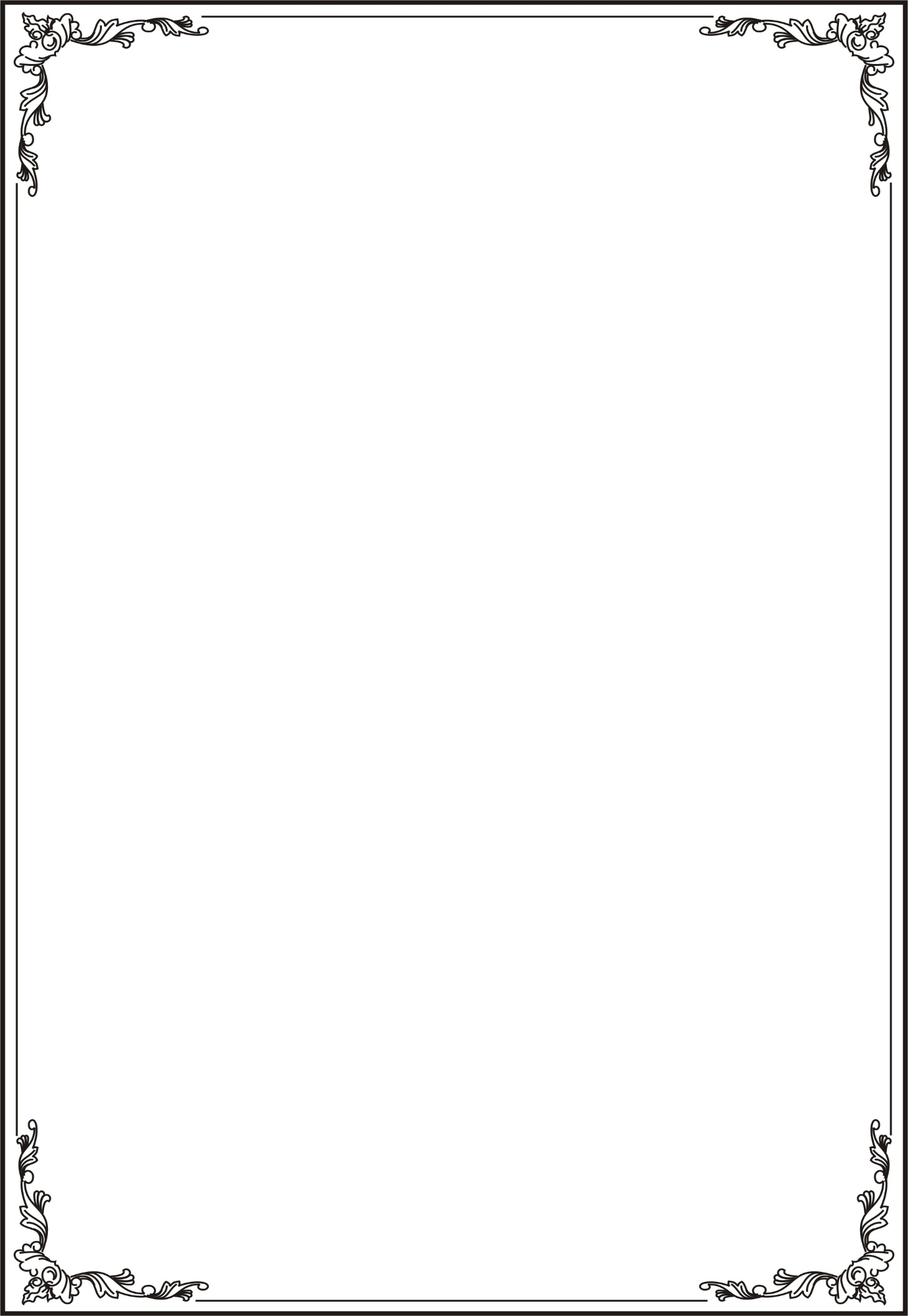
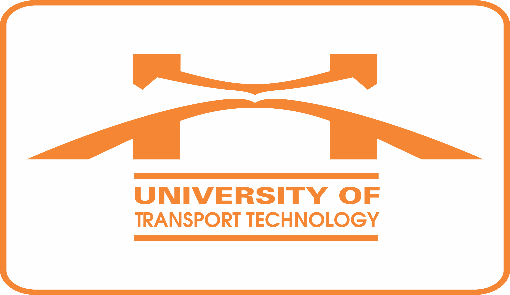
**ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

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

****

**HỌC PHẦN**

**QUY TRÌNH VÀ CÔNG CỤ PHÁT TRIỂN PHẦN MỀM**

**ĐỀ TÀI: PHẦN MỀM QUẢN LÝ QUÁN CÀ PHÊ**

|  |  |
| --- | --- |
| **GIẢNG VIÊN HD :** | **BÙI HỒNG NHUNG** |
| **SINH VIÊN TH :** | **TẠ VĂN ĐỒNG**  **NGUYỄN QUANG HUY** |
|  | **NGUYỄN T.MINH GIANG**  **GIANG NGUYÊN PHONG** |

**Vĩnh Phúc, năm 2024**

|  |  |
| --- | --- |
| Tạ Văn Đồng |  |
| Nguyễn Quang Huy |  |
| Nguyễn Thị Minh Giang |  |
| Giang Nguyên Phong |  |

**PHÂN CHIA CÔNG VIỆC**

**Mục Lục**

# **LỜI CẢM ƠN**

Trước tiên, chúng em xin gửi lời cảm ơn chân thành đến cô Bùi Hồng Nhung đã tạo điều kiện và cung cấp cho chúng em cơ hội tham gia tìm hiểu bài tập lớn môn Báo cáo quy trình và công cụ phát triển phần mềm. Bài tập lớn này đóng vai trò quan trọng trong quá trình đánh giá kiến thức và kỹ năng của chúng em về lĩnh vực này.

Trong báo cáo này, chúng em xin trình bày những kết quả nghiên cứu và phân tích về các khái niệm, phương pháp và ứng dụng. Mục tiêu của báo cáo là trình bày các kiến thức đã học, đánh giá khả năng áp dụng và phân tích các vấn đề liên quan đến môn Báo cáo quy trình và công cụ phát triển phần mềm.

Trong quá trình nghiên cứu và phân tích, chúng em đã gặp phải một số thách thức và khó khăn. Tuy nhiên, chúng em đã nỗ lực vượt qua chúng bằng cách tìm hiểu sâu hơn, tham khảo các tài liệu tham khảo uy tín và thực hành với các công cụ và phần mềm tương ứng.

Cuối cùng, chúng em xin chân thành cảm ơn cô Bùi Hồng Nhung vì đây là một trải nghiệm quý báu và chúng em đã học được nhiều điều từ việc làm việc cùng nhau. Chúng em hy vọng rằng kết quả của bài tập này sẽ đáp ứng được yêu cầu của cô.

Chúng em trân thành cám ơn!

# **LỜI MỞ ĐẦU**

Trong thời đại số ngày nay, ngành quản lý quán cà phê đang chứng kiến sự biến đổi đáng kể nhờ vào sự phát triển không ngừng của công nghệ thông tin. Việc áp dụng các giải pháp phần mềm để tối ưu hoá hoạt động của các quán cà phê không chỉ là cần thiết mà còn là một xu hướng không thể bỏ qua. Sự hiện đại hóa qua phần mềm không chỉ giúp cải thiện hiệu suất vận hành mà còn mang đến cho khách hàng những trải nghiệm tuyệt vời.

Bài viết này tập trung vào việc áp dụng quy trình và công cụ phát triển phần mềm để xây dựng một hệ thống quản lý quán cà phê hiện đại và hiệu quả. Chúng tôi sẽ khám phá quy trình phát triển phần mềm và triển khai hệ thống bằng cách sử dụng công nghệ và công cụ hiện đại. Ngoài ra, chúng tôi cũng giới thiệu các phương pháp quản lý dự án để đảm bảo tính hiệu quả và đáng tin cậy của quá trình phát triển.

Mục tiêu của bài viết không chỉ là giới thiệu sản phẩm mà còn là chia sẻ quá trình học tập và nghiên cứu của chúng tôi trong việc áp dụng kiến thức vào thực tiễn. Chúng tôi hy vọng rằng thông qua bài viết này, cả nhóm và từng thành viên sẽ có cơ hội học hỏi và áp dụng kiến thức môn học vào thực tế, góp phần nâng cao chất lượng quản lý quán cà phê trong bối cảnh công nghệ số ngày càng tiến bộ.

# **CHƯƠNG I: GIỚI THIỆU**

## **1. Mục đích**

Báo cáo này nhằm tìm hiểu và trình bày về sử dụng Git, VPN, SSH, TeamView và SFTP trong quy trình phát triển phần mềm cho đề tài quản lý quán cà phê.

## **2. Phạm vi**

Tập trung vào việc giới thiệu và mô tả các công cụ và giao thức quan trọng sau đây:

−Git và VPN: Hai hệ thống quản lý mã nguồn phiên bản phổ biến được sử dụng để theo dõi và quản lý mã nguồn phần mềm.

−SSH (Secure Shell): Một giao thức mạng được sử dụng để thiết lập kết nối an toàn giữa các máy tính và máy chủ từ xa.

\_ TeamViewer: Một phần mềm cho phép điều khiển từ xa, hỗ trợ hội nghị trực tuyến và chia sẻ tập tin giữa các thiết bị khác nhau thông qua Internet.

Chúng ta sẽ giải thích cách mỗi công cụ hoạt động, cách chúng được sử dụng trong môn học Công cụ và phát triển phần mềm và cách chúng sẽ được kết hợp để đạt được mục tiêu triển khai dự án.

# **CHƯƠNG II: GIT VÀ VPN**

## **2.1. Git**

**Tại sao cần dùng Git?**

Việc sử dụng Git để lưu trữ, kiểm soát phiên bản là cực kỳ cần thiết. Sử dụng Git sẽ không tạo ra xung đột giữa các lập trình viên khi thực hiện code hệ thống. Ngoài ra, với việc các dự án thường xuyên có thay đổi, sử dụng Git sẽ giúp nhà phát triển có thể dễ dàng tìm lại, quay lại các phiên bản cũ hơn. Việc kiểm soát sẽ cực kỳ đơn giản và hiệu quả.

Git cũng có các lợi ích tuyệt vời đối với các dự án đang chạy song song.

Đặc biệt là các dự án có cùng một cơ sở code. Việc sử dụng Git sẽ giúp phân nhánh tốt hơn. Qua đó giúp việc thực hiện các dự án dễ dàng và hiệu quả hơn:

- Thao tác thực hiện nhanh, gọn lẹ hơn, việc sử dụng an toàn hơn.

