# Hướng dẫn sử dụng github cơ bản

## Git là gì?

Git chính là một hệ thống VCS cho dự án của bạn. Chính vì là một VCS ở trên nên Git cũng ghi nhớ lại toàn bộ lịch sử thay đổi của source code trong dự án. Bạn sửa file nào, thêm dòng code nào, xóa dòng code nào, bỏ thừa dấu ở đâu .etc. Mọi thứ đều được Git ghi lại.

Điểm khác biệt của Git so với các VCS khác (như Subversion...) chính là "cách Git nghĩ về dữ liệu". Phần lớn các hệ thống khác coi thông tin được lưu trữ như là một tập hợp các file kèm theo các thay đổi được thực hiện trên mỗi file theo thời gian, như hình bên dưới.



Tuy nhiên ở Git, lại coi thông tin được lưu trữ là một tập hợp các snapshot (ảnh chụp toàn bộ nội dung tất cả các file tại thời điểm). Mỗi khi bạn "commit", Git sẽ "chụp" và tạo ra một snapshot cùng một tham chiếu tới snapshot đó, như này:



Đó là cách Git nghĩ về dữ liệu. Để cài đặt Git, vui lòng truy cập [trang Download của Git](https://git-scm.com/downloads) để tải về và làm theo hướng dẫn cài đặt.

## Sử dụng Git như thế nào?

Sau khi cài đặt, Git được sử dụng thông qua cửa sổ dòng lệnh với terminal (command-line). Ví dụ như này:

git init

git add .

git commit -m "Save my first changed"

## Repository

### Repository là gì?

Trong Git, Repository là một kho chứa, lưu trữ source code của bạn. Có hai loại repository, mình tạm gọi là:

**Local repository**: Là repository được lưu tại một máy tính không. Bạn có thể thêm, sửa, xóa file, tạo ***"commmit"*** để lưu lại nhưng chưa thể dùng để chia sẻ tới người khác được. Để tạo local repository, đơn giản tại folder chứa source, sau đó bạn thực hiện command line:

# Create local repository:

> git init

**Server repository**: Là repository nhưng được lưu tại server của các hosting-service sử dụng Git. Một số hosting-service các bạn có thể biết như: [Github](https://github.com/), [Gitlab](https://gitlab.com/), [Bitbucket](https://bitbucket.org/)... Repository này có thể dùng để share tới những người khác để họ có quyền truy cập và lấy source code về. Để tạo server repository, bạn đơn giản truy cập vào trang web của hosting-service bạn dùng và thực hiện qua vài cú click tùy nhà cung cấp dịch vụ.

### Làm việc với repository

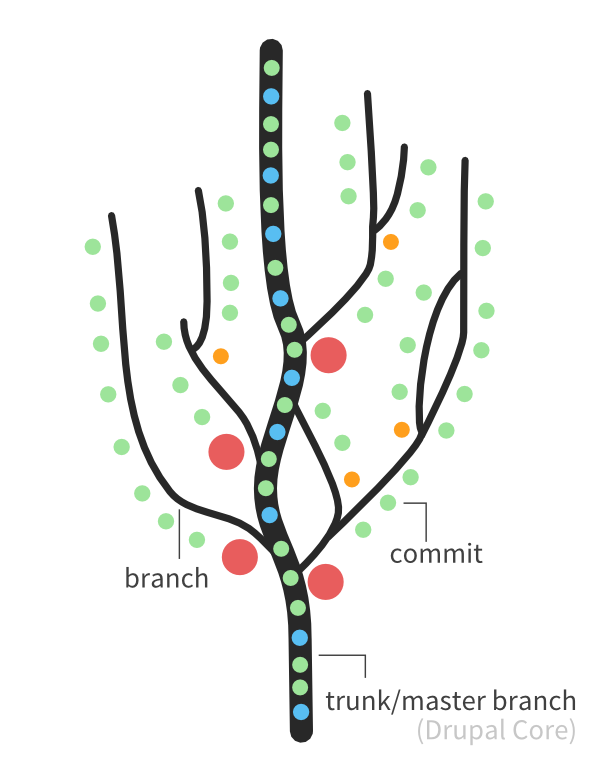
Tạo một repository trên máy, ta chỉ cần tạo một folder cho nó, rồi thực hiện câu lệnh:

> git init

Tạo một local repository từ server repository:

> git clone repository\_url

## Commit



### Commit là gì?

Trên mỗi ***branch*** bạn làm việc, sau khi sửa đổi các file source code... Những thay đổi đấy cần **được lưu lại** bằng cách tạo ra một điểm mốc đánh dấu. Điểm đánh dấu các thay đổi này gọi là **Commit**. Tại mỗi commit, git chụp lại toàn bộ dữ liệu, tạo ra một snapshot version hóa dữ liệu.

