

Giới Thiệu

Tôi có cơ hội biết và sử dụng Git khi tham gia khóa đào tạo học viên tại Công ty GMO Runsystem. Sau thời gian tìm hiểu và sử dụng, tôi đã thấy được lợi ích của việc sử dụng Git trong việc quản lý phiên bản. Qua bài viết này, tôi mong có thêm được một chút đóng góp nhỏ cho nhu cầu quản lý source code của các bạn.

Như các bạn đã biết, đối với các dự án phát triển phần mềm lớn nhỏ, việc quản lý mã nguồn theo cách thông thường sẽ mất nhiều thời gian, chi phí và kém hiệu quả. Các thành viên trong nhóm phát triển sẽ phải tìm một cách nào đó để phiên bản làm việc trên máy tính cá nhân của mình luôn là mới nhất. Khi dự án có sự thay đổi, việc phục hồi lại thời điểm cần thiết là khá khó khăn. Do đó, chúng ta cần một phương pháp nào đó có thể khắc phục được các nhược điểm kể trên. Một trong những phương pháp hiệu quả nhất là sử dụng một hệ thống **Version Control System** (VCS) mà tiêu biểu là **Git**. Sau đây chúng ta cùng làm quen với Git và các command cơ bản của Git.

Git là gì¹

Git là một VCS (Hệ thống quản lý phiên bản – Version Control System), là thế hệ mới nhất trong hệ thống các VCS, dùng để quản lý và kiểm tra phiên bản phân tán² khác nhau trong quá trình lưu trữ và phát triển mã nguồn, được phát triển bởi Linus Torvalds dành cho việc phát triển Linux kernel. Git là phần mềm mã mở được phân phối theo giấy phép công GPL2. Git có khả năng hoạt động đa nền tảng³, có sẵn cho các nền tảng Linux, Windows, Mac OSX, ...

Tại sao nên sử dụng Git

So sánh Git và các phương pháp quản lý dự án thông thường, lẽ dĩ nhiên Git là lựa chọn hoàn hảo. Tuy nhiên Git còn vượt trội hơn một số phần mềm quản lý khác nhờ những ưu điểm sau:

- Dễ sử dụng và an toàn vì mỗi bản copy của thành viên đều là full copy từ repository gốc, khi server bị down.
- Mô hình phân chia branch (nhánh) giúp cho công việc dễ dàng được chia nhỏ và hoàn thành độc lập.
- Có khả năng làm việc ngoại tuyến. Người sử dụng có thể sao lưu các thay đổi, commit trên kho lưu trữ cục bộ (Local repository) mà không cần đến kết nối Internet. Các thay đổi này sẽ được đưa thủ công lên Remote repository khi Internet hoạt động.

¹ Xem : <http://git-scm.com/book/en/v2>

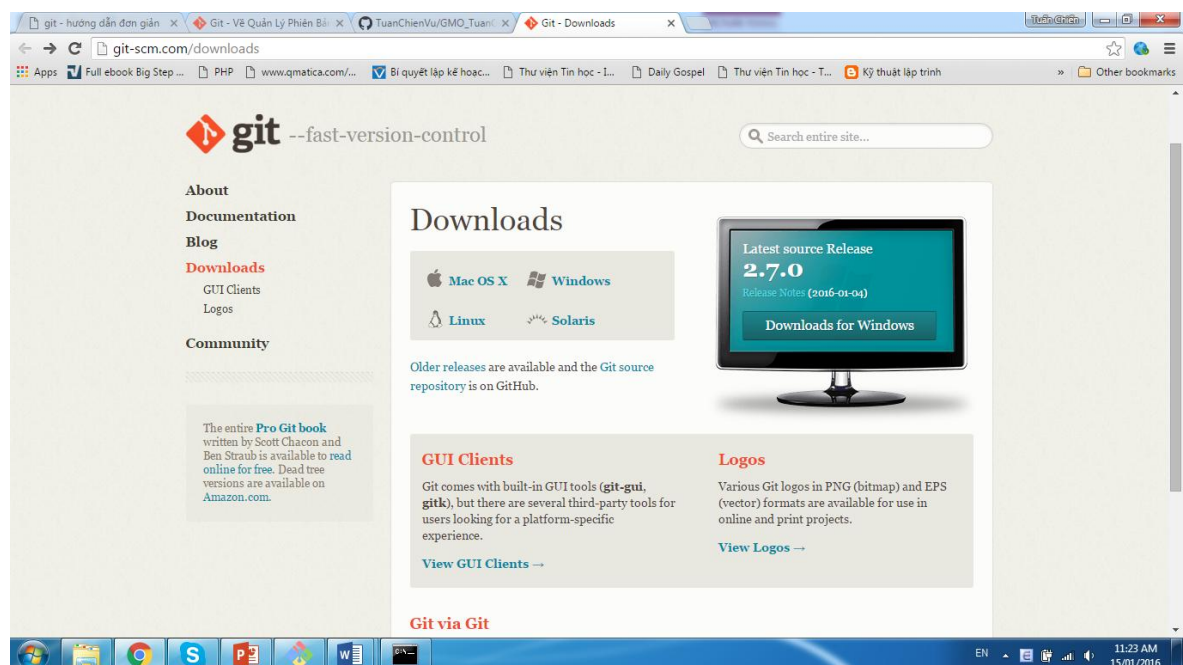
² Xem : Phụ lục về phân tán

³ Xem : <http://git-scm.com/downloads>

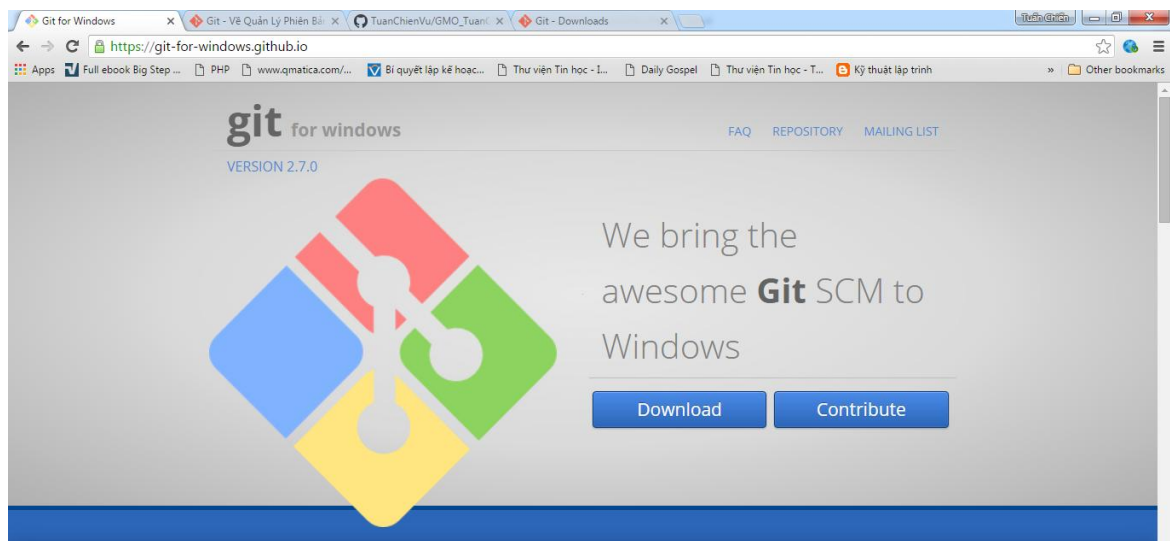
Dĩ nhiên, với khả năng như vậy thì việc làm quen sử dụng Git sẽ phức tạp hơn so với các công cụ khác. Tuy nhiên với lợi ích mà nó mang lại thì việc học cách sử dụng thành thạo Git để áp dụng vào công việc là hoàn toàn xứng đáng.

Cài đặt Git

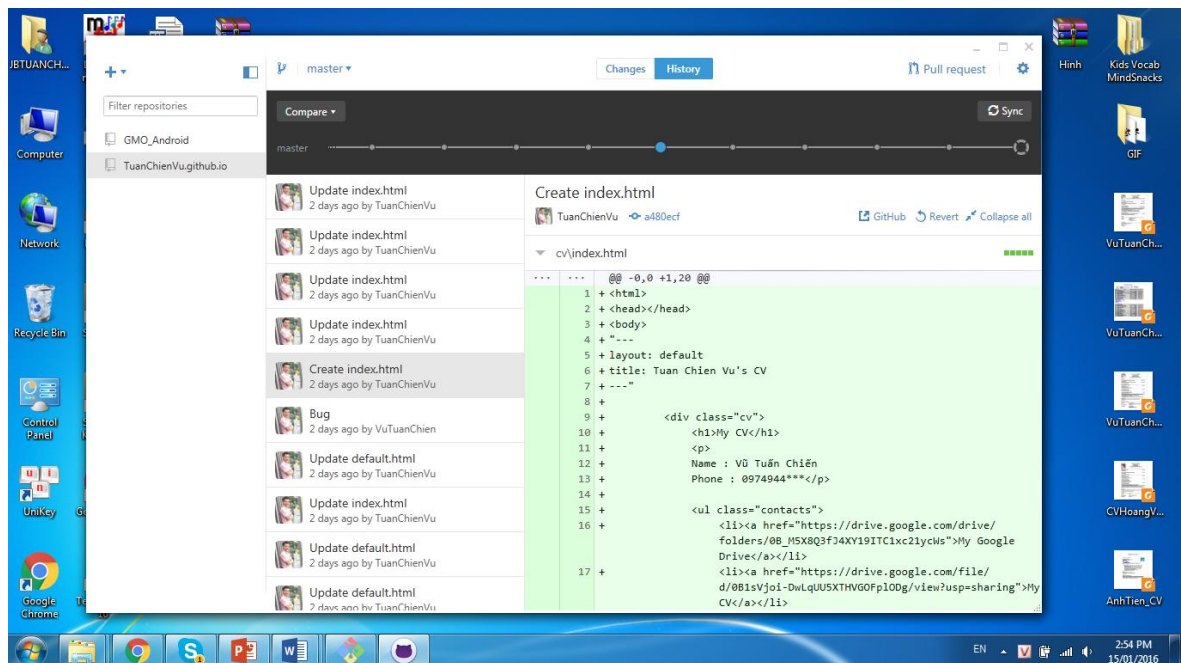
Các bạn có thể tải về Git thông qua trang chủ chính thức của Git : <https://git-scm.com/downloads>



Trong bài viết, tôi sử dụng phiên bản Git dành cho hệ điều hành Windows bạn có thể tải tại: <https://git-for-windows.github.io/>

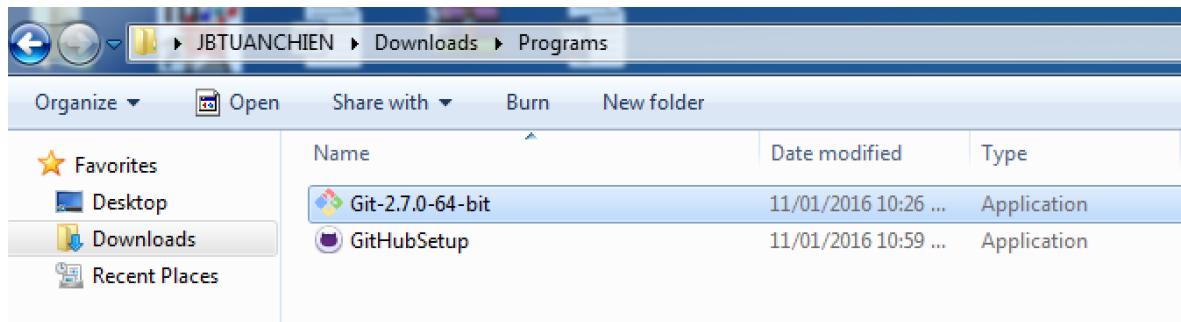


Ngoài ra, có một số công cụ làm việc tương tự như Git, nhưng sử dụng giao diện đồ họa trực quan trong các thao tác là **Github**, **TortoiseGit** các bạn có thể tìm hiểu thêm

