**CHƯƠNG I: GIỚI THIỆU**

1. **QUẢN LÝ PHIÊN BẢN SỬ DỤNG GIT**
   1. **Giới thiệu về quản lý phiên bản sử dụng Git**

Quản lý phiên bản là một khía cạnh quan trọng trong quá trình phát triển phần mềm. Git là một trong những công cụ quản lý phiên bản phổ biến nhất hiện nay, được sử dụng rộng rãi trong cộng đồng phát triển phần mềm. Git được phát triển bởi Linus Torvalds vào năm 2005 như là một công cụ quản lý mã nguồn cho dự án Linux. Sau đó, nó đã trở thành một công cụ quản lý phiên bản độc lập và có thể được sử dụng cho các dự án phần mềm khác.

* 1. **Tầm quan trọng của Git trong quản lý phiên bản**

Git giúp cho quá trình quản lý phiên bản trở nên dễ dàng và hiệu quả hơn. Nó cho phép các nhà phát triển phần mềm theo dõi, quản lý và kiểm soát các thay đổi của mã nguồn một cách chặt chẽ, đồng thời cho phép các phiên bản khác nhau của sản phẩm tồn tại đồng thời.

Git là một trong những Hệ thống Quản lý Phiên bản Phân tán, vốn được phát triển nhằm quản lý mã nguồn (source code) hữu hiệu của Linux.

Trên Git, có thể lưu trạng thái của file khi có nhu cầu dưới dạng lịch sử cập nhật. Vì thế, có thể đưa file đã chỉnh sửa một lần về trạng thái cũ hay có thể hiển thị sự khác biệt ở nơi chỉnh sửa.

Thêm nữa, khi định ghi đè (overwrite) lên file mới nhất đã chỉnh sửa của người khác bằng file đã chỉnh sửa dựa trên file cũ, thì khi đăng (upload) lên server sẽ hiện ra cảnh cáo. Vì thế, sẽ không xảy ra thất bại về việc đã ghi đè lên nội dung chỉnh sửa của người khác mà không hề hay biết.

1. **GIT**
   1. **Sơ lược về Git**

Git là một đại diện tiêu biểu của hệ thống quản lý phiên bản phân tán DVCS mã nguồn mở, một trong những dạng hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay.

Git có những đặc điểm sau:

* Có hướng tiếp cận mới so với các hệ thống Source Control khác như SVN hay CVS truyền thống.
* Có nhiệm vụ theo dõi những thay đổi, chỉnh sửa trong Source Code của người dùng vào mội thời điểm và đồng bộ những Source Code do họ chỉnh sửa lên Server cùng đồng nghiệp.

Ngoài ra, Git còn có thể chạy trên nhiều hệ điều hành khác nhau như Linux, Windows, MacOS,…

* 1. **Lịch sử phát triển Git**

Git được tạo ra bởi Linus Torvalds vào năm 2005. Lúc đó, dự án Linux đang sử dụng một hệ thống quản lý phiên bản tên là BitKeeper để quản lý mã nguồn của mình. Tuy nhiên, khi quyền sử dụng BitKeeper của dự án Linux bị thu hồi, Linus Torvalds quyết định tạo ra một hệ thống quản lý phiên bản của riêng mình.

Ban đầu, Git được sử dụng cho việc phát triển và quản lý mã nguồn của dự án Linux. Tuy nhiên, sau đó Git đã được sử dụng rộng rãi trong cộng đồng phát triển phần mềm và trở thành một trong những hệ thống quản lý phiên bản phổ biến nhất trên thế giới.

Hiện nay, Git được sử dụng trong các dự án phát triển phần mềm lớn như Android, jQuery, Ruby on Rails, và nhiều dự án phần mềm mã nguồn mở khác trên toàn cầu. Git cũng được tích hợp vào nhiều dịch vụ lưu trữ mã nguồn như GitHub, GitLab, Bitbucket, để giúp các nhà phát triển làm việc và quản lý mã nguồn một cách dễ dàng.

* 1. **Ưu điểm và nhược điểm của Git**

Dựa vào những đặc tính và sự phổ biến của Git đối với người dùng, nhất là lập trình viên, ta sẽ đưa ra những lợi ích khi sử dụng Git. Tuy nhiên, một công cụ hay phần mềm nào cũng đều có lợi và có hại. Vì vậy chúng ta sẽ tìm hiểu ưu và nhược điểm của nó.

