**TÀI LIỆU KHÓA HỌC**

**Git Zero - Tự Học Git Siêu Tốc với GitHub & GitLab**

Tác giả: Hỏi Dân IT & Eric

Version: 1.0

[**Chapter 1: Bắt buộc xem - Không bỏ qua chương học này 3**](#_ejt11nhq0dp0)

[#1. Hướng Dẫn Sử Dụng Khóa Học Hiệu Quả 3](#_ely3h9od1ad0)

[#2 + #3. Hướng Dẫn Truy Cập Tài Liệu Khóa Học 5](#_xahoelobukgs)

[#4. Demo kết quả đạt được 6](#_b4pwvy2zyc9s)

[#5. Cách Dùng Udemy - Hỗ Trợ Hỏi Đáp Q&A 7](#_xrjc9lkm1294)

[#6. Về Tác giả 8](#_79bhrk5i6nez)

[**Chapter 2: Setup Environment 9**](#_hfi1wf1948ie)

[#7. Tổng quan về chapter 9](#_3us78pd57myp)

[#8. Cài đặt Git 9](#_m745gczh1u59)

[#9. Cài đặt Visual Studio Code 10](#_e8si6rgdvrri)

[#10. Cài đặt Google Chrome (Extra) 10](#_bvfks76ruu2s)

[#11. Tạo Tài Khoản Github 10](#_xvz57cmpds02)

[**Chapter 3: Tổng Quan Về Git 11**](#_uf6wmyexsora)

[#12. Tổng quan về chapter 11](#_a84autzg94v4)

[#13. Git là gì ? 11](#_tlt3jd6z51g)

[#14. Lịch sử ra đời của Git ? (Extra) 12](#_tsuolaow9gsu)

[#15. Git Workflow/Architecture 12](#_x4xm8ja3h3gn)

[#16. Phân biệt Git và Github ? 13](#_g0qnor2awsi6)

[#17. Thao tác với Git ? 14](#_f6ttt030wbp)

[**Chapter 4: Kỹ Năng Dùng Git mà Beginners Bắt Buộc Phải Biết 15**](#_64cbqjqfwwet)

[#18. Tổng quan về chapter 15](#_t3lrn4ms249v)

[#19. Khái niệm Repository 15](#_ue07ovtgeiyt)

[#20. Quy trình dùng Git thường gặp 16](#_z0gsz14g0qtm)

[#21. Config Git Email/Name 17](#_lrtubig64jjw)

[#22. Câu Lệnh git init 17](#_rvtqyjbn4ek2)

[#23. Câu Lệnh git status & git add 18](#_lco0fygjx0rz)

[#24. Câu Lệnh git commit 19](#_m48gra1m8d75)

[#25. Git Commit với Vim (Extra) 19](#_lorfhht7andj)

[#26. Câu Lệnh git remote 20](#_s4o60ddhl30f)

[#27. Câu Lệnh git push 21](#_p3yrlikypu80)

[#28. Phân biệt working directory/ staging area/ repository (Extra) 22](#_ptahttkxyp82)

[**Chapter 5: Thực Hành Quản Lý Mã Nguồn với Git 01 23**](#_aayh1oaz3i70)

[#29. Tổng quan về chapter 23](#_rhnh4jdt11d2)

[#30. Bài Tập tạo dự án tại máy tính bạn 23](#_n7drnga7o83c)

[#31. Chữa Bài Tập tạo dự án tại máy tính bạn 25](#_yt79p6m7fhig)

[#32. Bài Tập đẩy mã nguồn lên Github 25](#_h6hmzd3nmga)

[#33. Chữa Bài Tập đẩy mã nguồn lên Github 25](#_l5x2lmwn10cb)

[#34. Tổng Kết Quy Trình Coding dự án (Basic) 26](#_83wzapvhzjkg)

[#35. Câu Lệnh git ignore (Extra) 27](#_80rzcafr0t6o)

[#36. Câu Lệnh git clone 27](#_uqxce7o3p68)

[**Chapter 6: Thực Hành Quản Lý Mã Nguồn với Git 02 28**](#_pro29vmak650)

[#37. Tổng quan về chapter 28](#_2ncp79cr1cy5)

[#38. Bài Tập sử dụng Git với Remote Repository 29](#_gs2gdgedh3mk)

[#39. Chữa Bài Tập sử dụng Git với Remote Repository 30](#_pt0zsmdv1nfg)

[#40. Clone và Thay đổi Remote Repository (Extra) 30](#_bi8rbngeechn)

[**Chapter 7: Sử dụng Git với Branch 31**](#_fy3h4syvuvxy)

[#41. Tổng quan về chapter 31](#_mzjkk741taou)

[#42. Branch là gì ? 31](#_bl8jnuv7k4m0)

[#43. Master/Main Branch ? 31](#_eqhucoi7tbd)

[#44. Tạo mới Branch 32](#_kjo9bd7til31)

[#45. Luyện Tập sử dụng Git Branch 33](#_d97lvy7569az)

[#46. Git switch vs Git checkout 33](#_x9izzjrmsm0v)

[#47. Bài Tập Git Branch 34](#_h42ocipr34uv)

[#48. Chữa Bài Tập Git Branch 35](#_a8622t6lkvpw)

[**Chapter 8: Merge Branches 36**](#_qchvjfcjxnr2)

[#49. Tổng quan về chapter 36](#_fy7qui5w40h)

[#50. Cài Đặt Git Graph (Git HEAD) - Extra 36](#_ou13pbor6fbz)

[#51. Tại Sao Cần Merge Branch ? 37](#_butxcmsl7v2t)

[#52. Luyện Tập Git Merge 01 38](#_ulaediqekhgk)

[#53. Luyện Tập Git Merge 02 39](#_jmfiodfki33v)

[#54. Luyện Tập Git Merge 03 ( Merge tại Local) 40](#_c9uo4kapw5u9)

[#55. Luyện Tập Git Merge 03 ( Merge tại Github với Pull Request) 41](#_8taht3f9639u)

[#56. Merge Conflict 42](#_7pyxr6yv0k72)

[#57. Resolve Conflict 42](#_rbl1e259tdr0)

[#58. Bài Tập Merge Conflict 43](#_48z067v33j1m)

[#59. Chữa Bài Tập Merge Conflict 43](#_6yo6mop4jhk9)

[#60. Tổng kết các kiến thức đã học 44](#_4zkqisw0mlav)

[**Chapter 9: Thực Hành Quản Lý Mã Nguồn với GitLab 45**](#_acrbnt9l0mcc)

[#61. Tổng quan về chapter 45](#_d786gowz2xgc)

[#62. GitLab là gì ? 45](#_h5u28s4jvff2)

[#63. Tạo Tài Khoản GitLab 46](#_t04hpw50idf8)

[#64. Luyện Tập Sử Dụng GitLab 01 47](#_htp3ecefuma4)

[#65. Luyện Tập Sử Dụng GitLab 02 47](#_jkbzc75s5y)

[#66. Cảm Nhận Cá Nhân Mình về GitLab 48](#_t3xgkqk0x8y)

[**Đánh Giá (Review/Rating) Khóa Học 49**](#_qxd4hhbcxb4n)

## **Chapter 1: Bắt buộc xem - Không bỏ qua chương học này**

*Hướng dẫn sử dụng khóa học hiệu quả, đạt chất lượng cao nhất*

### #1. Hướng Dẫn Sử Dụng Khóa Học Hiệu Quả

Bạn vui lòng "xem video lần lượt" theo trình tự. Vì khóa học như 1 dòng chảy, video sau sẽ kế thừa lại kết quả của video trước đó.

1. Dành cho học viên "có ít thời gian"

Nếu bạn vội, cần học nhanh, hoặc "bạn đã biết rồi", thì "vẫn xem video, cơ mà không cần code theo".

