## Git là gì

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository) riêng chứa toàn bộ lịch sử thay đổi.

Git ra đời vào năm 2005, được tạo bởi người sáng lập Linux Kernel - Linus Torvalds. Sau đó, nó được vận hành hiệu quả bởi Junio Hamano, một lập trình viên người Nhật.

Git được phát triển để thay thế BitKeeper, hệ thống quản lý phiên bản trước đó mà cộng đồng phát triển Linux kernel đã sử dụng. Do mâu thuẫn với nhà cung cấp BitKeeper, cộng đồng Linux cần một hệ thống quản lý phiên bản mới, mạnh mẽ và linh hoạt hơn.

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”).

Các bạn có thể tìm hiểu sâu hơn về Git ở đây [Introduction Git](https://yunwuxin1.gitbooks.io/git/content/vi/c8cf03f9bf3367e7d612a0fa058ed068/)

## Phân biệt giữa Git và Git Server

Có khá nhiều bạn (trong đó có cả mình =)))) khi mới tiếp xúc thì cứ tưởng git với github là một nhưng thật ra 2 cái đó là khái niệm khác nhau =)))

Máy chủ Git, hay còn gọi là Git server, là một máy chủ mà trên đó đã được cài đặt dịch vụ Git. Chức năng chính của máy chủ này là lưu trữ và quản lý mã nguồn từ các dự án khác nhau. Khi nhà phát triển làm việc trên dự án, họ có thể tương tác với máy chủ Git để tải lên, tải xuống và thực hiện các thao tác quản lý phiên bản trên mã nguồn. Máy chủ Git đóng vai trò quan trọng trong việc duy trì tích hợp và quản lý sự phát triển của dự án.

Ví dụ: Github đóng vai trò như một máy chủ Git. Bạn có thể tạo một kho chứa (repository) trên Github, gọi là kho chứa từ xa (remote repository). Sau đó, bạn có thể sao chép (clone) kho chứa từ xa này về máy tính của mình để tạo thành kho chứa cục bộ (local repository).

Một số git server phố biến hiện nay như: Github, Gitlab, Azure DevOps Services, Bitbucket,…

Tóm lại dễ hiểu thì :

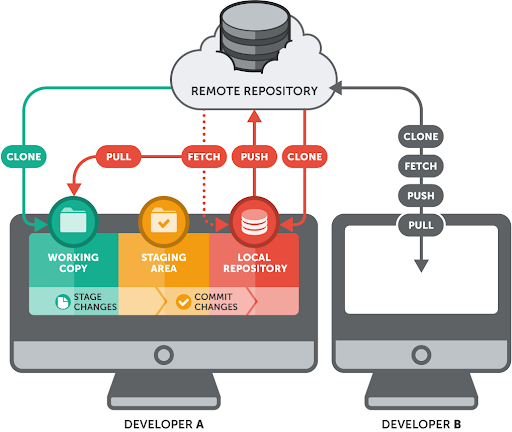
* Git là một hệ thống quản lý phiên bản phân tán, dùng các câu lệnh để theo dõi, quản lý thay đổi trong mã nguồn cục bộ.
* Github là máy chủ lưu trữ và quản lý kho Git, cung cấp truy cập mạng, lưu trữ kho Git, cung cấp dịch vụ truy cập và hợp tác

## Các lệnh git cơ bản

Bên dưới là các lệnh thường sài trong git.

|  | git init  git add <file\_name>  git add .  git status  git commit -m"add your commit"  git log  git remote add origin https://github.com/<your\_repository>.git  git push origin <name\_branch> // usually is main  git branch <name\_branch>  git checkout <name\_branch>  git checkout -b <name\_branch>  git pull  git merge  git branch -d tên-nhánh  git clone URL |
| --- | --- |

## Mô hình minh họa trạng thái của Git



Khu vực đầu tiên gọi là Working Copy

* Đại khái đây là nơi bạn làm việc với các thư mục hoặc là clone code, pull code từ trên remote về.

Khu vực trung gian gọi là Staging Area

* Staging Area là khu vực lưu trữ những thay đổi trên tập tin so sánh giữa repo chính và repo clone trước commit
* Khi commit tập tin thì tập tin đó phải nằm trong Staging Area. Một tập tin khi nằm trong Staging Area sẽ có trạng thái là Stagged

Khu vực cuối cùng gọi là Local Repository

* Sau khi bạn commit xong, đây là nơi mà các file cũng như các commit chờ được push lên để đồng bộ trên remote repository.

Remote Repository: Đặt trên máy chủ Git (git server), có khả năng chia sẻ với nhiều người.

Local Repository: Đặt trên máy cá nhân, dành riêng cho người dùng. Bạn có thể sao chép (clone) Remote Repository về tạo thành Local Repository. Khi có sửa đổi ở Local Repository, bạn có thể đẩy (push) mã lên Remote Repository.

## Trạng thái chỉ thị Commit

Git có hai loại trạng thái chính:

### Tracked

Là tập tin đã được đánh dấu theo dõi trong Git. Trạng thái Tracked sẽ có thêm các trạng thái:

* Unmodified (chưa chỉnh sửa gì)
* Modified (đã chỉnh sửa)
* Staged (đã sẵn sàng để commit).
* Untracked – Là tập tin không muốn làm việc với nó trong Git.

### Untracked

* Xuất hiện khi tạo mới hoặc thêm vào một tập tin mới vào trong thư mục repo đang làm việc. Nó sẽ ở trạng thái Untracked.

## Sử dụng Git

### Cài đặt Git

Các bạn có thể cài từ trang chủ của Git dựa vào hệ điều hành máy tương ứng nhé

[Link Tải Git](https://git-scm.com/downloads)

Sau khi cài xong, vào terminal và chạy 2 dòng lệnh sau

|  | git config --global user.name "Tên của bạn"  git config --global user.email "email@example.com" |
| --- | --- |

### Tạo một Repository trên Github

Ở đây mình sẽ sử dụng Git Server là Github nhé, ngoài ra thì vẫn còn rất nhiều nền tảng git server khác như gitlab hay Azure DevOps Services,…

* Đầu tiên các bạn tạo một tài khoản github từ trang chủ của [Github](https://github.com/)
* Sau khi tạo xong, chọn vào phần “Repository” sau đó chọn “New”
* Rồi gõ vào ô “Repository Name” để tạo tên của Repository sau đó ấn “Create Repository” là được

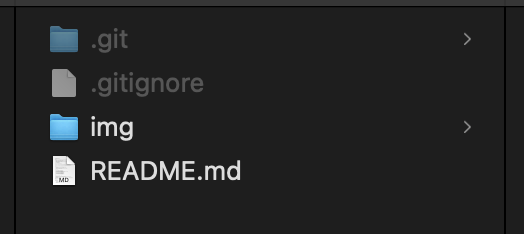
### Git init

Khởi tạo 1 kho lưu trữ (Repository) trên máy của bạn, ở đây mình sẽ hướng dẫn trên Terminal của VSCode nhé.

Đầu tiên, các bạn mở thư mục chứa dự án mình cần quản lý trên VSCode, sau đó gõ vào terminal

|  | git init |
| --- | --- |

Lệnh này sẽ tạo một thư mục mới có tên .git, thư mục này chứa tất cả các tập tin cần thiết cho kho chứa - đó chính là bộ khung/xương của kho chứa Git. (Thường thì thư mục này sẽ bị ẩn đi)



Không nên động vào thư mục .git

Sau khi khởi tạo kho lưu trữ trong folder bạn chọn, nếu bạn sử dụng VSCode như mình thì khi nhìn qua folder bạn sẽ thấy các file trong folder có màu xanh lá và chữ U như sau

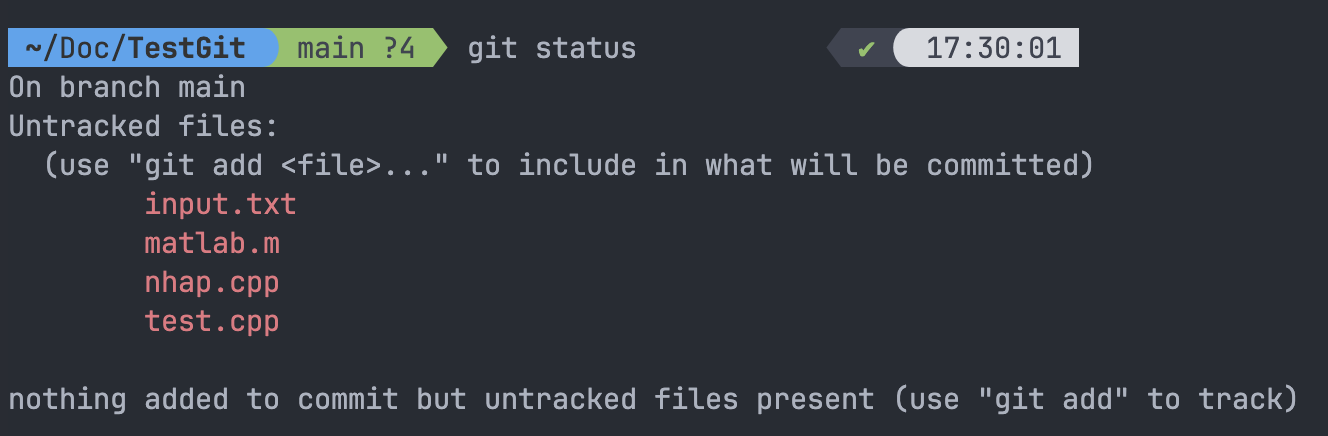


### Git status

Để kiểm tra trạng thái các file, các bạn gõ vào terminal câu lệnh

|  | git status |
| --- | --- |

Sau khi gõ xong nó sẽ hiển thị như hình



Nó nói ràng : trên nhánh main (On branch main) thì đang có 4 file chưa được add cũng như commit (cũng hiển thị luôn đó là 4 file nào)

Và công việc tiếp theo của chúng ta chính là add file lên

### Git add

Có 2 cách để add file với git

#### 1.Add từng file / chọn file

|  | git add <file\_name> |
| --- | --- |

Đây là cách add từng file có trong folder. Ví dụ, bạn muốn add file tên nhap.cpp và test.cpp thì

|  | git add nhap.cpp  git add test.cpp |
| --- | --- |

#### 2.Add tất cả file trong thư mục

Để add toàn bộ file trong thư mục đang quản lý, ta dùng lệnh

|  | git add . |
| --- | --- |

Khi sử dụng lệnh git add thành công, bạn đã đưa file vào khu vực “Staging Area”

### Git commit

Sau khi add file xong, việc tiếp theo cần làm là ta commit code. Dễ hiểu là bạn sẽ ghi lại thông tin của cái file đó (kiểu là mình update gì trong file đó, hay là add file này,…)

Commit là chỉ thị trên Git, lưu lại bản chụp (snapshot) các sự thay đổi trong thư mục làm việc, và các tập tin và thư mục được thay đổi đã phải nằm trong Staging Area.

Mỗi commit sẽ được lưu lại lịch sử chỉnh sửa của mã nguồn kèm theo tên và địa chỉ email của người commit.

Ngoài ra Git cho phép khôi phục lại tập tin trong lịch sử commit, cho phép phân nhánh (branch).

|  | git commit -m"add your messages" |
| --- | --- |

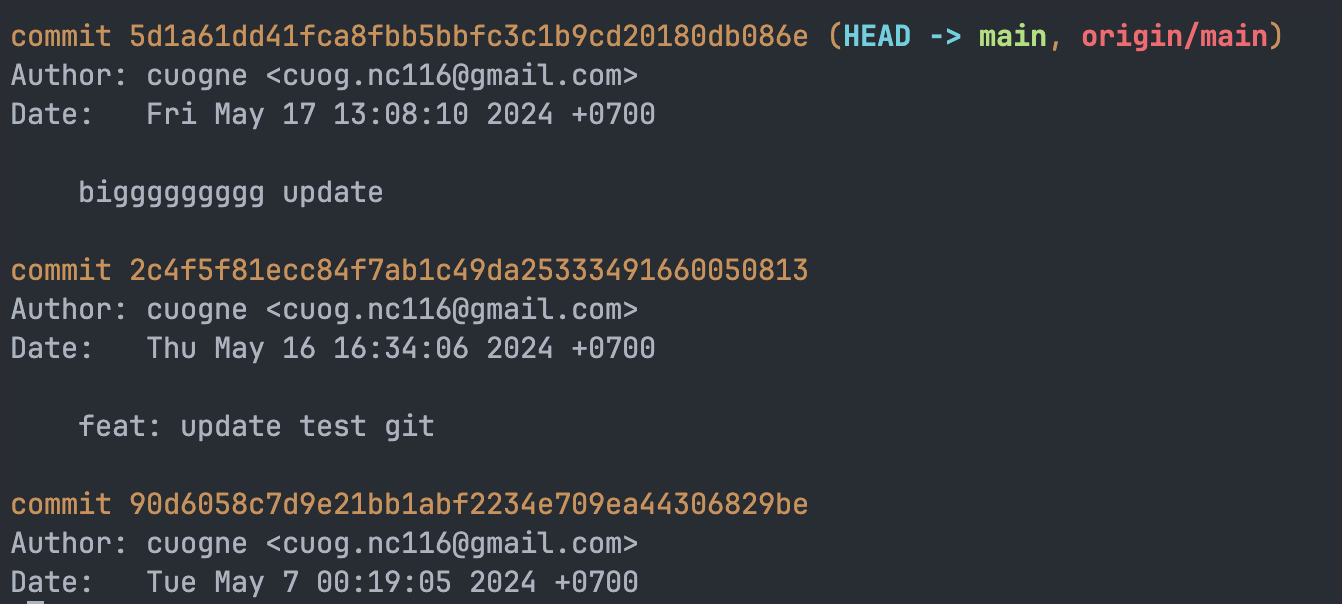
Thay dòng chữ {add your messages} đi là được

### Git log

Để kiểm tra lịch sử commit, ta dùng lệnh

Ta dùng lệnh

|  | git log |
| --- | --- |



Các bạn có thể thấy là nó hiển thị toàn bộ các commit trước đó bao gồm tên người commit, ngày giờ và cũng như nội dung.

### Git remote add origin

Đồng bộ code từ máy Local lên Github

|  | git remote add origin https://github.com/<user\_name\_github>/<name\_of\_repository>.git |
| --- | --- |

Bên trên chính là lệnh đồng bộ code từ máy local lên remote.

Phần URL chính là link github + username github của bạn + tên repo bạn tạo + .git

Ví dụ như của mình thì sẽ là

|  | git remote add origin https://github.com/cuogne/Testgit.git |
| --- | --- |

Sau khi bạn thực hiện lệnh này thì ở những lần push sau trong folder này, bạn sẽ không cần lặp lại lệnh này nữa.

### Git push

Đây là bước cuối cùng, đưa code từ local repo lên remote repo, ở đây server chính là github

|  | git push origin <name of branch> |
| --- | --- |

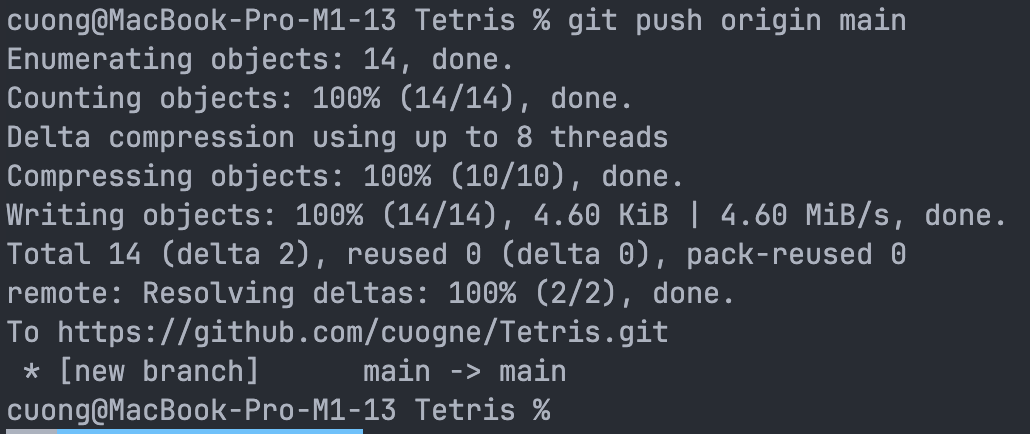
Lệnh này sẽ push code từ máy bạn lên repo bạn tạo trong github.

Name of branch tức là tên mà nhánh bạn đang làm việc (phần này mình sẽ nói bên dưới)

Ngày xưa thì nó là master nhma giờ thì đa số đổi thành main hết rồi

|  | git push origin main |
| --- | --- |

Sau khi thực hiện xong, đợi một chút để code được push lên



Sau đó lên repo của bạn trên github và reload lại ta sẽ có thành quả.

Hmm, tới đây các bạn đã có thể push code lên, gửi cho mọi người xem cũng như có một nơi để lưu trữ code rồi.

Bên trên chỉ là những lệnh cơ bản nhất. Từ bên dưới, chúng ta sẽ tìm hiểu cách làm việc trên các repo này, cách merge code cũng như cách tạo nhánh để có thể làm việc trên nhiều nhánh khác nhau

### Git 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.

Trong quá trình làm việc với Git, bạn có thể tạo nhiều nhánh (branch) khác nhau để phát triển tính năng, sửa lỗi hoặc thử nghiệm mà không ảnh hưởng tới nhánh chính

Tạo một nhánh mới

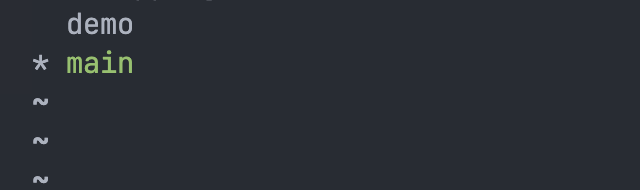
|  | git branch {name\_branch} |
| --- | --- |

với name\_branch là tên nhánh mà bạn muốn đặt

Giả sử ở đây mình đặt tên nhánh là demo nha

Để kiểm tra xem bạn đang ở branch nào thì chỉ cần

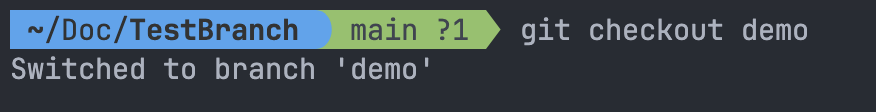
|  | git branch |
| --- | --- |



Đây là mình đang ở nhánh main, và nhánh còn lại là nhánh demo mình vừa tạo

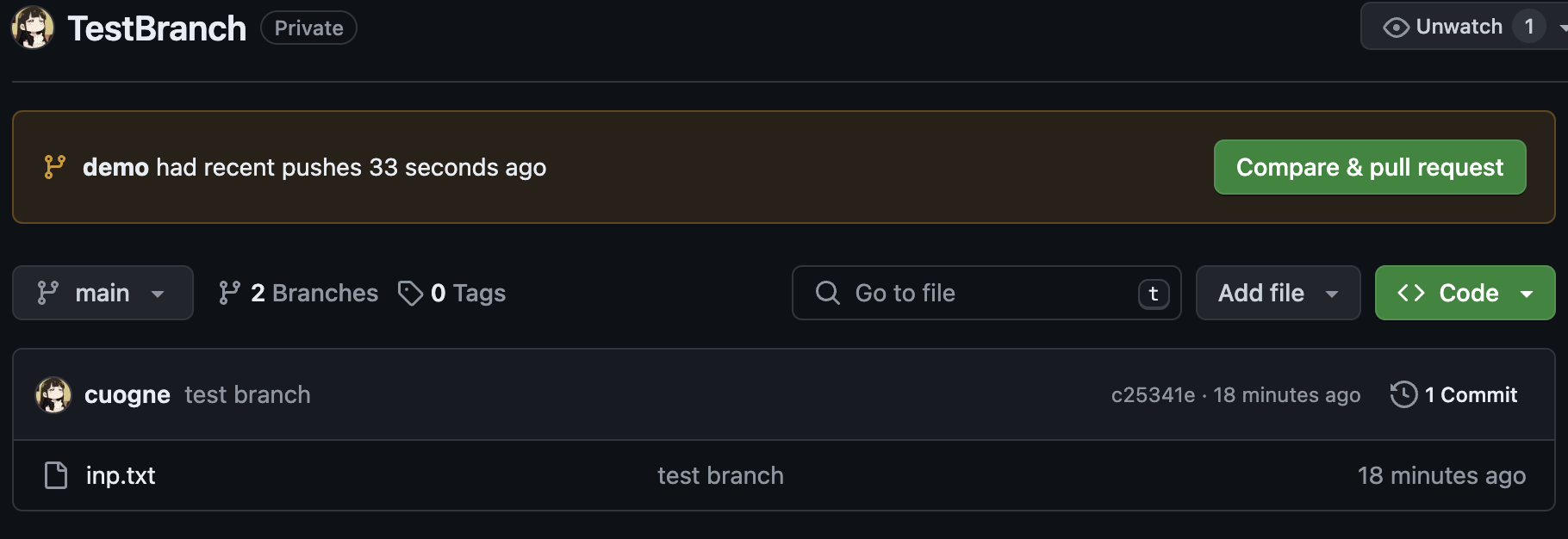
Để chuyển qua nhánh còn lại thì :

|  | git checkout demo |
| --- | --- |

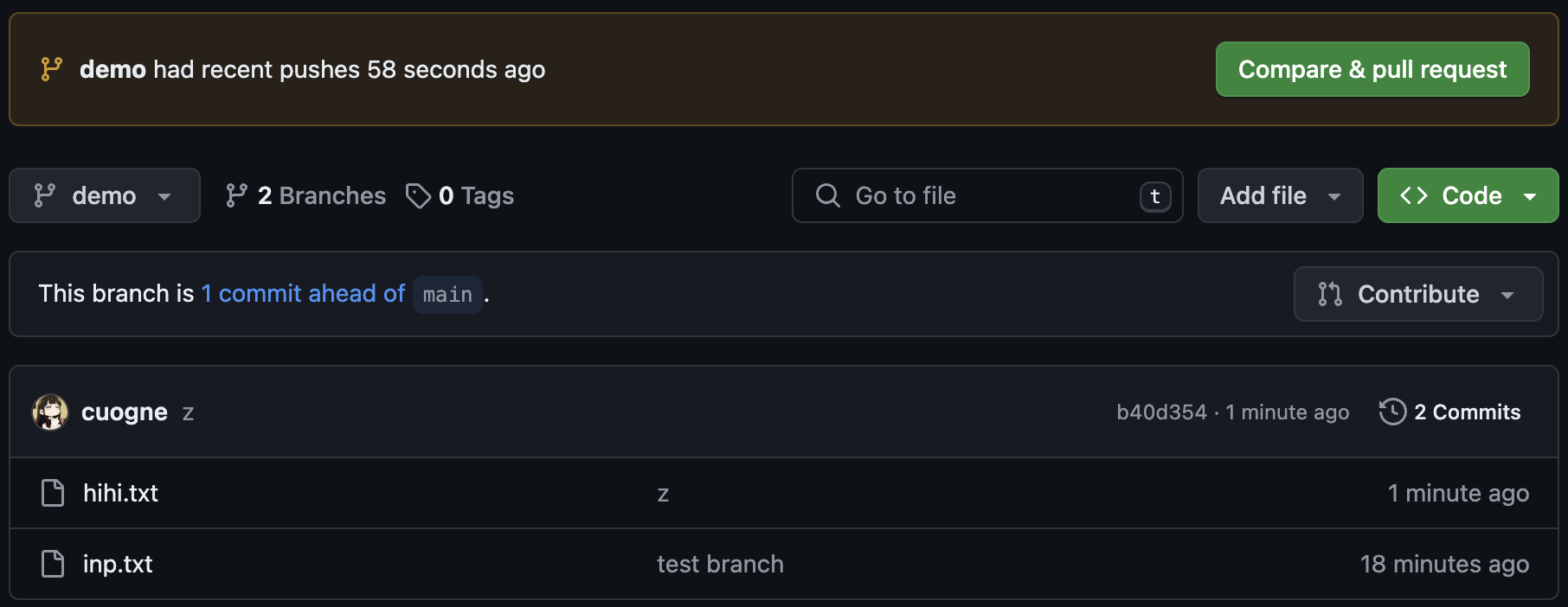


Sau đó, bạn có thể tạo file cũng như làm việc trên nhánh này mà không ảnh hưởng tới nhánh chính (main).

