Everything you need in project management is Git

Tran Anh Tuan

Ngày 14 tháng 5 năm 2024

Saigon University, Vietnam
Dasanbob22122002@gmail.com

Tóm tắt nội dung

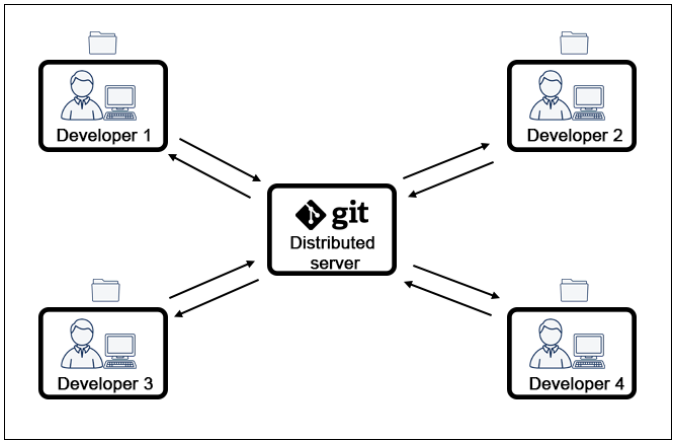
Quản lý mã nguồn bằng Git là 1 hướng tiếp cận mà gần như các lập trình viên ngày nay rất ưa thích sử dụng. Ngày nay với việc quản lý và chia sẽ mã nguồn để làm việc cùng nhau là điều không thể tránh khỏi. Bởi thế việc hiểu và sử dụng được các câu lệnh Git được xem là bắt buộc cho các lập trình viên mới vô ngành.

Từ khóa: Quản lý mã nguồn, làm việc cùng nhau, Git.

1 Giới thiệu

Quản lý mã nguồn là quá trình quản lý và điều chỉnh mã nguồn của một dự án phần mềm từ lúc bắt đầu phát triển đến khi hoàn thành và duy trì. Điều này bao gồm việc theo dõi các phiên bản của mã nguồn, quản lý sự thay đổi, và hợp nhất các thay đổi từ nhiều nguồn khác nhau.

Git là một hệ thống quản lý phiên bản phân tán, được phát triển bởi Linus Torvalds vào năm 2005 để quản lý mã nguồn cho dự án Linux. Được xem là một trong những công cụ quản lý mã nguồn phổ biến nhất hiện nay, Git cung cấp một cách linh hoạt và mạnh mẽ để theo dõi, quản lý và hợp nhất các phiên bản của mã nguồn trong một dự án phần mềm.



Hình 1: Minh họa các thức hoạt động của Git

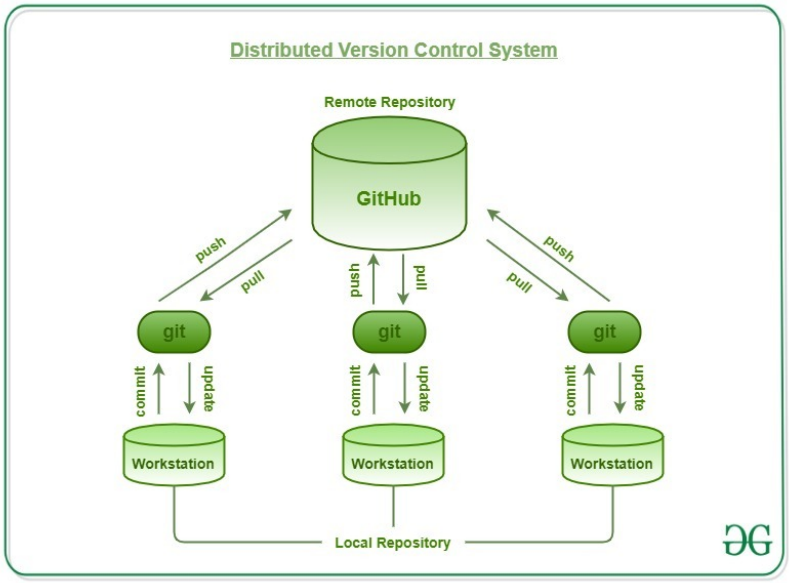
Với Git, mỗi thành viên trong dự án có thể làm việc độc lập trên các phiên bản của mã nguồn mà không cần kết nối mạng, sau đó họ có thể hợp nhất các thay đổi của mình với phiên bản chính của dự án một cách dễ dàng. Git cũng cung cấp các tính năng như nhánh (branching) và hợp nhất (merging), cho phép các nhà phát triển làm việc song song trên nhiều tính năng và sửa đổi mà không gây xung đột.