- Người dùng có thể dễ dàng kết hợp các nhánh với nhau, qua đó giúp đơn

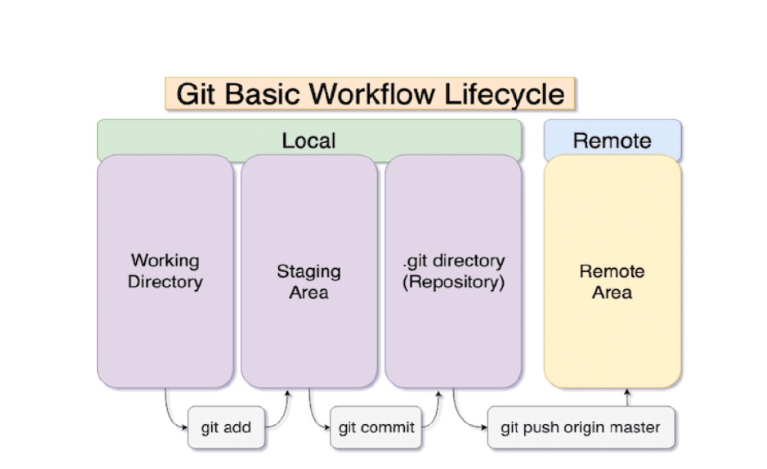
giản hóa quy trình code theo nhóm.

Khi sử dụng Git, người dùng chỉ cần sử dụng clone mã nguồn từ kho chứa/phiên bản nào đó có sẵn trong kho lưu trữ/ một nhánh trong kho lưu trữ. Hệ thống lưu trữ chính xác sẽ cho phép bạn có thể làm việc bất cứ lúc nào, ở bất cứ đâu. Giờ đây, việc lập trình các dự án song song sẽ cực kỳ dễ dàng, hiệu quả và chính xác.

### **2.1.1. Giới thiệu**

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository) riêng chứa toàn bộ lịch sử thay đổi.

### **2.1.2. Cách thức hoạt động**

****

**Working Directory:** đây là bản sao mã cục bộ, nơi có thể thêm sửa, xóa đổi mã.

**Staging area:** là khu vực trung gian giữa thư mục làm việc (working directory) và kho lưu trữ (repository). Khi bạn thay đổi các tệp trong thư mục làm việc, trước khi đưa những thay đổi đó vào kho lưu trữ, bạn cần commit chúng bằng cách thêm vào staging area. Điều này cho phép bạn quản lý rõ ràng việc nào sẽ được bao gồm trong phiên bản tiếp theo của dự án. Sau khi đã đưa thay đổi vào staging area, bạn có thể thực hiện commit để lưu lại các thay đổi này vào kho lưu trữ.

**Repository** (.git folder) là nơi bạn đặt các tập tin sau khi đã sắp xếp chúng, và khi bạn muốn lưu lại sự thay đổi của chúng, bạn chạy lệnh git commit để đưa những tập tin này vào trong kho lưu trữ. Điều này giúp bạn theo dõi lịch sử và quản lý phiên bản của dự án một cách cụ thể và có tổ chức.

**Remote** (Actual git repository): Đó là đích đến cuối cùng của các tệp, khi chúng nằm trong cơ sở mã git của bạn. Nơi bất kỳ ai có quyền truy cập vào mã của bạn đều có thể lấy từ đó.

### **2.1.3. Một số thuật ngữ cơ bản**

**Repository:** Repository là nơi sẽ ghi lại trạng thái của thư mục và file. Bằng việc đặt thư mục muốn quản lý lịch sử thay đổi dưới sự quản lý của repository, có thể ghi chép lại lịch sử thay đổi của thư mục và file trong thư mục đó. Repository của Git được phân thành 2 loại là remote repository và local repository.

−**Remote repository:** Là repository để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng.

−**Local repository:** Là repository bố trí trên máy của bản thân mình, dành cho một người dùng sử dụng.

**Index**: là một khu vực trung gian nằm giữa thư mục làm việc (working directory) và kho lưu trữ (repository). Nó có vai trò là nơi bạn chuẩn bị các thay đổi để sau đó thêm chúng vào kho lưu trữ thông qua một commit.

**Branch**: Các **Branch** (nhánh) đại diện cho các **phiên bản cụ thể** của một kho lưu trữ tách ra từ project chính của bạn.Branch cho phép bạn theo dõi các thay đổi thử nghiệm bạn thực hiện đối với kho lưu trữ và có thể hoàn nguyên về các phiên bản cũ hơn.

**Commit**: Để ghi lại việc thêm/ thay đổi file hay thư mục vào repository thì sẽ thực hiện thao tác gọi là Commit. Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) đã ghi lại sự khác biệt từ trạng thái đã commit lần trước đến trạng thái hiện tại nối tiếp nhau theo thứ tự thời gian.

**Push**: Được sử dụng để đưa nội dung từ kho lưu trữ cục bộ lên server.

**Pull**: Lấy thay đổi từ remote repository về.

**Head**: Các commit ở đầu của một branch được gọi là head. Nó đại diện cho commit mới nhất của repository mà bạn hiện đang làm việc.

**Master**: Nhánh chính của tất cả các repository của bạn. Nó bao gồm những thay đổi và commit gần đây nhất.

**Origin**: Đây là phiên bản mặc định của hệ thống lưu trữ. Origin sẽ đóng vai trò liên lạc với nhánh chính.

**Fetch:** Sử dụng lệnh Fetch sẽ cho phép bạn nạp và tải các bản sao trên hệ thống lưu trữ và tải các tệp nhánh về máy tính cá nhân. Bạn có thể sử dụng lệnh Fetch để lưu các thay đổi mới nhất vào kho lưu trữ. Fetch cho phép bạn có thể lưu trữ nhiều nhánh trong cùng một thời điểm.

**Fork**: là thuật ngữ thể hiện các bản sao của một kho lưu trữ. Tiện ích fork cho phép người dùng - các lập trình viên có thể thử nghiệm các thay đổi. Nhờ việc sử dụng trên bản sao nên người dùng có thể thoải mái thay đổi mà không lo ảnh hưởng tới các dự án chính.

**Tags:** được sử dụng để theo dõi commit quan trọng trong hệ thống. Người dùng có thể sử dụng các tags để chú thích commit, giúp việc tra cứu đối tượng nhanh chóng và đầy đủ hơn.

### **2.1.4. Các lệnh cơ bản**

|  |  |
| --- | --- |
| **Câu Lệnh Git** | **Mô Tả** |
| *git init* | Khởi tạo một kho lưu trữ Git trống. |
| *git clone <URL>* | Clone dự án từ kho lưu trữ xa. |
| *git add <tên\_tệp>* | Thêm tệp/tập tin vào Git (staging) |
| *git add* | Thêm tất cả tệp/tập tin thay đổi vào Git (staging). |
| *git commit -m "Thông điệp commit"* | Lưu các thay đổi vào Git. |
| *git status* | Xem trạng thái sự thay đổi trong thư mục làm việc. |
| *git diff* | Xem sự thay đổi trong các tệp/tập tin |
| *git push origin <tên\_nhánh>* | Gửi các thay đổi lên kho lưu trữ xa |
| *git pull origin <tên\_nhánh>* | Lấy sự thay đổi từ kho lưu trữ xa. |
| *git checkout -b <tên\_nhánh\_mới>* | Tạo một nhánh mới và chuyển đến nhánh đó. |
| *git checkout <tên\_nhánh>* | Chuyển đổi giữa các nhánh. |
| *git checkout <ID\_commit>* | Tạo và chuyển sang một nhánh từ một commit cụ thể (detached HEAD). |
| *git branch -a* | Xem danh sách tất cả các nhánh cục bộ và nhánh từ xa. |
| *git branch -d <tên\_nhánh>* | Xóa một nhánh cục bộ đã được hợp nhất. |
| *git branch -D <tên\_nhánh>* | Xóa một nhánh cục bộ chưa được hợp nhất. |
| *git branch <tên\_nhánh\_mới>* | Tạo một nhánh từ một nhánh khác (không chuyển đến nhánh mới). |
| *git branch* | Xem danh sách các nhánh cục bộ. |
| *git branch -m <tên\_nhánh\_cũ> <tên\_nhánh\_mới>* | Đổi tên một nhánh cục bộ. |
| *git log* | Xem lịch sử commit. |
| *git log <tên\_nhánh>* | Xem lịch sử commit dưới dạng một dòng cho mỗi commit. |
| *git log --oneline* | Xem lịch sử commit dưới dạng một dòng cho mỗi commit |
| *git rm <tên\_tệp>* | Xóa các tệp/tập tin khỏi Git (đã bỏ tracking). |
| *git rm --cached <tên\_tệp>* | Xóa tệp/tập tin khỏi Git (vẫn còn tracking) |
| *git reset <tên\_tệp>* | Bỏ tệp đã thêm ra khỏi staging arae |
| *git stash* | Tạo và áp dụng một stash (lưu trữ tạm thời các thay đổi chưa commit) |
| *git stash apply* | Ap dụng lại stash. |
| *git stash list* | Xem danh sách các stash đã tạo. |
| *git stash drop <stash\_index>* | Xóa một stash cụ thể. |
| *git stash save --keep-index* | Tạo và áp dụng một ghim (chứa thay đổi tạm thời nhưng không xóa khỏi stash). |
| *git stash branch*  *<tên\_nhánh\_mới>* | Tạo một nhánh mới từ stash và áp dụng stash vào nhánh mới. |
| *git stash clear* | Xóa tất cả stash |
| *git show <ID\_commit>* | Xem thông tin về một commit cụ thể |
| *git show <tên\_nhãn>* | Xem thông tin về một nhãn cụ thể |
| *git tag -a <tên\_nhãn> -m "Thông điệp tag" <ID\_commit>* | Tạo và quản lý nhãn (tag) cho các commitcụ thể. |
| *git tag* | Xem danh sách các nhãn đã tạo |
| *git tag -d <tên\_nhãn>* | Xóa một nhãn cục bộ hoặc từ xa. |
| *git tag -a <tên\_nhãn>*  *<ID\_commit>* | Tạo một nhãn (tag) cho một commit cụ thể. |
| *git merge <tên\_nhánh>* | Merge các thay đổi từ một nhánh khác vào nhánh hiện tại. |
| *git fetch -- all* | Lấy tất cả các nhánh trên reponsitory về local |

### **2.1.5. Quy trình làm việc với Git hiệu quả**

Để sử dụng GIT hiệu quả trong các dự án, sẽ có những quy tắc và quy trình làm việc giúp cho quá trình phát triển mã nguồn được đồng bộ, giảm thiểu xảy ra sai sót, xung đột. Các thành viên trong nhóm cần tuân thủ các quy tắc của GIT working flow như sau:

−Repository của dự án bắt buộc phải có hai nhánh là master và develop.

Trong đó nhánh master là nhánh stable chứa code hoàn chỉnh được dùng cho production. Nhánh develop là nhánh có mã nguồn mới nhất về các tính năng được phát triển và đang được deploy lên server để testing.

−Người phát triển phần mềm cần fork repository gốc của dự án về thành một repository của mình, khi phát triển về một tính năng mới hay sửa lỗi, người lập trình cần tạo ra một branch mới từ repository của mình rồi mới thực hiện công việc trên branch đó. Khi hoàn thành công việc thì commit code của mình vào branch sau đó tạo merge request để merge code của mình vào repository gốc của dự án.

−Cần pull code từ repository gốc khi phát triển một tính năng mới để đảm bảo code trên máy mình là code mới nhất từ dự án, tránh xảy ra xung đột (conflict code) sau khi tạo merge request.

