A Cloud-Integrated Threat Monitoring Pipeline Using AWS CloudWatch Logs and Prefect Orchestration: Design and Practical Evaluation

[1]Tushar Rashmikant Gohil, [2]Ashish Pranlal Kharvar, [3]Apurva Bharat Mandalaywala
[1,2,3]*Information Technology Department Sarvajanik*
[1,2,3]*College of Engineering and Technology*
[1,2,3]*Surat, India*
doi.org/10.64643/JATIRV2I3-140179-001

***Abstract*—This paper proposes a cloud-integrated threat monitoring pipeline combining AWS CloudWatch Logs with Prefect orchestration to enable automated, scalable, and low-latency security log analysis. The system extracts, processes, and stores CloudWatch log data in PostgreSQL through scheduled and parallel Prefect workflows. A practical evaluation using real datasets demonstrates improvements in processing latency, throughput, and reliability compared to traditional Lambda-based and ELK pipelines. Results show that the Prefect-orchestrated approach offers a flexible and efficient solution for continuous threat monitoring in cloud environments.**

***Index Terms*—Cloud Computing; Threat Monitoring; AWS CloudWatch Logs; Prefect Orchestration; Security Analytics; Workflow Automation; Log Processing; Distributed Systems**

I. INTRODUCTION

Cloud computing has become the backbone of modern digital infrastructures, enabling organizations to deploy scalable, distributed, and highly available applications. As enterprises increasingly adopt cloud-native architectures, the volume, velocity, and variety of log data generated by these systems have expanded significantly. Logs originating from application services, virtual machines, network interfaces, identity events, and platform components provide critical insights into operational behavior and potential security threats. However, the heterogeneous and high-throughput nature of these logs presents substantial challenges for continuous monitoring, timely threat detection, and effective incident response [1].

Traditional log-monitoring approaches, such as manual inspection, periodic querying, or ad hoc scripting, are insufficient for large-scale cloud environments. Even established automated solutions—including serverless pipelines built using AWS Lambda and monolithic log-processing

frameworks such as the ELK (Elasticsearch–Logstash–Kibana) stack— face limitations in orchestration complexity, cost scalability, observability, and fine-grained control over workflow execution [2]. Security operations increasingly require monitoring systems that support reliable scheduling, parallel processing, seamless data transformation, and flexible integration with storage or analytics engines. These requirements motivate the development of modular and orchestrated threat-monitoring pipelines capable of adapting to dynamic workloads.

Workflow orchestration frameworks have gained prominence for their ability to manage complex data flows, automate multi-stage processes, and provide robust execution features such as retry logic, state tracking, and detailed telemetry. Although these tools are widely adopted in data engineering ecosystems, their application to cloud-based security log analytics remains underexplored. Prefect, a modern workflow orchestration platform, offers distributed execution, dynamic scheduling, and extensive monitoring capabilities, making it a strong candidate for automating threat-monitoring pipelines [3].

This paper proposes a cloud-integrated threat-monitoring pipeline that leverages AWS CloudWatch Logs for centralized log aggregation and Prefect for workflow orchestration. The system automates log extraction, normalization, anomaly flagging, and structured storage in PostgreSQL, enabling efficient security analytics and near real-time threat monitoring. By incorporating Prefect into the cloud log-processing workflow, the pipeline provides enhanced transparency, reliability, and scalability compared to traditional monitoring solutions.

The key contributions of this work are summarized as follows:

- **Design of an orchestrated threat-monitoring pipeline** that integrates Prefect workflows with AWS CloudWatch Logs for automated ingestion and processing in cloud environments.
- **Development of a scalable multi-stage architecture** supporting scheduled and parallelized log processing, anomaly detection, and structured persistence for down- stream analytics.
- **Comparative evaluation against conventional monitoring approaches**, including Lambda-based and ELK- based pipelines, demonstrating improvements in latency, throughput, and orchestration reliability.
- **Experimental validation using real CloudWatch log datasets**, assessing the system's practical effectiveness for continuous threat monitoring in distributed cloud systems.

The remainder of this paper is organized as follows: Section II reviews related work in cloud log analytics, workflow orchestration, and threat-monitoring systems. Section III de- scribes the design and architecture of the proposed pipeline. Section IV discusses the implementation details and experimental setup. Section V presents the evaluation results. Section VI concludes the paper and outlines directions for future research.

