



# Bài 1

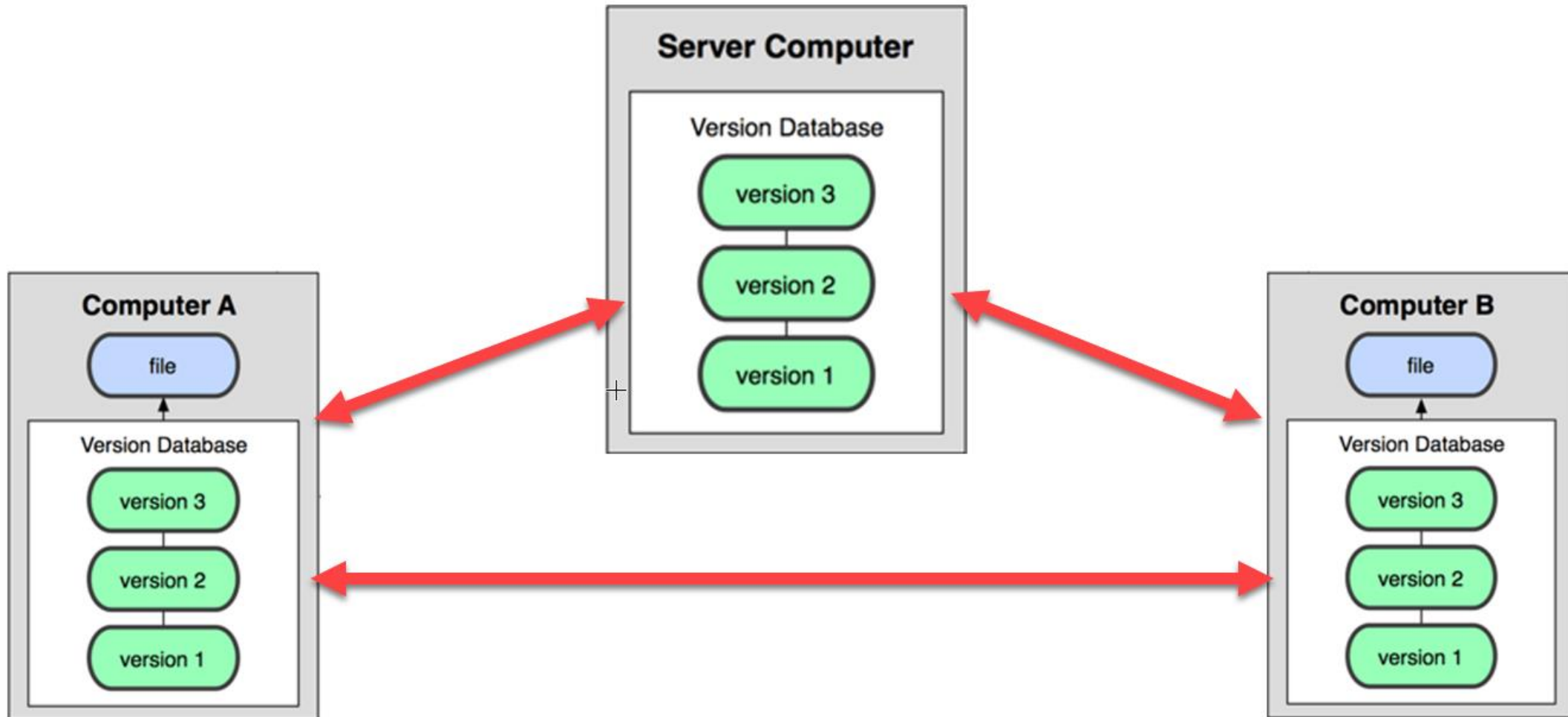
## Cài đặt và Sử dụng bằng dòng lệnh

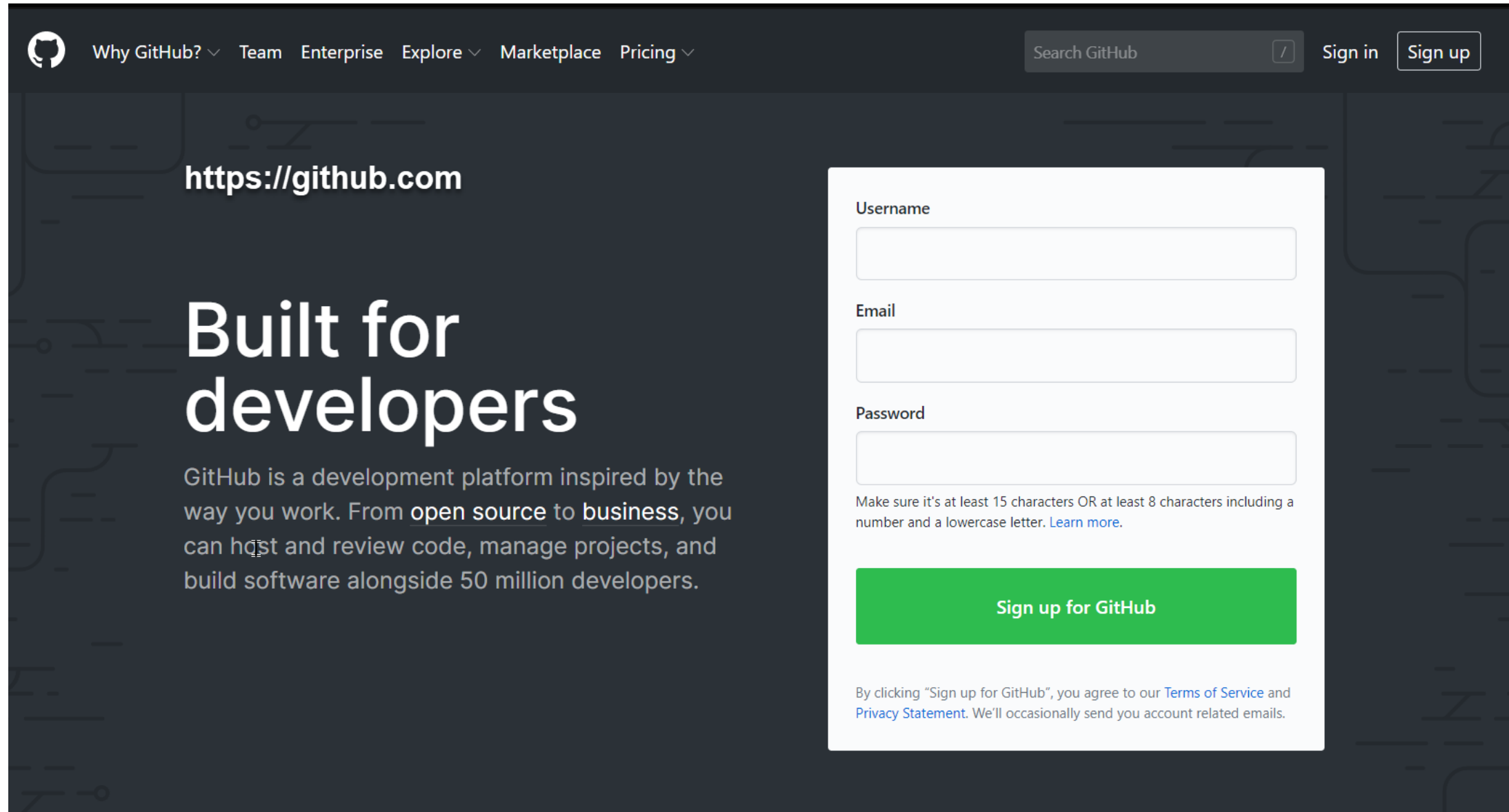


- ✓ Git & GitHub là gì ?
- ✓ Tại sao nên sử dụng Git
- ✓ Các cài đặt và thiết lập ban đầu
- ✓ Tạo repository
- ✓ Tìm hiểu về commit và staging area
- ✓ Tìm hiểu về remote repository và origin
- ✓ Branch và kỹ thuật phân nhánh
- ✓ Git log và undo commit
- ✓ Đánh dấu commit với tag

- ❖ Git là tên gọi của một Hệ thống quản lý phiên bản phân tán (**Distributed Version Control System – DVCS**) phổ biến nhất hiện nay.
- ❖ Giúp máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository),
- ❖ Mỗi thay đổi mã nguồn trên máy tính có thể được commit rồi đưa lên kho chứa mã nguồn (repository) của máy chủ
- ❖ Máy tính khác cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia.
- ❖ Trong Git, thư mục làm việc trên máy tính gọi là Working Tree
- ❖ **Github** chính là một dịch vụ máy chủ repository công cộng, mỗi người có thể tạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc

- Distributed Version Control System – DVCS





The image shows the GitHub homepage with a dark background. At the top, there is a navigation bar with the GitHub logo, links for 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing', a search bar, and 'Sign in' and 'Sign up' buttons. The main content area features the URL 'https://github.com' and the heading 'Built for developers'. Below this, a paragraph describes GitHub as a development platform. On the right, a white sign-up form is displayed with fields for 'Username', 'Email', and 'Password', followed by a green 'Sign up for GitHub' button. At the bottom of the form, there is a disclaimer about agreeing to the Terms of Service and Privacy Statement.

https://github.com

## Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.

