# 1.Khởi tạo Git cho một project đã tồn tại

Để khởi tạo Git cho một project, đầu tiên ta đi vào thư mục gốc project đó.

Mở cmd hoặc terminal để thao tác:

cd /simple-calculator

Sau khi đã ở trong thư mục project :

git init

Lúc này git sẽ tự động tạo cho chúng ta thư mục .git để lưu thông tin.

# ****2. Kết nối Local repository với remote repository****

Để kết nối local repo với remote repo:

git remote add origin https://github.com/nghuuquyen/simple-calculator.git

trong đó origin là tên đặt cho remote repo, origin là tên chuẩn được sử dụng nhiều.

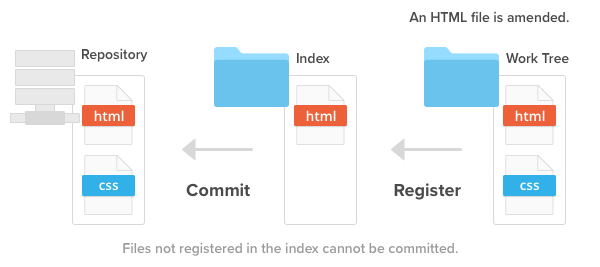
# 3. Clone a reponsitory

Để clone một reponsitory trên remote server : Git clone <remoteUrl>

# 4.Workflow

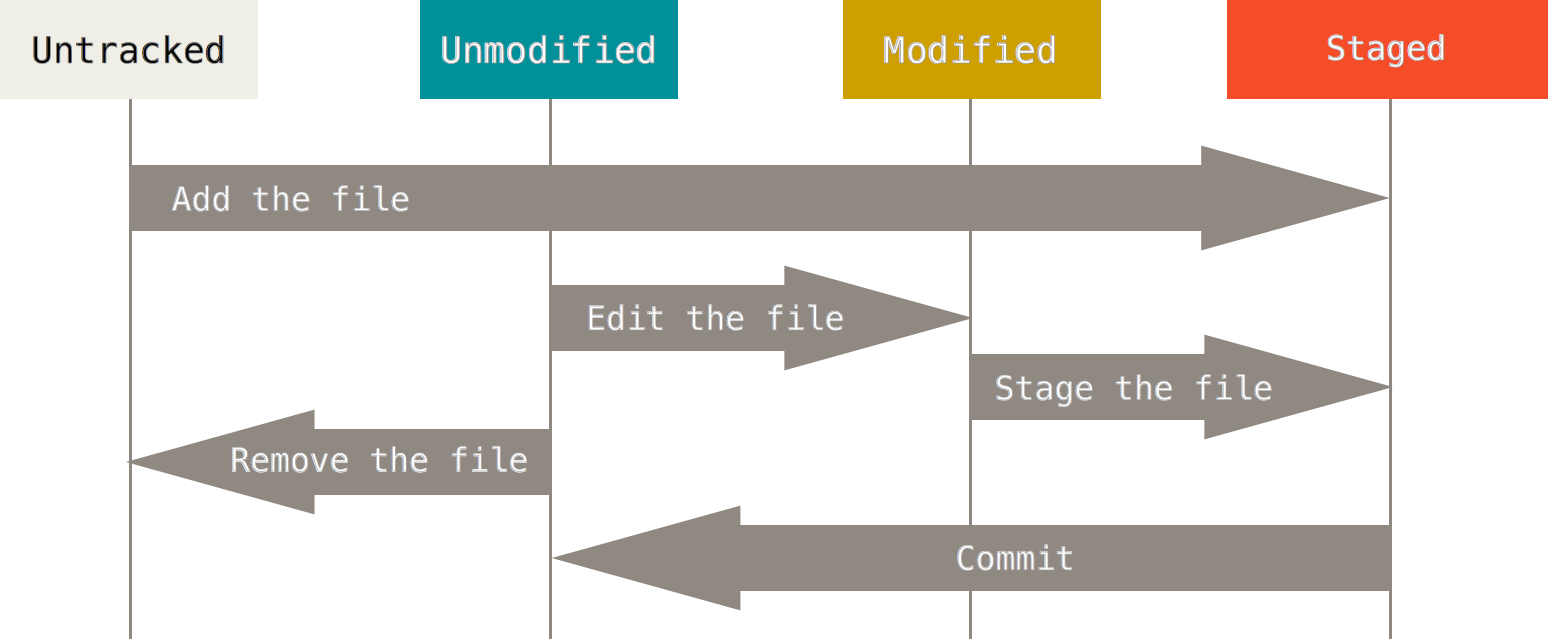
Có 3 thành phần chính của một git project:

* working directory(menu tree) có chứa các tập tin hiện tại.
* Index hay staging area là khu vực sẵn sang để commit.
* HEAD trỏ đến commit gần nhất.



