INDEPENDENT HANDS-ON PROJECT

Reliable Serverless Event-Driven Platform

Durable event buffering, retries, dead-letter handling and bounded concurrency

Portfolio track	Independent Hands-on Engineering
Evidence level	Terraform fmt, init and validate passed
Author	Saurabh Dindokar

Evidence boundary: This is a public-safe personal implementation lab. It is not client production work, and no live deployment is claimed.

Overview

An event-driven Terraform design where S3 object-created events pass through EventBridge to an encrypted SQS queue, Lambda processes bounded batches, failed messages reach a DLQ and CloudWatch alarms notify through encrypted SNS.

Engineering Problem

Direct event-to-function integration can hide retry pressure and make failure recovery harder. The platform introduces a durable queue, a dead-letter path, bounded concurrency and operational alarms so failed work remains visible and recoverable.

Architecture

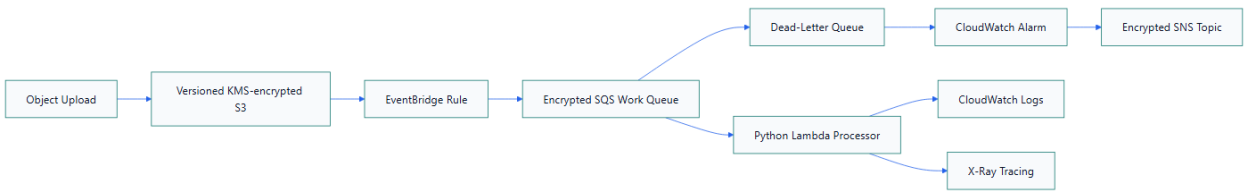

Figure 1. Reliable Serverless Event-Driven Platform architecture

What I Implemented

- Enabled S3 public-access blocking, versioning, KMS encryption and bounded object lifecycle.
- Routed S3 events through EventBridge into SQS using a source-restricted queue policy.
- Configured queue visibility timeout above Lambda timeout and a DLQ after three receives.
- Limited Lambda concurrency, enabled X-Ray and used partial-batch response support.
- Created a DLQ-visible-message alarm with optional encrypted SNS email notification.

Important Technical Decisions

- SQS decouples event arrival rate from processing capacity.
- Reserved concurrency protects downstream systems.
- The DLQ retains failed messages for investigation rather than silently discarding them.
- The handler logs metadata only; production business logic must be idempotent.

Security Controls

- S3 public access is blocked and objects use a customer-managed KMS key.
- Queue policies restrict EventBridge sends to the expected rule.
- The Lambda role contains only queue and log permissions required by the sample.

Reliability and Operations

- Queue buffering absorbs bursts and supports retry.
- Partial-batch failure reporting prevents successful records from being retried unnecessarily.
- DLQ alarms make exhausted retries operationally visible.

Cost and Cleanup Guardrails

- S3 lifecycle expires lab objects and old versions.
- Reserved concurrency bounds Lambda scaling.
- CloudWatch retention and optional notification subscription are explicit.

Validation Evidence

The following local checks passed on 2026-08-13:

```
terraform fmt -check -recursive
terraform init -backend=false
terraform validate
```

Scope boundary: Format, initialization and validation passed on 2026-08-13. No plan, event injection, retry exhaustion, DLQ alarm or AWS deployment is claimed. The sample handler is not a business processor.

Delivery and Verification Runbook

- 1. Upload object
- 2. Match EventBridge rule
- 3. Buffer event in SQS
- 4. Invoke Lambda in bounded batches
- 5. Retry failures
- 6. Move exhausted messages to DLQ
- 7. Alarm and investigate before redrive

Technology Stack

Terraform | Amazon S3 | Amazon EventBridge | Amazon SQS | AWS Lambda | Amazon SNS | CloudWatch | AWS KMS | Python

Key Learnings

- Infrastructure evidence must distinguish code validation, plan review and live deployment.
- Security, reliability, cost and cleanup decisions should be documented before apply.
- A useful platform lab includes verification and rollback thinking, not only resource declarations.