Lưu ý: vẫn xem qua tài liệu khóa học để biết "video hướng dẫn gì".

Đã "Không xem video", thì cần "đọc giáo án".

Có như vậy mới biết khóa học nó làm cái gì.

2. Dành cho học viên "thông thường"

Nguyên tắc:

- Xem video lần lượt

- Xem video kết hợp với giáo án. Bạn không cần take note, vì những điều quan trọng đã có trong giáo án

- Bạn vui lòng code theo video.

Nếu bạn "code theo ý bạn", vui lòng "không hỏi khi có bugs".

Câu chuyện này giống như việc bạn đi khám bệnh, nhưng không tin lời bác sĩ

=> Nếu bạn giỏi, bạn làm luôn bác sĩ, còn đi khám bệnh làm gì.

- Bạn có thể "code theo ý bạn muốn", sau khi "đã kết thúc khóa học"

- Nếu bạn có thắc mắc (hoặc có ý tưởng/nhận thấy bugs), take note lại, bạn hỏi, rồi mình giải đáp.

Chứ không phải là "tự ý làm theo điều các bạn muốn".

Vì đa phần, các bugs trong khóa học mình đã fix hết rồi.

Nên là yên tâm để học theo bạn nhé.

3. Về cách code.

Bạn vui lòng code theo video, từ cách đặt tên biến, hàm. Vì mình đã tuân theo "convention tối thiểu" khi bạn đi làm đấy

4. Về bài tập thực hành

Đối với bài tập thực hành, bạn cứ code theo cách bạn hiểu, và kết hợp với "search Google, stackoverflow..."

KHÔNG DÙNG CHATGPT. Đây giống kiểu chưa học "phép tính", mà đã "dùng máy tính".

Nên nhớ 1 điều, trước 2023, không có chat gpt, thì mình học như thế nào ?

Khi bạn đã đi làm, bạn có quyền dùng cái gì bạn thích, còn với beginner, hãy biết say NO với CHAT GPT.

tương tự bạn dạy con bạn:

học lớp cấp 1: không chịu học tính nhầm => đưa luôn máy tính cho nó. Rời máy tính ra, là không biết làm phép tính

còn với học sinh cấp 2, 3 : dùng máy tính tùy thích

5. Về source của cả khóa học

Source code của cả khóa học ĐƯỢC CUNG CẤP (full cả khóa và theo từng video)

### #2 + #3. Hướng Dẫn Truy Cập Tài Liệu Khóa Học

//bỏ qua video này

### **#**4. **Demo kết quả đạt được**

1. Mục tiêu của khóa học

Git là một công cụ giúp quản lý phần mềm được sử dụng phổ biến, vì vậy, các mục tiêu trọng tâm của khóa học hướng tới, bao gồm:

1. **Git giúp bạn bảo vệ, sao lưu (backup) mã nguồn** (source code) của dự án lập trình, tránh tình trạng máy tính bạn bị hư hỏng, mất hết code.

Với Git, bạn có thể yên tâm là source code đã được sao lưu, có thể phục hồi theo ý bạn muốn.

1. **Git giúp bạn chia sẻ mã nguồn dự án, phối hợp làm việc nhóm (teamwork) trong dự án lập trình hiệu quả.**

Việc học Git sẽ hỗ trợ bạn theo dõi các khóa học lập trình dễ dàng, đồng thời đóng góp công sức vào xây dựng phần mềm nếu dự án có nhiều thành viên tham gia.

1. **Git là một công cụ đi làm mà gần như 99% các công ty lĩnh vực CNTT sử dụng**.

Luyện tập kỹ năng sử dụng Git, biết về Git, là một lợi thế cực kì lớn khi bạn đi thực tập và đi làm

2. Học viên nào có thể học ?

* Học viên có nhu cầu muốn sao lưu và kiểm soát version của phần mềm.
* Học viên có tư duy và đam mê về lập trình.
* Biết sử dụng terminal gõ câu lệnh là một lợi thế

### **#**5. **Cách Dùng Udemy - Hỗ Trợ Hỏi Đáp Q&A**

Lưu ý: không bỏ qua video này. Xem để biết cách sử dụng Udemy, cũng như cách đặt Q/A khi cần hỗ trợ (support)

1. Sử dụng trên máy tính

Xem hướng dẫn tài liệu chi tiết [tại đây](https://support.udemy.com/hc/en-us/sections/206457187-Course-Player)

* [Cách bắt đầu sử dụng khóa học](https://support.udemy.com/hc/en-us/articles/229603648-How-to-Use-The-Course-Player-and-Start-Your-Course) (bắt buộc xem)
* [Cách đặt câu hỏi cho khóa học](https://support.udemy.com/hc/en-us/articles/229233387-How-to-Ask-a-Question-About-a-Course-You-re-Taking) (bắt buộc xem)
* Hướng dẫn cách sử dụng Q&A
* Hướng dẫn cách liên hệ Instructor qua Message
* [Cách sử dụng phím tắt](https://support.udemy.com/hc/en-us/articles/13687901918999-How-to-Use-Keyboard-Shortcuts)
* [Take note trực tiếp trên video đang xem](https://support.udemy.com/hc/en-us/articles/360038492834-How-to-Create-And-Use-Notes-on-a-Browser)

2. Sử dụng trên điện thoại

Udemy có hỗ trợ ứng dụng trên điện thoại Android/IOS

Xem hướng dẫn tài liệu chi tiết [tại đây](https://support.udemy.com/hc/en-us/categories/204119668-Mobile)

### **#**6. **Về Tác giả**

Về tác giả:

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Website chính thức: <https://hoidanit.vn/>

Youtube “Hỏi Dân IT” : <https://www.youtube.com/@hoidanit>

Tiktok “Hỏi Dân IT” : <https://www.tiktok.com/@hoidanit>

Fanpage “Hỏi Dân IT” : <https://www.facebook.com/askITwithERIC/>

Udemy Hỏi Dân IT: <https://www.udemy.com/user/eric-7039/>

Nếu bạn muốn nói chuyện với mình (giao lưu trao đổi võ công :v), có thể xem mình livestream trực tiếp tối thứ 2 & thứ 5 hàng tuần trên [Youtube Hỏi Dân IT](https://www.youtube.com/@hoidanit/streams)

## **Chapter 2: Setup Environment**

*Cài đặt & chuẩn bị môi trường thực hiện dự án*

### #7. Tổng quan về chapter

Lưu ý: không bỏ qua chapter này. Việc bạn “đã biết” công cụ, không đồng nghĩa với việc bạn sử dụng nó hiệu quả.

Ngoài việc cài đặt công cụ, đang còn thực hiện cấu hình công cụ đấy nữa.

Vì vậy, chapter này đảm bảo rằng môi trường thực hiện khóa học của chúng ta là giống hệt nhau, hạn chế tối đa bug/lỗi có thể xảy ra.

Các công cụ sẽ cài đặt trong chương học này:

* Git
* Visual Studio code (phục vụ coding và sử dụng Git)
* Google Chrome (phục vụ coding)
* Tài khoản Github (phục vụ lưu trữ code)

### #8. Cài đặt Git

Link download git: <https://git-scm.com/downloads>

Kiểm tra xem git đã cài đặt chưa, bằng câu lệnh:

**git --version**

1.Cài đặt với windows

Lưu ý về show more options trên windows từ windows 11 trở đi.

