**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\*\*\*\*\***

A picture containing text, clipart

Description automatically generated

**Chủ đề: TÌM HIỂU VỀ PHẦN MỀM GIT**

**Giảng viên hướng dẫn**: Đỗ Bảo Sơn

**Lớp**: 71DCHT23

**Nhóm thực hiện**: Nhóm 9

**Các thành viên nhóm**: Trần Văn Nam

Vũ Văn Nam

Khuất Thị Thanh Ngoan

Hà Nội, 2021

**MỤC LỤC NỘI DUNG**

1. Giới thiệu phần mềm Git 4
2. Giấy phép phần mềm Git 5
3. Các phiên bản của phần mềm Git 6
4. Các tính năng, đặc điểm, kiến trúc phần mềm 10
5. Tính năng của Git 10
6. Đặc điểm của Git 11
7. Kiến trúc phần mềm Git 13
8. Tải mã nguồn, cài đặt và sử dụng trên Linux 18
9. Cài đặt Git trên Linux 18
10. Sử dụng Git trên Linux 19
11. Tải mã nguồn trên Linux 20

**MỤC LỤC ẢNH**

Hình 1.1. Phần mềm quản lí mã nguồn Git 4

Hình 4.a.1: Tính năng của Git 10

Hình 4.b.1. Phân tán dữ liệu trên Git 12

Hình 4.c.1. Git lưu trữ dữ liệu dưới dạng ảnh chụp (Snapshot) 14

Hình 4.c.2.Hình ảnh các tệp khi lưu trữ trạng thái trong Git 14

Hình 4.c.3. 3 trạng thái của Git 16

Hình 5.a.1. Mở terminal và cập nhật mục cài đặt 18

Hình 5.a.2. Chạy lệnh tải và cài đặt Git 18

Hình 5.a.3. Xác nhận cài đặt Git 19

Hình 5.a.4 Nhập lệnh xác minh cài đặt Git-version 19

Hình 5.b.1. Thiết lập username, email, password và kiểm tra các lệnh 20

Hình 5.b.2. Xóa email, password ở Git 20

Hình 5.c.1. Tải mã nguồn 20

Hình 5.c.2. Tải mã nguồn 21

Hình 5.c.3. Tải mã nguồn 21

1. **Giới thiệu phần mềm Git**



Hình 1.1. Phần mềm quản lí mã nguồn Git

**- Git** là [phần mềm quản lý mã nguồn](https://vi.wikipedia.org/w/index.php?title=Ph%E1%BA%A7n_m%E1%BB%81m_qu%E1%BA%A3n_l%C3%BD_m%C3%A3_ngu%E1%BB%93n&action=edit&redlink=1) [phân tán](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_t%C3%A1n&action=edit&redlink=1) được phát triển bởi [Linus Torvalds](https://vi.wikipedia.org/wiki/Linus_Torvalds) vào năm [2005](https://vi.wikipedia.org/wiki/2005), ban đầu dành cho việc phát triển [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux). Hiện nay, Git trở thành một trong các phần mềm quản lý [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) phổ biến nhất. Git là phần mềm mã nguồn mở được phân phối theo giấy phép công cộng [GPL2](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_C%C3%B4ng_c%E1%BB%99ng_GNU).

**-** Git là tên gọi của một **Hệ thống quản lý phiên bản phân tán** ( Distributed Version Control System) dùng để quản lý và kiểm tra các phiên bản source code khác nhau trong quá trình phát triển.

**-** VCS có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository), mỗi thay đổi vào mã nguồn trên local sẽ có thể ủy thác (commit) rồi đưa lên server nơi đặt kho chứa chính.

**-** 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.

**2.** **Giấy phép phần mềm Git**

**-** GNU GPL v2 and GNU LGPL v2.1

**+** Giấy phép công cộng **GNU** (hay GNU General Public License - GNU GPL) là giấy phép phần mềm tự do phổ biến nhất, đảm bảo cho người dùng cuối tự do chạy, nghiên cứu, sửa đổi và chia sẻ phần mềm, mới đầu do Richard Stallman viết cho dự án GNU.

* **Tư tưởng của GNU GPL** là:
* Phần mềm GPL phải là phần mềm tự do.
* Phần mềm phái sinh từ phần mềm GPL cũng phải là phần mềm GPL.

+ GPL: một khi sử dụng và phân phối, bắt buộc phải sử dụng giấy phép GPL, không được phép đóng mã nguồn và thay đổi giấy phép.

+ LGPL: là giấy phép ít ràng buộc hơn, là giấy phép sửa đổi của GPL,, được sử dụng cho một số thư viện, phần mềm.

