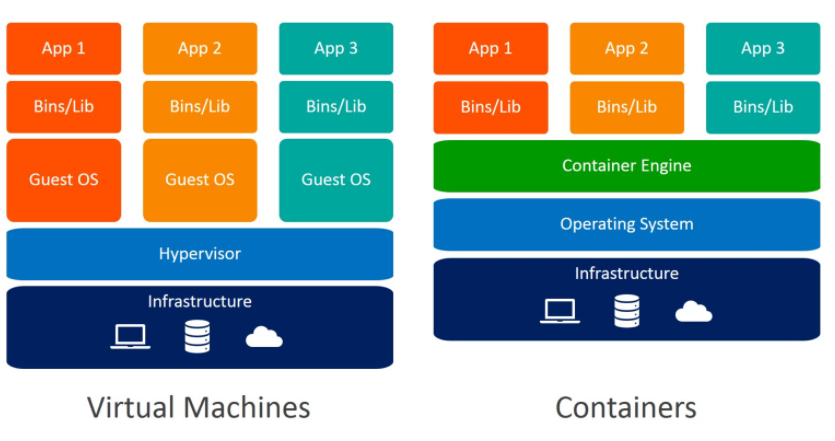
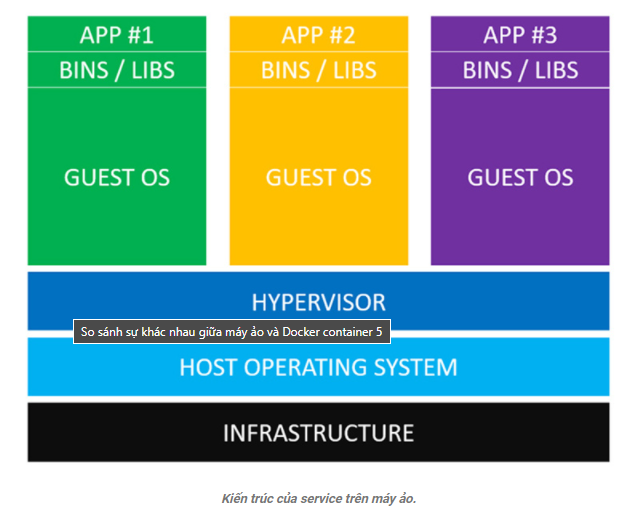
**So sánh docker với máy ảo**

****

**Kiến trúc của service hoạt động trên máy ảo**

Máy ảo (Virtual Machine – VM) là một mô phỏng của hệ thống máy tính. Nói một cách đơn giản, công nghệ này giúp người dùng có thể tạo ra nhiều “máy tính logic” trên một “máy tính vật lý”, dễ dàng quản lý vấn đề bảo mật. Kiến trúc của service khi hoạt động trên nền tảng máy ảo thông qua ảnh minh họa bên dưới:

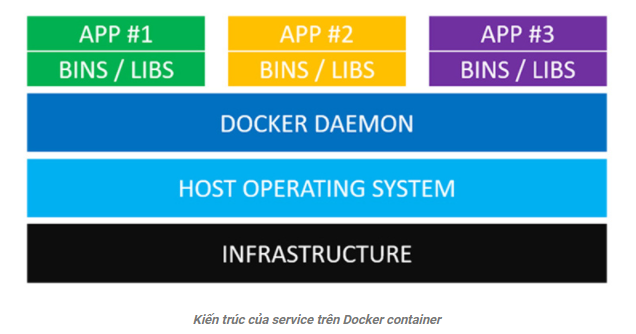


**Kiến trúc của service trên máy ảo.**

* **Infrastructure (hạ tầng):**có thể là laptop, một server (máy chủ) chuyên dụng (dedicated server) hoat động trong một trung tâm dữ liệu (Data Center) hoặc một server riêng ảo (Virtual Private Server – VPS) trên cloud như DigitalOcean, Amazon, GCP,…
* **Host Operating System (hệ điều hành của server):**hệ điều hành đang sử dụng cho server đang sử dụng, có thể là macOS, Windows hoặc Linux.
* **Hypervisor** (phần mềm giám sát máy ảo): có thể coi máy ảo như một máy tính độc lập được đóng gói vào một tập tin duy nhất, nhưng cần phải có một phần mềm để có thể chạy tập tin đó. Các hypervisor được sử dụng để tạo, startup, dừng và reset lại các máy ảo, cho phép mỗi máy ảo hoặc “guest” truy cập vào lớp tài nguyên phần cứng vật lý bên dưới, chẳng hạn như CPU, RAM và ổ cứng. Phần mềm này cũng có thể giới hạn số lượng tài nguyên hệ thống mà mỗi máy ảo có thể sử dụng để đảm bảo cho nhiều máy ảo cùng hoạt động đồng thời trên một hệ thống. Có thể kể đến các hypervisor phổ biến trên thị trường hiện nay như HyperKit cho macOS, Hyper-V cho Windows và KVM cho Linux. Ngoài ra, 2 hypervisor phổ biến khác là VirtualBox và VMWare.
* **Guest OS (hệ điều hành của máy ảo):**với mỗi máy ảo được tạo ra, người quản trị cần phải cài đặt một hệ điều hành đi kèm cho máy ảo đó để cài đặt và triển khai các service cần thiết. Ví dụ bạn cần triển khai 3 service trên 3 máy ảo khác nhau và mỗi “Guest OS” sẽ tiêu tốn ít nhất 700 MB dung lượng ổ cứng, vậy sẽ tiêu tốn khoảng 2.1 GB dung lượng ổ cứng của server để tạo 3 máy ảo để triển khai 3 service khác nhau. Chưa kể để các “Guest OS” này sẽ sử dụng một lượng tài nguyên khác của hệ thống như CPU, RAM.
* **Bins / Libs (các tập tin / thư viện cần thiết):**các service / application sẽ cần phải có các gói tập tin, thư viện đi kèm để có thể hoạt động.
* **App (application – service):** mã nguồn của các ứng dụng, phần mềm.

## ****Kiến trúc của service hoạt động trên Docker container****

Nếu so sánh với kiến trúc của service khi hoạt động trên nền tảng máy ảo, service hoạt động trên Docker container sẽ loại bỏ lớp “Guest OS”, với tốc độ khởi tạo service nhanh hơn hẳn so với việc sử dụng máy ảo. Docker Container sẽ giảm thiểu và đơn giản hóa các bản cập nhật bảo mật. Kiến trúc của service khi hoạt động trên nền tảng Docker container thông qua ảnh minh họa bên dưới:



***Kiến trúc của service trên Docker container***

* **Infrastructure (hạ tầng):**có thể là laptop, một server chuyên dụng (dedicated server) hoat động trong một trung tâm dữ liệu (Data Center) hoặc một server riêng ảo (Virtual Private Server – VPS) trên cloud như DigitalOcean, Amazon, GCP,…
* **Host Operating System (hệ điều hành của server):**hệ điều hành đang sử dụng cho server đang sử dụng, có thể là macOS, Windows hoặc Linux.
* **Docker daemon (còn gọi là Docker Engine):**đây là service hoạt động trên server, được dùng để quản lý các thành phần cần thiết để khởi tạo và tương tác với Docker container.
* **Bins / Libs (các tập tin / thư viện cần thiết):**các gói tập tin, thư viện đi kèm của service được thêm vào Docker image.
* **App (application – service):** mã nguồn của các ứng dụng, phần mềm được thêm vào Docker container.

## ****Sự khác nhau giữa máy ảo và Docker Container****

Docker daemon có thể giao tiếp trực tiếp với hệ điều hành của server và phân bổ tài nguyên cho các Docker container đang chạy, đảm bảo mỗi container hoạt động độc lập với các container khác và hệ điều hành của server. Thay vì phải đợi một phút để máy ảo khởi động, người dùng có thể khởi động Docker container chỉ trong vài mili giây và tiết kiệm được rất nhiều dung lượng ổ đĩa và các tài nguyên hệ thống khác do không cần phải sử dụng “guest OS” cồng kềnh cho mỗi ứng dụng. Người dùng sẽ không cần ảo hóa vì Docker chạy trực tiếp trên hệ điều hành của server.

|  |  |
| --- | --- |
| **Máy ảo** | **Docker container** |
| Kích thước (dung lượng) lớn. | Kích thước (dung lượng) nhỏ. |
| Hiệu suất hạn chế. | Hiệu suất gốc (native). |
| Mỗi máy ảo sẽ có một hệ điều hành riêng. | Container sẽ sử dụng hệ điều hành của host. |
| Ảo hóa về mặt phần cứng | Ảo hóa về mặt hệ điều hành |
| Thời gian khởi động tính theo phút | Thời gian khởi động tính theo mili giây |
| Phân bổ bộ nhớ theo nhu cầu cần thiết | Yêu cầu ít dung lượng bộ nhớ hơn |
| Hoàn toàn bị cô lập và an toàn hơn | Cô lập ở mức tiến trình, có thể kém an toàn hơn |

Việc sử dụng máy ảo hay Docker container sẽ phụ thuộc vào nhu cầu sử dụng của người dùng. Máy ảo rất phù hợp trong việc cách ly tài nguyên hệ thống và toàn bộ môi trường làm việc. Đây sẽ là lựa chọn tốt hơn để chạy các ứng dụng yêu cầu tất cả tài nguyên và chức năng của hệ điều hành khi bạn cần chạy nhiều ứng dụng trên server hoặc có nhiều hệ điều hành khác nhau để quản lý. Ví dụ: công ty của bạn cung cấp dịch vụ web hosting, bạn có thể sẽ sử dụng máy ảo để phân phối tài nguyên của server công ty cho từng khách hàng.

Mặt khác, triết lý của Docker là cô lập các ứng dụng riêng lẻ, không phải toàn bộ hệ thống. Một ví dụ hoàn hảo về điều này sẽ là chia nhỏ một loạt các dịch vụ ứng dụng web thành các Docker image của riêng chúng và triển khai chúng bằng Docker Container. Docker Container là lựa chọn tốt hơn khi ưu tiên lớn nhất của bạn là tối đa hóa số lượng ứng dụng đang chạy trên một số lượng server tối thiểu.

<===================================================================🡺

## ****ỨNG DỤNG CỦA DOCKER****

Docker là một platform ở tầng OS (operating system) có thể tinh chỉnh được, phục vụ cho việc chạy ảo hóa các dịch vụ/ ứng dụng một cách nhanh chóng.

Ngày xưa, mọi người hay dùng máy ảo (VMs – Virtual Machines) để triển khai ứng dụng, việc này mất rất nhiều thời gian. Còn khi sử dụng Docker thì thời gian triển khai một ứng dụng nằm trong Docker container được rút ngắn đi rất nhiều.

#### ****Lợi ích của việc sử dụng Docker****

* Thứ nhất là**tính tiện lợi, nhanh chóng.** Nếu như trước đây triển khai dịch vụ/ứng dụng trên VMs tốn vài chục phút thì với Docker, nó sẽ giảm xuống chỉ còn vài phút.
* Thứ hai là**tiết kiệm resource (tài nguyên).** Mỗi docker container sử dụng bao nhiêu resource thì sẽ tốn bấy nhiêu, không sử dụng thì resource sẽ được trả về cho máy chủ host.
* Thứ ba là**hệ thống có mức độ tự động mở rộng cao hơn** để đáp ứng nhu cầu phục vụ người dùng. Việc khởi tạo và tắt đi của một docker container diễn ra khá dễ dàng và dựa trên các metric của hệ thống.
* Thứ tư là **dễ dàng automate (tự động hóa) việc quản lý các docker container** thông qua Kubernetes hoặc Docker Swarm.

Chẳng hạn khi có 1 docker container chết đi thì hệ thống sẽ tự động khởi tạo 1 docker container tương tự như vậy trong cluster và trả về thông báo. DevOps Engineer không cần phải kiểm tra thủ công.

Ngoài ra, anh Thành còn chia sẻ rằng Docker sẽ là một điểm sáng trong CV của DevOps vì không chỉ ở hiện tại mà xu hướng trong khoảng 1-2 năm tới, người ta đều sẽ dùng Docker để triển khai dịch vụ/ứng dụng.

#### ****Khi nào cần dùng Docker?****

Trước khi quyết định có nên sử dụng Docker, anh Thành đưa ra lời khuyên rằng bạn có thể thử trả lời 2 câu hỏi này:

* Mức độ tiếp xúc của lập trình viên với Docker như thế nào, lập trình viên đã làm việc với Docker hay chưa?
* Hệ thống mà DevOps Engineer đang làm có yêu cầu scalability (khả năng mở rộng) cao hay không?

Nếu 2 câu trả lời đều là “có” thì ta sẽ dùng Docker.

Tuy nhiên, bổ sung thêm rằng DevOps Engineer không nhất thiết phải biết sử dụng Docker mà còn tùy công ty, quan trọng là công ty có yêu cầu hay không. Vẫn còn rất nhiều công ty vẫn đang dùng máy chủ ảo VPS. Hiện tại thì chắc chỉ dừng ở mức độ “nên biết sử dụng Docker” chứ không phải bắt buộc.