**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](#_Toc182737730)

[Chương 1. GIỚI THIỆU VỀ DOCKER 1](#_Toc182737731)

[1.1. Lịch sử hình thành. 1](#_Toc182737732)

[1.2. Các khái niệm của Docker. 2](#_Toc182737733)

[1.2.a. Docker là gì? 2](#_Toc182737734)

[1.2.b. Docker Engine là gì? 4](#_Toc182737735)

[1.2.c. Docker Hub là gì? 5](#_Toc182737736)

[1.2.d. DockerFile, Images là gì? 6](#_Toc182737737)

[1.2.e. Docker Container là gì? 9](#_Toc182737738)

[1.2.f. Docker Registry là gì? 10](#_Toc182737739)

[1.2.g. Docker Network là gì? 10](#_Toc182737740)

[1.2.h. Docker Host là gì? 12](#_Toc182737741)

[1.2.i. Tìm hiểu về Docker Build Cloud. 14](#_Toc182737742)

[1.3. Hệ thống lưu trữ của Docker. 14](#_Toc182737743)

[1.3.a. Union File System 14](#_Toc182737744)

[1.3.b. Storage của Docker. 15](#_Toc182737745)

[1.3.c. Volumes, Bind Mounts và Tmpfs Mounts 16](#_Toc182737746)

[1.4. Nguyên lý và qui trình hoạt động của Docker. 18](#_Toc182737747)

[1.5. Cài đặt môi trường Docker trên Windows. 20](#_Toc182737748)

[1.5.a. Cấu hình yêu cầu của hệ thống. 20](#_Toc182737749)

[1.5.b. Cài đặt WSL 2. 21](#_Toc182737750)

[1.5.c. Cài đặt Docker Desktop. 22](#_Toc182737751)

[1.5.d. Đăng nhập Docker Desktop. 24](#_Toc182737752)

[1.5.e. Giới thiệu các chức năng chính có trong Docker Desktop. 25](#_Toc182737753)

[Chương 2. THỰC NGHIỆM 33](#_Toc182737754)

[2.1. Triển khai môi trường, cấu hình cơ bản cho Docker và đóng gói và triển khai ứng dụng To-Do Lists. 33](#_Toc182737755)

[2.1.a. Tìm base Images trên Docker Hub và các câu lệnh cơ bản. 33](#_Toc182737756)

[2.1.b. Giới thiệu ứng dụng To-Do Lists. 33](#_Toc182737757)

[2.1.c. Tạo Docker Files và xây dựng Docker Images. 33](#_Toc182737758)

[2.1.d. Tạo Container từ Docker Images. 33](#_Toc182737759)

[2.1.e. Cài các thư viện liên quan lên Docker Images. 33](#_Toc182737760)

[2.1.f. Sử dụng Docker Compose. 33](#_Toc182737761)

[2.2. Triển khai máy chủ trên dịch vụ đám mây (Azure). 33](#_Toc182737762)

[2.2.a. Lưu trữ container trên Azure Container Registry (ACR). 33](#_Toc182737763)

[2.2.b. Triển khai Docker Container lên máy chủ. 33](#_Toc182737764)

[2.2.c. Kết nối đến SQL server trên máy chủ. 33](#_Toc182737765)

[2.2.d. Cài đặt, chạy ứng dụng lên máy Client. 33](#_Toc182737766)

[Chương 3. TỔNG KẾT 34](#_Toc182737767)

[3.1. Kết quả thực nghiệm. 34](#_Toc182737768)

[3.2. So sánh Docker và máy ảo (Virtual Machine). 34](#_Toc182737769)

[3.3. Đánh giá ưu và nhược điểm của Docker. 34](#_Toc182737770)

[3.4. Hướng phát triển của Docker trong tương lai. 34](#_Toc182737771)

# 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 web booking tour du lịch, 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 web booking tour du lịch. 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. . 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. . Hypervisor

A blue container with a blue whale on it

Description automatically generated

1. . 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. . 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. . 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. .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. . 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. Tìm hiểu về Docker Build Cloud.

Với sự phát triển của công nghệ đám mây, Docker Build Cloud là một dịch vụ của Docker cho phép build images trên môi trường đám mây thay vì trên máy cục bộ. Điều này sẽ giảm tải công việc trên các máy cục bộ hoặc là server on-premises, tăng tốc độ quá trình xây dựng hình ảnh nhờ tài nguyên của các dịch vụ đám mây.

