



MÃ NGUỒN MỞ TRONG KHOA HỌC DỮ LIỆU

Bài 01. QUẢN LÝ MÃ NGUỒN Git / Github



Git / Version Control System - VCS

- VCS (Version Control System) là hệ thống quản lý phiên bản, giúp theo dõi và quản lý các thay đổi đối với tệp và mã nguồn trong quá trình phát triển dự án.
- Git là một trong những VCS phổ biến nhất hiện nay.

Git / Version Control System - VCS

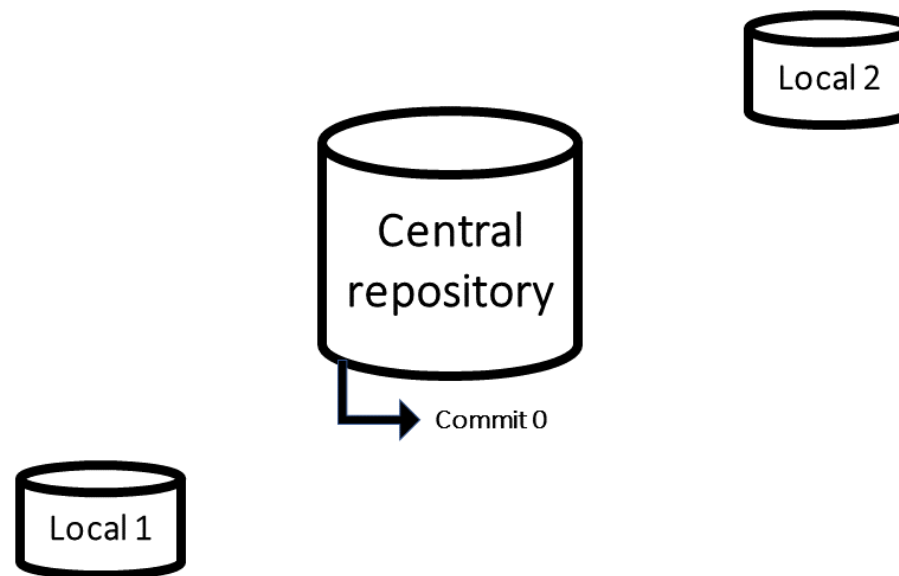
- **Lưu trữ lịch sử thay đổi:** VCS ghi nhận mọi thay đổi, bao gồm thông tin về những gì đã thay đổi, ai thay đổi, và khi nào thay đổi.
- **Quản lý các phiên bản:** Người dùng có thể quay lại các phiên bản trước nếu cần.
- **Cộng tác:** Nhiều thành viên có thể làm việc cùng lúc trên một dự án mà không lo xung đột.
- **Tích hợp quy trình tự động:** Một số VCS hỗ trợ kiểm tra mã (code review) và kiểm thử tự động trước khi hợp nhất thay đổi vào phiên bản chính.

Nguyên tắc

- Có một kho lưu trữ trung tâm - nơi lưu giữ phiên bản chính xác duy nhất của dự án.
- Bạn bắt đầu bằng cách lấy phiên bản mới nhất từ kho lưu trữ trung tâm.
- Công việc được chia thành các phần nhỏ (gọi là commit).
- Git cung cấp các công cụ để giải quyết xung đột tệp, v.v.

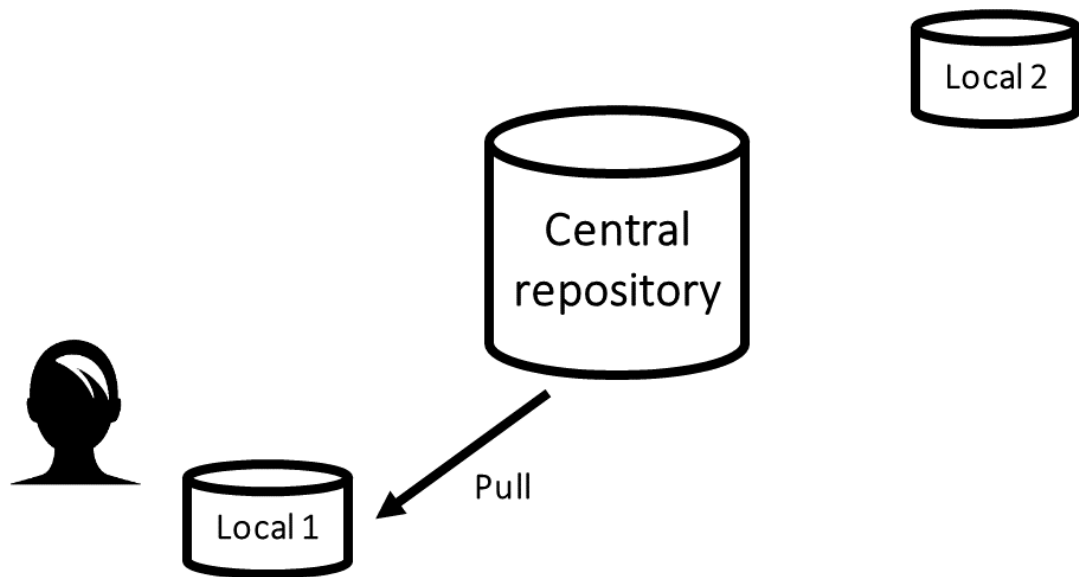
Ví dụ

Dự án được lưu trữ trong kho lưu trữ trung tâm.



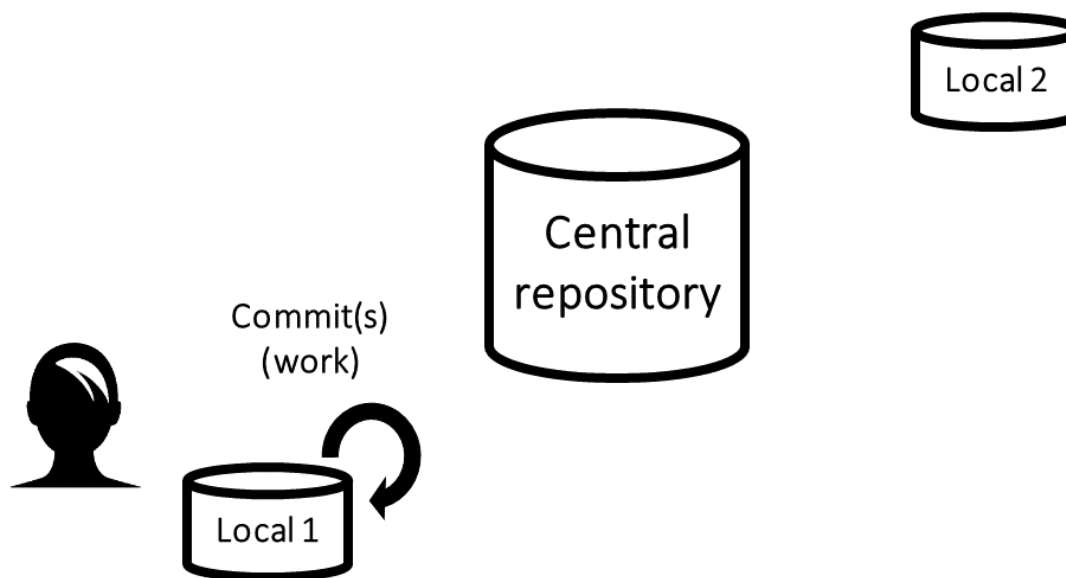
Ví dụ

Người đóng góp 1 lấy phiên bản mới nhất.



