

Contents

Bài 03. Các câu lệnh cơ bản để quản lý file và thư mục.....	2
Bài 04. Cách tạo Repository mới trong Git.....	2
Bài 05. Cấu hình thông tin cho Repository	3
Bài 06. Thực hành GIT add GIT commit GIT status GIT diff GIT log.....	4
Bài 07. Cấu hình GITIGNORE để bỏ qua các file không cần giám sát.....	4
Bài 08. Cách tương tác với Remote Repository.....	5
Bài 10. Câu lệnh GIT CHECKOUT chuyển đổi giữa các commit trong Git	5
Bài 11. Branches - cách làm việc với nhiều nhánh trong Git	6
Bài 12. Git Merge - Kết hợp nội dung từ các nhánh.....	6
Bài 13. Git Rebase - tái cơ sở cho một nhánh trong Git.....	7
Bài 14. Cách xóa nhánh trong GIT	8
Bài 15. Git Reset - Hủy bỏ commit.....	9
Bài 16. Git Revert - Quay lại các commit trước đây.....	10
Làm việc với Github	11
Bài 19. Cách clone dự án từ GitHub về máy	11
Bài 20. Đẩy dự án từ máy cá nhân lên GitHub	12
Bài 21. Fork và cập nhật dựa trên Repo của người khác trên GitHub	13
Bài 22. Cách tạo Pull Request trong Github	13
Bài 23. Cài đặt và sử dụng GitHub Desktop	13
Bài 24. Sử dụng Git và GitHub trong Visual Studio Code	13

Những dòng được highlight là câu lệnh

Bài 03. Các câu lệnh cơ bản để quản lý file và thư mục

- 1) `cd ..` : trở về thư mục trước
- 2) `cd` tên thư mục : di chuyển đến thư mục mong muốn
- 3) `clear` : xóa toàn bộ nội dung trên gitbash
- 4) `ls` hoặc `dir` : hiển thị các file, folder bên trong thư mục hiện tại
- 5) `mkdir` "tên thư mục" : tạo 1 thư mục mới
- 6) `touch` "tên File" : tạo 1 file mới (ex: touch "vidu1.txt")
- 7) `echo` "text" > "File" : ghi đè text vào toàn bộ nội dung của file
- 8) `echo` "text" >> "File" : ghi thêm "text" vào nội dung của file
- 9) `echo` > "tên file" : câu lệnh này cũng có thể tạo file mới
- 10) `cat` "File" : in nội dung trong file ra git bash
- 11) `diff` "file thứ nhất" "file thứ 2" : hiển thị sự khác nhau giữa nội dung của file thứ nhất và nội dung của file thứ 2
- 12) `rm` "File" : xóa 1 file
- 13) `rm -d` "folder" : xóa 1 folder (chỉ xóa được thư mục rỗng)
- 14) `rm -r` "folder" : xóa 1 folder (có chứa dữ liệu, cẩn thận khi dùng câu lệnh này tránh mất hết dữ liệu)

Bài 04. Cách tạo Repository mới trong Git

I. Từ khóa

- repository (repo): kho lưu trữ
- commit : mỗi đơn vị làm việc
- branch : nhánh
- main hoặc master : tên của repo chính (main repo)
- merge hoặc rebase : kết hợp 2 nhánh
- develop : tên của nhánh, lập trình viên

II. Câu lệnh

- `git --help` : trợ giúp, hướng dẫn
- `git --version` : hiển thị thông tin phiên bản Git
- `git status` : hiển thị trạng thái kho lưu trữ
- `git log` (`Git log --oneline`): hiển thị lịch sử các commit

- `git init [repo name]` : tạo ra 1 kho lưu trữ trống
- `git clone [repo name] [clone name]` : để tạo 1 bản sao được liên kết với repo
- `git config -l` : xem cấu hình hiện tại
- `git config -l [--scope] [option name] [value]` :
 - `--scope` : `--system` => tất cả người dùng
 - `--global` => liên quan đến repo(s)
 - `--local` => liên quan đến 1 repo

