

AWS Lambda

Thế giới Serverless trong tầm tay bạn

It's a serverless world



Function as a Service



Cloud Computing



Serverless Architecture

Serverless là gì?

Serverless là một mô hình phát triển mới, nơi các nhà phát triển không cần phải quản lý máy chủ. Thay vào đó, họ chỉ cần triển khai mã (code), cụ thể là các hàm (functions).



Mã hóa chức năng

Triển khai chỉ cần mã, không cần quản lý máy chủ



Mở rộng tự động

Tự động mở rộng dựa trên nhu cầu



Chi phí linh hoạt

Thanh toán theo số lượng yêu cầu và thời gian tính toán



Server-based

- ✓ Quản lý máy chủ
- ✓ Giới hạn bởi RAM và CPU
- ✓ Chạy liên tục
- ✓ Can thiệp để thêm/bớt máy chủ

VS



Serverless

- ✓ Không cần quản lý máy chủ
- ✓ Giới hạn bởi thời gian thực thi
- ✓ Chạy theo yêu cầu (on-demand)
- ✓ Tự động mở rộng



Lưu ý: "Serverless" không có nghĩa là không có máy chủ. Nó chỉ có nghĩa là bạn không cần phải quản lý, cấp phát hoặc nhìn thấy chúng. AWS sẽ chịu trách nhiệm quản lý cơ sở hạ tầng bên dưới.

Các dịch vụ Serverless trong AWS

AWS cung cấp một hệ sinh thái serverless phong phú, bao gồm nhiều dịch vụ tích hợp chặt chẽ với AWS Lambda để xây dựng các ứng dụng mạnh mẽ.



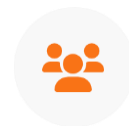
AWS Lambda

Dịch vụ điện toán cốt lõi, chạy mã của bạn mà không cần cấp phát hoặc quản lý máy chủ.



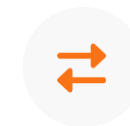
DynamoDB

Cơ sở dữ liệu NoSQL được quản lý hoàn toàn, có khả năng mở rộng cao.



AWS Cognito

Quản lý danh tính và truy cập người dùng cho ứng dụng di động và web.



AWS API Gateway

Tạo, xuất bản, bảo mật và quản lý các API RESTful và WebSocket.



Amazon S3

Dịch vụ lưu trữ đối tượng có khả năng mở rộng, độ bền cao.



AWS SNS & SQS

Dịch vụ nhắn tin và hàng đợi tin nhắn để xây dựng các ứng dụng phân tán.



AWS Kinesis Data Firehose

Dịch vụ thu thập, chuyển đổi và tải dữ liệu streaming vào các kho dữ liệu.



Kiến trúc Serverless

Tích hợp các dịch vụ để xây dựng ứng dụng hoàn chỉnh





Lambda


DynamoDB

API Gateway

So sánh EC2 và Lambda

EC2 và Lambda represent hai mô hình khác nhau trong điện toán đám mây AWS. Dưới đây là sự khác biệt chính giữa chúng:

Đặc điểm	EC2 (Máy ảo trong đám mây)	Lambda (Hàm ảo)
 Quản lý máy chủ	cần quản lý máy chủ ảo	không cần quản lý máy chủ
 Giới hạn	giới hạn bởi RAM và CPU	giới hạn bởi thời gian thực thi (thực thi ngắn)
 Trạng thái	chạy liên tục	chạy theo yêu cầu (on-demand)
 Mở rộng	cần can thiệp để thêm/bớt máy chủ (scaling means intervention)	tự động mở rộng (scaling is automated)

 **Kết luận:** Lambda mang lại nhiều lợi ích đáng kể so với EC2, bao gồm việc không cần quản lý máy chủ, mở rộng tự động và định giá theo yêu cầu và thời gian thực thi.

Lợi ích của AWS Lambda



Định giá dễ dàng

Thanh toán theo số lượng yêu cầu và thời gian tính toán.

Tặng miễn phí bao gồm 1.000.000 yêu cầu và 400.000 GB-giây thời gian tính toán.



Tích hợp

Tích hợp với toàn bộ bộ dịch vụ AWS.

Hỗ trợ nhiều ngôn ngữ lập trình khác nhau.



Giám sát dễ dàng

Thông qua AWS CloudWatch.

Theo dõi hiệu suất và nhật ký dễ dàng.



Mở rộng tài nguyên

Có thể cấp phát đến 10GB RAM cho mỗi hàm.

Việc tăng RAM cũng sẽ cải thiện hiệu suất CPU và mạng.



Thanh toán theo sử dụng

Chỉ trả tiền cho thời gian thực thi của hàm.

Rất tiết kiệm chi phí cho các ứng dụng nhỏ và trung bình.



Không cần quản lý máy chủ

AWS sẽ chịu trách nhiệm quản lý cơ sở hạ tầng.

Tập trung vào mã hóa chức năng thay vì cấu hình máy chủ.

AWS Lambda mang lại nhiều lợi ích đáng kể so với các mô hình điện toán truyền thống

Hỗ trợ ngôn ngữ lập trình trong AWS Lambda



Node.js

JavaScript



Python

Ngôn ngữ lập trình phổ biến



Java

Ứng dụng Java



C#

.NET Core / Powershell



Ruby

Ứng dụng Ruby



Custom Runtime

Rust, Golang, v.v.

Lambda Container Image

- ✓ Đóng gói ứng dụng Lambda dưới dạng hình ảnh container
- ✓ Hình ảnh container phải triển khai Lambda Runtime API
- ✓ For running arbitrary Docker images, ECS / Fargate is preferred

Tích integration AWS Lambda với các dịch vụ khác

Các dịch vụ tích integration chính



Amazon S3



Amazon API Gateway



Amazon CloudWatch



Amazon CloudFront



Amazon Simple Email Service



Amazon SNS & SQS

Ví dụ ứng dụng serverless



Tạo ảnh thu nhỏ (Thumbnail) Serverless



1. Tải hình ảnh lên S3



2. S3 kích hoạt
Lambda



3. Lambda tạo
thumbnail



4. Lưu thumbnail vào
S3



Trường hợp sử dụng: Tự động tạo ảnh thu nhỏ khi người dùng tải lên hình ảnh, cải thiện tốc độ tải trang và tiết kiệm băng thông.



CRON Job Serverless



1. Lên lịch
CloudWatch



2. Lambda thực thi



3. Xử lý định kỳ



4. Kết quả



Trường hợp sử dụng: Thay vì sử dụng máy chủ để chạy các tác vụ định kỳ, bạn có thể sử dụng

Mô hình định giá AWS Lambda

Hai thành phần định giá



Thanh toán theo số lượng yêu cầu

1.000.000 yêu cầu đầu tiên là miễn phí
\$0.20 cho mỗi 1 triệu yêu cầu tiếp theo



Thanh toán theo thời gian thực thi

400.000 GB-giây thời gian tính toán mỗi tháng là MIỄN PHÍ
\$1.00 cho 600.000 GB-giây tiếp theo



Tặng miễn phí

Bao gồm 1.000.000 yêu cầu AWS Lambda và 400.000 GB-giây thời gian tính toán mỗi tháng



Ví dụ về tính toán chi phí

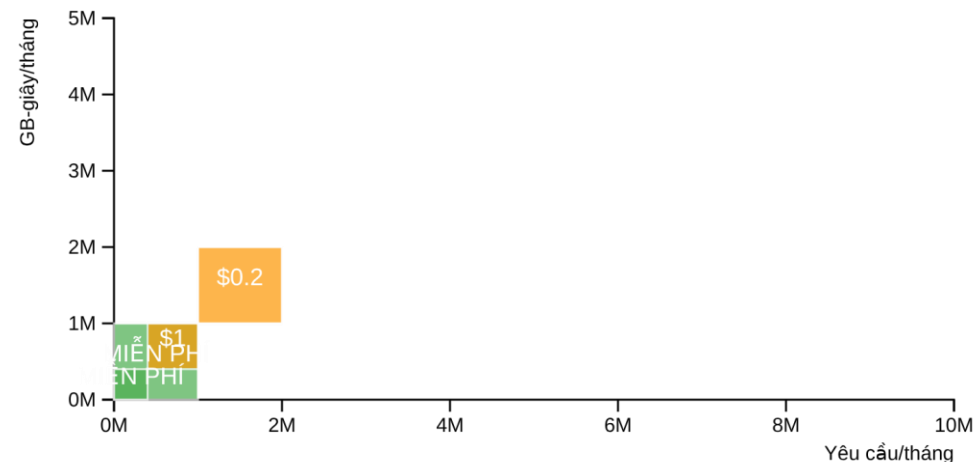
Ví dụ 1: Backend ứng dụng di động - 3 triệu yêu cầu/tháng, 120ms mỗi yêu cầu

Ví dụ 2: Làm giàu dữ liệu telemetry streaming - 7.44 triệu tin nhắn/tháng

Ví dụ 3: Thực hiện ML trên các yêu cầu hỗ trợ khách hàng - 6 triệu yêu cầu/tháng



Bảng giá AWS Lambda



Ví dụ 1: Backend ứng dụng di động

3 triệu yêu cầu/tháng
Chi phí yêu cầu: \$0.40
Chi phí tính toán: \$2.33
Tổng: \$2.73/tháng

Ví dụ 2: Telemetry streaming

7.44 triệu tin nhắn/tháng
Chi phí yêu cầu: \$1.49
Chi phí tính toán: \$248.00
Tổng: \$249.49/tháng

Lời gọi đồng bộ (Synchronous Invocations)

Lời gọi đồng bộ là kiểu gọi mà kết quả được trả về ngay lập tức. Việc xử lý lỗi (thử lại, exponential backoff, v.v.) phải được thực hiện ở phía client.



Client



AWS Lambda



Resource



Lưu ý: Khi một hàm Lambda được gọi đồng bộ, client phải chờ kết quả trả về trước khi tiếp tục thực thi. Nếu có lỗi, client phải tự xử lý (thử lại, v.v.)

Các dịch vụ kích hoạt đồng bộ:

Do người dùng kích hoạt:



Elastic Load Balancing
(Application Load Balancer)



Amazon API Gateway



Amazon CloudFront
(Lambda@Edge)



Amazon S3 Batch

Do dịch vụ kích hoạt:



Amazon Cognito



AWS Step Functions

Các dịch vụ khác:



Amazon Lex



Amazon Alexa



Amazon Kinesis
Data Firehose

Lời gọi bất đồng bộ (Asynchronous Invocations)

Lời gọi bất đồng bộ các sự kiện được đặt vào một hàng đợi sự kiện (Event Queue). Lambda sẽ cố gắng thử lại khi có lỗi.

Thời gian chờ_RETRY



3 lần thử lại tổng cộng
1 phút chờ sau lần thử đầu tiên, sau đó chờ 2 phút

Xử lý lỗi



Đảm bảo quá trình xử lý là idempotent (trong trường hợp thử lại)
Nếu hàm được thử lại, bạn sẽ thấy các mục nhật ký trùng lặp trong CloudWatch Logs

DLQ (Dead Letter Queue)

Có thể định nghĩa một DLQ (dead-letter queue) – SNS hoặc SQS – cho các quá trình xử lý thất bại



Lợi ích

Lời gọi bất đồng bộ cho phép bạn tăng tốc độ xử lý nếu bạn không cần chờ kết quả (ví dụ: bạn cần xử lý 1000 tệp).

Quy trình gọi bất đồng bộ



Dịch vụ



Hàng đợi



Lambda

Dịch vụ AWS sử dụng bất đồng bộ



Amazon S3



Amazon SNS



CloudWatch Events



EventBridge



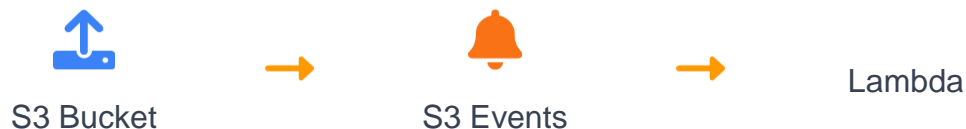
CodeCommit



CodePipeline

Thông báo sự kiện S3 và Event Source Mapping

S3 Event Notifications



Loại sự kiện

S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...



Lọc tên đối tượng

Có thể lọc theo tên đối tượng (*.jpg)



Thời gian giao sự kiện

Thông báo sự kiện S3 thường được gửi trong vài giây nhưng đôi khi có thể mất một phút hoặc lâu hơn



Lưu ý: Nếu hai thao tác ghi được thực hiện trên một đối tượng không có phiên bản cùng một lúc, có thể chỉ có một thông báo sự kiện được gửi. Để đảm bảo rằng một thông báo sự kiện được gửi cho mỗi thao tác ghi thành công, bạn có thể bật tính năng lập phiên bản (versioning) trên bucket của mình.

Event Source Mapping



Các nguồn dữ liệu

Kinesis Data Streams, SQS & SQS FIFO queue, DynamoDB Streams



Cách hoạt động

Hàm Lambda của bạn được kích hoạt đồng bộ (synchronously)



Điểm chung

Các bản ghi cần được thăm dò (polled) từ nguồn



Lưu ý: Khi bạn sử dụng Event Source Mapping, Lambda sẽ tự động xử lý việc thăm dò dữ liệu từ các nguồn này và gọi hàm Lambda của bạn với từng batch sự kiện. Điều này cho phép bạn xây dựng các ứng dụng phân tán mạnh mẽ mà không cần phải quản lý phức tạp của việc kết nối các dịch vụ.

Đối tượng Event và Context trong Lambda



Đối tượng Event

- ✓ Tài liệu định dạng JSON chứa dữ liệu để hàm xử lý
- ✓ Chứa thông tin từ dịch vụ kích hoạt (ví dụ: EventBridge, tùy chỉnh, ...)
- ✓ Lambda runtime chuyển đổi sự kiện thành một đối tượng (ví dụ: kiểu dict trong Python)

```
# Ví dụ: Truy cập event object trong Lambda function
import json
```

```
def lambda_handler(event, context):
    # Access event data
    print("Event data:", event)
    print("Input arguments:", event.get('input'))
    return 'Success'
```



Đối tượng Context

- ✓ Cung cấp các phương thức và thuộc tính cung cấp thông tin về lời gọi, hàm và môi trường runtime
- ✓ Được Lambda truyền vào hàm của bạn tại thời điểm runtime
- ✓ Cung cấp thông tin như ID yêu cầu, tên hàm, giới hạn bộ nhớ, v.v.

```
# Ví dụ: Truy cập context object trong Lambda function
import json
```

```
def lambda_handler(event, context):
    # Access context information
    print("Request ID:", context.aws_request_id)
    print("Function name:", context.function_name)
    print("Memory limit (MB):", context.memory_limit_in_mb)
    return 'Success'
```



Dịch vụ kích hoạt



Lambda



Mã của bạn

Điểm đến Lambda (Lambda Destinations)

Giới thiệu

Từ tháng 11 năm 2019, bạn có thể cấu hình để gửi kết quả của hàm Lambda đến một điểm đến cụ thể.



Lưu ý: AWS khuyến nghị sử dụng Destinations thay vì DLQ (nhưng cả hai đều có thể được sử dụng cùng lúc).

⚡ Lời gọi bất đồng bộ



Thành công



Thất bại



Amazon SQS

Hàng đợi tin nhắn



Amazon SNS

Dịch vụ nhắn tin



AWS Lambda

Hàm Lambda khác



Amazon EventBridge bus

Sự kiện EventBridge

↔ Ảnh xạ nguồn sự kiện



Dành cho các lô sự kiện bị loại bỏ (discarded event batches)



Amazon SQS

Hàng đợi tin nhắn



Amazon SNS

Dịch vụ nhắn tin

Vai trò thực thi Lambda và chính sách dựa trên tài nguyên

Vai trò thực thi IAM

Vai trò thực thi Lambda (IAM Role) cấp quyền cho hàm Lambda truy cập các dịch vụ/tài nguyên AWS.

