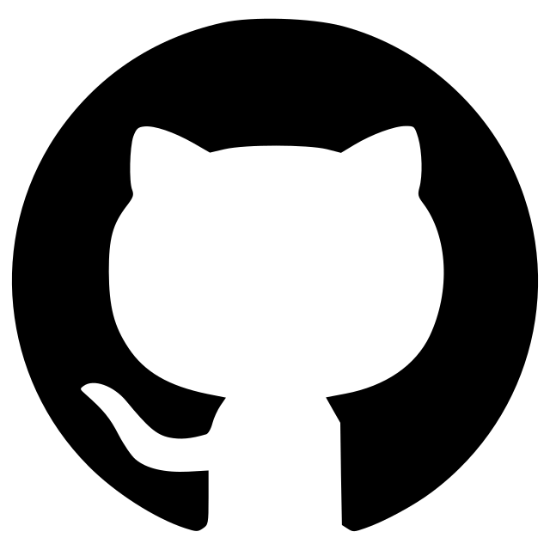
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**- ĐHQG TPHCM**

🙞🕮🙜

Ảnh có chứa biểu tượng, thiết kế

Mô tả được tạo tự động

**BÀI BÁO CÁO**

**KỸ NĂNG NGHỀ NGHIỆP**

**ĐỀ TÀI: “Tìm hiểu về Git – GitHub”**

LỚP HỌC PHẦN: **SS004.O12**

GIẢNG VIÊN HƯỚNG DẪN: **THÁI HUY TÂN**

NHÓM 11: **BREAKTHROUGH**

THÀNH VIÊN:

1. **Trần Đình Khánh Đăng - 22520195**
2. **Minh Nhựt - 22521060**
3. **Bùi Hồng Sơn - 22521246**
4. **Tăng Nhất - 22521027**
5. **Lưu Đoàn Ngọc Phát - 22521070**
6. **Huỳnh Danh Đạt - 2252021**

**TP HỒ CHÍ MINH**, 9/2023

## **MỤC LỤC**

[**DANH SÁCH HÌNH ẢNH** 3](#_Toc146443326)

[**CHƯƠNG 1: TỔNG QUAN VỀ GIT VÀ GITHUB** 5](#_Toc146443327)

[**I.** **GIT** 5](#_Toc146443328)

[**1.** **Khái niệm :** 5](#_Toc146443329)

[**2.** **Cách thức hoạt động :** 5](#_Toc146443330)

[**3.** **Lợi ích:** 7](#_Toc146443331)

[**II.** **GITHUB** 7](#_Toc146443332)

[**1.** **Github là gì?** 7](#_Toc146443333)

[**2.** **Giao diện cơ bản** 10](#_Toc146443334)

[**CHƯƠNG 2: CÁC TẬP LỆNH GIT CƠ BẢN** 26](#_Toc146443335)

[**Giới thiệu các tập lệnh cơ bản của Git, bao gồm:** 26](#_Toc146443336)

[**1.** **Tạo một kho lưu trữ Git mới.** 26](#_Toc146443337)

[**1.** **Kiểm tra trạng thái và thêm các tệp.** 26](#_Toc146443338)

[**2.** **Tạo và xác nhận các thay đổi.** 27](#_Toc146443339)

[**CHƯƠNG 3: CÁC LƯU Ý KHI SỬ DỤNG GIT VÀ GITHUB** 32](#_Toc146443340)

[**1.** **Đặt tên cho nhánh 1 cách rõ ràng:** 32](#_Toc146443341)

[**2.** **Thường xuyên commit và đặt message sáng tạo:** 32](#_Toc146443342)

[**3.** **Tránh commit lớn và không liên quan:** 33](#_Toc146443343)

[**4.** **Thường xuyên “Pull anh Push”:** 33](#_Toc146443344)

[**5.** **Bảo mật và quản lý truy cập:** 34](#_Toc146443345)

[**6.** **Backup và khôi phục dự án:** 36](#_Toc146443346)

[**7.** **Lưu ý về vấn đề khôi phục (restore):** 37](#_Toc146443347)

[**CHƯƠNG 4: TỔNG KẾT** 37](#_Toc146443348)

[**1.** **Kết quả đạt được.** 37](#_Toc146443349)

[**2.** **Các phần chưa đạt được.** 38](#_Toc146443350)

[**3.** **Hướng phát triển** 38](#_Toc146443351)

[**TÀI LIỆU THAM KHẢO:** 39](#_Toc146443352)

## **DANH SÁCH HÌNH ẢNH**

[Hình 1 Mô tả cách hoạt động của hệ thống 4](#_Toc146741778)

[Hình 2 - Mô tả cách hoạt động của Git 5](#_Toc146741779)

[Hình 3 - Github là một trong các nền tảng lưu trữ mã nguồn lớn nhất hiện nay 7](#_Toc146741780)

[Hình 4 - Các phiên bản Git hiện nay 8](#_Toc146741781)

[Hình 5 - Giao diện đăng ký GitHub 9](#_Toc146741782)

[Hình 6 - Giao diện đăng nhập của GitHub 10](#_Toc146741783)

[Hình 7 - Giao diện giành cho người dùng GitHub 11](#_Toc146741784)

[Hình 8 - Phần tạo mới một repository 11](#_Toc146741785)

[Hình 9 - Giao diện phần tạo repository 12](#_Toc146741786)

[Hình 10 - Một số cách Setup repository mà GitHub cung cấp 12](#_Toc146741787)

[Hình 11 - Giao diện tab Code 13](#_Toc146741788)

[Hình 12 - Mục Code 13](file:///D:\Hoc%20Tap\Dai%20Hoc\UIT\HK3\Kỹ%20năng%20nghề%20nghiệp\GitLearning\document_full.docx#_Toc146741789)

[Hình 14 - Mục Go to file 13](#_Toc146741790)

[Hình 13 - Mục Add file 13](file:///D:\Hoc%20Tap\Dai%20Hoc\UIT\HK3\Kỹ%20năng%20nghề%20nghiệp\GitLearning\document_full.docx#_Toc146741791)

[Hình 15 - Phần truy cập và tạo mới branch 14](#_Toc146741792)

[Hình 16 - Giao diện của phần chức năng Releases & Tag 14](#_Toc146741793)

[Hình 17 - Giao diện tab Issues 15](#_Toc146741794)

[Hình 18 - Nút tạo issue 15](#_Toc146741795)

[Hình 19 – Phần filter của tab Issue 16](file:///D:\Hoc%20Tap\Dai%20Hoc\UIT\HK3\Kỹ%20năng%20nghề%20nghiệp\GitLearning\document_full.docx#_Toc146741796)

[Hình 20 - Mục labels và Milestone 16](#_Toc146741797)

[Hình 21 - Giao diện mục Labels 16](#_Toc146741798)

[Hình 22 - Giao diện mục Milestones 17](#_Toc146741799)

[Hình 23 - Giao diện tab Pull Requests 17](#_Toc146741800)

[Hình 24 - Giao diện tab Actions 18](#_Toc146741801)

[Hình 25 - Một số actions được cung cấp sẵn bởi Github 19](#_Toc146741802)

[Hình 26 - Giao diện Tab Projects 19](#_Toc146741803)

[Hình 27 - Liên kết với Project 20](file:///D:\Hoc%20Tap\Dai%20Hoc\UIT\HK3\Kỹ%20năng%20nghề%20nghiệp\GitLearning\document_full.docx#_Toc146741804)

[Hình 28 - Giao diện tab Wiki 20](#_Toc146741805)

[Hình 29 - Tạo trang wiki 20](#_Toc146741806)

[Hình 30 - Giao diện tab Security 21](#_Toc146741807)

[Hình 31 - Giao diện của mục Security Advisories trong Security 21](#_Toc146741808)

[Hình 32 - Giao diện chính tab Insights 22](#_Toc146741809)

[Hình 33 - Các mục của tab insights 22](file:///D:\Hoc%20Tap\Dai%20Hoc\UIT\HK3\Kỹ%20năng%20nghề%20nghiệp\GitLearning\document_full.docx#_Toc146741810)

[Hình 34 - Giao diện tab Settings 24](#_Toc146741811)

[Hình 35 - Danger Zone trong setting của repository 25](#_Toc146741812)

[Hình 36 - Delete một repository 25](#_Toc146741813)

[Hình 37 - Branch của repository 32](#_Toc146741814)

[Hình 38 - Minh hoạ quá trình làm việc với GitHub 34](#_Toc146741815)

[Hình 39 - Quá trình xác thực hai yếu tố (2FA) 35](#_Toc146741816)

[Hình 40 - Settings mục Collaborators 35](#_Toc146741817)

[Hình 41 - GitHub Backup 36](#_Toc146741818)

# **CHƯƠNG 1: TỔNG QUAN VỀ GIT VÀ GITHUB**

## **GIT**

### **Khái niệm :**

* **Git** là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) có mã nguồn mở và là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay.
* **Git** cung cấp cho lập trình viên một kho lưu trữ riêng (**repository**) để lưu lại các lịch sử thay đổi.

### **Cách thức hoạt động :**

* Trong khi các hệ thống quản lý phiên bản khác lưu trữ thông tin các thay đổi trên file theo thời gian dưới dạng danh sách (Subversion, Perforce, Bazaar, …) thì Git lưu trữ những thông tin ấy dưới dạng snapshot (ảnh).

A diagram of a process

Description automatically generated

Hình 1 Mô tả cách hoạt động của hệ thống

* Khi mỗi lần lập trình viên thực hiện những thay đổi hay muốn lưu lại, Git sẽ chụp lại các thông tin của file tại thời điểm đó và tạo ra một snapshot, và tạo ra một tham chiếu tới snapshot đó.
* Bên cạnh đó, nếu tập tin không có sự thay đổi thì Git không lưu trữ lại tập tin đấy mà tạo một đường dẫn tới tập tin đã tồn tại trước đó. Qua đó giúp giảm thời gian xử lý các thông tin cần lưu trữ, và giảm lãng phí bộ nhớ do trùng lặp.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 2 - Mô tả cách hoạt động của Git

**Tính phân tán (Distributed) của Git:**

* Sở dĩ nói Git có tính phân tán, vì mọi lập trình viên đều có thể tham gia vào việc xây dựng, phát triển và đóng góp mã nguồn vào một kho lưu trữ (repository) và tạo ra một kho lưu trữ công khai khác để những người bên ngoài có thể sử dụng nó cho công việc của họ và cũng qua đó đóng góp thêm vào đó.

### **Lợi ích:**

Git mang lại rất nhiều lợi ích cho các lập trình viên:

* **Khả năng phân nhánh:** Git cung cấp cho mỗi lập trình viên các branch (nhánh), cho lập trình viên một môi trường độc lập, riêng biệt cho việc thay đổi mã nguồn. Qua đó, lập trình viên có thể tự do thay đổi, cập nhật, phát triển mã nguồn và kiểm thử trước khi đưa vào nhánh chính. Tạo điều kiện thuận lợi cho việc code theo nhóm.
* **Thuận tiện cho việc phát triển:** Git lưu lại các phiên bản khác nhau của mã nguồn, qua đó dễ dàng cho việc phát triển, chỉnh sửa, cập nhật.
* **Cho phép làm việc offline:** Git cho phép chúng ta làm việc offline một khoảng thời gian, ta chỉ cần internet cho các việc như họp nhóm, hay các thao tác commit code lên kho lưu trữ, trong khi đó các phần mềm quản lý phiên bản khác yêu cầu người dùng phải liên tục kết nối với máy chủ.
* **Git miễn phí:** Một phần Git được cộng động ưa chuộng sử dụng là vì Git miễn phí. Mọi người đều có thể sử dụng các chức năng của Git mà không cần phải trả bất kì chi phí nào.

