

Quản lý phiên bản trong thiết kế (Git)

Nhóm 5

CHƯƠNG 1:

Giới thiệu

Tìm hiểu về Git, quản lí phiên bản
sử dụng Git



GIT

Là một đại diện tiêu biểu của hệ thống quản lý phiên bản phân tán DVCS mã nguồn mở, một trong những dạng hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay



Lịch sử phát triển

Git được tạo ra bởi Linus Torvalds vào năm 2005.

Ban đầu, Git được sử dụng cho việc phát triển và quản lý mã nguồn của dự án Linux.

Git đã được sử dụng rộng rãi trong cộng đồng phát triển phần mềm.

Git được sử dụng trong các dự án phát triển phần mềm lớn như Android, jQuery, Ruby on Rails, và nhiều dự án phần mềm mã nguồn mở khác trên toàn cầu.

Đặc điểm của GIT

- Có hướng tiếp cận mới so với các hệ thống Source Control khác như SVN hay CVS truyền thống.
- Có nhiệm vụ theo dõi những thay đổi, chỉnh sửa trong Source Code của người dùng vào mỗi thời điểm và đồng bộ những Source Code do họ chỉnh sửa lên Server cùng đồng nghiệp.

Ưu điểm

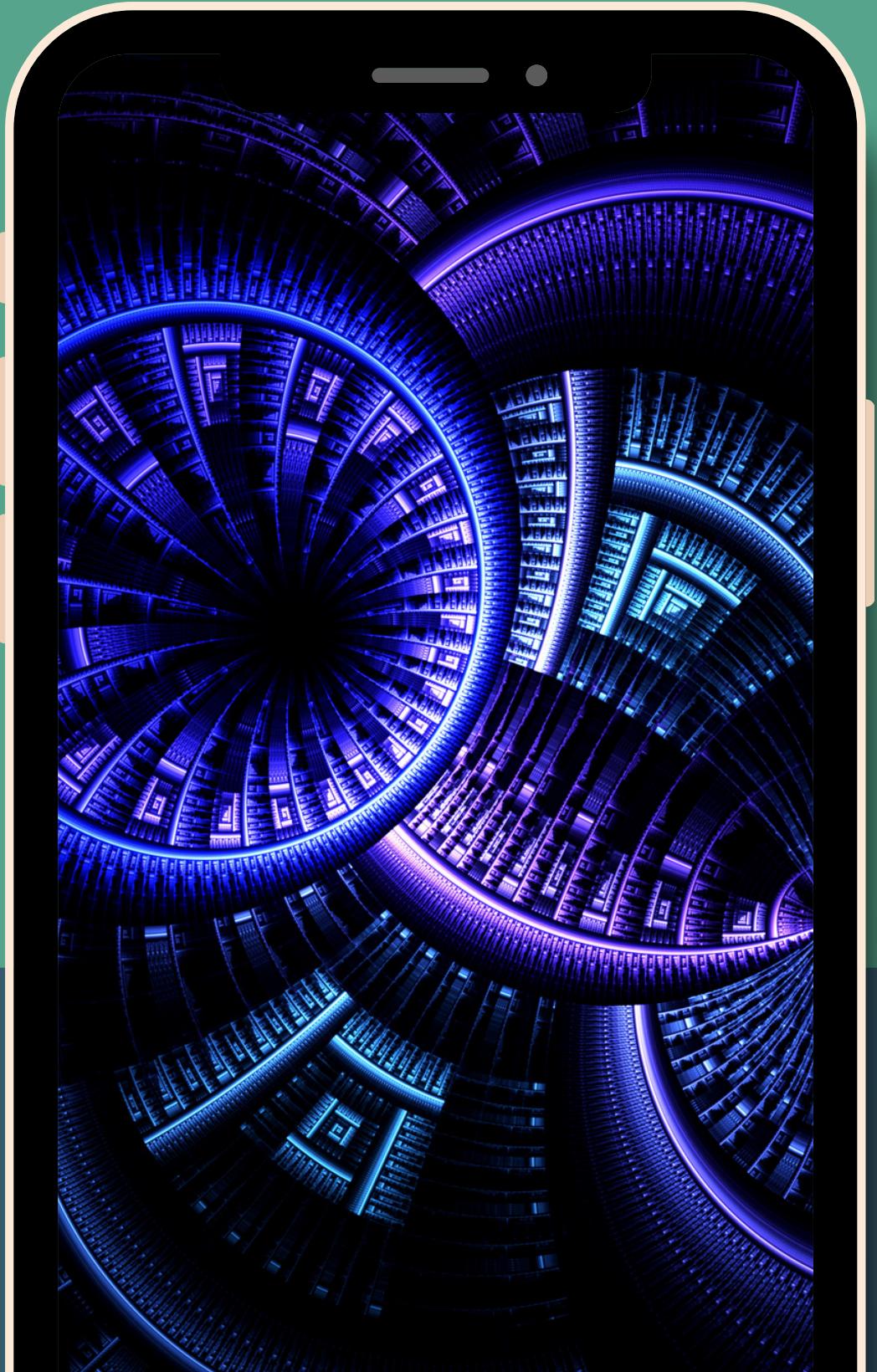
- Tốc độ xử lý nhanh hơn so với nhiều hệ thống quản lý phiên bản khác.
- Khả năng xử lý các xung đột giữa các thay đổi dữ liệu giữa các nhánh một cách tốt hơn.
- Tính bảo mật cao với các tính năng chống phá hoại, sao chép dự phòng, và mã hóa.
- Sử dụng dễ dàng và hỗ trợ đa nền tảng.

Nhược điểm

- Học cú pháp Git ban đầu có thể khó khăn đối với người mới bắt đầu.
- Quản lý phiên bản phân tán có thể dẫn đến việc tăng kích thước repository nhanh hơn so với hệ thống quản lý phiên bản tập trung.
- Không có tính năng giải quyết xung đột tự động hoàn toàn, yêu cầu sự can thiệp của người dùng.
- Không thể xóa các commit trước đó một cách dễ dàng mà ảnh hưởng đến lịch sử phiên bản.
- Có thể gặp phải các vấn đề về hiệu suất nếu repository quá lớn hoặc có quá nhiều branch.

Quản lý phiên bản sử dụng Git

Quản lý phiên bản là một khía cạnh quan trọng trong quá trình phát triển phần mềm. Git là một trong những công cụ quản lý phiên bản phổ biến nhất hiện nay, được sử dụng rộng rãi trong cộng đồng phát triển phần mềm, là một công cụ quản lý mã nguồn cho dự án Linux.