−Đối với người có quyền merge code từ các branch khác vào branch

develop cần có review kĩ càng, khi xảy ra conflict phải gọi những người

tham gia vào việc phát triển tính năng ấy để có thể giải quyết một cách tốt

nhất, tránh trường hợp sau khi merge code không chạy được hoặc chạy sai

hoàn toàn so với mục đích của người lập trình.

## **2. 2. VPN**

### **2.2.1. Giới thiệu về VPN**

**a, VPN là gì?**

VPN (Virtual Private Network) là một công nghệ cho phép tạo ra một kết nối mạng an toàn và mã hóa qua một mạng công cộng, thường là Internet. Bằng cách sử dụng VPN, người dùng có thể bảo vệ thông tin cá nhân, truy cập vào các tài nguyên mạng từ xa và vượt qua các hạn chế địa lý hoặc chặn nội dung trực tuyến. VPN cung cấp quyền riêng tư và bảo mật khi truyền dữ liệu, giúp ngăn chặn việc theo dõi hoặc can thiệp từ các bên thứ ba.

**b, Quá trình phát triển của SVN**

**Những năm 1990:**

* **Năm 1996**: VPN lần đầu tiên được phát minh bởi một nhân viên của Microsoft để cung cấp một giải pháp bảo mật cho các kết nối mạng từ xa. Giao thức đầu tiên được phát triển là PPTP (Point-to-Point Tunneling Protocol).

**Những năm 2000:**

* **Đầu năm 2000**: VPN trở nên phổ biến hơn trong doanh nghiệp, giúp nhân viên truy cập an toàn vào mạng nội bộ của công ty từ xa. Các giao thức như IPsec và L2TP (Layer 2 Tunneling Protocol) bắt đầu được sử dụng rộng rãi hơn.
* **Giữa năm 2000**: SSL VPN (Secure Sockets Layer VPN) được giới thiệu, cho phép người dùng truy cập bảo mật vào mạng nội bộ thông qua trình duyệt web mà không cần cài đặt phần mềm VPN riêng biệt.

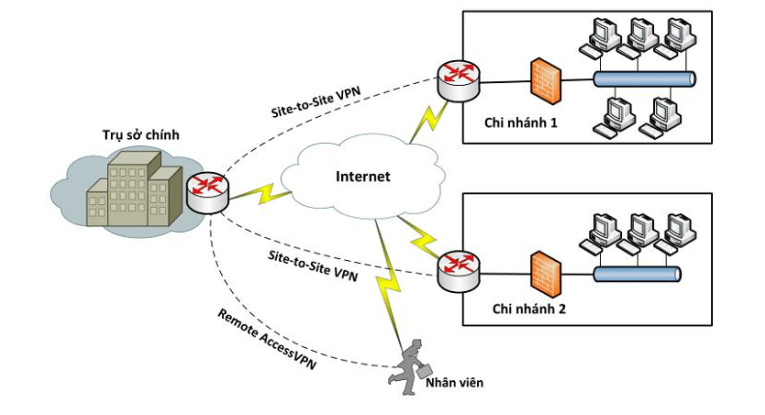
**Những năm 2010:**

* **Đầu năm 2010**: OpenVPN trở thành một trong những giao thức VPN mã nguồn mở phổ biến nhất, nhờ vào tính bảo mật và khả năng cấu hình linh hoạt.
* **Giữa năm 2010**: VPN bắt đầu được sử dụng rộng rãi hơn bởi người dùng cá nhân để bảo vệ quyền riêng tư trực tuyến và truy cập vào nội dung bị hạn chế địa lý. Nhiều dịch vụ VPN thương mại bắt đầu xuất hiện, cung cấp các ứng dụng dễ sử dụng cho người dùng cá nhân.

**Những năm 2020:**

* **Năm 2020**: Sự phát triển của công nghệ mạng và yêu cầu về bảo mật tăng cao dẫn đến sự ra đời của các giao thức mới như WireGuard, được thiết kế để đơn giản, nhanh chóng và bảo mật hơn so với các giao thức truyền thống.
* **Năm 2021-2023**: VPN ngày càng trở nên quan trọng trong bối cảnh làm việc từ xa và bảo vệ dữ liệu cá nhân. Các dịch vụ VPN thương mại không ngừng cải tiến để cung cấp tốc độ nhanh hơn, bảo mật mạnh mẽ hơn và tính năng tiên tiến như bỏ chặn nội dung streaming, bảo vệ chống lại các phần mềm độc hại và theo dõi trực tuyến

### **2.2.2. Cách thức hoạt động**



**Cách thức hoạt động của VPN:**

VPN (Mạng riêng ảo) hoạt động bằng cách tạo ra một đường hầm an toàn và được mã hóa giữa thiết bị của bạn và máy chủ VPN. Khi bạn kết nối với VPN, tất cả lưu lượng truy cập internet của bạn sẽ đi qua đường hầm này, thay vì trực tiếp qua nhà cung cấp dịch vụ internet (ISP) của bạn.

Dưới đây là các bước chi tiết về cách thức hoạt động của VPN:

1. **Kết nối với máy chủ VPN:** Bạn sử dụng phần mềm VPN hoặc ứng dụng trên thiết bị của mình để kết nối với máy chủ VPN. Máy chủ VPN có thể được đặt ở bất kỳ đâu trên thế giới.
2. **Xác thực:** Bạn cần nhập thông tin đăng nhập VPN của mình để xác thực với máy chủ VPN. Thông tin đăng nhập này thường bao gồm tên người dùng và mật khẩu.
3. **Tạo đường hầm VPN:** Sau khi xác thực thành công, một đường hầm VPN được tạo ra giữa thiết bị của bạn và máy chủ VPN. Đường hầm này được mã hóa bằng một giao thức bảo mật, chẳng hạn như OpenVPN hoặc IKEv2.
4. **Lưu lượng truy cập internet được định tuyến qua VPN:** Tất cả lưu lượng truy cập internet của bạn, bao gồm cả email, web và lưu lượng truy cập ứng dụng, sẽ được định tuyến qua đường hầm VPN được mã hóa này.
5. **Giải mã lưu lượng truy cập:** Khi lưu lượng truy cập đến máy chủ VPN, nó sẽ được giải mã và sau đó được gửi đến đích dự định của nó trên internet.
6. **Ẩn địa chỉ IP của bạn:** Khi bạn sử dụng VPN, địa chỉ IP thực của bạn sẽ bị ẩn và thay thế bằng địa chỉ IP của máy chủ VPN. Điều này giúp bảo vệ quyền riêng tư của bạn vì nó khiến các bên thứ ba khó có thể theo dõi hoạt động trực tuyến của bạn.

**Lợi ích của việc sử dụng VPN:**

Có nhiều lợi ích khi sử dụng VPN, bao gồm:

* **Bảo mật:** VPN mã hóa lưu lượng truy cập internet của bạn, giúp bảo vệ dữ liệu của bạn khỏi những kẻ xâm nhập và tin tặc.
* **Riêng tư:** VPN ẩn địa chỉ IP thực của bạn, giúp bảo vệ quyền riêng tư trực tuyến của bạn và khiến các bên thứ ba khó có thể theo dõi hoạt động trực tuyến của bạn.
* **Truy cập nội dung bị chặn:** VPN cho phép bạn truy cập các trang web và dịch vụ bị chặn ở khu vực của bạn.
* **Bỏ qua kiểm duyệt:** VPN cho phép bạn bỏ qua kiểm duyệt internet và truy cập các trang web bị chính phủ hoặc nhà cung cấp dịch vụ internet của bạn chặn.

**Một số lưu ý khi sử dụng VPN:**

* **Không phải tất cả các VPN đều được tạo ra như nhau:** Có rất nhiều nhà cung cấp VPN khác nhau và chất lượng dịch vụ có thể khác nhau. Điều quan trọng là chọn một nhà cung cấp VPN có uy tín và cung cấp bảo mật và hiệu suất tốt.
* **VPN miễn phí có thể không an toàn:** VPN miễn phí thường có hạn chế về băng thông hoặc có thể ghi nhật ký hoạt động trực tuyến của bạn. Nên sử dụng VPN trả phí để có được bảo mật và hiệu suất tốt nhất.
* **VPN có thể làm chậm kết nối internet của bạn:** Mã hóa và định tuyến lưu lượng truy cập internet của bạn qua VPN có thể làm chậm kết nối internet của bạn một chút. Tuy nhiên, hầu hết người dùng không nhận thấy sự chậm lại đáng kể.

### **2.2.3. Quy trình làm việc với VPN hiệu quả.**

Để làm việc hiệu quả khi sử dụng VPN, bạn cần tuân thủ một số quy trình cụ thể. Dưới đây là các bước chi tiết để thiết lập và duy trì một quy trình làm việc hiệu quả với VPN:

**1. Chọn và Cài đặt Dịch vụ VPN Phù Hợp**

* **Lựa chọn dịch vụ VPN**: Chọn một dịch vụ VPN uy tín, đáng tin cậy và phù hợp với nhu cầu của bạn (bảo mật, tốc độ, hỗ trợ kỹ thuật).
* **Cài đặt VPN**: Tải và cài đặt ứng dụng VPN trên thiết bị của bạn (máy tính, điện thoại, máy tính bảng).

**2. Thiết lập Kết nối VPN**

* **Cấu hình VPN**: Đăng nhập vào ứng dụng VPN và cấu hình kết nối theo hướng dẫn của nhà cung cấp dịch vụ. Đảm bảo rằng các thiết lập bảo mật (mã hóa, giao thức) được cấu hình đúng cách.
* **Kết nối VPN**: Kết nối đến máy chủ VPN. Chọn máy chủ gần vị trí của bạn hoặc máy chủ tối ưu để đảm bảo tốc độ và hiệu suất tốt nhất.

**3. Kiểm tra Kết nối và Bảo mật**

* **Kiểm tra IP**: Kiểm tra địa chỉ IP của bạn để đảm bảo rằng bạn đang sử dụng địa chỉ IP của máy chủ VPN (sử dụng các trang web kiểm tra IP như whatismyip.com).
* **Kiểm tra bảo mật**: Kiểm tra xem kết nối VPN có mã hóa dữ liệu của bạn hay không và đảm bảo rằng không có rò rỉ DNS.

**4. Làm việc Qua VPN**

* **Kết nối Git hoặc Các Dịch Vụ Khác**: Sử dụng các lệnh Git hoặc truy cập các dịch vụ cần thiết qua kết nối VPN.

[ git clone https://github.com/username/repository.git

git add .

git commit -m "Your commit message"

git push origin main]

* **Sử dụng Các Công Cụ Làm Việc**: Sử dụng các công cụ làm việc khác như email, truy cập nội bộ của công ty, ứng dụng web, và dịch vụ đám mây qua kết nối VPN để bảo vệ dữ liệu.

**5. Duy trì Kết nối và Giám sát**

* **Giám sát Kết Nối**: Đảm bảo rằng kết nối VPN không bị ngắt trong quá trình làm việc. Nếu kết nối bị ngắt, hãy kiểm tra và kết nối lại ngay lập tức.
* **Cập nhật Ứng Dụng VPN**: Luôn cập nhật ứng dụng VPN và phần mềm bảo mật khác để đảm bảo rằng bạn luôn sử dụng phiên bản mới nhất với các bản vá bảo mật mới nhất.