# **GITHUB**

### **Github là gì?**

* Github được tạo ra bởi ba lập trình viên người Mỹ Tom Preston-Werner, Chris Wanstrath, và PJ Hyett. Dự án được bắt đầu vào ngày 19 tháng 10 năm 2007 và được phát hành chính thức vào tháng 4 năm 2008. Cho tới hiện này thì Github được đánh giá là một trong các dịch vụ lưu trữ mã nguồn lớn nhất hiện nay.



Hình 3 - Github là một trong các nền tảng lưu trữ mã nguồn lớn nhất hiện nay

* Github là một dịch vụ cung cấp kho lưu trữ mã nguồn Git và có đầy đủ các tính năng của Git. Bên cạnh đó, Github còn có thêm một số tính năng hỗ trợ người dùng quản lý, và các tính năng khác như social để các lập trình viên có thể tương tác với nhau.
* Github hiện nay có 3 phiên bản: Free, Team, Enterprise. Với các phiên bản Team, Enterprise thường được các doanh nghiệp sử dụng hơn vì có các tính năng nhằm tăng tính bảo mật dự án, và phân quyền trong dự án. Phần lớn người dùng bình thường chúng ta sử dụng Github bản Free.

Ảnh có chứa văn bản, ảnh chụp màn hình, thiết kế

Mô tả được tạo tự động

Hình 4 - Các phiên bản Git hiện nay

1. **Lợi ích của Github:**

* **Quản lý source code dễ dàng :** Khi tạo một repository, thì mọi source code của lập trình viên sẽ được lưu trên Github. Từ đó người dùng có thể theo dõi quá trình mình đã làm thông qua các lần commit.
* **Theo dõi (Tracking) thay đổi qua các phiên bản:** Thường thì một dự án sẽ có nhiều thành viên tham gia code nên lập trình viên rất khó để theo dõi sự thay đổi của source code, nhưng Github sẽ lưu lại những thay đổi lên repository nên người dùng có thể dễ dàng theo dõi những thay đổi diễn ra với source code.
* **Làm đẹp CV :** Trong mắt các nhà tuyển dụng thì việc có sản phẩm, source code luôn là minh chứng tốt nhất cho trình độ của bạn. Vì vậy, nếu có link Github trong CV, nó sẽ làm nổi bật và thu hút các nhà tuyển dụng.
* **Cải thiện khả năng lập trình:** Github tạo ra một môi trường rất thuận tiện cho các lập trình viên có thể phát triển kĩ năng của mình qua các dự án Open source luôn được cập nhật mỗi ngày, cung cấp cho lập trình viên những kinh nghiệm để hoàn thiện khả năng của mình.

### **Giao diện cơ bản**

**Đăng nhập và đăng ký**

* Để có thể sử dụng Github thì người dùng phải có tài khoản Github, nếu chưa có người dùng có thể đăng ký như hình dưới.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động

Hình 5 - Giao diện đăng ký GitHub

* Sau khi có tài khoản ta tiến hành đăng nhập:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 6 - Giao diện đăng nhập của GitHub

* Sau khi đăng nhập, ta sẽ có một giao diện trực quan, cơ bản như sau:

Ảnh có chứa văn bản, phần mềm, số, Biểu tượng máy tính

Mô tả được tạo tự động

Hình 7 - Giao diện giành cho người dùng GitHub

**Tạo một repository**

* Để tạo một repository mới, người dùng nhấn vào dấu cộng phía trên bên phải:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 8 - Phần tạo mới một repository

* Sau đó người dùng đặt tên cho repo và các miêu tả cần thiết. Github đưa ra cho người dùng 2 lựa chọn : Private, Public. Trong đó public là chế độ mặc định, cho phép bất kì ai cũng có thể xem được repo của mình. Bên cạnh đó, nếu người dùng muốn sự riêng tư, hay không muốn công khai, ta có thể chọn Private.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Hình 9 - Giao diện phần tạo repository

* Sau khi hoàn thành các bước trên người dùng nhấn Create repository để hoành thành. Dưới đây là giao diện ta có được sau khi hoàn thành việc tạo mới một repository.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 10 - Một số cách Setup repository mà GitHub cung cấp

#### Tab < > Code

* Tab < > Code là giao diện chính của một repository, đây cũng là nơi những tệp tin/mã nguồn được đưa thành viên tham gia đưa lên GitHub lưu trữ, xem xét và kiểm tra.

A screenshot of a computer

Description automatically generated

Hình 11 - Giao diện tab Code

* Ở tab này, người dùng có thể dùng các chức năng mà Github có sẵn để dễ dàng truy cập, sửa đổi các file đã được đưa lên thông qua mục “Go to file”. Ngoài ra ta còn có thể các files một cách trực tiếp vào repository thông qua mục “Add file”.
* A green rectangle with white text

  Description automatically generatedA close up of a sign

  Description automatically generatedNgoài ra, ở Tab < > Code còn cung cấp thêm một mục là “Code”, dùng để clone một repository, hoặc sử dụng các tài nguyên trong repository đó ở những nơi khác như “GitHub Desktop, Visual Studio,…).

A white rectangle with black text

Description automatically generated

Hình 12 - Mục Code

Hình 14 - Mục Go to file

Hình 13 - Mục Add file

* Ta cũng có thể truy cập, tạo mới các branches một cách dễ dàng trên GitHub.

A screenshot of a computer

Description automatically generated

Hình 15 - Phần truy cập và tạo mới branch

* Bên cạnh đó, người dùng có thể thêm Tag, một chức năng cho phép người dùng xác định rõ ràng các phiên bản của mã nguồn dự án.
* Ngoài ra, Github còn có chức năng Releases. Releases dựa trên chức năng Tag, nó đánh dấu một điểm cụ thể trong lịch sử repository. Chức năng này dùng để chia sẻ đóng gói ứng dụng, ghi chú và các liên kết liên quan cho những người khác sử dụng

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 16 - Giao diện của phần chức năng Releases & Tag

#### Tab ☉ Issues

A screenshot of a computer

Description automatically generated

Hình 17 - Giao diện tab Issues

* Tab Issues được dùng để theo dõi các tác vụ cần làm, lỗi, yêu cầu tính năng,… Khi các Issues được tạo mới thông qua “New issuse”, chúng sẽ xuất hiện ở giao diện chính của tab này thành một danh sách. Danh sách này có thể được tìm kiếm và lọc qua dễ dàng thông qua các tính năng bổ sung như “Labels”, “Milestone”.



Hình 18 - Nút tạo issue

A screenshot of a computer

Description automatically generated

Hình 19 – Phần filter của tab Issue

A black arrow with a white background

Description automatically generated with medium confidence

Hình 20 - Mục labels và Milestone

A screenshot of a computer

Description automatically generated

Hình 21 - Giao diện mục Labels

A screenshot of a computer

Description automatically generated

A green sign with white text

Description automatically generatedHình 22 - Giao diện mục Milestones

Hình 22 - Tạo một milestone mới

* Labels có thể hiểu một cách đơn giản là “dán nhãn”, thông qua việc dán nhãn các Issue, người dùng sẽ dễ dàng kiểm soát và quản lý các Issues.
* Ở mục Milestones, ta có thể đặt ra các “cột mốc” để quản lý các Issues, Pull Requests (Sẽ nói đến trong phần sau) dành cho một lần phát hành hoặc một dự án cụ thể nào đó.

#### Tab ⭿ Pull Request

A screenshot of a computer

Description automatically generated

Hình 23 - Giao diện tab Pull Requests

* Khi những thành viên làm việc trên các branches khác nhau, họ có thể gửi đi một Pull Requests để yêu cầu sát nhập, những người khác có thể xem xét, đưa ra ý kiến và đồng ý chấp nhận sát nhập những thay đổi đó vào nhánh chính.
* Đây là một tính năng tuyệt vời để việc cộng tác giữa các thành viên với nhau và đảm bảo rằng dự án được kiểm tra và thử nghiệm kỹ lưỡng trước khi được đưa vào sản phẩm.
* Pull Requests cũng sẽ được lưu trữ thành các danh sách có thể tìm kiếm và lọc được tương tự như Issues, ngoài ra chúng cũng có các labels, milestones,…

#### Tab ⏵Actions

A screenshot of a computer

Description automatically generated

Hình 24 - Giao diện tab Actions

* GitHub Actions là một nền tảng tích hợp và phân phối liên tục (CI/CD) cho phép người dùng có thể tự động hoá quá trình xây dựng, thử nghiệm và triển khai các dòng lệnh. Người dùng có thể tự định nghĩa các workflows tự động hoá trong quá trình phát triển project.
* Mỗi workflow là một tập hợp các hành động (actions). Các hoạt động này là các lệnh cụ thể như build, test, deploy code. Người dùng có thể sử dụng các actions được cung cấp sẵn bởi GitHub để tạo ra các workflow phù hợp với dự án của mình. Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

  Mô tả được tạo tự động

Hình 25 - Một số actions được cung cấp sẵn bởi Github

#### Tab 🗏Projects

A screenshot of a computer

Description automatically generated

Hình 26 - Giao diện Tab Projects

* Projects là một bảng tính (spreadsheets), bảng tác vụ (task-board), lộ trình (road map) linh hoạt để tích hợp với các Issues và Pull Requests trên GitHub để giúp người dùng lập ra các kế hoạch và theo dõi công việc hiệu quả hơn.
* Một số tính năng của tab Projects:
* Tạo và tùy chỉnh nhiều chế độ xem bằng cách lọc, sắp xếp, nhóm Issues và Pull Requests của bạn.
* Trực quan hóa công việc với các biểu đồ có thể cấu hình.
* Thêm các trường tùy chỉnh để theo dõi dữ liệu cụ thể cho nhóm.
* Tích hợp với Issues và Pull requests của người dùng trên GitHub.

Hình 27 - Liên kết với Project

#### Tab 🕮Wiki

A screenshot of a computer

Description automatically generated

Hình 28 - Giao diện tab Wiki

* Đây là nơi mà người dùng có thể chia sẻ các thông tin, nội dung về dự án của họ. Ví dụ như cách sử dụng một tính năng nào đó, cách mà dự án được thiết kế nên,…
* Một file README cũng có thể xem như là một cái wiki cỡ nhỏ, file này sẽ cho ta biết những thông tin sơ bộ về cách hoạt động của dự án người dùng đang thực hiện

Hình 29 - Tạo trang wiki

#### Tab 🞚Security

A screenshot of a computer

Description automatically generated

Hình 30 - Giao diện tab Security

* Tab này sẽ quản lý toàn bộ quá trình bảo mật, quy cách sử dụng một repository, bạn có thể tạo ra các chính sách “Security Policy”, Quét qua các mã code “Code scanning” để tìm ra các lỗi sai.
* Security Policy giúp bạn có thể đưa ra các chính sách giúp người dùng biết được làm sao để báo cáo các vấn đề bảo mật liên quan tới repository một cách an toàn.
* Advisories cho phép bạn thảo luận riêng với các cộng tác viên về các vấn đề bảo mật của dự án. Sau khi giải quyết xong các vấn đề ấy, bạn có thể chia sẻ nó lên với các người quan tâm tới repository của bạn.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 31 - Giao diện của mục Security Advisories trong Security

Tab Insight A screenshot of a computer

Description automatically generated

Hình 32 - Giao diện chính tab Insights

A screenshot of a computer

Description automatically generated

Hình 33 - Các mục của tab insights

* Tab Insight là nơi quản lý về các thông số của một repository, xuyên suốt quá trình phát triển dự án, mọi hoạt động của các thành viên đều được ghi nhận và thống kê ở tab này. Đây là một tab khá quan trọng để theo dõi tiến độ công việc, mức độ hoàn thành của một project.
* Tab Insights gồm các mục:
  + Contributors (Người đóng góp), ở mục này, người dùng có thể quản lý xem những người tham gia đã đóng góp những gì, đóng góp bao nhiêu cho main, cho các branches, kể cả các merge commit, và đánh giá mức độ hiệu quả của một người.
  + Community (Cộng đồng), mục này bao gồm việc tạo một cuộc thảo luận (discussion) trong quá trình phát triển dự án, và mỗi luận thảo luận như vậy, Insights sẽ ghi nhận lại và cho biết mức độ sôi nổi, đóng góp của mỗi thành viên trong mỗi cuộc thảo luận như thế.
  + Community Standards (Tiêu chuẩn cộng đồng): Đây là nơi sẽ so sánh dự án này với các tiêu chuẩn cộng đồng được để xuất.
  + Traffic (Giao thông): Thống kê lượt truy cập từ người dùng vào dự án.
  + Commits: Thống kê commits.
  + Code Frequency (Độ thường xuyên của việc code): Thống kê số lần **thêm** và **xoá**.
  + Dependencies graph (Đồ thị các phụ thuộc).
  + Network graph (Đồ thị mạng): Đưa ra đồ thị các commits của người dùng đến với repository này theo thứ tự được push lên gần nhất.
  + Fork:[[1]](#footnote-1) Danh sách những người đã fork repository này.

