**Phần 10 Git Commit**

**Git Commit là gì?**

Git commit giống như một snapshot của dự án của bạn tại một thời điểm cụ thể. Khi bạn thực hiện commit, Git sẽ lưu trạng thái của dự án, bao gồm tất cả các tệp được theo dõi và gán một mã định danh duy nhất (một hàm băm commit) cho snapshot này. Mỗi commit được xác định duy nhất bằng hàm băm SHA-1, đảm bảo tính toàn vẹn và duy nhất của nó. Điều này cho phép bạn:

* **Theo dõi thay đổi:** Lưu lại những thay đổi đã thực hiện, ai thực hiện và khi nào.
* **Hợp tác:** Làm việc liền mạch với các nhà phát triển khác mà không làm ảnh hưởng đến công việc của nhau.
* **Hoàn nguyên thay đổi:** Quay lại phiên bản trước nếu có lỗi xảy ra.

**Tại sao Git Commit lại quan trọng?**

1. **Theo dõi lịch sử:** Git commit tạo ra lịch sử theo trình tự thời gian của dự án, giúp bạn dễ dàng xem những thay đổi nào đã được thực hiện và khi nào. Điều này vô cùng có giá trị để hiểu được sự phát triển của mã của bạn.
2. **Hợp tác:** Trong môi trường làm việc nhóm, người cam kết giúp phối hợp công việc bằng cách cung cấp hồ sơ rõ ràng về những gì mỗi thành viên trong nhóm đã làm.
3. **Khả năng đảo ngược:** Nếu một thay đổi gây ra lỗi hoặc làm hỏng thứ gì đó, bạn có thể quay lại lần xác nhận trước đó khi mọi thứ vẫn hoạt động bình thường.

**Cách thực hiện một cam kết Git**

Để thực hiện điều này, git commit được sử dụng và lệnh là:

*git commit -m “thông điệp cam kết”*

Lệnh git commit được sử dụng để di chuyển các tệp từ vùng dàn dựng đến kho lưu trữ cục bộ của bạn. Lệnh này được chạy sau git add và có thể được xem như một điểm kiểm tra. Sau khi thực hiện git commit, vùng dàn dựng của bạn sẽ trống.

**Làm việc với Git Commit**

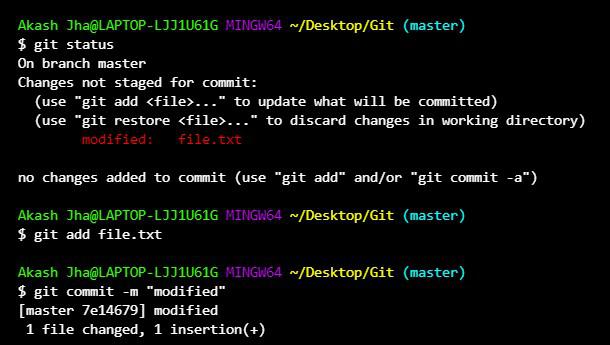
Giả sử, bạn đang ở nhánh master và bạn đã thực hiện một commit, giờ đây git sẽ tạo một block, tức là một node, của commit mà bạn đã thực hiện. Trong git, chúng ta có một con trỏ có tên là HEAD (Trong trường hợp của chúng ta, đây là tham chiếu đến commit trong nhánh hiện tại, tức là nhánh master). Khi bạn commit một cái gì đó, HEAD sẽ được trỏ đến commit mới và một khóa băm được gán cho commit mới đó. Nếu bạn thực hiện một commit khác, HEAD sẽ lại di chuyển và trỏ đến commit có một khóa băm khác. Có thể coi đây là một danh sách liên kết đơn từ phải sang trái, bất cứ khi nào một commit mới xảy ra, một node mới sẽ được tạo và trỏ đến HEAD, sau đó HEAD được cập nhật và đi qua commit mới, tức là commit mới nhất.

**Cờ trong Git Commit**

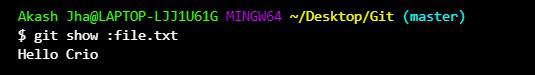
Có một số cờ nhất định trong git commit như -a, -m, -am.

* ***Git commit -m “commit message”*** : Lệnh tạo một commit kèm theo thông điệp commit được viết trong dấu ngoặc kép.
* ***Git commit -a*** : Lệnh này chỉ bao gồm các sửa đổi đối với các tệp đã được thêm bằng git add tại bất kỳ thời điểm nào, tức là tất cả các tệp được theo dõi.
* ***Git commit -am “commit message”*** : Đây là sự kết hợp của cả lệnh -m và -a.

