

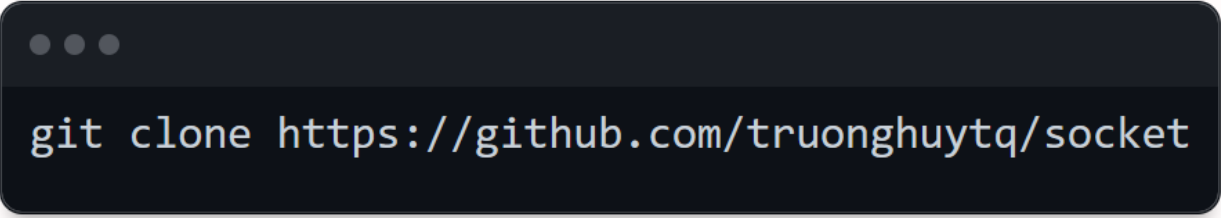
Bước 1: BẮT ĐẦU

1.1 Phân chia công việc

1.1.1 Clone code về

Sau khi nhận được link github từ trưởng nhóm. Các thành viên thực hiện lấy code về bằng cách sử dụng clone git.

- ❖ Khi bạn nhận được link github từ trưởng nhóm. Thực hiện câu lệnh clone theo link được nhận, **chú ý** : link dùng để clone không có hậu tố **invitations** trong URL

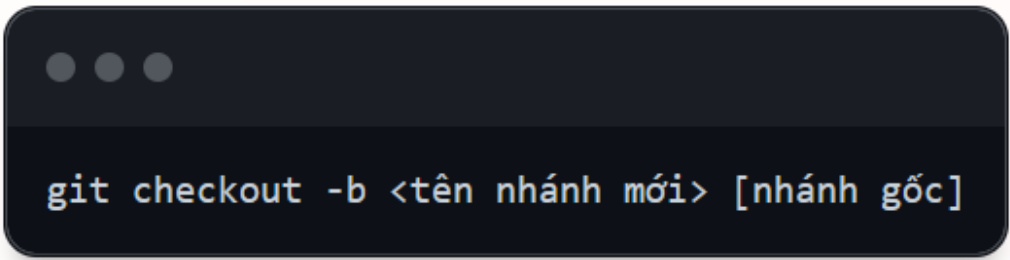


```
git clone https://github.com/truonghuytq/socket
```

Hình 1: Clone với link "https://github.com/truonghuytq/socket"

1.1.2 Tạo nhánh mới

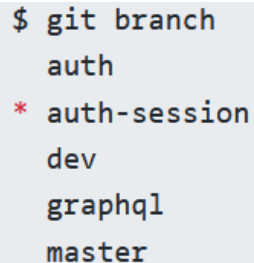
- ❖ **Nhắc bài chút xíu** để tạo nhánh mới trong Git bạn dùng lệnh :



```
git checkout -b <tên nhánh mới> [nhánh gốc]
```

Hình 2: Cú pháp tạo nhánh mới

- ❖ Để xem nhánh hiện tại là nhánh nào, bạn có thể dùng lệnh **git branch**.

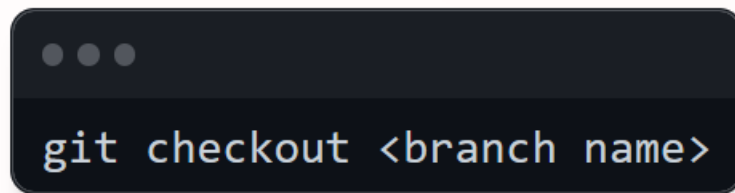


```
$ git branch
auth
* auth-session
dev
graphql
master
```

Hình 3: Xem nhánh hiện tại với Git branch

Trong ví dụ trên thì nhánh hiện tại chính là : `auth-session` .