2.Cài đặt với macos/linux

Tham khảo: <https://www.youtube.com/watch?v=B4qsvQ5IqWk>

### **#**9**. Cài đặt Visual Studio Code**

Công cụ code trong dự án sử dụng VSCode, 1 Editor hoàn toàn miễn phí

Link download:

<https://code.visualstudio.com/download>

### **#**10**. Cài đặt Google Chrome (**Extra)

Ở đây, sử dụng google chrome vì nó là ứng dụng phổ biến nhất (trình duyệt web được dùng nhiều nhất)

Bạn nên dùng Google Chrome (thay vì Firefox/Edge…) để đảm bảo rằng thao tác sử dụng giữa bạn và mình là giống nhau (tránh gây khó khăn không cần thiết)

Lưu ý: sử dụng version tiếng anh

=> change language

Mục tiêu:

* Sử dụng Google Chrome để chạy ứng dụng web
* Ngôn ngữ hiển thị là Tiếng Anh
* Set default app là google chrome (nếu nó mở app, thì chạy với google chrome)

Xem hướng dẫn setup default app cho windows [tại đây](https://support.microsoft.com/en-us/windows/change-your-default-browser-in-windows-020c58c6-7d77-797a-b74e-8f07946c5db6)

### #11. Tạo Tài Khoản Github

//todo

## Chapter 3: Tổng Quan Về Git

*Lịch sử ra đời và ý nghĩa của Git*

### #12. Tổng quan về chapter

//todo

### #13. Git là gì ?

1.Git là gì ? (What)

<https://git-scm.com/>

**Git** là một phần mềm **miễn phí** và **mã nguồn mở**, phục vụ mục đích **kiểm soát version** cho các dự án từ nhỏ tới lớn, đảm bảo hiệu năng cao nhất.

Git là version control system

=> giải thích ý nghĩa của version control

có nhiều version control system => but git wins

2.Tại sao dùng Git ? (Why)

tác dụng của GIT:

1. Có thể thấy được sự thay đổi của files (theo thời gian)

2. "time travel" (rollback/revert) -> quay lại version bất kỳ

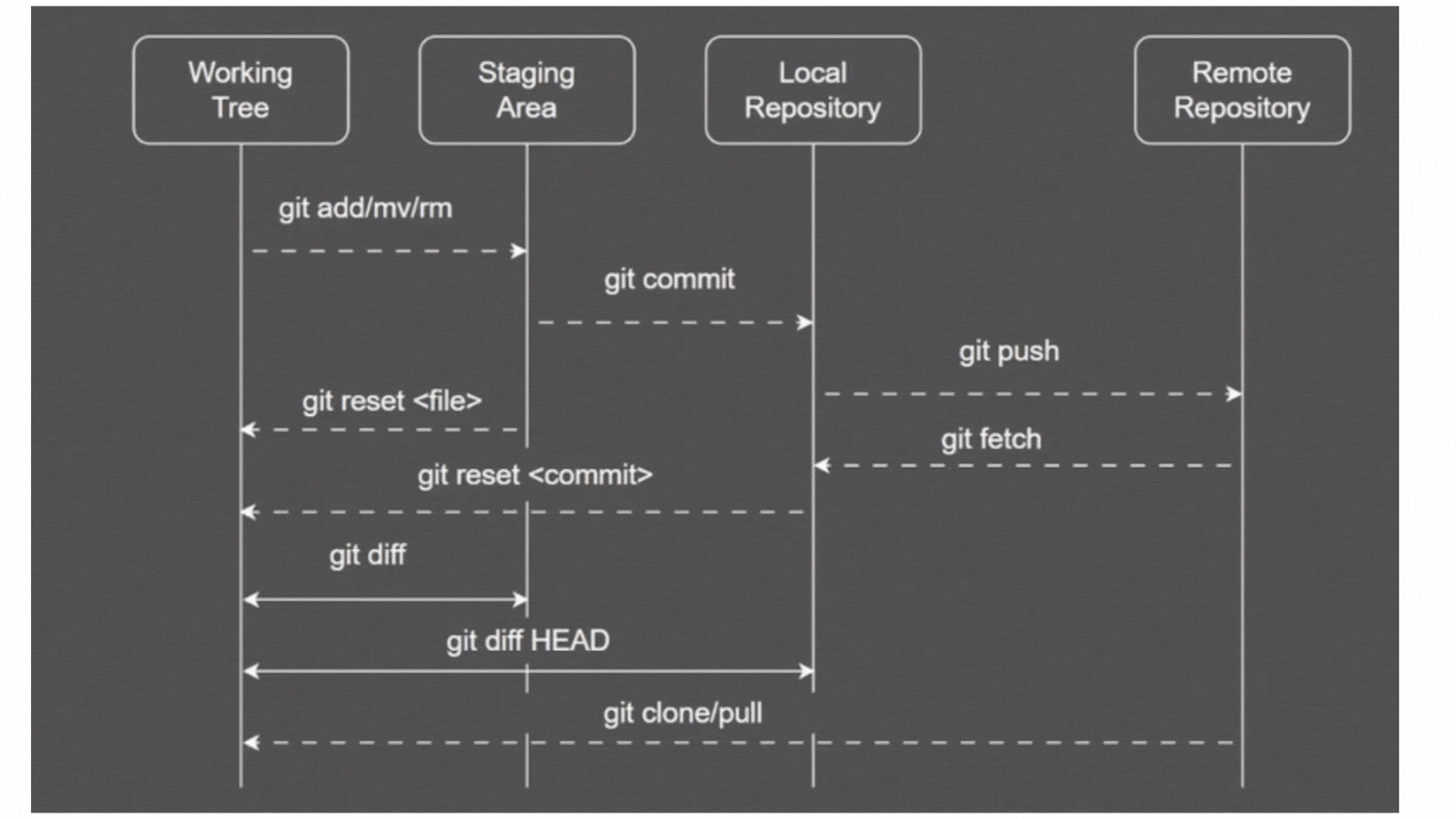
rollback/revert tới version nhất định

### #14. Lịch sử ra đời của Git ? (Extra)

Tham khảo: <https://en.wikipedia.org/wiki/Git>

### #15. Git Workflow/Architecture

Kiến trúc tổng quát của Git:



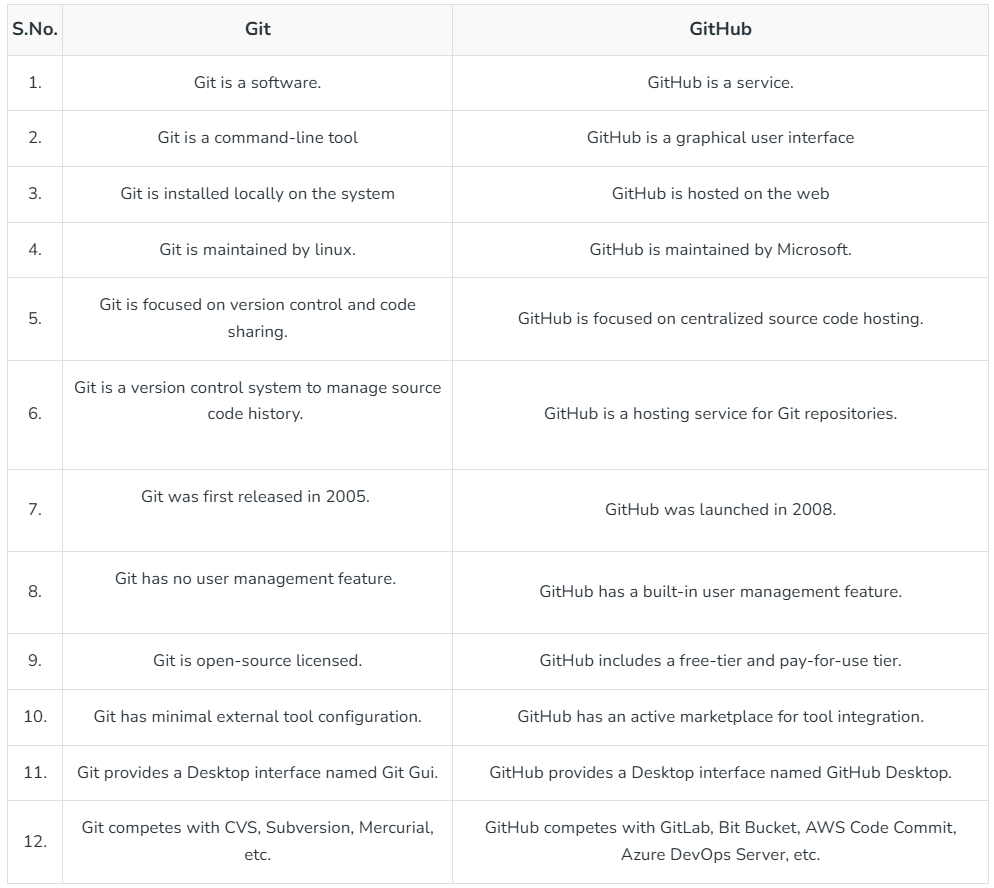
### #16. Phân biệt Git và Github ?

