

Evaluating the Effectiveness of a Serverless AWS Log Forwarding Architecture for Threat Detection and Compliance Automation using Terraform.

Detailed Project Proposal

MSc Cyber Security

Module: Advanced Computer Science Masters Project(7COM1039-0509-2024)

Student Name: Shaistha Khanum

Student ID: 22076898

Academic Year: 2023-2025

University of Hertfordshire



Contents

[Introduction 2](#_Toc200453606)

[Project Goals 2](#_Toc200453607)

[Research Question 2](#_Toc200453608)

[Problem Statement 3](#_Toc200453609)

[Analysis Current Industry Gaps 3](#_Toc200453610)

[Project Scope 4](#_Toc200453611)

[Core Features 4](#_Toc200453612)

[Advanced Project Aims 4](#_Toc200453613)

[Resources Required 5](#_Toc200453614)

[Methodology 6](#_Toc200453615)

[Project Plan: 6](#_Toc200453616)

[Comparative Analysis 7](#_Toc200453617)

[Serverless v/s Server-based Architecture 7](#_Toc200453618)

[Manual Configurations v/s Terraform Automation 7](#_Toc200453619)

[Literature Review 8](#_Toc200453620)

[Serverless Architecture and Its Benefits 8](#_Toc200453621)

[Terraform (IAC) deployment in AWS, its key benefits, and challenges 8](#_Toc200453622)

[Monitoring and Log Forwarding in Serverless Architectures 9](#_Toc200453623)

[Risk Management 11](#_Toc200453624)

[Project Risks and their Mitigation 11](#_Toc200453625)

[Conclusion: 12](#_Toc200453626)

[Reference: 13](#_Toc200453627)

# Introduction

Organizations are depending more and more on cloud platforms like Amazon Web Services (AWS), Azure, and GCP to host vital workloads as a result of the quick and extensive adoption of cloud technology. Complex security issues are brought about by this digital transition, particularly in dynamic and scalable contexts where manual monitoring techniques are insufficient. Delays in threat detection, unpatched misconfigurations, and possible compliance violations can arise from manual operations via the AWS Management Console, which are frequently sluggish, inconsistent, and prone to errors.  
  
This study suggests using Terraform, an Infrastructure as Code (IaC) tool, to design, implement, and assess a serverless, automated AWS monitoring and log forwarding architecture. The architecture combines AWS-native services. like Lambda, CloudTrail, S3, CloudWatch, GuardDuty, AWS Config, and Security Hub to enable real-time detection, alerting, and remediation of security threats and compliance violations without the need of servers and manual interventions.  
  
Terraform enforces automation, consistency, and repeatability throughout the cloud architecture, while the system uses a serverless approach to ensure scalability, cost-effectiveness, and high availability. This project also uses technologies like CloudGoat to mimic attack situations in order to assess remediation effectiveness, alarm responsiveness, and detection accuracy. Important metrics including time-to-detect, time-to-remediate, and cost-performance trade-offs will be evaluated, along with a comparison of the serverless architecture deployed via Terraform and the server-based design deployed manually, in order to determine the architecture's impact.

## Project Goals

* Design a modular AWS architecture for log ingestion, aggregation, alerting, and storage using services like Lambda, S3, CloudWatch, and EC2.
* Automate infrastructure deployment using Terraform, including IAM, VPCs, subnets, ALB, and S3 lifecycle management policies.
* Integrate AWS native security services such as AWS Config, GuardDuty, and CloudTrail for compliance monitoring and threat detection.
* Generates alerts through SNS to notify the stakeholders
* Simulate misconfigurations and attacks using CloudGoat or similar tools to test detection and remediation.
* Measure performance metrics, including time-to-detect, time-to-remediate, alert accuracy, and cost.
* Deliver a working prototype with reusable Terraform modules and a comprehensive research report.

## Research Question

**RQ1**. How effective is a serverless AWS monitoring and log forwarding architecture, deployed via Terraform, in detecting, alerting, and remediating security misconfigurations and threats in real-time?

**RQ2.** How does a serverless AWS monitoring and log forwarding architecture deployed via Terraform compare to traditional and manually deployed setups in detecting, alerting, and remediating security misconfigurations and threats in real-time?

## Problem Statement

As cloud adoption grows across sectors, so do the challenges of securing dynamic and scalable AWS environments. Traditional **server-based architectures** and **manual configurations** often lead to delayed threat detection, inconsistent deployments, and a heightened risk of misconfigurations due to human error. These limitations hinder the establishment of a robust security posture and complicate compliance efforts in rapidly evolving cloud landscapes. Moreover, existing security solutions frequently lack full automation and seamless integration, particularly in scenarios requiring real-time threat detection, log monitoring and forwarding, and automated remediation.