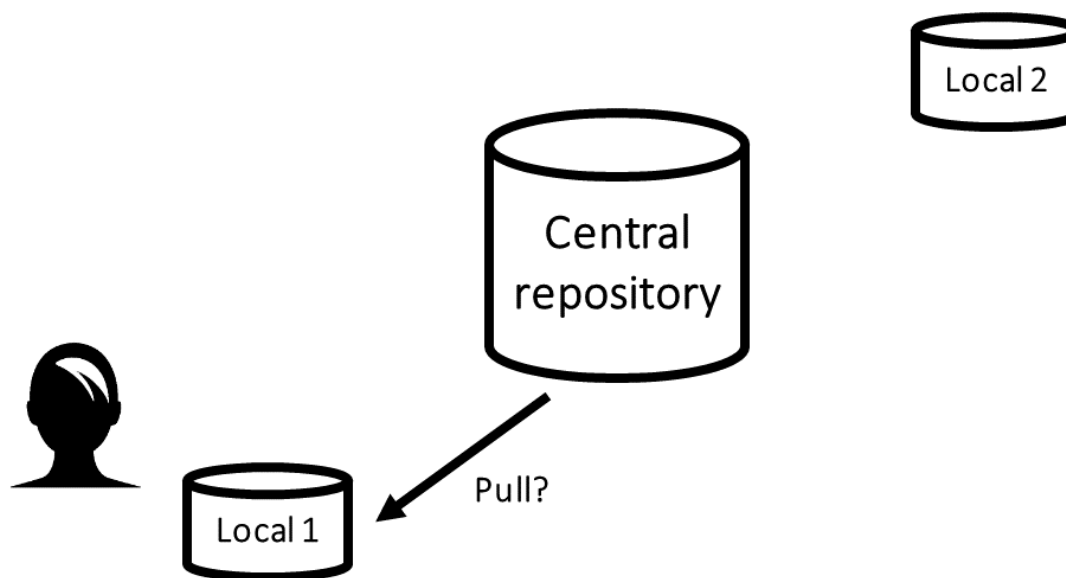
Ví dụ

Người đóng góp 1 thực hiện một số công việc mới tại máy cục bộ.



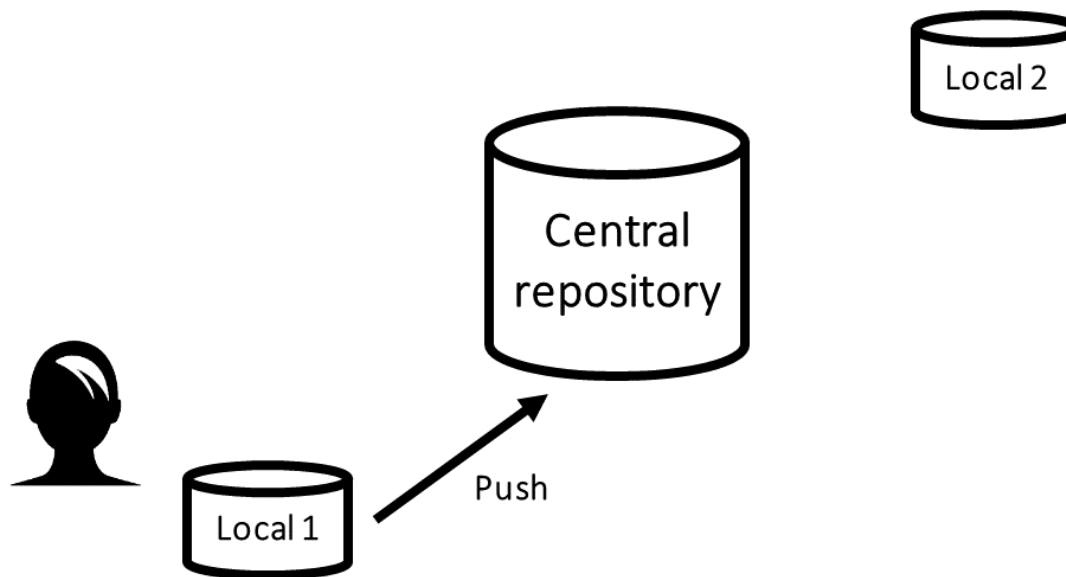
Ví dụ

Người đóng góp 1 kiểm tra xem phiên bản trung tâm có thay đổi trong thời gian đó không.



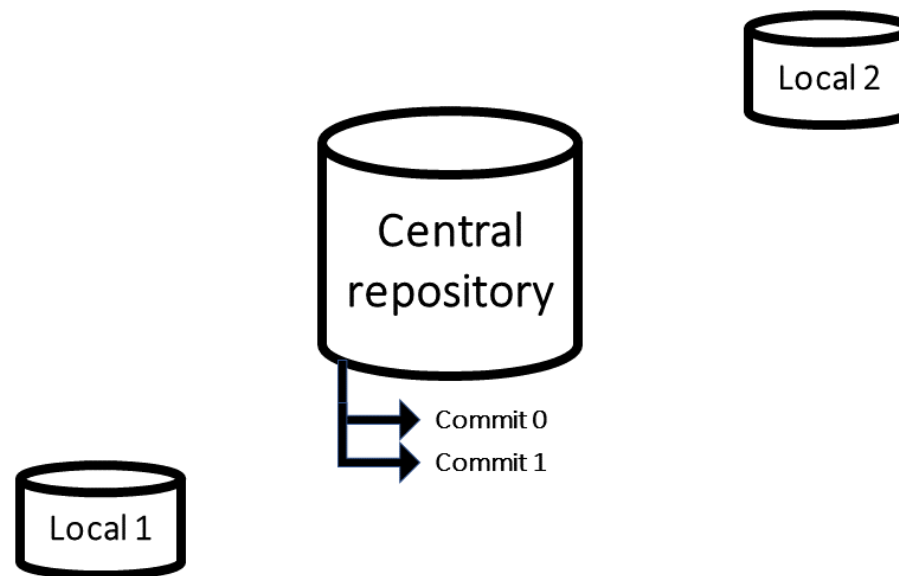
Ví dụ

Người đóng góp 1 đưa
những thay đổi của họ
vào kho lưu trữ trung tâm.



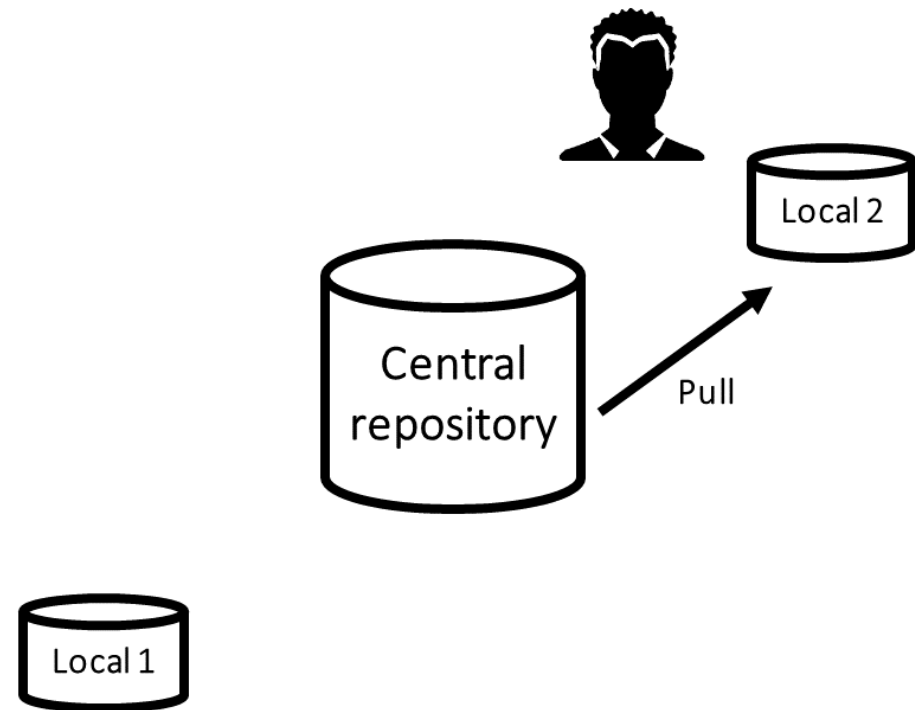
Ví dụ

Kho lưu trữ trung tâm
hiện lưu trữ bước mới ở
trên bước trước đó.

