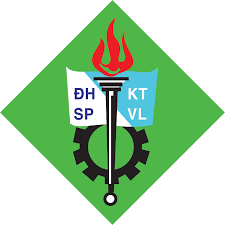
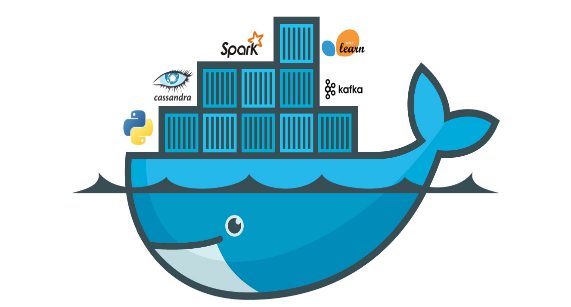
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KĨ THUẬT VĨNH LONG**

**KHOA: CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN CÔNG NGHỆ THÔNG TIN 2**

**TÌM HIỂU VỀ DOCKER VÀ XÂY DỰNG**

**HỆ THỐNG MÁY CHỦ WEB**

**Sinh viên thực hiện:** Nguyễn Minh Châu MSSV: 18004012

Lê Thị Thu Cẩm MSSV: 18004010

**Khóa :** 43

**Lớp:** ĐH.CÔNG NGHỆ THÔNG TIN 2018 ( Khóa : 2018 - 2022)

**Giáo Viên Hướng Dẫn:** ThS.Trần Thu Mai

**Vĩnh Long, tháng 10 năm 2020**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KĨ THUẬT VĨNH LONG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**PHIẾU GIAO ĐỒ ÁN 2**

**Tên đồ án:**

**TÌM HIỂU VỀ DOCKER VÀ XÂY DỰNG**

**HỆ THỐNG MÁY CHỦ WEB**

**Phương pháp đánh giá:** 🗖 Báo cáo trước hội đồng 🗖 Chấm thuyết minh

**Ngày giao đồ án :** Ngày……..Tháng………Năm………...

**Ngày hoàn thành đồ án:** Ngày……..Tháng………Năm………...

**Sinh viên thực hiện đồ án:**

**Họ và tên Sinh Viên:** Nguyễn Minh châu MSSV: 18004012

Lê Thị Thu Cẩm MSSV: 18004010

Vĩnh Long, Ngày……..Tháng……Năm……….

Trưởng Khoa/Bộ Môn Người Hướng Dẫn

ThS. Trần Thu Mai

NHẬN XÉT VÀ ĐÁNH GIÁ ĐIỂM CỦA NGƯỜI HƯỚNG DẪN

* Ý thức thực hiện: ………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………
* Nội dung thực hiện: ………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………
* Hình thức trình bày: ………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………
* Tổng hợp kết quả: ………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

Vĩnh Long, Ngày……..Tháng……Năm……….

Người Hướng Dẫn

ThS.Trần Thu Mai

**LỜI CẢM ƠN**

Trong suốt quá trình học tập, đặc biệt là khoảng thời gian hoàn thành đồ án, em đã nhận được những lời động viên từ phía gia đình, bạn bè và sự giúp đỡ ân cần của quý thầy cô trong Khoa Công Nghệ thông tin, trường ĐH Sư phạm Kĩ Thuật Vĩnh Long. Với lòng biết ơn sâu sắc em xin cảm ơn được gửi lời cảm ơn chân thành tới.

Em xin được gửi lời cảm ơn chân thành đến ThS.Trần Thu Mai. Người đã trực tiếp định hướng, truyền đạt kiến thức và những kinh nghiệm thực tế quý báu của mình, hết sức quan tâm hướng dẫn em thực hiện đồ án 2 với đề tài “TÌM HIỂU VỀ DOCKER VÀ XÂY DỰNG HỆ THỐNG MÁY CHỦ WEB”.

Em cũng chân thành cảm ơn Khoa Công Nghệ Thông Tin đã tạo điều kiện cho em thực hiện đồ án này.

Trong công tác chuẩn bị và hoàn thành đồ án, mặc dù đã rất kỹ lưỡng và tập trung, nhưng em chắc rằng sẽ có những thiếu sót, mong nhận được sự thông cảm của Cô. Em xin chân thành cảm ơn Cô đã hỗ trợ em trong đồ án học kì này!

Sinh viên thực hiện: Nguyễn Minh Châu MSSV: 18004012

Lê Thị Thu Cẩm MSSV: 18004010

**MỤC LỤC**

[LỜI MỞ ĐẦU 1](#_Toc58236168)

[CHƯƠNG 1 : TỔNG QUAN VỀ ĐỀ TÀI 2](#_Toc58236169)

[1.1 Tên đề tài: 2](#_Toc58236170)

[1.2 Lý do chọn đề tài: 2](#_Toc58236171)

[1.3 Mục tiêu cần đạt: 2](#_Toc58236172)

[CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT 3](#_Toc58236173)

[2.1 Tổng quan về Docker 3](#_Toc58236174)

[2.1.1 Tổng quan: 3](#_Toc58236175)

[2.1.2 Nền tảng: 3](#_Toc58236176)

[2.1.3 Công cụ: 3](#_Toc58236177)

[2.1.4 Kiến trúc: 4](#_Toc58236178)

[2.2 Cài đặt Docker trên Ubuntu Sever 5](#_Toc58236179)

[2.2.1 Đăng nhập vào Clould Google (https://cloud.google.com): 5](#_Toc58236180)

[2.2.2 Tạo máy ảo ubuntu 20.04 LTS 6](#_Toc58236181)

[2.2.3 Cài đặt Docker: 9](#_Toc58236182)

[2.2.4 Kiểm tra Docker: 9](#_Toc58236183)

[2.3 Các lệnh cơ bản trong Docker 9](#_Toc58236184)

[2.3.1 Tải image về server: 9](#_Toc58236185)

[2.3.2 Hiển thị danh sách các images : 10](#_Toc58236186)

[2.3.3 Xóa một images: 11](#_Toc58236187)

[2.3.4 Chạy một image: 11](#_Toc58236188)

[2.3.1 Liệt kê các container: 16](#_Toc58236189)

[2.3.2 Dừng container đang chạy: 17](#_Toc58236190)

[2.3.3 Khởi động lại container đã dừng: 18](#_Toc58236191)

[2.3.4 Truy cập vào một container đang chạy: 18](#_Toc58236192)

[2.3.5 Xóa container không còn sử dụng: 19](#_Toc58236193)

[2.3.6 Lưu một images : 19](#_Toc58236194)

[2.3.7 Tải images từ file đã lưu: 20](#_Toc58236195)

[2.3.8 Export container : 20](#_Toc58236196)

[2.3.9 Import container : 21](#_Toc58236197)

[2.3.10 Tạo thêm một số thuộc tính cho Docker: 21](#_Toc58236198)

[2.4 Dockerfile: 25](#_Toc58236199)

[2.4.1 Quy tắc viết chỉ thị Dockerfile: 25](#_Toc58236200)

[2.4.2 Các chỉ thị Dockerfile: 25](#_Toc58236201)

[2.5 Docker-compose: 28](#_Toc58236202)

[2.5.1 Giới thiệu: 28](#_Toc58236203)

[2.5.2 Sử dụng docker-compose: 28](#_Toc58236204)

[2.5.3 Quy trình soạn thảo Docker: 29](#_Toc58236205)

[CHƯƠNG 3 : XÂY DỰNG MÁY CHỦ WEB 32](#_Toc58236206)

[3.1 Đăng kí tên miền: 32](#_Toc58236207)

[3.2 Xây dựng máy chủ web: 34](#_Toc58236208)

[CHƯƠNG 4 : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 40](#_Toc58236209)

[4.1 Kết luận: 40](#_Toc58236210)

[4.1.1 Ưu điểm đã làm được: 40](#_Toc58236211)

[4.1.2 Hạn chế: 40](#_Toc58236212)

[4.2 Hướng Phát triển: 40](#_Toc58236213)

**MỤC LỤC HÌNH**

[Hình 2‑1: Mô hình Docker tổng quát 4](#_Toc56114364)

[Hình 2‑2: Mô hình container 4](#_Toc56114365)

[Hình 2‑3: Giao diện đăng nhập google clould 5](#_Toc56114366)

[Hình 2‑4: Chọn VM instances 6](#_Toc56114367)

[Hình 2‑5: Tạo VM instances 6](#_Toc56114368)

[Hình 2‑6: Chọn cấu hình VM 7](#_Toc56114369)

[Hình 2‑7: VM đã được tạo 8](#_Toc56114370)

[Hình 2‑8: Đăng nhập vào VM 8](#_Toc56114371)

[Hình 2‑9: Cài đặt docker 9](#_Toc56114372)

[Hình 2‑10: Kiểm tra cài đặt 9](#_Toc56114373)

[Hình 2‑11: Tải image ubuntu 10](#_Toc56114374)

[Hình 2‑12: Hiển thị tất cả image 10](#_Toc56114375)

[Hình 2‑13:Xóa image debian 11](#_Toc56114376)

[Hình 2‑14: Chạy một images ubuntu với tên ubuntu và chạy dưới nền 15](#_Toc56114377)

[Hình 2‑15:Chạy một images ubuntu với tên ubuntu-1 và truy cập ngay 15](#_Toc56114378)

[Hình 2‑16:Chạy một images ubuntu với tên ubuntu-2, tên vm test và truy cập ngay 15](#_Toc56114379)

[Hình 2‑17:Chạy một images nginx với tên docker-nginx , port 80 và chạy dưới nền 16](#_Toc56114380)

[Hình 2‑18:Kiểm tra image đã chạy 16](#_Toc56114381)

[Hình 2‑19:Liệt kê tất cả các container 17](#_Toc56114382)

[Hình 2‑20:Dừng container docker-nginx 17](#_Toc56114383)

[Hình 2‑21:Kiểm tra lại image đã dừng 17](#_Toc56114384)

[Hình 2‑22:Khởi động lại container và kiểm tra 18](#_Toc56114385)

[Hình 2‑23:Kiểm tra container docker-nginx 18](#_Toc56114386)

[Hình 2‑24:Truy cập vào container ubuntu-2 19](#_Toc56114387)

[Hình 2‑25:Xóa container ubuntu và kiểm tra 19](#_Toc56114388)

[Hình 2‑26: Lưu image nginx 20](#_Toc56114389)

[Hình 2‑27:Tải image nginx từ file 20](#_Toc56114390)

[Hình 2‑28:Xuất container thành file nginx:web.tar.gz 21](#_Toc56114391)

[Hình 2‑29:Khôi phục lại image 21](#_Toc56114392)

[Hình 2‑30:Tạo images centos được cài đặt “git” 28](#_Toc56114393)

[Hình 2‑31:Hiển thị image đã được tạo 28](#_Toc56114394)

[Hình 2‑32:Kiểm tra "git" đã được cài đặt 28](#_Toc56114395)

[Hình 3‑1:Trang web freemon.com 33](#_Toc56114396)

[Hình 3‑2:Nhập tên miền muốn đăng kí 33](#_Toc56114397)

[Hình 3‑3:Thêm tên miền vào giỏ hàng 34](#_Toc56114398)

[Hình 3‑4:Thanh toán mua tên miền 34](#_Toc56114399)

[Hình 3‑5:Tên miền đã được mua 34](#_Toc56114400)

[Hình 3‑6:Thêm địa chỉ ip của host vào tên miền 35](#_Toc56114401)

[Hình 3‑7:Cấu trúc thư mục để tạo server web 35](#_Toc56114402)

[Hình 3‑8:Nội dung file index.php 36](#_Toc56114403)

[Hình 3‑9:Dockerfile mysql 36](#_Toc56114404)

[Hình 3‑10:Dockerfile phpmyadmin 36](#_Toc56114405)

[Hình 3‑11:Dockerfile php:apache 37](#_Toc56114406)

[Hình 3‑12:File docker-compose 38](#_Toc56114407)

[Hình 3‑13:Có 3 container đang chạy 39](#_Toc56114408)

[Hình 3‑14:Kết nối thành công với cơ sở dữ liệu mysql 39](#_Toc56114409)

[Hình 3‑15:Đăng nhập thành công cơ sở dữ liệu mysql 39](#_Toc56114410)

# LỜI MỞ ĐẦU

Với Machine Learning và Data Science, Docker sẽ hỗ trợ cho việc xây dựng các Container mà trong đó có chứa các thư viện, gói phần mềm cần thiết cho việc học tập, nghiên cứu các dự án về Machine Learning và Data Science. Các Container này có thể được đóng gói thành các Images và chia sẻ cho những người có nhu cầu nghiên cứu và học tập về Machine Learning và Data Science một cách nhanh chóng. Chính nhờ điều này sẽ giúp cho các nhóm nghiên cứu phối hợp làm việc với nhau dễ dàng hơn nhờ một hệ thống các phần mềm và thư viện chuẩn và thống nhất. Việc cập nhật các thư viện, phần mềm trong các Docker Images cũng được thực hiện dễ dàng.