Mỗi commit được tạo ra gồm một số thông tin chính:

Tên và email của ngưởi tạo commit

Ngày giờ commit được tạo ra

Message: Mô tả cho sự các sửa đổi (Để sau này xem lại còn biết nó sửa vì mục đích gì)

Một ID là một chuỗi SHA gồm 2 dạng: Full và Short.

Full: Chuỗi SHA đầy đủ

Short: Là 7 ký tự đầu tiên được cắt ra từ chuỗi Full

### Làm việc với commit

#### Tạo một commit lưu lại các thay sửa đổi.

# Cú pháp: git commit -m <message>

> git commit -m "Message mô tả thay đổi"

# Lưu lại thay đổi nhưng gì đè lên commit trước đó:

> git commit --amend -m "Message mô tả thay đổi"

# Ghi đè commit trước đó và đổi người tạo commit:

# git commit --amend --author "Your name <your email>" # Chú ý ghi cả dấu < > trong case này.

> git commit --amend --author "Tran Tan Loc <ttanloc37@gmail.com>"

#### Xem lại danh sách các commit đã tạo gần đây:

# Cú pháp: git log -<n>

> git log -5

# Hiển thị mỗi commit trên một dòng

> git log --oneline -5

So sánh cá thay đổi của code trước khi add thay đổi vào commit

> git diff

#### Tạo commit vào message trên nhiều dòng (mặc định chỉ trên một dòng), sử dụng dấu ' thay thế " khi tạo message:

> git commit -m '- First line

> - Second line'

#### Gộp nhiều commit làm 1 commit.

VD: Gộp 3 commit gần nhất làm 1.

Thực hiện command:

> git rebase -i HEAD~3

Một file mới hiện ra với danh sách 3 commit gần nhất.

pick fd3fea3 Three

pick fd3fea2 Two

pick fd3fea1 One

Dòng commit đầu giữ nguyên, hai dòng thứ 2, 3 sửa pick thành s để gộp. Sửa pick thành d để xóa commit tại dòng đó. Ctrl+O, Enter để lưu lại.

Sửa lại commit message, chèn dấu # vào đầu dòng các commit message để loại bỏ message đó. Để lại một message cho commit mới sau khi gộp. Ctrl+O, Enter lưu lại là xong.

## Resolve conflict sau khi tạo merge request. Checkout về branch mà bạn dùng để gửi merge\_request.

#### Fetch mới lại target branch của merge request trên máy bạn:

> git fetch <remote\_name> <target\_branch\_name>:<target\_branch\_name>

#### Rebase lại target branch

> git rebase <target\_branch>

Sau khi chạy lệnh rebase, những file bị conflict sẽ hiện thị với từ khóa conflict. Bạn mở các file đấy lên và xem những phần code bị conflict. Những đoạn code này sẽ được chia ra thành 2 phần, phần nửa trên là code hiện tại của target\_branch, nửa dưới là code mà bạn đang sửa. Hãy đọc và giữ lại những phần code phù hợp. xóa bỏ các dấu == phân cách.

#### Thêm lại các file sửa đổi vào commit và kết thúc rebase

> git add .

> git rebase --continute

#### Trường hợp bạn chưa biết resolve những đoạn conflict như nào, có thể huỷ bỏ quá trình rebase bằng cách:\

> git rebase --abort

#### Trước khi tạo một commit, nếu bạn chưa config thông tin định danh của bạn, git sẽ yêu cầu bạn config thông tin định danh của bạn trước khi cho phép commit. Câu lệnh config:

> git config user.name "Tran Tan Loc"

> git config user.email "trantanloc3689@gmail.com"

## Làm việc với branch

Học đi đôi với hành, hãy đút túi những câu lệnh sau để sử dụng branch trong Git nhé. Giả sử bạn đang ở master sau khi tạo mới repository nhé.

#### Tạo một branch mới có tên là feature/a:

# Cách 1, chỉ thực hiện tạo branch:

> git branch feature/a

# Cách 2, tạo mới branch và chuyển luôn sang branch đấy để code:

> git checkout -b feature/a

#### Xem danh sách branch hiện có:

> git branch

Branch hiện tại bạn đang working sẽ có dấu \* ở đầu dòng. Khi có qua nhiều branch trong danh sách, git sẽ hiển thị dạng ScrollView, bạn nhấn nút điều hướng lên/xuống để cuộn lên/xuống. Muốn thoát khỏi chế độ cuộn, gõ:

> :q

Rồi nhấn Enter để thoát khỏi danh sách.

