**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

-------------o0o------------



**TIỂU LUẬN MÔN**

**DỰ ÁN PHÁT TRIỂN PHẦN MỀM**

**ĐỀ TÀI:**

**DOCKER FOR DEVOPS AND DEVELOPER**

**Giảng viên hướng dẫn**: QUÁCH XUÂN TRƯỞNG

**Nhóm thực hiện** : Nhóm 9

**Lớp**: KTPM - K17B

**Tên thành viên:** Trần Minh Long

Dương Văn Định

Nguyễn Văn Nghiệp

Nguyễn Văn Hiếu

Trần Đức Long

*Thái Nguyên, tháng 11 năm 2021*



x**Mục lục**

[Chương 1:Giới thiệu chung 4](#_Toc96588289)

[1.1. tổng quản về DevOps và Developer 4](#_Toc96588290)

[1.1.2. Developer 4](#_Toc96588291)

[1.2. Docker cho devops và developer 4](#_Toc96588292)

[1.2.1. Docker cho devops 4](#_Toc96588293)

[1.2.2. Docker cho Developer 5](#_Toc96588294)

[Chương 2: Cơ sở lý thuyết 7](#_Toc96588295)

[2.1. Tổng quản về Docker 7](#_Toc96588296)

[2.1.1. Định nghĩa 7](#_Toc96588297)

[2.1.2. Lịch sử ra đời 8](#_Toc96588298)

[2.1.3. Cài đặt Docker trên Window 8](#_Toc96588299)

[2.2. Các thành phần cơ bản của Docker 12](#_Toc96588300)

[2.2.1. Docker Engine 12](#_Toc96588301)

[2.2.2. Docker Image 12](#_Toc96588302)

[2.2.3. Docker Container 13](#_Toc96588303)

[2.2.4. Dockerfile và Docker Compose 13](#_Toc96588304)

[2.2.5. Docker Hub 14](#_Toc96588305)

[2.3.Tính Chất của docker 15](#_Toc96588306)

[2.3.1. Ưu điểm 15](#_Toc96588307)

[2.3.2. Nhược điểm 16](#_Toc96588308)

[2.4. Quy trình triển khai docker 16](#_Toc96588309)

[2.4.1. Quy trình 16](#_Toc96588310)

[2.4.2.  Các câu lệnh thường dùng với Docker Container 17](#_Toc96588311)

[2.4.3.   Các câu lệnh thường dùng với Docker Image 18](#_Toc96588312)

[2.5. So sánh Docker với Kubernetes 18](#_Toc96588313)

[2.5.1. tổng quan về Kubernetes 18](#_Toc96588314)

[2.5.2. Tổng quan Docker Swarm 19](#_Toc96588315)

[2.5.3.So sánh Kubernetes vs Docker Swarm 19](#_Toc96588316)

# Chương 1:Giới thiệu chung

## 1.1. tổng quản về DevOps và Developer

**1.1.1 DevOps**

DevOps (kết hợp của cụm từ tiếng Anh "software DEVelopment" và "information technology OPerationS") là một thuật ngữ để chỉ một tập hợp các hành động trong đó nhấn mạnh sự hợp tác và trao đổi thông tin của các lập trình viên và chuyên viên tin học khi cùng làm việc để tự động hóa quá trình chuyển giao sản phẩm phần mềm và thay đổi kiến trúc hệ thống. Điều này nhằm thiết lập một nền văn hóa và môi trường nơi mà việc build, kiểm tra, và phát hành phần mềm có thể xảy ra nhanh chóng, thường xuyên, và đáng tin cậy hơn.

### 1.1.2. Developer

Developer (gọi tắt là Dev) là những người dùng ngôn ngữ lập trình để sáng tạo, xây dựng và bảo trì các chương trình, phần mềm, ứng dụng. Một số ngôn ngữ lập trình thường được sử dụng như Python, JavaScript, C, C++, C#,…

Developer thường đóng vai trò quan trọng trong quá trình tạo ra phần mềm. Bởi họ chính là người sử dụng thành thạo các ngôn ngữ lập trình để viết ra các đoạn mã lập trình. Hiện nay, rất nhiều bạn trẻ lựa chọn theo đuổi công việc này bởi những tiềm năng cũng như đãi ngộ mà nó mang lại.

Developer là một tên gọi khá chung cho các kỹ sư phần mềm. Trên thực tế, các developer thường có chuyên môn khác nhau. Và những công việc của họ sẽ xoay quanh chuyên môn đó. Cụ thể công việc của Developer là gì?

* Frontend Developer
* PHP Developer
* Backend Developer
* iOS Developer
* Android Developer

## 1.2. Docker cho devops và developer

### 1.2.1. Docker cho devops

Docker như một công cụ phù hợp hoàn toàn tốt trong hệ sinh thái DevOps. Nó được xây dựng cho các công ty phần mềm hiện đại đang bắt kịp với những thay đổi nhanh chóng trong công nghệ. Không thể bỏ qua Docker trong chuỗi công cụ DevOps; nó đã trở thành một công cụ thực tế và gần như không thể thay thế.

Những lợi ích của Docker với việc hỗ trợ DevOps là các use case và lợi thế mà nó mang lại cho quy trình phát triển phần mềm bằng cách container hóa các ứng dụng giúp phát triển dễ dàng và chu kỳ phát hành nhanh.

Docker có thể giải quyết hầu hết các vấn đề Dev và Ops, cho phép cả hai nhóm cộng tác hiệu quả và hoạt động hiệu quả.

Theo Báo cáo của RightScale 2019, Docker đã chiến thắng trong lĩnh vực container với sự tăng trưởng áp dụng YoY tuyệt vời.

Với Docker, doanh nghiệp có thể tạo ra các môi trường dev, thử nghiệm và sản xuất bất biến. Họ sẽ có mức độ kiểm soát cao đối với tất cả các thay đổi vì chúng được thực hiện bằng cách sử dụng Docker container và image bất biến. Họ luôn có thể quay lại phiên bản trước tại bất kỳ thời điểm nào nếu muốn.

Môi trường phát triển, thử nghiệm và sản xuất trở nên giống nhau hơn. Với Docker, đảm bảo rằng nếu một tính năng hoạt động trong môi trường phát triển, nó cũng sẽ hoạt động trong thử nghiệm và sản xuất.

### 1.2.2. Docker cho Developer

Docker giúp developer dễ dàng phát triển và triển khai các ứng dụng bên trong các môi trường ảo được container hóa. Điều này có nghĩa là các ứng dụng chạy giống nhau cho dù chúng ở đâu và chúng đang chạy trên máy nào.

Docker container có thể được triển khai cho bất kỳ máy nào mà không gặp vấn đề về sự tương thích, vì vậy hệ thống phần mềm sẽ giữ nguyên agnostic, giúp phần mềm sử dụng đơn giản hơn, ít phải phát triển hơn và dễ bảo trì và triển khai.

Một developervthường sẽ bắt đầu bằng cách truy cập Docker Hub, kho lưu trữ đám mây trực tuyến của các container Docker và kéo về một container chứa môi trường được cấu hình sẵn cho ngôn ngữ lập trình cụ thể của họ, như Ruby hoặc NodeJS, với tất cả các tệp và framework cần có để bắt đầu.

Docker là một trong những công cụ thực sự tuân theo tiêu chí "Xây dựng, Vận chuyển và Chạy".

Trên toàn thế giới và trên toàn ngành, rất nhiều công ty và viện nghiên cứu đang sử dụng Docker để tăng tốc các hoạt động phát triển của họ. PayPal hiện có hơn 700 ứng dụng và họ đã chuyển đổi tất cả chúng thành các ứng dụng dựa trên container. Họ chạy 150.000 container và điều này đã giúp họ tăng 50% năng suất dev.

MetLife, một ví dụ tuyệt vời khác, đã tiết kiệm rất lớn cho infrastructure của họ nhờ sử dụng ít hệ điều hành hơn nhưng quản lý được nhiều ứng dụng hơn. Điều này mang lại cho họ rất nhiều phần cứng, và tiết kiệm rất nhiều tiền cho infrastructure và giảm chi phí. Sau khi chuyển sang Docker, MetLife đã giảm 70% chi phí VM, ít hơn 67% CPU, giảm gấp 10 lần sử dụng CPU trung bình và giảm 66% chi phí. Đó là sức mạnh của Docker.

## 1.3. lý do lựa chọn Doker

**Thời gian “start” và “stop” cực ngắn:**

Lý do đầu tiên mà Docker đem lại đó chính là thời gian bắt đầu và dùng cực ngắn. Docker được so sánh với các máy ảo và cho thấy sự khác biệt của mình. Thời gian trong hai công đoạn này của Docker nhanh hơn, ấn tượng hơn hẳn.

**Tự do trong chọn hệ thống:**

Lý do thứ hai của Docker chính là lập trình viên có thể tự do chọn hệ thống. Người dùng có thể tiến hành khởi chạy container trong bất cứ hệ thống nào họ muốn. Đây chính là lợi ích cực kỳ độc đáo mà Docker đem lại.

**Tốc độ làm việc nhanh:**

Thời gian với dân lập trình chắc hẳn là điều rất quan trọng. Tốc độ làm việc của các nền tảng luôn được đặt lên hàng đầu. Thao tác build và loại bỏ được thực hiện bởi container nhanh hơn so với máy ảo. Chính vì thế, đến với Docker các bạn sẽ có tốc độ cũng như hiệu suất làm việc nhanh hơn. Đây chính là lời giải đáp cho câu hỏi tính năng nổi bật của docker là gì?

Đơn giản trong việc thiết lập môi trường: Các lập trình viên khi sử dụng Docker chỉ cần config 1 lần. Họ sẽ không cần cài đặt lại Dependencies sau đó nữa. Nếu có sự thay đổi mới về thành viên hay thiết bị, người dùng chỉ cần chia sẻ config mà thôi.

Hỗ trợ xóa: Lợi ích nổi bật cuối cùng của Docker chính là giúp cho word-space sạch sẽ hơn. Trong trường hợp lập trình viên cần xóa những mối trường có ảnh hưởng tới phần khác. Thì Docker hỗ trợ xóa nhanh chóng điều này.

## 1.4. Phạm vi của chủ đề

# Chương 2: Cơ sở lý thuyết

## 2.1. Tổng quản về Docker

### 2.1.1. Định nghĩa

Docker là một dự án mã nguồn mở giúp tự động triển khai các ứng dụng Linux và Windows vào trong các container ảo hóa. Docker cung cấp một lớp trừu tượng và tự động ảo hóa dựa trên Linux.



Docker là một công cụ giúp cho việc tạo ra và triển khai các container để phát triển, chạy ứng dụng được dễ dàng. Các container là môi trường, mà ở đó lập trình viên đưa vào các thành phần cần thiết để ứng dụng của họ chạy được, bằng cách đóng gói ứng dụng cùng với container như vậy, nó đảm bảo ứng dụng chạy được và giống nhau ở các máy khác nhau (Linux, Windows, MacOS, Windows Server …).

Docker có vẻ rất giống máy ảo (nhiều người từng tạo máy ảo với công cụ ảo hóa như Virtualbox, VMWare), nhưng có điểm khác với VM: thay vì tạo ra toàn bộ hệ thống (dù ảo hóa), Docker lại cho phép ứng dụng sử dụng nhân của hệ điều hành đang chạy Docker để chạy ứng dụng bằng cách bổ sung thêm các thành phần còn thiếu cung cấp bởi container. Cách này làm tăng hiệu suất và giảm kích thước ứng dụng.

Hiểu đơn giản hơn thì Docker cung cấp cho người sử dụng những service và những công cụ cần thiết để giúp người sử dụng có thể chạy các chương trình và đóng gói chúng ở trong tất cả môi trường khác nhau một cách nhanh chóng và đơn giản nhất.

Tóm lại, Docker là một công cụ giúp deploy dự án phần mềm một cách nhanh chóng trong các container. Các môi trường liên quan cần thiết để phần mềm hoạt động đều được Docker cung cấp đầy đủ. Triển khai dự án phần mềm một cách nhanh chóng và dễ dàng. Có rất nhiều lợi ích đáng để sử dụng Docker.

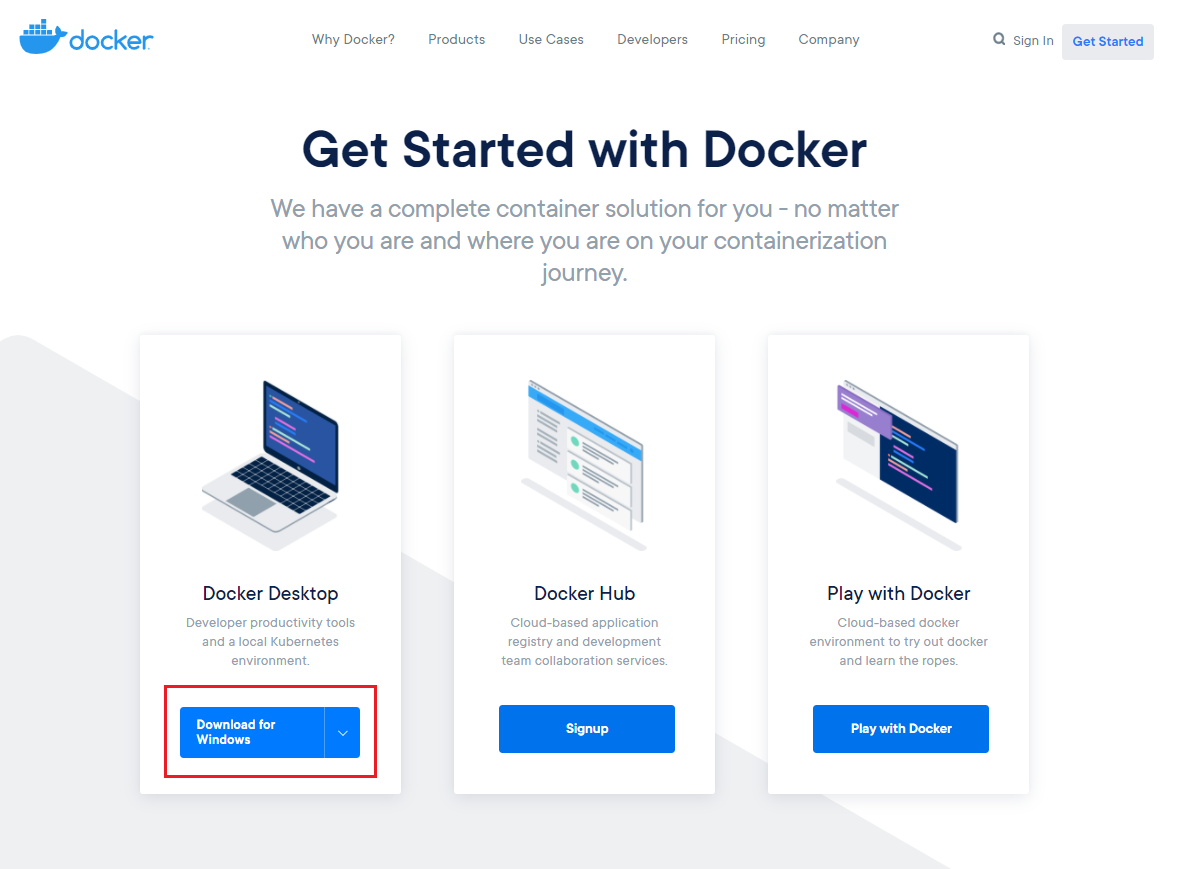
### 2.1.2. Lịch sử ra đời

Solomon Hykes bắt đầu tạo ra Docker khi làm việc ở Pháp, trong một dự án nội bộ của dotCloud, một công ty nền tảng như dịch vụ ban đầu có thêm sự đóng góp của các kỹ sư dotCloud là Andrea Luzzardi và Francois-Xavier Bourlet.[ Jeff Lindsay cũng đã tham gia như là một nhà cộng tác độc lập.

Docker được phát hành dạng mã nguồn mở trong tháng 3 năm 2013. Vào ngày 13 năm 2014, với phiên bản 0.9, Docker bỏ đi LXC và thay thế nó với thư viện của mình là libcontainer được viết bằng ngôn ngữ Go. Tới tháng 10 năm 2015, dự án Docker đã có hơn 25,600 sao trên GitHub (trở thành top 20 dự án có số sao cao nhất trên GitHub), có hơn 6,800 fork, và gần 1.100 lập trình viên tham gia đóng góp.

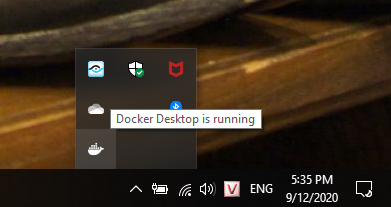
### 2.1.3. Cài đặt Docker trên Window

Tải và cài đặt Doker cho window tại <https://www.docker.com/get-started>

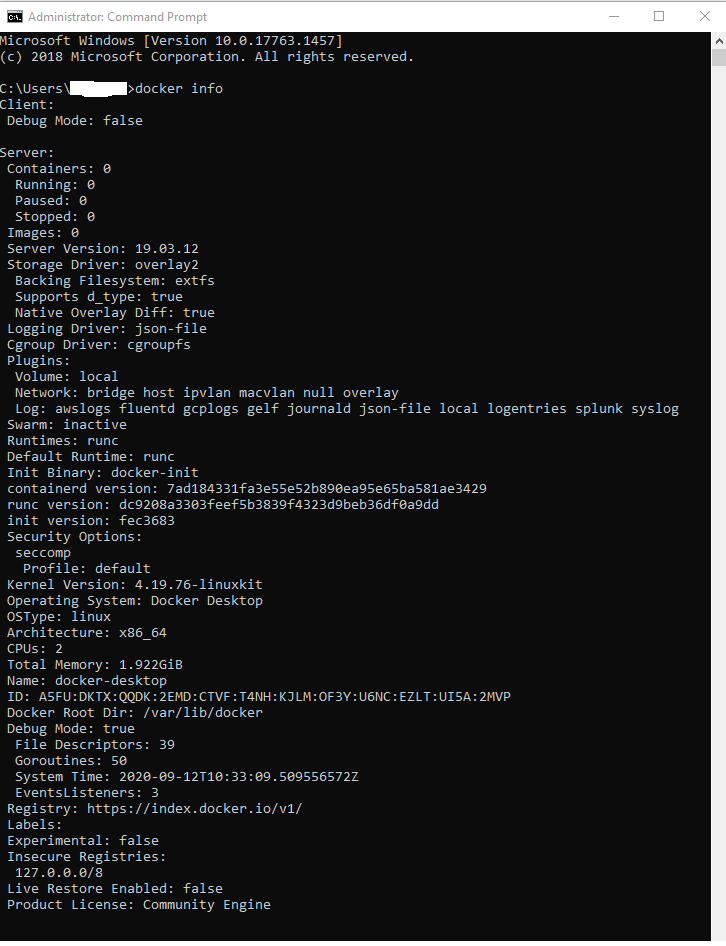


Sau khi đã tải về, các bạn tiến hành cài đặt thông qua file Docker Desktop Installer.exe vừa tải về. Như mình đã giới thiệu, để chạy được doker cần bật tính năng Hyper-V của windows và trong quá trình cài đặt docker nếu windows chưa được bật Hyper-V, sẽ có một checkbox hỏi xem có muốn bật Hyper-V luôn không, thì mình nên lựa chọn checkbox để bật Hyper-V luôn.

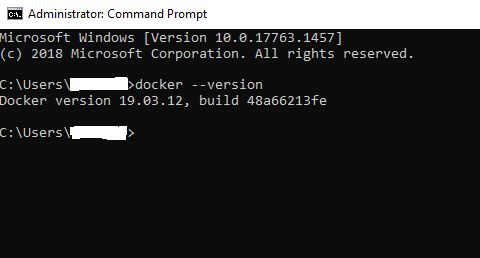
Sau khi cài đặt hoàn tất, cần khởi động lại máy để có thể chạy docker, sau khi khởi động chúng ta có thể thấy biểu tượng Docker ở Tray Icon:



Để kiểm tra thông tin chi tiết: chúng ta thực hiện lệnh cmd : >docker info

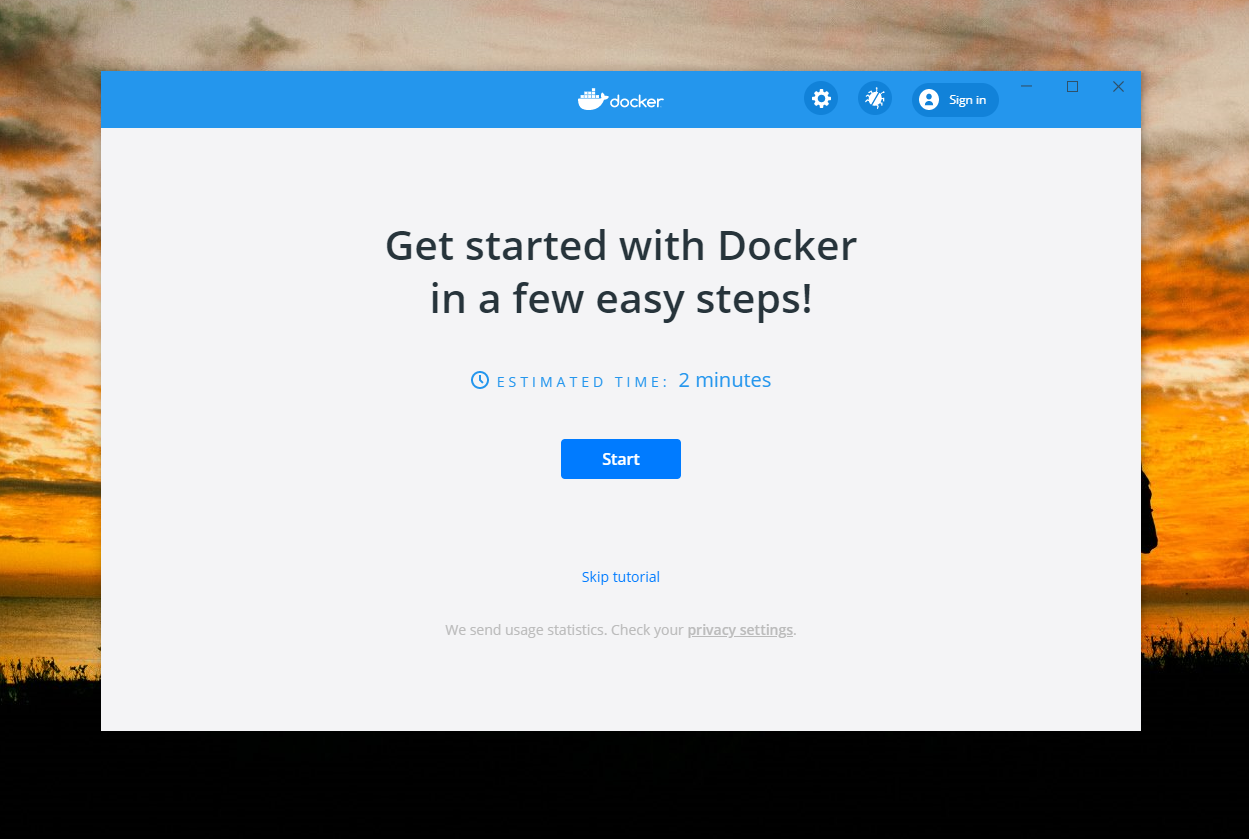


Để kiểm tra phiên bản của Docker chúng ta thực hiện lệnh cmd: >docker --version​



Khi quá trình khởi chạy hoàn tất, chúng ta có một màn hình giới thiệu khi mở Docker Desktop. Trong hướng dẫn đã có một số bài tập đơn giản để chúng ta xây dựng và làm quen với Docker.

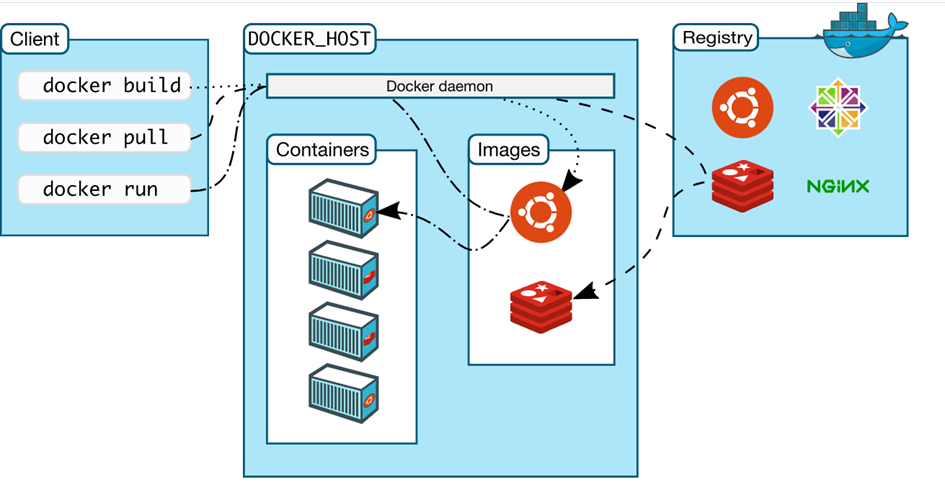
Khi quá trình khởi chạy hoàn tất, chúng ta có một màn hình giới thiệu khi mở Docker Desktop. Trong hướng dẫn đã có một số bài tập đơn giản để chúng ta xây dựng và làm quen với Docker.



## 2.2. Các thành phần cơ bản của Docker

### 2.2.1. Docker Engine

Docker Engine (Docker Daemon) là một thành phần quan trọng của Docker. Nó chính là thành phần chính giúp vận hành Docker. Có thể hiểu đây chính là cốt lõi của Docker.



Docker Engine được cài trên các máy chủ sử dụng Docker (Docker Host), nó chứa các đối tượng là thành phần con của docker (Container, Image) quản lý và vận hành chúng.

Docker Engine là công cụ Client - Server hỗ trợ công nghệ container để xử lý các nhiệm vụ và quy trình công việc liên quan đến việc xây dựng các ứng dụng dựa trên vùng chứa (container). Engine tạo ra một quy trình daemon phía máy chủ lưu trữ images, containers, networks và storage volumes. Daemon cũng cung cấp giao diện dòng lệnh phía máy khách (CLI) cho phép người dùng tương tác với daemon thông qua giao diện lập trình ứng dụng Docker.

### 2.2.2. Docker Image

Docker Image là một file có chứa tất cả các source code, libraries, dependencies, tools và các files khác cần thiết để ứng dụng có thể hoạt động.

Docker Image chỉ là các mẫu để từ đó build các Container tương ứng. Docker Image không cho phép deploy trực tiếp phần mềm lên chúng.

Image là là thành phần để đóng gói ứng dụng và các thành phần mà ứng dụng phụ thuộc để chạy. Image có thể lưu local hoặc lưu Registry

Hiểu theo cách đơn giản nhất, Docker Image giống như hình ảnh mô phòng của một môi trường thực thi với mục đích nhất định. Chúng chứa các source code, các thư viện để phục vụ cho việc chạy một phần mềm.

Ví dụ dễ hiểu về các Image: Image npm dùng chạy nodeJs, Image Java JDK dùng để chạy Java, Python . . .

### 2.2.3. Docker Container

Nếu như coi Docker Image là một class thì Docker Container là một instance của image. Container cung cấp môi trường thực cho ứng dụng hoạt động. Ở đây, người dùng có thể chạy các ứng dụng phần mềm.

Container nắm giữ toàn bộ các package cần thiết để khởi động và chạy ứng dụng. Các Container sẽ sử dụng chung tài nguyên của hệ thống nên rất nhẹ và giúp các thao tác kết nối, tương tác diễn ra nhanh chóng, tiện lợi hơn.

Container có công dụng giống hệt với tên của nó. Container như một “vùng chứa”. Source code phần mềm của bạn khi deploy sẽ chạy trực tiếp trong container này. Bạn có thể hình dung khi đó container như một server riêng, máy ảo riêng để chạy phần mềm.

Ưu điểm của Container là rất nhẹ và dễ dàng tạo, xóa. Đây chính là một trong những ưu điểm lớn nhất của docker container so với các công nghệ máy ảo khác.

### 2.2.4. Dockerfile và Docker Compose

***a, Docker file***

Dockerfile là thành phần của hệ sinh thái Docker, các Docker file khi build sẽ tạo thành Docker Image. Chính nhờ Dockerfile mà lập trình viên có thể tự xây dựng các Image, Container cần thiết cho project của bạn.

Dockerfile là một file dạng text không có đuôi, Dockerfile có cấu trúc theo quy chuẩn nhất định của Docker. Nội dung gồm tập hợp các câu lệnh cũng như phụ thuộc cần thiết để tạo ra Docker Image.

Ví dụ project của bạn cần một Image có cài cả hai môi trường NodeJS và Python thì trong Dockerfile bạn có gọi chúng vào là được. Tất cả việc còn lại Docker sẽ giúp bạn.

Dockerfile sinh ra giúp việc deploy thực nhanh hơn rất nhiều, khi triển khai phần mềm chỉ cần có dockerfile. Điều này giúp tạo Docker Image và Container nhanh hơn rất nhiều.

***b, Docker Compose***

Docker Compose là công cụ dùng để định nghĩa và run multi-container cho Docker application. Docker Compose, chúng ta sử dụng một file YAML để thiết lập các service cần thiết cho chương trình. Cuối cùng, với một câu lệnh, chúng ta sẽ tạo và khởi động tất cả các service từ các thiết lập đó.

Khi phát triển một chương trình, việc chạy một chương trình trong một môi trường cô lập và tương tác là rất cần thiết. Compose cho phép thiết lập và chạy tất cả các service cần thiết cho chương trình. Chỉ với một câu lệnh docker-compose up, các service đó sẽ được chạy với các container tương ứng

### 2.2.5. Docker Hub

Docker Hub là Registry, Container Image Library chỉnh thức của Docker, là một dịch vụ do Docker cung cấp, cho phép tìm kiếm và chia sẻ các Container Image.

Các tính năng chính của Docker Hub là:

**Repositories**: Push và pull container images.

**Teams & Organizations**: Quản lý quyền truy cập vào private repositories của container images.

**Official Images**: Pull sử dụng container images chất lượng cao của Docker.

**Publisher Images**: Pull và sử dụng container images được cung cấp bởi vendors khác.

**Builds**: Tự động tạo container images từ GitHub và Bitbucket. Push chúng lên Docker Hub.

**Webhooks**: Kích hoạt các actions sau khi push thành công một repository lên Docker Hub với các dịch vụ khác.

## 2.3.Tính Chất của docker

### 2.3.1. Ưu điểm

**Tính đồng nhất**: Đây cũng chính là ưu điểm nổi bật nhất khi sử dụng Docker. Trong trường hợp nhiều người cùng phát triển một dự án trong môi trường, việc sử dụng Docker sẽ giúp hạn chế được sự sai khác nhất định giữa thành viên.

**Tính nhất quán**: Với Docker, bạn có thể test container được dùng để phát triển bằng CI. Dễ dàng deploy container đã được test bằng CI lên server. Ngoài ra, bạn cũng có thể thực hiện scale container đã được deploy.

**Tính đóng gói**: Với Docker, bạn có thể ẩn môi trường bao gồm cả App vào Container, có thể test được Container đồng thời cũng có thể dễ dàng bỏ hạy tạo Container.

**Dễ dàng sử dụng:** Docker rất dễ cho mọi người sử dụng từ developers, systems admins, architects…v…v.. nó tận dụng lợi thế của container để build, test nhanh chóng. Có thể đóng gói ứng dụng trên laptop của họ và chạy trên public cloud, private cloud..v.v… “Build once, run anywhere”.

**Tốc độ:** Docker container rất nhẹ và nhanh, bạn có thể tạo và chạy docker container trong vài giây so sánh với VMs thì mỗi lần chạy VMs cần rất nhiều thời gian khởi động.

**Khả năng di động:** môi trường develop được dựng lên bằng docker có thể chuyển từ người này sang người khác mà không làm thay đổi cấu hình ở trong.

**Chia sẻ:** DockerHub là một “app store for docker images”. Trên DockerHub có hàng ngàn public images được tạo bởi cộng đồng. Dễ dàng tìm thấy những image mà bạn cần và chỉ cần pull về và sử dụng với một số sửa đổi nhỏ.

**Môi trường chạy và khả năng mở rộng:** Bạn có thể chia nhỏ những chức năng của ứng dụng thành các container riêng lẻ. Ví dụng Database chạy trên một container và Redis cache có thể chạy trên một container khác trong khi ứng dụng Node.js lại chạy trên một cái khác nữa. Với Docker, rất dễ để liên kết các container với nhau để tạo thành một ứng dụng, làm cho nó dễ dàng scale, update các thành phần độc lập với nhau.

### 2.3.2. Nhược điểm

* Nó không cung cấp một tùy chọn lưu trữ.
* Theo dõi tồi tệ.
* Không có lập trình lại tự động các nút không hoạt động.
* Các hành động phải được thực hiện trong CLI.
* Quản lý thủ công nhiều trường hợp.
* Bạn cần hỗ trợ các công cụ khác.
* Triển khai cụm thủ công phức tạp.
* Không hỗ trợ kiểm tra sức khỏe.
* Docker là một công ty hoạt động vì lợi nhuận và một số thành phần quan trọng của nó, chẳng hạn như Docker Engine và Docker Desktop, không phải là mã nguồn mở.

## 2.4. Quy trình triển khai docker

### 2.4.1. Quy trình

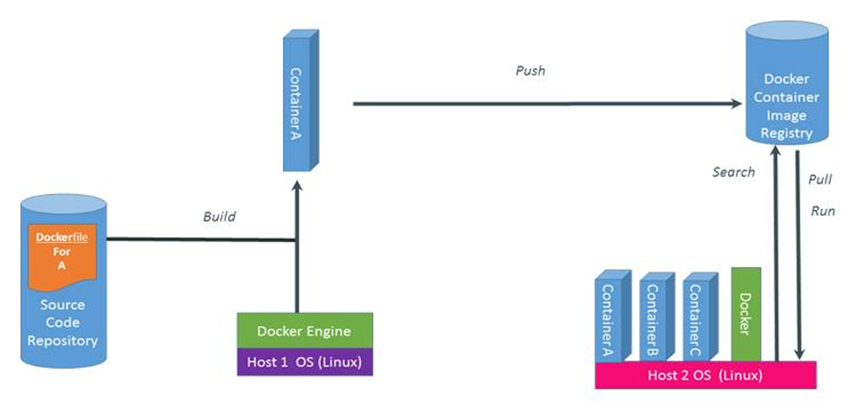
Docker được xây dựng dựa trên 3 thành phần chính:

* Docker daemon: daemon (server) nhận lệnh từ docker client thông qua CLI hoặc RestAPI
* Docker client: Có thể ở trên cùng một host hoặc khác host với docker daemen
* Docker registry (hub): 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. Một số dịch vụ Docker Registry phổ biến như : Azure Container Registry, Docker Hub...

Một hệ thống Docker thường sẽ được thực thi dựa theo 3 bước chính như sau:

**Bước 1: Build**

Đầu tiên, bạn cần tạo một dockerfile, trong dockerfile này thì nó chính là code của chúng ta. Nó sẽ được Build tại một máy tính đã được cài đặt sẵn Docker Engine. Sau khi đã build, thì ta sẽ có được Container, trong Container đã có chứa ứng dụng đi kèm bộ thư viện của chúng ta.



*Quy trình, các bước hoạt động của Docker là gì?*

**Bước 2: Push**

Sau khi đã có được Container thì chúng ta cần phải thực hiện Pusch Container này lên cloud rồi lưu nó tại đó.

**Bước 3: Pull và Run**

Nếu như một máy tính khác đang muốn sử dụng Container thì chúng ta cần buộc máy thực hiện việc Pull container này về máy. Tất nhiên, máy này cũng cần phải được cài Docker Engine, sau đó sẽ thực hiện Run Container.

### 2.4.2.  Các câu lệnh thường dùng với Docker Container

* Docker container commit: Tạo image mới từ những thay đổi của container.
* Docker container create: Tạo container mới.
* Docker container exec: Chạy các command khi container đang hoạt động.
* Docker container kill: Chấm dứt hoạt động của một hoặc nhiều container.
* Docker container pause: Tạm dừng tất cả tiến trình bên trong một hoặc nhiều container.
* Docker container run: Chạy các command trong một container mới.
* Docker container start: Chạy một container hoặc nhiều container đã dừng.
* Docker container rename: Đổi tên container.
* Docker container restart: Khởi động lại một hoặc nhiều container.

### 2.4.3.   Các câu lệnh thường dùng với Docker Image

* Docker images: Liệt kê các image
* Docker image build: Build image từ file Dockerfile.
* Docker image history: Hiện thị lịch sử của image.
* Docker image import: Import nội dung từ tarball để tạo ra filesystem của image.
* Docker image inspect: Hiển thị thông tin chi tiết của một hoặc nhiều image.
* Docker image load: Nạp image từ file \*.tar hoặc STDIN.
* Docker image prune: Xóa các image không sử dụng.
* Docker image pull: Pull một image hoặc repository từ Docker HUB đăng ký.
* Docker image push: Đẩy image, repository lên Docker HUB.
* Docker image save: Lưu một hoặc nhiều image vào file \*.tar.
* Docker image tag: Gắn tag cho TARGET IMAGE tương ứng với SOURCE IMAGE

## 2.5. So sánh Docker với Kubernetes

Khi hầu hết mọi người nói về “Kubernetes và Docker”, ý của họ thực sự là “Kubernetes và Docker Swarm”. Về sau, Docker đã tự xây dựng giải pháp cluster cho các container Docker, và lợi thế là được tích hợp chặt chẽ vào hệ sinh thái của Docker và sử dụng API của riêng nó. Giống như hầu hết các bộ lập lịch, Docker Swarm cung cấp cách quản lý một số lượng lớn các container trải rộng trên các cụm máy chủ. Hệ thống lọc và lập lịch của nó cho phép lựa chọn các node tối ưu trong một cụm để deploy các container.

### 2.5.1. tổng quan về Kubernetes

Kubernetes là nền tảng được xây dựng dựa trên nhiều năm kinh nghiệm của Google về việc chạy workload ở quy mô lớn trong quy trình production. Theo định nghĩa tại website của Kubernetes, “Kubernetes là một hệ thống nguồn mở (open-source system) để tự động hóa việc triển khai, thay đổi kích thước và quản lý các ứng dụng được container hoá”.

### 2.5.2. Tổng quan Docker Swarm

Docker Swarm là công cụ điều phối container của Docker, sử dụng Docker API tiêu chuẩn và networking nhằm giúp người dùng dễ dàng thích nghi với môi trường làm việc của các Docker container. Docker Swarm được thiết kế để hoạt động theo 04 nguyên tắc chủ đạo:

* Tính năng mạnh mẽ cùng thiết kế đơn giản, thân thiện với người dùng.
* Kiến trúc resilient zero single-point-of-failure
* Bảo mật mặc định với các certificate được tạo tự động
* Khả năng tương thích với các component (thành phần) hiện có

### 2.5.3.So sánh Kubernetes vs Docker Swarm

|  |  |  |
| --- | --- | --- |
|  | **Kubernetes** | **Docker** |
| Định nghĩa | một ứng dụng có thể được deploy bằng cách sử dụng kết hợp các pod, deployments và services (hoặc micro-services). | các ứng dụng có thể được deploy như một service (hoặc micro-service) trong một Swarm cluster. File YAML có thể được dùng để cụ thế hoá multi-container. Hơn nữa, Docker Compose có thể deploy ứng dụng. |
| Cài đặt và thiết lập | bước install được thực hiện thủ công và cần có kế hoạch cụ thể để Kubernetes hoạt động trơn tru. Hướng dẫn cài đặt thường được không thống nhất giữa các nhà cung cấp. Ngoài ra, cần nắm được cấu hình cluster như địa chỉ IP của một node hoặc nhiệm vụ của mỗi node. | Với Docker, chỉ cần một bộ công cụ tùy chọn để build theo môi trường và cấu hình. Docker Swarm cũng cung cấp tính linh hoạt bằng cách cho phép tất cả các node mới tham gia vào một cluster hiện có với tư cách là manager hoặc worker. |
| Yêu cầu | Kubernetes yêu cầu kiến thức về CLI (Command Line Interface) để chạy trên Docker. Cần hiểu về Docker CLI để điều hướng bên trong một cấu trúc, sau đó bổ sung infrastructure ngôn ngữ chung Kubernetes để chạy các program đó. | sử dụng ngôn ngữ chung để điều hướng trong một cấu trúc. Điều này cung cấp tính biến thiên và tốc độ cho công cụ này. |
| Giám sát | Kubernetes hỗ trợ nhiều phiên bản logging (ghi nhật ký) và giám sát khi các service được triển khai trong cluster | Docker Swarm được hỗ trợ để chỉ giám sát với các ứng dụng của bên thứ ba. Lời khuyên là nên sử dụng Docker với Reimann để giám sát, tuy nhiên vì Docker Swarm có API mở, nên việc kết nối với nhiều ứng dụng dễ dàng hơn. |
| Khả năng thay đổi quy mô | Kubernetes là một all-in-one framework cho các hệ thống phân tán. Đây là một hệ thống phức tạp vì nó cung cấp một bộ API thống nhất và đảm bảo mạnh mẽ về trạng thái cluster, làm chậm deployment và thay đổi quy mô container. | So với Kubernetes, Docker Swarm có thể deploy các container nhanh hơn; điều này cho phép thời gian phản ứng nhanh để thay đổi quy mô theo yêu cầu. |
| Tính sẵn sàng | Trong Kubernetes, tất cả các pod được phân phối giữa các node và điều này cung cấp tính sẵn sàng cao bằng cách chấp nhận lỗi ứng dụng. Hơn nữa, các load-balancing của Kubernetes phát hiện các pod không lành mạnh và loại bỏ chúng, điều này hỗ trợ tính sẵn sàng cao. | Docker Swarm cũng cung cấp tính sẵn sàng cao vì các service có thể được nhân bản trong các node Swarm. Trong Docker Swarm, các node quản lý Swarm chịu trách nhiệm cho toàn bộ cluster và quản lý tài nguyên của các node worker. |
| Networking | Kubernetes network có tính chất phẳng vì nó cho phép tất cả các pod giao tiếp với nhau. Trong Kubernetes, model yêu cầu 02 CIDR: 01 CIDR để yêu cầu các pod lấy địa chỉ IP, 01 CIDR là cho các service. | Trong Docker Swarm, một node tham gia một cluster tạo ra một overlay network các service bao trùm tất cả các host trong Swarm và một network cầu nối Docker duy nhất cho các container. Người dùng Docker Swarm có thể tùy chọn mã hóa data traffic (lưu lượng dữ liệu) container khi tự tạo overlay network. |