# Chapter 1: Getting Started

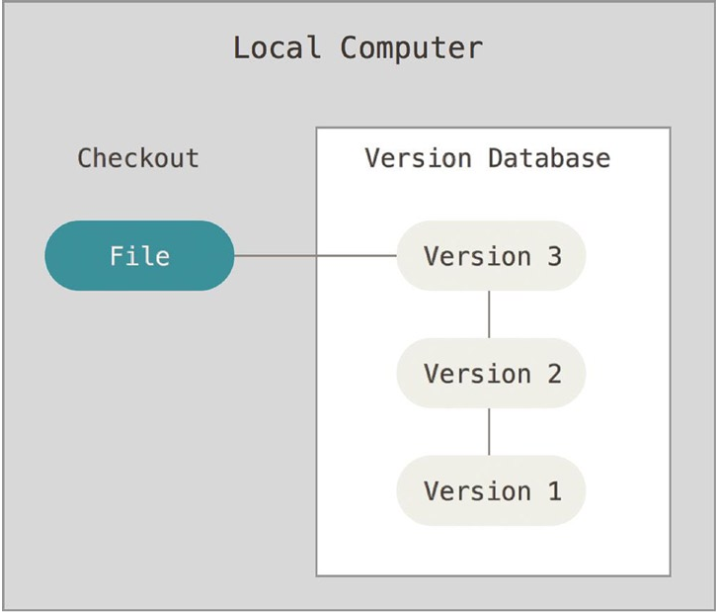
## About Version Control

Cái gì là “version control”, và tại sao bạn cần để tâm tới nó? Version control là một hệ thống cái mà ghi lại các thay đổi một file hoặc các files qua thời gian vì vậy bạn có thể lấy lại chính xác versions trước đó.

### Local Version Control Systems

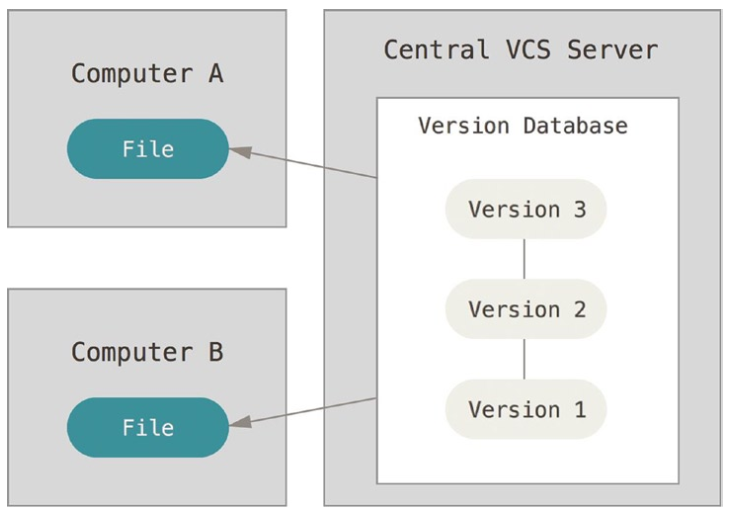
Nhiều người lựa chọn cách thức copy các files tới một nơi khác và lưu nó như là capture version. Cách này phổ biến bởi vì nó đơn giane, nhưng nó dex bị lỗi. Quên cmn mất thư mục nào viết sai file hoặc copy tất cả các files bạn không nghĩ tới.

Để giải quyết vấn đề này các lập trình viên từ lâu đã phát triển local VCSs cái này là một database đơn giản là lưu tất cả các thay đổi các files dưới sự control revision.



