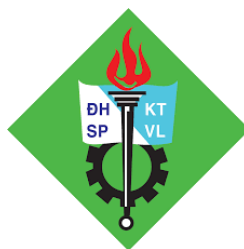


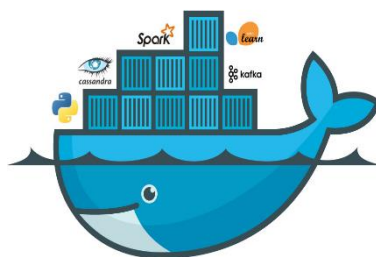
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT VĨNH LONG

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

Khóa : 43

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

Giáo Viên Hướng Dẫn: ThS.Trần Thu Mai

Vĩnh Long, tháng 12 năm 2020

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ 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: 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

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

KẾ HOẠCH THỰC HÀNH ĐỒ ÁN 2

HỌC KỲ 1 NĂM HỌC 2020-2021

Họ tên sinh viên thực hiện đồ án:

Nguyễn Minh Châu MSSV: 18004012

Lê Thị Thu Cẩm MSSV: 18004010

Lớp: ICTT18A1

Giảng viên hướng dẫn: Trần Thu Mai

Tuần	Nội dung thực hiện	Dự kiến kết quả đạt được	Ghi chú
37	Gặp giảng viên nhận đồ án		
38	Tìm hiểu tổng quan về Docker	Hiểu được Docker là gì?	
39	Châu: Cài đặt được Docker	Cài được Docker trên máy ảo Ubuntu 20.04	
	Cẩm: Cài đặt được Docker		
40	Châu: Tìm hiểu về Docker container	Hiểu được cách hoạt động cơ bản của Docker	Báo cáo qua mail
	Cẩm: Tìm hiểu về Docker create		
41	Châu: Tìm hiểu về Docker network	Cấu hình được network từ image Docker	
	Cẩm: Tìm hiểu về Docker image		
42	Châu: Tìm hiểu về Docker build	Hiểu được không gian lưu trữ của container Docker	
	Cẩm: Tìm hiểu về Docker volume		
43	Châu: Các lệnh cơ bản trong Docker	Hiểu được các lệnh trong Docker	
	Cẩm: Các lệnh cơ bản trong Docker		

44	Châu: Tìm hiểu về DockerFile	Biết được cách viết file DockerFile và Docker Compose	Báo cáo qua mail
	Cầm: Tìm hiểu về Docker Compose		
45	Châu: Viết file DockerFile	Tạo được file DockerFile và Docker Compose để tạo ra máy chủ Web	
	Cầm: Viết file Docker Compose		
46	Tạo ra file Docker image	Đóng gói DockerFile và Docker Compose thành image	
47	Chạy file Docker image	Hoàn thành cơ bản về máy chủ Web	Báo cáo qua mail
48	Kiểm tra bảo mật máy chủ Web	Đảm bảo an toàn bảo mật của máy chủ Web	
49	Tạo SSL cho máy chủ Web	Tạo kết nối an toàn cho máy khách	
50	Báo cáo đồ án lần 1		Báo cáo trực tiếp với GVHD
51	Báo cáo đồ án lần 2		Báo cáo trực tiếp với GVHD

Sinh viên lập kế hoạch

Nguyễn Minh Châu

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 Kỹ 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

MỤC LỤC

LỜI MỞ ĐẦU	1
CHƯƠNG 1 : TỔNG QUAN VỀ ĐỀ TÀI	2
1.1 Tên đề tài:.....	2
1.2 Lý do chọn đề tài:.....	2
1.3 Mục tiêu cần đạt:	2
CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT	3
2.1 Tổng quan về Docker.....	3
2.1.1 Tổng quan:.....	3
2.1.2 Nền tảng:	3
2.1.3 Công cụ:.....	3
2.1.4 Kiến trúc:	4
2.2 Cài đặt Docker trên Ubuntu Sever	5
2.2.1 Đăng nhập vào Cloud Google (https://cloud.google.com):	5
2.2.2 Tạo máy ảo ubuntu 20.04 LTS	6
2.2.3 Cài đặt Docker:	9
2.2.4 Kiểm tra Docker:.....	9
2.3 Các lệnh cơ bản trong Docker.....	9
2.3.1 Tải image về server:	9
2.3.2 Hiển thị danh sách các images :.....	10
2.3.3 Xóa một images:	11
2.3.4 Chạy một image:.....	11
2.3.1 Liệt kê các container:	16
2.3.2 Dừng container đang chạy:.....	17
2.3.3 Khởi động lại container đã dừng:	18
2.3.4 Truy cập vào một container đang chạy:	18
2.3.5 Xóa container không còn sử dụng:	19

2.3.6	Lưu một images :	19
2.3.7	Tải images từ file đã lưu:	20
2.3.8	Export container :	20
2.3.9	Import container :	21
2.3.10	Tạo thêm một số thuộc tính cho Docker:	21
2.4	Dockerfile:	25
2.4.1	Quy tắc viết chỉ thị Dockerfile:	25
2.4.2	Các chỉ thị Dockerfile:	25
2.5	Docker-compose:	28
2.5.1	Giới thiệu:	28
2.5.2	Sử dụng docker-compose:	28
2.5.3	Quy trình soạn thảo Docker:	29
CHƯƠNG 3 : XÂY DỰNG MÁY CHỦ WEB		32
3.1	Đăng kí tên miền:	32
3.2	Xây dựng máy chủ web:	34
CHƯƠNG 4 : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN		40
4.1	Kết luận:	40
4.1.1	Ưu điểm đã làm được:	40
4.1.2	Hạn chế:	40
4.2	Hướng Phát triển:	40
TÀI LIỆU THAM KHẢO		41

MỤC LỤC HÌNH

Hình 2-1: Mô hình Docker tổng quát	4
Hình 2-2: Mô hình container.....	4
Hình 2-3: Giao diện đăng nhập google cloud	5
Hình 2-4: Chọn VM instances.....	6
Hình 2-5: Tạo VM instances	6
Hình 2-6: Chọn cấu hình VM.....	7
Hình 2-7: VM đã được tạo	8
Hình 2-8: Đăng nhập vào VM.....	8
Hình 2-9: Cài đặt docker.....	9
Hình 2-10: Kiểm tra cài đặt.....	9
Hình 2-11: Tải image ubuntu	10
Hình 2-12: Hiển thị tất cả image	10
Hình 2-13:Xóa image debian	11
Hình 2-14: Chạy một images ubuntu với tên ubuntu và chạy dưới nền.....	15
Hình 2-15:Chạy một images ubuntu với tên ubuntu-1 và truy cập ngay	15
Hình 2-16:Chạy một images ubuntu với tên ubuntu-2, tên vm test và truy cập ngay	15
Hình 2-17:Chạy một images nginx với tên docker-nginx , port 80 và chạy dưới nền.....	16
Hình 2-18:Kiểm tra image đã chạy	16
Hình 2-19:Liệt kê tất cả các container.....	17
Hình 2-20:Dừng container docker-nginx.....	17
Hình 2-21:Kiểm tra lại image đã dừng	17
Hình 2-22:Khởi động lại container và kiểm tra	18
Hình 2-23:Kiểm tra container docker-nginx.....	18
Hình 2-24:Truy cập vào container ubuntu-2.....	19
Hình 2-25:Xóa container ubuntu và kiểm tra	19
Hình 2-26: Lưu image nginx.....	20
Hình 2-27:Tải image nginx từ file.....	20
Hình 2-28:Xuất container thành file nginx:web.tar.gz.....	21
Hình 2-29:Khôi phục lại image	21
Hình 2-30:Tạo images centos được cài đặt “git”	27
Hình 2-31:Hiển thị image đã được tạo	28

