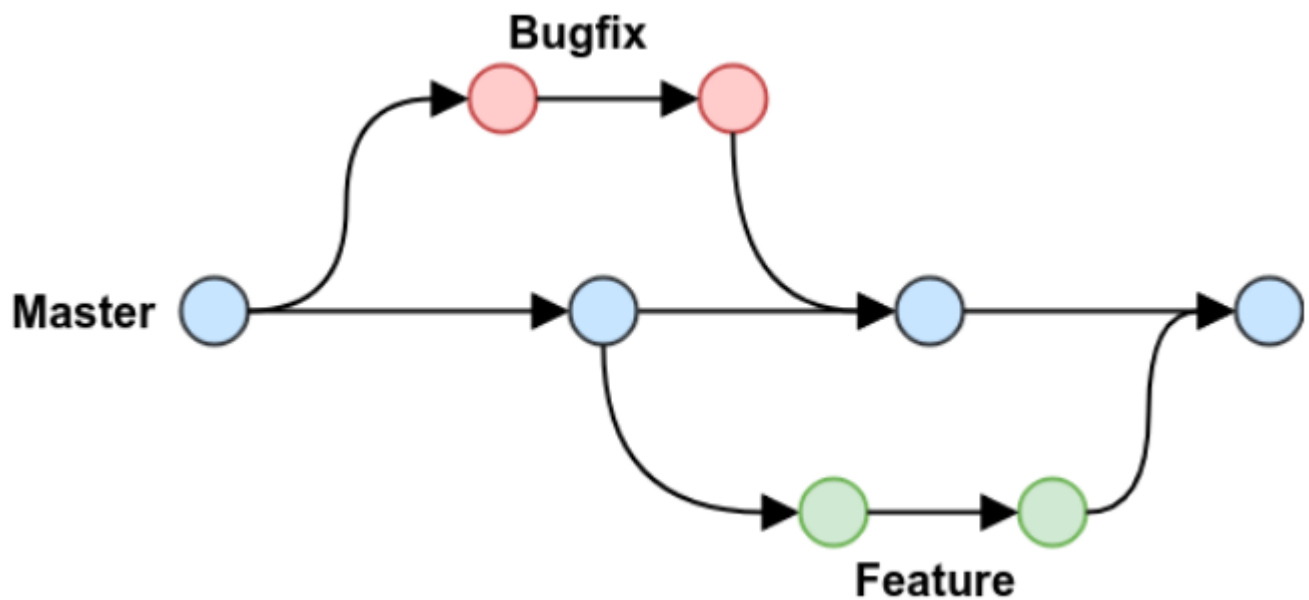
MÃ NGUỒN MỞ TRONG KHOA HỌC DỮ LIỆU

Bài 01.
QUẢN LÝ MÃ NGUỒN
Git / Github (tiếp theo)

