**GIT COMMANDS**

[Learn Git Branching](https://learngitbranching.js.org/?fbclid=IwAR3XC7O3jOmGKzcIl99y8hQfiUVhi4-qNeNjQPa4aNdG7zrGwAZblkZxcv4)

* Git init : create new local respo in order to use git.
* Gitk : to open git UI.
* Git add :
  + Git add [filename] [filename] [filename]
  + Git add . \\ add all changes into staging area
* Git status : check any Modified, Untrack, ....
* Git commit –m “ .... “ : commit to files which are added to local respository
* Git push [origin master]: push/ up all files in local respo to remote respo.
* Git log : show all commit versions.
* Git show [code of commit] : show any change in this commit.
* Git diff : see changes in **modified** files
* Git checkout -- [filename] : discharge changes in working dir (undo that file)
* Git reset : to unstaged changes from staging area
* Git branch : to see the branch we are working in.
* Git checkout –b [branch-name] : create new branch and move to that branch.
* Git merge [branch-name]: merge/combine all changes from branches.
  + Step 1: checkout the branch we want to merge (usually, branch **master**)
  + Step 2: git merge the branches.
* Git branch –D [branch-name]: delete the branch.
* Git reset
  + --soft
  + --hard [code-name]: delete all after that
* .gitignore : add file name in this file to prevent pushing to remote repo. (usually, node\_package,node\_module,...). Note: ignore the files that have never pushed before or you will fix them.
* Git push origin [branch\_name]: to push a branch to remote repo.
* Git pull: use it before doing anything. For update new things
* Git remote –v : to see name of origin.
* Git fetch [branch\_name]: to checkout the branch without seeing. Usually use to test code before merging request.
* How to resolve Conflict:
  + Use rebase
  + Use merge master.

## 5 câu lệnh cơ bản để đưa dự án lên repository

Đầu tiên mình sẽ nêu lên 5 câu lệnh cơ bản nhất cho người mới có thể thao tác với Git được dự án của mình lên trên repository lần lượt là:

* **git init**
* **git add <file>**
* **git commit –m ”message”**
* **git remote add origin <url\_repository>**
* **git push origin master**

 5 câu lệnh ở trên sẽ tương tự với 5 câu lệnh gợi ý của Github cho bạn mỗi khi tạo một repository mới trên đó. Và chúng ta sẽ cùng đi tìm hiểu sâu hơn về ý nghĩa 5 câu lệnh trên.

### Lệnh git init

Câu lệnh **git init** cần phải được sử dụng đầu tiên nếu bạn muốn thao tác với git. Lệnh này sẽ tạo ra một thư mục **.git**ẩn trong thư mục dự án của chúng ta và nếu bạn muốn sử dụng được các lệnh khác của git bắt buộc chúng ta phải có thư mục **.git** trong thư mục dự án đó. Thư mục **.git**này sẽ lưu các thông tin liên quan đến repository trong đó và giúp cho các câu lệnhđược vận hành. Do đó các bạn cần lưu ý gõ lệnh này trước khi sử dụng các lệnh khác nhé ^^.

### Lệnh git add

Lệnh **git add** được sử dụng để xác nhận những thay đổi của một file hay nhiều file và đưa đến vùng Staging Area.

Cấu trúc của lệnh **git add**sẽ là: **git add**<file>

Giả sử mình muốn thêm file readme.MD của thư mục hiện tại thì câu lệnh sẽ là: **git add readme.MD**

Ngoài cách sử dụng như trên, nếu chúng ta muốn add tất cả các file thay đổi của thư mục hiện tại (bao gồm cả các thư mục con bên trong nó) thì chúng ta sẽ sử dụng câu lệnh: **git add .**

Để kiểm tra xem có những file nào đang được add các bạn sử dụng lệnh **git status**nhé.

### Lệnh git commit

Đi cùng với lệnh **git add** sẽ là lệnh **git commit.**Lệnh này được sử dụng để cập nhật các thay đổi lên Local Repository và nó giống như một lát cắt(snapshot) giúp chúng ta có thể lưu giữ lịch sử các phiên bản.

Cấu trúc của câu lệnh này sẽ như sau: **git commit -m”thông điệp mà bạn muốn ghi”**

### Lệnh git remote

Lệnh git remote được sử dụng để kiểm tra xem remote/source bạn có chưa nếu chưa có sẽ add thêm remote.

Cấu trúc câu lệnh của nó sẽ là: **git remote add origin <remote\_url>**

Trong đó remote url sẽ là đường dẫn tới repository của bạn.

Lưu ý:

* bạn hoàn toàn có thể đổi **origin** thành một cái tên khác mà bạn muốn . Ví dụ mình sẽ để là **git remote add quan <remote\_url>**
* Không được để tên remote trùng nhau. Ví dụ mình đã sử dụng câu lệnh **git remote add origin remote\_url1** thì sau đó mình không thể sử dụng lại **origin**cho câu lệnh remote: **git remote add origin remote\_url2** nữa.