AWSLambdaBasicExecutionRole

Tải nhật ký lên CloudWatch

AWSLambdaDynamoDBExecutionRole

Đọc từ DynamoDB Streams

AWSLambdaVPCAccessExecutionRole

Triển khai hàm Lambda trong VPC


 **Thực hành tốt nhất:** Tạo một Vai trò thực thi Lambda cho mỗi hàm.

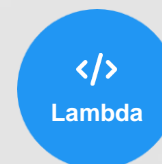
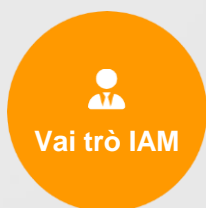
Chính sách dựa trên tài nguyên

Sử dụng chính sách dựa trên tài nguyên để cấp quyền cho các tài khoản và dịch vụ AWS khác sử dụng tài nguyên Lambda của bạn.

Cấp quyền truy cập

- ✓ Một IAM principal có thể truy cập Lambda nếu:
 - > Chính sách IAM gắn với principal cho phép (ví dụ: truy cập của người dùng)
 - > HOẶC chính sách dựa trên tài nguyên cho phép (ví dụ: truy cập của dịch vụ)

 **Ví dụ:** Khi một dịch vụ AWS như Amazon S3 gọi hàm Lambda của bạn, chính sách dựa trên tài nguyên sẽ cấp quyền truy cập cho nó.



Biến môi trường và giám sát Lambda



Biến môi trường Lambda

Biến môi trường là cặp khóa/giá trị ở dạng "String" được sử dụng để cấu hình hành vi của hàm Lambda mà không cần cập nhật mã.

KEY = VALUE

- ✓ Điều chỉnh hành vi của hàm mà không cần cập nhật mã
- ✓ Các biến môi trường có sẵn cho mã của bạn
- ✓ Hỗ trợ lưu trữ bí mật (mã hóa bằng KMS)



Mẹo: Biến môi trường rất hữu ích để lưu trữ cấu hình môi trường, chuỗi kết nối database, tên bucket S3, v.v. mà không cần phải hardcode vào mã.

Giám sát Lambda



CloudWatch Logs

- > Nhật ký thực thi AWS Lambda được lưu trữ trong AWS CloudWatch Logs
- > Đảm bảo hàm AWS Lambda của bạn có vai trò thực thi với chính sách IAM cho phép ghi vào CloudWatch Logs



CloudWatch Metrics

- > Invocations (Số lần gọi), Durations (Thời lượng), Concurrent Executions (Số lần thực thi đồng thời)
- > Error count (Số lỗi), Success Rates (Tỷ lệ thành công), Throttles (Bị điều tiết)



X-Ray Tracing

- > Kích hoạt trong cấu hình Lambda (Active Tracing)
- > Sử dụng AWS X-Ray SDK trong mã
- > Biến môi trường để giao tiếp với X-Ray: `_X_AMZN_TRACE_ID`, `AWS_XRAY_CONTEXT_MISSING`, `AWS_XRAY_DAEMON_ADDRESS`

Tùy chỉnh tại Edge với CloudFront Functions và Lambda@Edge



Nhiều ứng dụng hiện đại thực hiện một số logic tại edge để giảm thiểu độ trễ và cải thiện hiệu suất.

So sánh CloudFront Functions & Lambda@Edge

CloudFront Functions

</> JavaScript

⚡ Sub-ms startup

👤 Viewer Request/Response

Lambda@Edge

</> NodeJS/Python

⚡ 1000s req/second

👤 Origin Request/Response

Trường hợp sử dụng

CloudFront Functions

🛡️ Bảo mật trang web

🔍 SEO tối ưu

🖼️ Chuyển đổi hình ảnh

Lambda@Edge

📋 Định tuyến thông minh

👤 Xác thực người dùng

📈 Phân tích người dùng

💡 Lưu ý: CloudFront Functions chỉ thay đổi yêu cầu và phản hồi của người xem, trong khi Lambda@Edge có thể thay đổi tất cả các loại yêu cầu và phản hồi.

💡 Ví dụ: Tạo ảnh thu nhỏ của hình ảnh được tải lên S3 thông qua CloudFront Functions.

Lambda trong VPC

Cấu hình Lambda trong VPC

- ✓ Must define VPC ID, Subnets, and Security Groups
- ✓ Lambda creates an ENI (Elastic Network Interface) in your subnets
- ✓ Must attach AWSLambdaVPCAccessExecutionRole policy

Truy cập Internet cho Lambda trong VPC

- ⚠ Lambda in VPC does not have internet access by default
- ℹ Public subnet deployment does not provide internet access
- ✓ Private subnet with NAT Gateway/Instance provides internet access
- ✓ Use VPC endpoints to privately access AWS services without NAT

Lưu ý khi triển khai

- > Network configuration affects function performance
- > Security groups control access to your VPC resources
- > Consider latency when accessing VPC resources

Kiến trúc Lambda trong VPC



AWS Region

us-east-1



VPC

☒ Public Subnet

Lambda Function

ENI

? No internet access

☒ Private Subnet

Lambda Function

ENI

✓ With NAT Gateway

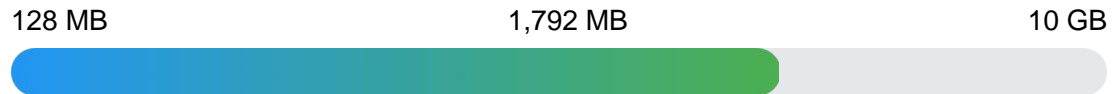


Internet

Cấu hình Lambda và ngữ cảnh thực thi

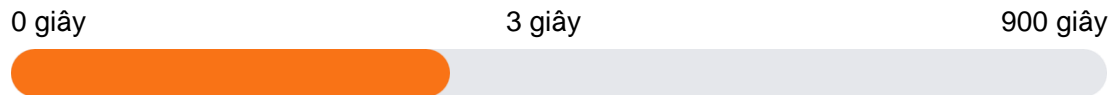
Cấu hình RAM

- ✓ Từ 128MB đến 10GB (tăng 1MB)
- ✓ RAM nhiều hơn = CPU nhiều hơn (tối đa 6 vCPU)
- ✓ 1,792 MB tương đương 1 vCPU đầy đủ




Cấu hình Timeout

- ✓ Thời gian mặc định: 3 giây
- ✓ Giá trị tối đa: 900 giây (15 phút)



Ngữ cảnh thực thi (Execution Context)

Ngữ cảnh thực thi là môi trường tạm thời được tạo ra để khởi tạo các phụ thuộc bên ngoài của mã Lambda.


 **Lưu ý:** Mã được thực thi bên ngoài handler sẽ chạy trên mỗi lần gọi (cold start).

Các thứ được lưu trong ngữ cảnh:

 Kết nối cơ sở dữ liệu

 HTTP clients

 SDK clients

 Thư mục /tmp

Thực hành tốt nhất

- ✓ Thực hiện công việc nặng bên ngoài handler
- ✓ Kết nối cơ sở dữ liệu bên ngoài handler
- ✓ Khởi tạo AWS SDK bên ngoài handler
- ✓ Sử dụng biến môi trường cho cấu hình

Concurrency, Cold Starts và Provisioned Concurrency



Concurrency và Asynchronous Invocations

- > Nếu hàm không có đủ concurrency để xử lý tất cả các sự kiện, các yêu cầu bổ sung sẽ bị chặn
- > Đối với lỗi chặn (429) và lỗi hệ thống (500-series), Lambda trả về sự kiện vào hàng đợi và attempts để chạy hàm lại trong tối đa 6 giờ
- > Khoảng thời gian retry tăng dần từ 1 giây sau lần thử đầu tiên lên đến tối đa 5 phút



Cold Starts

- > Khởi tạo instance mới => code được tải và code bên ngoài handler được chạy (init)
- > Nếu init lớn (code, dependencies, SDK...), quá trình này có thể mất một thời gian
- > Yêu cầu đầu tiên được phục vụ bởi các instance mới có độ trễ cao hơn so với các yêu cầu khác



Provisioned Concurrency

- > Concurrency được allocate trước khi hàm được invoke (in advance)
- > Nên cold start never happens và tất cả các invocation đều có độ trễ

So sánh: Normal vs Cold Start vs Provisioned Concurrency



Normal Execution

Độ trễ bình thường



Cold Start

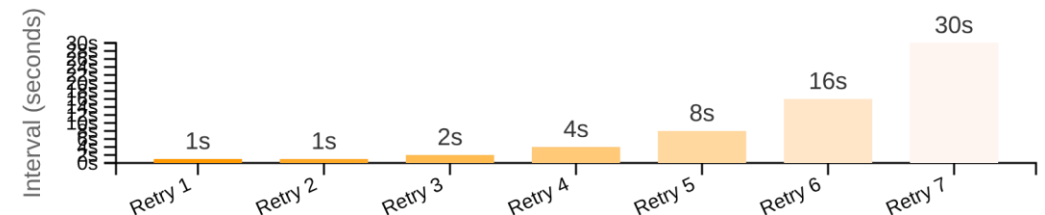
Độ trễ cao cho request đầu tiên



Provisioned

Độ trễ thấp cho tất cả request

Retry Interval Pattern for Throttled Requests



Lambda attempts to retry throttled events for up to 6 hours with increasing intervals

Phiên bản, Alias và URL cho hàm Lambda



Phiên bản (Versions)

- ✓ Phiên bản ổn định của mã hàm Lambda
- ✓ Được tạo tự động khi bạn triển khai
- ✓ Cung cấp khả năng rollback nhanh

💡 Mỗi phiên bản có một ARN ổn định và có thể được triển khai riêng biệt



Alias

- ✓ Tên gọi tham chiếu đến phiên bản Lambda
- ✓ Cho phép chuyển đổi lưu lượng
- ✓ Hỗ trợ triển khai canary và blue/green

💡 Alias giống như một bí danh cho phép bạn hướng đến nhiều phiên bản khác nhau



Lambda Function URL

- ✓ Điểm cuối HTTP(S) cho hàm Lambda
- ✓ URL duy nhất cho mỗi hàm
- ✓ Hỗ trợ CORS và chính sách dựa trên tài nguyên

 `https://[unique-id].lambda-url.[region].on.aws/`

📘 Có thể được áp dụng cho alias hoặc \$LATEST

Luồng hoạt động



Triển khai mã



Tạo phiên bản



Tạo alias



Truy cập qua URL

Giới hạn và thực hành tốt nhất với AWS Lambda

⚠ Giới hạn quan trọng của AWS Lambda

🔧 Execution

- ▶ Bộ nhớ: 128 MB – 10 GB (tăng 1 MB)
- ▶ Thời gian thực thi tối đa: 900 giây (15 phút)
- ▶ Biến môi trường: 4 KB
- ▶ Disk capacity (/tmp): 512 MB – 10 GB
- ▶ Concurrency executions: 1000 (có thể tăng)

📦 Deployment

- ▶ Kích thước nén: 50 MB
- ▶ Kích thước giải nén: 250 MB

✅ Thực hành tốt nhất



Thực hiện công việc nặng bên ngoài handler



Kết nối database bên ngoài handler



Khởi tạo AWS SDK bên ngoài handler



Nạp dependencies bên ngoài handler



Sử dụng biến môi trường cho:

- Connection strings, S3 buckets
- Mật khẩu, giá trị nhạy cảm (mã hóa KMS)



Giảm thiểu package deployment



Chia nhỏ function nếu cần



Sử dụng Layers khi cần



Tránh code đệ quy