**GITHUP**

**I.Tìm hiểu Sơ về Version Control, Git và Github**

**a. về Version Control:**

-Hệ thống giúp lưu trư mọi thay đổi của source code

-Hỗ trợ nhiều người làm việc cùng lúc

-Xem đứa nào thay đổi code (Rất tiện khi cần tìm thằng đổ tội)

-Revert các thay đổi , đưa code về version cũ, không lo mất code

**b. cơ chế hoạt động:**

**1.Repository(kho lưu trữ)**: Repository hay còn gọi là Repo, dịch ra tiếng Việt có nghĩa là kho, đây chính là nơi chứa tất cả mã nguồn cho một dự án được quản lý bởi Git. Bạn cũng có thể hiểu một cách khác là Repository chính khai báo thư mục chứa dự án của bạn trên local hoặc remote. Môi repo sẽ có hai cấu trúc dữ liệu chính đó là Object store và Index được lưu trữ ẩn trong thư mục

- Có hại loại repository đó là **local repository** và **remote repository**.

* **Local repository**: Là repo được cài đặt trên máy tính của lập trình viên, repo này sẽ đồng bộ hóa với remote repo bằng các lệnh của git.
* **Remote repositorytr21`/**: Là repo được cài đặt trên server chuyên dụng, điển hình hiện nay là Github.

**2.Snapshots(ảnh chụp nhanh):**

-Ảnh chụp nhanh là trạng thái của một cái gì đó (ví dụ như một thư mục) tại một thời điểm cụ thể. Trong trường hợp này, ảnh chụp nhanh có nghĩa là nội dung hiện tại của nhánh thử nghiệm, đây không phải là bản sửa đổi chính.

-là cách mà git nó theo giõi source code , history, nó là 1 bản lưu trữ của tất cả các hành động tác động đến source code của chúng ta tại thời điểm nào đó

-mỗi lần commit thì nó xẽ tạo ra 1 snapshot

**3.Commit:**

-Để ghi lại việc thêm/ thay đổi file hay thư mục vào repository thì sẽ thực hiện thao tác gọi là Commit.

- Khi thực hiện commit, trong repository sẽ tạo ra commit (hoặc revision) đã ghi lại sự khác biệt từ trạng thái đã commit lần trước đến trạng thái hiện tại.

**4.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 repository.

- Khi tiến hành commit lần đầu trong repository thì Git sẽ tạo ra một branch có tên là master. Vì thế những lần commit sau sẽ được thêm vào branch master cho đến khi chuyển đổi branch.

**2.Cài đặt Git trên máy:**

**-**Git là tên gọi của một Hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia. Trong Git, thư mục làm việc trên máy tính gọi là Working Tree.

**a.INSTALL Git:**

-Window: <http://git-scm.com/download/win>

**b.Một số lệnh cơ bản:**

-git init : tạo ra 1 repository(kho lưu trữ) githup

-git clone: lấy 1 cái repository có sẵn và đưa về máy của mình

-git pull: lệnh được sử dụng để lấy và tải nội dung từ một kho lưu trữ từ xa và ngay lập tức cập nhật kho địa phương cho phù hợp với nội dung đó

-git add và git commit:

+Trước khi commit thì bạn phải sử dụng lệnh git add <file\_name> để khai báo là sẽ commit file này, trường hợp bạn muốn commit tất cả các file có thay đổi thì sử dụng lệnh git commit ..

-git push: đưa những thay đổi từ bản local sang bản Repo(reponsie)kho lưu trữ

**II/.Thiết lập lần đầu:**

**1.thiết lập danh tính:**

-git config –global user.name “sang thai”

-git config –global user.email [thaibasang8@gmail.com](mailto:thaibasang8@gmail.com)

**2.sao chép một kho chứa đã tồn tại:**

-$ git clone <https://github.com/user/repository.git>

**3.cập nhật thay đổi lên Local Repository:**

-$ git diff: xem những thay đổi của code trước khi đẩy lên Local Reposition

-$ git add : Sau khi bạn thay đổi source code: thêm mới, sửa, xoá files,… Bạn cần phải cập nhật lên Staging Area. git add. Là add toàn bộ hoặc git add text.txt (text.txt là tên file cần add)

-$ git commit -m "Message": Sau lệnh add, bạn cần sử dụng câu lệnh Commit để đây thông tin thay đổi lên Local Respository: “Message là chú thích thay đổi”

**4.cập nhật lên sever**:

-$ git push : Sau câu lệnh Commit, thông tin mới chỉ được cập nhật lên Local Repository. Nếu muốn cập nhật ên server thì bạn phải sử dụng câu lệnh push:

**5.Kiểm tra trạng thái** :

-$ git status : hiển thị trạng thái cây làm việc

**6.Cập nhật những thay đổi mới nhất của Remote(Git từ xa)**

-$ git remote -v : liệt kê các kết nối từ xa mà bạn có đến các kho lưu trữ khác -v (bao gồm hiển thị cả url)

-$ git pull : lấy những thay đổi mới nhất trên Remote để bỏ về Local

**7.Xem lịch sử commit ( git log)** :

-$ git log: liệt kê các commit được thực hiện trong kho chứa đó theo thứ tự thời gian. Đó là, commit mới nhất được hiển thị đầu tiên. Như bạn có thể thấy, lệnh này liệt kê từng commit với mã băm SHA-1, tên người commit, địa chỉ email, ngày lưu, và thông điệp của chúng.