This project designs and evaluates a fully automated, serverless AWS monitoring and log forwarding architecture using Terraform. It leverages AWS-native tools like Config, Security Hub, and GuardDuty for real-time threat detection, compliance monitoring, and automated remediation. Simulated attacks using CloudGoat will test system effectiveness. The research compares this IaC-based serverless model with traditional, manually configured server-based setups, assessing improvements in consistency, detection speed, operational efficiency, and security posture. The goal is to validate serverless and IaC as more secure and efficient strategies for modern cloud environments.

## Analysis Current Industry Gaps

While enterprise security tools exist, many cloud-native organizations still lack unified and fully automated solutions for real-time threat detection, alerting, and remediation. Traditional approaches often depend heavily on manual console interactions, non-scalable agent-based deployments, and fragmented security tooling. Additionally, many organizations continue to use traditional server-based architectures, such as EC2 instances, which require constant patching, configuration, and monitoring. This not only increases operational overhead but also expands the attack surface, making systems more vulnerable to exploitation. These limitations hinder the ability to maintain a strong security posture in fast-evolving cloud environments.

This project aims to bridge these gaps by implementing:

* **Terraform-based Infrastructure as Code (IaC)** to enforce consistency, repeatability, and automation.
* **Serverless services** (Lambda, S3, CloudWatch) to enable real-time, scalable, and resilient log processing.
* **Integrated threat detection and compliance auditing** using AWS-native tools like GuardDuty, AWS Config, and Security Hub.
* **Automated remediation workflows** to promptly address misconfigurations and suspicious activity, enhancing response efficiency.
* Compares this IaC-based serverless model with traditional server-based architectures configured manually via the AWS console.
* Through this comparative analysis, the research aims to validate the effectiveness of **serverless architectures** and **IaC automation** as superior strategies for securing modern cloud infrastructures.

# Project Scope

## Core Features

1. **Serverless Log Forwarding Pipeline**
   * Forward logs from S3 to Lambda for processing and analysis.
   * Send findings to Security Hub for centralized visibility.
2. **Threat Detection and Alerting**
   * Enable AWS GuardDuty, CloudTrail, and Config to detect and report security events.
   * Generate SNS alerts for findings above a severity threshold.
3. **Automated Remediation**
   * Use AWS Config Rules with remediation actions (e.g., revoke public access, stop exposed EC2).
   * Trigger Lambda functions to auto-fix or isolate misconfigured resources.
4. **Terraform Automation**
   * Deploy complete architecture via Terraform.
   * Modular, reusable IaC components for VPCs, IAM, logging, security controls.
5. **Security Dashboards and Reports**
   * Enable Security Hub to collect findings from GuardDuty, Config, and third-party tools.
   * Evaluate real-time compliance status and risk posture.

## Advanced Project Aims

**Simulate Attacks Using CloudGoat**

* Test architecture resilience using CloudGoat scenarios (e.g., exposed IAM roles, over-permissioned users).

**Evaluate Detection & Remediation Timings**

* Measure time-to-detect, time-to-alert, time-to-remediate.

**Cost and Performance Optimization**

* Analyze trade-offs of serverless vs. agent-based approaches in cost, speed, and accuracy.

**AI-Assisted Remediation**

* Integrate AWS CodeWhisperer to analyze Terraform templates for misconfigurations and apply auto-corrections.

## Resources Required

**Cloud Resources**

* AWS Free Tier
* Test EC2, Lambda, S3, GuardDuty, and Config setups

**Software**

* Terraform, AWS CLI, VS Code, GitHub
* Checkov, Prowler (open-source CSPM tools)
* Python scripts for analysis and remediation triggers

**Security Tools**

* AWS Security Hub, GuardDuty, AWS Config Rules
* CloudTrail logging and alert integration

# Methodology

This project will follow a structured research methodology involving the following phases:

1. Literature Review: Conduct secondary research on cloud monitoring, Terraform, serverless s v/s server-based architecture attack surface and security, AWS services (CloudTrail, GuardDuty, Config), and related architectures.
2. Design: Define and document the architecture, security workflows, and automation strategies using Terraform.
3. Implementation: Deploy Serverless and server-based AWS infrastructure using Terraform, including EC2, ALB, Lambda, CloudTrail, GuardDuty, AWS Config, S3, IAM, and CloudWatch for comparative analysis.
4. Testing: Simulate attack scenarios using CloudGoat to evaluate system performance, detection, and remediation.
5. Evaluation: Compare metrics like detection time, remediation time, alert accuracy, and cost against benchmarks or published literature.
6. Optimization: Refine infrastructure based on feedback and results, focusing on scalability, detection accuracy, and cost efficiency.
7. Documentation: Report findings, evaluate the impact, and provide recommendations for industry adoption.