Ngoài ra, Docker cũng có thể được ứng trong các môi trường phát triền phần mềm khác như ASP.NET Core, Python, Java,... Việc sử dụng Docker cho phát triển phần mềm có thể giúp tạo ra môi trường sạch cho việc phát triển và kiểm thử phần mềm. Nhờ vậy sẽ giảm thiểu các lỗi phần mềm do không đồng nhất các thư viện giữa các thiết bị dùng để phát triển và kiểm thử.

# TỔNG QUAN VỀ ĐỀ TÀI

## Tên đề tài:

Tìm hiểu về Docker và xây dựng máy chủ Web.

## Lý do chọn đề tài:

Em đã học xong môn Phần mềm Mã Nguồn Mở và Mạng máy tính. Có được kiến thức cơ bản để xây dựng hệ thống ảo hóa Docker. Docker là hệ thống ảo hóa chủ yếu chạy trên nền tảng mã nguồn mở và có nhiều ứng dụng cho việc làm sau khi em ra trường. Nên em chọn tìm hiểu về Docker để tạo thêm cơ hội việc làm về sau.

## Mục tiêu cần đạt:

* Hiểu được Docker
* Biết được các lệnh cơ bản trong Docker
* Tạo được một máy chủ chạy Web

# CƠ SỞ LÝ THUYẾT

## Tổng quan về Docker

### Tổng quan:

Docker là một nền tảng mở để phát triển, vận chuyển và chạy các ứng dụng. Docker cho phép bạn tách các ứng dụng khỏi cơ sở hạ tầng để bạn có thể phân phối phần mềm một cách nhanh chóng. Với Docker, bạn có thể quản lý cơ sở hạ tầng của mình giống như cách bạn quản lý các ứng dụng của mình. Bằng cách tận dụng các phương pháp luận của Docker để vận chuyển, thử nghiệm và triển khai mã một cách nhanh chóng, bạn có thể giảm đáng kể độ trễ giữa việc viết mã và chạy mã trong sản xuất.

### Nền tảng:

Docker cung cấp khả năng đóng gói và chạy ứng dụng trong một môi trường cô lập lỏng lẻo được gọi là vùng chứa. Sự cô lập và bảo mật cho phép bạn chạy nhiều vùng chứa đồng thời trên một máy chủ nhất định. Các vùng chứa có trọng lượng nhẹ vì chúng không cần tải thêm của một siêu giám sát, nhưng chạy trực tiếp trong nhân của máy chủ. Điều này có nghĩa là bạn có thể chạy nhiều vùng chứa hơn trên một tổ hợp phần cứng nhất định so với khi bạn đang sử dụng máy ảo. Bạn thậm chí có thể chạy các vùng chứa Docker trong các máy chủ thực sự là máy ảo!

### Công cụ:

Docker Engine là một ứng dụng khách-máy chủ với các thành phần chính sau:

Máy chủ là một loại chương trình chạy lâu dài được gọi là tiến trình daemon ( dockerdlệnh).

Một API REST chỉ định các giao diện mà các chương trình có thể sử dụng để nói chuyện với daemon và hướng dẫn nó phải làm gì.

Máy khách giao diện dòng lệnh (CLI).

CLI sử dụng API Docker REST để kiểm soát hoặc tương tác với trình nền Docker thông qua các lệnh CLI kịch bản hoặc trực tiếp. Nhiều ứng dụng Docker khác sử dụng API và CLI cơ bản.

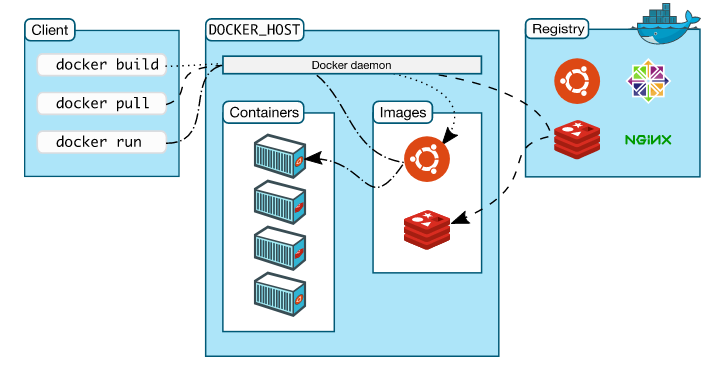
Daemon tạo và quản lý các đối tượng Docker , chẳng hạn như hình ảnh, vùng chứa, mạng và ổ đĩa.



Hình ‑: Mô hình Docker tổng quát

### Kiến trúc:

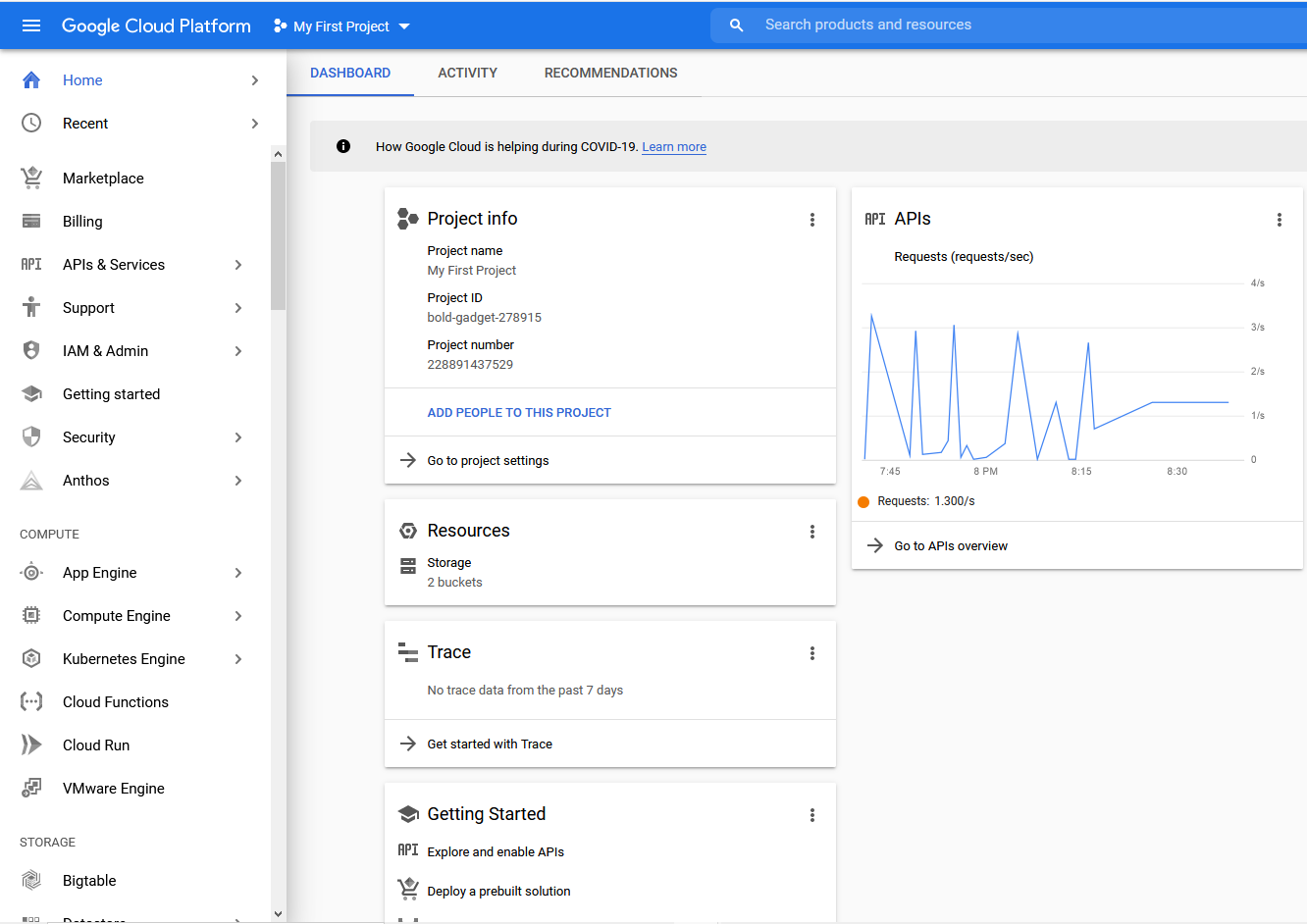
Docker sử dụng kiến ​​trúc máy khách-máy chủ. Ứng dụng khách Docker nói chuyện với Docker daemon , trình nền này thực hiện công việc xây dựng, chạy và phân phối các vùng chứa Docker của bạn. Ứng dụng khách Docker và daemon có thể chạy trên cùng một hệ thống hoặc bạn có thể kết nối ứng dụng khách Docker với trình nền Docker từ xa. Ứng dụng khách Docker và daemon giao tiếp bằng API REST, qua ổ cắm UNIX hoặc giao diện mạng.



Hình ‑: Mô hình container

## Cài đặt Docker trên Ubuntu Sever

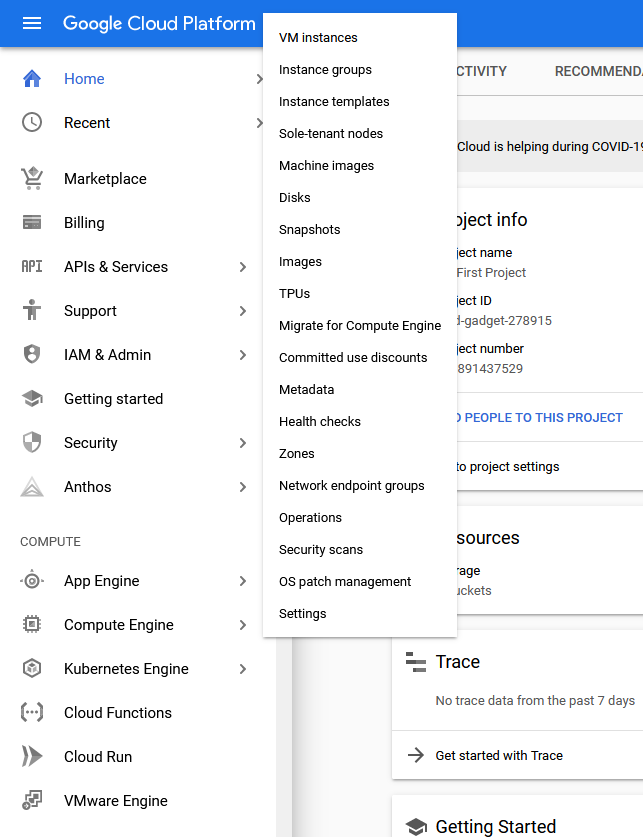
### Đăng nhập vào Clould Google (<https://cloud.google.com>):