#### Chuyển đổi qua lại giữa các branch Để thực hiện chuyển đổi qua lại giữa các branch, thực hiện command:

> git checkout <branch\_name>

# Ví dụ:

> git checkout feature/a

Khi muốn chuyển sang một branch khác, hãy hết sức lưu ý rằng bạn cần phải tạo thói quen commit lại các thay đổi trước khi chuyển bằng lệnh:

> git commit -m "My message for note"

#### Xóa branch:

# Xóa local branch

> git branch -D branch/name

# Xóa remote branch <xem phần dưới>

> git push remote\_name branch/name --delete

#### Đẩy code lên remote branch

# Cú pháp: git push <remote\_name> <branch\_name>

> git push origin master

# Nếu muốn bắt buộc ghi đè lên remote branch, thêm -f (force)

> git push orgin master -f

#### Merge branch, để merge code từ nhánh B vào nhánh A.

Checkout về nhánh A

> git checkout A

Merge nhánh B vào

> git merge B

Nhập message, lưu lại là xong.

## Remote và Remote branch

### Remote

Để đẩy code lên server repository, chúng ta cần các tham chiếu tới server repository tương ứng. Các tham chiếu này được gọi là Remote. Mỗi remote sẽ có các thông tin:

Tên remote

URL: đường link tới repository (Có hai dạng tương ứng cho 2 giao thức http và ssh).

#### Remote Branch

Như bạn đã biết ở trên, mỗi repository có các branch. Local repository của chúng lại tham chiếu tới nhiều repository khác thông qua remote. Dĩ nhiên các repository này sẽ có các branch bị trùng tên với nhau, chẳng hạn như master. Vậy làm sao phân biệt branch nào của repository nào?

Câu trả lời là Remote Branch. Các remote branch cũng là các branch bình thường, nhưng có prefix được git thêm vào chính là tên của remote.

VD: Chúng ta có:

master

origin/master

upstream/master

### Làm việc với remote

#### Thêm một remote

# Cú pháp: git remote add <remote\_name> <remote\_url>

> git remote add upstream git@github.com/kimnguyen-ict/demo-repository.git

#### Xem danh sách remote đang có:

> git remote -v

#### Thay đổi url của remote

# Cú pháp: git remote set-url <remote\_name> <remote\_url>

> git remote set-url upstream https://github.com/kimnguyen-ict/demo-repository

#### Thay đổi tên của remote

# Cú pháp: git remote rename <old\_name> <new\_name>

> git remote rename upstream demo

#### Xóa một remote

# Cú pháp: git remote remove <remote\_name>

> git remote remove demo

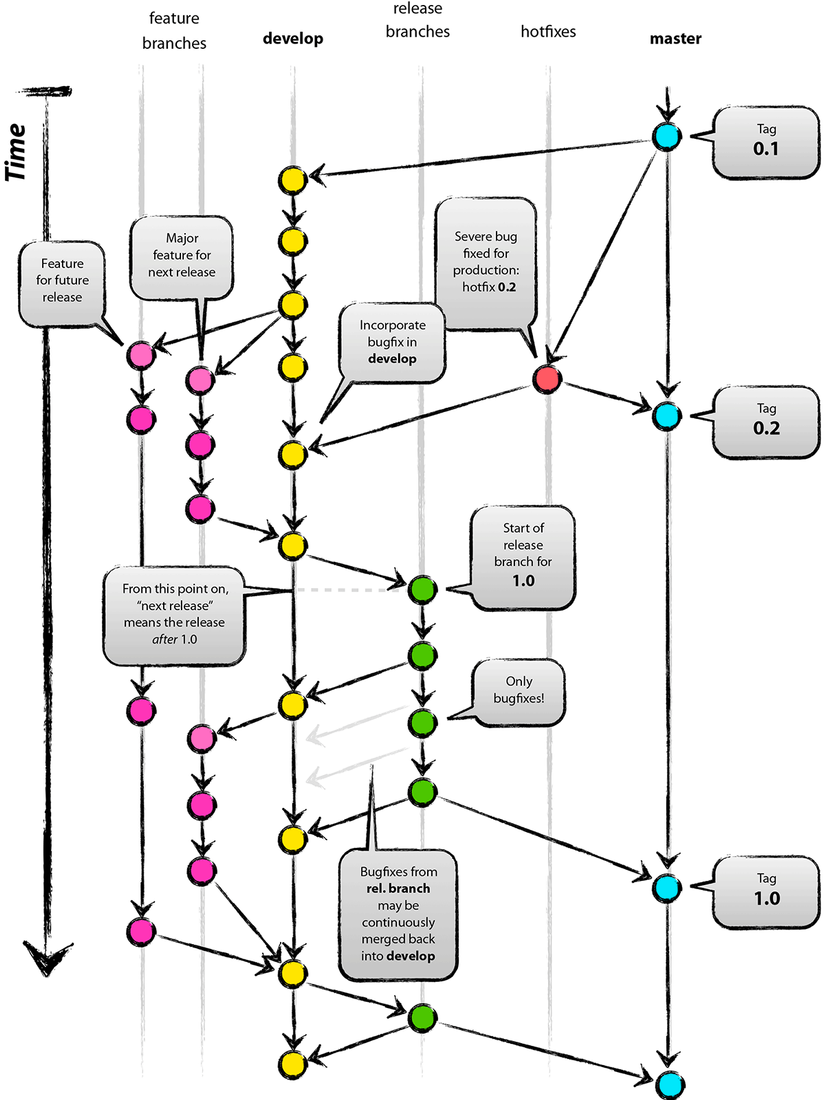
## Git flow

Để làm việc với Git hiệu quả, cả team của dự án cần tuân thủ các quy tắc của git flow.Repository có 2 nhánh chính: master và develop. Master chứa code hoàn chỉnh, sử dụng để deploy lên production. Develop chứ code mới nhất đang được phát triển; sử dụng để deploy lên server testing. Hai nhánh này cần được bảo vệ (protected), chỉ người có quyền mới merge, push commit lên nhánh này.

Mỗi developer (hay contributor) khi phát triển, cần fork Repository gốc về thành một repository của mình. Khi thực hiện sửa đổi một chức năng A, cần checkout từ develop ra một branch mới có tên theo cú pháp: feature/branch-name hoặc fix/feature-a và làm việc trên branch đó. Sau khi hoàn thành, commit lại và tạo merge request (pull request) về repository gốc.

Khi tạo có bug nghiêm trọng cần fix gấp trên production, có thể tạo pull merge request tới thẳng master bỏ qua testing. Tuy nhiên, commit hotfix này cũng phải được merge lại vào nhánh develop để đảm bảo develop có code mới nhất đã được fix lỗi trên đồng thời phục vụ test lại.

Flow chính: Code sửa đổi -> Merge code vào develop -> testing -> merge develop vào master cho production.

Lúc này, quá trình làm việc với git flow có thể được mô tả qua bức ảnh sau:

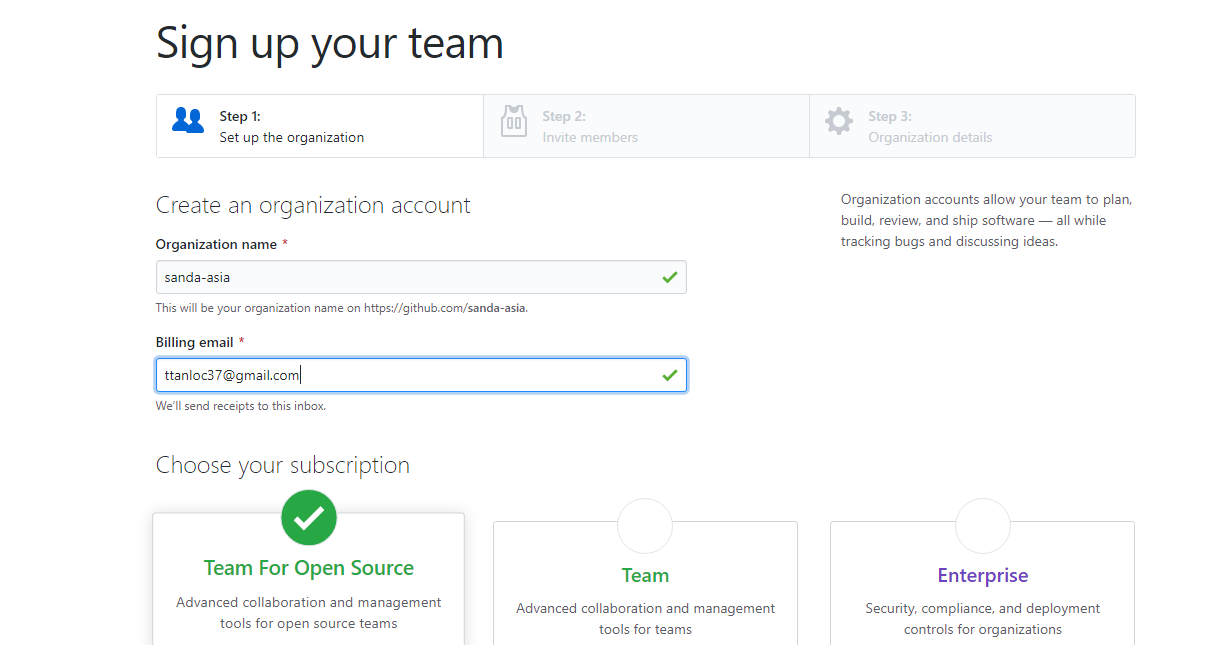
# Github của nhóm cho đồ án cuối kì

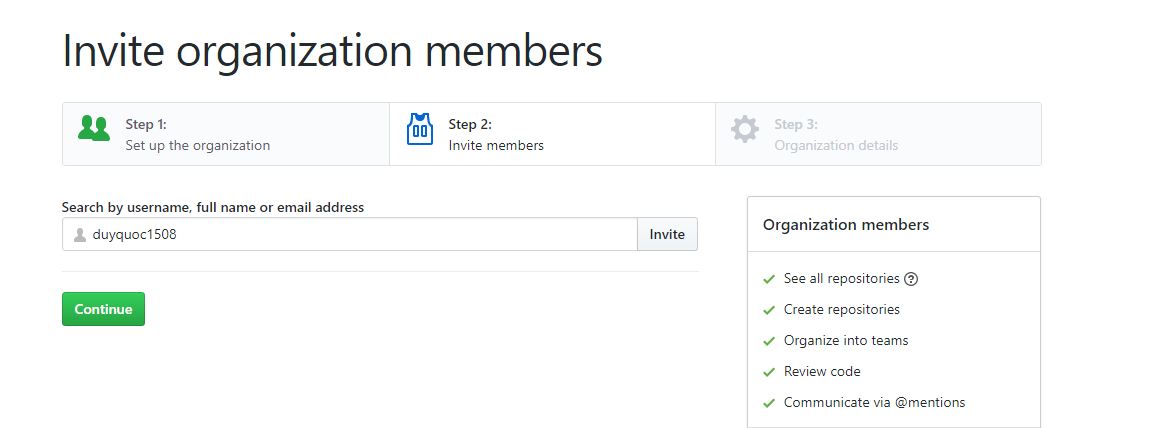
## Github của nhóm:

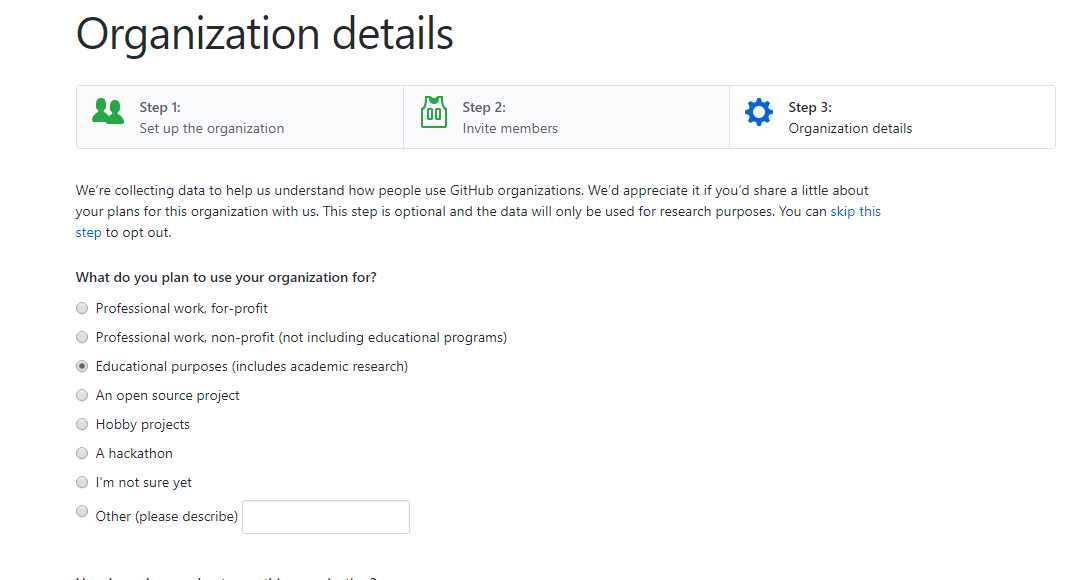
Link github nhóm: <https://github.com/sanda-asia>

Link repository đồ án cuối kì của nhóm: <https://github.com/sanda-asia/tracuunhadat>

## Quá trình khởi tạo github cho tổ chức, nhóm







## Tạo một repository trên nhóm:

