**Tìm hiều về GIT**

**1. Những khái niệm chung**

**Git là gì?**

Git là 1 phần mềm cài đặt trên máy tính giúp cho chúng ta có thể quản lí file, có thể biết được khi nào thì 1 file nào trong folder đó bị thay đổi bằng 1 lệnh git

**Git repository là gì?**

Repository hay còn được gọi là repo.

Git repositorylà 1 nơi chứa tất cả các mã nguồn cho 1 dự án được quản lí bởi Git.

Trong Repo có 2 cấu trúc dữ liệu chính là Object Store và Index. Tất cả dữ liệu của Repo đều được chứa trong thư mục bạn đang làm việc dưới dạng folder ẩn có tên là .git.

Repository của Git được phân thành 2 loại : remote repository và local repository.

* Remote repository: Là repository để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng.
* Local repository: Là repository bố trí trên máy của bản thân mình, dành cho một người dùng sử dụng.

**Git & Github & Gitlab giống/khác gì nhau?**

**Khác nhau :**



**+** Git là 1 phần mềm còn GitHub và GitLab là 1 cái website.

+ Git là 1 công cụ còn GitHub và GitLab là 1 dịch vụ.

**Giống nhau :** Tất cả đều được liên kết với kiểm soát phiên bản

**Các trạng thái có thể có trong git repository**

Repo của Git được phân thành 2 loại : remote repo và local repo.

Các trạng thái có thể có của file trong repo :

**Local repo** **Remote repo**

Untracked files

Tracked files

racked file

Remote

racked file

Staged

Unmodified

Modified

**Untracked Files**: Các file này dù có thay đổi / thêm / xoá thì git cũng ko quan tâm, vì nó ko nằm trong danh sách theo dõi của nó.

**Tracked Files**: Những files đã được thêm vào danh sách theo dõi của git được gọi là Tracked Files, những file này khi chúng ta thay đổi / thêm / xoá thì git sẽ nhận biết được điều đó và lưu các thay đổi này lại theo yêu cầu của chúng ta.

**Modified Files**: Các file bị THAY ĐỔI

**Staged Files**: Các file bị THAY ĐỔI đã được ĐÁNH DẤU để commit

**Unmodified** **Files**: Khi các file đã được ĐÁNH DẤU sau khi được COMMIT sẽ lại trở về trạng thái ko thay đổi.

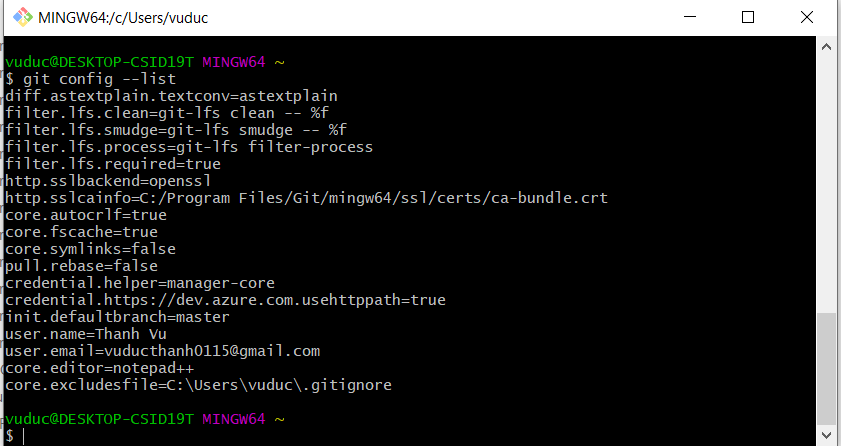
🡺 Do các thay đổi này sau khi được commit sẽ được lưu vào thư mục .git, và chúng ta có thể restore về trạng thái đó bất cứ lúc nào

**2. Config cho git với git config**

- Config thông tin cá nhân

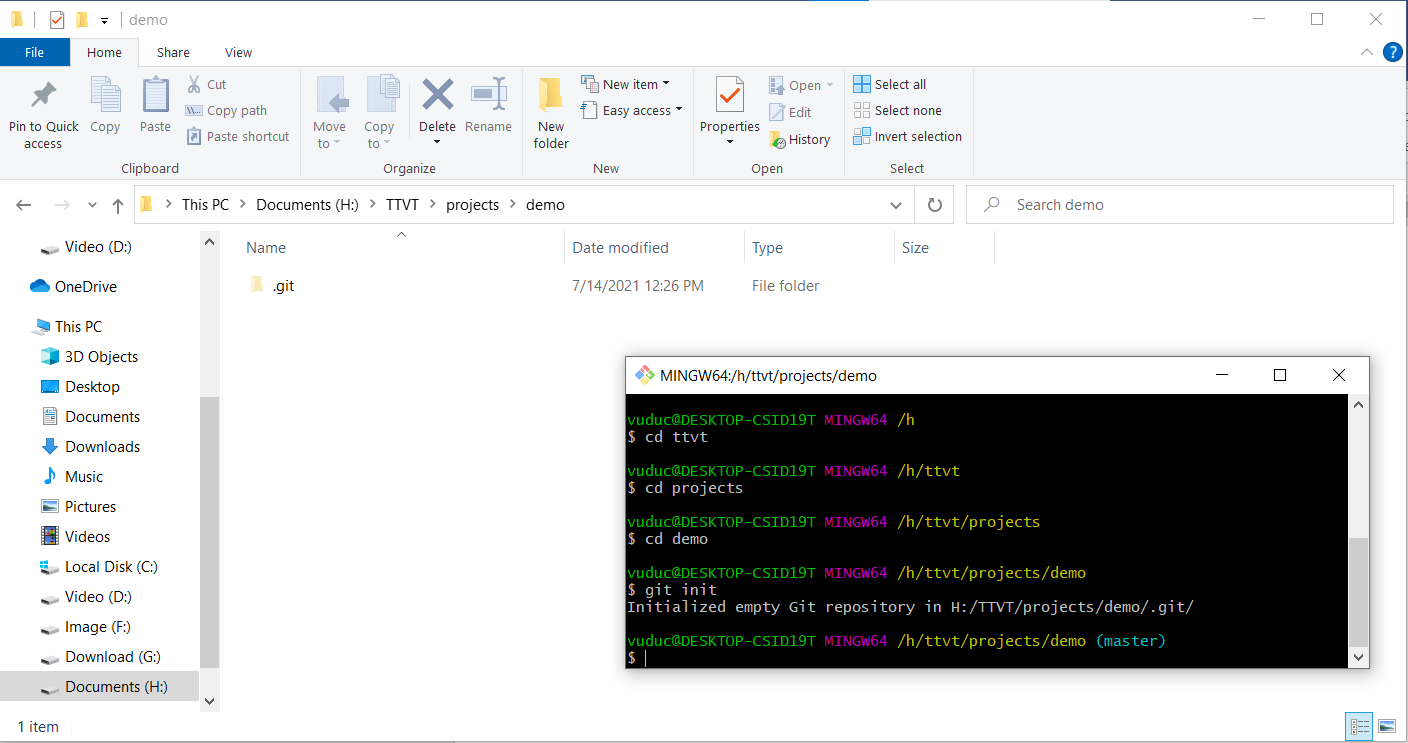
- Config default/prefered editor

- Config global .gitignore

****

**3. Các thao tác ban đầu**

**Khởi tạo repo với git init**

****

**4. Git commit**

**Commit trong git là gì ?**

Commit là thao tác báo cho hệ thống biết mình muốn lưu lại trạng thái hiện hành, ghi nhận lại lịch sử các xử lý như đã thêm, xóa, cập nhật các file hay thư mục nào đó trên repo.

Khi thực hiện commit, trong repo sẽ ghi lại sự khác biệt từ lần commit trước với trạng thái hiện tại. Các commit ghi nối tiếp với nhau theo thứ tự thời gian.

**Một commit chứa những thông tin gì ?**

Một commit chứa tất cả những thay đổi của mình tất cả những gì mình đã làm và có ngày giờ cụ thể tại thời điểm chúng ta thay đổi.