- ❖ Git khi lưu trữ mã nguồn miễn phí.
- ❖ Git dễ sử dụng, an toàn và nhanh chóng.
- ❖ Dễ dàng quản lý, chia sẻ mã nguồn theo các phiên bản
- ❖ Xem lại lịch sử cập nhật các phiên bản để dàng
- ❖ Giúp quy trình làm việc code theo nhóm đơn giản hơn bằng việc kết hợp các phân nhánh (branch).
- ❖ Có thể làm việc ở bất cứ đâu vì chỉ cần clone mã nguồn từ kho chứa
- ❖ Dễ dàng trong việc deployment sản phẩm
- ❖ Trên github còn cung cấp các chức năng **issues** cho phép comment, trao đổi, đóng góp...

Hướng dẫn này cài đặt trên windows

Tải file .exe tại link <https://git-scm.com/download/win>

- ❖ Chọn phiên bản phù hợp với hệ điều hành windows của bạn.
- ❖ Thường là bản 64bit
- ❖ Sau khi tải về tiến hành cài đặt, tại các màn hình cài đặt cứ để mặc định và nhấn next cho tới khi kết thúc



- ❖ Để kiểm tra git đã cài đặt thành công, mở command line của windows lên và gõ lệnh **git --version**



```
C:\Windows\system32\cmd.exe

C:\Users\HAITHANH>git --version
git version 2.30.2.windows.1

C:\Users\HAITHANH>
```

**GIT --VERSION**

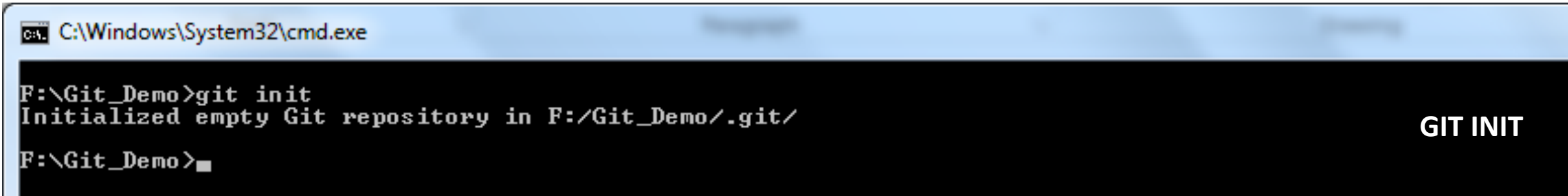
- ❖ Tiến hành đăng ký tài khoản git tại link <https://github.com/> và đăng nhập vào
- ❖ Vẫn tại màn hình CMD trên tiến hành cấu hình tài khoản để sử dụng trong quá trình gõ lệnh
- ❖ **git config --global user.name** "Họ tên của bạn"
- ❖ **git config --global user.email** "địa chỉ email đã đăng ký tài khoản git ở bước trên"
- ❖ Để xem lại thông tin cấu hình tài khoản của bước trên, truy cập vào thư mục **C:\Users\pc\_name** và mở file **.gitconfig**, pc\_name là tên user trên máy tính của bạn, như trường hợp này là **HAITHANH**
- ❖ Hoặc có thể dùng lệnh **git config --list**



C:\Windows\system32\cmd.exe

```
C:\Users\HAITHANH>git config --global user.name "bkaphaibt"
C:\Users\HAITHANH>git config --global user.email "bthai93@gmail.com"
C:\Users\HAITHANH>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
core.editor="C:\Users\HAITHANH\AppData\Local\Programs\Microsoft VS Code\Code.exe" --wait
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
user.name=bkaphaibt
user.email=bthai93@gmail.com
username.email=bthai93@gmail.com
```