Hình 2-32:Kiểm tra "git" đã được cài đặt.....	28
Hình 3-1:Trang web freemon.com	32
Hình 3-2:Nhập tên miền muốn đăng kí	33
Hình 3-3:Thêm tên miền vào giỏ hàng.....	33
Hình 3-4:Thanh toán mua tên miền.....	33
Hình 3-5:Tên miền đã được mua.....	34
Hình 3-6:Thêm địa chỉ ip của host vào tên miền	34
Hình 3-7:Cấu trúc thư mục để tạo server web	35
Hình 3-8:Nội dung file index.php	36
Hình 3-9:Dockerfile mysql	36
Hình 3-10:Dockerfile phpmyadmin	36
Hình 3-11:Dockerfile php:apache	37
Hình 3-12:File docker-compose.....	38
Hình 3-13:Có 3 container đang chạy.....	39
Hình 3-14:Kết nối thành công với cơ sở dữ liệu mysql	39
Hình 3-15:Đăng nhập thành công cơ sở dữ liệu mysql	39

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 dụng trong các môi trường phát triển 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.

CHƯƠNG 1 : TỔNG QUAN VỀ ĐỀ TÀI

1.1 Tên đề tài:

Tìm hiểu về Docker và xây dựng máy chủ Web.

1.2 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.

1.3 Mục tiêu cần đạt:

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

CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về Docker

2.1.1 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.

2.1.2 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!

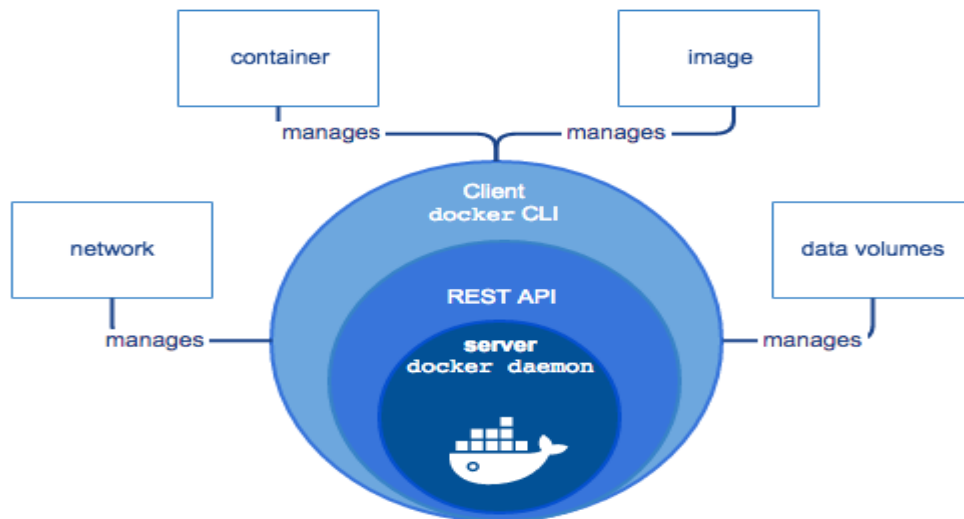
2.1.3 Công cụ:

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 (dockerd). 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.

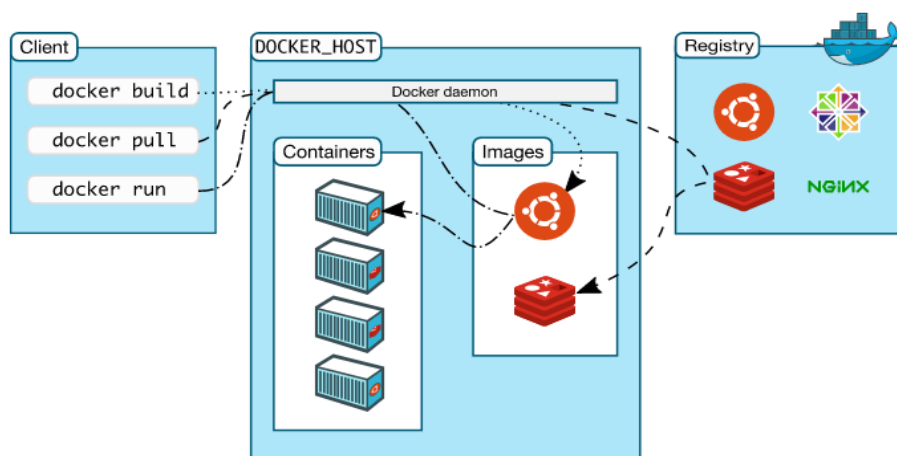
Daemon tạo và quản lý các đối tượng Docker , chẳng hạn như hình ảnh, vùng chứa, mạng và ổ đĩa.