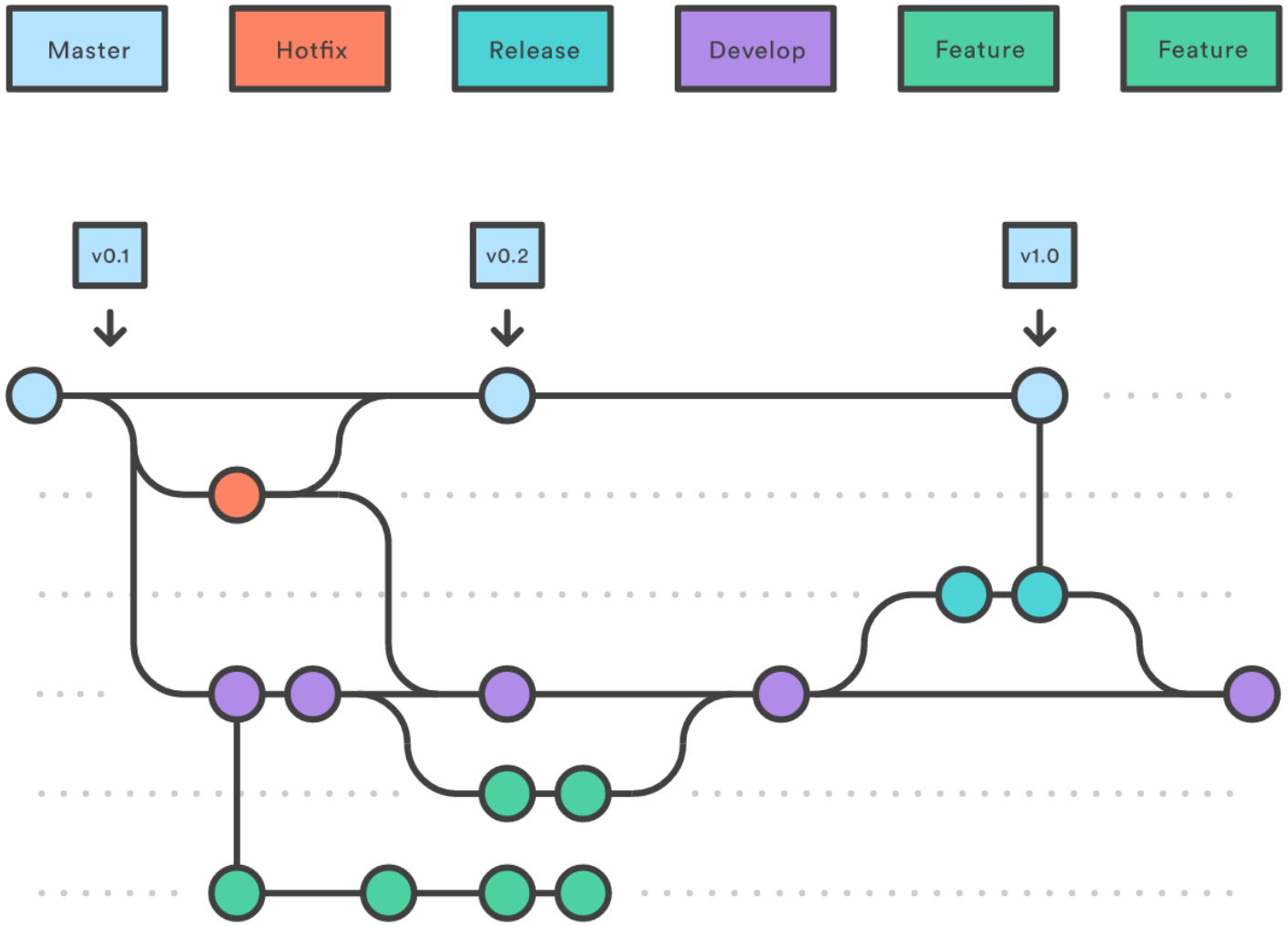
ThS. Lê Nhật Tùng

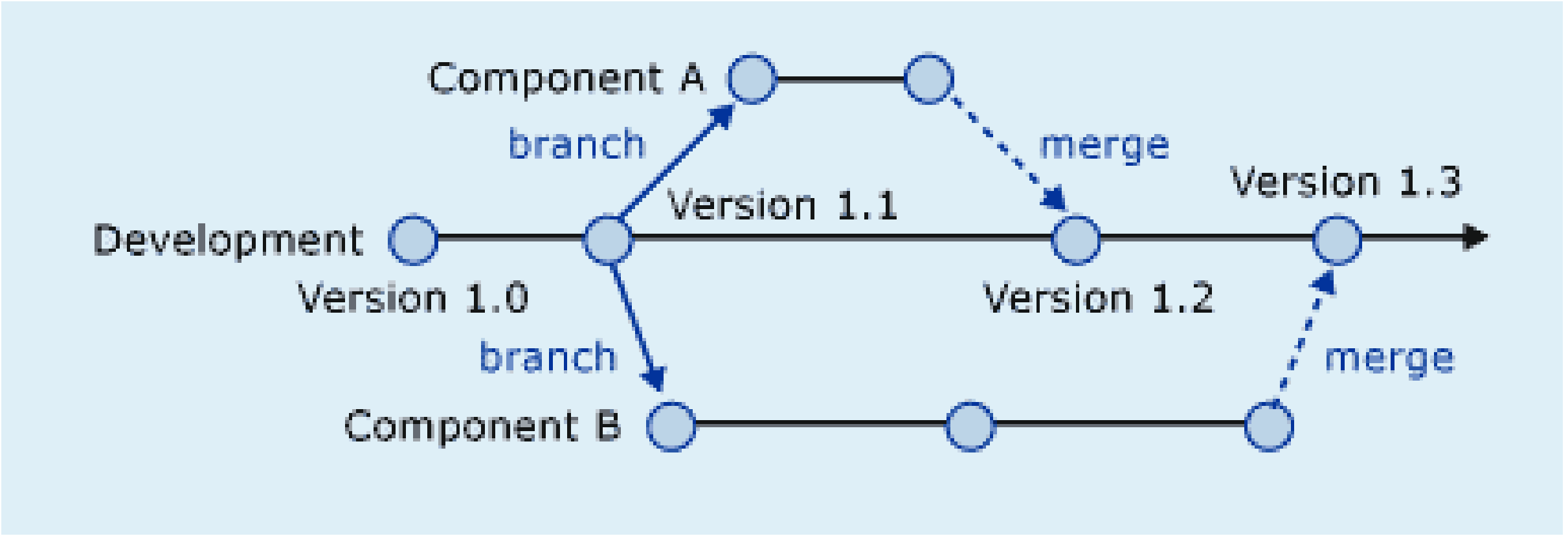

Bugfix
Master
Feature

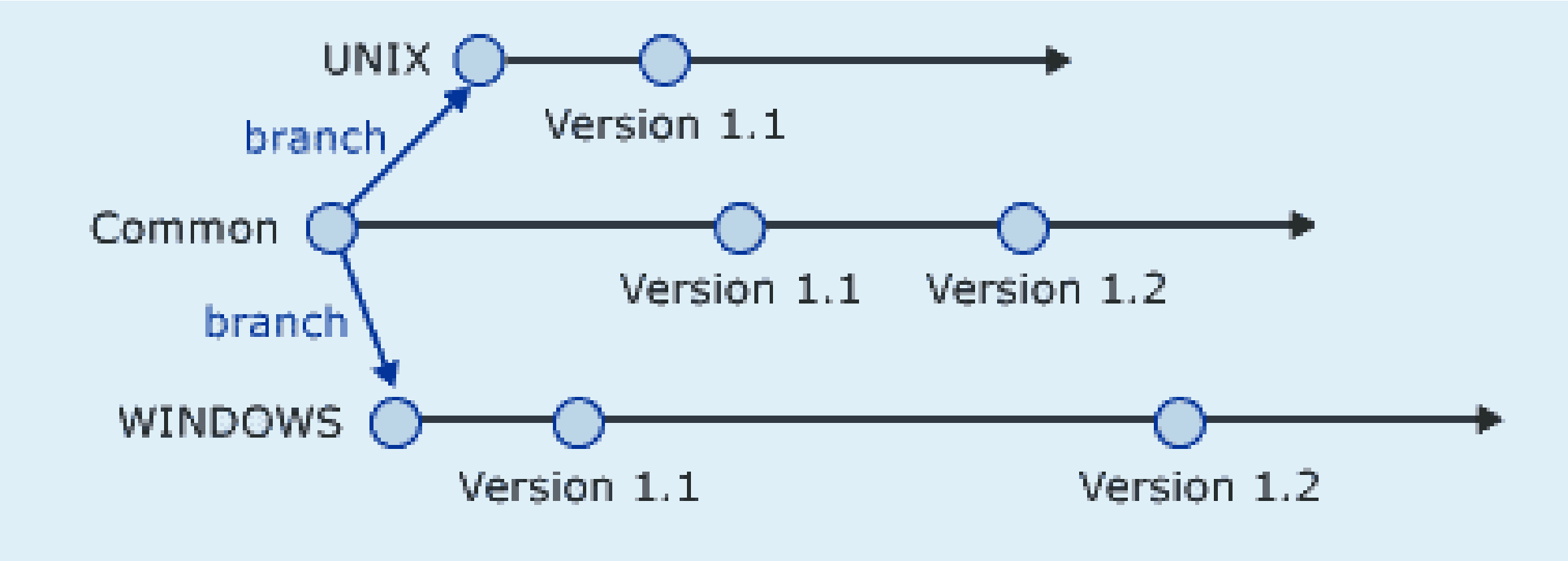
Lợi ích của việc sử dụng nhánh:

- Nhánh chính giữ lịch sử phát hành sạch sẽ
- Có không gian để thử nghiệm mà không làm ảnh hưởng đến hệ thống chính
- Nhiều người có thể làm việc cùng lúc trên các nhánh khác nhau
- Dễ dàng quay lại công việc trên một tính năng và tiếp tục với nhánh chính

Các nhánh có thể có các nhánh con




Component A
branch
merge
Version 1.3
Version 1.1
Development
Version 1.0
Version 1.2
branch
merge
Component B



Software Branching and Parallel Universes (codinghorror.com)

Nhánh là gì?

- Nhánh là một chỉ báo (indicator) đến một commit cụ thể trong lịch sử của dự án. Khi bạn tạo một nhánh mới, Git chỉ đơn giản tạo một con trỏ trỏ đến commit hiện tại, sau đó bạn có thể tiếp tục làm việc trên nhánh mới mà không ảnh hưởng đến các nhánh khác.
- Nhánh chính (master/main): Đây là nhánh chính của dự án, nơi lưu trữ mã nguồn đã được kiểm tra và ổn định để phát hành. Trong nhiều dự án, nhánh chính hiện nay thường được gọi là main thay vì master.

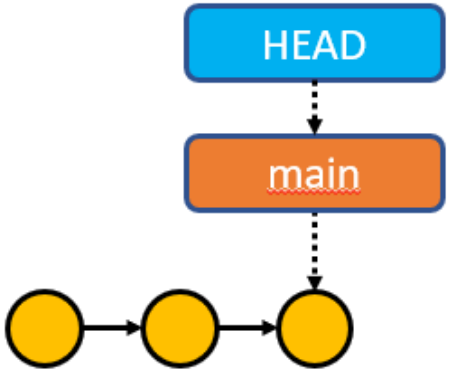
Cách tạo một nhánh mới

- Để tạo nhánh mới trong Git, sử dụng lệnh sau:

git branch <tên-nhánh>

- Sau khi tạo nhánh, bạn sẽ có một chỉ báo mới trỏ đến commit hiện tại. Lúc này, nhánh này có lịch sử giống với nhánh bạn đang ở, nhưng khi bắt đầu làm việc trên nhánh mới, bạn có thể thực hiện các thay đổi mà không ảnh hưởng đến nhánh chính

Chúng ta bắt đầu với một nhánh: main

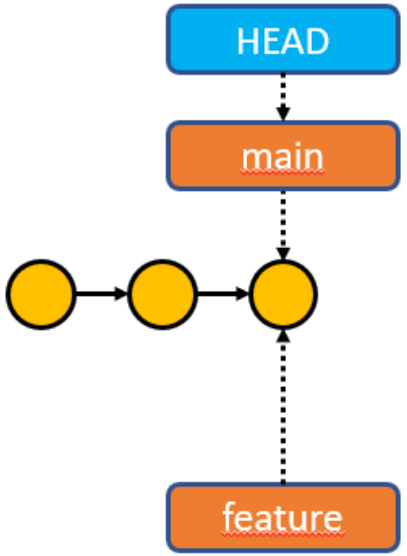


HEAD là một con trỏ đặc biệt chỉ vào commit hiện tại mà bạn đang làm việc. Nói cách khác, nó cho biết "bạn đang ở đâu" trong lịch sử commi

Tạo nhánh mới

- Chúng ta tạo một nhánh mới mà chúng ta đặt tên là "feature" bằng lệnh:

git branch **feature**

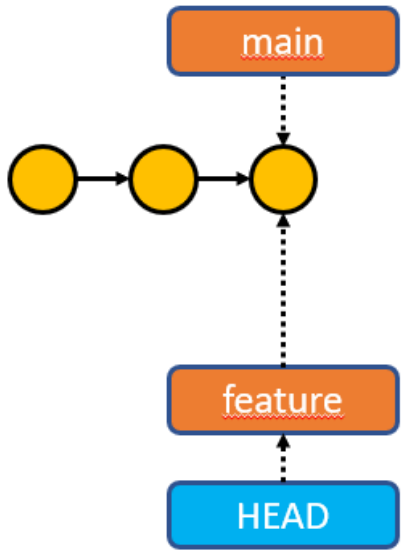


Điều này tạo ra một chỉ báo mới trỏ đến vị trí hiện tại của chúng ta. Tuy nhiên, HEAD vẫn trỏ đến nhánh main.