CHƯƠNG 2

CÀI ĐẶT GIT



Trên WINDOWS

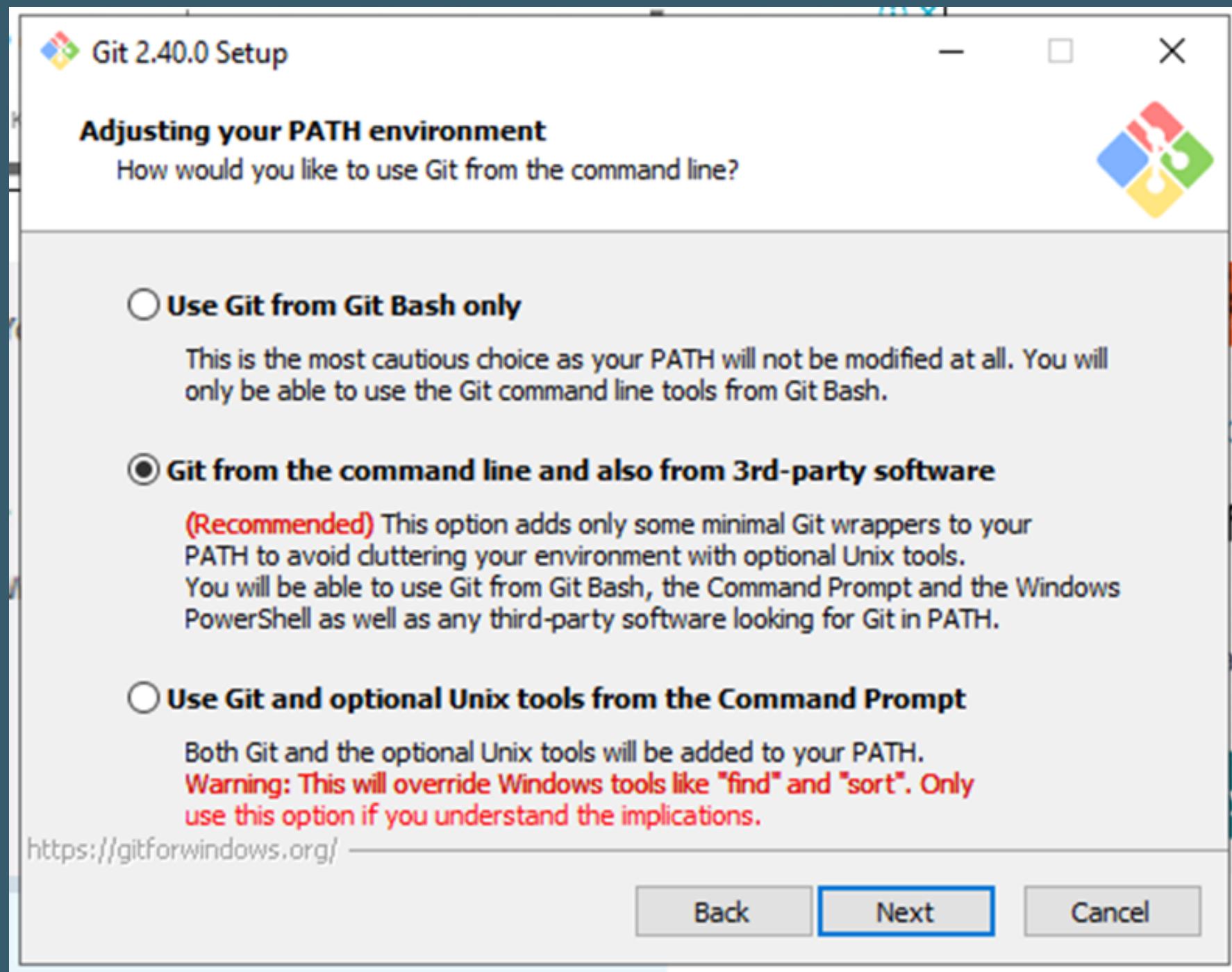
Bước 1: Truy cập vào đường dẫn <https://gitforwindows.org/>

Bước 2: Tải xuống một file *.exe, nhấn chuột vào file để tiến hành cài đặt

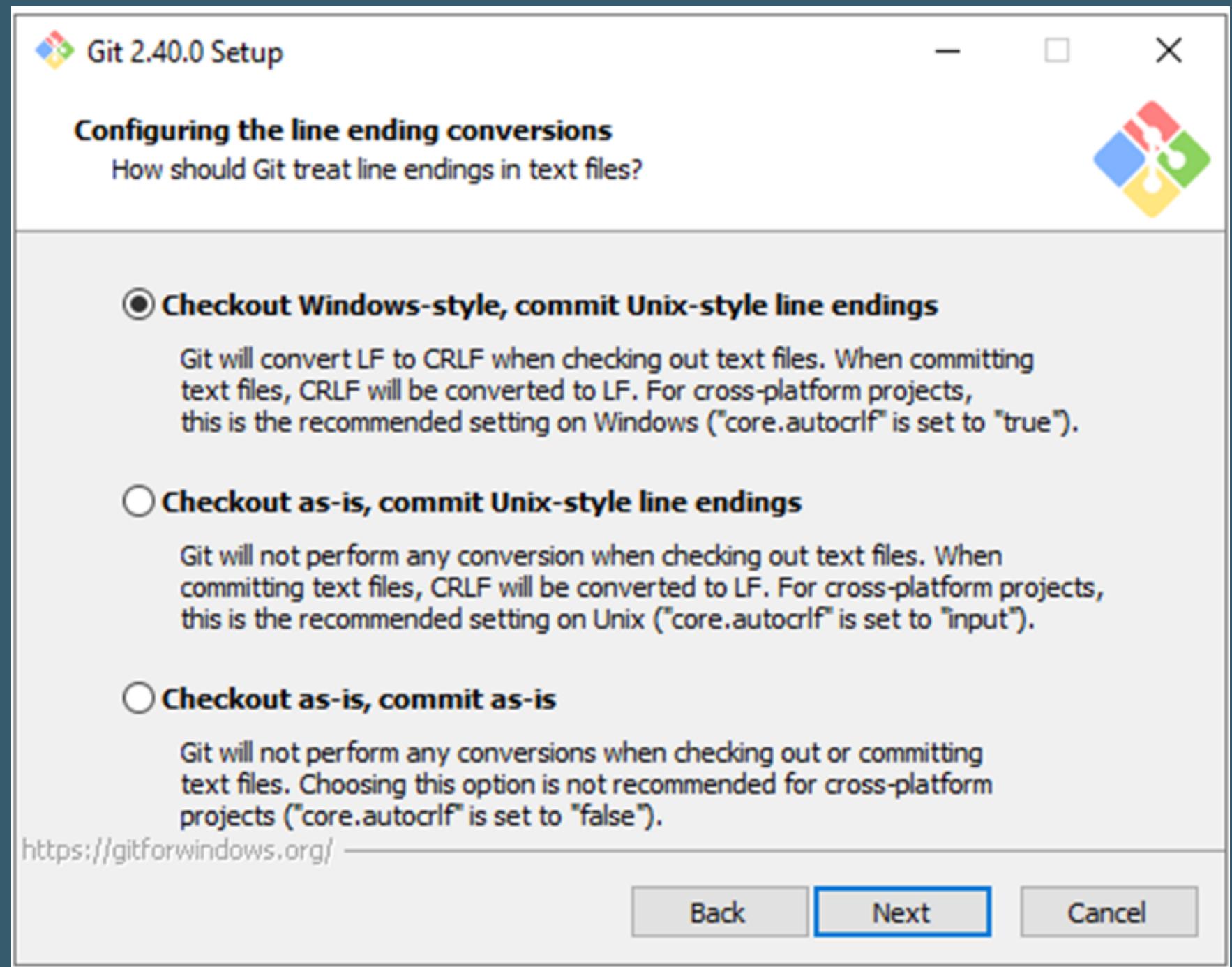
Bước 3: Cài đặt bình thường như các phần mềm khác có hỗ trợ dao diện cho Windows, dựa vào yêu cầu
Của mỗi người dùng mà chọn lựa cho phù hợp.

