Project Report

Cloud Threat Detection & Response Engine

Submitted in the partial fulfilment of the requirements for

the degree of

Master of technology

by

Darshan Pardeshi

Roll No: 16031024005

Guide

Prof. Yogita Borse
Prof. Purnima Ahirao

Department of Information Technology (specialising in Information Security)

K. J. Somaiya College of Engineering

(Constituent College of Somaiya Vidyavihar University)

Batch 2024 -2026

Somaiya Vidyavihar University

K. J. Somaiya College of Engineering

(Constituent College of Somaiya Vidyavihar University)

Certificate

This is to certify that the Project Report on "Cloud Threat Detection & Response Engine" is a bona fide record of the project work done by Darshan Pardeshi during the academic years 2024–2026, under the guidance of Prof. Yogita Borse, Department of Information Technology, in partial fulfilment of the requirements for the Master of Technology (M.Tech.) degree in Information Technology (Specialisation in Information Security) of Somaiya Vidyavihar University.

Yogita Borse
Guide

Purnima Ahirao
Co-Guide

Principal/Director/Dean

Date: 28/07/2024

Place: Mumbai-77

Somaiya Vidyavihar University

K. J. Somaiya College of Engineering

(Constituent College of Somaiya Vidyavihar University)

Certificate of Approval of Examiners

We certify that this Project Report on *" Cloud Threat Detection & Response Engine "* is a bona fide record of the project work done by Darshan Pardeshi for the partial fulfilment of the requirements for the Master of Technology (M.Tech.) degree in Information Technology (Specialisation in Information Security) of Somaiya Vidyavihar University.

Yogita Borse — Guide

Purnima Ahirao — Co-Guide

Date: 28/07/2024

Place: Mumbai-77

Somaiya Vidyavihar University

K. J. Somaiya College of Engineering

(Constituent College of Somaiya Vidyavihar University)

Declaration

I hereby declare that this Project Report is a genuine and original work carried out by me. The content presented in this report is based on my own work and/or the ideas of others, which have been properly cited and referenced wherever applicable.

I affirm that I have upheld the principles of academic honesty and integrity. I have not misrepresented, fabricated, or falsified any idea, data, fact, source, or original work in this submission.

I understand that any violation of the above may lead to disciplinary action by the college and could also invite legal or academic consequences from external sources if proper citations or permissions have not been provided.

_____
Signature of the student

_____
Name of the student

_____
Roll No.

Date : 28/07/2024

Place: Mumbai-77

ABSTRACT

In today's cloud-driven world, security threats such as unauthorized access, data leaks, and privilege misuse are becoming increasingly complex and difficult to detect manually. This project, titled "Cloud Threat Detection & Response Engine," presents a comprehensive, fully automated solution built entirely on native AWS services to simulate, detect, analyze, and respond to real-world cloud security threats.

The system was designed to identify critical threat scenarios, including publicly exposed S3 buckets containing sensitive data and unauthorized port scanning against EC2 instances. Using intelligent detection services such as Amazon GuardDuty and Amazon Macie, the engine successfully flagged misconfigurations and data exposure events in near real-time. All API activity and network behavior were logged using AWS CloudTrail and VPC Flow Logs, and further analyzed using Amazon Athena to uncover reconnaissance behavior and access anomalies.

To support compliance, change tracking, and forensic readiness, the solution integrated AWS Config and Amazon Detective, enabling deep visibility into resource configuration histories, IAM actions, and relationships between entities involved in the threats. Amazon CloudWatch was used to monitor Lambda executions, view logs from event responses, and ensure operational observability across the detection pipeline.

A serverless response layer was built using Amazon EventBridge, AWS Lambda, and Amazon SNS, enabling automated alerting and response actions. When threats were detected, the pipeline automatically triggered Lambda functions and sent email alerts to designated security personnel ensuring rapid response with zero manual intervention.

This project demonstrates how AWS-native security, monitoring, and automation services can be orchestrated into a unified, scalable, and cost-effective cloud threat detection and response system. By leveraging integrated AWS services, the solution replicates a real-world SOC (Security Operations Center) environment, capable of enforcing least-privilege principles, ensuring continuous compliance, and responding dynamically to evolving cloud threats — all without the need for third-party tools or agents.

Keywords :- Cloud Security, AWS Security, Threat Detection, Incident Response, GuardDuty, Amazon Macie, AWS Config, Amazon Detective, CloudTrail, VPC Flow Logs, Amazon Athena, Amazon CloudWatch, EventBridge, AWS Lambda, Amazon SNS, S3 Security, EC2 Port Scanning, IAM Policy Monitoring, Serverless Automation, Cloud Compliance, Zero Trust Architecture, SOC Automation, Cloud Forensics, Security Monitoring

Contents :

List Of Figures

5.1	CloudTrail Tracking of Configuration Changes and Login Attempts	
5.2	CloudTrail Logging of IAM Activity and Console Login Events	
5.3	AWS Config Findings: Detection of Security Drift and Configuration Changes	
5.4	GuardDuty Finding: Public S3 Bucket Access Detected via Anonymous Access Policy	
5.5	GuardDuty Finding: S3 Public Access Granted via IAM Policy Misconfiguration	
5.6	Amazon Macie Findings: Detection of Sensitive Data Exposure in Public S3 Bucket	
5.7	SNS Alert: GuardDuty Detection of Block Public Access Disabled	
5.8	SNS Alert: GuardDuty Detection of S3 Bucket Anonymous Access	
5.9	CloudWatch Insights Dashboard: Detection of Port Scan Attempts via VPC Flow Logs	
5.10	Amazon Detective Visualization: High-Volume IAM API Activity for Threat Attribution	
5.11	Amazon Detective Graph: EC2 Instances Ranked by Network Traffic Volume	

List of Abbreviations

Abbreviation	Full Form
AWS	Amazon Web Services
SOC	Security Operations Center
SIEM	Security Information and Event Management
VPC	Virtual Private Cloud
SSN	Simple Notification Service
S3	Simple storage service
JSON	JavaScript Object Notation
EC2	Elastic Compute Cloud
GUI	Graphical User Interface
CLI	Command Line Interface
Lambda	AWS Lambda (Serverless Compute Service)
CloudTrail	Aws Cloudtrail(Logging & Governance)
GuardDuty	AWS GuardDuty (Threat Detection)
Athena	Amazon Athena (Serverless Query Engine)
IAM	Identity and Access Management
Detective	AWS Detective (Threat Investigation Tool)
Cloudwatch	Monitoring and Logging Service
EventBridge	Amazon EventBridge (Event-Driven Orchestration)

Chapter 1

Introduction

1.1 Overview of Cloud Security Challenges

As organizations shift critical workloads to the cloud, they face an evolving set of security challenges that traditional on-premises models are ill-equipped to handle. The dynamic and distributed nature of cloud environments introduces risks such as misconfigured storage buckets, over-permissive IAM roles, and unmonitored API activity, which can lead to data exposure or unauthorized access. In the absence of proper visibility and control, threats like port scanning, brute-force login attempts, and privilege escalation often go undetected. Cloud providers operate on a shared responsibility model, which places significant security responsibilities like identity management, data protection, and compliance enforcement on the user. Additionally, cloud services generate massive volumes of logs (from CloudTrail, VPC Flow Logs, IAM), making real-time detection and analysis complex without automation. Failure to monitor these signals effectively can delay response, escalate risk, and lead to compliance violations. These challenges underscore the need for a solution that combines real-time threat detection, behavior analysis, and automated response precisely what this project aims to deliver using native AWS services.

1.2 Need for Threat Detection & Response in Cloud

As organizations increasingly migrate to cloud platforms, the attack surface expands and traditional security models fail to keep up. Cloud environments are dynamic resources spin up and down in seconds, identities are distributed, and threats evolve rapidly. This complexity demands more than manual monitoring or static defenses. Cloud threats such as port scanning, brute-force login attempts, misconfigurations, and privilege escalations can occur within minutes and often go unnoticed without real-time detection. Delayed responses can lead to data breaches, compliance failures, and financial loss. To address this, cloud-native environments need automated, scalable, and intelligent security systems that can detect threats in real time and initiate immediate response actions. Integrating services like AWS GuardDuty, CloudTrail, Lambda, and EventBridge enables organizations to stay ahead of attackers with minimal manual effort. This project fulfills that need by building a fully automated Cloud Threat Detection & Response Engine on AWS designed to protect cloud workloads through continuous monitoring, fast mitigation, and centralized threat visibility.

1.3 Problem Statement

As cloud adoption accelerates across industries, so does the complexity of securing cloud-native environments. Traditional security tools and practices often fall short in addressing modern cloud threats, which are fast-evolving, automated, and capable of exploiting misconfigurations, excessive permissions, and identity-based vulnerabilities. Manual detection methods are reactive, slow, and resource-intensive making them ineffective in identifying real-time attacks such as port scanning, brute-force login attempts,

and privilege escalations. Additionally, the fragmented nature of cloud logs and the lack of centralized, actionable insights often lead to delayed responses and missed threats.
The core problem lies in the absence of an integrated, automated, and intelligent system that can continuously monitor cloud environments, detect anomalous activity, and trigger immediate response actions without requiring manual intervention. This project aims to bridge that gap by developing a Cloud Threat Detection & Response Engine using AWS-native services that offer real-time visibility, serverless automation, and centralized threat intelligence enabling organizations to proactively defend against cloud-based attacks.

1.4 Objectives of the Project

The core objective of this project is to develop an automated, scalable, and serverless engine on AWS that can detect and respond to cloud security threats in real time.

Specific objectives include:

(1) To simulate real-world cloud security threats such as public S3 bucket exposure and EC2 port scanning in a controlled environment.
(2) To implement continuous monitoring and detection of threats using AWS-native services like Amazon GuardDuty, Macie, and AWS Config.
(3) To collect and analyze API activity and network traffic logs through AWS CloudTrail, VPC Flow Logs, and Amazon Athena.
(4) To build an automated alerting and response pipeline using EventBridge, Lambda, and SNS for real-time notifications and minimal manual intervention.
(5) To investigate findings and trace threat origins using Amazon Detective for contextual analysis and forensic insights.
(6) To evaluate the accuracy, efficiency, and response time of the threat detection pipeline under simulated attack scenarios.
(7) To demonstrate how serverless and integrated AWS services can achieve Zero Trust enforcement, least-privilege governance, and real-time threat mitigation without relying on third-party tools.

1.5 Scope of the Project

(1) To develop a real-time threat detection and response engine leveraging only AWS-native services, ensuring cost-efficiency and platform-native integration.
(2) To simulate multiple real-world cloud attack scenarios, including public S3 bucket exposure, IAM policy misconfiguration, and EC2 port scanning.
(3) To enable comprehensive log collection and analysis using services like AWS CloudTrail, VPC Flow Logs, and Amazon Athena.
(4) To detect threats using services such as Amazon GuardDuty, Amazon Macie, and AWS Config, and investigate findings using Amazon Detective.
(5) To automate threat response workflows using Amazon EventBridge, AWS Lambda, and Amazon SNS, reducing reliance on manual security operations.
(6) To implement monitoring and operational visibility through Amazon CloudWatch for serverless components and detection workflows.
(7) To evaluate system performance in terms of detection accuracy, response time,

and cost within AWS Free Tier limits.
(8) To provide a scalable and extensible architecture that can be enhanced for future use cases, including integration with SIEM tools, machine learning–based threat scoring, and multi-cloud support.

1.6 Methodology Overview

This project follows a structured and modular methodology to design, implement, and evaluate a cloud-native threat detection and response system using AWS. The key phases of the methodology are outlined below:

(1) Requirement Analysis:-
Identified key cloud security risks such as S3 misconfigurations, IAM privilege misuse, and EC2 exposure to external threats.
(2) Attack Simulation Planning:-
Defined and designed real-world cloud attack scenarios (e.g., public bucket exposure, port scanning) for testing detection capabilities.
(3) Environment Setup:-
Configured the AWS environment including IAM roles, VPC, EC2 instances, S3 buckets, and permissions for controlled simulation and monitoring.
(4) Log Collection and Monitoring:-
Enabled services like AWS CloudTrail and VPC Flow Logs to capture API-level activity and network traffic for visibility and analysis.
(5) Threat Detection Configuration:
Integrated Amazon GuardDuty, Macie, and AWS Config to detect anomalies, misconfigurations, and sensitive data exposure in real time.
(6) Automated Response Workflow:-
Implemented EventBridge, Lambda, and SNS to automate alerting and initiate response actions without manual intervention.
(7) Data Analysis and Investigation:-
Queried logs using Amazon Athena, and used Amazon Detective to analyze attacker behavior, entity relationships, and threat origins.
(8) Performance Evaluation:-
Measured system effectiveness based on detection accuracy, response time, and cost efficiency within AWS Free Tier constraints.
(9) Documentation and Reporting:-
Recorded each step, finding, and outcome to generate a comprehensive project report with architectural diagrams, logs, and screenshots.

This structured methodology ensures efficiency, scalability, and real-time protection, making the solution suitable for production-level deployment in modern cloud environments.

1.7 Report Organization

This report is divided into seven chapters, each presenting a structured view of the design, development, and evaluation of the project Cloud Threat Detection & Response Engine.

- Chapter 1 – Introduction
  Introduces cloud security challenges, the need for cloud threat detection and response, the problem statement, project objectives, scope, methodology, and an overview of the report structure.
- Chapter 2 – Literature Review
  Reviews existing cloud security solutions, compares them with traditional systems, discusses limitations in current industry practices, and identifies the gaps this project aims to fill.
- Chapter 3 – System Architecture and Design
  Describes the proposed architecture, data flow, and system components organized into three core layers: data collection, threat detection, and response automation. It also explains the AWS services used, their justification, and analyzes security and cost-performance factors.
- Chapter 4 – Implementation
  Covers the environment setup, detailed configuration of AWS services, development of the serverless automation pipeline, log analysis using Athena and OpenSearch, and the simulation of attack scenarios such as port scanning, brute-force, and privilege escalation.
- Chapter 5 – Results and Analysis
  Presents evaluation metrics including detection accuracy, response times, sample attack outcomes, visualization dashboards, AWS free tier cost estimates, and how well the system meets the defined project objectives.
- Chapter 6 – Challenges and Limitations
  Highlights technical and practical challenges faced during the project, such as service integration issues, data volume and query latency, AWS region/service limitations, and constraints of working within the free-tier environment.
- Chapter 7 – Conclusion and Future Work
  Summarizes achievements and technical learnings of the project. It also outlines possible future enhancements, SIEM tool integration, support other cloud platforms like Azure and GCP, and custom dashboard development for real-time monitoring.

Chapter 2

Literature Review

2.1 Existing Solutions for Cloud Threat Detection

Cloud security vendors and cloud providers offer various solutions to detect and respond to threats. AWS, Azure, and GCP provide native services such as Amazon GuardDuty, AWS Macie, and AWS Config to identify misconfigurations, sensitive data exposure, and anomalous behavior. Similarly, Microsoft Defender for Cloud and Google Security Command Center offer threat visibility within their ecosystems. Third-party platforms like Prisma Cloud, CrowdStrike, Wiz, and Datadog extend multi-cloud protection with advanced scanning, vulnerability detection, and threat correlation. SIEM tools like Splunk, Microsoft Sentinel, and QRadar also integrate with cloud logs to provide rule-based detection and incident response capabilities. While these tools are powerful, they often involve high cost, integration complexity, or dependency on agents. This project addresses those gaps by using serverless, native AWS services to build an efficient, automated, and cost-effective cloud threat detection and response engine.

2.2 Comparison with Traditional Security Systems

Traditional security systems were built for static, perimeter-based infrastructures, where assets were predictable and tightly controlled. However, the cloud dissolves that perimeter introducing dynamic resources, ephemeral workloads, and decentralized access which legacy tools struggle to secure effectively. In contrast, modern cloud-native environments demand context-aware, scalable, and automated security solutions that integrate directly with cloud APIs and services.

Key Differences :

Feature	Traditional Security Systems	Cloud-Native Detection
Architecture	Static, on-premises	Elastic, dynamic, API-driven
Scalability	Limited, manual	Auto-scaling, serverless
Detection Speed	Delayed (log review/manual)	Real-time (GuardDuty, CloudWatch, Lambda)
Automation	Minimal	Fully automated detection and response

Maintenance	Requires dedicated team	Minimal (managed services)
Cost Efficiency	High fixed costs	Pay-as-you-go
Threat Intelligence Integration	Manual updates	Built-in (GuardDuty, Security Hub, etc.)
Response Mechanism	Manual incident response	Automated mitigation via Lambda

2.3 Limitations in Current Industry Practices

While cloud adoption is soaring, many organizations still rely on outdated or fragmented security practices that struggle to match the speed and complexity of cloud-native threats. Key limitations include:

1) Reactive Detection: Most systems detect threats after damage is done due to lack of real-time monitoring and automation.
2) Siloed Security Tools: Fragmented solutions lead to poor visibility and coordination across cloud services.
3) Static Rule-Based Systems: Traditional methods miss zero-day and behavioral-based threats common in dynamic cloud setups.
4) Limited Context Awareness: Alerts often lack depth generating false positives and delaying response.
5) Operational Overhead: Many solutions are expensive, complex to manage, and hard to scale across multi-cloud environments.

These shortcomings highlight the urgent need for intelligent, automated, and cloud-native security solutions precisely what this project delivers using tightly integrated AWS services**.**

2.4 Summary of Gaps Identified

Based on the review of existing solutions and current industry practices, several critical gaps have been identified that highlight the need for a more effective cloud threat detection and response approach:

- Lack of Real-Time, Automated Response:
  Most solutions can detect threats but rely on manual workflows for mitigation, increasing the response time and impact.
- Insufficient Integration Across Services:
  Many tools operate in isolation, making it difficult to correlate events across logs, users, and cloud services for accurate threat analysis.
- Limited Context and Intelligence:
  Static rule-based alerts often lack behavioral insight and contextual awareness, leading to high false positives or missed detections.
- High Complexity and Cost:-

Advanced commercial platforms offer robust features but are too complex or expensive for small to mid-sized organizations to adopt effectively.

- Poor Customization and Scalability:-
  Existing systems may not adapt well to evolving workloads or allow custom threat response flows tailored to specific cloud use cases.

Chapter 3

System Architecture and Design

3.1 Proposed System Architecture

The proposed architecture presents a comprehensive, fully serverless security pipeline leveraging native AWS services to simulate, detect, analyze, and respond to cloud-based threats in real time. The system is designed to address common cloud vulnerabilities such as unauthorized port access on EC2 instances, IAM privilege misconfigurations, and publicly exposed S3 buckets containing sensitive information. Data collection is handled through AWS CloudTrail for API activity logging and VPC Flow Logs for capturing network-level traffic. These logs are stored in Amazon S3 and CloudWatch Log Groups, serving as the foundational data layer for threat detection and analysis.

Threat detection is orchestrated through a triad of services: Amazon GuardDuty for identifying reconnaissance and anomalous behavior, Amazon Macie for scanning S3 buckets for sensitive data,AWS Config monitoring compliance and resource configuration changes.

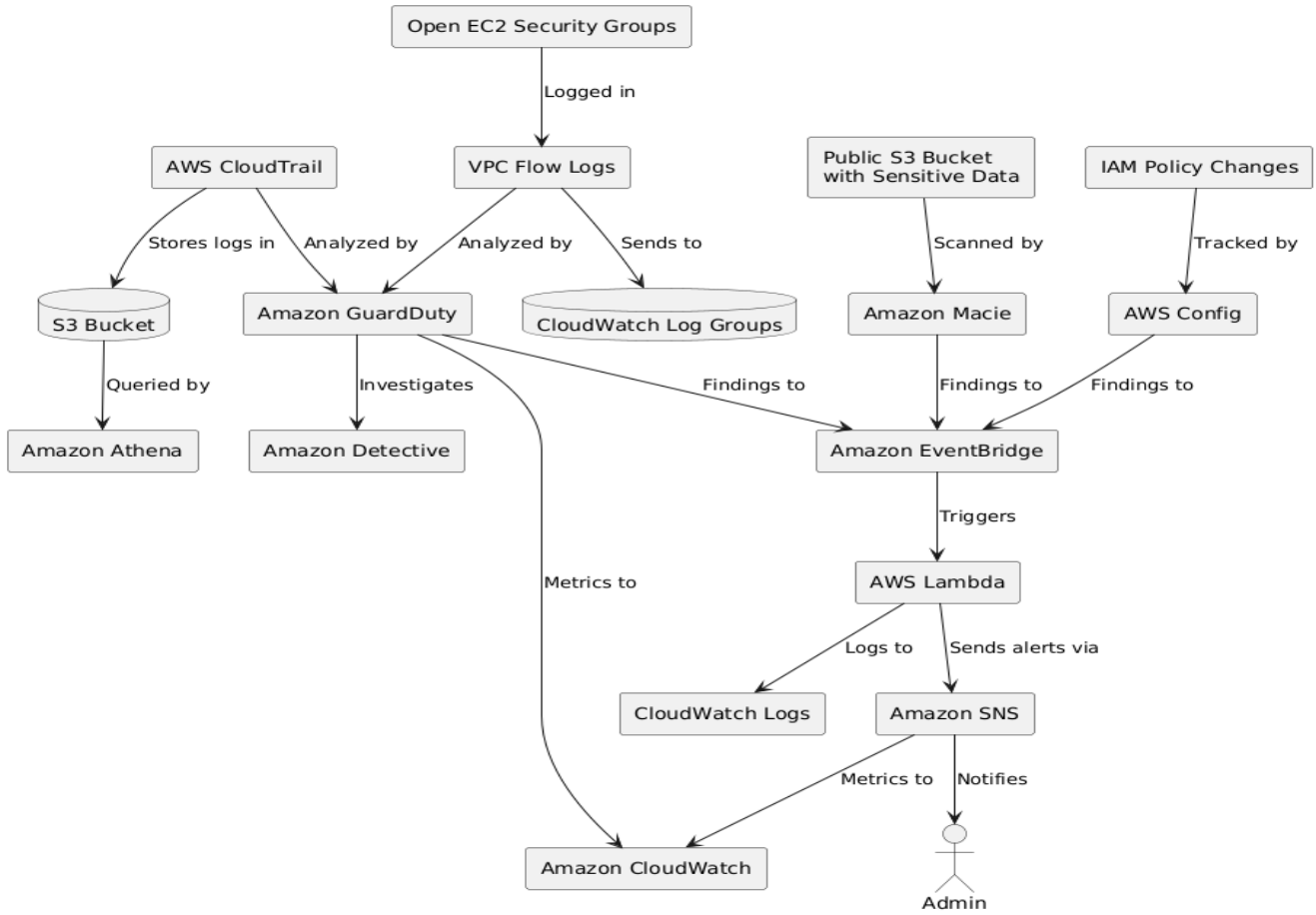

Fig 3.1 Cloud Threat Detection & Response Architecture

Identified threats and anomalies are forwarded as structured events to Amazon EventBridge, which dynamically triggers AWS Lambda for alert handling and remediation

actions. Automated alerts are dispatched through Amazon SNS, notifying administrators in real time. Simultaneously, findings are further analyzed using Amazon Athena for log querying and Amazon Detective for behavioral correlation and forensic insights. All operational metrics, Lambda executions, and log events are monitored via Amazon CloudWatch, ensuring continuous observability across the detection pipeline.

This architecture is purposefully designed to be scalable, cost-efficient, and agentless, aligning with modern Zero Trust and DevSecOps principles. It demonstrates how a cohesive, cloud-native threat detection and response system can be built entirely using AWS services without reliance on third-party tools or infrastructure.

3.2 Data Flow Diagram

The Data Flow Diagram (DFD) for the Cloud Threat Detection & Response Engine represents the end-to-end movement of data across various AWS services used to simulate, detect, analyze, and respond to cloud-based security threats. The flow begins with simulated threat events such as EC2 port scans, IAM policy modifications, and the exposure of sensitive data through public S3 buckets. These actions serve as inputs to the pipeline, generating valuable logs and telemetry for detection.

AWS CloudTrail captures API-level activities, including IAM changes, S3 access attempts, and administrative actions. Simultaneously, VPC Flow Logs record network-level traffic metadata, such as source/destination IPs, protocols, and accepted or rejected requests. These logs are stored in Amazon S3 and CloudWatch Log Groups, forming the foundation of the system's observability and analytics capabilities.

The detection layer comprises Amazon GuardDuty, which analyzes both VPC Flow Logs and CloudTrail for known threat signatures and anomalies, Amazon Macie, which scans S3 buckets for sensitive information, and AWS Config, which tracks resource compliance and configuration drift. These services generate structured findings when a potential threat or misconfiguration is detected.

Detected events are routed through Amazon EventBridge, which evaluates them against defined rules and triggers AWS Lambda functions for automated processing. Lambda formats the findings into alert messages and forwards them to Amazon SNS, which instantly notifies system administrators via email. Simultaneously, Amazon Athena enables ad-hoc querying of raw logs stored in S3 for deeper investigation, while Amazon Detective visualizes entity behavior and relationships to support forensic analysis.All metrics, logs, and Lambda executions are monitored using Amazon CloudWatch, ensuring complete visibility into the system's operations and response flow.

This data pipeline demonstrates how native AWS services can be orchestrated to build a highly scalable, automated, and real-time security detection framework—eliminating the need for third-party tools while aligning with Zero Trust and cloud-native security principles.

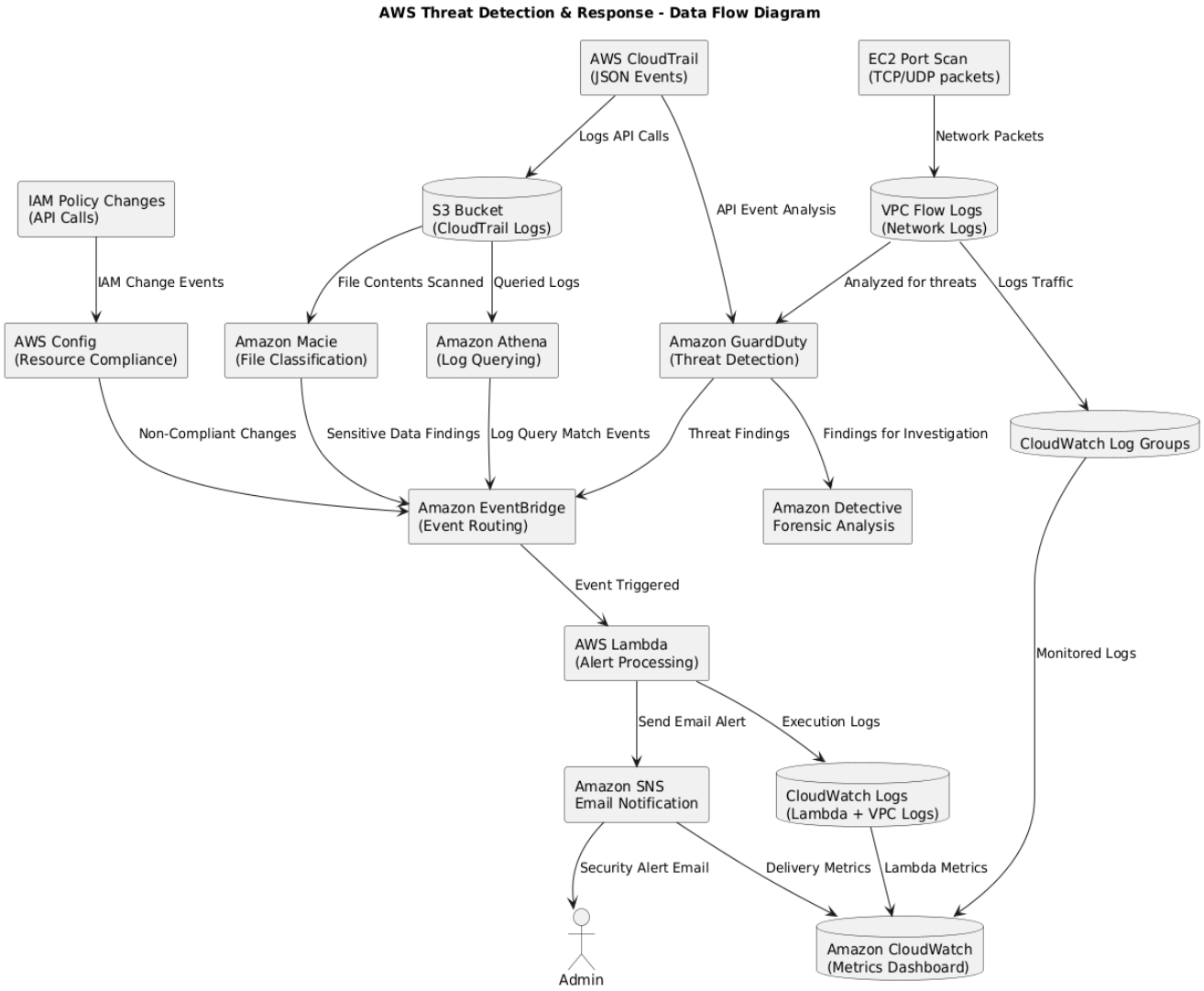

Fig 3.2 : Data Flow Diagram of Cloud Threat Detection & Response Pipeline

3.3 Component-Wise Design Overview

The proposed system architecture is structured into modular components, each responsible for a specific function in the threat detection and response pipeline. The Data Collection Layer captures API and network activity through CloudTrail and VPC Flow Logs. The Threat Detection Layer uses GuardDuty, Macie, and Config to identify suspicious behavior and misconfigurations. The Analysis Layer leverages Athena and Detective for investigation and correlation of findings. Finally, the Response Automation Layer utilizes EventBridge, Lambda, and SNS to automate alerting and notifications. Each component operates in a serverless, scalable manner, contributing to a secure, cost-effective, and cloud-native security solution.

