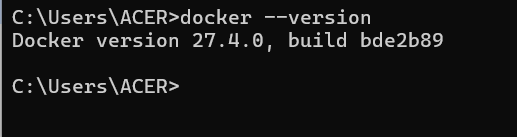
|  |  |
| --- | --- |
| **Họ và tên** | Phan Nguyễn Khôi Nguyên |
| **Mssv** | 21068021 |
| **Môn** | Kiến trúc và thiết kế phần mềm |

# **Kiểm tra phiên bản docker**

docker --version

* **Ảnh minh chứng**



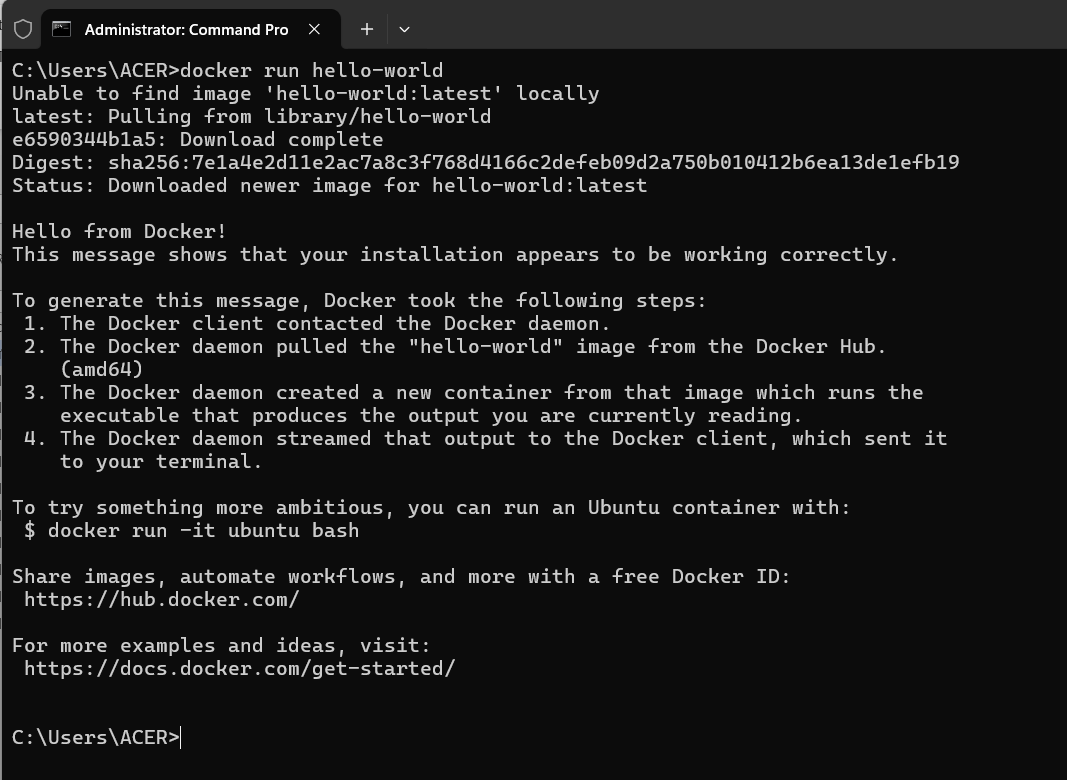
* **Giải thích**

Lệnh này dùng để kiểm tra phiên bản hiện tại của Docker đang cài đặt trên máy.

1. **Chạy container từ image**

docker run hello-world

* **Ảnh minh chứng**

****

* **Giải thích**

Docker kiểm tra xem image “hello-world” đã có trên máy chưa.

Nếu chưa có, Docker sẽ tự động tải image “hello-world” từ **Docker Hub**.

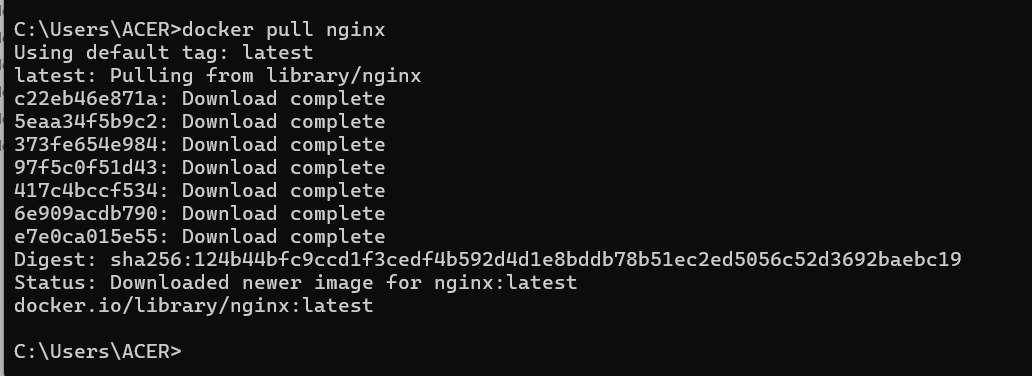
Sau đó, Docker sẽ tạo một container từ image này và chạy nó.

Chạy chương trình bên trong container, in ra thông điệp chào mừng từ Docker.

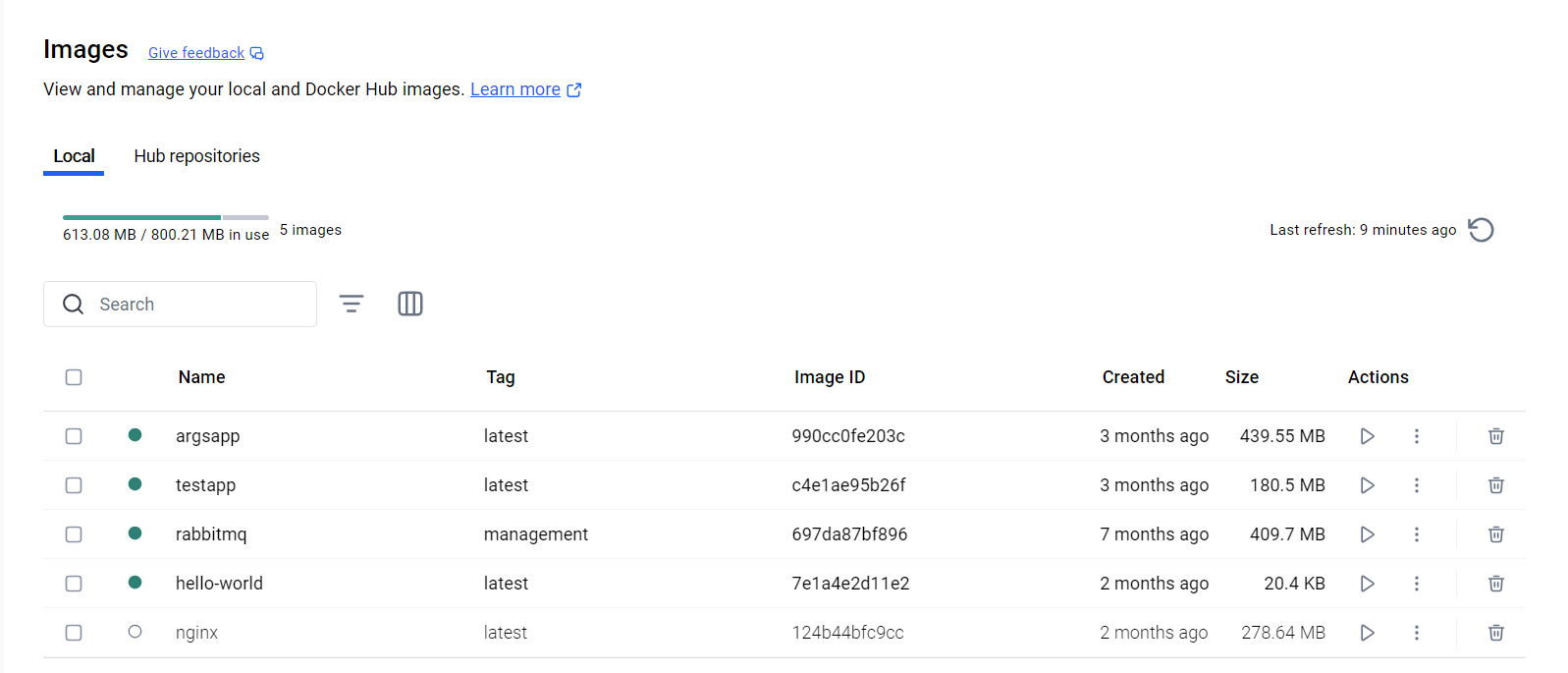
# **Tải (pull) image từ Docker Hub về máy cục bộ.**

docker pull nginx

* **Ảnh minh chứng**

****

Sau khi thực hiện lệnh trên. Mở **docker desktop** sẽ thấy image ‘nginx’ như sau:



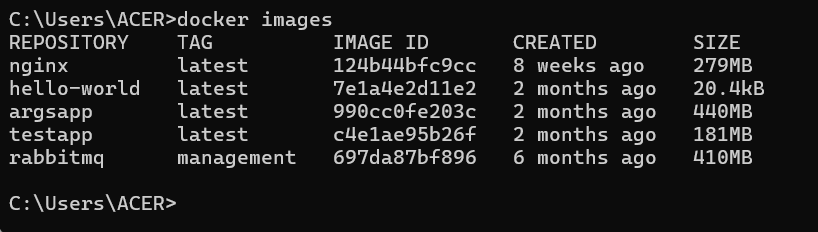
* **Giải thích**

Kết nối đến Docker Hub để tìm image nginx. Sau đó tải image về máy (nếu chưa có hoặc có bản cập nhật). Lưu image vào bộ nhớ local, để có thể sử dụng sau này mà không cần tải lại.

# **Hiển thị danh sách các image Docker**

docker images

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh trên dùng để liệt kê tất cả các image đã được tải xuống hoặc tạo trên hệ thống.

Trong đó:

* REPOSITORY: Tên của image (ví dụ: nginx, ubuntu).
* TAG: Phiên bản của image (ví dụ: latest, 22.04).
* IMAGE ID: ID duy nhất của image.
* CREATED: Thời gian image được tạo.
* SIZE: Dung lượng của image.

# **Tạo vào chạy container từ image**

docker run -d nginx

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh này tạo và chạy một container từ image nginx trong chế độ detached (chạy nền).

Khi chạy lệnh trên, Docker sẽ:

**Tạo một container mới** từ image “nginx”.

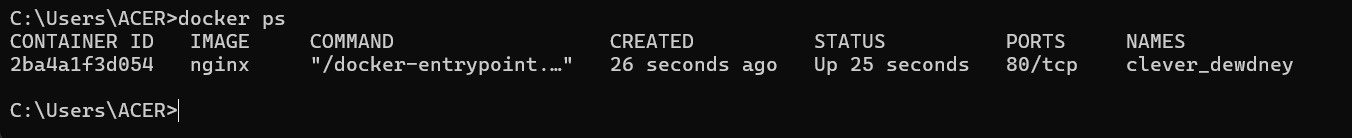
**Chạy container ở chế độ detached**, không hiển thị output trong terminal.

**Nginx sẽ bắt đầu chạy** bên trong container, như một máy chủ web.

# **Hiển thị danh sách các container Docker đang chạy**

docker ps

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh này sẽ liệt kê tất cả các container đang chạy (đang hoạt động) trên hệ thống

Trong đó:

* CONTAINER ID: ID của container.
* IMAGE: Image được sử dụng để tạo container (ví dụ: nginx).
* COMMAND: Lệnh đang chạy trong container.
* CREATED: Thời gian container được tạo.
* STATUS: Trạng thái của container (ví dụ: Up 25 seconds cho thấy container đang chạy được 25 giây).
* PORTS: Cổng mạng mà container đang sử dụng, nếu có.
* NAMES: Tên container (do Docker tự tạo hoặc người dùng đặt).

# **Hiển thị danh sách tất cả các container, bao gồm cả các container đang chạy và các container đã dừng.**

docker ps -a

* **Ảnh minh chứng**

****

* **Giải thích**

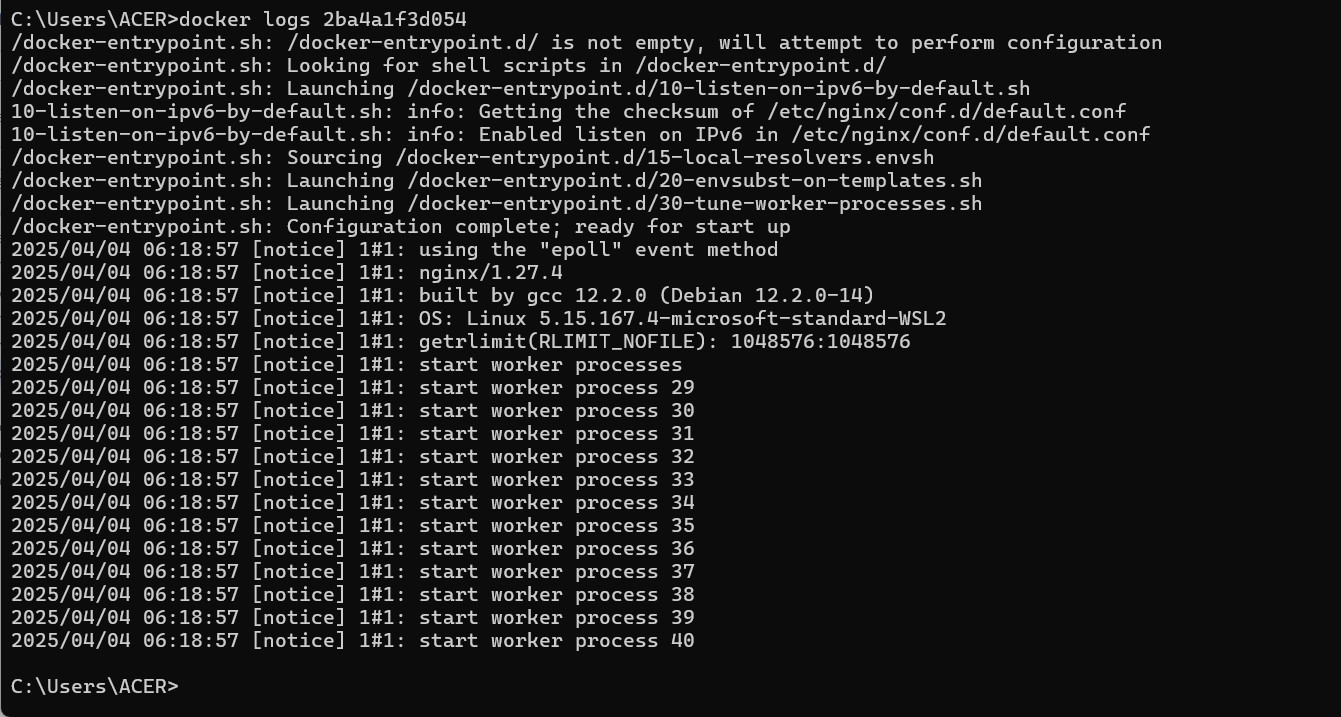
Các cột tương tự với lệnh docker ps nhưng côt STATUS sẽ có thêm 2 trang thái là :

* Exited (0): Đã dừng nhưng không có lỗi (mã thoát 0).
* Exited (1): Đã dừng và có lỗi.

