



Samsung Innovation Campus

| Vạn vật kết nối – IoT

Together for Tomorrow!
Enabling People

Education for Future Generations

Chương 3.

Quản lý phiên bản mã lệnh với Git & GitHub

Vạn vật kết nối – IoT

Mô tả chương học

◆ Mục tiêu của chương học

- ✓ Hiểu về tính năng Quản lý phiên bản (Version Control) mã nguồn (mã lệnh) bằng Git và GitHub.
- ✓ Hiểu về giao diện người dùng của GitHub và các chức năng liên quan.
- ✓ Hiểu cách phân bổ phần mềm với GitHub.
- ✓ **Thử nghiệm dự án mã nguồn phần mềm** để hiểu về cách phối hợp làm việc trên Git và GitHub.

◆ Nội dung chương học

- ✓ Bài 1. Quản lý phiên bản là gì?
- ✓ Bài 2. Tổng quan & Cài đặt **Git**
- ✓ Bài 3. Tổng quan về GitHub và các thuật ngữ
- ✓ Bài 4. Cách sử dụng GitHub
- ✓ Bài 5. Phối hợp trong công việc **through** GitHub

Bài 1.

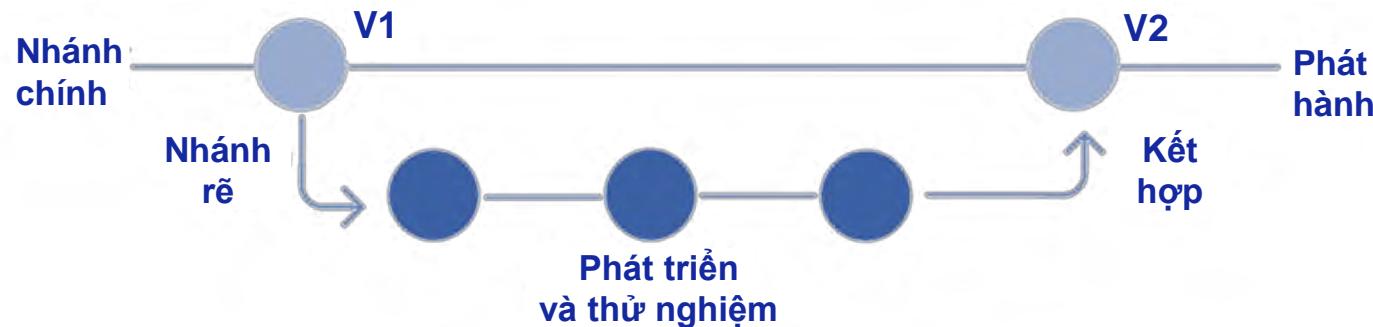
Quản lý phiên bản là gì?

- | 1.1. Giới thiệu về tính năng Quản lý phiên bản
- | 1.2. Phân loại Quản lý phiên bản
- | 1.3. Lợi ích khi sử dụng tính năng Quản lý phiên bản

Quản lý phiên bản

'Quản lý phiên bản' là một trong những kỹ năng thiết yếu trong kỹ thuật phần mềm bởi vì tất cả các phần mềm đều có mã nguồn và chức năng được phát triển thường xuyên liên tục.

- I Quản lý phiên bản có nghĩa là quản lý nhiều phiên bản khác nhau của mã nguồn. Và không chỉ là mã nguồn mà còn cả những thay đổi trong bộ tài liệu liên quan, bao gồm cả tài liệu thiết kế, cũng được phân loại bằng cách **gắn chúng với một 'phiên bản' ký hiệu thông qua chữ số hoặc chữ cái**.
 - Thông qua 'phiên bản' đã được xác định, có thể theo dõi thời gian thay đổi, các nội dung đã thay đổi và người đã thực hiện thay đổi đó.
- I Với việc dung lượng của phần mềm ngày càng lớn hơn, phức tạp hơn và đa dạng hơn, tất cả các nhà phát triển phần mềm cần hiểu về Quản lý phiên bản để sử dụng nó một cách hiệu quả nhất có thể.



I Hệ thống Quản lý phiên bản (Version Control system - VCS)

- Các hệ thống Quản lý phiên bản là những công cụ phát triển phần mềm được thiết kế để giúp các nhà phát triển có thể theo dõi bất kỳ thay đổi nào với mã nguồn của bất kỳ ứng dụng cụ thể nào.

I Sự cần thiết của Quản lý phiên bản

- Giúp phục hồi lại công việc khi có sự cố xảy ra.
- Có thể quay ngược lại phần công việc đã thực hiện ở một thời điểm nào đó trong một dự án.
- Tự động đồng bộ các thay đổi được thực hiện bởi mỗi cá nhân khi có nhiều người cùng tham gia vào một dự án.
- Theo dõi các thay đổi mã nguồn và biết được ai đã thực hiện các thay đổi đó.
- Giúp việc điều chỉnh quy mô lớn trở nên an toàn và đảm bảo hơn.
- Được ứng dụng trong nhiều dự án mã nguồn mở **theo nhiều hình thức khác nhau**.
- Giúp theo dõi lý do tại sao một đoạn mã lệnh cụ thể **được xây dựng tới thời điểm hiện tại**.

I Các hệ thống Quản lý phiên bản phổ biến nhất

- Phổ biến nhất hiện nay là Git, được hàng triệu nhà phát triển và lập trình viên trên toàn thế giới sử dụng.

I Một số hệ thống khác được sử dụng nhiều:

- Perforce Helix Core
- Apache Subversion
- AWS CodeCommit



HelixCore



SUBVERSION®



AWS CodeCommit

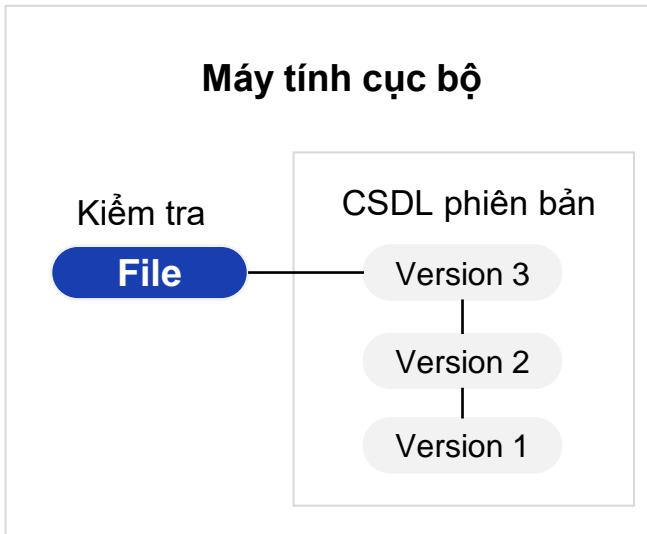
Bài 1.

Quản lý phiên bản là gì?

- | 1.1. Giới thiệu về tính năng Quản lý phiên bản
- | 1.2. Phân loại Quản lý phiên bản
- | 1.3. Lợi ích khi sử dụng tính năng Quản lý phiên bản

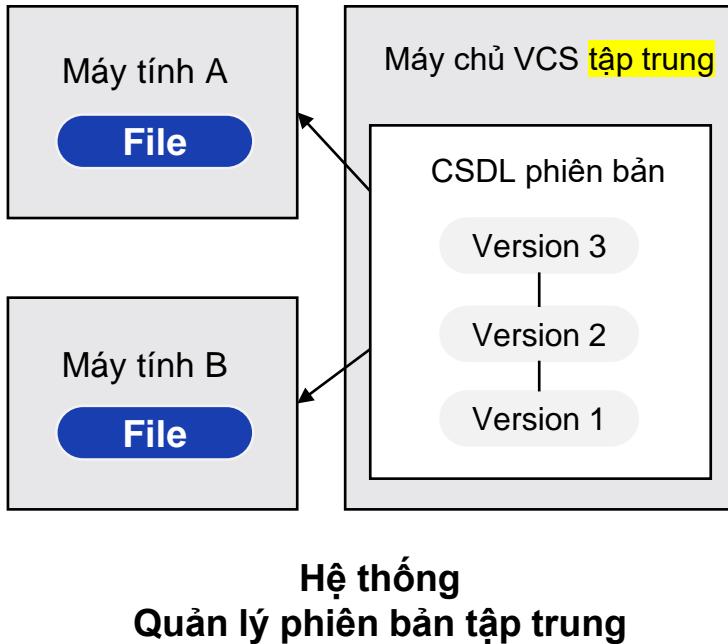
3 nhóm hệ thống Quản lý phiên bản

| Các hệ thống Quản lý phiên bản cục bộ



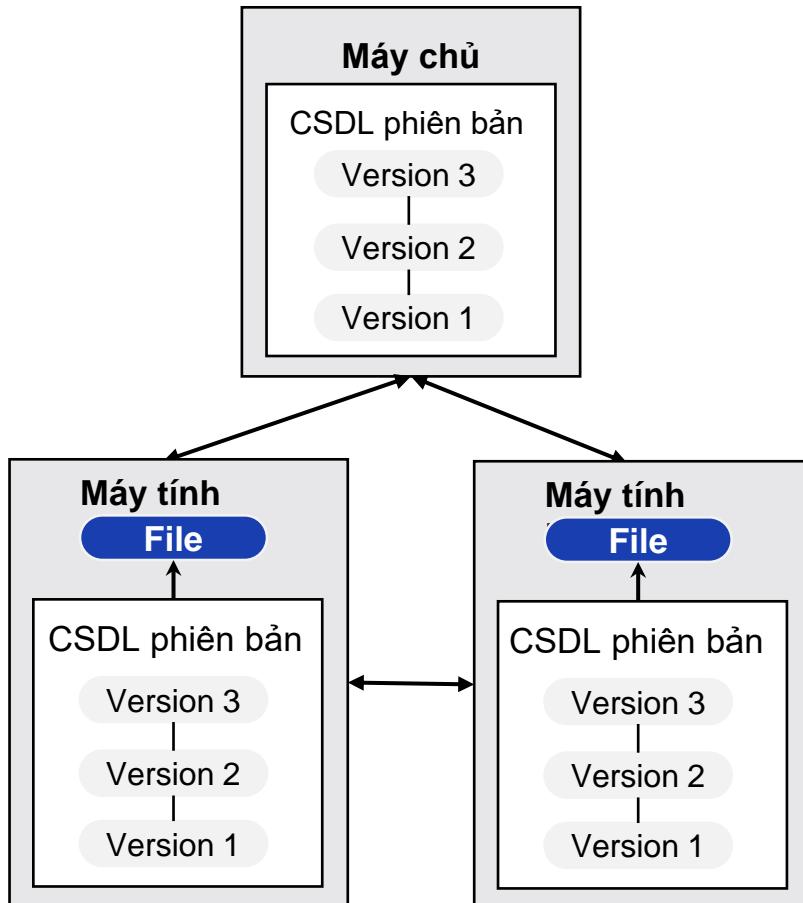
- ▶ LVCS (LVCS – Local VCS) sử dụng một cơ sở dữ liệu đơn giản để quản lý các thay đổi trên file.
- ▶ Trong các hệ thống Quản lý phiên bản (VCS) được sử dụng rộng rãi, hệ thống quản lý hiệu chỉnh (RCS – Revision Control System) đang được nhiều công ty lựa chọn sử dụng. RCS về cơ bản quản lý bộ các bản vá (**những phần được thay đổi trong file**).
- ▶ Các bản vá được lưu dưới dạng file có định dạng đặc biệt. Có thể khôi phục tất cả các file về phiên bản ở một thời điểm cụ thể bằng cách áp dụng một loạt các bản vá.

I Các hệ thống Quản lý phiên bản tập trung



- ▶ Các hệ thống Quản lý phiên bản tập trung (CVCS – Centralized Version Control Systems) là một loại hệ thống Quản lý phiên bản với một máy chủ duy nhất chứa tất cả các phiên bản của các file mã nguồn, số lượng nhân sự tham gia và các file sao lưu cho dự án chính.
- ▶ CVCS giúp giao tiếp và quản lý nhân sự dễ dàng hơn dựa trên các nhiệm vụ cụ thể do nhân sự cùng nhóm thực hiện dự án sẽ biết những gì người khác đang làm.
- ▶ Nhược điểm chính của CVCS là tất cả các file mã nguồn và file sao lưu của **dự án** được lưu trữ trên một máy chủ tập trung. Nếu có sự cố xảy ra với máy chủ, toàn bộ các file sẽ có nguy cơ biến mất.
- ▶ Các công cụ Subversion và Perforce là những ví dụ về CVCS.

I Các hệ thống Quản lý phiên bản phân tán



- Hệ thống Quản lý phiên bản phân tán (DVCS - Distributed Version Control Systems), chẳng hạn như Git, Mercurial, hoặc Bazaar, là các hệ thống Quản lý phiên bản với một hoặc nhiều máy chủ chính chứa các file mã nguồn.
- Trong DVCS, các file mã nguồn được ánh xạ trên máy tính của các nhân sự dự án khác với CVCS. Điều này có nghĩa là mỗi mã nguồn có phiên bản riêng, bao gồm toàn bộ lịch sử làm việc cục bộ.
- Ngay cả khi máy chủ dừng hoạt động, bộ nhớ lưu trữ của nhân sự dự án có thể được sao chép trở lại máy chủ và lịch sử của các file mã nguồn có thể được khôi phục. Thực hiện việc này rất đơn giản vì mọi bản sao của file mã nguồn gốc thực chất là bản sao lưu đầy đủ của tất cả dữ liệu từ dự án. DVCS cho phép nhân sự từ các địa điểm địa lý khác nhau cộng tác đồng thời trong cùng một dự án. Đây là một lợi thế không thể có trong một hệ thống tập trung.

Bài 1.

Quản lý phiên bản là gì?

- | 1.1. Giới thiệu về tính năng Quản lý phiên bản
- | 1.2. Phân loại Quản lý phiên bản
- | 1.3. Lợi ích khi sử dụng tính năng Quản lý phiên bản

Lợi ích khi sử dụng hệ thống Quản lý phiên bản (1/2)

| Tạo các bản sao lưu

- Lợi ích quan trọng nhất của việc sử dụng hệ thống Quản lý phiên bản là bắt cứ khi nào nhân sự dự án tiến hành sao chép dữ liệu được lưu trữ, hệ thống sẽ tạo ra một bản sao lưu cho phiên bản hiện tại của kho lưu trữ đó.

| Kiểm tra và Thử nghiệm

- Khi cùng thực hiện một dự án phần mềm, các nhóm lập trình viên thường có các bản sao khác nhau của dự án chính để phát triển và thử nghiệm tính năng mới. Trước khi thêm các tính năng mới này vào dự án chính của bạn, hệ thống sẽ đảm bảo rằng sản phẩm **phần mềm** đang hoạt động bình thường. Việc này giúp tiết kiệm thời gian vì bạn có thể phát triển các tính năng khác nhau của chương trình **phần mềm** đồng thời cùng lúc.

Lợi ích khi sử dụng hệ thống Quản lý phiên bản (2/2)

| Lưu lịch sử và theo dõi thay đổi

- Lưu trữ bản ghi chép các thay đổi trong một file mã nguồn cụ thể sẽ giúp bạn và những nhân sự mới tham gia biết được có thay đổi nào với phần mã nguồn đó.
- Ngoài ra, nếu việc bổ sung một số tính năng nhất định theo thời gian gây khó khăn cho việc đánh giá hoặc mở rộng quy mô dự án, khi này tính năng Quản lý phiên bản cho phép nhà phát triển theo dõi một số tính năng nhất định và thay đổi hoặc loại bỏ chúng mà không ảnh hưởng đến chức năng tổng thể của dự án.

| Cộng tác và đóng góp

- Một trong những lợi ích chính khi sử dụng hệ thống Quản lý phiên bản, đặc biệt là DVCS, là các bạn có thể đóng góp công sức cho các dự án yêu thích, ngay cả khi các bạn ở các quốc gia khác nhau.

Bài 2.

Git : Tổng quan & Cài đặt

- | 2.1. Tổng quan về Git
- | 2.2. Cài đặt Git & GitHub

Git là gì?

- Khi phát triển phần mềm, rất khó để quản lý phiên bản mã nguồn mới nhất giữa các thành viên trong nhóm thực hiện. Vì lý do này, công cụ quản lý cấu hình (configuration management tool) đã được tạo ra để giải quyết các vấn đề của Quản lý phiên bản thường xuyên xảy ra trong một dự án. Một trong những công cụ quản lý cấu hình này là Git, được phát triển bởi Linus Torvalds vào năm 2005 như một hệ thống quản lý phiên bản cho dự án phát triển xử lý lõi (kernel) của HĐT Linux.
- | Git tiếp cận dễ dàng vì nó ít phức tạp hơn so với công cụ cấu hình SVN (Subversion) hiện có. Đó là một hệ thống quản lý phiên bản phân tán, trong đó tất cả các nhân sự đều có bản gốc của mã nguồn và tiến độ thực hiện công việc rất nhanh.
- | Git có thể thực hiện hầu hết các công việc trong mạng cục bộ. Nó là một công cụ quản lý cấu hình được sử dụng rộng rãi do ưu điểm về tốc độ khi thao tác với nhánh/thẻ/master và các bản chỉnh sửa khác.



Đặc điểm của Git

- | Là một hệ thống quản lý phiên bản phân tán cho phép người dùng duy trì và đồng bộ hóa kho dữ liệu Git từ xa.
- | Git thực thi hầu hết tất cả các lệnh một cách cục bộ, đồng thời việc tạo / chuyển đổi / xử lý nhánh rất nhanh.
- | Git quản lý bản sao lưu toàn bộ hệ thống khác với các hệ thống Quản lý phiên bản khác, chỉ lưu trữ các thay đổi cho từng file riêng lẻ.
- | Chức năng chính
 - Máy chủ quản lý phiên bản mã nguồn (Source), tài liệu và các thuộc tính của máy khách
 - Các thao tác cục bộ như add, commit, reset, branch, checkout, merge, rebase,
 - Các thao tác từ xa như push, pull, fetch,

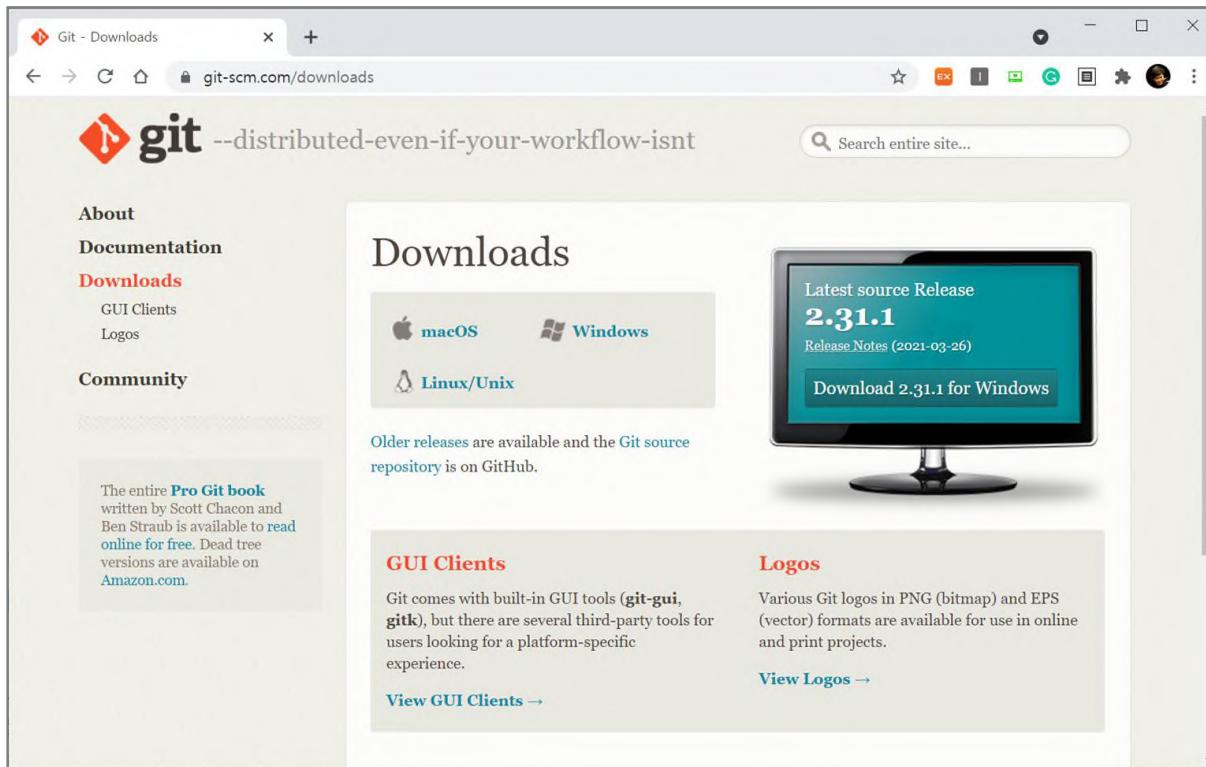
Bài 2.

Git : Tổng quan & Cài đặt

- | 2.1. Git Overview
- | 2.2. Git : Tổng quan & Cài đặt

Cài đặt (Windows 10)

Hướng dẫn cài đặt Git

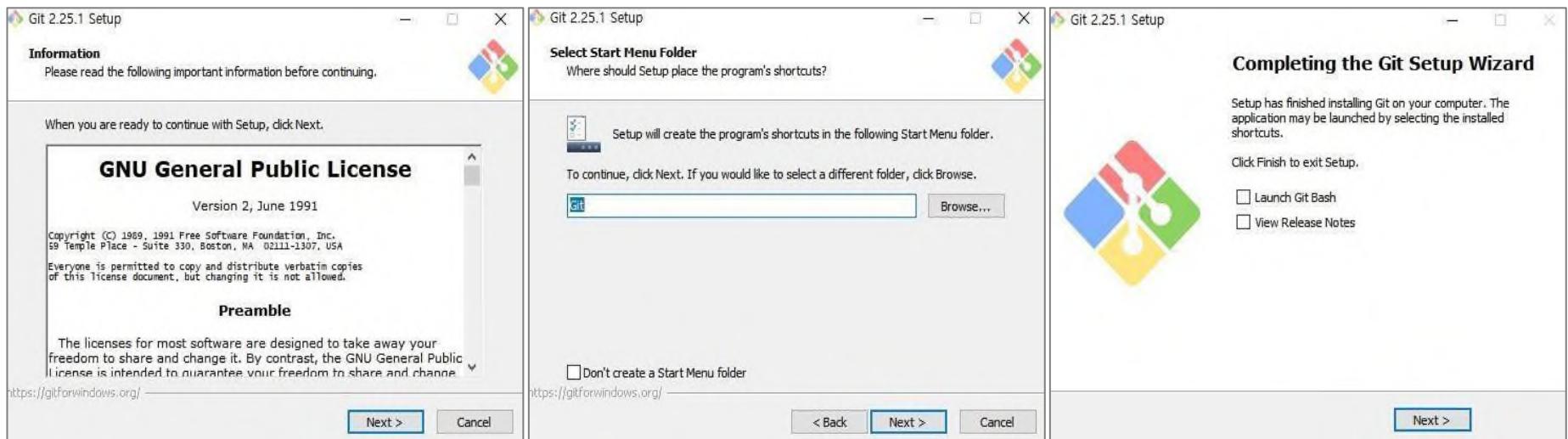


- Truy cập tới <https://git-scm.com/downloads> để tải về bản cài và tiến hành cài đặt

Cài đặt (Windows 10)

Hướng dẫn cài đặt Git

- Truy cập tới <https://git-scm.com/downloads> để tải về và tiến hành cài đặt công cụ Git.



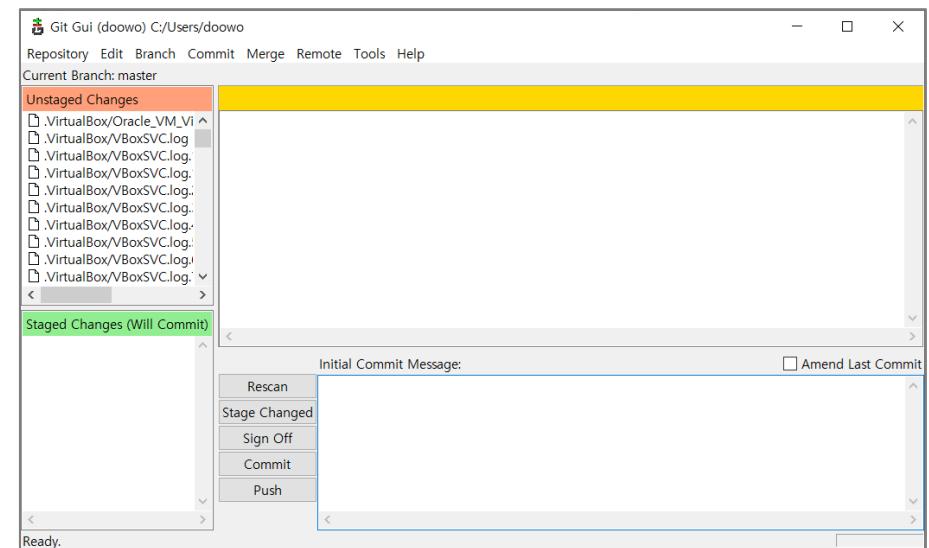
Cài đặt (Windows 10)

| 2 cách sử dụng Git

- ▶ CLI (Command Line Interface) – Giao diện dòng lệnh
- ▶ GUI (Graphical User Interface) – “Giao diện cửa sổ”



```
MINGW64:/c/Users/doowo
doowo@SIC MINGW64 ~ (master)
$ git --version
git version 2.31.1.windows.1
doowo@SIC MINGW64 ~ (master)
$
```



Cài đặt trên Windows 10

| Khởi tạo Git

- Thiết lập ban đầu - thiết lập thông tin người dùng (**Chọn và chạy công cụ Git Bash**)



MINGW64:/c/Users/doowo
doowo@SIC MINGW64 ~ (master)
\$ |

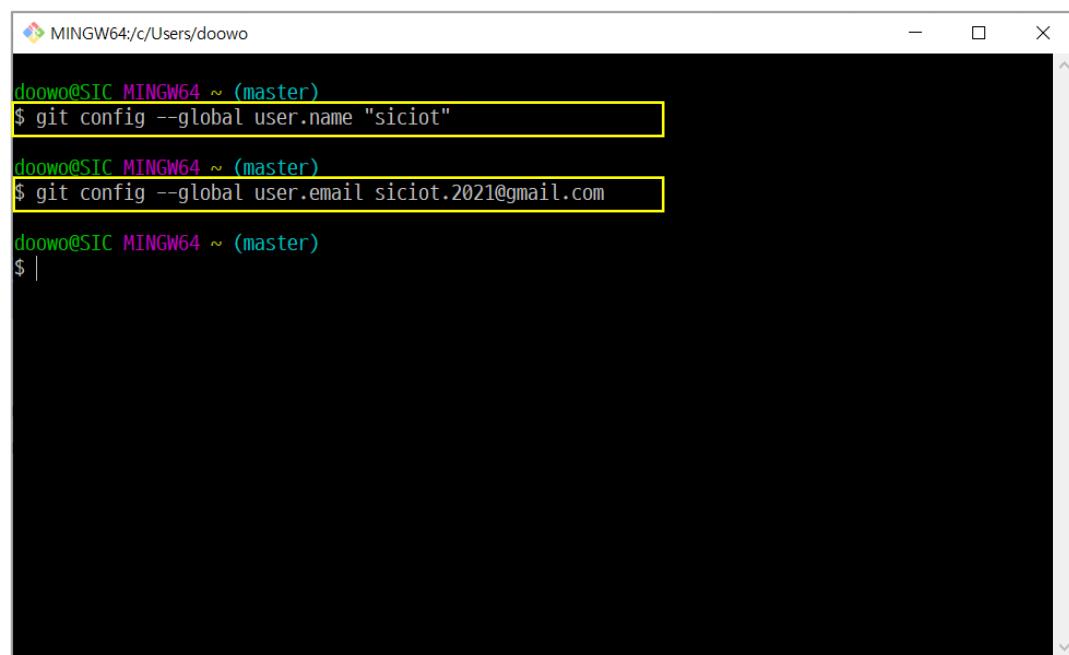
Master
Nhánh chính (Nhánh chủ)
(Môi trường làm việc chứa phiên bản cuối cùng của phần mềm/dự án)

Cài đặt trên Windows 10

| Khởi tạo Git

- Cài đặt thông tin người dùng

```
$ git config --global user.name "siciot"  
$ git config --global user.email siciot.2021@gmail.com
```



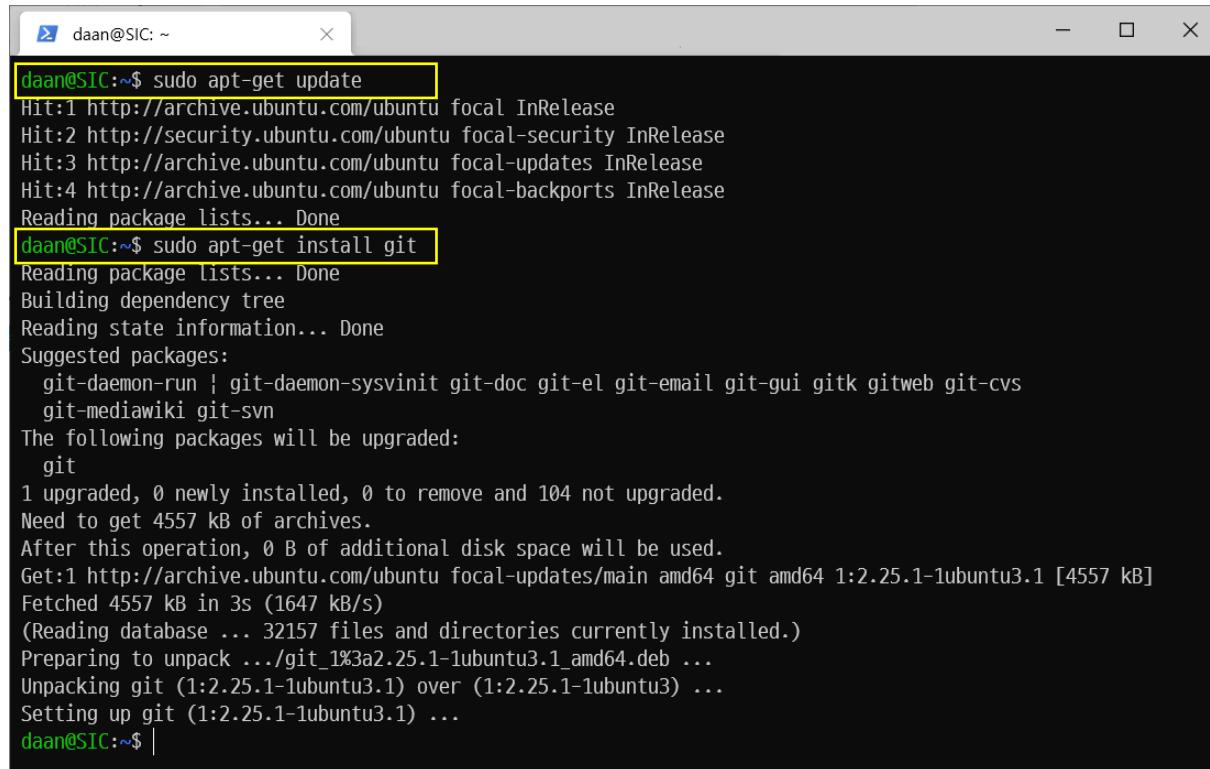
The screenshot shows a terminal window titled 'MINGW64:/c/Users/doowo'. It displays the following command-line session:

```
doowo@SIC MINGW64 ~ (master)  
$ git config --global user.name "siciot"  
doowo@SIC MINGW64 ~ (master)  
$ git config --global user.email siciot.2021@gmail.com  
doowo@SIC MINGW64 ~ (master)  
$ |
```

The first two lines of the session are highlighted with yellow boxes.

Cài đặt trong ubuntu®/ Raspberry Pi OS

Hướng dẫn cài đặt Git trên hệ điều hành ubuntu (hoặc Raspberry Pi OS)



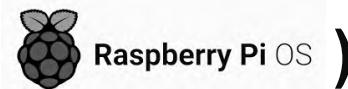
The screenshot shows a terminal window titled "daan@SIC: ~". It displays the command \$ sudo apt-get update followed by its output, which includes hits from various Ubuntu repositories. Below this, the command \$ sudo apt-get install git is shown, followed by its output, which details the upgrade of the git package from version 1.2.25.1-1ubuntu3.1 to 1.2.25.1-1ubuntu3.1, along with file unpacking and setting up.

```
daan@SIC:~$ sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
daan@SIC:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn
The following packages will be upgraded:
  git
  1 upgraded, 0 newly installed, 0 to remove and 104 not upgraded.
  Need to get 4557 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:2.25.1-1ubuntu3.1 [4557 kB]
Fetched 4557 kB in 3s (1647 kB/s)
(Reading database ... 32157 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.25.1-1ubuntu3.1_amd64.deb ...
Unpacking git (1:2.25.1-1ubuntu3.1) over (1:2.25.1-1ubuntu3) ...
Setting up git (1:2.25.1-1ubuntu3.1) ...
daan@SIC:~$ |
```

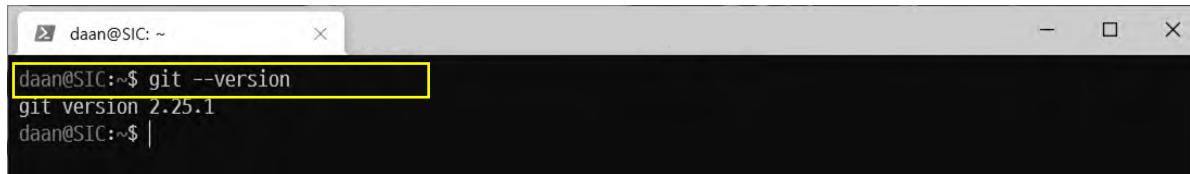
- Mở (chạy) công cụ Terminal
- Cài đặt bằng lệnh trong Terminal

```
$ sudo apt-get update
$ sudo apt-get install git
```

Cài đặt (Ubuntu)



Hướng dẫn cài đặt Git



A screenshot of a terminal window titled "daan@SIC: ~". The window shows the command "git --version" being run, which outputs "git version 2.25.1". The terminal has a light gray header bar and a dark gray body.

```
daan@SIC:~$ git --version
git version 2.25.1
daan@SIC:~$ |
```

▶ Kiểm tra phiên bản Git

```
$ git --version
```

Cài đặt (Ubuntu Raspberry Pi OS)

Hướng dẫn cài đặt Git

- Cài đặt ban đầu - cài đặt thông tin người dùng
- user.name và user.email là tên và địa chỉ email được hiển thị trên git.

```
$ git config --global user.name "siciot"  
$ git config --global user.email siciot.2021@gmail.com
```

```
daan@SIC: ~  
daan@SIC:~$ git config --global user.name "siciot"  
daan@SIC:~$ git config --global user.email siciot.2021@gmail.com  
daan@SIC:~$ |
```

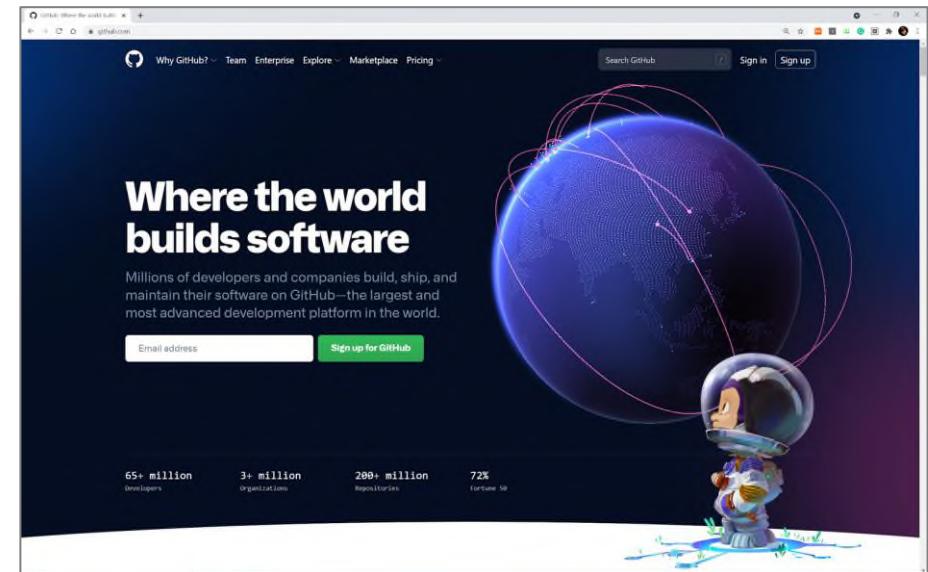
Bài 3.

Tổng quan về GitHub và các thuật ngữ

- | 3.1. Tổng quan về GitHub
- | 3.2. Thuật ngữ GitHub

GitHub là gì?

- GitHub là một kho lưu trữ mã nguồn mở trực tuyến, cung cấp các chức năng khác nhau để lưu trữ và quản lý phiên bản mã nguồn nhằm phát triển phần mềm mã nguồn mở, cũng như cộng tác và trao đổi trực tuyến trong cộng đồng các nhà phát triển.
- Người dùng có thể duyệt kho lưu trữ ở Github và tải về các đoạn mã nguồn. Ngoài ra, nó còn cung cấp các chức năng như thảo luận, quản lý kho lưu trữ, cộng tác với các kho lưu trữ khác và xem xét các thay đổi đã thực hiện trên mã nguồn.



