International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Advancements in Automated Incident Management: A Survey within Cloud-Native SRE (Site Reliability Engineering) Practices

Pooja Chandrashekar*

Independent Researcher

Received 01 Dec 2023, Accepted 24 Dec 2023, Available online 26 Dec 2023, **Vol.13, No.6 (Nov/Dec 2023)**

Abstract

Cloud-native environments that are central to modern infrastructures have, to a large extent, increased the complexity of system management, thereby making the traditional methods of incident management less efficient in terms of reliability and scalability. SRE as a technology to a large extent has been very influential in bringing together the elements of automation, observability, and AI in ensuring the delivery of essential services across the network of distributed systems. This article reviews the transition to cloud-native SRE automated incident management solutions featuring predictive incident detection, intelligent alert correlation, and self-healing mechanisms. Through AI and ML, the SRE team is enabled to uncover anomalies before they happen, eliminate noise, perform root cause analysis, and execute the recovery process in an automated manner over hybrid and multi-cloud setups. Observability integration through Prometheus, Grafana, and ELK allows for real-time system monitoring and performance tuning. In general, the transition to automated and AI-involved strategies has been instrumental in the decrease of MTTD and MTTR resulting in an increase of the scalability, reliability, and resilience of cloud-native operations. The research also points to the importance of feedback-driven learning models in proactive recovery enhancement. These innovations at the core of operational resilience capabilities serve as a vehicle for the next level of autonomous, intelligent SRE ecosystems.

Keywords: *Cloud-Native Systems, Site Reliability Engineering (SRE), Automated Incident Management, Observability, Mean Time to Resolution (MTTR, AI-Driven Analytics.*

1. Introduction

Site Reliability Engineering (SRE) is a discipline that attempts to solve the problem of scaling, reliability, and efficiency of the system by using the methods of software engineering to infrastructure and operations [1]. SRE, which started at Google in 2003, has now been adopted by different sectors that are in need of very high system availability and performance. SRE is often seen as a feasible implementation of the DevOps concept, which among other things, features the use of automation, setting reliability goals as measurable entities, observability, and being risk management in a proactive manner [2]. At the core, the SRE work focus on creating Service Level Indicators (SLIs) along with Service Level Objectives (SLOs), figuring out capacity needs, and also running a well-structured change management process that, if seen together, signify a method of ensuring service reliability that can both be quantified and kept for a long time.

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.13.6.13

Cloud computing systems which are highly distributed and provide on-demand computing, storage, and networking resources to support scalable and dynamic applications are essentially the core of the digital era [3]. Over the past decade, cloud providers have expanded their service portfolios from distributed databases and service meshes to AI-powered platforms, thus, they are not only changing the technical infrastructures but also the development methodologies [4]. Modern software engineering has become a part of agile and DevOps paradigms which allow continuous integration, testing, and deployment (CI/CD) of services several times a day. Such a continuous delivery model, hence, has deployment velocity to a large extent, however, it has also contributed to the operational complexity, thus, manual incident management has become practically impossible.

Microservices architectures along with containerized workloads are typical examples of the cloud-native world. Container orchestration tools like Kubernetes handle compute, storage, and networking across clusters in an efficient way, hence they are compatible with the modular design principles of

microservices [5]. The maintenance of such configurations necessitates nonstop monitoring, scaling which is adaptive, and remediation which is automated to be able to preserve reliability when there are variable workloads. That is the main reason why automation has become a vital part of SRE operations.

Automated Incident Management (AIM) is the following stage of reliability engineering, which incorporates the usage of artificial intelligence, machine learning, and automation for the handling of incidents without any human intervention [6]. Nowadays, enhancements in AIM mostly focus on intelligent alerting, automated root cause analysis (RCA), anomaly detection, predictive maintenance, and self-healing methods. These capabilities are implemented through data-driven observability systems that integrate logs, metrics, and traces to provide detailed information about distributed system behaviors [7]. In addition, AIOps (Artificial Intelligence for IT Operations) usage empowers SRE teams to connect signals, eliminate noise, and automate complex operational workflows at a large scale. AIM is still confronted with issues that it is trying to resolve such as data heterogeneity, model transparency, and finding the right balance between automation and human control, while it is making significant progress [8]. The demand for resilient, self-managing systems is increasing exponentially as more and more organizations are adopting cloud-native architectures and continuous deployment pipelines.