Hình 2-1: Mô hình Docker tổng quát

2.1.4 Kiến trúc:

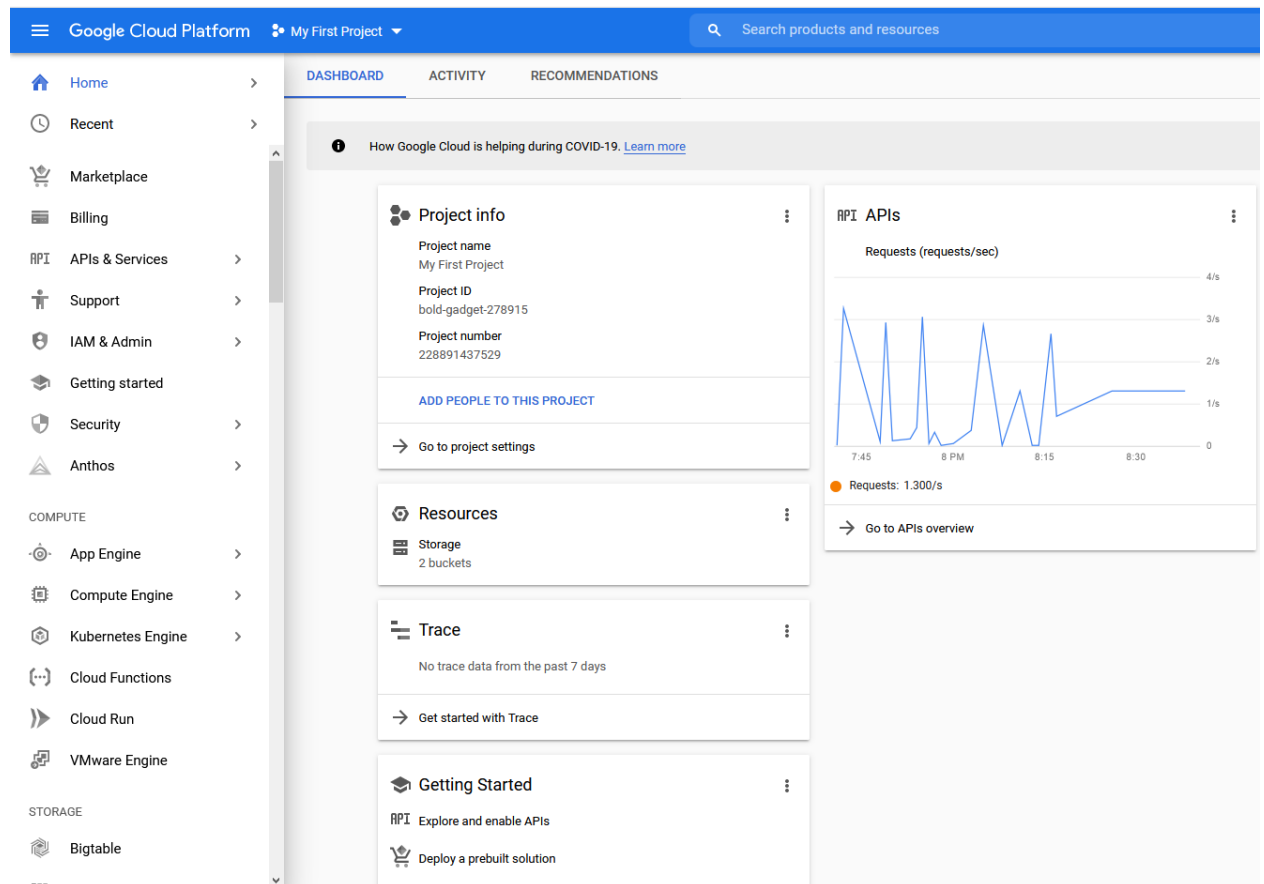
Docker sử dụng kiến trúc máy khách-máy chủ. Ứng dụng khách Docker nói chuyện với Docker daemon, trình nền này thực hiện công việc xây dựng, chạy và phân phối các vùng chứa Docker của bạn. Ứng dụng khách Docker và daemon có thể chạy trên cùng một hệ thống hoặc bạn có thể kết nối ứng dụng khách Docker 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 2-2: Mô hình container

2.2 Cài đặt Docker trên Ubuntu Sever

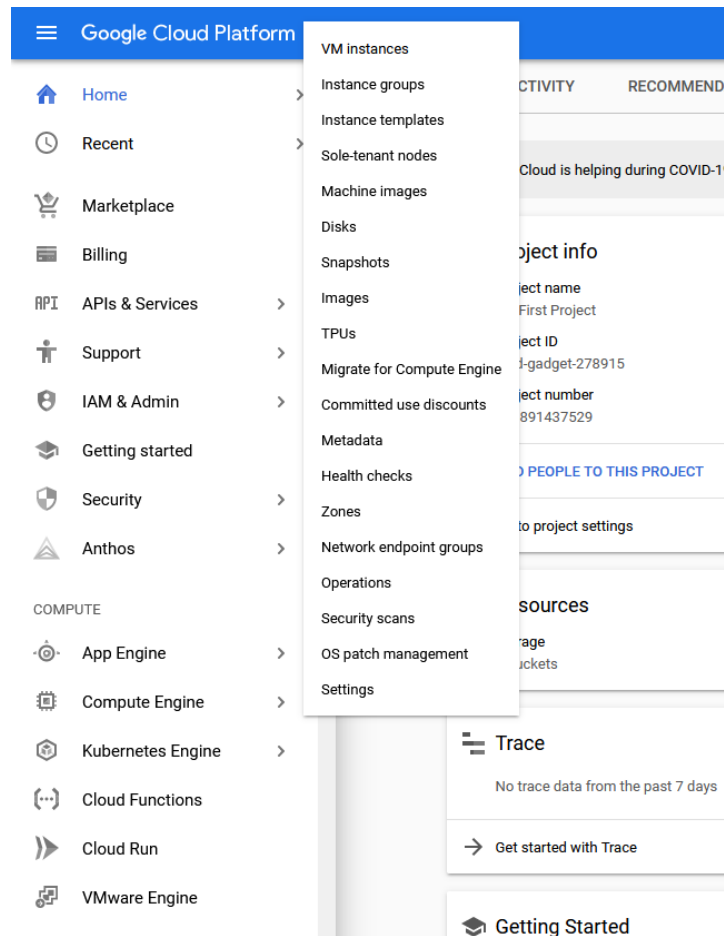
2.2.1 Đăng nhập vào Cloud Google (<https://cloud.google.com>):



Hình 2-3: Giao diện đăng nhập google cloud

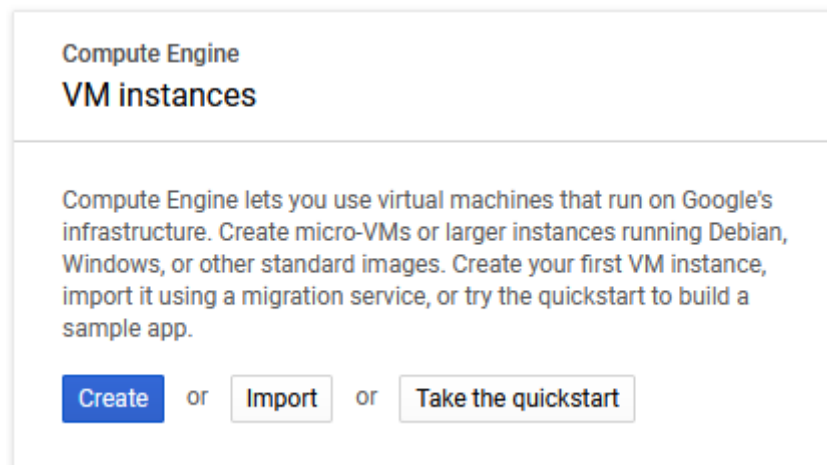
2.2.2 Tạo máy ảo ubuntu 20.04 LTS

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




Hình 2-4: Chọn VM instances


2.2.2.2 Bước 2: Chọn Create





Hình 2-5: Tạo VM instances

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

Name 
Name is permanent

Labels  (Optional)

Region 
Region is permanent

Zone 
Zone is permanent

Machine configuration


Machine family

Machine types for common workloads, optimized for cost and flexibility


Series

CPU platform selection based on availability


Machine type

	vCPU	Memory	GPUs
	1 shared core	2 GB	-


⌵ CPU platform and GPU


Confidential VM service 

☐ Enable the Confidential Computing service on this VM instance.


