**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---o0o---**



**TIỂU LUẬN**

**ẢO HOÁ ĐIỆN TOÁN ĐÁM MÂY**

**ĐỀ TÀI: TÌM HIỂU VÀ TRIỂN KHAI CÔNG NGHỆ DOCKER**

**Nhóm Ảo ảo ảo**

**Thành phố Hồ Chí Minh, tháng 09 năm 2024**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---o0o---**



**TIỂU LUẬN**

**ẢO HOÁ ĐIỆN TOÁN ĐÁM MÂY**

**ĐỀ TÀI: TÌM HIỂU VÀ TRIỂN KHAI CÔNG NGHỆ DOCKER**

**Nhóm : Ảo ảo ảo**

Trưởng nhóm: Nguyễn Văn Thành

Thành viên:

1. Bùi Tuấn Kiệt
2. Lăng Minh Hải
3. Trần Trọng Quí
4. Nguyễn Minh Sang
5. Nguyễn Hoành Thịnh

**Thành phố Hồ Chí Minh, tháng 09 năm 2024**

**MỤC LỤC**

[GIỚI THIỆU iii](#_Toc178873202)

[Chương 1. GIỚI THIỆU VỀ DOCKER 1](#_Toc178873203)

[1.1. Lịch sử hình thành. 1](#_Toc178873204)

[1.2. Các khái niệm của Docker. 1](#_Toc178873205)

[1.2.a. Docker là gì? 1](#_Toc178873206)

[1.2.b. Docker Engine là gì? 1](#_Toc178873207)

[1.2.c. Docker Hub là gì? 1](#_Toc178873208)

[1.2.d. DockerFile, Images là gì? 1](#_Toc178873209)

[1.2.e. Docker Container là gì? 1](#_Toc178873210)

[1.2.f. Docker Registry là gì? 1](#_Toc178873211)

[1.2.g. Docker Network là gì? 1](#_Toc178873212)

[1.3. Hệ thống lưu trữ của Docker. 1](#_Toc178873213)

[1.3.a. Docker Host là gì? 1](#_Toc178873214)

[1.3.b. Union File System 1](#_Toc178873215)

[1.3.c. Volumes 1](#_Toc178873216)

[1.3.d. Bind Mounts và Tmpfs Mounts 1](#_Toc178873217)

[1.4. Nguyên lý và qui trình hoạt động của Docker. 1](#_Toc178873218)

[1.5. Cài đặt môi trường Docker trên Windows. 1](#_Toc178873219)

[1.5.a. Cấu hình yêu cầu và . 1](#_Toc178873220)

[1.5.b. Cài đặt, Update WSL. 1](#_Toc178873221)

[1.5.c. Cài đặt Docker Desktop. 1](#_Toc178873222)

[1.5.d. Đăng nhập Docker Desktop. 1](#_Toc178873223)

[Chương 2. THỰC NGHIỆM 2](#_Toc178873224)

[2.1. Triển khai môi trường, cấu hình cơ bản cho Docker và đóng gói và triển khai ứng dụng To-Do Lists. 2](#_Toc178873225)

[2.1.a. Tìm base Images trên Docker Hub và các câu lệnh cơ bản. 2](#_Toc178873226)

[2.1.b. Giới thiệu ứng dụng To-Do Lists. 2](#_Toc178873227)

[2.1.c. Tạo Docker Files và xây dựng Docker Images. 2](#_Toc178873228)

[2.1.d. Tạo Container từ Docker Images. 2](#_Toc178873229)

[2.1.e. Cài các thư viện liên quan lên Docker Images. 2](#_Toc178873230)

[2.1.f. Sử dụng Docker Compose. 2](#_Toc178873231)

[2.2. Triển khai máy chủ trên dịch vụ đám mây (Azure). 2](#_Toc178873232)

[2.2.a. Lưu trữ container trên Azure Container Registry (ACR). 2](#_Toc178873233)

[2.2.b. Triển khai Docker Container lên máy chủ. 2](#_Toc178873234)

[2.2.c. Kết nối đến SQL server trên máy chủ. 2](#_Toc178873235)

[2.2.d. Cài đặt, chạy ứng dụng lên máy Client. 2](#_Toc178873236)

[Chương 3. TỔNG KẾT 3](#_Toc178873237)

[3.1. Kết quả thực nghiệm. 3](#_Toc178873238)

