# Exercise 3: Git and Github

# 

* **Workspace (Không gian làm việc)**: *Nơi bạn thực hiện các thay đổi trên mã nguồn.*
* **Staging (Khu vực tổ chức)**: *Nơi bạn đánh dấu những thay đổi trong mã nguồn để chuẩn bị commit.*
* **Local Repository (Kho lưu trữ cục bộ)**: *Là bản lưu của dự án của bạn trên máy tính cá nhân.*
* **Remote Repository (Kho lưu trữ từ xa):** *Là bản lưu trữ của dự án trên một server từ xa.*

Dòng màu xanh lá cây, cam, vàng đại diện cho các lệnh git cụ thể:

* **git add/mv/rm**: *Thêm, di chuyển, hoặc xóa các tệp từ không gian làm việc sang khu vực tổ chức.*
* **git commit**: *Lưu trữ những thay đổi từ khu vực tổ chức vào kho lưu trữ cục bộ.*
* **git commit -a:** *Tự động thêm tất cả thay đổi từ không gian làm việc và commit chúng.*
* **git reset <file>**: *Loại bỏ các thay đổi đã đánh dấu trong khu vực tổ chức.*
* **git reset <commit>**: *Đặt lại trạng thái của kho lưu trữ cục bộ về một commit cụ thể.*
* **git diff**: *So sánh những thay đổi trong không gian làm việc với những thay đổi đã được commit.*
* **git diff HEAD:** *So sánh những thay đổi trong không gian làm việc với lần commit gần nhất.*
* **git clone/pull**: *Sao chép hoặc cập nhật mã nguồn từ kho lưu trữ từ xa.*
* **git push**: *Đẩy những thay đổi từ kho lưu trữ cục bộ lên kho lưu trữ từ xa.*
* **git fetch:** *Lấy tất cả những thay đổi từ kho lưu trữ từ xa mà không tự động merge hay rebase.*
* ***Pull request***: *Yêu cầu chủ sở hữu của kho lưu trữ từ xa merge những thay đổi của bạn.*

Phía bên phải của hình vẽ có hai cột là "Origin" và "Upstream":

* **Origin**: *Thường là bản sao của kho lưu trữ từ xa mà bạn có quyền ghi. Đây là nơi bạn đẩy (push) những thay đổi từ kho lưu trữ cục bộ của mình.*
* **Upstream**: Kho lưu trữ từ xa chính mà bạn fork từ đó. "Upstream" thường được dùng để đồng bộ hóa những thay đổi từ dự án gốc.

1. **Git**

***Objectives and Outcomes***

*Trong bài tập này, bạn sẽ học cách cài đặt Git trên máy tính của mình. Git là một công cụ cần thiết để sử dụng tất cả các công cụ khác dựa trên Node.js mà chúng ta sẽ gặp phải trong phần còn lại của khóa học. Kết thúc bài tập này, bạn sẽ có khả năng:*

1. **Cài đặt Git trên máy tính của bạn**: Bạn sẽ tải và cài đặt phần mềm Git, đây là bước đầu tiên để sử dụng công cụ quản lý phiên bản này.
2. **Đảm bảo rằng Git có thể được sử dụng từ dòng lệnh hoặc cửa sổ command-prompt trên máy tính của bạn:** Bạn sẽ cần mở terminal (trên macOS hoặc Linux) hoặc command-prompt / PowerShell (trên Windows) và gõ các lệnh git để kiểm tra xem Git đã được cài đặt chính xác và có thể chạy được không.
3. **Thiết lập một số cấu hình toàn cục cơ bản cho Git:** Bạn sẽ cấu hình một số tùy chọn toàn cục cho Git trên máy tính của mình, bao gồm việc thiết lập tên người dùng và địa chỉ email, những thông tin này sẽ được ghi lại mỗi khi bạn thực hiện một "commit" (lưu lại một phiên bản mới của mã nguồn).