Container 

☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk 

 New 10 GB standard persistent disk

Image

 Ubuntu 20.04 LTS

Hình 2-6: Chọn cấu hình VM

2.2.2.4 Bước 4: Máy ảo Ubuntu 20.04 LTS đã tạo

VM instances						
<div>CREATE INSTANCE</div> <div>IMPORT VM</div> <div>REFRESH</div> <div>START / RESUME</div> <div>STOP</div> <div>SUSPEND</div> <div>RESET</div> <div>DELETE</div>						
<div>Filter VM instances</div> <div>Columns</div>						
<input type="checkbox"/>	Name ^	Zone	Recommendation	In use by	Internal IP	External IP
<input type="checkbox"/>	docker	us-central1-a			10.128.0.10 (nic0)	34.66.76.255 ↗
						Connect SSH

Hình 2-7: VM đã được tạo

2.2.2.5 Bước 5: Truy cập vào máy Ubuntu

```
minhchau_music1999@docker: ~ - Mozilla Firefox
https://ssh.cloud.google.com/projects/bold-gadget-278915/zones/us-central1-a/instances/docker?useAdminProxy=...
System information as of Fri Sep 25 13:59:43 UTC 2020
System load: 0.06      Processes:            117
Usage of /:  14.1% of 9.52GB   Users logged in:    0
Memory usage: 13%      IPv4 address for ens4: 10.128.0.10
Swap usage:  0%

1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

minhchau_music1999@docker:~$
```

Hình 2-8: Đăng nhập vào VM

2.2.3 Cài đặt Docker:

```
minhchau_music1999@docker:~$ sudo su
root@docker:/home/minhchau_music1999# snap install docker
```

Hình 2-9: Cài đặt docker

2.2.4 Kiểm tra Docker:

```
root@docker:/home/minhchau_music1999# docker version
Client:
 Version:           19.03.11
 API version:       1.40
 Go version:        go1.13.12
 Git commit:        dd360c7
 Built:             Mon Jun  8 20:23:26 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          19.03.11
  API version:       1.40 (minimum version 1.12)
  Go version:        go1.13.12
  Git commit:        77e06fd
  Built:             Mon Jun  8 20:24:59 2020
  OS/Arch:           linux/amd64
  Experimental:      false
 containerd:
  Version:          v1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
root@docker:/home/minhchau_music1999#
```

Hình 2-10: Kiểm tra cài đặt

2.3 Các lệnh cơ bản trong Docker

2.3.1 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

```
root@ubuntu:/home/docker# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
e6ca3592b144: Pull complete
534a5505201d: Pull complete
990916bd23bb: Pull complete
Digest: sha256:cbcf86d7781dbb3a6aa2bcea25403f6b0b443e20b99
8e4b9
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@ubuntu:/home/docker#
```

Hình 2-11: Tải image ubuntu

2.3.2 Hiện thị danh sách các images :

```
docker image [OPTIONS]
```

Cú pháp:

Tùy chọn

Tên, viết tắt	Mặc định	Sự miêu tả
--all , -a		Hiện thị tất cả hình ảnh (mặc định ẩn hình ảnh trung gian)
--digests		Hiện 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 hiện thị ID dạng số

Ví dụ:

```
root@docker:/home/docker# docker images -a
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    9140108b62dc   2 weeks ago   72.9MB
root@docker:/home/docker#
```

Hình 2-12: Hiện thị tất cả image

2.3.3 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:

```
root@docker:/home/docker# docker images -a
REPOSITORY      TAG              IMAGE ID         CREATED          SIZE
nginx           latest          992e3b7be046    6 days ago      133MB
ubuntu         latest          9140108b62dc    2 weeks ago     72.9MB
debian         latest          f6dcff9b59af    4 weeks ago     114MB
root@docker:/home/docker# docker image rm debian
Untagged: debian:latest
Untagged: sha256:439a6bae1ef351ba9308fc9a5e69ff7754c14516f6be8ca26975fb564cb7fb76
Deleted: sha256:f6dcff9b59af55f031c7fe19a19930aa54ac9213986f4def36cee02811758337
Deleted: sha256:4ef54afed7804a44fdeb8cf6562c2a1eb745dcaabd38b1ac60126f0966bf6aef
root@docker:/home/docker# docker images -a
REPOSITORY      TAG              IMAGE ID         CREATED          SIZE
nginx           latest          992e3b7be046    6 days ago      133MB
ubuntu         latest          9140108b62dc    2 weeks ago     72.9MB
root@docker:/home/docker#
```

Hình 2-13: Xóa image debian

2.3.4 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	Đề 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:

```
root@docker:/home/docker# docker run --name ubuntu -d ubuntu
541b183a71b7f9e9a9cf4bc967516029a4b4c77b1db2bb6dcb689080b0ab50
root@docker:/home/docker#
```

Hình 2-14: Chạy một images ubuntu với tên ubuntu và chạy dưới nền

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

```
root@docker:/home/docker# docker run --name ubuntu-1 -it ubuntu
root@24501417c6bb:/#
```

Hình 2-15: 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:

```
root@docker:/home/docker# docker run --name ubuntu-2 --hostname test -it ubuntu
root@test:/#
```

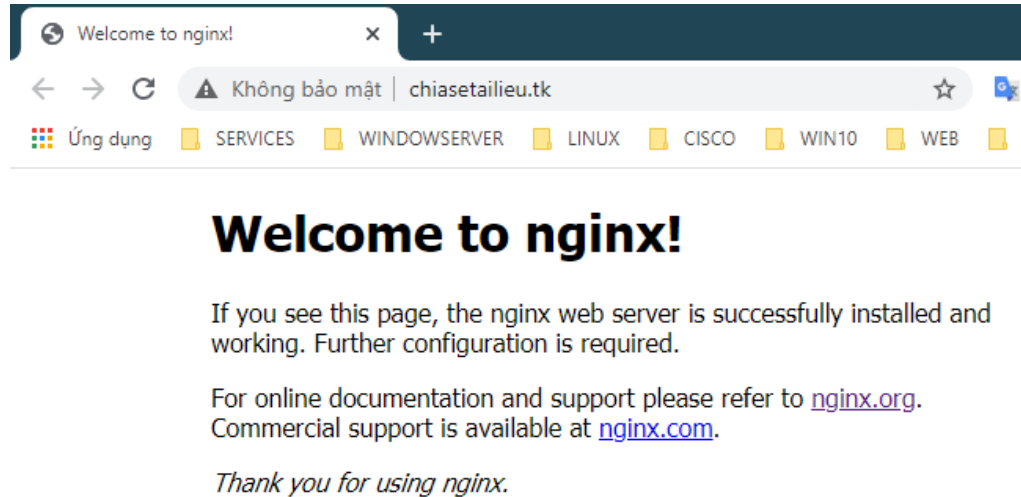
Hình 2-16: Chạy một images ubuntu với tên ubuntu-2, tên vm test và truy cập ngay

