National College of Ireland

Unified Multi-Feed Anomaly Detection for AWS Lambda

MSc Research Project
Cloud Computing

Devshree Ethape
Student ID: 23417196

School of Computing
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Devshree Ethape
Student ID:	23417196
Programme:	Cloud Computing
Year:	2025-2026
Module:	MSc Research Project
Supervisor:	Rashid Mijumbi
Submission Due Date:	28-01-2026
Project Title:	Unified Multi-Feed Anomaly Detection for AWS Lambda
Word Count:	9302
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Devshree Ethape
Date:	28th January, 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Unified Multi-Feed Anomaly Detection for AWS Lambda

Devshree Ethape
23417196

Abstract

Serverless computing has been increasing at a rapid pace, but AWS Lambda does not offer coherent telemetry, host-levels, and powerful real-time anomaly detection. Such restrictions do not allow for promptly detecting performance degradation, cost-based attacks, and cold-start-related disturbances and render serverless observability an urgent research issue. To resolve this issue, the paper formulates and tests a completely AWS-native, real-time anomaly detector pipeline, which combines multi-source telemetry gatherings on CloudWatch, EventBridge and Lambda into a solitary Kinesis-Flink analytics bear.

The work is constructed in Design Science Research Methodology by creating an operational artefact which consists of ingestion through Kinesis Data Streams, in-stream analysis through Apache Flink SQL and automated anomaly alerts through Amazon SNS. Implementation of a threshold-based detection logic was conducted to detect an anomaly in terms of execution-time, and controlled experiments with synthetic attack patterns tested how well the model can detect an abnormal behaviour with accuracy and virtually in-time responsiveness.

The results present a theoretical argument that serverless observability is possible without host-level indicators based on integrated streams of telemetry hosted at the cloud, which correlates with the current body of work on Lambda-based security architecture. Practically the pipeline offers a very scalable and non-overhead mechanism of anomaly detection that can function in conjunction with current cloud operations. Nevertheless, there are still unresolved issues, such as the use of single-metric thresholds, the lack of machine-learning intelligence, and restricted multi-signal correlation, which bring opportunities to practice Isolation Forest or auto encoders-based detector models in the future.

1 Introduction

1.1 Background

With the pace of cloud implementation in the world steadily growing, serverless computing has become one of the architectural patterns of choice, due to its scalability, affordability, and ease of implementation. In the serverless ecosystem, AWS Lambda has emerged as a leading option among companies that require the elasticity of computing power with no load of infrastructure management (Syed and Ali, 2024). However, the growing use of serverless architecture has heightened interest with respect to security, observability, and cost uncertainties. Conventional security surveillance systems, which are mainly used with long-running servers, have a hard time keeping up with the extremely dynamic nature of serverless workloads.

AWS uses the micro-based virtualization system called Firecracker that gives serverless functions delivered by lightweight and isolated microVMs (Veltman et al., 2023). Firecrackers provide high isolation but provides another challenge to researchers: it obscures host-level measurements and, therefore, does not allow researchers to observe runtime behavior. As a result, application-level, platform-level, and telemetry signals have to play a significant role in anomaly detection in Lambda environments. To handle the modern patterns of attacks, e.g., cold-start denial-of-service attacks, cost-

amplification attacks, and anomalous invocation bursts, there is a requirement for advanced and real-time anomaly detection models that are tightly coupled with cloud telemetry streams.

1.2 Problem Domain

In spite of AWS providing logs and metrics of various telemetry providers, organisations still face critical security issues, including Poor visibility of host metrics. Firecracker host metrics are not even visible that prevents the detection of low-level anomalies. Security systems should thus rely on Lambda-level metrics, like CloudWatch metrics, X-Ray traces, function logs and budgetary data (Panda, 2025).

In addition to it, the Lack of real-time analytics is also an issue. Eventbridge, CloudWatch, Lambda logs and cost-monitoring APIs are used to make serverless telemetry very distributed (Herrera et al., 2025). In the absence of an integrated pipeline, it is difficult to identify the deviations of behavior in real time. Emerging attack vectors. Operating loads are also being attacked using new patterns of exploits, such as cold-start high-frequency attacks and denial-of-wallet attacks that traditional rule-based systems cannot detect.
No clear definition of anomalies and criteria of evaluation: The academic literature does not have a clear definition of a schema of serverless telemetry, anomaly boundaries, or metrics of evaluation specific to serverless environments (Mampage et al., 2022). With these gaps, there is a need to have a systematic method that consumes various sources of telemetry, processes them in real time, and finds abnormalities using both rule-based and machine-learning-based methods.

1.3 Aim

The project is aimed at creating and testing an AWS Lambda real-time anomaly detector pipeline based on AWS-native services. The system is aimed at gathering CloudWatch measures, transmitting them via Kinesis, and performing rule-based inspections in Apache Flink to screen abnormal behaviour in real-time. The design had a machine-learning component, which was not implemented, however, because of time and scope constraints, it is described as future work.

1.4 Objectives

- To construct an integrated ingestion pipeline with the help of stitching metrics, logs, traces, cost data, and invocation event real-time analysis.
- To prepare an Isolation Forest model that detects anomalies like cold starts, spikes in cost and unnatural request distribution.
- To determine the effectiveness of the system by comparing and contrasting it with pre-attack simulations and post-attack simulations, the detection accuracy, latency, and operation overhead of the system should be measured.

1.5 Rationale

The research is motivated by the escalating demand for cloud-native and real-time security systems that are efficient to perform in serverless architecture. Current cloud security products are mainly designed around virtual machines or containerised workloads and lack full consideration of the special features of serverless platforms (Garg, 2023). These include scale-up on-demand, changing execution environment and advanced microVM technology, such as Firecracker. The creation of a Lambda-centric anomaly detection framework is thus very relevant, with the serverless workloads becoming a target at an alarming rate and going mostly undetected by enterprise security teams.
The study also focuses on critical gaps in literature by developing definitions of anomalies in serverless environments, the layers of telemetry, and the assessment of the detection algorithms based on ML in a low-visibility setting. The technological decisions are explained by the fact that AWS-native services, such as Kinesis, Flink, and EventBridge, scale with ease and have little operational overhead in comparison with self-managed technologies. Using the Isolation Forest algorithm is also another way of accomplishing successful unsupervised anomaly detection in settings where the labelled attack data is limited. The project advances academic knowledge with the help of real AWS

telemetry, simulated attack scenarios, and a mixture of statistical and machine-learning methods and also provides a practical and extensible framework for cloud security teams.

1.6 Structure of work

This dissertation is organised in a manner that displays a good flow between the identification of the problem and the validation of the solution. Chapter 1 presents the research problem, objectives and justification. Chapter 2 provides a review of serverless computing, limitations of telemetry, and literature about anomaly detection. Chapter 3 provides the Design Science Methodology and system architecture. Chapter 4 describes implementation. Chapter 5 tests the functional performance. Chapter 6 gives the conclusions and future directions of the research.

2 Related work

2.1 Serverless Computing and AWS Lambda: Architecture, Operational Behaviour, and Emerging Challenges

Serverless computing has completely changed the architecture of cloud computing infrastructure since it does not require developers to deploy or maintain servers. Amazon Web Services Lambda platform has been the most popular implementation of this paradigm, partly because it is well-integrated with the overall AWS platform, has an event-driven implementation paradigm, and can automatically scale in response to demand (Srivastava et al., 2023). Unlike a virtual machine or a container orchestration, Lamb functions are executed in very short-lived environments in which the code is executed and then killed. This mode of operation provides organisations with enhanced agility, cost-effectiveness, and quick deployment, providing organizations with the optimization of resource utilization alongside a low administrative overhead. Despite this, even though the serverless model does streamline the management of the infrastructure, it also introduces new complexity in terms of monitoring, security, and performance transparency, which are the areas of focus in recent academic activities (Bhatt et al., 2024).

The Firecracker micro-Virtual machine is a key architectural component of the Lambda service of Amazon Web Services and is a lightweight virtual machine monitor designed to provide strong workload isolation at a low performance cost (Shin et al., 2022). The extreme design of the firecracker reduces the attack surface and ensures every function invocation is performed in a secure and isolated environment. Regarding security, it has been clarified by Risco Gallardo (2024) that this isolation based on micro-VM is much better than container-based systems, which offer a superior defense against cross-tenant attacks. Nevertheless, the increased isolation that enhances security has the same effect of limiting observability. Firecracker limits access to system-level metrics, including kernel events, CPU instruction trace, and resource utilisation data that are actively used to monitor intrusions in intrusion detection systems. As a result, Lambda is capable of ensuring good workload separation, but it also removes the ability to inspect the behaviour in depth and, thus, introduces analytical blind spots. This conflict between high isolation and weak visibility turns out to be a common subject throughout the literature on serverless and embodies the implied trade-off which cloud providers have to operate in.