[3.2. So sánh Docker và máy ảo (Virtual Machine). 3](#_Toc178873239)

[3.3. Đánh giá ưu và nhược điểm của Docker. 3](#_Toc178873240)

[3.4. Hướng phát triển của Docker trong tương lai. 3](#_Toc178873241)

# GIỚI THIỆU

Kể từ khi Docker ra đời đến nay, theo thống kê lượt tìm kiếm của Google cho từ khoá “Docker” vẫn tăng dần theo từng năm và theo trang thống kê của StackOverflow nền tảng cộng đồng dành cho các lập trình viên, Docker vẫn luôn có chỗ đứng cao. Điều này cho chúng ta thấy sức hút của Docker không hề giảm sút. Vậy vì sao Docker lại mang một độ hot như vậy?

Về cơ bản, Docker cho phép các nhà phát triển gói các ứng dụng cùng với tất cả các thành phần cần thiết như thư viện, framework, và môi trường hệ điều hành vào một thùng chứa được gọi là container. Hãy hình dung một hệ thống server cùng một thời điểm có thể chạy hàng trăm chương trình và mỗi chương trình lại yêu cầu chúng ta phải cài các thư viện để hỗ trợ, nếu như chúng ta cài mọi thứ trực tiếp lên server thì hệ thống sẽ dần trở nên phức tạp, lộn xộn và dễ xảy ra xung đột. Docker sẽ giúp tự động hóa việc triển khai các ứng dụng cũng như là các môi trường được cài đặt ổn định trong các container. Các container này có thể chạy độc lập trên bất kỳ hệ thống nào có cài Docker mà không gặp các vấn đề về tính tương thích môi trường. Docker giúp tối ưu hóa quy trình phát triển, triển khai ứng dụng nhanh chóng và dễ dàng, đồng thời đảm bảo tính nhất quán trong suốt chuỗi cung ứng phần mềm.

Nhìn về thời điểm trước khi Docker ra đời vào năm 2013, các công nghệ container và ảo hóa đã có từ lâu, với nhiều bước phát triển từ Unix V7 năm 1979, FreeBSD Jails năm 2000, đến LXC (Linux Containers) năm 2008. Tuy nhiên, các giải pháp này thường gặp phải một số hạn chế lớn như khó sử dụng và cài đặt phức tạp, thiếu sự đồng nhất và tiêu chuẩn hóa nguyên nhân là do sử dụng máy ảo nên chiếm nhiều tài nguyên, việc triển khai trên nhiều máy ảo khác nhau sẽ gây khó khăn đến tái tạo và quản lý. Khả năng chia sẻ và tái sử dụng bị giới hạn và thiếu môi trường quản lý toàn diện. Từ đó Docker được phát triển và ra đời để khắc phục những vấn đề đang gặp phải.

Khi Docker ra đời, nó tạo ra một bước đột phá lớn so với những sản phẩm chỉ sử dụng LXC (Linux Containers). Docker mang lại những đặc điểm nổi bật như vận chuyển phần mềm nhanh gấp 7 lần so với người không sử dụng Docker. Điều này tạo ra một ưu điểm đáng chú ý về khả năng vận chuyển dịch vụ được tách biệt với tần suất mong muốn. Tiêu chuẩn hóa quá trình vận hành bằng cách đóng gói ứng dụng vào các container nhỏ, làm cho việc triển khai và xác định vấn đề trở nên dễ dàng hơn. Di chuyển trơn tru hiệu quả đạt hiệu suất cao từ việc di chuyển giữa các máy phát triển địa phương đến các môi trường triển khai sản xuất trên AWS. Container Docker giúp chạy nhiều ứng dụng trên mỗi máy chủ một cách dễ dàng, cũng như cải thiện khả năng tận dụng và giảm chi phí đáng kể.

Trong tiểu luận này nhóm sinh viên sẽ thực hiện và trình bày cách cài đặt, sử dụng Docker trên Windows để thực hiện tạo một môi trường khép kín, đóng gói ứng dụng To-Do Lists , các thư viện liên quan và cài đặt môi trường này lên hệ thống máy chủ sử dụng dịch vụ đám mấy (Azure) thông qua 3 chương như sau:

Chương 1: Giới thiệu về Docker. Ở chương này nhóm sẽ trình bày những khái niệm cơ bản, qui trình hoạt động và cách cài đặt lên máy.