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

```
root@docker:/home/docker# docker run --name docker-nginx -d -p 80:80 nginx
73f4531c7a0a8b9cea0033662ba10f879b40e0b72aeb963e74632e7126574f49
root@docker:/home/docker#
```

Hình 2-17: Chạy một images nginx với tên docker-nginx, port 80 và chạy dưới nền

Kiểm tra:



Hình 2-18: Kiểm tra image đã chạy

2.3.1 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:

```

root@docker:/home/docker# docker container ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
73f4531c7a0a   nginx    "/docker-entrypoint..." 7 minutes ago  Up 7 minutes  0.0.0.0:80->80/tcp      docker-nginx
b52e4b9b17c2   ubuntu  "/bin/bash"             11 minutes ago Exited (0) 10 minutes ago
24501417c6bb   ubuntu  "/bin/bash"             16 minutes ago Exited (130) 12 minutes ago  ubuntu-1
541b183a71b7   ubuntu  "/bin/bash"             18 minutes ago Exited (0) 17 minutes ago    ubuntu
root@docker:/home/docker#

```

Hình 2-19: Liệt kê tất cả các container

2.3.2 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:

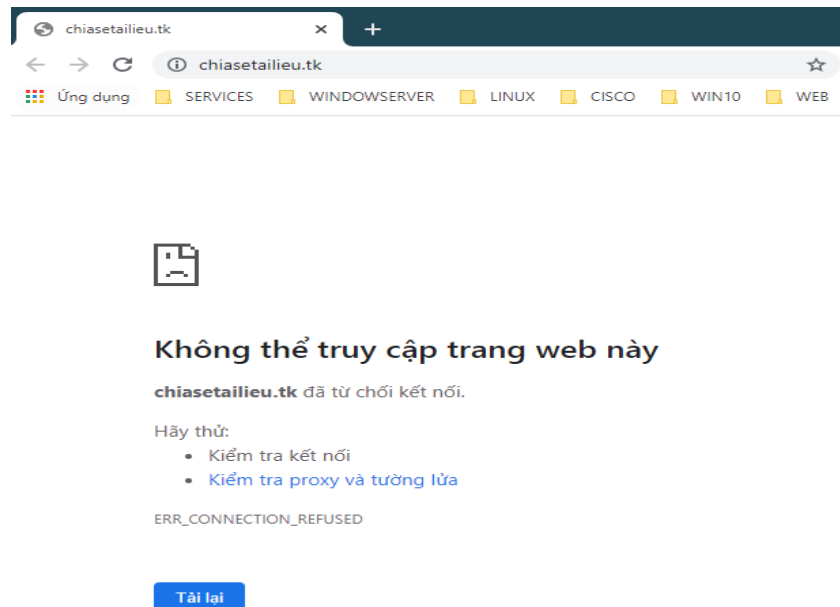
```

root@docker:/home/docker# docker stop docker-nginx
docker-nginx
root@docker:/home/docker# docker container ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
73f4531c7a0a   nginx    "/docker-entrypoint..." 12 minutes ago Exited (0) 3 seconds ago
b52e4b9b17c2   ubuntu  "/bin/bash"             16 minutes ago Exited (0) 14 minutes ago
24501417c6bb   ubuntu  "/bin/bash"             20 minutes ago Exited (130) 17 minutes ago  ubuntu-1
541b183a71b7   ubuntu  "/bin/bash"             22 minutes ago Exited (0) 22 minutes ago    ubuntu
root@docker:/home/docker#

```

Hình 2-20: Dừng container docker-nginx

Kiểm tra:



Hình 2-21: Kiểm tra lại image đã dừng

2.3.3 Khởi động lại container đã dừng:

Cú pháp:

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

Tùy chọn

Tên, viết tắt Mặc định

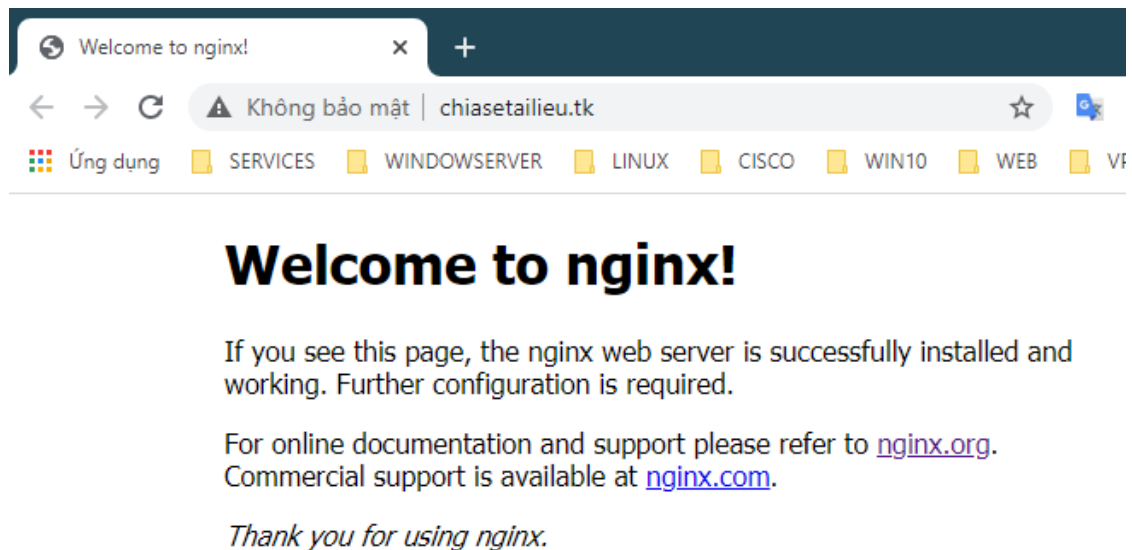
Sự miêu tả

--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:

```
root@docker:/home/docker# docker start docker-nginx
docker-nginx
root@docker:/home/docker# docker container ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS      PORTS               NAMES
73f4531c7a0a   nginx     "/docker-entrypoint..." 18 minutes ago    Up 3 seconds    0.0.0.0:80->80/tcp   docker-nginx
b52e4b9b17c2   ubuntu   "/bin/bash"              22 minutes ago    Exited (0) 20 minutes ago                   ubuntu-2
24501417c6bb   ubuntu   "/bin/bash"              26 minutes ago    Exited (130) 23 minutes ago                  ubuntu-1
541b183a71b7   ubuntu   "/bin/bash"              28 minutes ago    Exited (0) 28 minutes ago                   ubuntu
```

