ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
QUẢN TRỊ MẠNG VÀ HỆ THỐNG

TÌM HIỂU VÀ TRIỂN KHAI DOCKER

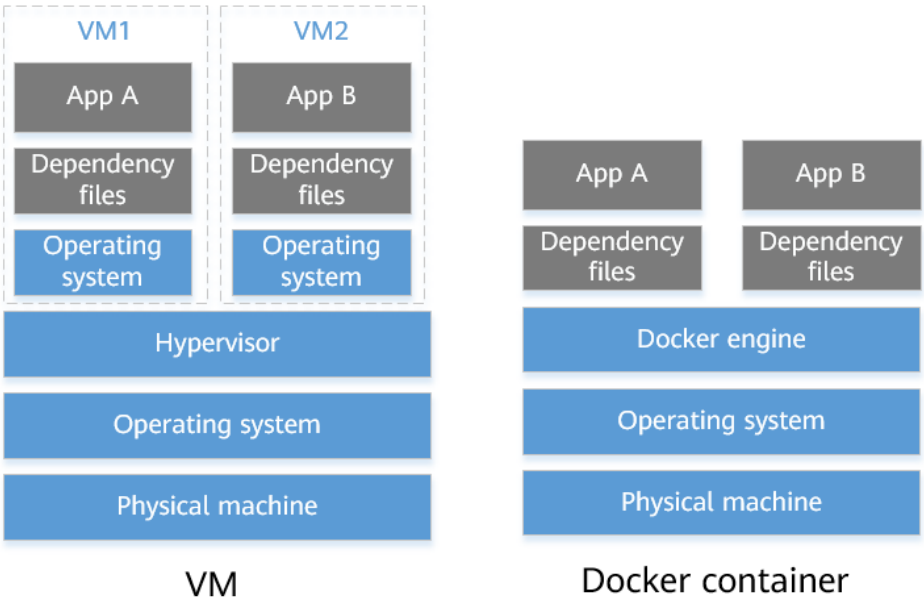
NHÓM 12

Giáo viên hướng dẫn - Bùi Thanh Bình

Sinh viên thực hiện:

19522196 – Lê Đức Thắng
21520712 – Tô Lý Tiến Đạt
21521034 – Ngô Tuấn Kiệt

1. Cơ sở lý thuyết:



Container là một phương pháp ảo hóa phần mềm giúp cho việc triển khai ứng dụng diễn ra nhanh chóng và dễ dàng hơn. Một container bao gồm tất cả các phần mềm và thư viện cần thiết để chạy một ứng dụng cụ thể, và có thể chạy trên bất kỳ nền tảng nào mà nó được xây dựng cho.

Các dịch vụ trong một hệ thống phần mềm thường phải chia sẻ các tệp. Nếu một dịch vụ gặp sự cố hoặc bị tấn công, nó có thể ảnh hưởng đến các dịch vụ khác trong hệ thống, gây ra sự cố hoặc suy yếu hiệu suất.

Với container, mỗi dịch vụ sẽ được chạy trong một container riêng biệt, được cô lập hoàn toàn với các container khác trong hệ thống. Điều này có nghĩa là nếu một container bị sự cố hoặc bị tấn công, nó sẽ không ảnh hưởng đến các container khác trong hệ thống. Điều này giúp tăng tính bảo mật, độ tin cậy và hiệu suất của hệ thống phần mềm.

Ngoài ra, container cũng giúp đơn giản hóa việc triển khai và quản lý ứng dụng. Khi các ứng dụng được đóng gói trong các container, việc triển khai trở nên đơn giản và dễ dàng hơn, vì các container có thể chạy trên bất kỳ nền tảng nào mà nó được xây dựng cho. Container cũng cho phép việc quản lý và giám sát hệ thống phần mềm trở nên đơn giản hơn, vì các container có thể được khởi động, tắt hoặc mở rộng một cách độc lập với các container khác trong hệ thống.