3.3.1 Data Collection Layer

The Data Collection Layer forms the foundation of the threat detection pipeline by capturing critical logs and telemetry data from various AWS resources. This layer ensures that both network-level traffic and API-level activity are continuously recorded and made available for

downstream analysis and detection.

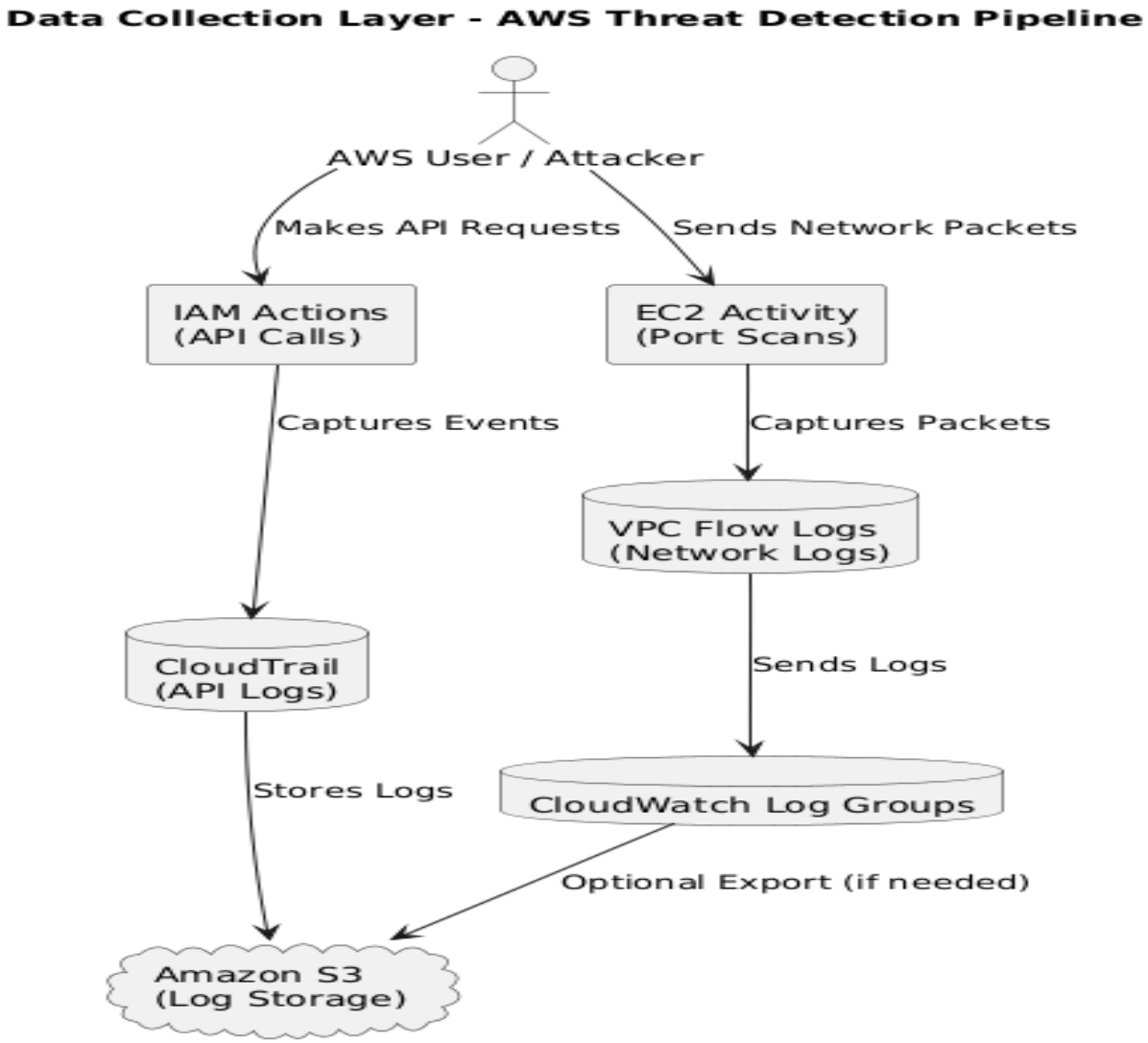

Fig 3.3 Data Collection Layer

Key Components:-

1. AWS CloudTrail
   CloudTrail records all API calls made within the AWS environment, including those from the AWS Management Console, SDKs, CLI, and other services. It provides full visibility into user activity, service usage, and security-relevant changes. The logs are stored in Amazon S3, enabling centralized and persistent storage for querying and alerting.
2. VPC Flow Logs
   VPC Flow Logs capture IP-level traffic information for network interfaces within the VPC. These logs include metadata such as source/destination IP, ports, protocol, and traffic direction (accept/reject). They are forwarded to CloudWatch Log Groups, allowing near real-time analysis of network behavior and potential reconnaissance activities.
3. Amazon S3 (Log Storage)
   Acts as a durable storage layer for CloudTrail logs and other collected telemetry. It serves as the primary source for log analysis tools like Amazon Athena and Amazon

Macie to query and scan for suspicious events and sensitive data.

4. CloudWatch Log Groups
   Stores logs generated by VPC Flow Logs and Lambda functions, making them accessible for monitoring, metric extraction, and correlation with other findings. CloudWatch Logs also power alerting dashboards via Amazon CloudWatch.

Purpose & Outcome:-
This layer guarantees the availability, persistence, and accessibility of raw security data. By centralizing log collection, it enables advanced detection, querying, and automated response mechanisms in upper layers of the architecture.

3.3.2 Threat detection layer:

The Threat Detection Layer serves as the core security engine of the system, analyzing collected logs to identify malicious activity. This layer leverages Amazon GuardDuty, Amazon Macie, and AWS Config to detect threats across multiple attack vectors unauthorized API calls, exposed sensitive data, and misconfigured resources. GuardDuty processes CloudTrail management events and VPC Flow Logs using machine learning and threat intelligence feeds to identify anomalies like brute-force attacks or suspicious instance behavior. Meanwhile, Macie scans S3 buckets for sensitive data exposure (PII, credentials) and policy violations, while AWS Config continuously monitors resource configurations for compliance deviations. Detected threats are converted into structured findings and forwarded to Amazon EventBridge, which orchestrates automated responses.

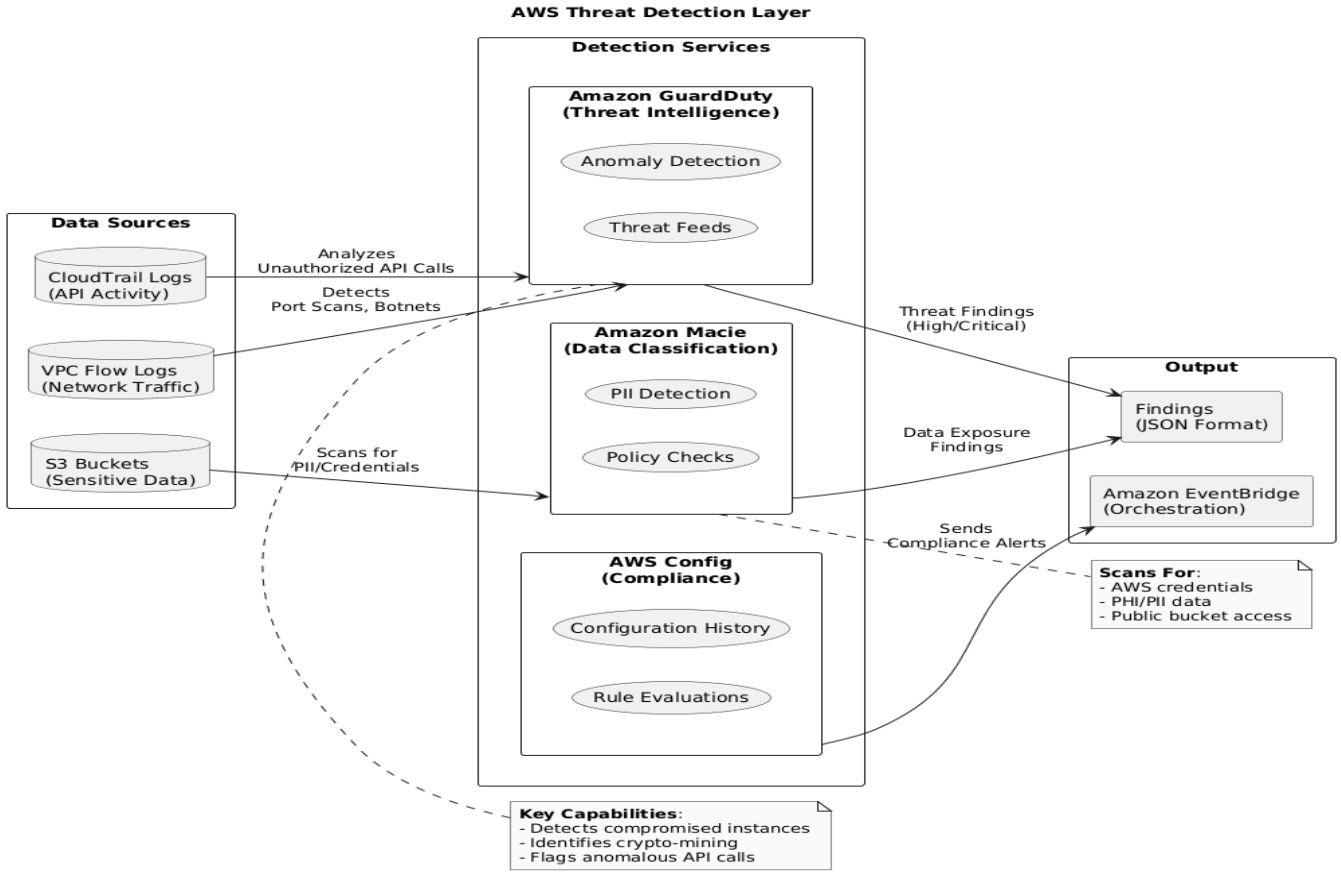

Fig 3.4 : Threat Detection Layer

3.3.3 Response Automation Layer

The Response Automation Layer is the reactive backbone of the threat detection system, designed to execute immediate and standardized countermeasures when security incidents are identified. This layer leverages Amazon EventBridge as the central nervous system to ingest findings from GuardDuty, Macie, and AWS Config, then triggers targeted actions through AWS Lambda functions. For example, upon detecting a critical GuardDuty finding (e.g., UnauthorizedAccess:IAMUser/MaliciousIPCall), EventBridge invokes a Lambda function that automatically revokes IAM keys, isolates compromised EC2 instances, or triggers AWS Systems Manager for remediation. Notifications are routed through Amazon SNS to alert security teams via email or SMS, with severity-based prioritization. The architecture adheres to the NIST Incident Response Framework (Identify-Contain-Eradicate) while enabling customizable playbooks through Infrastructure-as-Code (Terraform/CloudFormation). By automating repetitive tasks, this layer reduces mean-time-to-response from hours to seconds and ensures consistent enforcement security policies.

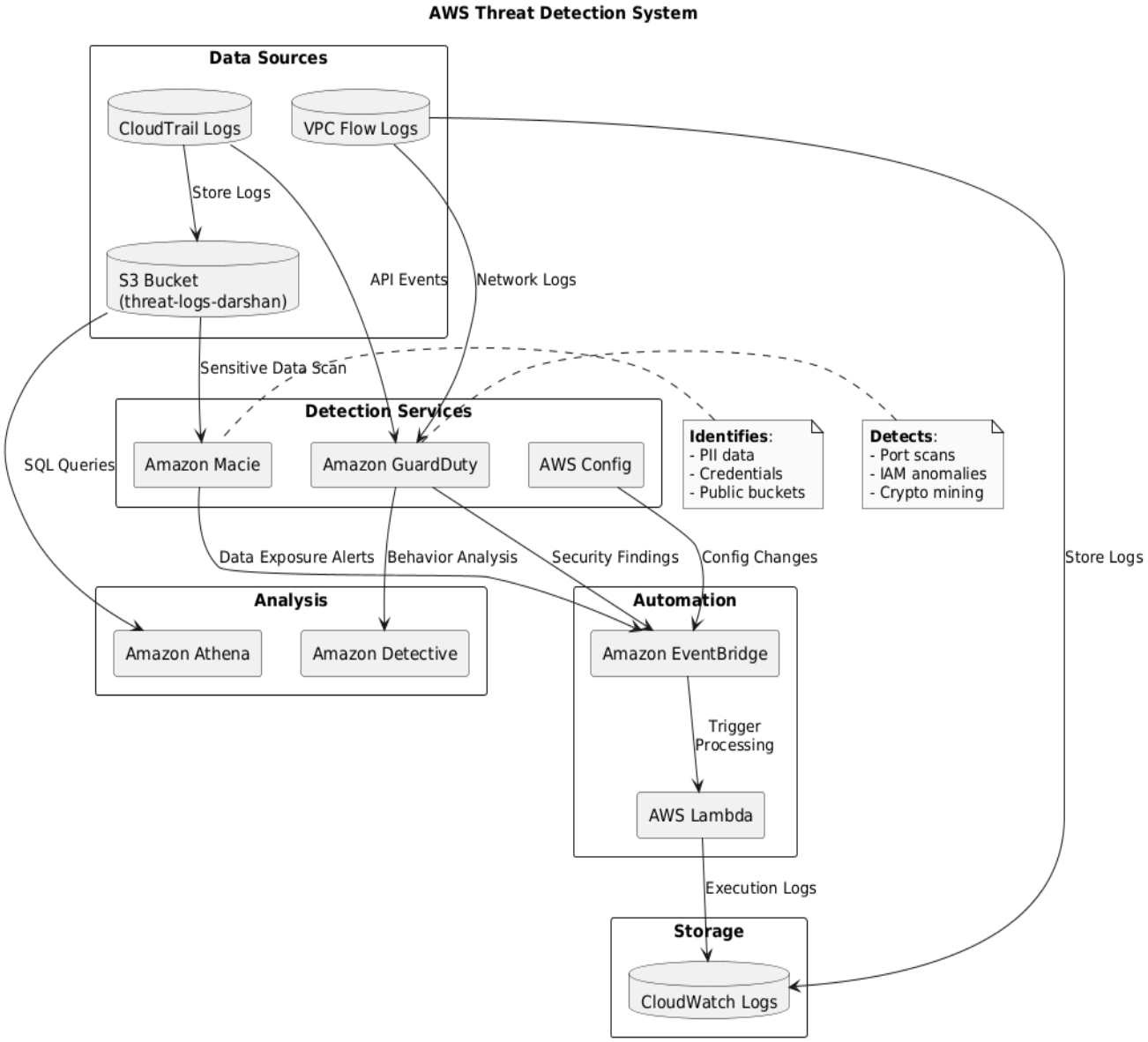

Fig 3.5 : Response Automation Layer

3.4 AWS Services Used & Justification

The Cloud Threat Detection & Response Engine integrates a carefully curated set of AWS services, selected for their seamless interoperability, scalability, and ability to provide comprehensive threat visibility without external dependencies. Each service plays a distinct and critical role within the multi-layered architecture.

- **AWS CloudTrail** is the cornerstone of auditability, capturing every API interaction across the AWS environment. This includes sensitive actions such as IAM policy changes, EC2 provisioning, and S3 access attempts, allowing for granular activity tracking.
- **VPC Flow Logs** provide critical network-level visibility by recording IP traffic data across ENIs (Elastic Network Interfaces), which is vital for detecting port scanning, brute force attempts, and lateral movement.
- **Amazon S3** serves as the centralized and durable storage layer for CloudTrail logs, making them queryable by downstream services such as Amazon Athena and Amazon Macie.
- **CloudWatch Log Groups** store log streams generated by Lambda functions and VPC Flow Logs, enabling real-time monitoring, anomaly detection, and metric visualization through Amazon CloudWatch.
- **Amazon GuardDuty** continuously monitors data from CloudTrail and VPC Flow Logs to detect suspicious activity, including reconnaissance scans, credential compromise, and known malware behavior. It forms the backbone of threat intelligence within the system.
- **Amazon Macie** adds a layer of data security by scanning S3 buckets for sensitive information such as emails, names, and credentials, and alerts when publicly accessible data is detected.
- **AWS Config** enforces governance by evaluating configurations of AWS resources against security best practices, flagging misconfigurations such as publicly accessible buckets and permissive IAM roles.
- **Amazon Athena** enables on-demand, serverless querying of logs stored in S3 using SQL, allowing analysts to perform forensic investigations and validate threat hypotheses without provisioning infrastructure.
- **Amazon Detective** provides interactive visualizations of threat timelines, IP addresses, and resource behavior to accelerate incident triage and root cause analysis.
- **Amazon EventBridge** serves as the automation backbone, routing findings from detection services to **AWS Lambda** based on event rules.
- **AWS Lambda** acts as the processing engine, formatting threat data, triggering workflows, and integrating response logic all executed in a scalable and stateless manner.
- **Amazon SNS** is used for real-time alerting, pushing formatted notifications to administrators via email and enabling immediate response.
- **Amazon CloudWatch** serves as the observability hub, aggregating logs, metrics, and alert delivery status, thereby ensuring full visibility across the detection and response lifecycle.

Each service was purposefully selected to ensure a secure, scalable, and fully automated architecture that reflects modern cloud-native security principles.

3.5 Security Considerations

Security was embedded into every stage of the system's design, with the architecture following a **Zero Trust model** and adhering to **AWS Security Best Practices**. The principle of **least privilege** was strictly enforced when assigning IAM roles and policies, ensuring that services and automation components operate with only the permissions they require.

To ensure full visibility and auditability, **AWS CloudTrail** and **VPC Flow Logs** were enabled across the account, capturing API calls and network interactions, respectively. Logs were stored securely in **Amazon S3**, with server-side encryption (SSE-S3) enabled by default to protect data at rest. Additionally, **CloudWatch Logs** stored Lambda outputs and network traffic metadata in encrypted streams.

Security controls also extended to compliance validation. **AWS Config** continuously evaluated the configuration state of resources, detecting drift and unauthorized changes to IAM roles, security groups, or S3 access policies. **Amazon Macie** actively scanned S3 buckets for unintentional data exposure, preventing sensitive information leakage.

All Lambda functions ran within private subnets to minimize public exposure. Communication between services was handled via IAM-secured, event-driven mechanisms (EventBridge → Lambda → SNS), ensuring that data never left the AWS ecosystem. **SNS notifications** were formatted for clarity and sent only to trusted recipients.

This layered defense-in-depth strategy ensured protection from both external threats and insider misconfigurations, while maintaining a robust audit trail and immediate alerting mechanism to reduce detection and response times.

3.6 Cost & Performance Analysis

The solution was architected with cost-efficiency and high performance in mind, taking full advantage of the **serverless and pay-as-you-go** nature of AWS services. By utilizing **event-driven components** and **on-demand querying**, the system incurs charges only when actions are performed, making it both scalable and economically viable even under heavy load.

- **AWS CloudTrail** and **VPC Flow Logs**, both covered under the AWS Free Tier, provide continuous data collection with minimal cost. **Amazon S3** offers durable log storage with infrequent access, optimized using Standard storage class during experimentation.
- **Lambda functions**, being stateless and fast-executing, incur millisecond-based charges and scale automatically with incoming events. Combined with **EventBridge**, this allowed near real-time automation without any provisioning overhead.
- **Amazon Athena** follows a per-query pricing model, and was used selectively to extract insights from S3-stored logs, avoiding the cost of running long-standing compute instances.
- **Amazon GuardDuty** and **Amazon Macie**, although metered services, were operated within free trial limits for demonstration purposes, showcasing their capabilities without incurring additional cost.
- **Amazon SNS** and **CloudWatch Logs** generated minimal usage-based charges, since alert volumes and log sizes were moderate. **Amazon Detective** was used for visualization during triage and remained within acceptable usage thresholds.

Overall, the architecture delivered on performance and detection accuracy

Chapter 4

IMPLEMENTATION

4.1 Environment Setup

The project was deployed in an isolated AWS environment within the us-east-1 (N. Virginia) region to ensure compatibility with all required services. A serverless architecture was adopted, leveraging native AWS tools without provisioning EC2 instances or external agents.

A dedicated S3 bucket (threat-logs-darshan) was created to store logs and findings from services such as CloudTrail, AWS Config, and Macie. IAM roles were configured with the principle of least privilege to secure inter-service communication. Core services including CloudTrail, VPC Flow Logs, GuardDuty, Macie, AWS Config, and Detective were activated to enable full visibility into API activity, network behavior, compliance drift, and security events. To automate threat response, Amazon EventBridge, Lambda, and SNS were integrated into a serverless alerting pipeline. The entire environment operated within AWS Free Tier limits and was configured manually via the AWS Management Console to reflect real-world deployment conditions. This setup laid the foundation for simulating, detecting, and responding to cloud security threats in real time.

4.2 Step-by-Step Configuration of AWS Services

This section outlines the systematic configuration of each AWS service integrated into the Cloud Threat Detection & Response Engine. The setup focused on enabling comprehensive visibility, real-time threat detection, and automated response using only native, serverless AWS components.

4.2.1 Amazon S3 (Log Storage)

Amazon S3 is a highly scalable object storage service used to store and manage large volumes of data securely. In this project, it was configured as the central log repository to store output from AWS CloudTrail, AWS Config, and other detection tools. By serving as the backbone of the logging infrastructure, S3 enabled long-term, queryable, and tamper-proof storage for security logs making it essential for threat detection, auditing, and incident investigation across the cloud environment.

STEP 1: Create S3 Bucket for Logs
Action:

1. ” Go to **S3 Console** → Click **"Create bucket"**
2. Name it: threat-logs-darshan *(or any unique name)*
3. Region: **N. Virginia (us-east-1)** (Recommended)
4. **Uncheck** “Block all public access”

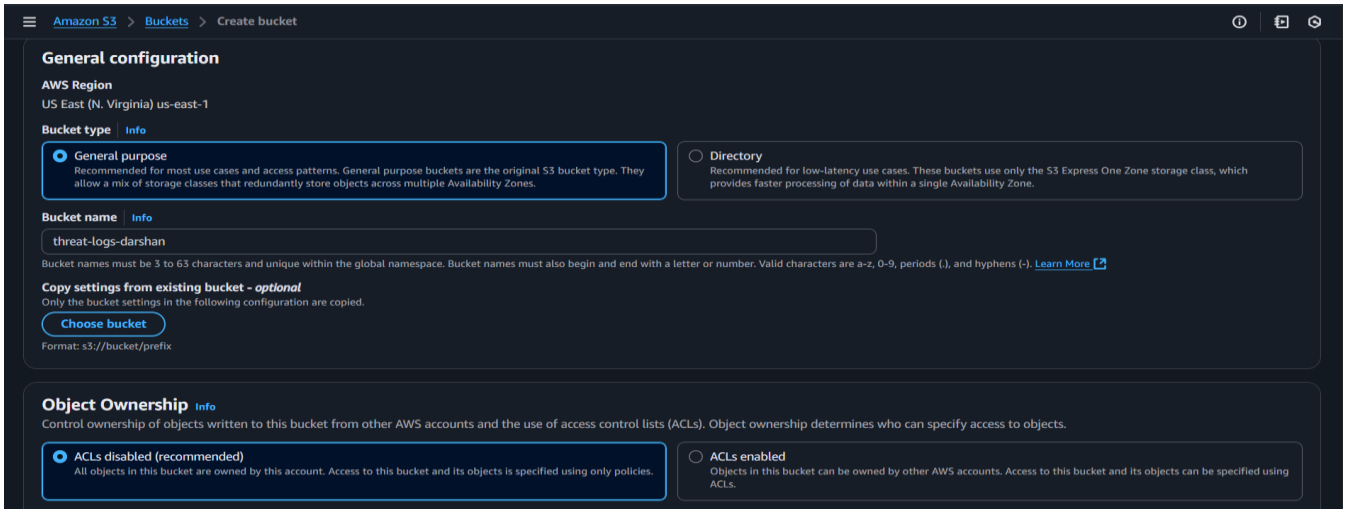

Fig 4.1 : Log Storage Setup using Amazon S3

5. Keep rest settings default → Click **Create Bucket**

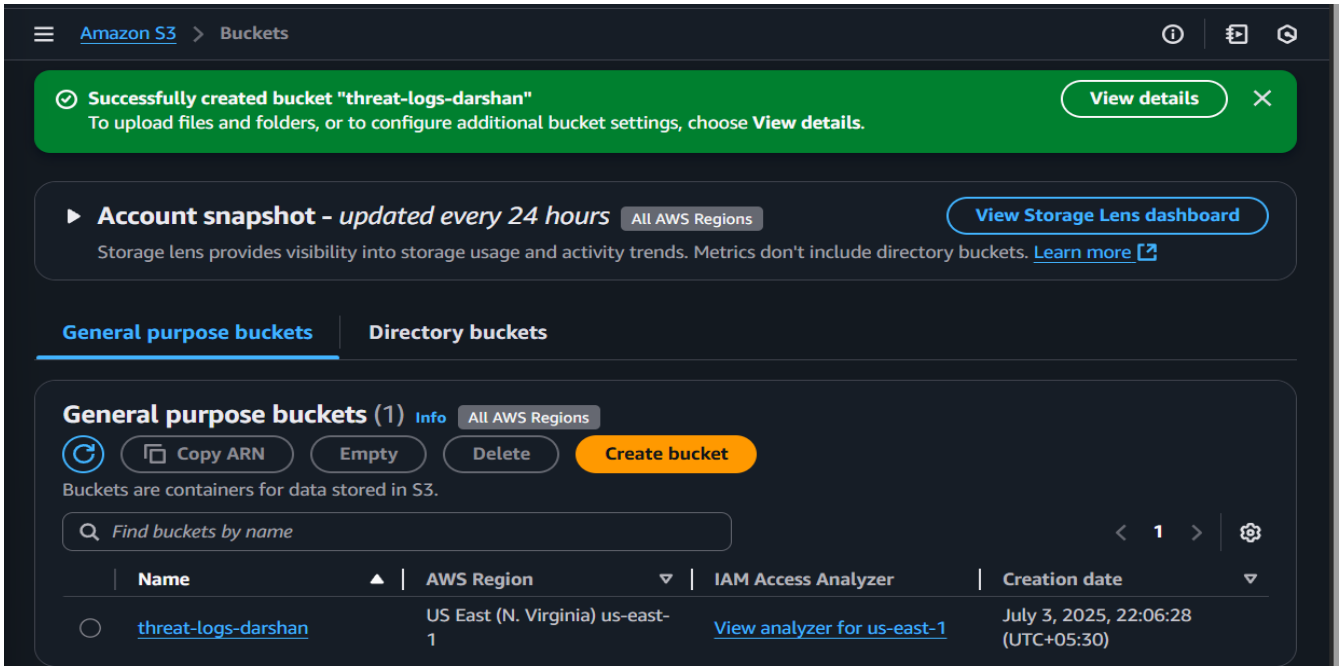

Fig 4.2 : Successfully Created S3 Bucket for Centralized Log Storage

4.2.2 AWS CloudTrail (API Logging)

AWS CloudTrail is a logging service that records all API calls made within an AWS account, providing a complete audit trail of user and service activity. In this project, CloudTrail was used to capture high-fidelity logs related to IAM changes, S3 access, and other critical events. It played a foundational role in the threat detection pipeline by enabling services like GuardDuty, AWS Config, and Athena to analyze API behavior, detect anomalies, and support forensic investigation.

