**Git thực hành (1) - Hệ thống quản lý phiên bản**

**1. Phiên bản (Version):**

* Phiên bản là các bản khác nhau của dự án (tập tin, thư mục, mã nguồn).
* Giúp theo dõi và khôi phục các thay đổi theo thời gian.

**2. Hệ thống quản lý phiên bản (VCS):**

* Phần mềm giúp lưu trữ và quản lý các phiên bản của dự án.
* Chức năng chính:
  + Khôi phục phiên bản.
  + Theo dõi thay đổi.
  + Khôi phục nội dung bị xóa.
  + So sánh các phiên bản.
* Phân loại:
  + Hệ thống quản lý phiên bản cục bộ (Local VCS).
  + Hệ thống quản lý phiên bản tập trung (Centralized VCS - CVCS).
  + Hệ thống quản lý phiên bản phân tán (Distributed VCS - DVCS).

**3. Hệ thống quản lý phiên bản cục bộ (Local VCS):**

* Phương pháp đơn giản, lưu trữ các phiên bản trên máy tính cục bộ.
* Ví dụ: RCS.
* Hạn chế: khó quản lý khi số lượng phiên bản lớn, không hỗ trợ cộng tác.

**4. Hệ thống quản lý phiên bản tập trung (Centralized VCS - CVCS):**

* Sử dụng máy chủ trung tâm để lưu trữ các phiên bản.
* Máy khách lấy và cập nhật phiên bản từ máy chủ.
* Ví dụ: CVS, Subversion (SVN).
* Hạn chế: phụ thuộc vào máy chủ trung tâm, có thể mất dữ liệu nếu máy chủ gặp sự cố.

**5. Hệ thống quản lý phiên bản phân tán (Distributed VCS - DVCS):**

* Mỗi máy khách có bản sao đầy đủ của kho chứa (repository).
* Không phụ thuộc vào máy chủ trung tâm.
* Ví dụ: Git, Mercurial, Darcs.
* Ưu điểm:
  + Không bị mất dữ liệu nếu máy chủ gặp sự cố.
  + Hỗ trợ cộng tác linh hoạt.
  + Cho phép liên kết với nhiều kho chứa từ xa.

**Tóm lại:**

* Hệ thống quản lý phiên bản là công cụ quan trọng trong phát triển phần mềm.
* Hệ thống quản lý phiên bản phân tán (DVCS) như Git mang lại nhiều ưu điểm so với hệ thống quản lý phiên bản cục bộ và tập trung.
* Git là một hệ thống quản lý phiên bản phân tán rất mạnh mẽ và đang được sử dụng rộng rãi hiện nay.

BÀI TẬP

**Câu 1.1 Các đặc điểm của hệ thống quản lý phiên bản cục bộ. Phát biểu nào không đúng?**

A. Có thể thực hiện thủ công

B. Không hỗ trợ trong môi trường cộng tác nhiều người

C. Có thể dùng phần mềm để quản lý phiên bản kiểu cục bộ

D. Các phiên bản của dự án được lưu tập trung trên một máy server

(Sai vì Local VCS lưu trên máy cục bộ, không phải server)

**Câu 1.2 Các đặc điểm của hệ thống quản lý phiên bản tập trung. Phát biểu nào không đúng?**

A. Các phiên bản của dự án được lưu tập trung trên máy server

B. Các máy client sẽ chứa tất cả các phiên bản của thư mục dự án cùng với lịch sử thay đổi

(Sai vì client trong CVCS không chứa toàn bộ lịch sử thay đổi)

C. Máy client không thể tải phiên bản của dự án về, khi máy server không hoạt động

D. Hỗ trợ làm việc cộng tác nhiều người

**Câu 1.3 Các đặc điểm của hệ thống quản lý phiên bản phân tán. Phát biểu nào không đúng?**

A. Các máy client sẽ chứa toàn bộ các phiên bản của dự án, cùng lịch sử thay đổi

B. Hỗ trợ làm việc cộng tác nhiều người

C. Các phiên bản của dự án được lưu trên máy server

(Sai vì không chỉ server mà client cũng lưu trữ toàn bộ phiên bản trong DVCS)

D. Bạn không thể tạo và lưu phiên bản khi không có kết nối mạng tới máy server

**Câu 1.4 Tìm trang chủ của các phần mềm quản lý phiên bản.**

| **Tên phần mềm** | **Trang chủ** |
| --- | --- |

|  |  |
| --- | --- |
| **RCS (Revision Control System)** | [**https://www.gnu.org/software/rcs**](https://www.gnu.org/software/rcs) |

|  |  |
| --- | --- |
| **CVS (Concurrent Versions System)** | [**https://www.nongnu.org/cvs**](https://www.nongnu.org/cvs) |

|  |  |
| --- | --- |
| **Subversion (SVN)** | [**https://subversion.apache.org**](https://subversion.apache.org) |

|  |  |
| --- | --- |
| **Git** | [**https://git-scm.com**](https://git-scm.com) |

|  |  |
| --- | --- |
| **Mercurial** | [**https://www.mercurial-scm.org**](https://www.mercurial-scm.org) |

|  |  |
| --- | --- |
| **Darcs** | [**http://darcs.net**](http://darcs.net) |

**Git thực hành (2) - Tổng quan về Git**

**2.1. Git là gì:**

* Git là một hệ thống quản lý phiên bản phân tán (DVCS) phổ biến trong phát triển phần mềm.
* Nó cho phép theo dõi và quản lý các thay đổi trong mã nguồn dự án một cách hiệu quả.
* Các tính năng chính:
  + Theo dõi lịch sử thay đổi.
  + Cộng tác hiệu quả.
  + Phân nhánh và hợp nhất.
  + Bảo mật.
  + Phân tán.