<https://github.com/>



GitHub là máy chủ lớn nhất phục vụ lưu trữ Git

- Nhiều kho lưu trữ Git được đặt trên GitHub. Nhiều dự án mã nguồn mở sử dụng GitHub để lưu trữ Git, theo dõi sự cố, đánh giá kết quả và các tác vụ khác.
- Mặc dù GitHub không phải là một phần trực tiếp của dự án mã nguồn mở Git, nhưng nếu muốn sử dụng Git một cách chuyên nghiệp, bạn vẫn sẽ cần phải làm quen với GitHub.



git

Quản lý phiên bản



GitHub

Collaboration & Backup

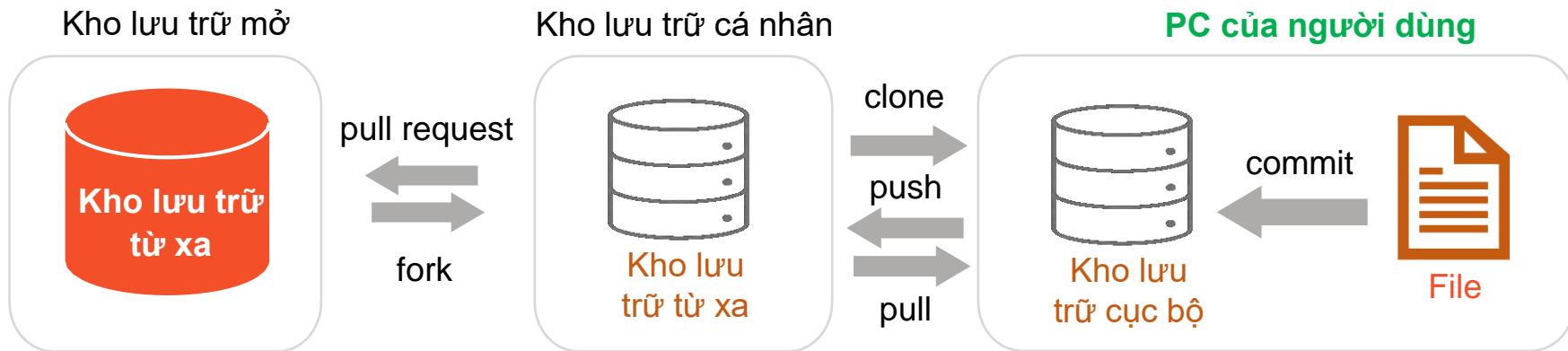
Bài 3.

Tổng quan về GitHub và các thuật ngữ

- | 3.1. Tổng quan về GitHub
- | 3.2. Thuật ngữ GitHub

GitHub là gì?

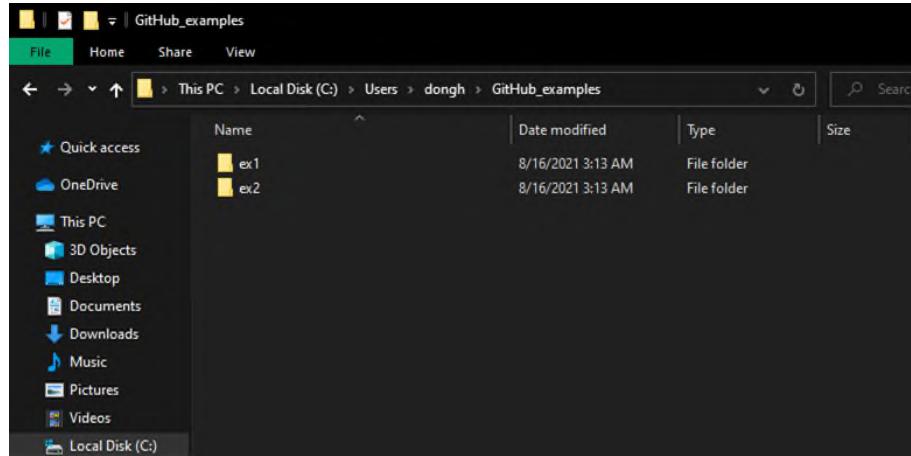
| Tìm hiểu thuật ngữ GitHub



3.2. Thuật ngữ GitHub

Bài 03

Tìm hiểu thuật ngữ GitHub



A screenshot of the GitHub web interface. The top navigation bar includes 'Overview', 'Repositories 30', 'Projects', and 'Packages'. A search bar says 'Find a repository...'. Below, there are two repository cards: 'ex1' (For doing some examples, updated 10 seconds ago) and 'News_Topic_Classification' (updated 15 days ago). Each card has a 'Star' button.

- ▶ **Kho lưu trữ (Repository):** Nơi (thư mục) lưu trữ file hoặc thư mục con
- ▶ **Kho lưu trữ cục bộ (Local Repository):** Kho lưu trữ trên PC cục bộ
- ▶ **Kho lưu trữ từ xa (Remote Repository):** Kho lưu trữ trên Internet hoặc trực tuyến như GitHub

Tóm tắt các câu lệnh

- git init : Lệnh này tạo ra một kho lưu trữ Git mới trong thư mục hiện tại. (Tạo một thư mục ẩn tên là .git bên trong thư mục hiện tại, chuyển đổi thư mục thành một khi Git cục bộ) Nó có thể được sử dụng để chuyển đổi một dự án hiện có, không được đánh dấu phiên bản thành kho lưu trữ Git hoặc khởi tạo một kho lưu trữ trống mới. Hầu hết các lệnh Git khác không có sẵn bên ngoài kho lưu trữ được khởi tạo, vì vậy đây thường là lệnh đầu tiên bạn sẽ chạy trong một dự án mới.

```
$ git init
```

- git config : Lệnh này kiểm tra và thay đổi cài đặt. Bạn có thể đặt tên người dùng hoặc địa chỉ e-mail Khi nó được sử dụng để thiết lập môi trường git, bạn có thể kiểm tra cài đặt bằng cách sử dụng lệnh git config --list.

```
$ git config
```

- Ex** \$ git config --global user.email johndoe@example.com
- git status : Lệnh này kiểm tra trạng thái kho lưu trữ. Bạn có thể xem nội dung nào trong kho lưu trữ, những thay đổi nào cần được thực hiện và nhánh nào của kho lưu trữ bạn hiện đang làm việc.

```
$ git status
```

```
siciot2021@siciot2021-VirtualBox:~/example1$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

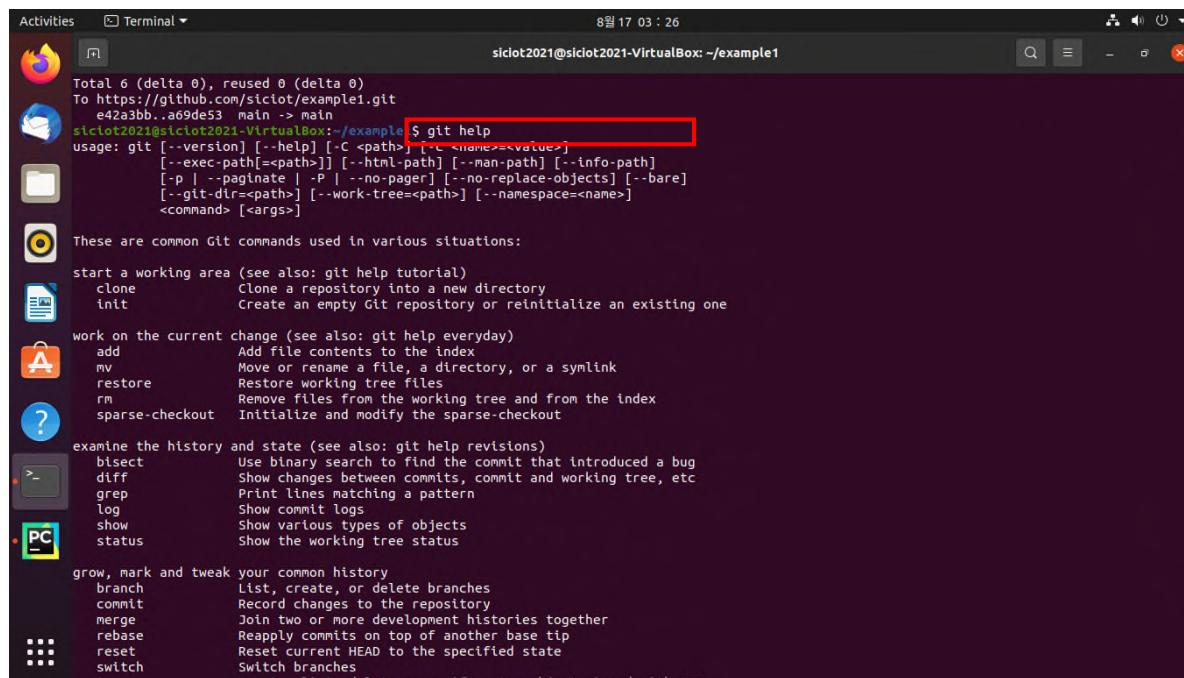
3.2. Thuật ngữ GitHub

Tóm tắt các câu lệnh

- git help : Lệnh này kiểm tra các lệnh Git khi bạn quên một lệnh để sử dụng
- git help init : Lệnh này cũng kiểm tra các lệnh Git cụ thể

```
$ git help
```

```
$ git help init
```



The screenshot shows a terminal window titled "Activities Terminal" with the command "sictot2021@sictot2021-VirtualBox: ~/example1". The terminal displays the help output for "git" and "git init".

```

Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/sictot/example1.git
  e42aabb..a69de53  main -> main
usage: git [<version>] [<--help> [<-->]] [<--name>=<value>]
        [<--exec-path[=<path>]] [<--html-path>] [<--man-path>] [<--info-path>
        [<p>] [<--paginate> | <-P>] [<--no-pager>] [<--no-replace-objects>] [<--bare>
        [<git-dtr=<path>] [<--work-tree=<path>] [<--namespace=<name>]
        <>command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv        Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm        Remove files from the working tree and from the index
  sparse-checkout Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  diff     Show changes between commits, commit and working tree, etc
  grep     Print lines matching a pattern
  log      Show commit logs
  show    Show various types of objects
  status   Show the working tree status

grow, mark and tweak your common history
  branch   List, create, or delete branches
  commit   Record changes to the repository
  merge   Join two or more development histories together
  rebase   Reapply commits on top of another base tip
  reset   Reset current HEAD to the specified state
  switch  Switch branches
  tlog

```

Tóm tắt các câu lệnh

- git add : Lệnh này không thêm file mới vào kho lưu trữ. Thay vào đó, lệnh này thông báo cho git có một file mới, hoặc được thay đổi vào kho lưu trữ ảnh chụp (snapshot) trong Git và chuyển các file này vào khu vực chuẩn bị (khu vực theo dõi) (staging area) trước khi commit.

```
$ git add
```

- git commit : Lệnh này thực hiện lưu các "ảnh chụp" (snapshot) của kho lưu trữ Git sau khi thực hiện bất kỳ thay đổi nào. Thông thường, nó được viết dưới dạng "git commit -m "Message"". Trong đó, -m là một tùy chọn, hiển thị một thông báo ngay sau khi lệnh được thực hiện.

```
$ git commit -m "Message"
```

- git branch : Lệnh này tạo một nhánh mới và tạo một dòng thời gian xác nhận các thay đổi và bổ sung file trong dự án. Sử dụng biểu mẫu tương tự như git branch {new branch name}. Trong tình huống phát triển dự án theo nhóm, từ kho lưu trữ công khai có thể tạo một nhánh mới để công việc của một thành viên không bị ảnh hưởng bởi các thành viên khác trong nhóm.

```
$ git branch {new branch name}
```

- git checkout : Lệnh này kiểm tra một kho lưu trữ hiện không được định vị. Nó là một lệnh điều hướng đưa bạn đến kho lưu trữ bạn muốn kiểm tra. Nếu bạn muốn xem nhánh chính, bạn có thể sử dụng git checkout master và git checkout {destination branch} để chuyển đến một nhánh khác. Các bạn cũng có thể sử dụng git log để xem commit id và quay lại commit trước đó.

```
$ git checkout
```

Tóm tắt các câu lệnh

- **git merge** : Lệnh này kết thúc công việc trên một nhánh và hợp nhất nội dung của nhánh vào nhánh chính, để tất cả các thành viên trong nhóm có thể nhìn thấy. Ví dụ: Lệnh Git Merge Cats thêm tất cả các thay đổi được thực hiện trong nhánh "Cats" vào nhánh chính (Master).

```
$ git merge
```

- **git push** : Lệnh này đẩy (upload) các thay đổi trong kho lưu trữ cục bộ lên Git server (GitHub).

```
$ git push
```

- **git pull** : Lệnh này tải xuống các thay đổi từ GitHub khi bạn đang làm việc trên máy cục bộ và muốn có phiên bản mới nhất của kho lưu trữ trên Git server (GitHub).

```
$ git pull
```

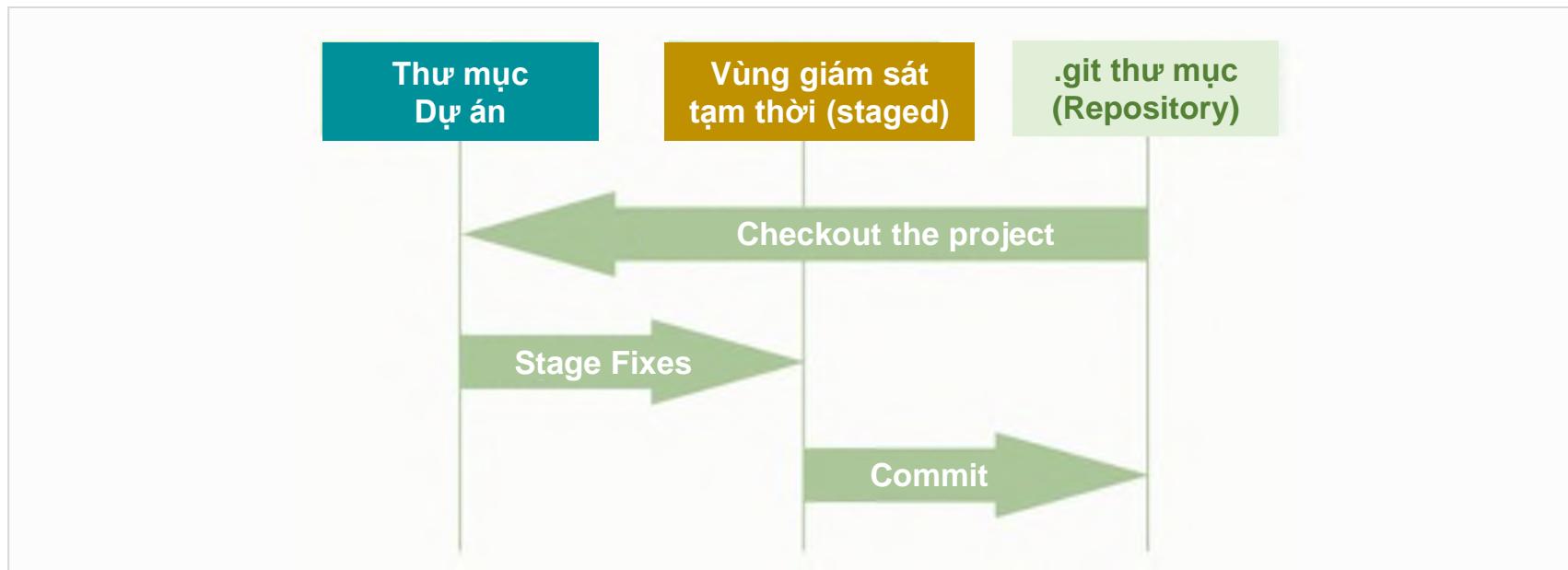
- **git clone** : Lệnh này tạo một bản sao (copy) của một kho lưu trữ trên GitHub để bắt đầu làm việc với dự án. Nó thực hiện chức năng kết hợp của các lệnh: `git init + git remote add origin { URL } + git pull origin main`.

```
$ git clone
```

```
$ git init + git remote add origin { URL } + git pull origin main
```

3 trạng thái của File

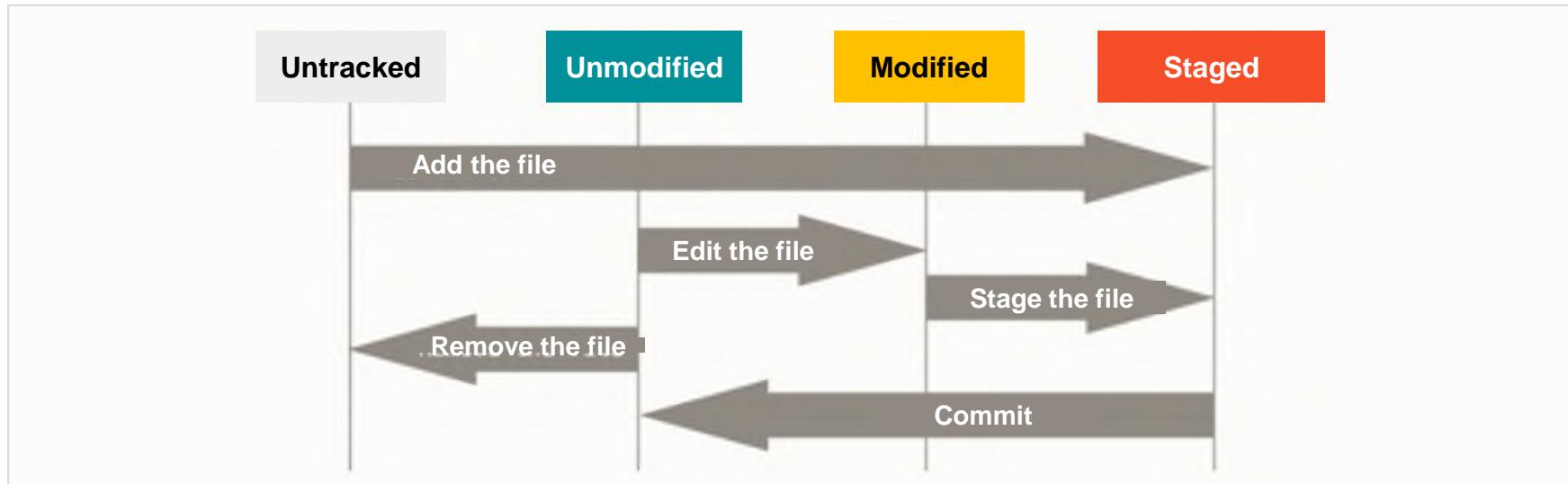
- Git quản lý các file ở ba trạng thái: Committed, Modified, and Staged.
 - Committed : Điều đó có nghĩa là dữ liệu được lưu trữ an toàn trong cơ sở dữ liệu cục bộ.
 - Modified : Nó có nghĩa là file đã được thay đổi nhưng chưa được xác nhận với cơ sở dữ liệu cục bộ.
 - Staged : Điều đó có nghĩa là bạn đã đánh dấu một file đã sửa đổi trong phiên bản hiện tại của nó để chuyển sang ảnh chụp nhanh xác nhận tiếp theo của bạn.



<https://paransilverlight.tistory.com/242>

Vòng đời của File

- Tất cả các file được chia thành 2 loại Theo dõi và Không theo dõi. Các file được theo dõi đã được gắn với ảnh chụp (snapshot). Các file được theo dõi có thể là file chưa sửa đổi, được sửa đổi hoặc được đánh dấu (staged) (đã chuẩn bị được committed). Nói một cách đơn giản, các file được theo dõi là các file mà Git quản lý.
- Tất cả các file không được giám sát đều là các file không được theo dõi. Các file không được theo dõi không gắn với ảnh chụp cũng như không nằm trong vùng theo dõi tạm thời. Khi thực hiện sao chép (clone) một kho lưu trữ lần đầu tiên, nó sẽ ở trạng thái Chưa sửa đổi được theo dõi vì không có gì được sửa đổi ngay sau khi checkout (chuyển đổi hoặc khôi phục một nhánh).
- Nếu một file được sửa đổi sau lần commit sau cùng trong khi không chỉnh sửa nào, Git sẽ nhận diện file đó đã bị thay đổi. Để có thể commit, file đã chỉnh sửa cần được đưa vào trạng thái staged. Vòng đời này – từ việc commit các file ở trạng thái staged – sẽ được lặp đi lặp lại.



Nhánh là gì?

- Hầu hết các hệ thống quản lý phiên bản đều hỗ trợ chức năng phân nhánh.
 - Trong quá trình phát triển phần mềm, cần phải sao chép nhiều mã nguồn. Bạn có thể tạo toàn bộ mã nguồn cùng nhau, nhưng thường xảy ra trường hợp bạn có thể chia toàn bộ mã nguồn thành từng phần để phát triển riêng biệt.
 - Sau khi sao chép mã nguồn, bạn có thể phát triển nó một cách độc lập với mã nguồn gốc. Nói cách khác, chức năng cho phép bạn phát triển độc lập là Nhánh (Branch).
 - Nhánh theo nghĩa đen giống như một nhánh cây; nó cho phép bạn làm việc riêng lẻ độc lập với toàn bộ dự án. Bạn có thể tạo một không gian làm việc duy nhất để bắt đầu làm việc. Nếu kết quả tốt, bạn có thể áp dụng chúng cho nhánh chính để có kết quả ổn định. Chức năng này cải thiện hiệu quả và tính ổn định của dự án.

Nhánh là gì?

- Nhánh của Git rất nhẹ và nhanh chóng để chuyển đổi, vì vậy có thể tạo một nhánh hoặc di chuyển giữa nhiều nhánh trong thời gian ngắn.
 - Không giống như các hệ thống quản lý phiên bản khác, Git khuyên bạn nên tạo các nhánh và hợp nhất chúng sau cùng.
 - Tạo một branch là một công việc rất đơn giản như cách tạo một tập tin 41-byte duy nhất, nhưng nó hỗ trợ Quản lý phiên bản rất hiệu quả.
 - **Toàn bộ quá trình sao chép mất hàng chục giây đến hàng chục phút tùy thuộc vào kích thước của dự án, nhưng Git thì rất nhanh. Ngoài ra, vì thông tin lần commit trước đó được lưu lại mỗi khi bạn commit, bạn có thể dễ dàng xác định vị trí để bắt đầu quá trình hợp nhất (Merge Base).**
 - Tính năng này là một tính năng mạnh mẽ để các nhà phát triển thường tạo và sử dụng các nhánh.

| Nếu chúng ta gặp một số vấn đề phát sinh,

- ▶ Trong quá trình phát triển thực tế, chúng ta có thể sử dụng Branch theo cách thức sau:
 - Viết mã lệnh và làm việc trên dự án.
 - Tạo một nhánh mới (có tên là issue_1) để xử lý các vấn đề phát sinh.
 - Bắt đầu làm việc trên nhánh mới vừa tạo.
- ▶ Trong khi giải quyết vấn đề này, nếu bạn thấy một vấn đề quan trọng khác và bạn cần khắc phục ngay lập tức (Hotfix) :
 - Chuyển đến nhánh Production (trong đó không xảy ra vấn đề gì) trước khi xử lý các vấn đề mới.
 - Tạo một nhánh hotfix mới.
 - Sau khi kiểm tra Hotfix đã sửa, hợp nhất nhánh Hotfix vào nhánh Production.
 - Đến nhánh (issue_1) bạn đang thực hiện và tiến hành các nhiệm vụ còn lại.

Bài 4.

Cách sử dụng GitHub

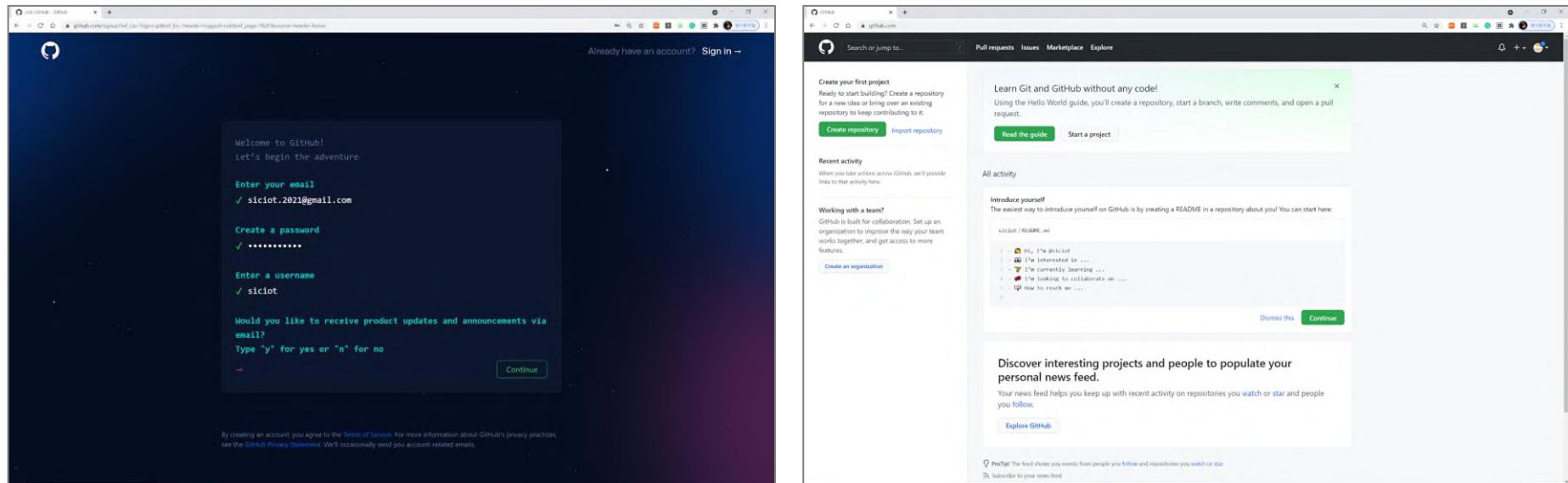
- | 4.1. Bài luyện tập GitHub đơn giản
- | 4.2. Cách sử dụng GitHub khi làm việc theo cá nhân
- | 4.3. Cách sử dụng GitHub thông qua môi trường GUI

4.1. Bài luyện tập GitHub đơn giản

Cài đặt và Cấu hình tài khoản

Tạo một tài khoản

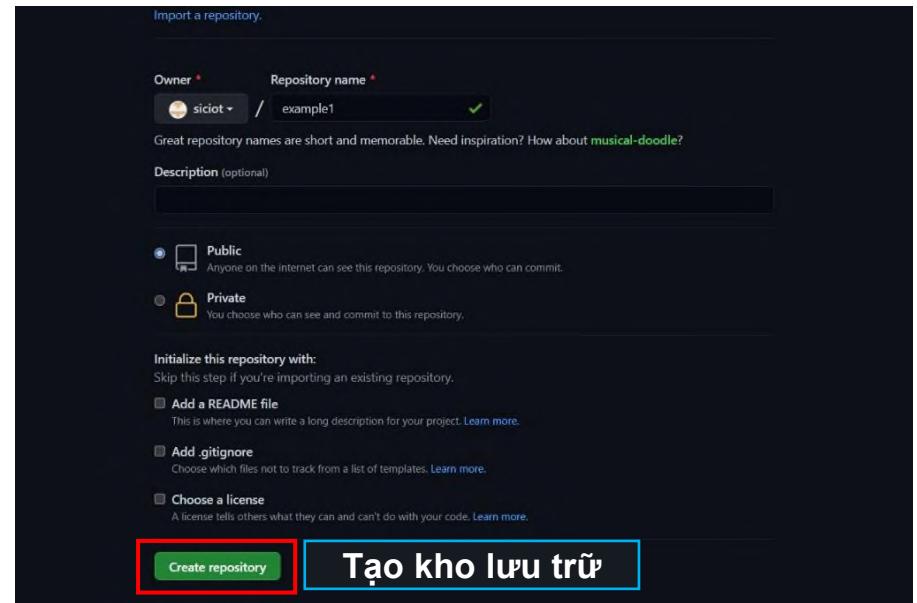
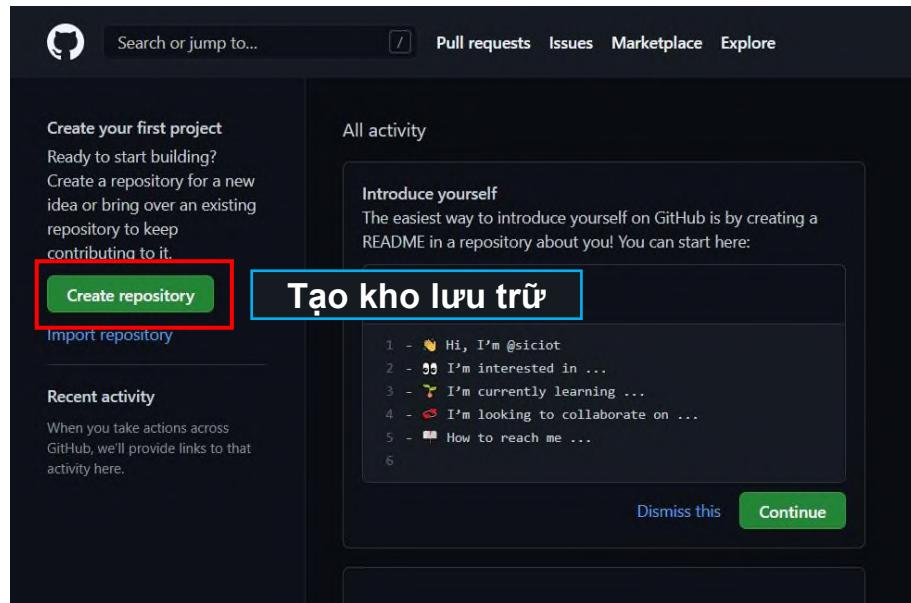
- Truy cập tới <https://github.com>
- Điền thông tin và nhấn vào mục “Sign up”.
- Chọn mục miễn phí (free).
- Tiến hành xác thực người dùng bằng email khi đăng ký.



Tạo Kho lưu trữ cá nhân

Tạo kho lưu trữ cá nhân

- Sau khi đăng nhập, nhấp vào nút **Create repository** ở bên trái.
- Đặt tên Kho lưu trữ và nhấp vào nút **Create repository** ở dưới cùng.



Tạo Kho lưu trữ cá nhân

Tạo Kho lưu trữ cá nhân

- Xác nhận rằng Kho lưu trữ từ xa có tên **example1** đã được tạo trong tài khoản siciot.
- Tiến hành như được mô tả trong ... hoặc tạo một kho lưu trữ mới thông qua các lệnh (theo hướng dẫn của Github).

The screenshot shows the GitHub repository page for 'siciot/example1'. At the top, the URL <https://github.com/siciot/example1.git> is highlighted with a blue box. Below it, under 'Quick setup — if you've done this kind of thing before', there are three options: 'Set up in Desktop' (disabled), 'HTTPS' (selected), and 'SSH'. The 'HTTPS' option is also highlighted with a red box. A blue arrow points from the highlighted URL at the top to the 'HTTPS' link here. To the right, a large blue box contains the text "...or create a new repository on the command line" followed by a series of Git commands:

```
echo "# example1" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/siciot/example1.git  
git push -u origin main
```

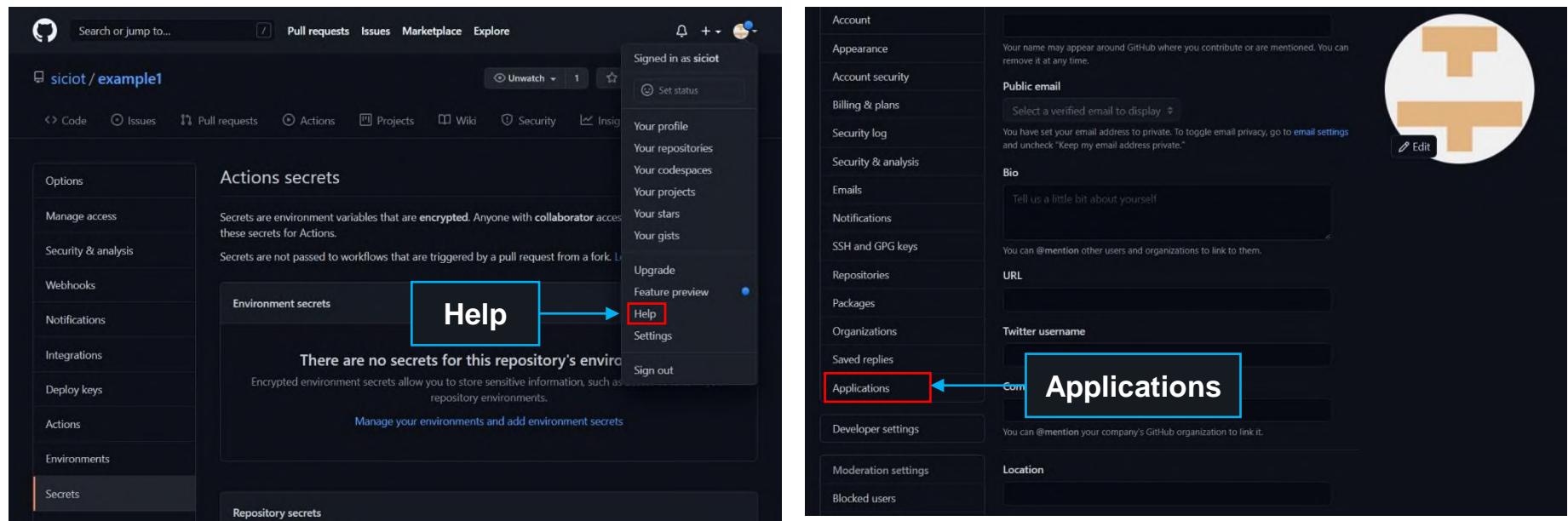
Below this, another blue box contains the text "...or push an existing repository from the command line" followed by:

```
git remote add origin https://github.com/siciot/example1.git  
git branch -M main
```

Tạo Kho lưu trữ cá nhân

Mã token truy cập cá nhân

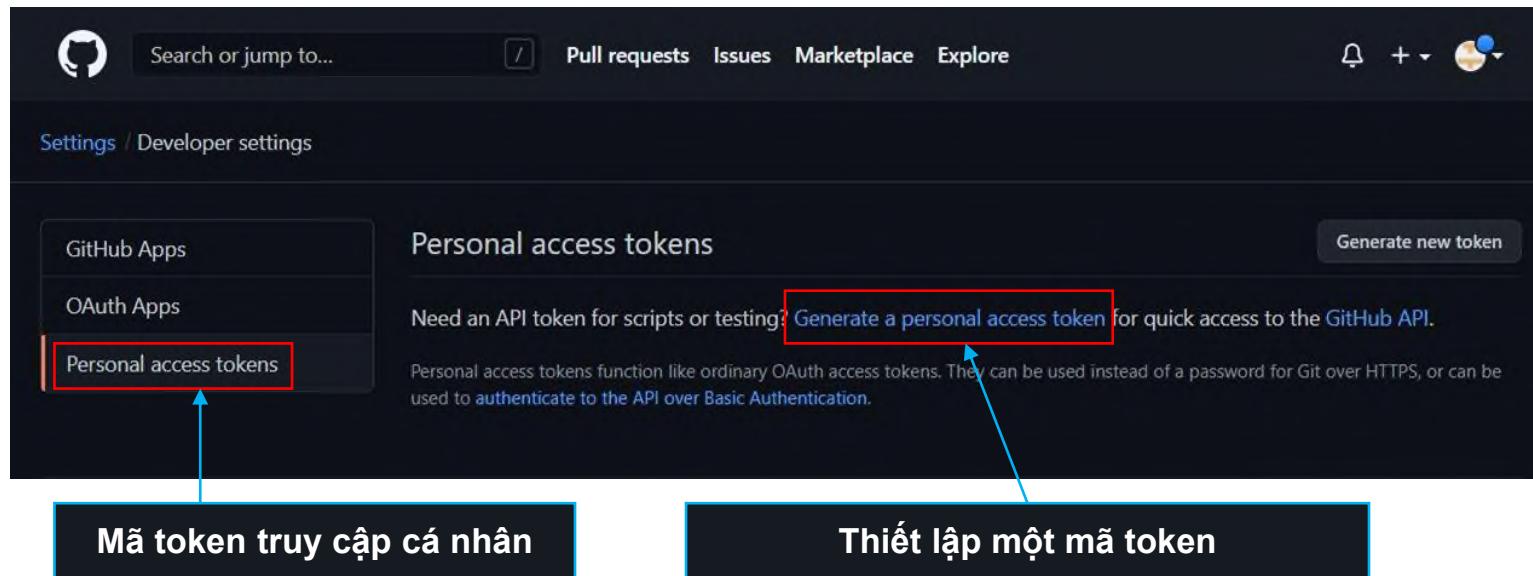
- Do sự thay đổi gần đây của phương thức xác thực của GitHub, yêu cầu một mã xác thực (token) người dùng để sử dụng cho quá trình xác thực.
- Nhập vào mũi tên xuống trên mục profile của người dùng và chọn → Settings → Developer Settings.



Tải tài liệu vào kho lưu trữ Github

Mã token truy cập cá nhân

- Nhấn vào Mục “Personal access tokens”.
- Tiếp theo, nhấn vào mục “Generate a personal access token” để thiết lập một mã token cá nhân.

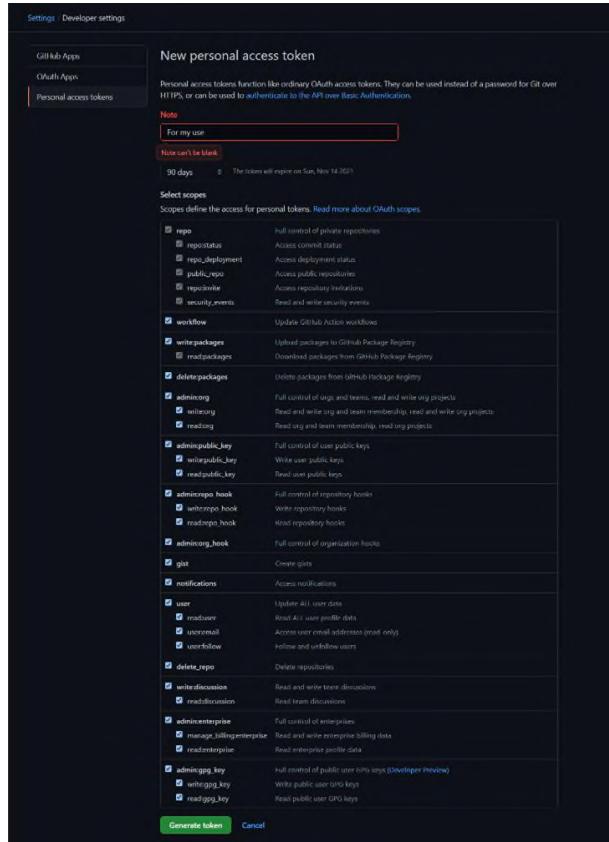


4.1. Bài luyện tập GitHub đơn giản

Bài 04

Tải tài liệu vào kho lưu trữ Github

Mã token truy cập cá nhân

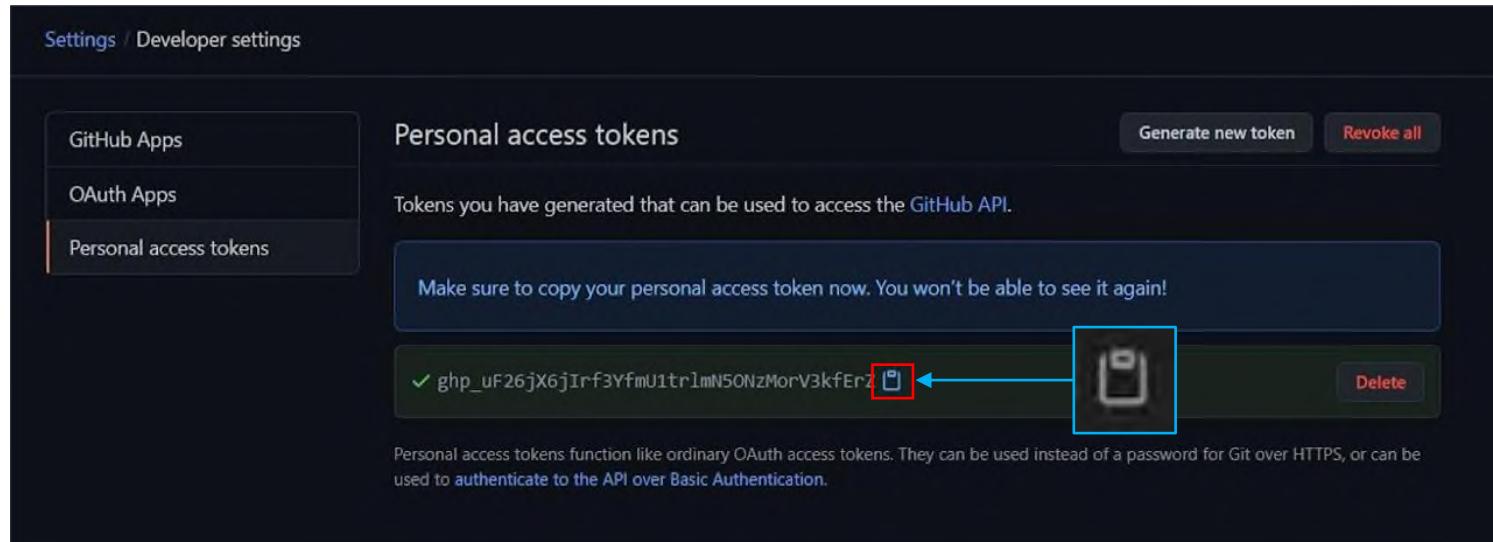


- Nếu bạn chưa biết chi tiết cách sử dụng GitHub, hãy chọn tất cả các checkbox để thoải mái sử dụng token cho bất kỳ mục đích nào.
- Trường Note được sử dụng như một ghi chú ngắn gọn về mục đích phát hành token.
- Đặt ngày hết hạn trong khoảng thời gian thuận tiện cho bạn.
- Sau khi hoàn thành, hãy nhấp vào nút **Generate token** ở dưới cùng.

Tải tài liệu vào kho lưu trữ Github

Mã token truy cập cá nhân

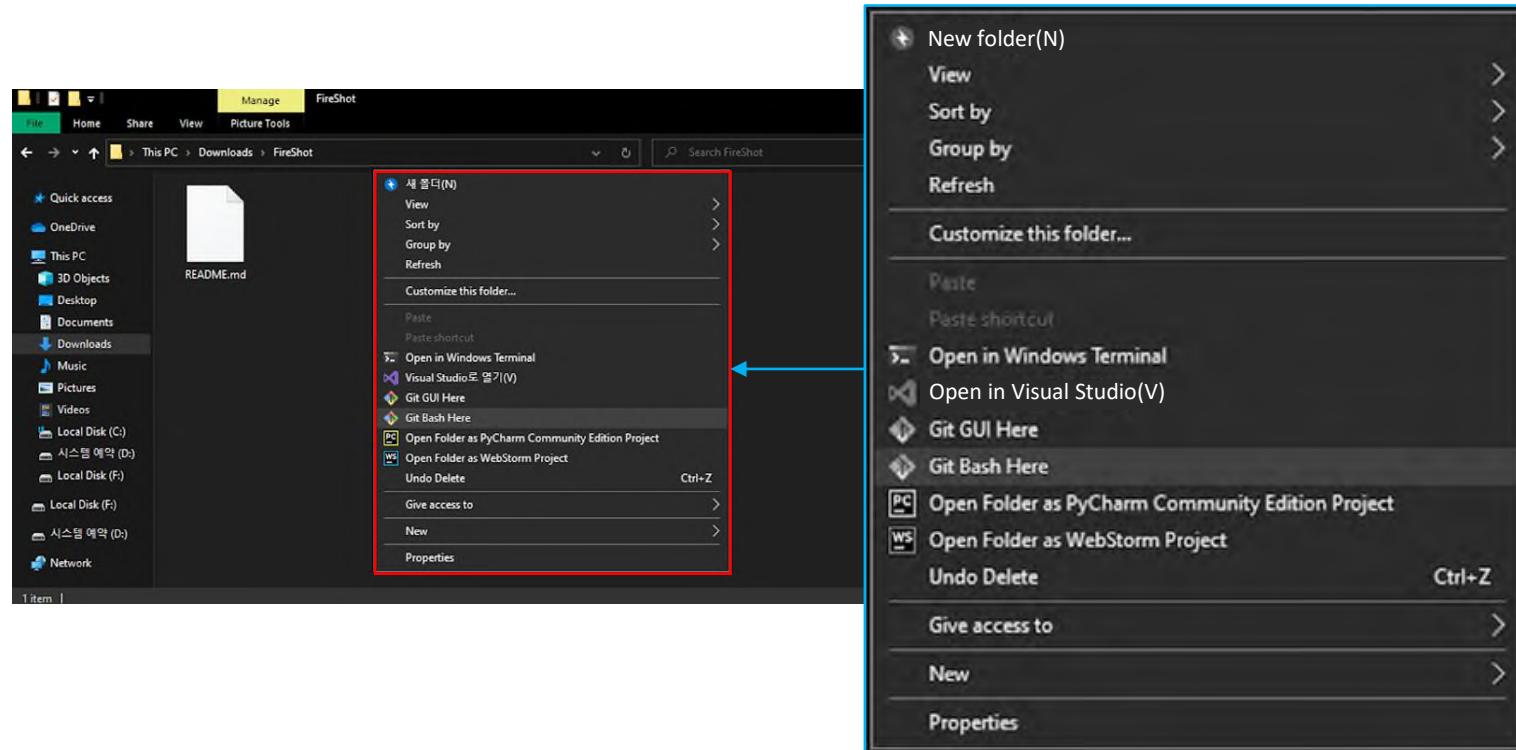
- ▶ Kiểm tra và sao chép token của bạn.
- ▶ Nhấp vào biểu tượng đĩa mềm màu xanh phía bên phải để sao chép.



Tải tài liệu vào kho lưu trữ Github

Tạo kho lưu trữ cục bộ

- Trên PC, hãy chuyển đến thư mục mà bạn muốn kết nối tới Kho lưu trữ từ xa GitHub → Nhấp chuột phải → Chọn mục “Git Bash Here”.



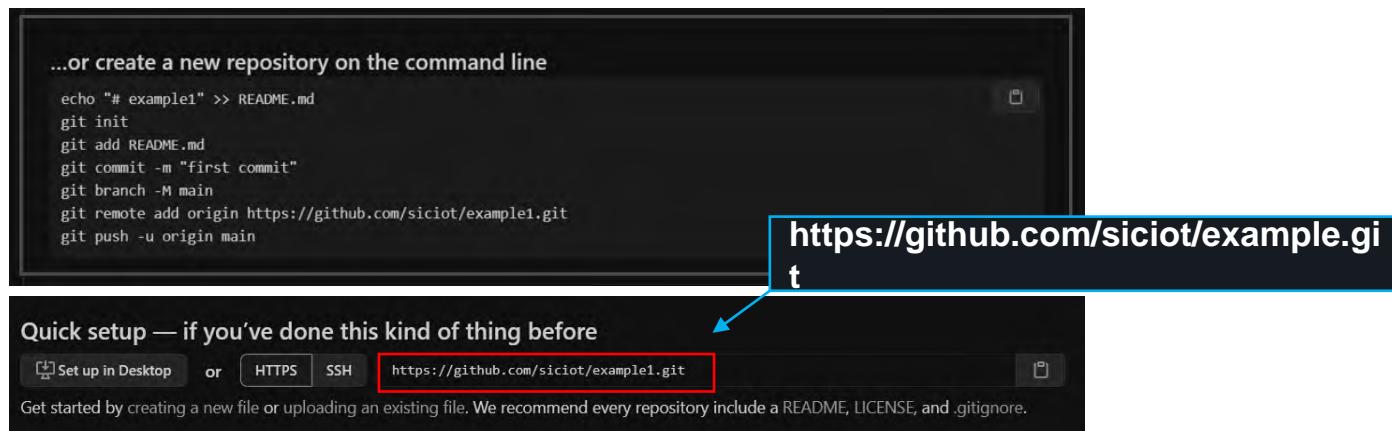
Tải tài liệu vào kho lưu trữ Github

Tạo kho lưu trữ cục bộ

- Lưu văn bản giữa dấu ngoặc kép dưới dạng file có tên README.md trong thư mục hiện tại thông qua lệnh echo. Tạo một thư mục con (thư mục ẩn) có tên .git thông qua lệnh git init.

```
$ git init
```

- Lệnh này tạo một thư mục con mới có tên .git chứa tất cả các file tạo kho lưu trữ cần thiết của bạn - một khung xương kho lưu trữ Git.
- Thêm file README.md vào vùng theo dõi của Git.
- Xác nhận và lưu các file được theo dõi bằng lệnh commit, với một thông điệp đặt giữa các dấu nháy kép.



Tải tài liệu vào kho lưu trữ Github

Tạo kho lưu trữ cục bộ

- **Ghi đè dữ liệu lên nhánh chính.**
- **Liên kết giữa kho lưu trữ cục bộ với kho lưu trữ từ xa có tên là origin và địa chỉ nằm sau origin. Lưu ý rằng địa chỉ tại thời điểm này là file git của Kho lưu trữ GitHub mà bạn sử dụng. Ngoài ra, nếu bạn sao chép và dán địa chỉ trong git bash, lỗi giao thức https không được hỗ trợ có thể xảy ra. Do đó, cần thêm https vào địa chỉ.**
- **Đẩy từ nhánh chính sang nhánh gốc (origin) (Kho lưu trữ từ xa).**

```
...or create a new repository on the command line

echo "# example1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/siciot/example1.git
git push -u origin main
```

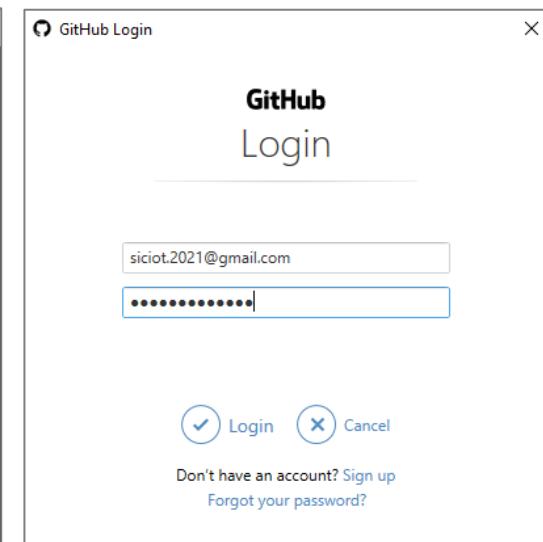
Tải tài liệu vào kho lưu trữ Github

Tạo kho lưu trữ cá nhân

- Kiểm tra các tin nhắn theo thứ tự.
- Khi thực hiện lệnh git push -u origin main. Bạn có thể thấy một cửa sổ yêu cầu bạn đăng nhập vào GitHub, sau đó thực hiện đăng nhập theo tài khoản cá nhân đã đăng ký.

```
$ git push -u origin main
```

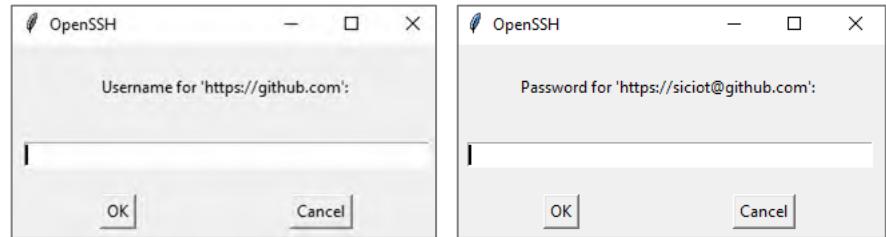
```
MINGW64 ~/Downloads/FireShot
$ echo "# example" >> README.md
$ echo "# example1" >> README.md
$ git init
Initialized empty Git repository in C:/Users/dongh...
dongh@DESKTOP-K00CA2A MINGW64 ~/Downloads/FireShot
$ git add README.md
$ git commit -m "first commit"
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your
dongh@DESKTOP-K00CA2A MINGW64 ~/Downloads/FireShot (master)
$ git commit -m "first commit"
[master (root-commit) b444ee8] first commit
 1 file changed, 1 insertion(+)
create mode 100644 README.md
dongh@DESKTOP-K00CA2A MINGW64 ~/Downloads/FireShot
$ git branch -M main
```



Tải tài liệu vào kho lưu trữ Github

Tạo kho lưu trữ cá nhân

- Sau khi đăng nhập, một cửa sổ mới hiện lên với hai câu hỏi. Khi được yêu cầu nhập tên người dùng của bạn, bạn có thể tìm thấy tên người dùng của mình được đánh dấu bằng hộp màu đỏ trong Kho lưu trữ từ xa GitHub của bạn.
- Tiếp theo, bạn sẽ được yêu cầu nhập mật khẩu. Dán mã token đã được xác nhận trước đó vào hộp nhập mã.
- Nếu push thành công, bạn có thể check nó trong cửa sổ git bash.



The screenshot shows a GitHub repository named 'siciot/example1'. The URL 'siciot/example1' is highlighted with a red box. The repository name 'siciot' is highlighted with a blue box. The page displays basic repository statistics: 1 branch and 0 tags.

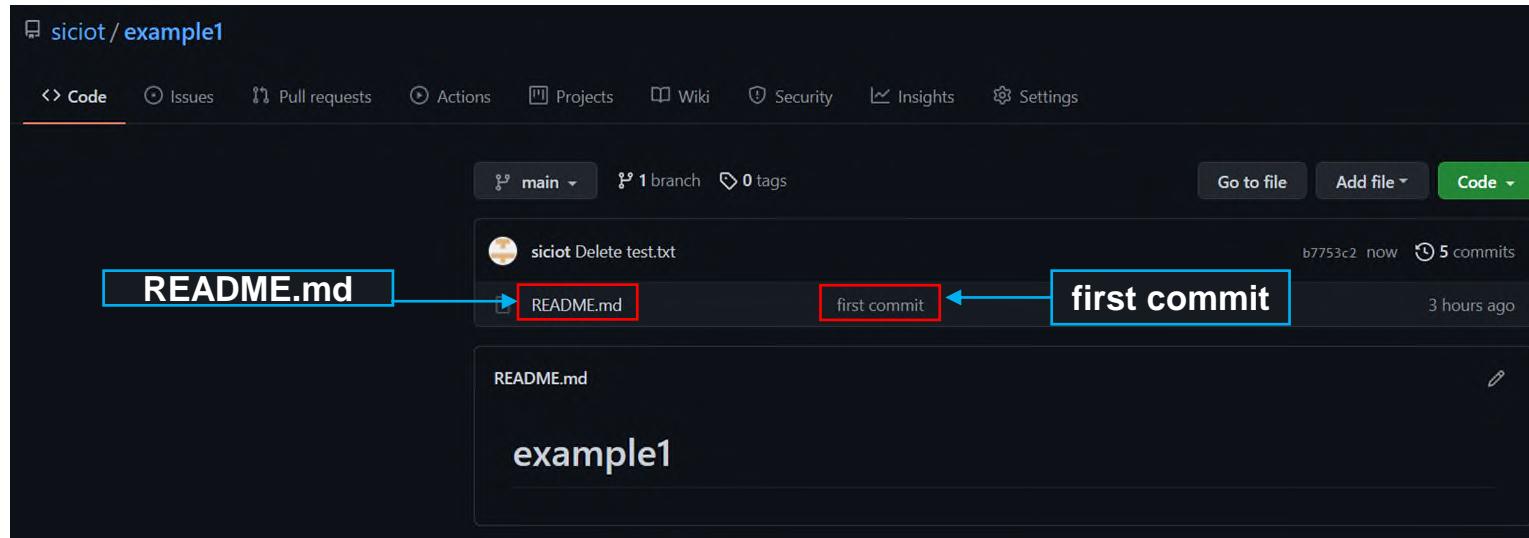
```
dongh@DESKTOP-K00CA2A MINGW64 ~/Downloads/FireShot (main)
$ git push -u origin main
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 227 bytes | 227.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/siciot/example1.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

The terminal window shows the command `$ git push -u origin main` being run. The output indicates a logon failure and provides details about the push operation, including object enumeration, writing, and tracking setup. A red box highlights the error message 'Logon failed, use ctrl+c to cancel basic credential prompt.' A blue box highlights the command `$ git push -u origin main`.

Tải tài liệu vào kho lưu trữ Github

Tạo kho lưu trữ cá nhân

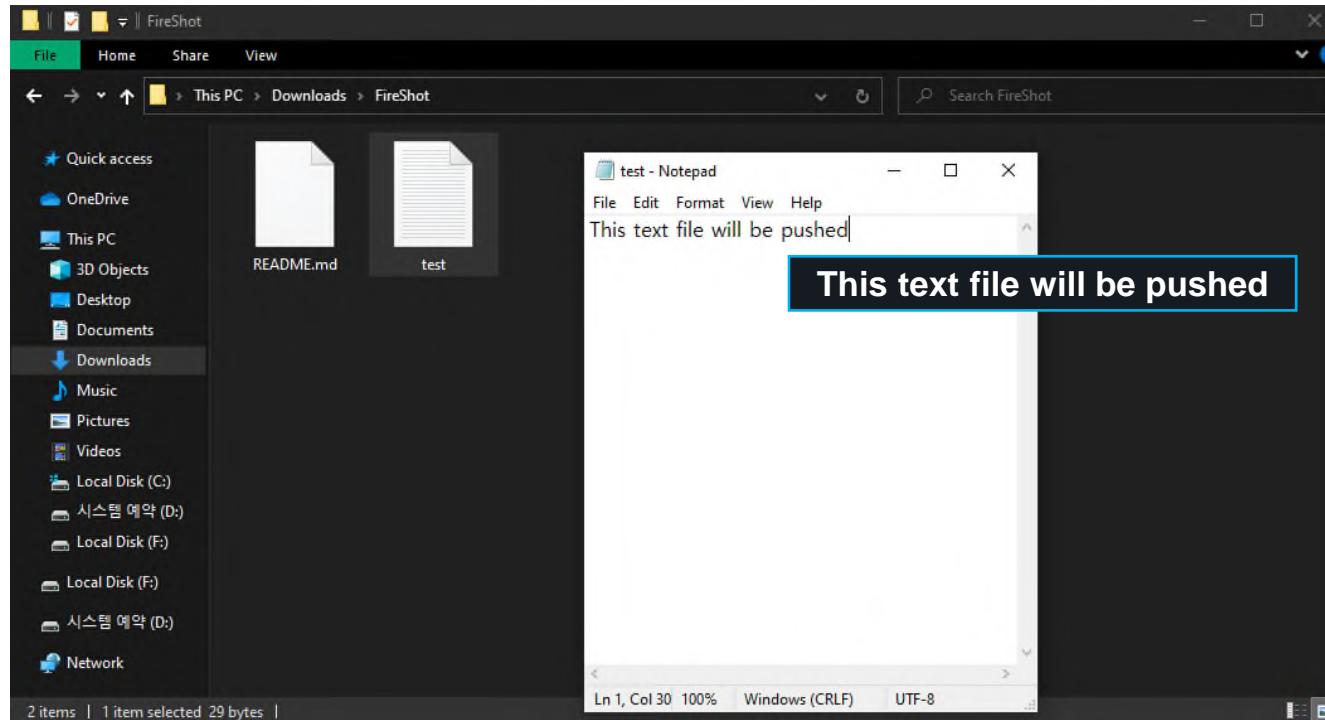
- Chúng ta hãy vào GitHub Repository để kiểm tra xem thao tác push đã được thực hiện thành công chưa. Nếu vậy, bạn có thể kiểm tra các file đã được tải lên bằng cách thêm git từ kho lưu trữ cục bộ trong kho lưu trữ từ xa trên Github.
- Bạn có thể kiểm tra thông báo đã xác nhận gần đây nhất về file hoặc thư mục trong Kho lưu trữ GitHub.



Nếu thêm file

Lưu kết quả đã chỉnh sửa

- Tạo test.txt ở cùng vị trí với README.md trong thư mục làm việc.



Nếu thêm file

Lưu kết quả đã chỉnh sửa

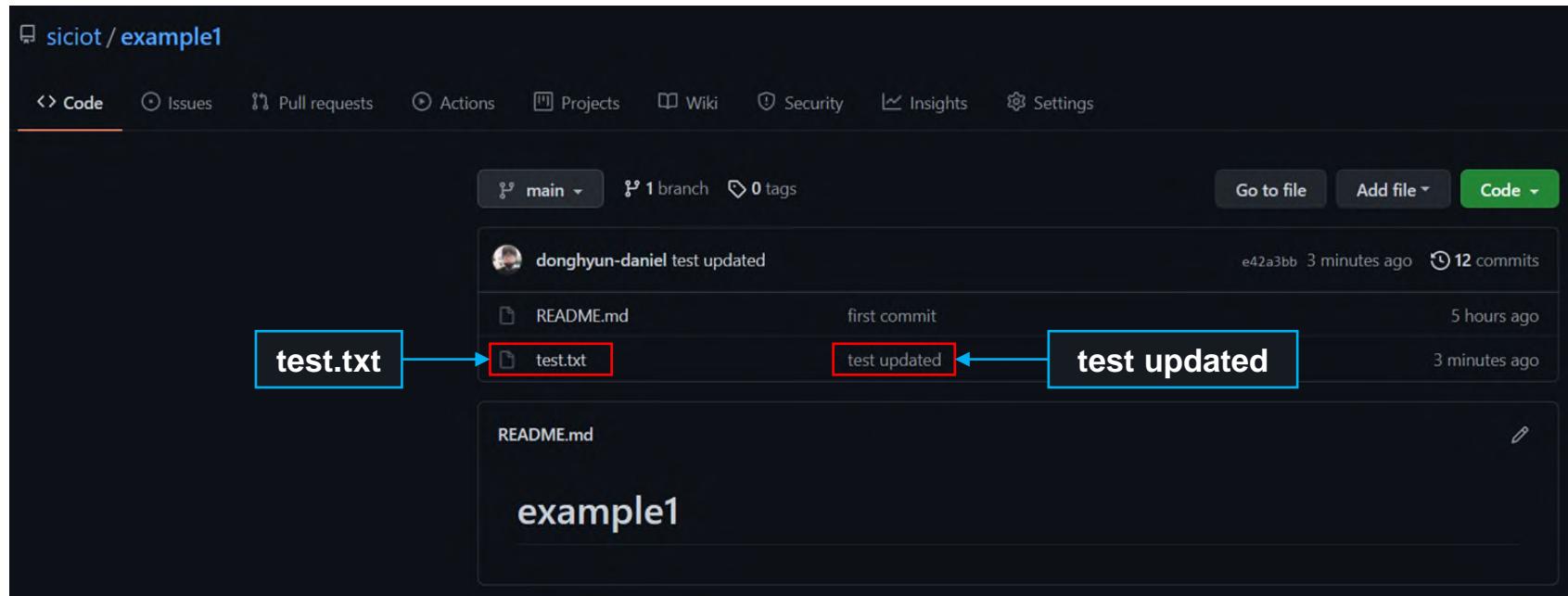
```
MINGW64:/c/Users/dongh/Downloads/FireShot
dongh@DESKTOP-K00CA2A MINGW64 ~/Downloads/FireShot (main)
$ ls
README.md test.txt
$ git add *
$ git add *
$ git commit -m "test updated"
[main e42a3bb] test updated
1 file changed, 1 insertion(+)
create mode 100644 test.txt
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 497 bytes | 497.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/siciot/example1.git
 26461ad..e42a3bb main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

- ▶ Kiểm tra lại file đã tạo.
- ▶ Tham số (*) có nghĩa là thêm tất cả các file (bao gồm file test.txt) trong thư mục hiện tại vào vùng giám sát tạm thời của Git.
- ▶ Commit các thay đổi đang được giám sát, kèm theo thông báo "test updated".
- ▶ Đẩy đến nhánh chính origin của GitHub.

Nếu thêm file

Lưu kết quả đã chỉnh sửa

- Kiểm tra văn bản đã thêm (file text.txt) trong kho lưu trữ từ xa (bấm F5 để cập nhật hiển thị thông tin trên GitHub).



Bài 4.

Cách sử dụng GitHub

- | 4.1. Bài luyện tập GitHub đơn giản
- | 4.2. Cách sử dụng GitHub khi làm việc theo cá nhân
- | 4.3. Cách sử dụng GitHub thông qua môi trường GUI

Nếu làm việc trong hệ điều hành khác

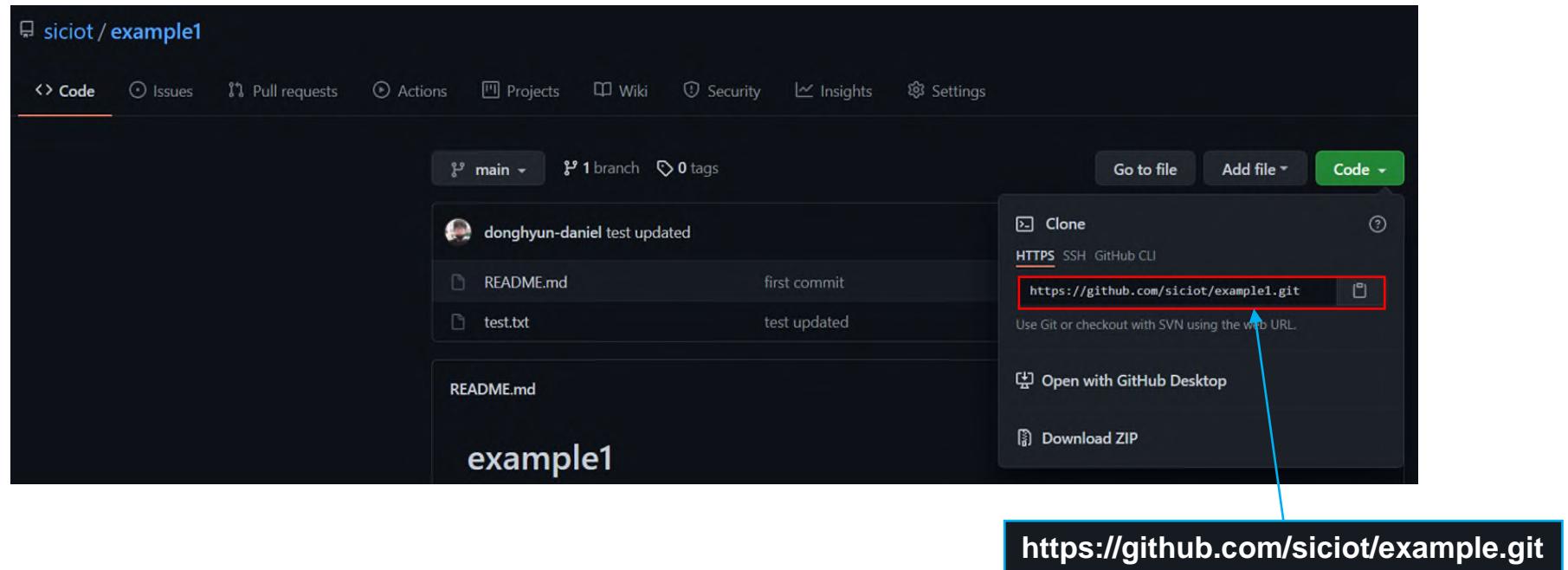
Sử dụng GitHub trong Linux

- Bạn đã tìm hiểu về cách tạo Kho lưu trữ trên GitHub bằng cách sử dụng Window 10, Commit, Push và Pull.
- GitHub cũng có thể được sử dụng trong Linux (hay trên Raspbian OS) theo cách tương tự như Windows.
- Bạn sẽ tìm hiểu trường hợp trong đó tài liệu làm việc của dự án trên Windows được tải vào kho lưu trữ từ xa GitHub và ta có thể làm việc với tài liệu này trên PC thứ hai chạy hệ điều hành Linux.

Nếu cần nhân bản dự án (1/3)

Sử dụng GitHub trong Linux

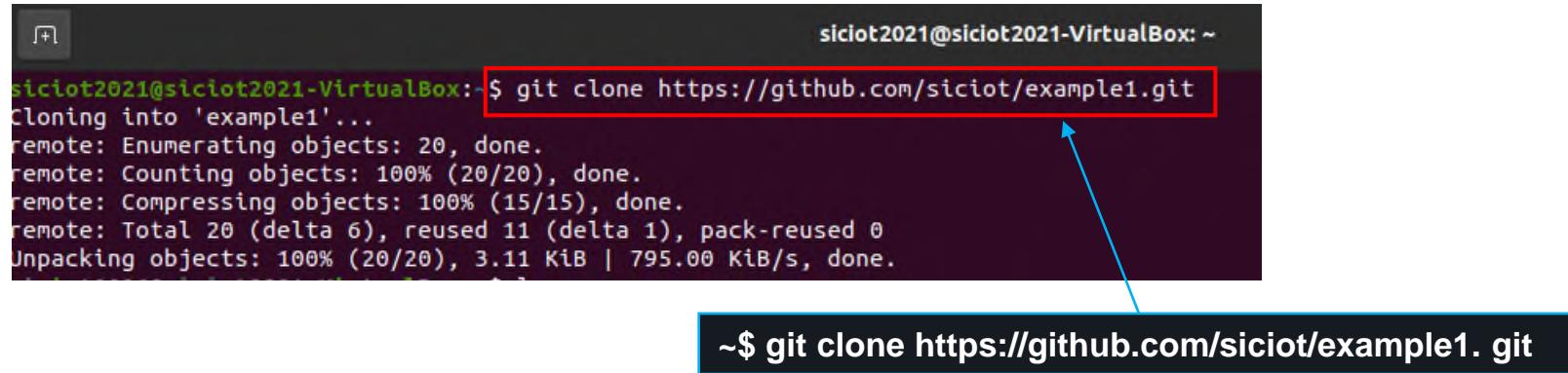
- Lấy địa chỉ của file git trong Kho lưu trữ GitHub chưa dự án hiện có, cần được truy xuất.



Nếu cần nhân bản dự án (2/3)

Sử dụng GitHub trong Linux

- Nhập một dự án hiện có bằng cách sử dụng lệnh git clone trong Linux Terminal. Đặt địa chỉ đã xác định trước đó trong câu lệnh để thực hiện sao chép (clone) git.



A screenshot of a Linux terminal window titled "siciot2021@siciot2021-VirtualBox: ~". The terminal shows the command \$ git clone https://github.com/siciot/example1.git being entered. A red box highlights the command line. The output shows the cloning process: Cloning into 'example1'..., remote: Enumerating objects: 20, done. remote: Counting objects: 100% (20/20), done. remote: Compressing objects: 100% (15/15), done. remote: Total 20 (delta 6), reused 11 (delta 1), pack-reused 0 Unpacking objects: 100% (20/20), 3.11 KiB | 795.00 KiB/s, done. A blue arrow points from the bottom text box to the highlighted command line in the terminal.

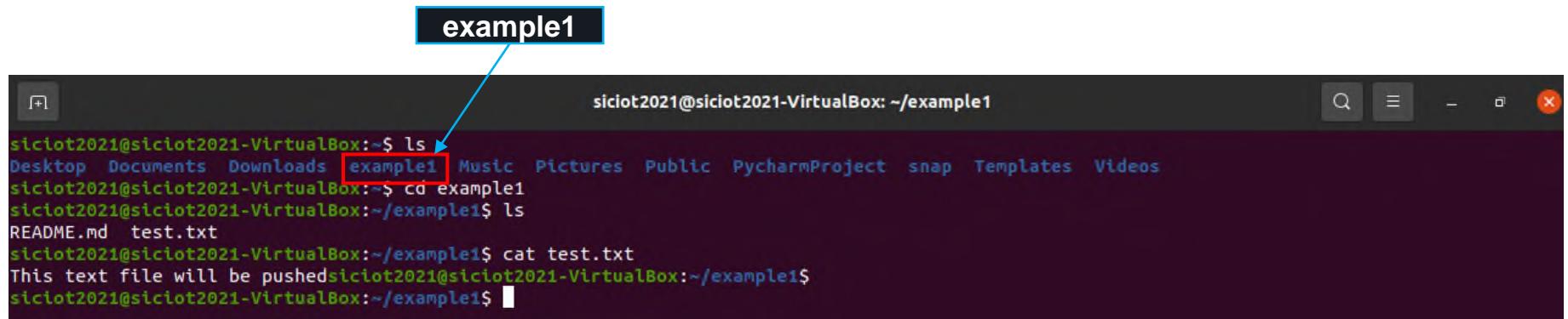
```
siciot2021@siciot2021-VirtualBox:~$ git clone https://github.com/siciot/example1.git
Cloning into 'example1'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 20 (delta 6), reused 11 (delta 1), pack-reused 0
Unpacking objects: 100% (20/20), 3.11 KiB | 795.00 KiB/s, done.
```

~\$ git clone https://github.com/siciot/example1. git

Nếu cần nhân bản dự án (3/3)

Sử dụng GitHub trong Linux

- ▶ Kiểm tra danh sách bằng lệnh ls. Bạn có thể thấy rằng thư mục có tên example1 đã được tạo.
- ▶ Di chuyển và kiểm tra nội dung. Bạn có thể thấy rằng ngay cả nội dung của test.txt được tạo trước đó cũng giống nhau.
- ▶ Khi dự án hiện có được lưu trữ trong Kho lưu trữ GitHub, bạn có thể dễ dàng nhập nó bất kể hệ điều hành nào.



The screenshot shows a terminal window with a dark background. At the top, there is a blue header bar with the text "example1". Below the header, the terminal prompt is "siciot2021@siciot2021-VirtualBox: ~/example1". The terminal window contains the following text:

```
siciot2021@siciot2021-VirtualBox:~$ ls
Desktop Documents Downloads example1 Music Pictures Public PycharmProject snap Templates Videos
siciot2021@siciot2021-VirtualBox:~$ cd example1
siciot2021@siciot2021-VirtualBox:~/example1$ ls
README.md test.txt
siciot2021@siciot2021-VirtualBox:~/example1$ cat test.txt
This text file will be pushed
siciot2021@siciot2021-VirtualBox:~/example1$
```

A red box highlights the "example1" folder in the first "ls" command output. A blue arrow points from the word "example1" in the blue header bar down to the highlighted "example1" in the terminal output.

Nếu cần chỉnh sửa file trên Linux (1/6)

Sử dụng GitHub trong Linux

- Sử dụng Pycharm nếu bạn không quen thuộc với trình soạn thảo Linux cơ bản.
- Pycharm là một IDE rất phổ biến có thể chỉnh sửa và kiểm soát các file thuộc nhiều loại khác nhau cũng như Python.
- Trong Pycharm, có thể chọn phiên bản Python cho mỗi mã lệnh và nó hỗ trợ cài đặt gói dễ dàng.
- Trong Pycharm, việc thực thi mã rất đơn giản, quản lý hàm dễ dàng và các chức năng cho môi trường ảo được hỗ trợ như gốc.
- Pycharm luôn là một IDE được xếp hạng cao vì nhiều ưu điểm của nó, bao gồm nhiều plugin khác nhau.

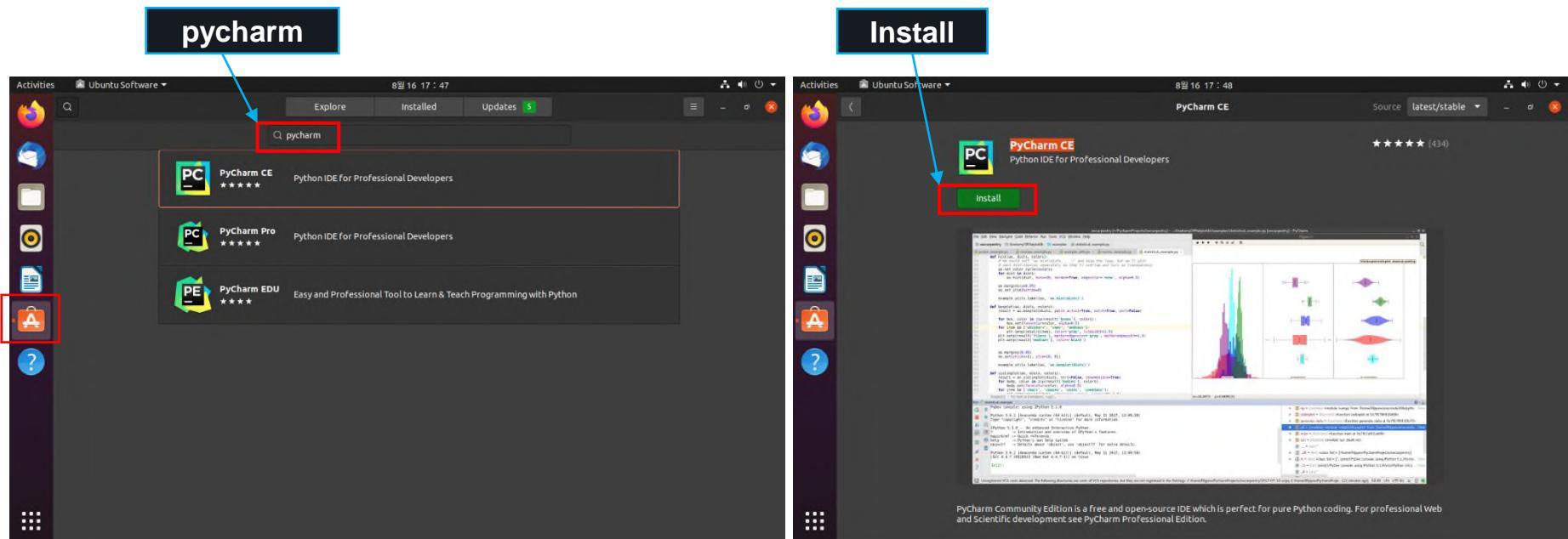


Nếu cần chỉnh sửa file trên Linux (2/6)

Sử dụng GitHub trong Linux

- Cài đặt Pycharm rất đơn giản.

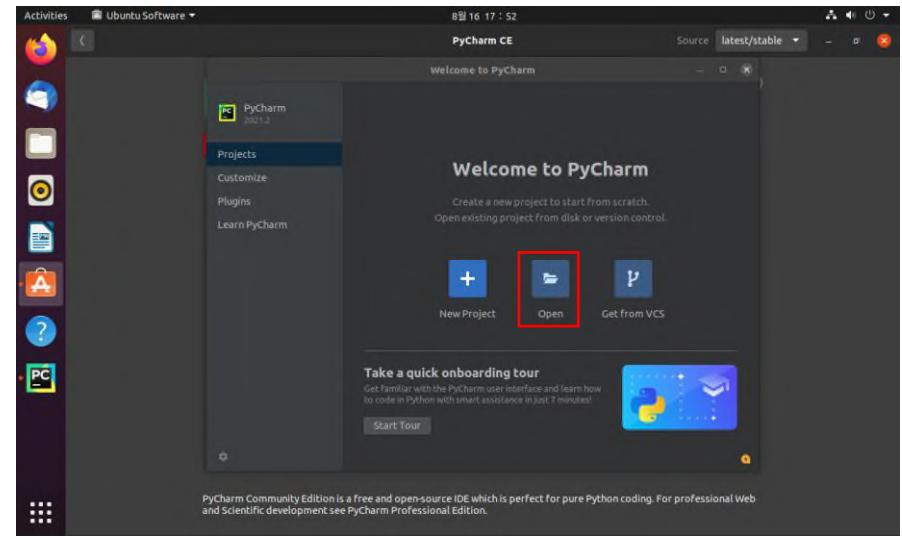
Chạy Ubuntu Software, tìm kiếm Pycharm và nhấp vào nút Install để cài đặt nó.



Nếu cần chỉnh sửa file trên Linux (3/6)

Sử dụng GitHub trong Linux

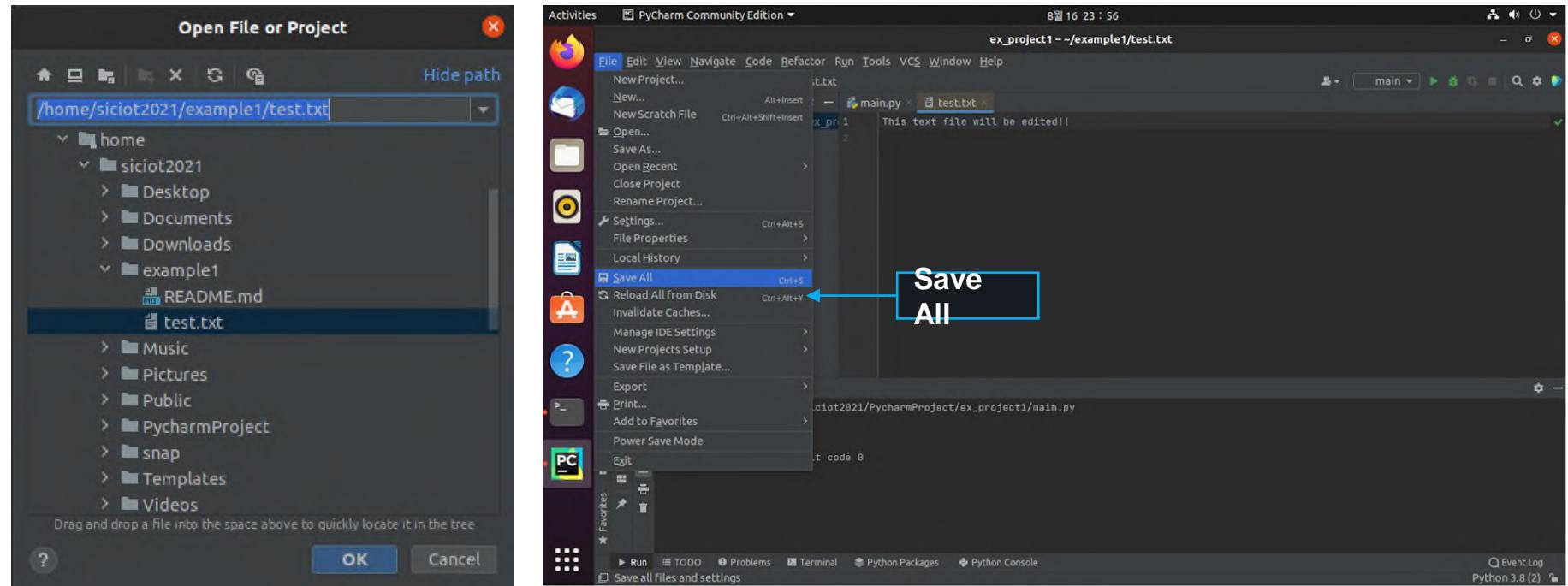
- Nhấp vào Show Applications in góc dưới bên trái và nhấp vào Pycharm. Nếu bạn không thấy biểu tượng này, hãy tìm kiếm Pycharm.
- Nhấp vào Open để duyệt file.



Nếu cần chỉnh sửa file trên Linux (4/6)

Sử dụng GitHub trong Linux

- Tiến hành chỉnh sửa nội dung của file test.txt thuộc example1 được nhân bản trước đó.
- Sau khi chỉnh sửa nội dung, nhấp vào Save All để lưu và thoát chương trình.



Nếu cần chỉnh sửa file trên Linux (5/6)

Sử dụng GitHub trong Linux

- Thêm vào mục tiêu cần theo dõi trong cùng một luồng xử lý tương tự trong Window.
- Thực hiện commit, kèm theo một thông báo xác nhận.**
- Thực hiện thao tác Push tới nhánh chính (main) của nhánh gốc.

```
siciot2021@siciot2021-VirtualBox:~/example1$ git add *
siciot2021@siciot2021-VirtualBox:~/example1$ git commit -m "edit test.txt"
[main a69de53] edit test.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
siciot2021@siciot2021-VirtualBox:~/example1$ git push origin main
Username for 'https://github.com': siciot
Password for 'https://siciot@github.com':
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 579 bytes | 579.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/siciot/example1.git
  e42a3bb..a69de53  main -> main
siciot2021@siciot2021-VirtualBox:~/example1$
```

git add *

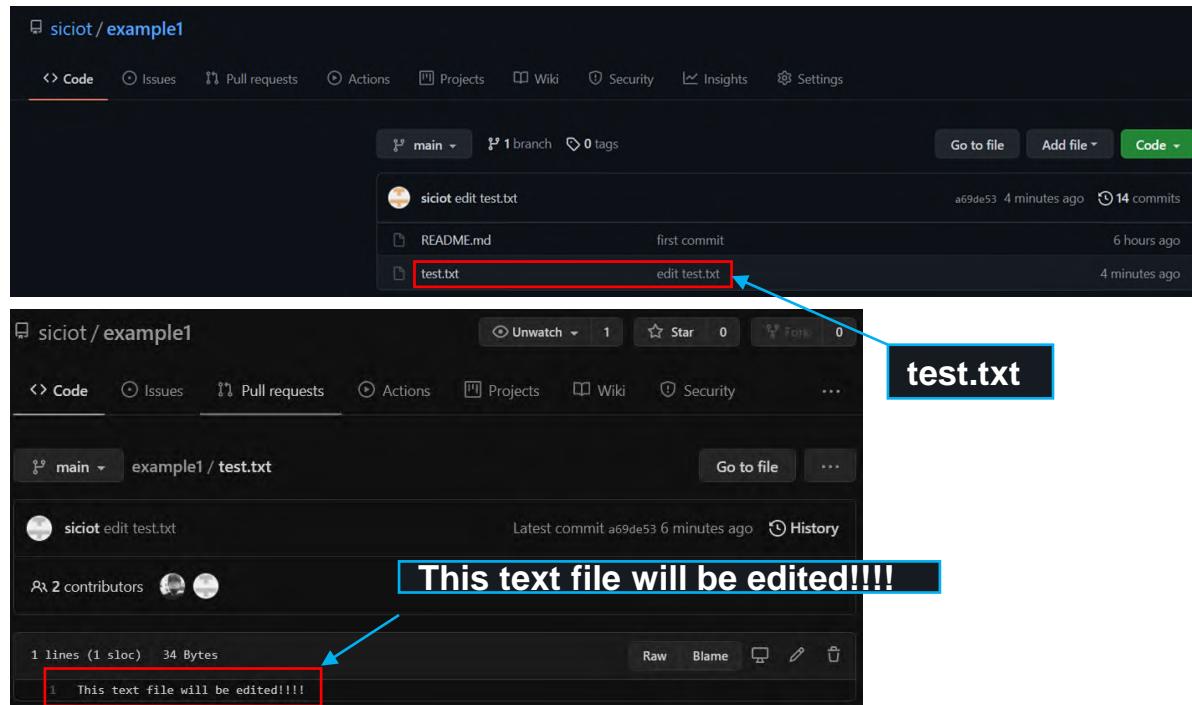
git commit -m “edit test.txt”

git push origin main

Nếu cần chỉnh sửa file trên Linux (6/6)

Sử dụng GitHub trong Linux

- Kiểm tra kho lưu trữ từ xa GitHub. Xác nhận rằng tệp test.txt đã được chỉnh sửa và ánh xạ lại cùng với một thông báo xác nhận kiểm tra chỉnh sửa.txt.



Tổng kết

| Nếu cần lưu một dự án hiện có trên PC cục bộ của bạn vào Kho lưu trữ GitHub, hãy tiến hành như sau

- ▶ Chuẩn bị thư mục project để sử dụng git → git init

```
$ git init
```

- ▶ Gộp nội dung của thư mục dự án vào vùng theo dõi của git bằng lệnh git add * (Nếu bạn chỉ muốn gộp một số tệp vào trong đống theo dõi, hãy nhập tên cụ thể thay vì *)

```
$ git add *
```

- ▶ Thực hiện lệnh commit để ghi lại (lưu lại) các thay đổi trong đống tượng theo dõi → git commit.

```
$ git commit
```

- ▶ Thực hiện lệnh Push để ánh xạ các thay đổi với Kho lưu trữ từ xa.

```
$ git push
```

Tổng kết

| Nếu muốn gọi tới và thao tác với project nằm trong GitHub về PC cục bộ, thực hiện thao tác nhập (import) sử dụng địa chỉ .git của GitHub Repository → git clone.

```
$ git clone
```

| Mặc dù phương pháp xử lý file có điểm khác nhau, hai HĐH Linux và Windows không có nhiều điểm khác nhau nhiều khi đề cập tới quy trình sử dụng Git.

Bài 4.

Cách sử dụng GitHub

- | 4.1. Bài luyện tập GitHub đơn giản
- | 4.2. Cách sử dụng GitHub khi làm việc theo cá nhân
- | 4.3. Cách sử dụng GitHub thông qua môi trường GUI

Hướng dẫn sử dụng GUI

| Phần mềm mã nguồn mở hỗ trợ GUI sử dụng GitHub

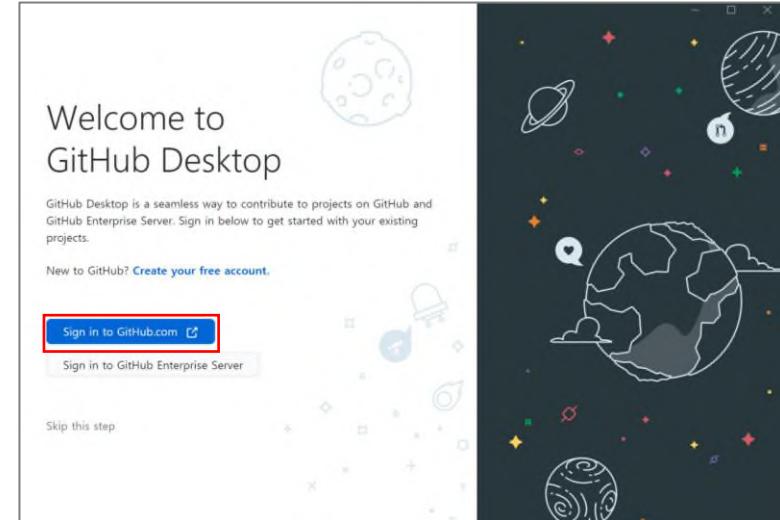
- ▶ Giới thiệu GitHub Desktop, một phần mềm mã nguồn mở cung cấp môi trường GUI (Giao diện đồ họa người dùng – Graphic User Interface) cho người dùng chưa quen thuộc với môi trường CLI (Giao diện dòng lệnh – Command Line Interface).
- ▶ GitHub Desktop cung cấp môi trường GUI dễ sử dụng và trực quan hơn so với môi trường CLI.



GitHub Desktop (1/11)

Cài đặt GitHub Desktop

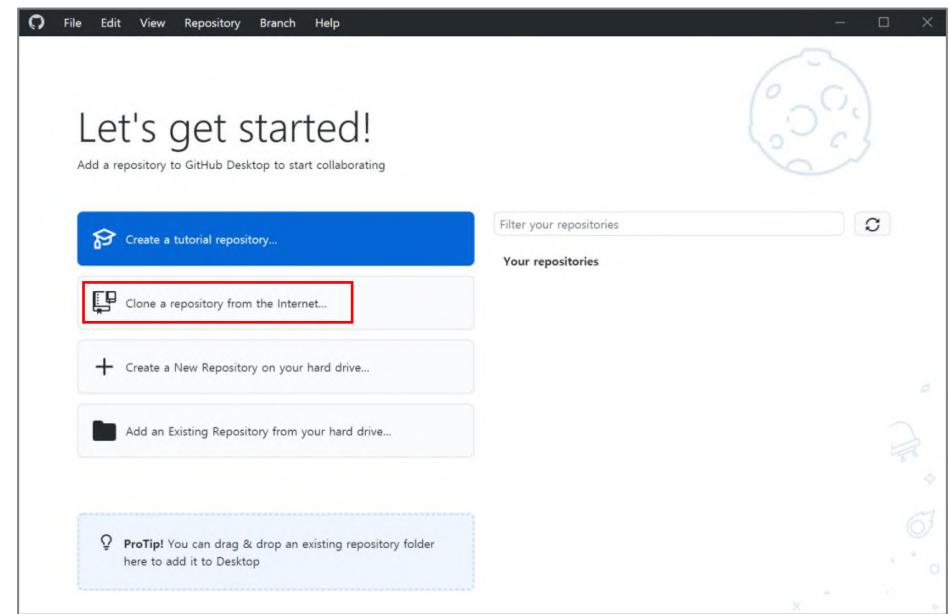
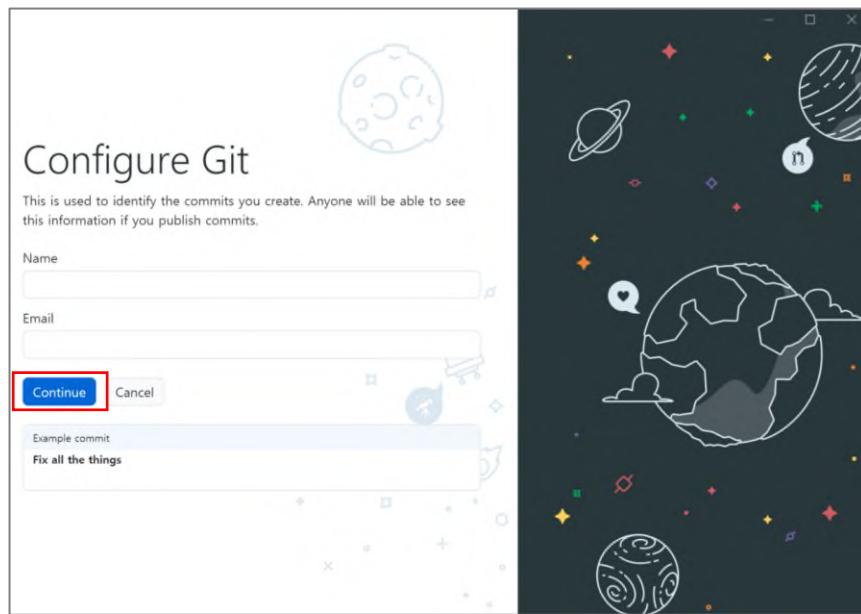
- Chạy file .exe sau khi tải xuống theo HĐH và môi trường.
- Đăng nhập bằng tài khoản GitHub của bạn.



GitHub Desktop (2/11)

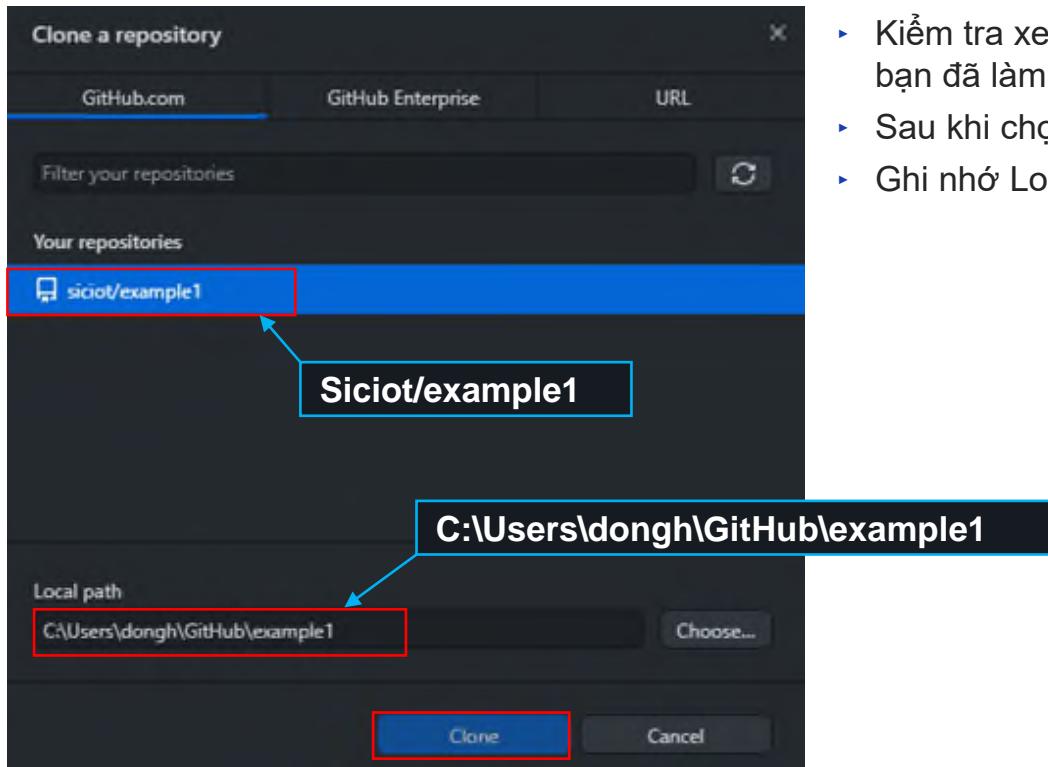
Cài đặt GitHub Desktop

- Đặt tên và địa chỉ e-mail cho commit và nhấn nút Continue (chức năng tương tự như git config... trong môi trường CLI).
- Sau quá trình xác thực, hãy nhấp vào **clone a repository from the Internet...**



GitHub Desktop (3/11)

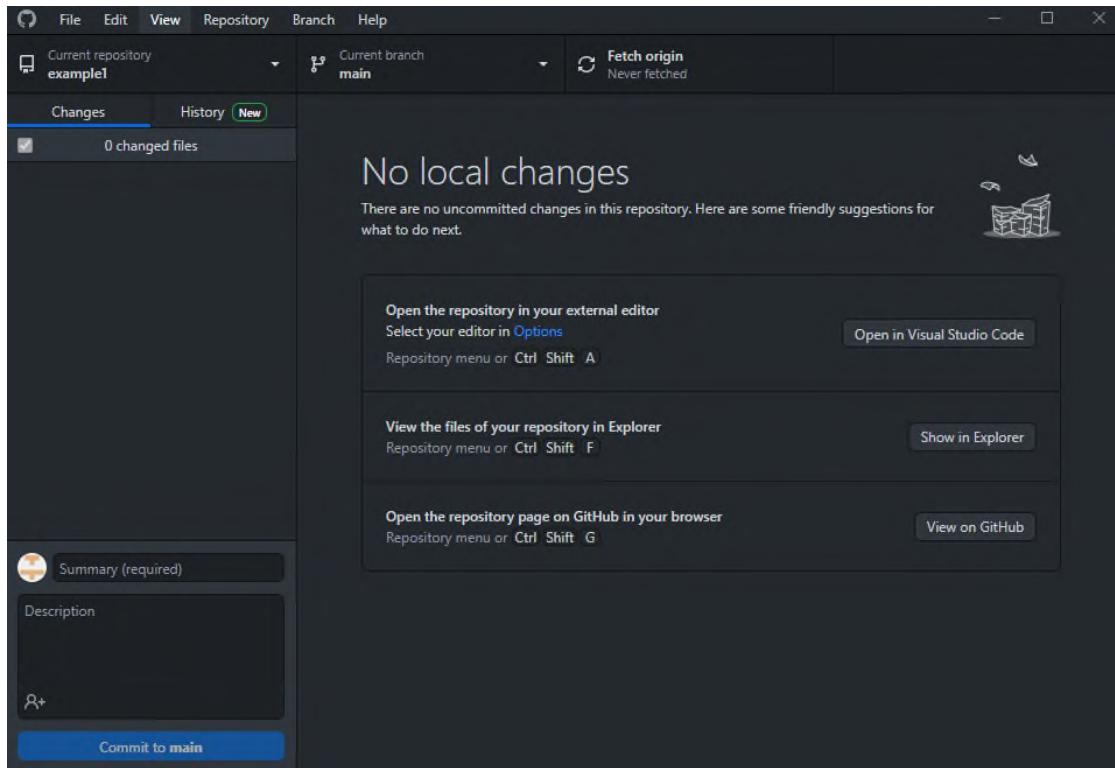
Sử dụng GitHub Desktop



- ▶ Kiểm tra xem bạn có thể tìm thấy Kho lưu trữ example1 mà bạn đã làm việc trước đó không.
- ▶ Sau khi chọn, hãy nhấp vào **Clone** ở dưới cùng.
- ▶ Ghi nhớ Local Path (đường dẫn tới thư mục chứa).

GitHub Desktop (4/11)

Sử dụng GitHub Desktop

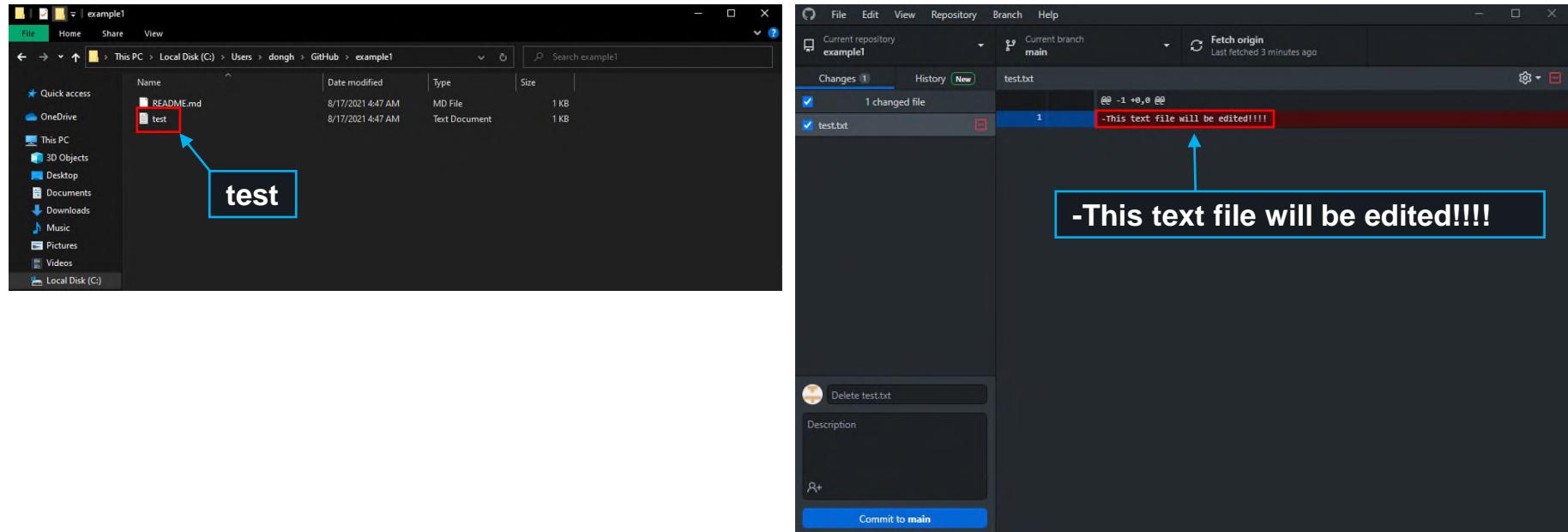


- Truy cập vào bản sao của example1.
- Chỉnh sửa kho lưu trữ cục bộ trong môi trường Windows để kiểm tra xem nó có được quản lý tốt ngay cả khi không có dòng lệnh cho git hay không.

GitHub Desktop (5/11)

Sử dụng GitHub Desktop

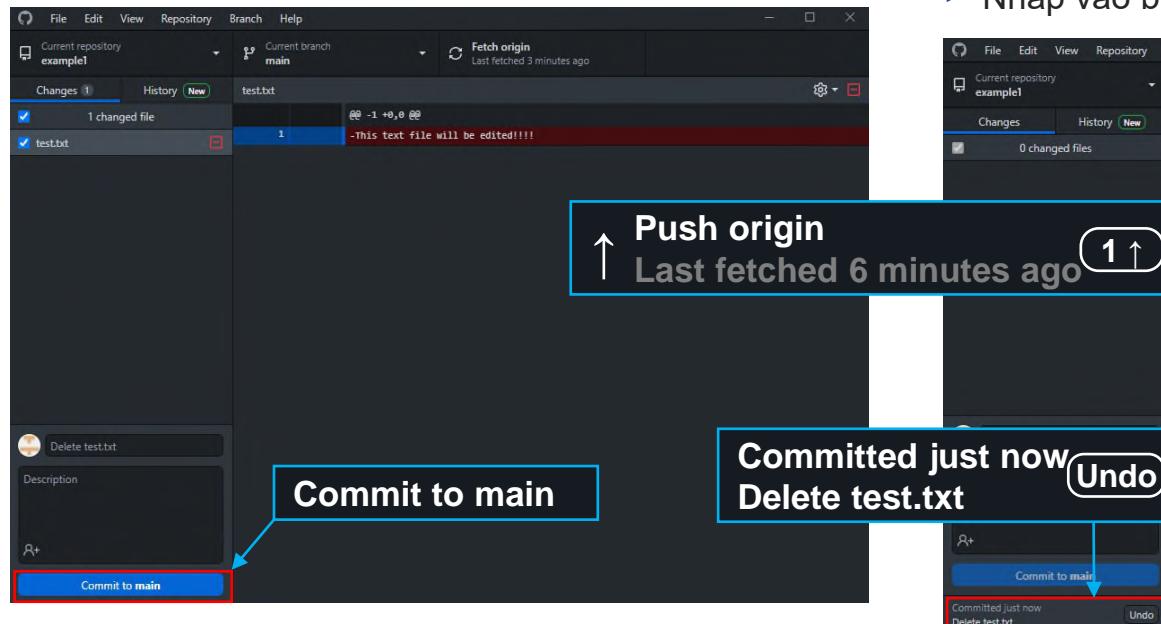
- Xóa file test.txt nằm trong thư mục example1, theo đường dẫn cục bộ.
- Khi quay trở lại chương trình GitHub Desktop, bạn có thể xác nhận rằng file test.txt đã được xác định. Ta có thể quan sát thông báo khi thực hiện lệnh commit.



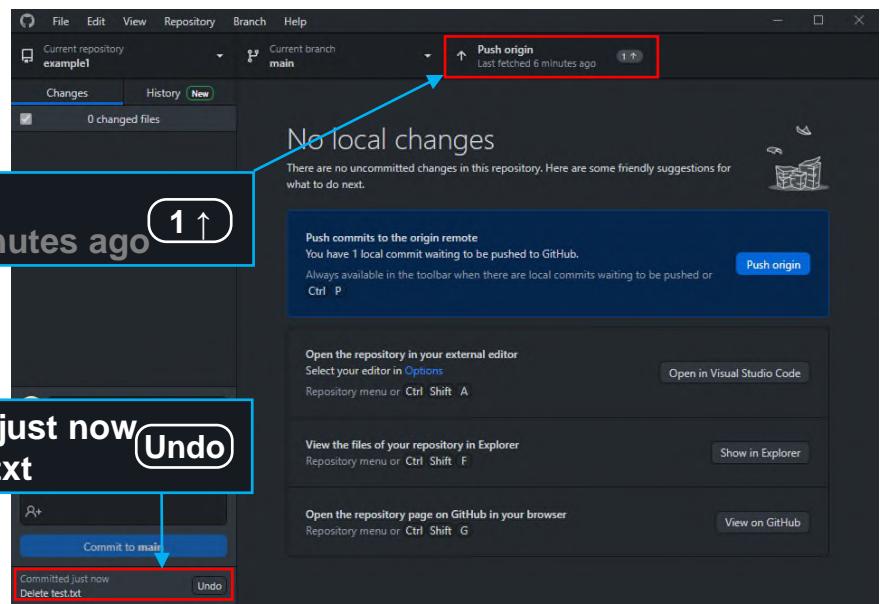
GitHub Desktop (6/11)

Sử dụng GitHub Desktop

- Nếu bạn muốn đẩy trạng thái đã xóa này vào Kho lưu trữ GitHub, hãy nhấp vào **Commit to main** vì bạn cần commit trước.



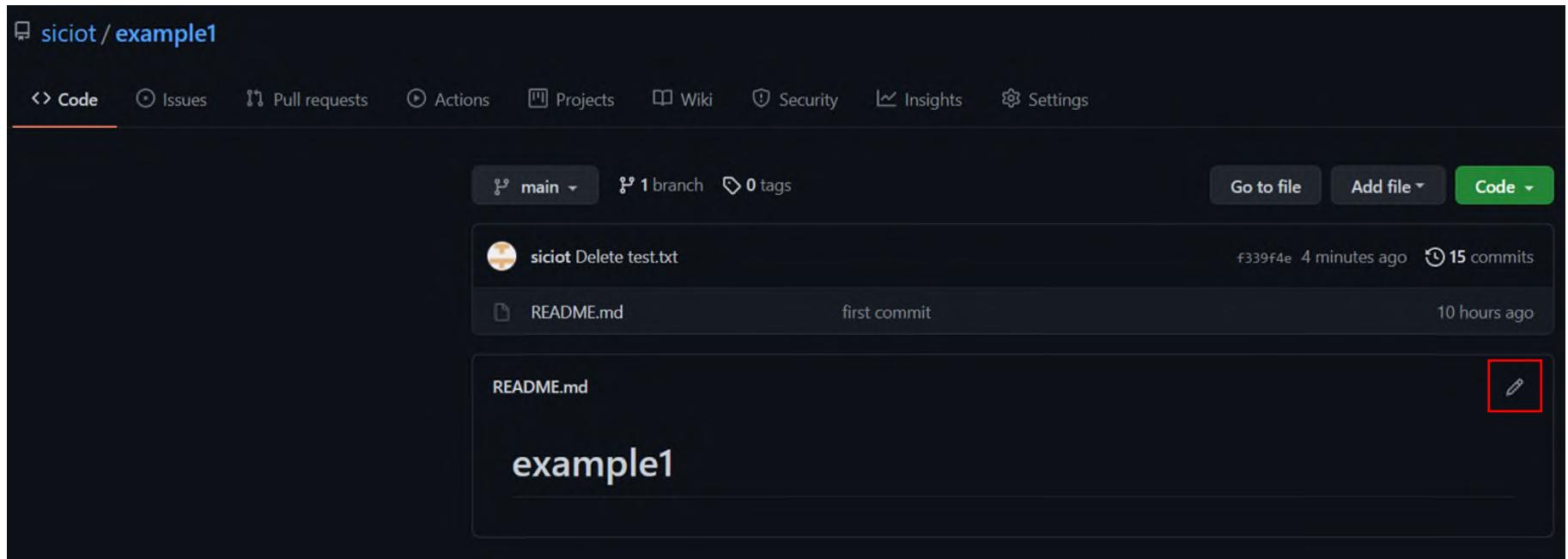
- Kiểm tra nút ở trên đã thay đổi thành **Push origin hay chưa**, và thông báo Committed lúc này ở dưới cùng bên trái.
- Nhấp vào biểu tượng Push origin.



GitHub Desktop (7/11)

Sử dụng GitHub Desktop

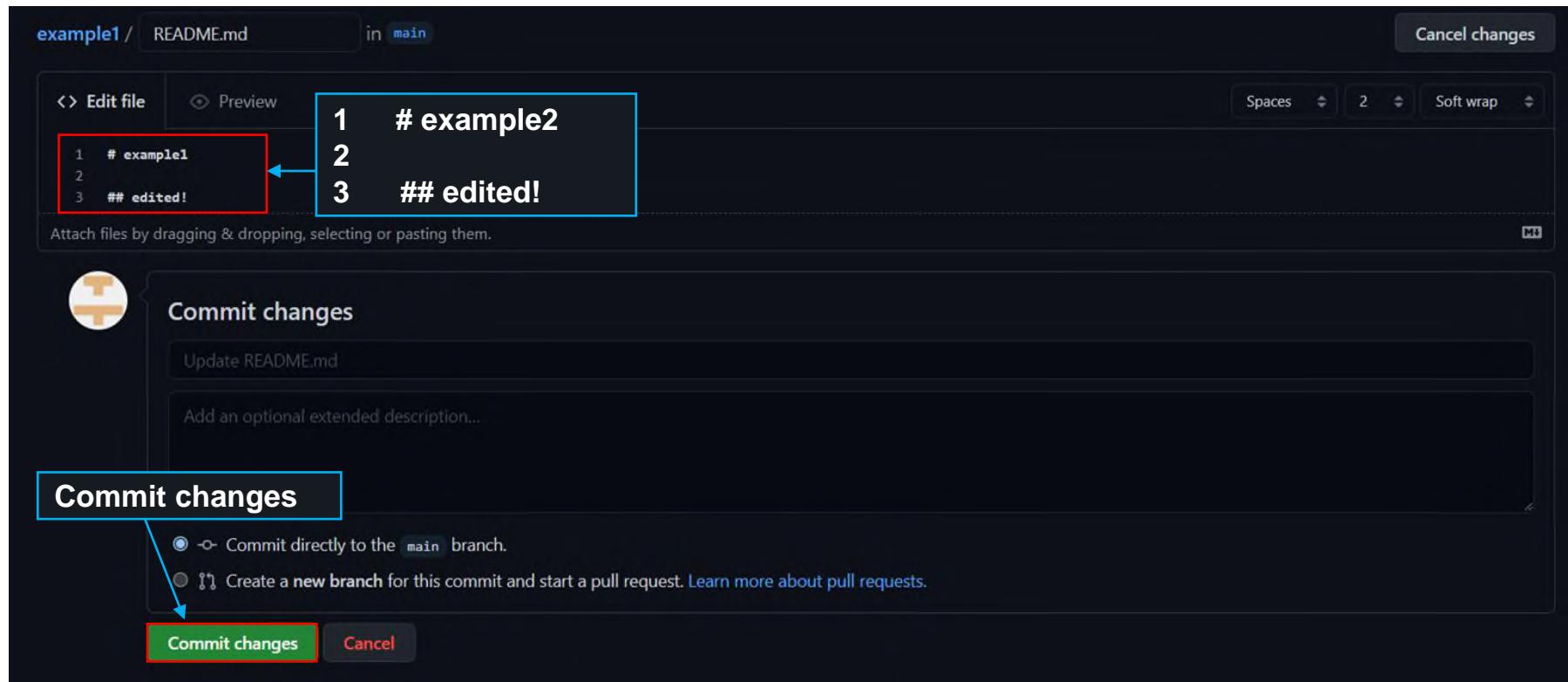
- ▶ Kiểm tra GitHub Repository để xem thao tác push có thành công không.
- ▶ Kiểm tra file test.txt đã bị xóa và ánh xạ sang chưa.
- ▶ Lần này thao tác ngược lại, nhấp vào biểu tượng chỉnh sửa (**pen icon**) để chỉnh sửa nội dung của Kho lưu trữ GitHub và kiểm tra trên máy cục bộ.



GitHub Desktop (8/11)

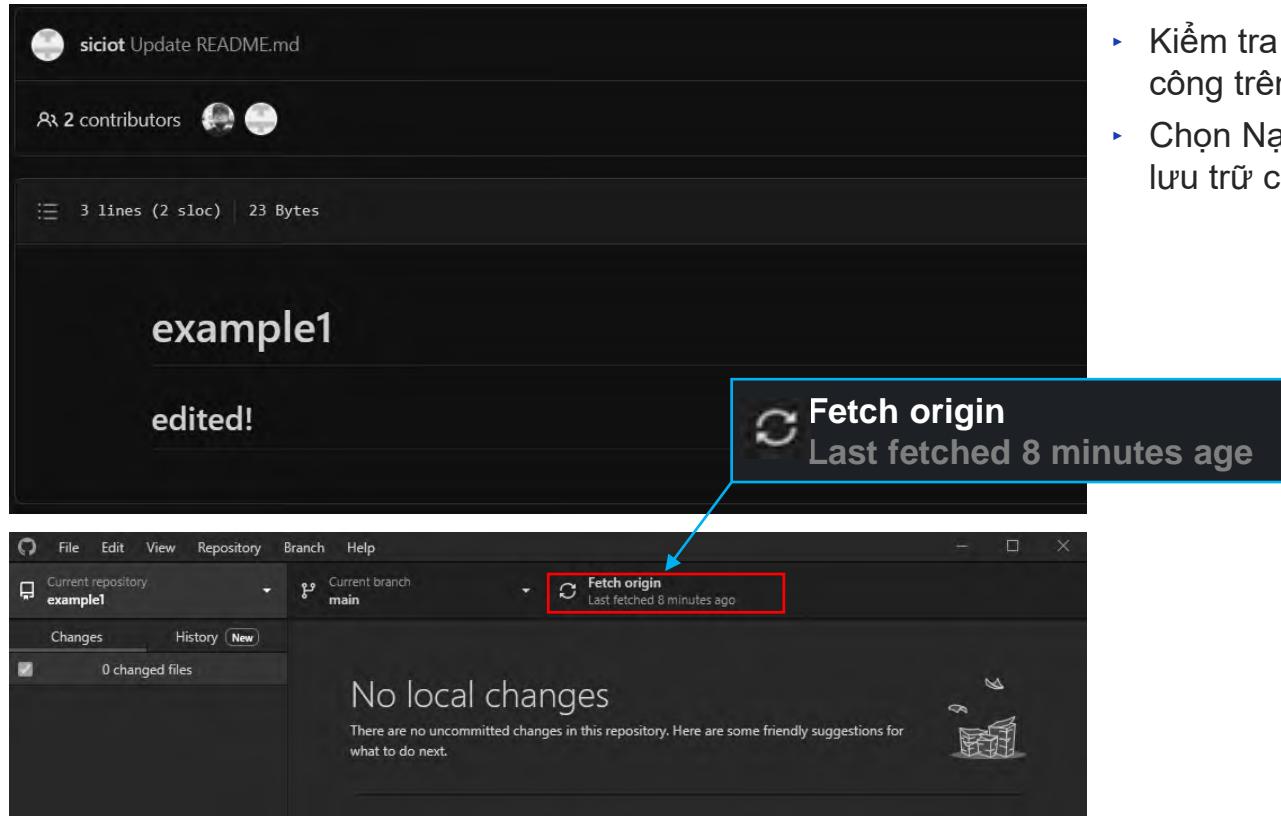
Sử dụng GitHub Desktop

- Chọn commit changes sau khi chỉnh sửa file README.md



GitHub Desktop (9/11)

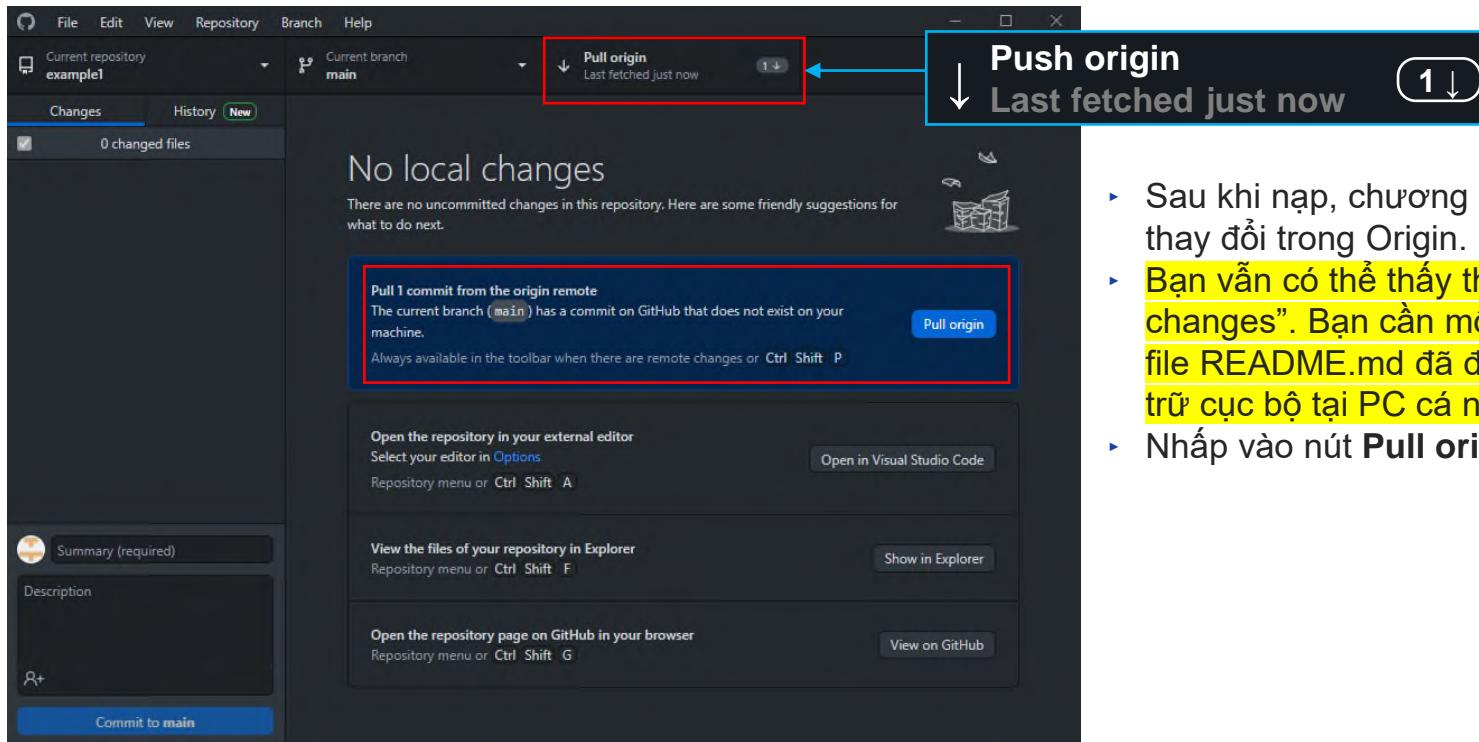
Sử dụng GitHub Desktop



- ▶ Kiểm tra xem nó đã được chỉnh sửa thành công trên Kho lưu trữ GitHub chưa.
- ▶ Chọn Nạp (Fetch) để kiểm tra thay đổi vì kho lưu trữ cục bộ sẽ không được chỉnh sửa.

GitHub Desktop (10/11)

Sử dụng GitHub Desktop

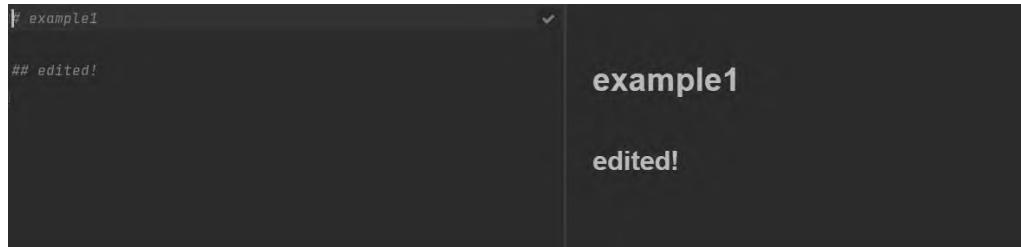


- ▶ Sau khi nạp, chương trình thông báo đã có sự thay đổi trong Origin.
- ▶ Bạn vẫn có thể thấy thông báo “No local changes”. Bạn cần một thao tác Pull để ánh xạ file README.md đã được thay đổi vào Kho lưu trữ cục bộ tại PC cá nhân.
- ▶ Nhấp vào nút **Pull origin**.

GitHub Desktop (11/11)

Sử dụng GitHub Desktop

- Đi tới Kho lưu trữ cục bộ để kiểm tra file được đánh dấu. Nếu thao tác pull thành công, có thể thấy các sửa đổi đã được ánh xạ xong.
- Như vậy, chúng ta đã học phương thức trực quan và đơn giản để Sử dụng GitHub Desktop và thử nghiệm với chức năng pull.

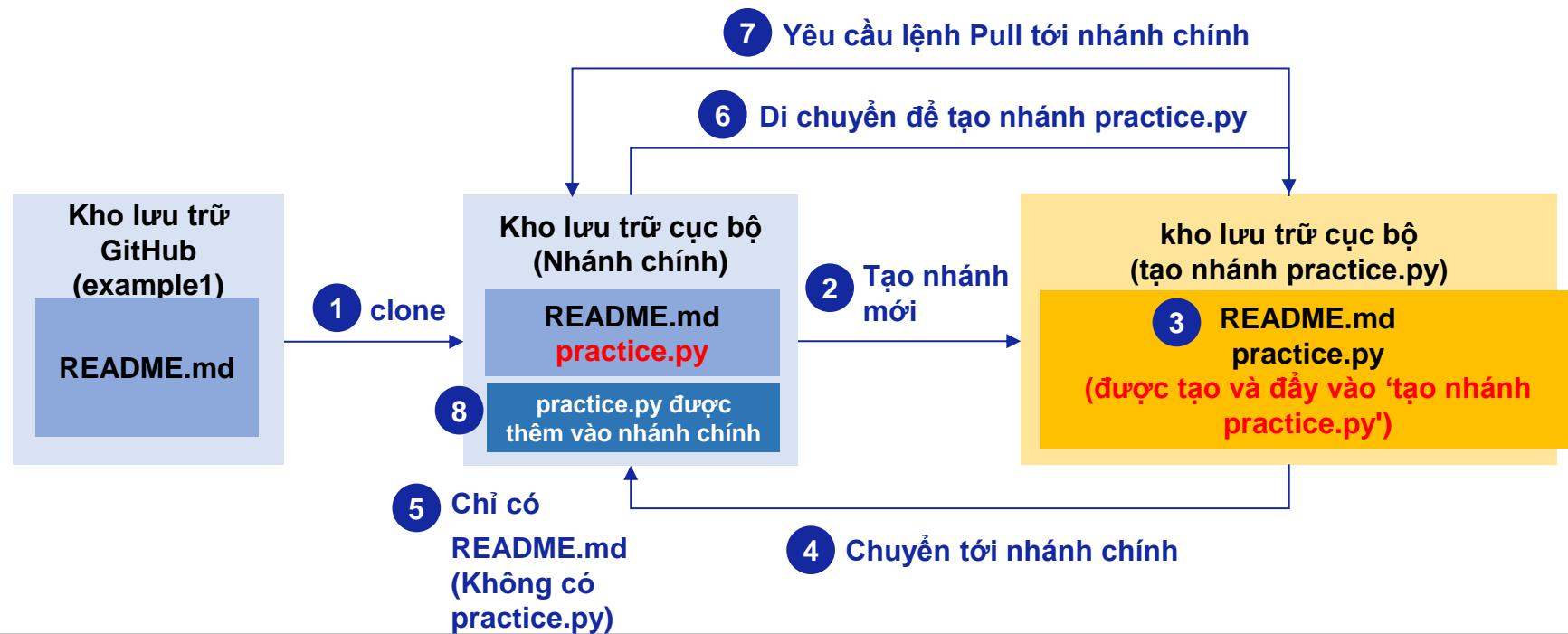


A screenshot of the GitHub Desktop application interface. On the left, there is a dark sidebar with a dropdown menu. The main area shows a file named "example1" with the content "## edited!". The word "example1" is bolded, indicating it is the current file being viewed.

Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

- Sử dụng GitHub Desktop và Nhánh (Branch) trong môi trường Linux.
- Sử dụng Nhánh để làm việc độc lập và quy trình sáp nhập vào Nhánh chính.
- Sơ đồ bên dưới cho thấy luồng xử lý thực tế liên quan đến việc sử dụng Nhánh.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

- Truy cập <https://github.com/shiftkey/desktop> để cài đặt GitHub Desktop trong môi trường Linux.
- Trong dòng chữ "To setup the package repository, run these commands", hãy nhập 3 dòng lệnh vào Linux Terminal.

The screenshot shows a terminal window with the following content:

Debian/Ubuntu distributions

To setup the package repository, run these commands:

```
$ wget -qO - https://packagecloud.io/shiftkey/desktop/gpgkey | sudo tee /etc/apt/trusted.gpg.d/shiftkey.asc > /dev/null
$ sudo sh -c 'echo "deb [arch=amd64] https://packagecloud.io/shiftkey/desktop/any/ any main" > /etc/apt/sources.list.d/packagecloud-shiftkey-desktop.list'
$ sudo apt-get update
```

Then install GitHub Desktop:

```
$ sudo apt install github-desktop
```

Terminal history (scrollable):

```
siciot2021@siciot2021-VirtualBox:~$ wget -qO - https://packagecloud.io/shiftkey/desktop/gpgkey | sudo tee /etc/apt/trusted.gpg.d/shiftkey-des
ktop.asc > /dev/null
siciot2021@siciot2021-VirtualBox:~$ sudo sh -c 'echo "deb [arch=amd64] https://packagecloud.io/shiftkey/desktop/any/ any main" > /etc/apt/sou
rces.list.d/packagecloud-shiftkey-desktop.list'
siciot2021@siciot2021-VirtualBox:~$ sudo apt-get update
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu focal InRelease
Get:4 http://kr.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:1 https://packagecloud.io/shiftkey/desktop/any any InRelease
Get:5 http://kr.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:6 http://kr.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [282 kB]
Get:7 http://kr.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [351 kB]
Get:8 http://kr.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-11 Metadata [944 B]
Get:9 http://kr.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-11 Metadata [10.3 kB]
Fetched 859 kB in 5s (160 kB/s)
Reading package lists... Done
```

Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

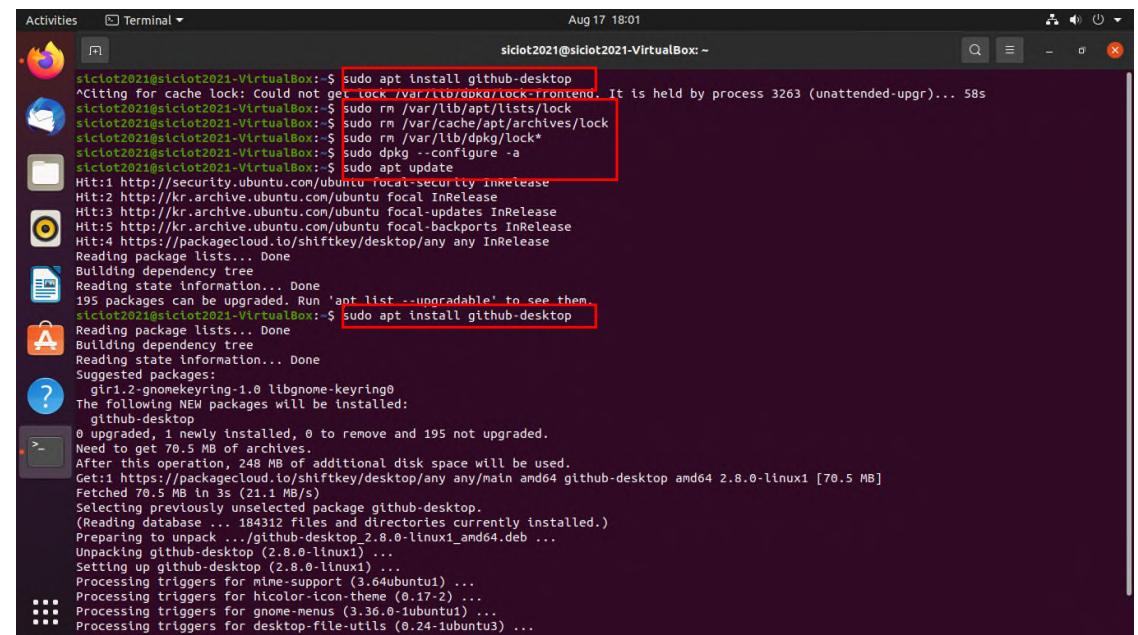
```
$ sudo apt install github-desktop
```

- Khi thực hiện lệnh trên, có thể sẽ không tiến hành được do sự cố liên quan đến bộ nhớ cache. Trong trường hợp này, nhấn (Ctrl + c) để dừng và nhập các lệnh bên dưới để khắc phục sự cố.

```
$ sudo rm /var/lib/apt/lists/lock  
$ sudo rm /var/cache/apt/archives/lock  
$ sudo rm /var/lib/dpkg/lock*
```

