**TÌM HIỂU VỀ GIT VÀ GITHUB**

**I. Giới thiệu**

**1. Source control/version control**

- Source control (hoặc version control) là một hệ thống máy tính giúp ta theo dõi sự thay đổi của mã nguồn, tài liệu và các tệp khác trong dự án. Giúp ta khôi phục lại các phiên bản cũ của tệp, xem các thay đổi đã được thực hiện và giải quyết xung đột khi nhiều người cùng làm việc trên cùng một tệp.

- Có nhiều hệ thống source control khác nhau được sử dụng trong công việc phát triển phần mềm, bao gồm Git, Mercurial, Subversion và CVS. Mỗi hệ thống có ưu và nhược điểm riêng, và ta có thể tùy ý lựa chọn sử dụng bất cứ hệ thống nào phù hợp với nhu cầu của dự án.

**2. Git**

- Git là 1 hệ thống kiểm soat version nổi tiếng. Được tạo ra bởi Linus Torvalds vào năm 2005, và được duy trì bởi Junio Hamano đến tận bây giờ.

- Git là một hệ thống quản lý phiên bản phần mềm dựa trên mô hình phân tán. Là một trong những công cụ quản lý phiên bản phần mềm phổ biến nhất hiện nay và được sử dụng rộng rãi trong các dự án phần mềm mã nguồn mở và các dự án phần mềm đóng gói.. Git cho phép lưu trữ lịch sử thay đổi và xác định các thay đổi nào được thực hiện trên mã nguồn. Git cũng cho phép nhiều người làm việc trên cùng một dự án cùng lúc, và có thể giúp ta quản lý các thay đổi và hợp nhất các nhánh khác nhau của dự án.

**3. Github**

- GitHub là một dịch vụ lưu trữ và quản lý mã nguồn dựa trên Git. Cho phép người dùng lưu trữ và quản lý các dự án phần mềm đồng thời cung cấp nhiều tính năng khác như tìm bug, các công cụ phát triển trực tuyến, và các công cụ hỗ trợ cho việc làm việc nhóm. GitHub cũng là một nền tảng phù hợp với các dự án mã nguồn mở, cung cấp các tính năng như giới thiệu dự án, theo dõi và quản lý các vấn đề, và hợp tác trên mã nguồn cùng nhau. GitHub được sử dụng rộng rãi trong cộng đồng phát triển phần mềm và được coi là một trong những nền tảng quản lý mã nguồn phổ biến nhất hiện nay.

**II. Các thuật ngữ quan trọng và một số lệnh cơ bản**

**1. Các thuật ngữ quan trọng**

- Repository: chứa tất cả các tệp dự án

- Branch: hay nhánh, là đại diện cho các phiên bản cụ thể của 1 kho lưu trữ tách ra từ project chính. Branch cho phép ta theo dõi các thay đổi thực hiện với kho lưu trữ và có thể hoàn trả về phiên bản cũ hơn.

- Commit: đại diện cho 1 thời điểm cụ thể trong lịch sử dự án.

- Checkout: để chuyển giữa các branch hoặc để trở về branch chính

- Fetch: là tìm nạp các bản sao và tải xuống tất cả các tệp branch vào máy tính cá nhân. Sử dụng để lưu các thay đổi mới nhất vào kho lưu trữ cá nhân.

- Fork: là 1 bản sao của 1 kho lưu trữ repository

- Head: là các commit ở đầu của 1 branch. Đại diện cho commit mới nhất mà ta từng làm việc

- Master: là nhánh chính của tất cả repository

- Merge: là thêm các thay đổi từ nhành này sang nhánh khác

- Origin: là phiên bản mặc định của repository

- Pull: thêm các thay đổi vào nhánh chính

- Push: cập nhật các nhánh từ xa với những thay đổi mới nhất mà ta đã commit

- Rebase: phân tích, di chuyển một commit nào đó

- Remote: là một bản sao của nhánh chính

- Upstream: là thuật ngữ để chỉ nơi có thể push các thay đổi trong nhóm chính.

**2. Một số lệnh cơ bản**

- git init

+ Là một lệnh trong Git dùng để khởi tạo một repository mới. Khi ta sử dụng lệnh này trong một thư mục đã tồn tại, khi đó sẽ tạo ra một thư mục có tên ".git", và khởi tạo các cài đặt cơ bản cho repository.

+ Ví dụ:

$ cd /path/to/myproject

$ git init

> Sau khi sử dụng lệnh "git init", thư mục "myproject" sẽ trở thành một repository Git.