Với sự phổ biến của nền tảng mã nguồn mở và các dự án phần mềm phức tạp, Git đã trở thành một công cụ không thể thiếu cho các nhà phát triển phần mềm ở mọi cấp độ, từ các dự án cá nhân đến các tổ chức lớn. Ngày nay, với nhu cầu lớn về việc quản lý mã nguồn và yêu cầu khác nhau. Bởi thế có rất nhiều dịch vụ cung cấp kho lưu trữ mã nguồn Git dựa trên nền web cho các dự án phát triển phần mềm

Ở Bài viết này, sứ mệnh sẽ mang đến cho người đọc 1 cái nhìn tổng quát và rõ ràng về việc quản lý mã nguồn bằng dịch vụ Git. Phần 2 của bài viết sẽ viết về kiến trúc của Git. Tất tần tật các câu lệnh xử lý trên Git sẽ được nói ở phần 3. Và Phần 4 sẽ thực nghiệm trên 1 số dịch vụ Git.

2 Kiến Trúc của Git

Kiến trúc của Git được thiết kế để linh hoạt, phân tán và bảo đảm tính nhất quán trong quản lý mã nguồn. Một số điểm cơ bản về kiến trúc của có thể nói đến như: hệ thống phiên bản phân tán, lịch sử và phiên bản, cấu trúc nhánh, phân tích hợp nhất, vùng làm việc. Hệ thống Phiên bản Phân tán (Distributed Version Control System - DVCS):



Hình 2: Kiến trúc mô hình của Git

Git là một hệ thống quản lý phiên bản phân tán, điều này có nghĩa là mỗi người dùng có bản sao đầy đủ của toàn bộ lịch sử mã nguồn trên máy của mình.

Lịch sử và Phiên bản: Git lưu trữ lịch sử của dự án dưới dạng một chuỗi các phiên bản (commits). Mỗi commit đều chứa thông tin về sự thay đổi trong mã nguồn cũng như thông tin về người tạo ra thay đổi và thời gian thực hiện.

Cấu trúc nhánh (Branching): Git sử dụng cấu trúc nhánh linh hoạt, cho phép người dùng tạo ra, hợp nhất, và quản lý các nhánh của dự án dễ dàng.

Phân tích hợp nhất (Merge): Git cung cấp các công cụ phân tích hợp nhất mạnh mẽ để tự động hoặc thủ công hợp nhất các thay đổi từ các nhánh khác nhau. Vùng làm việc (Working Directory), Kho (Repository) và Kho từ xa (Remote Repository): Git sử dụng một cấu trúc ba lớp gồm vùng làm việc, kho (local repository), và kho từ xa (remote repository).

3 Tất tần tật các câu lệnh xử lý trên Git

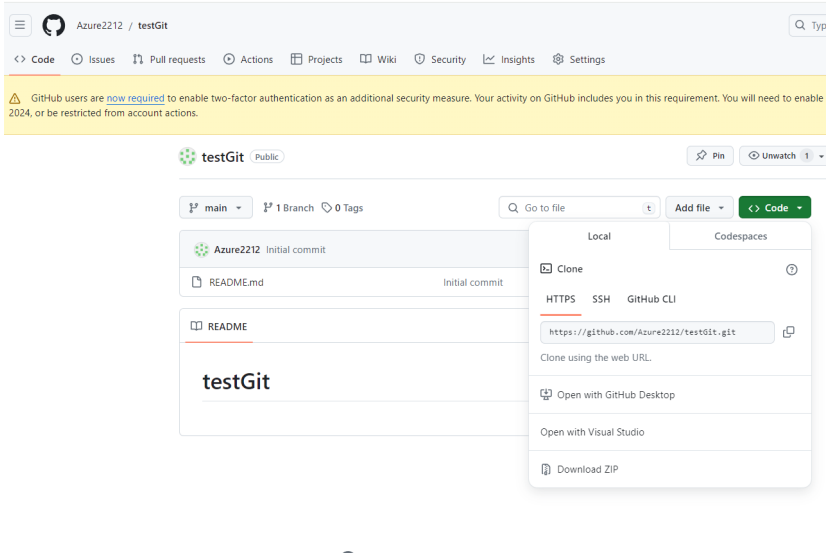
Như đã nói Git là như một bộ công cụ linh hoạt và mạnh mẽ, giúp bạn kiểm soát hoàn toàn quy trình phát triển phần mềm. Từ việc theo dõi lịch sử mã nguồn đến việc quản lý các nhánh và hợp nhất các thay đổi, Git cung cấp mọi công cụ bạn cần để điều hành dự án của mình một cách hiệu quả. Các thao tác cơ bản sẽ được nêu lên ở bảng 1, ngoài ra có thể tham khảo thêm Git Tutorial[1] của trang học trực tuyến nổi tiến W3school

Bảng 1: Các câu lệnh Git

	cú pháp	mô tả
1	git clone repo_url	lấy dự án từ địa chỉ repo_url về máy
2	git clone --branch x repo_url	lấy dự án từ địa chỉ repo_url với nhánh x về máy
3	git clone repo_url your_name	lấy dự án từ địa chỉ repo_url về máy và đặt tên là your_name
4	git pull	cập nhật những gì đã thay đổi trên remote vào local
5	git checkout	kiểm tra nhánh hiện tại
6	git checkout branch_x	chuyển đến nhánh branch_x (chuyển luôn code chưa cập nhật sang theo)
7	git checkout -b x	tạo nhánh x từ nhánh hiện tại và chuyển đến x
8	git branch -D branch_x	xóa nhánh branch_x trên local
9	git push origin -- branch_x	xóa nhánh branch_x trên remote
10	git branch a	cho xem danh sách các nhánh trên local và cả remote với nhánh hiện tại có dấu *
11	git stash push	cất hết những file chưa commit vào vùng giấu của nhánh hiện tại
12	git stash pop	lấy lại những gì đã cất
13	git stash clear	xóa hết những gì đã cất
14	git add x	thêm x vào khu vược hoạt động của git
15	git add .	thêm tất cả vào khu vược hoạt động của git
16	git check -- x .	hủy các thay đổi của file tên x
17	git reset -- hard Head .	hủy tất cả thay đổi trên nhánh
18	git commit -m "commit message"	cập nhật những thay đổi hiện tại với thông tin trong ""
19	git push origin branch_x	cập nhật vào nhánh $branch_x$ trên remote
20	git log	xem log dự án (nếu muốn xem chi tiết thì thêm --summary)

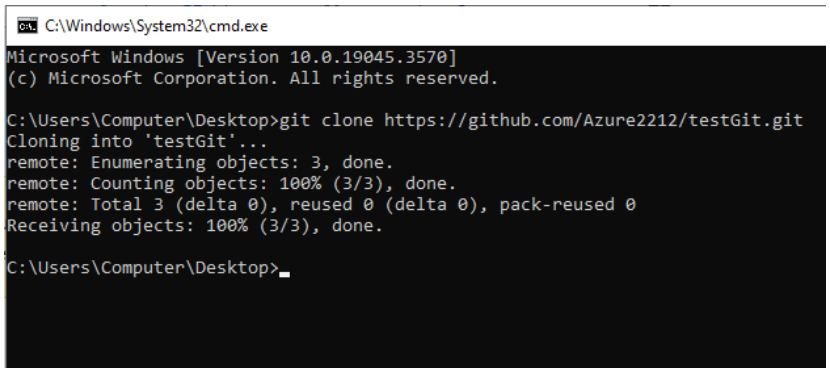
4 Thực nghiệm

Để dễ hình dung các lý thuyết và câu lệnh đã được viết trong bài báo này. Phần thực nghiệm sẽ sử dụng dịch vụ github để quản lý mã nguồn dự án. Đầu tiên ta sẽ truy cập vào trang Github.com để đăng ký sau đó đăng nhập và tạo 1 repository(dự án)(chọn Add a README file để dự án được tạo ngay lập tức).



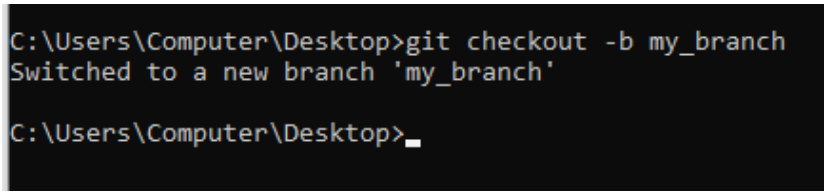
Hình 3: Tạo dự án trên Github

Với đường dẫn HTTPS ở hình 3, ta có thể sử dụng câu lệnh thứ 1 trong bảng 1 để lấy dự án về máy nếu muốn đặt tên khác mặc định có thể sử dụng câu lệnh thứ 2 cùng trong bảng 1



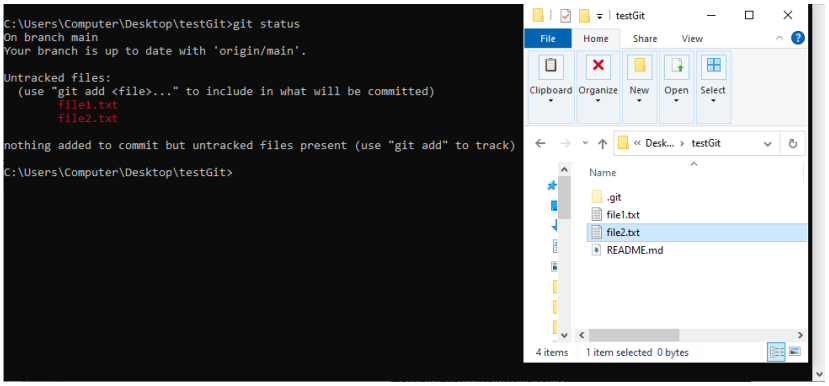
Hình 4: lấy dự án về máy

Ở đây, tùy vào mục đích sử dụng bạn có thể sử dụng bất kì câu lệnh nào trong bảng 1. đầu tiên hãy bắt đầu với việc tạo nhánh khi thực hiện một công việc hiện tại tránh ảnh hưởng không mong muốn để mã nguồn đã được hoàn thiện trên nhánh hiện tại.



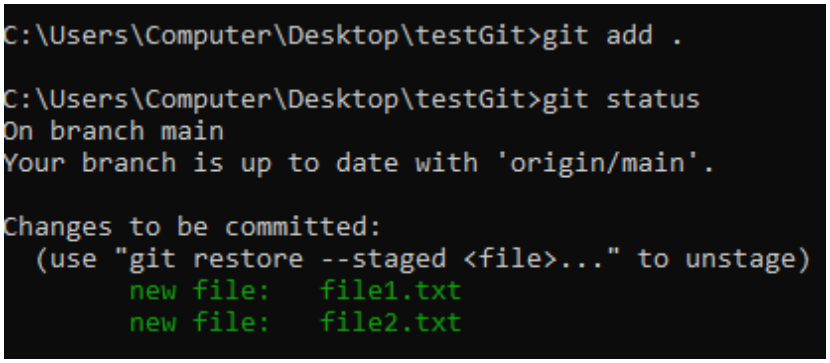
Hình 5: tạo nhánh mới tên là my_branch

Bạn cũng có thể tạo các tập tin(File), thư mục với các nội dung khác nhau để thực hiện nhu cầu của mình. sử dụng git status để biết trang thái các cập nhật



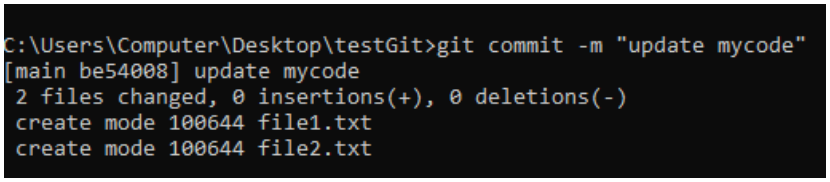
Hình 6: kiểm tra trạng thái các tệp đã thêm

Câu lệnh git status sẽ cho bạn biết các trạng thái của file. Nếu tên file màu đỏ nghĩa là các file đó chưa được thêm vào khu vực hoạt động của git nếu muốn thêm vào khu vực này bạn có thể sử dụng git add [tên file] cho file bạn muốn thêm vô hoặc git add . cho tất cả các file chưa được thêm vào. Còn nếu tên file màu xanh nghĩa là file đã sẵn sàng cho việc cập nhật.



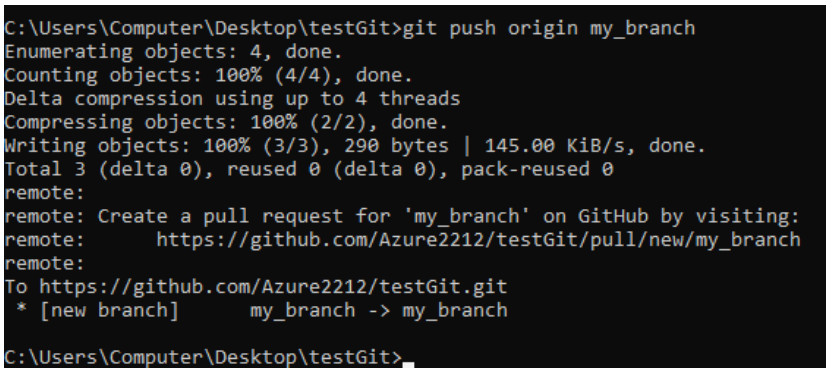
Hình 7: kiểm tra trạng thái các tệp đã thêm vào khu vực hoạt động của git