Hình ‑: Giao diện đăng nhập google clould

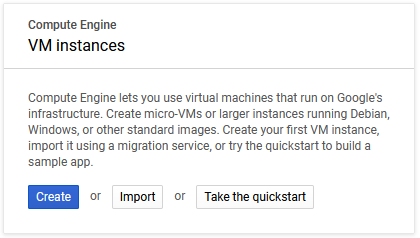
### Tạo máy ảo ubuntu 20.04 LTS

#### Bước 1: Chọn Compute Engine -> Chọn VM instances



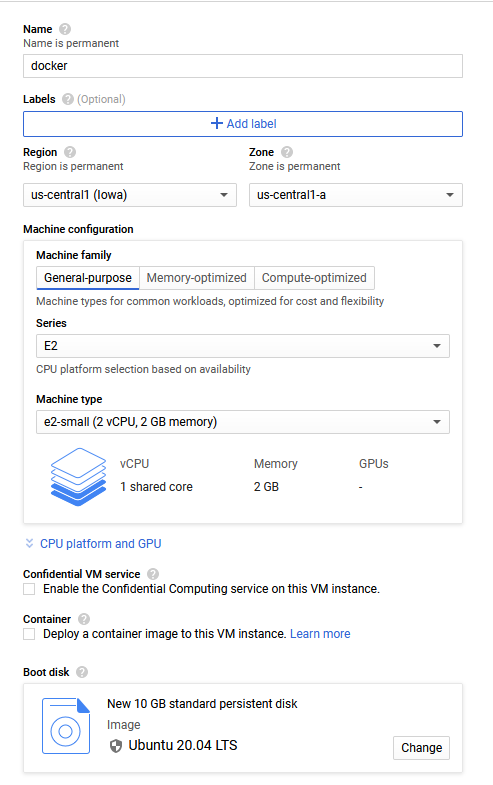
Hình ‑: Chọn VM instances

#### Bước 2: Chọn Create



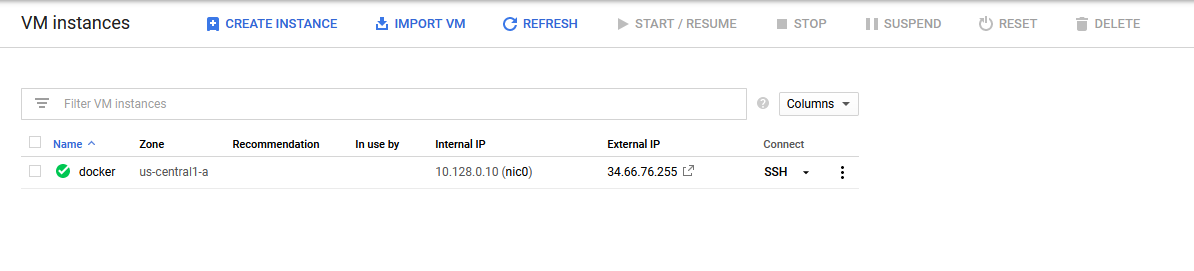
Hình ‑: Tạo VM instances

#### Bước 3: Điền các thông số -> Tạo máy ảo



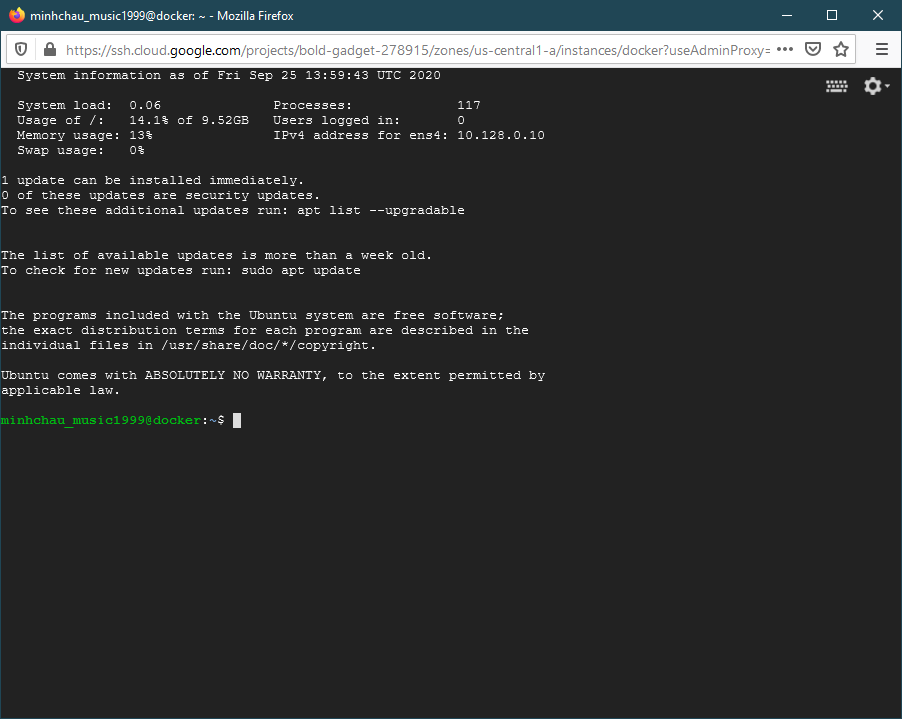
Hình ‑: Chọn cấu hình VM

#### Bước 4: Máy ảo Ubuntu 20.04 LTS đã tạo



Hình ‑: VM đã được tạo

#### Bước 5: Truy cập vào máy Ubuntu



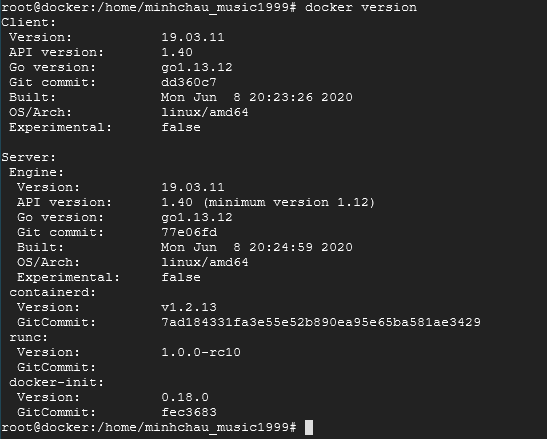
Hình ‑: Đăng nhập vào VM

### Cài đặt Docker:



Hình ‑: Cài đặt docker

### Kiểm tra Docker:



Hình ‑: Kiểm tra cài đặt

## Các lệnh cơ bản trong Docker

### Tải image về server:

docker pull <name\_image:tag> [OPTIONS]

**Cú pháp:**

(phần :tag là options, nếu để trống thì mặc định download bản latest )

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--all-tags , -a Tải xuống tất cả các hình ảnh được gắn thẻ trong kho lưu trữ

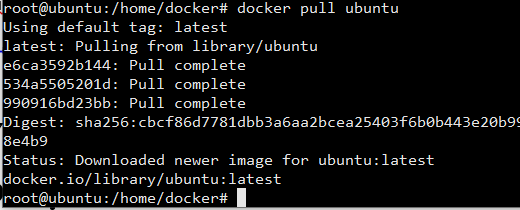
--disable-content-trust Bỏ qua xác minh hình ảnh

--platform Đặt nền tảng nếu máy chủ có khả năng đa nền tảng

--quiet , -q Ngăn chặn đầu ra dài dòng

Ví dụ: docker pull ubuntu => download ubuntu latest

docker pull ubuntu:20.04 => download ubuntu version 20.04



Hình ‑: Tải image ubuntu

### Hiển thị danh sách các images :

docker image [OPTIONS]

**Cú pháp:**

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--all , -a Hiển thị tất cả hình ảnh (mặc định ẩn hình ảnh trung gian)

--digests Hiển thị thông báo

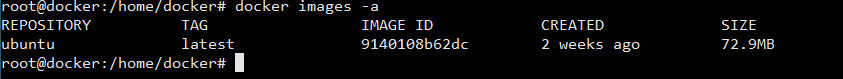
--filter , -f Lọc đầu ra dựa trên các điều kiện được cung cấp

--format Hình ảnh in đẹp bằng cách sử dụng mẫu

--no-trunc Đừng cắt ngắn đầu ra

--quiet , -q Chỉ hiển thị ID dạng số

Ví dụ:



Hình ‑: Hiển thị tất cả image

### Xóa một images:

**Cú pháp:**

docker image rm [OPTIONS] IMAGE [IMAGE...]

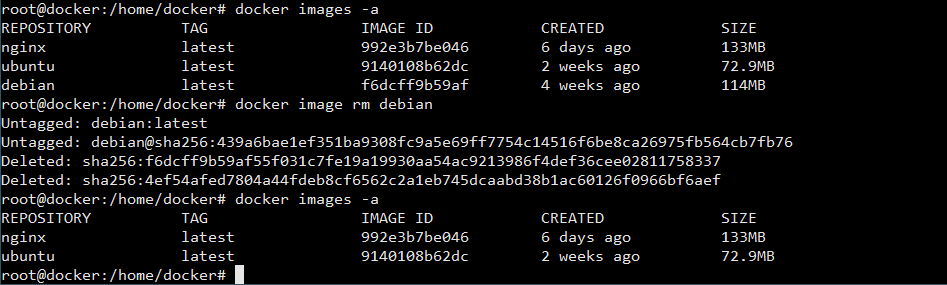
**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--force , -f Buộc xóa hình ảnh

--no-prune Không xóa cha mẹ không được gắn thẻ

Ví dụ:



Hình ‑:Xóa image debian

### Chạy một image:

**Cú pháp:**

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--add-host Thêm ánh xạ từ máy chủ đến IP tùy chỉnh (máy chủ: ip)

--attach , -a Đính kèm vào STDIN, STDOUT hoặc STDERR

--blkio-weight Chặn IO (trọng lượng tương đối), từ 10 đến 1000 hoặc 0 để tắt (mặc định là 0)

--blkio-weight-device Khối lượng IO (khối lượng thiết bị tương đối)

--cap-add Thêm các tính năng của Linux

--cap-drop Bỏ qua các khả năng của Linux

--cgroup-parent Nhóm mẹ tùy chọn cho vùng chứa

--cidfile Ghi ID vùng chứa vào tệp

--cpu-count Số lượng CPU (chỉ dành cho Windows)

--cpu-percent Phần trăm CPU (chỉ dành cho Windows)

--cpu-period Giới hạn thời gian CFS của CPU (Bộ lập lịch hoàn toàn công bằng)

--cpu-quota Giới hạn hạn ngạch CPU CFS (Trình lập lịch hoàn toàn công bằng)

--cpu-rt-period Giới hạn khoảng thời gian thực của CPU tính bằng micro giây

--cpu-rt-runtime Giới hạn thời gian chạy thời gian thực của CPU trong micro giây

--cpu-shares , -c Chia sẻ CPU (trọng lượng tương đối)

--cpus Số lượng CPU

--cpuset-cpus CPU cho phép thực thi (0-3, 0,1)

--cpuset-mems MEM cho phép thực thi (0-3, 0,1)

--detach , -d Chạy vùng chứa trong nền và in ID vùng chứa

--detach-keys Ghi đè chuỗi khóa để tách vùng chứa

--device Thêm thiết bị chủ vào vùng chứa

--device-cgroup-rule Thêm quy tắc vào danh sách thiết bị được phép nhóm

--device-read-bps Giới hạn tốc độ đọc (byte mỗi giây) từ một thiết bị

--device-read-iops Giới hạn tốc độ đọc (IO trên giây) từ một thiết bị

--device-write-bps Giới hạn tốc độ ghi (byte mỗi giây) cho một thiết bị

--device-write-iops Giới hạn tốc độ ghi (IO trên giây) cho một thiết bị

--disable-content-trust Bỏ qua xác minh hình ảnh

--dns Đặt máy chủ DNS tùy chỉnh

--dns-opt Đặt tùy chọn DNS

--dns-option Đặt tùy chọn DNS

--dns-search Đặt miền tìm kiếm DNS tùy chỉnh

--domainname Tên miền NIS vùng chứa

--entrypoint Ghi đè ENTRYPOINT mặc định của hình ảnh

--env , -e Đặt các biến môi trường

--env-file Đọc trong tệp các biến môi trường

--expose Để lộ một cổng hoặc một loạt các cổng

--gpus Các thiết bị GPU để thêm vào vùng chứa ('tất cả' để vượt qua tất cả các GPU)

--group-add Thêm nhóm bổ sung để tham gia

--health-cmd Lệnh chạy để kiểm tra sức khỏe

--health-interval Thời gian từ khi chạy kiểm tra (ms | s | m | h) (0s mặc định)

--health-retries Các lỗi liên tiếp cần báo cáo không lành mạnh

--health-start-period Khoảng thời gian bắt đầu để vùng chứa khởi tạo trước khi bắt đầu đếm ngược kiểm tra lại tình trạng (ms | s | m | h) (0s mặc định)

--health-timeout Thời gian tối đa để cho phép một lần kiểm tra chạy (ms | s | m | h) (0s mặc định)

--help Sử dụng in

--hostname , -h Tên máy chủ vùng chứa

--init Chạy một init bên trong vùng chứa để chuyển tiếp tín hiệu và thu thập các quy trình

--interactive , -i Giữ STDIN mở ngay cả khi không được đính kèm

--io-maxbandwidth Giới hạn băng thông IO tối đa cho ổ đĩa hệ thống (chỉ dành cho Windows)

--io-maxiops Giới hạn IOps tối đa cho ổ đĩa hệ thống (chỉ dành cho Windows)

--ip Địa chỉ IPv4 (ví dụ: 172.30.100.104)

--ip6 Địa chỉ IPv6 (ví dụ: 2001: db8 :: 33)