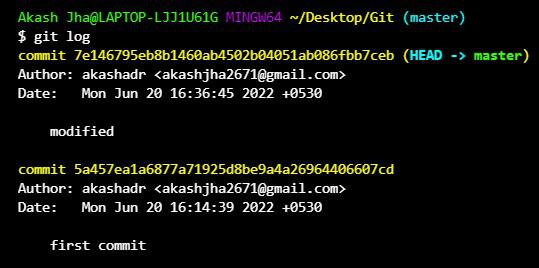
Nếu bây giờ bạn thấy rằng bạn vô tình bỏ sót một cái gì đó trong lần commit cuối cùng của mình, có thể là một tệp hoặc một thay đổi bổ sung đối với tệp mà bạn vừa commit, thì bạn cũng có thể sửa lỗi đó. Bây giờ chúng ta hãy thực hiện một số sửa đổi trong "file.txt", tức là tôi viết "Hello Crio" trong đó và bây giờ chúng ta phải lặp lại quy trình thêm và commit.



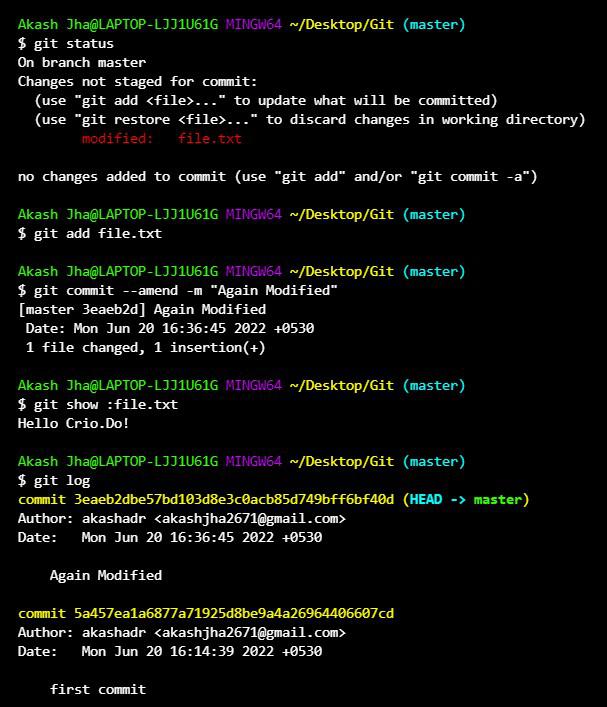
Sau đó, tệp “file.txt” đã được sửa đổi và bạn có thể kiểm tra bằng lệnh git git show.



Nếu bạn muốn xem lịch sử mọi thứ xảy ra với một kho lưu trữ thì bạn có thể sử dụng lệnh git log. Git log là bản ghi đang chạy của các lần commit.



Như bạn có thể thấy ở trên có hai lịch sử cam kết, giả sử bạn đã phạm lỗi trong "file.txt" nên bạn phải lặp lại quy trình một lần nữa và do đó có thêm một lịch sử cam kết được tạo. Đối với số lượng lịch sử nhỏ, điều này ổn, nhưng nếu bạn mắc lỗi nhiều lần và cam kết chúng thì có một số lượng lớn lịch sử được tạo ra và điều này hơi khó hiểu đối với các nhà phát triển khác khi làm việc với các loại dự án phức tạp. Vì vậy, để xử lý điều đó, lệnh ***git commit –-amend*** được sử dụng. ***git commit –amend -m “commit-msg”***  trong đó  ***-m***  là cờ được sử dụng cho thông báo cam kết là lệnh được sử dụng để không tạo lịch sử cam kết khác và chỉ ghi đè cam kết trước đó.



Ở đây trong hình ảnh trên, bạn có thể thấy không có lịch sử thứ ba nào được tạo. Điều ở đây là commit trước đó đã bị ghi đè và do đó lịch sử commit vẫn giữ nguyên.

**Hoàn nguyên các thay đổi đã cam kết**

Nếu bạn muốn hoàn nguyên một cam kết đã thực hiện, bạn phải sử dụng:

git reset --hard HEAD^

Lệnh này sẽ khôi phục lại commit mới nhất, tức là commit đầu tiên.