STEP 2: Enable CloudTrail

Action:

1. Go to **CloudTrail Console**

2. Click **“Create trail”**

3. Trail name: threat-detection-trail

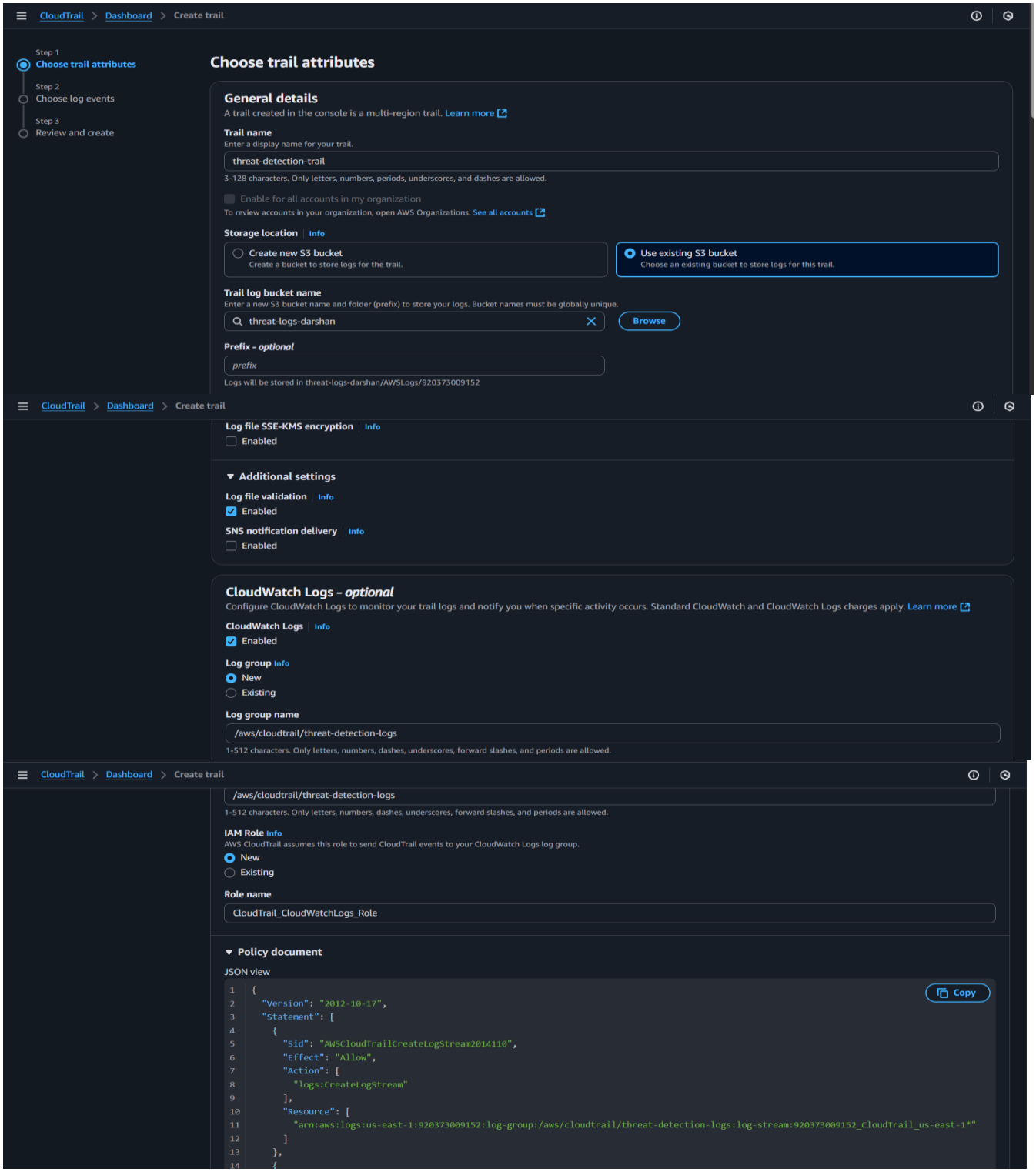

Fig 4.3 : AWS CloudTrail Setup for Centralized API Activity Logging

4. Choose Apply to all regions

5. Log type: Management events + Data events

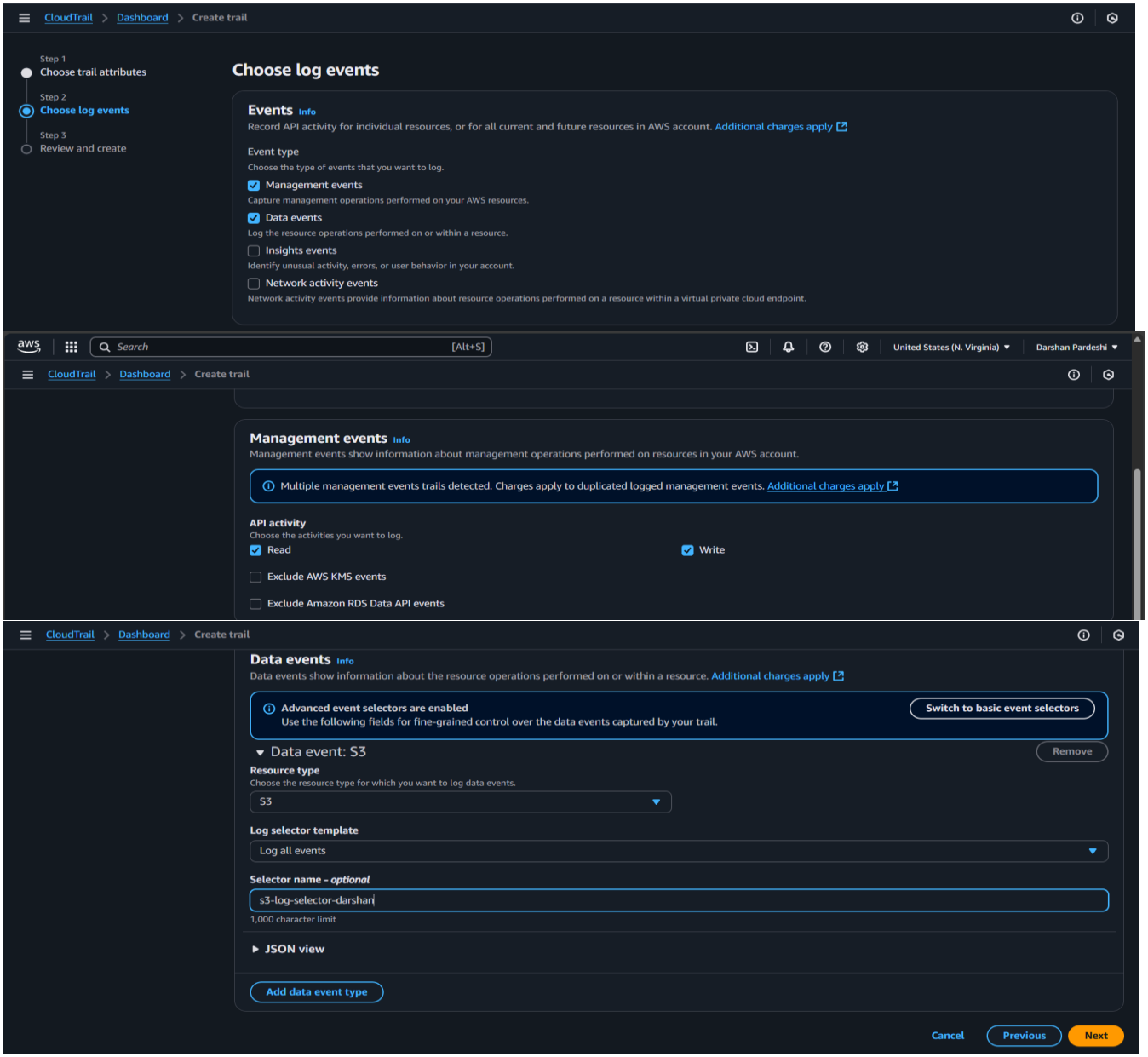

Fig 4.4 : CloudTrail Trail Configuration with S3 & CloudWatch Integration

6. Choose the S3 bucket you just created (threat-logs-darshan)
7. Enable CloudWatch Logs

Then click **Create Trail**

4.2.3 AWS IAM

AWS Identity and Access Management (IAM) enables secure access control by defining who can access specific AWS resources and under what conditions. In this project, IAM was used to assign least-privilege roles to services like Lambda, CloudTrail, and Macie, ensuring secure interactions between components. It also played a critical role in simulating privilege misconfigurations, helping validate the system's ability to detect and respond to excessive or risky permissions.

Step 3: Create IAM Role
Go to IAM → Roles → Create role
Trusted entity type:- Custom trust policy

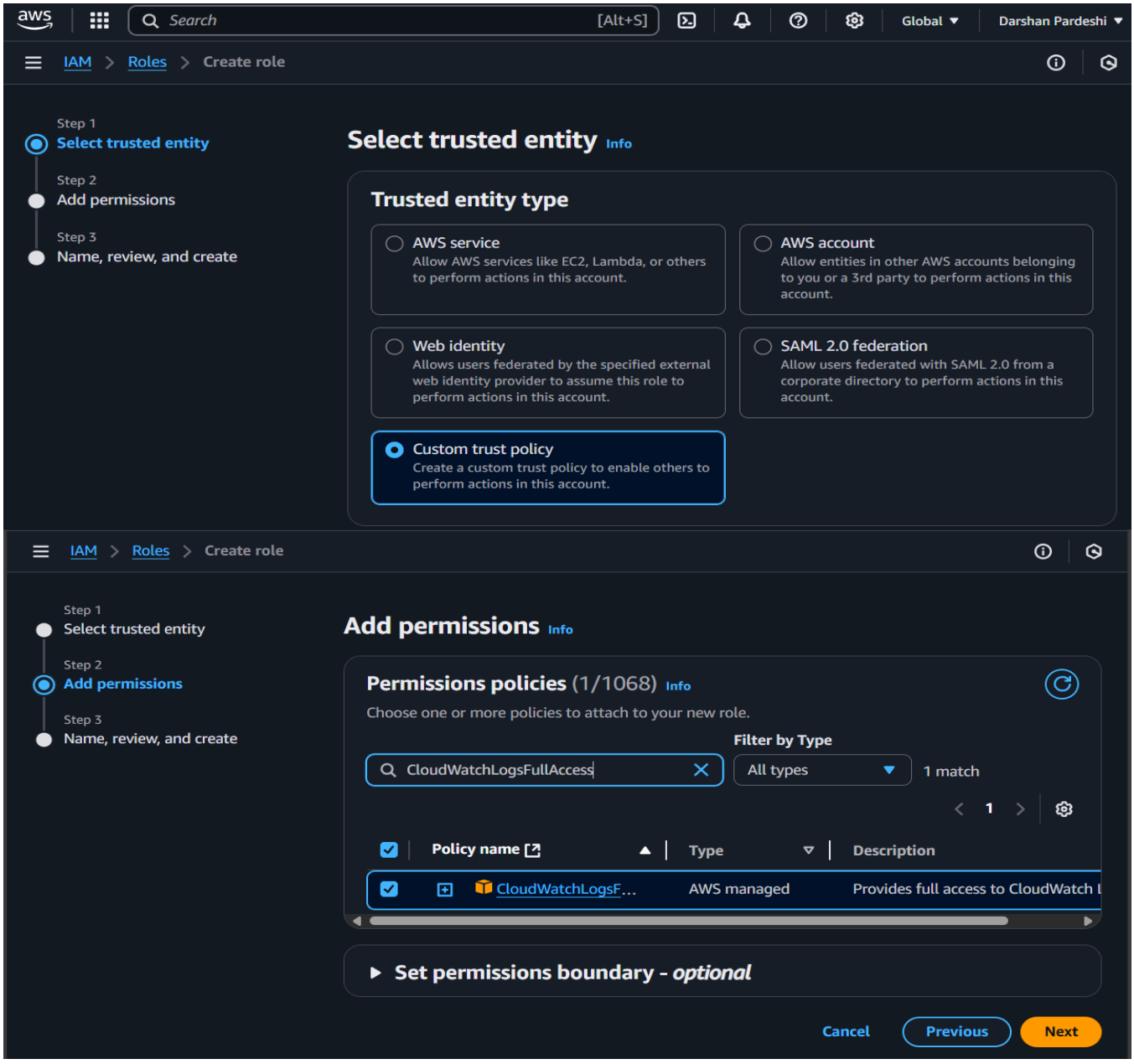

Fig 4.5 : IAM Role Creation for Secure Inter-Service Communication

Role name :- vpc-flowlogs-cloudwatch-role-Darshan

Click Next .

4.2.4 VPC Flow Logs (Network Traffic)

VPC Flow Logs capture metadata about IP traffic going to and from network interfaces within a Virtual Private Cloud (VPC). In this project, they were used to monitor

network activity such as port scans and unauthorized access attempts. By feeding traffic data to GuardDuty and CloudWatch, VPC Flow Logs provided essential visibility into low-level network behavior, enabling early detection of suspicious traffic patterns.

STEP 4: Enable VPC Flow Logs
This captures **network activity** (accepted/rejected connections) for GuardDuty, Athena, and CloudWatch.
Action:

1. Go to **VPC Console → Your VPCs**
2. Choose your main VPC (usually named "default" or one you created)
3. Scroll down → Click **"Create flow log"**

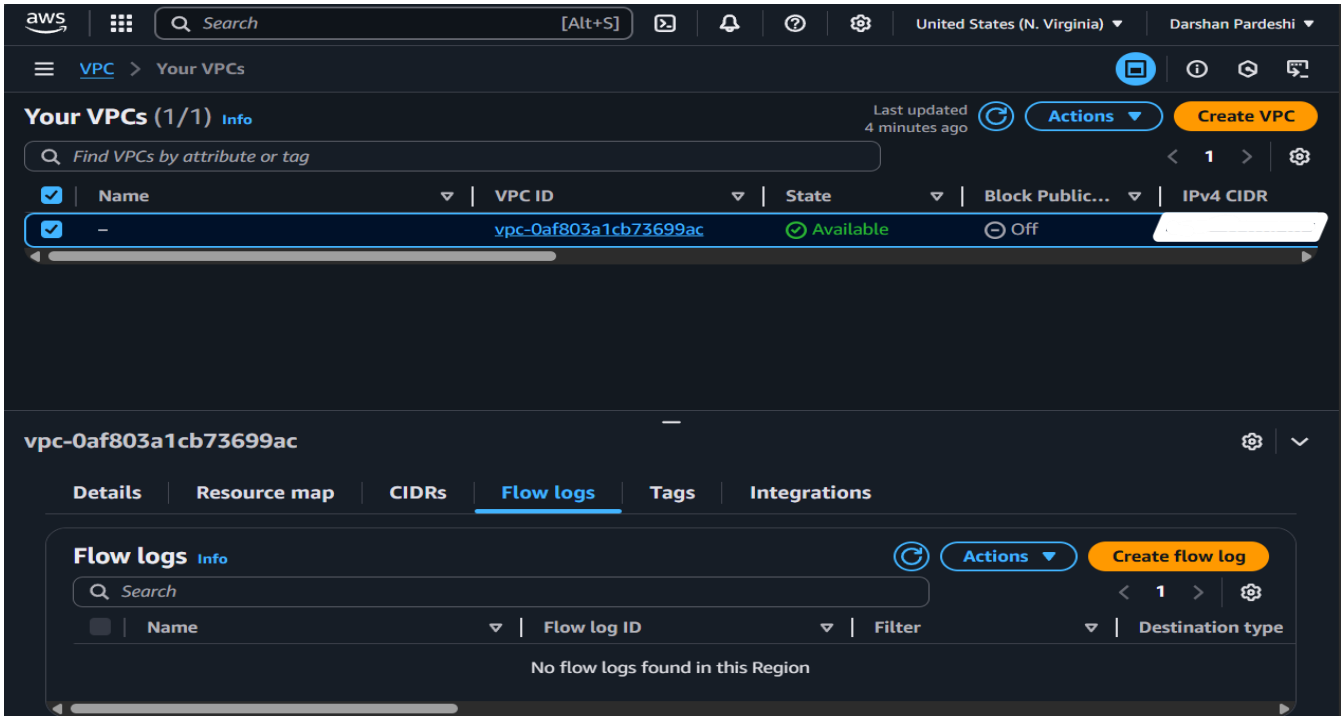

Fig 4.6 : VPC Flow Logs Configuration for Capturing Network Traffic Metadata

Filter :- All
Maximum aggregation interval :- 1 minute
Destination :- Send to CloudWatch Logs
Log group :- Create new: vpc-flow-logs-darshan

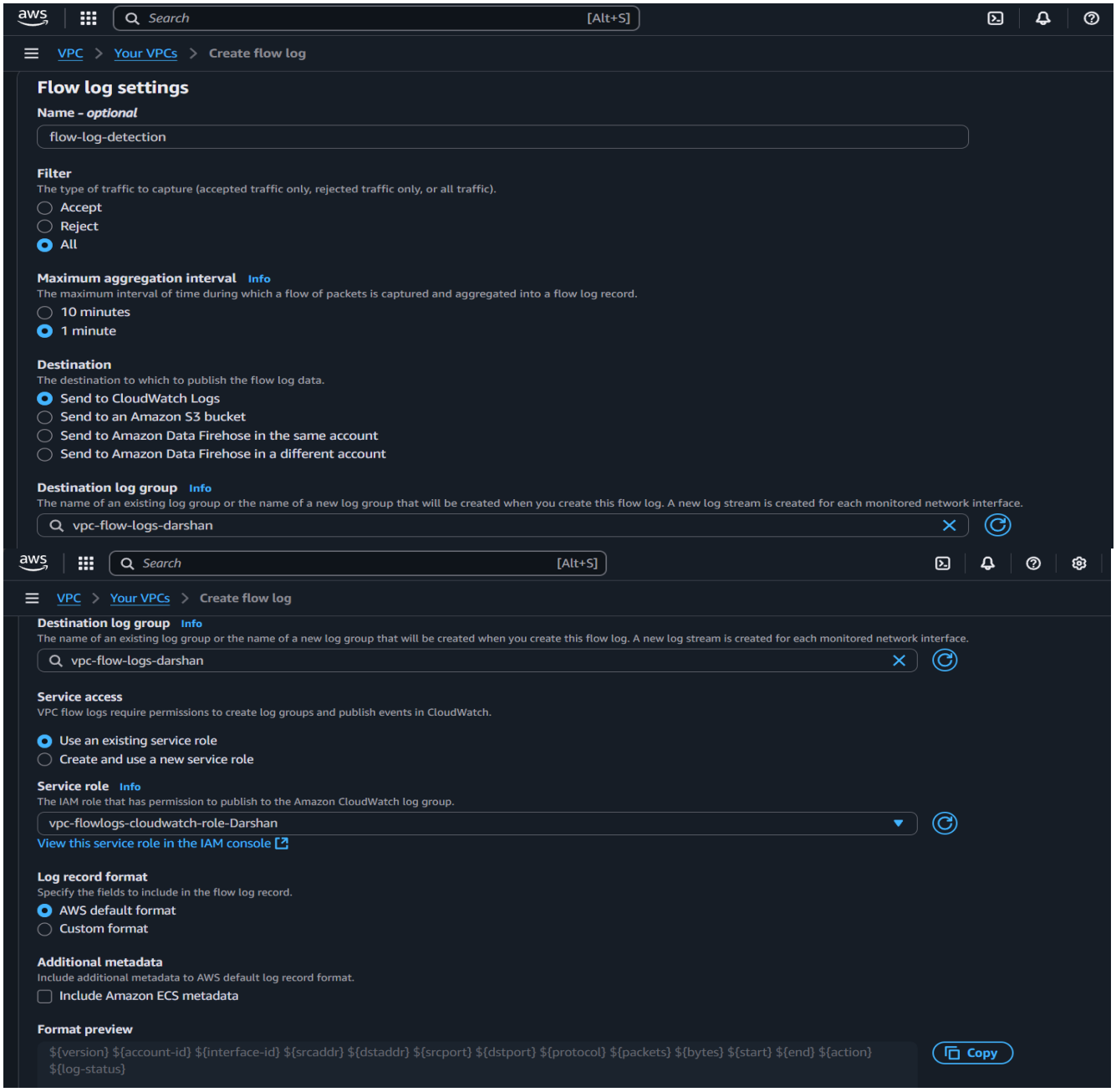

Fig 4.7 : Final VPC Flow Log Configuration with Custom Log Group Setup

Click Create Flow log.

4.2.5 AWS GuardDuty (Threat Detection)

Amazon GuardDuty is a threat detection service that continuously analyzes CloudTrail, VPC Flow Logs, and DNS logs to identify malicious or unauthorized activity. In this project, it was used to detect simulated threats such as port scans and public access misconfigurations. GuardDuty played a critical role in the detection layer by generating real-time security findings that triggered automated alerting workflows.

STEP 5: Enable Amazon GuardDuty (Threat Detection Engine)
GuardDuty will now start analyzing:

- CloudTrail logs (IAM activity)
- VPC Flow Logs (network activity)
- DNS logs
  To detect threats like brute force, port scans, credential misuse, etc.

Action:

1. Go to **Amazon GuardDuty Console**
2. Click **"Enable GuardDuty"**
3. Keep default settings
    - Protect current region
    - Auto-enable for new resources (optional)

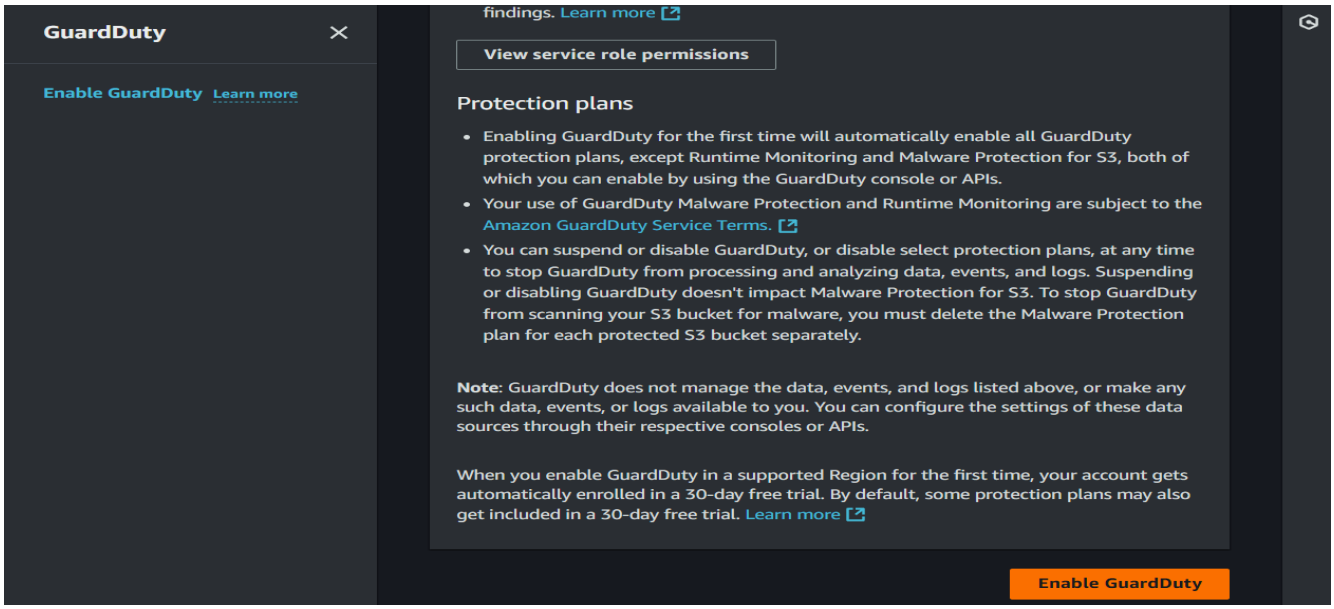

Fig 4.8 : Enabling Amazon GuardDuty for Real-Time Threat Detection

Once enabled:

- It will start analyzing logs from CloudTrail and VPC
- You'll start seeing **Findings** after simulations

4. Click **Enable**

4.2.6 Amazon Macie (Sensitive Data Discovery)

Amazon Macie is a data security service that uses machine learning to discover and protect sensitive information in S3 buckets. In this project, Macie was used to scan for exposed data such as names, emails, and passwords uploaded during simulated attacks. It

strengthened the pipeline by identifying data leakage risks in real time and integrating with alerting workflows for immediate response.

STEP 6: Enable Amazon Macie (Data Exposure Detection)
Macie scans your S3 buckets to:

- Detect **sensitive data** (like emails, names, credentials)
- Warn if buckets are **public** or misconfigured

Action:

1. Go to **Amazon Macie Console**
2. Click **"Get Started"**

Fig 4.9 : Enabling Amazon Macie for Sensitive Data Discovery in S3 Buckets

3. Macie will auto-enable for your region
4. Choose your S3 bucket (threat-logs-darshan) for scanning
5. Click **Enable Macie**

It will start scanning automatically in the background. You can later:

- View **findings** about sensitive data or risky configurations.
- Integrate those into GuardDuty via EventBridge (we'll do that soon).

4.2.7 AWS Config (Compliance & Change Tracking)

AWS Config is a monitoring service that continuously records configuration changes and evaluates them against predefined compliance rules. In this project, it was used to detect misconfigurations such as public S3 access and overly permissive IAM policies. Config added a compliance layer to the pipeline by identifying deviations from best practices and helping maintain secure resource states over time.