Các thao tác thêm, sửa, xóa file sẽ theo chúng ta đến bất kì đâu khi chúng ta chuyển nhánh, để cập nhật thông tin riêng cho nhánh hiện tại của mình. ta sử dụng câu lệnh git commit -m "update mycode"những thứ trong ""chính là nội dụng của lần cập nhật (commit này)



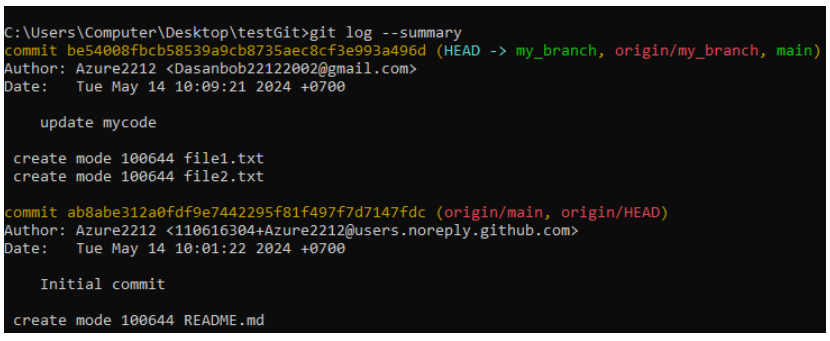
Hình 8: cập nhất các thay đổi vào nhánh hiện tại

Những thao tác / thay đổi trên chỉ ảnh hưởng đến mã nguồn của bạn trên máy tính hiện tại(local). Nếu không còn thay đổi gì và muốn cập nhật những thay đổi tên remote để các thành viên khác trong nhóm có thể sử dụng những thay đổi đó. Để thực hiện được điều này ta phải đẩy chúng lên remote thông câu lệnh git push origin [nhánh mình đã commit].



Hình 9: cập nhất các thay đổi lên remote

Sau khi cập nhất toàn bộ các thay đổi lên remote, các thành viên khác có thể cập nhất các thay đổi đó bằng câu lệnh git pull và xem các thay đổi đã diễn ra trên nhánh đó qua câu lệnh git log --summary để hiện ra các lần thay đổi(1 cách chi tiết).



Hình 10: lich sử các thay đổi trên nhánh

5 CONCLUSION

Trong bài giới thiệu này, chúng ta đã khám phá sâu hơn về Git, một hệ thống quản lý phiên bản phân tán mạnh mẽ được sử dụng rộng rãi trong phát triển phần mềm ngày nay.

Git có một kiến trúc hoạt động phân tán, cho phép nhiều người làm việc trên cùng một dự án mà không cần phụ thuộc vào một máy chủ trung tâm. Thay vào đó, mỗi máy tính trong mạng Git đều chứa toàn bộ lịch sử phiên bản của dự án. Điều này tạo điều kiện cho sự phân phối linh hoạt và đồng bộ hóa dữ liệu giữa các thành viên trong nhóm.

Một số cú pháp thông dụng để sử dụng của Git đã được liệt kê sử dụng để tương tác với các kho lưu trữ từ xa như GitHub hoặc GitLab. Ngoài ra để trực quan hóa các hoạt động của lập trình viên cho việc quản lý mã nguồn bằng Git mà ngay nay các công cụ trực quan hóa Git đã ra đời. Có thể kể đến tiêu biểu nhất là Github Desktop[2], cho phép người dụng thao tác các câu lệnh Git 1 cách dễ hình dung mà không cần sử dụng lệnh

Việc hiểu và sử dụng thành thạo Git là một kỹ năng quan trọng đối với các nhà phát triển phần mềm, giúp họ quản lý hiệu quả mã nguồn và làm việc hiệu quả trong môi trường phát triển phần mềm phức tạp.

Tài liệu

[1] *Git Tutorial*, [Online].
Available: `https://www.w3schools.com/git/`

[2] *Github Desktop*, [Online].
Available: `https://docs.github.com/en/desktop/overview/getting-started-with-github-desktop`