Chương 2: Thực nghiệm: Cấu hình cơ bản, tạo, cài đặt và đóng gói thư viện và ứng dụng To-Do Lists. Triển khai môi trường lên dịch vụ đám mây của Azure.

Chương 3: Tổng kết: Đánh giá kết quả thực nghiệm, so sánh và đưa ra ưu nhược điểm và hướng phát triển của công nghệ.

1. GIỚI THIỆU VỀ DOCKER
   1. Lịch sử hình thành.

Docker được phát triển bởi công ty dotCloud, một nền tảng dịch vụ Platform-as-a-Service (PaaS) do Solomon Hykes sáng lập vào năm 2010.

Năm 2013, Docker chính thức được giới thiệu dưới dạng mã nguồn mở, dựa trên nền tảng Linux Containers (LXC), cho phép nhà phát triển đóng gói ứng dụng cùng toàn bộ các thư viện và phụ thuộc của nó trong các container. Sự ra mắt của Docker đã thu hút được sự chú ý lớn từ cộng đồng phát triển phần mềm. Một trong những yếu tố khiến Docker nổi bật so với các công nghệ container khác là khả năng quản lý và triển khai dễ dàng, cũng như việc tích hợp tốt với hệ sinh thái Linux.

Docker nhanh chóng thu hút sự chú ý từ cộng đồng và các công ty công nghệ lớn. Năm 2014, Docker Inc. được thành lập, nhận được đầu tư từ Google, Microsoft, và IBM. Trong cùng năm, Docker Hub ra mắt, trở thành kho lưu trữ trung tâm cho các container. Cũng trong năm này, Docker chuyển từ LXC sang libcontainer để tăng tính linh hoạt trong quản lý container, và giới thiệu Docker Engine như một môi trường triển khai thống nhất.

Để thúc đẩy tiêu chuẩn hóa, Docker cùng nhiều công ty khác thành lập Open Container Initiative (OCI) vào năm 2015, nhằm thiết lập tiêu chuẩn cho container. Docker cũng giới thiệu các công cụ như Docker Compose (quản lý ứng dụng đa container), Docker Swarm (quản lý container trên nhiều máy chủ), và Docker Machine (cài đặt Docker trên máy từ xa).

A diagram of docker

Description automatically generated

1. 1. Các giai đoạn hình thành của Docker

Khi Kubernetes nổi lên như một tiêu chuẩn mới trong quản lý container, Docker tích hợp Kubernetes vào Docker Desktop và Docker Enterprise năm 2017, cho phép người dùng lựa chọn giữa Kubernetes và Docker Swarm. Docker Desktop trở thành công cụ quan trọng cho nhà phát triển trên các hệ điều hành Windows và macOS.

Do khó khăn tài chính, Docker bán lại Docker Enterprise vào năm 2019 cho Mirantis và chuyển hướng tập trung vào Docker Desktop, Docker Hub và các công cụ hỗ trợ nhà phát triển. Ngày nay, Docker vẫn đóng vai trò quan trọng trong DevOps và CI/CD, với Docker Desktop và Docker Hub hỗ trợ mạnh mẽ cho quá trình phát triển ứng dụng.

* 1. Các khái niệm của Docker.
     1. Docker là gì?

Docker là một nền tảng mã nguồn mở dành cho lập trình viên và quản trị hệ thống để xây dựng, vận chuyển, và chạy các ứng dụng phân tán một cách hiệu quả và linh hoạt. Ban đầu được phát triển bằng Python và hiện tại chuyển sang Go-lang, Docker cung cấp giải pháp mới cho vấn đề ảo hóa, khác biệt so với phương pháp truyền thống sử dụng máy ảo (VM) kiểu hypervisor. Docker là một chương trình ảo hoá ứng dụng.

Thay vì tạo ra các máy ảo độc lập với hệ điều hành và phần cứng ảo riêng, Docker đóng gói ứng dụng cùng với các thư viện và phần phụ thuộc vào các container (công-ten-nơ) riêng lẻ. Các container này chia sẻ chung nhân hệ điều hành với nhau thông qua LXC (Linux Containers), sử dụng chung tài nguyên của máy chủ, nhờ đó hoạt động nhẹ và nhanh hơn các máy ảo thông thường. Docker giúp loại bỏ các vấn đề về sự khác biệt giữa các môi trường, mang lại tính nhất quán cao cho quá trình phát triển, kiểm thử, và triển khai ứng dụng.