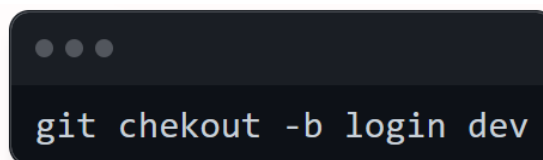
❖ Để chuyển nhánh bạn dùng lệnh như **Hình 4**



```
git checkout <branch name>
```

Hình 4: Cú pháp chuyển nhánh

❖ **Mỗi khi phát triển tính năng mới**, bạn sẽ tạo một nhánh mới từ `dev`



```
git checkout -b login dev
```

Hình 5: Tạo nhánh "login" từ nhánh "dev"

Như ví dụ trên dùng để tạo nhánh `login` từ `dev`.

Sau khi tạo nhánh xong , bạn sẽ thực hiện `code` trên nhánh của mình vừa tạo.

Lưu ý:

- Tên nhánh là tên chức năng mà thành viên đảm nhận.
- Thành viên không được Push code của nhánh chức năng lên Github.
- ➔ Thành viên cần nắm rõ lý thuyết của những câu lệnh: Commit, Push, Fetch, Reset, Rebase, Merge, các câu lệnh Git cơ bản,... trước khi thực thi tiếp quy trình.

Bước 2: COMMIT SAU KHI CODE

2.1 Commit khi có file mới thay đổi.

- ❖ Bạn thực hiện lần lượt các câu lệnh sau :
 - Câu lệnh **add** để thêm các thay đổi mới vào vùng **staging** chuẩn bị cho **commit** tiếp theo.

A terminal window with a dark background and three dots in the title bar. The command `git add .` is entered in a light-colored monospace font.

Hình 6: Câu lệnh add

- Sau khi thực hiện câu lệnh **add** có thể hủy các file trong vùng staging trước khi thực hiện câu lệnh **commit**.

A terminal window with a dark background and three dots in the title bar. The command `git reset` is entered in a light-colored monospace font.

Hình 7: Rollback câu lệnh add

- Câu lệnh **commit** với các tham số:
 - **-m** : để thêm message commit
 - **-s** : để thêm chữ ký khi commit

A terminal window with a dark background and three dots in the title bar. The command `git commit -m "message commit" -s` is entered in a light-colored monospace font.

Hình 8: Câu lệnh commit

2.2 Commit khi không có file mới thay đổi

- ❖ Bạn có thể sử dụng cùng một lúc 2 câu lệnh **add** và **commit**

A terminal window with a dark background and three dots in the title bar. The command `git commit -a -m "new message commit" -s` is entered in a light-colored monospace font.

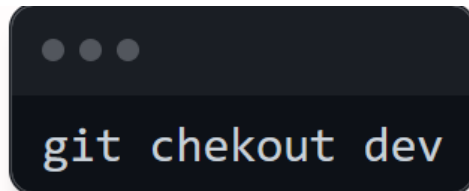
Hình 9: Kết hợp câu lệnh add và commit

Bước 3: CHUẨN BỊ MERGE VÀO DEV

3.1 Tiến hành lấy code mới nhất của nhánh “dev” trên Github

3.1.1 Tiến hành “Git Fetch”

- ❖ Đảm bảo chắc chắn bạn đang ở nhánh `dev`, nếu chưa hãy chuyển sang nhánh `dev`. Có thể bỏ qua bước này nếu đang ở nhánh `dev`.



```
git checkout dev
```

Hình 10: Chuyển sang nhánh "dev"

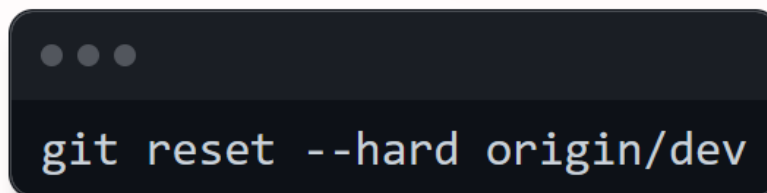
- ❖ Sử dụng lệnh `git fetch` để tải xuống code mới nhất từ Github.



```
git fetch --all
```

Hình 11: Câu lệnh Git Fetch

- ❖ Sử dụng lệnh `git reset` để ghi đè code của nhánh `dev` local với code của nhánh `dev` trên Github (thay thế code của `dev` local bằng code của `dev` trên Github).