**Three states of Git files.**

1. **Modified**
2. **Staged**
3. **Committed**



# ****Add and commit****

**Để add tập tin vào staging area(index) sử dụng**

**Git add <file name> hoặc git add .(dấu chấm)**

**Để commit tập tin vào HEAD sử dụng :**

**Git commit –m ‘commit message’**

**Hiện tại tập tin đã được commit vào HEAD ,nhưng chưa được update lên remote repo.**

**Khi chúng ta đã commit tuy nhiên do cần chỉnh sửa thêm mà không muốn tạo commit mới**

**Chúng ta sẽ chỉnh sửa các file rồi sử dụng git add,sau đó sẽ sử dụng commit –amend.**

$ **git commit --amend**

**Khi sử dụng commit –amend thì chỉnh sửa sẽ được update vào commit cuối cùng mà không tạo ra commit mới.**

# ****Pushing Changes.****

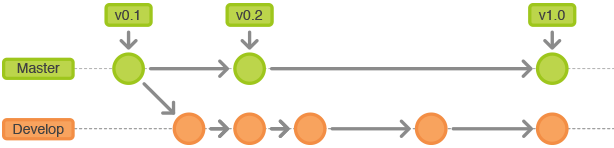
**Để send những thay đổi trong HEAD lên remote repo ,sử dụng:**

**Git push origin <branch\_name>**

**Khi muốn ghi đè local lên remote repo thì sử dụng : git push -force**

# ****GIT Branch****

**Branch dung để phân nhánh và ghi lại luồng của lịch sử.Branch đã phân nhánh không gây ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng một repo.**



* 1. **HEAD Con trỏ vị trí**

**HEAD tượng trưng cho commit hiện tại đang được trỏ đến, về căn bản nó là commit bạn đang làm việc.**

**HEAD luôn trỏ đến commit gần nhất.Bình thường HEAD trỏ tới tên nhánh (ví dụ fixBug).Khi bạn commit thì trạng thái của fixBug thay đổi và thay đổi này thông qua HEAD.**

**Git checkout <commit\_id> : đưa HEAD đến commit\_id**

**Vd:**

**Git checkout master^: dịch chuyển HEAD tới cha đầu tiên của master.**

**Git checkout HEAD~2: dịch chuyển HEAD 2 lần.**

**Gỉa sử có log của commit như sau:**

$ git log --oneline

272519d The last commit

ce81389 Commit one more time

6586599 Commit second time

64c24fc Commit message first time

Hiện tại HEAD của bạn đang ở commit mới nhất là 272519d,giả sử đang sảy ra lỗi và bạn muốn xem commit ce81389 có phải là commit gây ra lỗi không.

$ git checkout ce81389

Và HEAD lúc này trỏ đến commit ce81389 ,đây được gọi là trạng thái detached HEAD.Khi ở trạng thái này nếu bạn tiếp tục chỉnh sửa thì sẽ không lưu lại được,do vậy checkout sang branch mới tại đây để chỉnh sửa hoặc quay trỏ lại vị trí HEAD đầu tiên để chỉnh sửa.

Để kiểm tra HEAD đang trỏ tới đâu : git log hoặc cat .git/HEAD.

* 1. **Tạo branch**

**$ git branch develop**

* 1. **Checkout branch**

**$ git checkout develop**

* 1. **Tạo branch và checkout**

**$ git checkout -b feature\_x <commit\_id>**

**7.5. Push branch tới remote repo**

**$ git push origin <branch\_x>**

**7.6. Xóa branch**

**$** **git branch -d develop**

Deleted branch develop (was 8c68896).