Chuyển sang nhánh mới

Chúng ta sử dụng lệnh sau để chuyển sang nhánh "feature" mới:

git checkout feature



Chỉ báo HEAD sẽ chuyển đến nhánh feature.

Bài tập 1

- Tạo thư mục mới RR_git4
- Thực hiện các lệnh sau đây:

mkdir RR_git4

cd RR_git4

git init CentralRepo –bare

git clone CentralRepo Dev1

git clone CentralRepo Dev2

Bài tập 1

- Tạo một lịch sử ngắn về các commit (tối thiểu hai) và đẩy nó lên dev1

```
cd dev1
echo "a first file" > file1.txt
git add .
git commit -m "Added a first file"
echo "a second file" > file2.txt
git add .
git commit -m "Added a second file"
git push
```

Bài tập 1

- Kéo thông tin về dev2

  cd ../dev2

  git pull

Bài tập 1

- Kéo thông tin về dev2

  cd ../dev2

  git pull

Bài tập 2

- Tạo nhánh mới trong dev1

cd ../dev1

git branch **feature1**

Bài tập 2

- Để liệt kê các nhánh trong Git, bạn có thể sử dụng các lệnh sau:
  - Liệt kê nhánh cục bộ: **git branch**
  - Liệt kê tất cả nhánh (cục bộ và từ xa): **git branch -a**
  - Liệt kê chỉ nhánh từ xa: **git branch -r**

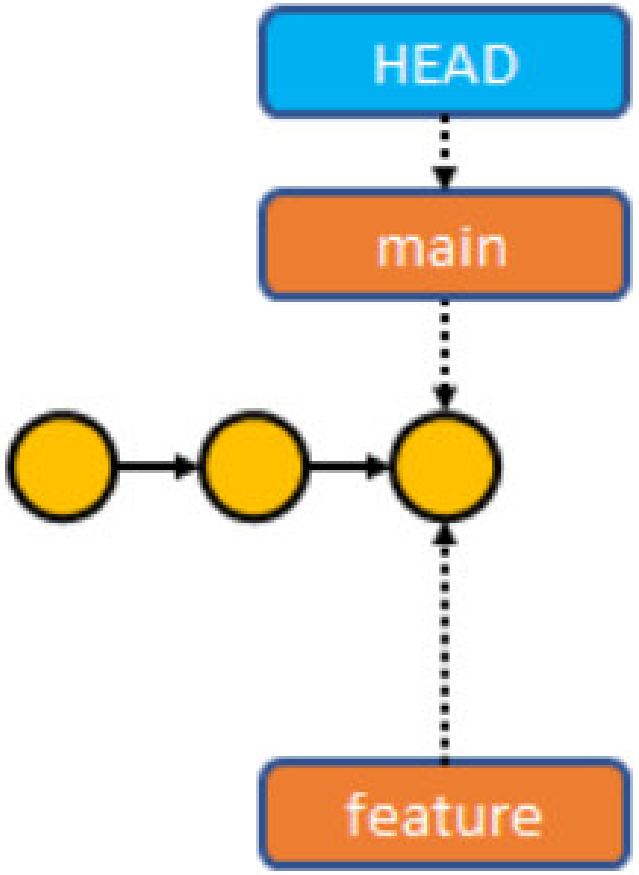
Bài tập 2

- Để chuyển sang nhánh mới trong Git, bạn có thể sử dụng lệnh sau:
  - git checkout <branch-name>
  - **git checkout feature1**
- Nếu bạn muốn tạo và chuyển sang nhánh mới trong một bước, bạn có thể sử dụng:
  - git checkout -b <new-branch-name>

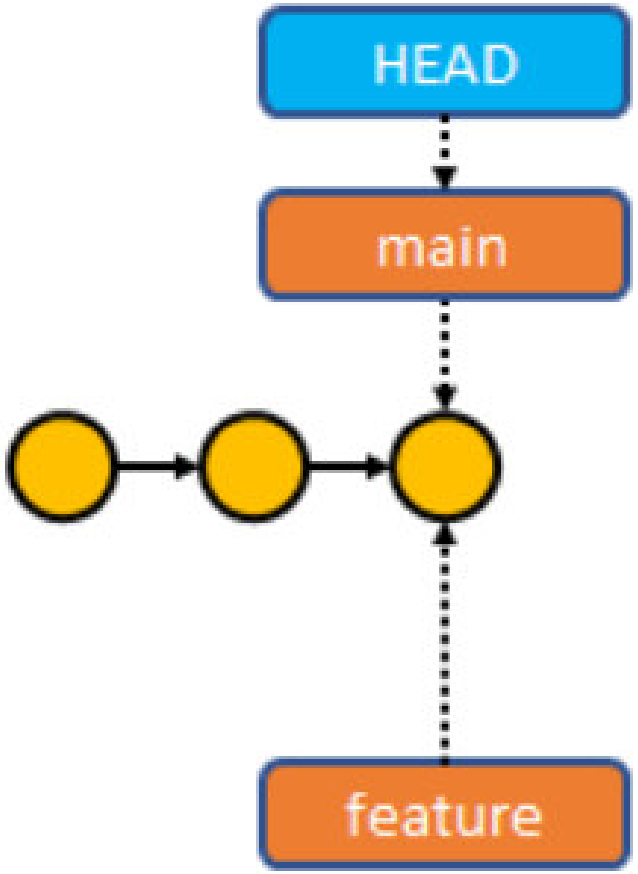
Quy trình làm việc mới là gì?

- Bạn chỉ có thể thêm các cam kết vào một trong các nhánh tại một thời điểm.
- Khi bạn chuyển giữa các nhánh, Git sẽ 'làm mới' khu vực staging và thư mục làm việc của bạn để khớp với .HEAD.

Hãy bắt đầu với một lịch sử cam kết đơn giản:

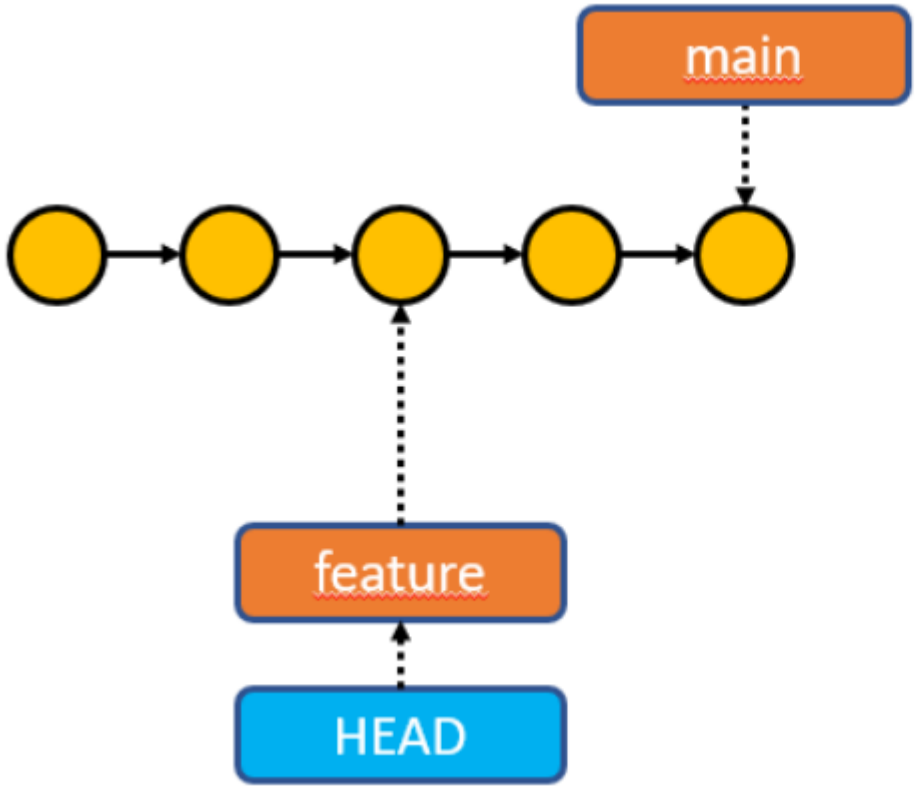


Các cam kết mới chỉ ảnh hưởng đến nhánh hiện tại.