#### Tab ⚙ Settings

A screenshot of a computer

Description automatically generated

Hình 34 - Giao diện tab Settings

* Đây là nơi chỉnh sửa các thông số, cài đặt của một repository. Cụ thể như thêm những chính sách bảo vệ các branch, hay các cài đặt về bảo mật và phân tích code. Bên cạnh đó, tại đây người dùng có thể thay đổi tên cũng như thêm thành viên tham gia, hay thảo luận về repository của mình trên cộng đồng của Github

#### Xoá một repository

Để xoá repository nào đó, ta vào trang chính của repository đó, và chọn vào nút

. Sau đó người dùng kéo đến mục Danger Zone và chọn Delete this repository .

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 35 - Danger Zone trong setting của repository

* Sau đó người dùng xác nhận các cảnh báo của Github và cuối cùng là nhập lại tên Repository muốn xoá và xác nhận xoá. Như vậy là chúng ta đã xoá thành công repository.

**Lưu ý**: Người dùng có thể khôi phục lại repository trong **90 ngày**.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 36 - Delete một repository

# **CHƯƠNG 2: CÁC TẬP LỆNH GIT CƠ BẢN**

Khi làm việc với Git ta cần nắm rõ các câu lệnh để có thể thao tác một cách chính xác, việc sử dụng thành thạo chúng sẽ là một kĩ năng cần thiết để chúng ta quản lí các dự án sau này. Chương này sẽ giới thiệu cho chúng ta các thao tác và câu lệnh cơ bản để thực hiện được mục đích trên.

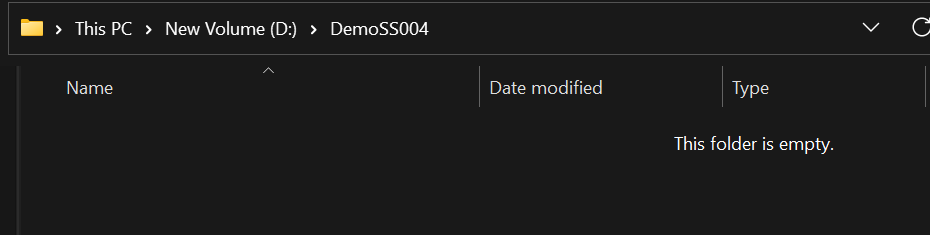
### **Một số thao tác liên quan đến thư mục và file thường dùng.**

* **Tạo một thư mục mới**

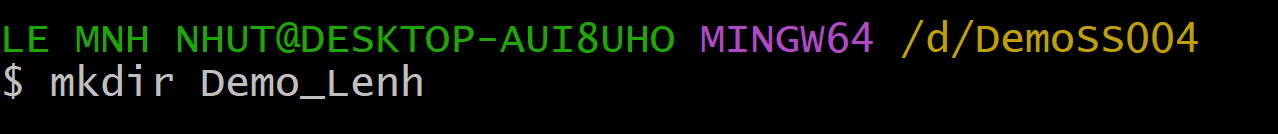
**mkdir** **<Tên thư mục>**

**Ví dụ:**

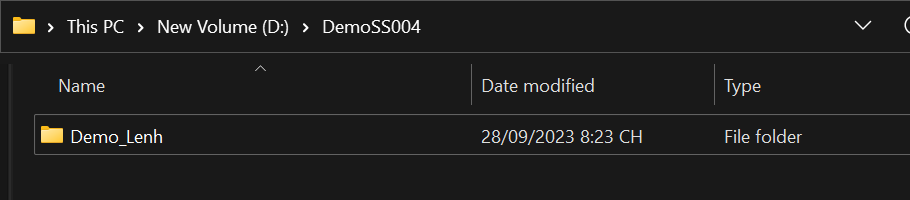
*- Đường dẫn ban đầu không chứa thư mục nào:*



*- Ta dùng lệnh* ***mkdir*** *để tạo một thư mục có tên* ***Demo\_Lenh****:*



*- Ta thấy lúc này thư mục* ***Demo\_Lenh*** *đã được tạo:*

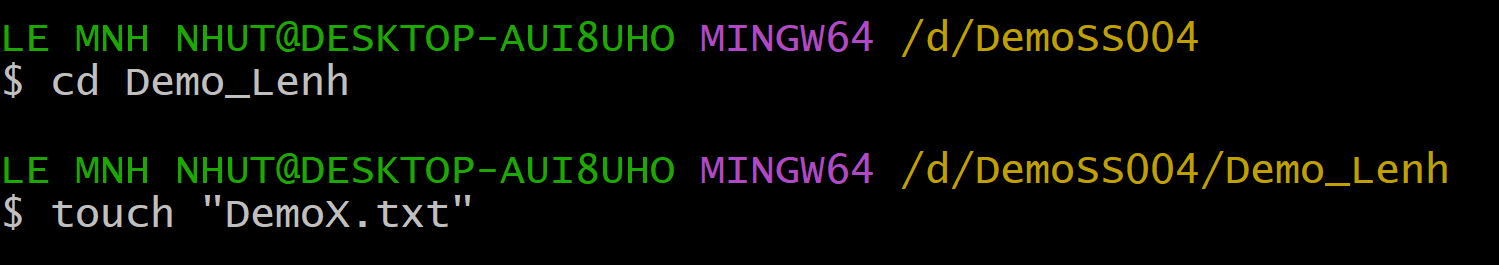


* **Tạo file mới**

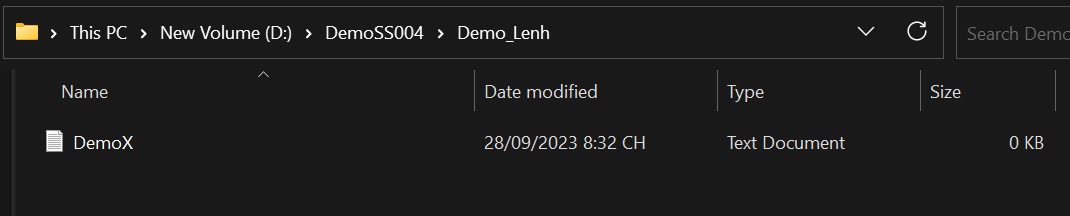
**touch** “**<Tên file>**”

**Ví dụ:**

*- Trong ví dụ ở trên, thư mục* ***Demo\_Lenh*** *vừa được tạo nên chưa chứa bất kì file nào, ta dùng lệnh* ***cd******Demo\_Lenh*** *để chuyển đến thư mục* ***Demo\_Lenh****, rồi dùng lệnh* ***touch*** *để tạo một file có tên* ***DemoX.txt****:*



*- File* ***DemoX.txt*** *đã được thêm vào thư mục* ***Demo\_Lenh****:*



* **Thêm và hiển thị nội dung trong file**

**- Thêm nội dung:**

+ Nếu muốn ghi đè nội dung lên nội dung trước đó, ta dùng:

**echo “Nội dung cần thêm” > <Tên file>**

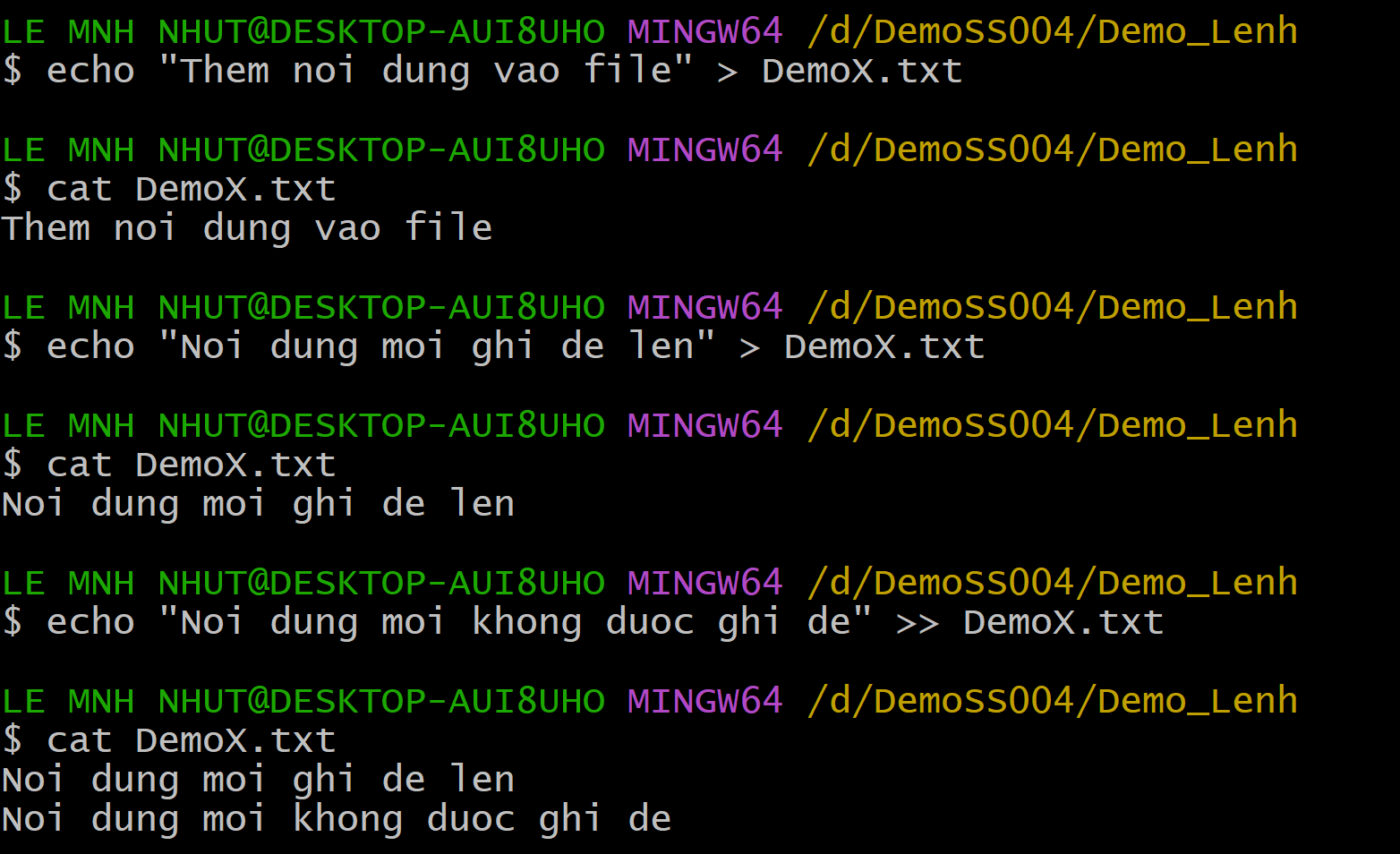
+ Nếu muốn thêm nội dung mà không ghi đè lên nội dung trước, ta dùng:

**echo “Nội dung cần thêm” >> <Tên file>**

**- Để hiển thị nội dung có trong file ta dùng lệnh:**

**cat <Tên file>**

**Ví dụ:**

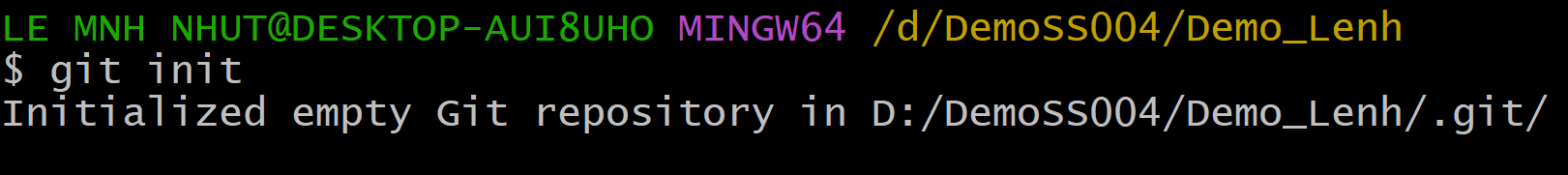


### **Tạo một kho lưu trữ Git mới.(Git repository)**

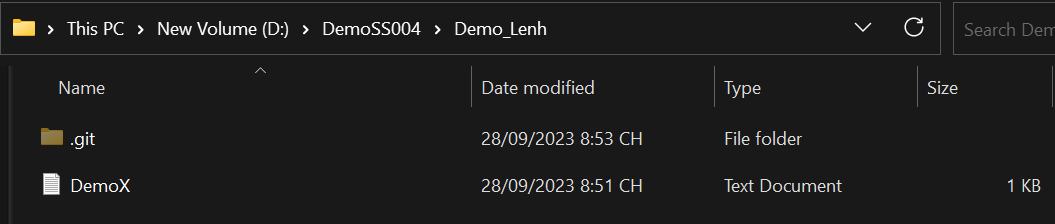
* Để tạo một kho lưu trữ mới, ta có thể sử dụng lệnh **git init**. Lệnh này sẽ tạo một thư mục mới với các tệp và thư mục cần thiết cho một kho lưu trữ git.

**git init**

**Ví dụ:**



*- Sau khi dùng lệnh* ***git init****, ta thấy 1 dòng lệnh được trả về có ý nghĩa báo rằng 1 kho lưu trữ đã được tạo thành công trong thư mục có đường dẫn là* D:/DemoSS004/Demo\_Lenh/.git/.



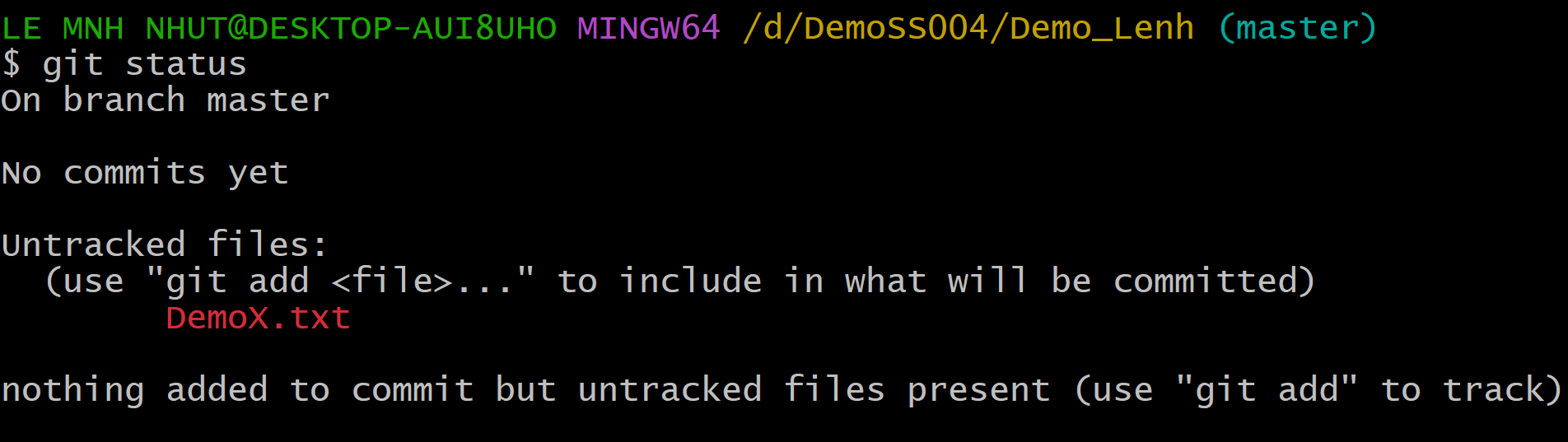
### **Kiểm tra trạng thái và thêm các tệp.**

* Để kiểm tra trạng thái của các tệp, bạn có thể sử dụng lệnh **git status**. Lệnh này sẽ hiển thị danh sách các tệp đã được thêm, thay đổi hoặc chưa được theo dõi.

**git status**

**Ví dụ** :

*- Nhập lệnh* ***git status*** *để kiểm tra trạng thái các file trong thư mục* ***Demo\_Lenh****:*



*- Ta thấy:*

*+* ***On branch master:*** *cho biết ta đang ở trong nhánh master*

*+* ***No commits yet:*** *cho biết ta chưa thực hiện commit nào trong nhánh master*

*+* ***Untracked files:*** *liệt kê danh sách các tệp chưa được theo dõi trong nhánh master(ở ví dụ này là tệp* ***DemoX.txt****)*

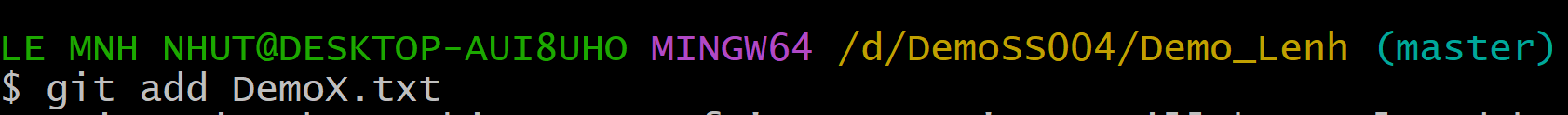
*+* ***nothing added to commit but untracked files present (use “git add” to track):*** *cho biết chúng ta chưa thêm bất kỳ tệp nào vào vùng staging area để commit và có một số tệp chưa được theo dõi.*

* Để thêm các vào kho lưu trữ Git, ta có thể sử dụng lệnh **git add**. Lệnh này sẽ thêm các tệp chưa được theo dõi vào danh sách các các tệp sẽ được kiểm tra trạng thái và xác nhận.

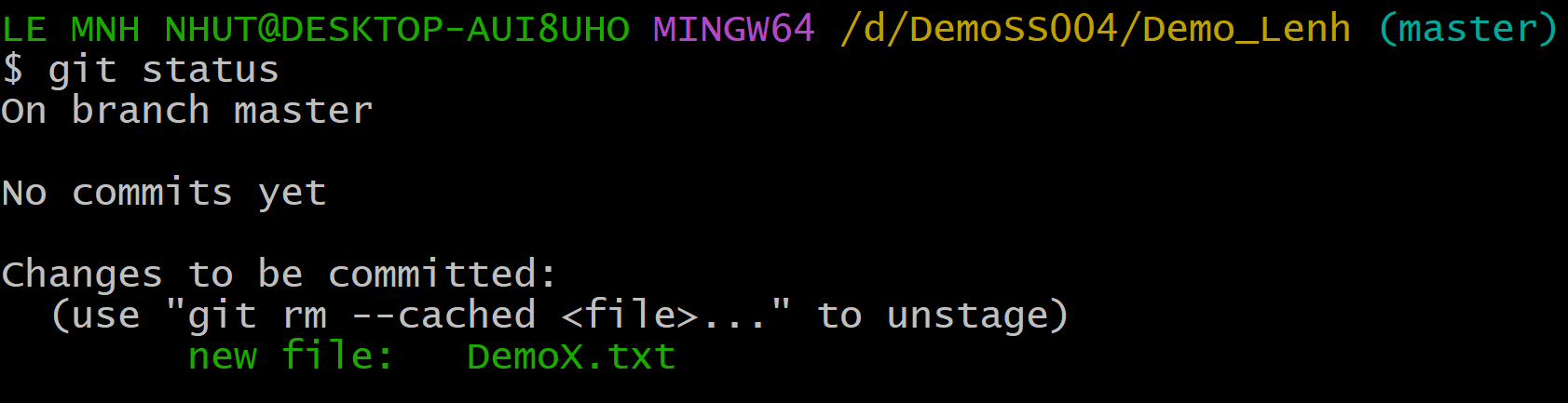
**git add <tệp cần thêm>**

**Ví dụ:**

*- Nhập lệnh* ***git add DemoX.txt:***



*- Dùng lệnh* ***git status*** *để kiểm tra trạng thái, ta thấy file* ***DemoX.txt*** *đã được thêm vào kho lưu trữ:*



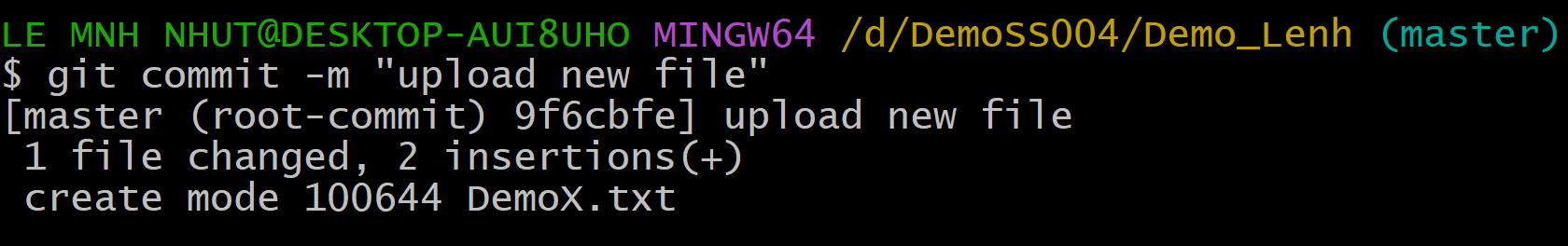
### **Tạo và xác nhận các thay đổi.**

* Để tạo một bản sao lưu của các thay đổi đã được theo dõi, ta có thể sử dụng lệnh **git commit**. Lệnh này sẽ tạo một bản ghi lịch sử của các thay đổi và lưu vào kho lưu trữ Git. Lệnh **git** **commit** thường đi kèm với một thông điệp để mô tả những thay đổi bạn cần lưu.

