TCVN **TIÊU CHUẨN QUỐC GIA**

TCVN 13811:2023
ISO/IEC TS 23167:2020

Xuất bản lần 1

CÔNG NGHỆ THÔNG TIN – TÍNH TOÁN MÂY – CÁC CÔNG NGHỆ VÀ KỸ THUẬT PHỔ BIẾN

Information technology — Cloud computing — Common technologies and techniques

HÀ NỘI - 2023

Mục lục

Lời nói đầu

TCVN 13811:2023 hoàn toàn tương đương với ISO/IEC TS 23167:2020.

TCVN 13811:2023 do Ban kỹ thuật tiêu chuẩn quốc gia TCVN/JTC 1 "*Công nghệ thông tin*" biên soạn, Viện Tiêu chuẩn Chất lượng Việt Nam đề nghị, Tổng cục Tiêu chuẩn Đo lường Chất lượng thẩm định, Bộ Khoa học và Công nghệ công bố.

Công nghệ thông tin – Tính toán máy – Các công nghệ và kỹ thuật phổ biến

Information technology — Cloud computing — Common technologies and techniques

1. Phạm vi áp dụng

Tiêu chuẩn này cung cấp mô tả về tập hợp các công nghệ và kỹ thuật phổ biến được sử dụng cùng với tính toán mây. Bao gồm các:

- máy ảo (VM) và trình giám sát máy ảo;
- vùng chứa và hệ thống quản lý vùng chứa (CMS);
- tính toán không server;
- kiến trúc vi dịch vụ;
- tự động hóa;
- nền tảng như một hệ thống dịch vụ và kiến trúc;
- dịch vụ lưu trữ;
- bảo mật, khả năng mở rộng và kết nối mạng như được áp dụng cho các công nghệ tính toán mây ở trên.

2. Tài liệu viện dẫn

Các tài liệu viện dẫn dưới đây là cần thiết cho việc áp dụng tiêu chuẩn này. Đối với các tài liệu ghi năm công bố thì áp dụng phiên bản được nêu. Đối với các tài liệu không ghi năm công bố thì áp dụng phiên bản mới nhất, bao gồm cả các sửa đổi, bổ sung (nếu có).

TCVN 13809-1:2023 (ISO/IEC 22123-1:2021) Công nghệ thông tin – Tính toán mây – Phần 1: Từ vựng.

3. Thuật ngữ và định nghĩa

Tiêu chuẩn này áp dụng thuật ngữ và định nghĩa trong TCVN 13809-1:2023 (ISO/IEC 22123-1:2021) và các thuật ngữ và định nghĩa sau đây.

3.1

Hệ điều hành khách (guest operating system)

OS khách (guest OS)

Hệ điều hành chạy trong một máy ảo.

[NGUỒN: 3.2, ISO/IEC 21878:2018]

3.2

Hệ điều hành máy chủ (host operating system)

OS máy chủ (host OS)

Hệ điều hành được cài đặt phần mềm ảo hóa.

CHÚ THÍCH 1: "phần mềm ảo hóa" có thể bao gồm cả trình giám sát máy ảo và máy ảo cũng như Daemon vùng chứa (3.4) và vùng chứa.

3.3

Tính toán không server (serverless computing)

Thể loại dịch vụ mây trong đó khách hàng dịch vụ mây có thể sử dụng các loại chức năng mây khác nhau mà không cần khách hàng dịch vụ mây phải cung cấp, triển khai và quản lý tài nguyên phần cứng hoặc phần mềm, ngoài việc cung cấp mã ứng dụng của khách hàng dịch vụ mây hoặc cung cấp dữ liệu khách hàng dịch vụ mây.

CHÚ THÍCH 1: Tính toán không server cung cấp khả năng tự động mở rộng quy mô với khả năng phân bổ tài nguyên linh hoạt của nhà cung cấp dịch vụ mây, phân phối tự động trên nhiều địa điểm cũng như bảo trì và sao lưu tự động.

CHÚ THÍCH 2: Chức năng tính toán không server được kích hoạt bởi một hoặc nhiều sự kiện do khách hàng dịch vụ mây xác định và có thể thực thi trong một khoảng thời gian giới hạn theo yêu cầu để xử lý từng sự kiện.

CHÚ THÍCH 3: Chức năng tính toán không server có thể được gọi bằng cách gọi trực tiếp từ các ứng dụng web và di động.

3.4

Daemon[1] vùng chứa (container daemon)

Dịch vụ phần mềm thực thi trên hệ điều hành máy chủ (3.2) và chịu trách nhiệm tạo, bắt đầu và dừng các vùng chứa trên hệ thống đó.

3.5

Hệ thống quản lý vùng chứa (container management system)

CMS

Phần mềm cung cấp sự quản lý và điều phối các phiên bản vùng chứa.

CHÚ THÍCH 1: Các khả năng bao gồm tạo và sắp xếp ban đầu, lập lịch biểu, giám sát, mở rộng quy mô, cập nhật và triển khai song song các khả năng như bộ cân bằng tải, tường lửa, mạng ảo và ghi nhật ký.

3.6

Ứng dụng mây bẩm sinh (cloud native application)

Ứng dụng được thiết kế rõ ràng để chạy bên trong và tận dụng các khả năng cũng như môi trường của dịch vụ mây.

[1] Trong một số tài liệu 'Deamon' được gọi là "trình nền"

3.7

Phân rã chức năng (functional decomposition)

Kiểu phân rã mô-đun trong đó một hệ thống được chia nhỏ thành các thành phần tương ứng với các chức năng và chức năng con của hệ thống.

VÍ DỤ: Phân tách theo thứ bậc, sàng lọc từng bước.

[NGUỒN: ISO/IEC/IEEE 24765:2017, 3.1695]

3.8

Triển khai liên tục (continuous deployment)

Cách tiếp cận công nghệ phần mềm trong đó các nhóm sản xuất phần mềm theo các chu kỳ ngắn sao cho phần mềm có thể được phát hành vào sản xuất bất kỳ lúc nào và việc triển khai vào sản xuất được tự động hóa.

3.9

Bàn giao liên tục (continuous delivery)

Triển khai liên tục (3.8) trong đó giai đoạn triển khai được bắt đầu theo cách thủ công.

3.10

DevOps

Phương pháp kết hợp phát triển phần mềm và vận hành CNTT cùng nhau để rút ngắn vòng đời phát triển và vận hành.

3.11

DevSecOps

DevOps (3.10) được mở rộng để bao gồm các khả năng bảo mật như một phần thiết yếu và không thể thiếu trong quá trình phát triển và vận hành.

3.12

Điều phối (orchestration)

Kiểu kết hợp trong đó một phần tử cụ thể được kết hợp sử dụng để giám sát và chỉ đạo các phần tử khác.

CHÚ THÍCH 1: Phần tử chỉ đạo một điều phối không phải là một phần của chính điều phối (ví dụ kết hợp cụ thể).

CHÚ THÍCH 2: Xem ISO/IEC 18384-3:2016, 8.3.

[NGUỒN: ISO/IEC 18384-1:2016, 2.16]

3.13

Ảnh tượng máy ảo (virtual machine image)

Ảnh tượng VM (VM image)

Thông tin và mã thực thi cần thiết để chạy một máy ảo.

3.14

Siêu dữ liệu máy ảo (virtual machine metadata)

Siêu dữ liệu VM (VM metadata)

Thông tin về cấu hình và khởi động máy ảo.

3.15

Vi dịch vụ (microservice)

Tạo tác có thể độc lập triển khai để cung cấp dịch vụ thực hiện một phần chức năng cụ thể của ứng dụng.

3.16

Kiến trúc vi dịch vụ (microservice architecture)

Cách tiếp cận thiết kế chia ứng dụng thành một tập hợp các vi dịch vụ (3.15).

3.17

Các chức năng như một dịch vụ (functions as a service)

Chức năng như một dịch vụ (function as a service)

FaaS

Thể loại dịch vụ mây trong đó khả năng được cung cấp cho khách hàng dịch vụ mây là thực thi mã ứng dụng của khách hàng dịch vụ mây, dưới dạng một hoặc nhiều chức năng, mỗi chức năng được kích hoạt bởi một sự kiện do khách hàng dịch vụ mây chỉ định.

3.18

Cơ sở dữ liệu không server (serverless database)

Thể loại dịch vụ mây trong đó khả năng cung cấp cho khách hàng dịch vụ mây là cơ sở dữ liệu được nhà cung cấp dịch vụ mây quản lý đầy đủ được cung cấp thông qua giao diện lập trình ứng dụng.

3.19

Tường lửa (firewall)

Kiểu hàng rào bảo mật được đặt giữa các môi trường mạng - bao gồm một thiết bị chuyên dụng hoặc tổ hợp của một số thành phần và kỹ thuật - qua đó tất cả lưu lượng truy cập từ môi trường mạng này đến môi trường mạng khác và ngược lại, và chỉ lưu lượng được ủy quyền, như được xác định bởi an ninh cục bộ chính sách, được phép qua.

[NGUỒN: ISO/IEC 27033-1:2015, 3.12]

3.20

Sổ lưu ký[2] vùng chứa (container registry)

Thành phần cung cấp khả năng lưu trữ và truy cập ảnh tượng vùng chứa.

3.21

Tương đồng tài nguyên (resource affinity)

Sự bố trí hai hoặc nhiều tài nguyên tương đồng nhau.

CHÚ THÍCH 1: Tính tương đồng liên quan đến các yếu tố như tốc độ truy cập hoặc băng thông truy cập cao giữa các tài nguyên.

4. Ký hiệu và thuật ngữ viết tắt

API	Application programming interface	Giao diện lập trình ứng dụng
CMS	Container management system	Hệ thống quản lý vùng chứa
CSC	Cloud service customer	Khách hàng dịch vụ mây
CSP	Cloud service provider	Nhà cung cấp dịch vụ mây
DNS	Domain name service	Dịch vụ tên miền
GUI	Graphical user interface	Giao diện người dùng đồ họa
HTTP	Hypertext transfer protocol	Giao thức truyền siêu văn bản
IaaS	Infrastructure as a service	Hạ tầng như một dịch vụ
IP	Internet protocol	Giao thức Internet
MAC	Media access control	Kiểm soát truy cập phương tiện
OCI	Open containers initiative	Sáng kiến các vùng chứa mở
OS	Operating system	Hệ điều hành
OVF	Open virtualization format	Định dạng ảo hóa mở
PaaS	Platform as a service	Nền tảng như một dịch vụ
SaaS	Software as a service	Phần mềm như là một dịch vụ
VPN	Virtual private network	Mạng riêng ảo

5. Tổng quan về các công nghệ và kỹ thuật phổ biến sử dụng trong tính toán mây

5.1 Quy định chung

Tiêu chuẩn này cung cấp mô tả về tập hợp các công nghệ và kỹ thuật phổ biến được sử dụng cùng với tính toán mây.

Công nghệ phổ biến là công nghệ được sử dụng để triển khai một hoặc nhiều thành phần chức năng

[2] Trong một số TCVN là Sổ đăng ký

của tính toán mây được mô tả trong kiến trúc tham chiếu tính toán mây 9.2, TCVN 12481:2019 (ISO/IEC 17789:2014) [2]. Các công nghệ phổ biến thường tạo thành một phần của dịch vụ mây hoặc được khách hàng dịch vụ mây (CSC) sử dụng khi sử dụng dịch vụ mây.

Một kỹ thuật phổ biến là một phương pháp hoặc cách tiếp cận để thực hiện một số hoạt động liên quan đến tính toán mây, như được mô tả trong 10.2.2, TCVN 12481:2019 (ISO/IEC 17789:2014), [2]. Điển hình của các kỹ thuật phổ biến là giảm nỗ lực cần thiết để sử dụng dịch vụ mây hoặc cho phép sử dụng đầy đủ các khả năng do dịch vụ mây cung cấp.

Nhiều công nghệ và kỹ thuật phổ biến được sử dụng kết hợp khi phát triển và vận hành các ứng dụng mây bẩm sinh.

Các công nghệ và kỹ thuật phổ biến khác nhau được mô tả chi tiết trong các Điều sau.

Trong phần tiếp theo, văn bản được trích xuất từ các tiêu chuẩn khác được biểu thị bằng cách đặt văn bản được trích xuất trong dấu ngoặc kép, sử dụng văn bản in nghiêng và cung cấp tham chiếu chính xác ở cuối văn bản được trích xuất.

5.2 Công nghệ

5.2.1 Quy định chung

Các công nghệ phổ biến chủ yếu liên quan đến ảo hóa, kiểm soát và quản lý tài nguyên ảo hóa trong quá trình phát triển và vận hành các ứng dụng mây bẩm sinh. Một ứng dụng mây bẩm sinh là một ứng dụng được thiết kế rõ ràng để chạy bên trong và tận dụng các khả năng cũng như môi trường của các dịch vụ mây. Các công nghệ này giải quyết ba tài nguyên phần cứng chính được xác định trong 9.2.4.2, TCVN 12481:2019 (ISO/IEC 17789:2014) [2] về xử lý, lưu trữ và kết nối mạng nhưng cũng giải quyết kiểu khả năng nền tảng của dịch vụ mây. Những công nghệ này bao gồm:

- Quá trình xử lý ảo hóa được giải quyết bằng máy ảo (xem Điều 6), bằng vùng chứa (xem Điều 7), bằng tính toán không server (xem Điều 8).
- Lưu trữ ảo hóa được giải quyết bằng nhiều loại Lưu trữ dữ liệu như một dịch vụ (xem Điều 12).
- Mạng ảo hóa là một trong những nhóm công nghệ chính để cung cấp và sử dụng các khả năng kết nối mạng liên quan đến các dịch vụ mây (xem Điều 13).
- Thể loại Nền tảng dưới dạng Dịch vụ của các dịch vụ mây được thiết kế để cho phép phát triển, thử nghiệm và sản xuất các ứng dụng gốc trên mây nhanh hơn (xem Điều 11).

Các công nghệ bảo mật và khả năng mở rộng áp dụng chung cho tất cả các loại dịch vụ mây, mặc dù việc CSC sử dụng rõ ràng các công nghệ này phổ biến hơn đối với một số loại dịch vụ mây (xem Điều 14 và 15).

5.2.2 Kiểu khả năng hạ tầng của dịch vụ mây

Các công nghệ thường được sử dụng với kiểu khả năng hạ tầng của dịch vụ mây bao gồm:

- máy ảo;
- vùng chứa;

- lưu trữ ảo hóa;
- mạng ảo hóa;
- an ninh.

5.2.3 Các khả năng nền tảng của dịch vụ mây

Các công nghệ thường được sử dụng với khả năng nền tảng của loại dịch vụ mây bao gồm:

- vùng chứa;
- tính toán không server;
- dịch vụ mây PaaS;
- lưu trữ ảo hóa;
- mạng ảo hóa;
- an ninh.

5.2.4 Kiểu khả năng ứng dụng của dịch vụ mây

Các công nghệ thường được sử dụng với kiểu khả năng ứng dụng của dịch vụ mây bao gồm:

- lưu trữ ảo hóa;
- mạng ảo hóa;
- an ninh.

5.3 Kỹ thuật

Các kỹ thuật phổ biến thường áp dụng cho tất cả các thể loại dịch vụ mây, mặc dù một số kỹ thuật hữu ích hơn với một số thể loại dịch vụ mây so với các thể loại khác.

Điều phối và quản lý tài nguyên ảo hóa đạt được bằng công cụ, bao gồm CMS (xem Điều 10 và 7.4).

Các kỹ thuật thường được sử dụng với tính toán mây bao gồm:

- Các loại tự động hóa khác nhau, được áp dụng xuyên suốt các quy trình DevOps (xem Điều 10).
- Các cách tiếp cận khả năng mở rộng như các phiên bản song song (xem Điều 14).
- Cách tiếp cận thiết kế vi mô dịch vụ cho các ứng dụng và hệ thống (xem Điều 9).
- Các kỹ thuật tường lửa, mã hóa và Quản lý truy cập và nhận dạng (IAM) để bảo mật và bảo vệ quyền riêng tư (xem Điều 15).

6 Máy ảo và trình giám sát máy ảo

6.1 Quy định chung

Máy ảo và trình giám sát máy ảo là những công nghệ cung cấp khả năng xử lý ảo hóa (còn được gọi là tính toán ảo hóa) cho các dịch vụ mây. Những công nghệ này chủ yếu liên quan đến các dịch vụ mây thuộc kiểu khả năng hạ tầng và IaaS như được mô tả trong TCVN 12480 (ISO/IEC 17788) và TCVN

12481 (ISO/IEC 17789).

Một trong những đặc trưng chính của tính toán mây là khả năng chia sẻ tài nguyên. Đây là nền tảng về tính kinh tế, nhưng nó cũng quan trọng đối với các đặc trưng như khả năng mở rộng và khả năng phục hồi. Chia sẻ tài nguyên xử lý yêu cầu một số mức độ ảo hóa. Ảo hóa nói chung có nghĩa là một số tài nguyên được cung cấp để sử dụng ở dạng không tồn tại về mặt vật lý như vậy nhưng được phần mềm tạo ra để làm như vậy. Nói cách khác, ảo hóa cung cấp sự trừu tượng hóa tài nguyên cơ bản, được chuyển đổi thành dạng do phần mềm xác định để các thực thể phần mềm khác sử dụng. Phần mềm thực hiện ảo hóa cho phép nhiều người dùng đồng thời chia sẻ việc sử dụng một tài nguyên vật lý duy nhất mà không can thiệp lẫn nhau và thường không để họ biết về nhau. (Xem 5.5).

Một cách tiếp cận để ảo hóa các tài nguyên xử lý là sử dụng các máy ảo, bao gồm một trình giám sát máy ảo cung cấp phần cứng hệ thống trừu tượng và cho phép nhiều máy ảo chạy trên một hệ thống vật lý nhất định, với mỗi máy ảo chứa hệ điều hành khách của chính nó (OS khách), như thể hiện trong Hình 1. Điều này cho phép hệ thống được chia sẻ bởi các ứng dụng đang chạy trong mỗi VM.

Trình giám sát máy ảo thường là phần mềm được cài đặt và vận hành bởi CSP. Dịch vụ mây chạy VM cung cấp khả năng cho CSU tải phần mềm từ ảnh tượng VM và chạy phần mềm trong VM trên hệ thống CSP. VM được quản lý bởi trình giám sát máy ảo, nhưng điều này không được CSU nhìn thấy trực tiếp.

6.2 Máy ảo và ảo hóa hệ thống

Máy ảo (VM) là một môi trường thực thi biệt lập để chạy phần mềm sử dụng tài nguyên vật lý được ảo hóa. Nói cách khác, điều này liên quan đến việc ảo hóa hệ thống – và phần mềm trong mỗi VM được cấp quyền truy cập được kiểm soát cẩn thận vào các tài nguyên vật lý để cho phép chia sẻ các tài nguyên đó mà không bị can thiệp. Đôi khi được gọi là máy ảo hệ thống, máy ảo cung cấp chức năng cần thiết để thực thi ngăn xếp phần mềm hoàn chỉnh bao gồm toàn bộ hệ điều hành và mã ứng dụng sử dụng hệ điều hành (Xem 5.5.1). Điều này được mô tả bởi "OS khách" và "App x" trong mỗi VM được thể hiện trong Hình 1.

Mục đích của máy ảo là cho phép nhiều ứng dụng chạy cùng lúc trên một hệ thống phần cứng, trong khi các ứng dụng đó vẫn bị cô lập với nhau. Phần mềm chạy trong mỗi VM dường như có phần cứng hệ thống riêng, chẳng hạn như bộ xử lý, bộ nhớ thời gian chạy, (các) thiết bị lưu trữ và phần cứng mạng. Bị cô lập có nghĩa là phần mềm chạy trong một máy ảo được tách biệt và không biết phần mềm chạy trong các máy ảo khác trên cùng hệ thống và cũng được tách biệt khỏi hệ điều hành máy chủ. Ảo hóa thường có nghĩa là một tập hợp con của các tài nguyên vật lý có sẵn sàng thể được cung cấp cho mỗi VM, chẳng hạn như số lượng bộ xử lý hạn chế, RAM hạn chế, không gian lưu trữ hạn chế và quyền truy cập có kiểm soát vào các khả năng kết nối mạng.

Mỗi VM chứa một chồng phần mềm hoàn chỉnh, bắt đầu với hệ điều hành và tiếp tục với bất kỳ phần mềm nào khác được yêu cầu để chạy (các) ứng dụng được thực thi trong VM. Ngăn xếp phần mềm có thể rất đơn giản (ví dụ: ứng dụng gốc được viết bằng ngôn ngữ như C, chỉ sử dụng các chức năng do chính hệ điều hành cung cấp) hoặc phức tạp (ví dụ: ứng dụng được viết bằng ngôn ngữ như Java™ yêu cầu thời gian chạy và mở rộng việc sử dụng các thư viện và/hoặc dịch vụ không có trong hệ điều

hành và phải được cung cấp riêng).

Mỗi VM về nguyên tắc có thể chứa bất kỳ hệ điều hành nào. Các máy ảo khác nhau trên một hệ thống phần cứng duy nhất có thể chạy các hệ điều hành hoàn toàn khác nhau như Linux® và Windows®. Yêu cầu duy nhất là tất cả phần mềm chạy trong VM được thiết kế cho kiến trúc phần cứng của hệ thống bên dưới – phần cứng được ảo hóa nhưng không được mô phỏng. Vì vậy, ví dụ, mã được tạo cho bộ xử lý ARM sẽ không chạy trong máy ảo chạy trên hệ thống Intel x86.

6.3 Trình giám sát máy ảo

6.3.1 Quy định chung

Trình giám sát máy ảo, đôi khi được gọi là trình trình giám sát máy ảo, là phần mềm ảo hóa tài nguyên vật lý và cho phép chạy các máy ảo. Ảo hóa có nghĩa là kiểm soát sự trừu tượng của các tài nguyên vật lý cơ bản của hệ thống. Trình giám sát máy ảo cũng quản lý hoạt động của máy ảo. Trình giám sát máy ảo phân bổ tài nguyên cho từng máy ảo đang chạy bao gồm bộ xử lý (CPU), bộ nhớ, lưu trữ đĩa, khả năng kết nối mạng và băng thông.

Các trình giám sát máy ảo tồn tại dưới dạng một trong hai loại:

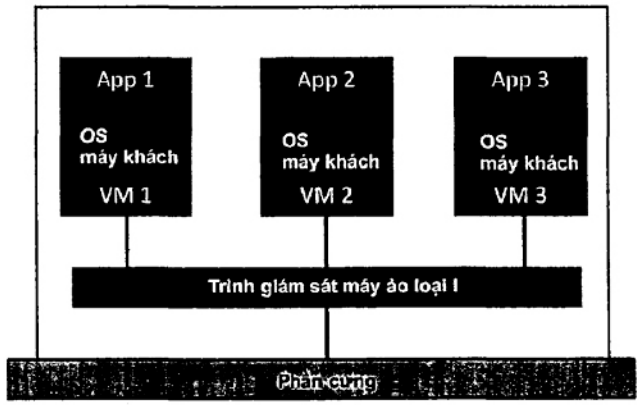
— "Kim loại trần", "gốc" hoặc "loại I";

— "Được nhúng", "được lưu trữ" hoặc "loại II".

Trình giám sát máy ảo loại I có thể nhanh hơn và hiệu quả hơn vì chúng không cần phải hoạt động thông qua hệ điều hành máy chủ. Trình giám sát máy ảo loại II có thể chậm hơn, nhưng có ưu điểm là thường dễ thiết lập hơn và tương thích với nhiều loại phần cứng hơn so với trình giám sát máy ảo loại I, vì các biến thể phần cứng phải được xử lý trong mã trình giám sát máy ảo loại I, trong khi trình giám sát máy ảo loại II tận dụng hỗ trợ phần cứng được tích hợp trong hệ điều hành máy chủ.

6.3.2 Các trình giám sát máy ảo loại I

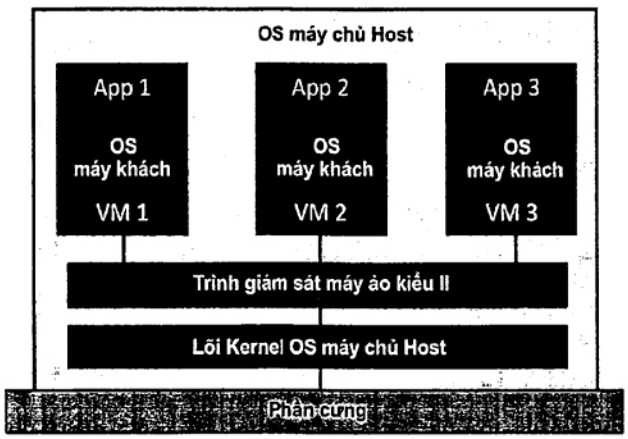
Trình giám sát máy ảo loại I chạy trực tiếp trên phần cứng hệ thống cơ bản và kiểm soát trực tiếp phần cứng đó cũng như quản lý máy ảo. Tổ chức của một hệ thống sử dụng trình giám sát máy ảo loại I được thể hiện trong Hình 1.



Hình 1 – Trình giám sát máy ảo loại I của phần cứng hệ thống

6.3.3 Các trình giám sát máy ảo loại II

Trình giám sát máy ảo loại II chạy trên hệ điều hành máy chủ, cụ thể hơn là nhân hệ điều hành máy chủ. Đó là hệ điều hành máy chủ điều khiển phần cứng hệ thống, trong khi trình giám sát máy ảo sử dụng các khả năng để chạy và quản lý máy ảo. Tổ chức của một hệ thống với trình giám sát máy ảo loại II được thể hiện trong Hình 2.



Hình 2 - Ảo hóa trình giám sát máy ảo loại II của phần cứng hệ thống

6.4 Bảo mật các VM và trình giám sát máy ảo

Đối với hệ thống phần cứng, hệ điều hành chạy ở mức đặc quyền cao nhất vì nó phải kiểm soát quyền truy cập vào tất cả các tài nguyên phần cứng. Tuy nhiên, trong máy chủ ảo hóa, vì trình giám sát máy ảo phải kiểm soát tất cả quyền truy cập vào CPU và bộ nhớ của các VM khách (cung cấp ảo hóa bộ xử lý và bộ nhớ), nên nó sẽ chạy ở mức đặc quyền cao hơn tất cả các VM. Để tạo điều kiện thuận lợi cho việc này, các trình giám sát máy ảo được cài đặt trên các hệ thống phần cứng cung cấp hỗ trợ cho ảo hóa. Cụ thể, hệ thống phần cứng cung cấp hai trạng thái bộ xử lý: chế độ gốc (trình giám sát máy ảo) và chế độ không gốc (khách). Tất cả các OS khách đều chạy ở chế độ không gốc trong khi chỉ riêng trình giám sát máy ảo chạy ở chế độ gốc.

Mặc dù có hỗ trợ phần cứng cho ảo hóa, quá trình cách ly thời gian chạy cho các VM do trình giám sát máy ảo cung cấp có thể bị phá hoại bởi các máy ảo giả mạo hoặc bị xâm nhập đã giành được quyền truy cập vào các vùng bộ nhớ thuộc về trình giám sát máy ảo hoặc các VM khác. VM giả mạo hoặc bị xâm nhập khai thác một số lỗ hổng thiết kế của trình giám sát máy ảo đối với một số cấu trúc phần mềm như khối điều khiển máy ảo (VMCB) và bảng trang bộ nhớ được trình giám sát máy ảo sử dụng để theo dõi trạng thái thực thi của các VM và ánh xạ bộ nhớ từ địa chỉ VM sang địa chỉ bộ nhớ máy chủ tương ứng. Những lỗ hổng này của trình giám sát máy ảo đã được biết đến trong một thời gian và do đó, nhiều lỗ hổng đã được giải quyết hoặc đang được giải quyết. Các phiên bản trình giám sát máy ảo gần đây đã được cập nhật và tăng cường. CSC và CSP nên kiểm tra xem bất kỳ trình giám sát máy ảo nào đang được sử dụng đều được cập nhật và tăng cường khả năng chống lại các lỗ hổng bảo mật đã biết.