- Repository (kho chứa) nghĩa là nơi mà bạn sẽ lưu trữ mã nguồn và một người khác có thể sao chép (clone) lại mã nguồn đó để làm việc.
- Repository có hai loại là **Local Repository** (Kho chứa trên máy cá nhân) và **Remote Repository** (Kho chứa trên một máy chủ từ xa).
- **Tạo Local Repository**
- Tạo một thư mục bất kỳ trên máy tính hoặc thư mục dự án đã có sẵn
- Mở **CMD** lên truy cập vào thư mục dự án : VD git\_demo, sau đó gõ lệnh **git init**



```
C:\Windows\System32\cmd.exe

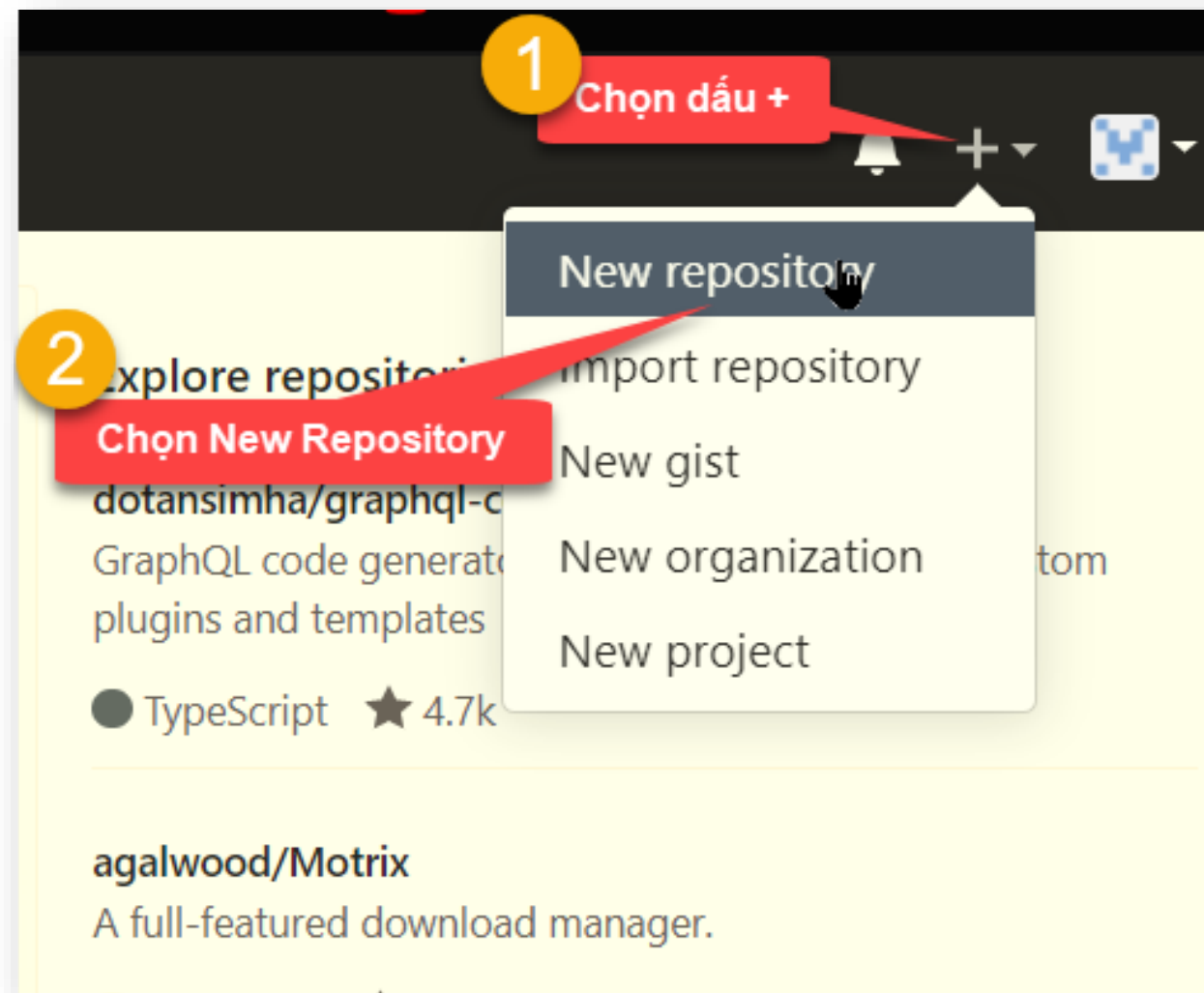
F:\Git_Demo>git init
Initialized empty Git repository in F:/Git_Demo/.git/


F:\Git_Demo>
```

