



Trung Tâm Tin học
Trường Đại Học KHTN TP.HCM

GITHUB

CẨM NANG HƯỚNG DẪN SỬ DỤNG



Mục lục

1. GIT – HỆ THỐNG QUẢN LÝ PHIÊN BẢN PHÂN TÁN	3
1.1. GIỚI THIỆU.....	3
1.2. QUY TRÌNH XỬ LÝ CÔNG VIỆC (WORKFLOW) TRÊN GIT.....	4
1.3. CÁC KHÁI NIỆM CƠ BẢN	6
2. GITHUB – DỊCH VỤ LƯU TRỮ MÃ NGUỒN ONLINE	15
2.1. TÀI KHOẢN GITHUB.....	16
2.2. GITHUB DESKTOP	22
3. SỬ DỤNG GITHUB	26
3.1. TẠO KHO (REPOSITORY) LƯU TRỮ PROJECT.....	26
3.2. CLONE KHO LƯU TRỮ TỪ GITHUB VỀ GITHUB DESKTOP.....	30
3.3. COMMIT VÀ ĐỒNG BỘ KHO LƯU TRỮ GIỮA GITHUB DESKTOP VÀ GITHUB.COM	31
3.4. KHÔI PHỤC LẠI PHIÊN BẢN CŨ.....	33
4. PHỐI HỢP THỰC HIỆN PROJECT.....	34
4.1. MỜI THÀNH VIÊN THAM GIA PROJECT	34
4.2. THẢO LUẬN - TRAO ĐỔI	35
4.3. QUẢN LÝ DỰ ÁN VỚI TAB PROJECTS	37
4.4. QUẢN LÝ CÁC VẤN ĐỀ (ISSUE)	39
4.5. LƯU TRẠNG THÁI (COMMIT) VÀ YÊU CẦU GỘP (PULL REQUEST)	40
4.6. XỬ LÝ TRÊN CÁC YÊU CẦU GỘP	44

Lời ngỏ

Cẩm nang này



..... • 1

Là tất cả những gì chúng tôi nghĩ bạn cần biết, để Git và GitHub trở thành công cụ không thể thiếu mỗi khi bạn mở máy lên và viết mã.



..... • 2

Là tất cả những gì bạn sẽ cần dùng khi làm việc trên Git và GitHub, để có thể đi xa hơn nữa trong thế giới Lập trình.



..... • 3

Bạn đã biết hay chưa biết gì về Git và GitHub? Không sao cả, chúng tôi hy vọng bạn sẽ tìm được điều gì đó hữu ích để bạn có thể bắt đầu sử dụng ngay và luôn.



..... • 4

Chúng tôi trân quý tất cả những ý kiến đóng góp để cẩm nang có thể hoàn thiện hơn. Hãy cho chúng tôi biết góp ý của bạn qua email daotaolaptrinh@gmail.com, bạn nhé!

1. Git – Hệ thống quản lý phiên bản phân tán

1.1. Giới thiệu

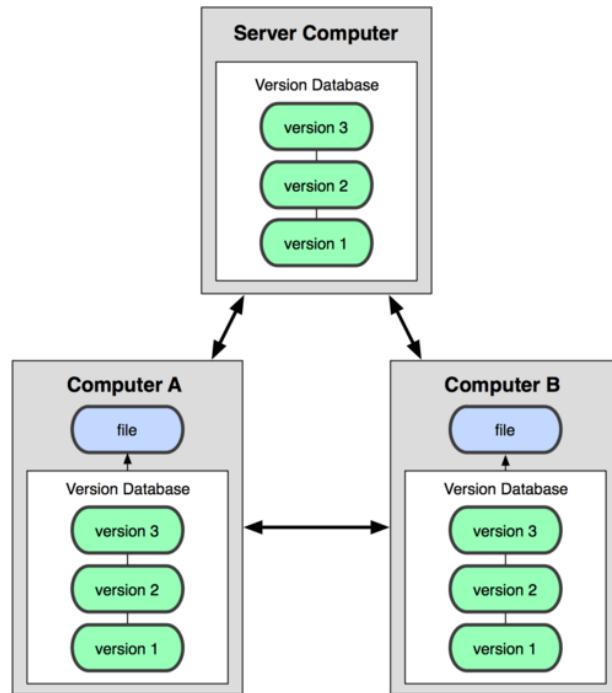
Khi lập trình, sẽ có lúc bạn lỡ tay xoá một đoạn code vì nghĩ rằng đoạn code đó không phù hợp nữa, nhưng sau đó lại phát hiện là đoạn code đó bạn vẫn cần dùng. Bạn nghĩ mình có thể nhớ lại chính xác những gì mình đã viết không? Thật sự là rất rất khó. Nhưng nếu bạn có dùng phần mềm quản lý phiên bản mã nguồn thì mọi việc sẽ trở nên đơn giản hơn rất nhiều vì phần mềm quản lý phiên bản mã nguồn sẽ cho phép bạn dễ dàng quay lại một phiên bản trước của tập tin đó. Có thể nói phần mềm quản lý mã nguồn là công cụ không thể thiếu đối với lập trình viên và một trong những phần mềm hỗ trợ quản lý phiên bản rất phổ biến hiện nay là Git.

Git sẽ giúp người dùng lưu lại các phiên bản của những lần thay đổi mã nguồn để dễ dàng khôi phục lại phiên bản cũ mà không cần phải nhớ là mình đã chỉnh ở đâu, tất cả phiên bản bạn cần đều đã được sao lưu.

Git giúp bạn:

- ✓ *Lưu lại được các phiên bản khác nhau của mã nguồn dự án phần mềm*
- ✓ *Khôi phục lại mã nguồn từ một phiên bản bất kỳ*
- ✓ *Dễ dàng so sánh giữa các phiên bản*
- ✓ *Phát hiện được ai đã sửa phần nào làm phát sinh lỗi*
- ✓ *Khôi phục lại tập tin bị mất*
- ✓ *Dễ dàng thử nghiệm, mở rộng tính năng của dự án mà không làm ảnh hưởng đến phiên bản chính (master branch)*
- ✓ *Giúp phối hợp thực hiện dự án trong nhóm một cách hiệu quả*

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) ra đời vào năm 2005 và hiện được dùng rất phổ biến. So với các hệ thống quản lý phiên bản tập trung khi tất cả mã nguồn và lịch sử thay đổi chỉ được lưu một nơi là máy chủ thì trong hệ thống phân tán, các máy khách không chỉ "check out" phiên bản mới nhất của các tập tin mà là sao chép (mirror) toàn bộ kho mã nguồn (repository). Như vậy, nếu như máy chủ ngừng hoạt động, thì bạn hoàn toàn có thể lấy kho chứa từ bất kỳ máy khách nào để sao chép ngược trở lại máy chủ để khôi phục lại toàn bộ hệ thống. Mỗi checkout thực sự là một bản sao đầy đủ của tất cả dữ liệu của kho chứa từ máy chủ.



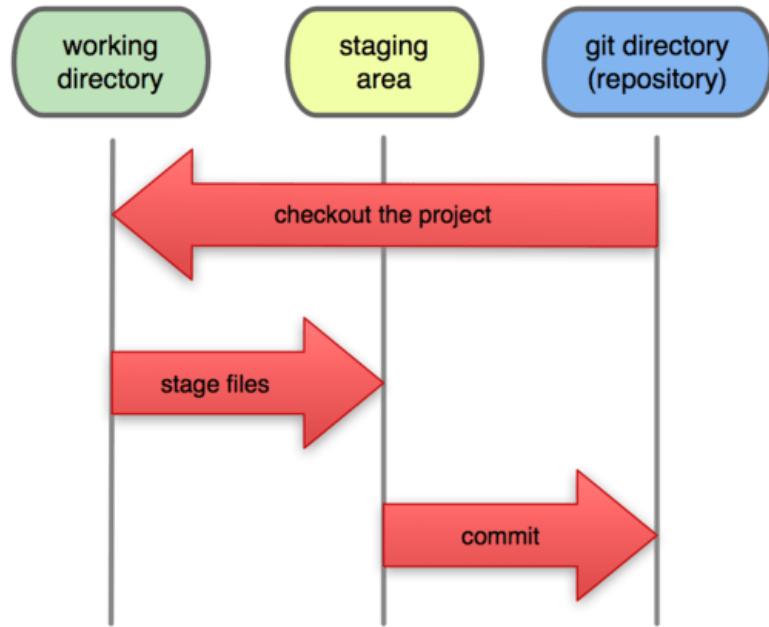
Mô hình tổ chức dữ liệu phân tán trên Git

1.2. Quy trình xử lý công việc (workflow) trên Git

Bạn lưu ý là bạn dùng Git để giúp bạn quản lý phiên bản mã nguồn, bạn không thể chỉnh code trong Git. Bạn vẫn phải thực hiện các công việc của mình trên môi trường làm việc với các chức năng tiện ích trên IDE của ngôn ngữ lập trình bạn đang làm việc. Thông thường, quy trình xử lý công việc trên Git sẽ như sau:

- Bạn thực hiện các công việc của mình, thay đổi các tập tin trong thư mục làm việc (working directory).
- Bạn chọn các các tập tin cần commit vào khu vực tổ chức (staging area) của Git.
- Bạn commit và các tập tin trong khu vực tổ chức sẽ được lưu trữ vĩnh viễn vào thư mục Git.

Local Operations



Trong đó:

- **Working directory:** là bản sao một phiên bản của dự án. Thư mục này là tất cả tập tin, thư mục của dự án được bung ra từ cơ sở dữ liệu được nén trong thư mục `.git` giúp bạn dễ dàng cập nhật, thay đổi.

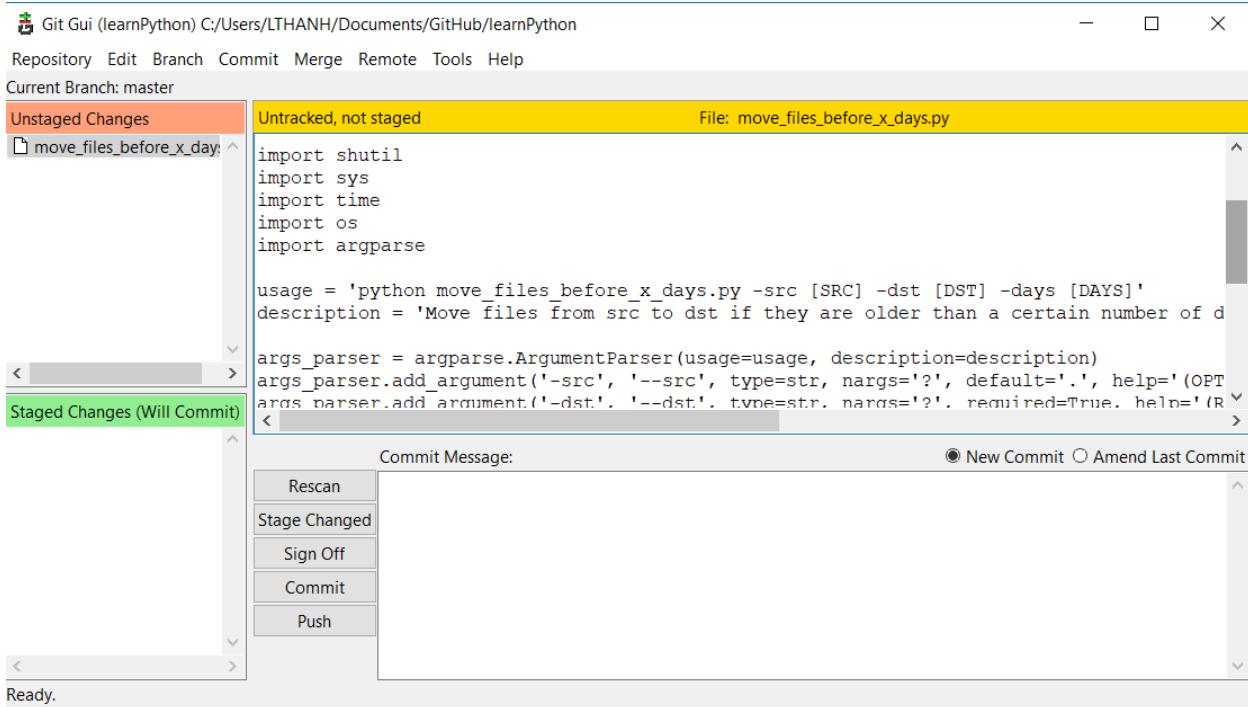
- **Staging area:**

Staging Area là nơi giữ tất cả những gì bạn cần commit. Trong các hệ thống quản lý phiên bản (Version Control System) thì các dữ liệu sẽ được lưu trữ ở hai nơi, một là thư mục bạn đang làm việc trên máy tính (working directory) và một là kho chứa mã nguồn (repository) sau khi bạn đã thực hiện thay đổi. Nhưng trên Git thì có thêm một khu vực trung gian gọi là Staging Area và đây chính là một lợi thế lớn của Git. Staging Area nghĩa là khu vực sẽ lưu trữ những tập tin thay đổi của bạn để có thể được commit, bạn muốn commit tập tin nào thì tập tin đó phải nằm trong Staging Area.

Mỗi tập tin trong Git được quản lý dựa trên ba trạng thái: committed, modified, và staged. Committed có nghĩa là dữ liệu đã được lưu trữ một cách an toàn trong cơ sở dữ liệu Git. Modified có nghĩa là bạn đã thay đổi tập tin nhưng chưa commit vào cơ sở dữ

liệu. Staged là bạn đã đánh dấu và sẽ commit tập tin đã chỉnh sửa đó trong lần commit sắp tới.

- **git directory:** Thư mục Git là nơi Git lưu trữ các "siêu dữ kiện" (metadata) và cơ sở dữ liệu phục vụ cho việc quản lý mã nguồn dự án của bạn. Đây là phần quan trọng nhất của Git, nó là phần được sao lưu về khi bạn tạo một bản sao (clone) của một kho chứa từ một máy tính khác.



Git GUI cho phép bạn xem được trạng thái trong working directory

1.3. Các khái niệm cơ bản

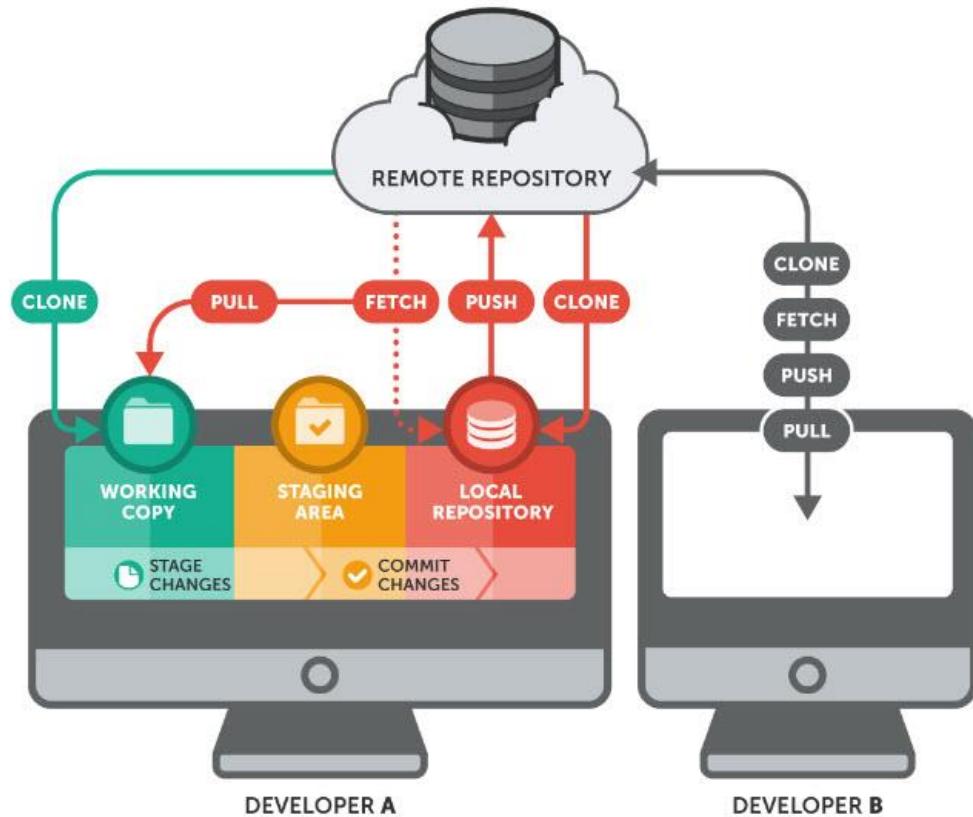
1.3.1. Repository – Kho lưu trữ

Trong Git, **Repository** là nơi lưu trữ, quản lý tất cả những thông tin cần thiết (thư mục, tập tin, ảnh, video, bảng biểu, dữ liệu...) cũng như các sửa đổi và lịch sử của toàn bộ dự án. Khi tạo mới repository, bạn nên tạo thêm tập tin README hoặc một tập tin thông tin giới thiệu về dự án của bạn.

Bạn có thể có nhiều cách tổ chức cho repository, trong lập trình C# trên Visual Studio, bạn có thể lưu trữ một solution trong một kho, solution đó có thể chứa nhiều project.

Có hai loại repository, đó là local repository và remote repository.

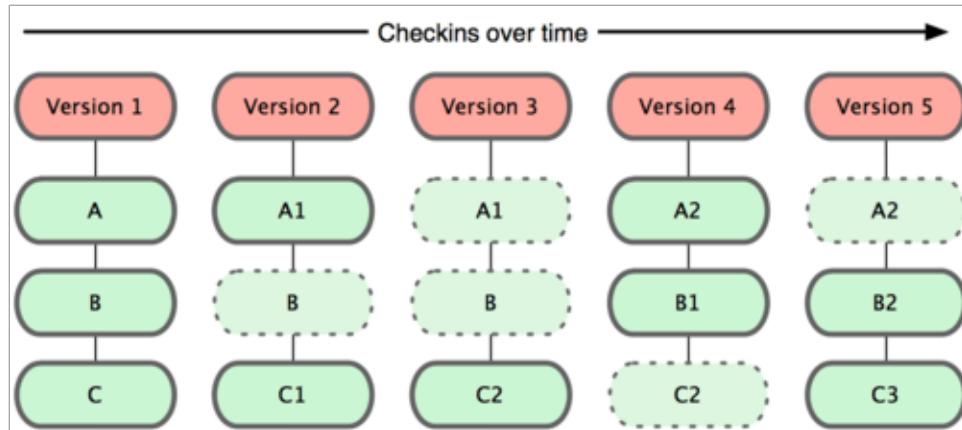
- Local repository: Là repository trên máy tính cục bộ, dành cho một người sử dụng. Local repository sẽ được đồng bộ hóa với remote repository bằng các lệnh của git.
- Remote repository: Là repository chia sẻ giữa nhiều người và được cài đặt trên server chuyên dụng, có thể trên GitHub.