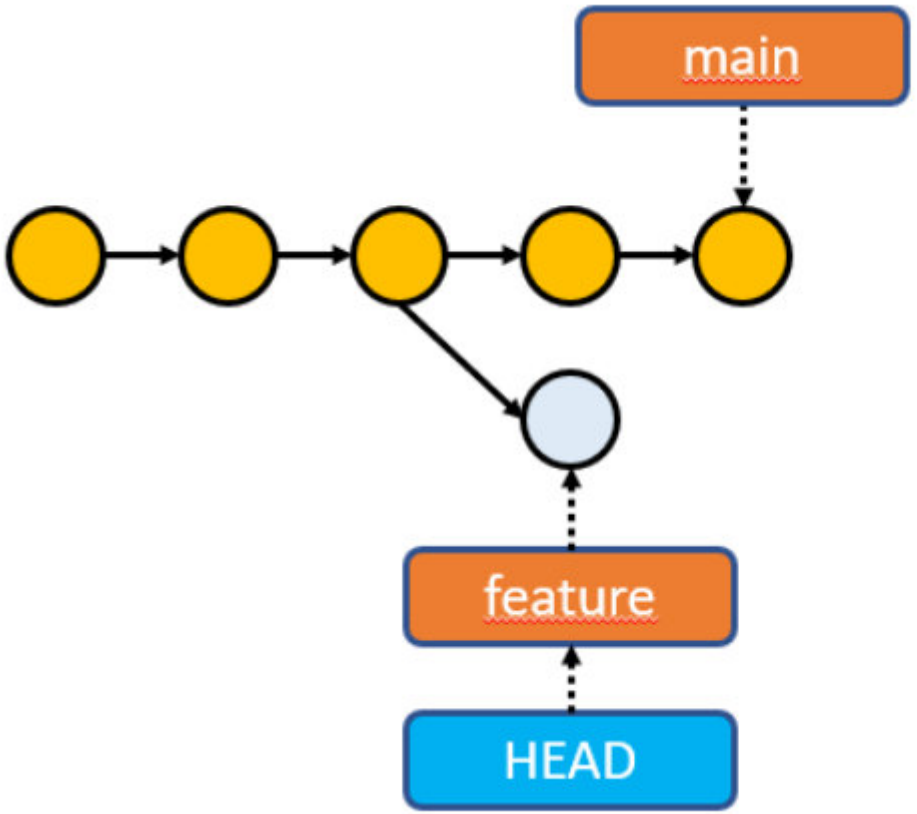


Trong một dự án lớn, ví dụ như nếu bạn là người làm việc trên nhánh chính, thì các thay đổi có thể không phải là của bạn (các người khác có thể đẩy các tính năng/sửa lỗi của họ, v.v. trong thời gian đó).

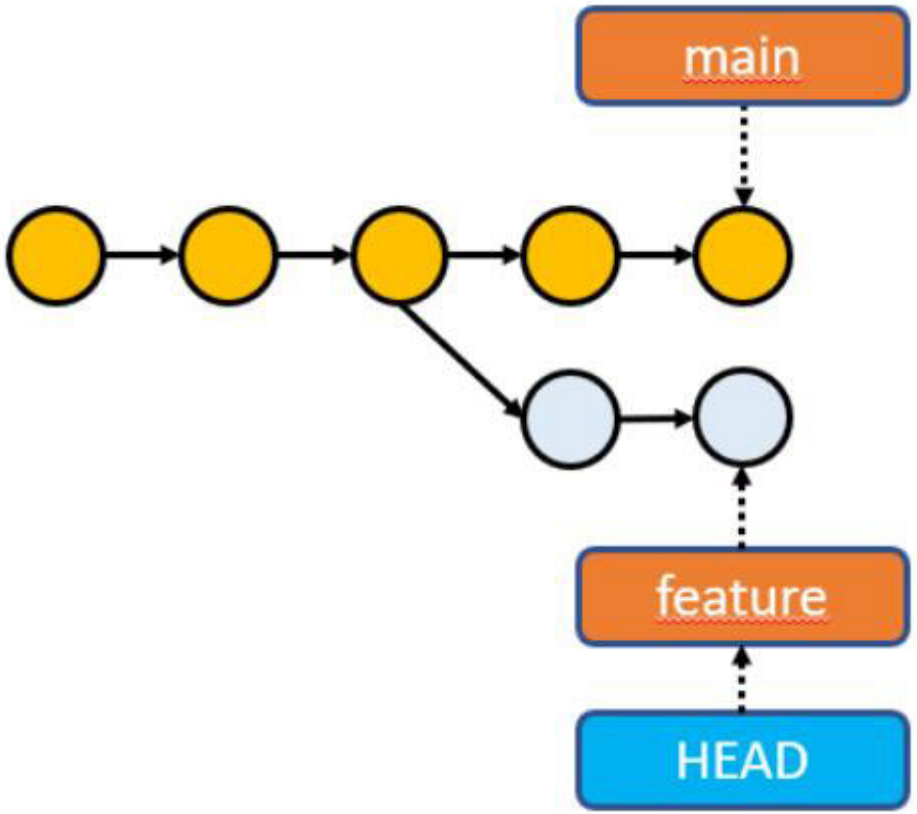
Hãy chuyển sang nhánh feature
git checkout feature



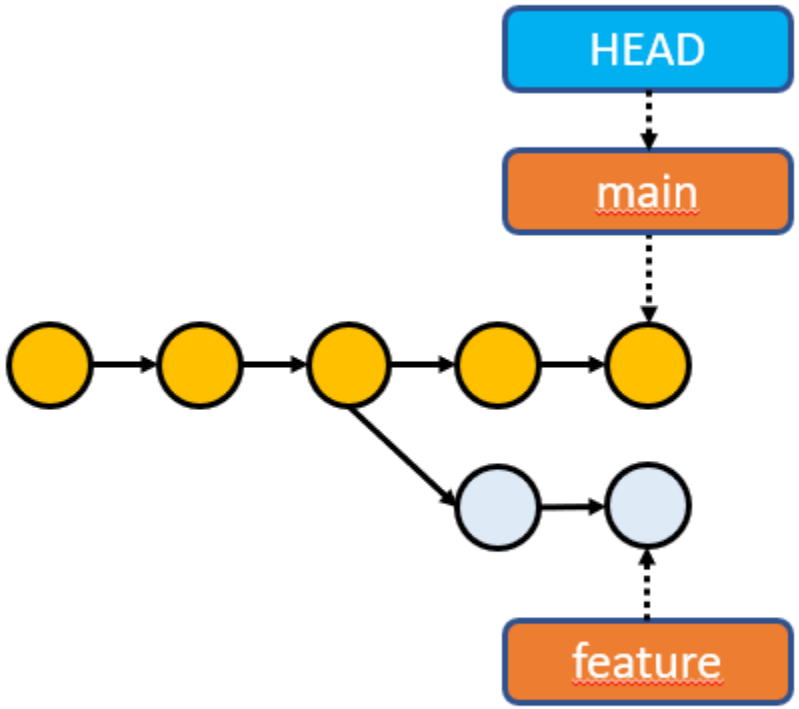
Và thêm một số commit



Thêm một commit



- Tại bất kỳ thời điểm nào, bạn có thể quay lại nhánh chính bằng lệnh: **git checkout main**



