Git là một hệ thống quản lý phiên bản phân tán được sử dụng để theo dõi các thay đổi trong mã nguồn trong quá trình phát triển phần mềm. Nó cho phép nhiều người cùng làm việc trên cùng một dự án và quản lý các phiên bản khác nhau của mã nguồn.

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/PhanManh2003/reactJS\_Fall24.git

git push -u origin main

- "Commit" trong quản lý mã nguồn (version control) không phải là một bức ảnh chụp toàn bộ mã nguồn, mà là một bản ghi lưu trữ các thay đổi cụ thể đã được thực hiện đối với mã nguồn.

- phải cấu hình thông tin cho người dùng git để gắn vào commit ( trước khi chạy lệnh commit):

$ git config user.name “your name”

$ git config user.email “your email”

- lệnh commit : $ git commit -m "first commit"

-m để thêm thông điệp của commit , "first commit" là nội dung của thông điệp

- lệnh branch: $ git branch –M main

-M để đổi tên, main là tên mới

- lệnh git remote add origin <https://github.com/PhanManh2003/reactJS_Fall24.git> để kết nối kho git cục bộ với kho git từ xa. Sau đó bạn có thể push mã nguồn từ kho cục bộ lên kho từ xa( github) HOẶC pull mã nguồn từ kho github về máy cá nhân.

- lệnh: git push -u origin main

+ git push để **đẩy các thay đổi từ kho Git cục bộ lên kho Git từ xa (remote repository)**. Các thay đổi mà bạn đã commit trong kho cục bộ sẽ được đẩy lên kho từ xa. Lệnh này giúp bạn đồng bộ mã nguồn giữa kho cục bộ và kho từ xa (ví dụ, trên GitHub, GitLab).

+ -u để lần sau chỉ cần đánh git push

=> Tóm lại: Git sẽ **đẩy nhánh main của kho cục bộ (local)** lên **nhánh main của kho từ xa (remote)** có tên là origin.

- lệnh git status:

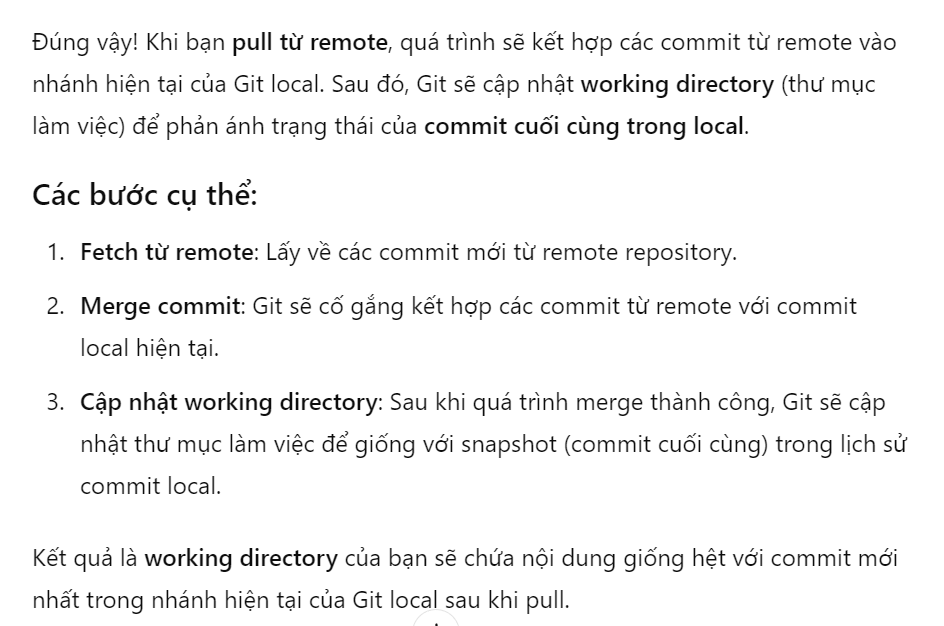
 **On branch main**: Bạn đang ở nhánh main.

 **No commits yet**: Chưa có commit nào trong kho Git mới khởi tạo.

 **Changes to be committed**: Các tệp index.html và style.css đã được thêm vào staging và sẵn sàng để commit.

 **Untracked files**: Tệp script.js chưa được thêm vào Git, bạn cần chạy git add script.js nếu muốn theo dõi nó trong kho Git. **Untracked files** trong Git là những tệp tin hiện có trong thư mục làm việc (working directory) của bạn nhưng **chưa được đưa vào stage** (staging area) và Git chưa theo dõi chúng.’

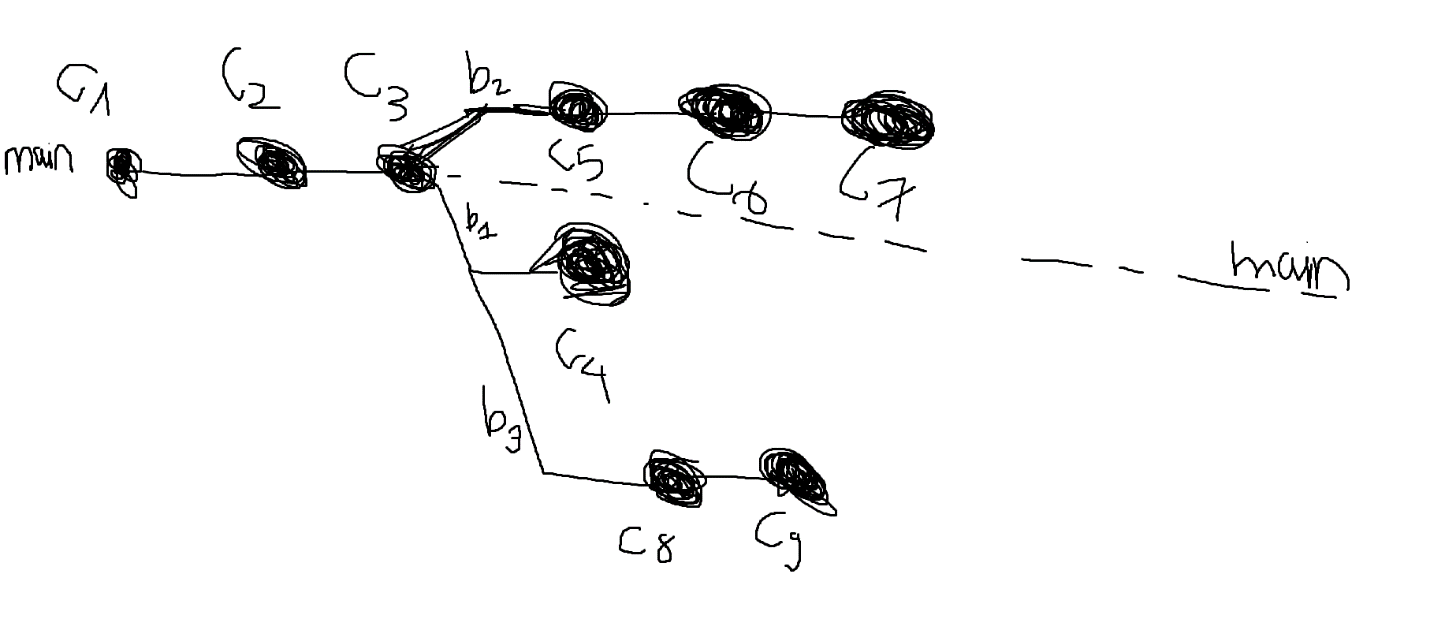
Ngoài ra còn có git pull , git checkout , git clone, git config credential.helper store ( để lưu account github vào máy tính để khi push từ git local lên nó ko hỏi )

****

### Tổng quan về branch:

* Khi bạn **tạo một nhánh mới**, Git tạo ra một con trỏ mới cho nhánh đó.
* **HEAD** trỏ đến nhánh mà bạn đang làm việc. Khi bạn tạo một nhánh mới và chuyển sang nhánh đó, HEAD sẽ được cập nhật để trỏ đến nhánh mới.

\*LÀM VIỆC NHÓM:



dự án tôi gồm 3 người. mỗi người làm 1 chức năng. Tôi đã commit c1 c2 c3 vô nhánh main để làm code cơ sở cho từng người phát triển dựa vào đó. Lúc này , tôi là người 1 tạo branch 1, người 2 tạo branch 2 (b2) , người 3 tạo branch 3 ( b3) để phát triển từng tính năng riêng của dự án mà tránh xung đột code. Tôi muốn hỏi như sau:

1. Tôi là người chủ kho github thì tôi có cần thiết phải tạo nhánh b1 không hay chỉ việc commit trực tiếp vào main còn 2 người kia tạo nhánh của họ thôi?

2. Cách tạo branch và chuyển sang nhánh đó như thế nào để commit cho các lần tiếp theo?

3. Người 2 có cần clone source code từ github tại commit 3 (c3) rồi mới tạo nhánh mới không? và làm sao để biết mình đang tạo nhánh từ commit nào của nhánh main ?

4. Sau khi mỗi người hoàn thành xong chức năng của họ thì tôi cần làm gì để merge đống code đó vào main hoàn chỉnh ?

5. Nếu người 2 , và 3 đang làm việc trên nhánh của họ, ở máy tôi thực hiện lệnh git pull ( trước đó đã là nhánh main ) thì nó chỉ pull code ở main về thôi phải ko ? Tóm lại là muốn pull nhánh nào thì phải chuyển sang nhánh đó trước phải không ?

=> ANSWER**:**

Dựa trên hình vẽ và mô tả về quy trình làm việc nhóm của bạn, tôi sẽ trả lời từng câu hỏi:

**1. Bạn có cần tạo nhánh b1 không hay commit trực tiếp vào main?**

* Việc bạn có cần tạo nhánh b1 hay không phụ thuộc vào cách quản lý dự án của nhóm. Tuy nhiên, **khuyến nghị tốt nhất** là **nên tạo nhánh riêng** để làm việc ngay cả khi bạn là người quản lý dự án. Lý do:
  + **Tránh xung đột trực tiếp trên nhánh main**: Mỗi khi phát triển một tính năng mới, bạn có thể phát hiện lỗi hoặc phải thử nghiệm. Nếu bạn commit trực tiếp vào main, rất dễ gây ra lỗi trong mã chung của dự án.
  + **Dễ dàng kiểm tra và code review**: Nếu bạn phát triển trên một nhánh riêng, khi bạn hoàn thành tính năng, nhóm có thể review và kiểm tra kỹ càng trước khi merge vào main.

**Tóm lại**: Bạn **nên tạo nhánh b1** cho mình, giống như các thành viên khác tạo nhánh b2 và b3 để phát triển tính năng riêng.

**2. Cách tạo branch và chuyển sang branch để commit lần tiếp theo**

* Để tạo một nhánh mới và chuyển sang nhánh đó, bạn làm theo các bước sau:

1. **Tạo nhánh**:

bash

Sao chép mã

git checkout -b <tên-nhánh>

Ví dụ, để tạo nhánh b1:

bash

Sao chép mã

git checkout -b b1

1. **Kiểm tra nhánh hiện tại**: Sau khi tạo nhánh, bạn đã tự động chuyển sang nhánh đó. Để kiểm tra nhánh hiện tại, dùng:

bash

Sao chép mã

git branch

1. **Commit trên nhánh mới**: Sau khi đã ở nhánh mới (như b1), bạn có thể commit những thay đổi của mình như bình thường:

bash

Sao chép mã

git add .

git commit -m "Miêu tả commit của bạn"

**3. Người 2 có cần clone source từ commit C3 để tạo nhánh không?**

* Người 2 **không cần phải clone lại từ đầu**. Sau khi người 2 đã clone repository lần đầu tiên, họ chỉ cần thực hiện các bước sau để đảm bảo họ bắt đầu từ commit C3 của nhánh main:

1. **Đảm bảo có commit mới nhất từ remote (nhánh main)**: Người 2 có thể dùng lệnh git fetch để cập nhật các commit mới nhất từ kho **remote** mà chưa áp dụng chúng vào nhánh local.

git fetch origin

Sau đó chuyển sang nhánh main và kéo các thay đổi mới nhất về:

git checkout main

git pull origin main  
  
  
\* Nếu muốn tạo nhánh từ commit C2 thì làm như sau :

git log --oneline => để biết mã hash của commit c2

git checkout -b <tên-nhánh-mới> <hash-of-C2>

1. **Tạo nhánh mới từ main**: Khi người 2 đã chắc chắn đang ở commit mới nhất của nhánh main, họ có thể tạo nhánh mới b2 từ commit C3:

git checkout -b b2

1. **Kiểm tra nhánh bắt nguồn từ commit nào**: Để biết mình đang tạo nhánh từ commit nào, người 2 có thể dùng lệnh:

git log --oneline

Lệnh git log --oneline trong Git được sử dụng để hiển thị lịch sử commit của một nhánh theo cách ngắn gọn và dễ đọc.

**4. Sau khi mọi người hoàn thành, làm sao để merge code vào nhánh main?**

Khi cả ba thành viên đã hoàn thành tính năng của mình và muốn merge code vào nhánh main, bạn sẽ cần thực hiện các bước sau để hợp nhất các thay đổi:

1. **Đảm bảo nhánh main đã cập nhật commit mới nhất**: Trước khi merge bất kỳ nhánh nào vào main, hãy đảm bảo nhánh main của bạn là mới nhất:

git checkout main

git pull origin main

1. **Merge từng nhánh vào main**: Giả sử người 1 merge nhánh b1, người 2 merge b2, người 3 merge b3:
   * Merge nhánh b1 vào main:

git merge b1

* + Nếu có xung đột (conflict), Git sẽ thông báo, và bạn cần chỉnh sửa các file bị xung đột, sau đó:

git add .