**III/ Đưa Thư Mục Code có sẵn trong máy Lên Repository Remote:**

* echo "# App\_nhac" >> README.md : Taoj 1 file readme
* git init : tạo ra 1 repository(kho lưu trữ) githup
* git add . :add tất cả các file vào Staging Area
* git commit –m “project land au tien” : Sau lệnh add, bạn cần sử dụng câu lệnh Commit để đây thông tin thay đổi lên Local Respository: “Message là chú thích thay đổi”
* git remote add origin <https://github.com/conchonha/App_nhac.git> : Tạo liên kết từ xa để Local Repository làm việc với Remote Repository
* git push -u origin master : nó xẽ buff cái nhánh master của mình lên cái master của origin của thư mục git trên sever

**IV/ Git Nâng Cao:**

1.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 repository.

- Hơn nữa, branch đã phân nhánh có thể chỉnh sửa tổng hợp lại thành 1 branch bằng việc hợp lại (merge) với branch khác.

- Khi tiến hành commit lần đầu trong repository thì Git sẽ tạo ra một branch có tên là master. Vì thế những lần commit sau sẽ được thêm vào branch master cho đến khi chuyển đổi branch.

## *a. Branch tích hợp (Integration branch)*

## -Branch tích hợp là branch có thể tạo ra bản phát hành bất cứ khi nào. Hơn nữa, nó cũng được sử dụng như là nguồn phân branch của branch chủ đề. Vì thế, việc duy trì trạng thái ổn định là điều cần thiết

## *b. Branch chủ đề (Topic branch)*

## -Branch chủ đề là branch tạo ra nhằm tiến hành công việc liên quan đến chủ đề như là chỉnh sửa lỗi hay là thêm chức năng. Khi tiến hành cùng lúc những công việc có liên quan đến nhiều chủ đề thì số lượng branch chủ đề tương ứng sẽ được tạo ra.

# C. ***Chuyển đổi branches***

## -Để chuyển đổi branch làm việc thì sẽ thực hiện thao tác gọi là checkout.

## *d. HEAD*

## -HEAD là tên hiển thị phần đầu của branch đang sử dụng hiện tại. Mặc định là đang hiển thị phần đầu của master. Bằng việc di chuyển HEAD thì branch đang sử dụng sẽ được thay đổi.

## *e. Stash*

## Khi những file được thêm mới hay nội dung thay đổi vẫn chưa commit vẫn còn lưu lại Index và Work tree, mà thực hiện checkout đến branch khác thì nội dung thay đổi đó sẽ di chuyển từ branch ban đầu đến branch chuyển đến.

Tuy nhiên ở branch di chuyển đến, trường hợp có file giống như vậy đã được tiến hành thay đổi cái gì đó thì checkout sẽ thất bại. Khi xảy ra trường hợp này thì sẽ commit lại nội dung thay đổi 1 lần nữa hoặc là sử dụng stash để lưu tạm thời nội dung thay đổi, sau đó phải thực hiện commit.

Stash là khu vực ghi lại tạm thời nội dung thay đổi của file. Bằng việc sử dụng stash, trong work tree và index, những thay đổi chưa được commit có thể lưu lại tạm thời.

Những thay đổi lưu tạm này về sau có thể lấy ra và hiển thị trên branch ban đầu hay là phản ánh lên branch khác.

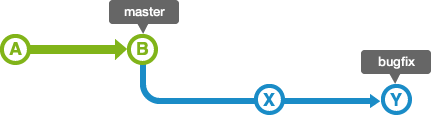
# ***F. Tích hợp branches***

Branch chủ đề đã hoàn thành công việc, cuối cùng sẽ được tích hợp vào branch tích hợp. Việc tích hợp branch thì có hai phương pháp, phương pháp sử dụng merge và phương pháp sử dụng rebase. Dựa vào việc sử dụng phương pháp nào mà lịch sử của branch sau khi tích hợp sẽ có sự khác biệt lớn.

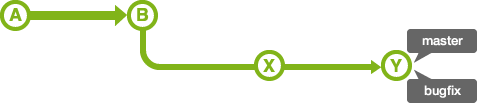
## *Merge*

Khi sử dụng merge, có thể tổng hợp nhiều luồng lịch sử.

