**BÁO CÁO TÌM HIỂU VỀ GIT**

1. **Giới thiệu về Git**

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) rất được ưa chuộng, được sử dụng để theo dõi và quản lý các sự thay đổi trong mã nguồn trong suốt quá trình phát triển phần mềm. Được tạo ra bởi Linus Torvalds vào năm 2005, Git đã nhanh chóng trở thành một trong những công cụ quản lý phiên bản phổ biến và quan trọng nhất trên toàn cầu.

Mục đích chính của Git là giúp đội ngũ phát triển làm việc cùng nhau trên cùng một dự án mà không gặp khó khăn về việc ghi đè lên công việc của nhau. Nó cho phép mọi người làm việc trên các phiên bản riêng của mã nguồn và sau đó hợp nhất các thay đổi này thành phiên bản chính thức, được gọi là "nhánh chính" (thường được gọi là "master" hoặc "main").

1. **Các Tính Năng Chính và Ưu Điểm của Git**

* **Version Control:** Git giúp theo dõi thay đổi mã nguồn theo thời gian, cho phép quay trở lại phiên bản trước và hiểu lịch sử sửa đổi.
* **Collaboration:** Cho phép nhiều nhà phát triển cùng làm việc trên dự án, tạo nhánh riêng để xử lý tính năng hoặc lỗi, sau đó hợp nhất vào mã nguồn chính để phát triển đồng thời.
* **Branching and Experimentation**: Hệ thống nhánh (branch) của Git cho phép tạo các branch độc lập cho tính năng mới, bảo vệ mã nguồn chính. Nếu thành công, nhánh này có thể hợp nhất (merge) vào nhánh chính.
* **Conflict Resolution**: Git giúp giải quyết xung đột khi nhiều nhà phát triển cùng sửa mã nguồn. Công cụ của Git đảm bảo tích hợp thay đổi không xảy ra xung đột.
* **Traceability**: Git lưu lịch sử chi tiết thay đổi code, bao gồm tác giả, thời gian và lý do. Khả năng này hữu ích để hiểu lịch sử thay đổi code và khắc phục lỗi.
* **Code Reviews**: Nền tảng Git như GitHub và GitLab,... hỗ trợ code review. Lập trình viên có thể tạo các pull request , sau đó người khác review code, cải thiện chất lượng code và chia sẻ kiến thức.
* **Rollbacks and Bug Fixes**: Khi gặp lỗi hoặc thay đổi không mong muốn, Git cho phép quay trở lại phiên bản trước ổn định, giảm nguy cơ lỗi và tiết kiệm thời gian.
* **Backup and Redundancy**: Mỗi nhà phát triển có bản sao đầy đủ của mã nguồn, giảm rủi ro mất dữ liệu. Các bản sao được lưu ở git server.
* **Offline Work**: Git có tính phân tán, cho phép nhà phát triển làm việc ngoại tuyến, hỗ trợ nhóm làm việc từ xa hoặc phân tán.
* **Open Source and Collaboration**: Git là mã nguồn mở và được sử dụng rộng rãi. Nhiều dự án mã nguồn mở được lưu trữ trên các nền tảng như GitHub, giúp các nhà phát triển đóng góp và hợp tác trên các dự án khác nhau một cách dễ dàng.
* **Scalability:** Git có thể xử lý các dự án với các kích thước khác nhau, từ các dự án cá nhân nhỏ đến các dự án mã nguồn cấp doanh nghiệp lớn.
* **Custom Workflows**: Có thể tùy chỉnh workflow linh hoạt dựa trên git, tạo ra các branch, các release khác nhau theo nhu cầu cụ thể.

1. **Tìm hiểu Git cơ bản**
2. **Repository**

Repository là nơi chứa mã nguồn và thông tin liên quan đến sửa đổi và lịch sử dự án. Nó lưu trữ tất cả các tệp tin và thư mục liên quan đến dự án cùng với lịch sử thay đổi của chúng. Mỗi lần có sự thay đổi trong mã nguồn, repository ghi lại thông tin về thay đổi đó, bao gồm tác giả, thời gian và nội dung thay đổi. Điều này giúp theo dõi quá trình phát triển, phục hồi phiên bản trước và quản lý việc tích hợp mã từ nhiều nguồn khác nhau.

***Repository có hai dạng:***

* Remote Repository: Đặt trên máy chủ Git, có khả năng chia sẻ với nhiều người.
* Local Repository: Đặt trên máy cá nhân, dành riêng cho người dùng. có thể sao chép (clone) Remote Repository về tạo thành Local Repository. Khi có sửa đổi ở Local Repository, có thể đẩy (push) mã lên Remote Repository.

1. **Branch**

Branch là một phần của repository, tương đương với khu vực làm việc độc lập. Khi tạo repository, một nhánh chính (master/main) sẽ được tạo sẵn. Nhánh này có thể chia thành nhiều nhánh con. Những thay đổi trên nhánh con không ảnh hưởng đến nhánh chính, cho phép làm nhiều sửa đổi trên cùng một kho chứa. Ta cũng có thể kết hợp (merge) các nhánh lại với nhau.

