**Serverless Product & Image Management System on AWS**

**1. Executive Summary**

Trong bối cảnh thương mại điện tử (E‑Commerce) ngày càng phát triển, nhu cầu quản lý sản phẩm linh hoạt, chi phí tối ưu và khả năng mở rộng cao trở thành yếu tố sống còn cho các cửa hàng online. Hiện nay, nhiều doanh nghiệp nhỏ và vừa (SMEs) đang gặp khó khăn khi phải duy trì hạ tầng truyền thống, tốn kém chi phí server nhàn rỗi và mất nhiều thời gian cho tác vụ bảo trì. Đồng thời, quy trình upload và xử lý hình ảnh sản phẩm thường diễn ra thủ công hoặc qua server trung gian, dẫn đến trải nghiệm người dùng không đồng bộ và dễ phát sinh lỗi.

Đề xuất giải pháp serverless trên AWS sử dụng API Gateway, AWS Lambda, Amazon DynamoDB, Amazon S3, và Amazon Cognito sẽ giúp giải quyết triệt để những vấn đề này. Hệ thống cho phép:

* Tạo, cập nhật, xóa và truy vấn danh mục sản phẩm thông qua API REST, không cần quản lý máy chủ.
* Cấp URL upload ảnh sản phẩm an toàn (Presigned URL) lên S3 bucket originals.
* Kích hoạt Lambda resize ảnh tự động khi có file mới, lưu kết quả vào S3 bucket resized.
* Lưu trữ metadata và liên kết ảnh trong DynamoDB để truy xuất nhanh.
* Bảo mật API qua Amazon Cognito, phân quyền linh hoạt.
* Host giao diện front‑end tĩnh trên S3 + CloudFront, tối ưu hiệu năng và chi phí.

Giải pháp serverless của chúng tôi mang lại lợi ích:

* **Tiết kiệm chi phí**: trả phí theo actual usage, không có chi phí server nhàn.
* **Khả năng mở rộng tự động**: đáp ứng bất kỳ biến động traffic nào.
* **Tăng tốc độ triển khai**: focus vào nghiệp vụ, không phải vận hành hạ tầng.
* **Trải nghiệm người dùng tốt hơn**: hình ảnh sản phẩm chuẩn hóa, tải nhanh.

Dự án dự kiến hoàn thành trong 10 tuần với 5 giai đoạn triển khai, tổng đầu tư AWS ước tính khoảng 2.500 USD/năm. ROI dự kiến đạt điểm hòa vốn sau 6 tháng vận hành khi hệ thống tiết kiệm tối thiểu 40% chi phí hạ tầng so với mô hình EC2 truyền thống.

**Ví dụ tính toán Return on Investment - ROI:**

*Chi phí mô hình EC2 truyền thống:*

* EC2 instance (t3.medium): 2 instance × 0.0416 USD/hour × 24 hours × 30 days = 59.9 USD/tháng
* EBS volume 100GB: 0.10 USD/GB × 100GB = 10 USD/tháng
* Chi phí nhân công vận hành: ~500 USD/tháng
* Tổng cộng EC2 truyền thống: ~570 USD/tháng (~6,840 USD/năm)

*Chi phí Serverless AWS:*

* AWS Lambda: ~100 USD/tháng
* Amazon DynamoDB (On-demand): ~50 USD/tháng
* Amazon S3: ~20 USD/tháng
* Amazon CloudFront: ~20 USD/tháng
* Amazon Cognito: ~10 USD/tháng
* Tổng cộng Serverless AWS: ~200 USD/tháng (~2,400 USD/năm)

*Tiết kiệm chi phí:*

* Tiết kiệm hàng tháng: 570 USD - 200 USD = 370 USD/tháng
* Tiết kiệm hàng năm: 6,840 USD - 2,400 USD = 4,440 USD/năm (~65% tiết kiệm)

Như vậy, giải pháp serverless không chỉ tiết kiệm chi phí đáng kể mà còn nâng cao hiệu suất hoạt động và cải thiện trải nghiệm khách hàng.

**2. Problem Statement**

***2.1. Current Situation***

Nhiều SME sử dụng máy chủ EC2 hoặc VPS để chạy CMS và lưu trữ hình ảnh, tốn chi phí cố định và công sức quản trị.

Quá trình upload, resize ảnh thường thực hiện thủ công hoặc qua script trên server, dễ gây nghẽn và lỗi khi số lượng ảnh lớn.

Hạ tầng không tự động scale, khi peak traffic (khuyến mãi, flash sale) dễ gặp downtime.

***2.2. Key Challenges***

Chi phí vận hành cao: trả phí server idle liên tục.

Quy trình xử lý ảnh không đồng bộ: mất thời gian resize, chuyển định dạng.

Khó mở rộng: phải cấu hình manual auto‑scaling hoặc thêm máy chủ.

Bảo mật và phân quyền: thiếu giải pháp IAM/Cognito chuyên biệt.

***2.3. Stakeholder Impact***

Chủ shop: không có công cụ quản lý tập trung, dễ nhầm lẫn thông tin.

Khách hàng: ảnh sản phẩm load chậm, không đúng kích thước, ảnh hưởng trải nghiệm mua sắm.

Dev/Ops: tốn nhiều thời gian bảo trì server, cập nhật bản vá.

Bảo mật: rủi ro từ server nhàn và cấu hình lỏng lẻo.

***2.4. Business Consequences***

Mất doanh thu do trải nghiệm kém và downtime.

Chi phí cố định cao, không linh hoạt theo mùa vụ.

Khó cạnh tranh với các nền tảng nhanh gọn, chuyên nghiệp.

**3. Solution Architecture**

***3.1. Architecture Overview***

Giải pháp được xây dựng dựa trên mô hình kiến trúc serverless và event-driven, gồm các thành phần chính:

* **Amazon API Gateway**: Cung cấp REST API để frontend hoặc bên thứ ba gọi đến.
* **AWS Lambda**: Chịu trách nhiệm xử lý toàn bộ logic ứng dụng như CRUD sản phẩm/danh mục, cấp Presigned URL, resize ảnh.
* **Amazon S3**: Chứa ảnh gốc (bucket upload-originals) và ảnh đã resize (bucket upload-resized). Đồng thời dùng làm nơi host giao diện frontend tĩnh.
* **Amazon DynamoDB**: Cơ sở dữ liệu NoSQL để lưu trữ metadata sản phẩm, danh mục và liên kết hình ảnh.
* **Amazon Cognito**: Xác thực và phân quyền người dùng (User Pool và Identity Pool).
* **Amazon CloudFront**: Dịch vụ CDN giúp cache nội dung frontend và giảm độ trễ truy cập toàn cầu.

Kiến trúc này không yêu cầu quản lý máy chủ, tự động scale và có độ sẵn sàng cao, rất phù hợp cho SMEs với chi phí tối ưu.

***3.2. AWS Services Used***

Dưới đây là các dịch vụ AWS chính được sử dụng trong hệ thống:

* **Amazon API Gateway** – Giao tiếp giữa client và backend serverless
* **AWS Lambda** – Xử lý nghiệp vụ chính
* **Amazon S3** – Lưu trữ ảnh và host website tĩnh
* **Amazon DynamoDB** – Lưu metadata sản phẩm, danh mục
* **Amazon Cognito** – Quản lý người dùng và phân quyền truy cập API
* **Amazon CloudFront** – Caching và phân phối frontend nhanh chóng

***3.3. Component Design***

Luồng xử lý chính của hệ thống bao gồm:

1. Người dùng frontend gọi đến API Gateway, request được xác thực thông qua Cognito Authorizer.

2. API Gateway forward request sang Lambda tương ứng:

* + Lambda CRUD thao tác trên bảng DynamoDB.
  + Lambda GetPresignedUrl tạo URL tạm để upload ảnh lên bucket upload-originals.

3. Khi ảnh được upload thành công lên S3 (bucket gốc), S3 Event Trigger sẽ gọi Lambda Resize:

* + Lambda Resize xử lý ảnh và lưu lại ảnh resize vào bucket upload-resized.

4. Metadata ảnh resize (tên file, URL, thời gian tạo, v.v.) được cập nhật vào DynamoDB.

***3.4. Security Architecture***

Giải pháp tuân thủ nguyên tắc bảo mật **least privilege**:

* **IAM Role riêng biệt** cho từng Lambda function, chỉ cấp quyền tối thiểu cần thiết để truy cập DynamoDB hoặc S3.
* **Cognito User Pool** dùng để đăng ký/đăng nhập người dùng, **Identity Pool** cấp quyền tạm thời nếu frontend cần gọi trực tiếp vào S3.
* **S3 Bucket Policy** giới hạn PUT object chỉ cho phép thông qua Presigned URL và deny mọi truy cập công khai khác.
* **API Gateway Authorizer** bảo vệ toàn bộ API backend bằng token của Cognito.

***3.5. Scalability Design***

Hệ thống có khả năng tự mở rộng theo nhu cầu sử dụng mà không cần can thiệp thủ công:

* **Lambda Functions**: Tự động scale theo concurrency thực tế.
* **DynamoDB**: Sử dụng chế độ On-Demand giúp scale linh hoạt theo số lượng truy vấn.
* **S3**: Scale gần như vô hạn, có khả năng phục vụ hàng nghìn request đồng thời.
* **CloudFront**: Giảm tải truy cập API/S3 bằng cơ chế cache toàn cầu, tối ưu tốc độ tải frontend và ảnh resize.
* **Presigned URL**: Tránh bottleneck khi upload ảnh do client upload trực tiếp lên S3 thay vì qua backend.

**4. Technical Implementation**

***4.1. Implementation Phases***

Hệ thống sẽ được triển khai theo từng giai đoạn cụ thể nhằm đảm bảo tính toàn vẹn, kiểm thử hiệu quả và khả năng mở rộng linh hoạt:

*1. Triển khai Lambda CRUD kết nối DynamoDB*:

* + Tạo các hàm Lambda thực hiện Create, Read, Update, Delete trên bảng Products và Categories trong DynamoDB với chế độ On-Demand.
  + Tích hợp các hàm Lambda này với Amazon API Gateway (REST API Proxy Integration).

*2. Tạo Pre-signed URL để upload ảnh lên S3*:

* + Viết Lambda function cấp URL tạm thời (Presigned URL) cho phép người dùng upload ảnh trực tiếp từ frontend lên bucket upload-originals.
  + Tối ưu cấu hình CORS cho S3 bucket và API Gateway để đảm bảo upload ảnh từ frontend hoạt động mượt mà.

*3. Triển khai Lambda Resize ảnh và cấu hình trigger từ S3:*

* + Khi ảnh gốc được upload vào bucket, S3 sẽ kích hoạt Lambda function để tự động resize ảnh và lưu vào bucket upload-resized.
  + Sử dụng thư viện Sharp trong Lambda (build trong Docker image) để đảm bảo khả năng resize nhanh và chuẩn hóa kích thước ảnh.

*4. Thiết lập Amazon Cognito cho xác thực người dùng:*

* + Tạo User Pool để đăng ký/đăng nhập người dùng.
  + Cấu hình Identity Pool nếu cần cấp quyền truy cập AWS service trực tiếp từ frontend.
  + Tích hợp Cognito User Pool với API Gateway Authorizer để bảo vệ các endpoint quan trọng.

*5. Triển khai giao diện người dùng:*

* + Build frontend bằng React/Next.js (hoặc HTML đơn giản), sau đó deploy lên S3 bucket dùng cho static website hosting.
  + Cấu hình CloudFront để phân phối nội dung frontend, đảm bảo tốc độ truy cập nhanh và ổn định trên toàn cầu.

***4.2. Technical Requirements***

* **Ngôn ngữ & Runtime**: Node.js 22.x, sử dụng cú pháp ES Module.
* **Thư viện chính**: AWS SDK v3 (Modular SDK).
* **Dịch vụ AWS sử dụng**:
  + Amazon DynamoDB (On-Demand mode)
  + AWS Lambda
  + Amazon API Gateway (REST API Proxy Integration)
  + Amazon S3
  + Amazon Cognito (User Pool & Identity Pool)
  + Amazon CloudFront
* **Kiến trúc**: Event-driven, không dùng máy chủ, scale tự động, sử dụng S3 Events và API Gateway làm điểm kích hoạt.

***4.3. Development Approach (Định hướng triển khai – chưa thực hiện hoàn chỉnh)***

Lên kế hoạch thiết lập CI/CD pipeline với GitHub Actions để tự động hoá build, test, deploy các hàm Lambda và tài nguyên liên quan.

***4.4. Testing Strategy (Định hướng – chưa thực hiện đầy đủ)***

Định hướng triển khai theo mô hình môi trường: Dev → Test → Prod.

Sử dụng AWS CodePipeline để thiết lập rollback tự động khi deployment thất bại (chưa triển khai ở giai đoạn hiện tại).

**5. Timeline & Milestones**

***5.1. Project Timeline***

|  |  |  |
| --- | --- | --- |
| **Giai đoạn** | **Tuần** | **Công việc chính** |
| Phase 1 | 1–2 | Thiết lập Cognito, API Gateway, Lambda CRUD, DynamoDB |
| Phase 2 | 3–4 | Presigned URL & S3, Resize Lambda & Event Trigger |
| Phase 3 | 5–6 | Front-end & CloudFront, Kiểm thử & Triển khai |

***5.2.* Key Milestones**

**M1 – API CRUD hoạt động:** Thiết kế và triển khai đầy đủ các API (Create, Read, Update,Delete)cho sản phẩm, danh mục,... Tích hợp với DynamoDB hoặc Aurora Serverless tùy kiến trúc API Gateway + Lambda hoạt động ổn định

**M2 – Upload ảnh & Resize thành công:** Cho phép người dùng upload ảnh sản phẩm Resize ảnh tự động thông qua AWS Lambda (hoặc AWS Step Functions). Ảnh được lưu trữ an toàn trên S3 với các phiên bản ảnh phù hợp

**M3 – Front-end hiển thị live qua CloudFront**: Giao diện người dùng được triển khai bằng S3 Static Website Hosting. Phân phối nội dung thông qua AWS CloudFront để tăng tốc độ truy cập Website hoạt động live, phản hồi nhanh với domain tạm hoặc tùy chỉnh

***5.3. Dependencies***

Tạo sẵn AWS account & IAM roles

***5.4. Resource Allocation***

Toàn bộ dự án được triển khai bởi một cá nhân

**6. Budget Estimation**

***6.1. Infrastructure Costs (Chi phí hạ tầng hàng tháng)***

|  |  |
| --- | --- |
| **Dịch vụ** | **Chi phí ước tính** |
| AWS Lambda | 100 USD/tháng |
| DynamoDB (On-Demand) | 50 USD/tháng (tuỳ mức sử dụng) |
| Amazon S3 (storage + I/O) | 20 USD/tháng |
| Amazon CloudFront | 20 USD/tháng |
| Amazon Cognito | 10 USD/tháng |

* **Tổng chi phí hạ tầng hàng tháng: ~200 USD**
* **Tổng chi phí hạ tầng hàng năm: ~2,400 USD**

***6.2 Development Costs (Chi phí phát triển)***

Tự triển khai toàn bộ dự án → 0 USD (không tính vào ngân sách vì tự thực hiện)

***6.3 Operational Costs (Chi phí vận hành & hỗ trợ)***

**Support & Monitoring:** ~500 USD/tháng

***6.4 ROI Analysis (Phân tích lợi tức đầu tư)***

* Tổng chi phí AWS trung bình: 200 USD/tháng ≈ 2,400 USD/năm
* So với kiến trúc EC2/VPS truyền thống, tiết kiệm khoảng 65% chi phí, tương đương ~4,440 USD/năm
* Dự kiến hoàn vốn sau 6 tháng vận hành nhờ tối ưu chi phí và hiệu quả từ mô hình serverless

**7. Risk Assessment**

***7.1. Risk Matrix***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rủi ro** | **Xác suất** | **Ảnh hưởng** | **Mức độ rủi ro** | **Ưu tiên** |
| Cold start Lambda cao | Medium | High | Medium (M) | 1 |
| Quá tải DynamoDB | Medium | High | Medium (M) | 2 |
| Lỗi Permission IAM | Low | Medium | Low (L) | 3 |
| S3 downtime (ngừng dịch vụ) | Low | High | Low (L) | 4 |

***7.2 Mitigation Strategies***

Enable provisioned concurrency cho Lambda nhạy độ trễ.

Sử dụng IAM least privilege, review policy.

Thiết kế retry/backoff, DLQ cho S3 Event.

DynamoDB on‑demand mode, CloudWatch alarms.

***7.3. Contingency Plans***

Rollback Canary deployment.

Chuyển sang batch processing nếu DynamoDB over-provision.

Thay thế S3 bằng EFS (nếu cần độ bền cao).

**8. Expected Outcomes**

***8.1. Success Metrics***

Thời gian phản hồi API <100ms

Tỷ lệ thành công resize ảnh 100%

Availability hệ thống ≥99,9%

***8.2. Business Benefits***

Tiết kiệm chi phí hạ tầng 60%

Rút ngắn thời gian triển khai feature mới 50%

Technical Improvements

Loại bỏ server quản lý, hạ tầng serverless linh hoạt

Quy trình xử lý ảnh tự động, chuẩn hóa

***8.3. Long-term Value***

Dễ mở rộng vào peak sale, Black Friday

Nền tảng sẵn sàng tích hợp ML (recommendation)

**9. Appendices**

***9.1 Technical Specifications***

* **Ngôn ngữ & Runtime**: Node.js 22.x (ES Modules)
* **Thư viện chính**: AWS SDK v3 (Modular architecture)
* **Lambda Function Configuration**:
  + Memory: 512MB
  + Timeout: 30 giây
  + Logging: AWS CloudWatch Logs, AWS X-Ray (bật theo dõi tracing nâng cao)
* **S3 Configuration**:
  + 2 buckets: upload-originals, upload-resized
  + Cấu hình CORS rõ ràng, chính sách public access block toàn diện
* **API Gateway:**
  + REST API Proxy Integration
  + Cognito User Pool Authorizer tích hợp để bảo vệ endpoint
* **Cơ chế Resize ảnh:**
  + Thư viện Sharp tích hợp qua Docker image Lambda layer
  + Resize kích thước chuẩn về chiều dài 600px (hoặc tuỳ chỉnh qua ENV)