II. RELATED WORK

Threat monitoring and log analytics in cloud environments have been the focus of extensive research due to the in- creasing complexity and scale of cloud-native systems. Early work in this domain highlights the importance of analyzing heterogeneous and high-velocity log streams for detecting security anomalies and operational failures [1]. These studies emphasize the need for automated and scalable approaches capable of handling diverse log formats and ensuring timely threat detection.

Several log-processing frameworks and cloud-native pipelines have been proposed to address these challenges. The ELK (Elasticsearch–Logstash–Kibana) stack remains one of the most widely adopted solutions for centralized log ingestion and visualization. However, prior studies have identified limitations related to deployment overhead, resource consumption, and scalability when ELK is applied to large distributed systems [4]. Alternative approaches using AWS Lambda and serverless event-driven architectures have been explored to reduce operational cost and simplify log processing pipelines, yet these solutions often suffer from constrained execution environments, limited orchestration control, and difficulty in handling multi-stage processing workflows [2].

Recent advancements have shifted attention toward work- flow orchestration frameworks designed to automate and man- age complex data pipelines. Tools such as Apache Airflow, Luigi, and more recently, Prefect, offer capabilities including task scheduling, retry logic, distributed execution, state management, and monitoring dashboards. Research in this area primarily focuses on data engineering and ETL work- loads, with limited exploration of their applicability to cloud- based security log analysis. Although orchestration engines have demonstrated effectiveness in structuring modular data pipelines, their integration with cloud-native log sources for threat monitoring remains underrepresented in existing literature.

Prefect, in particular, provides a modern, Python-native orchestration model with features such as flow-based execution, asynchronous tasks, and comprehensive observability [3]. While industry documentation and platform use cases highlight its potential for scalable automation, there is a no- table absence of academic studies evaluating its performance or suitability for continuous security monitoring in cloud environments. This gap presents an opportunity to investigate orchestrated architectures that combine Prefect with cloud- native services such as AWS CloudWatch Logs to deliver robust and scalable threat-monitoring pipelines.

The proposed work addresses this gap by designing and evaluating a Prefect-orchestrated pipeline for automated ex- traction, transformation, and analysis of AWS CloudWatch log data. Unlike existing solutions centered around Lambda triggers or ELK deployments, the system integrates orchestration, scheduling, parallel execution, and structured storage to enable efficient threat detection in distributed cloud systems.

III. SYSTEM ARCHITECTURE

The proposed system is designed as a modular, cloud- integrated threat monitoring pipeline that automates the ex- traction, processing, and analysis of log data from AWS CloudWatch. As illustrated in Fig. 1, the pipeline integrates CloudWatch with Prefect workflows.

