GIT

1. Tổng quan

Khi tham gia vào một dự án, việc sử dụng **VCS** (**Version Control System**) rất quan trọng. VCS sẽ giúp quản lý mã nguồn trong suốt quá trình làm việc. Nó theo dõi việc sửa đổi code của các thành viên trong team cùng làm trong cùng project. Có 2 loại VCS hiện nay : VCS tập trung và VCS phân tán.

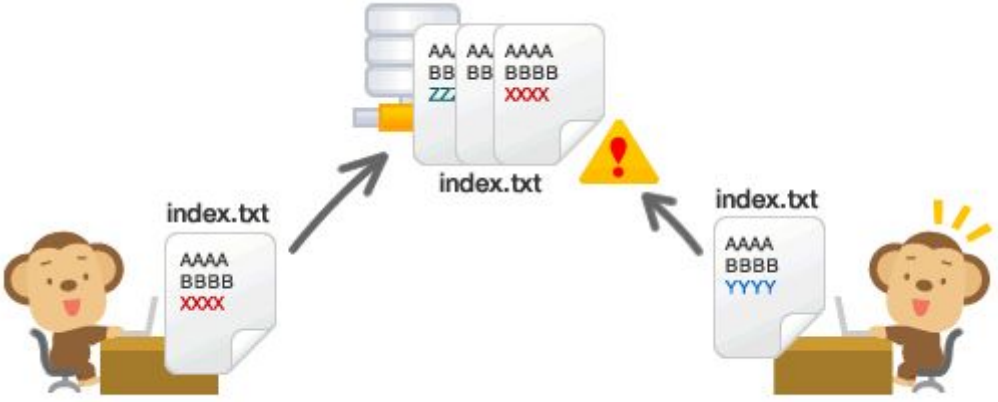
Với VCS tập trung sẽ giúp lưu trữ code ở trong server-side. Mỗi cá nhân khi đóng góp mã nguồn sẽ lấy file từ server và gửi sự thay đổi đến server. Những người này có thể truy cập đến server để làm việc lấy file và thay đổi mã nguồn. **Các hệ thống quản lý phiên bản cục bộ phải đối diện với vấn đề tương tự như thế này mỗi khi toàn bộ lịch sử của dự án được lưu ở một nơi, bạn có nguy cơ mất tất cả.**

VCS phân tán cho phép mỗi thành viên có bản clone code của riêng họ. Song với đó, nó lưu trữ code trong một server riêng biệt, nó cho phép mỗi người làm việc độc lập trên bản clone mà họ có.

Git là một trong những Hệ thống Quản lý Phiên bản Phân tán, vốn được phát triển nhằm quản lý mã nguồn (source code) hữu hiệu của Linux.

Trên Git, có thể lưu trạng thái của file khi có nhu cầu dưới dạng lịch sử cập nhật. Vì thế, có thể đưa file đã chỉnh sửa một lần về trạng thái cũ hay có thể hiển thị sự khác biệt ở nơi chỉnh sửa.

Thêm nữa, khi định ghi đè lên file mới nhất đã chỉnh sửa của người khác bằng file đã chỉnh sửa của mình, dựa trên file cũ, thì khi upload lên server sẽ hiện ra cảnh cáo. Vì thế, sẽ không xảy ra thất bại về việc đã ghi đè lên nội dung chỉnh sửa của người khác mà không hề hay biết.