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| * Hệ thống quản lý phiên bản phân tán (DVCS) với tính năng nhánh và định nghĩa linh hoạt. * Tốc độ xử lý nhanh hơn so với nhiều hệ thống quản lý phiên bản khác. * Khả năng xử lý các xung đột đến từ sự thay đổi dữ liệu giữa các nhánh một cách tốt hơn. * Tính bảo mật cao với các tính năng chống phá hoại, sao chép dự phòng và mã hóa. * Sử dụng dễ dàng và hỗ trợ đa nền tảng. | * Học cú pháp Git ban đầu có thể khó khăn đối với người mới bắt đầu. * Quản lý phiên bản phân tán có thể dẫn đến việc tăng kích thước repository nhanh hơn so với hệ thống quản lý phiên bản tập trung. * Không có tính năng giải quyết xung đột tự động hoàn toàn, yêu cầu sự can thiệp của người dùng. * Không thể xóa các commit trước đó một cách dễ dàng mà không ảnh hưởng đến lịch sử phiên bản. * Có thể gặp phải các vấn đề liên quan đến hiệu suất nếu repository quá lớn hoặc có quá nhiều branch. |

**CHƯƠNG II: CÀI ĐẶT GIT**

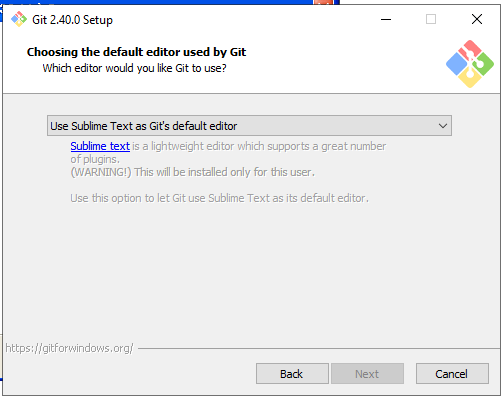
1. **CÀI ĐẶT GIT TRÊN WINDOWS**

Windows vốn là hệ điều hành quen thuộc với đại đa số người dùng.

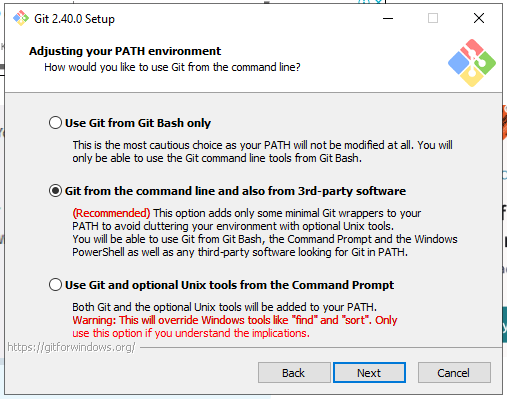
* Bước 1: Truy cập vào đường dẫn: [**https://gitforwindows.org/**](https://gitforwindows.org/)và sau đó ấn vào chữ Download.



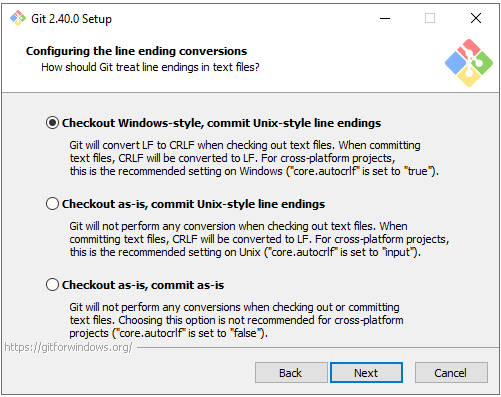
* Bước 2: Sau khi đã tải xuống thành công, click vào file để tiến hành cài đặt.
* Bước 3: Cài đặt bình thường như các phần mềm khác, về các options thì dựa vào yêu cầu của mỗi người để có những lựa chọn phù hợp.
* Bước 4: Cài đặt Git Editor cho Windows, có thể là các text editor như Vin, Notepad++, Sublime Text,…