**Amending - Thay Đổi Commit Cuối Cùng**

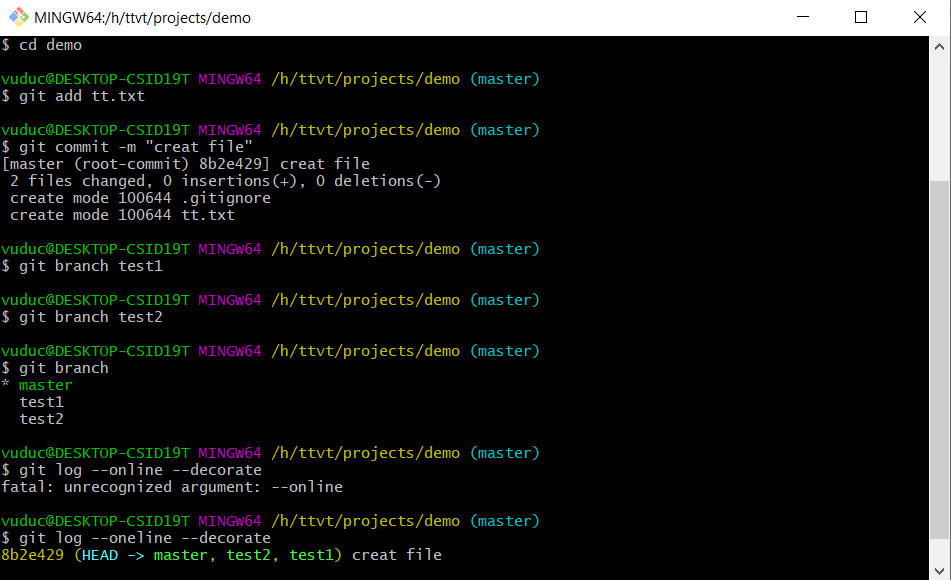
- Khi chúng ta commit nhưng bị quên add một số file nào đó và không muốn tạo ra một commit mới thì có thể sử dụng lệnh $ git commit --amend để gộp các file đó và bổ sung vào commit cuối cùng, vì vậy không tạo ra commit mới.

- Muốn loại bỏ một file ra khỏi stage thì sử dụng lệnh $ git reset HEAD <file\_name>.

**5. Làm việc với branch**

Branch là cái dùng để phân nhánh và ghi lại luồng của lịch sử. Branch đã phân nhánh sẽ không ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng 1 repo.

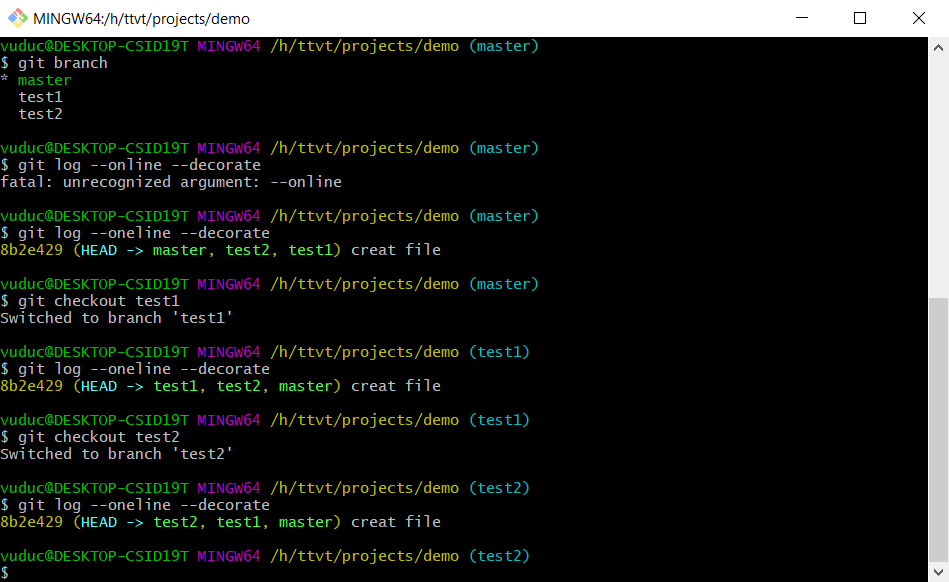
**Tạo mới branch**

****

* Khi tạo mới nhánh thì nhánh master và test đều trỏ tới cái commit hiện tại.