- git clone

+ là một lệnh trong Git dùng để tải về một repository từ một kho lưu trữ Git có địa chỉ URL.

+ Ví dụ:

$ git clone https://github.com/tên-tài-khoản/tên-repository.git

> Sau khi lệnh được thực thi, Git sẽ tạo ra một thư mục mới với tên là tên-repository và sao chép toàn bộ nội dung của repository vào trong đó. Ta có thể chỉ định một tên khác cho thư mục sao chép bằng cách thêm tên thư mục vào cuối lệnh git clone

$ git clone https://github.com/tên-tài-khoản/tên-repository.git tên-thư-mục-mới

> Trong trường hợp này, ta clone một repository từ kho lưu trữ khác bằng cách thay đổi địa chỉ URL trong lệnh git clone.

- git pull

+ Là một lệnh trong Git dùng để tải về các thay đổi mới từ một kho lưu trữ Git và sao chép chúng vào nhánh hiện tại. Lệnh này thường được sử dụng để cập nhật nhánh của ta với nhánh chính (thường là master) của repository.

+ Ví dụ:

$ git pull <tên nhánh>

> Trong trường hợp bạn muốn cập nhật nhánh hiện tại với nhánh chính của repository, ta có thể không cần chỉ định tên nhánh:

$ git pull

- git add

+ Là một lệnh trong Git dùng để thêm tập tin hoặc thư mục vào Git index. Khi thêm một tập tin hoặc thư mục vào index, nó sẽ được ghi lại trạng thái và đưa vào tính toán khi tạo một commit.

+ Ví dụ:

$ git add <tập tin hoặc thư mục>

> Để thêm tập tin file.txt vào index, ta có thể sử dụng lệnh sau:

$ git add file.txt

- git commit

+ Cho phép lưu trữ một bản sao của mã nguồn vào lịch sử phiên bản của Git. Khi bạn thực hiện một commit, bạn sẽ đóng gói các thay đổi trong mã nguồn của bạn vào một gói, và ghi lại gói đó vào lịch sử phiên bản của Git.

+ Để thực hiện một commit trong Git, ta cần thêm các tập tin muốn commit vào thể hộp chọn bằng lệnh "git add" và sau đó sử dụng lệnh "git commit" để thực hiện commit. Ta cũng có thể thêm một tin nhắn để mô tả những thay đổi đã thực hiện trong commit này bằng cách sử dụng tham số "-m" với lệnh "git commit".

+ Ví dụ:

$ git add file1.txt file2.txt

$ git commit -m "Thêm tập tin file1.txt và file2.txt"

> Trong ví dụ trên, chúng ta đang thêm hai tập tin mới vào thể hộp chọn và thực hiện một commit với tin nhắn "Thêm tập tin file1.txt và file2.txt" để mô tả những thay đổi đã thực hiện.

- git push

+ Cho phép đẩy các commit mới những thay đổi liên quan đến chúng lên kho lưu trữ trên máy chủ.

+ Để thực hiện một push trong Git, cần sử dụng lệnh "git push" với tên của kho lưu trữ và tên nhánh mà ta muốn đẩy lên.

+ Ví dụ, để đẩy một nhánh tên "master" lên một kho lưu trữ tên "origin", bạn có thể sử dụng lệnh sau:

$ git push origin master

> Chúng ta đang đẩy nhánh "master" lên kho lưu trữ "origin". Nếu kho lưu trữ này được cấu hình để yêu cầu xác thực có thể cần phải cung cấp tên người dùng và mật khẩu để thực hiện push thành công.

- git log

+ Cho phép xem lịch sử phiên bản của mã nguồn trong một kho lưu trữ Git. Khi sử dụng lệnh, một danh sách các commit trong lịch sử phiên bản sẽ được hiển thị bao gồm thông tin về người thực hiện commit, ngày thực hiện commit và tin nhắn mô tả commit. Ta cũng có thể sử dụng các tham số khác để điều chỉnh cách hiển thị của lịch sử phiên bản.

+ Ví dụ, có thể sử dụng tham số "--pretty=format:" để điều chỉnh định dạng hiển thị của lịch sử phiên bản, hoặc sử dụng tham số "--max-count" để giới hạn số lượng commit hiển thị:

$ git log --pretty=format:"%h - %an, %ar : %s" --max-count=5

> Trong ví dụ trên, chúng ta sử dụng lệnh "git log" với tham số "--pretty=format:" để điều chỉnh định dạng hiển thị của lịch sử phiên bản, và tham số "--max-count

**III. Branch và merge**

- git branch

+ Cho phép bạn tạo ra, quản lý và chuyển giữa các nhánh trong một kho lưu trữ Git.