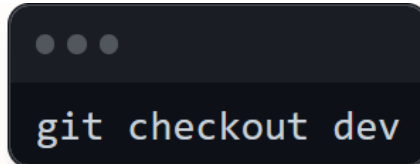
```
git reset --hard origin/dev
```

Hình 12: Câu lệnh Git Reset

Bước 4: MERGE VÀO DEV

4.1 Merge vào nhánh “dev” khi không có CONFLICT

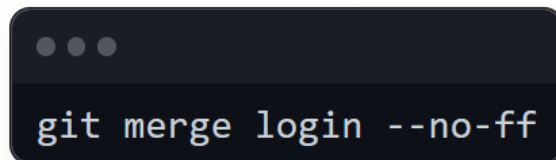
- ❖ Chắc chắn rằng bạn đang ở nhánh `dev`, nếu chưa bạn chuyển qua nhánh `dev`



```
git checkout dev
```

Hình 13: Chuyển qua nhánh "dev"


- ❖ Thực hiện lệnh `merge non-fast-forward` (Bắt buộc có tham số `--no-ff`)



```
git merge login --no-ff
```

Hình 14: Merge non-fast-forward nhánh "login" vào "dev"

- ❖ Thông báo khi đã merge thành công



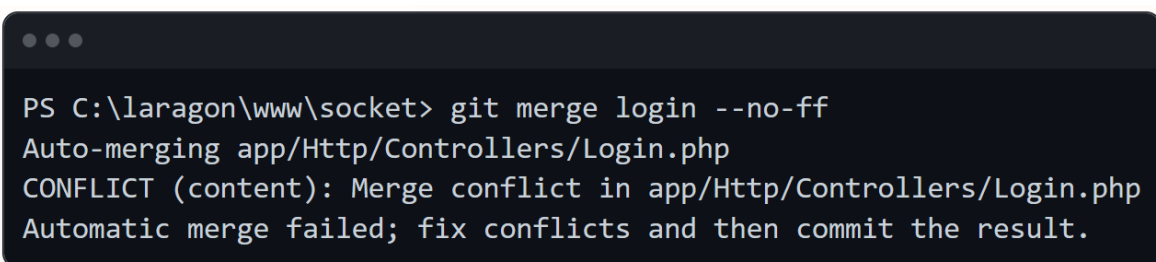
```
PS C:\laragon\www\socket> git merge login --no-ff
Already up to date.
```

Hình 15: Thông báo có thể có khi merge "login" vào "dev" không bị CONFLICT

4.2 Merge vào nhánh “dev” khi có CONFLICT

Chúng ta thực hiện các câu lệnh lần lượt như mục 4.1. Xung đột code sẽ xảy ra khi thực hiện câu lệnh `merge non-fast-forward` ở Hình 14.

- ❖ VD lỗi khi xung đột code xảy ra với lệnh `merge non-fast-forward` trong Hình 16

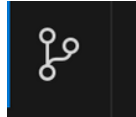


```
PS C:\laragon\www\socket> git merge login --no-ff
Auto-merging app/Http/Controllers/Login.php
CONFLICT (content): Merge conflict in app/Http/Controllers/Login.php
Automatic merge failed; fix conflicts and then commit the result.
```

Hình 16: Thông báo khi merge "login" vào "dev" bị CONFLICT

4.2.1 Xử lý xung đột code khi tiến hành “Git Merge” có CONFLICT

- ❖ Click vào biểu tượng source control :



