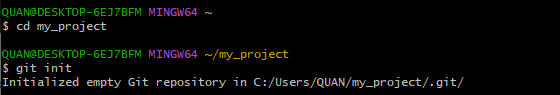
**CHƯƠNG IV: CÁC LỆNH CƠ BẢN TRONG GIT**

1. **GIT INIT**

* **Git Init:** Khởi tạo một kho lưu trữ Git mới trong thư mục hiện tại.
* Khi sử dụng câu lệnh git init, Git sẽ tạo ra một thư mục ẩn có tên .git trong thư mục hiện tại của bạn. Thư mục này sẽ chứa toàn bộ các thông tin cần thiết để quản lý phiên bản của dự án của bạn, bao gồm các commit, nhánh, và các thiết lập khác.
* **Ví dụ:** Để khởi tạo một kho lưu trữ Git mới cho một dự án đang được lưu trữ trong thư mục "my\_project", bạn có thể chạy lệnh sau:



1. **GIT CLONE**

* **Git clone:** Sao chép một kho lưu trữ Git từ một kho lưu trữ trực tuyến sang máy tính cá nhân của bạn.
* Câu lệnh git clone được sử dụng để sao chép một kho lưu trữ Git từ một kho lưu trữ trực tuyến, chẳng hạn như GitHub hoặc Bitbucket, sang máy tính cá nhân của bạn. Khi thực thi lệnh, Git sẽ tải về toàn bộ lịch sử của kho lưu trữ đó và tạo ra một bản sao của nó trên máy tính của bạn.
* **Ví dụ:** Để sao chép một kho lưu trữ Git từ GitHub sang máy tính của bạn, bạn có thể chạy lệnh sau trong dòng lệnh:



Trong đó, <https://github.com/username/repo.git> là đường dẫn đến kho lưu trữ trên GitHub mà bạn muốn sao chép. Git sẽ tải về toàn bộ lịch sử của kho lưu trữ đó và tạo ra một bản sao của nó trên máy tính của bạn.

1. **GIT ADD**

* **Git Add:** Thêm các tệp đã sửa đổi hoặc tạo mới vào mục theo dõi của Git.
* Câu lệnh git add được sử dụng để thêm các tệp đã sửa đổi hoặc tạo mới vào mục theo dõi của Git. Khi thực thi lệnh, Git sẽ đánh dấu các tệp đã được thêm vào mục theo dõi và chuẩn bị cho việc tạo commit.
* **Ví dụ:** Để thêm tệp "index.html" vào mục theo dõi của Git, bạn có thể chạy lệnh sau trong dòng lệnh:



Nếu bạn muốn thêm tất cả các tệp đã sửa đổi hoặc tạo mới trong thư mục hiện tại vào mục theo dõi của Git, bạn có thể chạy lệnh sau:



1. **GIT COMMIT**

* **Git Commit:** Tạo một "commit" mới để lưu trữ các thay đổi vào kho lưu trữ.
* Các câu lệnh **Git Commit**:
  + **Git commit:** Tạo một commit mới với các thay đổi đã được đưa vào staged area, và hiển thị trình soạn thảo để nhập thông điệp commit.
  + **Git commit -m “message”:** Tạo một commit mới và nhập thông điệp commit trực tiếp trong lệnh.
  + **Gid commit -a:** Tạo một commit mới với các thay đổi đã được thêm và commit các tệp tin đã được chỉnh sửa hoặc đã bị xóa mà không cần sử dụng lệnh git add.
  + **Git commit --amend:** Chỉnh sửa commit cuối cùng. Lệnh này cho phép bạn thay đổi thông điệp commit hoặc thay đổi các thay đổi đã được thêm vào commit cuối cùng.
  + **Git commit -s:** Tạo một commit ký số với chữ ký GPG. Lệnh này được sử dụng khi bạn muốn xác minh tính toàn vẹn của commit và đảm bảo rằng nó chưa bị sửa đổi.
  + **Git commit --allow-empty:** Tạo một commit trống, không có thay đổi nào được thêm vào. Lệnh này có thể hữu ích khi bạn muốn thêm một commit để đánh dấu một sự kiện quan trọng, nhưng không có thay đổi nào cần được thêm.
  + **Git commit --squash:** Tạo một commit mới và gộp nó vào commit trước đó. Lệnh này cho phép bạn gộp nhiều commit thành một commit lớn hơn, dễ đọc hơn và dễ hiểu hơn.