Hiện nay Docker cung cấp dịch vụ xây dựng container trên các dịch vụ đám mây của AWS, Google Cloud, Azure hoặc thông qua các dịch vụ của Docker Hub và Github Action.

* Lấy dịch vụ của Azure làm ví dụ:

Người dùng có thể sử dụng dịch vụ PaaS của Azure là *Azure Container Instances* để triển khai, quản lý và chạy container trực tiếp.

Docker sử dụng dịch vụ của Amazon Web Services (AWS) làm đám mây mặc định. Mỗi builder được cung cấp trên một Amazon EC2 và bộ nhớ cache lưu trữ trên mỗi volume Amazon EBS. Người dùng chỉ cần tương tác với Docker mà không cần làm việc trực tiếp với AWS.

* 1. Hệ thống lưu trữ của Docker.
     1. Union File System

Docker sử dụng Union File System (UFS) làm nền tảng cho hệ thống tệp tin phân lớp của mình vì nó cho phép quản lý các container một cách hiệu quả, linh hoạt và cô lập. UFS cho phép nhiều lớp tệp tin được “xếp chồng” hoặc “kết hợp” thành một cấu trúc thống nhất, điều này rất quan trọng cho cách Docker tạo và quản lý container, đây là lý do tại sao UFS là thiết yếu với Docker.

Ví dụ: Nếu không có UFS, một hình ảnh 200 MB chạy 5 lần với 5 vùng chứa riêng biệt có nghĩa là 1GB dung lượng ổ đĩa.

* *Cấu trúc ảnh phân lớp*

Các ảnh (images) trong Docker được xây dựng thành các lớp, mỗi lớp đại diện cho một tập hợp thay đổi trên hệ thống tệp (ví dụ: thêm tệp, chỉnh sửa cấu hình hoặc cài đặt phần mềm). UFS cho phép kết hợp các lớp này một cách liền mạch. Cấu trúc phân lớp này giúp Docker:

* Tránh việc sao chép các tệp giữa các ảnh và container khác nhau.
* Tiết kiệm dung lượng lưu trữ bằng cách chia sẻ các lớp giữa các ảnh.
* Đơn giản hóa việc cập nhật ảnh khi chỉ cần thêm lớp mới cho các thay đổi, thay vì sao chép toàn bộ ảnh.
  + 1. Storage của Docker.

Storage trong Docker là một tính năng quản lý data của Docker. Data ở đây có thể hiểu là các file được sinh ra trong quá trình chạy ứng dụng. Ví dụ như file log, file data, file report …

Mặc định khi một container được khởi chạy, data trong quá trình vận hành được chứa ở writeable layer và sẽ bị mất đi khi container bị xóa. . Ngoài ra thì một nhược điểm nữa đó là data ở container này khó có thể chia sẻ data với các container khác.

Vậy để giải quyết các vấn đề này, Docker đưa ra một cơ chế để quản lý data của các Container đó là Docker Storage

Về bản chất, Docker Storage là một cơ chế cho phép lưu trữ các data của Container vào Docker Host bằng cách mount một folder từ Docker Container vào Docker Host.

Bằng việc mount này, data trong container giờ đây sẽ được an toàn hơn, dễ dàng chia sẻ giữa các container với nhau hơn. Một số folder chứa setting hay log có thể được đọc dễ dàng hơn trong quá trình troubleshoot các Container.

A diagram of a computer system

Description automatically generated

1. . Các kiểu mount của Docker

* *Có 3 kiểu mount của Docker Storage đó là:*
* Volumes: Mount-point sẽ nằm ở /var/lib/docker/volumes/ của Docker Host và được quản lý bằng Docker.
* bind mounts: Mount-point có thể nằm ở bất kỳ đâu Docker Host không được quản lý bởi Docker.
* tmpfs mounts: Data sẽ được lưu vào memory của Docker Host và sẽ mất đi khi khởi động lại hoặc stop container.
  + 1. Volumes, Bind Mounts và Tmpfs Mounts
* *Volumes*

Volume được quản lý bởi Docker. Trong khi bind mounts, file hoặc thư mục cần mount phải được tồn tại trên docker host. Volume khi tạo ra sẽ nằm ở thư mục /var/lib/docker/volumes/

Volume cũng support cơ chế của volume drivers, cho phép lưu trữ dữ liệu tới một server remote hoặc Cloud …

Bạn có thể quản lý volume sử dụng CLI hoặc API.