Hình 17: Biểu tượng source control trong Visual Studio Code

- ❖ Mở file bị xung đột ở phần **Merge Changes** và tìm đến code bị xung đột, ở đây có các lựa chọn để chúng ta giải quyết xung đột code :
 - **Accept Current Change**: áp dụng code hiện có của nhánh **dev**
 - **Accept Incoming Change**: áp dụng code mới của nhánh **login**
 - **Accept Both Changes**: áp dụng cả hai code của nhánh **dev** và nhánh **login**
 - **Compare Changes**: So sánh 2 thay đổi giữa 2 nhánh **dev** và nhánh **login**

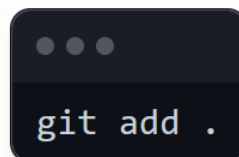


Hình 18: Màn hình xung đột code trong file

Lưu ý : Chỉ sửa các file bị xung đột trong **Merge Changes**. Khi xử lý xung đột code không tự ý sửa code của các thành viên khác.

4.2.2 Sau khi xử lý xong xung đột code

- ❖ Chắc chắn rằng bạn đang ở nhánh **dev**. Tiến hành lưu file lại và tiếp tục thực hiện các câu lệnh Git add.



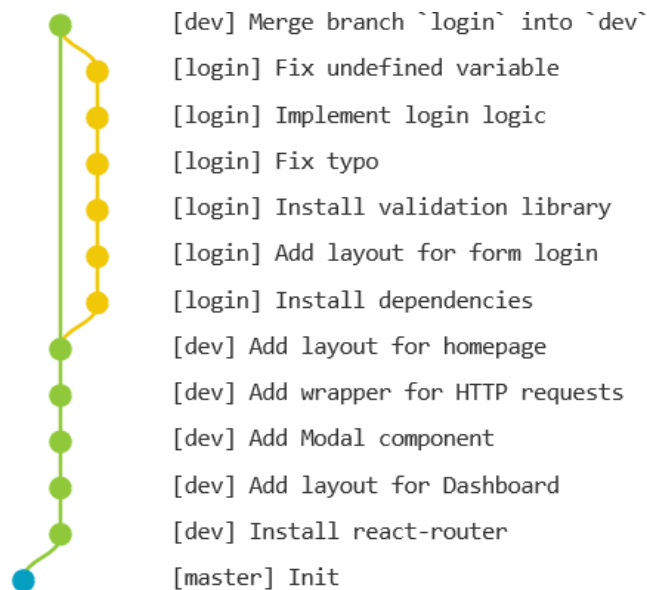
Hình 19: Câu lệnh Git add trên nhánh “dev”

- ❖ Tiến hành commit cho nhánh **dev** với message commit (Chú thích)
VD : message commit là “new message commit”



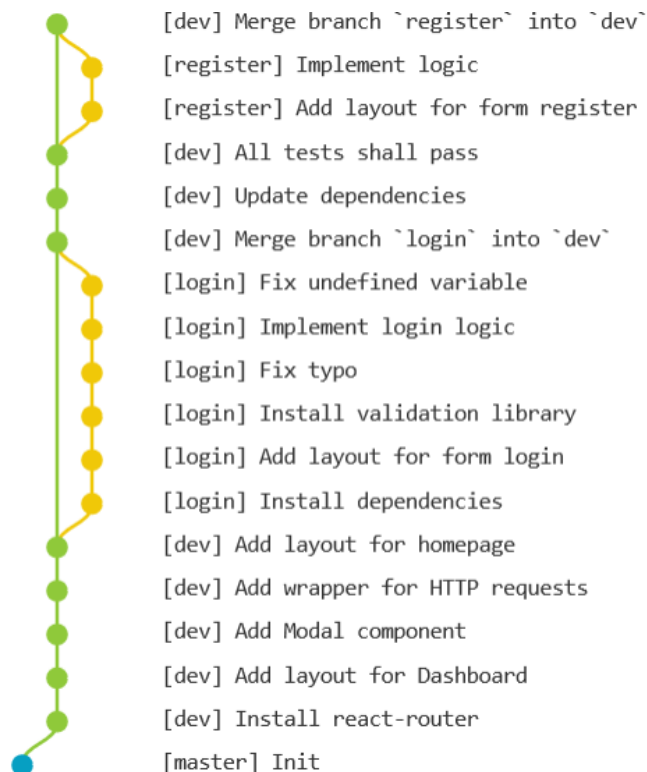
Hình 20: Câu lệnh commit trên nhánh “dev”