**2.2. Tải và cài đặt Git:**

* Hướng dẫn tải Git từ trang web chính thức (git-scm.com).
* Hướng dẫn kiểm tra cài đặt Git bằng lệnh git --version hoặc git -v trong cửa sổ dòng lệnh.

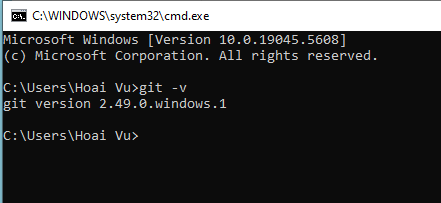
**2.3. Nhúng Git vào dự án:**

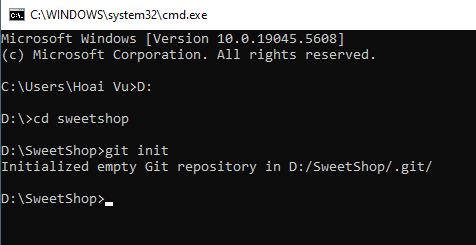
* Hướng dẫn tạo thư mục dự án và nhúng Git bằng lệnh git init.
* Giải thích về thư mục .git và vai trò của kho chứa (repository).
* Các lệnh cơ bản để di chuyển trong command line, như cd, dir...
* Hướng dẫn hiện các thư mục ẩn.

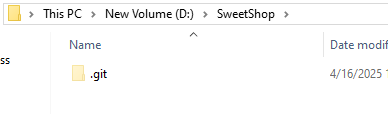
**Tóm lại:**

* Git là một công cụ mạnh mẽ để quản lý mã nguồn.
* Việc nhúng Git vào dự án giúp theo dõi và quản lý các thay đổi một cách hiệu quả.
* Việc sử dụng thành thạo các lệnh trong Git giúp cho việc quản lý mã nguồn trở nên dể dàng hơn.

BÀI TẬP:







**2.2 Lệnh nào được sử dụng để nhúng Git vào thư mục dự án?**

A. git init

**Đúng** - Lệnh này tạo thư mục ẩn **.git** trong thư mục hiện tại, đánh dấu thư mục đó thành kho lưu trữ Git.

B. git --init

C. git initialize

D. git embed

**2.3 Lệnh nào sử dụng để kiểm tra trên máy tính đã có phần mềm Git hay chưa?**

A. git ver

B. git version

**Đúng** - Lệnh này hiển thị phiên bản Git đang cài đặt.

C. git -v

D. git –ver

**2.4 Trong Git, kho lưu trữ (repo, repository) là gì?**

A. Là thư mục dự án

B. Là thư mục dự án đã được nhúng Git

**Đúng** - Sau khi chạy git init, thư mục trở thành kho lưu trữ Git.

C. Là thư mục cài đặt phần mềm Git

D. Là thư mục *.git* (trong thư mục dự án)

**Git thực hành (3) - Cấu hình định danh người dùng**

Tuyệt vời! Bạn đã trình bày chi tiết và rõ ràng về cách cấu hình định danh người dùng trong Git. Dưới đây là tóm tắt các điểm chính:

**3.1. Định danh người dùng:**

* **Tại sao cần cấu hình:**
  + Xác định tác giả của các thay đổi (commit).
  + Theo dõi lịch sử đóng góp.
  + Phân công trách nhiệm.
* **Lệnh git config:**
  + Được sử dụng để đọc, ghi hoặc cập nhật cấu hình Git.
  + Có ba phạm vi cấu hình:
    - system: Áp dụng cho tất cả người dùng và kho lưu trữ trên hệ thống.
    - global: Áp dụng cho tài khoản người dùng hiện tại và tất cả kho lưu trữ của người dùng đó.
    - local: Áp dụng cho một kho lưu trữ cụ thể.
* **Độ ưu tiên:**
  + local > global > system.
* **Cấu hình tên và email:**
  + Sử dụng git config --<scope> user.name "Tên của bạn" và git config --<scope> user.email "email@example.com".
* **Xem thông tin cấu hình:**
  + Sử dụng git config --list.
* **Kinh nghiệm thực tế:**
  + Kiểm tra cấu hình trước khi commit.
  + Nên cấu hình global.
  + Cấu hình local cho dự án đặc biệt.

**3.2. Xóa hoặc thay đổi định danh:**

* **Xóa định danh:**
  + Sử dụng git config --<scope> --unset user.name và git config --<scope> --unset user.email.
* **Thay đổi định danh:**
  + Sử dụng git config --<scope> user.name "Tên mới" và git config --<scope> user.email "email\_mới@example.com".

**Tóm lại:**

* Cấu hình định danh người dùng là bước quan trọng để theo dõi và quản lý các thay đổi trong Git.
* Hiểu rõ về các phạm vi cấu hình giúp bạn linh hoạt hơn trong việc quản lý thông tin.

BÀI TẬP:

Bài tập 3.1 Bài tập tình huống “Quản lý cấu hình Git cho nhiều dự án”

Giả sử bạn là Nguyễn Văn Tèo, một lập trình viên đang làm việc tại công ty TeoTech. Ngoài công việc chính, bạn còn tham gia phát triển một dự án mang tên VienVong (viển vông) cùng với một nhóm bạn vào thời gian rảnh.

Để quản lý mã nguồn hiệu quả, bạn sử dụng Git cho cả hai dự án. Tuy nhiên, bạn muốn đảm bảo rằng:

- Khi làm việc tại công ty TeoTech, các commit của bạn sẽ được gắn với thông tin định danh của công ty (tên và email công ty).

- Khi làm việc với dự án "viển vông", các commit của bạn sẽ được gắn với thông tin định danh cá nhân.

***Yêu cầu:***

- Tạo hai thư mục dự án riêng biệt: TeoTechProject (cho dự án công ty). VienVongProject (cho dự án cá nhân)

- Khởi tạo kho lưu trữ Git trong mỗi thư mục dự án.