Volume hoạt động trên cả Linux và Windows container

* *Sử dụng Volume khi nào ?*
* Khi chia sẻ dữ liệu giữa nhiều container đang chạy.
* Lưu dữ liệu tới một server remote hoặc cloud.
* Khi cần backup, restore hoặc migrate dữ liệu từ Docker Host này sang Docker Host khác.

A screenshot of a computer program

Description automatically generated

1. . Thao tác với Volumes

* *Bind mounts*

Bind mounts trong Docker xuất hiện trước Volume. Bind mounts bị giới hạn một số các tính năng hơn so với volume. Binds mount có hiệu năng tốt nhưng bị phụ thuộc vào cấu trúc filesystem của Docker host.

Khi sử dụng bind mount thì một file hoặc một folder trong docker host được mount vào trong container. File hoặc Folder này yêu cầu đường dẫn tuyệt đối. Vậy nên, trong trường hợp file hoặc folder này không tồn tại trên docker host, thì quá trình mount sẽ bị lỗi.

* *Sử dụng bind mounts khi nào ?*
* Khi chia sẻ file cấu hình từ docker host với container.
* Chia sẻ thư mục source code hoặc một trường từ docker host với container
* Khi cấu trúc filesystem của docker host là rõ ràng và phù hợp với các yêu cầu của bind mount
* *Một số chú ý với binds mount*

Khi sử dụng bind mounts và flag –mount thì phải đảm bảo file hoặc folder từ docker host đã được tồn tại.

* Tmpfs Mounts

Volumes và bind mounts cho phép bạn chia sử dữ liệu giữa docker host và container kể cả khi container bị stop hoặc xóa đi. Nếu bạn sử dụng Docker trên Linux thì bạn có một option thứ ba là tmpfs mounts

* *Sử dụng tmpfs mounts khi nào ?*
* Khi bạn không muốn giữ data trên docker host hoặc trong container.
* Khi muốn bảo mật, hoặc muốn đảm bảo hiệu suất container khi ứng dụng cần ghi một số lượng lớn dữ liệu không liên tục.
* tmpfs mount sẽ lưu dữ liệu tạm thời trên memory. Khi container stop, tmpfs mount sẽ bị xóa bỏ
  1. Nguyên lý và qui trình hoạt động của Docker.

A screenshot of a computer

Description automatically generated

1. . Nguyên lý hoạt động của Docker

Docker daemon (dockerd) lắng nghe các yêu cầu từ Docker API và quản lý các đối tượng Docker như hình ảnh (images), container, mạng (networks), và ổ đĩa (volumes). Daemon cũng có thể giao tiếp với các daemon khác để quản lý các dịch vụ Docker.

Docker client (docker) là cách chính mà nhiều người dùng Docker tương tác với Docker. Khi bạn sử dụng các lệnh như `docker run`, client sẽ gửi các lệnh này tới dockerd, nơi chúng được thực hiện. Lệnh `docker` sử dụng Docker API. Docker client có thể giao tiếp với nhiều daemon.

Docker registry lưu trữ các hình ảnh Docker. Docker Hub là một registry công khai mà ai cũng có thể sử dụng, và Docker sẽ mặc định tìm kiếm hình ảnh trên Docker Hub. Bạn cũng có thể chạy registry riêng của mình.

Khi bạn sử dụng các lệnh `docker pull` hoặc `docker run`, Docker sẽ tải các hình ảnh cần thiết từ registry được cấu hình. Khi bạn sử dụng lệnh `docker push`, Docker sẽ đẩy hình ảnh của bạn tới registry đã cấu hình.

* Ví dụ lệnh **docker run -i -t ubuntu /bin/bash**

Lệnh sau chạy một container `ubuntu`, kết nối tương tác với phiên làm việc dòng lệnh cục bộ của bạn và chạy `/bin/bash`:

Khi bạn chạy lệnh này, các bước sau sẽ xảy ra (giả sử bạn đang sử dụng cấu hình registry mặc định):

