

TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN



BÀI BÁO CÁO

Học phần: Thực Tập Viết Niên Luận - Nhóm 1

Mã học phần: 2024-2025.2.TIN3142.001

KHÁM PHÁ VÀ TRIỂN KHAI DỰ ÁN WEB DEMO TRÊN AMAZON WEB SERVICE

Giảng viên hướng dẫn: **Thầy Nguyễn Quang Hưng**

Danh sách thành viên: **22T1020684_Phó Đức Minh Nghĩa**

HUẾ . THÁNG 8 - 2025

MỤC LỤC

Giới thiệu khái quát

CHƯƠNG 1: KHÁI NIỆM CHUNG

1. Điện toán đám mây là gì?
 - Sơ lược về công nghệ ứng dụng trong điện toán đám mây.
 - Ưu điểm của điện toán đám mây.
2. Các mô hình dịch vụ điện toán đám mây.
 - IaaS – Infrastructure as a Service.
 - PaaS – Platform as a Service.
 - SaaS – Software as a Service.
3. Các mô hình triển khai điện toán đám mây.
 - Điện toán đám mây riêng tư (Private cloud).
 - Điện toán đám mây công cộng (Public cloud).
 - Điện toán đám mây chung (Community cloud).
 - Điện toán đám mây lai (Hybrid cloud).

CHƯƠNG 2: ĐIỆN TOÁN ĐÁM MÂY AWS.

1. Giới thiệu nhanh về AWS.
 - Giải pháp theo trường hợp sử dụng.
 - Giới thiệu các dịch vụ của AWS Cloud.
 - Dịch vụ tính toán (Compute).
 - Kết nối mạng (Networking & Content Delivery).
 - Cơ sở dữ liệu (Database).
 - Ban quản lý (Management & Governance).
 - Thùng (Container).
 - Lưu trữ phân phối nội dung (Storage).
 - Các dịch vụ khác (Other).

CHƯƠNG 3: NHỮNG DỊCH VỤ CỦA AWS

1. Những dịch vụ phổ biến của AWS.
 - Amazon EC2
 - Amazon RDS
 - Amazon S3
 - Amazon Lambda
 - AWS API Gateway
 - AWS Load Balancer
 - AWS VPC (Networking)
 - Amazon ECS
 - Amazon Route 53
 - Amazon IAM

CHƯƠNG 4: THỰC HÀNH VỚI CÁC DỊCH VỤ AWS

Cơ sở lý thuyết

1. Ngôn ngữ lập trình Java
2. Khái quát về Spring Boot Framework
3. Angular
4. MySQL
5. Docker

Thực hành

1. LAB-1
2. LAB-2
3. LAB-3

Tổng Kết

LINK TÀI LIỆU THAM KHẢO:

GIỚI THIỆU

Với sự phát triển vượt bậc của Internet, điện toán đám mây đã trở nên rất quen thuộc trong môi trường doanh nghiệp. Hoạt động kinh doanh, sản xuất và thương mại điện tử đã khẳng định được vai trò quan trọng trong việc thúc đẩy sự phát triển của các doanh nghiệp. Do đó, triển khai mô hình kinh doanh đòi hỏi các doanh nghiệp phải tìm cách đưa tài nguyên hiện có của họ lên Internet. Cách thông thường đầu tiên là sử dụng máy chủ VPS Server, tuy nhiên, phương pháp này tốn kém và phức tạp trong việc thiết lập, duy trì và khắc phục sự cố.

Hiện nay, có một số cách triển khai thông dụng như Dedicated Server, Virtual Private Server (VPS) và Shared Hosting. Dedicated Server là máy chủ vật lý dành riêng cho một doanh nghiệp, chạy một ứng dụng hoặc trang web duy nhất. Tuy nhiên, phương pháp này tốn kém và yêu cầu bảo trì cao và bảo mật cao. VPS là máy chủ vật lý dành riêng cho một doanh nghiệp, nhưng được ảo hóa thành các máy con chạy nhiều ứng dụng web hoặc trang web. Điều này giúp quản lý tài nguyên tốt hơn, nhưng tốn kém khi mở rộng trong mô hình lớn. Shared Hosting là máy chủ vật lý được chia sẻ bởi hàng trăm doanh nghiệp, giá thành rẻ nhưng có chức năng hạn chế và bảo mật kém.

Điện toán đám mây là một giải pháp tốt hơn, trong đó nhiều máy chủ vật lý hoạt động như một hệ thống do một công ty uy tín thiết lập. Các hệ thống này được trùm tượng hóa thành nhiều dịch vụ đám mây, ví dụ như AWS (Amazon Web Services) của Amazon. Cloud Hosting mang lại tính linh hoạt, khả năng mở rộng, độ an toàn và tiết kiệm chi phí. Nó cũng cung cấp khả năng cấu hình cao và hỗ trợ các công nghệ như chạy ứng dụng, ảo hóa, lưu trữ tài nguyên, phân tích dữ liệu, máy học, IoT và big data.

Với sự phát triển của các tập đoàn công nghệ lớn như Amazon, Microsoft và Google, điện toán đám mây đã trở thành một phần không thể thiếu trong cơ sở hạ tầng công nghệ của nhiều doanh nghiệp. Việc triển khai các dịch vụ điện toán đám mây giúp doanh nghiệp tiết kiệm thời gian và nguồn lực trong việc quản lý và vận hành hệ thống. Ngoài ra, điện toán đám mây cung cấp khả năng mở rộng linh hoạt, cho phép doanh nghiệp tăng hoặc giảm quy mô theo nhu cầu thực tế.

Bên cạnh đó, điện toán đám mây cũng mang lại lợi ích về bảo mật dữ liệu. Các nhà cung cấp dịch vụ điện toán đám mây đầu tư mạnh vào việc bảo vệ thông tin của khách hàng, từ việc mã hóa dữ liệu đến cung cấp các công nghệ bảo mật tiên tiến như xác thực hai yếu tố và quản lý quyền truy cập.

Không chỉ dành riêng cho doanh nghiệp lớn, điện toán đám mây cũng là một lựa chọn phù hợp cho các doanh nghiệp nhỏ và vừa. Bằng cách sử dụng dịch vụ điện toán đám mây, các doanh nghiệp nhỏ có thể tiết kiệm chi phí đầu tư ban đầu cho việc mua sắm

và duy trì cơ sở hạ tầng máy chủ riêng. Thay vào đó, họ chỉ cần trả phí cho việc sử dụng tài nguyên theo nhu cầu thực tế.

Tóm lại, điện toán đám mây đã thay đổi cách các doanh nghiệp triển khai và vận hành hệ thống công nghệ thông tin. Với tính linh hoạt, khả năng mở rộng và tiết kiệm chi phí, điện toán đám mây đóng vai trò quan trọng trong việc nâng cao hiệu suất và hiệu quả của các doanh nghiệp. Việc triển khai mô hình điện toán đám mây không chỉ là một xu hướng, mà còn là một yếu tố quan trọng để đảm bảo sự phát triển bền vững và cạnh tranh của các doanh nghiệp trong thời đại số.

GITHUB:

Link: <https://github.com/minhnhgia0910/NIENLUAN>

Mô tả:

- Nhánh main: [AWS-ECS-Lab3](#) (microservice basic demo cloudformation)
- Nhánh tree/Lab1-EC2: [AWS-EC2-Lab1](#) (basic spring + RDS + EC2 + S3)
- Nhánh tree/Lab2-Lambda: [AWS-SERVERLESS-Lab2](#) (basic spring + API Gateway +Lambda)

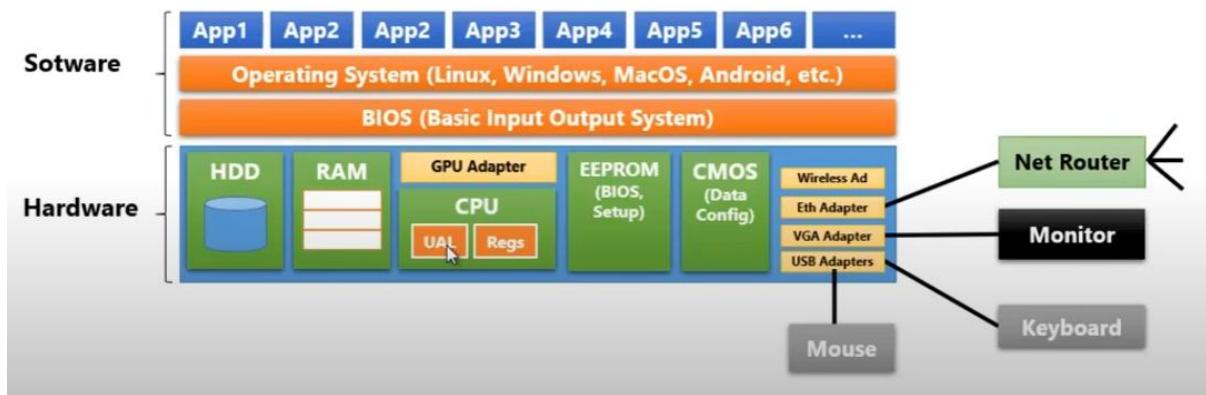
Học tập và thực hành cơ bản việc triển khai ứng dụng lên AWS trong thời gian học tập hè chú trọng vào các dịch vụ chính của Amazon Web Service.

Chương I: Khái niệm chung:

1 Điện toán đám mây là gì (Cloud Computing)?

Định nghĩa đơn giản nhất là những tòa nhà lớn chứa đầy máy tính được kết nối với nhau các máy tính này được xem như là các máy chủ con nhận nhiệm vụ cung cấp dịch vụ cho khách hàng. Nền tảng dịch vụ đám mây cung cấp khả năng truy cập nhanh chóng với chi phí thấp và tính sẵn sàng cao không cần phải đầu tư trả trước vào phần cứng và cấu hình quản lý chúng. Bạn có thể truy cập bao nhiêu tài nguyên dịch vụ tùy thích, gần như là ngay lập tức vì tính sẵn sàng cao và chỉ trả tiền cho những gì bạn sử dụng. Quyền truy cập theo yêu cầu, trả tiền khi sử dụng dịch vụ, quản lý bill khi sử dụng là những dịch vụ cơ bản của mô hình điện toán đám mây.

1.1 Sơ lược về công nghệ ứng dụng trong điện toán đám mây



Classique computer architecture

Kiến trúc máy tính cổ điển đề cập đến thiết kế và tổ chức truyền thống của một hệ thống máy tính. Nó bao gồm một số thành phần chính hoạt động cùng nhau để thực hiện các hướng dẫn và thực hiện tính toán. Dưới đây là các thành phần chính của kiến trúc máy tính cổ điển:

Kiến trúc máy tính cổ điển bao gồm các thành phần chính hoạt động cùng nhau để thực hiện các hướng dẫn và tính toán trong một hệ thống máy tính. Các thành phần này bao gồm CPU, bộ nhớ, thiết bị vào/ra (I/O), bus, ISA, hệ điều hành, đơn vị điều khiển, pipelining, bộ nhớ đệm và ngắt.

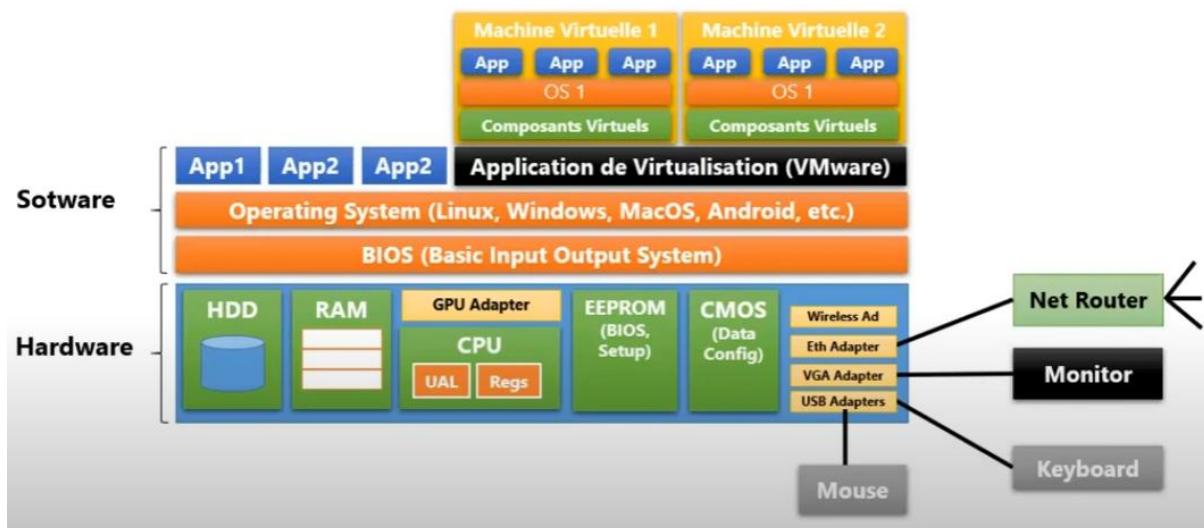
Tuy nhiên, kiến trúc máy tính cổ điển có một số hạn chế về việc phân chia tài nguyên vật lý giữa các ứng dụng và hệ điều hành. Đó đã mở ra một bước nhảy vọt mới trong công nghệ, đó là công nghệ ảo hóa. Công nghệ ảo hóa cho phép trùu tượng hóa tài nguyên máy tính, cho phép truy cập dễ dàng vào các tài nguyên trước khi được trùu tượng hóa. Điều này không bị giới hạn bởi việc triển khai, vị trí địa lý hoặc cấu hình vật lý của các tài nguyên cơ bản.

Công nghệ ảo hóa đã có một sự phát triển mạnh mẽ và được ứng dụng rộng rãi trong điện toán đám mây. Nó mang lại những lợi ích về quản lý hiệu quả và thuận tiện cho quản trị viên trung tâm dữ liệu và cải thiện tỷ lệ sử dụng tài nguyên của trung tâm dữ liệu. Công nghệ ảo hóa đã trở thành công nghệ tiên tiến nhất trong một loạt các cuộc cách mạng công nghệ, tăng mức độ ảo hóa hệ thống và tăng hiệu suất làm việc của máy tính lên một cấp độ chưa từng có trước đây.

Công nghệ ảo hóa(*)

Công nghệ ảo hóa (virtualization) là công nghệ quan trọng nhất ứng dụng trong điện toán đám mây. Công nghệ ảo hóa ra đời cùng với sự xuất hiện của công nghệ máy tính và luôn đóng vai trò thiết yếu trong sự phát triển của công nghệ máy tính. Từ việc giới thiệu khái niệm ảo hóa vào những năm 1950 đến việc thương mại hóa ảo hóa trên máy tính lớn của IBM vào những năm 1960, từ bộ nhớ ảo của hệ điều hành đến máy ảo Java, đến công nghệ ảo hóa máy chủ dựa trên kiến trúc x86 Sự phát triển mạnh mẽ Sự phát triển của ảo hóa đã bổ sung thêm những ý nghĩa vô cùng phong phú cho khái niệm ảo hóa có vẻ trùu tượng. Trong những năm gần đây, với sự phổ biến của công nghệ ảo hóa máy chủ, các phương pháp quản lý và triển khai trung tâm dữ liệu mới đã phát triển rộng rãi và không còn xa lạ đối với người trong ngành, mang lại trải nghiệm quản lý hiệu quả và thuận tiện cho các quản trị viên trung tâm dữ liệu. Công nghệ này cũng có thể cải thiện tỷ lệ sử dụng tài nguyên của trung tâm dữ liệu. Tất cả những điều này làm cho công nghệ ảo hóa trở thành tâm điểm của toàn bộ ngành công nghiệp thông tin là công nghệ tiên tiến nhất trong một loạt các cuộc cách mạng công nghệ nhằm tăng mức độ ảo hóa hệ thống, cho phép tăng hiệu suất làm việc của máy tính lên một cấp độ chưa từng có.

Định nghĩa: Theo Wikipedia, ảo hóa là một phương pháp trùu tượng để thể hiện tài nguyên máy tính. Thông qua ảo hóa, các tài nguyên được trùu tượng hóa có thể được truy cập giống như các tài nguyên trước khi được trùu tượng hóa. Phương pháp tài nguyên trùu tượng này không bị giới hạn bởi việc triển khai, vị trí địa lý hoặc cấu hình vật lý của các tài nguyên cơ bản.



Virtualization Architecture VM (virtual server)

Công nghệ ảo hóa cho phép tạo ra các thực thể ảo có tính tương đương như các thực thể vật lý, chẳng hạn như thiết bị lưu trữ và bộ vi xử lý. Ảo hóa phần cứng (hardware

virtualization) tạo ra các máy ảo (virtual machines) hoạt động với hệ điều hành được cài đặt như một máy tính thực nghĩa. Bạn có thể tạo nguyên một máy ảo Ubuntu trên máy tính Windows hoặc hệ điều hành khác.

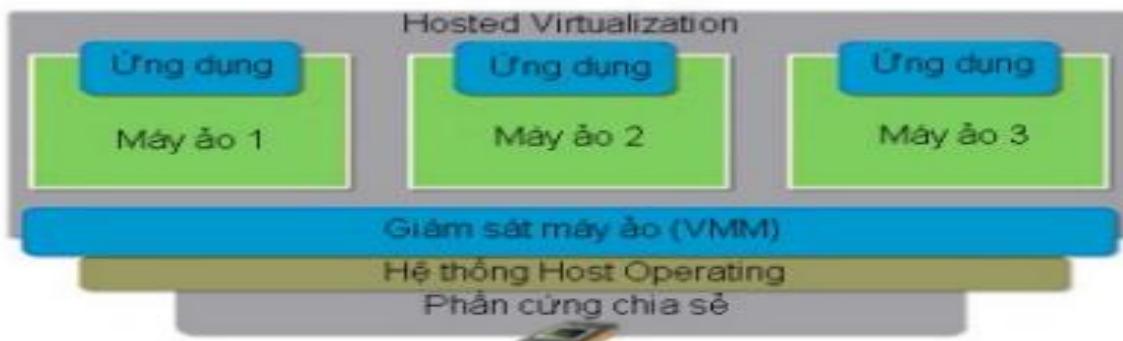
Ảo hóa phần cứng giúp chia nhỏ tài nguyên vật lý để tối ưu hiệu suất sử dụng. Nó cho phép khởi tạo nhiều máy ảo trên một máy chủ vật lý duy nhất. Máy chủ vật lý được gọi là host machine, trong khi máy ảo được gọi là máy khách (guest machine).

Hypervisor hoặc virtual machine manager là phần mềm hay firmware tạo ra các máy ảo.

Ảo hóa hệ thống cho phép phân chia một máy chủ thành nhiều máy chủ ảo hoặc kết hợp nhiều máy chủ vật lý thành một máy chủ logic. Người dùng nhận biết và sử dụng các server ảo như các máy tính vật lý độc lập. Các máy ảo sử dụng tài nguyên được cấp từ máy chủ vật lý.

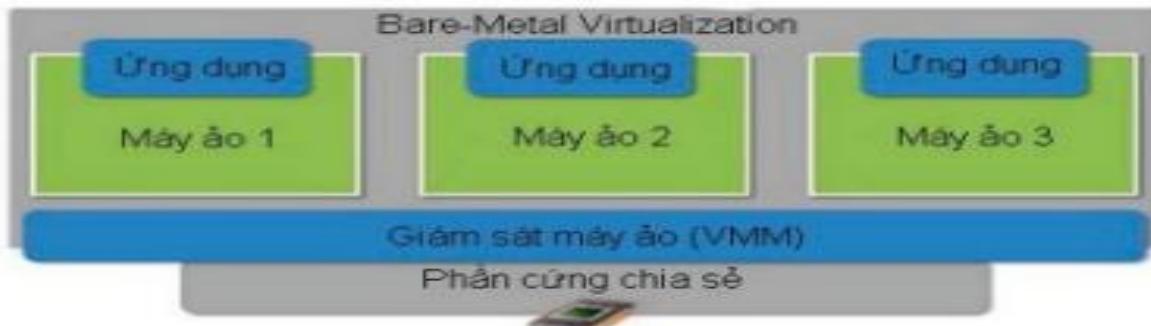
Có ba kiến trúc ảo hóa hệ thống máy chủ chính: Host-based, Hypervisor-based và Hybrid. Sản phẩm ảo hóa được triển khai (VMWare, Microsoft Hyper V, Citrix Xenserver) và mức độ ảo hóa cụ thể sẽ tạo ra các kiến trúc khác nhau.

Kiến trúc ảo hóa Hosted-based còn gọi là hosted hypervisor, kiến trúc này sử dụng một lớp hypervisor chạy trên nền tảng hệ điều hành, sử dụng các dịch vụ được hệ điều hành cung cấp để phân chia tài nguyên tới các máy ảo. Nếu ta xem hypervisor là một phần mềm riêng biệt, thì các hệ điều hành khách của máy ảo sẽ nằm trên lớp thứ ba so với phần cứng máy chủ.



Hình 1.4. Mô hình ảo hóa Hosted-based

Kiến trúc ảo hóa Hypervisor-based



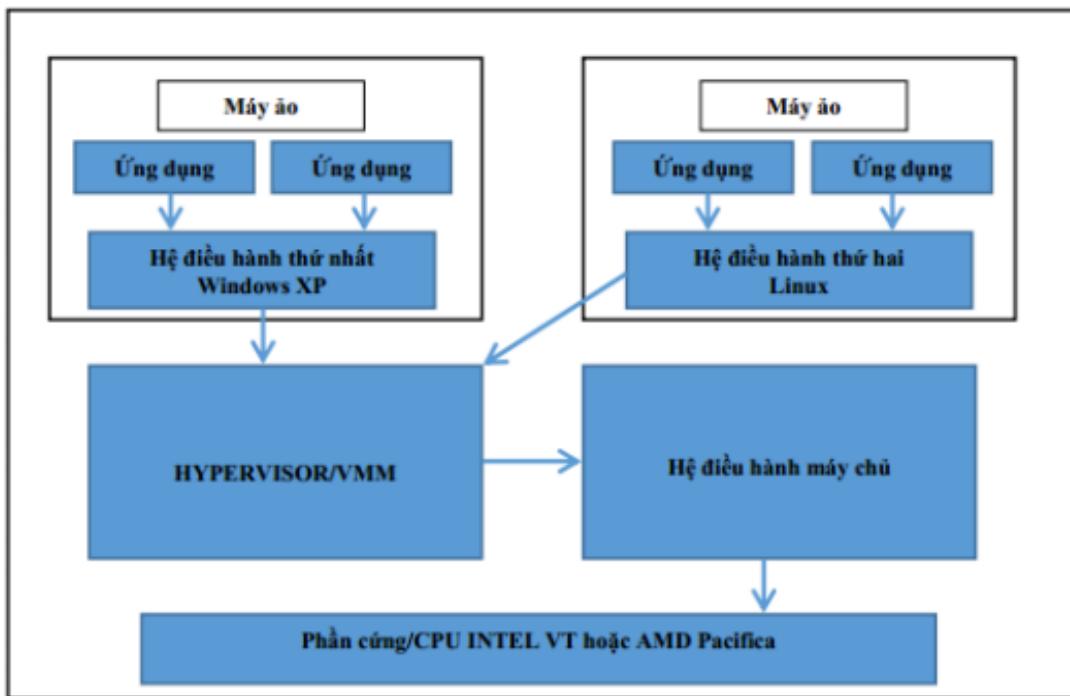
Hình 1.5. Kiến trúc Hypervisor-based

Trong mô hình này, ta thấy lớp phần mềm hypervisor chạy trực tiếp trên nền tảng phần cứng của máy chủ, không thông qua bất kỳ một hệ điều hành hay một nền tảng nào khác. Qua đó, các hypervisor này có khả năng điều khiển, kiểm soát phần cứng của máy chủ. Đồng thời nó cũng có khả năng quản lý các hệ điều hành chạy trên nó. Một hệ thống ảo hóa máy chủ sử dụng nền tảng Bare – Mental hypervisor bao gồm ba lớp chính:

- Nền tảng phần cứng: bao gồm các thiết bị nhập xuất, thiết bị lưu trữ (Hdd, Ram), bộ vi xử lý CPU và các thiết bị khác (các thiết bị mạng, vi xử lý đồ họa, âm thanh...).
- Lớp nền tảng ảo hóa: Virtual Machine Monitor thực hiện việc liên lạc trực tiếp với nền tảng phần cứng phía dưới, quản lý và phân phối tài nguyên cho các hệ điều hành khác nằm trên nó.
- Các ứng dụng máy ảo: các máy ảo này sẽ lấy tài nguyên từ phần cứng, thông qua sự cấp phát và quản lý của hypervisor.

Kiến trúc lai Hybrid

Hybrid là một kiểu ảo hóa mới hơn vì có nhiều ưu điểm. Trong đó lớp ảo hóa hypervisor chạy song song với hệ điều hành máy chủ. Tuy nhiên, trong cấu trúc ảo hóa này, các máy chủ ảo vẫn phải đi qua hệ điều hành máy chủ để truy cập phần cứng nhưng khác biệt ở chỗ cả hệ điều hành máy chủ và các máy chủ ảo đều chạy trong chế độ hạt nhân.



Hình 1.6. Kiến trúc ảo hóa Hybrid

Công nghệ tự động hóa giám sát điều phối tài nguyên (automation, dynamic orchestration)

Công nghệ giám sát điều phối tài nguyên động là nền tảng để điện toán đám mây thực hiện cam kết chất lượng cung cấp dịch vụ điện toán. Với công nghệ điều phối tài nguyên động, việc lắp đặt thêm hay giảm bớt các tài nguyên máy chủ vật lý hoặc máy chủ lưu trữ dữ liệu được thực hiện tự động để hệ thống điện toán luôn đáp ứng được giao kèo trong hợp đồng dịch vụ đã ký với bên người sử dụng.

Công nghệ tính toán phân tán, hệ phân tán

Điện toán đám mây là một dạng hệ phân tán xuất phát từ yêu cầu cung ứng dịch vụ cho lượng người sử dụng. Tài nguyên tính toán của điện toán đám mây là hình thức chia sẻ trị tổng thể kết hợp của hạ tầng mạng và hàng nghìn máy chủ vật lý phân tán trên một hay nhiều trung tâm dữ liệu số (data centers).

Công nghệ Web 2.0

Web 2.0 là nền tảng công nghệ phát triển các sản phẩm ứng dụng hướng dịch vụ trên nền điện toán đám mây. Công nghệ Web 2.0 phát triển cho phép phát triển giao diện ứng dụng web dễ dàng và nhanh chóng và trên nhiều thiết bị giao diện khác nhau. Web 2.0 phát triển làm xóa đi khoảng cách về thiết kế giao diện giữa ứng dụng máy tính thông thường và ứng dụng trên nền web, cho phép chuyển hóa ứng dụng qua dịch vụ trên nền điện toán đám mây mà không ảnh hưởng đến thói quen người sử dụng.

1.2 Ưu điểm của điện toán đám mây

+Đây là một phiên bản rút gọn của ưu điểm của điện toán đám mây:

+Thay chi phí đầu tư bằng chi phí linh động: Trả tiền chỉ khi sử dụng tài nguyên điện toán thực tế, không cần đầu tư mạnh vào phần cứng trước.

+Lợi ích từ tính kinh tế cao theo quy mô: Sử dụng điện toán đám mây giúp giảm chi phí và mang lại lợi ích kinh tế cao hơn nhờ quy mô hàng trăm ngàn khách hàng.

+Không cần ước tính năng lực: Không cần dự đoán nhu cầu năng lực cơ sở hạ tầng, có thể tăng hoặc giảm quy mô theo yêu cầu chỉ trong vài phút.

+Tăng tốc độ và tính linh hoạt: Cung cấp tài nguyên tính toán theo nhu cầu thực tế, giảm thời gian để có tài nguyên và tăng tính linh hoạt cho tổ chức.

+Dùng chi tiền vào việc duy trì trung tâm dữ liệu: Tập trung vào các dự án quan trọng, không phải vào việc vận hành cơ sở hạ tầng máy chủ.

+Phát triển toàn cầu trong vài phút: Triển khai ứng dụng trên nhiều khu vực trên thế giới chỉ với vài cú nhấp chuột, giúp cung cấp trải nghiệm tốt hơn với độ trễ thấp hơn cho khách hàng và tiết kiệm chi phí.