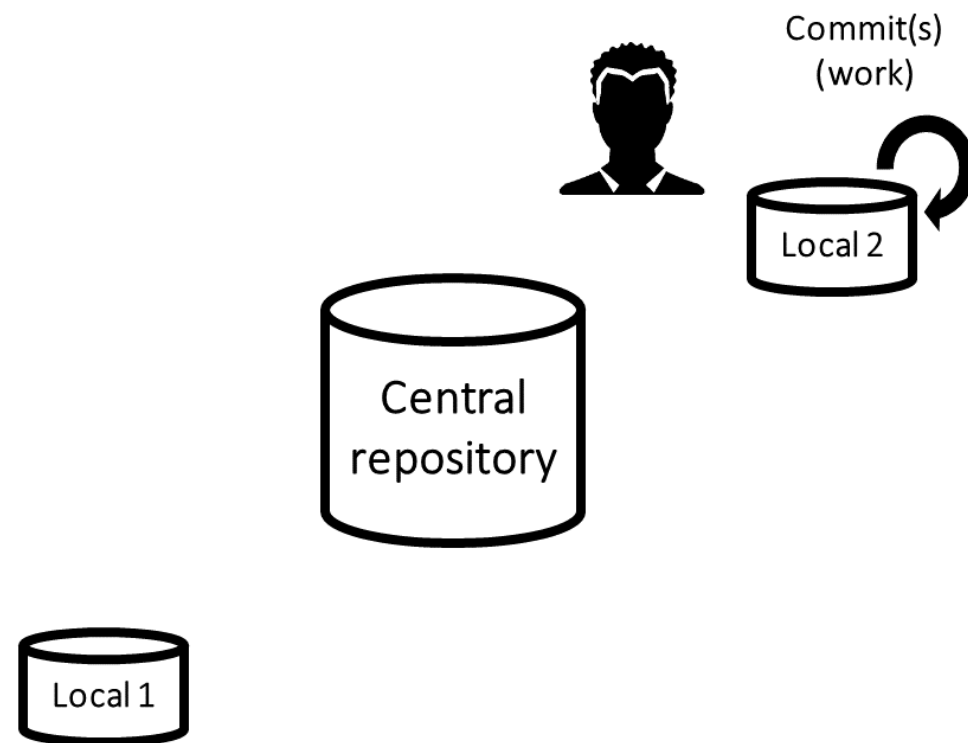


Ví dụ

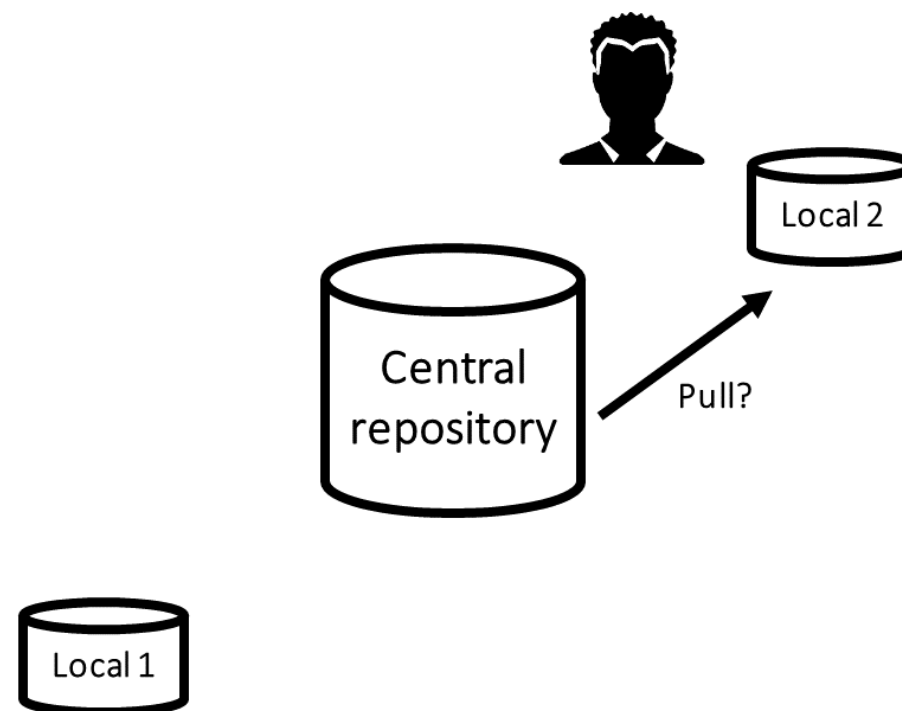
Người đóng góp 2 tham gia và thực hiện các bước tương tự.



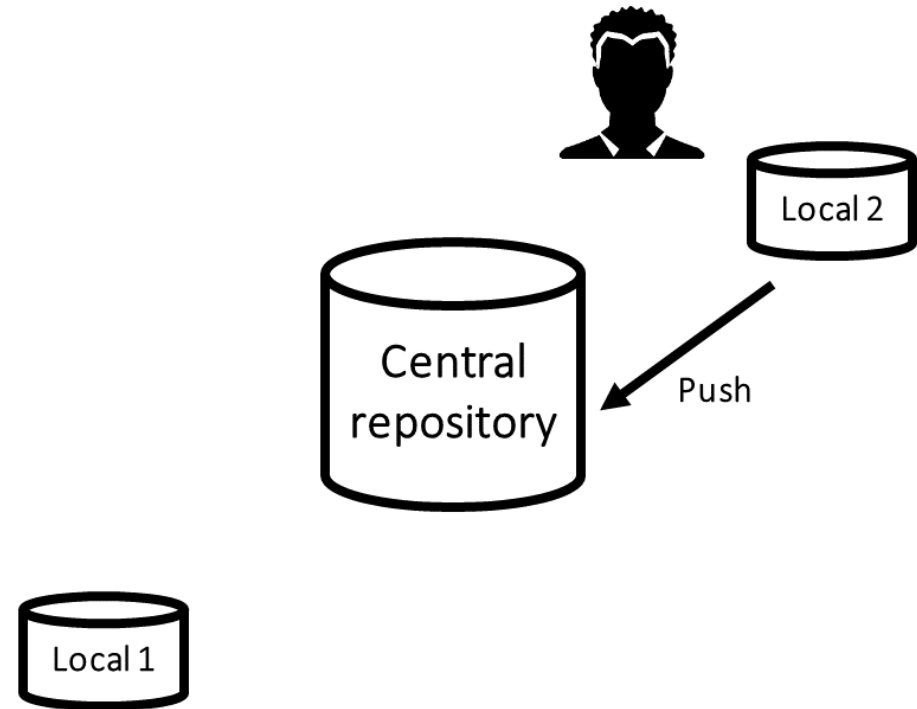
Ví dụ



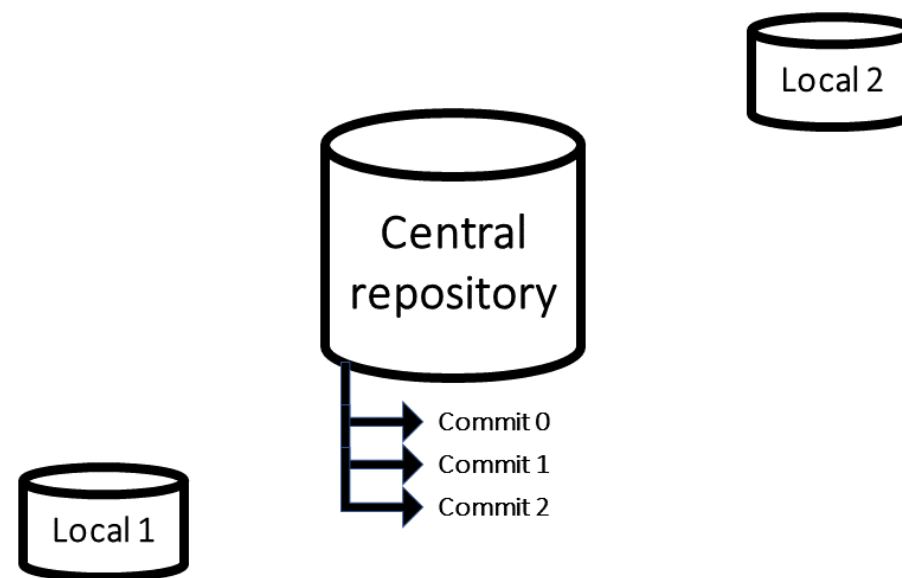
Ví dụ



Ví dụ

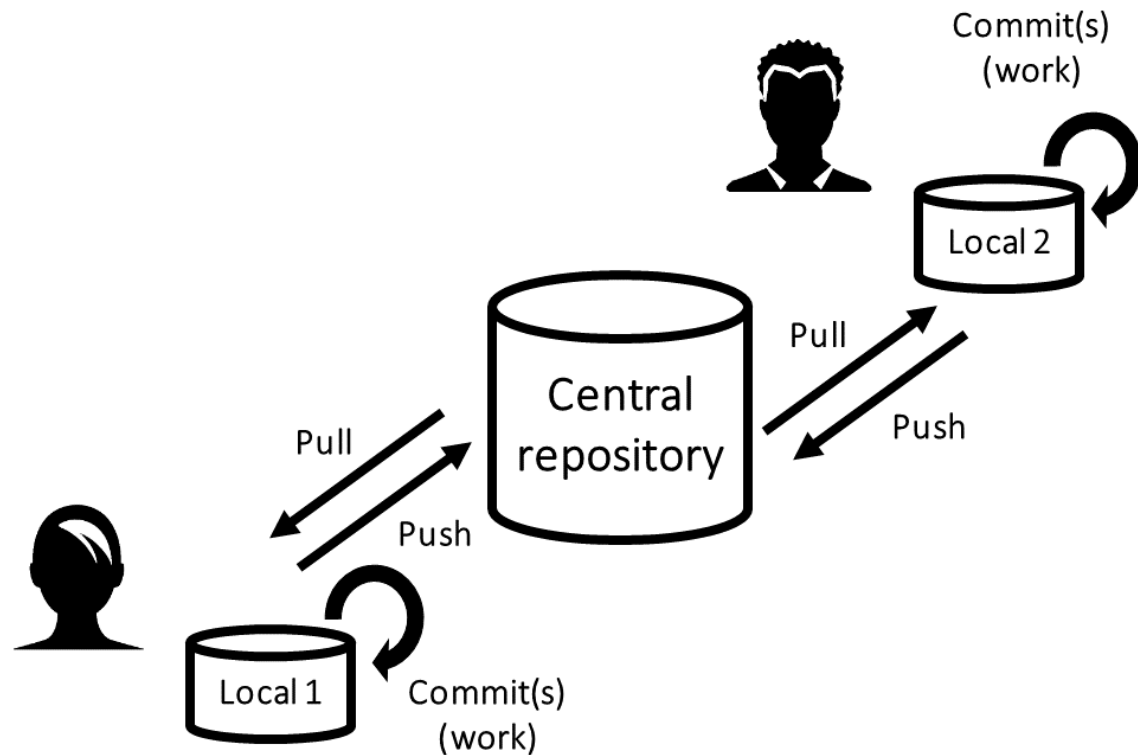


Ví dụ



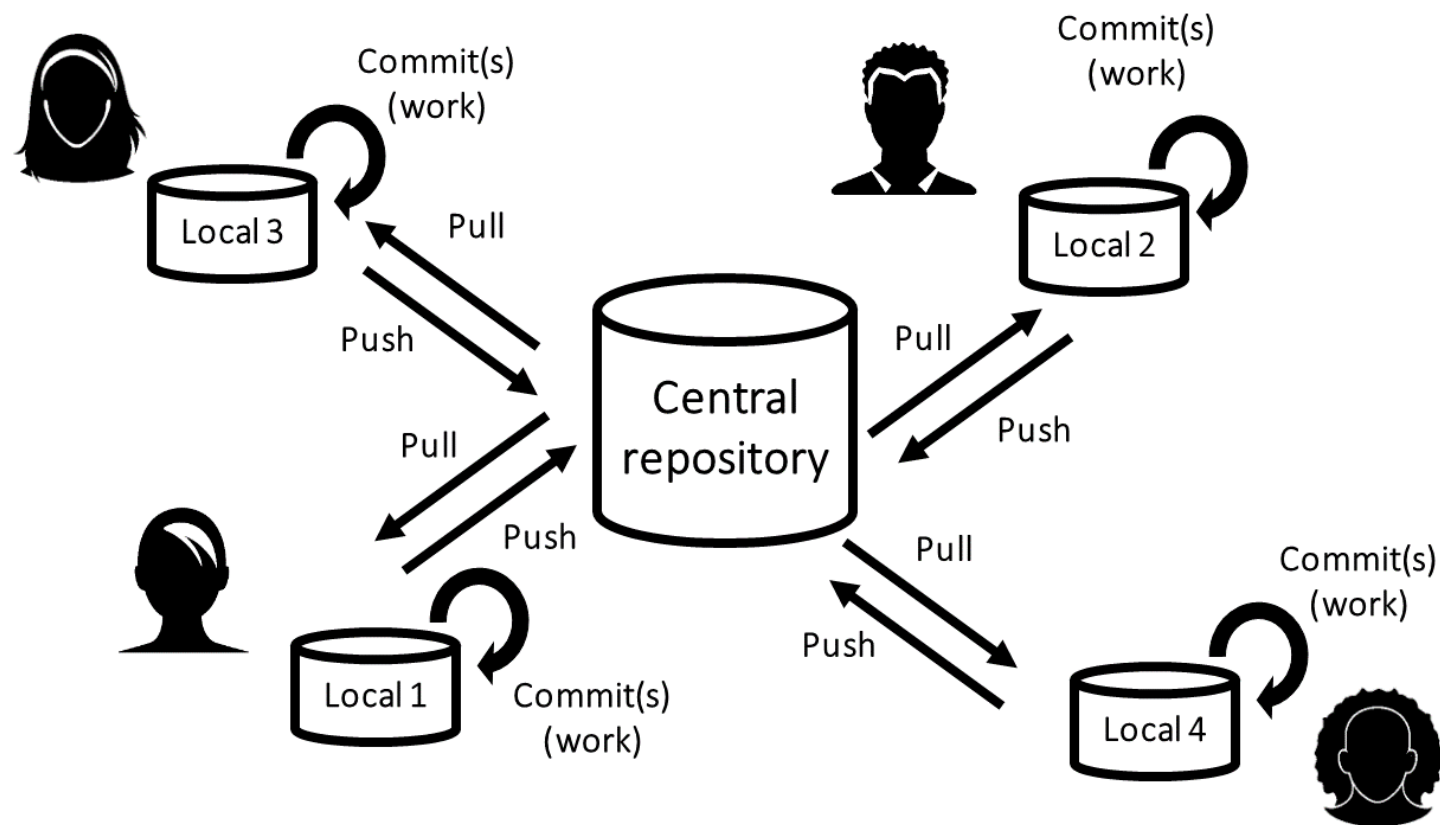
Ví dụ

Điều này có thể tiếp tục



Ví dụ

Và thu hút nhiều người
hơn nữa tham gia



Ưu điểm

- VCS giúp đảm bảo rằng chúng ta không làm hỏng dự án vĩnh viễn.
- Dễ dàng cộng tác với người khác trên cùng một dự án.
- Cho phép kiểm thử và phát triển các phiên bản mới (sử dụng nhánh - branching) mà không ảnh hưởng đến phiên bản đang hoạt động.
- Lưu trữ đầu ra, chia sẻ với cộng đồng, và cho phép người khác đóng góp (ví dụ: cập nhật mã nguồn). Có thể quay lại phiên bản trước bất cứ khi nào cần.