Git và Github là 2 công cụ khác hẳn nhau

Hãy tưởng tượng, nếu Git là 1 bức ảnh (image) lưu tại máy tính bạn, thì Github chính là Google Photo, nơi lưu trữ ảnh online

Tham khảo:

<https://www.geeksforgeeks.org/difference-between-git-and-github/>



### #17. Thao tác với Git ?

Có thể dùng qua GUI hoặc các phần mềm ?

1. Git client (GUI) nổi tiếng

[Gitkraken](https://www.gitkraken.com/)

[SourceTree](https://sourcetreeapp.com/)

[tortoisegit](https://tortoisegit.org/)

[Github Desktop](https://github.com/apps/desktop)

[Tower](https://www.git-tower.com/windows)

2. Nên sử dụng GUI hay terminal

GUI: graphical user interface

Khóa học sẽ sử dụng terminal.

Có thể dùng với vscode (tích hợp sẵn)

## Chapter 4: Kỹ Năng Dùng Git mà Beginners Bắt Buộc Phải Biết

*Học và sử dụng các câu lệnh thiết yếu nhất của Git*

### #18. Tổng quan về chapter

//todo

### #19. Khái niệm Repository

Repository: thùng chứa

Bao gồm : local repository và remote repository

Repository chịu trách nhiệm lưu trữ source code và lịch sử version của source code

### #20. Quy trình dùng Git thường gặp

Nhắc lại video [#19](#_ue07ovtgeiyt):

**…or create a new repository on the command line**

echo "# my-first-project" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin …

git push -u origin main

**…or push an existing repository from the command line**

git remote add origin …

git branch -M main

git push -u origin main

Có 2 trường hợp khóa học sẽ giải quyết cho bạn, ứng với 2 phần recommend ở trên:

**Trường hợp 1:** Bạn chưa sử dụng git cho dự án cá nhân của bạn

Đây là lúc bạn cần “**tạo mới**” git repository, tuy nhiên, phạm vi của nó

là “**local repository**”

Ví dụ: bạn đã và đang coding tại máy tính cá nhân, muốn backup source code này lên cloud, hoặc muốn share source code để cho người khác code cùng bạn

**Trường hợp 2:** Bạn đã sử dụng git để quản lý mã nguồn, đã sử dụng/tạo remote repository.

Ví dụ: source code bạn kéo về đang lưu trữ tại **repository A**, bây giờ, bạn muốn đẩy code sang **repository B**

Trong các khóa học coding của tác giả Hỏi Dân IT, bạn kéo source code starter của tác giả về (repository A), sau đấy, bạn muốn quản lý mã nguồn của bạn tại chính github của bạn (đẩy code lên repository B)

### #21. Config Git Email/Name

Tài liệu: <https://stackoverflow.com/a/66108560>

Mục đích: dùng để định danh bạn là ai khi làm việc teamwork

Kiểm tra cấu hình mặc định của Git:

**git config --list**

**git config --list --show-scope**

Cấu hình Email/Name:

**git config --global user.name "your Name"**

**git config --global user.email "your\_email@example.com"**

### #22. Câu Lệnh git init

Tài liệu: <https://git-scm.com/docs/git-init>

Mục đích: khởi tạo an empty Git repository (local)

Khởi tạo một thư mục trống chứa Git (empty, vì chưa có lịch sử sử dụng Git)

Local: vì phạm vi sử dụng, là tại máy tính của chúng ta

Sử dụng câu lệnh: **git init**

(Init: initialize - khởi tạo giá trị)

Kết quả của câu lệnh git init, là tạo ra thư mục **.git**

//Cách kiểm tra

thư mục .git (lưu ý máy windows và macos)

Mở git bash ( **ls -a** )

**//Lưu ý về gõ câu lệnh 2 lần trở lên**

Trong hầu hết các trường hợp thường gặp, việc gõ git init từ 2 lần trở lên, hoàn toàn không gây ra hậu quả nghiêm trọng (an toàn)

### #23. Câu Lệnh git status & git add

Tài liệu:

<https://git-scm.com/docs/git-status>

<https://git-scm.com/docs/git-add>

1.Định nghĩa đơn giản, chưa đầy đủ

git status là cách bạn kiểm tra trạng thái của file

git add là thêm file vào git (chưa commit, stage area)

2. Thực hành với VSCode

**Bước 1:**  Khởi tạo dự án với git init

**Bước 2:** Tạo file mới

//index.text

git status

git add

### #24. Câu Lệnh git commit

Tài liệu: <https://git-scm.com/docs/git-commit>

Mục đích: ghi nhận sự thay đổi đối với một khối lượng công việc đã làm

1. Thực hành với VSCode

Tạo thêm about.html

git status

git add

git commit

Tạo thêm project.html

git status

git add

git commit

### #25. Git Commit với Vim (Extra)

để commit: nhấn chữ i

để thoát: nhấn ESC , sau đấy là dấu :

wq => write and quit

git commit -m "message" => dễ hơn

Nếu bạn muốn thoát ra khỏi vim, mà không cần commit, sử dụng:

<https://stackoverflow.com/a/11828573>

### #26. Câu Lệnh git remote

Yêu cầu: cần tạo remote repository trên Github (tham khảo video [#19](#_ue07ovtgeiyt))

Tài liệu: <https://git-scm.com/docs/git-remote>

Git có 2 loại repository : local và remote

**Local Repository:** repository tại máy tính của bạn

**Remote Repository**: được tạo trên các máy chủ cloud, ví dụ Github, Gitlab

Nếu bạn gõ câu lệnh **git init** => đây là việc bạn tạo ra một local repository (vì nó đang nằm tại máy tính của bạn)

Để có thể chia sẻ mã nguồn (cho người khác code cùng), hoặc backup code (tránh trường hợp máy tính hư, mấy hết code), **chúng ta cần đồng bộ hóa local repository và remote repository (sync data)**

Các câu lệnh thường dùng với remote:

//đặt tên cho remote là origin, và url = gitusername@domain.com:myproject.git

**git remote add origin** gitusername@domain.com:myproject.git

//hiển thị remote name và remote url

**git remote -v**

//thông tin chi tiết của remote có tên là origin

**git remote show origin**

### #27. Câu Lệnh git push

Tài liệu: <https://git-scm.com/docs/git-push>

<https://stackoverflow.com/questions/15381198/remove-credentials-from-git>

Mục đích: đẩy các commit mới từ local repository lên remote repository

Hiểu đơn giản hơn, đẩy code từ máy của bạn (local) lên máy tính trên cloud (remote)

Cú pháp được dùng:

**git push** tên-remote tên-nhánh-code

Ví du:

**git push origin master**

//origin: là tên remote (tham khảo video #26)

master: tên nhánh code (branch - sẽ học về khái niệm branch tại các chapter tiếp theo)

/// sử dụng git push với -u

### #28. Phân biệt working directory/ staging area/ repository (Extra)

Khi sử dụng Git tại máy tính của bạn, Git được chia làm 3 không gian chính:

* Working directory (working tree)
* Staging area
* Local Repository

1. Phân biệt các thành phần

**Working directory (working tree):**

* Là nơi bạn thao tác với code (không gian code)
* Bạn có thể thêm mới, sửa đổi, xóa files …

**Staging Area:**

* Là nơi lưu trữ các file “đang thay đổi”
* Bao gồm các file dùng để commit (lưu lại lịch sử/snapshot)

**Repository (local repository):**

* Chính là thư mục .git
* Lưu trữ lại lịch sử version của git (commit)

2.Cách câu lệnh được sử dụng

git add : chuyển file từ working directory => staging area

git commit : chuyển từ staging area => git repository (local)

git push : chuyển từ local repository => remote repository

## Chapter 5: Thực Hành Quản Lý Mã Nguồn với Git 01

*Luyện tập các câu lệnh cơ bản nhất của git*

### #29. Tổng quan về chapter

//todo

### #30. Bài Tập tạo dự án tại máy tính bạn

Mục tiêu: làm quen với quá trình coding một dự án sử dụng Git để quản lý mã nguồn.

