**BÁO CÁO**

**MÔN HỌC: KỸ NĂNG NGHỀ NGHIỆP**

**CHỦ ĐỀ: TÌM HIỂU GIT VÀ GITHUB**

GVHD: Thái Huy Tân

Ngày báo cáo: 13/9/2024

1. **Thông tin chung**

Lớp: SS004.P14

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | MSSV | Email |
| 1 | Phạm Lê Đăng Kha | 23520669 | [23520669@gm.uit.edu.vn](mailto:23520669@gm.uit.edu.vn) |
| 2 | Phan Cảnh Đăng Huân | 23520552 | [23520552@gm.uit.edu.vn](mailto:23520552@gm.uit.edu.vn) |
| 3 | Võ Duy Hiếu | 23520498 | [23520498@gm.uit.edu.vn](mailto:23520498@gm.uit.edu.vn) |
| 4 | Hồng Huy Hoàng | 23520517 | [23520517@gm.uit.edu.vn](mailto:23520517@gm.uit.edu.vn) |

DANH SÁCH HÌNH ẢNH





Nguồn: https://education.github.com/git-cheat-sheet-education.pdf

MỤC LỤC

CHƯƠNG 1: Giới thiệu về Git và GitHub 3

CHƯƠNG 2: Các lệnh cơ bản về Git 5

CHƯƠNG 3: Tổng kết 11

Tài liệu tham khảo

**Chương 1: Giới thiệu về Git và Github**

Trước khi giới thiệu về Git cũng như Github, ta hãy đi tìm hiểu lí do Git ra đời. Trước đây, khi mà chúng ta làm việc chung với nhau trên 1 dự án, thì mỗi người trong nhóm sẽ được chia phần và tự viết code của chính mình sau đó gửi cho nhau để xem và sửa lại, đến 1 lúc nào đó thì các thành viên trong nhóm sẽ ngồi lại để ráp code vào nhau để đưa vào dự án. Tuy nhiên, vào một ngày thì có 1 thành viên đã viết code sai vào phần mềm khiến cho sản phẩm bị lỗi, thế nhưng nhóm lại không thể khôi phục đoạn code bị lỗi về trước đó do phần mềm quá lớn để sao lưu. Và thế là các thành viên trong nhóm lại phải viết code lại từ khi sản phẩm bị lỗi rồi lại phải ngồi lại để ráp code với nhau. Điều này sẽ tốn rất nhiều thời gian vì mỗi lần ráp code là mỗi lần phải sửa đổi để chương trình có thể chạy được (vì mỗi người một ý khác nhau), kế đến là mỗi khi có một thành viên viết code khiến phần mềm bị lỗi, chúng ta sẽ không biết lỗi đó là do ai, lỗi đó có từ lúc nào, và cũng rất khó khôi phục khi mà lỗi xảy ra (do không hề có sao lưu trước đấy). Vì vậy, nên Git đã xuất hiện và đã giải quyết được các vấn đề này một cách nhanh gọn và dễ dàng.

1. **Vậy Git là gì?**

* Git là một hệ thống quản lý/kiểm soát phiên bản phân tán (Distributed Version Control System – DVCS).
* Trước tiên ta sẽ tìm hiểu về hệ thống quản lý phiên bản. Nó là 1 hệ thống giúp duy trì các phiên bản khác nhau của một dự án (hay nó quản lí các thay đổi đối với các tệp). Vì vậy, ta hoàn toàn có thể xem lại lịch sử thay đổi với mã nguồn (ví dụ như: cái gì đã được thêm/bớt/thay đổi ở khúc nào, ai đã thực hiện những thay đổi đó, và những thay đổi đó đã được thực hiện vào lúc nào, lí do thay đổi, …). Và hơn thế nữa, mỗi khi có lỗi, ta hoàn toàn có thể biết được điều gì đã gây ra lỗi để có thể khắc phục một cách nhanh hơn hoặc thậm chí ta còn có thể quay lại bất kì phiên bản nào ta muốn.
* Tiếp đến là hệ thống quản lí phiên bản phân tán, nghĩa là mọi cộng tác viên đều có kho lưu trữ cục bộ (local repository) trên máy cục bộ của mỉnh, khác với trung tâm (central) nơi mà các thành viên phải có kết nối Internet để cập nhật công việc của họ lên kho lưu trữ trung tâm chính (main central repository). Các thành viên có thể nhân bản (clone) từ main central repo về máy cục bộ để làm việc, cũng như giúp việc đồng bộ source code dễ dàng hơn.
* **Lợi ích của Git:**
* Giúp các thành viên có thể cộng tác dễ dàng mà không cần phải gửi file qua lại, tốn kém thời gian, đôi khi còn mất file, không bảo mật.
* Giúp việc quản lí source code dễ dàng.
* Giúp ta khôi phục lại phiên bản trước đó nếu muốn (ta cũng không còn phải copy paste để lưu lại mã nguồn trước khi thay đổi)
* Giúp giải quyết các xung đột khi nhiều thành viên làm việc với nhau.