```
# Package Update  
$ sudo dpkg --configure -a  
$ sudo apt update
```

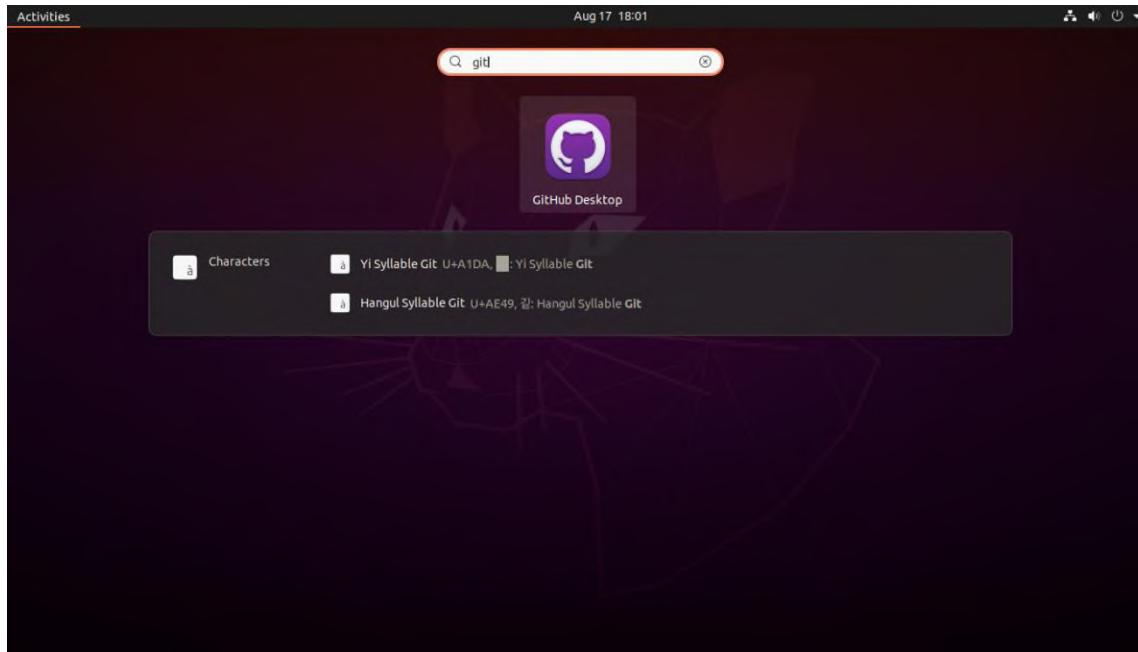
```
# Retry Install  
$ sudo apt install github-desktop
```



Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

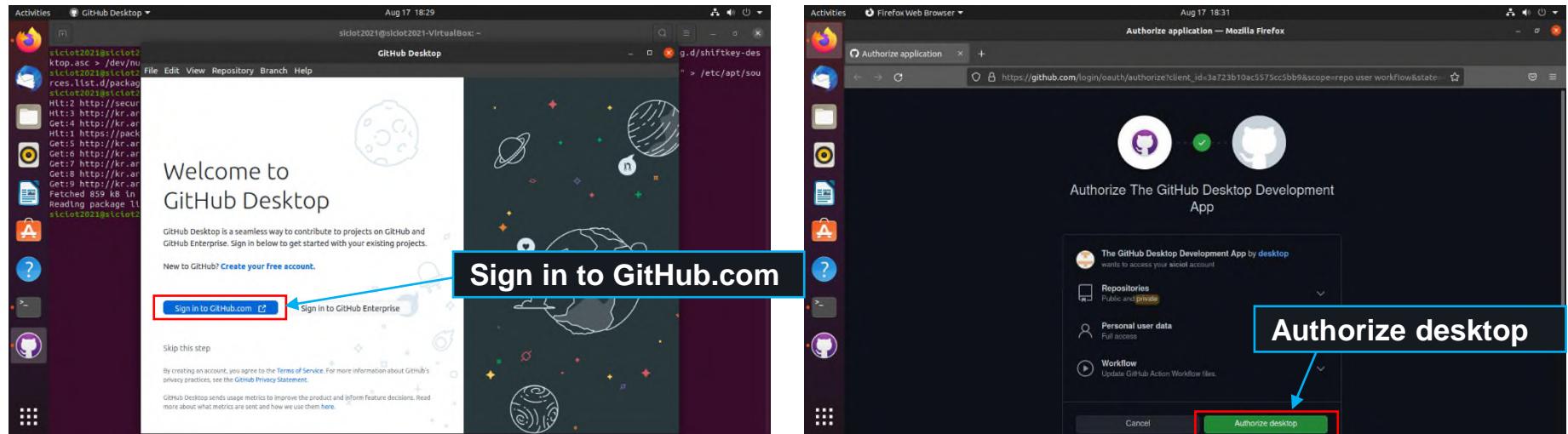
- Sau khi cài đặt thành công, bạn có thể tìm thấy GitHub Desktop. Sau khi chạy, cách sử dụng cũng tương tự với Windows.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

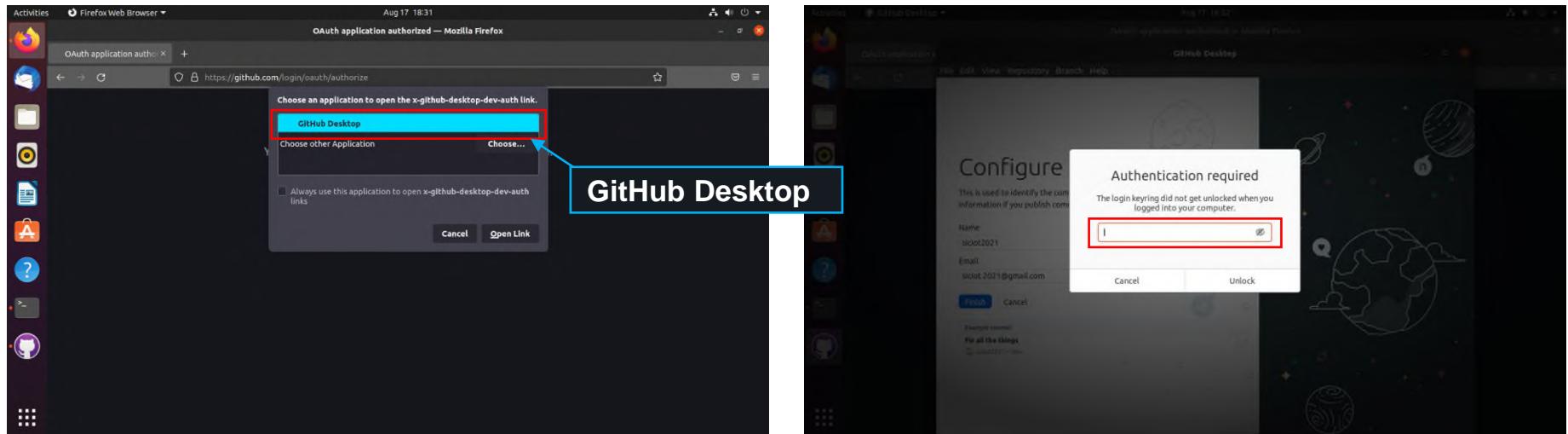
- Nhấp vào biểu tượng **Sign in to GitHub.com** và đăng nhập bằng tài khoản GitHub của bạn.
- Nhập thông tin đăng nhập → Nhấp vào nút **Authorize desktop**.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

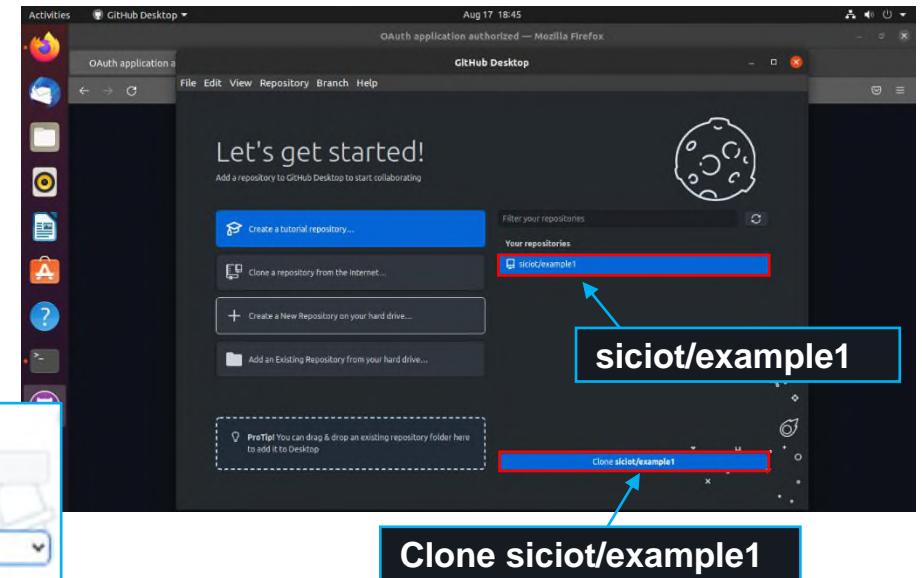
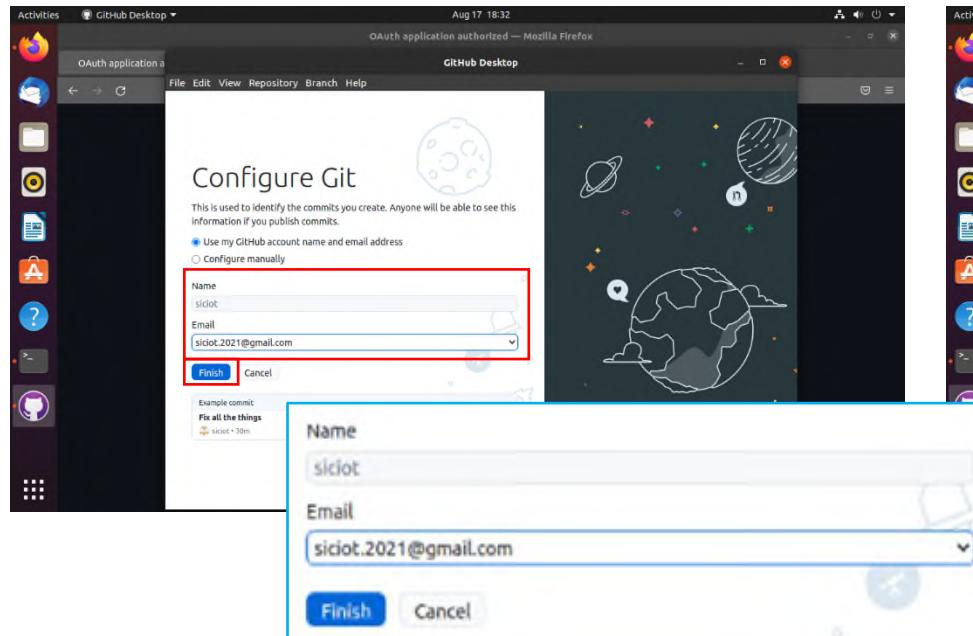
- Nhấp đúp vào GitHub Desktop để chọn.
- Nhập mật khẩu của tài khoản Linux cho câu hỏi xác thực.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Cài đặt GitHub Desktop trên Linux

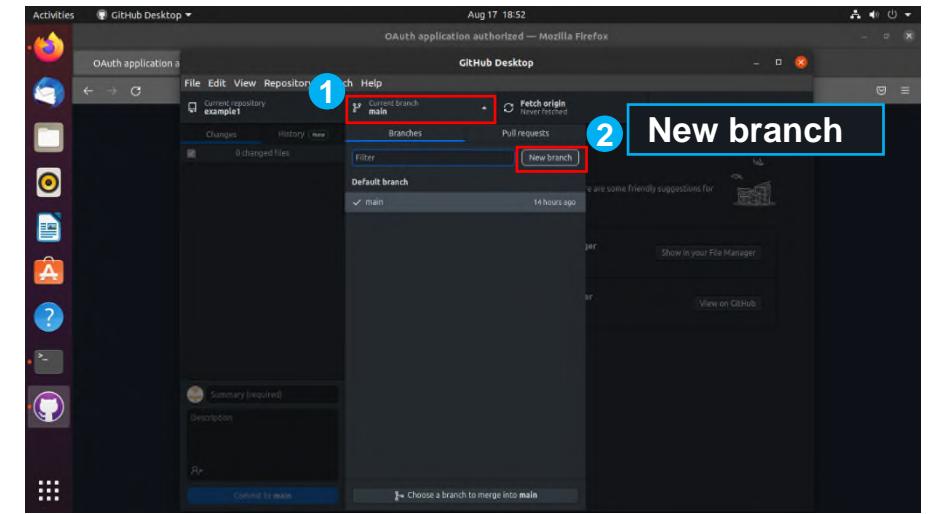
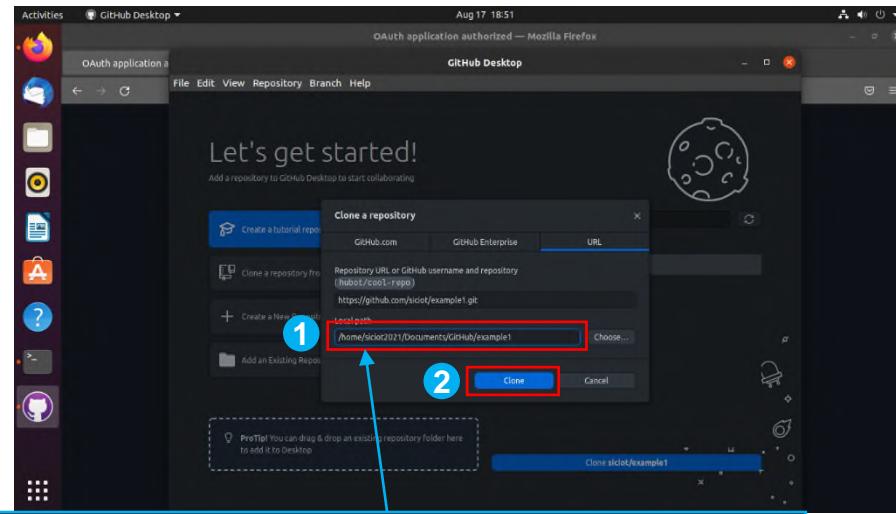
- Đăng ký tên và địa chỉ e-mail cần đăng ký khi xác nhận. Sau khi cài đặt, bạn có thể thấy màn hình của GitHub Desktop trong môi trường Linux.
- Chọn kho lưu trữ example1 được tạo làm ví dụ, sao chép nó và nhập nó.



Sử dụng Nhánh trong GitHub Desktop (Linux)

I Sử dụng Nhánh trong GitHub Desktop trên Linux

- Giả sử rằng trạng thái hiện tại của dự án Clone là nhánh hoạt động phục vụ cho một tác vụ vận hành ổn định. Nếu bạn muốn làm việc mà không ảnh hưởng đến dự án này, bạn có thể tạo một Nhánh mới để làm việc.
- Kiểm tra **Local Path** và nhấp vào biểu tượng **clone**.
- Đi tới Nhánh hiện tại để rời khỏi Nhánh chính hiện tại và tạo Nhánh mới → Nhấp chọn **New branch**

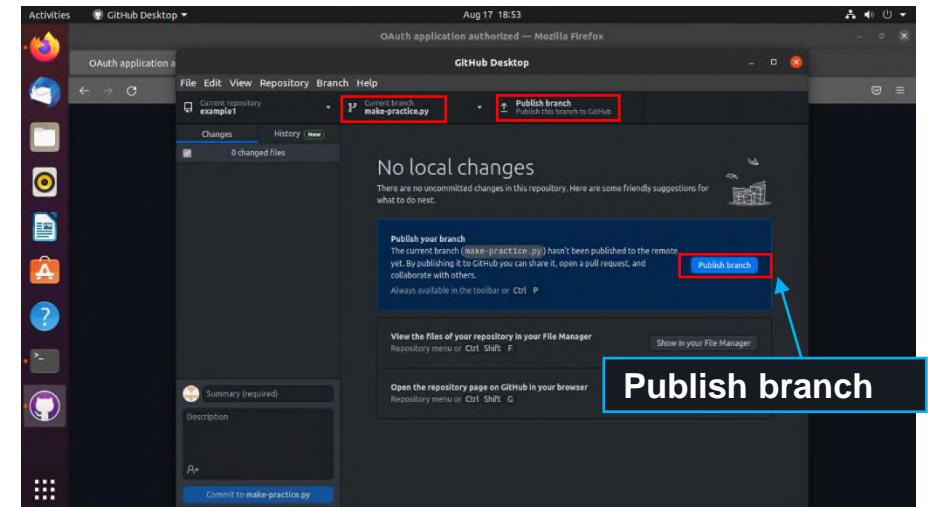
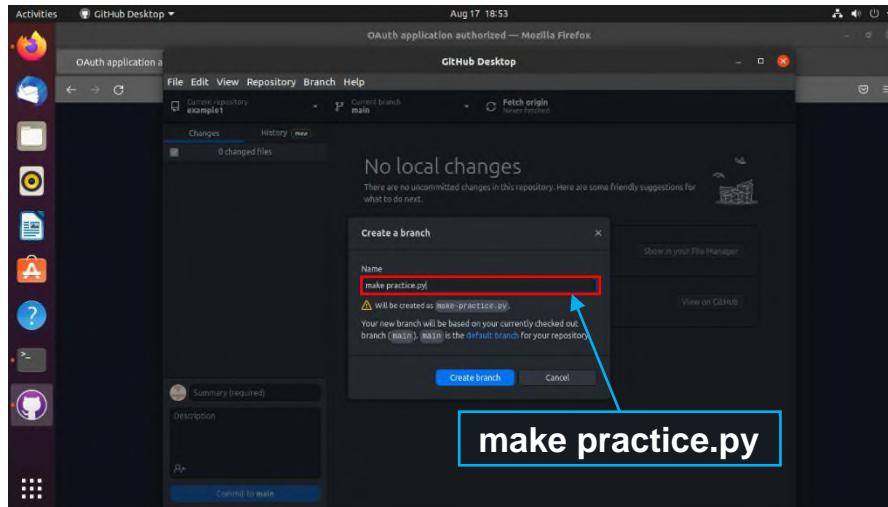


\home \sictiot2021\Documents\GitHub\example1

Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

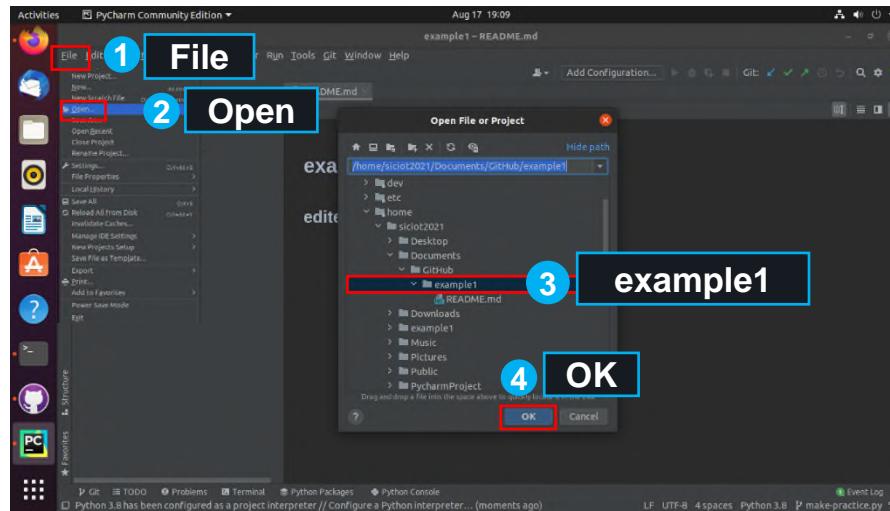
- Tạo một nhánh có tên là **make practice.py**. Nó là một nhánh để tạo một tệp Python mới có tên `practice.py`.
- Kiểm tra xem Nhánh hiện tại đã được thay đổi thành **make practice.py** chưa. Nhấp vào nút **Publish branch** để tạo nhánh do Kho lưu trữ từ xa chưa xác định branch mới tạo.



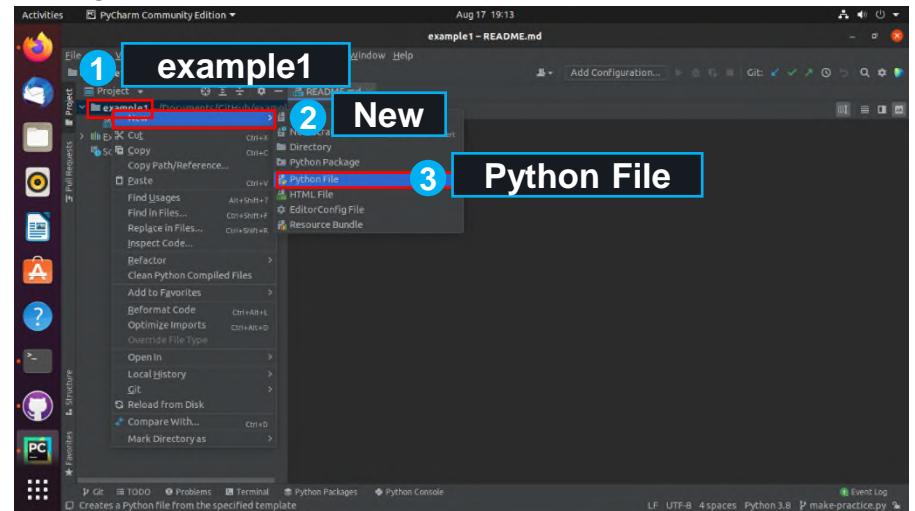
Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

- Chạy Pycharm để tạo file practice.py trong Local Path.
- File → Open → Select directory example1 → OK



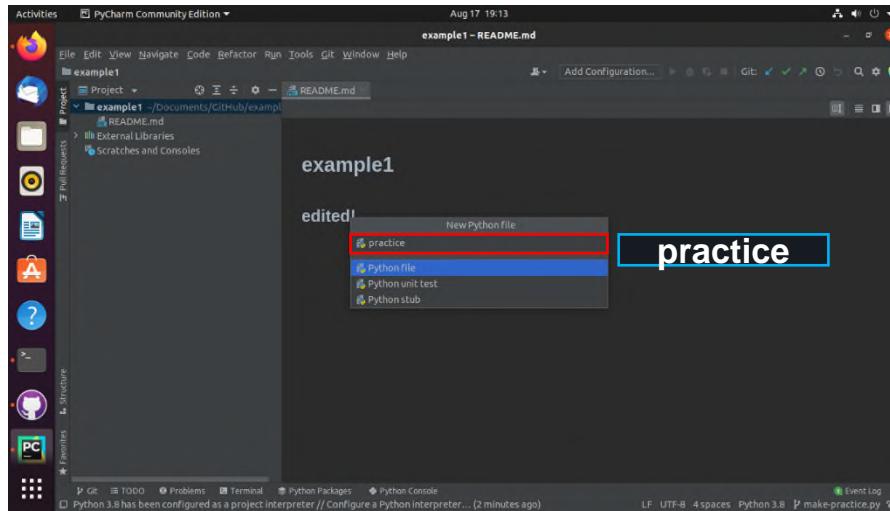
- Bấm chuột phải vào example1 → New → Click Python File



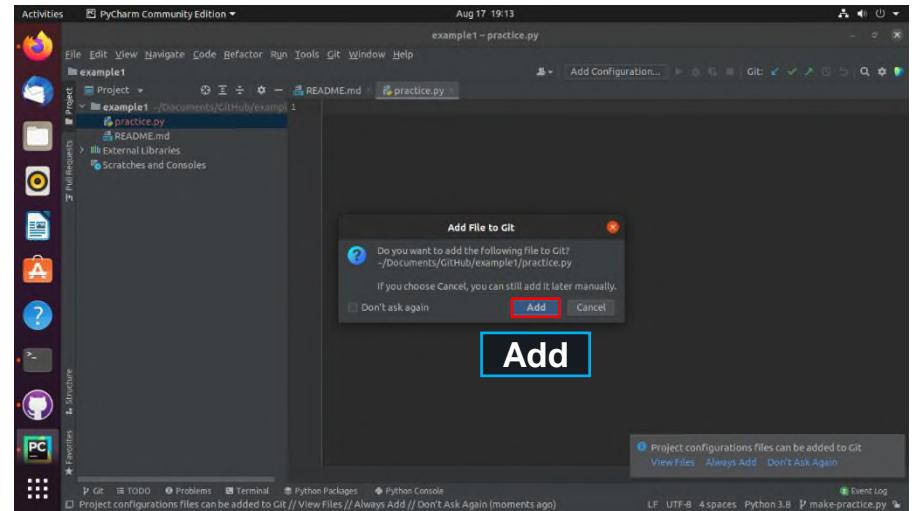
Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

- Nhập practice rồi bấm phím enter. Bạn có thể thấy rằng tệp practice.py đã được tạo thành công.
- Một cửa sổ bật lên hỏi bạn có muốn git theo dõi file này không.



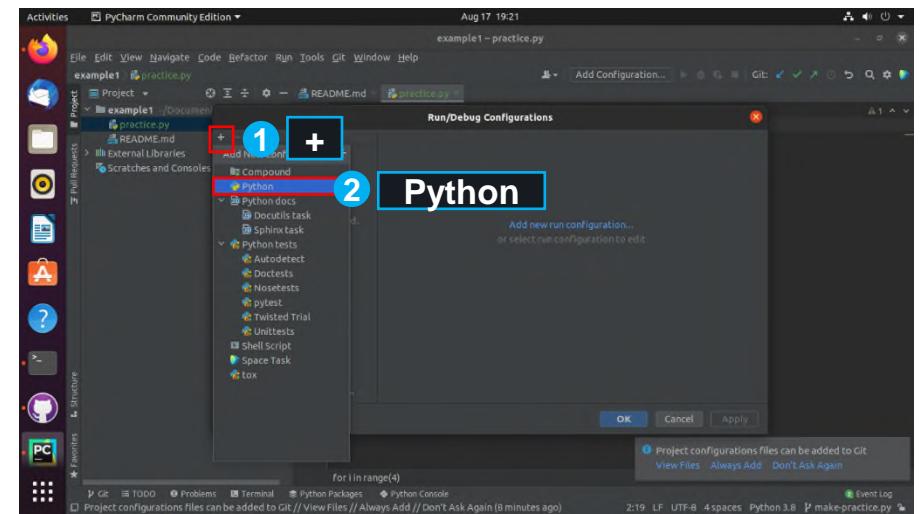
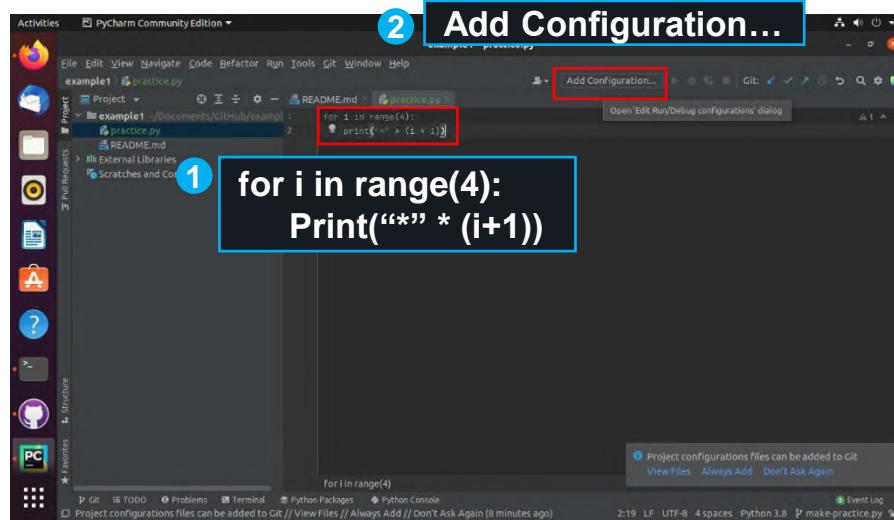
- Nhấp chọn Add để đưa file vào dõi theo dõi ngay lập tức (Không phải thực hiện git add practice.py).
- Nếu bạn không muốn theo dõi, hãy nhấp vào Cancel. Nhấp vào Add for this practice.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

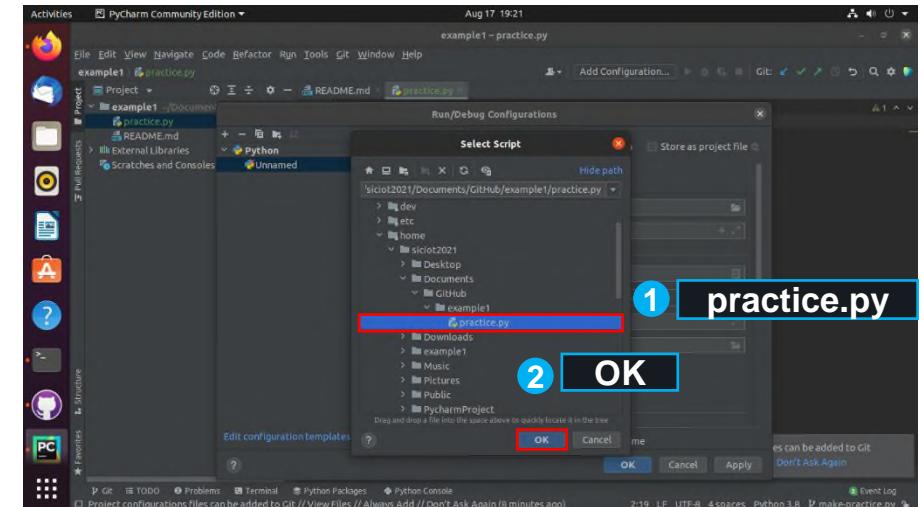
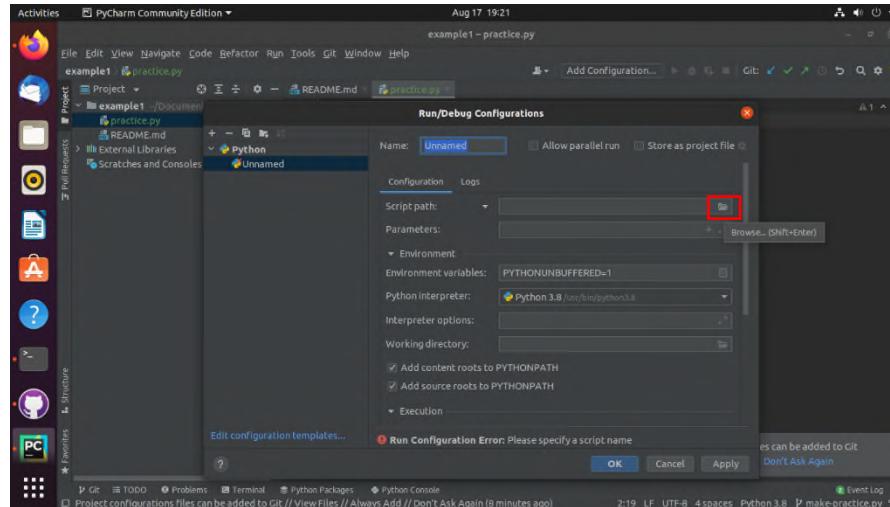
- Viết mã để tạo một chương trình đơn giản trong practice.py. Nó xuất ra "*" trên 4 dòng bằng cách tăng tuần tự.
- Nhấp chuột chọn Add Configuration vì chưa được cấu hình để chạy.
- Nhấp vào biểu tượng + → Chọn Python



Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

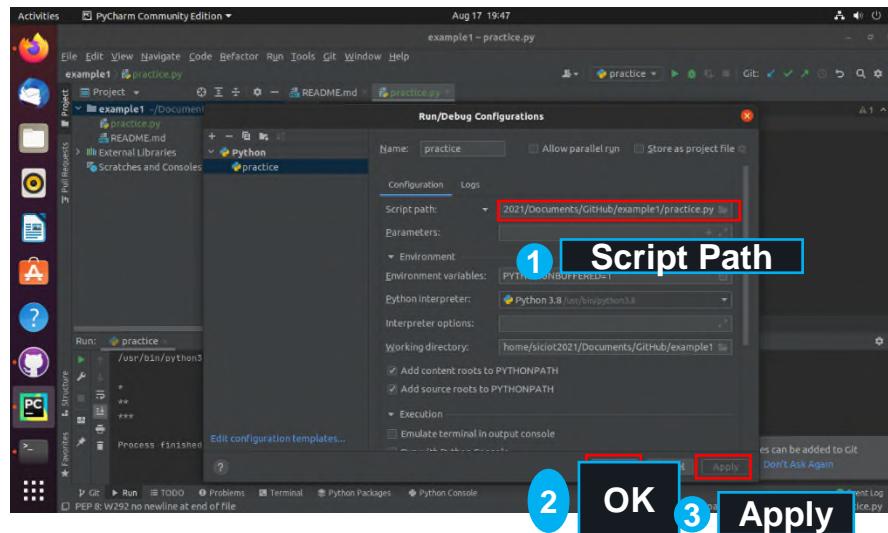
- Nhấp **Browse** (biểu tượng thư mục) để thêm đường dẫn tập lệnh.
- Nhấp vào file practice.py trong my Local Path → OK



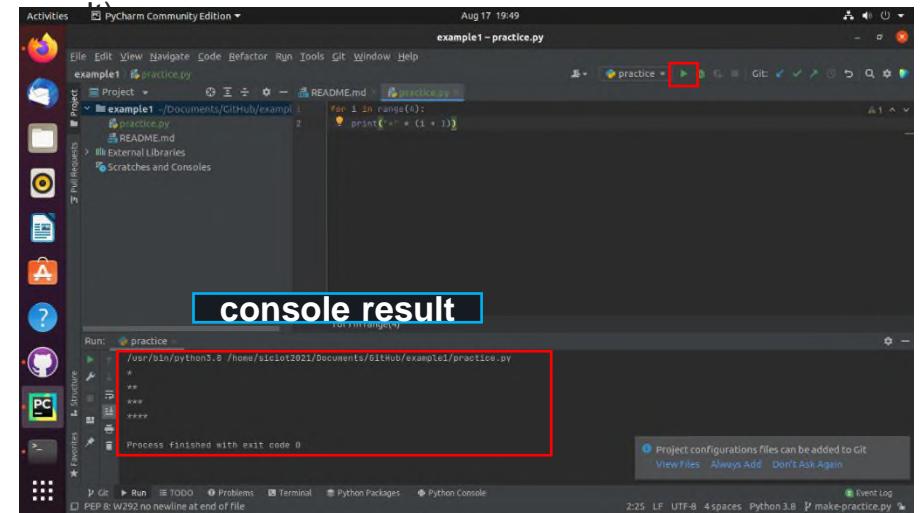
Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

- Kiểm tra phần mới thêm vào Script Path → Apply → OK



- Chạy với cấu hình đã cài đặt
→ Kiểm tra kết quả qua bảng điều khiển (console)



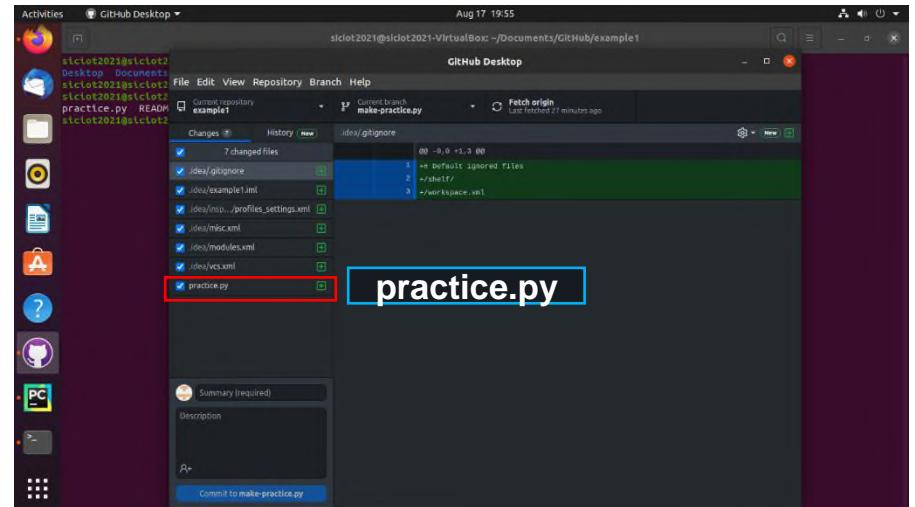
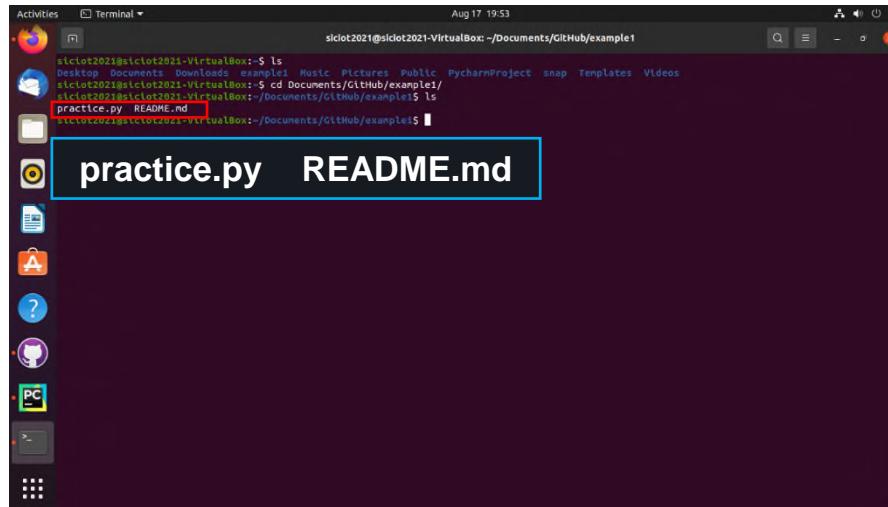
Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

- ▶ Truy cập tới Local Path và kiểm tra lại.

```
$ cd {local path}  
$ ls
```

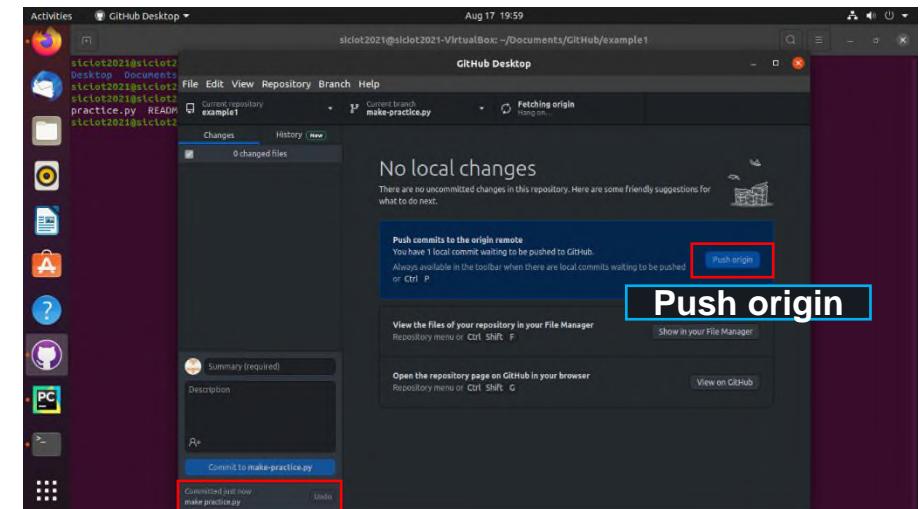
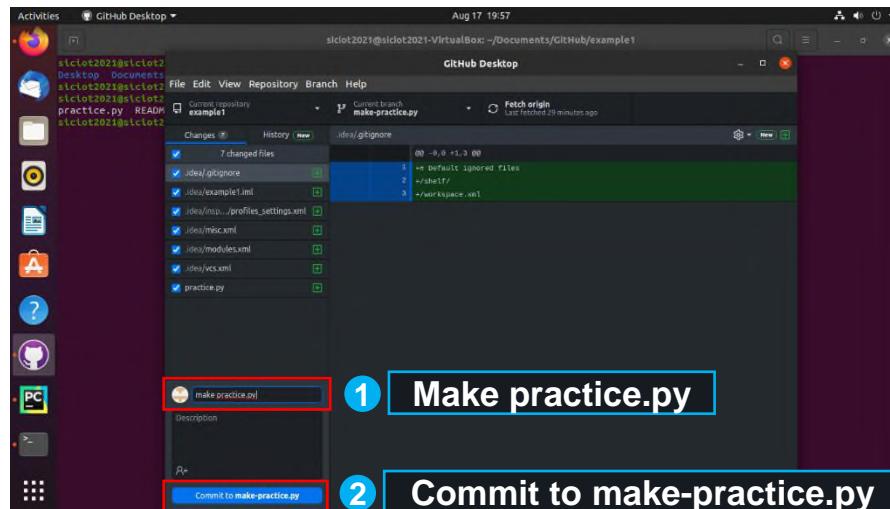
- ▶ Kiểm tra GitHub Desktop, bạn có thể thấy rằng một số file bao gồm cả practice.py đã được thêm vào nhánh make practice.py. Có thể thấy git hỗ trợ rất tốt trong việc quản lý các file ngoài file practice.py.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

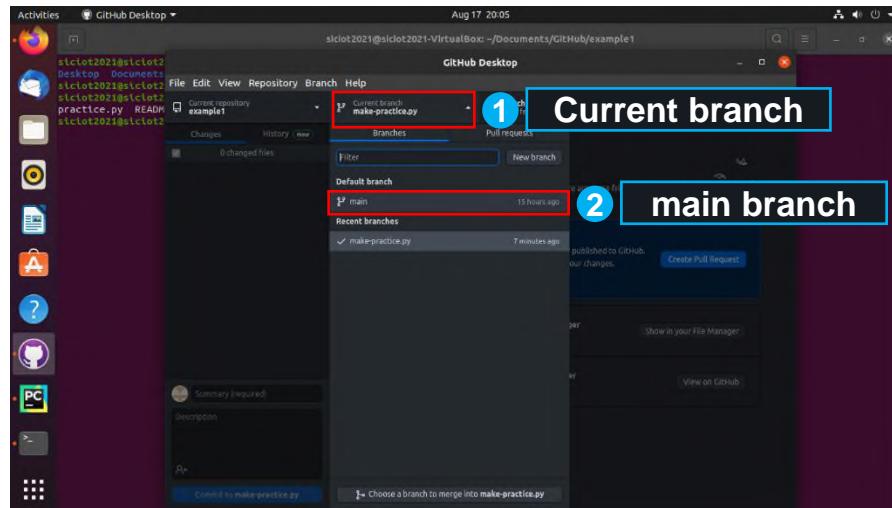
- Xác nhận nhánh make practice.py với thông điệp xác nhận "make practice.py".
- Nhấp chọn **Push origin** để đẩy practice.py vào nhánh make practice.py.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Sử dụng Nhánh trong GitHub Desktop trên Linux

- Tới thẻ current branch
→ Nhập vào main để quay về nhánh chính
- Kiểm tra danh sách thông qua lệnh ls trong Local Path. Bạn có thể thấy practice.py đã biến mất.