**Bước 1:** Tạo 1 thư mục lưu code (đặt tên folder là dự án của bạn)

Tạo thư mục: **chapter5-01**

Lưu ý: không nên đặt tên dự án coding bằng tiếng việt hoặc sử dụng ký tự đặc biệt

**Bước 2:** Coding vào dự án

Tạo file **index.html** bên trong dự án thực hành

//index.html

Nội dung file, đảm bảo hiển thị dòng text : Hello World with git

**Bước 3:** Tạo Git Local Repository (hoặc dễ nhớ hơn : khởi tạo git cho dự án)

Sử dụng câu lệnh: git init

**Bước 4:** Tạo commit đầu tiên

Sử dụng các câu lệnh:

git add .

git commit -m “your message”

**Bước 5:** Update source code thực hành

Tạo thêm vào dự án file **chapter1.html**

Nội dung file tùy ý, tuy nhiên, cần đảm bảo, từ file index.html, có thể điều hướng sang file chapter1.html

Gợi ý: sử dụng thẻ a tại file index.html, với href=”./chapter1.html”

**Bước 6:** Tiến hành commit ứng với chapter 1

Sử dụng 2 câu lệnh git add và git commit để lưu lịch sử

**Bước 7:** Update source code thực hành

Tạo thêm vào dự án file **chapter2.html**

Nội dung file tùy ý, tuy nhiên, cần đảm bảo, từ file index.html, có thể điều hướng sang file chapter2.html

Gợi ý: sử dụng thẻ a tại file index.html, với href=”./chapter2.html”

**Bước 8:** Tiến hành commit ứng với chapter 2

Sử dụng 2 câu lệnh git status và git commit để lưu lịch sử

Kết thúc bước 8, chúng ta chợt nhận ra nhu cầu, cần “backup” source code, tránh trường hợp máy tính bị hỏng (mất hết code)

Lưu ý: code đã commit, tuy nhiên đang lưu “local”, tức là đang lưu tại máy tính bạn

### #31. Chữa Bài Tập tạo dự án tại máy tính bạn

//todo

### #32. Bài Tập đẩy mã nguồn lên Github

Mục tiêu: tiếp tục video [#30](#_n7drnga7o83c), sử dụng quản lý mã nguồn với Github

**Bước 1:** Tạo git remote repository

Tại giao diện github, tạo mới repository, đặt tên **chapter5-01**

Bước 2 và bước 3: sử dụng các câu lệnh mà github gợi ý

**Bước 2:** Set remote

Mục đích, là nói cho git biết, dự án tại máy tính bạn, sẽ đẩy code lên repository nào trên github

**git remote add origin** gitusername@domain.com:myproject.git

**Bước 3:** Đẩy code từ máy tính cá nhân lên remote repository

Sử dụng câu lệnh git push

git push origin master

**Bước 4:** Tạo commit mới, và đẩy code

Tạo thêm chapter3.html

Tiến hành commit và đẩy code.

Sử dụng option -u để lần sau gõ code ngắn hơn:

git push -u origin master

git push

### #33. Chữa Bài Tập đẩy mã nguồn lên Github

//todo

### #34. Tổng Kết Quy Trình Coding dự án (Basic)

Quá trình dùng git cơ bản nhất:

Bạn đã và đang coding tại máy tính, xuất hiện nhu cầu muốn backup code, tránh tình trạng máy tính hư, là mất hết code, thì thực hiện các câu lệnh sau:

**git init** : chỉ cần làm 1 lần, mục đích là khởi tạo git cho dự án

git status : có thể dùng (hoặc không cần dùng), mục đích là xem trạng thái của files

**git add** : dùng câu lệnh này để thêm files vào staging area, chuẩn bị cho việc commit

**git commit** : tạo ra version của phần mềm (snapshot), đây chính là cách lưu lịch sử.

**git remote** : chỉ làm 1 lần, dùng để kết nối local repository tới remote repository

**git push**: dùng để đẩy code từ local lên remote repository

Tiếp tục code thêm vào dự án:

Chỉ cần thực hiện 3 câu lệnh sau, bỏ qua git init và git remote:

**git add**

**git commit**

**git push**

### #35. Câu Lệnh git ignore (Extra)

Tài liệu: <https://git-scm.com/docs/gitignore>

Mục đích: loại bỏ các files mà bạn không muốn git quản lý, ví dụ như api key, mật khẩu…

Cách sử dụng (basic nhất):

Tạo file **.gitignore** tại root

Khi báo nội dung cho file trên

Test với git

### #36. Câu Lệnh git clone

Tài liệu: <https://git-scm.com/docs/git-clone>

Mục đích: clone (sao chép) source code của một dự án, thông thường là copy remote repository

**Trường hợp 1:** bạn clone empty repository

* Bạn không cần gõ lại câu lệnh git init và git remote
* Bạn chỉ code, commit và push code

**Trường hợp 2:** bạn clone 1 source code đã code sẵn

* Bạn thao tác giống hệt trường hợp 1
* Chỉ có 1 điểm khác biệt là, khi kéo code về, bạn đã có sẵn code (note empty)

## Chapter 6: Thực Hành Quản Lý Mã Nguồn với Git 02

*Luyện tập các câu lệnh cơ bản nhất của git*

### #37. Tổng quan về chapter

Quy trình sử dụng git tại chapter 5, gồm 3 bước:

**1.Bạn coding tại máy tính bạn**

git init, git add, git commit

**2. Bạn tạo remote repository** trên Github (để backup code)

git remote (set remote url)

git push (đẩy code)

**3. Bạn tiếp tục code, và tiến hành đẩy code lên remote repository**

git add, git commit, git push

Quy trình sử dụng git tại chapter này (chapter 6):

**1.Bạn tạo remote repository**

**2.Bạn clone dự án** về, và **tiến hành coding vào đấy**

**git clone**

git add, git commit, git push

=> không cần dùng 2 câu lệnh : git init và git remote (do khi clone về, mặc định nó đã khởi tạo git, và set git remote)

**3. Bạn tiếp tục code, và tiến hành đẩy code lên remote repository**

git add, git commit, git push

### #38. Bài Tập sử dụng Git với Remote Repository

Thực hiện lần lượt các bước sau:

**Bước 1:** Tạo một remote repository

Tại giao diện github, tạo mới repository, đặt tên **chapter6-01**

**Bước 2:** Clone repository trên về máy tính của bạn

git clone…

**Bước 3:** Tạo file index.html, nội dung: hello world

**Bước 4:** Tiến hành commit và đẩy code lên github

git add

git commit

Lưu ý: khi bạn dùng câu lệnh git init, nó khởi tạo nhánh code là **master**

Khi bạn dùng github (clone), nhánh code khởi tạo là **main**

Câu chuyện về nhánh code, sẽ được mình hướng dẫn và giải thích chi tiết tại chapter tiếp theo.