Các thao tác xử lý giữa remote repository và local repository

1.3.2. Snapshot

Cơ chế lưu trữ phiên bản của Git là sau mỗi lần bạn thực hiện lưu trạng thái (commit) sẽ tạo ra một “ảnh chụp” (snapshot) lưu lại nội dung tất cả các tập tin, thư mục tại thời điểm đó rồi tạo tham chiếu tới snapshot đó. Để hiệu quả hơn, nếu như tập tin không có thay đổi, Git không lưu trữ tập tin đó lại mà chỉ tạo liên kết tới tập tin gốc đã tồn tại trước đó. Sau đó khi cần bạn hoàn toàn có thể khôi phục và sử dụng lại một snapshot, hay còn gọi là phiên bản nào đó. Đây cũng chính là lợi thế của Git khi nó không lưu trữ dữ liệu mà sẽ lưu dạng snapshot, giúp tiết kiệm không gian lưu trữ.



Git lưu trữ dữ liệu dưới dạng ảnh chụp (snapshot) của mã nguồn dự án theo thời gian

1.3.3. Commit

Commit là thao tác báo cho hệ thống biết bạn muốn lưu lại trạng thái hiện hành, ghi nhận lại lịch sử các xử lý như đã thêm, xóa, cập nhật các file hay thư mục nào đó trên repository.

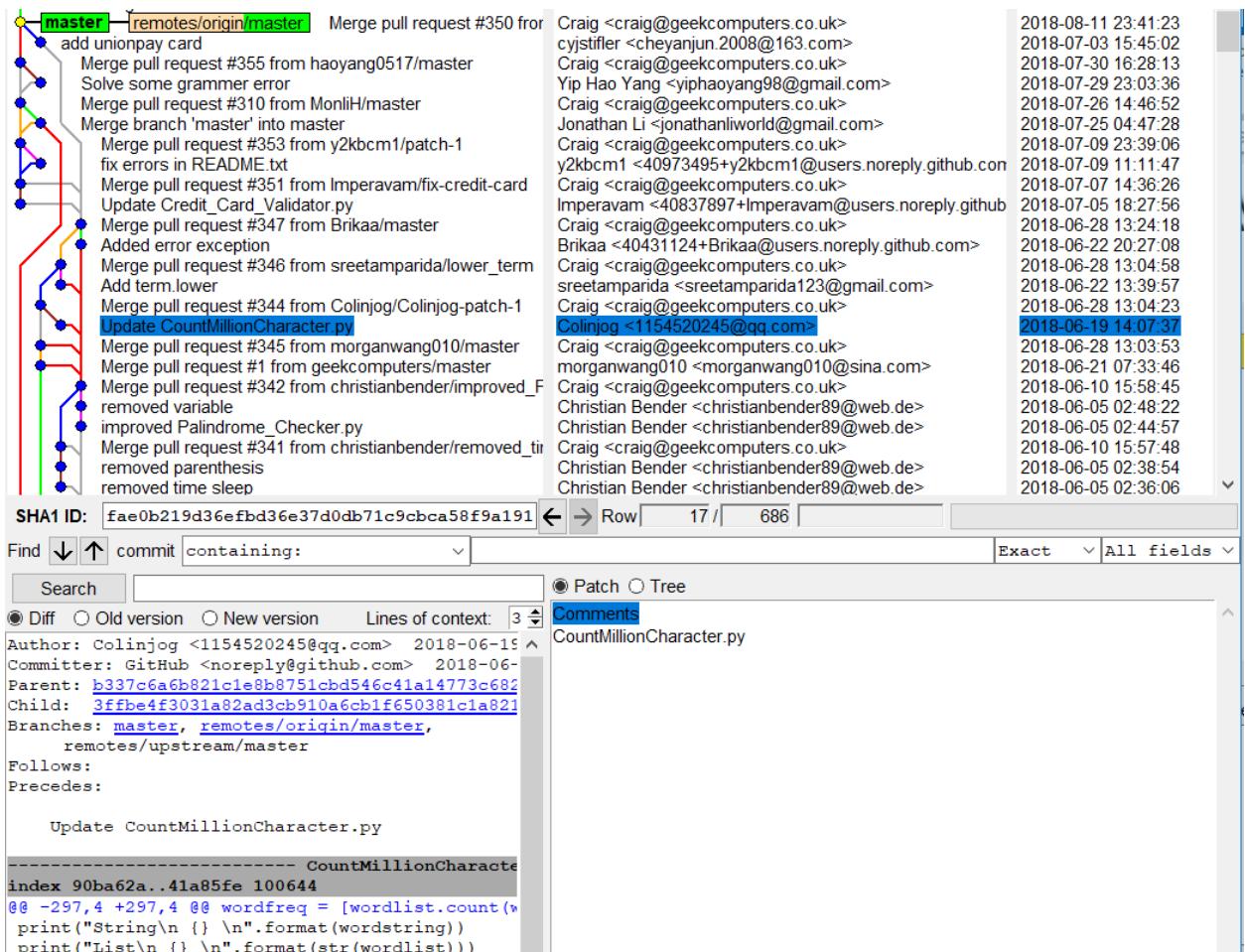
Khi thực hiện commit, trong repository sẽ ghi lại sự khác biệt từ lần commit trước với trạng thái hiện tại. Các commit ghi nối tiếp với nhau theo thứ tự thời gian do đó chỉ cần theo vết các commit thì có thể biết được lịch sử thay đổi trong quá khứ.



Khi bạn thực hiện commit, hệ thống đều yêu cầu bạn phải nhập vào commit message để ghi chú tóm tắt là trong lần commit này là bạn đã thực hiện những thay đổi nào, có ý nghĩa thế nào,...

1.3.4. Clone

Nếu bạn muốn có một bản sao của một kho chứa Git có sẵn, có thể là một dự án mà bạn tham gia – thì bạn hãy thực hiện **clone**. Đây là điểm khác biệt của Git so với một số hệ thống quản lý phiên bản mã nguồn khác vì **clone** là tạo ra một bản sao của gần như tất cả những gì của repository mà máy chủ đang lưu trữ. Bạn sẽ có được tất cả lịch sử đã xảy ra trên repository và hoàn toàn có thể quay lại, undo lại từ bất kỳ thời điểm commit nào. Và một điểm nữa là nếu ổ cứng máy chủ bị hư, bạn có thể sử dụng bất kỳ bản sao trên bất kỳ máy khách nào để khôi phục lại trạng thái của máy chủ.



The screenshot shows a GitHub repository's commit history. The main pane displays a complex merge history with many branches and pull requests. A specific commit is highlighted with a blue box: `Colinjog <1154520245@qq.com> 2018-06-19 14:07:37`. The commit message is: `Update CountMillionCharacter.py`. The commit details pane shows the commit's author, committer, parent, child, branches, and follows. The commit content pane shows the code changes for `CountMillionCharacter.py`, including the addition of a word frequency counting function.

```

Author: Colinjog <1154520245@qq.com> 2018-06-19 14:07:37
Committer: GitHub <noreply@github.com> 2018-06-19 14:07:37
Parent: b337c6a6b821c1e8b8751cbd546c41a14773c682
Child: 3ffbe4f3031a82ad3cb910a6cb1f650381c1a821
Branches: master, remotes/origin/master
Follows:
Precedes:

Update CountMillionCharacter.py

--- CountMillionCharacter.py
index 90ba62a..41a85fe 100644
@@ -297,4 +297,4 @@ wordfreq = [wordlist.count(w) for w in wordlist]
print("String\n{} \n".format(wordstring))
print("List\n{} \n".format(str(wordlist)))

```

Lịch sử các thay đổi trên repository khi clone về máy cục bộ

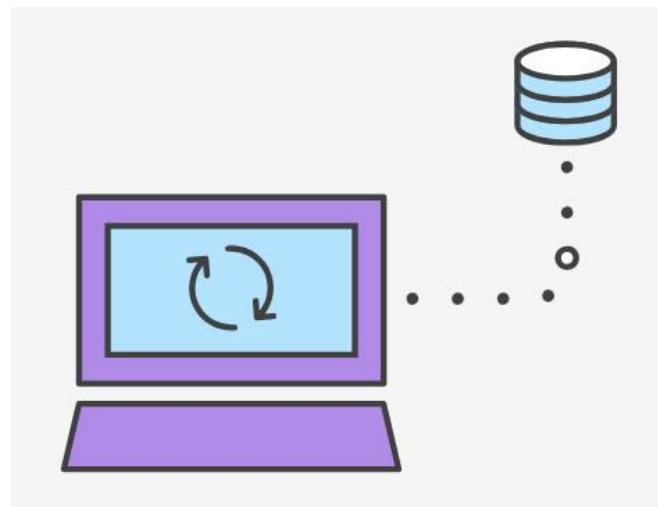
Bạn có thể **clone** từ bất kỳ kho chứa nào, nó sẽ sao chép luôn các thiết lập về repository và sẽ tự động tạo một master branch trên máy tính của bạn.

Có lưu ý ở đây là trên GitHub bạn có một cách khác để sao chép kho từ người khác là bạn thực hiện **fork** trên repository bạn cần. Điểm khác của fork là bạn có thể đóng góp thêm vào repository gốc bằng cách thực hiện

pull request. Khi chủ sở hữu của repository nơi bạn fork nhận được yêu cầu sẽ xem xét chỉnh sửa của bạn, nếu thấy hay sẽ tiến hành merge nội dung chỉnh sửa của bạn vào source gốc.

1.3.5. Push

Lệnh push được sử dụng để đưa nội dung kho lưu trữ cục bộ lên server. Push là cách bạn chuyển giao các commit từ kho lưu trữ cục bộ của bạn lên server.



1.3.6. Fetch

Khi ta thực hiện **fetch**, lệnh này sẽ truy cập vào repository trên server và kéo toàn bộ dữ liệu mà bạn chưa có từ repository trên server về. Sau đó, bạn có thể tích hợp dữ liệu vào branch bất kỳ lúc nào.

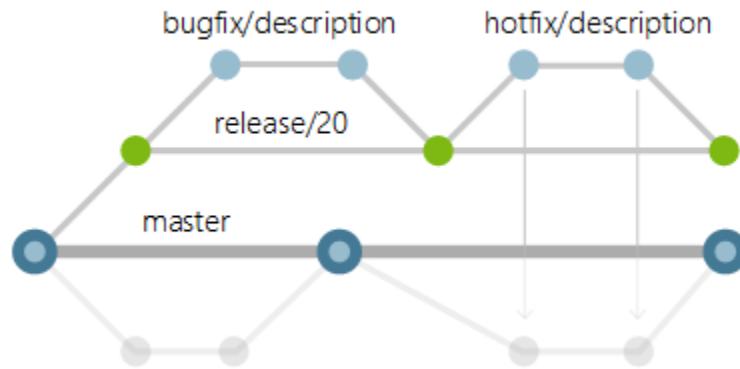
1.3.7. Pull

Lệnh này sẽ tự động lấy toàn bộ dữ liệu từ repository trên server và gộp vào cái branch hiện tại bạn đang làm việc.

1.3.8. Branch - Nhánh

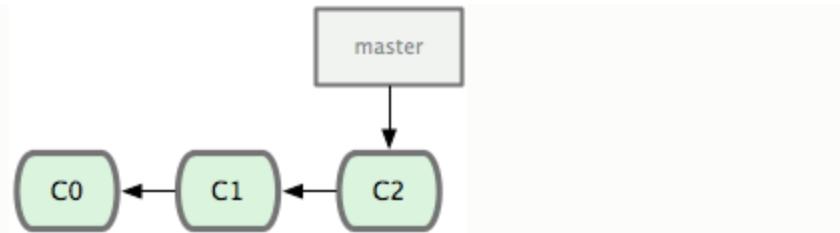
Nhánh là khái niệm rất hay trong Git, với nhánh bạn có thể tách riêng các tính năng của dự án, thử nghiệm các tính năng mới hay cũng có thể dùng nhánh để khắc phục, hoàn chỉnh lỗi nào đó của dự án,... Khi bắt đầu khởi tạo một repository hoặc clone một repository, bạn sẽ có một nhánh (branch) chính tên là **master**, đây là branch chứa toàn bộ các mã nguồn chính trong repository. Từ nhánh master này, trong quá trình thực hiện dự

án bạn có thể rẽ thêm nhiều nhánh khác tùy theo nhu cầu thực tế. Tất cả các nhánh đều được hệ thống lưu lại lịch sử các lần commit trên nhánh và bạn hoàn toàn quay lại mốc commit nào mà mình muốn.

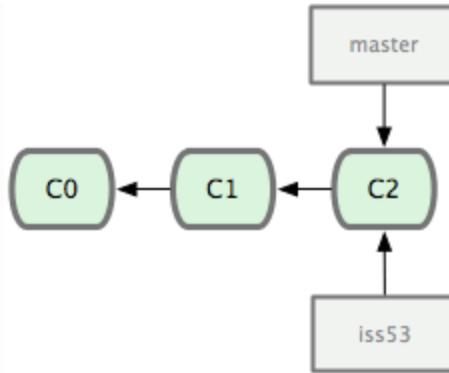


Một lưu ý nhỏ chõ này là tất cả thao tác fetch, push hoặc pull mặc định thực hiện trên nhánh hiện hành, nên bạn lưu ý nhánh mình đang thao tác là nhánh master hay nhánh nào để tránh sai sót.

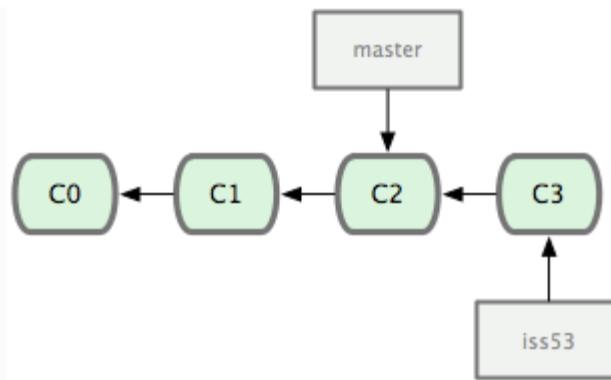
Giả sử bạn đang làm việc trên một dự án đã có một số commit từ trước như hình sau:



Bạn quyết định sẽ nâng cấp dự án qua việc giải quyết vấn đề số #53 (*cách đặt mã số vấn đề theo quy ước của dự án*) nên bạn sẽ tạo một nhánh mới **iss53** để cho lần nâng cấp đó, bạn sẽ giải quyết vấn đề #53. Bạn sẽ tạo nhánh mới **iss53**, vậy là từ lần commit C2 trong repository của bạn sẽ có 2 nhánh là **master** và **iss53**



Bạn tiếp tục làm việc trên nhánh **iss53** và sau đó thực hiện commit ở C3.



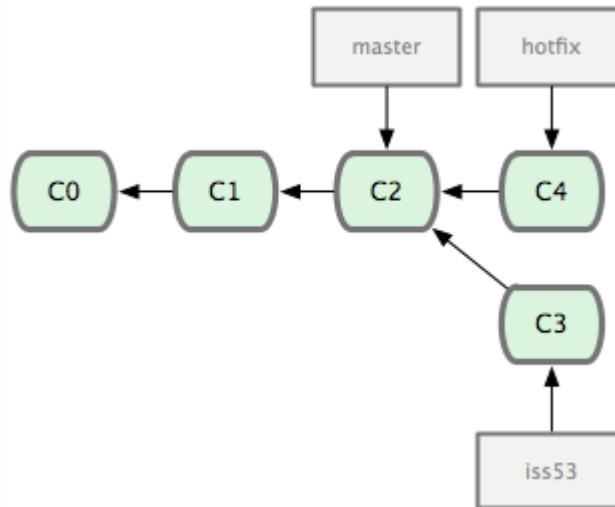
Bây giờ bạn nhận được thông báo rằng có một vấn đề với dự án và bạn cần khắc phục nó ngay lập tức. Bạn có vấn đề mình đang giải quyết (**iss53**) nhưng chưa xong và lỗi mới phát hiện, cần khắc phục. Với Git, bạn không cần phải tốn quá nhiều công sức để khôi phục lại các thay đổi trước khi bạn làm **iss53** để vá lỗi rồi làm tiếp **iss53**. Hoàn toàn độc lập, tất cả những gì bạn cần phải làm là chuyển lại nhánh master và lấy lại mã lệnh ở thời điểm mình cần.

Tuy nhiên, trước khi làm điều này, bạn nên lưu ý rằng nếu thư mục làm việc hoặc khu vực tổ chức có chứa các thay đổi chưa được commit mà xung đột với nhánh bạn đang làm việc, Git sẽ không cho phép bạn chuyển nhánh. Tốt nhất là bạn nên ở trạng thái làm việc "sạch" (đã commit hết) trước khi chuyển nhánh.

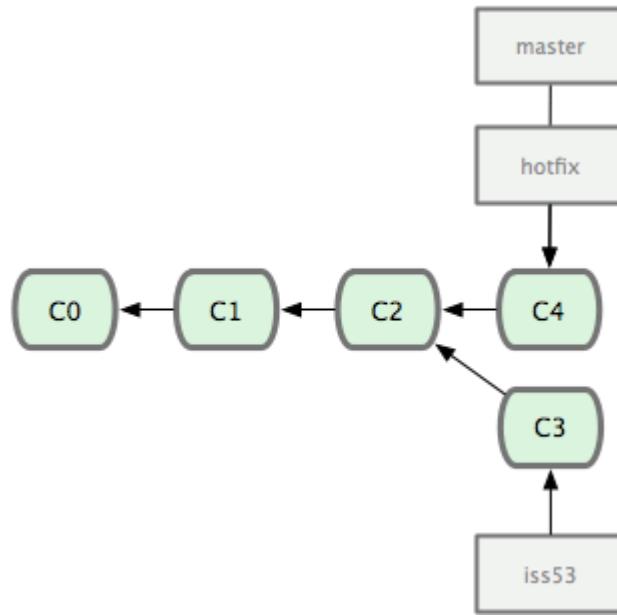
Bạn sẽ quay lại thời điểm commit mình cần, giả sử C2, lúc này thư mục làm việc của dự án giống hệt như trước khi bạn bắt đầu giải quyết vấn đề #53 và

bạn có thể tập trung vào việc sửa lỗi. Điểm quan trọng cần ghi nhớ ở đây là Git sẽ khôi phục lại thư mục làm việc của bạn giống như snapshot của lần commit C2. Git sẽ tự động thêm, xóa và sửa các tập tin sao cho đảm bảo rằng thư mục làm việc của bạn giống như lần commit C2.