Bước 4: Bước tiếp theo là cài git editor cho windows, có thể là các text editor như vim, notepad++, sublime text,...

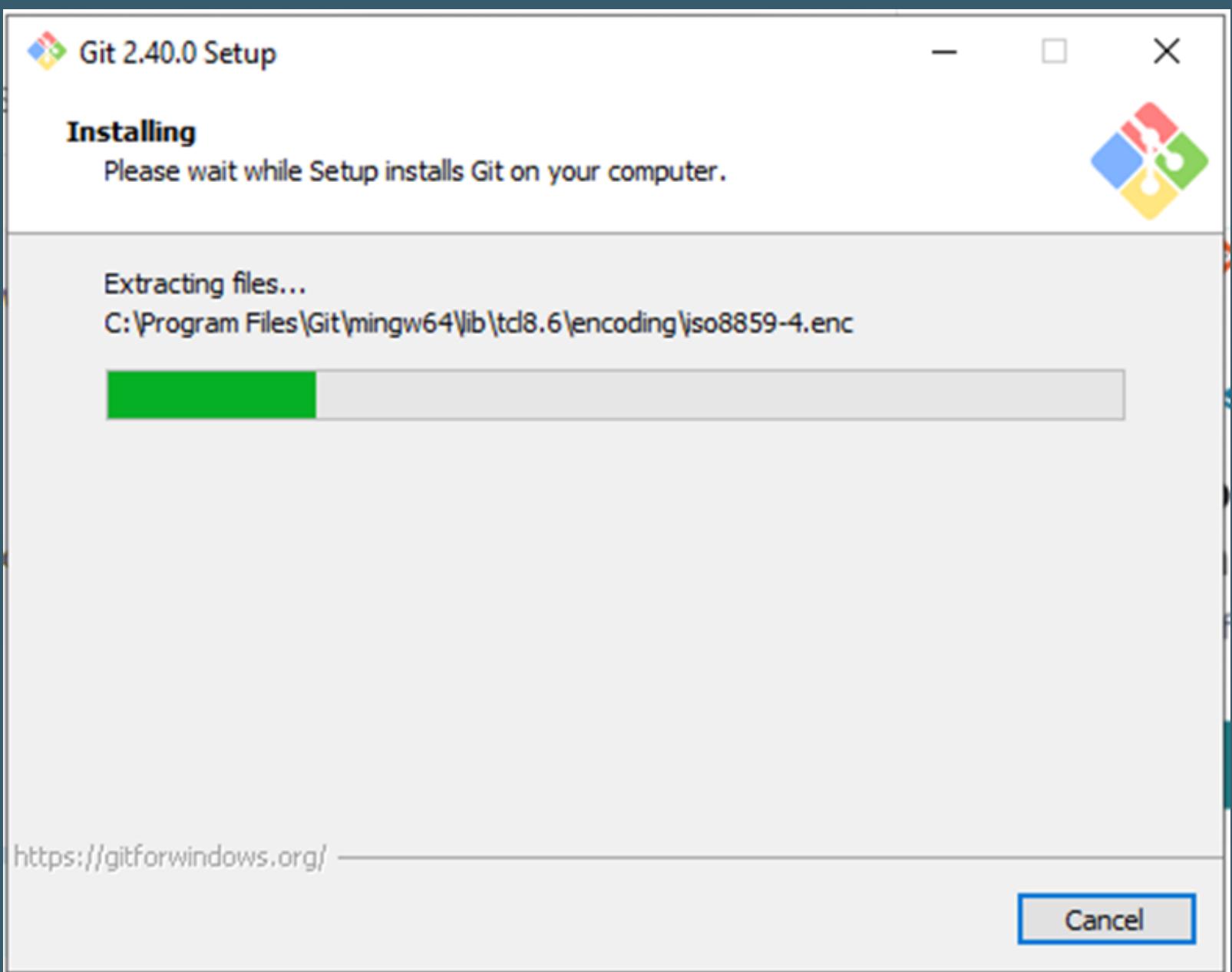
Bước 5: Tiếp theo, lựa PATH môi trường thích hợp với nhu cầu



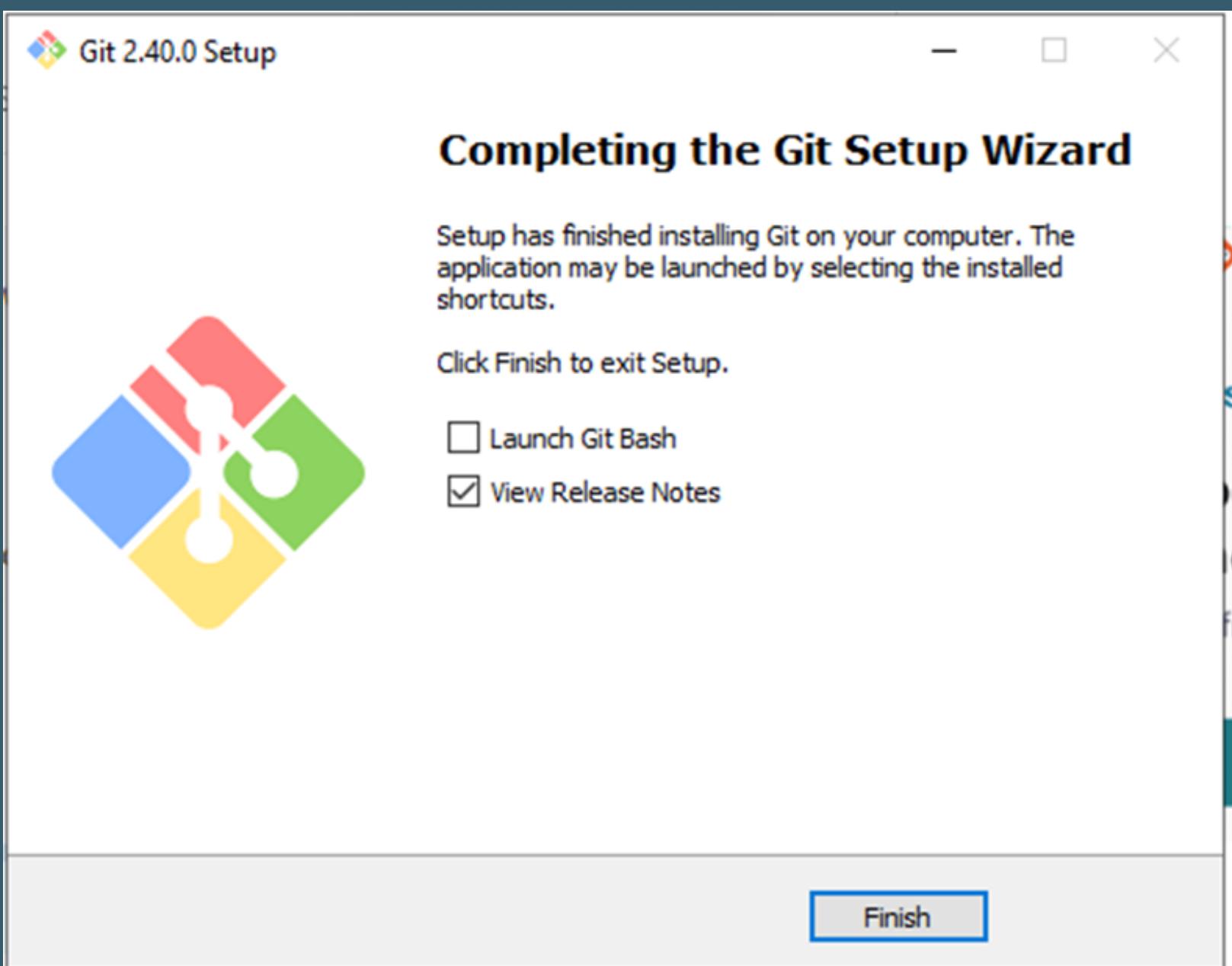
Bước 6: Chọn phong cách dòng lệnh như thế nào