**3.** **Các phiên bản của phần mềm Git**

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1Phiên bản** | **Ngày phát hành ban đầu** | **Phiên bản (bản vá) mới nhất** | **Ngày phát hành (của bản vá)** | **Những thay đổi đáng chú ý** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0,99 | 2005-07-11 | 099,9n | 2005-12-15 | |  |
| 1,0 | 2005-12-21 | 1.0.13 | 2006-01-27 | |  |
| 1.1 | 2006-01-08 | 1.1.6 | 2006-01-30 | |  |
| 1,2 | 2006-02-12 | 1.2.6 | 2006-04-08 | |  |
| 1,3 | 2006-04-18 | 1.3.3 | 2006-05-16 | |  |
| 1,4 | 2006-06-10 | 1.4.4.5 | 2008-07-16 | |  |
| 1,5 | 2007-02-14 | 1.5.6.6 | 2008-12-17 | |  |
| 1,6 | 2008-08-17 | 1.6.6.3 | 2010-12-15 | |  |
| 1,7 | 2010-02-13 | 1.7.12.4 | 2012-10-17 | |  |
| 1,8 | 2012-10-21 | 1.8.5.6 | 2014-12-17 | |  |
| 1,9 | 2014-02-14 | 1.9.5 | 2014-12-17 | |  |
| 2.0 | 2014-05-28 | 2.0.5 | 2014-12-17 | |  |
| 2.1 | 2014-08-16 | 2.1.4 | 2014-12-17 | |  |
| 2,2 | 2014-11-26 | 2.2.3 | 2015-09-04 | |  |
| 2.3 | 2015-02-05 | 2.3.10 | 2015-09-29 | |  |
| 2,4 | 2015-04-30 | 2.4.12 | 2017-05-05 | |  |
| 2,5 | 2015-07-27 | 2.5.6 | 2017-05-05 | |  |
| 2,6 | 2015-09-28 | 2.6.7 | 2017-05-05 | |  |
| 2,7 | 2015-10-04 | 2.7.6 | 2017-07-30 | |  |
| 2,8 | 2016-03-28 | 2.8.6 | 2017-07-30 | |  |
| 2,9 | 2016-06-13 | 2.9.5 | 2017-07-30 | |  |
| 2,10 | 2016-09-02 | 2.10.5 | 2017-09-22 | |  |
| 2,11 | 2016-11-29 | 2.11.4 | 2017-09-22 | |  |
| 2,12 | 2017-02-24 | 2.12.5 | 2017-09-22 | |  |
| 2,13 | 2017-05-10 | 2,13,7 | 2018-05-22 | |  |
| 2,14 | 2017-08-04 | 2,14,6 | 2019-12-07 | |  |
| 2,15 | 2017-10-30 | 2.15.4 | 2019-12-07 | |  |
| 2,16 | 2018-01-17 | 2,16,6 | 2019-12-07 | |  |
| 2,17 | 2018-04-02 | 2,17,6 | 2021-03-09 | |  |
| 2,18 | 2018-06-21 | 2.18.5 | 2021-03-09 | |  |
| 2,19 | 2018-09-10 | 2,19,6 | 2021-03-09 | |  |
| 2,20 | 2018-12-09 | 2,20,5 | 2021-03-09 | |  |
| 2,21 | 2019-02-24 | 2.21.4 | 2021-03-09 | |  |
| 2,22 | 2019-06-07 | 2,22,5 | 2021-03-09 | |  |
| 2,23 | 2019-08-16 | 2,23,4 | 2021-03-09 | |  |
| 2,24 | 2019-11-04 | 2,24,4 | 2021-03-09 | |  |
| 2,25 | 2020-01-13 | 2,25,5 | 2021-03-09 | | Dễ dàng quản lý thanh toán thưa thớt |
| 2,26 | 2020-03-22 | 2.26.3 | 2021-03-09 | | * Giao thức phiên bản 2 hiện là mặc định * Một số thủ thuật cấu hình mới * Nội dung cập nhật đối với git Slice-checkout |
| 2,27 | 2020-06-01 | 2.27.1 | 2021-03-09 | |  |
| 2,28 | 2020-07-27 | 2.28.1 | 2021-03-09 | | * Giới thiệu init.defaultBranch * Bộ lọc Bloom đường dẫn đã thay đổi |
| 2,29 | 2020-10-19 | 2.29.3 | 2021-03-09 | | * Hỗ trợ SHA-256 thử nghiệm * Refspec tiêu cực * Thủ git shortlogthuật mới |
| 2,30 | 2020-12-27 | 2.30.2 | 2021-03-09 | | * Userdiff cho bản cập nhật PHP, bản cập nhật Rust, CSS * Tập lệnh hoàn thành dòng lệnh (trong Contrib /) đã biết rằng "git stash show" nhận các tùy chọn "git diff". |
| 2,31 | 2021-03-15 | 2.31.1 | 2021-04-02 | | * git difftoolthêm --skip-totùy chọn * --format cải tiến để máy có thể đọc được * git pull cảnh báo để chỉ định rebase hoặc hợp nhất |
| 2,32 | 2021-06-06 |  |  | |  |
| **2,33** | 2021-08-16 |  |  | |  |
| **Truyền thuyết:**  Phiên bản cũ  Phiên bản cũ hơn, vẫn được duy trì  **Phiên bản mới nhất** | | |  |