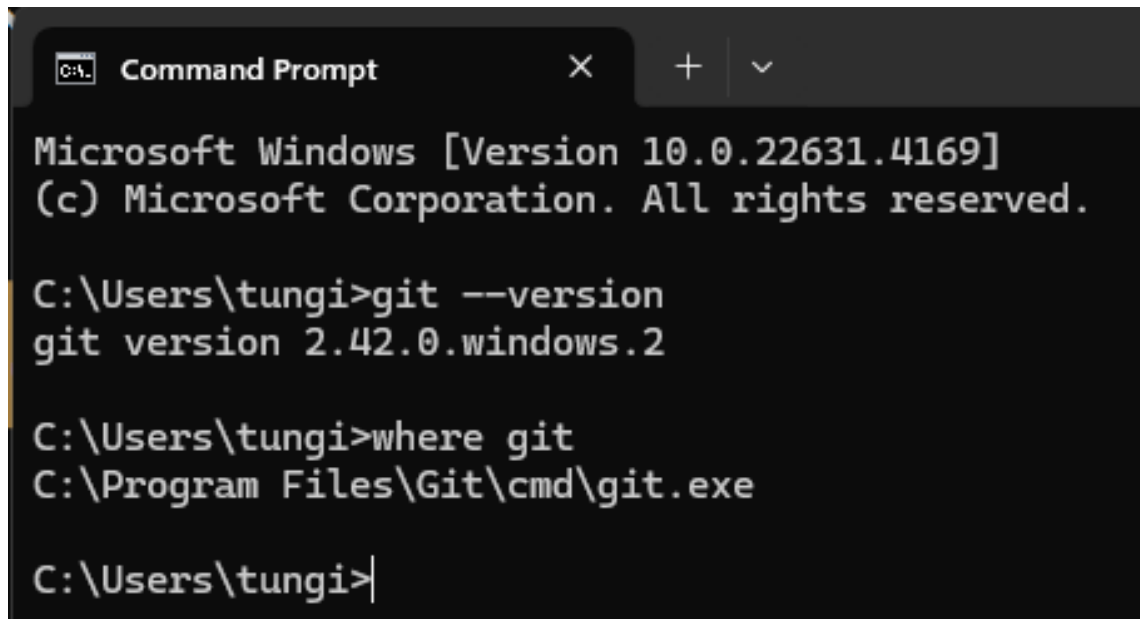
Giới thiệu về GIT



- **Hệ thống kiểm soát phiên bản phân tán nguồn mở**
 - Không giống như VCS tập trung từng phổ biến, DVCS như Git không cần kết nối liên tục với kho lưu trữ trung tâm
- Được Linus Torvalds tạo ra vào đầu những năm 2000 trong quá trình làm việc trên dự án hạt nhân Linux
- Khá khó học, rất khó thành thạo
- Trở nên phổ biến đến mức nó thay thế hiệu quả các công cụ cũ (svn, mercurial, cvs)

Cài đặt GIT

- Kiểm tra xem chúng ta có nó không?



```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tunggi>git --version
git version 2.42.0.windows.2

C:\Users\tunggi>where git
C:\Program Files\Git\cmd\git.exe

C:\Users\tunggi>
```

Cài đặt GIT

- Linux:
 - `$ sudo apt install git-all`
- Windows: tải Git tại: <https://gitforwindows.org/>
- macOS:
 - Chạy lệnh `git --version`, hệ thống có thể yêu cầu cài đặt trên các phiên bản macOS mới.
 - Nếu không có Xcode, chạy lệnh: `$ xcode-select --install`
 - Hoặc sử dụng Homebrew để cài Git: `$ brew install git`
- <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Tại sao gọi là "Git"?

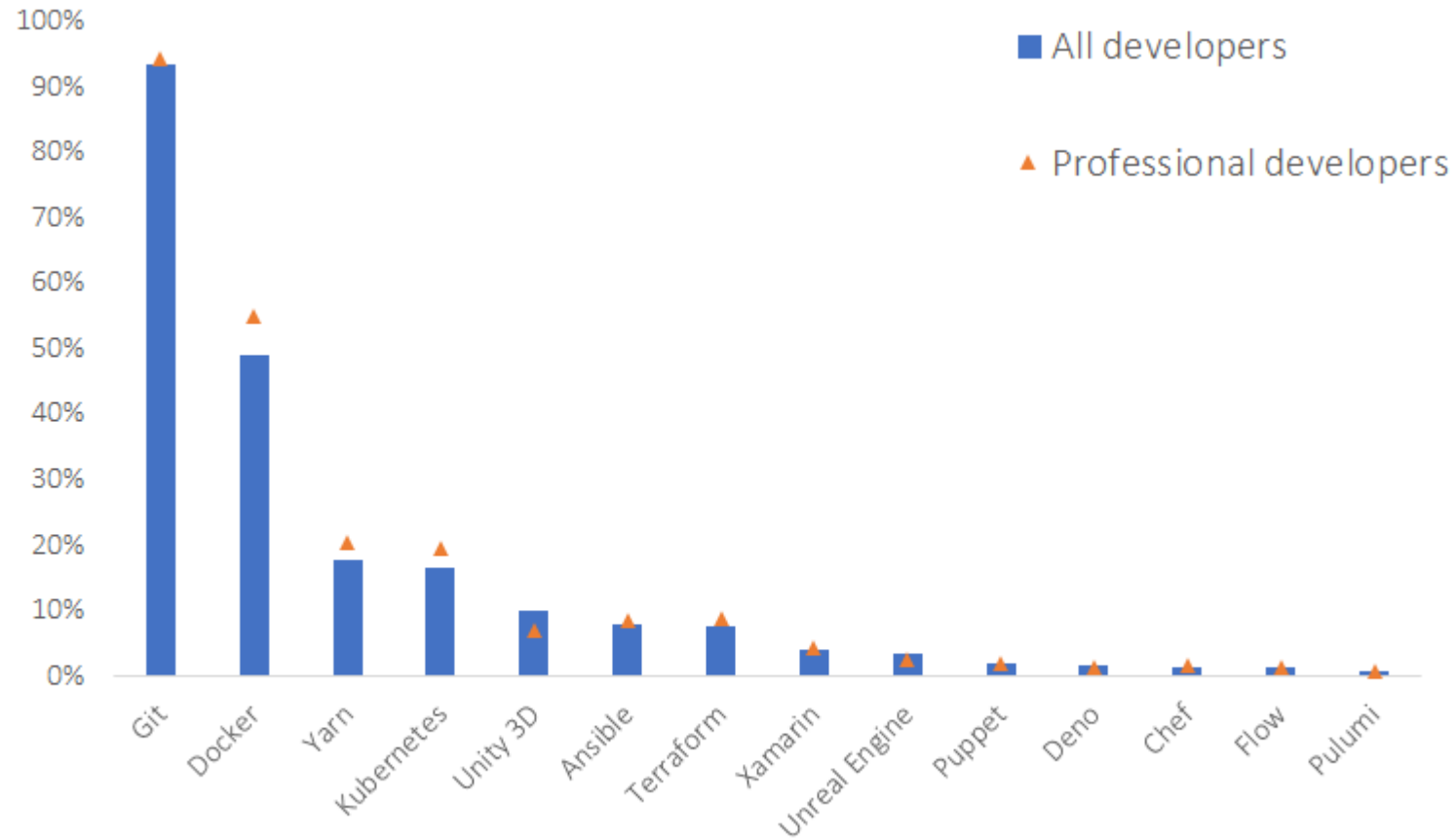


"Tôi là một kẻ tự cao, và tôi đặt tên tất cả các dự án của mình theo tên tôi. Đầu tiên là 'Linux', bây giờ là 'git'."

GIT

- ... và tính linh hoạt của nó giúp sử dụng các quy trình làm việc hiện đại hơn (như Scrum/Agile, v.v.).
- Git có cả giao diện dòng lệnh (CLI) và các plugin trực quan trong các môi trường phát triển tích hợp (IDEs), và tích hợp với nhiều ứng dụng, dịch vụ khác nhau.