Hình 2-22: Khởi động lại container và kiểm tra



Hình 2-23: Kiểm tra container docker-nginx

2.3.4 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:

```
root@docker:/home/docker# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                NAMES
73f4531c7a0a   nginx    "/docker-entrypoint..." 42 minutes ago Up 24 minutes   0.0.0.0:80->80/tcp    docker-nginx
b52e4b9b17c2   ubuntu  "/bin/bash"              46 minutes ago Up 7 seconds                ubuntu-2
root@docker:/home/docker# docker attach ubuntu-2
root@test:/#
```

Hình 2-24: Truy cập vào container ubuntu-2

2.3.5 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 ổ danh được liên kết với vùng chứa

Ví dụ:

```
root@docker:/home/docker# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                NAMES
73f4531c7a0a   nginx    "/docker-entrypoint..." 44 minutes ago Up 26 minutes   0.0.0.0:80->80/tcp    docker-nginx
b52e4b9b17c2   ubuntu  "/bin/bash"              48 minutes ago Exited (0) 7 seconds ago
24501417c6bb   ubuntu  "/bin/bash"              52 minutes ago Exited (130) 49 minutes ago
541b183a71b7   ubuntu  "/bin/bash"              54 minutes ago Exited (0) 54 minutes ago
root@docker:/home/docker# docker rm ubuntu
ubuntu
root@docker:/home/docker# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                NAMES
73f4531c7a0a   nginx    "/docker-entrypoint..." 46 minutes ago Up 28 minutes   0.0.0.0:80->80/tcp    docker-nginx
b52e4b9b17c2   ubuntu  "/bin/bash"              51 minutes ago Exited (0) 2 minutes ago
24501417c6bb   ubuntu  "/bin/bash"              55 minutes ago Exited (130) 51 minutes ago
root@docker:/home/docker#
```

Hình 2-25: Xóa container ubuntu và kiểm tra

2.3.6 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:

```
root@docker:/home/docker# docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest              992e3b7be046       6 days ago         133MB
ubuntu               latest              9140108b62dc       2 weeks ago        72.9MB
root@docker:/home/docker# docker save nginx > nginx-new.tar
root@docker:/home/docker# ls -sh
total 131M
4.0K DockerFile  4.0K my_web  131M nginx-new.tar  4.0K snap
root@docker:/home/docker#
```

Hình 2-26: Lưu image nginx

2.3.7 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:

```
root@docker:/home/docker# ls -sh
total 376M
4.0K DockerFile  4.0K my_web  114M new-debian.tar  131M nginx-new.tar  131M nginx:new.tar  4.0K snap
root@docker:/home/docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest              992e3b7be046       6 days ago         133MB
ubuntu               latest              9140108b62dc       2 weeks ago        72.9MB
root@docker:/home/docker# docker load < new-debian.tar
4ef54afed780: Loading layer [=====] 119.2MB/119.2MB
Loaded image: debian:latest
root@docker:/home/docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest              992e3b7be046       6 days ago         133MB
ubuntu               latest              9140108b62dc       2 weeks ago        72.9MB
debian               latest              f6dcff9b59af       4 weeks ago        114MB
root@docker:/home/docker#
```

Hình 2-27:Tải image nginx từ file

2.3.8 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:

```

root@docker:/home/docker# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
64efc496aee7       nginx              "/docker-entrypoint..." 3 minutes ago       Up 3 minutes        0.0.0.0:80->80/tcp   docker-nginx
b52e4b9b17c2       ubuntu            "/bin/bash"         2 hours ago         Exited (0) About an hour ago
24501417c6bb       ubuntu            "/bin/bash"         2 hours ago         Exited (130) 2 hours ago
root@docker:/home/docker# docker export docker-nginx > nginx.web.tar.gz
root@docker:/home/docker# ls
DockerFile  my_web  new-debian.tar  nginx.web.tar.gz  snap
root@docker:/home/docker#

```

Hình 2-28: Xuất container thành file nginx.web.tar.gz

2.3.9 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:

```

root@docker:/home/docker# ls
DockerFile  my_web  new-debian.tar  nginx.web.tar.gz  snap
root@docker:/home/docker# docker images -a
REPOSITORY    TAG        IMAGE ID           CREATED            SIZE
nginx          latest     992e3b7be046       6 days ago        133MB
ubuntu         latest     9140108b62dc       2 weeks ago       72.9MB
debian         latest     f6dcff9b59af       4 weeks ago       114MB
root@docker:/home/docker# zcat nginx.web.tar.gz | docker import - nginx-web

gzip: nginx.web.tar.gz: not in gzip format
sha256:ed565b384d1584e2a4a981b37cf594772d3dbc2b5dd52e51abeb8b03ef62b1c8
root@docker:/home/docker# docker images -a
REPOSITORY    TAG        IMAGE ID           CREATED            SIZE
nginx-web     latest     ed565b384d15       5 seconds ago     0B
nginx         latest     992e3b7be046       6 days ago        133MB
ubuntu         latest     9140108b62dc       2 weeks ago       72.9MB
debian         latest     f6dcff9b59af       4 weeks ago       114MB
root@docker:/home/docker#

```

Hình 2-29: Khôi phục lại image

2.3.10 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>])
--usersns		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

2.4 Dockerfile:

2.4.1 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

2.4.2 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:latest 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 số 1", "tham số 2" ...]

VOLUME Trong Dockerfile

Chỉ thị 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 chỉ 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 command_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:

```
root@docker:/home/docker/image-centos# ls
Dockerfile
root@docker:/home/docker/image-centos# cat Dockerfile
FROM centos
RUN yum update -y
RUN yum install git
root@docker:/home/docker/image-centos#
```

Hình 2-30:Tạo images centos được cài đặt “git”

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

Kết quả:

```
root@docker:/home/docker/image-centos# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos               git                273e02bf71c3       24 seconds ago     360MB
<none>              <none>            0b53d5723493       15 minutes ago     98.6MB
nginx-web            latest             ed565b384d15       25 hours ago       0B
nginx                latest             992e3b7be046       7 days ago         133MB
ubuntu               latest             9140108b62dc       2 weeks ago        72.9MB
debian               latest             f6dcff9b59af       4 weeks ago        114MB
centos               latest            0d120b6ccaa8       2 months ago       215MB
root@docker:/home/docker/image-centos#
```

Hình 2-31:Hiển thị image đã được tạo

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

```
[root@0d064ff122f9 /]# git --version
git version 2.18.4
[root@0d064ff122f9 /]#
```

Hình 2-32:Kiểm tra "git" đã được cài đặt

2.5 Docker-compose:

2.5.1 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.

2.5.2 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:

2.5.2.1 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.