***Downloading and Installing Git***

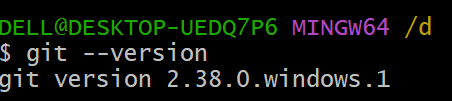
*Tải xuống bộ cài đặt Git: Truy cập vào trang web https://git-scm.com/downloads để tải xuống bộ cài đặt Git phù hợp với nền tảng máy tính của bạn (Windows, macOS, hoặc Linux).*

1. **Thực hiện cài đặt:** Sau khi đã tải xuống, mở bộ cài đặt và làm theo các bước hướng dẫn để cài đặt Git.
2. **Tìm hiểu thêm về cách cài đặt Git:** Nếu bạn cần thông tin chi tiết hơn về cách cài đặt, bạn có thể tham khảo tại https://git-scm.com/book/en/v2/Getting-Started-Installing-Git. Tài liệu này cung cấp nhiều cách khác nhau để cài đặt Git trên các nền tảng khác nhau.
3. **Cài đặt thông qua các công cụ GUI:** Cài đặt một số công cụ giao diện người dùng đồ họa (GUI) như GitHub Desktop cũng sẽ tự động cài đặt Git trên máy tính của bạn.
4. **Trên máy Mac:** Nếu bạn cài đặt công cụ dòng lệnh của XCode, Git cũng sẽ được cài đặt tự động.
5. Bạn có thể chọn bất kỳ phương pháp nào mà bạn thấy tiện lợi nhất để cài đặt Git. Mỗi phương pháp có những bước cụ thể riêng, nhưng kết quả cuối cùng là bạn sẽ có Git được cài đặt và sẵn sàng sử dụng trên máy tính của mình.

***Some Global Configuration for Git***

* **Mở cửa sổ dòng lệnh (cmd) hoặc terminal**: *Tùy thuộc vào hệ điều hành mà bạn đang sử dụng, mở cửa sổ Command Prompt (cmd) nếu bạn dùng Windows, hoặc Terminal nếu bạn dùng macOS hoặc Linux.*
* **Kiểm tra phiên bản Git:** Đ*ể chắc chắn rằng Git đã được cài đặt và có thể sử dụng được trong dòng lệnh, nhập lệnh sau và nhấn Enter:*

git --version



*Nếu Git đã được cài đặt, lệnh này sẽ hiển thị phiên bản của Git đang được cài trên máy bạn.*

* **Cấu hình tên người dùng cho Git**: *Để cài đặt tên người dùng mà Git sẽ sử dụng trong mỗi lần commit, nhập lệnh sau và thay "Your Name" bằng tên thật của bạn:*

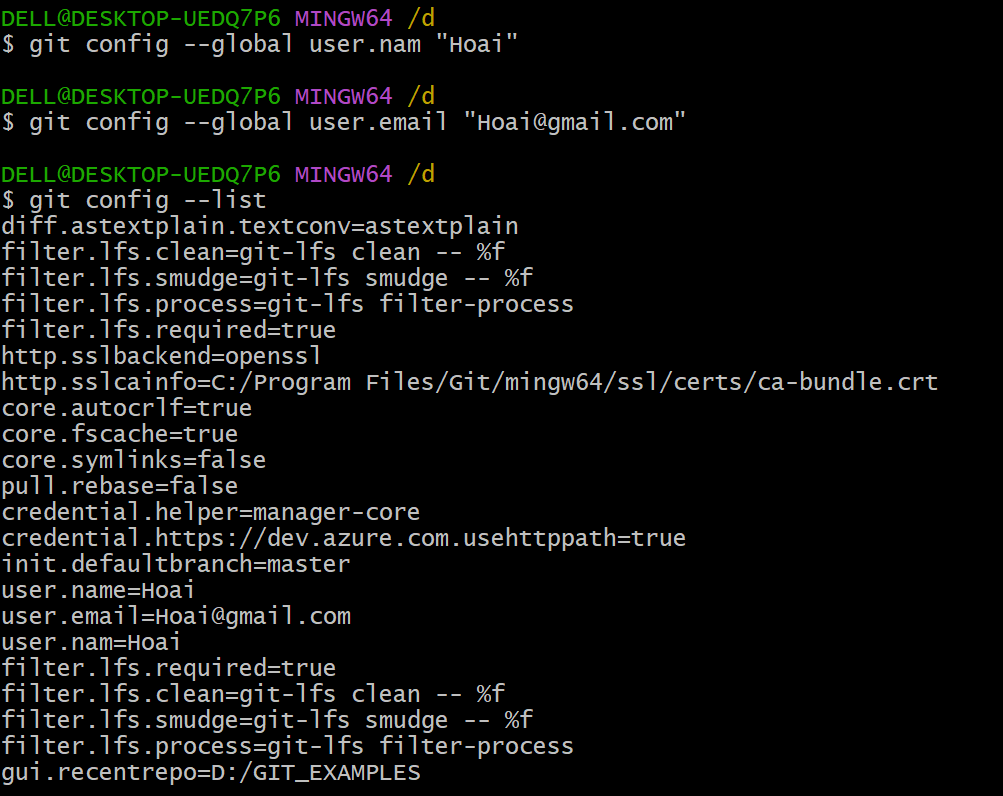
git config --global user.name "Your Name"