--ipc Chế độ IPC để sử dụng

--isolation Công nghệ cách ly container

--kernel-memory Giới hạn bộ nhớ nhân

--label , -l Đặt dữ liệu meta trên vùng chứa

--label-file Đọc trong tệp nhãn được phân cách bằng dòng

--link Thêm liên kết vào một vùng chứa khác

--link-local-ip Vùng chứa địa chỉ liên kết IPv4 / IPv6 cục bộ

--log-driver Trình điều khiển ghi nhật ký cho vùng chứa

--log-opt Ghi nhật ký các tùy chọn trình điều khiển

--mac-address Địa chỉ MAC của vùng chứa (ví dụ: 92: d0: c6: 0a: 29: 33)

--memory , -m Giới hạn bộ nhớ

--memory-reservation Giới hạn mềm của bộ nhớ

--memory-swap Giới hạn hoán đổi bằng bộ nhớ cộng với hoán đổi: '-1' để cho phép hoán đổi không giới hạn

--memory-swappiness Điều chỉnh sự thay đổi bộ nhớ vùng chứa (0 đến 100)

--mount Đính kèm liên kết hệ thống tệp vào vùng chứa

--name Gán tên cho vùng chứa

--net Kết nối vùng chứa với mạng

--net-alias Thêm bí danh phạm vi mạng cho vùng chứa

--network Kết nối vùng chứa với mạng

--network-alias Thêm bí danh phạm vi mạng cho vùng chứa

--no-healthcheck Tắt mọi HEALTHCHECK do vùng chứa chỉ định

--oom-kill-disable Tắt OOM Killer

--oom-score-adj Điều chỉnh tùy chọn OOM của máy chủ lưu trữ (-1000 đến 1000)

--pid Không gian tên PID để sử dụng

--pids-limit Điều chỉnh giới hạn pids vùng chứa (đặt -1 cho không giới hạn)

--platform Đặt nền tảng nếu máy chủ có khả năng đa nền tảng

--privileged Cấp các đặc quyền mở rộng cho vùng chứa này

--publish , -p Xuất bản (các) cổng của vùng chứa lên máy chủ

--publish-all , -P Xuất bản tất cả các cổng tiếp xúc với các cổng ngẫu nhiên

--read-only Gắn hệ thống tệp gốc của vùng chứa dưới dạng chỉ đọc

--restart no Chính sách khởi động lại áp dụng khi vùng chứa thoát ra

--rm Tự động loại bỏ vùng chứa khi nó thoát ra

--runtime Thời gian chạy để sử dụng cho vùng chứa này

--security-opt Tùy chọn bảo mật

--shm-size Kích thước của / dev / shm

--sig-proxy true Proxy đã nhận tín hiệu cho quá trình

--stop-signal SIGTERM Báo hiệu dừng container

--stop-timeout Thời gian chờ (tính bằng giây) để dừng một vùng chứa

--storage-opt Tùy chọn trình điều khiển lưu trữ cho vùng chứa

--sysctl Tùy chọn Sysctl

--tmpfs Gắn một thư mục tmpfs

--tty , -t Phân bổ TTY giả

--ulimit Bỏ qua các tùy chọn

--user , -u Tên người dùng hoặc UID (định dạng: <name | uid> [: <group | gid>])

--userns Không gian tên người dùng để sử dụng

--uts Không gian tên UTS để sử dụng

--volume , -v Ràng buộc một tập

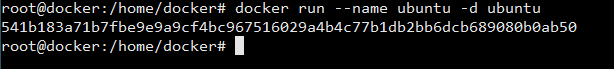
--volume-driver Trình điều khiển âm lượng tùy chọn cho vùng chứa

--volumes-from Gắn khối lượng từ (các) vùng chứa được chỉ định

--workdir , -w Thư mục làm việc bên trong vùng chứa

Ví dụ:

* Chạy một images ubuntu với tên ubuntu và chạy dưới nền:



Hình ‑: Chạy một images ubuntu với tên ubuntu và chạy dưới nền

* Chạy một images ubuntu với tên ubuntu-1 và truy cập ngay:



Hình ‑:Chạy một images ubuntu với tên ubuntu-1 và truy cập ngay

* Chạy một images ubuntu với tên ubuntu-2, tên vm test và truy cập ngay:



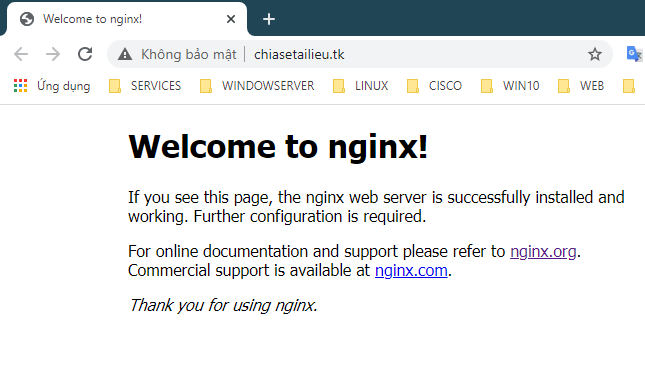
Hình ‑:Chạy một images ubuntu với tên ubuntu-2, tên vm test và truy cập ngay

* Chạy một images nginx với tên docker-nginx , port 80 và chạy dưới nền:



Hình ‑:Chạy một images nginx với tên docker-nginx , port 80 và chạy dưới nền

Kiểm tra:



Hình ‑:Kiểm tra image đã chạy

### ****Liệt kê các container:****

**Cú pháp:**

docker ps [OPTIONS]

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--all , -a Hiển thị tất cả các vùng chứa (các chương trình mặc định chỉ đang chạy)

--filter , -f Lọc đầu ra dựa trên các điều kiện được cung cấp

--format Hộp đựng in đẹp bằng cách sử dụng mẫu Go

--last , -n -1 Hiển thị n vùng chứa được tạo lần cuối (bao gồm tất cả các trạng thái)

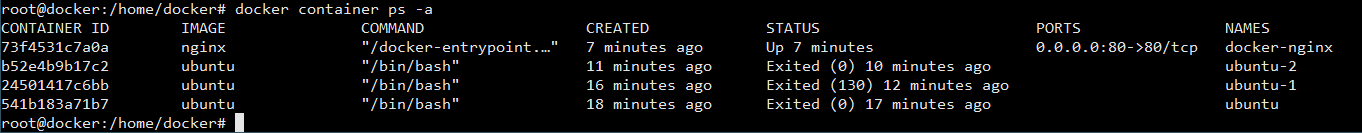
--latest , -l Hiển thị vùng chứa được tạo mới nhất (bao gồm tất cả các trạng thái)

--no-trunc Đừng cắt ngắn đầu ra

--quiet , -q Chỉ hiển thị ID số

--size , -s Hiển thị tổng kích thước tệp

Ví dụ:



Hình ‑:Liệt kê tất cả các container

### Dừng container đang chạy:

**Cú pháp:**

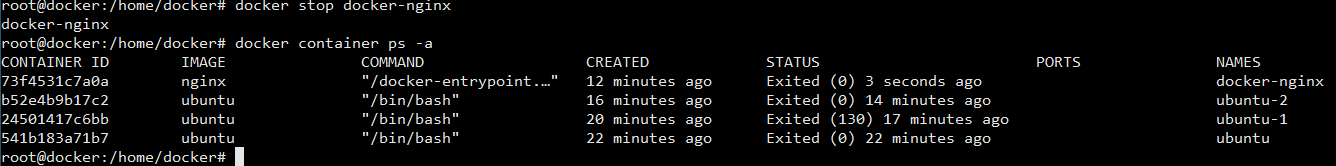
docker stop [OPTIONS] CONTAINER [CONTAINER...]

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

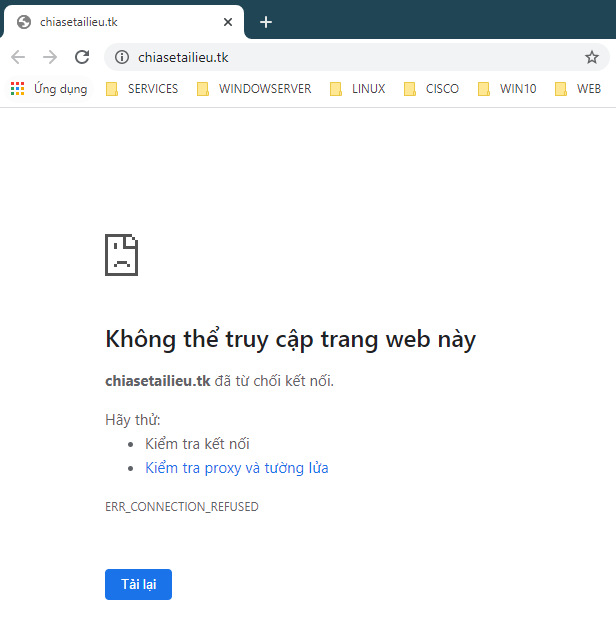
--time , -t 10 Vài giây để chờ dừng trước khi giết nó

Ví dụ:



Hình ‑:Dừng container docker-nginx

Kiểm tra:



Hình ‑:Kiểm tra lại image đã dừng

### Khởi động lại container đã dừng:

**Cú pháp:**

docker start [OPTIONS] CONTAINER [CONTAINER...]

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--attach , -a Đính kèm các tín hiệu STDOUT / STDERR và chuyển tiếp

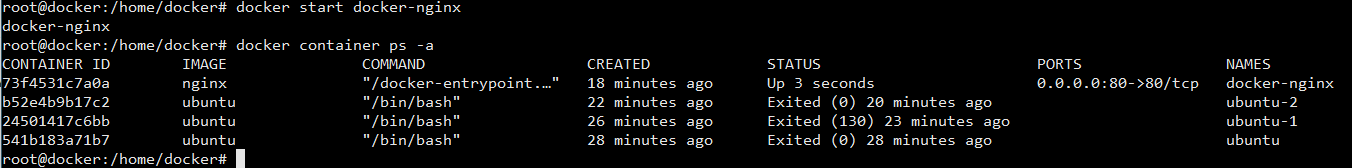
--checkpoint Khôi phục từ trạm kiểm soát này

--checkpoint-dir Sử dụng thư mục lưu trữ trạm kiểm soát tùy chỉnh

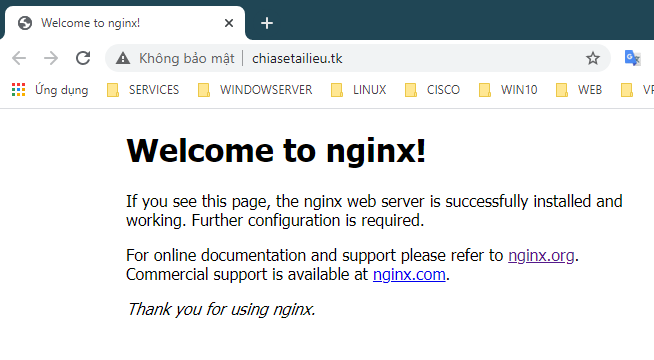
--detach-keys Ghi đè chuỗi khóa để tách vùng chứa

--interactive , -i Đính kèm STDIN của vùng chứa

Ví dụ:



Hình ‑:Khởi động lại container và kiểm tra



Hình ‑:Kiểm tra container docker-nginx

### **Truy cập vào một container đang chạy:**

**Cú pháp:**

docker attach [OPTIONS] CONTAINER

**Tùy chọn**

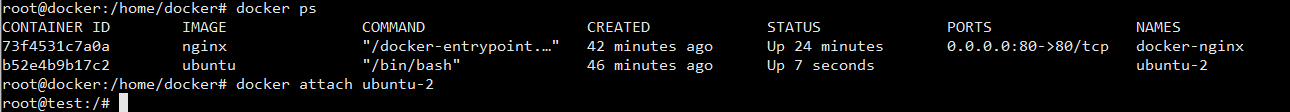
**Tên, viết tắt Mặc định Sự miêu tả**

--detach-keys Ghi đè chuỗi khóa để tách vùng chứa

--no-stdin Không đính kèm STDIN

--sig-proxy true Proxy tất cả các tín hiệu nhận được cho quá trình

Ví dụ:



Hình ‑:Truy cập vào container ubuntu-2

### Xóa container không còn sử dụng:

**Cú pháp:**

docker rm [OPTIONS] CONTAINER [CONTAINER...]