Bài tập 3

- Là Dev1, trong khi ở nhánh feature1, hãy thêm một tệp, ghi lại và cam kết nó. Sau đó thực hiện lệnh đẩy (push).

```
echo "some text" > file_on_feature1.txt
git add .
git commit -m "First file on feature1 branch"
git push

Read the message and follow the hints.
git push --set-upstream origin feature1
```

Bài tập 3

- Chuyển lại về nhánh chính và chú ý những gì đang xảy ra trong thư mục làm việc của bạn.

ls

git checkout main

ls

(or look at it in Explorer/Finder)

Bài tập 3

- Thêm một tệp (không cam kết nó) và thử chuyển nhánh.

```
echo "some other text" > some_file.txt
git checkout feature1
```

Bài tập 3

- Là Dev2, thực hiện lệnh:

```
cd ../Dev2
git pull
git branch -l
```

Bài tập 3

- Thực hiện và kiểm tra lại

```
git checkout feature1
git branch -l
```

Bài tập 3

- Thực hiện và kiểm tra lại

```
git checkout feature1
git branch -l
```

Bài tập 4

- Để Dev2 thực hiện một số công việc, tạo một cam kết mới trên nhánh chính (main

```
git checkout main
echo "another text" > file_dev2main.txt
git add .
git commit -m "A file added to main by dev2"
```

Bài tập 4

- Tạo một cam kết mới trên nhánh feature1

```
git checkout feature1
echo "another text" > file_dev2feature.txt
git add .
git commit -m "A file added to feature1 by dev2"
```

Bài tập 4

- Chạy lệnh trên nhánh chính, đẩy các thay đổi:

```
git checkout main
git status
git push
git status
```

Bài tập 4

- Chạy lệnh trên nhánh feature1, đẩy các thay đổi:

  git checkout feature1

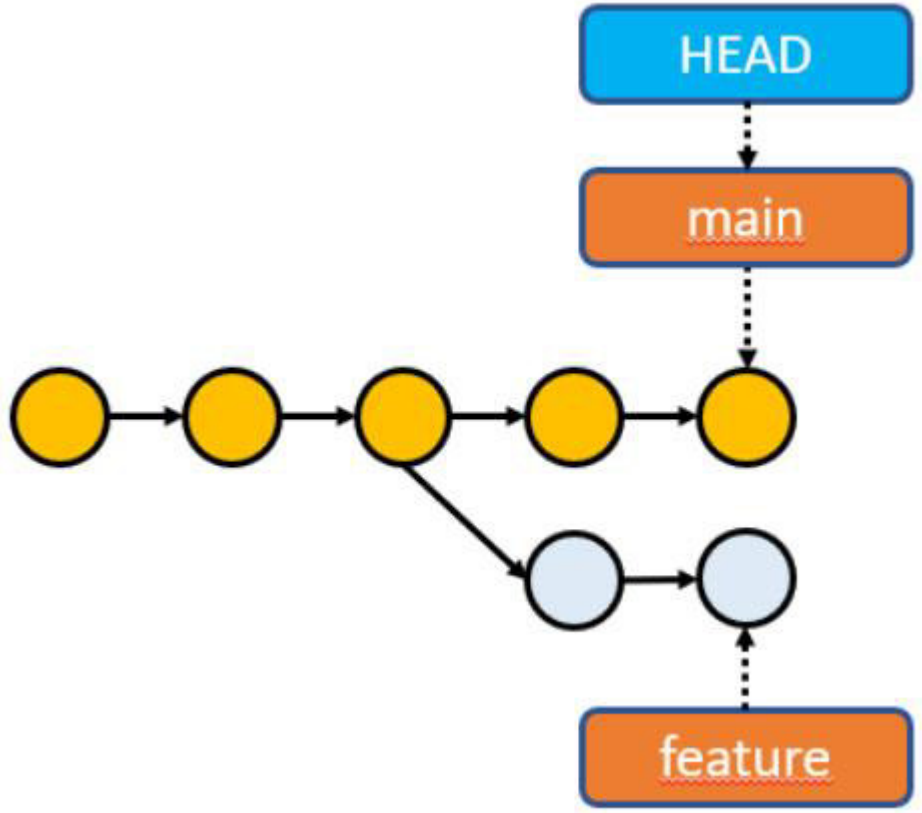
  git status

  git push

  git status

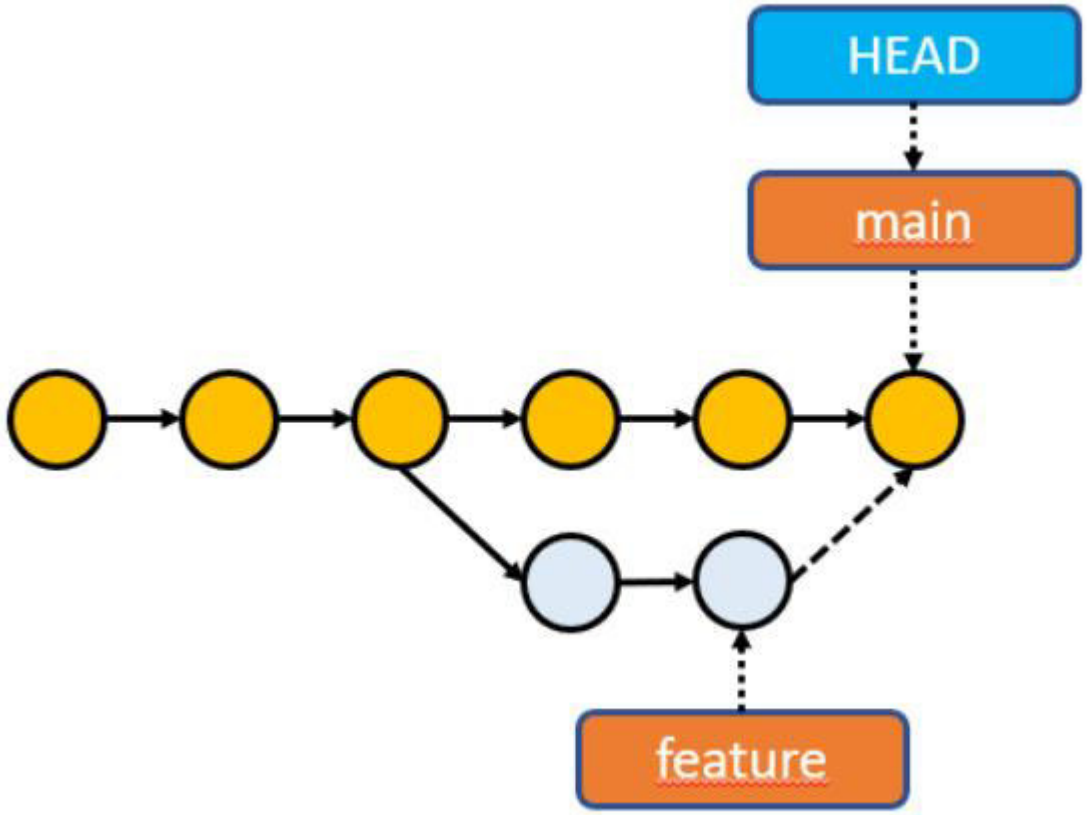
Có hai cách chính để gộp các kết quả lại với nhau:

- Bạn có thể nhớ cách mà chúng ta gộp lại với nhánh chính bằng lệnh:
- **git merge origin/main**