## Project Plan:

|  |  |
| --- | --- |
| **Week** | **Activity** |
| 1-2 | Literature review, AWS environment setup |
| 3-4 | Design Terraform IAC architecture |
| 5-7 | Develop and test monitoring and log forwarding |
| 8-9 | Simulate attacks using CloudGoat and also look for other similar type of tools and what kind of attack surface it focused. |
| 10 | Evaluate detection and remediation performance |
| 11 | Write research paper |
| 12 | Final submission and presentation preparation |

## Comparative Analysis

### Serverless v/s Server-based Architecture

The Analysis of Serverless v/s Server -based Architecture will be focused on Threat detection speed, attack surface, scalability of logs, remediations, misconfiguration detections, operation overhead, and security maintenance.

### Manual Configurations v/s Terraform Automation

The configuration of cloud infrastructure, particularly within AWS environments, has a direct impact on security, compliance, and operational reliability. Traditionally, manual configurations through the AWS Management Console or CLI have been the go-to approach for many organizations. However, this method is highly susceptible to human error and leads to inconsistent deployments across environments. Inconsistent resource configurations, such as differing IAM roles or security group rules across development, staging, and production environments, are common. This lack of standardization not only increases operational risk but also makes auditing and compliance more difficult.

In contrast, Terraform automation offers a declarative and consistent approach to infrastructure deployment using Infrastructure as Code (IaC). It allows teams to define and manage infrastructure in reusable, version-controlled code, eliminating manual errors and promoting standardization across all environments. Terraform also supports automated security scanning through integrations with tools like Checkov and TFLint, helping identify misconfigurations before deployment.

Literature Review

## Serverless Architecture and Its Benefits

Serverless computing is a key enabler of **scalable, agile, and cost-effective** cloud infrastructure, particularly suited for dynamic workloads such as real-time monitoring and threat detection. **Bhardwaj, Mishra, and Mehta (2024)** provide a detailed exploration of how serverless computing enhances cloud security by reducing the attack surface and simplifying operational complexity. Their study highlights major benefits including automatic scalability, pay-as-you-go cost efficiency, and the abstraction of infrastructure management—features that directly support this project's goal of building a lightweight, secure, and event-driven log forwarding architecture using AWS Lambda.

The paper also outlines challenges such as cold start latency and vendor lock-in, which are acknowledged in this project and mitigated by optimizing Lambda configurations and using Infrastructure as Code (IaC) with Terraform for portability and reproducibility. The authors’ recommendations for best practices in secure serverless deployment validate the use of AWS-native services such as **Config, CloudTrail, and Security Hub** to enforce compliance and automate security checks.

Similarly, **Ortiz (2019)** emphasizes the advantages of serverless microservices through **AWS Lambda** and **DynamoDB** in educational environments. The paper presents a modular approach to application design, showcasing how event-driven, serverless architectures enhance maintainability and responsiveness. This aligns closely with the current project’s use of **Lambda to trigger log processing functions based on S3 and CloudWatch events**. Ortiz’s focus on practical, hands-on implementation further supports the project’s Terraform-based automation and Lambda-driven alerting model.

While both studies focus on foundational design and benefits, this research project extends their findings by integrating automated security remediation and compliance enforcement using Terraform, AWS Config, and GuardDuty. This fusion of serverless design with real-time threat detection represents a novel contribution to cloud-native security architectures.

## Terraform (IAC) deployment in AWS, its key benefits, and challenges

Infrastructure as Code (IaC) has transformed the way cloud infrastructure is provisioned and managed. Among the various tools available, **Terraform** has gained wide adoption due to its ability to automate deployments, enforce configuration consistency, and manage infrastructure across multiple providers. In the context of AWS, Terraform plays a pivotal role in deploying scalable, secure, and repeatable environments.

**Vignesh and Kanna (2020)** underscore the need for consistent configuration across cloud infrastructure to enhance both operational efficiency and security. Their research points out that manual provisioning of AWS resources can result in misconfigurations and deployment delays, which expose cloud systems to security threats such as IP-based attacks and unauthorized access. To mitigate this, they advocate using DevOps automation tools like **Terraform, Jenkins, and Chef** to convert static OVA templates into secure Amazon Machine Images (AMIs). This method significantly reduces human error and accelerates secure deployment workflows.