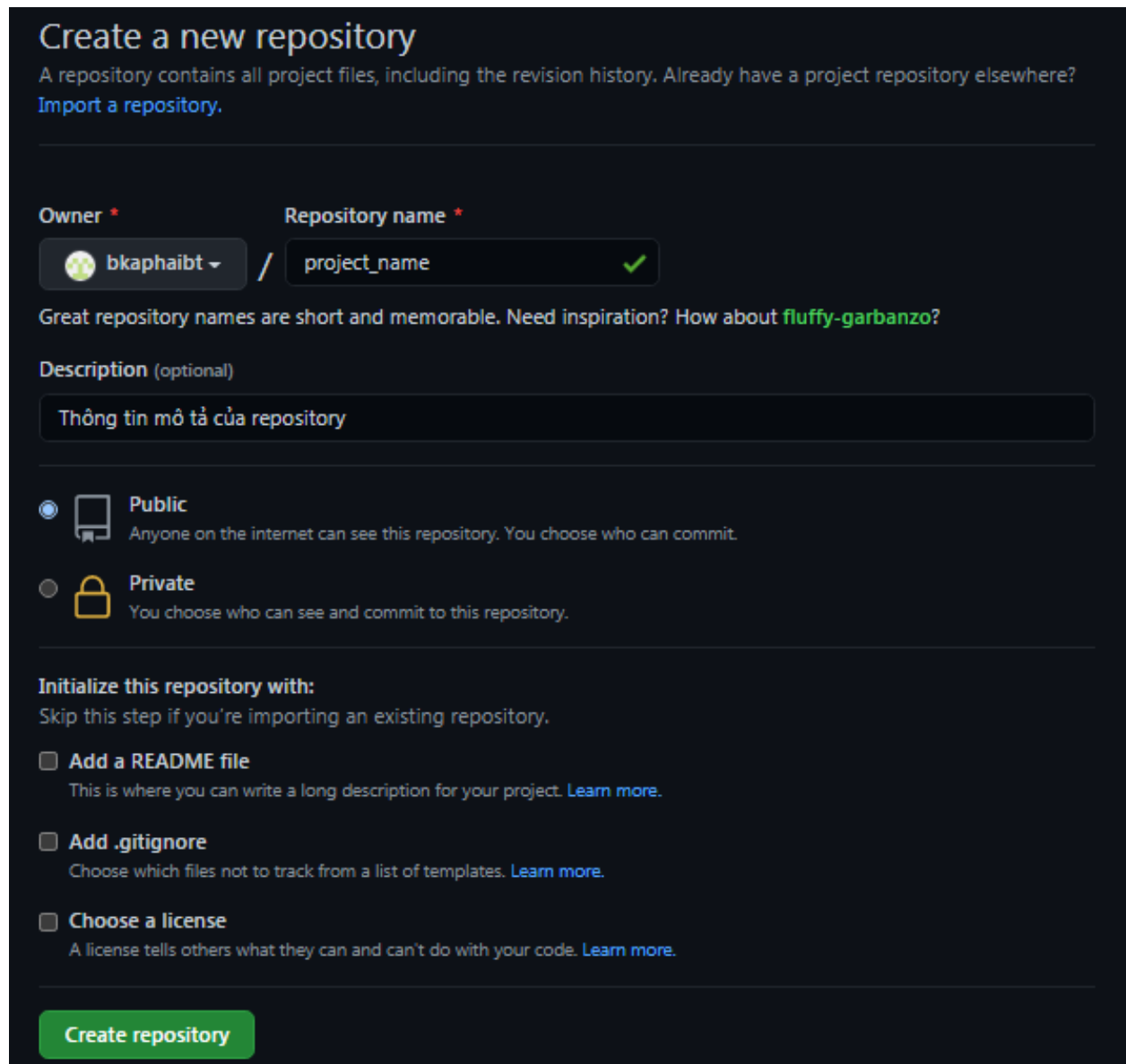
**GIT INIT**

- Lúc đó trong thư mục dự án sẽ có thêm thư mục **.git** chứa các thông tin cấu hình cho **Local Repository** này (Mặc định thư mục này ở dạng ẩn, không nhìn thấy)

- ❖ Tạo Remote Repository
- ❖ Truy cập vào <https://github.com> và đăng nhập bằng tài khoản đã đăng ký
- ❖ Nhìn bên góc phải trên cùng của trình duyệt và làm theo các bước sau



- ❖ Tại màn hình này nhập
- ❖ Repository name (Bắt buộc) VD `my_project`
- ❖ Description (Không bắt buộc)
- ❖ Click vào nút 





**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---


**Owner \*** **Repository name \***


 bkaphaibt /  

Great repository names are short and memorable. Need inspiration? How about [fluffy-garbanzo?](#)

**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.

---

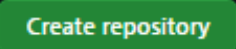
**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **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.](#)

---



- Github là một máy chủ repository từ xa nên mình sẽ gọi nó là Remote Repository
- Để đưa tài nguyên lên **remote repository** có thể sử dụng lệnh
- **git push origin master** sau khi commit
- Để kết nối với một remote repository, sử dụng lệnh: **git remote add <name> <url>**
- Để kiểm tra tên remote, bạn có thể gõ lệnh **git remote -v**.

```
C:\Windows\System32\cmd.exe

F:\Git_Demo>git remote add origin https://github.com/bkaphaibt/project_name

F:\Git_Demo>git remote -v
origin    https://github.com/bkaphaibt/project_name (fetch)
origin    https://github.com/bkaphaibt/project_name (push)
```

Bây giờ khi commit hay push bạn có thể gõ  
**git push origin master** để gửi mã nguồn lên remote repository này.

