

Original Article

AI-Driven Incident Detection and Automated Remediation for Cloud-Native Applications

Uday Allala

Lead SRE Engineer, USA.

Received: 08-05-2020

Revised: 08-25-2020

Accepted: 09-04-2020

Published: 09-08-2020

ABSTRACT

Cloud-native applications are built with distributed micro services, containers, orchestration platforms and dynamic infrastructure and incident detection and recovery is becoming more complex. Traditional monitoring and incident management techniques typically rely on predefined thresholds and static rules, which can lead to late detection, excessive false alarms and inability to detect complex and evolving failures. This paper introduces an AI-based incident detection and automated remediation approach for cloud-native apps that combines real-time observability data, machine learning-based anomaly detection, intelligent incident classification, root cause analysis, and automated recovery. It constantly checks for abnormal behavior in the application logs, infrastructure metrics, distributed traces, container events, and service-level telemetry, to define the severity of the incident. Algorithms such as machine learning and deep learning have been added to enhance the ability to detect, and adaptively recognize, new incidents that have not yet been seen. The proposed framework also introduces an intelligent remediation pipeline which correlates detected incidents with identified causes and matches them to appropriate recovery steps such as container restart, service rescheduling, scaling of resources, configuration fix, and rollback of applications. The framework is assessed using different workloads and incident scenarios in a cloud-native experimental environment to compare against traditional rule-based and machine learning-based baseline environments. The accuracy, precision, recall, F1-score, ROC-AUC, detection latency, mean time to detect (MTTD), mean time to remediate (MTTR) and remediation success rate are used to measure the performance. The experimental assessment verifies how effective AI-driven detection and automated remediation can be in minimizing incident response time, enhance detection accuracy and operational resilience. The study brings an end-to-end approach for intelligent, adaptive and self-healing cloud-native operations.

KEYWORDS

Cloud-Native Applications, Artificial Intelligence, Incident Detection, Automated Remediation, Anomaly Detection, AIOps, Kubernetes, Machine Learning, Predictive Analytics, Self-Healing Systems.

1. INTRODUCTION

Cloud-native applications have revolutionized today's software delivery through the use of micro services, containers, Kubernetes orchestration, dynamic resource management, and distributed service architectures. These technologies enable scalability, flexibility and quick deployments, but come with hefty operational complexity. Increased complexity and an increasing number of incident detection and diagnosis challenges arise when a single user facing failure may be triggered by a combination of multiple services, infrastructure components, network conditions, configuration changes or limitations in resources. [1,2] There are a number of traditional monitoring solutions that depend on static thresholds, rules and manually-tuned alerts and these are often insufficient for detecting complex failure patterns, multivariate conditions and failure patterns that have never previously been encountered. With the growing number and variety of application logs, metrics, traces, and infrastructure logs and events, it becomes even more challenging to separate incidents from transient anomalies. This makes it imperative that cloud operations have intelligent mechanisms that continuously monitor the behavior of systems, detect new incidents, infer probable causes and trigger the right recovery actions to recover with few or no human interaction.

AI can be used to further improve cloud-native operations by learning the normal system behavior, identifying anomalous patterns, categorizing the types of incidents, and informing automated remediation decisions. But, current solutions often focus on one aspect of incident detection, root cause analysis or remediation, without actually establishing an integrated operational process. In this paper, therefore, we present a framework for detecting incidents and automating their remediation using AI that integrates all these components within a single architecture: real-time observability, anomaly detection and classification using machine learning and deep learning, root cause analysis, remediation decision making and automated recovery. The main goals are to enhance the detection accuracy, lower the false alarm rate, minimize the mean time to detect (MTD) and the mean time to remediate (MTTR), and increase the reliability of automated recovery for different workloads and failure situations. The primary features are an end-to-end AI-powered incident-management architecture, an intelligent mechanism to correlate among diverse telemetry to detect incidents and likely root causes, and an auto-remediation pipeline with continual feedback for adaptive cloud operations. The rest of this paper addresses the theoretical background and related work, proposed framework, implementation architecture, experimental methodology, performance evaluation, discussion, limitation and conclusion.

2. BACKGROUND AND THEORETICAL FOUNDATIONS

Application design for the cloud follows loosely coupled micro services, containers, service meshes, and automated orchestration platforms like Kubernetes, making them scalable and capable of continuous deployment. In these settings, services and infrastructure elements are spread out and have many layers of dependency, generating extensive amounts of operational telemetry. [3,4] Containers offer lightweight and reproducible execution environments, and Kubernetes takes care of scheduling, scaling, service discovery, load balancing, and failure recovery. Managing is therefore dependent on an abundance of observability of application logs, infra metrics, distributed traces,

events and service-level indicators. Cloud-native observability is different from the traditional observability, in the sense that it should integrate heterogeneous telemetry from rapidly changing services to give a unified view of system health. Incident detection is the process of recognizing abnormal behavior and categorizing incidents based on their type and severity; anomaly detection methods, such as statistical, machine learning, clustering, auto encoder, and deep learning models can be used to detect deviations from learned normal behavior and detect complex incidents that may not be detected by predefined thresholds.