Với Docker, các container được quản lý và triển khai một cách dễ dàng nhờ vào các công cụ như Docker Engine, Docker Compose, và Docker Hub. Docker đã trở thành một phần quan trọng trong quy trình DevOps, giúp tối ưu hóa hiệu suất, tăng khả năng mở rộng, và đảm bảo tính linh hoạt trong việc quản lý ứng dụng trên nhiều môi trường khác nhau.

A screenshot of a computer

Description automatically generated

1. 2. Hypervisor

A blue container with a blue whale on it

Description automatically generated

1. 3. Docker

* + 1. Docker Engine là gì?

Docker Engine là thành phần cốt lõi của nền tảng Docker, hoạt động như một công cụ xử lý và quản lý các container. Nó giúp tạo ra, vận hành, và quản lý các container trên hệ thống. Docker Engine chịu trách nhiệm cho phép các container chạy trên cùng một máy chủ, chia sẻ tài nguyên hệ điều hành và tạo nên một môi trường chạy nhẹ và độc lập.

Docker Engine bao gồm ba thành phần chính:

Server: Đây là thành phần Docker Daemon (dockerd), chạy dưới nền và chịu trách nhiệm xây dựng, chạy, quản lý các container trên máy chủ Docker. Docker Daemon lắng nghe các yêu cầu từ Docker API và thực hiện các thao tác với container dựa trên yêu cầu đó.

REST API: Đây là giao diện giúp Docker Engine tương tác với các công cụ khác và cho phép người dùng gửi lệnh đến Docker Daemon. Thông qua API này, người dùng có thể thực hiện các thao tác như tạo container, dừng container, xóa container, hoặc quản lý images và network của Docker.

CLI (Command Line Interface): Docker cung cấp giao diện dòng lệnh để người dùng có thể dễ dàng tương tác với Docker Engine bằng các lệnh đơn giản. Công cụ này cho phép người dùng thực hiện các thao tác như chạy, dừng, kiểm tra, và quản lý container bằng cách sử dụng các lệnh Docker như docker run, docker stop, docker ps, và nhiều lệnh khác.

Cách hoạt động của Docker Engine

Khi người dùng thực hiện một lệnh thông qua Docker CLI (ví dụ: docker run), lệnh này sẽ được gửi đến Docker Daemon thông qua REST API. Docker Daemon sẽ thực hiện lệnh đó, tạo ra một container mới từ hình ảnh (image) được chỉ định, sau đó quản lý vòng đời của container này (bao gồm khởi động, dừng, xóa, v.v.).

Docker Engine hỗ trợ tạo và quản lý container một cách hiệu quả, giúp việc triển khai ứng dụng trở nên dễ dàng và nhất quán trên các môi trường khác nhau mà không phụ thuộc vào nền tảng hệ điều hành cụ thể.

A diagram of a server

Description automatically generated

1. 4. Các tầng thành phần của Docker

* + 1. Docker Hub là gì?

Docker Hub là dịch vụ do Docker cung cấp để tìm kiếm và chia sẻ hình ảnh container.

Đây là kho lưu trữ hình ảnh (images) container lớn nhất thế giới với nhiều nguồn nội dung bao gồm các nhà phát triển cộng đồng container, các dự án nguồn mở và các nhà cung cấp phần mềm độc lập (ISV) xây dựng và phân phối mã của họ trong container.

Docker Hub cũng là nơi bạn có thể thực hiện các tác vụ quản trị cho tổ chức . Nếu bạn có đăng ký Docker Team hoặc Business, bạn cũng có thể thực hiện các tác vụ quản trị trong Docker Admin Console .

Các tính năng chính của Docker Hub là:

Repositories cho phép lưu trữ các container images và sử dụng phương thức Push để đẩy dữ liệu lên hub và Pull để kéo dữ liệu về máy.

Teams & Organizations sử dụng để 1uản lý quyền truy cập vào private repositories của container images.

Official Images là một phương thức Pull sử dụng container images chất lượng cao của Docker.

Publisher Images là sử dụng phương thức Pull và sử dụng container images được cung cấp bởi những nhà cung cấp khác.

