Technical Design Document: Event-Driven S3 Backup & Validation System

Version: 1.0
Status: Final
Author: Cloud Engineering Team
Last Updated: 2025-11-12

1. Introduction & Scope

1.1. Overview

This document outlines the technical architecture and design of a serverless, event-driven system for backing up, replicating, and validating objects in Amazon S3. The system is designed for high availability, resilience, security, and cost-efficiency, managed entirely through Infrastructure as Code (IaC) and deployed via a secure CI/CD pipeline.

1.2. Problem Statement

Standard S3 Cross-Region Replication (CRR) provides disaster recovery but lacks a built-in, programmatic mechanism for verifying the integrity of replicated objects. Additionally, manual or scheduled (cron-based) backup validation scripts are brittle, do not scale, and lack robust error handling. This system aims to solve these problems by providing an automated, near-real-time validation workflow that is both reliable and observable.

1.3. Requirements

- **R1 (Data Redundancy):** All objects uploaded to primary S3 buckets must be automatically replicated to a secondary AWS region.
- **R2 (Data Integrity):** The bit-for-bit integrity of every replicated object must be programmatically verified.
- **R3 (Categorization):** The system must support distinct storage locations for different data categories (e.g., documents, media, databases).
- **R4 (Automation):** The entire validation process must be triggered automatically upon object creation without manual intervention.

- **R5 (Resilience):** The system must be fault-tolerant, capable of handling transient errors (like replication lag) and isolating persistent "poison pill" messages.
- **R6 (Security):** All data must be encrypted at rest. All services must operate under the Principle of Least Privilege. Public access must be disabled.
- **R7 (Observability):** The system must log successful validations and provide immediate, actionable alerts for unrecoverable failures.
- **R8 (IaC & GitOps):** All infrastructure must be defined as code, version-controlled, and deployed via an automated, secure pipeline.

2. System Architecture

The architecture is designed around loosely coupled, serverless components, forming an asynchronous, event-driven workflow.

2.1. High-Level Diagram

https://github.com/yisakm9/Project-13-Automatic-Backup-System/blob/main/Automatic%20Backup%20System.drawio.png

2.2. Core Components & Data Flow

1. **Storage Layer (Amazon S3):**
    - Consists of multiple pairs of S3 buckets (Primary and Replica). Each pair is dedicated to a specific data category.
    - **Primary Buckets:** The entry point for all uploads. Configured with versioning, server-side encryption (SSE-S3), blocked public access, and event notifications enabled for EventBridge.
    - **Replica Buckets:** Reside in a secondary AWS region. Receive objects via Cross-Region Replication (CRR) and have identical security configurations.
    - **Lifecycle Policies:** Both bucket types have policies to transition non-current versions to S3 Glacier Instant Retrieval and eventually expire them, optimizing for cost. An additional policy aborts incomplete multipart uploads after 7 days.
2. **Event Bus (Amazon EventBridge):**
    - The default event bus is used to receive s3:ObjectCreated:* events from all primary S3 buckets.

- A single rule (autobackup-s3-upload-trigger-dev) is configured with an event pattern that matches these S3 events.
- A resource-based policy on the event bus explicitly grants the S3 service principal (s3.amazonaws.com) permission to publish events.

3. **Messaging Layer (Amazon SQS & KMS):**
    - **Customer-Managed Key (CMK):** A dedicated CMK (autobackup-sqs-key-dev) is used to encrypt all queues, ensuring data is protected at rest. The key policy is critical, granting usage permissions to:
        - events.amazonaws.com (to send messages).
        - lambda.amazonaws.com (for the DLQ integration).
        - The checksum-validator IAM role (to decrypt messages).
    - **Validation Queue (autobackup-validation-queue-dev):** The target of the EventBridge rule. It acts as a durable buffer for incoming validation tasks.
    - **Validation DLQ (autobackup-validation-dlq-dev):** Receives messages from the validation queue after 3 failed processing attempts. This isolates "poison pill" messages.
    - **Failure Queue (autobackup-failure-queue-dev):** A separate queue that serves as the trigger for the notification workflow. It is intended to receive messages manually redriven from the validation DLQ.
    - **Failure DLQ (autobackup-failure-dlq-dev):** The DLQ for the failure queue itself, providing an extra layer of resilience.
4. **Compute Layer (AWS Lambda):**
    - **checksum-validator Function:**
        - **Trigger:** The SQS validation-queue.
        - **Execution Role:** iam_checksum_validator-role grants permissions to read from S3 buckets, consume from the SQS queue, and use the SQS KMS key.
        - **Logic:**
            1. Receives and decrypts a batch of SQS messages.
            2. For each message, parses the S3 object details.
            3. Fetches the ETag of the primary and replica objects.
            4. **Handles Multipart Uploads:** Normalizes ETags by stripping the -N suffix before comparison.
            5. **Handles Replication Lag:** Catches 404 Not Found errors from the replica, logs a warning, and re-raises an exception to trigger an SQS retry.

6. Logs success to CloudWatch and returns successfully, allowing the message to be deleted from the queue.

- **failure-notifier Function:**
  - **Trigger:** The SQS failure-queue.
  - **Execution Role:** iam_failure_notifier-role grants permissions to consume from the failure queue, publish to the SNS topic, and use both the SQS and SNS KMS keys.
  - **Logic:** Parses the failed message, formats a human-readable alert, and publishes it to the SNS topic.

5. **Notification Layer (Amazon SNS & KMS):**
   - **Customer-Managed Key (CMK):** A dedicated CMK (autobackup-sns-key-dev) encrypts the topic. Its policy grants usage permission to the failure-notifier IAM role.
   - **Failure Topic (autobackup-failure-topic-dev):** Receives alerts from the failure-notifier Lambda.
   - **Subscription:** An email endpoint is subscribed to the topic to deliver alerts to operators.

3. Security Design

- **Authentication:** The CI/CD pipeline uses OIDC for secure, short-lived credentials. All service-to-service communication relies on IAM roles and resource-based policies.
- **Authorization (IAM):** The Principle of Least Privilege is enforced. For example, the checksum-validator role has read-only access to S3 and cannot publish to SNS.
- **Encryption at Rest:**
  - S3 buckets use SSE-S3.
  - SQS queues and SNS topics are encrypted with dedicated, customer-managed KMS keys, giving us full control over access policies.
  - Lambda environment variables are encrypted with the default AWS-managed Lambda KMS key.
- **Network Security:** All Lambda functions operate on the AWS public network and do not require VPC placement, reducing complexity and cost. Communication is secured via AWS PrivateLink for service endpoints where applicable and TLS for public endpoints.

4. CI/CD Pipeline (GitHub Actions)

The project utilizes two distinct workflows for a professional GitOps approach.

- **terraform-deploy.yml:**
  - **Triggers:** On push to main or pull request to main.
  - **Jobs:**
    1. tflint: Lints the code for best practices.
    2. checkov: Scans the code for security misconfigurations.
    3. terraform: Depends on the success of the scan jobs.
       - On pull_request, runs terraform plan and posts the output as a PR comment for review.
       - On push to main, runs terraform apply -auto-approve to deploy the changes.
- **terraform-destroy.yml:**
  - **Trigger:** Manual workflow_dispatch only.
  - **Safety:** Requires the user to type a specific confirmation phrase (destroy all resources) to proceed.
  - **Action:** Runs terraform destroy -auto-approve to tear down all managed resources.

5. Operational Procedures

- **Monitoring:** The primary source of monitoring is **CloudWatch Logs**. The log group for /aws/lambda/autobackup-checksum-validator-dev should be monitored for SUCCESS messages and WARNING messages related to replication lag.
- **Alerting:** Alerts are generated only for un-processable messages. An email alert signifies that a message is in the autobackup-validation-dlq-dev and requires manual intervention.
- **Incident Response:**
  1. Receive an email alert from SNS.
  2. Navigate to the autobackup-validation-dlq-dev SQS queue.
  3. Inspect the message body to understand the cause of failure (e.g., malformed event, bug in the Lambda code).

4. After resolving the root cause, the message can be redriven to the autobackup-validation-queue-dev for reprocessing or discarded if it is irrelevant.