2. Repository quản lý lịch sử

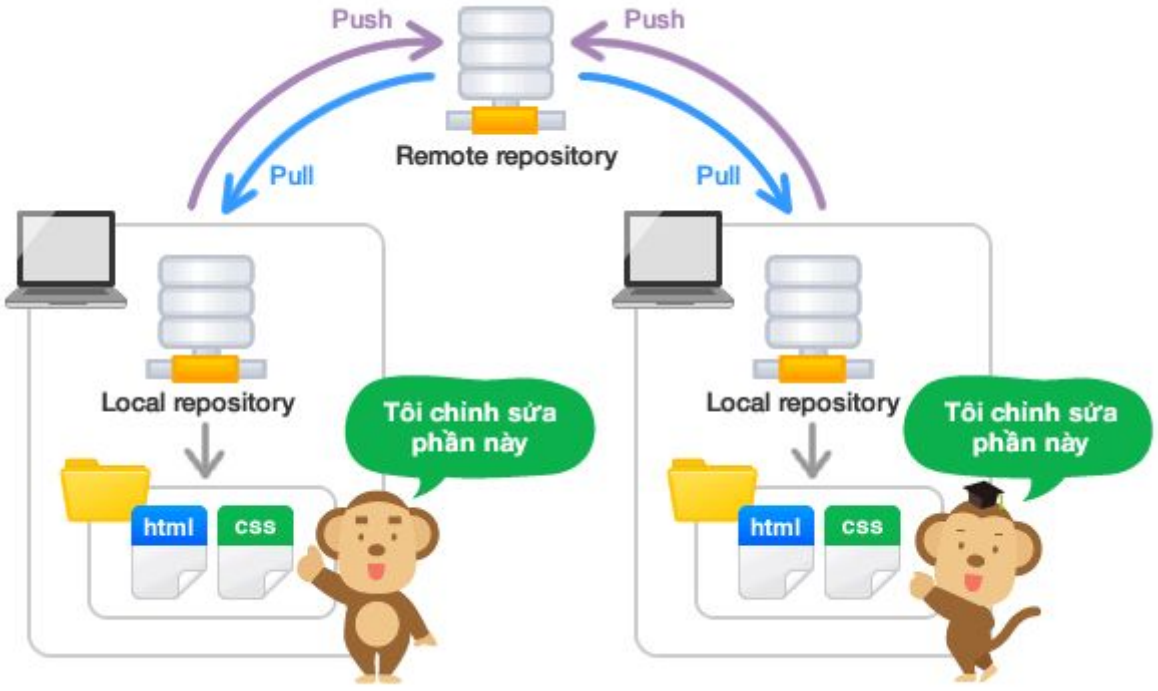
Repository là nơi sẽ ghi lại trạng thái của thư mục và file. Trạng thái được lưu lại đang được chứa như là lịch sử thay đổi của nội dung. Bằng việc đặt thư mục muốn quản lý lịch sử thay đổi dưới sự quản lý của repository, có thể ghi chép lại lịch sử thay đổi của thư mục và file trong thư mục đó.

2.1 Các loại repository

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

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

Do repository phân thành 2 loại là local và remote nên với những công việc bình thường thì có thể sử dụng local repository và thực hiện trên toàn bộ máy sẵn có. Khi muốn công khai nội dung công việc mà bản thân đã làm trên local repository, thì sẽ upload lên remote repository rồi công khai. Thêm nữa, thông qua remote repository cũng có thể lấy về nội dung công việc của người khác.



2.2 Tạo repository

Có 2 cách tạo local repository mà mình có sẵn.

- Cách thứ nhất là tạo repository hoàn toàn mới,
- Cách thứ hai là sao chép remote repository rồi tạo.

Cách 1: Khởi Tạo Một Kho Chứa Từ Thư Mục Cũ
Nếu như bạn muốn theo dõi một dự án cũ trong Git, bạn cần ở trong thư mục của dự án đó và gõ lệnh sau:

```
$ git init
```