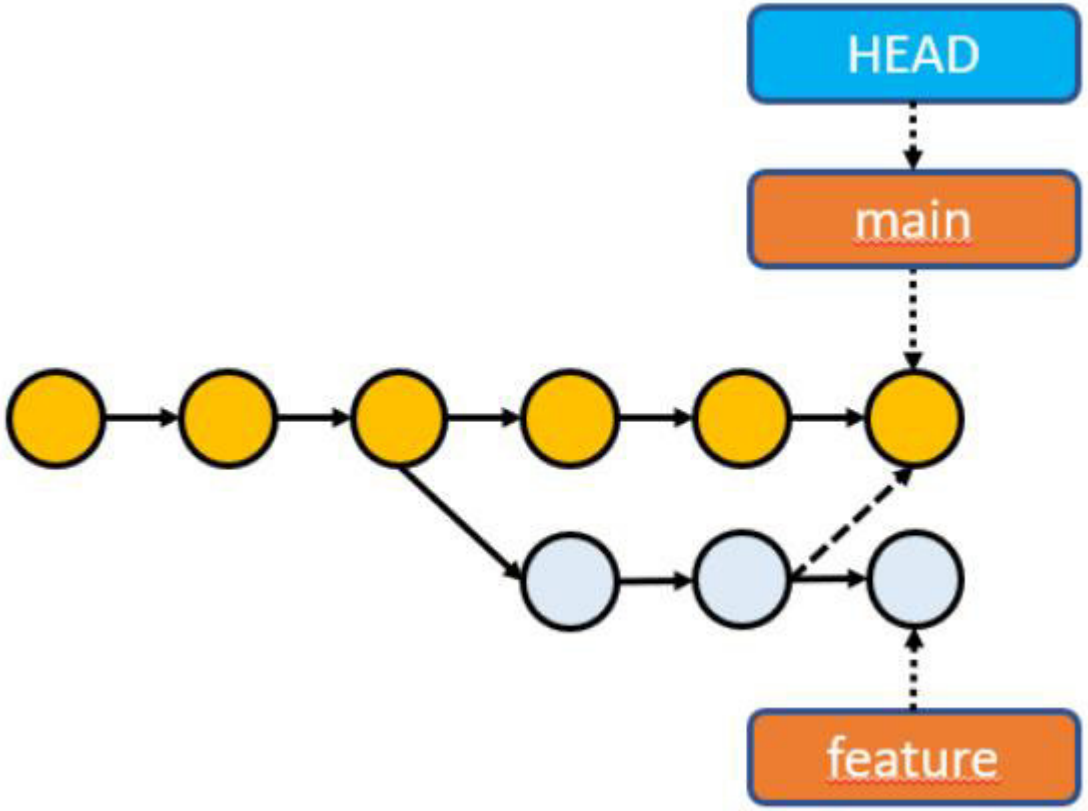
Trong khi ở nhánh chính (main), hãy thực hiện lệnh:

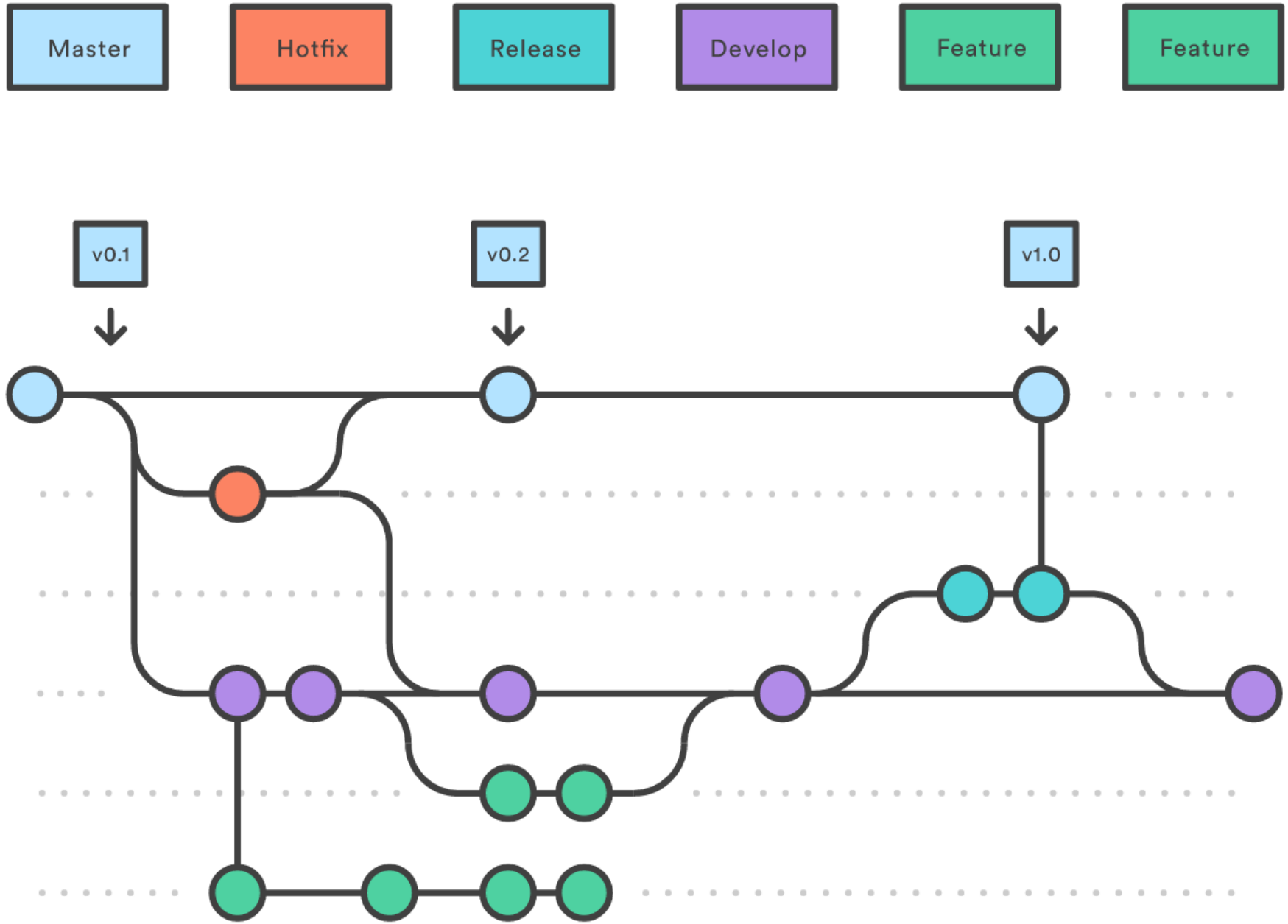
- git merge feature
- Lưu ý commit sau khi merge



Bạn vẫn có thể làm việc trên nhánh feature

- Nó không bao gồm việc hợp nhất (trừ khi bạn cũng thực hiện khi ở trên nhánh):
- git merge main




Master
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0

Bài tập 5

- Dev1 nhận được tất cả các cập nhật mới nhất cho cả hai nhánh.

```
cd ../dev1
git pull
git checkout main
git pull
```