* **Cấu hình email cho Git:** *Để cài đặt địa chỉ email mà Git sẽ ghi vào mỗi lần commit, nhập lệnh sau và thay thế <your email address> bằng địa chỉ email thực của bạn:*

git config --global user.email <your email address>

* **Kiểm tra cấu hình Git toàn cục:** *Nếu bạn muốn xem tất cả các cài đặt cấu hình Git toàn cục của bạn, nhập lệnh sau:*

git config -–list

****

*Lệnh này sẽ liệt kê tất cả các cấu hình, bao gồm tên người dùng và email mà bạn đã thiết lập.*

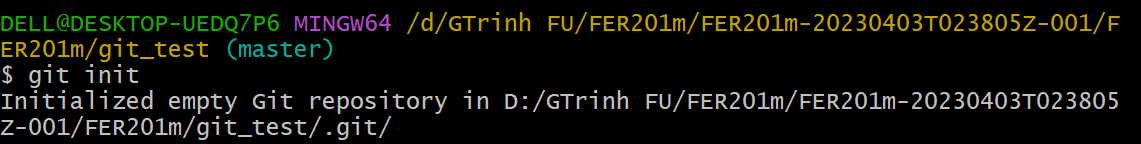
*Nhớ thay thế "Your Name" và "<your email address>" bằng thông tin cá nhân của bạn khi nhập các lệnh. Thực hiện những bước này giúp đảm bảo rằng các commit bạn thực hiện được gắn liền với danh tính của bạn, điều này quan trọng trong quá trình làm việc nhóm và quản lý mã nguồn.*

***Git Commands***

* ***Git init***

**Lệnh git init trong Git được sử dụng để khởi tạo một kho lưu trữ Git mới trong một thư mục cụ thể. Khi bạn chạy lệnh này, nó sẽ tạo ra một kho lưu trữ Git mới và trống hoặc tái khởi tạo một kho lưu trữ đã tồn tại.**

1. Khi bạn muốn bắt đầu theo dõi (quản lý phiên bản) cho một dự án mới trên máy tính của mình với Git, bạn cần phải tạo một "kho lưu trữ Git". Một kho lưu trữ Git là nơi mà tất cả các thông tin về những thay đổi bạn đã làm trong các tệp của dự án được lưu trữ.
2. Để làm điều này, bạn mở cửa sổ dòng lệnh (command line) hoặc terminal, di chuyển (cd) đến thư mục chứa dự án của bạn, và nhập lệnh git init. Ngay lập tức, một thư mục ẩn mới có tên .git sẽ được tạo ra trong thư mục dự án của bạn. Thư mục .git này chứa tất cả các tệp cần thiết để quản lý các phiên bản của dự án.

******

*Ví dụ trong hình ảnh bạn đã cung cấp, lệnh git init đã được chạy trong thư mục FER201lm/git\_test, và một kho lưu trữ Git mới đã được khởi tạo ở đó.*

*Bây giờ bạn đã có một kho lưu trữ Git, bạn có thể bắt đầu làm việc trên dự án của mình và sử dụng các lệnh Git khác để theo dõi các thay đổi, lưu chúng vào lịch sử của dự án, và chia sẻ dự án với người khác thông qua một kho lưu trữ từ xa như GitHub, GitLab, hoặc Bitbucket.*

* ***Git branch***

**Lệnh git branch** trong Git rất quan trọng và thường xuyên được sử dụng khi làm việc với các dự án phần mềm. Các nhánh (branch) cho phép nhiều nhà phát triển có thể làm việc đồng thời và song song trên cùng một dự án mà không làm ảnh hưởng đến nhau. Dưới đây là cách sử dụng lệnh git branch để tạo, xem và xóa các nhánh:

**Creating a new branch:**

git branch <tên-nhánh>

Thay <tên-nhánh> bằng tên bạn muốn đặt cho nhánh. Lệnh này chỉ tạo nhánh trên máy tính cá nhân của bạn (kho lưu trữ cục bộ).