AI and machine learning have therefore emerged as vital elements of AIOps, which include the ability to analyze vast amounts of operational data, identify patterns across time and multiple variables, anticipate potential failures, and aid in automated operational decisions. Combining anomaly detection with incident classification and root cause analysis, AI can help classify incidents and detect anomalies to differentiate the true service incident from normal workload fluctuations and transients. Automated incident remediation takes this capability a step further by converting a detection result to remediation actions like restarting failed containers, rescheduling workloads, scaling resources, rolling back deployments, or correcting configurations. Self-healing cloud-native systems fill this gap by constantly monitoring application behavior, identifying deviations, performing validated recovery actions and determining if the system is healthy again. This forms a feedback-driven operational model where detection, diagnosis, remediation and verification are carried out in a continuous cycle, laying the theoretical groundwork for the AI-driven framework for incident detection and automated remediation proposed in this study.

3. RELATED WORK AND RESEARCH GAP

Most traditional methods of cloud incident detection are based on thresholds, pre-defined rules, event correlation, and manually configured alerts, which are used to detect service and infrastructure failures. [5] While these techniques may be simple to use, they are not readily adaptable to dynamic cloud-based environments, and create too many false positives or are unable to detect incidents that have not been seen before. Machine learning based techniques enhance this ability by learning patterns from the telemetry data generated during operation, and detecting the anomalous behavior by means of decision-trees, random forests, support vector machines, clustering and isolation models. Auto encoders, recurrent neural networks, LSTM networks and transformer networks have also been used for deep learning techniques to handle high-dimensional and temporal relationships between logs, metrics and traces.

Concurrently, AIOps platforms have incorporated AIOps capabilities such as event correlation, anomaly detection, alert prioritization, and incident management by leveraging machine learning. Many of the current approaches focus mainly on detection and diagnosis and automated remediation is typically done by a predetermined playbook or through individual recovery rules. Predictive incident-management practices try to anticipate the failures before the service becomes degraded, but these methods are dependent on quality data, adaptability of the model and accurate representation of evolving service relationships.

To this end, there remains a key research challenge of creating a single platform that enables a continuous cloud-native operational loop that links real-time incident detection, intelligent classification, root cause analysis, predictive assessment and automatic remediation. Current solutions often view these capabilities as individual units and restrict their autonomy when responding to complex multi-service failures and changing workload requirements. In addition, there has been little focus on integrating the diverse observability data with adaptive AI models and remediation verification processes, while maintaining operational safety and human involvement.

The proposed approach overcomes these limitations by an integrated, AI-based architecture that continuously processes the telemetry data from the cloud-native system, detects abnormal behavior, classifies incident severity, identifies the likely causes of incidents, decides on potential remediation actions, performs recovery operations and checks the resulting system state. The integration is to “enable more accurate detection, lower false alarm rate and response time, shorten MTTD and MTTR, and provide an adaption base for self-healing cloud-based applications.”

4. PROBLEM FORMULATION AND SYSTEM REQUIREMENTS

4.1. Cloud-Native Incident Management Problem

Cloud-native applications run across distributed micro services, containers, orchestration platforms, databases, networks, and dynamically provisioned infrastructure and thus generate complex interdependencies and highly dynamic system states. [6] An incident can be caused by resource exhaustion, application failure, and network degradation, dependency failure of service, configuration error, and abnormal workload behavior. The incident-management problem is, then, defined as being ongoing, as it involves the ability to identify, analyses, prioritise and resolve abnormal system conditions in order to maintain application availability and service-level objectives. To tackle this challenge, the proposed framework includes the incorporation of heterogeneous observability data and the use of AI technology to detect, classify, analyze, and remediate incidents automatically.

4.2. AI-Based Incident Detection and Root-Cause Problem

The incident detection problem is the problem of detecting abnormal states of a system from continually generated metrics, logs, traces, events, and service-level indicators. The AI model is given information X_t at time t , and produces an anomaly score $A_t = f(X_t)$ an incident is detected. Once detected, correlated telemetry and services is analyzed to determine the incident's severity and the most likely cause. The goal is to minimize the number of false alarms and detection latency, and maximize the accuracy, precision, recall and accuracy of root cause identification in dynamic cloud-native environments.

4.3. Automated Remediation and System Requirements

Whereas the remediation problem decides which is the best recovery action to take for each detected and diagnosed incident, such as restarting the container, rescheduling the pods, scaling resources, correcting configuration, rolling back services, redeploying services, etc. The framework

should complete these activities with reliability, taking into account the severity of the incident, the confidence in the cause of the incident, service dependency, operational risk, and current conditions of the system. Real time telemetry processing, scalability, security, availability, low detection latency, high remediation success rate, and continuous verification of recovery are key requirements. The system can ultimately help decrease MTTD and MTTR, maintain application reliability, and prevent unnecessary and unsafe automated interventions.