Bước 7: Bấm install để cài đặt



Bước 8: Cài đặt thành công



Trên Linux

Bước 1: Mở terminal và gõ 2 lệnh sau

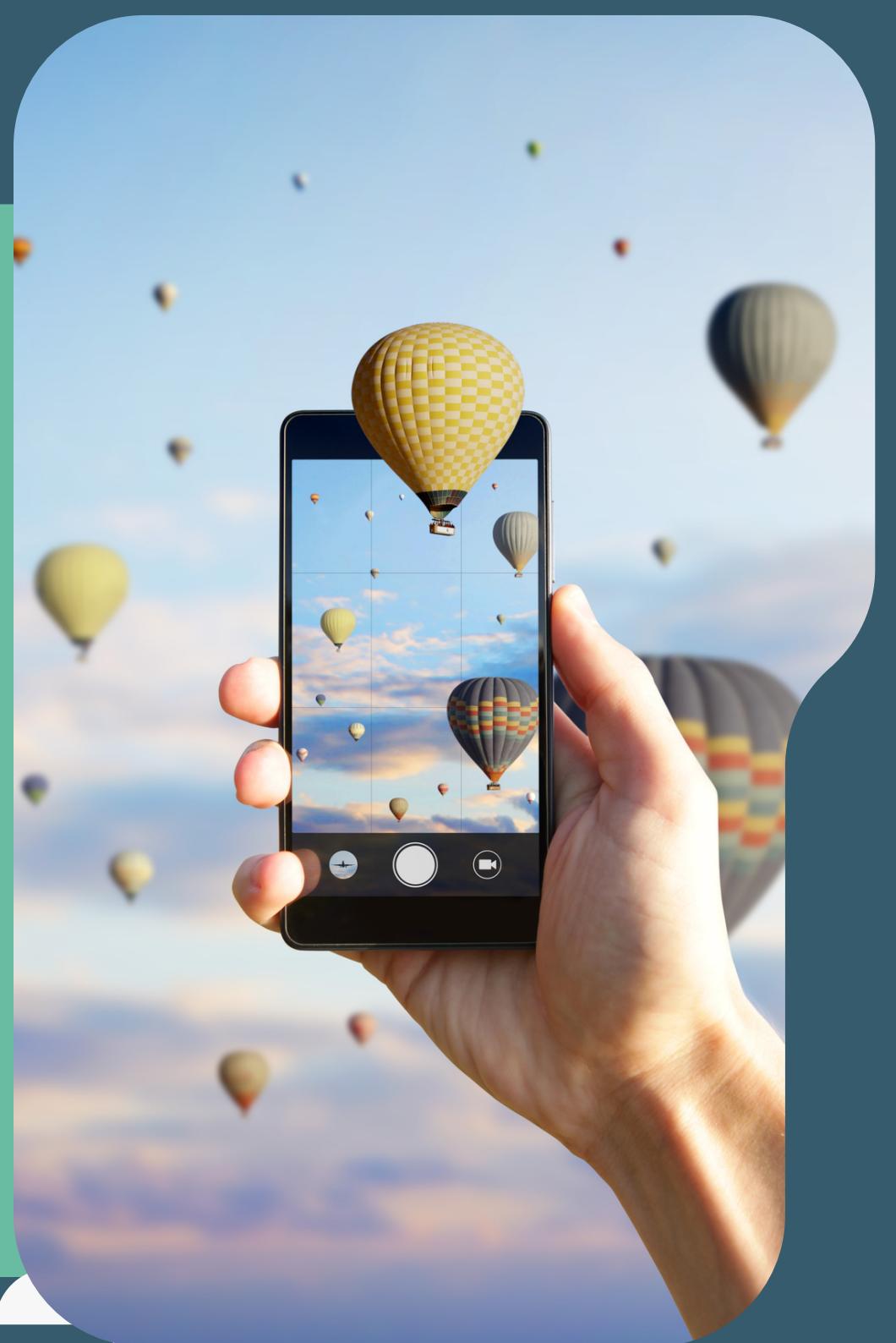
```
Sudo apt-get update  
Sudo apt-get install git
```

Bước 2: Gõ lệnh 'git-version' để kiểm tra đã cài đặt thành công

```
student@nathanlee:~$ git --version  
git version 2.20.1
```

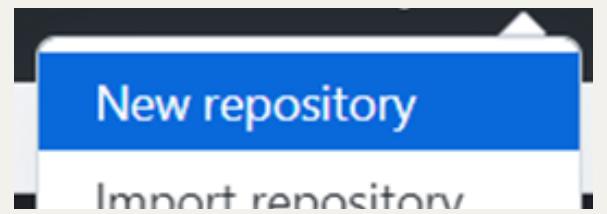
CHƯƠNG 3

CẤU HÌNH GIT



Tạo Repository

Bước 1: Người dùng bấm vào ký hiệu dấu cộng been cạnh ảnh đại diện của tài khoản và chọn New Repository, Github sẽ chuyển qua trang khởi tạo Repo mới

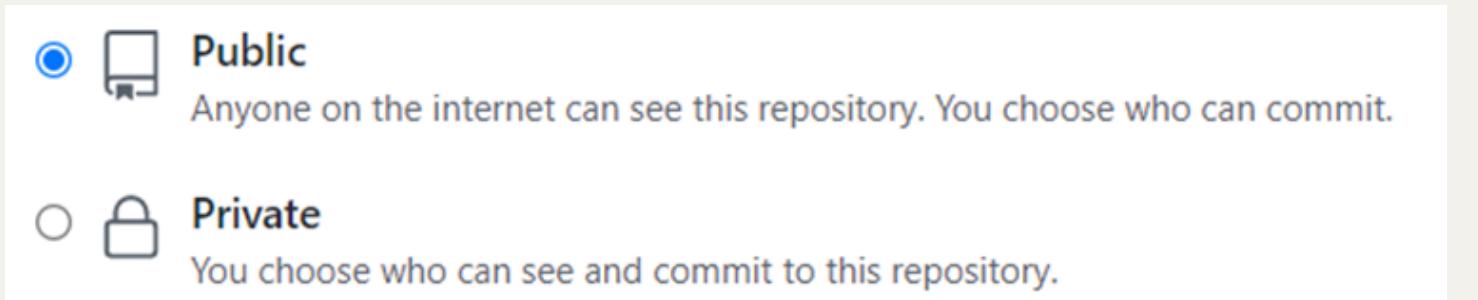


Bước 2: Đặt tên Repo cũng như miêu tả dự án cần thiết

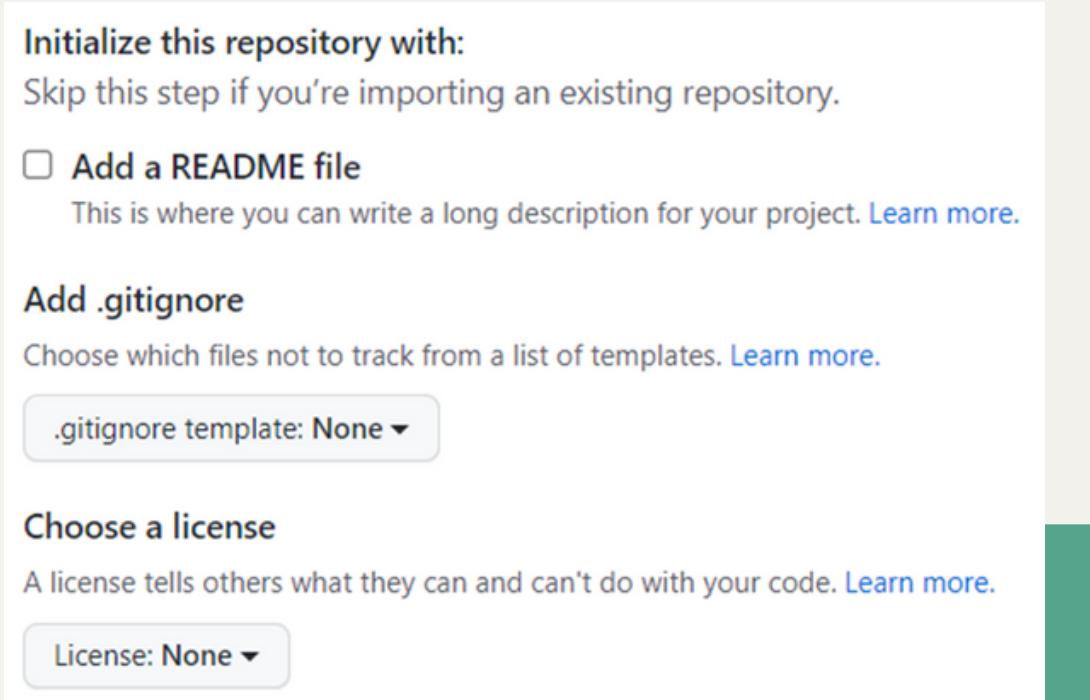
A screenshot of the GitHub "Create new repository" form. The "Owner" field shows a user icon and the username "ducquan2511". The "Repository name" field contains "Demo_Creat_Repo" with a green checkmark icon. Below the form, a message says "Great repository names are short and descriptive. Demo_Creat_Repo is available. How about reimaged-couscous?"

Bước 3: Chọn một trong hai tùy chọn do Github đưa ra. Hai tùy chọn có nội dung như sau:

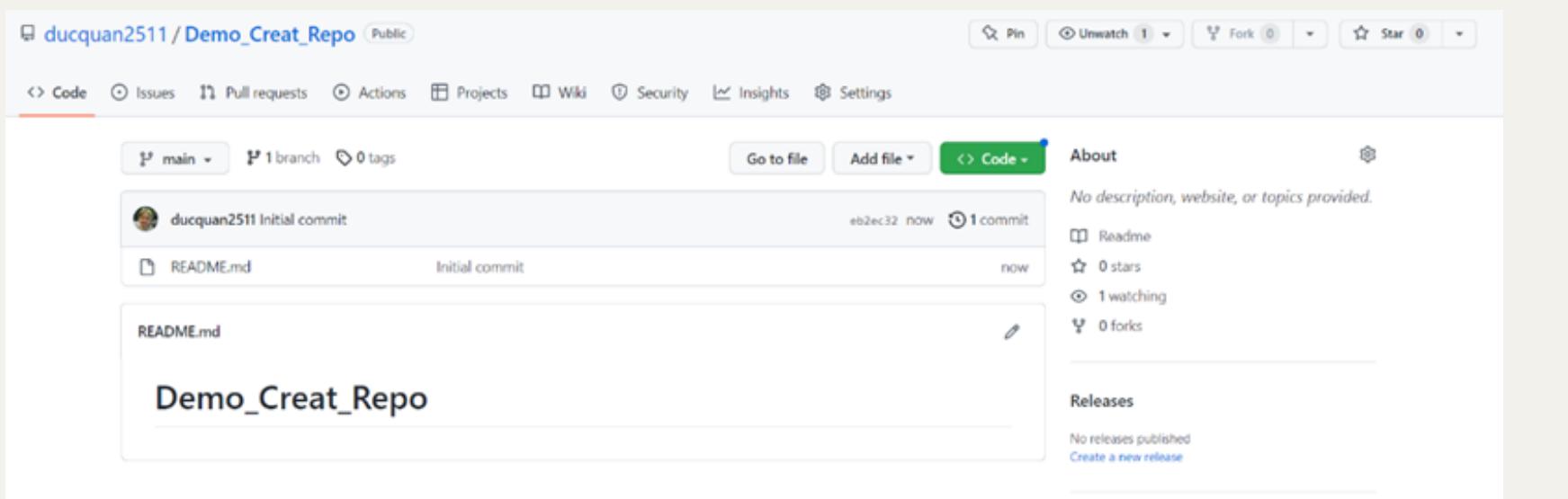
- Hai chế độ phân quyền là Public và Private. Public là chế độ mặc định, cho phép bất cứ ai cũng có thể xem được Repo. Nếu người tạo không muốn công khai có thể chuyển sang chế độ Private.



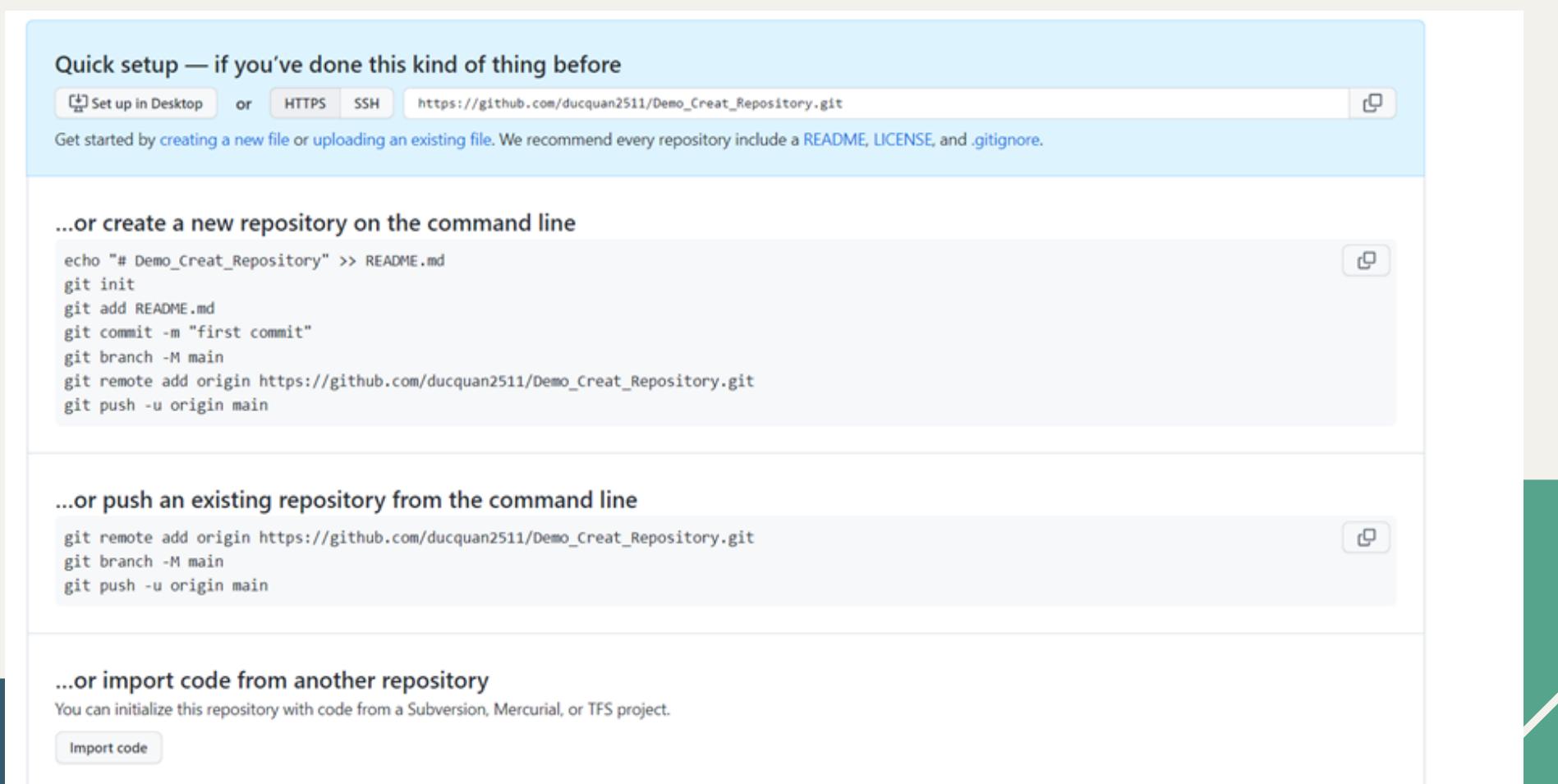
- Tập tin README giới thiệu Repo kèm một tệp tin .gitignore. Đây là một template có sẵn trong Github và cho mọi người tùy chọn



Bước 4: Bấm nút Create repository tạo Repo, Github sẽ chuyển trang Repo đã hoàn thành

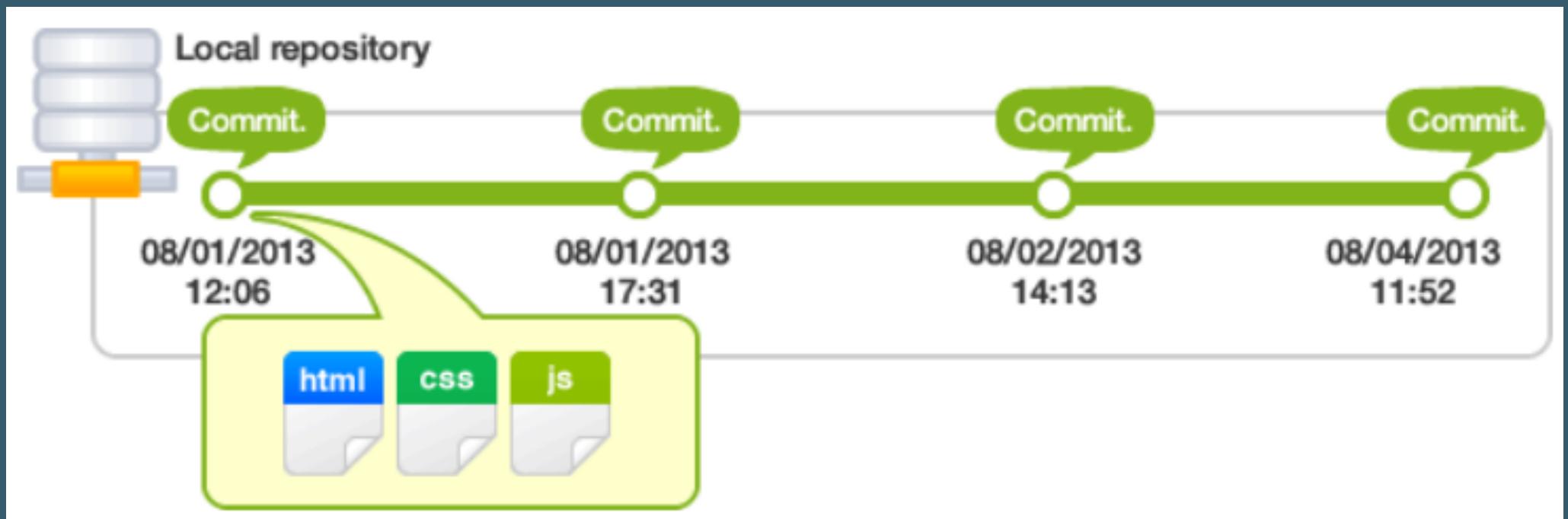


Trường hợp, người dùng không thêm tùy chọn, Github sẽ mặc định đây là người dùng mới với dịch vụ nó sẽ chuyển đến một trang hướng dẫn cho người dùng này.



Lệnh Commit

Lệnh git commit là lệnh cam kết lưu một ảnh chụp nhanh (snapshot) trong kho lưu trữ nội bộ (local repository) về các thay đổi code hiện tại trong dự án của bạn. Lệnh Git commit là một trong số các lệnh được sử dụng thường xuyên nhất trong Git.



Lệnh Pull

Cho biết những thay đổi trong Source Code, yêu cầu chủ sở hữu xem xét, hỗ trợ mọi người đóng góp công sức vào dự án. Tránh sự nhầm lẫn về lệnh PULL.

- Lệnh Pull Request: yêu cầu chủ sở hữu xem xét thay đổi trước khi gộp vào nhánh chính.
- Lệnh Pull: lệnh đơn thuần của Git để nâng cấp dữ liệu từ server về Local, có thể xảy ra xung đột.