Builds (Docker Cloud) là một chức năng cho phép kết nối và tự động tạo container images từ GitHub và Bitbucket khi có sự thay đổi và sử dụng phương thức Push để đẩy chúng lên Docker Hub.

Webhooks Kích hoạt tự động các tác vụ sau khi push thành công một repository lên Docker Hub với các dịch vụ khác. Ví dụ như cấu hình webhooks để thông báo cho một URL khi có một image mới được push lên.

* + 1. DockerFile, Images là gì?

Khi nhắc đến Docker, Dockerfile và Images là hai khái niệm cơ bản, giúp hiểu cách Docker xây dựng và quản lý các môi trường container hóa.

* *Khái niệm Docker file:*

Dockerfile là một tệp dữ liệu text có chứa các câu lệnh mà bạn sẽ thực hiện thủ công để xây dựng một docker image. Docker có thể xây dựng các hình ảnh một cách tự động bằng việc đọc các chỉ dẫn của chúng ta trong các dockerfile.

Mỗi một tệp tạo ra một docker image được biết tới như một layer (lớp). Các layers này đến từ các images, được tạo trên các image khác trong nhiều stages. Mỗi một layer độc lập với layer nằm trực tiếp phía dưới của nó. Thứ tự của các layer đóng vai trò then chốt trong việc quản lý vòng đời docker image của bạn môt cách hiệu quả.

* Cách viết một Dockerfile:

Ở trên cùng chúng ta luôn phải chọn một nền (base) cho ứng dụng bằng cách sử dụng From cộng với tên của gói nền đó. Những gói nền này là các images được đăng trên Docker Hub. Ví dụ viết một ứng dụng bằng ngôn ngữ Python, chúng ta cần khai báo một image nền của Python là python:3.9-slim đây là bản rút gọn của python:3.9

FROM python:3.9-slim

LABEL — cung cấp metadata cho image. Có thể sử dụng để add thông tin maintainer. Để xem các label của images, dùng lệnh docker inspect.

ENV — thiết lập một biến môi trường.

RUN — Có thể tạo một lệnh khi build image. Được sử dụng để cài đặt các package vào container.

COPY — Sao chép các file và thư mục vào container.

ADD — Sao chép các file và thư mục vào container.

CMD — Cung cấp một lệnh và đối số cho container thực thi. Các tham số có thể được ghi đè và chỉ có một CMD.

WORKDIR — Thiết lập thư mục đang làm việc cho các chỉ thị khác như: RUN, CMD, ENTRYPOINT, COPY, ADD,…

ARG — Định nghĩa giá trị biến được dùng trong lúc build image.

ENTRYPOINT — cung cấp lệnh và đối số cho một container thực thi.

EXPOSE — khai báo port lắng nghe của image.

VOLUME — tạo một điểm gắn thư mục để truy cập và lưu trữ data.

* *Docker Images*

Docker image là một thành phần quan trọng trong hệ thống Docker, đóng vai trò như một mẫu chuẩn chứa tất cả các thành phần cần thiết để chạy ứng dụng. Cụ thể, một Docker image bao gồm mã nguồn, các thư viện, phụ thuộc và các cấu hình cần thiết cho ứng dụng. Nói cách khác, image là một tập hợp các file hệ thống và file thực thi được gói gọn lại để có thể chạy trong môi trường cô lập của Docker. Các đặc điểm chính của Docker Image:

Đọc Chỉ (Read-Only): Docker image là không thể thay đổi. Khi bạn khởi tạo một container từ một image, Docker sẽ tạo một lớp ghi đè (write layer) lên lớp image chỉ đọc đó.

Tầng (Layers): Một Docker image được xây dựng từ nhiều tầng khác nhau. Mỗi thay đổi trong Dockerfile (cấu hình cho image) tạo ra một tầng mới. Tầng này giúp tiết kiệm dung lượng và tối ưu hóa, bởi nếu nhiều container cùng sử dụng một image, các tầng giống nhau chỉ cần lưu trữ một lần.

Môi trường nhất quán: Docker image đảm bảo rằng ứng dụng sẽ chạy chính xác trong mọi môi trường, từ phát triển đến sản xuất, bởi tất cả phụ thuộc của ứng dụng đều được gói gọn trong image.

Một image sẽ được build dựa trên những chỉ dẫn của Dockerfile.

**A blue whale and black arrows

Description automatically generated with medium confidence**

1. 5. Qui trình tạo căn bản của Docker

* + 1. Docker Container là gì?