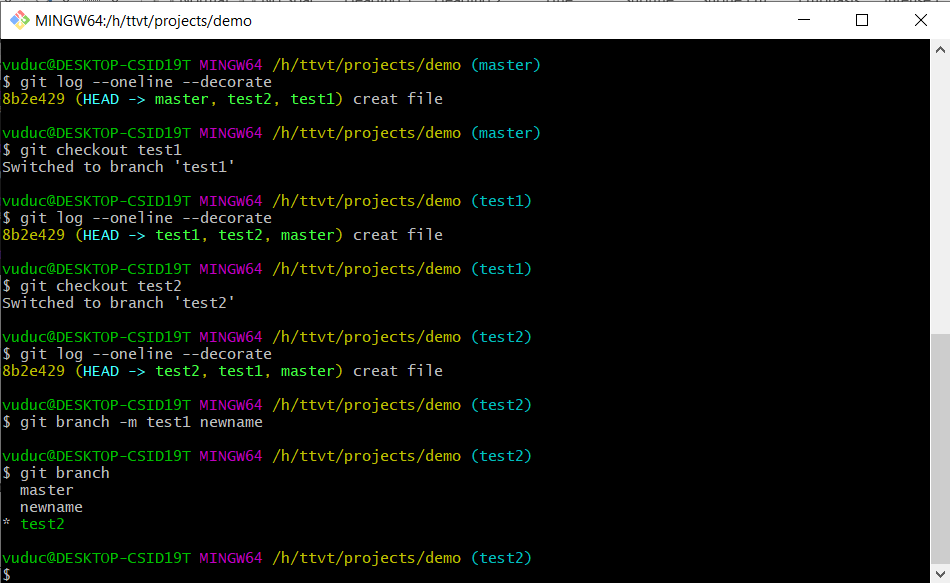
**Nhảy từ branch này sang branch khác**

Khi nhảy từ branch này sang branch khác ta dùng câu lệnh: $ git checkout <branch\_name>



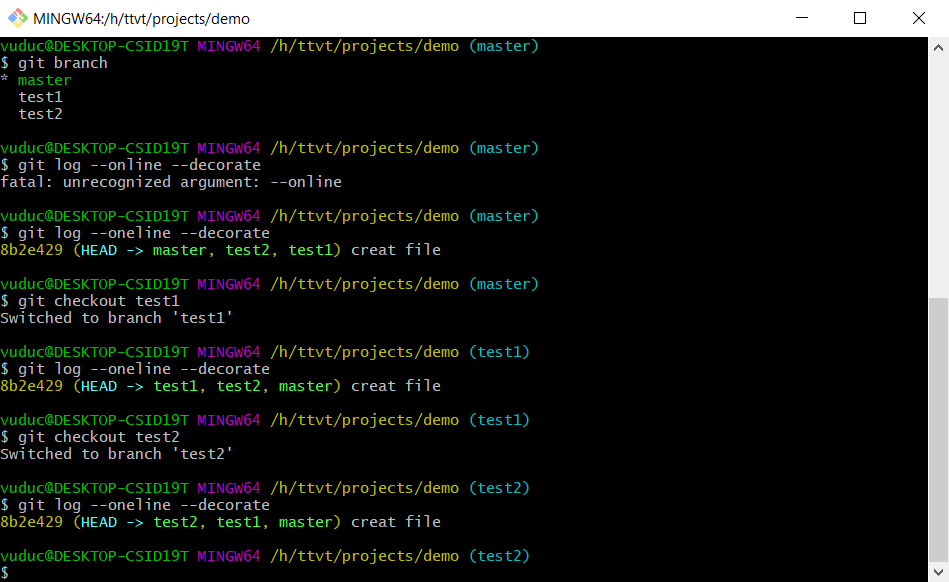
**Đổi tên branch**

Muốn đổi tên branch ta dùng lệnh: $ git branch -m <oldbranch> <newbranch>



**6. Git checkout**

Dùng để chuyển con trỏ HEAD sang một nhánh khác.