git push -u <remote> <branch-name>

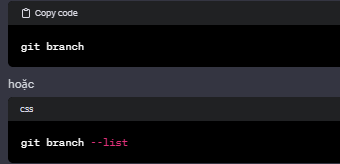
****

*Thay <tên-kho-từ-xa> với tên của kho lưu trữ từ xa (thường là origin) và <tên-nhánh> với tên nhánh bạn vừa tạo.*

**Viewing branches:**

*Để xem danh sách các nhánh trong kho lưu trữ của bạn, sử dụng lệnh:*

git branch or git branch --list

****

**Deleting a branch:**

*Khi bạn đã hoàn thành công việc trên một nhánh và muốn xóa nó, bạn có thể dùng lệnh sau:*

git branch -d <branch-name>

*Nhớ rằng mỗi nhánh trong Git giống như một luồng phát triển độc lập, nơi bạn có thể thêm các thay đổi mới mà không ảnh hưởng đến các nhánh khác. Việc này giúp quản lý dự án trở nên dễ dàng và linh hoạt hơn, đặc biệt là trong môi trường làm việc nhóm.*

* ***Git checkout***

Lệnh git checkout là một trong những lệnh thường xuyên được sử dụng nhất trong Git. Lệnh này cho phép bạn chuyển đổi giữa các nhánh khác nhau, hoặc khôi phục các tệp tin từ các commit trước đó.

* **Chuyển đổi nhánh:**

*Để làm việc trên một nhánh khác nhau, bạn cần chuyển đổi sang nhánh đó. Bạn sử dụng lệnh sau:*

git checkout <name-of-your-branch>

**

*Thay <tên-nhánh-của-bạn> với tên nhánh mà bạn muốn chuyển đến.*

* **Yêu cầu khi chuyển đổi nhánh:**
  + *Các thay đổi trên nhánh hiện tại của bạn phải được commit (lưu lại) hoặc được stash (lưu tạm) trước khi bạn chuyển đổi sang nhánh khác.*
  + *Nhánh mà bạn muốn chuyển đến phải tồn tại trên máy tính của bạn.*
* **Tạo và chuyển đổi nhánh cùng một lúc:**

*Nếu bạn muốn tạo một nhánh mới và chuyển đến nhánh đó ngay lập tức, bạn có thể sử dụng lệnh:*