Bài 05. Cấu hình thông tin cho Repository

1. Tạo repository mới : `git init EX2` / `cd EX2`
2. Xem cấu hình : `git config -l`
3. Xem cấu hình global : `git config -l --global`
4. Xem cấu hình local : `git config -l --local`
5. Cấu hình (global) 'user.name'


```
git config --global user.name "Le Nhật Tung"
```
6. Cấu hình (global) 'user.email'


```
git config --global user.email "lnt@gmail.com"
```
- 7-8. Lặp lại bước 5 và 6, nhưng cấu hình local

Bài 06. Thực hành GIT add | GIT commit | GIT status | GIT diff | GIT log

- `git add [file name (s)]` : thêm tệp vào Index
- `git add .` : thêm tệp (tất cả)
- `git commit -m "Nội dung"` : tạo commit → repo
- `git status` : sự khác biệt giữa 3 cây
- `git diff` : so sánh với commit trước cùng
- `git log` : để lịch sử

Bài 07. Cấu hình GITIGNORE để bỏ qua các file không cần giám sát

- `DS_store`
- `Thumbs.db`
- `log`
- `ThưMuc`

bỏ qua file log
`*.log`

bỏ qua thư mục
`node1/`

Tên file
`read.txt`

Bài 08. Cách tương tác với Remote Repository

Câu lệnh:

- `git init --bare`: Tạo một central repo
- `git clone [repo_name] [clone_name]`:
Sao chép và liên kết repo_name
- `git fetch`: lấy các thông tin về commit mới từ central
- `git pull`: lấy dữ liệu từ central về local repo
- `git push`: đẩy các commit từ local về central.

Bài 10. Câu lệnh GIT CHECKOUT chuyển đổi giữa các commit trong Git

Cú pháp:

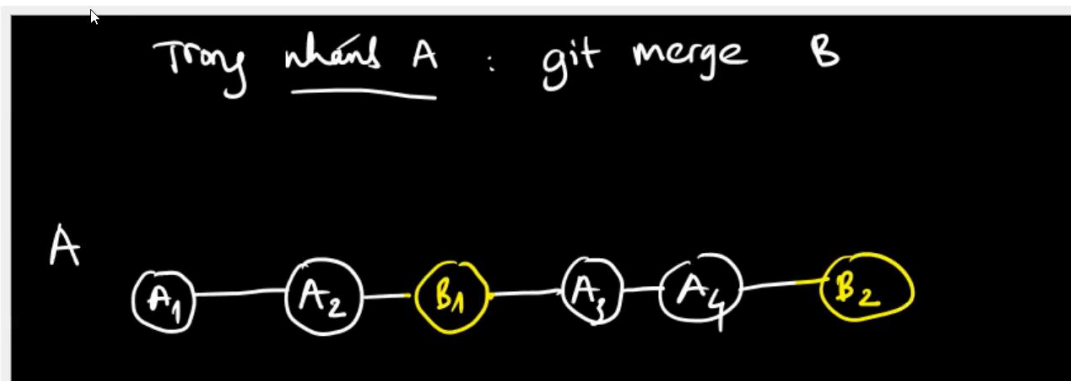
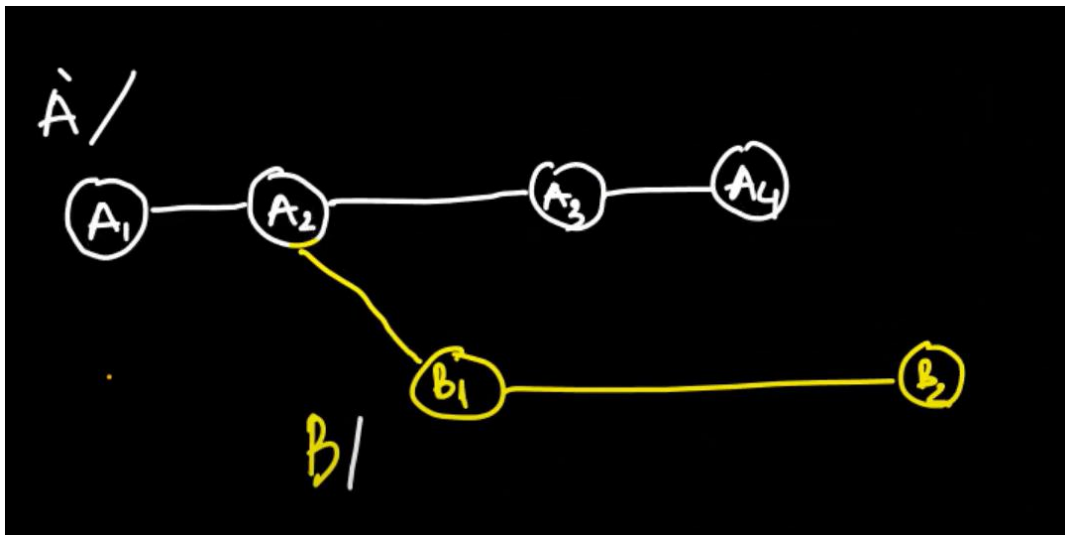
`git checkout` <commit_hash>