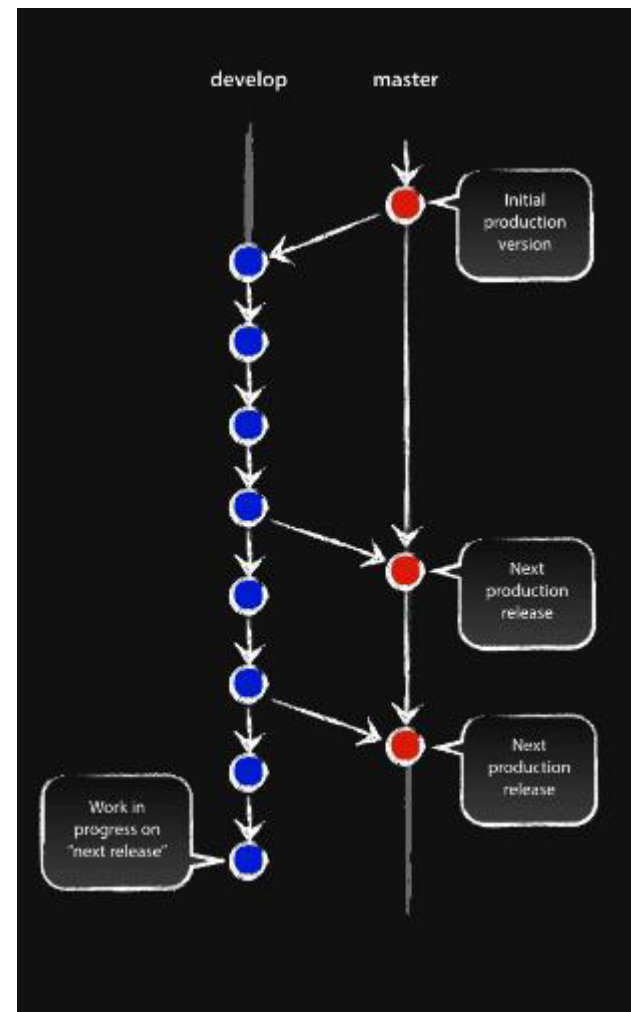
Các công cụ được sử dụng rộng rãi



<https://survey.stackoverflow.co/2021#most-popular-technologies-tools-tech-prof>

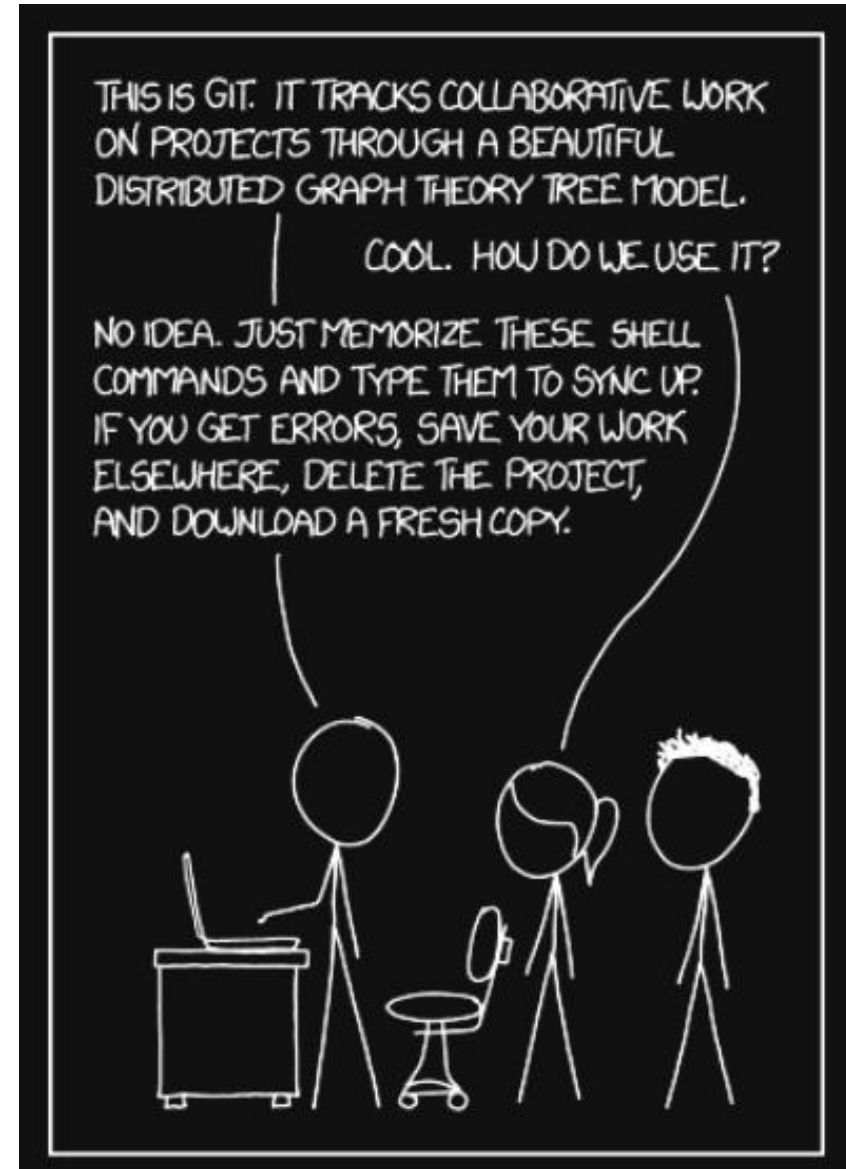
Một dự án đơn giản trong Git

- **repository**: Toàn bộ bộ sưu tập các tệp và thư mục của dự án; cùng với lịch sử phiên bản (bao gồm các phiên bản thay thế). Viết tắt: "repo".
- **commit**: Đơn vị cơ bản của công việc (ở đây là một vòng tròn).
- **branch**: Tập hợp các đơn vị công việc (dòng chảy xuống).
- **main (/master)**: Tên thông thường cho nhánh chính của repo (nhánh sản xuất).
- **develop**: Đây chỉ là tên được đặt cho nhánh với các tính năng đã phát triển. Bạn có thể chọn tên nhánh theo ý mình.
- **merge/rebase**: Được sử dụng để kết hợp hai nhánh.



CLI vs GUI

- Hôm nay chúng ta sẽ làm việc với Git bằng cách sử dụng CLI git bash để làm quen với các lệnh cơ bản



Một số lệnh terminal cơ bản

- **cd pathname**: Di chuyển đến thư mục theo đường dẫn (để lên một cấp, sử dụng **cd ..**).
- **mkdir foldername**: Tạo một thư mục mới.
- **dir**: Liệt kê các tệp/thư mục trong đường dẫn hiện tại (trên OS/Linux/bash, dùng **ls**).
- **echo text**: In một thông báo.
- **>**: Là ký hiệu chuyển hướng (redirection).
- **>>**: Chuyển hướng với việc nối thêm (append).

Một số lệnh terminal cơ bản

- **echo text > file**: In một văn bản vào tệp (và ghi đè lên tệp nếu đã tồn tại).
- **echo text >> file**: Thêm dòng mới vào tệp. Xem thêm **man echo**.
- **touch file**: Chỉ tạo một tệp mới (rỗng); Lưu ý: trên Linux, nếu tệp đã tồn tại, lệnh này sẽ có tác dụng khác.
- **cat file/message**: Hiển thị nội dung văn bản của tệp.
- **diff file1 file2**: Hiển thị sự khác biệt giữa hai tệp.
- **rm**: Xóa một tệp và xóa thư mục (dùng **rmdir** để xóa thư mục).

Bài tập 0

- Mở terminal
- Chọn một không gian cho các tài liệu nghiên cứu có thể tái tạo của bạn và điều hướng đến đó
- Ví dụ: bạn có thể sử dụng để điều hướng hoặc thực hiện "Git Bash tại đây" trong Windows.cd <pathname>

Bài tập 0

- Tạo thư mục RR_git1
`mkdir RR_git1`
- Di chuyển vào thư mục mới
`cd RR_git1`
- Tạo file classes.txt và ghi nội dung là ngày hôm nay
`echo "DD/MM/YYYY" > classes.txt`

Một số lệnh cơ bản trong GIT

- **git --help**: Hiển thị trợ giúp nội tuyến về Git.
- **git [cmd] --help**: Hiển thị trợ giúp web về lệnh [cmd].
- **git --version**: Hiển thị thông tin chẩn đoán (phiên bản).
- **git status**: Hiển thị trạng thái của kho lưu trữ cục bộ.
- **git log**: Hiển thị lịch sử các commit.

Các lệnh GIT cơ bản – dùng để setup repo

- **git init [repo_name]**: Khởi tạo một kho lưu trữ rỗng trong thư mục hiện tại (hoặc thư mục được chỉ định).
- **git clone [repo_name] [clone_name]**: Tạo một bản sao liên kết của kho lưu trữ.
- **git config -l**: Xem tất cả các tùy chọn cấu hình.
- **Cấu trúc cấu hình**: **git config [-l] [--scope] [option_name] [value]**
- Có ba cấp độ cấu hình (tức là phạm vi):
 - **--system**: Áp dụng cho tất cả các kho lưu trữ của người dùng hệ thống.
 - **--global**: Áp dụng cho tất cả các kho lưu trữ của người dùng, ghi đè các thiết lập hệ thống.
 - **--local**: Áp dụng cho kho lưu trữ hiện tại, ghi đè các thiết lập toàn cầu. (Mặc định).
- **Lưu ý**: Cấu hình sẽ chỉ hiển thị nếu bạn đã sử dụng Git trước đó (và thêm một số tùy chọn).
- **Lưu ý 2**: Cấu hình sẽ chỉ hiển thị nếu bạn đang ở trong một kho lưu trữ Git.