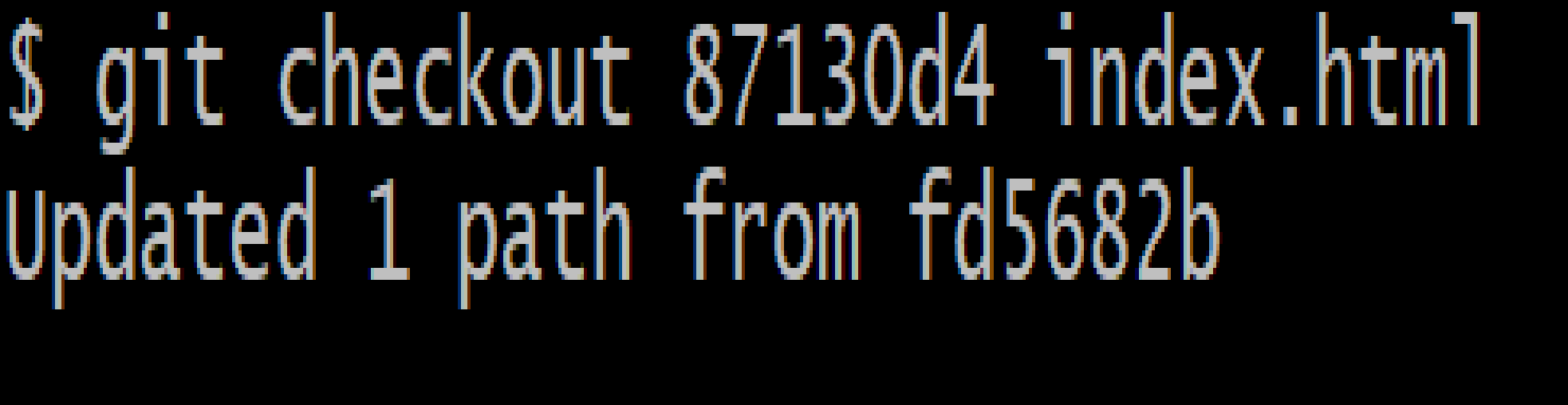
git checkout -b <name-of-your-branch>

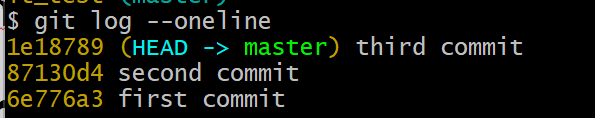


*Lệnh này sẽ tạo ra một nhánh mới (-b là viết tắt của từ "branch") và chuyển bạn đến nhánh đó ngay sau khi nó được tạo.*

Khôi phục tệp tin từ commit:

* Bạn cũng có thể sử dụng git checkout để khôi phục tệp tin từ một commit cụ thể trong quá khứ. Ví dụ trong hình ảnh bạn đã cung cấp, lệnh:





*Đây là những cách sử dụng cơ bản của lệnh git checkout, giúp bạn linh hoạt quản lý công việc trên các nhánh khác nhau và khôi phục các tệp tin cũ.*

* ***Git status***

*Lệnh git status trong Git cung cấp thông tin cần thiết về nhánh hiện tại mà bạn đang làm việc. Đây là một lệnh hữu ích giúp bạn theo dõi trạng thái của các tệp trong dự án và những thay đổi đã được thực hiện.*

git status

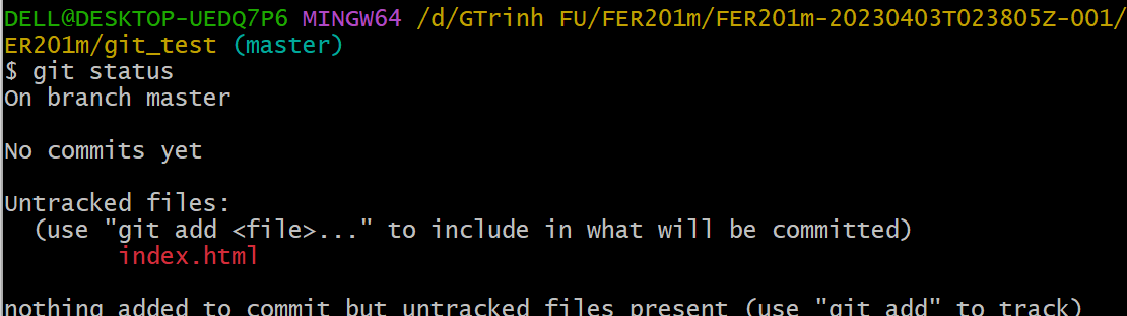
Khi bạn chạy lệnh git status trong terminal hoặc command prompt, bạn sẽ nhận được thông tin như sau:

* **Trạng thái cập nhật của nhánh**: *Lệnh sẽ cho bạn biết liệu nhánh hiện tại có đang cập nhật với nhánh tương ứng trên kho lưu trữ từ xa (remote) hay không.*
* **Thay đổi cần được commit, push hoặc pull:**
  + **Commit:** *Nếu có thay đổi trong các tệp đã được thêm vào khu vực tổ chức (staged), bạn cần commit những thay đổi này.*
  + **Push:** *Nếu đã commit nhưng chưa đẩy (push) những thay đổi lên kho lưu trữ từ xa, lệnh sẽ thông báo.*
  + **Pull:** *Nếu nhánh từ xa có những thay đổi mà bạn chưa cập nhật, lệnh sẽ nhắc bạn cần phải kéo (pull) chúng về.*
* **Tình trạng của các tệp:**
  + **Staged**: *Các tệp đã được thêm vào khu vực tổ chức, sẵn sàng cho việc commit.*
  + **Unstaged:** *Các tệp có thay đổi nhưng chưa được thêm vào khu vực tổ chức.*
  + **Untracked:** *Các tệp mới được thêm vào dự án nhưng chưa được Git theo dõi.*
* **Các tệp được tạo, sửa đổi hoặc xóa:**

Lệnh git status sẽ cung cấp danh sách cụ thể về tình trạng của từng tệp.