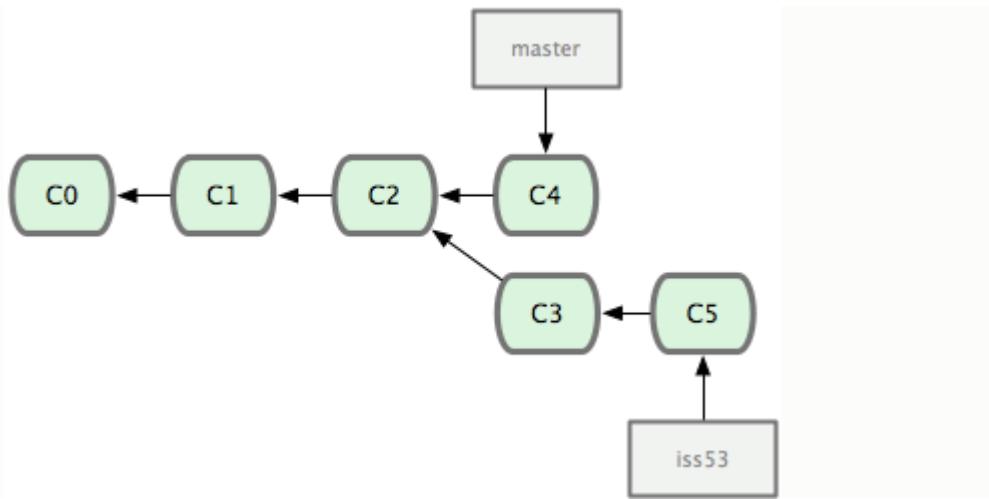
Để sửa lỗi, bạn tạo nhánh mới **hotfix** dựa trên nhánh master để giải quyết các lỗi mới phát sinh và bạn giải quyết xong vấn đề với lần commit C4.



Bạn có thể chạy để kiểm tra và chắc chắn rằng bản vá lỗi hoạt động đúng theo ý bạn muốn rồi sau đó tích hợp nó lại nhánh chính **master** để triển khai với thao tác **merge**. Để merge, bạn sẽ chuyển sang nhánh **master** trước rồi thực hiện merge nhánh **master** với nhánh **hotfix**. Sau khi merge những cập nhật mới của bạn bây giờ ở trong snapshot của commit C4 được trả tới bởi nhánh **master** và bạn có thể mang đi triển khai thực tế phiên bản này.

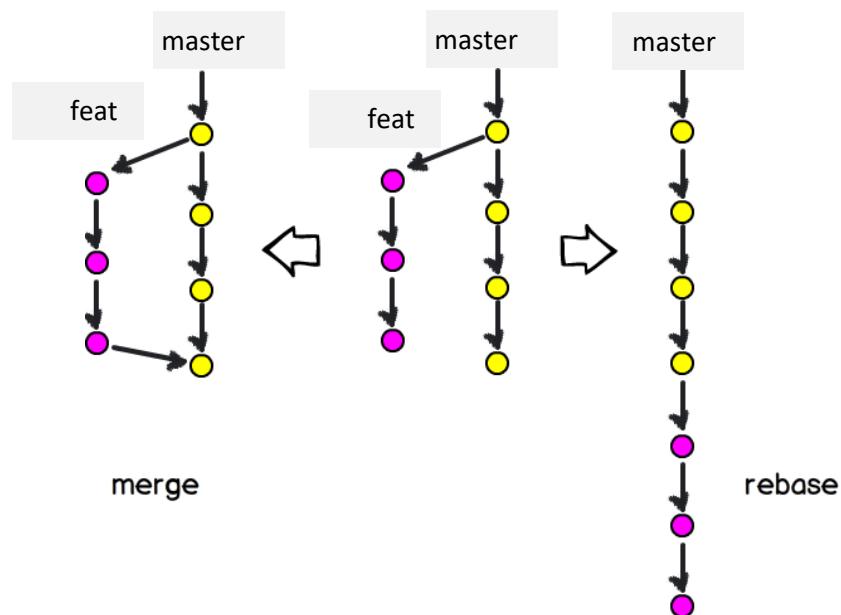


Vậy là xong, lúc này bạn cần xóa nhánh **hotfix** đi vì bạn không còn cần tới nó nữa, nhánh **master** đã trở đến lần commit bạn cần là C4. Bạn đã triển khai xong bản vá lỗi quan trọng và sẵn sàng quay lại với công việc bị gián đoạn trước đó. Bạn sẽ chuyển sang nhánh **iss53** mà bạn đang làm việc trước đó để tiếp tục giải quyết vấn đề #53



Như vậy với Git, bạn hoàn toàn chủ động trong các giai đoạn của dự án mà không làm ảnh hưởng đến tiến độ chung cũng như công việc đang thực hiện của người khác. Ngoài cách dùng merge để tích hợp những thay đổi từ một nhánh vào một nhánh khác bạn còn có thể sử dụng **rebase**.

Rebase cũng là cách để bạn tích hợp các thay đổi từ một nhánh này sang nhánh khác. Nhưng so với merge là gộp nhánh, tức là lấy snapshot mới nhất của mỗi branch rồi kết hợp lại với nhau. Như vậy, mỗi khi bạn merge một feature branch về master branch thì đơn giản là nó sẽ tạo ra một merge commit ở master branch. Trong khi đó nếu chúng ta sử dụng rebase, ví dụ như cần tích hợp feature branch vào nhánh master thì nó sẽ đem tất cả các thay đổi, các commit từ nhánh feature vào nhánh branch hay nói cách khác là nó sẽ sao chép tất cả các thay đổi từ nhánh feature đặt lên trước nhánh master lần lượt theo thứ tự.



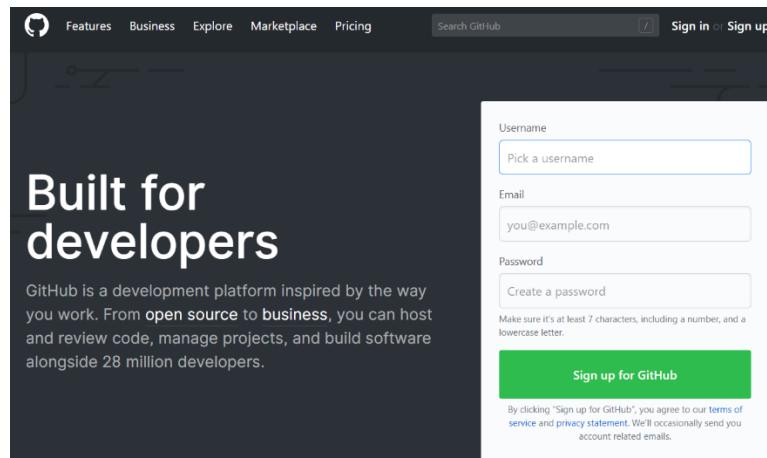
Vậy thì khi nào bạn dùng rebase và khi nào bạn dùng merge? Rõ ràng bạn sẽ thấy lịch sử trên git của merge nhìn rõ hơn, đặc biệt là khi bạn có nhiều nhánh con trong khi rebase thì bạn cho một lịch sử dễ nhìn hơn. Tuy nhiên, merge là cách gộp nhánh an toàn, đặc biệt khi có nhiều người tham gia cùng dự án. Nó sẽ lưu lại thật sự tất cả những gì đã diễn ra trong quá trình thực hiện dự án. Trong khi đó, nếu bạn sử dụng rebase chủ quan, không đúng cách thì có thể mất commit trên nhánh của người khác.

2. GitHub – Dịch vụ lưu trữ mã nguồn online

Bạn nghe nhiều đến GitHub, vậy GitHub là gì?

GitHub là một dịch vụ lưu trữ mã nguồn dựa trên hệ thống quản lý phiên bản phân tán **Git**. GitHub cung cấp cả phiên bản có phí lẫn miễn phí (áp

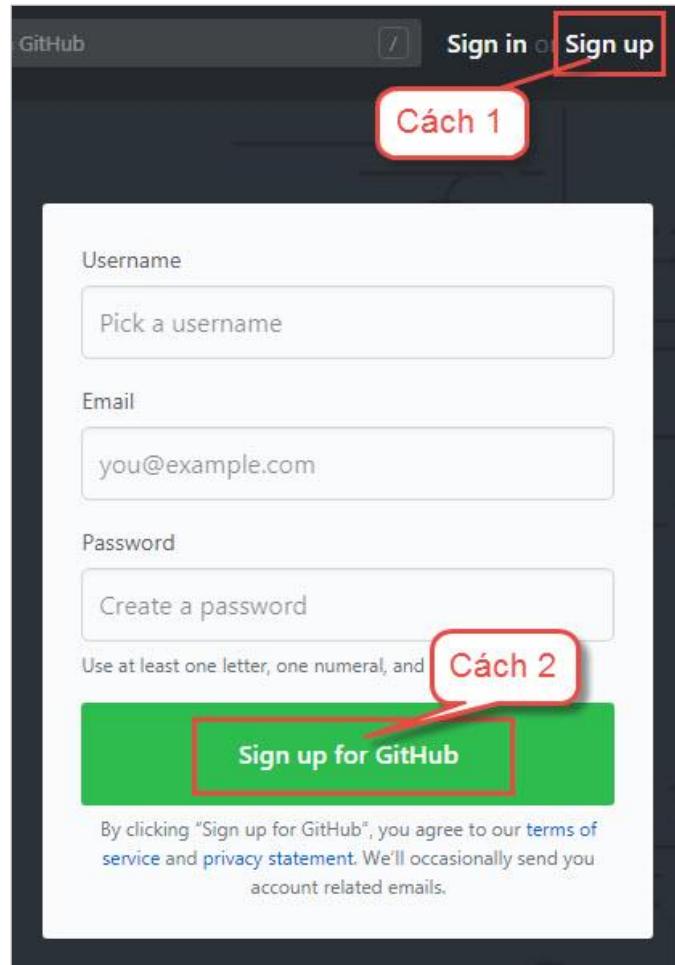
dụng với các dự án mở, mã nguồn public). GitHub hỗ trợ lập trình cộng tác hiệu quả với quy trình phối hợp làm việc khá đơn giản. Các thành viên trong cùng một team khi làm việc với nhau hoàn toàn có thể theo dõi online được các thay đổi trong nhóm ở từng phiên bản làm việc. Không nhất thiết phải ngồi gần nhau, ai cũng có thể biết được những thay đổi đó để rồi gộp phiên bản của thành viên khác vào phiên bản của họ. Do đó, trên GitHub không chỉ có lập trình viên mà còn có rất nhiều tổ chức, trường học, công ty chọn GitHub để lưu trữ, quản lý mã nguồn cho các dự án của mình. Đặc biệt GitHub cũng là nơi giúp bạn tiếp cận được rất nhiều thư viện, kinh nghiệm lập trình giá trị cũng như bạn có thể chia sẻ, đóng góp các source code giá trị của mình cho cộng đồng.



2.1. Tài khoản GitHub

a) Tạo tài khoản

- Để bạn có thể lưu trữ mã nguồn và sử dụng các dịch vụ của GitHub, trước tiên, bạn phải tạo một tài khoản của GitHub. Để tạo tài khoản, trên trang web chính của GitHub, bạn click chọn **Sign up** hoặc **Sign up for GitHub**.



- Tiếp theo, bạn điền thông tin username, email và password. Sau đó, click chọn **Create an account**.

Join GitHub
The best way to design, build, and ship software.

Step 1: Create personal account	Step 2: Choose your plan
------------------------------------	-----------------------------

Create your personal account

Username Điền Username
tutrinhnguyen ✓

This will be your username. You can add the name to the end of it.

Email address Điền Email
nttrinh@csc.hcmus.edu.vn ✓

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password Điền Password
***** ✓

Use at least one lowercase letter, one numeral, and seven characters.

Verify account

By clicking "Create an account" below, you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

Create an account

- Bạn sẽ lựa chọn loại tài khoản. Hiện tại, GitHub cung cấp hai loại tài khoản là: miễn phí cho việc lưu trữ các dự án mở (chọn Unlimited public repertoires for free) và có phí (7 đô/tháng) cho lưu trữ các dự án đóng, có tính bảo mật riêng tư (chọn Unlimited private repertoires for \$7/month) -> **Continue**.

Welcome to GitHub

You've taken your first step into a larger world, @tutrinhnguyen.

Completed Set up a personal account	Step 2: Choose your plan
--	-----------------------------

Choose your personal plan

Unlimited public repositories for free.

Unlimited private repositories for \$7/month. (view in VND)

Don't worry, you can cancel or upgrade at any time.

Help me set up an organization next
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations](#)

Send me updates on GitHub news, offers, and events
Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

- Bạn trả lời một số câu hỏi điều tra của GitHub, rồi bấm **Submit**.

Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @tutrinhnnguyen.

 Completed Set up a personal account	 Step 2: Choose your plan	 Step 3: Tailor your profile
--	---	--

How would you describe your level of programming experience?

Very experienced Somewhat experienced Totally new to programming

What do you plan to use GitHub for? (check all that apply)

Research Development School projects
 Project Management Design Other (please specify)

Which is closest to how you would describe yourself?

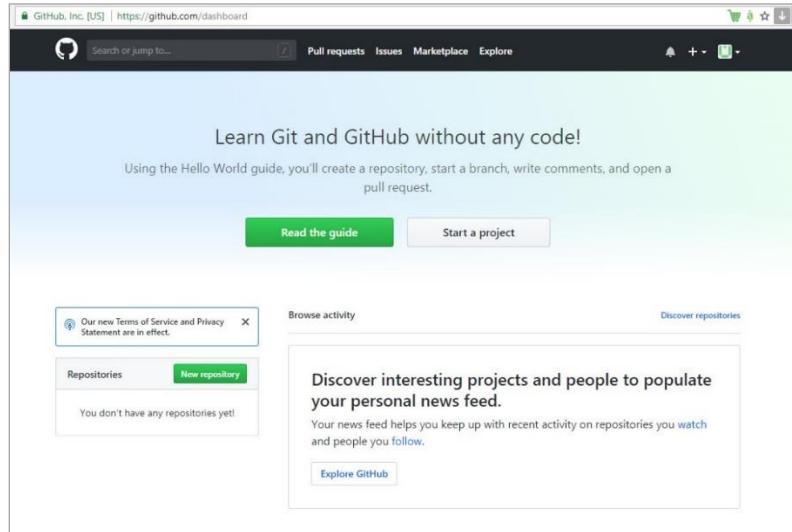
I'm a student I'm a hobbyist I'm a professional
 Other (please specify)

What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

Submit [skip this step](#)

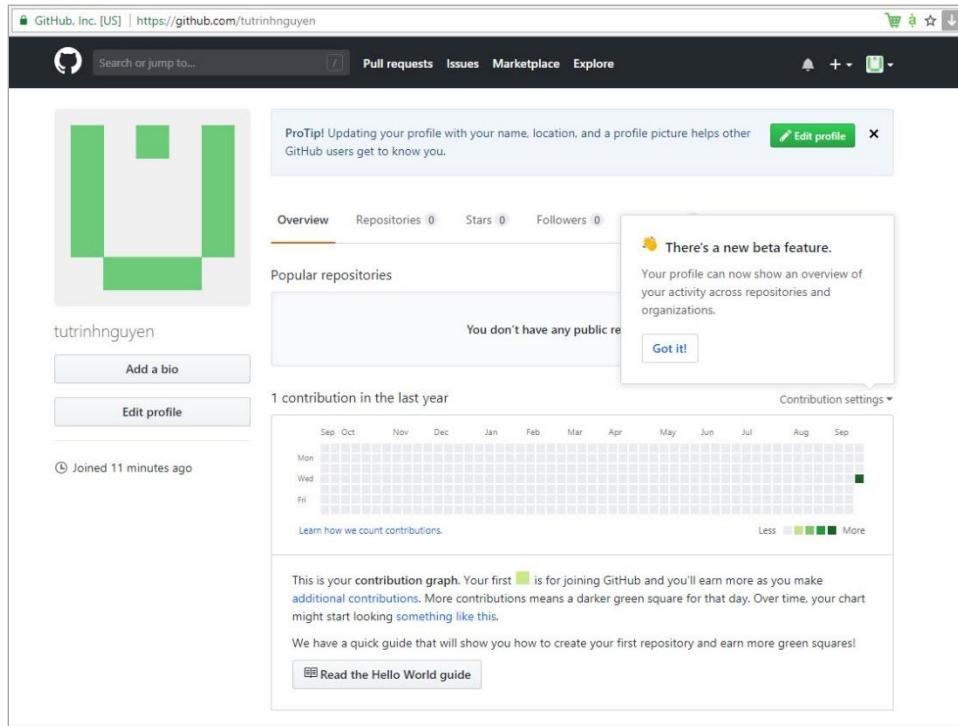
- Bạn nhấp nút Submit. Như vậy là bạn đã tạo xong tài khoản GitHub, lúc này giao diện của GitHub sẽ như sau:



b) Giao diện GitHub

- Khi bạn nhập vào địa chỉ <http://github.com/<username>> là bạn đang đến nơi lưu trữ thông tin và các kho mã nguồn của <username> đã

public. Nếu username cũng chính là bạn, thì bạn sẽ biết được các mã nguồn dự án của mình đang diễn ra như thế nào cũng như xem được các vấn đề, trao đổi, đóng góp từ người khác.



- Khu vực bên trái là ảnh avatar, username, thông tin mô tả ngắn gọn về bạn và thời gian tạo tài khoản. Tất cả những thông tin này bạn có thể cập nhật lại bằng cách click vào nút **Edit profile** ở góc bên phải.
- Phần chính của giao diện gồm 5 tab:
 - **Overview** chứa thông tin tóm tắt về các kho mã nguồn dự án bạn xây dựng hoặc do bạn fork từ một kho khác. Ngay bên dưới bạn sẽ thấy một biểu đồ cho thấy những hoạt động đóng góp của bạn trên các kho dự án trong 1 năm gần đây.
 - **Repositories** giúp bạn quản lý tất cả các kho mã nguồn dự án bạn đang làm việc.
 - **Stars** hiển thị các kho lưu trữ mà bạn có đánh dấu sao.
 - **Followers**: Nếu bạn có những mã nguồn chất lượng chia sẻ thì bạn sẽ thấy những “fan hâm mộ” của mình trong tab này.
 - **Following** hiển thị danh sách những người bạn đang theo dõi họ, bất kỳ khi nào họ có những kho dữ liệu nguồn mới public bạn sẽ được thông báo ngay.

Chip Huyen
chiphuyen

Deep Learning Algorithm @NVIDIA.
Taught "TensorFlow for Deep Learning Research" @Stanford.
Author of 3 bestselling books. Insta: huyenchip19

Follow

Block or report user

@NVIDIA
📍 Mountain View, CA
🔗 <https://huyenchip.com>

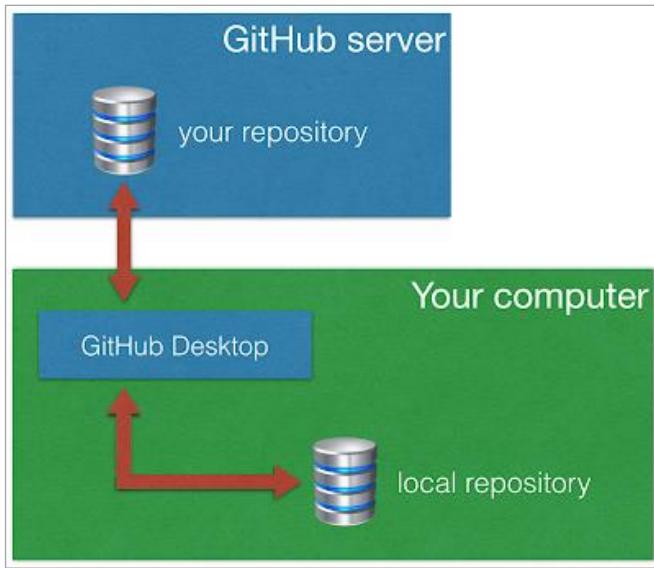
61 contributions in the last year

Learn how we count contributions.