STEP 7: Enable AWS Config (Change Tracking & Compliance)
AWS Config continuously monitors and records changes to:

- IAM roles and policies
- S3 bucket permissions
- Security groups
- All other resources

This helps detect misconfigurations, privilege changes, or compliance violations.
Action:

1. Go to **AWS Config Console**
2. Click **"Get started"** or **"Setup"**
3. **Resource recording**:
    - Record all resources in this region

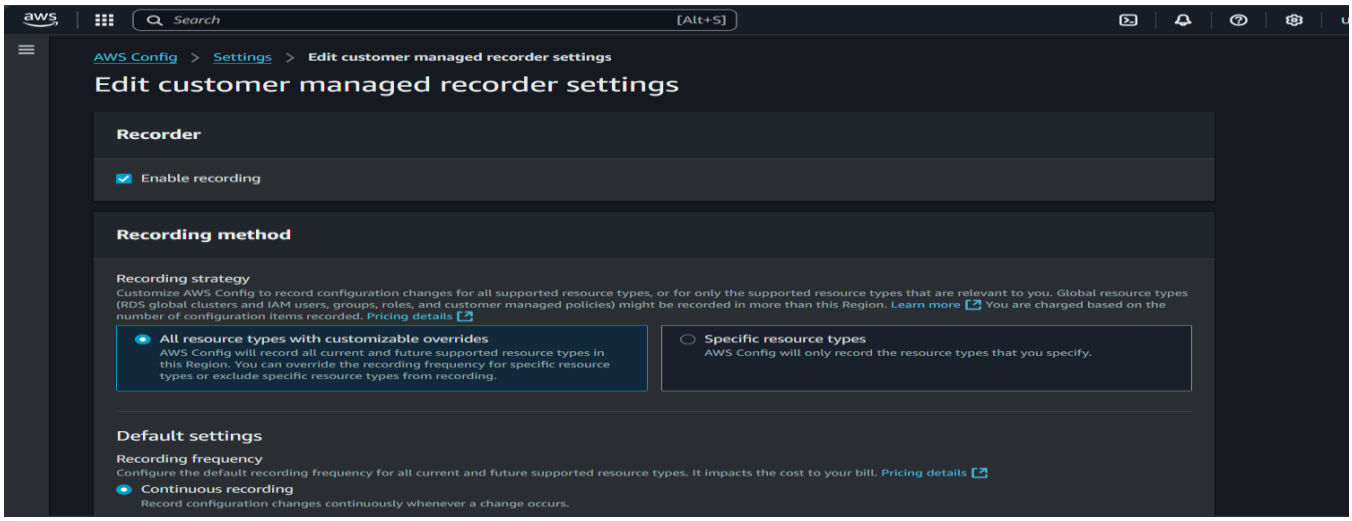

Fig 4.10 : AWS Config Setup for Continuous Compliance and Resource Change Tracking

4. **S3 bucket for delivery**:
    - Choose your bucket: threat-logs-darshan

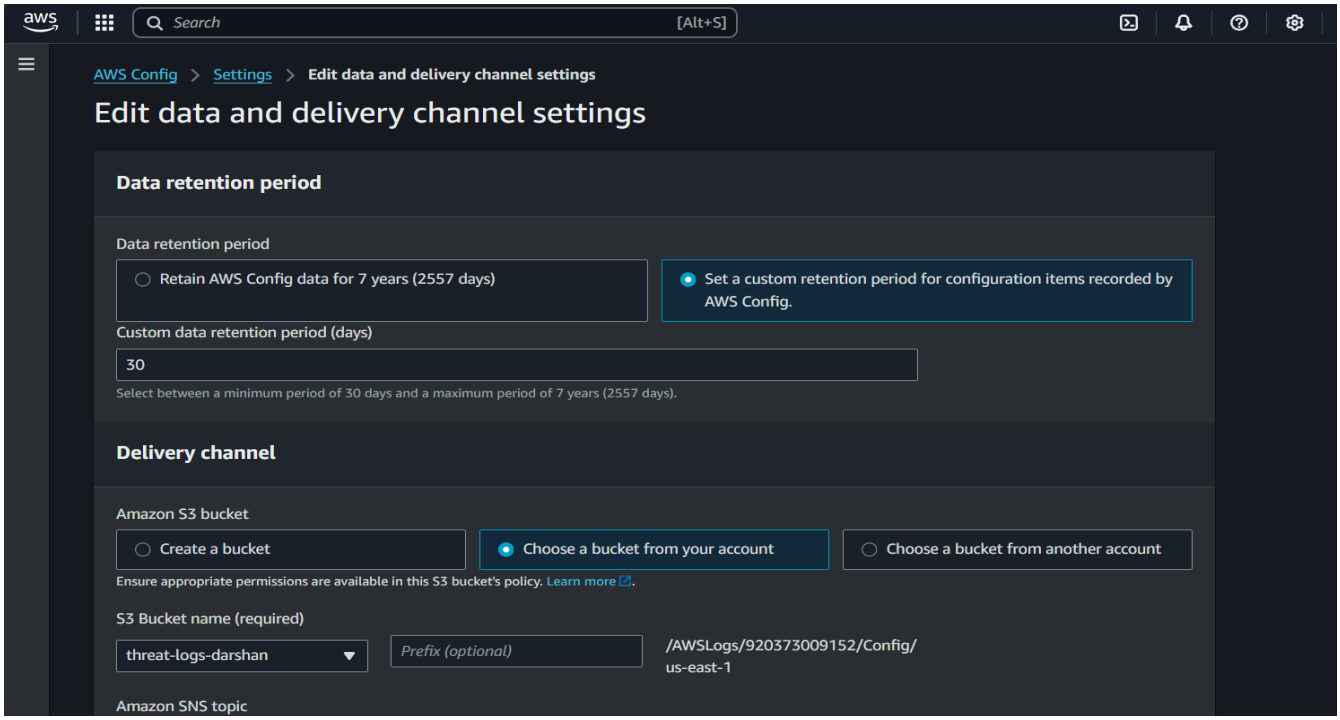

Fig 4.11 : Selecting S3 Bucket as Delivery Channel for AWS Config Snapshots

Keep SNS box uncheked here .

5. **AWS Config role**:
    - Choose **Create AWS Config service-linked role**

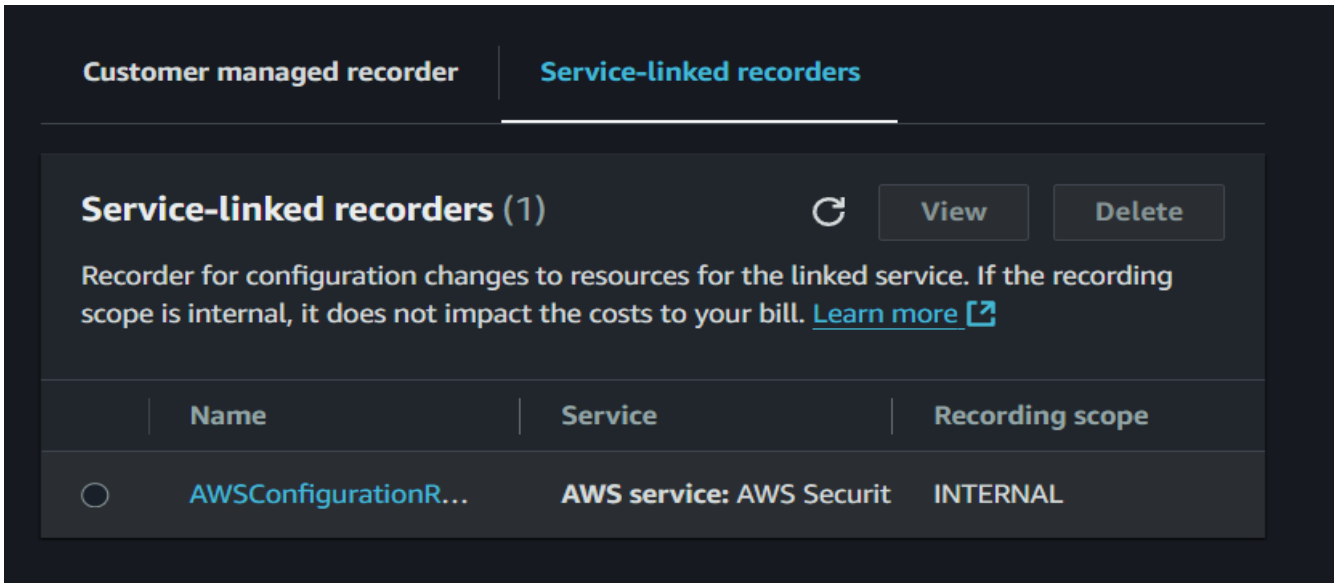

Fig 4.12 : Finalizing AWS Config Setup with Service-Linked Role and Resource Recording

6. Keep rest default → Click **Confirm / Submit**

Once enabled:

- Config will start tracking changes
- You'll see a **timeline of configuration changes**

- It helps when correlating with GuardDuty and Detective

4.2.8 AWS Detective (Investigation & Behavior Analysis)

AWS Detective is a security analysis service that helps visualize relationships and activity patterns across AWS resources. In this project, it was used to investigate findings from GuardDuty and analyze IAM behavior, IP traffic, and resource interactions. Detective enhanced the response pipeline by enabling rapid root cause analysis and timeline reconstruction during simulated attacks.

STEP 8: Enable AWS Detective (Investigation Engine – Optional but Powerful)
Detective helps you:

- Investigate **GuardDuty findings** with **graph-based analysis**
- Visualize IAM activity, EC2 behavior, and network flows over time
- Perform forensic triage after incidents

Action:

1. Go to **AWS Detective Console**
2. Click **"Enable Detective"**
3. Keep default settings
    - Choose your GuardDuty detector
    - Select the IAM role AWS recommends (auto-created)

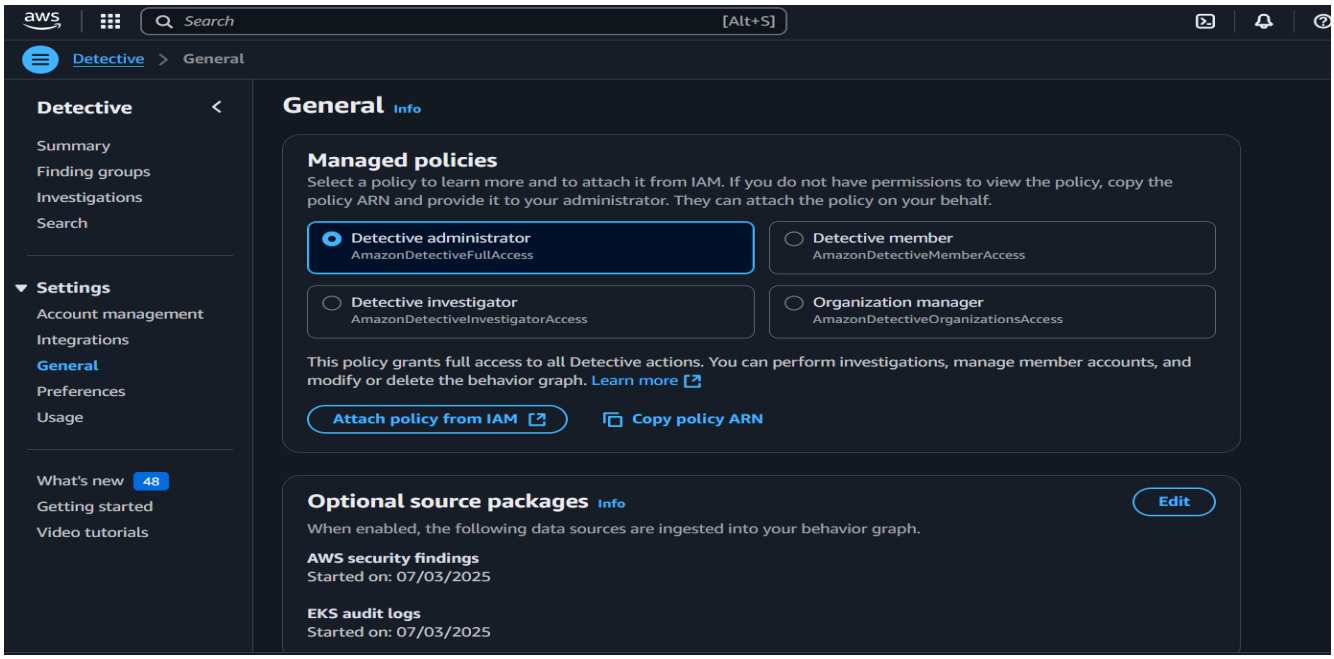

Fig 4.13 : Enabling AWS Detective for Graph-Based Threat Investigation and Analysis

4. Click **Enable**

Detective will now:

- Start collecting metadata
- Let you investigate any **GuardDuty finding** with deep context

4.2.9 Amazon SNS (Alert Notification)

Amazon Simple Notification Service (SNS) is a fully managed messaging service used to send real-time alerts to subscribers. In this project, SNS was integrated to deliver security notifications via email whenever threats were detected. It played a key role in the response pipeline by ensuring timely communication to administrators for rapid awareness and action.

STEP 9 : Create SNS Topic (Email Alerts)
Action:

1. Go to Amazon SNS Console
2. Click “Topics” → Create topic
3. Type: Standard
4. Name: threat-alerts-darshan
5. Click Create topic

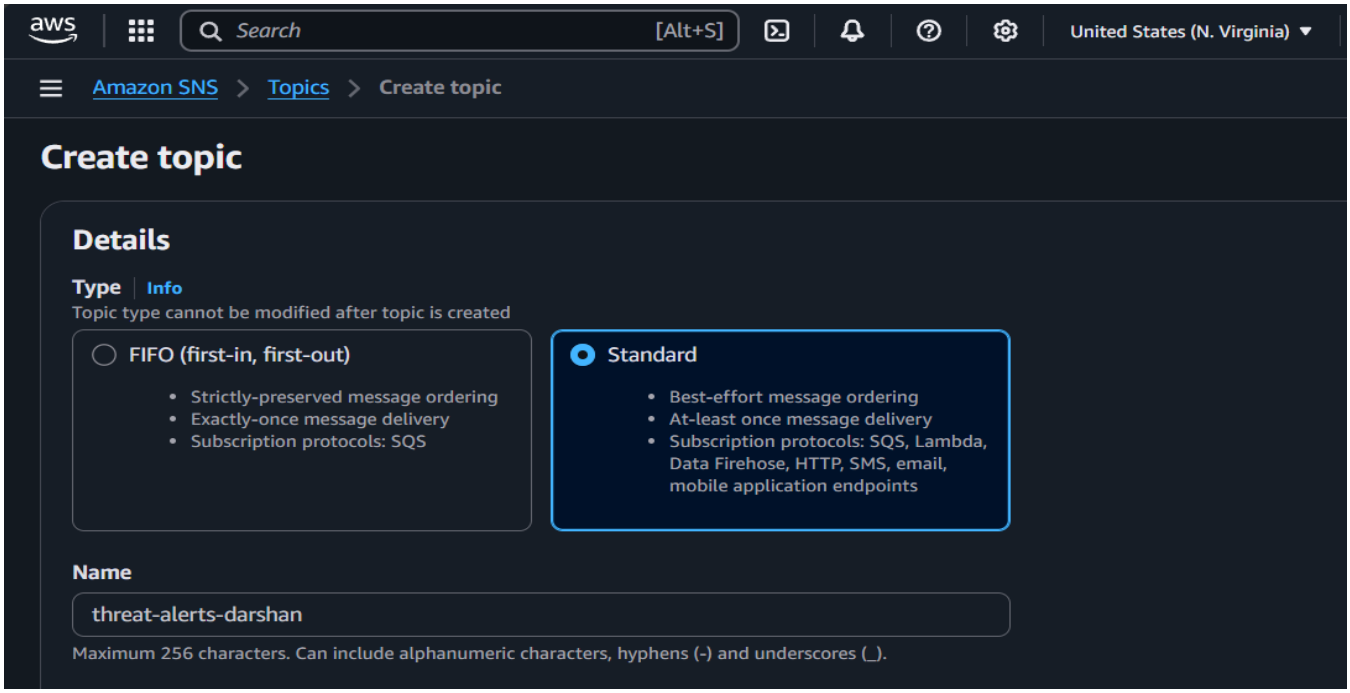

Fig 4.14 : **Amazon SNS Topic Creation for Real-Time Security Alert Notifications**

Add Subscription:

1. Click the topic you just created
2. Go to “Subscriptions” → Create subscription

3. Protocol: Email
4. Endpoint: Your email address (use one you can check now)
5. Click Create subscription
6. Then, go to your email inbox and confirm the subscription

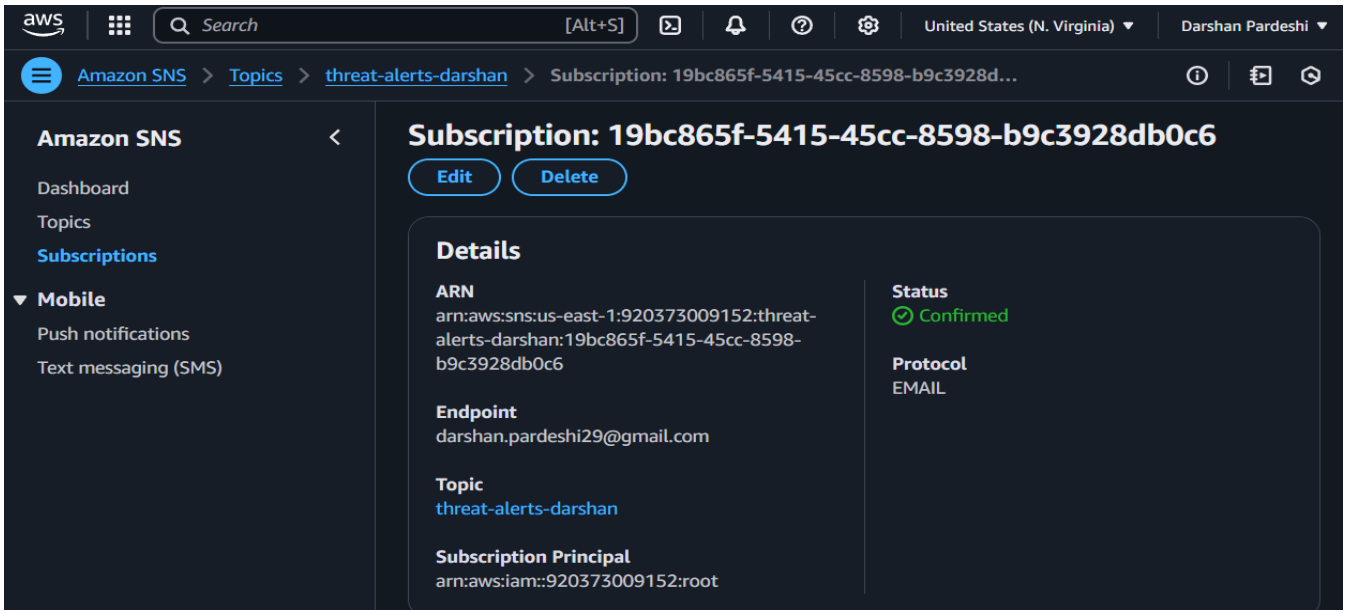

Fig 4.15 : Email Subscription Setup and Confirmation for SNS Alerts

4.2.10 Amazon EventBridge (Event Routing)

Amazon EventBridge is a serverless event bus that routes events between AWS services based on defined rules. In this project, it was used to detect security findings from GuardDuty, Macie, and Config, and trigger automated responses via Lambda. EventBridge was central to the automation pipeline, ensuring real-time, rule-based routing of threat events without manual intervention.

STEP 10 : EventBridge Rule (Trigger on GuardDuty Finding)
This rule listens for new GuardDuty findings and immediately sends an alert to your **SNS topic** (threat-alerts-darshan).
Action:

1. Go to **EventBridge Console**
2. Click **“Rules” → Create Rule**

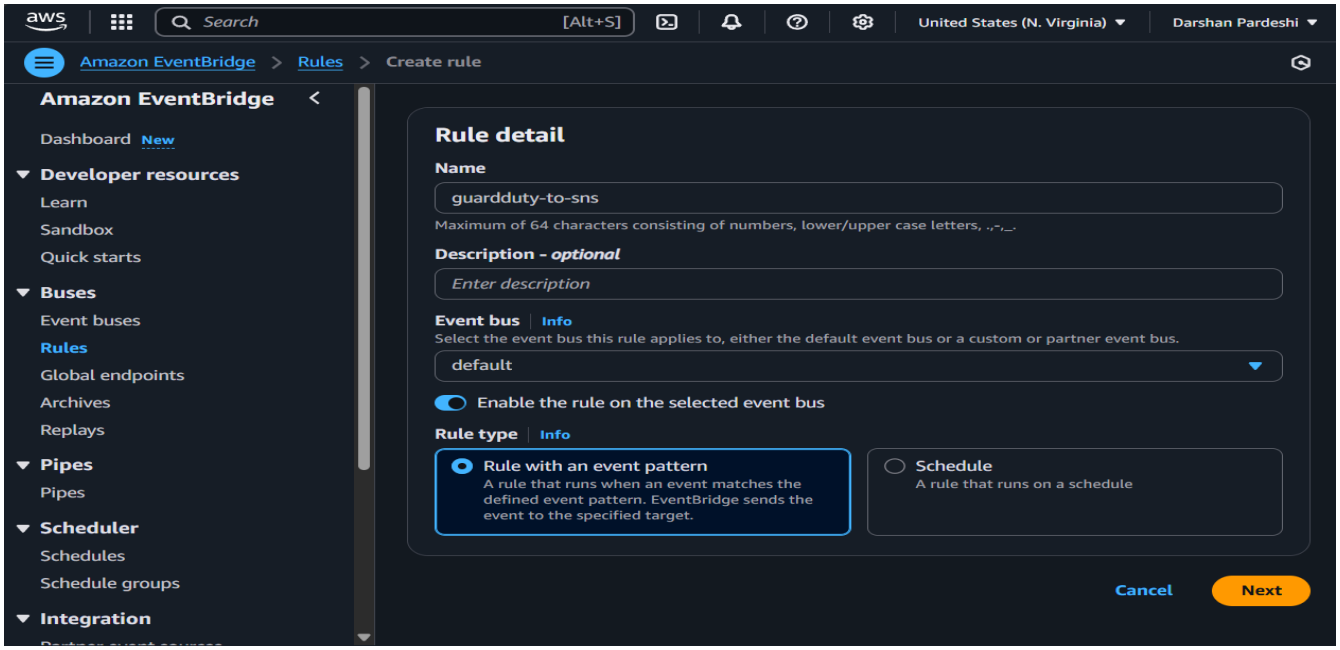

Fig 4.16 : EventBridge Rule Configuration for Automated Threat Alert Routing from GuardDuty to SNS

Define Event Pattern:

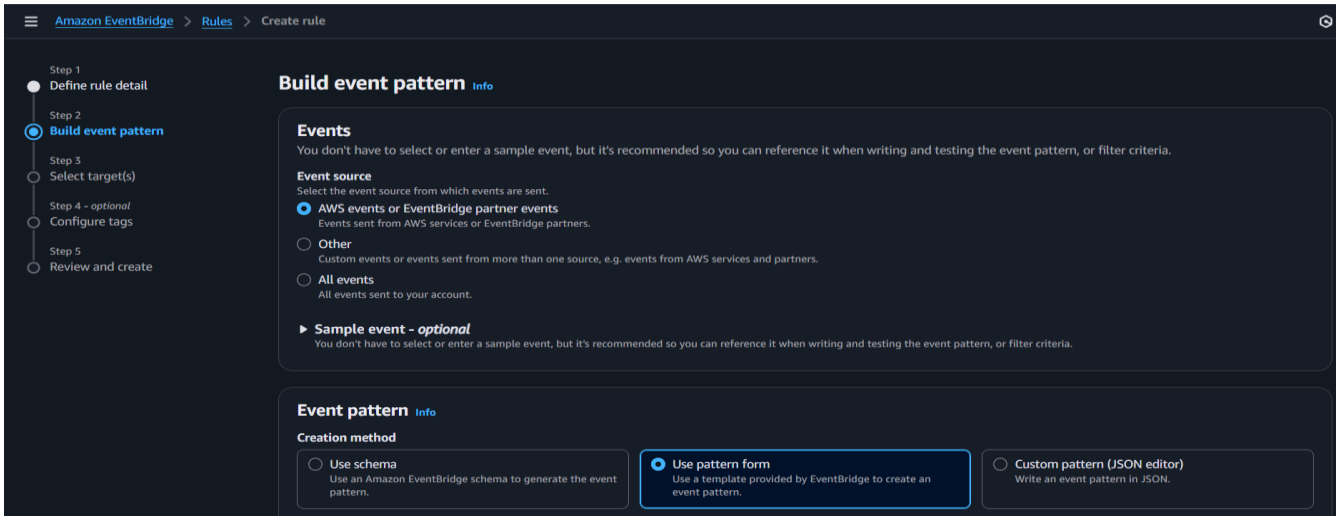

Fig 4.17 : Defining Event Pattern for GuardDuty in EventBridge

Choose:- AWS Services

- Service name: GuardDuty
- Event type: GuardDuty Finding

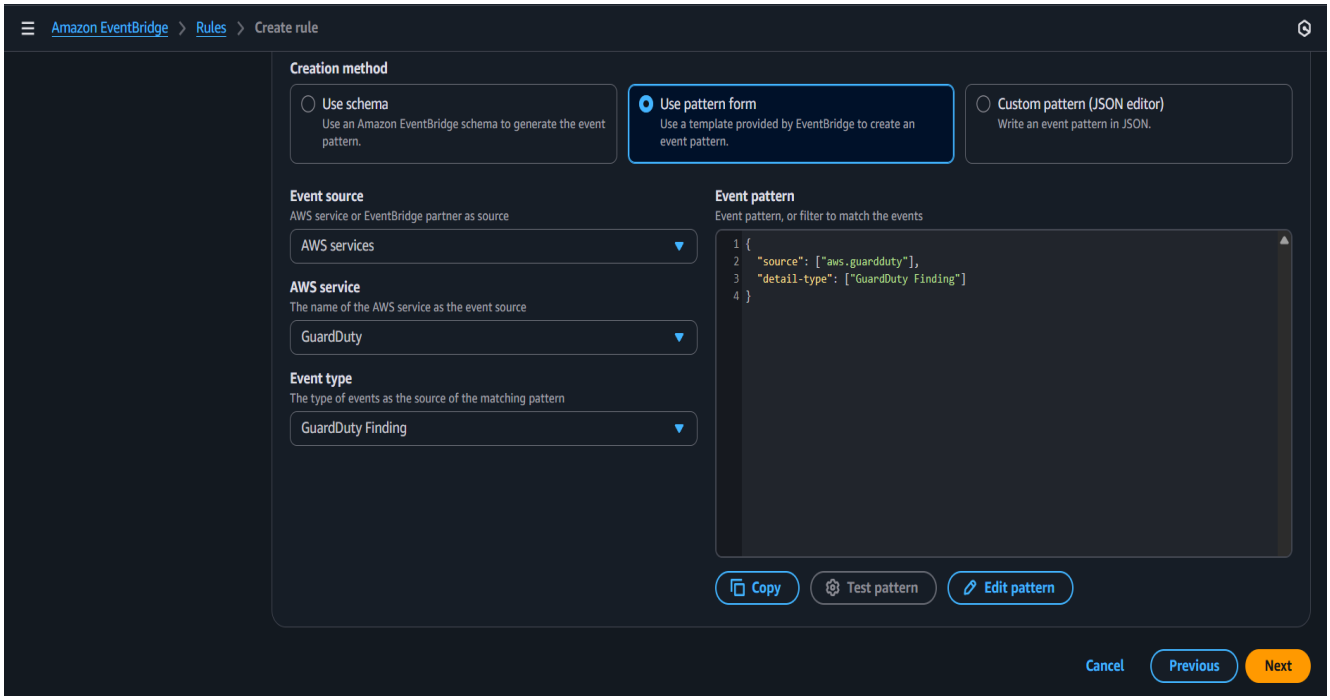

Fig 4.18 : Mapping GuardDuty Findings to SNS Topic via EventBridge

Target:

- **Target type**: SNS topic
- **Topic**: threat-alerts-darshan (select the one you made)

Click **Next → Create Rule**

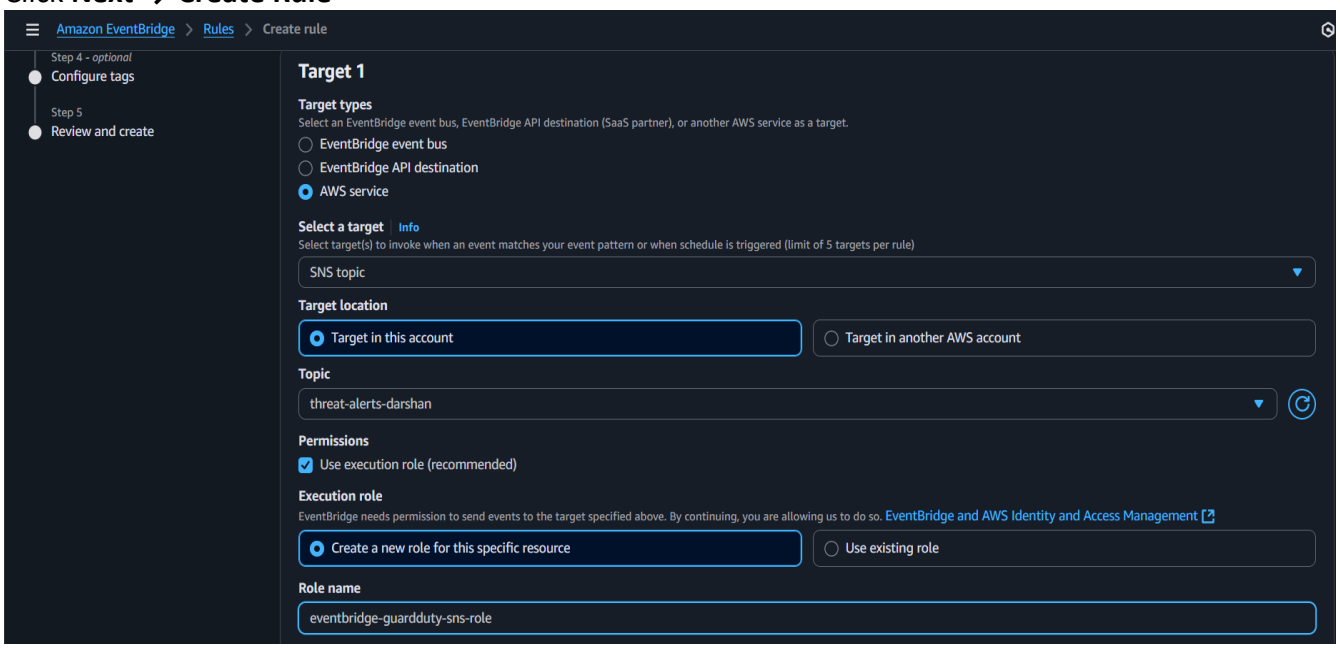

Fig 4.19 : Final EventBridge Rule Target Configuration with SNS and Execution Role

4.2.11 AWS Lambda (Alert Processing)

AWS Lambda is a serverless compute service that runs code in response to events without managing servers. In this project, Lambda was used to process threat findings

received from EventBridge, format the alert data, and forward it to SNS. It served as the execution engine of the automation layer, enabling fast, scalable, and hands-free alert processing within the threat response workflow.