*Thông qua lệnh git status, bạn có thể dễ dàng quản lý công việc của mình, biết được những gì cần làm tiếp theo để đảm bảo rằng công việc phát triển phần mềm của bạn được theo dõi và cập nhật một cách chính xác.*





**Git status gives information about the branch & files**

* ***Git add***

*Lệnh git add trong Git được sử dụng để thêm các thay đổi vào khu vực tổ chức (staging area) trước khi chúng được commit vào kho lưu trữ. Đây là những điều cơ bản bạn cần biết về lệnh này:*

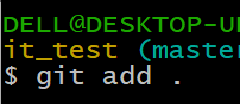
*Khi tạo, chỉnh sửa hoặc xóa một tệp: Những thay đổi này chỉ xảy ra trên máy tính cá nhân của bạn và sẽ không được tự động bao gồm trong lần commit tiếp theo, trừ khi bạn có cấu hình đặc biệt nào đó.*

**To add a single file:** *Sử dụng lệnh git add <tên-tệp>, thay <tên-tệp> bằng tên thực tế của tệp tin bạn muốn thêm.*

git add <file>

**To add everything at once:** *Sử dụng lệnh* ***git add -A*** *hoặc* ***git add .*** *để thêm tất cả các thay đổi trong thư mục hiện tại vào khu vực tổ chức.*

git add -A



* **Tệp tin có màu đỏ**: *Trong giao diện dòng lệnh, các tệp tin màu đỏ là những tệp tin chưa được đưa vào khu vực tổ chức (unstaged). Nghĩa là những thay đổi trong các tệp này sẽ không được ghi nhận khi bạn thực hiện commit.*
* **Tệp tin có màu xanh**: *Sau khi bạn sử dụng lệnh git add, các tệp tin sẽ chuyển sang màu xanh là tín hiệu cho biết chúng đã được thêm vào khu vực tổ chức (staged) và sẽ được ghi nhận trong commit tiếp theo.*
* **Quan trọng**: *Lệnh git add không làm thay đổi kho lưu trữ và các thay đổi không được lưu cho đến khi bạn sử dụng lệnh git commit để tạo một bản ghi (commit) mới trong lịch sử của kho lưu trữ.*

*⇒ Nhớ rằng git add chỉ là bước đầu tiên trong chu trình làm việc với Git để theo dõi các thay đổi. Sau khi đã thêm các thay đổi vào khu vực tổ chức, bạn sẽ cần phải sử dụng lệnh git commit để lưu chúng vào lịch sử của dự án.*

* ***Git commit***

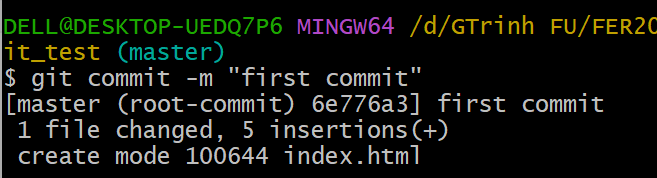
*Lệnh git commit trong Git rất quan trọng và thường xuyên được sử dụng. Khi bạn đạt đến một điểm nhất định trong quá trình phát triển phần mềm, bạn sẽ muốn "lưu" các thay đổi của mình, có thể là sau khi hoàn thành một nhiệm vụ cụ thể hoặc giải quyết một vấn đề nào đó.*

* **"Đặt một điểm kiểm tra":** *Bạn có thể coi việc thực hiện git commit như là việc đặt một "điểm kiểm tra" (checkpoint) trong quá trình phát triển phần mềm. Bạn có thể quay trở lại điểm kiểm tra này sau này nếu cần thiết.*
* **Viết một thông điệp ngắn:** *Khi thực hiện git commit, bạn cần viết một thông điệp ngắn gọn để giải thích những gì bạn đã phát triển hoặc thay đổi trong mã nguồn. Điều này giúp người khác (hoặc chính bạn sau này) hiểu được lý do tại sao thay đổi được thực hiện. Ví dụ, sử dụng lệnh sau để commit với thông điệp "commit message":*

git commit -m "commit message"

*Thay "commit message" bằng mô tả cụ thể về những gì bạn đã thay đổi.*

* **Lưu thay đổi cục bộ:** *Quan trọng là phải nhớ rằng git commit chỉ lưu các thay đổi của bạn một cách cục bộ trên máy tính của bạn, và không ảnh hưởng đến kho lưu trữ từ xa (remote repository) cho đến khi bạn sử dụng lệnh git push để đẩy các commit đó lên.*
* *Trong hình ảnh mà bạn đã cung cấp, có một ví dụ về cách sử dụng git commit. Cụ thể, lệnh git commit -m "first commit" đã được sử dụng để tạo một commit đầu tiên với thông điệp "first commit". Commit này đã lưu trữ các thay đổi vào lịch sử của nhánh master trong kho lưu trữ cục bộ.*