Kho sourcode của Huyen Chip trên GitHub

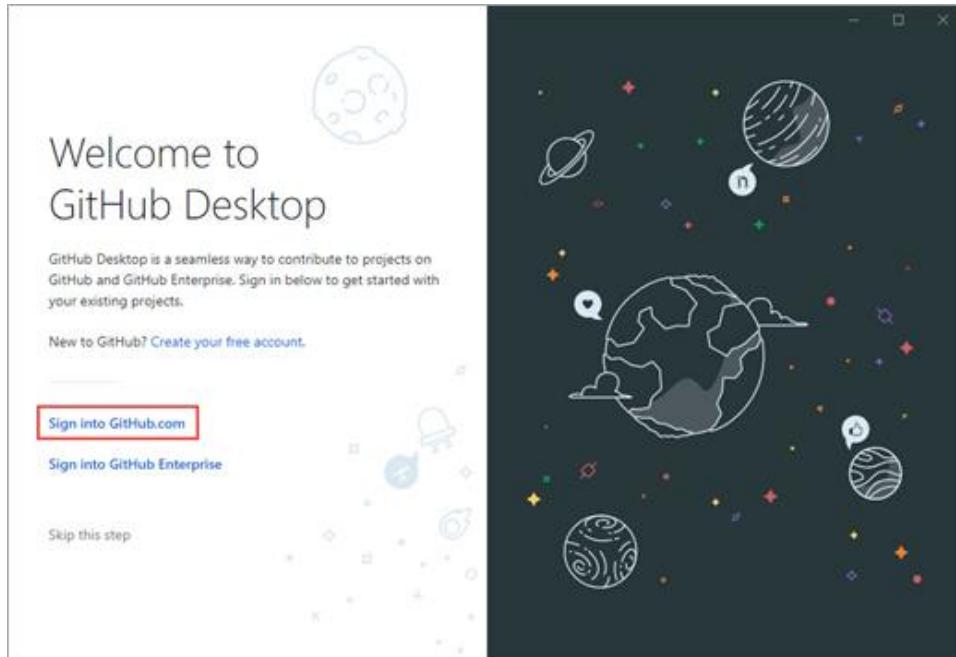
2.2. GitHub Desktop

Sau khi đã tạo tài khoản GitHub, bước tiếp theo là bạn sẽ tạo kho mã nguồn trên máy tính cục bộ của mình. Bạn sẽ thực hiện các công việc chỉnh sửa trên máy cục bộ rồi sau đó mình sẽ cập nhật các thay đổi đó lên kho mã nguồn chung của dự án. Có điều lưu ý là phần lớn các thao tác/hoạt động trong Git chỉ cần yêu cầu các tập tin hay tài nguyên cục bộ. Toàn bộ dự án hoàn toàn nằm trên ổ cứng của bạn nên các thao tác được thực hiện gần như ngay lập tức.

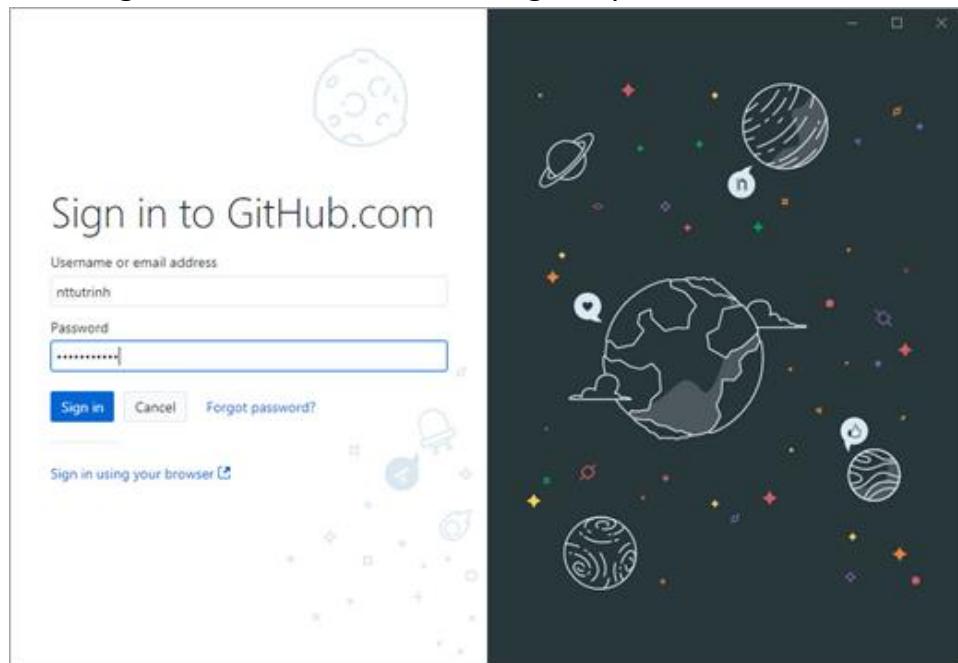


GitHub Desktop cung cấp cho người dùng giao diện đồ họa (GUI) để thực hiện các thao tác cục bộ trên Git. Với giao diện đồ họa trực quan, người dùng có thể dễ dàng thực hiện việc đồng bộ với các dự án trên GitHub về máy của mình thay vì dựa vào các lệnh thô cứng, khó nhớ trên Git.

- Bạn download GitHub Desktop tại <https://desktop.github.com/> và cài đặt trên máy tính cá nhân của mình.
- Sau khi cài đặt xong, bạn chọn **Sign into GitHub.com**, và đăng nhập Username, Password -> click chọn **Sign in** -> Gõ Name, Email -> chọn **Continue** để bắt đầu làm việc trên GitHub Desktop.

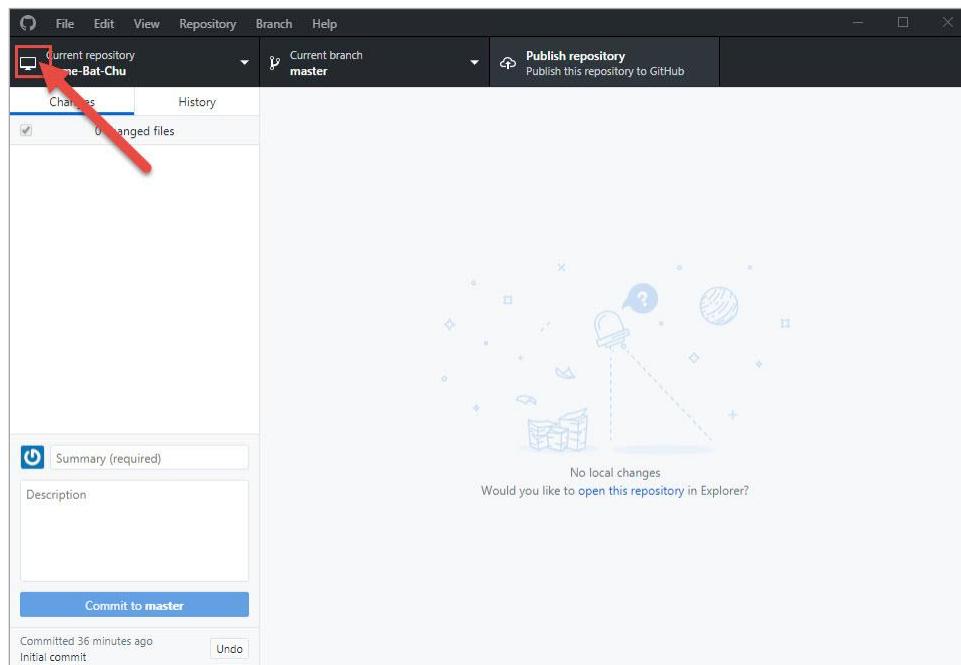


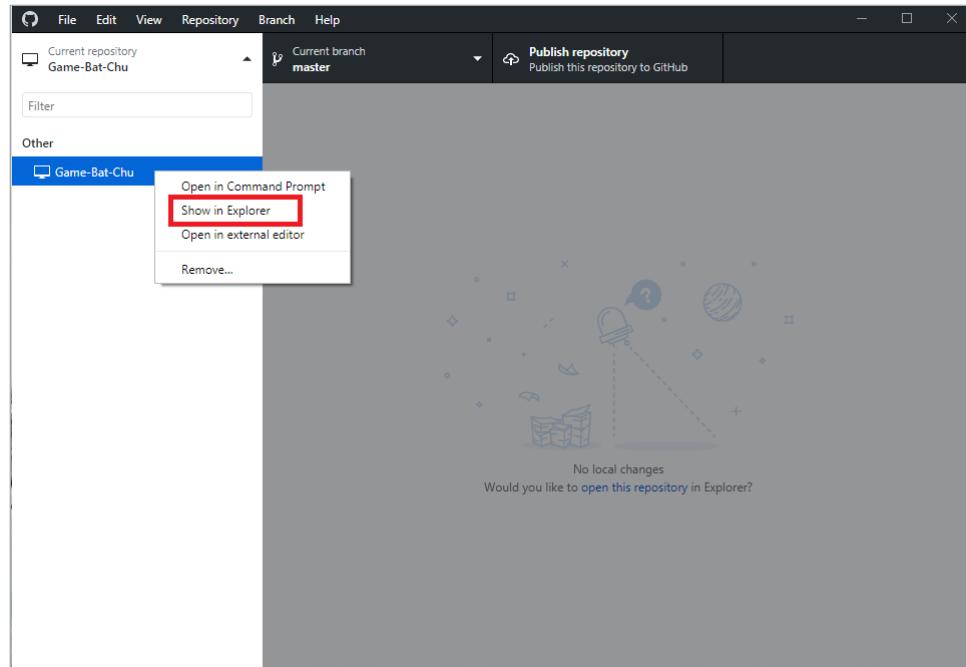
Bạn chọn *Sign into GitHub.com* để đăng nhập vào tài khoản trên GitHub



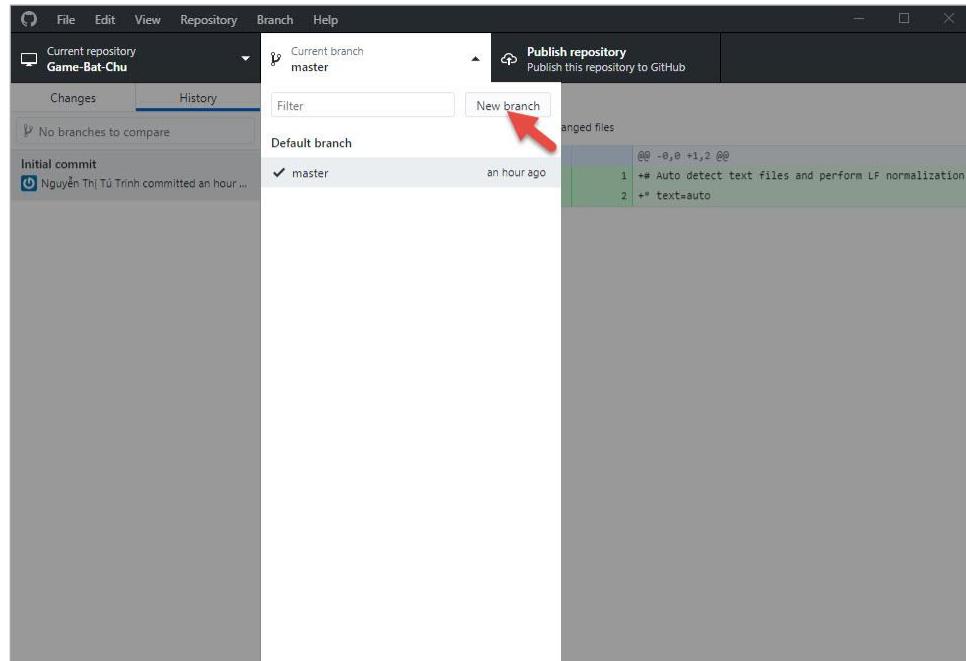
Nhập vào thông tin tài khoản GitHub

- Trên giao diện của GitHub Desktop :
 - **Current repository**: liệt kê tất cả các kho đang nằm trên máy của bạn. Bạn có thể bấm phải chuột vào tên các kho, rồi bấm **Show in Explorer** để xem qua thư mục chứa nội dung của kho sử dụng Windows Explorer.



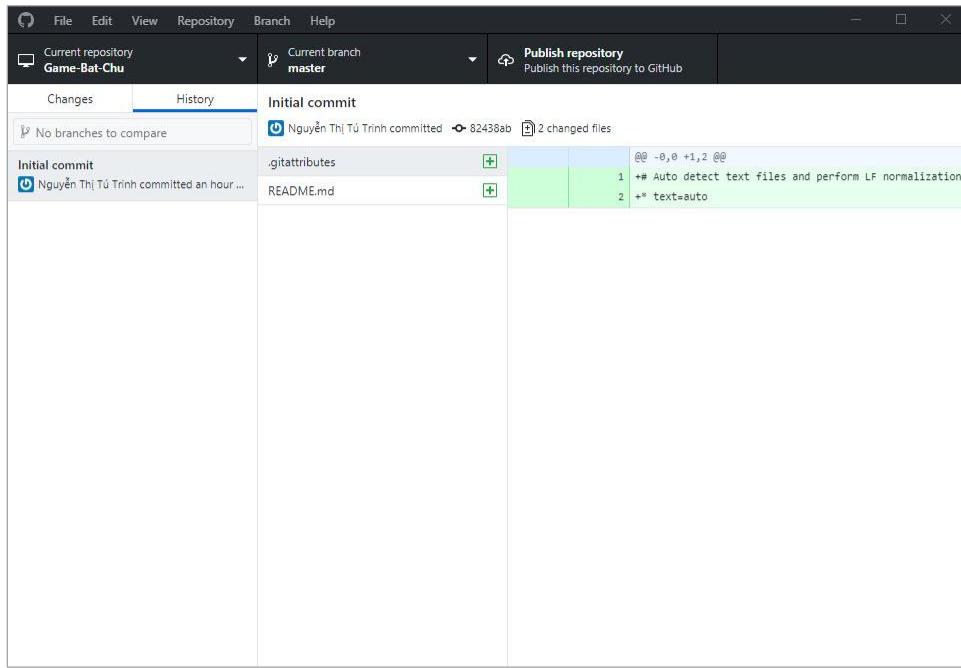


- **Current branch:** cho phép bạn chọn nhánh hiện hành, nhánh bạn làm việc. Nhánh master là nhánh gốc của bất kỳ một kho lưu trữ nào, để tạo nhánh mới, bạn click chọn **New branch**.



- **Publish repository:** Cập nhật kho lưu trữ cục bộ lên tài khoản trên trang web GitHub.

- **Changes**: thể hiện những thay đổi trên các tập tin, thư mục trong kho.
- **History** ghi nhận lịch sử các lần lưu trạng thái (commit). Trong đó bạn sẽ biết được không những biết được các tập tin/thư mục nào được thay đổi mà còn biết được chi tiết là nội dung thay đổi như thế nào, để bạn có thể quay lại trạng thái mà bạn muốn.

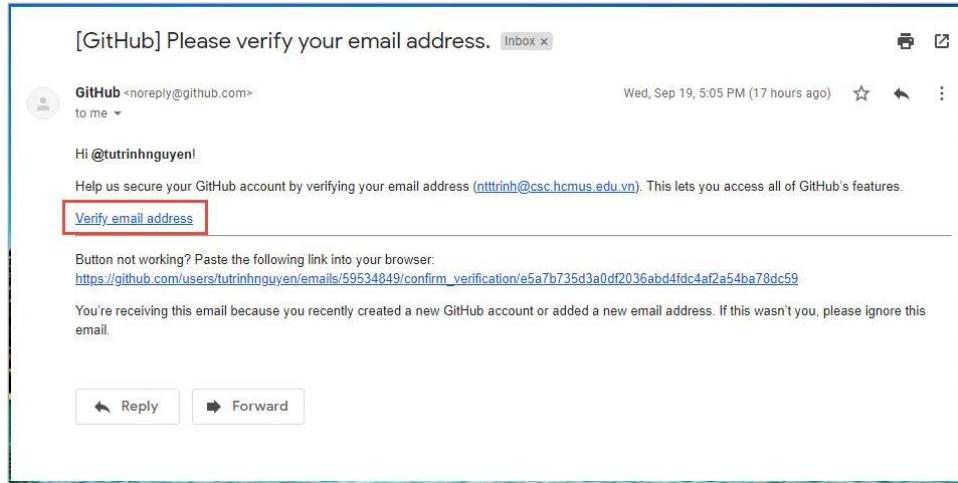


Lưu ý: Khi bạn clone một kho (repository) với GitHub thì kho đó sẽ được sao chép một bản về trên máy của bạn, độc lập với bản ở trên GitHub. Do đó, mọi thao tác sửa đổi, lưu trạng thái (commit) sẽ chỉ được thực hiện ở máy của bạn. Chỉ khi nào bạn đồng bộ với tài khoản trên GitHub thì những sửa đổi và lần lưu trạng thái đó mới được chuyển lên GitHub.

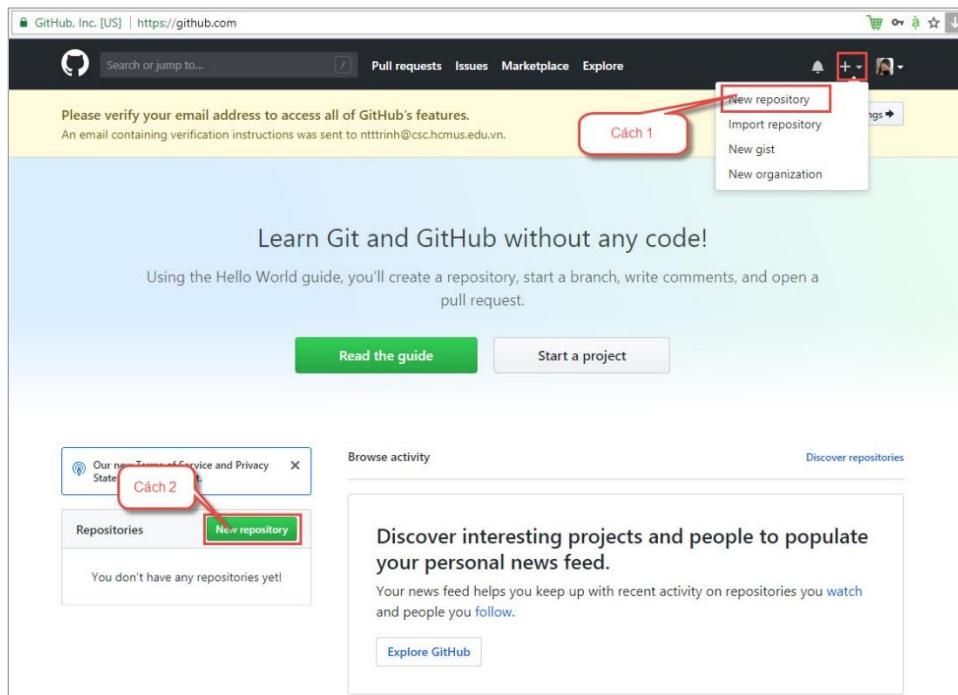
3. Sử dụng GitHub

3.1. Tạo kho (repository) lưu trữ project

- GitHub sẽ gửi cho bạn một email vào địa chỉ mail của bạn đã cung cấp cho GitHub để xác thực. Trong email đó, bạn bấm đường link **Verify email address** để xác thực. Lúc này, GitHub sẽ gửi email chúc mừng và cho phép bạn tạo repository đầu tiên trên GitHub.