Dùng để quay lại bất kỳ thời điểm nào mà mình commit, nên dùng thêm câu lệnh `git log` để xem lại lịch sử commit và thấy được commit_hash.

Bài 11. Branches - cách làm việc với nhiều nhánh trong Git

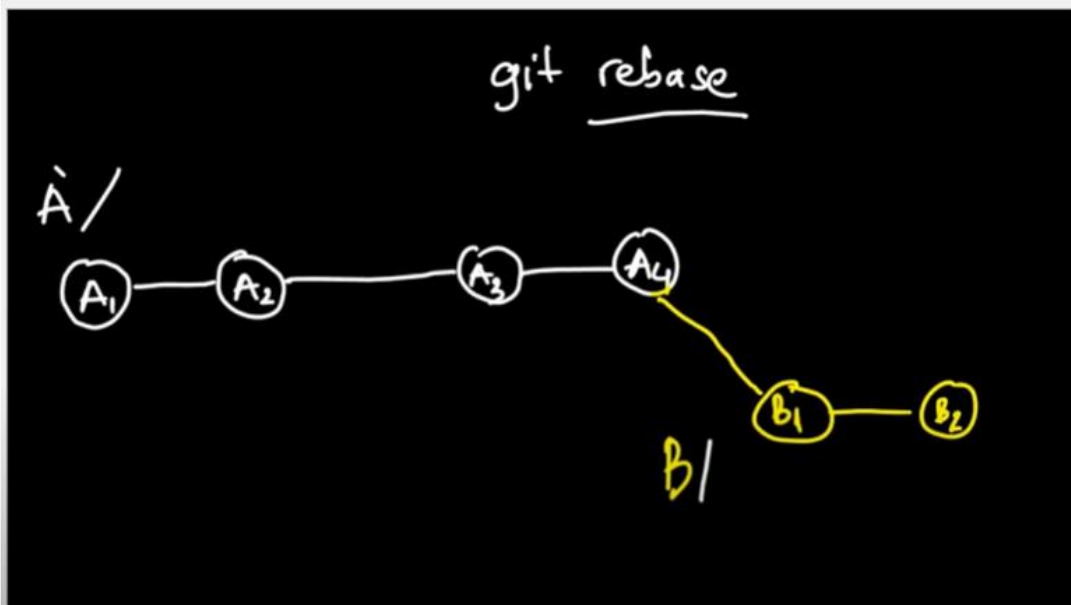
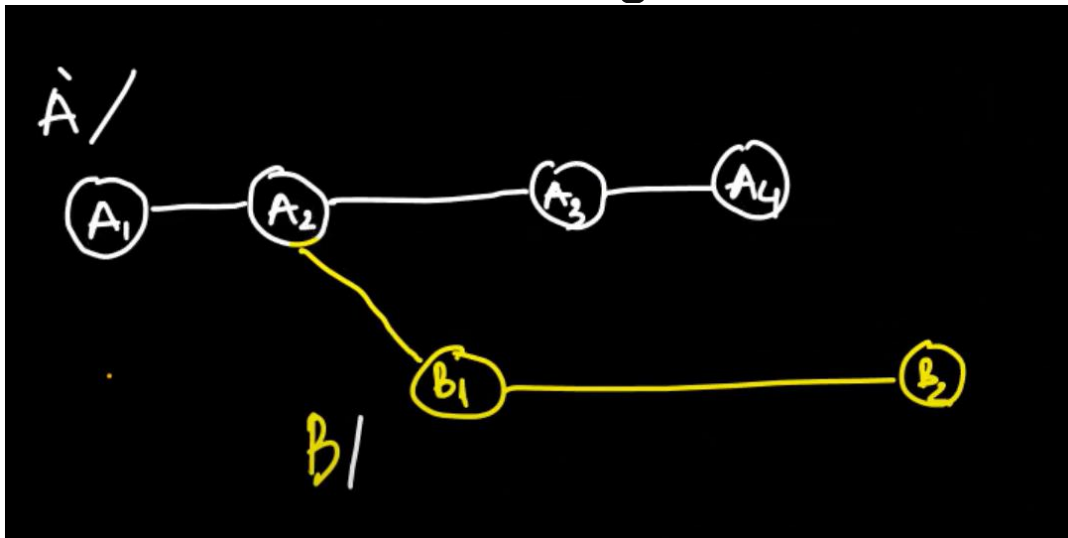
- `git branch <branch-name>` : tạo
- `git checkout <branch-name>` : chuyển
- `git branch -l` : xem nhánh hiện tại