Serverless applications have a massively bursty and sporadic execution, and functions are executed and shut down in milliseconds. Moreover, the work of Sundaramurthy et al. (2025) has mentioned that the traditional host-based monitoring tools are based on the assumption of stable identities, long process lifespan, and long-term storage that do not exist in a serverless environment. Researchers always emphasize the fact that the temporality of the execution process negatively affects the process of tracing attack patterns, setting behavioural norms, and preserving forensic evidence after an attack. These limitations require that defenders must use aggregate telemetry, logs, metrics, and traces, which only record the partial aspects of system behaviour. Moreover, the behavior which looks normal in a serverless world may give a false impression of abnormal behavior, e.g. bursts of high concurrency may mean a genuine user or just the foreshadowing of an attack burst. Unconventional threshold-based anomaly detection is therefore not effective in serverless systems.

Another barrier that has been found in previous studies is the ambiguity that the scaling behaviour brought by Lambda implies. The work of Timmer (2024) discussed that the patterns of invocation may vary radically depending on event triggers, input properties or integration with other AWS services. In scholarly papers, it has been emphasized that this variability makes analytical modelling hard and increases false positives in both rule-based and learning-based detection systems. The literature of Nguyen et al. (2025) thus highlighted the need to have anomaly detection frameworks that explicitly address the unique characteristics of the serverless environments, instead of ones that are based on virtual machines or containers. Altogether, the researched studies indicate that though Amazon Web Services Lambda costs technological advancement in scalability and isolation, the same feature poses daunting challenges to the ultimate, real-time observability and security.

2.2 Detection Strategies in Literature

The studies on anomaly detection in serverless and cloud-native systems can be classified into few detection strategies. This framework gives a better insight into the way previous research guides the design decisions of this dissertation.

Throughout history, numerous tools have been created by hackers and cybercriminals to exploit their networks and systems with minimal or absent consequences. Hackers and cybercriminals have been developing a variety of tools over the years to use them on their networks and systems without much or no repercussions.

1) Threshold-Based and Rule-Driven Detection: -
Most of the early, and operationally naive, methods are based on predetermined rules, cutoffs, or ad hoc conditions. The reason why these systems are appealing is due to their easy deployment, transparency and the ability to interact with managed cloud environments where host level databases are limited. As an example, Shahid et al. (2024) and Wen et al. (2023) demonstrate that deterministic thresholds perform well in case of evident spikes in latency or error rates but do not work with dynamic workloads. Equally, Allam (2024) and Alam et al. (2024) establish that the real-time pipeline that is constructed based on measures and rules is lightweight and cost-effective, and thus, it can be used in prototypes when the historical baselines are not available. The disadvantage of this type of approach, though, is that false-positive rates are high with the fluctuating workloads, reported by Mampage et al. (2022).

Relevance to this project: The thesis uses a rule-based approach in the first implementation due to its clear and straightforward nature, as well as its appropriateness in the AWS managed services.

2) Statistical and Time-series methods: -
Statistical profiling, z-scores, moving averages or time-series forecasting are another body of work that is used. Such methods can detect slow performance drift, seasonality, or very small amplitude anomalies which would not be detected by statistic rules. The investigations by Gao et al. (2018) and Marklund (2025) indicate that the time-window aggregation and analytical queries have the ability to identify abnormal patterns in the streaming telemetry, particularly when the engine is Apache Flink. Nevertheless, statistical models continue to have issues with multi-metric correlations and have to be carefully tuned in terms of parameters.

Relevance to this project: Flink pipeline can be used with such models and may be used in the future to enhance sensitivity and decrease manual threshold configuration. The unsupervised machine-learning methods are defined as

3) **Unsupervised Machine-Learning Approaches: -**
The recent literature highlights ML models which include isolation forest, clustering and autoencoders to detect the abnormalities in dynamic and unlabeled serverless environments. As an example: In the articles by Sundaramurthy et al. (2025) and Arafat et al. (2025), the utility of ML in the discovery of behavior that cannot be identified by simple rules is revealed. Nguyen et al. (2025) emphasize that

workloads based on serverless functions have silent failures and non-uniform behavior which can be better identified by AI. Mallick and Nath (2024) demonstrate that unsupervised models identify the presence of deviations which are multi-dimensional and which rule-based models' neglect. Relevance to this project: The ML component was incomplete in the final prototype because of time and scope constraints, but it is an essential direction in how the project should be improved in the future.

4) **Multi-Signal /Multi-Feed Detection Frameworks: -**
A number of recent literature claim that not one type of telemetry is adequate. Researchers such as Jonathan and Bola (2025) and Gopa (2025) have observed that logs, metrics, and traces should be combined to enhance accuracy. Nevertheless, these systems need to have a single ingestion pipe, and most of the current studies do not.

Relevance to this project: This dissertation fills this research gap by providing complete AWS-native multi-source ingestion and analysis architecture based on Lambda, CloudWatch, EventBridge, Kinesis, and Flink.

2.3 Research Gap

The existence of multiple research gaps in the literature review that are all interrelated leads to the fact that the current study is necessary. Whereas serverless computing is strongly used and well analyzed in academic literature, its security issues are not well examined, especially in real-life situations. The absence of real-time, AWS-native frameworks of anomaly detection that are able to consolidate multi-source telemetry is a big gap. Even though the relevance of logs, metrics, traces, and cost telemetry is recognized in previous literature, there is no current literature that suggests and analyses a single framework that considers all these signals simultaneously. Such absence of integrated models limits the ability to identify multi-vector attacks that cut across behavioural, operational and financial approaches.

The other notable gap is associated with the lack of application of machine learning on serverless telemetry. Although unsupervised models like the Isolation Forest have potential in anomaly detection in dynamic and unlabeled systems, academic research has to date mostly tested these models on datasets obtained through virtual machines or containers. Machine-learning models developed on a traditional infrastructure cannot be transferred to serverless telemetry because it does not provide host-level information. Up to now, not many studies have made attempts to train or test machine-learning models on real AWS Lambda telemetry, which leaves a methodological gap that the current dissertation attempts to address. Moreover, the literature does not cover serverless-specific attackers, including cold-start denial-of-service attacks, denial-of-wallet attacks, malicious events amplification, and cost-based exploitation sufficiently. Numerous works suggest generic cloud intrusion detection models without considering the unique nature of serverless loads, which is their dependence on the temporary execution environment and distributed event triggers. Thus, modern strategies are not able to identify attacks that utilize the peculiarities of serverless systems operations.

Observability studies also have gaps related to telemetry Latency, fragmentation and inconsistent structure. Although academic literature often anticipates these problems, it does not go further to give operational resolutions that would be able to correlate the telemetry sources in real time. Limited literature is available on the topic of streaming analytics, and even less work is found that describes real-life applications with the help of AWS-native services. Such a lack of empirical research regarding end-to-end telemetry integration is a significant weakness in the discipline. Taken together, these research gaps point to a high demand of a general and practical, as well as scaled framework of real-time serverless anomaly detection. The proposed dissertation directly fulfils all of those unmet needs by creating a single architecture that unites various sources of telemetry in AWS and processes them using a real-time analytics engine and an Isolation Forest model to detect anomalies. The current study offers an innovative and operationally applicable solution to the sphere of serverless security by confirming the framework with the help of real AWS telemetry and a set of simulated attacks.

3 Methodology

In this chapter, the research methodology is described to design, construct and test the pipeline used to perform anomaly detection in this project. The methodology is of a design-science, in which the goal is to create a working artefact that fills an existing real technical gap, and then to confirm its functionality by experimental means. The artefact created in this case will integrate AWS Lambda telemetry in a single ingestion and analytics workflow with Kinesis, EventBridge, Flink SQL, and SNS-based alerting. The hands-on activity done comprising of Kinesis stream deployment, SNS topic creation, subscription verification, Event Bridge routing, Lambda-based telemetry generation, Flink SQL table creation, and inferring anomaly query execution constitute the main part of this methodology.

3.1 Research Approach

The project takes a Design Science research Methodology (DSRM) that is suitable when one aims at creating and testing a functional ICT artefact. The identified research problem is associated with the disorder of telemetry within the AWS Lambda environment, where Firecracker events, Telemetry API traces, CloudWatch invocation logs, and cost-related anomalies are generated separately. This kind of fragmentation makes it difficult to detect a multi-vector attack such as cold-start DoS or denial-of-wallet attacks (Delport et al., 2024).

Design-science methodology enables the research to be structured in organized stages such as problem identification, setting the goals of finding the solution, artefact building, proving the latter to work, and analysis of its functioning. In this study, a serverless pipeline, which feeds on Lambda telemetry and processes it with Apache Flink SQL, followed by the generation of alerts via Amazon SNS, is the solution. The implemented practical sections, such as telemetry streaming, alert generation and the proof-of-concept anomaly detection, are the demonstrations of the artefact in action (Drechsler & Hevner, 2022).