5. PROPOSED AI-DRIVEN INCIDENT DETECTION AND AUTOMATED REMEDIATION FRAMEWORK

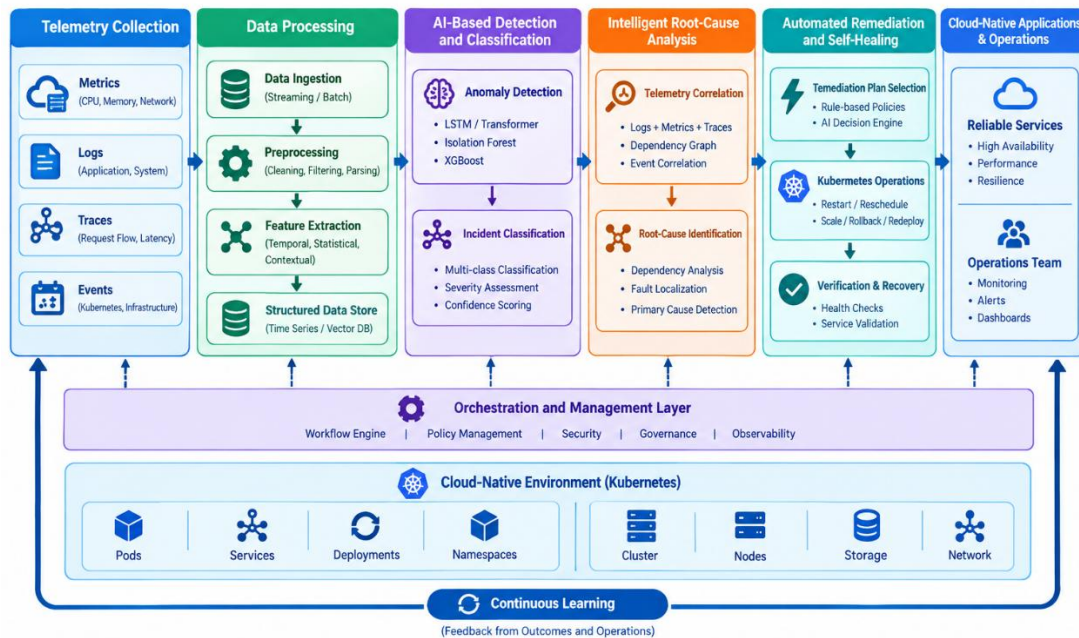


Fig 1 - Proposed AI-Driven Incident Detection and Automated Remediation Framework

5.1. Framework Overview

The proposed solution provides for an end-to-end intelligent pipeline that combines cloud-native telemetry collection, data preprocessing, AI-based incident detection and severity classification, root cause analysis, automated remediation, and continuous feedback. [7] The framework is a closed-loop system where operational data is continually gathered from applications and infrastructure, and is converted into meaningful behavioral attributes and then analyzed to detect abnormal conditions and evaluated to decide on recovery actions. This combined approach allows the system to evolve from a purely monitoring and detecting solution, to identifying, diagnosing, recovering and verifying incidents following remediation.

5.2. Cloud-Native Data Acquisition and Observability

The data acquisition layer automatically extracts various operational data from application logs, infrastructure metrics, Kubernetes events, container states, distributed traces, network telemetry, and service-level indicators. These data sources augment information about application behavior, resource utilization, service dependencies and infrastructure health. A single observability

pipeline takes in telemetry, normalizes it and assigns it timestamps before sending it along to the analytical parts, allowing for the ability to correlate events in real time in a distributed systems. The goal of this layer is to detect isolated failures and complex incidents with multiple cloud-native components.

5.3. AI-Based Incident Detection and Classification

The AI based detection layer provides statistical, machine learning and deep learning analysis of processed telemetry to detect deviations from learned normal operational behavior. [8] The temporal and multivariate behavioral patterns are used to generate anomaly scores and the detected anomalies are also classified based on incident type and severity. Isolation-based algorithms, Random Forest, XGBoost, auto encoders, LSTM networks, transformer-based approaches, etc. can be employed based on the nature of the telemetry. The adaptive thresholds and confidence scores allow the framework to identify changes in workload that are temporary, and not real incidents, and reduce false-positive alerts.

5.4. Intelligent Root-Cause Analysis

The root cause analysis layer ties events that are detected to infrastructure dependencies, infrastructure relationships, temporal patterns, logs, metrics, and distributed traces to find the most likely cause of failure. With dependency-aware analysis, the framework can distinguish primary causes from secondary symptoms, especially in the case of a failure cascading through multiple micro services and infrastructure. The candidate root causes are sorted by their level of correlation to the issues, by their level of anomaly, by dependency relationships, and by the level of model confidence. The final diagnosis feeds into the remediation engine to give it context that helps it determine what recovery action to perform.

5.5. Automated Remediation and Continuous Learning

The automated remediation layer transforms verified diagnoses of incidents into relevant recovery steps that span from container restart to pod rescheduling, scaling resources, rolling back services, fixing configurations, and reed deployment. [9] To minimize the risk of inappropriate intervention, incident severity, root cause confidence, service dependencies and operational policies are taken into account before making a remediation decision.

After execution, the framework checks the health of the system based on new telemetry data and decides if the incident has been solved or not, if the system isn't recovered, other solutions or human escalation may be triggered. The results of remediation are then passed back into the continuous learning mechanism, allowing the AI models and decision policies to learn and get better at detecting, diagnosing and recovering over time.