* ***Git push***

*Lệnh git push trong Git được sử dụng để đẩy (upload) các commit từ kho lưu trữ cục bộ của bạn lên kho lưu trữ từ xa (remote repository), thường là để chia sẻ với những người khác hoặc để lưu trữ bản sao dự phòng.*

**

git push <remote> <branch-name>

*Thay <tên-kho-từ-xa> với tên của kho lưu trữ từ xa (thường là origin) và <tên-nhánh> với tên nhánh mà bạn muốn đẩy lên.*

**Đẩy nhánh mới lên kho lưu trữ từ xa**: *Nếu bạn đã tạo một nhánh mới và muốn đẩy nó lên kho lưu trữ từ xa lần đầu tiên, bạn sẽ cần thiết lập một upstream (dòng chảy từ xa) cho nhánh đó:*

**

git push --set-upstream <remote> <name-of-your-branch>

or



git push -u origin <branch\_name>

* **Quan trọng:** *git push chỉ đẩy những thay đổi đã được commit. Nếu bạn có các tệp hoặc thay đổi trong trạng thái tổ chức (staged) hoặc chưa được tổ chức (unstaged) mà chưa được commit, chúng sẽ không được đẩy lên kho lưu trữ từ xa.*

*Khi sử dụng git push, nó sẽ gửi tất cả các commit trên nhánh hiện tại của bạn lên kho lưu trữ từ xa, làm cho các thay đổi của bạn có sẵn cho những người khác hoặc lưu lại trên server. Điều này là một phần quan trọng của quy trình làm việc với Git, đặc biệt khi làm việc nhóm.*

* ***Git pull***

*Lệnh git pull trong Git được sử dụng để cập nhật những thay đổi mới nhất từ kho lưu trữ từ xa (remote repository) về máy tính cá nhân của bạn.*

* *Lấy cập nhật từ kho từ xa: Khi bạn chạy lệnh git pull, Git sẽ thực hiện hai bước:*
  + ***git fetch:*** *Lệnh này sẽ lấy tất cả các thay đổi từ nhánh hiện tại của kho lưu trữ từ xa mà bạn chưa có trên máy tính cá nhân của mình.*
  + ***git merge:*** *Sau khi lấy các thay đổi, lệnh này sẽ tự động hợp nhất (merge) chúng vào nhánh hiện tại mà bạn đang làm việc trên máy tính cá nhân.*

***Để sử dụng lệnh git pull, bạn nhập:***

******

git pull <remote>

Thay <tên-kho-từ-xa> với tên của kho lưu trữ từ xa mà bạn muốn cập nhật (thường là origin).

* **Xung đột có thể xảy ra:** Khi thực hiện git pull, có thể xảy ra xung đột giữa các thay đổi từ kho lưu trữ từ xa và những thay đổi cục bộ của bạn. Nếu điều này xảy ra, bạn sẽ cần phải giải quyết xung đột bằng tay trước khi có thể tiếp tục.

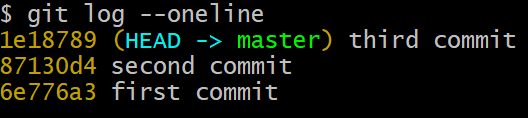
Ví dụ: Nếu một đồng nghiệp đã thay đổi một dòng mã trong một tệp mà bạn cũng đang chỉnh sửa, khi bạn git pull, Git sẽ thông báo xung đột và yêu cầu bạn chọn giữa thay đổi của bạn hay thay đổi từ kho từ xa.

Sử dụng git pull thường xuyên là một phần quan trọng trong quy trình làm việc để đảm bảo rằng bạn luôn làm việc trên phiên bản mới nhất của dự án và giảm thiểu khả năng xung đột với công việc của người khác.