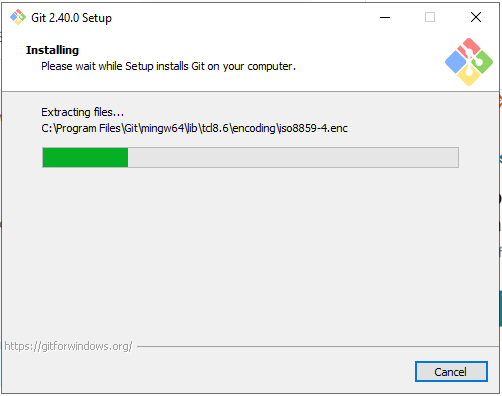
* Bước 5: Lựa chọn PATH môi trường phù hợp với nhu cầu.



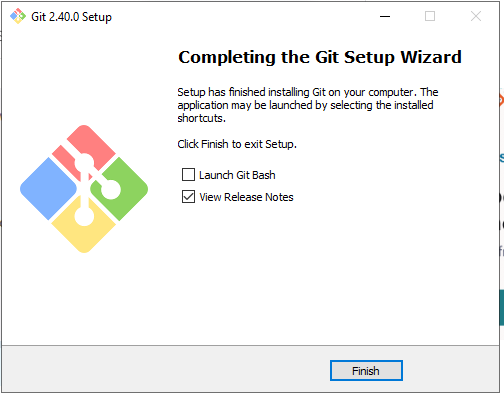
* Bước 6: Chọn phong cách dòng lệnh.



* Bước 7: Bấm Install để cài đặt.



* Bước 8: Kết thúc sau khi đã cài đặt thành công.



1. **CÀI ĐẶT GIT TRÊN LINUX**

Hệ điều hành Linux hỗ trợ Git rất tốt, đặc biệt là với công cụ Terminal mặc định của Linux, giúp Git phát huy được tối đa sức mạnh vốn có. Git sẽ có nhiều cách cài đặt khác nhau dựa trên từng Distro từ nhân Linux. Người dùng truy cập vào đường dẫn: <https://git-scm.com/download/linux> và làm theo đường dẫn sau.

* **Bước 1:** Mở Terminal và gõ 2 lệnh sau.



* **Bước 2:** Gõ lệnh ‘**git-version**’ để kiểm tra đã cài đặt thành công hay chưa.



**CHƯƠNG III: CẤU HÌNH**

1. **CẤU HÌNH GITHUB**
   1. **Tạo Github Repository**

Để làm việc, trước hết cần tạo một kho lưu trữ mã nguồn lệnh cho dự án, việc này về bản chất cũng giống như thao tác trên các dòng lệnh Git. Ở đây, người dùng sẽ làm qua giao diện với các cú nhấp chuột. Các bước tạo Repo ban đầu được mô tả như sau:

* **Bước 1:** Người dùng bấm vào ký hiệu dấu cộng been cạnh ảnh đại diện của tài khoản và chọn New Repository, Github sẽ chuyển qua trang khởi tạo Repo mới.

Graphical user interface, text, application, chat or text message

Description automatically generated

*Hình 3.1: Khởi tạo Github Repository.*

* **Bước 2:** Đặt tên Repo cũng như miêu tả dự án cần thiết.

Graphical user interface, text, application, chat or text message

Description automatically generated

*Hình 3.2: Đặt tên cho Repo.*



*Hình 3.3: Mô tả Repo dự án.*

* **Bước 3:** Chọn một trong hai tùy chọn do Github đưa ra. Hai tùy chọn có nội dung như sau:
  + Hai chế độ phân quyền là Public và Private. Public là chế độ mặc định, cho phép bất cứu ai cũng có thể xem được Repo. Nếu người tạo không muốn công khai có thể chuyển sang chế độ Private.

A screenshot of a computer

Description automatically generated

*Hình 3.4: Hai chế độ phân quyền của Repo Github.*

* + Tập tin README giới thiệu Repo kèm một tệp tin .gitgnore. Đây là một template có sẵn trong Github và cho mọi người tùy chọn.