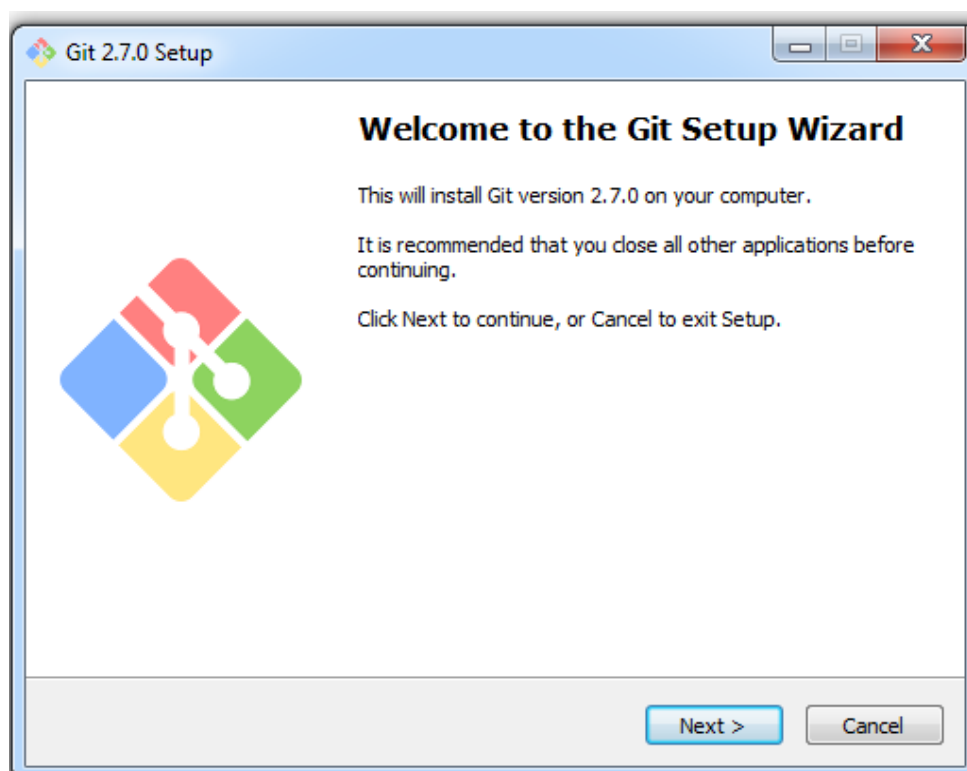


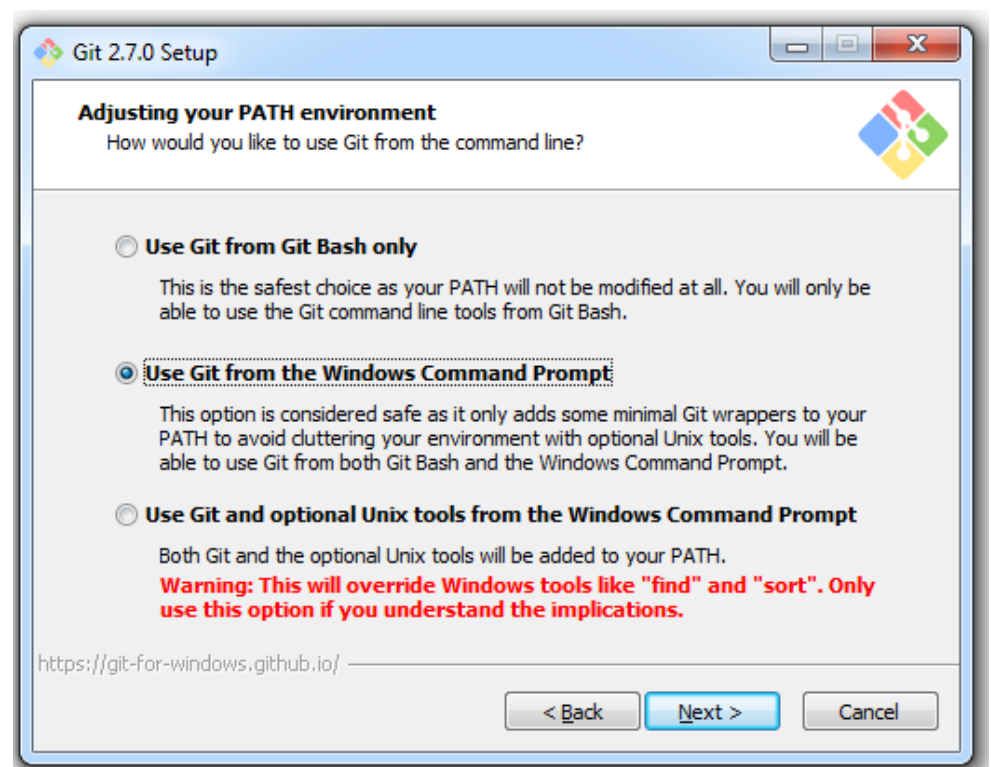
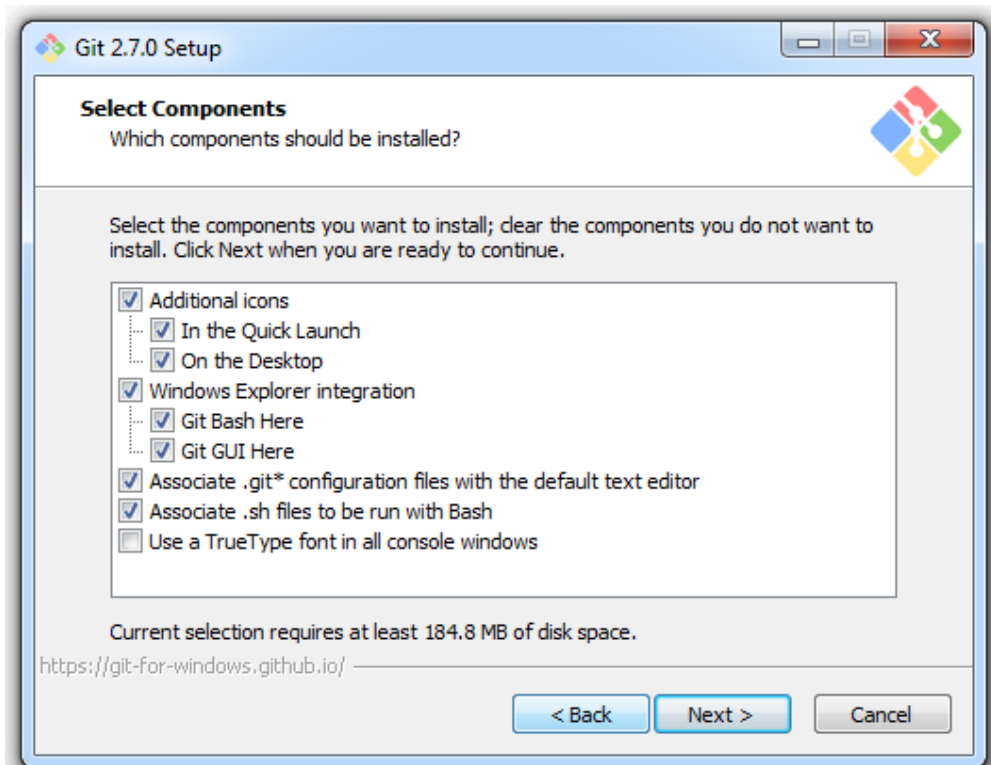
Trong bài viết, tôi sẽ sử dụng Git Bash - Command line để hướng dẫn các thao tác, việc cài đặt Git diễn ra nhanh chóng và tương tự như cài đặt các phần mềm hiện nay.

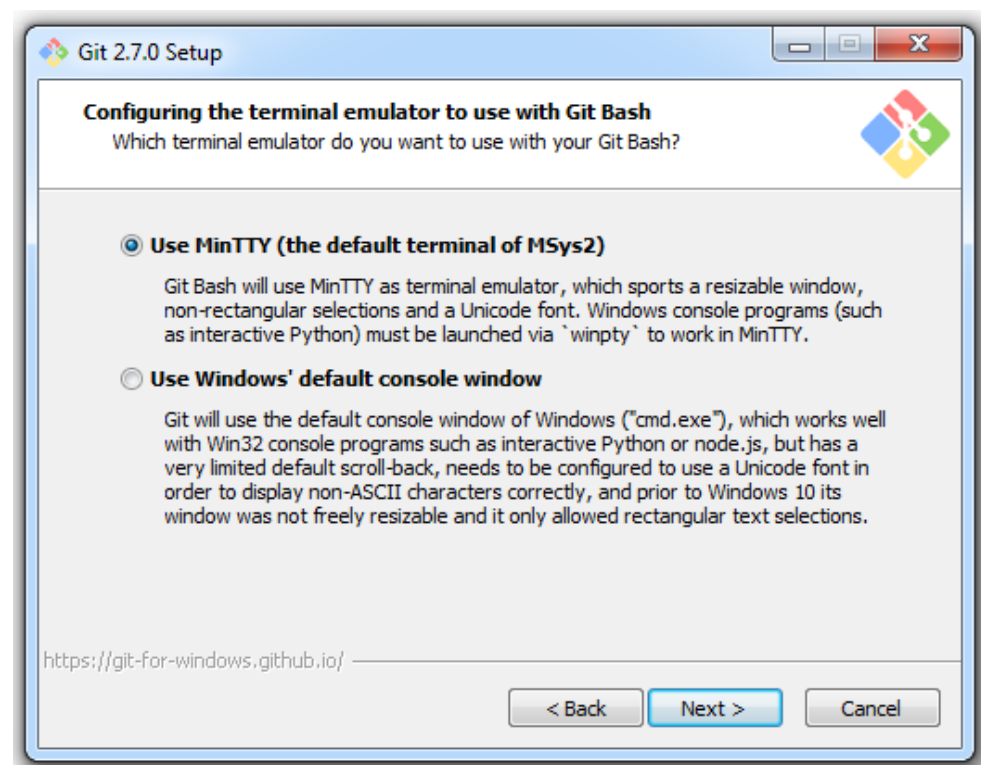
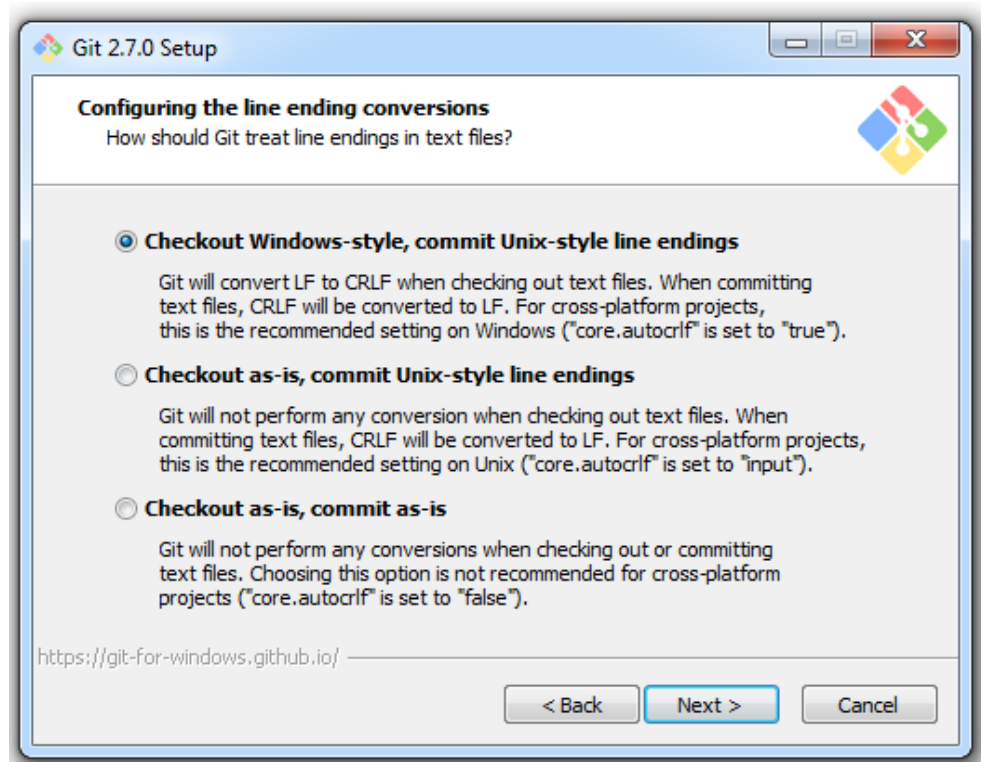
Sau khi tải về ta có file.exe