git commit

* + Làm tương tự với các nhánh b2 và b3:

git merge b2

git merge b3

1. **Push các thay đổi sau khi merge**: Sau khi đã merge các nhánh thành công vào main, bạn cần push nhánh main lên kho remote:

git push origin main

1. **Xóa các nhánh tạm nếu không cần thiết**: Nếu các nhánh tính năng (b1, b2, b3) không còn cần thiết, bạn có thể xóa chúng:

git branch -d b1

git branch -d b2

git branch -d b3

Vậy tóm lại:

* Bạn nên tạo nhánh b1 để làm việc riêng.
* Người 2 không cần clone lại, chỉ cần cập nhật source code và tạo nhánh từ nhánh main sau khi có commit C3.
* Khi tất cả hoàn thành, bạn merge từng nhánh vào nhánh main và push code lên kho remote.

**5. Khi bạn đang ở nhánh main và thực hiện lệnh git pull, nó chỉ pull code từ nhánh main về thôi phải không?**

* **Chính xác**: Nếu bạn đang ở nhánh main và thực hiện lệnh git pull, Git sẽ chỉ pull các thay đổi từ nhánh main của kho **remote** về nhánh main trên máy của bạn.

bash

Sao chép mã

git pull origin main

Lệnh này sẽ đảm bảo rằng nhánh main của bạn có tất cả các thay đổi mới nhất từ kho **remote** (cụ thể là nhánh main của remote).

**Tóm lại:**

* **Nếu bạn muốn pull từ nhánh nào**, bạn phải **chuyển sang nhánh đó trước**.
  + Ví dụ, nếu bạn muốn cập nhật nhánh b2 trên máy của mình từ kho **remote**, bạn cần:
    1. **Chuyển sang nhánh b2**:

bash

Sao chép mã

git checkout b2

* + 1. **Pull code từ kho remote cho nhánh b2**:

git pull origin b2

* Nếu bạn không chuyển sang nhánh b2 mà vẫn ở nhánh main, lệnh git pull sẽ chỉ cập nhật nhánh main.

**Lưu ý:**

* **git fetch**: Nếu bạn muốn chỉ tải về thông tin về tất cả các nhánh từ kho remote mà không hợp nhất (merge) ngay lập tức vào nhánh hiện tại, bạn có thể dùng lệnh git fetch:

git fetch origin

Lệnh này sẽ lấy tất cả các thay đổi từ remote, bao gồm cả các nhánh như b2, b3, nhưng không thay đổi mã nguồn của bạn. Sau đó, bạn có thể kiểm tra hoặc chuyển sang các nhánh cụ thể để xem xét và pull về nếu cần.

**6. tôi muốn git pull cả branch của người khác về dc ko ?**

Có, bạn hoàn toàn có thể **pull** về các nhánh của người khác từ **remote** về máy của mình. Tuy nhiên, bạn cần làm theo các bước sau để pull các nhánh đó về local, vì thông thường bạn chỉ pull các nhánh mà bạn đang làm việc (ví dụ như main hoặc nhánh riêng của bạn).

### Các bước để pull nhánh của người khác về máy:

1. **Lấy tất cả thông tin về các nhánh từ remote**:

Đầu tiên, bạn cần dùng lệnh git fetch để cập nhật thông tin về tất cả các nhánh trên **remote**:

git fetch origin

Lệnh này sẽ tải về thông tin về tất cả các nhánh trên remote mà bạn chưa có trên máy local của mình.

1. **Xem các nhánh từ remote**:

Sau khi fetch, bạn có thể xem tất cả các nhánh trên remote bằng lệnh:

git branch -r

Lệnh này sẽ hiển thị tất cả các nhánh hiện có trên **remote**, bao gồm các nhánh của người khác mà bạn muốn pull về.

1. **Checkout nhánh của người khác**:

Giả sử bạn muốn làm việc trên nhánh của người 2 có tên là b2 (hoặc nhánh nào đó của người khác), bạn cần checkout nhánh đó về local:

bash

Sao chép mã

git checkout b2

Nếu nhánh b2 chưa tồn tại trên local của bạn, lệnh git checkout sẽ tự động tạo một bản sao của nhánh b2 từ remote về local của bạn.

1. **Pull các thay đổi từ remote về nhánh local**:

Sau khi checkout thành công nhánh của người khác, bạn có thể pull các thay đổi từ remote về nhánh này như bình thường:

bash

Sao chép mã

git pull origin b2

Lệnh này sẽ đảm bảo rằng nhánh b2 của bạn là phiên bản mới nhất từ remote.

**7. tóm lại tránh tình trạng 2 lập trình viên cùng sửa 1 file thì lập trình viên A nên pull về trước khi commit thay đổi ở file đó.**