- Đổi tên remote: **git remote rename <old\_name> <new\_name>**

```
C:\Windows\System32\cmd.exe
```

```
F:\Git_Demo>git remote rename origin haibt
```

```
F:\Git_Demo>
```

- Thêm một remote
- Sử dụng lệnh git remote add tên\_remote URL. VD:
- git remote add remote\_new https://github.com/bkaphaibt/project\_name

```
C:\Windows\System32\cmd.exe
```

```
F:\Git_Demo>git remote -v
```

```
haibt https://github.com/bkaphaibt/project_name (fetch)
```

```
haibt https://github.com/bkaphaibt/project_name (push)
```

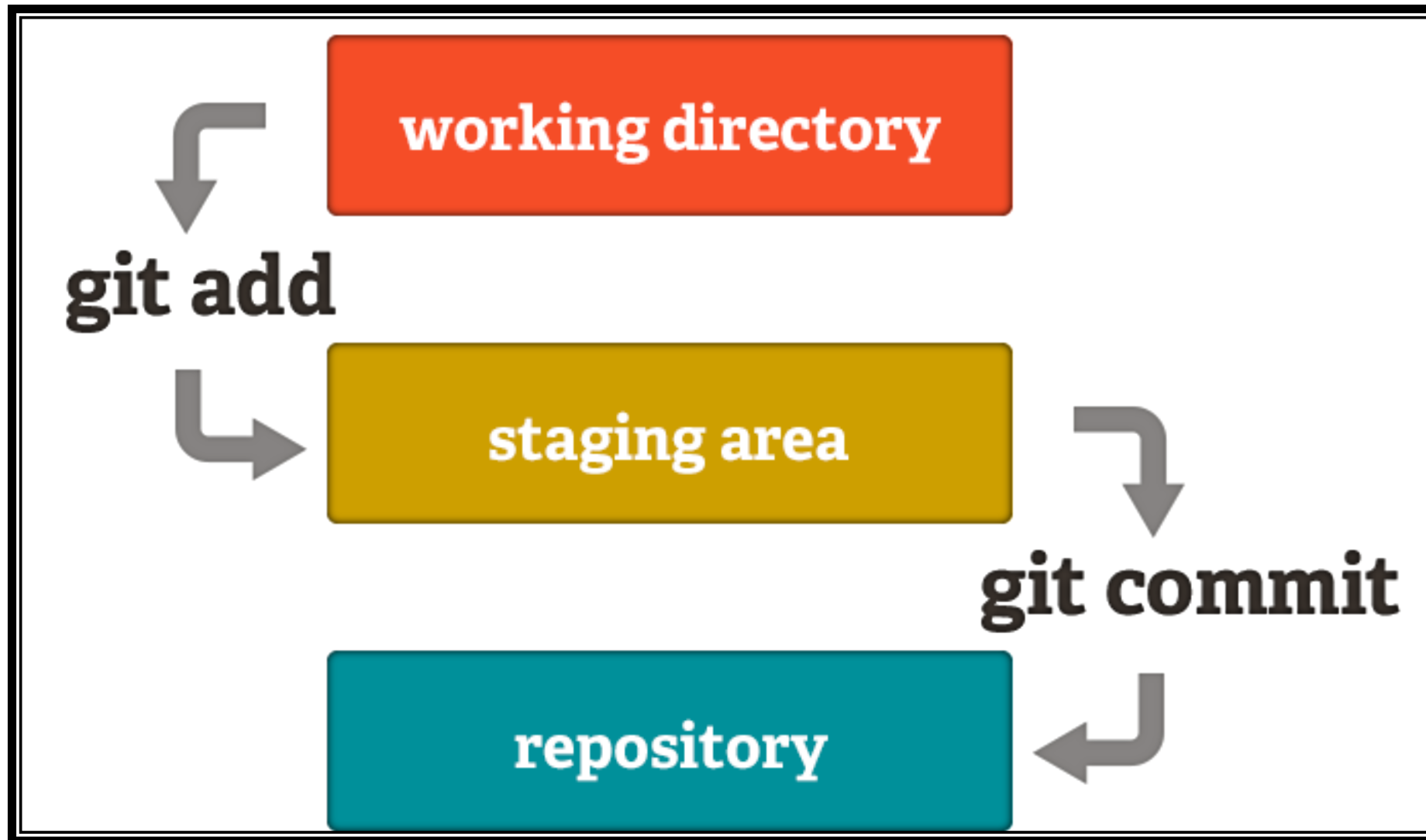
```
remote_new https://github.com/bkaphaibt/project_name (fetch)
```

```
remote_new https://github.com/bkaphaibt/project_name (push)
```

- **git clone**
- Lệnh này chỉ nên sử dụng khi bạn cần tạo mới một Git mới trên máy tính với toàn bộ dữ liệu và thiết lập của một remote repository.
- **git pull**
- Lệnh này sẽ tự động lấy toàn bộ dữ liệu từ remote repository và gộp vào cái branch hiện tại bạn đang làm việc.
- **git fetch**
- Lệnh này sẽ lấy toàn bộ dữ liệu từ remote repository nhưng sẽ cho phép bạn gộp thủ công vào một branch nào đó trên thư mục Git ở máy tính.