The current study is a thorough evaluation of the innovations, architectures, and mechanisms that drive the automation of incident management as part of cloud-native SRE practices. In addition to mapping out the development of AIM systems, the report also investigates the helpfulness of AI-driven observability and AIOps in order to identify the fresh features, challenges, and research questions that affect the establishment of operations that are self-governing and reliable.

Organization of the Paper

The paper is organized in the following way: In detail, Section II provides an overview of the use of the cloud-based system for incident management in SRE. Section III is dedicated to discussing Observability and Monitoring. The core theme of Section IV is the Exploration of the Cloud-Native Site Reliability Engineering Environment. Section V sums up the review of the literature along with the crucial findings, and Section VI concludes the paper with the author's insights and possible future research directions.

Cloud-Native Incident Management Systems in SRE

The incident management frameworks in SRE that are used in cloud-native environments aim to limit the pretty complicated microservices and containerized architectures that come with these technologies by automation and observability. Usually, these frameworks have monitoring, alerting, and AI-powered analytics that interact to find, in a very short time, the place, the cause, and the solution of incidents without a human agent intervention. It is to say that Prometheus is the tool that collects live metrics, Grafana is there to show in real-time what is going on and Open Telemetry is the one that sends metadata so that a service owner can have the context of an incident [9]. AI-powered predictive analytics and alerting, as it is the case with PagerDuty, eliminate the need for the human intervention and hence, bring more efficient templates for alerting or escalation. Cloud-native SRE incident management lessens or abolishes the time in which a service is not available, decreases the Mean Time to Resolution (MTTR), and enhances the overall system reliability.

The Role of Cloud Infrastructure in SRE.

Cloud infrastructure remains at the centre of Site Reliability Engineering (SRE) operations by offering the required environments that are scalable, resilient, and cost-efficient thus supporting automation, observability, and continuous delivery. The usage of cloud-native techs like containerization, Kubernetes orchestration, and edge computing really helps the system performance since the resources are distributed closer to the data traffic which in turn reduces the latency and increases the reliability. The changes of cloud models have allowed the deployment to be more flexible and the security and maintenance to be the shared responsibilities of providers and users. There are three models that fall under this category: IaaS, PaaS, and SaaS (Figure 1 shows an example of SaaS).

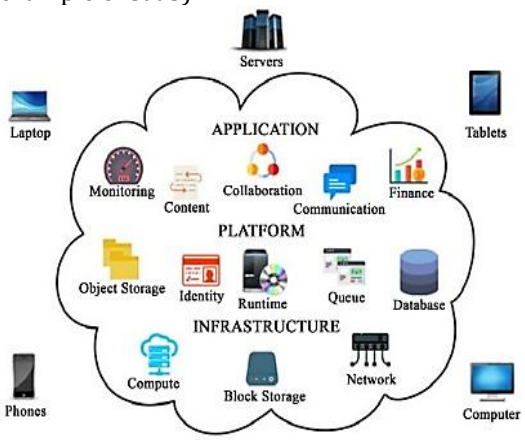


Fig.1 Cloud Computing

Major cloud providers such as AWS, Microsoft Azure and Google Cloud Platform (GCP) are equipped with various capabilities that go hand in hand with SRE principles. Among these features are automated scaling, fault tolerance, monitoring and self-healing mechanisms. AWS works well with global reach and service diversity; Azure is a perfect pick for integration with enterprise systems and hybrid cloud environments; while GCP is considered the best for AI-

driven performance optimization and efficient networking [10]. These platforms are the wells of power for SRE squads to achieve their reliability objectives through automation, proactive incident detection, fast recovery, and data-driven optimization thus making resilient, high-performing, and continuously available cloud-native systems.

Cloud Automated Incident Management In SRE

Automated incident management refers to a system, which is automated and planned, that is built to handle problems in a way that maintains the flow of normal operations. The SRE team employs orchestration technologies for handling responses. This is the reason why automation is so vital.

Digital Resilience: Automated incident management makes an organization's digital resilience stronger. This means that the firm can keep running and adapt when things go wrong, as when an app stops working.

App Availability: Automation makes it easy to discover and fix problems fast, which makes apps more available and improves service levels.

Platform Reliability: Streamlined automated processes make the platform more reliable by promptly finding and fixing problems, which makes sure that every user has the same experience.

Enabling Innovation: Automation frees up the SRE team to spend more time on new ideas and product development by cutting down on the time they spend on repetitive chores. Studies suggest that 72% of teams spend half their time fixing problems instead of coming up with new ideas. Automation helps change that balance.

Excessive Data: Hybrid computing setups generate a massive amount of data from numerous different sources such as ITSM tickets, logs, network data, and alarms. Since this data is scattered and stored in different places, it is difficult to assemble and review event patterns, thus the time for problem resolution is increased.

Complex Architectures: Hybrid cloud, though it provides flexibility, makes the management of incidents more complex [11]. As a result, well-thought-out IT plans have to be in place in order to keep track of and control the systems that are spread over on-premises, cloud, and edge environments, because numerous systems might not have the feature of algorithmically scanning data and automatically recommending remedies for frequently happening incidents.

Post-Incident Review: The process of analyzing and learning from incidents requires, among different things, input from various stakeholders about the cause, impact, and resolution actions of the incident, as well as future mitigation plans. The distribution of computing resources in a cloud-native environment makes it difficult to get all the necessary information and properly assign the ownership for a detailed post-mortem.

AI and Machine Learning in Automated Incident Management

Artificial Intelligence (AI) and Machine Learning (ML) have changed the way automated incident management in Cloud-Native Site Reliability Engineering (SRE) is handled. They have gone from being reactive to proactive, intelligent, and self-healing operations. Predictive incident detection employs time-series forecasting and anomaly detection models to identify failures even before users experience them, thus enabling proactive intervention [12]. Intelligent alert correlation and noise reduction use clustering, graph-based learning, and Bayesian inference to deliver anti-noise capabilities that eliminate the noise of redundant alerts, thus facilitating the listing of significant alert notifications to the SRE team. ML-powered Root Cause Analysis (RCA) localizes the fault automatically by employing dependency graphs, decision trees, and explainable AI models for a rapid origin of the incidents identification. Among self-healing technologies, reinforcement learning methods, such as Deep Q-Networks and policy optimization, allow for the creation of systems that able to perform recovery actions on their own and even make operations more efficient. Finally, Natural Language Processing (NLP) facilitates communication and documentation through the execution of automated incident summary generation, chatbot-assisted ticket triaging, and date extraction (NLP) from system logs or chat. To sum up, developments in AI and ML not only increase the system's reliability but also result in a drop in the meantime to detect (MTTD) and mean time to resolve incidents (MTTR) and make it possible to build adaptive and resilient cloud-native (distributed) infrastructure.

Observability and Monitoring in Cloud-Native Systems**.**

Observability and monitoring are important tools for cloud-native systems that show precise information about distributed architectures. This lets find and fix performance problems before they happen. These measures are the basis of SRE as they guarantee system stability, growth, and ongoing access to the system's operational state.