# **Hiển thị log của container**

docker logs <container\_id>

* **Ảnh minh chứng**

****

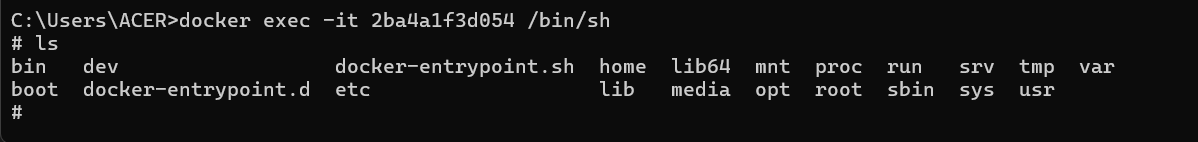
* **Giải thích**

Lệnh này dùng để lấy thông tin log của một container, giúp xem các sự kiện, lỗi hoặc thông báo mà container đã ghi lại trong quá trình hoạt động.

# **Thực thi lệnh bên trong một container đang chạy.**

docker exec -it <container\_id> /bin/sh

* **Ảnh minh chứng**

****

Cách thoát: gõ exit hoặc CTRL + D

* **Giải thích**

Lệnh này cho phép chạy một shell interactively (/bin/sh) bên trong một container đang chạy, giúp thao tác và kiểm tra trực tiếp trong môi trường container.

* **Lưu ý:**

Khi sử dụng Docker để tương tác với container, có thể mở một shell bằng lệnh **docker exec -it <container\_id> bash** hoặc **docker exec -it <container\_id> sh**. Tuy nhiên với:

* **Bash** (Bourne Again Shell) là một shell mạnh mẽ, phổ biến trên các hệ thống Linux hiện đại. Nó hỗ trợ nhiều tính năng nâng cao, như autocompletion (hoàn thành tự động), command history (lịch sử lệnh), và cú pháp phức tạp hơn trong scripting, chẳng hạn như các cấu trúc điều kiện và vòng lặp. Vì thế, **bash** là lựa chọn lý tưởng khi bạn cần một môi trường shell đầy đủ và tiện ích, đặc biệt là khi làm việc với các script phức tạp hoặc khi cần dễ dàng gỡ lỗi và kiểm tra lệnh.
* **sh** (Bourne Shell) là một shell cổ điển hơn, vốn đã có mặt từ lâu và có sẵn trên tất cả các hệ thống Unix/Linux. Mặc dù không có các tính năng mạnh mẽ như **bash**, **sh** vẫn là một shell đơn giản và đủ dùng để thực thi các lệnh cơ bản. Nó có thể thích hợp hơn trong các container nhẹ hoặc môi trường yêu cầu sự tương thích tối đa với các hệ thống khác mà không cần tính năng nâng cao. Do không có các tính năng phức tạp của **bash**, **sh** có kích thước nhẹ hơn và có thể sử dụng tài nguyên ít hơn.

Tóm lại:

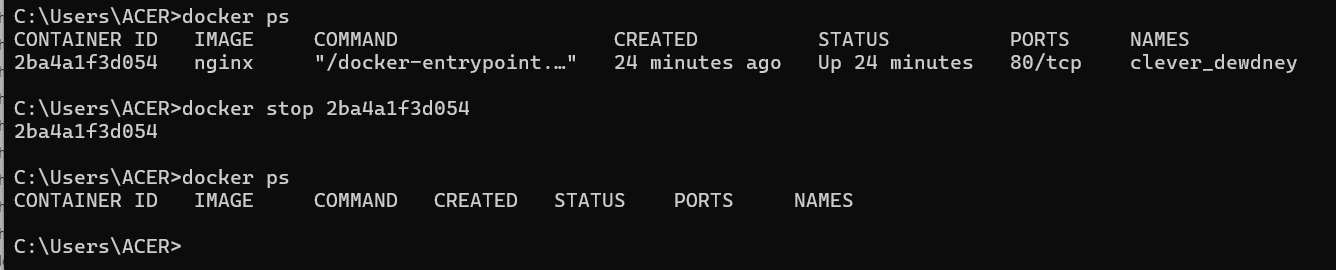
Dùng bash nếu bạn cần tính năng nâng cao và môi trường lập trình đầy đủ, đặc biệt khi làm việc với các script phức tạp.

Dùng sh nếu bạn muốn một môi trường shell nhẹ, tương thích với hầu hết các hệ thống và container nhỏ gọn.

# **Dừng một container đang chạy trên Docker**

docker stop <container\_id>

* **Ảnh minh chứng**

****

* **Giải thích**

Gửi tín hiệu dừng đến container, yêu cầu container dừng các tiến trình bên trong một cách an toàn.

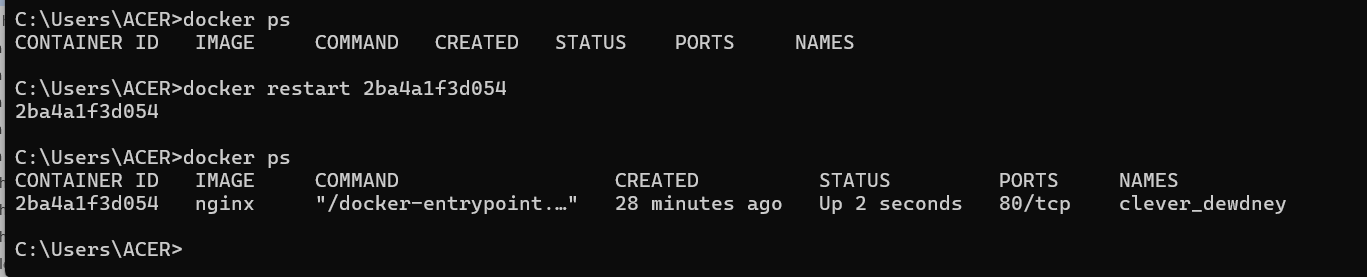
Đợi một thời gian ngắn (mặc định là 10 giây) để container có thể tắt các dịch vụ đang chạy. Nếu không dừng được trong thời gian này, Docker sẽ gửi tín hiệu SIGKILL để buộc container dừng ngay lập tức.

Dừng container: Container sẽ không còn chạy nữa, và không thể tương tác cho đến khi bắt đầu lại nó.

# **Khởi động lại container**

docker restart <container\_id>

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh này dùng để dừng container đang chạy và khởi động lại nó. Đây là cách nhanh chóng để áp dụng các thay đổi hoặc xử lý khi container gặp sự cố mà không cần phải xóa hoặc tạo lại container từ đầu. Nếu container đang ở trạng thái dừng khi bạn chạy lệnh, Docker sẽ khởi động lại container đó.

# **Xóa container đã dừng**

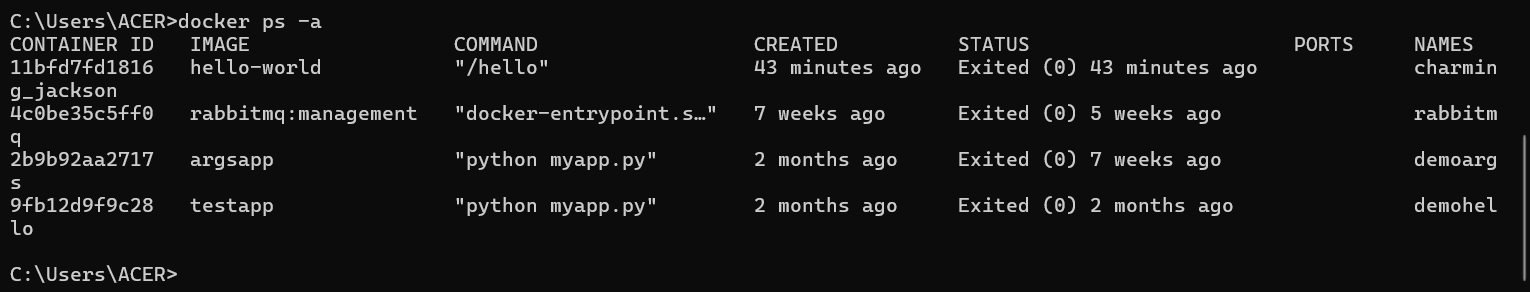
docker rm <container\_id>

* **Ảnh minh chứng**

****

Lưu ý: Trước khi xóa container cần dừng lại nếu container đang start.