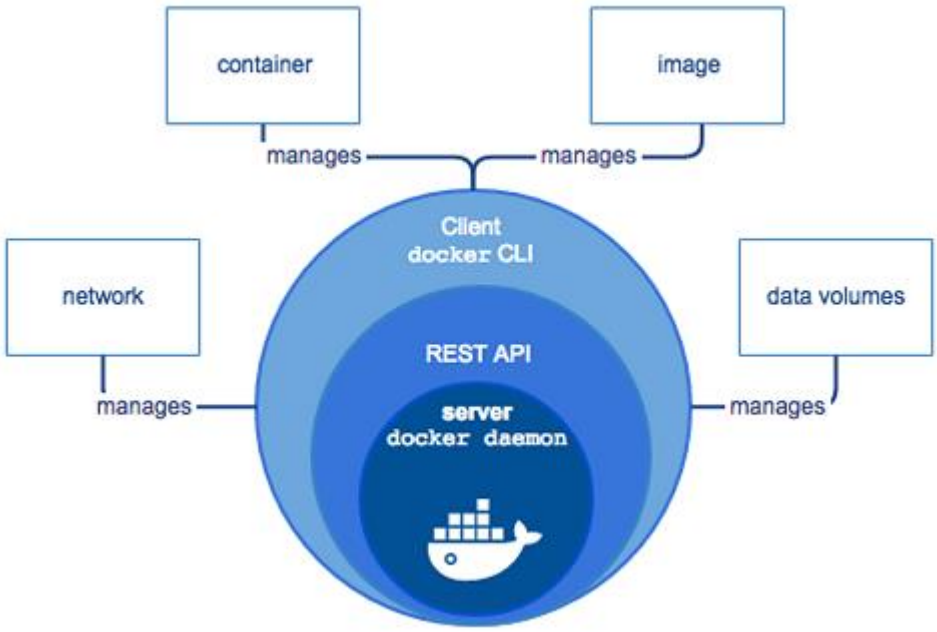
Docker là một nền tảng ảo hóa phần mềm cho phép các ứng dụng chạy trên các container. Các container được xây dựng trên cơ sở của hệ điều hành host và chứa các phần mềm, môi trường cần thiết để chạy các ứng dụng đó. Docker cung cấp một cách để đóng gói, chuyển giao và triển khai các ứng dụng một cách nhanh chóng và dễ dàng, bằng cách tạo ra các container độc lập với hệ điều hành host.

Một container là một môi trường ảo hóa nhẹ, nơi mà các ứng dụng và các phụ thuộc (dependencies) của chúng có thể được đóng gói vào một gói duy nhất và chạy độc lập với các container khác. Các container Docker được xây dựng trên cơ sở của hệ điều hành host, do đó chúng có thể chạy trên bất kỳ hệ điều hành nào có hỗ trợ Docker.

Docker cũng cung cấp một cách để quản lý các container, cho phép người dùng tạo, bắt đầu, dừng và xóa các container một cách dễ dàng. Docker cũng hỗ trợ quản lý các mạng và lưu trữ, cho phép các container truy cập tài nguyên một cách dễ dàng.

2. Mô hình triển khai:

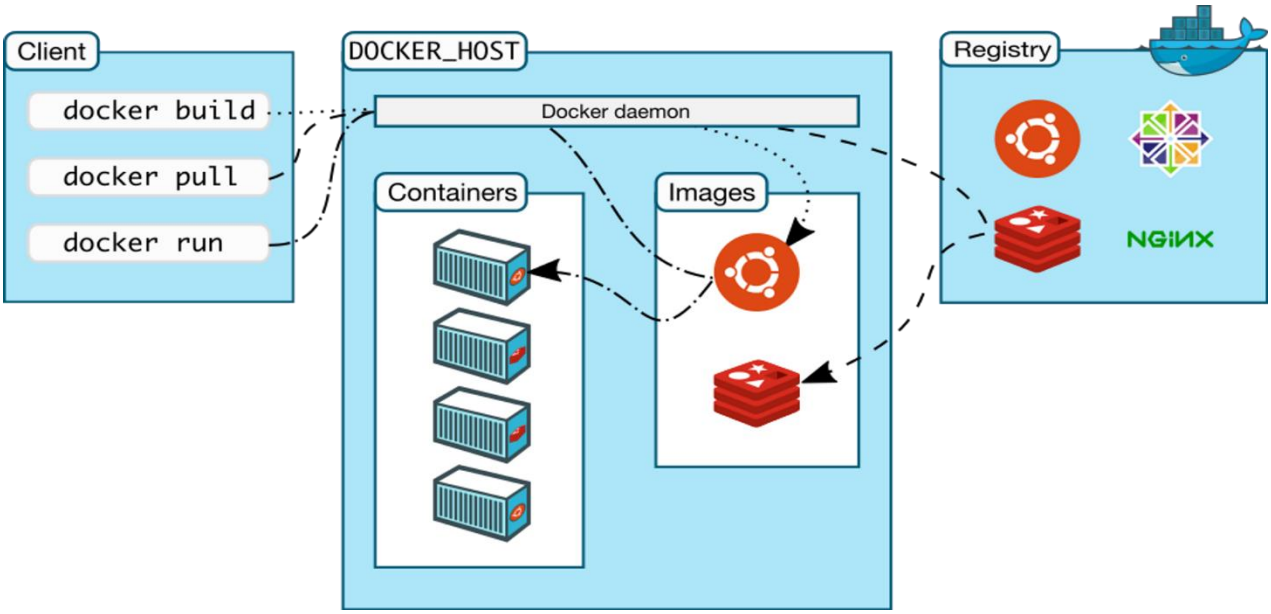
Docker engine và các thành phần:



- **Docker daemon** : Một quy trình nền liên tục quản lý Docker images, containers, networks, và storage volumes. Docker daemon liên tục lắng nghe các yêu cầu API Docker và xử lý chúng.
- **Docker Engine REST API**: Một API được các ứng dụng sử dụng để tương tác với Docker daemon. Nó có thể được truy cập bởi một máy khách HTTP.
- **Docker CLI**: (Command line interface): Cung cấp tập hợp các lệnh để quản lý các container, images và các thực thể khác của Docker

Docker client giao tiếp với Docker daemon, trình nền này thực hiện việc xây dựng, chạy cũng như phân phối các Docker container.

Kiến trúc Docker:



- Docker's Client: Người dung tương tác với Docker thông qua client. Khi bất kỳ lệnh docker nào chạy, client gửi chúng đến Docker daemon để thực hiện các lệnh đó.
- Docker's Host: Cung cấp môi trường hoàn chỉnh để thực thi và chạy các ứng dụng (gồm Docker daemon và các docker objects như: images, containers, networks, storage).
- Docker's Registry: Lưu trữ và tải xuống các images. Các lệnh phổ biến là docker push, docker pull và docker run

Về docker objects:

- Images: chứa dữ liệu mô tả khả năng và nhu cầu của container. Thường được dùng để lưu trữ và vận chuyển ứng dụng.
- Containers
- Network: là cách mà các containers biệt lập giao tiếp với nhau.
    - Bridge : Khi các ứng dụng đang chạy trên các containers độc lập.
    - Host: Loại bỏ sự cô lập mạng giữa các containers docker và docker host.
    - Overlay: Cho phép các dịch vụ bầy đàn giao tiếp với nhau. Được sử dụng khi muốn các containers chạy trên các docker hosts khác nhau hoặc muốn hình thành các swarm services.
    - None: Vô hiệu hoá kết nối mạng
    - Macvlan: Gắn địa chỉ mac cho các containers để làm chúng trông giống các thiết bị vật lý.
- Storage: Dữ liệu không bền bỉ, nó có thể chết bất cứ khi nào containers không chạy. Hơn nữa không dễ dàng để chuyển dữ liệu này. Do đó, đối với lưu trữ liên tục, Docker cung cấp 4 tuỳ chọn:
    - Data volumes: Cung cấp khả năng lưu trữ liên tục với khả năng đổi tên volume, liệt kê volume và liệt kê container được liên kết với volume.
    - Volume container: Làm volume container độc lập với ứng dụng container để có thể chia sẻ nó trên nhiều containers.
    - Directory Mounts: Bất kỳ thư mục nào trên máy chủ đều có thể được sử dụng làm nguồn cho volume.
    - Storage Plugins: Cung cấp khả năng kết nối với các nền tảng lưu trữ bên ngoài.