**Tùy chọn**

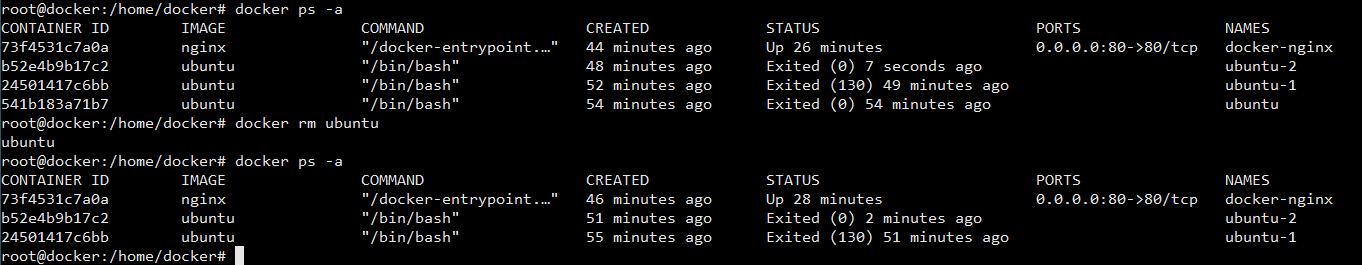
**Tên, viết tắt Mặc định Sự miêu tả**

--force , -f Buộc xóa vùng chứa đang chạy (sử dụng SIGKILL)

--link , -l Xóa liên kết được chỉ định

--volumes , -v Xóa các tập ẩn danh được liên kết với vùng chứa

Ví dụ:



Hình ‑:Xóa container ubuntu và kiểm tra

### **Lưu một images :**

**Cú pháp:**

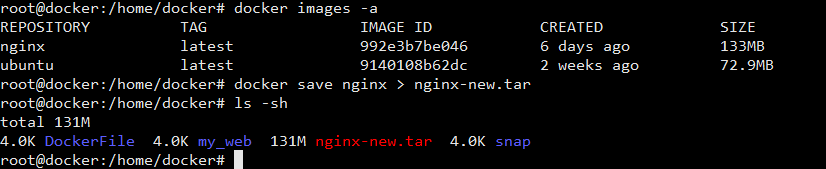
docker save [OPTIONS] IMAGE [IMAGE...]

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--output , -o Ghi vào tệp, thay vì STDOUT

Ví dụ:



Hình ‑: Lưu image nginx

### **Tải images từ file đã lưu:**

**Cú pháp**:

docker export [OPTIONS] CONTAINER

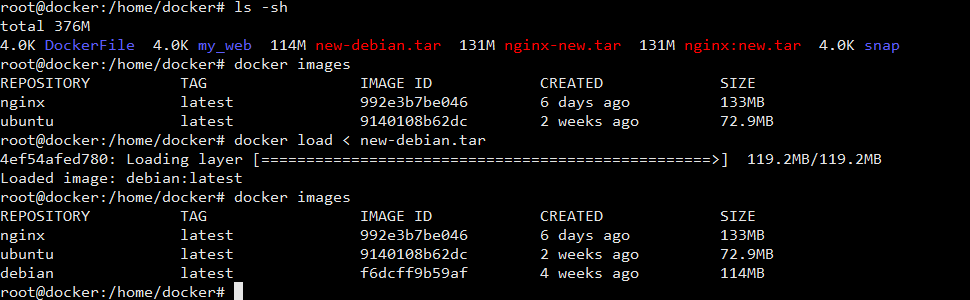
**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--input , -i Đọc từ tệp lưu trữ tar, thay vì STDIN

--quiet , -q Ngắt đầu ra tải

Ví dụ:



Hình ‑:Tải image nginx từ file

### **Export container :**

**Cú pháp:**

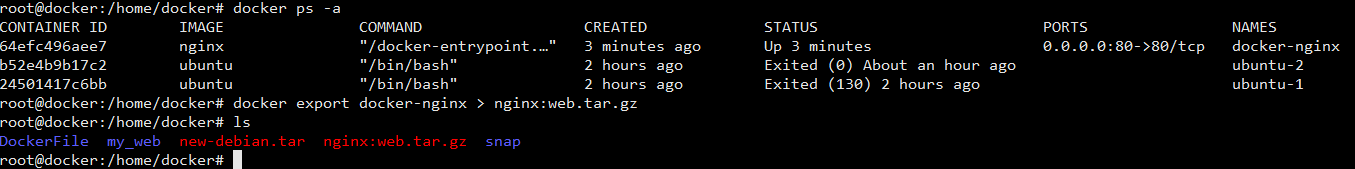
docker export [OPTIONS] CONTAINER

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--output , -o Ghi vào tệp, thay vì STDOUT

Ví dụ:



Hình ‑:Xuất container thành file nginx:web.tar.gz

### **Import container :**

**Cú pháp:**

docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]

**Tùy chọn**

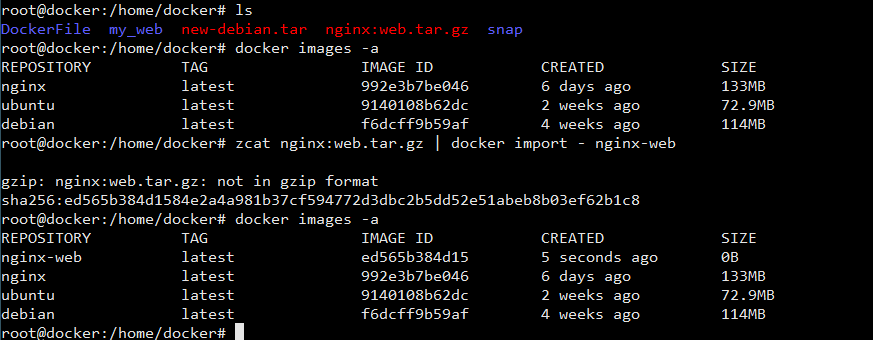
**Tên, viết tắt Mặc định Sự miêu tả**

--change , -c Áp dụng hướng dẫn Dockerfile cho hình ảnh đã tạo

--message , -m Đặt thông báo cam kết cho hình ảnh đã nhập

--platform Đặt nền tảng nếu máy chủ có khả năng đa nền tảng

Ví dụ:



Hình ‑:Khôi phục lại image

### Tạo thêm một số thuộc tính cho Docker:

**Cú pháp:**

docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

**Tùy chọn**

**Tên, viết tắt Mặc định Sự miêu tả**

--add-host Thêm ánh xạ từ máy chủ đến IP tùy chỉnh (máy chủ: ip)

--attach , -a Đính kèm vào STDIN, STDOUT hoặc STDERR

--blkio-weight Chặn IO (trọng lượng tương đối), từ 10 đến 1000 hoặc 0 để tắt (mặc định là 0)

--blkio-weight-device Khối lượng IO (khối lượng thiết bị tương đối)

--cap-add Thêm các tính năng của Linux

--cap-drop Bỏ qua các khả năng của Linux

--cgroup-parent Nhóm mẹ tùy chọn cho vùng chứa

--cidfile Ghi ID vùng chứa vào tệp

--cpu-count Số lượng CPU (chỉ dành cho Windows)

--cpu-percent Phần trăm CPU (chỉ dành cho Windows)

--cpu-period Giới hạn thời gian CFS của CPU (Bộ lập lịch hoàn toàn công bằng)

--cpu-quota Giới hạn hạn ngạch CPU CFS (Trình lập lịch hoàn toàn công bằng)

--cpu-rt-period Giới hạn khoảng thời gian thực của CPU tính bằng micro giây

--cpu-rt-runtime Giới hạn thời gian chạy thời gian thực của CPU trong micro giây

--cpu-shares , -c Chia sẻ CPU (trọng lượng tương đối)

--cpus Số lượng CPU

--cpuset-cpus CPU cho phép thực thi (0-3, 0,1)

--cpuset-mems MEM cho phép thực thi (0-3, 0,1)

--device Thêm thiết bị chủ vào vùng chứa

--device-cgroup-rule Thêm quy tắc vào danh sách thiết bị được phép nhóm

--device-read-bps Giới hạn tốc độ đọc (byte mỗi giây) từ một thiết bị

--device-read-iops Giới hạn tốc độ đọc (IO trên giây) từ một thiết bị

--device-write-bps Giới hạn tốc độ ghi (byte mỗi giây) cho một thiết bị

--device-write-iops Giới hạn tốc độ ghi (IO trên giây) cho một thiết bị

--disable-content-trust Bỏ qua xác minh hình ảnh

--dns Đặt máy chủ DNS tùy chỉnh

--dns-opt Đặt tùy chọn DNS

--dns-option Đặt tùy chọn DNS

--dns-search Đặt miền tìm kiếm DNS tùy chỉnh

--domainname Tên miền NIS vùng chứa

--entrypoint Ghi đè ENTRYPOINT mặc định của hình ảnh

--env , -e Đặt các biến môi trường

--env-file Đọc trong tệp các biến môi trường

--expose Để lộ một cổng hoặc một loạt các cổng

--gpus Các thiết bị GPU để thêm vào vùng chứa ('tất cả' để vượt qua tất cả các GPU)

--group-add Thêm nhóm bổ sung để tham gia

--health-cmd Lệnh chạy để kiểm tra sức khỏe

--health-interval Thời gian từ khi chạy kiểm tra (ms | s | m | h) (0s mặc định)

--health-retries Các lỗi liên tiếp cần báo cáo không lành mạnh

--health-start-period Khoảng thời gian bắt đầu để vùng chứa khởi tạo trước khi bắt đầu đếm ngược thử nghiệm lại tình trạng (ms | s | m | h) (0s mặc định)

--health-timeout Thời gian tối đa để cho phép một lần kiểm tra chạy (ms | s | m | h) (0s mặc định)

--help Sử dụng in

--hostname , -h Tên máy chủ vùng chứa

--init Chạy một init bên trong vùng chứa để chuyển tiếp tín hiệu và thu thập các quy trình

--interactive , -i Giữ STDIN mở ngay cả khi không được đính kèm

--io-maxbandwidth Giới hạn băng thông IO tối đa cho ổ đĩa hệ thống (chỉ dành cho Windows)

--io-maxiops Giới hạn IOps tối đa cho ổ đĩa hệ thống (chỉ dành cho Windows)

--ip Địa chỉ IPv4 (ví dụ: 172.30.100.104)

--ip6 Địa chỉ IPv6 (ví dụ: 2001: db8 :: 33)

--ipc Chế độ IPC để sử dụng

--isolation Công nghệ cách ly container

--kernel-memory Giới hạn bộ nhớ nhân

--label , -l Đặt dữ liệu meta trên vùng chứa

--label-file Đọc trong tệp nhãn được phân cách bằng dòng

--link Thêm liên kết vào một vùng chứa khác

--link-local-ip Vùng chứa địa chỉ liên kết IPv4 / IPv6 cục bộ

--log-driver Trình điều khiển ghi nhật ký cho container

--log-opt Ghi nhật ký các tùy chọn trình điều khiển

--mac-address Địa chỉ MAC của vùng chứa (ví dụ: 92: d0: c6: 0a: 29: 33)

--memory , -m Giới hạn bộ nhớ

--memory-reservation Giới hạn mềm của bộ nhớ

--memory-swap Giới hạn hoán đổi bằng bộ nhớ cộng với hoán đổi: '-1' để cho phép hoán đổi không giới hạn

--memory-swappiness Điều chỉnh sự thay đổi bộ nhớ vùng chứa (0 đến 100)

--mount Đính kèm liên kết hệ thống tệp vào vùng chứa

--name Gán tên cho vùng chứa

--net Kết nối vùng chứa với mạng

--net-alias Thêm bí danh phạm vi mạng cho vùng chứa

--network Kết nối vùng chứa với mạng

--network-alias Thêm bí danh phạm vi mạng cho vùng chứa

--no-healthcheck Tắt mọi HEALTHCHECK do vùng chứa chỉ định

--oom-kill-disable Tắt OOM Killer

--oom-score-adj Điều chỉnh tùy chọn OOM của máy chủ lưu trữ (-1000 đến 1000)

--pid Không gian tên PID để sử dụng

--pids-limit Điều chỉnh giới hạn pids vùng chứa (đặt -1 cho không giới hạn)