**6. Ngắt Kết Nối VPN Khi Hoàn Tất**

* **Ngắt Kết Nối**: Sau khi hoàn thành công việc, ngắt kết nối VPN để giảm tải tài nguyên hệ thống và tránh sử dụng băng thông không cần thiết.

[sudo killall openvpn]

**7. Thực Hiện Sao Lưu và Bảo Mật**

* **Sao Lưu Dữ Liệu**: Thực hiện sao lưu dữ liệu quan trọng để tránh mất mát dữ liệu.
* **Bảo Mật Thiết Bị**: Đảm bảo thiết bị của bạn được bảo mật với phần mềm diệt virus và các biện pháp bảo mật khác.

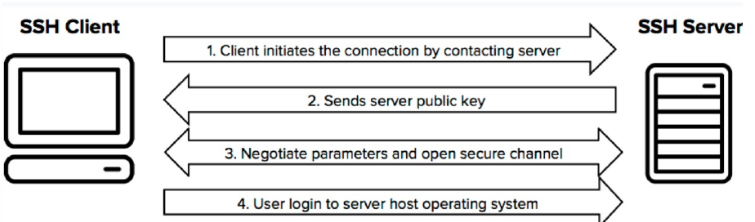
# **CHƯƠNG III: SSH VÀ TEAM VIEW**

## **3.1. SSH**

### **3.1.1. Định nghĩa**

SSH (Secure Shell) là giao thức mạng được mã hoá bảo mật, nó tạo ra kết nối bảo mật (tạo ra một đường dẫn tách biệt với cách dịch vụ khác để duy trì kết nối riêng tư giữa máy trạm và máy chủ) trên một mạng lưới không bảo mật.

Nó cũng được hiểu như ứng dụng dùng để kết nối đăng nhập vào 1 hệ thống máy tính bởi người dùng.



### **3.1.2. Mục đích sử dụng**

Giao thức SSH được dùng ở nhiều nền tảng từ Linux, MacOS, Window với các mục đích như:

- Thiết lập kết nối an toàn giữa hai máy tính, cho phép người dùng truy cập và quản lý máy chủ từ xa một cách bảo mật.

- Mục đích chính là bảo vệ thông tin truyền qua mạng khỏi bị đánh cắp hoặc thay đổi với bất kỳ ai có ý định xấu.

### **3.1.3. Điều kiện và cách sử dụng**

Để sử dụng kết nối SSH, bạn phải được cung cấp một dịch vụ SSH từ máy Remote (từ xa, server), sau đó ở máy Client (local) sử dụng các chương trình SSH Client để kết nối đến máy remote và thực hiện các tác vụ tùy nhu cầu.

**-Máy chủ dịch vụ Remote**: là máy tại đó chạy một SSH Server, nó cung cấp dịch vụ kết nối SSH đến nó, thông qua xác thực tài khoản user/pass hoặc xác thực bằng public/private key.

-**SSH Client**: là các chương trình chạy ở máy trạm (local) có chức năng

kết nối đến SSH Server. Bản thân OpenSSH cũng là Client để kết nối - nên ở máy trạm ta sẽ tập trung vào sử dụng OpenSSH Client để thựcn hành, ngoài ra còn nhiều SSH Client khác, như Putty chạy trên Windows.

Khi kết nối có thể tùy cấu hình từ OpenSSH Server có thể xác thực bằng cách nhập username/password hoặc bằng cặp file public/private key.

### **3.1.4. Cách thức xác thực và bảo mật trong SSH**

**a) Cách xác thực trong SSH**

SSH có nhiều các xác thực một người dùng, nhưng 2 cách thông dụng nhất là xác thực bằng mật khẩu và xác thực bằng public key.

- Xác thực bằng mật khẩu: Xác thực dựa trên mật khẩu đơn giản là bạn chỉ việc sử dụng mật khẩu của user bạn tạo để truy cập, server sẽ lưu chúng, và đối chiếu với mật khẩu của bạn khi đăng nhập. Cách này thì không đủ an toàn do bạn có khả năng bị đánh cắp mật khẩu.

- Xác thực bằng public-key: Cách này sử dụng một cặp khóa – public-key và private-key – được tạo ra dựa trên thuật toán mã hóa public-key. Cặp khóa sau khi được tạo ra từ một máy tính, ta sẽ lấy public-key lưu vào server, khi truy cập ta sẽ dựa vào private-key lưu trên máy local và đặc tính liên quan mật thiết tới nhau của chúng để thiết lập kết nối. Kiểu xác thực này còn cho cho phép chúng ta thiết lập một kết nối an toàn một cách tự động hóa (automation).

**b) Bảo mật trong SSH**

**Mã Hóa:** SSH hỗ trợ nhiều loại thuật toán mã hóa public-key

**-rsa**: thuật toán được sử dụng nhiều nhất, ra đời từ năm 1977 dựa trên sự phức tạp của việc phân tích thừa số nguyên tố. Khi sử dụng nên kèm theo kích thước của khóa ít nhất là 2048 bits, tốt nhất nên là 4096 bits.

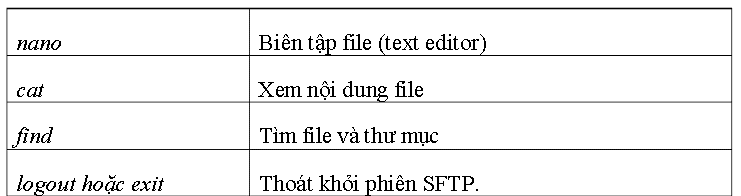
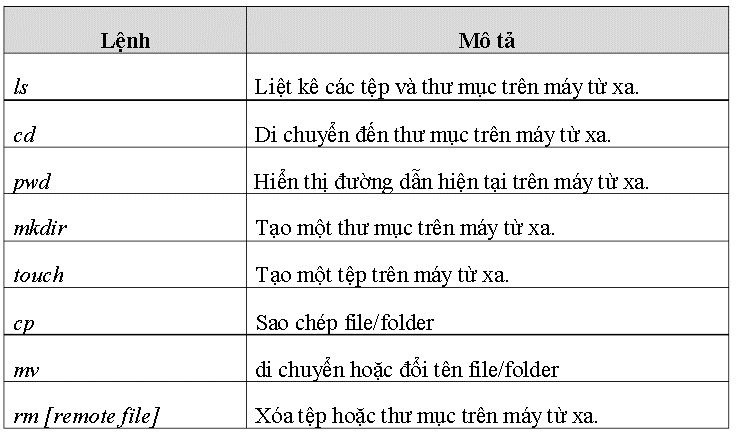
**-dsa:** thuật toán dựa trên tính phức tạp của việc tính toán logarit rời rạc. Đã bị loại bỏ ở OpenSSH version 7 vì lý do bảo mật.

**-ecdsa:** thuật toán dựa trên toạ độ của các điểm dựa trên đường cong Elliptic. Có thể thay thế cho RSA bởi mức an toàn và tốc độ xử lý cao hơn, kèm theo đó là việc sử dụng khoá có độ dài nhỏ hơn so với RSA. Từ đó làm tăng tốc độ xử lý một cách đáng kể. Chỉ hỗ trợ với 3 loại kích thước khóa: 256, 384, and 521 bits. Để an toàn nhất thì nên sử dụng 521bits !

**-ed25519:** một thuật toán được thêm vào OpenSSH từ version 6.5. Là bản cải tiến của ECDSA, cung cấp bảo khả năng mật tốt hơn với hiệu suất nhanh hơn so với DSA hoặc ECDSA. Chưa thực sự phổ cập trên toàn thế giới.

**Bảo Mật Cổng**: SSH thường chạy trên cổng mạng 22. Điều này đảm bảo rằng máy chủ SSH chỉ nghe trên cổng này và giúp ngăn chặn các cuộc tấn công từ các cổng khác.

### **3.1.5. Một số lệnh cơ bản**



## **3.2. TeamViewer**

### **3.2.1. Giới thiệu về Teamviewer**

TeamViewer là phần mềm dùng để điều khiển máy tính từ xa hay còn gọi là remote từ một máy tính khác. TeamViewer cho phép máy tính của người khác kết nối vào máy tính của bạn nhằm mục đích để trao đổi, chia sẻ dữ liệu và người khác có thể sử dụng máy tính của bạn từ xa. Phần mềm này nổi tiếng với lợi ích giúp bạn có thể nhận sự hỗ trợ từ các chuyên gia công nghệ nếu như bạn gặp sự cố với chiếc máy tính để bàn hoặc laptop của mình.

TeamViewer có các phiên bản dành cho máy tính, laptop và cũng có thể là điện thoại thông minh, hỗ trợ nhiều hệ điều hành khác nhau như Windows, MacOS, Linux, IOS, Android,… Chỉ cần nhập đúng mã liên kết cùng với mật khẩu được đối phương cung cấp là người dùng có thể nhận được sự hỗ trợ giải quyết các vấn đề trong học tập và công việc một cách trực tuyến mà không cần gặp mặt trực tiếp.

Với các chức năng ưu việt trên cùng với giao diện thân thiện, dễ dàng làm quen và sử dụng mà TeamViewer từ lâu đã trở thành phần mềm điều khiển máy tính từ xa phổ biến và được ưa chuộng hàng đầu trên toàn thế giới.

### **3.2.2. Cách thức hoạt động**

Cách thức hoạt động của TeamViewer là thông qua mã định danh của máy tính (ID) và mật khẩu được cung cấp (Password) mỗi lần kết nối. Để sử dụng TeamViewer, bạn cần phải cài đặt phần mềm và có kết nối với Internet.

Khi khởi động TeamViewer lên thì ở giữa giao diện ứng dụng sẽ cho bạn biết về mã định danh của máy tính ở phần Your ID, còn mật khẩu kết nối nằm ở phần Password. Bạn chỉ cần gửi cho người bạn muốn kết nối vào máy tính mình hai đoạn mã trên là người đó có thể đăng nhập vào máy tính của bạn rất dễ dàng và nhanh chóng. Bạn cũng có thể làm điều tương tự với đối phương nếu như bạn có mã định danh máy tính và mật khẩu kết nối của người đó bằng cách sử dụng nút Connect.

Cần lưu ý rằng mã định danh của máy tính sẽ luôn luôn cố định nhưng mật khẩu kết nối sẽ thay đổi khác nhau ở mỗi lần kết nối.