- Repository thường được dùng để tổ chức một dự án (project), nó có thể chứa các thư mục, tập tin, ảnh, video, bảng biểu và dữ liệu - tất cả những gì mà dự án của bạn cần. Khi tạo một kho, bạn nên thêm một tập tin README hoặc một tập tin thông tin giới thiệu về dự án của bạn, hoặc một tập tin chứa thông tin license.
- Trên trang web GitHub, sau khi đăng nhập, bạn nhấp chuột vào nút dấu + (bên cạnh avatar của bạn ở góc trên bên phải) và chọn mục **New repository**. Hoặc bạn click vào **New repository** ở bên trái của màn hình GitHub.



- Giao diện trang web để tạo kho lưu trữ sẽ xuất hiện ra như sau:

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner / Repository name

Great repository names are short and memorable. Need inspiration? How about [psychic-sniffle](#).

Description (optional)

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 let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

- Bạn điền tên của kho lưu trữ vào ô **Repository name**, điền mô tả của dự án vào ô **Description**, chọn **Public** cho loại kho rồi click chọn **Create repository** để tạo. Khi đó, giao diện của trang web sẽ chuyển vào trang nội dung của kho mới được tạo. Nếu ở bước trên bạn có chọn mục *Initialize this repository with a README*, nên GitHub sẽ tự động tạo tập tin **README.md** với nội dung được lấy từ ô **Description**.

[tutrinhnghuyen / 1Quan-Ly-Danh-Ba](#)

Ứng dụng Quản lý danh bạ

Manage topics

Branch: master

tutrinhnghuyen Initial commit Latest commit 56e0838 just now

README.md Initial commit just now

README.md

1Quan-Ly-Danh-Ba

Ứng dụng Quản lý danh bạ

- Giao diện trang web của một kho lưu trữ sẽ bao gồm 7 tab:
 - Tab **Code**: bạn có thể truy cập vào tất cả các tập tin được lưu trữ trong kho. Bạn có thể xem nội dung của từng tập tin, các lần lưu trạng thái (commit), so sánh mã nguồn giữa các lần chuyển giao với nhau, xem và chuyển đổi giữa các nhánh (branch), thêm tập tin vào kho, tạo một yêu cầu gộp (pull requests) mới và clone về GitHub Desktop, ...
 - Tab **Issues**: cho phép bạn quản lý các vấn đề (issue), nhãn (label) và điểm mốc (milestone) của dự án do bạn hoặc thành viên trong nhóm tạo ra.
 - Tab **Pull requests**: cho phép bạn quản lý các yêu cầu gộp (pull request) của dự án.
 - Tab **Projects**: là công cụ cho phép quản lý dự án sử dụng các bảng dự án.

The screenshot shows a project management interface with three columns: Backlog, In Progress, and Ready to deploy. The In Progress column contains a card for 'Share farms' soil moisture data' which is currently being edited by Eddie. A tooltip for this card indicates 'internationalization'.

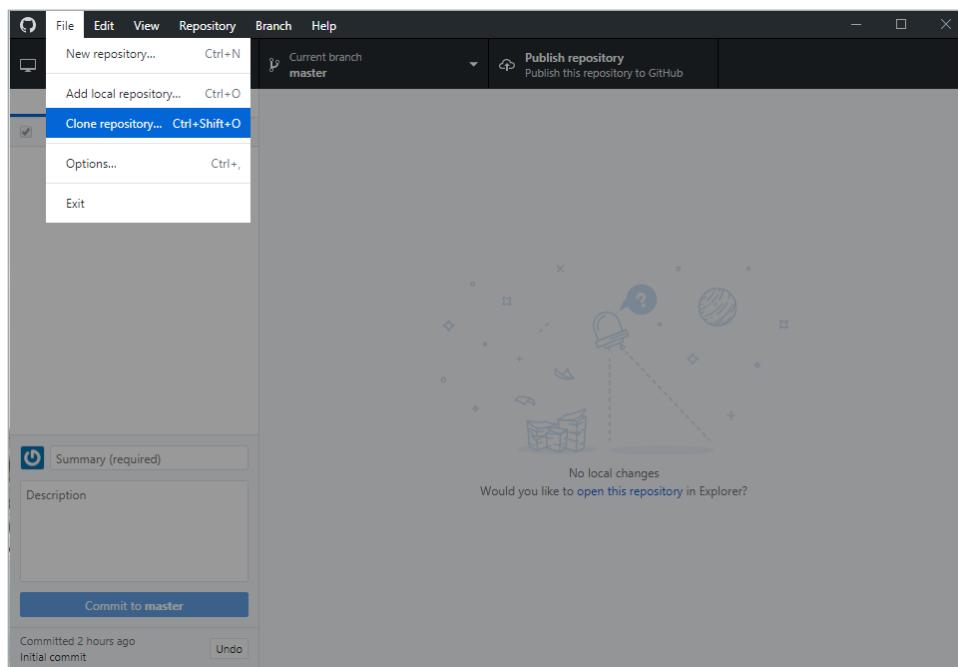
Column	Card Description
Backlog	Collect satellite data and deliver to farmers (Added by Sam)
In Progress	Crawl tractor engine data (John Deere) (Added by Melinda), Performance updates for data script (Added by Sam), Share farms' soil moisture data (internationalization, Eddie), User testing with farmers in China (Added by Eddie)
Ready to deploy	[Data] Soil data collection scripts (Added by Vijay)

- Tab **Wiki**: cho phép bạn bố trí lộ trình dự án của bạn, tạo mô tả dự án (document) cho dự án của bạn.

- Tab **Insights**: cho phép bạn xem các thống kê về tình hình cập nhật và sửa đổi của dự án.
- Tab **Settings**: cho phép bạn thiết lập các tuỳ chọn cho dự án.

3.2. Clone kho lưu trữ từ GitHub về GitHub Desktop

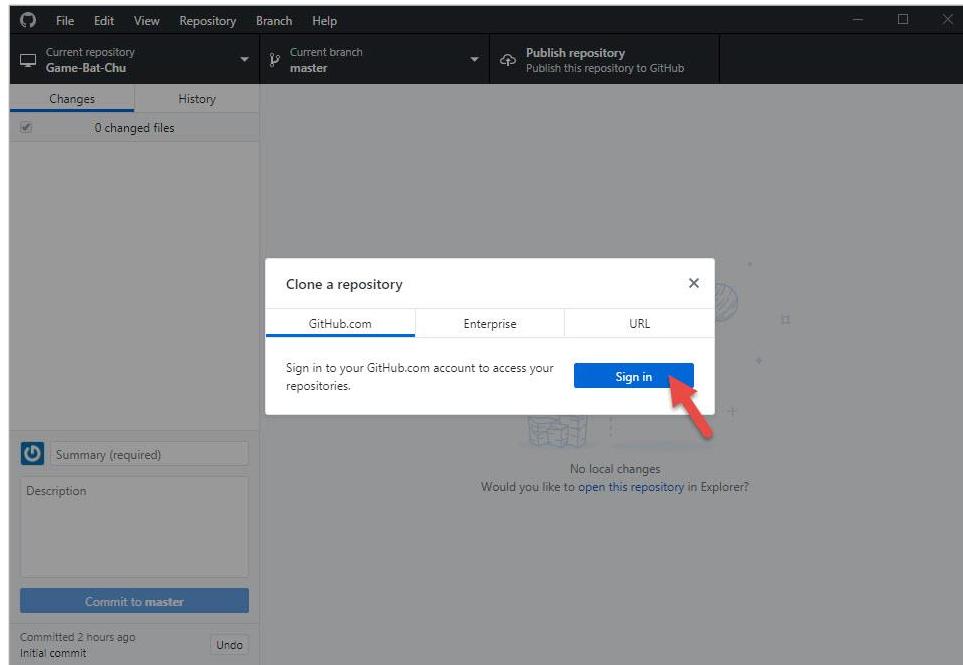
Bạn có thể clone repository của bạn hoặc của lập trình viên khác để tham khảo, học hỏi thêm mã nguồn. Để clone kho lưu trữ từ trang web GitHub, bạn làm như sau:



Chức năng Clone repository

Nếu bạn clone kho dữ liệu từ tài khoản của bạn trên GitHub.com thì bạn chọn **GitHub.com** rồi đăng nhập vào tài khoản GitHub rồi chọn kho bạn cần clone.

Nếu bạn muốn clone từ tài khoản bất kỳ nào khác thì bạn chọn **URL** rồi nhập vào thông tin đường dẫn URL đầy đủ của kho và đường dẫn bạn sẽ lưu tại máy cục bộ của mình.



3.3. Commit và đồng bộ kho lưu trữ giữa GitHub Desktop và GitHub.com

Để commit lưu lại các thay đổi, trong GitHub Desktop, bạn click vào tab **Changes** để xem những thay đổi trong kho lưu trữ (ví dụ: kho lưu trữ ở đây là **Quan-Ly-Danh-Ba**) được GitHub Desktop ghi nhận như thế nào:

- Bên trái, bạn sẽ thấy danh sách tất cả các tập tin có sự thay đổi (tạo mới hay sửa nội dung). Các dòng màu xanh với dấu thể hiện là các tập tin/thư mục mới được thêm vào, màu đỏ với dấu thể hiện tập tin/thư mục bị xoá, màu cam với dấu thể hiện tập tin/thư mục có thay đổi, chỉnh sửa.
- Còn ở cửa sổ bên phải là nội dung một tập tin/thư mục đang được chọn xem.

The screenshot shows a GitHub commit interface. The top bar includes 'File', 'Edit', 'View', 'Repository', 'Branch', and 'Help'. The 'Repository' dropdown is set to 'Quan-ly-Danh-Ba'. The 'Branch' dropdown is set to 'master'. A 'Publish repository' button is visible. The main area shows a list of 184 changed files, with several files selected. On the right, a diff view for the 'Contact.java' file is displayed, showing the following code:

```

@@ -0,0 +1,130 @@
1+package tiendung.com.quanlydanhba.obj;
2+
3+import tiendung.com.quanlydanhba.db.ContactDatabaseOpenHelper;
4+import android.database.Cursor;
5+
6+public class Contact {
7+
8+    private long id;
9+    private String name;
10+   private String phone;
11+   private String mail;
12+   private String birthday;
13+   private String regionName;
14+   private int regionId;
15+   private String groupName;
16+   private int[] groupId;
17+
18+   public long getId() {
19+       return id;
20+   }
21+   public void setId(long id) {
22+       this.id = id;
23+   }
24+   public String getName() {
25+       return name;
26+   }
27+   public void setName(String name) {

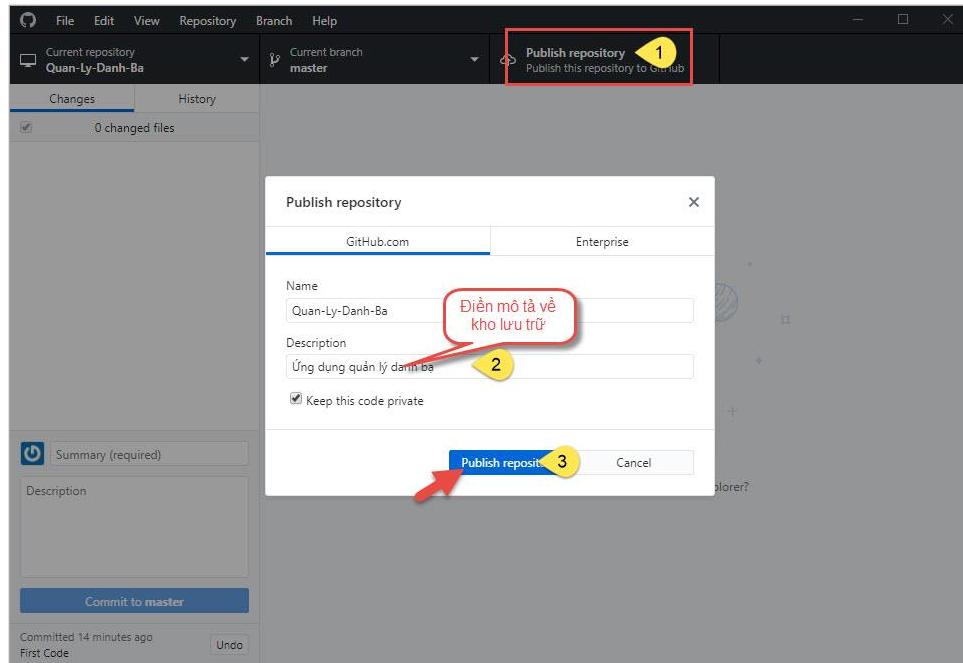
```

Below the diff view, there are buttons for 'Commit to master', 'Commit message', and 'Commit message (optional)'. The status bar at the bottom indicates 'Committed 18 hours ago' and 'Initial commit'.

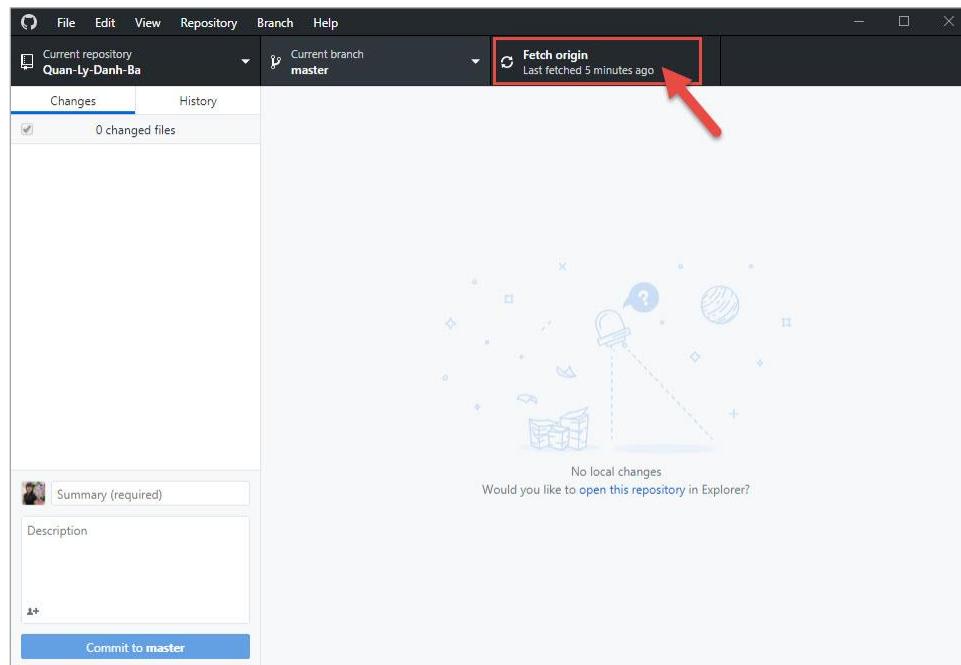
Mặc định hệ thống sẽ tự động đánh dấu tất cả các thay đổi để commit, nhưng bạn có thể chọn/bỏ chọn các tập tin/thư mục nào cần được commit. Sau đó, bạn có thể điền mô tả ngắn cho những thay đổi vào ô **Summary**, điền mô tả chi tiết của những thay đổi vào ô **Description**, rồi click chọn **Commit to master**.

Vậy là bạn đã commit các thay đổi vào repository cục bộ. Để đồng bộ kho cục bộ lên kho lưu trữ trên GitHub, bạn có thể click chọn **Publish repository** để đẩy kho lưu trữ lên GitHub. Trong khung mới hiện ra, bạn điền mô tả cho dự án vào ô **Description** và click chọn **Publish repository**.

Lưu ý: Sau mỗi lần chỉnh sửa quan trọng, bạn nên commit để dễ dàng phục hồi lại bản cũ khi cần.



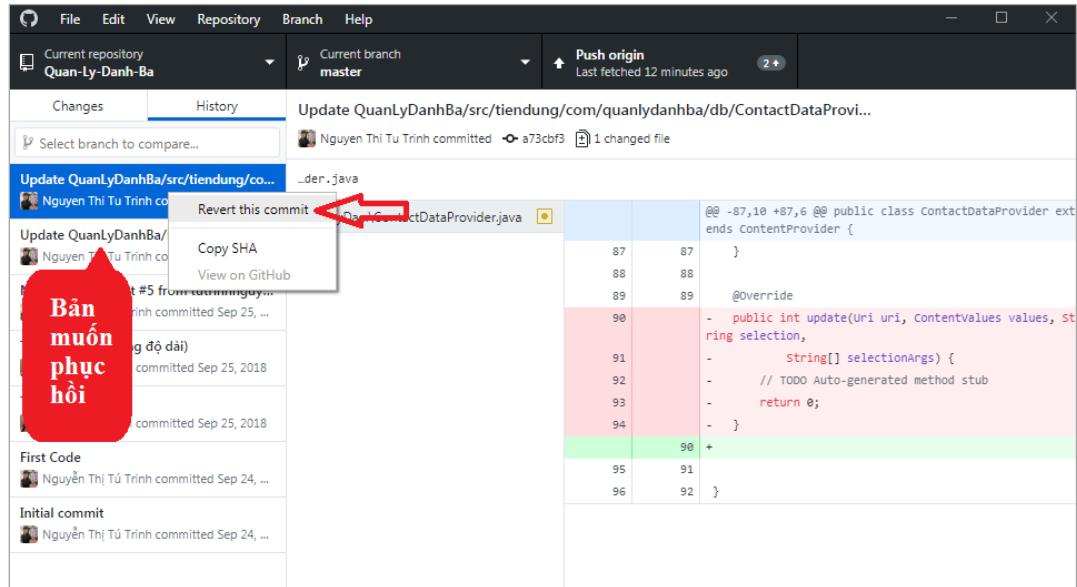
Ngược lại, nếu bạn muốn cập nhật các thay đổi từ kho GitHub.com về kho cục bộ trên máy tính của bạn thì bạn chọn **Fetch origin** để đồng bộ với mã nguồn mới (nếu có) trên trang web GitHub.