+ Một nhánh trong Git là một con trỏ đến một commit cụ thể trong lịch sử phiên bản của mã nguồn. Mỗi khi thực hiện một commit, nó sẽ tạo ra một commit mới và con trỏ của nhánh sẽ được di chuyển đến commit mới đó. Nhánh có thể được sử dụng để phát triển mã nguồn một cách độc lập, và sau đó hợp nhất các thay đổi vào nhánh chính (thường là nhánh "master") khi chúng đã được kiểm tra và xác nhận là an toàn. Để tạo một nhánh mới trong Git, có thể sử dụng lệnh "git branch" với tên của nhánh mới.

+ Ví dụ, để tạo một nhánh mới tên "feature", ta sử dụng lệnh sau:

$ git branch feature

> Để liệt kê tất cả các nhánh trong một kho lưu trữ Git, ta có thể sử dụng lệnh "git branch" mà không có tham số nào.

- git checkout

+ Cho phép chuyển giữa các nhánh và các commit cụ thể trong một kho lưu trữ Git. Để chuyển giữa các nhánh trong Git, có thể sử dụng lệnh "git checkout" với tên của nhánh mà muốn chuyển đến.

+ Ví dụ, để chuyển đến nhánh "feature", ta sử dụng lệnh sau:

$ git checkout feature

- git merge

+ Dùng để hợp nhất những thay đổi trong nhiều nhánh khác nhau vào một nhánh chính. Điều này có thể giúp dễ dàng hơn trong việc quản lý mã nguồn của dự án và đảm bảo rằng tất cả các thay đổi được hợp nhất vào một nhánh duy nhất. Để thực hiện Merge trong Git, ta cần phải chuyển sang nhánh chính cần hợp nhất và sử dụng lệnh "git merge" và tên của nhánh cần hợp nhất.

+ Ví dụ, để hợp nhất nhánh "feature" vào nhánh "master", ta có thể sử dụng lệnh sau:

$ git checkout master

$ git merge feature

> Nếu có xung đột trong quá trình hợp nhất, cần phải xác định xem sửa đổi nào sẽ được giữ lại và xóa bỏ những sửa đổi khác. Sau khi giải quyết xung đột cần phải thực hiện commit để lưu lại các thay đổi hợp nhất.

- git rebase

+ Là một tính năng của Git dùng để chuyển các commit trên một nhánh sang trên một nhánh khác. Khi sử dụng Rebase, các commit sẽ được chuyển từ nhánh đích sang trên nhánh nguồn và được đặt lên đầu của nhánh nguồn.

+ Việc sử dụng Rebase có thể giúp giữ cho lịch sử commit trên nhánh gọn gàng hơn vì nó loại bỏ các commit không cần thiết và chỉ giữ lại những commit quan trọng. Một điều cần lưu ý khi sử dụng Rebase là nó sẽ thay đổi các commit trên nhánh nguồn, do đó nên chỉ sử dụng Rebase trên nhánh riêng và không nên sử dụng trên nhánh chung với người khác.

+ Ví dụ, nếu bạn muốn chuyển tất cả các commit trên nhánh "feature" sang trên nhánh "master", bạn có thể sử dụng lệnh sau:

$ git checkout master

$ git rebase feature

- git cherry-pick

+ Là một tính năng của Git cho phép ta chọn một commit hoặc nhiều commit và áp dụng chúng trên một nhánh khác. Điều này có thể giúp dễ dàng hơn trong việc tách riêng các thay đổi trong dự án và áp dụng chúng tới những nhánh khác. Để sử dụng Cherry-pick, ta cần phải chuyển sang nhánh cần áp dụng commit và sử dụng lệnh "git cherry-pick" và id của commit cần chọn.

+ Nếu có xung đột trong quá trình Cherry-pick, ta sẽ phải xác định xem sửa đổi nào sẽ được giữ lại và xóa bỏ những sửa đổi khác. Sau khi giải quyết xung đột, ta cần phải thực hiện commit để lưu lại các thay đổi được chọn.

+ Ví dụ, để chọn commit có id là "abc123" và áp dụng nó trên nhánh "master", ta có thể sử dụng lệnh sau:

$ git checkout master

$ git cherry-pick abc123