**4. Các tính năng, đặc điểm, kiến trúc phần mềm của Git**

**a. Tính năng của Git:**

Hình 4.a.1: Tính năng của Git

+ Tiết kiệm thời gian: Như đã giới thiệu ở trên, Git thực hiện công việc với tốc độ rất nhanh, giúp chúng ta có thêm thời gian để thực hiện các công việc khác.

+ Làm việc offline: Với Git, hầu hết mọi việc đều có thể thực hiện trên máy cục bộ: tạo một hoàn tác, xem lịch sử của dự án, tạo ra các nhánh hoặc gộp chúng lại… Git cho phép ta quyết định nơi nào và khi nào muốn làm việc.

+ Khôi phục khi gặp lỗi: Khi gặp lỗi, ta có thể “undo” cho công việc đó, thậm chí có thể khôi phục những gì đã xóa thông qua Reflog, bởi vì Git hiếm khi thực hiện việc xóa vĩnh viễn một nội dung nào đó.

+Yên tâm: Với Git ta có thể hoàn toàn yên tâm là mỗi bản sao của ai đó trong nhóm có trên máy tính là một bản sao có thể sử dụng được. Ngoài ra, hầu hết mọi hành động trong Git chỉ là thêm dữ liệu (rất hiếm khi xóa).

+Tạo ra những hoàn tác (commit) hữu ích: Git giúp ta tránh khỏi sự lộn xộn nhờ tạo ra các hoàn tác chi tiết. Với khái niệm “staging area” duy nhất đã nói ở trên, ta có thể xác định chính xác những thay đổi sẽ được đưa vào trong các hoàn tác tiếp theo, dù chỉ là một dòng thay đổi.

+ Làm việc theo cách của riêng mình: Khi làm việc với Git, ta có thể sử dụng workflow của riêng mình. Ta có thể kết nối với nhiều kho từ xa, phân nhánh thay vì gộp lại, và làm việc với các module con khi ta cần. Hoặc ta cũng có thể dễ dàng làm việc với một kho trung tâm từ xa.

+ Không trộn lẫn các công việc: chia các nhánh làm việc để dễ dàng theo dõi mọi thứ. Git là một công cụ sử dụng việc chia nhánh một cách nhanh chóng và dễ dàng.

+ Nguồn mở và miễn phí: Git được phát hành theo giấy phép nguồn mở của GPL (Giấy phép công cộng chung). Bạn không cần phải mua Git. Nó hoàn toàn miễn phí. Và vì nó là nguồn mở, bạn có thể sửa đổi mã nguồn theo yêu cầu của bạn.

**b. Đặc điểm của Git:**

+ Hỗ trợ mạnh mẽ cho sự phát triển phi tuyến tính - Strong support for non-linear development: Git cho phép và khuyến khích chúng ta nên có nhiều nhánh cục bộ có thể làm việc hoàn toàn độc lập với nhau. Và việc tạo mới, gộp lại hay xóa các kết quả trong quá trình phát triển chỉ mất vài giây.

+ Phát triển phân tán – Distributed development: có nghĩa là thay vì thực hiện “checkout” source code hiện tại, ta có thể “clone” toàn bộ kho lưu trữ (repository).

Diagram

Description automatically generated

Hình 4.b.1. Phân tán dữ liệu trên Git

+ Khả năng tương thích với các hệ thống và giao thức hiệ có - Compatibility with existent systems and protocols: Các kho lưu trữ có thể được xuất bản thông qua Giao thức truyền siêu văn bản - Hypertext Transfer Protocol (HTTP), Giao thức truyền tệp - File Transfer Protocol (FTP) giao thức Git qua ổ cắm hoặc một phương thức để đăng nhập từ xa an toàn, từ máy tính này sang máy tính khác - Secure Shell (ssh).

