**Buổi 1: Github làm quen cơ bản**

**1.1. Setup git**

- what/why

- cài đặt git (check chuột phải, option git bash)

- create github account

- setup username/email

**1.2. Tạo project với git**

- clone 1 project có sẵn

=> git clone link\_https

**Buổi 2:Github cơ bản**

**2.1. Tạo 1 project trên github:**

- Bước 1:Tạo 1 github repository (folder) - nơi chứa code

- Bước 2:Clone project này: git clone link\_your\_repo

- Bước 3:Viết code

+git status ( kiểm tra trạng thái files)

+git add . (thêm files)

+git commit -m “your\_message” (tạo nội dung commit)

+git push origin main(Đẩy các thay đổi lên repository từ xa)

**2.2. Tạo project local và đẩy lên github:**

**-Hoặc nếu branch tồn tại trên remote:**

+git fetch origin

+git checkout (ten nhanh,vd:” Exercise-Map”)

**-Kiểm tra danh sách các branch**:

+git branch -a : Lệnh này sẽ hiển thị tất cả branch hiện có trên local và remote. Xác nhận xem branch Exercise-Map có tồn tại hay không.

**-Push branch lên GitHub**: Sử dụng lệnh sau để đẩy branch lên remote:

+git push -u origin <branch-name>

-Kiểm tra remote repository đã được thiết lập chưa:

+git remote -v

**-Tạo và chuyển sang nhánh mới** Nếu bạn muốn làm việc trên nhánh mới mà vẫn giữ nguyên mã hiện tại:

git checkout -b ten-nhanh-moi

* **-b**: Tạo nhánh mới.
* Sau khi chạy lệnh này, toàn bộ mã hiện tại sẽ được gắn với nhánh mới.

**-Xóa nhánh cục bộ (local branch)**Dùng lệnh sau để xóa nhánh trong repository local:

-git branch -d ten-nhanh

* **-d**: Xóa nhánh, nhưng chỉ khi nhánh đó đã được hợp nhất hoàn toàn vào nhánh hiện tại hoặc một nhánh khác.
* Nếu nhánh chưa được hợp nhất, bạn cần dùng:

-git branch -D ten-nhanh

**-D**: Xóa nhánh mà không kiểm tra trạng thái hợp nhất (force delete).

**-Xóa nhánh trên remote (remote branch)** Sử dụng lệnh sau để xóa nhánh khỏi remote repository:

git push origin --delete ten-nhanh

* **--delete**: Xóa nhánh trên remote.

-Sau khi xóa, chạy lệnh sau để xác nhận các nhánh remote đã bị xóa:

git fetch --prune

git branch -r

* **git fetch --prune**: Xóa các tham chiếu nhánh remote đã bị xóa khỏi danh sách local của bạn.
* **git branch -r**: Hiển thị các nhánh remote còn tồn tại.

-Nếu bạn không muốn thiết lập tracking, bạn có thể chỉ định rõ remote và nhánh trong lệnh git pull:

-git pull origin master

**-Bật chế độ hỗ trợ đường dẫn dài trong Git**

Chạy lệnh sau trong Git Bash hoặc Command Prompt (Admin):

git config --global core.longpaths true

\* Bước 1: Tạo 1 github repo

\* Bước 2:

***- git init:*** Lệnh này được sử dụng để khởi tạo một kho lưu trữ Git mới trong thư mục hiện tại. Nó sẽ tạo ra một thư mục ẩn .git chứa tất cả các tệp tin cần thiết cho một kho lưu trữ Git.

\* Bước 3: Viết code

***- git status:***

+ Lệnh này hiển thị trạng thái hiện tại của kho lưu trữ. Nó sẽ cho bạn biết những tệp nào đã được thay đổi nhưng chưa được thêm vào khu vực staging, và những tệp nào đã được thêm vào staging nhưng chưa được commit.

+ Ví dụ: Sau khi viết mã, bạn có thể chạy git status để xem những tệp nào đã thay đổi.

***- git add . :***

+ Lệnh này thêm tất cả các tệp đã thay đổi vào khu vực staging (sẵn sàng để commit). Dấu chấm (.) biểu thị rằng tất cả các tệp tin đã thay đổi trong thư mục hiện tại sẽ được thêm vào staging.

+ Ví dụ: Sau khi kiểm tra trạng thái với git status, bạn sử dụng git add . để chuẩn bị tất cả các thay đổi cho commit.

***- git commit -m "your message":***

+ Lệnh này tạo một commit mới với tất cả các thay đổi trong khu vực staging.

+ Thông điệp commit (ở đây là "your message") nên mô tả ngắn gọn nhưng rõ ràng về những thay đổi bạn đã thực hiện.

***- git remote add origin https://.....:***

+ Lệnh này thêm một remote repository (kho lưu trữ từ xa) có tên là "origin" với URL bạn cung cấp. Thường thì "origin" là tên mặc định cho kho lưu trữ từ xa chính.

+Ví dụ: git remote add origin https://github.com/username/repo.git sẽ thêm kho lưu trữ từ xa của bạn từ GitHub.

***- git push origin master:***

+ Lệnh này đẩy (push) các commit của nhánh hiện tại (thường là nhánh master) lên kho lưu trữ từ xa được chỉ định (ở đây là "origin").

+ Ví dụ: git push origin master sẽ đẩy các commit của nhánh master lên kho lưu trữ từ xa tại URL được thiết lập trong git remote add origin.

