HƯỚNG DẪN SỬ DỤNG GIT VÀ GITHUB CĂN BẢN

A) Quảng cáo GIT

Git là một công cụ giúp quản lý phiên bản của các phiên bản của một file.Lấy ví dụ cụ thể là hôm nay bạn vẽ ra một nhân vật game rất đẹp và lưu file ảnh đó là *player.png* . Sáng hôm sau bạn chợt nảy ra ý tưởng vẽ thêm cho nhân vật game đó 2 cái sừng , nhưng vì bạn sợ là sau khi vẽ thêm sừng mà chẳng may nó xấu thì bạn lại sẽ mất đi nhân vật game đẹp đẽ lúc đầu nên bạn save bức hình thứ 2 thành *player2.png* . Một số bạn kỹ hơn thì sẽ lưu lại thành *player-VeThemSung.png* . Cứ như vậy cho đến khi bạn có được nhân vật game ưng ý nhất thì sẽ thòi ra hàng chục hàng trăm tấm hình khác nhau trong thư mục của bạn. Điều này vẫn ổn nếu như thư mục của bạn có ít file thay đổi. Hãy thử tưởng tượng một chương trình được xây dựng từ hàng trăm file source code khác nhau, nếu mỗi lần sửa một file mà cũng phải lưu lại thành một phiên bản khác của file đó thì thật là kinh dị.

Rắc rối này có thể được giải quyết bởi Git và 10 dòng lệnh cơ bản nhất của nó. Git giải quyết được vấn đề này bằng cách theo dõi các files, nhận biết sự thay đổi của các files, lưu lại thông tin về những thay đổi trên các files đó (lưu ý là ở đây git chỉ lưu lại thông tin về những thay đổi của một file trên dữ liệu của nó). Nếu bất cứ khi nào mà những thứ bạn vừa thay đổi trong file trở nên không vừa ý hoặc gây ra lỗi, bạn đều có thể ra lệnh cho Git phục hồi lại file này trở lại trạng thái mà bạn cảm thấy ưng ý nhất. Hoặc sau khi đã phục hồi lại mà bạn cảm thấy hối tiếc vì phát hiện ra file lúc nãy ngon hơn thì bạn vẫn có thể ra lệnh cho Git phục hồi ngược lại một lần nữa.

Nói chung những thay đổi nào đã được lưu vào Git thì sẽ không bao giờ mất đi, và luôn có thể phục hồi lại được (trừ khi bạn cố ý muốn xóa hoặc vô tình làm hư máy).

Hoạt động của này của git giống như việc Backup và Restore trên hệ điều hành. Mỗi khi bạn muốn cài một phần mềm mới vào hệ điều hành, thì bạn nên Backup lại HĐH để lỡ như phần mềm này gây ra lỗi thì bạn có thể Restore lại HĐH ngay trước thời điểm cài phần mềm, và HĐH lại hoạt động bình thường trở lại. Tuy nhiên việc Backup&Restore này một khi đã quay ngược về trạng thái xưa cũ thì không thể quay trở lại trạng thái vừa trước đó được nữa.

B) Cách sử dụng Git

1) Cài và thiết lập git:

+ Bước 1: với Windows vào trang https://git-scm.com/ và download file cài đặt của git về, với Linux thì đối với những phiên bản mới đã có git sẵn cho các bạn sử dụng. Hãy down đi đừng ngại, Git rất nhẹ chỉ có 17Mb thôi (lưu ý server git siêu rùa bò).

+ Bước 1b (Chỉ áp dụng cho Windows): tất nhiên là cài git đê, cứ để mặc định hết mà bấm Next. Sau khi cài xong mở Command Prompt lên gõ lệnh [git] (từ nay về sau các lệnh command line hay terminal đều sẽ để trong ngoặc vuông nhé, nhớ là đừng có gõ cái ngoặc vuông vào đó), nếu lệnh này chạy ra một đống gì đó kiểu như hình dưới thì ok. Đối với linux thì mở terminal lên gõ [git] vào thử xem linux đã cài sẵn git chưa .

+ Bước 2: Thiết lập user name và email . Cài này khá là quan trọng nếu sau này làm việc với các server lưu trữ online của git, đừng bỏ qua bước nào nhé. Việc này ta chỉ làm một lần duy nhất , nếu như muốn sửa đổi user name hay email thì mới thực hiện lại. Ta dùng 2 câu lệnh sau:

- [git config --global user.name "nhập tên bạn vào giữa 2 cái dấu nháy kép này nha"]

- [git config --global user.email <nhập địa chỉ mail của bạn ở đây, bỏ 2 cái dầu lớn, bé đi nha>]

+ Bước 2z: Kiểm tra lại thiết lập bằng câu lệnh [git config user.name] và [git config user.email] nếu ra đúng user name và địa chỉ mail lúc đầu thì việc config thành công.

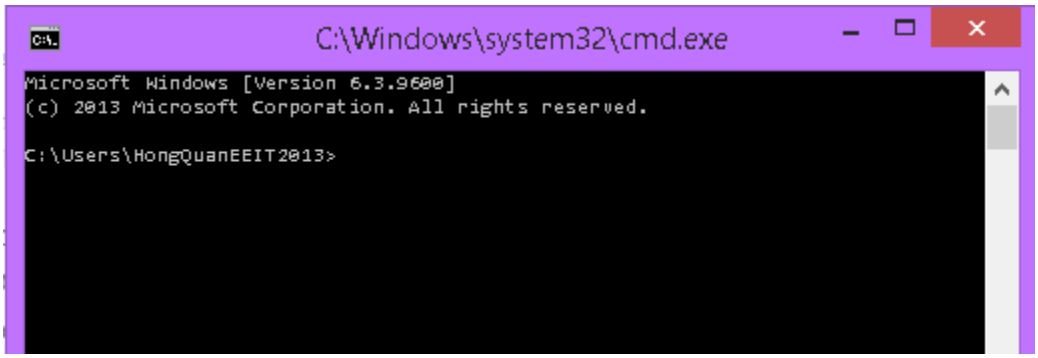
2) Backup một file trong Git:

Đối với phần này chúng ta sẽ thực hành trên một ví dụ và mình sẽ từng bước giải thích cho các bạn về cách làm việc đại khái của git qua từng bước, từng dòng lệnh. Lưu ý là ở đây mình sẽ sử dụng hoàn toàn bằng Command Prompt của Windows , nếu có chỗ khác trên Terminal của Linux thì mình cũng sẽ chỉ ra cho các bạn. Mình khuyến khích các bạn khi dùng git nên sử dụng hoàn toàn bằng CMD (Windows) hay Terminal (Linux) vì sẽ tiện lợi và nhanh hơn.

Ở phần thực hành này chúng ta sẽ làm việc trên một file là "gitbasic.txt". Chúng ta sẽ tạo ra nó ngay sau đây.

+ Bước 1: các bạn ở Command Prompt hay Terminal lên (tất nhiên rồi =)))

Lưu ý: đối với các bạn sử dụng Windows thì khi mở CMD lên, chúng ta đang nằm ở thư mục mặc định của CMD có đường dẫn như sau C:\Users\<user name trên máy tính của bạn> như trong hình. (user name trên máy tính của mình là *HongQuanEEIT2013*).



Đối với Linux , thư mục mặc định của Terminal là thư mục /home/<username>/

+ Bước 2: Tạo một thư mục tên là project bằng lệnh [mkdir project]. Bạn vào thư mục mặc định của CMD hay Terminal kiểm tra xem có thư mục tên là *project* không. Nếu có bạn dùng tiếp lệnh [cd project] để đi chuyển vào trong thư mục *project* . Thư mục project này trong thực tế sẽ là thư mục chứa dự án của chúng ta, hoặc thư mục chứa mã nguồn của một phần mềm mà các bạn tạo ra. Ở ví dụ này nó sẽ là thư mục chứa file *gitbasic.txt* của chúng ta.

+ Bước 3: dùng tiếp lệnh [git init]. Lệnh này có chức năng tạo ra một bộ theo dõi, nhận biết và sao lưu những thay đổi diễn ra bên trong thư mục *project* . Lưu ý sau khi gõ lệnh [git init] một thông báo là đã khởi tạo một "empty Git Repository" sẽ hiện lên.Git Repository có thể hiểu nôm na ở đây chính là bao gồm thư mục project và bộ theo dõi, sao lưu của git.

+ Bước 4: sau khi đã thực hiện lệnh [git init], ta thực hiện tiếp lệnh [git status]. Sẽ có 3 thông báo sau hiện lên:

On branch master

Initial commit

nothing to commit (...)

- 3 thông báo đó có nghĩa là trên nhánh master đã khởi tạo một commit, nhưng chưa có gì để commit tiếp. Chúng ta sẽ bàn về ý nghĩa của "nhánh master" và "commit" này sau.

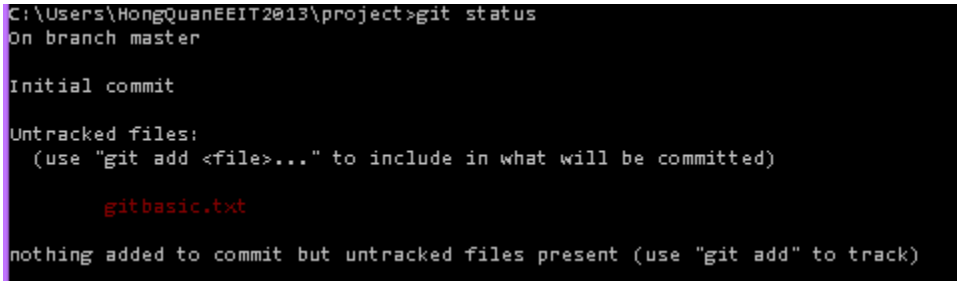
- Lệnh [git status] này chính là lệnh gọi bộ theo dõi của git, sau này sẽ được chúng ta sử dụng thường xuyên để kiểm tra trạng thái thay đổi của thư mục project.

+ Bước 5: bây giờ chung ta mới tạo file "gitbasic.txt".

- trên Windows ta sử dụng lệnh: [REM. > gitbasic.txt]. Lưu ý là ngay sau từ REM phải có dấu chấm và phải có dầu lớn hơn ở giữa, y chang như mình gõ ở trên.

- trên Linux ta sử dụng lệnh [touch gitbasic.txt].

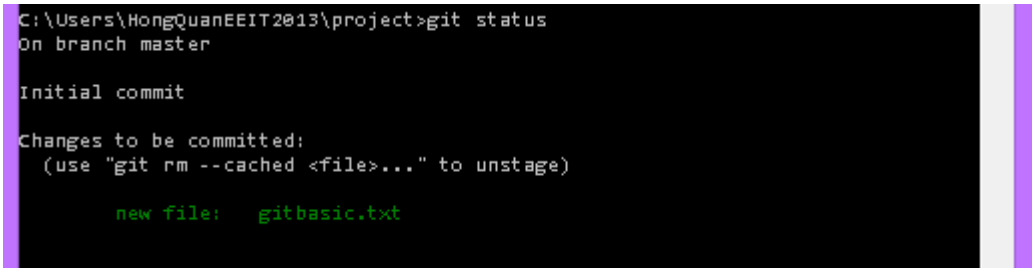
+ Bước 5b: sau khi đã tạo ra file *gitbasic.txt* ta sử dụng lệnh [dir] (trên Windows) hoặc lệnh [la] (trên Linux) để kiểm tra xem trên thư mục *project* đã có file *gitbasic.txt* hay chưa. Nếu đã có ta sử dụng lại lệnh [git status] sẽ hiện lên thông báo như hình dưới.



Ở đây git thông báo là: "trên nhánh master, có file *gitbasic.txt* chưa được nhận biết thay đổi, sử dụng lệnh [git add] để add nó vào bộ nhận biết sự thay đổi."

+ Bước 6: chúng ta sẽ dùng lệnh sau [git add gitbasic.txt] để add file *gitbasic.txt* vào bộ nhận biết sự thay đổi của git. Hoặc chúng ta thường sẽ dùng lệnh [git add --all] để add tất cả mọi thứ trong thư mục *project* vào bộ nhận biết sự theo dõi của git. Ở đây mình khuyên các bạn nên dùng lệnh [git add --all].

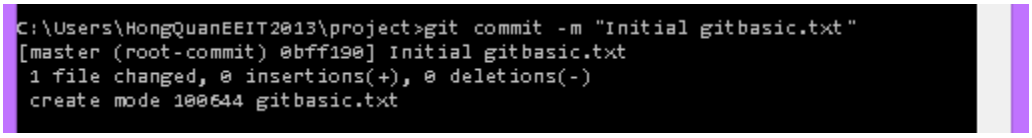
+ Bước 7: bây giờ chúng ta lại dùng lệnh [git status] một lần nữa. Sẽ có thông báo như trong hình hiện ra.