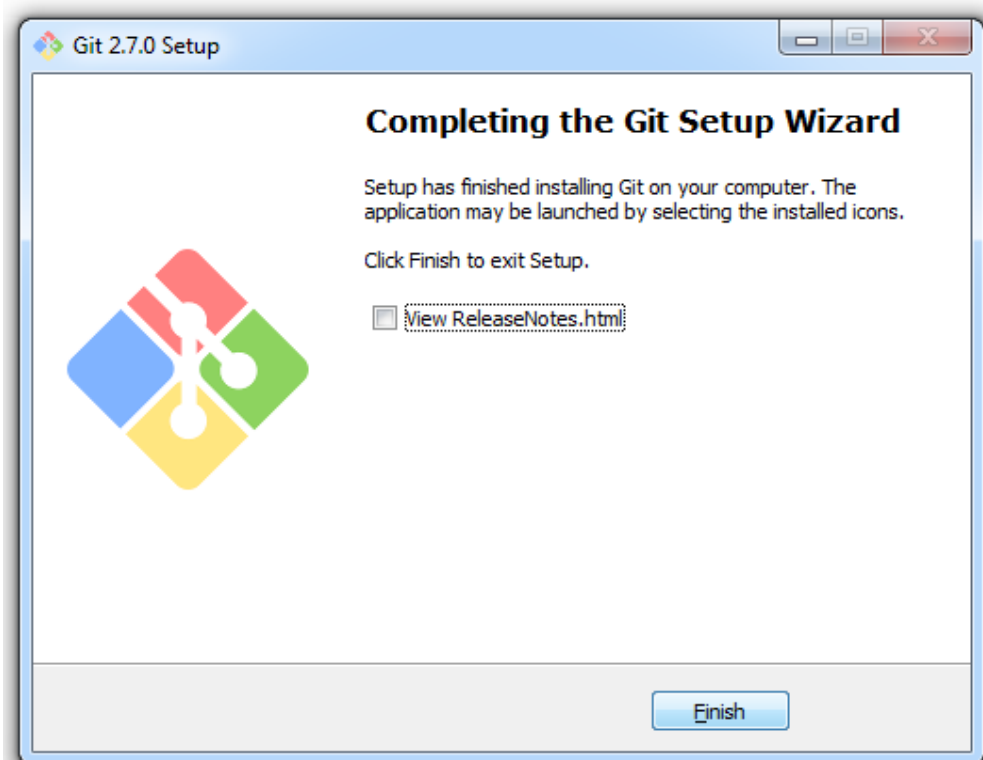
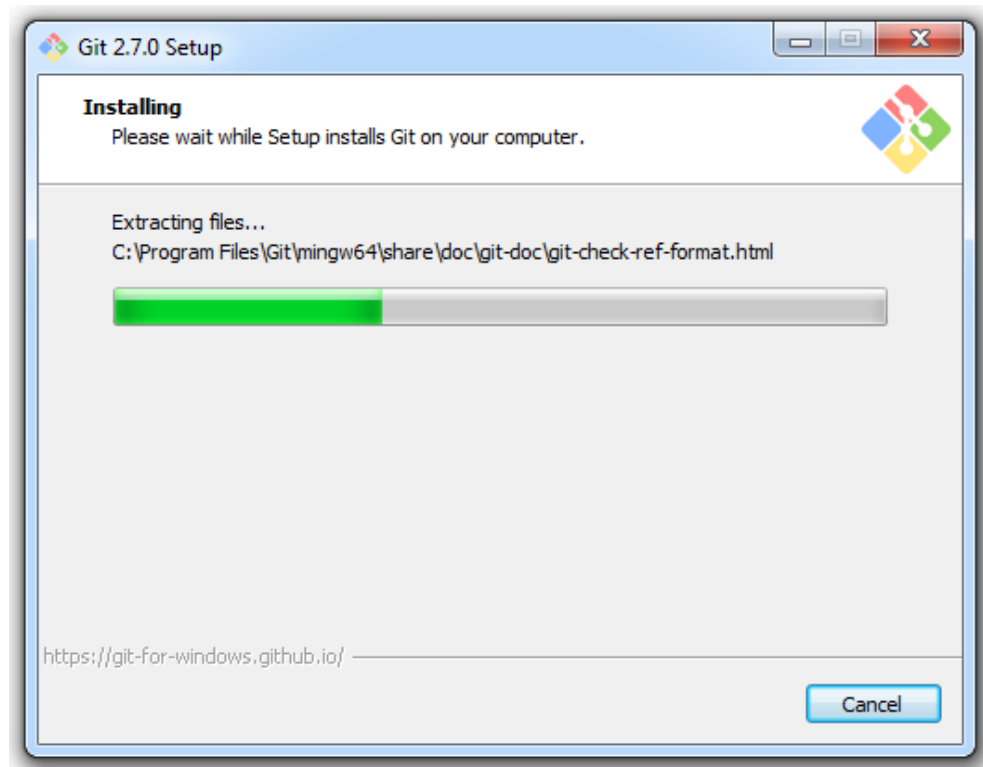


Sau đó cứ tiếp tục nhấn next ta sẽ hoàn thành việc cài đặt (các bạn có thể tham khảo thêm những hình dưới đây)

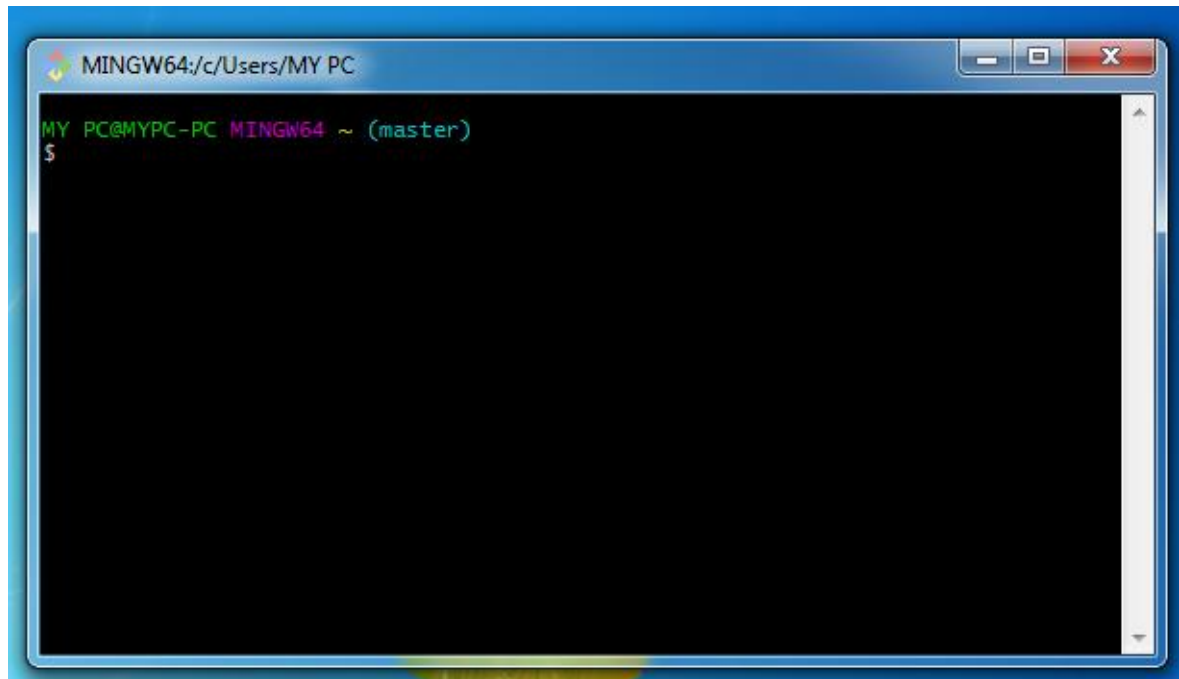






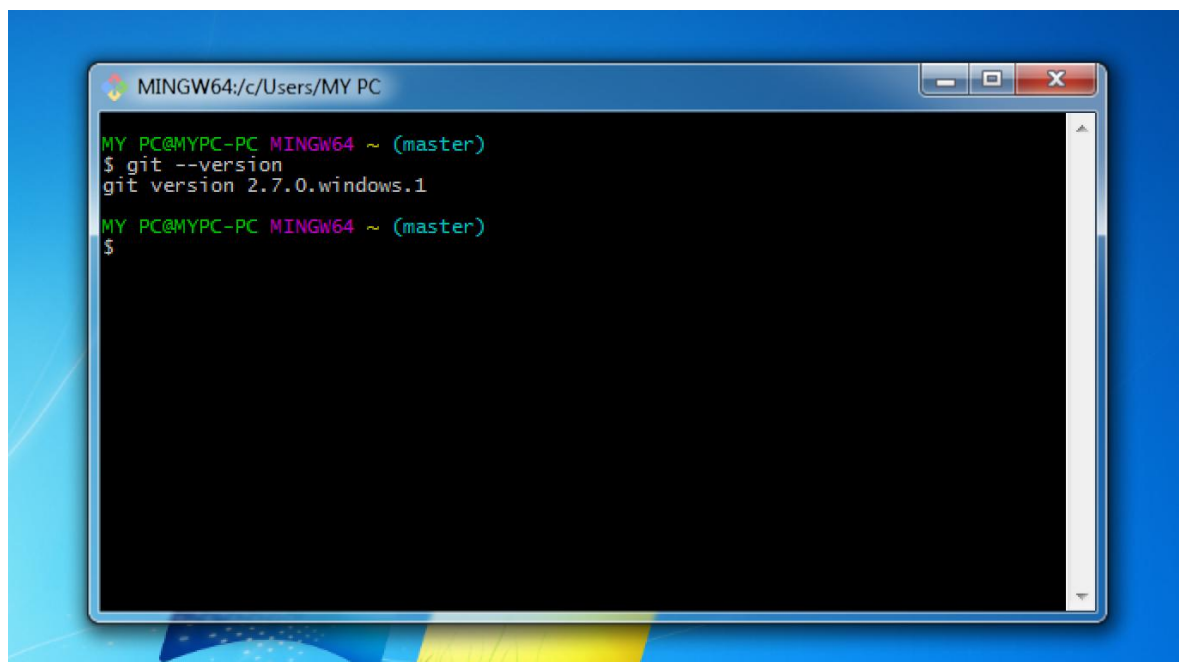


Và đây là Gitbash



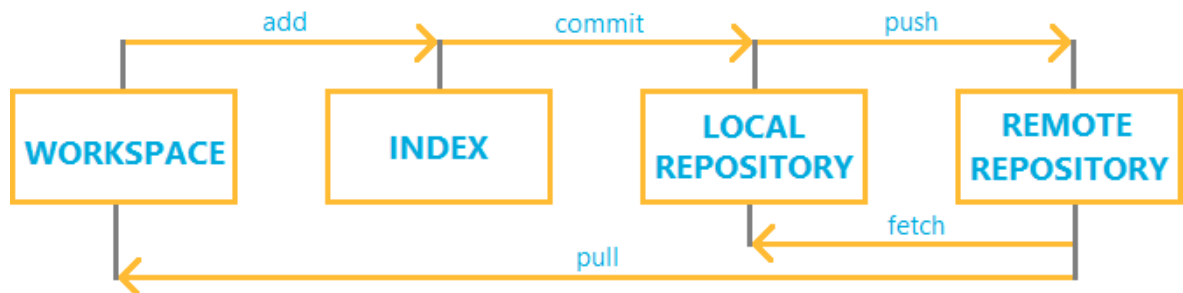
Để kiểm tra xem việc cài đặt đã thành công chưa và phiên bản của Git ta mở Gitbash hoặc cmd và gõ lệnh:

```
$ git --version
```