STEP 11 : Create Lambda Function for GuardDuty Findings

Action 1: Create Lambda Function

1. Go to **Lambda Console** → Click **Create function**
2. Choose **Author from scratch**
3. Name: guardduty-alert-logger
4. Runtime: **Python 3.12**

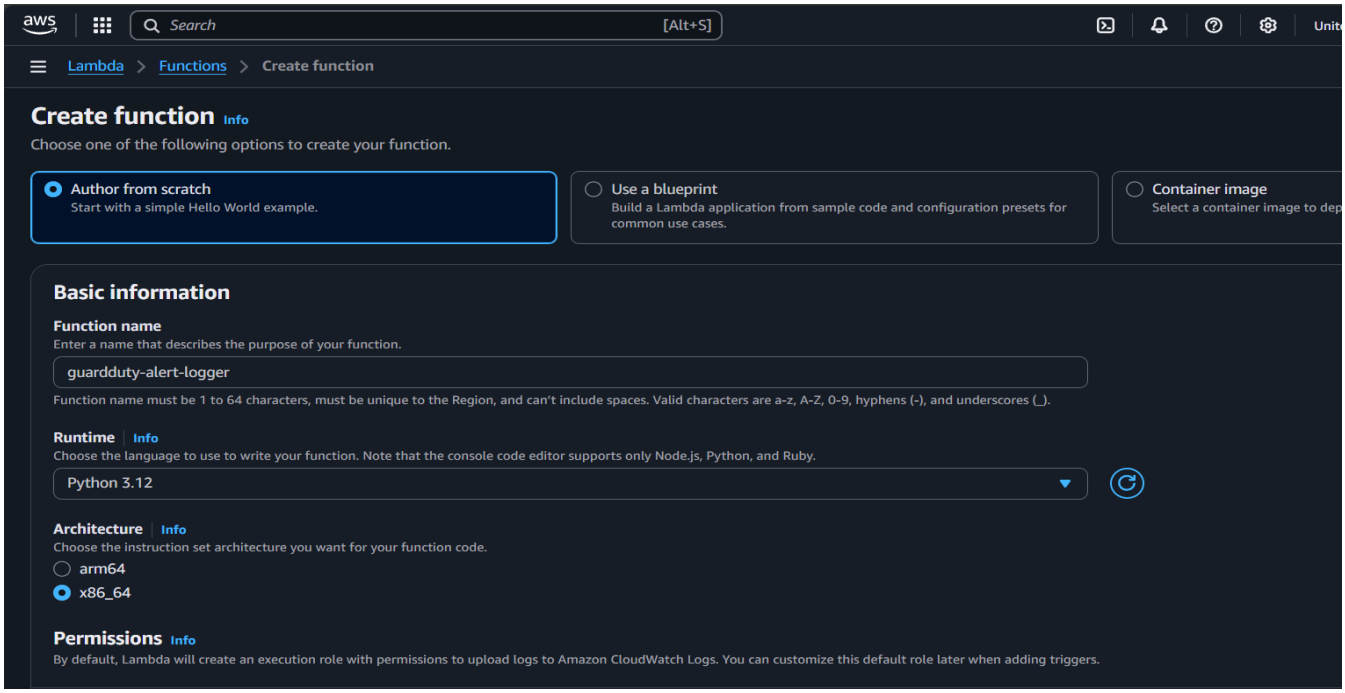

Fig 4.20 : Creating AWS Lambda Function for Processing GuardDuty Security Findings

5. Role:
    - Choose: **Create a new role with basic Lambda permissions**

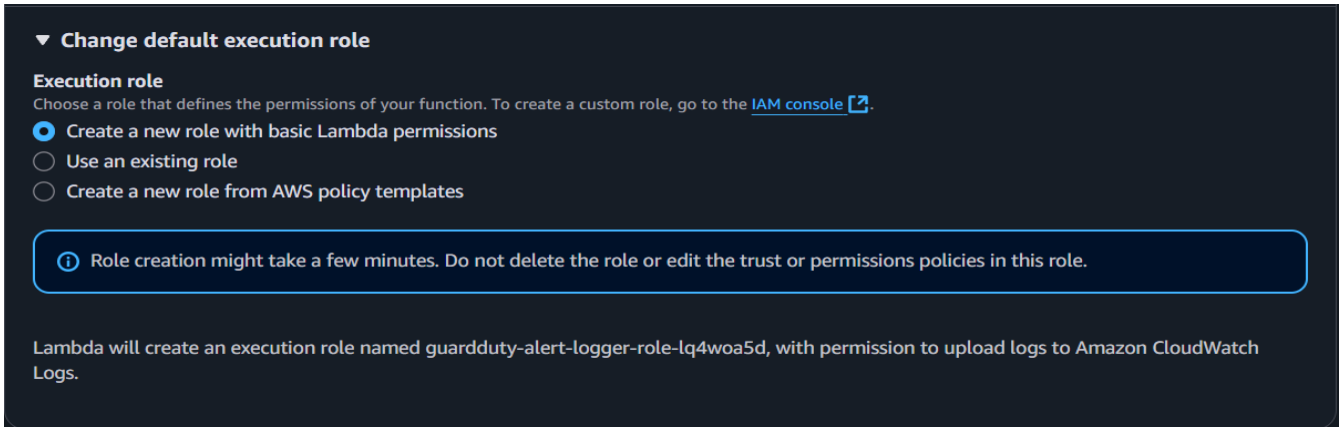

Fig 4.21 : Assigning Execution Role with Basic Lambda Permissions

6. Click **Create function**

Action 2: Add SNS Publish Permissions

1. After function is created, click **“Configuration”** → **Permissions**
2. Click the **IAM role name**
3. In IAM Console:

   - Go to IAM Console:
   - Click : **Roles**
   - Search for : guardduty-alert-logger-role
   (or whatever name your Lambda function used)
   - Click the role → Click **“Add permissions”** → **“Attach policies”**
   - Search and attach : AmazonSNSFullAccess

This allows Lambda to publish to your threat-alerts-darshan topic.

Action 3: Add Python Code

Go back to **Code tab**, and **paste the code below:**

```
import json
import boto3

sns = boto3.client('sns')
SNS_TOPIC_ARN = 'arn:aws:sns:REGION:ACCOUNT_ID:threat-alerts-darshan'  # Replace with your actual SNS ARN

def lambda_handler(event, context):
    detail = event.get("detail", {})
    finding_type = detail.get("type", "Unknown")
    description = detail.get("description", "No description available")
    severity = detail.get("severity", "Unknown")
    region = detail.get("region", "Unknown")
    account = detail.get("accountId", "Unknown")

    message = f"""
🚨 GuardDuty Finding 🚨
Type: {finding_type}
Severity: {severity}
Region: {region}
Account: {account}
Description: {description}
"""

    sns.publish(
        TopicArn=SNS_TOPIC_ARN,
```

```
    Subject="🚨 GuardDuty Alert",
    Message=message
  )

  return {
    'statusCode': 200,
    'body': json.dumps('Alert sent to SNS')
  }
```

Replace:-

- REGION with your region (e.g., us-east-1)
- ACCOUNT_ID with your AWS account ID
  (You can find both in the top-right corner of AWS console)\

Action 4: Deploy and Test

1. Click **Deploy**
2. Don't test manually — it will trigger automatically from EventBridge

Action 5: Update EventBridge Rule Target

Now let's update the EventBridge rule to use Lambda **instead of SNS directly**.

1. Go to **EventBridge → Rules → guardduty-to-sns**
2. Edit → Change target:
   - **Target type**: Lambda function
   - **Function**: guardduty-alert-logger

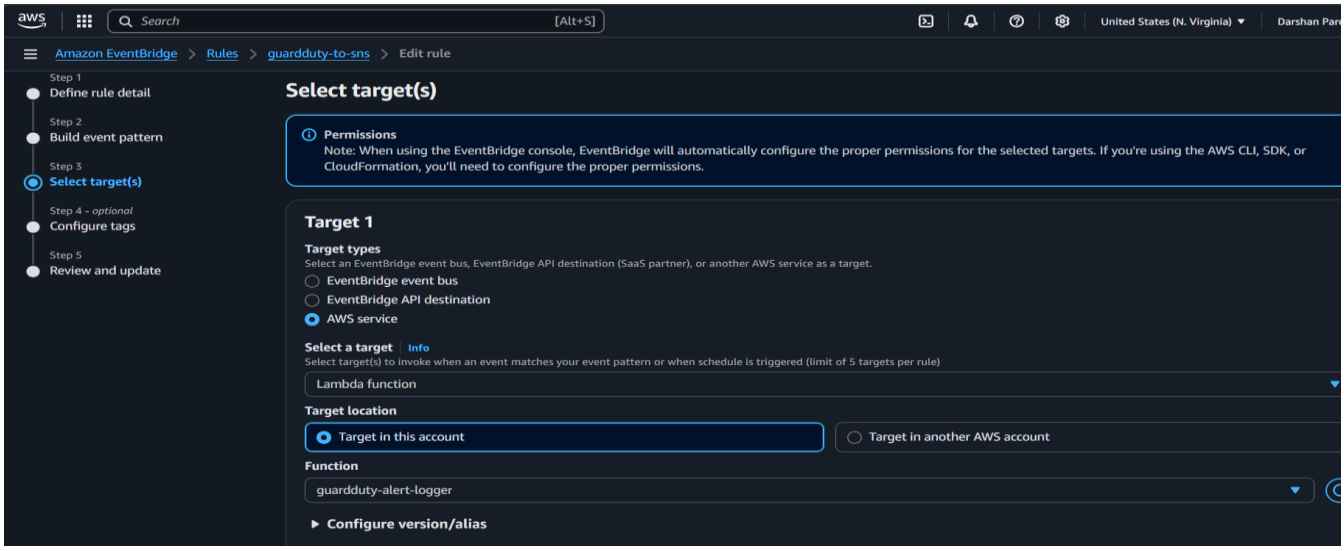

Fig 4.22 : Updating EventBridge Rule Target to AWS Lambda

4.2.12 Amazon Athena (Log Querying)

Amazon Athena is a serverless query service that enables analysis of data stored in Amazon S3 using standard SQL. In this project, Athena was used to query CloudTrail logs and VPC Flow Logs for deeper investigation of API activity and network behavior. It added a powerful forensic layer to the pipeline, allowing quick, cost-effective threat hunting without provisioning any infrastructure.

STEP 12: Set Up Athena for Log Queries
Athena lets you **run SQL queries on S3 logs** like:

- CloudTrail
- VPC Flow Logs
- S3 access logs (if enabled)
  Without needing a server or database.

Action 1: Configure Athena

1. Go to **Amazon Athena Console**
2. Click **"Get started"**

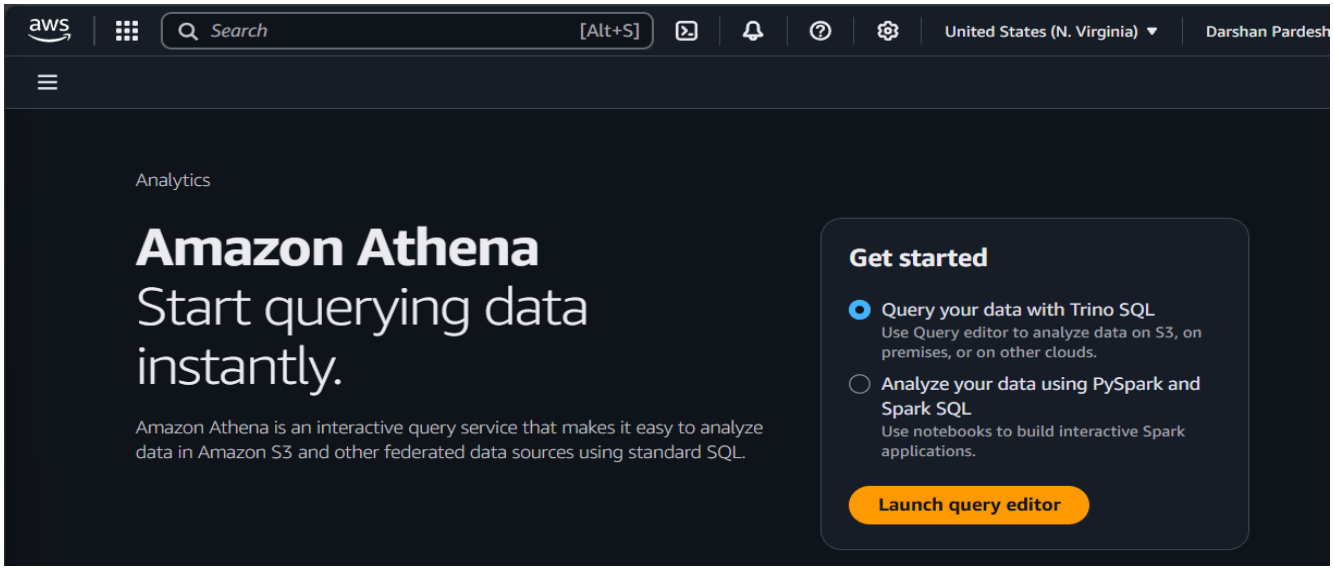

Fig 4.23 : Setting Up Amazon Athena for Serverless Log Analysis and Threat Hunting

3. For Query result location → click **"Browse S3"**
   - Select your bucket threat-logs-darshan

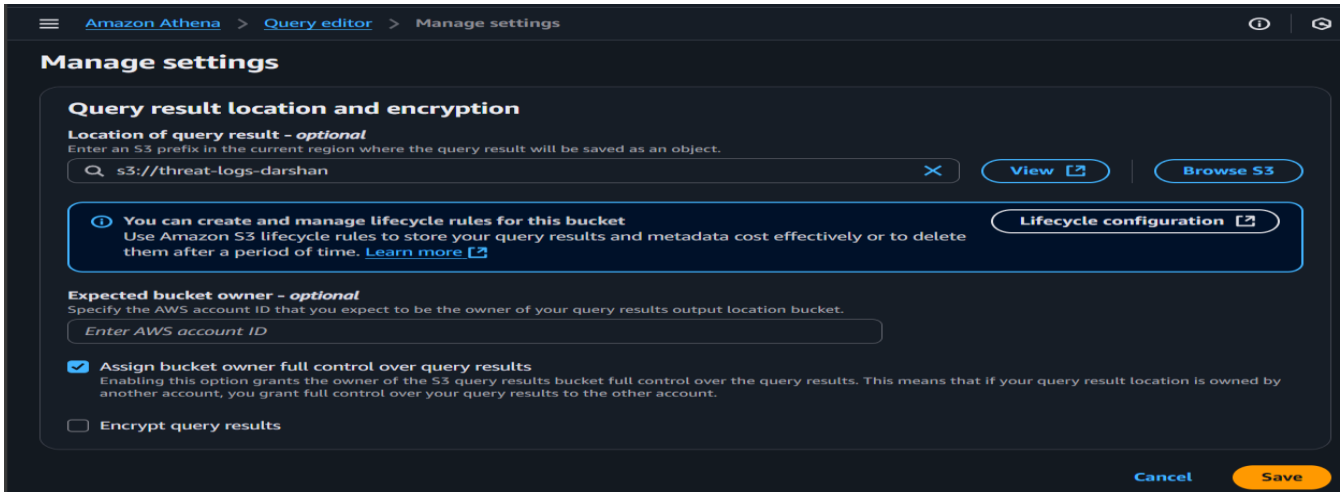

Fig 4.24 : Selecting S3 Bucket as Query Result Destination

Create folder: /athena-results/

How to Create /athena-results/ Folder in S3

Step-by-Step:

1. Go to the **S3 Console**
2. Click your bucket : threat-logs-darshan
3. Click **"Create folder"** (top-right corner)

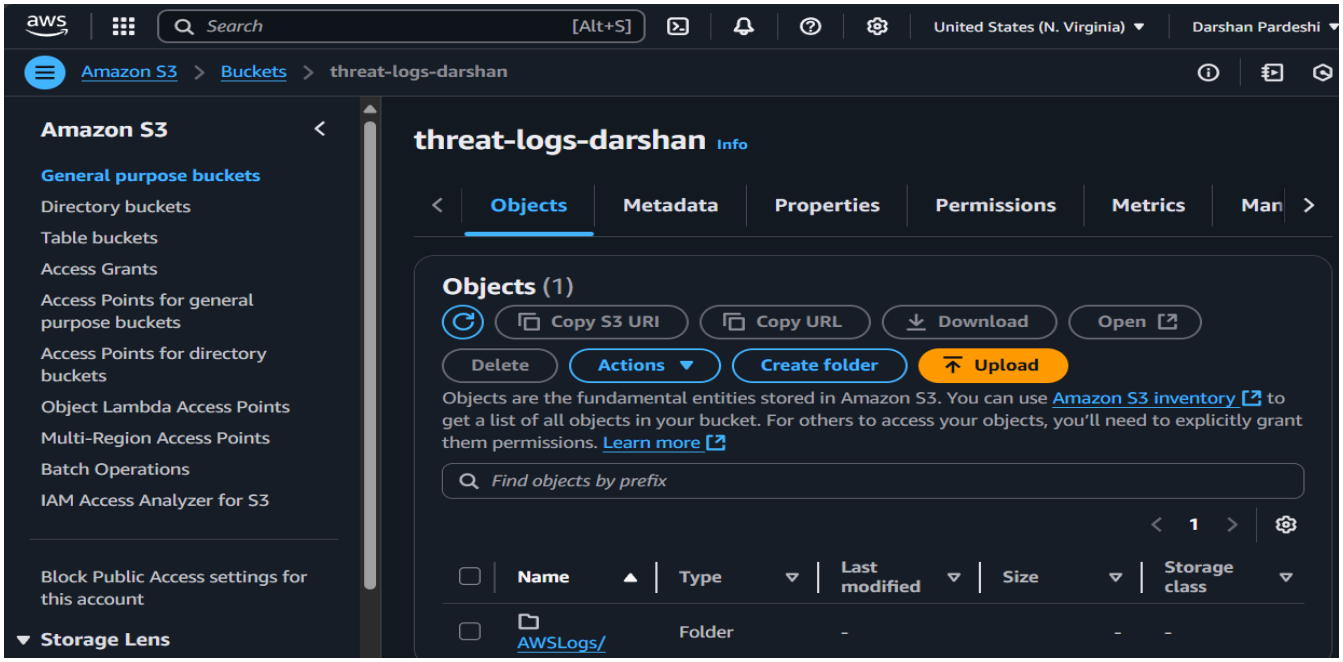

Fig 4.25 : Creating /athena-results/ Folder in S3 for Athena Query Output

4. In the popup:
   - **Folder name** : athena-results
5. Leave encryption & ACLs as default (no need to change)

Click Create folder .

Come back again to Athena

- Final location will look like:

Save.

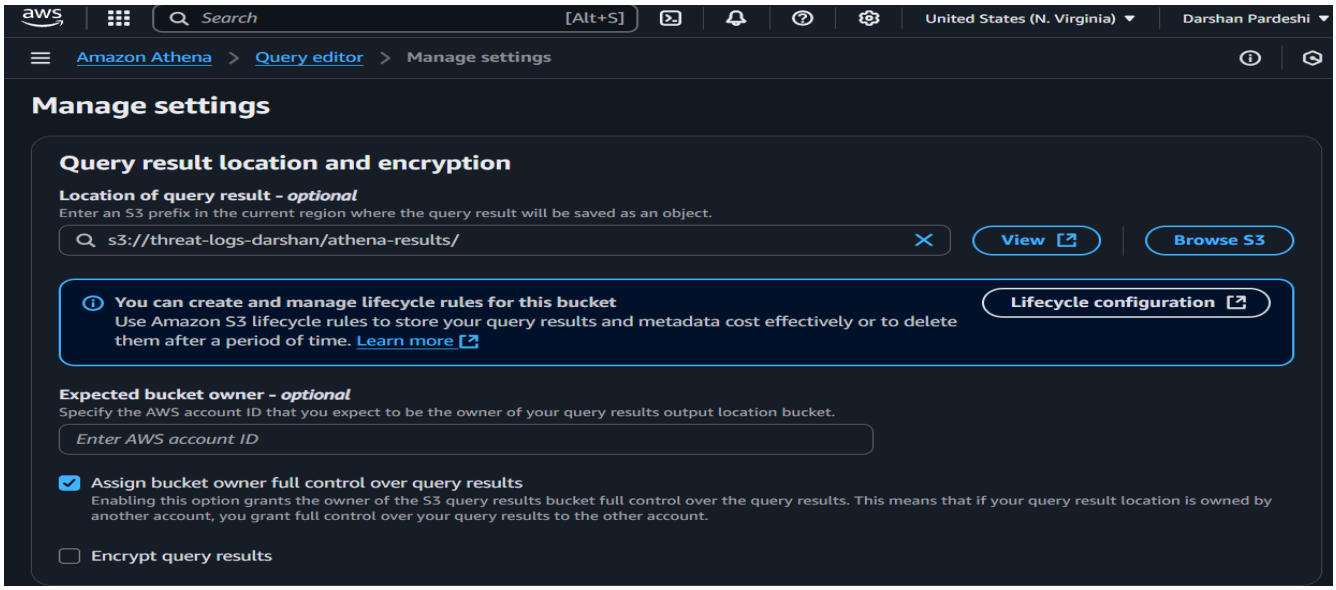

Fig 4.26 : Finalizing Athena Output Location with /athena-results/ S3 Path

Action 2: Create Database

Go to the Athena Query Editor and run:

CREATE DATABASE threat_logs;

1. In the left sidebar of the Athena Query Editor, find:
    - Catalog : leave it as AwsDataCatalog
    - Database: click the dropdown next to default
2. Select : threat_logs
3. If threat_logs is not showing, click the refresh icon next to the dropdown.