Bài tập 5

- Trộn các nhánh

  git checkout main

  git merge feature1

Bài tập 5

- Trộn các nhánh

  git checkout main

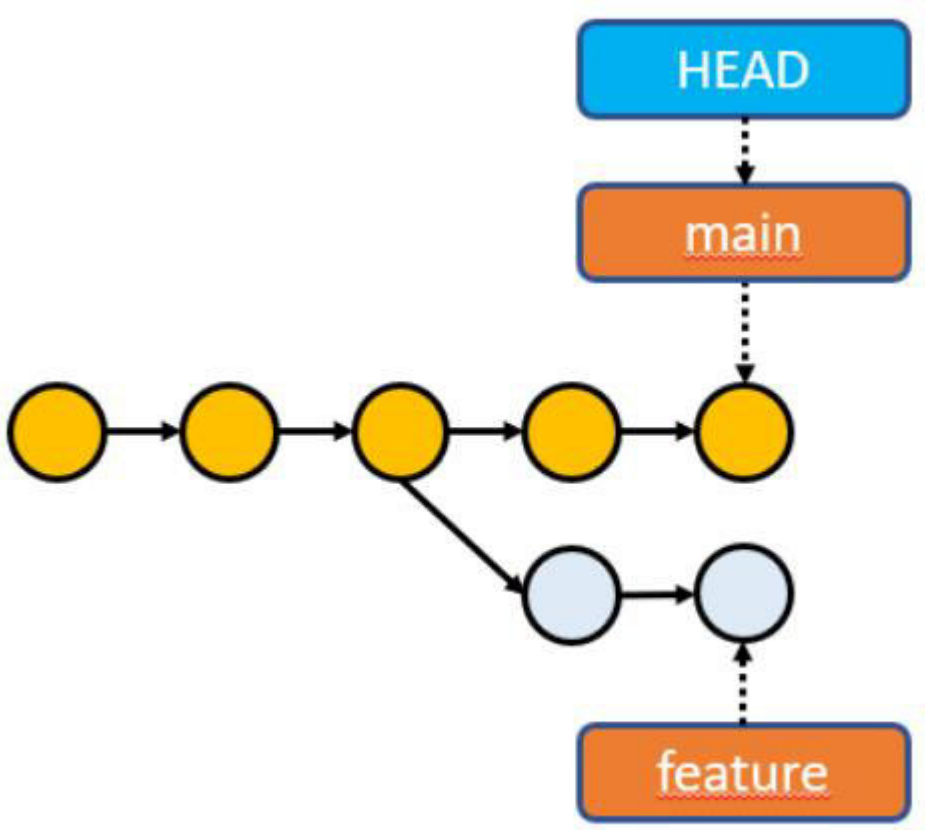
  git merge feature1

  git push

  Giải quyết đụng độ nếu có

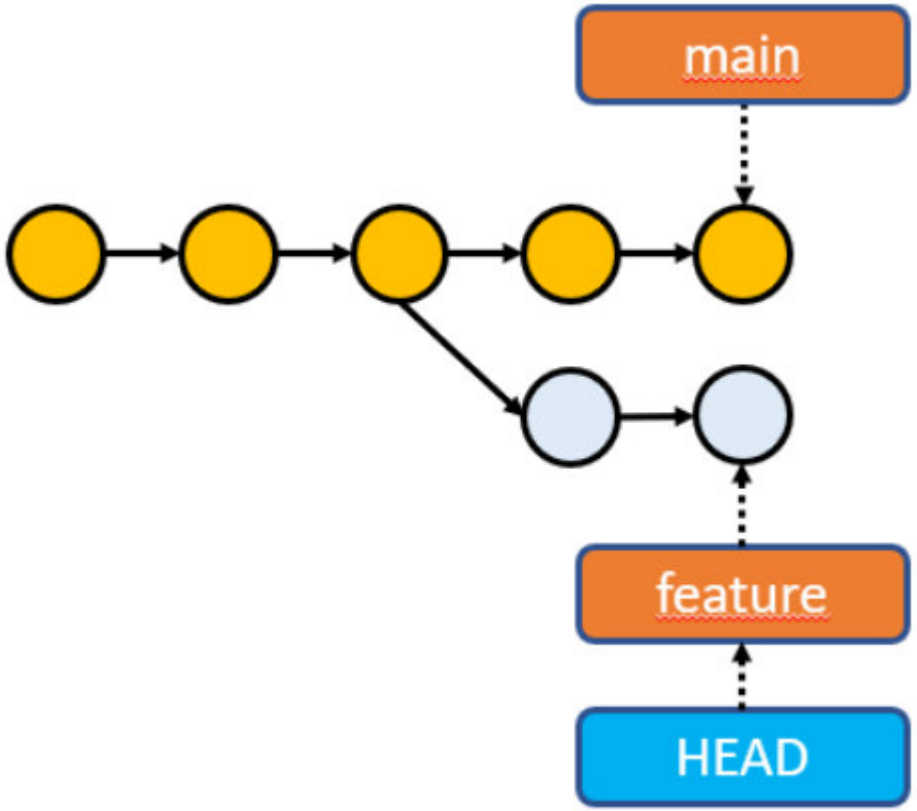
Cách tiếp cận thứ hai là rebase.

- Rebase là làm cho nhánh của bạn bắt đầu từ một điểm khác.