Quy trình làm việc

Quy trình làm việc là rất quan trọng và cần được rèn luyện nhiều nếu người sử dụng muốn thành thạo trong việc sử dụng Git. Quy trình làm việc của Git được biểu diễn bằng sơ đồ sau:



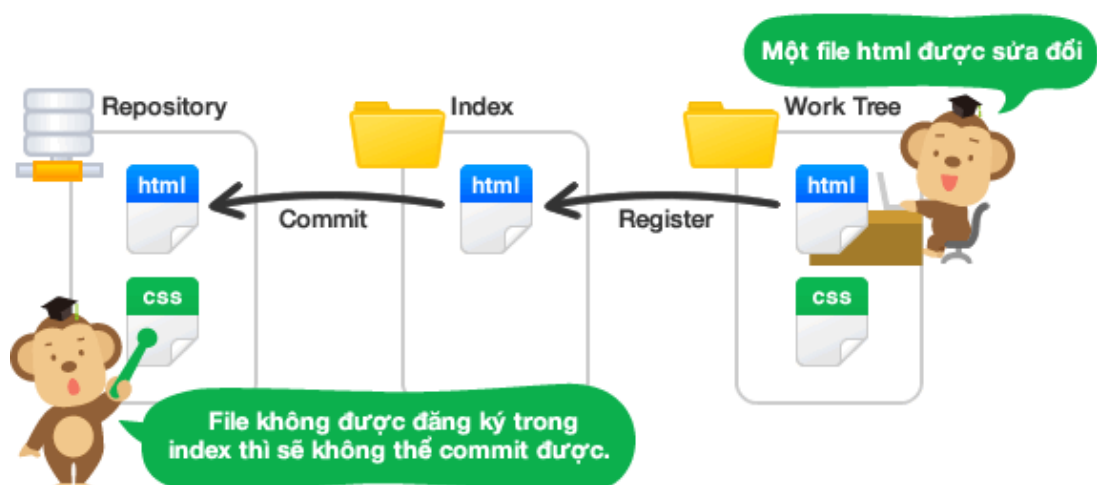
Các thành phần trong quy trình

WORKSPACE: Trạng thái làm việc hiện tại. Các thao tác làm thay đổi mã nguồn, thêm hoặc xóa tài nguyên, ... sẽ được lưu trữ ở đây.

INDEX: Trạng thái khi các thay đổi đã được lưu lại (bằng các lệnh add).

LOCAL REPOSITORY: Sau khi lưu lại, bạn thực hiện thao tác commit, đưa toàn bộ mã nguồn vào local repository. Mỗi commit sẽ có 1 tag (nhãn) riêng để có thể phục hồi lại bất cứ lúc nào.

REMOTE REPOSITORY: Bằng thao tác push, toàn bộ mã nguồn sẽ được đưa lên remote repository và có thể chia sẻ với các thành viên khác trong nhóm thông qua Internet.



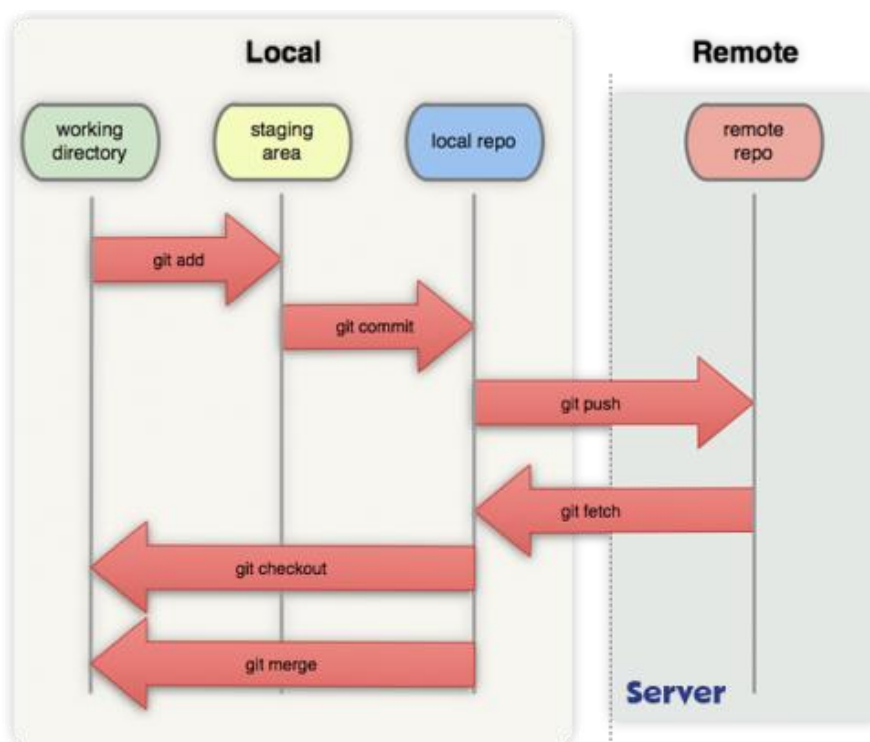
Tiến trình công việc (workflow) cơ bản của Git⁴:

1. Bạn thay đổi các tập tin trong thư mục làm việc.
2. Bạn tổ chức các tập tin, tạo mới ảnh của các tập tin đó vào khu vực tổ chức.
3. Bạn commit, ảnh của các tập tin trong khu vực tổ chức sẽ được lưu trữ vào thư mục Git.

Đăng ký tài khoản Github

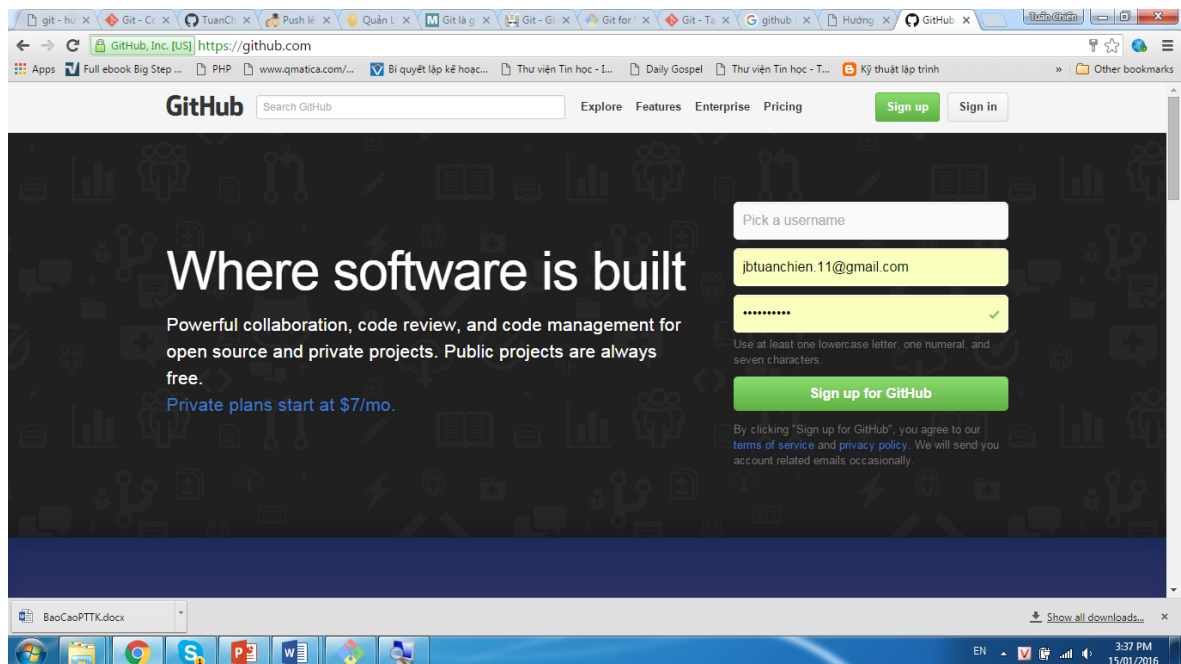
Github còn được gọi là social network dành cho developer đi vào hoạt động tháng 2 năm 2008, là một dịch vụ sử dụng hệ thống quản lý phân tán GIT giúp người dùng lưu trữ source code cho các dự án.

Github được viết bằng Ruby on Rails. GitHub cung cấp dịch vụ thương mại và cả tài khoản miễn phí cho các dự án nguồn mở. Theo khảo sát của người sử dụng Git vào năm 2009, Github hiện đang là server Git lưu trữ source code phổ biến nhất hiện nay



⁴ Xem : <https://git-scm.com/book/vi/v1> Phần Bắt đầu cơ bản về Git

Trước hết bạn cần phải đăng ký miễn phí một tài khoản GitHub. Bạn có thể vào trang chủ của GitHub tại: <https://github.com/>