Để khôi phục branch sau khi xóa :

1. Tìm commit gần nhất có lịch sử liên quan đến branch bị xóa

$ git reflog

1. Khôi phục branch đến commit đã tìm được

$ git checkout -b old-master cbc2046

# ****8 GIT PULL****

**Sử dụng để update thông tin mới từ remote repo xuống local repo :**

**Git pull = git fetch + git merge.**

**Khi pull thì những thông tin mới từ remote sẽ được update vào local luôn.**

# ****9 GIT FETCH****

**Sử dụng để lấy thông tin tử remote repo xuống git repo,nhưng chưa update vào local file.**

**Để tích hợp nội dung xuống local thì cần merge hoặc pull lại.**

# ****10 Git MERGE****

**Sử dụng để tích hợp nhánh develop vào nhánh master hoặc ngược lại.**

1. **Di chuyển đến branch cần được merge**

**$ git checkout master**

1. **Chạy lệnh merge**

**$ git merge develop**

**Trường hợp sảy ra confic thì sẽ phải resolve conflict,sau đó add và commit lên remote repo**

git add .

git commit -m "Merge branch 'dev'"

**\*Trường hợp bị confic nhưng chưa thể giải quyết ngay được , cần quay lại trạng thái ban đầu thì sử dụng :** git merge –abort(khuyên dùng) hoặc $ git reset --hard ORIG\_HEAD.

# 11 GIT REBASE

**Sử dụng để tích hợp nhánh develop vào nhánh master hoặc ngược lại.**

1. **Merge branch issue3 vào branch master**
2. $ **git checkout issue3**
3. Switched to branch 'issue3'
4. $ **git rebase master**

**Chỉnh sửa conflict xong thực hiện**

$ **git add myfile.txt**

$ **git rebase --continue**

**Lúc này các commit mới của issue3 đã được đưa lên trước các commit của master(lúc này master đã được merger vào issue3).Tuy nhiên master chưa được cập nhật trạng thái ngay.Chúng ta cần thực hiện merge branch issue3 vào master**

$ **git checkout master**

Switched to branch 'master'

$ **git merge issue3**

**\*Nếu quá trình sửa conflict chưa thực hiện được thì có thể quay lại trạng thái trước rebase** $ **git rebase –**abort

# ****12 GIT Revert****

**Sử dụng để hoàn tác thay đổi của một commit.**

$ **git revert <commit\_id>**

**Git revert sẽ tạo ra một commit mới ,chính là commit chứa các thay đổi của commit được revert.**

# ****13 GIT Reset****

$ git reset <commit\_id>

Thực hiện xóa các commit mới hơn commit\_id,đưa branch về commit\_id

git reset [--soft | --mixed | --hard | --merge | --keep] <commit>

$ **git reset --hard <commit\_id>**

git reset [<mode>] [<commit>] : default mode là --mixed

git reset -–soft [<commit>] : Sẽ không thay đổi index file hoặc working tree.Các file sẽ vẫn ở trong staging area.

git reset -–mixed [<commit>] : Thực hiện reset index file nhưng không thay đổi working tree.Các file đã thay đổi được giữ nguyên nhưng không nằm trong staging area.

git reset -–hard [<commit>]: Thực hiện reset index file và working tree.Tất cả các file tracked trong working tree từ <commit> trở đi đều bị discard.

Nếu muốn kết quả reset về được update trên remote responitory thì cần dung

Git push –force .Lúc này các commit khác đều bị ghi đè nên cân nhắc kỹ trước khi dung **(sử** dụngpush --force-with-lease**)**

 Giả sử repository của chúng ta có các commit như sau:

A - B - C (HEAD -> master)

Khi đó:

* git reset --soft B: Sẽ đưa HEAD về vị trí commit B, giữ nguyên các thay đổi đã được git add tại B trong **Staging Area**. Tương ứng với tại B ta vừa git add xong, khi này chỉ việc git commit là ta lại có commit như C bên trên.
* git reset B: Sẽ đưa HEAD về vị trí commit B, đồng thời đưa các thay đổi về **Working directory**. Tương ứng với tại B ta vừa cột xong, chưa git add.
* git reset --hard B: Sẽ đưa HEAD về vị trí commit B, đồng thời loại bỏ tất cả các thay đổi tại B. Tương ứng với tại B ta chưa cột gì cả, xong phim.