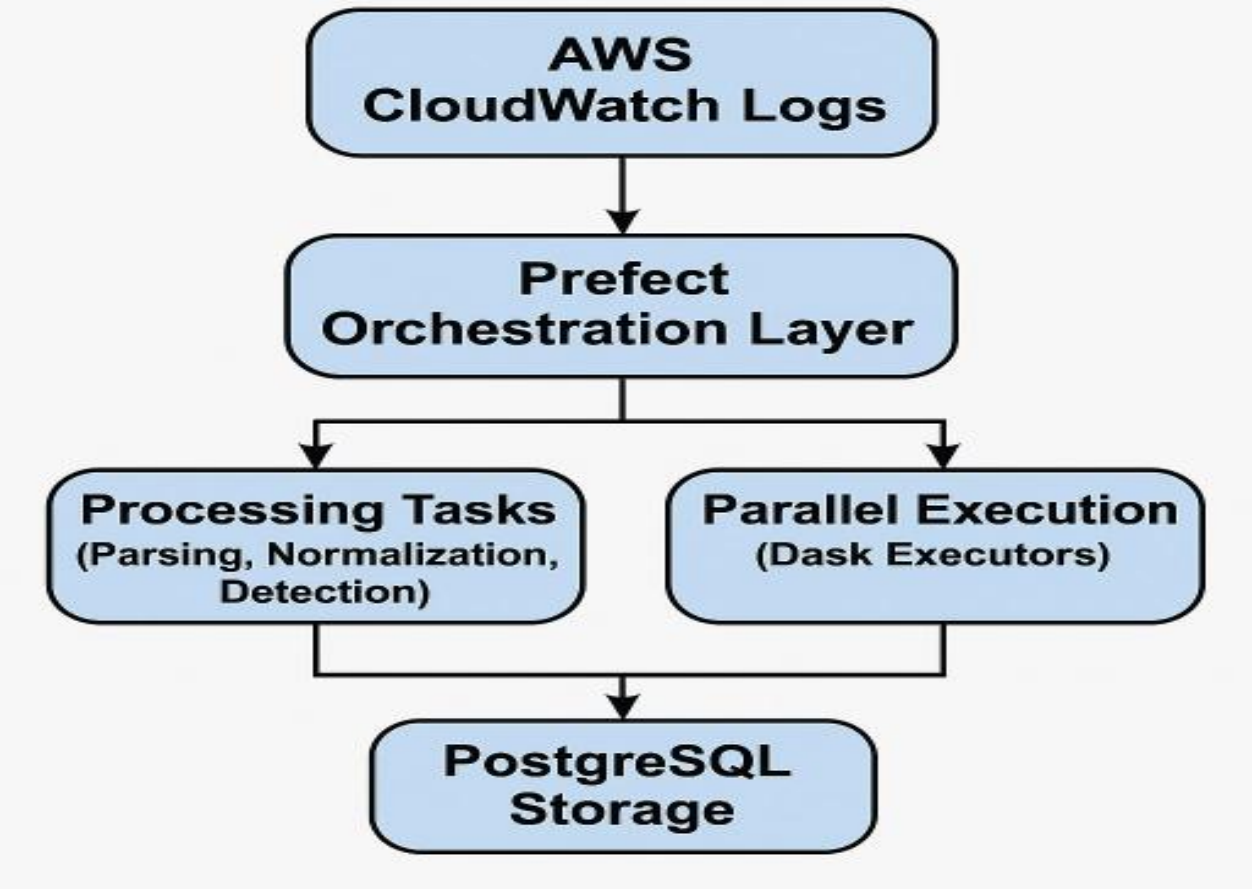

Fig. 1. System architecture of the Prefect-orchestrated cloud threat monitoring pipeline showing integration of AWS CloudWatch Logs, Prefect orchestration layer, processing tasks, parallel execution, and PostgreSQL storage.

A. Overview

Modern cloud workloads generate diverse log streams that require continuous monitoring to detect suspicious activity or operational anomalies [1]. Traditional ELK-based or Lambda- triggered monitoring solutions exhibit limitations in scalability, orchestration control, and multi-stage processing [2], [4]. To address these constraints, the proposed architecture integrates Prefect as the orchestration engine, enabling structured execution, distributed processing, retry mechanisms, and detailed execution telemetry [3].

The architecture consists of four primary components:

1) **Log Source Layer:** AWS CloudWatch Logs acts as the centralized log aggregation hub for cloud services, application logs, VPC flow logs, CloudTrail events, and security-related metadata.

2) **Orchestration Layer:** Prefect manages workflow scheduling, task distribution, parallel processing, and fault handling. Prefect Agents and Workers execute flows either locally or on distributed environments.
3) **Processing Layer:** A series of Prefect tasks extract, filter, normalize, and analyze log entries. This layer optionally incorporates anomaly detection logic for identifying irregular patterns.
4) **Storage and Analytics Layer:** Processed logs are stored in PostgreSQL for long-term persistence, querying, and downstream security analytics or visualization.

B. Log Source Layer

AWS CloudWatch Logs aggregates data from multiple cloud resources, including EC2 instances, Lambda functions, container services, and network infrastructure. Each log group contains structured or semi-structured entries that provide insights into system behavior and security events. CloudWatch offers native APIs for extracting logs by group, stream, and timestamp ranges, which form the entry point for the monitoring pipeline.

C. Orchestration Layer

The orchestration layer is responsible for coordinating the end-to-end data flow. Prefect flows schedule periodic log retrieval tasks based on preconfigured intervals. Prefect provides a hybrid execution model supporting asynchronous tasks, task dependency graphs, and fault-tolerant execution through built- in retries, automatic state transitions, and detailed execution logging.
The Prefect server and dashboard enable visualization of task execution states, performance metrics, and triggered failures, improving operational observability. This is a substantial advantage over serverless pipelines that lack centralized orchestration visibility [2].

