CÁC ĐIỂM CẦN ÔN TẬP (UPDATED – ver 2)

Chương I – VI

Lưu ý quan trọng: đặc biệt quan tâm đến những điểm *bôi xanh lá cây.* Những đoạn chữ đen thông thường mô tả rõ hơn khái niệm liên quan, nếu có thời gian có thể đọc thêm, nếu không, quay lại các điểm *bôi xanh lá cây*.

----------######----------

Các từ viết tắt cần lưu ý

XML	Extensible markup language.
SAN	Storage area network
WSDL	Web Services Description Language
DFS	Distributed File System
NFS	Network File System
Shadow data	Thường là những dữ liệu nằm trong nhóm chưa/không được quản lý
NIST	National Institute of Standards and Technology
S3 trong Amazon S3	Simple Storage Service

CHƯƠNG I – TỔNG QUAN

- Kiến trúc đám mây có thể kết hợp phần mềm chạy trên phần cứng ảo hóa ở nhiều địa điểm để cung cấp dịch vụ theo yêu cầu.
- **Điện toán đám mây dựa trên bộ giao thức (đòi hỏi một số giao thức chuẩn) để quản lý thông tin liên lạc giữa các quy trình (inter-process communications).**
- Một trong những ứng dụng của nền tảng điện toán đám mây là tạo ra các phần mềm phức tạp hơn.
- Các thiết bị/công nghệ ảo hóa đang trở thành một đối tượng triển khai điện toán đám mây tiêu chuẩn rất quan trọng
- **Điện toán đám mây mang đến những cơ hội mới cho người dùng và developers**
- Mô hình NIST ban đầu không yêu cầu đám mây phải sử dụng công nghệ ảo hóa để tập hợp tài nguyên. (https://www.nist.gov/system/files/documents/itl/cloud/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf)
- Lợi ích liên quan đến việc tạo ra các tài nguyên được gộp chung trong một hệ thống hỗ trợ nhiều người dùng (multi-tenant): tự phục vụ theo yêu cầu (on-demand self-service)
- Đặc tính của điện toán điện toán đám mây giúp giảm chi phí quản lý: điện toán đám mây cho phép quản lý tài nguyên tự động (automated resource management) & tự phục vụ (self-service)
- Thuật ngữ điện toán đám mây bắt nguồn từ sơ đồ mạng (network diagrams)

- Mô hình cung cấp dịch vụ điện toán đám mây Public cloud có nhiều nét tương đồng nhất với dịch vụ điện lưới hiện tại (electrical utility grid)

Cloud bursting

Mở rộng đám mây (cloud bursting) là *một phương pháp cấu hình sử dụng tài nguyên điện toán đám mây bất cứ khi nào cơ sở hạ tầng tại chỗ đạt đến công suất cao nhất*. Khi các tổ chức dùng hết tài nguyên điện toán trong trung tâm dữ liệu nội bộ của mình, họ sẽ *mở rộng phần khối lượng công việc phụ sang các dịch vụ đám mây của bên thứ ba bên ngoài*. Mở rộng đám mây là một cách thuận tiện và tiết kiệm chi phí để hỗ trợ khối lượng công việc với các kiểu nhu cầu khác nhau và khi nhu cầu tăng đột biến theo mùa. *Ví dụ: cloud bursting cho phép cho tràn giao dịch (transaction overflow) trong hệ thống đặt chỗ/vé.*

Elasticity

Tính co giãn của điện toán đám mây (cloud elasticity) là khả năng tăng hoặc giảm quy mô của dịch vụ điện toán đám mây: tài nguyên điện toán đám mây trong thời gian thực — chẳng hạn như sức mạnh xử lý của CPU, bộ nhớ, dung lượng lưu trữ, cũng như băng thông đầu vào và đầu ra — để đáp ứng những thay đổi đột xuất trong lưu lượng truy cập trực tuyến.

Trong các lý do sau đây, lý do nào là hợp lý để không áp dụng giải pháp điện toán đám mây?
Phần cứng cục bộ (local hardware) đang đáp ứng tốt/đầy đủ cho các khối lượng công việc CNTT không thay đổi
Số lượng nhân viên trong doanh nghiệp/tổ chức thường xuyên thay đổi.
Một doanh nghiệp trải qua những đợt tăng đột biến của dự án không thể đoán trước trong suốt cả năm

Đặc điểm nào của điện toán đám mây giúp giảm chi phí trung tâm dữ liệu?
Tính linh hoạt và tính bền vững (sustainability) của các mô hình dịch vụ điện toán đám mây
Sử dụng công nghệ tiết kiệm năng lượng trong các trung tâm dữ liệu điện toán đám mây
Cho phép các dịch vụ được tự động chuyển (migrate) giữa các trung tâm dữ liệu khi cần thiết
Khả năng sử dụng từ xa cho các thiết bị di động

Yếu tố nào góp phần gây ra độ trễ mạng chủ yếu do tình trạng đăng ký quá mức?
Sự tắc nghẽn
Số lượng hops (bước nhảy)
Số lượng node mạng
Độ trễ do giao thức

Xác định câu sai về điện toán đám mây.
Không có cái nào được đề cập
Ảo hóa gán một tên logic cho một tài nguyên vật lý và sau đó cung cấp một con trỏ đến tài nguyên vật lý đó khi có yêu cầu được thực hiện
Các thiết bị ảo đang trở thành một đối tượng triển khai điện toán đám mây tiêu chuẩn rất quan trọng
Điện toán đám mây đòi hỏi một số giao thức chuẩn

Chỉ ra câu sai.
Điện toán đám mây công cộng (public cloud) có thể được quản lý bởi các tổ chức thành viên hoặc bởi bên thứ ba
Điện toán đám mây cộng đồng (community cloud) có thể được quản lý bởi các tổ chức thành viên hoặc bởi bên thứ ba
Điện toán đám mây riêng có thể ở trong hoặc ngoài cơ sở hạ tầng CNTT của doanh nghiệp
Không có đáp án đúng

Chỉ ra câu sai.
Không có đáp án đúng
Điện toán đám mây dựa vào bộ giao thức cần thiết để quản lý thông tin liên lạc giữa các quy trình
Điện toán đám mây đòi hỏi một số giao thức chuẩn
Các thiết bị ảo đang trở thành đối tượng triển khai rất quan trọng trong điện toán đám mây tiêu chuẩn

Mô hình dịch vụ điện toán đám mây nào cho phép tính linh hoạt cao nhất trong việc phát triển ứng dụng?
IaaS
PaaS
SaaS
XaaS

CHƯƠNG II – ẢO HÓA

MỘT SỐ KHÁI NIỆM

- Workload là đơn vị cơ bản của máy khách ảo hóa trong triển khai IaaS. Máy ảo (VM) được cung cấp trên nền tảng dịch vụ IaaS (ví dụ AWS EC2)

Amazon EC2 cung cấp môi trường điện toán ảo, được gọi là
Instances
Messages
Chunks
Instances

- Mục đích của việc sao chép/nhân bản (cloning) máy ảo là để tạo một bản sao giống hệt của máy ảo để triển khai hoặc thử nghiệm
- Ưu điểm chính của việc sử dụng máy ảo để thử nghiệm và phát triển phần mềm là cô lập (Isolation) môi trường thử nghiệm khỏi máy chủ (host system)
- Công nghệ ảo hóa, bằng cách cho phép nhiều máy ảo chạy trên một máy vật lý duy nhất, góp phần vào hiệu quả sử dụng tài nguyên trong trung tâm dữ liệu.

Chỉ ra câu phát biểu đúng.
Tất cả các đáp án trên
Nhược điểm của công nghệ máy ảo là việc có một số tài nguyên được sử dụng gián tiếp có nghĩa là sẽ phát sinh tài nguyên phụ trội (overhead) cần thiết để vận hành
Máy ảo cung cấp khả năng chạy nhiều phiên bản máy, mỗi phiên bản có hệ điều hành riêng
Máy ảo là máy tính được tách biệt khỏi máy tính vật lý mà máy ảo đang chạy trên đó

- Máy ảo có thể cải thiện khả năng phục hồi sau thảm họa bằng cách cho phép chuyển VM sang máy chủ vật lý khác trong trường hợp xảy ra lỗi

Máy ảo góp phần như thế nào vào việc hợp nhất phần cứng trong các trung tâm dữ liệu?
Bằng cách cho phép nhiều máy chủ vật lý chia sẻ cùng một hệ thống phần cứng (hardware)
Bằng cách giảm số lượng máy ảo trên mỗi máy chủ vật lý
Bằng cách tăng số lượng máy chủ vật lý cần thiết
Bằng cách cung cấp quyền truy cập trực tiếp vào phần cứng vật lý cho mỗi máy ảo

- *Tính di động (portability) của máy ảo có nghĩa là Khả năng di chuyển máy ảo giữa các máy chủ vật lý.* Khả năng di động của máy ảo cho phép bạn di chuyển các máy ảo (VM)

giữa các cấu hình phần cứng hoặc nền tảng khác nhau (different hardware configurations or platforms). Điều này giúp dễ dàng mở rộng tài nguyên tính toán và sao lưu dữ liệu.

- *Snapshot máy ảo: Để giúp khôi phục máy ảo về trạng thái trước đó.* Snapshot là một bản sao tức thời của máy ảo đang chạy, được lưu trên cùng ổ cứng vật lý. Nó hữu ích khi bạn cần thử nghiệm cài đặt phần mềm, cấu hình mới trên máy ảo, và nếu gặp lỗi có thể nhanh chóng phục hồi về nguyên trạng trước đó.
- Trong ảo hóa, hệ thống máy chủ (host system) là hệ điều hành chính chạy trên máy vật lý. Hệ điều hành khách (guest operating system) là Hệ điều hành được cài đặt trên máy ảo
- Cô lập máy ảo (virtual machine isolation) có nghĩa là các máy ảo chạy độc lập và tách biệt với nhau cũng như với hệ thống máy chủ.
- **AWS EC2 chạy trên Xen Hypervisor** và là nền tảng máy chủ ảo cho phép người dùng tạo và chạy máy chủ ảo trên hạ tầng máy của của Amazon.
- **Virtual Machined Isolation** là một kỹ thuật tạo ra môi trường thực thi biệt lập trên một máy vật lý duy nhất. *Mỗi máy ảo chạy trong môi trường ảo của riêng nó, biệt lập với các máy ảo khác chạy trên cùng một máy vật lý*. Điều này cho phép nhiều hệ điều hành hoặc phiên bản của cùng một hệ điều hành chạy đồng thời, mỗi hệ điều hành có tài nguyên phần cứng ảo hóa, bộ nhớ và lưu trữ riêng.
- Trong ảo hóa dữ liệu, lớp dữ liệu ảo là lớp logic hợp nhất (abstract) dữ liệu từ nhiều nguồn khác nhau và cung cấp quyền truy cập vào dữ liệu đó. Lợi ích chính của việc sử dụng ảo hóa dữ liệu là giúp giảm yêu cầu về sao chép & lưu bảo sao (replication) dữ liệu. Mục tiêu chính là cung cấp chế độ xem thống nhất dữ liệu từ nhiều nguồn mà không cần di chuyển hoặc sao chép dữ liệu. Ảo hóa dữ liệu thường được sử dụng nhất để tích hợp dữ liệu từ những nguồn như Cơ sở dữ liệu quan hệ và không quan hệ, nền tảng dữ liệu lớn, ứng dụng điện toán đám mây và tệp phẳng

Đặc điểm nào sau đây KHÔNG phải là đặc điểm của ảo hóa dữ liệu?
Ảo hóa dữ liệu liên quan đến việc di chuyển vật lý của dữ liệu giữa các hệ thống
Ảo hóa dữ liệu cung cấp chế độ xem dữ liệu theo thời gian thực
Ảo hóa dữ liệu tích hợp dữ liệu từ các nguồn không đồng nhất
Ảo hóa dữ liệu tóm tắt các nguồn dữ liệu cơ bản từ người dùng cuối

- Mục đích chính của việc di chuyển Máy ảo (VM) là để chuyển các máy ảo giữa các máy chủ vật lý để cân bằng tải hoặc bảo trì.

Chỉ ra câu sai.
Tất cả ứng dụng trên nền tảng điện toán đám mây **đều kết hợp tài nguyên thành các nhóm dùng chung**, có thể linh hoạt phân phối cho người dùng theo nhu cầu
Tất cả các đáp án trên

Ảo hóa gán một tên logic cho một tài nguyên vật lý và sau đó cung cấp một con trỏ đến tài nguyên vật lý đó khi có yêu cầu được thực hiện
Việc trừu tượng (abstraction) hóa cho phép điện toán đám mây đạt được lợi ích quan trọng nhất: khả năng truy cập dùng chung, mọi lúc mọi nơi

Công nghệ ảo hóa nào sau đây là mã nguồn mở?
Oracle Virtualbox
VMware vSphere
Microsoft Hyper-V
Citrix Xenserver

HYPERVISOR

Hypervisor (VMM) là một phần mềm, phần cứng hoặc một chương trình, có vai trò khởi tạo, quản lý và điều khiển nhiều máy ảo trên cùng một máy chủ vật lý duy nhất. Máy ảo (Virtual Machine - VM) là môi trường ảo được tạo ra bởi Hypervisor, nhằm mục đích khởi chạy các hệ điều hành và ứng dụng. *Hypervisor phân bổ tài nguyên phần cứng cho máy ảo*

Mỗi Hypervisor cho phép mỗi một máy ảo hoặc khách truy cập vào các lớp tài nguyên vật lý bên dưới, bao gồm CPU, bộ nhớ lưu trữ hoặc RAM. Ngoài ra, Hypervisor cũng có thể giới hạn số lượng tài nguyên phân chia cho từng máy ảo, đảm bảo nhiều máy ảo có thể cùng hoạt động trên một hệ thống.

Hypervisor tạo một lớp ảo hóa trung gian hoạt động giữa phần cứng Server (máy chủ) và hệ điều hành ảo. Qua đó, Hypervisor giúp chia tách các tài nguyên của máy chủ vật lý cho các máy ảo bên trong và quản lý từng máy ảo độc lập nhau.

Sau khi tạo ra một máy ảo, Hypervisor sẽ cung cấp các tài nguyên được như CPU, RAM, hệ điều hành, bộ lưu trữ, etc. cho máy ảo. Thông thường, muốn sao chép máy ảo, doanh nghiệp cần sao chép thủ công toàn bộ thông tin của nó. Nhưng với Hypervisor, doanh nghiệp chỉ cần chọn các bộ phận cần sao chép, Hypervisor sẽ tự thực hiện quy trình này thay thế cho doanh nghiệp.

Có 2 loại Hypervisor chính:

Loại 1: Native Hypervisors

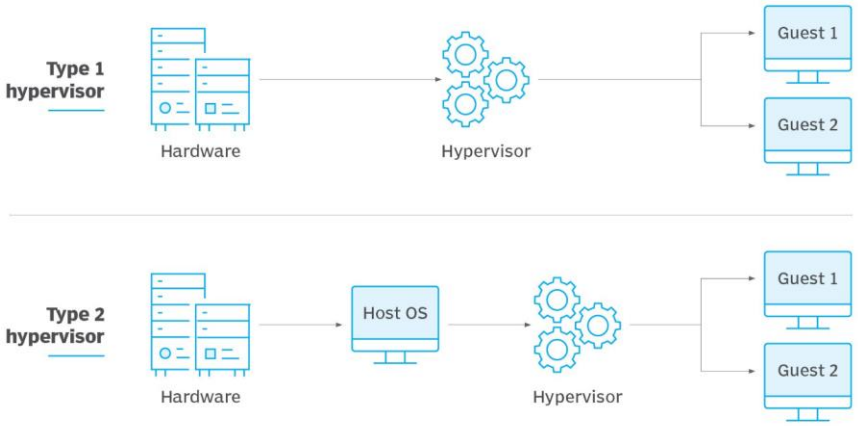
> Native Hypervisors hoạt động trực tiếp trên phần cứng máy chủ vật lý, thay vì hoạt động trên hệ điều hành nào đó như các Hypervisor khác.
>
> Native Hypervisor cho phép doanh nghiệp chia sẻ tài nguyên giữa các máy ảo hiệu quả, tăng hiệu suất và giảm độ trễ tốt. Giải pháp này cũng mang lại môi trường ảo hóa độc lập, không phụ thuộc vào các hệ điều hành chủ (Host OS).

Hiện nay, Native Hypervisor được sử dụng chính trong các trung tâm dữ liệu (Data Center) và môi trường ảo hóa của doanh nghiệp, nhằm mục đích tăng hiệu quả và tính linh hoạt cho hệ thống.

Loại 2: Hosted Hypervisors

Hosted Hypervisors hoạt động trên một hệ điều hành chủ như một ứng dụng. Với Hosted, các hệ điều hành khách (Guest OS) có thể chạy cùng lúc trên cùng một máy tính vật lý.

Ngày nay, Hosted Hypervisor được dùng nhiều trong việc kiểm thử phần mềm hoặc thử nghiệm những chương trình, hệ thống mới mà không cần phải cài đặt trên các máy tính có hệ điều hành khác nhau. Tuy nhiên, doanh nghiệp cần lưu ý rằng Hosted Hypervisor có thể sẽ làm giảm hiệu suất, do chúng sử dụng nguồn tài nguyên chia sẻ với Host OS.



Trong các giải pháp sau đây, giải pháp nào là hypervisor được sử dụng cho ảo hóa?
VMware ESXi
Apache
Linux
MySQL

Tham khảo:

- https://aws.amazon.com/what-is/hypervisor/#:~:text=A%20hypervisor%20is%20a%20software,individual%20virtual%20machines%20as%20required.
- *https://www.ibm.com/think/topics/hypervisors*
- *https://www.techtarget.com/searchitoperations/definition/hypervisor*

Ảo hóa hoàn toàn (Full virtualization)
& Ảo hóa bán phần - ảo hóa song song (Paravirtualization)

Full Virtualization (Ảo hóa toàn phần)
Full Virtualization là một kỹ thuật ảo hóa trong đó phần mềm ảo hóa (hypervisor) cho phép tạo ra và quản lý các máy ảo (virtual machines - VMs) trên một phần cứng duy nhất mà không yêu cầu thay đổi hoặc điều chỉnh hệ điều hành của các máy ảo này. Trong mô hình full virtualization, mỗi máy ảo chạy độc lập và có thể thực thi hệ điều hành của riêng nó, hoàn toàn tách biệt với phần còn lại của hệ thống vật lý. Trong Full Virtualization (Ảo hóa toàn phần), VM được cài đặt dưới dạng Hypervisor loại 1 trực tiếp trên phần cứng.

Paravirtualization (Ảo hóa song song)
Ảo hóa song song là loại ảo hóa mà trong đó nó không ảo hóa phần cứng để chạy hệ điều hành ảo mà thay vào đó tạo ra một lớp giao diện phần mềm để các hệ điều hành ảo và hypervisor giao tiếp với nhau.

Ví dụ: Hyper-V của Microsoft sử dụng công nghệ "paravirtualization", có nghĩa là nó cung cấp một môi trường ảo hóa nơi các hệ điều hành khách có thể tương tác trực tiếp với hypervisor thông qua các trình điều khiển tối ưu hóa, cho phép hiệu suất tốt hơn so với ảo hóa hoàn toàn, nơi hệ điều hành khách cần phải mô phỏng phần cứng hoàn toàn; về cơ bản, đây là một phương pháp kết hợp giữa ảo hóa hoàn toàn và truy cập trực tiếp vào phần cứng trong môi trường ảo hóa.

Khác biệt giữa ảo hóa hoàn toàn (Full virtualization) và ảo hóa bán phần (Paravirtualization) là gì?
Ảo hóa bán phần đòi hỏi phải sửa đổi hệ điều hành khách, trong khi ảo hóa hoàn toàn thì không.
Ảo hóa hoàn toàn đòi hỏi phải sửa đổi hệ điều hành khách, trong khi ảo hóa bán phần thì không.
Ảo hóa hoàn toàn nhanh hơn ảo hóa bán phần.
Ảo hóa bán phần chỉ được sử dụng để ảo hóa lưu trữ.

Trong một trung tâm dữ liệu ảo hóa, giải pháp nào sau đây cho phép tự động cung cấp máy ảo và khối lượng công việc?
Phần mềm quản lý như VMware vCenter hoặc Microsoft System Center
Phần cứng lưu trữ vật lý
Hypervisor
Thời gian hoạt động (uptime) được đảm bảo bởi nhà cung cấp dịch vụ đáp ứng được nhu cầu về tính khả dụng của tổ chức/doanh nghiệp.

Virtual appliances (VA)

Virtual appliance thường bao gồm một phiên bản của một hệ điều hành được cấu hình để chạy một application duy nhất theo kiểu appliance dưới dạng một máy ảo (VM). Tận dụng công nghệ máy ảo và thừa hưởng tính chất của một appliance, virtual appliance đem lại một số lợi ích:

- Mỗi application bây giờ tách biệt với nhau → bảo mật

- Tương tự như việc sử dụng hardware appliance, hệ điều hành trong virtual appliance sẽ trở nên trong suốt, điều duy nhất mà bạn phải quan tâm là application chạy trên đó.
- Nếu bạn là một nhà sản xuất phần mềm, virtual appliance sẽ giúp bạn phân phối phần mềm mà không phải lo lắng đến việc tích hợp chúng vào hệ thống của khách hàng.

Sự khác nhau giữa VA và VM

- Một máy ảo có thể xem như một máy tính thông thường, trong đó các tài nguyên như CPU, bộ nhớ RAM, đĩa cứng và card mạng là ảo. Để có thể chạy được ứng dụng, người dùng cần phải cài đặt và cấu hình hệ điều hành và các ứng dụng trên đó. VA khác máy ảo một chút ở chỗ, hệ điều hành và các ứng dụng đã được cài đặt và cấu hình sẵn.

Ví dụ: Các ứng dụng như máy chủ Web hoặc máy chủ cơ sở dữ liệu có thể chạy trên ảnh máy ảo (virtual machine image) được gọi là Virtual appliances

---

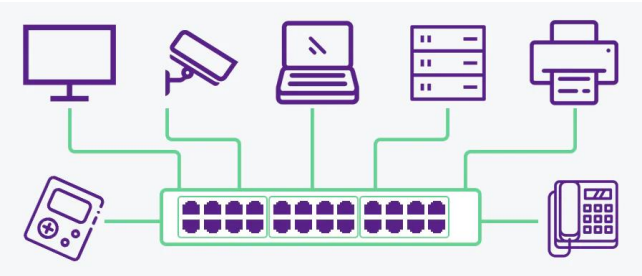
Thiết bị chuyển mạch (switch) ảo

Switch mạng, hay thường được gọi là switch kết nối các thiết bị trong một mạng với nhau, cho phép chúng giao tiếp bằng cách trao đổi gói dữ liệu. Nó có thể là các thiết bị phần cứng quản lý mạng vật lý hoặc các thiết bị ảo dựa trên phần mềm.

Mục đích của bộ chuyển mạch ảo (virtual switch) trong ảo hóa là để cung cấp kết nối mạng dành riêng cho mỗi máy ảo

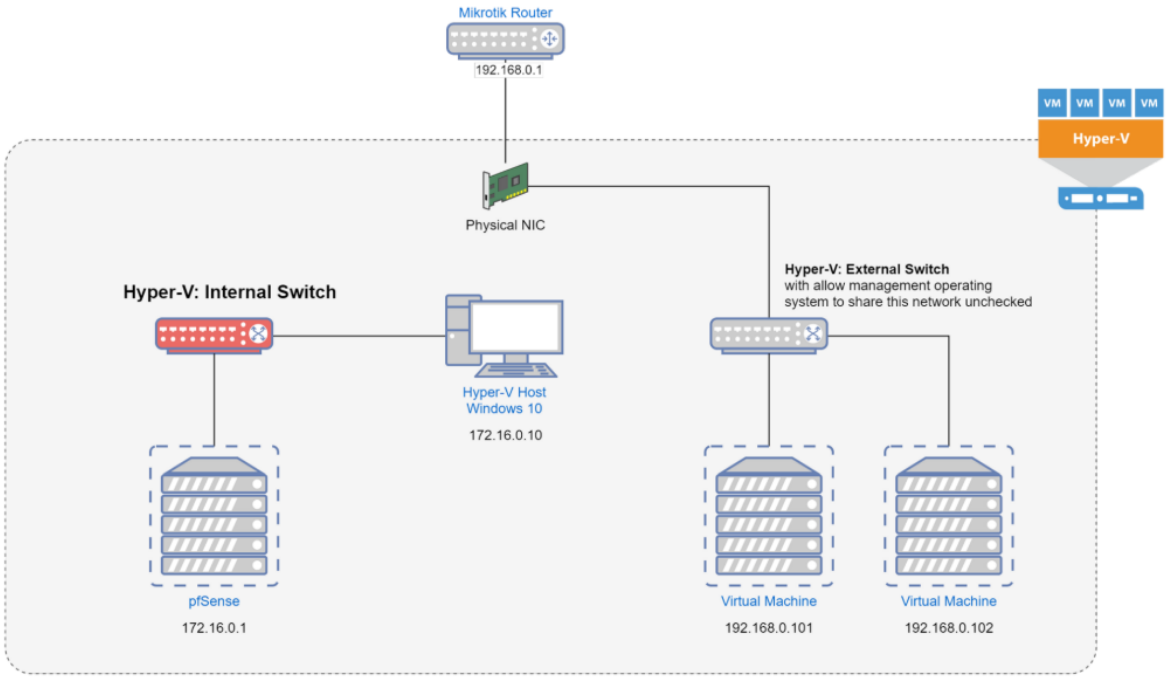
Physical Switch (thiết bị phần cứng)

Một switch hoạt động trên tầng liên kết dữ liệu, hay tầng 2, của mô hình Open Systems Interconnection (OSI). Trong một mạng cục bộ (LAN) sử dụng Ethernet, một switch mạng xác định nơi gửi mỗi khung tin nhắn đến bằng cách xem địa chỉ điều khiển truy cập phương tiện (MAC). Nó duy trì các bảng khớp mỗi địa chỉ MAC với cổng nhận địa chỉ MAC đó.



Virtual Switch (bộ chia mảng ảo)

Virtual Switch (bộ chia mảng ảo) vận hành y hệt như một Physical Switch (bộ chia mạng vật lý). Nó hoạt động ở Layer 2 của OSI Protocol với nhiệm vụ gửi gói dữ liệu đến địa chỉ MAC.

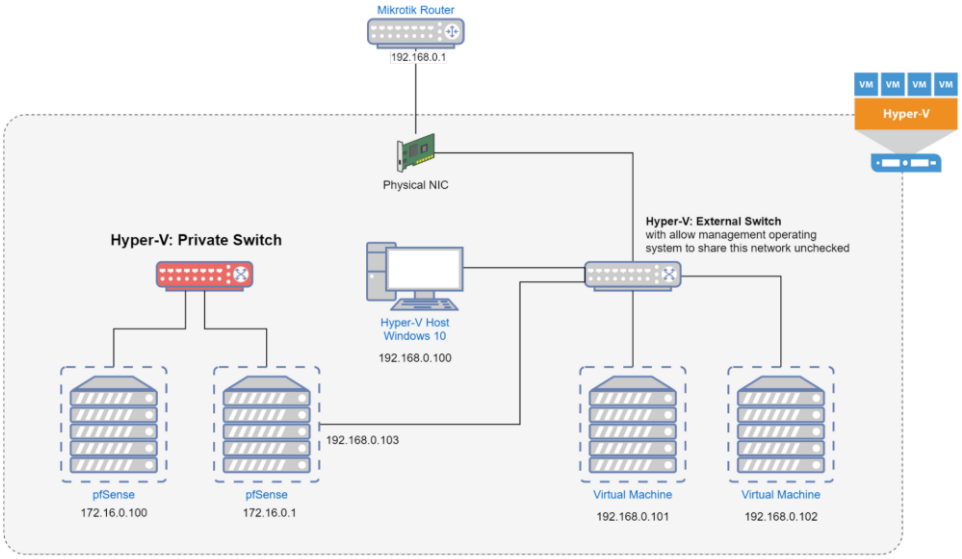


Một số loại switch ảo, bao gồm các loại sau:

External virtual switches. Loại switch này được liên kết với một thẻ mạng vật lý (physical network card) và cung cấp cho các máy ảo (VM) kết nối với mạng bên ngoài. Các VM kết nối với một switch ảo bên ngoài chung có thể giao tiếp với nhau. Hệ điều hành máy chủ cũng có thể giao tiếp qua các switch ảo bên ngoài.

Internal virtual switches. Khác với switch ảo bên ngoài, switch ảo nội bộ không được liên kết với một bộ điều hợp mạng vật lý. Thay vào đó, các mạng được kết nối với nhau thông qua một switch ảo nội bộ hoàn toàn được định nghĩa bằng phần mềm. Các máy chủ kết nối với một switch ảo nội bộ có thể giao tiếp với nhau cũng như với các VM đã được kết nối với switch ảo nội bộ. Tuy nhiên, các VM kết nối với một switch ảo nội bộ không thể kết nối với internet hoặc truy cập các tài nguyên mạng không được kết nối với switch ảo nội bộ. Điều này hữu ích cho việc tạo ra các môi trường cách ly mà vẫn có thể kiểm soát máy chủ.

Private virtual switches. Một switch ảo riêng tư hoàn toàn cách ly VM. Các VM kết nối với mạng switch ảo riêng tư có thể giao tiếp với nhau, nhưng không thể giao tiếp với bất kỳ tài nguyên nào bên ngoài switch ảo riêng tư.


Mikrotik Router
192.168.0.1
VM
VM
VM
VM
Hyper-V
Physical NIC
Hyper-V: External Switch
with allow management operating
system to share this network unchecked
Hyper-V: Private Switch
Hyper-V Host
Windows 10
192.168.0.100
192.168.0.103
pfSense
172.16.0.100
pfSense
172.16.0.1
Virtual Machine
192.168.0.101
Virtual Machine
192.168.0.102

CHƯƠNG III – DATA & STORAGE

MỘT SỐ KHÁI NIỆM

- Một dịch vụ lưu trữ đám mây được quản lý: có nghĩa là nhà cung cấp dịch vụ quản lý hầu hết cơ sở hạ tầng và các hoạt động liên quan đến lưu trữ dữ liệu, cho phép người dùng tập trung vào việc truy cập và quản lý dữ liệu của họ mà không cần phải quan tâm đến việc quản lý hệ thống phần cứng hoặc phần mềm chạy bên dưới. **S3 (Dịch vụ Lưu trữ Đơn giản của Amazon) được coi là dịch vụ lưu trữ đám mây được quản lý (managed cloud storage service)**
- Lợi thế của quản lý dữ liệu điện toán đám mây là cho phép mở rộng quy mô dễ dàng và khả năng xử lý hiệu quả các tập dữ liệu lớn. Nếu so với lưu trữ tại chỗ (on-premises storage) truyền thống lưu trữ điện toán đám mây mở rộng dễ dàng hơn với dung lượng lưu trữ linh hoạt. Ngoài ra, lợi ích của việc sử dụng điện toán đám mây để lưu trữ dữ liệu còn nằm ở các điểm: Cải thiện bảo mật dữ liệu nhờ lưu trữ tập trung và mã hóa tiên tiến, Giảm chi phí so với các giải pháp lưu trữ dữ liệu tại chỗ truyền thống., Tăng cường khả năng truy cập dữ liệu từ bất kỳ đâu có kết nối Internet.
- Hệ thống lưu trữ đám mây có độ tin cậy (reliable) cao do có thiết lập các hệ thống dự phòng: network, server lưu trữ, server name (máy chủ tên) & có thực hiện Sao chép (Replication)
- Giải pháp lưu trữ sao lưu đám mây yêu cầu ít nhất 256 bits mã hóa dữ liệu (https://aws.amazon.com/vi/what-is/cryptography/)
- Shadow data: dữ liệu được lưu trữ trong các nguồn dữ liệu không được quản lý
- Mục đích của việc quản lý phiên bản dữ liệu trong lưu trữ điện toán đám mây là để duy trì các phiên bản dữ liệu trước đó cho mục đích phục hồi và kiểm tra
- Khoảng cách địa lý giữa người dùng và trung tâm dữ liệu là yếu tố chính ảnh hưởng đến tốc độ lưu trữ điện toán đám mây
- Phân tầng lưu trữ (storage tiering) điện toán đám mây là di chuyển dữ liệu ít được truy cập đến bộ lưu trữ chậm hơn, chi phí thấp hơn
- Data lifecycle management (DLM) là quản lý vòng đời dữ liệu: quản lý dữ liệu thông qua các giai đoạn tạo, lưu trữ, sử dụng và xóa

Tính năng lưu trữ điện toán đám mây nào có thể giúp khôi phục dữ liệu trong trường hợp vô tình xóa hoặc hỏng?
Versioning and snapshots
Data replication
Data backup
Data compression

- Cổng lưu trữ điện toán đám mây (cloud storage gateways): Để kết nối môi trường CNTT on-premises với lưu trữ điện toán đám mây

---

WEB SERVICES

Webservices là tập hợp các giao thức và tiêu chuẩn mở được sử dụng để ***trao đổi dữ liệu giữa các ứng dụng hoặc giữa các hệ thống.***

SOAP Web Service SOAP là viết tắt của Simple Object Access Protocol. Nó là một giao thức dựa trên XML để truy cập các web services. Vì dựa trên XML nên SOAP là một giao thức không phụ thuộc platform cũng như bất kì ngôn ngữ lập trình nào

- Việc giao tiếp giữa các dịch vụ thường được thực hiện bằng giao thức SOAP

RESTful Web Service

REST là viết tắt của REpresentational State Transfer, là một loại kiến trúc phần mềm (architectural style), không phải là một protocol. RESTful web service nhanh vì không có đặc tả nghiêm ngặt như SOAP. Nó chiếm ít băng thông và tài nguyên hơn.

Các thành phần của Web Services

Nền tảng web services cơ bản là XML HTTP. Tất cả các web services chuẩn đều hoạt động bằng các thành phần sau:

WSDL (Web Services Description Language) - WSDL, còn được gọi là Ngôn ngữ Mô tả Dịch vụ Web, là một cách tiêu chuẩn để mô tả các chức năng được cung cấp bởi một dịch vụ web - điều này bao gồm cả SOAP APIs!

Các tệp WSDL hoạt động như một hợp đồng giữa nhà cung cấp dịch vụ web và người sử dụng dịch vụ web, mô tả các phương thức, tham số đầu vào và các kiểu dữ liệu mà dịch vụ web có thể hỗ trợ.

Sự khác biệt giữa SOAP và REST là gì?

SOAP và REST là hai cơ chế trao đổi dữ liệu Internet. Ví dụ: tưởng tượng rằng hệ thống tài khoản nội bộ của bạn chia sẻ dữ liệu với hệ thống kế toán của khách hàng để tự động hóa các tác vụ lập hóa đơn. Hai ứng dụng chia sẻ dữ liệu bằng cách sử dụng một API xác định các quy tắc giao tiếp. SOAP và REST là hai cách tiếp cận khác nhau đối với thiết kế API. Cách tiếp cận SOAP có cấu trúc nghiêm ngặt và sử dụng định dạng dữ liệu XML. REST linh hoạt hơn và cho phép các ứng dụng trao đổi dữ liệu ở nhiều định dạng.

Tham khảo:

- *https://aws.amazon.com/vi/compare/the-difference-between-soap-rest/*

BLOCK, OBJECT & FILE STORAGE

	Object storage	Block storage	File storage
Type of storage	Objects stored in scalable buckets	Fixed-size blocks in a rigid arrangement	Files organized hierarchically in folders and directories
Volume of data	Supports high data volumes	Supports high data volume	Better for lower volumes of data
Data management	Custom metadata provides easy searchability	More limited search and analytics capabilities	Hierarchical structure works well for simpler, smaller datasets
Cost	Pay-as-you-go pricing, more cost-effective	More costly, storage purchased as fixed blocks of storage	More costly, requires purchasing new storage devices to scale out
Performance	Slower performance, longer processing times	**Super low latency and high performance**	Performance impacted by higher data volume
Scalability	Highly scalable	Limited scalability	Limited scalability
Ideal for	Big data storage, static unstructured data, analytics, rich media files, and backups	Transactional, structured data, storage for databases, disks for VMs, and caching	Shared file storage, unstructured data

Source: https://cloud.google.com/discover/object-vs-block-vs-file-storage?hl=en

- **File storage** (Lưu trữ tệp) là khi tất cả dữ liệu được lưu trữ cùng nhau trong một tệp duy nhất với loại phần mở rộng tệp được xác định bởi ứng dụng được sử dụng để tạo tệp hoặc loại tệp, như .jpg, .docx hoặc .txt. Ví dụ, khi bạn lưu một tài liệu trên mạng doanh nghiệp hoặc ổ cứng của máy tính, bạn đang sử dụng lưu trữ tệp. Lưu trữ tệp sử dụng ***cấu trúc phân cấp***, nơi các tệp được tổ chức bởi người dùng trong các thư mục và thư mục con, điều này giúp dễ dàng tìm kiếm và quản lý các tệp. Để truy cập một tệp, người dùng chọn hoặc nhập đường dẫn cho tệp, bao gồm các thư mục con và tên tệp. Hầu hết người dùng quản lý lưu trữ tệp thông qua một hệ thống tệp đơn giản, chẳng hạn như Trình quản lý tệp (File Manager).

- **Block storage** (Lưu trữ khối) là một kiến trúc lưu trữ dữ liệu mà trong đó dữ liệu được chia thành các ***khối có kích thước cố định có thể được đọc và ghi một cách riêng lẻ***. Mỗi khối được gán ***một định danh duy nhất*** và sau đó được lưu trữ trên một máy chủ vật lý. Hệ thống lưu trữ đặt các khối ở bất kỳ đâu mà nó hiệu quả hơn, có nghĩa là các khối có thể được phân tán trên các hệ thống và môi trường khác nhau.

Khi bạn yêu cầu dữ liệu, hệ thống lưu trữ khối sẽ ***tái cấu trúc*** các khối dữ liệu liên quan ***từ bất kỳ đâu chúng được lưu trữ***. Tương tự như lưu trữ đối tượng, lưu trữ khối không phụ thuộc vào một đường dẫn duy nhất đến dữ liệu như lưu trữ tệp. Tuy nhiên, một sự khác biệt quan trọng khi so sánh lưu trữ khối và lưu trữ đối tượng là metadata trong lưu trữ khối bị hạn chế hơn. Bạn chỉ có thể bao gồm các thuộc tính tệp cơ bản, trong khi với lưu trữ đối tượng, bạn có thể tùy chỉnh metadata để bao gồm thông tin chi tiết hơn.

Nhờ vào khả năng kiểm soát chi tiết và tối ưu hóa, lưu trữ khối là lựa chọn tốt cho các khối lượng công việc quan trọng ***cần độ trễ thấp*** và thay đổi thường xuyên.

Giải pháp lưu trữ nào sau đây cho phép nâng cao tốc độ lưu trữ nhưng đồng thời tăng tài nguyên bổ sung (additional overhead)
Block storage
File Storage
File Server
Tất cả các đáp án trên

Trường hợp sử dụng chính của lưu trữ khối (block storage) điện toán đám mây là gì?
Để triển khai cơ sở dữ liệu và ứng dụng có yêu cầu độ trễ thấp khi truy cập dữ liệu
Lưu trữ các tập tin video
Lưu trữ các tập tin sao lưu cho các doanh nghiệp lớn
Quản lý lưu trữ email

Câu nào sau đây mô tả về lưu trữ khối (block storage) trong điện toán điện toán đám mây?
Lưu trữ dữ liệu trong các khối (block), là các đơn vị có thể định địa chỉ riêng lẻ
Nó lưu trữ các tập tin dưới dạng các đối tượng ở định dạng không có cấu trúc
Nó chỉ lưu trữ dữ liệu trong các tập tin
Nó mã hóa tất cả dữ liệu vì mục đích bảo mật

- **Object storage** (Lưu trữ đối tượng) là một kiến trúc lưu trữ dữ liệu nơi dữ liệu được lưu trữ trong các container tách biệt gọi là đối tượng. Lưu trữ dựa trên đối tượng chia dữ liệu thành các đơn vị riêng biệt, mỗi đơn vị chứa một định danh duy nhất và siêu dữ liệu để mô tả dữ liệu, giúp việc truy cập và lấy dữ liệu dễ dàng hơn so với các loại lưu trữ khác.

  Khi so sánh lưu trữ tệp với lưu trữ đối tượng, chẳng hạn, không có cấu trúc thư mục hay thư mục phân cấp. Thay vào đó, các đối tượng được lưu trữ trong một môi trường dữ liệu phẳng, hay còn gọi là storage pool. Các đối tượng có thể được lưu trữ tại chỗ, nhưng thường được lưu trữ trên đám mây, để các tổ chức và nhóm có thể truy cập dữ liệu từ

bất kỳ đâu. Khi bạn muốn truy cập một đối tượng, hệ thống sử dụng ***định danh duy nhất và metadata*** để lấy nó.

Mục đích chính của object storage trong môi trường điện toán đám mây là gì?
Để lưu trữ và quản lý khối lượng lớn dữ liệu phi cấu trúc
Để lưu trữ các bản ghi cơ sở dữ liệu có cấu trúc
Để lưu trữ máy ảo trên điện toán đám mây
Để cung cấp lưu trữ cấp độ tệp (file-level storage) cho các ứng dụng

Mô hình bộ nhớ phẳng này làm cho nó trở nên *lý tưởng cho việc xử lý khối lượng lớn dữ liệu phi cấu trúc, chẳng hạn như nội dung mạng xã hội, video hoặc dữ liệu cảm biến, mà thường khó lưu trữ theo cách phân cấp*. Nó cũng có nghĩa là lưu trữ đối tượng dễ dàng mở rộng hơn nhiều so với các loại lưu trữ khác, vì dữ liệu được tổ chức trong một storage pool toàn cầu duy nhất. Bạn vẫn có thể dễ dàng truy cập và quản lý dữ liệu ngay cả khi nó được lưu trữ trên nhiều thiết bị phần cứng và vị trí địa lý khác nhau.

Tham khảo:

- *https://viettelidc.com.vn/tin-tuc/so-sanh-file-storage-block-storage-va-object-storage*
- *https://aws.amazon.com/compare/the-difference-between-block-file-object-storage/*
- *https://cloud.google.com/discover/object-vs-block-vs-file-storage?hl=en*
- *https://www.ibm.com/think/topics/object-vs-file-vs-block-storage*

GOOGLE FILE SYSTEM

Sự cần thiết: Google File System (GFS), hiện nay được gọi là Colossus, là một hệ thống lưu trữ phân tán được thiết kế và phát triển bởi Google để phục vụ các nhu cầu lưu trữ dữ liệu quy mô lớn trong môi trường tính toán phân tán. Lý do tại sao Google cần GFS có thể được giải thích qua một số điểm chính: **(ChatGPT/Grok3)**

Lưu trữ dữ liệu quy mô lớn

Google hoạt động trên một lượng dữ liệu khổng lồ, từ trang web tìm kiếm, dữ liệu video trên YouTube, đến các dịch vụ khác như Google Maps, Gmail, và Google Drive. Một hệ thống file truyền thống không thể đáp ứng được yêu cầu về hiệu suất và khả năng mở rộng để xử lý khối lượng dữ liệu khổng lồ này. GFS được thiết kế để:

- Lưu trữ dữ liệu lớn: GFS hỗ trợ lưu trữ các file có kích thước cực kỳ lớn (từ GB đến TB), điều này rất quan trọng khi xử lý các dữ liệu như video, hình ảnh và logs.
- Quy mô phân tán: Dữ liệu được phân tán trên nhiều máy chủ để tận dụng khả năng tính toán của các máy tính trong mạng phân tán.

Đảm bảo tính sẵn sàng và độ tin cậy

Trong môi trường phân tán, các sự cố phần cứng (máy chủ hỏng, ổ cứng lỗi) là điều không thể tránh khỏi. GFS đã giải quyết vấn đề này bằng cách:

- Sao lưu và phân tán dữ liệu: Dữ liệu được sao chép trên nhiều máy chủ khác nhau, đảm bảo rằng nếu một máy chủ hỏng, các bản sao của dữ liệu vẫn có thể được truy xuất.
- Tự phục hồi sau sự cố: GFS có cơ chế phát hiện và khôi phục nhanh chóng các lỗi phần cứng, giảm thiểu tối đa sự gián đoạn dịch vụ.

Hiệu suất cao với truy cập dữ liệu đồng thời

Google cần một hệ thống cho phép nhiều người dùng hoặc nhiều ứng dụng có thể truy cập đồng thời vào dữ liệu mà không làm giảm hiệu suất. GFS được thiết kế để:

- Tối ưu hóa truy cập dữ liệu: GFS cho phép các ứng dụng và dịch vụ của Google truy cập dữ liệu với độ trễ thấp và hiệu suất cao, ngay cả khi số lượng yêu cầu đồng thời là rất lớn.
- Phân vùng và tối ưu hóa truy vấn: Các file lớn được chia thành các block nhỏ (128MB hoặc lớn hơn) và có thể được truy cập song song, tối ưu hóa tốc độ xử lý.

Dễ dàng mở rộng

GFS có thể dễ dàng mở rộng theo chiều ngang bằng cách thêm các máy chủ mới vào hệ thống mà không làm gián đoạn quá trình vận hành. Điều này rất quan trọng đối với Google vì nhu cầu lưu trữ của họ không ngừng tăng lên. Hệ thống này được thiết kế để:

- Quy mô dễ dàng: Khi Google cần mở rộng, GFS có thể thêm các nút vào mà không gặp phải vấn đề phức tạp trong việc tích hợp.
- Tiết kiệm chi phí: Việc sử dụng phần cứng giá rẻ và dễ dàng mở rộng giúp Google giảm chi phí đáng kể so với việc sử dụng các giải pháp lưu trữ tốn kém khác.

Đảm bảo tính toàn vẹn của dữ liệu

GFS sử dụng các cơ chế mạnh mẽ để đảm bảo tính toàn vẹn của dữ liệu, đặc biệt trong các môi trường phân tán với hàng triệu file và yêu cầu độ tin cậy cao.

- Kiểm tra và sửa lỗi: GFS có các cơ chế để phát hiện lỗi khi ghi hoặc đọc dữ liệu và thực hiện các biện pháp khôi phục để đảm bảo tính toàn vẹn của dữ liệu.
- Khả năng kiểm tra và khôi phục: Các hệ thống theo dõi và báo cáo lỗi giúp đảm bảo mọi dữ liệu quan trọng luôn được bảo vệ.

Quản lý dữ liệu lớn và phân tích

Google không chỉ lưu trữ dữ liệu mà còn sử dụng GFS để phân tích và xử lý dữ liệu lớn. Các công cụ phân tích dữ liệu như MapReduce (sau này phát triển thành Google Cloud Dataflow) yêu cầu một hệ thống lưu trữ mạnh mẽ và có khả năng xử lý song song, mà GFS đã cung cấp.

Tóm lại

Google cần Google File System (GFS) vì những lý do sau:

- Lưu trữ và quản lý dữ liệu quy mô lớn.
- Đảm bảo tính sẵn sàng và độ tin cậy của dữ liệu trong môi trường phân tán.
- Tối ưu hóa hiệu suất truy cập đồng thời và khả năng mở rộng hệ thống.

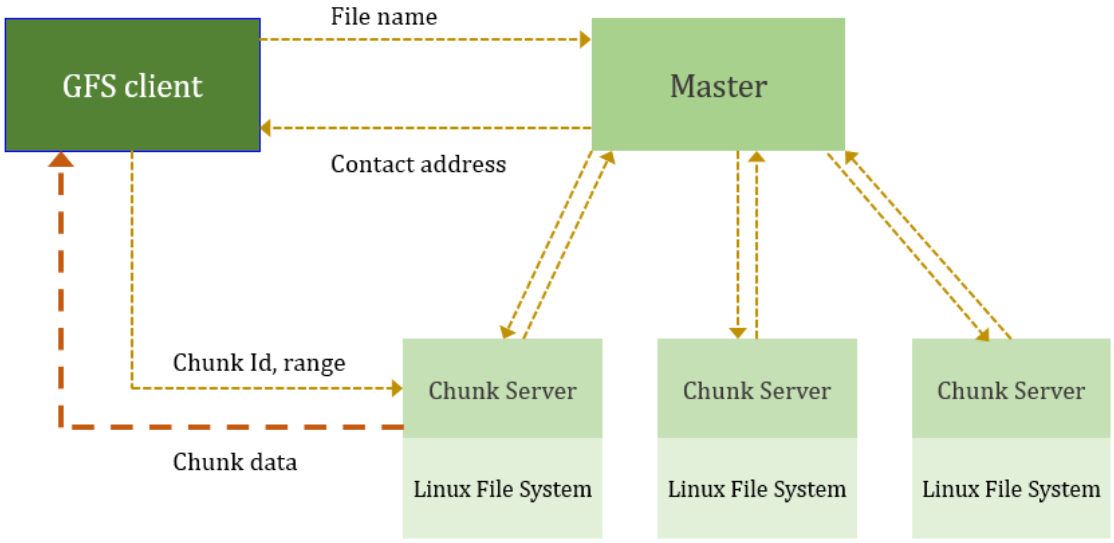
- Bảo vệ và duy trì tính toàn vẹn của dữ liệu.
- Hỗ trợ các ứng dụng và công cụ phân tích dữ liệu quy mô lớn.

GFS đã đóng vai trò then chốt trong sự thành công của Google, giúp công ty duy trì khả năng xử lý khối lượng dữ liệu cực kỳ lớn trong các ứng dụng của mình.

- Google File System đạt được thông lượng cao cho các tệp lớn bằng cách chia nhỏ các tệp thành nhiều phần (chunks) và phân phối chúng trên nhiều máy/cụm
- Mục đích chính của Google File System (GFS) để quản lý và lưu trữ các tập dữ liệu lớn trên các hệ thống phân tán

Thiết kế và hoạt động:

Master-Worker Architecture: Kiến trúc master-worker giống như một hệ thống điều khiển trung tâm (master) giám sát một đội công nhân (worker nodes). Master node quản lý thông tin quan trọng về hệ thống file, như nơi lưu trữ các files, quyền truy cập và cách chúng được sao chép. Worker nodes chịu trách nhiệm lưu trữ và truy xuất data. Việc chỉ sử dụng master node để lưu trữ các metadata mà không read/write trực tiếp trên master node giúp giảm thiểu bottleneck (nghẽn cổ chai) ở master.



Tham khảo:

- *https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf*
- *https://brabalawuka.cc/posts/study/gfs/*

MAP REDUCE

MapReduce là một mô hình được Google phát triển độc quyền, nhằm mục đích xử lý các dữ liệu lớn theo hướng phân tán và song song thuật toán trong một cụm máy tính. Đầu tiên, dữ liệu được chia nhỏ và sau đó được kết hợp để tạo ra kết quả cuối cùng. Các thư viện cho MapReduce được viết bằng rất nhiều ngôn ngữ lập trình với nhiều tối ưu hóa khác nhau.

- MapReduce được thiết kế để xử lý khối lượng dữ liệu lớn song song bằng cách chia công việc thành một tập hợp các tác vụ độc lập.
- MapReduce là lớp xử lý dữ liệu của Hadoop

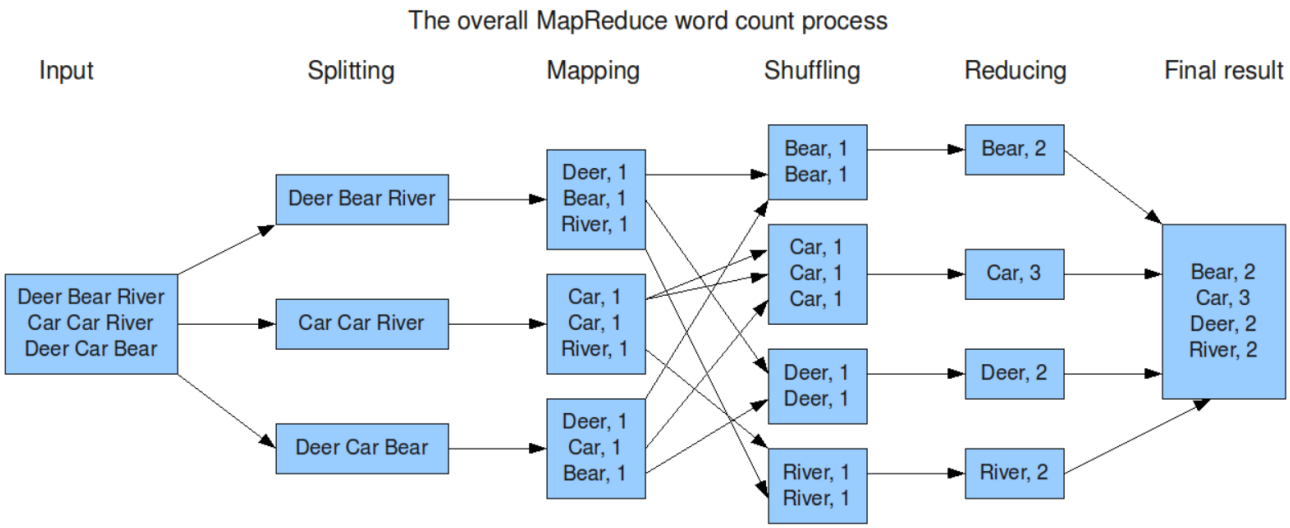
MapReduce bao gồm 2 thủ tục chính:

- **Thủ tục Map:** có vai trò lọc và phân loại dữ liệu. Thời gian cần để thủ tục Map thực hiện xong việc được giao chủ yếu *phụ thuộc vào vị trí (location) của các blocks cần thiết cho thủ tục Map.* **Số lượng Maps *thông thường sẽ bằng số lượng inputs (hình dưới).***
- **Thủ tục Reduce:** tổng hợp dữ liệu. *Hàm Reduce có trách nhiệm hợp nhất các kết quả do từng hàm/nhiệm vụ Map() tạo ra*

Ví dụ WordCount - WordCount là một bài toán kinh điển để minh họa cho MapReduce

Ví dụ bây giờ chúng ta cần thực hiện 1 task là đếm số lần xuất hiện của mỗi từ trong 1 văn bản. Đầu tiên chúng ta sẽ chia tách văn bản đó thành các dòng, mỗi dòng sẽ được đánh 1 số thứ tự. Đầu vào của hàm Map sẽ là các cặp key/value chính là số thứ tự dòng / đoạn văn bản trên dòng đó.

Trong Map chúng ta sẽ xử lý tách từng từ trong 1 dòng ra và gán cho chúng giá trị tần suất suất hiện ban đầu là 1. Sẽ có một tiến trình ở giữa giúp việc gộp các output đầu ra của Map có cùng key với nhau thành 1 mảng các giá trị. Đầu vào của Reduce sẽ là (cặp key/value) là (từ / tần suất xuất hiện của từ đó). Trong Reduce chúng ta chỉ cần thực hiện cộng các giá trị trong mảng và đưa ra kết quả chính là số lần xuất hiện của từng từ trong văn bản đầu vào.



- **Shuffle & Sort xảy ra đồng thời**
- Đầu ra được sắp xếp của Mapper là Đầu vào cho Reducer

Lớp nào sau đây ánh xạ cặp khóa/giá trị đầu vào thành một tập hợp các cặp khóa/giá trị trung gian?
Mapper
Cả Mapper & Reducer
Reducer
Không có đáp án đúng

Số lượng hàm map cần thiết thường được đưa ra bởi tổng số lượng của
Inputs
Tasks
Outputs
Không có đáp án đúng

Trình theo dõi công việc và trình theo dõi tác vụ xử lý MapReduce như thế nào:

- **Job Tracker:** Công việc của Job Tracker là quản lý tất cả các tài nguyên và tất cả các công việc trên toàn cụm và cũng lên lịch cho từng bản đồ trên Task Tracker chạy trên cùng một nút dữ liệu vì có thể có hàng trăm nút dữ liệu có sẵn trong cụm.
- **Task Tracker:** *Task Tracker có thể được coi là các slave thực sự đang làm việc theo lệnh do Job Tracker đưa ra. Task Tracker này được triển khai trên mỗi node có sẵn trong cụm thực hiện tác vụ Map và Reduce theo lệnh của Job Tracker.*

Một node _____ hoạt động như một Slave và chịu trách nhiệm thực hiện một Nhiệm vụ được giao cho nó bởi JobTracker
TaskTracker
MapReduce
Mapper
JobTracker

Hãy xem xét chương trình WordCount của Hadoop: đối với một văn bản nhất định, hãy tính tần suất của từng từ trong đó. Đầu vào được đọc theo từng dòng. Khi nhập, bạn được cung cấp một tệp chứa một dòng văn bản duy nhất: “A Ram Sam Sam”. Có bao nhiêu đối tượng

Mapper và đối tượng Reducer được tạo? Có bao nhiêu lệnh gọi đến map() và reduce() được thực hiện?
1 đối tượng Mapper, 1 đối tượng Reducer, 1 lệnh gọi map(), 3 lệnh gọi reduce()
3 đối tượng Mapper, 1 đối tượng Reducer, 3 lệnh gọi map(), 1 lệnh gọi reduce()
3 đối tượng Mapper, 3 đối tượng Reducer, 1 lệnh gọi map(), 1 lệnh gọi reduce()
1 đối tượng Mapper, 3 đối tượng Reducer, 3 lệnh gọi map(), 3 lệnh gọi reduce()

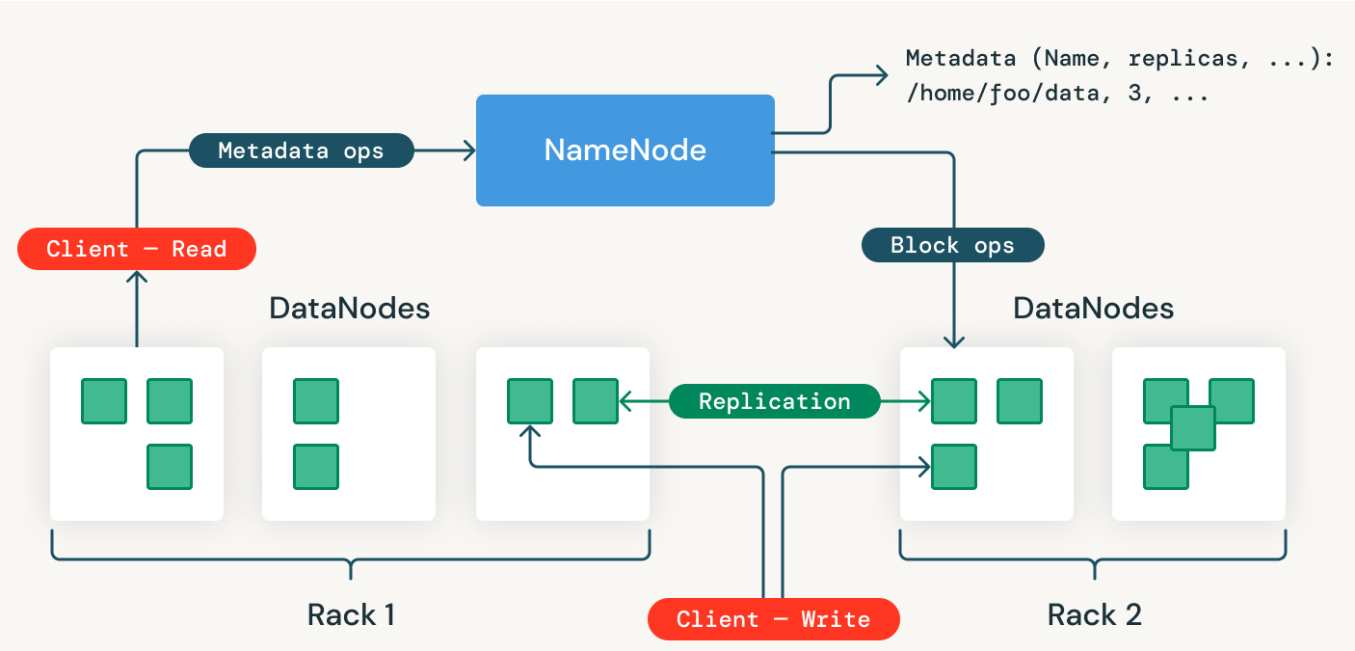
Trong các tình huống sau đây, tình huống nào cần xử lý theo thời gian thực?
Người quản trị điện toán đám mây tại Netflix phải được thông báo khi hơn 1% máy chủ của họ ghi lại lượng lỗi bất thường để có thể ngăn chặn sự cố dẫn đến ngừng hoạt động (và do đó gây ra tổn thất lớn về chi phí).
Một ngân hàng muốn tổng hợp số tiền tiết kiệm của tất cả khách hàng trên các tài khoản ngân hàng cá nhân để cung cấp thông tin này cho cơ quan thuế.
Twitter nhận được hàng nghìn tin nhắn mỗi giây. Twitter muốn tính toán có bao nhiêu tin nhắn được đăng vào năm ngoái bởi tất cả người dùng ở Hoa Kỳ.
Mỗi thiết bị Android sẽ gửi thông tin sử dụng đến Google theo định kỳ để xác định mức độ phân mảnh hệ điều hành của thiết bị (tức là: bao nhiêu phần trăm thiết bị vẫn đang chạy Android & phiên bản Android nào?)

Tham khảo:

- *https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf*

HADOOP

Với HDFS, dữ liệu được ghi trên 1 máy chủ và có thể đọc lại nhiều lần sau đó tại bất cứ máy chủ khác trong cụm HDFS. HDFS bao gồm 1 Namenode chính và nhiều Datanode kết nối lại thành một cụm (cluster).
Namenode



HDFS chỉ bao gồm duy nhất 1 NameNode được gọi là master node thực hiện các nhiệm vụ:

- ***Lưu trữ metadata của dữ liệu thực tế*** *(tên, đường dẫn, blocks id, cấu hình datanode vị trí blocks,etc),* ***không lưu trữ dữ liệu thực tế - Tên tệp, vị trí khối***
- Quản lý không gian tên của hệ thống file (ánh xạ các file name với các blocks, ánh xạ các block vào các datanode)
- Quản lý cấu hình của cụm
- Chỉ định công việc cho datanode
- *Nhật ký hoạt động (operation log) trong Hadoop là cần thiết để đảm bảo rằng NameNode có thể khôi phục khi có sự cố xảy ra.*

Mục đích của các phương pháp setup/cleanup trong tác vụ Hadoop là gì?
Cho phép cả mappers & reducers khởi tạo tài nguyên.
Cho phép trình kết hợp khởi tạo tài nguyên.
Cho phép mappers & reducers thực thi một số mã trước mỗi lệnh gọi map()/reduce().
Để cấu hình tác vụ Hadoop.

DataNode

Chức năng của **DataNode**

- ***Lưu trữ dữ liệu thực tế***
- Trực tiếp thực hiện và xử lý công việc (đọc/ghi dữ liệu)
- *Read and write requests send Clients directly to the Datanodes → ko thông qua Namenode (hình trên)*

Trong các thao tác sau, thao tác nào KHÔNG cần giao tiếp với NameNode?
Một máy khách đang đọc một khối dữ liệu (block of data) từ cụm (cluster)
Một máy khách đang ghi tệp vào HDFS.
Một máy khách yêu cầu tên tệp của một khối dữ liệu nhất định.
Một máy khách đang đọc tệp từ cụm.

- **Replication factor**: HDFS sử dụng replication factor để đảm bảo tính tin cậy của dữ liệu. Khi một DataNode bị hỏng, HDFS sẽ tăng replication factor lên để tạo ra thêm các bản sao của các khối dữ liệu trên các DataNode khác nhau để đảm bảo tính tin cậy của dữ liệu. ***Hệ số sao chép (replication factor) mặc định cho HDFS là 3***
- **Heartbeat**: HDFS sử dụng cơ chế heartbeat để kiểm tra tính khả dụng của các DataNode. Khi một DataNode không phản hồi trong khoảng thời gian quy định, HDFS sẽ coi DataNode đó là bị hỏng và thực hiện các cơ chế xử lý lỗi như đã nêu trên. ***Khoảng thời gian phản hồi default (gửi tin heartbeat mặc định) là 3 giây*** *cho đến dưới 10 phút.*

Một số lệnh cơ bản của HDFS (https://demanejar.github.io/posts/hdfs-commands/):

- Mkdir - Tạo folder trong HDFS
- *Put - Sử dụng lệnh put để copy 1 file từ thư mục local vào HDFS*
- Get - Sử dụng lệnh để lấy 1 file từ hdfs về thư mục local

Tham khảo:

- *https://www.cl.cam.ac.uk/teaching/1920/CloudComp/lectures/storage-systems.pdf*

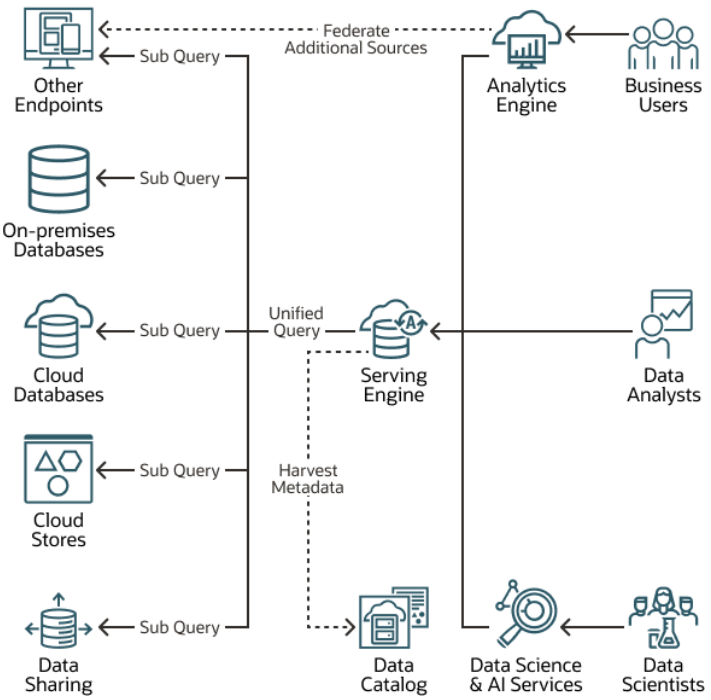
DATA FEDERATION

Liên kết dữ liệu (Data federation) là một kỹ thuật cho phép tích hợp, hợp nhất và quản trị dữ liệu được lưu trữ tại các kho dữ liệu khác nhau bằng cách sử dụng một công cụ truy vấn liên kết (federated query engine). *Công cụ này sẽ chuyển một truy vấn duy nhất thành nhiều truy vấn con, gửi đến từng kho dữ liệu nguồn tương ứng. Sau đó, kết quả từ các truy vấn con sẽ được hợp nhất và trình bày cho người dùng hoặc ứng dụng*

Ví dụ:

- Dữ liệu khách hàng trong MySQL
- Dữ liệu đơn hàng trong PostgreSQL
- Dữ liệu kho trong một API ngoài

Với data federation, có thể viết 1 câu truy vấn SQL như thể tất cả dữ liệu đều nằm trong cùng một bảng – hệ thống sẽ tự động kết nối các nguồn, xử lý truy vấn và trả kết quả.



Liên hợp dữ liệu (data federation) là gì trong bối cảnh quản lý dữ liệu điện toán đám mây?
Kết hợp dữ liệu từ nhiều nguồn khác nhau mà không cần di chuyển dữ liệu về mặt vật lý
Chỉ lưu trữ dữ liệu trong cơ sở dữ liệu trung tâm
Sao lưu dữ liệu vào nhiều hệ thống lưu trữ
Mã hóa dữ liệu vì mục đích bảo mật

Dịch vụ điện toán đám mây nào được tối ưu hóa cho phân tích dữ liệu thời gian thực và xử lý dữ liệu lớn?
Amazon Redshift
Amazon S3
Google Cloud Pub/Sub
Microsoft Azure Blob Storage

Trong các ví dụ sau đây, ví dụ nào là ví dụ về kho dữ liệu điện toán đám mây (cloud data warehouse)?
Google BigQuery
Amazon S3
Dropbox
Microsoft Excel

CHƯƠNG IV – CLOUD ARCHITECTURE

- Định nghĩa kiến trúc điện toán đám mây: cách thức các dịch vụ điện toán đám mây được tổ chức, triển khai và quản lý.
- Một số thành phần cốt lõi của kiến trúc điện toán đám mây: storage/Networking/Tài nguyên tính toán (compute resources)
- Runtime environment trong môi trường điện toán đám mây (cloud) là môi trường thực thi nơi mã nguồn của ứng dụng được chạy, quản lý và vận hành. Mục đích chính của runtime environment trong điện toán điện toán đám mây là Để quản lý và triển khai các ứng dụng hoặc dịch vụ trong thời gian chạy (runtime)
- Lớp điều khiển điện toán đám mây (cloud control plane) quản lý triển khai và quản lý vòng đời của các dịch vụ điện toán đám mây
- Cloud API: Cloud API (Application Programming Interface) là giao diện lập trình ứng dụng cho phép các ứng dụng hoặc dịch vụ bên ngoài (dịch vụ của bên thứ ba) tương tác với các dịch vụ, ứng dụng điện toán đám mây.

Trong kiến trúc điện toán đám mây, công nghệ nào sau đây cho phép mở rộng nhanh chóng các dịch vụ backend?
Các công cụ chứa và điều phối (Containers & orchestration tools/ví dụ: Docker, Kubernetes)
Nâng cấp phần cứng vật lý
Máy ảo đơn lẻ
Thiết kế ứng dụng nguyên khối (monolithic)

MÔ HÌNH TRIỂN KHAI

Private cloud

Đám mây riêng là môi trường điện toán đám mây dành riêng cho một tổ chức. Bất kỳ cơ sở hạ tầng đám mây nào cũng có tài nguyên điện toán cơ bản như CPU và kho lưu trữ mà bạn cung cấp theo yêu cầu thông qua cổng thông tin tự phục vụ. Trong một đám mây riêng, tất cả các tài nguyên được tách biệt và thuộc quyền kiểm soát của một tổ chức. Do đó, đám mây riêng còn được gọi là đám mây nội bộ hoặc đám mây của doanh nghiệp. *Đám mây riêng có thể ở trong hoặc ngoài cơ sở (on- or off-premises). Private cloud cho phép tài nguyên điện toán đám mây được sử dụng độc quyền/duy nhất (exclusive) bởi một tổ chức/doanh nghiệp*

Public cloud

Nhà cung cấp dịch vụ đám mây sẽ quản lý các tài nguyên điện toán cơ sở. Nhà cung cấp chịu trách nhiệm duy trì tài nguyên và đảm bảo sự sẵn sàng, độ tin cậy và bảo mật thông qua thỏa thuận mức dịch vụ (SLA). Lợi thế của mô hình triển khai pubic cloud (điện toán đám mây công cộng) *là cung cấp các giải pháp có chi phí thấp, có khả năng mở rộng với các tài nguyên được chia sẻ cho nhiều bên thuê dịch vụ (tenants)*

Hybrid cloud (Đám mây hỗn hợp)

Đám mây hỗn hợp là *thiết kế cơ sở hạ tầng CNTT tích hợp liền mạch đám mây riêng tư (private cloud) và đám mây công khai (public cloud) của một tổ chức*. Do đó, bạn có thể thoải mái lưu trữ dữ liệu và chạy ứng dụng trên nhiều môi trường. Môi trường đám mây hỗn hợp giúp hợp nhất cơ sở hạ tầng của bạn, nhờ đó bạn có thể cung cấp, điều chỉnh quy mô và quản lý tập trung mọi tài nguyên điện toán.

Multi-cloud (Kiến trúc đa đám mây)

Điện toán đa đám mây (Multi-cloud) là việc sử dụng dịch vụ đám mây từ nhiều nhà cung cấp dịch vụ cloud khác nhau. Điều này mang lại cho tổ chức sự linh hoạt trong việc tối ưu hiệu suất, kiểm soát chi phí và tránh bị phụ thuộc vào một nhà cung cấp duy nhất.

Multi-cloud có thể đơn giản là việc sử dụng các phần mềm dạng dịch vụ (SaaS) từ các nhà cung cấp khác nhau, ví dụ như Gmail hoặc Outlook. Tuy nhiên, trong môi trường doanh nghiệp, multi-cloud thường đề cập đến việc vận hành các ứng dụng doanh nghiệp trên nền tảng dạng dịch vụ (PaaS) hoặc hạ tầng dạng dịch vụ (IaaS) từ nhiều nhà cung cấp dịch vụ đám mây như Amazon Web Services (AWS), Google Cloud Platform, IBM Cloud® và Microsoft Azure.

Tham khảo:

- *https://aws.amazon.com/vi/what-is/private-cloud/*
- *https://www.ibm.com/think/topics*

MÔ HÌNH DỊCH VỤ

IaaS

IaaS bao gồm các khối dựng cơ bản dành cho CNTT đám mây và thường cung cấp quyền truy cập vào các tính năng kết nối mạng, máy tính (phần cứng ảo hoặc trên phần cứng chuyên dụng) và không gian kho lưu trữ dữ liệu. Một cách ngắn gọn, IaaS cung cấp tài nguyên điện toán (ảo hóa).

PaaS

PaaS giúp các tổ chức loại bỏ nhu cầu quản lý cơ sở hạ tầng cơ bản (thường là phần cứng và hệ điều hành) và việc tích hợp này cho phép bạn tập trung vào công tác triển khai cũng như quản lý các ứng dụng của mình. PaaS cung cấp các công cụ và frameworks phát triển, sản xuất ứng dụng

SaaS

Nhà cung cấp SaaS cung cấp cho bạn các ứng dụng phần mềm do chính nhà cung cấp vận hành và quản lý. Trong hầu hết các trường hợp, khi nhắc đến SaaS, mọi người thường nghĩ đến ứng dụng dành cho người dùng cuối của bên thứ ba. Với dịch vụ của SaaS, bạn không phải lo lắng về việc duy trì dịch vụ hay quản lý cơ sở hạ tầng cơ bản; bạn sẽ chỉ cần suy nghĩ về việc mình sẽ sử dụng phần mềm cụ thể đó như thế nào. Một ví dụ thường thấy của ứng dụng SaaS là email trên nền tảng web: bạn có thể gửi và nhận email mà không phải quản lý việc bổ sung tính năng hay bảo trì máy chủ và hệ điều hành dùng để vận hành chương trình email.

Serverless architecture

Serverless computing là một mô hình phát triển và vận hành ứng dụng cho phép lập trình viên xây dựng và chạy mã ứng dụng *mà không cần cung cấp hoặc quản lý máy chủ hay hạ tầng phụ trợ.*

Serverless ("không máy chủ") không có nghĩa là "không có máy chủ - server". Trong Serverless, các máy chủ vẫn tồn tại nhưng được quản lý bởi nhà cung cấp dịch vụ đám mây (CSP). Serverless mô tả trải nghiệm của lập trình viên với các máy chủ đó—chúng hoàn toàn ẩn khỏi lập trình viên, lập trình viên không cần sở hữu/ nhìn thấy/quản lý hay tương tác với các máy chủ (servers) theo bất kỳ cách nào.

Với điện toán Serverless, lập trình viên có thể tập trung vào việc viết mã (code) giao diện người dùng và logic nghiệp vụ một cách tốt nhất. Tất cả những gì họ cần làm là viết mã ứng dụng và triển khai nó lên các nền tảng do CSP quản lý.

Nhà cung cấp dịch vụ đám mây sẽ xử lý mọi việc còn lại—cung cấp hạ tầng cần thiết để chạy mã, mở rộng hoặc thu hẹp hạ tầng theo nhu cầu thực tế—đồng thời chịu trách nhiệm quản lý và bảo trì hạ tầng định kỳ, như cập nhật và vá hệ điều hành, quản lý bảo mật, lập kế hoạch dung lượng, giám sát hệ thống, và nhiều việc khác.

Hơn nữa, với điện toán Serverless, lập trình viên *không bao giờ phải trả tiền cho tài nguyên không được sử dụng*. Nhà cung cấp đám mây sẽ khởi tạo và cấp phát tài nguyên tính toán cần thiết theo yêu cầu khi mã (code) được thực thi, và thu hồi chúng lại (scaling up/scaling down). *Việc tính phí bắt đầu khi mã được chạy và kết thúc khi mã (code) dừng lại*; thông thường, *chi phí được tính dựa trên thời gian thực thi và tài nguyên sử dụng*.

Serverless là kiến trúc phù hợp nhất cho xử lý dữ liệu từ các thiết bị IoT

Kiến trúc không có máy chủ (Serverless) tập trung vào việc cung cấp các dịch vụ quản trị trọn gói (managed services) để tăng khả năng mở rộng và hiệu quả về chi phí

Trong serverless runtime environment, điều gì xảy ra khi một hàm (chức năng – function) hoặc dịch vụ được gọi?
Chức năng/dịch vụ được thực hiện ngay lập tức và tài nguyên được phân bổ tự động
Một máy ảo được khởi động để chạy chức năng/dịch vụ
Chức năng được thực hiện trên một số lượng cố định các máy chủ chuyên dụng
Một container được triển khai và duy trì để thực hiện chức năng

FaaS

Function as a Service (FaaS) là một dịch vụ điện toán đám mây cho phép khách hàng chạy mã nguồn để phản hồi các sự kiện, mà không cần quản lý hạ tầng phức tạp vốn thường đi kèm với việc xây dựng và triển khai các ứng dụng microservices.

Thông thường, việc lưu trữ một ứng dụng phần mềm trên internet đòi hỏi phải cấp phát và quản lý máy chủ ảo hoặc vật lý, cùng với việc quản lý hệ điều hành và các tiến trình lưu trữ web. Với FaaS, phần cứng vật lý, hệ điều hành máy ảo và phần mềm máy chủ web đều được nhà cung cấp dịch vụ đám mây tự động quản lý. Tính năng này cho phép các lập trình viên tập trung hoàn toàn vào từng hàm riêng lẻ trong mã ứng dụng của họ.

- Lợi ích chính của FaaS: Các nhà phát triển ứng dụng chỉ cần quản lý các hàm (chức năng – function) mà không cần lo lắng về cơ sở hạ tầng cơ bản.
- Điều gì kích hoạt việc thực thi một chức năng trong FaaS: thời gian hoặc sự kiện đã lên lịch (chẳng hạn như yêu cầu HTTP, tải tệp lên hoặc thay đổi cơ sở dữ liệu)

Ví dụ

- IaaS (Infrastructure as a Service): AWS EC2, Microsoft Azure Virtual Machines, Google Compute Engine
- PaaS (Platform as a Service): AWS Elastic Beanstalk, Azure App Service, Google App Engine
- SaaS (Software as a Service): Microsoft Office 365, Google Workspace, Salesforce
- Serverless: AWS Fargate, Google Cloud Functions, Microsoft Azure Functions, Cloudflare Workers
- FaaS: AWS Lambda, Google Cloud Platform

Tham khảo:

- *https://aws.amazon.com/vi/types-of-cloud-computing/*
- *https://genezio.com/deployment-platform/blog/best-serverless-platforms-providers/#list-of-10-best-serverless-platforms*
- *https://www.ibm.com/think/topics*

CLOUD MIGRATION

Di chuyển lên đám mây (Cloud migration) là *quá trình chuyển dữ liệu, ứng dụng, các hệ thống phục vụ kinh doanh và các khối lượng công việc khác từ hạ tầng CNTT/trung tâm dữ liệu tại chỗ (on-premises) lên hạ tầng đám mây*, hoặc từ môi trường đám mây này sang môi trường đám mây khác — được gọi là di chuyển giữa các đám mây (cloud-to-cloud migration).

- Thách thức khi di chuyển dữ liệu (data migration) trong lưu trữ điện toán đám mây: di chuyển dữ liệu từ hệ thống lưu trữ on-premise lên điện toán đám mây mà không gây ra thời gian gián đoạn dịch vụ.

Một công ty/doanh nghiệp có thể di chuyển lên một đám mây duy nhất hoặc nhiều đám mây. Họ có thể sử dụng mô hình đám mây công cộng, nơi các dịch vụ được cung cấp qua internet công cộng, hoặc mô hình đám mây riêng, với hạ tầng đám mây bảo mật và độc quyền chỉ mình họ có quyền truy cập. Nhiều tổ chức lựa chọn môi trường đám mây lai (hybrid cloud), kết hợp dịch vụ của cả đám mây công cộng và đám mây riêng để tạo ra một hạ tầng CNTT linh hoạt,

hiệu quả về chi phí, hỗ trợ và tự động hóa việc quản lý khối lượng công việc giữa các môi trường đám mây.

- Lý do phổ biến khiến các doanh nghiệp chuyển sang sử dụng dịch vụ điện toán đám mây là do khả năng mở rộng và tính linh hoạt cũng như hiệu quả các nguồn lực CNTT được cải thiện. Thách thức chính khi di chuyển các ứng dụng cũ lên điện toán đám mây là sự phức tạp của việc tích hợp các hệ thống cũ với công nghệ điện toán đám mây hiện đại
- Trước khi di chuyển lên điện toán đám mây (cloud migration), doanh nghiệp/công ty/tổ chức cần phải có các bước đánh giá, phân tích nhu cầu, xác định các ứng dụng đã sẵn sàng cho điện toán đám mây, đồng thời cần lập dự toán ngân sách (lên tổng chi phí cho việc di chuyển & duy trì dịch vụ cloud sau di chuyển) & xác định thời gian triển khai dự án.
- Cloud migration assessment: đánh giá cơ sở hạ tầng và ứng dụng hiện có để xác định chiến lược chuyển lên điện toán đám mây tốt nhất. Khi lựa chọn công cụ chuyển lên đám mây, điều quan trọng cần được cân nhắc là khả năng tích hợp với các hệ thống CNTT hiện có
- Sau khi di chuyển lên điện toán đám mây, việc kiểm tra các ứng dụng có đang hoạt động như mong đợi trong môi trường điện toán đám mây hay không sẽ là 1 trong những hoạt động chính, cần thực hiện.
- Bên cạnh đó, so sánh tổng chi phí sở hữu (TCO) trước và sau khi di chuyển (cloud migration) là 1 trong những chỉ số quan trọng để đánh giá việc di chuyển có thành công hay không.
- Cloud Migration Factory: chiến lược sử dụng các công cụ và quy trình tự động để di chuyển khối lượng lớn dữ liệu và ứng dụng lên điện toán đám mây
- Pilot migration: di chuyển một tập hợp nhỏ các ứng dụng để kiểm tra chất lượng quy trình trước khi di chuyển toàn bộ.
- Phương pháp hay được sử dụng cho việc di chuyển lên điện toán đám mây là cập nhật và kiểm tra thường xuyên kế hoạch dự án di chuyển lên điện toán đám mây

Các chiến lược di chuyển (migration)

- *Rehosting (Tái lưu trữ - "Lift and Shift"):* Đây là cách tiếp cận đơn giản nhất, trong đó team IT *"nhấc" toàn bộ ứng dụng hoặc hệ thống từ hạ tầng tại chỗ (on-premises) và "đặt" lên đám mây mà không cần thay đổi đáng kể mã nguồn hoặc kiến trúc*. Ví dụ: Chuyển một máy ảo (VM) từ trung tâm dữ liệu nội bộ sang AWS EC2 (IaaS)
- *Replatforming (Tái nền tảng*): *Chuyển ứng dụng lên đám mây đồng thời sửa đổi ứng dụng để tận dụng các điểm mạnh của điện toán đám mây.*
- *Repurchasing (Thay thế):* Thay vì di chuyển hệ thống cũ, bạn mua một giải pháp đám mây sẵn có (thường là SaaS - Software as a Service) để thay thế. Ví dụ: Thay thế hệ thống CRM nội bộ bằng Salesforce, SAP, Oracle.
- *Re-architecting (Tái cấu trúc):* Hay còn gọi là **Refactoring** *thiết kế lại hoặc viết lại ứng dụng để tận dụng tối đa các tính năng của đám mây, như kiến trúc microservices, serverless, hoặc tự động mở rộng và tính đàn hồi.* Ví dụ: Chuyển một ứng dụng monolith sang kiến trúc microservices
- *Retaining:* giữ nguyên một số ứng dụng, dữ liệu hoặc khối lượng công việc ở lại hạ tầng tại chỗ (on-premises) và không di chuyển lên đám mây. Ví dụ: Một hệ thống tài chính

kế toán cũ chạy trên máy chủ vật lý nội bộ, có nhiều tích hợp sâu và chi phí/nguồn lực di chuyển lên cloud cao, có thể được "retain" lại và duy trì tại chỗ trong khi các hệ thống khác được di chuyển lên cloud.

- *Retiring (Loại bỏ):* Trong quá trình đánh giá, nếu một số hệ thống hoặc ứng dụng không còn cần thiết, có thể loại bỏ chúng thay vì di chuyển lên đám mây

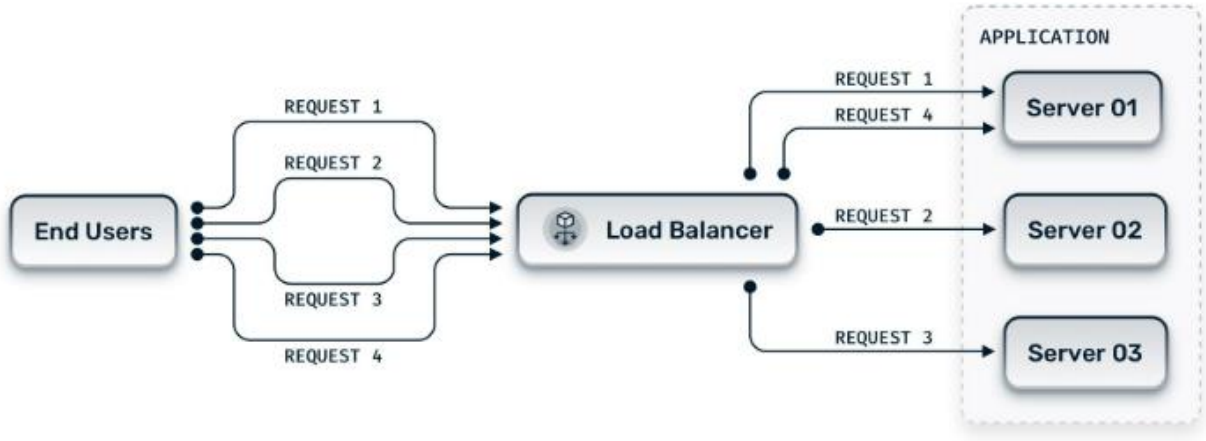
Live migration trong môi trường ảo hóa là gì?
Di chuyển máy ảo giữa các máy chủ vật lý khác nhau mà không bị ngừng hoạt động
Di chuyển máy ảo đến vị trí lưu trữ mới mà không cần tắt máy
Di chuyển máy vật lý sang máy ảo
Chuyển máy ảo sang điện toán đám mây công cộng

CLOUD DELIVERY

Cloud load balancing

Cân bằng tải (Load balancing) là quá trình phân phối lưu lượng mạng một cách hiệu quả giữa nhiều máy chủ nhằm tối ưu hóa khả năng sẵn sàng của ứng dụng và đảm bảo trải nghiệm tích cực cho người dùng cuối.

Vì các trang web có lưu lượng truy cập cao và các ứng dụng điện toán đám mây nhận hàng triệu yêu cầu (requests) từ người dùng mỗi ngày, nên cân bằng tải là một năng lực thiết yếu trong việc phân phối ứng dụng hiện đại. Ví dụ, các trang thương mại điện tử phụ thuộc vào cân bằng tải để đảm bảo rằng các ứng dụng web có thể truyền tải dữ liệu, hình ảnh, video và thông tin giá cả từ máy chủ web đến người tiêu dùng mà không bị chậm trễ hay gián đoạn.



Giải pháp nào sau đây thường được sử dụng ở phần backend trong kiến trúc điện toán đám mây để xử lý quy mô yêu cầu lớn (large-scale requests)?

Bộ cân bằng tải
Content Delivery Network (CDN)
Định tuyến phía máy khách hàng
Tăng năng lực hệ thống máy chủ

Content Delivery Network (CDN)

Mạng phân phối nội dung (CDN), viết tắt của Content Delivery Network, *là một hệ thống các máy chủ phân tán được đặt ở nhiều vị trí địa lý khác nhau, hoạt động cùng nhau để phân phối, cung cấp nội dung số (như trang web, video, hình ảnh, tệp tin) đến người dùng một cách nhanh chóng và hiệu quả. CDN giúp giảm độ trễ, tăng tốc độ tải và cải thiện trải nghiệm người dùng (end-user) bằng cách đưa nội dung đến gần hơn với vị trí của họ.*

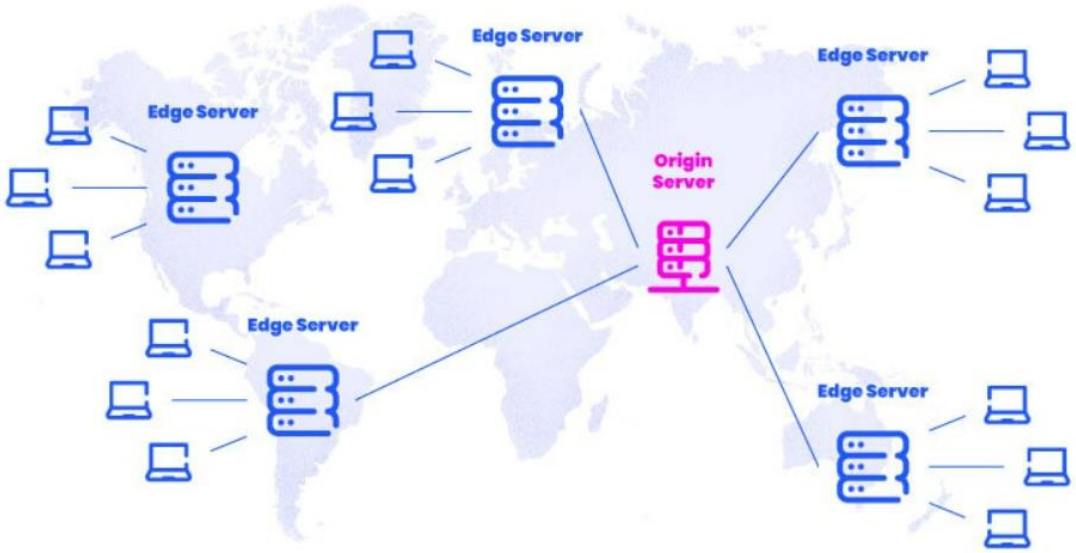
Bộ nhớ đệm (cache)

Cache trong CDN (Content Delivery Network) là bộ nhớ tạm thời được sử dụng để *lưu trữ các nội dung tĩnh (như hình ảnh, video, file CSS/JS, HTML, etc) gần người dùng cuối & được truy cập thường xuyên*, nhằm giảm thời gian tải trang, giảm tải máy chủ gốc và tiết kiệm băng thông.

Lợi thế chính của việc sử dụng CDN trên nền tảng điện toán đám mây để phát trực tuyến video là giảm thiểu buffering và truyền phát nhanh hơn bằng cách lưu trữ (cache) video tại các máy chủ biên (edge server) gần người dùng (end-user) hơn

Máy chủ biên (edge servers) là thành phần điển hình trong kiến trúc CDN, cho phép phân phối, lưu trữ nội dung (content) gần người dùng cuối (end-user) hơn.

Trong kiến trúc CDN, các yêu cầu từ người dùng cuối (end-user) được định tuyến đến máy chủ biên theo vị trí địa lý của người dùng (end-user)

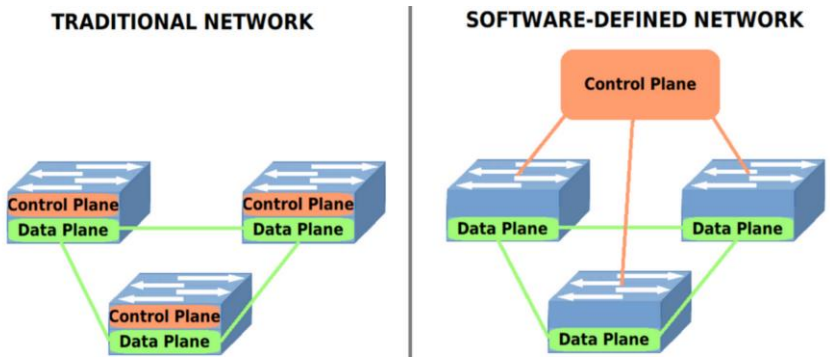


Software-defined networking (SDN)

Software-defined networking – SDN là một kiến trúc mạng được điều khiển bằng phần mềm, dựa trên các giao diện lập trình ứng dụng (API). SDN tận dụng một nền tảng tập trung để giao tiếp với hạ tầng CNTT và điều hướng lưu lượng mạng.

Mục đích chính của SDN (Software-Defined Networking) trong điện toán đám mây là cung cấp khả năng kiểm soát tập trung và tự động hóa lưu lượng mạng, bằng cách tập trung hóa lớp điều khiển (control plane) và tách rời lớp dữ liệu (data plane)

Hiện nay, các doanh nghiệp đang tìm đến SDN để mang lại các lợi ích của điện toán đám mây vào quản lý và triển khai mạng. Với ảo hóa mạng, tổ chức có thể đạt hiệu quả cao hơn nhờ vào các công cụ và công nghệ mới như phần mềm dưới dạng dịch vụ (SaaS), hạ tầng dưới dạng dịch vụ (IaaS) và các dịch vụ điện toán đám mây khác, cũng như tích hợp với mạng được định nghĩa bằng phần mềm thông qua API. *SDN cho phép nhiều môi trường điện toán đám mây kết nối với nhau một cách an toàn.*



Lợi ích chính của việc kiểm soát mạng tập trung trong SDN cho môi trường điện toán đám mây là cấu hình và quản lý mạng đơn giản, cho phép quản lý tự động các mạng ảo và mạng vật lý.

SDN hỗ trợ ảo hóa mạng trong điện toán điện toán đám mây như thế nào?
Bằng cách tách biệt phần kiểm soát mạng (network control) khỏi phần cứng để tạo ra các mạng ảo
Bằng cách cung cấp kết nối vật lý chuyên dụng cho mỗi máy ảo
Bằng cách giảm nhu cầu về máy ảo trên điện toán đám mây
Bằng cách sử dụng phần cứng vật lý để quản lý lưu lượng mạng ảo

SDN cũng giúp tăng tính linh hoạt và khả năng quan sát hành vi mạng. Trong mô hình truyền thống, một bộ định tuyến hay chuyển mạch – dù đặt trên đám mây hay tại trung tâm dữ liệu vật lý – chỉ biết được trạng thái của các thiết bị mạng lân cận. Trong khi đó, SDN tập trung hóa thông tin này, cho phép tổ chức có thể quan sát và kiểm soát toàn bộ mạng lưới và các thiết bị trong đó.

- SDN hỗ trợ ảo hóa mạng trong điện toán điện toán đám mây như thế nào: bằng cách tách biệt phần kiểm soát mạng (network control) khỏi phần cứng để tạo ra các mạng ảo

- Tính năng SDN nào sau đây hữu ích nhất cho việc mở rộng quy mô linh hoạt trong môi trường điện toán đám mây: cấu hình lại nhanh chóng các đường dẫn mạng dựa trên những thay đổi về nhu cầu lưu lượng

CHƯƠNG V – BẢO MẬT

- Thách thức bảo mật đám mây bao gồm: Quản lý môi trường/hệ thống phức tạp; Tuân thủ các quy định và quy tắc; Các vấn đề về bảo mật trung tâm dữ liệu hoặc bảo mật hệ thống vật lý
- Một trong những thách thức trong việc duy trì khả năng giám sát dữ liệu trên đám mây là Dịch vụ điện toán đám mây được truy cập bên ngoài mạng nội bộ doanh nghiệp
- Mã hóa dữ liệu và áp dụng chính sách kiểm soát truy cập giúp đảm bảo dữ liệu đám mây luôn được bảo mật
- Lập kế hoạch an ninh mạng: cho phép khôi phục dữ liệu và duy trì hoạt động liên tục
- Một trong những thách thức về chính sách tuân thủ trong môi trường điện toán đám mây là tuân thủ các yêu cầu quy định như HIPAA và PCI

Tại sao việc duy trì quyền kiểm soát đối với dữ liệu trên nền tảng đám mây lại gặp nhiều thách thức khi sử dụng môi trường của nhà cung cấp dịch vụ đám mây bên thứ ba?
So với việc tự quản lý máy chủ on-premises, đội ngũ quản trị hệ thống CNTT có ít quyền truy cập vào dữ liệu hơn khi sử dụng hạ tầng đám mây
Vi phạm dữ liệu trong môi trường on-premises
Dịch vụ đám mây được truy cập bên ngoài mạng nội bộ doanh nghiệp
Dễ dàng truy cập dữ liệu và ứng dụng đám mây

Tại sao các phương pháp truyền thống dùng để giám sát lưu lượng mạng không hiệu quả trong việc duy trì khả năng giám sát dữ liệu đám mây?
Dịch vụ đám mây được truy cập bên ngoài mạng nội bộ doanh nghiệp
Hạn chế quyền kiểm soát dữ liệu đám mây
Dễ dàng truy cập dữ liệu và ứng dụng đám mây

Vi phạm dữ liệu trong môi trường tại chỗ (on-premises environments)

CÁC TỪ VIẾT TẮT CẦN LƯU Ý

CIA	Confidentiality, Integrity & Availability
IAM	Identity and Access Management
SIEM	Security Information and Event Management
CSA	Cloud Security Alliance: tổ chức chuyên thúc đẩy, phát triển các tiêu chuẩn bảo mật cho điện toán đám mây
AES	Advanced Encryption Standard: Kỹ thuật mã hóa thường được sử dụng để bảo vệ dữ liệu trong quá trình truyền giữa người dùng và dịch vụ đám mây
MitM	Man-in-the-Middle (MitM) attack: Thuật ngữ nào mô tả một cuộc tấn công bảo mật khi kẻ tấn công chặn và thay đổi thông tin liên lạc giữa hai bên trong môi trường đám mây
DDoS	Từ chối Dịch vụ Phân tán (DDoS): Mục tiêu chính của tấn công Từ chối Dịch vụ Phân tán (DDoS) trong bảo mật đám mây là làm gián đoạn tính sẵn sàng của dịch vụ
Penetration test	Loại kiểm thử nào mô phỏng một cuộc tấn công vào hệ thống mạng
Data redundancy	Lưu trữ lặp lại một thông tin hoặc tập dữ liệu được ở nhiều nơi/ nhiều vị trí trong hệ thống để đảm bảo tính sẵn sàng và khả năng phục hồi
Data mirroring	Data mirroring là quá trình sao chép dữ liệu từ một vị trí này sang một vị trí khác theo thời gian thực hoặc gần như thời gian thực, để đảm bảo rằng cả hai nơi luôn có bản sao giống hệt nhau.
Data segregation	Data segregation, hay còn gọi là phân tách dữ liệu, *là quá trình tách biệt dữ liệu thành các phần riêng biệt dựa trên các tiêu chí cụ thể, chẳng hạn như loại dữ liệu, mức độ nhạy cảm, quyền truy cập hoặc mục đích sử dụng*. Mục tiêu chính của data segregation là tăng cường bảo mật, quản lý hiệu quả và tuân thủ các quy định liên quan đến quyền riêng tư và bảo vệ dữ liệu.
Data encryption	Data encryption, hay mã hóa dữ liệu, là quá trình chuyển đổi dữ liệu từ dạng dễ đọc (plaintext) sang dạng mã hóa (ciphertext) bằng cách sử dụng các thuật toán mã hóa và khóa bí mật. Mục đích chính của mã hóa dữ liệu là bảo vệ

	thông tin khỏi bị truy cập hoặc đọc bởi những người không có quyền, đảm bảo tính bảo mật trong quá trình lưu trữ hoặc truyền tải dữ liệu.
Các giao thức bảo mật mạng	Các giao thức bảo mật mạng đảm bảo tính xác thực và bảo mật của dữ liệu khi truyền qua mạng
Access control	Access control (kiểm soát truy cập) là cơ chế dùng để xác định ai được phép truy cập vào tài nguyên nào, trong hoàn cảnh nào, và được phép làm gì với tài nguyên đó.
Backup	Backup (sao lưu dữ liệu) là quá trình tạo bản sao của dữ liệu quan trọng và lưu trữ ở nơi an toàn, để phục hồi lại khi có sự cố xảy ra. Các bản sao lưu (backup) được lưu trữ tại các trung tâm dữ liệu dự phòng được đặt tại các địa điểm khác nhau về mặt địa lý.
Encryption	Encryption (mã hóa) là quá trình chuyển đổi dữ liệu từ dạng dễ đọc (plaintext) sang một dạng khó hiểu hoặc không thể đọc được (ciphertext) – trừ khi có "chìa khóa" giải mã. Mục đích chính là bảo vệ dữ liệu khỏi truy cập trái phép. Ngay cả khi ai đó đánh cắp được dữ liệu đã mã hóa, họ cũng không thể đọc được nội dung nếu không có khóa giải mã.
Data replication	Data replication (sao chép dữ liệu) là quá trình tạo và duy trì các bản sao giống hệt nhau của dữ liệu từ một nguồn chính (master/primary) sang một hoặc nhiều vị trí khác (secondary/replica), để đảm bảo tính sẵn sàng, hiệu suất và an toàn cho dữ liệu.

MỘT SỐ KHÁI NIỆM

SSL/TLS: Giao thức bảo mật được sử dụng trên trình duyệt web để đảm bảo kết nối an toàn với các trang thương mại điện tử.

Network segmentation Biện pháp bảo mật giúp phân tách một mạng thành nhiều vùng để ngăn chặn truy cập trái phép vào dữ liệu nhạy cảm

Availability

Tính khả dụng (tính sẵn sàng) về cơ bản có nghĩa là khi người dùng được ủy quyền (được cấp phép/quyền truy cập) cần truy cập dữ liệu hoặc thông tin, họ có thể. *Tính khả dụng là đảm bảo rằng người có thẩm quyền (được cấp phép/quyền truy cập) cần phải dễ dàng truy cập vào dữ liệu, sản phẩm, dịch vụ (24/07). Availability bao gồm: Đảm bảo hệ thống hoạt động ổn định và dịch vụ không bị từ chối đối với người dùng đã được cấp phép truy cập, Thông tin có thể truy cập và sử dụng được khi người dùng đã được cấp phép truy cập và có nhu cầu hợp lý, Phòng ngừa hành vi che giấu thông tin không được phép*

Integrity

Tính toàn vẹn: là sự đảm bảo *dữ liệu là đáng tin cậy và chính xác. Thông tin chỉ được phép xóa hoặc sửa bởi những người có thẩm quyền (được cấp phép) và phải đảm bảo rằng thông tin vẫn còn chính xác khi được lưu trữ hay truyền đi (in rest & in transit).*

Đảm bảo tính nhất quán và chính xác của dữ liệu theo thời gian là 1 ví dụ về tính toàn vẹn dữ liệu (data integrity) trong lưu trữ điện toán đám mây.

Tham khảo

- *https://vi.wikipedia.org/wiki/H%C3%A0m_b%C4%83m*
- *https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_t%E1%BB%95ng_ki%E1%BB%83m*

Confidentiality

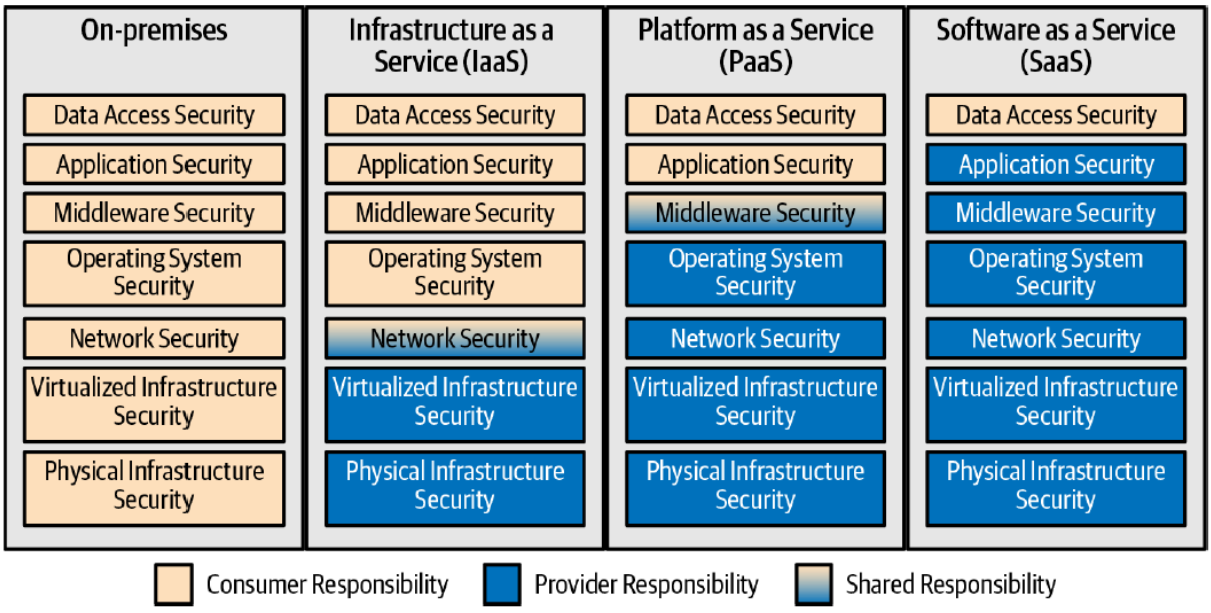
Tính bảo mật: Là nguyên tắc đảm bảo kiểm soát truy cập thông tin. Thông tin *chỉ được phép truy cập bởi những đối tượng (nhân sự, chương trình, ứng dụng máy tính. etc) có thẩm quyền (được cấp phép).*

Disaster recovery

Khôi phục sau thảm họa (Disaster Recovery - DR) bao gồm các công nghệ và các phương pháp (best practices) tốt nhất được thiết kế *nhằm ngăn chặn hoặc giảm thiểu mất mát dữ liệu và gián đoạn hoạt động kinh doanh* do các sự kiện thảm họa gây ra — từ sự cố thiết bị, mất điện cục bộ cho đến các cuộc tấn công mạng, tình trạng khẩn cấp dân sự, tấn công phạm tội hoặc quân sự và thiên tai.

Lập kế hoạch khôi phục sau thảm họa bao gồm việc xây dựng chiến lược, lập kế hoạch, triển khai công nghệ phù hợp và kiểm tra liên tục. Việc duy trì các bản sao lưu dữ liệu (backup) là một yếu tố quan trọng trong kế hoạch khôi phục sau thảm họa, tuy nhiên, *quy trình sao lưu và phục hồi dữ liệu đơn thuần không thể xem là một kế hoạch khôi phục sau thảm họa hoàn chỉnh.*
Khôi phục sau thảm họa cũng bao gồm việc đảm bảo có đủ dung lượng lưu trữ và tài nguyên tính toán để duy trì các quy trình chuyển đổi dự phòng (failover) và chuyển đổi trở lại (failback) một cách hiệu quả. Failover là quá trình chuyển khối lượng công việc sang các hệ thống dự phòng nhằm giảm thiểu tối đa sự gián đoạn đối với quy trình sản xuất và trải nghiệm của người dùng. Failback là quá trình chuyển hoạt động trở lại các hệ thống chính ban đầu.
Các tổ chức nên thiết lập hệ thống khắc phục thảm họa (DR) với mục đích để có những hành động nhanh chóng trong trường hợp dữ liệu bị vi phạm (data breach) và giảm thiểu gián đoạn dịch vụ (downtime)

Shared responsibility model



- Mô hình xác định nhiệm vụ nào do khách hàng chịu trách nhiệm và nhiệm vụ nào do nhà cung cấp đám mây chịu trách nhiệm. Khách hàng sử dụng dịch vụ là đối tượng nào chịu trách nhiệm chính trong việc đảm bảo an toàn dữ liệu trong môi trường đám mây
- Như hình vẽ bên trên, IaaS cung cấp mức bảo mật tích hợp sẵn thấp nhất. Khách hàng sử dụng dịch vụ IaaS phải chịu trách nhiệm bảo mật hệ điều hành, kiểm soát mạng, ứng dụng và bảo vệ dữ liệu
- Các nhà cung cấp dịch vụ SaaS quản lý và bảo vệ ngoại trừ yếu tố kiểm soát truy cập

Quản lý danh tính và truy cập (Identity and Access Management - IAM) là một lĩnh vực của an ninh mạng, liên quan đến cách người dùng truy cập vào các tài nguyên số và những gì họ có thể làm với các tài nguyên đó. Hệ thống IAM giúp ngăn chặn tin tặc xâm nhập, đồng thời đảm bảo rằng mỗi người dùng chỉ có đúng quyền truy cập cần thiết để thực hiện công việc của họ, không hơn. *IAM tập trung vào việc kiểm soát và thực thi chính sách cho người dùng cuối (end-user) khi truy cập các môi trường đám mây khác nhau*

Mạng lưới doanh nghiệp trung bình bao gồm cả người dùng là con người (nhân viên, khách hàng, nhà thầu) và thiết bị (thiết bị IoT và thiết bị đầu cuối, các tác vụ tự động). Với sự gia tăng của làm việc từ xa và điện toán đám mây, cả người dùng và tài nguyên họ cần truy cập ngày càng phân tán. *Quản lý danh tính (Identity management) đảm bảo ngăn chặn sử dụng trái phép, duy trì vai trò người dùng & kiểm soát truy cập dữ liệu trên đám mây*

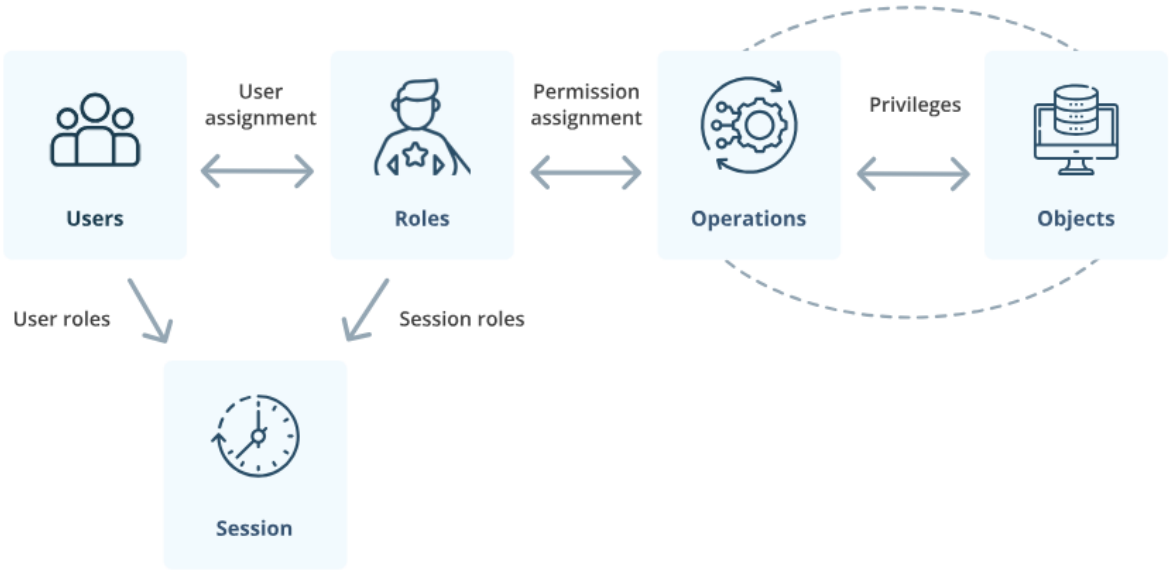
- Xác thực nhằm chứng minh danh tính của người dùng
- Một số phương pháp bảo vệ quyền riêng tư: Xác thực sinh trắc học (Biometric authentication) / Xác thực bằng ID và mật khẩu / Xác thực hai yếu tố (2FA)

- Multi-factor authentication – MFA: Multi-Factor Authentication (MFA) – hay còn gọi là Xác thực đa yếu tố, là một cơ chế bảo mật yêu cầu người dùng phải xác minh danh tính bằng từ 2 yếu tố trở lên, thay vì chỉ dùng mỗi mật khẩu.

  3 yếu tố xác thực chính:

Loại yếu tố	Example
1. Bạn biết	Mật khẩu, mã PIN
2. Bạn có	Điện thoại, mã OTP, USB token, thẻ bảo mật
3. Sinh trắc học	Vân tay, khuôn mặt, giọng nói

- Role-based access control: hay Kiểm soát truy cập dựa trên vai trò — là một mô hình quản lý quyền truy cập trong hệ thống, trong đó người dùng được gán các vai trò (roles), và các vai trò này quyết định người đó được phép làm gì trong hệ thống.



Phòng ngừa mất dữ liệu (DLP) là giải pháp bảo mật giúp xác định và ngăn chặn việc chia sẻ, truyền hoặc sử dụng dữ liệu nhạy cảm không an toàn hoặc không phù hợp. *Giải pháp này có thể giúp tổ chức giám sát và bảo vệ thông tin nhạy cảm trên các hệ thống tại chỗ, vị trí trên nền điện toán đám mây và thiết bị điểm cuối, ngăn chặn việc tiết lộ hoặc rò rỉ dữ liệu trái phép*. Điều này cũng giúp các doanh nghiệp tuân thủ các quy định, chẳng hạn như Đạo luật về Trách nhiệm Giải trình và Cung cấp Thông tin Bảo hiểm Y tế (HIPAA).

Mã hóa dữ liệu

Mã hóa là quá trình chuyển đổi dữ liệu văn bản rõ (plaintext) thành dữ liệu mã hóa không thể đọc được (ciphertext) nhằm che giấu thông tin nhạy cảm khỏi những người dùng không được phép truy cập. Các tổ chức thường xuyên sử dụng mã hóa trong bảo mật dữ liệu để bảo vệ dữ liệu nhạy cảm khỏi việc truy cập trái phép và các sự cố rò rỉ dữ liệu.

Mã hóa hoạt động bằng cách sử dụng các thuật toán mã hóa để làm rối loạn dữ liệu thành một định dạng không thể hiểu được. Chỉ những bên được ủy quyền (được cấp quyền) có khóa bí mật phù hợp, gọi là khóa giải mã, mới có thể khôi phục lại dữ liệu.
Mã hóa có thể bảo vệ dữ liệu khi lưu trữ, khi truyền tải (data at rest & in transit) và trong quá trình xử lý, bất kể dữ liệu đó nằm trong hệ thống máy tính tại chỗ hay trên đám mây. Vì lý do đó, mã hóa đã trở thành một yếu tố then chốt trong các nỗ lực bảo mật đám mây cũng như các chiến lược an ninh mạng nói chung.

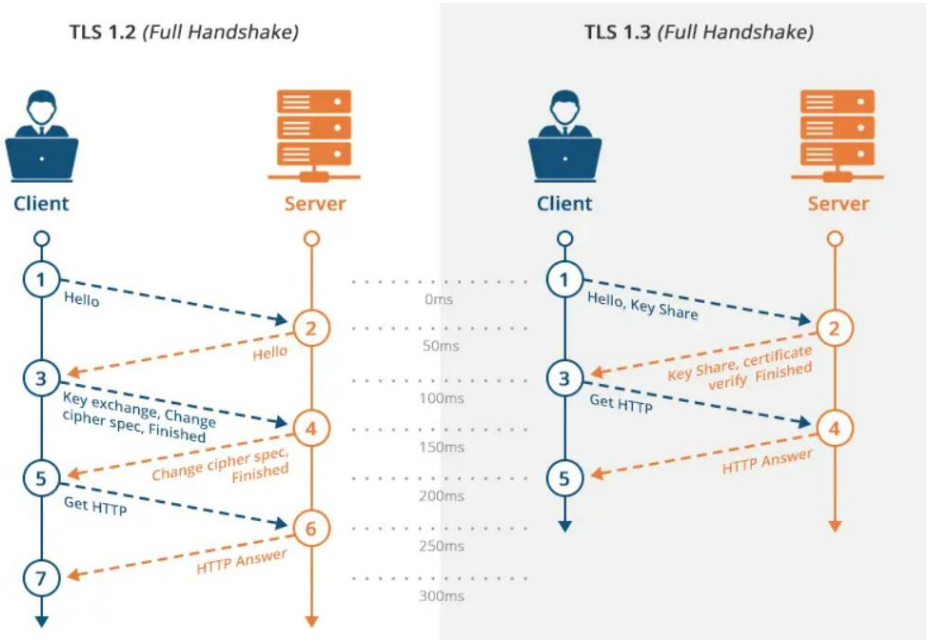
- Mã hóa dữ liệu giúp đảm tính bảo mật của dữ liệu trong lưu trữ đám mây & bảo vệ dữ liệu
- AES (Advanced Encryption Standard): Kỹ thuật mã hóa nào thường được sử dụng để bảo vệ dữ liệu trong quá trình truyền giữa người dùng và dịch vụ đám mây

Tham khảo

- https://www.ibm.com/think/topics/encryption

Thao tác nào cần được thực hiện trước khi chuyển những dữ liệu nhạy cảm lên điện toán đám mây?
Tạo chính sách mã hóa và kiểm soát truy cập
Sao lưu tất cả dữ liệu điện toán đám mây vào hệ thống tại chỗ (on-premise system)
Chỉ lưu trữ dữ liệu không nhạy cảm trên điện toán đám mây
Giảm footprint dữ liệu bằng cách xóa các tệp không cần thiết

TLS



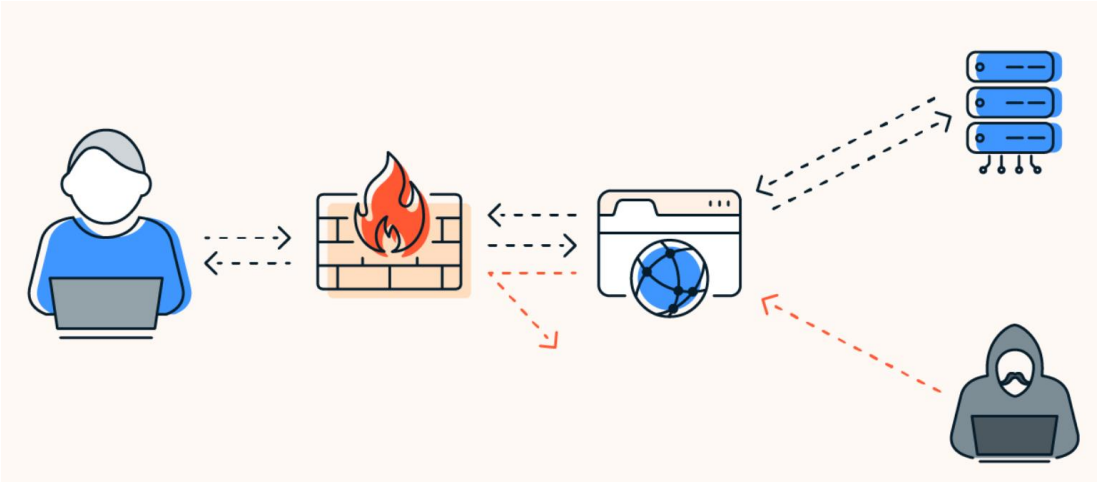
TLS là viết tắt của Transport Layer Security – một giao thức mã hóa được sử dụng để bảo mật dữ liệu truyền qua Internet. TLS chịu trách nhiệm:

- Mã hóa (Encryption): Dữ liệu được mã hóa để người ngoài không đọc được nội dung.
- Xác thực (Authentication): Đảm bảo bạn đang giao tiếp với đúng máy chủ (không bị giả mạo).
- Toàn vẹn dữ liệu (Integrity): Đảm bảo dữ liệu không bị thay đổi trong quá trình truyền tải.

- Tóm lại: TLS đảm bảo Tính toàn vẹn của dữ liệu truyền tải, Tính bảo mật của dữ liệu truyền tải & Xác thực máy chủ

Firewall

- Mục đích của Firewall (tường lửa) là để ngăn chặn các truy cập trái phép vào hệ thống
- Firewall (tường lửa) có thể được triển khai giữa hạ tầng mạng điện toán đám mây và người dùng để bảo vệ và bảo mật mạng



Firewall (hay Tường lửa) là một giải pháp bảo mật được thiết kế để bảo vệ mạng máy tính hoặc các hệ thống điện toán đám mây khỏi những mối đe dọa từ bên ngoài. Giải pháp này hoạt động bằng cách giám sát, kiểm soát lưu lượng mạng, thực hiện lọc dữ liệu và quản lý quyền truy cập chặt chẽ nhằm ngăn chặn các hành vi không hợp lệ đến từ những phần mềm độc hại, đảm bảo dữ liệu và tài nguyên nội bộ luôn được giữ an toàn.

Multi-tenancy

Multi-tenant trong điện toán đám mây là một kiến trúc điện toán đám mây cho phép khách hàng chia sẻ tài nguyên điện toán. Dữ liệu của mỗi người được tách biệt và vẫn ẩn đối với những người khác. Trong hệ thống Multi-tenant Cloud, người dùng có không gian riêng để lưu trữ các dự án và dữ liệu của họ. *Mối quan ngại bảo mật phổ biến liên quan đến multi-tenancy trong điện toán đám mây là Tách biệt dữ liệu*

SLA

- Mục tiêu chính của Thỏa thuận mức dịch vụ đám mây (SLA) là xác định các điều khoản và điều kiện dịch vụ giữa nhà cung cấp đám mây và khách hàng.
- Vai trò chính của Thỏa thuận cấp độ dịch vụ (SLA) trong các dịch vụ đám mây là Thiết lập khuôn khổ về điều khoản cung cấp dịch vụ và các hình phạt khi vi phạm
- Thỏa thuận cấp độ dịch vụ (SLA) thường quy định các thông số: Độ trễ hoặc thời gian phản hồi, Trách nhiệm của từng bên; Mức độ sẵn sàng của dịch vụ

Virtual Private Network

- Tạo ra một đường dẫn cô lập qua mạng công cộng, cho phép các thiết bị điện toán giao tiếp và nhận dữ liệu một cách bảo mật như thể chúng được kết nối trực tiếp qua mạng riêng (private network).
- VPN cung cấp quyền truy cập được mã hóa cho quản trị từ xa
- Mục đích chính của Mạng riêng ảo (VPN) trong bảo mật đám mây là để cô lập mạng
- Đám mây riêng ảo (VPC - Virtual Private Cloud) giúp tăng cường bảo mật trong môi trường đám mây bằng cách cung cấp các kết nối VPN an toàn

Tuân thủ

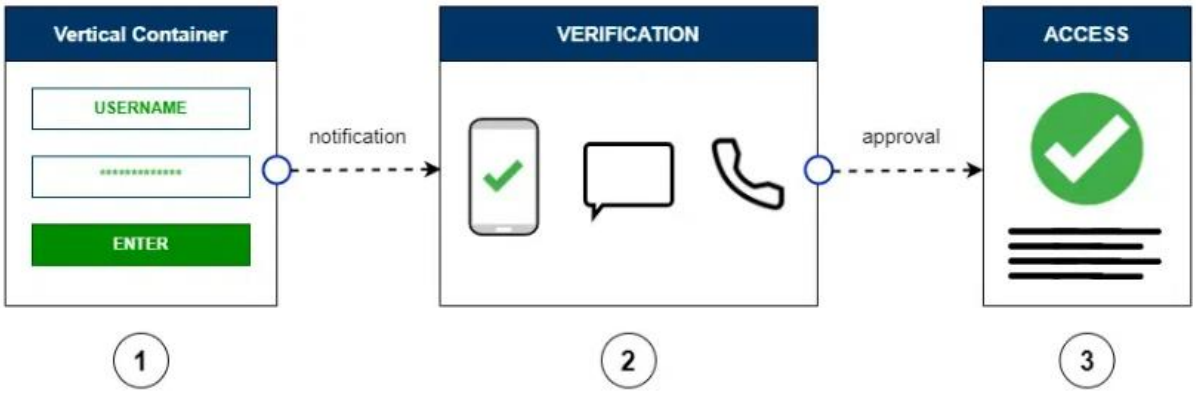
- Tầm quan trọng của việc tuân thủ các quy định và tiêu chuẩn bảo mật là để tránh các hình phạt pháp lý và đảm bảo bảo vệ dữ liệu

Security Information and Event Management (SIEM)

- Mục đích của Security Information and Event Management (SIEM) trong bảo mật điện toán đám mây là để giám sát và phân tích các sự kiện bảo mật

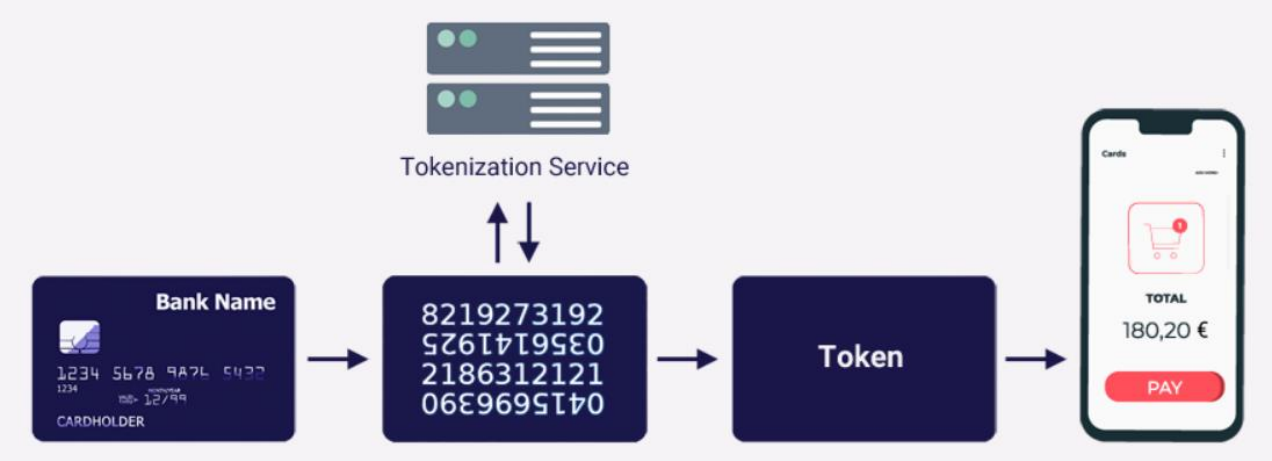
Multi-Factor Authentication (MFA)

- MFA là phương pháp xác thực kết hợp giữa yếu tố mà người dùng biết và yếu tố mà người dùng sở hữu



Tokenization

Mục đích của tokenization trong bảo mật điện toán đám mây là để bảo vệ dữ liệu nhạy cảm



Trong bảo mật điện toán đám mây, tokenization (tạm dịch: mã hóa token) là một kỹ thuật bảo mật dữ liệu nhằm thay thế các thông tin nhạy cảm (như số thẻ tín dụng, thông tin cá nhân, hoặc dữ liệu bí mật) bằng một chuỗi ký tự ngẫu nhiên gọi là token. Token này không có giá trị thực tế hoặc ý nghĩa nếu bị đánh cắp, nhưng nó có thể được ánh xạ ngược lại dữ liệu gốc thông qua một hệ thống an toàn (thường là một kho lưu trữ token riêng biệt).

Tokenization giống như bạn giấu dữ liệu thật đi, và đưa cho hệ thống hoặc người dùng một "thẻ mã hóa" đại diện. Chỉ những ai có quyền truy cập vào "kho giải mã" (token vault) mới lấy được dữ liệu gốc.

Cách hoạt động của tokenization trong điện toán đám mây:

- Thay thế dữ liệu nhạy cảm: Khi dữ liệu được gửi lên cloud, thay vì lưu trữ trực tiếp thông tin gốc (ví dụ: số thẻ tín dụng 1234-5678-9012-3456), hệ thống sẽ tạo ra một token (ví dụ: X7K9-P2M4-Q8R1-T5N3).
- Lưu trữ an toàn: Dữ liệu gốc được lưu trữ trong một kho dữ liệu bảo mật cao (thường tách biệt khỏi môi trường cloud), còn token thì được sử dụng trong các ứng dụng hoặc dịch vụ cloud.
- Ánh xạ ngược: Khi cần truy cập dữ liệu gốc, hệ thống sẽ sử dụng token để tra cứu thông tin thực tế từ kho lưu trữ an toàn, nhưng chỉ với quyền truy cập hợp lệ.

Lợi ích trong bảo mật cloud:

- Giảm rủi ro: Nếu hệ thống cloud bị tấn công, kẻ xâm nhập chỉ lấy được token vô nghĩa thay vì dữ liệu nhạy cảm.
- Tuân thủ quy định: Tokenization giúp các tổ chức tuân thủ các tiêu chuẩn bảo mật như PCI DSS, GDPR, hoặc HIPAA.
- Dễ tích hợp: Không cần mã hóa toàn bộ dữ liệu, giúp tiết kiệm tài nguyên so với mã hóa truyền thống (encryption).

Khác biệt với mã hóa (encryption):

- Mã hóa: Dữ liệu được biến đổi thành dạng mã hóa có thể giải mã bằng khóa. Nếu khóa bị lộ, dữ liệu có thể bị truy cập.
- Tokenization: Dữ liệu gốc không được lưu trong token, nên ngay cả khi token bị lộ, kẻ tấn công không thể truy xuất thông tin mà không vào được kho lưu trữ gốc.

Kiến trúc zero-trust

Kiến trúc Zero Trust (Zero Trust Architecture – ZTA) là một mô hình an ninh mạng với nguyên tắc cốt lõi: "Không tin ai cả, dù họ ở trong hay ngoài hệ thống – luôn xác minh trước khi cấp quyền truy cập." Không mặc định tin tưởng bất kỳ dữ liệu hoặc người dùng nào, và xác minh mọi yêu cầu truy cập

CHƯƠNG VI – CLOUD NATIVE

Cloud native là một cách tiếp cận trong việc phát triển, triển khai và vận hành ứng dụng phần mềm được thiết kế đặc biệt để tận dụng tối đa các lợi ích của môi trường điện toán đám mây (cloud computing). Thay vì chỉ đơn thuần chạy phần mềm truyền thống trên đám mây, các ứng dụng cloud native được xây dựng từ đầu để hoạt động hiệu quả trong các hệ thống phân tán, linh hoạt và có khả năng mở rộng của đám mây.

Cloud-Native stack

Kiến trúc điện toán đám mây nào thường được sử dụng cho **các ứng dụng có tính năng động cao (highly dynamic), cần nhiều tài nguyên** như xử lý video (video processing) hoặc học máy (machine learning)?
Cloud-native architecture với cơ chế auto-scaling
Monolithic architecture
Serverless architecture
Microservices architecture

Git là loại hệ thống quản lý phiên bản nào?
Hệ thống quản lý phiên bản phân tán
Hệ thống quản lý phiên bản tập trung
Hệ thống quản lý phiên bản dùng riêng
Hệ thống quản lý phiên bản cục bộ

Đặc điểm chính của Cloud Native:

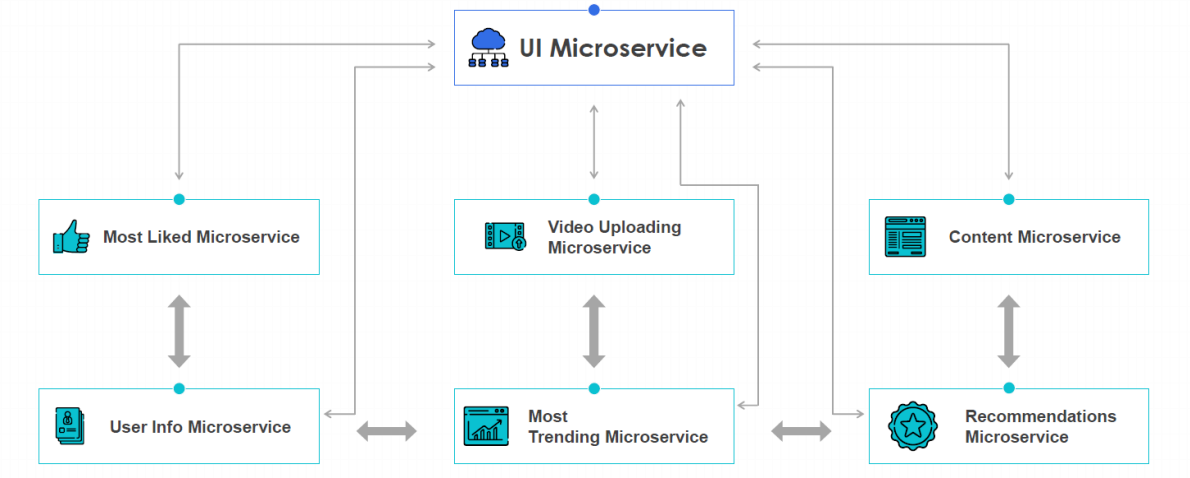
- Kiến trúc microservices: Ứng dụng được chia thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ thực hiện một chức năng cụ thể và giao tiếp qua API Gateway.
- Container hóa: Sử dụng các container để đóng gói ứng dụng và các thành phần phụ thuộc, đảm bảo tính nhất quán giữa các môi trường.
- Tự động hóa và quản lý: Dựa vào các công cụ để tự động triển khai, mở rộng và quản lý ứng dụng.
- Khả năng mở rộng linh hoạt: Có thể dễ dàng mở rộng hoặc thu hẹp tài nguyên theo nhu cầu mà không cần can thiệp thủ công.
- Thiết kế cho khả năng phục hồi: Ứng dụng được xây dựng để tự động khôi phục khi xảy ra lỗi, nhờ vào tính phân tán và dự phòng.

Lợi ích của Cloud-Native Tooling là gì?
Tự động hóa quy trình triển khai

Khả năng mở rộng linh hoạt
Hỗ trợ phát triển liên tục
Hỗ trợ tích hợp liên tục

Cloud native khác với cách tiếp cận truyền thống (monolithic) ở chỗ nó tận dụng sự linh hoạt và phân tán của đám mây, thay vì chỉ "chuyển" một ứng dụng cũ lên cloud.

Example:
Một ứng dụng e-commerce sử dụng kiến trúc microservices để xử lý content, recommendation, most trending, video uploading, user info, etc, chạy trên nền tảng điện toán đám mây với khả năng mở rộng tức thì khi lượng người truy nhập tăng đột biến.



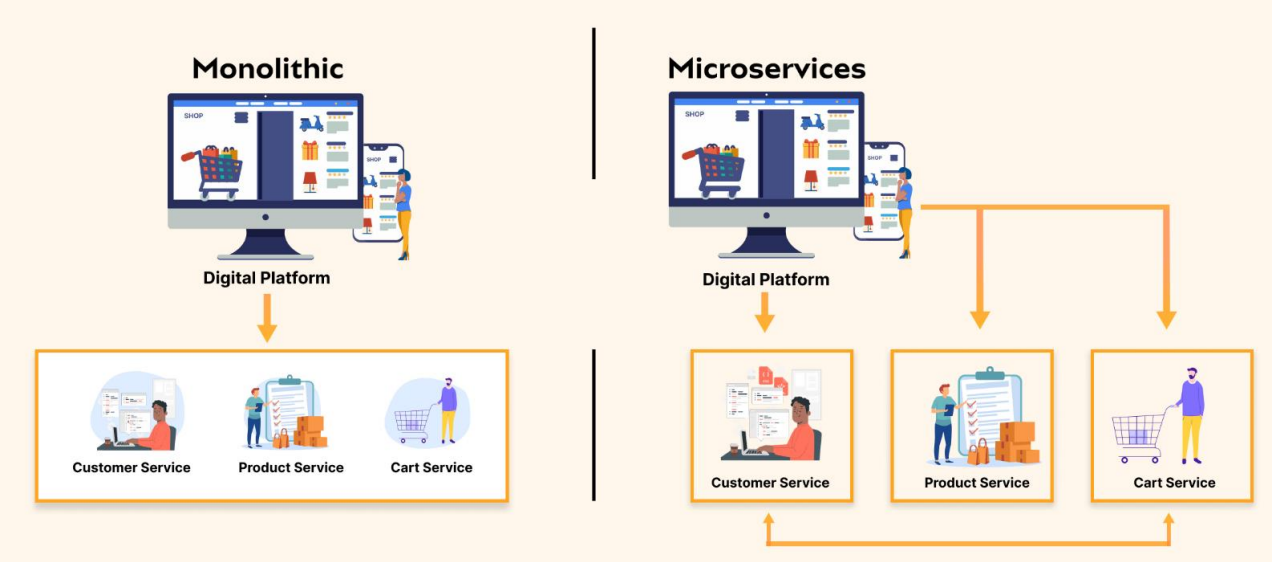
Kiến trúc Monolithic

Kiến trúc nguyên khối (monolithic architecture) là một mô hình phát triển phần mềm truyền thống, trong đó toàn bộ các chức năng nghiệp vụ được thực thi trong một mã nguồn duy nhất. Trong một hệ điều hành nguyên khối, kernel (hạt nhân hệ điều hành) điều phối toàn bộ chức năng.

Kiến trúc nguyên khối (monolithic) là mô hình thiết kế phần mềm trong đó tất cả các thành phần được tích hợp thành một đơn vị duy nhất.

Chiến lược mở rộng quy mô (scale) phù hợp nhất với ứng dụng nguyên khối (monolithic): Mở rộng theo chiều dọc (vertical scaling) bằng cách nâng cấp máy chủ duy nhất

Trong phần mềm nguyên khối (monolithic software), toàn bộ mã nguồn cần thiết cho một ứng dụng được lưu trữ tại một vị trí trung tâm duy nhất. Điều này mang lại lợi ích đơn giản hóa cho các nhà phát triển, vì hệ thống chỉ cần xử lý các giao tiếp theo một định dạng duy nhất. Hệ thống không phải chịu gánh nặng trong việc chuyển đổi giao tiếp giữa các dịch vụ khác nhau.



Hạn chế của kiến trúc nguyên khối

- *Khó tiếp cận công nghệ mới:* Do các ứng dụng nguyên khối thường được liên kết chặt chẽ với nhau, nên rất khó để tích hợp các công nghệ mới. Trên thực tế, thường phải chỉnh sửa lại toàn bộ ứng dụng nguyên khối để có thể chấp nhận sự bổ sung mới.
- *Khả năng mở rộng hạn chế:* Khó khăn trong việc mở rộng độc lập tính năng của ứng dụng. Đây là thách thức lớn nhất mà kiến trúc nguyên khối phải đối mặt. Ngay cả khi nhu cầu mở rộng chỉ ở mức nhỏ (chẳng hạn điều chỉnh một chức năng duy nhất), bạn vẫn có thể phải "tháo rời" gần như toàn bộ hệ thống và xây dựng lại để phản ánh thay đổi đó. Điều này có thể rất tốn thời gian và công sức.

Kiến trúc Microservices

Kiến trúc microservices là một phương thức thiết kế phần mềm trong đó một ứng dụng lớn được chia thành nhiều dịch vụ nhỏ (microservices), mỗi dịch vụ thực hiện một chức năng cụ thể, hoạt động độc lập và giao tiếp với nhau qua API. Đây là một cách tiếp cận đối lập với kiến trúc truyền thống (monolithic), nơi toàn bộ ứng dụng được xây dựng thành một khối duy nhất.

- Một ứng dụng vi dịch vụ (microservice) đáp ứng 1 yêu cầu (request) từ người dùng bằng cách gọi nhiều vi dịch vụ (microservice) nội bộ để tạo ra phản hồi.
- Sự khác biệt chính giữa kiến trúc vi dịch vụ (microservice) và kiến trúc nguyên khối (monolithic) là Vi dịch vụ tách biệt ứng dụng thành các phần nhỏ hơn, trong khi kiến trúc nguyên khối gộp ứng dụng thành một khối duy nhất

Mục tiêu chính của kiến trúc vi dịch vụ (microservice) là Decentralization (Trong bối cảnh hệ thống Công nghệ Thông tin (CNTT), decentralization (phi tập trung hóa) là việc thiết kế và triển khai hệ thống sao cho quyền kiểm soát, xử lý dữ liệu và ra quyết định không nằm ở một điểm trung tâm duy nhất, mà được phân tán ra nhiều nút (nodes), máy chủ hoặc người dùng khác nhau.)

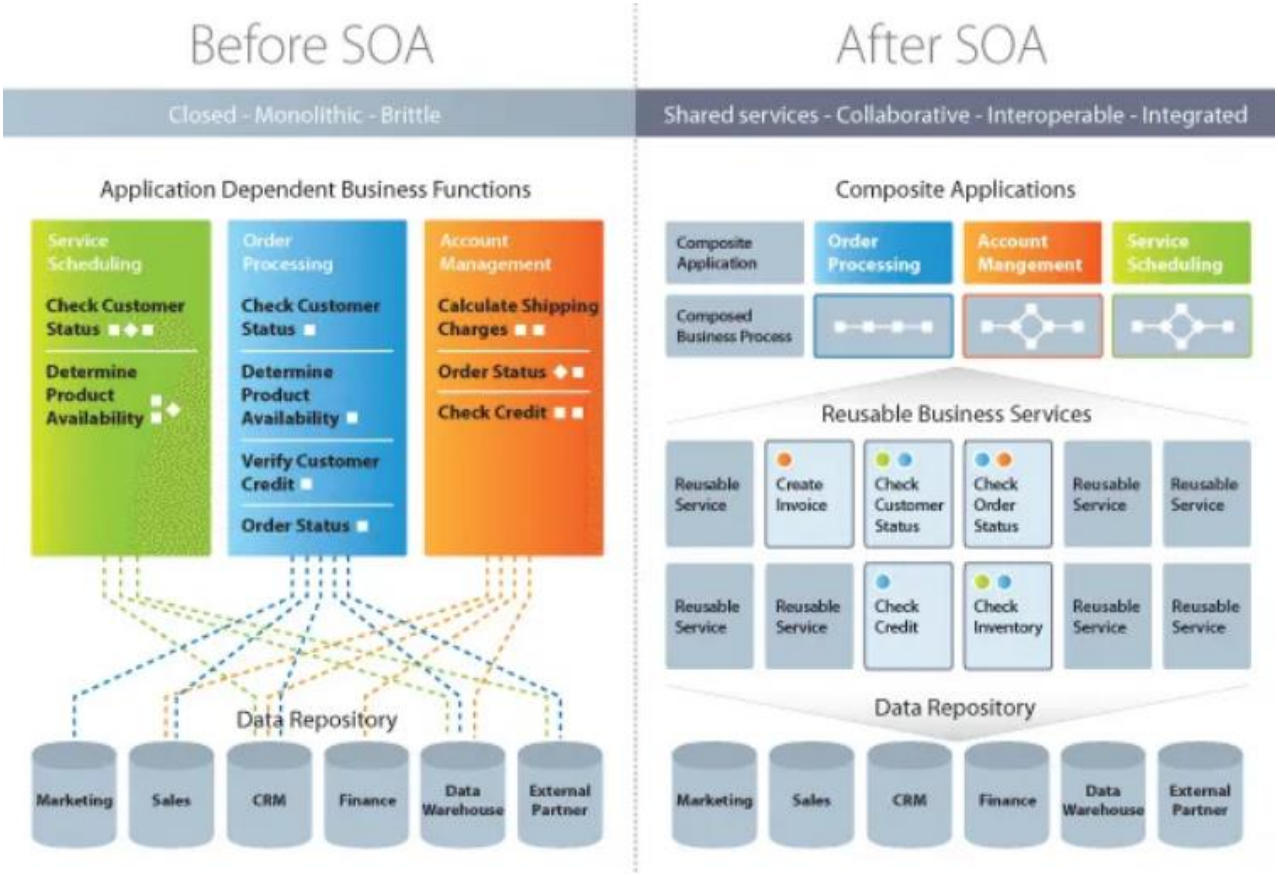
Mô hình kiến trúc nào giúp tạo ra các hệ thống phân tán & có khả năng mở rộng trên nền tảng điện toán đám mây?
Microservices architecture
Monolithic architecture
Layered architecture
Event-driven architecture

Các microservices thường có những đặc điểm & lợi ích sau:

- Tính linh hoạt (Flexibility)
- Khả năng chịu lỗi cao (Resilience)
- Khả năng mở rộng (Scalability)
- Sở hữu bộ công nghệ riêng, bao gồm cả cơ sở dữ liệu và mô hình quản lý dữ liệu.
- Mã nguồn dễ cập nhật hơn — có thể bổ sung tính năng hoặc chức năng mới mà không cần chỉnh sửa toàn bộ ứng dụng.
- Các nhóm phát triển có thể sử dụng các công nghệ hoặc ngôn ngữ lập trình khác nhau cho từng thành phần (dịch vụ) khác nhau.
- Từng thành phần (dịch vụ) có thể được mở rộng độc lập, giúp giảm lãng phí và chi phí so với việc phải mở rộng toàn bộ ứng dụng chỉ vì một tính năng nào đó đang chịu tải cao. Khả năng triển khai độc lập (Independent deployment)/ Một thành phần của vi dịch vụ (microservice) có thể thay đổi mà không ảnh hưởng đến các thành phần khác
- Service registry có nhiệm vụ lưu trữ và quản lý thông tin của tất cả các microservice trong hệ thống, còn service discovery giúp các ứng dụng hoặc dịch vụ khách định vị và kết nối đến đúng microservice phù hợp
- Kiến trúc vi dịch vụ (microservice) trong một trung tâm dữ liệu on-premises không phải là lựa chọn lý tưởng vì doanh nghiệp sẽ không thể tận dụng khả năng mở rộng linh hoạt của đám mây

Service-Oriented Architecture: SOA, viết tắt của Service-Oriented Architecture (Kiến trúc hướng dịch vụ), là một mô hình thiết kế phần mềm trong đó các chức năng của ứng dụng được tổ chức thành các dịch vụ độc lập, có thể tái sử dụng và giao tiếp với nhau qua các giao thức chuẩn (thường là qua mạng). Mục tiêu của SOA là tạo ra một hệ thống linh hoạt, dễ mở rộng và có thể tích hợp với các hệ thống khác.

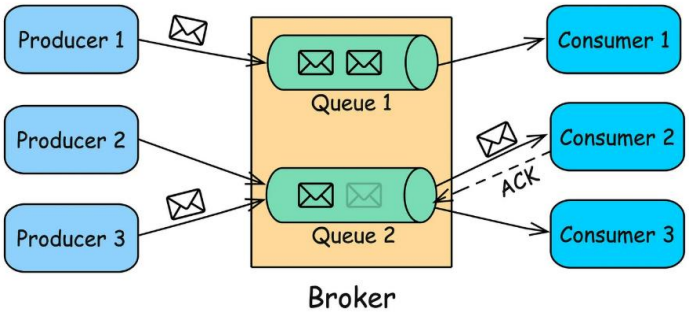
- Trong kiến trúc vi dịch vụ (microservice), kiểu kiến trúc nào phát triển các dịch vụ nhỏ, mỗi dịch vụ chạy trong một quy trình riêng biệt: Kiến trúc hướng dịch vụ (Service-Oriented Architecture)



Bản tin broker

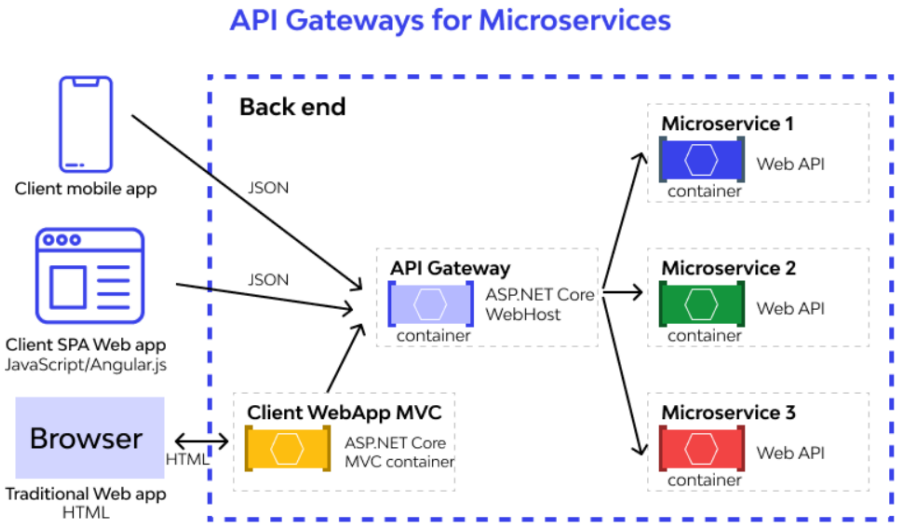
Trong bối cảnh của microservices, "message broker" là một thành phần trung gian dùng để quản lý và điều phối việc giao tiếp giữa các microservice thông qua các tin nhắn (messages). Đây là một phần quan trọng trong kiến trúc microservices, đặc biệt khi các dịch vụ cần giao tiếp bất đồng bộ (asynchronous) thay vì gọi trực tiếp lẫn nhau. Lợi ích của việc sử dụng bản tin broker trong kiến trúc vi dịch vụ (microservice)

- Cung cấp khả năng cân bằng tải và mở rộng linh hoạt
- Cho phép các dịch vụ giao tiếp không đồng bộ
- Tách biệt (decouple) các dịch vụ



Trong Microservices, vai trò của API Gateway: Định tuyến yêu cầu giữa các dịch vụ.

API Gateway đóng vai trò gì trong kiến trúc vi dịch vụ (microservice): Xử lý xác thực và cấp quyền cho các dịch vụ, Cung cấp điểm truy cập duy nhất cho khách hàng khi sử dụng dịch vụ, Định tuyến yêu cầu đến đúng vi dịch vụ (microservice) cần thiết



So sánh giữa bảo mật trong kiến trúc nguyên khối (monolithic) và và bảo mật trong cloud-native, phát biểu nào đúng?
Dễ bảo mật hơn khi ứng dụng nguyên khối (monolithic) chạy trên một máy chủ duy nhất so với một hệ thống phân tán trên cloud-native
Dễ bảo mật hơn khi các vi dịch vụ được triển khai trên nhiều máy chủ ảo trên đám mây
Các ứng dụng nguyên khối không thể chạy trên kết nối bảo mật SSL

CÁC TỪ VIẾT TẮT CẦN LƯU Ý

DevOps	Development and Operations
IAM	Identity and Access Management
SIEM	Security Information and Event Management

DevOps

DevOps là một phương pháp (methodology) phát triển phần mềm nhằm đẩy nhanh quá trình cung cấp các ứng dụng và dịch vụ chất lượng cao hơn thông qua việc *kết hợp và tự động hóa công việc của các nhóm phát triển phần mềm và vận hành CNTT.*

Theo định nghĩa, DevOps vừa là một quy trình phát triển phần mềm, vừa là một *sự thay đổi về văn hóa tổ chức nhằm thúc đẩy sự phối hợp và cộng tác giữa nhóm phát triển và nhóm vận hành – hai nhóm vốn trước đây thường hoạt động tách biệt*

Trên thực tế, các quy trình và văn hóa DevOps hiệu quả nhất không chỉ giới hạn trong phát triển và vận hành, mà còn kết hợp đầu vào từ tất cả các bên liên quan đến ứng dụng trong suốt vòng đời phát triển phần mềm. Điều này bao gồm các kỹ sư hạ tầng và nền tảng, bộ phận an ninh, tuân thủ, quản trị, quản lý rủi ro, các nhóm nghiệp vụ, người dùng và khách hàng.

Tại sao tự động hóa lại quan trọng trong DevOps: Cải thiện hiệu suất và tính nhất quán trong các quy trình

Sự thay đổi văn hóa trong DevOps:

- Xóa bỏ tình trạng các phòng ban làm việc một cách biệt lập, thiếu phối hợp với nhau trong doanh nghiệp
- Chia sẻ trách nhiệm giữa các các phòng ban khác nhau trong doanh nghiệp
- Hợp tác chặt chẽ giữa nhóm phát triển ứng dụng và nhóm vận hành CNTT

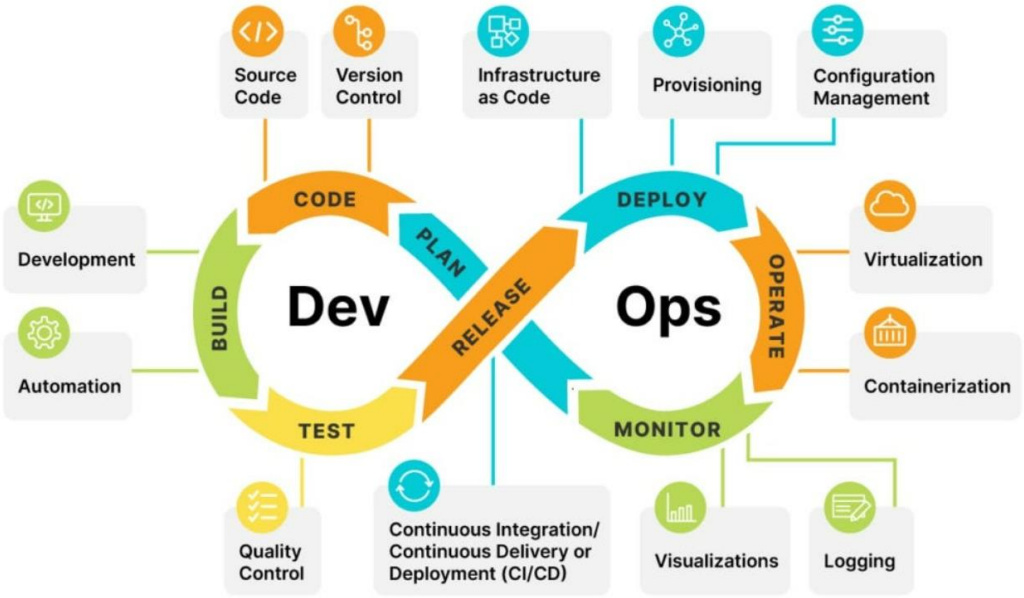
DevOps giải quyết những vấn đề

- Chu kỳ triển khai phần mềm kéo dài
- Kết nối (bridge) hạ tầng và ứng dụng
- Ứng dụng kém hiệu quả hoặc lỗi thời

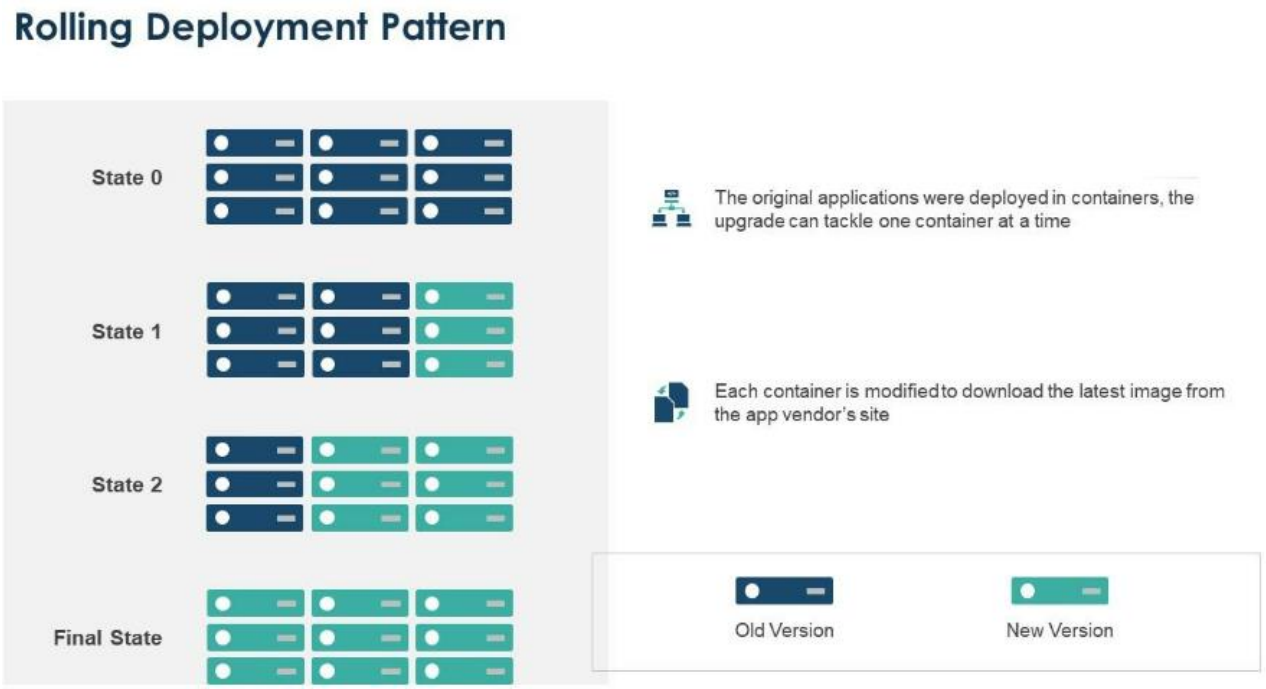
Môi trường kiểm thử (Test Environment) trong DevOps: Hệ thống nơi mã nguồn mới được kiểm thử trước khi triển khai

Chu trình DevOps bao gồm những bước: Plan, code, build, test, deploy (release – deploy), operate, monitor, plan. Ngắn gọn có thể coi các thành phần chính của DevOps bao gồm Development, Testing, Deployment, Operations

Quy trình nào trong DevOps tập trung vào tự động hóa quy trình xây dựng, kiểm thử và triển khai mã nguồn?
Tích hợp liên tục (CI) và Triển khai liên tục (CD)
Kiểm thử thủ công
Hạ tầng dưới dạng mã (IaC)
Công nghệ container



Rolling Deployment là một chiến lược triển khai phần mềm trong đó phiên bản mới của ứng dụng được *cập nhật dần dần lên các máy chủ hoặc môi trường sản xuất mà không gây gián đoạn toàn bộ dịch vụ, sau đó triển khai dần dần cho toàn hệ thống.* Thay vì thay thế toàn bộ hệ thống cùng một lúc, rolling deployment thực hiện cập nhật từng phần, đảm bảo rằng hệ thống vẫn hoạt động liên tục trong suốt quá trình.



CI/CD

Phân phối liên tục (CD): quy trình tự động triển khai mã nguồn sau khi kiểm thử thành công

Tích hợp liên tục (CI): liên tục hợp nhất những thay đổi trong mã nguồn vào một kho lưu trữ chung

Mục đích chính của CI/CD Pipelines là để tự động hóa quy trình phát triển, kiểm thử và triển khai phần mềm. Build Pipeline trong CI/CD là quy trình trong đó mã nguồn được xây dựng, kiểm thử và triển khai tự động.

Sự khác biệt chính giữa Phân phối liên tục và Triển khai liên tục (Continuous Delivery & Continuous Deployment) là Phân phối liên tục triển khai phần mềm thủ công, trong khi Triển khai liên tục tự động hóa toàn bộ quy trình triển khai

Trong CI/CD, tích hợp liên tục (Continuous Integration - CI) có nghĩa là gì?
Liên tục tích hợp các thay đổi mã từ nhiều lập trình viên một cách tự động và thường xuyên
Tự động triển khai phần mềm vào môi trường sản xuất
Tự động hóa quy trình kiểm thử hiệu suất ứng dụng
Tối ưu hóa quản lý kho lưu trữ mã nguồn

Mục tiêu chính của tích hợp liên tục (CI) trong quy trình DevOps là gì?
Kiểm thử ứng dụng mỗi khi có thay đổi trong mã nguồn
Lưu trữ mã nguồn trong kho tập trung
Cải thiện mức độ tương tác của người dùng
Đảm bảo mã nguồn được triển khai tự động lên môi trường sản xuất

Infrastructure as Code (IaC)

Mã nguồn để tự động hóa việc cung cấp và quản lý cơ sở hạ tầng CNTT

CONTAINER

Event-based architecture

Event-based architecture (kiến trúc dựa trên sự kiện) là một mô hình thiết kế phần mềm trong đó các thành phần của hệ thống giao tiếp với nhau chủ yếu thông qua việc phát ra và xử lý các sự kiện. Thay vì các thành phần gọi trực tiếp lẫn nhau (như trong kiến trúc truyền thống), chúng tạo ra các sự kiện khi có điều gì đó quan trọng xảy ra, và các thành phần khác sẽ lắng nghe, phản ứng với những sự kiện đó.

Loại ứng dụng nào phù hợp nhất để triển khai bằng container: Ứng dụng theo sự kiện (event-based architecture/event-based application)

Mô hình thiết kế nào thường được sử dụng trong các ứng dụng cloud-native để xử lý giao tiếp giữa các vi dịch vụ (microservice): Kiến trúc hướng sự kiện

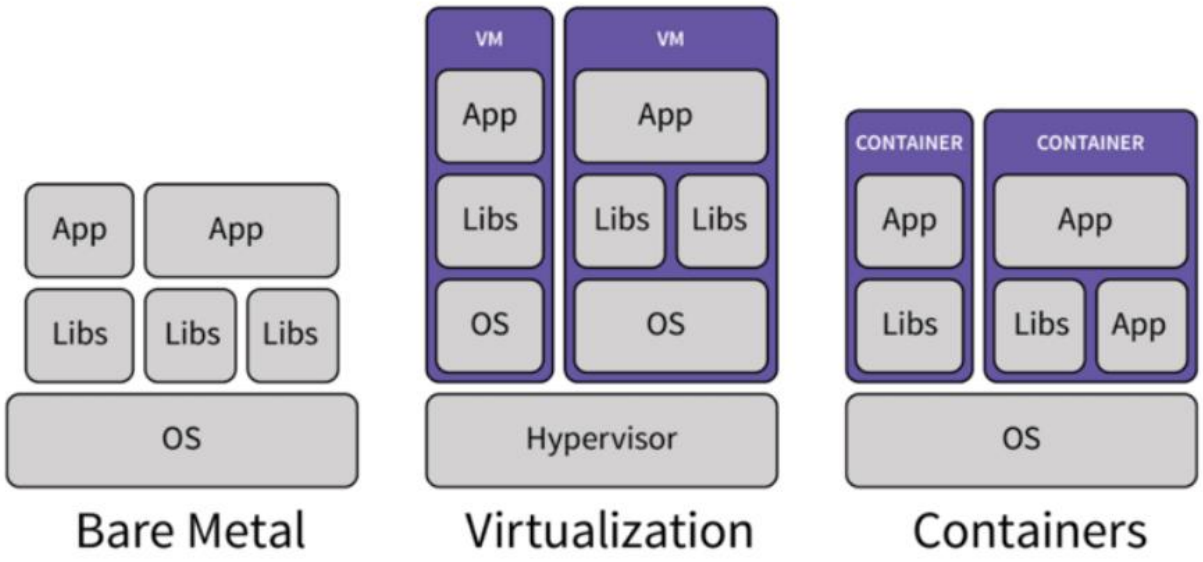
Container

Container là các đơn vị phần mềm có thể thực thi, đóng gói mã ứng dụng cùng với các thư viện và các phụ thuộc cần thiết. Chúng cho phép mã có thể chạy trong bất kỳ môi trường tính toán nào, dù là máy tính cá nhân, hạ tầng CNTT truyền thống hay môi trường điện toán đám mây.

Mục đích chính của container trong kiến trúc điện toán đám mây hiện đại là gì?
Để chạy các ứng dụng trong môi trường biệt lập có thể dễ dàng triển khai và mở rộng quy mô
Để quản lý vòng đời của máy ảo
Để lưu trữ dữ liệu ứng dụng
Để sao lưu các ứng dụng trên điện toán đám mây

Linh hoạt hơn và sử dụng tài nguyên hiệu quả hơn so với máy ảo (VM), container đã trở thành đơn vị tính toán mặc định của các ứng dụng hiện đại theo kiến trúc cloud-native. Ngoài ra, container còn đóng vai trò then chốt trong hạ tầng CNTT nền tảng cho các môi trường đám mây lai (hybrid multicloud)—kết hợp giữa hệ thống tại chỗ (on-premises), đám mây riêng, đám mây công cộng và nhiều dịch vụ đám mây từ nhiều nhà cung cấp khác nhau.

Container trong bối cảnh vi dịch vụ (microservice) là một gói phần mềm chứa tất cả các phụ thuộc cần thiết để một dịch vụ có thể chạy độc lập



Sự khác biệt giữa container và máy ảo (VM) về tài nguyên hệ thống là gì?
Container có thể chạy trên cùng một nhân hệ điều hành (OS kernel)

Đặc điểm quan trọng của ứng dụng được triển khai bằng container là gì?

Có thể được quản lý & triển khai độc lập đối với các ứng dụng khác

Khi chuyển đổi từ hệ thống từ kiến trúc nguyên khối (monolithic) sang kiến trúc vi dịch vụ (microservice), thách thức nào dưới đây không phải là vấn đề chính?
Điều phối container trên nhiều máy chủ

Docker là gì?
Nền tảng đóng gói, triển khai và quản lý các container

Mục đích chính của Docker Engine là gì?
Dùng để tạo và chạy container

Trong Docker, bridge network là gì?
Một mạng kết nối các container trên cùng một máy chủ

Mục đích chính của Docker Swarm là gì?
Điều phối container trên nhiều máy chủ (multi-host)

Trong Serverless Computing, thành phần nào được loại bỏ hoàn toàn?
Quản lý máy chủ
Máy chủ vật lý
Điều phối container
Tự động mở rộng hệ thống

Đặc điểm quan trọng của ứng dụng được triển khai bằng container là gì?
Có thể được quản lý & triển khai độc lập đối với các ứng dụng khác

Nhược điểm của việc sử dụng container là gì?
Khả năng lưu trữ hạn chế

Những thách thức mà tổ chức có thể gặp phải khi sử dụng VM thay vì container hoặc serverless là gì?
Sử dụng tài nguyên không hiệu quả

Loại kiến trúc nào phù hợp nhất cho các ứng dụng nguyên khối lâu đời và hệ thống cũ/lỗi thời (monolithic applications & legacy systems)?

Máy ảo (VM)

Trong Kubernetes, đơn vị triển khai nhỏ nhất là gì?
Pod
Container
Kubelet
Node