### **3.2.3. Ưu điểm của Teamviewer**

* Giao diện thân thiện giúp người dùng dễ dàng làm quen và sử dụng trong thời gian ngắn. các chứ năng trực năng, dễ hiểu và sắp xếp khoa học, thuận tiện thao tác.
* Kết nối nhanh chóng chỉ trong vòng vài giây đầu. Teamviewer có thể chạy sau tường lửa và tự động khi gặp phải những máy tính có cấu hình proxy
* Độ bảo mật cao của phần mềm. Với phương thức kết nối cần phải xác nhận từ hai chiều của thiết bị thong qua ID vs Password cùng với việc giới hạn số luồng truy cập cố định được kết nối với nhau, nói cách khác, chi khi bạn cho phép người khác kết nối vào máy tính của mình việc cung cấp ID vs Passvord thì người đó mới có thể thao tác trên máy tính của bạn được. Rủi ro chỉ xảy ra khi bạn bị người khác lừa, có nghĩa là không phải rủi ro về phần mềm
* Teamviewer là phần mềm miễn phí dành cho người dùng cá nhân sử dụng một các thoải mái
* Kết nối đa chiều giữa các thiết bị với nhau giữa máy tính với máy tính hoặc smartphone
* Hỗ trợ hơn 30 ngôn ngữ khác nhau trên toàn thế giới, trong đó có cả tiếng việt.
  + 1. **Nhước điểm của Teamvewer**
* Miễn phí đối với người dùng cá nhân nhưng đối với đối tượng người dùng doanh nghiệp và thương mại thì chi phí tương đối cao.
* TeamViewer hoạt động có ổn định hay không còn phụ thuộc vào chất lượng đường truyền Internet ở hai đầu thiết bị.
* Việc chia sẻ các tệp dữ liệu lớn qua TeamViewer có thể không ổn định, mất nhiều thời gian hoặc thậm chí không thể thực hiện thành công.
  + 1. **Hướng dẫn cách sử dụng TeamViewer**

**Bước 1:** Sau khi cài đặt TeamViewer, hãy mở ứng dụng lên từ máy tính của bạn.

**Bước 2:** Giao diện TeamViewer hiện ra. Ở chính giữa có hai phần chính:

**Allow Remote Control**: Phần này sẽ hiển thị thông tin kết nối của bạn. Nếu bạn muốn người khác kết nối vào máy tính của mình thì hãy cung cấp thông tin này cho đối phương, bao gồm:

**Your ID**: mã định danh máy tính của bạn

**Password**: mật khẩu kết nối

**Control Remote Computer**: Phần này sẽ giúp bạn kết nối đến máy tính của một người khác bao gồm các tùy chọn sau:

**Remote control** (điều khiển từ xa) và **File transfer** (chia sẻ tệp tin). **Enter Partner ID** là phần bạn sẽ nhập ID từ máy tính bạn muốn kết nối.

**Bước 3**: Để kết nối đến một máy tính khác, bạn nhập ID vào **Enter Partner ID.**

**Bước 4:** Nhấn **Connect**.

**Bước 5**: Hộp thoại **TeamViewer Authentication** hiện ra. Nhập mật khẩu kết nối vào ô **Password**. Sau đó nhấn **Log On**. Như vậy là bạn đã có thể thành công kết nối và có thể điều khiển máy tính đó.

**3.3 Apache Neatbeans**

**3.3.1 Tổng quan về Apache Netbeans**

**3.3.2 Các chức năng của Apache Netbeans**

# **CHƯƠNG IV: TRIỂN KHAI DỰ ÁN**

**4.1. Giới thiệu.**

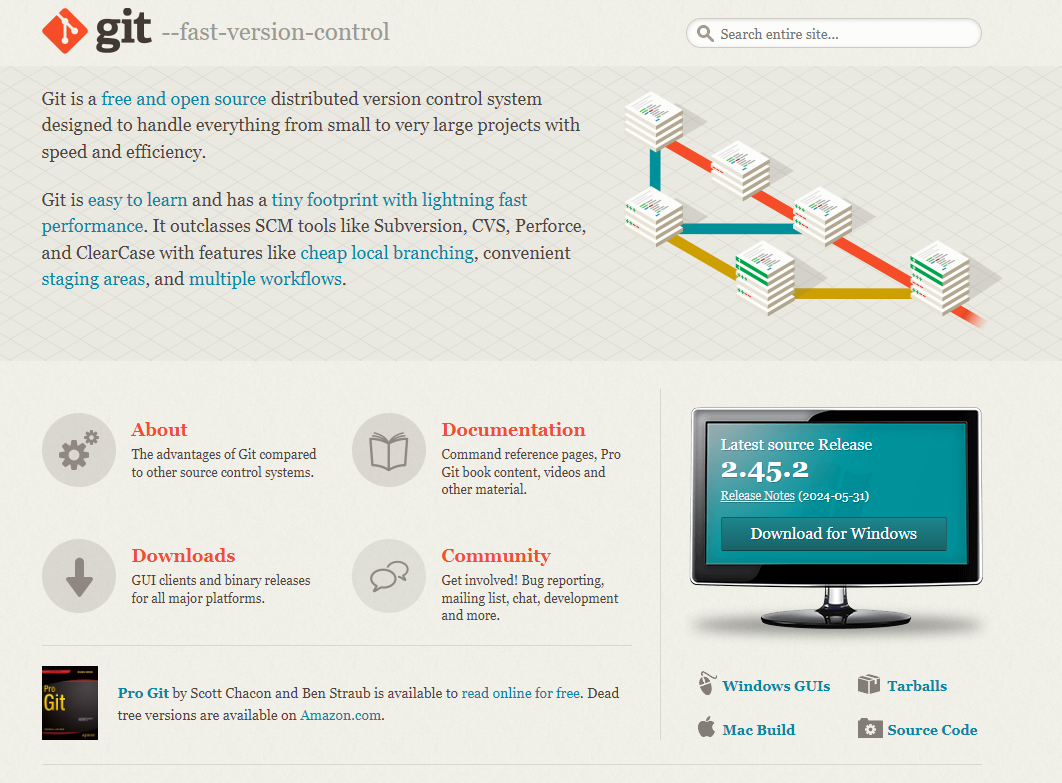
Ở chương 4 này sẽ tập trung vào quá trình và công cụ phát triển phần mềm, đặc biệt là việc đưa projecr “Quản lý quán cà phê” lên Gitthub để quản lý nó trên đó. Trong quá trình này, nhóm chúng em đã học cách sử dụng của một loạt công cụ quan trọng như Git, VPN hay Teamviewer để các thành viên trong nhóm có thể truy cập và lập trình trên một máy chủ chung.

**4.2. Cách triển khai.**

**A Cài đặt Git và Gitbash trên máy tính**

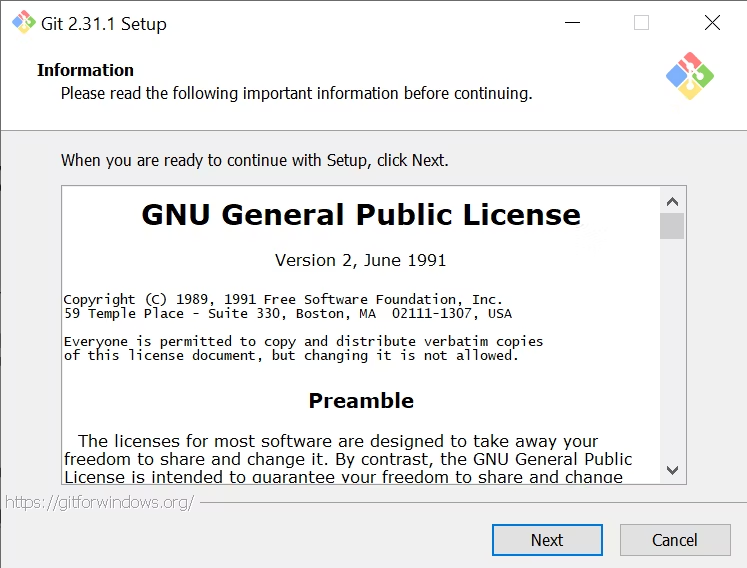
**Bước 1:** Truy cập trang web Git chính thức

Truy cập trang web chính thức của Git và nhấp tải xuống phiên bản cho Windows. Quá trình tải xuống sẽ tự động bắt đầu sau khi ấn nút.



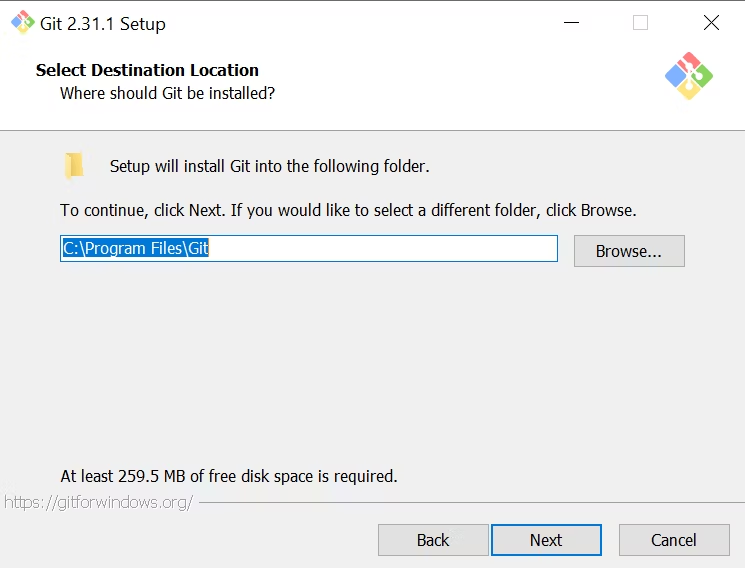
**Bước 2:** Chạy tệp đã tải xuống

Sau khi tải xuống tệp thực thi, hãy nhấp vào tệp đó để chạy trình cài đặt. Một số cửa sổ bật lên yêu cầu quyền truy cập hiển thị. Bấm yes để chấp nhận, cửa sổ cài đặt Git sẽ được mở.



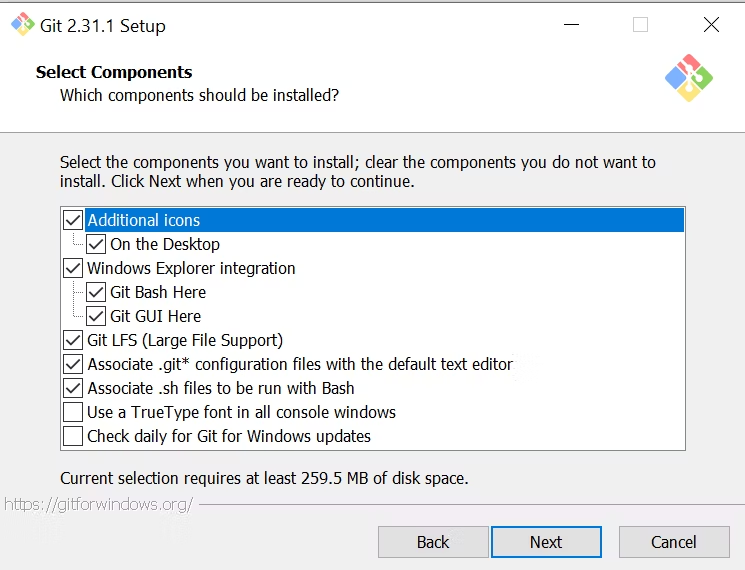
**Bước 3:** Chọn vị trí cài đặt

Nhấn nút Browe để chọn vị trí bạn muốn cài đặt Git.