Triển khai Docker:

Để triển khai Docker, người dùng cần cài đặt Docker trên hệ điều hành host và tạo các container từ các Docker image. Docker image là một gói đóng gói đầy đủ của một ứng dụng hoặc một phần mềm bao gồm tất cả các phụ thuộc (dependencies) và tài nguyên (resources) cần thiết để chạy ứng dụng đó trên Docker. Nó bao gồm một tập hợp các lệnh và tệp cấu hình, và được sử dụng để tạo ra các container Docker. Người dùng có thể tạo các ảnh Docker của riêng mình hoặc sử dụng các ảnh có sẵn trên Docker Hub.

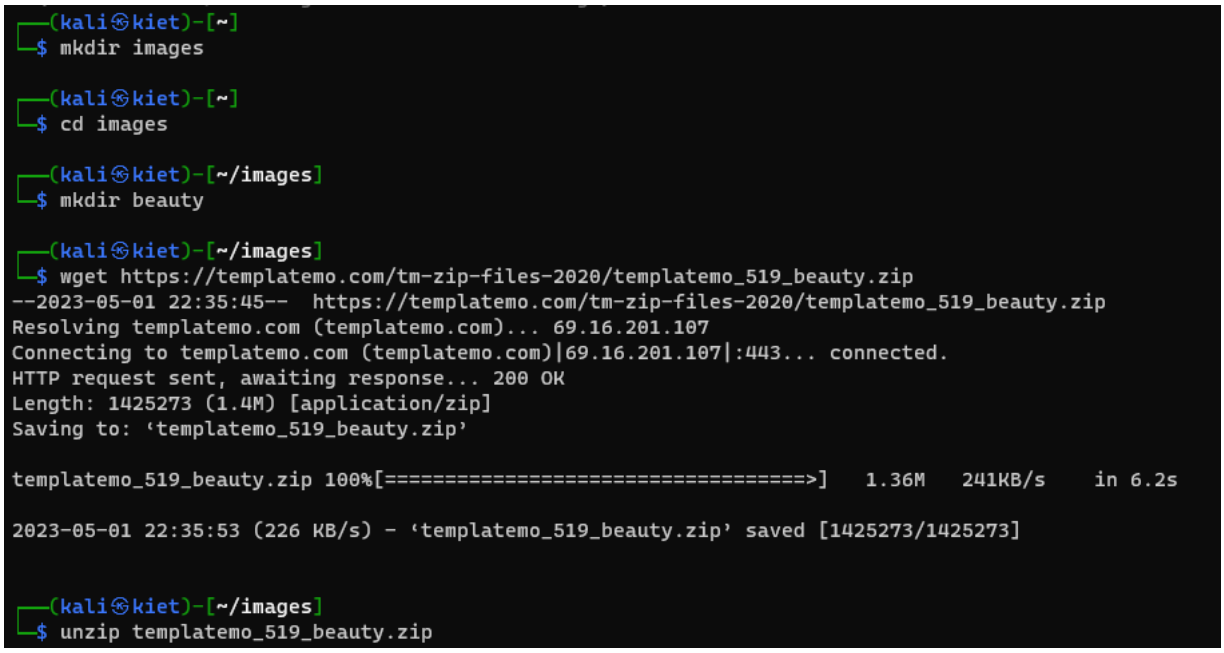
Sau khi tạo các ảnh Docker, người dùng có thể tạo các container từ các Docker image. Các container Docker có thể chạy độc lập hoặc được kết nối với các container khác hoặc với mạng bên ngoài.

Các container Docker cũng có thể được quản lý bằng Docker Compose, một công cụ cho phép người dùng định nghĩa và quản lý nhiều container trong một tệp cấu hình đơn. Docker Compose cho phép người dùng xây dựng các môi trường đa container phức tạp một cách dễ dàng và nhanh chóng.