6. CLOUD-NATIVE DATA ACQUISITION AND OBSERVABILITY

CPU	•Utilization, throttling •CPU overload
Memory	•Usage, allocation, OOM events •Memory exhaustion
Network	•Latency, packet loss •Network degradation
Application	•Error rate, response time •Application failure
Kubernetes	•Pod restarts, scheduling failures •Pod/container failure
Logs	•Error frequency, exception patterns •Application errors
Traces	•Span latency, failed requests •Service dependency failure

Fig 2 - Cloud-Native Telemetry and Incident Features

6.1. Application Telemetry Collection

The application telemetry layer is responsible for gathering runtime data that is needed to model the behavior of cloud-native applications. [10] It collects information about the service response, request rate, error rate, outcome of transactions, dependency health, and application level metrics from individual micro services. These telemetry streams carry context information about the workload to help differentiate normal workload variations from abnormal application behavior and are sent constantly to the downstream observability and AI processing layers.

6.2. Infrastructure, Container, and Kubernetes Metrics

The infrastructure and orchestration monitoring layer aggregates metrics from both virtual machines, nodes, containers, pods and Kubernetes components. Key metrics are CPU/Memory usage, Resource throttling, Disk usage, Network usage, Pod restarts, Scheduling failures, Container health and Node availability. This is done by correlating such measures with application telemetry, which allows the framework to detect incidents caused by infrastructure and to understand the propagation of resource constraints between cloud-native services.

6.3. Logs, Traces, Events, and Real-Time Data Ingestion

All application logs, distributed traces, system events, alerts, and other operational signals are connected via a real-time ingestion pipeline. [11] A log contains information about errors and exceptions; a distributed trace offers information about a request path, service relationships, and latency propagation through micro services; events and alerts offer further temporal detail about changes in the state of the infrastructure and the application. The collected data is normalized, time stamped, converted to structured data e.g. error frequency, response latency, resource usage, restart frequency, failed request rates etc and then passed to the AI based incident detection & root-cause analysis.

6.4. Observability Data Processing, Normalization, and Feature Extraction

Since cloud native telemetry data comes from various sources and formats, the framework is used for pre-processing of data such as data cleaning, handling missing data, noise reduction, timestamp synchronization, normalization, aggregation, and filtering of duplicate events. The processed telemetry is then converted into numerical and categorical attributes of the resource behavior, application performance, service dependencies and failure patterns. These features provide the analytical input to detect anomalies, classify incidents, assess their severity and find root causes, while the consistent representations of these features help make analyses more consistent and scalable across various cloud-native deployments.

7. AI-BASED INCIDENT DETECTION AND CLASSIFICATION

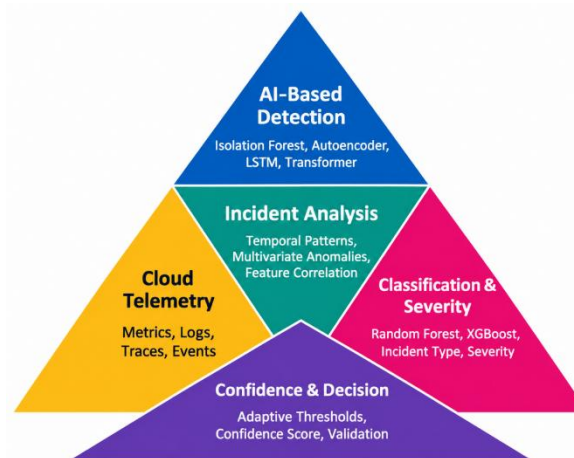


Fig 3 - AI-Based Incident Detection and Classification Workflow

7.1. AI-Based Incident Detection Architecture

The proposed incident detection architecture combines statistical analysis, machine learning and deep learning techniques to detect abnormal behavior from a variety of cloud-native telemetry. [12] Processed metrics, logs, traces and events are converted to behavioral features and analyzed to return anomaly scores and incident predictions. Isolation Forest is able to detect an outlier that is isolated from the rest of the data, whereas Random Forest and XGBoost can be used for incident classification, and auto encoders, LSTM networks, and transformer-based models can be used to detect complex nonlinear and temporal patterns. Overall, the detection architecture enables adaptive detection of known and unknown incidents and real-time cloud-native operations.

7.2. Machine Learning and Deep Learning-Based Detection

Machine learning and deep learning models are used depending on type of telemetry and incident pattern collected. Isolation Forest offers unsupervised anomaly detection and does not need a lot of incident labels to classify detected events, while Random Forest and XGBoost classify detected events based on a number of actions. The advantage of the auto encoders is that they detect deviations in the reconstruction of the high-dimensional telemetry and the advantage of the LSTM and transformer-based models is that they are able to interpret the temporal dependence and the

long-range operation patterns. This pairing provides the framework with better detection capability for abnormal traffic, application errors, service failures and complex multivariate incidents as well as resource anomalies.

7.3. Temporal and Multivariate Anomaly Detection

Incidents often start as a series of events and changes over time and multiple services, rather than as a single event. The proposed framework will therefore focus on break down the temporal order and multi-variate correlation among CPU utilization, memory usage, latency, error counts, pod activity, network conditions and service dependences. LSTM models and transformer-based models are capable of capturing the temporal dependencies, whereas multivariate anomaly detection detects coordinated deviations in several dimensions of telemetry data. This provides better signal of when things are slowly getting worse, cascading into failure, or when incidents are complex and the single metric threshold doesn't pick them up.