* Nếu bạn chưa có image `ubuntu` tại chỗ, Docker sẽ tải nó từ registry đã cấu hình, như thể bạn đã chạy lệnh `docker pull ubuntu` thủ công.
* Docker tạo một container mới, như thể bạn đã chạy lệnh `docker container create` thủ công.
* Docker cấp phát một hệ thống tập tin đọc-ghi cho container, như là lớp cuối cùng của nó. Điều này cho phép container chạy tạo hoặc chỉnh sửa các tập tin và thư mục trong hệ thống tập tin cục bộ của nó.
* Docker tạo một giao diện mạng để kết nối container với mạng mặc định, vì bạn không chỉ định bất kỳ tùy chọn mạng nào. Điều này bao gồm việc cấp phát một địa chỉ IP cho container. Mặc định, các container có thể kết nối với các mạng bên ngoài bằng cách sử dụng kết nối mạng của máy chủ chủ.
* Docker khởi động container và thực thi `/bin/bash`. Vì container đang chạy tương tác và được kết nối với terminal của bạn (nhờ các flag`-i` và `-t`), bạn có thể cung cấp đầu vào bằng bàn phím trong khi Docker ghi lại đầu ra vào terminal của bạn.
* Khi bạn chạy `exit` để kết thúc lệnh `/bin/bash`, container sẽ dừng lại nhưng không bị xóa. Bạn có thể khởi động lại hoặc xóa nó.
  1. Cài đặt môi trường Docker trên Windows.
     1. Cấu hình yêu cầu của hệ thống.
* Các yêu cầu hệ thống chính để chạy Docker Desktop trên Windows.

*Đối với Windows 11 64-bit:* Phiên bản Home hoặc Pro 22H2 trở lên hoặc phiên bản Enterprise hoặc Education 22H2 trở lên.

*Đối với Windows 10 64-bit:* Home hoặc Pro 22H2 (bản dựng 19045) trở lên hoặc Enterprise hoặc Education 22H2 (bản dựng 19045) trở lên.

*Đối với Windows System Linux (WSL):* phiên bản 1.1.3.0 trở lên.

*Ram* hệ thống tối thiểu 4GB.

Lưu ý: Hỗ trợ ảo hoá phần cứng phải được bật bên trong cài đặt của BIOS.

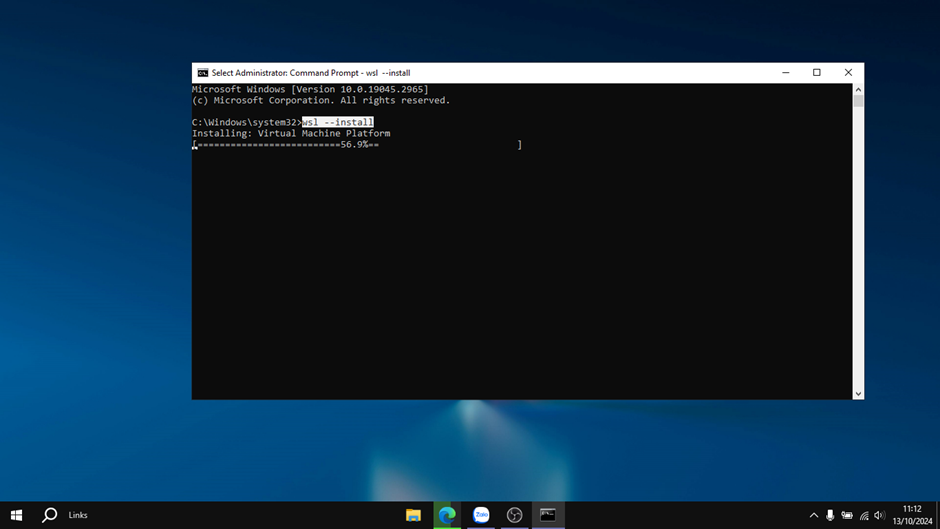
* Vì sao Docker lại cần WSL hoặc là Hyper-V?

Docker hoạt động dựa trên công nghệ container, và các container yêu cầu một nhân nhân Linux để chạy. Tuy nhiên, Windows không có nhân Linux tích hợp sẵn, vì vậy cần một công nghệ để cung cấp môi trường Linux.

Hiện nay Docker khuyến khích người dung sử dụng WSL 2 trên Windows vì WSL 2 tạo nhân Linux ảo hoá bên trong Windows mà không cần cấu hình phức tạp.

* + 1. Cài đặt WSL 2.

Để cài đặt hệ thống WSL trên Windows, chúng ta sử dụng cửa sổ Command Prompt và gõ lệnh *wsl --install* sau đó ấn *Enter.*

****

1. . Cài đặt WSL 2

A screenshot of a computer

Description automatically generated

1. . Khởi động lại hoàn tất cài đặt WSL

* + 1. Cài đặt Docker Desktop.

Truy cập trang web

<https://docs.docker.com/desktop/setup/install/windows-install/#system-requirements>

A screenshot of a computer

Description automatically generated

1. . Tải Docker Desktop

Tải *Docker Desktop for Windows – x86\_64* về máy tính.

* Sau khi hoàn tất tải:

Vào thư mục chưa tệp tải về và khởi chạy *Docker Desktop Installer.exe.*

A screenshot of a computer

Description automatically generated

1. . Quá trình cài đặt Docker Desktop

A screenshot of a computer

Description automatically generated

1. . Lựa chọn WSL 2

* + 1. Đăng nhập Docker Desktop.
* *Sau khi cài đặt hoàn tất, mở ứng dụng Docker Desktop tiến hành đăng nhập.*

A screenshot of a computer

Description automatically generated

1. . Đăng nhập vào Docker

Sau khi ấn *Continue* Docker sẽ chuyển hướng sang trang web đăng nhập để tạo username.

A screenshot of a computer

Description automatically generated

1. . Tạo username

A screenshot of a computer

Description automatically generated

1. . Giao diện Docker Desktop

* + 1. Giới thiệu các chức năng chính có trong Docker Desktop.
* *Các gói tính phí của Docker.*

A screenshot of a website

Description automatically generated

1. . Các gói của Docker

Docker có gói hỗ trợ miễn phí dành cho đối tượng như nhà phát triển mới, học sinh và người dùng muốn tìm hiểu về Docker.

*Hạn chế chính của gói Personal:*

* Số lượng pull các images bị hạn chế 200 lần mỗi 6 tiếng đối với tài khoản đã được xác minh.
* Chỉ có 1 kho lưu trữ riêng tư trên Docker Hub.
* Docker Cloud chỉ cho phép xây dựng 50 phút mỗi tháng và chỉ được chạy song song 1 bản dựng.

Những hạn chế của gói miễn phí sẽ ảnh hưởng đến quá trình trải nghiệm cũng như làm việc nâng cao. Vì vậy Docker cung cấp 3 gói tuỳ chọn dành cho các đối tượng khách nhau bao gồm *Pro, Team và Business*:

* Bản trả phí sẽ được phép sử dụng trong môi trường thương mại, gồm các điều khoản được cấp phép và phù hợp với các doanh nghiệp.
* Phiên bản Business sẽ cung cấp cho doanh nghiệp một tính năng bảo mật cao (Docker Content Trust và quét lỗ hổng bảo mật tự động), giúp doanh nghiệp tuân thủ các tiêu chuẩn an ninh.
* Tăng thời lượng và số lượng đối với docker build cloud tuỳ theo gói tính phí.
* Các bản tính phí sẽ được tăng tốc độ và số lượng pull và không giới hạn số lượng kho riêng tư trên Docker Hub, ngoài ra còn cung cấp các quyền kiểm soát và quản lý người dùng trong tổ chức doanh nghiệp.
* Các phiên bản tính phí sẽ được hỗ trợ trực tiếp và nhanh chóng từ đội kỹ thuật cao của Docker.
* Docker Business cung cấp các tính năng quản lý người dùng, kiểm soát quyền, giám sát, và khả năng tích hợp vào quy trình CI/CD dễ dàng hơn.
* Các tính năng trong Docker Desktop.

A screenshot of a computer

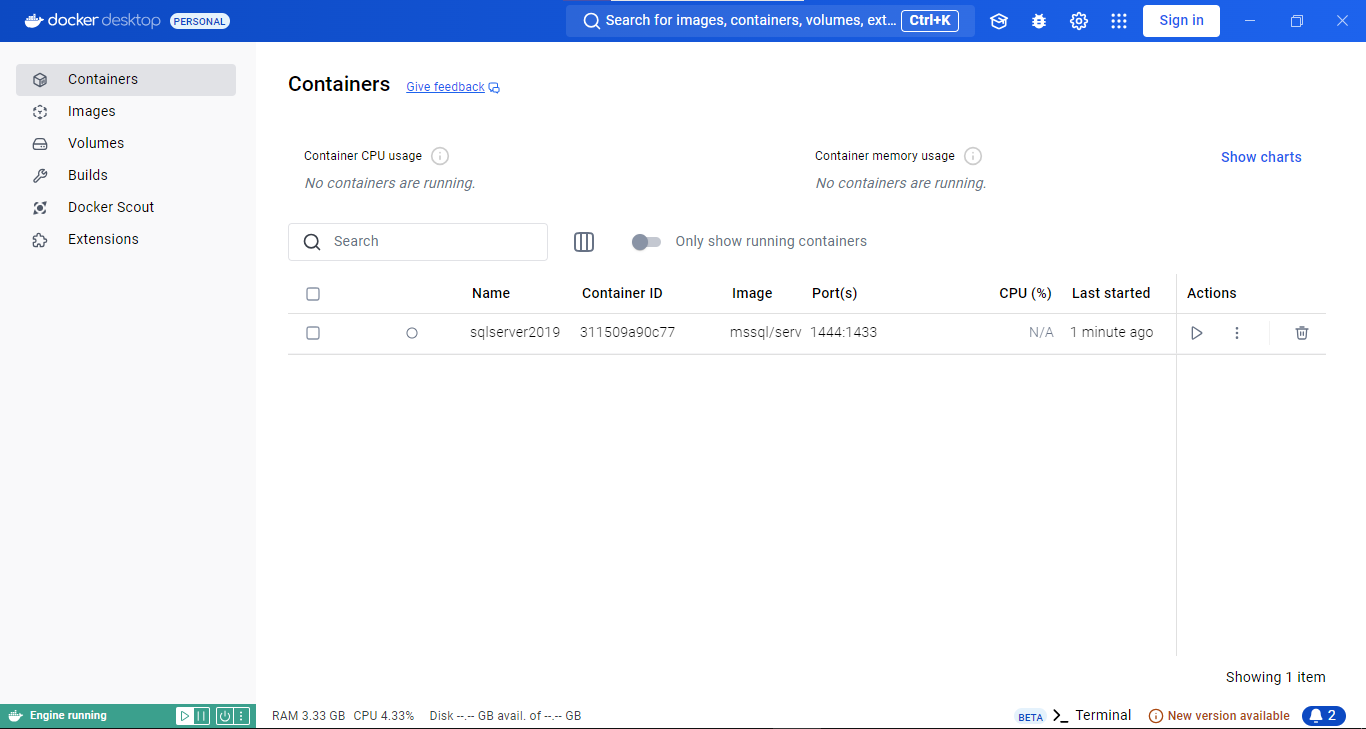
Description automatically generated

1. . Tính năng chính trong Docker Desktop

**Container:**

Hoạt động giống như 1 thư mục chứa tất cả những thứ cần thiết để 1 ứng dụng có thể chạy được. Mỗi 1 docker container được tạo ra từ 1 docker image. Các thao tác với 1 container: chạy, mở, dừng, di chuyển và xóa.Các file và setting được sử dụng trong container được lưu và sử dụng lại gọi chung là image của docker. Docker hub là nơi chia sẻ và lưu trữ các file image này.

Giao diện Containers liệt kê tất cả các container và ứng dụng đang chạy. Để xem danh sách này, bạn cần có các container hoặc ứng dụng đang chạy hoặc đã dừng. Ngoài ra còn xem được mức tiêu thụ tài nguyên của container.



1. . Container

**Images:**

Giao diện Images giúp bạn quản lý các Docker images mà không cần sử dụng lệnh trong CLI(giao diện các dòng lệnh). Mặc định, nó hiển thị danh sách tất cả các images có trên ổ đĩa cục bộ.

Nếu bạn đăng nhập vào Docker Hub, bạn có thể xem các images trên Hub, giúp bạn hợp tác với đội nhóm và quản lý images trực tiếp từ Docker Desktop.

A screenshot of a computer

Description automatically generated

1. . Images

**Volumes:**

Giao diện Volumes trong Docker Dashboard cho phép bạn tạo, xóa và thực hiện các thao tác khác trên các volumes. Bạn cũng có thể xem volumes nào đang được sử dụng và kiểm tra các file, thư mục bên trong volumes.

Ở giai đoạn của các công nghệ ảo hoá lên đám mây, tính năng xuất volume hiện đang ở giai đoạn Beta. Bạn có thể xuất nội dung của một volume ra file, hình ảnh Docker (image), hoặc lên Docker Hub, hay cloud provider được hỗ trợ như Amazon Web Services, Microsoft Azure và Google Cloud...

A screenshot of a computer

Description automatically generated

1. . Volumes

**Builds:**

Builds view cung cấp một giao diện đơn giản để xem lịch sử build và quản lý các builders. Bạn có thể kiểm tra các build đã hoàn thành và các build đang chạy, đồng thời quản lý các builder cục bộ hoặc builder trên đám mây.

**A screenshot of a computer

Description automatically generated**

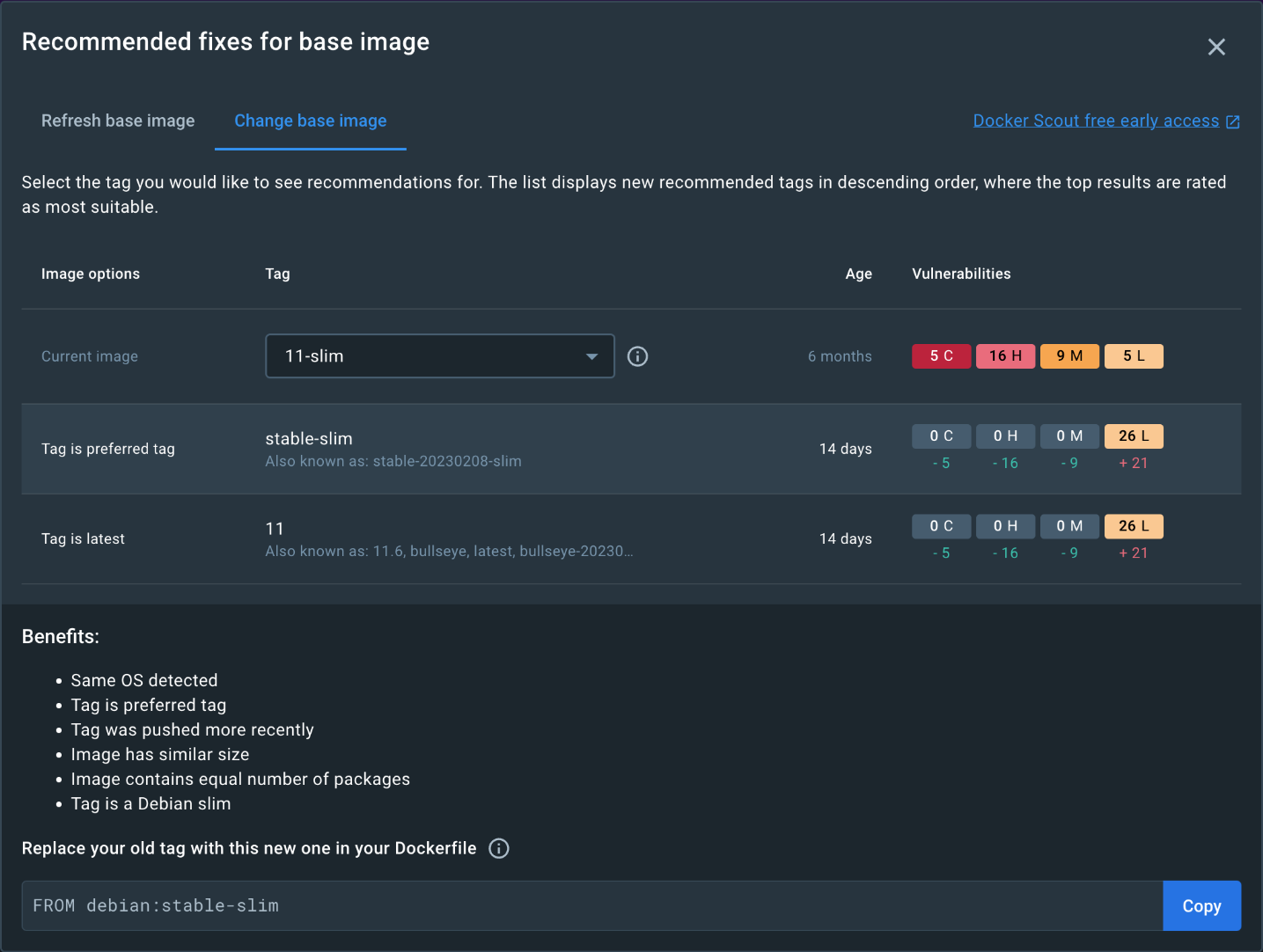
1. . Builds

**Docker Scout:**

Docker Scout là một công cụ được thiết kế để quản lý bảo mật và giám sát các container image. Docker Scout tập trung vào việc cung cấp thông tin chi tiết về lỗ hổng bảo mật, các bản vá, và các gợi ý tối ưu hóa cho các images mà bạn đang sử dụng hoặc phát triển. Nó giúp cải thiện tính bảo mật và độ tin cậy của hệ thống container.