2.5.2.2 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.

2.5.2.3 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.

2.5.3 Quy trình soạn thảo Docker:

2.5.3.1 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

2.5.3.2 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:** xác định thư mục gốc, dựa vào thư mục này để khai báo tiếp đường dẫn Dockerfile

2.5.3.3 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 (Mặc 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	Chúng 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

CHƯƠNG 3 : XÂY DỰNG MÁY CHỦ WEB

3.1 Đăng kí tên miền:

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



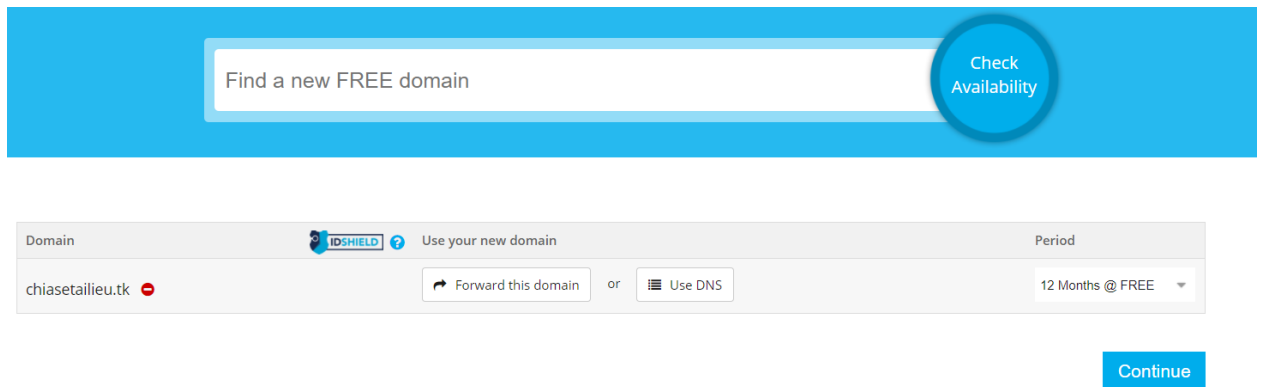
Hình 3-1: Trang web freedom.com

Bước 2: Nhập tên miền và kiểm tra



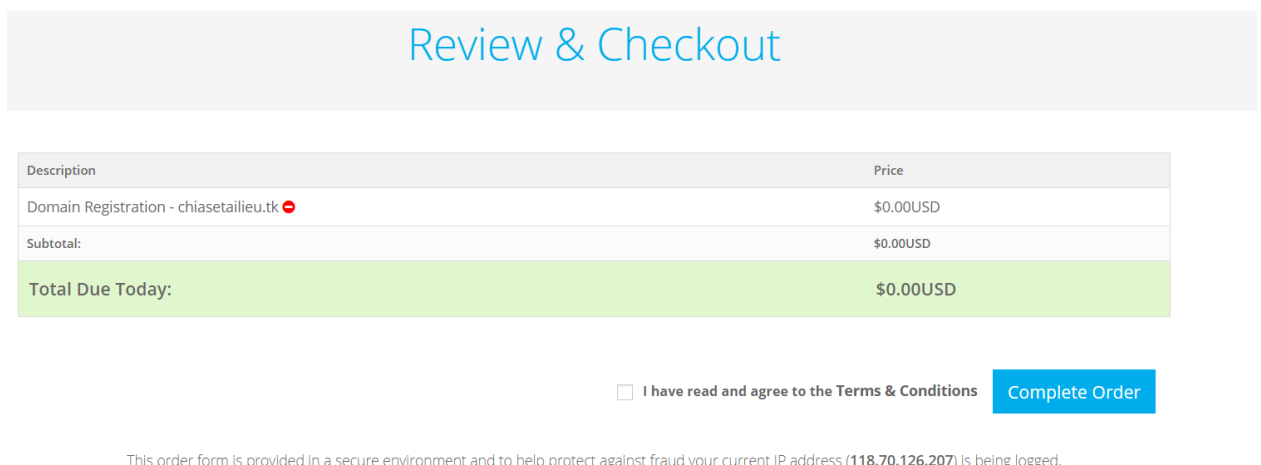
Hình 3-2: Nhập tên miền muốn đăng kí

Bước 3: Thêm vào giỏ hàng



Hình 3-3: Thêm tên miền vào giỏ hàng

Bước 4: Thanh toán

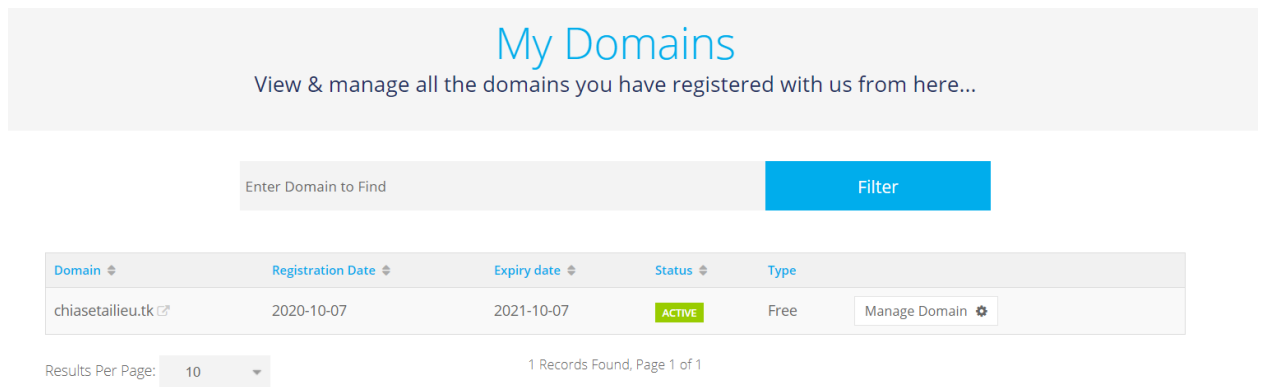


Description	Price
Domain Registration - chiasetailieu.tk	\$0.00USD
Subtotal:	\$0.00USD
Total Due Today:	\$0.00USD

This order form is provided in a secure environment and to help protect against fraud your current IP address (118.70.126.207) is being logged.

