AurumGuard Inventory Tracking

AurumGuard is a serverless inventory tracking solution designed for both businesses and individuals who need to manage and protect high-value jewelry assets. It focuses on reducing losses, improving visibility, and making it easier to prove ownership when working with insurers or auditors.

The system is built entirely on Amazon Web Services (AWS) using a lightweight, scalable architecture: a static web dashboard hosted on Amazon S3, a REST API hosted on Amazon API Gateway, Lambda functions written in Python, and a DynamoDB table that stores jewelry inventory data. All of this was deployed and tested from AWS CloudShell using the AWS Command Line Interface (CLI).

From the user’s perspective, AurumGuard provides a clean jewelry dashboard that shows items such as rings, earrings, and sapphire necklaces, along with their category, metal, stone, size, cost, retail price, and whether they are unique pieces.

How We Built It (Architecture)

1. Frontend (Presentation Layer):
• A modern, single-page dashboard built as a static HTML file.
• Hosted on an Amazon S3 bucket configured for static website hosting.
• Styled with custom CSS for a luxury, dark-themed jewelry look.
• Uses JavaScript to call a REST API and dynamically render product cards.

2. API Layer:
• Amazon API Gateway exposes a REST API endpoint (for example, /items).
• The API forwards requests to an AWS Lambda function.
• The API is regional, secured by IAM permissions, and returns JSON data that the frontend can easily consume with fetch().

3. Compute Layer (AWS Lambda):
• Lambda is implemented in Python.
• It reads and writes inventory records in Amazon DynamoDB.
• It formats the results as JSON for the frontend, including fields such as name, category, metal, stone, size, costPrice, retailPrice, isUnique, and imageUrl.
• Lambda is fully serverless: no servers to patch, and it scales automatically with traffic.

Data Layer and AWS Services

4. Data Layer (Amazon DynamoDB):
• DynamoDB is used as the primary data store for jewelry items.
• Each record represents a single piece of jewelry, identified by an itemId.
• The schema supports multi-tenant scenarios using a tenantId field.
• DynamoDB’s managed scalability and durability make it ideal for inventory workloads.

5. Static Assets and Images (Amazon S3 and External URLs):
• The HTML dashboard is served from an S3 website endpoint.
• Jewelry images are referenced by URL, stored in S3 or loaded from reputable image CDNs.
• This keeps the architecture simple while still allowing rich, high-quality product photos.

6. Tooling and Security:
• AWS CloudShell and the AWS CLI were used for deployment, configuration, and testing.
• AWS Identity and Access Management (IAM) controls which services can call each other (for example, API Gateway invoking Lambda, and Lambda reading from DynamoDB).
• Logging and monitoring can be added via Amazon CloudWatch to trace requests end-to-end.

Benefits for Companies and Jewelry Owners

Business Benefits:
• Loss Reduction: By tracking every high-value item with cost and retail price, companies can quickly identify missing or unaccounted-for pieces.
• Real-Time Visibility: Inventory data is available through the API and dashboard at any time, from anywhere.
• Operational Efficiency: Serverless architecture reduces the need for infrastructure management, allowing teams to focus on business processes instead of servers.
• Auditability: With consistent records, it is easier to create audit trails and demonstrate control over high-value stock.

Benefits for Individuals with Expensive Jewelry:
• Proof of Ownership: Each item can be registered with details such as metal, stone, size, and photos. This acts as strong documentation in case of loss or theft.
• Easier Insurance Claims: When an insured piece goes missing or is damaged, the owner can provide clear, structured evidence of the item’s existence and value, simplifying discussions with insurance companies.
• Centralized Record: Instead of scattered receipts and photos, owners have one centralized record of their premium pieces.
• Privacy and Security: Because the system is built on AWS, access can be controlled using IAM and application logic, limiting who can see what data.

Why AWS and Serverless Technology

AWS is a strong fit for this kind of solution because it provides secure, pay-as-you-go, fully managed services that reduce operational overhead. Serverless building blocks like API Gateway, Lambda, and DynamoDB scale automatically with usage, so the same design can support a single user, a small jewelry shop, or a large enterprise retailer.

Key advantages of using AWS serverless services for AurumGuard include:
- Scalability: The system can handle more reads and writes without redesigning the infrastructure.
- Cost Efficiency: You pay primarily for actual usage rather than idle capacity.
- Security: IAM, encryption options, and managed services reduce the risk of misconfiguration and data exposure.
- Agility: It is easy to add new fields, endpoints, or analytics features without provisioning new servers.

Overall, AurumGuard demonstrates how a focused, serverless AWS architecture can protect high-value jewelry assets, support better insurance and audit processes, and deliver a modern, visually appealing dashboard for both companies and individual owners.