Container là một đơn vị phần mềm chuẩn đóng gói mã và tất cả các phụ thuộc của nó để ứng dụng chạy nhanh và đáng tin cậy từ môi trường điện toán này sang môi trường điện toán khác. Docker container image là một gói phần mềm nhẹ, độc lập, có thể thực thi bao gồm mọi thứ cần thiết để chạy ứng dụng: mã, thời gian chạy, công cụ hệ thống, thư viện hệ thống và cài đặt.

Container images trở thành container khi chạy và trong trường hợp Docker container – image trở thành container khi chúng chạy trên Docker Engine . Có sẵn cho cả ứng dụng chạy trên Linux và Windows, phần mềm được chứa trong container sẽ luôn chạy giống nhau, bất kể cơ sở hạ tầng. Container cô lập phần mềm khỏi môi trường của nó và đảm bảo rằng nó hoạt động đồng đều bất chấp sự khác biệt giữa phát triển và dàn dựng.

* Các container Docker chạy trên Docker Engine:

Docker đã tạo ra tiêu chuẩn công nghiệp cho các container, do đó chúng có thể di động ở bất cứ đâu.

Các container chia sẻ nhân hệ điều hành của máy và do đó không yêu cầu hệ điều hành cho mỗi ứng dụng, giúp tăng hiệu quả của máy chủ và giảm chi phí cấp phép và máy chủ.

Các ứng dụng an toàn hơn trong các container và Docker cung cấp khả năng cô lập mặc định mạnh nhất trong ngành.

* + 1. Docker Registry là gì?

Docker Registry là một dịch vụ máy chủ cho phép lưu trữ các docker image của cá nhân, công ty, team,... Dịch vụ Docker Registry có thể được cung cấp bởi tổ chức thứ 3 hoặc là dịch vụ nội bộ được xây dựng riêng nếu bạn muốn.

A diagram of a docker register

Description automatically generated

1. 6.Docker registry

Một số dịch vụ Docker Registry phổ biến như:

* Azure Container Registry
* Docker Hub
* Quay Enterprise
* Google Container Registry.
* AWS Container Registry
  + 1. Docker Network là gì?

Docker network là nơi sẽ đảm nhiệm nhiệm vụ cho container kết nối vào network.

* Các container cùng một network có thể liên lạc với nhau qua tên của container và cổng (port) được lắng nghe của container trên mạng đó.
* Kết nối trên 1 host hoặc nhiều host.
* Kết nối giữa các cụm (swarm) docker containers.
* Kết nối container với các mạng khác nằm ngoài docker.
* Có thể cung cấp hầu hết các chức năng mà một hệ thống mạng bình thường cần có.
* Có 3 loại networks được tự động tạo ra trong docker là:
* bridge
* none
* host

Chúng ta có thể xem bằng lệnh:



A screen shot of a computer

Description automatically generated

Bridge đây là driver mạng mặc định của Docker, bridge là driver phù hợp nhất cho việc giao tiếp các containers độc lập. Các container trong cùng mạng có thể giao tiếp với nhau qua địa chỉ IP, nếu không chỉ định driver thì bridge sẽ là driver mạng mặc định khi khởi tạo.

None driver cung cấp cho một container networking stack và không gian mạng riêng của nó, thường được dùng với mạng tùy chỉnh, driver này không thể dùng trong cụm swarm.

Host dùng khi container cần giao tiếp với host và sử dụng trực tiếp mạng của máy chủ đang chạy

Ngoài ra, Docker còn cung cấp các mạng nâng cao như Overlay và Macvlan:

*Mạng lớp phủ - Overlay network* tạo một mạng phân tán giữa nhiều máy chủ Docker. Kết nối nhiều Docker daemons với nhau và cho phép các cụm services giao tiếp với nhau. Chúng ta có thể sử dụng overlay network để giao tiếp dễ dàng giữa cụm các services với một container độc lập, hay giữa 2 container với nhau ở khác máy chủ Docker Daemons.

Nhờ Overlay network, không cần các công việc thiết lập routing giữa các container thông qua hệ điều hành. Overlay network tạo nên một lớp phủ trên mạng của máy chủ và cho phép container kết nối đến (bao gồm cả các cụm containers) để giao tiếp một cách bảo mật. Docker đảm bảo định tuyến các gói tin đến và đi đúng container đích.