**git commit -m “My massage”**

**Ví dụ:**

*- Nhập lệnh* ***git commit***

**

*- Ý nghĩa các dòng lệnh trên:*

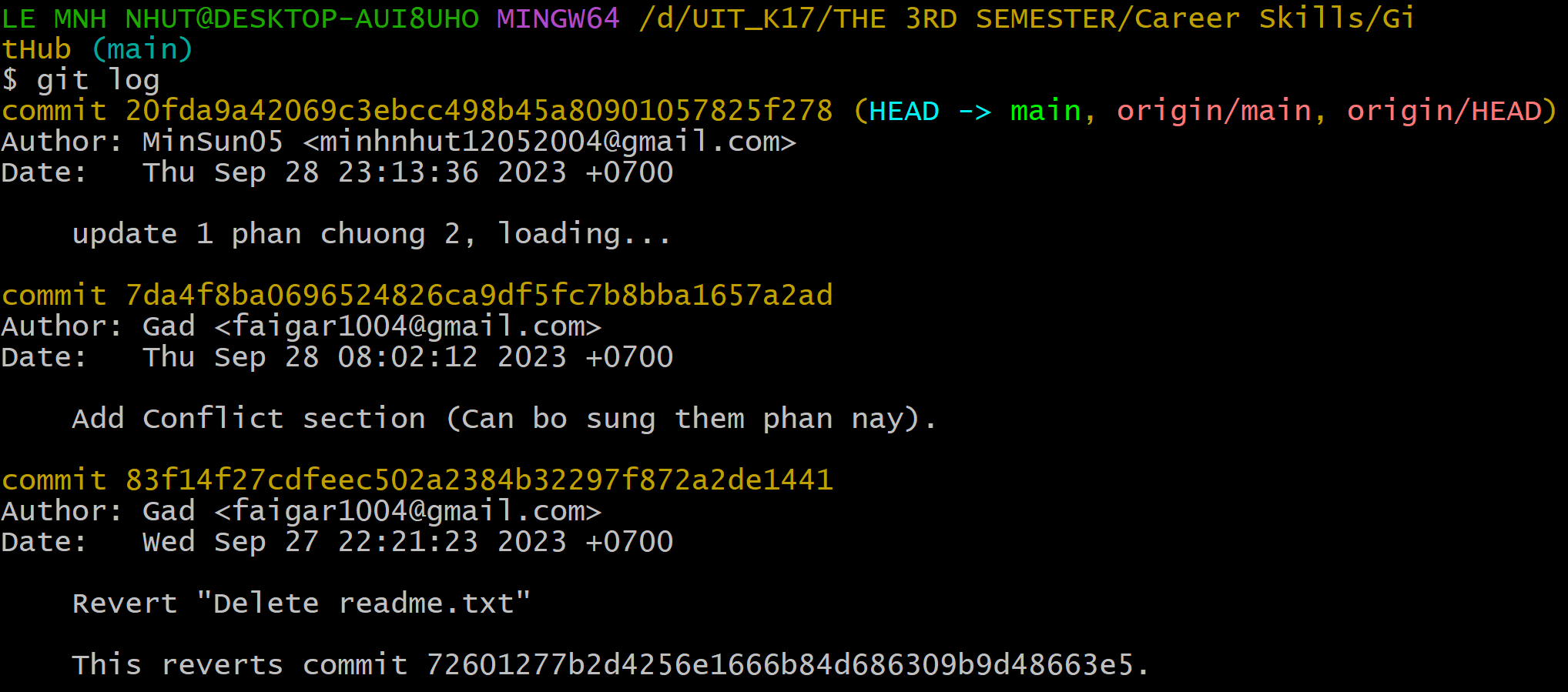
*+* ***Dòng lệnh trả về thứ nhất*** *cho biết đây là commit đầu tiên trong nhánh master và có* ***SHA-1*** *là 9f6cbfe. Commit này có tiêu đề là* ***upload new file.***

*+* ***Dòng lệnh trả về thứ hai*** *cho biết commit này đã thay đổi 1 tệp và thêm 2 dòng vào tệp đó.*

*+* ***Dòng lệnh trả về thứ ba*** *cho biết commit này đã tạo một tệp mới có tên là* ***DemoX.txt*** *với chế độ truy cập là* ***100644*** *(có thể đọc và viết)*

* Sau khi đã commit ta cũng có thể sử dụng lệnh **git log** để xem lịch sử thay đổi của kho lưu trữ Git. Nó sẽ liệt kê tất cả các commit đã được thực hiện, bao gồm tác giả, ngày giờ và thông báo commit.

**Ví dụ:**



*-Sau khi dùng git log ta thấy các thông tin được hiển thị bao gồm:*

*+* ***Author:*** *tên và địa chỉ email của người đã thực hiện commit.*

*+* ***Date:*** *gồm những thông tin như giờ, thứ, ngày tháng, năm đã commit, +0700 là múi giờ UTC +7.*

*+* ***Thông điệp*** *những người đã commit gửi đến.*

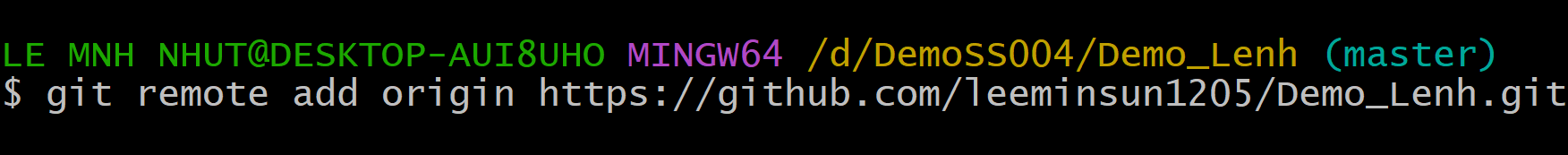
*+ Dòng cuối trong ảnh có ý nghĩa commit có ID trên đang được hoàn tác. Điều này sẽ đưa mã của chúng ta về trạng thái trước khi ta thêm tính năng mới.*

1. **Đẩy các thay đổi lên GitHub.**

* Đầu tiên chúng ta cần thêm một kho lưu trữ từ xa trên GitHub, ta có thể sử dụng lệnh **git remote add origin**. Lệnh này cần ta cung cấp địa chỉ của kho lưu trữ từ xa.

**git remote add origin** **<địa chỉ kho lưu trữ từ xa>**

**Ví dụ:**

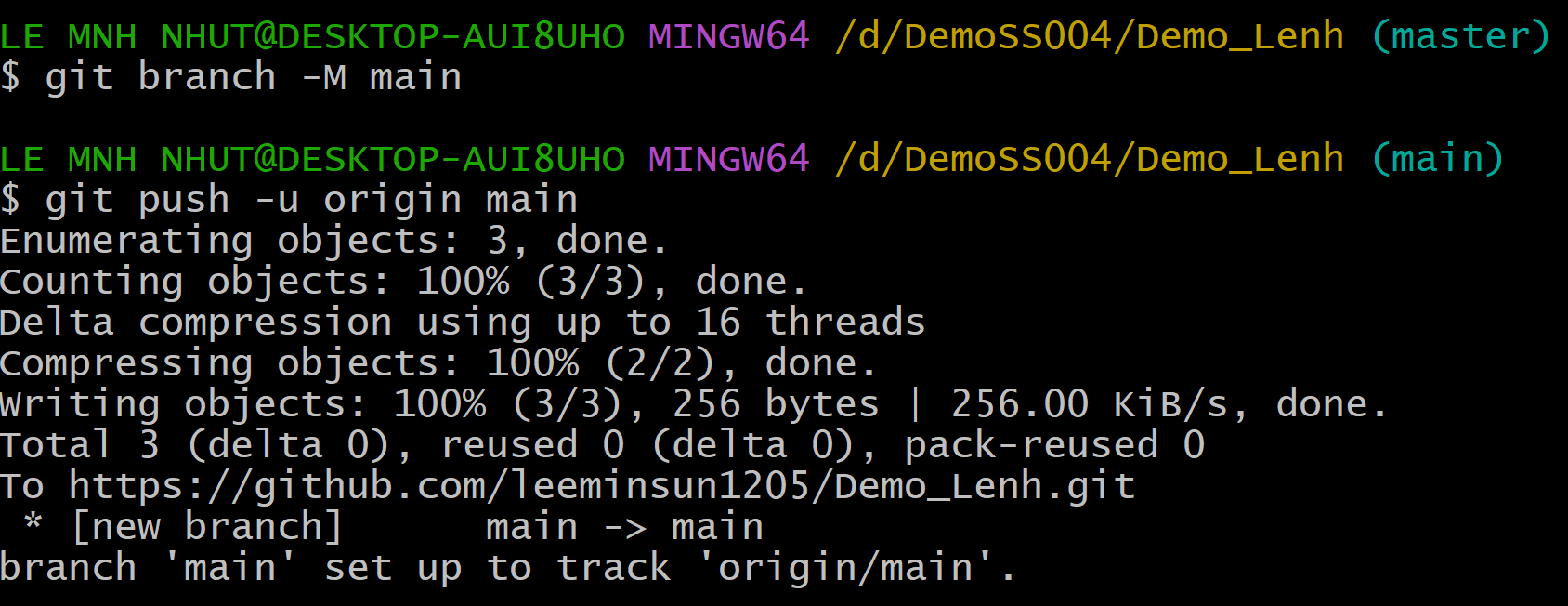


*- Lệnh này sẽ thêm một kho lưu trữ từ xa có tên là* ***origin*** *vào dự án hiện tại. Kho lưu trữ từ xa này sẽ chứa tất các thay đổi trong dự án của chúng ta.*

* Để đẩy các thay đổi lên GitHub, bạn có thể sử dụng lệnh **git push**. Lệnh này sẽ gửi các thay đổi lên kho lưu trữ GitHub của bạn.

**Ví dụ:**

*- Hình bên dưới dùng lệnh* ***git branch -M main*** *để đổi nhánh master thành main (lệnh về* ***branch*** *sẽ được nói rõ hơn ở phần sau), sau đó dùng lệnh* ***git******push*** *để đẩy các thay đổi lênh nhánh main (trong đó* ***origin*** *được xem như tên của kho lưu trữ hiện tại)*



- Sau khi đẩy các thay đổi ta thấy các dòng lệnh trả về với ý nghĩa:

+ **Enumeratin objects**: liệt kê tất cả các đối tượng cần được đẩy lên kho lưu trữ từ xa.

+ **Counting objects**: đếm số đối tượng cần được đẩy lên kho lưu trữ GitHub.

+ **Delta compression using up to 4 threads**: sử dụng nén delta để nén các đối tượng cần được đẩy lên kho lưu trữ từ xa. Nén delta là một kỹ thuật nén hiệu quả cho các đối tượng có liên quan chặt chẽ.

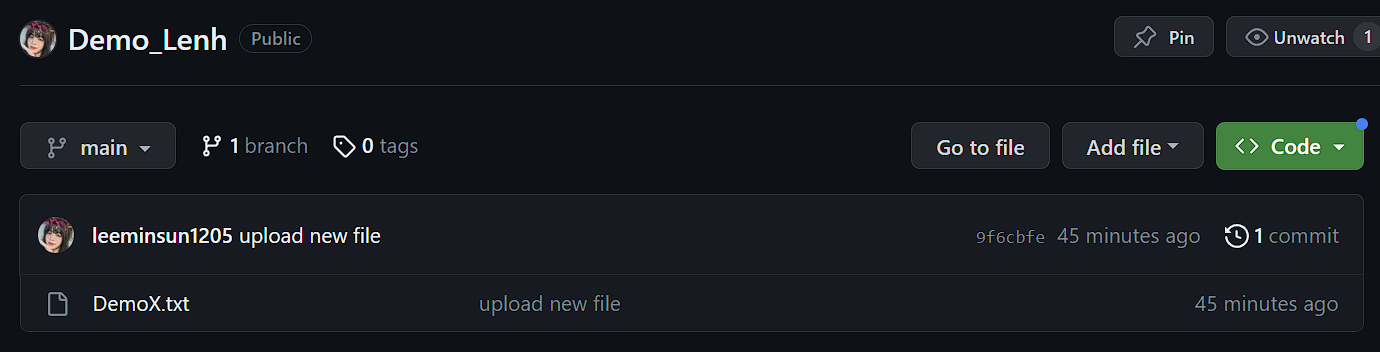
+ **Compressing objects**: nén các đối tượng cần đẩy lên kho lưu trữ từ xa.