Bài 12. Git Merge - Kết hợp nội dung từ các nhánh



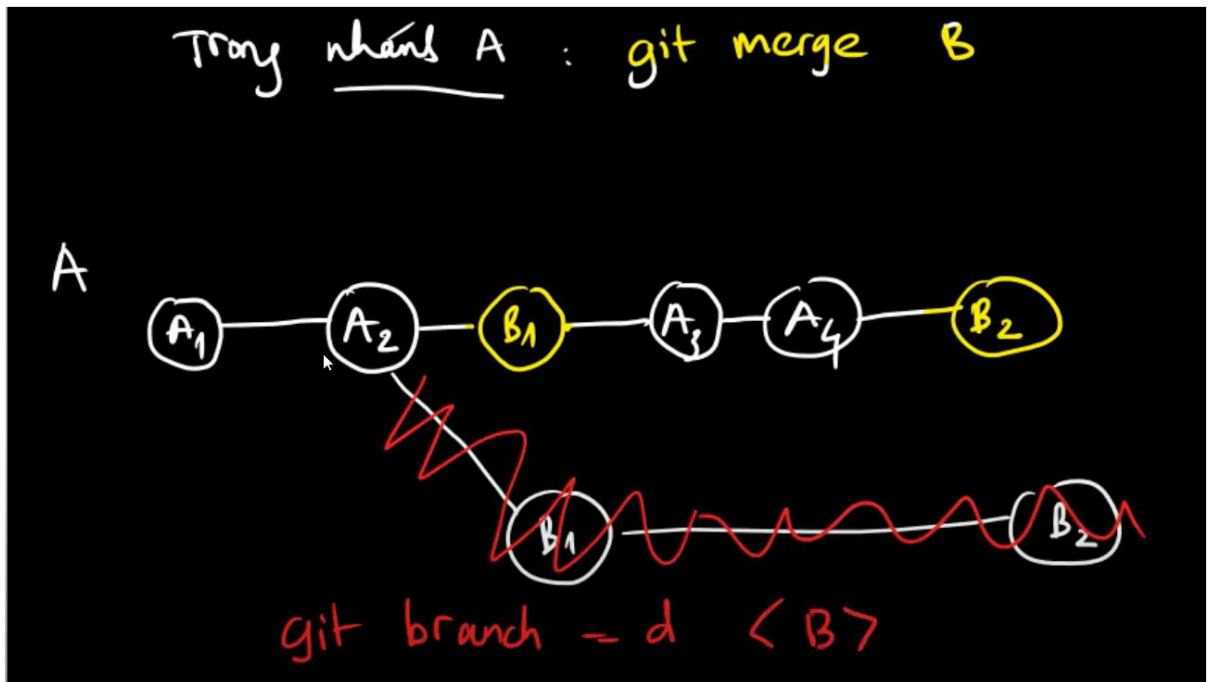
- `Git merge B` : gộp nhánh B vào nhánh chính(với B là tên nhánh bất kỳ)
- `Git log --oneline` : xem lại các commit thay đổi