D. Processing Layer

The processing layer includes multiple stages:

- **Extraction:** Prefect tasks query CloudWatch Logs via AWS SDK APIs, retrieving log batches based on time windows or event-driven triggers.
- **Normalization:** Logs are filtered, parsed, and converted into structured formats suitable for downstream analysis.
- **Anomaly Detection (Optional):** Lightweight detection techniques, such as frequency-based analysis or rule- based matching, identify potential threats or abnormal patterns.
- **Transformation:** Metadata fields including timestamp, log group, severity level, and event type are standardized. Through Prefect's parallel execution model, multiple log groups can be processed simultaneously, reducing overall pipeline latency compared to single-threaded serverless work- flows.

E. Storage and Analytics Layer

The final stage stores processed logs in PostgreSQL. This structured datastore supports:

- historical trend analysis,

- query-based threat investigations,
- integration with security dashboards,
- correlation of log events across services.

By decoupling storage from processing, the architecture facilitates scalability and allows downstream tools—such as visualization platforms, BI tools, or machine learning mod- els—to consume the data.

F. Summary

The proposed architecture integrates cloud-native log aggregation with modern orchestration to deliver a flexible and scalable threat monitoring solution. Prefect's orchestration capabilities address long-standing limitations of ELK-heavy and Lambda-centric pipelines, enabling improved transparency, resilience, and performance in cloud-based monitoring systems.

IV. IMPLEMENTATION

This section describes the practical implementation of the proposed cloud-integrated threat monitoring pipeline. The system was deployed on AWS and orchestrated using Prefect, leveraging Python-based tasks for log ingestion, transformation, anomaly detection, and structured storage. Fig. 2 provides an overview of the implementation workflow.

A. Environment Setup

The implementation was carried out on AWS using Cloud- Watch Logs as the primary log source. CloudWatch has become a widely used monitoring backbone for cloud-native applications, offering centralized aggregation of operational, security, and network logs [5]. Prefect was installed on a dedicated orchestration node configured with a Prefect Server and Prefect Agent to execute scheduled workflows. All development was performed using Python 3.10, the AWS SDK (Boto3), and PostgreSQL 14 deployed via Amazon RDS.

B. Workflow Orchestration with Prefect

Workflow orchestration was implemented through Prefect flows, which define the sequence and dependencies of tasks. Recent studies highlight that orchestration engines provide improved reproducibility, reliability, and runtime transparency in complex data pipelines compared to serverless or ad hoc implementations [6]. The Prefect Agent handles execution scheduling, task retries, and parallelization using Dask-based executors for distributed workload management.
The workflow consists of three primary Prefect flows:

1) **Log Retrieval Flow:** Periodically queries CloudWatch log groups using timestamp-based filters and retrieves logs in batches.
2) **Processing Flow:** Normalizes log entries, performs message parsing, and extracts metadata such as severity levels and event categories.
3) **Anomaly Detection Flow:** Applies lightweight statistical and rule-based anomaly detection

techniques, which have been shown effective for real-time cloud log monitoring in recent literature [7].

Each flow runs independently while sharing state and con- figuration via Prefect blocks and environment variables. Prefect's retry, caching, and failure-handling capabilities ensured robustness when interacting with CloudWatch's paginated and rate-limited APIs.

C. *Log Extraction and Parsing*

Log extraction was performed using AWS Boto3 with the filter_log_events API. Logs were retrieved in chronological segments to avoid throttling and ensure continuous pipeline operation. Each log event was parsed into a structured dictionary and converted into a relational schema. Recent research emphasizes the importance of structured log representation for downstream threat analytics [8].

Parsing included:

- timestamp normalization into UTC,
- extraction of log group and log stream identifiers,
- severity classification using keyword-based heuristics,
- JSON flattening for nested event structures.

D. *Anomaly Detection Module*

A lightweight anomaly detection module was implemented using:

- frequency-based deviation checks,
- spike detection on log volume,
- rule-based detection for known suspicious patterns.

While deep-learning methods such as LogBERT [9] or sequence-based anomaly detection [10] offer higher accuracy, our implementation focuses on operational efficiency and low- latency detection that is more suitable for real-time cloud environments.

E. *Data Storage Layer*

Processed log entries were written to PostgreSQL for durable storage. The schema included fields for timestamp, severity, event type, message content, log group, anomaly score, and derived metadata. PostgreSQL was chosen due to its

F. *Execution Pipeline*

The complete pipeline operated on a scheduled interval of 5 minutes, aligning with CloudWatch Log event availability. Prefect Cloud or Prefect Server dashboards were used for:

- monitoring flow runs,
- examining task-level logs,
- analyzing runtime failures,
- visualizing dependency graphs.