❖ Kết quả khi sử dụng **merge non-fast-forward**



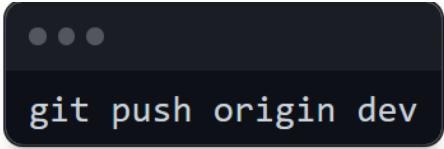
Hình 21: History sau khi merge non-fast-forward "login" vào "dev"

- ❖ Như bạn thấy trong **Hình 21**, một commit mới được tạo ra, giúp bạn dễ dàng nhận biết thời điểm nhánh login được merge vào. Khi dự án phát triển dần theo thời gian, history của dev sẽ như **Hình 22**.



Hình 22: History sau khi thực thi đồng bộ nhiều merge non-fast-forward

- ❖ Hiện tại thì chúng ta đã cập nhật code được vào nhánh **dev**. Nhưng code vẫn chỉ đang ở trên local, để đưa code lên Github bạn thực hiện **push** code lên Github

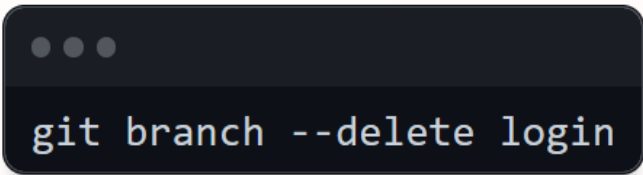


```
git push origin dev
```

Hình 23: Cập nhật code local lên Github từ nhánh “dev”

4.3 Xóa nhánh cũ vừa merge thành công vào nhánh “dev”

Chú ý : Sau khi đã thực hiện xong merge vào **dev**, xóa nhánh **login** cũ từ nhánh **dev**



```
git branch --delete login
```

Hình 24: Câu lệnh Xóa nhánh "login" từ "dev"

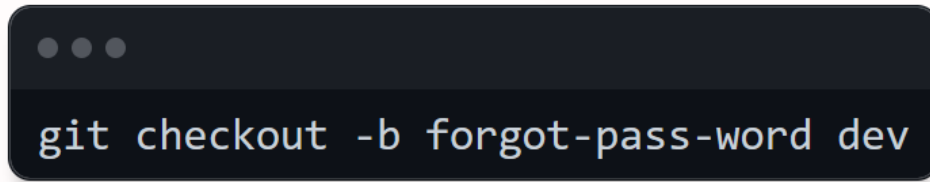
Bước 5: KHI VIẾT CHỨC NĂNG MỚI

5.1 Chắc chắn code nhánh “dev” là code mới nhất

- ❖ Tiến hành **Git Fetch** và **Git Reset** như mục 3.1 khi ở nhánh **dev**

5.2 Tạo nhánh mới từ nhánh “dev” khi viết chức năng mới hoặc fix code

- ❖ Tiến hành tạo nhánh mới theo chức năng thành viên đảm nhận ví dụ tạo nhánh mới **forgot-pass-word** từ nhánh **dev** và thực thi quy trình lại từ đầu.



Hình 25: Câu lệnh tạo nhánh mới "forgot-pass-word" từ nhánh "dev"

➔ Sau khi tạo nhánh con mới từ nhánh **dev** . Bạn sẽ tiếp tục code trên nhánh mình vừa tạo và thực thi lại quy trình từ đầu.