Một hàm ý bảo mật khác trong nền tảng máy chủ ảo hóa bắt nguồn từ phần mềm được sử dụng để cung cấp ảo hóa thiết bị. Không giống như tập lệnh và ảo hóa bộ nhớ, ảo hóa thiết bị không được xử lý

trực tiếp bởi trình giám sát máy ảo mà bằng cách sử dụng các mô-đun phần mềm hỗ trợ. Các nguồn lỗ hổng chính bao gồm: (a) mã mô phỏng các thiết bị phần cứng vật lý chạy trong trình giám sát máy ảo dưới dạng mô-đun lõi có thể tải và (b) trình điều khiển thiết bị cho các thiết bị có khả năng truy cập bộ nhớ trực tiếp (DMA) có thể truy cập các vùng bộ nhớ thuộc về các VM khác hoặc thậm chí cả trình giám sát máy ảo.

Các tác động xuôi dòng tiềm ẩn của một VM lừa đảo chiếm quyền kiểm soát của trình giám sát máy ảo bao gồm việc cài đặt bộ phần mềm hoặc tấn công vào các VM khác trên cùng một máy chủ ảo hóa. Tất cả phần mềm ảo hóa thiết bị phải được xác minh để tránh các lỗi bảo mật trước khi cài đặt và sử dụng trên hệ thống sử dụng trình giám sát máy ảo và VM.

6.5 Định dạng, siêu dữ liệu và ảnh tượng VM

Ảnh tượng máy ảo (ảnh tượng VM) là một gói dữ liệu chứa thông tin và mã thực thi cần thiết để chạy một phiên bản của một VM. ảnh tượng VM được sử dụng để khởi tạo một phiên bản mới của VM, theo yêu cầu. ảnh tượng VM có thể bao gồm ngăn xếp phần mềm hoàn chỉnh cần thiết để chạy ứng dụng, bắt đầu với hệ điều hành, thư viện, thời gian chạy, chính mã ứng dụng, tệp cấu hình và siêu dữ liệu khác được ứng dụng sử dụng. ảnh tượng VM cũng có thể bao gồm siêu dữ liệu được liên kết với việc khởi tạo chính VM.

Siêu dữ liệu VM chứa thông tin về cấu hình và khởi động của VM. Điều này có thể bao gồm các thuộc tính của VM như kích thước RAM, các yêu cầu CPU, v.v. Siêu dữ liệu VM cũng thường tham chiếu các ảnh tượng đĩa có trong ảnh tượng VM, cụ thể là chỉ ra cách chúng được triển khai trong một phiên bản VM.

Khái niệm về ảnh tượng VM là nó phải chứa tất cả các thực thể cần thiết để chạy một trường hợp của VM, trong đó ảnh tượng VM được sử dụng làm dữ liệu đầu vào cho một trình giám sát máy ảo để cho phép nó tạo và khởi động VM. Nói chung, ảnh tượng máy ảo bao gồm hai bộ dữ liệu – thứ nhất là siêu dữ liệu VM và thứ hai là ảnh tượng đĩa. Điều quan trọng là phải hiểu rằng tồn tại nhiều định dạng khác nhau của cả siêu dữ liệu VM và ảnh tượng đĩa. Một trình giám sát máy ảo cụ thể được sử dụng để khởi tạo VM chỉ có thể hiểu các định dạng cụ thể cho siêu dữ liệu VM và ảnh tượng đĩa. Một số định dạng là độc quyền, trong khi những định dạng khác là mở hoặc chuẩn hóa. Xem Phụ lục A để biết thông tin về các định dạng ảnh tượng VM.

Ảnh tượng VM dựa trên dữ liệu được giữ trong các tệp – các tệp trên hệ thống tệp, được lưu trữ trong ảnh tượng VM dưới dạng một hoặc nhiều ảnh tượng đĩa. Các tệp này có thể là tệp của hệ điều hành, ứng dụng và bất kỳ phần nào khác của ngăn xếp phần mềm được yêu cầu. Có ít nhất một ảnh tượng đĩa, nhưng có thể có nhiều ảnh tượng đĩa nếu đây là cách tổ chức các tệp được ứng dụng và ngăn xếp phần mềm sử dụng. Thường xảy ra trường hợp khối lượng dữ liệu được giữ trong ảnh tượng đĩa là rất lớn và kết quả là các định dạng được sử dụng để lưu trữ dữ liệu liên quan đến việc sử dụng nén ở dạng này hay dạng khác.

Có nhiều định dạng ảnh tượng VM và ảnh tượng đĩa đang được sử dụng, một tỷ lệ đáng kể trong số đó là độc quyền hoặc là mã nguồn mở. Ví dụ về các định dạng ảnh tượng đĩa và ảnh tượng máy ảo được tiêu chuẩn hóa bao gồm:

— OVF ("Định dạng ảo hóa mở (Open Virtualization Format)") (xem ISO/IEC 17203:2017 [18])

  Gói OVF có một số tệp được đặt trong một thư mục. Có một tệp mô tả OVF (có phần mở rộng

.ovf) có nội dung định dạng XML mô tả máy ảo được đóng gói bao gồm siêu dữ liệu như tên, yêu cầu phần cứng và tham chiếu đến các tệp khác trong gói. Gói OVF cũng chứa một hoặc nhiều ảnh tượng đĩa, cộng với một số tệp tùy chọn, chẳng hạn như tệp chứng chỉ. Định dạng ảnh tượng OVF có phạm vi hỗ trợ tương đối rộng, trực tiếp hoặc thông qua các công cụ nhập/xuất.

— Định dạng đĩa ISO – định dạng lưu trữ được sử dụng cho nội dung đĩa quang (xem ISO 9660[72] và ISO/IEC 13346 [73])

ISO 9660 là một hệ thống tệp cho phương tiện đĩa quang, chủ yếu là CD-ROM.

ISO/IEC 13346 (còn được gọi là Định dạng đĩa chung hoặc UDF) thường được sử dụng trên các định dạng đĩa DVD và đĩa Blu-ray (BD) và đặc biệt phù hợp với phương tiện quang học có thể ghi và (ghi) lại.

Ảnh tượng đĩa thường được nén do kích thước lớn của chúng, mặc dù có những trường hợp ảnh tượng đĩa thô không nén được sử dụng để đạt được hiệu suất khởi động VM tốt hơn với chi phí tiêu tốn nhiều dung lượng hơn.

Các định dạng ảnh tượng VM và ảnh tượng đĩa nào được chấp nhận bởi một trình giám sát máy ảo cụ thể được nêu trong tài liệu dành cho trình giám sát máy ảo đó.

7 Vùng chứa và hệ thống quản lý vùng chứa (CMS)

7.1 Quy định chung

Vùng chứa là một công nghệ có thể cung cấp quá trình xử lý ảo hóa cho các dịch vụ mây. Công nghệ liên quan đến cả kiểu khả năng hạ tầng và kiểu khả năng nền tảng của dịch vụ mây như được mô tả trong TCVN 12480 (ISO/IEC 17788) và TCVN 12481 (ISO/IEC 17789).

Vùng chứa cung cấp môi trường thực thi phần mềm thông qua ảo hóa của lõi hệ điều hành chạy trên hệ thống. Các vùng chứa đại diện cho một cách tiếp cận khác đối với việc cung cấp môi trường thực thi phần mềm bằng cách sử dụng ảo hóa tài nguyên tính toán. Các vùng chứa liên quan đến việc ảo hóa của lõi hệ điều hành, so với việc ảo hóa phần cứng hệ thống liên quan đến các VM. Mục tiêu của các vùng chứa là cho phép nhiều bộ phần mềm khác nhau chạy trên một hệ thống cùng một lúc mà không can thiệp lẫn nhau, tức là chúng cho phép chia sẻ hệ thống một cách an toàn.

Dịch vụ mây hỗ trợ vùng chứa cung cấp khả năng cho CSU tải phần mềm từ ảnh tượng vùng chứa và chạy phần mềm đó trong vùng chứa trên hệ thống CSP. Vùng chứa được quản lý bởi CSP hoặc bởi CSU, tùy thuộc vào kiểu khả năng của dịch vụ mây. Trong cả hai trường hợp, thông thường việc quản lý được thực hiện bằng CMS (xem 7.4).

7.2 Vùng chứa và ảo hóa hệ điều hành

7.2.1 Mô tả vùng chứa

Vùng chứa là một môi trường thực thi biệt lập để chạy phần mềm sử dụng lõi hệ điều hành ảo hóa. Các vùng chứa chạy trong một hệ điều hành được gọi là hệ điều hành máy chủ (OS máy chủ).

Như được mô tả trong 6.2, một VM trình bày phiên bản ảo hóa của phần cứng hệ thống cho phần mềm trong VM. Quyền truy cập vào các tài nguyên phần cứng ảo hóa được thực hiện qua trung gian và phần mềm trong VM chỉ được xem và sử dụng phiên bản hạn chế và được kiểm soát cẩn thận của các

tài nguyên đó (ví dụ: số lượng CPU hạn chế, số lượng luồng hạn chế, RAM hạn chế).

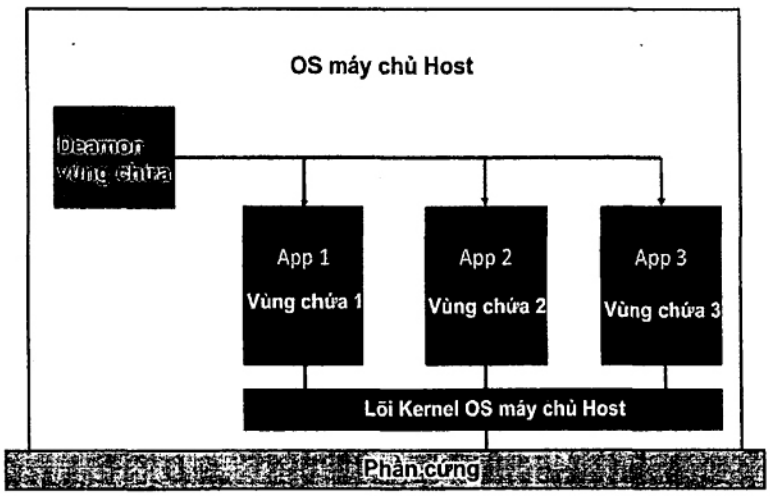
Theo cách tương tự, một vùng chứa trình bày một phiên bản ảo hóa của lõi OS máy chủ. Quyền truy cập vào các tài nguyên ảo hóa của lõi OS được dàn xếp và phần mềm trong vùng chứa sẽ được xem và sử dụng phiên bản các tài nguyên OS có giới hạn và được kiểm soát cẩn thận.

Sự cô lập của môi trường thực thi có nghĩa là phần mềm chạy trong một vùng chứa được tách biệt và không biết phần mềm đang chạy trong các vùng chứa khác, đồng thời cũng được tách biệt khỏi OS máy chủ. Phần mềm duy nhất chạy bên ngoài vùng chứa có thể truy cập hoặc ảnh tượng hưởng đến phần mềm chạy bên trong vùng chứa là Daemon vùng chứa.

Hình 3 chỉ ra ba vùng chứa đang chạy trên phần cứng hệ thống. Hệ thống vật lý có OS máy chủ riêng. Mỗi vùng chứa chứa phần mềm ứng dụng riêng (App x) và chạy phần mềm đó trong một hoặc nhiều quy trình OS bằng cách sử dụng các tài nguyên như bộ nhớ, CPU, bộ lưu trữ và kết nối mạng, tách biệt với các vùng chứa khác chạy trên cùng một hệ thống, nhưng tất cả đều chia sẻ lõi của OS máy chủ.

Lõi của OS máy chủ đang được chia sẻ bởi tất cả các vùng chứa, điều này về cơ bản có nghĩa là OS được phần mềm trong các vùng chứa sử dụng phải tương thích với lõi OS máy chủ. Điều này có thể cho phép các vùng chứa khác nhau có khả năng sử dụng các biến thể khác nhau của OS Linux trong đó OS máy chủ là một biến thể Linux, nhưng nó không cho phép sử dụng OS Windows trong một vùng chứa nếu OS máy chủ là Linux biến thể (và ngược lại).

Các vùng chứa được tạo lập và quản lý bởi một Daemon vùng chứa, chạy như một quy trình riêng biệt trong OS máy chủ.



Hình 3 - Ảo hóa vùng chứa

Ngăn xếp phần mềm chạy trong mỗi vùng chứa có thể khác nhau, nhưng thông thường, ngăn xếp này chứa chính ứng dụng ("App x") và mọi phụ thuộc phần mềm ứng dụng có. Về nguyên tắc, ngăn xếp phần mềm khá "tinh gọn", đặc biệt khi mã ứng dụng chỉ phụ thuộc vào các chức năng do lõi OS máy chủ cung cấp. Tuy nhiên, có trường hợp mã vùng chứa có thể bao gồm các thành phần của OS bên ngoài lõi, chẳng hạn như thư viện và tiện ích, đặc biệt nếu mã ứng dụng phụ thuộc vào phiên bản cụ thể của các thư viện và tiện ích này.

Lưu ý rằng hệ điều hành máy chủ được sử dụng bởi các vùng chứa có thể đang chạy bên trong một máy ảo, thay vì chạy trực tiếp trên phần cứng vật lý. Phần mềm chạy trong vùng chứa không biết hệ điều hành máy chủ có đang chạy trong máy ảo hay không.

7.2.2 Daemon vùng chứa

Trình nền vùng chứa là một dịch vụ phần mềm thực thi trên hệ điều hành máy chủ và chịu trách nhiệm tạo lập và quản lý các vùng chứa trên hệ thống đó. Một vùng chứa cụ thể là một trường hợp thực thi của ngăn xếp phần mềm được giữ trong một ảnh tượng vùng chứa (xem 7.3 để biết mô tả về các ảnh tượng vùng chứa). ảnh tượng vùng chứa bao gồm siêu dữ liệu và tham số được sử dụng bởi Daemon vùng chứa. Daemon vùng chứa sử dụng siêu dữ liệu vùng chứa để chỉ định các khả năng nhất định của vùng chứa và nó sử dụng các tham số dịch vụ vùng chứa để tác động đến cách một vùng chứa được khởi tạo và chạy.

Daemon vùng chứa cung cấp một giao diện dịch vụ mà qua đó các khả năng có thể được gọi bởi các ứng dụng khách. Các ứng dụng khách có thể chạy trên cùng một hệ thống với Daemon vùng chứa hoặc có thể chạy trên các hệ thống từ xa và gọi các khả năng của Daemon vùng chứa qua mạng.

Daemon vùng chứa cung cấp một tập các thao tác vùng chứa:

- **Tạo**: thao tác này tạo một vùng chứa mới. Vận hành tham chiếu ảnh tượng vùng chứa để sử dụng, được khởi tạo trong thư mục hệ thống tệp (được gọi là "gói") mà hệ điều hành máy chủ có thể truy cập. Việc tạo vùng chứa phân bổ một tập tài nguyên cho vùng chứa và cấu hình vùng chứa như được mô tả trong siêu dữ liệu vùng chứa được giữ trong gói. Vùng chứa được cung cấp một ID duy nhất, sau đó nó được tham chiếu.
- **Khởi động**: thao tác này khởi động vùng chứa bằng cách chạy chương trình ứng dụng được chỉ định cho vùng chứa, với bất kỳ tham số được cung cấp bởi siêu dữ liệu liên quan đến vùng chứa.
- **Chấm dứt**: thao tác này dừng (các) chương trình đang chạy trong vùng chứa. Điều này thường được thực hiện bằng cách gửi một tín hiệu cụ thể đến quy trình đang chạy trong vùng chứa.
- **Xóa bỏ**: thao tác này xóa các tài nguyên được phân bổ cho vùng chứa và hủy vùng chứa. ID duy nhất không còn định danh vùng chứa, mặc dù sau này có thể sử dụng cùng ID đó để tạo vùng chứa mới.

Daemon vùng chứa thường cũng cung cấp một giao diện sự kiện, cho phép Daemon vùng chứa báo cáo về các sự kiện quan trọng liên quan đến các vùng chứa mà nó quản lý. Giao diện sự kiện cho phép một hoặc nhiều thành phần phần mềm máy khách lắng nghe các sự kiện cụ thể và phản ứng với chúng.

7.2.3 Tài nguyên vùng chứa, cách ly và kiểm soát

Một vùng chứa cung cấp môi trường thực thi phần mềm, được cách ly và kiểm soát tài nguyên.

Cách ly có nghĩa là phần mềm chạy bên trong vùng chứa tạo ảo giác rằng nó có toàn bộ hệ thống – rằng (các) quy trình duy nhất tồn tại là những quy trình được bắt đầu bên trong vùng chứa. Trong thực tế, có thể có nhiều quy trình khác đang chạy trên cùng một hệ thống, nhưng phần mềm bên trong vùng chứa không thể nhận biết và không thể nhìn thấy hoặc tương tác với chúng.

Tài nguyên được kiểm soát có nghĩa là tập hợp các tài nguyên có sẵn cho phần mềm trong vùng chứa được phân bổ cho vùng chứa (bởi Daemon vùng chứa) khi vùng chứa được tạo và các tài nguyên này được theo dõi và giới hạn. Các tài nguyên này bao gồm phân bổ CPU, bộ nhớ thời gian chạy, mạng, (các) hệ thống tệp. Có vẻ như phần mềm trong vùng chứa chỉ tồn tại những tài nguyên này. Các tài nguyên được phân bổ theo cách mà các tài nguyên được phân bổ cho một vùng chứa không thể can thiệp vào các tài nguyên được phân bổ cho các vùng chứa khác đang chạy trên cùng một hệ thống.

Việc cách ly và kiểm soát tài nguyên được xử lý thông qua các khả năng của hệ điều hành máy chủ, được khai thác và quản lý bởi Daemon vùng chứa. Các khả năng chi tiết có sẵn và được sử dụng cho các vùng chứa khác nhau tùy theo hệ điều hành. Các khả năng được sử dụng trên hệ điều hành Linux được mô tả trong tiêu chuẩn này để minh họa. Tham khảo tài liệu liên quan đến các hệ điều hành khác để hiểu các khả năng tương đương.

Để có quyền truy cập vào khối I/O, theo mặc định, vùng chứa có quyền truy cập vào hệ thống tệp bao gồm ảnh tượng vùng chứa được sử dụng để tạo vùng chứa ở chế độ chỉ đọc, cùng với việc bổ sung lớp vùng chứa đọc/ghi. Hệ thống tệp mặc định là tạm thời và lớp vùng chứa bị xóa khi vùng chứa bị xóa. Các phương tiện lưu trữ bổ sung, thường là vĩnh viễn, có thể được cung cấp cho phần mềm bên trong vùng chứa thông qua việc cấu hình vùng chứa bởi Daemon vùng chứa, hoặc là gắn kết vào hệ thống tệp trong vùng chứa từ một số vị trí bên ngoài vùng chứa hoặc thông qua việc cung cấp một hoặc các dịch vụ lưu trữ cụ thể hơn (thường là dịch vụ lưu trữ mây). Nếu có nhu cầu về phần mềm trong vùng chứa để truy cập các đối tượng lưu trữ có thời gian tồn tại lâu dài, thì các phương tiện lưu trữ bổ sung như vậy là cần thiết. Trong mọi trường hợp, vị trí rõ ràng của các tệp và đối tượng lưu trữ trong vùng chứa được ánh xạ tới các vị trí thực tế bên ngoài vùng chứa.

Tương tự, để truy cập vào các khả năng kết nối mạng, các tài nguyên có sẵn cho mã đang chạy trong vùng chứa được kiểm soát bởi Daemon vùng chứa và cấu hình được áp dụng cho vùng chứa, tức là các khả năng kết nối mạng có thể cắm và có thể định cấu hình. Có thể không cung cấp khả năng mạng nào cho vùng chứa (tức là không có cổng bị lộ, không có thiết bị mạng nào khả dụng, do đó không có tuyến đường đến bất kỳ thiết bị đầu cuối mạng đích nào). Cũng có thể kiểm soát truy cập vào vùng chứa và truy cập từ vùng chứa một cách chi tiết.

Các cổng do vùng chứa hiển thị có thể được kiểm soát và có thể được ánh xạ giữa các địa chỉ mạng và cổng do vùng chứa hiển thị và những cổng có thể nhìn thấy bên ngoài. Ví dụ: vùng chứa có thể hiển thị cổng 80 cho lưu lượng HTTP nhưng cổng này có thể được ánh xạ tới cổng 8080 để truy cập bên ngoài (ví dụ: internet). Nói chung, địa chỉ IP, cổng, tên máy chủ, địa chỉ MAC, dịch vụ định tuyến và dịch vụ DNS có thể được kiểm soát và ánh xạ.

Một hình thức kết nối mạng quan trọng được sử dụng với các vùng chứa là nơi mạng kết nối độc quyền một tập hợp các vùng chứa có liên quan, ví dụ: các vùng chứa đại diện cho một ứng dụng duy nhất được triển khai bằng kiến trúc vi dịch vụ với các thành phần khác nhau của ứng dụng chạy trong các vùng chứa khác nhau. Đây là một dạng mạng ảo, trong đó chỉ các vùng chứa được chỉ định mới có thể nói chuyện với nhau (không phải bất kỳ thiết bị đầu cuối cụ thể nào được thể hiện bên ngoài) như thể chúng là các thực thể duy nhất trên mạng. Mạng có thể trải rộng trên các hệ thống khác nhau và cả

trên các mạng cơ bản khác nhau, cho phép tính linh hoạt cao ở vị trí của từng vùng chứa, tức là tính minh bạch của vị trí vùng chứa được cung cấp trong khi vẫn có khả năng kiểm soát và hạn chế liên lạc vì mục đích bảo mật.

Trên Linux, quyền kiểm soát tài nguyên được xử lý thông qua một khả năng được gọi là nhóm kiểm soát hoặc cgroup. cgroups cung cấp quyền kiểm soát các tài nguyên có sẵn cho các bộ quy trình, bao gồm CPU, bộ nhớ, I/O để chặn thiết bị (tức là hệ thống tệp), quyền truy cập vào thiết bị, kết nối mạng.

Trên Linux, cách ly được thực hiện thông qua các không gian tên. Trên thực tế, bất kỳ tài nguyên nào được truy cập bởi một vùng chứa đều là một phần của một vùng tên, trong khi các tài nguyên được các vùng chứa khác truy cập được phân bổ cho các vùng tên khác. Các không gian tên hoạt động theo cách mà phần mềm đang chạy trong một quy trình được bắt đầu dưới một không gian tên chỉ có thể nhìn thấy các tài nguyên trong không gian tên đó.

Các loại không gian tên sau tồn tại trong Linux (kể từ nhân Linux 4.10):

— Giao tiếp liên tiến trình (ipc): liên quan đến giao tiếp liên quá trình. Chỉ các quy trình trong cùng một không gian tên mới có thể thiết lập giao tiếp (ví dụ: cấp phát bộ nhớ dùng chung).

— Mount (mnt): liên quan đến các điểm mount, tức là những nơi hệ thống tệp (bổ sung) được gắn kết. Một bộ giá treo ban đầu khả dụng khi vùng chứa được tạo bởi Daemon vùng chứa, nhưng sau đó, bất kỳ giá treo mới nào chỉ hiển thị trong vùng chứa.

— Mạng (net): chứa các tài nguyên liên quan đến mạng như giao diện, địa chỉ IP, bảng định tuyến, danh sách ổ cắm, bảng theo dõi kết nối, v.v...

— ID tiến trình (pid): chứa một tập ID tiến trình – tiến trình đầu tiên trong không gian tên có id số 1 – và tiến trình này có cách xử lý đặc biệt tương đương với tiến trình duy nhất trên hệ điều hành cơ bản.

— ID người dùng (người dùng): cung cấp ID người dùng cho phép cả nhận dạng người dùng và cả kiểm soát đặc quyền – không gian tên người dùng ánh xạ ID người dùng trong không gian tên với ID người dùng trong hệ thống cơ sở – điều này cho phép kiểm soát chặt chẽ các đặc quyền và có thể cung cấp các đặc quyền cao hơn trong không gian tên không được cung cấp cho bất kỳ tài nguyên nào bên ngoài không gian tên đó.

— UTS: cho phép các quy trình khác nhau có vẻ như có các tên máy chủ và tên miền khác nhau.

Sự kết hợp của các nhóm và không gian tên cùng nhau cung cấp khả năng kiểm soát và cách ly tài nguyên cần thiết cho các vùng chứa.

7.3 Ảnh tượng vùng chứa và phân lớp hệ thống tệp

7.3.1 Mục đích và nội dung ảnh tượng

Ảnh tượng vùng chứa là một gói thực thi chứa mọi thứ cần thiết để chạy phần mềm, chẳng hạn như ứng dụng hoặc vi dịch vụ. Điều này có thể bao gồm mã của chính ứng dụng, thời gian chạy, thư viện, biến môi trường, tệp cấu hình và siêu dữ liệu khác được ứng dụng sử dụng. Mục đích là ảnh tượng vùng chứa tự mô tả và được đóng gói để Daemon vùng chứa có thể lấy ảnh tượng vùng chứa và tạo một vùng chứa từ nó mà không cần phụ thuộc thêm và bất kể hệ thống bên dưới ("bất khả tri về hạ tầng") và bất kể nội dung của ảnh tượng vùng chứa ("nội dung bất khả tri").

Ảnh tượng vùng chứa chứa các tập hợp tệp đại diện cho mã của ứng dụng, phần phụ thuộc cũng như

các tệp và siêu dữ liệu khác được ứng dụng sử dụng. ảnh tượng vùng chứa cũng chứa siêu dữ liệu có cấu trúc về chính nội dung của ảnh tượng vùng chứa và cách chuyển đổi những nội dung đó thành vùng chứa. Siêu dữ liệu vùng chứa có thể khác nhau tùy thuộc vào định dạng ảnh tượng vùng chứa cụ thể có liên quan. Siêu dữ liệu vùng chứa được mô tả trong đặc tả định dạng ảnh tượng OCI [9] bao gồm:

— Chỉ số ảnh tượng. Siêu dữ liệu "mức cao nhất" có mục đích hỗ trợ ảnh tượng vùng chứa hỗ trợ nhiều nền tảng khác nhau (điều này đôi khi được gọi là bản kê FAT). Khi nhiều nền tảng được hỗ trợ, mỗi nền tảng có ảnh tượng cụ thể của riêng nó chứa các tạo tác được sử dụng khi chạy vùng chứa trên nền tảng đó. Thực tế, chỉ mục ảnh tượng tham chiếu một hoặc nhiều bản kê ảnh tượng.

— Bản kê ảnh tượng. Chứa thông tin cho một ảnh tượng vùng chứa duy nhất cho kiến trúc CPU và hệ điều hành cụ thể, bao gồm cấu hình và tập hợp các lớp.

— Bố cục ảnh tượng. Bố cục cụ thể của các thư mục và tệp trong ảnh tượng với siêu dữ liệu về các lớp hệ thống tệp.

— Các lớp hệ thống tệp. Một hoặc nhiều hệ thống tệp được tuần tự hóa (tức là cấu trúc của thư mục và tệp) và thay đổi hệ thống tệp (tệp đã xóa hoặc cập nhật). Các lớp được áp dụng chồng lên nhau để tạo một hệ thống tệp hoàn chỉnh trong vùng chứa đang chạy. (Xem 7.3.2 để biết mô tả về phân lớp hệ thống tệp).

Chức năng đằng sau chỉ số ảnh tượng hoặc bản kê FAT cho phép một ảnh tượng vùng chứa duy nhất cung cấp hỗ trợ cho các ảnh tượng cụ thể của nền tảng. Nền tảng có thể bao gồm loại CPU (kiến trúc máy), loại hệ điều hành và cấp hệ điều hành tiềm năng. Do đó, một ảnh tượng vùng chứa duy nhất có thể được cấu trúc để cho phép bàn giao và triển khai cùng một ứng dụng cho một số hệ thống đích khác nhau.

Một trong những đặc trưng điển hình của siêu dữ liệu vùng chứa có trong ảnh tượng vùng chứa là nó cung cấp các tính năng bảo mật mở rộng nhằm đảm bảo rằng nội dung của ảnh tượng vùng chứa không bị giả mạo kể từ khi được tạo. Độ dài dữ liệu được ghi lại, cùng với các bản tóm tắt dữ liệu (về cơ bản là hàm băm mật mã chống va chạm của các byte dữ liệu). Dữ liệu liên quan có thể là nội dung của các lớp hệ thống tệp hoặc các thành phần của siêu dữ liệu. Thông báo cũng có thể đóng vai trò là mã định danh duy nhất cho nội dung, cũng có thể được sử dụng để hỗ trợ quyền truy cập theo địa chỉ nội dung vào dữ liệu. Giao tiếp an toàn riêng biệt của bản tóm tắt tới người dùng ảnh tượng vùng chứa cho phép xác minh nội dung của ảnh tượng vùng chứa ngay cả khi nó được truy xuất từ một nguồn không đáng tin cậy.