--platform Đặt nền tảng nếu máy chủ có khả năng đa nền tảng

--privileged Cấp các đặc quyền mở rộng cho vùng chứa này

--publish , -p Xuất bản (các) cổng của vùng chứa lên máy chủ

--publish-all , -P Xuất bản tất cả các cổng tiếp xúc với các cổng ngẫu nhiên

--read-only Gắn hệ thống tệp gốc của vùng chứa dưới dạng chỉ đọc

--restart no Chính sách khởi động lại để áp dụng khi vùng chứa thoát ra

--rm Tự động loại bỏ vùng chứa khi nó thoát

--runtime Thời gian chạy để sử dụng cho vùng chứa này

--security-opt Tùy chọn bảo mật

--shm-size Kích thước của / dev / shm

--stop-signal SIGTERM Báo hiệu dừng container

--stop-timeout Thời gian chờ (tính bằng giây) để dừng một vùng chứa

--storage-opt Tùy chọn trình điều khiển lưu trữ cho vùng chứa

--sysctl Tùy chọn Sysctl

--tmpfs Gắn một thư mục tmpfs

--tty , -t Phân bổ TTY giả

--ulimit Bỏ qua các tùy chọn

--user , -u Tên người dùng hoặc UID (định dạng: <name | uid> [: <group | gid>])

--userns Không gian tên người dùng để sử dụng

--uts Không gian tên UTS để sử dụng

--volume , -v Ràng buộc một tập

--volume-driver Trình điều khiển âm lượng tùy chọn cho vùng chứa

--volumes-from Gắn khối lượng từ (các) vùng chứa được chỉ định

--workdir , -w Thư mục làm việc bên trong vùng chứa

## Dockerfile:

### Quy tắc viết chỉ thị Dockerfile:

Chú ý: Khi viết các chỉ thị trong Dockerfile thì một chỉ thị có thể viết theo cấu trúc:

# Ghi chú chỉ thị

TÊN\_CHỈ\_THỊ các\_tham\_số

### Các chỉ thị Dockerfile:

**Phần này tìm hiểu các chỉ thị cơ bản:**

* FROM : mọi Docker file đều có chỉ thị này, chỉ định image cơ sở
* COPY ADD : sao chép dữ liệu
* ENV : thiết lập biến môi trường
* RUN : chạy các lệnh.
* VOLUME : gắn ổ đĩa, thư mục
* USER : người dùng
* WORKDIR : thư mục làm việc
* EXPOSE : thiết lập cổng

**FROM Trong Dockerfile**

Như trên đã nói, chỉ thị này chỉ ra image cơ sở để xây dựng nên image mới. Để xây dựng từ image nào đó thì bạn cần đọc document của Image đó để biết trong đó đang chứa gì, có thể chạy các lệnh gì trong đó ... Ví dụ, nếu bạn chọn xây dựng từ image centos:laste thì bạn bắt đầu bằng hệ điều hành CentOS và bạn có thể cài đặt, cập nhật các gói với yum, ngược lại nếu bạn chọn ubuntu:latest thì trình quản lý gói của nó là APT ...

**COPY và ADD Trong Dockerfile**

Được dùng để thêm thư mục, file vào Image. Cú pháp viết đó là:

ADD thư\_mục\_nguồn thư\_mục\_đích

Trong đó thư\_mục\_nguồn là thư mục ở máy chạy Dockerfile, chứa dữ liệu cần thêm vào. thư\_mục\_đích là nơi dữ liệu được thêm vào ở container.

**ENV Trong Dockerfile**

Chỉ thị này dùng để thiết lập biến môi trường, như biến môi trường PATH ..., tùy hệ thống hay ứng dụng yêu cầu biến môi trường nào thì căn cứ vào đó để thiết lập.

ENV biến giá\_trị

**RUN Trong Dockerfile**

Thi hành các lệnh, tương tự bạn chạy lệnh shell trên OS từ terminal.

RUN lệnh-và-tham-số-cần-chạy

RUN ["lệnh", "tham số1", "tham số 2" ...]

**VOLUME Trong Dockerfile**

Chỉ thi tạo một ổ đĩa:

VOLUME /dir\_vol

Chia sẻ dữ liệu giữa các Container

**USER Trong Dockerfile**

Bạn thêm người dùng được dùng khi chạy các lệnh ở chỉ thị RUN CMD WORKDIR.

USER <user>[:<group>]

**WORKDIR Trong Dockerfile**

Thiết lập thư mục làm việc hiện tại chi các chỉ thị CMD, ENTRYPOINT, ADD thi hành.

WORKDIR path\_current\_dir

**EXPOSE Trong Dockerfile**

Thiết lập cổng mà container lắng nghe, cho phép các container khác trên cùng mạng liên lạc qua cổng này hoặc ánh xạ cổng host vào cổng này.

EXPOSE port

**ENTRYPOINT, CMD Trong Dockerfile**

Chạy lệnh trong chỉ thị này khi container được chạy.

ENTRYPOINT commnad\_script

ENTRYPOINT ["command", "tham-số", ...]

CMD ý nghĩa tương tự như ENTRYPOINT, khác là lệnh chạy bằng shell của container.

CMD command param1 param2

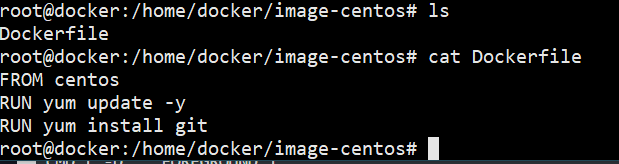
Chú ý: Ở dạng sau của CMD thì nó lại là thiết lập tham số cho ENTRYPOINT

CMD ["tham-số1", "tham-số2"]

Ví dụ: Tạo images centos được cài đặt “git”

Đầu tiên chúng ta cần tạo một file Dockerfile: -> nano Dockerfile

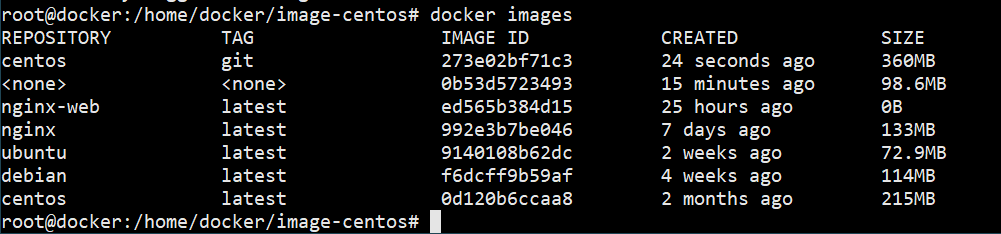
nội dung file như sau:



Hình ‑:Tạo images centos được cài đặt “git”

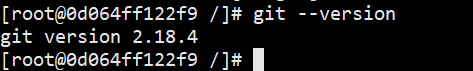
Để tạo images ta dùng lệnh: docker build -t centos:git .

Kết quả:



Hình ‑:Hiển thị image đã được tạo

Kiểm tra images có git không ta chạy images: docker run -it centos:git



Hình ‑:Kiểm tra "git" đã được cài đặt

## Docker-compose:

### Giới thiệu:

Sự phổ biến của Docker như một công cụ phát triển đang gia tăng. Docker đã thổi một luồng sinh khí mới vào phong trào container. Các nhà phát triển thích sử dụng nó vì nó nhanh và dễ học. Nó giúp các nhóm phát triển chia sẻ môi trường tiêu chuẩn mà không phải lo lắng về việc lãng phí thời gian và tài nguyên.

Các nhà phát triển có thể thiết lập môi trường mong muốn trong vùng chứa Docker, lưu vùng chứa dưới dạng hình ảnh và dễ dàng chia sẻ nó với các nhóm phát triển của họ. Quá trình hoạt động tốt cho một vùng chứa duy nhất. Tuy nhiên, môi trường nhiều container khó duy trì hơn. Docker Compose cung cấp giải pháp.

Với Docker Compose, các nhà phát triển có thể xác định một tệp YAML để thiết lập cấu hình cho nhiều dịch vụ. Sau đó, họ có thể bắt đầu các dịch vụ đa vùng chứa bằng một lệnh duy nhất. Nó đơn giản hóa quá trình làm việc với các ứng dụng nhiều vùng chứa.

### Sử dụng docker-compose:

Hiện tại, Docker chủ yếu được sử dụng trong môi trường phát triển. Một số cách sử dụng phổ biến của Docker Compose là:

#### Tạo mẫu và Phát triển:

Quá trình tạo mẫu và phát triển ứng dụng bị chậm lại do thiếu môi trường tiêu chuẩn. Các nhà phát triển thường phải mất thời gian thiết lập cùng một môi trường nhiều lần. Ngoài ra, việc đọc hướng dẫn để thiết lập các thông số môi trường cũng tốn nhiều thời gian.

Docker Compose đơn giản hóa quy trình. Sau khi một môi trường được định cấu hình, các nhóm phát triển có thể chia sẻ các tệp Docker trong toàn tổ chức. Nó có thể tiết kiệm rất nhiều thời gian lãng phí cho các vấn đề quản lý cấu hình.

#### Quy trình kiểm tra và tự động hóa:

Tích hợp liên tục và phân phối liên tục (CI / CD) đang trở thành quy trình tiêu chuẩn trong môi trường phát triển nhanh ngày nay. Kiểm tra tự động là một thành phần quan trọng của CI / CD. Docker Compose giúp xác định quy trình kiểm thử tự động. Tất cả những phức tạp của việc bắt đầu các dịch vụ mới có thể được đưa vào các tệp cấu hình docker. Người kiểm tra có thể sử dụng các tệp này để kích hoạt các dịch vụ tạm thời, chạy các tập lệnh văn bản và hủy các dịch vụ sau khi thu thập kết quả kiểm tra. Nó tiết kiệm thời gian vì khởi động các dịch vụ theo cách thủ công rất tốn thời gian và dễ xảy ra lỗi.

#### Triển khai sản xuất trong tương lai:

Docker chủ yếu được sử dụng trong môi trường phát triển. Tuy nhiên, khi các chức năng của Docker trở nên mạnh mẽ hơn, Docker sẽ được sử dụng cho nhiều công việc ở cấp độ sản xuất hơn. Docker Compose có thể là một công cụ có giá trị cho việc triển khai máy chủ duy nhất.

### Quy trình soạn thảo Docker:

#### Xác định môi trường ứng dụng:

Sử dụng Dockerfile để xác định môi trường ứng dụng để làm cho nó dễ dàng tái tạo.

Sử dụng Dockerfile để tự động tạo các image trong Docker

#### Xác định Môi trường Soạn thư Docker:

Sử dụng docker-compose.yml để xác định các dịch vụ trong ứng dụng.

Cấu trúc file docker-compose.yml viết theo cấu trúc của ngôn ngữ yaml và viết theo từng phiên bản của docker-compose

Cách viết file docker-compose:

* **version:** chỉ ra phiên bản docker-compose đang sử dụng.
* **services:** chỉ ra các dịch vụ ( container ) muốn cài đặt và chạy.
* **image:** chỉ ra image trong lúc chạy container.
* **build:** dùng để chạy container.
* **ports:** thiết lập port chạy máy localhost và trong container.
* **restart:** tự động chạy khi container bị tắt.
* **environment:** thiết lập biến môi trường
* **depends\_on:** chỉ ra sự phụ thuộc. Tức là services nào phải được cài đặt và chạy trước khi services được config tại đó mới được cài đặt.
* **volume:** dùng để mount hai thư mục trên host và container với nhau.
* **links:** cho phép đăng kí cái tên kí danh cho server khác có thể gọi. Chỉ cần sử dụng tên là có thể liên kết với các services mong muốn.
* **context:** xát định thư mục gốc, dựa vào thư mục này để khai báo tiếp đường dẫn Dockerfile

#### Chạy ứng dụng :

Sử dụng docker-compose để chạy ứng dụng nhiều vùng chứa.