***Branch chia thành 2 loại, tương tự repository***:

* Remote Branch: Lưu trên git server, có thể chia sẻ.
* Local branch: Lưu trên máy cá nhân, có thể liên kết với remote branch hoặc không.

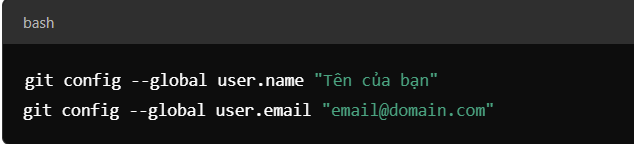
1. **Cơ chế hoạt động cơ bản của Git**

Cơ chế hoạt động của Git dựa trên hệ thống quản lý phiên bản phân tán. Khi làm việc với Git, quá trình hoạt động diễn ra như sau:

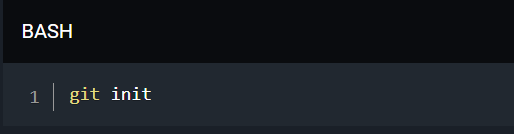
1. **Khởi tạo Repository:**  bắt đầu bằng cách khởi tạo một kho chứa (repository). Kho chứa này chứa mã nguồn và lịch sử thay đổi của dự án.
2. **Tạo Commits:** Mỗi lần thay đổi mã nguồn, tạo một commit. Mỗi commit lưu trữ thay đổi cụ thể cùng với thông tin về tác giả, thời gian và mã băm duy nhất.
3. **Tạo và Quản lý Nhánh:**  có thể tạo nhiều nhánh để phát triển tính năng, sửa lỗi hoặc thử nghiệm mà không ảnh hưởng đến nhánh chính. Nhánh giúp làm việc song song và tổ chức công việc.
4. **Merge và Rebase:** Khi hoàn thành công việc trên một nhánh, có thể kết hợp (merge) hoặc sắp xếp lại (rebase) thay đổi từ một nhánh vào nhánh khác.
5. **Kết nối với Remote:** Có thể kết nối kho chứa của với kho chứa từ xa như GitHub hoặc GitLab. Điều này giúp chia sẻ và làm việc cùng nhau trên dự án.
6. **Push và Pull:** Bằng cách sử dụng lệnh push, có thể đẩy các commit từ kho chứa local lên kho chứa từ xa. Lệnh pull được sử dụng để đồng bộ hóa thay đổi từ kho chứa từ xa về kho chứa local.
7. **Giải Quyết Xung Đột:** Khi có sự thay đổi xung đột, Git cung cấp các công cụ để giải quyết và chọn cách hợp nhất các thay đổi một cách đúng đắn.
8. **Lịch Sử và Truy vết:** Git theo dõi lịch sử toàn bộ quá trình thay đổi, cho phép theo dõi sự phát triển của dự án và xem lại các thay đổi cụ thể.

*Cơ chế này cho phép các nhà phát triển làm việc hiệu quả, theo dõi quá trình phát triển và quản lý mã nguồn một cách linh hoạt*

1. **Các lệnh Git cơ bản**
   1. **Cấu hình Git ban đầu**

****

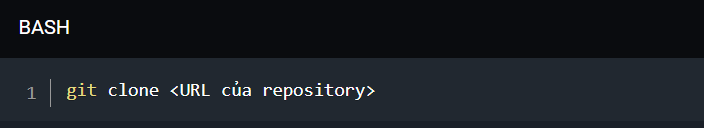
* 1. **Khởi tạo một Repository Git**

****

Khởi tạo một kho chứa (repository) Git tạo ra một không gian lưu trữ để theo dõi và quản lý mã nguồn cùng với lịch sử thay đổi.

* 1. **Clone một Repository sẵn có trong Git**

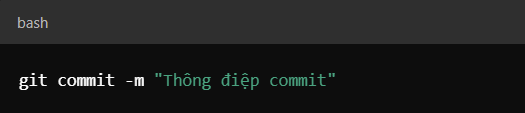
Việc "sao chép một repository sẵn có" (clone a repository)  tạo một bản sao hoàn chỉnh của kho chứa (repository) từ một nguồn gốc đã tồn tại, thường là từ các dịch vụ như GitHub, GitLab hoặc máy chủ Git cá nhân. Khi thực hiện thao tác sao chép này, tạo ra một bản nhân bản của tất cả dữ liệu trong kho chứa, bao gồm cả mã nguồn và lịch sử thay đổi.