Sau khi xóa kiểm tra bằng lệnh docker ps



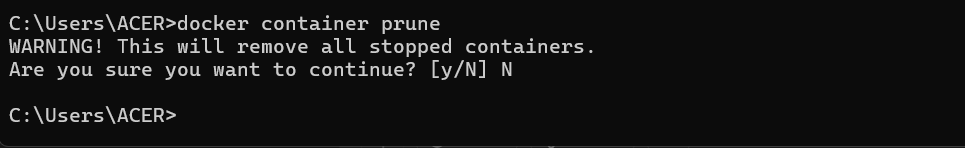
* **Giải thích**

Lệnh này dùng để xóa một hoặc nhiều container đã dừng. Sau khi container bị xóa, nó sẽ không còn tồn tại trên hệ thống Docker của bạn nữa, và bạn không thể tương tác với nó nữa.

# **Xóa tất cả container đã dừng**

docker container prune

* **Ảnh minh chứng**

****

Nếu muốn xóa toàn bộ container đã dừng thì chọn y

* **Giải thích**

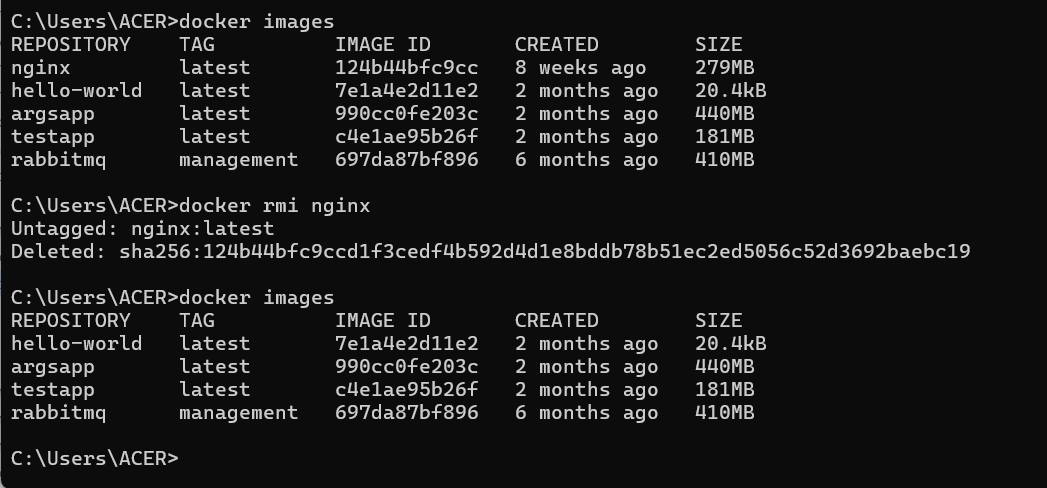
Lệnh này sẽ xóa tất cả các container đã dừng mà không cần chỉ định container cụ thể. Đây là cách nhanh chóng để dọn dẹp các container không còn sử dụng và đã bị dừng, giúp tiết kiệm dung lượng đĩa.

Khi bạn chạy lệnh này, Docker sẽ yêu cầu xác nhận trước khi tiến hành xóa các container đã dừng. Bạn có thể thêm tùy chọn -f (force) để bỏ qua yêu cầu xác nhận và tự động xóa các container.

# **Xóa một hoặc nhiều image**

docker rmi <image\_id>

* **Ảnh minh chứng**

****

* **Giải thích**

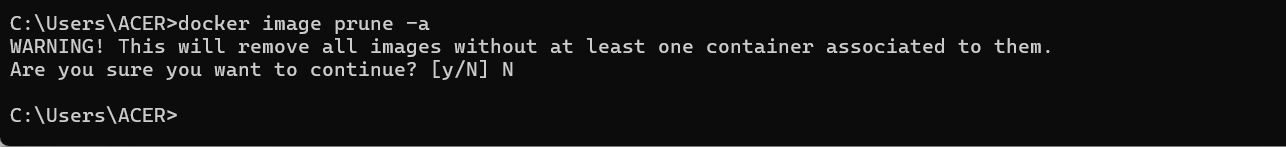
Lệnh này dùng để xóa Docker image khỏi hệ thống . Một Docker image là mẫu hoặc bản sao của hệ điều hành và ứng dụng được sử dụng để tạo các container. Sau khi xóa image, bạn không thể tạo container từ image đó nữa trừ khi bạn tải lại hoặc xây dựng lại image.

Lưu ý: Nếu một container đang sử dụng image bạn muốn xóa, Docker sẽ không cho phép bạn xóa image đó cho đến khi bạn xóa container liên quan. Bạn sẽ cần phải dừng và xóa container trước khi có thể xóa image. Nếu muốn xóa image dù có container phụ thuộc, có thể sử dụng tùy chọn -f để ép xóa image

# **Xóa tất cả các Docker images không sử dụng**

docker image prune -a

* **Ảnh minh chứng**

****

Do ở đây không cần xóa nên chọn N. Nếu vẫn muốn xóa thì chọn y

* **Giải thích**

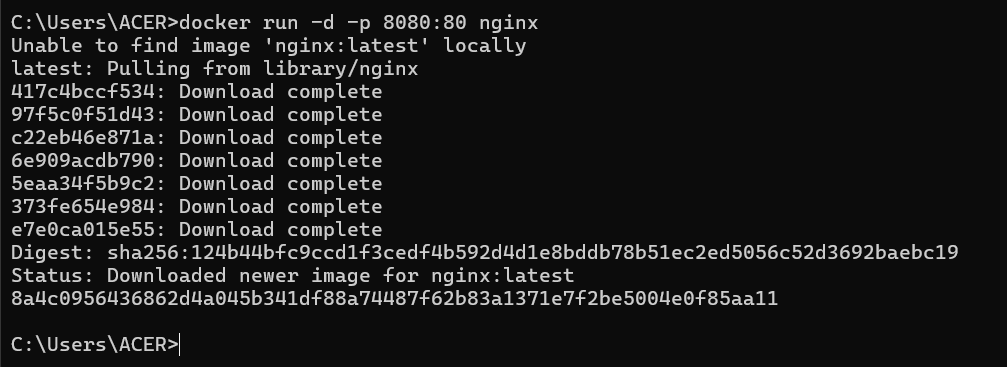
docker image prune: Lệnh này giúp bạn dọn dẹp các Docker images không còn sử dụng, giúp giải phóng không gian đĩa.

-a: Tham số này chỉ định rằng bạn muốn xóa tất cả các image không sử dụng, không chỉ những image bị treo hoặc không còn container nào đang sử dụng. Điều này bao gồm các image mà bạn đã tải xuống, xây dựng nhưng chưa sử dụng hoặc không còn container nào liên kết đến chúng.