### Centralized Version Control Systems

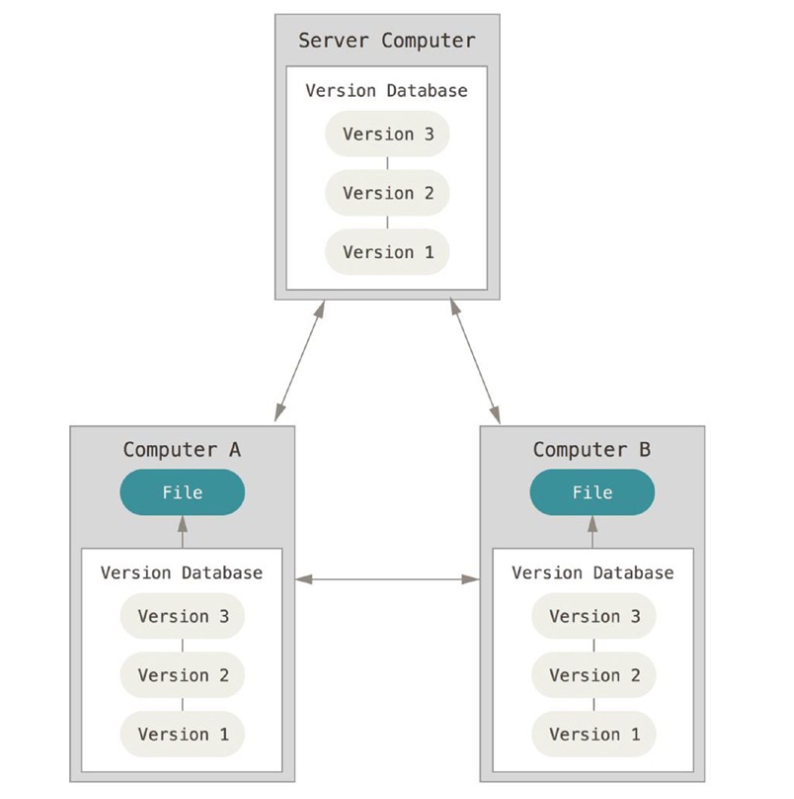
Vấn để gặp phải là giờ nhiều developers cần làm việc với nhau. Để giải quyết vấn đề này, Centralized Version Control Systems (CVCSs) đã được phát triển. Hệ thống này cũng giống như CVS, Subversion, and Perforce, có một single server chứa tất cả version các files và các clients check out files. Sau nhiều năm, thì nó đã trở thành tiêu chuẩn của version control.



Thằng này có nhiều vượt trội so với local VCSs. Ví dụ, mọi người đều biết người khác trong dự án đang làm gì. Administrators có quyền kiểm soát chi tiết đối với những người có thể làm.

### Distributed Version Control Systems

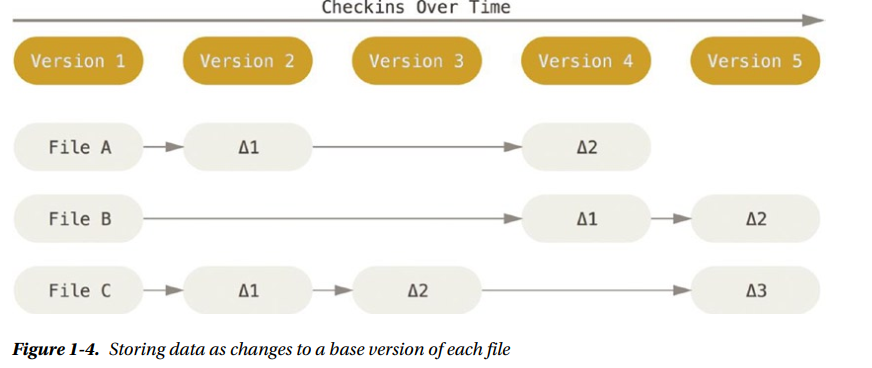
Trong một DVCS (như là Git) các clients họ không chỉ checkout phiên bản mới nhất của các files – họ có full bản sao của repository. Vì vậy nếu server chết thì bất kì client repository nào cũng có thể được copied để back up tới sserver