+ **Writing objects**: ghi các đối tượng đã được nén lên kho lưu trữ từ xa.

+ **Total 3 (delta 0), reused 0 (delta 0), pack-reused 0**: cho biết 3 đối tượng đã được đẩy lên không có đối tượng nào được nén delta và được sử dụng lại.

+ **2 lệnh cuối** thể hiện tên của đối tượng được đẩy và nhánh đã được cập nhật lênh kho lưu trữ từ xa.

*- Lệnh này đã đẩy tệp* ***DemoX.txt*** *lên nhánh main của kho lưu trữ GitHub:*

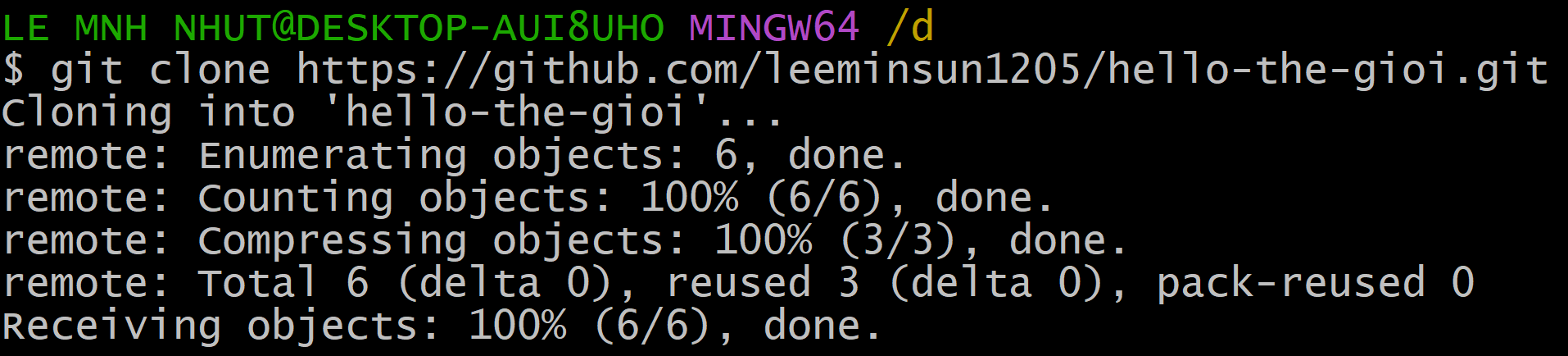


1. **Một số lệnh cơ bản khác:**

* **Git clone**: Lệnh này được sử dụng để sao chép một kho lưu trữ Git từ xa về máy cục bộ.

***git clone <địa chỉ của dự án>***

**Ví dụ**:

**

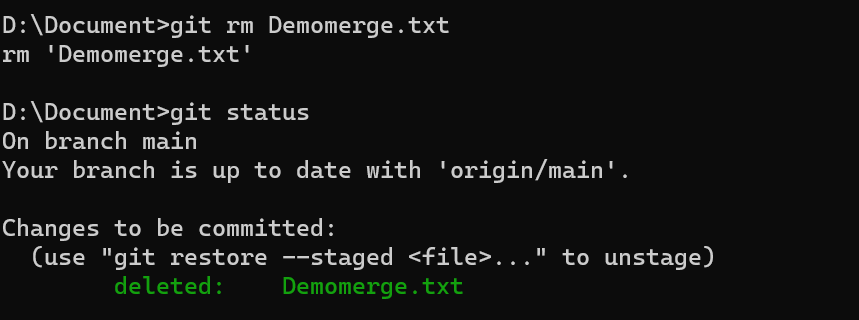
- *Ta thấy thư mục* ***hello-the-gioi*** *của kho lưu trữ từ xa đã được* ***clone*** *về máy*

* **Git rm**: Lệnh này dùng để xóa một file hoặc thư mục khỏi kho lưu trữ Git.

**git rm** **<tên tệp/ thư mục>**

**Ví dụ**:

- *Xóa file* ***Demomerge.txt****:*



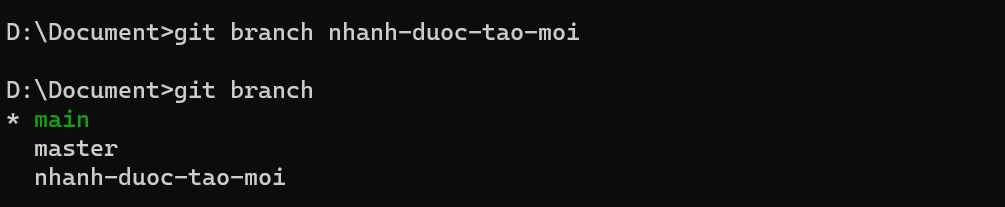
*- Ta thấy trạng thái trả về đã xóa thư mục* ***Demomerge.txt.***

* **Git branch**: Lệnh này được sử dụng để tạo một nhánh mới trong kho lưu trữ Git hoặc kiểm tra xem bạn đang ở nhánh nào. Điều này giúp ta quản lí mã nguồn một cách hiệu quả bằng cách tách biệt các công việc, kiểm thử các thay đổi và tái thạo các trạng thái mã trước đó.

**git branch <tên nhánh mới>**

**Ví dụ**:

- ***git branch nhanh-duoc-tao-moi*** *để tạo một nhánh mới, sau đó dùng lệnh* ***git branch*** *kiểm tra thì thấy nhánh mới tạo đã xuất hiện.*

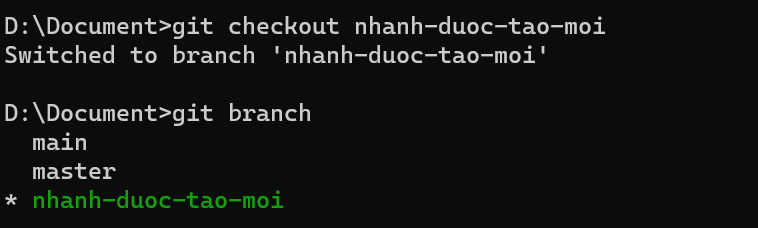


*- Dòng trạng thái bên dưới git branch cho thấy kho lưu trữ có 3 nhánh main, master và nhanh-duoc-tao-moi, trong đó ta đang ở nhánh main.*

* **Git checkout**: Lệnh này được sử dụng để chuyển sang một nhánh khác trong kho lưu trữ Git.

***git checkout <tên nhánh chuyển sang>***

**Ví dụ:**



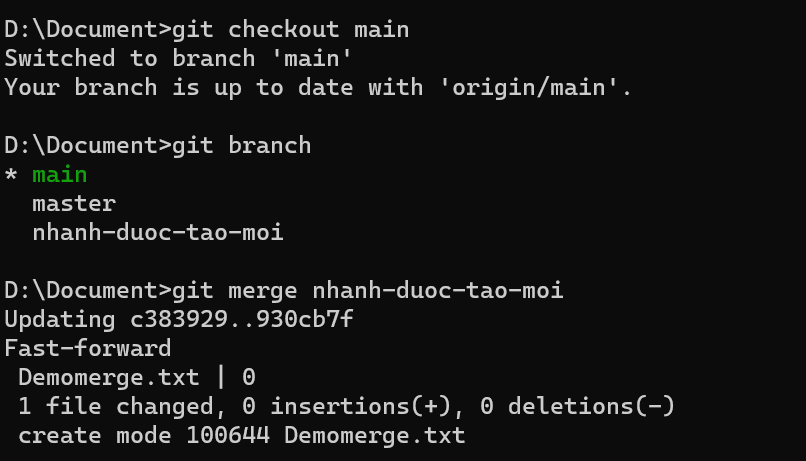
- Trong ví dụ trước ta đang ở nhánh main, sau khi sử dụng **git checkout** ta thấy một dòng lệnh trả về thông báo rằng ta đãchuyển sang **nhanh-duoc- tao-moi**. Kiểm tra lại 1 lần nữa bằng lệnh **git branch** cũng cho ta kết quả là ta đang ở **nhanh-duoc-tao moi**.

* **Git merge**: Lệnh này được dùng để hợp nhất các thay đổi từ một nhánh khác vào nhánh hiện tại. Ví dụ:

+ *Chuyển sang main bằng* ***git checkout****, sau đó thực hiện lệnh*

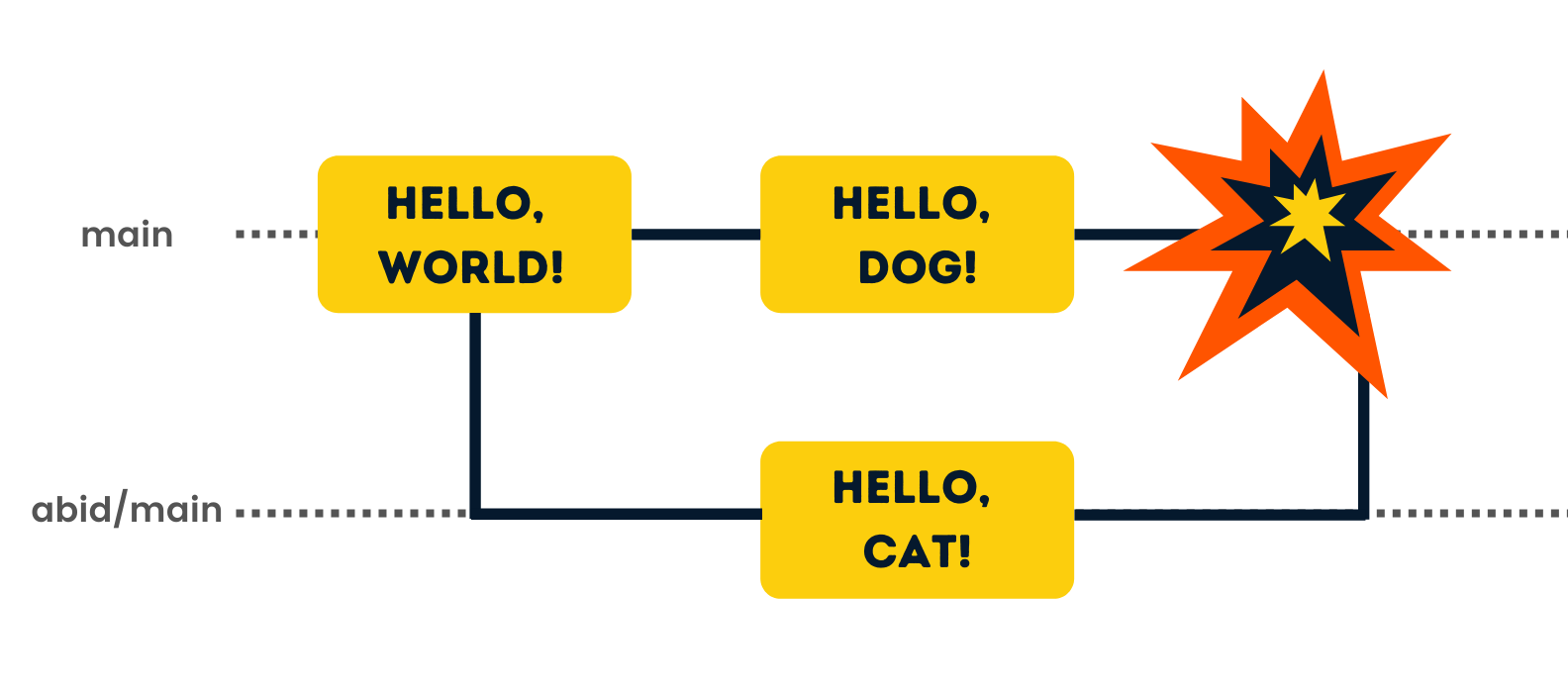
***git merge nhanh-duoc-tao-moi****. Lúc này “nhanh-duoc-tao-moi”*