3.4. Khôi phục lại phiên bản cũ

Git giúp bạn lưu lại các phiên bản mã nguồn mà bạn có thể dễ dàng khôi phục lại tại thời điểm bất kỳ nào đó. Rất đơn giản, trong **History** là danh sách các lần

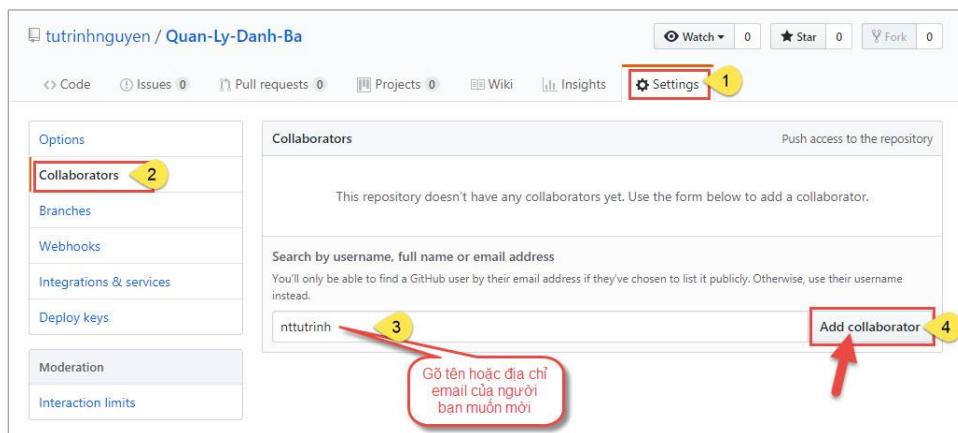
bạn đã commit, bạn chọn bản commit cần phục hồi rồi click phải và chọn **Revert this commit** và bạn sẽ có lại bản cũ như ý bạn mong muốn. Ví dụ hình bên dưới, bạn vừa chỉnh sửa code trong file ContactDataProvider.java và đã commit, nhưng bạn phát hiện mình code chưa đúng nên bạn muốn lấy lại bản cũ trước đó rồi chỉnh lại, bạn thực hiện:



4. Phối hợp thực hiện project

4.1. Mời thành viên tham gia project

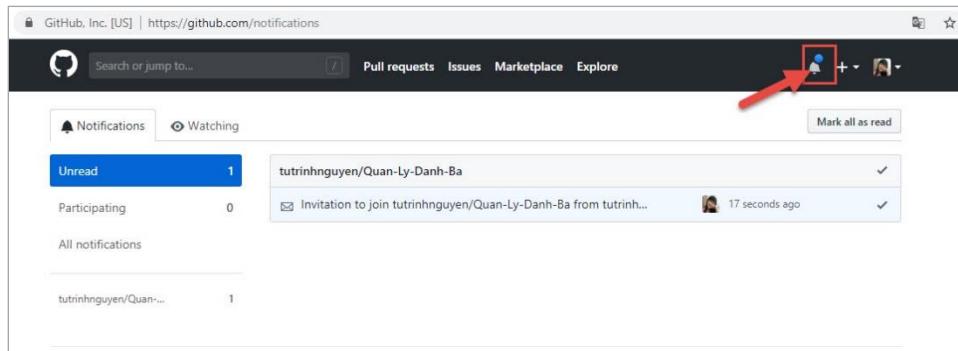
Để mời thành viên khác (đã có tài khoản của GitHub), trong trang web của kho, bạn click vào tab **Settings**, chọn **Collaborator**, gõ tên hoặc địa chỉ email của người mà bạn muốn mời vào ô **Search by username, fullname or email address**.



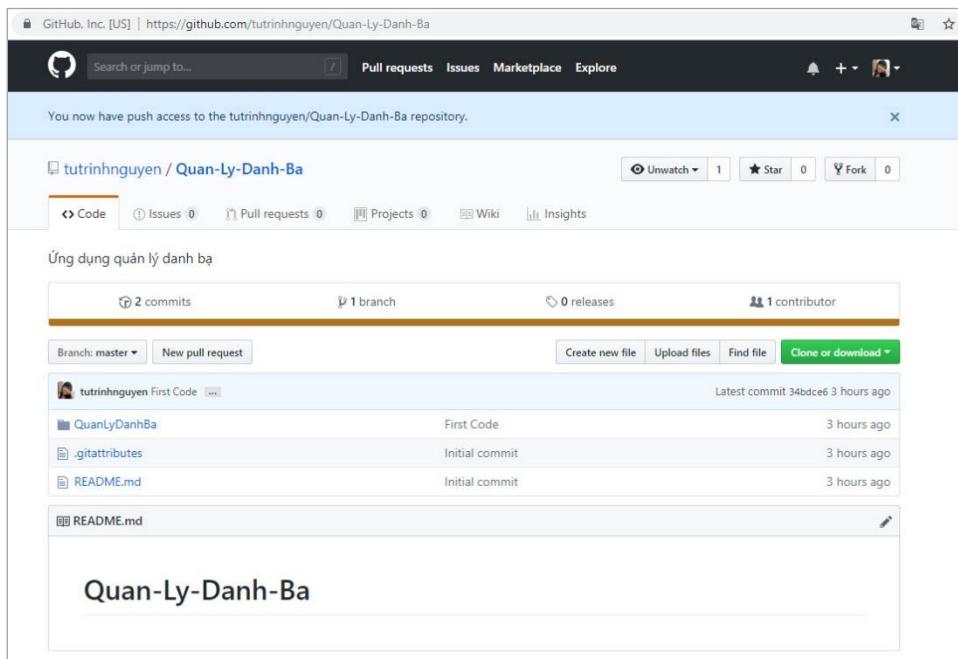
Sau khi chọn tài khoản người muốn mời xong, bạn click chọn **Add collaborator**.

Chấp nhận lời mời và sao chép (clone) kho về máy để xem mã nguồn:

- Giả sử: A là người mời, B là người được mời.
- Lúc này, B sẽ nhận được một thư mời. B vào trang web GitHub và click vào **Notifications** để xem thông báo, thư mời.



- B mở thư ra xem và **Accept Invitation** để chấp nhận lời mời. Sau đó trong giao diện của web GitHub, B chọn **Clone or download** để sao chép mã nguồn của A về máy của mình để làm việc.



4.2. Thảo luận - Trao đổi

Như vậy, B đã có thể xem được mã nguồn của A và có thể viết thêm vào các lần commit lưu trạng thái các nhận xét, trao đổi, thắc mắc hoặc chỉ ra những tính năng cần làm....

4 comments on commit 34bdce6

tutrinhnghuyen replied 7 minutes ago

Mã nguồn tương đối tốt. Tuy nhiên, cần bổ sung thêm một số kỹ thuật để mã nguồn ngắn gọn hơn:

1. Thuộc tính
2. Đọc và lưu dữ liệu ra tập tin

nttutrinh replied 4 minutes ago

@tutrinhnghuyen ý kiến hay đó, mình đã nghiên cứu thêm độ dài các ký tự trong thuộc tính.

tutrinhnghuyen replied 2 minutes ago

@nttutrinh Bạn hãy tạo ra Project, quản lý các issue và gán người xử lý đi.

nttutrinh replied 2 minutes ago

@tutrinhnghuyen okie bạn

Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Styling with Markdown is supported

Comment on this commit

Như vậy, A và B xác định sẽ cùng nhau phát triển thêm 2 tính năng mới:

- Thêm thuộc tính, có kiểm tra độ dài chuỗi ký tự được nhập vào
- Đọc và ghi dữ liệu danh bạ xuống tập tin

Để xem các thông báo của các trao đổi này, bạn phải chuyển ra trang chủ của GitHub

GitHub, Inc. [US] | <https://github.com>

Search or jump to...

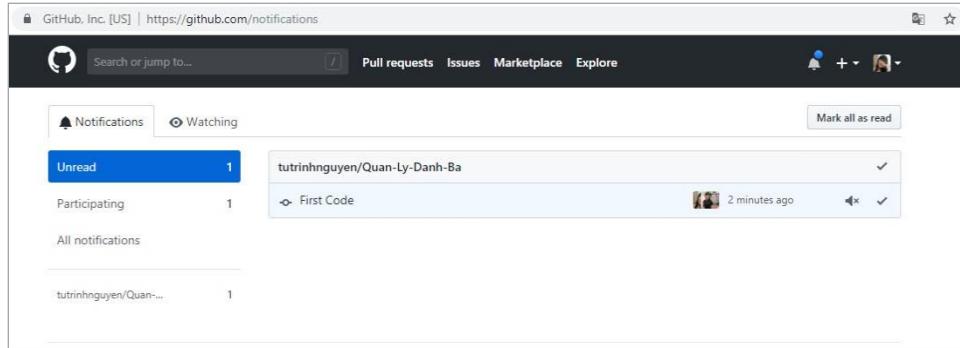
Pull requests Issues Marketplace Explore

Workshops at GitHub Universe Register to attend a workshop in San Francisco on October 15

Our new Terms of Service and Privacy Statement are in effect.

Introducing the dashboard Beta

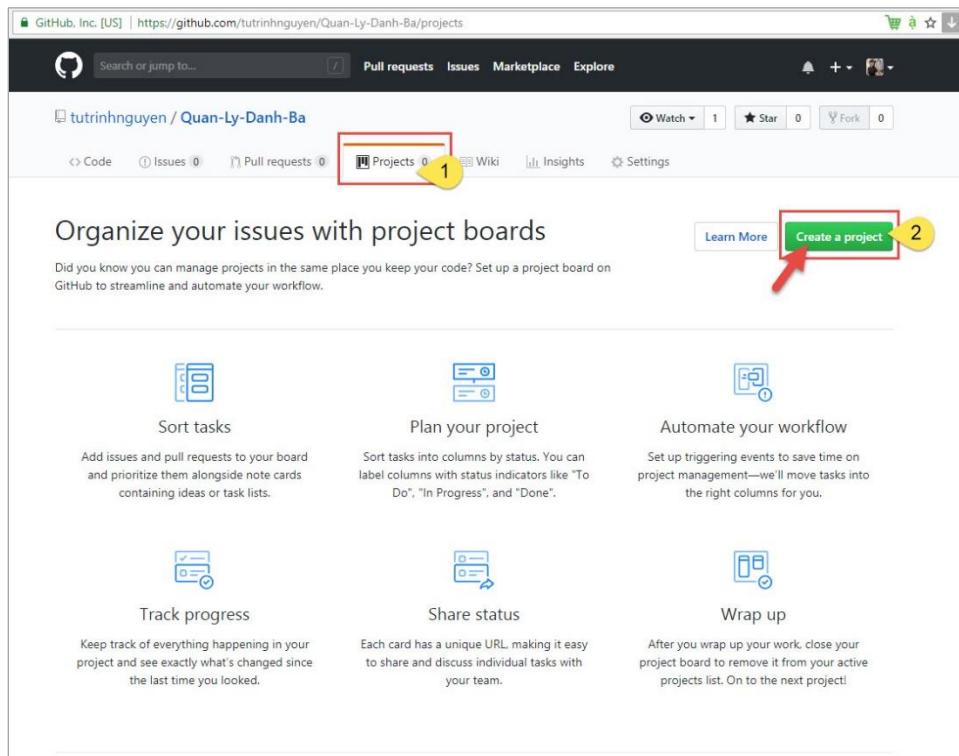
Opt-in No thanks



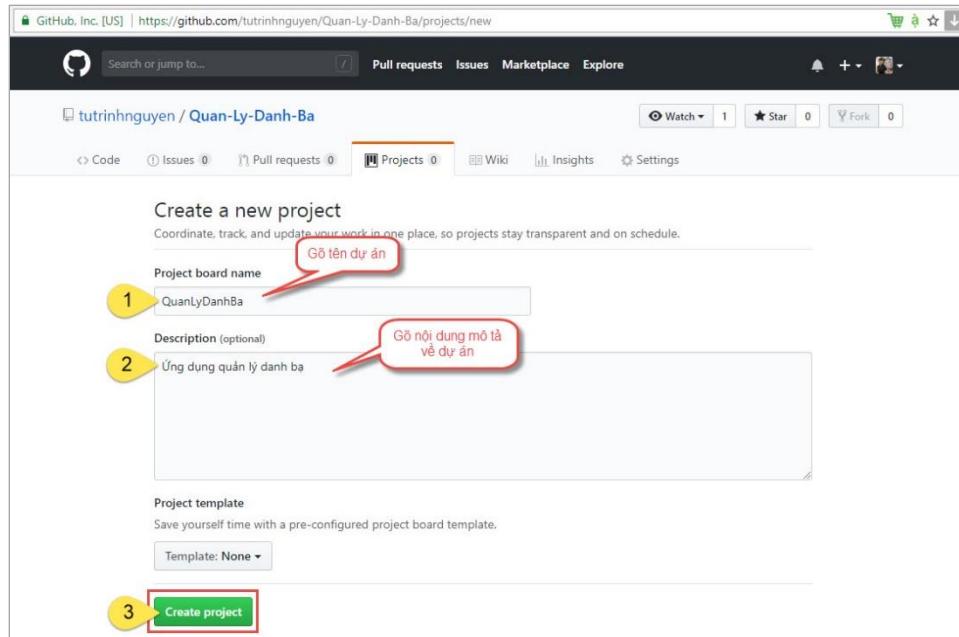
4.3. Quản lý dự án với tab Projects

Năm 2016, GitHub bổ sung thêm chức năng Projects cho phép người dùng GitHub có thể quản lý tiến độ của dự án được thuận tiện và dễ dàng hơn.

Để theo dõi tiến độ dự án, A vào tab **Projects** trong kho Quan-Ly-Danh-Ba, click chọn **Create a project** để bắt đầu tạo một trang quản lý dự án mới.

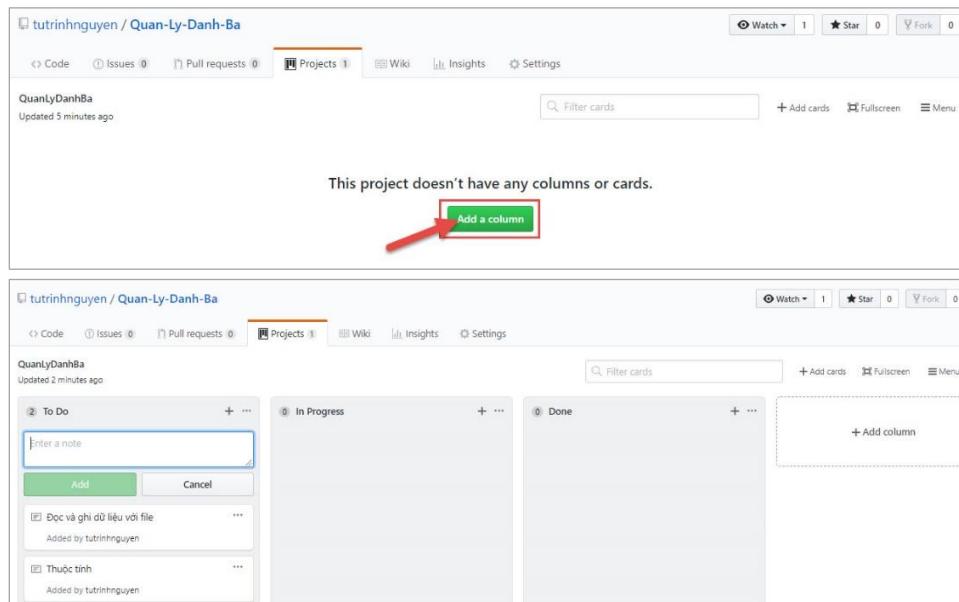


GitHub sẽ hiển thị form cho phép bạn điền tên (**Project board name**) và mô tả (**Description**) cho dự án. Rồi bạn click chọn **Create project** để tạo trang quản lý.



Lúc này, trong project mới tạo, bạn có thể tạo các cột tương ứng với các nhiệm vụ (task) cần thực hiện và cập nhật tiến độ của các task đó. Mỗi cột có thể chứa nhiều nhiệm vụ. Bạn có thể kéo thả các nhiệm vụ qua lại giữa các cột để chuyển trạng thái các nhiệm vụ cho phù hợp với tình hình thực hiện dự án.

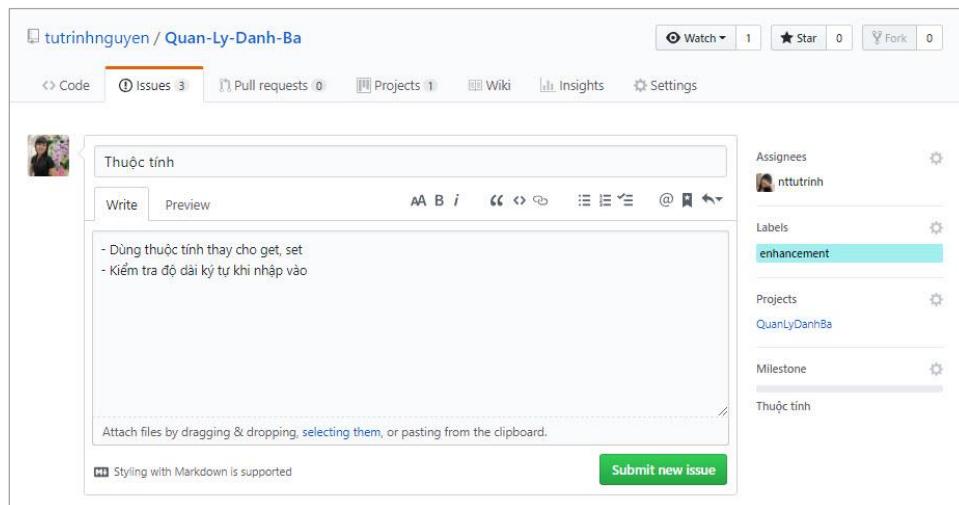
Ví dụ: với dự án QuanLyDanhBa hiện tại, A sẽ tạo 3 cột To Do, In Progress và Done (tức là: cần làm, đang làm và đã xong) và có 2 nhiệm vụ đang cần làm ở cột To Do.