Docker Scout tích hợp trực tiếp với Docker Hub và các công cụ CI/CD như GitHub Actions, giúp bạn thực hiện kiểm tra bảo mật tự động trong suốt quá trình phát triển.

Cơ bản Docker cho phép phân tích trên Scout một image có dung lượng tối đa là 10GB. Và chỉ cho tối đa 3 kho lưu trữ cho tài khoản miễn phí và phân tích images không giới hạn.



1. . Docker Scout

**Extensions:**

Docker Desktop Extensions là tính năng cho phép người dùng mở rộng chức năng của Docker Desktop thông qua các tiện ích (extensions) từ bên thứ ba hoặc do chính Docker phát triển. Các extension này giúp bạn thêm các công cụ, tiện ích mới trực tiếp vào Docker Desktop để dễ dàng tích hợp với các quy trình phát triển hiện tại.

A screenshot of a computer

Description automatically generated

1. . Extensions

1. THỰC NGHIỆM
   1. Triển khai môi trường, cấu hình cơ bản cho Docker và đóng gói và triển khai web booking tour du lịch.
      1. Giới thiệu web booking tour du lịch.

Web booking-vivu là một trang web dành cho những ai thích đi du lịch, web giúp người dùng tìm kiếm và đặt tour du lịch trực tiếp.

A screenshot of a website

Description automatically generated

2. . Giới thiệu Booking-vivu

A screenshot of a travel website

Description automatically generated

2. . Giới thiệu về Booking-vivu

Trang web được xây dựng trên công nghệ MVC ASP.NET và SQL Server. Sử dụng công nghệ đám mây Azure SQL Database và .

* + 1. Tạo Docker Files và xây dựng Docker Images.

Tại thư mục chứa dự án. Sử dụng CMD và gọi lệnh *Docker init* để khởi tạo các file.

A screenshot of a computer program

Description automatically generated

2. . Khởi tạo Docker

A screenshot of a computer

Description automatically generated

2. . Chọn và khởi tạo Docker

A screenshot of a computer

Description automatically generated

2. . Docker file

Vẫn ở tại thư mục của dự án mở CMD và chạy lệnh **Docker build -t booking-vivu .** để xây dụng image.

A screen shot of a computer

Description automatically generated

Lỗi ngoài dự tính (.net 4.8 đã bị xoá image khỏi docker hub nên không tải đóng gói được)

* + 1. Tạo Container từ Docker Images.
    2. Sử dụng Docker Compose.
  1. Triển khai trên dịch vụ đám mây (Azure).
     1. Tạo và kết nối đến SQL server bằng dịch vụ SQL Database của Azure.
* Tạo database trên Azure SQL Database.

Vào mục *SQL Database* và chọn *create* để tạo database mới.

A screenshot of a computer

Description automatically generated

2. . Tạo database trên Azure

Cấu hình database.

A screenshot of a computer

Description automatically generated

2. . Cấu hình server database và database

Chọn tạo mới Server.

A screenshot of a computer

Description automatically generated

2. . Tạo server database.

Chọn *configure database* để chọn và cấu hình máy và kho lưu trữ.

A screenshot of a computer

Description automatically generated

2. . Cấu hình máy và storage

A screenshot of a computer

Description automatically generated

2. . Cấu hình máy và storage

Sau khi *Create,* đăng nhập vào *Query Editor* để tạo bảng dữ liệu.

A screenshot of a computer

Description automatically generated

2. . Đăng nhập Query Editor

A screenshot of a computer

Description automatically generated

2. . Tạo bảng dữ liệu

Sau khi hoàn tất vào mục *connection string* để lấy chuỗi kết nối với cơ sở dữ liệu.

A screenshot of a computer

Description automatically generated

2. . Chuỗi kết nối database

A screenshot of a computer

Description automatically generated

2. . Chuỗi kết nối database

* + 1. Lưu trữ container trên Azure Container Registry (ACR).
    2. Triển khai Docker Container lên máy chủ.

1. 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.