**Buổi 3:Github tiếp**

**3. Https vs ssh**

**3.1. HTTPS**

***-Định nghĩa****:* HTTPS (HyperText Transfer Protocol Secure) là một phiên bản an toàn của HTTP, sử dụng mã hóa SSL/TLS để bảo mật dữ liệu được truyền tải giữa máy tính của bạn và máy chủ.

***-Ưu điểm:***

+Dễ sử dụng: Không cần cài đặt thêm phần mềm hoặc cấu hình thêm. Bạn chỉ cần tên đăng nhập và mật khẩu GitHub của mình.

+Phổ biến: HTTPS thường được mở trên nhiều mạng và không yêu cầu cấu hình đặc biệt của tường lửa hoặc proxy.

-***Nhược điểm:***

+Yêu cầu xác thực thường xuyên: Mỗi lần bạn đẩy (push) hoặc kéo (pull) từ GitHub, bạn cần nhập lại tên đăng nhập và mật khẩu. Điều này có thể gây phiền toái, mặc dù có thể được giảm thiểu bằng cách sử dụng các công cụ quản lý thông tin đăng nhập như Git Credential Manager.

**3.2. SSH**

***- Định nghĩa:*** SSH (Secure Shell) là một giao thức mạng để vận hành các dịch vụ mạng một cách an toàn qua một mạng không bảo mật. Trong trường hợp của GitHub, SSH cung cấp một cách an toàn để kết nối với repository của bạn.

***- Ưu điểm:***

+ Bảo mật cao: SSH sử dụng cặp khóa công khai và khóa riêng để xác thực, thay vì sử dụng mật khẩu. Điều này cung cấp mức độ bảo mật cao hơn.

+ Không cần xác thực thường xuyên: Sau khi thiết lập, bạn không cần phải nhập mật khẩu mỗi lần bạn tương tác với GitHub.

***- Nhược điểm:***

+ Cài đặt phức tạp hơn: Cần phải tạo và cấu hình cặp khóa SSH. Điều này có thể phức tạp đối với người dùng mới.

+ Khả năng tương thích: Một số mạng hoặc tường lửa có thể chặn giao thức SSH, làm cho việc sử dụng SSH trở nên khó khăn trong một số môi trường mạng.

**3.3. Cách sử dụng**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Buổi 5:Github cuối**

**5.1. Làm việc nhóm**

**\* git pull = git fetch + git merge**

-git fetch: Lệnh này tải về tất cả các thay đổi từ repository từ xa (remote repository) về máy cục bộ của bạn mà không thực hiện hợp nhất (merge). Các thay đổi này sẽ được lưu trữ trong các nhánh từ xa (remote branches).

git merge: Lệnh này kết hợp các thay đổi từ nhánh hiện tại với một nhánh khác. Sau khi tải về các thay đổi từ repository từ xa bằng git fetch, bạn cần sử dụng git merge để kết hợp các thay đổi đó vào nhánh hiện tại.

-git pull: Lệnh này là sự kết hợp của git fetch và git merge. Khi bạn chạy git pull, Git sẽ tự động tải về các thay đổi từ repository từ xa và sau đó thực hiện hợp nhất (merge) các thay đổi đó vào nhánh hiện tại của bạn.

**\* git merge**

- git merge <branch>: Lệnh này kết hợp các thay đổi từ nhánh được chỉ định (nhánh nguồn) vào nhánh hiện tại (nhánh đích). Ví dụ, nếu bạn đang ở trên nhánh main và chạy git merge feature, Git sẽ hợp nhất các thay đổi từ nhánh feature vào nhánh main.

- Lệnh git merge có thể tạo ra các xung đột nếu có các thay đổi không tương thích giữa hai nhánh. Bạn sẽ cần phải giải quyết xung đột này bằng cách chỉnh sửa các tệp tin và thực hiện commit để hoàn tất quá trình hợp nhất.

**\* git conflict**

- Xung đột (conflict) xảy ra khi Git không thể tự động hợp nhất các thay đổi giữa hai nhánh. Điều này thường xảy ra khi cùng một phần của một tệp tin đã được sửa đổi theo những cách khác nhau trên hai nhánh khác nhau.

- Khi gặp xung đột, Git sẽ đánh dấu các khu vực bị xung đột trong các tệp tin.

-Bạn cần phải mở các tệp tin này, giải quyết các xung đột bằng cách chọn hoặc chỉnh sửa các thay đổi, và sau đó thêm các tệp đã chỉnh sửa vào khu vực staging và commit các thay đổi.

**\* git checkout branch/commit**

- git checkout <branch>: Lệnh này chuyển nhánh làm việc hiện tại của bạn sang nhánh được chỉ định. Ví dụ, git checkout feature sẽ chuyển bạn sang nhánh feature.

- git checkout <commit>: Lệnh này cho phép bạn chuyển đến một commit cụ thể bằng cách sử dụng mã hash của commit. Điều này giúp bạn kiểm tra mã nguồn tại thời điểm của commit đó. Ví dụ: git checkout abc1234 sẽ chuyển bạn đến commit với mã hash abc1234.

**\* git checkout -b branch\_name**

- git checkout -b <branch\_name>: Lệnh này tạo một nhánh mới có tên <branch\_name> và chuyển bạn ngay lập tức sang nhánh mới đó. Ví dụ, git checkout -b feature sẽ tạo một nhánh mới tên là feature và chuyển bạn sang nhánh đó.

**\* git checkout branch\_name**

- git checkout <branch\_name>: Lệnh này chuyển bạn sang nhánh có tên <branch\_name>. Ví dụ, nếu bạn đã tạo một nhánh tên là feature, bạn có thể chuyển sang nhánh đó bằng cách sử dụng lệnh này.