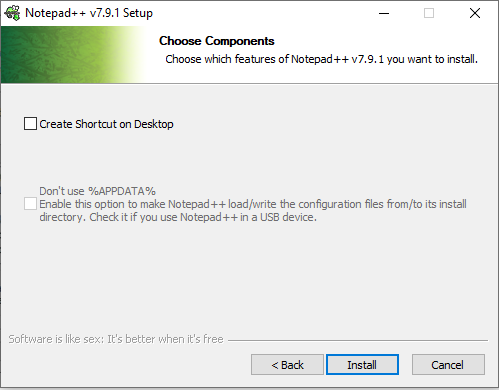
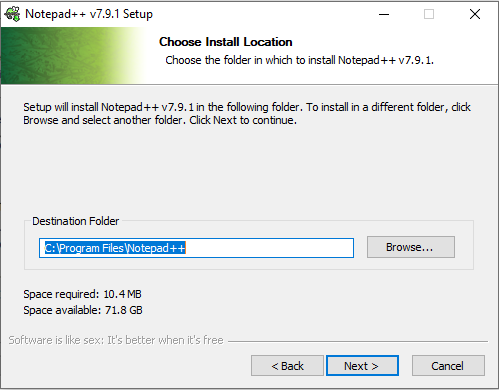
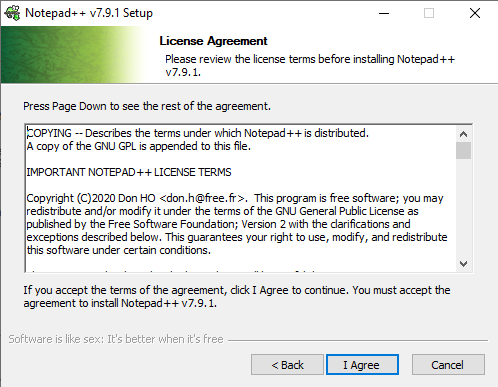
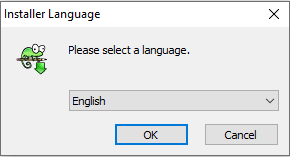
**Hướng dẫn sử dụng GitHub với Git Extensions**

**1. Cài các phần mềm cần thiết, gồm Notepad++, Git, KDiff3 và Git Extensions.**

**1.1. Cài đặt Notepad++**

Tải từ <https://notepad-plus-plus.org/downloads/> hoặc dùng bộ cài đi kèm

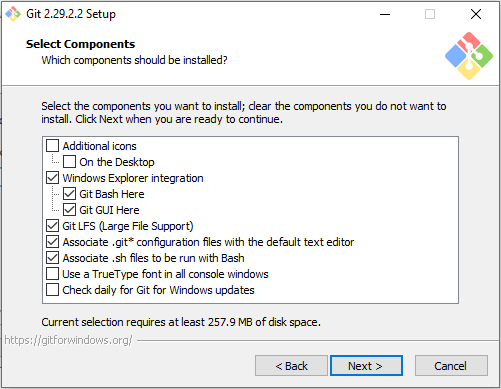
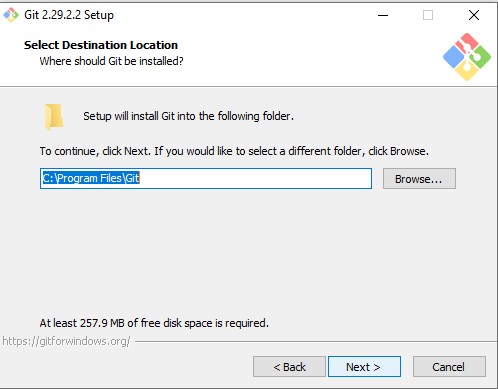
Màn hình diễn biến cài đặt như sau:

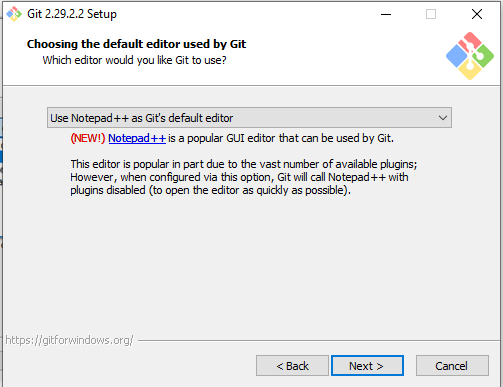
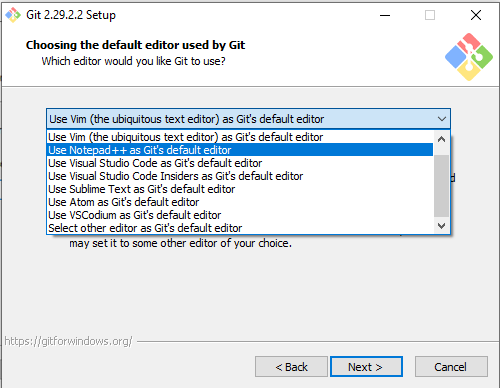
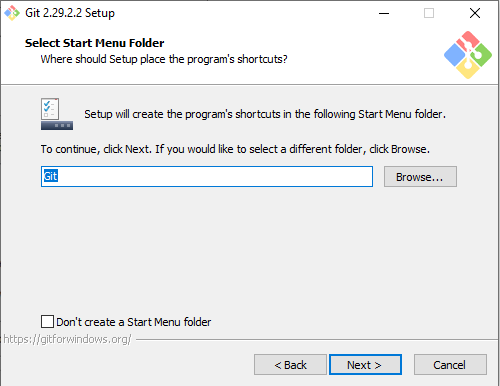


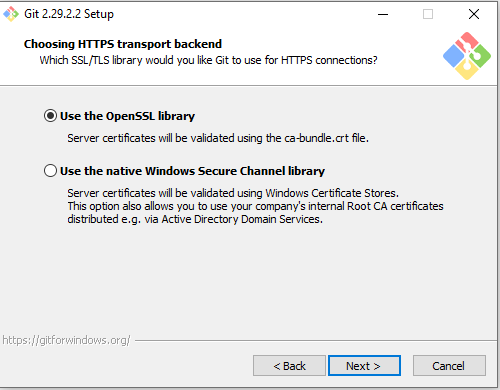
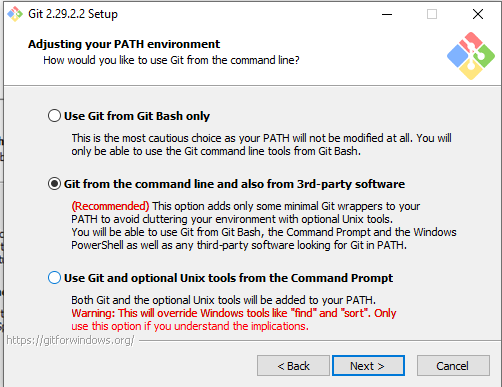
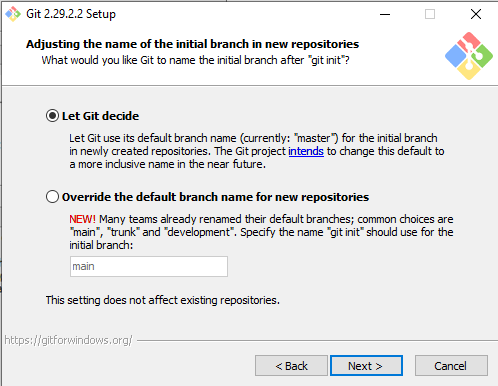
**1.2. Cài đặt phần mềm Git**

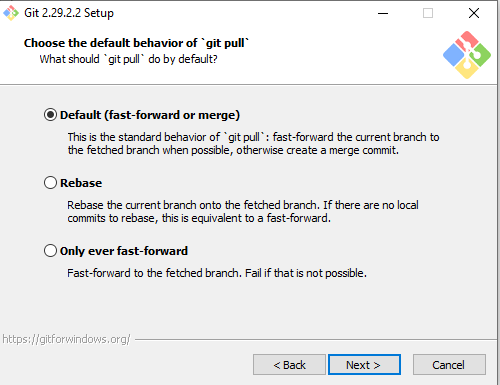
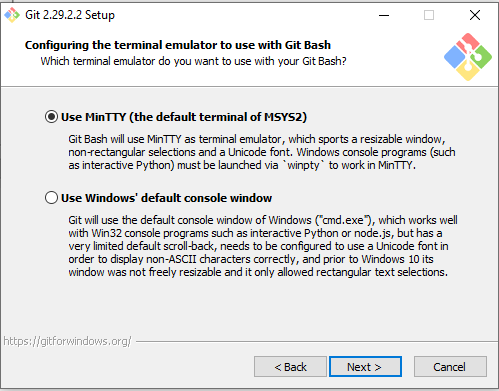
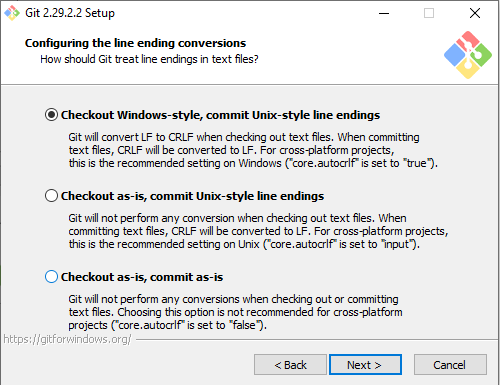
Tải bản mới nhất từ <https://git-scm.com/downloads> hoặc sử dụng bộ cài đi kèm

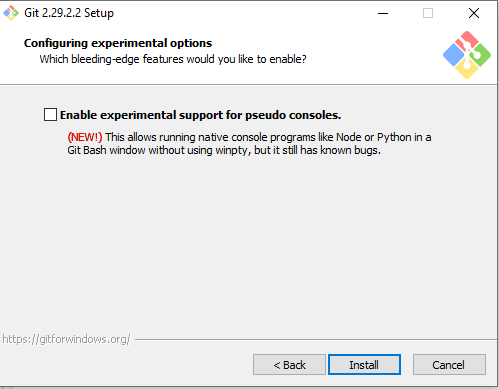
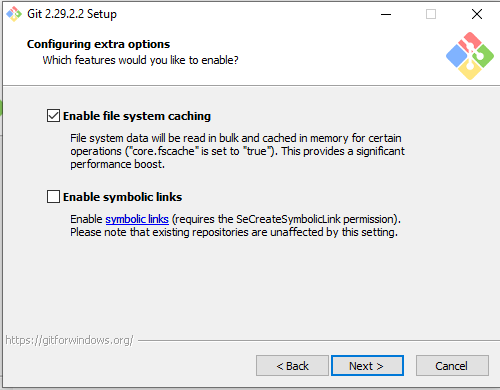
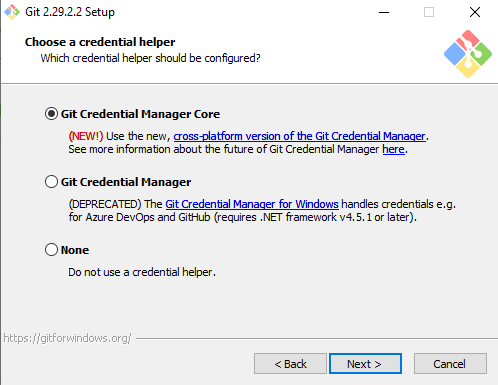
Mành hình diễn biến cài đặt như sau:

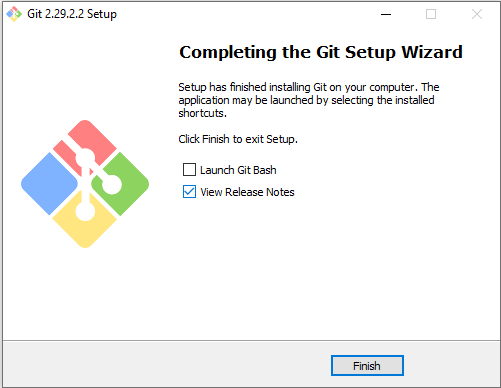
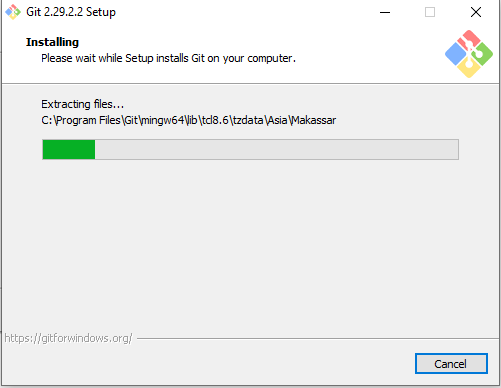






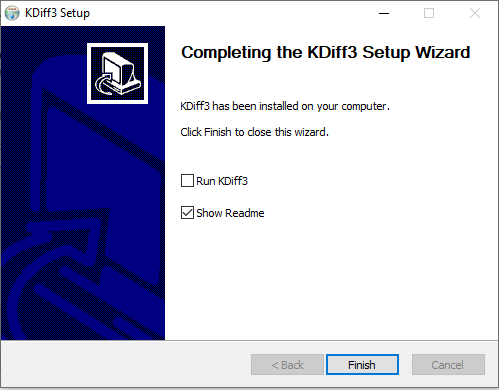
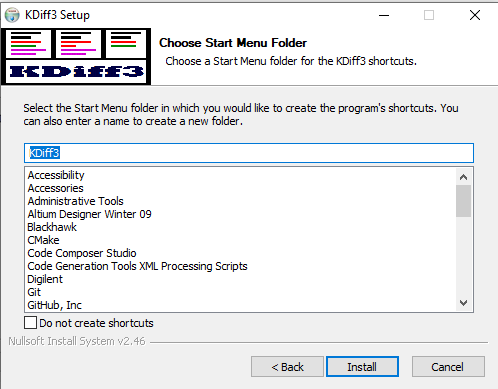
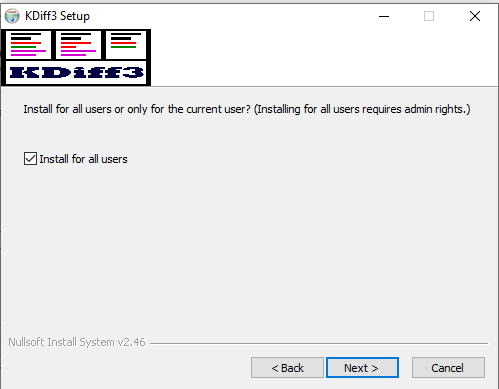
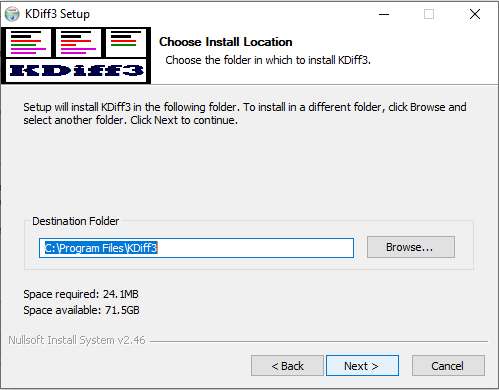
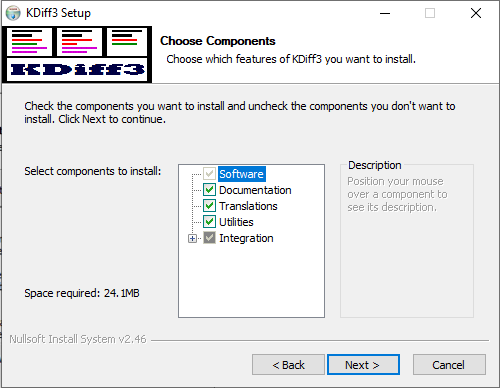
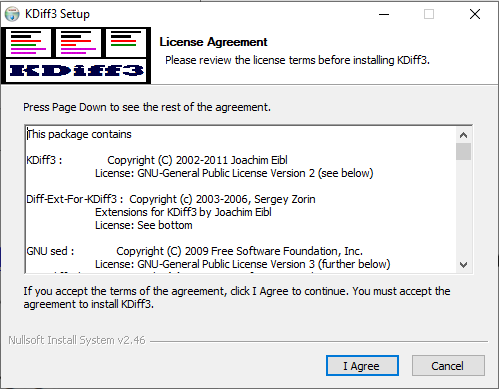
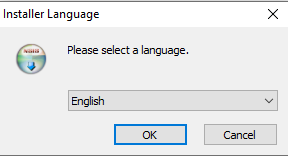






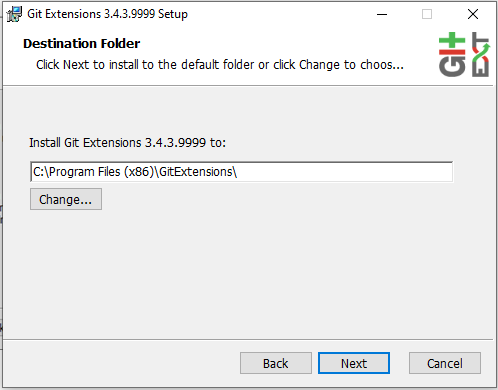
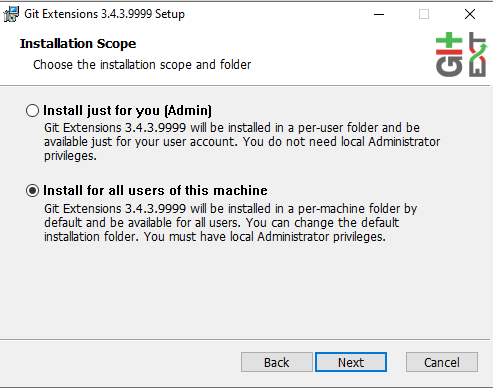
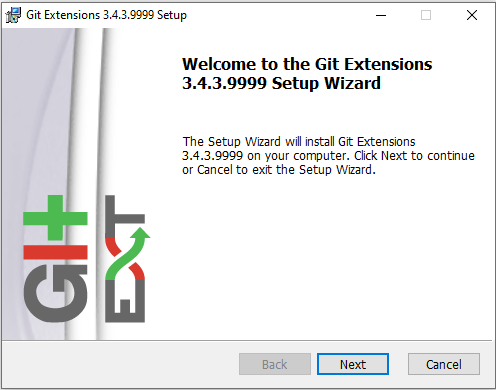
**1.3. Cài đặt Kdiff3**

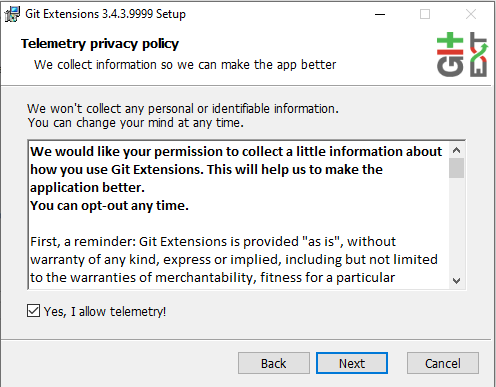
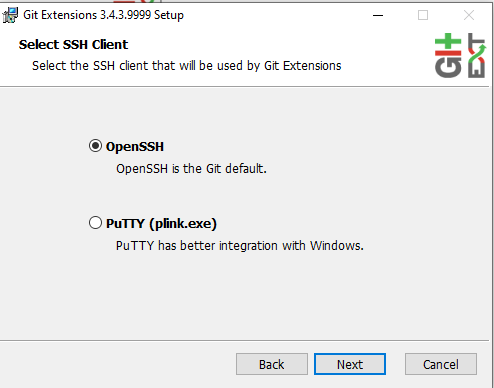
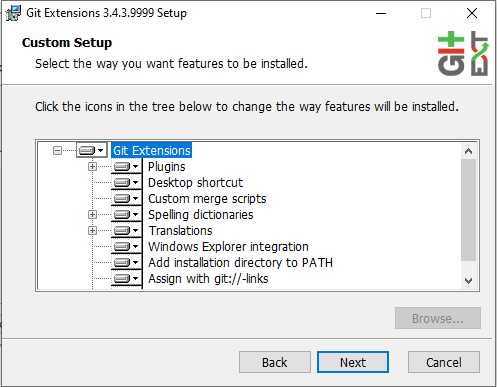
Tải tại <https://sourceforge.net/projects/kdiff3/> hoặc dùng bộ cài đi kèm