**Các lệnh Docker-compose:**

**Cách viết:**

docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]

docker-compose -h | --help

**Tùy chọn:**

**Tên, viết tắt Mặc định Sự miêu tả**

-f, --file FILE Chỉ định một tệp soạn thay thế

(Mặc định: docker-compose.yml)

-p, --project-name NAME Chỉ định tên dự án thay thế

(Mặt định: tên thư mục)

--verbose Hiển thị thêm đầu ra

--log-level LEVEL Đặt mức nhật ký (GỢI Ý, THÔNG TIN, CẢNH BÁO, LỖI)

--no-ansi Không in các ký tự điều khiển ANSI

-v, --version In phiên bản và thoát

-H, --host HOST Ổ cắm Daemon để kết nối với

--tls Sử dụng TLS; ngụ ý bởi –tlsverify

--tlscacert CA\_PATH Chứng chỉ tin cậy chỉ do CA này ký

--tlscert CLIENT\_CERT\_PATH Đường dẫn đến tệp chứng chỉ TLS

--tlskey TLS\_KEY\_PATH Đường dẫn đến tệp khóa TLS

--tlsverify Sử dụng TLS và xác minh điều khiển từ xa

--skip-hostname-check Không kiểm tra tên máy chủ của daemon với

tên được chỉ định trong chứng chỉ khách hàng

--project-directory PATH Chỉ định một thư mục làm việc thay thế

(Mặc định: đường dẫn của tệp Soạn)

--compatibility Nếu được đặt, Soạn sẽ cố gắng chuyển đổi các khóa trong các tệp v3 tương đương không phải Swarm của chúng