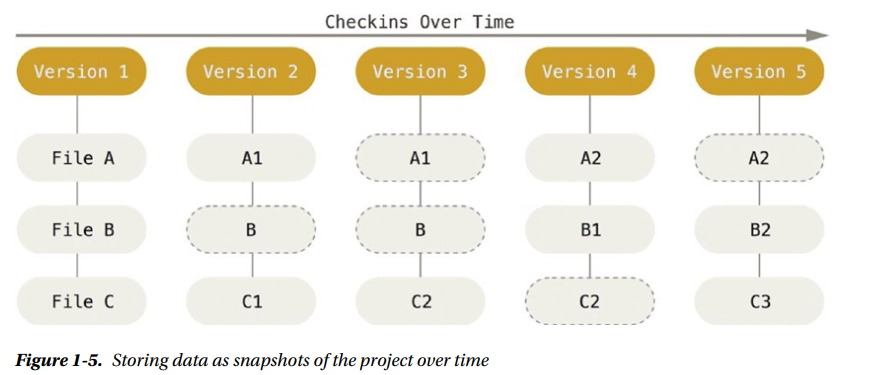
## Git Basics

### Snapshots, Not Differences

Sự khác nhau cơ bản giữa Git và bất kì thằng VCS nào khác là cách mà Git nghĩ về data của nó. Phần lớn các hệ thống lưu trữ thông tin là một list các thay đổi dựa trên các files. Các hệ thống này nghĩ về thông tin của họ là giữ các files và các thay đổi mỗi file trên toàn bộ thời gian.



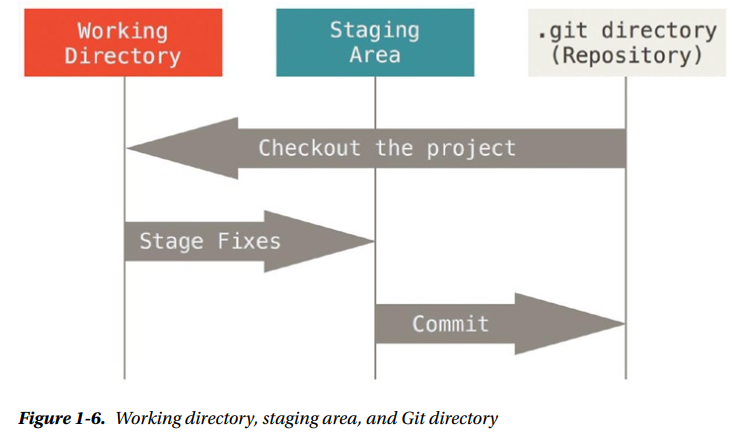
Git không nghĩ việc lưu trữ data có nó theo cách này. Thay vì vậy, Git nghĩ data của nó như là set of snapshots của một hệ thống tệp thu nhỏ. Mỗi lần bạn commit hoặc save trạng thái của project của bạn trong Git, nó chụp lại tất cả các files của bạn tại khoảnh khắc đó và lưu trữ một refrence tới snapshots. Để hiệu quả, nếu các files không bị thay đổi, Git không lưu các file lại, chỉ có một link tới thằng trước đó đã được lưu. Git nghĩ về data như là một stream of snapshots



### The Three States

Git có 3 trạng thái chinh, cái mà các files của bạn có thể được gán đến: commited, modified, and staged. Commited có nghĩa là data thì được lưu một cách an toàn trong local database của bạn. Modified có nghĩa rằng bạn đã thay đổi file nhưng chưa commited nó tới databsae của bạn. Staged có nghĩa rằng bạn đã đánh dấu một file là đã chỉnh sửa trong version hiện hành để nó được snapshot commit tới

Điều này dẫn tới 3 phần chính của một Git project: the Git directory, thư working directory and the staging area



The Git directory là nơi Git lưu metadata và object database cho project của bạn. Đây là phần quan trọng nhất của Git, và nó được copied khi bạn clone một repository từ computer khác.

The working directory là một single checkout of one version of the project (tức là ta đang làm việc trên checkout hiện hành trên repository). Các files được kéo ra khỏi database nén trong thư mục Git và được đặt trên disk cho bạn sử dụng và sửa đổi.

The staging area là một file, thường thì được chứa trong Git directory cuỷa bạn, thằng này lưu thông tin về cái mà sẽ được đưa vào commit tới

Workflow cơ bản của Git sẽ như sau:

1. Bạn chỉnh sửa các files trong working directory
2. Bạn stage các files, thêm snapshots của chúng tới staging area
3. Bạn thực hiện commit, lấy các files trong staging area và lưu trữ snapshot vĩnh viễn vào Git directory của bạn.