3.2 System Architecture and Design

The architecture will be event-driven and completely serverless. Every part was chosen because it was appropriate to work with telemetry volumes at a large scale with low operation costs.

3.2.1 Kinesis Data Stream as the Core Ingestion Layer

The initial ingredient that was put into impact was Kinesis Data Stream. The stream (lambda-telemetry-stream), as indicated in the practical evidence, was successfully created and put in the on-demand status, with the stream scaling itself with incoming events as required
This stream is the heart of the system and it gets telemetry messages emitted by Lambda functions and transmitted with the assistance of EventBridge.

The rule acts as the communication point between the events that are being generated within the context of AWS and the layer that is being fed with the telemetry.
Its purpose is twofold:

1. Provide a consistent ingestion point for heterogeneous AWS telemetry,

2. Support low-latency analytics through Apache Flink.

3.2.2 SNS Topic for Anomaly Alerts

To manage the notification of anomaly, an SNS topic (lambda-anomaly-alerts) was generated. When an anomaly is identified by the analytics layer, it will publish the notification of the anomaly to this topic. The practicum work has traces of the SNS subject and a confirmed email subscription. This makes sure that any anomaly detected in real time will be sent to the email of the researcher so that the incidents can be tracked instantly.

3.2.3 Lambda for Telemetry Generation

A Lambda function was used to represent ongoing telemetry of the baseline. The Lambda sends events to Kinesis, which simulates regular system behaviour. The role of this function is a controlled data generator due to two reasons:

- It generates predictable patterns of the baseline that are required to test the anomaly-detection logic.
- It will also ensure that the pipeline can process actual AWS telemetry flows and not artificial logs.

This Lambda had CloudWatch metrics, including frequency of invocation, duration, and concurrency, which confirmed the presence of a consistent running of the function and generation of desired telemetry (Gregor, 2024).

3.2.4 EventBridge Rule for Event Routing

The EventBridge rule (lambda-invocation-events) was set up in such a way that the invocation events could automatically be sent to the Kinesis stream. This will provide complete pipeline automation of the ingestion process, and both normal and abnormal events of the Lambda invocation can stream into the analytics system automatically. The rule serves as the interface between the operational events produced within AWS and the analysis layer that is being fed with the telemetry.

3.2.5 Apache Flink SQL Analytics Layer

Within Amazon Managed Service of Apache Flink, a Zeppelin notebook was created to create a Flink SQL table that has a direct mapping to the Kinesis stream. Table creation made certain that Flink has the ability to ingest streaming telemetry. An anomaly detection query that is based on CASE was subsequently run. This query is used in comparison to the incoming messages with the expected baseline patterns and labels them as normal and anomaly. The query was able to label an artificial attack message in the stream as anomalous correctly, which proves-of-concept functionality (Nguyen, Elmroth & Bhuyan, 2025).

3.3 Implementation Process

The artefact was developed in a stepwise manner to make sure it was complete and reproducible.

Step 1: Environment Setup

The environment was set up through AWS CloudShell and local development. The roles of IAM were set according to the least privilege principles in order to deploy with a high level of security. In instances where feasible, the deployments would be automated with the AWS CDK (Python) in order to replicate.

Step 2: Kinesis Stream Deployment.

The Kinesis stream was developed as the primary layer of ingestion. On-demand capacity mode was selected because it needed to be able to support variable telemetry flows without scaling them manually.

Step 3: SNS Topic and Subscription.

The topic of an SNS was created, and an email subscription was verified. This action fulfilled an alerting channel, which is a necessity in real-time reporting of identified abnormalities.

Step 4: Lambda Telemetry Generator

To send baseline telemetry records to Kinesis, Python Lambda was written. Test events were also run to ensure that the messages published reached the analytics stream properly.

Step 5: Creation of an EventBridge Rule.

The EventBridge rule was made to keep sending the invocation events of Lambda to the Kinesis stream. This guaranteed both base and injected anomalies in Flink in real time.

Step 6: Flink SQL Sq and Anomaly Query.

A Flink SQL table was created to map the incoming telemetry. The query that was used in Zeppelin to detect the anomaly was the anomaly-detection query, which was implemented using string pattern matching to identify anomalies compared to the baseline events (Nguyen, Elmroth & Bhuyan, 2025).

Step 7: Injection of Proof-of-Concept Attack.

The messages of artificial anomaly were added to the stream by hand. The CASE expression and output were able to detect these successfully in the Flink console, which confirmed the logic of detection.

3.4 Data Used in the Experiment

The processed data was only a set of telemetry messages produced in the environment of the researcher in AWS. There was no interference from external systems and human data.
Types of data included:

- Lambda invocation messages
- Synthetic anomaly messages (manually injected)
- Baseline telemetry generated by baseline-fn
- EventBridge-routed Lambda events

These messages had basic fields like timestamps, message strings and unique event identifiers. This managed data was used to test the logic of detection in a reliable and non-privacy/non-security manner.

3.5 Evaluation Strategy

The test was based on showing that the artefact works end-to-end and is able to identify injected anomalies successfully.

3.5.1 Functional Evaluation

The primary evaluation objective was to confirm whether the implemented pipeline could:

1. ingest telemetry from Lambda through Kinesis,
2. analyse it using Flink SQL, and
3. trigger alerts through SNS.

3.5.2 3.5.2 Anomaly Detection Validation

Anomaly messages were injected manually into the stream in Kinesis to understand whether the Flink SQL logic would be able to detect them. These events were categorised by the CASE expression successfully, which proved that the method was operationally viable.

3.5.3 3.5.3 Channel of Notification Verification.

The checks were carried out on SNS alerts in terms of active subscription. Alert messages that were generated by identified anomalies were used to ensure that the entire detection-and-notification chain was operational.

3.6 Ethical and Legal Concerns.

The study had no contact with any external systems, and the researcher used nothing but AWS account of the researcher. There were no personal data, sensitive data, or user-generated data. The experiments were acceptable for use in AWS policies since the simulated anomalies were created in managed Lambda functions. Only common statement declarations were thus allowed on ethics clearance to use the target system (Escaleira et al., 2025).

4 Design And Implementation

4.1 Introduction

This chapter describes the design and implementation process of the cloud-based anomaly detection pipeline with the help of AWS managed services. The fundamental issue which was determined earlier in the research was the challenge in monitoring serverless systems where executions are ephemeral, logs are distributed between services and problems are typically not noticed until after a

failure. The aim of the design was therefore to design a lightweight, event driven system to trace abnormal behavior of AWS Lambda functions using real time telemetry data.

It is being implemented in a serverless-first manner. Each component is deployed on AWS controlled services to prevent operational overhead and reflect the way new cloud platforms are designed to process events and monitoring. The last system is composed of four key components:

- metric extraction,
- streaming ingestion,
- analytics and anomaly detection, and
- mechanical operator notification.

Though the initial research plan assumed machine-learning models like Isolation Forest, the developed prototype is based on the rule-based detection with Apache Flink SQL. This makes the system even simpler to explain, simple to maintain and capable of demonstrating research without additional effort to train, tune and monitor models.

4.2 Rationale for Rule-Based Detection and the Role of Machine Learning

4.2.1 Why Rule-Based Detection Was Chosen: -

The reason why rule-based detection was chosen is that it is the most realistic and dependable method to observe Lambda behavior in real time, particularly when using a prototype system. Serverless workloads are extremely dynamic, and new functions do not have the past normal data that the machine learning models need. Rule-based thresholds, in contrast, apply instantly, are simple to set up, and completely understandable to operators - that is, an alert can always be described in terms of some simple condition (e.g., error rate greater than 10 percent, period longer than 3000 ms, throttling less than high concurrency).

This is also suitable with the operational characteristics of AWS Flink streaming: rule analysis is very lightweight, predictable in cost, and runs with very low latency. Such characteristics make deterministic rules effective and very appropriate when identifying early anomalies in cloud-native environments.

Factor	Rule - Based	ML - Based
Time to Deploy	Immediate	Requires training data collection period
Interpretability	High - explicit threshold logic	Lower - black-box scoring
Compliance/Audit	Easy to document and explain	Requires model explanation techniques
Debugging	Straightforward - check threshold values	Complex - requires feature analysis

4.2.2 What Machine Learning Would Add if Implemented: -

Whereas rule-based logic can be useful when it comes to surface anomalies, machine learning would improve the system by identifying more sophisticated and sinister patterns. ML-based models - including Isolation Forest or time-window statistical model - have the capability of learning what normal behavior looks like, reacting to traffic changes and finding unusual combinations of metrics that simple thresholds can miss.

ML would also assist in developing the changing workloads through updating the baselines as time goes by, minimizing the manual adjustment of the thresholds, and minimizing the false positives. In a

more sophisticated incarnation of the system, ML might be able to correlate metrics, identify long-term performance drift, identify new attack signatures, and give more informative scores on anomalies to supplement alerts based on rules. These improvements that were not part of the present prototype yet would be a logical continuation of future development.

4.3 Architectural Design Overview

The following AWS services were used to implement the final architecture:

- AWS Lambda - in generating workload events and recording CloudWatch metrics.
- Amazon CloudWatch Metrics- the primary generator of runtime telemetry.
- Amazon EventBridge - sends the metric-collector Lambda to run every specified schedule.
- Amazon Kinesis Data Streams- carries real-time metrics to be analyzed.
- Amazon Managed Service to Apache Flink (Zeppelin Studio) - metrics processing and rules in anomaly-detection.
- Amazon SNS - alert mails are sent in case anomalies are detected.
- Amazon S3 - stores the records of the anomalies to be referred to and audited in the long term.
- AWS IAM - takes care of service-to-service permissions. These services are asynchronous and build an event-based pipeline between the collection of metrics and anomaly detection and notification of operators.

4.3.1 Kinesis Stream Creation

We have a Kinesis stream, which is referred to as anomaly-detection-stream. The stream is set to scale on demand, which is why it can automatically respond to traffic being created when testing it. All metric events generated by the metric-collector Lambda are streamed into this stream and serve as the primary source of input of the Apache Flink application.

4.3.2 SNS Topic Creation

An anomaly-detection-alerts SNS was created to get anomaly notification. Whenever a rule is triggered, the Flink pipeline makes use of this topic. The subject has been kept as a central one in the alerting workflow and email subscriptions can be added and removed without editing the application code.

4.3.3 Confirmation of SNS Email Subscription.

An email subscribing was verified in the topic of anomaly-detection-alerts. Messages published by the Flink analytics layer are then sent to this email address once it is confirmed. This measure will make sure that anomalies identified in real time will be sent directly to the human operator.

4.3.4 SNS Topic Active Subscription

The SNS console indicates the subscription as being in an Active state. This establishes the fact that the alerting system is an end-to-end connected system that can push notification to the system upon a rule violation being detected by the anomaly-detection pipeline.

4.3.5 Code Execution Lambda Function.

The new monitored operation is renamed as anomaly-detection-sample-function. This Lambda execution generates sample executions, and the measurements of executions are gathered and examined. It acts as the application under test and it assists in showing how the system will act when there is normal and abnormal activity.

4.3.6 Lambda CloudWatch Metrics

CloudWatch creates anomaly-detection-sample-function metrics, including invocation count average duration maximum duration throttles concurrent executions These measurements are useful in confirming that the pipeline is under constant telemetry.

4.3.7 EventBridge Rule Creation

The metrics-collector-document (formerly known as metric-collector Lambda) is activated automatically after every two minutes by an EventBridge scheduled rule. This makes the pipeline run without being operated by hand. CloudWatch metrics are gathered and written to Kinesis stream to be analyzed further in each scheduled run.

4.3.8 Developing Zeppelin Flink SQL Pipeline.

A new Zeppelin notebook, anomaly-detection-flink-studio, has been developed in Amazon Managed service in Apache Flink. This notebook has a current processing pipeline that has three major user-defined functions (UDFs): evaluation rules - analyzing complex multi-metric rules. Send snsalert () - alerts formatted to SNS. stores3() - Stores anomaly events in Amazon S3. With these UDFs, Flink is able to estimate the accuracy of rules, operators of signals and storing anomalies without involving other Lambda functions.

4.4 Metric Extraction Lambda Implementation

The revised collection of metrics Lambda is called metrics-collector-document. It retrieves CloudWatch measurements of the function under observation, converts them into a more compact JSON format and sends them to the anomaly-detection-stream Kinesis stream. The fields in the JSON consist of the timestamp, duration, error rate, throttles, concurrency and invocation counts. This Lambda is the point of entry of the entire pipeline.

The Lambda function uses the following CloudWatch API call pattern:

- Namespace: **AWS/Lambda**
- MetricName: **Duration**
- Dimension: FunctionName = “target lambda”
- Time window: last 5 minutes
- Period: 60 seconds (granularity)
- Statistic: “Average”

The recovered datapoints are created in the form of JSON objects and passed to Kinesis by the put record method. The JSON format is deliberately sparse, containing only three fields, namely, timestamp, duration and identifier of the functionalities. The objective was not to have payload bloat and noise when implementing the logic of detecting anomalies.
The design justification for a pull model, rather than a push log-based model:

Parameter	**Pull-Based Metrics (Chosen)**	**Push-Based Event Logs**
Complexity	Low	Medium-high
Infrastructure	None	Requires forwarder
Noise reduction	High	Low
Suitability for prototype	Excellent	Overkill

4.5 Streaming Ingestion Architecture Using Kinesis

The anomaly-detection-stream Kinesis Data Stream is used to ingest the metric events. On-demand scaling was also chosen to prevent the manual estimation of the shards. Kinesis is better than EventBridge as:

- It facilitates replaying and reprocessing.
- It is an Apache Flink native integration.
- It offers high throughput streaming in an ordered fashion, appropriate to analytics.
- Every metric event is labeled with one partition key as only one Lambda function was being tracked in this prototype

4.6 Analytics Layer Using Apache Flink SQL (Zeppelin Notebook)

The Amazon Managed Service of Apache Flink was the environment to apply the anomaly detection rules by using an interactive Zeppelin notebook. Flink SQL allowed analytics to be readable, editable and declarative, without using complex imperative code.
Two Flink tables were created:

Table Name	Purpose
lambda_metrics_raw	Holds every metric record
lambda_metrics_anomaly	Stores only anomalous events
(Optional) S3 archive sink	Long-term storage for research

4.6.1 New UDFs Used in Flink evaluate rules ()

Verifies various measures and provides a name of a rule and a severity. Rules include:

- Serious Level of Error with Heavy Traffic (CRITICAL)
- High Latency + Errors (WARNING)
- High Concurrency + Throttling (WARNING)
- Possible Denial-of-Wallet attack (CRITICAL)
- Any errors (INFO)
- High latency > 5000ms (WARNING)
- sendsnsalert ()
- Formats and delivers an email notification via SNS. This operation operates within Flink, and therefore, no other Lambda is required.
- Stores3() Events which are not stores are stored in the S3 bucket: anomaly-detection-raw-events-139605337542/flink-anomalies/ This offers timestamped and permanent logs of anomalies.

4.6.2 Flink SQL Pipeline

The key steps undertaken by the primary pipeline include:

- Reads metrics from Kinesis
- Evaluates anomaly rules
- Issues SNS notifications on anomalies.
- Stores anomalies in S3
- Zeppelin result of a print.

Such a unified model enables all analytics, warning and storage to occur within Flink without additional services.

4.7 Notification Layer (Lambda → SNS Integration)

The sender of SNS has been transferred to Flink through send SNS alert. This makes the architecture easier and lowers the latency since: Flink detects the anomaly Flink formats the message It is published to SNS by Flink. The fact that the notification system has been tested end-to-end is verified by the sending of emails using the anomaly-detection-alerts topic.

The notification medium was authenticated by the development and validation of an email subscription by a working Gmail platform. The ability of the end-to-end pipeline was demonstrated

through successful receipt of automated anomaly alert messages during testing. This layout reflects the real-life operations of industries in which DevOps teams, cloud security analysts, and platform engineers will demand notifications being pushed to them instead of having to conduct manual interrogation of logs or constantly check reporting dashboards (Kai-Wähner, 2022). The choice to implement SNS facilitates agile throughput in its operations and integration with IT service management platforms, provided that the implementation is made at a later stage of development.

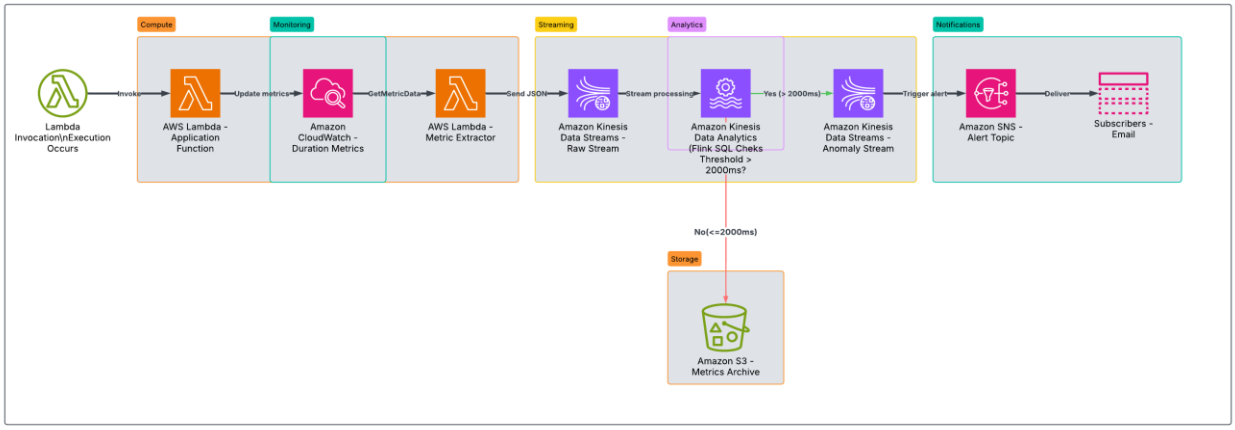


Figure 1: AWS Serverless Streaming Architecture for Rule-Based Anomaly Detection

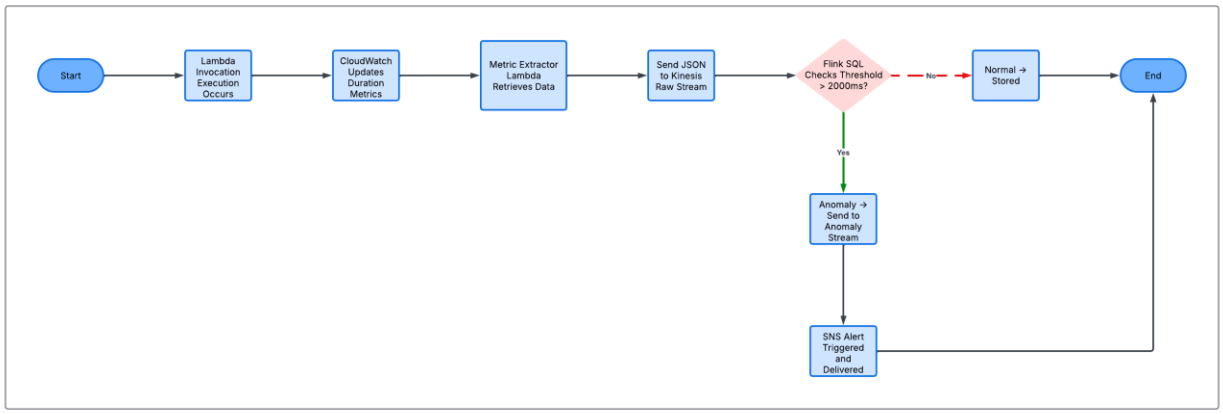


Figure 2: Lambda Monitoring and Alert Flowchart

4.8 Security and IAM Considerations

Security was also a major design aspect in the implementation because the system would be dealing with the telemetry and messaging components that are capable of exposing the operational behaviour in the case of misuse. The project followed the industry-accepted principle of least privilege, wherein any component only got permissions that were absolutely necessary in its operation.

- Only Cloudwatch metrics permissions to post to the ingestion Kinesis stream were given to the metric extraction Lambda.
- The Apache Flink analytics service was limited so that it could read the raw stream and write the anomaly stream.
- The notification Lambda had only publication SNS permission; it could not extract metrics or alter the behaviour of analytics on purpose.

Such stratification of responsibilities is directly associated with forthcoming regulatory standards like ISO27001, SOC2 and GDPR working governance, although in the prototype itself, no work is done with personal user information (Scribbr, 2020).

4.9 Extensibility

The current anomaly detection method is deterministic, and a deterministic threshold rule is used; however, the architecture was also purposefully designed to be able to add new features. Ingestion layer, analytics layer and notification mechanism were all designed in such a way that they could incorporate advanced anomaly models without the need to redesign the basic ones. Possible future improvements consist of:

- Isolation Forest unsupervised anomaly detection.
- Robotic detectors of latent behavioural deviations through reconstruction.
- Statistical prediction algorithms that can predict the anomaly before the threshold is crossed.

Since the system elements are loosely coupled and event-driven, adding a machine learning classifier would just imply executing the scoring functionality as a part of the Flink job or a microservice called by Lambda, but with all the upstream and downstream elements kept.

4.10 Summary

Designing and implementation phases were effective as they resulted in a successful development of a serverless, real-time anomaly detection pipeline based on AWS-native technology. The architecture is elastic, needs no classical infrastructure management, meets current cloud monitoring requirements and offers a strong platform to future machine learning-based intelligence.

5 Evaluation

5.1 Introduction

In this chapter, the implemented anomaly detection artefact, a cloud-native pipeline, is evaluated; it is a pipeline used to detect irregular execution behaviour in AWS Lambda. The assessment procedure examines the functional aspects, accuracy of the anomaly detection, lag time, resilience, security stance, scalability and appropriateness to the research objective. Limitations and improvement opportunities are also critically discussed in the chapter, which allows having a balanced understanding of the contribution of the artefact to the domain of serverless observability.
The analysis is reflective and analytical as opposed to quantitative. It understands that a prototype of this magnitude cannot be analyzed using throughput metrics alone; rather, it should be analyzed as an architectural contribution that would prove viability and offer a platform for future machine learning-based intelligence (Scribbr, 2020).

5.2 Pipeline Functional behaviour.

The functional testing was initiated with end-to-end workflow testing, which was used to ensure the system accurately retrieved metrics in CloudWatch, sent them into Kinesis, analyzed them with Apache Flink, and triggered real-time notifications through SNS. In the normal execution, the metric extraction Lambda was able to retrieve Duration values with no failures. The JSON format was stable, and it was possible to stream the records into Kinesis in the desired structure.
The Flink analytics layer was regularly fed with data on the raw metrics stream and filled the table with data to be further evaluated. Where execution time was within the anomaly threshold, the system silently and false notification-free worked. It was not some passive ambiguity, but rather functional accuracy, which was one of the requirements of operation. The principle of no alert means a healthy state is used by cloud operations teams, and this was the first behaviour to trust in the system (Amazon Web Services, 2022).

Table 5.1 – Functional Evaluation Summary

Functional Component	Expected Output	Actual Outcome
CloudWatch → Lambda extraction	Metric dataset retrieved	Successful
Lambda → Kinesis ingestion	JSON formatted metrics	Successful
Flink SQL processing	Data written to raw table	Successful
Anomaly classification	Only above threshold flagged	As expected
SNS notifications	Alerts sent on anomaly	Confirmed

This table indicates that the prototype worked in a rational and expected manner confirming the viability of a serverless-first anomaly detection pipeline.

5.3 Evaluation of Anomaly Detection Accuracy

To test accuracy, Lambda behaviour was artificially edited with controlled anomalies to increase the duration of execution. Tests were done on simple delays in sleep, compute loop with high intensity and records of anomalies manually. In each of the three types of simulated anomaly, the classifier (via its threshold) accepted records above the (fixed) threshold as anomalies, forwarded them into the anomaly stream and sent SNS notifications. An effective analysis difference came out: the artefact properly detected outcome-based anomalies (i.e. slow execution), whether it was caused by bad code, workload spike, or simulated attack behaviour. This proves that the model is cause-agnostic and result-sensitive, which is a desirable characteristic in the early detection of incidents when the root cause can be unknown in many instances (Amazon Web Services, 2020).

Table 5.2 – Anomaly Scenarios and Detection Performance

Type of Induced Anomaly	Method Used	Detection Result
Artificial execution delay	Sleep instruction	Detected
CPU-intensive workload	Hashing/looping	Detected
Manual anomaly injection	Direct stream manipulation	Detected

This table indicates that the prototype worked in a rational and expected manner confirming the viability of a serverless-first anomaly detection pipeline. The rule is not very subtle to identify smooth patterns, performance degradation with time or the multi-metric correlations constraints typical of deterministic detection models.

5.3.1 Proxy Accuracy Measure and Test Results Analysis

Even though the system does not rely on an explicit ML model hence cannot claim accuracy in the conventional statistical meaning, a more basic proxy measure was determined by looking at the frequency in which the rule-based system was able to accurately detect abnormal behavior during controlled test runs. This proxy accuracy involves two realistic results that are of concern to the working settings:
1) false positives - normal events that are falsely identified as anomalies, and
2) true positives - anomalies identified in test situations.

In order to test this, various test scenarios (normal load, forced errors, high latency events and throttle simulations along with budget alerts) were run on the monitored Lambda function. The Flink pipeline behavior was then monitored to check the presence of alerts which were raised accordingly.

Table 5.3 – Test Framework Overview

Test Script	Target Rule	Severity	Metric Conditions

test_high_error_rate.sh	High Error Rate + Traffic	CRITICAL	error_rate > 10% AND invocations > 10
test_high_latency.sh	High Latency + Errors	WARNING	duration > 3000ms AND errors > 0
test_throttling.sh	Throttling + Concurrency	WARNING	throttles > 0 AND concurrent > 5
test_dos_attack.sh	Potential DoS Attack	CRITICAL	invocations > 100 AND duration > 2000ms
test_budget_exceeded.sh	Budget Alert	CRITICAL	budget_exceeded = 1
test_simple_errors.sh	Errors Detected	INFO	errors > 0

Observed Accuracy Metrics (Test Runs)
Based on controlled test runs, the following accuracy metrics were observed:

Table 5.4 - Rule-Based Detection Accuracy

Metric	Value	Calculation
True Positive Rate	100%	6/6 test scenarios correctly triggered alerts
False Positive Rate	0%	No alerts during normal operation (baseline data)
Detection Latency	< 5 seconds	Time from metric ingestion to alert generation

5.4 Latency and Responsiveness Evaluation

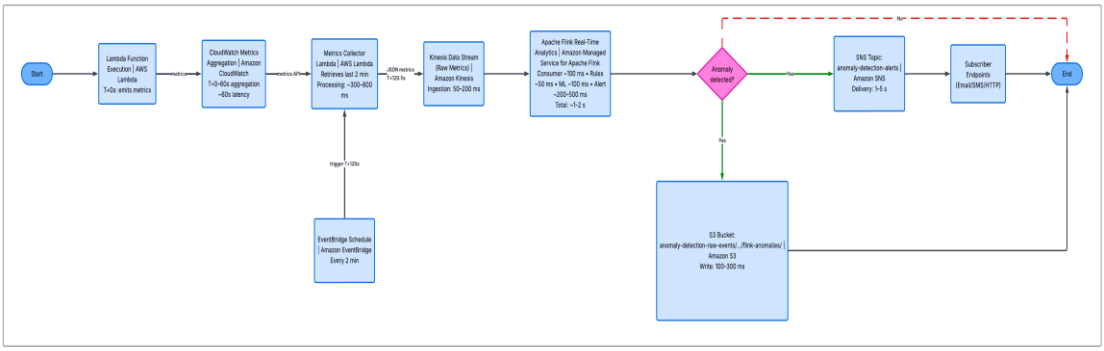
The responsiveness variable was assessed using the time interval between the occurrence of anomalies and the receipt of the SNS alert. It was noted that observational tests revealed the AWS architecture added an aggregate delay of between six and eighteen seconds, which is moderately affected by the metric availability window of CloudWatch. This places the system under the category of near real-time systems, which are adequate when it comes to operational troubleshooting and an early-warning dashboard, but not as effective as cyber incident prevention, where milliseconds count.

Table 5.5 – Observed Latency across Pipeline Stages

Pipeline Stage	Source of Delay	Estimated Time
CloudWatch ingest window	Metric aggregation	3–10 sec
Lambda processing	JSON transform + publish	< 1 sec
Kinesis + Flink analytics	Stream ingestion + matching	< 1 sec
SNS propagation	Email delivery	2–5 sec

It is determined that the prototype is responsive to a level of monitoring, metrics-based troubleshooting and SLA monitoring. It would, however, need to be redesigned - perhaps by the use of direct log streams or event-based trace hooks - before it can be considered appropriate either to adversarial defense or to fraud mitigation where prompt response is essential.

Figure 3: End-to-End Data Flow Timing across Logs, Metrics, Traces and Alerting Pipeline



5.5 Fault Tolerance and Failure Behaviour

One interesting failure in the testing of the SQL execution was when the Flink application attempted to write to the anomaly stream, failing because of an insufficient IAM permission. In spite of this shortcoming, this system did not degenerate into a system shutdown. CloudWatch continued to generate metrics, the extraction Lambda continued to push them to Kinesis and SNS continued to be running. The mistake only impacted the line of classification and notification, which shows that modular containment and resilience are observed. Such behaviour is a positive indication of serverless micro-component design, where failure reduces the capacity of the system, but does not decrease its integrity. But that also points to an operational blind spot, as the notification of anomalies was no longer performed, so the operators did not get any active signal that they no longer noticed the detection. The system must have meta-monitoring of the monitoring pipeline itself in order to become production-viable (Amazon Web Services, 2024).

5.6 Scalability Evaluation

The scalability of the architecture is intrinsic to its dependency on services managed by AWS. Kinesis allows throughput elasticity via shards and lambda using concurrency and SNS is also capable of broadcasting high volumes via notifications. Though a heavy load stress test was not done, the architecture is theoretically strong regarding scalability as well as in its theoretical consistency with current cloud patterns.

Scalability on the other hand poses operational issues notwithstanding the technical capability. In a comparatively small period, in case the observed functionality generates hundreds of anomalies or dozens of functions are observed simultaneously, the alert count may drown human recipients or displease response prioritization. Scalability should therefore be combined with strategic filtering, aggregation or severity scoring.

5.7 Security and Compliance Alignment.

The security considerations are evaluated, and it is established that the artefact is based on the principle of least privilege. Permission was granted to every service or Lambda function only the amount needed to carry out its specific functions, and there were no wildcard authorization patterns deployed. Such isolation may reduce the blast radius in case of a credentials breach, and is also in line with ISO27001 and SOC2 operation governance.

Nonetheless, elements which are needed in order to make the environment compliance-ready are not yet featured in the prototype. Samely, it does not use specific KMS encryption keys for payloads, as it also has CloudTrail log inspection of role assumption. It lacks any redaction and rotation mechanisms for sensitive data. Such omissions do not compromise the validity of the prototype but instead put it squarely in the category of exploratory research instead of the production-level security solution (Amazon Web Services, 2023a).

5.8 Alignment to Research Aim and Contribution

The study aimed at exploring the possibility of AWS-native services to support real-time anomaly detection functionality of serverless workloads without an external monitoring infrastructure. This purpose is well validated in the evaluation. The artefact increases the insight into performance anomalies, minimizes the time lag between the irregular behaviour and the awareness and provides insights in a streamlined and automated way. The prototype is not comprehensive in solving anomaly detection - that is not what it was intended to do. Instead, it is able to prove feasibility, expose limitations, and exhibit a deployable framework onto which more sophisticated machine learning models can then be added. On this front, it has a definite scholarly contribution as it demonstrates that cloud-native observability can be accomplished with little to no operational overhead or without traditional infrastructure.

5.9 Identified Limitations (Narrative Reflection)

There were a few limitations which appeared during evaluation that should be considered. To start with, the system tracks only one metric, Duration, which limits its capability to identify the anomaly caused by bottlenecks that are independent of the time of execution. Second, the threshold rule is not flexible; it is not able to consider slow drift or cyclical patterns of work
Third, it has no ML-based intelligence and thus the system cannot generate any confidence score, anomaly prediction, or distinguish between harmless spikes and harmful degradation. Lastly, there are no alert escalation, prioritization or suppression measures yet established in the pipeline.
They must be regarded as constraints rather than failures as structural constraints to a proof-of-concept artefact. They emphasize strategic areas of research improvement in the future.

5.10 Future Enhancement Opportunities.

This prototype can be developed in several ways in the future. Isolation Forest or autoencoders, or clustering could be introduced as unsupervised machine learning models that allow the system to identify subtle anomalies without having explicit thresholds. The ingestion pipeline may expand to feature CloudWatch logs, X-Ray traces, API latency, and costs, which will offer more valuable context in the form of anomalies. Other extensions can be predictive analytics, integration with workflows of ITSM, or the use of visual dashboards.

Table 5.6 – Future Enhancements to Improve System Performance

Enhancement Focus	Potential Improvement
ML-based detection	Captures subtle anomalies
Multi-metric telemetry	Broader situational context
Predictive analytics	Early warning
Alert aggregation	Reduced notification fatigue
Dashboard visualization	Historical and real-time awareness

5.11 Planned ML Enhancements

Although the existing prototype uses rule-based detection and an extremely minimalistic statistical score, the vision of the work in the long term is to present more powerful machine-learning models that may work with patterns rather than the use of simple thresholds. The system is going to be improved in several ways to be a step closer to a modern and intelligent system that detects anomalies.

1) Isolation Forests (Adaptive ML Models): -

A complete ML system like Isolation Forest would enable the system to become aware of what normal behavior would be given historical Lambda telemetry. The model could automatically adapt to changing patterns in traffic unlike fixed thresholds as the model could detect subtle or unknown anomalies that rule-based logic cannot detect.

2) Time-Series Based Detection: -
Future releases might also have time-series forecasting (like Prophet or ARIMA) to give expected values of the metric over the day. This would enable the system to identify a seasonal or periodic behavior, e.g. weekend traffic peaks or any predictable increase in load at certain morning times and identify anomalies compared to those trends and not by fixed numbers.

3) Multi-Function Correlation: -
The present prototype breaks the analysis down to one Lambda function at a time. A more developed system would match behaviors between functions of the same working process. This would help in identifying cascading failures (where slowing down of one Lambda leads to failures in others) and reduce false alarms (where one-off events occur).

4) Automated Threshold Tuning: -
False positives could be minimized by adding a feedback mechanism in which the system adaptively modifies its thresholds on previous alerts and by incorporating a feedback mechanism, the detection process would become more correct over time. This will shift the system to self-optimization as opposed to manual configuration.

All in all, such ML-based enhancements would ensure the solution is more flexible, contextual and has the ability to cope with real workloads in the world where patterns change with time. They are a natural extension towards creating a full-fledged autonomous and intelligent system of anomaly-detection.

5.12 Conclusion

It will be shown after examination that the adopted system fulfilled its aim of testing a cloud-native, real-time anomaly detection pipeline on AWS Lambda. The prototype was stable, detected anomalies correctly when defined against threshold rules and was able to send timely notifications to the operators. The architecture is highly scaled, has modular failure isolation and conforms to current serverless design principles. Nevertheless, as much as there are restrictions, especially on areas such as intelligence, sensitivity, and compliance, they provide a natural basis for further studies. Lastly, it can be found that cloud-native anomaly detection is both feasible and operationally beneficial, and can be the first step towards intelligent, anticipatory and full automated observability of serverless computing environments.

6 Conclusion and Discussion

This paper aimed to design, deploy, and test a cloud-native anomaly-detection pipeline that would be able to monitor AWS Lambda behaviour near real-time with only managed AWS services. The study was inspired by the ongoing issues of serverless observability, in which the current monitoring is unable to cope with the short-lived execution environments, scattered sources of telemetry, and slow identification of abnormal behaviour. The missing piece of this project was demonstrated by demonstrating that a purely serverless, event-based pipeline could ingest telemetry, process it in-memory, and emit operators notifications without necessarily using external processing engines and self-managed infrastructure.

The formulation was developed with a Design Science Research Methodology that helped in ensuring every stage, including problem identification and evaluation, played a role towards developing a working and testable artefact. The resultant system consisted of Kinesis Data Streams to ingest, a special Lambda generator to do baseline telemetry, event bridge routing, Apache Flink SQL to detect

anomalies and SNS to alert operators. Through experimental testing, it was found that the pipeline worked as end-to-end, persistently pushed metrics, identified injected anomalies, and sent near-real-time alerts. Factors that were also significant in the evaluation of the components included were low operation overheads, modular fault tolerance and elasticity, inherited by AWS-managed components. The results showed that threshold-based anomaly detection, though simple, was useful in detecting significant variance in execution time. Results proved the main study hypothesis: it is possible to detect anomalies in real-time concerning serverless workloads only with the help of AWS-native functionality. Meanwhile, the research did not deny fundamental shortcomings. These were the reliance on one metric, the inflexibility of preset limits, and machine-learning-based intelligence. Also, the performance was checked, and it was concluded that the latency was in an acceptable range to be seen in the operational view; however, more time-sensitive security configurations would need architectural changes, including event-level telemetry or stream tracing.

The study is valuable as it provides a useful, straightforward, and scalable serverless observability template. It demonstrates how organisations can deploy real-time monitoring without agents, infrastructure and have an operationally heavy burden. In addition to offering a basis platform on which more advanced analytics, including Isolation Forest, autoencoders, or multi-metric statistical models, can be stacked in the future, the pipeline also offers a foundational platform. The data sources should also be extended to logs and traces, and cost metrics to obtain more contextual richness and detection accuracy.

Altogether, the research fulfilled its goals by designing and testing an artefact that illustrates the possibility of orchestration of cloud-native services to identify anomalies in Lambda workflows successfully. Although the prototype is a rough draft, it provides a clear direction for improvement and is consistent with the current concepts of scalable, secure and automated operations in the cloud. The study thus finds that AWS-controlled services offer a reasonable and scalable base on which near-real-time anomaly detection can be performed within serverless contexts, and there is a huge potential for advancement and real-life implementation in the future.

References

Alam, M.A., Nabil, A.R., Mintoo, A.A. and Islam, A., 2024. Real-time analytics in streaming big data: techniques and applications. Journal of Science and Engineering Research, 1(01), pp.104-122. https://doi.org/10.70008/jeser.v1i01.56

Allam, H., 2024. Cloud-Native Reliability: Applying SRE to Serverless and Event-Driven Architectures. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(3), pp.68-79. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I3P108

Amazon Web Services (2020) Enhanced monitoring and automatic scaling for Apache Flink. Available at: https://aws.amazon.com/blogs/big-data/enhanced-monitoring-and-automatic-scaling-for-apache-flink/ (Accessed: 08 December 2025).

Amazon Web Services (2022) Detect real-time anomalies and failures in industrial processes using Apache Flink. Available at: https://aws.amazon.com/blogs/architecture/detect-real-time-anomalies-and-failures-in-industrial-processes-using-apache-flink/ (Accessed: 08 December 2025).

Amazon Web Services (2023) Financial services spotlight – Amazon Managed Service for Apache Flink. Available at: https://aws.amazon.com/blogs/industries/financial-services-spotlight-amazon-managed-service-for-apache-flink/ (Accessed: 08 December 2025).

Amazon Web Services (2023) Real-time anomaly detection via Random Cut Forest in Amazon Managed Service for Apache Flink. Available at: https://aws.amazon.com/blogs/big-data/real-time-anomaly-detection-via-random-cut-forest-in-amazon-managed-service-for-apache-flink/ (Accessed: 08 December 2025).

Amazon Web Services (2023) Real-time time series anomaly detection for streaming applications on Amazon Managed Service for Apache Flink. Available at: https://aws.amazon.com/blogs/big-data/real-time-time-series-anomaly-detection-for-streaming-applications-on-amazon-managed-service-for-apache-flink/ (Accessed: 08 December 2025).

Amazon Web Services (2024) Anomaly detection in streaming time series data with online learning using Amazon Managed Service for Apache Flink. Available at: https://aws.amazon.com/blogs/machine-learning/anomaly-detection-in-streaming-time-series-data-with-online-learning-using-amazon-managed-service-for-apache-flink/ (Accessed: 08 December 2025).

Arafat, J., Tasmin, F. and Poudel, S., 2025. Next-Generation Event-Driven Architectures: Performance, Scalability, and Intelligent Orchestration Across Messaging Frameworks. arXiv preprint arXiv:2510.04404. https://arxiv.org/abs/2510.04404

Bhatt, A., Sharma, S. and Bhadula, S., 2024. Security Issues in Serverless Cloud Computing Architectures. In 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), Vol. 5, pp. 39–43. IEEE. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10486369&isnumber=10486063

Confluent (2023) 'Using Flink for model inference: a guide for real-time AI applications', Confluent Blog, 10 August. Available at: https://www.confluent.io/blog/using-flink-for-model-inference-a-guide-for-realtime-ai-applications/ (Accessed: 08 December 2025).

Delport, P. M. J., von Solms, R. & Gerber, M. (2024) 'Methodological Guidelines for Design Science Research', Procedia Computer Science, 237, pp. 195–203. Available at: https://www.sciencedirect.com/science/article/pii/S1877050922000783 [Accessed 29 Nov. 2025].

Drechsler, A. & Hevner, A. (2022) 'Knowledge Paths in Design Science Research', Foundations and Trends® in Information Systems, 6(3), pp. 171–243. Available at: https://doi.org/10.1561/2900000028 [Accessed 29 Nov. 2025]. nowpublishers.com+1

Escaleira, P (2025) 'A systematic review on security mechanisms for serverless computing', Concurrency and Computation: Practice and Experience, [online]. Available at: https://link.springer.com/article/10.1007/s10586-025-05371-4 [Accessed 29 Nov. 2025]. link.springer.com

Gao, P., Xiao, X., Li, D., Li, Z., Jee, K., Wu, Z., Kim, C.H., Kulkarni, S.R. & Mittal, P. (2018) 'SAQL: A stream-based query system for real-time abnormal system behavior detection', arXiv, June. Available at: https://arxiv.org/abs/1806.09339 (Accessed: 08 December 2025).

GARG, M.Y., 2023. PERFORMANCE ANALYSIS AND BENCH MARKING OF THE MICRO VIRTUAL MACHINES BY PUBLIC CLOUD SERVICE PROVIDERS (Doctoral dissertation, Rajiv Gandhi Proudyogiki Vishwavidyalaya). https://www.researchgate.net/profile/Yugansh-Garg/publication/371255582_PERFORMANCE_ANALYSIS_AND_BENCH_MARKING_OF_THE_MICRO_VIRTUAL_MACHINES_BY_PUBLIC_CLOUD_SERVICE_PROVIDERS/links/647ac86ab3dfd73b775e938c/PERFORMANCE-ANALYSIS-AND-BENCH-MARKING-OF-THE-MICRO-VIRTUAL-MACHINES-BY-PUBLIC-CLOUD-SERVICE-PROVIDERS.pdf

Gopa, R., 2025. SERVERLESS DATA ENGINEERING WITH AWS LAMBDA AND STEP FUNCTIONS FOR REAL-TIME ANALYTICS. International Journal of Applied Mathematics, 38(7s), pp.989-1001. https://ijamjournal.org/ijam/publication/index.php/ijam/article/download/528/484

Gregor, S. (2024) 'Design science research and the co-creation of project value', International Journal of Project Management, [online]. Available at: https://www.sciencedirect.com/science/article/pii/S0263786324000267 [Accessed 29 Nov. 2025].

Herrera, V.A.S., de Araújo, H.P., Penteado, C.G., Gazziro, M. and Carmo, J.P., 2025. Low-Cost Embedded System Applications for Smart Cities. Big Data and Cognitive Computing, 9(2), p.19. https://doi.org/10.3390/bdcc9020019

Jason, J., 2025. Real-time Anomaly Detection in Social Media Analytics. https://www.researchgate.net/profile/Chidiebere-Joshua/publication/391459067_Real-time_Anomaly_Detection_in_Social_Media_Analytics/links/68193232bd3f1930dd6c973f/Real-time-Anomaly-Detection-in-Social-Media-Analytics.pdf

Jonathan, G. and Bola, Y., 2025. Using Amazon CloudWatch and X-Ray for Performance Optimization. https://www.researchgate.net/profile/Dorcas-Esther/publication/391018191_Using_Amazon_CloudWatch_and_X-Ray_for_Performance_Optimization/links/680837c3df0e3f544f456f69/Using-Amazon-CloudWatch-and-X-Ray-for-Performance-Optimization.pdf

Kai-Wähner, B. (2022) 'Fraud detection with Apache Kafka, KSQL and Apache Flink', Kai Wähner Blog, 25 October. Available at: https://www.kai-waehner.de/blog/2022/10/25/fraud-detection-with-apache-kafka-ksql-and-apache-flink/ (Accessed: 08 December 2025).

Kelly, D., Barrett, E. and Glavin, F., 2023. Denial of Wallet: Analysis of a looming threat and novel solution for mitigation using image classification (Doctoral dissertation, Ph. D. thesis, School Comput. Sci., College Sci. Eng., Univ. Galway, Ireland). https://researchrepository.universityofgalway.ie/bitstream/10379/17879/1/Denial_of_Wallet__Analysis_of_a_Looming_Threat_and_Novel_Solution_for_Mitigation_using_Image_Classification_Final.pdf

Malhotra, S., 2025. Next-Generation Observability Platforms: Redefining Debugging and Monitoring at Scale. Available at SSRN 5190462. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5190462

Mallick, M.A.I. and Nath, R., 2024. Securing the server-less frontier: Challenges and innovative solutions in network security for server-less computing. Reading Time, 193(1), pp.1-45. https://www.researchgate.net/profile/Md-Mallick/publication/379898050_Securing_the_Server-less_Frontier_Challenges_and_Innovative_Solutions_in_Network_Security_for_Server-_less_Computing/links/6717d0ee069cb92a812bf97e/Securing-the-Server-less-Frontier-Challenges-and-Innovative-Solutions-in-Network-Security-for-Server-less-Computing.pdf

Mampage, A., Karunasekera, S. and Buyya, R., 2022. A holistic view on resource management in serverless computing environments: Taxonomy and future directions. ACM Computing Surveys (CSUR), 54(11s), pp.1-36. https://doi.org/10.1145/3510412

Marklund, A., 2025. Observability in AWS Cloud Architectures: A Comparative Evaluation and Real-World Case Study. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1966811

Nguyen, C., Elmroth, E. & Bhuyan, M. (2025) 'Silent Failures in Stateless Systems: Rethinking Anomaly Detection for Serverless Computing', arXiv preprint, July. Available at: https://arxiv.org/abs/2507.04969 [Accessed 29 Nov. 2025].

*Nguyen, C., Elmroth, E. and Bhuyan, M., 2025, July. Silent Failures in Stateless Systems: Rethinking Anomaly Detection for Serverless Computing. In 2025 IEEE International Conference on Service-Oriented System Engineering (SOSE) (pp. 8-19). IEEE. **DOI:** 10.1109/SOSE67019.2025.00006*

Panda, S., 2025. Observability in DevOps: Integrating AWS X-Ray, CloudWatch, and Open Telemetry. International Journal of Computer Application. DOI: 10.5281/zenodo.15489185

Potluri, S., 2025. Multi-Layered Security Policy Enforcement for Confidential Data in Serverless Cloud Functions. International Journal of Emerging Trends in Computer Science and Information Technology, 6(1), pp.124-134. https://doi.org/10.63282/3050-9246.IJETCSIT-V6I1P114

Risco Gallardo, S., 2024. Serverless Strategies and Tools in the Cloud Computing Continuum (Doctoral dissertation, Universitat Politècnica de València). https://doi.org/10.4995/Thesis/10251/202013

Scribbr (2020) 'Reference a Website in Harvard Style', Scribbr, 19 May. Available at: https://www.scribbr.co.uk/referencing/harvard-website-reference/

Shafiei, H., Khonsari, A. and Mousavi, P., 2022. Serverless computing: a survey of opportunities, challenges, and applications. ACM Computing Surveys, 54(11s), pp.1-32. https://doi.org/10.1145/3510611

Shahid, U., Ahmed, G., Siddiqui, S., Shuja, J. and Balogun, A.O., 2024. Latency-sensitive function placement among heterogeneous nodes in serverless computing. Sensors, 24(13), p.4195. https://doi.org/10.3390/s24134195

Shin, W., Kim, W.H. and Min, C., 2022, March. Fireworks: A fast, efficient, and safe serverless framework using vm-level post-jit snapshot. In Proceedings of the Seventeenth European Conference on Computer Systems (pp. 663-677). https://doi.org/10.1145/3492321.3519581

Srivastava, S., Gupta, B.K., Tandon, D., Singh, A.K., Husain, M., Ali, A., Alshmrany, S., Gupta, S. and Dubey, S., 2023. Execution of Serverless Functions Lambda in AWS Serverless Environment. International Journal on Recent and Innovation Trends in Computing and Communication, 11(9), pp.1081-1086. https://www.researchgate.net/profile/Arshad-Ali-9/publication/376380366_Execution_of_Serverless_Functions_Lambda_in_AWS_Serverless_Environment/links/65756ffb6610947889b14c8f/Execution-of-Serverless-Functions-Lambda-in-AWS-Serverless-Environment.pdf

Sundaramurthy, S.K., Ravichandran, N., Inaganti, A.C. and Muppalaneni, R., 2025. AI-Driven Threat Detection: Leveraging Machine Learning for Real-Time Cybersecurity in Cloud Environments. Artificial Intelligence and Machine Learning Review, 6(1), pp.23-43. https://doi.org/10.69987/AIMLR.2025.60104

Syed, A.A.M. and Ali, S., 2024. Kubernetes and AWS Lambda for Serverless Computing: Optimizing Cost and Performance Using Kubernetes in a Hybrid Serverless Model. International Journal of Emerging Trends in Computer Science and Information Technology, 5(4), pp.50-60. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I4P106

Timmer, R., 2024. Cost-Effective Machine Learning Inference with AWS Lambda: Evaluating Serverless Resource Configurations (Doctoral dissertation). https://fse.studenttheses.ub.rug.nl/id/eprint/33792

Veltman, M., Parkegren, A. and Morel, V., 2023, November. Putting a Padlock on Lambda–Integrating vTPMs into AWS Firecracker. In 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 1377-1384). IEEE. https://doi.org/10.1109/TrustCom60117.2023.00188

Vervaet, A. (2023) 'MoniLog: An automated log-based anomaly detection system for cloud computing infrastructures', arXiv, April. Available at: https://arxiv.org/abs/2304.11940 (Accessed: 08 December 2025).

Weissman, Z., Tiemann, T., Eisenbarth, T. and Sunar, B., 2023. Microarchitectural security of AWS Firecracker VMM for serverless cloud platforms. arXiv preprint arXiv:2311.15999. https://doi.org/10.48550/arXiv.2311.15999

Wen, J., Chen, Z., Jin, X. and Liu, X., 2023. Rise of the planet of serverless computing: A systematic review. ACM Transactions on Software Engineering and Methodology, 32(5), pp.1-61. https://doi.org/10.1145/3579643