Sử dụng GIT

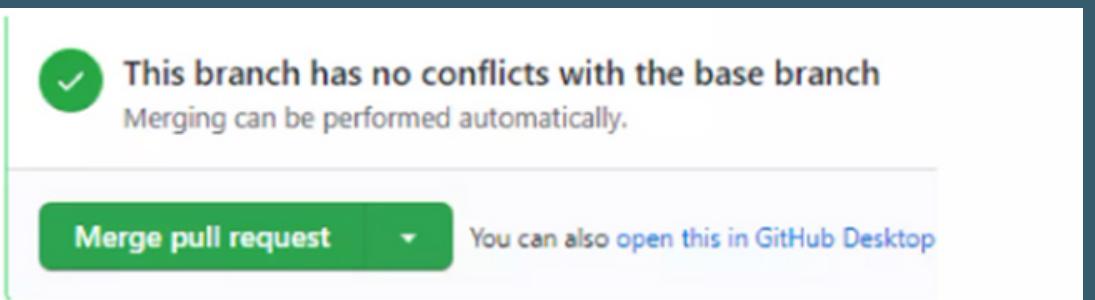
Github có đầy đủ dòng lệnh của Git, trong đó 3 câu lệnh cấu hình Github cơ bản nhất là: Commit, Pull, Merge. Ba câu lệnh này mô tả cả một quy trình làm việc của Github.



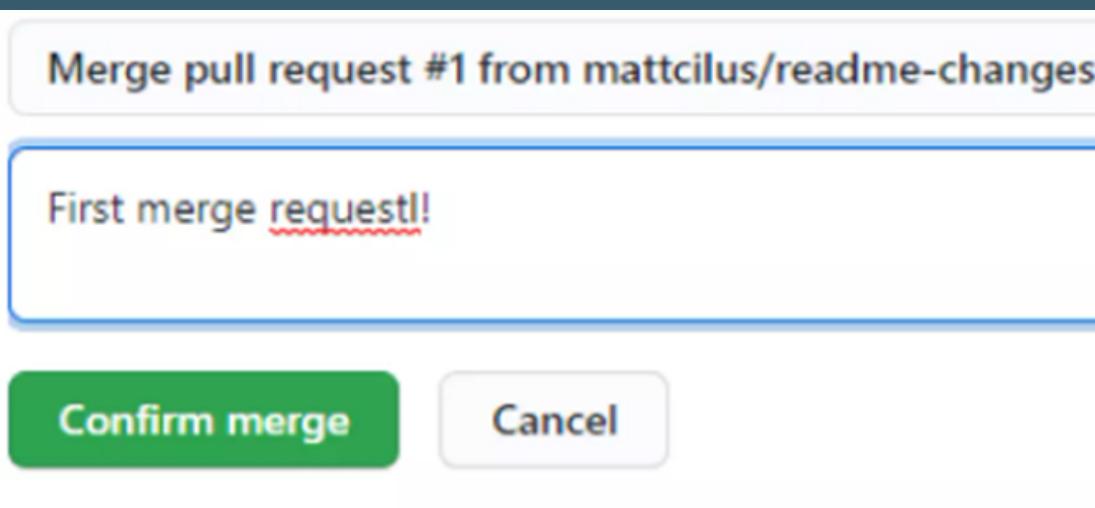
Lệnh Merge

Lệnh hợp nhất các nhánh trên Github trải qua các bước sau:

Bước 1: Bấm nút Merge pull request để hợp nhất những thay đổi vào nhánh Master



Bước 2: Xác nhận gộp nhánh Master và hoàn thành



Chương 4

CÁC LỆNH CƠ BẢN



git init : Khởi tạo một kho lưu trữ Git mới trong thư mục hiện tại.

git clone : Sao chép một kho lưu trữ Git từ một kho lưu trữ trực tuyến sang máy tính cá nhân của bạn

git add : Thêm các tệp đã sửa đổi hoặc tạo mới vào mục theo dõi của Git.

git commit: Tạo một "commit" mới để lưu trữ các thay đổi vào kho lưu trữ.

`git commit -m "message"`

`git commit -a`

`git commit --amend`

`git commit -S`

`git commit --allow-empty`

`git commit --squash`

git pull: Đẩy các commit mới lên một kho lưu trữ trực tuyến, chẳng hạn như GitHub hoặc Bitbucket.

- + Để liệt kê các kho chứa từ xa : **git remote -v**
- + Để liệt kê các nhánh của các kho chứa: **git branch -a**

git push: Lấy các commit mới từ kho lưu trữ trực tuyến và áp dụng chúng vào nhánh hiện tại.

git push: Lấy các commit mới từ kho lưu trữ trực tuyến và áp dụng chúng vào nhánh hiện tại.

git merge: Kết hợp các thay đổi từ một nhánh khác vào nhánh hiện tại.

git checkout: Chuyển đổi giữa các nhánh hoặc khôi phục các tệp đã bị xóa.

git status: Câu lệnh git status trong Git được sử dụng để xem trạng thái hiện tại của mục theo dõi của Git.

Chương 5: LÀM VIỆC với GIT

Phục hồi(RECOVERY), đánh dấu(TAG), nhánh, giải quyết xung đột trong nhánh



Phục hồi (recovery)

Thay đổi Commit cuối cùng:

Khi commit quá sớm và có thể quên thêm vào đó một số tập tin.
Muốn thực hiện lại commit đó, có thể chạy lệnh commit với tham số
`--amend: git commit --amend`

Phục hồi tập tin đã thay đổi:

Git status sẽ phục hồi lại những thay đổi trước đó-
phục hồi nó lại trạng thái giống như sau khi thực
hiện commit cuối hoặc như lúc mới đưa chúng vào thư
mục làm việc.