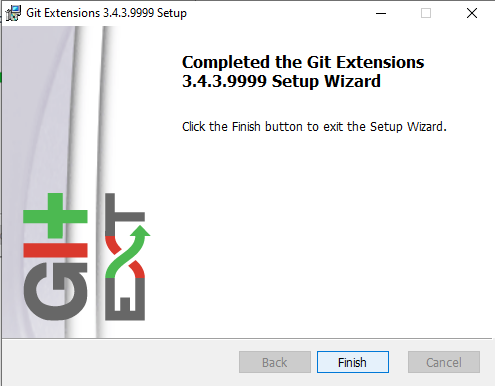
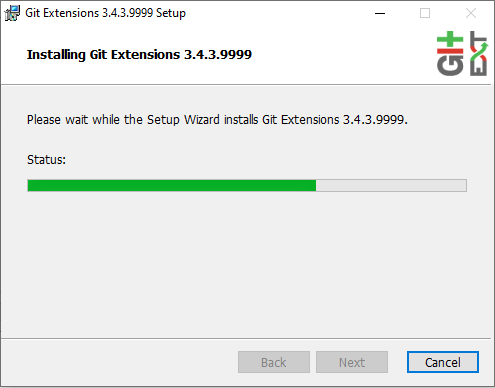
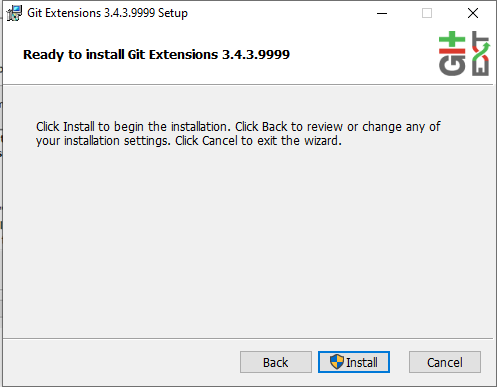
****

**1.4. Cài đặt GitExtension**

Tải tại <https://github.com/gitextensions/gitextensions/releases/tag/v3.4.3> hoặc dùng bộ cài đi kèm







**2. Thiết lập phần mềm Git Extensions**

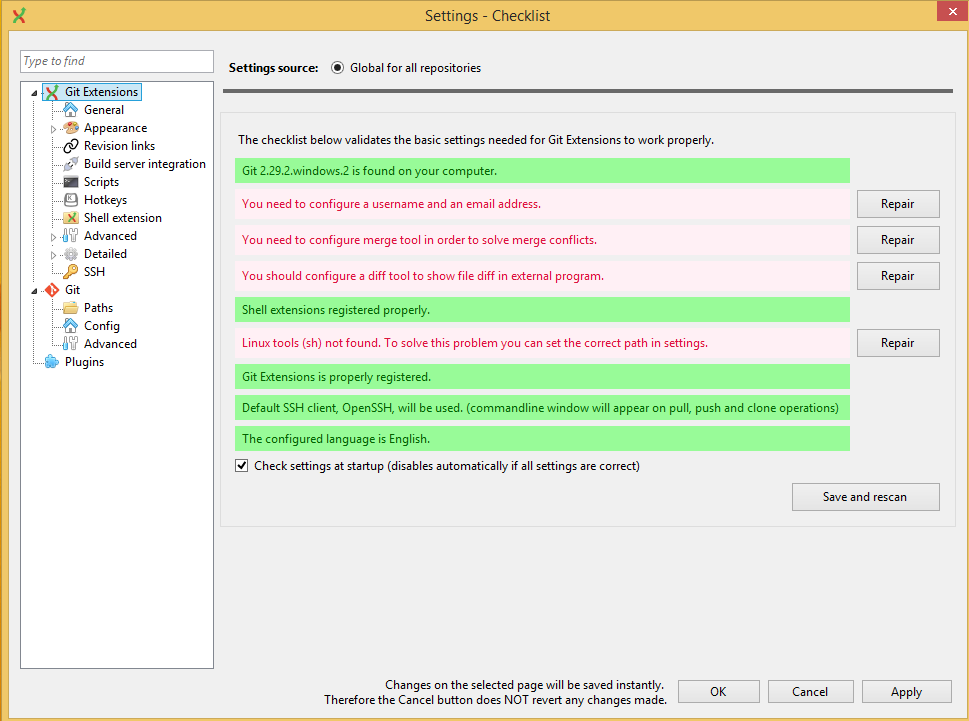
***2.1. Thiết lập ngôn ngữ***

Khi bật lần đầu phần mềm sẽ yêu cầu chọn ngôn ngữ (tiếng Anh) như hình

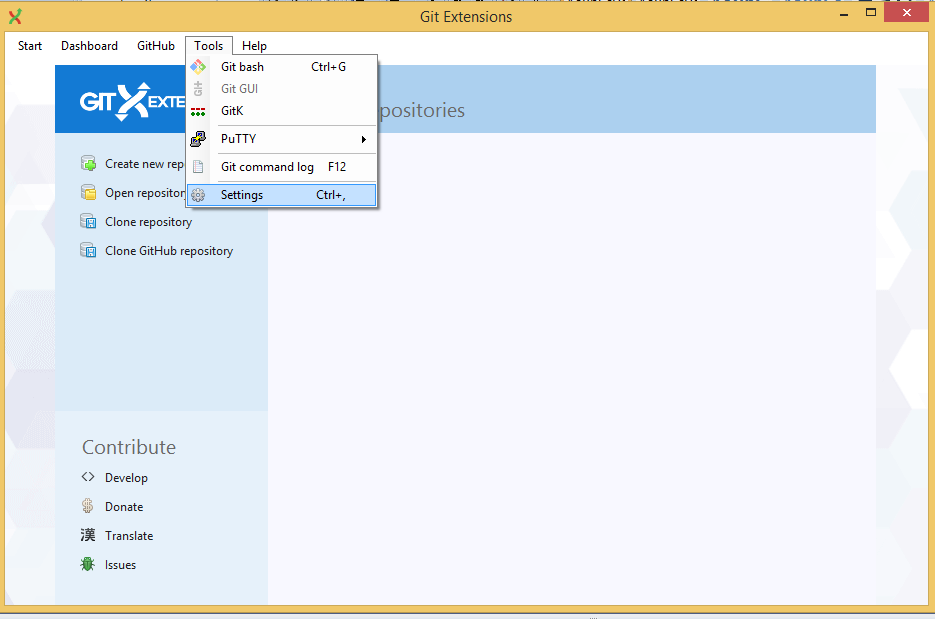


***2.2. Bổ sung thông tin người dùng và công cụ***

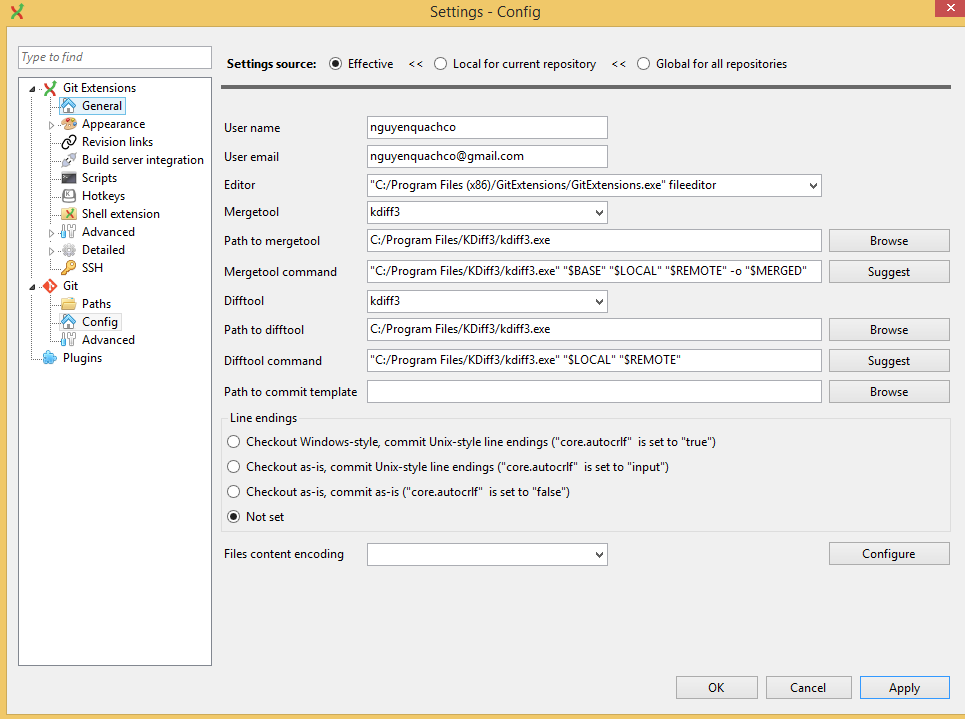
Tiếp theo sẽ chuyển qua cửa sổ Settings.



Nếu phần mềm không mở Settings thì vào Tools\Settings để mở:



Ở cửa sổ Settings, những cài đặt còn thiếu sẽ bị tô màu đỏ và có nút Repair bên cạnh. Ta click đúp vào dòng màu đỏ hoặc click vào nút Repair. Cửa sổ Config hiện ra:

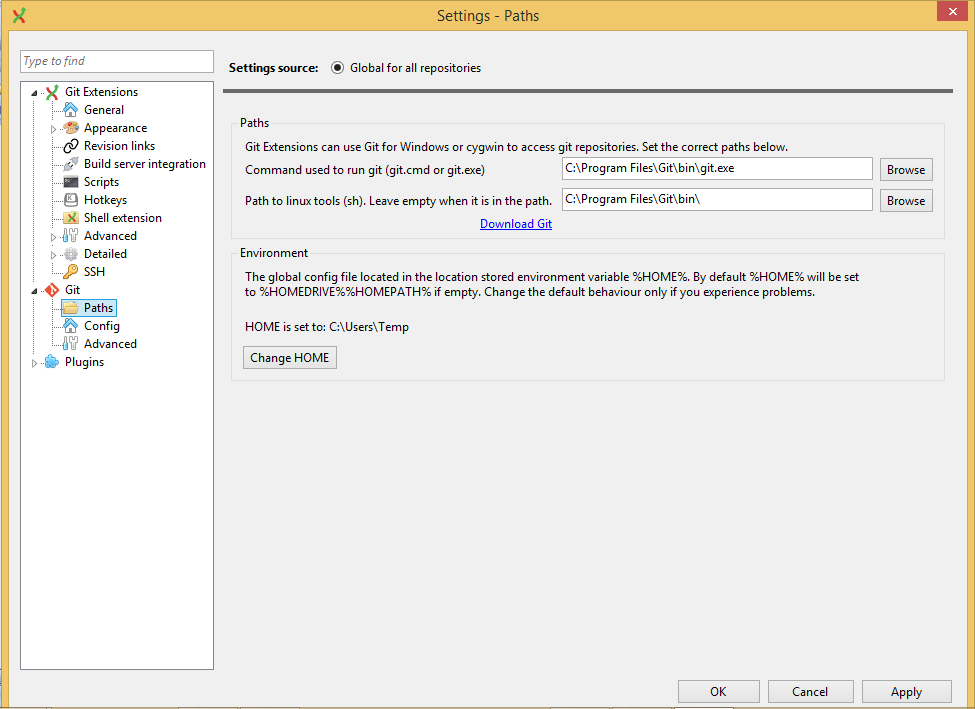


Ta sẽ cần nhập vào User name và User email. Hai thông tin này được phần mềm dùng để ghi công người đã đóng góp vào dự án (chỉ là để phân biệt giữa những người đóng góp với nhau, không cần phải chính xác lắm).

Tiếp theo là chọn Mergetool là kdiff3. Chọn Difftool cũng là kdiff3. Rồi bấm vào Apply.

***2.3. Thiết lập Git***

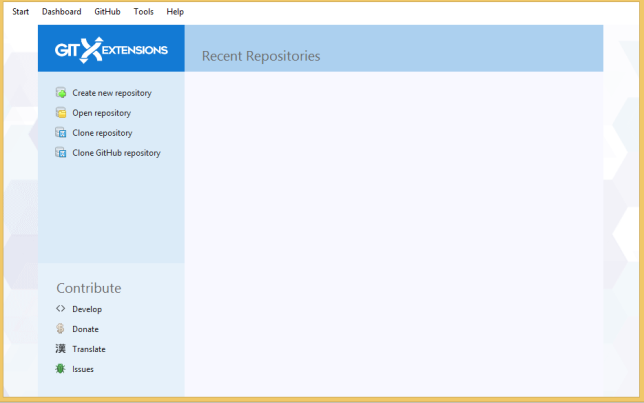
Trong cửa sổ Settings click vào nút Git ở cột bên trái, sau đó click vào mục Paths. Ta phải chỉ đường dẫn đến thư file Git\bin\git.exe và thư mục Git\bin\ như hình vẽ (là địa chỉ ta đã cài đặt Git ở bước 1).



Trong cửa sổ Settings, click vào Git Extensions ở cột bên trái để . Nếu không còn dòng nào màu đỏ (tất cả các dòng đều chuyển qua màu xanh) là xong phần thiết lập Git Extensions:



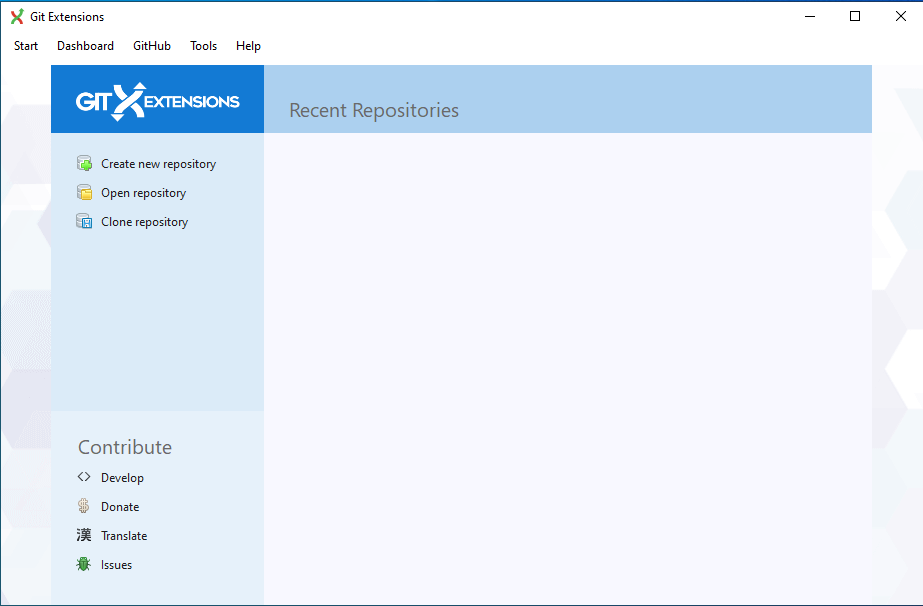
Sau khi thiết lập xong, bấm OK để quay về cửa sổ làm việc của Git Extensions như hình sau:



**3. Sử dụng Git Extensions**

***3.1. Tạo kho dữ liệu***

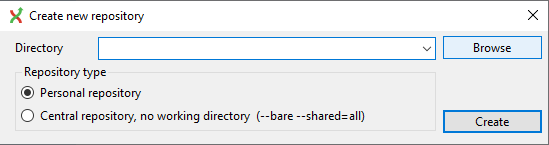
Bật phần mềm Git Extensions lên sẽ có giao diện như hình:



Git Extensions là phần mềm hỗ trợ quản lý phiên bản. Giả sử ta có một project (có thể là word, bản vẽ, mã nguồn lập trình, v.v.), về tổng quát là một thư mục chứa các file và thư mục khác. Git Extensions cho phép ta quản lý các thay đổi của các file và thư mục con trong thư mục đó như ghi nhận thời giant hay đổi, người thay đổi, v.v. và cho phép khôi phục lại tình trạng của thư mục (tức các file và thư mục con chứa trong đó) về một phiên bản nào đó đã lưu trước đó, đồng thời so sánh sự khác nhau của các phiên bản.

Git Extensions gọi các thư mục chứa dữ liệu mà nó quản lý là repository (kho dữ liệu) và sẽ quản lý các kho đó.

Để tạo 1 kho, ta click chọn Create new repository hoặc vào menu Start\Create new repository. Cửa sổ hiện ra cho phép ta chọn thư mục cần quản lý:

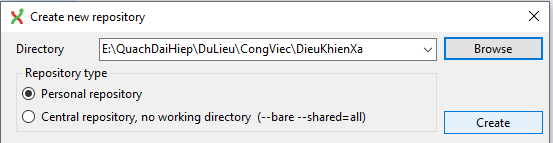


Có 2 loại repository (kho):

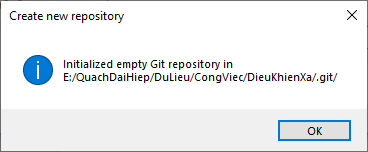
- Personal repository là loại kho cá nhân, tức thư mục làm việc của ta. Tại đó chứa các file và dữ liệu của ta và ta có thể thay đổi các dữ liệu này. Thông thường được lưu trữ trên máy tính cá nhân của ta.

- Central repository là loại kho trung tâm, chứa bản sao của dữ liệu ta. Các dữ liệu này thường được nén để giảm dung lượng nên ta thường không xem được và không làm việc trực tiếp được với các dữ liệu này, nhưng có thể tải về kho Personal repository của ta (và các dữ liệu sẽ được khôi phục lại ở dạng ban đầu để ta có thể thao tác trực tiếp với nó). Central repository cũng cho phép nhiều người cùng truy cập và gửi dữ liệu (các phiên bản của họ về dữ liệu) cũng như tải về kho cá nhân của họ. Central repository có thể được lưu trên máy tính cá nhân, máy chủ hoặc trên mạng (như GitHub chẳng hạn).

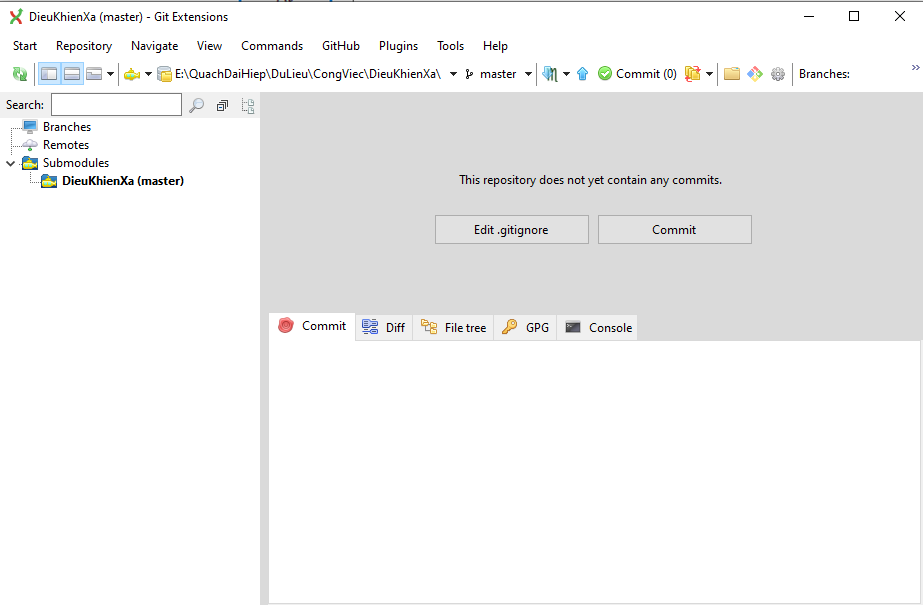
Ta click Browse và chọn đường dẫn đến thư mục chứa dữ liệu và chọn Personal repository rồi bấn Create để tạo kho cá nhân:



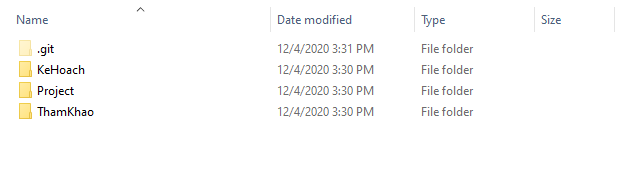
Git Extensions sẽ báo tạo repository mới như hình:



Đầu tiên khi tạo xong thì Git Extensions sẽ báo là tạo repository mới như hình:

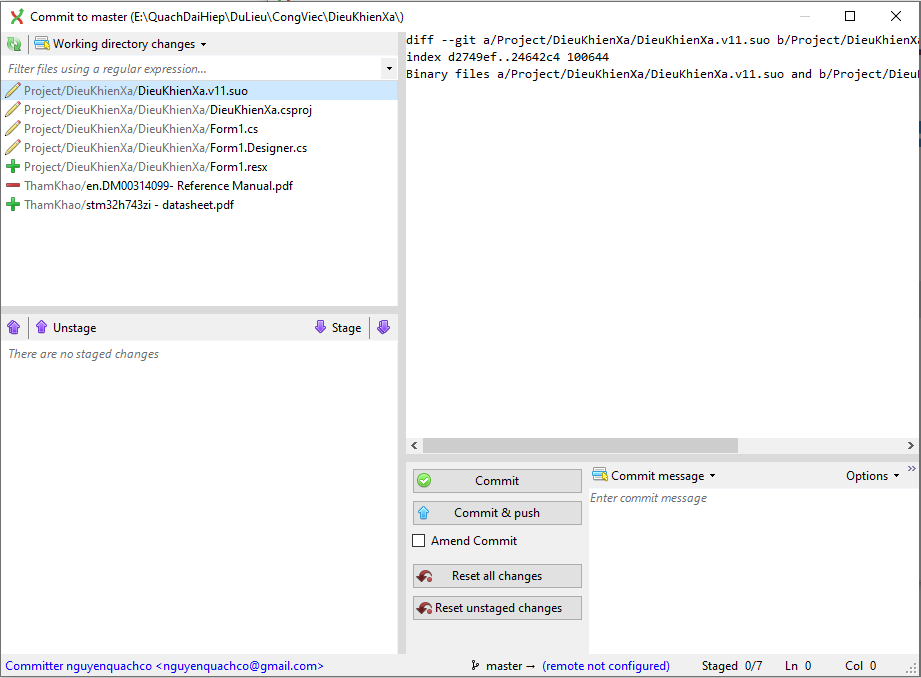


Sau khi tạo thì trong thư mục của ta sẽ có thêm một thư mục ẩn với tên là .git như hình dưới. Đây chính là nơi lưu trữ thông tin về các phiên bản dữ liệu của thư mục.