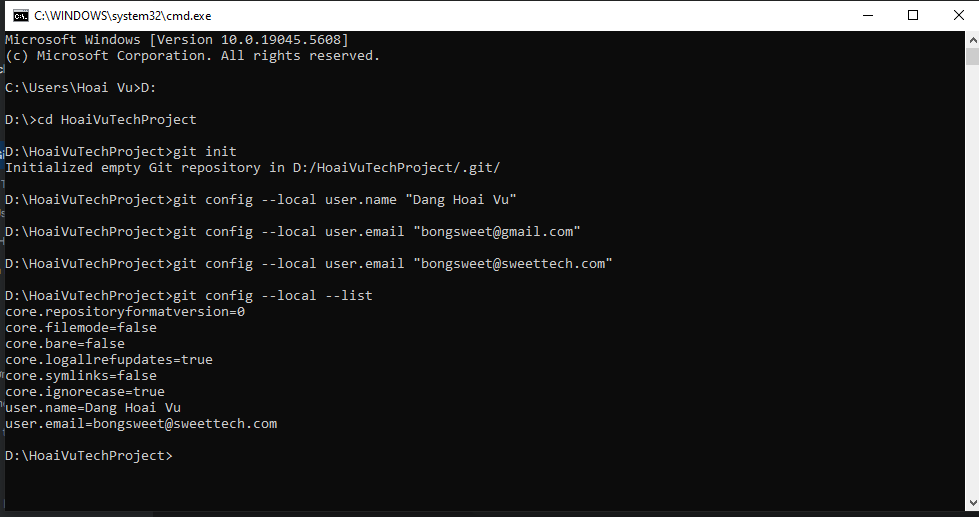
- Cấu hình Git sao cho:

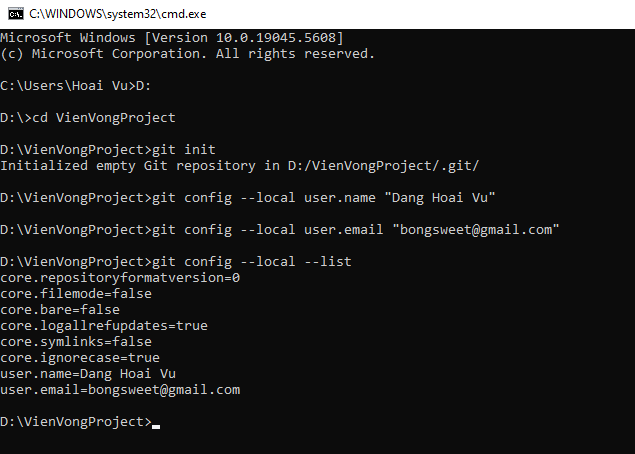
  + Khi commit trong dự án TeoTechProject, Git sử dụng thông tin định danh: Tên: "Nguyen Van Teo"; Email: nvteo@teotech.com

  + Trong commit trong dự án VienVongProject, Git sử dụng thông tin định danh: Tên: "Nguyen Van Teo"; Email: nvteo@gmail.com

***Mục tiêu:***

Bài tập này giúp các bạn hiểu rõ hơn về cách cấu hình Git cho nhiều dự án với các thông tin định danh khác nhau, đồng thời rèn luyện kỹ năng làm việc với Git trong môi trường thực tế.





**Câu 3.2 Phạm vi cấu hình định danh trong Git là gì? Phát biểu nào sau đây không đúng?**

A. Phạm vi *system*áp dụng cho tất cả người dùng và kho lưu trữ trên hệ thống.

B. Phạm vi *global* áp dụng cho tài khoản người dùng hiện tại và tất cả kho lưu trữ mà người dùng đó làm việc.

C. Phạm vi local áp dụng cho một kho lưu trữ cụ thể và có độ ưu tiên thấp nhất.

**Sai** - Phạm vi **local** có độ ưu tiên **cao nhất**, không phải thấp nhất. Nếu có cấu hình **local**, nó sẽ ghi đè lên cấu hình **global** và **system** cho kho lưu trữ đó.

D. Phạm vi local áp dụng cho một kho lưu trữ cụ thể và có độ ưu tiên cao nhất.

**Git thực hành (4) - Các khu vực làm việc của Git**

**4. Các khu vực làm việc của Git:**

* **Ba khu vực làm việc:**
  + Thư mục làm việc (working directory): Nơi bạn chỉnh sửa tập tin trực tiếp.
  + Khu vực tổ chức tạm (staging area): Vùng trung gian, lưu trữ các thay đổi đã chọn để commit.
  + Kho chứa (repository): Lưu trữ tất cả các tập tin và lịch sử thay đổi của dự án.
* **Mối quan hệ giữa các khu vực:**
  + Chỉnh sửa mã nguồn ở Thư mục làm việc.
  + Chuẩn bị nội dung commit ở Khu tạm (staging area).
  + Lưu trữ mã nguồn và lịch sử ở Kho chứa (repository).
* **Khảo sát 3 khu vực trong dự án:**
  + Thư mục làm việc: Thư mục dự án (ví dụ: TeoShop).
  + Khu tạm: Tập tin index trong thư mục .git.
  + Kho chứa: Thư mục .git.