1. **Thế còn Github là gì?**

* Đa số ai trong chúng ta cũng đã nghe đến Git và Github, thậm chí còn đã sử dụng Github để tham khảo tài liệu, nhưng đa phần hay nhầm lẫn Git và Github là một.
* Trong khi Git là một phần mềm, được cài đặt cục bộ trên hệ thống, và nó dùng chủ yếu để quản lí lịch sử của mã nguồn (source code) và được duy trì bới Linux. Thì Github lại là một dịch vụ, nền tảng trên web để lưu trữ cho các Git repo và được duy trì bởi Microsoft. Github cho phép nhiều cộng tác viên tham gia vào dự án, xem xét các thay đổi cũng như đóng góp vào mã nguồn.

**Chương 2: Các tập lệnh cơ bản**

1. **SETUP & INIT**



1. Git init:

Mô tả: Khởi tạo một thư mục hiện có thành một repository Git.

Tác dụng: Khi muốn bắt đầu theo dõi lịch sử của các tệp trong thư mục, lệnh này sẽ tạo một repository Git mới trong thư mục đó. Nó sẽ tạo một thư mục ẩn .git để lưu trữ toàn bộ lịch sử của repository.

1. Git clone [url]:

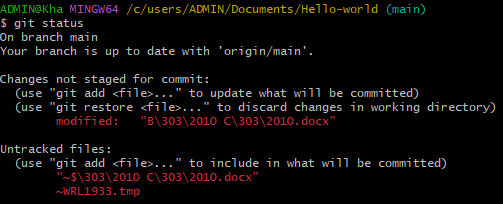
Mô tả: sao chép toàn bộ repository thông qua URL.

Tác dụng: Tải xuống một bản sao của repository Git từ URL về máy, bao gồm toàn bộ lịch sử commit và các nhánh (branch).

1. **STAGE & SNAPSHOT**



1. Git status:



Mô tả: Hiển thị trạng thái của các tệp trong thư mục đang làm việc, cho biết tệp nào đã được sửa đổi, thêm mới hoặc xóa, và những tệp nào trong khu vực staging (để chuẩn bị cho lần commit kế tiếp)

Tác dụng: Kiểm tra trạng thái của tệp trước khi commit, để chắc chắn những thay đổi đều được đưa vào đúng cách.

1. Git add [file]:

Mô tả: Thêm tệp được chỉ định vào khu vực staging.

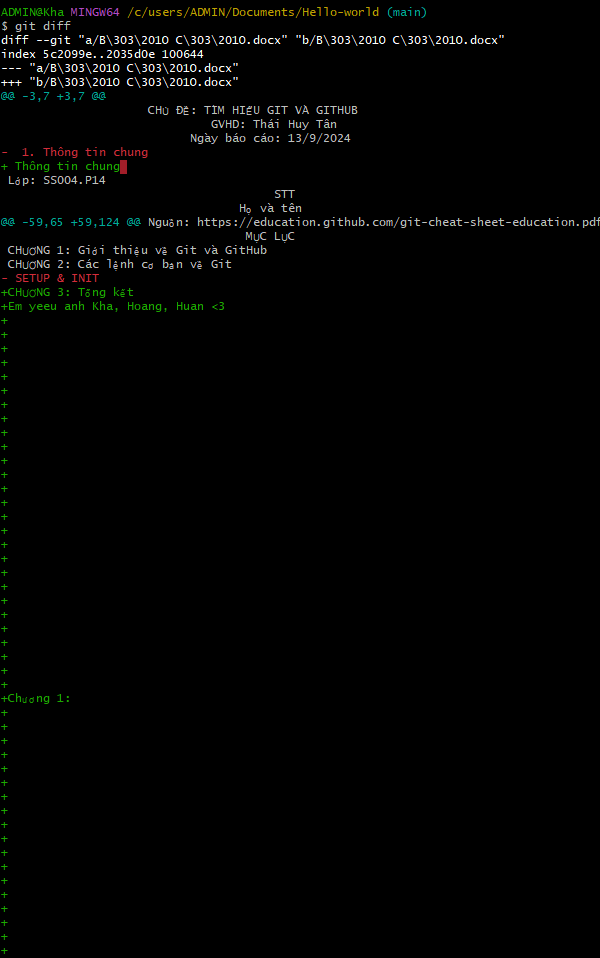
Tác dụng : Đưa các thay đổi từ thư mục đang làm việc vào khu vực staging. Nếu muốn commit các thay đổi cho một tệp phải sử dụng lệnh này trước.

1. Git reset [file]:

Mô tả : Gỡ bỏ tệp ra khỏi khu vực staging nhưng vẫn giữ lại các thay đổi trong thư mục làm việc.