* ***Git log -- oneline***

*Lệnh git log --oneline trong Git được sử dụng để xem lịch sử commit trong một định dạng ngắn gọn và đơn giản, hiển thị mỗi commit trên một dòng riêng. Nó cung cấp một cái nhìn tổng quan và rút gọn về các thông điệp commit và mã hash đi kèm với chúng.*

* ***Hiển thị lịch sử commit****: Khi bạn chạy lệnh này, bạn sẽ nhìn thấy danh sách các commit đã được thực hiện trong kho lưu trữ của bạn, với mỗi commit được hiển thị trên một dòng.*
* ***Mỗi dòng bao gồm****: Mã hash của commit (là một chuỗi các ký tự dùng để định danh duy nhất cho mỗi commit) và thông điệp mô tả nhanh chóng về những gì commit đó thay đổi.*



* **Ví dụ: Nếu bạn chạy lệnh git log --oneline, bạn có thể thấy kết quả như sau:**
  + le18789 (HEAD -> master) third commit: Mã hash le18789 là mã định danh cho commit thứ ba trên nhánh master, và HEAD đang chỉ tới commit này, nghĩa là đây là commit cuối cùng bạn đang đứng.
  + 87130d4 second commit: Mã hash 87130d4 là mã định danh cho commit thứ hai.
  + 6e776a3 first commit: Mã hash 6e776a3 là mã định danh cho commit đầu tiên.

Đây là một công cụ hữu ích để nhanh chóng xem qua và hiểu được dòng thời gian của các thay đổi trong dự án. Đặc biệt khi bạn muốn tìm một commit cụ thể mà không cần xem đầy đủ chi tiết của nó.

* ***Git revert***

Sometimes we need to undo the changes that we've made. There are various ways to undo our changes locally or remotely (depends on what we need), but we must carefully use these commands to avoid unwanted deletions.

A safer way that we can undo our commits is by using **git revert**. To see our commit history, first we need to use

Then we just need to specify the hash code next to our commit that we would like to undo:

git revert 3321844

After this, you will see a screen like below - just press **shift + q** to exit

The advantage of using **git revert** is that it doesn't touch the commit history. This means that you can still see all of the commits in your history, even the reverted ones.

Another safety measure here is that everything happens in our local system unless we push them to the remote repo. That's why git revert is safer to use and is the preferred way to undo our commits.



* ***Git merge***

When you've completed development in your branch and everything works fine, the final step is merging the branch with the parent branch (dev or master). This is done with the **git merge** command.

Git merge basically integrates your feature branch with all of its commits back to the dev (or master) branch. It's important to remember that you first need to be on the specific branch that you want to merge with your feature branch.

For example, when you want to merge your feature branch into the dev branch:

**First you should switch to the dev branch:**

git checkout dev

**Before merging, you should update your local dev branch:**

git fetch

**Finally, you can merge your feature branch into dev:**

git merge <branch-name>

* ***Push helloapp into Git***

You can initialize your project to be a Git repository by typing the following commands at the prompt:

git init

git add .

git commit -m "Initial Setup"

Thereafter you can set up an online Git repository and synchronize your project to the online repository. Make sure that the online Git repository is a ***private*** repository.

* ***Conclusions***

At the end of this exercise, you should have Git available on the command-line of your computer.

1. **GitHub**

***Objectives and Outcomes***

In this exercise you will learn to use GitHub. GitHub has become an essential platform for developers, enabling collaboration, version control, and open-source contributions. It provides a robust infrastructure for managing projects, tracking changes, and facilitating teamwork. Whether you're an individual developer or part of a team, GitHub offers a powerful set of tools to enhance your development workflow:

* Set up the online repository as a remote repository for your local Git repository
* Push your commits to the online repository
* Clone an online Git repository to your computer

***Setting up an Online Git repository***

* Sign up for an account either at Bitbucket ([https://bitbucket.org](https://bitbucket.org/)) or GitHub ([https://github.com](https://github.com/)).
* Then set up an online Git repository named **git-test**. Note the URL of your online Git repository. Note that private repositories on GitHub requires a paid account, and is not available for free accounts.

***Git Commands***

* ***Set the local Git repository to set its remote origin***

At the prompt, type the following to set up your local repository to link to your online Git repository:

git remote add origin <repository URL>

* ***Pushing your commits to the online repository***

At the prompt, type the following to push the commits to the online repository:

git push -u origin master

* ***Cloning an online repository***

To clone an online repository to your computer, type the following at the prompt:

git clone <repository URL>

***Conclusions***

In this exercise you have learnt to set up an online Git repository, synchronize your local repository with the remote repository, and clone an online repository.