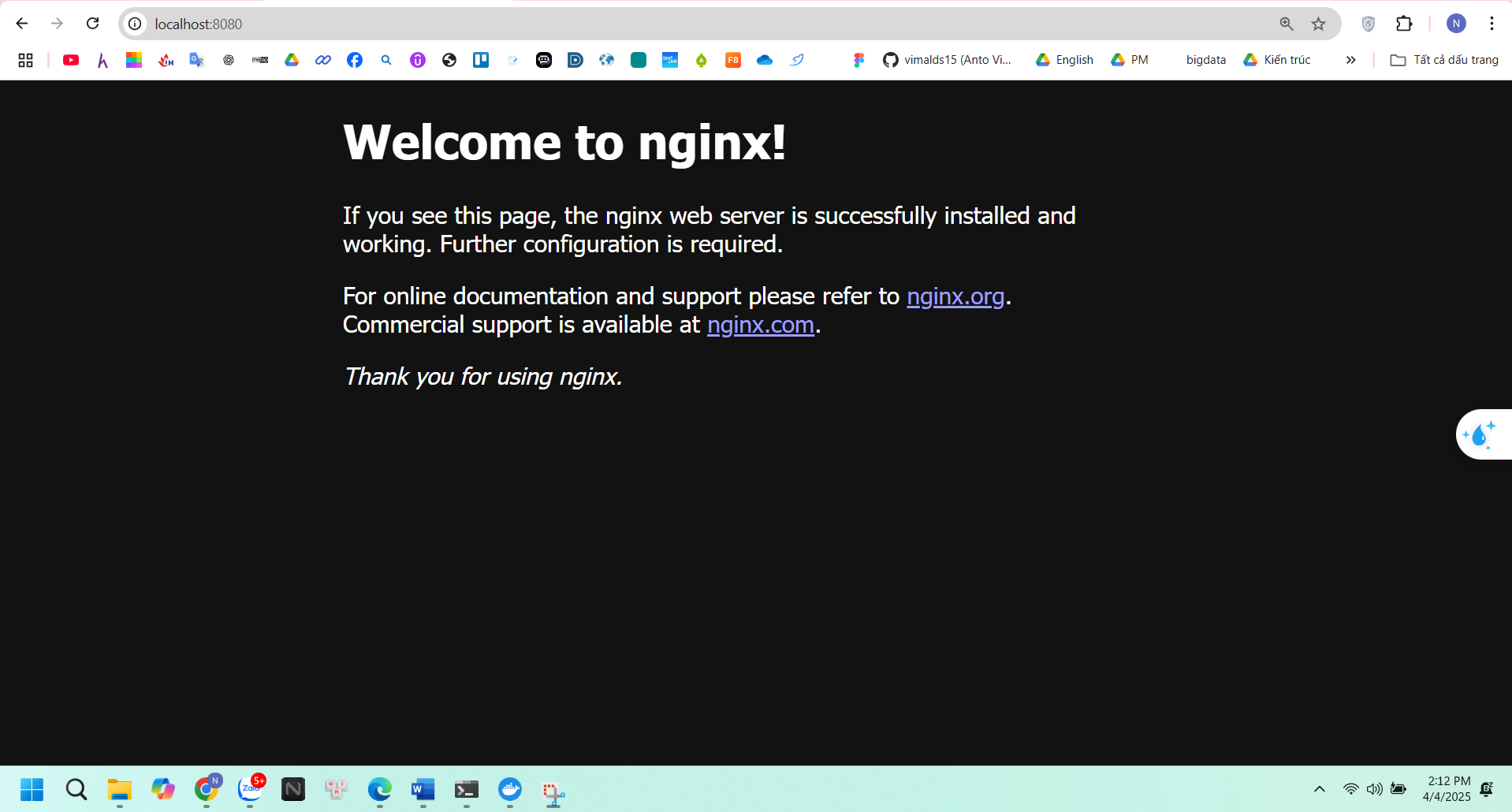
# **Chạy một container từ image, ánh xạ cổng của container với cổng của máy chủ (host) và chạy container ở chế độ nền.**

docker run -d -p 8080:80 nginx

* **Ảnh minh chứng**

****

Sau khi container được chạy như trên, truy cập nginx bằng cách vào trình duyệt và nhập <http://localhost:8080>.



* **Giải thích**

docker run: Lệnh này được dùng để tạo và khởi động một container mới từ một image Docker.

-d: Tùy chọn này chỉ định rằng container sẽ chạy ở chế độ nền (detached mode). Điều này có nghĩa là container sẽ chạy ngầm và không chiếm dụng terminal của bạn.

-p 8080:80: Tùy chọn này ánh xạ cổng của container với cổng của máy chủ. Cấu trúc 8080:80 có nghĩa là:

* 8080: Cổng trên máy chủ (host) mà bạn sẽ truy cập.
* 80: Cổng trong container mà ứng dụng (nghĩa là nginx trong trường hợp này) đang lắng nghe.

Vậy, khi bạn truy cập http://localhost:8080, bạn sẽ được chuyển tiếp đến cổng 80 của container, nơi nginx đang chạy.

nginx: Đây là tên của Docker image bạn muốn sử dụng để tạo container. Trong trường hợp này, bạn sử dụng image chính thức của nginx.

# **Xem thông tin chi tiết về một container**

docker inspect <container\_id>

* **Ảnh minh chứng**

****

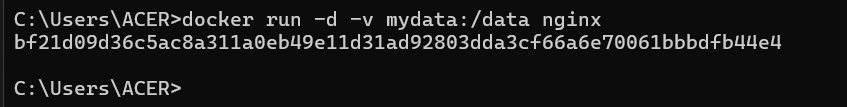
* **Giải thích**

Lệnh này cung cấp thông tin chi tiết về một container, image, volume, hoặc network. Kết quả trả về là một cấu trúc dữ liệu JSON, cung cấp rất nhiều thông tin về container hoặc đối tượng Docker.

# **Tạo volume**

docker run -d -v mydata:/data nginx

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh docker run -d -v mydata:/data nginx sẽ chạy một container nginx ở chế độ nền, và ánh xạ một volume từ hệ thống Docker vào thư mục /data trong container.

Trong đó:

-v mydata:/data: Đây là tùy chọn để gắn kết (mount) một volume vào container.

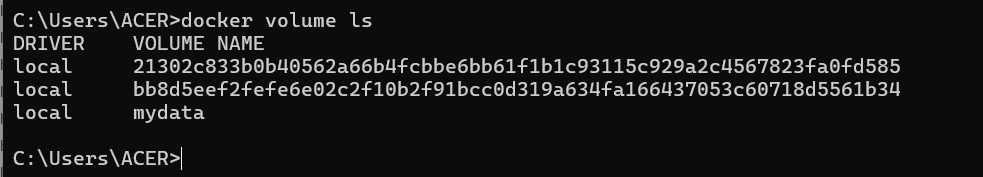
* mydata: Tên của volume. Nếu volume này chưa tồn tại, Docker sẽ tự động tạo nó.
* /data: Đường dẫn thư mục trong container mà volume sẽ được gắn vào. Tất cả dữ liệu được ghi vào thư mục này trong container sẽ được lưu trong volume mydata và có thể được truy xuất lại sau khi container dừng hoặc xóa.

**Volume**: Volume là một cách để lưu trữ dữ liệu lâu dài, vì nó tồn tại ngoài phạm vi của container. Nếu container bị xóa, volume vẫn tồn tại và có thể được sử dụng lại trong các container khác.

# **Liệt kê tất cả các volumes hiện có trên hệ thống Docker**

docker volume ls

* **Ảnh minh chứng**

****

Kết quả sẽ thấy có “mydata” được tạo ở bước trên

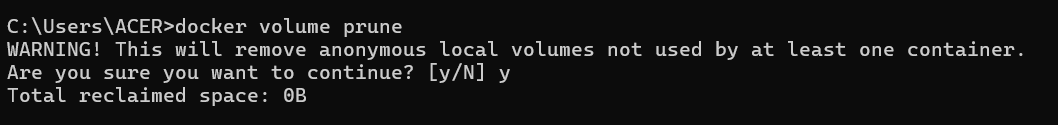
* **Giải thích**

Lệnh này hiển thị danh sách tất cả các volumes mà Docker đang quản lý trên hệ thống. Volumes là các khu vực lưu trữ dữ liệu lâu dài, dùng để lưu trữ thông tin hoặc tệp tin mà người dùng muốn giữ lại, ngay cả khi container bị xóa hoặc dừng.