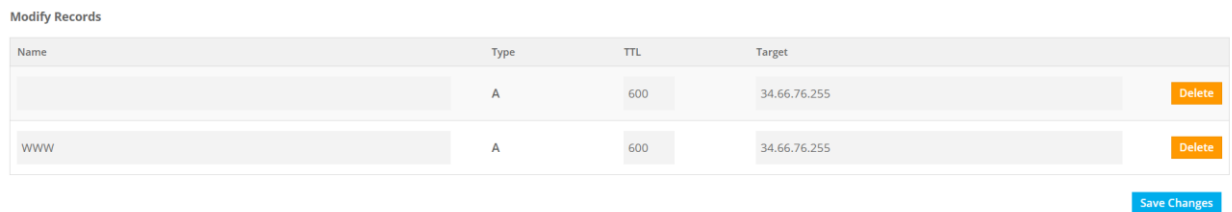
Hình 3-4: Thanh toán mua tên miền

Bước 5: Truy cập vào tên miền của mình



Hình 3-5: Tên miền đã được mua

Bước 6: Thêm địa chỉ ip address (host)



Hình 3-6: Thêm địa chỉ ip của host vào tên miền

3.2 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:

```
root@docker:/home/docker# tree my_web/
my_web/
├── README.md
├── bin
│   ├── mysql
│   │   └── Dockerfile
│   ├── phpmyadmin
│   │   └── Dockerfile
│   └── webserver
│       └── Dockerfile
├── docker-compose.yml
└── www
    └── index.php

5 directories, 6 files
root@docker:/home/docker#
```

Hình 3-7: 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”: chứa 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

```
<?php
$link = mysqli_connect("34.66.76.255", "root", "password","demo");
if (!$link) {
    echo "Error: Unable to connect to MySQL." . PHP_EOL;
    echo "Debugging errno: " . mysqli_connect_errno() . PHP_EOL;
    echo "Debugging error: " . mysqli_connect_error() . PHP_EOL;
    exit;
}
echo "Success: A proper connection to MySQL was made!" . PHP_EOL. "<br/>";
echo "Host information: " . mysqli_get_host_info($link) . PHP_EOL. "<br/>";
echo "MySQL Server version: ".$link->server_version;
mysqli_close($link);
phpinfo();
?>
```

Hình 3-8:Nội dung file index.php

- /bin/mysql/Dockerfile

```
FROM mysql:8.0
```

Hình 3-9:Dockerfile mysql

- /bin/phpmyadmin/Dockerfile

```
FROM phpmyadmin/phpmyadmin:latest

RUN echo -e "\
file_uploads = On\n \
memory_limit = 512M\n \
upload_max_filesize = 512M\n \
post_max_size = 512M\n \
max_execution_time = 600\n \
" > /usr/local/etc/php/conf.d/uploads.ini

EXPOSE 80
```

Hình 3-10:Dockerfile phpmyadmin

- /bin/webserver/Dockerfile

```
FROM php:7.4-apache
RUN apt-get -y update --fix-missing
RUN apt-get upgrade -y
# Install useful tools

RUN apt-get -y install apt-utils nano wget dialog

# Install important libraries

RUN apt-get -y install --fix-missing apt-utils build-essential git curl libcurl4 libcurl4-openssl-dev zip
# Composer
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
# Install xdebug
RUN pecl install xdebug-beta
RUN docker-php-ext-enable xdebug
# Other PHP7 Extensions
RUN apt-get -y install libsqlite3-dev libsqlite3-0 mariadb-client
RUN docker-php-ext-install pdo_mysql
RUN docker-php-ext-install pdo_sqlite
RUN docker-php-ext-install mysqli
RUN docker-php-ext-install curl
RUN docker-php-ext-install tokenizer
RUN docker-php-ext-install json
RUN apt-get -y install zlib1g-dev
RUN apt-get install -y libzip-dev
RUN docker-php-ext-install zip
RUN apt-get -y install libicu-dev
RUN docker-php-ext-install -j$(nproc) intl
# RUN docker-php-ext-install mbstring
RUN docker-php-ext-install bcmath
RUN apt-get install -y libfreetype6-dev libjpeg62-turbo-dev libpng-dev
# RUN docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-dir=/usr/include/
# RUN docker-php-ext-install -j$(nproc) gd
# Enable apache modules
RUN a2enmod rewrite headers
RUN a2enmod ssl
WORKDIR /var/www/html
ADD . /var/www/html
RUN chown -R www-data:www-data /var/www
```

Hình 3-11: Dockerfile php:apache

- docker-compose.yml

```
version: "3"

services:
  webserver:
    build: ./bin/webserver
    ports:
      - "80:80"
    volumes:
      - ./www:/var/www/html/
    networks:
      - default

  mysql:
    build: ./bin/mysql
    container_name: 'png-mysql-test'
    # restart: 'always'
    ports:
      - "3306:3306"
    volumes:
      - ${MYSQL_DATA_DIR}./data/mysql:/var/lib/mysql
      - ${MYSQL_LOG_DIR}./logs/mysql:/var/log/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      # MYSQL_DATABASE: demo
      # MYSQL_USER: user
      # MYSQL_PASSWORD: password

  phpmyadmin:
    build: ./bin/phpmyadmin
    links:
      - mysql
    environment:
      PMA_HOST: mysql
      PMA_PORT: 3306
    ports:
      - '8010:80'
    volumes:
      - /sessions
```

Hình 3-12: File docker-compose

Bước 3: Chạy các file đã viết:

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

Kiểm tra:


```

root@docker:/home/docker/my_web# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
68051102ccda   my_web_phpmyadmin  "/docker-entrypoint..." About a minute ago Up About a minute 0.0.0.0:8010->80/tcp   my_web_phpmyadmin_1
c6ebb64f7509   my_web_mysql      "docker-entrypoint.s..." About a minute ago Up About a minute 0.0.0.0:3306->3306/tcp, 33060/tcp  png-mysql-test
68e681339af8   my_web_webserver  "docker-php-entrypoi..." 7 minutes ago  Up 7 minutes  0.0.0.0:80->80/tcp     my_web_webserver_1
root@docker:/home/docker/my_web#

```

Hình 3-13: Có 3 container đang chạy

PHP Version 7.4.11	
System	Linux 68e681339af8 5.4.0-1024-gcp #24-Ubuntu SMP Sat Sep 5 02:07:13 UTC 2020 x86_64
Build Date	Oct 13 2020 10:00:41
Configure Command	/configure '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-bcmath.ini, /usr/local/etc/php/conf.d/docker-php-ext-intl.ini, /usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-zip.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.NTS
PHP Extension Build	API20190902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

Hình 3-14: Kết nối thành công với cơ sở dữ liệu mysql

Hình 3-15: Đăng nhập thành công cơ sở dữ liệu mysql

CHƯƠNG 4 : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận:

4.1.1 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

4.1.2 Hạn chế:

- Cần phải biết các lệnh về Docker.

4.2 Hướng Phát triển:

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

- Có thể chạy được nhiều container.
- Thích hợp cho xây dựng những app dự án nhiều thành viên.

TÀI LIỆU THAM KHẢO

- [1]. _ <https://docker.com>
- [2]. Xuan Thu _ <https://xuanthulab.net/su-dung-dockerfile-de-tu-dong-cao-cac-image-trong-docker.html>
- [3]. Xuan Thu _ <https://xuanthulab.net/lenh-docker-compose-cao-va-chay-cac-dich-vu-docker.html>