mon Feb 4 21:55:29 2013 +0700

    add new line

commit abffed551d647a357fe8df6996d0151222d76707
Author: Le Bao Triet <cris.azn@gmail.com>
Date:   Mon Feb 4 21:45:22 2013 +0700

    add new file

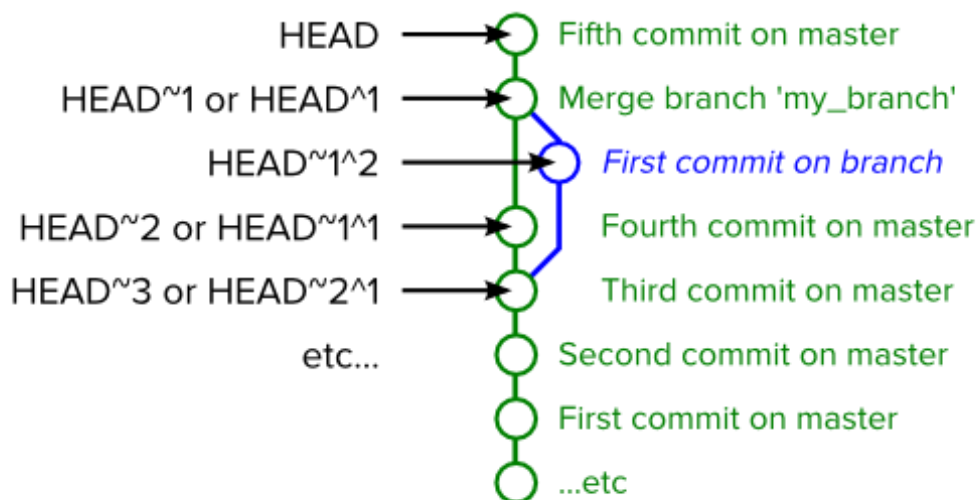
commit 30a4984802a66cb7621bffa4b84a4492faa60147e
goto mark: █
```

Ta thấy log đã lùi về thời điểm mà ta đã chọn. Ta có thể dùng từ khóa là HEAD để chỉ định thời điểm lùi.

VD : git reset –hard HEAD sẽ lùi về 1 bước cha trước đó. tương tự

git reset –hard HEAD^2 : lùi 2 bước ... Ngoài ra ta có thể dùng thêm dấu “~”. Tức là git reset –hard HEAD~2. Ta cũng có thể kết hợp dấu “^” và dấu “~”. Theo hình sau:

### Referencing commits from HEAD using ~ and ^



Sau khi thực hiện xong các bước thay đổi. Giả sử ta đang làm ở branch beta khi ta thực hiện xong thì ta có nhu cầu cho phép người ta sử dụng tính năng do mình tạo ra ở branch beta. Thì ở branch beta ta sẽ thực hiện lệnh *git merge master* lệnh này cho phép hợp nhất các chức năng trong branch beta với branch master.



Cuối cùng ta dùng lệnh *git push origin <branch>* để cập nhật thay đổi ở local lên remote trên *GitHub*.

```

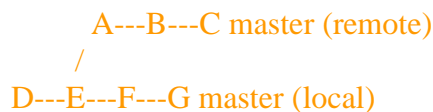
david@user131: ~/test-repo/product-evaluation
File Edit View Search Terminal Help
david@user131:~/test-repo/product-evaluation$ git push origin master
Username for 'https://github.com': cristazn
Password for 'https://cristazn@github.com':
Everything up-to-date
david@user131:~/test-repo/product-evaluation$

```

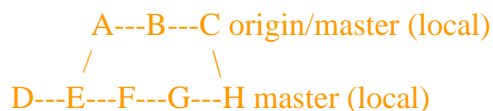
Nó sẽ yêu cầu ta gõ username/password. Sau đó trên remote đã có thay đổi .

Khi ta bắt đầu làm việc trên project việc đầu tiên là chúng ta cần kiểm tra xem ở remote đã có nội dung thay đổi chưa và nếu có thay đổi thì cập nhật trạng thái mới nhất về project local để sử dụng. Khi đó ta sẽ sử dụng lệnh *git pull <remote> <branch>* ***git pull sẽ lấy dữ liệu từ remote với branch do ta chỉ định. Mặc định remote là origin và branch là tất cả branch sau đó hợp nhất với code ở local.*** Lệnh này là ghép nối của 2 lệnh là git fetch và git merge vì thế đôi khi ta sử dụng pull nó sẽ gây ra kết quả không mong muốn. VD : khi ta có dữ liệu trên repo chưa hoàn chỉnh thì lệnh này sẽ đưa các code không hoàn chỉnh đó và hợp nhất với code mới và vì thế làm gây ra lộn xộn. Do đó cách hay hơn là ta dùng *git fetch* và *git merge* riêng rẽ để ta có thể kiểm soát tốt.

VD : branch trên là ở remote. Ở dưới là local



Khi dùng lệnh git pull origin thì nó sẽ chuyển sang



Ta để ý có trạng thái H được tạo ra. H thực ra là trạng thái do lệnh git merge tạo ra. Thêm 1 câu hỏi nữa là tại sao ở hình trên lại remote và local còn hình dưới là local – local.

Hình trên là sự so sánh dữ liệu giữa remote và local :

remote : D<-E<-A<-B<-C

local : D<-E<-F<-G

Còn hình dưới nói về các trạng thái sau khi thực hiện lệnh ở local. Bản thân lệnh git không sync (đồng bộ hóa) mà nó chỉ lấy dữ liệu về còn sau này thì có nhu cầu cập nhật lại ta dùng lệnh git push. Do đó khi thực hiện git pull dữ liệu trên remote không đổi.

***Git fetch <remote> nó sẽ lấy dữ liệu bên trên remote về local. Nó sẽ gắn 1 branch khác trên trạng thái vì thế nó không ảnh hưởng đến trạng thái của branch hiện hành mặc dù branch là giống nhau.***

*Ngoài lề tỳ :* Lệnh này khác so với ***git clone*** vì git clone cho tác dụng khi ta khởi tạo nó sẽ sao chép tất cả các dữ liệu trên remote repo và tạo cho ta 1 local repo trên chính máy tính các remote branch master, .... được sao chép và chuyển thành master local và chúng ta thao tác trên branch này. Thật ra khi tạo branch thì nó tạo luôn cả origin/master (cái này ở local nha). Dùng lệnh ***git branch -r*** sẽ thấy... cái này là để đánh dấu cho cái remote branch trên remote repo để có thể cập nhật đúng khi ta dùng lệnh push.