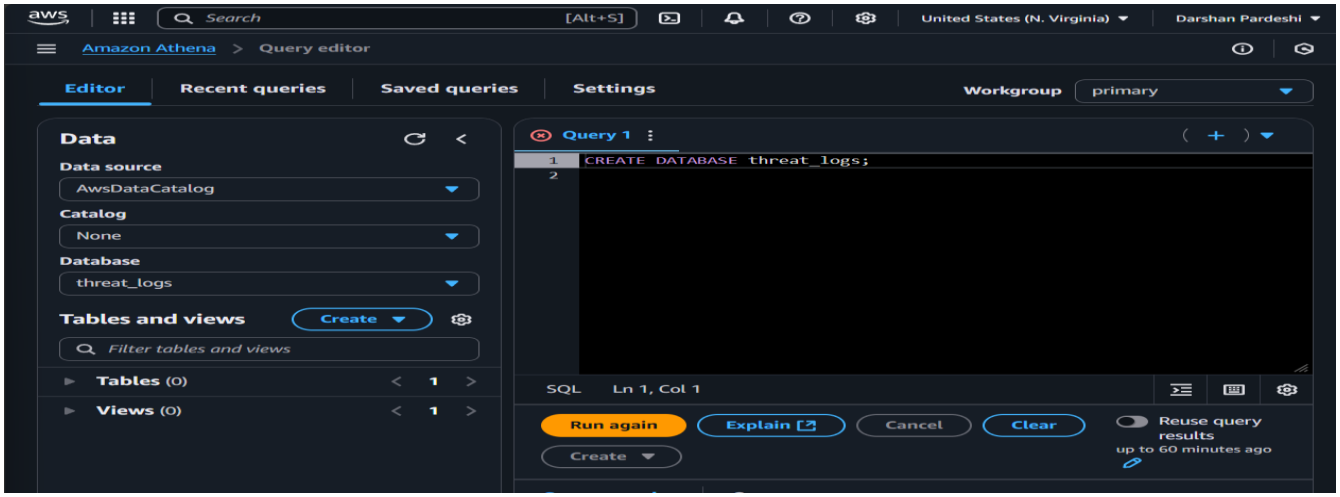

Fig 4.27 : Creating and Verifying Athena Database for Log Analytics

Action 3: Create Table for VPC Flow Logs
Assuming your VPC Flow Logs are stored in **CloudWatch** , not S3. So skip this for now unless you:

- Export VPC flow logs to S3 (we can do later if needed)
- Want to query S3 access logs or CloudTrail

For now, your **CloudTrail logs are already going to S3**. So run this to create a table for CloudTrail:

Action 4: Create CloudTrail Table

Step-by-Step:

1. **Go to Athena > Query Editor**
2. **In Table And Views :-** Create with SQL

Select: Create with SQL
Then choose : CREATE TABLE

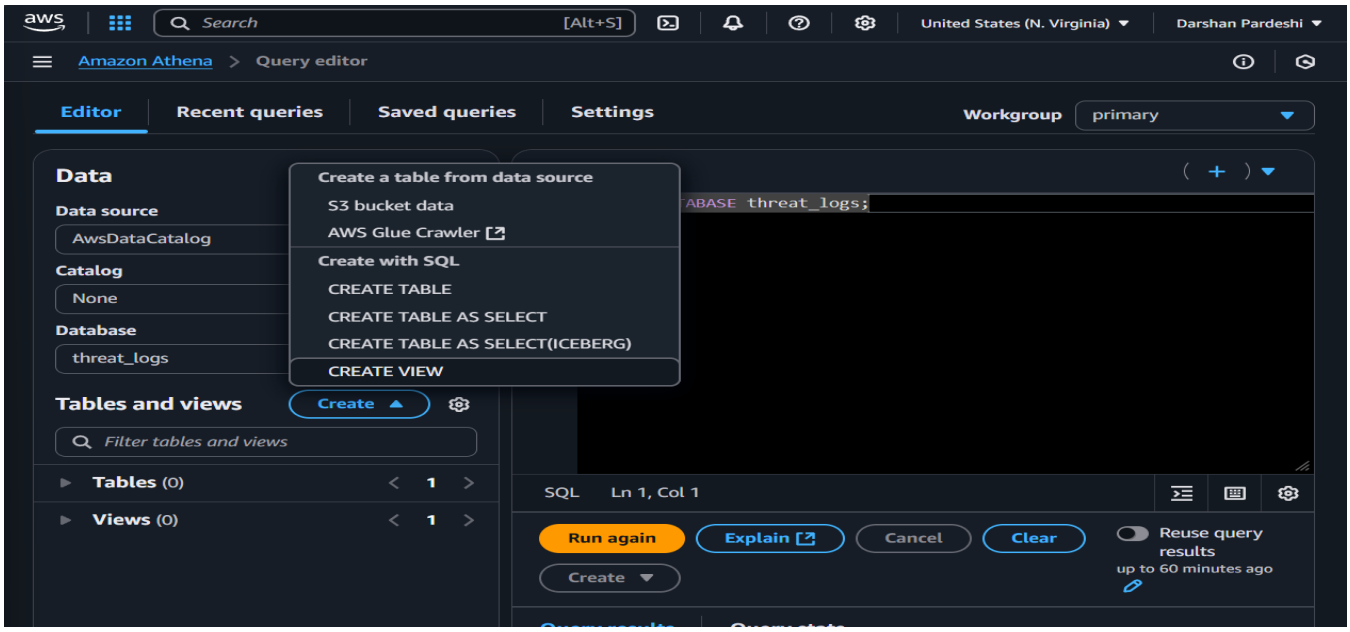

Fig 4.28 : Creating Athena Table for CloudTrail Logs Using SQL Editor

3. **This will open a SQL editor where you paste the full code :**

```
CREATE EXTERNAL TABLE IF NOT EXISTS threat_logs.cloudtrail_logs (
  eventVersion STRING,
  userIdentity STRUCT<
    type:STRING,
    principalId:STRING,
    arn:STRING,
    accountId:STRING,
    invokedBy:STRING,
    accessKeyId:STRING,
    userName:STRING,
```

```
   sessionContext:STRUCT<
     attributes:STRUCT<
       mfaAuthenticated:STRING,
       creationDate:STRING>,
     sessionIssuer:STRUCT<
       type:STRING,
       principalId:STRING,
       arn:STRING,
       accountId:STRING,
       userName:STRING>>>,
  eventTime STRING,
  eventSource STRING,
  eventName STRING,
  awsRegion STRING,
  sourceIPAddress STRING,
  userAgent STRING,
  errorCode STRING,
  errorMessage STRING,
  requestParameters STRING,
  responseElements STRING,
  additionalEventData STRING,
  requestId STRING,
  eventId STRING,
  resources ARRAY<STRUCT<ARN:STRING, accountId:STRING, type:STRING>>,
  eventType STRING,
  apiVersion STRING,
  readOnly STRING,
  recipientAccountId STRING,
  serviceEventDetails STRING,
  sharedEventID STRING,
  vpcEndpointId STRING
)
PARTITIONED BY (region STRING, year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://threat-logs-darshan/AWSLogs/920373009152/CloudTrail/'
TBLPROPERTIES ('classification'='cloudtrail');
```

Trigger the Lambda Manually (Recommended)

1. Go to **Lambda Console**
2. Select guardduty-alert-logger
3. Click **"Test"**

4. Configure a simple test event:

```
{
  "detail": {
    "type": "Recon:EC2/PortProbeUnprotectedPort",
    "description": "Test GuardDuty finding",
    "severity": 5,
    "region": "us-east-1",
    "accountId": "123456789012"
  }
}
```

5. Click **Test**

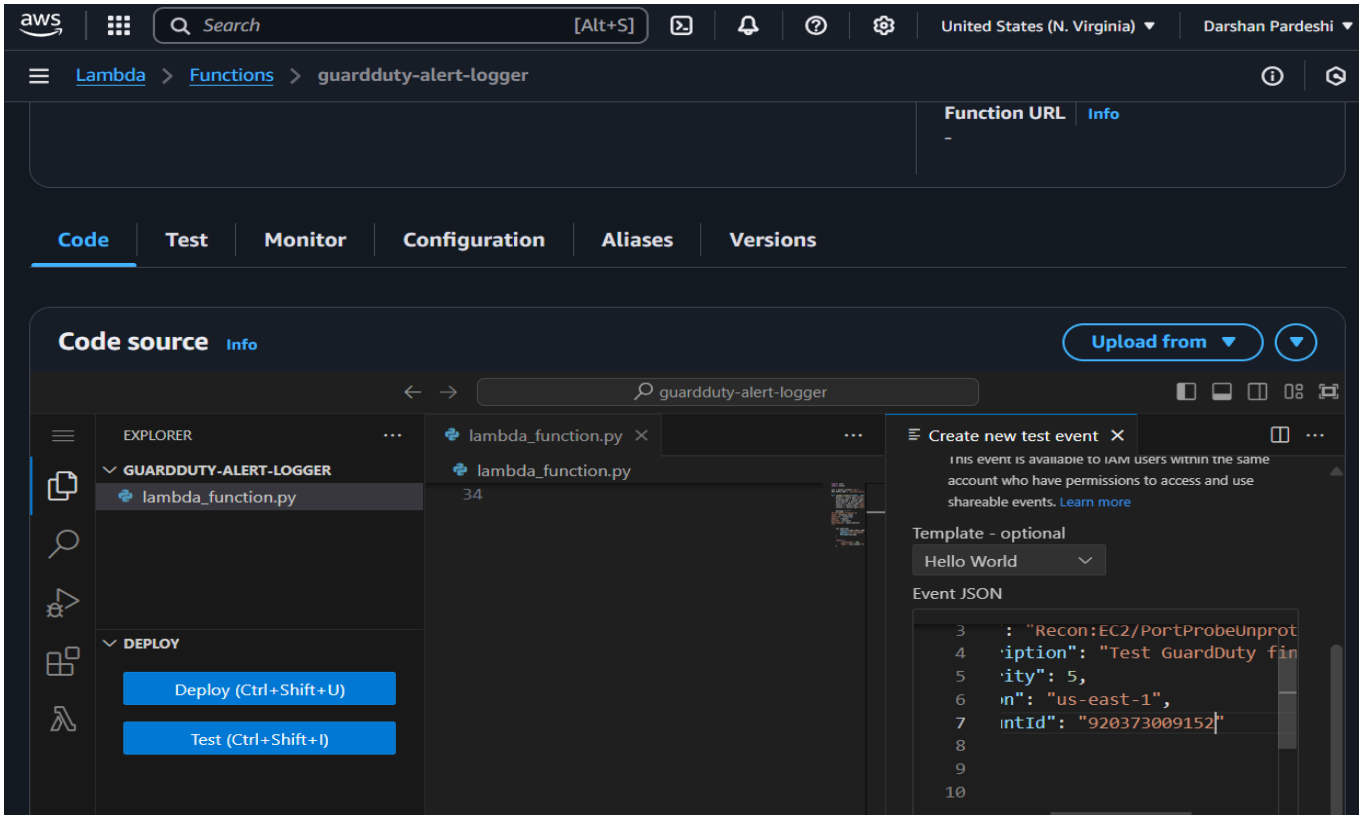

Fig 4.29 : Manually Testing Lambda Function with Simulated GuardDuty Finding

This will:

- Invoke the function
- Trigger an SNS alert
- Make the metric appear in CloudWatch

4.2.13 Amazon CloudWatch (Metrics & Dashboards)

Amazon CloudWatch is a monitoring and observability service that collects logs, metrics, and events from AWS resources in real time. In this project, it was used to monitor Lambda executions, VPC Flow Logs, and alert delivery status. CloudWatch added operational

visibility to the pipeline through customized dashboards, helping validate system performance and detect anomalies during threat simulations.

Step 13 : **Setup CloudWatch Dashboards (Optional but Valuable)**
This gives you a **real-time visual view** of:

- Number of GuardDuty findings
- Lambda invocations
- SNS messages sent
- VPC traffic (if logs go to CloudWatch)

Action: Create Dashboard

1. Go to **CloudWatch Console**
2. Click **Dashboards → Create Dashboard**

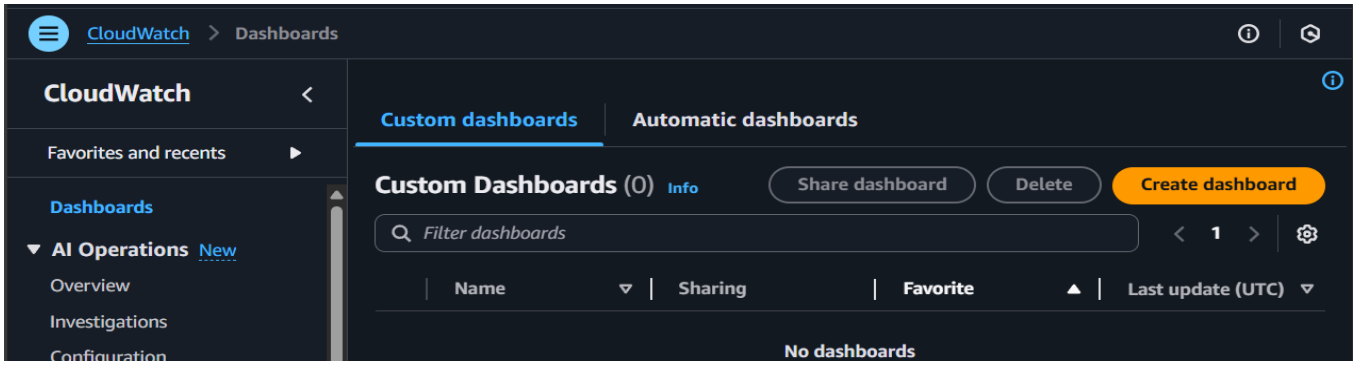

Fig 4.30 : Creating CloudWatch Dashboard for Real-Time Threat Visibility

3. Name: cloud-threat-dashboard
4. Add widgets:

1. Lambda Invocations
To track your function guardduty-alert-logger:

1. Go to CloudWatch → Dashboards → cloud-threat-dashboard
2. Click Add widget
3. Choose:
    - Data source: CloudWatch
    - Data type: Metrics
    - Widget type: Number

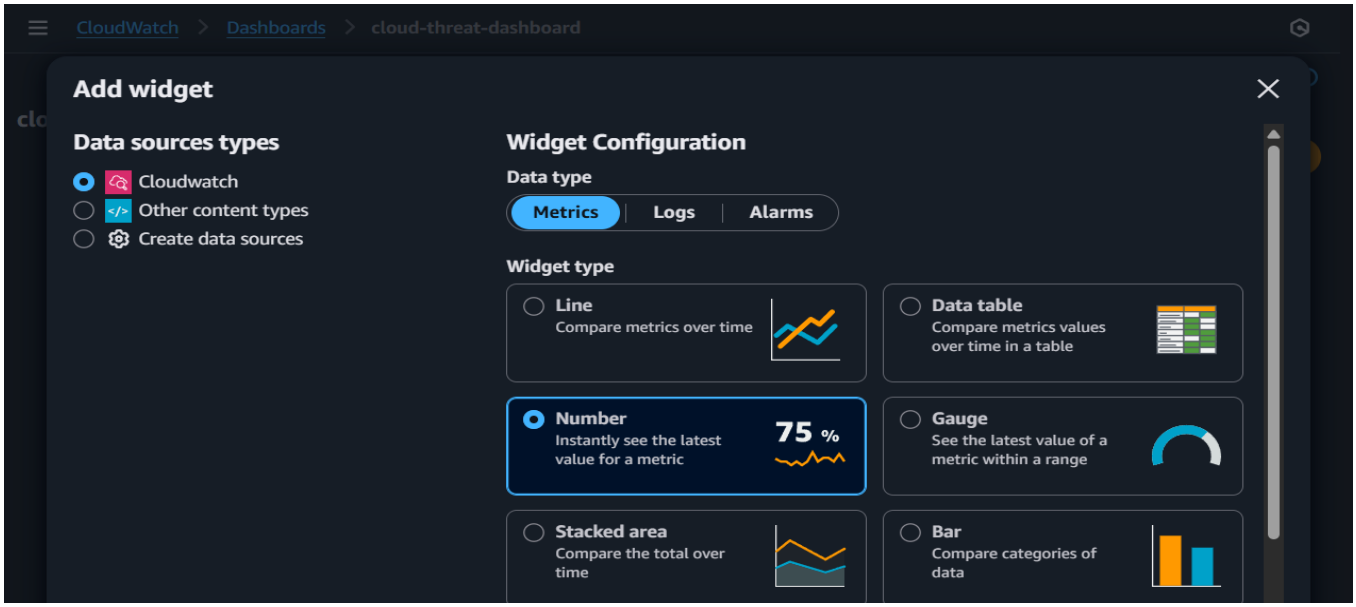

Fig 4.31 : Adding Lambda Invocation Metric to CloudWatch Dashboard

4. Click “Browse” metrics →
    - Namespace: AWS/Lambda
    - Choose By Function Name → Select guardduty-alert-logger → Metric: Invocations

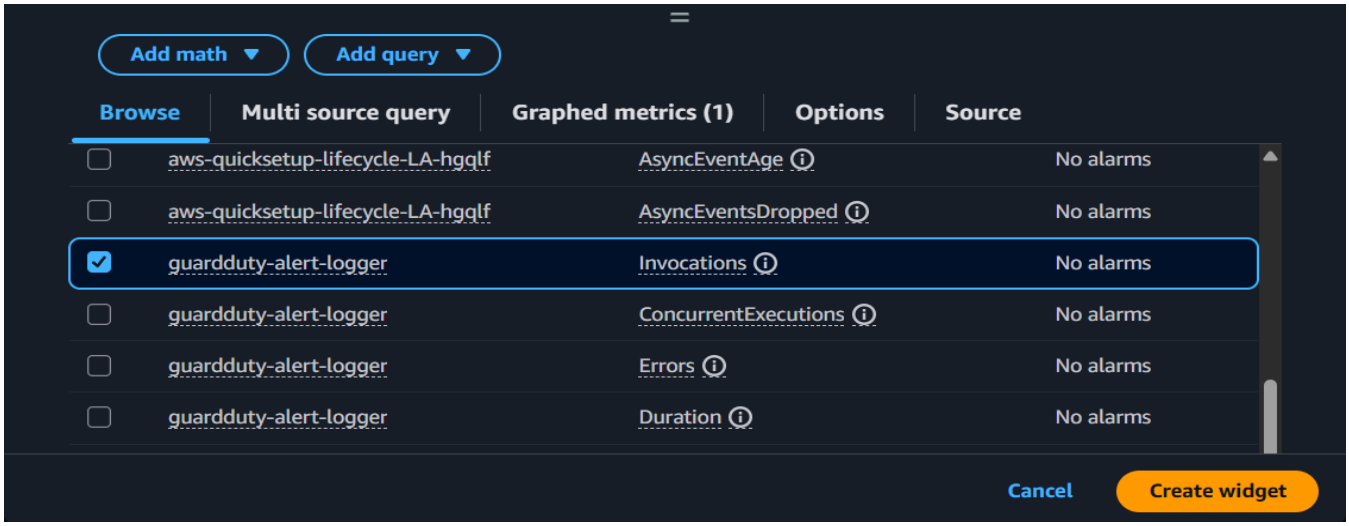

Fig 4.32 : Selecting Lambda Invocation Metric from AWS/Lambda Namespace

5. Click Create widget

2. SNS Published Messages

To track alerts published to your SNS topic:

1. Add another widget → Same flow
2. Choose:

- Widget Type: Number

3. Browse metrics →
    - Namespace: AWS/SNS
    - SNS > Topic Metrics → Select threat-alerts-darshan → Metric: NumberOfMessagesPublished

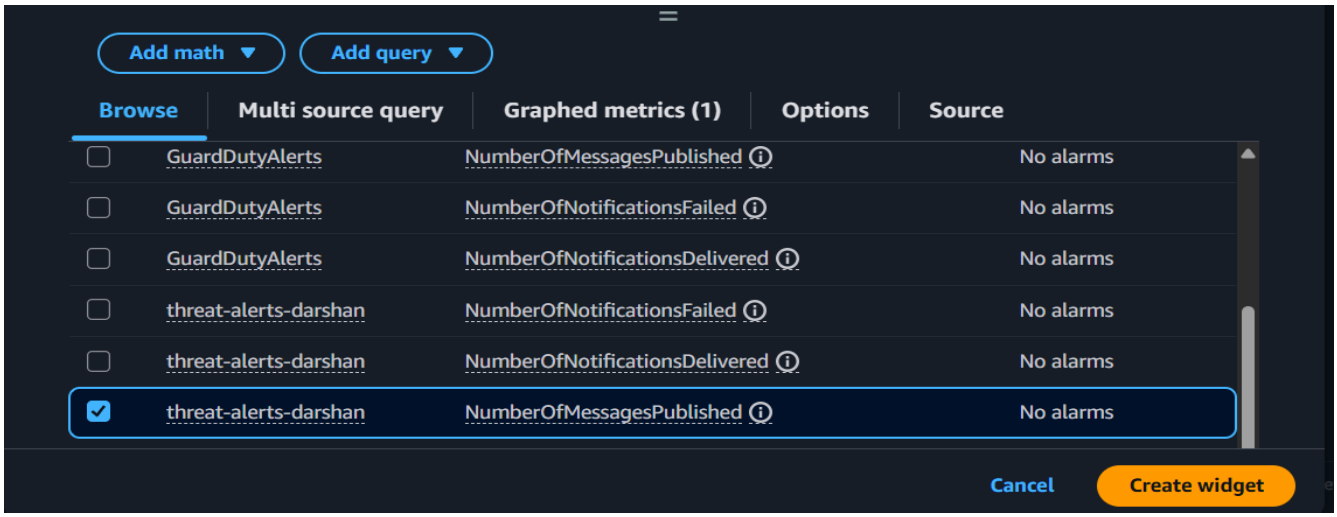

Fig 4.33 : Monitoring SNS Alert Delivery via Number of Messages Published Metric

4. Click Create widget

3. GuardDuty Proxy Metric (Optional)
You can't see "Findings" directly — instead, we use a proxy (e.g., GetFindings API calls):

1. Add another widget
2. Choose:
    - Widget Type: Stacked Area
3. Browse:
    - Namespace: AWS/Usage
    - Resource: GuardDuty
    - Operation: GetFindings

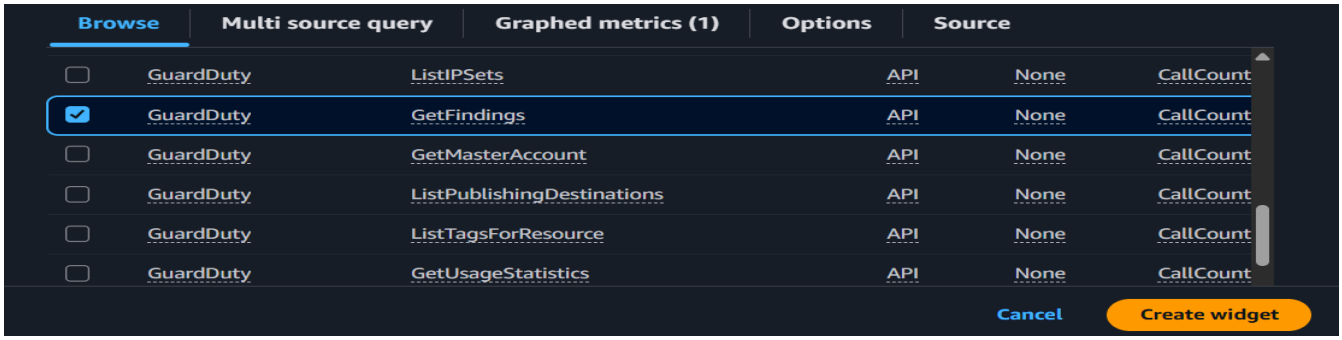

Fig 4.34 : Visualizing GuardDuty Activity via GetFindings API Proxy Metric

4. Click Create widget.

4.3 Serverless Automation Pipeline

The core of this project's real-time response capability lies in its fully serverless automation pipeline, designed to detect, process, and alert on cloud-based threats with minimal latency and zero infrastructure overhead. This pipeline ensures that once a threat is identified, appropriate action is taken automatically without human intervention delivering speed, consistency, and scalability.

The workflow begins when detection services like Amazon GuardDuty, Amazon Macie, or AWS Config generate a security finding. These findings are published as structured events and routed through Amazon EventBridge, which evaluates them against custom rules configured to detect specific types of threats or misconfigurations.

On rule match, AWS Lambda is triggered to process the finding. The Lambda function parses key information such as resource type, threat name, timestamp, and severity level. It then formats this data into a structured, human-readable alert message. This message is immediately published to an Amazon SNS topic, which then pushes notifications to designated subscribers via email. To ensure visibility and reliability, Amazon CloudWatch captures both execution logs and performance metrics throughout the pipeline, enabling monitoring of Lambda invocations, event processing latency, and alert delivery status. This observability ensures the pipeline remains healthy, auditable, and responsive under varying threat loads. The architecture is inherently scalable, event-driven, and fully aligned with the principles of Zero Trust and cloud-native security. It enables proactive defense by converting detection signals into actionable alerts automatically, instantly, and securely.

4.4 Log Analysis and Visualization

Log analysis and visualization played a pivotal role in validating the system's detection capabilities and providing actionable insights into cloud activity. By leveraging both structured log querying and real-time monitoring, the project enabled precise threat verification and enhanced behavioral visibility across the environment. All operational and security logs including AWS CloudTrail events, VPC Flow Logs, and Lambda execution outputs were centrally collected in Amazon S3 and CloudWatch Log Groups. These logs

were queried using Amazon Athena, allowing SQL-based investigations directly on S3 data without the need for persistent compute resources. Through Athena, the system efficiently identified critical events such as unauthorized IAM access, suspicious API calls, and unusual port-level traffic patterns.

For real-time observability, Amazon CloudWatch was used to visualize metrics and trends across Lambda, VPC, and SNS services. Custom dashboards were created to track function invocations, log processing latency, and alert delivery success ensuring the entire automation pipeline remained operational, responsive, and transparent. This layered approach combined forensic querying with live monitoring, empowering the system to move beyond passive detection. It enabled intelligent analysis, continuous visibility, and faster incident response transforming the architecture into a proactive and self-observing security solution.

4.5 Real-Time Attack Simulation

To validate the effectiveness and responsiveness of the detection engine, controlled real-time attack simulations were conducted in the AWS environment. These simulations emulated common cloud security threats such as unauthorized network probing and sensitive data exposure. The goal was to ensure that the detection, automation, and alerting pipeline performed seamlessly end-to-end, without human intervention.

Attack 1: Simulate Port Scan on Open Security Group
Action:

1. Go to **EC2 → Security Groups**

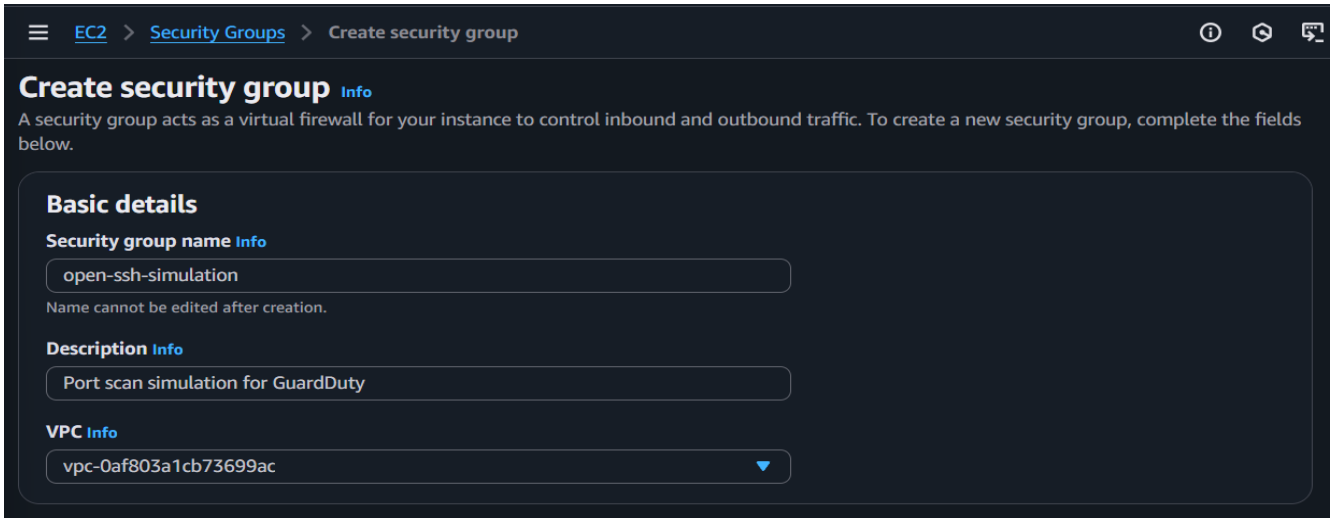

Fig 4.35 : Simulating Port Scan Exposure via Misconfigured EC2 Security Group Rules

2. Create or choose a security group with : Add Inbound Rule

(I) **SSH**

- **Protocol:** TCP (auto-filled)
- **Port Range:** 22

- **Source:** Anywhere (0.0.0.0/0)

(II) **HTTP**

- **Protocol:** TCP
- **Port Range:** 80
- **Source:** Anywhere (0.0.0.0/0)

(III) **HTTPS**

- **Protocol:** TCP
- **Port Range:** 443
- **Source:** Anywhere (0.0.0.0/0)

(IV) **Custom TCP Rule (Application Port)**

- **Protocol:** TCP
- **Port Range:** 8080
- **Source:** Anywhere (0.0.0.0/0)

(V) **Custom TCP Rule (FTP Port)**

- **Protocol:** TCP
- **Port Range:** 21
- **Source:** Anywhere (0.0.0.0/0)

{Leave Outbound Rule as Default}

Temporarily launch a tiny EC2 instance:-

1. Go to **EC2 → Launch Instance**
2. Name: nmap-scan-target
3. AMI: **Amazon Linux 2**
4. Instance type: t2.micro (Free tier)
5. Attach your **open SSH security group**
6. Ensure **Auto-assign Public IP = Enabled**
7. Launch it (no need to SSH into it)