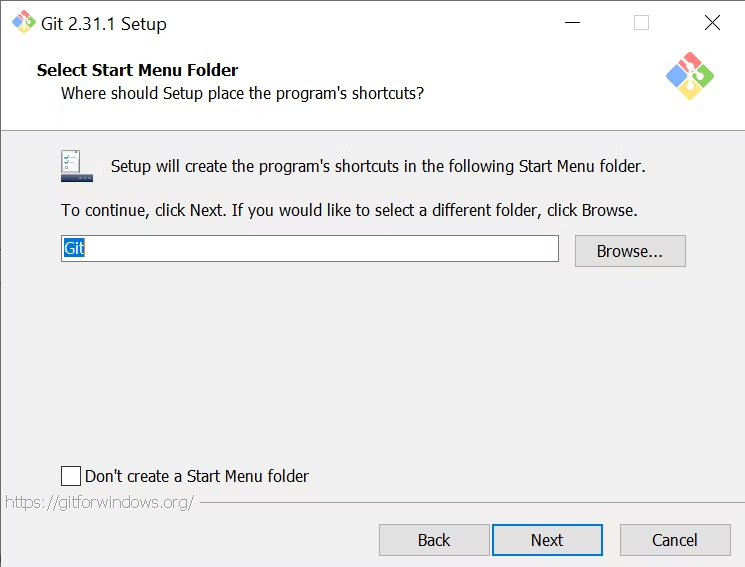
**Bước 4:** Chọn các thành phần

Nhấp vào hộp để cài đặt các thành phần bổ sung như biểu tượng trên màn hình. Đảm bảo hộp “Gitbash here” đã được chọn. Nhấn Next để chuyển sang bước tiếp theo.



### **Bước 5:** Chọn thư mục Menu Bắt đầu

Nếu muốn, bạn có thể thay đổi tên thư mục menu bắt đầu. Nhấn **Next** để tiếp tục các bước tiếp theo.

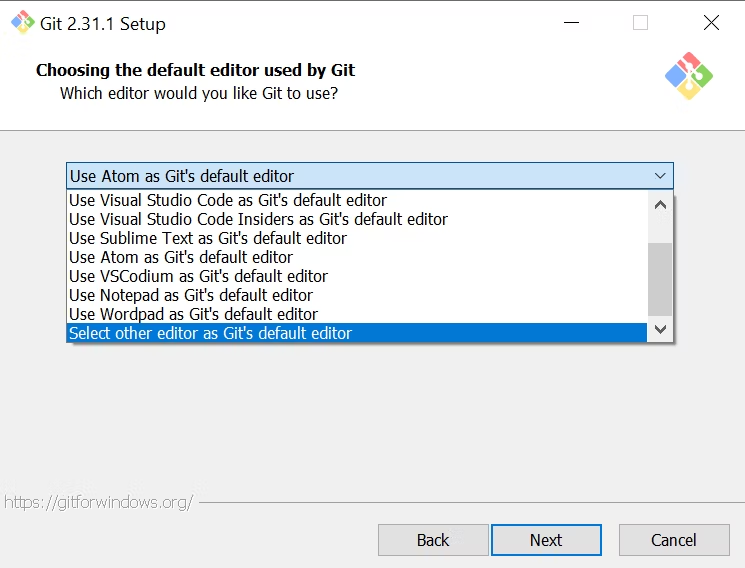


### **Bước 6:** Chọn Trình chỉnh sửa mặc định mà Git sẽ sử dụng

Chọn trình soạn thảo văn bản mặc định mà bạn muốn Git sử dụng. Các tùy chọn có sẵn là Vim, Atom, Visual Studio Code, Sublime Text, Notepad, Wordpad, v.v.

Bạn nên sử dụng Visual Studio Code hoặc Atom làm trình chỉnh sửa mặc định vì chúng được sử dụng rộng rãi nhất và có nhiều tính năng thú vị. Tuy nhiên, Vim hơi phức tạp với người mới bắt đầu.

Nhấp vào nút **Tiếp theo** (Next) để tiếp tục.

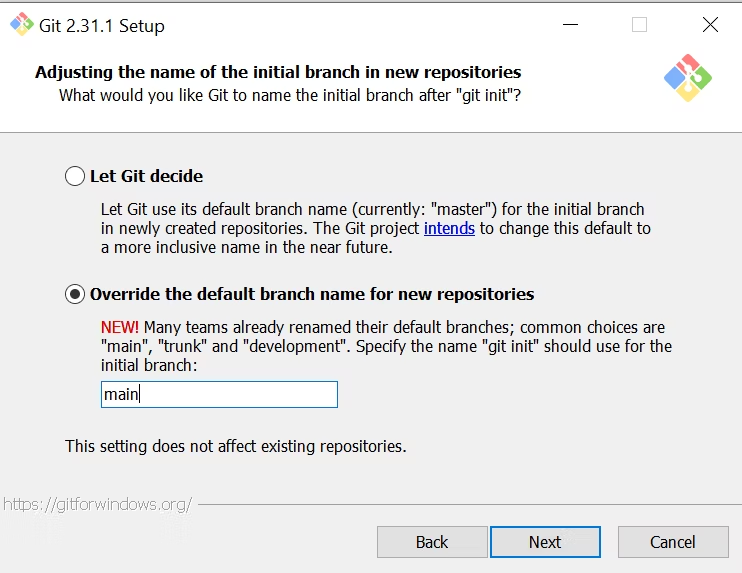


### **Bước 7:** Điều chỉnh tên của nhánh ban đầu trong kho lưu trữ mới

Bạn nên chọn **Ghi đè tên nhánh mặc định cho các kho lưu trữ mới** (Override the default branch name for new repositories) và sử dụng **main** làm tên nhánh ban đầu mặc định.

Lệnh “git init” sẽ sử dụng cùng một tên nhánh ban đầu trong khi khởi tạo các kho lưu trữ. Bạn cũng có thể sử dụng bất kỳ tên nhánh ban đầu nào khác như “mặc định” (default), “phát triển” (develop), “phát hành” (release), v.v.

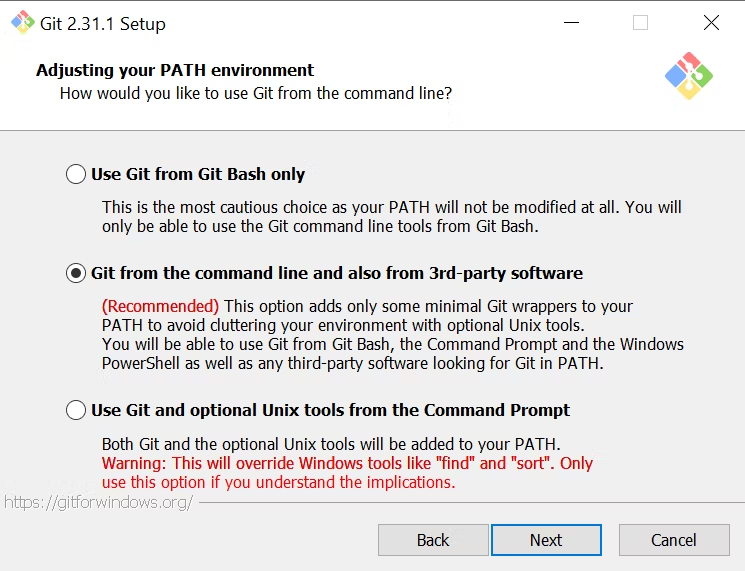
Cuối cùng, nhấp vào nút **Tiếp theo**(Next) để tiếp tục sau khi chỉ định tên chi nhánh.



### **Bước 8:** Điều chỉnh môi trường PATH của bạn

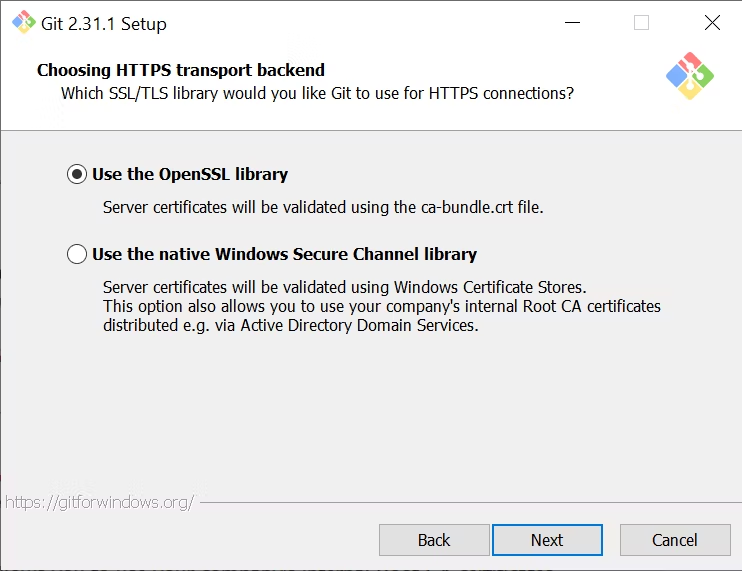
Chọn tùy chọn thứ 2 **Git từ dòng lệnh và cả từ phần mềm bên thứ 3** (Git from the command line and also from 3rd-party software). Bằng cách chọn tùy chọn này, bạn sẽ có thể sử dụng **Git** từ **Git Bash**, Command Prompt, Windows Powershell hoặc bất kỳ phần mềm bên thứ 3 nào khác đang tìm kiếm Git trong PATH.

Nhấn nút **Tiếp theo** (Next) để tiếp tục.



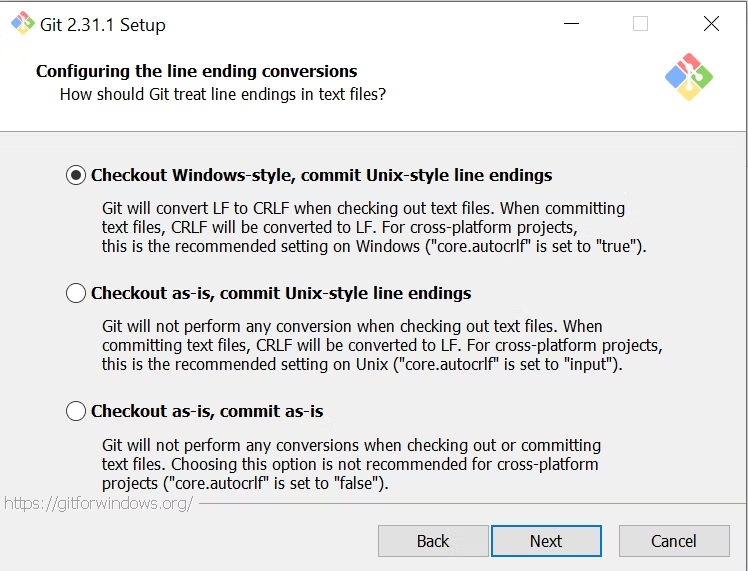
### Bước 9: Chọn HTTPS Transport Backend

Chọn tùy chọn **Sử dụng thư viện OpenSSL** (Use the OpenSSL Library) và nhấp vào **Tiếp theo** (Next).



