**GIT COMMANDS**

Table of Contents

[**1.** **Các lệnh git cơ bản** 2](#_Toc85962905)

[**2.** **Git push** 2](#_Toc85962906)

[**3.** **git checkout** 2](#_Toc85962907)

[**4.** **git switch** 2](#_Toc85962908)

[**5.** **git restore** 3](#_Toc85962909)

[**6.** **git remote** 3](#_Toc85962910)

[**7.** **git clone** 3](#_Toc85962911)

[**8.** **git branch** 3](#_Toc85962912)

[**9.** **git merge** 3](#_Toc85962913)

[**10.** **git reset** 3](#_Toc85962914)

[**11.** **git fetch** 4](#_Toc85962915)

[**12.** **git stash** 4](#_Toc85962916)

[**13.** **git revert** 5](#_Toc85962917)

[**14.** **git rebase** 5](#_Toc85962918)

[**15.** **git commit** 6](#_Toc85962919)

1. **Các lệnh git cơ bản**

* git clone [ssh/https]
* gitk
* git add [file/name/url]
* git commit -m ‘’
* git commit --am ‘’
* git log
* git diff
* Git show [code of commit] : show any change in this commit.
* .gitignore: là 1 file dùng để check những file ko dc add vào git

1. **Git push**

* **Git push origin [branch\_name]**: push lên nhánh ở origin/ remote
* Khi tạo nhánh mới (nhánh chưa có origin) thì **git push --set-upstream/-u origin**
* Dùng alias: **git push -u origin HEAD** để push lên nhánh origin với cái tên nhánh hiện tại

Về sau thì chỉ cần **git push,git pull** là ok

\* lợi ích của **upstream**: pull, push ko cần thêm tên nhánh. fetch về commit mới nhất.

* 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”**

1. **git checkout**

* **git checkout <tên nhánh>:** chuyển sang nhánh khác
* **git checkout -b <tên nhánh>:** vừa tạo nhánh mới và vừa chuyển nhánh
* **git checkout . :** Xóa hết các unstage file đang làm.(những file chưa commit)
* **git** **checkout <pathnamefile>:** hủy file theo đường dẫn (Relative Path)
* **git checkout [hashId]**: chuyển sang nhánh có commit và trỏ đến commit đó

(note: git checkout có rất nhiều công dụng - swiss army knife: tạo nhánh, chuyển nhánh, lùi version,... nhưng giờ thì nên tách ra từng lệnh riêng)

Vd: nếu có tên file và tên branch giống nhau thì khi sử dụng git checkout sẽ gây confuse

1. **git switch**

* **git switch <tên nhánh>:** chuyển sang nhánh khác
* **git switch -c < tên nhánh >:** vừa tạo nhánh mới và vừa chuyển nhánh
* **git switch -d [hashId]:** chuyển sang nhánh có commit và trỏ đến commit đó

1. **git restore**

* **git restore** : tương đương git checkout --<path\_to\_file>

1. **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

1. **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.

1. **git branch**

* **git branch <tên nhánh>:** tạo nhánh mới git branch staging (hoặc đang ở nhánh khác và muốn checkout wa nhánh mới sau khi tạo xong: **git switch -c <tên nhánh>**)
* **git branch -D <tên nhánh>:** để xóa nhánh ở local
* **git push -d origin <tên nhánh>:**  xóa nhánh ở remote

1. **git merge**

* Lệnh **git merge** được sử dụng để merge (trộn) 2 branch lại với nhau.
* Cấu trúc: **git merge <tên nhánh>**
* Note: lệnh này nghĩa là **merge nhánh khác** vào **nhánh đang đứng**
* Ví dụ bạn đang đứng ở nhánh **master** và muốn merge những thay đổi của nhánh **dev**: chúng ta sẽ sử dụng câu lệnh **git merge dev.**

**Note: có thể xảy ra conflict**

* Cú pháp gọn khi merge:

**git checkout feature  
git merge main**

shortcut: **git merge feature main** (có thể hiểu branch1 là branch cần giữ gốc, branch2 là branch cần merge vào branch1)