Chuyển qua nhánh master: (cho giống chapter 5)

**git switch -c master**

git push -u origin master

**Bước 5:** Tạo file chapter6.html

Tiến hành commit và đẩy code

git add

git commit

git push

### #39. Chữa Bài Tập sử dụng Git với Remote Repository

//todo

### #40. Clone và Thay đổi Remote Repository (Extra)

Mục đích:

Bạn clone code từ Repository A, tiến hành coding, và đẩy code lên Repository B

**Bước 1:** chọn dự án mà bạn muốn sẽ clone (repository A)

Ví dụ: <https://github.com/haryphamdev/udemy-ui-not-hiding-when-using-full-screen>

**Bước 2:** Tạo remote repository **của bạn** (repository B)

**Bước 3:** Bạn clone dự án

git clone

Sau đấy, bạn có thể tiến hành code, commit, bla bla … tùy thích

Nếu bạn push code, sẽ bị lỗi (do bạn không có quyền truy cập vào repository A), đồng thời, bạn cũng muốn đẩy code lên repository B (của bạn)

**Bước 4:** Bạn cần thay đổi remote url

git remote

git remote show origin

git remote set-url origin new-url

**Bước 5:** Đẩy code

git push -u origin master

**Bước 6:** Tiến hành coding như bình thường

git add

git commit

git push

## Chapter 7: Sử dụng Git với Branch

*Sử dụng branch để chia tách code với git*

### #41. Tổng quan về chapter

//todo

### #42. Branch là gì ?

Ý tưởng về tree (mỗi node là 1 commit)

Branch là cách bạn “nhân bản code” tại 1 thời điểm bất kỳ, sau đấy phát triển song song mà không làm ảnh hưởng tới nhánh code cũ.

Mục đích:

Việc chia tách branch, giúp làm công việc độc lập mà không ảnh hưởng tới nhau

(chỉ ảnh hưởng nếu merge branch - sẽ đề cập sau)

### #43. Master/Main Branch ?

1.Branch Master

Bạn bắt buộc phải code trên một “branch”

Mặc định (khi cài đặt git), chúng ta đã được khởi tạo branch “master”

Master là một nhánh code, tương tự như cách nhánh code khác (sẽ học cách tạo thêm nhánh code sau).

Đối với nhiều người, master branch là nhánh code final. Khi cần code, code trên nhánh code khác, rồi merge code vào.

2.Tên default là main hay master

<https://github.com/github/renaming?tab=readme-ov-file#why-main>

<https://redis.io/docs/latest/commands/slaveof/>

Việc chọn tên main hay master, không ảnh hưởng gì tới code của bạn, và phụ thuộc vào quan điểm của mỗi người.

Bạn coding trên main hay master, nó chỉ là tên nhánh code, không ảnh hưởng gì tới code của bạn.

### #44. Tạo mới Branch

Tạo project, đặt tên là **chapter7-01**

Câu lệnh dùng để biết chúng ta đang đứng tại nhánh code nào:

**git branch**

(để ý dấu \* )

1.Tạo mới Branch

//tạo mới

**git branch branch-name**

Khi tạo mới 1 nhánh code, nó sẽ dựa vào nơi bạn đang đứng (HEAD)

=> vị trí bạn tạo nhánh code rất quan trọng

2. Đổi nhánh code

**git switch branch-name**

### #45. Luyện Tập sử dụng Git Branch

1.Tạo mới branch và switch nhánh code

**git switch -c branch-name**

C: create

2.Cách trường hợp hay gặp khi chuyển nhánh code

**Trường hợp 1:** Modify gây ra conflict (sửa đổi file đã có sẵn), rồi chuyển nhánh code

sửa cùng 1 file => cần commit, vì file đã được git tracking

**Trường hợp 2:** Tạo mới file, rồi chuyển nhánh code

Tạo mới file => sẽ tạo mới file tại branch khác, vì file chưa được git tracking

Kinh nghiệm với beginners:

nên commit trước khi chuyển nhánh code (vì có thể xóa commit)

### #46. Git switch vs Git checkout

<https://git-scm.com/docs/git-checkout>

git checkout là cú pháp cũ, làm nhiều hơn so với việc, chỉ đơn thuần tọa mới branch

**git checkout -b branch-name**

git switch là cú pháp mới, và giới hạn chỉ thao tác với branch

**git switch -c branch-name**

### #47. Bài Tập Git Branch

**Bước 1:** Tạo dự án thực hành tại máy tính của bạn

Tạo project, đặt tên là **chapter7-02**

**Bước 2:** Tạo file main.html, nội dung hello word

**Bước 3:** Khởi tạo git, và tiến hành commit trên nhánh code hiện tại

git init

git add

git commit

**Bước 4:** Tạo git repository trên Github (remote repository), đặt tên là **chapter7-02**

**Bước 5:** Tiến hành đẩy code lên github (ứng với nhánh code master)

git remote…

git push origin master

**Bước 6:** Tạo thêm nhánh code: **chapter1 (từ nhánh master)**

git switch -c chapter1

**Bước 7:** Tạo file chapter1.html, nội dung: chapter 1

**Bước 8:** Tiến hành commit

git add

git commit

**Bước 9:** Tạo thêm nhánh code: **chapter2 (từ nhánh master)**

Cần chuyển qua nhánh master, rồi mới tạo mới nhánh chapter 2. Vì nếu không chuyển qua master, chapter2 sẽ kế thừa nhánh chapter1

git switch master

git switch -c chapter2

**Bước 10:** Tạo file chapter2.html, nội dung: chapter 2

**Bước 11:** Tiến hành commit

git add

git commit

**Bước 12:** Tạo thêm nhánh code: **chapter3 (từ nhánh master)**

Cần chuyển qua nhánh master, rồi mới tạo mới nhánh chapter 3. Vì nếu không chuyển qua master, chapter3 sẽ kế thừa nhánh chapter2

git switch master

git switch -c chapter3

**Bước 13:** Tạo file chapter3.html, nội dung: chapter 3

**Bước 14:** Tiến hành commit

git add

git commit

**Bước 15:** Tiến hành đẩy code lên github

//origin: là tên của remote, tượng trưng cho url của remote repository

(git remote/git remote show origin)

Đẩy riêng lẻ từng nhánh code

**git push origin branch-name**

Đẩy tất cả branch

**git push origin --all**

### #48. Chữa Bài Tập Git Branch

//todo

## Chapter 8: Merge Branches

*Merge branch và xử lý conflict với git*

### #49. Tổng quan về chapter

//todo

### #50. Cài Đặt Git Graph (Git HEAD) - Extra

1.Cài đặt Git Graph

Với VSCode, cài đặt thêm extension: Git Graph

<https://marketplace.visualstudio.com/items?itemName=mhutchie.git-graph>

2.Git HEAD

Git HEAD, hay còn gọi là HEAD, là một con trỏ đặc biệt, trỏ tới commit bạn đang đứng (thông thường là commit mới nhất)

Tham khảo: <https://stackoverflow.com/a/54935492>

Sử dụng câu lệnh:

**cat .git/HEAD**

* Nếu kết quả tương tự: **ref: refs/heads/master**, có nghĩa là bạn đang đứng trên 1 nhánh code (branch), và HEAD đang trỏ tới commit mới nhất của branch đấy
* Nếu kết quả tương tự: a3c485d9688e3c6bc14b06ca1529f0e78edd3f86

Có nghĩa là, bạn đang tại 1 commit ID, và HEAD đang trở tới commit đấy

Sử dụng câu lệnh: **git log**

### #51. Tại Sao Cần Merge Branch ?

1. Merge Code là gì ?

Nếu bạn merge code từ (from) branch A, vào (to) branch B, có nghĩa là, bạn sẽ cộng gộp tất cả commit của branch A vào branch B