7.3.2 Phân lớp hệ thống tệp

Ảnh tượng vùng chứa và các vùng chứa được tạo ra từ chúng, sử dụng công nghệ phân lớp hệ thống tệp khi xử lý các tệp mà chúng chứa.

Phân lớp hệ thống tệp là một cách tiếp cận để tạo nội dung của hệ thống tệp được vùng chứa sử dụng. Nguyên tắc là nội dung hệ thống tệp được xây dựng dưới dạng một chồng các lớp, mỗi lớp chứa một số tập hợp thư mục và tệp, tất cả đều có một thư mục gốc chung. Các thư mục và tệp lần lượt được

đóng góp từ mỗi lớp, bắt đầu với lớp cơ sở và tiếp tục đi lên thông qua chồng các lớp. Mỗi lớp tiếp theo có thể đóng góp các tệp mới, nhưng cũng có thể thay thế một tệp ở lớp thấp hơn bằng một phiên bản khác hoặc nó có thể xóa ("xóa sổ") một tệp có trong lớp thấp hơn.

Phân lớp hệ thống tệp cho phép xử lý hiệu quả các tệp trong ảnh tượng vùng chứa. Phân lớp hệ thống tệp cũng là một cách tiếp cận thực tế để tạo ảnh tượng vùng chứa, với điều kiện là các ứng dụng điển hình có ngăn xếp phần mềm dưới dạng phụ thuộc cho phép chúng chạy.

Hãy xem xét ví dụ về ứng dụng node.js. Mã của ứng dụng node.js có thể được chứa trong tệp tập lệnh app.js, cùng với các tệp tập lệnh, tệp dữ liệu và tệp cấu hình khác. Ứng dụng node.js có sự phụ thuộc vào thời gian chạy node.js cộng với một số gói (bên ngoài). Đổi lại, thời gian chạy node.js phụ thuộc vào các thư viện hệ điều hành khác nhau.

Trong ảnh tượng vùng chứa cho ứng dụng node.js, điều này có thể được thể hiện bằng 3 lớp (bắt đầu từ lớp trên cùng):

— tập lệnh app.js và các thành phần tạo tác được liên kết tạo thành lớp trên cùng;

— thời gian chạy node.js và các gói liên quan tạo thành lớp tiếp theo;

— các thư viện hệ điều hành tạo thành lớp dưới cùng.

Lưu ý rằng lõi hệ điều hành không cần phải có trong ảnh tượng vùng chứa. Lõi hệ điều hành được cung cấp bởi hệ điều hành máy chủ mà các vùng chứa chạy trên đó.

Phân lớp phản ánh một quy trình hiệu quả để xây dựng (tạo) ảnh tượng vùng chứa. Mặc dù có thể tạo một ảnh tượng vùng chứa với một lớp duy nhất chứa tất cả các tệp cần thiết, nhưng việc tách ngăn xếp phần mềm được ứng dụng sử dụng thành các lớp riêng biệt sẽ hiệu quả hơn nhiều, vì ứng dụng và từng phần phụ thuộc thường độc lập riêng biệt. bộ tệp, như được mô tả cho nút. ví dụ về ứng dụng js.

Một ảnh tượng vùng chứa có thể được tạo bằng cách sử dụng ảnh tượng vùng chứa khác làm cơ sở (hoặc "gốc") khác. Do đó, bằng cách sử dụng lại ví dụ về ứng dụng node.js, ảnh tượng vùng chứa đầu tiên được tạo có thể là ảnh tượng cho hệ điều hành. Sau đó, một ảnh tượng vùng chứa thứ hai có thể được tạo cho thời gian chạy node.js và các gói liên quan, sử dụng ảnh tượng hệ điều hành làm cha. Cuối cùng, ảnh tượng vùng chứa thứ ba có thể được tạo cho ứng dụng. js sử dụng thời gian chạy node.js làm ứng dụng gốc. Mỗi ảnh tượng gốc cung cấp các lớp thấp hơn cho ảnh tượng mới được tạo ở trên cùng. Do đó, trong ví dụ này, các thư viện hệ điều hành trở thành lớp thấp nhất, thời gian chạy node.js ở lớp giữa và ứng dụng app.js ở lớp trên cùng.

Điều này cho phép mỗi ảnh tượng chỉ quan tâm đến nhu cầu của chính nó. Ví dụ: nếu thời gian chạy node.js không cần tất cả các tệp từ thư viện hệ điều hành, nó có thể xóa các tệp không cần thiết. Do đó, mối quan tâm chính khi xây dựng ảnh tượng vùng chứa trở thành câu hỏi nên sử dụng (những) ảnh tượng cơ sở nào.

Phân lớp hệ thống tệp cũng áp dụng cho các vùng chứa, với một sự thay đổi. Khi vùng chứa được khởi tạo từ ảnh tượng vùng chứa, các lớp hệ thống tệp tương tự được tạo như trong ảnh tượng vùng

chứa, nhưng chúng được coi là chỉ đọc. Các lớp này được gọi là các lớp ảnh tượng. Ứng dụng đang chạy trong vùng chứa không thể sửa đổi các tệp trong các lớp ảnh tượng. Tuy nhiên, một lớp có thể ghi bổ sung được thêm vào trên cùng của các lớp có trong ảnh tượng vùng chứa – đây được gọi là lớp vùng chứa (được gọi là lớp hộp cát trong một số môi trường vùng chứa). Tất cả các thay đổi đối với tệp do ứng dụng chạy trong vùng chứa thực hiện đều được ghi vào lớp vùng chứa, cho dù tạo tệp mới, sửa đổi tệp hiện có hay xóa tệp. Điều này ngụ ý một chiến lược sao chép khi ghi cho các tệp trong vùng chứa.

Hệ quả của các lớp ảnh tượng chỉ đọc và chiến lược sao chép khi ghi là các lớp ảnh tượng có thể *được chia sẻ giữa các vùng chứa khác nhau*, tiết kiệm bộ nhớ lưu trữ và thời gian chạy, đồng thời giảm thời gian khởi động cho các vùng chứa.

7.3.3 Kho lưu trữ và sổ lưu ký ảnh tượng vùng chứa

Khả năng lưu trữ và truy cập ảnh tượng vùng chứa là một khía cạnh quan trọng của hệ sinh thái vùng chứa. Các ảnh tượng vùng chứa riêng lẻ thường được sử dụng trong nhiều hệ thống khác nhau, chẳng hạn như để hỗ trợ khả năng mở rộng và kích hoạt dự phòng để cải thiện tính khả dụng. Quy trình được đề xuất để xây dựng ảnh tượng cũng nhấn mạnh vào việc truy cập các ảnh tượng hiện có tạo thành ảnh tượng gốc cho ảnh tượng đang được xây dựng.

Tái sử dụng được khuyến khích trong hệ sinh thái vùng chứa. ảnh tượng vùng chứa cho các thành phần chung của (các) ngăn xếp phần mềm được ứng dụng sử dụng được sử dụng làm ảnh tượng gốc cho nhiều ứng dụng. Các ví dụ điển hình là những ví dụ dành cho thư viện hệ điều hành và những ví dụ dành cho phần mềm trung gian và thời gian chạy. Rất có khả năng các ảnh tượng vùng chứa cho các gói phần mềm này sẽ được (tái) sử dụng lại nhiều lần trong các ảnh tượng vùng chứa cho các ứng dụng sử dụng các gói phần mềm đó trong ngăn xếp phần mềm của chúng.

Việc sử dụng lại ảnh tượng vùng chứa do người có chuyên môn về gói phần mềm liên quan tạo ra thường tốt hơn và đỡ tốn công hơn là tạo ảnh tượng mới cho gói phần mềm đó. Ngoài ra, những ảnh tượng như vậy thường được cập nhật với các bản sửa đổi đối với phần mềm cơ bản.

Việc cung cấp khả năng lưu trữ và truy cập ảnh tượng vùng chứa là trách nhiệm của cơ quan *sổ lưu ký vùng chứa*. Cơ quan sổ lưu ký vùng chứa có thể được cung cấp dưới dạng dịch vụ mây công cộng hoặc có thể được cung cấp dưới dạng dịch vụ mây riêng. Một ví dụ về dịch vụ sổ lưu ký vùng chứa công cộng là Docker Hub [80]. Cơ quan sổ lưu ký vùng chứa có giao diện dịch vụ ít nhất cung cấp cho các hoạt động đẩy và kéo. Thao tác đẩy tải một hoặc nhiều ảnh tượng lên sổ lưu ký trong khi thao tác kéo tải xuống một hoặc nhiều ảnh tượng từ sổ lưu ký.

Kho lưu trữ là một tập hợp các ảnh tượng vùng chứa có liên quan. Một ví dụ về tập hợp các ảnh tượng vùng chứa có liên quan là một tập hợp các ảnh tượng vùng chứa cho các thư viện hệ điều hành dành cho một hệ điều hành cụ thể, trong đó mỗi ảnh tượng dành cho một phiên bản cụ thể của hệ điều hành đó.

Ví dụ: kho lưu trữ vùng chứa có thể chứa một bộ bốn ảnh tượng vùng chứa có tên *some_os_libs:16.01, some_os_libs:16.02, some_os_libs:16.03, some_os_libs:latest*. Trong trường hợp

(đơn giản) này, kho lưu trữ có bốn mục cho các phiên bản khác nhau của hệ điều hành, mỗi mục có một thẻ cho biết số phiên bản. Phiên bản được gắn thẻ "mới nhất" thực sự trỏ đến cùng một ảnh tượng vùng chứa với phiên bản được gắn thẻ "16.03".

Lý do cho sự sắp xếp này là khi các ảnh tượng vùng chứa khác muốn luôn sử dụng phiên bản mới nhất của ảnh tượng vùng chứa hệ điều hành làm cha, chúng có thể sử dụng thẻ "mới nhất" khi truy xuất ảnh tượng và điều này sẽ tự động được cập nhật khi ảnh tượng vùng chứa mới của các phiên bản mới hơn của hệ điều hành được tải lên sổ sổ lưu ký vùng chứa. Các cách sử dụng khác cho thẻ được áp dụng cho kho lưu trữ ảnh tượng là để làm cho ảnh tượng được thiết kế cho các môi trường mục tiêu cụ thể, mặc dù ảnh tượng bản kê FAT là một cách tiếp cận khác để đạt được khả năng này.

7.4 Hệ thống quản lý vùng chứa (CMS)

7.4.1 Quy định chung

Như được mô tả trong 7.2.2, Daemon vùng chứa điển hình và các công cụ liên quan cung cấp khả năng quản lý vòng đời của một vùng chứa đơn. Tuy nhiên, việc triển khai các ứng dụng tính toán mây điển hình thường liên quan đến việc triển khai và vận hành nhiều vùng chứa thường trên nhiều hệ thống máy chủ. Một ứng dụng có thể liên quan đến nhiều trường hợp của một vùng chứa cụ thể chạy song song, vừa để cung cấp khả năng dự phòng chống lại sự cố của một trường hợp, vừa để cung cấp khả năng mở rộng để xử lý khối lượng công việc của các yêu cầu tương lai. Một ứng dụng cũng có thể bao gồm nhiều thành phần khác nhau, với mỗi thành phần chạy trong (các) trường hợp vùng chứa riêng, thông qua việc sử dụng kiến trúc vi dịch vụ hoặc phân tách các khả năng trong nhiều tầng, chẳng hạn như ứng dụng web sử dụng cơ sở dữ liệu. Một CMS sắp xếp và quản lý các vùng chứa đã xác định.

CMS có thể trừu tượng hóa hạ tầng cơ bản, coi tập các vùng chứa là một mục tiêu triển khai duy nhất, đồng thời thực thi các chính sách để triển khai, chẳng hạn như tách các trường hợp vùng chứa song song cho mục đích dự phòng và chuyển đổi dự phòng.

Nhiều CMS khác nhau có sẵn và được sử dụng phổ biến, bao gồm Docker Swarm [79], Kubernetes [17], Apache Mesos [57], HashiCorp Nomad [58], và CloudFoundry (một hệ thống PaaS) [59].

7.4.2 Các khả năng CMS phổ biến

Các khả năng phổ biến của một CMS bao gồm:

a) Điều phối

Các CMS cung cấp sự điều phối các trường hợp vùng chứa, bao gồm việc tạo và sắp xếp ban đầu, lập lịch biểu, giám sát, mở rộng quy mô, cập nhật và triển khai song song các khả năng như bộ cân bằng tải, tường lửa, mạng ảo và khả năng ghi nhật ký.

Về bản chất, các công cụ điều phối CMS có thể trừu tượng hóa hạ tầng cơ bản, coi tập các vùng chứa là một mục tiêu triển khai duy nhất, đồng thời thực thi các chính sách để triển khai, chẳng hạn như tách các trường hợp vùng chứa song song cho mục đích dự phòng và chuyển đổi dự phòng.

Điều phối là thành phần chính mà CMS yêu cầu để hỗ trợ quy mô, vì quy mô yêu cầu tự động hóa hiệu quả.

b) Lập lịch biểu

Người lập lịch biểu đảm bảo rằng các nhu cầu về tài nguyên được đặt trên hạ tầng có thể được đáp ứng toàn thời gian. Người lập lịch biểu chọn nút dựa trên đánh giá về tính khả dụng của tài nguyên, sau đó theo dõi việc sử dụng tài nguyên để đảm bảo thành phần không vượt quá mức phân bổ. Nó duy trì và theo dõi các yêu cầu về tài nguyên, tính khả dụng của tài nguyên và một loạt các ràng buộc và chỉ thị chính sách do người dùng cung cấp khác.

c) Giám sát, kiểm tra tình trạng

Tự động hóa là bản chất của các hệ thống tính toán mây, đặc biệt là khi có liên quan đến khả năng chịu lỗi và khả năng mở rộng nhanh chóng. Việc tự động hóa như vậy chỉ có thể được cung cấp cho các hệ thống sản xuất bằng cách CMS liên tục giám sát tập các vùng chứa được phân phối của ứng dụng và đánh giá tình trạng của chúng.

Khả năng này cho phép phát hiện lỗi, sai lỗi và thực hiện các hành động để đảm bảo rằng cấu hình mong muốn của ứng dụng được duy trì, như được xác định trong cấu hình khai báo. Điều này có thể liên quan đến việc loại bỏ các trường hợp bị lỗi và bắt đầu các trường hợp mới.

d) Tự động tính tỷ lệ

Giám sát có thể hỗ trợ điều chỉnh quy mô động của các tài nguyên được áp dụng cho ứng dụng để phù hợp với khối lượng công việc sắp tới, với mục đích giữ cho các tài nguyên được triển khai ở mức tối thiểu cần thiết để phục vụ tải, vì tính toán mây thường tính cơ sở các tài nguyên được sử dụng.

e) Quản lý tài nguyên

Trong một CMS, tài nguyên là một cấu trúc logic mà bộ điều phối có thể khởi tạo lập và quản lý, chẳng hạn như triển khai dịch vụ hoặc ứng dụng.

f) Kết nối mạng (Ảo)

Một ứng dụng điển hình bao gồm nhiều thành phần riêng biệt hoạt động cùng nhau để cung cấp chức năng của ứng dụng. Các thành phần riêng biệt thường cần giao tiếp với nhau thông qua mạng, vì các thành phần thường chạy ở các vị trí khác nhau. CMS chịu trách nhiệm thiết lập mạng cần thiết để cho phép các thành phần giao tiếp. Mạng thường là mạng ảo, loại bỏ nhu cầu các thành phần hiểu hạ tầng mạng cơ bản và cũng cải thiện bảo mật bằng cách hạn chế giao tiếp với các thành phần đó thuộc ứng dụng.

g) Khám phá dịch vụ

Khám phá là một yếu tố chính liên quan đến việc triển khai vùng chứa – các ứng dụng bao gồm nhiều vùng chứa và các thành phần được liên kết chạy trên hạ tầng có khả năng bàn giao rộng rãi. Do đó, các thành phần riêng lẻ cần khám phá các thành phần khác mà chúng phụ thuộc vào.

Ví dụ: bộ cân bằng tải cần xác định tất cả các trường hợp thành phần mà nó đang sử dụng để bàn giao các yêu cầu đến. CMS thường cung cấp các khả năng hỗ trợ khám phá.

h) Cập nhật và nâng cấp

Các CMS thường quản lý quá trình cập nhật và nâng cấp các thành phần của ứng dụng. Những thay đổi như vậy có thể là kết quả của chức năng mới hoặc bản sửa lỗi cho chính mã ứng dụng hoặc có thể là kết quả của các bản cập nhật cho ngăn xếp phần mềm được sử dụng bởi mã ứng dụng, chẳng hạn như thời gian chạy. CMS thường quản lý việc nâng cấp để không có thời gian ngừng hoạt động, thông qua việc giới thiệu theo giai đoạn các phiên bản sử dụng mã cập nhật và loại bỏ các phiên bản sử dụng mã cũ hơn.

i) Cấu hình khai báo

Các CMS thường cung cấp phương tiện để nhóm DevOps định cấu hình điều phối cho một ứng dụng theo cách khai báo, sử dụng lược đồ xác định được viết bằng ngôn ngữ như YAML [81] và JSON [82]. Cấu hình khai báo thường cũng chứa thông tin cần thiết về kho chứa, cấu hình mạng, phương tiện lưu trữ và khả năng bảo mật hỗ trợ ứng dụng. Cấu hình khai báo rất cần thiết trong việc cho phép các CMS tự động hóa quy trình quản lý ứng dụng và các thành phần. Về bản chất, cấu hình khai báo chỉ ra cho CMS cấu hình mong muốn và CMS nhằm mục đích tạo và sau đó duy trì cấu hình đó, quyết định các hành động cần thiết để đạt được điều này bằng cách sử dụng kiến thức về hệ thống đích và chiến lược triển khai nội bộ.

8 Tính toán không server

8.1 Quy định chung

Tính toán không server là một thể loại dịch vụ mây trong đó CSC có thể sử dụng các kiểu khả năng mây khác nhau mà CSC không phải cung cấp, triển khai và quản lý tài nguyên phần cứng hoặc phần mềm, ngoài việc cung cấp mã ứng dụng CSC hoặc cung cấp dữ liệu CSC. Tính toán không server cung cấp tính năng tự động mở rộng quy mô với khả năng phân bổ tài nguyên linh hoạt của CSP, bàn giao tự động trên nhiều địa điểm cũng như bảo trì và sao lưu tự động. Các khả năng tính toán không server được kích hoạt bởi một hoặc nhiều sự kiện do CSC xác định và thực thi trong một khoảng thời gian giới hạn theo yêu cầu để xử lý từng sự kiện. Các khả năng không server có thể được gọi bằng cách gọi trực tiếp từ các ứng dụng web và di động.

Khái niệm cơ bản đằng sau tính toán không server bắt đầu với các chức năng như một dịch vụ, trong đó ý tưởng là để CSP phân bổ các tài nguyên thời gian chạy phù hợp cần thiết cho mã ứng dụng CSC theo yêu cầu mà CSC không cần phải phân bổ trước và quản lý các máy cụ thể, VM hoặc vùng chứa và bất kỳ chồng phần mềm liên quan. Các loại dịch vụ mây khác cũng có sẵn theo mô hình tính toán không server, đáng chú ý là cơ sở dữ liệu không server.

Một cách khác để mô tả tính toán không server: đây là một dạng thể loại dịch vụ mây nền tảng (hoặc PaaS), vì chỉ có mã ứng dụng và/hoặc chính dữ liệu được cung cấp bởi CSC, trong khi tất cả các tài nguyên và khả năng khác cần thiết để chạy ứng dụng là được cung cấp và quản lý bởi CSP.

Thông thường, các dịch vụ mây thuộc thể loại tính toán không server tự động mở rộng quy mô để xử lý

các yêu cầu gửi đến. Các dịch vụ mây thuộc thể loại tính toán không server thường có khả năng chịu lỗi, chẳng hạn như khả năng đặt mã ứng dụng CSC ở nhiều vị trí với khả năng chuyển đổi dự phòng tự động khi xảy ra lỗi.

Tính toán không server vẫn cần máy chủ để chạy, vì vậy theo nghĩa đó, cái tên này là một cách gọi sai. Điều không bắt buộc là CSC phân bổ và quản lý tài nguyên máy chủ.

Tính toán không server thường có mô hình tính phí tính phí cho công việc được thực hiện bởi dịch vụ mây theo kiểu chi tiết, thay vì tính phí cho các tài nguyên được phân bổ (ví dụ: VM hoặc vùng chứa). Do đó, tính phí có thể theo lệnh gọi API hoặc theo yêu cầu HTTP chẳng hạn. Đây có thể được coi là một hình thức tính phí "trả tiền khi bạn sử dụng" cực đoan hơn.

Lợi ích của tính toán serverless đối với CSC có thể bao gồm:

— giảm chi phí vận hành, đặc biệt là chi phí thấp hơn liên quan đến việc mở rộng quy mô để đáp ứng các khối lượng công việc khác nhau, vì việc tính phí liên quan trực tiếp đến công việc được thực hiện bởi dịch vụ mây, thay vì chi phí liên quan đến tài nguyên được phân bổ. Ngoài ra, ngăn xếp phần mềm bắt buộc không cần xử lý như trường hợp điển hình khi sử dụng máy ảo và vùng chứa.

— giảm chi phí phát triển và giảm thời gian phát triển, do các nhà phát triển không cần quan tâm đến bất kỳ khía cạnh nào của quản lý tài nguyên bao gồm triển khai và mở rộng quy mô, ví dụ: mã ứng dụng có thể được tải lên và thực thi một cách đơn giản.

— độ phức tạp triển khai và đóng gói thấp hơn. Đặc biệt, không cần xem xét bất cứ điều gì ngoại trừ mã ứng dụng CSC hoặc dữ liệu CSC. Mọi ngăn xếp phần mềm liên quan đều được CSP cung cấp như một phần của dịch vụ mây và không được CSC đóng gói hoặc triển khai.

8.2 Chức năng như một dịch vụ

8.2.1 Tổng quan

Một hình thức phổ biến của tính toán không server là các chức năng như một dịch vụ (FaaS). FaaS là một dạng tính toán không server, trong đó khả năng được CSC sử dụng là thực thi mã ứng dụng CSC, dưới dạng một hoặc nhiều chức năng, mỗi chức năng được kích hoạt bởi một sự kiện do CSC chỉ định. FaaS cũng là một dạng Tính toán dưới dạng Dịch vụ (CompaaS) như được định nghĩa trong TCVN 12480 (ISO/IEC 17788).

FaaS có thể thực thi mã ứng dụng của khách hàng được viết bằng một hoặc nhiều ngôn ngữ lập trình. Mỗi dịch vụ mây FaaS hỗ trợ các ứng dụng được viết bằng một hoặc nhiều ngôn ngữ lập trình bao gồm nhưng không giới hạn ở C#, Go, JavaTM, JavaScript (node.js [78]), PHP, Python, Ruby, Swift. FaaS thể hiện khả năng của nền tảng như một dịch vụ trong việc cung cấp ngăn xếp phần mềm thời gian chạy theo yêu cầu của mã ứng dụng, nghĩa là CSC không bắt buộc phải triển khai và duy trì ngăn xếp phần mềm thời gian chạy. Thông thường, FaaS không yêu cầu viết mã ứng dụng CSC để sử dụng bất kỳ khung ứng tạo công cụ thể nào. Ngoài ra, FaaS đảm nhận trách nhiệm chạy ứng dụng và ngăn xếp phần mềm thời gian chạy theo yêu cầu để phục vụ bất kỳ sự kiện nào kích hoạt ứng dụng.

Thực tế, FaaS nhằm mục đích làm cho hạ tầng trở nên vô hình đối với nhà phát triển ứng dụng và dịch

vụ. Khi sử dụng FaaS, các máy chủ, máy ảo và/hoặc vùng chứa cơ bản sẽ ẩn đối với người dùng dịch vụ. Nhà phát triển không những không truy cập được mà còn không thể truy cập được vì chúng được CSP tự động quản lý như một phần của dịch vụ mây.

Một khía cạnh quan trọng của FaaS là các tài nguyên chỉ được sử dụng trong khi một chức năng cụ thể đang được thực thi. Thông thường, khi một ứng dụng sử dụng VM hoặc vùng chứa trong dịch vụ tính toán mây, mỗi trường hợp VM hoặc vùng chứa đang chạy sẽ liên tục tiêu thụ tài nguyên, cho dù ứng dụng đó có đang thực thi các yêu cầu đến hay không. Đối với FaaS, khi không có sự kiện nào được xử lý, sẽ không có tài nguyên nào được sử dụng. Do đó, điều này có nghĩa là chi phí CSC sẽ ít hơn, đặc biệt là đối với các chức năng ít được sử dụng thường xuyên hơn. FaaS kích hoạt các tài nguyên cần thiết (chẳng hạn như vùng chứa bên dưới) khi xảy ra kích hoạt sự kiện cho một chức năng nhất định.

Quản lý tự động do FaaS cung cấp là chìa khóa – khả năng mở rộng được cung cấp tự động. Nếu tỷ lệ sự kiện cho một FaaS nhất định tăng lên, thì các tài nguyên được phân bổ cho dịch vụ mây đó sẽ tự động tăng lên để xử lý các sự kiện đó và được phân bổ lại sau khi tỷ lệ sự kiện giảm xuống.

Có sẵn một số dịch vụ mây thời gian chạy không server, bao gồm Apache OpenWhisk [6], AWS Lamba [5], Azure Functions [60], Google App Engine [3], Google Cloud Functions [61], IBM Cloud Functions [62], Oracle Fn [63].

8.2.2 Các chức năng trong FaaS

Sử dụng FaaS có nghĩa là viết một hoặc nhiều chức năng, trong đó mỗi chức năng là một đoạn mã dành riêng cho một nhiệm vụ cụ thể. Chính khía cạnh này của việc lập trình thời gian chạy không server đã tạo nên tên gọi cho các dịch vụ mây này: Chức năng như một dịch vụ (FaaS). Trên thực tế, đây là một thay đổi lớn trong cách phát triển các ứng dụng và dịch vụ, thể hiện một số nguyên tắc của kiến trúc vi dịch vụ (xem Điều 9). Có một số nguyên tắc áp dụng cho các hàm và cách chúng thực thi.

Các hàm là không trạng thái, có nghĩa là mỗi hàm không giữ bất kỳ trạng thái nào giữa các lần gọi hàm liên tiếp. Thực tế, điều này có nghĩa là các chức năng không tự lưu trữ bất kỳ dữ liệu nào. Nếu một chức năng cần lưu trữ và truy cập dữ liệu có thời gian tồn tại lâu hơn một lần gọi chức năng, thì chức năng đó sẽ tích hợp với một hoặc nhiều dịch vụ lưu trữ mây (xem Điều 12).

Mỗi chức năng được thực thi khi được kích hoạt bởi một số sự kiện, trong đó thông báo sự kiện là kết quả của một số thay đổi trạng thái, tức là FaaS có kiến trúc hướng sự kiện liên quan và điều này liên quan đến hành vi không đồng bộ liên quan đến việc gửi và nhận sự kiện. Cách tiếp cận này có thể cho phép khả năng mở rộng và khả năng phục hồi lớn hơn nhiều cho các ứng dụng, đặc biệt là khi các ứng dụng được triển khai trên các hệ thống bàn giao.

Các chức năng bị giới hạn thời gian ở chỗ chúng không thể thực thi trong hơn một khoảng thời gian đã chỉ định, do CSP xác định. Giới hạn thời gian thay đổi từ ưu đãi FaaS này sang ưu đãi FaaS khác nhưng thường là một số phút nhỏ. Do đó, bất kỳ tác vụ tồn tại lâu dài nào đều không phù hợp để triển khai như một chức năng.