7.4. Incident Classification and Severity Assessment

After anomaly detection, the framework categorizes the incidents based on their type, affected component and operational severity. [13] The classification models leverage the characteristics of the telemetry and additional context to classify incidents of CPU overload, memory exhaustion, pod failure, network degradation, application error, and service dependency failure. A severity assessment mechanism prioritizes anomalies, based on anomaly size, service impacted, service level objective violations, duration, and user impact, as low, medium, high, and critical. This classification and severity information will direct the following manual and automated Root Cause Analysis and corrective action.

7.5. Adaptive Thresholding and Detection Confidence

The framework features adaptive thresholds and confidence scoring that helps to enhance detection accuracy in the presence of varying workloads. Detection boundaries are not set by a fixed threshold, but rather the boundaries are adjusted dynamically based on historical behavior, workload variation, temporal patterns and recent system conditions. A confidence score is given to each incident detected, and is based on how well the model agrees with the incident, how anomalous the incident is, how consistent the features are and how much evidence there is in the context. Incidents with high confidence can be passed straight to automated diagnosis and remediation, while uncertain situations can be validated before or passed to a human for evaluation, minimizing nuisance and unsafe automated actions.

8. INTELLIGENT ROOT-CAUSE ANALYSIS

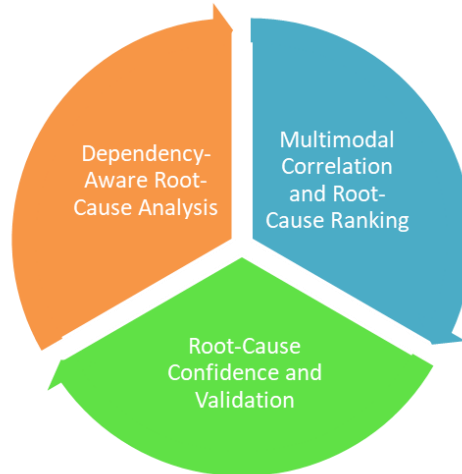


Fig 4- Intelligent Root-Cause Analysis

8.1. Dependency-Aware Root-Cause Analysis

The proposed root-cause analysis mechanism will be able to determine the root cause of a detected incident by examining the relationship between micro services, containers, Kubernetes resources, Infrastructure nodes, databases and the network components. [14] Detected anomalies are correlated with service dependencies and temporal execution patterns, to differentiate between primary failures and secondary symptoms. An incident dependency graph is built to show the affected components and the relationships between them, allowing the framework to follow the propagation of failure through the cloud-native environment and determine which the most closely connected components were to the incident that was experienced.

8.2. Multimodal Correlation and Root-Cause Ranking

The framework relates metrics, logs, distributed traces, events and alerts to gain a complete picture of incident behavior. Temporal correlation detects events that happen in the given time frame, and cross-source correlation correlates resource anomalies, application errors, failed requests and service-level degradation. These are then given a rank based on anomaly magnitude, dependency relationships, and their proximity in time, their level of correlation, and historical patterns of incidents. This helps the system focus on the most likely cause of a problem instead of looking at the events that correlate with each other and identifying these as failures.

8.3. Root-Cause Confidence and Validation

A confidence score is assigned to each identified root cause and is based on consistency and strength of evidence from the multiple telemetry sources and service dependencies. High-confidence diagnostics can move forward to the automatic remediation phase, while low-confidence diagnostics can be further analyzed or validated by humans and then corrected. After remediation the framework confirms the diagnosis by checking whether the corresponding anomaly conditions go away and the health indicators get back to a normal state. Any inappropriate remediation actions

decrease due to this validation mechanism and feedback is given to enhance future root cause identification.

9. AUTOMATED REMEDIATION AND SELF-HEALING MECHANISM

9.1. AI-Based Remediation Decision and Action Selection

The proposed remediation mechanism identifies the best recovery action, depending on the type and severity of the incident, the root-cause confidence, affected service, [15] dependency relationships and the current state of the system. The actions are linked to pre-defined actions or those recommended by AI, including: container restart, pod rescheduling, resource scaling, service rollback, configuration correction and redeployment. The decision engine prioritizes actions based on the expected effectiveness of recovery and operational risk, allowing the framework to select remediation actions based on the context instead of just static playbooks.

9.2. Kubernetes-Based Automated Recovery

The remediation engine works in conjunction with Kubernetes to run recovery actions within the Kubernetes cloud-native environment. The system can perform rerun of unhealthy containers, rescheduling of failed pods, scale up and down of application resources, replacement of unhealthy workloads, and service redeployment or rollback procedures depending on the diagnosed failure. Dependencies are taken into account for resources and services before they are launched to prevent cascading effects and ensure the availability of the applications. This automation based on Kubernetes allows for quick recovery from common infrastructure or application events while minimizing manual involvement and operational response.

9.3. Remediation Verification, Recovery, and Human Escalation

Once a recovery action is completed, the framework continues to monitor new metrics, logs, traces, Kubernetes events and service-level indicators to see if the incident has been resolved. The successful recovery is evidenced that abnormal state is removed and appropriate health parameters are returned to acceptable operating ranges. The system may also trigger an alternate recovery action or pass the incident up to an operator if the remediation fails, actions taken are out-of-scope or the defined risk limits are breached. Low-confidence diagnoses, critical services and high-risk actions have human in-the-loop controls as another safety net in the autonomous self-healing process.