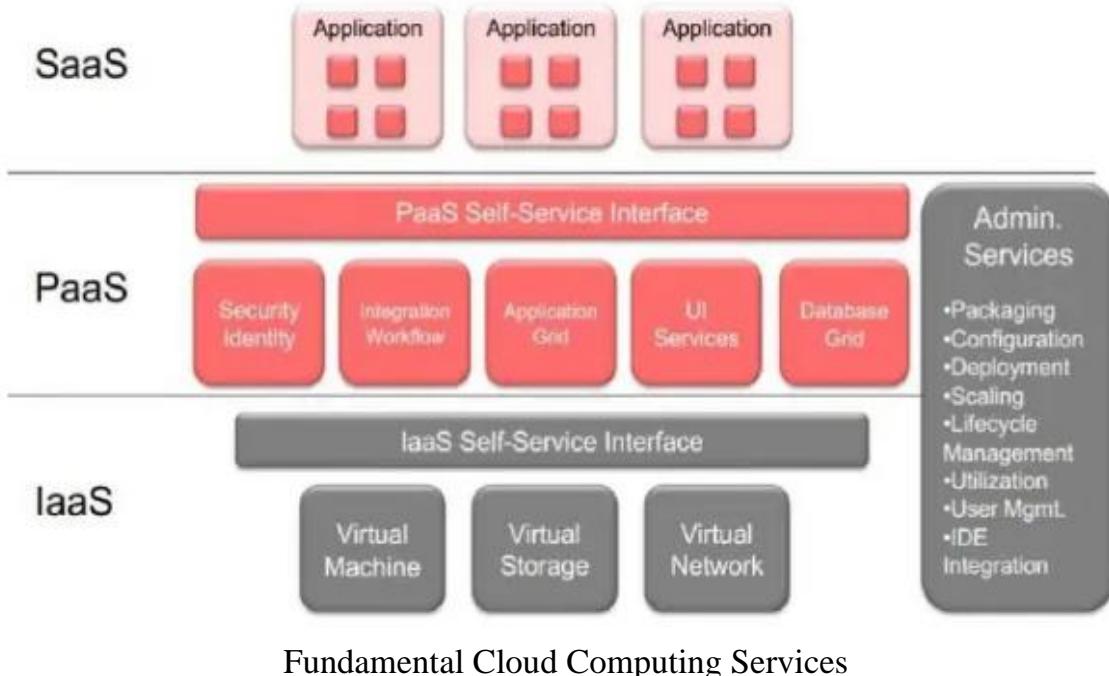
+Bảo mật: Mặc dù điện toán đám mây đã được cải thiện đáng kể về bảo mật thông qua công nghệ tiên tiến, vẫn còn rào cản về an ninh dữ liệu. Tuy nhiên, các nhà cung cấp đám mây có khả năng và nguồn lực để đảm bảo tính an toàn cho dữ liệu và giải quyết các vấn đề bảo mật.

+Quản lý quyền điều khiển dữ liệu: Mỗi quan tâm về việc mất quyền điều khiển dữ liệu nhạy cảm ngày càng tăng cao, đặc biệt là đối với các doanh nghiệp yêu cầu mức độ bảo mật thông tin cao.

2. Các mô hình dịch vụ điện toán đám mây.

Các nhà cung cấp dịch vụ điện toán đám mây cung cấp các dịch vụ của họ theo ba mô hình cơ bản:

- + Dịch vụ dành cho Cơ sở hạ tầng (IaaS – Infrastructure as a Service).
- + Dịch vụ dành cho Cơ sở nền tảng (PaaS – Platform as a Service).
- + Dịch vụ dành cho Phần mềm (SaaS – Software as a Service).



2.1. IaaS – Infrastructure as a Service.

Cơ sở hạ tầng dưới dạng dịch vụ (IaaS) trong điện toán đám mây là một mô hình cho phép người dùng thuê tài nguyên điện toán từ nhà cung cấp dịch vụ đám mây. Nhà cung cấp IaaS cung cấp bộ lưu trữ, mạng, máy chủ và ảo hóa, giúp người dùng không cần phải quản lý và duy trì trung tâm dữ liệu tại chỗ. Người dùng có thể lưu trữ tài nguyên này trong đám mây công cộng (chia sẻ tài nguyên với người dùng khác), đám mây riêng (không chia sẻ tài nguyên) hoặc đám mây lai (kết hợp cả hai). IaaS cung cấp các API cao cấp cho phép người dùng quản lý các chi tiết cấp thấp của cơ sở hạ tầng mạng, bao gồm sao lưu, phân vùng dữ liệu, mở rộng quy mô và bảo mật.

Đặc điểm của IaaS bao gồm cung cấp tài nguyên như máy chủ, mạng, bộ nhớ, CPU, không gian đĩa cứng và trang thiết bị trung tâm dữ liệu. Nó cũng cho phép khả năng mở rộng linh hoạt và thay đổi chi phí tùy theo nhu cầu thực tế. Người dùng có thể chia sẻ tài nguyên với nhiều người thuê khác trên cùng một hạ tầng. IaaS cung cấp lợi ích doanh nghiệp bằng cách tổng hợp tài nguyên tính toán cho công ty.

Một số nhà cung cấp IaaS nổi tiếng bao gồm AWS (Amazon Web Services) và Azure của Microsoft. Các nhà cung cấp này cung cấp các tài nguyên mạng, máy tính và lưu trữ cho khách hàng. Việc triển khai và quản lý máy chủ được thực hiện thông qua giao diện của nhà cung cấp, giúp người dùng tiết kiệm thời gian và công sức.

IaaS là một mô hình phổ biến trong điện toán đám mây, đặc biệt phù hợp cho những người muốn triển khai ứng dụng của mình và không muốn lo lắng về việc duy trì cơ sở hạ tầng phần cứng.

2.2. PaaS– Platform as a Service.

IaaS tập trung vào cung cấp cơ sở hạ tầng tính toán, lưu trữ và kết nối mạng. Trong khi đó, PaaS là một mức độ cao hơn so với IaaS. Với dịch vụ IaaS, bạn được cung cấp một máy chủ, nhưng bạn phải tự cài đặt nhiều thành phần như Web Server, Database, tùy thuộc vào ứng dụng web của bạn. Điều này có thể khá phức tạp và tốn thời gian.

Để giảm bớt công việc cài đặt và triển khai ứng dụng web, bạn có thể sử dụng dịch vụ PaaS. Với PaaS, bạn được cung cấp một nền tảng đã được cài đặt sẵn và phù hợp cho ứng dụng của bạn. Một số nhà cung cấp PaaS không cho phép sở hữu máy chủ riêng, thay vào đó, bạn sẽ sử dụng chung một máy chủ với người dùng khác và cơ sở dữ liệu của bạn cũng được đặt trong môi trường lưu trữ chung. Điều này giúp giảm chi phí, nhưng đồng thời cũng mang lại rủi ro về bảo mật và truy cập dữ liệu.

Dịch vụ PaaS phổ biến cho phép phát triển ứng dụng trên nhiều nền tảng và ngôn ngữ như .NET (Microsoft Windows Azure), Java, Python, Ruby. Tuy nhiên, việc hỗ trợ các ngôn ngữ, công cụ phát triển và API có thể không thống nhất.

Các đặc điểm tiêu biểu của PaaS bao gồm:

- Phục vụ cho việc phát triển, kiểm thử, triển khai và vận hành ứng dụng giống như một môi trường phát triển tích hợp.
- Cung cấp các công cụ khởi tạo với giao diện trên web.
- Tích hợp dịch vụ web và cơ sở dữ liệu.
- Hỗ trợ cộng tác nhóm phát triển.

Một nhà cung cấp PaaS nổi bật là Red Hat OpenShift, một phần mềm chạy dịch vụ mã nguồn mở có sẵn trên GitHub với tên "OpenShift Origin". OpenShift hỗ trợ phát triển ứng dụng bằng nhiều ngôn ngữ khác nhau và cung cấp tính tùy biến cao. Nó cũng hỗ trợ bảo trì dịch vụ và thông kê ứng dụng nếu cần thiết.

2.3. SaaS– Software as a Service.

Software as a Service (SaaS) là sự lựa chọn phù hợp nhất khi muốn tập trung vào người dùng cuối. Cho phép truy cập các phần mềm trên nền tảng đám mây mà không cần quản lý cơ sở hạ tầng và nền tảng đó.

Các dịch vụ SaaS đã xuất hiện từ lâu, ví dụ như các dịch vụ thư điện tử như Hotmail, Yahoo Mail, Gmail. Hiện nay, các dịch vụ SaaS cho doanh nghiệp đang phát triển nhiều hơn, bao gồm Office 365 của Microsoft cho ứng dụng văn phòng, Salesforce CRM cho quản lý khách hàng, Amazon cho thương mại điện tử.

SaaS mang lại nhiều lợi ích cho tổ chức và doanh nghiệp. Chi phí được trả theo mức độ sử dụng hàng tuần hoặc hàng tháng, giúp giảm thiểu chi phí ban đầu. Việc dùng thử và lựa chọn phần mềm phù hợp giúp tiết kiệm chi phí.

Đặc trưng tiêu biểu của SaaS bao gồm:

Truy xuất và quản lý phần mềm qua mạng.

- Quản lý từ một vị trí tập trung, cho phép truy xuất từ xa qua web.
- Cung cấp ứng dụng thông tin thường xuyên với nhiều mô hình ánh xạ và tính năng nâng cao.
- Tích hợp các phần mềm giao tiếp trên mạng diện rộng.
- Một nhà cung cấp SaaS nổi bật là Salesforce, với bộ giải pháp CRM toàn diện trên nền tảng điện toán đám mây. Salesforce được đánh giá cao nhờ tính năng vượt trội.

Cloud Computing có 5 đặc tính:

- Broad network access: Truy cập dịch vụ cloud từ bất kỳ thiết bị nào có kết nối mạng.
- On-demand self-service: Tự cấu hình dịch vụ theo nhu cầu mà không cần sự can thiệp từ nhà cung cấp.
- Resource pooling: Điều phối và chia sẻ linh hoạt các tài nguyên trên dịch vụ cloud.
- Rapid elasticity: Cung cấp tài nguyên nhanh chóng và linh hoạt theo nhu cầu sử dụng.
- Measured service: Giám sát và đo lường tài nguyên trên dịch vụ cloud.

3. Các mô hình triển khai điện toán đám mây.

- Điện toán đám mây riêng tư (Private cloud).
- Điện toán đám mây công cộng (Public cloud).
- Điện toán đám mây chung (Community cloud).
- Điện toán đám mây lai (Hybrid cloud).

3.1. Điện toán đám mây riêng tư (Private cloud)

Điện toán đám mây riêng (Private Cloud) là mô hình trong đó hạ tầng đám mây được sở hữu bởi một tổ chức phục vụ cho người dùng của tổ chức đó. Private Cloud có thể được vận hành bởi một bên thứ ba và hạ tầng đám mây có thể được đặt bên trong hoặc bên ngoài tổ chức sở hữu (tại bên thứ ba kiêm vận hành hoặc thậm chí là một bên thứ tư).

Private Cloud được các tổ chức, doanh nghiệp lớn xây dựng cho mình nhằm khai thác ưu điểm về công nghệ và khả năng quản trị của công nghệ đám mây. Với Private Cloud, các doanh nghiệp tối ưu được hạ tầng tài nguyên của mình, nâng cao được hiệu quả sử dụng, quản lý trong cấp phát và thu hồi tài nguyên, qua đó giảm thời gian đưa sản phẩm sản xuất, kinh doanh ra thị trường.

3.2 Điện toán đám mây công cộng (Public cloud)

Là mô hình mà hạ tầng của điện toán đám mây được một tổ chức sở hữu và cung cấp dịch vụ rộng rãi cho tất cả các khách hàng thông qua hạ tầng mạng Internet hoặc các mạng công cộng điện rộng. Các ứng dụng khác nhau chia sẻ chung tài nguyên tính toán, mạng lưu trữ. Do đó hạ tầng của công nghệ này được thiết kế để đảm bảo cô lập về dữ liệu giữa các dịch vụ Public Cloud hướng tới số lượng khách hàng lớn nên thường có năng lực về hạ tầng cao, đáp ứng nhu cầu tính toán linh hoạt, đem lại chi phí thấp cho khách hàng. Do đó khách hàng của dịch vụ trên Public Cloud sẽ bao gồm tất cả các tầng lớp mà khách hàng cá nhân và doanh nghiệp nhỏ sẽ được lợi thế trong việc dễ dàng tiếp cận các ứng dụng công nghệ cao, chất lượng mà không phải đầu ban đầu, chi phí sử dụng thấp, linh hoạt giữa các khách hàng và tách biệt về truy cập.

3.3 Điện toán đám mây chung (Community cloud)

Đám mây chung là mô hình trong đó hạ tầng đám mây được chia sẻ bởi một số tổ chức cho cộng đồng người dùng trong các tổ chức đó. Các tổ chức này do đặc thù không tiếp cận với các dịch vụ Public Cloud và chia sẻ chung một hạ tầng đám mây để nâng cao hiệu quả đầu tư và sử dụng.

3.4 Điện toán đám mây lai (Hybrid cloud)

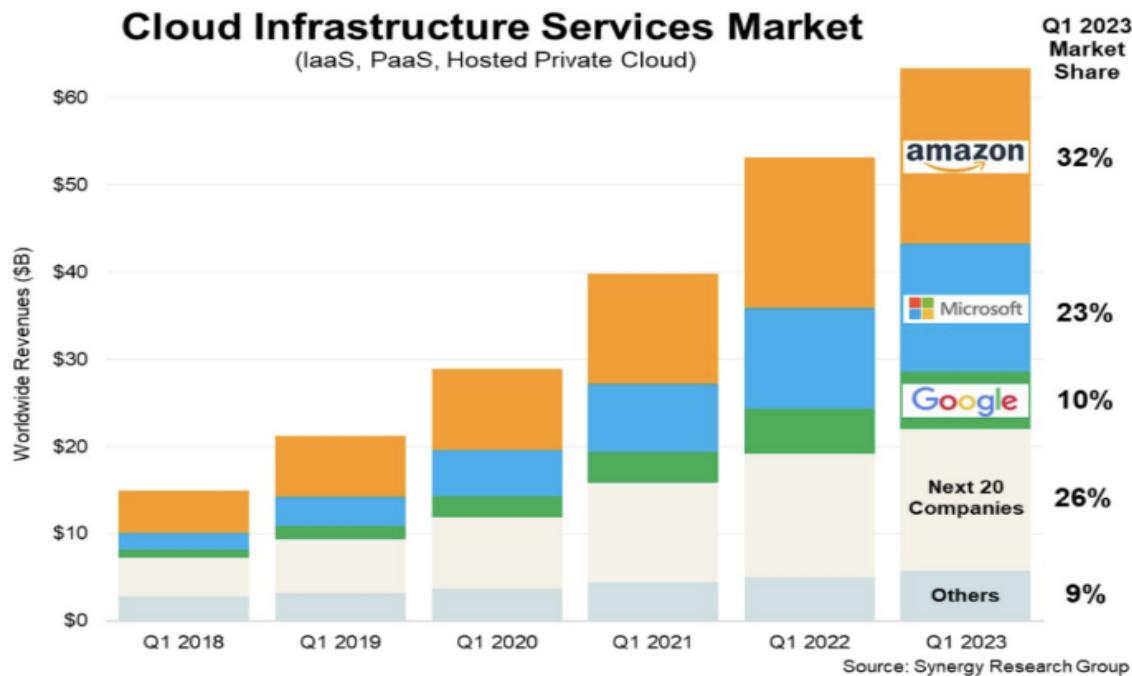
Mô hình đám mây lai là mô hình bao gồm hai hoặc nhiều hơn các đám mây trên tích hợp với nhau. Mô hình Hybrid Cloud cho phép chia sẻ hạ tầng hoặc đáp ứng nhu cầu trao đổi dữ liệu.

Chương II: ĐIỆN TOÁN ĐÁM MÂY AWS.

1. Giới thiệu nhanh về AWS.

Amazon web services là một nền tảng điện toán đám mây phát triển toàn diện được cung cấp bởi Amazon.com hiện là nền tảng bán lẻ trực tuyến hàng đầu thế giới, bán hàng triệu mặt hàng cho người tiêu dùng trên khắp Châu Mỹ, Châu Âu và một số khu vực của Châu Á (Amazon bổ sung dịch vụ siêu máy tính cho đám mây của mình vào năm 2011, tr.8). AWS được hình thành như một phương tiện cung cấp dịch vụ cho các trang web và người tiêu dùng khác dựa trên kinh nghiệm dày dặn sẵn có của công ty trong việc xây dựng, quản lý và vận hành cơ sở hạ tầng kỹ thuật số của riêng mình. Dịch vụ Web đôi khi được gọi là dịch vụ đám mây hoặc các dịch vụ điện toán từ xa. Các dịch vụ AWS đầu tiên đã được đưa ra vào năm 2006 để cung cấp các dịch vụ trực tuyến cho các trang web và các ứng dụng phía máy khách. Để giảm thiểu việc bị mất điện đột ngột và đảm bảo tính mạnh mẽ của hệ thống, AWS đa dạng về địa lý theo khu vực. Đám mây AWS trải rộng trên 99 Vùng sẵn sàng tại 31 khu vực địa lý trên khắp thế giới và đã công bố kế hoạch tăng thêm 15 Vùng sẵn sàng và 5 Khu vực AWS khác tại Canada, Israel, Malaysia, New Zealand và Thái Lan. Mỗi khu vực bao gồm nhiều khu vực địa lý nhỏ hơn được gọi là vùng sẵn có. Nó cung cấp cho doanh nghiệp những giải pháp về storage, computing power, database hay developer tools,...

Hiện nay, AWS đã có hơn 200 dịch vụ trên nền tảng của họ. Sử dụng AWS bạn sẽ có thể sử dụng một trung tâm dữ liệu với công nghệ điện toán đám mây ở bất cứ đâu trên toàn thế giới. Bởi với AWS thì đám mây không chỉ có cụm trong phạm vi của một quốc gia nữa. Hiện tại thông qua rất nhiều báo cáo chúng ta có thể thấy được rằng ba ông lớn gồm Amazon, Microsoft và Google vẫn đang dẫn đầu trong thị trường điện toán đám mây. Riêng Amazon đã đạt được mức thị phần cao kỷ lục 33% trong 4 năm liên tiếp. Trong đó AWS Việt Nam cũng là một dịch vụ được nhiều cá nhân và tổ chức sử dụng.



Như vậy, trên là những thông tin về Amazon Web Server – một dịch vụ điện toán đám mây lớn nhất hiện nay và hiện đang dẫn đầu trong môi trường dịch vụ doanh nghiệp trên thị trường.

1.1 Giải pháp theo trường hợp sử dụng

Amazon Web Services đưa ra giải pháp lưu trữ web trên đám mây trang bị cho các doanh nghiệp, tổ chức phi lợi nhuận một cách phân phối trang web và ứng dụng web với mức phí thấp. Cho dù bạn đang tìm kiếm một trang web tiếp thị, đa phương tiện hay thương mại điện tử, AWS sẽ cung cấp cho bạn hàng loạt các tùy chọn lưu trữ trang web để bạn có thể chọn được một tùy chọn phù hợp với mình.

Lý do bạn nên sử dụng AWS:

Hỗ trợ bất kỳ CMS nào bạn muốn bao gồm WordPress, Drupal, Joomla, v.v. AWS cũng hỗ trợ và cung cấp SDK cho các nền tảng phổ biến như Java, Ruby, PHP, Node.js và .Net.

Trung tâm dữ liệu trên toàn thế giới, khách hàng của bạn có thể ở khắp nơi trên thế giới. Với AWS, bạn có thể có một trung tâm dữ liệu hoặc CDN lưu trữ trang web của mình ở bất kỳ vị trí địa lý nào bạn chọn chỉ bằng một vài thao tác tối ưu đơn giản.

Có thể điều chỉnh quy mô ngay từ ngày đầu tiên, lưu lượng truy cập trang web có thể biến đổi rất nhiều. Từ thời điểm ít lượng truy cập cho đến thời điểm lượng truy cập tăng đột biến.

Mô hình định giá linh động, AWS chỉ tính phí đối với các tài nguyên bạn sử dụng, không yêu cầu trả phí trước hoặc hợp đồng dài hạn. Có thể tùy chọn trả ngay hoặc mức giá cố định hàng tháng.

Khuyến khích học tập trên nền tảng AWS với chính sách thu hút người dùng học trên AWS với các chương trình miễn phí tăng tính trải nghiệm thực tế tối thiểu cho người dùng (https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all) hỗ trợ hầu hết các dịch vụ giúp cho người học có thể thỏa sức học tập trên AWS về cách triển khai mà không lo tốn phí ngoại trừ một số dịch vụ liên quan đến tài nguyên tối thiểu của Amazon như IPv4 ($255^4 \sim 4.223$ tỷ địa chỉ IP trong khi dân số thế giới 7,888 tỷ => không cân xứng), cổng NAT,... nên khi tạo cũng như thực hành xong thì điều kiện tiên quyết là nó destroy những dịch vụ đó.

Chi phí vận hành và bảo trì tương đối ổn định theo đánh giá (dùng bao nhiêu trả bấy nhiêu). Sao lưu và khôi phục dễ dàng với AWS Backup, độ bền dữ liệu cao với tỉ lệ gần như tuyệt đối với các bản sao. Hỗ trợ nhiều dịch vụ bảo mật cho tài nguyên Network một cách hiệu quả nhất và đồng thời cũng là nơi cung cấp dịch vụ cho nhiều những đối tác lớn như Netflix, NASA, Disney, Nasdaq, Guardian, DBS, Robinhood,....

1.2 Giới thiệu các dịch vụ của Amazon Web Services Cloud

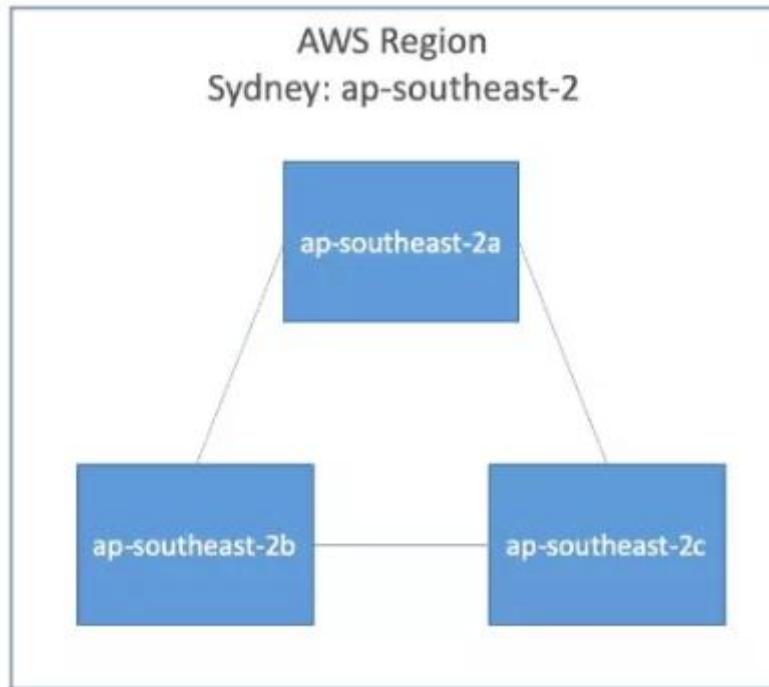
Amazon Web Services (AWS) cung cấp hàng loạt sản phẩm dựa trên nền tảng đám mây trên phạm vi toàn cầu, bao gồm điện toán, lưu trữ, sao lưu phục hồi, cơ sở dữ liệu, phân tích, network, di động, công cụ nhà phát triển, công cụ quản lý, IoT, bảo mật và ứng dụng doanh nghiệp. AWS có hơn 200 dịch vụ phục vụ cho nhiều ngành và công nghệ khác nhau.

Khi sử dụng AWS, ta cần hiểu về khái niệm Regions và Availability Zone (AZs). AWS có nhiều trung tâm dữ liệu trên toàn cầu, được gọi là Regions. Mỗi Region có tên riêng như us-east-1, ap-southeast-1. Các Region này chứa nhiều trung tâm dữ liệu (data center), được gọi là Availability Zone (AZs). Dưới đây là hình ảnh về Regions

United States	
N. Virginia	us-east-1
Ohio	us-east-2
N. California	us-west-1
Oregon	us-west-2
Asia Pacific	
Mumbai	ap-south-1
Osaka	ap-northeast-3
Seoul	ap-northeast-2
Singapore	ap-southeast-1
Sydney	ap-southeast-2
Tokyo	ap-northeast-1
Canada	
Central	ca-central-1
Europe	
Frankfurt	eu-central-1
Ireland	eu-west-1
London	eu-west-2
Paris	eu-west-3
Stockholm	eu-north-1
South America	
São Paulo	sa-east-1

Các dịch vụ của AWS được triển khai trong các Regions, ví dụ như region R1 và region R2. Khi bạn chuyển từ một region sang region khác, dữ liệu của dịch vụ A sẽ không được đồng bộ tự động và bạn phải khởi tạo lại dịch vụ A ở region mới(Làm mới hoàn toàn).

Tiếp theo, có khái niệm Availability Zones (AZs), là các vùng khả dụng trong mỗi Region. Mỗi Region có thể có nhiều AZs, thường là từ 2 đến 6. Ví dụ, Region Sydney (ap-southeast-2) có 3 AZs: ap-southeast-2a, ap-southeast-2b, ap-southeast-2c. Mỗi AZ là một hoặc nhiều trung tâm dữ liệu riêng biệt, với nguồn điện dự phòng và hệ thống mạng kết nối mạnh mẽ. Các AZ hoàn toàn tách biệt địa lý nhau, nhưng vẫn được kết nối với nhau thông qua một hệ thống mạng để đảm bảo đồng bộ dữ liệu.



1.2.1. dịch vụ tính toán (Compute).

- + Amazon Elastic Compute Cloud (EC2) là một dịch vụ IaaS cung cấp các máy chủ ảo có thể điều khiển bằng API, dựa trên siêu giám sát Xen. Các Dịch vụ từ xa tương đương bao gồm Microsoft Azure, Google Compute Engine và Rackspace; và các sản phẩm tương đương tại chỗ như OpenStack hoặc Eucalyptus.
- + Amazon Elastic Beanstalk cung cấp dịch vụ PaaS để lưu trữ các ứng dụng, các dịch vụ tương đương bao gồm Google App Engine hoặc Heroku hoặc OpenShift để sử dụng tại chỗ.
- + Amazon Lambda (AWS Lambda) chạy mã để phản hồi các sự kiện bên trong hoặc bên ngoài AWS (serverless), chẳng hạn như yêu cầu http, cung cấp tài nguyên được yêu cầu một cách minh bạch được tích hợp chặt chẽ với AWS nhưng các dịch vụ tương tự như Google Cloud Functions và các giải pháp mở như OpenWhisk đang trở thành đối thủ cạnh tranh.

1.2.2. Kết nối mạng (Networking & Content Delivery).

- + Amazon Route 53 cung cấp dịch vụ DNS được quản lý có thể mở rộng cung cấp Dịch vụ tên miền đồng thời aws cũng hỗ trợ bán tên miền.
- + Amazon Virtual Private Cloud (VPC) tạo ra một tập hợp các tài nguyên AWS được cô lập một cách hợp lý có thể được kết nối bằng cách sử dụng kết nối VPN. Điều này cạnh tranh với các giải pháp tại chỗ như OpenStack hoặc HPE Helion Eucalyptus được sử dụng cùng với phần mềm PaaS.