--env-file PATH Chỉ định một tệp môi trường thay thế

**Commands:**

build Xây dựng hoặc xây dựng lại dịch vụ

config Xác thực và xem compose-file

create Tạo dịch vụ

down Dừng và xóa vùng chứa, mạng, hình ảnh và khối lượng

events Nhận các sự kiện thời gian thực từ vùng chứa

exec Thực thi một lệnh trong một vùng chứa đang chạy

help Nhận trợ giúp về một lệnh

images Liệt kê hình ảnh

kill Dừng containers

logs Xem đầu ra từ containers

pause Tạm dừng dịch vụ

port In cổng công cộng để gắn cổng

ps Danh sách containers

pull Dịch vụ kéo images

push Dịch vụ đẩy images

restart Khởi động lại dịch vụ

rm Xóa đã dừng containers

run Chạy lệnh một lần

scale Đặt số lượng vùng chứa cho một dịch vụ

start Bắt đầu dịch vụ

stop Dừng dịch vụ

top Hiển thị các quy trình đang chạy

unpause Hủy tạm dừng dịch vụ

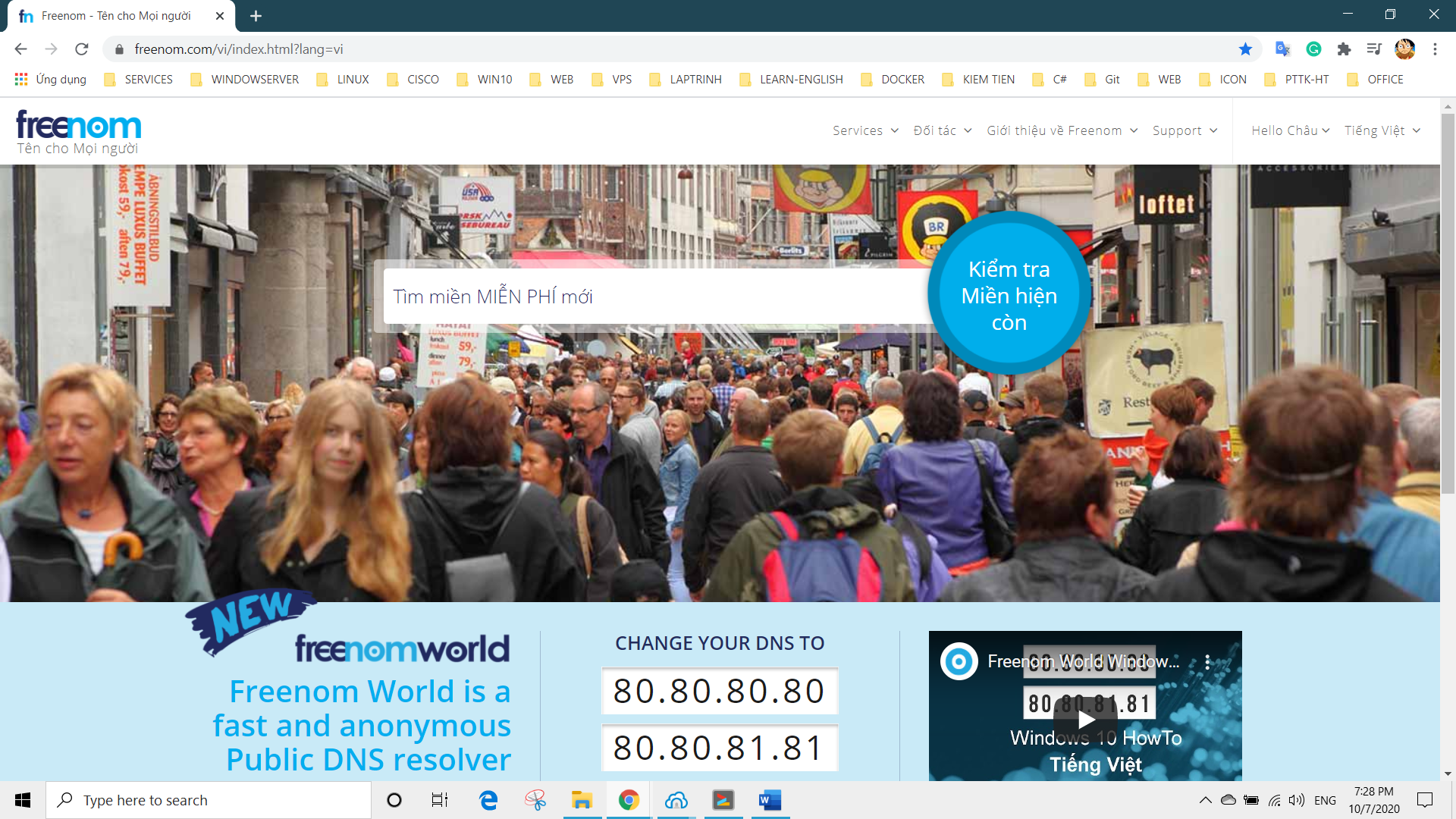
up Tạo và bắt đầu containers

version Hiển thị thông tin phiên bản docker-compose

# XÂY DỰNG MÁY CHỦ WEB

## Đăng kí tên miền:

**Bước 1: Truy cập vào trang web:** <https://www.freenom.com>



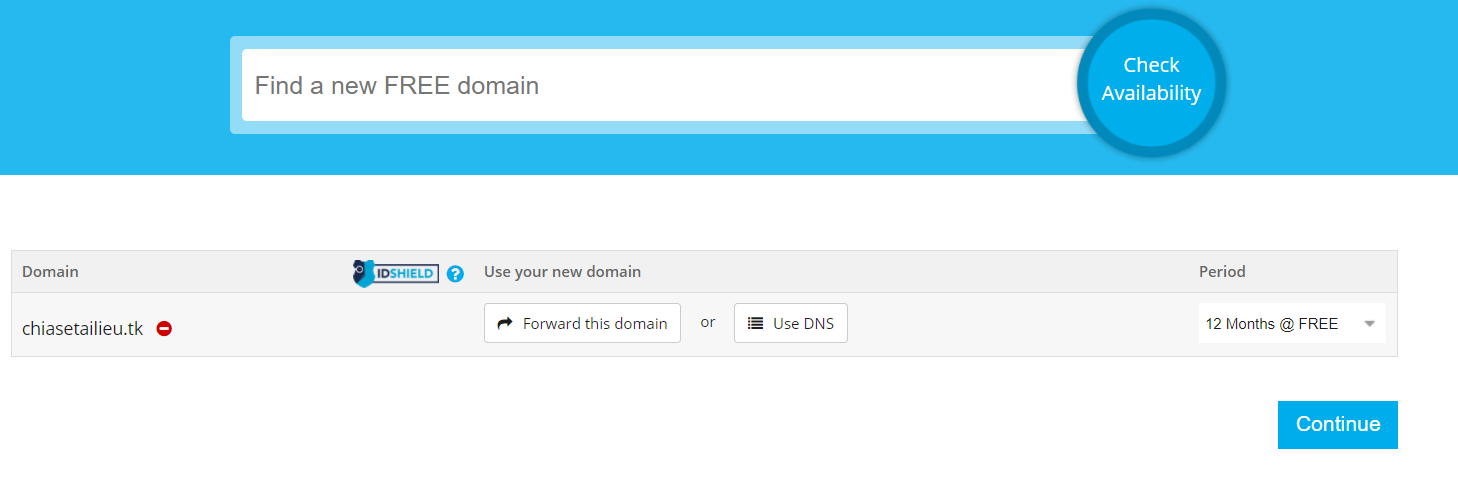
Hình ‑:Trang web freemon.com

**Bước 2: Nhập tên miền và kiểm tra**



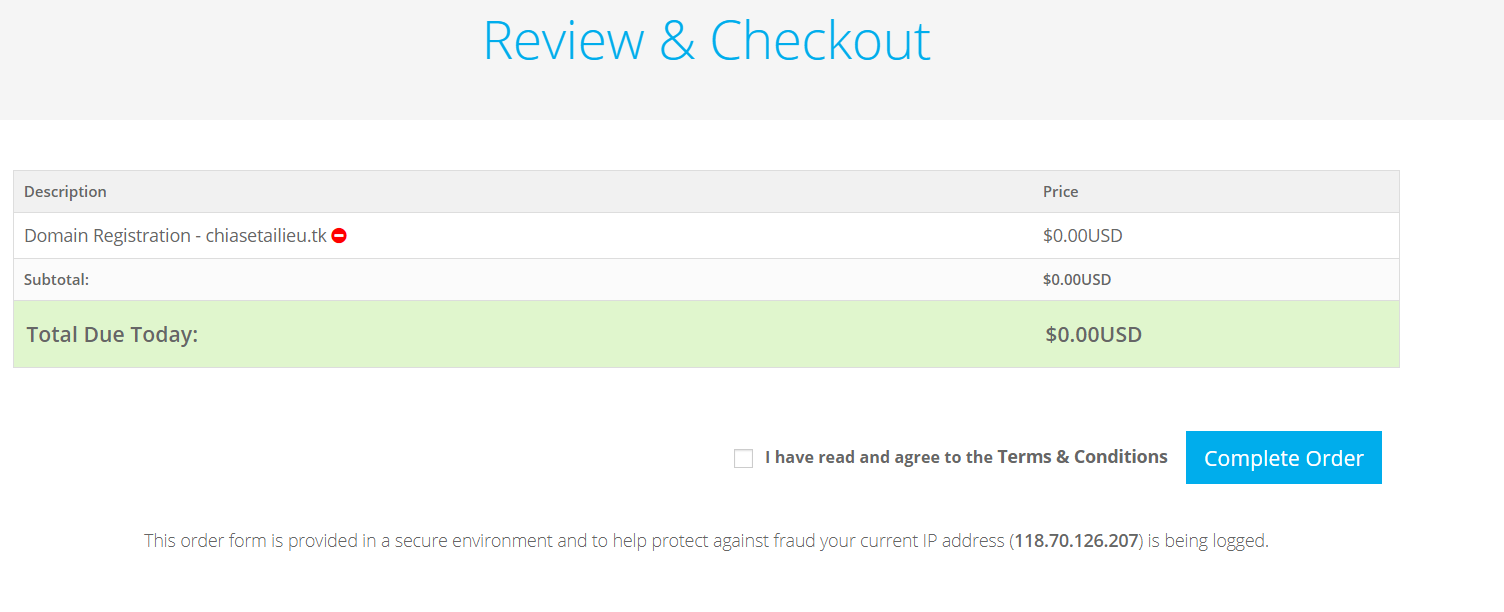
Hình ‑:Nhập tên miền muốn đăng kí

**Bước 3: Thêm vào giỏ hàng**



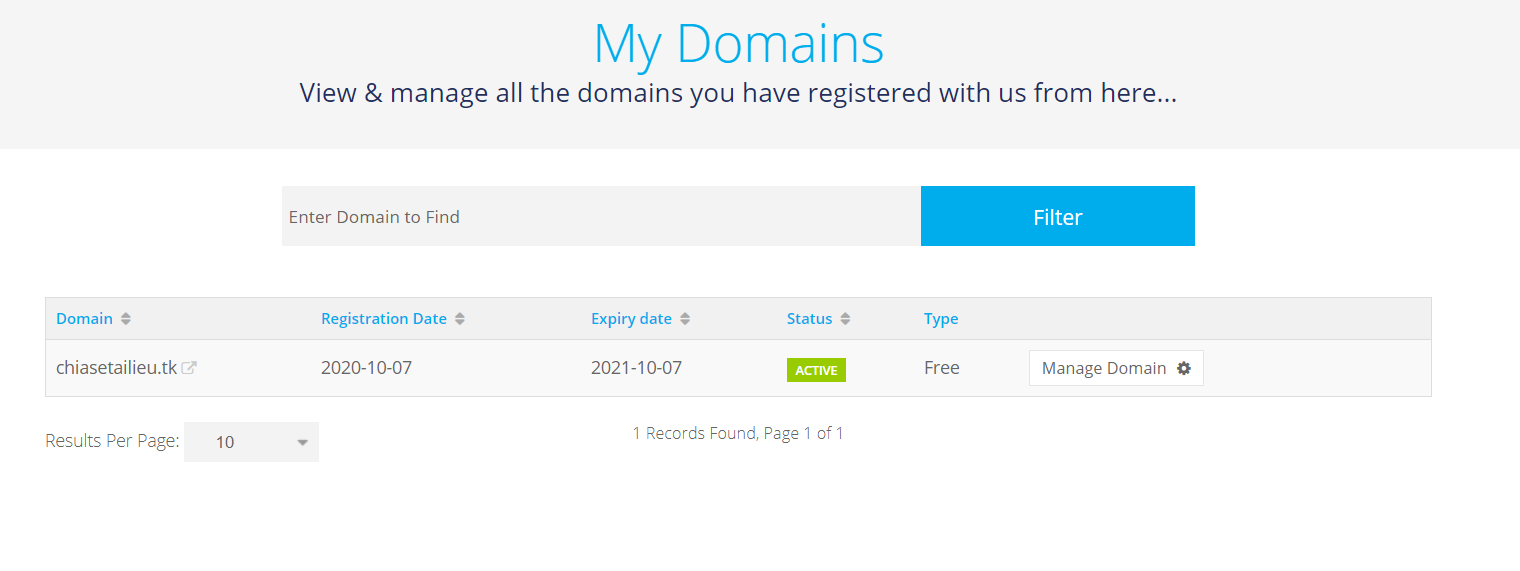
Hình ‑:Thêm tên miền vào giỏ hàng

**Bước 4: Thanh toán**



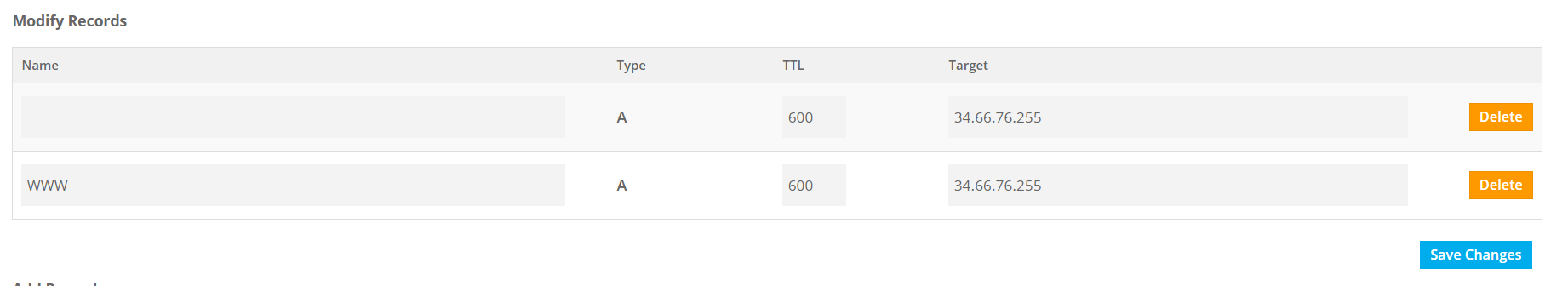
Hình ‑:Thanh toán mua tên miền

**Bước 5: Truy cập vào tên miền của mình**



Hình ‑:Tên miền đã được mua

**Bước 6: Thêm địa chỉ ip address ( host )**



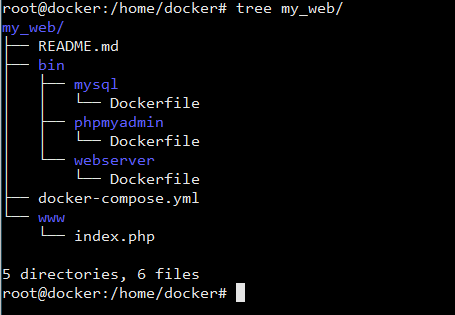
Hình ‑:Thêm địa chỉ ip của host vào tên miền

## Xây dựng máy chủ web:

**Yêu cầu các image:**

* mysql:8.0
* phpmyadmin/phpmyadmin:latest
* php:7.4-apache

**Bước 1: Tạo cấu trúc thư mục:**



Hình ‑:Cấu trúc thư mục để tạo server web

Trong đó:

* **Thư mục “my\_web”:** là thư mục chính.
* **README.md:** là file hướng dẫn cài đặt.
* **Thư mục “bin”:** chức các file Dockerfile.
* **Thư mục “www”:** chứa file web.
* **File “docker-compose.yml”:** file chạy để các image docker.

**Bước 2: Viết nội dung từng file:**

* /www/index.php



Hình ‑:Nội dung file index.php

* /bin/mysql/Dockerfile



Hình ‑:Dockerfile mysql

* /bin/phpmyadmin/Dockerfile



Hình ‑:Dockerfile phpmyadmin

* /bin/webserver/Dockerfile



Hình ‑:Dockerfile php:apache

* docker-compose.yml

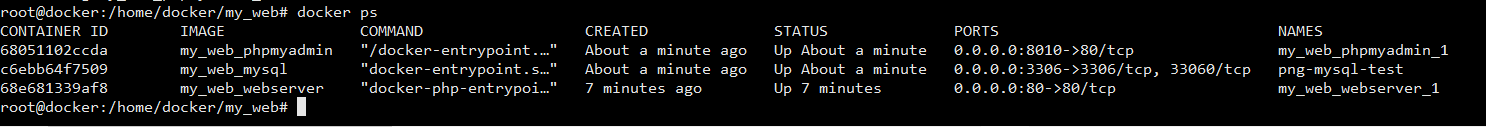


Hình ‑:File docker-compose

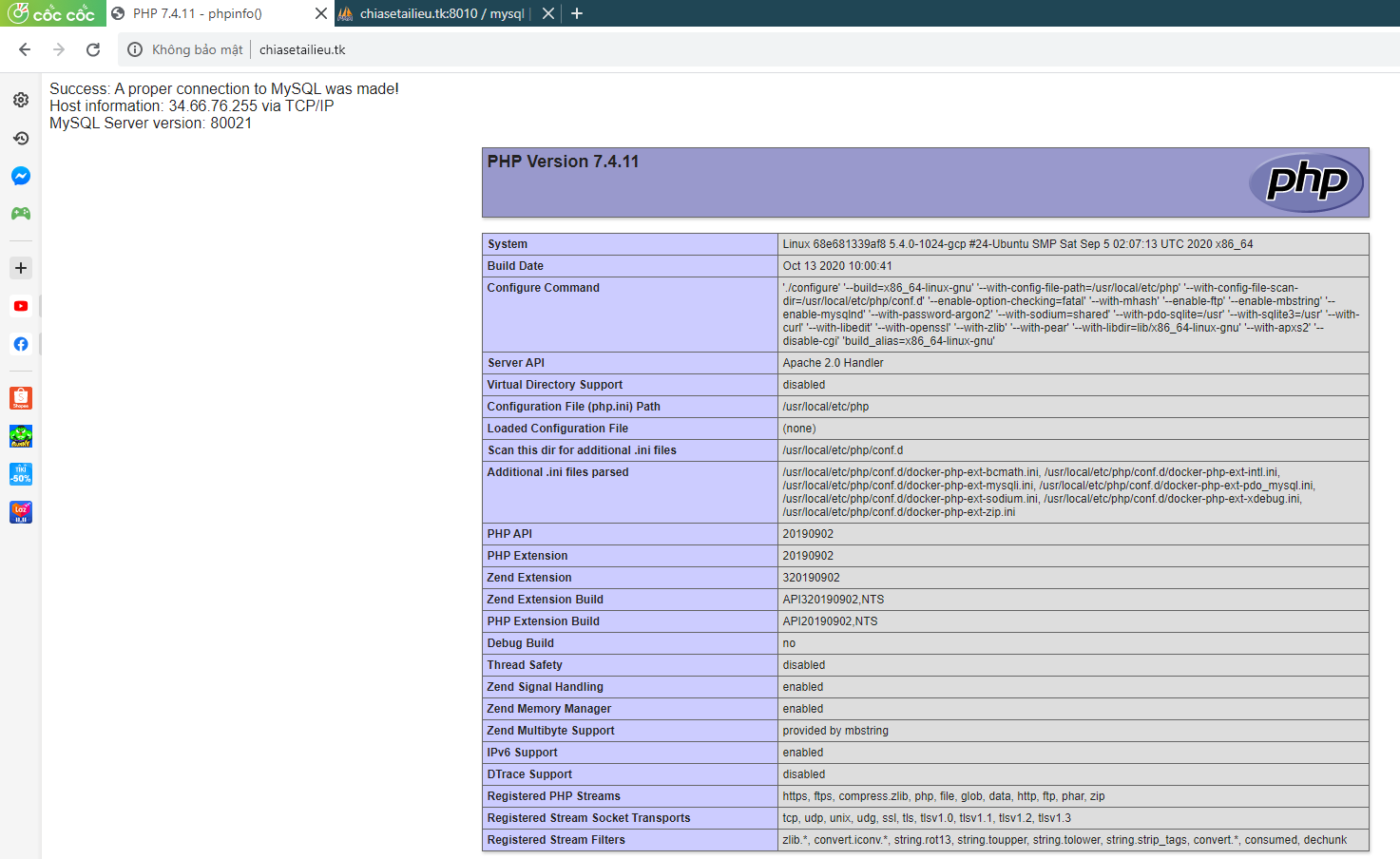
**Bước 3: Chạy các file đã viết:**

Dùng câu lệnh: “**docker-compose up -d**” để tạo các container

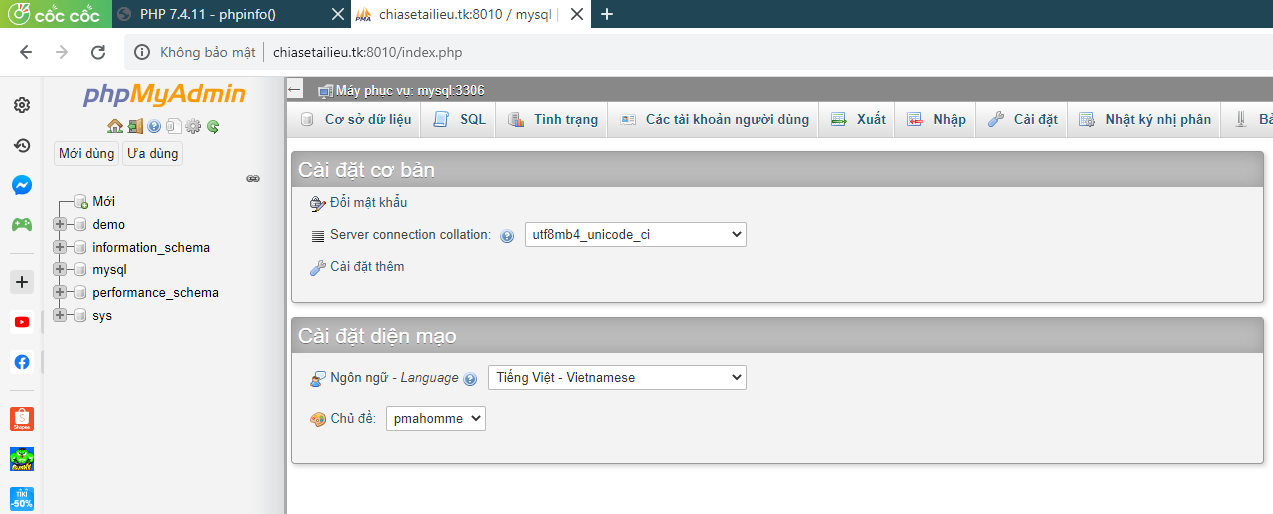
**Kiểm tra:**



Hình ‑:Có 3 container đang chạy



Hình ‑:Kết nối thành công với cơ sở dữ liệu mysql



Hình ‑:Đăng nhập thành công cơ sở dữ liệu mysql

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận:

### Ưu điểm đã làm được:

* Hiểu được ứng dụng của Docker
* Biết được các lệnh cơ bản trong Docker
* Tạo được một máy chủ chạy Web-server

### Hạn chế:

* Cần phải biết các lệnh về Docker.

## Hướng Phát triển:

Từ những ưu khuyết điểm trên ta đề ra những hướng phát triển trong tương lai:

* Có thể chạy được nhiều containner.
* Thích hợp cho xây đựng những app dự án nhiều thành viên.