9.4. Continuous Learning from Remediation Outcomes

The framework documents remediation actions taken, outcomes of remediation, recovery time and post-recovery system performance as feedback to enhance future incident-management decisions. [16] Actions that are successful enhance the correlation of incident conditions and effective recovery actions; actions that are unsuccessful contribute information to be used to refine the process of remediation actions and model predictions. This feedback mechanism allows the system to gradually learn and adjust to varying workloads, service configurations, and failure patterns, ensuring more accurate and reliable autonomous operation.

9.5. Security and Controlled Automated Remediation

To avoid recovery processes that are not appropriate or authorized, automated remediation is controlled by authorization policies, action privileges, service criticality and operational safety constraints. Every remediation action is checked against pre-defined policies before it is performed, and if necessary, some sensitive operations like configuration changes, recovery actions for databases, or for production rollback can need a separate approval. Audit records will document the incident that was detected, root cause determination, chosen action, action performed and recovery outcome, for traceability and accountability related to operational decisions made using AI.

10. PROPOSED SYSTEM ARCHITECTURE AND IMPLEMENTATION

10.1. Overall System Architecture

The proposed system architecture involves the collection of data through telemetry, its real-time processing, and analysis of incidents using AI, identification of the root cause, and the automatic remediation process in a cloud-native operational pipeline. [17] The application and infrastructure telemetry is gathered from micro services, containers, k8s resources, logs, metrics, traces and events, and the data is then streamed and feature-engineered. The resulting data are analyzed by AI/ML models to detect incidents, classify their severity, and identify probable root causes. The incident-management engine then orchestrates remediation actions and interacts with the remediation orchestrator to perform recovery actions via Kubernetes, which are then continually verified and fed back.

10.2. Streaming, AI/ML Processing, and Incident Management

The streaming and processing pipeline enables real-time ingestion, normalization, aggregation and feature extraction of heterogeneous observability data. Processed telemetry provided to AI/ML components for anomaly detection, incident classification, and temporal pattern analysis, as well as root cause inference. Model-serving components deliver low-latency inference, and the incident-management engine integrates model outputs with operational policies, service dependencies, and other factors to decide if the incident is a legitimate one. It allows for continuous analysis of high volumes of telemetry data, reducing the delay between the occurrence of incidents and their arrival for analysis and diagnosis.

10.3. Remediation Orchestrator and Kubernetes Integration

The remediation orchestrator takes validated incident diagnostics and converts them to recoverable actions and links directly into Kubernetes to carry out automated corrective actions. [18] The orchestrator can restart containers, reschedule pods, scale resources, change configurations, roll back deployments, and redeploy impacted services depending on the nature of the incident. Execution is governed by authorization policies, severity thresholds and remediation rules to stop unsafe actions from taking place. Once an operation completes, updated application and infrastructure telemetry is analyzed to find out the status of the application health – either restored or escalated.

10.4. Continuous Feedback, Scalability, and Fault Tolerance

A continuous feedback system has been included, and it records the outcomes of detection, root cause decisions, recovery actions, recovery outcomes, and the performance of the system to be used to improve the model in the future. The deployment architecture is designed to provide horizontally scalable processing and model serving components, allowing for a growing amount of telemetry and greater workload without appreciable loss of detection performance. Fault-tolerant processing, redundant components, health monitoring and recovery mechanisms are built in to avoid failures in the incident-management system. With this design, the framework can operate differently in diverse and large cloud-native environments, while keeping the AI operation reliable.

11. EXPERIMENTAL METHODOLOGY

11.1. Experimental Objectives

The challenge is to validate the proposed AI-driven incident detection and automated remediation framework through experimentation, under various cloud-native workload and failure scenarios. [19] The main goals are: Incident Detection accuracy, Classification Reliability, Root Cause Identification, Remediation Success, and Operational Response Time. Special attention is given to false positive rates, mean time to detect (MTTD), and mean time to remediate (MTTR), in comparison to traditional monitoring and baseline AI-based monitoring.

11.2. Experimental Environment and Cloud-Native Tested

The experimental platform is an on-demand cloud environment running on Kubernetes with containerized micro services, application services, databases, networking resources, and dynamically provisioned compute resources. The proposed telemetry pipeline collects and processes observability data like application logs, infrastructure metrics, Kubernetes events, and distributed traces on a continuous basis. Variations of workload intensity and controlled failure injection are performed to test the system's behavior in normal, degraded, and incident conditions, measure computational resource utilization, and process throughput.

11.3. Dataset and Incident Scenarios

The experimental data set is comprised of normal and anomalous cloud-native operational telemetry generated from the tested and annotated with information on incident type, incident time, cloud component impacted, severity, and recovery. Common scenarios of incidents are CPU overload, memory exhaust, pod crash, network latency, service failure, database connection failure, application error, and container resource exhaustion. The dataset is split into training, validation, and testing sets, with the AI models assessed on patterns of incidents that include those that have occurred in the past as well as those that haven't.

11.4. Baseline Methods and Proposed AI Model Configuration

The proposed framework is assessed against typical baseline techniques such as rule-based monitoring, statistical anomaly detection, traditional machine learning, and deep learning-based detection. [20] The proposed method integrates anomaly detection, incident classification, temporal

analysis and root cause inference with appropriate models including: Isolation Forest, Random Forest, XGBoost, auto encoders, LSTM, and transformer based models. The training and validation data are used to optimize the model parameters, and the test set is used to evaluate the detection, diagnosis, and remediation performance of the model in an unbiased way.