### Git Setup

Các biến có thể được lưu trong 3 file ở các vị trí khác nhau:

* /etc/gitconfig – chứa các giá trị mà mọi user trên hệ thống và tất cả các repositories của họ. Nếu bạn truyền vào options –system tới git config, nó đọc và viết file này.
* ~/.gitconfig hoặc ~/.gitconfig/git/config – xác định user của bạn. Bạn có thể đọc và viết file này bằng việc passing –global option
* config file trong Git directory của bất kì repository nào bạn đang sử dụng: chỉ định single repository

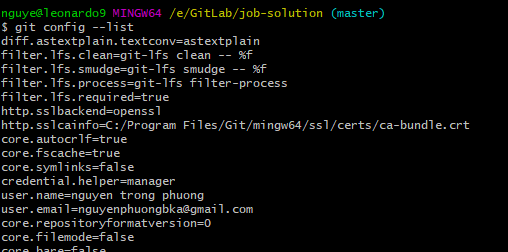
Việc đầu tiên của việc config sau khi cài đặt xong là thiết lập username và e-mail vì Git sử dụng thông tin này để commit

$ git config --global user.name "John Doe"  
$ git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)

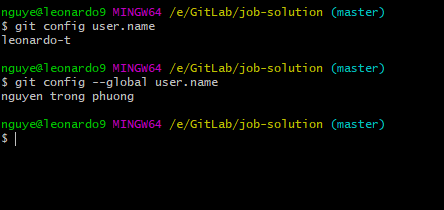
Bạn pass –global option, vì sau đó Git sẽ luôn sử dụng thông tin này tới mọi hệ thống của bạn. Nếu bạn muốn override name hoặc e-mail trên một projects xác định, bạn có thể run command mà không có –global option khi đó bạn trong project.

#### Checking Your Settings

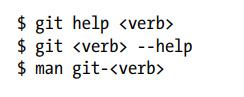
Nếu muốn kiểm tra tất cả các settings, bạn có thể sử dụng git config –list



Ta có thể show một giá trị xác định của variables



#### Getting Help



# Chapter 2: Git Basics

Chapter này covers mọi thứ về basic command bạn cần để làm việc với Git.

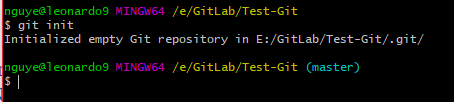
## Getting a Git Repository

Bạn có thể lấy một Git project bằng 2 cách tiếp cận chính. Cách đầu là lấy một project hoặc directory đã tồn tại và imports nó tới Git. Cách thứ 2 là clones một Git repositoru sẵn có trên server khác

### Initializing a Repository in an Existing Directory

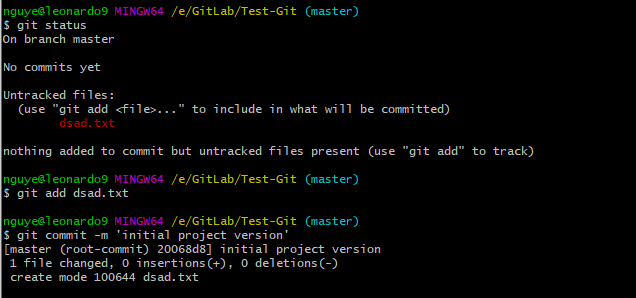
Nếu bạn đang bắt đầu theo dõi một project có sẵn trong Git, bạn cần đi tới directory của project và type

$git init



Thằng này tạo ra một subdirectory được đặt tên là .git chứa tất cả repository files bạn cần – một Git repository xương sống. Tại điểm này, chưa có cái gì trong project của bạn được theo dõi.

Nếu bạn muốn bắt đầu version-controlling với các files đang tồn tại, bạn tracking các files này và làm một initial commit. Bạn có thể sử dụng git add để chỉ định các files bạn muốn track, theo sau cùng là git commit



Tại thời điểm này, bạn có một Git repository với các files đã được tracked và một initial commit.