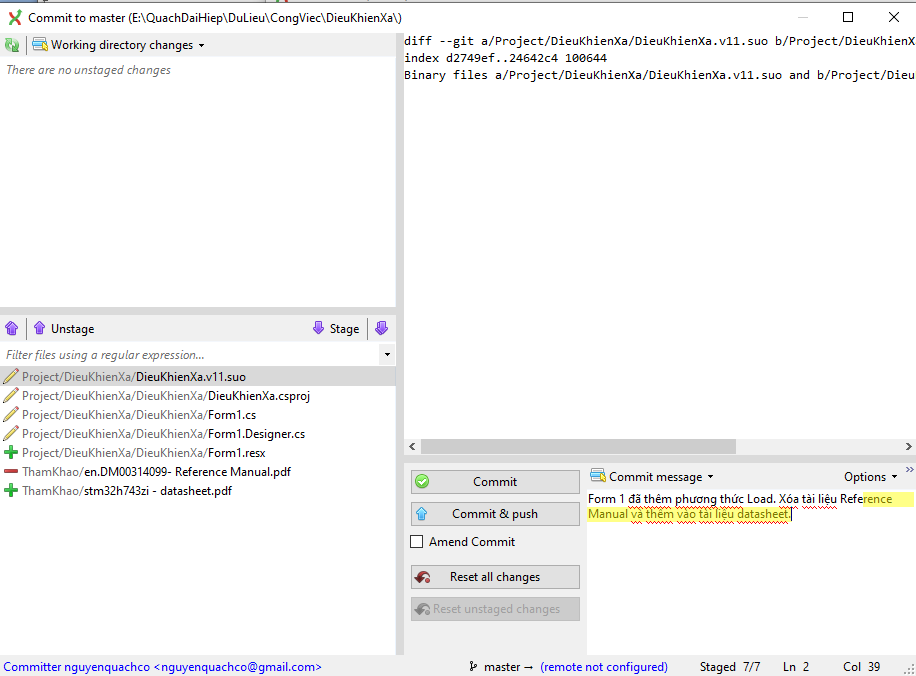


***3.2. Lưu các phiên bản***

Ta bấm vào commit để lưu một phiên bản của dữ liệu vào kho của Git Extensions. Mỗi lần commit, các thay đổi sẽ được lưu vào kho của Git Extensions và có thể quản lý, khôi phục lại khi cần.

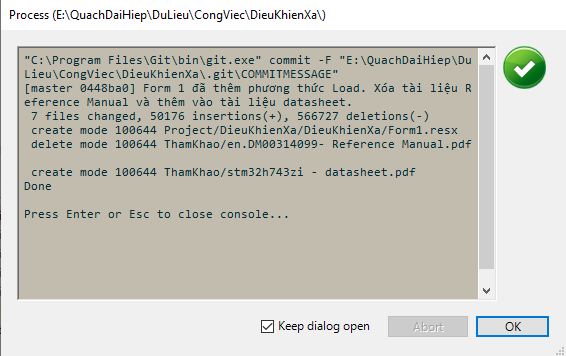


Cột bên trái chứa các thay đổi so với phiên bản trước đó như thêm mới (có biểu tượng dấu + màu xanh), chỉnh sửa (có biểu tưởng cái bút) hay xóa (có biểu tượng dấu – màu đỏ). Ta chọn các thay đổi (các file) cần lưu lại ở phiên bản này và bấm vào nút Stage (dấu mũi tên xuống màu tím). Nếu muốn lưu lại tất cả thì chọn vào nút Stage All bên cạnh.

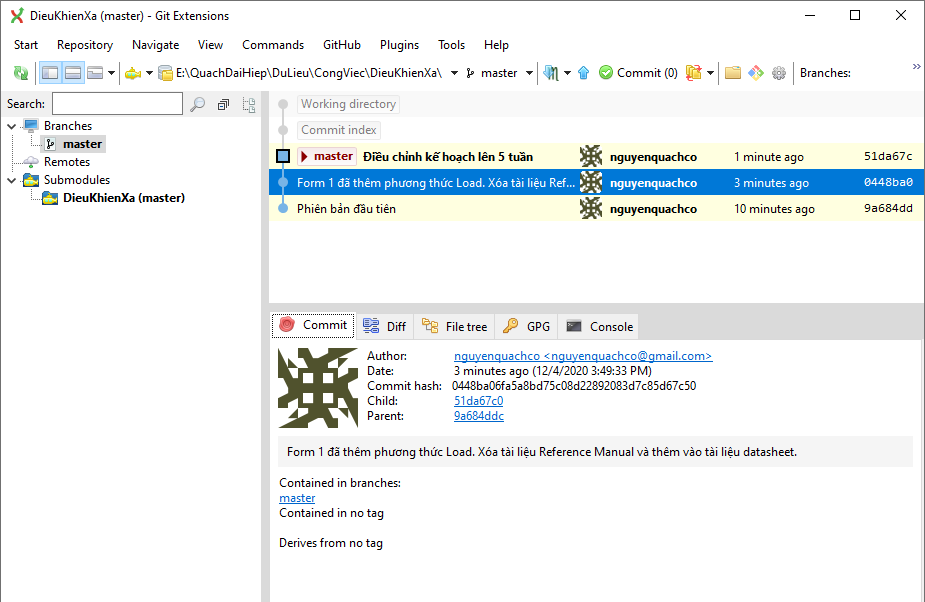


Sau đó thêm nội dung ghi chú của phiên bản này (commit message là nội dung vắn tắt miêu tả các nội dung đạt được hoặc các thay đổi, gợi nhớ, v.v. để sau này ta biết phiên bản đó là phiên bản nào), rồi bấm vào Commit.

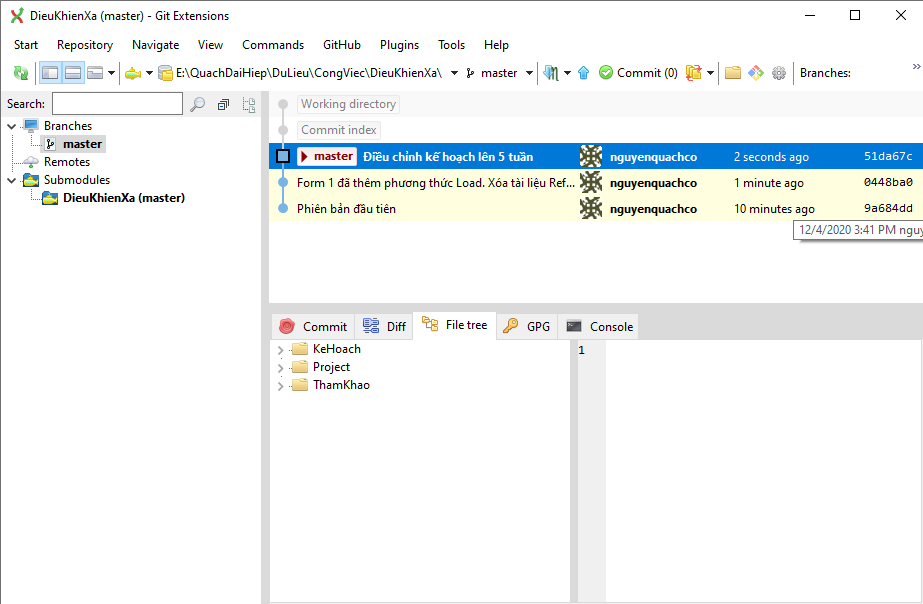
Git Extensions sẽ thêm phiên bản này vào kho và thông báo khi nào thành công.



Quay trở lại màn hình chính, Git Extensions sẽ hiện thị danh sách các phiên bản cùng với nội dung ghi chú của các phiên bản đó, kèm theo tên người đóng góp và thời gian đóng góp.

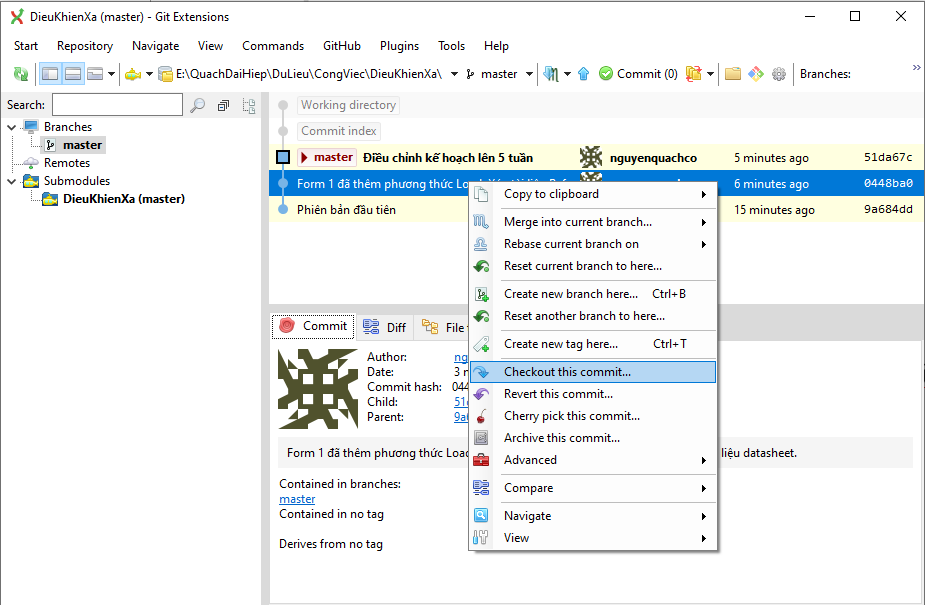


Cửa sổ phía dưới cho phép ta quan sát chi tiết hơn các phiên bản như so sánh khác nhau giữa các phiên bản, danh sách các file và thư mục của phiên bản, v.v.

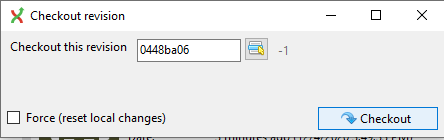


***3.3. Khôi phục phiên bản. Tạo nhánh mới***

Khi ta cần khôi phục lại phiên bản trước đó, ta click chuột phải vào phiên bản đó và bấm **Checkout this commit**:

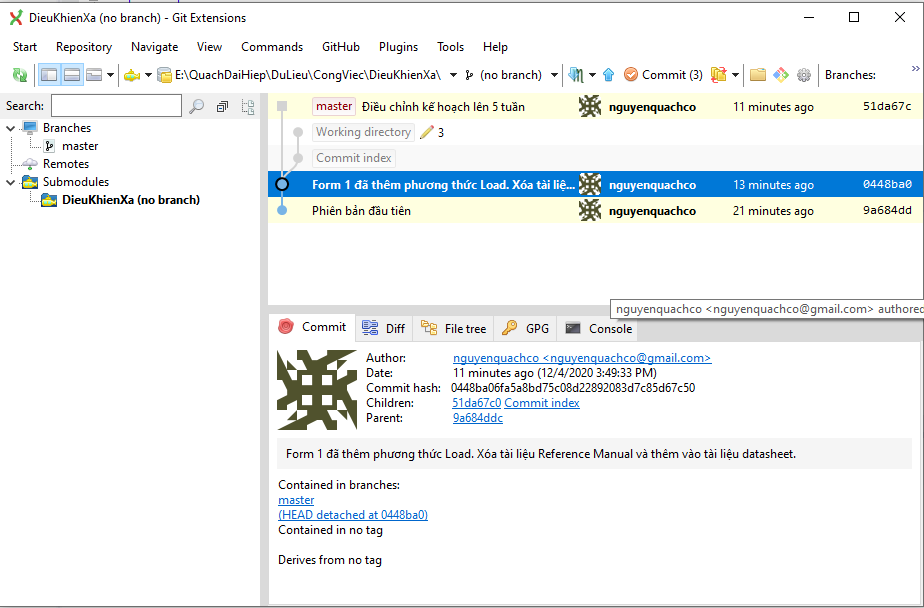


Cửa sổ hiện ra thông báo khôi phục phiên bản, ta bấm vào **Checkout** để khôi phục.

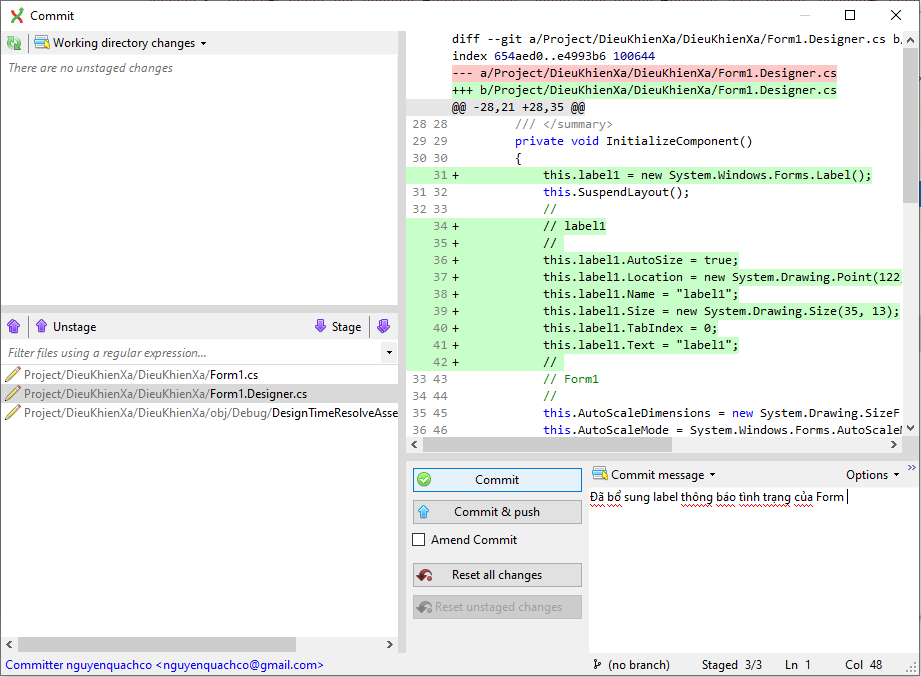


Chú ý rằng khi ta khôi phục lại, chỉ có các phần dữ liệu trong thư mục làm việc của ta là bị khôi phục lại (bản thân dữ liệu về các phiên bản được Git Extensions lưu trong kho sẽ không thay đổi).

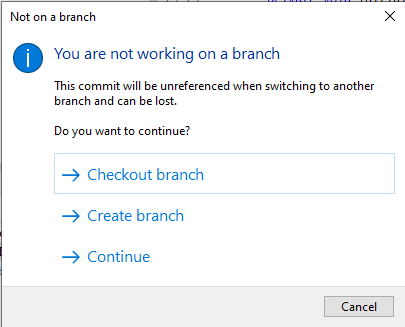
Ta có thể thực hiện các thay đổi ở phiên bản vừa khôi phục và muốn lưu lại. Lúc này, Git Extensions sẽ chuyển nút Commit sang biểu tượng **màu vàng** (tức là có thay đổi) và thông báo kèm theo số file đã thay đổi. Ta click vào **Commit**:



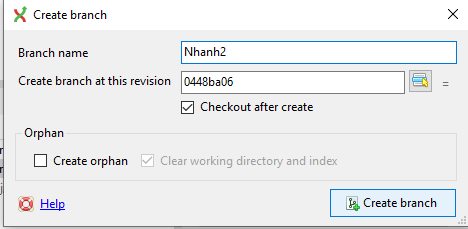
Quá trình Commit tương tự như trên. Ta chọn các file thay đổi cần lưu lại, viết nội dung vắn tắt của phiên bản và bấm **Commit**.



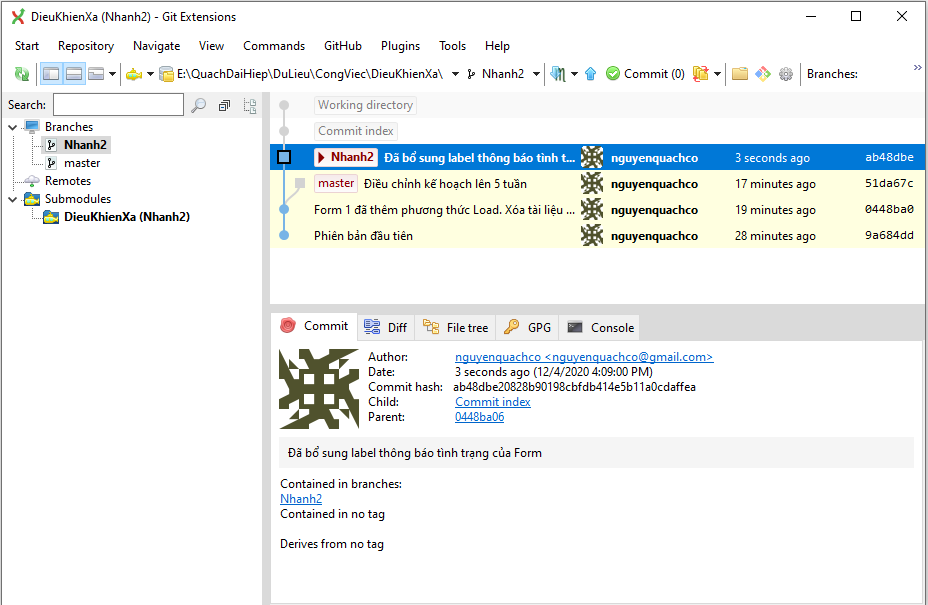
Lúc này cửa sổ hiện ra thông báo cho ta biết là ta không ở một nhánh nào cả (do ta khôi phục lại phiên bản cũ). Ta bấm vào **Create branch** để tạo ra nhánh mới:



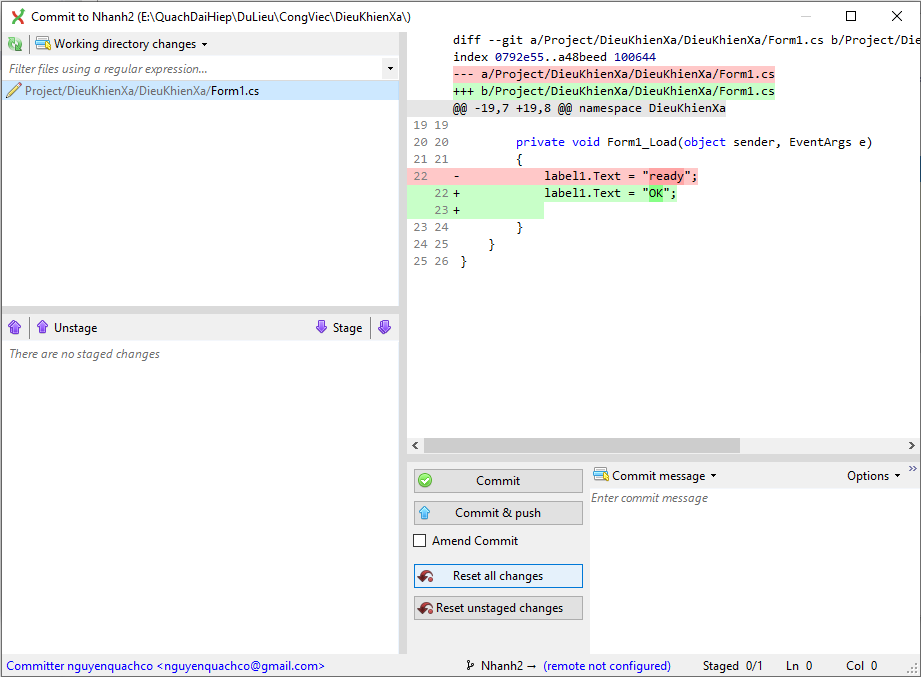
Ta nhập vào tên nhánh và bấm **Create branch**:



Sau khi tạo nhánh mới, quay trở lại Git Extensions sẽ hiển thị danh sách các phiên bản theo dạng sơ đồ cây như hình dưới:



Nếu ta thay đổi nhưng không muốn lưu lại thì ta cũng bấm Commit, sau đó trong cửa sổ Commit chọn **Reset all changes**



***Chú ý rằng, nếu ta đã thực hiện một thay đổi ở một phiên bản nào đó và ta muốn chuyển sang phiên bản khác thì phải Commit hoặc Reset all changes rồi mới chuyển được.***

Khi có nhiều nhánh, ta có thể khôi phục lại nhánh (về phiên bản cuối cùng của nhánh) bằng cách click chuột phải vào phiên bản rồi chọn **Checkout branch.**



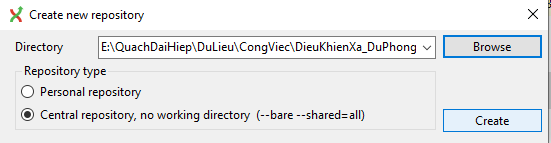
***Khi commit những thay đổi ở phiên bản cuối cùng của nhánh thì sẽ không cần phải tạo nhánh mới mà phiên bản mới sẽ được ghi thêm vào cuối của nhánh đó.***

***3.4. Tạo kho tập trung như phiên bản dự phòng***

Dữ liệu về các phiên bản của Git Extension cũng được lưu trong cùng thư mục với bản thân dữ liệu. Đôi khi ta muốn có một bản dự phòng để phòng trường hợp thư mục bị hỏng (do virus hay ổ cứng hỏng chẳng hạn), hoặc ta muốn tạo một kho chung để mọi người có thể truy cập vào và tải các phiên bản khác nhau về máy họ, thực hiện các thay đổi rồi lưu các phiên bản của họ vào kho chung. Lúc này ta cần tạo kho chung Central repository:

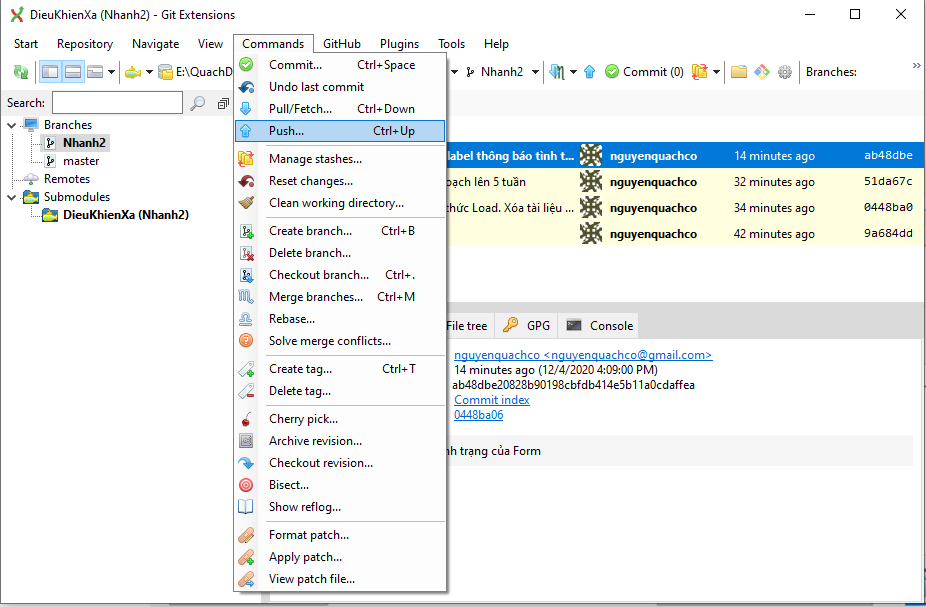
***3.4.1. Tạo kho tập trung***

Quá trình tương tự như tạo kho cá nhân, nhưng ta chọn Repository type là Central repository:

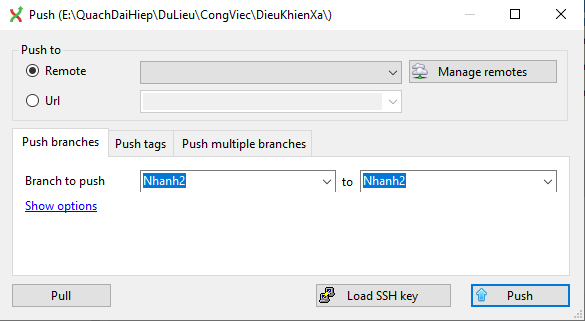


***3.4.2. Đẩy dữ liệu các phiên bản lên kho tập trung***

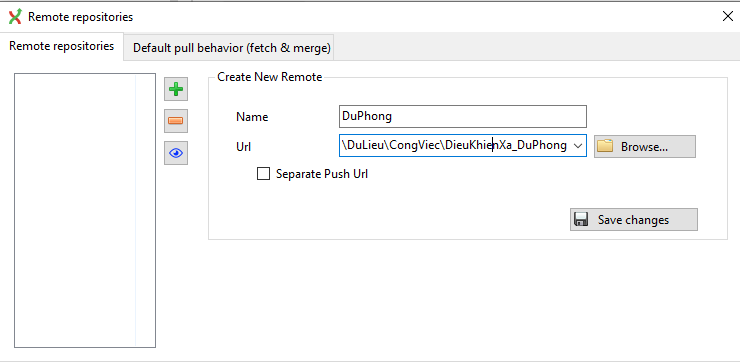
Quay trở lại repository cá nhân của ta, bây giờ ta muốn đẩy dữ liệu lên kho chung. Ta bấm vào nút Push ở thanh công cụ (có biểu tượng mũi tên đi lên màu xanh), hoặc chọn menu Commans\Push.



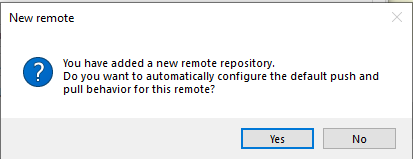
Nếu ta chưa cài đặt kho chung trước đó cho kho cá nhân của ta thì cửa sổ Push sẽ hiện ra:



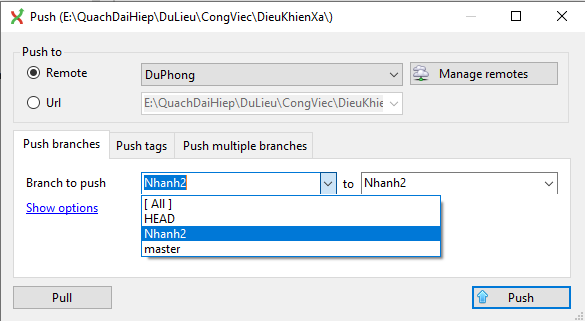
Ta bấm vào Manage remotes để cài đặt kho chung. Cửa sổ Remote repositories hiện ra và ta nhập vào tên của kho (tùy ý) và chỉ đường dẫn đến kho Central repository mà ta đã tạo trước đó. Đường dẫn này có thể đặt ở trên máy tính của ta, hoặc máy chủ nối mạng, hay thậm chí trên Internet (như GitHub).



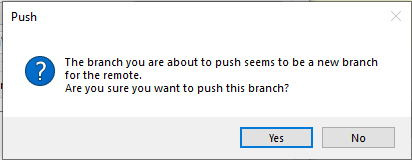
Bấm Save changes và cửa sổ New remote hiện ra, thông báo ta vừa thêm một kho liên kết vào và hỏi ta có muốn thiết lập đây làm kho mặc định để push (lần sau khi bấm vào nút push thì phiên bản sẽ được lưu vào đây và không hỏi lại). Ta có thể bấm Yes để thiết lập đây là kho mặc định.

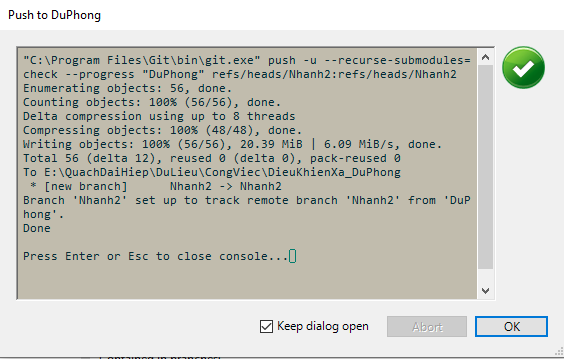


Quay trở lại cửa sổ Push, ô Remote bây giờ đã hiện ra kho mặc định. Ta chọn nhánh cần Push lên (hoặc chọn All nếu muốn push hết) và bấm push. Dữ liệu các phiên bản sẽ được đẩy lên kho tập trung đã chọn.

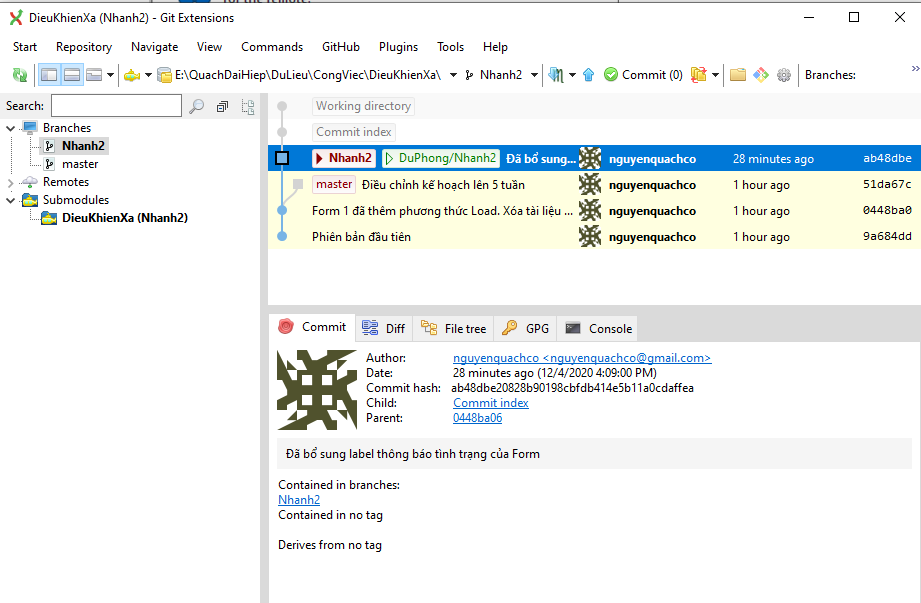


Nếu trong kho tập trung chưa có nhánh ta push lên thì sẽ có thông báo, ta chọn Yes là được.



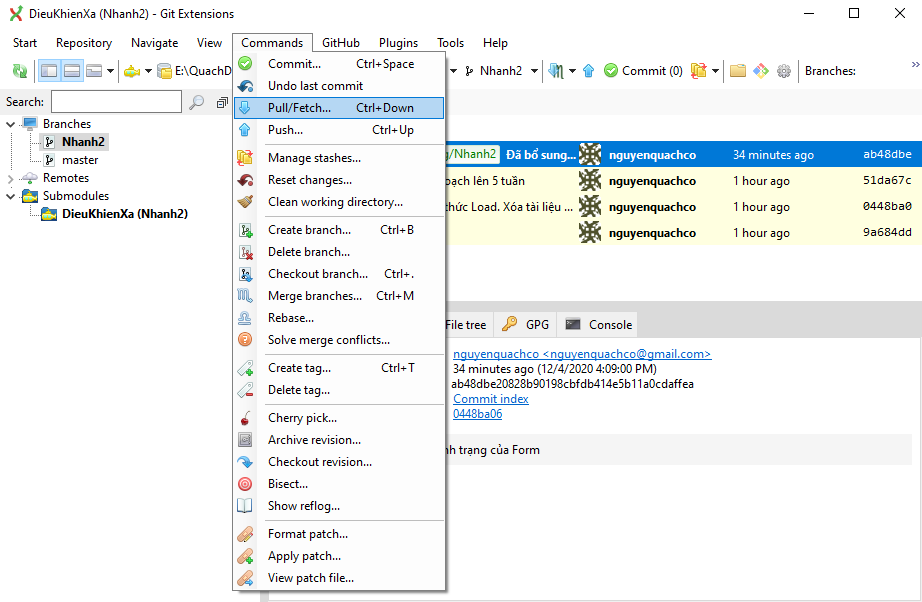


Sau khi push xong, thông báo sẽ hiện ra như hình trên. Quay trở lại cửa sổ Git Extensions sẽ hiển thị các phiên bản đi kèm thêm cả kho dự phòng như hình dưới.

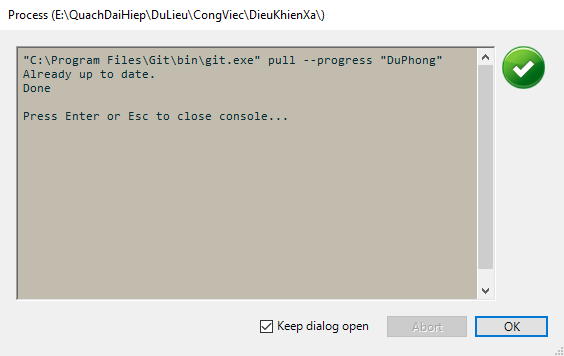


***3.4.3. Tải dữ liệu các phiên bản từ kho tập trung***

Để tải dữ liệu các phiên bản từ kho tập trung, ta bấm vào nút Pull ở thanh công cụ (có mũi tên xuống), hoặc menu Commans\Pull



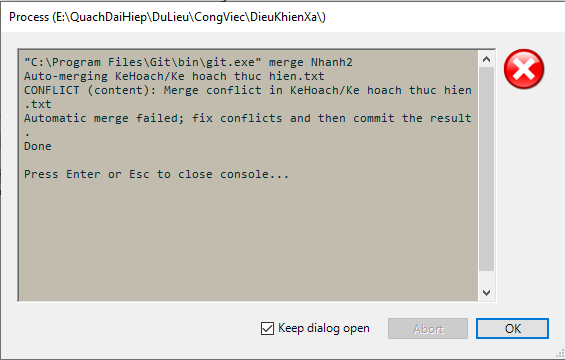
Nếu thành công thì thông báo sẽ được hiển thị như hình:



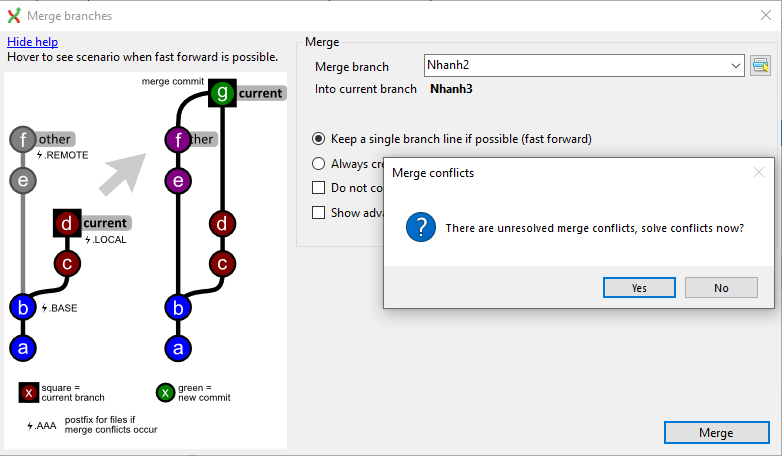
***3.4.4. Giải quyết conflict khi tải phiên bản về***

Khi tải dữ liệu các phiên bản từ trên kho tập trung về, dữ liệu này sẽ được cập nhật vào kho dữ liệu cá nhân trên máy tính của ta. Vấn đề sẽ đơn giản nếu dữ liệu về các nhánh trên kho tập trung và kho cá nhân có thể tương thích với nhau, tức là các nhánh của một trong hai kho là nhánh con của nhánh tương ứng trên kho còn lại. Khi đó, dữ liệu sẽ không cần cập nhật nếu nhánh trên kho tập trung là nhánh con của nhánh trong kho cá nhân, hoặc thêm các phiên bản mới trên kho tập trung vào cuối của nhánh trên kho cá nhân mà nó chưa có. Đây là trường hợp khi mỗi người tham gia đẩy các phiên bản của họ lên kho tập trung ở các nhánh khác nhau, hoặc hai người cũng phát triển một nhánh nhưng không đồng thời (người này phát triển phiên bản mới, đẩy lên kho tập trung rồi người khác mới tải về và phát triển tiếp trên phiên bản mới này, sau đó lại đẩy lên cho người kia, và cứ thế).

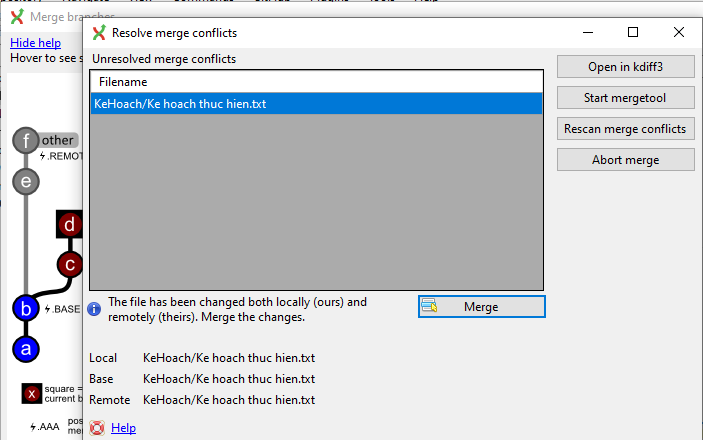
Trường hợp hai người cùng phát triển một nhánh một cách song song, mỗi người thay đổi một số file (thường xảy ra trong thực tế). Và như vậy, ta phải tiến hành trộn 2 nhánh (bản chất là một nhánh nhưng do 2 người phát triển theo 2 cách khác nhau). Nếu mỗi người thay đổi các file khác nhau thì khi trộn vào, các thay đổi sẽ được gộp lại (như vậy, người nào thêm file nào thì file đó sẽ có trong phiên bản sau trộn, xóa file hay thay đổi file nào thì file đó sẽ bị xóa hay thay đổi ở phiên bản sau trộn). Trường hợp hai người thay đổi cùng 1 file thì sẽ xảy ra mâu thuẫn: Git Extension không biết trộn như thế nào cả.