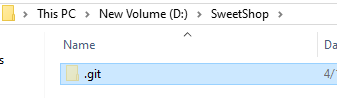
**4.2. Thư mục làm việc:**

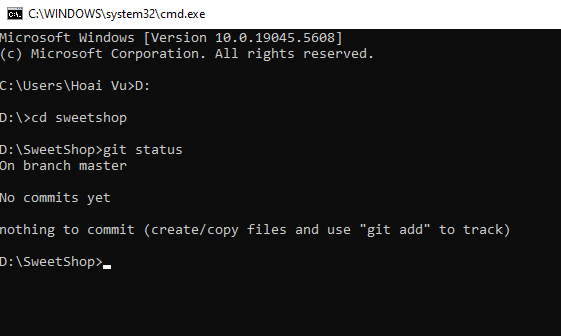
* **Mục đích:**
  + Chỉnh sửa mã nguồn.
  + Xem trạng thái dự án.
  + Thử nghiệm.
* **Cách có Thư mục làm việc:**
  + Khởi tạo kho chứa mới (git init).
  + Sao chép kho chứa có sẵn (git clone).
* **Lệnh git add:**
  + Dùng để đưa thay đổi từ Thư mục làm việc vào Khu tạm.
  + Cú pháp:
    - git add <tập\_tin>: Thêm một tập tin cụ thể.
    - git add .: Thêm tất cả thay đổi.
    - git add -p: thêm từng phần thay đổi.
* **Lệnh git status:**
  + Hiển thị trạng thái hiện tại của Kho lưu trữ và Thư mục làm việc.
* **Thực hành với git status và git add:**
  + Thực hiện các thao tác trên command line để kiểm tra trạng thái của file, và cách add file vào khu vực staging.
  + Giải thích các trạng thái của file trên git.

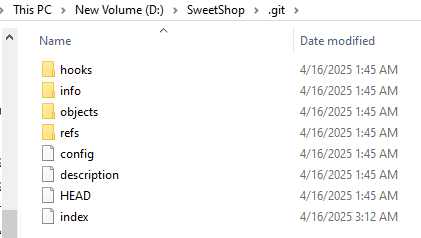
**Tóm lại:**

* Hiểu rõ về ba khu vực làm việc của Git giúp quản lý mã nguồn hiệu quả.
* Lệnh git add giúp chọn lọc các thay đổi để đưa vào commit.
* Lệnh git status giúp kiểm tra trạng thái của các file.

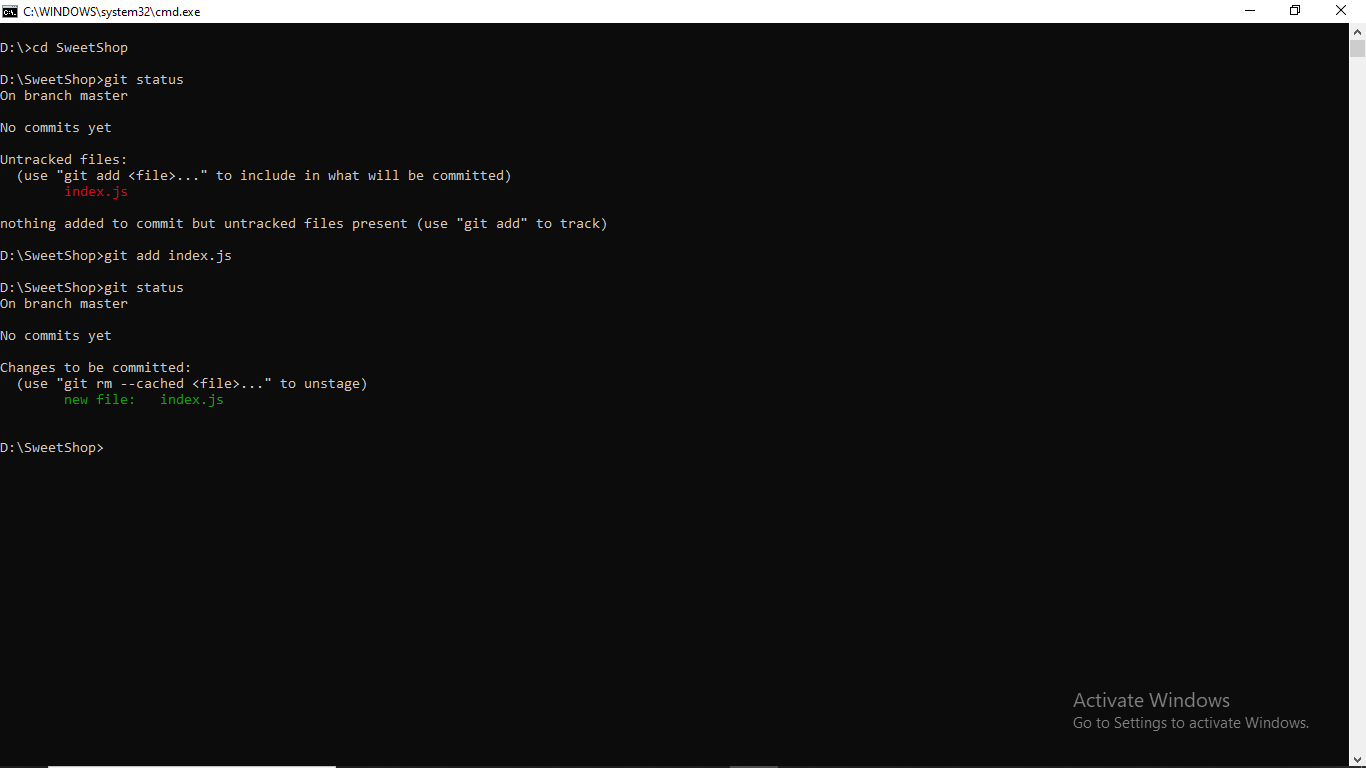
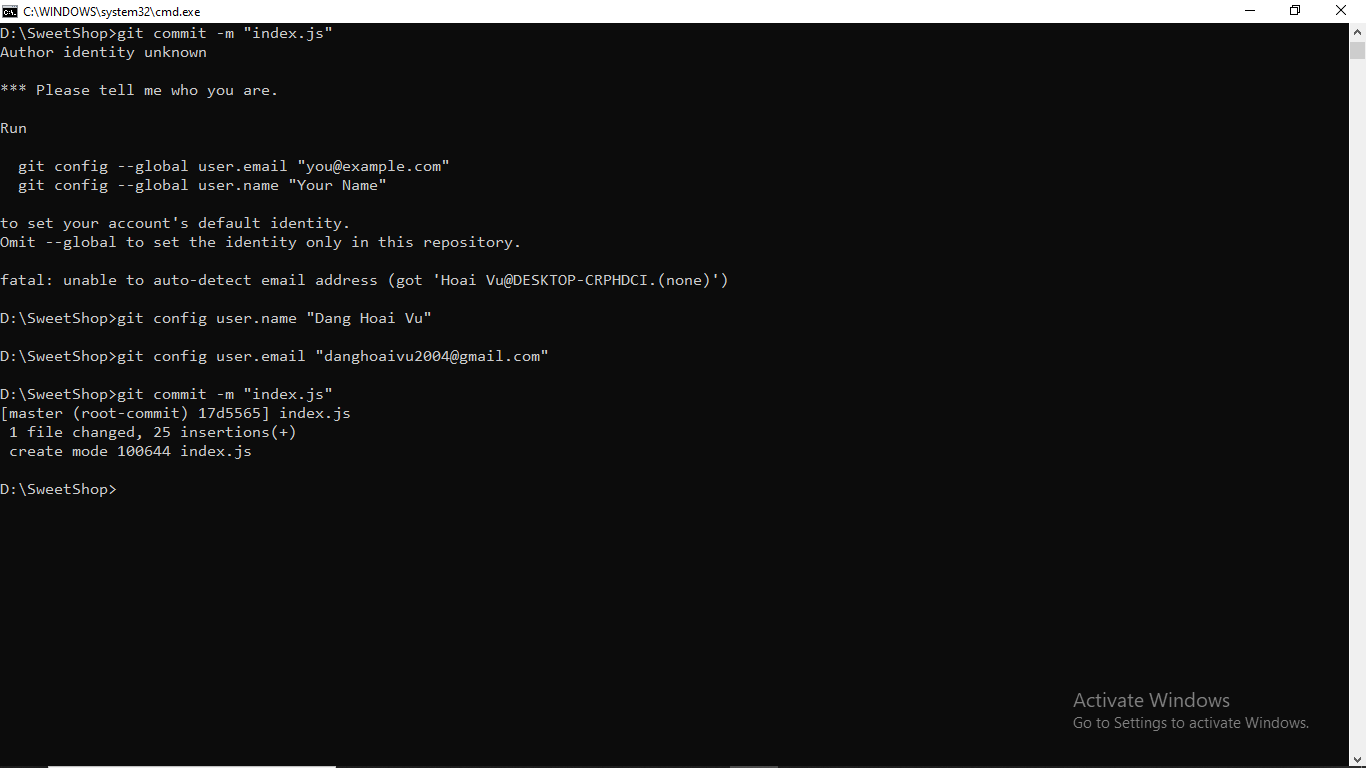
BÀI TẬP