### Cloning an Existing Repository

Nếu bạn muốn nhận một bản copy của một Git repository có sẵn – ví dụ, một project mà bạn muốn đóng góp – the command bạn cần là git clone. Nếu bạn quen thuộc với các hệ thống khác như VCS, bạn nên chú ý rằng the command là “clone” chứ không phải là “checkout”. Điều này rất quan trọng – thay vì nhận một bản conpy đang làm việc, Git nhận một full copy của tất cả dữ liệu cái mà server có. Mỗi version của mọi file cho history của the project được kéo xuống khi mà ta chạy git clone . Trong thực tế nếu server disk bị hỏng, bạn có thể sử dụng bất cứ clone nào gần nhất trên bất cứ clients nào đó để thiết lập lại server

Bạn clone một repository với git clone [url]. Ví dụ



Thằng này tạo một directory với tên là libgit2, khởi tạo một .git directory bên trong nó, kéo xuống toàn bộ data cho repository, và checkout working copy của version mới nhất. Nếu bạn bạn đi vào libgit2 directory bạn sẽ nhìn thấy các files ở đây, sẵn sằng để làm việc. Nếu bạn muốn clone repository mà đặt tên khác với libgit2 bạn có thể chỉ định option trên command line



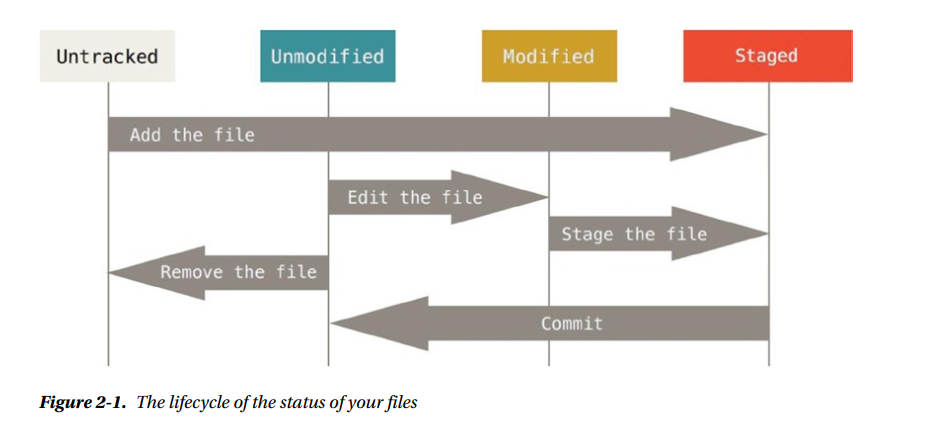
Git có vài giao thức vận chuyển bạn có thể sử dụng. Ví dụ trước chúng ta sử dụng https:// protocol, nhưng bạn có thể nhìn git:// hoặc [user@sserver:path/to/repo.git](mailto:user@sserver:path/to/repo.git), cái này sử dụng giao thức SSH

### Recording Changes to the Repository

Bạn có một Git repository và một checkout hoặc working copy of the files for project. Bạn cần thay đổi và commit snapshots của những sự thay đổi đó vào bên trong repository của bạn mỗi lần mà project muốn tới tới trạng thái bạn muốn lưu lại.

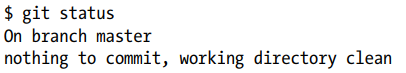
Nhớ rằng mỗi file trong working directory có thể là một trong 2 trạng thái: tracked hoặc untracked. Tracked files là files đã snapshot; chúng có thể là unmodified, modified, hoặc staged. Untracked files là tất cả các fules trong working directory mà chưa được snapshots và không có bên trong stagin area. Khi bạn clone repository, tất cả các file là tracked và unmodified

Bạn edit files, Git nhìn chúng là modified, bởi vì chúng đã thay đổi sau lần commit cuối. Bạn stae các files đã chỉnh sửa này và sau đó commit tất cả staged changes, và vòng lặp lại