1. **GIT PULL**

* **Git Pull:** Đẩy các commit mới lên một kho lưu trữ trực tuyến, chẳng hạn như GitHub hoặc Bitbucket. Trong đó, **“pull”** là một hành động để lấy các thay đổi từ một repository khác và đồng bộ hóa chúng với repository hiện tại. Khi bạn sử dụng lệnh pull, Git sẽ tự động tìm kiếm các thay đổi mới nhất trên repository khác và cập nhật chúng vào repository hiện tại. Quá trình này sẽ tự động thực hiện merge giữa các thay đổi đó với các thay đổi đã tồn tại trong repository hiện tại, nếu có.
* **Git pull** thực ra là viết tắt của **“git pull origin master”**. Trong đó:
  + **Origin** là tên của kho chứa từ xa (hay remote repository).
  + **Master** là tên của nhánh trên kho chứa từ xa. Một kho chứa có thể có nhiều nhánh khác nhau.
* Để liệt kê các kho chứa từ xa bạn có thể dùng câu lệnh sau: **git remote -v**
* Để liệt kê các nhánh của các kho chứa: **git branch –a**
* Về bản chất khi chạy câu lệnh **git pull origin master** thực sự là bạn đang sử dụng hai câu lệnh phía sau:



* + Câu lệnh **“git fetch origin master”** sẽ truy vấn thông tin của kho chứa từ xa trên máy chủ remote và sau đó kéo về máy local những thay đổi này. Tiếp đó câu lệnh trên sẽ thực hiện việc so sánh những thay đổi mới kéo về máy local và hiển thị thông tin.
  + Câu lệnh **“git merge orign master”** sẽ gộp những thay đổi mới kéo về (dùng câu lệnh **git fetch** ở trên) từ máy chủ từ xa với nhánh hiện tại trên máy local.

1. **GIT PUSH**

* **Git Push:** Lấy các commit mới từ kho lưu trữ trực tuyến và áp dụng chúng vào nhánh hiện tại. Trong đó, **“push”** được sử dụng để đẩy các thay đổi từ repository hiện tại lên repository khác. Thường thì push được sử dụng khi bạn đã hoàn thành một số thay đổi trên local repository của mình và muốn chia sẻ những thay đổi này với các thành viên khác trong nhóm hoặc lưu trữ trên một remote server. Khi sử dụng lệnh push, Git sẽ đẩy các thay đổi được commit từ local repository lên repository khác, đồng thời cập nhật các thay đổi trên các nhánh của repository hiện tại.
* Sau khi tạo commit bằng câu lệnh **git commit** thì thực tế commit của bạn mới chỉ được lưu lại ở máy local. Để những thay đổi này được đấy lên máy chủ từ xa của Git (hay remote repo) thì bạn cần sử dụng câu lệnh **git push** sau: **$ git push origin master**

1. **GIT BRANCH**

* **Git Branch:** Xem danh sách các nhánh hiện có và tạo ra một nhánh mới.
* Để xem danh sách các nhánh hiện có trong kho lưu trữ, chúng ta có thể chạy lệnh sau: **git branch**. Dòng lệnh này sẽ hiển thị danh sách các nhánh hiện có, với dấu sao (\*) nhằm chỉ ra nhánh đang được sử dụng.
* Để tạo ra một nhánh mới, chúng ta có thể sử dụng lệnh sau: **demo\_project**.
* Sau khi tạo ra nhánh mới, ta có thể chuyển sang nhánh đó bằng câu lệnh: **git checkout**.