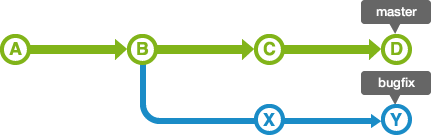
Ví dụ, có branch bugfix phân nhánh ra từ branch master như sơ đồ bên dưới.



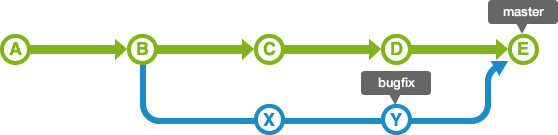
Khi merge branch bugfix này vào branch master, nếu trạng thái branch master không có gì thay đổi thì có thể thực hiện merge hết sức đơn giản. Vì lịch sử của branch bugfix sẽ bao gồm tất cả lịch sử của branch master, nên branch master chỉ có việc di chuyển đơn giản là có thể lấy được nội dung của branch bugfix. Và, người ta gọi merge như thế này là merge fast-forward (chuyển tiếp nhanh).



Nhưng, cũng có trường hợp lịch sử của branch master vẫn đang tiến triển sau khi phân branch bugfix. Trường hợp này thì cần thiết phải tổng hợp nội dung thay đổi của branch master và nội dung thay đổi của branch bugfix thành một.

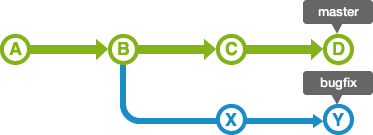


Vì thế, merge commit đã lấy thay đổi của cả hai branch sẽ được tạo ra. Đầu branch master sẽ di chuyển đến commit đó.

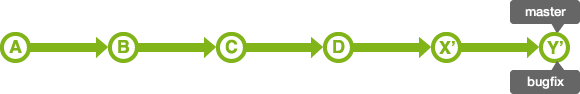


## *Rebase*

Giống với ví dụ merge, sẽ có branch bugfix phân nhánh từ branch master như sơ đồ bên dưới.

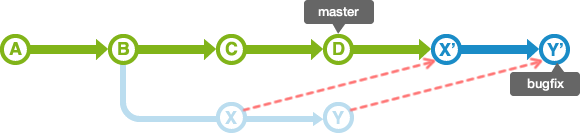


Trường hợp sử dụng rebase ở đây rồi tiến hành tích hợp branch thì lịch sử sẽ giống như sơ đồ tiếp theo. Bây giờ, sẽ trình bày đơn giản trình tự rebase như thế nào.



Trước hết, khi rebase branch bugfix vào branch master, lịch sử branch bugfix sẽ được thay đổi đính kèm sau branch master. Cho nên, lịch sử sẽ thành 1 đường như trong sơ đồ.

Khi này sẽ có trường hợp phát sinh xung đột tại commit di chuyển X và Y. Lúc đó tại từng commit cần thiết phải chỉnh sửa lại những chỗ phát sinh xung đột.



Nếu chỉ rebase không thôi thì vị trí đầu master vẫn cứ như vậy. Vì thế, merge branch bugfix từ branch master rồi di chuyển master đến phần đầu của bugfix.



2.thực hành:

-git reflog : hiển thị tất cả các commit

-start test.txt :mở file bằng lệnh cmd

-type> “ten file” : tao moi 1 file bằng lệnh cmd

-git checkout -b “nhanh moi” : tạo branch mới

-git checkout master : chuyển hướng đến nhánh master

-git merge branchmoi : gộp nhánh

- git reset --soft HEAD~1 : bạn sẽ xóa cam kết cuối cùng khỏi nhánh hiện tại, nhưng các thay đổi tệp sẽ ở trong [cây làm việc](https://mijingo.com/blog/what-is-the-working-tree-in-git) của bạn .

- git reset --keep HEAD~1là một trong những thú vị và hữu ích. Nó chỉ đặt lại các tệp khác nhau giữa cam kết **hiện tại** HEAD và cam kết đã cho. Nó hủy bỏ thiết lập lại nếu bất kỳ ai trong số các tệp này có các thay đổi không được cam kết

- Khi sử dụng, git reset --hard HEAD~1bạn sẽ **mất tất cả các thay đổi không được cam kết** ngoài các thay đổi được giới thiệu trong lần xác nhận cuối cùng. Các thay đổi sẽ không ở trong cây làm việc của bạn, vì vậy thực hiện một git statuslệnh sẽ cho bạn biết rằng bạn không có bất kỳ thay đổi nào trong kho lưu trữ của mình.

**3.Cơ bản: Tạo Githup Repository & Clone**

**+Pull & Push**

**+Cho các dự án đơn giản**

**4.Nâng cao: Branch, merge**

**+Conflict Resolving, Pull Request**

**+ Cho các dự án lớn hoặc đi làm**