**Có bao nhiêu loại merge** (basic) ?

Với level cơ bản (beginners), có 2 hình thức chính:

* Fast forward (master ko code)

Không tạo thêm commit khi merge

* Không phải fast forward (tạo thêm commit khi merge)/ master đã có commit

2. Câu lệnh dùng để merge code ?

<https://git-scm.com/docs/git-merge>

Bạn đứng tại branch, và muốn merge branch khác:

**git merge branch-name**

### #52. Luyện Tập Git Merge 01

Mục đích: merge branch với code được tạo sẵn (clone từ Github)

Hình thức sử dụng: Fast forward

Các bước thực hiện:

**Bước 1:** Clone dự án chapter8-01-starter

<https://github.com/hoidanit-live/chapter8-01-starter>

Visualize trên github: <https://github.com/hoidanit-live/chapter8-01-starter/network>

**Bước 2:** Merge code

Lưu ý: chỉ merge đúng 01 nhánh code

* Cần chuyển qua nhánh code mà bạn muốn được nhận code

**git switch main**

* Tiến hành merge code

**git merge branch-name**

**Bước 3:** Đẩy code lên github

Tạo dự án dự án **chapter8-01-final**

set remote

**git push --all**

//check phần visualize (network) trên github

//check visualize với Git Graph

### #53. Luyện Tập Git Merge 02

Mục đích: merge branch với code tại máy tính (tự coding local)

Hình thức sử dụng: Fast forward

Các bước thực hiện:

**Bước 1:** Tạo dự án thực hành với tên : **chapter8-02**

**Bước 2:** Khởi tạo git và tiến hành commit

git init

git add .

git commit

**Bước 3:** Tạo nhánh code mới và commit

git switch -c branch-name

git add .

git commit

**Bước 4:** Merge code

//todo

### #54. Luyện Tập Git Merge 03 ( Merge tại Local)

Mục tiêu: Không sử dụng fast forward, lưu ý chưa đề cập tới chuyện conflict code

Các bước thực hiện (Tiếp tục video #53):

**Bước 1:** Tạo repository trên github, đặt tên **chapter8-03**

Tiến hành đẩy code lên github

**Bước 2:** Merge code tại local

//todo

Giải thích git merge strategies:

Tài liệu: <https://git-scm.com/docs/merge-strategies>

**1. Tại sao cần git merge strategy** (chiến lược/thuật toán để merge code):

Git cần biết cách merge code như thế nào cho nó hiệu quả, để phục vụ các trường hợp khác nhau.

Tuy nhiên, trong hầu hết các trường hợp hay gặp, việc dùng cơ chế mặc định của Git (ort strategy) đã giải quyết rất tốt cho merge code.

Vì vậy, với level của beginners, các bạn có thời gian thì tìm hiểu thêm về git merge strategies.

Còn dùng với cơ chế mặc định đã giải quyết hầu hết các bài toán hay gặp rồi các bạn nhé

**2. Về 'ort' strategy**

Trong video, khi merge code thành công, xuất hiện dòng text:

**Merge made by the 'ort' strategy**

(ort là viết tắt của: Ostensibly Recursive’s Twin)

Từ version git 2.33 trở đi, 'ort' strategy là chiến lược merge code mặc định (thay vì recursive strategy)

Nếu bạn xem các video youtube, hoặc đọc các bài post, có thể thấy dòng text:

**Merge made by the 'recursive' strategy.**

So với 'recursive' strategy, 'ort' có ưu điểm:

Faster: xử lý large repositories nhanh hơn

More accurate: độ chính xác cao hơn

Better conflict handling: giảm tối thiểu số lượng conflict khi merge code

### #55. Luyện Tập Git Merge 03 ( Merge tại Github với Pull Request)

Mục tiêu: Không sử dụng fast forward, lưu ý chưa đề cập tới chuyện conflict code

Các bước thực hiện (Tiếp tục video #54):

**Bước 1:** Tạo repository trên github, đặt tên **chapter8-03-final**

Tiến hành đẩy code lên github

**Bước 2:** Merge code với Github

//todo

### #56. Merge Conflict

//demo conflict

Tạo dự án : đặt tên **chapter8-04**

1. Conflict là gì

Conflict (xung đột) là khi merge code, git không biết nên merge code như thế nào, do có nhiều sự thay đổi trên cùng một file

2. Các bước giải quyết conflict

**Bước 1:** mở files chứa conflict

**Bước 2:** sửa các file trên, quyết định nội dung cần giữ (hoặc xóa)

**Bước 3:** Xóa các conflict markers (HEAD, ===)

**Bước 4:** Commit sự thay đổi

### #57. Resolve Conflict

Các bước thực hiện:

**Bước 1:** mở files chứa conflict

**Bước 2:** sửa các file trên, quyết định nội dung cần giữ (hoặc xóa)

**Bước 3:** Xóa các conflict markers (HEAD, ===)

**Bước 4:** Commit sự thay đổi

### 

### #58. Bài Tập Merge Conflict

Mục tiêu: luyện tập kỹ năng merge code & giải quyết mâu thuẫn (resolve conflict)

**Bước 1:** Clone code

<https://github.com/hoidanit-live/chapter8-05>

**Bước 2:** Chuyển nhánh code để git biết sự có mặt của các nhánh code

Chuyển qua nhánh code **feature-about**

Chuyển qua nhánh code **feature-project**

**Bước 3:** Chuyển qua nhánh master và tiến hành merge code

**git switch master**

//merge branch feature-project vào master (sẽ xuất hiện conflict)

//mục tiêu giải quyết conflict, code hợp lý là được

**git merge feature-project**

//merge branch feature-about vào master (sẽ xuất hiện conflict)

//mục tiêu giải quyết conflict, code hợp lý là được

**git merge feature-about**

### #59. Chữa Bài Tập Merge Conflict

//todo

### #60. Tổng kết các kiến thức đã học

Mục tiêu: nhìn lại các kiến thức trọng tâm của Git đã học

Git được chia làm 2 không gian lưu trữ:

1. Local Repository

**git init**

**git add**

**git commit**

2. Remote Repository

**git remote**

**git push**

3.Thao tác với branch

**git branch**

**git switch**

**git merge**

## Chapter 9: Thực Hành Quản Lý Mã Nguồn với GitLab

*Luyện tập sử dụng git với GitLab*

### #61. Tổng quan về chapter

//todo

### #62. GitLab là gì ?

Tương tự Github, Gitlab là một nền tảng (với level beginners), cho phép bạn lưu trữ và lý source code dự án miễn phí.

1.So sánh Github và Gitlab

| Tiêu chí | Github | GitLab |
| --- | --- | --- |
| Chủ sở hữu | Microsoft (tính từ 2018) | GitLab Inc. |
| CI/CD tích hợp | Sử dụng qua Github Action, hoặc tích hợp bên thứ 3, ví dụ CircleCI | Tích hợp sẵn trong Gitlab |
| Phân quyền & bảo mật | Có hỗ trợ | Có hỗ trợ và làm chi tiết hơn Github |
| Mức độ phổ biến | Phổ biến nhất, đặc biệt dành cho khách hàng cá nhân | Ít phổ biến hơn, được doanh nghiệp sử dụng nhiều |

2.Nên dùng Github hay GitLab

Tương tự như việc bạn lựa chọn, dùng các ứng dụng chat như Messenger, Telegram, Skype, Zalo…

Việc dùng nền tảng nào, mà nó giải quyết tốt cho bài toán bạn gặp phải, thì cứ thế dùng.

Một khi nền tảng đang sử dụng không giải quyết được bài toán gặp phải, đấy là lúc chuyển qua nền tảng khác.