11.5. Evaluation Metrics

Accuracy, precision, recall, F1-score, ROC-AUC, false positive rate and false negative rate are used to evaluate performance in incident detection and classification. The accuracy and diagnostic reliability are known as the root cause accuracy, and the percentage of incidents that can be successfully resolved with automated actions is known as the remediation success rate. The parameters used to evaluate the operational performance are detection latency, MTTD, MTTR, recovery time, system throughput, CPU utilization, and memory utilization. All of these metrics combine to indicate not only the smartness of the proposed framework but also its value in bringing down cloud-native incident response and recovery time.

12. RESULTS AND PERFORMANCE EVALUATION

12.1. Overall Incident Detection Performance

Table 1: Incident Detection Performance Comparison

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
Rule-Based	78.4	74.8	71.6	73.2	76.5
Statistical	82.7	80.5	78.9	79.7	81.8
ML-Based	88.6	87.2	85.9	86.5	89.1
Deep Learning	92.1	91.4	90.2	90.8	93.0
Proposed AI Framework	95.8	95.1	94.6	94.8	96.4

The overall evaluation shows that the proposed AI-based framework works well in detecting cloud-native incidents under various conditions of applications and infrastructure. The framework is evaluated by accuracy, precision, recall, F1-score, and ROC-AUC, with a focus on its effectiveness in differentiating actual incidents from typical workload variations. The framework proposed is likely to offer a more balanced performance of detection than the rule-based, statistical or individual machine-learning based approaches, thanks to the integration of heterogeneous telemetry, adaptive anomaly detection, incident classification and contextual analysis.

12.2. Comparison with Baseline Detection Methods

The proposed framework is compared with rule based, statistical, machine learning based or deep learning based approaches to assess the relative detection capability of the proposed framework. For comparison, the accuracy, precision, recall, F1-score and ROC-AUC are taken into account, on the one hand to look at the reliability in which the detection takes place, and on the other hand to investigate the false-alarm behavior. The AI framework comes built-in, and is intended to be much more effective than traditional methods by leveraging multiple telemetry sources and adaptive detection and incident correlation, leading to greater recognition of complex and novel cloud-native failures.

12.3. Root-Cause Identification and Detection Latency

The effectiveness of root cause identification is reported as the percentage of incidents identified which are the root cause failure of the component or service. The analysis takes into account dependency relationships, correlated logs, metrics, traces and temporal patterns of incidents to differentiate root causes from secondary symptoms. Detection latency is also measured from the time an abnormal condition occurs to its detection by the AI engine, which can prove the feasibility of the proposed architecture for detecting abnormal conditions in real time and facilitating quick responses to operations.

12.4. Automated Remediation and MTTD/MTTR Analysis

Remediation success rate, recovery time, mean time to detect (MTTD), and mean time to remediate (MTTR) are used to measure automated remediation performance. The proposed framework will create a reduction in detection times as well as recovery times since the diagnosed incidents will be automatically correlated to the appropriate actions in Kubernetes (container restart, pod rescheduling, scaling, rollback, redeployment, etc.). The operational advantage of the AI-based diagnosis and integration of automated recovery and automated post-remediation verification is illustrated by comparing with manual and rule-based approaches.

12.5. Performance under Different Workloads and Resource Conditions

The architecture is tested for scalability and operational efficiency as load increases and the amount of telemetry grows. The system throughput, detection latency, CPU utilization, memory utilization, remediation response time and recovery performance are measured under various workloads to see if the system can perform consistently when more workloads are added to it. This analysis indicates that this framework does not hinder the viability of large scale cloud-native environment and its capacity to support reliable incident detection and automated remediation with minimal computational overutilization.

13. DISCUSSION

13.1. Effectiveness of AI-Based Incident Detection

The proposed detection mechanism is an AI based method that uses heterogeneous and multivariate cloud-native telemetry to enhance incident identification, not just static thresholds. By incorporating machine learning and deep learning models, the framework can identify temporal patterns, irregular resource usage, application failures, and intricate incident patterns. This enhanced accuracy, recall, F1-score and ROC-AUC suggests that AI-powered analysis can significantly decrease false alarms while ensuring accurate detection under varying conditions of incidents.

13.2. Impact of Root-Cause Analysis

Dependency-Aware Root Cause Analysis enhances incident management by helping to identify the primary failures and secondary symptoms of distributed services. Logs, metrics, traces, Kubernetes events and service dependencies can be correlated to give more context to the identification of the component that caused an incident. The ability to accurately identify the root

cause allows the remediation engine to pick more suitable recovery actions, resulting in less unnecessary intervention and making the automated incident response more reliable.

13.4. Effectiveness of Automated Remediation

The ability to automatically link incident diagnosis to validated recovery actions greatly enhances operational response. Incidents are solved without manual intervention, through various Kubernetes-based operations, including container restart, pod rescheduling, scaling resources, rollback and redeployment. The remediation success rate and recovery-time results highlight the benefits of intelligent incident analysis in conjunction with controlled automation, especially where frequent operational incidents and recurring failures are concerned.