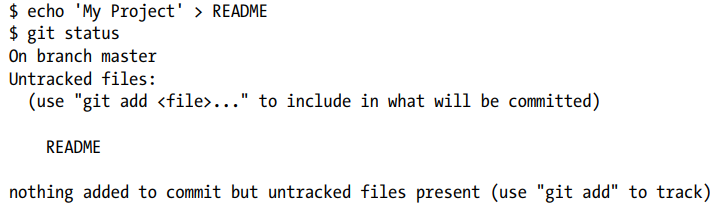
### Checking the Status of Your Files

Nếu bạn chạy command một cách trực tiếp sau khi clone, bạn sẽ nhìn thấy cái này sau khi gõ



Điều này có nghĩa rằng bạn có một clean working directory – nói cách khác, không có tracked và modified files. Git cũng không nhìn thấy bất kì untracked files.

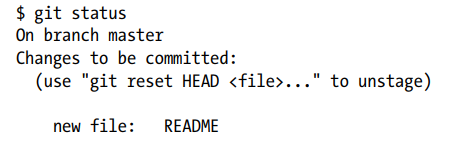
Nếu mà ta thêm một file mới thì như sau



### Tracking New Files

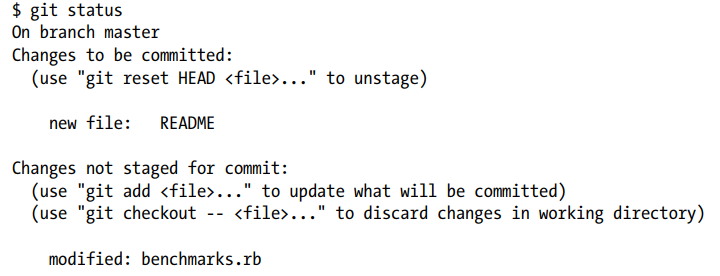
Để mà bắt đầu tracking một file mới, bạn sử dụng the commanr git add. Để bắt đầu tracking the README file, bạn có thể run:





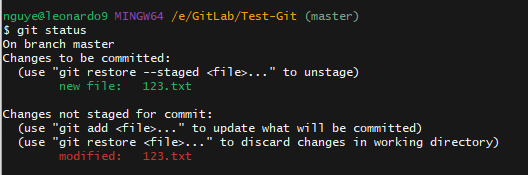
bạn có thể nói rằng nó đã staged bởi vì nó ở dưới the Changes to be commited heading.

### Staging Modified Files

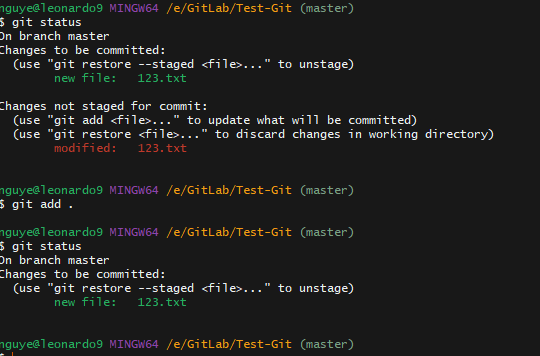




Giả dụ ta có một file mới, sau khi add ta biến nó thành staged sau đó ta lại edit nó và sẽ được như sau



giờ đây ta có nó ở cả 2 trạng thái là staged và unstaged. Nó chỉ ra rằng Git stages một file một cách chính xác khi bạn chạy command git add. Nếu bây giờ bạn commit, the version của 123.txt là khi bạn chạy lần cuối của command git add, không phải version của file nó tìm trong working directory khi bạn chạy commit. Nếu bạn modify một file sau khi bạn chạy git add, bạn phải chạy git add lại một lần nữa để stage (tạo) version mới nhất của file.



### Ingnoring Files

Thi thoảng, bạn có một lớp các files cái mà bạn không muốn Git tự động add hoặc thậm chí là show. Trong nhiều trường hợp, bạn có thể tạo một file list chúng lại được đặt tên là .gitignore. ví dụ: