**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙞 🕮 🙜

A picture containing room

Description automatically generated

**BÁO CÁO MÔN HỌC**

**CÁC CÔNG NGHỆ MỚI TRONG PHÁT TRIỂN PHẦN MỀM**

**Giảng viên hướng dẫn: GS.TS. Phan Viết Hoàng**

**Họ và Tên: Đặng Ngọc Thắng**

**MSSV: 2012378**

**Lớp: CTK44 - PM**

***Đà Lạt, tháng 12 năm 2023***

**Mục lục**

[**1.** **Tổng quan công nghệ DevOps** 5](#_Toc153804835)

[**1.1.** **Định nghĩa** 5](#_Toc153804836)

[**1.2.** **Ứng dụng** 5](#_Toc153804837)

[**1.3.** **Lịch sử phát triển** 6](#_Toc153804838)

[**1.4.** **Các tính năng của DevOps** 6](#_Toc153804839)

[**1.5.** **Ưu điểm và nhược điểm của DevOps** 7](#_Toc153804840)

[**2.** **Quy trình DevOps** 8](#_Toc153804841)

[**2.1.** **Kiến trúc của DevOps** 8](#_Toc153804842)

[**2.2.** **Quy trình làm việc** 9](#_Toc153804843)

[**3.** **Công cụ DevOps** 11](#_Toc153804844)

[**3.1.** **Phát triển Code** 11](#_Toc153804845)

[**3.1.1.** **Git** 11](#_Toc153804846)

[**3.1.2.** **Selenium** 11](#_Toc153804847)

[**3.2.** **Tích hợp** 11](#_Toc153804848)

[**3.2.1.** **Jenkins** 11](#_Toc153804849)

[**3.3.** **Triển khai** 12](#_Toc153804850)

[**3.3.1.** **Docker** 12](#_Toc153804851)

[**3.4.** **Giám sát** 12](#_Toc153804852)

[**3.4.1.** **Nagios** 12](#_Toc153804853)

[**4.** **Giới thiệu Docker** 13](#_Toc153804854)

[**4.1.** **Containerization** 13](#_Toc153804855)

[**4.2.** **Độc Lập Tài Nguyên** 13](#_Toc153804856)

[**4.3.** **Di Động và Tương Thích** 13](#_Toc153804857)

[**4.4.** **Docker Image** 13](#_Toc153804858)

[**4.5.** **Dockerfile** 14](#_Toc153804859)

[**4.6.** **Docker Compose** 14](#_Toc153804860)

[**4.7.** **Docker Swarm và Kubernetes** 14](#_Toc153804861)

[**5.** **Images & Containers** 14](#_Toc153804862)

[**5.1.** **Docker Images** 14](#_Toc153804863)

[**5.2.** **Docker Containers** 15](#_Toc153804864)

[**6.** **Data & Volumes** 16](#_Toc153804865)

[**6.1.** **Docker Data** 16](#_Toc153804866)

[**6.2.** **Docker Volumes** 16](#_Toc153804867)

[**7.** **Networking** 17](#_Toc153804868)

[**7.1.** **Bridge Network** 17](#_Toc153804869)

[**7.2.** **Host Network** 17](#_Toc153804870)

[**7.3.** **Overlay Network** 17](#_Toc153804871)

[**7.4.** **Custom Bridge Network** 18](#_Toc153804872)

[**7.5.** **Macvlan Network** 18](#_Toc153804873)

[**7.6.** **Linking Containers** 18](#_Toc153804874)

[**7.7.** **Exposed Ports** 19](#_Toc153804875)

[**8.** **Client-Server Application** 19](#_Toc153804876)

[**8.1.** **Xây Dựng Docker Image cho Server** 19](#_Toc153804877)

[**8.2.** **Xây Dựng Docker Image cho Client** 20](#_Toc153804878)

[**8.3.** **Tạo Docker Network cho Giao Tiếp giữa Client và Server** 20](#_Toc153804879)

[**8.4.** **Chạy Container cho Client và Kết Nối với Server** 20](#_Toc153804880)

[**8.5.** **Cấu Hình Client để Kết Nối với Server** 20](#_Toc153804881)

[**9.** **Các bài thực hành** 20](#_Toc153804882)

[**9.1.** **Bài tập thực hành 1 (Lệnh)** 20](#_Toc153804883)

[**9.2.** **Bài tập thực hành 2 (Hello)** 21](#_Toc153804884)

[**9.3.** **Bài tập thực hành 3 (Goal)** 22](#_Toc153804885)

[**9.4.** **Bài tập thực hành 4 (NotePython)** 23](#_Toc153804886)

[**9.5.** **Bài tập thực hành 5 (DockerHub)** 25](#_Toc153804887)

[**9.6.** **Bài tập thực hành 6 (Volumn)** 26](#_Toc153804888)

[**9.7.** **Bài tập thực hành 7 (Network)** 27](#_Toc153804889)

[**9.8.** **Bài tập thực hành 8 (Client-Server)** 29](#_Toc153804890)

LỜI CẢM ƠN

Đầu tiên em xin được gửi lời cảm ơn tới Thầy giáo Phan Viết Hoàng – người đã hướng dẫn tận tình, chỉ bảo và đóng góp những ý kiến quý giá trong học phần “Các Công nghệ mới trong phát triển phần mềm”. Thông qua bài tập lớn kết thúc học phần em đã có cơ hội hiểu sâu sắc hơn về môn học cũng như trang bị được những hành trang cần thiết cho con đường nghề nghiệp sau này.

Em xin chân thành cảm ơn đến các thầy cô trong Khoa Công nghệ Thông tin và các phòng ban nhà trường đã tạo điều kiện tốt nhất cho em và các bạn trong suốt quá trình học tập và thực hiện báo cáo.

Trong quá trình làm đề tài này mặc dù đã gặp nhiều khó khăn, kèm theo đó kiến thức còn hạn hẹp và thời gian thực hiện hạn chế nhưng em đã cố gắng hết sức để hoàn thiện báo cáo. Tuy nhiên, sẽ không tránh khỏi những thiếu sót khi tìm kiếm và tổng hợp nội dung. Em rất mong nhận được sự góp ý của Thầy để bài báo cáo của em được hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

1. **Tổng quan công nghệ DevOps**
   1. **Định nghĩa**

Giai đoạn phát triển (development) bao gồm phần việc của UI designer, developer, QA/QC…

Giai đoạn vận hành (operations) có sự tham gia của system engineer, system administrator, operation executive, release engineer, DBA, network engineer.

Hai giai đoạn này tương đối tách rời nhau. Đặc biệt là ở những công ty có quy mô từ trung bình trở lên. Chính vì thế, khái niêm devops ra đời nhằm tối ưu hóa chu trình phát triển phần mềm. Giúp sản phẩm phần mềm được release nhanh và thường xuyên hơn.

* 1. **Ứng dụng**

DevOps cũng có thể được định nghĩa là một chuỗi phát triển và hoạt động CNTT với giao tiếp và cộng tác tốt hơn.

DevOps đã trở thành một trong những lĩnh vực kinh doanh có giá trị nhất đối với các doanh nghiệp hoặc tổ chức. Với sự trợ giúp của DevOps, chất lượng và tốc độ phân phối ứng dụng đã được cải thiện rất nhiều.

DevOps không gì khác ngoài một phương pháp thực hành hoặc phương pháp làm cho những người “developer” và “Người vận hành” làm việc cùng nhau. DevOps đại diện cho một sự thay đổi trong văn hóa CNTT với sự tập trung hoàn toàn vào việc cung cấp dịch vụ CNTT nhanh chóng thông qua việc áp dụng các phương pháp linh hoạt trong bối cảnh của cách tiếp cận theo định hướng hệ thống.

DevOps là tất cả về việc tích hợp các hoạt động và quá trình phát triển. Các tổ chức đã áp dụng DevOps nhận thấy chất lượng phần mềm được cải thiện 22% và tần suất triển khai ứng dụng cải thiện 17% và đạt được mức độ hài lòng của khách hàng tăng 22% và 19% doanh thu tăng do triển khai DevOps thành công.

* 1. **Lịch sử phát triển**

Năm 2009, hội nghị đầu tiên mang tên DevOpsdays được tổ chức tại Ghent, Bỉ. Cố vấn người Bỉ và Patrick Debois thành lập hội nghị.

Năm 2012, báo cáo trạng thái DevOps được Alanna Brown tại Puppet đưa ra và hình thành.

Năm 2014, báo cáo State of DevOps hàng năm được xuất bản bởi Nicole Forsgren, Jez Humble, Gene Kim và những người khác. Họ nhận thấy việc áp dụng DevOps cũng đang tăng tốc trong năm 2014.

Năm 2015, Nicole Forsgren, Gene Kim và Jez Humble thành lập DORA (DevOps Research and Assignment).

Năm 2017, Nicole Forsgren, Gene Kim và Jez Humble đã xuất bản “Tăng tốc: Xây dựng và mở rộng quy mô các tổ chức công nghệ hiệu suất cao”.

* 1. **Các tính năng của DevOps**

**Tự động hóa**

Tự động hóa có thể giảm tiêu thụ thời gian, đặc biệt là trong giai đoạn thử nghiệm và triển khai. Năng suất tăng lên và các bản phát hành được thực hiện nhanh hơn nhờ tự động hóa. Điều này sẽ dẫn đến việc bắt lỗi nhanh chóng để có thể dễ dàng sửa lỗi. Đối với phân phối liền kề, mỗi code được xác định thông qua các thử nghiệm tự động, các dịch vụ dựa trên đám mây và các bản build. Điều này thúc đẩy sản xuất bằng cách sử dụng triển khai tự động.

**Hợp tác**

Nhóm Phát triển và Vận hành cộng tác với tư cách là một nhóm DevOps, nhằm cải thiện mô hình văn hóa khi các nhóm trở nên hiệu quả hơn với năng suất của họ, giúp tăng cường trách nhiệm giải trình và quyền sở hữu. Các nhóm chia sẻ trách nhiệm của họ và làm việc đồng bộ chặt chẽ, do đó làm cho việc triển khai đến sản xuất nhanh hơn.

**Tích hợp**

Các ứng dụng cần được tích hợp với các thành phần khác trong môi trường. Giai đoạn tích hợp là code hiện có được kết hợp với chức năng mới và sau đó được kiểm tra. Tích hợp và thử nghiệm liên tục (CI/CD) cho phép phát triển liên tục. Tần suất trong các bản phát hành và các dịch vụ vi mô dẫn đến những thách thức hoạt động đáng kể. Để khắc phục những vấn đề như vậy, việc tích hợp và phân phối liên tục được thực hiện để phân phối nhanh hơn, an toàn hơn và đáng tin cậy hơn.

**Quản lí cấu hình**

DevOps đảm bảo ứng dụng chỉ tương tác với những tài nguyên có liên quan đến môi trường mà DevOps chạy. Các file cấu hình không được tạo trong đó cấu hình bên ngoài của ứng dụng được tách biệt khỏi code nguồn. File cấu hình có thể được viết trong quá trình triển khai hoặc chúng có thể được tải vào lúc chạy, tùy thuộc vào môi trường mà file đang chạy.

* 1. **Ưu điểm và nhược điểm của DevOps**

**Ưu điểm:**

DevOps là một cách tiếp cận để phát triển và triển khai nhanh các ứng dụng.

Phản ứng nhanh hơn với những thay đổi của thị trường để cải thiện tăng trưởng kinh doanh.

Nâng cao lợi nhuận kinh doanh bằng cách giảm thời gian giao phần mềm và chi phí vận chuyển.

Hỗ trợ xóa quy trình mô tả, giúp rõ ràng về quá trình phát triển và phân phối sản phẩm.

DevOps cải thiện trải nghiệm và sự hài lòng của khách hàng.

Giúp đơn giản hóa việc cộng tác và đặt tất cả các công cụ trên đám mây để khách hàng truy cập.

DevOps có nghĩa là trách nhiệm tập thể, giúp tăng năng suất và gắn kết nhóm tốt hơn.

**Nhược điểm:**

Các developer chuyên nghiệp hoặc chuyên gia của DevOps ít có sẵn hơn.

Phát triển với DevOps rất tốn kém.

Việc áp dụng công nghệ DevOps mới vào các ngành công nghiệp khó quản lý trong thời gian ngắn.

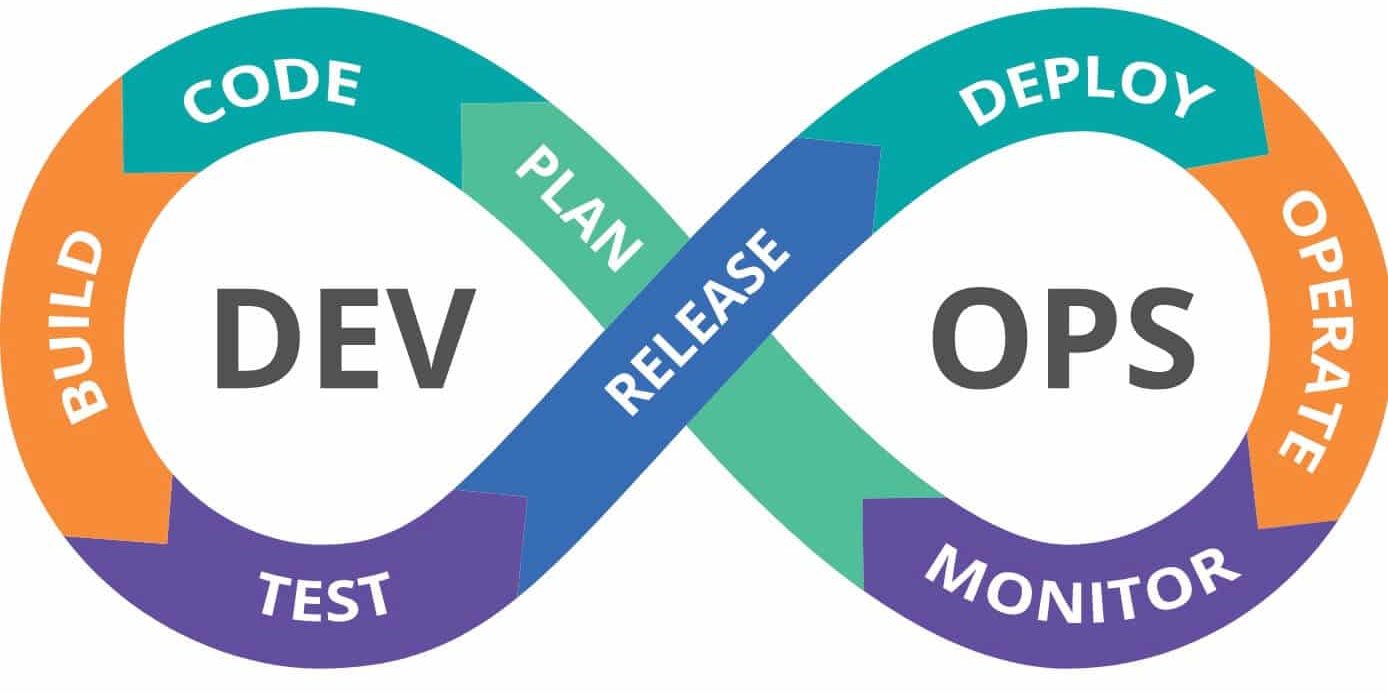
Thiếu kiến thức DevOps có thể là một vấn đề trong quá trình tích hợp liên tục các dự án tự động hóa.

Để học DevOps, cần phải có kiến thức cơ bản về Linux và ít nhất một ngôn ngữ Scripting.

1. **Quy trình DevOps**
   1. **Kiến trúc của DevOps**

Hoạt động bao gồm các quy trình quản trị, dịch vụ và hỗ trợ cho phần mềm. Khi cả quá trình phát triển và hoạt động được kết hợp với việc cộng tác, thì kiến trúc DevOps là giải pháp để khắc phục khoảng cách giữa các điều khoản triển khai và vận hành; do đó, giao hàng có thể nhanh hơn.

Kiến trúc DevOps được sử dụng cho các ứng dụng được lưu trữ trên nền tảng đám mây và các ứng dụng phân tán lớn. Phát triển Agile được sử dụng trong kiến trúc DevOps để tích hợp và phân phối có thể liền kề. Khi nhóm phát triển và vận hành làm việc tách biệt với nhau, thì việc thiết kế, kiểm tra và triển khai sẽ tốn nhiều thời gian. Và nếu các điều khoản không đồng bộ với nhau, thì có thể gây ra sự chậm trễ trong việc giao hàng. Vì vậy, DevOps cho phép các nhóm thay đổi những thiếu sót của họ và tăng năng suất.



*Hình 2.1. Thành phần của DevOps*

* 1. **Quy trình làm việc**



*Hình 2.2. Quy trình làm việc của DevOps*

**Quy trình:**

Quy trình làm việc DevOps cung cấp cái nhìn tổng quan trực quan về trình tự cung cấp đầu vào. Ngoài ra, quy trình cho biết một hành động nào được thực hiện và đầu ra được tạo ra cho một hoạt động.

Quy trình làm việc DevOps cho phép khả năng tách biệt và sắp xếp các công việc được người dùng yêu cầu hàng đầu. Ngoài ra, nó cung cấp khả năng phản chiếu quy trình lý tưởng của họ trong các công việc cấu hình.

**Nguyên tắc:**

Các nguyên tắc chính của DevOps là: phân phối liên tục, tự động hóa và phản ứng nhanh với phản hồi.

Trách nhiệm từ đầu đến cuối: Nhóm DevOps cần cung cấp hỗ trợ hiệu suất cho đến khi họ trở thành những người cuối đời. Nó nâng cao trách nhiệm và chất lượng của các sản phẩm được thiết kế.

Cải tiến liên tục: văn hóa DevOps tập trung vào cải tiến liên tục để giảm thiểu lãng phí. Nó liên tục tăng tốc độ phát triển của các sản phẩm hoặc dịch vụ được cung cấp.

Tự động hóa mọi thứ: Tự động hóa là một nguyên tắc thiết yếu của quy trình DevOps. Điều này là để phát triển phần mềm và cũng cho toàn bộ cảnh quan cơ sở hạ tầng.

Hành động trung tâm tùy chỉnh: Nhóm DevOps phải lấy khách hàng làm trung tâm để họ liên tục đầu tư vào các sản phẩm và dịch vụ.

Giám sát và kiểm tra mọi thứ: Nhóm DevOps cần có quy trình giám sát và kiểm tra mạnh mẽ.

Làm việc như một nhóm: Trong văn hóa DevOps, vai trò của người thiết kế, developer và người kiểm tra đã được xác định. Tất cả những gì họ cần làm là làm việc như một nhóm với sự cộng tác hoàn chỉnh.

Những nguyên tắc này đạt được thông qua một số thực hành DevOps, bao gồm triển khai thường xuyên, tự động hóa QA, phân phối liên tục, xác thực ý tưởng càng sớm càng tốt và cộng tác trong nhóm.

Phương pháp thực hành

Tự cấu hình các dịch vụ

Xây dựng liên tục

Tích hợp liên tục

Giao hàng liên tục

Thử nghiệm gia tăng

Cấp phép tự động

Quản lý phát hành tự động

1. **Công cụ DevOps**
   1. **Phát triển Code** 
      1. **Git**

Công cụ mã nguồn mở miễn phí.

Hỗ trợ phát triển phân tán.

Hỗ trợ yêu cầu kéo.

Giúp chu kỳ phát hành nhanh.

Giúp phát triển mở rộng.

An toàn và hoàn thành nhiệm vụ nhanh chóng

* + 1. **Selenium**

Công cụ mã nguồn mở miễn phí.

Hỗ trợ phát triển phân tán.

Hỗ trợ yêu cầu kéo.

Giúp chu kỳ phát hành nhanh.

Giúp phát triển mở rộng.

An toàn và hoàn thành nhiệm vụ nhanh chóng

* 1. **Tích hợp** 
     1. **Jenkins**

Tăng quy mô tự động hóa.

Dễ dàng thiết lập và cấu hình thông qua giao diện web.

Khả năng phân phối các nhiệm vụ trên nhiều máy, do đó tăng tính đồng thời.

Hỗ trợ tích hợp liên tục và phân phối liên tục.

Cung cấp nhiều loại plugin để hỗ trợ việc xây dựng và thử nghiệm hầu như bất kỳ dự án nào.

Yêu cầu bảo trì tối thiểu và có công cụ GUI tích hợp để cập nhật dễ dàng.

* 1. **Triển khai** 
     1. **Docker**

Công cụ DevOps cao cấp cho phép xây dựng, vận chuyển và chạy các ứng dụng phân tán trên nhiều hệ thống.

Giúp tập hợp các ứng dụng nhanh chóng từ các thành phần và nó thường thích hợp để quản lý vùng chứa.

Hổ trợ cấu hình hệ thống thoải mái và nhanh chóng.

Giúp tăng năng suất.

Cung cấp các thùng chứa được sử dụng để chạy ứng dụng trong một môi trường cô lập.

Định tuyến yêu cầu đến cho các cổng đã xuất bản trên các nút có sẵn.

Lưu các bí mật.

* + 1. **Ansible**

Dễ sử dụng để triển khai các ứng dụng mã nguồn mở.

Giúp tránh sự phức tạp trong quá trình phát triển phần mềm.

Giúp loại bỏ các công việc lặp đi lặp lại.

Quản lý các triển khai phức tạp và tăng tốc quá trình phát triển.

* 1. **Giám sát** 
     1. **Nagios**

Cung cấp khả năng giám sát hoàn chỉnh hệ điều hành máy tính để bàn và máy chủ.

Phân tích mạng giúp xác định tắc nghẽn và tối ưu hóa việc sử dụng băng thông.

Cung cấp để hoàn thành việc giám sát các Tiện ích mở rộng quản lý Java.

1. **Giới thiệu Docker**

Docker là một nền tảng ảo hóa cấp cao (containerization platform) cho việc phát triển, triển khai và chạy ứng dụng. Nó cung cấp một cách để đóng gói ứng dụng và tất cả các phụ thuộc của chúng vào một container đơn giản, di động và có thể chạy trên mọi nơi có Docker đã được cài đặt

* 1. **Containerization**

Docker sử dụng khái niệm "container" để đóng gói và triển khai ứng dụng. Mỗi container chứa tất cả những thứ cần thiết để chạy một ứng dụng cụ thể, bao gồm mã nguồn, thư viện, biến môi trường và cài đặt hệ điều hành

* 1. **Độc Lập Tài Nguyên**

Containers chia sẻ kernel của hệ điều hành với host, nhưng có các tài nguyên hệ thống cô lập như CPU, bộ nhớ, mạng. Điều này giúp tạo ra môi trường độc lập, dễ di động và không gian tài nguyên cô lập

* 1. **Di Động và Tương Thích**

Containers có thể chạy trên bất kỳ máy tính hoặc máy chủ nào có Docker, đảm bảo tính di động và tương thích giữa môi trường phát triển và triển khai

* 1. **Docker Image**

Docker Image là một bản chụp của một container và bao gồm tất cả các thứ cần thiết để chạy một ứng dụng cụ thể. Images được chia sẻ và quản lý thông qua Docker Registry

* 1. **Dockerfile**

Dockerfile là một tập tin cấu hình để xây dựng Docker Image. Nó mô tả các bước để tạo ra một image cùng với các cài đặt và cấu hình

* 1. **Docker Compose**

Docker Compose là một công cụ cho phép định nghĩa và quản lý nhiều container, giúp triển khai ứng dụng đa-container một cách dễ dàng

* 1. **Docker Swarm và Kubernetes**

Docker Swarm và Kubernetes là hai giải pháp cho quản lý và triển khai các container trong môi trường sản xuất, cung cấp khả năng mở rộng và quản lý các ứng dụng phức tạp

1. **Images & Containers**

Trong Docker, "Images" và "Containers" là hai khái niệm quan trọng giúp quản lý và triển khai ứng dụng.

* 1. **Docker Images**

**Khái Niệm:**

Một Docker Image là một bản chụp của một container. Nó bao gồm tất cả các thông tin cần thiết để chạy một ứng dụng cụ thể, bao gồm cả mã nguồn, thư viện, biến môi trường, và cài đặt hệ điều hành

**Tạo Docker Image:**

Docker Image thường được xây dựng từ một tập tin cấu hình gọi là Dockerfile. Dockerfile mô tả các bước để tạo ra một image. Command docker build sẽ sử dụng Dockerfile để xây dựng image

**Ví dụ:** *FROM node:14*

*WORKDIR /app*

*COPY . .*

*RUN npm install*

*CMD ["npm", "start"]*

**Quản Lý Docker Images:**

Docker Images được quản lý thông qua Docker Registry, nơi lưu trữ và chia sẻ images. Docker Hub là một Docker Registry công cộng phổ biến

*# Pull image từ Docker Hub*

*docker pull image\_name*

*# Push image lên Docker Hub*

*docker push image\_name*

* 1. **Docker Containers**

**Khái Niệm:**

Một Docker Container là một phiên bản thực hiện của một Docker Image. Nó là một thực thể chạy của một ứng dụng cụ thể với môi trường cô lập và tài nguyên độc lập

**Tạo Docker Container:**

Command docker run được sử dụng để tạo và chạy một container từ một Docker Image. Containers có thể chia sẻ một kernel với host nhưng có các namespace và cgroups cô lập

*# Tạo và chạy container từ image*

*docker run -d -p 8080:80 --name my\_container my\_image*

**Quản Lý Docker Containers:**

Command docker ps để liệt kê các container đang chạy. Command docker stop, docker start, và docker restart để quản lý trạng thái của container

*# Liệt kê các container đang chạy*

*docker ps*

*# Stop container*

*docker stop container\_id*

*# Start container*

*docker start container\_id*

1. **Data & Volumes**

Trong Docker, "Data" và "Volumes" là hai khái niệm quan trọng liên quan đến lưu trữ và quản lý dữ liệu của các container

* 1. **Docker Data**

**Khái Niệm:**

Docker Data thường đề cập đến dữ liệu được tạo và sử dụng bởi các container trong quá trình chạy. Điều này có thể bao gồm các file, thư mục, cơ sở dữ liệu, logs, và bất kỳ dữ liệu nào mà container cần để thực hiện công việc của mình.

**Tính Chất:**

Dữ liệu trong Docker Data có thể chỉ là tạm thời và không bền vững nếu container bị xóa.

Để giữ lại dữ liệu, các kỹ thuật như mounting volumes thường được ưa chuộng

* 1. **Docker Volumes**

**Khái Niệm:**

Docker Volumes là một cách để lưu trữ và duy trì dữ liệu của container qua các lần chạy, giúp giải quyết vấn đề của Docker Data về tính bền vững và di động

**Tính Chất:**

Volumes tạo ra một không gian lưu trữ riêng biệt nằm bên ngoài container, không bị ảnh hưởng bởi các thay đổi trong container.

Dữ liệu trong volumes được duy trì giữa các lần chạy của container, và volumes có thể chia sẻ giữa nhiều container.

Các volumes thường được sử dụng cho việc lưu trữ dữ liệu cơ sở dữ liệu, logs, và các tệp dữ liệu quan trọng khác.

**Các Thao Tác với Volumes:**

Tạo một Volume:

*docker volume create my\_volume*

Gắn Volume với Container khi chạy:

*docker run -v my\_volume:/path/in/container my\_image*

1. **Networking**

Networking trong Docker là quá trình quản lý và kết nối giữa các container và các mạng khác nhau. Docker cung cấp nhiều tùy chọn networking để cho phép các container giao tiếp với nhau và với mạng ngoại vi.

* 1. **Bridge Network**

**Khái Niệm:**

Mặc định, Docker tạo ra một bridge network khi cài đặt Docker. Mỗi container sẽ kết nối vào bridge network một cách tự động và có thể giao tiếp với các container khác trong cùng mạng.

**Tạo Container với Bridge Network:**

*docker run -d --name my\_container my\_image*

* 1. **Host Network**

**Khái Niệm**:

Container sử dụng mạng của host, không có lớp mạng cô lập cho container.

**Tạo Container với Host Network:**

*docker run -d --network host --name my\_container my\_image*

* 1. **Overlay Network**

**Khái Niệm:**

Overlay network cho phép kết nối các container trên nhiều host khác nhau, thích hợp cho các ứng dụng có nhiều container phân tán

**Tạo Overlay Network**

*docker network create -d overlay my\_overlay\_network*

* 1. **Custom Bridge Network**

**Khái Niệm:**

Có thể tạo ra các bridge network tùy chỉnh với các đặc tính và cài đặt riêng

**Tạo Custom Bridge Network:**

*docker network create --driver bridge --subnet=172.18.0.0/16 --gateway=172.18.0.1 my\_custom\_bridge*

* 1. **Macvlan Network**

**Khái Niệm:**

Macvlan network cho phép mỗi container có một địa chỉ MAC và IP thật từ mạng ngoại vi.

**Tạo Macvlan Network:**

*docker network create -d macvlan --subnet=192.168.1.0/24 --gateway=192.168.1.1 -o parent=eth0 my\_macvlan\_network*

* 1. **Linking Containers**

**Khái Niệm:**

Có thể kết nối các container với nhau thông qua "linking", cho phép chúng giao tiếp thông qua tên container

**Tạo Container với Linking:**

*docker run -d --name my\_webapp my\_webapp\_image*

*docker run -d --link my\_webapp:web my\_frontend\_image*

* 1. **Exposed Ports**

**Khái Niệm:**

Expose các cổng từ container để có thể truy cập từ mạng ngoại vi.

**Tạo Container với Exposed Port:**

*docker run -d -p 8080:80 --name my\_container my\_image*

1. **Client-Server Application**

Ứng dụng Client-Server trong Docker thường bao gồm việc triển khai và quản lý một phần của ứng dụng trên một hoặc nhiều container (Server) và một phần khác chạy trên máy tính của người dùng hoặc trình duyệt (Client)

* 1. **Xây Dựng Docker Image cho Server**

**Tạo Dockerfile cho Server:**

*# Dockerfile cho Server*

*FROM node:14*

*WORKDIR /app/server*

*COPY package\*.json ./*

*RUN npm install*

*COPY . .*

*EXPOSE 3000*

*CMD ["npm", "start"]*

**Build Docker Image cho Server:**

*docker build -t my-server-image .*

**Chạy Container cho Server:**

*docker run -d -p 3000:3000 --name my-server-container my-server-image*

* 1. **Xây Dựng Docker Image cho Client**

**Tạo Dockerfile cho Client:**

*# Dockerfile cho Client*

*FROM node:14*

*WORKDIR /app/client*

*COPY package\*.json ./*

*RUN npm install*

*COPY . .*

*EXPOSE 8080*

*CMD ["npm", "start"]*

**Build Docker Image cho Client:**

*docker build -t my-client-image .*

* 1. **Tạo Docker Network cho Giao Tiếp giữa Client và Server**

*docker network create my-network*

* 1. **Chạy Container cho Client và Kết Nối với Server**

*docker run -d -p 8080:8080 --name my-client-container --network my-network my-client-image*

* 1. **Cấu Hình Client để Kết Nối với Server**

Trong code của ứng dụng Client, đảm bảo rằng địa chỉ của Server được cấu hình đúng. Đối với ứng dụng web, có thể sử dụng biến môi trường hoặc tùy chọn cấu hình để xác định địa chỉ và cổng của Server.

*// Ví dụ trong ứng dụng React*

*const API\_SERVER\_URL = process.env.REACT\_APP\_API\_SERVER\_URL || "http://localhost:3000";*

1. **Các bài thực hành**
   1. **Bài tập thực hành 1 (Lệnh)**

D:\CongNgheMoi>md docker

D:\CongNgheMoi>cd docker

D:\CongNgheMoi\docker>docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

D:\CongNgheMoi\docker>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

D:\CongNgheMoi\docker>docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

…..v.v

* 1. **Bài tập thực hành 2 (Hello)**

D:\CongNgheMoi\docker\docker\hello>npm install

added 58 packages, and audited 59 packages in 939ms

8 packages are looking for funding

run `npm fund` for details

found 0 vulnerabilities

D:\CongNgheMoi\docker\docker\hello>node app.mjs

Error [ERR\_HTTP\_HEADERS\_SENT]: Cannot set headers after they are sent to the client

at new NodeError (node:internal/errors:399:5)

at ServerResponse.setHeader (node:\_http\_outgoing:663:11)

at ServerResponse.header (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\response.js:794:10)

at ServerResponse.contentType (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\response.js:624:15)

at ServerResponse.send (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\response.js:149:14)

at file:///D:/CongNgheMoi/docker/docker/hello/app.mjs:9:7

at Layer.handle [as handle\_request] (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\layer.js:95:5)

at next (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\route.js:144:13)

at Route.dispatch (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\route.js:114:3)

at Layer.handle [as handle\_request] (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\layer.js:95:5)

Error [ERR\_HTTP\_HEADERS\_SENT]: Cannot set headers after they are sent to the client

at new NodeError (node:internal/errors:399:5)

at ServerResponse.setHeader (node:\_http\_outgoing:663:11)

at ServerResponse.header (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\response.js:794:10)

at ServerResponse.contentType (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\response.js:624:15)

at ServerResponse.send (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\response.js:149:14)

at file:///D:/CongNgheMoi/docker/docker/hello/app.mjs:9:7

at Layer.handle [as handle\_request] (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\layer.js:95:5)

at next (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\route.js:144:13)

at Route.dispatch (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\route.js:114:3)

at Layer.handle [as handle\_request] (D:\CongNgheMoi\docker\docker\hello\node\_modules\express\lib\router\layer.js:95:5)

^C

D:\CongNgheMoi\docker\docker\hello>node app.mjs

……v.v

* 1. **Bài tập thực hành 3 (Goal)**

D:\CongNgheMoi>cd D:\CongNgheMoi\docker goal

D:\CongNgheMoi\docker\goal>md public

D:\CongNgheMoi\docker\goal>type nul > server.js

D:\CongNgheMoi\docker\goal>type nul > package.json

D:\CongNgheMoi\docker\goal>cd public

D:\CongNgheMoi\docker\goal\public>type nul > styles.css

D:\CongNgheMoi\docker\goal>npm install

added 75 packages, and audited 76 packages in 3s

8 packages are looking for funding

run `npm fund` for details

2 high severity vulnerabilities

To address all issues, run:

npm audit fix --force

Run `npm audit` for details.

D:\CongNgheMoi\docker\goal>node server.js

^C

……v.v

* 1. **Bài tập thực hành 4 (NotePython)**

D:\CongNgheMoi>cd docker

D:\CongNgheMoi\docker>md randompy

D:\CongNgheMoi\docker>cd randompy

D:\CongNgheMoi\docker\randompy>type nul > rng.py

D:\CongNgheMoi\docker\randompy>type nul > Dockerfile

D:\CongNgheMoi\docker\randompy>docker build -t random-image .

[+] Building 116.3s (9/9) FINISHED docker:default

=> [internal] load .dockerignore 0.2s

=> => transferring context: 2B 0.1s

=> [internal] load build definition from Dockerfile 0.4s

=> => transferring dockerfile: 107B 0.1s

=> [internal] load metadata for docker.io/library/python:latest 4.2s

=> [auth] library/python:pull token for registry-1.docker.io 0.0s

=> [1/3] FROM docker.io/library/python@sha256:3c51707dc38ba1332d1a71b3f2613357e3b9aaaf010df0778a715f84981d5e 111.3s

=> => resolve docker.io/library/python@sha256:3c51707dc38ba1332d1a71b3f2613357e3b9aaaf010df0778a715f84981d5e61 0.0s

=> => sha256:b94d01b4929562a35cdcfe80ca0d5fca222c3fdedf224af9a079dc23157e1363 7.53kB / 7.53kB 0.0s

=> => sha256:b47a222d28fa95680198398973d0a29b82a968f03e7ef361cc8ded562e4d84a3 24.03MB / 24.03MB 17.4s

=> => sha256:3c51707dc38ba1332d1a71b3f2613357e3b9aaaf010df0778a715f84981d5e61 2.14kB / 2.14kB 0.0s

=> => sha256:88880bc85b0e3342ff416c796df7ad9079b2805f92a6ebfc5c84ac582fb25de9 2.01kB / 2.01kB 0.0s

=> => sha256:167b8a53ca4504bc6aa3182e336fa96f4ef76875d158c1933d3e2fa19c57e0c3 49.56MB / 49.56MB 25.9s

=> => sha256:debce5f9f3a9709885f7f2ad3cf41f036a3b57b406b27ba3a883928315787042 64.11MB / 64.11MB 51.2s

=> => sha256:1d7ca7cd2e066ae77ac6284a9d027f72a31a02a18bfc2a249ef2e7b01074338b 211.04MB / 211.04MB 92.0s

=> => sha256:ff3119008f58beef8f336fa833707b0fe914db94ca6b7bb55abe3e1bf2b1ad56 6.39MB / 6.39MB 31.7s

=> => extracting sha256:167b8a53ca4504bc6aa3182e336fa96f4ef76875d158c1933d3e2fa19c57e0c3 5.7s

=> => sha256:e8095daf0d452128d830cb00924dbbcbbe023872853eff659453e3b17708f4d1 19.78MB / 19.78MB 51.3s

=> => extracting sha256:b47a222d28fa95680198398973d0a29b82a968f03e7ef361cc8ded562e4d84a3 1.4s

=> => extracting sha256:debce5f9f3a9709885f7f2ad3cf41f036a3b57b406b27ba3a883928315787042 8.4s

=> => sha256:b4b0db24b9acdf160c05860a6c98ba6047b35d1697f3915ad2a992a916cf5e99 244B / 244B 52.2s

=> => sha256:523ae3638d56dc770370a1380545d72c5eae2b76ab31abdb7d76e9cec293dc7c 3.11MB / 3.11MB 55.1s

=> => extracting sha256:1d7ca7cd2e066ae77ac6284a9d027f72a31a02a18bfc2a249ef2e7b01074338b 16.0s

=> => extracting sha256:ff3119008f58beef8f336fa833707b0fe914db94ca6b7bb55abe3e1bf2b1ad56 0.7s

=> => extracting sha256:e8095daf0d452128d830cb00924dbbcbbe023872853eff659453e3b17708f4d1 1.3s

=> => extracting sha256:b4b0db24b9acdf160c05860a6c98ba6047b35d1697f3915ad2a992a916cf5e99 0.0s

=> => extracting sha256:523ae3638d56dc770370a1380545d72c5eae2b76ab31abdb7d76e9cec293dc7c 0.7s

=> [internal] load build context 0.0s

=> => transferring context: 443B 0.0s

=> [2/3] WORKDIR /app 0.2s

=> [3/3] COPY . /app 0.0s

=> exporting to image 0.1s

=> => exporting layers 0.0s

=> => writing image sha256:6561346da42c5fd49a15a77c6084f0dbb1b9b1cc03fd3af7f2585be4f3281629 0.0s

=> => naming to docker.io/library/random-image 0.0s

What's Next?

View summary of image vulnerabilities and recommendations → docker scout quickview

…….v.v

* 1. **Bài tập thực hành 5 (DockerHub)**

D:\CongNgheMoi\docker\randompy>docker build -t random-image .

[+] Building 9.1s (9/9) FINISHED docker:default

=> [internal] load build definition from Dockerfile 0.1s

=> => transferring dockerfile: 107B 0.0s

=> [internal] load .dockerignore 0.1s

=> => transferring context: 2B 0.0s

=> [internal] load metadata for docker.io/library/python:latest 8.8s

=> [auth] library/python:pull token for registry-1.docker.io 0.0s

=> [1/3] FROM docker.io/library/python@sha256:2e376990a11f1c1e03796d08db0e99c36eadb4bb649 0.1s

=> => resolve docker.io/library/python@sha256:2e376990a11f1c1e03796d08db0e99c36eadb4bb649 0.1s

=> [internal] load build context 0.0s

=> => transferring context: 57B 0.0s

=> CACHED [2/3] WORKDIR /app 0.0s

=> CACHED [3/3] COPY . /app 0.0s

=> exporting to image 0.0s

=> => exporting layers 0.0s

=> => writing image sha256:6561346da42c5fd49a15a77c6084f0dbb1b9b1cc03fd3af7f2585be4f32816 0.0s

=> => naming to docker.io/library/random-image 0.0s

What's Next?

View summary of image vulnerabilities and recommendations → docker scout quickview

D:\CongNgheMoi\docker\randompy>docker run --name random-app -it random-image

Please enter the min number: 5

Please enter the max number: 50

26

……..v.v

* 1. **Bài tập thực hành 6 (Volumn)**

D:\CongNgheMoi\docker>md feedback

D:\CongNgheMoi\docker>cd feedback

D:\CongNgheMoi\docker\feedback>md feedback

D:\CongNgheMoi\docker\feedback>md pages

D:\CongNgheMoi\docker\feedback>md public

D:\CongNgheMoi\docker\feedback>md temp

D:\CongNgheMoi\docker\feedback>type nul > server.js

D:\CongNgheMoi\docker\feedback>type nul > package.json

D:\CongNgheMoi\docker\feedback>type nul > Dockerfile

D:\CongNgheMoi\docker\feedback>type nul > Dockerfile

D:\CongNgheMoi\docker\feedback>type nul > Dockerfile

D:\CongNgheMoi\docker\feedback>cd pages

D:\CongNgheMoi\docker\feedback\pages>type nul > exists.html

D:\CongNgheMoi\docker\feedback\pages>type nul > feedback.html

D:\CongNgheMoi\docker\feedback\pages>cd ..

D:\CongNgheMoi\docker\feedback>cd public

D:\CongNgheMoi\docker\feedback\public>type nul > styles.css

D:\CongNgheMoi\docker\feedback>npm install

added 172 packages, and audited 173 packages in 8s

22 packages are looking for funding

run `npm fund` for details

5 moderate severity vulnerabilities

To address all issues, run:

npm audit fix --force

Run `npm audit` for details.

D:\CongNgheMoi\docker\feedback>npm start

> data-volume-example@1.0.0 start

> nodemon server.js

[nodemon] 2.0.4

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): \*.\*

[nodemon] watching extensions: js,mjs,json

[nodemon] starting `node server.js`

Terminate batch job (Y/N)? y

D:\CongNgheMoi\docker>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

D:\CongNgheMoi\docker>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

D:\CongNgheMoi\docker>docker images

……v.v

* 1. **Bài tập thực hành 7 (Network)**

D:\CongNgheMoi\docker\network>npm install

npm WARN deprecated axios@0.20.0: Critical security vulnerability fixed in v0.21.1. For more information, see https://github.com/axios/axios/pull/3410

added 95 packages, and audited 96 packages in 7s

10 packages are looking for funding

run `npm fund` for details

1 high severity vulnerability

To address all issues (including breaking changes), run:

npm audit fix --force

Run `npm audit` for details.

D:\CongNgheMoi\docker\network>node app.js

(node:6528) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.

(Use `node --trace-warnings ...` to show where the warning was created)

^C

D:\CongNgheMoi\docker\network>docker build -t network-image .

[+] Building 6.4s (10/10) FINISHED docker:default

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 157B 0.0s

=> [internal] load metadata for docker.io/library/node:latest 2.7s

=> [1/5] FROM docker.io/library/node@sha256:6b3f9aa7eefa8d4c93d43914e78aa2bfea9a12808b0059e5da78854dfa8b8768 0.0s

=> [internal] load build context 3.0s

=> => transferring context: 13.53MB 2.9s

=> CACHED [2/5] WORKDIR /app 0.0s

=> CACHED [3/5] COPY package.json . 0.0s

=> CACHED [4/5] RUN npm install 0.0s

=> [5/5] COPY . . 0.4s

=> exporting to image 0.2s

=> => exporting layers 0.2s

=> => writing image sha256:4da0a1ac845b6dbb9cac9e9f27e7b32bc49201b548604ec0ff6fd617b7705bc7 0.0s

=> => naming to docker.io/library/network-image 0.0s

What's Next?

View summary of image vulnerabilities and recommendations → docker scout quickview

D:\CongNgheMoi\docker\network>node app.js

(node:6528) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.

(Use `node --trace-warnings ...` to show where the warning was created)

^C

……..v.v

* 1. **Bài tập thực hành 8 (Client-Server)**

D:\CongNgheMoi\docker\network>npm install

npm WARN deprecated axios@0.20.0: Critical security vulnerability fixed in v0.21.1. For more information, see https://github.com/axios/axios/pull/3410

added 95 packages, and audited 96 packages in 7s

10 packages are looking for funding

run `npm fund` for details

1 high severity vulnerability

To address all issues (including breaking changes), run:

npm audit fix --force

Run `npm audit` for details.

D:\CongNgheMoi\docker\network>node app.js

(node:6528) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.

(Use `node --trace-warnings ...` to show where the warning was created)

^C

D:\CongNgheMoi\docker\network>docker build -t network-image .

[+] Building 6.4s (10/10) FINISHED docker:default

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 157B 0.0s

=> [internal] load metadata for docker.io/library/node:latest 2.7s

=> [1/5] FROM docker.io/library/node@sha256:6b3f9aa7eefa8d4c93d43914e78aa2bfea9a12808b0059e5da78854dfa8b8768 0.0s

=> [internal] load build context 3.0s

=> => transferring context: 13.53MB 2.9s

=> CACHED [2/5] WORKDIR /app 0.0s

=> CACHED [3/5] COPY package.json . 0.0s

=> CACHED [4/5] RUN npm install 0.0s

=> [5/5] COPY . . 0.4s

=> exporting to image 0.2s

=> => exporting layers 0.2s

=> => writing image sha256:4da0a1ac845b6dbb9cac9e9f27e7b32bc49201b548604ec0ff6fd617b7705bc7 0.0s

=> => naming to docker.io/library/network-image 0.0s

What's Next?

View summary of image vulnerabilities and recommendations → docker scout quickview

D:\CongNgheMoi\docker\network>node app.js

(node:6528) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.

(Use `node --trace-warnings ...` to show where the warning was created)

^C

D:\CongNgheMoi\docker\network>docker build -t network-image .

[+] Building 6.4s (10/10) FINISHED docker:default

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 157B 0.0s

=> [internal] load metadata for docker.io/library/node:latest 2.7s

=> [1/5] FROM docker.io/library/node@sha256:6b3f9aa7eefa8d4c93d43914e78aa2bfea9a12808b0059e5da78854dfa8b8768 0.0s

=> [internal] load build context 3.0s

=> => transferring context: 13.53MB 2.9s

=> CACHED [2/5] WORKDIR /app 0.0s

=> CACHED [3/5] COPY package.json . 0.0s

=> CACHED [4/5] RUN npm install 0.0s

=> [5/5] COPY . . 0.4s

=> exporting to image 0.2s

=> => exporting layers 0.2s

=> => writing image sha256:4da0a1ac845b6dbb9cac9e9f27e7b32bc49201b548604ec0ff6fd617b7705bc7 0.0s

=> => naming to docker.io/library/network-image 0.0s

What's Next?

View summary of image vulnerabilities and recommendations → docker scout quickview

D:\CongNgheMoi\docker\network>docker build -t network-image .

[+] Building 1.9s (10/10) FINISHED docker:default

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 157B 0.0s

=> [internal] load metadata for docker.io/library/node:latest 1.3s