```
$ ls
sictot2021@sictot2021-VirtualBox: ~/Documents/GitHub/example1
sictot2021@sictot2021-VirtualBox: $ cd Documents/GitHub/example1/
sictot2021@sictot2021-VirtualBox: ~/Documents/GitHub/example1$ ls
practice.py README.md
sictot2021@sictot2021-VirtualBox: ~/Documents/GitHub/example1$ ls
README.md
```

The terminal window shows the directory structure of the GitHub repository. It starts with a command to list files in the current directory, followed by changing to the repository directory, and then listing files again. The first listing shows 'practice.py' and 'README.md', while the second listing shows only 'README.md', indicating that 'practice.py' has been deleted.

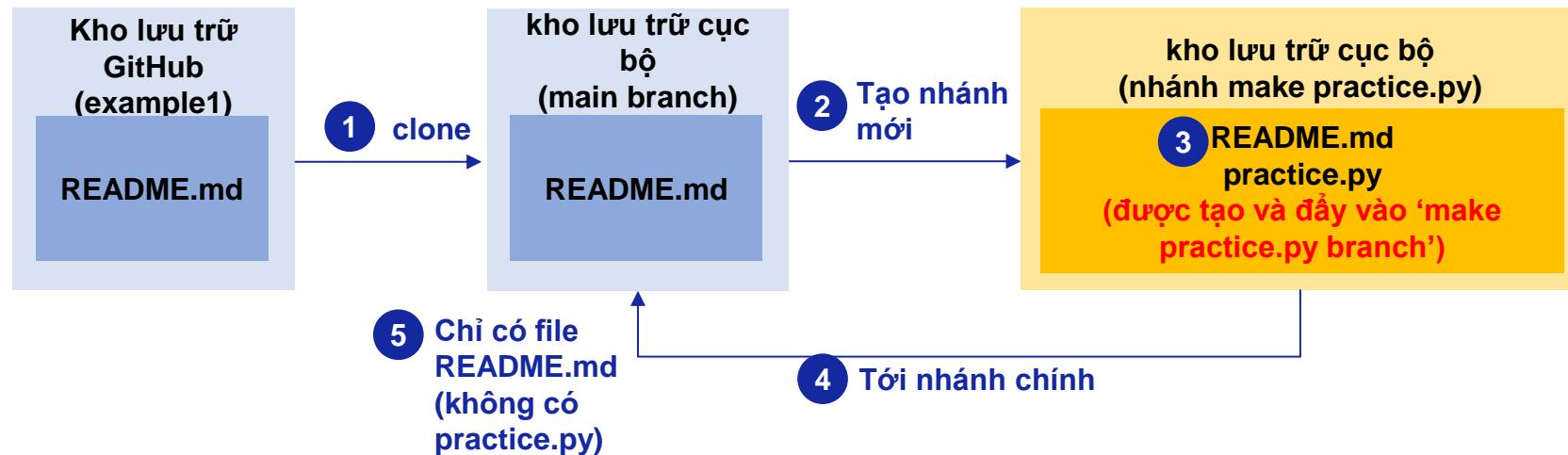
Sử dụng Nhánh trong GitHub Desktop (Linux)

| Sử dụng Nhánh trong GitHub Desktop trên Linux (1/2)

- Quá trình thực hành như sau:

Nạp dự án trong Kho lưu trữ GitHub hiện có vào Kho lưu trữ cục bộ Linux bằng cách tạo bản sao.
Tạo một nhánh mới có tên make practice.py để chạy thử nghiệm độ ổn định của thao tác trước khi chỉnh sửa.

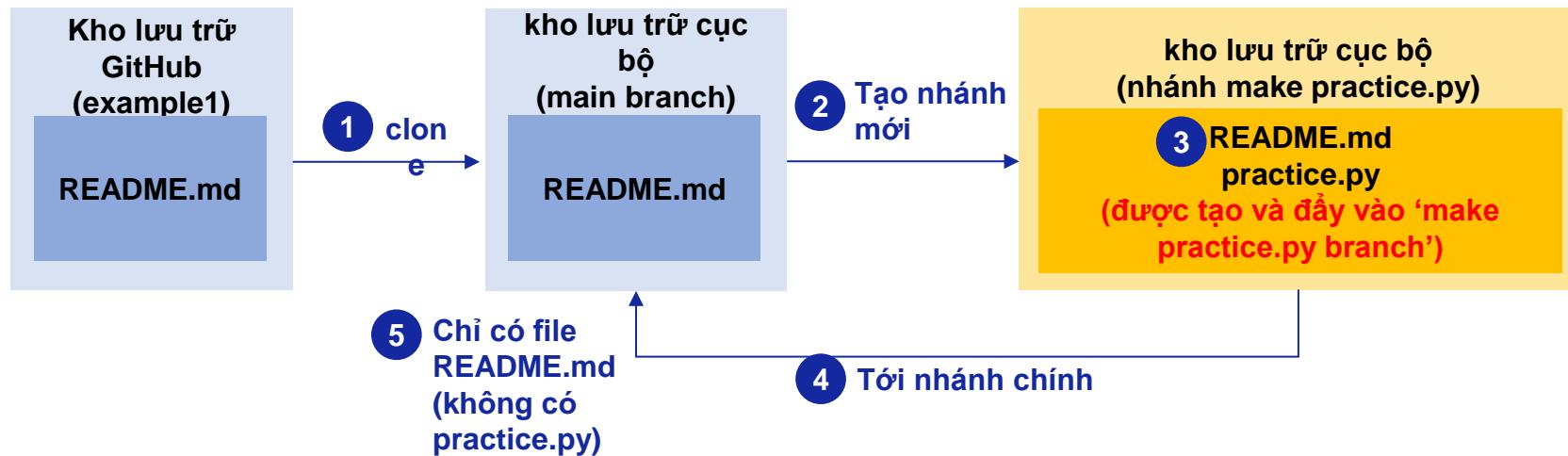
Tạo một tệp python trong nhánh make practice.py. Lưu nó rồi tiến hành quá trình Commit và Push.



Sử dụng Nhánh trong GitHub Desktop (Linux)

I Sử dụng Nhánh trong GitHub Desktop trên Linux (2/2)

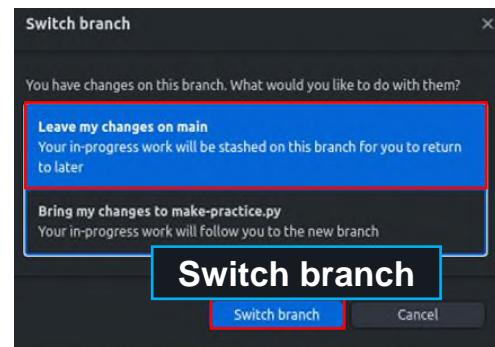
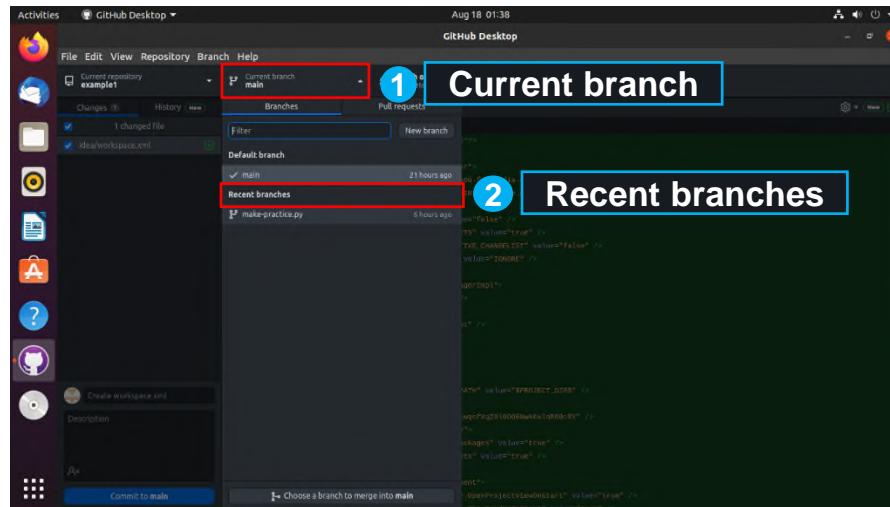
- Không có thay đổi trong nhánh chính.
- Điều này là do chúng ta đã thao tác trong nhánh make practice.py mới, là một không gian làm việc độc lập.
- Khi thực hiện công việc cá nhân hoặc tập thể, đều có thể chạy thử nghiệm trong khi làm việc độc lập mà không ảnh hưởng đến tính ổn định của dự án. Nếu kết quả thử nghiệm không ổn định, nó sẽ không được tích hợp vào nhánh chính. Khi bạn quay trở lại nhánh chính, nội dung của dự án cũng trở về trạng thái ban đầu.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

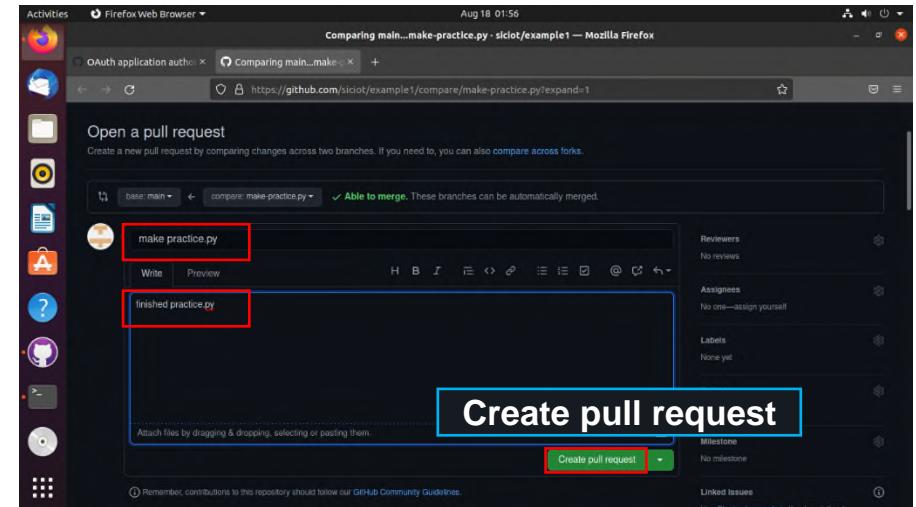
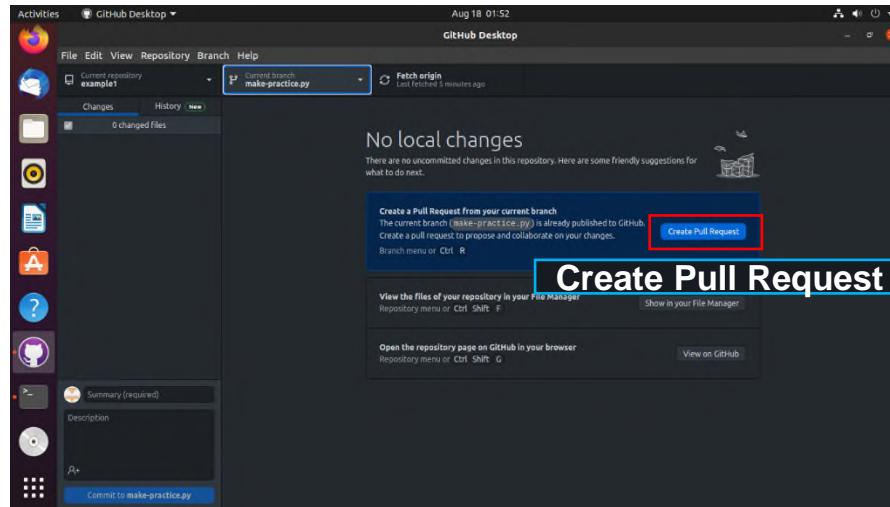
- Sau khi chạy đủ các thử nghiệm xem xét các vấn đề có thể xảy ra trong việc tạo file practice.py không, ta hợp nhất kết quả vào nhánh chính.
 - Tới current branch để di chuyển từ nhánh main tới nhánh make practice.py → current branch
- **Leave my changes on main** có nghĩa là nội dung của nhánh chính sẽ không được ánh xạ trong nhánh practice.py. Mặt khác, **Bring my changes to make practice.py** có nghĩa là nội dung của nhánh chính được ánh xạ trong nhánh make practice.py.
- Chọn **Leave my changes on the main** và nhấp vào biểu tượng **switch branch**, vì chúng ta cần giữ nguyên nội dung của nhánh chính.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

- Chọn **Create Pull Request** để hợp nhất các thay đổi vào nhánh chính.
- Di chuyển đến trang Open a pull request.
- Viết tin nhắn yêu cầu pull và nhấp vào nút **Create pull request button**.

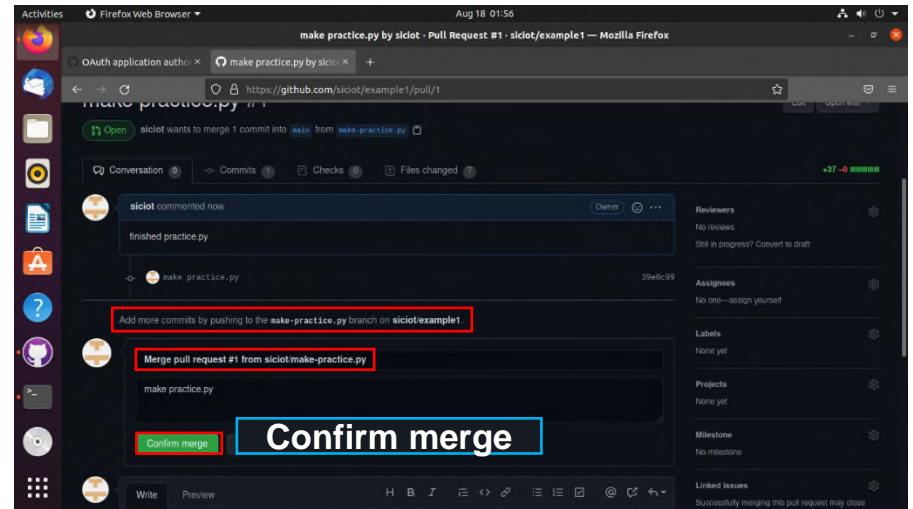
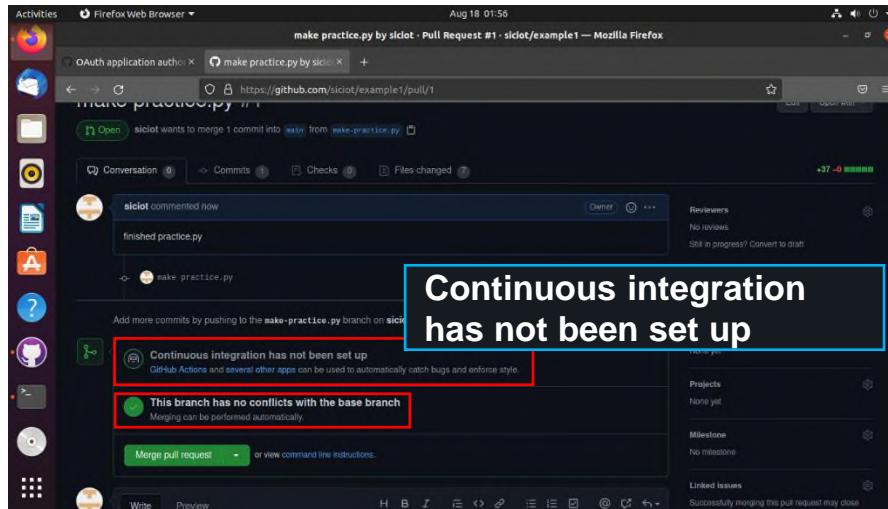


Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

- ▶ Kiểm tra tin nhắn “This branch has no conflicts with the base branch”
- ▶ Nhấp vào biểu tượng **Merge pull request** để hợp nhất nhánh make practice.py đã được nhận pull request vào main.

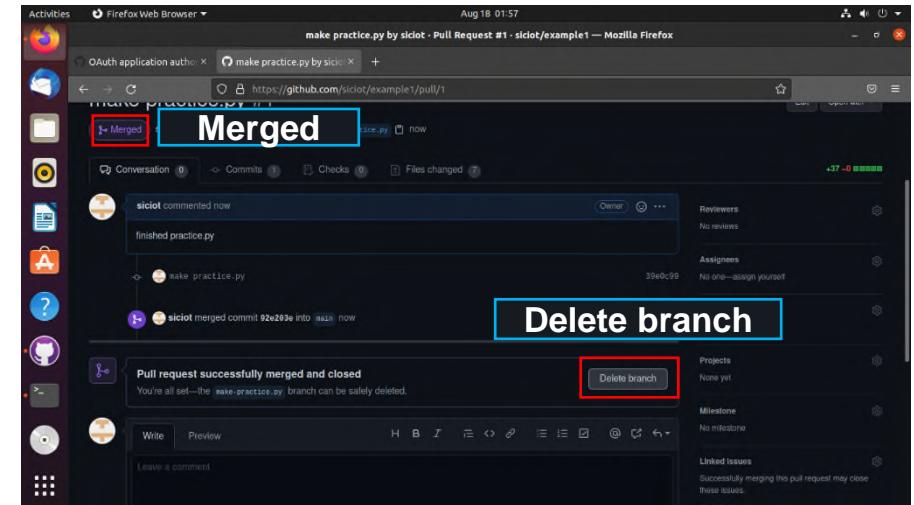
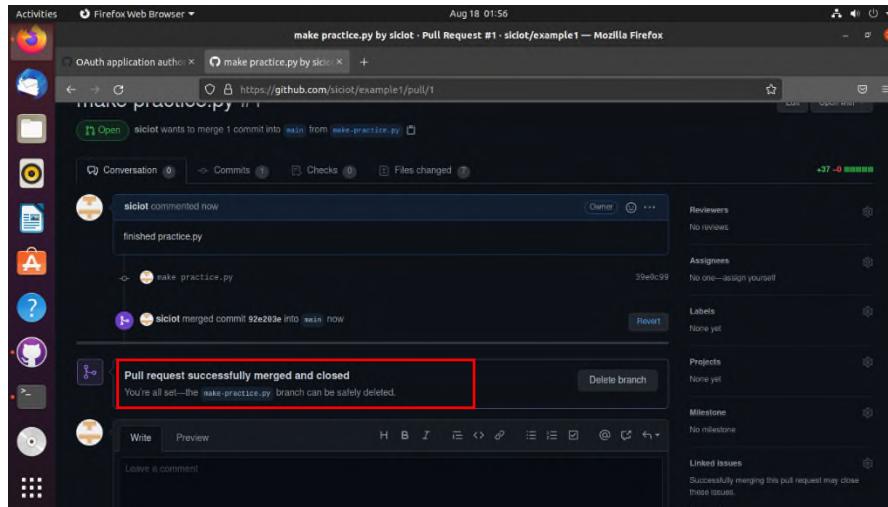
- ▶ Kiểm tra commit message để xác nhận các nội dung đã hợp nhất và nhấn vào **Confirm merge**.



Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

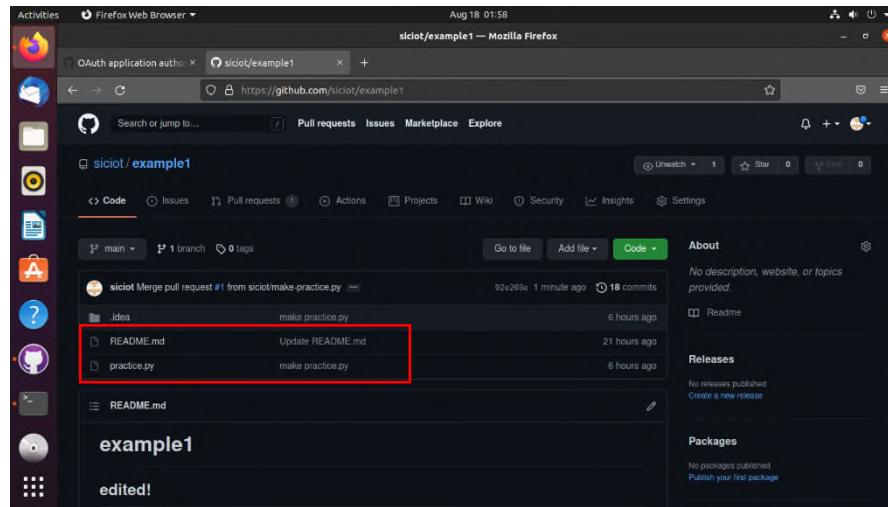
- Kiểm tra tin nhắn “Pull request successfully merged and closed”.
- Nhấp vào biểu tượng **Delete branch** ở bên phải để xóa nhánh `make.practice.py` đã hoàn thiện.
- Kiểm tra phần trên cùng bên trái để xem trạng thái đã thay đổi thành **Merged**.



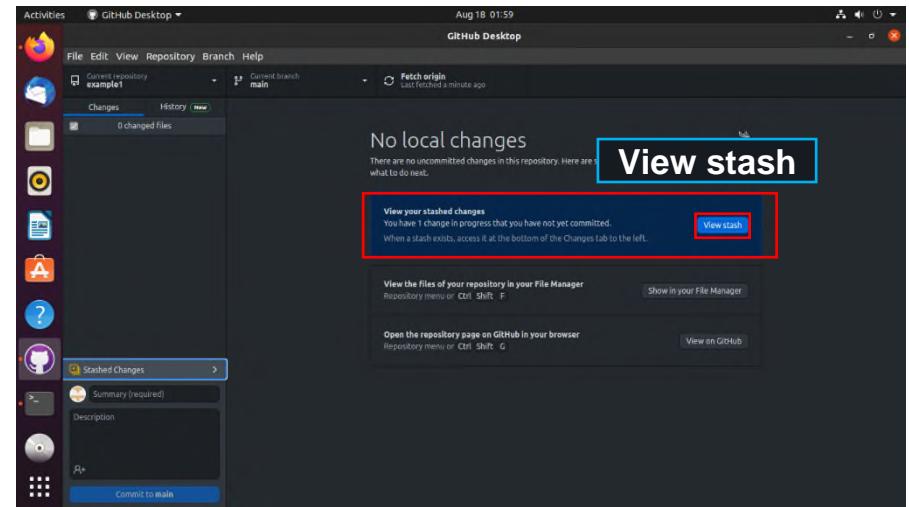
Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

- Quay trở lại Kho lưu trữ GitHub và kiểm tra xem practice.py có tồn tại hay không, mặc dù nó hiện đang nằm trong nhánh main.



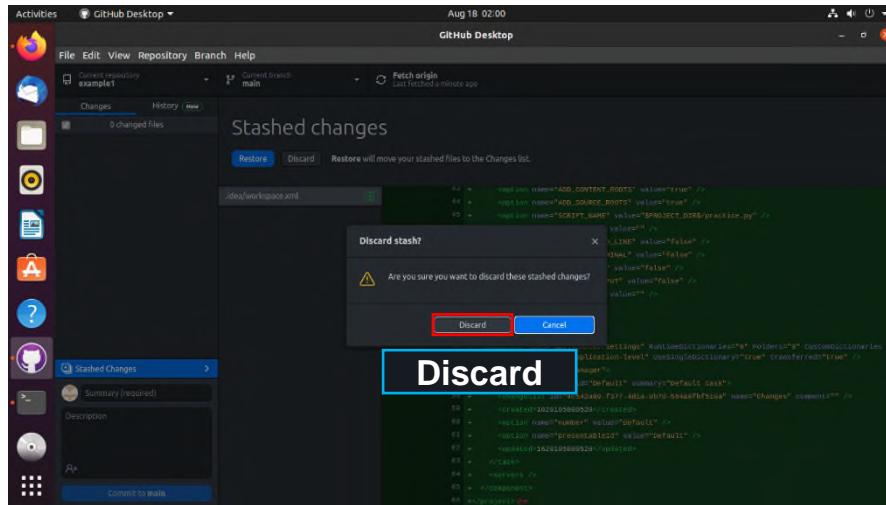
- Quay lại GitHub Desktop và kiểm tra nhánh chính. Bạn có thể thấy menu được gọi là View your stashed changes đang hoạt động. Nhấp vào biểu tượng **View stash**.



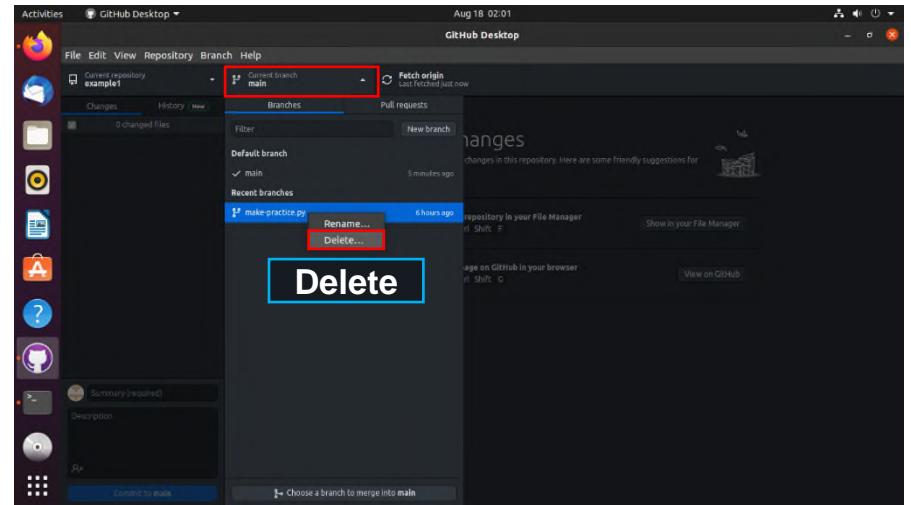
Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

- Chúng ta có thể tạm thời lưu các thay đổi mà không cần xác nhận chúng với một nhánh bằng cách lưu các thay đổi. Trong trường hợp này, chúng ta không cần phải lưu vùng nhớ stash. Và để loại bỏ chúng, nhấp vào biểu tượng **Discard**.



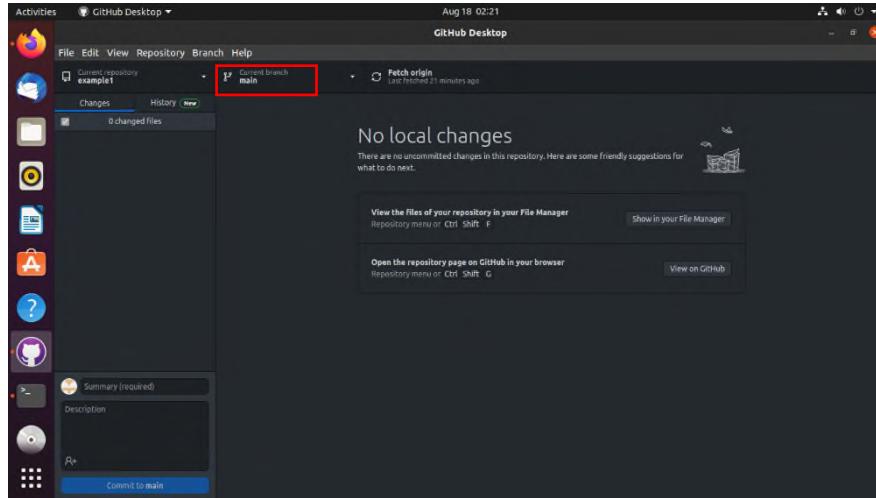
- Xóa nhánh make-practice.py vì bạn vẫn có thể thấy nó trên GitHub Desktop.
- Current branch → Bấm chuột phải make practice.py → Delete



Sử dụng Nhánh trong GitHub Desktop (Linux)

Gây tác động tới một nhánh mới

- ▶ Xóa nhánh make practice.py và quay lại nhánh chính. Cuối cùng, kiểm tra Kho lưu trữ cục bộ.
- ▶ Khi bạn di chuyển đến Local Path nơi example1 được lưu, bạn có thể thấy rằng practice.py và README.md tồn tại.
- ▶ Bạn cũng có thể kiểm tra nội dung của practice.py.



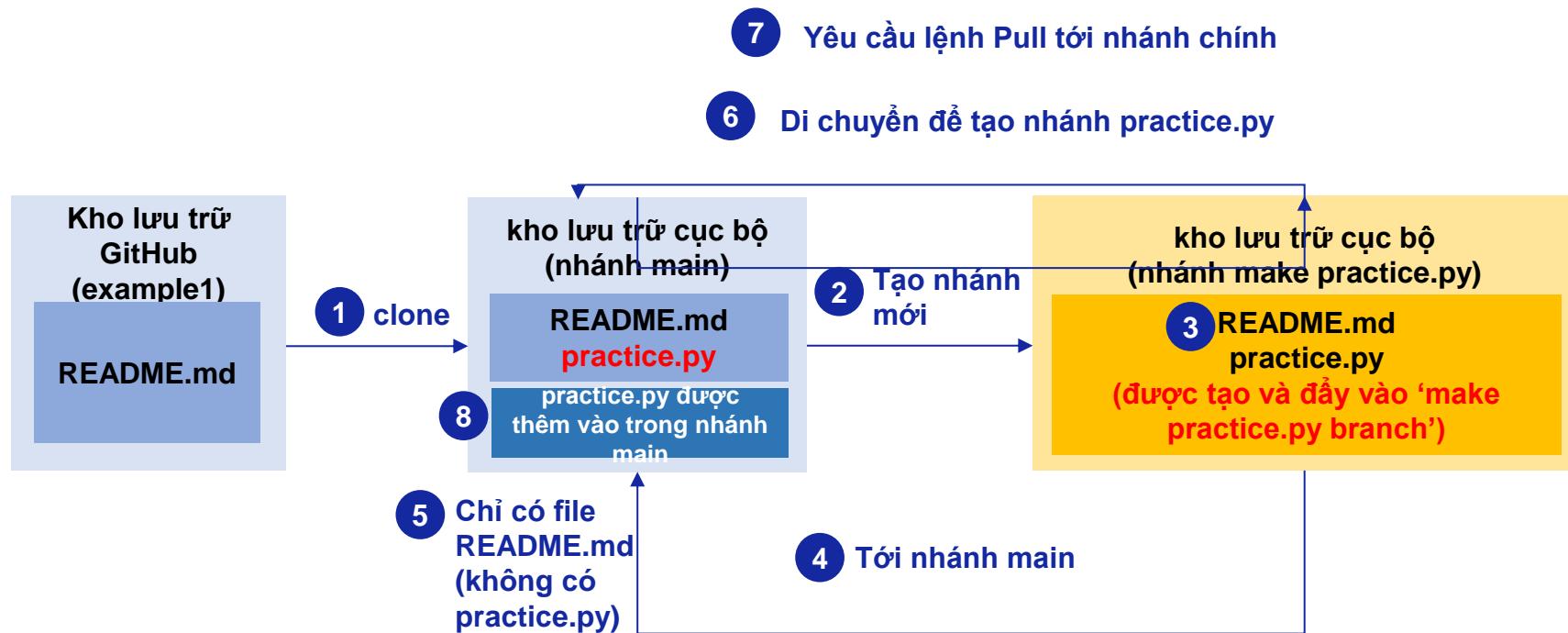
```
$ ls
```

```
$ cat practice.py
```

```
siciot2021@siciot2021-VirtualBox: ~/Documents/GitHub/example1
siciot2021@siciot2021-VirtualBox:~/Documents/GitHub/example1$ ls
practice.py README.md
siciot2021@siciot2021-VirtualBox:~/Documents/GitHub/example1$ cat practice.py
for i in range(4):
    print("*" * (i + 1))siciot2021@siciot2021-VirtualBox:~/Documents/GitHub/example1$
```

Sử dụng Nhánh trong GitHub Desktop (Linux)

| Lưu đồ của quy trình sử dụng Nhánh



Bài 5.

Cộng tác bằng GitHub

| 5.1. Bài thực hành đơn giản

Tổng quan Bài thực hành

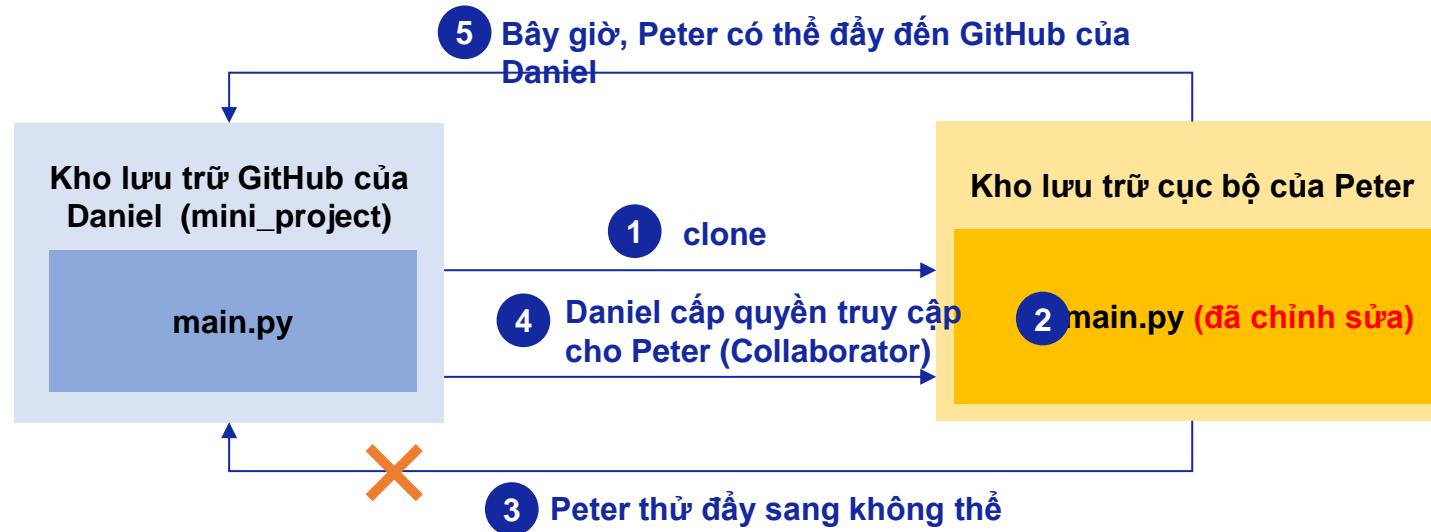
Tổng quan Bài thực hành



- ▶ Cùng thực hiện một Mini-Project về Python theo nhóm gồm Daniel (Project Manager - Quản lý dự án) và Peter (Developer - Lập trình viên).
- ▶ Daniel quản lý Kho lưu trữ trên GitHub và tổng thể dự án. Peter tạo ra một chương trình giải toán đơn giản với vai trò lập trình viên.
- ▶ Daniel sử dụng Window10, Peter sử dụng Linux (Ubuntu) và cả hai cùng sử dụng GitHub Desktop.
- ▶ Hãy cùng tìm hiểu cách thao tác Quản lý phiên bản và quản lý dự án khi làm việc cùng nhau trong một dự án.

Lưu đồ thực hiện Mini-Project

| Minh họa lưu đồ mô tả quá trình cộng tác



Bắt đầu với Mini-Project

Tạo kho lưu trữ dự án (Daniel)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Daniel-projectowner / Repository name * mini_project ✓

Great repository names are short and memorable. Need inspiration? How about [vigilant-octo-pancake](#)?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.
 Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore Choose which files not to track from a list of templates. [Learn more](#).

Choose a license A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

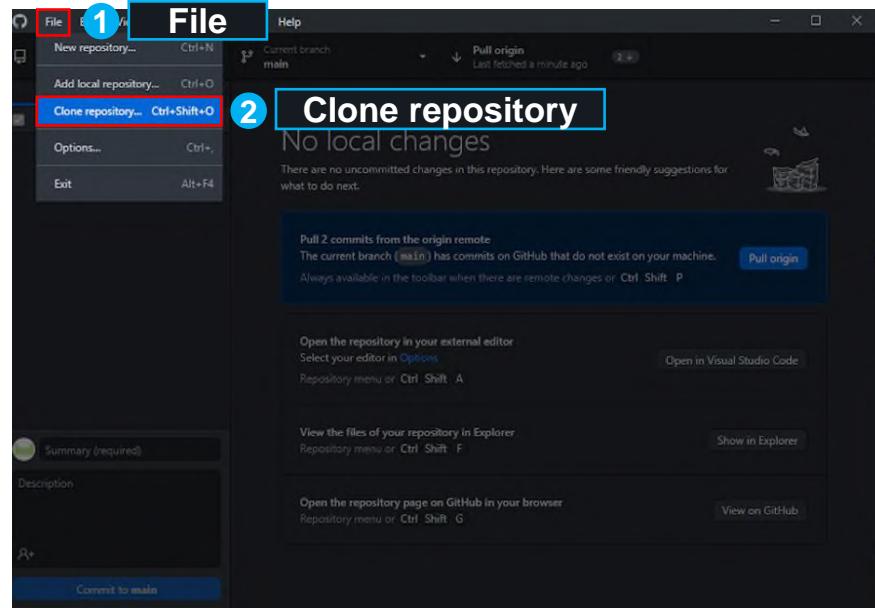
Create repository

- ▶ Daniel tạo một kho lưu trữ để làm việc và quản lý các dự án trên GitHub.
- ▶ Tên của kho lưu trữ được đặt là mini_project rồi nhấp vào **Create repository** để khởi tạo.

Bắt đầu với Mini-Project

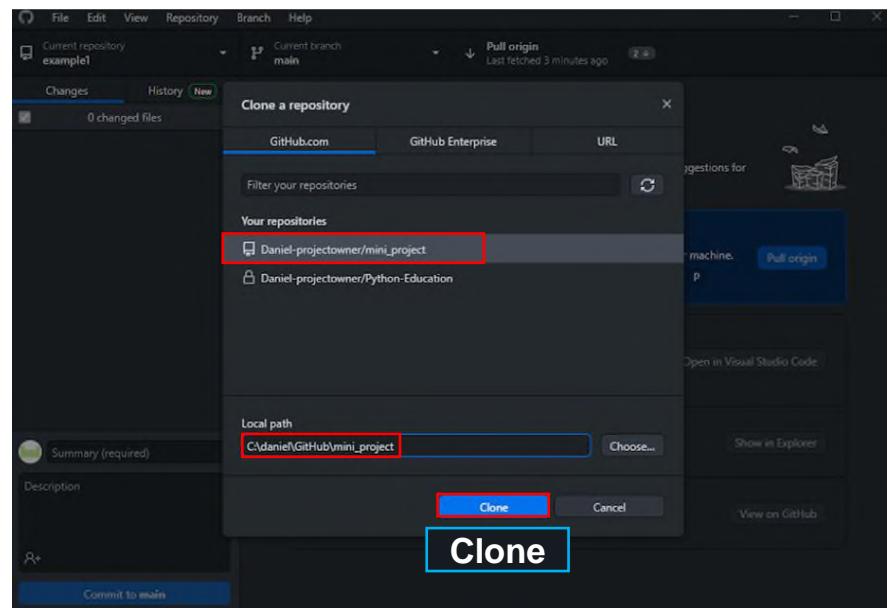
Tạo kho lưu trữ dự án (Daniel)

- Daniel cần sao chép Kho lưu trữ GitHub của mình vào Kho lưu trữ cục bộ.
- File → Nhấp vào Clone repository



- Nhấp vào mini_project Repository trên GitHub Repository.

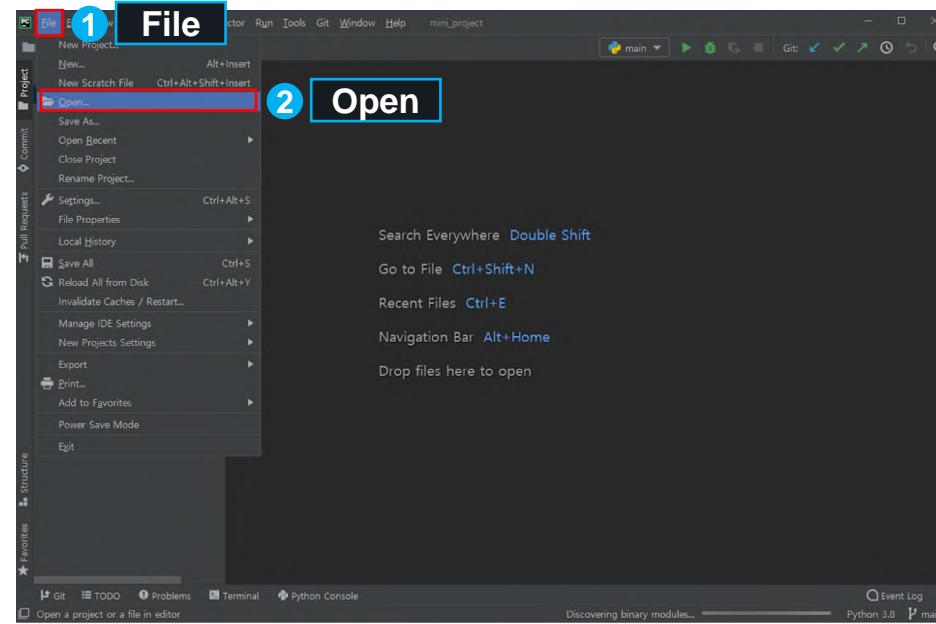
Đặt Local Path rồi nhấp vào nút Clone.



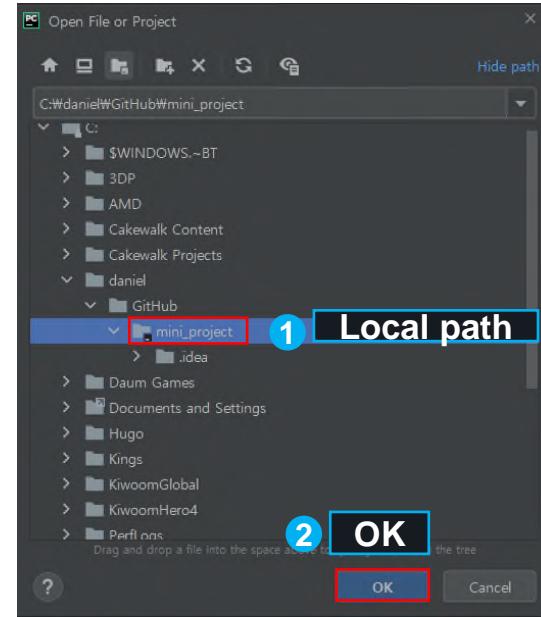
Bắt đầu với Mini-Project

Tạo file main.py (Daniel)

- Mở Pycharm để lập trình trong Kho lưu trữ cục bộ
- Nhấp vào File → Open



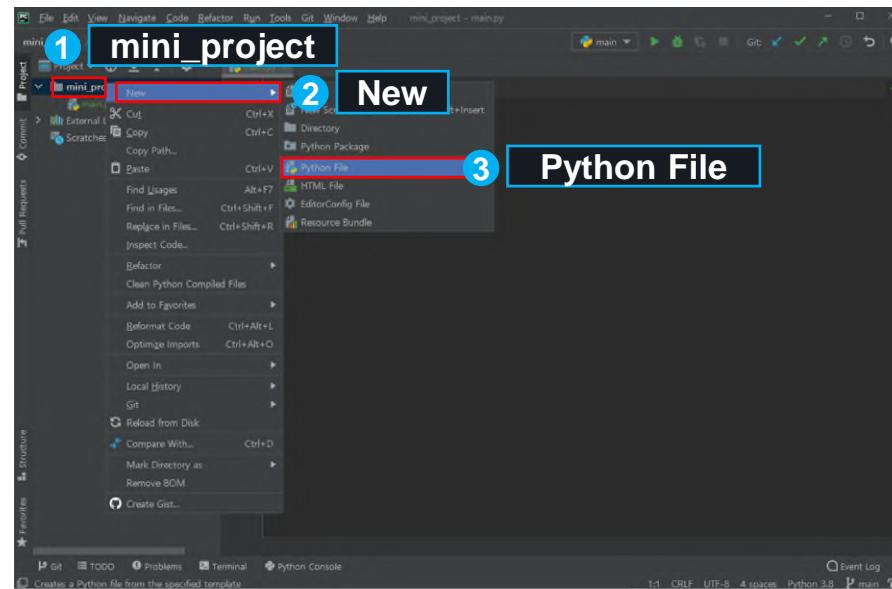
- Chọn mini_project trong Local Path kết nối với Kho lưu trữ GitHub → OK



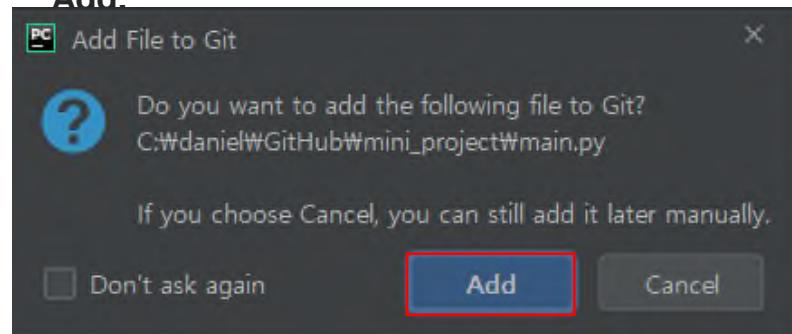
Bắt đầu với Mini-Project

Tạo file main.py (Daniel)

- Nhấp chuột phải mini_project → New → Click Python File → Nhập main là tên để khởi tạo main.py.



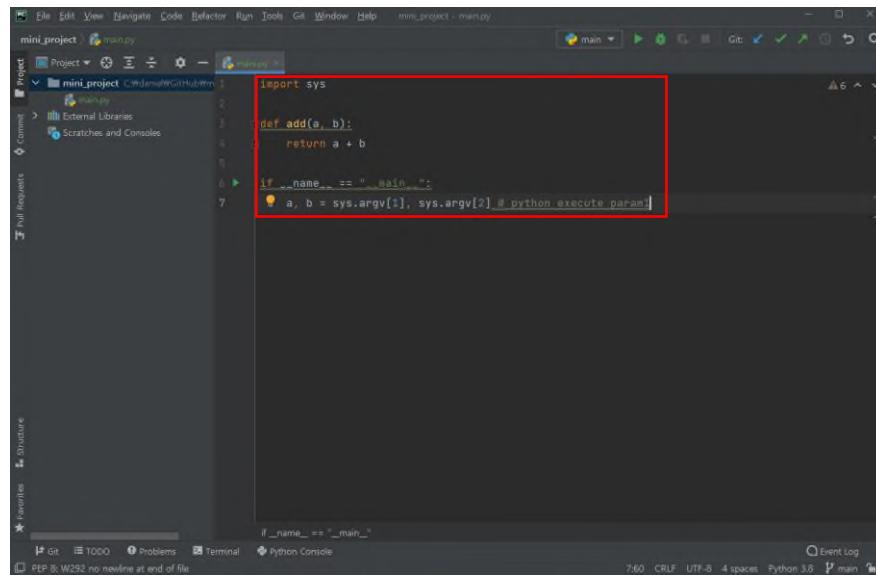
- Khi Daniel tạo file, chương trình sẽ hỏi liệu cậu ta có muốn Git theo dõi nó không. Nhấp vào Add.



Bắt đầu với Mini-Project

Tạo file main.py (Daniel)

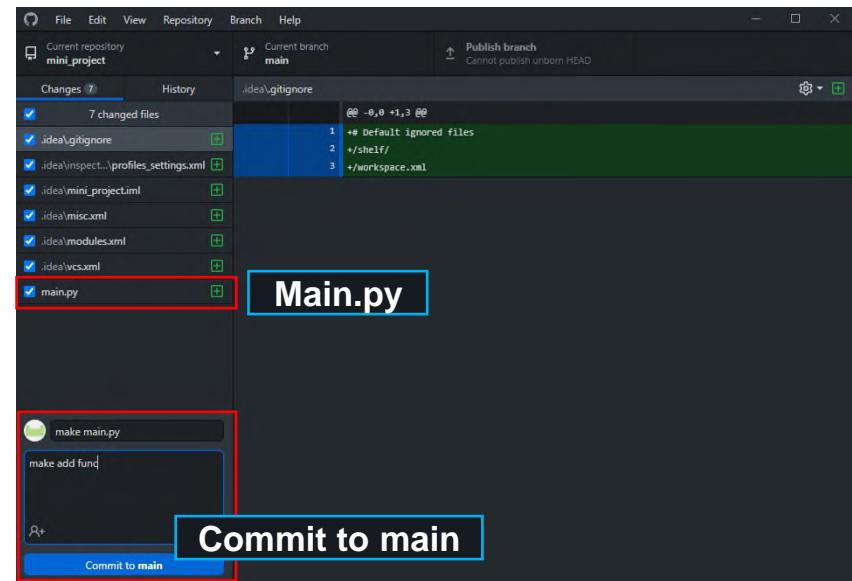
- Viết mã để tạo một chương trình tính toán các tham số thực thi Python làm đầu vào. sys.argv[1] và sys.argv[2] lần lượt là các tham số thực thi Python.
- Lưu main.py đã tạo và kiểm tra GitHub Desktop. Daniel có thể thấy rằng các tệp hỗ trợ quản lý git, bao gồm cả main.py, mới được thêm vào và theo dõi.
- Viết tin nhắn xác nhận và nhấp vào nút Commit to main.