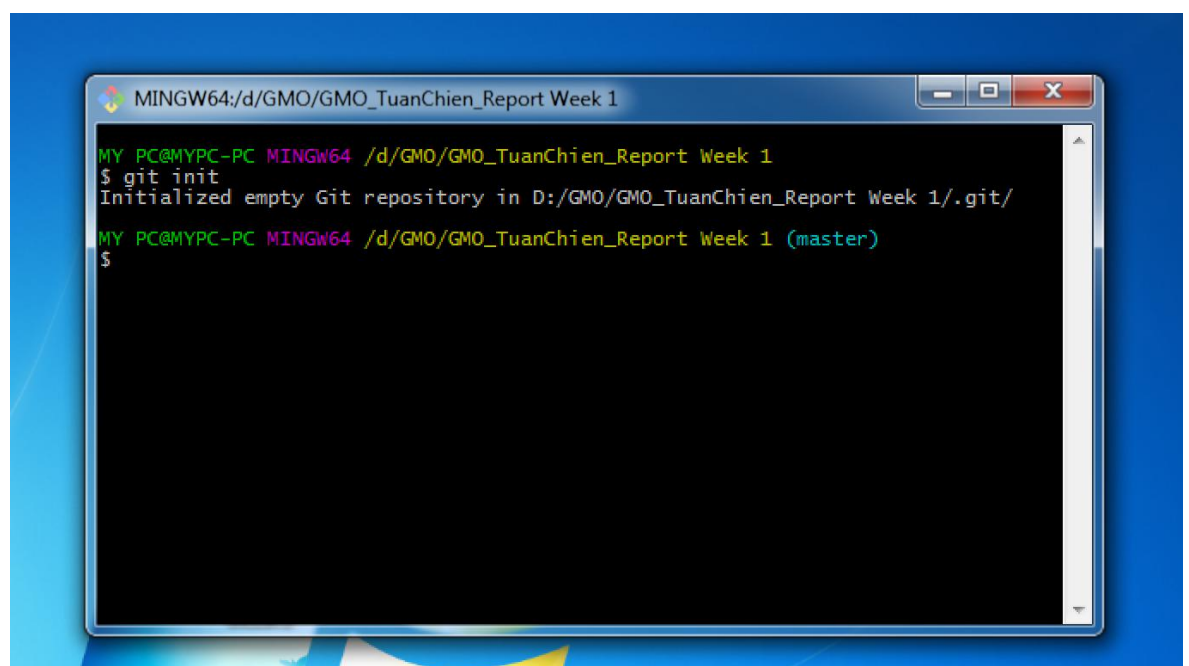
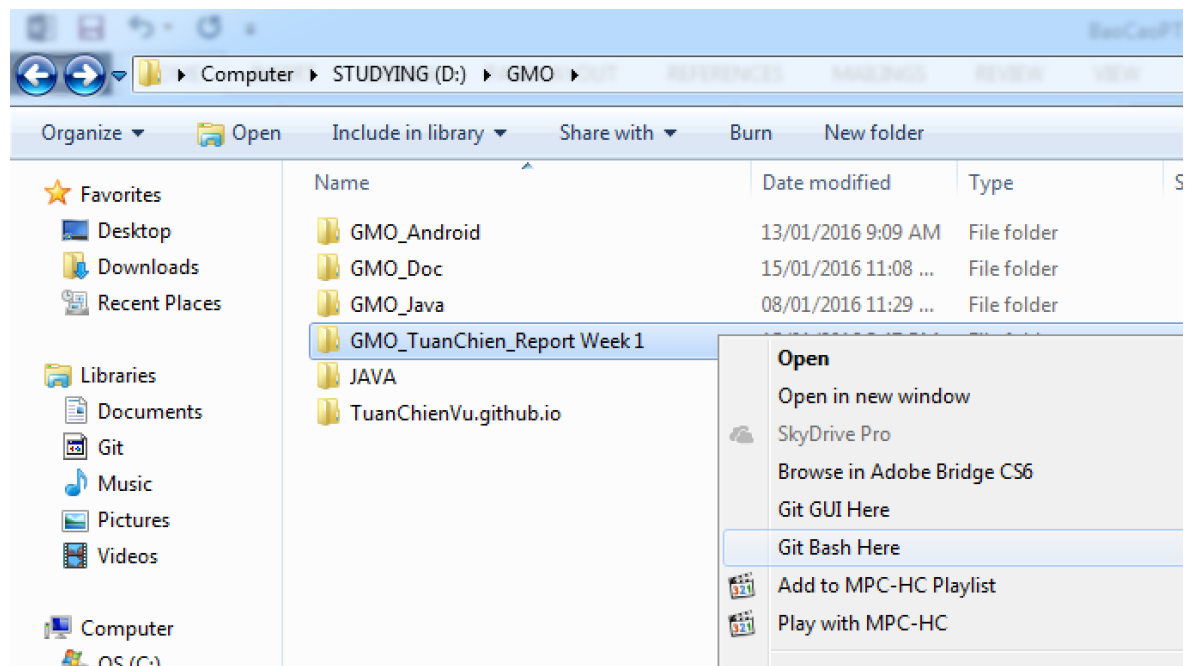
Việc đăng ký một tài khoản là đơn giản, bạn chỉ cần nhập username/password và địa chỉ email. Sau khi đăng ký xong bạn cần vào Email kích hoạt tài khoản. Mọi việc hoàn thành.

Các thao tác cơ bản về Git

Init

Khởi tạo môi trường làm việc cục bộ là bạn tạo một folder chứa code cái mà bạn muốn đưa lên Github. Folder phải hoàn toàn trống trước khi thực hiện lệnh init → click chuột phải vào folder → chọn Git Bash Here. Cú pháp:

```
1. git init
```



Clone

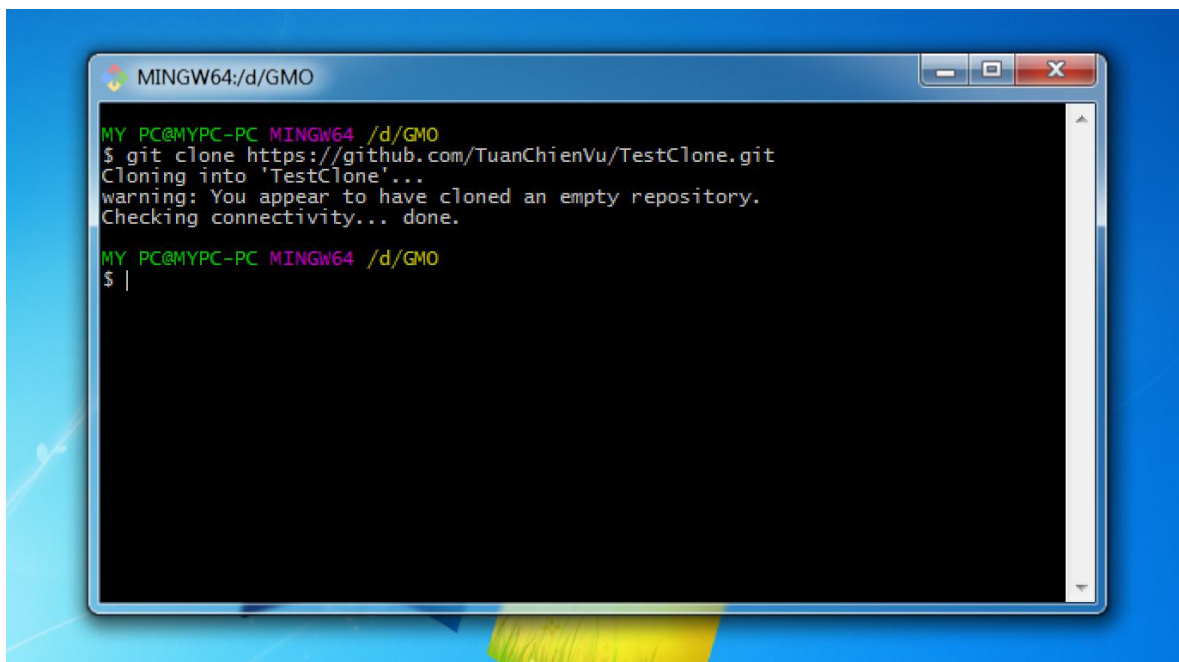
Khi 1 repository đã có sẵn trên máy cục bộ, việc cần làm là lấy các thông tin dữ liệu đó về. Lệnh clone sẽ giúp chúng ta thực hiện việc đó.

Cú pháp:

```
1. git clone /path-to/repository/
```

Nếu repository đó ở máy chủ khác thì bạn hãy gõ dòng lệnh sau:

```
1. git clone path
```



Sau khi clone thành công vào lại ổ D kiểm tra lại thư mục mình mới clone về

Commit

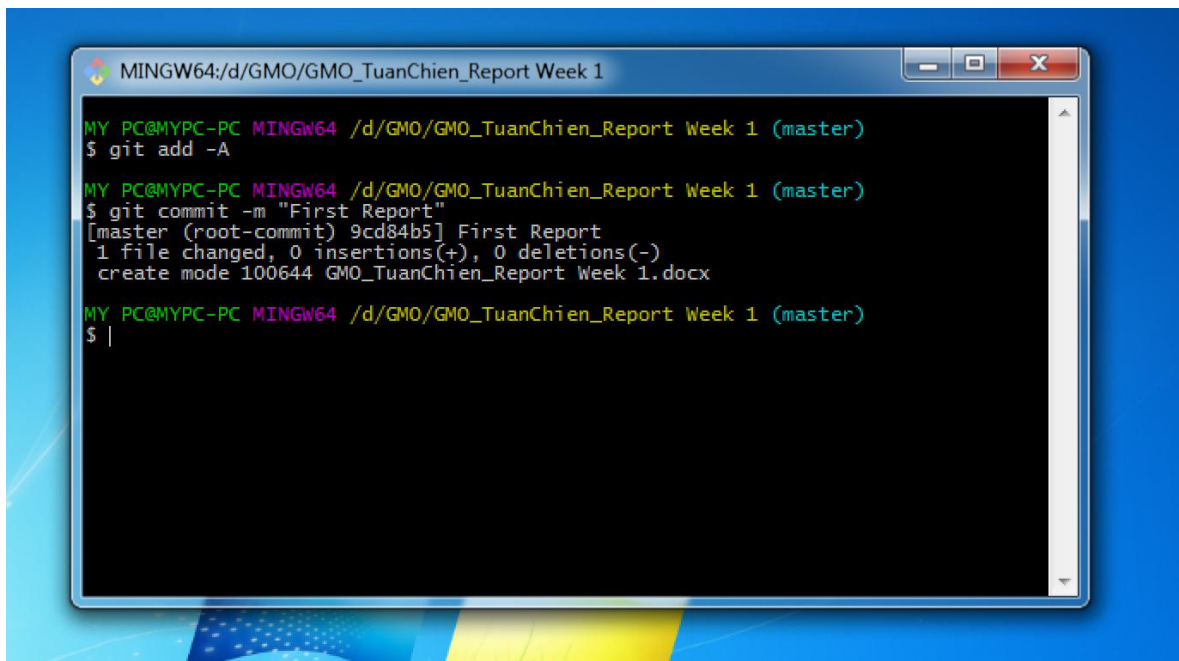
Sau khi thực hiện xong các thay đổi như thêm tập tin, source code, thay đổi chỉnh sửa code,... chúng ta sẽ đưa các thay đổi này vào Index và sau đó là Local repository. Trước khi commit, chúng ta sẽ sử dụng lệnh add để đưa tất cả vào Index.

Trong ví dụ này tôi đưa vào file: GMO_TuanChien_Report Week 1.docx

Cú pháp:

```
1. git add --all
2. git commit -m "Comment"
```

Lưu ý: Comment là một chuỗi ký tự đặt trong hai dấu “”. Comment có thể được chia thành nhiều dòng, nhưng vẫn đảm bảo nằm giữa hai dấu “”. Các bạn có thể sử dụng bất kỳ chuỗi ký tự nào để đưa vào Comment, nhưng tôi khuyến khích các bạn nên đưa vào Comment cụ thể những thay đổi đã được thêm vào commit đó. Điều này sẽ giúp việc quản lý được hiệu quả hơn vì ta dễ dàng nắm được tiến độ của dự án, đồng thời dễ dàng quay lại đúng thời điểm nếu cần thiết.



```
MINGW64:/d/GMO/GMO_TuanChien_Report Week 1
MY_PC@MYPC-PC MINGW64 /d/GMO/GMO_TuanChien_Report Week 1 (master)
$ git add -A
MY_PC@MYPC-PC MINGW64 /d/GMO/GMO_TuanChien_Report Week 1 (master)
$ git commit -m "First Report"
[master (root-commit) 9cd84b5] First Report
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 GMO_TuanChien_Report Week 1.docx
MY_PC@MYPC-PC MINGW64 /d/GMO/GMO_TuanChien_Report Week 1 (master)
$ |
```

Push

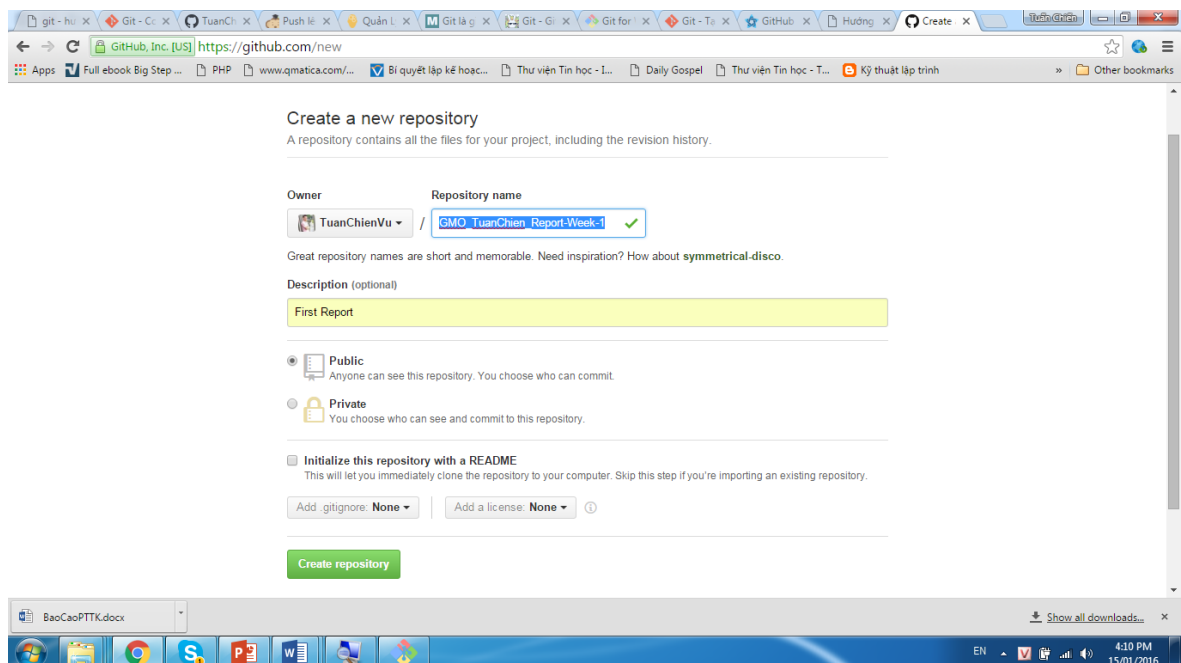
Sau khi commit file.docx của mình lên, các thay đổi sẽ được lưu trữ tại Local repository, thay đổi của bạn hiện đang nằm tại HEAD⁵ của bản sao cục bộ đang làm việc. Chúng ta sử dụng lệnh push để đưa toàn bộ dữ liệu lên Remote repository.

```
1. git push origin branch_name
```

Trong đó branch_name là tên của nhánh làm việc hiện tại. Hình dưới đây mình chọn nhánh chính là Master

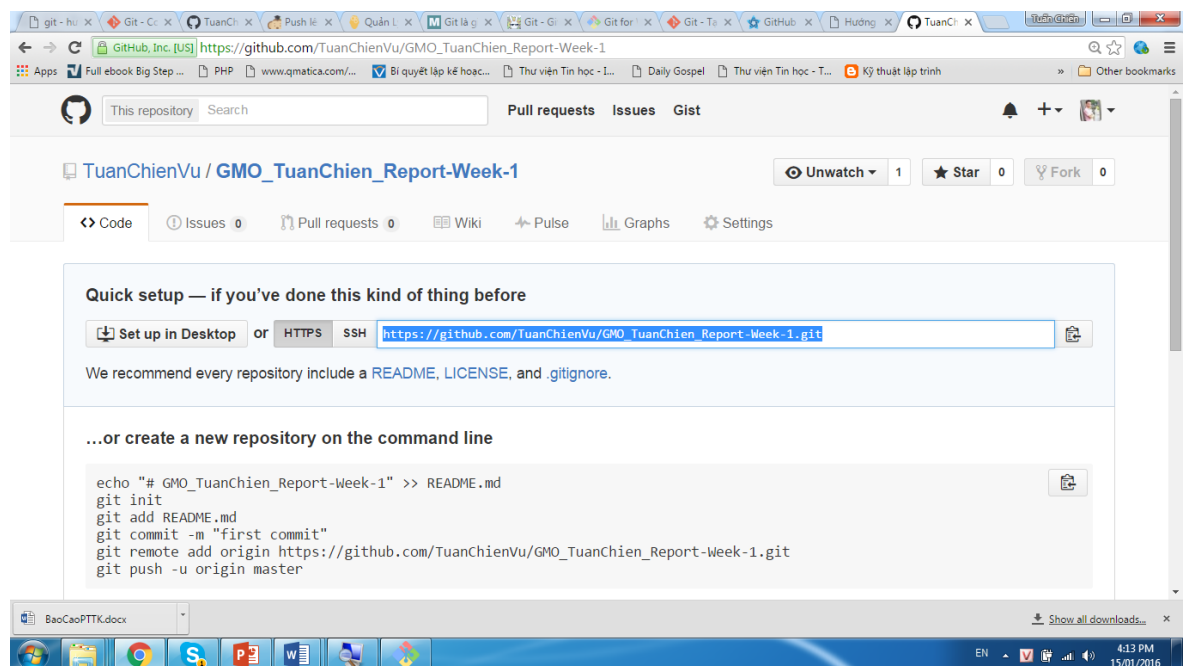
Trong trường hợp bạn chưa clone một repository hiện có trên Github bạn cần thực hiện công việc này trước khi Push, tạo một repository bằng cách vào <https://github.com>

➔ chọn New Repository ➔ Đặt tên cho repository và nhấn Create



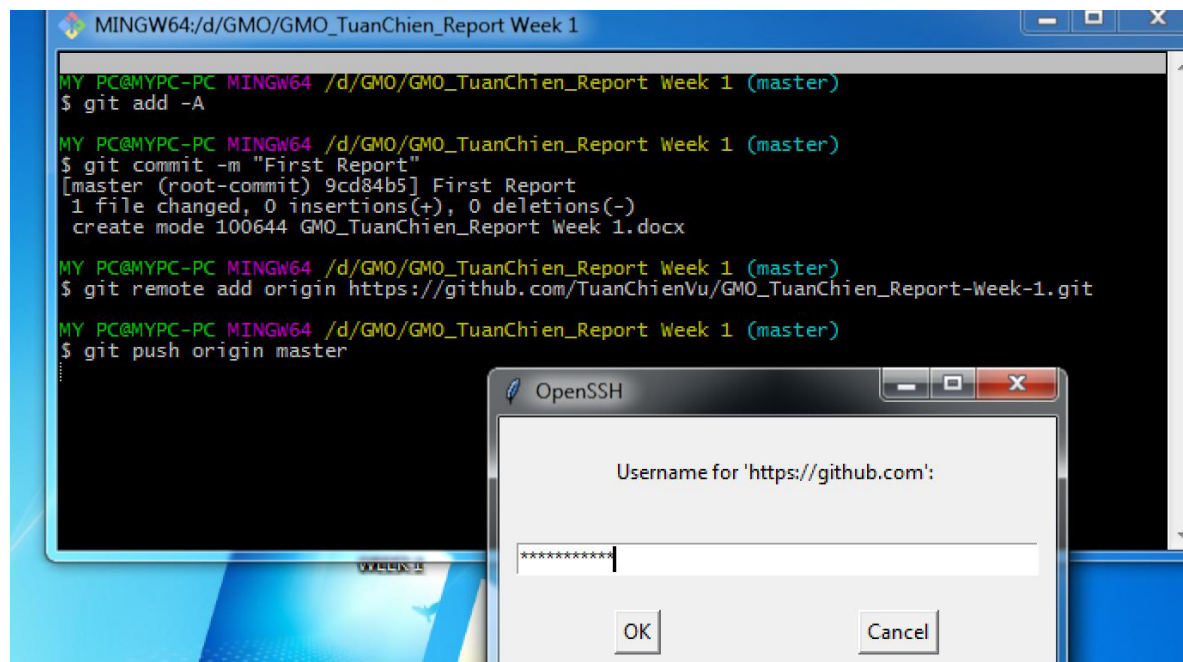
⁵ Xem phụ lục về HEAD

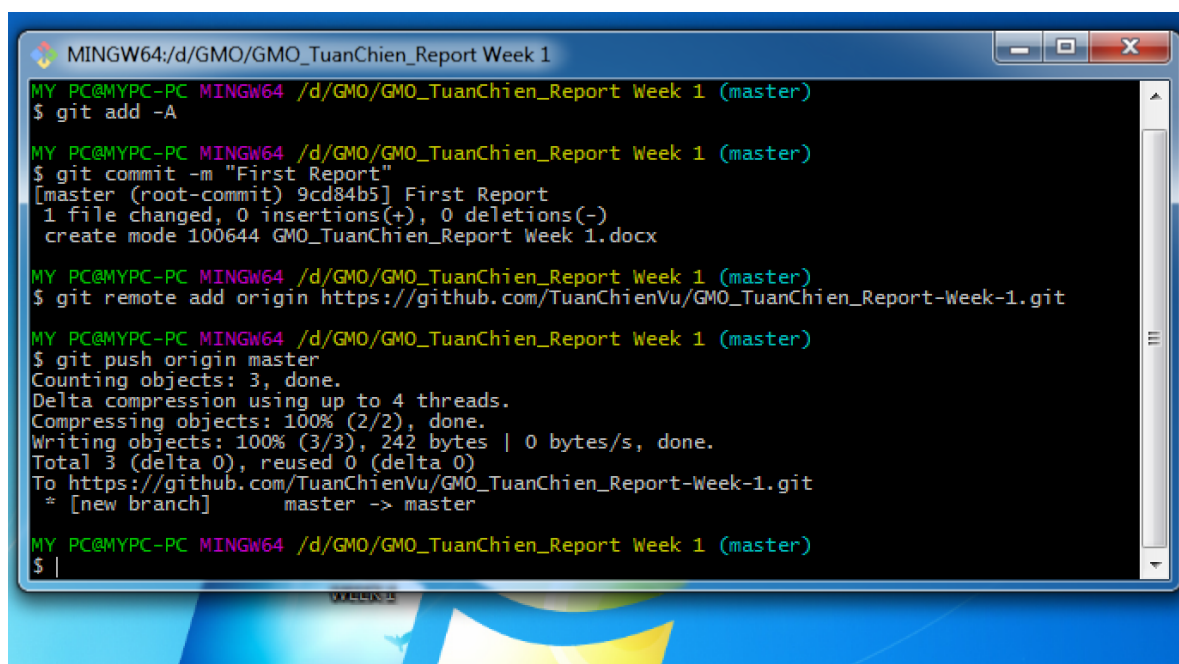
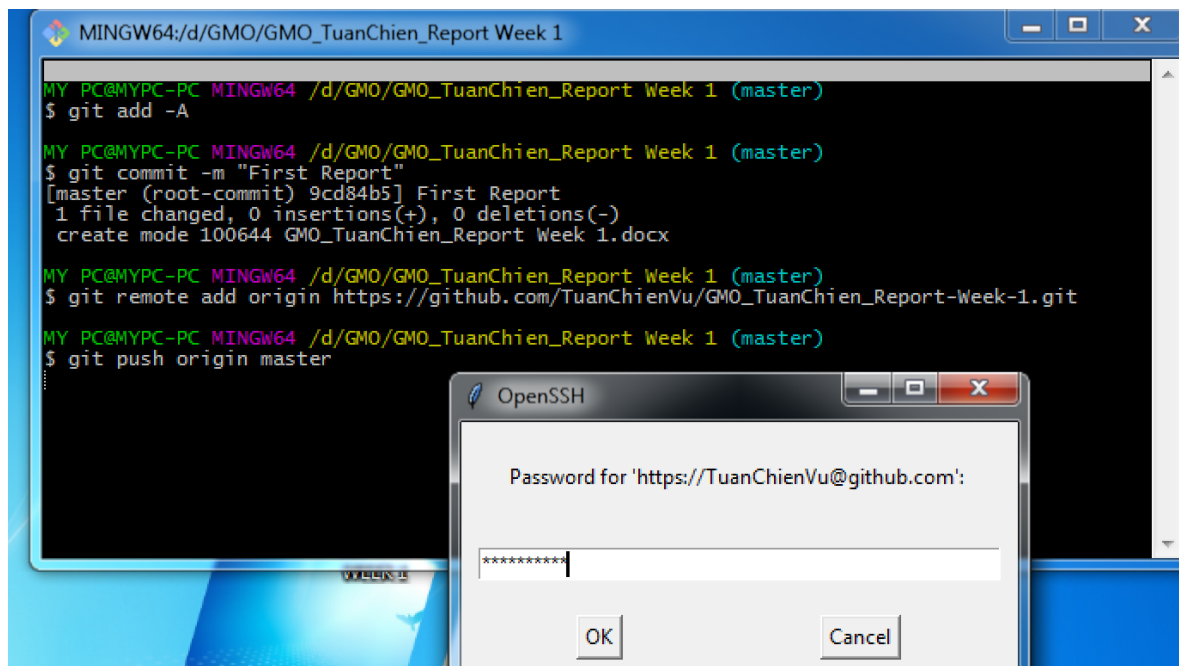
Sau khi tạo xong bạn có thể copy đường dẫn https để gán vào tên máy chủ như hình sau:



và muốn kết nối repository của bạn đến máy chủ Github, bạn phải thêm như sau:

```
1. git remote add origin <máy chủ>
```





Bây giờ bạn đã có thể đẩy các thay đổi của mình vào máy chủ đã chọn

Pull

Là hành động kéo dữ liệu từ trên Remote repository về máy tính cá nhân. Chúng ta có thể kéo toàn bộ dữ liệu, hoặc chỉ định một nhánh cụ thể nào đó. Cú pháp:

```
1. git pull
```

Fetch

Khi xảy ra lỗi, nếu ta muốn trở lại commit gần nhất để bắt đầu lại công việc, lệnh fetch sẽ giúp chúng ta thực hiện điều đó. Cú pháp như sau:

```
1. git fetch
```

Config

Trong trường hợp khởi tạo một môi trường làm việc mới hoàn toàn ở máy cục bộ, khi đưa các thay đổi lên Remote repository, bạn sẽ được yêu cầu khai báo username và email. Cú pháp như sau:

```
1. git config --global user.name "Your username"  
2. git config --global user.email "Your email"
```

Log

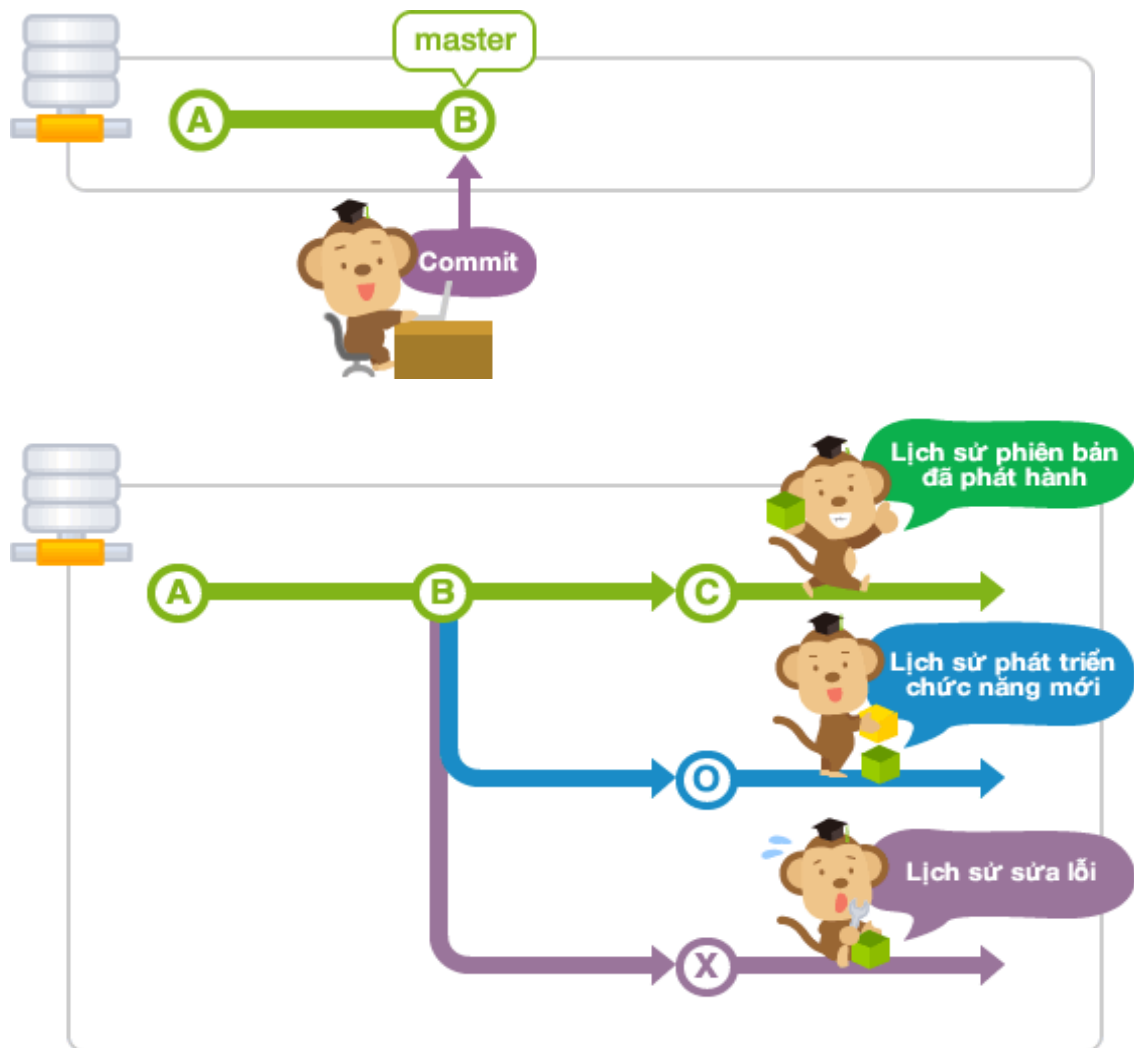
Lệnh **git log** cho phép ta xem lại lịch sử các lần commit trước.

Các lệnh ở trên có thể có một số tùy chọn đặc tả riêng cho từng lệnh. Trong giới hạn bài viết tôi chỉ hướng dẫn những gì cơ bản và thông dụng nhất. Các bạn có thể tự nghiên cứu thêm để sử dụng một cách hiệu quả nhất.

Branch là gì?

Branch là cái dùng để phân nhánh và ghi lại luồng của lịch sử. Branch đã phân nhánh sẽ không ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng 1 repository.

Khi tiến hành commit lần đầu trong repository thì Git sẽ tạo ra một branch có tên là master. Vì thế những lần commit sau sẽ được thêm vào branch master cho đến khi chuyển đổi branch.

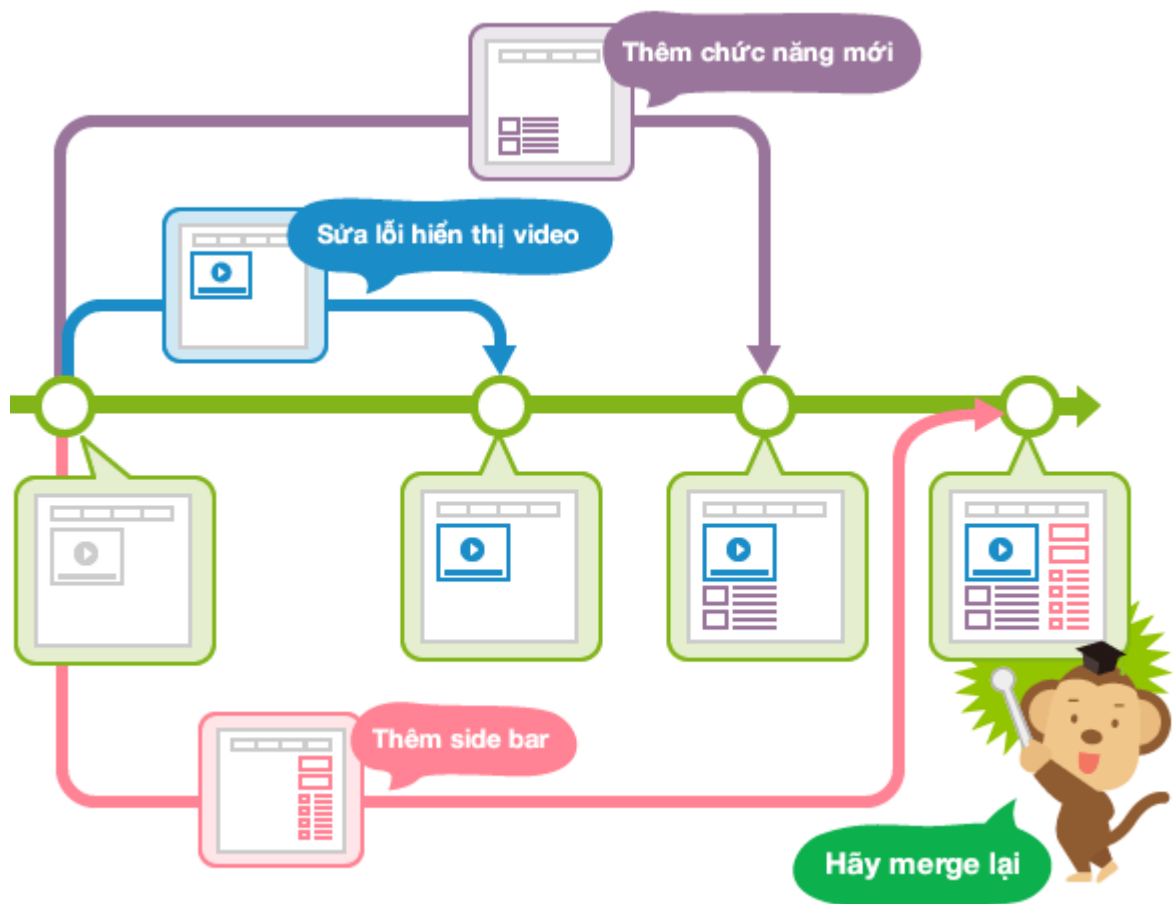


Branch đã phân nhánh có thể chỉnh sửa tổng hợp lại thành 1 branch bằng việc hợp lại (merge) với branch khác.