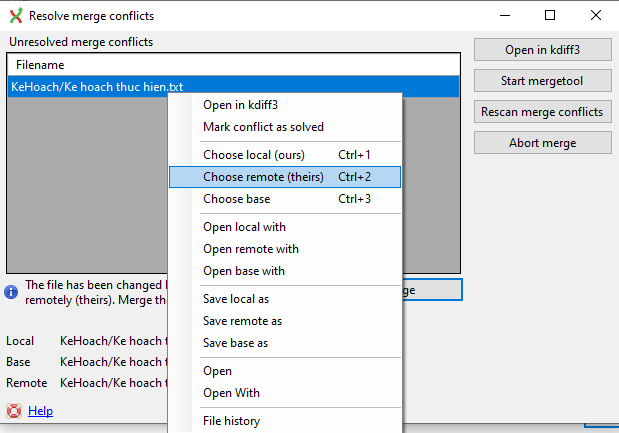
Trường hợp này cần phải giải quyết mâu thuẫn giữa 2 kho:



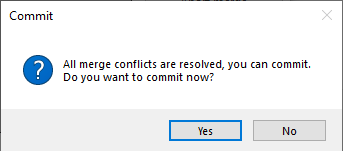
Bấm vào Yes thì cửa sổ Resolve merge conflicts hiện ra, thông báo các file mâu thuẫn:



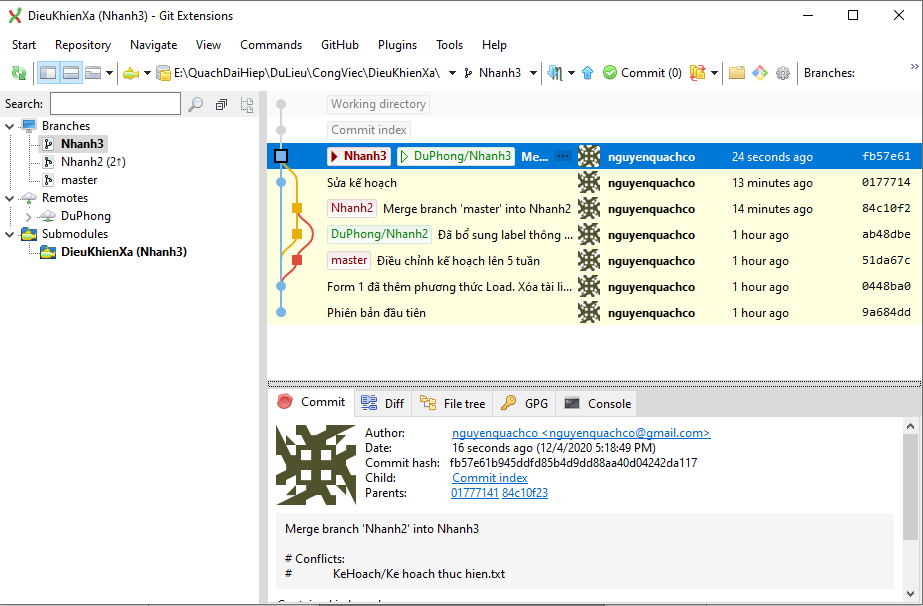
Lúc này, từng file bị confict sẽ được hiển thị để cho phép ta chọn một trong 3 phiên bản: Phiên bản có trong kho cá nhân của ta (current, local hay ours), phiên bản có trong kho tập trung (other, remote hay theirs) và phiên bản chung (là phiên bản chung mà từ đó mỗi người thực hiện thay đổi của mình) để đưa vào phiên bản trộn.



Ta click chuột phải vào từng file và chọn phiên bản muốn giữ lại. Sau khi tất cả mâu thuẫn đã giải quyết xong thì ta có thể Commit được:



Bây giờ trong cửa sổ Git Extensions sẽ hiện ra các phiên bản cùng các đường vòng biểu thị sự gộp của các nhánh như hình vẽ:



**4. Thao tác với tài khoản GitHub**

***4.1. Tạo tài khoản GitHub***

***4.1.1. Tạo 1 email (tự thực hiện)***

***4.1.2. Đăng ký:***

Vài trang github.com và click vào nút Sign up

Chọn username, điền email, mật khẩu và chơi trò chơi Verify rồi bấm vào nút Create Account.

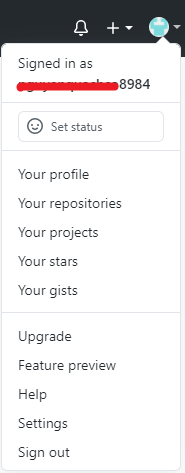
Một email được gửi đến địa chỉ email đã điền để yêu cầu xác thực. Ta bấm vào nút Verify email address trong email gửi đến để xác thực.

Sau đó thực hiện các cài đặt cá nhân như avatar, sở thích, kinh nghiệm, v.v. tùy thích

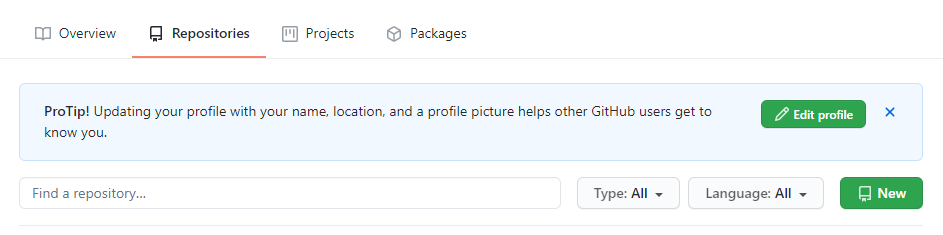
***4.2. Tạo repository trên GitHub***

Repository trên GitHub tương tự như Central repository của Git Extensions, dùng để lưu trữ các phiên bản, khi cần có thể up lên và down xuống.

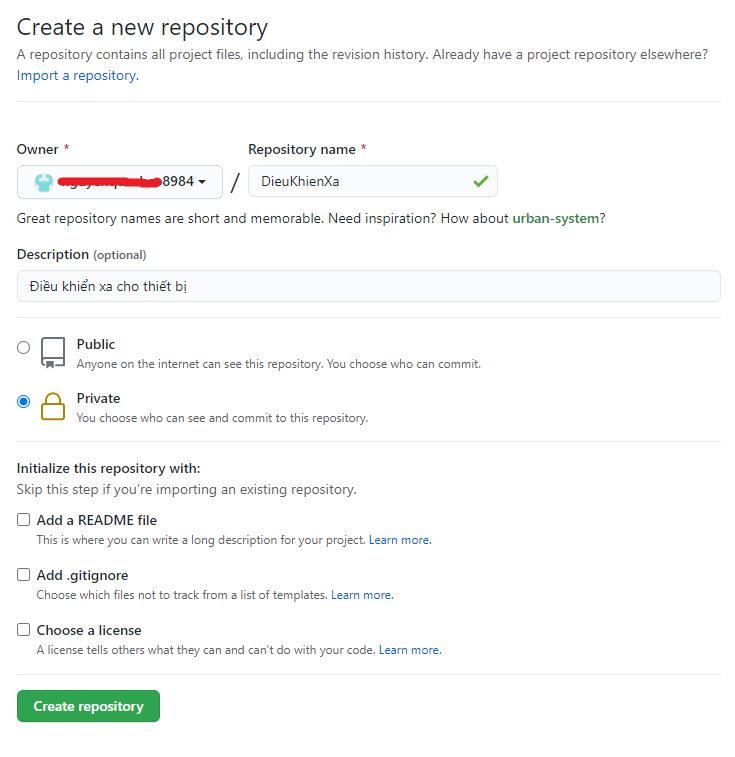
Đăng nhập vào tài khoản GitHub trên trình duyệt, rồi click vào nút bấm tùy chọn tài khoản ở gần biểu tượng avatar bên trái và chọn Your repositories.



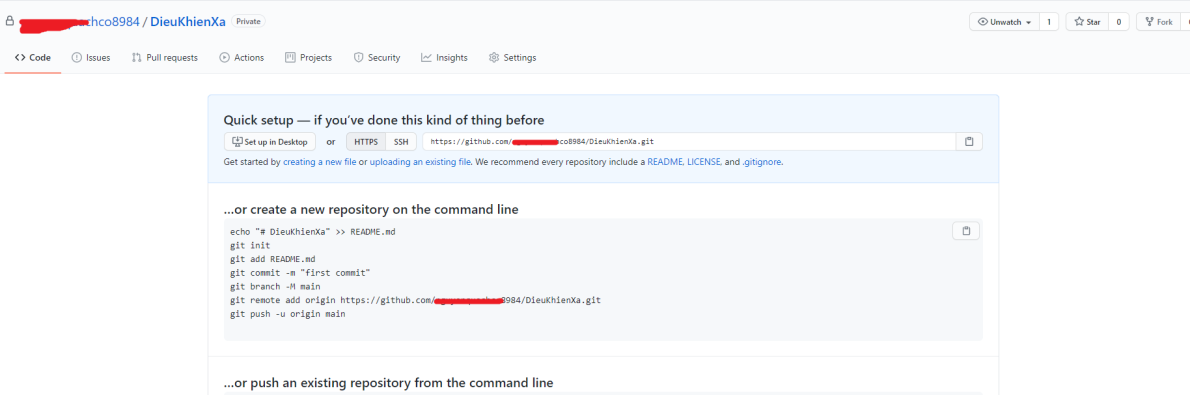
Trong trang repositories chọn New



Trong trang Create a new repository chọn tên của repository (giống như tên thư mục), mô tả, phạm vi cho phép truy cập (nếu repository này có thể truy cập công cộng, tức ai cũng xem được và tải về được thì chọn public, ngược lại nếu chỉ một số ít người được mời truy cập được thì chọn private, thông thường ta chọn private), tùy chọn thêm vào một số file như README, … và bấm vào Create repository:



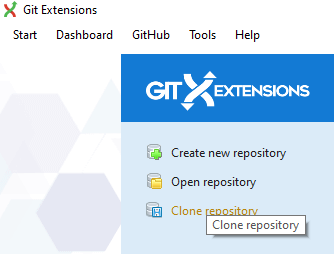
Sau khi tạo xong thì ta được repository. Bây giờ ta có thể đẩy dữ liệu lên đây.



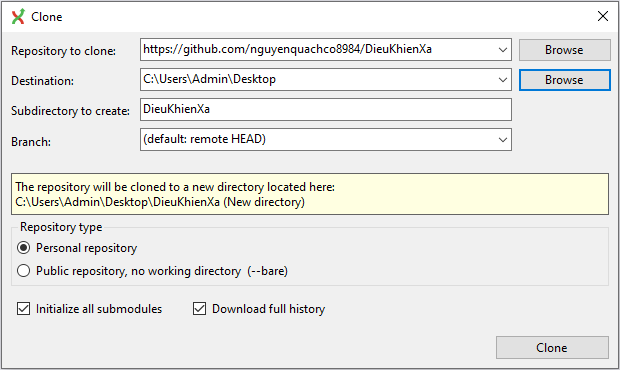
***4.3. Trao đổi dữ liệu với Git Extensions***

***4.3.1. Tải dữ liệu phiên bản từ GitHub về lần đầu***

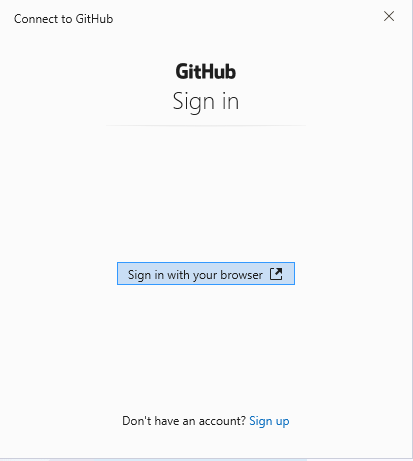
Để tải dữ liệu phiên bản từ GitHub về máy tính của ta khi trên máy chưa có gì (lần đầu tiên), ta tiến hành như sau: Bật Git Extensions lên và bấm vào dòng Clone repository ở cột bên trái hoặc menu Start\Clone repository



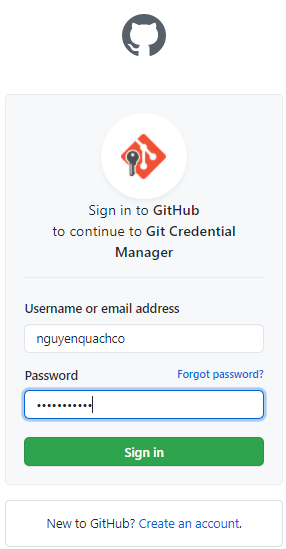
Trong cửa sổ Clone, nhập vào đường dẫn đến repository (bây giờ là trên GitHub, thường có dạng https://github.com/<tên tài khoản>/<tên repository>, rồi chọn nơi đặt thư mục Clone về. Thông thường, Git Extensions sẽ tạo ra một thư mục con có tên trùng với tên của repository tải về. Sau đó bấm vào nút Clone:



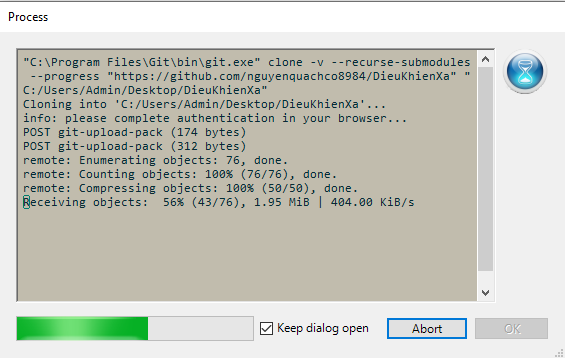
Để tải dữ liệu (private) từ GitHub đòi hỏi phải đăng nhập. Một cửa sổ hiện ra yêu cầu đăng nhập trên trình duyệt. Ta click vào nút Sign in with your browser



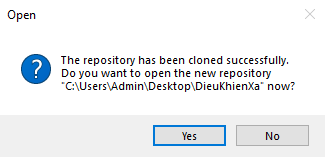
Một trang mới trên trình duyệt sẽ được mở ra cho phép ta đăng nhập. Ta điền tên, mật khẩu và click vào Sign in



Có thể một trang mới thông báo là xác thực một phần mềm Oauth gì gì đó (chỉ có ở lần truy cập đầu tiên), ta click vào nút màu xanh là được. Sau khi xác thực xong (chỉ cần một lần cho một tài khoản) thì Git Extension sẽ tiến hành tải nội dung repository về máy:

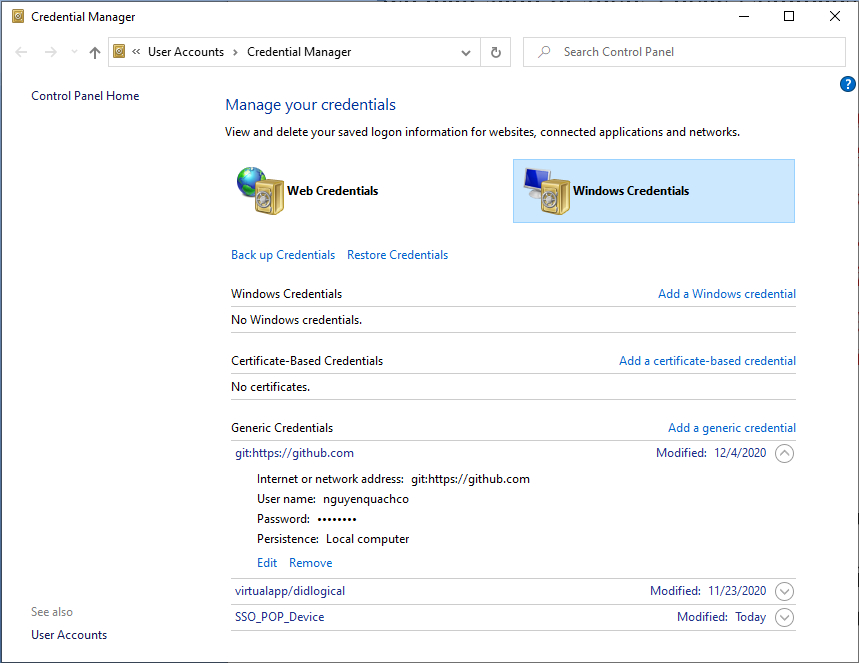


Sau khi tải xong có thể Git Extensions sẽ hỏi là có muốn mở repository vừa tải không. Ta có thể bấm Yes để mở ra và thao tác.



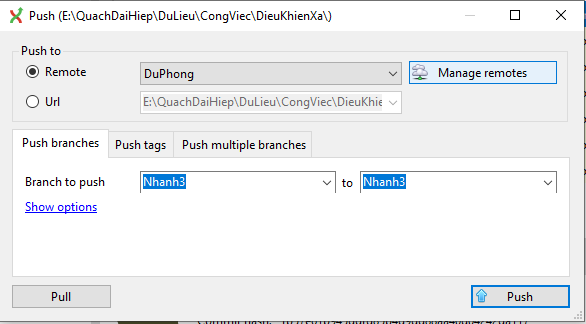
***Chú ý: Khi chuyển đến trang đăng nhập trên trình duyệt mà vốn trên trình duyệt ta đã đăng nhập vào một tài khoản GitHub nào đó thì sẽ có hai trường hợp: Nếu tài khoản GitHub ta đang đăng nhập có quyền truy cập vào repository mà ta cần tải về thì ta không cần phải đăng nhập lại, Git Extensions sẽ tự động quay lại cửa sổ tải về. Trường hợp tài khoản GitHub ta đang đăng nhập không có quyền truy cập vào repository mà ta cần tải về (do repository đó là private mà ta không được mời tham gia) thì Git Extensions có thể sẽ báo lỗi (thường là lỗi Not Found). Lúc này ta phải đăng xuất khỏi tài khoản GitHub đang đăng nhập trên trình duyệt và đăng nhập bằng một tài khoản có quyền truy cập vào repository cần tải.***

***Sau khi đăng nhập một lần, Git Extensions sẽ nhờ Windows Credential Manager lưu tài khoản và mật khẩu GitHub để lần sau sử dụng. Do đó, nếu ta đổi mật khẩu GitHub, Git Extensions sẽ không truy cập được nữa (và cũng không hiển thị cửa sổ đăng nhập nữa). Hoặc khi ta muốn Git Extensions đăng nhập bằng một tài khoản khác cũng vậy. Khi đó ta vào Control Panel\User Account\Credential Manager, chọn mục Windows Credential Manager và tìm đến mục chứa tài khoản của GitHub rồi chọn Remove (để đăng nhập lại) hoặc Edit để cập nhật lại.***

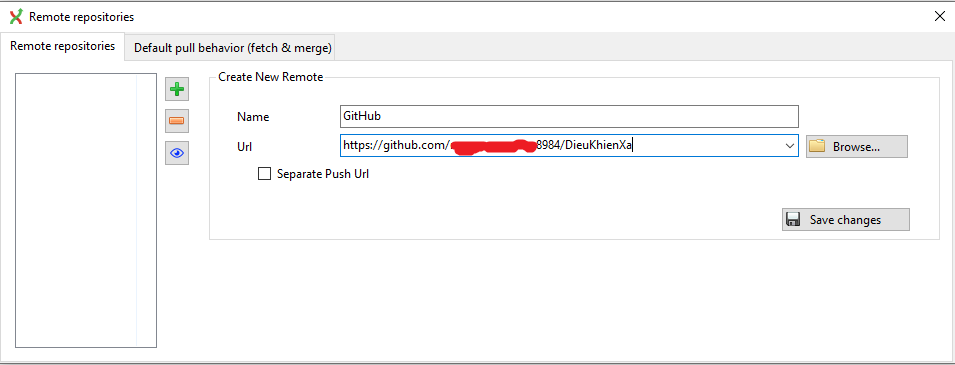


***4.3.2. Đẩy dữ liệu phiên bản của mình lên GitHub***

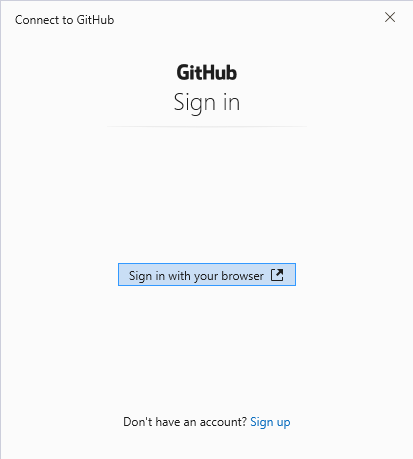
Tương tự như khi trao đổi dữ liệu với Central repository (đã giới thiệu ở bước 3), ta bấm chọn Push (mũi tên màu xanh đi lên) hoặc menu Commans\Push. Cửa sổ Push hiện ra. Ta bấm chọn Manage remotes



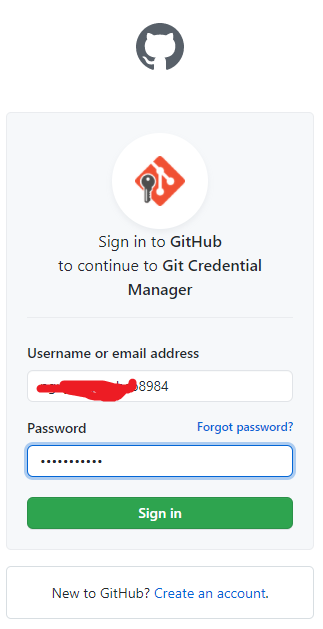
Ta đặt tên cho remote (để hiển thị) và chỉ đường dẫn đến repository (bây giờ là ở trên mạng, đường dẫn thường có dạng https://github.com/<tên tài khoản>/<tên repository>, chẳng hạn:



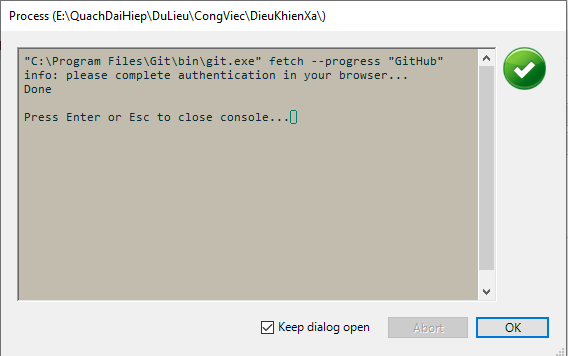
Khác với trên máy tính, khi ta bấm chọn Save changes và thiết lập thành push mặc định, một cửa sổ hiện ra yêu cầu ta đăng nhập vào GitHub trên trình duyệt Web:



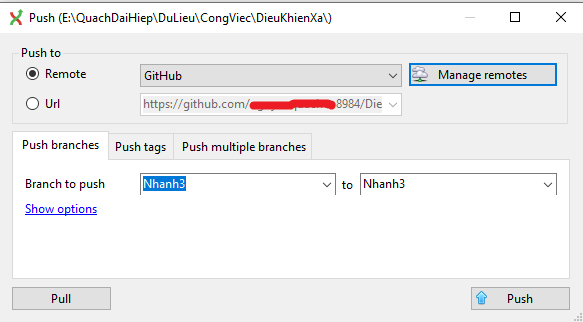
Ta click vào Sign in with your browser. Trình duyệt hiện ra để ta đăng nhập vào GitHub. Ta điền tên tài khoản, mật khẩu và bấm Sign in



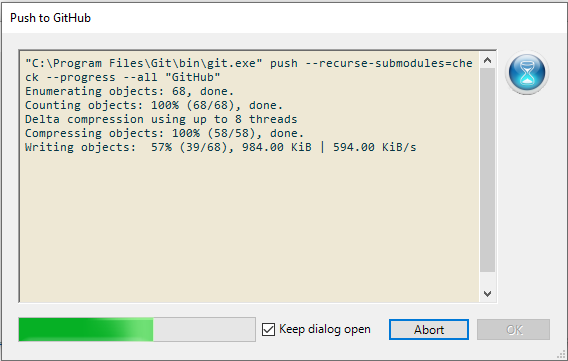
Sau khi Sign in xong thì quay về Git Extensions sẽ thực hiện thiết lập và thông báo kết quả:



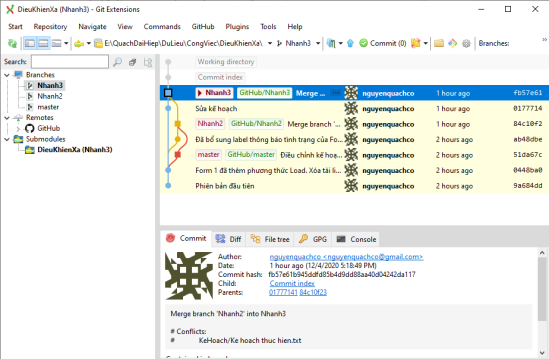
Bây giờ đóng cửa sổ remote repositories và quay về cửa sổ Push. Ta đã có một kho tập trung có địa chỉ trên GitHub. Ta chọn nhánh cần push (hoặc chọn All nếu muốn push tất cả) và bấm Push



Cửa sổ trạng thái hiện ra cho ta biến tiến đô Push

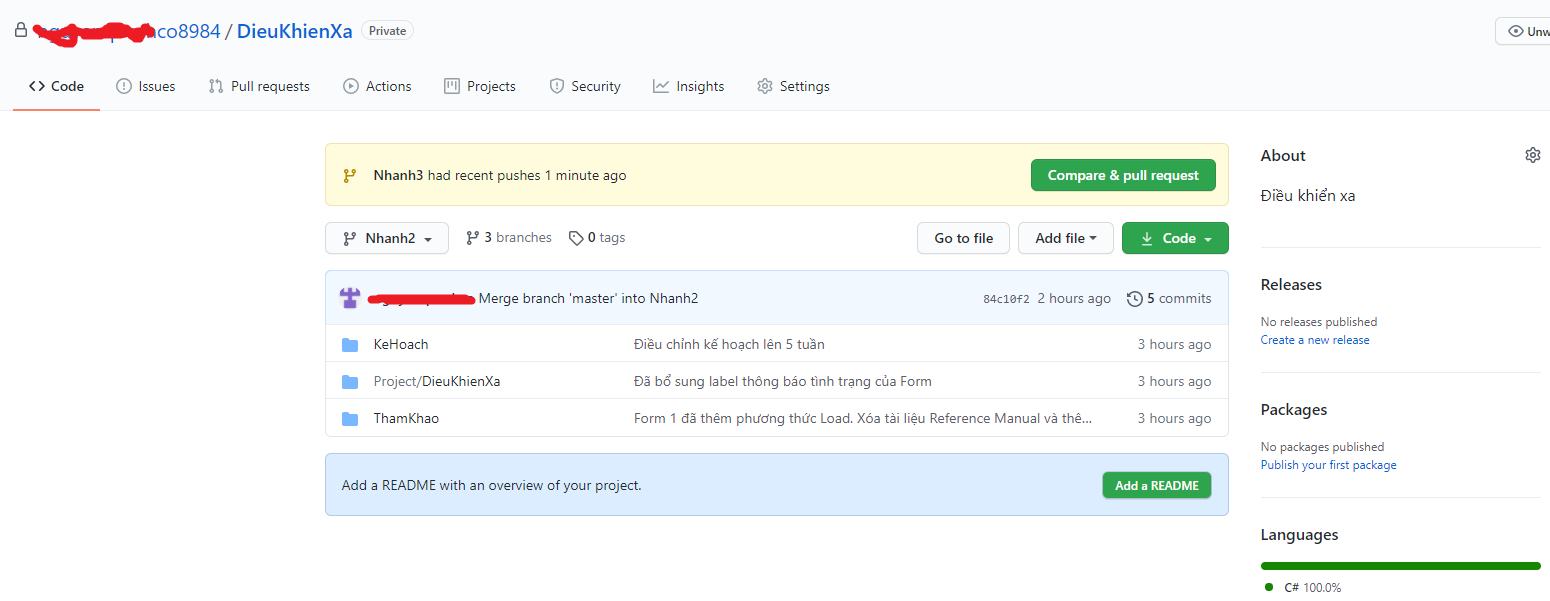


Sau khi Push xong thì quay về Git Extension sẽ hiển thị các phiên bản cùng với tên kho, ở đây ta đã thêm vào kho GitHub.

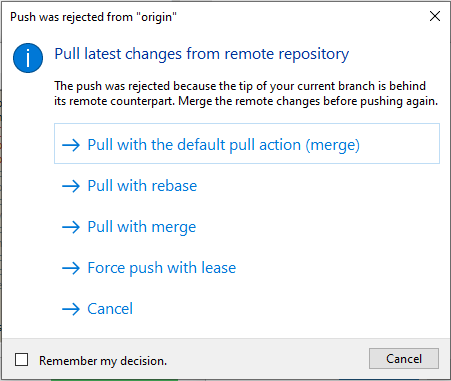


Thao tác đăng nhập trên chỉ cần thực hiện 1 lần khi Push lần đầu tiên. Các lần sau sẽ không cần đăng nhập nữa.

Sau khi push lên thì trong repository của ta ở trên GitHub sẽ được cập nhật dữ liệu mới:

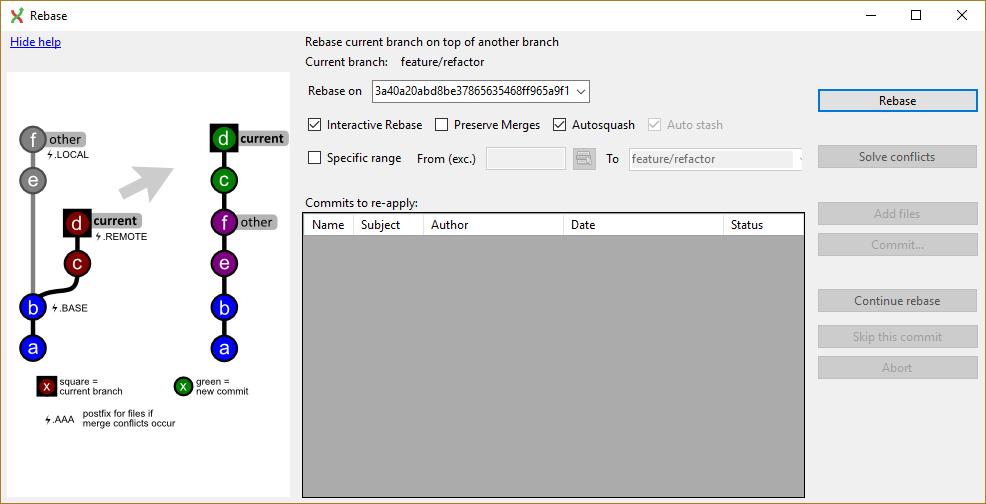


Trường hợp từ khi ta tải dữ liệu các phiên bản lần cuối xuống đến lúc ta đẩy phiên bản mới lên đã có người khác thay đổi dữ liệu các phiên bản trên GitHub thì GitHub đòi hỏi phải tải dữ liệu trên đó về trước, tiến hành các xử lý conflict (nếu có) rồi mới đẩy dữ liệu mới lên được:

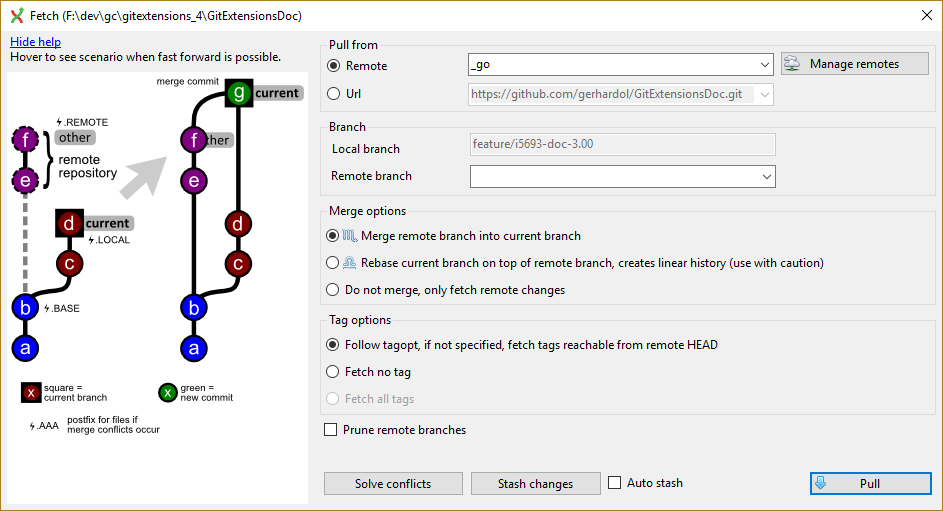
******

Có một số tùy chọn để ghép nối giữa dữ liệu các phiên bản do người khác đẩy lên và dữ liệu phiên bản của ta.

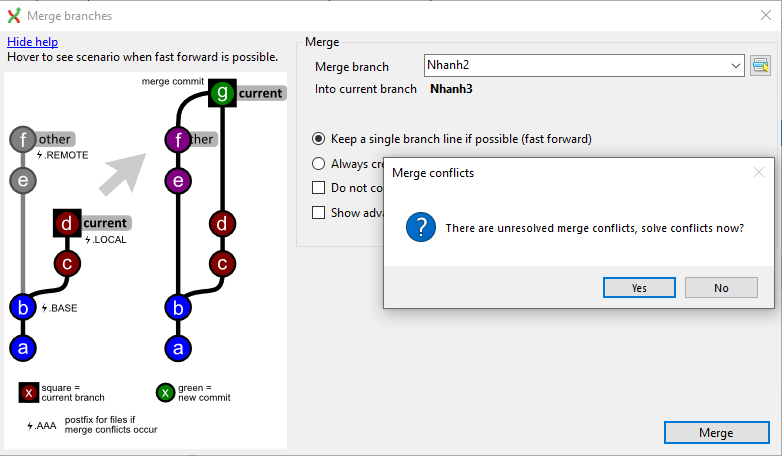
Rebase có nghĩa là đầu tiên tải dữ liệu các phiên bản từ trên GitHub về và coi như ta bắt đầu phát triển từ phiên bản cuối cùng trên đó, các phiên bản của ta khi đẩy lên sẽ được ghép vào sau phiên bản cuối trên đó. Như vậy, phiên bản của ta có thể coi như là phiên bản sau của các phiên bản đã có trên mạng (nếu nhiều người cùng phát triển một cách song song thì phương án này có vẻ không hợp lý lắm, vì sẽ bỏ qua mất những phần đã làm được của những người khác).



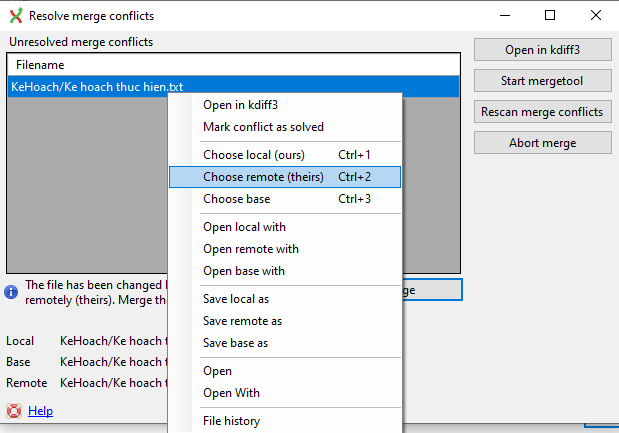
Merge nghĩa là gộp. Phiên bản của ta sẽ được gộp với phiên bản có trên GitHub (và sẽ được tải về trước để gộp với phiên bản hiện có của ta). Hai phiên bản sẽ được gộp với nhau để được phiên bản cuối cùng.



Những thay đổi do hai người thực hiện ở 2 file khác nhau sẽ được gộp chung. Trường hợp 2 người thay đổi cùng một file thì Git Extensions sẽ thông báo là có xung đột (conflict) và đòi hỏi ta phải giải quyết xung đột trước khi gộp.



Xung đột xảy ra khi ở cả 2 người cùng thay đổi cùng một file. Đối với từng file bị xung đột, ta sẽ phải chọn một trong 3 phương án để đưa vào bản gộp chung.

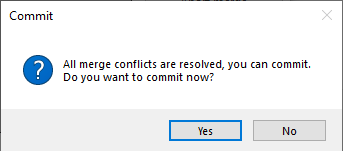


- Chọn phương án local (ours): tức lấy phiên bản của ta (lưu trong máy cá nhân hiện có của ta) để đưa vào bản gộp.

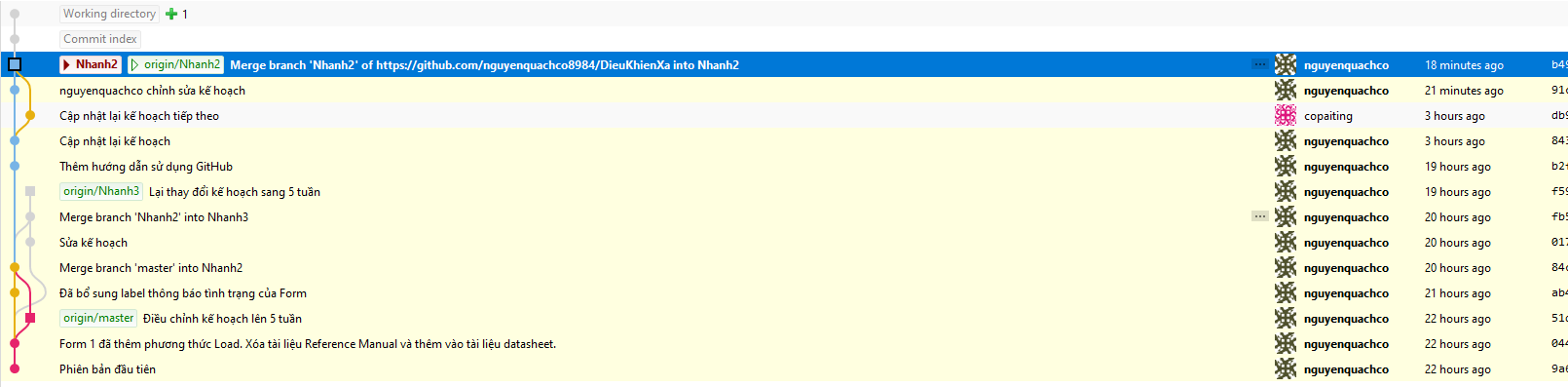
- Chọn phương án remote (theirs): tưc lấy phiên bản sẵn có trên GitHub để đưa vào bản gộp.

- Cọn phương án base, tức phương án chung của hai người mà từ đó mỗi người thay đổi theo một cách riêng để đưa vào bản gộp.

Sau khi giải quyết hết tất cả các file xung đột thì ta có thể đẩy dữ liệu phiên bản mới (phiên bản gộp) lên GitHub.



Lúc này, Git Extensions sẽ hiển thị các phiên bản cùng các đường vòng biểu thị sự gộp của hai phiên bản:

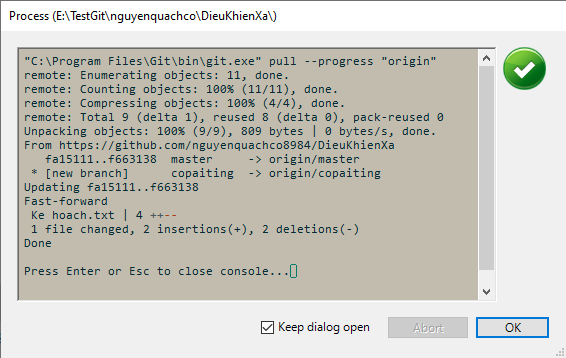


Tất nhiên trong thực tế sẽ có trường hợp phiên bản gộp của 2 phiên bản sẽ không chạy (vì phép gộp này chỉ được tiến hành một cách thô sơ). Lúc này, ta cần chỉnh sửa lại một chút để đảm bảo phiên bản kết hợp giữa 2 phiên bản (những cái giống nhau thì chung, những cái chỉ có ở một trong 2 phiên bản thì sẽ được giữ lại, còn những cái khác nhau ở 2 phiên bản thì sẽ chọn 1 trong 3 trường hợp như trên) sẽ chạy theo đúng yêu cầu của cả hai.

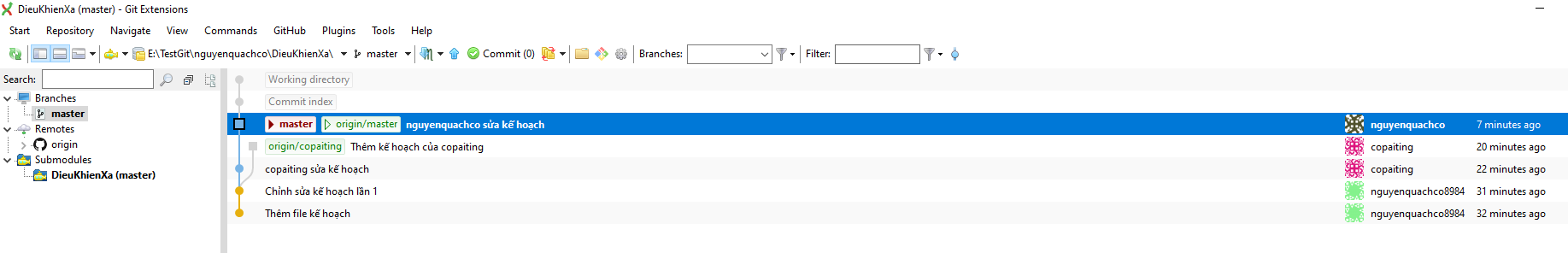
***4.3.3. Tải dữ liệu từ repository khi đã có repository tương ứng trên máy cá nhân***

Khi đã có repository cá nhân tương ứng với repository từ trên GitHub (do ta đã đẩy lên GitHub hoặc đã tải về từ đó) thì thao tác tải dữ liệu về cũng được tiến hành tương tự như trường hợp Central repository trên máy tính. Ta chỉ việc bấm vào nút Pull trên thanh công cụ (dấu mũi tên màu xanh đi xuống) hoặc menu Commans\Pull/Fetch.

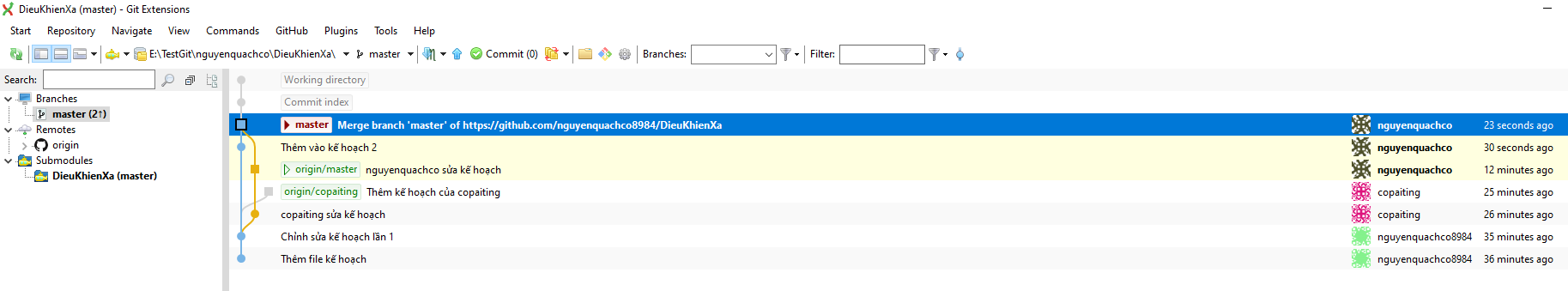
Nếu không có xung đột nào giữa dữ liệu giữa các phiên bản của ta và dữ liệu có trên GitHub (tức là không có thay đổi nào ở cùng một file) thì Git Extension sẽ báo kết quả thành công:



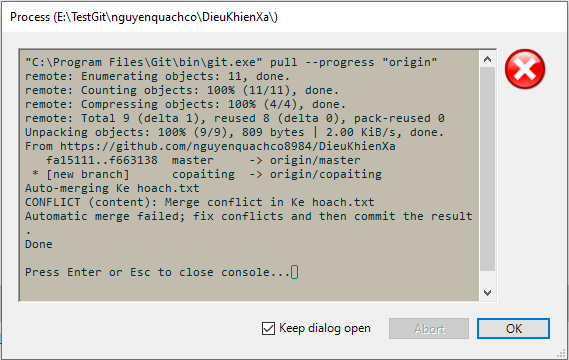
Sau khi tải về thì Git Extension sẽ hiển thị các phiên bản kèm theo các nhánh (có thể có một số nhánh ban đầu không nằm trong kho cá nhân của ta)



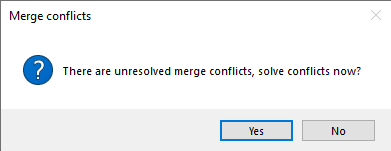
Nếu không có thay đổi nào ở cùng một file (tức không xảy ra xung đột) thì thường kết quả sẽ được trộn, với nhau:



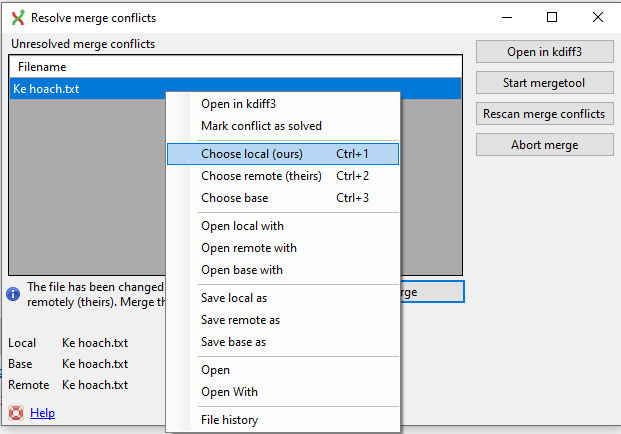
Trường hợp có xung đột, tức có thay đổi nào đó giữa phiên bản ta đang có với những gì được lưu trữ trên GitHub trong cùng một file thì Git Extension sẽ báo lỗi:



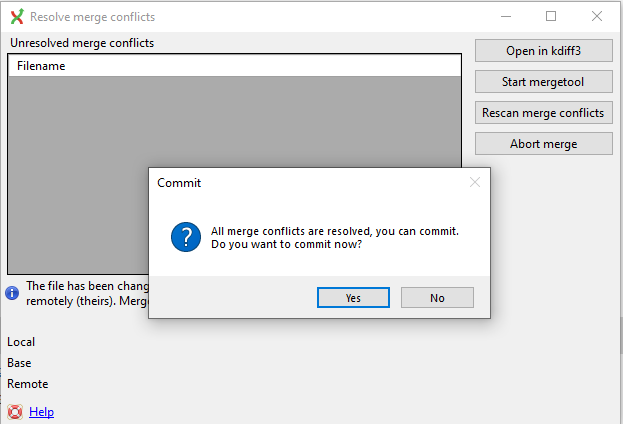
Lúc này ta phải xử lý xung đột:



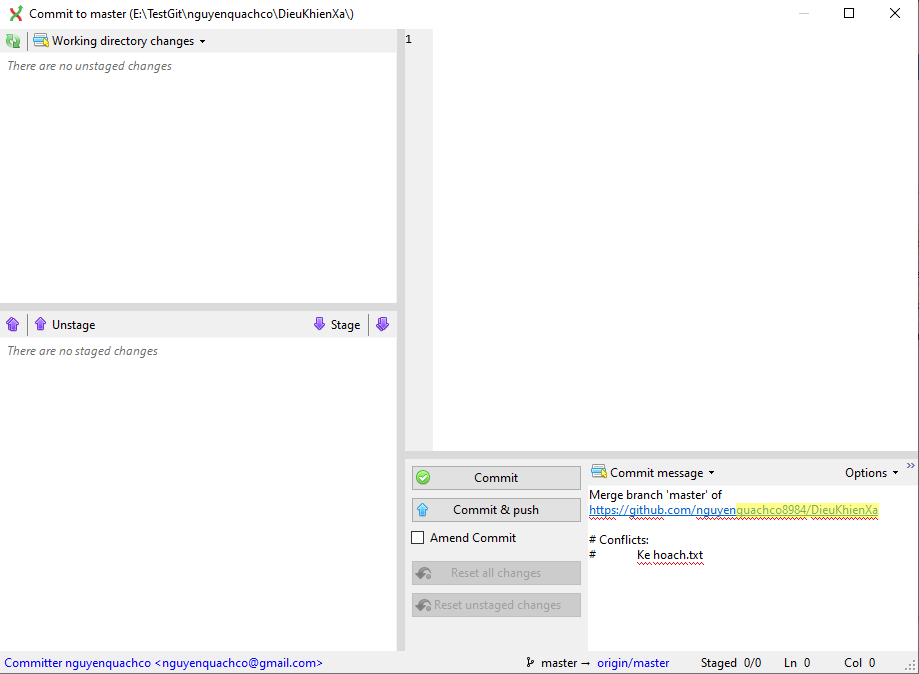
Đối với file xung đột, ta sẽ phải chọn một trong các phương án: ours (giữ lại phiên bản của ta), theirs (bỏ qua phương án của ta, lấy phương án có trên GitHub) hay base (lấy phương án chung mà từ đó ta và trên GitHub thực hiện các thay đổi):



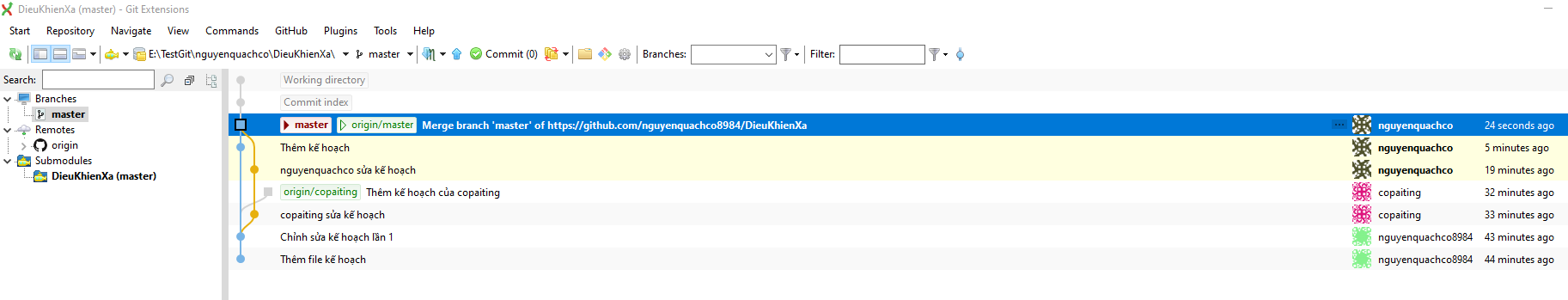
Sau khi xử lý hết các xung đột thì ta có thể tải:



Lúc này Git Extentions sẽ xem như có một phiên bản mới và ta có thể lưu lại:



Quay trở lại màn hình chính, Git Extensions hiển thị danh sách các phiên bản, trong đó phiên bản mới là gộp của 2 nhánh: một nhánh có trên GitHub và một nhánh của ta (thực tế là 2 phiên bản khác nhau của cùng một nhánh).

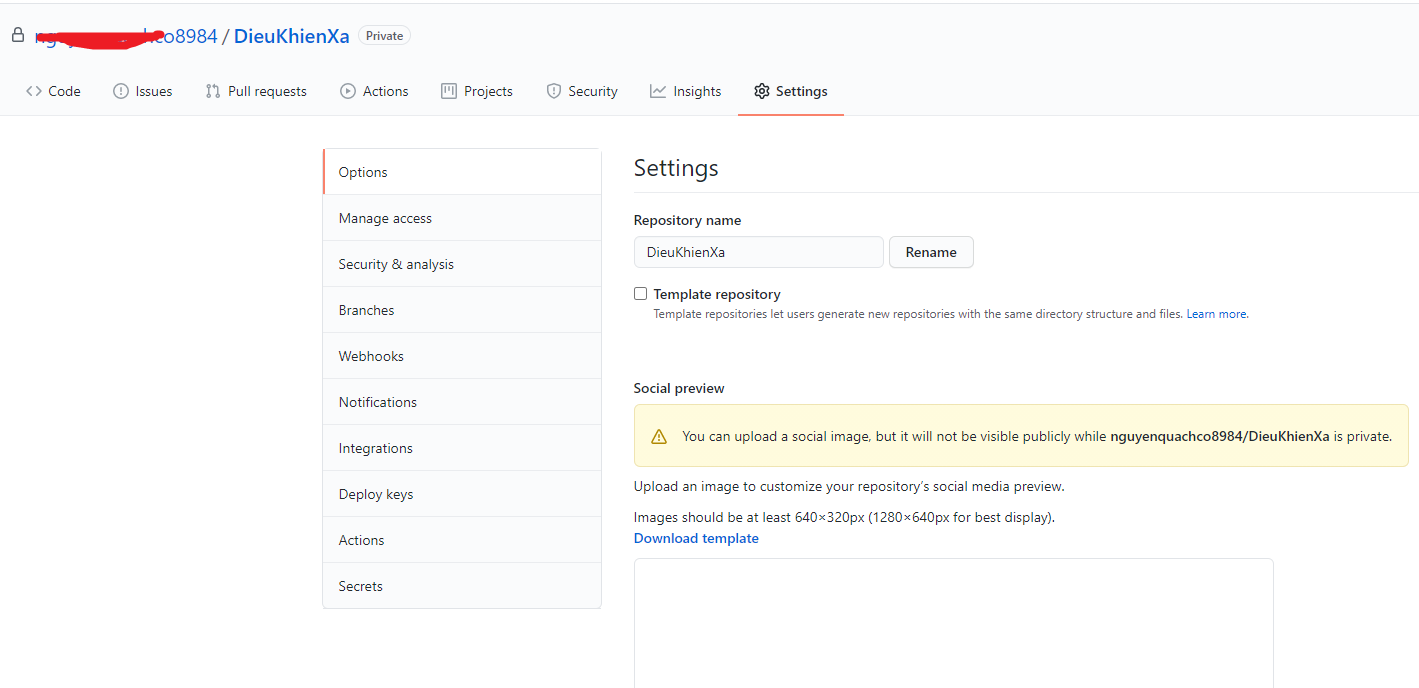


***Chú ý: Nếu tài khoản mà Git Extensions đang đăng nhập không có quyền truy cập repository mà ta đã tải (ta có thể sử dụng Git Extensions để đăng nhập vào nhiều tài khoản khác nhau và tham gia nhiều dự án khác nhau, tuy nhiên tại mỗi thời điểm chỉ có 1 tài khoản được đăng nhập) thì ta phải vào Windows Credential Manager điều chỉnh lại (có thể sẽ phải xóa tài khoản cũ và đăng nhập lại bằng tài khoản có quyền truy cập vào repository cần tải).***

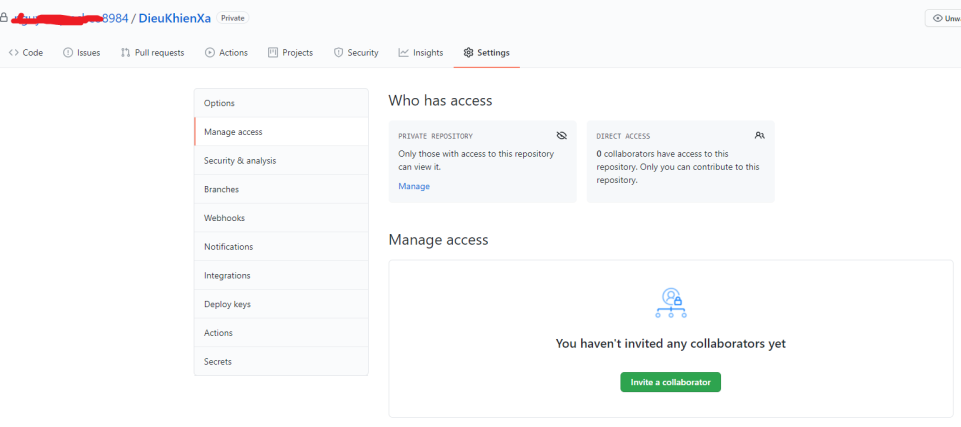
***4.4. Mời người khác tham gia vào repository***

Thông thường ta sử dụng GitHub để nhiều người cùng tham gia phát triển dự án. Để mời người khác tham gia cùng phát triển dự án, ta thực hiện như sau:

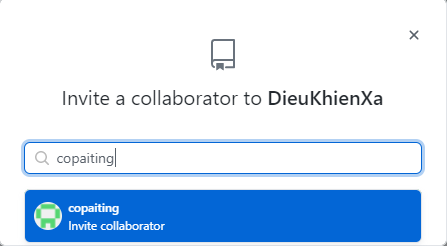
Đăng nhập vào tài khoản GitHub, vào repository cần thiết lập rồi chọn mục Settings:



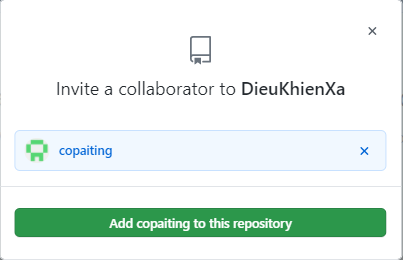
Cột bên trái chọn Manage access rồi trong trang Manage access chọn Invite a collaborator:



Cửa sổ Invite a collatorator to <tên repository> hiện ra cho phép ta gõ tên tài khoản của người cần mời:

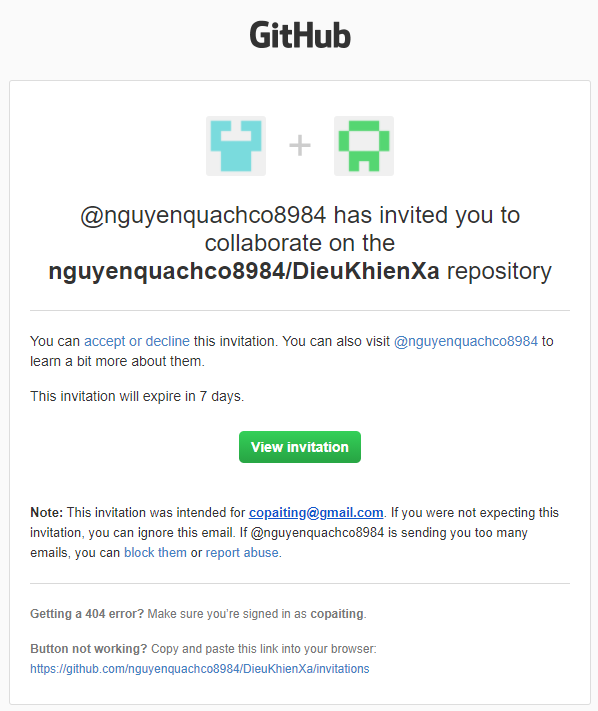


Sau đó bấm Add <tên tài khoản> to this repository:

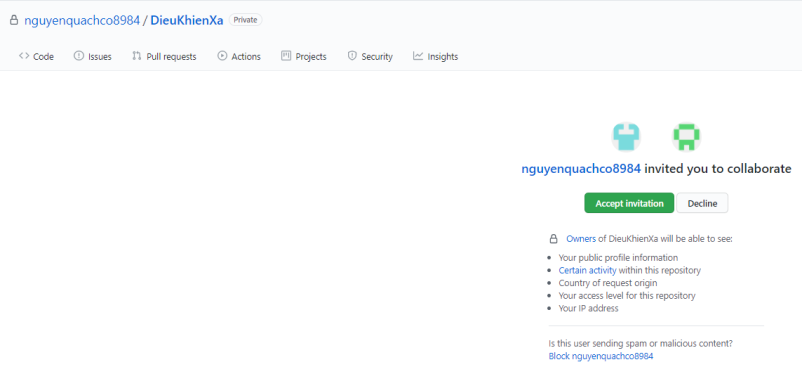


Một email sẽ được gửi đến địa chỉ email của người được mời với nội dung là <tài khoản mời> đã mời bạn tham gia vào <tên repository>

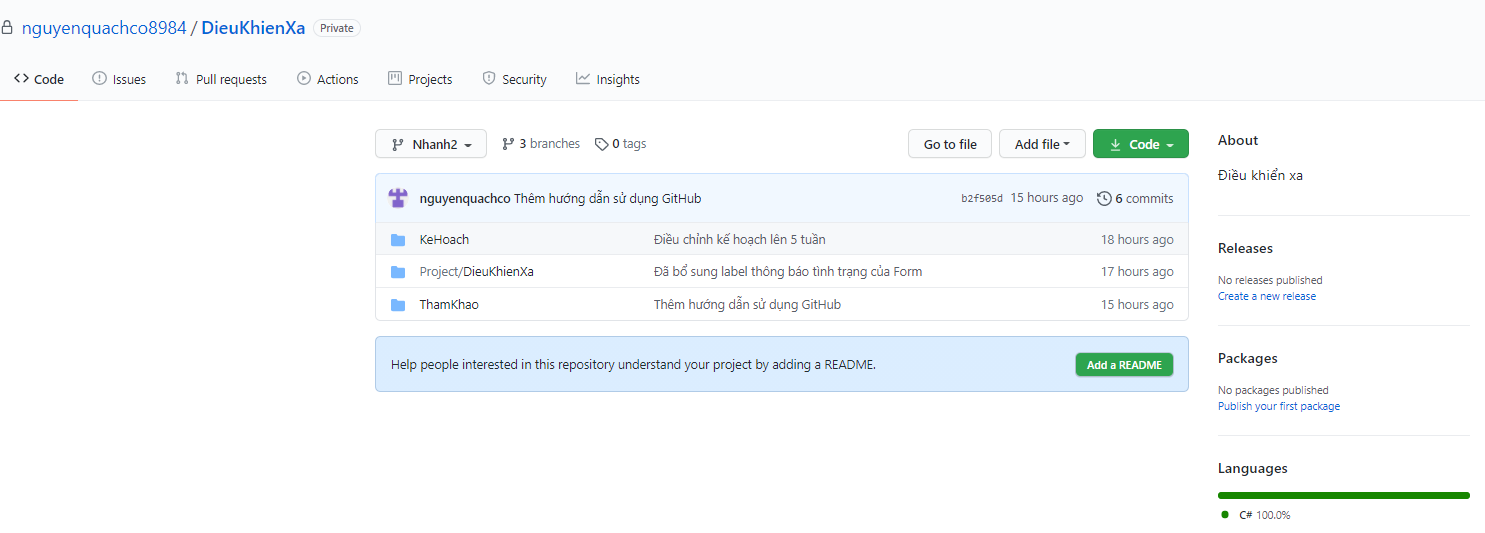
Người được mời vào đầu tiên login vào tài khoản GitHub trên trình duyệt, sau đó bấm vào nút View invitation trong email nhận được (hoặc truy cập đường link ở cuối email nhận được).