1. **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 (lùi về 1 commit)**

* **git reset với tham số --hard**

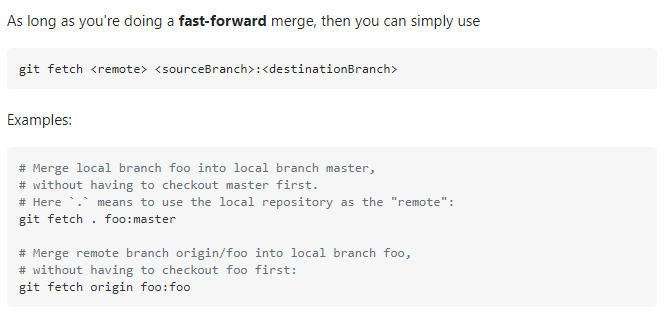
Khi dùng tham số --hard thì kết quả giống với dùng tham số –soft, chỉ có một khác biết là nội dung thay đổi của commit cuối không đưa đưa vào staging mà bị hủy luôn. Trường hợp này dùng khi bạn quyết định hủy hoàn toàn commit cuối: **git reset --hard HEAD~1(hoặc commitId)**

Để Hủy **git add** bạn có thể 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**

1. **git fetch**

* **git fetch** --all –prune
* **git fetch**: để fetch/get các commit mới nhất ở **origin** nhưng **ko merge** vào nhánh hiện tại
* **git fetch** . develop:master : để merge nhánh develop ở local vào master ko cần checkout
* **git fetch origin** develop:master: để merge nhánh develop vào master ko cần checkout
* **Note:** chỉ có thể làm cách này với fast-forward. Non-fast-forward merge



1. **git stash**

* **git stash**: lưu những j đang làm(unstaged) vào nháp
* **git stash pop**: lấy cái cuối cùng của stash ra và bỏ khỏi stash (note: khi git pull xong git stash pop sẽ có case conflict)
  + nếu conflict thì **git reset HEAD –hard** để xóa cái đang conflict và giữ stash
* **git stash clear**: xóa hết stash

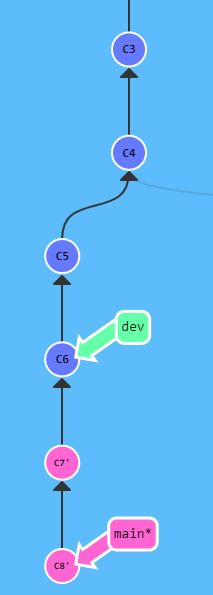
1. **git revert**

git revert [hashId] -m "message..." : revert lại commit cũ và tạo 1 commit mới ở nhánh hiện tại

1. **git rebase**

**git rebase <branch\_name>** : sẽ dùng **<branch\_name>** làm gốc cho nhánh đang đứng. Và commitID nhánh hiện tại sẽ bị thay đổi

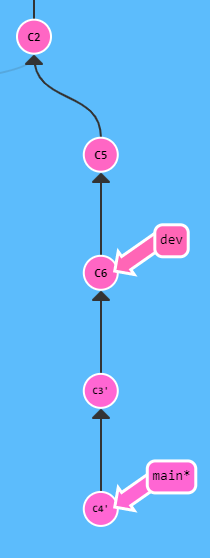
(re: làm lại, base: gốc)



* trường hợp commit C5, C6 nhánh dev **có trước** nhánh main C7, C8

Hình bên đang ở nhánh main\*

* **git rebase dev** : là dùng nhánh dev làm gốc, tất cả các commitID của nhánh main bị thay đổi hết



* trường hợp commit C5, C6 nhánh dev **có sau** nhánh main C3, C4

Hình bên đang ở **nhánh main\***

* **git rebase dev** : là dùng nhánh dev làm gốc, tất cả các commitID của nhánh main bị thay đổi hết
* ko khác j so với việc commit dev có trước nhưng key là dùng nhánh dev làm base và tất cả commit của main sẽ nằm lên đầu

1. **git commit**

**git commit –m ‘message’**

**git commit --amend -m 'message'** : dùng để gộp chung với commit phía dưới (nếu commit dưới chưa push)

**git add .** > **git commit --amend**