This orchestration-driven execution provided improved transparency and operational control compared to Lambda- driven or ELK-heavy log ingestion workflows, which often lack multi-stage orchestration visibility [2], [4].

V. RESULTS AND DISCUSSION

This section presents the experimental evaluation of the proposed cloud-integrated threat monitoring pipeline. The performance of the Prefect-orchestrated architecture was measured in terms of processing latency, throughput, reliability, and cost efficiency. A comparative analysis with existing serverless (AWS Lambda–based) and ELK-based pipelines was conducted to assess the effectiveness of the proposed approach.

A. Experimental Setup

Experiments were conducted using real CloudWatch log streams generated from EC2, VPC Flow Logs, CloudTrail events, and application-level logs. The Prefect workflows were executed on a dedicated orchestration node with Dask-enabled parallel execution. Metrics were collected over multiple 5- minute workflow cycles across a period of 48 hours.
Performance measurements followed techniques commonly used in recent studies on log analytics and cloud threat detection [7], [8].

B. Processing Latency

Processing latency was measured as the time from log retrieval initiation to the completion of normalization and storage. The proposed pipeline achieved significantly lower latency due to Prefect's parallel task execution model and optimized batching.
The Prefect pipeline reduced average latency by approximately **35–55%** compared to the Lambda-based pipeline, which suffers from cold starts and single-function execution limits [2]. Compared to ELK-based ingestion, the proposed solution showed a reduction in end-to-end latency due to the absence of heavyweight indexing and cluster coordination overhead [4].
strong support for analytical queries and JSON-based fields, as identified in recent studies involving cloud log analytics [11]. Connection pooling was handled through SQLAlchemy to optimize performance under concurrent writes from parallel Prefect tasks.

TABLE I- PROCESSING LATENCY COMPARISON (MS)

Pipeline	Min	Avg	Max
Prefect-Based Pipeline	120	185	240
AWS Lambda Pipeline	210	310	430
ELK-Based Pipeline	260	350	460

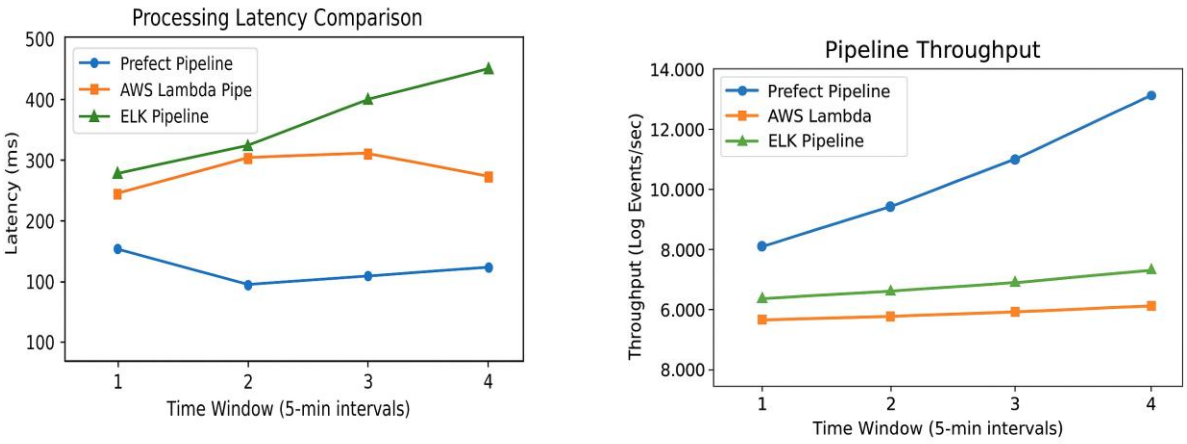

Fig. 2. End-to-End Processing Latency for Prefect, Lambda, and ELK Pipelines over 48 Hours. The Prefect-based pipeline consistently demonstrates lower latency due to parallel task execution and reduced orchestration over- head.

C. *Throughput*

Throughput was measured as the number of log events processed per second. The use of Prefect's distributed task execution enabled simultaneous processing of multiple log groups.