- + AWS Direct Connect cung cấp các kết nối mạng chuyên dụng vào trung tâm dữ liệu AWS.
- + Amazon Elastic Load Balancing (ELB) tự động phân phối lưu lượng truy cập đến trên nhiều phiên bản Amazon EC2. Trong đó 3 lựa chọn default sẽ là Application Load Balancer (ALB), Network Load Balancer (NLB) và Classic Load Balancer (CLB). Trong khi đó thì Network Load Balancer và Classic Load Balancer được gọi chung là Elastic Load Balancer (ELB).
- + Amazon API Gateway là một dịch vụ để xuất bản, duy trì và bảo mật các API dịch vụ web
- + Amazon CloudFront , một mạng phân phối nội dung (CDN) để phân phối các đối tượng đến cái gọi là "vị trí biên" gần yêu cầu

1.2.3. Cơ sở dữ liệu (Database)

- + Amazon DynamoDB cung cấp Dịch vụ cơ sở dữ liệu trực tuyến NoSQL có độ trễ thấp, có thể mở rộng được hỗ trợ bởi SSD.
- + Amazon ElastiCache cung cấp bộ nhớ đệm trong bộ nhớ cho các ứng dụng web, đây là cách Amazon triển khai Memcached và Redis.
- + Amazon Relational Database Service (RDS) cung cấp các máy chủ cơ sở dữ liệu có thể mở rộng với hỗ trợ MySQL, Oracle, SQL Server và PostgreSQL
- + Amazon Redshift cung cấp kho dữ liệu quy mô petabyte với tính năng lưu trữ dựa trên cột và tính toán nhiều nút.
- + Amazon SimpleDB cho phép các nhà phát triển chạy các truy vấn trên dữ liệu có cấu trúc. Nó hoạt động cùng với EC2 và S3.

1.2.4. Ban quản lý (Management & Governance)

- + Amazon CloudWatch cung cấp khả năng giám sát các ứng dụng và tài nguyên đám mây AWS, bắt đầu với EC2.
- + Amazon Identity and Access Management (IAM) là một dịch vụ ngầm, cung cấp cơ sở hạ tầng xác thực được sử dụng để xác thực quyền truy cập vào các dịch vụ khác nhau.

1.2.5 Thùng (Container)

- + Amazon ECS là một dịch vụ điều phối bộ chúa được quản lý hoàn toàn, giúp bạn dễ dàng triển khai, quản lý và thay đổi quy mô các ứng dụng có trong bộ chúa.

- + Sổ đăng ký bộ chúa linh hoạt của Amazon (Amazon ECR) là sổ đăng ký bộ chúa được quản lý hoàn toàn, cung cấp dịch vụ lưu trữ hiệu suất cao để bạn có thể triển khai thành phần lõi và hình ảnh ứng dụng ở bất kỳ đâu một cách đáng tin cậy.
- + Amazon (Amazon EKS) là một dịch vụ Kubernetes được quản lý để chạy Kubernetes trên đám mây AWS và trong những trung tâm dữ liệu tại chỗ. Khi ở trên đám mây, Amazon EKS tự động quản lý mức độ sẵn sàng và khả năng điều chỉnh quy mô của các nút trên mặt phẳng điều khiển Kubernetes chịu trách nhiệm lên lịch trình cho bộ chúa, quản lý mức độ sẵn sàng của các ứng dụng, lưu trữ dữ liệu cụm cũng như những tác vụ chính khác. Với Amazon EKS, bạn có thể tận dụng tất cả những lợi thế về hiệu năng, quy mô, độ tin cậy và mức độ sẵn sàng của cơ sở hạ tầng AWS, cũng như các tiện ích tích hợp với những dịch vụ kết nối mạng và bảo mật của AWS. Khi ở những cơ sở tại chỗ, EKS cung cấp một giải pháp Kubernetes nhất quán, được hỗ trợ đầy đủ với công cụ tích hợp và cách thức triển khai đơn giản sang AWS Outposts, máy ảo hoặc máy chủ bare metal.

1.2.6 Lưu trữ phân phối nội dung (Storage).

- + Amazon Simple Storage Service (S3) là một dịch vụ lưu trữ đối tượng cung cấp khả năng thay đổi quy mô, mức độ sẵn sàng của dữ liệu, độ bảo mật và hiệu suất hàng đầu trong ngành. Khách hàng thuộc mọi quy mô và ngành nghề có thể lưu trữ và bảo vệ dữ liệu thuộc mọi kích thước cho hầu hết tất cả các trường hợp sử dụng, chẳng hạn như hồ dữ liệu, ứng dụng hoạt động trên đám mây và ứng dụng di động. Với các lớp lưu trữ tiết kiệm chi phí và tính năng quản lý dễ sử dụng, bạn có thể tối ưu hóa chi phí, tổ chức dữ liệu và cấu hình các biện pháp kiểm soát quyền truy cập được tinh chỉnh để đáp ứng yêu cầu cụ thể về kinh doanh, tổ chức và tuân thủ.
- + Amazon Glacier được xây dựng nhằm mục đích lưu trữ dữ liệu, mang lại hiệu năng cao nhất, khả năng truy xuất linh hoạt nhất và không gian lưu trữ với chi phí thấp nhất trên đám mây. Tất cả các lớp lưu trữ của S3 Glacier đều cung cấp khả năng mở rộng hầu như không giới hạn và được thiết kế đảm bảo độ bền dữ liệu 99,999999999% (11 chữ số 9). Các lớp lưu trữ của S3 Glacier cung cấp các tùy chọn để truy cập nhanh nhất vào dữ liệu lưu trữ của bạn và không gian lưu trữ với chi phí thấp nhất trên đám mây

1.2.7 Các dịch vụ khác (Other)

- + Management Tools: Các công cụ quản lý.
- + Developer Tools: Các Công cụ phát triển.

- + Analysis: Phân tích.
- + Customer Engagement: Cam kết khách hàng.
- + Application Integration: Tích hợp ứng dụng.
- + Business Productivity: Năng suất nghiệp vụ.
- + Công nghệ thực tế ảo (AR & VR).
- + Machine Learning: Học máy.
- + Big Data: Xử lý dữ liệu lớn.
- + Desktop & App Streaming: Ứng dụng máy tính và Stream.

Chương III: NHỮNG DỊCH VỤ CỦA AWS

1. Những dịch vụ phổ biến của AWS.

1.1 Amazon EC2 là gì?

Amazon Elastic Compute Cloud (Amazon EC2) là một dịch vụ cung cấp cơ sở hạ tầng điện toán đám mây bởi Amazon Web Services (AWS), cho phép bạn thuê và quản lý các máy chủ ảo linh hoạt theo yêu cầu. Amazon EC2 cung cấp các máy chủ ảo có khả năng mở rộng và tính linh hoạt cao trong việc xử lý, cùng với các thành phần phần cứng ảo như bộ nhớ RAM, vi xử lý và các phân vùng lưu trữ dữ liệu khác nhau. Nó cung cấp một kiến trúc đám mây mạnh mẽ để quản lý dịch vụ một cách an toàn.

Với Amazon EC2, bạn có thể triển khai một hoặc nhiều máy chủ ảo để tạo ra các ứng dụng nhanh chóng và đảm bảo tính sẵn sàng cao. Bạn cũng có thể linh hoạt chỉnh sửa các tài nguyên máy chủ dựa trên nhu cầu sử dụng của bạn.

Mỗi EC2 Instance được tính phí dựa trên thời gian sử dụng theo giờ và loại Instance được chọn. AWS cung cấp nhiều loại Instance khác nhau để phù hợp với yêu cầu kinh doanh của người dùng.

Đối với việc triển khai một dự án web demo, thì cấu hình tối thiểu đề xuất cho EC2 Instance là (16 vCPU, 32GB) để đảm bảo hoạt động ổn định với mức truy vấn không quá 2000 query/s. Nếu có nhu cầu xử lý truy vấn lớn hơn, bạn có thể thực hiện một mô hình khác như sử dụng FIFO message queue (ví dụ như Kafka) để chuyển sang hình thức bất đồng bộ và tránh quá tải.

Việc lựa chọn cấu hình EC2 Instance phù hợp và áp dụng các biện pháp như message queue là rất quan trọng để đảm bảo hiệu suất và ổn định cho dự án web demo của bạn trên AWS.

2. AWS RDS

2.1 Amazon RDS là gì?

Amazon RDS(Amazon Relational Database Service) là dịch vụ đám mây do Amazon Web Services phát triển với mục tiêu cung cấp giải pháp cài đặt, vận hành và mở rộng dành cho relational database (cơ sở dữ liệu có quan hệ).

Nó cung cấp khả năng tiết kiệm chi phí hiệu quả và thay đổi kích thước, nó có thể tự động hóa các nhiệm vụ quản lý tồn tại nhiều thời gian như dự phòng phần cứng, thiết lập cơ sở dữ liệu, vá lỗi và sao lưu.

Amazon RDS cung cấp hiệu năng nhanh, tính sẵn sàng cao, tính bảo mật và khả năng tương thích. Các tính năng của AWS RDS:

- Replication(nhân rộng) RDS sử dụng tính năng Sao chép để tạo bản sao chỉ có quyền đọc. Đây là những bản sao chỉ đọc của phiên bản cơ sở dữ liệu mà các ứng dụng sử dụng mà không làm thay đổi cơ sở dữ liệu sản xuất ban đầu. Quản trị viên cũng có thể kích hoạt chuyển đổi dự phòng tự động trên nhiều vùng sẵn sàng thông qua triển khai RDS Multi-AZ và sao chép dữ liệu đồng bộ.
- Storage(kho chứa) RDS cung cấp ba loại lưu trữ:
 - General-purpose solid-state drive (SSD). Amazon khuyến nghị lưu trữ này là lựa chọn mặc định.
 - Provisioned input-output operations per second (IOPS). Lưu trữ SSD cho khối lượng công việc nặng về I/O.
 - Magnetic. Một lựa chọn chi phí thấp hơn.
- Monitoring(giám sát) Dịch vụ Amazon CloudWatch cho phép giám sát được quản lý. Nó cho phép người dùng xem dung lượng và số liệu I/O.
- Patching(vá) RDS cung cấp các bản vá cho bất kỳ công cụ cơ sở dữ liệu nào mà người dùng chọn.
- Backups(sao lưu) Một tính năng khác là phát hiện và phục hồi lỗi. RDS cung cấp các bản sao lưu phiên bản được quản lý với nhật ký giao dịch để cho phép khôi phục tại thời điểm. Người dùng chọn khoảng thời gian lưu giữ và khôi phục cơ sở dữ liệu về bất kỳ thời điểm nào trong khoảng thời gian đó. Họ cũng có thể chụp nhanh các phiên bản tồn tại theo cách thủ công cho đến khi chúng bị xóa theo cách thủ công.
- Incremental billing(thanh toán gia tăng) Người dùng trả phí hàng tháng cho các phiên bản họ khởi chạy.
- Encryption(mã hóa) RDS sử dụng mã hóa khóa chung để đảm bảo các bản sao lưu tự động, bản sao đọc, ảnh chụp nhanh dữ liệu và các dữ liệu khác được lưu trữ ở trạng thái nghỉ.

Các hệ quản trị cơ sở dữ liệu quan hệ mà Amazon RDS hỗ trợ hiện nay gồm có: SQL Server, Oracle (yêu cầu bản quyền).

MySQL, PostgreSQL, MariaDB (mã nguồn mở).

Amazon Aurora.

Giúp bạn dễ dàng thiết lập, vận hành và thay đổi quy mô cơ sở dữ liệu quan hệ trên đám mây.

Dịch vụ này cho phép bạn tập trung vào các ứng dụng của mình nhằm giúp ứng dụng có hiệu suất, tính sẵn sàng, mức độ bảo mật cũng như khả năng tương thích cao như mong đợi.

3. Amazon S3

3.1 Amazon S3 là gì?

Amazon Simple Storage Service (Amazon S3) là một dịch vụ lưu trữ đối tượng cung cấp khả năng mở rộng, tính sẵn có của dữ liệu, bảo mật và hiệu suất hàng đầu trong ngành. Khách hàng thuộc mọi quy mô và ngành nghề có thể sử dụng Amazon S3 để lưu trữ và bảo vệ mọi lượng dữ liệu cho nhiều trường hợp sử dụng, chẳng hạn như kho dữ liệu, trang web, ứng dụng di động, sao lưu và khôi phục, lưu trữ, ứng dụng doanh nghiệp, thiết bị IoT và dữ liệu lớn phân tích. Amazon S3 cung cấp các tính năng quản lý để bạn có thể tối ưu hóa, sắp xếp và định cấu hình quyền truy cập vào dữ liệu của mình nhằm đáp ứng các yêu cầu tuân thủ, tổ chức và kinh doanh cụ thể của bạn.

S3 an toàn vì AWS cung cấp:

Mã hóa dữ liệu mà bạn lưu trữ. Nó có thể xảy ra theo 2 cách:

- + Mã hóa ở client side
- + Mã hóa ở server side

Nhiều bản copy được duy trì để cho phép phục hồi dữ liệu trong trường hợp dữ liệu bị hỏng

Versioning, trong đó mỗi bản chỉnh sửa đều được lưu trữ để sử dụng khi cần thiết.

S3 bền vì:

Nó thường xuyên xác minh tính toàn vẹn của dữ liệu bằng cách sử dụng checksum: nếu phát hiện có bất kỳ trực trặc dữ liệu nó sẽ sửa chữa ngay lập tức với sự trợ giúp của các dữ liệu được sao chép

Ngay cả trong khi dữ liệu lưu trữ hoặc truy xuất dữ liệu, nó sẽ kiểm tra lưu lượng mạng đến cho bất kỳ gói dữ liệu bị hỏng nào

S3 có khả năng mở rộng cao vì nó tự động tăng dung lượng lưu trữ của bạn theo yêu cầu và bạn chỉ trả tiền cho bộ nhớ bạn sử dụng.

Vậy Loại dữ liệu như thế nào có thể lưu trữ trên S3:

Bạn có thể lưu trữ bất kỳ loại data với bất kỳ loại format nào. Trên S3 khi chúng ta nói về dung lượng, số lượng của đối tượng mà chúng ta có thể lưu trữ trên S3 là không giới hạn. Một đối tượng là thực thể cơ bản trong S3. Nó bao gồm dữ liệu, khóa và siêu dữ liệu (metadata)

Khi ta nói về dữ liệu, có 2 loại:

Dữ liệu truy cập thường xuyên

Dữ liệu truy cập không thường xuyên

Vì vậy AMAZON đưa ra 3 lớp lưu trữ cung cấp cho khách hàng trải nghiệm tốt nhất với chi phí hợp lý

- Amazon S3 Standard để truy cập dữ liệu thường xuyên

Phù hợp với các trường hợp sử dụng nhạy cảm với hiệu suất, nơi độ trễ phải được giữ ở mức thấp. Ví dụ như trong bệnh viện, dữ liệu thường xuyên truy cập sẽ là dữ liệu của bệnh nhân được nhập viện, dữ liệu này cần được truy xuất nhanh chóng.

- Amazon S3 Standard để truy cập dữ liệu không thường xuyên

Phù hợp với trường hợp sử dụng nơi dữ liệu được lưu trữ lâu dài và ít được truy cập thường xuyên, tức là vẫn lưu trữ dữ liệu nhưng vẫn mong đợi hiệu suất cao. Ví dụ trong bệnh viện, những người đã được xuất viện, hồ sơ của họ sẽ không cần truy cập hàng ngày nhưng nếu họ quay trở lại với bất kỳ biến chứng nào, thì hồ sơ của họ cần được truy xuất lấy ra một cách nhanh chóng.

- Amazon Glacier phù hợp với các trường hợp sử dụng nơi dữ liệu được lưu trữ hiệu suất cao là điều không cần thiết, nó có chi phí thấp hơn so với 2 dịch vụ trên. Ví dụ: Trong bệnh viện các báo cáo test, Scan docs, các đơn thuốc vv...vv hơn 1 năm sẽ không cần thiết sử dụng hàng ngày, ngay cả khi được yêu cầu độ trễ thấp là điều không cần thiết

Dữ liệu trên S3 tổ chức dưới dạng Bucket:

- Một Bucket là một đơn vị lưu trữ logic trong S3
- Chứa các đối tượng bao gồm dữ liệu và siêu dữ liệu

Trước khi thêm bất kỳ dữ liệu nào lên S3 chúng ta phải tạo 1 Bucket nơi sẽ được sử dụng để lưu trữ dữ liệu

Dữ liệu được lưu trữ ở đâu về mặt địa lý

Bạn có thể tự chọn nơi dữ liệu nên được lưu trữ. Đưa ra quyết định khu vực (region) là rất quan trọng do đó chúng phải được lên kế hoạch tốt. Có 4 chỉ số để chọn 1 vùng lưu trữ tối ưu đó là:

Pricing (Giá)

User/Customer Location (Nơi khách hàng sử dụng dịch vụ)

Latency (Độ trễ)

Service Availability (Tính khả dụng dịch vụ) Cùng tìm hiểu qua ví dụ cụ thể sau đây: Giả sử có một công ty phải khởi chạy dịch vụ lưu trữ cho 1 trang web khách hàng ở Hoa Kỳ và Ấn Độ. Để cung cấp trải nghiệm tốt nhất, công ty phải chọn khu vực phù hợp với yêu cầu

4. Amazon LAMBDA

4.1 Amazon Lambda là gì?

AWS Lambda là dịch vụ điện toán hướng sự kiện(event), không cần máy chủ, cho phép bạn chạy mã cho hầu như mọi loại ứng dụng hoặc dịch vụ phụ trợ mà không cần cung cấp hoặc quản lý máy chủ, không trả tiền cho những tài nguyên không được sử dụng, nghĩa là khác với EC2 mô hình hoạt động máy chủ nếu EC2 còn ở trạng thái running dù không xử lý 1 tác vụ request/response (không truy xuất) nào thì bạn vẫn phải trả tiền để duy trì dịch vụ EC2 đó còn ở Lambda bạn chỉ trả tiền khi Lambda xử lý nhiệm vụ được đề ra. Bạn có thể kích hoạt Lambda từ hơn 200 dịch vụ AWS và ứng dụng phần mềm dưới dạng dịch vụ (SaaS) và chỉ trả tiền cho những gì bạn sử dụng.

Cách sử dụng:

Chạy mã mà không cần cung cấp hoặc quản lý cơ sở hạ tầng. Chỉ cần viết và tải mã lên dưới dạng tệp .zip hoặc hình ảnh và lưu trữ.

Tự động phản hồi các yêu cầu thực thi mã ở mọi quy mô, từ hàng tá sự kiện mỗi ngày đến hàng trăm nghìn sự kiện mỗi giây.

Tiết kiệm chi phí bằng cách chỉ trả tiền cho thời gian tính toán mà bạn sử dụng—theo mili giây—thay vì cung cấp trước cơ sở hạ tầng cho công suất cao nhất.

Tối ưu hóa thời gian thực thi mã và hiệu suất với kích thước bộ nhớ chức năng phù hợp. Đáp ứng nhu cầu cao trong mili giây hai chữ số với Đồng thời được cung cấp.

Trường hợp sử dụng:

- Xử lý dữ liệu ở quy mô lớn
- Chạy web tương tác và phụ trợ di động
- Kích hoạt thông tin chuyên sâu về ML mạnh mẽ
- Tạo các ứng dụng hướng sự kiện

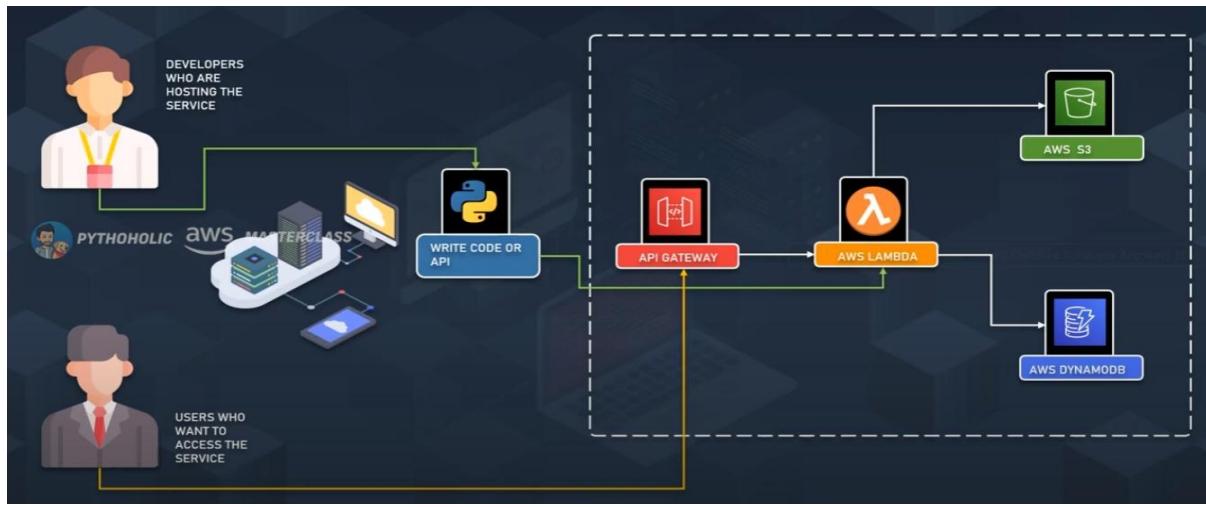
4.2 Quy trình thiết lập

Giả sử bạn có một trường hợp yêu cầu xây dựng một ứng dụng hướng sự kiện đơn giản không cần cầu kỳ về mặc tính toán quá nhiều về chi phí, cấu hình, thời gian setup hệ thống thì việc sử dụng AWS Serverless Lambda là một yếu tố được cân nhắc đến trước tiên.

Quá trình xây dựng một phiên bản với EC2(Serverful):



Cài đặt web server(apache,nginx,...), networking, setup các gói cho server, security cho network, load balancer, auto scaling. Nhưng đối với Lambda chúng ta sẽ không cần quản lý các yếu tố trên nữa (mặc dù vẫn là những server nhưng AWS đã làm giúp) mà thay vào đó chỉ cần chú tâm vào mã nguồn vào sản phẩm, phát triển phần mềm xây dựng những quy trình khác (giảm đi quy trình vận hành và cài đặt).



Hiện tại AWS Lambda vốn hỗ trợ mã Java, Go, PowerShell, Node.js, C#, Python và Ruby, đồng thời cung cấp API Thời gian chạy cho phép bạn sử dụng bất kỳ ngôn ngữ lập trình bổ sung nào để biên soạn các chức năng của mình. Bạn chỉ cần tạo một Lambda function, một cổng API Gateway, một bộ chứa S3 và một trình AWS database

4.3 AWS API Gateway

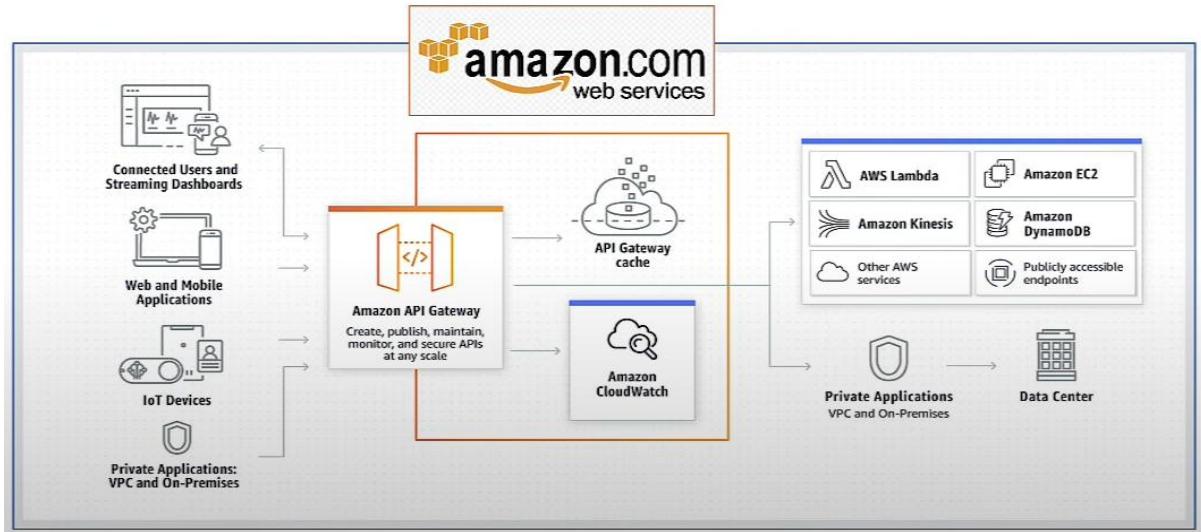
Amazon API Gateway đóng vai trò là "cửa trước" cho các ứng dụng để truy cập dữ liệu, logic nghiệp vụ hoặc chức năng từ các dịch vụ backend của bạn và API Gateway cũng được xếp vào một trong những dịch vụ Serverless. Bằng cách sử dụng API Gateway, bạn có thể tạo các API RESTful và API WebSocket để kích hoạt các ứng dụng giao tiếp hai chiều theo thời gian thực. API Gateway hỗ trợ các khối lượng công việc có trong container và serverless, cũng như các ứng dụng web. Đối với những dịch vụ web nói chung thì gateway đóng một vai trò chủ chốt không thể thiếu vì nó là nơi điều phối tài nguyên giữa các dịch vụ, một điểm giao tiếp chung giữa khách hàng và dịch vụ.

API Gateway của Amazon hỗ trợ 2 loại API:

- HTTP/REST : HTTP APIs cho phép bạn tạo các API RESTful với độ trễ thấp hơn và chi phí thấp hơn các API REST. Nếu API của bạn yêu cầu cả chức năng proxy API lẫn tính năng quản lý API trong cùng một giải pháp thì API Gateway cũng cung cấp cả API REST.
- WEBSOCKET: Thích hợp đối với các ứng dụng hai chiều theo thời gian thực, tiêu biểu là ứng dụng chat trực tuyến.

Một số đặc tính nổi bật mà dịch vụ này có thể đem lại là:

- Luôn có sẵn để nhận requests
- Có Auto-scale để xử lý hàng trăm nghìn lời gọi API tại một thời điểm.



So sánh với ELB: Đối với API Gateway, chúng ta có thể định nghĩa thông tin vào trong API. Những thông tin này gồm tên path và method.

Ngoài ra thì API Gateway cũng hỗ trợ trích xuất dữ liệu từ:

- Biến trên đường dẫn (param)
- Query String trên URL
- Headers (mô tả gói dữ liệu nhận vd: như phiếu nhận hàng được dán trên bưu kiện hàng)
- Body của request (dữ liệu nhận được vd: hàng hóa trong bưu kiện khi mở kiện hàng ra).

Từ những thông tin này bạn có thể hình thành các lời gọi đến backend. Từ đây bạn có thể xử lý các dữ liệu này trước khi trả về cho client. Đối với dịch vụ serverless thì API Gateway đóng vai trò chủ chốt giống như Lambda. Bạn cũng có thể cấu hình cho AWS API Gateway tùy ý theo yêu cầu dựa vào các tính năng của AWS hỗ trợ mà không phải mất công xây dựng lại.

Ngoài ra để tránh việc API bị quá tải và lạm dụng do quá nhiều request(tốn nhiều tiền), Amazon API Gateway cho phép thắt cổ chai bằng [thuật toán token bucket](#).

Ở đây bạn có thể cài đặt 2 thuộc tính:

- Rate: số request nhận tối đa trong một giây đồng hồ
- Burst: Số request tối đa được xử lý tại một thời điểm

Khi số lượng request vượt quá Rate hoặc Burst hệ thống sẽ trả về lỗi 429 - Too many Requests.

Bạn có thể deploy API trên AWS region, VPC (Virtual Private Cloud), Amazon CloudFront Network. Điều này cho phép bạn sử dụng nó ở rất nhiều cách khác nhau cho cả API nội bộ lẫn truy cập từ bên ngoài.

Ngay khi API Gateway nhận được 1 request, và hệ thống đã thực hiện việc bóc tách dữ liệu từ request đó xong, bây giờ là lúc để gọi đến cách dịch vụ backend như :

- Gửi request này đến 1 function Lambda
- Một EC2 (VPC)
- Containers
- Cách dịch vụ AWS khác
- Một public hoặc private URL (endpoint).

Ngoài ra AWS API Gateway còn cung cấp các tính năng như:

- Monitoring (Giám sát)
- Xác thực và phân quyền (Authentication và Authorization)
- API Key
- Phát triển API

5. AWS Load Balancer

Như đã đề cập ở trước AWS cung cấp 3 loại Elastic Load Balancer

- Classic Load Balancer (Cân bằng tải cổ điển)
Classic ELB hoạt động ở tầng thứ 4 (Network Layer). Tầng 4 đại diện cho lớp vận chuyển, nó được điều khiển bởi các giao thức (protocol) và được sử dụng để truyền yêu cầu. Đối với các ứng dụng web thì nó thường là các giao thức TCP/IP, cung cấp các cân bằng tải cơ bản. Dịch vụ này đã cũ, bị hạn chế về nhiều mặt nên hiện nay ít được sử dụng, khuyến nghị không nên sử dụng.
- Application Load Balancer (Cân bằng tải ứng dụng). ALB Không có địa chỉ IP tĩnh chỉ có đường dẫn URL bù lại ALB tích hợp nhiều tính năng mới nhất của AWS như Host-based Routing, phân phối hoạt động trên layer 7 trong mô hình OSI có nhiệm vụ nhận và phân phối yêu cầu đến các tài nguyên xử lý dựa vào header, URL. Tự scale thêm cho người dùng nếu Load Balancer hiện tại không chịu nổi số lượng request cao vì IP động nên có tính chất này khi scale thì DNS Load Balancer sẽ nhận địa chỉ IP mới.

- Network Load Balancer

Có khả năng chịu tải kết nối trong thời gian dài và chịu tải hàng triệu request trên giây. Có hỗ trợ IP tĩnh.

6. Amazon VPC (Networking)

Amazon Virtual Private Cloud (Amazon VPC) là một dịch vụ quan trọng trong nền tảng AWS, cho phép bạn khởi chạy các tài nguyên AWS trong một mạng ảo cô lập và riêng tư trong AWS Cloud. Với Amazon VPC, bạn có toàn quyền kiểm soát môi trường mạng ảo của mình.

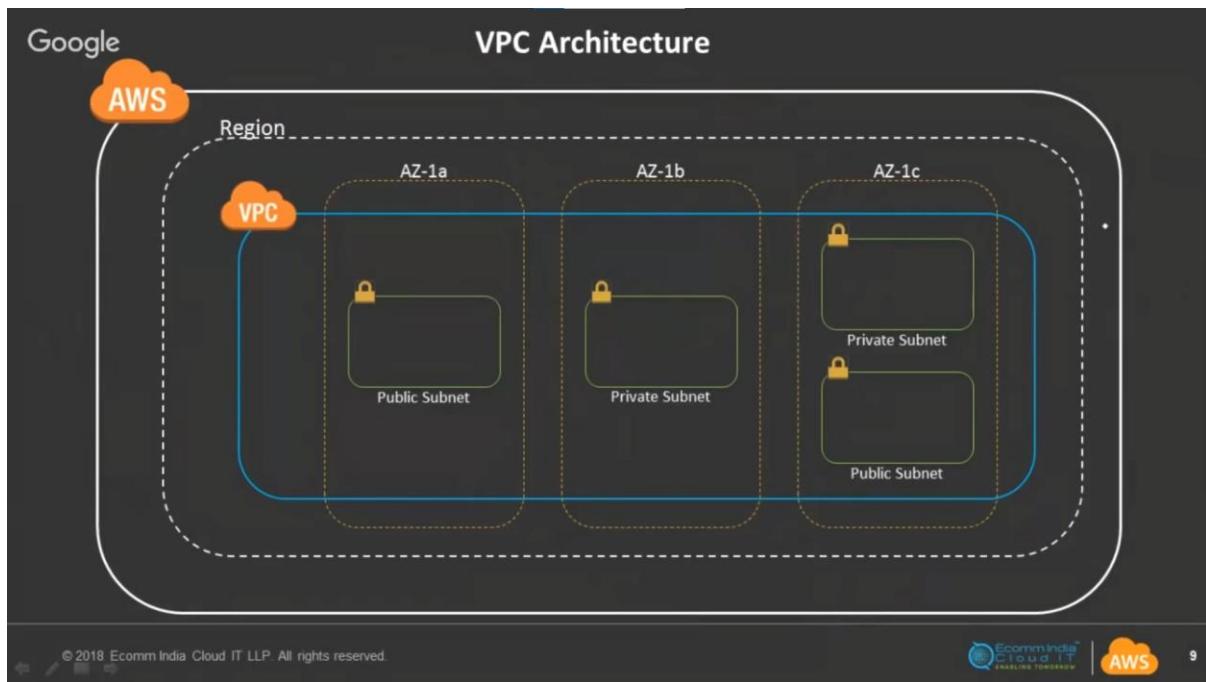
Khái niệm "Đám mây riêng tư ảo" trong tên gọi Amazon VPC mô tả việc tạo ra một mạng độc lập và cô lập mà không cho phép sự truy cập từ bên ngoài mạng này. Nó tương tự như việc triển khai một mạng riêng biệt độc lập trong một trung tâm dữ liệu on-premise. Tại Việt Nam, việc triển khai mạng độc lập như vậy vẫn khá phổ biến.

Amazon VPC cung cấp nhiều tính năng linh hoạt để tùy chỉnh mạng của VPC. Bạn có thể lựa chọn phạm vi địa chỉ IP, tạo subnet và cấu hình bảng định tuyến. Bên cạnh đó, VPC cũng hỗ trợ cả IPv4 và IPv6 để truy cập an toàn và dễ dàng vào các tài nguyên và ứng dụng trong VPC.

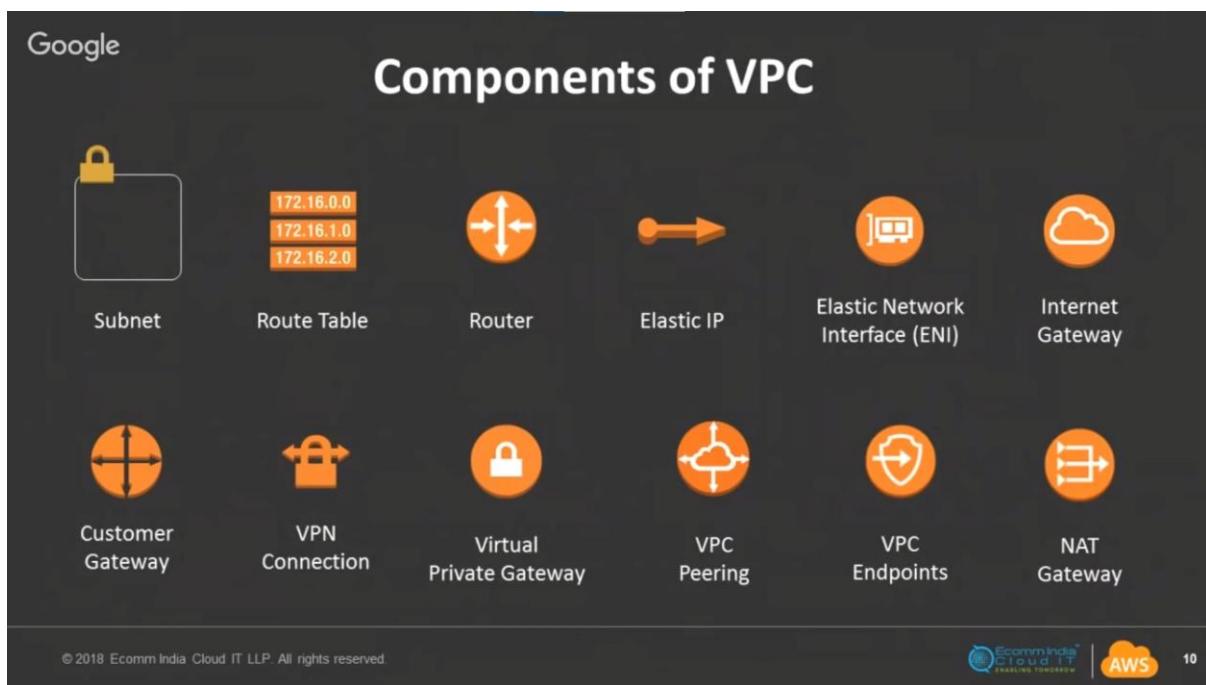
Mỗi Amazon VPC nằm trong một region, đại diện cho một trung tâm dữ liệu lớn của AWS. Trong một region, bạn có thể tạo nhiều VPC và mỗi VPC được phân biệt bởi những dải địa chỉ IP khác nhau. Phạm vi địa chỉ IPv4 của Amazon VPC không thể thay đổi sau khi tạo, và bạn có thể chọn phạm vi từ /16 (khoảng 65,536 địa chỉ khả dụng) đến /28 (khoảng 16 địa chỉ khả dụng).

Dịch vụ Amazon VPC được ra mắt sau dịch vụ Amazon EC2 và hiện chỉ hỗ trợ EC2-VPC. EC2-Classic là nền tảng mạng ban đầu, trong đó tất cả các EC2 instances đều chia sẻ kết nối trong một mạng chung. Tuy nhiên, từ tháng 12 năm 2013, AWS chỉ hỗ trợ EC2-VPC và mỗi region đã có một VPC mặc định được tạo ra với một subnet mặc định có CIDR block là 172.31.0.0/16.

Với Amazon VPC, bạn có thể xây dựng một mạng ảo riêng biệt, cung cấp tính riêng tư và an ninh cho các tài nguyên và ứng dụng của bạn. Việc sử dụng Amazon VPC trong AWS cung cấp cho bạn khả năng tùy chỉnh mạng, quản lý và kiểm soát môi trường mạng ảo của riêng bạn.



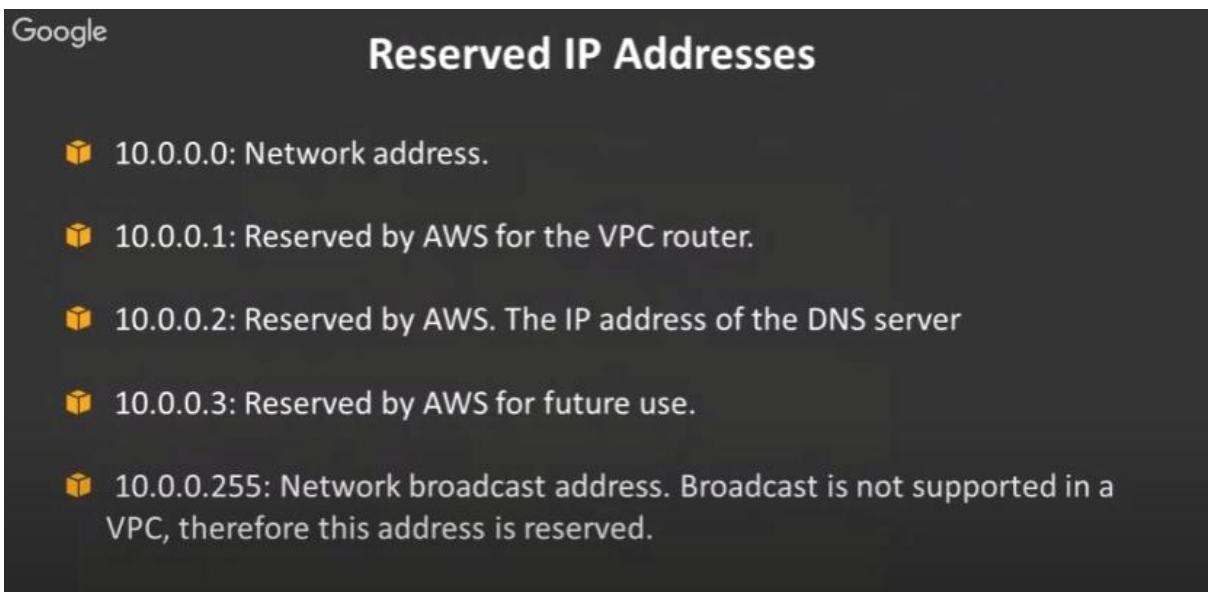
Trong đó các thành phần cốt lõi của AWS VPC bao gồm:



Những thành phần chính của VPC: Subnet, Route Table, Router, Security Group là mặc định của một kiến trúc VPC, Dynamic Host Configuration Protocol (DHCP) option sets. Bảo mật VPC gồm: Security groups, Network Access Control Lists (ACLs) và những thành phần khác là phụ trợ cho VPC tùy vào những mục đích giao tiếp khác nhau sau đây sẽ liệt kê tính năng của từng phần: Internet Gateways (IGWs), Elastic IP (EIP) addresses, Elastic Network Interfaces (ENIs), Endpoints, Peering,

Network Address Translation (NATs) instances and NAT gatewaysVirtual Private Gateway (VPG), Customer Gateways (CGWs), and Virtual Private Networks (VPNs)

- **Subnet:** Mạng con trong VPC là một dải địa chỉ IP. Đây là một phần của VPC có thể chứa các tài nguyên như dịch vụ Amazon EC2 và chia sẻ một thành phần địa chỉ chung. Mạng con công cộng nơi tài nguyên được tiếp xúc với internet thông qua Cổng Internet và Mạng con riêng nơi tài nguyên không được tiếp xúc với thế giới bên ngoài. Mỗi subnet sẽ là một dãy các IP address trừ đi 5 IP cố định và AWS không hỗ trợ broadcast VD:



- **Route Table:** Chúng là một bảng tập hợp các quy tắc được sử dụng để quyết định nơi lưu lượng mạng phải được quản lý. Nó chỉ định đích终极 là địa chỉ IP và mục tiêu. Mục tiêu có thể là cổng Internet, cổng NAT, cổng riêng ảo, v.v. Với việc sử dụng bảng định tuyến, người dùng có thể xác định nơi lưu lượng mạng sẽ được chuyển hướng từ mạng con hoặc cổng tùy cách cấu hình của bạn.
- **Virtual Private Gateway:** Đây là trung tâm VPN (Mạng riêng ảo) ở phía Amazon của kết nối VPN để có giao dịch an toàn. Người dùng có thể đánh kèm nó vào VPC mà họ muốn tạo kết nối VPN từ đó.
- **NAT Gateway:** Cổng dịch địa chỉ mạng (NAT) được sử dụng khi yêu cầu băng thông cao hơn, tính khả dụng với nỗ lực quản lý ít hơn. Nó cập nhật bảng định tuyến của mạng con riêng để gửi lưu lượng đến cổng NAT. Nó chỉ hỗ trợ các giao thức UDP, TCP và ICMP(tốn phí).
- **VPC Peering:** Kết nối ngang hàng VPC cho phép bạn định tuyến lưu lượng giữa hai đám mây riêng ảo bằng cách sử dụng địa chỉ riêng IPv4 hoặc IPv6. Người dùng có thể tạo kết nối ngang hàng VPC giữa VPC của chính họ với

VPC trong tài khoản AWS khác. Kết nối này giúp bạn truyền dữ liệu một cách trơn tru (lưu ý: không có tính bắc cầu giữa các kết nối ngang hàng với nhau).

- **Security Groups:** Nó bao gồm tập hợp các quy tắc tường lửa kiểm soát lưu lượng cho mẫu của bạn. Bạn có thể có một nhóm bảo mật duy nhất được liên kết với nhiều phiên bản (luôn phải chú ý định cấu hình cẩn thận nhóm bảo vệ).
- **Internet Gateway:** Tất cả các subnet public có router kết nối Internet public thì sẽ đi được ra bên ngoài môi trường mạng. Tất cả Instance sẽ kết nối ra môi trường qua igw này.
- Elastic IP: Đây là địa chỉ IP tĩnh, là địa chỉ IP công cộng dành riêng có thể được gán cho bất kỳ Trường hợp nào trong một khu vực cụ thể và không bao giờ thay đổi (tốn phí và nên hủy đi nếu không sử dụng vì là một tài nguyên có hạn).
- Network Access Control Lists (NACL): Đây là lớp bảo mật tùy chọn cho VPC của bạn hoạt động như một tường lửa để kiểm soát lưu lượng truy cập vào và ra khỏi một hoặc nhiều mạng con. Nó bổ sung thêm một lớp bảo mật cho VPC của bạn.
- Customer Gateway: Kết nối VPN liên kết mạng (hoặc dữ liệu) của bạn với Amazon VPC (đám mây riêng ảo). Khách hàng là người trình bày về phía bạn của kết nối đó. Nó có thể là một thiết bị vật lý hoặc phần mềm.
- Network Interface: Đó là kết nối giữa mạng riêng và mạng công cộng. Lưu lượng mạng được tự động chuyển sang phiên bản mới nếu bạn di chuyển nó từ phiên bản này sang phiên bản khác.
- VPC Endpoints: Nó cho phép VPC tạo kết nối với các dịch vụ khác của AWS mà không cần sử dụng internet. Chúng có hai loại, Điểm cuối giao thoa và Điểm cuối cổng. Chúng là các thành phần VPC được chia tỷ lệ, dự phòng và có tính sẵn sàng cao.
- IP addressing: Với Địa chỉ IP, bạn có thể chỉ định các VPC và mạng con, địa chỉ IPv4 và địa chỉ IPv6.
- ACLs : A network access control list(ACL) cho phép hoặc từ chối lưu lượng truy cập vào hoặc ra cụ thể ở cấp độ mạng con. Bạn có thể sử dụng ACL mạng mặc định cho VPC của mình hoặc bạn có thể tạo ACL mạng tùy chỉnh cho VPC của mình với các quy tắc tương tự như quy tắc cho security groups của bạn để thêm một lớp bảo mật bổ sung cho VPC của bạn.

Router: Mỗi tuyến đường trong một bảng chỉ định một đích đến và một mục tiêu. Ví dụ: để cho phép mạng con của bạn truy cập internet thông qua cổng internet, hãy thêm tuyến đường sau vào bảng định tuyến mạng con của bạn. Đích đến của tuyến đường là 0.0.0.0/0, đại diện cho tất cả các địa chỉ IPv4. Mục tiêu là cổng internet được gắn vào VPC của bạn.

Điểm đến	Mục tiêu
0.0.0.0/0	<i>igw-id</i>

Các khối CIDR cho IPv4 và IPv6 được xử lý riêng. Ví dụ: một tuyến đường có CIDR đích **0.0.0.0/0** không tự động bao gồm tất cả các địa chỉ IPv6. Bạn phải tạo một tuyến đường có CIDR đích **::/0** cho tất cả các địa chỉ IPv6.

Nếu bạn thường xuyên tham chiếu cùng một nhóm khối CIDR trên các tài nguyên AWS của mình, thì bạn có thể tạo **danh sách tiền tố do khách hàng quản lý** để nhóm chúng lại với nhau. Sau đó, bạn có thể chỉ định danh sách tiền tố làm đích trong mục nhập bảng lô trình của mình.

Mỗi bảng định tuyến chứa một tuyến cục bộ để liên lạc trong VPC. Tuyến đường này được thêm theo mặc định vào tất cả các bảng tuyến đường. Nếu VPC của bạn có nhiều hơn một khối IPv4 CIDR, thì các bảng định tuyến của bạn chứa một tuyến đường cục bộ cho mỗi khối IPv4 CIDR. Nếu bạn đã liên kết khối CIDR IPv6 với VPC của mình, bảng định tuyến của bạn sẽ chứa một tuyến đường cục bộ cho khối CIDR IPv6. Bạn có thể **thay thế hoặc khôi phục** mục tiêu của từng tuyến cục bộ nếu cần.

Quy tắc và cân nhắc

- Bạn có thể thêm một tuyến đường vào các bảng tuyến đường cụ thể hơn tuyến đường cục bộ. Đích đến phải khớp với toàn bộ khối IPv4 hoặc IPv6 CIDR của mạng con trong VPC của bạn. Mục tiêu phải là cổng NAT, giao diện mạng hoặc điểm cuối Bộ cân bằng tải cổng.
- Nếu bảng lô trình của bạn có nhiều tuyến đường, chúng tôi sẽ sử dụng tuyến đường cụ thể nhất phù hợp với lưu lượng truy cập (khớp tiền tố dài nhất) để xác định cách định tuyến lưu lượng truy cập.
- Bạn không thể thêm các tuyến đường vào địa chỉ IPv4 khớp chính xác hoặc tập con của dải sau: 169.254.168.0/22. Phạm vi này nằm trong không gian địa chỉ liên kết cục bộ và được dành riêng cho các dịch vụ AWS sử dụng. Ví dụ: Amazon EC2 sử dụng các địa chỉ trong phạm vi này cho các dịch vụ chỉ có thể

truy cập được từ các phiên bản EC2, chẳng hạn như Dịch vụ siêu dữ liệu phiên bản (IMDS) và máy chủ Amazon DNS. Bạn có thể sử dụng khối CIDR lớn hơn nhưng chòng lặp 169.254.168.0/22, nhưng các gói dành cho địa chỉ trong 169.254.168.0/22 sẽ không được chuyển tiếp.

- Bạn không thể thêm các tuyến đường vào địa chỉ IPv6 khớp chính xác hoặc tập con của dải sau: fd00:ec2::/32. Phạm vi này nằm trong không gian địa chỉ cục bộ duy nhất (ULA) và được dành riêng cho các dịch vụ AWS sử dụng. Ví dụ: Amazon EC2 sử dụng các địa chỉ trong phạm vi này cho các dịch vụ chỉ có thể truy cập được từ các phiên bản EC2, chẳng hạn như Dịch vụ siêu dữ liệu phiên bản (IMDS) và máy chủ Amazon DNS. Bạn có thể sử dụng khối CIDR lớn hơn nhưng chòng lặp fd00:ec2::/32, nhưng các gói dành cho các địa chỉ trong fd00:ec2::/32 sẽ không được chuyển tiếp.
- Bạn có thể thêm các thiết bị hộp trung gian vào đường dẫn định tuyến cho VPC của mình. Để biết thêm thông tin, hãy xem [Định tuyến cho thiết bị hộp trung gian](#).

Ví dụ

Trong ví dụ sau, giả sử rằng VPC có cả khối IPv4 CIDR và khối CIDR IPv6. Lưu lượng IPv4 và IPv6 được xử lý riêng biệt, như thể hiện trong bảng lộ trình sau.

Destination	Target
10.0.0.0/16	Local
2001:db8:1234:1a00::/56	Local
172.31.0.0/16	pcx- 11223344556677889
0.0.0.0/0	igw- 12345678901234567
::/0	eigw- aabcccddee1122334

- Lưu lượng IPv4 được định tuyến trong VPC (10.0.0.0/16) được bao phủ bởi Tuyến cục bộ.
- Lưu lượng IPv6 được định tuyến trong VPC (2001:db8:1234:1a00::/56) được bao phủ bởi Tuyến cục bộ.
- Lộ trình cho 172.31.0.0/16 gửi lưu lượng đến kết nối ngang hàng.
- Tuyến cho tất cả lưu lượng IPv4 (0.0.0.0/0) gửi lưu lượng đến một cổng internet. Do đó, tất cả lưu lượng truy cập IPv4, ngoại trừ lưu lượng truy cập trong VPC và kết nối ngang hàng, đều được định tuyến đến cổng internet.
- Lộ trình cho tất cả lưu lượng IPv6 (::/0) gửi lưu lượng đến một cổng internet chỉ dành cho lối ra. Do đó, tất cả lưu lượng IPv6, ngoại trừ lưu lượng trong VPC, được định tuyến đến cổng internet chỉ dành cho lối ra.

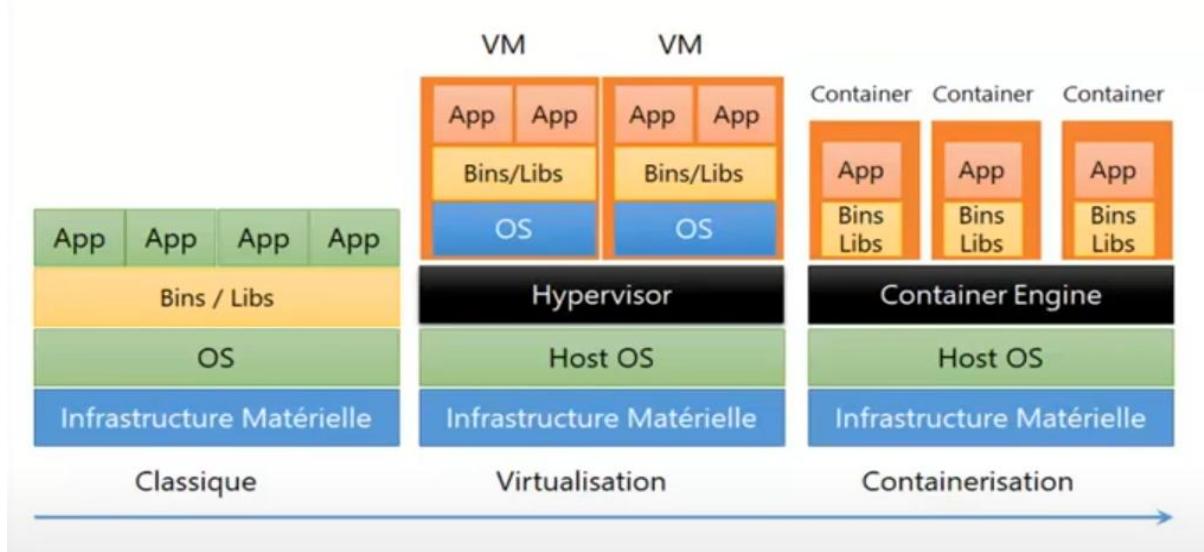
Chung quy lại nó sẽ định tuyến tài nguyên Instance của bạn nằm trong vùng Subnet đến một vùng tài nguyên khác thông qua cấu hình của Route Table và SG.

- **VPC CIDR Block:** Mỗi VPC được liên kết với một dải địa chỉ IP là một phần của Classless Inter-Domain Routing (CIDR) block mà được sử dụng để cấp phát địa chỉ IP private đến EC2 Instances, tất cả VPC mặc định sẽ được liên kết một dải địa chỉ 172.31.0.0/16 với IPv4 CIDR block nếu không khởi tạo. (lưu ý: có thể dùng các tool để tính dải địa chỉ IP cho mỗi subnet thích hợp khẩu trừ đi 5 IP vd: cidr.xyz)

7. Amazon ECS

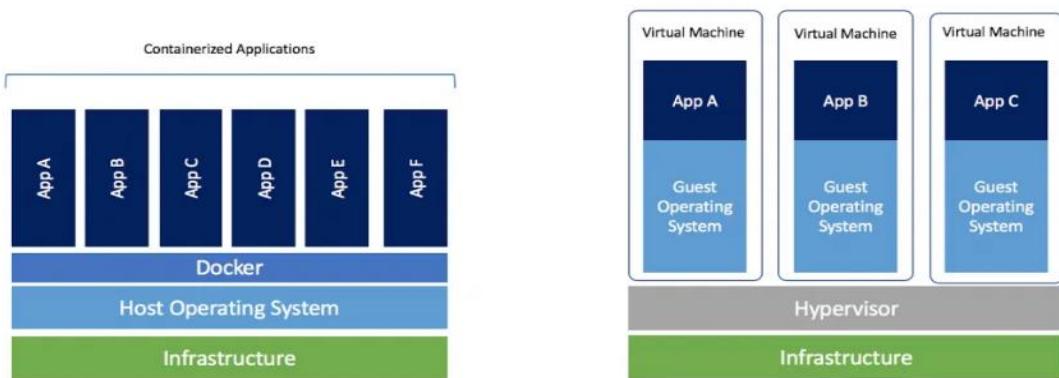
7.1 Docker là gì?

Để hiểu về Amazon ECS, trước hết chúng ta phải hiểu rõ khái niệm về Docker.



Hệ thống kiến trúc hệ thống máy tính qua từng giai đoạn

Docker – một container run time là phần mềm – công cụ cho phép xây dựng, triển khai, đóng gói ứng dụng một cách dễ dàng và nhanh chóng so với kiến trúc ảo hóa trước đây. Container được chạy bởi *Container Run Time* như Docker là một dạng tiêu chuẩn phần mềm đã đóng gói đầy đủ thư viện, mã nguồn cần thiết để chạy ứng dụng với tính ổn định, sẵn sàng cao mà không cần quan tâm đến hạ tầng vật lý. Nhiều container khác nhau có thể được chạy trên một host, miễn là host đó có cài đặt phần mềm Docker.



Compare Docker and Virtualization

Như bạn có thể thấy trong sơ đồ trên, Docker khác hẳn Ảo hóa thông thường. Docker nằm giữa ứng dụng và hệ điều hành máy chủ (OS). Docker có thể chia sẻ hệ điều hành chủ trên nhiều “Container” thay vì yêu cầu mỗi người phải có và chạy hệ điều hành riêng biệt.

Mỗi container có đầy đủ những gì một ứng dụng cần – ví dụ, các phiên bản nhất định của một ngôn ngữ hoặc thư viện – và không nhiều hơn những gì nó cần. Nhiều container có thể được sử dụng cho các phần khác nhau của ứng dụng nếu bạn muốn và chúng có thể được thiết lập để giao tiếp với nhau khi cần.

Điều này cho phép đóng gói ứng dụng của mình thành một mô-đun có thể tái sử dụng, có thể chạy trên bất kỳ máy nào có sẵn. Điều này cho phép phân bổ tài nguyên chi tiết hơn và có thể giảm thiểu lượng CPU hoặc tài nguyên bộ nhớ bị lãng phí.

Bằng cách sử dụng các Docker container được chỉ định để chạy production code, nên chắc chắn rằng môi trường dev và production là như nhau.

Khi số lượng các container mở rộng quá nhanh

Khi ứng dụng của bạn phát triển, việc quản lý công đoạn triển khai, cấu trúc, lập lịch và mở rộng các container phục vụ ứng dụng nhanh chóng trở nên rất phức tạp. Đây là lúc dịch vụ quản lý container (container management service) ra đời. Nó nhằm mục đích cho phép các tùy chọn cấu hình đơn giản và xử lý các công việc không phải chuyên môn trong khi bạn tập trung cho việc lập trình ứng dụng.

7.2 Amazon ECS là gì?

Amazon Elastic Container Service (Amazon ECS), theo định nghĩa của AWS là

Một dịch vụ quản lý container có khả năng mở rộng cao, dễ dàng run, stop, hay quản lý docker container ở trong một cluster. Bạn có thể host một serverless infrastructure bằng cách chạy service hay task sử dụng Fargate launch type hoặc sử dụng EC2 launch type để chạy các EC2 instance.

Amazon ECS được so sánh với Kubernetes, Docker Swarm và Azure Container Instances.

Amazon ECS chạy các containers trong cluster gồm nhiều Amazon EC2 instance được cài sẵn Docker (quản lý một cụm EC2 instance). Dịch vụ này xử lý việc cài đặt container, mở rộng quy mô, giám sát và quản lý những instance này (launch/stop) thông qua cả API và AWS Management Console.

Amazon Elastic Container Service cho phép đơn giản hóa chế độ xem các EC2 instance thành một pool tài nguyên, chẳng hạn như CPU và bộ nhớ.

Bạn có thể sử dụng Amazon ECS để cài đặt container thông qua cluster và dựa vào nguồn tài nguyên mà bạn cần, chính sách độc lập hay khả năng thay đổi. Với Amazon ECS, bạn không phải vận hành hệ thống quản lý cấu hình và quản lý cụm của riêng mình hoặc lo lắng về việc mở rộng cơ sở hạ tầng quản lý của mình.

Amazon ECS là một dịch vụ theo region, nó đơn giản hóa việc chạy ứng dụng containers trên nhiều AZ trong cùng một Region. Bạn có thể tạo một ECS cluster bên trong một VPC mới hoặc cũ. Sau khi một cluster được khởi tạo và chạy, bạn có thể định nghĩa các task và services mà nó chỉ định Docker container image sẽ chạy thông qua clusters.

Thuật ngữ cốt lõi và sơ đồ kiến trúc của Amazon ECS

Ở đây chúng ta đến với hai nhóm thuật ngữ mới:

- Task Definition, Task, và Service.
- ECS Container Instance, ECS Container Agent, và Cluster.

Task Definition

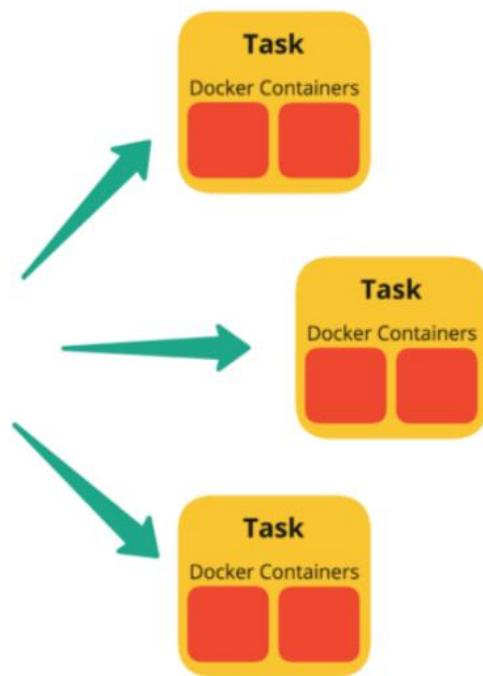
Đây là một text file (json format). Nó sẽ mô tả 1 hoặc nhiều container (lưu ý: tối đa là 10) để hình thành nên ứng dụng của bạn. Task definition sẽ chỉ ra một vài parameter cho ứng dụng, ví dụ như container nào sẽ được dùng, image được sử dụng, launch type sử dụng, cấu hình của container (CPU và bộ nhớ), port đang mở và data volume gì sẽ được tạo với container trong các task...

Parameter trong task definition phụ thuộc vào launch type nào đang được sử dụng. Ví dụ về task definition chứa một container dùng để chạy một NGINX web server sử dụng Fargate launch type.

```

taskDefinition:
  containerDefinitions:
    - name: "my-app"
      image: "xxx.ecr.{region}.amazonaws.com/xxx:{tag}"
      cpu: 300
      memory: 750
      portMappings:
        - containerPort: 5000
          protocol: "tcp"
      links:
        - statsd:statsd
      environment:
        - name: 'NODE_ENV'
          value: '{{ environment }}'
        - name: 'AWS_REGION'
          value: '{{ region }}'
        - name: 'UV_THREADPOOL_SIZE'
          value: '128'
    - name: "statsd"
      image: "xxx.ecr.{region}.amazonaws.com/xxx-statsd:4.5.0"
      cpu: 64
      memory: 64
      environment:
        - name: "TARGET"
          value: '{{ statsd_target }}'
  startupTimeout: 20
  taskRoleArn: {{ task_role_arn }}

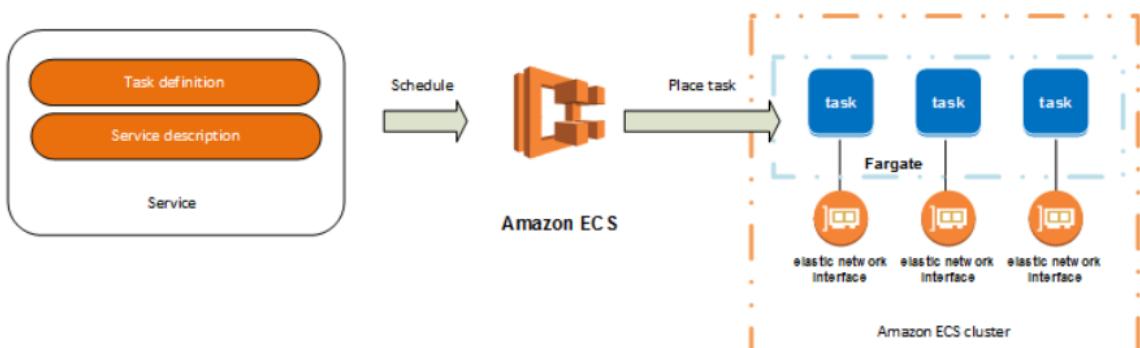
```



Với mỗi một task sử dụng Fargate launch type có một ranh giới riêng biệt và không chia sẻ kernel, tài nguyên cpu, bộ nhớ, hay elastic network interface với task khác.

Amazon ECS task scheduler chịu trách nhiệm cho việc thay thế các task bên trong cluster. Có một vài cách khác nhau để lên schedule cho task

- Service schedule
- Manually running task
- Running task on a cron-like schedule
- Custom scheduler



Service

Xác định mức tối thiểu và tối đa của một hoặc nhiều Task từ một Task Definition chạy tại bất kỳ thời điểm nhất định nào. Đây là tính năng scale và cân bằng tải.

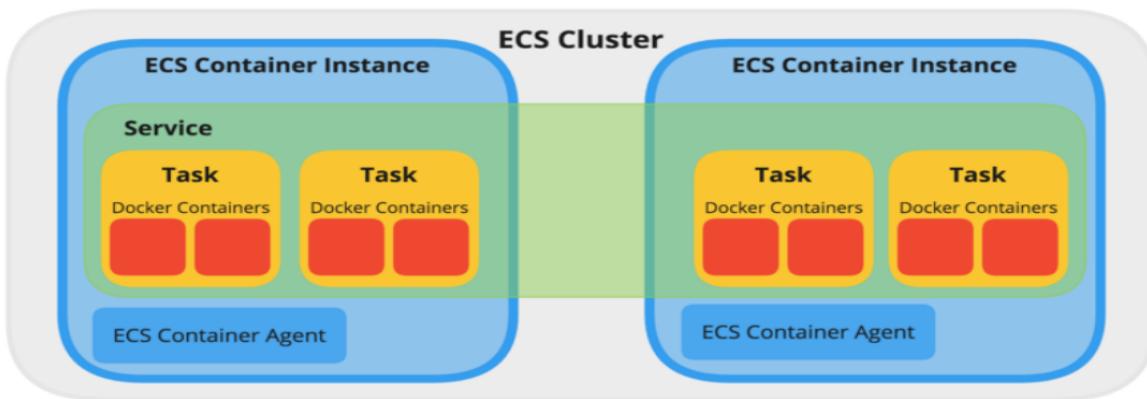
Bây giờ khi đã có Service, các Task của Service cần phải được chạy ở đâu đó để có thể truy cập được. Nó cần được đặt trên một Cluster và dịch vụ quản lý container sẽ xử lý nó khi chạy trên một hoặc nhiều ECS Container Instance(s).

Amazon ECS Container Instances and Amazon ECS Container Agents

Trong hình phía dưới là một EC2 instance đã được cài Docker Engine và một ECS Container Agent bên trong. Một ECS Container Instance có thể chạy nhiều Task giống hoặc khác nhau, từ cùng hoặc khác Services.

Agent được sử dụng để hỗ trợ kết nối trao đổi giữa ECS và các instance, cung cấp thông tin về các container đang chạy và quản lý các container mới tạo...

Cluster



Như đã thấy ở trên, Cluster là một nhóm các ECS Container Instance. Amazon ECS xử lý logic của việc lập lịch, duy trì và xử lý các yêu cầu mở rộng quy mô cho các instance này. Các task chạy trên ECS luôn nằm trong cluster.

Khi các Task được chạy trên Fargate, tài nguyên của cluster sẽ được quản lý bởi Fargate. Khi sử dụng EC2 launch type thì các cluster là những group ECS Container Instance (được chạy bằng các EC2 instance).

Một Cluster có thể chạy nhiều Service. Nếu sản phẩm có nhiều ứng dụng, bạn có thể đặt một số ứng dụng này vào một Cluster. Điều này giúp sử dụng hiệu quả hơn các tài nguyên có sẵn và giảm thiểu thời gian thiết lập so với mô hình quản lý truyền thống.

Amazon ECS tải các container image của bạn từ registry mà bạn đã setting trước đó sau đó sẽ run những images này trong cluster của bạn:

- Cluster là Region-specific.
- Cluster có thể chứa nhiều tasks sử dụng cả Fargate launch type và EC2 launch type.
- Khi các task sử dụng EC2 launch type, các clusters có thể chứa nhiều ECS Container Instance khác nhau. Tại cùng một thời điểm, một ECS Container Instance chỉ thuộc về duy nhất một cluster.
- Bạn có thể tạo một custom IAM policy cho cluster cho phép hoặc giới hạn user access tới clusters.

7.3 Tích hợp aws ECR

Khi ta đến aws ECS thì không thể thiếu aws ECR. Amazon (Amazon ECR) là sổ đăng ký bộ chia sẻ được quản lý hoàn toàn, cung cấp dịch vụ lưu trữ hiệu suất cao để bạn có thể triển khai các thành phần được cập nhật, bạn có thể tạo một hoặc nhiều kho lưu trữ trong sổ đăng ký của mình và lưu trữ image(container) trong đó. Kho lưu trữ Amazon ECR chứa hình ảnh Docker, Open Container Initiative (OCI) và các thành phần lạ tương thích với OCI của bạn.

8 Amazon Route 53

Amazon Route 53 là một dịch vụ quản lý DNS (Domain Name System) của Amazon Web Services (AWS). Nó cung cấp các chức năng quản lý tên miền và địa chỉ IP, cho phép người dùng đăng ký và quản lý các tên miền, tạo và quản lý bản ghi DNS, và điều hướng yêu cầu DNS đến các tài nguyên AWS và các máy chủ bên ngoài.

Dịch vụ này cho phép người dùng đăng ký tên miền mới hoặc chuyển tên miền từ nhà cung cấp khác vào AWS. Người dùng có thể quản lý bản ghi DNS như A (IPv4),

AAAA (IPv6), CNAME, MX (mail exchange), TXT (text), và nhiều loại bản ghi khác.

Amazon Route 53 cung cấp tính năng đáng chú ý như load balancing, routing hàng loạt, kiểm tra sẵn sàng và theo dõi hiệu suất. Điều này giúp đảm bảo rằng yêu cầu DNS được điều hướng đến các phiên bản ứng dụng khả dụng nhất và giảm thiểu thời gian chết của hệ thống.

Với Amazon Route 53, người dùng có thể xây dựng kiến trúc đám mây linh hoạt và đáng tin cậy, đồng thời tăng cường khả năng mở rộng và sẵn sàng của ứng dụng và trang web.

9 Amazon IAM

IAM (Identity and Access Management) cho phép bạn quản lý truy cập vào các dịch vụ và tài nguyên AWS một cách bảo mật. Khi sử dụng IAM, bạn có thể tạo, quản lý người dùng và nhóm AWS, sử dụng các quyền để cho phép và từ chối quyền truy cập vào tài nguyên AWS của họ.

IAM là một tính năng được cung cấp miễn phí của tài khoản AWS. Bạn chỉ phải trả phí cho việc sử dụng các dịch vụ AWS khác bởi người dùng của bạn.

Để bắt đầu sử dụng IAM hoặc nếu bạn đã đăng ký AWS, vui lòng truy cập Bảng điều khiển quản lý AWS và bắt đầu với Những biện pháp thực hành tốt nhất với IAM.

Tác dụng của IAM:

- Bạn có thể tạo, quản lý người dùng và nhóm AWS, sử dụng các quyền để cho phép và từ chối quyền truy cập vào tài nguyên AWS.
- Cấp cho người khác quyền quản trị tài nguyên AWS.
- Cấp các quyền khác nhau cho các tài nguyên khác nhau (Amazon S3, Amazon EC2, and other AWS services)...

Cách truy cập và sử dụng IAM:

- Bảng điều khiển quản lý AWS: Bảng điều khiển là giao diện dựa trên trình duyệt để quản lý tài nguyên IAM và AWS.
- Công cụ dòng lệnh AWS: Bạn có thể sử dụng các công cụ dòng lệnh AWS để phát lệnh tại dòng lệnh của hệ thống để thực hiện các tác vụ IAM và AWS. Sử dụng dòng lệnh có thể nhanh hơn và thuận tiện hơn so với bàn điều khiển. Các công cụ dòng lệnh cũng hữu ích nếu bạn muốn xây dựng các tập lệnh thực hiện các tác vụ AWS.
- AWS cung cấp hai bộ công cụ dòng lệnh: Giao diện dòng lệnh AWS (AWS CLI) và Công cụ AWS cho Windows PowerShell.

10. Những dịch vụ quan trọng khác.

Nhìn chung trong tất cả các dịch vụ quan trọng của AWS có khoảng 16 dịch vụ cơ bản xuất hiện nhiều trong mỗi doanh nghiệp bao gồm:

- + Amazon EC2 (*)
- + Amazon VPC (*)
- + Amazon Lambda (*)
- + Amazon API Gateway (*)
- + Amazon Elastic Beanstalk
- + Amazon S3 (*)
- + Amazon ELB (*)
- + Amazon IAM (*)
- + Amazon RDS (*)
- + Amazon DynamoDB
- + Amazon CloudFormation (*)
- + Amazon CloudWatch (*)
- + Amazon Amplify
- + Amazon Cognito
- + Amazon ECS (serverful->EC2,serverless->fargate) (*) hoặc EKS đối với những doanh nghiệp sử dụng môi trường K8S
- + Amazon Codepipeline

Chương IV: Thực hành với các dịch vụ AWS

1. Cơ sở lý thuyết

Công nghệ sử dụng:

- **Java Spring Boot:** nằm trong hệ sinh thái của JAVA Spring FW, hỗ trợ tạo các ứng dụng web nhanh chóng, dùng trong việc xây dựng các API. Spring là một khung công tác Java toàn diện cung cấp nhiều mô-đun khác nhau để xây dựng các ứng dụng cáp doanh nghiệp. Nó bao gồm các tính năng như tiêm phụ thuộc, lập trình hướng theo khía cạnh, truy cập dữ liệu, quản lý giao dịch, v.v. Spring được xây dựng dựa trên API Servlet và sử dụng nó để xử lý các yêu cầu và phản hồi HTTP. Mặt khác, API Servlet là một API Java tiêu chuẩn xác định hợp đồng để xây dựng các ứng dụng web bằng Java. Nó cung cấp các lớp và giao diện để xử lý các yêu cầu HTTP, quản lý phiên và tạo nội dung web động. Được mở rộng và thu gọn hơn so với API Servlet với các ưu điểm lớn như: Dependency Injection, AOP, Spring MVC, Quản lý Transaction, dễ dàng tích hợp và có một cộng đồng đông đảo.



- **Angular:** Angular là một framework phát triển ứng dụng web front-end mã nguồn mở được phát triển bởi Google. Nó cho phép xây dựng các ứng dụng web động và đa trang (single-page applications) một cách dễ dàng và hiệu quả.

Angular sử dụng ngôn ngữ TypeScript, một phiên bản tiên tiến của JavaScript, để viết mã. Nó cung cấp một kiến trúc phần mềm mạnh mẽ và các công cụ để quản lý các thành phần, giao diện người dùng, dữ liệu và các hoạt động khác trong ứng dụng.

Các tính năng chính của Angular bao gồm:

- Cấu trúc component-based: Angular sử dụng cấu trúc component-based, cho phép chia ứng dụng thành các thành phần nhỏ và tái sử dụng chúng trong toàn bộ ứng dụng.
- Two-way data binding: Angular cung cấp tính năng two-way data binding, cho phép đồng bộ hóa dữ liệu giữa giao diện người dùng và mã JavaScript một cách tự động.
- Routing: Angular cung cấp một hệ thống routing mạnh mẽ để điều hướng giữa các trang và hiển thị nội dung tương ứng.
- Dependency Injection: Angular hỗ trợ dependency injection, giúp quản lý và nạp các phụ thuộc giữa các thành phần trong ứng dụng.
- Testing: Angular cung cấp các công cụ và thư viện hỗ trợ cho việc kiểm thử đơn vị (unit testing) và kiểm thử tích hợp (integration testing) trong quá trình phát triển ứng dụng.
- Hỗ trợ đa nền tảng: Angular hỗ trợ phát triển ứng dụng web trên nhiều nền tảng khác nhau, bao gồm máy tính để bàn, điện thoại di động và các thiết bị di động khác.

Angular là một trong những framework phát triển web phổ biến và được sử dụng rộng rãi trong cộng đồng phát triển ứng dụng web. Nó cung cấp một cách tiếp cận mạnh mẽ và hiệu quả để xây dựng các ứng dụng web đa dạng và phức tạp.



- **Docker:** Docker là một nền tảng mã nguồn mở cho việc đóng gói, triển khai và chạy ứng dụng trong các môi trường độc lập gọi là containers. Containers là một cách

để đóng gói phần mềm và tất cả các phụ thuộc của nó vào một gói duy nhất, đảm bảo tính di động và khả năng tái sử dụng.

Docker cho phép bạn tạo ra các containers độc lập mà không cần quan tâm đến hệ điều hành hoặc môi trường máy tính đang chạy. Bằng cách sử dụng Docker, bạn có thể đóng gói ứng dụng và các phụ thuộc của nó vào một container duy nhất, và sau đó triển khai và chạy container này trên bất kỳ máy tính hoặc máy chủ nào đã được cài đặt Docker.

Một số lợi ích của Docker bao gồm:

- Đóng gói độc lập: Docker cho phép bạn đóng gói ứng dụng và các phụ thuộc của nó vào một container độc lập, giúp đảm bảo tính di động và khả năng tái sử dụng.
- Tính nhất quán và đồng nhất: Docker đảm bảo rằng ứng dụng của bạn sẽ chạy đúng như mong đợi trên mọi môi trường và hệ điều hành.
- Tính linh hoạt: Docker cho phép bạn triển khai và chạy các ứng dụng trên nhiều môi trường và hệ điều hành khác nhau một cách dễ dàng.
- Hiệu suất cao: Containers Docker khá nhẹ và có thể khởi động nhanh chóng, giúp giảm thời gian khởi động và tối ưu hóa sử dụng tài nguyên.
- Quản lý phụ thuộc: Docker cho phép bạn quản lý các phụ thuộc của ứng dụng một cách dễ dàng và đảm bảo tính nhất quán giữa các môi trường khác nhau.

Docker đã trở thành một công cụ phổ biến trong việc triển khai và quản lý ứng dụng, đặc biệt là trong môi trường đám mây và DevOps. Nó cung cấp một giải pháp tiện lợi và mạnh mẽ để xây dựng và triển khai các ứng dụng đa nền tảng.



Công cụ thiết kế ứng dụng:

- **IntelliJ IDEA:** là một IDE Java để phát triển các phần mềm máy tính và được phát triển bởi JetBrains.
- **H2 Database:** H2 là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở được viết bằng Java. Nó được thiết kế để cung cấp một cơ sở dữ liệu nhẹ nhưng mạnh mẽ cho các ứng dụng Java (viết và thực thi ngay trong mã chương trình).

H2 database hỗ trợ đa người dùng, khả năng nhúng và chạy trên nhiều nền tảng khác nhau. Nó cung cấp các tính năng phổ biến của một hệ quản trị cơ sở dữ liệu quan hệ, bao gồm:

- SQL và JDBC: H2 hỗ trợ ngôn ngữ truy vấn SQL tiêu chuẩn và cung cấp một giao diện JDBC để tương tác với cơ sở dữ liệu.
- In-memory database: H2 cho phép lưu trữ dữ liệu trong bộ nhớ chung không yêu cầu lưu trữ trên đĩa. Điều này giúp tăng tốc độ truy vấn và phù hợp cho các ứng dụng đòi hỏi hiệu suất cao.
- Cơ chế khóa: H2 hỗ trợ cơ chế khóa đồng thời (concurrency control) để đảm bảo tính nhất quán và đồng bộ hóa truy cập của nhiều người dùng đến cùng một tài nguyên.
- Hỗ trợ các tính năng SQL tiên tiến: H2 cung cấp các tính năng SQL tiên tiến như khung nhìn (views), chức năng (functions), ghi nhật ký (logging), và nhiều hơn nữa.
- Độ tin cậy và khả năng phục hồi: H2 hỗ trợ các tính năng như ghi nhật ký (logging) và khả năng sao lưu và phục hồi để đảm bảo tính tin cậy của dữ liệu.

H2 database thường được sử dụng trong các ứng dụng Java nhỏ và trung bình, đặc biệt là trong quá trình phát triển và kiểm thử. Nó cung cấp một giải pháp đơn giản và dễ sử dụng để quản lý dữ liệu trong các ứng dụng Java.

Môi trường triển khai, phát triển: Amazon Web Service (AWS)

2. Bài thực hành :

Gồm ba bài thực hành trên AWS để ôn tập lại những khái niệm trên:

Lab thứ nhất sẽ thao tác EC2

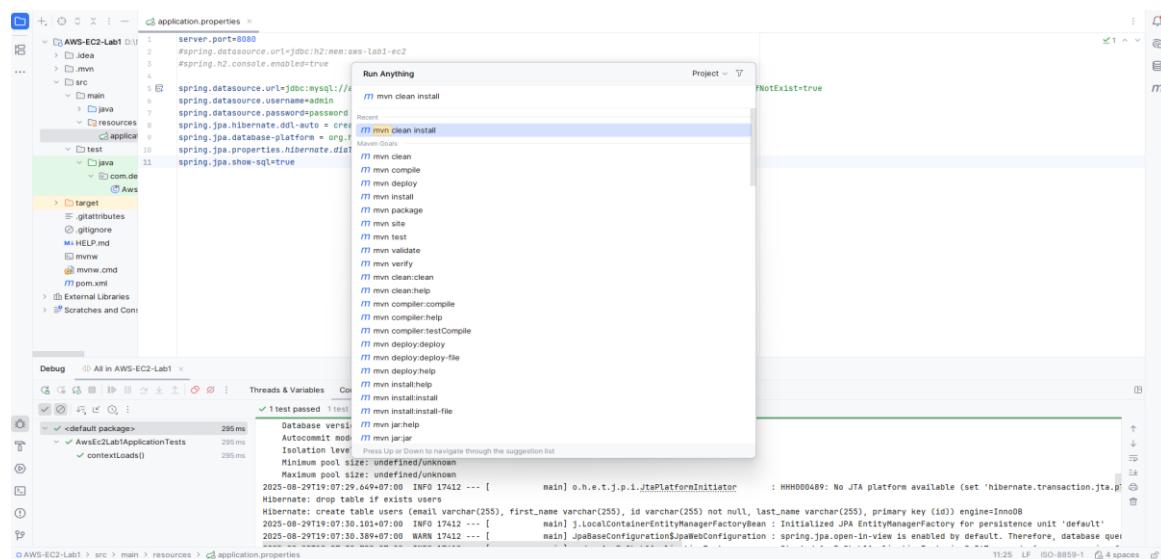
Lab thứ hai sẽ xây dựng Serverless Architecture với Lambda và API Gateway

Lab thứ ba sẽ tạo một VPC đồng thời thao tác trực tiếp với ECS và xây dựng một mô hình tự động CI/CD basic bằng AWS Codepipeline.

LAB-1

Mô tả: Sẽ thực hiện việc demo một chương trình spring boot đơn giản được kết nối với RDS(Mysql) qua public IPv4 DNS và chương trình được lưu trữ trên S3 và chạy trên EC2.

Bước 1: Xây dựng Spring boot basic, Thêm sửa xóa đơn giản.



Lab1-Hinh1

Tiến hành Build code thành file .jar để sử dụng lưu ý đường dẫn file .jar. Sau khi đã BUILD SUCCESS thì kết thúc bước 1.

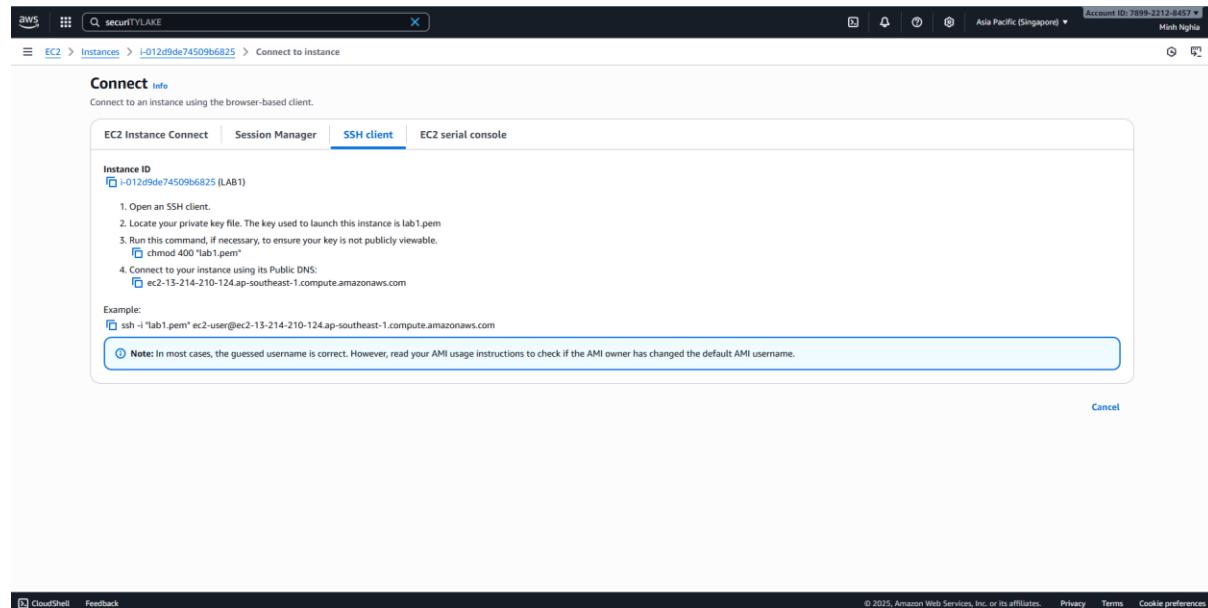
Bước 2: Tạo AWS EC2

Truy cập <https://aws.amazon.com/ec2> để tạo AWS EC2

Nhấn Create EC2 và đặt cấu hình mặc định như sau:

Name: LAB1, Key pair(login): lab1 và set những thuộc tính còn lại ở chế độ default và tiến hành khởi tạo EC2 Nhấn **Launch an instance**

Nhấn vào connect để lấy thông tin liên kết với EC2 (có thể sử dụng một số công cụ lệnh như git bash,... hoặc putty để truy cập vào Instance EC2)



Lab1-Hinh2

ở thư mục chứa key pair, click chuột trái git bash tại thư mục đó và thực hiện thao tác theo thứ tự.

```
ec2-user@ip-172-31-36-243:~  
Admin@Admin-PC MINGW64 ~  
$ cd ~/Downloads  
Admin@Admin-PC MINGW64 ~/Downloads  
$ chmod 400 lab1.pem  
Admin@Admin-PC MINGW64 ~/Downloads  
$ ssh -i "lab1.pem" ec2-user@ec2-13-214-210-124.ap-southeast-1.compute.amazonaws.com  
The authenticity of host 'ec2-13-214-210-124.ap-southeast-1.compute.amazonaws.com (13.214.210.124)' can't be established.  
ED25519 key fingerprint is SHA256:A8IIS+4brdUHTOWLElKmkQ+8W4nkdfZR6RTUFSwibT8.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-13-214-210-124.ap-southeast-1.compute.amazonaws.com' (ED25519) to the list of known hosts.  
' #  
~\_\_ ##### Amazon Linux 2023  
~~ \#####  
~~ \|##|  
~~ \|#/ https://aws.amazon.com/linux/amazon-linux-2023  
~~ \|/ .->  
~~ \|/ /  
~/m/  
[ec2-user@ip-172-31-36-243 ~]$ |
```

Lab1-Hinh3

Hoàn thành và kết thúc bước 2.

Bước 3: Tạo bộ chứa AWS S3

Truy cập <https://aws.amazon.com/s3> để tạo bộ chứa AWS S3 Nhấn Create bucket để tạo 1 bộ chứa

Tắt block all public access và nhấn Create bucket.

The screenshot shows the AWS S3 Buckets page. At the top, there's a search bar and a 'Create bucket' button. Below it, a table lists 'General purpose buckets (0)'. A message says 'Buckets are containers for data stored in S3.' and 'No buckets. You don't have any buckets.' There are also sections for 'Account snapshot' and 'External access summary - new'.

Lab1-Hinh4

Tiếp tục truy cập vào bucket và upload code lên đó.

The screenshot shows the AWS S3 Objects page for the 'aws-lab1-demo1' bucket. It has tabs for Objects, Properties, Permissions, Metrics, Management, and Access Points. The Objects tab is selected. A message says 'Objects are the fundamental entities stored in Amazon S3.' and 'No objects. You don't have any objects in this bucket.' There's a 'Upload' button at the bottom.

Lab1-Hinh5

Tiến hành upload file jar đã build lên bucket và nhấn Upload.

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 5 GB, use the AWS Command Line Interface or AWS Transfer Family.

Files and folders (1 total, 55.9 MB)

All files and folders in this table will be uploaded.

Name	Date modified	Type	Size
AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar	8/25/2023 9:44 PM	JAR File	57,799 KB

Destination [Info](#)

Destination [s3://aws-lab1-demo1](#)

Destination details

Bucket settings that impact new objects stored in the specified destination.

Permissions

Grant public access and access to other AWS accounts.

Properties

Specify storage class, encryption settings, tags, and more.

Lab1-Hinh6

Sau khi hoàn thành, click vào file đã upload để lấy Object URL

AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar [Info](#)

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

Properties [Permissions](#) [Versions](#)

Object overview

Owner tranadu123456

AWS Region Asia Pacific (Singapore) ap-southeast-1

Last modified August 29, 2025, 19:45:06 (UTC+07:00)

Size 55.9 MB

Type jar

Key [AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar](#)

S3 URI [s3://aws-lab1-demo1/AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar](#)

Amazon Resource Name (ARN) [arn:aws:s3:::aws-lab1-demo1/AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar](#)

Entity tag (tag) [c0df88d6f58c09ff164ecc716ffdb344-4](#)

Object URL [https://aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com/AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar](#)

Object management overview

The following bucket properties and object management configurations impact the behavior of this object.

Bucket properties

Bucket Versioning

When enabled, multiple variants of an object can be stored in the bucket to easily recover from unintended user actions and application errors.

Management configurations

Replication status

When a replication rule is applied to an object the replication status indicates the progress of the operation.

Lab1-Hinh7

Vào phần Permissions và chọn Bucket Policy để thêm chính sách cho bucket:

- Trong trình chỉnh sửa chính sách nhóm, bạn có thể chỉ định chính sách cho phép truy cập công khai cho phép đọc và lấy về. và nhấn save change

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::aws-lab1-demo1/*"

}

]

}

The screenshot shows the AWS S3 Bucket Policy editor. The policy is defined as follows:

```
1 {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Sid": "PublicReadGetObject",  
6             "Effect": "Allow",  
7             "Principal": "*",  
8             "Action": "s3:GetObject",  
9             "Resource": "arn:aws:s3:::aws-lab1-demo1/*"  
10        }  
11    ]  
12 }  
13
```

On the right side, there is a panel titled "Edit statement" with a sub-section "Select a statement". It contains the message "Select an existing statement in the policy or add a new statement." and a button "+ Add new statement".

Lab1-Hinh8

Kết thúc bước 3.

Bước 4: Tạo AWS RDS

Truy cập <https://aws.amazon.com/rds> để tạo AWS RDS. Click chọn Create database

Engine options: Mysql 8.0.43

Templates: Free tier

Settings:

- + DB instance identifier: aws-ec2-lab1
- + Master username: admin
- + Master password: password

Instance configuration: t3.micro

Còn lại set ở chế độ default.

Bật Public Access IP (vì RDS đang ở default subnet public nếu ở private thì phải tắt và tạo định tuyến cho RDS) và tiến hành tạo RDS.

The screenshot shows the AWS RDS Databases page. At the top, there's a search bar and account information. Below it, a table lists one database entry:

DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU	Current activity
aws-ec2-lab1	Creating	Instance	MySQL Co...	-	db.t3.micro	-	-	-

At the bottom of the page, there are links for CloudShell, Feedback, and a copyright notice.

Lab1-Hinh9

Bước 5: Kết nối và khởi động chương trình

Sau khi tạo xong RDS ta tiến hành lấy đường dẫn URL Endpoint cho chương trình demo.

The screenshot shows the AWS RDS DB instance details page for 'aws-ec2-lab1'. It includes sections for Summary, Connectivity & security, Networking, and Security.

Summary:

DB identifier	Status	Role	Engine	Recommendations
aws-ec2-lab1	Backing-up	Instance	MySQL Community	
CPU	Class	Current activity	Region & AZ	ap-southeast-1b
0.00%	db.t3.micro			

Connectivity & security:

Endpoint	Networking	Security
aws-ec2-lab1.cnsgug26ak2ru.ap-southeast-1.rds.amazonaws.com	Availability Zone: ap-southeast-1b VPC: vpc-099e821e6a9330ea0 Subnet group: default-vpc-099e821e6a9330ea0 Subnets: subnet-07834287e40748c92, subnet-0ba7c1c08ba9491a3, subnet-07fae06f1480f513c Network type:	VPC security groups: default (sg-0ca16e43fa126b07b) (Active) Publicly accessible: Yes Certificate authority: Info rds-ca-rsa2048-g1 Certificate authority date: May 22, 2061, 05:59 (UTC+07:00) DB instance certificate expiration date: August 29, 2026, 19:59 (UTC+07:00)

Lab1-Hinh10

Tiến hành tạo giao tiếp cho RDS qua SG như sau:

The screenshot shows the AWS EC2 Security Groups console with the path: EC2 > Security Groups > sg-0ca16e43fa126b07b - default > Edit inbound rules. The 'Inbound rules' section lists two rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
-	All traffic	All	All	Custom	sg-0ca16e43fa126b07b
-	MySQL/Aurora	TCP	3306	Anywhere	sg-0ca16e43fa126b07b

Buttons at the bottom include 'Add rule', 'Cancel', 'Preview changes', and 'Save rules'.

Lab1-Hinh11

Tiến hành gán Endpoint vào file cấu hình của chương trình demo. Để kết nối chương trình với AWS RDS trên Amazon (có thể dùng Mysql Workbench để kiểm tra kết nối).

The screenshot shows the IntelliJ IDEA interface with the 'application.properties' file open in the editor. The file contains the following configuration:

```
server.port=8080
spring.datasource.url=jdbc:mysql://aws-ec2-lab1.cnsg2ak2ru.ap-southeast-1.rds.amazonaws.com:3306/demo?createDatabaseIfNotExist=true
spring.datasource.username=admin
spring.datasource.password=password
spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.jpa.database-platform=org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```

The 'Run' tool window at the bottom shows a successful build log for the 'AWS-EC2-Lab1' project:

```
[INFO] --- install:3.1.4:install (@AWS-EC2-Lab1) ...
[INFO] Installing D:\NIENLUAN\AWS-EC2-Lab1\pom.xml to C:\Users\Admin.m2\repository\com\demo\AWS-EC2-Lab1\0.0.1-SNAPSHOT\AWS-EC2-Lab1-0.0.1-SNAPSHOT.pom
[INFO] Installing D:\NIENLUAN\AWS-EC2-Lab1\target\AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar to C:\Users\Admin.m2\repository\com\demo\AWS-EC2-Lab1\0.0.1-SNAPSHOT\AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar
[INFO] ...
[INFO] BUILD SUCCESS
[INFO] Total time: 4.784 s
[INFO] Finished at: 2025-08-29T20:05:33+07:00
[INFO] ...
```

Lab1-Hinh12

Build lại và tiến hành Upload lại file lên S3 để chương trình kết nối với RDS.

Bước 6: Triển khai code và chạy chương trình trên EC2

Mở EC2 và download những gói cần thiết để chạy chương trình

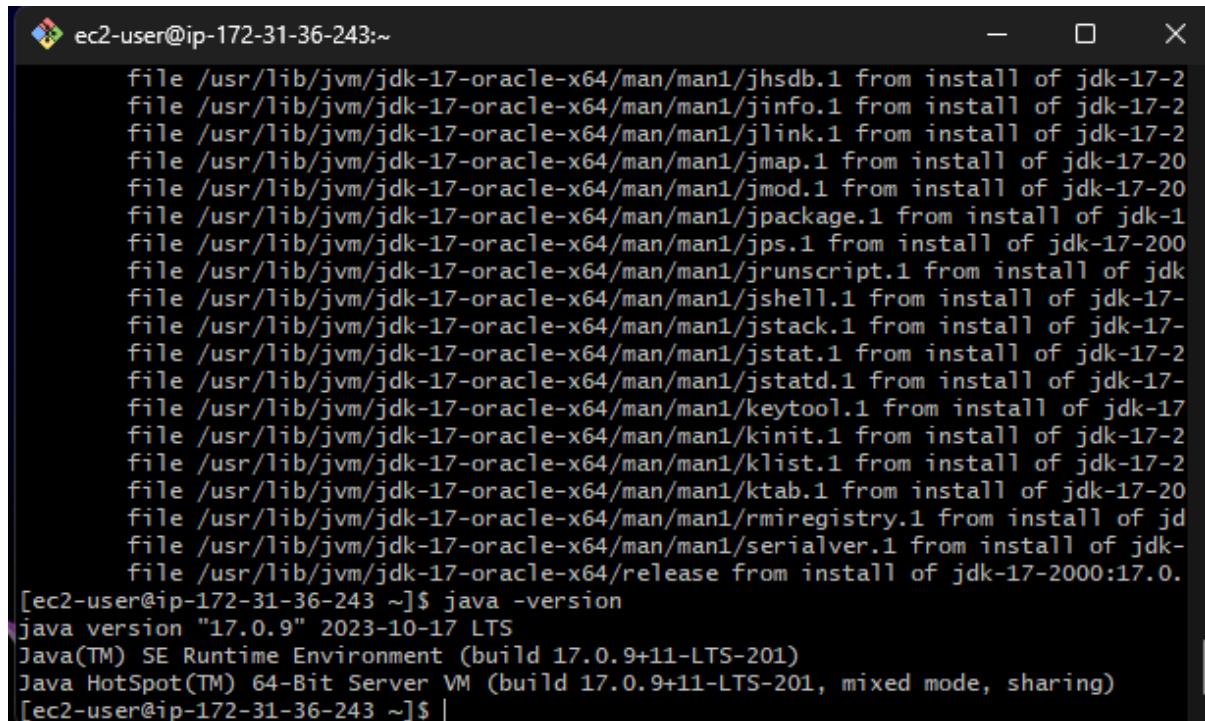
java 17: thực hiện các lệnh sau để download java 17 cho EC2.

```
sudo yum -y update
```

```
sudo yum -y install wget vim
```

```
wget https://download.oracle.com/java/17/archive/jdk-17.0.9\_linux-x64\_bin.rpm
```

```
sudo rpm -i jdk-17.0.9_linux-x64_bin.rpm
```



The screenshot shows a terminal window titled 'ec2-user@ip-172-31-36-243:~'. The window displays the output of several commands:

```
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jhsdb.1 from install of jdk-17-2
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jinfo.1 from install of jdk-17-2
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jlink.1 from install of jdk-17-2
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jmap.1 from install of jdk-17-20
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jmod.1 from install of jdk-17-20
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jpackage.1 from install of jdk-1
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jps.1 from install of jdk-17-200
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jrunscript.1 from install of jdk
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jshell.1 from install of jdk-17-
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jstack.1 from install of jdk-17-
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jstat.1 from install of jdk-17-2
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/jstatted.1 from install of jdk-17-
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/keytool.1 from install of jdk-17
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/kinit.1 from install of jdk-17-2
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/klist.1 from install of jdk-17-2
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/ktab.1 from install of jdk-17-20
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/rmiregistry.1 from install of jd
file /usr/lib/jvm/jdk-17-oracle-x64/man/man1/serialver.1 from install of jdk-
file /usr/lib/jvm/jdk-17-oracle-x64/release from install of jdk-17-2000:17.0.
[ec2-user@ip-172-31-36-243 ~]$ java -version
java version "17.0.9" 2023-10-17 LTS
Java(TM) SE Runtime Environment (build 17.0.9+11-LTS-201)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.9+11-LTS-201, mixed mode, sharing)
[ec2-user@ip-172-31-36-243 ~]$ |
```

Lab1-Hinh13

Thực hiện download file .jar từ S3 về bằng lệnh sau

```
wget https://aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com/AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar
```

```
[root@ip-172-31-36-243:~] AWS-EC2-Lab1-0.0.1-SNAPSHOT.ja 100%[=====>] 55.94M 113MB/s in 0.5s
2025-08-29 13:18:09 (113 MB/s) - 'AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar' saved [58652832/58652832]

[ec2-user@ip-172-31-36-243 ~]$ # ls
[ec2-user@ip-172-31-36-243 ~]$ sudo -i
[root@ip-172-31-36-243 ~]# ls
[root@ip-172-31-36-243 ~]# wget https://aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com/AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar
--2025-08-29 13:18:48- https://aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com/AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar
Resolving aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com (aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com)... 52.219.164.214,
3.5.148.100, 52.219.128.23, ...
Connecting to aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com (aws-lab1-demo1.s3.ap-southeast-1.amazonaws.com)|52.219.164.214|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 58652832 (56M) [binary/octet-stream]
Saving to: 'AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar'

AWS-EC2-Lab1-0.0.1-SNAPSHOT.ja 100%[=====>] 55.94M 106MB/s in 0.5s
2025-08-29 13:18:49 (106 MB/s) - 'AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar' saved [58652832/58652832]

[root@ip-172-31-36-243 ~]# ls
AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar
[root@ip-172-31-36-243 ~]#
```

Lab1-Hinh14

Sử dụng lệnh: java -jar AWS-EC2-Lab1-0.0.1-SNAPSHOT.jar để chạy ứng dụng

Lab1-Hinh15

Sau khi hoàn thành thì ta mở trình duyệt để test thử hoặc postman với đường dẫn trong EC2 public IPv4 DNS (nhớ xem cấu hình SG có mở cổng không). Nếu muốn

các dịch vụ khác(angular,reactjs,flutter,...) thì thực hiện download qua URL như trên những thành phần cho chương trình sau đó khởi chạy(bước cấu hình cho EC2).



```
[{"id": "0ab9c0fe-6804-4057-9980-02f2514b49f3", "firstname": "A10", "lastName": "Nguyen Van", "email": "nva10@gmail.com"}, {"id": "7d923424-e0d6-4c35-0de3-62c56d421731", "firstname": "A1", "lastName": "Nguyen Van", "email": "nva1@gmail.com"}, {"id": "4c595ab7-e11c-49e7-90c5-491e93c68701", "firstname": "A2", "lastName": "Nguyen Van", "email": "nva2@gmail.com"}, {"id": "73baa447f-2861-4d53-93c7-c23599cd6932", "firstname": "A3", "lastName": "Nguyen Van", "email": "nva3@gmail.com"}, {"id": "00998be9-8937-49d4-9bed-fcb0b196eac9", "firstname": "A3", "lastName": "Nguyen Van", "email": "nva3@gmail.com"}, {"id": "8fd27990-413a-4ff3-0765-db95bd56a624", "firstname": "A3", "lastName": "Nguyen Van", "email": "nva3@gmail.com"}, {"id": "c181179c-4726-4086-819e-bc26f7ed814d", "firstname": "A1", "lastName": "Nguyen Van", "email": "nva1@gmail.com"}, {"id": "ee79416a-7ade-4717-a02f-b77990121da8", "firstname": "A5", "lastName": "Nguyen Van", "email": "nva5@gmail.com"}]
```

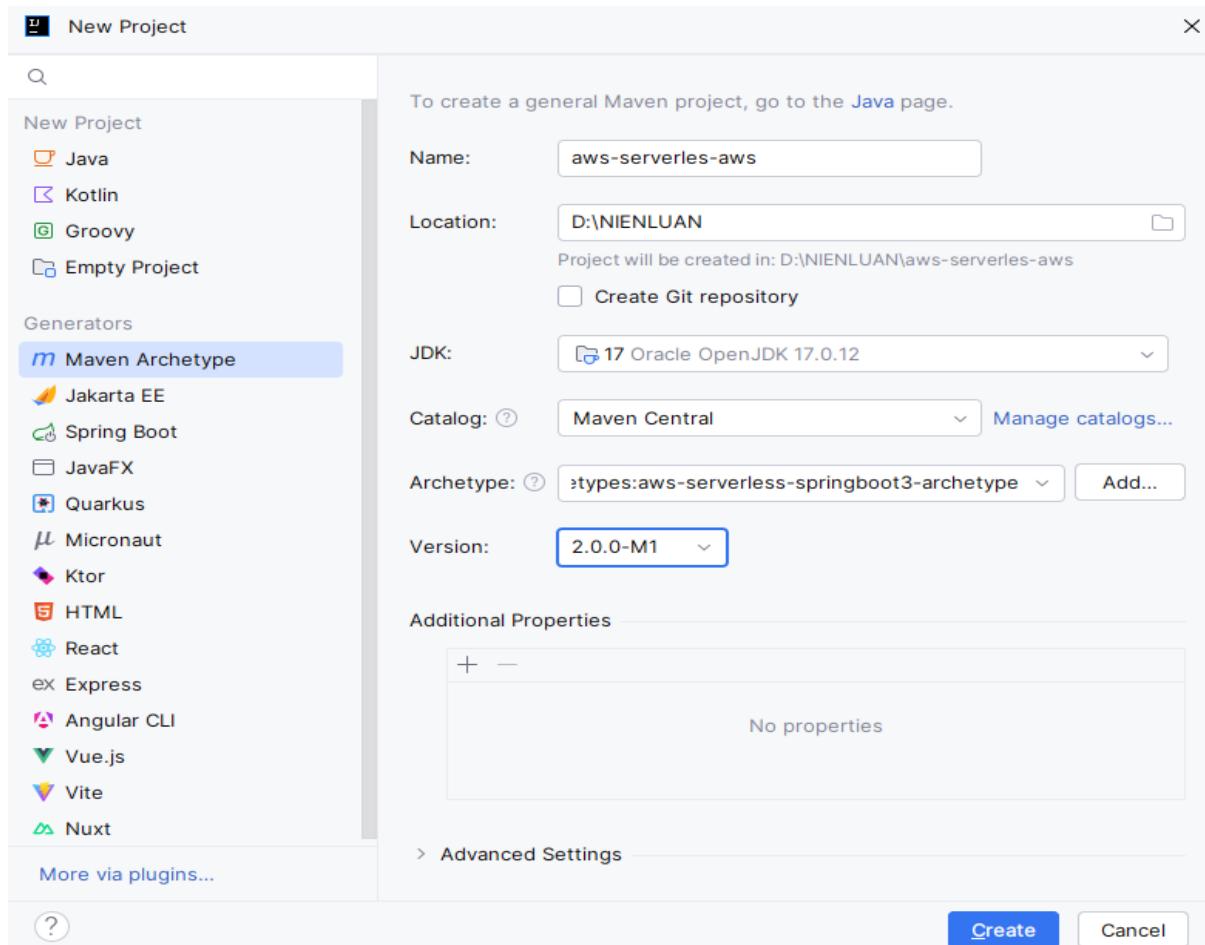
Lab1-Hinh16

Sau khi hoàn thành chạy được ứng dụng demo thì nhớ Shutdown EC2 và xóa nó đi nếu không cần thiết để tránh trả phí và kết thúc bài LAB-1.

LAB-2

Mô tả: Sẽ tránh những thao tác lặp lại nên việc tạo những dịch vụ AWS trùng lặp sẽ được chú thích và không thực hiện lại. Trong bài này sẽ thực hiện việc demo một chương trình spring boot đơn giản chạy trên Lambda và được kết nối với cổng AWS API gateway ra ngoài Internet.

Bước 1: Tạo Project Maven Archetype, build zip



Lab2-Hinh1

- + Ở phần Archetype chọn com.amazonaws.serverless.archetypes:aws-serverless-springboot3-archetype để maven build một project spring boot có cấu hình serverless lambda handler
- + Project Maven sẽ tự động tạo ra một LambdaHandler, trong file pom.xml đã có đầy đủ các dependency cần thiết để project spring khởi chạy trên lambda nên không cần import thêm phụ thuộc bao gồm cả tomcat,... .

Lambda function handle là phương thức trong mã hàm của bạn xử lý các sự kiện. Khi hàm của bạn được gọi, Lambda sẽ chạy phương thức xử lý (handleRequest). Hàm của bạn chạy cho đến khi trình xử lý trả về phản hồi, thoát hoặc hết thời gian chờ.

The screenshot shows the IntelliJ IDEA interface with the code editor open to the StreamLambdaHandler.java file. The code implements RequestStreamHandler and handles AWS Lambda proxy requests. The code includes logic for starting a Spring Boot application and handling the proxy stream.

```
package com.demo;

import ...

public class StreamLambdaHandler implements RequestStreamHandler {
    private static final String AWS_PROXY_REQUEST = "awsProxyRequest";
    private static final String AWS_PROXY_RESPONSE = "awsProxyResponse";
    static {
        try {
            handler = new SpringBootLambdaContainerHandler.getAwsProxyHandler(Application.class);
            // For applications that take longer than 10 seconds to start, use the async builder:
            // handler = new SpringBootProxyHandlerBuilder<AwsProxyRequest, AwsProxyResponse>()
            //     .defaultProxy()
            //     .asyncInit()
            //     .springBootApplication(Application.class)
            //     .buildAndInitialize();
        } catch (ContainerInitializationException e) {
            // if we fail here, we re-throw the exception to force another cold start
            e.printStackTrace();
            throw new RuntimeException("Could not initialize Spring Boot application", e);
        }
    }

    @Override
    public void handleRequest(InputStream inputStream, OutputStream outputStream, Context context)
        throws IOException {
        handler.proxyStream(inputStream, outputStream, context);
    }
}
```

Lab2-Hinh2

- + Tạo những class cơ bản cho project basic
(<https://github.com/minhnhgia0910/NIENLUAN>)
- + Sau khi khởi động ứng dụng spring boot sẽ chạy trên 127.0.0.1:8080 với giao diện JSON như dưới:

```
[{"id": "846026d0-7bd1-4010-afe2-5a785845fcfb", "name": "May Tinh", "price": 4300}, {"id": "711fb70b-95cc-4bf7-a13f-a1466301d4e3", "name": "TV", "price": 1200}, {"id": "34dcc8b1-3588-4271-a376-582f711f6f10", "name": "Tu Lanh", "price": 3400}, {"id": "c1cac297-a476-4ff6-80f0-86807be2a76c", "name": "Dien Thoai", "price": 1500}]
```

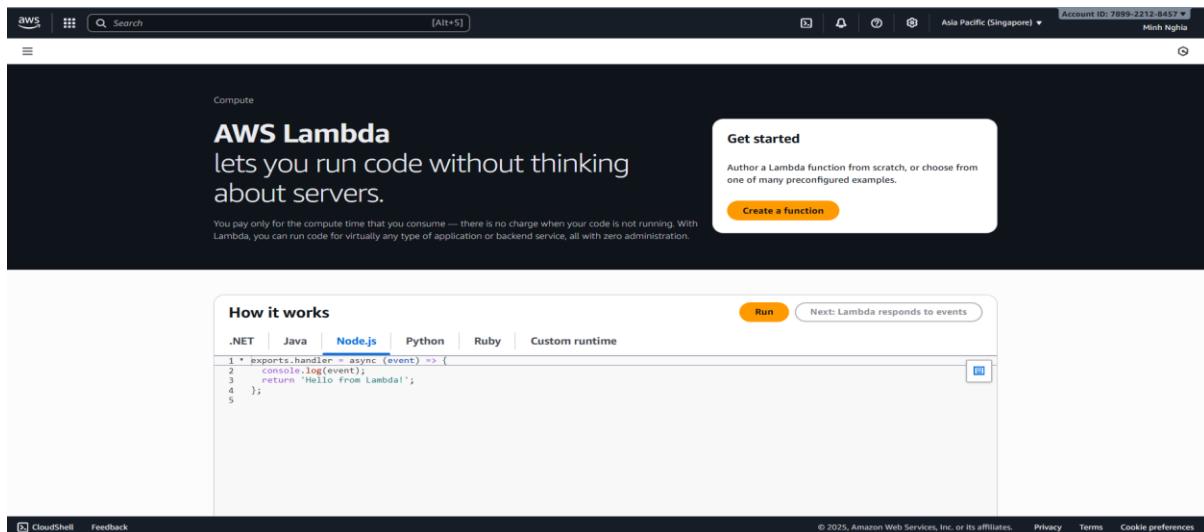
Lab2-Hinh3

- + Thực hiện build project bằng lệnh run maven build package + install,
Chú ý đường dẫn building zip

```
[INFO] --- assembly:3.5.0:single (zip-assembly) @ aws-serverless-lab2 ---
[WARNING] Parameter 'finalName' is read-only, must not be used in configuration
[INFO] Reading assembly descriptor: src\assembly\bin.xml
[INFO] Building zip: D:\MIENLUAN\aws-serverless-lab2\target\aws-serverless-lab2-1.0-SNAPSHOT-lambda-package.zip
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ aws-serverless-lab2 ---
[INFO] Skipping artifact installation
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.636 s
[INFO]
```

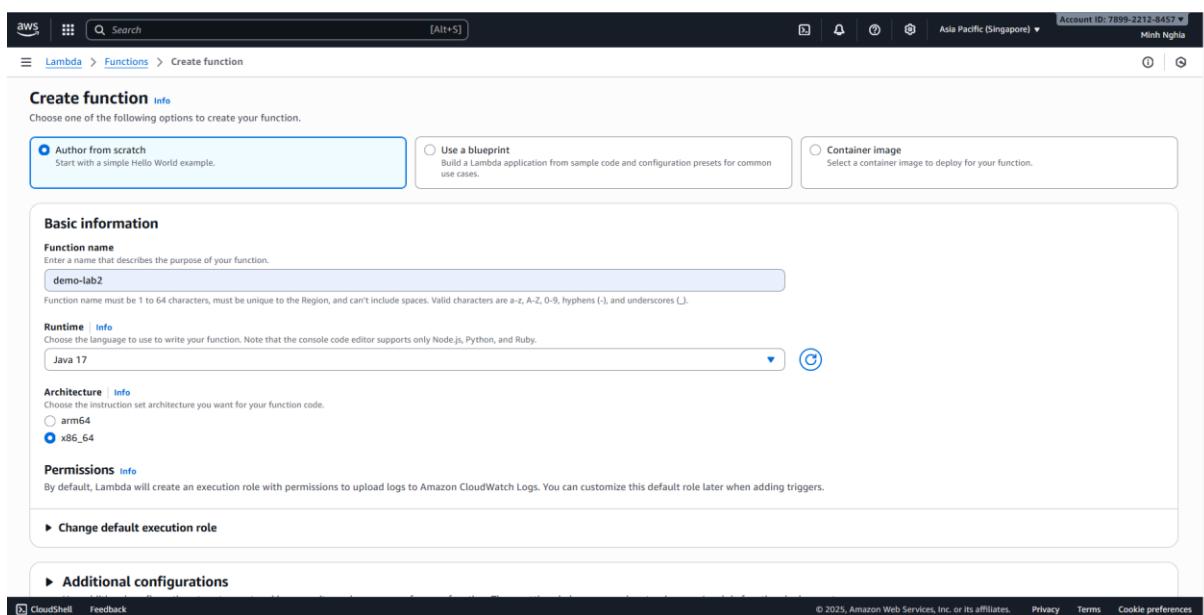
Bước 2: Tạo Lambda. Deploy code

- + đăng nhập tài khoản vào AWS và truy cập (<https://aws.amazon.com/lambda/>)



Lab2-Hinh4

- + Nhấn Create a function để tạo một Lambda



Lab2-Hinh5

- + Đặt tên và chọn Runtime là Java 17 với kiến trúc x86-64 và tiếp tục nhấn Create function để tạo Lambda

Successfully updated the function demo-lab2.

Function Overview

demo-lab2

Layers (0)

+ Add trigger + Add destination

Description

Last modified 9 minutes ago

Function ARN arn:aws:lambda:ap-southeast-1:789922128457:function:demo-lab2

Function URL | Info

Code Test Monitor Configuration Aliases Versions

Code source Info

The code editor does not support the Java 17 runtime.

Code properties Info

Package size 37.7 MB

SHA256 hash uhJWFt1tOQ8nKvo3QOQMwZmS3S8yei62p6eUcxOsYuc=

Last modified 9 minutes ago

CloudShell Feedback

Lab2-Hinh6

- + Upload code (đường dẫn .zip đã build ở trên) hoặc tạo kho chúa S3.

Upload a .zip or .jar file

When you upload a new .zip or .jar file package, it overwrites the existing code.

aws-serverless-lab2-1.0-SNAPSHOT-lambda-package.zip
39.52 MB

For files larger than 10 MB, consider uploading using Amazon S3.

Encryption with an AWS KMS customer managed key

By default, Lambda encrypts the .zip file archive using an AWS owned key.

Upload a file from Amazon S3

X

ⓘ When you upload a new .zip or .jar file package, it overwrites the existing code.

Amazon S3 link URL

Paste an S3 link URL to your function code .zip.

-1.amazonaws.com/aws-serverless-lab2-1.0-SNAPSHOT-lambda-package.zip

Encryption with an AWS KMS customer managed key

By default, Lambda encrypts the .zip file archive using an AWS owned key.

Cancel

Save

Lab2-Hinh7

- + Chú thích: Có thể tạo kho chứa S3 để upload và lưu trữ tùy ý thực hiện thao tác giống LAB-1, ở trong bài này sẽ upload kho chứa S3 vào Lambda.

The screenshot shows the AWS Lambda function configuration page for 'demo-lab2'. At the top, there's a green success message: 'Successfully updated the function demo-lab2.' Below it, a note says 'The code editor does not support the Java 17 runtime.' The 'Code properties' section shows a SHA256 hash and a timestamp of '7 minutes ago'. The 'Runtime settings' section shows 'Java 17' as the runtime, 'example.Hello::handleRequest' as the handler, and 'x86_64' as the architecture. A red arrow points to the 'Edit' button next to the Handler field. The 'Layers' section indicates 'There is no data to display.'

Lab2-Hinh8

- + Tiến hành sửa lại cấu hình Handle của Lambda.

The screenshot shows the AWS Lambda 'Edit runtime settings' interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information for 'Asia Pacific (Singapore)'. Below the navigation is a breadcrumb trail: 'Lambda > Functions > demo-lab2 > Edit runtime settings'. The main title is 'Edit runtime settings'. Under the 'Runtime' section, it says 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.' A dropdown menu shows 'Java 17' with a blue outline, and a circular refresh icon to its right. A blue callout box highlights 'New runtime available' with the message 'A new runtime is available for your function's language: Java 21'. In the 'Handler' section, the handler name 'com.demo.StreamLambdaHandler:handleRequest' is listed. Under 'Architecture', 'x86_64' is selected. A note at the bottom states: 'You can change either the function's runtime or the instruction set architecture in one update. To update both, you must repeat the update process.' At the bottom right are 'Cancel' and 'Save' buttons.

Lab2-Hinh9

- + Lấy đường dẫn thư mục vào hàm handleRequest (Lab2-Hinh2) và lưu cấu hình xử lý.

aws Search [Alt+S] Account ID: 7699-2212-8457 Asia Pacific (Singapore) Minh Nghia

Lambda > Functions > demo-lab2

Successfully updated the function demo-lab2.

Event name demo-lab2 Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings Private This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more Shareable This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - optional API Gateway AWS Proxy

Event JSON Format JSON

```
1 {  
2   "body": "eyJ0ZXN0IjoiYm9keSj9",  
3   "resource": "/proxy+",  
4   "path": "/api/proxy+",  
5   "httpMethod": "GET",  
6   "isBase64Encoded": true,  
7   "queryStringParameters": {  
8     "foo": "bar"  
9   },  
10  "multiValueQueryStringParameters": {  
11    "foo": [  
12      "bar"  
13    ]  
14  },  
15  "pathParameters": {  
16    "proxy": "/path/to/resource"  
17  },  
18  "stageVariables": {  
19    "baz": "qux"  
20  }
```

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lab2-Hinh10

- + Tiến hành test lại API (sửa ngay trên Even JSON để test API có thể clear dữ liệu ở body tùy thích vì trong Project basic này không sử dụng) và chuẩn bị tạo cổng Gateway

Executing function: succeeded ([logs](#))

Details

```
{
  "statusCode": 200,
  "multivaluemappers": {
    "Content-Type": [
      "application/json"
    ]
  },
  "body": "[{"id": "a03c26f3-777c-4e8d-a4b-6fc51bbbe1\\", "name": "May Tinh", "price": 4300.0}, {"id": "458897ed-9d93-443a-8ef2-d4e242d59644\\", "name": "TV\\", "price": 1200.0}, {"id": "64f98a5c-d06d-4eea-b1c1-e523324bae\\", "name": "Tu Lanh\\", "price": 3400.0}, {"id": "52e1fdcc-472f-4b3d-9725-c7b32432ca3b\\", "name": "Dien Thoai\\", "price": 1500.0}]",
  "isBase64Encoded": false
}
```

Summary

Code SHA-256 uhJWF11OQ8nKvo3QQMwZmS55Byei62p6eUcxOsYuc=	Execution time 1 minute ago
Function version \$LATEST	Request ID 3889b127-c243-4960-854b-e4c846a4264a
Duration 664.75 ms	Billed duration 6086 ms
Resources configured 512 MB	Max memory used 246 MB
Init duration 5420.65 ms	

Log output

The area below shows the last 4 KB of the execution log. [Click here](#) to view the corresponding CloudWatch log group.

```
configure spring.jpa.open-in-view to disable this warning
2025-08-29T16:48:21.421Z INFO 2 --- [           main] c.a.s.l.runtime.aws.Client AwsLambda : Started AwsLambda in 4.281 seconds (process running for 5.272)
2025-08-29T16:48:21.443Z INFO 2 --- [           main] o.s.w.s.DispatcherServlet : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-08-29T16:48:21.549Z INFO 2 --- [           main] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-08-29T16:48:21.558Z INFO 2 --- [           main] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
START RequestId: 3889b127-c243-4960-854b-e4c846a4264a Version: $LATEST
2025-08-29T16:48:22.181Z INFO 2 --- [           main] c.a.s.p.internal.LambdaContainerHandler : 127.0.0.1 null- null [09/04/2015:12:34:56Z] "GET /api/products HTTP/1.1" 200 312 "-" "custom User Agent String" combined
END RequestId: 3889b127-c243-4960-854b-e4c846a4264a
REPORT RequestId: 3889b127-c243-4960-854b-e4c846a4264a Duration: 664.75 ms     Billed Duration: 6086 ms      Memory Size: 512 MB   Max Memory Used: 246 MB Init Duration: 5420.65 ms
```

Lab2-Hinh11

Bước 4: Tạo API Gateway (Rest Gateway)

- + Truy cập (aws.amazon.com/apigateway) để tạo API-Gateway

Choose an API type [info](#)

HTTP API
Build low-latency and cost-effective REST APIs with built-in features such as OAuth2, and native CORS support.
Works with the following:
Lambda, HTTP backends

WebSocket API
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.
Works with the following:
Lambda, HTTP, AWS Services

REST API
Develop a REST API where you gain complete control over the request and response along with API management capabilities.
Works with the following:
Lambda, HTTP, AWS Services

REST API Private
Create a REST API that is only accessible from within a VPC.

Import **Build** Build

Lab2-Hinh12

- + Chọn công REST-API dành cho Developer

Create REST API Info

API details

New API Create a new REST API.

Clone existing API Create a copy of an API in this AWS account.

Import API Import an API from an OpenAPI definition.

Example API Learn about API Gateway with an example API.

API name
demo-lab2

Description - optional
demo simple project java on aws

API endpoint type
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.
Regional

IP address type | Info
Select the type of IP addresses that can invoke the default endpoint for your API.

IPv4 Supports only edge-optimized and Regional API endpoint types.

Dualstack Supports all API endpoint types.

Create API

Lab2-Hinh13

Create resource

Resource details

Proxy resource Info Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example (proxy+).

Resource path /

Resource name (proxy+)

CORS (Cross Origin Resource Sharing) Info Create an OPTIONS method that allows all origins, all methods, and several common headers.

Create resource

Lab2-Hinh14

- + Nhận Create Resource và tiến hành tạo tài nguyên cho gateway-api, chú ý bật Enable CORS(cho phép chia sẻ các tài nguyên bị hạn chế) và enable Configure as proxy resource và tiến hành Create Resource.

Edit integration request

Method details

Integration type

- Lambda function
- AWS service
- VPC link
- Mock

Lambda proxy integration

Send the request to your Lambda function as a structured event.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

ap-southeast-1 arn:aws:lambda:ap-southeast-1:789922128457:function:demo-lab2

Execution role

arn:aws:iam::myAccountRole/myRole

Credential cache

Do not add caller credentials to cache key

Integration timeout | Info

By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms

29000

Request body passthrough

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lab2-Hinh15

- + Attached Lambda vào API Gateway và tiến hành nhấn Save để lưu cấu hình và tiến hành test lại API

Client certificate

No client certificates have been generated.

Test

(/proxy) - ANY method test results

Request	Latency ms	Status
/api/products	6779	200

Response body

```
[{"id": "4dd1a73b-1be5-495d-85d5-524d7359d711", "name": "May Tinh", "price": 4300.0}, {"id": "fif01cf6-b998-4eb5-9331-6def29804106", "name": "TV", "price": 1200.0}, {"id": "1a69d82d-b321-4531-8d8b-b857813864", "name": "Tu Lanh", "price": 3400.0}, {"id": "3b00e085-5824-4169-a820-85b5bc607cde", "name": "Dien Thoai", "price": 1500.0}]
```

Response headers

```
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-68b1e1db-1a796930bd00f75cd35bc6be;Parent=606119e40ce15dd4;Sampled=0;Lineage=1:cba4dac40:0"
}
```

Logs

```
Execution log for request 84e18e4f-e3cc-4dea-8076-bde8a0c1e074
Fri Aug 29 17:22:35 UTC 2025 : Starting execution for request: 84e18e4f-e3cc-4dea-8076-bde8a0c1e074
Fri Aug 29 17:22:35 UTC 2025 : HTTP Method: GET, Resource Path: /api/products
Fri Aug 29 17:22:35 UTC 2025 : Method request path: (/proxy-api/products)
Fri Aug 29 17:22:35 UTC 2025 : Method request query string: {}
Fri Aug 29 17:22:35 UTC 2025 : Method request headers: {}
Fri Aug 29 17:22:35 UTC 2025 : Endpoint request body before transformations:
Fri Aug 29 17:22:35 UTC 2025 : Endpoint request URL: https://lambda.ap-southeast-1.amazonaws.com/2015-03-31/functions/arn:aws:lambda:ap-southeast-1:789922128457:function:demo-lab2/invocations
Fri Aug 29 17:22:35 UTC 2025 : Endpoint request headers: (X-Amz-Date=20250820T172235Z, X-amzn-apigateway-api-id=v0u978uoyne, Accept=application/json, User-Agent=AmazonAPIGateway_v0978uoyne, Host=lambda.ap-southeast-1.amazonaws.com, X-Amz-Content-Sha256=e18e4f-e3cc-4dea-8076-bde8a0c1e074, X-amzn-lambda-integration-tag-84e18e4f-e3cc-4dea-8076-bde8a0c1e074)
```

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lab2-Hinh16

- + Quá trình test thành công bây giờ ta tiến hành deploy ứng dụng. Chọn Deploy API->New Stage (stage là một giai đoạn được sử dụng và tạo ra các URL của các tài nguyên API và stage name là tên của quy trình triển khai ví dụ như

prod, dev, test,... . Mỗi stage name tương ứng với một phiên bản của API và có một đường dẫn URL duy nhất).

Deploy API X

Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage

New stage

Stage name

product

i A new stage will be created with the default settings. Edit your stage settings on the [Stage page](#).

Deployment description

[Cancel](#) Deploy

Lab2-Hinh17

Nhấn Deploy và tiến hành chạy chương trình trên aws bằng đường link đính kèm

- + Invoke URL: <https://vu978wuyme.execute-api.ap-southeast-1.amazonaws.com/product/api/products> (có thể truy cập test thử)

Bước 5: Chạy chương trình.

```
[{"id": "4dd1a73b-1be5-495d-85d5-524d7359d711", "name": "May Tinh", "price": 4300}, {"id": "f1f01cf6-b998-4eb5-9331-6def29804106", "name": "TV", "price": 1200}, {"id": "1a69d82d-b321-4531-8d8b-b85703813864", "name": "Tu Lanh", "price": 3400}, {"id": "30b0e085-5824-4169-a820-85b5bc607cde", "name": "Dien Thoai", "price": 1500}]
```

Lab2-Hinh18

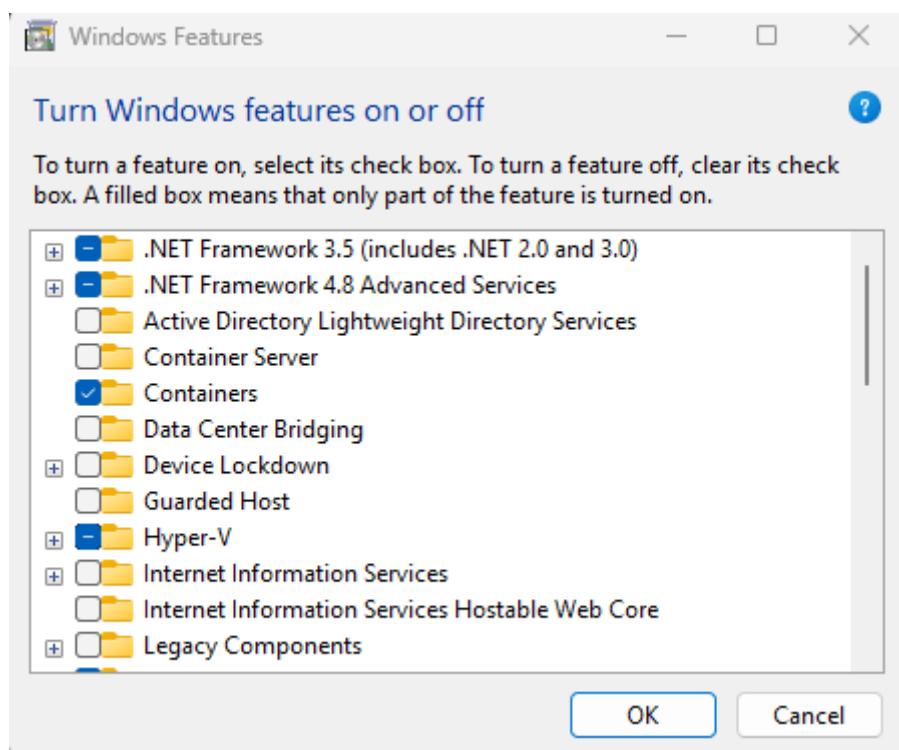
- + Sau khi hoàn thành LAB-2 có thể xóa hoặc không tùy ý, vì AWS sẽ không tính phí do cơ chế hoạt động của Lambda chỉ tính phí dựa vào yêu cầu xử lý khác với cơ chế hoạt động liên tục như một máy chủ giống như EC2(chiếm tài nguyên trên các máy chủ của AWS => mất tiền).

LAB-3

Mô tả: Sẽ tránh những thao tác lặp lại nên việc tạo những dịch vụ AWS trùng lặp sẽ được chú thích và không thực hiện lại. Trong bài này sẽ thực hiện việc demo một ứng dụng đơn giản được đóng gói lại thành các container và triển khai trên AWS ECS Fargate (chú thích: ECS Fargate là một mô hình serverless giống như Lambda nhưng nó được quản lý và chạy các container trên ECS. Loại bỏ, bỏ qua các nhu cầu quản lý máy chủ như EC2 cũng như thiết lập giống như bài LAB-1) được bao bọc trong một AWS VPC (Networking). Các container sẽ được lưu trữ trên ECR và được cập nhật thông qua AWS codepipeline. Vì bài Lab này quá dài nên sẽ không sử dụng hình ảnh mà chỉ nêu ra cách cấu hình bằng tay thông qua chữ viết đồng thời để giảm thiểu quá trình tạo trên trong bài này sẽ sử dụng đến AWS CloudFormation. Nó là một dịch vụ quản lý cơ sở hạ tầng như mã (Infrastructure as Code - IaC) từ Amazon Web Services (AWS). Nó cho phép bạn tạo và quản lý các tài nguyên AWS một cách tự động và

nhất quán bằng cách sử dụng các tệp mẫu YAML hoặc JSON được gọi là CloudFormation templates. Chỉ cần cấu hình tệp YAML và đẩy lên giảm thiểu đi quá trình khởi tạo bằng tay có thể dùng và quản lý rất dễ dàng.

Bước 1: Xây dựng một ứng dụng đơn giản, build docker-compose



Lab3-Hinh1

Setup môi trường chạy cho docker, Tích chọn Containers + Hyper-V + Windows Subsystem for Linux sau đó vào terminal chạy với vai trò quản trị viên và khởi tạo lệnh bcdedit /set hypervisorlaunchtype auto. Restart máy để cho môi trường containers được thiết lập. Ứng dụng đơn giản ở

Bước 2: Tạo AWS VPC, Tạo Subnet, Route Tables , SG, ALB

Để tối ưu cho quá trình sẽ tạo cơ bản bằng file .yml và sẽ nêu các thao tác cần thiết để tạo VPC(không đính kèm ảnh vì số lượng quá nhiều).

Đây là các bước để tạo một VPC với 2 khu vực khả dụng (Availability Zones) và 4 subnet (2 private và 2 public) trên AWS:

+ Trước đó ta tiến hành tạo 3 Security Group cho mỗi thành phần gồm: ALB-SG, PUBLIC-SG, PRIVATE-SG

+ Nhấn vào security group bên cạnh VPC tiến hành tạo PUBLIC-SG

* HTTP:Anywhere,Custom TCP:8888

+ Nhấn vào security group bên cạnh VPC tiến hành tạo PRIVATE-SG

* All traffic:Custom từ PUBLIC-SG

+ Nhập vào "Create VPC" để bắt đầu quá trình tạo VPC mới.

+ Đặt tên cho VPC của bạn và chỉ định phạm vi CIDR (Classless Inter-Domain Routing) cho VPC. CIDR 10.0.0.0/16.

+ Chọn "Enable DNS hostname" để cho phép tên miền DNS được gán cho các tài nguyên trong VPC.

+ Chọn "Enable DNS resolution" để cho phép giải quyết DNS trong VPC.

+ Nhập vào "Create" để tạo VPC.

+ Tiếp theo, tạo các subnet:

* Nhập vào "Subnets" ở bên cạnh VPC mà bạn vừa tạo.

* Nhập vào "Create subnet" để tạo subnet đầu tiên.

* Chọn VPC mà bạn đã tạo trước đó.

* Chọn khu vực khả dụng (Availability Zone) đầu tiên.

- * Đặt tên cho subnet và chỉ định phạm vi CIDR cho subnet. Sử dụng CIDR 10.0.100.0/24, 10.0.101.0/24 cho 2 subnet private và 10.0.0.0/24, 10.0.1.0/24 cho 2 subnet public lựa chọn 1 CIDR trên .
 - * Chọn "Create".
 - * Tiếp tục tạo thêm các subnet khác với khu vực khả dụng còn lại lặp lại các bước trên.
- + Tiếp theo, tạo 2 route table: 1 rtb-public và 1 rtb-private
- * Nhập vào "Route Tables" ở bên cạnh VPC.
 - * Nhấp vào "Create route table".
 - * Đính kèm VPC mà bạn đã tạo trước đó.
 - * Chọn "Create".
 - * Lặp lại các bước trên để tạo thêm một route table khác.
- + Gán các subnet vào route table:
- * Nhập vào "Subnet Associations" của route table đầu tiên.
 - * Chọn subnet đầu tiên và nhấp vào "Add subnet association" (rtb-public tương ứng với 2 subnet public và rtb-private tương ứng với 2 subnet private)
 - * Lặp lại các bước để gán các subnet còn lại vào các route table.
- + Tạo các Internet Gateway (IGW):
- * Nhập vào "Internet Gateways" ở bên cạnh VPC.
 - * Nhấp vào "Create internet gateway".
 - * Đặt tên cho Internet Gateway và nhập vào "Create".
- + Attached Internet Gateway vào VPC:

- * Chọn Internet Gateway mà bạn vừa tạo.
 - * Nhập vào "Actions" và chọn "Attach to VPC".
 - * Chọn VPC mà bạn đã tạo trước đó và nhấp vào "Attach".
- + Tạo các route để kết nối Internet Gateway với các subnet:
- * Nhập vào "Route Tables" ở bên cạnh VPC.
 - * Chọn route table đầu tiên.
 - * Nhập vào "Routes" và sau đó "Edit routes".
 - * Thêm một route mới với destination là 0.0.0.0/0 và target là Internet Gateway mà bạn đã attached vào VPC tương tự tiếp tục.
- + Bây giờ, các subnet được attached vào route table đầu tiên sẽ trở thành subnet public, trong khi các subnet gắn vào route table thứ hai sẽ trở thành subnet private.
- Qua các bước trên, bạn đã tạo thành công một VPC với 2 khu vực khả dụng và 4 subnet (2 private và 2 public). Bây giờ tiếp tục tạo NAT gateway cho private subnet kết nối ra ngoài (2 cổng NAT ở 2 subnet public):
- *Nhập vào "NAT Gateways" ở bên cạnh VPC mà bạn đã tạo trước đó.
 - *Nhập vào "Create NAT Gateway" để bắt đầu quá trình tạo cổng NAT.
 - *Trong trang "Create NAT Gateway", chọn subnet public mà bạn muốn đặt cổng NAT.
 - *Chọn Elastic IP (EIP) để gán cho cổng NAT. Nếu bạn chưa có EIP, bạn có thể tạo mới bằng cách nhập vào " Allocate Elastic IP" .
 - *Nhập vào "Create NAT Gateway" để tạo cổng NAT.
- +Đợi quá trình tạo cổng NAT hoàn thành. Sau khi tạo xong, cổng NAT sẽ có trạng thái "available".

+Tiếp theo, bạn cần cấu hình route table để định tuyến giao thông từ subnet private qua cổng NAT:

- * Nhập vào "Route Tables" ở bên cạnh VPC.
- * Chọn route table của subnet private mà bạn muốn định tuyến.
- * Nhập vào "Routes" và sau đó "Edit routes".
- * Thêm một route mới với destination là 0.0.0.0/0 và target là cổng NAT thứ 1 mà bạn đã tạo.
- * Lặp lại các bước route cho subnet private thứ 2 còn lại ứng với cổng NAT thứ 2.

+Bây giờ, các subnet private đã được cấu hình để sử dụng cổng NAT trong subnet public để kết nối ra ngoài.

Chú ý rằng khi sử dụng cổng NAT, bạn sẽ phải trả phí cho việc sử dụng Elastic IP và lưu lượng mạng đi qua cổng NAT. Tạo cân bằng tải cho ECS

- * Trong thanh điều hướng bên trái, chọn "Load Balancers".
- * Nhập vào "Create Load Balancer" để bắt đầu quá trình tạo ALB.
- * Chọn loại load balancer bạn muốn tạo, ví dụ như "Application Load Balancer".
- * Trong trang "Configure Load Balancer", đặt tên cho ALB và chọn VPC mà bạn đã tạo trước đó.
- * Chọn "Internet-facing" để ALB có thể được truy cập từ internet.
- * Chọn các subnet public mà bạn muốn đính kèm ALB. Đảm bảo rằng bạn chọn đúng các subnet public(2 public subnet đã tạo).
- * Tiếp theo, cấu hình các bước routing và chọn các target group để xử lý traffic.
- * Kiểm tra lại các cấu hình và nhập vào "Create" để tạo ALB.

- * Đợi quá trình tạo ALB hoàn thành. Sau khi tạo xong, ALB sẽ có trạng thái "active".
- * Bây giờ, bạn đã đính kèm ALB vào 2 subnet public. Bạn có thể cấu hình các luật routing và target group để điều phối traffic đến các máy chủ ứng dụng hoặc các dịch vụ.

Lưu ý rằng bạn cần đảm bảo rằng các máy chủ ứng dụng hoặc dịch vụ đang chạy trong các subnet public và được đăng ký vào các target group (thông thường là một con web-server hoặc web-jump) của ALB.

Bước 3: Tạo ECS Fargate.

- * Điều hướng đến dịch vụ Amazon ECS.
- * Chọn "Clusters" và nhấp vào "Create cluster".
- * Chọn loại cluster là "Networking only".
- * Đặt tên cho cluster và chọn VPC mà bạn đã tạo.
- * Chọn các subnet private mà bạn muốn đính kèm cluster.
- + Tạo task definition và service
 - * Chọn "Task Definitions" và nhấp vào "Create new Task Definition".
 - * Chọn loại network mode là "awsvpc".
 - * Đặt tên cho task definition và cấu hình các thông số khác theo yêu cầu của bạn.
 - * Chọn "Add container" để thêm container vào task definition.
 - * Đặt tên cho container và cấu hình các thông số khác như image, port, environment variables, v.v.
- * Lưu task definition.
- * Chọn "Services" và nhấp vào "Create".

- * Đặt tên cho service và chọn cluster mà bạn đã tạo.
- * Chọn task definition mà bạn đã tạo.
- * Cấu hình các thông số khác như số lượng task, cân bằng tải, v.v.
- * Chọn các subnet private mà bạn muốn đính kèm service.
- * Lưu service.

+ Kiểm tra và xác nhận

- * Kiểm tra các cấu hình và xác nhận rằng ECS Fargate đã được đính kèm vào subnet private.
- * Kiểm tra rằng service và các task đã được triển khai và hoạt động một cách ổn định trong VPC với subnet private.

Bước 5: Tạo ECR

Để tạo một Amazon Elastic Container Registry (ECR) để lưu trữ các container, bạn có thể thực hiện các bước sau:

- * Đăng nhập vào bảng điều khiển AWS và điều hướng đến dịch vụ Amazon ECR.
- * Chọn "Create repository" để tạo một lưu trữ mới.
- * Đặt tên cho lưu trữ của bạn và chọn các tùy chọn khác như encryption, scan on push, v.v. (tùy chọn).
- * Nhập vào "Create repository" để tạo lưu trữ.
- * Sau khi lưu trữ được tạo thành công, bạn sẽ nhìn thấy URL của lưu trữ. Bạn có thể sử dụng URL này để đăng nhập và quản lý lưu trữ. Và tiến hành đẩy container vào repository (có thể phát sinh lỗi không đủ role thì phải tạo IAM với user có role yêu cầu là EC2 Container Full Access)