Tác dụng : Nếu thêm nhầm tệp vào khu vực staging, lệnh này giúp gỡ bỏ nó khỏi danh sách mà không xóa nội dung tệp.

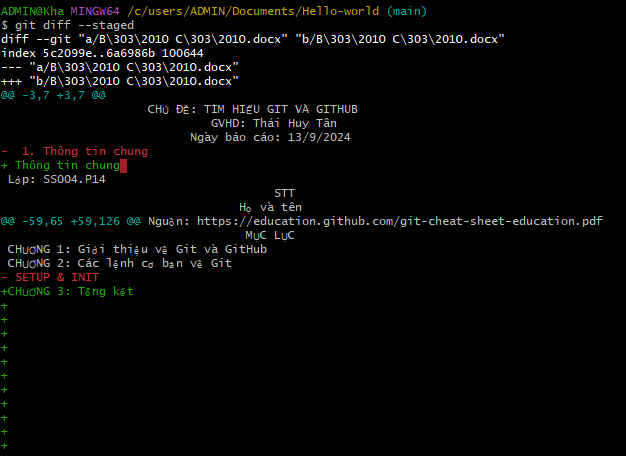
1. Git diff:



Mô tả : Hiện thị các thay đổi giữa nội dung trong thư mục làm việc và tệp trong khu vực staging.

Tác dụng : Xem sự khác biệt giữa tệp đã được chỉnh sửa và tệp chưa được đưa vào staging.

1. Git diff --staged:



Mô tả : Xem sự khác biệt giữa tệp đã ở khu vực staging và phiên bản cuối cùng đã commit.

Tác dụng : Xem các thay đổi đã sẵn sàng để commit.

1. Git commit -m [descriptive message]:

Mô tả : Tạo một bản commit với thông điệp miêu tả các thay đổi.

Tác dụng : Lưu các thay đổi đã ở trong khu vực staging vào lịch sử của Git kèm theo thông điệp giải thích.

1. **BRANCH & MERGE**



1. Git branch:

Mô tả : Liệt kê các nhánh trong repository. Một dấu \* sẽ xuất hiện bên cạnh nhánh hiện đang hoạt động.

Tác dụng : Cho biết tất cả các nhánh hiện có và xác định nhánh nào đang được sử dụng.

1. Git branch [branch-name]:

Mô tả: Tạo một nhánh mới tại commit hiện tại với tên được chỉ định.

Tác dụng: Tạo một nhánh riêng biệt để làm việc độc lập mà không ảnh hưởng đến nhánh chính.

1. Git checkout:

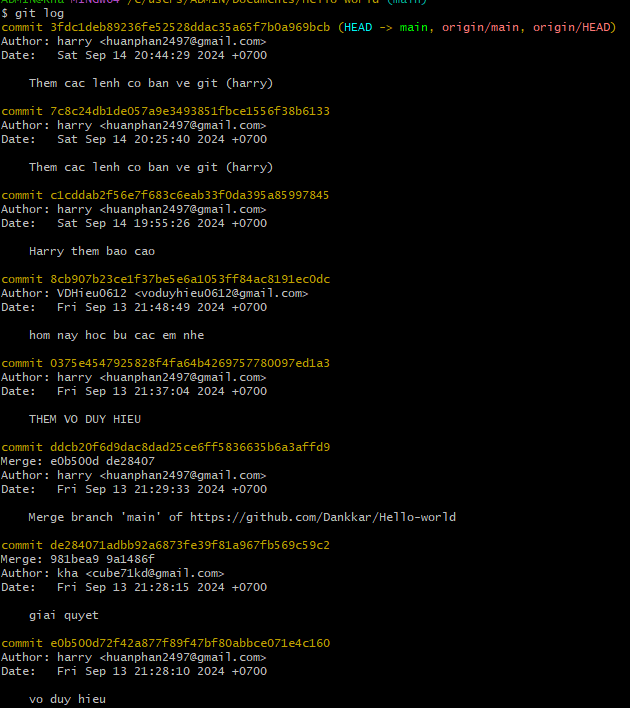
Mô tả: Chuyển sang một nhánh khác và kiểm tra nội dung của nó vào thư mục làm việc.

Tác dụng: Dùng để chuyển giữa các nhánh và làm việc trên các nhánh khác nhau mà không ảnh hưởng đến nhánh hiện tại.

1. Git merge [branch]:

Mô tả: Gộp lịch sử của nhánh được chỉ định vào nhánh hiện tại.

Tác dụng: Khi đã hoàn thành công việc trên một nhánh riêng biệt, lệnh này sẽ hợp nhất cac thay đổi của nhánh đó vào nhánh hiện tại.

1. Git log:

Mô tả: Hiển thị toàn bộ các commit trong lịch sử của nhánh hiện tại.

Tác dụng: Xem lại lịch sử các lần commit trên nhánh đang làm việc.