*đã được gộp vào main:*



### **\*Conflict:**

Tuy nhiên, khi merge hai nhánh trong một repository, sẽ có trường hợp xảy ra xung đột (*conflicts*), và nhiệm vụ của chúng ta là phải giải quyết các xung đột đó (*resolve conflict).*



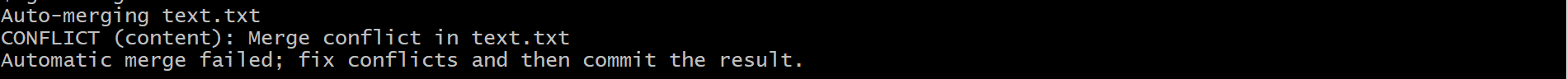
Hình 37 - Conflict trong GitHub

Vậy thì, conflict là gì ? Conflict là một dạng lỗi xảy ra khi merge hai nhánh trong cùng một repository bằng cách dùng GUI của Github hoặc dùng command: **git merge**. Lỗi này diễn ra là do một vấn đề gì đó xảy ra ngăn chặn Git tự động đồng nhất những thay đổi trong các dòng code giữa hai commits. Một số trường hợp dẫn đến conflicts có thể kể đến như:

* Khi nhiều hơn một người thay đổi cùng một dòng trong cùng một file và cố gắng merge thay đổi đó vào trong cùng một nhánh.
* Khi một người đã xoá một file, một người khác thì chỉnh sửa nó, và cùng cố gắng merge thay đổi đó vào trong cùng một nhánh.
* Khi một người xoá một dòng, và một người khác thì chỉnh sửa nó, họ cùng cố gắng merge thay đổi của họ vào trong cùng một nhánh.
* Khi một người cherry-pick một commit trong một nhánh rồi apply nó vào một nhánh khác
* Khi một người dùng rebase một nhánh.

**Một số ví dụ về conflict**:

CONFLICT (content): đây là loại conflict mà khi nhiều người dùng cùng thay đổi một nội dung trong cùng một file.

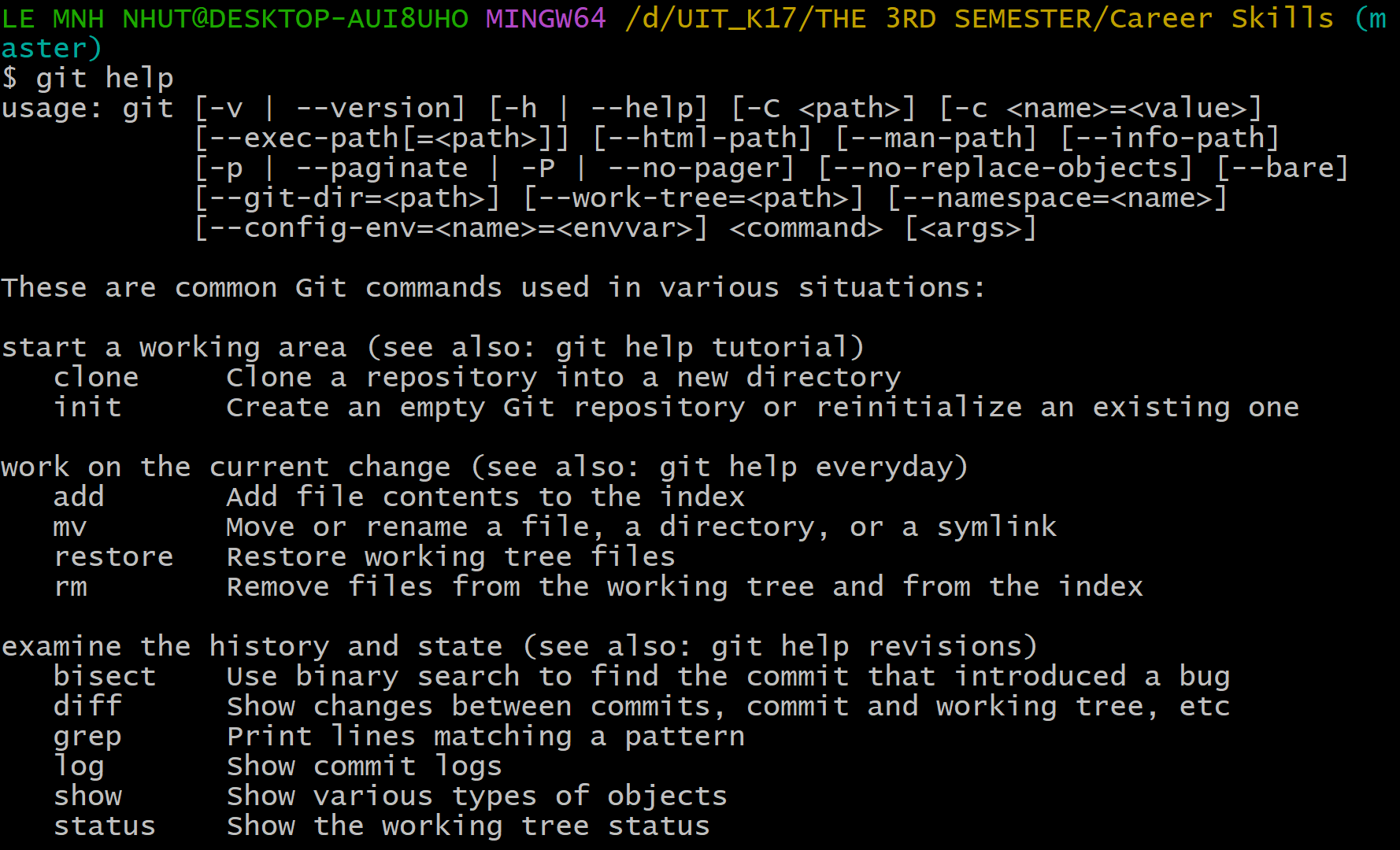


CONFLICT (modify/delete): đây là loại conflict mà khi một người dùng xoá một file, một khác người thì chỉnh sửa file đó.

Ngoài ra còn một số loại conflict có thể kể đến như add/add, …

* **Git help**: Lệnh này được sử dụng để xem trợ giúp về các lệnh Git. Nó sẽ liệt kê tất cả các lệnh Git, bao gồm cú pháp và mô tả.

**git help**

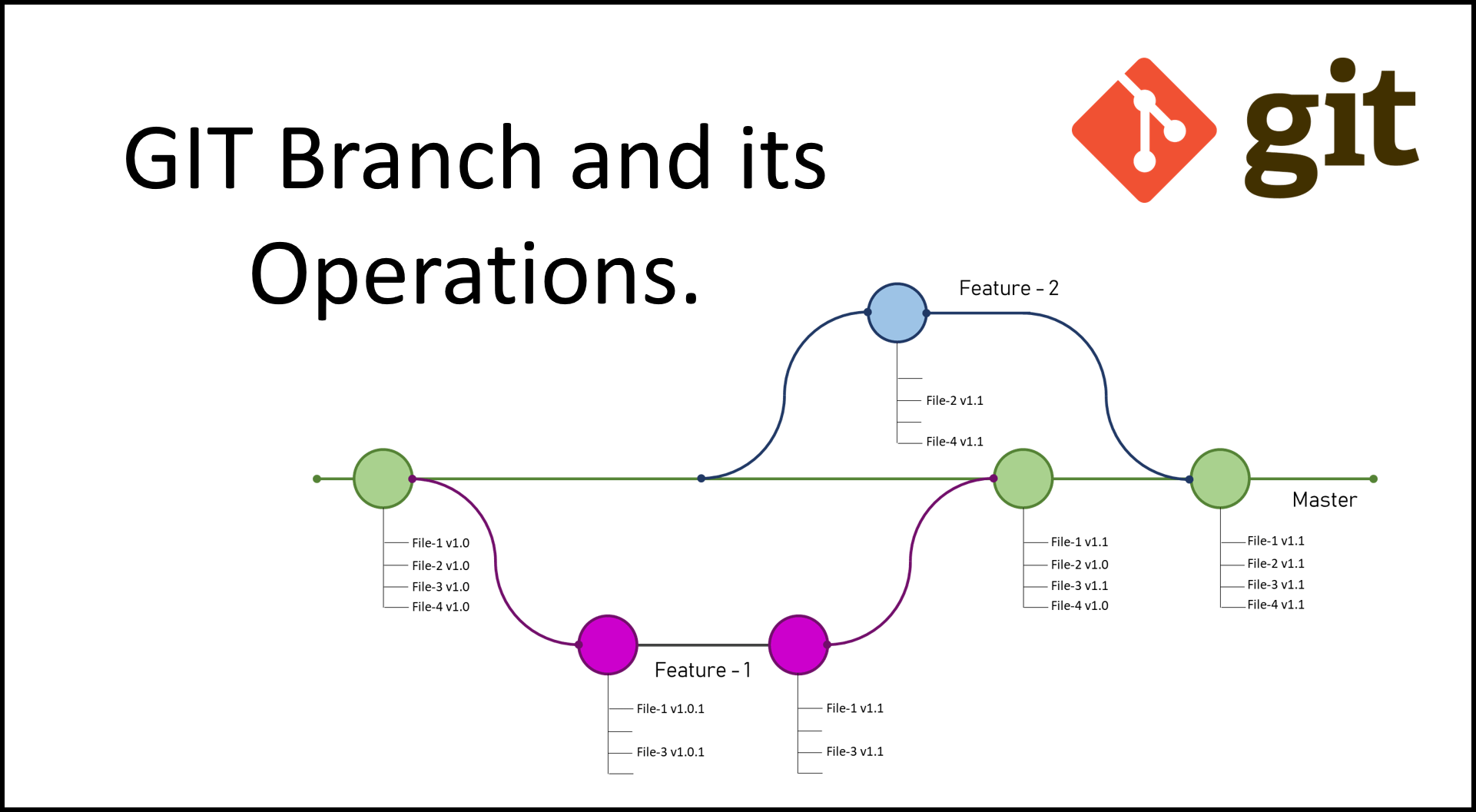


# **CHƯƠNG 3: CÁC LƯU Ý KHI SỬ DỤNG GIT VÀ GITHUB**

## **Đặt tên cho nhánh 1 cách rõ ràng:**

Tại sao việc đặt tên cho nhánh lại quan trọng:

* Đặt tên nhánh một cách rõ ràng giúp tạo sự hiểu biết và sắp xếp công việc trong dự án một cách dễ dàng.
* Tránh xung đột và hiểu nhầm: Tên nhánh rõ ràng giúp tránh xung đột giữa các thành viên và hiểu rõ nhiệm vụ của mỗi nhánh.



Hình 38 - Branch của repository

Gợi ý cách đặt tên nhóm: Sử dụng chuẩn đặt tên như “feature/ten-tinh-nang” hoặc “bugfix/ten-loi,… để dễ dàng quản lý và theo dõi chức năng của các nhánh.

## **Thường xuyên commit và đặt message sáng tạo:**

Quan trọng của việc thường xuyên commit để theo dõi sự thay đổi trong dự án:

* Duy trì lịch sử công việc: Thường xuyên commit giúp duy trì một lịch sử công việc chi tiết và dễ thay đổi trong dự án.
* Quản lý thay đổi hiệu quả: Các commit nhỏ giúp quản lý và kiểm tra thay đổi một cách nhanh chóng, giúp các thành viên nắm rõ các thay đổi của dự án.

Lưu ý về việc viết message commit có ý nghĩa và mô tả đầy đủ:

* Message commit có ý nghĩa: Mỗi message commit nên mô tả rõ ràng về thay đổi được thực hiện để các thành viên khác có thể hiểu.