Thiết lập remote repository :

```
$ git remote add [tên-máy-chủ] [url]
```

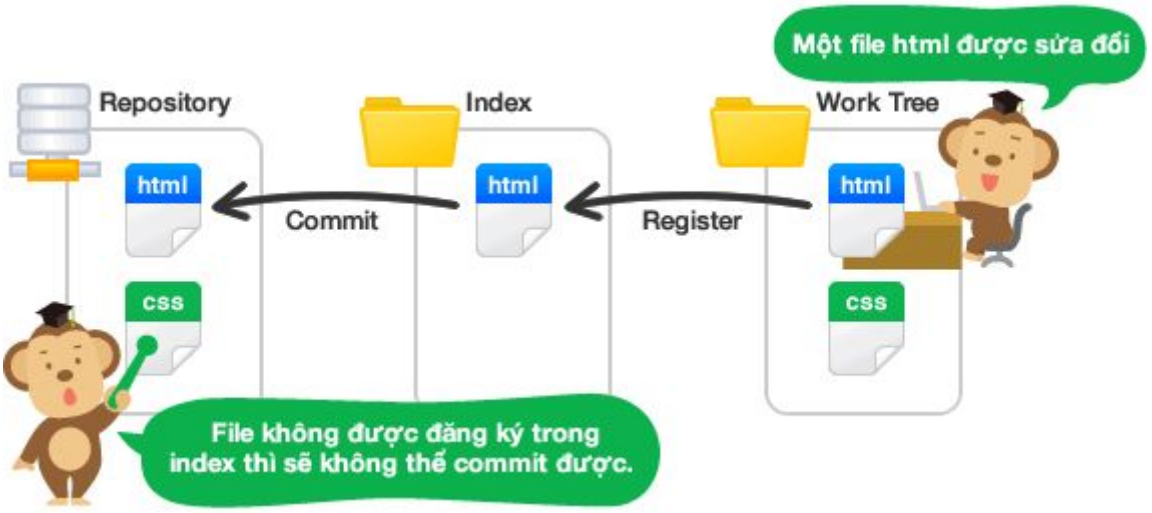
Cách 2: sao chép remote repository rồi tạo.
Nếu như bạn muốn có một bản sao của một kho chứa Git có sẵn - ví dụ như, một dự án mà bạn muốn đóng góp vào dự án và gõ lệnh:

```
$ git clone [url]
```

Trước giao thức git:// có thể sử dụng http(s):// hoặc user@server:/path.git thông qua giao thức ssh.

2.3 Working Tree và Index

Trên Git, những thư mục được đặt trong sự quản lý của Git mà mọi người đang thực hiện công việc được gọi là working tree.
Và trên Git, giữa repository và working tree tồn tại một nơi gọi là index. Index là nơi để chuẩn bị cho việc commit lên repository.



Mỗi tập tin trong working tree có thể ở một trong hai trạng thái : *tracked* hoặc *untracked*. Tập tin *tracked* là các tập tin đã có mặt trong ảnh (snapshot) trước; chúng có thể là *unmodified*, *modified*, hoặc *staged*. Tập tin *untracked* là các tập tin còn lại - bất kỳ tập tin nào trong thư mục làm việc mà không có ở ảnh (lần commit) trước hoặc không ở trong khu vực tổ chức (staging area).

Ban đầu, khi tạo bản sao của một kho chứa, tất cả tập tin ở trạng thái "đã được theo dõi" (tracked) và "chưa thay đổi" (unmodified) vì mới tải chúng về và chưa thực hiện bất kỳ thay đổi nào.

Trên Git, khi đã thực hiện commit thì trạng thái sẽ không được ghi trực tiếp trong repository từ working tree, mà sẽ ghi trạng thái đã được thiết lập của index được xây dựng ở giữa đó, là nơi đăng ký track tập tin. Vì thế, để ghi lại trạng thái của file bằng commit thì trước hết cần đăng ký file trong index.

Với việc chèn index vào giữa như thế này, có thể thực hiện commit không bao gồm những file không cần thiết trong working tree, hay có thể đăng ký chỉ một phần thay đổi của file trong index rồi commit.