Nếu bạn push code lên repo thì cũng sẽ tạo thành 2 nhánh như hình bên dưới và bạn cũng sẽ thấy sự khác biệt



Nhánh main

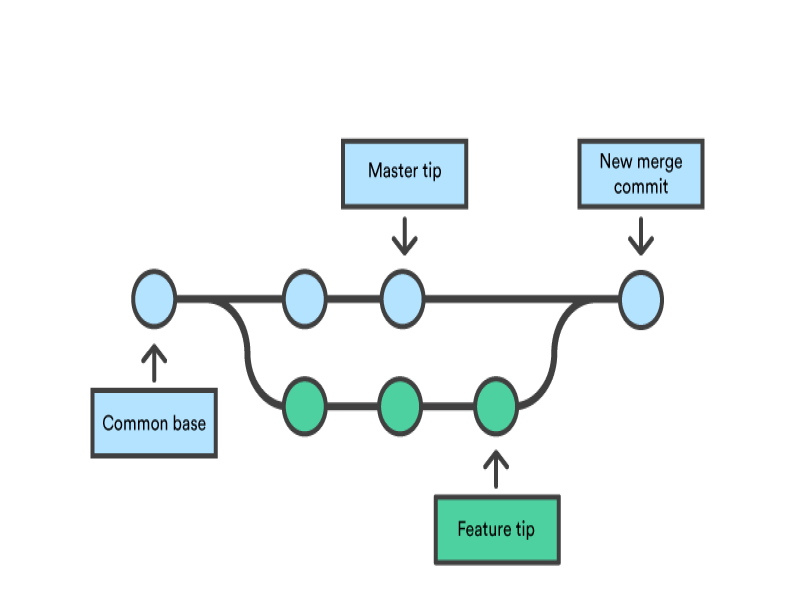


Nhánh demo

Vậy thì giờ làm cách nào để gộp các nhánh đó vào nhánh chính ?

### Git merge

Đây là hành động ghép code lại với nhau từ nhánh phụ vào nhánh chính



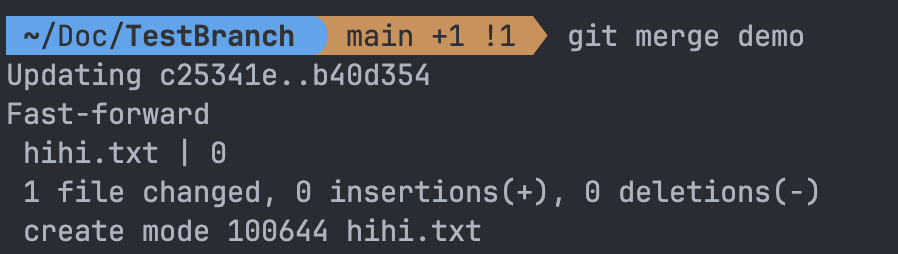
Ảnh minh họa nhánh và gộp nhánh

Trước hết, bạn phải quay về nhánh chính (main)

|  | git checkout main |
| --- | --- |

Sau đó điền vào nhánh muốn gộp

|  | git merge demo |
| --- | --- |

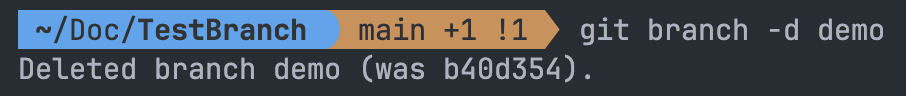


Bạn sẽ thấy nhánh demo đã được gộp thành công vào nhánh main

Nếu bạn đã hoàn thành, có thể xóa nhánh main trên local đi bằng cách

|  | git branch -d [branch\_name] |
| --- | --- |

Lưu ý : để xóa nhánh thì bạn phải ở một nhánh khác, không được ở nhánh đang xóa



Sau đó xóa nhánh trên remote

|  | git push origin --delete demo |
| --- | --- |

Xong rồi đó, giờ bạn có thể push lên lại nhánh main và tiếp tục thoi

### Git pull

Đây là lệnh sẽ giúp bạn đồng bộ code từ repo trên remote về máy local của bạn

|  | git pull |
| --- | --- |

Nếu bạn không đồng bộ, trong lúc làm việc nhóm sẽ xảy ra xung đột giữa các file.

### Git clone

Bạn có thể clone dự án của một người khác trên github bằng cách sử dụng lệnh

|  | git clone URL |
| --- | --- |

Khi bạn chạy lệnh git clone, Git sẽ tạo một bản sao của repository đó trên máy tính của bạn, bao gồm tất cả các file, lịch sử commit, nhánh, và cài đặt cấu hình.