Data Required for Observability System

The majority of manufacturers who offer observability solutions address the three primary data types—logs, metrics, and traces—at different levels, sometimes referred to as the "three pillars of observability" (as shown in Figure 2). The aforementioned forms of observable telemetry data are by no means exhaustive, but they do account for the vast majority of the data available [13]. It is common practice for several APM companies to brand this telemetry data with their own information:

Logs

The system generates logs, which are strings of text that can be either structured or unstructured,

whenever a particular section of code runs. Any time something happens inside an application, it recorded in a log. To find out how microservice components act in an unexpected or emergent way, can look at their logs. Due to the fact that the majority of application frameworks, languages, and libraries come with built-in logging capability, it is straightforward to create and indicate various severity levels in these logs and instruments [14]. To provide comprehensive information about the system, nearly every part of a distributed system generates logs and events at any given time.

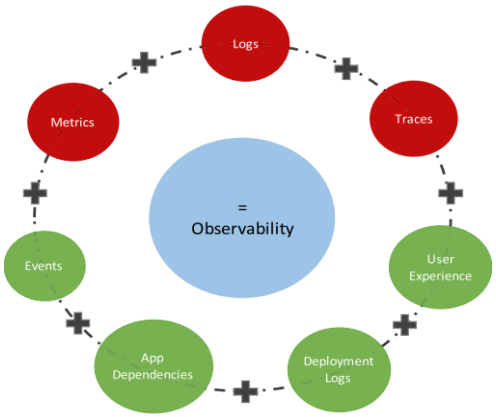


Fig.2 Core Observability Pillars (red) and Additional Vendor-Specific Data (green).

Metrics

A system, service, or network component's overall behaviour over time can be understood by looking at its metrics, which are numerical representations of data used by operations teams and can characterize applications or system performance using metrics, which are sets of attributes (like a timestamp, name, label, and value) that transmit information about service level agreements (SLAs). Also find metrics that are created for specific applications. Metrics are numerical values produced from the performance of the system during runtime, as opposed to an event log [15]. By correlating them across the full infrastructure, from application to network components, operators may have a comprehensive view of system health and performance, which is a huge time-saver. Performance indicators at the application level could include things like the number of instances, average response time, requests processed per second, and more.

Traces

A request or action's trace shows its complete path across a distributed system's components. Logs and metrics are great for figuring out how a single system is behaving and how well it's doing, but they don't really help when trying to comprehend how a request moves across a distributed IT system, like trying to figure out which microservice calls were causally related [16]. Operators can see and comprehend the full request lifecycle across several systems by using the tracing technique. System profiling, especially of containerized apps and microservices, is made possible by traces [17]. Operators can look at trace data to get a sense of how the system is doing generally, find slow places, fix performance issues faster, and focus their optimization efforts on areas that are most important.

Measurements from Telemetry Data

Latency, traffic, errors, and saturation are the four golden signals of observability in Site Reliability Engineering (SRE) that come from three categories of telemetry data: logs, metrics, and traces:

Latency, in other words request service time, is the measure of the shortness of time in which an application responds correctly to a certain request. Consequently, this metric can be used to identify a source of a performance problem in an application, e.g., a bottleneck microservice. Operators should set a success criterion for latency measurements and compare results with requests that do not meet the specified delay threshold in order to identify failed requests [18]. Using this method, they are able to quickly locate the occurrences, find the services that have decreased performance, and make the necessary changes.

The number of requests made to a certain application during a certain time frame is what the traffic, or user demand, refers to. That is why the respective values of traffic given by the operators may differ a lot for various applications. As a matter of fact, they may measure a web application's traffic by means of the following metrics: HTTP requests per second or bandwidth consumption. Keeping tabs on the total number of network chats is one way to keep an eye on traffic in a distributed system. Keeping tabs on application traffic allows operators to see how the app handles surges and prepare for sudden increases in demand.

The rate of unsuccessful requests is monitored by errors. Implicit failures (such as HTTP 200 success responses that do not deliver the necessary content) and explicit errors (such as HTTP 500) are the most prevalent types of mistakes [19]. Requests that exceed the agreed timeout interval result in policy violations. The only surefire way to tell whether something is amiss is to keep tabs on problems, as no system is foolproof. If wants to keep tabs on server and client failures individually, must report them separately. Because of this, the Ops team have a much easier time understanding the various error kinds and quickly identifying the source of any given problem.