A black background with white text

Description automatically generated

* Mô tả đầy đủ và ngắn gọn: Sử dụng mô tả đầy đủ nhưng ngắn gọn nhất có thể để tránh thông tin lặp lại và làm rối.

## **Tránh commit lớn và không liên quan:**

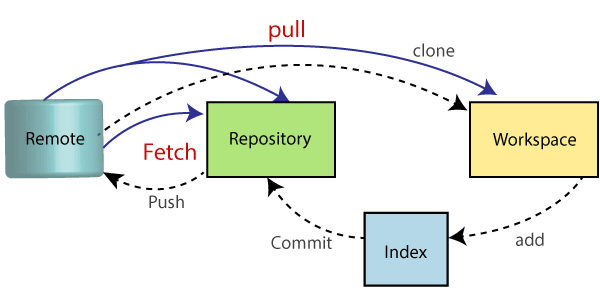
* Khuyến nghị tách công việc thành các commit nhỏ hơn, mỗi commit chỉ thực hiện một nhiệm vụ cụ thể: Các commit nhỏ hơn dễ dàng để theo dõi, xem xét và xác minh.

A screen shot of a computer

Description automatically generated

* Tránh commit những thay đổi không liên quan đến nhiệm vụ hiện tại: Mục đích để không rối loạn tiến trình làm việc.
* Kiểm tra trước khi commit: Trước khi commit, xác minh rằng thay đổi thực sự liên quan đến nhiệm vụ đang thực hiện.

## **Thường xuyên “Pull anh Push”:**



Hình 39 - Minh hoạ quá trình làm việc với GitHub

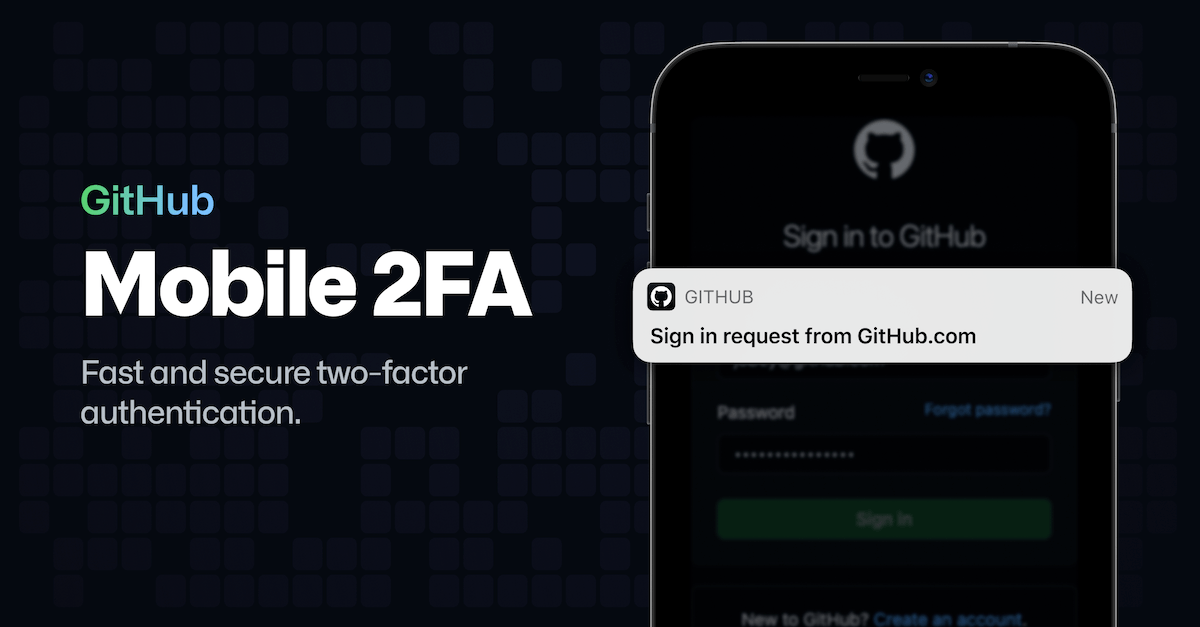
* Tầm quan trọng của việc thường xuyên pull:
* Đồng bộ hóa với nhánh chính: Thường xuyên pull giúp đồng bộ hóa với nhánh chính để tránh xung đột.
* Duy trì mã nguồn ổn định: Đảm bảo mã nguồn luôn ổn định và sẵn sàng cho việc làm tiếp theo.

Hướng dẫn cách thực hiện push an toàn:

## **Bảo mật và quản lý truy cập:**

Bảo mật là một phần quan trọng trong việc quản lý kho lưu trữ Github để ngăn chặn truy cập trái phép và bảo vệ dữ liệu quan trọng. Dưới đây là một số lưu ý quan trọng:

* Cài đặt tính năng xác thực hai yếu tốt (2FA): Sử dụng xác thực hai yếu tố để bảo vệ tài khoảng Github khỏi việc truy cập trái phép.



Hình 40 - Quá trình xác thực hai yếu tố (2FA)

* Quản lý token và mật khẩu một cách an toàn: Không chia sẻ token hoặc mật khẩu với người khác. Lưu chúng 1 cách an toàn.

Hướng dẫn cách thêm collaborators và quản lý quyền truy cập:

* Thêm collaborators: Để thêm thành viên vào dự án, hãy sử dụng tính năng “Settings” trên trang kho lưu trữ Github và chọn “Collaborators”.

A screenshot of a computer

Description automatically generated

Hình 41 - Settings mục Collaborators

* Cấp quyền truy cập: Khi thêm collaborators, bạn có thể cấp quyền cho họ dựa trên từng vai trò, chẳng hạn như quyền đọc (Read), quyền ghi (Write),… Hãy cân nhắc cẩn thận khi cấp quyền để đảm bảo tính bảo mật.

## **Backup và khôi phục dự án:**

A screenshot of a phone

Description automatically generated

Hình 42 - GitHub Backup

**Lưu ý về việc sao lưu (backup) dự án định kỳ:**

* Sao lưu định kỳ: Để đảm bảo tính an toàn của dự án, hãy thực hiện việc sao lưu định kỳ bằng cách sử dụng các công cụ sao lưu tự động hoặc lệnh Git. Sự đều đặn trong việc sao lưu giúp ngăn chặn mất dữ liệu quan trọng.
* Lưu trũ sao lưu an toàn: Lưu trữ sao lưu (backup) của dự án ở một nơi an toàn, có thể truy cập lại dễ dàng khi cần. Điều này đảm bảo rằng bạn có thể khôi phục dự án trong trường hợp có sự cố.

## **Lưu ý về vấn đề khôi phục (restore):**

* Xác định nhu cầu khôi phục: Trước hết, xác định lý do bạn cần phải khôi phục dự án. Có thể là do dữ liệu bị mất, lỗi nghiê trọng, hoặc cần quay lại một phiên bản cụ thể của dự án.
* Tính bản sao lưu phù hợp: Tìm và xác định bản sao lưu hoặc phiên bản cụ thể à bạn muốn phục. Điều này có thể là một commit cụ thể hoặc một phiên bản đã được sao lưu.
* Sử dụng lệnh Git reset hoặc checkout
* Khôi phục tệp và dữ liệu đã mất
* Kiểm tra và kiểm tra lại
* Commit và đồng bộ hóa lại
* Xem xét và học từ sự cố

**Lưu** **ý**:

Quá trình khôi phục có thể khác nhau tùy thuộc vào tình huống cụ thể và sự cố đã xảy ra. Việc duyệt qua lịch sử commit và phiên bản sao lưu là quan trọng để đảm bảo rằng bạn chọn được phiên bản phù hợp để khôi phục.

# **CHƯƠNG 4: TỔNG KẾT**

Trong thời đại mà công nghệ thông tin đang phát triển với mức độ chóng mặt như hiện nay, việc quản lý các mã nguồn, phiên bản và hợp tác phát triển các phần mềm, ứng dụng trở nên ngày càng quan trọng. Git và GitHub là hai công cụ mã nguồn mở phổ biến rất hiệu quả được sử dụng rộng rãi để giải quyết các vấn đề nói trên. Việc áp dụng Git và GitHub vào công việc là cần thiết đối với các dự án lớn nhỏ, đặc biệt là với các dự án có nhiều thành viên tham gia xây dựng.

Qua quá trình nghiên cứu và thực hành, nhóm chúng tôi đã có những hiểu biết nhất định về Git, GitHub. Trong phần tổng kết, chúng tôi sẽ tổng kết lại những nội dung chính của bài báo cáo và đưa ra một số đánh giá, nhận xét.

## **Kết quả đạt được.**

* Tìm hiểu về Git, GitHub.
* Cài đặt và sử dụng thử các tính năng (**Git Bash, Git GUI, thao tác trên CMD**).
* Thao tác các lệnh cơ bản để làm việc trong Repository (**init, clone, push, commit, pull, …**)
* Thao tác các lệnh cơ bản để làm việc ở các nhánh khác nhau trong Repository (**branch, merge, …**).
* Tìm hiểu, nhận biết và xử lý các loại conflict xảy ra khi merge hai nhánh.
* Rút ra được một số lưu ý, kinh nghiệm khi sử dụng Git và GitHub.

## **Các phần chưa đạt được.**

* Thao tác các lệnh nâng cao khác (**rebase, chery-pick, diff, stash …**).
* Làm rõ vai trò quyền hạn của từng người trong quá trình thực hành quản lý.
* Khai thác các tính năng khác của GitHub (**Workflow,** **Hooks**…)

## **Hướng phát triển**

* Tìm hiểu và học cách sử dụng các lệnh nâng cao cũng như các tính năng khác của GitHub và GitHub.
* Ứng dụng GitHub vào quản lý mã nguồn, xây dựng phần mềm cho những môn học khác, cũng như các dự án sau này.
* Tìm hiểu những công cụ đi kèm, hỗ trợ khác (**GitLab, SVN,…**) để quá trình phát triển các ứng dụng, phần mềm đạt được hiểu quả cao nhất.
* Áp dụng Git, GitHub vào quá trình thực tập, làm việc thực tế để hiểu rõ hơn về quy trình làm việc cũng như vai trò của từng người tham gia vào một dự án.

# **TÀI LIỆU THAM KHẢO:**

**{Sách tham khảo}**

1. Version Control with Git, Jon Loellger, Prem Kumar Ponuthorai, (2022).
2. Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer, Mariot Tsitoara (2019).

**{Các trang web tham khảo}**

1. Wikipedia, (2005). Git (Phần mềm). [https://vi.wikipedia.org/wiki/Git\_(phần\_mềm)](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m))
2. Wikipedia, (2005). GitHub. <https://vi.wikipedia.org/wiki/GitHub>
3. Xuanthulab, (2020). Git và GitHub. <https://xuanthulab.net/git-va-github/>
4. Youtube, Tôi đi code dạo, (2019). Từ gà tới pro Git và Github trong 20 phút – Tự học Git siêu tốc. <https://www.youtube.com/watch?v=1JuYQgpbrW0>
5. Git SCM, (2005).<https://git-scm.com/docs/gittutorial>
6. Geeks For Geeks, (2019) .What is a GIT Repository? <https://www.geeksforgeeks.org/what-is-a-git-repository/?ref=lbp>
7. GitHub Docs. <https://docs.github.com/en>
8. Dev, Rizèl Scarlett, (2022). How to prevent Merge Conflicts (or at least have less of them). <https://dev.to/github/how-to-prevent-merge-conflicts-or-at-least-have-less-of-them-109p>

1. Git fork là một bản copy của một repository, việc fork có thể giúp bạn dễ dàng chỉnh sửa, thay đổi các source code mà không ảnh hưởng đến source gốc, sau đó bạn có thể gửi một pull request đến tác giả để đóng góp cho project. [↑](#footnote-ref-1)