2.3.1 Để kiểm tra trạng thái gói tin, chạy lệnh:

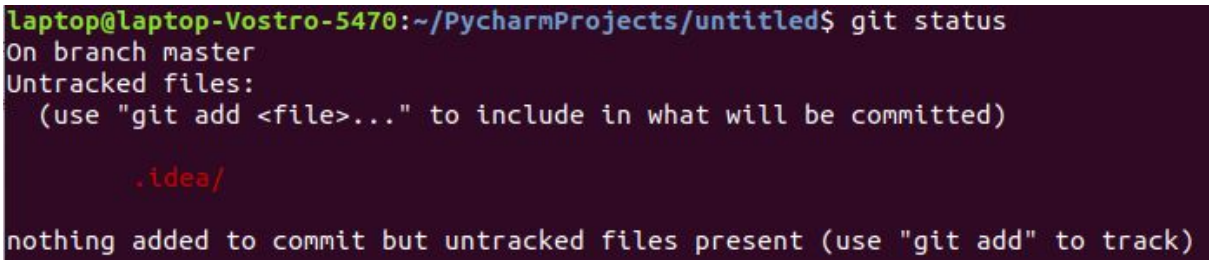
```
$ git status
```

Nếu kết quả như sau:

```
laptop@laptop-Vostro-5470:~/PycharmProjects/untitled$ git status
On branch master
nothing to commit, working tree clean
```

Nghĩa là không có tập tin đang theo dõi nào bị thay đổi. Git cũng không phát hiện ra tập tin chưa được theo dõi nào, nếu không thì chúng đã được liệt kê ra đây. Cuối cùng, lệnh này cho biết đang thao tác trên "nhánh" (branch) nào.

Trường hợp có tệp tin mới, chưa được theo dõi, kết quả:



Có thể thấy là tập tin *.idea/* mới chưa được theo dõi, bởi vì nó nằm trong danh sách "Các tập tin chưa được theo dõi:" (Untracked files) trong thông báo trạng thái được hiển thị, và nó cũng chưa đăng ký vào theo dõi. Git sẽ không tự động thêm nó vào các commit tiếp theo.

Để biết rõ trạng thái thay đổi của repo, dùng lệnh:

```
$ git diff --cached
//xem file thay đổi ntn
// git diff <tên-commit>
```

Hoặc dùng *--staged*

2.3.2 Để đăng ký tập tin vào index, dùng lệnh:

```
$ git add (file)
```

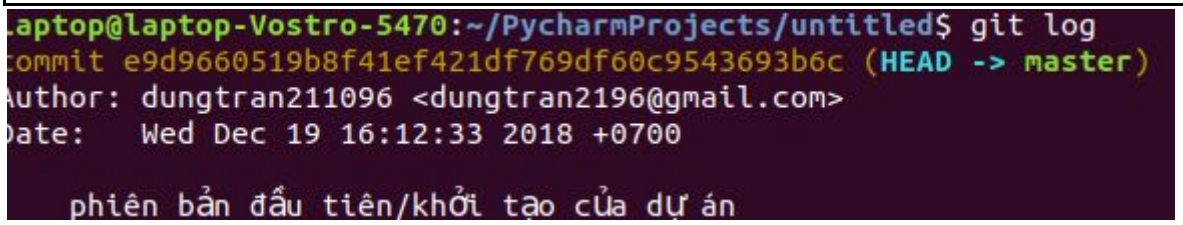
Khi file được thêm vào index và chuẩn bị để commit sẵn sàng cho vào repository, dùng lệnh:

```
$ git commit -m "<commit>"
```

2.3.3 Kiểm tra lịch sử commit

Để hiển thị lịch sử commit thì sử dụng lệnh log.

```
$ git log -p -number
-p : hiển thị cụ thể commit tương ứng với thay đổi trong tập tin
-number: giới hạn số lượng hiện thị commit gần nhất
```



2.3.4 Xoá Tập Tin

Để xoá một tập tin khỏi Git, tập tin đó phải được xóa khỏi danh sách được theo dõi (chính xác hơn, xoá nó khỏi khu vực working tree) và sau đó commit.

```
$ git rm [file]
```

```
laptop@laptop-Vostro-5470:~/PycharmProjects/untitled$ git rm amstrong.c
rm 'amstrong.c'
laptop@laptop-Vostro-5470:~/PycharmProjects/untitled$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    a.out
        deleted:    amstrong.c
```