Thông báo này có nghĩa như sau: trên nhánh master, file mới gitbasic.txt vừa được add vào bộ theo dõi sự thay đổi của git và đang nằm chờ được sao lưu.

+ Bước 8: bây giờ chúng ta sẽ sao lưu lại file này vào trong git bằng lệnh [git commit –m "Khoi tao file gitbasic.txt"].

- Lệnh này có ý nghĩa là hãy lưu lại những gì đã được add vào bộ nhận biết , và gán cho nó một cái comment là "Khoi tao file gitbasic.txt". Vì lúc trước ta đã add vào bộ nhận biết chỉ mình file gitbasic.txt đang ở trạng thái nguyên thủy. Sau khi gõ lệnh này sẽ hiện một thông báo kiểu tựa như vầy. Mình sẽ giải thích ý nghĩa của "[master (root-commit) 0bff190]" và "create node 100644 gitbasic.txt" sau. Những cái còn lại bạn cũng hiểu đại khái là đã có 1 file thay đổi,0 thêm vào và 0 xóa bớt đi.



- Một số bạn sẽ thắc mắc tại sao file gitbasic.txt đã có thay đổi gì đâu mà đã add vào bộ nhận biết sự thay đổi. Mình xin trả lời là, bộ nhận biết này không chỉ nhận biết sự thay đổi của file mà còn nhận biết sự thay đổi của cả thư mục project . Vả lại file gitbasic.txt cũng thay đổi từ trạng thái không tồn tại sang trạng thái tồn tại. Chỉ cần có sự thay đổi trạng thái là có thể add vào bộ nhận biết.

- Nếu bạn dùng [git add gitbasic.txt] thì nó chỉ nhận biết sự thay đổi của mỗi mình file gitbasic.txt, nếu bạn dùng lệnh [git add --all] thì nó sẽ nhận biết sự thay đổi của tất cả mọi thứ bên trong thư mục project (vì chúng ta dùng [git init] bên trong thư mục project mà, nhớ chứ).

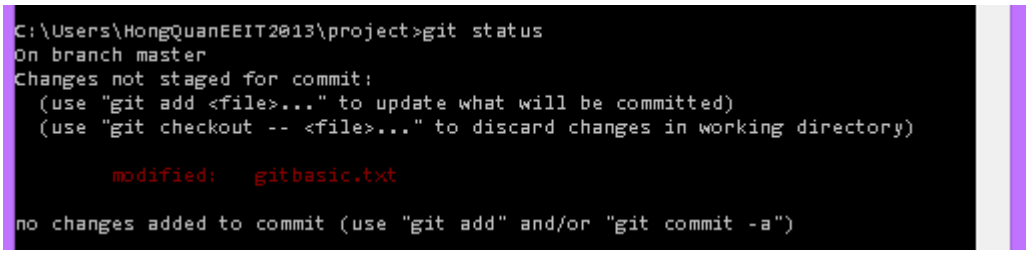
+ Bước 9: sau đó ta lại dùng lệnh [git status] một lần nữa. Lần này ta sẽ chỉ còn thấy 2 thông báo là:

On branch master

nothing to commit, working directory clean

điều này có nghĩa là chẳng còn gì để lưu cả, vì sau khi bạn dùng lệnh [git commit ....] thì nó sẽ tự động reset lại bộ nhận biết thay đổi, và báo với bộ theo dõi là: "không còn thay đổi nào đâu, tao đã sao lưu lại hết rồi".

+ Bước 10: bạn dùng lệnh sau [echo "Day la dong dau tien" > gitbasic.txt]. Sau đó bạn mở file gitbasic.txt lên sẽ thấy xuất hiện một đoạn text vừa được thêm vào là "Day la dong dau tien" xuất hiện trong file. Sau đó bạn cho chạy tiếp lệnh [git status] thì sẽ xuất hiện đoạn thông báo như hình dưới:



Bạn hãy chú ý dòng chữ đỏ trong hình, thông báo có nghĩa là file gitbasic.txt đã bị sửa đổi (đồng nghĩa với việc thay đổi, thêm mới hoặc xóa bớt nội dung trong file gitbasic.txt).