Although their study focuses on traditional EC2-based setups, the implications align closely with this project, which transitions the same automation and security principles into a **serverless AWS environment**. By using Terraform to deploy Lambda-based log monitoring pipelines and AWS-native threat detection tools like GuardDuty and AWS Config, this research extends the benefits of automation into modern, scalable architectures. Both projects emphasize the necessity of reducing manual errors and integrating automated compliance mechanisms to tackle evolving cloud threats.

Expanding on this, **Mehdi and Walia (2023)** provide a broader perspective on Terraform’s role in cloud infrastructure management. Their study highlights how Terraform reduces deployment time and operational complexity by replacing manual configurations with code-based templates. They also note Terraform’s support for **multi-cloud provisioning**, which, while beyond the current AWS-only scope, illustrates Terraform’s extensibility for hybrid cloud environments. Importantly, the authors discuss the emergence of **"click-on-go" infrastructure deployment**, where reusable Terraform modules allow infrastructure to be spun up quickly and consistently—this is particularly relevant for this project, which uses Terraform to orchestrate CloudWatch log groups, Lambda functions, S3 buckets, and AWS security services.

Despite its strengths, Mehdi and Walia also caution that organizations often fail to realize the full benefits of IaC tools like Terraform due to poor implementation and governance. This insight reinforces the importance of incorporating **monitoring, compliance enforcement, and remediation workflows** into any Terraform-based deployment. The present research addresses this by integrating Terraform with AWS Config rules and SNS alerts to not only detect misconfigurations but also initiate remediation steps, ensuring that security and compliance objectives are actively maintained.

In conclusion, both studies affirm Terraform’s effectiveness in automating infrastructure deployment and reducing security risks introduced by manual configuration. While Vignesh and Kanna (2020) focus on infrastructure provisioning and image hardening, and Mehdi and Walia (2023) explore Terraform’s broader cloud automation capabilities, both provide a foundational basis for this study. This research builds on their findings by applying Terraform in a serverless context with a focus on **log forwarding, threat detection, and compliance automation**, thus contributing to the expanding literature on secure, automated cloud operations using IaC.

## Monitoring and Log Forwarding in Serverless Architectures

Serverless computing has revolutionized cloud application design by abstracting away infrastructure management, allowing developers to focus purely on function logic. This architectural shift has also influenced how monitoring and logging are conducted in the cloud. Traditional host-based security and log collection approaches are insufficient in these ephemeral environments, necessitating the adoption of new paradigms for observability and threat detection.

To illustrate the evolution of serverless data handling, **Remala et al. (2024)** introduced a serverless data ingestion framework on AWS, leveraging services like **AWS Lambda, Amazon S3, AWS Glue, and API Gateway**. Their framework achieved enhanced scalability, cost efficiency, and reduced operational complexity compared to traditional monolithic or microservices architectures. Although their work focused on streamlining **data ingestion and transformation**, it offers a foundational blueprint for designing robust and scalable pipelines in serverless environments. The current research builds upon this framework by extending its use case from data processing to **security-focused log forwarding and monitoring**. By incorporating tools like **AWS Config, GuardDuty**, and **Terraform-based auto-remediation**, this project bridges the gap between serverless efficiency and **cloud security posture management (CSPM)**. Thus, the study by Remala et al. serves as an architectural precursor, validating the feasibility and scalability of serverless workflows, which this research strengthens with additional security automation layers.

Further advancing the need for visibility in serverless systems, **Mamiyev and Lazányi (2023)** propose a **cloud-native anomaly detection framework** that utilizes operational metrics—specifically **Lambda invocation frequency, execution time, and error rates**—to detect threats such as brute-force attacks, injection attempts, or API abuse. Their work highlights the inadequacy of static, rule-based models in serverless environments and underscores the need for **adaptive, metric-driven monitoring**. For instance, a sudden spike in Lambda invocations may indicate a denial-of-service attempt, while elevated error rates could signify unauthorized access or faulty input. Prolonged execution durations might flag resource exhaustion or latent performance vulnerabilities. The authors argue that effective security in serverless systems depends on **real-time metric analysis and log forwarding** via services like **AWS CloudWatch**, **CloudTrail**, and **centralized S3 buckets**.