### #63. Tạo Tài Khoản GitLab

Lưu ý xác thực tài khoản khi đăng ký

<https://forum.gitlab.com/t/preventing-crypto-mining-abuse-on-gitlab-com-saas/52911>

<https://about.gitlab.com/blog/2021/05/17/prevent-crypto-mining-abuse/>

Việc Gitlab yêu cầu xác thực thẻ credit card (VISA/MASTER), phụ thuộc vào “thuật toán” của GitLab (vị trí bạn ở, nhà mạng bạn sử dụng, IP address)

Việc xác thực này, chỉ với một mục đích, xác thực tài khoản của bạn là có thật, và không lạm dụng Gitlab vào các mục đích không chân chính (ví dụ như đào coin/crypto)

heroku

Vì đơn giản, nếu bạn dùng quá plan FREE, bạn sẽ cần trả phí (thông qua thẻ đã liên kết)

### #64. Luyện Tập Sử Dụng GitLab 01

Mục tiêu: tạo dự án tại máy tính cá nhân bạn, sau đấy đẩy code lên GitLab

**Bước 1:** Tạo dự án thực hành với tên : **chapter9-01**

**Bước 2:**  Tạo git và commit

git init

git add

git commit

**Bước 3:** Đẩy code lên gitlab

//todo

### #65. Luyện Tập Sử Dụng GitLab 02

Mục tiêu: tạo dự án trên GitLab, sau đấy clone code và code tại local

**Bước 1:** Tạo dự án thực hành tại gitlab với tên : **chapter9-02**

**Bước 2:** Clone code (lưu ý sử dụng https)

git clone

**Bước 2:**  Tạo git và commit

git init

git add

git commit

**Bước 3:** Đẩy code lên gitlab

//todo

### #66. Cảm Nhận Cá Nhân Mình về GitLab

Hỗ trợ nhiều tính năng out-of the box

Nếu có nhiều dự án, cần gom nhóm dự án (group), chọn gitlab

Ít dự án, chọn github

## **Đánh Giá (Review/Rating) Khóa Học**

1.Lời cảm ơn:

Mình rất hy vọng bạn thấy khóa học này có giá trị, nhưng dù thế nào đi nữa, hãy để lại đánh giá và chia sẻ trải nghiệm của bạn.

Hãy cho mình biết bạn nghĩ gì và viết đánh giá khi bạn có thể.

Mình cũng luôn sẵn sàng trả lời câu hỏi của bạn - hãy gửi tin nhắn trực tiếp cho mình bất cứ lúc nào.

2.Cách đánh giá (review) khóa học, tham khảo chi tiết [tại đây](https://business-support.udemy.com/hc/en-us/articles/115005537968-Leaving-and-Editing-a-Course-Review)

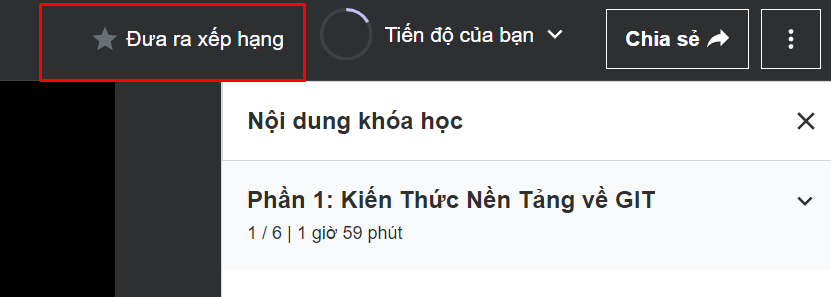
Tạo review đánh giá:

Nhấn vào “đưa ra xếp hạng”. Sau đấy rating (star) và viết đánh giá chia sẻ trải nghiệm.

Nếu màn hình của bạn không hiển thị, có 2 nguyên nhân:

1 là bạn đã đánh giá rồi, xem mục 3 bên dưới để biết cách sửa/xóa đánh giá

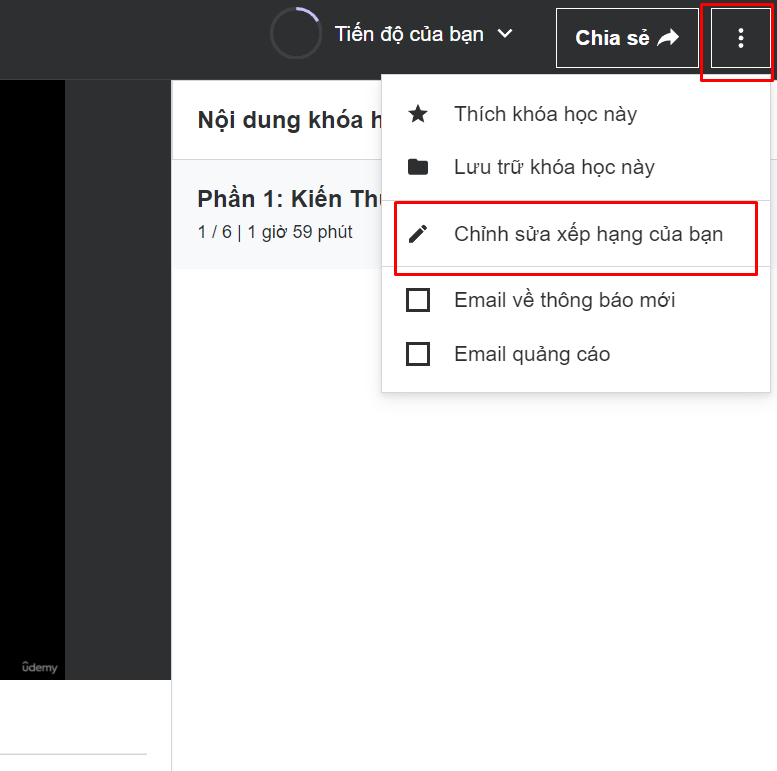
2 là thời gian bạn xem khóa học chưa đủ nên không thể đánh giá



3.Chỉnh sửa/xóa review đã đánh giá:

Bước 1: chọn vào hình 3 dấu chấm tại góc màn hình

Bước 2: chọn chỉnh sửa review



4.Mẹo để Viết Đánh Giá Tốt

Dưới đây là một số điều bạn nên lưu ý khi viết đánh giá cho một khóa học:

Hãy cho chúng tôi biết lý do: Ngoài việc để lại đánh giá sao, vui lòng chia sẻ suy nghĩ của bạn.

Ý kiến của bạn về khóa học rất có giá trị đối với những người học khác, nhưng khi chỉ có đánh giá sao, thật khó để người dùng khác hiểu vì sao bạn lại đưa ra đánh giá đó.

Cụ thể là tốt: Tính cụ thể giúp người học khác quyết định xem khóa học có phù hợp với họ không. Có điểm nào cần cải thiện không?

Khóa học có đáp ứng được kỳ vọng của bạn không?

Phần nào của khóa học khiến bạn thích nhất?

Trung thực: Đánh giá là một trong những yếu tố quan trọng nhất mà mọi người xem xét trước khi đăng ký một khóa học.

Tuy nhiên, đánh giá chỉ có giá trị khi bạn trung thực về cảm nhận của mình về khóa học.

Miễn là bạn chia sẻ cảm nhận một cách tôn trọng, ý kiến phản hồi của bạn rất có giá trị và hữu ích cho cộng đồng học tập của chúng tôi.

Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 70+ video về thực hành Git với Github/Gitlab dành cho lập trình viên.

Tất cả các kiến thức mình chia sẻ, đều được lấy từ kinh nghiệm đi làm của mình và…

các trang tài liệu về Git/Github/Gitlab

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót.

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.

<https://www.facebook.com/askITwithERIC>

Nếu bạn thấy khóa học này hữu ích, đừng quên Review đánh giá trên Udemy nhé ^^

Hẹn gặp lại các bạn ở các khóa học tiếp theo ….

Hỏi Dân IT (Eric)