```

Administrator: Windows PowerShell + 
PS D:\NIENLUAN\AWS-ECS-Lab3\Front-web-angular> docker tag web-angular:latest 789922128457.dkr.ecr.us-east-1.amazonaws.com/web-angular:latest
PS D:\NIENLUAN\AWS-ECS-Lab3\Front-web-angular> docker push 789922128457.dkr.ecr.us-east-1.amazonaws.com/web-angular:latest
The push refers to repository [789922128457.dkr.ecr.us-east-1.amazonaws.com/web-angular]
483ef251637: Pushed
483ef251637: Pushed
c9eb2f2d86f82: Pushed
c9eb2f2d86f82: Pushed
61b4b3d06670: Pushed
445e2bb57fb: Pushed
9a9fbae99cb7: Pushed
a992fbc61ecc: Pushed
89303a2a5349: Pushed
bb07ac6a129: Pushed
7aa8a6741c18: Pushed
latest: digest: sha256:374cefadb53b6567a305ffd71430e6d46168e7105010f6955b40e3e4be9143b size: 856
PS D:\NIENLUAN\AWS-ECS-Lab3\Front-web-angular> cd :\NIENLUAN\AWS-ECS-Lab3\gateway
PS D:\NIENLUAN\AWS-ECS-Lab3\gateway> docker build -t gateway .
[+] Building 2s (9/9) FINISHED
   => [internal] load build context from Dockerfile
   => [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine
   => [auth] Library/docker.io:pull token for registry-1.docker.io
   => [internal] load .dockerignore
   => [internal] transfering context: 2B
   => [internal] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c96ea7485154238902f2f25e9
   => [internal] COPY --chown root:root / /Library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c96ea7485154238902f2f25e9
   => [internal] load build context
   => [internal] transfering context: 32.03MB
   => [internal] transfering context: 32.03MB
   => CACHED [2/3] RUN addgroup -S demo && adduser -S -G demo demo
   => [3/3] ADD target/gateway+.jar app.jar
   => exporting to image
   => exporting layers
   => exporting manifest sha256:de35e1c79145af993a0ac0404189163c61f2725b032e1b78c8a730cae857b779e
   => exporting config sha256:8809f0eb03852324cdcc2bed8307016aa07839caa7f1ac651b921386900
   =>> exporting attestation manifest sha256:52b3b49a585d65e5c01ab796369ca6ea2780164648da7549eaf4dd832b00fb86d2
   =>> exporting manifest list sha256:a3d251e78526f1f1feffcc8ba266fbe6fe9e29f7898dec1d2ed3dc1849e0dff29
   =>> naming to docker.io/library/gateway:latest
   => exporting layers
PS D:\NIENLUAN\AWS-ECS-Lab3\gateway> docker tag gateway:latest 789922128457.dkr.ecr.us-east-1.amazonaws.com/gateway:latest
PS D:\NIENLUAN\AWS-ECS-Lab3\gateway> docker push 789922128457.dkr.ecr.us-east-1.amazonaws.com/gateway:latest
The push refers to repository [789922128457.dkr.ecr.us-east-1.amazonaws.com/gateway]
75696f68a22: Pushed
d8d715783808: Pushed
53c94661254d: Pushed
2232b523838: Pushed
ba022b23838: Pushed
5803a6ab374: Pushed
latest: digest: sha256:a3d251e78526f1f1feffcc8ba266fbe6fe9e29f7898dec1d2ed3dc1849e0dff29 size: 856
PS D:\NIENLUAN\AWS-ECS-Lab3\gateway>

```

Lab3-Hinh2

Thực hiện push từng image lên ECR(check view push commands). Hoặc sử dụng start.sh để push tự động bằng code với lệnh start run được đi kèm trong link git Lab3(hiệu cơ ché hoạt động rồi sử dụng lại).

```

dockercfg.yml start.sh
Would you like to install shellcheck to verify your shell scripts? Install No
1 > !#/bin/bash
2 # Tạo repo trước khi push và lấy link repo
3 DOCKER_REPO_URI=789922128457.dkr.ecr.ap-southeast-1.amazonaws.com
4
5
6 function build() {
7     echo "Building service"
8     docker-compose build .
9 }
10
11 function push() {
12     #DOWNLOAD AWS CLI VÀ TẠO IAM (ROLE=AWSSEC2CONTAINERFULLACCESS).
13     echo "Push images"
14     $(aws ecr get-login --no-include-email --region ap-southeast-1)
15
16     docker tag auth-jwt-service:latest 789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/auth-jwt-service:latest
17     docker tag customer-service:latest 789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/customer-service:latest
18     docker tag gateway:latest 789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/gateway:latest
19     docker tag redis:latest 789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/redis:latest
20     docker tag sec-service:latest 789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/sec-service:latest
21     docker tag web-angular:latest 789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/web-angular:latest
22
23     docker push $DOCKER_REPO_URI/auth-jwt-service:latest
24     docker push $DOCKER_REPO_URI/customer-service:latest
25     docker push $DOCKER_REPO_URI/gateway:latest
26     docker push $DOCKER_REPO_URI/redis:latest
27     docker push $DOCKER_REPO_URI/sec-service:latest
28     docker push $DOCKER_REPO_URI/web-angular:latest
29 }
30
31 function options() {
32     echo "./start build -> Build docker images"
33     echo "./start push -> Push images to docker registry. Plz configure repo before push."
34 }
35

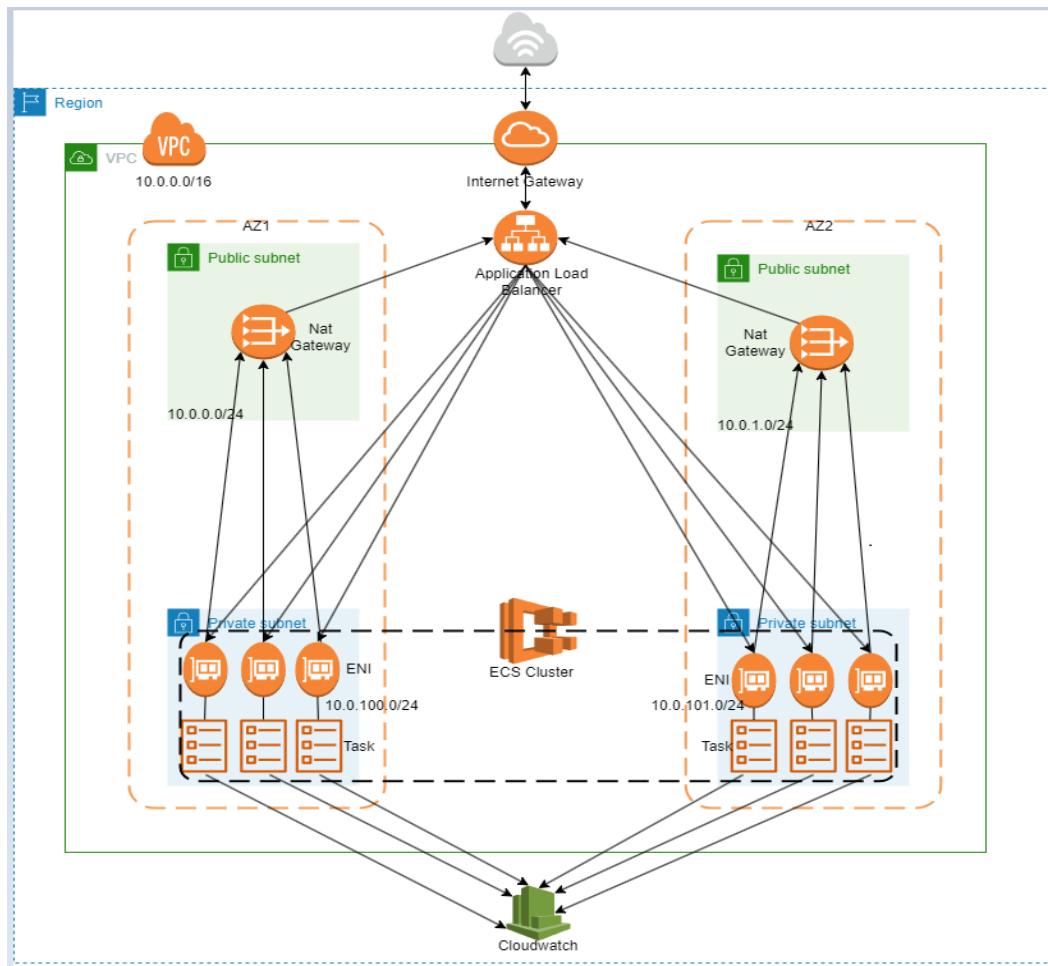
```

Lab3-Hinh3

Sau khi hoàn tất chuyển đến bước tiếp theo.

Bước 7: Áp dụng AWS CloudFormation

Sau khi đã mô tả quá trình tạo rất xong và thực hiện đầy đủ các bước trên thì ta thu được mô hình như bên dưới:



Lab3-Hinh4

- + Áp dụng AWS CloudFormation để xây dựng mô hình trên mà không cần thao tác tay quá nhiều:

*Chuẩn bị tệp mẫu CloudFormation: Tạo tệp YAML hoặc JSON mô tả cấu trúc và các tài nguyên AWS mà bạn muốn tạo (ở đây sẽ thực hành với file YAML).

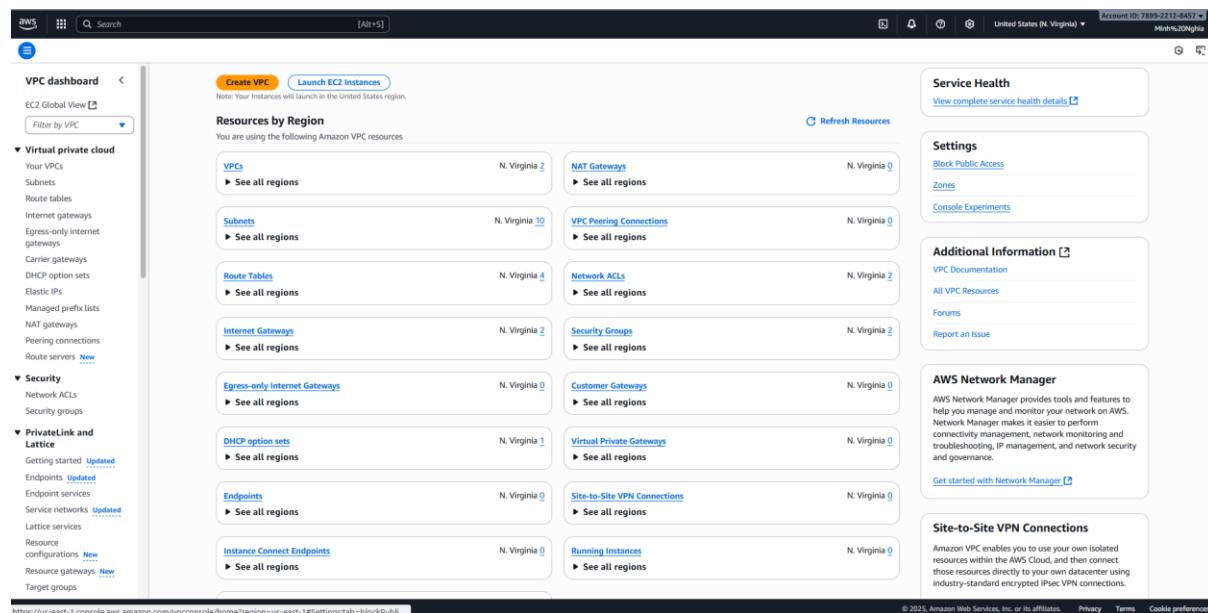
*Create stack: Sử dụng AWS Management Console, AWS CLI hoặc AWS SDKs để tạo stack từ tệp mẫu CloudFormation. Khi tạo stack, bạn cần chỉ định tên stack và đường dẫn đến tệp mẫu.

*Xem trạng thái stack: Sau khi tạo stack, bạn có thể xem trạng thái của nó để kiểm tra xem quá trình tạo đã thành công hay không. Bạn có thể sử dụng giao diện người dùng AWS hoặc AWS CLI để kiểm tra.

*Quản lý stack: Bạn có thể thực hiện các thao tác quản lý stack như cập nhật, xóa hoặc xem lại stack. Nếu bạn muốn thay đổi cấu trúc hoặc tài nguyên của mô hình, bạn có thể chỉnh sửa tệp mẫu CloudFormation và cập nhật stack.

*Xem kết quả: Sau khi stack hoàn thành, bạn có thể xem các tài nguyên đã được tạo và xác nhận xem mô hình đã được triển khai thành công hay không.

Đổi một region khác. Mọi thứ đều clean khi đổi region vì phiên bản free sẽ có một số giới hạn(cổng NAT, IGW, số VPC giới hạn,...) để tránh quá trình xảy ra lỗi không cần thiết do hết tài nguyên khởi tạo. Nhớ cài đặt AWS CLI để sử dụng ECR.



thực hiện đúng các bước trên new stack Upload a template file YAML (stack name với cú pháp ...-Lab3) và tham số Parameters là msdemo (microservice demo) hoặc tên thêm số bất kỳ mà bạn muốn sau đó nhấn next liên tục bỏ qua các role và policy ngoài lề khác để tạo stack. Bước cuối là check lại quá trình khởi tạo trong stacks. Nếu quá trình bị hủy bỏ nó sẽ ROLLBACK Transaction và hủy bỏ quá trình khởi tạo (debug và check lại YAML).

Logical ID	Physical ID	Type	Status	Module
GatewayAttachment	iGW vpc-070f6710662bb147	AWS::EC2::VPCGatewayAttachment	CREATE_COMPLETE	-
InternetGateway	igw-08118e60eef06dd86	AWS::EC2::InternetGateway	CREATE_COMPLETE	-
PrivateRouteTable	rtb-0a7b5f4bca559e47f	AWS::EC2::RouteTable	CREATE_COMPLETE	-
PrivateSubnetOne	subnet-0a494b0984fec23	AWS::EC2::Subnet	CREATE_COMPLETE	-
PrivateSubnetOneRouteTableAssociation	rtbasoc-0d0159afac7fb27b9	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-
PrivateSubnetTwo	subnet-00f4280b6b4219ea9	AWS::EC2::Subnet	CREATE_COMPLETE	-
PrivateSubnetTwoRouteTableAssociation	rtbasoc-083c1a0ffce628d5	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-
PublicRoute	rtb-0c2adce4db9354071 0.0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE	-
PublicRouteTable	rtb-0c2adce4db9354071	AWS::EC2::RouteTable	CREATE_COMPLETE	-
PublicSubnetOne	subnet-0904fe3344093b44	AWS::EC2::Subnet	CREATE_COMPLETE	-

Lab3-Hinh6

Rồi tới đi đến AWS VPC để kiểm tra quá trình khởi tạo.

Quá trình khởi tạo đã thành công tiếp tục tạo những stack khác cho đến khi upload hết tất cả các cấu hình YAML bạn đã tạo ra.

The screenshot shows the AWS VPC dashboard with the following details:

- Route tables (1/4) Info**: A table listing route tables:

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
rtb-0cae319493bb25dd	-	-	-	Yes	vpc-0428f56b51225fa38	789922128457
msdemo-PrivateRouteTable	rtb-06eca77be03ab2df2	2 subnets	-	No	vpc-0900605d2ek85921 msd...	789922128457
rtb-08e56c5a50b7fb0e2	-	-	-	Yes	vpc-0900605d2ek85921 msd...	789922128457
msdemo-PublicRouteTable	rtb-03572208b7e706a16	2 subnets	-	No	vpc-0900605d2ek85921 msd...	789922128457
- msdemo-PrivateRouteTable / msdemo-PrivateRouteTable**: A detailed view of the selected route table:

Routes (1)					
Details		Routes		Subnet associations	
Destination	Target	Status	Propagated	Route Origin	
10.0.0.0/16	local	Active	No	Create Route Table	

Lab3-Hinh7

Tiếp tục tạo YAML công NAT đính kèm vào public subnet và tạo liên kết với PrivateRouteTable để private subnet có thể truy cập ra bên ngoài từ công NAT. Sau khi quá trình hoàn tất ta tiếp tục tạo ALB bằng file YAML. Như vậy ta đã hoàn thành cấu hình VPC với (2 private subnet, 2 public subnet, 1 công nat được đặt ở subnet public và router với private subnet, 4 subnet đã được association với 4 rt và 1 alb để cân bằng tải cho 2 private subnet). Tiếp tục ta khởi tạo ECS cluster và các task definition.

Bây giờ muốn ôn lại khái niệm task, task definition, service (trong bài này sẽ run task và còn run service thì tương tự).

+Task Definition (Định nghĩa nhiệm vụ): Một task definition là bản mô tả về cách một nhóm các container (hoặc một container đơn lẻ) sẽ được chạy trong một môi trường ECS. Nó xác định các thông tin như image của container, cấu hình mạng, tài nguyên (CPU, bộ nhớ), các biến môi trường và nhiều thông số khác.

- Task definition là bản thiết kế của các task (nhiệm vụ) trong ECS.

+Task (Nhiệm vụ):

- Một task là một phiên bản cụ thể của task definition.
- Khi bạn chạy một task, ECS sẽ tạo ra một (hoặc nhiều) container dựa trên task definition tương ứng. nghĩa là chạy 1 lần 1 nhiệm vụ và lỗi sẽ tự stop

+Service:

- Một service là một khối xây dựng trên task definition và task.
- Nó cho phép bạn quản lý và duy trì một số lượng cụ thể các phiên bản của các task trong ECS cluster.
- Service đảm bảo rằng một số lượng nhất định các phiên bản của task đang chạy liên tục và tự động khởi động lại các task bị lỗi hoặc đã kết thúc.
- Service cũng có thể được cấu hình để cân bằng tải các phiên bản task trên các instance khác nhau trong cluster.

Tóm lại nếu có thiết kế task definition thì việc chạy Task hay Service chỉ là một vấn đề nhỏ.

Ta thực hiện create tất cả các task definition cần thiết để chạy ECS và tiến hành run task. Sau khi hoàn tất ta vào task definition để kiểm tra lại

Task definition	Status of last revision
auth-jwt-service	ACTIVE
customer-service	ACTIVE
gateway	ACTIVE
sec-service	ACTIVE
web-angular	ACTIVE

Lab3-Hinh8

như vậy đã hoàn thành khởi tạo các task definition cần thiết theo cấu hình YAML. Tuy nhiên ở đây ở task sec-service thì sec-service lại tồn tại 2 container nên ta phải chỉnh sửa lại. click vào sec-service và create new revision => add more container.

The screenshot shows the AWS Elastic Container Service (ECS) Task Definitions page. In the center, there is a table listing task definitions. One row is selected, labeled 'sec-service:1'. The table includes columns for 'Task definition: revision' and 'Status'. The status for 'sec-service:1' is shown as 'ACTIVE'. At the top right of the table, there is a button labeled 'Create new revision' with a dropdown arrow. To the left of the table, there is a sidebar with various AWS services like Clusters, Namespaces, Task definitions, and ECR repositories.

Lab3-Hinh9

Ta tiến hành add một container redis với cấu hình cơ bản host port 6379 và container port 6379 với đường dẫn là ECR lưu URI chứa container redis mà bạn đã push lên từ CLI như hình dưới.

The screenshot shows the 'Container - 2' configuration page for a task definition. It has sections for 'Container details', 'Image URI', 'Essential container', 'Private registry', and 'Port mappings'. Under 'Container details', the name is 'redis' and the image URI is '789922128457.dkr.ecr.ap-southeast-1.amazonaws.com/redis'. Under 'Port mappings', there is one mapping for port 6379 with TCP protocol and HTTP app protocol. A 'Remove' button is visible at the top right of the form.

Lab3-Hinh10

Sau khi chỉnh sửa xong nhớ vào JSON để thêm cấu hình dependson redis và kiểm tra lại

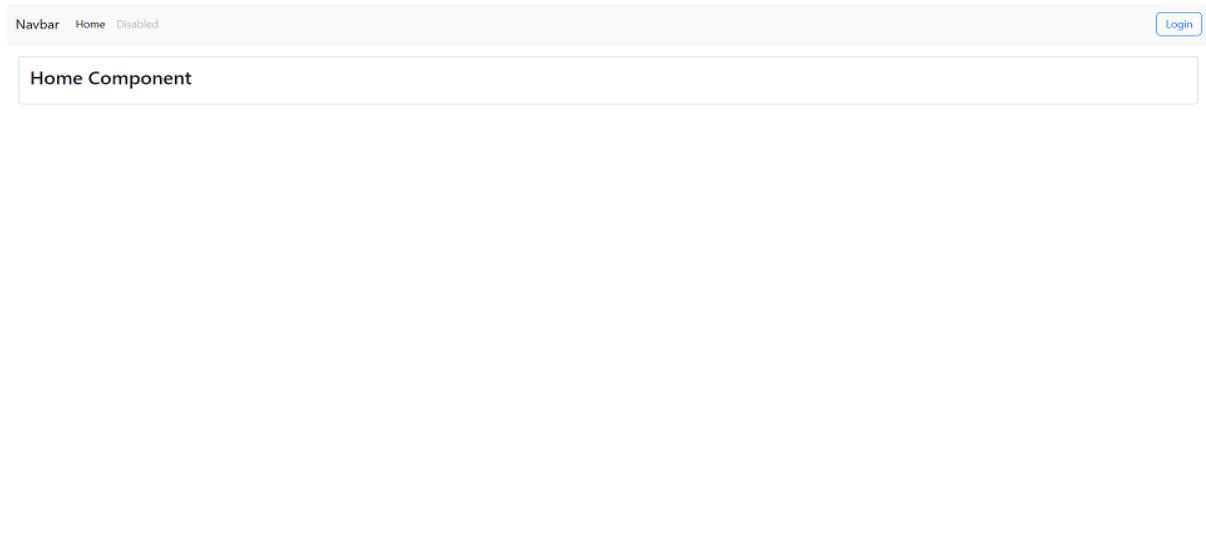
Quá trình khởi tạo task definition như vậy coi như đã hoàn tất. Tiếp tục tạo task lưu ý Networking cấu hình như dưới. Vì các container được đặt trong private subnet nên ta không IP public network

và khởi tạo các task tuân tự Family chọn đúng task cần chạy, revision chọn bản vá gần nhất, Desired tasks vì giới hạn của Free nên sẽ mặc định 1

The screenshot shows the AWS ECS console interface. On the left, there's a sidebar with various AWS services like ECR, Batch, and Proton. The main area is titled 'Amazon Elastic Container Service > Clusters > msdemo-Cluster > Tasks'. It displays the 'Cluster overview' with ARN (msdemo-Cluster), Status (Active), CloudWatch monitoring (Default), and Registered container instances (None). Below this, the 'Tasks' tab is selected, showing a table of 5 tasks. The table columns include Task, Last status, Desired st..., Task defi..., Revision, Health sta..., Started at, Container instan..., Launch type, CPU, and Mem. The tasks are: gateway (revision 2), sec-service (revision 6), auth-int-se... (revision 3), web-angular (revision 5), and customer-s... (revision 6). All tasks are running on FARGATE with 1 vCPU and 5 G memory.

Lab3-Hinh11

Quá trình run các task diễn ra thành công, có thể vào log để coi các log thông báo bây giờ ta sẽ test các task qua đường dẫn elb ở đây ta đã chạy các task theo dạng riêng lẻ chứ không quản lý các task dựa vào service(1 service gồm nhiều task nếu chạy quá nhiều task riêng lẻ sẽ khó quản lý nên thông thường chạy service sẽ tối ưu hơn).



Lab3-Hinh12

tiếp tục test các chức năng cơ bản thôi.

ID	Name	
Customer1	cust1@gmail.com	<button>Delete</button>
Customer2	cust2@gmail.com	<button>Delete</button>
Customer3	cust3@gmail.com	<button>Delete</button>
Customer4	cust4@gmail.com	<button>Delete</button>
Customer5	cust5@gmail.com	<button>Delete</button>
Customer6	cust6@gmail.com	<button>Delete</button>
Customer7	cust7@gmail.com	<button>Delete</button>
Customer8	cust8@gmail.com	<button>Delete</button>
Customer9	cust9@gmail.com	<button>Delete</button>
Customer10	cust10@gmail.com	<button>Delete</button>

Lab3-Hinh13

Các chức năng hoạt động rất ổn và trùng khớp với với docker được khởi tạo và chạy trên localhost như ban đầu (lưu ý: đường dẫn cú pháp để không bị lỗi 500). Sau khi hoàn thành ta tiến hành stop các task và kết thúc bài Lab 3 về VPC-ECS tại đây.

TỔNG KẾT

Qua bài thực hành em tóm lược

EC2 (Elastic Compute Cloud) là một dịch vụ cung cấp các máy ảo đàm hồi trên nền tảng đám mây của Amazon Web Services (AWS). Chỉ tạo nếu khi yêu cầu cần một máy chủ được kiểm soát cấu hình chặt chẽ.

Lambda là một dịch vụ tính toán mà không cần quản lý máy chủ. Ưu tiên cho sự đơn giản hóa quá trình.

ECS (Elastic Container Service) là một dịch vụ quản lý các container trên AWS. Tìm hiểu về cách triển khai các ứng dụng trong các container sử dụng ECS, cách cấu hình và quản lý các cluster, task definition và service trong ECS. Tối ưu trong việc quản lý các dịch vụ vi mô dưới dạng các container và ECR là nơi lưu trữ container.

VPC (Virtual Private Cloud) là một dịch vụ cung cấp môi trường mạng ảo riêng tư trên AWS là một phần không thiếu của mạng riêng.

Sau kỳ khóa luận hè em đã thu được kiến thức về cách triển khai và quản lý các dịch vụ EC2, Lambda và ECS trong một môi trường mạng ảo riêng tư (VPC). Điều đó giúp em hiểu rõ hơn về cách làm việc của các dịch vụ đám mây và cách chúng tương tác với nhau. Em đã học được cách tạo và quản lý các máy ảo EC2, triển khai các hàm Lambda và quản lý các container trong ECS. Em cũng đã nắm vững cách cấu hình và quản lý mạng VPC, bao gồm các subnet, routing, security groups và network ACLs. Việc thu được kiến thức này giúp em có khả năng triển khai và quản lý các ứng dụng đám mây một cách hiệu quả và an toàn với những bước cơ bản đầu tiên. Em có thể xây dựng các hệ thống phân tán, linh hoạt và có khả năng mở rộng sử dụng các dịch vụ EC2, Lambda và ECS trong môi trường mạng VPC. Đồng thời, em cũng hiểu rõ hơn về khái niệm và cách làm việc của các dịch vụ đám mây AWS, giúp em có thể nắm bắt và áp dụng những công nghệ mới và tiên tiến trong lĩnh vực công nghệ thông tin.

LINK TÀI LIỆU THAM KHẢO:

(<https://github.com/minhnghia0910/NIENLUAN/blob/main/Book-Dien-Toan-Dam-May.pdf>)

(<https://aws.amazon.com/documentation/>)

(<https://cloud.ibm.com/docs>)

(<https://docs.oracle.com/en-us/iaas/>)