# **Xóa tất cả các volumes không còn được sử dụng**

docker volume prune

* **Ảnh minh chứng**

****

* **Giải thích**

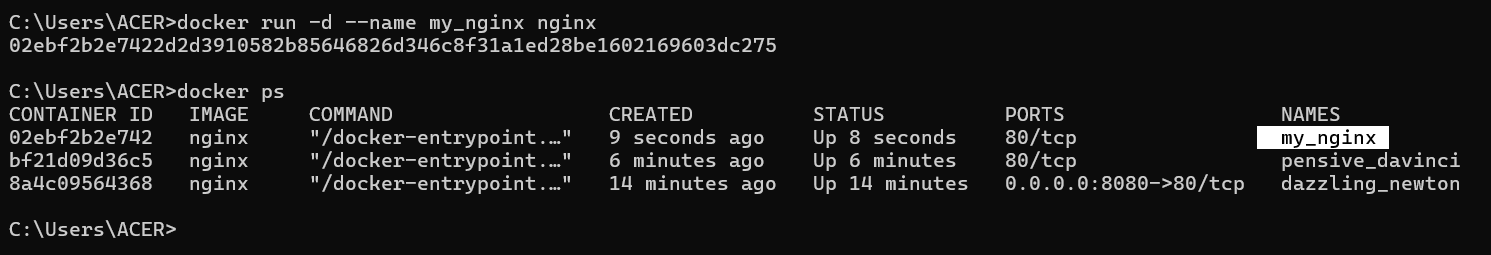
Lệnh này sẽ xóa tất cả các dangling volumes. Đây là các volumes không còn được bất kỳ container nào sử dụng, tức là chúng không được ánh xạ hoặc không còn liên kết với bất kỳ container nào.

Dangling volumes: Là các volumes không còn được sử dụng nữa nhưng vẫn tồn tại trên hệ thống, chiếm không gian đĩa mà không mang lại giá trị sử dụng.

# **Chạy container ở chế độ nền và đặt tên cho container đó**

docker run -d --name my\_nginx nginx

* **Ảnh minh chứng**

****

* **Giải thích**

docker run: Lệnh này được sử dụng để tạo và chạy một container mới từ một image Docker.

-d: Tùy chọn này chỉ định rằng container sẽ chạy ở chế độ nền (detached mode), có nghĩa là container sẽ chạy ngầm mà không chiếm dụng terminal của bạn.

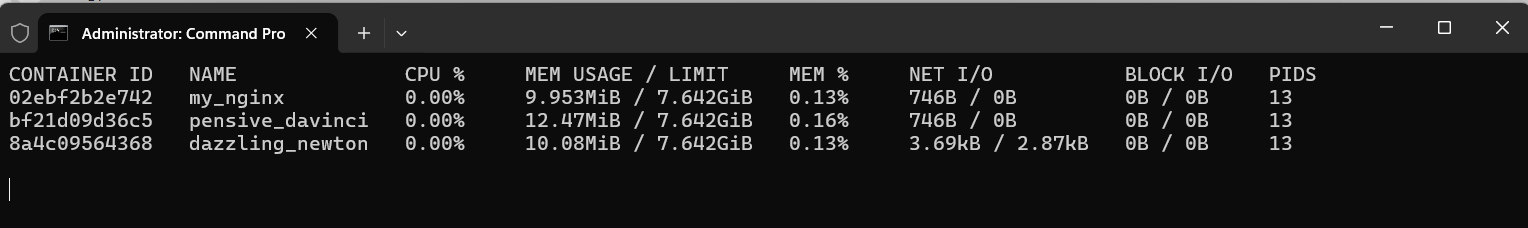
--name my\_nginx: Tùy chọn này dùng để đặt tên cho container. Trong trường hợp này, container sẽ được đặt tên là my\_nginx. Việc đặt tên này giúp dễ dàng quản lý và tham chiếu container sau này, thay vì phải sử dụng ID container dài.

nginx: Đây là tên của Docker image muốn sử dụng để tạo container.

# **Hiển thị thông tin thống kê tài nguyên**

docker stats

* **Ảnh minh chứng**

****

* **Giải thích**

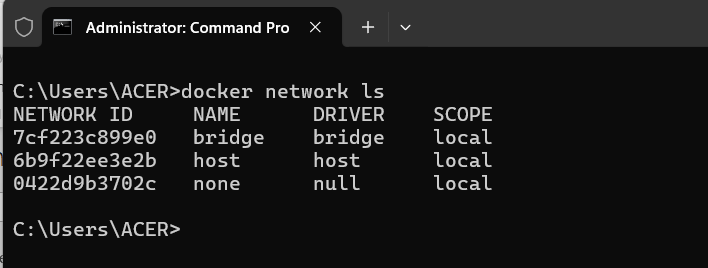
Lệnh này sẽ cung cấp thông tin thống kê về các container Docker đang chạy, bao gồm:

* CPU usage (Sử dụng CPU)
* Memory usage (Sử dụng bộ nhớ)
* Network I/O (Băng thông mạng)
* Block I/O (Sử dụng đĩa)
* PIDs (Số lượng tiến trình trong container)

# **Liệt kê tất cả các mạng (network) Docker hiện có**

docker network ls

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh này sẽ hiển thị danh sách các mạng Docker đang tồn tại trên hệ thống. Docker sử dụng các mạng để kết nối các container lại với nhau và với các tài nguyên khác như cơ sở dữ liệu hoặc web server.

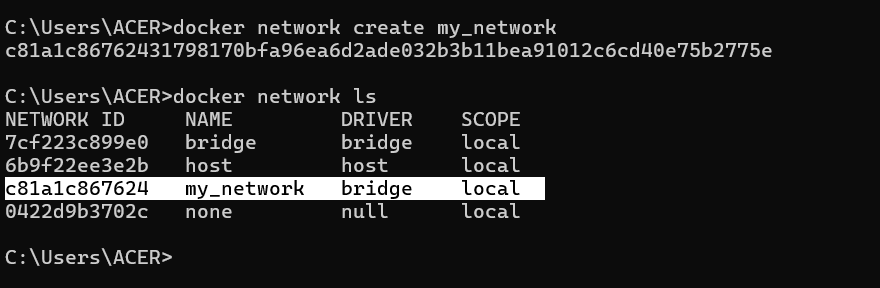
Trong đó:

* NETWORK ID: ID của mạng.
* NAME: Tên của mạng.
* DRIVER: Loại driver mạng được sử dụng (chẳng hạn như bridge, host, overlay, v.v.).
* SCOPE: Phạm vi của mạng, ví dụ như local cho mạng cục bộ hoặc global cho mạng có thể sử dụng trong các môi trường phân tán.

# **Tạo một mạng Docker mới**

docker network create my\_network

* **Ảnh minh chứng**

****

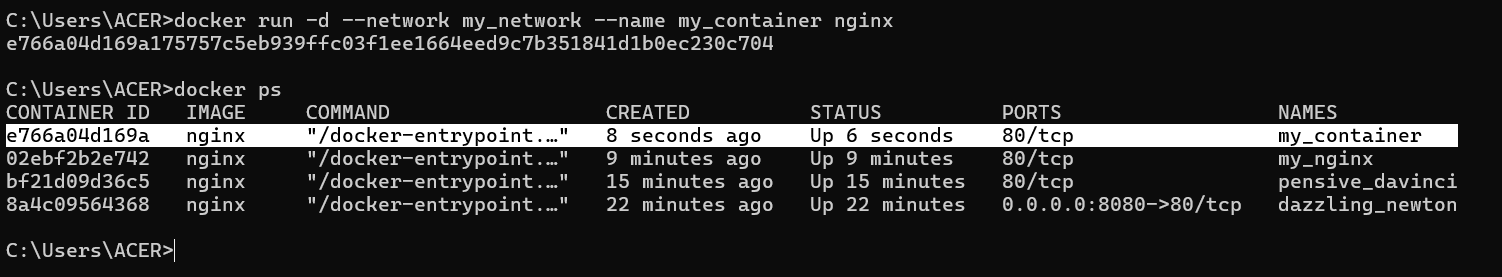
* **Giải thích**

Lệnh này tạo ra một mạng mới trong Docker, cho phép kết nối các container lại với nhau hoặc với các tài nguyên khác. Mạng Docker giúp các container giao tiếp với nhau qua mạng ảo.

# **Chạy container và kết nối với mạng**

docker run -d --network my\_network --name my\_container nginx

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh này sẽ chạy một container nginx ở chế độ nền, kết nối nó với mạng Docker my\_network, và đặt tên cho container là my\_container.