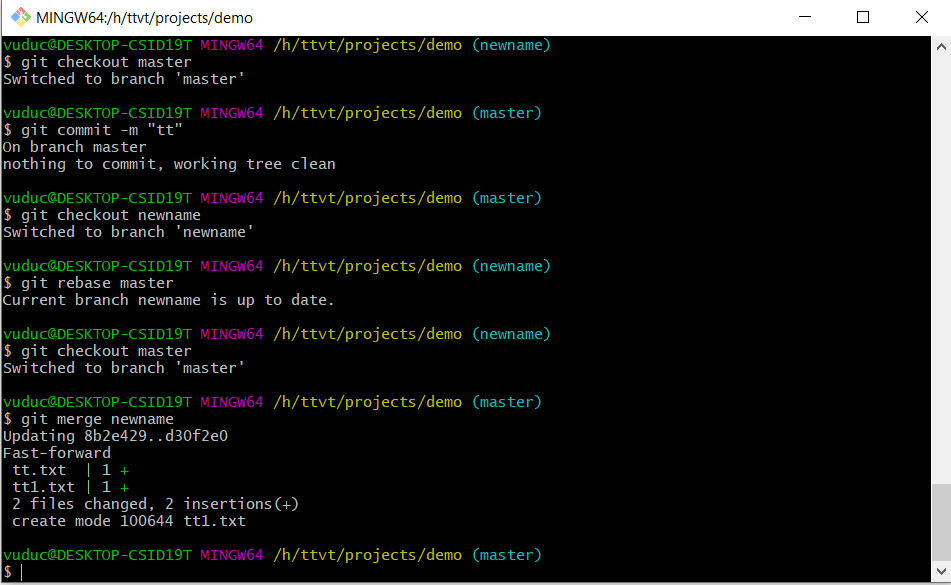
**7. Các thao tác liên quan khác**

**Git Merge**

Để merge một branch bất kì vào branch hiện tại thì bạn sử dụng lệnh:

$ git merge <branch\_name>

**Git Rebase**

****

**Git Pull**

Lệnh git pull lấy về thông tin từ remote và cập nhật vào các nhánh của local repo.

**Git Push**

Lệnh git push được sử dụng để đẩy các commit mới ở máy trạm (local repo) lên server (remote repo). Nguồn để đẩy lên là nhánh mà con trỏ HEAD đang trỏ tới (nhánh làm việc).

Lệnh git push cũng có thể xóa đi các nhánh của remote.

Một số tham số hay dùng như:

- all đẩy tất cả các nhánh lên server

- tags đẩy tất cả các tag lên server

- delete xóa một nhánh chỉ ra trên server

- u đẩy và tạo một upstream (luồng upload tương ứng với nhánh của local), hay sử dụng cho lần đầu đẩy lên server

**Git Rebase interactive**

**Git Rebase khi remote có cập nhật mới**

**Git Cherry-pick**

**Tùy chọn force**

**Git reset**

Sử dụng reset để xóa commit : $ git reset --hard HEAD

**8. Các thao tác với git remote**

**Add remote**

Để thêm một remote repo vào local repo thì bạn sử lệnh: $ git remote add [shortname] [url]

Trong đó:

shortname: là tên mà bạn muốn đặt cho remote repo

url: là đường dẫn trỏ đến repo, phần đuôi của URL sẽ là .git

**Rename remote**

Để đổi tên cho remote thì bạn sử dụng lệnh sau : $ git remote rename old\_name new\_name

**Update remote**

**Prune remote branch**

**9. Các thao tác với Stash**

**Tổng quan về git stash**

Lệnh git stash sẽ có tác dụng với tất cả dữ liệu đang hoạt động trong working directory với điều kiện là dữ liệu đó đã được đưa vào trạng thái Staged hoặc đã từng được committed.

**git stash list**

Xem danh sách bao nhiêu lệnh stash đã dùng.

Nếu chúng ta đã stash rất nhiều lần và muốn xem danh sách và địa chỉ của nó thì sử dụng lệnh :

$ git stash list

**git stash clear**

Nếu chúng ta muốn xóa tất cả stash ra khỏi history thì sử dụng tham số clear

Lệnh : $ git stash clear

* Lúc này danh sách stash sẽ rỗng.