****

* 1. **Thêm file vào vùng chuẩn bị (Staging Area)**

****

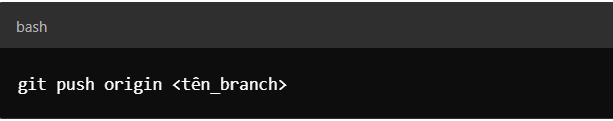
* 1. **Commit thay đổi**

****

* 1. **Xem trạng thái hiện tại**

****

* 1. **Đẩy thay đổi lên remote repository**

****

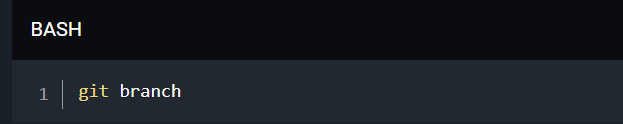
* 1. **Kéo các thay đổi từ remote repository về**

****

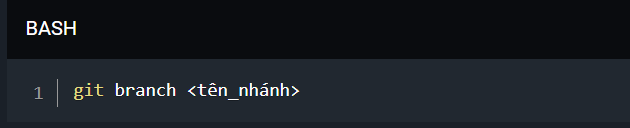
* 1. **Tạo Branch trong Git**

Trong quá trình làm việc với Git, có thể tạo nhiều nhánh (branch) khác nhau để phát triển tính năng, sửa lỗi hoặc thử nghiệm. Dưới đây là các lệnh sử dụng để làm việc với nhánh:

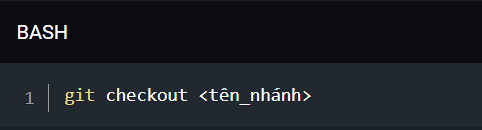
***Để kiểm tra nhánh hiện tại:***



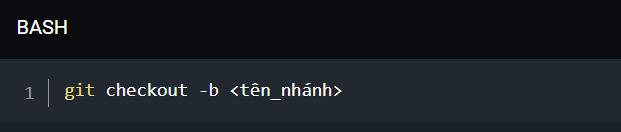
***Để tạo một nhánh mới:***



***Chuyển sang nhánh khác:***

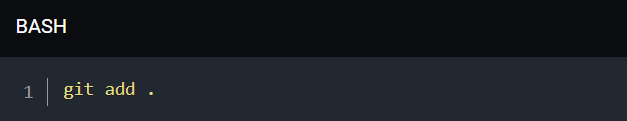
******

***Để chuyển đến một nhánh và tạo mới cùng một lúc:***

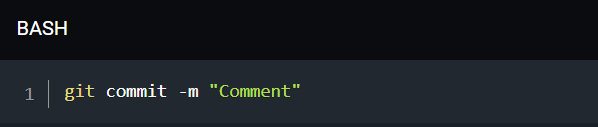


***Áp dụng các thay đổi:***

Sau khi đã chỉnh sửa mã nguồn, thêm, sửa hoặc xóa tệp, cần cập nhật chúng vào branch của. Để thêm tất cả các tệp ta dùng lệnh sau:

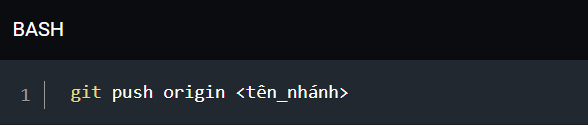


Sau khi sử dụng lệnh "add", cần thực hiện lệnh "commit" để đưa thông tin về các thay đổi lên Kho chứa Cục bộ (Local Repository):

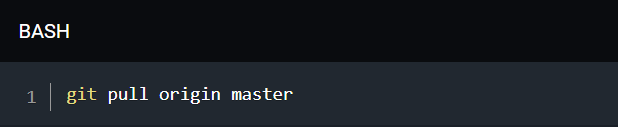


***Đồng bộ lên máy chủ (Git Server):***

Sau khi thực hiện lệnh "Commit", thông tin mới chỉ được lưu trữ trên Kho chứa Cục bộ (Local Repository). Để đồng bộ thông tin lên máy chủ, cần sử dụng lệnh "push":



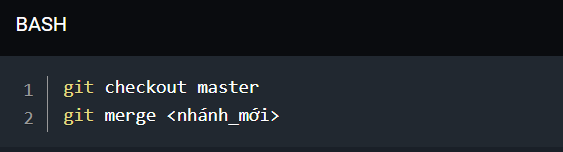
***Pull data từ Remote Repository:***



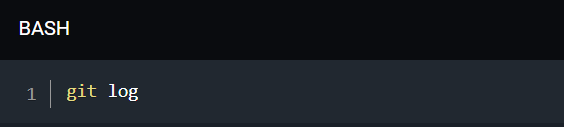
Lệnh trên sẽ hợp nhất các thay đổi mới từ máy chủ từ xa vào nhánh hiện tại trên máy cục bộ.

***Kết hợp nhánh:***

Sau khi đã thực hiện cập nhật tệp và đẩy chúng lên Git trên một nhánh mới, bước tiếp theo là kết hợp (merge) mã nguồn trở lại vào nhánh chính (master). Để làm điều này, bạn cần chuyển ra khỏi nhánh hiện tại và chuyển sang nhánh master, sau đó sử dụng lệnh merge để kết hợp nhánh mới vào master:



***Xem lịch sử commit:***



Lệnh "git log" sẽ cung cấp thông tin về người thực hiện commit, thời gian, và thông điệp của các lần commit.

1. **Tổng kết**

Git là một công cụ mạnh mẽ giúp theo dõi và quản lý mã nguồn hiệu quả, đặc biệt trong các dự án có quy mô lớn và yêu cầu sự cộng tác của nhiều thành viên. Việc sử dụng Git giúp tăng năng suất và đảm bảo tính an toàn của mã nguồn.