Các thành viên của nhóm sẽ tạo branch dùng riêng cho công việc của mình từ branch chính để không ảnh hưởng đến công việc của các thành viên khác. Sau đó, những thành viên đã hoàn thành công việc của mình sẽ thực hiện đưa thay đổi của mình vào branch chính.

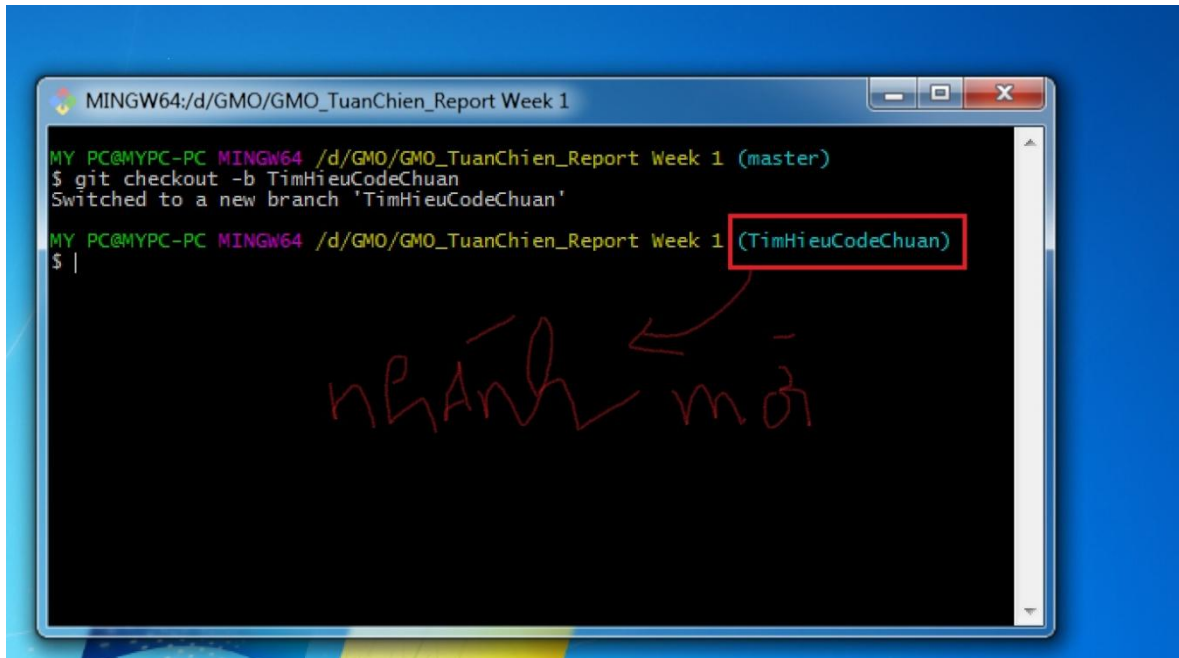
Theo cách như vậy, sẽ không bị ảnh hưởng từ công việc của các thành viên khác, và bản thân mình có thể thực hiện công việc của mình.

Hơn nữa, bằng việc để lại lịch sử theo đơn vị công việc, trong trường hợp có phát sinh vấn đề thì việc điều tra nguyên nhân ở những vị trí thay đổi cũng như việc tiến hành đối sách khắc phục sẽ trở nên dễ dàng hơn.



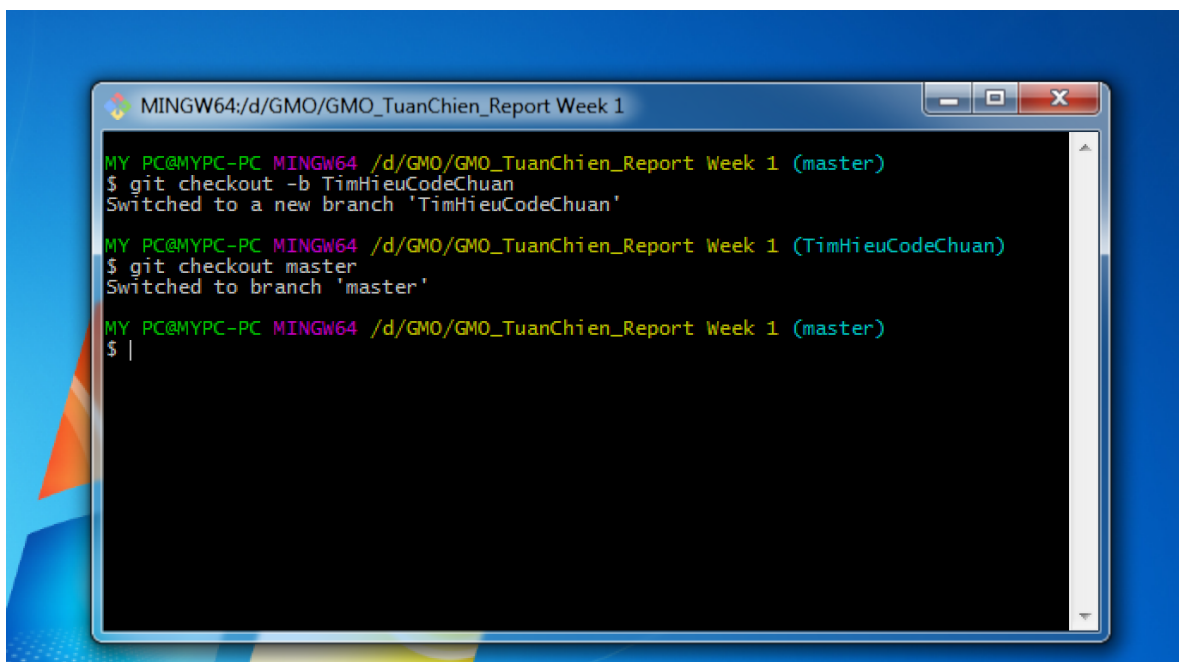
Tạo nhánh

```
1. git checkout -b branchname
```



Chuyển về nhánh Master

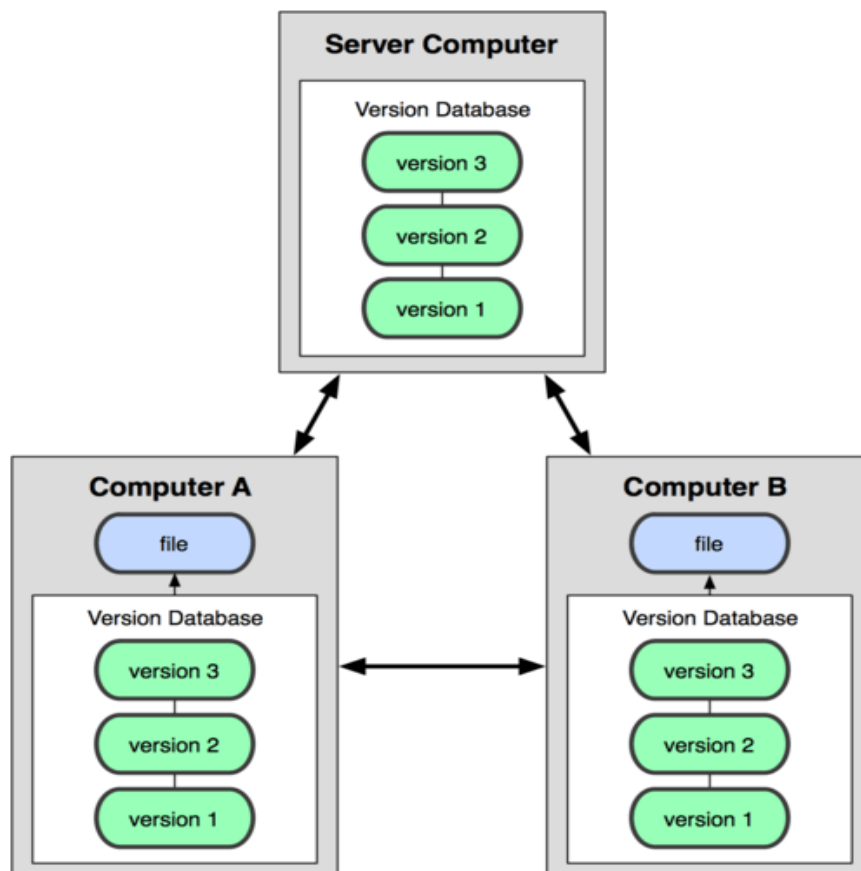
```
1. git checkout master
```



Phụ Lục

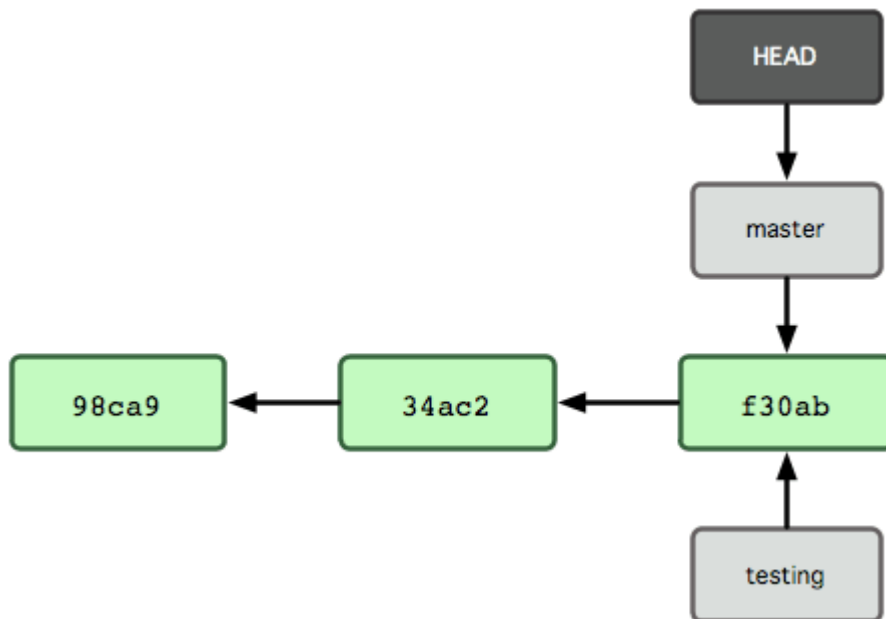
1. Hệ Thống Quản Lý Phiên Bản Phân Tán

Ngoài những hệ thống trước đây như hệ thống quản lý phiên bản cục bộ, quản lý phiên bản tập trung thì *Hệ Thống Quản Lý Phiên Bản Phân Tán* - Distributed Version Control Systems (DVCSs) bao gồm một số DVCS như Git, Mercurial, Bazaar hay Darcs, các máy khách không chỉ sao chép về máy tính cá nhân của mình phiên bản mới nhất của các tập tin mà chúng còn sao chép toàn bộ kho chứa (repository). Chính vì vậy nếu như một máy chủ nào đó đang ngừng hoạt động hay xảy ra lỗi thì kho chứa từ bất kỳ máy khách nào cũng có thể dùng để sao chép ngược trở lại máy chủ để khôi phục lại toàn bộ hệ thống (vì mỗi máy khách là một hệ thống riêng biệt). Mỗi lần checkout thực sự là một bản sao đầy đủ của tất cả dữ liệu.



2. HEAD

Làm sao Git có thể biết được rằng bạn đang làm việc trên nhánh nào? Git giữ một con trỏ đặc biệt có tên HEAD. HEAD là tên hiển thị phần đầu của branch đang sử dụng hiện tại. Mặc định là đang hiển thị phần đầu của master. Bằng việc di chuyển HEAD thì branch đang sử dụng sẽ được thay đổi.



Sau khi thực hiện lệnh Checkout