This insight directly informs the technical backbone of the current project. By designing a **Terraform-based log forwarding and monitoring pipeline**, this research integrates **CloudWatch, CloudTrail, Lambda, S3, GuardDuty, AWS Config**, and **Security Hub** to create a secure and automated observability stack. The project echoes the paper’s recommendation for forwarding logs and metrics to a centralized processing layer—where **Lambda functions analyze patterns**, **SNS alerts are triggered**, and **AWS Config handles automated remediation**. As such, Mamiyev and Lazányi’s framework justifies the selection and integration of specific AWS services within this project and reinforces the notion that **monitoring in serverless architectures is not just an operational necessity, but a critical security control.**

Together, these two studies provide a compelling case for the evolution and importance of monitoring in serverless systems. While Remala et al. underscore the architectural viability of serverless pipelines, Mamiyev and Lazányi validate the strategic role of log forwarding and anomaly detection for securing these environments—both of which directly shape the design and goals of the current research.

# Risk Management

## Project Risks and their Mitigation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Risk Description | Probability | Possible Effects | Mitigation Methods |
| 1 | Terraform Misconfiguration | Medium | Deployment failure or security risk | Use CodeWhisperer and Checkov to scan IaC |
| 2 | Overuse of AWS Resources | High | Cost overruns | Set resource quotas, monitor usage |
| 3 | Integration Failures | Medium | Broken pipeline | Use modular Terraform with testing |
| 4 | Tool Incompatibility | Medium | Loss of data or incomplete detection | Validate tool versions and test isolated |
|  | Incomplete Attack Simulation | Low | Biased evaluation | Use multiple CloudGoat scenarios |

# Conclusion:

The design, implementation, and efficacy of a serverless AWS monitoring and log forwarding architecture set up with Terraform for security automation and compliance enforcement are all rigorously assessed in this project. For scalable, automated, and reasonably priced security frameworks, a thorough analysis of AWS native services, including **AWS Lambda**, **CloudWatch**, **S3**, **CloudTrail**, **GuardDuty**, **AWS Config**, **and Security Hub** is conducted.

The feasibility of Infrastructure as Code (IaC) and serverless computing for proactive cloud protection is empirically supported by this study, which simulates real-world threats using tools like **CloudGoat** and analyses detection and remediation times. Additionally, it shows significant gains in detection speed, operational agility, automation consistency, and security posture when comparing this architecture to more conventional server-based solutions.

Ultimately, the research validates that **serverless, IaC-driven security architectures are not only feasible but highly effective** for modern enterprises seeking to reduce manual overhead, ensure real-time visibility, and maintain continuous compliance in AWS environments. The findings and prototype contribute meaningful insights for both academia and industry, offering a repeatable and practical model for securing dynamic cloud infrastructures.

# Reference:

Amazon Web Services (AWS), n.d**.** *AWS Lambda Functions*. [online] Available at: <https://docs.aws.amazon.com/lambda/latest/dg/lambda-functions.html> [Accessed 9 June 2025].

Bhardwaj, L., Mishra, N. & Mehta, A., 2024. *Elevating Cloud Security via Serverless Computing: An in Depth Exploration*. International Journal of Advanced Research in Science, Communication and Technology. Available at: <https://doi.org/10.48175/ijarsct-17618>.

Bysani, V. (2024) ‘Automation in Cloud Infrastructure Management: Enhancing Efficiency and Reliability’, *Indian Scientific Journal Of Research In Engineering And Management*, 08(06), pp. 1–5. doi: 10.55041/ijsrem35750.

HashiCorp, 2025. *AWS Provider — Terraform Registry*. [online] Available at: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs> [Accessed 9 June 2025].

Mamiyev, A. and Lazányi, K., Machine Learning-Driven Anomaly Detection in Serverless Computing to Enhance Cloud Security. In *XIX. FIKUSZ 2024 International Conference* (p. 302).

Ortiz, A., 2019. *Architecting Serverless Microservices on the Cloud with AWS*. Technical Symposium on Computer Science Education, p.1240. Available at: <https://doi.org/10.1145/3287324.3287533>

Remala, R., Mudunuru, K.R. and Nagarajan, S.K.S., Optimizing Data Ingestion Processes using a Serverless Framework on Amazon Web Services. *International Journal of Computer Trends and Technology*, *72*.

Vignesh, S., & Kanna, B. R. (2020). AWS Infrastructure Automation and Security Prevention Using DevOps (pp. 537–549). Springer, Singapore. https://doi.org/10.1007/978-981-15-0199-9\_46

Top of Form

Bottom of Form