- ❖ Staging Area là gì?
- ❖ Trong các hệ thống quản lý phiên bản (Version Control System) thì các dữ liệu sẽ được lưu trữ ở hai nơi,
  - ❖ Thư mục bạn đang làm việc trên máy tính (**working tree**)
  - ❖ Kho chứa mã nguồn (**repository**, ví dụ như kho chứa trên Github).
- ❖ Git có một khu vực trung gian gọi là **Staging Area** và đây chính là một lợi thế lớn của Git.
- ❖ **Staging Area** là một khu vực trung gian lưu trữ những thay đổi của bạn trên tập tin được chuẩn bị cho quá trình commit.
- ❖ Một tập tin khi nằm trong Staging Area sẽ có trạng thái là Staged
- ❖ Để đưa tập tin vào Staging Area cần phải sử dụng lệnh **git add tên\_file**





- ❖ Commit là gì và nó hoạt động ra sao?
- ❖ Commit là một hành động để Git lưu lại một bản chụp (snapshot) của các sự thay đổi trong thư mục làm việc trên máy tính (working tree).
- ❖ Các tập tin và thư mục được thay đổi đã phải nằm trong Staging Area.
- ❖ Mỗi lần commit nó sẽ được lưu lại lịch sử chỉnh sửa của mã nguồn kèm theo tên và địa chỉ email của người commit.
- ❖ Trong Git có thể khôi phục lại tập tin trong lịch sử commit của nó để phân nhánh (branch) khác
- ❖ Lệnh commit trong Git sẽ là **git commit -m** "Tin nhắn, mô tả, ghi chú..."

- ❖ Điều kiện gì để commit một tập tin?
- ❖ Để commit một tập tin, phải đưa tập tin đó vào trạng thái tracked bằng lệnh `git add tên_file`. Trong git có hai loại trạng thái chính đó là Tracked và Untracked, cụ thể:
  - ❖ **Tracked** – Là tập tin đã được đánh dấu theo dõi trong Git để bạn làm việc với nó. Và trạng thái Tracked nó sẽ có thêm các trạng thái phụ khác là Unmodified (chưa chỉnh sửa gì), Modified (đã chỉnh sửa) và Staged (đã sẵn sàng để commit).
  - ❖ **Untracked** – Là tập tin còn lại mà bạn sẽ không muốn làm việc với nó trong Git.
- ❖ Nếu tập tin đó đã được Tracked nhưng đang rơi vào trạng thái (Modified) thì nó vẫn sẽ không thể commit được, phải đưa nó về Staged cũng bằng lệnh **`git add tên_file`**

- ❖ Chuyển Tracked về UnTracked
- ❖ Tracked về Untracked với lệnh **git rm tên\_file**.
- ❖ Lệnh git rm sẽ giúp bạn đưa tập tin về trạng thái Untracked nhưng không xóa hẳn trong ổ cứng.
- ❖ Còn nếu bạn muốn xóa nó luôn thì dùng lệnh **git rm -f tên\_file** và nhớ cẩn thận khi dùng lệnh này.

- Nếu bạn cần xóa bỏ lần commit trước và cần undo để commit lại thì có thể sử dụng tham số **--amend** trong lệnh git commit.
- Cú pháp lệnh có dạng
- **\$ git commit --amend -m "Hehe"**
- [master 3682e56] Hehe
- ...
- Lưu ý rằng undo nghĩa là bạn quay trở lại bước commit lần trước, do vậy nếu cần bổ sung tập tin nào vào để commit thì hãy đưa tập tin đó vào Staging Area trước.
- **Bỏ tập tin ra khỏi Staging Area**
- Nếu bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo thì có thể sử dụng lệnh **git reset HEAD tên\_file**

- Nếu bạn cần xóa bỏ lần commit trước và cần undo để commit lại thì có thể sử dụng tham số **--amend** trong lệnh git commit.
- Cú pháp lệnh có dạng
- **\$ git commit --amend -m "Hehe"**
- [master 3682e56] Hehe
- ...
- Lưu ý rằng undo nghĩa là bạn quay trở lại bước commit lần trước, do vậy nếu cần bổ sung tập tin nào vào để commit thì hãy đưa tập tin đó vào Staging Area trước.
- **Bỏ tập tin ra khỏi Staging Area**
- Nếu bạn đã đưa một tập tin nào đó vào Staging Area nhưng bây giờ bạn muốn loại bỏ nó ra khỏi đây để không phải bị commit theo thì có thể sử dụng lệnh git reset HEAD tên\_file

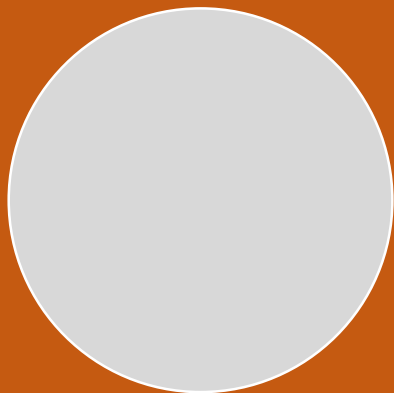
- Khi bắt đầu khởi tạo một repository hoặc clone một repository, bạn sẽ có một nhánh (branch) chính tên là master
- Đây là branch mà sẽ chứa toàn bộ các mã nguồn chính trong repository.