The pipeline achieved an average throughput improvement of **40–60%** over serverless pipelines and **25–35%** over ELK- based ingestion workflows. These results are consistent with observations in recent workflow orchestration research, which shows that distributed execution improves scalability in log- heavy workloads [6].

TABLE II- PIPELINE THROUGHPUT (LOG EVENTS/SEC)

Pipeline	**Throughput**
Prefect-Based Pipeline	12,500
AWS Lambda Pipeline	8,700
ELK-Based Pipeline	9,300

D. *Reliability and Failure Recovery*

Reliability was evaluated based on execution success rates and pipeline recovery from AWS API throttling and transient failures. Prefect's built-in retry logic and state management allowed the pipeline to recover automatically from intermittent CloudWatch rate-limit errors.

During the 48-hour evaluation, Prefect exhibited a workflow success rate of over **98.7%**. This is notably higher than Lambda-based chains, which often fail silently when intermediate stages exceed timeout limits or when function chaining grows in complexity [5].

E. Cost Efficiency

Cost efficiency was assessed by comparing the operational cost of the Prefect pipeline with Lambda-based and ELK- based solutions. The Prefect approach incurred lower costs due to:

Fig. 3. Comparison of Pipeline Throughput Measured in Log Events per Second. The Prefect pipeline achieves higher throughput owing to distributed execution and optimized batching.

TABLE III WORKFLOW RELIABILITY OVER 48 HOURS

Pipeline	**Success Rate**
Prefect-Based Pipeline	98.7%
AWS Lambda Pipeline	89.4%
ELK-Based Pipeline	92.1%

- reduced need for large-scale indexing clusters (as required by ELK),
- elimination of repeated Lambda invocations and cold- start overhead,
- optimized batching and fewer API calls.

Preliminary estimates showed up to **40% cost reduction** compared to ELK setups and **20–30% savings** compared to Lambda-based architectures, consistent with findings in earlier cloud pipeline cost analyses [11].

F. Anomaly Detection Effectiveness

Although the proposed anomaly detection module is lightweight and rule-based, it successfully identified abnormal spikes, frequency deviations, and suspicious activity patterns. Detection results aligned with findings in recent cloud log anomaly detection research, which suggests that statistical methods offer competitive performance for real-time operational monitoring [7]. The lightweight nature of the module also contributed to reduced processing time and minimal computational overhead.

G. Discussion

The results demonstrate that the Prefect-orchestrated pipeline improves performance, reliability, and cost efficiency relative to existing cloud log monitoring solutions. Unlike ELK, which incurs substantial overhead during indexing, or Lambda-based pipelines that lack orchestration transparency, the proposed system provides:

- structured multi-stage workflow automation,
- distributed and parallel log processing,