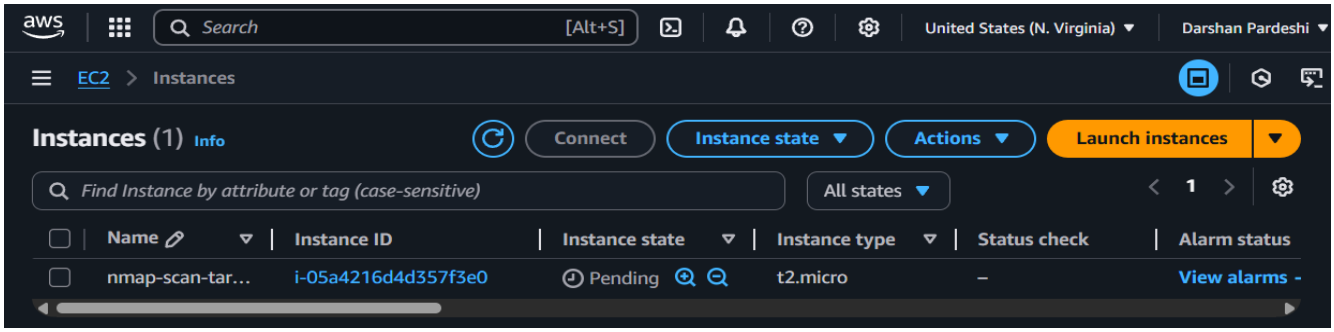

Fig 4.36 : Simulating External Port Scan Using Nmap on Public EC2 Instance

Simulating Port Scan Attack Using Nmap :-

Retrieve Public IP Address

- After launch, go to EC2 → Instances
- Note the IPv4 Public IP, e.g., 54.226.178.105

Install Nmap (on local machine) :-

- Download from: https://nmap.org/download.html
- Install and verify with : nmap -V

Execute Port Scan Commands :-

Open Command Prompt or PowerShell (as Administrator) and run:

Basic Port Scan:- nmap -Pn -p 22 54.226.178.105

- -A enables aggressive detection
- -T4 speeds up the scan
- -Pn skips ping checks

Attack 2: Public S3 Bucket with Sensitive File (Macie)

1. Go to **S3 → Your bucket**
2. Upload a .txt file with mock sensitive data:

Name: Darshan Pardeshi
Email: darshan@example.com
Password: 123456

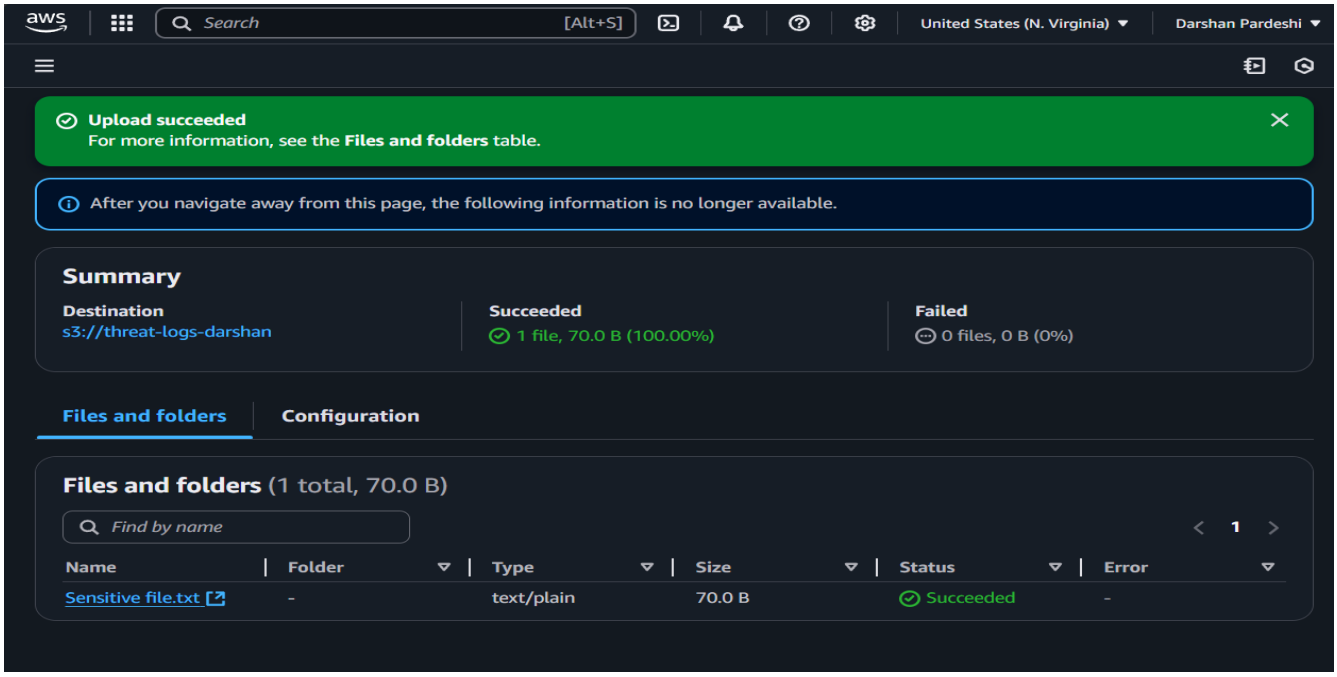

Fig 4.37 : Simulating Sensitive Data Exposure via Public S3 Bucket for Macie Detection

3. Make the file public

Permissions tab → Enable public access to object

Click "Edit" under "Bucket Policy"

Paste the following policy (replace <your-bucket-name> and <your-object-key>):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublicReadAccessToSpecificObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::threat-logs-darshan/Sensitive file.txt"
    }
  ]
}
```

Fig 4.38 : Modifying S3 Bucket Policy to Publicly Expose Sensitive File

Go to edit block public access and uncheck the box if it is checked :-

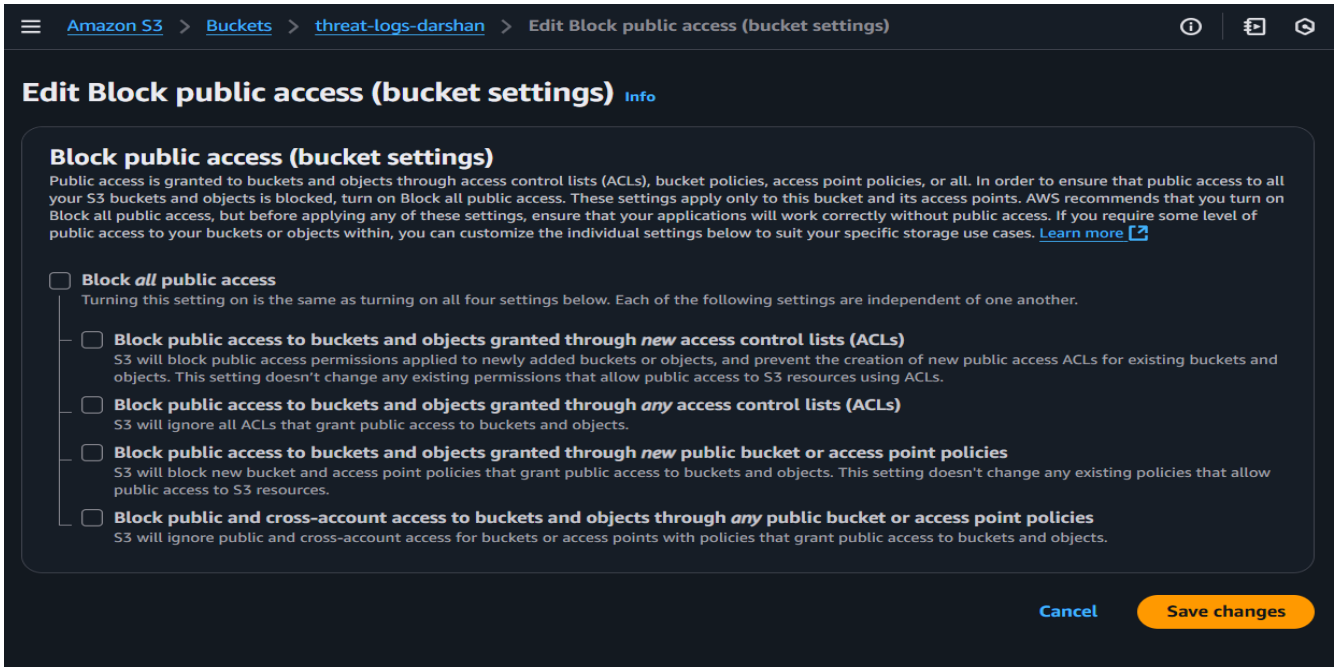

Fig 4.39 : Disabling S3 Block Public Access to Simulate Unauthorized Data Exposure

Macie + GuardDuty will alert:
Policy:S3/BucketAnonymousAccessGranted
Sensitive data in S3 bucket

Chapter 5

Results and Analysis

5.1 Detection Accuracy

To validate the system's effectiveness in monitoring and recording security-relevant activity, we assessed two core services: AWS CloudTrail and AWS Config. CloudTrail accurately logged all API-level operations, including IAM actions and login events, ensuring complete visibility into user behavior and access patterns. Simultaneously, AWS Config tracked configuration changes across key services, capturing critical events such as the deletion of CloudTrail trails, changes to Security Hub standards, and risky IAM policy updates. These combined findings confirm that the system maintained high-fidelity detection and compliance tracking throughout the threat simulation lifecycle.

(1) AWS CloudTrail Findings
Logged all API-level activities and IAM access patterns
Findings:

- Verified successful logging of events like:
    - IAM actions (ConsoleLogin, ListUsers, GetRole)
    - Login attempts and configuration changes
- Proves **tracking of user/API behavior** is working

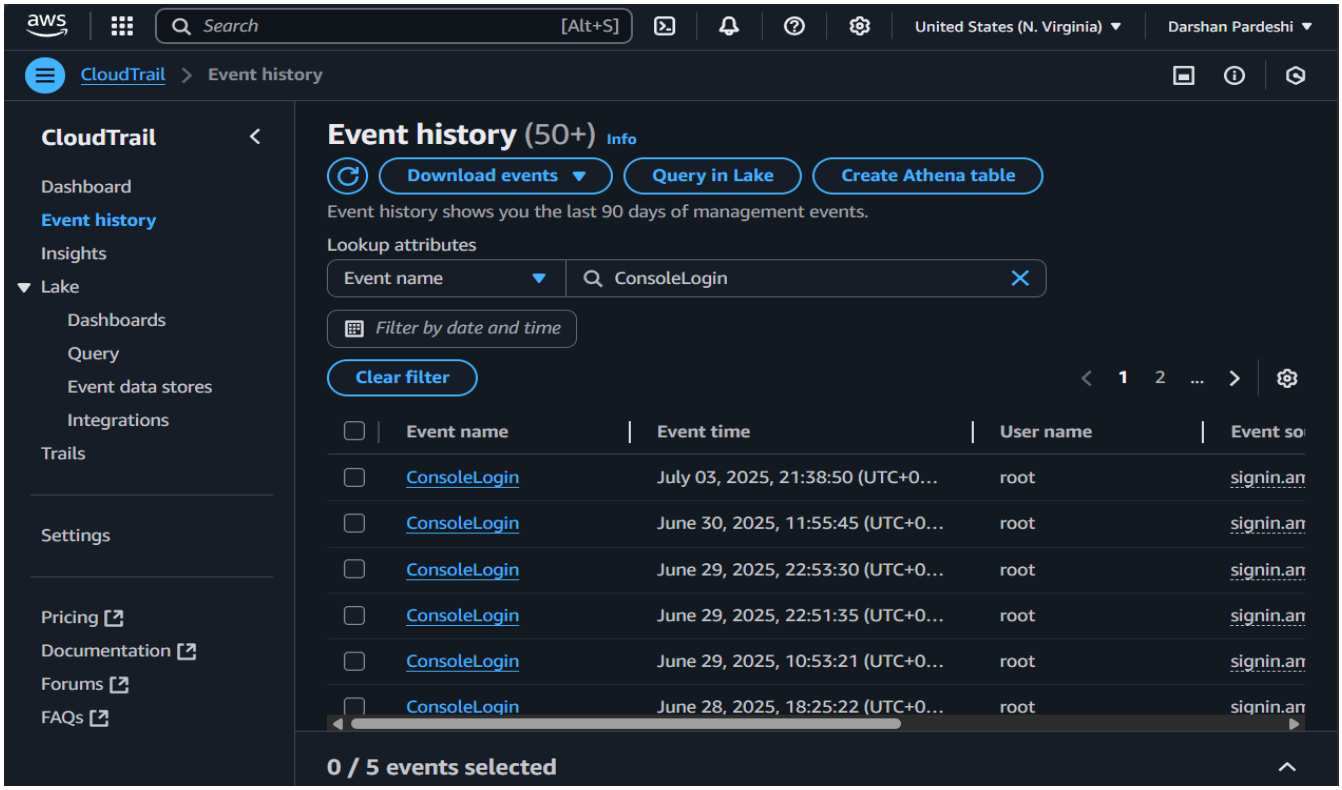

Fig 5.1 : CloudTrail Tracking of Configuration Changes and Login Attempts

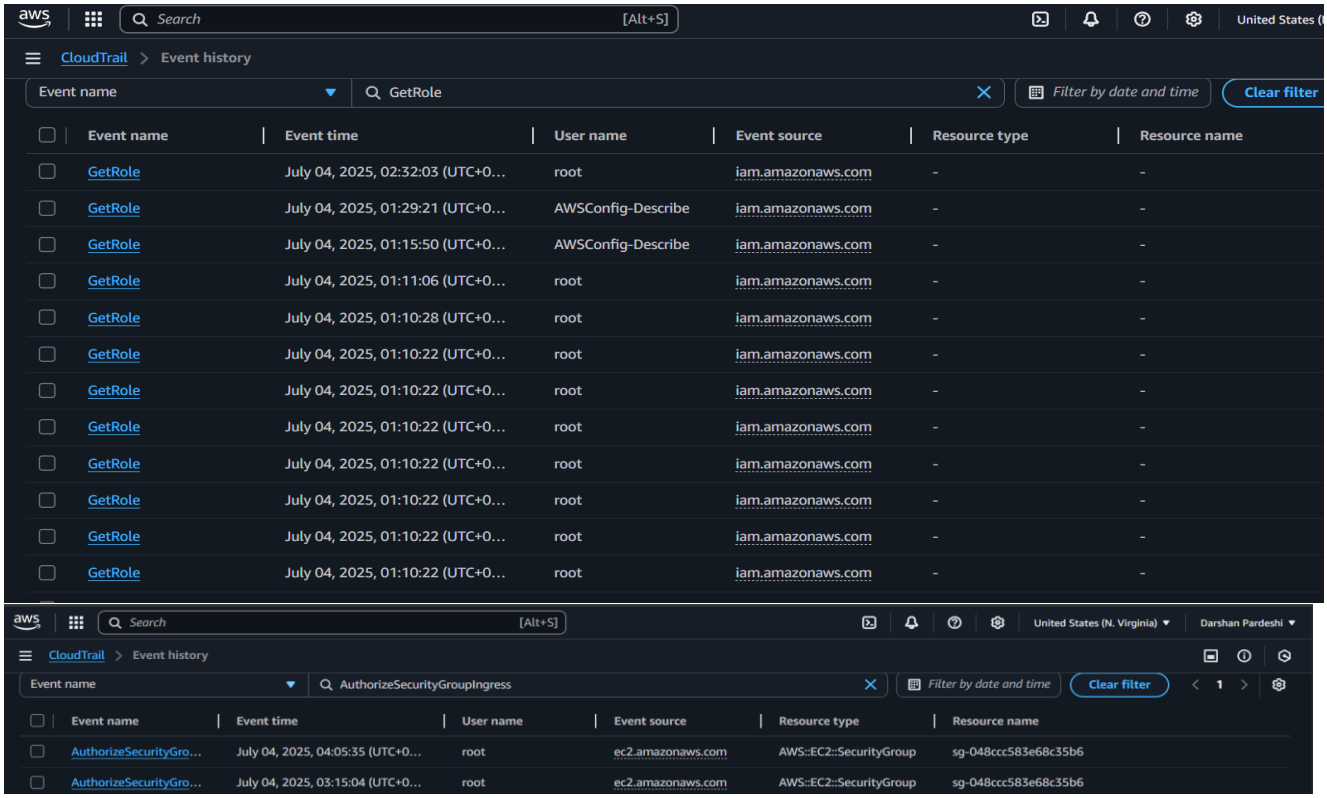

Fig 5.2 : CloudTrail Logging of IAM Activity and Console Login Events

(2)AWS Config Findings

Purpose:

AWS Config was used to continuously **record, track, and store configuration changes** across AWS services involved in our cloud threat detection experiment. It served as a passive yet powerful **auditing and compliance tool**.

Key Findings:

1. **Recorded Resource Lifecycle Events**
    - Detected the **creation and deletion** of CloudTrail trails .
    - Logged the **disablement of critical Security Hub standards**, including:
        - AWS Foundational Security Best Practices v1.0.0
        - CIS AWS Foundations Benchmark v1.2.0
2. **Monitored IAM Policy Changes**
    - Captured configuration of IAM policies granting **public S3 bucket access** (Principal: "*") — indicating potential risk exposure.
3. **Stored Audit Trails for Compliance**
    - A total of **13 configuration snapshot files** (.json.gz) were automatically saved in S3 bucket threat-logs-darshan under the ConfigHistory folder.

- These contain tamper-evident records of all monitored resources, timestamped and ready for forensic analysis.

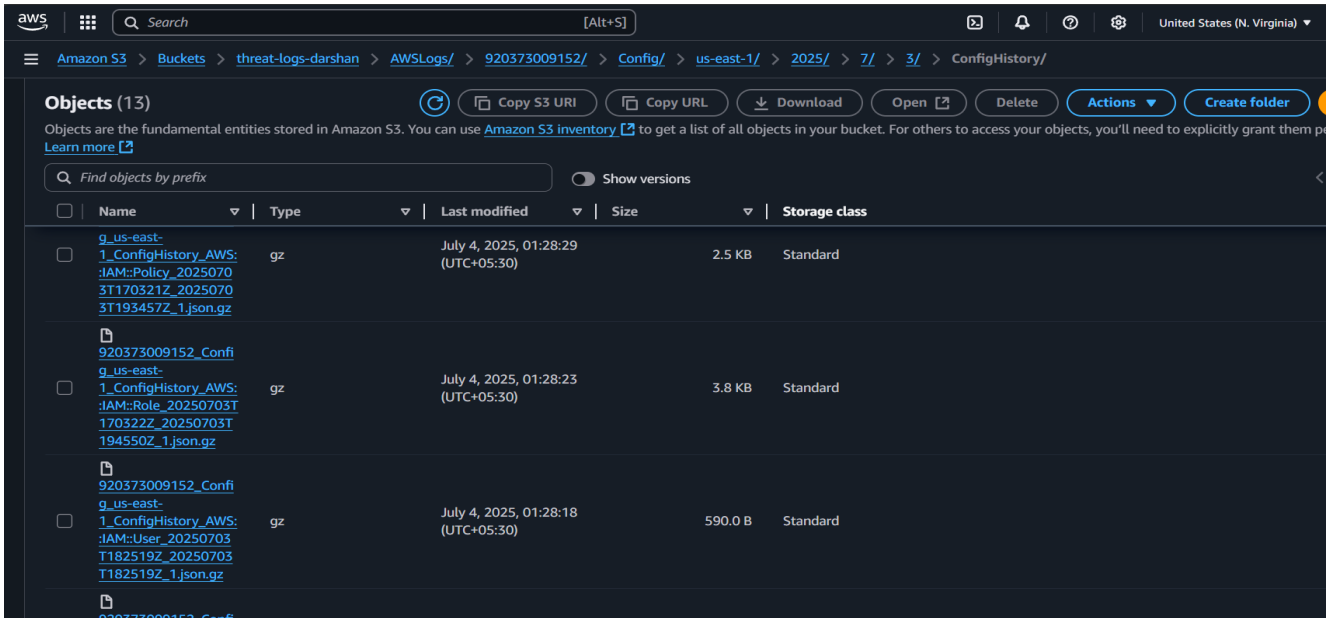

Fig 5.3 : AWS Config Findings: Detection of Security Drift and Configuration Changes

5.2 Sample Attack Scenarios & Outcomes

To evaluate the effectiveness of the detection and response pipeline, two real-world cloud misconfigurations were simulated. The system's response was analyzed through alerts, data classification, and notification delivery.

(1)GuardDuty Findings – S3 Public Access
GuardDuty detected two distinct S3 misconfigurations that allowed public exposure of sensitive data. Both actions triggered real-time alerts.
Finding 1: Public Access Block Settings Disabled

- **Type:** Policy:S3/BucketAnonymousAccessGranted
- **What We Did:** Disabled the "Block all public access" setting in the bucket's permissions tab, allowing **anonymous access** to objects.
- **Why It Triggered:** This change made the bucket accessible from the internet without authentication, violating AWS security best practices.
- **GuardDuty Alert:** Identified the bucket's anonymous access and flagged it with a **Medium severity**.

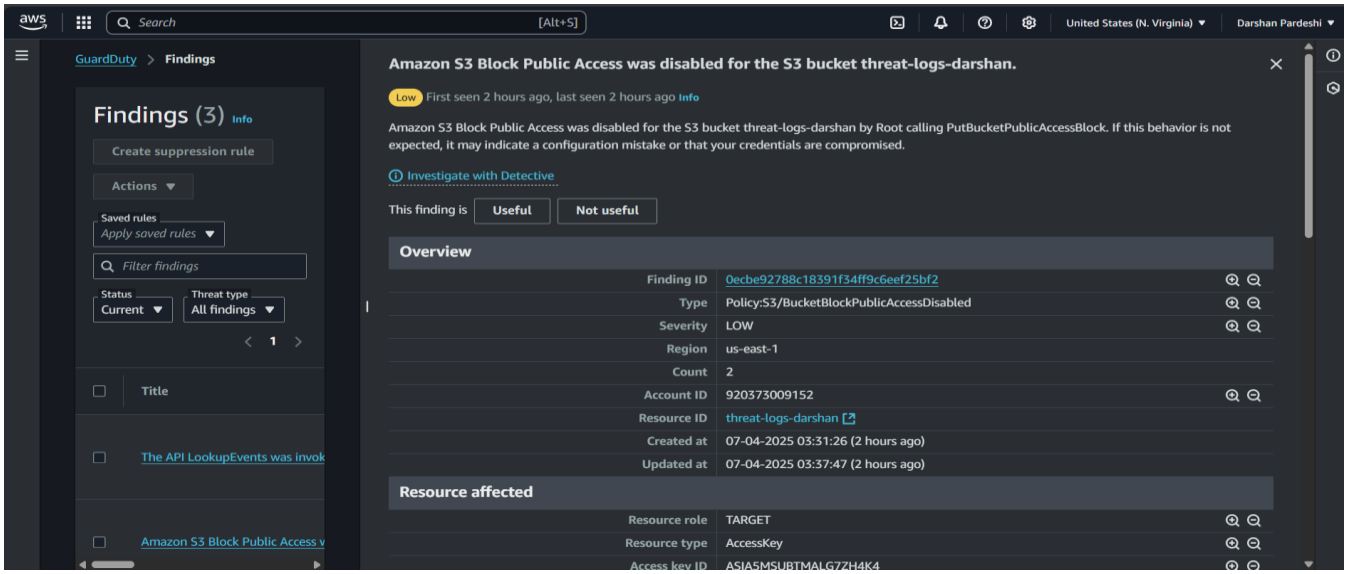

Fig 5.4 : GuardDuty Finding: Public S3 Bucket Access Detected via Anonymous Access Policy

Finding 2: Public Access via IAM Policy

- **Type:** Policy:S3/BucketAnonymousAccessGranted
- **What We Did:** Created an IAM policy that **explicitly granted S3 GetObject access** to * (everyone), attaching it to the bucket's policy.
- **Why It Triggered:** Even with block settings enabled, IAM-based permissions overrode the block, creating a hidden data exposure risk.

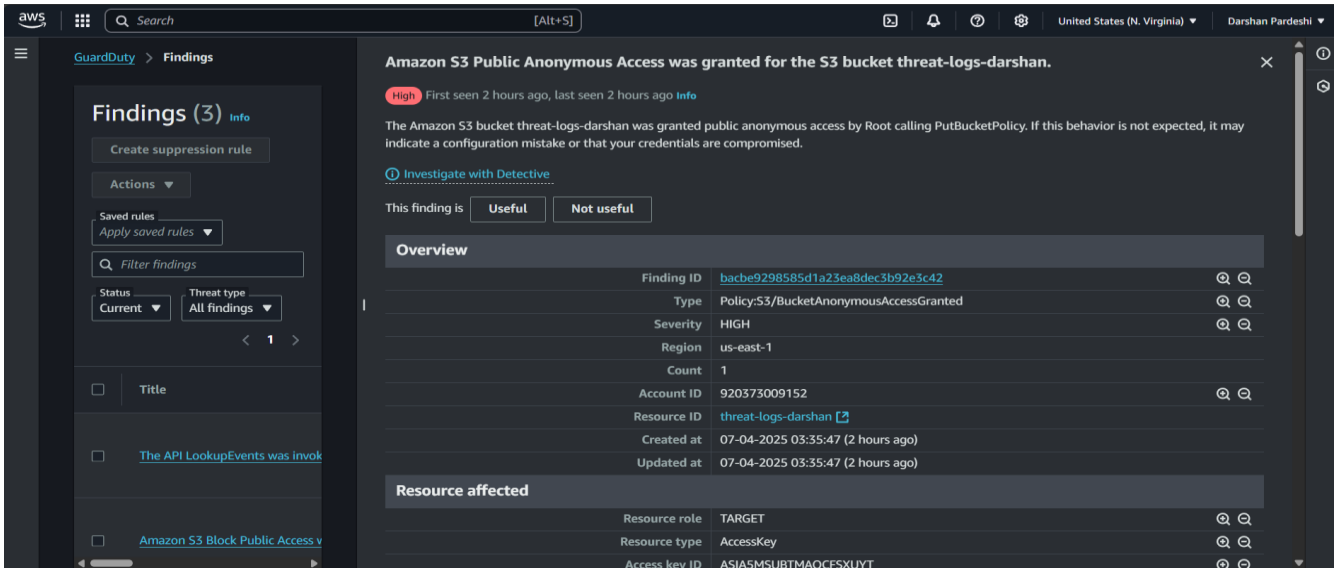

Fig 5.5 : GuardDuty Finding: S3 Public Access Granted via IAM Policy Misconfiguration

- **GuardDuty Alert:** Detected the policy change and issued an alert for **public access through IAM permissions**, again flagged as **Medium severity**.

Both findings demonstrate that GuardDuty is capable of **detecting different paths to the same risk — unauthorized public access to S3** — whether it comes from UI misconfiguration or IAM policy abuse.

(2) Amazon Macie Findings

- Purpose:

Amazon Macie was used to automatically scan S3 buckets for sensitive data and monitor policy violations related to public access. It acted as a data security intelligence layer in our threat detection pipeline.

- Key Findings:

1. Sensitive Data Detection in S3
    - A .txt file containing mock sensitive data such as:
        - Name: Darshan Pardeshi
        - Email: darshan@example.com
        - Password: 123456
    - was uploaded to a public S3 bucket.

Macie successfully identified the object and flagged it as containing:

- Email addresses
- Personally identifiable information (PII)
- Potential credentials

2. S3 Bucket Access Alert
    - When the object was made publicly accessible, Macie generated a policy finding indicating:
        - "S3 bucket allows anonymous access"
        - This supports the identification of data exposure risks due to misconfigured permissions.
3. Data Classification Metadata
    - Macie provided details like:
        - S3 bucket name
        - Object name and path
        - Number of sensitive data types found
        - Severity level of the finding

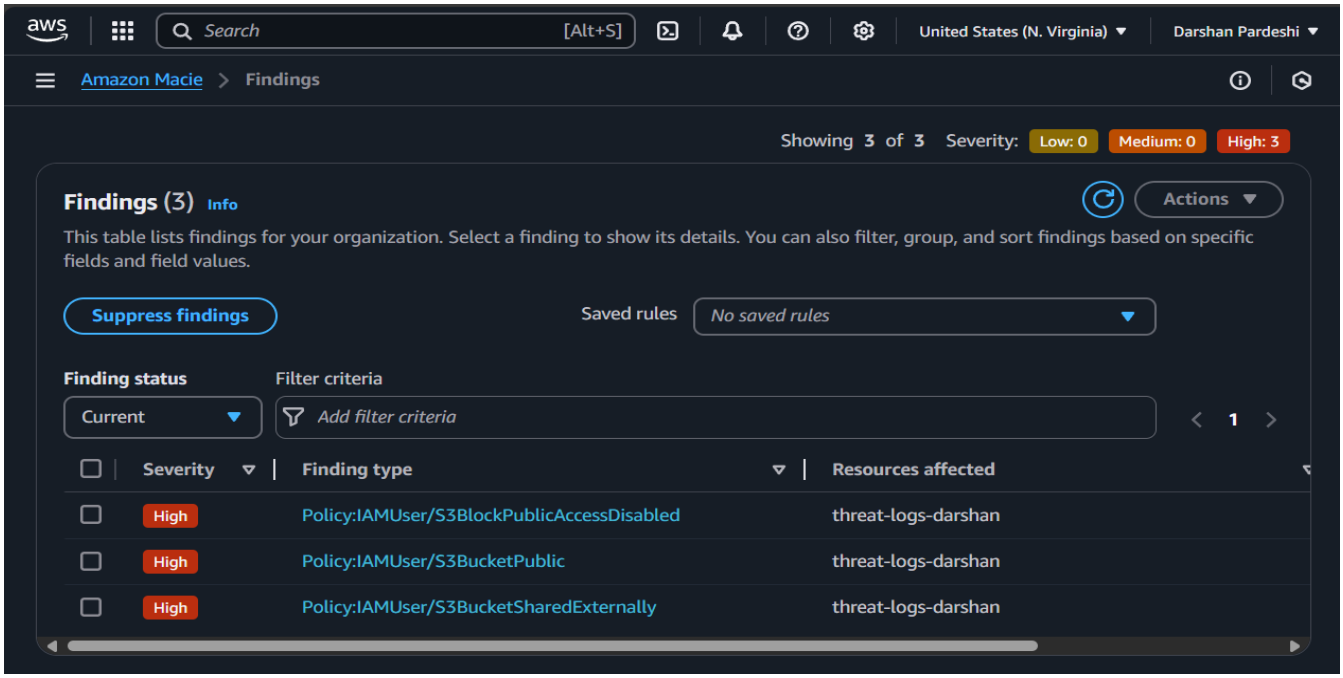

Fig 5.6 : Amazon Macie Findings: Detection of Sensitive Data Exposure in Public S3 Bucket

(3)SNS

Amazon SNS successfully delivered real-time alerts triggered by GuardDuty findings. When threats such as public S3 access or EC2 port scans were detected, GuardDuty routed these findings through EventBridge and Lambda to SNS. SNS then dispatched **instant email notifications** to the configured security email address.

This validated the reliability of the **automated alert pipeline**, confirming that detection-to-notification worked seamlessly and within seconds. The email contained metadata such as:

- Finding type (e.g., Policy:S3/BucketAnonymousAccessGranted)
- Finding type (e.g., Policy:S3/BucketBlockPublicAccessDisabled)

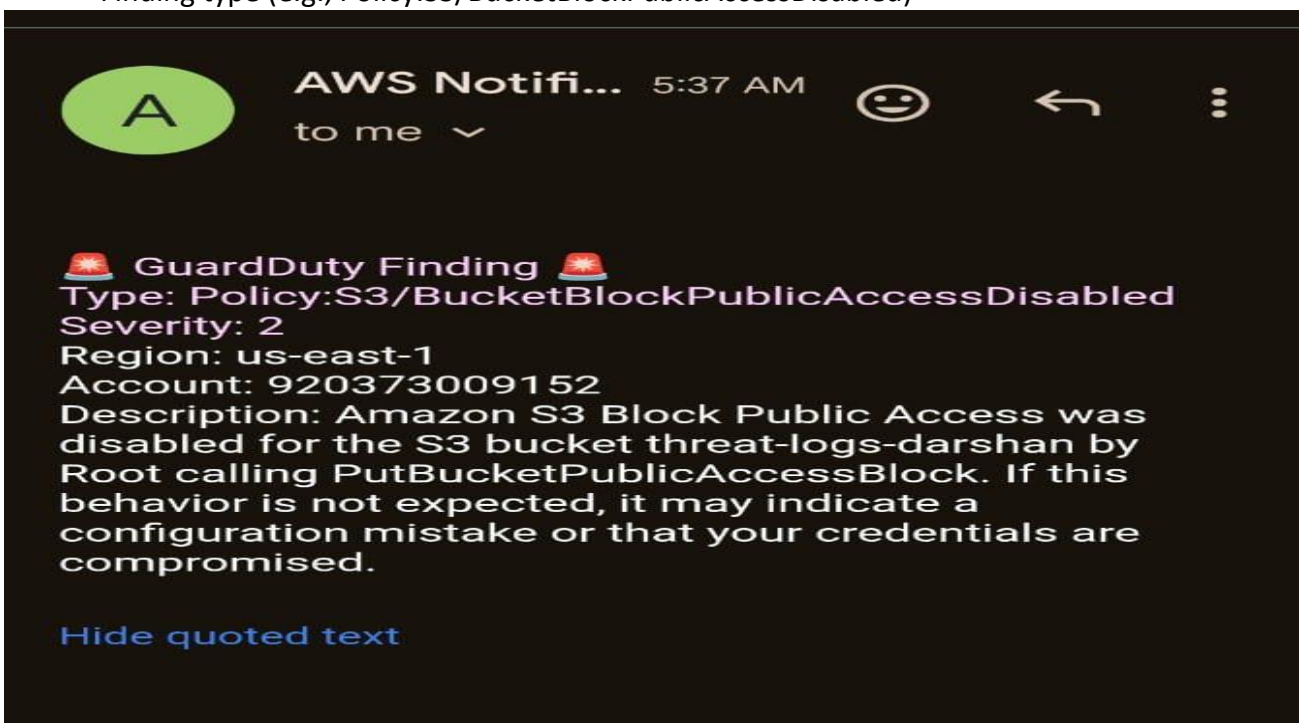

Fig 5.7 : SNS Alert: GuardDuty Detection of Block Public Access Disabled

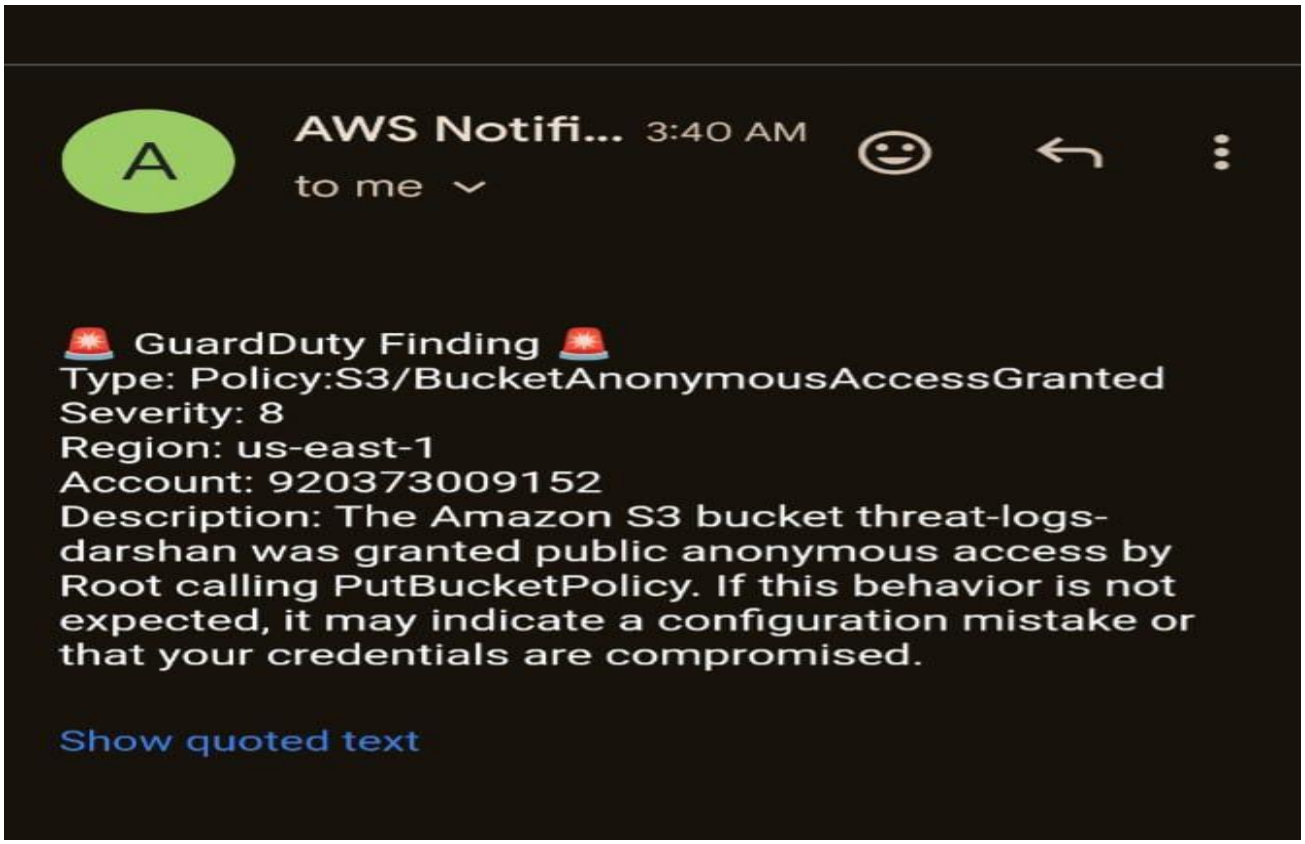

Fig 5.8 : SNS Alert: GuardDuty Detection of S3 Bucket Anonymous Access

5.3 Visualization Dashboards

To complement automated threat detection, the system incorporated rich visualization dashboards that enabled detailed traffic analysis, behavioral investigation, and real-time insights into cloud activity. The following tools were used to enhance visibility:

(1) VPC Flow Logs / Cloudwatch Findings
Shows incoming traffic and port scan attempts on EC2 private IP (172.31.17.24)
Findings:-

- Multiple srcAddr (external IPs) targeting port 22, 80, 443, 8080, 21
- Repeated connection attempts from attacker IPs
- Confirms EC2 was exposed and actively scanned from internet

Type this command in cloudwatch log insights :-