+ Xử lý hiệu quả các dự án lớn - Efficient handling of large projects: việc tìm nạp phiên bản trước đó từ một kho lưu trữ được lưu trữ cục bộ có thể nhanh hơn một trăm lần so với việc tìm nạp nó từ máy chủ từ xa.

+ Thiết kế dựa trên bộ công cụ - Toolkit-based design: Git được thiết kế như một tập hợp các chương trình được viết bằng C và một số tập lệnh cung cấp các trình bao bọc xung quanh các chương trình đó.

+ Các chiến lược hợp nhất phù hợp - Pluggable merge strategies: Git có một mô hình được xác định rõ ràng về việc hợp nhất không hoàn chỉnh và nó có nhiều thuật toán để hoàn thành nó, đỉnh điểm là thông báo cho người dùng rằng nó không thể hoàn thành việc hợp nhất tự động và cần chỉnh sửa thủ công.

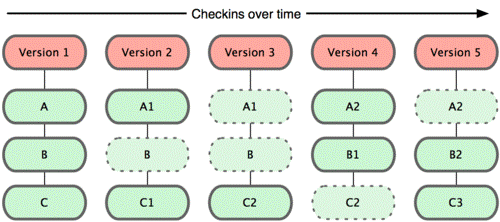
+ Rác tích tụ cho đến khi được thu gom – garbage accumulates until collected: Hủy bỏ các hoạt động hoặc sao lưu các thay đổi sẽ để lại các đối tượng vô ích trong cơ sở dữ liệu. Git sẽ tự động thực hiện thu dọn rác khi đã tạo đủ các đối tượng rời trong kho lưu trữ. Việc thu gom rác có thể được gọi một cách rõ ràng bằng cách sử dụng Git gc

+ Đóng gói đối tượng rõ ràng định kỳ - Periodic explicit object packing: Git lưu trữ mỗi đối tượng mới được tạo thành một tệp riêng biệt. Mặc dù được nén riêng lẻ, chiếm rất nhiều dung lượng và không hiệu quả. Điều này được giải quyết bằng cách sử dụng các gói lưu trữ một số lượng lớn các đối tượng được nén giữa chúng trong một tệp (hoặc luồng byte mạng) được gọi là tệp gói. Một tệp tương ứng được tạo cho mỗi packfile. Các đối tượng mới được tạo vẫn được lưu trữ dưới dạng các đối tượng đơn lẻ và cần đóng gói lại định kỳ để duy trì hiệu quả sử dụng không gian.

**c. Kiến trúc của Git**

**+ Snapshot:**

Như đã nói trong phần giới thiệu về Git, điểm mạnh của Git so với các Distributed Version Control System – DVCS khác chính là việc Git lưu trữ và xử lý dữ liệu. Với các hệ thống DVCS khác, việc lưu trữ thông tin thay đổi của các tập tin được lưu trữ dưới dạng danh sách, thông tin được lưu trữ như là một tập hợp các tập tin và các thay đổi được thực hiện trên mỗi tập tin theo thời gian



Hình 4.c.1. Git lưu trữ dữ liệu dưới dạng ảnh chụp (Snapshot) của mã nguồn dự án theo thời gian

Mà thay vào đó Git lưu dữ liệu dưới dạn một tập hợp các “ảnh” (snapshot). Mỗi lần có “commit”, hoặc thay đổi trạng thái, Git “chụp một bức ảnh” ghi lại nội dung của tất cả các tập tin tại thời điểm đó và tạo ra một tham chiếu tới “ảnh” đó. Trong trường hợp tập tin không có sự thay đổi nào, Git sẽ không lưu trữ tập tin đó mà chỉ tạo một liên kết tới tập tin gốc đã tồn tại trước đó.



Hình 4.c.2.Hình ảnh các tệp khi lưu trữ trạng thái trong Git

Đây chính là điểm tạo ra sự khác biệt giữa Git với các DVCS khác.

**+ Action**

Các thông tin về lịch sử thay đổi các file hoàn toàn nằm trên máy tính của bạn, các thao tác với các tập tin bạn gần như có thể thực hiện ngay lập tức mà không cần lo lắng đến vấn đề tốc độ internet. Ví dụ nếu bạn cần xem lịch sử thay đổi của 1 file ở thời điểm 1 tháng trước, Git sẽ tìm kiềm file đó trên máy bạn và ngay lập tức so sánh sự thay đổi giữa thời điểm hiện tại và 1 tháng trước thay vì việc bạn phải lấy thông tin tập tin đó từ 1 máy chủ từ xa. Đồng nghĩa với việc nếu Internet chỗ bạn quá kém, bạn bị mất kết nốt internet thì cũng không ảnh hưởng gì. Bạn vẫn có thể xem log, commit các thay đổi mới nhất ở local. Đến khi Internet quay trở lại, bạn chỉ đẩy code mới nhất lên.