Bài 13. Git Rebase - tái cơ sở cho một nhánh trong Git



- Git rebase “tên nhánh” :

Bài 14. Cách xóa nhánh trong GIT



- `git branch -d "B"` : xóa nhánh B (với B là tên nhánh)

Sau khi xóa xong nhánh B vẫn tồn tại trong file remotes, gõ lệnh

`git branch -a` để thấy rõ hơn như hình dưới

```
$ git branch -l
* master

Windows@TITVvn MINGW64 ~/OneDrive/Youtube/GIT/source/EX10/Dev1 (master)
$ git branch -a
* master
remotes/origin/B
remotes/origin/master

Windows@TITVvn MINGW64 ~/OneDrive/Youtube/GIT/source/EX10/Dev1 (master)
$ git branch -d origin B
error: branch 'origin' not found.
error: branch 'B' not found.

Windows@TITVvn MINGW64 ~/OneDrive/Youtube/GIT/source/EX10/Dev1 (master)
$ git branch -a
* master
remotes/origin/B
remotes/origin/master

Windows@TITVvn MINGW64 ~/OneDrive/Youtube/GIT/source/EX10/Dev1 (master)
$ |
```

Để xóa thì dùng câu lệnh `git push -d "B"` với B là tên nhánh

Kiểm tra lại bằng `git branch -a` sẽ thấy ko còn tồn tại nhánh B trong cả file remote nữa.

Bài 15. Git Reset - Hủy bỏ commit

`git reset --soft <commit id>`

Di chuyển HEAD về vị trí commit.
Trạng thái của stage và tất cả sự
thay đổi của file được giữ nguyên.

`git reset <commit id>`

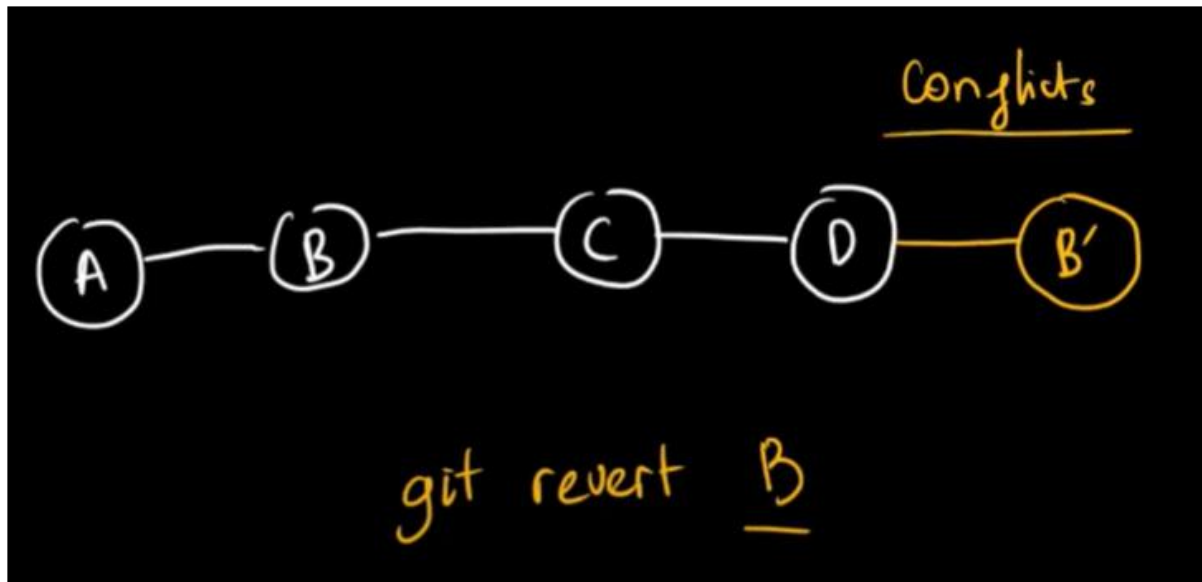
di chuyển HEAD về vị trí commit
reset, vẫn giữ tất cả thay đổi của
file, nhưng loại bỏ các thay đổi stage.
(`--mixed`)

`git reset --hard <commit id>`

Di chuyển con trỏ HEAD về vị trí
commit reset và loại bỏ tất cả
sự thay đổi của file.