3. Cài đặt, cấu hình:

Nhóm em sử dụng một template website ở trên mạng (src: [Free Template 519 Beauty (templatemo.com)](templatemo.com))

Đầu tiên sử dụng wget tải file về folder images sau đó giải nén



Sau khi giải nén thành công, chuyển đến thư mục templatemo_519_beauty và nén tất cả nội dung vào một tệp tar và nén bằng gzip.

```
┌──(kali㉿kiet)-[~/images]
└─$ ls
beauty  templatemo_519_beauty  templatemo_519_beauty.zip

┌──(kali㉿kiet)-[~/images]
└─$ cd templatemo_519_beauty

┌──(kali㉿kiet)-[~/images/templatemo_519_beauty]
└─$ tar czvf beauty.tar.gz *
```

Chuyển thư mục beauty.tar.gz đến thư mục beauty sau đó tạo Dọckerfile ở thư mục beauty.

```
┌──(kali㉿kiet)-[~/images/templatemo_519_beauty]
└─$ ls
beauty.tar.gz  css  img  index.html  js  slick  webfonts

┌──(kali㉿kiet)-[~/images/templatemo_519_beauty]
└─$ mv beauty.tar.gz ../

┌──(kali㉿kiet)-[~/images/templatemo_519_beauty]
└─$ cd ..

┌──(kali㉿kiet)-[~/images]
└─$ ls
beauty  beauty.tar.gz  templatemo_519_beauty  templatemo_519_beauty.zip

┌──(kali㉿kiet)-[~/images]
└─$ mv beauty.tar.gz beauty

┌──(kali㉿kiet)-[~/images]
└─$ cd beauty

┌──(kali㉿kiet)-[~/images/beauty]
└─$ ls
beauty.tar.gz

┌──(kali㉿kiet)-[~/images/beauty]
└─$ vim Dockerfile
```

Giải thích Dockerfile:

Tạo Docker image cho ứng dụng web sử dụng Apache trên hệ điều hành Kali Linux.

- Dòng đầu tiên khai báo rằng Docker sẽ sử dụng hệ điều hành Kali Linux phiên bản mới nhất để xây dựng ảnh.
- Tiếp theo là các LABEL, được sử dụng để định nghĩa thông tin về Docker image này như "NT132" và "Project".
- Dòng ENV khai báo một biến môi trường, DEBIAN_FRONTEND, với giá trị "noninteractive". Biến này cho phép cài đặt các gói phần mềm mà không cần xác nhận từ người dùng.

- Dòng RUN cập nhật kho lưu trữ và cài đặt gói phần mềm Apache2. Dòng CMD khai báo lệnh chạy để khởi động Apache2.
- Dòng EXPOSE khai báo cổng mạng 80 của máy ảo Docker sẽ được sử dụng để truy cập trang web.
- Dòng WORKDIR thiết lập thư mục làm việc mặc định của ứng dụng web là "/var/www/html".
- Dòng VOLUME khai báo rằng thư mục /var/log/apache2 trên máy Docker sẽ được chia sẻ với máy chủ chứa ứng dụng Docker.
- Dòng ADD để thêm tệp tin beauty.tar.gz vào thư mục "/var/www/html" trên máy Docker. Tệp tin này chứa các tài nguyên của ứng dụng web được sử dụng bởi Apache2.

```
FROM kalilinux/kali-rolling
LABEL "NT132"="Nhom12"
LABEL "Project"="beauty"
ENV DEBIAN_FRONTEND=noninteractive
RUN apt update && apt install apache2 -y
CMD ["/usr/sbin/apache2ctl","-D","FOREGROUND"]
EXPOSE 80
WORKDIR /var/www/html
VOLUME /var/log/apache2
ADD beauty.tar.gz /var/www/html
~
~
~
```