Graphical user interface, text, application, chat or text message

Description automatically generated

*Hình 3.5: Template tùy chọn Repo Github.*

* **Bước 4:** Bấm nút Creat repository tạo Repo, Github sẽ chuyển trang Repo đã hoàn thành.

Graphical user interface, text, application, email

Description automatically generated

*Hình 3.6: Repo sau khi khởi tạo.*

Trong trưởng hợp người dùng không thêm tùy chọn, Github sẽ mặc định đây là người dùng mới với dịch vụ nó sẽ chuyển dến một trang hướng dẫn cho người dùng này.

Graphical user interface, text, application, email

Description automatically generated

*Hình 3.7: Trang hướng dẫn cho người dùng.*

* 1. **Xóa Github Repository**

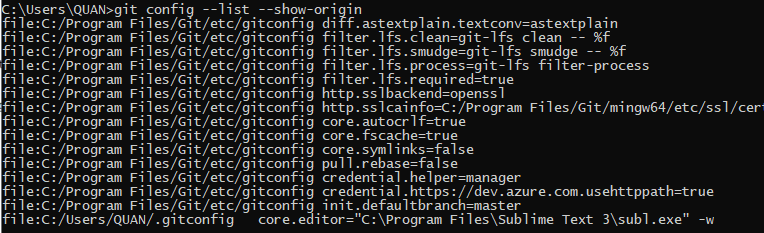
Nếu có nhu cầu xóa Repo thì yêu cầu tài khoản người dùng phải có đặc quyền hoặc là Admin cho một Repo của tổ chức. Các bước xóa Repo trên Github rất đơn giản nhưng có một vài điều cần lưu ý là tất cả các tệp đính kèm cùng phân quyền nhóm dự án đều bị xóa bỏ.

Người dùng có thể khôi phục lại hành động xóa Repo trong 90 ngày.

1. **CẤU HÌNH GIT CƠ BẢN**
   1. **Thiết lập ban đầu**

Điều đầu tiên cần làm sau khi cài đặt git là cấu hình chứng thực, thiết lập các tùy chọn ban đầu phù hợp với cá nhân người sử dụng.

Git có một số công cụ dòng lệnh là git config để người dùng tuy chỉnh các giá trị để git có giao diện và vận hành như họ muốn. Muốn xác định người dùng cần gõ lệnh git config ‘**–list –show-origin**’, nó sẽ hiện thị file config được lưu trên đường dẫn nào, cũng như các giá trị đã được cấu hình mặc định bởi Git.



*Hình 3.8: Xem thông số mặc định.*

* 1. **Danh tính của bạn**

Thông tin đầu tiên cần phải cấu hình trước khi sử dụng Git là tên và địa chỉ email của bạn. Git sẽ sử dụng hai thông tin này để gắn vào mỗi lệnh commit (mục đích để biết ai đã thực hiện commit).

Nên cấu hình danh tính của bạn ở mức global, và chỉ làm một lần. Git sẽ sử dụng thông tin này cho tất cả các thao tác bạn làm trên hệ thống. Nếu muốn sử dụng tên và email khác cho một dự án riêng, thì thực hiện thêm việc cấu hình danh tính tại mức local cho chính dự án đó.

Ví dụ: *$ git config --global user.name "Nam Cong"*

*$ git config --global user.email langyentrach@gmail.com*Mở **.gitconfig** để kiểm tra thông tin vừa cấu hình:

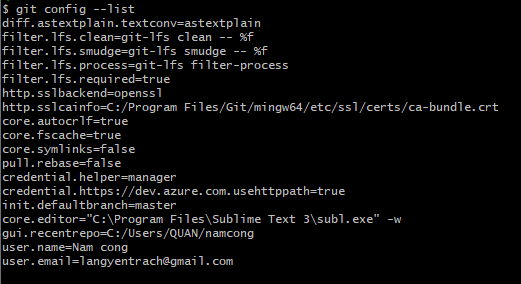
*[user]*

*name = Nam Cong*

*email = langyentrach@gmail.com*

* 1. **Kiểm tra cài đặt**

Git sẽ đọc một giá trị đến từ các file khác nhau, người dùng sẽ gõ lệnh ‘**git config –list**’ để liệt kê các thiết lập mà Git tìm thấy.



*Hình 3.9: Xem thiết lập.*