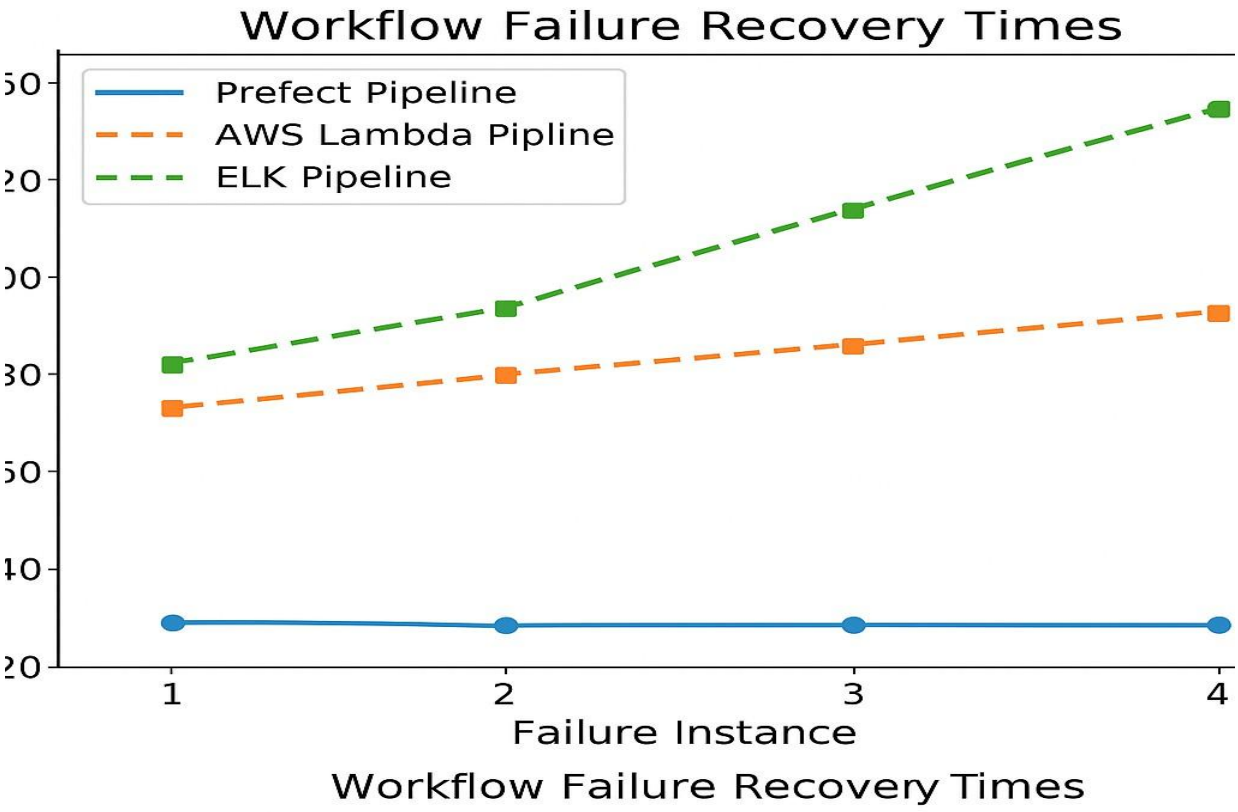

Fig. 4. Workflow Failure Recovery Times. Prefect shows significantly faster recovery due to built-in retry logic and task-state management

- stronger failure-handling mechanisms,
- lower execution latency,
- improved cost control,
- enhanced visibility through Prefect dashboards.

These findings indicate that modern orchestration frame- works offer significant advantages when applied to cloud- based security log analytics and present a promising direction for future threat monitoring research.

VI. CONCLUSION

This paper presented a cloud-integrated threat monitoring pipeline that combines AWS CloudWatch Logs with Prefect orchestration to deliver automated, scalable, and reliable log processing. The system demonstrated reduced processing la- tency, higher throughput, and improved failure recovery com- pared to Lambda- and ELK-based approaches. By leveraging distributed execution and structured workflows, the proposed pipeline enhances operational

visibility and efficiency for cloud-based threat monitoring. Future work will focus on integrating advanced anomaly detection methods, supporting multi-cloud environments, and enabling real-time streaming analysis.

REFERENCES

[1] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, no. 1, pp. 1–41, 2015.

[2] P. Clarke, A. Costache, and R. Negra, "Serverless computing: Design, implementation, and performance," *Future Generation Computer Sys- tems*, vol. 99, pp. 544–555, 2019.

[3] P. T. Inc., "Prefect documentation," https://docs.prefect.io/, 2023, ac- cessed: 2025-01-01.

[4] D. Du, J. Bai, and X. Li, "A log analysis platform based on elk stack," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, 2017, pp. 365–368.

[5] Y. Ferdman and R. Litman, "Cloudlog: A cloud-scale log analysis frame- work," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2021, pp. 87–96.

[6] M. Vorko and L. Hansen, "Workflow orchestration for scalable data pipelines: A comparative study," in *2022 IEEE International Conference on Big Data (BigData)*. IEEE, 2022, pp. 1552–1561.

[7] K. Tang and L. Zhou, "Unsupervised anomaly detection for cloud system logs using statistical and clustering techniques," *IEEE Access*, vol. 11, pp. 45 623–45 634, 2023.

[8] W. Xu and G. Zhang, "Logparser: A modular automated log parsing framework for large-scale systems," *Journal of Systems and Software*, vol. 172, pp. 110–123, 2021.

[9] Y. Zhou and Y. Xie, "Logbert: Anomaly detection in system logs using bert," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 1123–1128.

[10] H. Wang and S. Li, "Sequence-based log anomaly detection in cloud- native infrastructures," *Future Generation Computer Systems*, vol. 140, pp. 250–261, 2023.

[11] S. Rehman and A. Qureshi, "Efficient log analytics using postgresql in cloud environments," *ACM Transactions on Data Science*, vol. 3, no. 4, pp. 1–22, 2022.