Lệnh docker build để xây dựng một docker images từ một docker file, -t beautyimg đặt tên cho docker image và “.” để chỉ rằng dockerfile ở thư mục hiện tại

```
┌──(kali㉿kiet)-[~/images/beauty]
└─$ docker build -t beautyimg .
[+] Building 63.1s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 321B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/kalilinux/kali-rolling:latest                                            3.4s
 => [auth] kalilinux/kali-rolling:pull token for registry-1.docker.io                                               0.0s
 => [1/4] FROM docker.io/kalilinux/kali-rolling@sha256:4d3bcc90e4433284645dc5dec6a5de54aafba8ff3313cbe1079e33bf09  4.6s
 => => resolve docker.io/kalilinux/kali-rolling@sha256:4d3bcc90e4433284645dc5dec6a5de54aafba8ff3313cbe1079e33bf09  0.0s
 => => sha256:4d3bcc90e4433284645dc5dec6a5de54aafba8ff3313cbe1079e33bf09c8b9e9 1.19kB / 1.19kB                      0.0s
 => => sha256:94cc1959e4c264df076cf85fa665e2f2f2a55f2ce6c38c0bb8a545ccc01919ef 429B / 429B                          0.0s
 => => sha256:3e684bc3055fc18615776d88f84b50d8eeb8ab815476fbb77dad23290c501ef5 2.85kB / 2.85kB                      0.0s
 => => sha256:ef61f594885b9b7d0638eb2f2f6dabda41655fb444487e1f73fa17f456e0d613 51.17MB / 51.17MB                     2.9s
 => => extracting sha256:ef61f594885b9b7d0638eb2f2f6dabda41655fb444487e1f73fa17f456e0d613                            1.4s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 1.43MB                                                                                 0.0s
 => [2/4] RUN apt update && apt install apache2 -y                                                                 53.6s
 => [3/4] WORKDIR 80                                                                                                0.0s
 => [4/4] ADD beauty.tar.gz /var/www/html                                                                           0.1s
 => exporting to image                                                                                              1.2s
 => => exporting layers                                                                                             1.2s
 => => writing image sha256:a2c000cf04839fc01acc9f47db4b8fb52f843efd4d7c738f16552b4d65cba25d                        0.0s
 => => naming to docker.io/library/beautyimg                                                                        0.0s

┌──(kali㉿kiet)-[~/images/beauty]
└─$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED              SIZE
beautyimg         latest    a2c000cf0483   About a minute ago   268MB
getting-started   latest    6a3739f47b6c   2 weeks ago          318MB
```

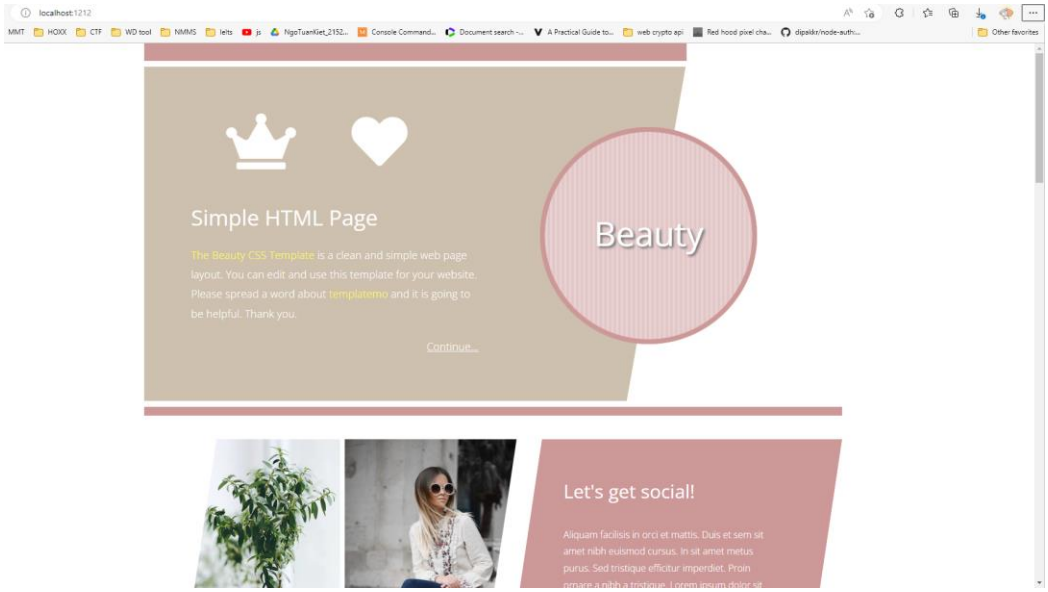
"docker run": đây là lệnh để chạy một container từ một ảnh Docker.
"-d": đây là tùy chọn để chạy container ở chế độ daemon (nền).
"--name beautywebsite": đây là tùy chọn để đặt tên cho container mới được tạo ra.
"-p 1212:80": đây là tùy chọn để chuyển tiếp cổng mạng. Nó sẽ ánh xạ cổng 1212 của máy host với cổng 80 của container Docker.
"beautyimg": đây là tên của Docker image được sử dụng để tạo container.

```
┌──(kali㉿kiet)-[~/images/beauty]
└─$ docker run -d --name beautywebsite -p 1212:80 beautyimg
033253f0d7f4e663dcc1bef73cb6a5f42411d5c3defd4b076d066c2d5a256571

┌──(kali㉿kiet)-[~/images/beauty]
└─$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                   NAMES
033253f0d7f4   beautyimg   "/usr/sbin/apache2ct…"   20 seconds ago   Up 18 seconds   0.0.0.0:1212->80/tcp   beautywebsite

└─$
```