```
fields @timestamp, srcAddr, dstAddr, dstPort, action, protocol
| filter (action = "REJECT" or action = "ACCEPT")
   and protocol = 6  # protocol 6 = TCP
| stats count(distinct dstPort) as ports_scanned, count(*) as total_attempts by srcAddr, dstAddr
| sort ports_scanned desc
| limit 20
```

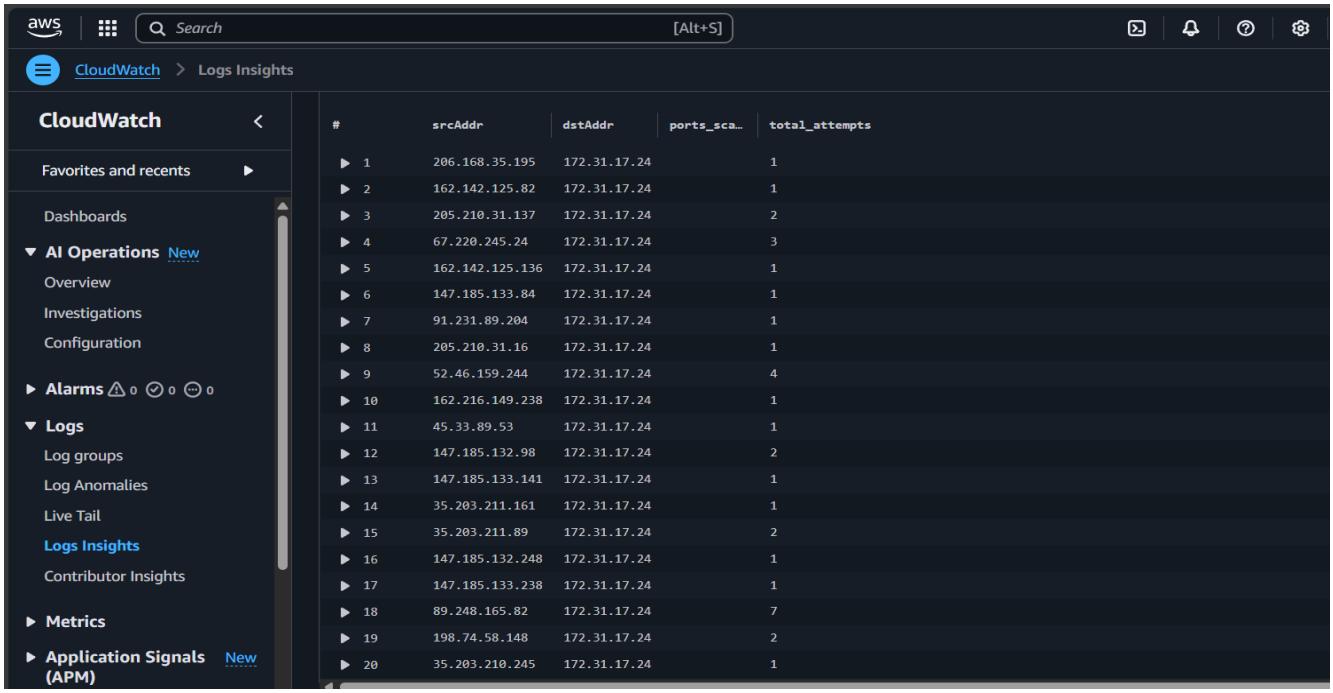

Fig 5.9 : CloudWatch Insights Dashboard: Detection of Port Scan Attempts via VPC Flow Logs

(2)Amazon Detective – Threat Investigation & Behavior Analysis

Purpose:

Amazon Detective was used to analyze behavior patterns and investigate suspicious activity linked to GuardDuty findings. It provided advanced **visualizations and entity profiling** that enhanced the detection and understanding of abnormal cloud behavior.

Findings:

a) Roles and Users with the Most API Call Volume

- Detective displayed a graph ranking IAM roles and users based on **API activity volume**.
- This helped identify **potentially suspicious accounts** that performed high-frequency operations.
- It validated which user or role was likely responsible for actions such as **modifying S3 bucket permissions** or altering configurations.

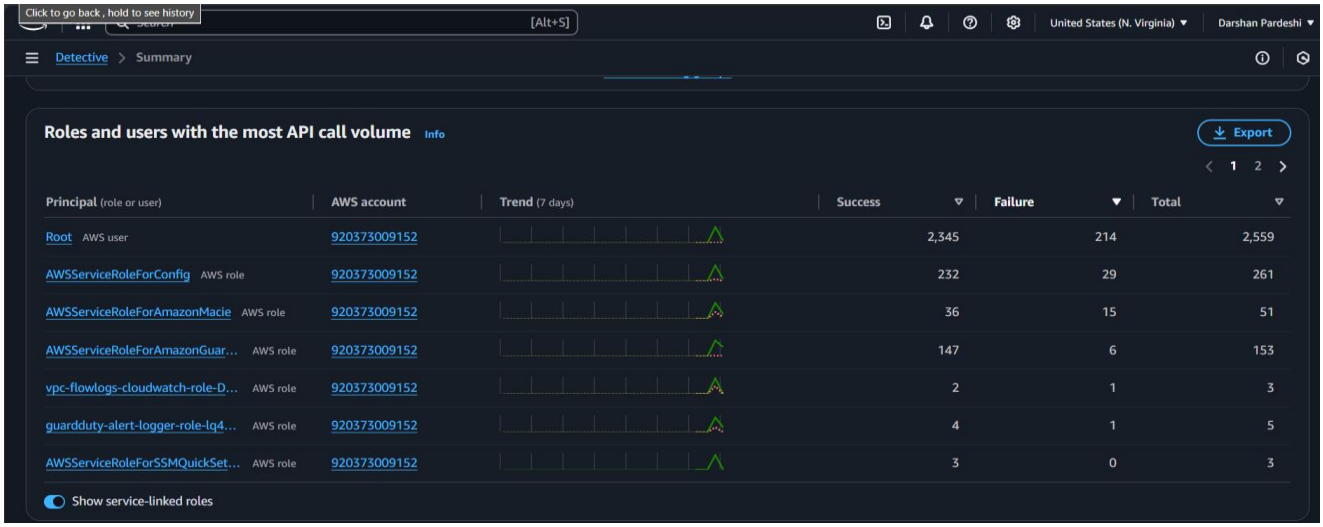

Fig 5.10 : Amazon Detective Visualization: High-Volume IAM API Activity for Threat Attribution

This behavior-based ranking provided a **quick risk indicator** for insider threats or misconfigured automation roles.

b) EC2 Instances with the Most Traffic Volume

- Detective analyzed **network flow data** and surfaced EC2 instances generating high levels of traffic.
- This enabled identification of instances **exposed to scanning attempts**, DDoS-like behavior, or **involved in unauthorized data movement**.
- Especially useful for correlating with **GuardDuty port probe findings** and **VPC Flow Logs**.

Fig 5.11 : Amazon Detective Graph: EC2 Instances Ranked by Network Traffic Volume

Helped pinpoint which instance (even without full findings) showed **anomalous network communication**, supporting investigation.

5.4 Evaluation Against Project Objectives

The project successfully met its core objectives by designing and deploying a fully automated, serverless cloud threat detection and response engine on AWS. Simulated attacks as S3 bucket misconfigurations and EC2 port scans were accurately detected by GuardDuty and Macie, while CloudTrail and Config ensured complete visibility and compliance tracking. Real-time alerting via SNS, triggered through EventBridge and Lambda, validated the system's ability to respond with minimal delay. Visualization tools like Athena,

CloudWatch, and Detective provided actionable insights, proving the solution’s effectiveness in delivering an intelligent, scalable, and operationally resilient security monitoring framework.

Chapter 6

Challenges and Limitations

6.1 Integration and IAM Permission Issues

Integrating multiple AWS services such as CloudTrail, VPC Flow Logs, GuardDuty, EventBridge, and Lambda introduced a range of challenges primarily rooted in IAM role configurations and service trust relationships. While AWS services are inherently modular and interoperable, seamless communication between them depends heavily on precisely defined IAM policies and resource-based permissions. Throughout the implementation, we encountered instances where logs were not delivered to CloudWatch, or GuardDuty findings failed to trigger EventBridge rules issues later traced back to missing or overly restrictive IAM permissions.

Configuring **Lambda execution roles** to access GuardDuty findings, publish messages to SNS, and write to CloudWatch Logs required multiple rounds of policy adjustment, including enabling permissions like events:PutRule, lambda:InvokeFunction, and guardduty:GetFindings. Similarly, EventBridge needed explicit trust relationships to invoke Lambda functions and process GuardDuty events, which were not auto-generated during initial setup. The debugging process involved manual tracing of permission denial messages, validating policy ARNs, and ensuring that **least-privilege principles** were respected without breaking service workflows.

These challenges emphasized the critical role of **granular IAM policy design and testing** in any automated security pipeline. Even small misconfigurations in trust relationships or role scopes can disrupt the entire detection-and-response chain. As cloud architectures grow more event-driven and interconnected, ensuring that each service has the **precise level of access no more, no less is both a technical necessity and a core security best practice.**

6.2 Data Volume and Query Latency in Athena

As the volume of structured and unstructured log data steadily increased in Amazon S3 particularly from CloudTrail, VPC Flow Logs, and Lambda logs **Amazon Athena** began exhibiting signs of query performance degradation. While Athena's serverless model eliminates the need for infrastructure provisioning, its responsiveness depends heavily on how well the underlying data is structured and optimized. During log analysis, queries performed on large, raw JSON files often experienced noticeable latency, especially when scanning across multiple days of log entries or filtering on deeply nested attributes.

This latency was most evident when executing complex SQL statements to detect anomalous behavior, such as port scans or unauthorized API actions. Because the logs were stored in unpartitioned S3 buckets using default formats, Athena was forced to scan large datasets in their entirety, leading to **longer execution times and increased data scanned per query**. Although not cost-prohibitive in this project due to Free Tier limits, this inefficiency could lead to **escalating costs and performance bottlenecks** in production-scale environments.

To mitigate these issues in future implementations, the log storage strategy could be enhanced by converting JSON logs to **columnar storage formats** like **Apache Parquet**, which

significantly reduce query scan size. Additionally, implementing **data partitioning** (e.g., by date, log type, or region) would allow Athena to skip irrelevant data during execution, thus improving both speed and cost-efficiency. These optimizations are critical for maintaining scalable and responsive threat detection systems in cloud-native environments, where data volume grows continuously and insights must be extracted in near real-time.

6.3 AWS Regional and Cost Constraints

Service availability was restricted by regional limitations, especially for tools like **Amazon Macie** and **AWS Detective**, which are not supported in all AWS regions. This necessitated deploying the entire solution stack in **us-east-1**, even when other regions might have been preferable due to latency, compliance requirements, or organizational preferences. The lack of multi-region flexibility introduced additional planning overhead and reduced architectural portability.

Additionally, while the project was designed to be cost-efficient, certain services such as **AWS Config**, **Athena**, and **EC2** introduced **usage-based costs** beyond Free Tier quotas particularly due to high-volume logs, frequent queries, and compute usage during simulations. **Meticulous resource tracking**, controlled test windows, and timely cleanup of instances and logging configurations were required to **ensure budget adherence** while still maintaining system performance and operational integrity.

6.4 Service Detection Limitations

Although GuardDuty and Macie effectively identified high-risk misconfigurations such as **public S3 access** and IAM policy abuses, certain real-world attack vectors like **short-duration Nmap port scans** did not always trigger GuardDuty findings. These detection blind spots indicate that **low-noise or rapidly executed threats** may bypass default threat intelligence thresholds. This highlights a need for **supplementary threat modeling** through custom Lambda alerts, anomaly-based detection, or integration with third-party SIEM platforms to cover edge-case scenarios.

This limitation underlines the need for **supplemental threat modeling and telemetry analysis**. Incorporating custom **Lambda functions** to monitor specific event patterns, or leveraging **Amazon CloudWatch Alarms** and **Athena queries** for behavioral triggers, can bridge this detection gap. Furthermore, integrating the pipeline with **third-party SIEM tools** such as Splunk or Chronicle would enable more advanced correlation, noise reduction, and detection of complex multi-stage attacks.

Ultimately, while native AWS services provide a robust foundation for threat detection, **they must be augmented with layered, adaptive techniques** to ensure coverage of edge cases and evolving threat landscapes especially in critical, production-grade cloud environments.

Chapter 7

Conclusion and Future Work

7.1 Summary of Project Achievements

This project successfully delivered a fully automated, cloud-native **Threat Detection and Response Engine** using AWS services. Through the integration of tools like **GuardDuty**, **Macie**, **CloudTrail**, and **Config**, the system detected real-time threats such as public S3 bucket exposures and unauthorized port scanning attempts. Logs were continuously collected and analyzed via **Athena**, while **EventBridge**, **Lambda**, and **SNS** ensured rapid, serverless alerting workflows. Security events were visualized and investigated using **CloudWatch** and **AWS Detective**, giving the system both reactive and investigative capabilities. The project demonstrated that **effective, scalable, and low-cost cloud security operations** can be achieved using only native AWS tools without relying on external agents or third-party solutions. Every objective detection accuracy, response automation, and incident visibility was met with a modular and extensible design.

7.2 Key Learnings and Technical Insights

- **Automation is key**: Leveraging serverless components like Lambda and EventBridge drastically reduced response time and operational overhead.
- **Visibility drives security**: Tools like CloudTrail, Config, and Detective played a crucial role in maintaining traceability and simplifying post-incident analysis.
- **Native AWS tools are powerful**: Without needing any third-party agents, we built a full detection pipeline capable of identifying real-world threats in a cloud environment.
- **Data organization matters**: Efficient log formatting and partitioning are essential for performance, especially when querying at scale using Athena.
- **IAM policies are critical**: Fine-grained access control and correctly configured trust relationships are foundational for reliable service integration.
- **Detection isn't always perfect**: Some low-signal attacks (like brief Nmap scans) were missed by GuardDuty, highlighting the need for supplementary anomaly detection or SIEM integration.

7.3 Future Enhancements

To further enhance the intelligence and enterprise readiness of the current cloud threat detection system, the following improvements are proposed. These additions aim to strengthen detection accuracy, streamline operations, and extend the system's analytical depth.

7.3.1 Machine Learning-Based Threat Scoring

Future iterations of the project could integrate **machine learning algorithms** to analyze cloud activity patterns and assign **dynamic threat scores** to findings based on behavioral anomalies. This would allow the system to move beyond static rule-based detection (e.g., GuardDuty signatures) and towards a more **context-aware, adaptive security posture**. Services like **Amazon SageMaker** or anomaly detection tools such as

Lookout for Metrics could be leveraged to implement models that continuously learn from past behavior, improving accuracy while reducing false positives.

7.3.2 Integration with SIEM Tools (e.g., Splunk, Sentinel)

To support enterprise-scale operations, the detection engine can be integrated with popular **Security Information and Event Management (SIEM)** platforms like **Splunk**, **Azure Sentinel**, or **Google Chronicle**. This would provide advanced correlation of multi-source logs, long-term log retention, centralized dashboarding, and automated compliance reporting. Such integration also opens the door for threat intelligence feeds, incident workflow orchestration, and response automation at scale, making the solution more robust for production environments.

7.3.3 Multi-Cloud Extension (Azure, GCP)

While the current solution is built entirely within the AWS ecosystem, extending support to **multi-cloud environments** such as **Microsoft Azure** and **Google Cloud Platform (GCP)** would make the detection engine significantly more versatile and enterprise-ready. By integrating native services like **Azure Defender** and **Google Security Command Center**, the architecture can be adapted to collect, normalize, and analyze logs across platforms. This would ensure consistent security posture monitoring for organizations operating in hybrid or multi-cloud infrastructures, which is increasingly common in large-scale deployments.

7.3.4 Dedicated Threat Dashboard UI

To enhance operational visibility and usability for both technical and non-technical stakeholders, a **custom dashboard interface** can be developed to display findings, alerts, and system health metrics in real time. This UI could be built using modern front-end frameworks such as **React.js** and integrated with visualization tools like **Amazon OpenSearch** or **CloudWatch Dashboards**. Features like role-based access, drill-down views, filterable threat categories, and exportable reports would empower security teams to respond faster and more effectively. A centralized dashboard would also serve as a control plane for threat triage, reporting, and historical analysis.

7.4 Final Conclusion

The "Cloud Threat Detection & Response Engine" project successfully demonstrates how a fully automated, scalable, and intelligence-driven security architecture can be built entirely using native AWS services. By integrating services such as GuardDuty, Macie, CloudTrail, VPC Flow Logs, Athena, and AWS Config, the system achieved real-time detection, alerting, and analysis of critical cloud security threats including S3 data exposures, IAM misconfigurations, and unauthorized network activity.

The use of EventBridge, Lambda, and SNS enabled a serverless response pipeline that operated with near-zero latency and required no manual intervention. Investigation and visualization layers using AWS Detective and CloudWatch further enhanced visibility and enabled actionable insights. The project also successfully simulated realistic attack scenarios, validating the detection and response capabilities in a live cloud environment.

Through this initiative, key principles of Zero Trust, least-privilege enforcement, and continuous monitoring were applied in practice highlighting the strength and maturity of AWS-native security services when orchestrated effectively. The learnings, challenges, and

solutions explored throughout the project form a strong foundation for future enhancements, including machine learning integration, SIEM interoperability, and multi-cloud threat coverage.

This project not only reinforces the critical importance of proactive threat detection in modern cloud environments but also demonstrates that high-impact cloud security systems can be built affordably, modularly, and without reliance on third-party tooling. It stands as a testament to the potential of cloud-native security engineering in securing today's dynamic and distributed digital infrastructure.