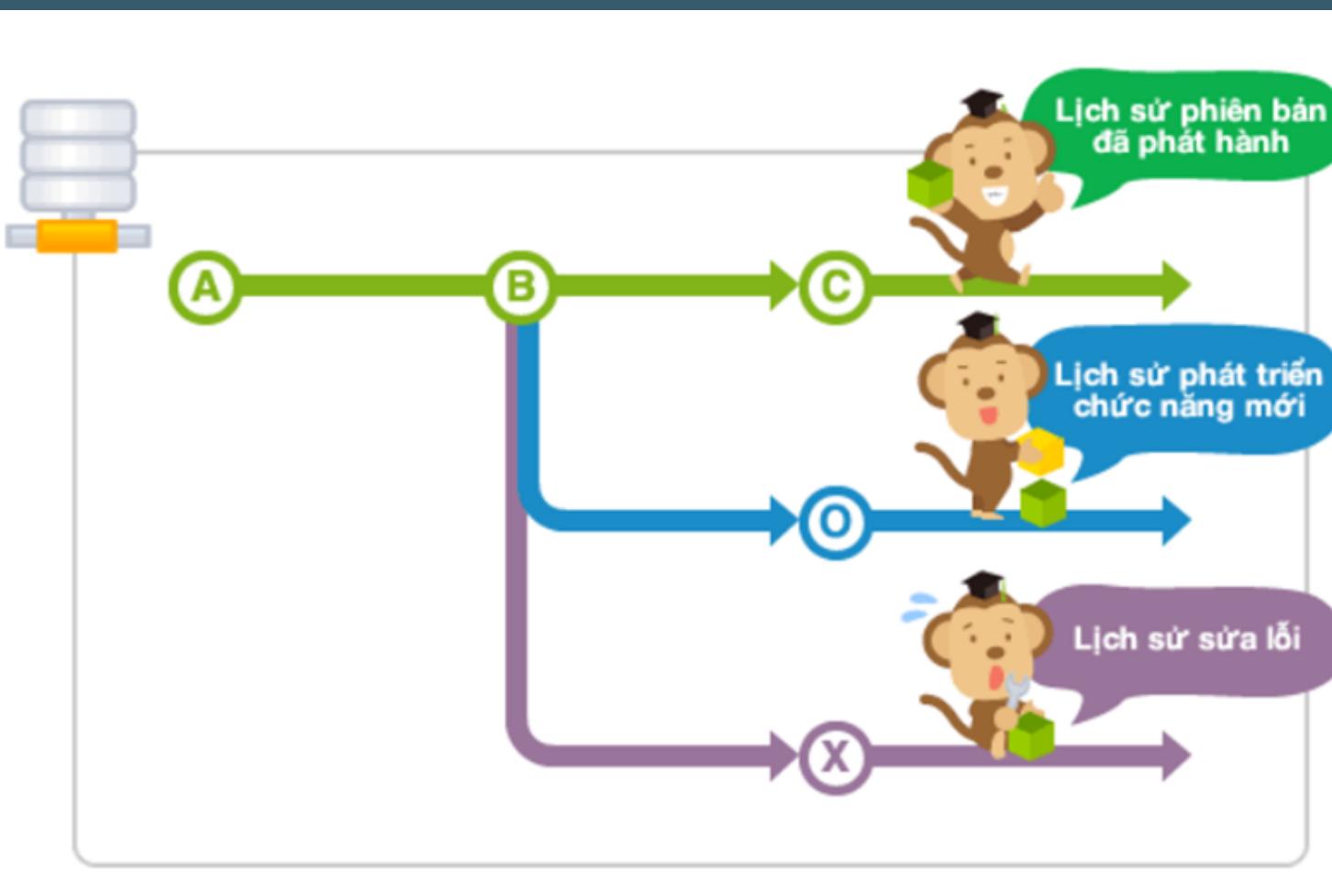
Đánh dấu (Tag)

Tag là một cái tên dùng để đánh dấu một điểm nào đó trong lịch sử quá trình commit khi cho rằng điểm đó là quan trọng, cần chú ý.

Liệt kê các Tag	git tag
Tạo ra một Tag mới đánh dấu vào commit cuối	<code>git tag -a -m</code>
Xem thông tin về commit được gắn tag	<code>git show tagname</code>
Cập nhật tag lên Remote	<code>git push origin tagname</code>
Xóa một tag (cần phải xóa ở local và remote)	<code>git push --delete origin tagname</code> <code>git tag -d tagname</code>

Nhánh

Trong Git nhánh branch là hướng rẽ phát triển code, mới mục đích không làm rối hướng phát triển chính, sau đó một nhánh có thể tích hợp vào nhánh chính. Nói chung trong Git luôn làm việc với các nhánh, mặc định bao giờ cũng có một nhánh chính tên là master



Quy trình phân chia nhánh

TẠO NHÁNH

Bước 1: Kiểm tra nhánh

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git branch
* master
```

Bước 2: Phân nhánh

Dùng lệnh git branch +<Tên nhánh>

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git branch b1
```

Bước 3: Kiểm tra nhánh đã tạo

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git branch
  b1
* master
```

PHÂN NHÁNH

Bước 1: Dùng lệnh git checkout+<tên nhánh chuyển>

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git checkout b1
Switched to branch 'b1'
```

Bước 2: Kiểm tra nhánh người dùng đang truy cập
Dùng lệnh git branch

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (b1)
$ git branch
* b1
  master
```

Dấu sao trước tên nhánh cho biết vị trí nhánh hiện tại

GỘP NHÁNH

Bước 1: Chuyển nhánh

Chuyển sang bước b1

Bước 2: Tạo file text có tên “nhanh.txt”

Bước 3: Thêm file “nhanh.txt” vào nhánh b1

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git checkout b1
Switched to branch 'b1'
```

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (b1)
$ cat > nhanh.txt
nhanh.txt
```

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (b1)
$ git add nhanh.txt
```

XÓA NHÁNH

Bước 1: Chuyển sang nhánh Master

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (b1)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Bước 2: Xóa nhánh b1

Dùng lệnh git branch -d+<tên nhánh xóa>

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git branch -d b1
Deleted branch b1 (was 59590bc).
```

Bước 3: Kiểm tra

Dùng lệnh git branch để xem nhánh hiện có

```
HP APPLE@HPAPPLE MINGW64 ~/Documents/Github/test (master)
$ git branch
* master
```

Quản lý các nhánh

lệnh git branch -v: xem commit mới nhất trên từng nhánh:

lệnh git branch -merged: xem nhánh nào đã được tích hợp vào nhánh hiện tại

lệnh git branch -no-merged: xem cách nhánh chứa các công việc/thay đổi chưa
được tích hợp vào

Quy trình phân chia nhánh

Lệnh "git merge"

Trong Git, hợp nhất(merge) là một thủ tục để kết nối lịch sử đã phân nhánh. Git merge sẽ liên kết một loạt các commit thành một lịch sử thống nhất.

Lệnh hợp nhất git như sau: **Git merge <query>**

- Để hợp nhất commit được chỉ định với nhánh hiện đang hoạt động:

Git merge < commit >

- Để hợp nhất các commit vào nhánh chính:

- + Sử dụng lệnh log để tìm id commit cụ thể:

Git log

- + Để hợp nhất các commit vào nhánh chính, hãy chuyển sang nhánh chính:

Git merge master

- Git merge nhánh:

Git cho phép gộp toàn bộ nhánh này vào nhánh khác.

XUNG ĐỘT KHI GIT MERGE

Khi hai nhánh đang cố gắng hợp nhất và cả hai đều được chỉnh sửa cùng một lúc và trong cùng một file, Git sẽ không thể xác định phiên bản nào sẽ thực hiện để thay đổi. Tình huống như vậy được gọi là xung đột hợp nhất. Nếu tình huống như vậy xảy ra, nó sẽ dừng ngay trước khi hợp nhất commit để có thể giải quyết xung đột theo cách thủ công.



Giải quyết xung đột

Để giải quyết xung đột, cần biết xung đột có xảy ra hay không và tại sao lại xảy ra sử dụng lệnh: **Git mergetool**

Trong kho lưu trữ, nó sẽ dẫn đến trạng thái file bị xung đột. Để giải quyết xung đột, vào chế độ insert bằng cách nhấn phím I. Nhấn phím Esc để thoát ra từ chế độ chèn. Nhập: w! ở cuối trình chỉnh sửa để lưu và thoát các thay đổi. Để chấp nhận các thay đổi, sử dụng lệnh rebase: git rebase -continue





**Thank
You**



Arowwai
Industries