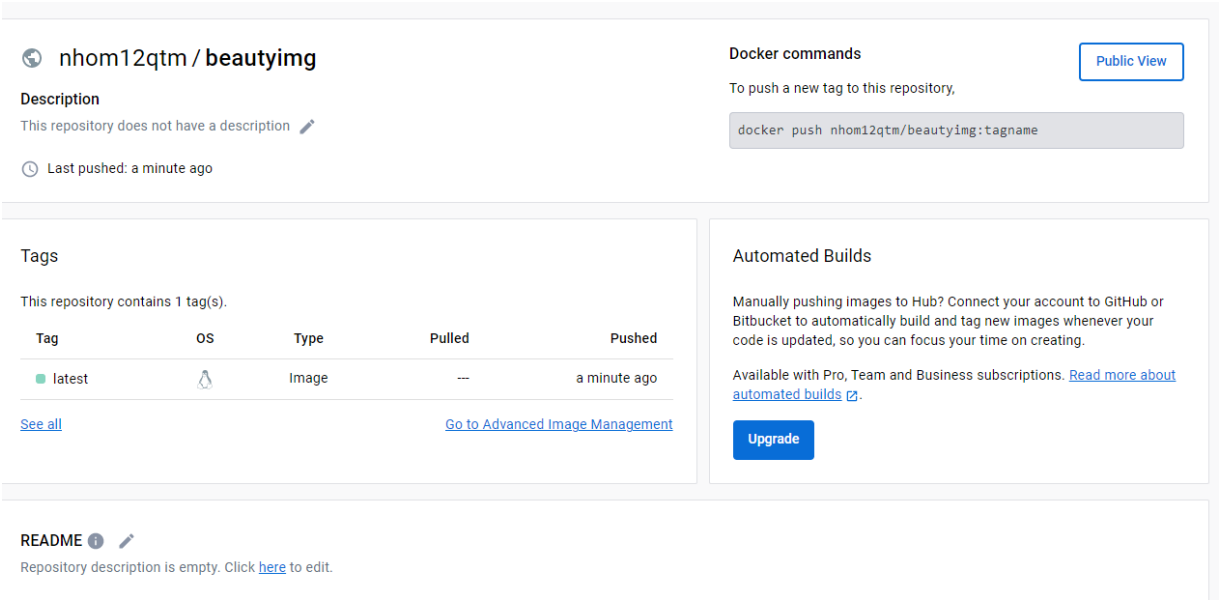
Kết quả trên local host:



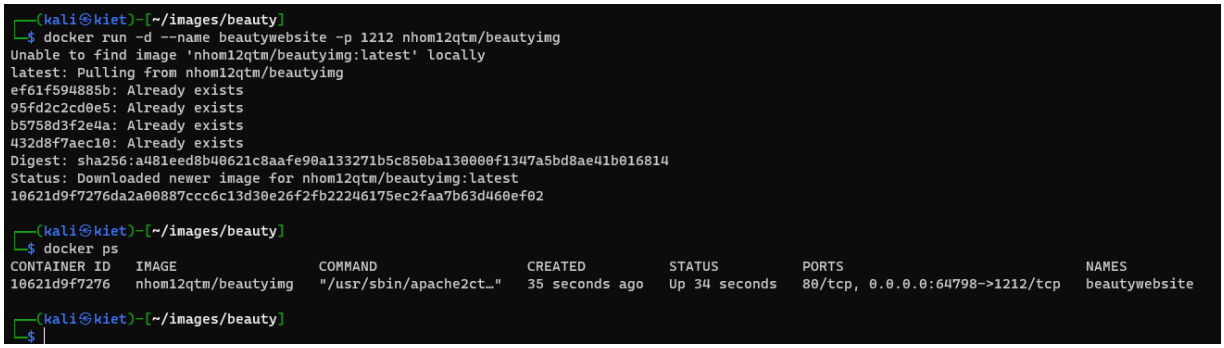
Docker push:

```
┌──(kali㉿kiet)-[~/images/beauty]
└─$ docker build -t nhom12qtm/beautyimg .
[+] Building 2.4s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 38B                                                                     0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/kalilinux/kali-rolling:latest                                2.2s
 => [auth] kalilinux/kali-rolling:pull token for registry-1.docker.io                                   0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 36B                                                                        0.0s
 => [1/4] FROM docker.io/kalilinux/kali-rolling@sha256:4d3bcc90e4433284645dc5dec6a5de54aafba8ff3313cbe1079e33  0.0s
 => CACHED [2/4] RUN apt update && apt install apache2 -y                                               0.0s
 => CACHED [3/4] WORKDIR 80                                                                             0.0s
 => CACHED [4/4] ADD beauty.tar.gz /var/www/html                                                        0.0s
 => exporting to image                                                                                  0.0s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:a2c000cf04839fc01acc9f47db4b8fb52f843efd4d7c738f16552b4d65cba25d            0.0s
 => => naming to docker.io/nhom12qtm/beautyimg                                                          0.0s

┌──(kali㉿kiet)-[~/images/beauty]
└─$ docker push nhom12qtm/beautyimg
Using default tag: latest
The push refers to repository [docker.io/nhom12qtm/beautyimg]
8ea797f39cab: Pushed
fbb2f025ba84: Pushed
b2db2acc1ad1: Pushed
eb4a528895c0: Mounted from kalilinux/kali-rolling
latest: digest: sha256:a481eed8b40621c8aafe90a133271b5c850ba130000f1347a5bd8ae41b016814 size: 1159
```



Docker pull:



4. Kết quả và kết luận:

Ta có thể thấy Docker giúp người sử dụng có thể đóng gói ứng dụng vào các container Docker, và triển khai chúng trên bất kỳ môi trường nào một cách dễ dàng và đáng tin cậy.

Docker giúp giải quyết nhiều vấn đề trong việc phát triển và triển khai phần mềm, bao gồm việc cung cấp môi trường đồng nhất giữa các bên khác nhau, hỗ trợ đóng gói ứng dụng và các phụ thuộc của nó vào một gói đóng gói duy nhất, cũng như đơn giản hóa quá trình phát triển và triển khai ứng dụng.

Kết luận, Docker đã trở thành một công cụ quan trọng cho các nhà phát triển và kỹ sư hệ thống để quản lý các ứng dụng phức tạp trên nhiều môi trường khác nhau. Sử dụng Docker giúp tăng tính đồng nhất và khả năng

mở rộng trong quá trình phát triển và triển khai phần mềm, đồng thời giảm thiểu các sự cố và tăng tính sẵn sàng của ứng dụng khi triển khai trên các môi trường khác nhau.

5. Tài liệu tham khảo:

1. *Docker Docs: How to build, share, and run applications*. (2023, May 12). Docker Documentation. https://docs.docker.com/
2. Templatemo. (n.d.). *Free Template 519 Beauty*. Templatemo. https://templatemo.com/tm-519-beauty
3. Đức, P. (2023). Docker Tutorial – Docker Architecture: Why is it important? (5/11). *Viblo*. https://viblo.asia/p/docker-tutorial-docker-architecture-why-is-it-important-511-RnB5pVpbZPG