Bài tập 1 – Tạo repo

- Trong thư mục RR_git1 của bạn, hãy khởi tạo một kho lưu trữ Git có tên là EX1 và vào thư mục đó:

```
git init EX1
```

```
cd EX1
```

or

```
mkdir EX1
```

```
cd EX1
```

```
git init
```

Bài tập 1 – Tạo repo

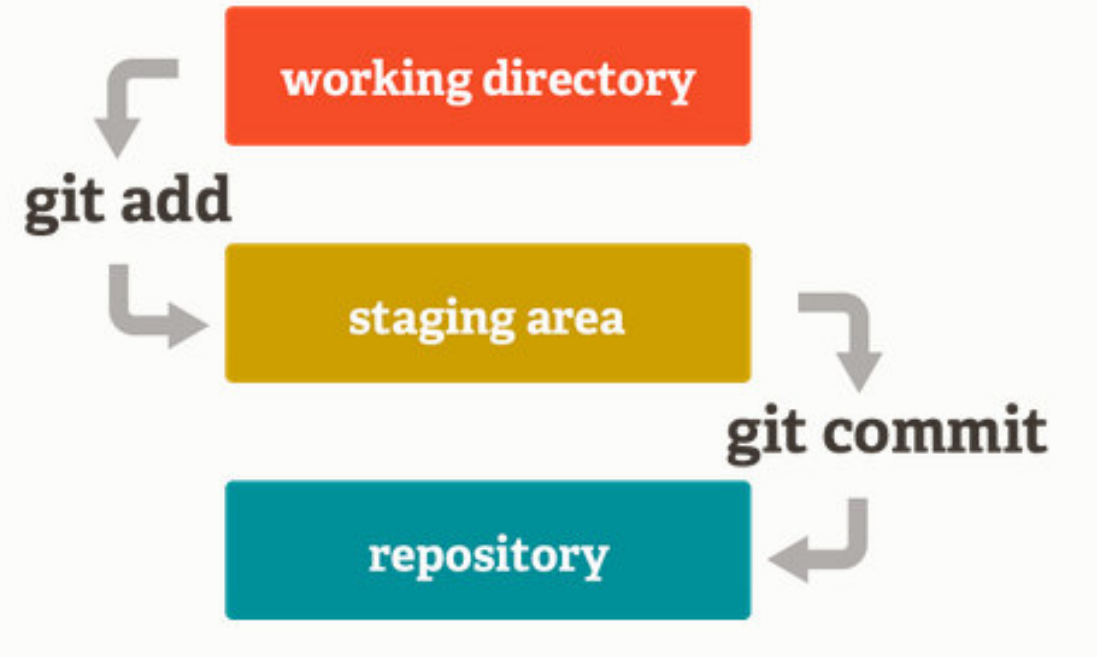
- Hiển thị tất cả các cấu hình: **git config -l**
- Liệt kê tất cả các tùy chọn global: **git config -l --global**
- Liệt kê tất cả các tùy chọn local: **git config -l --local**
- Gán global 'user.name'
git config --global user.name "Name Surname"
- Gán global 'user.email'
git config --global user.email "your.email@smth.smth"
- Liệt kê lại các lựa chọn để kiểm tra: **git config -l --global**
- Gán local option 'user.name': **git config --local user.name "AB"**
- Kiểm tra: **git config -l --local**

Bài tập 1 – Tạo repo

- Hiển thị tất cả các cấu hình: **git config -l**
- Liệt kê tất cả các tùy chọn global: **git config -l --global**
- Liệt kê tất cả các tùy chọn local: **git config -l --local**
- Gán global 'user.name'
git config --global user.name "Name Surname"
- Gán global 'user.email'
git config --global user.email "your.email@smth.smth"
- Liệt kê lại các lựa chọn để kiểm tra: **git config -l --global**
- Gán local option 'user.name': **git config --local user.name "AB"**
- Kiểm tra: **git config -l --local**

Ba trạng thái của GIT

- Không giống như các VCS khác, Git có một thứ gọi là "staging area" hoặc "index". Đây là khu vực trung gian nơi các cam kết có thể được định dạng và xem xét trước khi hoàn tất cam kết.



Ba cây của GIT

Tree	Role
HEAD	Last commit snapshot, next parent
Index	Proposed next commit snapshot
Working Directory	Sandbox

HEAD

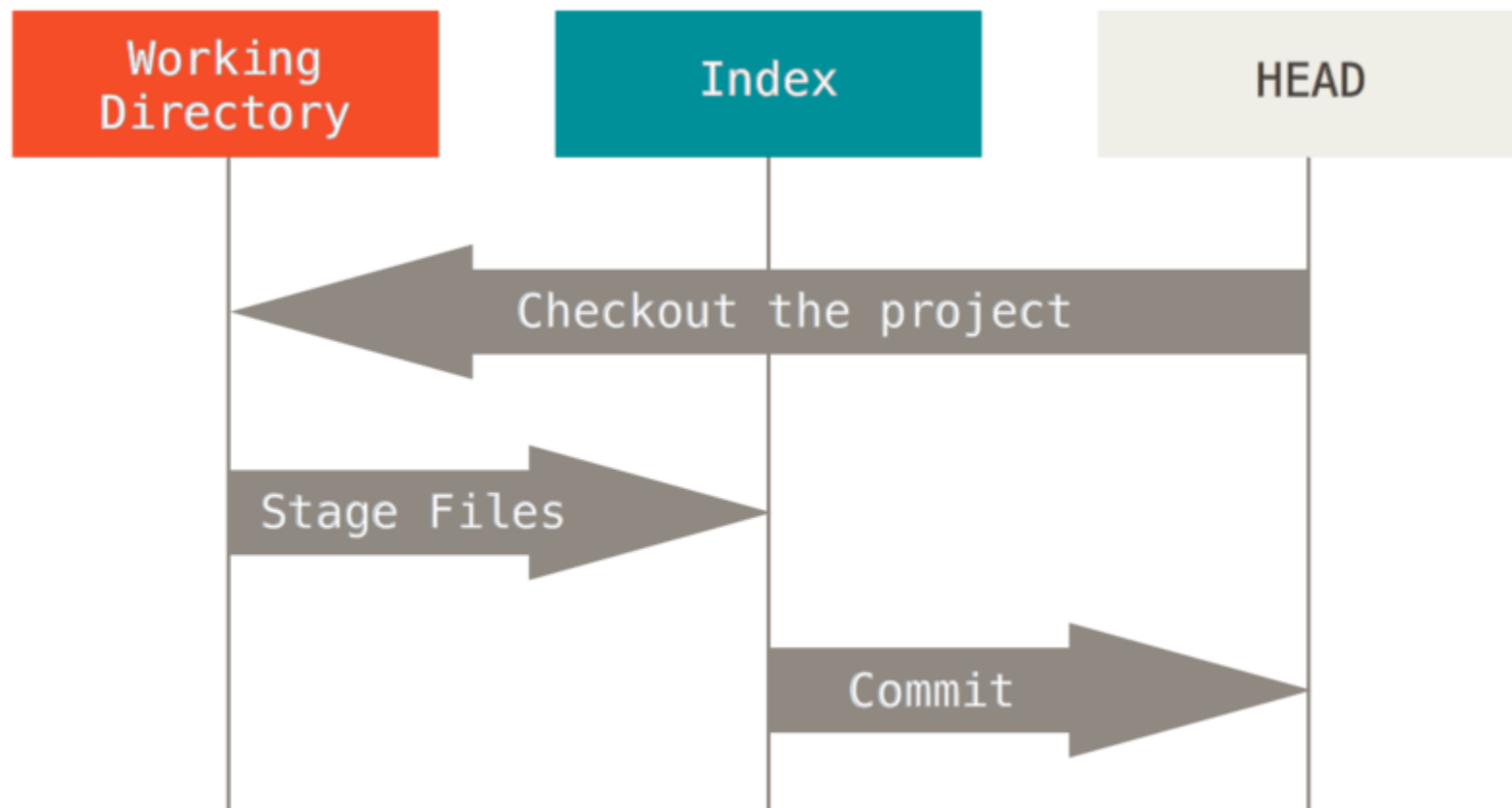
- HEAD là snapshot của commit gần nhất trên nhánh hiện tại.
- Lệnh để xem snapshot: **\$ git cat-file -p HEAD**

Staging Area (Index)