***Tập tin trong khu vực tổ chức tạm.***

  
  
**Câu 4.2 Ba khu vực làm việc chính của Git là gì? Phát biểu nào sau đây không đúng?**

A. Thư mục làm việc (working directory) là nơi bạn chỉnh sửa tập tin trực tiếp.

B. Khu tạm (staging area) là nơi lưu trữ lịch sử phiên bản của dự án.

**Sai** - Khu tạm không lưu trữ lịch sử phiên bản của dự án. Nó chỉ lưu trữ các thay đổi tạm thời mà bạn đã thêm vào để chuẩn bị commit.

C. Kho chứa (repository) là nơi lưu trữ dữ liệu chính thức của dự án.

D. Lệnh git add được sử dụng để chuyển các thay đổi từ Thư mục làm việc sang Khu tạm.

**Câu 4.3 Thư mục làm việc (working directory) trong Git là gì? Phát biểu nào sau đây không đúng?**

A. Là thư mục thực tế trên máy tính, nơi bạn làm việc trực tiếp với các tập tin của dự án.

B. Là nơi bạn tạo, sửa, hoặc xóa tập tin trước khi đưa thay đổi vào .git.

C. Là nơi lưu trữ lịch sử phiên bản và dữ liệu chính thức của dự án.

**Sai** - Thư mục làm việc không lưu trữ lịch sử phiên bản, mà kho chứa (repository) mới là nơi lưu trữ lịch sử phiên bản.

D. Phản ánh các thay đổi so với commit cuối cùng, giúp bạn quyết định những gì cần thêm vào Khu tạm (staging area).

**Git thực hành (5) - Các khu vực làm việc của Git (tiếp)**

**5.1. Khu tạm (Staging Area):**

* Là khu vực trung gian giữa Thư mục làm việc và Kho chứa.
* Cho phép chọn lọc các thay đổi để commit.
* Lệnh git rm --cached <tên\_tập\_tin>: Gỡ bỏ tập tin khỏi Khu tạm.

**5.2. gitignore:**

* Cơ chế bỏ qua các tập tin/thư mục không cần thiết.
* Sử dụng tập tin .gitignore để chỉ định các mẫu bỏ qua.
* Công dụng:
  + Tránh commit rác.
  + Giữ lịch sử sạch.
  + Tăng hiệu quả.
* Thực hành:
  + Tạo tập tin .gitignore và thêm các mẫu bỏ qua.
  + Kiểm tra kết quả bằng git status.
* Lưu ý:
  + Tập tin .gitignore cũng cần được commit.
  + Có thể tham khảo các mẫu file .gitignore có sẵn trên mạng.

**5.3. Kho chứa (Repository):**

* Nơi lưu trữ tất cả tập tin và lịch sử thay đổi.
* Thư mục .git là Kho chứa cục bộ.
* Hai loại kho chứa:
  + Kho chứa cục bộ (local repository).
  + Kho chứa ở xa (remote repository).
* Lệnh git clone để sao chép kho chứa từ xa.