**+ Tính toàn vẹn**

Các file trong Git đều được băm trước khi lưu và được tham chiếu bằng mã băm đó vậy nên việc bạn bị mất trong khi truyền tải hoặc nhận dữ liệu là điều không thể. Cơ chế băm mà Git sử dụng là SHA-1. Một mã SHA-1 có định dạng như sau : 24b9da6552252987aa493b52f8696cd6d3b00373  
Thực tế, Git không sử dụng tên của các tập để lưu trữ mà bằng các mã băm từ nội dung của tập tin vào một cơ sở dữ liệu có thể truy vấn được.  
Tất các các hành động của bạn đều được Git ghi lại vào trong cơ sở dữ liệu. Mọi hành động này đều có thể khôi phục được, bạn không cần lo lắng đền việc bạn đã làm code lộn xộn sau mỗi lần commit mà không revert lại được code trước đấy. Lưu ý là chỉ sau khi bạn đã commit nhé.

**+ Trạng thái**

- Mỗi file trong Git được quản lý với 3 trạng thái: commited, modified, stage.

* committed : các file của bạn đã được Git lưu trữ thành công
* modified : các file đã bị thay đổi nhưng chưa commit
* stage : đánh dấu sẽ commit phiên bản hiện tại của 1 tập tin đã modified trong lần commit sắp tới

- Trong 1 dự án sử dụng Git sẽ có 3 phần : thư mục Git, thư mục làm việc, staging area.



Hình 4.c.3. 3 trạng thái của Git

* Thư mục Git  
  Nơi lưu trữ các metadata và database cho dự án. Đây là phần quan trọng nhất của Git, được sao lưu về máy tính cá nhân khi bạn clone code từ 1 repository.
* Thư mục làm việc  
  Nơi lưu trữ source code dự án. Dữ liệu được lấy từ database nén trong thư mục Git
* Staging area

- Là một tập tin nằm trong thư muc Git, bao gồm các thông tin về những gì sẽ được commit trong lần commit sắp tới. Hay còn được gọi là index.

**5. Tải mã nguồn, cài đặt và sử dụng trên môi trường Linux**

**a. Cài đặt Git trên môi trường Linux**

* Bước 1: Mở terminal và cập nhật chỉ mục cài đặt gói bằng câu lệnh: **sudo apt update**

Hãy nhập tài khoản mật khẩu nếu được yêu cầu

Text

Description automatically generated

Hình 5.a.1. Mở terminal và cập nhật mục cài đặt

* Bước 2: Chạy lệnh sau để tải về và cài đặt Git: **sudo apt install git**

A picture containing chart

Description automatically generated

Hình 5.a.2. Chạy lệnh tải và cài đặt Git

Nhập Y và Enter để xác nhận cài đặt

A picture containing graphical user interface

Description automatically generated

Hình 5.a.3. Xác nhận cài đặt Git

* Bước 3: Xác minh cài đặt bằng cách nhập lệnh sau sẽ in phiên bản Git: **git –version**

A picture containing text

Description automatically generated

Hình 5.a.4 Nhập lệnh xác minh cài đặt Git-version

**b. Sử dụng Git**

* Thiết lập username cho git:
* git config --global user.username <tên>
* Thiết lập email cho git:
* git config --global user.email <email>
* Thiết lập password cho git:
* git config --global user.password <password>
* Kiểm tra tất cả cái vừa thiết lập:
* git config --list

Text

Description automatically generated

Hình 5.b.1. Thiết lập username, email, password và kiểm tra các lệnh ở Git

* Xóa bỏ email khỏi git:
* git config --global --unset user.email
* Xóa bỏ password khỏi git:
* Git config --global --unset user.password

Graphical user interface, text

Description automatically generated

Hình 5.b.2. Xóa email, password ở Git

1. **Tải mã nguồn**

Graphical user interface, text, application, email

Description automatically generated

Hình 5.c.1. Tải mã nguồn

Graphical user interface, text, application, email

Description automatically generated

Hình 5.c.2. Tải mã nguồn

Graphical user interface, text, application

Description automatically generated

Hình 5.c.3. Tải mã nguồn