```
import sys

def add(a, b):
    return a + b

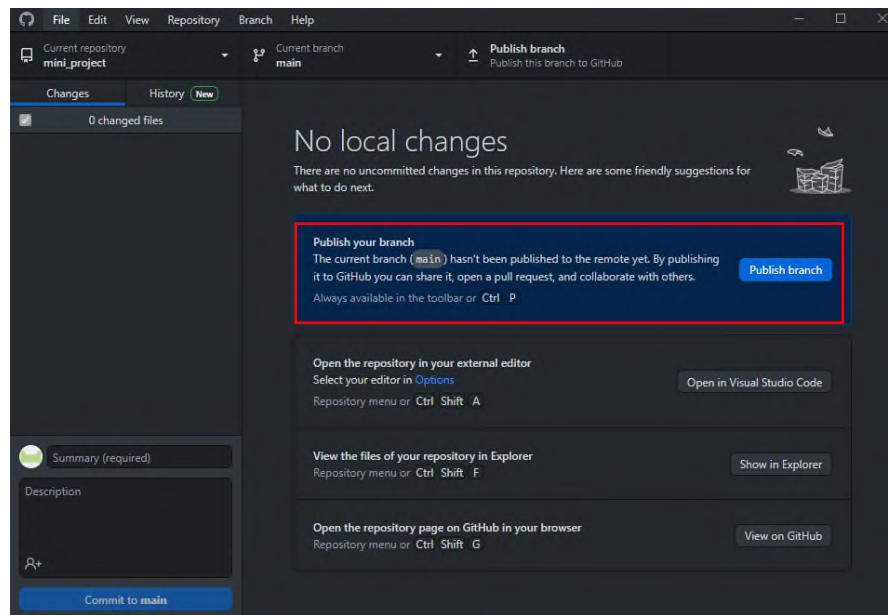
if __name__ == "__main__":
    a, b = sys.argv[1], sys.argv[2] # python execute param
```



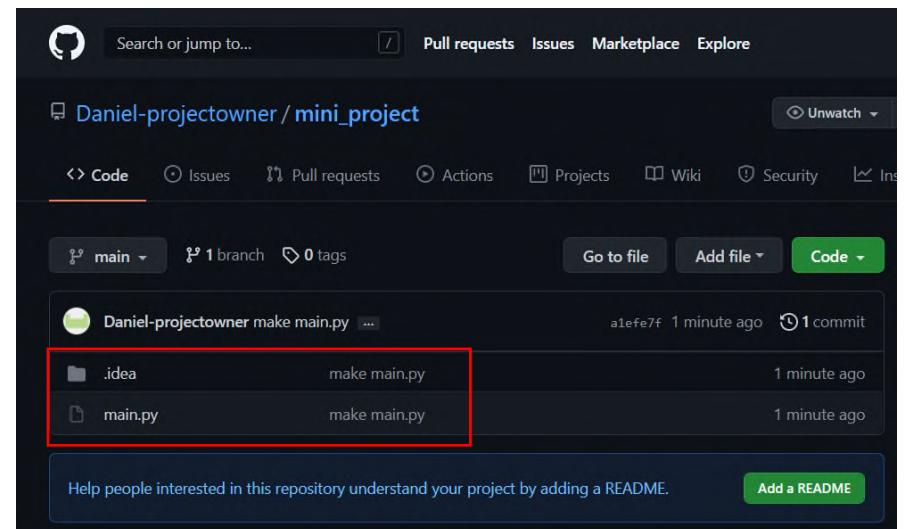
Bắt đầu với Mini-Project

Tạo file main.py (Daniel)

- Sau khi xác nhận, Daniel có thể thấy thông báo tạo nhánh. Chọn **Publish branch** để đưa nhánh đó vào Kho lưu trữ GitHub.



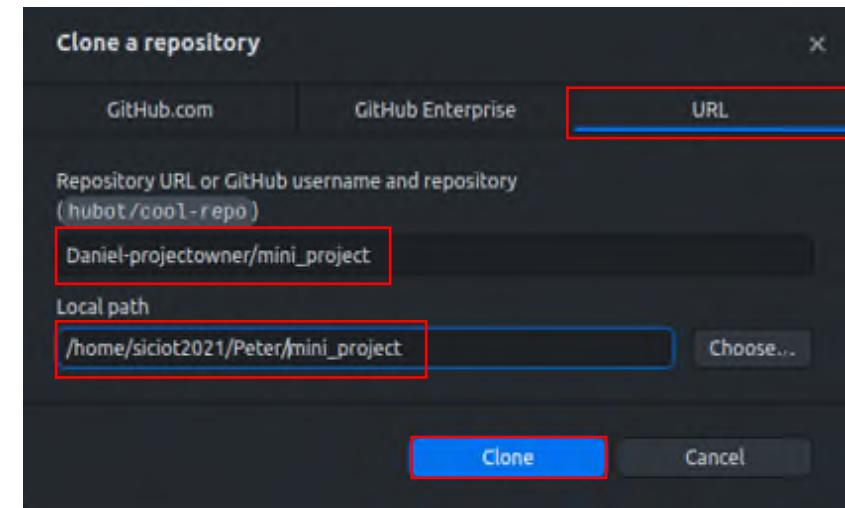
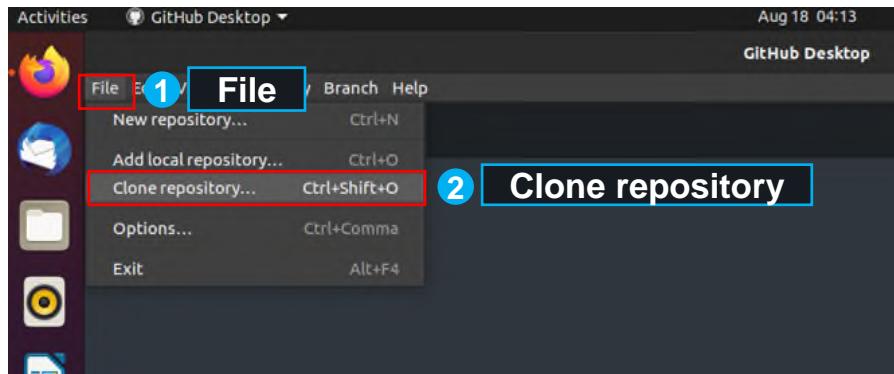
- Quay trở lại Kho lưu trữ GitHub. Daniel có thể kiểm tra xem file main.py có được tạo bằng thông báo xác nhận hay không, có nghĩa là mọi thứ hoạt động hoàn hảo.



Quy trình thực hiện Mini-Project

I Sao chép Kho dữ liệu (Peter)

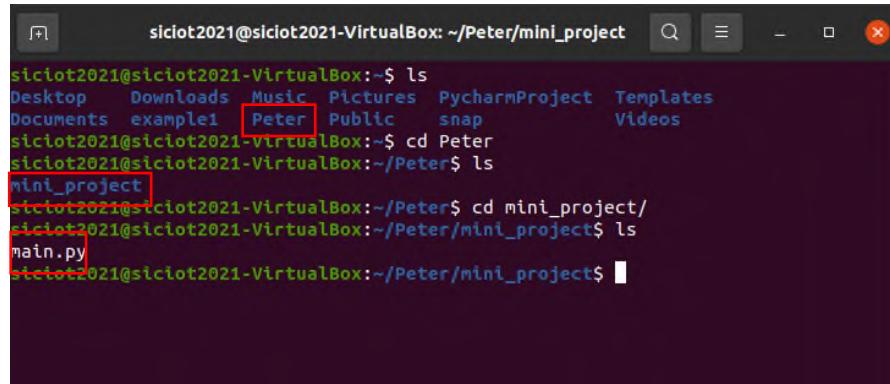
- ▶ Daniel ngừng phát triển main.py trong một thời gian để bắt đầu làm việc khác. Trong khi đó, Daniel hướng dẫn Peter để thêm và bổ sung một hàm trữ trong main.py để sử dụng thời gian của mình một cách hiệu quả.
- ▶ Chọn File trên GitHub Desktop để Peter sao chép Kho lưu trữ GitHub của Daniel → Clone repository
 - ▶ Sử dụng URL để sao chép. Viết tên GitHub và tên Kho lưu trữ của Daniel. Đặt Local Path đến một vị trí mong muốn và nhấp vào nút **Clone**.



Bắt đầu với Mini-Project

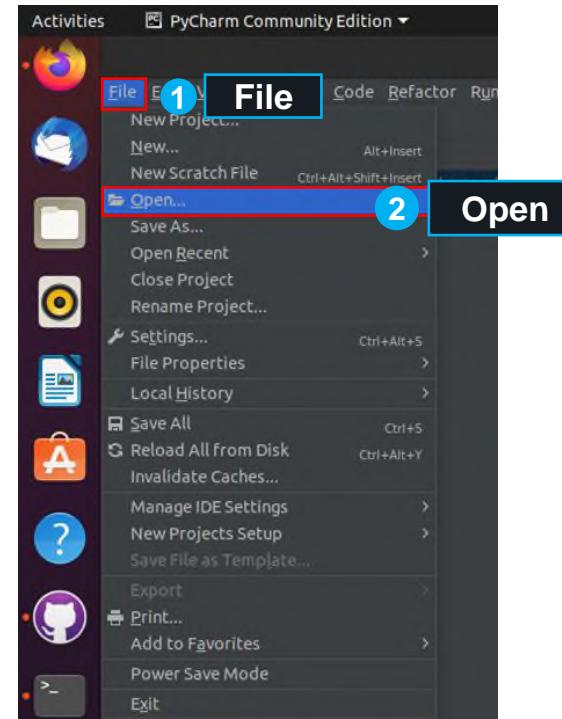
Sao chép Kho dữ liệu (Peter)

- Duyệt lại Local Path để kiểm tra xem việc sao chép có thành công hay không. Peter có thể thấy main.py đang ở trong mini_project.



```
siciot2021@siciot2021-VirtualBox: ~/Peter/mini_project
siciot2021@siciot2021-VirtualBox:~$ ls
Desktop  Downloads  Music  Pictures  PycharmProject  Templates
Documents example1  Peter   Public    snap          Videos
siciot2021@siciot2021-VirtualBox:~$ cd Peter
siciot2021@siciot2021-VirtualBox:~/Peter$ ls
mini_project
siciot2021@siciot2021-VirtualBox:~/Peter$ cd mini_project/
siciot2021@siciot2021-VirtualBox:~/Peter/mini_project$ ls
main.py
siciot2021@siciot2021-VirtualBox:~/Peter/mini_project$
```

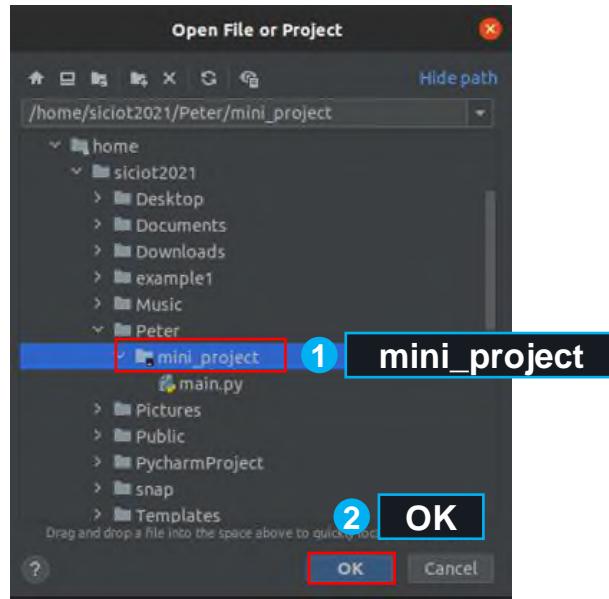
- Bắt đầu thao tác trên main.py bằng Pycharm.
- File → Open



Bắt đầu với Mini-Project

Chỉnh sửa main.py(Peter)

- Sau khi tìm thấy vị trí tương ứng với Local Path, chọn thư mục mini_project → OK
- Chọn mini_project và bấm đúp vào main.py để mở nó. Peter có thể kiểm tra đoạn mã Python do Daniel viết.



```
def add(a, b):
    return a + b

if __name__ == "__main__":
    a = sys.argv[1], sys.argv[2] # python execute parent
```

The screenshot shows the PyCharm interface with the 'main.py' file open. A red box highlights the 'main.py' file in the project tree. The code editor shows the following Python script:

```
def add(a, b):
    return a + b

if __name__ == "__main__":
    a = sys.argv[1], sys.argv[2] # python execute parent
```

Bắt đầu với Mini-Project

Chỉnh sửa main.py(Peter)

The screenshot shows the PyCharm interface with the following details:

- File Structure:** A project named "mini_project" is open, containing a file "main.py".
- Code Editor:** The code in "main.py" is as follows:

```
1 import sys
2
3 def add(a, b):
4     return a + b
5
6 def sub(a, b):
7     return a - b
8
9 if __name__ == "__main__":
10     a, b = int(sys.argv[1]), int(sys.argv[2]) # python execute param
11     print(a, b)
12     print(add(a, b))
13     print(sub(a, b))
```

- Terminal:** The terminal window shows the command \$ python3 main.py 4 1 and its output 4 1 5 3.
- Status Bar:** Shows indexing completed in 47 sec, shared indexes applied to 23% of files (832 of 3,595), Python 3.8, and the current file is main.py.

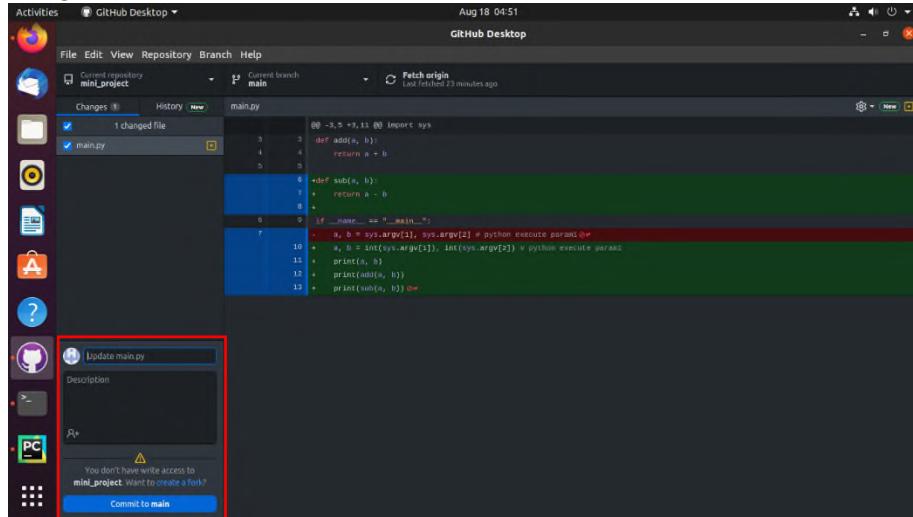
- Do các tham số không phải là kiểu số nguyên, vì vậy Peter không thu được kết quả như mong đợi. Cậu ta đã thay đổi các tham số thành loại int, thêm hàm sub và thêm một câu lệnh xuất.
- Thực thi main.py với các tham số trong Terminal để kiểm tra. Lúc này, Peter xác nhận rằng kết quả đã đúng như mong đợi.
- Lưu main.py và thoát.

```
$ python3 main.py 4 1
```

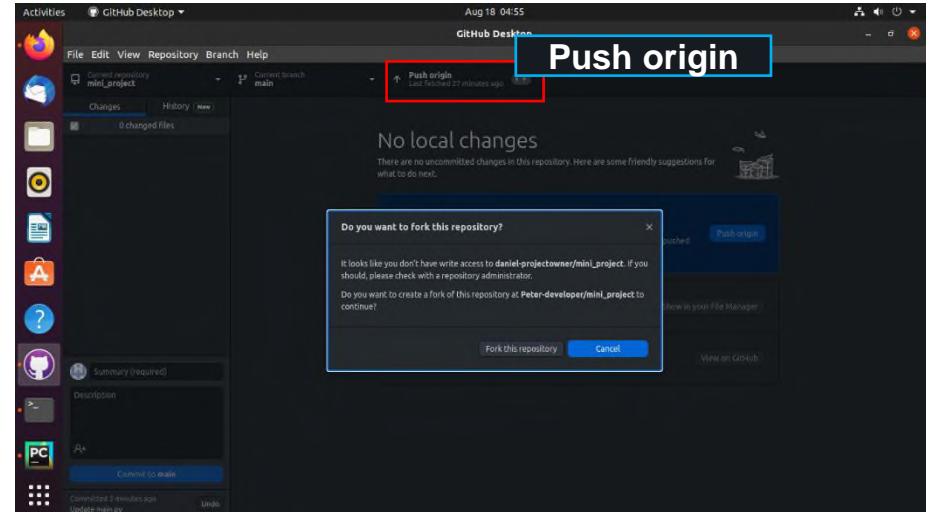
Bắt đầu với Mini-Project

Chỉnh sửa main.py(Peter)

- Quay lại GitHub Desktop. Khi Peter thử xác nhận thông qua một thông báo xác nhận, xuất hiện cảnh báo rằng cậu ta không có quyền chỉnh sửa. Lúc này cậu ta nhấp vào **Commit to main** để thử đẩy lên.



- Khi Peter cố gắng đẩy lên bằng cách nhấn nút push origin, một thông báo xuất hiện cho anh biết rằng không có quyền chỉnh sửa.



Bắt đầu với Mini-Project

- | Làm thế nào Peter có thể đẩy vào kho lưu trữ GitHub của Daniel?
- ▶ Vì kho lưu trữ GitHub mini_project thuộc sở hữu của Daniel, Peter không được phép đẩy vào kho lưu trữ này. Có hai cách chính để tránh tình trạng này.

Cách thứ nhất

- ▶ Cách đầu tiên là cấp cho Peter quyền thao tác trên Kho lưu trữ này. Khi Daniel chỉ định Peter làm Cộng tác viên, Peter có thể tự do đẩy vào Kho lưu trữ. Do kiểm soát khá lỏng lẻo, phương pháp này đòi hỏi sự quản lý chặt chẽ và thảo luận trong nhóm hơn.

Bắt đầu với Mini-Project

| Làm thế nào Peter có thể đẩy vào kho lưu trữ GitHub của Daniel?

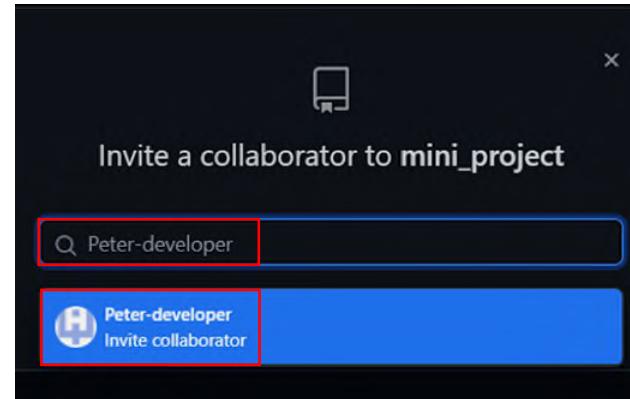
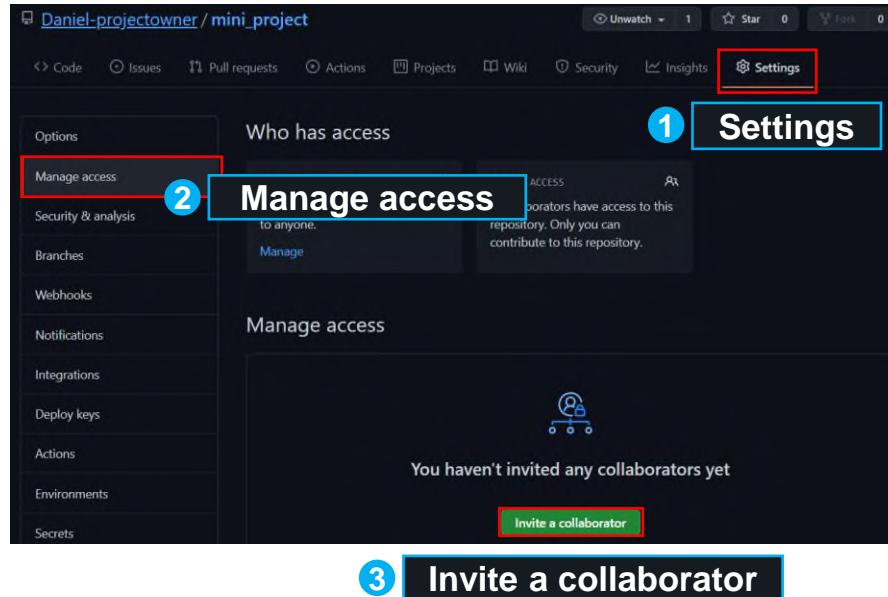
Cách thứ hai

- Cách thứ hai là Peter chia nhánh (fork) kho lưu trữ mini_project của Daniel.
 - Fork theo nghĩa đen có nghĩa là phân nhánh Kho lưu trữ của người khác và đưa nó vào Kho lưu trữ GitHub của riêng bạn.
 - Trong trường hợp này, Peter phân nhánh Kho lưu trữ của Daniel và đưa nó vào Kho lưu trữ GitHub của riêng mình, và nó khiến Kho lưu trữ GitHub của Peter tách biệt với Kho lưu trữ GitHub của Daniel.
 - Do đó, nếu Peter đẩy nội dung lên, chỉ có nội dung của Kho lưu trữ GitHub của Peter được ánh xạ, không phải Kho lưu trữ GitHub của Daniel.
- Làm thế nào Peter có thể ánh xạ tới Kho lưu trữ của Daniel với Kho lưu trữ mà anh ấy phân nhánh?
 - Có một phương thức được gọi là yêu cầu kéo (pull request). Đây là cách Kho lưu trữ GitHub của Daniel lưu các thay đổi. Kéo là một lệnh để lưu các thay đổi.
 - Sau khi Peter thực hiện Yêu cầu kéo tới Daniel và Daniel chấp thuận nó, Kho lưu trữ của Daniel ánh xạ những thay đổi có trong Yêu cầu kéo của Peter.
 - Phương pháp này thường được gọi là PR viết tắt từ Pull Request.
- Trong số hai phương pháp này, **chúng ta sẽ tìm hiểu phương pháp đầu tiên để mời Cộng tác viên.**

Quy trình thực hiện Mini-Project

Mời Peter (Daniel)

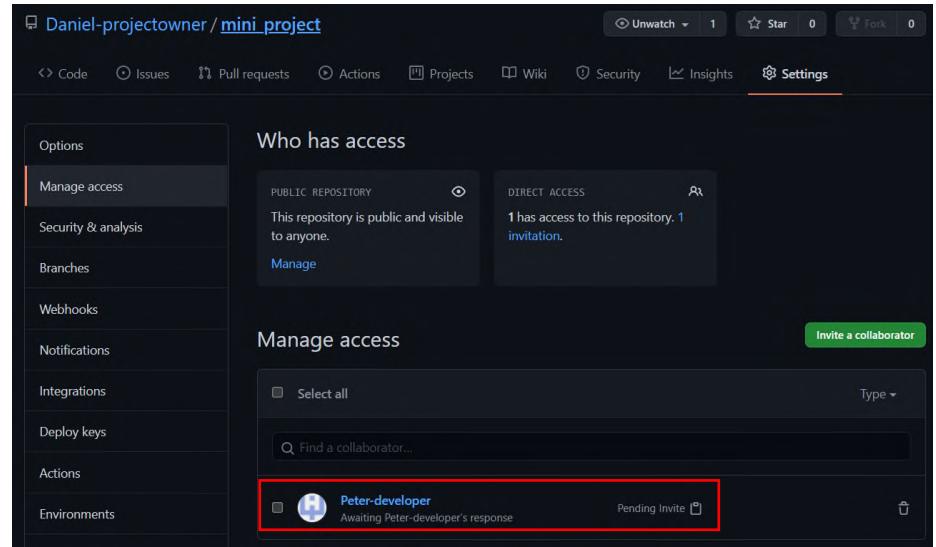
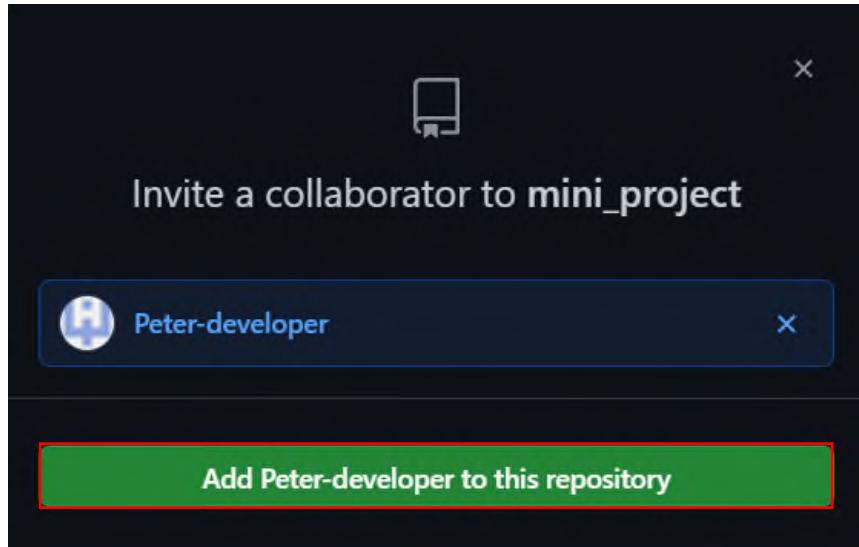
- Tới Settings trong Kho lưu trữ GitHub → Manage access → Invite a collaborator
- Đây là nơi Daniel có thể mời một tài khoản GitHub khác. Sau khi nhập tên GitHub của Peter, hãy nhấp vào tên GitHub của Peter trong số các ứng cử viên bên dưới.



Quy trình thực hiện Mini-Project

Mời Peter (Daniel)

- Nhấp vào **Add Peter-developer to this repository** để mời Peter làm cộng tác viên.
- Daniel có thể thấy rằng lời mời đã được gửi đến Peter. Và trạng thái đang chờ phản hồi của Peter. Peter cần kiểm tra email được gửi từ GitHub.

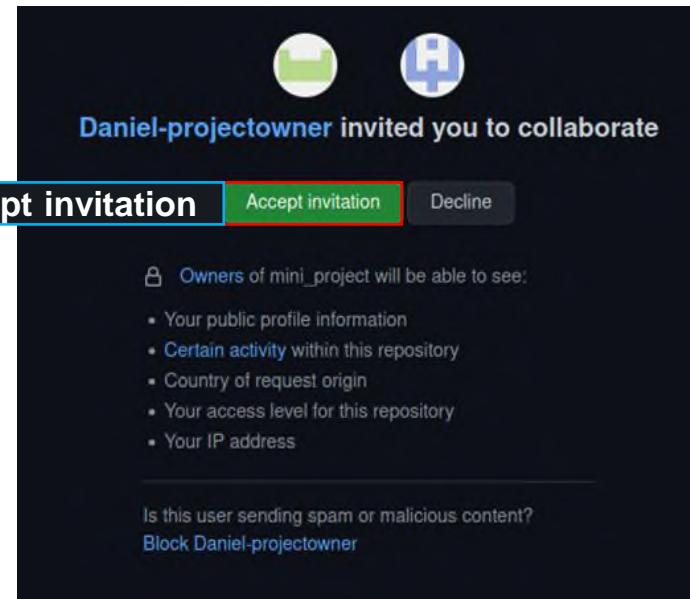
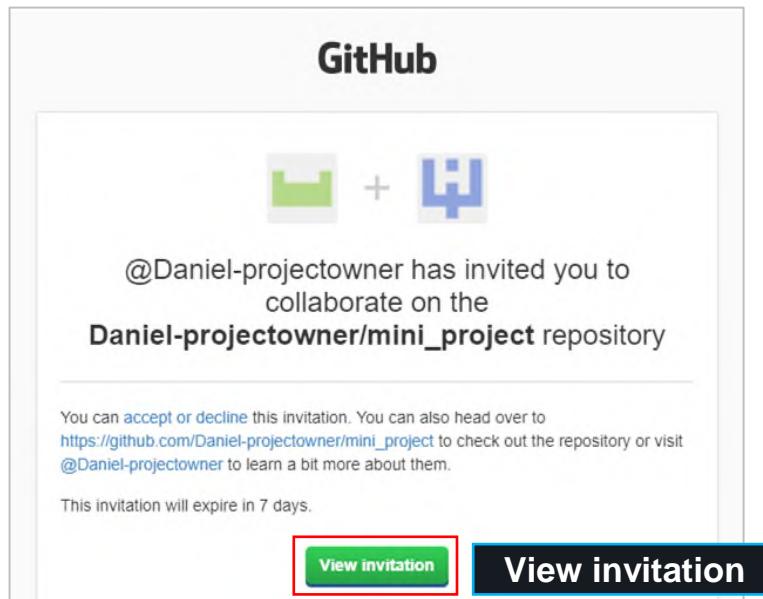


The screenshot shows the GitHub repository settings for 'mini_project'. The 'Who has access' section indicates a public repository with one user having direct access. The 'Manage access' section lists the user 'Peter-developer' with a pending invite status.

Quy trình thực hiện Mini-Project

Mời Peter (Daniel)

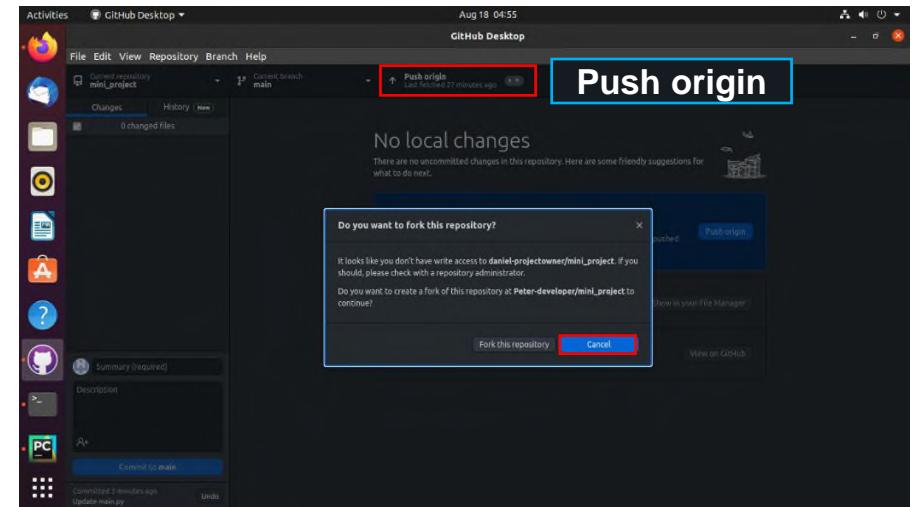
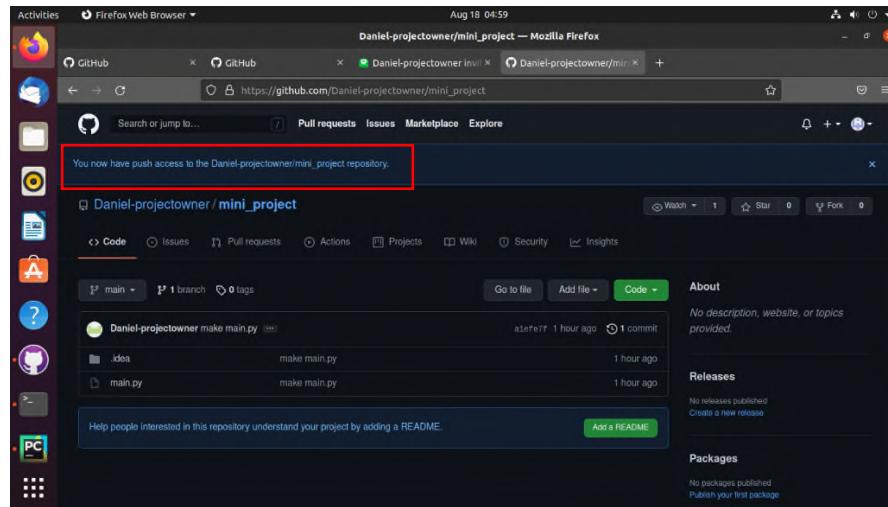
- Peter kiểm tra xem thư mời đã đến chưa. Nhấp vào biểu tượng **View invitation** để xác nhận.
- Bấm **Accept invitation** để chấp nhận lời mời và xin phép đẩy lên Kho lưu trữ của Daniel.



Quy trình thực hiện Mini-Project

Đẩy lên main.py đã chỉnh sửa (Peter)

- ▶ Kiểm tra thông báo push access đã được thực hiện.
- ▶ Quay trở lại Github Desktop. Hủy cửa sổ bật lên liên quan đến đẩy không thành công và nhấp vào **Push origin** một lần nữa để tiến hành đẩy.



Quy trình thực hiện Mini-Project

Kiểm tra kho dữ liệu Project (Daniel & Peter)

- Peter liên lạc với Daniel mà anh ấy đã thúc đẩy. Khi Daniel kiểm tra Kho lưu trữ GitHub của mình, được xác nhận rằng nó đã được ánh xạ cùng với thông điệp xác nhận của Peter. Bấm đúp vào main.py để kiểm tra main.py đã được chỉnh sửa.
- Cả Daniel và Peter đều có thể kiểm tra nội dung của main.py.

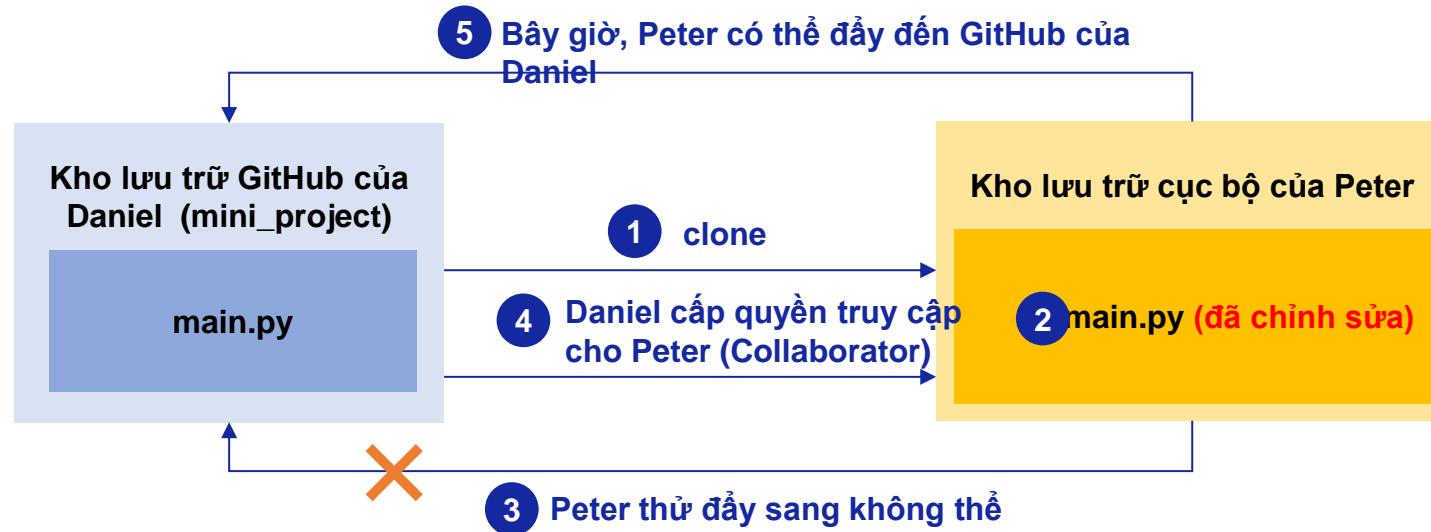
The screenshot shows a GitHub repository page for 'Daniel-projectowner / mini_project'. The 'Code' tab is selected. A commit from 'Peter-developer' is highlighted with a red box, showing the update 'Update main.py'. Another commit by 'Peter-developer' is also visible. The repository has 1 branch and 0 tags. A button to 'Add a README' is at the bottom.

The screenshot shows the code content of 'main.py' in the GitHub repository. The code is as follows:

```
1 import sys
2
3 def add(a, b):
4     return a + b
5
6 def sub(a, b):
7     return a - b
8
9 if __name__ == "__main__":
10     a, b = int(sys.argv[1]), int(sys.argv[2]) # python execute param1
11     print(a, b)
12     print(add(a, b))
13     print(sub(a, b))
```

Lưu đồ thực hiện Mini-Project

| Minh họa lưu đồ mô tả quá trình cộng tác



A photograph of a person's hands working at a desk. One hand holds a black coffee cup with a white lid, while the other hand uses a pen to point at a laptop keyboard. In the background, there is a computer monitor and a stack of papers or books.

End of Document



Together for Tomorrow! Enabling People

Education for Future Generations

©2022 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of book.

This book is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this book other than the curriculum of Samsung Innovation Campus or to use the entire or part of this book, you must receive written consent from copyright holder.