Một trang mới sẽ được mở ra thông báo về lời mời. Người được mời bấm nút Accept invitaion để tham gia vào dự án.



Sau khi accept thì tài khoản được mời có thể xem nội dung repository của tài khoản mời mà mình tham gia. Từ đây, người được mời có thể dùng tài khoản của mình để tải dữ liệu các phiên bản về từ repository này, hoặc đẩy phiên bản của mình lên đây (sử dụng Git Extensions chẳng hạn như đã hướng dẫn ở trên).

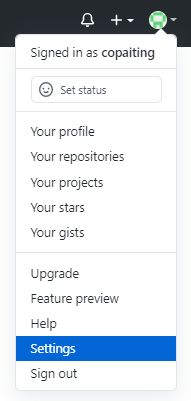


***Chú ý: Nếu trên trình duyệt ta đang đăng nhập vào một tài khoản GitHub khác (không phải tài khoản GitHub được mời) thì khi click vào đường link lời mời trong email, ta sẽ nhận được lỗi 404. Lúc này ta phải đăng xuất khỏi tài khoản GitHub đang đăng nhập, sau đó đăng nhập lại bằng tài khoản GitHub được mời rồi click lại vào được link lời mời trong email để xem lời mời.***

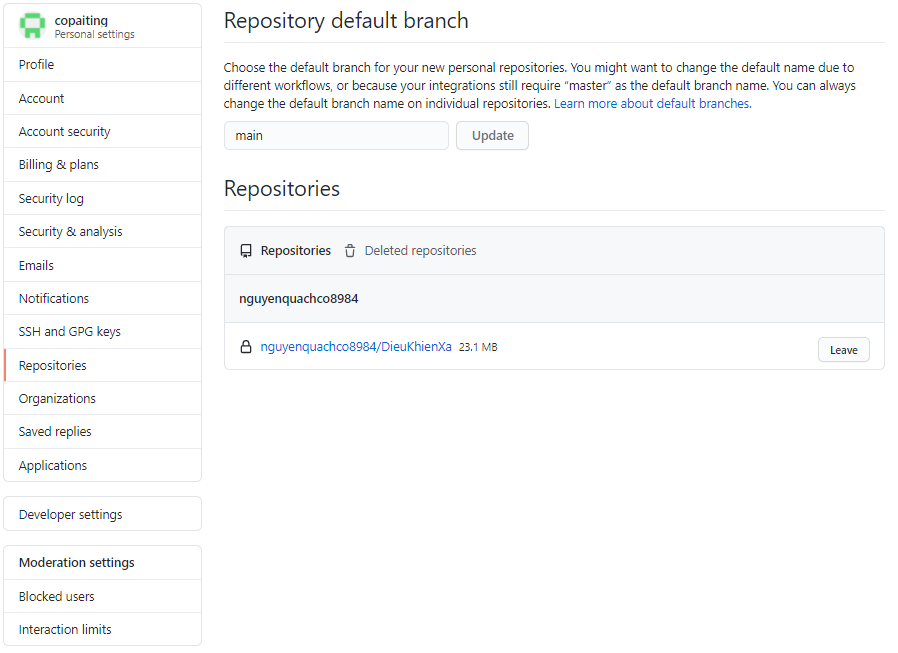
***4.5. Xem các repository đã tham gia, rời khỏi nhóm tham gia dự án***

Ta có thể xem các dự án của mình và các dự án mình được mời tham gia.

Đăng nhập vào tài khoản GitHub, vào repository cần thiết lập rồi chọn mục Settings:

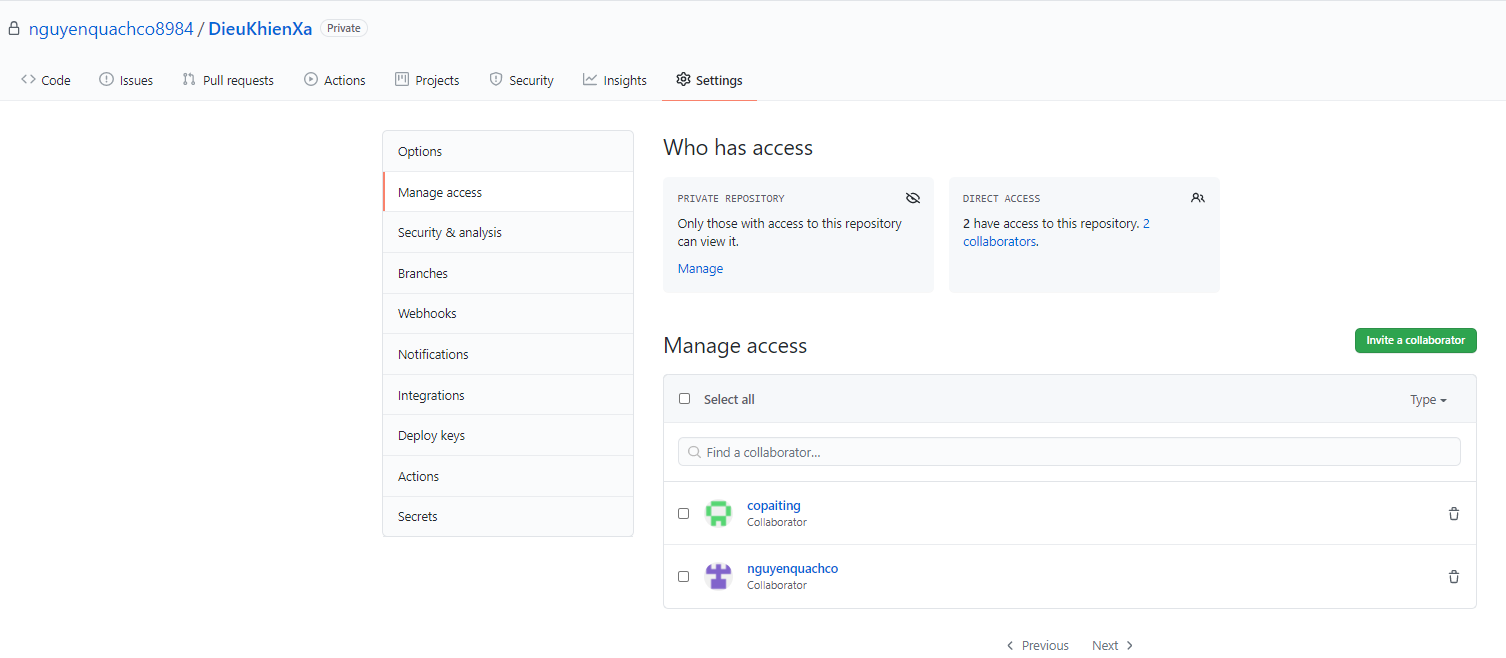


Cột bên trái chọn Repository, bên phải sẽ hiện ra các repository (của mình và mình được mời tham gia). Đối với các repository mình được mời tham gia, ta có thể rời khỏi nhóm dự án này bằng cách bấm vào nút Leave ở cuối dòng thông tin về repository đó.



***4.6. Xem các thành viên tham gia dự án, xóa thành viên tham gia dự án***

Tài khoản đã tạo repository có thể xem các thành viên tham gia dự án và xóa các thành viên tham gia dự án đã mời. Đăng nhập vào tài khoản GitHub và mở repository cần thao tác. Sau đó, chuyển qua tab Settings, chọn mục Manage access (chỗ mà ta vào để mời thành viên mới tham gia). Danh sách các thành viên tham gia vào dự án này sẽ được hiển thị. Để xóa một thành viên nào đó, ta click vào biểu tượng thùng rác ở cuối dòng thông tin về thành viên đó:



**5. Bảo mật dữ liệu Git**

Trong trường hợp kho dữ liệu Git được up lên các trang lưu trữ online như GitHub, ta có thể up lên các repository với thuộc tính private và chỉ có những người được mời tham gia mới nhìn thấy và tải dữ liệu được.

Trường hợp kho dữ liệu Git được lưu trữ ở một máy chủ tập trung, ta có thể cấu hình mật khẩu cho máy chủ để hạn chế khả năng truy cập. Tuy nhiên, về nguyên tắc, admin của máy chủ luôn truy cập được repository của ta và có thể lấy dữ liệu trên đó.

Để tăng cường độ bảo mật nữa, ta có thể sử dụng tính năng filter của Git với các chế độ clean, smudge, merge và textconv để thực hiện mã hóa. Chức năng được dùng để làm sạch dữ liệu trước khi đưa vào lưu trữ. Chức năng smudge dùng để khôi phục dữ liệu sau khi làm sạch. Chức năng textconv được dùng để chuyển đổi file binary thành file text để thực hiện việc so sánh. Git cho phép ta lựa chọn các chương trình ngoài để thực hiện các chức năng này, do đó ta có thể lợi dụng để thực hiện mã hóa. Dữ liệu trước khi push lên được mã hóa bằng clean. Sau khi tải về, dữ liệu được giải mã bằng smudge. Khi so sánh, dữ liệu được chuyển thành dạng text bằng textconv (cũng là giải mã).

Để thực hiện điều đó, đầu tiên ta định nghĩa các thao tác filter clean, filter smudge, diff textconv và merge trong file .git/config, sau đó yêu cầu các chức năng đó trong file .gitattributes, hoặc .git/info/attributes.

Có nhiều hướng dẫn để thực hiện, ta có thể tham khảo trên mạng bằng từ khóa “git transparent encryption”.

Ở đây, ta có thể dùng script transcrypt - <https://github.com/elasticdog/transcrypt> để thực hiện do nó cũng khá dễ thực hiện.

Cách dùng như sau:

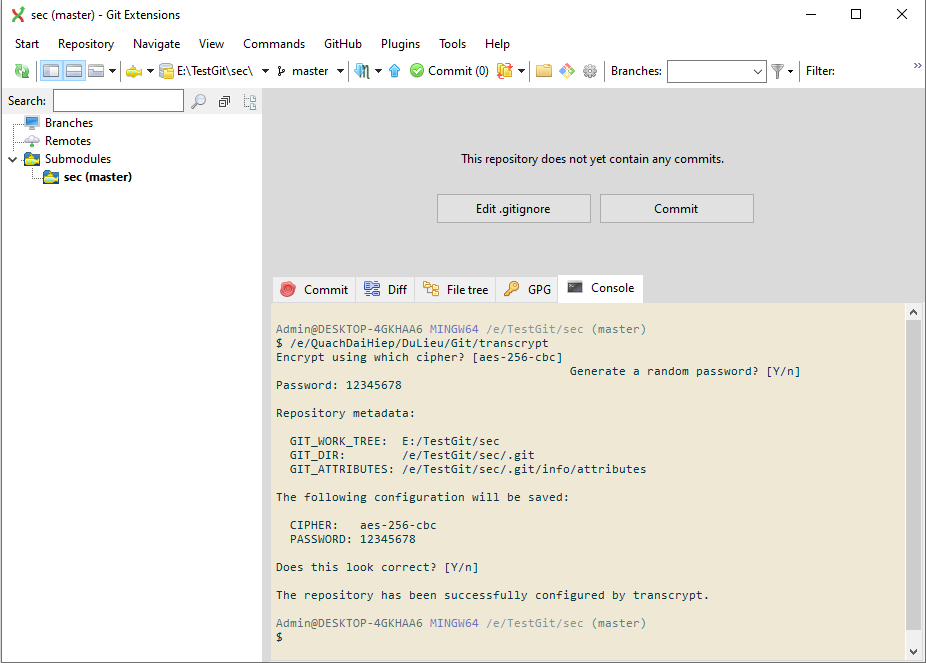
- Bước 0: Download file transcrypt ở trên về và sửa những chỗ cần thiết. Có 2 chỗ nên sửa là: copy dòng 82 paste vào dòng 84 (cả 2 trường hợp ta nên vào file .git/info/attributes), và sửa lại dòng 1139 sau lệnh print thay #pattern bằng dấu \*.

- Bước 1: Tạo 1 personal repository trống (vào Git Extensions và chọn Create New Repository, chọn kiểu persional repository, chỉ đường dẫn và bấm nút Create).

- Bước 2: Mở repository vừa tạo bằng Git Extensions, ở dãy công cụ phía dưới chọn Console.

- Bước 3: Gõ lệnh <đường dẫn đến file transcrypt>/transcrypt. Chú ý là phải gõ cả đường dẫn. Trong đường dẫn, dấu / đầu tiên đại diện cho máy tính, sau đó là ổ đĩa (không có dấu :), rồi đến thư mục cho đến file transcrypt. Nếu trong đường dẫn có dấu cách thì ta phải thêm vào dấu “ ở đầu và cụm đường dẫn và tên file, ví dụ:

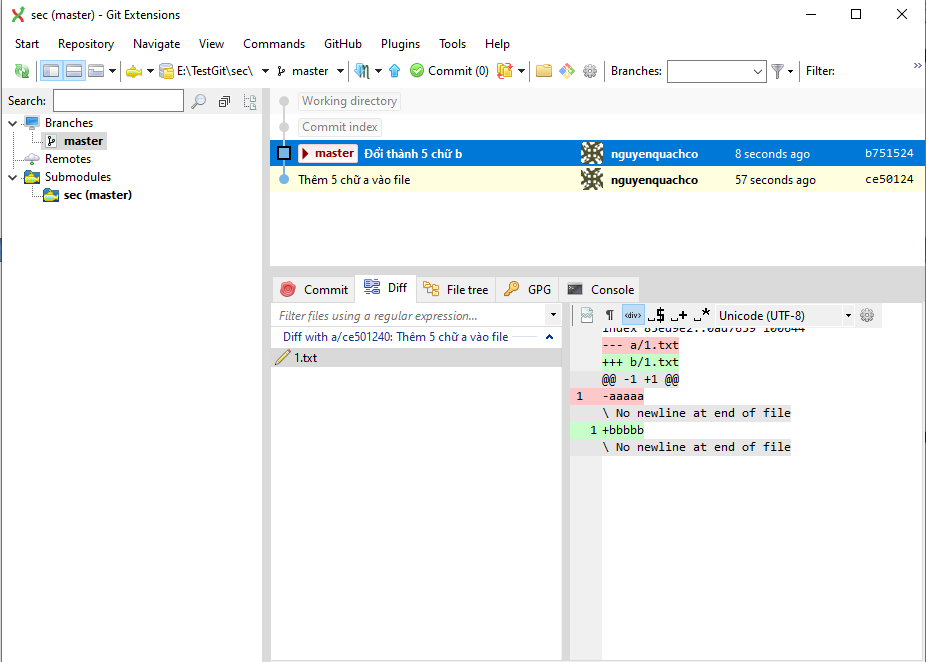
“/D/Lap trinh/Git/transcrypt”



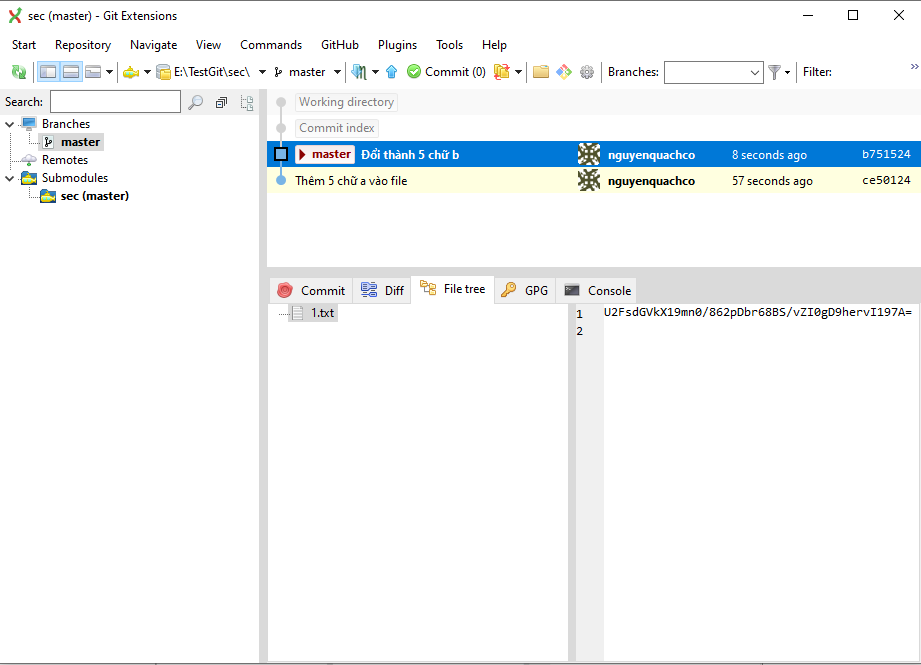
Chương trình sẽ hỏi sử dụng aes-256-cbc, ta bấm enter, sau đó hỏi có tạo mật khẩu ngẫu nhiên không (mật khẩu sẽ được lưu ở file cấu hình config của repository). Nếu ta muốn nhập mật khẩu thì ta bấm n và nhập mật khẩu. Sau đó, chương trình sẽ cấu hình repository và thông báo thành công.

- Bước 4: Khởi động lại Git Extensions và mở lại repository.

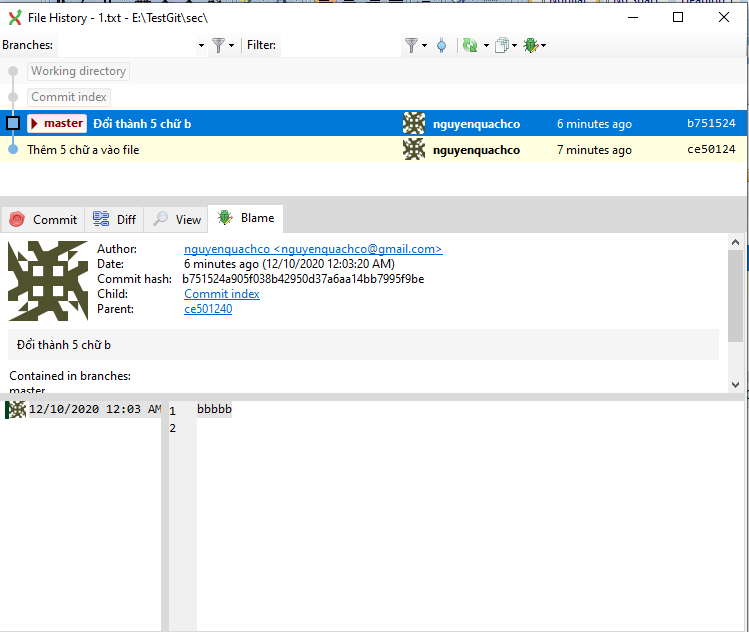
Bây giờ sử dụng giống như repository bình thường. Ta để ý rằng khi chọn 2 phiên bản và so sánh điểm khác nhau thì cửa sổ khác nhau vẫn hiển thị ra đúng nội dung (khác nhau):



Nhưng khi chọn sang tab File tree để xem chi tiết từng file thì không xem được nữa, nội dung file đã được mã hóa (và do trong repository của ta có file .config chứa mật khẩu nên Diff vẫn có thể đọc được, có thể push lên được và tải xuống đúng nội dung).



Để xem nội dung của file ở các phiên bản khác, ta chọn File tree, click đúp vào file để mở File History rồi chọn tab Blame.



Việc up hay kéo repository từ trên mạng (hoặc máy chủ khác) về máy được tiến hành một cách tương tự như khi không có bảo mật.

Để clone 1 repository từ kho tập trung có bảo mật, ta phải biết được mật khẩu. Sau đó, tiến hành clone như bình thường. Sau khi clone xong thì bản thân các file trong project ban đầu sẽ bị sai do ta chưa cấu hình mật khẩu. Bây giờ ta mở repository vừa clone, chọn tab Console và chạy file transcrypt tương tự như khi ta cấu hình mã hóa ở trên máy cá nhân. Lần này ta không thể tạo mật khẩu ngẫu nhiên mà nhập vào mật khẩu đã dùng để mã hóa repository. Sau khi transcrypt cấu hình repository xong, ta khởi động lại Git Extensions và vào thư mục làm việc, xóa hết tất cả các file và thư mục (chỉ trừ lại thư mục .git và các file cấu hình của git nếu có). Git Extensions sẽ báo có nhiều thay đổi, ta tiến hành commit và chọn Reset All Changes để cập nhật lại nội dung trong thư mục làm việc.