**5.4. Lệnh git commit:**

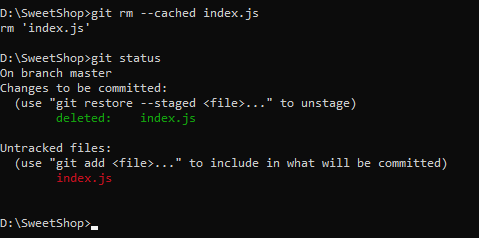
* Lưu thay đổi từ Khu tạm vào Kho chứa.
* Tạo "ảnh chụp" (snapshot) mới trong lịch sử phiên bản.
* Vai trò:
  + Ghi nhận thay đổi.
  + Tạo lịch sử.
  + Hỗ trợ cộng tác.
* Cú pháp:
  + git commit -m "Thông điệp": Commit với thông điệp ngắn.
  + git commit: Mở trình soạn thảo để viết thông điệp chi tiết.
  + git commit -a -m "Thông điệp": Tự động thêm và commit các tập tin đã theo dõi.
* Quy trình làm việc:
  + Chỉnh sửa tập tin.
  + git add.
  + git commit.
* Cách viết thông điệp commit:
  + Dòng đầu tiên (Subject): Ngắn gọn, dùng động từ mệnh lệnh.
  + Dòng trống.
  + Phần mô tả chi tiết (nếu cần).
  + Ví dụ về cách viết thông điệp ngắn gọn, và thông điệp chi tiết.
* Kinh nghiệm:
  + Viết thông điệp ngay khi commit.
  + Viết thông điệp dài nếu cần.
  + Sử dụng công cụ hỗ trợ kiểm tra thông điệp commit.

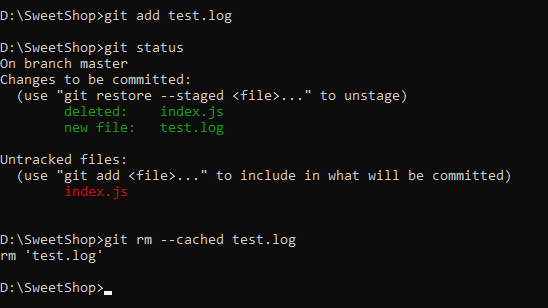
**Tóm lại:**

* Khu tạm giúp tổ chức các thay đổi trước khi commit.
* .gitignore giúp loại bỏ các tập tin không cần thiết.
* Kho chứa là nơi lưu trữ lịch sử dự án.
* git commit là lệnh quan trọng để lưu lại các thay đổi.
* Cách viết thông điệp commit rõ ràng giúp quản lý dự án hiệu quả.

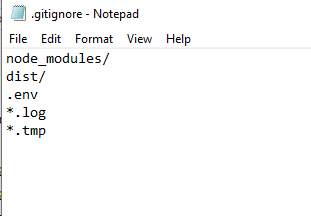
BÀI TẬP:

**gitignore**

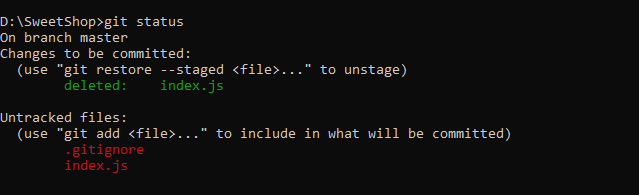


****

***Thực hiện thêm và gỡ test.log ra khỏi Khu tạm***

******

***File .gitignore cho phép bạn chỉ định những file và thư mục nào Git sẽ bỏ qua.***

******

**Câu hỏi 5.2 Khu tạm (staging area) trong Git là gì? Phát biểu nào sau đây không đúng?**

A. Khu tạm là một khu vực trung gian giữa Thư mục làm việc (working directory) và Kho chứa (repository).

B. Khu tạm là một thư mục vật lý trong Thư mục làm việc.

**Sai** - Khu tạm không phải là một thư mục vật lý. Nó là một khu vực trong bộ nhớ được Git sử dụng để lưu trữ các thay đổi trước khi commit.

C. Khu tạm cho phép bạn chọn lọc và tổ chức các thay đổi bạn muốn đưa vào commit tiếp theo.

D. Lệnh git rm --cached <tên\_tập\_tin> được sử dụng để gỡ bỏ tên tập tin ra khỏi Khu tạm.

**Câu hỏi 5.3 gitignore là gì? Phát biểu nào sau đây không đúng?**

A. gitignore giúp thêm các tập tin không liên quan (ví dụ: node\_modules, tập tin log) vào kho chứa.

**Sai** - gitignore giúp **bỏ qua** các tệp không liên quan, không phải thêm chúng vào kho chứa.

B. gitignore là một cơ chế trong Git, cho phép bạn chỉ định các tập tin hoặc thư mục mà Git sẽ bỏ qua khi thực hiện các lệnh như git add hoặc git status.

C. gitignore được thực hiện thông qua tập tin có tên là .gitignore, thường đặt ở thư mục gốc của dự án.

D. Mục đích của gitignore là loại bỏ các tập tin không cần thiết khỏi lịch sử phiên bản, như tập tin tạm, tập tin nhị phân, hoặc thư viện phụ thuộc.

**Câu hỏi 5.4 Kho chứa (repository) trong Git là gì? Phát biểu nào sau đây không đúng?**

A. Kho chứa là nơi lưu trữ tất cả các tập tin và lịch sử thay đổi của một dự án.

B. Thư mục .git trong dự án chính là Kho chứa.

C. Lệnh git commit được sử dụng để chuyển tập tin từ Thư mục làm việc (working directory) sang Kho chứa.

D. Kho chứa cục bộ (local repository) là kho chứa được lưu trữ trên một máy chủ ở xa.

**Sai** - Kho chứa cục bộ (local repository) là kho chứa trên máy tính của bạn. Kho chứa trên máy chủ ở xa gọi là kho chứa từ xa (remote repository).