1. **GIT MERGE**

* **Git Merge:** Kết hợp các thay đổi từ một nhánh khác vào nhánh hiện tại.
* Câu lệnh **git merge** trong Git được sử dụng để kết hợp các thay đổi từ một nhánh khác vào nhánh hiện tại. Thông thường, bạn sử dụng git merge để hợp nhất các thay đổi từ một nhánh đặc biệt (ví dụ nhánh feature hoặc release) vào nhánh chính (ví dụ nhánh master).
* Để kết hợp các thay đổi từ một nhánh khác vào nhánh hiện tại, ta có thể sử dụng lệnh sau: “**git merge <ten-nhanh-khac>”**. Trong đó **<ten-nhanh-khac>** là tên của nhánh khác mà ta muốn hợp nhất vào nhánh hiện tại. Lệnh trên sẽ kết hợp tất cả các thay đổi từ nhánh khác vào nhánh hiện tại và tạo ra một commit mới.
* Nếu có xung đột giữa các thay đổi trên hai nhánh, Git sẽ hiển thị thông báo và yêu cầu chúng ta giải quyết xung đột trước khi tiếp tục kết hợp.
* **Ví dụ:** Để kết hợp các thay đổi từ nhánh feature-branch vào nhánh master, ta có thể chạy lệnh sau:



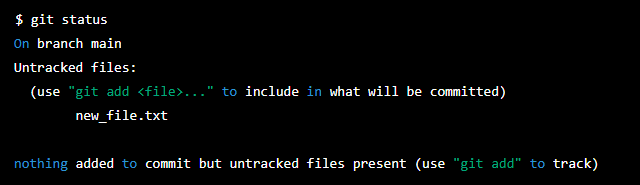
Lệnh trên sẽ chuyển sang nhánh master và kết hợp tất cả các thay đổi từ nhánh feature-branch vào nhánh master.

1. **GIT CHECKOUT**

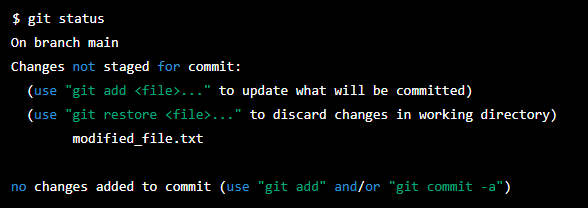
* **Git Checkout:** Chuyển đổi giữa các nhánh hoặc khôi phục các tệp đã bị xóa.
* Để chuyển đổi giữa các nhánh, bạn có thể sử dụng lệnh sau: **git checkout <ten-nhanh>**. Trong đó, **<ten-nhanh>** là tên của nhánh mà bạn muốn chuyển sang. Lệnh trên sẽ chuyển đổi kho lưu trữ sang nhánh đó và cập nhật các tệp trong thư mục làm việc để phản ánh trạng thái của nhánh đó.

1. **GIT STATUS**

* Câu lệnh git status trong Git được sử dụng để xem trạng thái hiện tại của mục theo dõi của Git. Nó cho phép bạn biết được các tệp nào đã thay đổi, đã được thêm vào hoặc đã được xóa khỏi kho lưu trữ của bạn.
* Khi bạn chạy lệnh git status, Git sẽ trả về một danh sách các tệp đã được sửa đổi, đã được thêm vào hoặc đã bị xóa khỏi kho lưu trữ của bạn. Nó cũng sẽ cho bạn biết trạng thái của các tệp này, chẳng hạn như đã được sửa đổi nhưng chưa được đánh dấu để được tạo commit, đã được đánh dấu để được tạo commit hoặc đã được xóa khỏi kho lưu trữ của bạn.
* **Ví dụ:** Nếu bạn thêm một tệp mới vào kho lưu trữ của bạn, khi bạn chạy lệnh git status, bạn sẽ thấy tên của tệp đó được hiển thị với trạng thái untracked. Nghĩa là Git chưa bắt đầu theo dõi tệp này và nó chưa được đánh dấu để được tạo commit.



Nếu bạn sửa đổi một tệp đã được theo dõi và chưa được đánh dấu để được tạo commit, khi bạn chạy lệnh git status, bạn sẽ thấy tên của tệp đó được hiển thị với trạng thái modified. Nghĩa là tệp đã được sửa đổi và đang chờ để được đánh dấu để được tạo commit.



Nếu bạn đã thêm một tệp và đánh dấu nó để được tạo commit, khi bạn chạy lệnh git status, bạn sẽ thấy tên của tệp đó được hiển thị với trạng thái changes to be committed. Nghĩa là tệp đã được đánh dấu để được tạo commit và sẽ được bao gồm trong commit tiếp theo.