+ Bước 11: tiếp theo ta dùng lệnh [git add gitbasic.txt] hoặc [git add –all] để add những sửa đổi (thay đổi) này vào bộ nhận biết. Rồi ta lại dùng lệnh [git commit –m "them text vao gitbasic.txt"] để sao lưu những sửa đổi (thay đổi) của file gitbasic.txt lại.

Lưu ý: 2 lệnh trên có thể gộp chung làm một bằng cách sử dụng lệnh [git commit –a –m "them text vào gitbasic.txt"]. Tuy nhiên khi sử dụng 2 lệnh trên riêng rẽ, ta sẽ dễ quyết định xem thay đổi của file nào là cần thiết để đưa nó vào bộ nhận viết nhằm sao lưu. Còn nếu dùng [commit –a –m "..."] thì chúng sẽ sao lưu tất cả mọi sửa đổi của các files trong thư mục project.

Tổng kết tạm thời: Như vậy sau khi trải qua 11 bước trên ta đã hiểu cơ chế "theo dõi-nhận biết-sao lưu" của git hoạt động như thế nào và có những khác biệt gì khi ta tạo một file mới với khi ta sửa đổi một file cũ, phải sử dụng lệnh nào cho phù hợp. Hiện nay ta đã nắm trong tay 4 lệnh tối quan trọng của git là:

+ git init
+ git add (với option là --all)
+ git status
+ git commit (với 2 option là –a và –m)

Các bạn hãy ngồi ôn lại ý nghĩa của những lệnh này cùng option của nó là gì trước khi ta chuyển sang phần phục hồi trạng thái của file.

3) Phục hồi trạng thái của một file

Ở trong phần này, chúng ta sẽ tìm cách khiến cho file gitbasic.txt quay trở lại trạng thái ban đầu là một file rỗng, bên trong chưa hề có dòng "Day la dong dau tien". Tiếp đó chúng ta lại đưa file gitbasic.txt từ trạng thái rỗng mà ta vừa phục hồi sang ngược trở lại trạng thái mà trong đó có dòng "Day la dong dau tien".

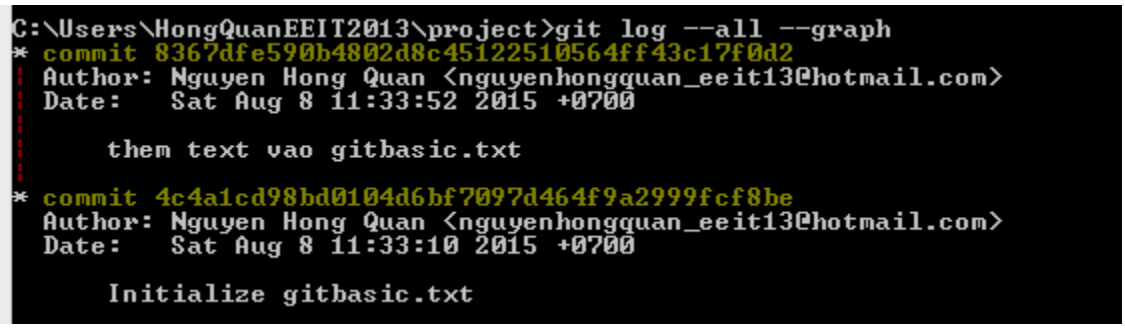
Lưu ý: đối với những dự án lớn có rất nhiều file, nếu chúng ta thay đổi nội dung ở nhiều file khác nhau, rồi sau đó [git add –all] và rồi [git commit –m "comment"] thì toàn bộ những thay đổi ở các file này sẽ được lưu lại. Ta sẽ gọi bản sao lưu đó là "**1 commit**".

Đối với việc phục hồi trạng thái, git cho ta 2 lựa chọn đó là phục hồi tất cả các file ở "1 commnit" hoặc phục hồi một số file nhất định trong "1 commit" đó.

a) Xem history của git

Trước khi chúng ta phục hồi trạng thái của file gitbasic.txt , chúng ta cần phải mở xem bảng log (history) của git, trong bảng này bao gồm thông tin của các lần commit (tác giả - thời điểm commit – nội dung comment trong lần commit đó – mã tên của lần commit đó).

Chúng ta sẽ dùng lệnh sau [git log --all --graph] , rồi chùng ta sẽ ra được bảng log giống như hình bên dưới.