Trong đó:

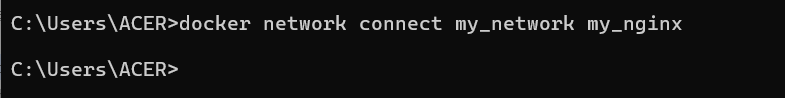
--network my\_network: Tùy chọn này kết nối container vào một mạng Docker đã tồn tại. Trong trường hợp này, container sẽ được kết nối vào mạng my\_network. Việc kết nối container vào một mạng giúp các container có thể giao tiếp với nhau qua mạng nội bộ.

--name my\_container: Tùy chọn này đặt tên cho container là my\_container.

# **Kết nối một container đang chạy vào mạng**

docker network connect my\_network my\_nginx

* **Ảnh minh chứng**

****

* **Giải thích**

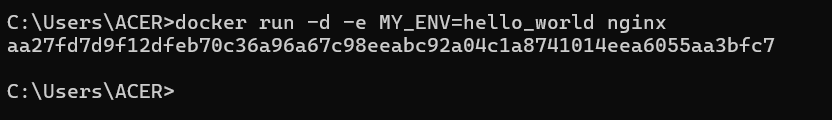
Lệnh này dùng để kết nối một container đã chạy vào một mạng Docker. Sau khi lệnh này được thực hiện, container sẽ có thể giao tiếp với các container khác trong cùng mạng đó.

Tức là: Lệnh này sẽ kết nối container my\_nginx vào mạng my\_network. Sau khi thực hiện lệnh, container my\_nginx sẽ có thể giao tiếp với các container khác trong mạng my\_network.

# **Chạy container ở chế độ nền và có thiết lập môi trường**

docker run -d -e MY\_ENV=hello\_world nginx

* **Ảnh minh chứng**

****

* **Giải thích**

Lệnh docker run -d -e MY\_ENV=hello\_world nginx sẽ chạy một container nginx ở chế độ nền (detached), đồng thời thiết lập một biến môi trường MY\_ENV với giá trị hello\_world trong container. Trong đó:

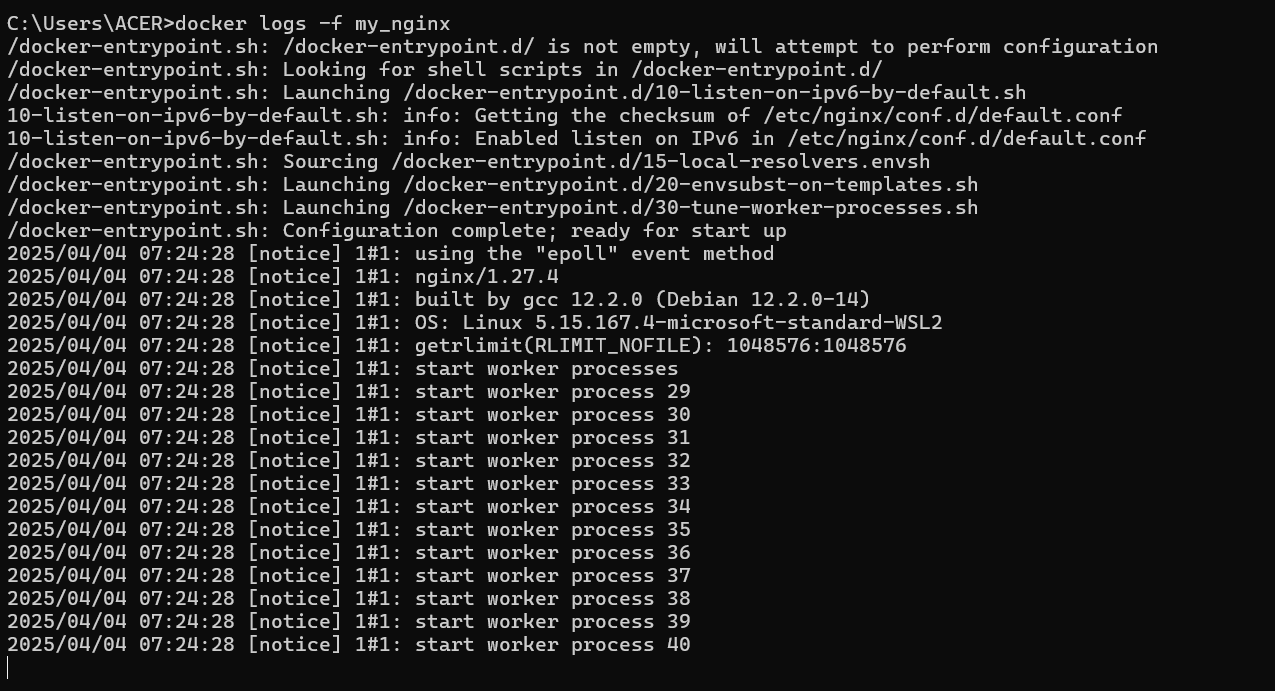
-e MY\_ENV=hello\_world: Tùy chọn này thiết lập một biến môi trường trong container. Trong trường hợp này, biến môi trường có tên là MY\_ENV và giá trị của nó là hello\_world. Các biến môi trường có thể được sử dụng trong các ứng dụng chạy trong container để cấu hình hoặc thay đổi hành vi của ứng dụng. Có thể chỉ định nhiều biến môi trường bằng cách thêm nhiều tùy chọn -e.

Biến môi trường thường được sử dụng để cấu hình ứng dụng mà không cần phải thay đổi mã nguồn. Ví dụ, bạn có thể sử dụng biến môi trường để cấu hình các thông số như cơ sở dữ liệu, cổng ứng dụng, v.v.

# **Hiển thị các log của container và cập nhật liên tục (theo dõi thời gian thực) khi có log mới được ghi vào container.**

docker logs -f my\_nginx

* **Ảnh minh chứng**

****

* **Giải thích**

**docker logs**: Lệnh này dùng để **xem các log đầu ra của một container**. Các log này thường chứa thông tin về trạng thái và hoạt động của container hoặc ứng dụng bên trong container.