2.3.5 Bỏ qua các tập tin không cần theo dõi

Thường thì hay có một số loại tập tin mà bạn không muốn Git tự động thêm nó vào hoặc thậm chí hiển thị là không được theo dõi. Những tập tin này thường được tạo ta tự động ví dụ như các tập tin nhật ký (log files) hay các tập được sinh ra khi biên dịch chương trình. Trong những trường hợp như thế, bạn có thể tạo một tập tin liệt kê các "mẫu" (patterns) để tìm những tập tin này có tên .gitignore. Đây là một ví dụ của .gitignore:

```
laptop@laptop-Vostro-5470:~/PycharmProjects/untitled$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        deleted:    a.out
        deleted:    amstrong.c

laptop@laptop-Vostro-5470:~/PycharmProjects/untitled$ vim t.txt
laptop@laptop-Vostro-5470:~/PycharmProjects/untitled$ git add *
The following paths are ignored by one of your .gitignore files:
t.txt
Use -f if you really want to add them.
```

Để thực hiện, tạo một file .gitignore:

```
$ vim .gitignore
# không theo dõi tập tin có đuôi .a
*.a
$ git add .gitignore
```

2.3.6 Phục Hồi

Khi muốn phục hồi (undo) một phần nào đó. Bây giờ, chúng ta sẽ cùng xem xét một số công cụ cơ bản dùng cho việc phục hồi các thay đổi đã thực hiện. Hãy cẩn thận, bởi vì không phải lúc nào bạn cũng có thể làm được điều này. Đây là một trong số ít thuộc thành phần của Git mà bạn có thể mất dữ liệu nếu làm sai.

Thay Đổi Commit Cuối Cùng:

```
$ git commit --amend
```

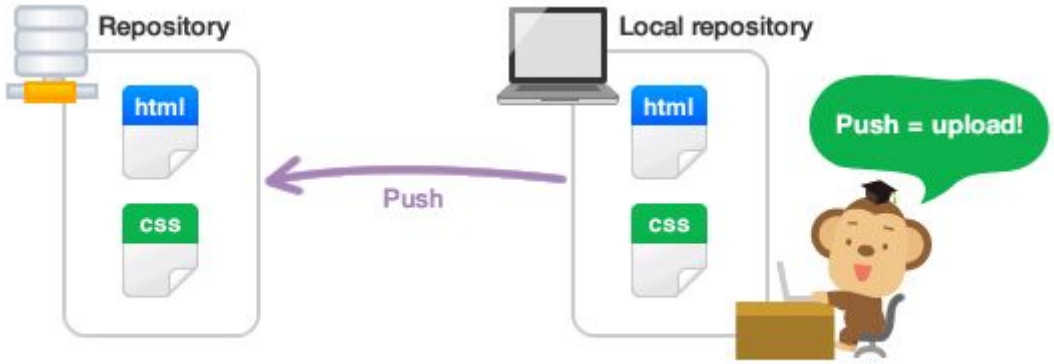
Thay đổi tập tin ở trong index.

2.4 Cập nhật thay đổi

2.4.1 Từ local repository đến máy remote repository:

Để chia sẻ lịch sử thay đổi của local repository mà bản thân đang có bằng remote repository, cần phải upload lịch sử thay đổi trong local repository.

```
$ git push -u [tên-máy-chủ] [tên-nhánh]
```



2.4.2 Nhánh:

Kieerm tra branch

```
$ git branch
```

Tạo branch:

```
$ git checkout -b <branch-name>
```

Xóa branch :

```
$ git branch -d <branch-name> //local
$ git push <remote_name> --delete <branch_name> //remote
```

Tích hợp branch:

```
$ git checkout <branch-name>
$ git merge <branch-cần-merge>
$ git add *
$ git commit -m "để merge"
$ git push -u <tên máy chủ> <tên nhánh>
```

nếu merge 1 file ở nhánh khác:

```
$ git checkout <branch-name> <tên-file>
$ git add <Tên-file>
$ git commit
$ git push -u <tên máy chủ> <tên nhánh>
```