Như chúng ta thấy trong hình có 2 dòng màu vàng , đằng trước là 2 dấu sao. Đây chính là mã tên của 2 lần commit mà chúng ta đã thực hiện trong phần trước. Lần commit mới nhất sẽ nằm ở bên trên, sắp xếp theo thứ tự thời gian. Còn lại là các thông tin về tác giả của lần commit đó (mình sẽ giải thích về phần tác giả này sau), thời gian thực hiện lần commit đó và nội dung comment (thường thì chúng ta sẽ dùng comment để nói lên nội dung của một lần commit tựa như "sua loi phan hien thi hinh anh").

Chúng ta có thể tùy chỉnh bàng git log này với nhiều option khác nhau (ngoài 2 option là --all và --graph). Nhưng chúng ta sẽ bàn về vấn đề này sau, hiện tại với phần hướng dẫn cơ bản này chúng ta chỉ cần quan tâm tới lệnh [git log --all --graph] là đủ.

b) Phục hồi lại trạng thái của gitbasic.txt

Bây giờ chúng ta nhìn lại vào bảng log (hình dưới).

```
C:\Users\HongQuanEEIT2013\project>git log --all --graph
* commit 8367dfe590b4802d8c45122510564ff43c17f0d2
| Author: Nguyen Hong Quan <nguyenhongquan_eeit13@hotmail.com>
| Date:   Sat Aug 8 11:33:52 2015 +0700
|
|     them text vao gitbasic.txt
|
* commit 4c4a1cd98bd0104d6bf7097d464f9a2999fcf8be
  Author: Nguyen Hong Quan <nguyenhongquan_eeit13@hotmail.com>
  Date:   Sat Aug 8 11:33:10 2015 +0700

      Initialize gitbasic.txt
```

Các bạn hãy để ý lại lần nữa dòng màu vàng có chữ commit và dấu sao ở phía trước (lưu ý, có thể trên máy tính khác sẽ có màu sắc và hình dấu khác không phải màu vàng và dấu sao).

Trên dòng này sau chữ commit sẽ có một dãy mã (mã tên). Bây giờ chúng ta muốn phục hồi lại file gitbasic.txt trở lại trạng thái ban đầu (trạng thái rỗng). Trạng thái đó đã được chúng ta lưu lại ở lần commit đầu tiên có comment là "Initialize gitbasic.txt". Như vậy nếu chúng ta muốn phục hồi lại trạng thái này chúng ta chỉ cần phục hồi lại lần commit đầu tiên đó, và lần commit đó có mã tên là ***4c4a1cd98bd0104d6bf7097d464f9a2999fcf8be***

Đề làm được việc này ta dùng lệnh : **[git checkout 4c4a1cd]**

Các bạn có để ý thấy là khi dùng lệnh [git checkout <mã tên>] ta chỉ sử dụng 7 ký tự đầu tiên trong mã tên của 1 commit. Tuy nhiên ta có thể sử dụng nhiều hơn 7 mã tên này (tức là từ 7 tới 40 ký tự trong mã tên theo thứ tự từ trái qua phải) nếu như số lần commit của chúng ta vượt quá 36^7 =78364164096 lần vì khi đó 7 mã đầu tiên sẽ bị lặp lại =)). Khi đó ta sẽ thấy hiện lên thông bao như sau. ***Lưu ý là mã tên trên mỗi máy mỗi khác, các bạn nhớ thay bằng mã tên của mình.***

```
Note: checking out '4c4a1cd98bd0104d6bf7097d464f9a2999fcf8be'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 4c4a1cd... Initialize gitbasic.txt
```

Ta không cần quan tâm tới các dòng khác ta chỉ lần quan tâm với dòng cuối cùng. Thông báo có nghĩa là con trỏ HEAD đang nằm tại commit có mã 4c4a1cd... với comment là Initialize gitbasic.txt. Mình sẽ giải thích về con trỏ HEAD này cho các bạn sau và các kỹ thuật nâng cao với lệnh [git checkout ...].

Sau khi đã thực hiện xong bạn hãy mở thở file gitbasic.txt để xem coi nó đã quay lại trạng thái trống rỗng ban đầu chưa. Bây giờ các bạn hãy thử quay lại trạng thái ở lần commit có comment là "them text vao gitbasic.txt" xem sao.