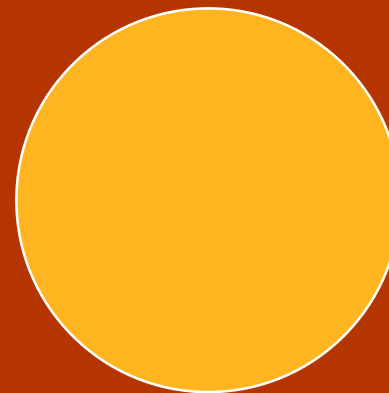
- Bây giờ nếu bạn muốn tạo một sự thay đổi nào đó mà không ảnh hưởng đến branch master thì sẽ cần tạo ra một branch mới với tên là develop chẳng hạn.
- Và từ đó mỗi khi bạn thực hiện lệnh checkout vào branch nào đó thì toàn bộ mã nguồn trên working tree của bạn sẽ được đổi sang môi trường dành cho branch đang checkout.

- **Xem toàn bộ** các branch mà bạn đang có bằng lệnh **git branch**.
- **Tạo thêm branch**, chỉ cần gõ lệnh **git branch tên\_branch**. Ví dụ mình cần tạo branch develop.
- **git branch develop**
- **git checkout develop** (Chuyển đổi sang branch develop)
- **git push origin develop** (Tải tài nguyên lên branch develop)





git checkout master  
(Chuyển về branch cần  
gộp master)



git merge develop (Gộp  
branch develop vào  
branch hiện tại là master)



- Tag giúp việc đánh dấu các lệnh commit để tìm kiếm dễ dàng hơn
- Nó rất hữu ích khi bạn commit quá nhiều lần, sau này tìm kiếm lại các commit với git log khá khó khăn, trong khi với tag lại dễ dàng hơn

- ❖ Một việc bạn sẽ khá thường xuyên làm trong Git nếu làm việc theo nhóm đó là kiểm tra xem những ai đã commit vào dự án bạn đang làm việc,
- ❖ Xem git log
- ❖ Để xem lịch sử của các lần commit trước đó, bạn sử dụng lệnh **git log** là sẽ thấy.

```
commit 6e729a49a36b31919daa6263f8f98f3a59d5bab3
Author: BKAP GITLAB <bkapgitlab@bachkhoa-apttech.edu.vn>
Date: Tue Apr 21 14:47:47 2020 -0700
First commit on Github
```

- Mỗi lần commit sẽ có một checksum riêng, và nó cũng có ghi rõ ai là người commit vào và commit vào ngày bao nhiêu, lúc nào.
- Ngoài ra, bạn có thể chèn thêm tham số **-p** vào để hiển thị chi tiết của mỗi lần commit.
- Bạn còn có thể sử dụng thêm một số tùy chọn xem log sau để tối ưu hơn quy trình đọc log.
- **--since, --after**: Xem các lần commit kể từ ngày nhất định.
- **--until**: Xem các lần commit trước từ ngày nhất định.
- **--author**: Xem các lần commit của một người nào đó.
- **--grep**: Lọc các chuỗi trong log và in ra.

- Cách sử dụng có dạng
- **git log --pretty="%tag"** , VD \$ **git log --pretty="%an - %s"**
- Kết quả
- **BKAP GITLAB - Hihi**
- **BKAP GITLAB - First commit on Github**
- Các %tag phải được đặt trong cặp dấu ngoặc kép và bạn có thể sử dụng nhiều %tag khác nhau.
- **Danh sách các %tag:**

<b>%H</b> – Commit hash	<b>%ae</b> – Author e-mail
<b>%h</b> – Abbreviated commit hash	<b>%ad</b> – Author date (format respects the – <b>date=option</b> )
<b>%T</b> – Tree hash	<b>%ar</b> – Author date, relative
<b>%t</b> – Abbreviated tree hash	<b>%cn</b> – Committer name
<b>%P</b> – Parent hashes	<b>%ce</b> – Committer email
<b>%p</b> – Abbreviated parent hashes	<b>%cd</b> – Committer date
<b>%an</b> – Author name	<b>%cr</b> – Committer date, relative
	<b>%s</b> – Subject

# HỎI ĐÁP





# TRẢI NGHIỆM THỰC HÀNH



## TRÂN TRỌNG CẢM ƠN!



238 Hoàng Quốc Việt, Bắc Từ Liêm, Hà Nội



0968.27.6996



[tuyensinh@bachkhoa-apttech.edu.vn](mailto:tuyensinh@bachkhoa-apttech.edu.vn)



[www.bachkhoa-apttech.edu.vn](http://www.bachkhoa-apttech.edu.vn)