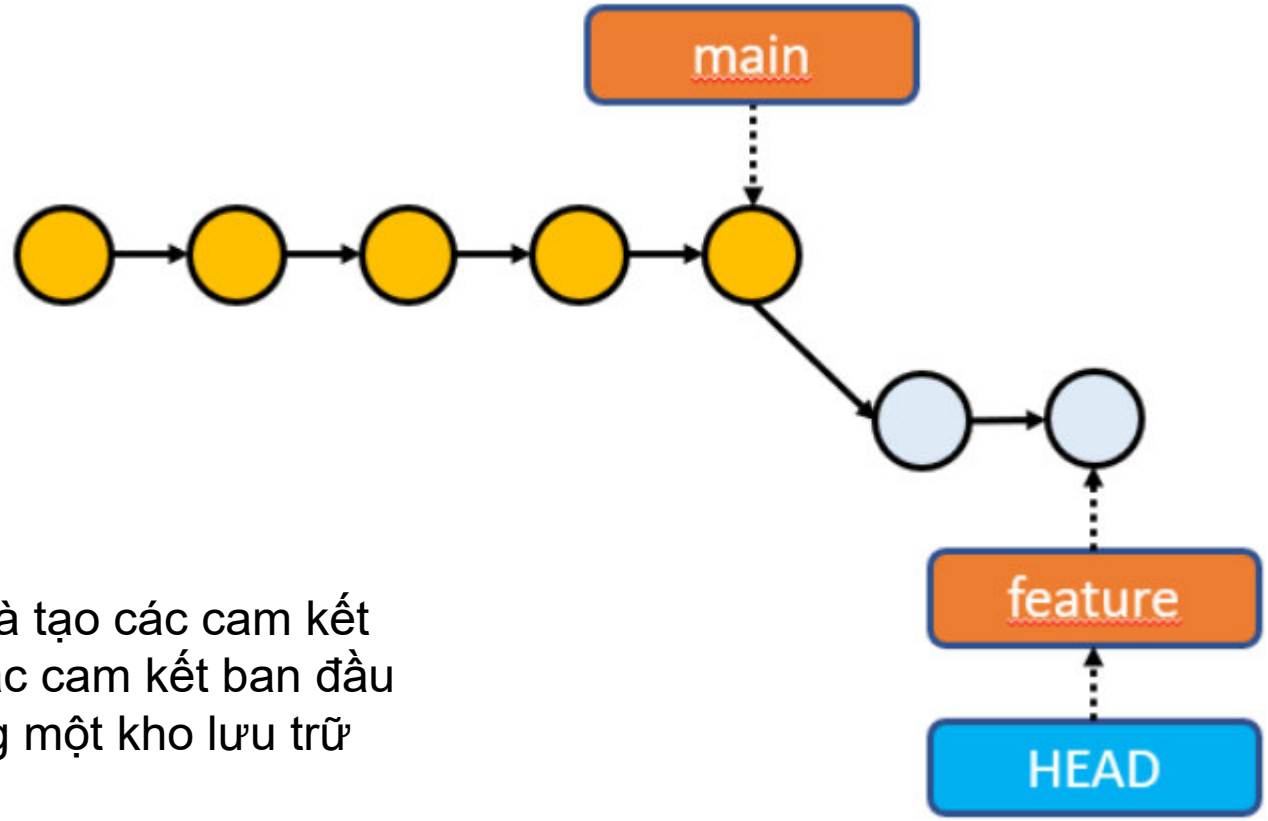


Chuyển nhánh

- git checkout feature

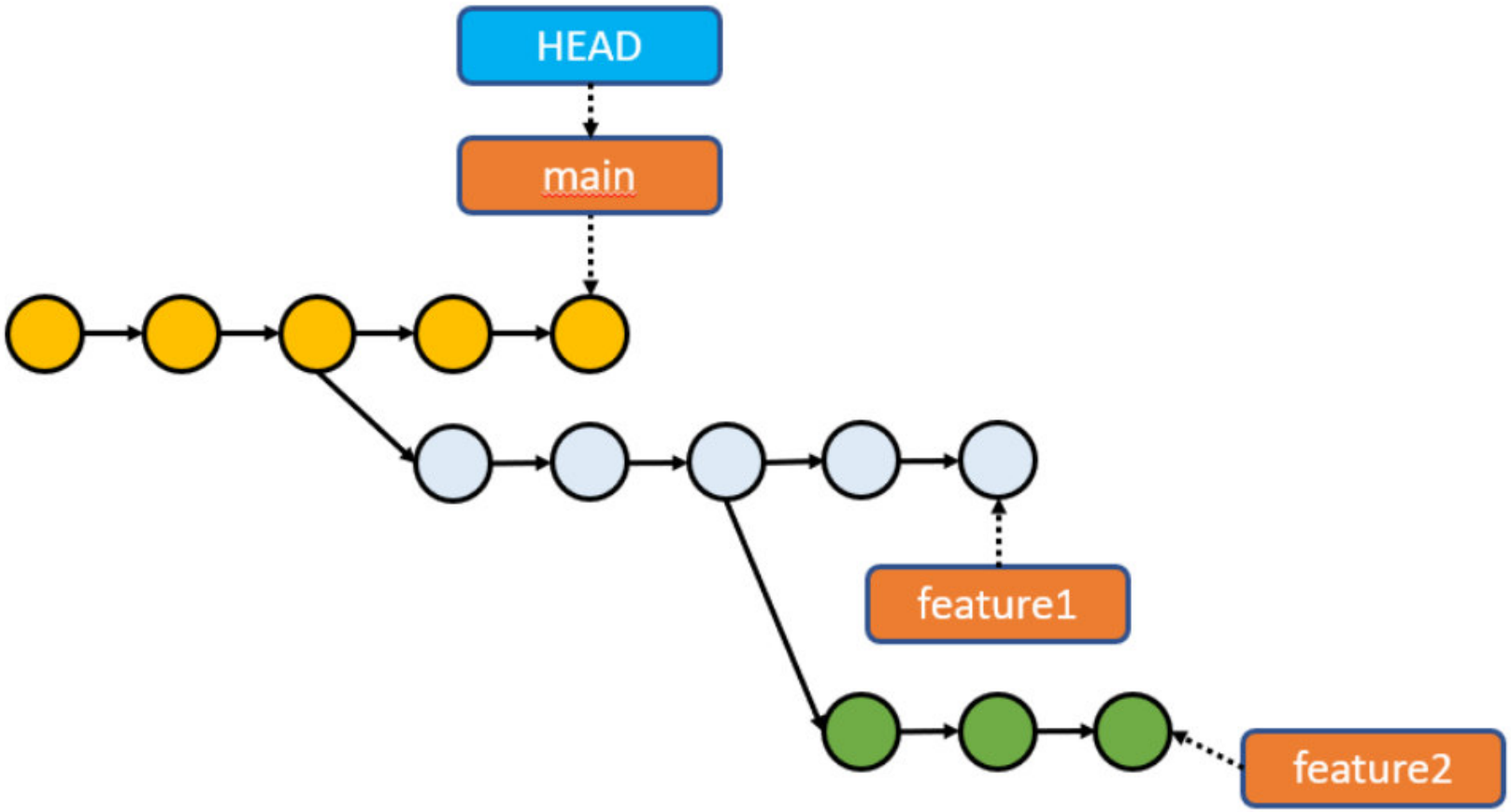


git rebase main

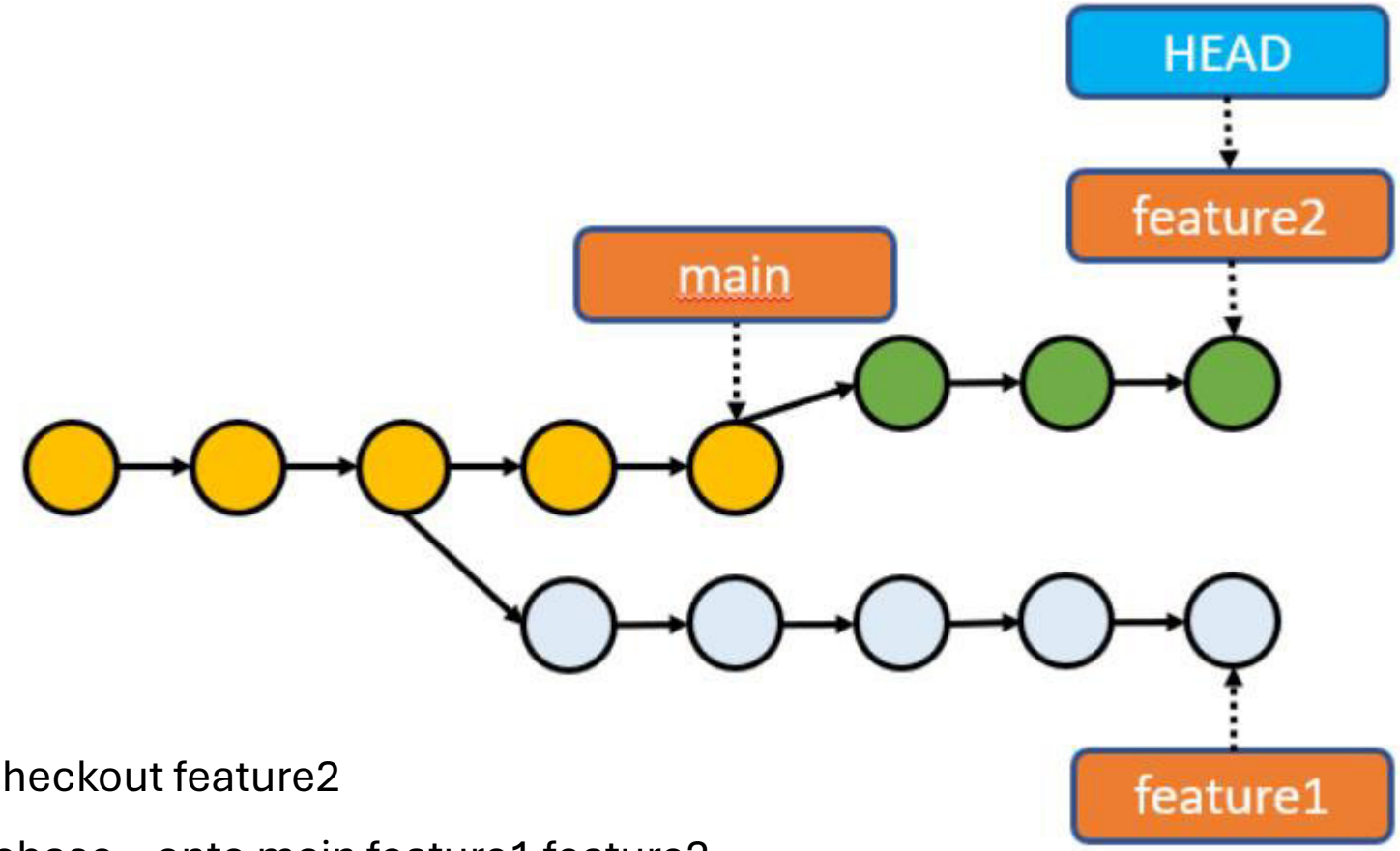


Quan trọng: hủy các cam kết ban đầu và tạo các cam kết mới! Có lẽ không phải là điều tốt nếu các cam kết ban đầu được chia sẻ với người khác hoặc trong một kho lưu trữ từ xa!rebase

Ngoài ra, bạn không cần phải rebase chỉ trong cùng một nhánh



Giải pháp là chuyển sang feature2 main



git checkout feature2

git rebase --onto main feature1 feature2

git rebase –onto <new-base> <old-base> <moved-branch>