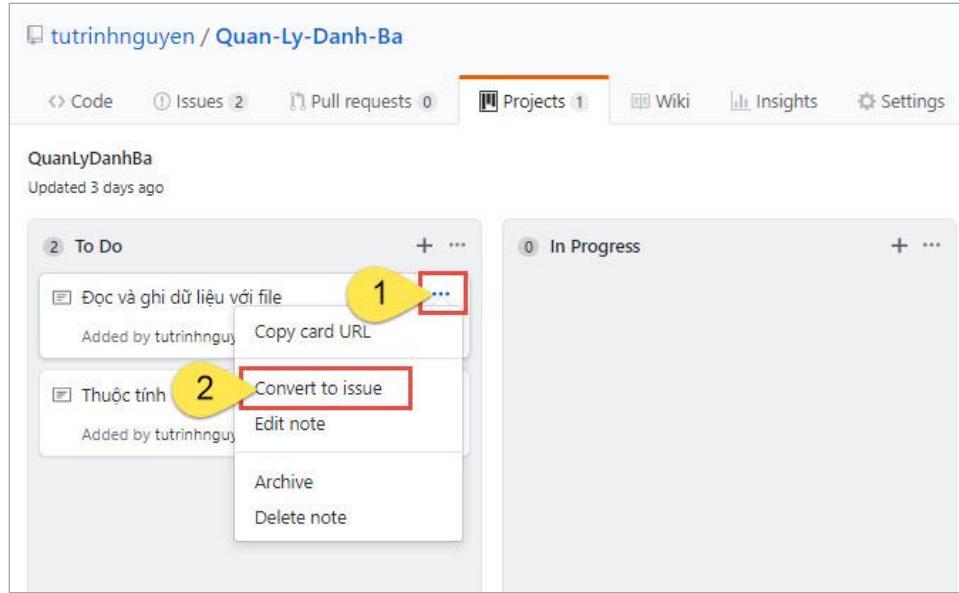
4.4. Quản lý các vấn đề (Issue)

Có thể xem quản lý issue cũng là cách quản lý các vấn đề liên quan đến dự án. Issue cho phép bạn ghi nhận toàn bộ các các đề xuất, cải tiến, các lỗi phát sinh, ... trong quá trình thực hiện dự án. Tất cả issue đều được chia sẻ và thảo luận chung trong nhóm thực hiện.

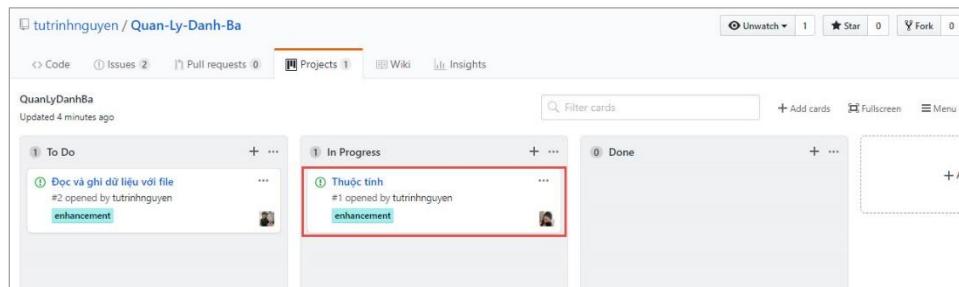
Sau khi tạo xong danh sách các công việc cần làm, A chuyển sang tab **Issues** để tạo các vấn đề (issue)



Ngoài ra, trong tab **Projects** bạn có thể tạo một vấn đề từ các nhiệm vụ trong tab **Projects**. Để làm việc đó, bạn click vào dấu (...) bên phải các nhiệm vụ, rồi chọn **Convert to issue**.



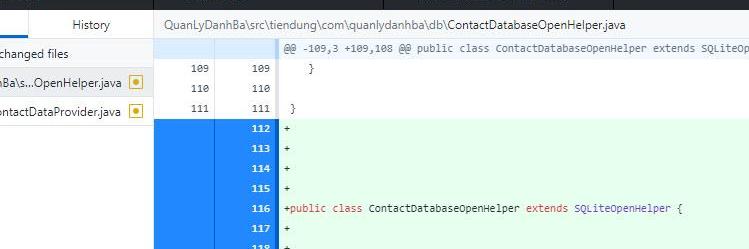
Với B, mỗi khi có một vấn đề (issue) mới được tạo ra hay thay đổi, thì GitHub đều thông báo cho B biết. B có thể xem, bình luận và có thể thao tác trên dự án QuanLyDanhBa trong tab Projects. Ví dụ như B có thể kéo nhiệm vụ Thuộc tính sang cột **In Progress** trước khi bắt đầu giải quyết nhiệm vụ.



4.5. Lưu trạng thái (commit) và yêu cầu gộp (pull request)

Sau khi A chỉnh sửa hoàn thành chỉnh sửa mã nguồn cho tính năng A đảm nhận, A sẽ sử dụng GitHub Desktop để thực hiện commit lưu trạng thái và yêu cầu gộp vào nhánh **Nhanh-file**.

GitHub Desktop lúc này ghi nhận 2 tập tin có sự thay đổi:



Current repository: Quan-ly-Danh-Ba

Current branch: Nhanh-file

Publish branch: Publish this branch to GitHub

Changes: 2 changed files

QuanLyDanhBa\src\tiendung\com\quanlydanhba\db>ContactDatabaseOpenHelper.java

```
@@ -109,3 +109,108 @@ public class ContactDatabaseOpenHelper extends SQLiteOpenHelper {  
    109     }  
    110     }  
    111     }  
    112     +  
    113     +  
    114     +  
    115     +  
    116     +public class ContactDatabaseOpenHelper extends SQLiteOpenHelper {  
    117     +  
    118     +  
    119     +    public static final String TABLE_CONTACT = "DOI_TAC";  
    120     +  
    121     +    public static final String COL_CONTACT_ID = "_id";  
    122     +    public static final String COL_CONTACT_NAME = "Ho_ten";  
    123     +    public static final String COL_CONTACT_PHONE = "Dien_thoai";  
    124     +    public static final String COL_CONTACT_BIRTH_DAY = "Ngay_sinh";  
    125     +    public static final String COL_CONTACT_MAIL = "Mail";  
    126     +    public static final String COL_CONTACT_REGION_ID = "ID_Khu_vuc";  
    127     +    public static final String COL_CONTACT_REGION_NAME = "Ten_khu_vuc";  
    128     +    public static final String COL_CONTACT_GROUP_IDS = "Chuoi_Danh_sach_ID_Nhom";  
    129     +    public static final String COL_CONTACT_GROUP_NAMES = "Chuoi_Danh_sach_Ten_Nhom";  
    130     +  
    131     +  
    132     +    private static String DB_PATH = "/data/data/hts1.com.quanlydanhba/databases/";  
    133     +    private static final String DB_NAME = "contact.sqlite";  
    134     +  
    135     +    public ContactDatabaseOpenHelper(Context context) {
```

Summary (required)

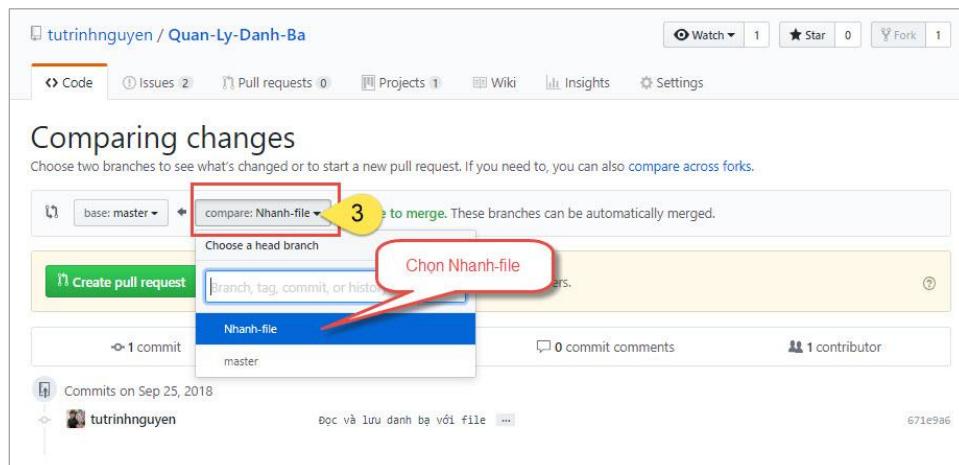
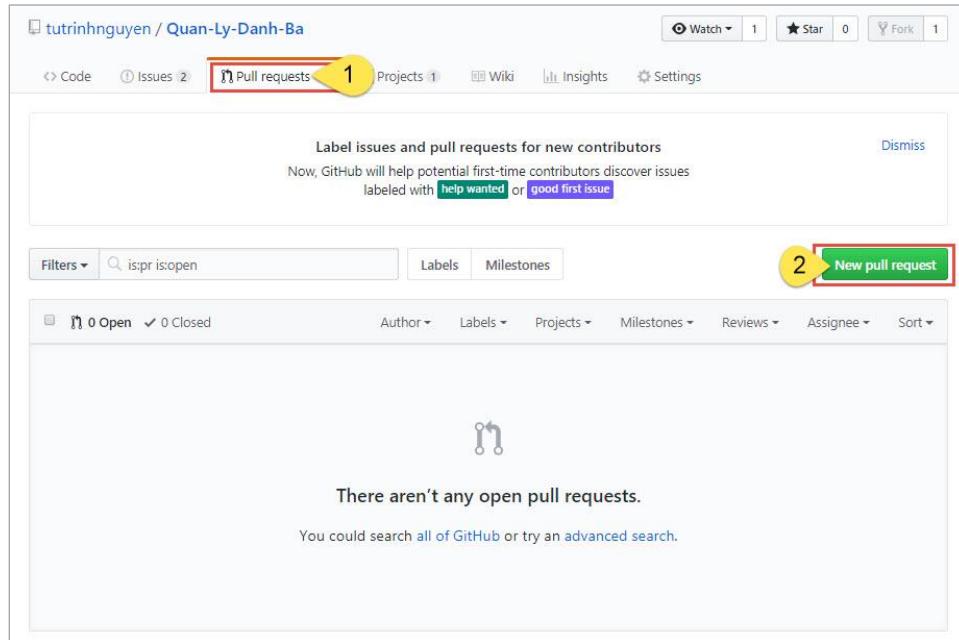
Description

Commit to Nhanh-file

A điền thông tin vào ô **Summary** và **Description**, rồi click **Commit to Nhanh-file**. Sau đó, click **Publish branch** để đẩy nhánh này lên web GitHub.

Sau khi A đẩy mã nguồn lên GitHub, để gửi yêu cầu gộp (pull request), A sẽ click sang tab **Pull requests**, chọn **New pull request**. Bạn lưu ý là mình sẽ gộp nhánh nào vào nhánh nào. Sau khi chọn nhánh cần gộp xong, click **Create pull**

request, và nhập tiêu đề, nội dung mô tả , rồi tiếp tục chọn **Create pull request**.

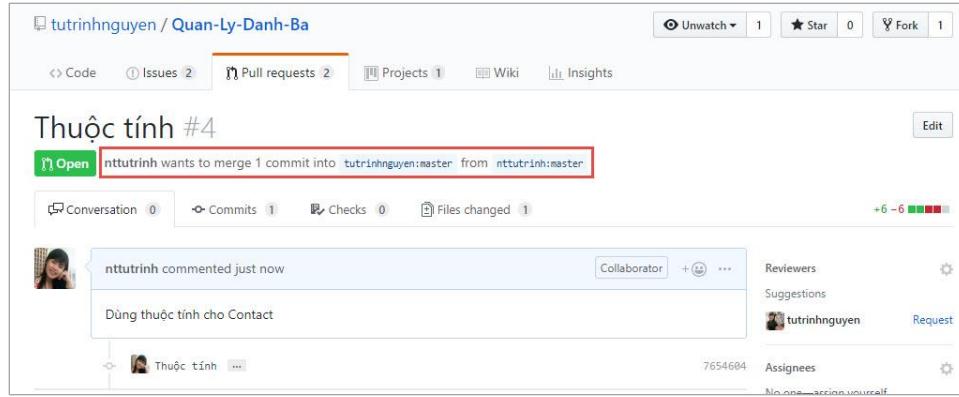


The image shows two screenshots of the GitHub interface. The top screenshot is titled 'Comparing changes' and shows a comparison between 'base: master' and 'compare: Nhanh-file'. It indicates that the branches are 'Able to merge'. A green button labeled 'Create pull request' is highlighted with a yellow circle and the number '4'. The bottom screenshot is titled 'Open a pull request' and shows the pull request creation form. The title 'Đọc và lưu danh bạ với file' is entered in the title field. The description field contains the text 'Chỉnh sửa ContactDatabaseOpenHelper và ContactDataProvider'. A green 'Create pull request' button is highlighted with a yellow circle and the number '5'.

Vậy là yêu cầu của A #3 – Đọc và lưu danh bạ với file đã được ghi nhận vào hệ thống.

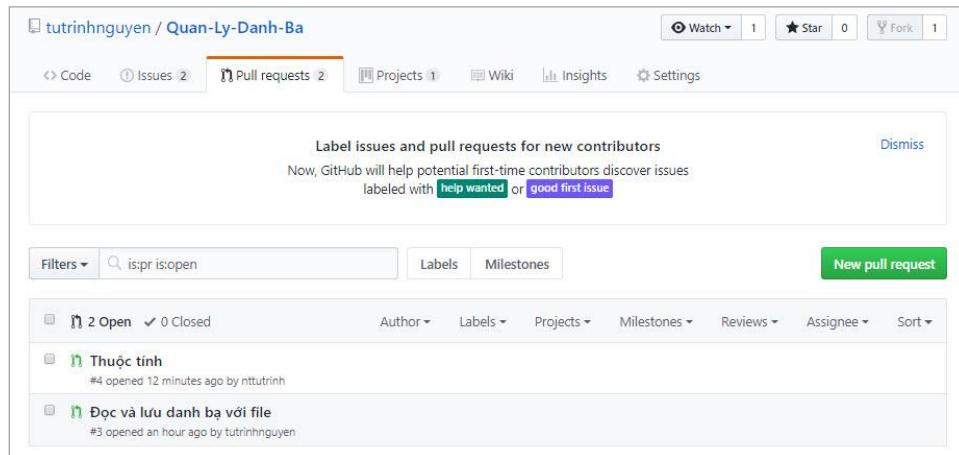
The image shows the GitHub pull request details for #3. The title is 'Đọc và lưu danh bạ với file #3'. A green 'Open' button is highlighted with a red box. The description field contains the text 'Chỉnh sửa ContactDatabaseOpenHelper và ContactDataProvider'. The pull request has 1 commit from 'tutrinhnghuyen' into the 'master' branch from 'Nhanh-file'. The status bar shows '+199 -0'.

Tương tự, B cũng tạo yêu cầu #4 - Thuộc tính như sau:

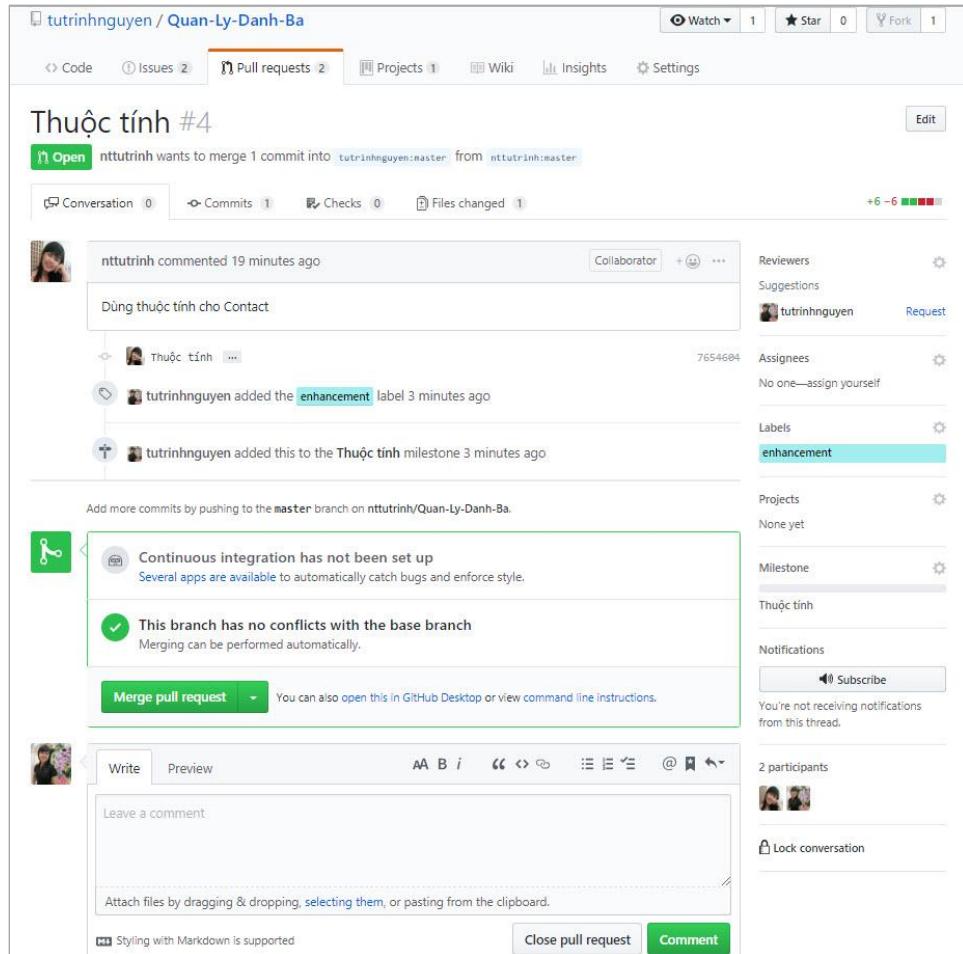


4.6. Xử lý trên các yêu cầu gộp

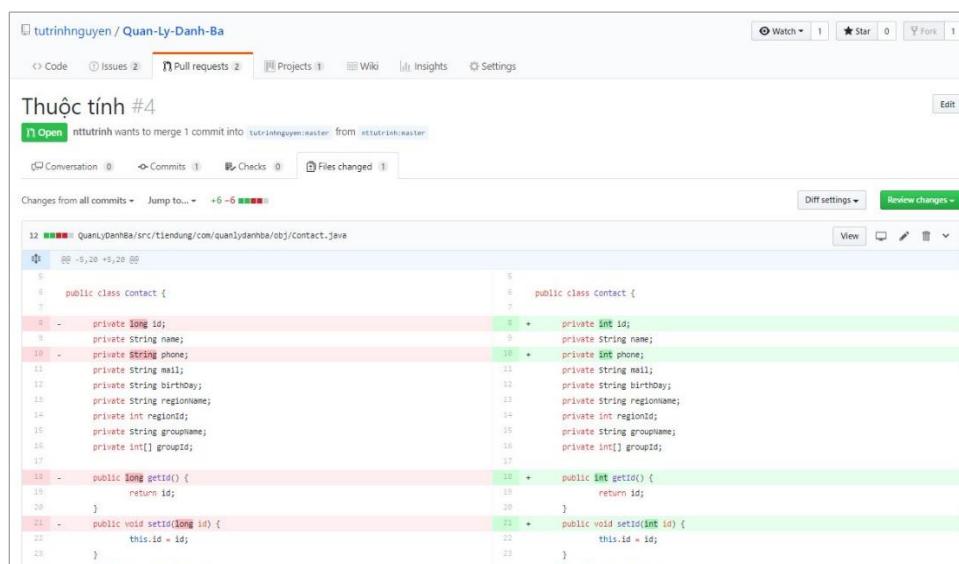
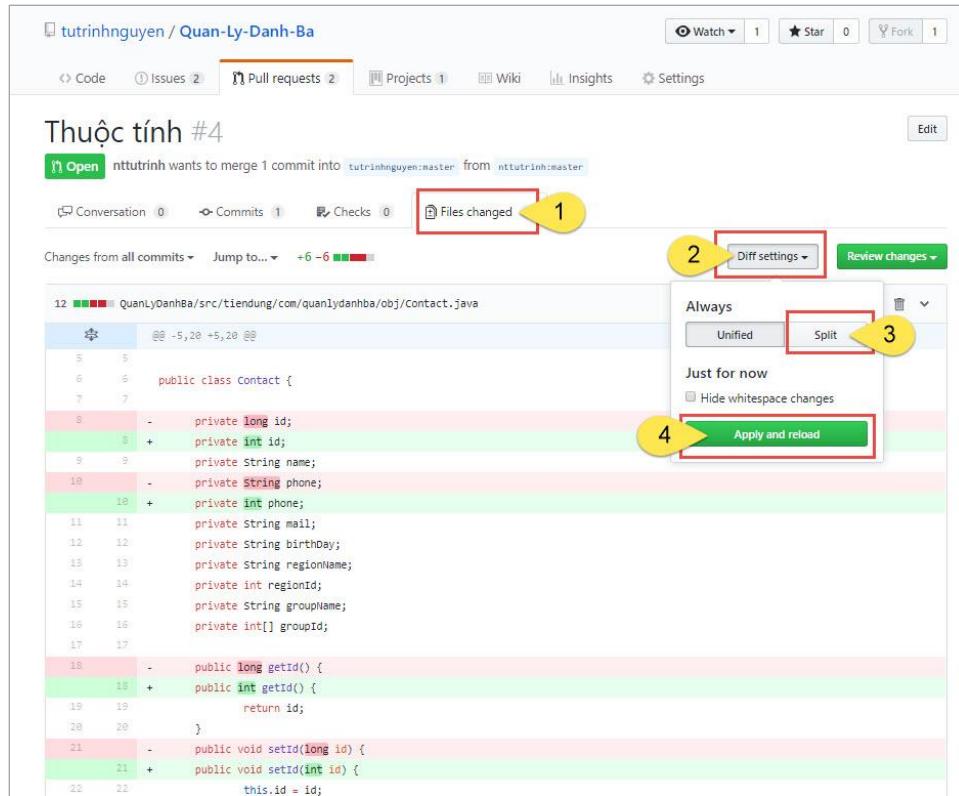
Lúc này A nhận được thông báo có yêu cầu gộp #4 từ nhánh master của B vào nhánh master của A. B nhận được thông báo có yêu cầu gộp #3 từ nhánh Nhanh-file vào nhánh master của A. Cả 2 yêu cầu gộp này đều xảy ra trên nhánh master của A, nên khi truy cập vào kho của A trên GitHub bạn có thể xem được hiện đang có các yêu cầu Pull Requests nào.