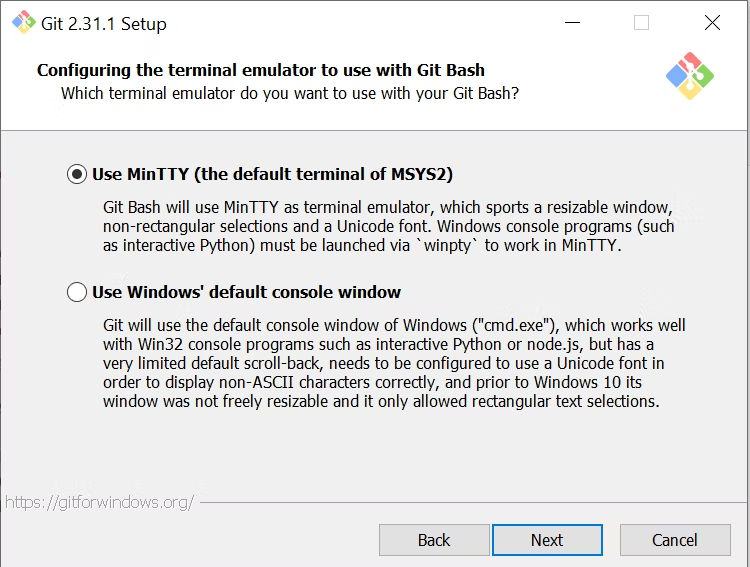
### **Bước 10:** Định cấu hình chuyển đổi kết thúc dòng

Tiếp tục với tùy chọn được chọn theo mặc định **Checkout Windows-style, commit Unix-style line endings** và sau đó nhấp vào **Tiếp theo** (Next).



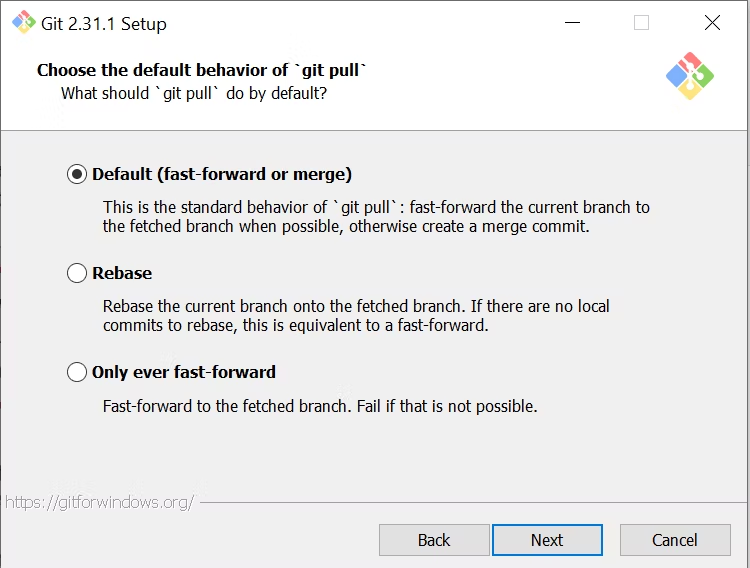
### **Bước 11:** Định cấu hình Trình mô phỏng đầu cuối để sử dụng với Git Bash

Một lần nữa tiếp tục với tùy chọn được chọn mặc định **Sử dụng MinTTY (thiết bị đầu cuối mặc định của MSYS2)** (Use MinTTY (the default terminal of MSYS2)) và sau đó nhấp vào **Tiếp theo** (Next).



### **Bước 12:** Chọn Hành vi mặc định của “git pull”

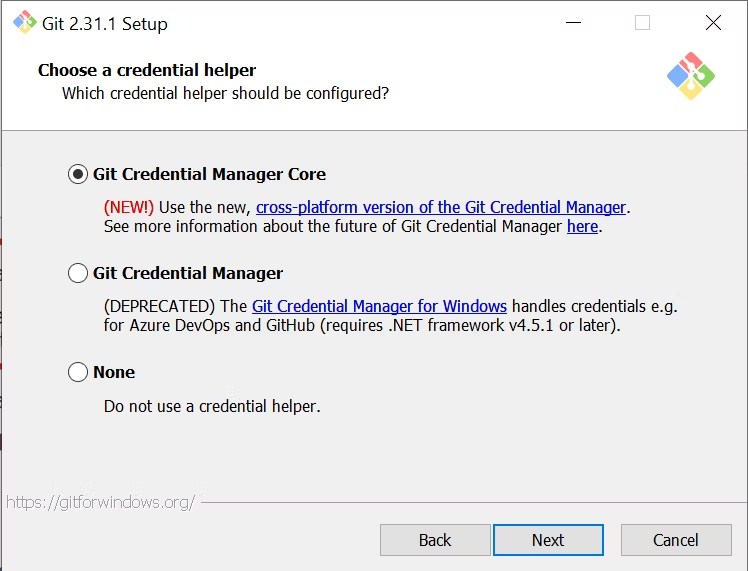
Chọn tùy chọn đầu tiên **Mặc định (tua nhanh hoặc hợp nhất)** (Default (fast-forward or merge)). Bằng cách chọn tùy chọn này, khi “git pull” được sử dụng, nó sẽ chuyển tiếp nhanh nhánh hiện tại sang nhánh đã tìm nạp. Nếu không thể làm như vậy, nó sẽ tạo một merge commit.



### **Bước 13:** Chọn một Credential Helper

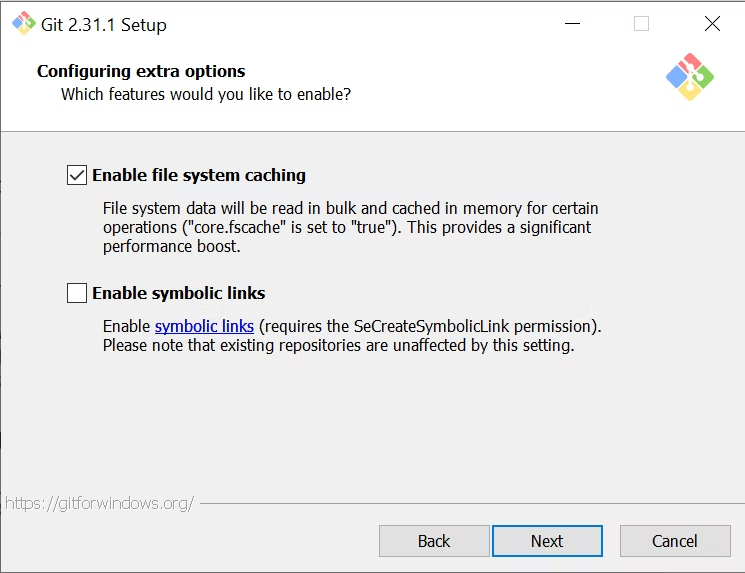
Bạn nên chọn tùy chọn đầu tiên, **Git Credential Manager Core**, vì tùy chọn này cung cấp trải nghiệm xác thực nhất quán trên tất cả các nền tảng.

Sau đó, nhấp vào nút **Tiếp theo** (Next) để tiếp tục.



### Bước 14: Định cấu hình các tùy chọn bổ sung

Tiếp tục với các tùy chọn được chọn theo mặc định và sau đó nhấp vào **Tiếp theo** (Next).

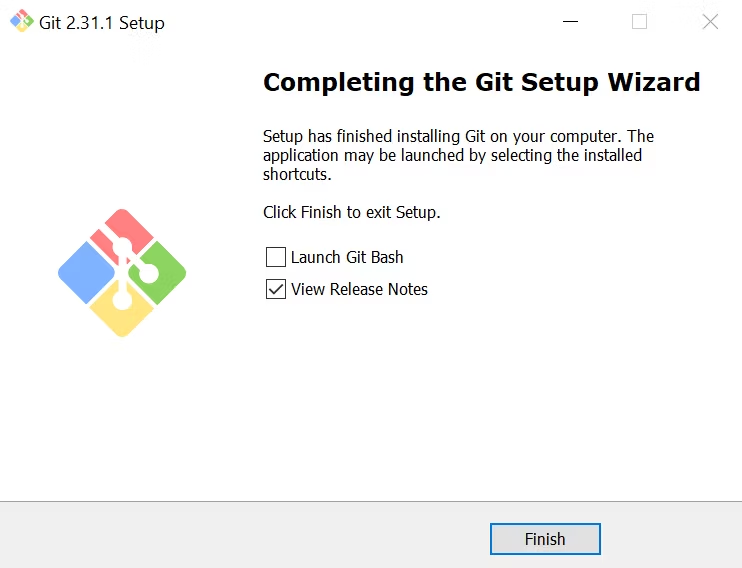


### Bước 15: Định cấu hình Tùy chọn Thử nghiệm

Nếu bạn muốn kích hoạt một số tính năng vượt trội với cài đặt này thì bạn có thể chọn bất kỳ tùy chọn nào có sẵn. Cuối cùng, nhấp vào nút **Cài đặt** (Install).

### Bước 16: Đợi cài đặt

Đợi vài phút để Git và Git Bash được cài đặt trên hệ thống của bạn. Sau khi cài đặt xong nhấn **Finish** để thoát khỏi Setup.

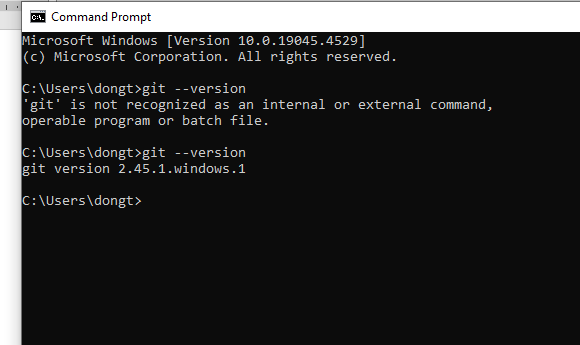


Bây giờ Git và Git Bash đã được cài đặt thành công trên máy của bạn!

## Xác nhận rằng Git đã được cài đặt thành công

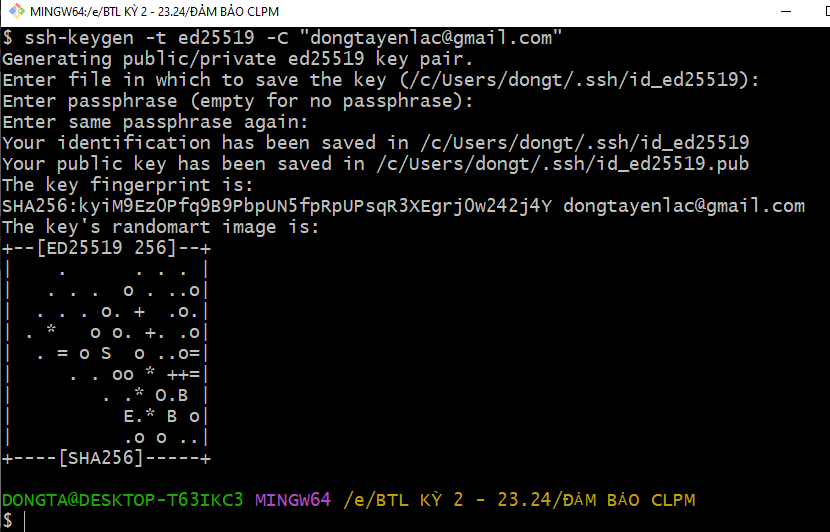
Mở Command Prompt và nhập lệnh sau để xác minh rằng Git đã cài đặt thành công.