13.5. Reduction in MTTD, MTTR, and Operational Overhead

Continuous detection, root cause and automated recovery play a role in MTTD and MTTR reduction. Continuous telemetry processing allows incidents to be identified at an earlier stage and automated action selection removes a number of manual actions from the recovery process. The reduction in response/recovery time increases service availability and operational resilience, resource utilization and throughput measures can be used to see how much computational overhead is needed to keep the intelligent incident-management pipeline going.

14. LIMITATIONS AND FUTURE RESEARCH

The proposed framework still presents a number of limitations in terms of diversity of datasets, model generalization, and complex cloud-native failure conditions. The experimental evaluation may not cover all the incidents that could be experienced in an extensive production environment, especially those events that are rare and involve complex cascading failures of several services, layers of infrastructure, and external dependencies. The generalization of AI models can also be limited for other applications with different architectures, workloads, telemetry characteristics and operational policies. Also, explain ability is a difficulty as the complex deep-learning models can often offer little transparency about why a specific incident or root cause was detected. Another operational risk with automated remediation is when the diagnosis confidence is low or when a wrong recovery action will impact dependent services.

Going forward, the framework will be expanded using generative artificial intelligence for context-based incident reasoning, autonomous root cause analysis and reinforcement learning-based remediation policies. Generative AI can be used to interpret logs, traces and incident histories in natural language without human intervention, and reinforcement learning can be used to learn from recovery results and dynamically prioritise effective actions via a remediation policy. There are also significant opportunities for further investigation in multi-cloud and edge-cloud deployments, which entail distributed dependencies, heterogeneous infrastructure, and different operational policies, increasing the complexity of the system. Such extensions can increase the adaptability of models, their independence in decision making, explain ability, and ability to self-heal safely in the context of a large-scale cloud-native environment.

15. CONCLUSION

15.1. Summary of the Proposed Framework and Major Findings

The proposed incident detection and automated remediation system is an AI-based solution that offers a holistic solution for handling incidents within complex cloud-native applications. The framework integrates heterogeneous telemetry collection, AI-powered anomaly detection, incident classification, dependency-aware root cause analysis and Kubernetes based automated remediation within a single operation loop for continuous detection and recovery. The evaluation framework shows how using intelligent detection and automated recovery systems can lead to better incident identification, fewer false alarms, and quicker response/recovery than traditional systems.

15.2. Contributions and Final Remarks

An end-to-end architecture is the primary contribution of this study which links incident detection, diagnosis, remediation, verification, and continuous learning to ensure reliable self-healing cloud-native operations. The framework offers a starting point to lower MTTD and MTTR, while remaining scalable, operationally controlled, and remediation safe. In summary, the proposed approach showcases the capabilities of AI-powered automation to shift the paradigm in cloud-native incident management from reactive monitoring to intelligent, adaptive, and increasingly autonomous operational resilience.

REFERENCES

- [1] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015, April). Large-scale cluster management at Google with Borg. In Proceedings of the tenth european conference on computer systems (pp. 1-17).
- [2] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, omega, and kubernetes. Communications of the ACM, 59(5), 50-57.
- [3] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. Present and ulterior software engineering, 195-216.
- [4] Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015, September). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In 2015 10th computing colombian conference (10ccc) (pp. 583-590). IEEE.
- [5] Hochenbaum, J., Vallis, O. S., & Kejariwal, A. (2017). Automatic anomaly detection in the cloud via statistical learning. arXiv preprint arXiv:1704.07706.
- [6] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.
- [7] Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 93-104).
- [8] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In 2008 eighth ieee international conference on data mining (pp. 413-422). Ieee.
- [9] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
- [10] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
- [11] Graves, A. (2012). Long short-term memory. Supervised sequence labelling with recurrent neural networks, 37-45.
- [12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30(1), 123-135.
- [13] Du, M., Li, F., Zheng, G., & Srikumar, V. (2017, October). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security (pp. 1285-1298).

-
- [14] He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017, June). Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE international conference on web services (ICWS) (pp. 33-40). IEEE.
 - [15] Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., ... & Zhou, R. (2019, August). Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Ijcai (Vol. 19, No. 7, pp. 4739-4745).
 - [16] Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., ... & Zhang, D. (2019, August). Robust log-based anomaly detection on unstable log data. In Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering (pp. 807-817).
 - [17] Wang, P., Xu, J., Ma, M., Lin, W., Pan, D., Wang, Y., & Chen, P. (2018, May). Cloudranger: Root cause identification for cloud native systems. In 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 492-502). IEEE.
 - [18] Wu, L., Tordsson, J., Elmroth, E., & Kao, O. (2020, April). Microrca: Root cause localization of performance issues in microservices. In IEEE/IFIP Network Operations and Management Symposium (NOMS).
 - [19] Dang, Y., Lin, Q., & Huang, P. (2019, May). Aiops: real-world challenges and research innovations. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 4-5). IEEE.
 - [20] Li, Y., Jiang, Z. M., Li, H., Hassan, A. E., He, C., Huang, R., ... & Chen, P. (2020). Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(2), 1-24.
 - [21] Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., ... & Shanbhag, C. (2010). Dapper, a large-scale distributed systems tracing infrastructure.
 - [22] Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2), 74-80.
 - [23] Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). Site reliability engineering: how Google runs production systems. "O'Reilly Media, Inc."