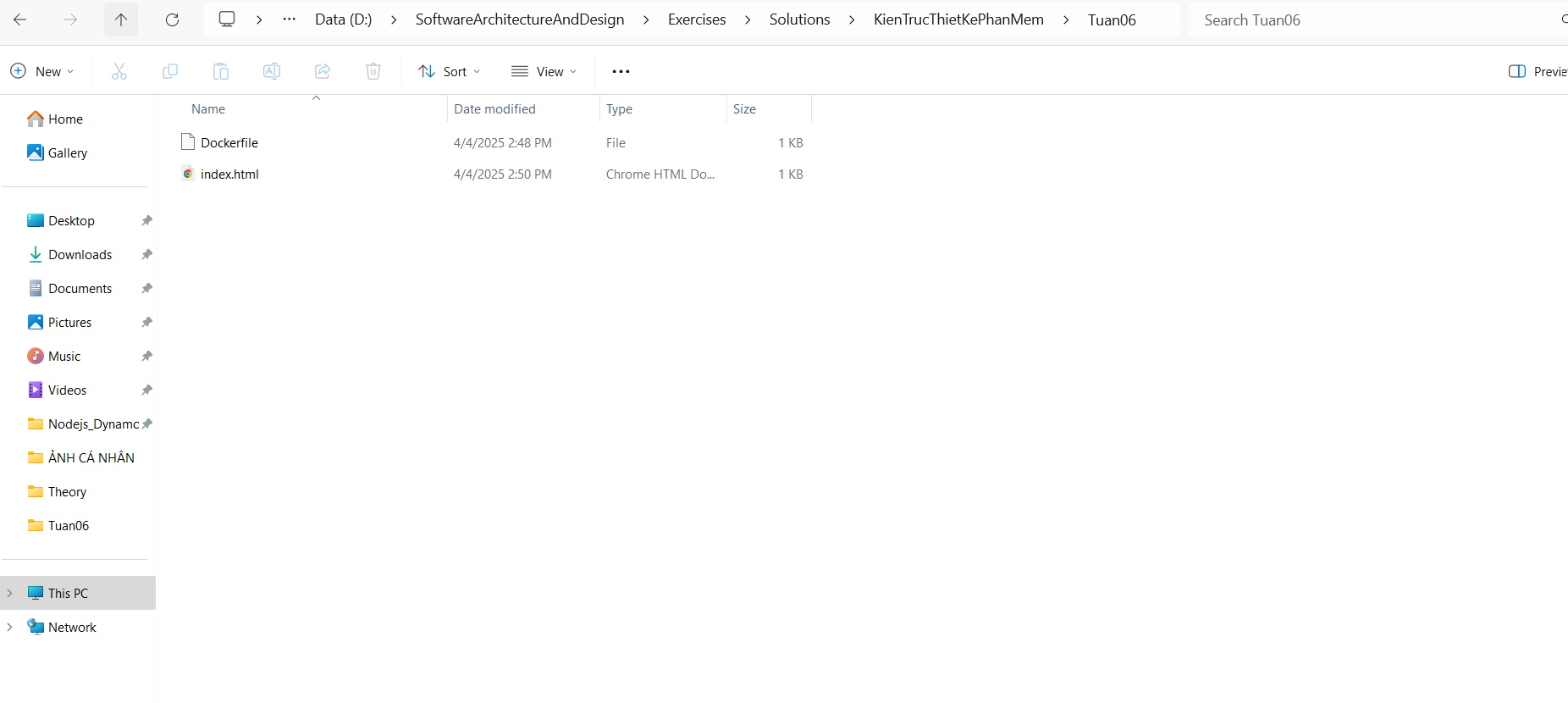
**-f**: Tùy chọn -f (hoặc --follow) sẽ làm cho lệnh **theo dõi log liên tục**, tức là nó sẽ tiếp tục hiển thị các log mới ngay khi chúng được tạo ra trong khi container đang chạy.

**my\_nginx**: Đây là tên của container mà bạn muốn xem log. Trong trường hợp này, bạn đang xem log của container my\_nginx.

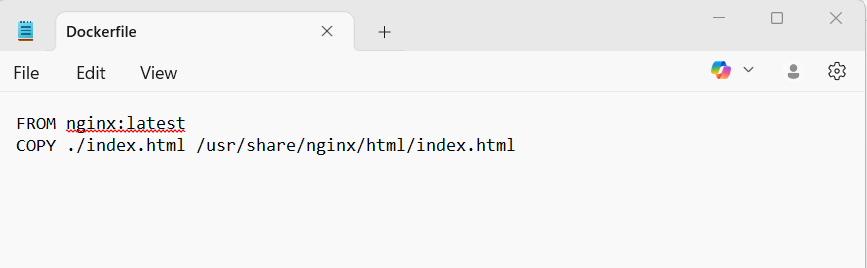
# **Tạo docker file**

* **Ảnh minh chứng**

**Giả sứ có 1 trang index cơ bản.**

****

Tạo dockerfile với nội dung dockerfile như sau:

****

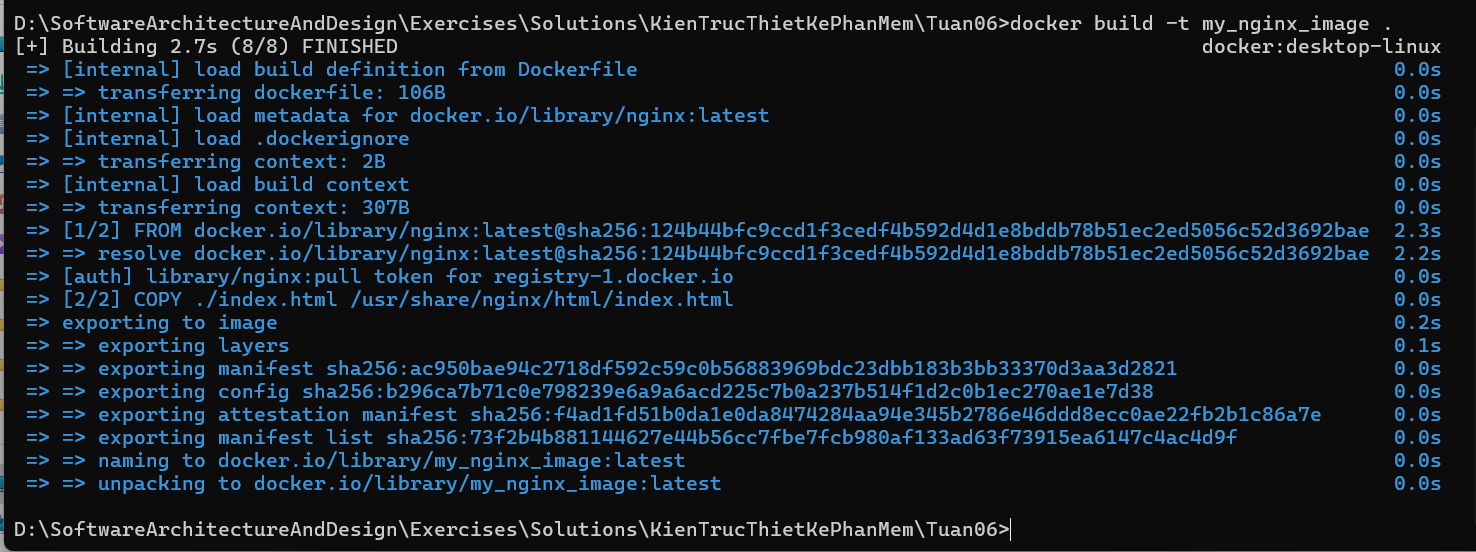
* **Giải thích**

# **Xây dựng một image Docker từ Dockerfile**

docker build -t my\_nginx\_image .

* Ảnh minh chứng

Sau khi có docker file tiến hành build image từ dockerfile



* **Giải thích**

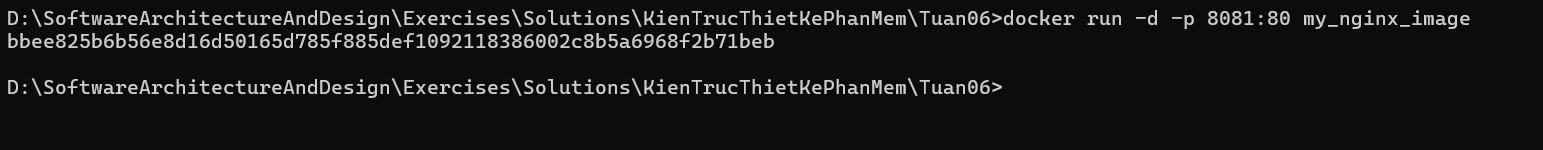
docker build: Lệnh này được sử dụng để xây dựng một Docker image từ một Dockerfile. Dockerfile là một tệp văn bản chứa các hướng dẫn để Docker biết cách tạo một image.

-t my\_nginx\_image: Tùy chọn -t được sử dụng để đặt tên (tag) cho image Docker. Trong trường hợp này, image sẽ có tên là my\_nginx\_image.

. : Đây là đường dẫn đến thư mục chứa Dockerfile mà bạn muốn sử dụng để xây dựng image. Dấu chấm (.) đại diện cho thư mục hiện tại, tức là Dockerfile phải nằm trong thư mục mà đang chạy lệnh này. Nếu Dockerfile ở thư mục khác, cần cung cấp đường dẫn chính xác tới thư mục đó.

# **Test sau khi build từ dockerfile**

docker run -d -p 8081:80 my\_nginx\_image



Sau khi chạy container từ image. Vào trình duyệt truy cập localhost:8081 để chạy project

****

# **docker run hello-world**

* **Ảnh minh chứng**
* **Giải thích**

# **docker run hello-world**

* **Ảnh minh chứng**
* **Giải thích**