- Khu vực staging là commit dự kiến tiếp theo. Đây là nơi bạn đặt các tệp mà bạn muốn đưa vào commit tiếp theo. Là một "commit chưa hoàn thành" để chuẩn bị.
- Lệnh để xem nội dung: **git ls-files**

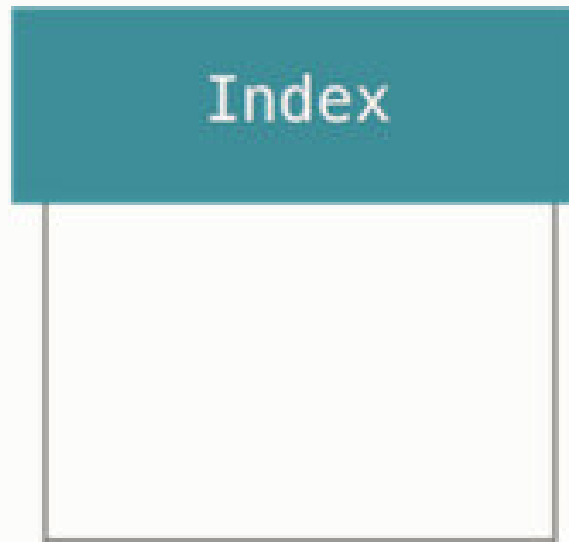
Working Directory (Thư mục làm việc)

- Thư mục làm việc là khu vực thử nghiệm thay đổi trước khi gửi chúng đến khu vực staging và sau đó vào lịch sử.



Ví dụ

- #0 Tại thời điểm này, chỉ có cây thư mục làm việc là có nội dung.



Ví dụ

- #1 Chúng ta sử dụng **git add** để lấy nội dung trong thư mục làm việc và sao chép vào index.
- (Lưu ý rằng chúng ta giữ lại tất cả các hộp. Hiện tại có hai hộp lưu trữ cùng một thông tin)



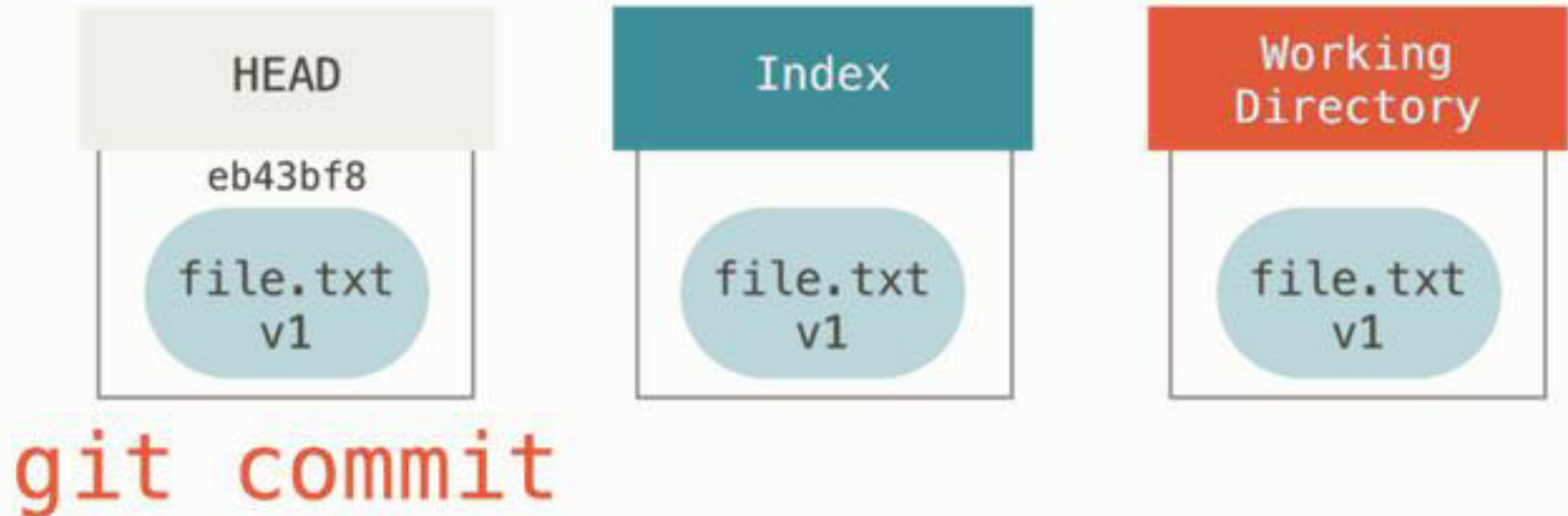
Ví dụ

- #2 Chúng ta sử dụng **git commit**, lệnh này lấy nội dung của chỉ mục và lưu dưới dạng ảnh chụp nhanh vĩnh viễn, tạo đối tượng commit trỏ đến ảnh chụp nhanh đó và cập nhật master để trỏ đến commit đó.
- (Nói chung, hiện có một commit mới trên nhánh chính và có một chỉ báo trỏ đến commit đó và nói rằng "này, đây là nơi chúng ta đang ở".)



Ví dụ

- #3 Nếu chúng ta chạy lệnh git status, chúng ta sẽ không thấy thay đổi nào vì cả ba cây đều giống nhau.



Quy trình Làm việc

- **git add [filename(s)]**: Thêm các tệp vào khu vực staging.
- **git add .** : Thêm tất cả các tệp mới hoặc đã thay đổi vào khu vực staging.
- **git commit -m "<commit description>"**: Tạo một commit mới với các thay đổi trong khu vực staging.

Tại bất kỳ thời điểm nào, bạn có thể:

- **git status** : Xác minh vị trí hiện tại và sự khác biệt giữa ba khu vực (working directory, staging area, và repository).
- **git diff** : So sánh commit cuối cùng với những gì đang có trong thư mục làm việc.
- **git log** : Xem lịch sử commit.

Bài tập 2: thêm commits

- Trong thư mục RR_git1, hãy khởi tạo kho lưu trữ git có tên EX2

cd ..

git init EX2

(kiểm tra git status và git diff để hiểu rõ hơn về điều này)

Bài tập 2: thêm commits

- Truy cập vào bài tập 2: **cd EX2**
- Tạo một tệp có tên **README.md**, thêm một dòng văn bản vào bên trong, lưu tệp [gợi ý: bạn có thể sử dụng echo hoặc tạo thủ công bằng ví dụ như **Notepad**]:

```
touch README.md
```

```
echo "one line" >> README.md
```

Bài tập 2: thêm commits

- Truy cập vào bài tập 2: `cd EX2`
- Tạo một tệp có tên **README.md**, thêm một dòng văn bản vào bên trong, lưu tệp [gợi ý: bạn có thể sử dụng echo hoặc tạo thủ công bằng ví dụ như **Notepad**]:

```
touch README.md
```

```
echo "one line" >> README.md
```

(kiểm tra git status và git diff để hiểu rõ hơn)

Bài tập 2: thêm commits

- Chuẩn bị tập tin mới: **git add README.md**
- Xác nhận tệp (nhớ bao gồm mô tả xác nhận hữu ích!):
git commit -m "Added README.md with one line of text"

(check git status and git diff and git log to get a better feel of this)

Bài tập 3: thêm commits

- Thêm một dòng văn bản khác vào tệp bạn đã tạo:
echo "a second line" >> README.md
- Tạo file mới readme.txt: **touch readme.txt**
- Thực hiện các câu lệnh:
 - **mkdir data**
 - **git status**
 - **git add .**
 - **git commit -m "Modified README.md and added readme.txt"**
 - **git log**
 - **git status**

Bài tập 4: thêm commits

- Tạo tệp **data/data1.csv** và điền vào đó một dòng dữ liệu ngẫu nhiên (có thể chỉ là văn bản được phân tách bằng dấu phẩy, không quan trọng), kiểm tra trạng thái và sự khác biệt.
 - `echo "var1,var2\n1,2" > data/data1.csv`
 - `git status`
- Tạo một tệp **.gitignore** (có, bắt đầu bằng dấu chấm), đặt từ 'data' vào bên trong (đó là tên thư mục của chúng tôi), kiểm tra trạng thái và sự khác biệt
 - `touch .gitignore`
 - `echo "data" >> .gitignore`
 - `git status`
- .gitignore là một tệp yêu cầu git bỏ qua một số phần tử nhất định.