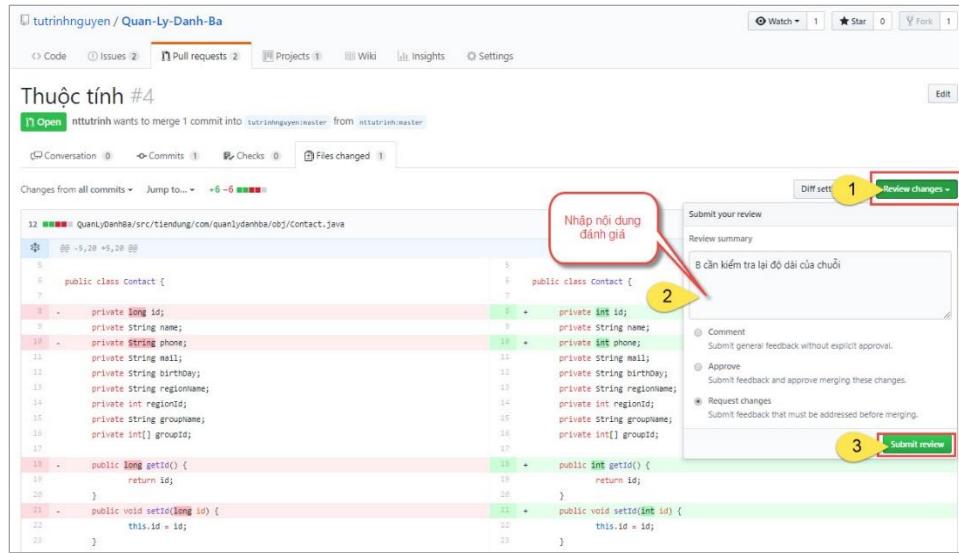
A vào xem yêu cầu gộp có tên **Thuộc tính** (của B) và chi tiết thông tin của lần lưu trạng thái muốn được gộp, những thay đổi trên các tập tin... Nếu thấy ổn, A có thực hiện gộp (**Merge pull request**).



Ngoài ra, GitHub còn hỗ trợ công cụ cho phép so sánh mã nguồn giữa các lần lưu trạng thái. Để xem phần so sánh này, bạn vào mục **Files changed**.

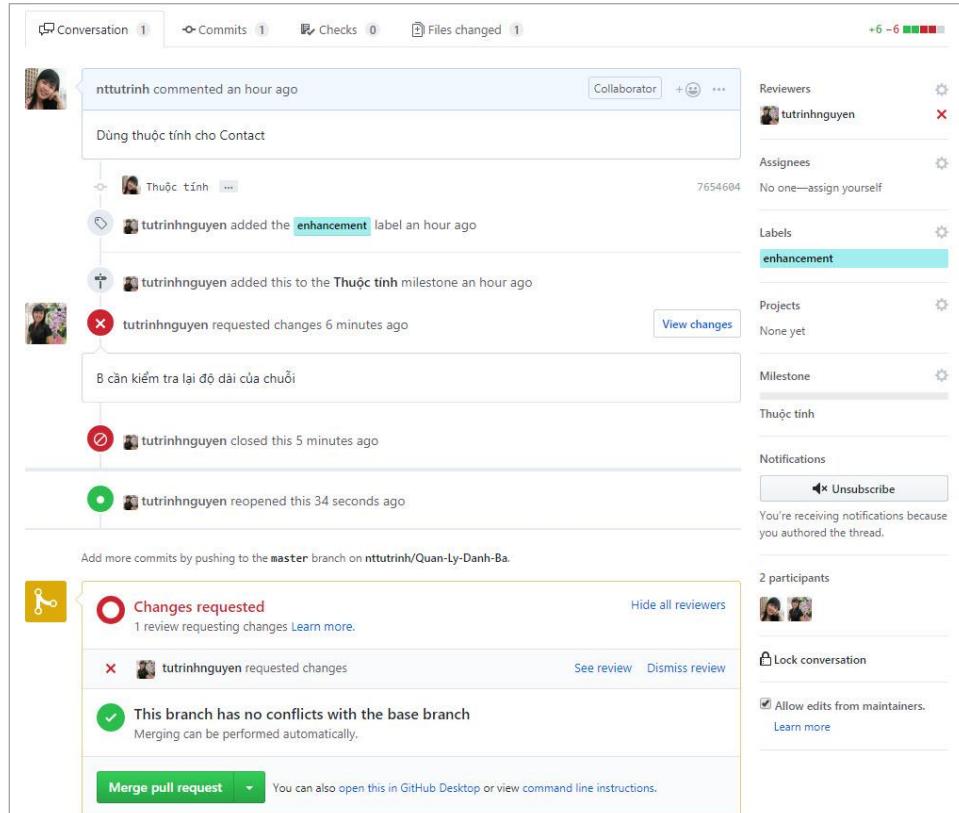


Khi so sánh mã nguồn (Files changed), bạn có thể chọn **Review changes** để đánh giá các thay đổi là **Comment**, **Approve** (chấp nhận cho gộp) hoặc **Request changes** (yêu cầu chỉnh sửa) và nhập vào các thông tin ghi chú thêm trong ô **Review summary**.

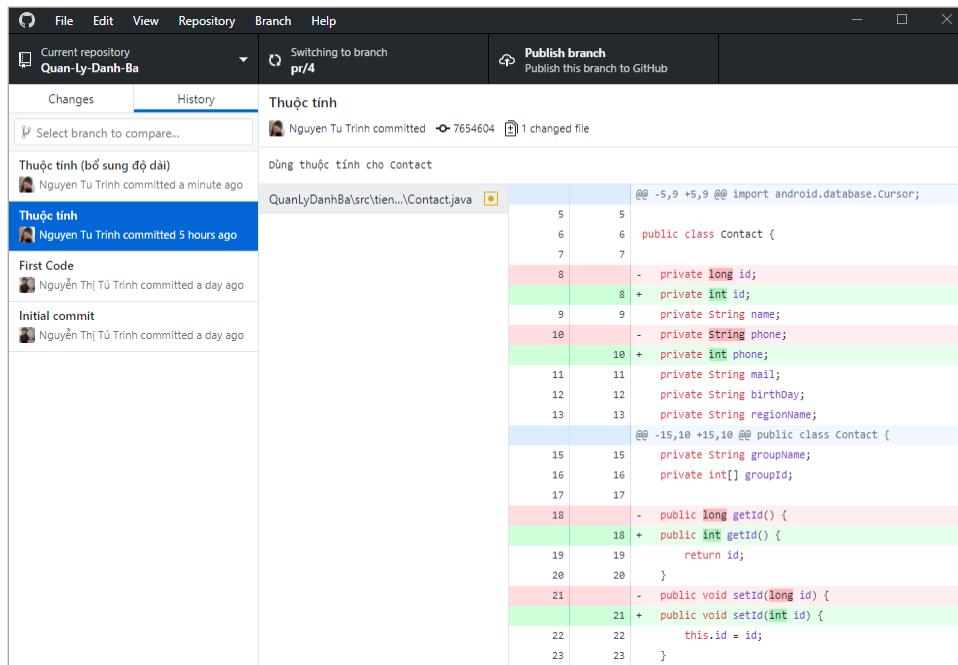
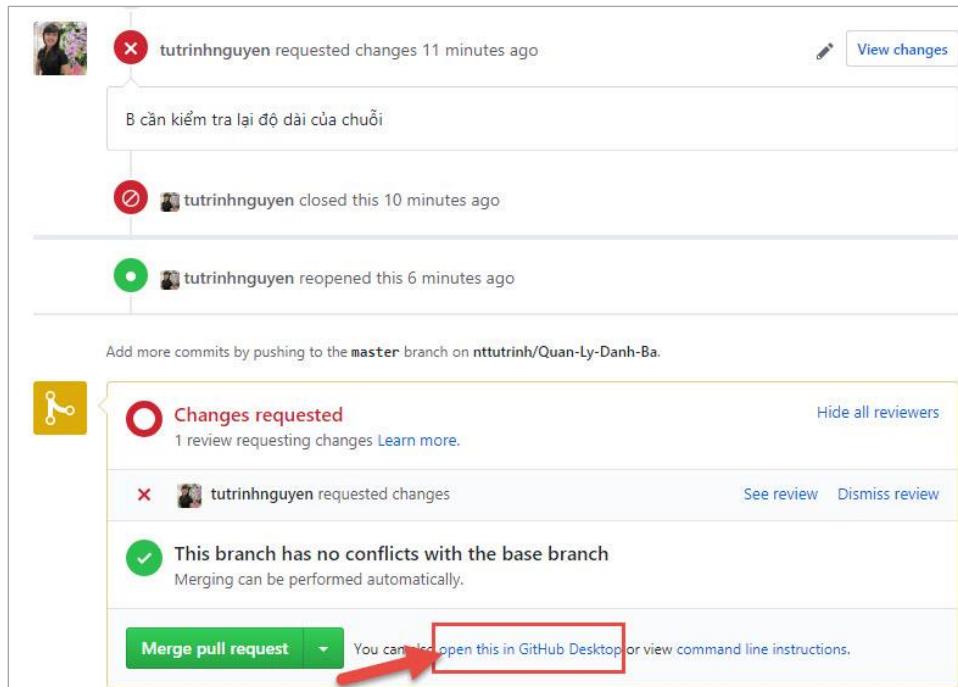


Giả sử A đã không đồng ý cho B gộp mã nguồn với lý do là B chưa có phần kiểm tra độ dài của các chuỗi được gán vào nên A chọn là **Request changes** và ghi chú để B biết lý do tại sao. Sau khi A click **Submit review** thì B sẽ nhận được thông báo. Lúc này A có thể click **Close Pull Request** để đóng yêu cầu gộp này lại.

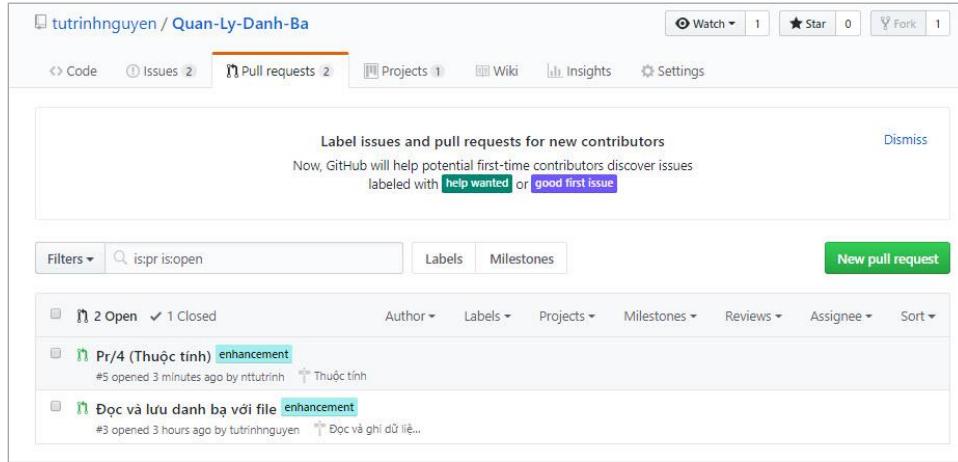
B sau khi xem thông báo sẽ biết được yêu cầu của mình tại sao bị từ chối



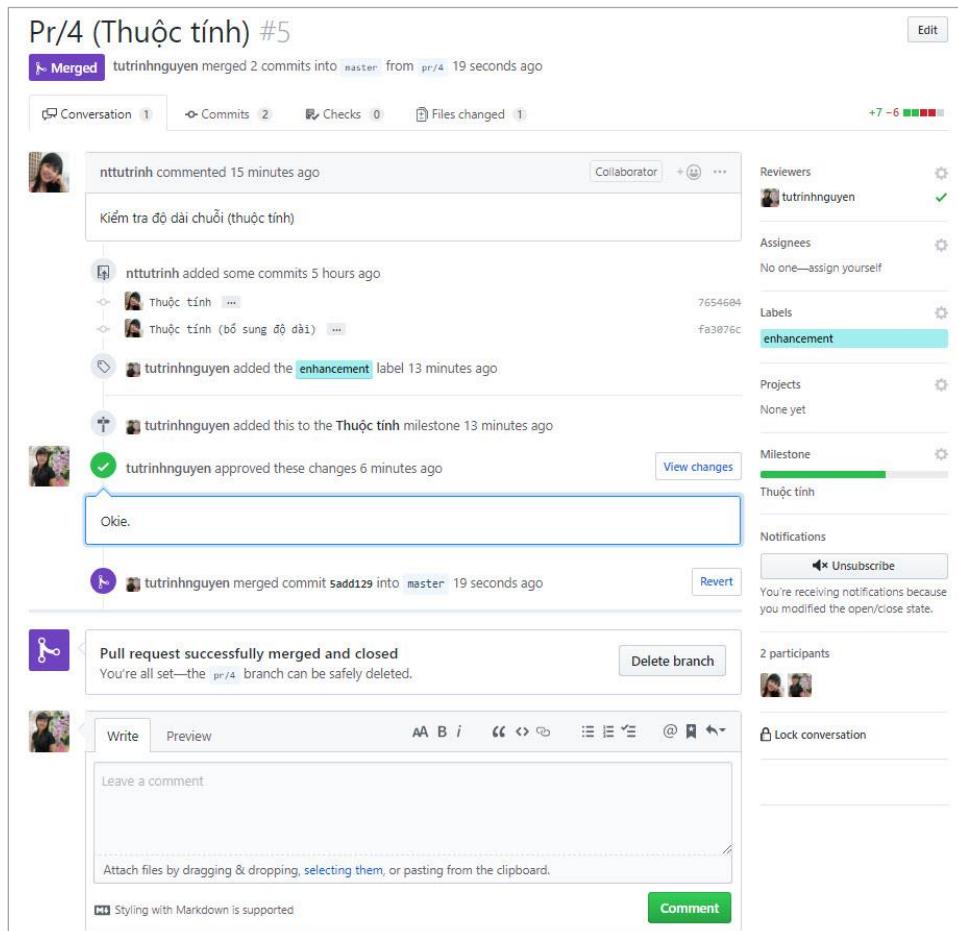
Như vậy, B sẽ biết được tại sao yêu cầu gộp của mình bị từ chối. Để hiệu chỉnh, B có thể bấm vào đường liên kết **open this in GitHub Desktop** và GitHub sẽ tự động lấy kho của A về và tạo một nhánh mới (ví dụ hình dưới, nhánh này tên là pr/4).



Sau khi B đã lập trình phần kiểm tra độ dài chuỗi ký tự và commit lưu trạng thái cho các đoạn mã nguồn, B sẽ thực hiện yêu cầu gộp mới. Lúc này trong kho của A sẽ có một yêu cầu gộp mới là Pr/4.



Sau khi A nhận được yêu cầu, A kiểm tra và đồng ý và click **Merge pull request** để thực hiện gộp.



Như vậy, vấn đề **Thuộc tính** đã được xử lý xong. A vào tab **Issues**, chọn vấn đề đã hoàn thành, bấm **Mark as** và chọn **Closed** để đóng vấn đề đó. Sau đó, A vào **Projects** để kéo nhiệm vụ **Thuộc tính** vào cột **Done**.

The image shows two screenshots of a GitHub repository named 'Quan-Ly-Danh-Ba'.

Top Screenshot (Issues Tab):

- Step 1: The 'Issues' tab is selected, highlighted with a red box and a yellow circle.
- Step 2: An issue titled 'Thuộc tính enhancement' is selected, highlighted with a yellow circle.
- Step 3: A context menu is open for the selected issue, with the 'Mark as' dropdown highlighted with a red box and a yellow circle.
- Step 4: The 'Closed' option in the dropdown menu is highlighted with a yellow circle.

Bottom Screenshot (Projects Tab):

A project board titled 'QuanLyDanhBa' is shown with three columns: 'To Do', 'In Progress', and 'Done'. The 'Done' column contains the issue 'Thuộc tính enhancement'.

Lời kết

Cám ơn bạn đã đọc đến trang cuối của cuốn cẩm nang này. Hy vọng bạn đã có được những kiến thức hữu ích về Git và GitHub, công cụ không thể thiếu với Lập trình viên và những ai yêu thích lập trình.

Mọi đóng góp, phản hồi, bạn vui lòng gửi qua qua email daotaolaptrinh@gmail.com. Chúng tôi thật sự trân quý và mong muốn lắng nghe.

Hẹn gặp lại bạn trong các cẩm nang sau....