### Lệnh git push

Sau khi đã thực hiện xong 4 câu lệnh trên công việc còn lại của chúng ta sẽ là đưa nó lên trên remote. Và để thực hiện công việc này chúng ta sẽ sử dụng câu lệnh **git push origin master.**

**git push -d** <remote\_name>(origin) <branch\_name>

Lưu ý:

* Nếu bạn đã đổi tên **origin** trong câu lệnh **git remote** ví dụ như ở trên mình đổi **origin** thành **quan** lúc này câu lệnh push sẽ là: **git push quan master**
* Nếu khi bạn push lên remote và bị yêu cầu đăng nhập để xác thực tài khoản Git chúng ta sẽ sử dụng 2 câu lệnh sau:
  + **git config – global user.name = “điền username Github của bạn”**
  + **git config – global user.email = “điền email Github của bạn”**

### Một số lưu ý trong 5 câu lệnh trên

Trong 5 câu lệnh trên, câu lệnh **git init**sẽ chỉ cần sử dụng đúng duy nhất 1 lần trong xuyên suốt dự án của bạn. Lệnh**git remote** cũng vậy nếu như bạn không có ý định thêm remote khác vào trong repository của mình. Còn 3 câu lệnh **git add, git commit -m””, git push**sẽ được sử dụng liên tục trong dự án của bạn.

## Một số câu lệnh khác cần phải biết

### Lệnh git pull

Trái ngược với lệnh **git push** mình đã nêu ra ở trên đó là lệnh **git pull**. Nếu như để đưa dự án của mình từ local lên trên remote chúng ta sẽ sử dụng lệnh **git push**và để đưa cập nhật của dự án từ trên remote về máy chúng ta sẽ cần sử dụng tới câu lệnh**git pull**

Cấu trúc của câu lệnh này tương tự như **git push**và chúng ta chỉ cần thay chữ push thành pull là được: **git pull origin master**

### Lệnh git clone

Câu lệnh git clone được sử dụng để sao chép một dự án có sẵn trên remote về máy mình.

Câu lệnh sẽ được sử dụng như sau:**git clone <remote\_url>**

Lưu ý: Khi bạn sử dụng câu lệnh **git clone**thì chúng ta sẽ không cần phải sử dụng câu lệnh **git init** để tạo thư mục **.git** nữa do thư mục chúng ta clone về đã có sẵn thư mục **.git**trong đó rồi.

### Lệnh git branch

Ở trên phần câu lệnh **git push**mình có đề cập tới branch. Để tạo một branch chúng ta sẽ sử dụng câu lệnh sau: **git branch <tên nhánh>** ví dụ mình muốn tạo một nhánh dev thì câu lệnh sẽ là **git branch dev**

### Lệnh git checkout

Lệnh **git checkout**được sử dụng để chuyển từ branch này sang branch khác.

Cấu trúc câu lệnh sẽ là: **git checkout <tên nhánh>**

Ví dụ bạn đang ở nhánh master và muốn chuyển sang nhánh dev thì câu lệnh sẽ là: **git checkout dev**

Ngoài ra bạn cũng có thể sử dụng câu lệnh **git checkout -b <tên nhánh>** để vừa chuyển nhánh và vừa tạo nhánh mới.

### Lệnh git merge

Lệnh **git merge** được sử dụng để merge (trộn) 2 branch lại với nhau.

Cấu trúc: **git merge <tên nhánh>**

Ví dụ bạn đang đứng ở nhánh master và muốn merge những thay đổi của nhánh dev sau khi commit vào đây chúng ta sẽ sử dụng câu lệnh **git merge dev.** Sau khi merge xong nhánh master của chúng ta sẽ có thêm cả phần những thay đổi của nhánh dev.Hay giả sử nhánh master mình có một file là Main.java, nhánh dev có một file là User.java thì sau khi sử dụng câu lệnh **git merge dev** thì nhánh master của chúng ta lúc này sẽ có 2 file là Main.java và User.java.

### Lệnh git reset

Khi đã thực hiện commit, commit đó chưa public (chưa đẩy lên Remote Repo bằng lệnh git push) thì bạn có thể hủy (undo) commit đó với hai trường hợp bằng lệnh **git reset** như sau:

#### git reset với tham số –soft

Trường hợp này sẽ hủy commit cuối, con trỏ HEAD sẽ chuyển về commit cha. Đồng thời những thay đổi của commit cuối được chuyển vào vùng staging nhằm để có cơ hội commit lại hoặc sửa đổi, cú pháp lệnh như sau: **git reset –soft HEAD~1**

### Lệnh git fetch

**git fetch** --all --prune

để cập nhật lại các branches origin

Chúng ta có thể sửa đổi số sau dấu ~ tùy thuộc vào việc bạn muốn reset về commit thứ mấy sau nó.

#### git reset với tham số –hard

Khi dùng tham số –hard thì kết quả giống với dùng tham số –soft, chỉ có một khác biết là nội dung thay đổi của commit cuối không đưa đưa vào staging mà bị hủy luôn. Trường hợp này dùng khi bạn quyết định hủy hoàn toàn commit cuối: **git reset --hard HEAD~1(hoặc commitId)**

#### Hủy git add

Nếu bạn đã dùng lệnh git add để cập nhật thay đổi vào vùng staging, bạn có thể hủy thao tác này bằng cách thực hiện lệnh: **git reset**

#### Hủy đưa một file vào staging

Nếu muốn hủy một file nào đó trong vùng staging chứ không phải toàn bộ thì dùng lệnh: **git reset — filename**

**hủy các file ở working repository(files chưa add vào staging)**

git checkout . : sẽ hủy hết toàn bộ.

git checkout <pathnamefile>: hủy file theo đường dẫn (Relative Path)

**lưu những gi làm working repo**

git stash

revert 1 file về lại commit cũ

git checkout <commitId> <pathFile>

revert merge: git merge --abort. trở về trạng thái trước khi merge

#### git revert

A -> B -> C -> D -> E

Let's say C is the target commit. Save the hash of C. Lets say 1df455v631fca.

git revert 1df455v631fca -m "message..."

This removes the changes done by C from the current branch.

A -> B -> C -> D -> E -> -C