git –version

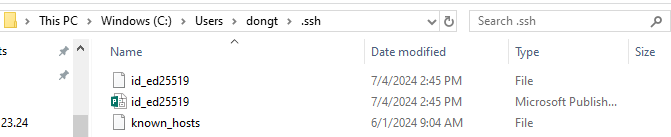


**B) Cấu hình SSH trên Github**

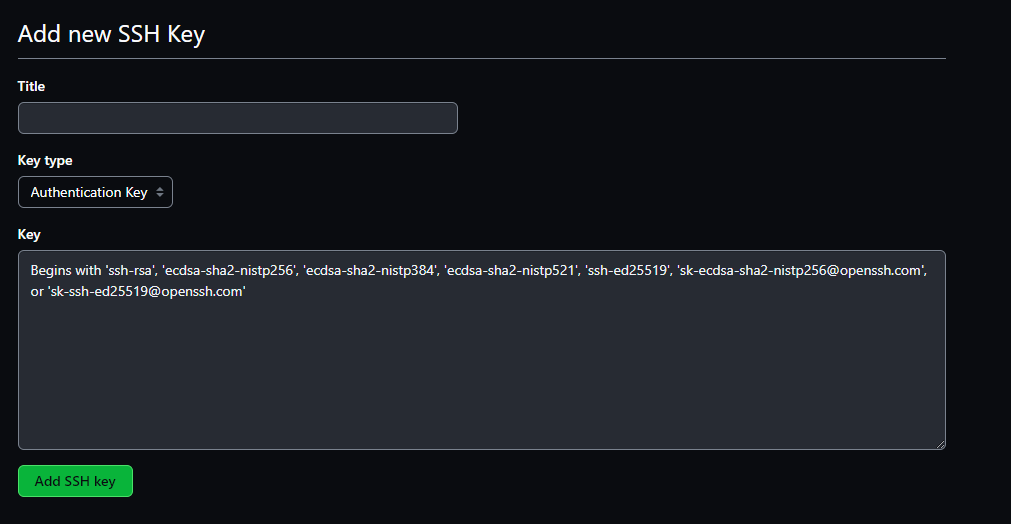
**Bước 1:** Mở Gitbash here và nhập câu lệnh “**ssh-keygen -t ed25519 -C "dongtayenlac@gmail.com"** “



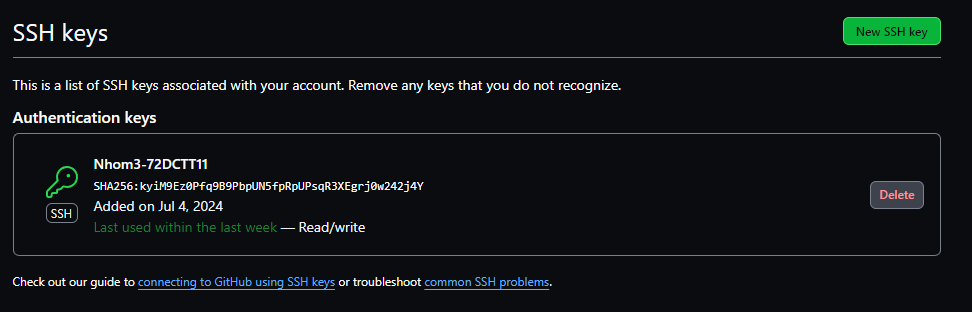
**Bước 2:** Thư mục SSH đã được tạo, chọn thư mục C:\Users\dongt\.ssh để lấy khoá



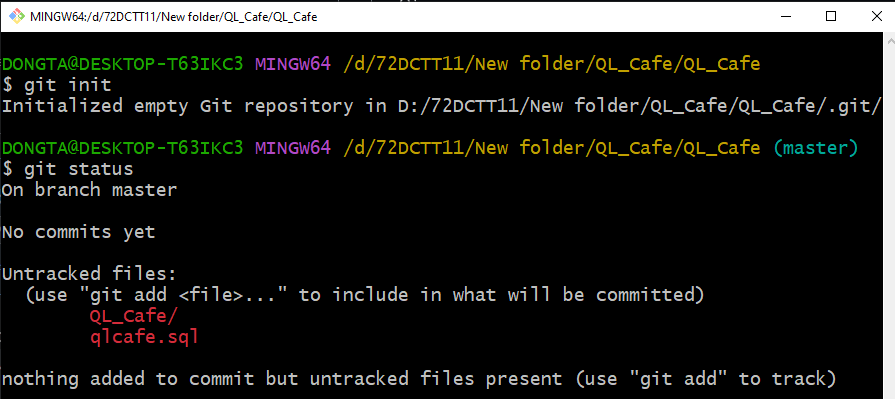
**Bước 3:** Tạo SSH và đặt tên



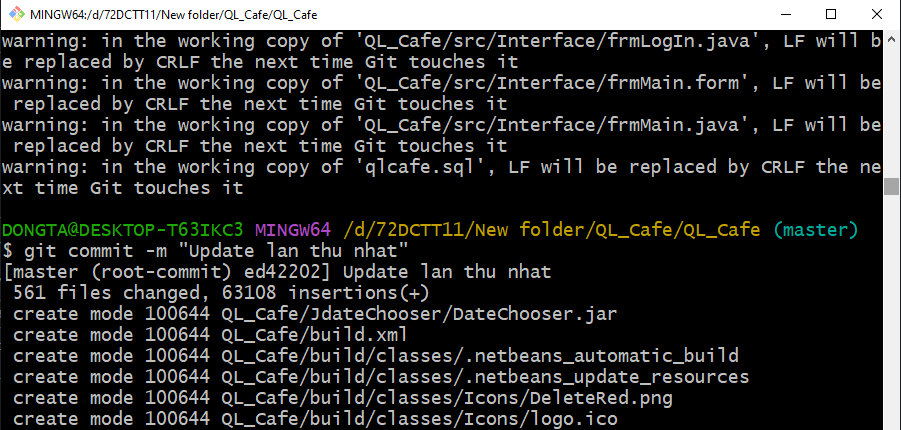
Giao diện sau khi cấu hình SSH thành công:



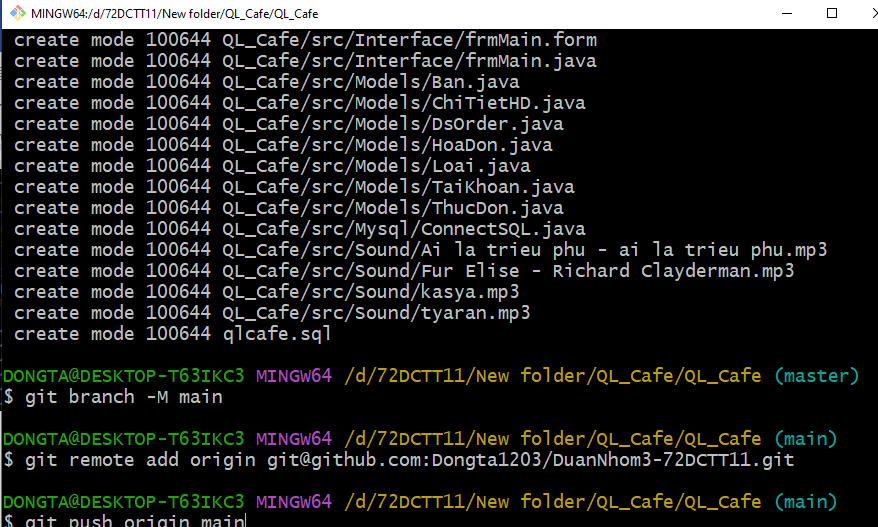
**C)** Đẩy dự án lên Github bằng SSH và tạo nhánh cho từng thành viên



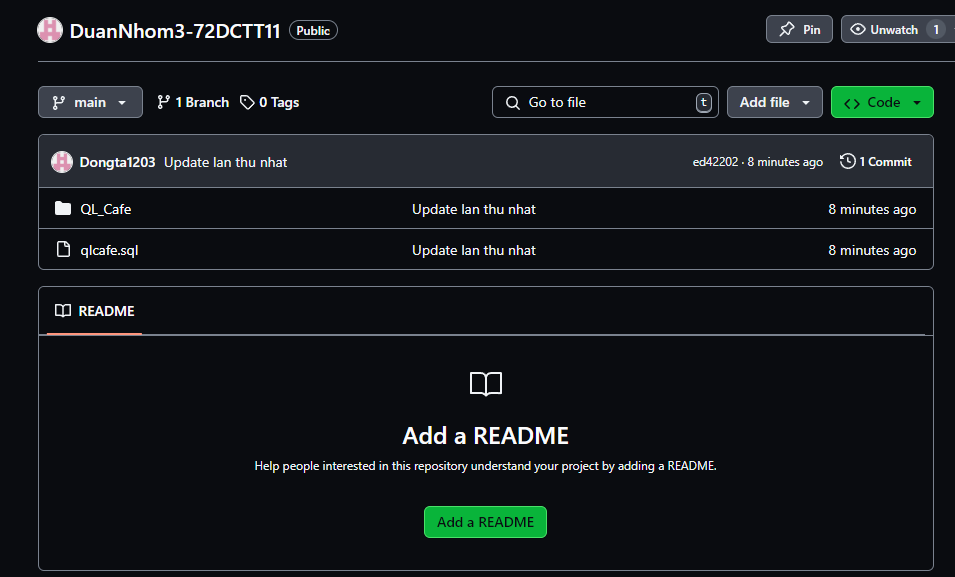
Git status: Kiểm tra trạng thái



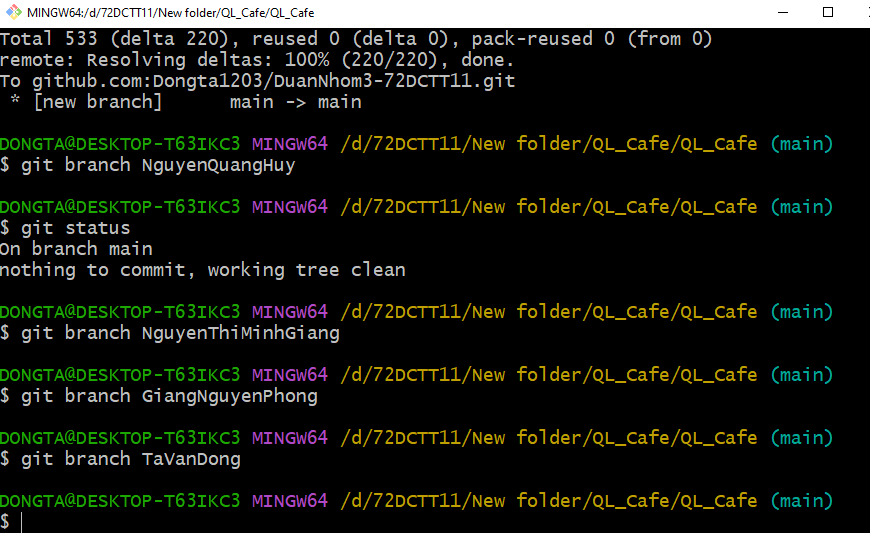
Git commit



Git push: Đẩy code lên Respository



Tạo nhánh cho từng thành viên nhóm để clone phát triển dự án



# **KẾT LUẬN**

# **TÀI LIỆU THAM KHẢO**