Bài 16. Git Revert - Quay lại các commit trước đây

Nên sử dụng git revert hơn là git reset vì git revert sẽ giúp quay trở lại phần commit tại bất kỳ thời điểm nào mà mình mong muốn



git revert “địa chỉ của B” : quay trở lại phần commit tại địa chỉ B

```
b1cc725 Revert "B"
```

Địa chỉ của B là phần màu vàng

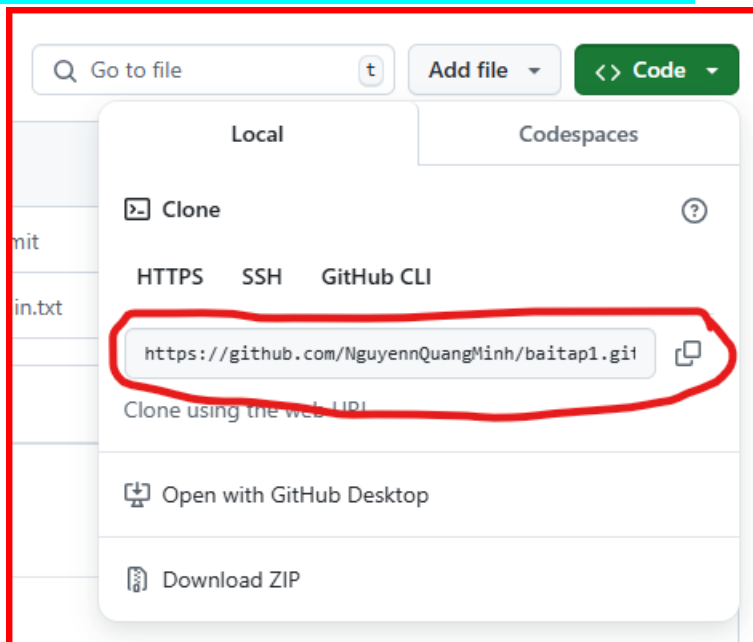
Làm việc với Github

(từ bài 17 trở đi)

chú ý: coi 1 repo của github tương tự như một repo bình thường

Bài 19. Cách clone dự án từ GitHub về máy

- Git clone “link https của github”



Tạo folder trên máy tính trước rồi clone dự án trên github vào đó vào folder đó

Bài 20. Đẩy dự án từ máy cá nhân lên GitHub

- B1: tạo 1 repo trên github
- B2: Vào folder muốn đẩy lên github trên máy tính và mở gitbash
- B3: `git init` : tạo file .git
- B4: thêm file .gitignore để bỏ qua những file ko muốn tải lên
- B5: `git add .` và `git commit -m " "`
- B6: `git remote add origin "https repo github"` : add dự án từ máy tính vào repo đã tạo trên github (chưa đăng nhập trên máy tính thì cần đăng nhập sau bước này)
- B7: `git branch -M main` : đổi tên nhánh master trên máy tính thành main để giống với nhánh chính trên github
- B8: `git branch -l` : xem lại tên nhánh trên máy đã đúng là main chưa
- B9: `git push -u origin main` : đẩy folder lên github

Bài 21. Fork và cập nhật dựa trên Repo của người khác trên GitHub

Lấy dự án của người khác về trên github của mình, pull về máy và chỉnh sửa, cập nhật và push lại lên trên github của mình

Bài 22. Cách tạo Pull Request trong Github

Đóng góp vào repo của người mà mình fork dự án của họ về

Bài 23. Cài đặt và sử dụng GitHub Desktop

Bài 24. Sử dụng Git và GitHub trong Visual Studio Code

Làm việc với Terminal của VS code tương tự như với git bash