**Câu 5.5 Commit trong Git là gì? Phát biểu nào sau đây không đúng?**

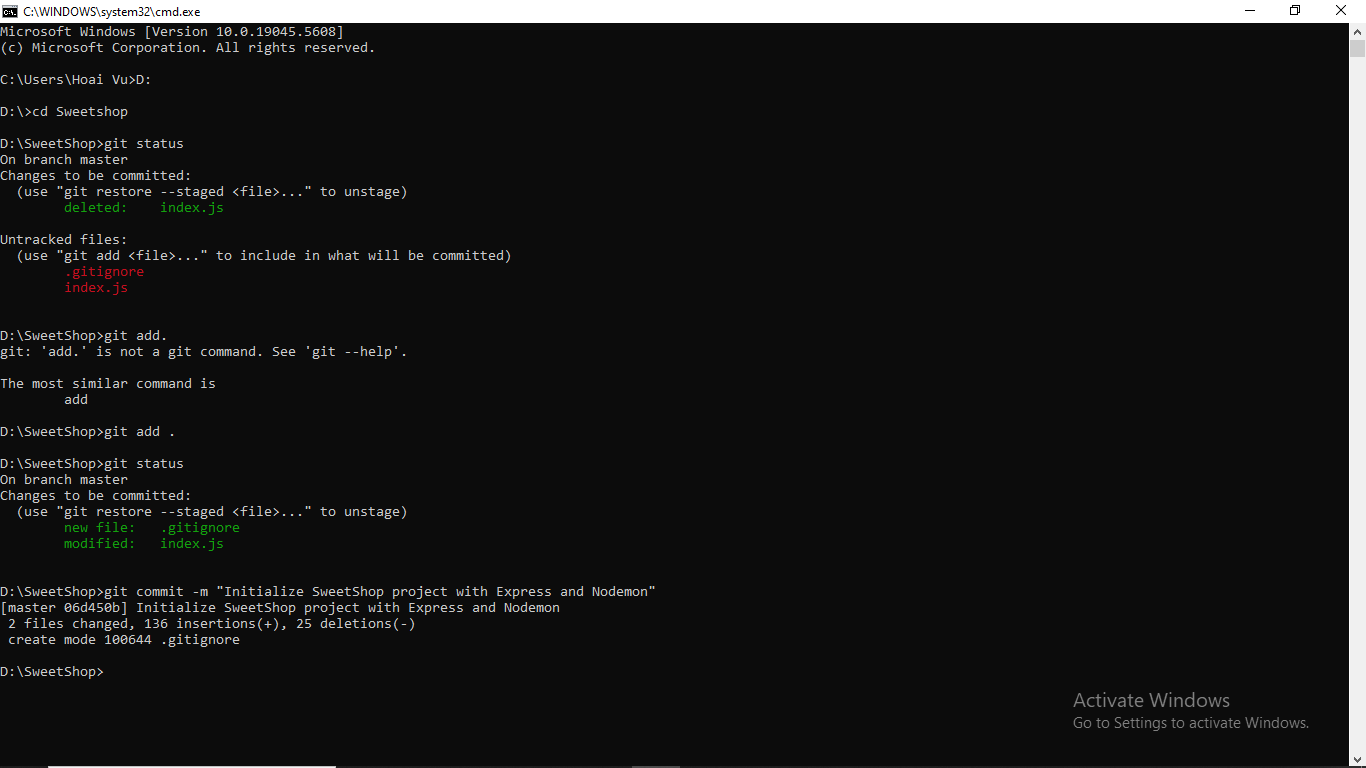
A. Commit là lệnh trong Git dùng để lưu các thay đổi từ Khu tạm vào Kho chứa.

B. Commit tạo ra một "ảnh chụp" (snapshot) mới trong lịch sử phiên bản.

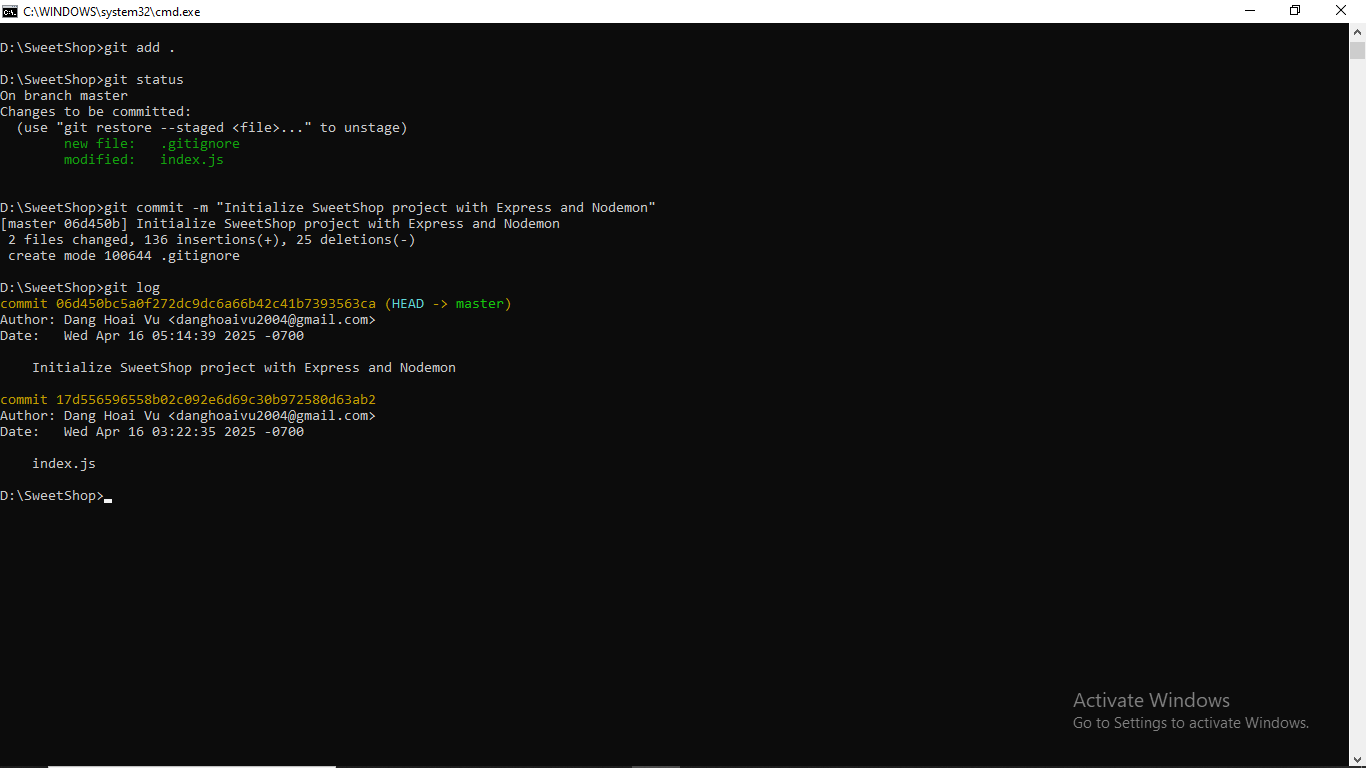
C. Commit dùng để chuyển tập tin từ Kho chứa sang Khu tạm.