*Mạng Macvlan* cho phép chúng ta gán địa chỉ MAC cho container, điều này làm cho mỗi container như là một thiết bị vật lý trong mạng. Docker daemon định tuyến truy cập tới container bởi địa chỉ MAC. Sử dụng driver macvlan là lựa chon tốt khi các ứng dụng khác cần phải connect đến theo địa chỉ vật lý hơn là thông qua các lớp mạng của máy chủ.

* + 1. Docker Host là gì?

A diagram of a docker architecture

Description automatically generated

1. 7. Kiến trúc Docker

Docker Host là môi trường mà trong đó Docker Engine chạy và quản lý các Docker Containers. Nó có thể là một máy vật lý, một máy ảo, hoặc một dịch vụ đám mây. Docker Host cung cấp các tài nguyên và dịch vụ cần thiết để chạy các containers một cách hiệu quả.

* *Các thành phần của Docker Host:*

**Docker Engine:** Là thành phần chính của Docker, bao gồm Docker Daemon, REST API và CLI để quản lý các containers.

**Containers:** Là các đơn vị ứng dụng được đóng gói, chạy trên Docker Host.

**Images:** Là các khuôn mẫu không thay đổi được sử dụng để tạo ra containers.

* *Chức năng của Docker Host:*

Docker Host là nơi mà Docker Engine hoạt động, thực hiện các lệnh Docker và quản lý vòng đời của các containers.

Ngoài ra còn cung cấp tài nguyên như CPU, RAM, và ổ cứng cho các containers.

Quản lý mạng cho các containers, cho phép chúng giao tiếp với nhau và với các dịch vụ bên ngoài.

* *Lợi ích của Docker Host*

Docker Host là nền tảng quan trọng giúp nâng cao hiệu suất và tối ưu hóa việc sử dụng tài nguyên trong các hệ thống hiện đại. Khi triển khai Docker, các containers chạy trên Docker Host có thể đạt hiệu suất cao do sử dụng tài nguyên một cách tối ưu, giúp cải thiện tốc độ và hiệu quả vận hành. Bên cạnh đó, Docker Host còn cung cấp các công cụ mạnh mẽ giúp cho việc triển khai, cập nhật, và quản lý containers trở nên dễ dàng và thuận tiện hơn.

Khả năng mở rộng là một ưu điểm đáng chú ý của Docker Host. Khi hệ thống phát triển, chúng ta có thể mở rộng quy mô bằng cách bổ sung thêm tài nguyên hoặc triển khai nhiều Docker Hosts trong một cụm hệ thống (cluster), đảm bảo đáp ứng nhu cầu công việc tăng cao.

* 1. Hệ thống lưu trữ của Docker.
     1. Union File System
     2. Volumes
     3. Bind Mounts và Tmpfs Mounts
  2. Nguyên lý và qui trình hoạt động của Docker.
  3. Cài đặt môi trường Docker trên Windows.
     1. Cấu hình yêu cầu và .
     2. Cài đặt, Update WSL.
     3. Cài đặt Docker Desktop.
     4. Đăng nhập Docker Desktop.
     5. Giới thiệu các chức năng có trong Docker Desktop.

1. THỰC NGHIỆM
   1. Triển khai môi trường, cấu hình cơ bản cho Docker và đóng gói và triển khai ứng dụng To-Do Lists.
      1. Tìm base Images trên Docker Hub và các câu lệnh cơ bản.
      2. Giới thiệu ứng dụng To-Do Lists.
      3. Tạo Docker Files và xây dựng Docker Images.
      4. Tạo Container từ Docker Images.
      5. Cài các thư viện liên quan lên Docker Images.
      6. Sử dụng Docker Compose.
   2. Triển khai máy chủ trên dịch vụ đám mây (Azure).
      1. Lưu trữ container trên Azure Container Registry (ACR).
      2. Triển khai Docker Container lên máy chủ.
      3. Kết nối đến SQL server trên máy chủ.
      4. Cài đặt, chạy ứng dụng lên máy Client.
2. TỔNG KẾT
   1. Kết quả thực nghiệm.
   2. So sánh Docker và máy ảo (Virtual Machine).
   3. Đánh giá ưu và nhược điểm của Docker.
   4. Hướng phát triển của Docker trong tương lai.