Liên quan đến giới hạn thời gian của các chức năng là câu hỏi về độ trễ khởi động chức năng, tức là

lượng thời gian cần thiết để FaaS cung cấp phiên bản đang chạy của chức năng khi một sự kiện xảy ra. Điều này có thể liên quan đến khởi động nguội, trong đó phiên bản mới phải được bắt đầu lại từ đầu hoặc khởi động ấm khi FaaS có thể sử dụng lại phiên bản được sử dụng để xử lý sự kiện trước đó. Khởi động nguội liên quan đến độ trễ lớn hơn nhiều so với khởi động ấm. Khởi động nguội có nhiều khả năng xảy ra hơn đối với các chức năng không được sử dụng thường xuyên do FaaS thường giải phóng một phiên bản không được sử dụng trong hơn một khoảng thời gian nhất định (thường là ngắn).

Mỗi chức năng được cung cấp thông qua API và có thể được gọi bởi ứng dụng khách hoàn toàn bên ngoài hệ thống mây (ví dụ: ứng dụng người dùng cuối chạy trên thiết bị khách) hoặc bởi ứng dụng khách là một phần khác của ứng dụng tổng thể. Mỗi chức năng có thể được coi là một vi dịch vụ và đến lượt nó, mỗi chức năng có thể phụ thuộc vào việc sử dụng các vi dịch vụ khác để đạt được khả năng.

Cách các sự kiện được mô tả trong cấu trúc dữ liệu trở thành mối quan tâm đáng kể đối với các chức năng trong thời gian chạy serverless. Do đó, các thông số kỹ thuật đã xuất hiện để giúp mô tả các sự kiện một cách rõ ràng và nhất quán, chẳng hạn như thông số kỹ thuật CloudEvents của Cloud Native Computing Foundation [49].

Vì có thể triển khai một chức năng duy nhất tại một thời điểm, nên có sự linh hoạt đáng kể trong cách tiếp cận không server. Các ứng dụng có thể được xây dựng một chức năng tại một thời điểm, mỗi lần được triển khai và mở rộng độc lập. Điều này cũng ngụ ý tăng tốc độ phát triển và triển khai (không cần đợi xây dựng ứng dụng hoặc dịch vụ chứa nhiều khả năng) mỗi khả năng có thể được tạo, thử nghiệm và triển khai riêng.

8.2.3 Các khung không server

Khung không server là một công cụ hỗ trợ tạo và triển khai các chức năng cho các dịch vụ mây FaaS, cụ thể là hỗ trợ triển khai các chức năng cho các dịch vụ FaaS khác nhau của các CSP khác nhau.

Thông thường, các dịch vụ FaaS là độc quyền của CSP, mặc dù có một số triển khai FaaS nguồn mở.

Để giải quyết vấn đề phát triển các chức năng để triển khai trên bất kỳ dịch vụ CSP FaaS nào, các khung không server đã được phát triển cho phép phát triển các chức năng có thể nhắm mục tiêu đến các dịch vụ FaaS khác nhau theo yêu cầu, quan tâm đến sự khác biệt giữa các dịch vụ FaaS khác nhau. các dịch vụ, đặc biệt đối với quy trình tải lên và triển khai.

Ví dụ về thời gian chạy không server mã nguồn mở là Apache OpenWhisk [6] và Oracle Fn [63]. Một ví dụ về khung không server Serverless Framework mã nguồn mở[50].

8.2.4 Mối quan hệ của FaaS đối với vi dịch vụ và vùng chứa

Việc sử dụng FaaS về cơ bản liên quan đến việc sử dụng kiến trúc vi dịch vụ mây cho các ứng dụng. FaaS ngụ ý sử dụng cách tiếp cận "mây đầu tiên" cho các ứng dụng, rất khác về phong cách so với các ứng dụng "một khối" bao gồm tất cả các chức năng trong một gói duy nhất.

Do đó, sử dụng FaaS và các chức năng là một cách để triển khai kiến trúc ứng dụng dựa trên vi dịch

vụ, nhưng không yêu cầu sử dụng vùng chứa và các CMS liên quan như Kubernetes.

Tuy nhiên, có thể kết hợp việc sử dụng FaaS với các dịch vụ vi mộ được triển khai bằng vùng chứa (hoặc sử dụng các VM), với các hàm gọi vi dịch vụ dựa trên vùng chứa và các vi dịch vụ dựa trên vùng chứa gọi các chức năng FaaS như mong muốn.

8.3 Cơ sở dữ liệu không server

Cơ sở dữ liệu không server là một dạng tính toán không server trong đó khả năng được CSC sử dụng là cơ sở dữ liệu, trong đó cơ sở dữ liệu được cung cấp, quản lý và vận hành bởi CSP và các chức năng được cung cấp thông qua API.

Đối với tính toán không server, việc phân bổ tài nguyên lưu trữ được quản lý bởi CSP. Dung lượng lưu trữ được tự động và linh hoạt thay đổi tỷ lệ để phù hợp với lượng dữ liệu CSC được đặt vào cơ sở dữ liệu. Việc sao chép và sao lưu được quản lý bởi CSP và điều này bao gồm việc đặt dữ liệu ở các vị trí phù hợp với mục đích sử dụng dữ liệu và đồng thời giữ cho nhiều bản sao đồng bộ với nhau. Tương tự, các tài nguyên xử lý cần thiết để phục vụ các truy vấn và cập nhật cơ sở dữ liệu cũng được CSP quản lý và mở rộng quy mô.

Ví dụ về cơ sở dữ liệu không server bao gồm Amazon Aurora Serverless [64], FaunaDB [65], Google Cloud Firestore [66], IBM Cloudant [67], Microsoft Azure Data Lake [68], Oracle Autonomous Database [83], Oracle NoSQL Database [84].

9 Kiến trúc vi dịch vụ

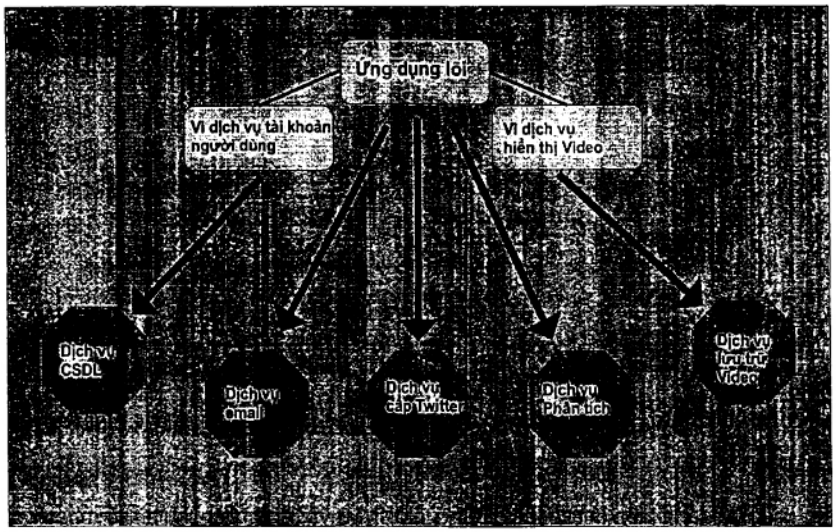
9.1 Quy định chung

Kiến trúc vi dịch vụ là một phương pháp thiết kế để xây dựng một ứng dụng mây bẩm sinh. Ứng dụng mây bẩm sinh là một ứng dụng được thiết kế rõ ràng để chạy bên trong và tận dụng các khả năng cũng như môi trường của các dịch vụ mây. Kiến trúc vi dịch vụ là một kiểu kiến trúc liên quan đến việc chia nhỏ ứng dụng thành các vi dịch vụ có thể triển khai độc lập có thể được triển khai nhanh cho bất kỳ tài nguyên hạ tầng nào theo yêu cầu [10][11]. Trong kiến trúc vi dịch vụ, ứng dụng được chia thành một loạt các quy trình riêng biệt được gọi là *các vi dịch vụ*, được triển khai độc lập và kết nối với nhau thông qua các giao diện dịch vụ. Khái niệm này là các vi dịch vụ trong ứng dụng được thiết kế để triển khai một số lĩnh vực chức năng cụ thể, có thể là một quy trình kinh doanh cụ thể hoặc một khả năng kỹ thuật cụ thể. Kiến trúc cho phép vận hành và cập nhật từng vi dịch vụ một cách độc lập. Do đó, kiến trúc vi dịch vụ là kỹ thuật bao trùm trong đó vi dịch vụ là thành phần chính.

CHÚ THÍCH Các thuật ngữ dịch vụ và giao diện dịch vụ được sử dụng ở đây với các định nghĩa được đưa ra bởi ISO/IEC 18384-1 [69], định nghĩa này cũng cung cấp giải thích tốt về kiến trúc theo hướng dịch vụ trong đó kiến trúc vi dịch vụ là một ví dụ cụ thể. Một dịch vụ hoặc một vi dịch vụ nên được phân biệt với một dịch vụ mây. Có thể xảy ra trường hợp vi dịch vụ được triển khai dưới dạng dịch vụ mây, nhưng đây là lựa chọn của nhà phát triển ứng dụng và không bắt buộc phải làm như vậy.

Một ví dụ đơn giản về ứng dụng dựa trên vi dịch vụ được đưa ra trong Hình 4. Trong ví dụ này, ứng dụng có một lõi cộng với hai vi dịch vụ, một cái xử lý khả năng kinh doanh của việc xử lý tài khoản người dùng với cái thứ hai là xử lý hiển thị và kiểm soát các chuỗi video. Ví dụ còn đi xa hơn và cho thấy rằng các ứng dụng vi dịch vụ cũng có thể sử dụng các dịch vụ khác, điển hình là các dịch vụ mây

cung cấp các khả năng mà ứng dụng cần. Do đó, trong ví dụ này, vi dịch vụ tài khoản người dùng sử dụng dịch vụ cơ sở dữ liệu để lưu trữ và truy xuất thông tin tài khoản; vi dịch vụ hiển thị video sử dụng dịch vụ lưu trữ video lưu trữ các chuỗi video; ứng dụng cốt lõi sử dụng dịch vụ email, dịch vụ cấp dữ liệu twitter và dịch vụ phân tích.



Hình 4 - Ví dụ về ứng dụng được cấu trúc sử dụng kiến trúc vi dịch vụ

Điều quan trọng là phải hiểu rằng kiến trúc vi dịch vụ là một kỹ thuật và các vi dịch vụ là phần chính của kiến trúc vi dịch vụ. Điều này tách biệt với các công nghệ có thể được sử dụng để triển khai vi dịch vụ. Các vi dịch vụ có thể được triển khai bằng cách sử dụng các vùng chứa hoặc các VM hoặc sử dụng tính toán không server và được kết nối bằng một vài mạng ảo hóa, nhưng điều này tách biệt với kỹ thuật được sử dụng để xây dựng ứng dụng.

Phân tách chức năng thường trong ngữ cảnh thiết kế theo hướng miền là chìa khóa để xây dựng một kiến trúc vi dịch vụ thành công. Một quan điểm liên quan đến kiến trúc này là đó là sự sàng lọc và đơn giản hóa của kiến trúc hướng dịch vụ (SOA). Một số đặc trưng của kiến trúc vi dịch vụ như sau [12]:

— Mỗi thành phần kiến trúc được gọi là "dịch vụ" có giao diện được xác định rõ ràng và được xuất bản rõ ràng.

— Mỗi dịch vụ hoàn toàn tự chủ.

— Việc thay đổi triển khai dịch vụ không ảnh tượng hưởng đến các dịch vụ khác vì giao tiếp giữa các dịch vụ chỉ diễn ra bằng cách sử dụng các giao diện (thường là giao diện REST).

— Liên kết lỏng lẻo và gắn kết cao giữa các dịch vụ cho phép kết hợp nhiều dịch vụ để xác định các dịch vụ hoặc ứng dụng cấp cao hơn.

Các ứng dụng dựa trên vi dịch vụ tương phản với "ứng dụng một khối", trong đó tất cả các thành phần của ứng dụng được xây dựng và kết hợp với nhau trong một quy trình duy nhất, điển hình hơn cho các

ứng dụng doanh nghiệp cũ hơn, không phải mây.

9.2 Ưu điểm và thách thức của vi dịch vụ

Lợi ích [13][14] của kiến trúc vi dịch vụ là:

— Cơ sở mã đơn giản hơn cho các dịch vụ riêng lẻ.

— Khả năng cập nhật và mở rộng quy mô từng dịch vụ một cách riêng biệt.

— Cho phép các dịch vụ được viết bằng các ngôn ngữ khác nhau để đáp ứng nhu cầu về hiệu suất và dễ dàng phát triển. Điều này được gọi là "lập trình đa ngôn ngữ".

— Sử dụng các ngăn xếp phần mềm trung gian đa dạng và thậm chí các tầng dữ liệu đa dạng cho các dịch vụ khác nhau (tính linh hoạt).

Một trong những lợi thế của việc sử dụng kiến trúc vi dịch vụ là mỗi thành phần của ứng dụng được xây dựng dưới dạng vi dịch vụ có thể được điều chỉnh tỷ lệ riêng biệt để phù hợp với tải của riêng thành phần đó. Điều này khác với cách tiếp cận ứng dụng một khối. Trong kiến trúc ứng dụng một khối, tất cả các thành phần được triển khai và vận hành như một thực thể duy nhất, chỉ có thể mở rộng quy mô bằng cách tăng/giảm quy mô toàn bộ ứng dụng. Điều này có thể dẫn đến sự kém hiệu quả về tài nguyên đối với những thành phần của ứng dụng không chịu tải nặng.

Các hệ thống PaaS giúp dễ dàng triển khai độc lập từng vi dịch vụ và liên kết chúng lại với nhau để tạo ứng dụng đầy đủ. Mỗi vi dịch vụ có thể được quản lý độc lập: mở rộng quy mô, bàn giao, cập nhật.

Một ưu điểm khác của việc sử dụng kiến trúc vi dịch vụ là mỗi thành phần của ứng dụng được xây dựng dưới dạng một vi dịch vụ có thể có vòng đời phát triển riêng biệt. Điều này cho phép các thành phần ứng dụng nhỏ hơn có thể được sửa đổi, mở rộng, thử nghiệm và triển khai nhanh hơn.

Những lợi ích gia tăng đi kèm với những thách thức cần giải quyết [15] để hiện thực hóa những lợi ích đó. Một cuộc thảo luận ngắn gọn về những thách thức này được đưa ra dưới đây:

— Tối ưu hóa giao tiếp: Chạy một ứng dụng trong các quy trình khác nhau dẫn đến tăng chi phí giao tiếp do các lệnh gọi API giữa các dịch vụ so với các lệnh gọi chức năng trong một quy trình. Chiến lược tổng thể liên quan đến việc xác định giao thức phù hợp, kỳ vọng về thời gian phản hồi, thời gian chờ và thiết kế API, với các tạo phẩm như Cổng API (9.7), Bộ ngắt mạch (9.6), Bộ cân bằng tải (14.2) và các Proxy (14.2).

— Khám phá dịch vụ: Điều này đề cập đến khả năng các dịch vụ khám phá lẫn nhau một cách nhất quán. Cần có một quy trình chuẩn và nhất quán để các dịch vụ tự đăng ký và công bố. Các dịch vụ tiêu thụ sẽ có thể khám phá các thiết bị đầu cuối và vị trí của các dịch vụ khác. Thông số kỹ thuật về cách các cổng API được định cấu hình để báo cáo tính khả dụng của dịch vụ và cho phép khám phá là cần thiết.

— Kết quả thực hiện: Việc cố gắng đáp ứng một yêu cầu chức năng kinh doanh duy nhất có thể dẫn đến việc phối hợp nhiều cuộc gọi dịch vụ lại với nhau. Điều này có thể gây ra độ trễ bổ sung trong thời gian phản hồi. Hơn nữa, dữ liệu thường được sử dụng bởi một vi dịch vụ có thể được sở hữu bởi một vi dịch vụ khác. Điều này yêu cầu khả năng đồng bộ hóa và chia sẻ dữ liệu để tránh chi phí liên lạc do sao chép dữ liệu trong khi yêu cầu dịch vụ.

— Khả năng chịu lỗi: Đây là khả năng phục hồi của hệ thống sau lỗi một phần. Các nhà phát triển vi dịch vụ cần cung cấp các cơ chế để phục hồi hoặc dừng bất kỳ lỗi nào lan truyền đến các phần khác của hệ thống. Hơn nữa, một số dịch vụ được chạy trong nhiều bản sao vì lý do khả năng mở rộng và tính khả dụng. Số lượng bản sao, tính nhất quán của phiên bản giữa các bản sao, cơ chế cân bằng tải và vị trí mạng là những yếu tố quyết định chính để đảm bảo khả năng chịu lỗi.

— Bảo mật: Một quyết định quan trọng là quyết định mối quan hệ tin cậy giữa các vi dịch vụ dựa trên các cách khác nhau mà các dịch vụ giao tiếp với nhau. Khi gọi một dịch vụ khác, một dịch vụ có thể sử dụng giao thức đồng bộ hoặc không đồng bộ. Tất cả các yếu tố này sẽ được xem xét khi chỉ định chuỗi ủy quyền trong mã thông báo truy cập. Các mẫu giao tiếp giữa các dịch vụ nên có các cơ chế xác thực và ủy quyền cụ thể và hiệu quả dựa trên các chính sách bảo mật dựa trên rủi ro. Tăng cường giao tiếp giữa các thành phần (như được mô tả trong Tối ưu hóa giao tiếp ở trên) yêu cầu các giao thức giao tiếp an toàn đáp ứng các yêu cầu cho ứng dụng.

— Theo dõi và ghi nhật ký: Quá trình phân tách các ứng dụng một khối thành các vi dịch vụ khác nhau tạo ra nhu cầu về các kỹ thuật và giải pháp bổ sung liên quan đến hệ thống gỡ lỗi và định hình. Một tính năng cần thiết được gọi là theo dõi phân tán yêu cầu khả năng theo dõi một chuỗi các cuộc gọi dịch vụ để xác định một giao dịch kinh doanh đơn lẻ hoặc một yêu cầu của người dùng. Một hệ thống ghi nhật ký trung tâm thường được yêu cầu để có được chế độ xem toàn diện về hành vi của hệ thống và điều này đòi hỏi khả năng tổng hợp để tích hợp thông tin nhật ký từ các vi dịch vụ riêng lẻ.

— Triển khai: Sự phổ biến của các quy trình dịch vụ yêu cầu cơ chế triển khai tự động. Khả năng mở rộng và tính toàn vẹn của hệ thống là mối quan tâm chính trong việc triển khai vi dịch vụ. Vùng chứa là cơ chế chiếm ưu thế được sử dụng để triển khai vi dịch vụ và việc sử dụng CMS (xem 7.4) (chỉ định tài nguyên và triển khai cấu trúc liên kết kết nối) giải quyết các mối lo ngại về triển khai. Tuy nhiên, một số giả định và yêu cầu trong các mô hình triển khai có thể không phù hợp với yêu cầu chức năng của một số ứng dụng dựa trên vi dịch vụ. Một ví dụ là giả định về tình trạng không trạng thái của vùng chứa làm chủ một vi dịch vụ, trong đó yêu cầu ứng dụng/hệ thống tổng thể yêu cầu một vi dịch vụ có trạng thái.

— Phân rã chức năng: Trong khi phân rã một ứng dụng một khối, có các vấn đề cần quyết định như:

a) ranh giới thích hợp của các dịch vụ khác nhau, và

b) khi một dịch vụ quá lớn và do đó cần phải chia nhỏ.

9.3 Đặc tả của vi dịch vụ

Thiết kế một kiến trúc vi dịch vụ yêu cầu sử dụng các sơ đồ mô tả và ngôn ngữ mô tả nền tảng trung lập do tính không đồng nhất liên quan đến thiết kế của các thành phần vi dịch vụ. Mặc dù UML chủ yếu được sử dụng cho sơ đồ mô tả, các ngôn ngữ sau đây thường được sử dụng:

— Ngôn ngữ mô hình hóa tiêu chuẩn, chẳng hạn như RAML và YAML.

— Các ngôn ngữ đặc tả tiêu chuẩn, chẳng hạn như Javascript (Node.js), JSON và Ruby.

— Mã giả cho các thuật toán.

— Ngôn ngữ đặc tả giao diện triển khai trung lập, chẳng hạn như đặc tả Open API [70].

9.4 Kiến trúc nhiều lớp

Thiết kế theo hướng miền [54] và kiến trúc nhiều lớp liên quan [52] là một mẫu phổ biến được sử dụng trong công nghệ phần mềm. Bằng cách chia các ứng dụng theo chức năng thành các lớp riêng biệt, kiến trúc nhiều lớp cung cấp các ưu điểm sau:

— Cộng tác hiệu quả

Mỗi lớp được phát triển bởi chuyên gia của mỗi lớp: GUI dựa trên trình duyệt Web được phát triển bởi các nhà thiết kế Web và logic miền bởi các lập trình viên Java™ chẳng hạn. Các chuyên gia có thể tập trung vào mối quan tâm của riêng họ với ít sự can thiệp.

— Bảo trì dễ dàng

Mã chương trình của mỗi lớp độc lập về mặt logic với mã chương trình của các lớp khác. Miễn là các lập trình viên không phá vỡ giao diện với các lớp khác, việc thay đổi mã chương trình linh hoạt.

— Khả năng tái sử dụng

Một ứng dụng được chia thành các thành phần nhỏ hơn trong kiến trúc nhiều lớp. Các thành phần phần mềm chi tiết có thể được sử dụng lại dễ dàng hơn so với các thành phần chi tiết thô.

Việc sử dụng kiến trúc nhiều lớp có hiệu quả trong các ứng dụng dựa trên vi dịch vụ. Nó đã được áp dụng cho các ứng dụng được phát triển bằng các vi dịch vụ và một số thực tiễn liên quan đến việc sử dụng kiến trúc nhiều lớp có sẵn trong tài liệu đã xuất bản (ví dụ: Xem Toby Clemson [27]). Mặc dù không có kiến trúc nhiều lớp được tiêu chuẩn hóa cho các vi dịch vụ, nhưng kiến trúc lớp được đề xuất trong Thiết kế theo hướng miền [52] đã được tham chiếu và bản chất có thể được cung cấp bằng bốn lớp thành phần như trong Hình 5:

— Giao diện người dùng

Thành phần phần mềm xác định chấp nhận yêu cầu từ người dùng và cung cấp phản hồi.

— Ứng dụng

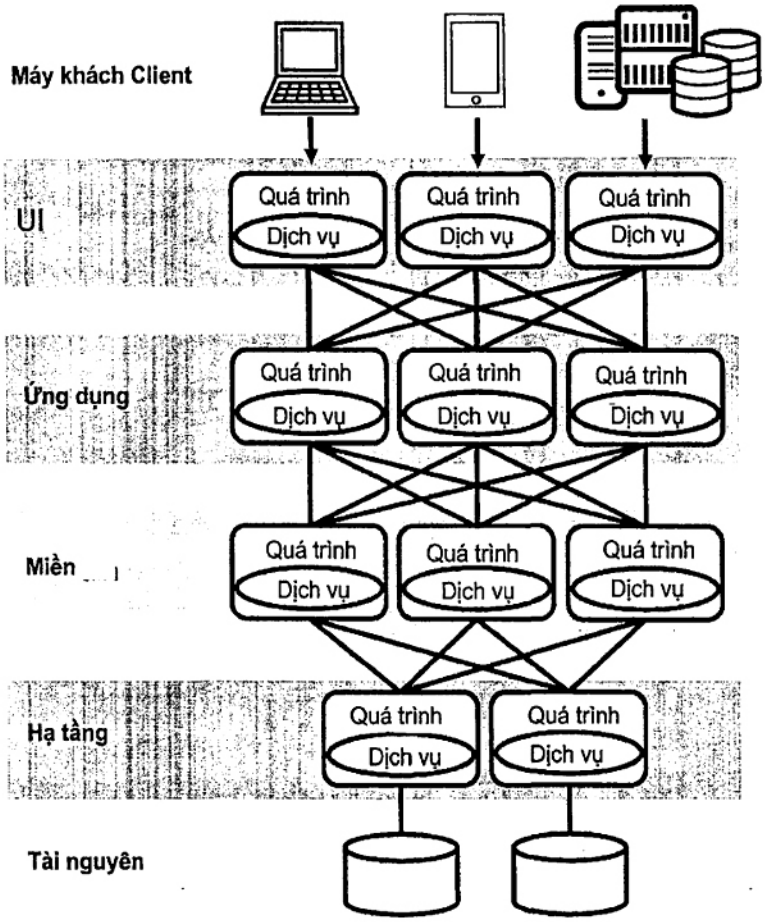
Thành phần phần mềm xác định ranh giới của ứng dụng. Nó là thiết bị đầu cuối để tương tác với các khách hàng và chịu trách nhiệm dàn xếp các yêu cầu và phản hồi, gọi logic miền và quản lý các bối cảnh giao dịch.

— Miền

Thành phần phần mềm thực hiện logic nghiệp vụ.

— Hạ tầng

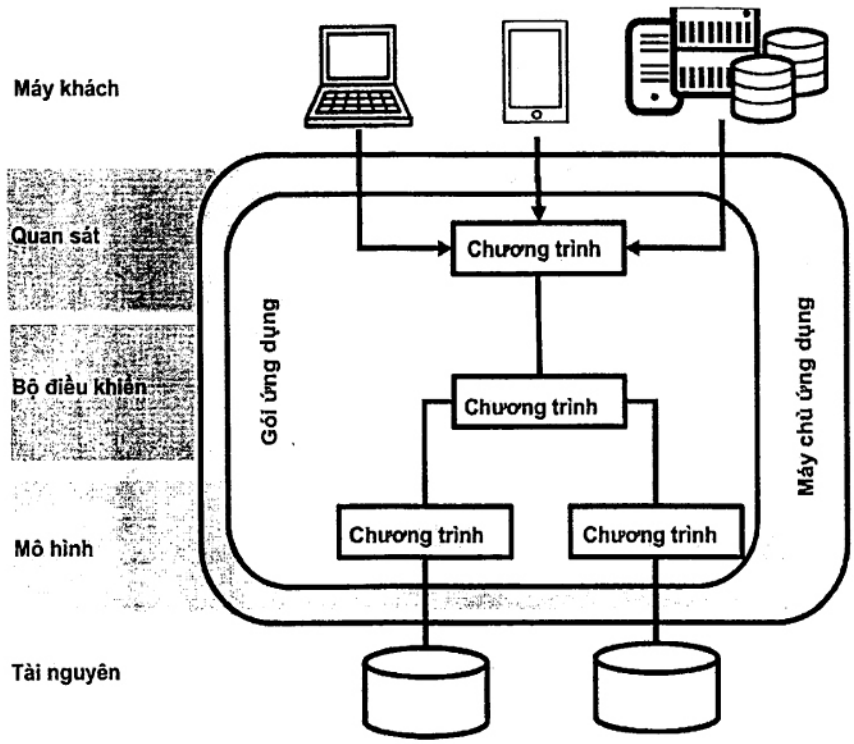
Thành phần phần mềm đóng gói các tài nguyên vật lý bao gồm dữ liệu và cung cấp cho lớp miền một giao diện trừu tượng để truy cập dữ liệu.



Hình 5 — Kiến trúc nhiều lớp được sử dụng với các vi dịch vụ

Mỗi thành phần được thể hiện trong Hình 5 là một vi dịch vụ riêng biệt, chạy trong quy trình riêng và gọi các vi dịch vụ khác khi cần thiết.

Các ứng dụng web một khối trước đây đã được phát triển dựa trên kiến trúc nhiều lớp được gọi là bộ điều khiển xem mô hình như trong Hình 6. Tuy nhiên, có một số khác biệt trong việc triển khai kiến trúc nhiều lớp giữa một ứng dụng được thiết kế bằng vi dịch vụ và một ứng dụng có thiết kế một khối, được kết hợp với việc đóng gói thành phần phần mềm và thời gian chạy ứng dụng.



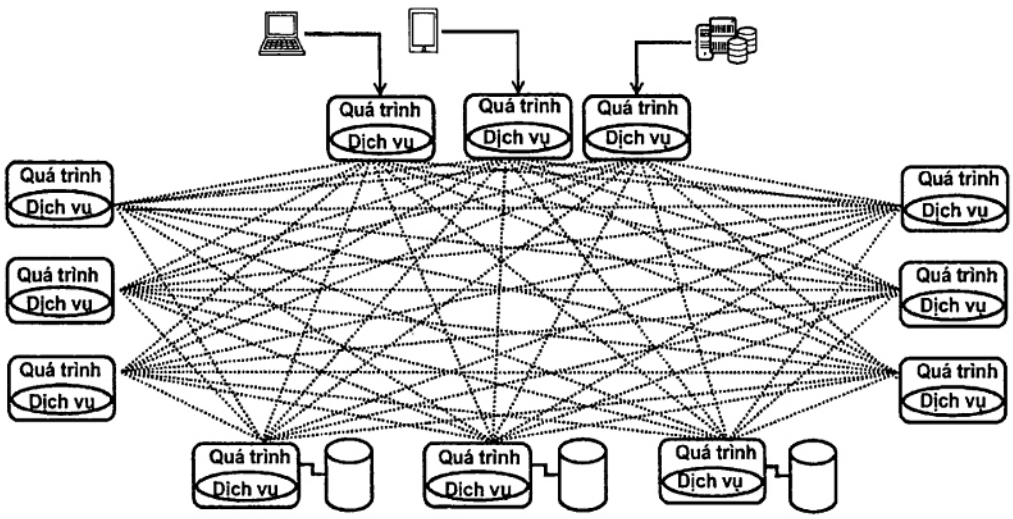
Hình 6 - Mẫu ứng dụng web một khối

Trong thiết kế ứng dụng một khối, mặc dù một ứng dụng được thiết kế và triển khai dựa trên kiến trúc nhiều lớp, tất cả các thành phần phần mềm được tập hợp trong một gói phần mềm và được triển khai trong một thời gian chạy ứng dụng. Ngay cả khi một nhà thiết kế web thêm một bản cập nhật nhỏ vào GUI, toàn bộ gói phần mềm phải được xây dựng, kiểm tra và thời gian chạy máy chủ phải dừng lại để triển khai gói phần mềm mới. Đây có thể là một quá trình dài, ngay cả đối với một thay đổi nhỏ đối với ứng dụng.

Mặt khác, trong một thiết kế ứng dụng các vi dịch vụ, mỗi thành phần phần mềm trong mỗi lớp được đóng gói dưới dạng một vi dịch vụ riêng biệt và được triển khai độc lập cho một quy trình riêng biệt. Mỗi quy trình có thể được triển khai bằng sử dụng máy ảo hoặc vùng chứa hoặc dưới dạng các chức năng không server, mỗi quy trình có thể được bắt đầu và dừng riêng biệt. Nếu mỗi vi dịch vụ được thiết kế tốt theo kiểu kết hợp lỏng lẻo, thì nhà phát triển có thể cập nhật một thành phần mà không cần phải xây dựng, thử nghiệm hoặc triển khai lại các vi dịch vụ khác. Kiến trúc vi dịch vụ cho phép thay đổi ứng dụng dễ dàng và linh hoạt.

9.5 Mạng lưới dịch vụ

Trong kiến trúc vi dịch vụ, số lượng các vi dịch vụ được liên kết với một ứng dụng có thể trở nên lớn. Trong trường hợp này, mỗi dịch vụ thường chạy với nhiều phiên bản có cấu hình cụm, mỗi phiên bản có một quy trình riêng biệt được triển khai dưới dạng máy ảo hoặc vùng chứa. Số lượng quy trình có thể gấp nhiều lần số lượng dịch vụ. Điều này làm cho cấu trúc liên kết tổng thể trở thành một mạng phức tạp được gọi là mạng lưới dịch vụ như trong Hình 7.



Hình 7 - Mạng lưới dịch vụ cho ứng dụng dựa trên các vi dịch vụ

Để chạy một ứng dụng dựa trên các vi dịch vụ và gặt hái những lợi ích, cần phải giải quyết những thách thức mà mạng lưới dịch vụ đưa ra:

a) Quản lý tắc nghẽn lưu thông

- Cân bằng tải chi tiết cho một phiên bản vi dịch vụ cụ thể.
- Triển khai Blue/Green, để cập nhật vi dịch vụ mà không dừng ứng dụng.
- Phát hành Canary.
- Bộ ngắt mạch. (Xem 9.7)

b) Khám phá dịch vụ

- Đăng ký dịch vụ.
- Tra cứu dịch vụ.

c) Kiểm tra/thử nghiệm

- Tiêm lỗi.

d) An ninh

- Xác thực.
- Ủy quyền.
- Mã hóa.

e) Đo từ xa

- Tích hợp Log và Trace.
- Tích hợp số liệu.

— Bản điều khiển

Có các cách tiếp cận khác để quản lý mạng lưới dịch vụ: a) API và b) kết cấu mạng lưới dịch vụ. Đối với cách tiếp cận API, các nhà phát triển ứng dụng sử dụng một API cụ thể trong các chương trình của họ để quản lý mạng lưới dịch vụ. Tuy nhiên, để làm được điều này, các nhà phát triển phải nỗ lực triển khai các yêu cầu phi chức năng cũng như các yêu cầu chức năng và kết quả là mã ứng dụng bao gồm các chi tiết triển khai phi chức năng, điều này trái với nguyên tắc "tách các mối quan tâm" của công nghệ phần mềm. và làm cho mã phức tạp hơn và khó sửa đổi hơn. MicroProfile là một ví dụ về API mạng lưới dịch vụ[28].

Cấu trúc mạng lưới dịch vụ là một giải pháp hạ tầng ứng dụng, nằm bên dưới lớp ứng dụng và trên lớp điều phối, đồng thời làm trung gian cho tất cả lưu lượng truy cập giữa các vi dịch vụ. Nó quản lý mạng lưới dịch vụ bằng cách điều khiển lưu lượng đến và đi qua chính nó. Sau đó, chương trình ứng dụng được giải phóng khỏi việc triển khai các khả năng cần thiết để quản lý mạng lưới dịch vụ. Istio [29] và Linkerd [30] là những ví dụ về triển khai vải mạng lưới dịch vụ.

9.6 Bộ ngắt mạch

Bộ ngắt mạch là một mẫu thiết kế và cũng là một thành phần phần mềm dựa trên mẫu đó [55].

Bộ ngắt mạch áp dụng khi một thành phần phần mềm gọi một thành phần phần mềm khác (chẳng hạn như vi dịch vụ) thông qua lệnh gọi API. Các thành phần phần mềm liên quan đang chạy trong các quy trình khác nhau và lệnh gọi API thường diễn ra qua mạng. Các cuộc gọi API từ xa như vậy có thể không thành công hoặc bị treo mà không có phản hồi. Khi thành phần đích là một dịch vụ thường được sử dụng, điều này có thể dẫn đến một loạt lỗi trên ứng dụng hoặc hệ thống.

Ý tưởng của bộ ngắt mạch là bất kỳ lệnh gọi API từ xa nào như vậy đều được bao bọc bởi một thành phần bộ ngắt mạch, đây thực sự là một phần của thành phần phần mềm máy khách. Khi máy khách gọi lệnh gọi API, nó sẽ được xử lý bởi thành phần bộ ngắt mạch giám sát các lỗi. Khi trạng thái lỗi được nhận dạng, các lệnh gọi tới API sẽ được bộ ngắt mạch phản hồi lỗi nhanh chóng. Bộ ngắt mạch cũng có thể tạo cảnh báo cho mục đích giám sát trong những trường hợp này. Bộ ngắt mạch có thể tiếp tục theo dõi API và thành phần đích để biết tính khả dụng và tự động thiết lập lại sau khi sự cố được khắc phục.

Việc nhận dạng trạng thái lỗi có thể thay đổi từ bộ ngắt mạch này sang bộ ngắt mạch khác và bộ ngắt mạch có thể có các tham số có thể cài đặt để kiểm soát hành vi (ví dụ: ngưỡng lỗi, ngưỡng hết thời gian chờ).

Bộ ngắt mạch không loại bỏ nhu cầu đối với thành phần máy khách để xử lý lỗi của lệnh gọi API, nhưng nó giúp phát triển các cơ chế xử lý thích hợp dễ dàng hơn.

9.7 Cổng API

Cổng API là một tố phần phần mềm có thể được dùng để cung cấp một API tích hợp duy nhất cho một nhóm vi dịch vụ đang được một thành phần máy khách cụ thể sử dụng cùng nhau. (Xem Microsoft, 2019 [56])

Mỗi vi dịch vụ trình bày API của riêng mình, dựa trên khả năng. Một khách hàng cụ thể có thể đang sử

dụng toàn bộ chuỗi các vi dịch vụ để đạt được mục tiêu của mình. Việc xử lý tất cả các lệnh gọi API khác nhau cần được thực hiện đối với các vi dịch vụ khác nhau có liên quan có thể trở nên phức tạp đối với phần mềm máy khách. Một cổng API có thể trình bày một API nhất quán đơn giản hơn cho phần mềm máy khách và gọi các API vi dịch vụ nếu cần để triển khai API đơn giản hơn. Do đó, cổng API là một thành phần tập trung vào máy khách và có thể cần nhiều cổng API khác nhau để đáp ứng yêu cầu của các máy khách khác nhau.

10 Tự động hóa

10.1 Quy định chung

Tự động hóa là một tính năng chính của cả việc cung cấp và sử dụng các dịch vụ mây. Tự động hóa được áp dụng cho toàn bộ vòng đời, thông qua thiết kế, phát triển, thử nghiệm, triển khai, sản xuất và ngừng hoạt động. Tự động hóa là điều cần thiết để đạt được năng suất và cũng để giảm các kỹ năng và nỗ lực cần thiết để cung cấp và sử dụng các dịch vụ mây.

Một trong những mục tiêu của tự động hóa là giảm nỗ lực và gánh nặng triển khai các ứng dụng và dữ liệu vào các dịch vụ mây, thừa nhận rằng điều này được thực hiện trên cơ sở tương đối thường xuyên, để khắc phục sự cố hoặc để cung cấp các cải tiến cho chức năng. Tự động hóa nhất thiết phải được kết nối với việc áp dụng một loạt các kỹ thuật công nghệ phần mềm, mặc dù không dành riêng cho tính toán mây nhưng đã trở thành các yếu tố quan trọng trong việc áp dụng thành công tính toán mây.

10.2 Tự động hóa vòng đời phát triển

Một trong những yếu tố quan trọng của tự động hóa là việc áp dụng triển khai liên tục hoặc bàn giao liên tục. Triển khai liên tục là một cách tiếp cận kỹ thuật phần mềm trong đó các nhóm sản xuất phần mềm theo các chu kỳ ngắn để phần mềm có thể được phát hành vào sản xuất bất kỳ lúc nào và việc triển khai vào sản xuất được tự động hóa. Bàn giao liên tục tương tự như triển khai liên tục, ngoại trừ giai đoạn triển khai được bắt đầu thủ công (nghĩa là quyết định triển khai được đưa ra bởi con người chứ không phải một hệ thống tự động nào đó – bản thân quá trình triển khai thường được tự động hóa). Nói chung, việc sử dụng triển khai liên tục hoặc bàn giao liên tục cũng liên quan đến việc tổ chức áp dụng DevOps. DevOps liên quan đến một phương pháp kết hợp hoạt động phát triển phần mềm và CNTT với nhau để rút ngắn vòng đời phát triển, cho phép bàn giao thường xuyên các bản sửa lỗi và chức năng nâng cao phù hợp chặt chẽ với các mục tiêu kinh doanh.

Việc triển khai liên tục và bàn giao liên tục đặt trọng tâm vào việc phát triển phần mềm theo từng bước nhỏ, đồng thời nhấn mạnh vào thử nghiệm tự động trong và sau các bước xây dựng và triển khai. Các bước gia tăng nhỏ liên quan chặt chẽ đến việc phát triển các ứng dụng sử dụng vi dịch vụ (mỗi vi dịch vụ cung cấp một phần chức năng tổng thể) và việc sử dụng các dịch vụ (mây) riêng biệt cho chức năng phổ biến hơn (ví dụ: khả năng cơ sở dữ liệu, khả năng nhắn tin).

Tích hợp liên tục là một phần vốn có của quá trình triển khai liên tục và bàn giao liên tục, trong đó các bản cập nhật của nhà phát triển đối với cơ sở mã được thực hiện thường xuyên và cơ sở mã được xây dựng và kiểm tra thường xuyên (thường nhiều lần trong ngày). Tích hợp liên tục được xây dựng trên cơ sở phát triển dựa trên thử nghiệm, với mục đích tự động chạy thử nghiệm đơn vị và thử nghiệm tích

hợp để kiểm tra xem các bản cập nhật cho cơ sở mã có phá vỡ mã theo bất kỳ cách nào không và cung cấp phản hồi nhanh chóng cho nhà phát triển trong các trường hợp nơi có vấn đề.

Quản lý tự động là một yếu tố chính của hoạt động cho các dịch vụ mây. Các tác vụ như khôi phục các phiên bản phần mềm bị lỗi, tăng và giảm quy mô tài nguyên, đặc biệt là các phiên bản song song của các thành phần ứng dụng, sao chép dữ liệu và sao lưu dữ liệu. Tất cả những điều này cần phải được tự động hóa khi sử dụng các dịch vụ mây, nếu không các tác vụ này có thể khiến nhân viên vận hành choáng ngợp.

Một phần mở rộng quan trọng của phương pháp DevOps được gọi là DevSecOps. Đối với DevSecOps, khả năng bảo mật được coi là một phần thiết yếu và không thể thiếu trong quá trình phát triển và vận hành. Ý tưởng là tự động hóa các nhiệm vụ bảo mật song song với việc tự động hóa các nhiệm vụ phát triển và vận hành vốn là trọng tâm của DevOps. Sự gia tăng về tốc độ của các nhiệm vụ phát triển và vận hành do phương pháp DevOps mang lại phù hợp với DevSecOps bằng sự gia tăng về tốc độ của các nhiệm vụ liên quan đến bảo mật, trong suốt vòng đời của một ứng dụng.

10.3 Tạo công cụ tự động hóa

Các công cụ là một phần thiết yếu của tất cả các giai đoạn của quá trình phát triển.

Thông thường, công cụ bắt đầu với hệ thống quản lý kiểm soát nguồn (SCM), hệ thống này lưu giữ mã nguồn và cung cấp các quy trình được kiểm soát để thực hiện cập nhật mã, bao gồm theo dõi tất cả các thay đổi. Hệ thống SCM tạo cơ sở cho các công cụ xây dựng, thử nghiệm, bàn giao và triển khai hoạt động. Có nhiều hệ thống SCM khác nhau đang được sử dụng, tuy nhiên, Git SCM mã nguồn mở [32] được sử dụng rất rộng rãi, với rất nhiều công cụ liên quan, bao gồm cả khả năng của máy chủ lưu trữ.

Máy chủ tự động hóa là một công cụ được sử dụng để tự động hóa các bước tích hợp liên tục, bàn giao liên tục và triển khai liên tục. Nó đặc biệt hữu ích để thực hiện các bản dựng mã từ SCM và thực hiện kiểm tra (kiểm tra đơn vị, kiểm tra tích hợp) trên mã đã xây dựng. Có một số công cụ máy chủ tự động hóa có sẵn và đang được sử dụng, mặc dù công cụ mã nguồn mở Jenkins [32] thường được sử dụng.

Tự động hóa bảo mật hỗ trợ DevSecOps có thể bao gồm các công cụ kiểm tra mã để tìm lỗ hổng tại thời điểm mã được kiểm tra trong SCM và kiểm tra lỗ hổng thông qua thử nghiệm trong quá trình xây dựng và trong giai đoạn tích hợp liên tục. Điều này cũng sẽ liên quan đến việc sử dụng an toàn các cơ sở mã đáp ứng các phụ thuộc của ứng dụng, ví dụ: thư viện phần mềm trung gian, ảnh tượng vùng chứa và dịch vụ sao lưu. Các phần phụ thuộc như vậy phải được gắn với các chính sách bảo mật để xác định phần phụ thuộc nào phù hợp để sử dụng, được hỗ trợ bởi thử nghiệm thích hợp và hệ thống quản lý phản hồi thông báo về các lỗ hổng và nhu cầu thay đổi sang phiên bản sửa lỗi sau này.

Phần mềm quản lý cấu hình được sử dụng để tự động hóa việc cung cấp phần mềm, quản lý cấu hình và triển khai ứng dụng. Kiến trúc được sử dụng cho các ứng dụng gốc trên mây làm tăng nhu cầu về phần mềm quản lý cấu hình, vì thường có nhiều thành phần được cài đặt trong nhiều dịch vụ mây và vị trí liên quan, tất cả đều phải được phối hợp để cho phép ứng dụng hoạt động chính xác. Một loạt các công cụ phần mềm quản lý cấu hình đang được sử dụng. Một số công cụ nguồn mở thường được sử dụng bao gồm Ansible [34], CFEngine [35], Chef [36] và Puppet [37].

Các công cụ phần mềm quản lý cấu hình khác nhau về kiến trúc của chúng. Ansible sử dụng kiến trúc không có tác nhân, trong khi các công cụ khác dựa trên tác nhân (tức là chúng yêu cầu deamon phần mềm được cài đặt trên các nút đích hoặc trên một máy chủ được liên kết).

Yếu tố chính của việc triển khai các ứng dụng trong môi trường mây là điều phối, vì các ứng dụng thường bao gồm một số lượng đáng kể các thành phần riêng biệt phải được triển khai, định cấu hình và vận hành cùng nhau. Tự động hóa điều phối là phạm vi của các công cụ, chẳng hạn như CMS như được mô tả trong 7.4.

Một yếu tố quan trọng hỗ trợ tự động hóa là cung cấp các khả năng của dịch vụ mây thông qua API (giao diện lập trình ứng dụng). API cho phép các công cụ khác nhau định cấu hình, triển khai, kiểm soát và giám sát từng dịch vụ mây. Các ứng dụng được triển khai và chạy trong sản xuất trong các dịch vụ mây phải được theo dõi và quản lý về hiệu suất và tính khả dụng. Giám sát và quản lý thường được thực hiện thông qua các API do CSP cung cấp. Các công cụ để quản lý việc khởi động lại các phiên bản bị lỗi, các công cụ để tăng và giảm quy mô số lượng phiên bản của một thành phần phần mềm cụ thể để đáp ứng với những thay đổi về khối lượng công việc, tất cả đều phụ thuộc vào khả năng giám sát và quản lý đó. Có trường hợp bản thân một số khả năng này được cung cấp dưới dạng dịch vụ mây (ví dụ: "tự động mở rộng quy mô"), nhưng trong các trường hợp khác, chúng được cung cấp dưới dạng công cụ riêng biệt phải được cài đặt và định cấu hình.

11 Kiến trúc của các hệ thống PaaS

11.1 Quy định chung

Nền tảng như một Dịch vụ (PaaS) là một thể loại dịch vụ mây liên quan đến việc cung cấp các khả năng của nền tảng, được định nghĩa trong TCVN 12480 (ISO/IEC 17788) là các khả năng mà *khách hàng dịch vụ mây có thể triển khai, quản lý và chạy do khách hàng tạo hoặc khách hàng có được các ứng dụng sử dụng một hoặc nhiều ngôn ngữ lập trình và một hoặc nhiều môi trường thực thi được nhà cung cấp dịch vụ mây hỗ trợ*. Một hệ thống PaaS thường bao gồm một tập hợp các dịch vụ mây PaaS nhất quán nhằm hoạt động cùng nhau.

Các hệ thống PaaS chủ yếu liên quan đến việc phát triển, triển khai và vận hành các ứng dụng của khách hàng. Các khả năng khác thường liên quan, chẳng hạn như sử dụng ứng dụng, xử lý, lưu trữ và tài nguyên mạng, nhưng chúng không phải là trọng tâm chính. Các hệ thống PaaS thường bao gồm các khả năng đa dạng của hạ tầng phần mềm ứng dụng (phần mềm trung gian) bao gồm nền tảng ứng dụng, nền tảng tích hợp, nền tảng phân tích kinh doanh, dịch vụ truyền phát sự kiện và dịch vụ back-end di động, cùng với các bộ công cụ hỗ trợ quá trình phát triển (Xem Gartner, 2014 và Gartner , 2018). Ngoài ra, dịch vụ PaaS thường bao gồm một tập hợp các khả năng vận hành như giám sát, quản lý, triển khai và các khả năng liên quan.

Các hệ thống PaaS được nhắm mục tiêu đến các nhà phát triển ứng dụng và cả nhân viên vận hành, đồng thời hỗ trợ khái niệm kết hợp về DevOps.

Một cách mô tả các hệ thống PaaS là chúng đại diện cho dịch vụ mây kết xuất hạ tầng ứng dụng được cung cấp bởi các thực thể như máy chủ ứng dụng, hệ thống quản lý cơ sở dữ liệu, môi giới tích hợp,

hệ thống quản lý quy trình kinh doanh, công cụ quy tắc và hệ thống xử lý sự kiện phức tạp. Hạ tầng ứng dụng như vậy hỗ trợ nhà phát triển ứng dụng viết các ứng dụng kinh doanh, giảm lượng mã cần viết đồng thời mở rộng khả năng chức năng của ứng dụng. Bản chất của hệ thống PaaS là nhà cung cấp dịch vụ mây chịu trách nhiệm cài đặt, cấu hình và vận hành môi trường thực thi ứng dụng (bao gồm mọi máy ảo cơ bản, hệ điều hành, vùng chứa, thời gian chạy, thư viện), chỉ để lại chính mã ứng dụng cho khách hàng dịch vụ mây và nhà phát triển của họ để cung cấp. Do đó, điểm khác biệt cơ bản giữa IaaS và PaaS là đối với IaaS, khách hàng phải xây dựng ảnh tượng VM hoặc ảnh tượng vùng chứa để thực thi mã ứng dụng của họ, trong khi PaaS cung cấp mọi thứ cần thiết để tải lên và thực thi mã ứng dụng trực tiếp.

Các dịch vụ PaaS cũng thường mở rộng khả năng nền tảng của phần mềm trung gian bằng cách cung cấp cho các nhà phát triển ứng dụng một bộ dịch vụ và các API đa dạng và đang phát triển, cung cấp chức năng cụ thể theo kiểu có sẵn liên tục, được quản lý. Cách tiếp cận này nhằm mục đích che giấu thực tế là có phần mềm trung gian hiện tại, cho phép các nhà phát triển tạo ra năng suất ngay lập tức. Một số hệ thống PaaS cũng kết hợp các tính năng của dịch vụ mây IaaS và SaaS, một mặt cung cấp một số quyền kiểm soát phân bổ tài nguyên cơ bản và mặt khác cung cấp các khả năng phần mềm hoàn chỉnh có sẵn.

Ngoài ra, các hệ thống PaaS thường cung cấp các khả năng của chúng theo cách cho phép các ứng dụng được phát triển trên chúng tận dụng các đặc trưng riêng của dịch vụ mây mà nhà phát triển ứng dụng thường không cần phải thêm mã đặc biệt vào chính ứng dụng đó. Điều này cung cấp một cách tiếp cận để xây dựng các ứng dụng gốc trên mây mà không yêu cầu các kỹ năng chuyên môn.

Ngoài ra, các hệ thống PaaS thường cung cấp các khả năng của chúng theo cách cho phép các ứng dụng được phát triển trên chúng tận dụng các đặc trưng riêng của dịch vụ mây mà nhà phát triển ứng dụng thường không cần phải thêm mã đặc biệt vào chính ứng dụng đó. Điều này cung cấp một cách tiếp cận để xây dựng *các ứng dụng mây bẩm sinh* mà không yêu cầu các kỹ năng chuyên môn.

11.2 Các đặc trưng của hệ thống PaaS

Các hệ thống PaaS thường thể hiện một tập các đặc trưng chính:

1. *Hỗ trợ các ứng dụng tùy chỉnh*:

   Hỗ trợ phát triển, triển khai và vận hành các ứng dụng tùy chỉnh. Các hệ thống PaaS thường hỗ trợ các ứng dụng mây bẩm sinh có khả năng tận dụng tối đa khả năng mở rộng, linh hoạt và phân tán của hạ tầng mây. Điều này thường đạt được mà không cần nhà phát triển ứng dụng viết mã đặc biệt để tận dụng các khả năng này.

2. *Cung cấp các môi trường thời gian chạy*:

   Các hệ thống PaaS thường cung cấp môi trường thời gian chạy cho các ứng dụng, trong đó mỗi môi trường thời gian chạy hỗ trợ một hoặc một nhóm nhỏ ngôn ngữ lập trình và khung, ví dụ: các tời gian chạy Node.js, Ruby và PHP. Một đặc trưng của nhiều dịch vụ PaaS là hỗ trợ nhiều môi trường thời gian chạy. Điều này cho phép các nhà phát triển chọn công nghệ thích hợp nhất cho nhiệm vụ đang thực hiện, đôi khi được gọi là môi trường *đa ngôn ngữ*.

Các môi trường thời gian chạy có thể bao gồm việc sử dụng các vùng chứa (xem Điều 7) và tính toán không server (xem Điều 8).

3. *Các cơ chế triển khai nhanh*:

Nhiều dịch vụ PaaS cung cấp cho các nhà phát triển và người vận hành cơ chế "đẩy và chạy" tự động để triển khai và chạy ứng dụng, cung cấp khả năng phân bổ tài nguyên động khi mã ứng dụng được chuyển đến dịch vụ mây PaaS thông qua API. Các yêu cầu cấu hình được giữ ở mức tối thiểu theo mặc định, mặc dù có khả năng kiểm soát cấu hình nếu được yêu cầu, ví dụ: kiểm soát số lượng phiên bản chạy song song của một ứng dụng để xử lý khối lượng công việc dự kiến hoặc để đáp ứng các mục tiêu về khả năng phục hồi.

4. *Hỗ trợ một dải các khả năng của phần mềm trung gian*:

Các ứng dụng có nhiều yêu cầu khác nhau và điều này được phản ánh trong việc cung cấp nhiều loại hạ tầng ứng dụng ("phần mềm trung gian") hỗ trợ một dải các khả năng. Một ví dụ là quản lý cơ sở dữ liệu, với cả công nghệ cơ sở dữ liệu SQL và NoSQL được cung cấp. Các khả năng khác bao gồm dịch vụ tích hợp, quản lý quy trình kinh doanh, dịch vụ phân tích kinh doanh, công cụ quy tắc, dịch vụ xử lý sự kiện và dịch vụ phụ trợ di động.

5. *Cung cấp các dịch vụ:*

Các hệ thống PaaS thường cung cấp một số khả năng dưới dạng một loạt các dịch vụ riêng biệt, thường được gọi thông qua một loại API nào đó. Các dịch vụ được cài đặt và chạy bởi nhà cung cấp dịch vụ, loại bỏ trách nhiệm và nỗ lực khỏi khách hàng dịch vụ mây. Ví dụ, trong trường hợp dịch vụ cơ sở dữ liệu, trách nhiệm đảm bảo tính khả dụng và độ tin cậy, có bản sao và sao lưu dữ liệu cơ sở dữ liệu, bảo mật dữ liệu, v.v... đều thuộc về nhà cung cấp dịch vụ. Các dịch vụ của nhà cung cấp là một khái niệm thiết yếu trong việc giảm thiểu nỗ lực và độ phức tạp trong việc xây dựng các hệ thống phần mềm, thay vì phải cài đặt và quản lý một số bộ phần mềm phức tạp tiềm tàng, khả năng này có sẵn từ nhà cung cấp.

6. *Các khả năng cấu hình sẵn*:

Nhiều hệ thống PaaS được đặc trưng bởi các khả năng được cấu hình sẵn của nhà cung cấp, với cấu hình tối thiểu dành cho nhà phát triển và nhân viên vận hành khách hàng. Điều này làm giảm độ phức tạp, tăng năng suất và giảm khả năng xảy ra sự cố không mong muốn, với khả năng quản lý đơn giản hơn và dễ gỡ lỗi hơn. Một số dịch vụ có thể tự động điều chỉnh cấu hình như vậy dựa trên các mẫu và tải sử dụng; điều này càng làm giảm chuyên môn và thời gian cần thiết để chạy các ứng dụng theo cách hiệu quả nhất.

7. *Các khả năng quản lý API*:

Các ứng dụng kinh doanh thường cần hiển thị một số khả năng thông qua API. Điều này có thể được yêu cầu bởi bản chất của giao diện người dùng đối với ứng dụng. Các ứng dụng dành cho thiết bị di động thường cần một API để khi hoạt động độc lập với ứng dụng kinh doanh, chúng có thể truy cập dữ liệu và giao dịch khi được yêu cầu. Trong các trường hợp khác, một phần của giải pháp doanh nghiệp là cho phép các bên khác (đối tác, khách hàng, nhà cung cấp) tích hợp các

ứng dụng của riêng họ với các ứng dụng của doanh nghiệp. Việc tích hợp như vậy được thực hiện thông qua API. Việc cung cấp API yêu cầu một mức độ kiểm soát để chỉ những người dùng được ủy quyền mới có thể truy cập API và mỗi người dùng chỉ có thể truy cập những khả năng mà họ có quyền. Điều này yêu cầu một số khả năng quản lý API và khả năng quản lý API được cung cấp bởi nhiều hệ thống PaaS.

8. *Các khả năng bảo mật*:

Bảo mật là một trong những khía cạnh quan trọng nhất của bất kỳ giải pháp nào. Các hệ thống PaaS thường cung cấp khả năng bảo mật tích hợp, do đó giảm tải cho các nhà phát triển và nhà điều hành. Các khả năng bao gồm tường lửa, quản lý thiết bị đầu cuối, xử lý giao thức an toàn, quyền truy cập và ủy quyền, mã hóa dữ liệu chuyển động và ở trạng thái nghỉ, kiểm tra tính toàn vẹn, cùng với các cơ chế phục hồi như bản sao dữ liệu dự phòng và sao lưu tự động. Các hệ thống PaaS có thể cung cấp các khả năng này với tác động tối thiểu hoặc không ảnh tượng hưởng đến mã ứng dụng, giúp đơn giản hóa các tác vụ của nhà phát triển. Ngoài ra, vì môi trường thực thi cơ bản là một phần của nền tảng, CSP đảm nhận trách nhiệm đối với các bản vá bảo mật hệ điều hành, phát hiện và loại bỏ phần mềm độc hại cũng như các tác vụ bảo trì bảo mật thiết yếu khác, do đó, khách hàng có thể tập trung vào việc tránh các lỗ hổng bảo mật được đưa vào mã của chính họ .

9. *Các công cụ dành cho nhà phát triển*:

Nhiều hệ thống PaaS nhằm mục đích hợp nhất và hợp lý hóa quá trình phát triển và vận hành, tức là hỗ trợ DevOps bằng cách phá vỡ sự phân chia giữa phát triển và vận hành. Các công cụ phát triển được cung cấp bao gồm trình chỉnh sửa mã, kho lưu trữ mã, công cụ xây dựng, công cụ triển khai, công cụ kiểm tra và dịch vụ cũng như công cụ bảo mật. Người ta thường tìm thấy các dịch vụ giám sát và phân tích ứng dụng, bao gồm các khả năng như ghi nhật ký, phân tích nhật ký, phân tích sử dụng ứng dụng và bảng điều khiển.

10. *Các khả năng vận hành*:

Các hệ thống PaaS hỗ trợ người vận hành thông qua các khả năng vận hành cho cả ứng dụng đã triển khai và cho chính hệ thống PaaS, thông qua bảng điều khiển và cả thông qua API cho phép khách hàng cắm bộ công cụ vận hành của riêng họ. Ví dụ: khả năng tăng hoặc giảm số lượng phiên bản đang chạy của một ứng dụng là phổ biến (để xử lý tải ứng dụng khác nhau), trong một số trường hợp được xử lý bởi các dịch vụ tự động thay đổi số lượng phiên bản dựa trên một bộ quy tắc do CSC thiết lập nhân viên vận hành.

11. *Hỗ trợ chuyển các ứng dụng hiện có*:

Nhiều khách hàng sử dụng dịch vụ mây có các ứng dụng hiện có có thể được chuyển sang môi trường PaaS mang lại lợi ích kinh doanh. Một số hệ thống PaaS có môi trường ứng dụng nhằm mục đích quan hệ chặt chẽ với những môi trường có sẵn trên ngăn xếp phần mềm trung gian không phải mây hiện có và các công cụ liên quan hỗ trợ quá trình chuyển.

12. *Hỗ trợ các ứng dụng sử dụng kiến trúc vi dịch vụ:*

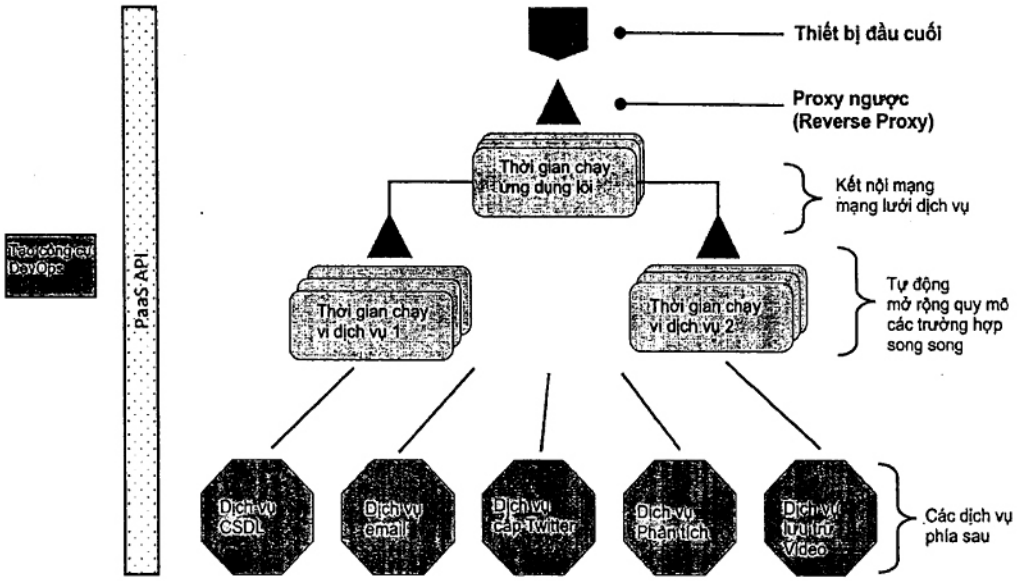
Các hệ thống PaaS thường cung cấp nhiều loại hỗ trợ cho các ứng dụng được xây dựng bằng kiến trúc vi dịch vụ (xem Điều 9). Điều này bao gồm hỗ trợ cho thời gian chạy được sử dụng cho chính các vi dịch vụ, hỗ trợ cho các dịch vụ cơ bản được sử dụng bởi các vi dịch vụ và hỗ trợ cho mạng lưới dịch vụ liên kết tất cả các thành phần lại với nhau.

13. *Các khả năng kết nối mạng*:

Do CSP kiểm soát và có khả năng hiển thị đầy đủ các ngăn xếp giao thức mạng đang được sử dụng nên các hệ thống PaaS có thể được tích hợp sâu hơn với khả năng mạng của CSP chủ. Điều này tương đối dễ dàng (đối với cả nhà phát triển CSP và CSC) để tích hợp PaaS với ảo hóa mạng, cân bằng tải mạng, chuyển đổi dự phòng, tối ưu hóa mạng, tối ưu hóa mạng, lưu vào bộ nhớ đệm, chuyển và xếp hàng tin nhắn cũng như các công nghệ liên quan đến mạng khác.

11.3 Kiến trúc các thành phần chạy trong hệ thống PaaS

Việc kết hợp một số thành phần của một hệ thống PaaS điển hình sẽ dẫn đến một kiến trúc sơ đồ gồm các thành phần của một ứng dụng điển hình được tạo và triển khai bằng hệ thống PaaS, như chỉ ra trong Hình 8.



Hình 8 - Kiến trúc sơ đồ của các thành phần chạy trong hệ thống PaaS

— Thiết bị đầu cuối ngoài: cung cấp thiết bị đầu cuối khả dụng bên ngoài (ví dụ: hiển thị trên internet), với bảo mật thiết bị đầu cuối liên quan (ví dụ: hỗ trợ https, quản lý chứng chỉ, xử lý tấn công DDoS, quản lý ID và quyền truy cập).

— Proxy ngược (Reverse Proxy): đối với mỗi thành phần của ứng dụng được mở rộng quy mô thông qua việc sử dụng các trường hợp song song (ứng dụng lõi và vi dịch vụ), cần có một Proxy ngược (Reverse Proxy) và chức năng cân bằng tải để bàn giao đồng đều các yêu cầu đến trên tất cả các trường hợp đang chạy.

— Mạng lưới dịch vụ: dành cho các kết nối nội bộ giữa các thành phần ứng dụng và dành cho các kết nối với dịch vụ, khả năng cho phép kết nối hiệu quả và hiệu quả.

— Tự động mở rộng quy mô của các trường hợp song song: cách tiếp cận điển hình để mở rộng quy mô của các thành phần ứng dụng là chạy song song nhiều phiên bản của từng thành phần và bàn giao các yêu cầu đến trên các phiên bản này (xem Điều 14). Số lượng trường hợp đang chạy cùng một lúc có thể được tăng và giảm để phù hợp với khối lượng công việc mà các yêu cầu yêu cầu. Thông thường, điều này yêu cầu hệ thống PaaS giám sát các trường hợp để xác định mức độ bận rộn của chúng. Khả năng này đôi khi có thể được liên kết với cân bằng tải mạng tự động PaaS, để các mức lưu lượng truy cập vào các phiên bản cụ thể có thể được khớp động với dung lượng và tính khả dụng hiện tại của chúng.

— Dịch vụ đằng sau (Backing service): thường xảy ra trường hợp nhiều khả năng theo yêu cầu của các thành phần ứng dụng được cung cấp bởi một tập hợp các dịch vụ mây mà các thành phần này được kết nối khi cần thiết. Các dịch vụ như vậy có thể rất đa dạng, nhưng các ví dụ bao gồm các khả năng như cơ sở dữ liệu hoặc các dịch vụ lưu trữ khác (xem Điều 12).

— API PaaS: các khả năng của hệ thống PaaS và các dịch vụ mây riêng lẻ tạo nên hệ thống được cung cấp cho các tạo công cụ DevOps khác nhau để sử dụng bằng một hoặc nhiều API PaaS. Ví dụ: một API như vậy có thể cho phép đẩy mã của một thành phần ứng dụng sang một dịch vụ thời gian chạy để thực thi.

— Tạo công cụ DevOps: các nhà phát triển và nhân viên vận hành, lý tưởng nhất là hợp nhất thành một nhóm DevOps liền mạch, sử dụng nhiều công cụ DevOps khác nhau để thực hiện công việc của họ. Các công cụ phát triển và thử nghiệm được sử dụng trong quá trình tạo và thử nghiệm một ứng dụng cũng như các vi dịch vụ, trong khi các công cụ quản lý và giám sát được sử dụng để quan sát và kiểm soát các thành phần ứng dụng trong quá trình sản xuất.

12 Lưu trữ dữ liệu như một dịch vụ

12.1 Quy định chung

Tính toán mây dựa trên việc cung cấp các dịch vụ mây, tất cả đều dựa trên ba loại tài nguyên hạ tầng là tính toán, lưu trữ và kết nối mạng. Điều này xem xét các dịch vụ mây cung cấp tài nguyên lưu trữ.

Các công nghệ của máy ảo và của vùng chứa được mô tả trong Điều 6 và 7 về cơ bản là cung cấp các loại tài nguyên tính toán, tức là một phương tiện để thực thi phần mềm trong một số loại môi trường ảo hóa. Một số khả năng lưu trữ được liên kết với cả các máy ảo và các vùng chứa. Có các hệ thống tệp được liên kết với chúng chứa các tệp đại diện cho phần mềm cũng như cấu hình và siêu dữ liệu được liên kết trực tiếp và các hệ thống tệp này cũng thường được yêu cầu để hỗ trợ việc thực thi phần mềm.

Tuy nhiên, cần phải hiểu rằng các hệ thống tệp được liên kết với cả các máy ảo và các vùng chứa về cơ bản là không bền lâu ở chỗ chúng tồn tại khi máy ảo hoặc vùng chứa được tạo ra và chúng bị loại bỏ khi cùng một máy ảo hoặc vùng chứa đó bị dừng và phá hủy. Điều này có nghĩa là các hệ thống tệp như vậy không có khả năng được sử dụng để lưu trữ thông tin lâu dài. Chúng cũng không thể được sử dụng cho thông tin cần được truy cập bởi nhiều máy ảo hoặc vùng chứa khác nhau, vì các hệ thống tệp trong máy ảo hoặc trong vùng

chứa được thiết kế riêng biệt.

Lưu trữ thông tin dài hạn được cung cấp bởi lưu trữ dữ liệu dưới dạng dịch vụ (DSaaS). Các dịch vụ DSaaS cung cấp nhiều kiểu khả năng lưu trữ khác nhau (các loại được mô tả chi tiết hơn ở phần sau của điều này) và những khả năng đó có thể được cung cấp cho cả hệ thống CSC và cả các dịch vụ mây khác. Các dịch vụ DSaaS dựa trên tài nguyên lưu trữ của CSP, thường được truy cập qua mạng thông qua API, mặc dù trong một số trường hợp, có thể và nên đặt cùng một dịch vụ DSaaS với một dịch vụ tính toán để loại bỏ độ trễ của mạng.

12.2 Các tính năng phổ biến của DsaaS

DSaaS cho phép người dùng lưu trữ và truy xuất thông tin ở bất cứ đâu và bất cứ lúc nào miễn là có kết nối với dịch vụ DSaaS. Các dịch vụ DSaaS hỗ trợ khả năng mở rộng về khối lượng thông tin được lưu trữ và độ tin cậy về quyền truy cập thông tin từ bất kỳ loại ứng dụng nào độc lập với hệ thống hoặc thiết bị mà các ứng dụng đó chạy trên đó.

Các ứng dụng và hệ thống sử dụng DSaaS để truy cập bộ nhớ mây thông qua các giao thức liên quan. Các giao thức này có thể hỗ trợ các tài nguyên lưu trữ từ xa về mặt địa lý và hỗ trợ ảo hóa các vị trí lưu trữ được sử dụng, để khi được yêu cầu, lưu trữ dự phòng hoặc sao chép được cung cấp để cung cấp khả năng phục hồi chống lại lỗi điểm.

Các dịch vụ lưu trữ có các đặc trưng chung sau:

a) **Độ bền**: Dữ liệu được lưu trữ ở một hoặc nhiều vị trí, do CSP kiểm soát. Các dịch vụ DSaaS sẽ cung cấp khả năng lưu trữ dữ liệu mà không bị mất mát do thiên tai, lỗi của con người hoặc lỗi kỹ thuật. Điều này có thể đạt được thông qua các bản sao hoặc sao lưu dữ liệu, có thể được cung cấp như một phần của dịch vụ hoặc có thể được CSC triển khai bằng cách sử dụng các khả năng của dịch vụ DSaaS.

b) **Tính sẵn sàng**: Các dịch vụ DSaaS cung cấp khả năng lưu trữ và truy xuất dữ liệu theo yêu cầu để đáp ứng nhu cầu của các ứng dụng và hệ thống của CSC.

c) **Bảo mật**: Các dịch vụ DSaaS nên lưu trữ thông tin một cách an toàn. Đặc biệt, không được có quyền truy cập trái phép vào dữ liệu khách hàng của dịch vụ mây. Bạn nên mã hóa thông tin nếu thích hợp, mặc dù khả năng này có thể được để cho CSC thực hiện vì nó có thể gây ra các tác động về chi phí và hiệu suất.

d) **Chi phí giới hạn**: Với DSaaS, khách hàng thường chỉ trả tiền cho dung lượng lưu trữ thực sự được sử dụng. Đối với một số dịch vụ DSaaS, thông tin được sử dụng ít thường xuyên hơn có thể được lưu trữ ở khả năng chi phí thấp hơn, có thể cung cấp khả năng truy cập chậm hơn như một phương tiện để giảm chi phí.

e) **Khả năng quản lý**: CSP của DSaaS có các chính sách và quy trình quản lý vòng đời lưu trữ cho phép người dùng và nhà phát triển tập trung vào giải quyết các vấn đề của ứng dụng và không phải lo lắng về việc quản lý thông tin.

DSaaS có thể được phân loại theo loại lưu trữ và theo thể loại dịch vụ như được mô tả trong Bảng 1 và 2 tương ứng. Mỗi dịch vụ lưu trữ có một hoặc nhiều giao diện dịch vụ, chẳng hạn như trình điều khiển thiết bị

khối, giao diện hệ thống tệp hoặc API lưu trữ đối tượng, được sử dụng bởi phần mềm máy khách. Các API này dựa trên mạng vì trong hầu hết các trường hợp, khả năng lưu trữ được mong đợi tồn tại trên một hệ thống không phải hệ thống chạy phần mềm máy khách.

Bảng 1 - DSaaS theo kiểu loại lưu trữ

Các dịch vụ	Các tính năng
Dịch vụ lưu trữ tệp	Các dịch vụ lưu trữ tệp cung cấp dung lượng lưu trữ bằng mô hình hệ thống tệp thông thường, với các tệp được chứa trong các thư mục trong các tập. Lưu trữ thường được cung cấp cho phần mềm máy khách bằng giao thức NFS (NFS 4.2 - IETF 7862) và (các) tập có liên quan được gắn vào môi trường máy khách thông qua trình điều khiển máy khách NFS. Đặc biệt, các dịch vụ lưu trữ tệp có thể được gắn vào các máy ảo và các vùng chứa để cung cấp khả năng lưu trữ lâu dài cho các môi trường đó. Các dịch vụ lưu trữ tệp thường dựa trên mạng (tương tự như các thiết bị Lưu trữ Đính kèm Mạng (NAS)) và có thể được nhiều khách hàng chia sẻ đồng thời. Vì bộ lưu trữ được ảo hóa bởi dịch vụ lưu trữ tệp nên bộ lưu trữ có khả năng mở rộng cao và thường bền, với các bản sao dự phòng được sao chép của các tệp được duy trì, có khả năng ở các vị trí tách biệt về mặt vật lý. Dữ liệu được lưu trữ cũng có thể được mã hóa. (Xem Amazon EFS, Lưu trữ tệp IBM). Nhiều ứng dụng cần quyền truy cập vào tệp được chia sẻ và hệ thống tệp. Loại dịch vụ lưu trữ này chủ yếu được hỗ trợ trên Lưu trữ Đính kèm Mạng (Network Attached Storage (NAS)). Loại dịch vụ này lý tưởng cho các trường hợp sử dụng như kho lưu trữ nội dung quy mô lớn, môi trường phát triển, cửa hàng phương tiện hoặc thư mục chính của người dùng.
Dịch vụ lưu trữ đối tượng	Các dịch vụ lưu trữ đối tượng lưu trữ dữ liệu như là các đối tượng dữ liệu trong một mặt phẳng, không gian tên không phân cấp (bể lưu trữ, vùng chứa hoặc vùng chứa), trong đó mỗi đối tượng có một khóa hoặc danh tính duy nhất. Thực tế, dịch vụ lưu trữ đối tượng hoạt động trên cơ sở mô hình khóa/giá trị, với giá trị là đối tượng. Ngoài ra, mỗi đối tượng dữ liệu có thể có một lượng siêu dữ liệu tùy ý do người dùng chỉ định được liên kết với nó, có khả năng phong phú hơn nhiều so với mức có thể có với các hệ thống tệp tiêu chuẩn. Các dịch vụ lưu trữ đối tượng có khả năng mở rộng cao vì chúng không bị ràng buộc với phần cứng lưu trữ cụ thể và có thể mở rộng trên nhiều thiết bị lưu trữ. Các đối tượng riêng lẻ cũng có thể rất lớn. Các dịch vụ lưu trữ đối tượng có thể được sử dụng đồng thời bởi nhiều ứng dụng khách. Các dịch vụ lưu trữ đối tượng thường được cung cấp thông qua một REST API, có khả năng kết nối mạng một cách tự nhiên và có thể truy cập được từ các máy khách từ xa. REST API (ví dụ: Amazon S3 API) không giống như giao diện hệ thống tệp thông thường và do đó, các ứng dụng khách phải được thiết kế và viết riêng để sử dụng các dịch vụ lưu trữ đối tượng. Các dịch vụ lưu trữ đối tượng đặc biệt hữu ích đối với dữ liệu phi cấu trúc (ví dụ: các ảnh tượng) được cập nhật tương đối không thường xuyên (việc cập nhật được thực hiện bằng cách thay thế toàn bộ đối tượng bằng một phiên bản mới). Dịch vụ lưu trữ đối tượng cũng thường chậm hơn dịch vụ lưu trữ tệp.
Dịch vụ lưu trữ khối	Các dịch vụ lưu trữ khối cung cấp khả năng truy cập băng thông cao với độ trễ thấp vào các thiết bị lưu trữ ở cấp độ khối. Các dịch vụ này đang cung cấp tương đương với Lưu trữ được đính kèm trực tiếp (Direct Attached Storage -DAS) hoặc Mạng vùng lưu trữ (Storage Area Network - SAN) cho hệ thống máy khách. Hình thức truy cập cấp thấp này vào tài nguyên lưu trữ cho phép khách hàng kiểm soát nhiều hơn và có hiệu suất cao hơn so với các loại DSaaS khác. Do đó, khi sử dụng dịch vụ lưu trữ khối, nó giống như các thiết bị lưu trữ phần cứng bổ sung đang được gắn vào hệ thống máy khách. Các dịch vụ lưu trữ khối thường được thiết kế để hoạt động trong một trung tâm dữ liệu, vì độ trễ tăng đáng kể nếu các dịch vụ được truy cập qua các mạng từ xa. Thông thường, các giao thức SAN chuyên dụng như iSCSI (IETF 7143) được sử dụng để cung cấp các dịch vụ lưu trữ khối qua mạng cho các máy khách. Các thiết bị lưu trữ

Các dịch vụ	Các tính năng
	từ xa được trình bày cho phần mềm máy khách dưới dạng một ổ đĩa được gắn, giống như nó là một đĩa được gắn cục bộ vào hệ thống mà phần mềm máy khách đang chạy trên đó. Các dịch vụ lưu trữ khối có thể có các hệ thống tệp được xây dựng trên chúng bởi phần mềm máy khách hoặc cách khác, phần mềm máy khách có thể sử dụng trực tiếp giao diện khối. Trường hợp thứ hai này có thể xảy ra khi phần mềm máy khách là phần mềm cơ sở dữ liệu (ví dụ: cơ sở dữ liệu SQL), hoặc phần mềm xử lý luồng (ví dụ: Apache Kafka).

Có các loại dịch vụ mây vốn dựa trên khả năng lưu trữ, nhưng ở đó các khả năng được cung cấp phát triển hơn so với lưu trữ dữ liệu đơn giản và ở đó các giao diện dịch vụ được cung cấp cho máy khách dịch vụ thường phản ánh các yêu cầu cụ thể của phần mềm máy khách. Các loại dịch vụ mây này, được mô tả trong Bảng 2, mỗi loại sử dụng một hoặc nhiều loại lưu trữ được mô tả trong Bảng 1, nhưng khía cạnh lưu trữ không phải là khả năng chính được cung cấp cho khách hàng.

Bảng 2 - Dịch vụ lưu trữ theo thể loại dịch vụ

Các dịch vụ lưu trữ	Các tính năng
Dịch vụ cơ sở dữ liệu NoSQL	Các dịch vụ cơ sở dữ liệu NoSQL cung cấp khả năng lưu trữ và truy xuất các dạng dữ liệu phi cấu trúc khác nhau như tài liệu, ảnh tượng, phim và dữ liệu nhị phân lớn. Có rất nhiều công nghệ cơ bản trong thể loại này, có thể được phân loại theo nhiều cách khác nhau, chẳng hạn như: - Bộ đệm khóa-giá trị - Lưu trữ khóa-giá trị - Lưu trữ khóa-giá trị (Cuối cùng nhất quán) - Lưu trữ khóa-giá trị (Đã đặt hàng) - Máy chủ cấu trúc dữ liệu - Lưu trữ Tuple - Cơ sở dữ liệu đối tượng - Lưu trữ tài liệu - Lưu trữ cột rộng - Cơ sở dữ liệu đa mô hình gốc - Cơ sở dữ liệu đồ thị
Dịch vụ cơ sở dữ liệu SQL	Các dịch vụ cơ sở dữ liệu SQL (hoặc quan hệ) lưu trữ dữ liệu có cấu trúc ở định dạng bảng cho phép thực hiện các truy vấn phức tạp tiềm tàng để trích xuất dữ liệu cho phù hợp với yêu cầu của khách hàng và thực hiện các cập nhật động trên nội dung cơ sở dữ liệu. SQL là một ngôn ngữ lập trình tương tác tiêu chuẩn được thiết kế để truy vấn, cập nhật và quản lý dữ liệu cũng như tập hợp dữ liệu trong hệ thống quản lý cơ sở dữ liệu. SQL được chuẩn hóa trong ISO/IEC 9075-1:2016 [71]. Cơ sở dữ liệu SQL hiện đại hỗ trợ khám phá các cột trên một loạt các tập dữ liệu: không chỉ bảng/cách nhìn về quan hệ, mà còn cả XML, JSON, đối tượng không gian, đối tượng kiểu ảnh tượng (Đối tượng lớn nhị phân và Đối tượng lớn ký tự) và ngữ nghĩa các đối tượng. Nhiều công nghệ cơ sở dữ liệu SQL cơ bản khác tồn tại và được cung cấp dưới dạng dịch vụ mây.
Lưu trữ dịch vụ hàng đợi thông điệp	Các khả năng của hàng đợi thông điệp sẵn sàng như là một dịch vụ mây và thường được liên kết với dạng xử lý không đồng bộ phân tán và kiến trúc hệ thống đang ngày càng phổ biến. Nhiều hệ thống hàng đợi thông điệp có khả năng lưu giữ thông điệp trong bộ nhớ. Các thông điệp có thể được duy trì để đảm bảo rằng chúng không bị mất và có thể được truy xuất bất cứ khi nào cần thiết, nhưng cũng có thể được duy trì để hỗ trợ xử lý luồng yêu cầu phân tích quy mô lớn của nhiều sự kiện nhằm trích xuất thông tin chi tiết hữu ích. Tính bền vững của thông điệp có thể đặt ra yêu cầu đáng kể đối với các hệ thống lưu trữ cơ bản do khối lượng lớn và tốc độ gửi tin nhắn cao.
Dịch vụ công nghệ chuỗi khối và sổ cái phân tán	Các dịch vụ mây công nghệ sổ cái phân tán (DLT), chẳng hạn như các dịch vụ mây chuỗi khối (Blockchain), hỗ trợ cung cấp và sử dụng sổ cái phân tán, là một dạng cơ sở dữ liệu giao dịch. Các dịch vụ mây DLT thường cung cấp khả năng chạy nút DLT, bao gồm cả trường hợp của phần mềm nền tảng DLT và cung cấp khả năng lưu trữ cho bản sao của chính sổ cái phân tán. Để biết thêm chi tiết, xem ISO 23257 [26].

Các dịch vụ lưu trữ	Các tính năng
Dịch vụ phân tích	Khả năng xử lý phân tích dựa trên việc xử lý số lượng lớn dữ liệu. Các tập lớn như vậy và tốc độ xử lý dữ liệu cao như vậy cần có sự hỗ trợ đặc biệt từ các dịch vụ lưu trữ chứa dữ liệu. Nhiều CSP cung cấp dịch vụ lưu trữ mây chuyên dụng hỗ trợ phân tích.
Dịch vụ quản lý tệp	Các dịch vụ quản lý tệp cung cấp một kiểu khả năng ứng dụng của dịch vụ mây, thường cho phép tự động sao chép tệp giữa thiết bị người dùng và bộ nhớ mây, đồng thời cung cấp khả năng chia sẻ và cập nhật tệp cho nhiều người dùng.
Dịch vụ lưu trữ liên kết	Trong trường hợp dịch vụ lưu trữ được liên kết, tài nguyên lưu trữ có thể được kết hợp một cách minh bạch trên một tập hợp các vị trí lưu trữ mây khác nhau, bao gồm tại chỗ, mây riêng hoặc mây công cộng, cho dù cung cấp lưu trữ tệp, khối hay đối tượng. Lưu ý rằng các khả năng sao chép hoặc chuyển đổi dự phòng đơn giản thường không ngụ ý rằng một dịch vụ lưu trữ là một dịch vụ lưu trữ được liên kết.

CHÚ THÍCH Các dịch vụ lưu trữ này có thể được cung cấp như lưu trữ liên tục hoặc lưu trữ trong bộ nhớ.

12.3 Kiểu khả năng của DSaaS

Điều 6.4, TCVN 12480:2020 (ISO/IEC 17788:2014), xác định ba kiểu khả năng:

- Hạ tầng (như đã thấy trong IaaS);
- Nền tảng (như đã thấy trong PaaS);
- Ứng dụng (như đã thấy trong SaaS).

DSaaS có thể cung cấp một hoặc nhiều kiểu khả năng này như được mô tả trong Bảng 3:

Bảng 3 – Các kiểu khả năng của DsaaS

Kiểu khả năng	Các dịch vụ
Hạ tầng	Dịch vụ lưu trữ tệp Dịch vụ lưu trữ đối tượng Dịch vụ lưu trữ khối Dịch vụ lưu trữ liên kết
Nền tảng	Lưu trữ dữ liệu do khách hàng lập trình, nơi mã do khách hàng viết có thể được tải lên và sử dụng để thao tác lưu trữ dữ liệu Dịch vụ phân tích Dịch vụ cơ sở dữ liệu NoSQL Dịch vụ cơ sở dữ liệu SQL Dịch vụ hàng đợi thông điệp Dịch vụ chuỗi khối và DLT
Ứng dụng	Giao diện người dùng hướng tới con người để thao tác lưu trữ, chẳng hạn như kho lưu trữ tài liệu dựa trên web: Dịch vụ quản lý tệp

Các kiểu khả năng của mây áp dụng cho các dịch vụ mây được đề cập trong Bảng 1 và 2 khác nhau như thể hiện trong Bảng 3. Hầu hết các dịch vụ mây được đề cập trong Bảng 1 thường được cung cấp dưới dạng các dịch vụ mây thuộc kiểu khả năng hạ tầng. Đối với Bảng 2, các dịch vụ mây thường thuộc kiểu khả năng của nền tảng hoặc ứng dụng, tùy thuộc vào chi tiết của dịch vụ cung cấp, ngoại trừ trường hợp dịch vụ lưu trữ được liên kết, thường là các dịch vụ kiểu khả năng hạ tầng.

12.4 Các khả năng bổ sung đáng kể của DSaaS

Ngoài khả năng lưu trữ dữ liệu dự kiến, nhiều dịch vụ lưu trữ mây còn cung cấp các khả năng bổ sung đáng kể có thể quan trọng đối với khách hàng sử dụng dịch vụ mây.

Nhóm khả năng đầu tiên liên quan đến khả năng phục hồi và khả năng chống lại các lỗi điểm trong hạ tầng của nhà cung cấp dịch vụ mây. Nhiều dịch vụ lưu trữ mây lưu trữ dữ liệu trong nhiều bản sao dự phòng, do đó, lỗi của một thiết bị lưu trữ hoặc lỗi truy cập vào một thiết bị không dẫn đến việc dịch vụ không khả dụng hoặc tệ hơn là mất dữ liệu. Bản chất của sự sao chép có thể khác nhau. Trong một số trường hợp, các bản sao được cố ý đặt ở một vị trí vật lý riêng biệt (ví dụ: một trung tâm dữ liệu khác hoặc một vùng khả dụng khác trong một trung tâm dữ liệu), với mục đích giải quyết các lỗi nghiêm trọng của một trung tâm dữ liệu hoàn chỉnh. Trong các trường hợp khác, đặc biệt là dịch vụ mây liên quan đến quyền truy cập hiệu suất cao, các vị trí được sao chép có thể cố ý gần nhau. Một số dịch vụ lưu trữ mây cung cấp khả năng cho khách hàng dịch vụ mây để chọn chính sách liên quan đến vị trí của các bản sao.

Nhóm khả năng liên quan chặt chẽ thứ hai liên quan đến việc tạo và lưu trữ các bản sao lưu dữ liệu. Các bản sao lưu này có thể được tự động hóa hoặc thực hiện theo yêu cầu của khách hàng dịch vụ mây. Các bản sao lưu có thể ở một vị trí "trực tiếp" (tức là bộ nhớ trực tuyến và có sẵn, nhưng ở một vị trí khác) hoặc đến một vị trí ngoại tuyến. Trường hợp thứ hai có thể được sử dụng để lưu giữ lâu dài với chi phí thấp hơn.

Một khả năng khác được cung cấp bởi một số dịch vụ lưu trữ mây là mối quan hệ tài nguyên. Điều này liên quan đến vị trí vật lý tương đối của các phiên bản dịch vụ lưu trữ mây so với các dịch vụ mây khác, chủ yếu là các trường hợp dịch vụ tính toán như máy ảo và vùng chứa. Một trong những lý do chính để đặt các phiên bản dịch vụ tính toán gần với các phiên bản dịch vụ lưu trữ là hiệu suất, cả về mặt giảm độ trễ xuống mức tối thiểu cũng như về mặt tối đa hóa băng thông cho dữ liệu truyền giữa các trường hợp tính toán và lưu trữ. Một số loại dịch vụ như dịch vụ lưu trữ khối thường chỉ được cung cấp ở dạng này. Ví dụ: các dịch vụ lưu trữ khối chỉ có thể được cung cấp để sử dụng cho các trường hợp dịch vụ tính toán chạy trên các nút trong cùng một trung tâm dữ liệu nơi có liên kết tốc độ cao giữa nút lưu trữ và nút tính toán, chẳng hạn như Ethernet hoặc Kênh sợi quang.

13 Kết nối mạng trong tính toán mây

13.1 Các khía cạnh chính của kết nối mạng

Kết nối mạng là một yếu tố chính của tính toán mây. Chính định nghĩa về tính toán mây dựa trên các khả năng được truy cập qua mạng: *dạng thức cho phép mạng truy cập vào nhóm tài nguyên ảo hoặc vật lý có thể chia sẻ linh hoạt và có thể mở rộng với khả năng cung cấp và quản trị tự phục vụ theo yêu cầu* [TCVN 12480:2020 (ISO/IEC 17788:2014)].

Mạng và các khả năng liên quan đến mạng cũng là một số tài nguyên thường được cung cấp bởi các dịch vụ mây.

Vì vậy, có hai lĩnh vực quan tâm lớn liên quan đến kết nối kết nối mạng trong tính toán mây. Đầu tiên là mạng mà một dịch vụ mây nhất định được truy cập và mọi khả năng trong dịch vụ mây được truy cập, chẳng hạn như một ứng dụng chạy trong dịch vụ tính toán. Điều này được gọi là "kết nối mạng truy cập mây" hoặc "kết nối mạng truy cập mây công cộng" nơi các dịch vụ mây công cộng có liên quan. Thứ hai là mạng được sử dụng để kết nối các phiên bản dịch vụ mây với nhau. Mạng này thường theo thiết kế nhằm mục đích không được truy cập bên ngoài các phiên bản dịch vụ mây cụ thể có liên quan. Điều này được gọi là "mạng trong mây".

13.2 Kết nối mạng truy cập mây

Các dịch vụ mây có các giao diện có thể truy cập từ bên ngoài cho phép người dùng mây và các hệ thống hoạt động thay mặt cho người dùng mây sử dụng các khả năng của chúng. Một số dịch vụ mây cũng có các giao diện có thể truy cập từ bên ngoài dành cho các khả năng của CSC chạy bên trong dịch vụ mây: ví dụ bao gồm các giao diện như giao diện web hoặc API cho các ứng dụng CSC chạy trong một phiên bản dịch vụ tính toán (ví dụ: trong một máy ảo hoặc trong một vùng chứa) và cả các giao diện với dữ liệu khả năng lưu trữ của phiên bản dịch vụ lưu trữ (ví dụ: kho lưu trữ tệp).

Thường xảy ra trường hợp các giao diện có thể truy cập từ bên ngoài được thể hiện công khai trên internet, ít nhất là đối với các dịch vụ mây công cộng. Tuy nhiên, trong một số trường hợp, các giao diện có thể truy cập từ bên ngoài có thể được cố tình ẩn khỏi chế độ xem công khai trên internet, ngay cả khi người dùng truy cập diễn ra trên hạ tầng internet. Các dịch vụ mây riêng có thể được triển khai theo cách khiến chúng chỉ có thể truy cập được trong các mạng riêng của tổ chức (nghĩa là các dịch vụ mây đó không thể truy cập được từ internet), mặc dù vẫn có thể có yêu cầu đối với một số giao diện có thể truy cập từ bên ngoài, chẳng hạn như nơi một ứng dụng web được triển khai cho một dịch vụ mây riêng cần trình bày một giao diện có sẵn công khai mà người dùng có thể truy cập được.

Các ứng dụng chạy trong dịch vụ mây có thể yêu cầu giao diện hiển thị bên ngoài có sẵn trên địa chỉ mạng và số cổng có thể truy cập công khai. Thông thường, giao diện có thể truy cập công khai như vậy được cung cấp dưới dạng "giao diện ảo" trong đó giao diện bên ngoài được trình bày trên địa chỉ mạng và cổng mà chính dịch vụ mây biết đến, trong khi mã chạy trong dịch vụ mây chạy trên một số mạng nội bộ địa chỉ và cổng, mà địa chỉ bên ngoài và cổng được ánh xạ tới. Quá trình giám sát máy ảo thiết bị đầu cuối này là cần thiết để cho phép chia sẻ tài nguyên vốn là nền tảng cho các dịch vụ mây. Ảo hóa thiết bị đầu cuối cũng hỗ trợ các khả năng như cân bằng tải trên nhiều phiên bản của một ứng dụng và khả năng bảo mật bao gồm tường lửa và xử lý tấn công DDoS.

Giao diện hiển thị công khai cũng có thể yêu cầu cấu hình cụ thể. Đặc biệt, một tổ chức đang chạy ứng dụng trên dịch vụ mây có thể thấy rất mong muốn rằng địa chỉ được sử dụng cho giao diện là địa chỉ thuộc về tổ chức chứ không phải địa chỉ thuộc về CSP. Khả năng hỗ trợ điều này thường được gọi là "Mang theo địa chỉ IP của riêng bạn" (BYOIP) và là khả năng mà CSC có thể định cấu hình giao diện hiển thị công khai cho ứng dụng chạy trong dịch vụ mây có địa chỉ thuộc về CSC. Điều này có thể áp dụng cho cả dịch vụ mây công cộng và dịch vụ mây riêng.

13.3 Kết nối mạng nội bộ mây

Trong môi trường dịch vụ mây, thông thường mạng được sử dụng để kết nối các thành phần khác nhau tạo nên một hệ thống. Đối với các dịch vụ mây có khả năng tính toán, chẳng hạn như chạy phần mềm trong máy ảo hoặc vùng chứa, thường có nhiều trường hợp đang chạy cần giao tiếp với nhau (ví dụ: trong đó ứng dụng được chia thành các thành phần chạy riêng biệt, như với kiến trúc vi dịch vụ) hoặc với các thành phần khác của giải pháp (ví dụ: với bộ cân bằng tải trong đó tỷ lệ ngang được sử dụng với nhiều phiên bản song song của một thành phần nhất định). Các dịch vụ lưu trữ cũng thường được kết nối với các trường hợp tính toán và điều này thường được thực hiện qua mạng.

Cũng có thể có nhiều lớp kết nối mạng nội bộ mây riêng biệt. Lớp ứng dụng như được mô tả trong

đoạn trước và cũng là lớp quản lý hoặc kiểm soát, được sử dụng để giám sát và kiểm soát từng dịch vụ mây. Các lớp khác nhau này được cố tình cách ly với nhau để chúng không thể can thiệp lẫn nhau.

Thông thường, mạng trong mây được ảo hóa. Các dịch vụ mây khác nhau không trực tiếp sử dụng các khả năng kết nối mạng mà sử dụng các khả năng kết nối mạng ảo hóa vừa cho phép chia sẻ tài nguyên mạng cơ bản, vừa cung cấp sự cách ly giữa các nhóm phiên bản dịch vụ mây khác nhau vì lý do bảo mật và cũng để tránh nhiễu.

Cấu trúc và tổ chức của các mạng ảo hóa cũng có thể cố tình tránh phản ánh tổ chức của các mạng vật lý bên dưới. Thường xảy ra trường hợp các thành phần của giải pháp được bàn giao trên nhiều vùng khả dụng trong một trung tâm dữ liệu hoặc trên nhiều trung tâm dữ liệu. Việc tổ chức vật lý này hiển thị với các thành phần của giải pháp có thể rất không mong muốn và do đó, một mạng ảo hợp nhất duy nhất được thể hiện cho các thành phần đó, phủ lên trên các mạng vật lý.

Môi trường tính toán ảo hóa, cả các VM và vùng chứa, liên quan đến việc kiểm soát chặt chẽ và ảo hóa tài nguyên mạng, bao gồm cả thiết bị đầu cuối do mỗi VM/vùng chứa hiển thị và cả thiết bị đầu cuối mạng đích được sử dụng bởi phần mềm chạy trong môi trường. Khả năng chạy nhiều máy ảo trên một hệ thống hoặc nhiều vùng chứa trên một hệ điều hành, rõ ràng yêu cầu ánh xạ từng thiết bị đầu cuối mạng mà phần mềm hiển thị với các thiết bị đầu cuối thực tế trong hệ thống chứa, chỉ để cho phép chia sẻ hệ thống mà không dẫn đến đụng độ. Việc triển khai cả máy ảo và vùng chứa đều yêu cầu cấu hình để xử lý các sự cố này.

Cũng thường xảy ra trường hợp cả các máy ảo và vùng chứa đều được sử dụng trong môi trường phân tán. Nhiều trường hợp của cùng một phần mềm có thể chạy trên các hệ thống khác nhau và những hệ thống đó có thể chạy ở các vị trí thực tế khác nhau. Các ứng dụng cũng thường được chia thành nhiều thành phần độc lập (dịch vụ, vi dịch vụ) và mỗi thành phần này có thể chạy ở các vị trí riêng biệt. Một thực tế tốt là các vị trí thực tế của các thành phần được ẩn khỏi phần mềm vì các thành phần ứng dụng cần liên lạc với nhau một cách liền mạch ở bất cứ nơi nào chúng đang chạy. Ngoài ra, tốt nhất là các thành phần ứng dụng chỉ có thể giao tiếp với nhau. Giao tiếp bên ngoài nên được kiểm soát cẩn thận thông qua các thiết bị đầu cuối ngoài được chỉ định cụ thể.

Thực tiễn tốt nhất liên quan đến kết nối mạng cho các kiến trúc phần mềm này là kết nối mạng được xác định hiệu quả ở cấp ứng dụng. Đây là một mạng ảo chỉ được sử dụng bởi các thành phần ứng dụng và mở rộng trong suốt tất cả các vị trí mà các thành phần ứng dụng đang chạy. Hàm ý là mỗi ứng dụng có một mạng ảo được liên kết và với các môi trường tính toán ảo như vùng chứa, mỗi mạng ảo được cách ly với các mạng khác.

Mạng ảo có thể được xây dựng bằng nhiều kỹ thuật và công nghệ khác nhau, bao gồm mạng được xác định bằng phần mềm (SDN) và ảo hóa chức năng mạng (NFV), một số mạng được xây dựng cho các loại thành phần và dịch vụ mây cụ thể như:

— VxLAN: công nghệ mạng lớp phủ, được chỉ định trong IETF RFC 7348 [41];

— Mạng Kubernetes [17];

— Hệ thống mạng vùng chứa, chẳng hạn như Calico [43] và WeaNet [44].

13.4 Mạng riêng ảo (VPN) và tính toán mây

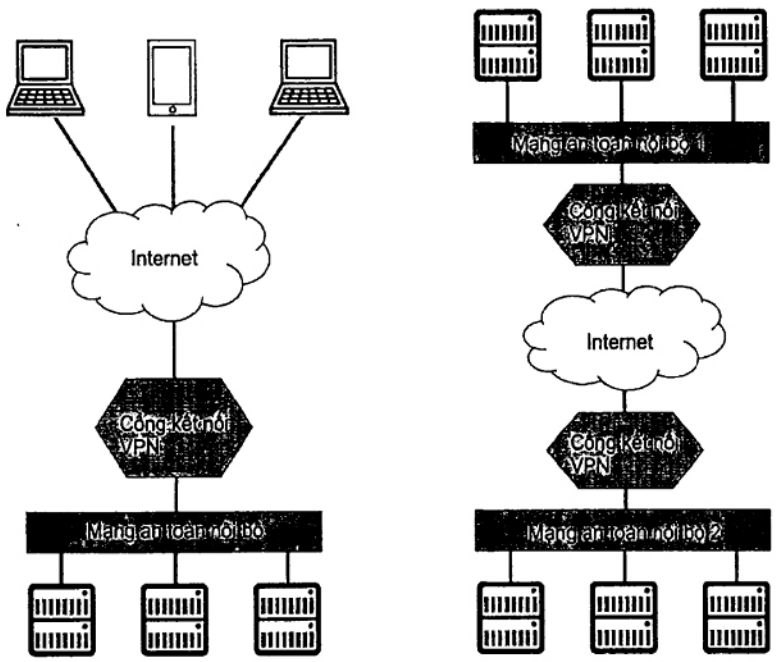
Một công nghệ có thể hữu ích trong việc xây dựng các giải pháp sử dụng tính toán mây là mạng riêng ảo (VPN). Các VPN cung cấp một phương tiện an toàn để kết nối các hệ thống với nhau trong đó một phần của hạ tầng mạng liên quan đến việc sử dụng các môi trường không đáng tin cậy như internet. Có hai cấu hình điển hình của VPN như trong Hình 9, phản ánh các trường hợp sử dụng khác nhau:

— Máy chủ đến cổng

Đây là nơi một hệ thống độc lập, điển hình là máy khách hoặc thiết bị, truy cập mạng được bảo mật từ xa. Trong trường hợp tính toán mây, ca sử dụng điển hình sẽ dành cho thiết bị khách truy cập các dịch vụ và ứng dụng mây cũng như các tài nguyên khác chạy bên trong chúng.

— Cổng-đến-Cổng

Đây là nơi truyền thông mạng an toàn được cung cấp giữa hai mạng an toàn riêng biệt. Ca sử dụng điển hình là khi các dịch vụ mây trong môi trường mây cần được kết nối với các ứng dụng và hệ thống nội bộ của CSC hoặc cần được kết nối với các dịch vụ mây khác đang chạy trong một môi trường mây khác.



Hình 9 — Các cấu hình VPN

Nhiều công nghệ khác nhau có sẵn để triển khai VPN, chẳng hạn như L2TP/IPsec và OpenVPN. Một hoặc nhiều loại VPN thường được hỗ trợ bởi các CSP công cộng.

14 Khả năng mở rộng tính toán mây

14.1 Các phương pháp tiếp cận khả năng mở rộng

Trong TCVN 12480 (ISO/IEC 17788), tính linh hoạt và khả năng mở rộng nhanh chóng là một trong những đặc trưng chính của tính toán mây. Tính linh hoạt và khả năng mở rộng nhanh chóng được mô tả là *một tính năng trong đó tài nguyên vật lý hoặc tài nguyên ảo có thể được điều chỉnh nhanh chóng và linh hoạt, trong một số trường hợp, tự động, để tăng hoặc giảm tài nguyên một cách nhanh chóng.*

Độ linh hoạt là mức độ mà các dịch vụ mây có thể thích ứng với những thay đổi về khối lượng công việc bằng cách cung cấp và hủy cung cấp tài nguyên, thường là tự động, để các tài nguyên có sẵn phù hợp với nhu cầu hiện tại nhất có thể. Khả năng mở rộng là khả năng của các dịch vụ mây tăng hoặc giảm các tài nguyên được phân bổ cho một khối lượng công việc nhất định.

Khả năng mở rộng được chia thành hai loại lớn: tỷ lệ ngang và tỷ lệ dọc:

- tỷ lệ ngang: còn được gọi là mở rộng quy mô ngoài (scale out) và mở rộng quy mô trong (scale in). Trong loại chia tỷ lệ này, nhiều tài nguyên được sử dụng song song và số lượng tài nguyên được sử dụng song song được thay đổi để cung cấp mức tài nguyên cần thiết. Đối với tài nguyên tính toán, nhiều máy được sử dụng (máy vật lý, máy ảo, vùng chứa). Đối với tài nguyên lưu trữ, nhiều thiết bị lưu trữ được sử dụng (nhiều đĩa). Đối với tài nguyên mạng, nhiều mạng được sử dụng.
- tỷ lệ dọc: còn được gọi là tăng tỷ lệ (scale up) và giảm tỷ lệ (scale down). Trong thể loại mở rộng quy mô này, kích thước của một tài nguyên riêng lẻ được thay đổi để cung cấp mức tài nguyên cần thiết. Đối với tài nguyên tính toán, điều này có nghĩa là thay đổi số lượng CPU hoặc dung lượng bộ nhớ (RAM) của tài nguyên đó. Đối với tài nguyên lưu trữ, điều này có nghĩa là thay đổi kích thước của thiết bị (đĩa lớn hơn/nhỏ hơn) hoặc thay đổi tốc độ đọc/ghi có sẵn trên tài nguyên. Đối với tài nguyên mạng, điều này có nghĩa là thay đổi mạng có băng thông cao hơn/thấp hơn.

Tỷ lệ dọc thường đơn giản hơn từ góc độ thiết kế giải pháp và khối lượng công việc. Tuy nhiên, tỷ lệ dọc có các giới hạn, trong đó các tài nguyên tính toán vật lý riêng lẻ có sẵn số lượng CPU tối đa, lượng bộ nhớ tối đa khả dụng. Không thể ảo hóa một tải tính toán duy nhất trên nhiều tài nguyên tính toán vật lý, do đó giới hạn tài nguyên tính toán vật lý trực tiếp giới hạn khả năng mở rộng theo chiều dọc tính toán có sẵn cho các dịch vụ mây. Đối với tài nguyên lưu trữ, các thiết bị lưu trữ riêng lẻ có giới hạn dung lượng, tuy nhiên, các dịch vụ lưu trữ mây ảo hóa có thể sử dụng nhiều tài nguyên lưu trữ vật lý cơ bản nhưng làm cho chúng xuất hiện dưới dạng một tài nguyên duy nhất và do đó cung cấp khả năng mở rộng theo chiều dọc đáng kể. Đối với tài nguyên mạng, một mạng vật lý duy nhất có giới hạn băng thông. Một dịch vụ mạng ảo hóa có thể hỗ trợ khả năng sử dụng nhiều tài nguyên mạng cơ bản song song và do đó cung cấp các giới hạn cao hơn.

Tỷ lệ ngang, trong đó nhiều phiên bản của tài nguyên được sử dụng song song, thường yêu cầu khối lượng công việc và thiết kế giải pháp được định hướng theo các đặc trưng của loại tỷ lệ này. Các kỹ thuật cụ thể được sử dụng để giải quyết những thách thức mà tỷ lệ ngang mang lại. Đối với tài nguyên lưu trữ, có thể xảy ra trường hợp ảo hóa tài nguyên lưu trữ cơ bản có thể che giấu sự hiện diện của nhiều thiết bị lưu trữ vật lý. Tuy nhiên, tài nguyên lưu trữ ảo hóa như vậy vẫn có thể tiết lộ các khía

cạnh của tài nguyên vật lý cơ bản, chẳng hạn như giới hạn về kích thước của các tệp hoặc đối tượng riêng lẻ vì có thể xảy ra trường hợp các thực thể lưu trữ đó không thể trải rộng trên nhiều thiết bị vật lý. Một thiết kế hỗ trợ việc sử dụng tỷ lệ ngang có tính đến các giới hạn đó. Tỷ lệ ngang của tài nguyên mạng ngụ ý rằng thiết kế giải pháp sử dụng nhiều kết nối riêng biệt trên các tài nguyên mạng khác nhau có sẵn. Tuy nhiên, một dịch vụ mạng ảo hóa có thể hỗ trợ khả năng sử dụng nhiều tài nguyên mạng cơ sở song song một cách minh bạch và loại bỏ nhu cầu thiết kế giải pháp để xử lý trực tiếp vấn đề này.

Tỷ lệ ngang cho tài nguyên tính toán thường được sử dụng trong tính toán mây, vì lý do về mặt kỹ thuật là không khả thi để ảo hóa ranh giới của tài nguyên tính toán. Tỷ lệ ngang của tài nguyên tính toán ngụ ý rằng nhiều phiên bản của một thành phần phần mềm nhất định được chạy song song, với công việc sắp tới được bàn giao trên các phiên bản đó. Thiết kế như vậy thừa nhận rõ ràng các giới hạn đối với tài nguyên tính toán (CPU, bộ nhớ) và điều này ủng hộ phương pháp thiết kế phân chia phần mềm thành các thành phần riêng biệt chạy trong các quy trình riêng biệt. Cách tiếp cận như vậy làm giảm số lượng CPU cần thiết và dung lượng bộ nhớ cần thiết cho mỗi thành phần. Các phương pháp thiết kế như kiến trúc vi dịch vụ giải quyết rõ ràng khía cạnh này của tỷ lệ ngang.

14.2 Các trường hợp song song và cân bằng tải

Khi sử dụng tài nguyên tính toán theo chiều ngang và nhiều phiên bản song song của một thành phần phần mềm xử lý các yêu cầu đến, sẽ có những hậu quả phải được giải quyết bằng thiết kế giải pháp.

Nhu cầu cấp thiết nhất là các yêu cầu đến cho một tố phần phần mềm cụ thể được chia sẻ trên tất cả các trường hợp của thành phần phần mềm đó. Về mặt logic, điều này có thể được xem như là xử lý một luồng yêu cầu hướng đến một thiết bị đầu cuối dịch vụ cụ thể (trên danh nghĩa thiết bị đầu cuối này có một địa chỉ mạng là đích của các yêu cầu). Mỗi trường hợp song song của thành phần phần mềm triển khai dịch vụ đều có thiết bị đầu cuối (riêng tư) riêng, mỗi trường hợp có một tổ hợp địa chỉ mạng và cổng riêng biệt. Một thành phần được gọi là bộ cân bằng tải được sử dụng để quản lý thiết bị đầu cuối dịch vụ và hướng từng yêu cầu dịch vụ đến một trong các trường hợp song song.

Bộ cân bằng tải được gắn vào thiết bị đầu cuối dịch vụ và xử lý tất cả các yêu cầu dịch vụ đến. Bộ cân bằng tải duy trì danh sách tất cả các trường hợp của thành phần phần mềm liên quan, ở mức tối thiểu, địa chỉ thiết bị đầu cuối và cổng của từng trường hợp cần được biết. Mỗi yêu cầu đến được gửi đến một trong các trường hợp thành phần phần mềm. Việc lựa chọn sử dụng trường hợp nào cho một yêu cầu đến được quyết định bởi một thuật toán điều phối (thuật toán điều phối nào được sử dụng có thể khác nhau).

Một thuật toán điều phối đơn giản là luân phiên, trong đó mỗi trường hợp được sử dụng lần lượt, nhằm cung cấp cùng một tỷ lệ phần trăm yêu cầu cho tất cả các trường hợp. Một biến thể của điều này là luân phiên theo trọng số, trong đó mỗi trường hợp được cho một trọng số và tỷ lệ phần trăm yêu cầu được cung cấp cho một trường hợp phản ánh trọng số đó. Một thuật toán điều phối tinh vi hơn có liên quan đến tải, một thuật toán dựa trên các yêu cầu chưa xử lý. Trong trường hợp này, bộ cân bằng tải phải biết số lượng yêu cầu vẫn đang được xử lý bởi mỗi trường hợp và hướng các yêu cầu mới đến đến trường hợp được tải ít nhất.

Bộ cân bằng tải phải xử lý một trường hợp động, Có thể bắt đầu các trường hợp mới và có thể dừng các trường hợp hiện có. Điều này có thể xảy ra vì lý do mở rộng quy mô và tính linh hoạt hoặc có thể xảy ra do lỗi của một trường hợp.

Bộ cân bằng tải có thể được triển khai bằng cách sử dụng máy chủ Proxy ngược (Reverse Proxy), máy chủ này có thể cung cấp các khả năng bổ sung, chẳng hạn như lưu vào bộ nhớ đệm nội dung tĩnh, xử lý SSL (mã hóa/giải mã) và bảo vệ chống tấn công.

14.3 Tính linh hoạt và tự động hóa

Tính linh hoạt là sự thích ứng của các tài nguyên được sử dụng bởi một thành phần cụ thể để thay đổi khối lượng công việc trên thành phần đó. Tính linh hoạt đạt được tốt nhất thông qua tự động hóa, mặc dù nó cũng có thể được thực hiện thủ công. Việc triển khai tính linh hoạt theo cách thủ công là không nên vì nó ngụ ý rằng nhân viên luôn sẵn sàng thực hiện các hành động và có khả năng đó cũng là một quy trình chậm hơn.

Tính linh hoạt tự động ngụ ý rằng có sự giám sát liên tục các thành phần có liên quan đối với việc sử dụng tài nguyên của chúng bằng công cụ linh hoạt. Công cụ linh hoạt được cấu hình với các quy tắc liên quan đến việc sử dụng tài nguyên, sao cho nếu việc sử dụng tài nguyên vượt quá một số mức đặt trước, hành động sẽ được thực hiện để tăng nguồn tài nguyên được phân bổ, trong khi nếu việc sử dụng tài nguyên giảm xuống dưới một số mức đặt trước, hành động sẽ được thực hiện để giảm mức tài nguyên được phân bổ.

Các quy tắc linh hoạt nên tính đến thực tế là có thể mất thời gian để phân bổ và giải quyết các nguồn lực. Điều này đặc biệt quan trọng khi phân bổ thêm nguồn lực, vì có khả năng cạn kiệt nguồn lực hiện tại trước khi có nguồn lực bổ sung.

14.4 Mở rộng quy mô cơ sở dữ liệu

Mở rộng quy mô cơ sở dữ liệu là một chủ đề quan trọng đối với tính toán mây. Mở rộng quy mô cơ sở dữ liệu áp dụng cho cả kích thước vật lý của cơ sở dữ liệu và cả tốc độ giao dịch của các hoạt động đối với cơ sở dữ liệu, cả hoạt động truy vấn và cập nhật.

Một cách tiếp cận phổ biến được sử dụng để mở rộng quy mô cơ sở dữ liệu là một hoặc một hình thức mở rộng theo chiều ngang khác bằng cách sử dụng nhiều máy và nhiều thiết bị lưu trữ cho một cơ sở dữ liệu. Vấn đề cơ bản với việc sử dụng khả năng mở rộng theo chiều ngang là giữ cho nhiều tài nguyên đồng bộ với nhau.

Một cách tiếp cận để giữ cho các tài nguyên được đồng bộ hóa là phân vùng hoặc phân đoạn cơ sở dữ liệu, để các nhóm bản ghi cơ sở dữ liệu khác nhau được giữ và xử lý bởi các tài nguyên khác nhau. Các yêu cầu đến được chia nhỏ và chuyển đến từng tài nguyên có liên quan và kết quả từ mỗi tài nguyên được kết hợp trước khi phản hồi cho yêu cầu được trả về.

Một cách tiếp cận khác để giữ cho các tài nguyên được đồng bộ hóa là phân cụm, sao cho các tài nguyên ở gần nhau (về thời gian giao tiếp giữa các tài nguyên) và có thể được quản lý bằng cách sử dụng trình giám sát giao dịch toàn cầu nhằm giữ mọi thứ nhất quán giữa các tài nguyên khác nhau. Cả hai cách tiếp cận này đều triển khai cái được gọi là tính nhất quán mạnh mẽ giữa các tài nguyên khác

nhau: một máy khách sử dụng cơ sở dữ liệu như vậy luôn nhận được kết quả giống nhau bất kể tài nguyên nào thực sự được sử dụng để xử lý yêu cầu. Tính nhất quán cao là cần thiết cho các ứng dụng mang tính giao dịch và trong đó thứ tự của các giao dịch là trung tâm của thiết kế.

Một cách tiếp cận rất khác để mở rộng quy mô cơ sở dữ liệu cũng tồn tại đối với tính toán mây, sử dụng khái niệm về tính nhất quán cuối cùng, có thể được sử dụng cho các loại ứng dụng phi giao dịch. Trong những trường hợp như vậy, các bản sao của cơ sở dữ liệu được lưu trữ ở nhiều vị trí và các bản cập nhật được thực hiện độc lập trên từng bản sao, với khả năng có sự khác biệt giữa nội dung của các bản sao trong một khoảng thời gian và cũng có khả năng xung đột các bản cập nhật với diễn ra. Tính nhất quán cuối cùng được triển khai bởi một số cơ sở dữ liệu NoSQL, chẳng hạn như mã nguồn mở CouchDB.

15 Bảo mật và các công nghệ phổ biến mây

15.1 Quy định chung

Bảo mật là một khía cạnh quan trọng của mọi hệ thống tính toán mây. Bảo mật liên quan đến một tập các biện pháp kiểm soát có khả năng giải quyết các rủi ro khác nhau. Các biện pháp kiểm soát được mô tả trong Điều này là chung, theo nghĩa có thể áp dụng cho các hệ thống không phải mây, nhưng được áp dụng cho các hệ thống mây theo những cách cụ thể được mô tả trong Điều này.

15.2 Tường lửa

Tường lửa là một thành phần bảo mật mạng giám sát và kiểm soát lưu lượng truy cập mạng, cả vào và ra, đồng thời cho phép hoặc chặn lưu lượng truy cập cụ thể dựa trên một bộ quy tắc bảo mật. Mục đích của tường lửa là tạo thành một rào cản giữa các mạng nội bộ được kiểm soát và bảo mật đáng tin cậy với các mạng bên ngoài không đáng tin cậy, về cơ bản ngăn chặn truy cập trái phép và không được kiểm soát vào các mạng nội bộ.

Đối với tính toán mây, thường xảy ra trường hợp các mạng bên ngoài không đáng tin cậy là internet hoặc các kết nối mạng truy cập mây khác như được mô tả trong 13.2. Các mạng bảo mật đáng tin cậy là những mạng được sử dụng cho kết nối mạng nội bộ mây như được mô tả trong 13.3.

CSC có thể triển khai phần mềm tường lửa của riêng họ vào môi trường mây bằng cách sử dụng dịch vụ mây IaaS thích hợp. Điều này có thể được mong muốn nếu CSC muốn sử dụng phần mềm tường lửa cụ thể hoặc khi CSC cần kiểm soát chi tiết cấu hình của tường lửa.

Tuy nhiên, tình huống phổ biến hơn là tường lửa được CSP cung cấp dưới dạng dịch vụ mây trong môi trường mây. Tường lửa do CSP cung cấp có thể dựa trên phần cứng hoặc dựa trên phần mềm và CSP quyết định (những) cái nào được cung cấp.

Tường lửa thường được cung cấp cho bất kỳ thiết bị đầu cuối nào kết nối từ mạng bên ngoài với mạng nội bộ.

Đối với các dịch vụ mây có khả năng ứng dụng, CSP thường cung cấp các khả năng tường lửa như một phần của dịch vụ mây.

15.3 Bảo vệ thiết bị đầu cuối

Việc sử dụng các dịch vụ mây thường liên quan đến việc CSC hiển thị một hoặc nhiều thiết bị đầu cuối dưới dạng hiển thị công khai từ mỗi giải pháp. Mọi thiết bị đầu cuối hiển thị công khai như vậy đều cần được bảo vệ áp dụng cho nó và điều này thường được cung cấp như một phần của dịch vụ mây. Bảo vệ thường bao gồm:

- tường lửa;
- chống tấn công từ chối dịch vụ phân tán (DDoS);
- quét lỗ hổng.

15.4 Quản lý danh tính và quyền truy cập

Kiểm soát quyền truy cập vào tài nguyên tính toán mây là một khả năng cần thiết cho mọi hoạt động sử dụng tính toán mây. Các tài nguyên liên quan không chỉ bao gồm bản thân các dịch vụ mây mà còn cả các thành phần CSC như ứng dụng, dữ liệu và dịch vụ được triển khai trong các dịch vụ mây.

Bản thân các dịch vụ mây thường được CSP cung cấp khả năng quản lý truy cập và nhận dạng. Khả năng nhận dạng và truy cập đối với các tài nguyên khác thường được cung cấp dưới dạng dịch vụ mây do CSP cung cấp. Thông thường, cùng một dịch vụ mây quản lý danh tính và truy cập có thể được sử dụng cho cả bản thân các dịch vụ mây cũng như cho các tài nguyên CSC.

Các khả năng cần được xem xét đối với dịch vụ mây quản lý truy cập và nhận dạng mây bao gồm:

- hỗ trợ các kỹ thuật xác thực nâng cao bao gồm sinh trắc học và đa yếu tố;
- hỗ trợ tích hợp các khả năng quản lý truy cập và nhận dạng tại chỗ với dịch vụ mây;
- hỗ trợ đăng nhập một lần (SSO);
- hỗ trợ ủy quyền chi tiết.

15.5 Mã hóa dữ liệu

Để hỗ trợ tính bảo mật và quyền riêng tư, mã hóa dữ liệu là một chức năng chính. Đối với tính toán mây, dữ liệu có thể được mã hóa khi ở trạng thái nghỉ và khi chuyển động.

Mã hóa dữ liệu đang chuyển động thường đạt được thông qua việc sử dụng giao thức bảo mật, chẳng hạn như HTTPS và SSL/TLS hoặc thông qua việc sử dụng các khả năng toàn diện hơn như VPN. Các khả năng như vậy có thể được cung cấp bởi các dịch vụ mây.

Mã hóa dữ liệu ở trạng thái nghỉ có thể được thực hiện bởi CSC (hoặc bằng mã do CSC cài đặt và vận hành), nhưng nhiều dịch vụ mây lưu trữ cũng cung cấp khả năng mã hóa dữ liệu được lưu trữ.

15.6 Quản lý khóa

Việc sử dụng mã hóa nhất thiết liên quan đến việc sử dụng các khóa mã hóa. Việc quản lý các khóa mã hóa là cần thiết để cung cấp một giải pháp an toàn phù hợp. Chẳng hạn, lưu trữ khóa cùng với ứng dụng là không khôn ngoan, trong trường hợp kẻ tấn công có thể lấy được bản sao của khóa và do đó xâm phạm dữ liệu được mã hóa.

Các khả năng quản lý khóa thường được cung cấp dưới dạng dịch vụ mây, để tích hợp với các tài nguyên có thể là chính các dịch vụ mây hoặc là các tài nguyên do CSC triển khai trong các dịch vụ mây. Các dịch vụ mây quản lý khóa thường dựa trên việc CSP sử dụng các mô-đun bảo mật phần cứng để cung cấp mức độ bảo mật thích hợp cho các khóa mã hóa mà chúng quản lý. Một số dịch vụ mây quản lý khóa tạo và lưu trữ khóa, trong khi các dịch vụ khác cho phép "mang theo khóa của riêng bạn" (BYOK) nơi CSC có thể tạo các khóa cần thiết và chuyển chúng đến dịch vụ mây để lưu trữ và sử dụng. BYOK có thể có tầm quan trọng đặc biệt khi triển khai nhiều mây và khi các yêu cầu mã hóa trải rộng trên nhiều dịch vụ mây khác nhau. Một ví dụ là nơi một bản sao của một số dữ liệu được mã hóa được lưu trữ trong một dịch vụ mây khác với dịch vụ mây ban đầu (có thể có một CSP khác).

Phụ lục A

(tham khảo)

Ảnh tượng VM và ảnh tượng đĩa

A.1 Các định dạng ảnh tượng VM và ảnh tượng đĩa

Ví dụ về các định dạng ảnh tượng đĩa và ảnh tượng máy ảo đang được sử dụng bao gồm:

Ảnh tượng máy Amazon (Amazon machine image (AMI))

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

Độc quyền

Ảnh tượng máy VMWare VMX

https://www.vmware.com/support/ws5/doc/ws_learning_files_in_a_vm.html

Độc quyền

Định dạng đĩa Virtual Disk Image

https://www.virtualbox.org/

Nguồn mở theo giấy phép GPL Phiên bản 2

Định dạng đĩa Virtual Hard Disk eXtended (VHDX)

https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-VHDX/[MS-VHDX].pdf

Độc quyền của Microsoft nhưng được cung cấp theo "lời hứa về thông số kỹ thuật mở".

Định dạng đĩa VMWare Virtual Disk Format (VMDK)

https://www.vmware.com/support/developer/vddk/vmdk_50_technote.pdf

Được phát triển bởi VMWare nhưng hiện là định dạng mở.

Định dạng đĩa QEMU Sao chép-trong-bản viết (QCOW, QCOW2)

https://github.com/libyal/libqcow/blob/master/documentation/QEMU%20Copy-On-Write%20file%20format.asciidoc

Nguồn mở theo giấy phép GPL Phiên bản 1.3

Thư mục tài liệu tham khảo

[1] TCVN 12480:2020 (ISO/IEC 17788:2014), Tính toán mây – Tổng quan và từ vựng [3]

[2] TCVN 12481:2019 (ISO/IEC 17789:2014), Tính toán mây – Kiến trúc tham chiếu [4]

[3] Google, *Google App Engine*, https://cloud.google.com/appengine/

[4] IBM, *IBM Cloud Functions*, https://www.ibm.com/cloud/functions

[5] Amazon, *AWS Lambda*, https://aws.amazon.com/lambda/

[6] Apache Software Foundation, Dự án nền tảng mây không server OpenWhisk, [trực tuyến] [đã xem 2019-10-7], https://openwhisk.apache.org/

[7] Cloud Standards Customer Council, Hướng dẫn thực tế về Nền tảng như một dịch vụ (2015), http:// www.cloud-council.org/deliverables/CSCC-Practical-Guide-to-PaaS.pdf

[8] Open Containers Initiative, Đặc tả thời gian chạy v1.0.1, https://github.com/ opencontainers/runtime-spec/releases/download/v1.0.1/oci-runtime-spec-v1.0.1.pdf

[9] Open Containers Initiative, Đặc tả định dạng ảnh tượng, v1.0.1, https://github .com/opencontainers/image-spec/releases/download/v1.0.1/oci-image-spec.-v1.0.1.pdf

[10] Newman S., Building Microservices. O'Reilly (2015).

[11] Lewis J., Fowler M. Microservices, [trực tuyến]. [đã xem 2019-10-7], Có sẵn tại http://martinfowler.com/articles/microservices.html

[12] Paul C., Jamshidi P., *Các vi dịch vụ: Một nghiên cứu lập bản đồ có hệ thống*, Kỷ yếu Hội nghị Quốc tế lần thứ 6 về Khoa học Dịch vụ và Tính toán Mây - Tập 1 và 2, (CLOSER 2016).

[13] Namiot D., Sneps-Sneppe M., Về kiến trúc vi dịch vụ, *Tạp chí Quốc tế về Công nghệ Thông tin Mở*, tập 2,số 9, 2014.

[14] Thones J., Microservices, Phần mềm IEEE, 32(1), 2015.

[15] Alshuqayran N., Ali N., Evans R. A, *Nghiên cứu lập bản đồ hệ thống trong kiến trúc vi dịch vụ*, Kỷ yếu của IEEE tại Hội nghị quốc tế lần thứ 9 về Ứng dụng và tính toán theo hướng dịch vụ, 2016.

[16] The New Stack, Tự động hóa và điều phối với Docker và Vùng chứa (2016), https:// thenewstack.io/ebooks/docker-and-containers/automation-orchestration-docker-containers/

[17] Cloud Native Computing Foundation, *Kubernetes*, https://kubernetes.io/

[18] ISO/IEC 17203:2017, *Information technology - Open Virtualization Format (OVF) specification* (Công nghệ thông tin - Đặc tả Định dạng ảo hóa mở (OVF))[5]

[19] Brad Calder et al, *Lưu trữ Windows Azure: Dịch vụ lưu trữ mây tính có sẵn cao với tính nhất quán cao*, SOSP '11 Kỷ yếu của Hội nghị chuyên đề ACM lần thứ 23 về Các nguyên tắc hệ điều hành, 2011

[3] https://www.iso.org/standard/60544.html
[4] https://www.iso.org/standard/60545.html
[5] https://www.iso.org/standard/72081.html

[20] IETF, *Hệ thống tệp mạng (NFS) Phiên bản 4 Giao thức Phiên bản nhỏ 2*, https://tools.ietf .org/html/rfc7862

[21] Amazon, *Hệ thống tệp linh hoạt của Amazon*, [trực tuyến] [đã xem 2019-10-7], Có sẵn tại https://aws.amazon.com/efs/

[22] IBM, *Lưu trữ tệp*, [trực tuyến] [đã xem 2019-10-07], https://console.bluemix.net/catalog/infrastructure/file-storage

[23] IETF RFC 7143, *Giao thức Giao diện Hệ thống Máy tính Nhỏ trên Internet* (iSCSI) [6]

[24] Apache Software Foundation, *Nền tảng phát trực tuyến bàn giao Kafka*, [trực tuyến] [đã xem 2019-10- 7], https://kafka.apache.org/

[25] Amazon, *Tham chiếu API dịch vụ lưu trữ đơn giản của Amazon*, [trực tuyến] [đã xem 2019-10-7], https://docs.aws.amazon.com/AmazonS3/latest/API/s3-api.pdf

[26] ISO 23257 [7], *Blockchain and distributed ledger technologies - Reference architecture* (Công nghệ chuỗi khối và sổ cái phân tán - Kiến trúc tham khảo) [8]

[27] Toby Clemson, Các chiến lược thử nghiệm trong kiến trúc vi dịch vụ, [trực tuyến] [đã xem 2019-10-7], https://martinfowler.com/articles/microservice-testing/#anatomy-modules

[28] Eclipse Foundation, *Tệp vi mô API*, [trực tuyến] [đã xem 2019-10-7] [9]

[29] Istio, *Nền tảng mạng lưới dịch vụ*, [trực tuyến] [đã xem 2019-10-7], https://istio.io/

[30] Linkerd, *Dịch vụ phụ xe*, [trực tuyến] [đã xem 2019-10-7], https://linkerd.io/

[31] Eric Evans, Thiết kế hướng miền: Giải quyết sự phức tạp trong trung tâm của phần mềm, Giáo sư Addison-Wesley (2003)

[32] Git, *Quản lý kiểm soát nguồn Git*, [trực tuyến] [đã xem 2019-10-7], https://git-scm.com

[33] Jenkins, *Máy chủ tự động Jenkins*, [trực tuyến] [đã xem 2019-10-7], https://jenkins.io

[34] Red Hat Ansible, *Nền tảng tự động hóa CNTT Ansible*, [trực tuyến] [đã xem 2019-10-7], https://www.ansible.com

[35] CFEngine, *Công nghệ quản lý cấu hình CFEngine*, [trực tuyến] [đã xem 2019-10-7], https://cfengine.com/product/community/

[36] Chef, *Tự động hóa đầu bếp*, [trực tuyến] [đã xem 2019-10-7], https://github.com/chef/chef-workstation

[37] Puppet, Phần mềm tự động hóa con rối, [trực tuyến] [đã xem 2019-10-7], https://github.com/puppetlabs/puppet

[38] Gartner, DevSecOps: Cách tích hợp liền mạch bảo mật vào DevOps, (2016), https://www.gartner.com/doc/3463417/devsecops-seamlessly-integrate-security-devops

[39] Gartner, 10 điều cần làm ngay để DevSecOps thành công, (2017),

[6] https://tools.ietf.org/html/rfc7143
[7] Đã công bố.
[8] https://www.iso.org/standard/75093.html
[9] https://microprofile.io/

https://www.gartner.com/doc/3811369/-things-right-successful-devsecops

[40] NIST SP 800-77 (2005), Hướng dẫn về các IPsec VPN [10]

[41] IETF RFC 7348 (2014), Mạng cục bộ có thể mở rộng ảo (VXLAN): Khung cho lớp phủ mạng lớp 2 được ảo hóa trên mạng lớp 3 [11]

[42] VMWare (2018), *Các vùng chứa và mạng vùng chứa dành cho các kỹ sư mạng*, [trực tuyến] [đã xem 2019-10-7], https://www.vmware.com/content/dam/digitalmarketing/ vmware/en/pdf/products/nsx/vmware-containers-and-container-networking-whitepaper.pdf

[43] Project Calico, Tổng quan dự án Calico, [trực tuyến] [đã xem 2019-10-7], https://www. projectcalico.org/

[44] Weaveworks, Mạng đan xen, [trực tuyến] [đã xem 2019-10-7], https://www.weave.works/ oss/net/

[45] Gartner (2014): *Nền tảng như một dịch vụ: Định nghĩa, phân loại và bối cảnh nhà cung cấp, 2014*. https://www.gartner.com/doc/2833022/platform-service-definition-taxonomy-vendor

[46] Gartner (2018): *Chu kỳ cường điệu cho nền tảng như một dịch vụ*, 2018. https://www. gartner.com/doc/3885966/hype-cycle-platform-service-

[47] Apache Software Foundation, *Dự án CouchDB*, [trực tuyến] [đã xem 2019-10-7], http://couchdb.apache.org/

[48] The New Stack, *Hướng dẫn về Công nghệ không server (2018)*, https://thenewstack.io/ ebooks/ serverless/guide-to-serverless-technologies/

[49] Cloud Native Computing Foundation, *Đặc trưng Sự kiện mây*, [trực tuyến] [đã xem 2019-10-7] https://cloudevents.io/

[50] Serverless Inc, *Khung không server*, [trực tuyến] [đã xem 2019-10-7], https://github.com/ serverless/serverless

[51] Cloud Native Computing Foundation (2018), *Sách trắng không server v1.0,* [trực tuyến] [đã xem 2019-10-7], https://github.com/cncf/wg-erverless/blob/master/whitepapers/ serverless-overview/cncf_serverless_whitepaper_v1.0.pdf

[52] Archfirst (2015), *Thiết kế theo hướng miền - Kiến trúc phân lớp*, [trực tuyến] [đã xem 2019-10-7], https://archfirst.org/domain-driven-design-6-layered-architecture/

[53] Microsoft (2018), Thiết kế một DDD-hướng vi dịch vụ, [trực tuyến] [đã xem 2019-10-7], https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/ microservice-ddd-cqrs-patterns/ddd-oriented-microservice

[54] Methods & Tools, *Giới thiệu về thiết kế theo hướng miền*, http://www.methodsandtools. com/archive/archive.php?id=97

[55] Martin Fowler (2014), *Bộ ngắt mạch*, [trực tuyến] [đã xem 2019-10-7], https://martinfowler. com/bliki/CircuitBreaker.html

[56] Microsoft (2019), *Mẫu cổng API so với Giao tiếp trực tiếp giữa máy khách với vi dịch vụ*, [trực

[10] https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-77.pdf
[11] https://tools.ietf.org/html/rfc7348

tuyến] [đã xem 2019-10-7] https://docs.microsoft.com/enus/dotnet/standard/ microservices-architecture/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern

[57] Apache Software Foundation, Dự án Mesos, http://mesos.apache.org/

[58] HashiCorp, *Dự án Nomad*, [trực tuyến] [đã xem 2019-10-7], https://www.nomadproject.io/

[59] Cloud Foundry, *Nền tảng ứng dụng mây mã nguồn mở*, [trực tuyến] [đã xem 2019-10-7], https:// www.cloudfoundry.org/

[60] Microsoft, *Các chức năng* Azure, [trực tuyến] [đã xem 2019-10-7], https://azure.microsoft. com/en-us/services/functions/

[61] Google, *Nền tảng tính toán không server của các chức năng mây*, [trực tuyến] [đã xem 2019-10-7], https://cloud.google.com/functions/

[62] IBM, *Chức năng mây Nền tảng FaaS*, [trực tuyến] [đã xem 2019-10-7], https://console. bluemix.net/openwhisk/

[63] Oracle, *Nền tảng không server Fn*, [trực tuyến] [đã xem 2019-10-7], https://fnproject.io/

[64] Amazon, *Cơ sở dữ liệu không server Aurora*, [trực tuyến] [đã xem 2019-10-7], https:// aws.amazon.com/rds/aurora/serverless/

[65] Fauna, FaunaDB, [trực tuyến] [đã xem 2019-10-7], https://fauna.com/

[66] Google, *Cloud Firestore*, [trực tuyến] [đã xem 2019-10-7], https://cloud.google.com/ firestore/

[67] IBM, *Cơ sở dữ liệu Cloudant*, [trực tuyến] [đã xem 2019-10-7], https://console.bluemix.net/ catalog/services/cloudant

[68] Microsoft, *Hồ dữ liệu Azure*, [trực tuyến] [đã xem 2019-10-7], https://azure.microsoft.com/ en-us/solutions/data-lake/

[69] ISO/IEC 18384-1:2016, Information technology - Reference Architecture for Service Oriented Architecture (SOA RA) - Part 1: Terminology and concepts for SOA (Công nghệ thông tin - Kiến trúc tham chiếu cho Kiến trúc hướng dịch vụ (SOA RA) – Phần 1: Thuật ngữ và khái niệm về SOA) [12]

[70] Linux Foundation, *Đặc tả API mở*, [trực tuyến] [đã xem 2019-10-7], https:// www.openapis.org/

[71] ISO/IEC 9075:2016, *Information technology - Database languages - SQL - Part 1: Framework (SQL/Framework)* (Công nghệ thông tin – Ngôn ngữ cơ sở dữ liệu – SQL – Phần 1: Khung (SQL/khung)) [13]

[72] ISO/IEC 9660:1988, *Information processing - Volume and file structure of CD-ROM for information interchange* (Xử lý thông tin – Khối lượng và cấu trúc tệp của CD-ROM để trao đổi thông tin) [14]

[73] ISO/IEC 13346-1:1995, *Information technology - Volume and file structure of write-once and*

[12] https://www.iso.org/standard/63104.html
[13] https://www.iso.org/standard/63555.html
[14] https://www.iso.org/standard/17505.html

rewritable media using non-sequential recording for information interchange - Part 1: General (Công nghệ thông tin – Khối lượng và cấu trúc tệp của phương tiện ghi một lần và ghi lại bằng cách sử dụng ghi không tuần tự để trao đổi thông tin – Phần 1: Chung) [15]

[74] Amazon, *Lưu trữ mây AWS*, [trực tuyến] [đã xem 2019-10-7], https://aws.amazon. com/what-is-cloud-storage

[75] Redhat Inc, *Lưu trữ Redhat*, [trực tuyến] [đã xem 2019-10-7], https://www.redhat.com/ en/topics/data-storage

[76] Microsoft, *Lưu trữ Azure Microsoft*, [trực tuyến] [đã xem 2019-10-7], https://azure. microsoft.com/en-us/services/storage

[77] IETF RFC 7230 (2014), Giao thức truyền siêu văn bản HTTP [16]

[78] Linux Foundation, Thời gian chạy Node.js®, [trực tuyến] [đã xem 2019-10-7], https:// nodejs.org/en/

[79] Docker Inc, *Docker Swarm*, [trực tuyến] [đã xem 2019-10-7], https://github.com/docker/ swarm

[80] Docker Inc, *Docker Hub*, [trực tuyến] [đã xem 2019-10-7], https://hub.docker.com

[81] YAML.org, *Tiêu chuẩn tuần tự hóa dữ liệu YAML*, [trực tuyến] [đã xem 2019-10-7], https://yaml.org/

[82] ECMA-404:2017, *Cú pháp trao đổi dữ liệu JSON*, http://www.ecma-international.org/ publications/files/ECMA-ST/ECMA-404.pdf

[83] Oracle, *Cơ sở dữ liệu tự điều khiển của Oracle*, [trực tuyến] [đã xem 2019-10-7], https:// www.oracle.com/database/autonomous-database.html

[84] Oracle, Cơ sở dữ liệu Oracle NoSQL, [trực tuyến] [đã xem 2019-10-7], https://cloud.oracle. com/nosql

[85] ISO/IEC 21878:2018, *Information technology - Security techniques - Security guidelines for design and implementation of virtualized servers* (Công nghệ thông tin – Kỹ thuật bảo mật – Hướng dẫn bảo mật cho thiết kế và triển khai máy chủ ảo hóa) [17]

[86] ISO/IEC/IEEE 24765:2017, *Systems and software engineering - Vocabulary* (Các hệ thống và kỹ thuật phần mềm – Từ vựng) [18]

[87] ISO/IEC 18384-3, *Information technology - Reference Architecture for Service Oriented Architecture (SOA RA) - Part 3: Service Oriented Architecture ontology* (Công nghệ thông tin – Kiến trúc tham chiếu cho Kiến trúc hướng dịch vụ (SOA RA) – Phần 3: Bản thể luận kiến trúc hướng dịch vụ) [19]

---

[15] https://www.iso.org/standard/26783.html
[16] https://tools.ietf.org/html/rfc7230
[17] https://www.iso.org/standard/72029.html
[18] https://www.iso.org/standard/71952.html
[19] https://www.iso.org/standard/63106.html