**Sai** - git commit không chuyển tập tin từ kho chứa vào khu tạm. Lệnh dùng để đưa các thay đổi vào khu tạm là git add

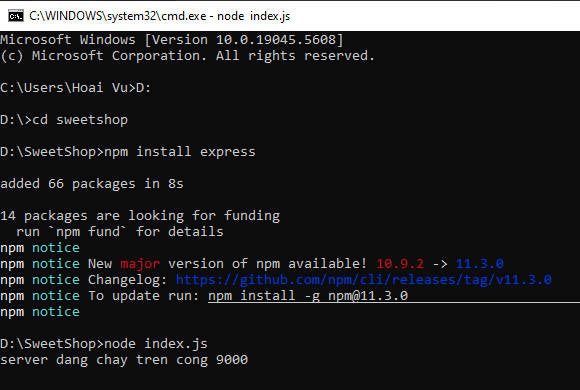
D. Mỗi commit đi kèm một thông điệp (message) mô tả thay đổi.



Sử dụng lệnh git add . (dấu chấm đại diện cho tất cả các file trong thư mục hiện tại):



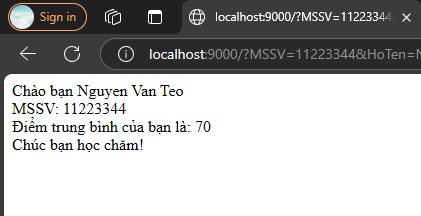
Kết quả sẽ hiển thị commit đầu tiên mà bạn vừa tạo, cùng với thông tin về tác giả, ngày commit, và thông điệp commit.



* ***Bạn đã di chuyển đến thư mục dự án của mình và cài đặt gói express cần thiết để chạy ứng dụng web của bạn.***
* ***Bạn đã khởi động thành công web server bằng Node.js, và server đang chạy trên cổng 9000.***
* ***Bạn có thể truy cập ứng dụng web của mình bằng cách mở trình duyệt và truy cập*** [***http://localhost:9000***](http://localhost:9000)***.***

**Web back-end (13) - Lập trình client-server**

**Bài 13.1**

****

**Bài 13.2**

Câu hỏi 13.2 URL (Uniform Resource Locator) là gì? Phát biểu nào sau đây không đúng?

A. URL là một địa chỉ duy nhất xác định vị trí của một tài nguyên trên Internet.

B. URL bao gồm các thành phần như giao thức, tên miền, đường dẫn, truy vấn và phân mảnh.

C. Phần "truy vấn" (query) trong URL được sử dụng để truyền dữ liệu đến máy chủ thông qua các tham số.

D. Giao thức "FTP" là giao thức phổ biến nhất được sử dụng trong URL để truy cập các trang web.

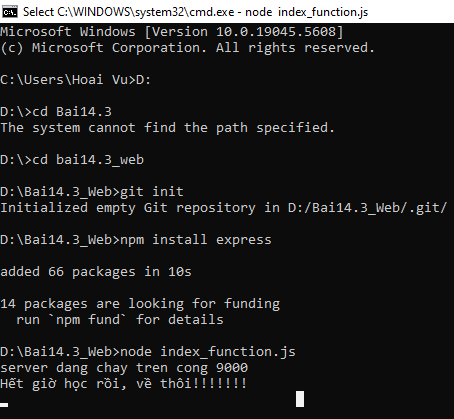
**HTTP (Hypertext Transfer Protocol)** và **HTTPS (HTTP Secure)** là hai giao thức phổ biến nhất được sử dụng trong URL để truy cập các trang web. HTTP là giao thức cơ bản cho World Wide Web, còn HTTPS là phiên bản bảo mật hơn với mã hóa SSL/TLS.

**FTP (File Transfer Protocol)** là một giao thức được sử dụng để truyền tệp tin giữa máy khách và máy chủ. Mặc dù URL có thể sử dụng giao thức ftp:// để trỏ đến các tài nguyên FTP, nhưng nó không phải là giao thức phổ biến nhất để truy cập các trang *web*.

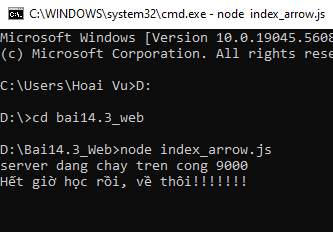
**Web back-end (14) - Lập trình hướng sự kiện**

**Bài 14.3**

***[1] Viết theo kiểu hàm thông thường***

****

***[2] Viết theo kiểu hàm mũi tên***

****