## **Vài điều về HEAD**

Đôi khi bạn thấy lệnh **git reset HEAD~1** thì nó cũng tương đương với lệnh **git reset B** như trên, vì HEAD~1 chính là lùi HEAD lại 1 commit.

* HEAD chính là commit hiện tại ta đang đứng.
* HEAD~1 là commit ngay trước HEAD trên cùng nhánh
* HEAD~n là commit trước HEAD n commit trên cùng nhánh
* Có thể dùng @ thay cho HEAD: git log @
* HEAD~ != HEAD^ (trông có vẻ giống nhưng khác ý nghĩa)

Commit trước khi reset có thể tham chiếu bằng tên ORIG\_HEAD. Trường hợp đã reset nhầm chẳng hạn, nếu reset ở trong ORIG\_HEAD thì có thể quay về trạng thái trước khi reset.

$ **git reset --hard ORIG\_HEAD**

# ****14 Git Cherry-pick****

**Apply một số commit trên branch khác vào branch hiện tại.Hay copy commit xuống dưới vị trí hiện tại của HEAD.**

$ **git checkout master**

Switched to branch 'master'

$ **git cherry-pick 99daed2 989189**

**ở trên đang lấy 2 commit của branch khác đưa vào branch master.**

**\*Để quay lại trạng thái trước khi cherry-pick :** $ **git cherry-pick –abort**

# 15.Các Chú ý quan trọng

## **15.1 Xem reflog để khôi phục lại 1 commit bất kì**

Reflog như cỗ máy thời gian, nó ghi lại toàn bộ những gì bạn đã làm, kể cả khi bạn xóa 1 commit nào đó.

0979a9e HEAD@{1}: reset: moving to HEAD~

4d77eb9 HEAD@{2}: commit: Add UI for data showcase.

0979a9e HEAD@{3}: commit: Finish UAF module, need update xls file normaly

4d3afe9 HEAD@{8}: commit: Finish UFM Module

c1fe83d HEAD@{9}: commit: Add faculty to model

b5d314a HEAD@{10}: commit: Add faculty\_code

c528ae5 HEAD@{11}: pull: Fast-forward

0979a9e hoặc HEAD@{3} là id của commit Finish UAF module, need update xls file normaly Để khôi phục lại commit đó

git reset --hard 0979a9e

HEAD is now at 0979a9e Finish UAF module, need update xls file normaly

# Đầu tiên là xem lại toàn bộ lịch sử commit

$ git reflog

# Từ đó chọn commit muốn phục hồi và khôi phục lại

# ví dụ）git reset --hard HEAD@{2}

$ git reset --hard <commit>

## **15.2 Lỡ tay reset —hard**

Tương tự như trên, ta xem lại reflog

0979a9e HEAD@{1}: reset: moving to HEAD~

4d77eb9 HEAD@{2}: commit: Add UI for data showcase.

0979a9e HEAD@{3}: commit: Finish UAF module, need update xls file normaly

237e662 HEAD@{4}: commit: Add noti token update in UFM module

0979a9e là commit reset HEAD, 4d77eb9 là commit bị mất bạn cần sẽ nhảy đến. Để khôi phục

git reset --hard 0979a9e

HEAD is now at 0979a9e Finish UAF module, need update xls file normaly

## **15.3 Khi lỡ tay xoá mất branch và muốn lấy lại**

# Đầu tiên là xem lại toàn bộ lịch sử commit

$ git reflog

# Từ các commit này, chọn rồi tạo branch mới

# ví dụ）git branch new-branch HEAD@{2}

$ git branch <tên branch> <commit>

## **15.4 Khi pull từ remote về thì có quá nhiều conflict nên tạm thời muốn trở lại như cũ**

# Lấy từ code mới remote

$ git pull origin master

# Phát sinh conflict

# Suy nghĩ lại thì trong pull(fetch + merge) muốn bỏ phần merge đi

$ git reset --hard ORIG\_HEAD

## **15.5 Lỡ tay push force lên master trên remote**

Trường hợp chỉ cần push về 1 commit trước đó

# Cái này ko được làm đâu nhé, push force

$ git commit -m "commit message"

$ git push -f origin master

# Nếu là một công ty nghiêm túc thì đến đây chắc chắn sẽ nổi giận đấy

# Nếu muốn đưa về 1 commit tước đó thì làm như sau

$ git push -f origin HEAD~:master