An indicator of the demand on a server or network resource is saturation, which is the overall capacity of the system. It is a measure of a service's activity and is frequently used to detect system slowdowns or problems early on. Common indicators of production system saturation include CPU, disk space, and memory utilization. It is recommended to use

measurements that constrain a system's performance for measuring saturation [15]. programs that rely heavily on processors can be managed by CPU load, whereas programs that rely heavily on memory can be managed by memory load.

Implementing Observability Best Practices.

Best practices in implementing observability for cloud-native systems. Figure 3 highlights key SRE best practices, including defining SLAs, SLOs, and SLIs, adopting automation, enhancing observability, fostering collaboration, using error budgets, and ensuring scalability to achieve reliable and efficient cloud-native operations, include.

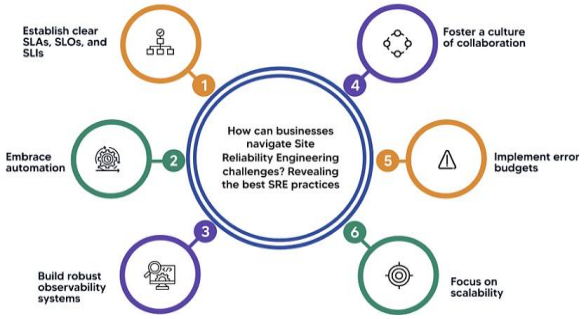


Fig.3 Best Practices for Observability for Cloud-Native

Unified Telemetry Collection: By adopting open standards such as Open Telemetry, consistent instrumentation and data collection are possible across heterogeneous environments that provide interoperability across multiple tools and platforms.
Centralized Data Aggregation: Using scalable observability stacks such as Prometheus–Grafana or ELK (Elasticsearch, Logstash, Kibana) lets put all of monitoring data onto one dashboard, making it easier to see and compare data in real time.
Automated Alerting and Noise Reduction: Smart alerting systems, which are usually supported by AIOps, give a hierarchy to those "important" events that are most likely to have a significant impact. They help with the problem of "fatigue" from too many alerts and make MTTD, or Mean Time to Detect, better.
Context-aware Dashboards and Visualization: Dynamic dashboards give SRE teams the information they need to make decisions and figure out what's wrong when anything goes wrong.
Continuously Feedback and Improvement: Observability is not something do once and forget, it is a process that requires to keep changing the metrics monitor, the logging levels, and the traces that configure, the final outcome being understanding systems more deeply over time.

As a result of the implementation of these steps, companies might build a data-driven culture of trust which is not only reactive but also proactive and adaptable by nature. Such a combination makes it possible for the system to automatically take care of incidents, shorten the time of error source identification, and increase the resilience of cloud-native infrastructures against operational disruptions.
Insights into Cloud-Native Site Reliability Engineering Environment

The pilot implementation revolves around a multi-cloud environment simulation scenario from the real world to verify SRE activities' effectiveness in the case of monitoring, logging, and incident response. The setup is targeting to solving problems of advanced businesses that are using several cloud providers, mainly AWS and Google Cloud.

Incident Detection and Response in Multi-Cloud Environments

A well-integrated, multi-layered approach is required for incident detection in multi-cloud setups. There are a few open-source observability tools that serve this function such as the ELK Stack (Elasticsearch, Logstash, Kibana) and Grafana which are used to gather and visualize performance metrics, application logs, and event traces of various, independent cloud environments [20]. These observability tools transform the SRE teams' local zones into wide-area networks, thus, enabling them to find anomalies and probable service failures in shorter time intervals because the tools have features like cross-cloud correlation/backgrounding as well as alerting.

SRE teams follow defined playbooks and automated incident workflows to maintain uniformity in their reaction to incidents in the different cloud-based environments [21]. Besides that, the monitoring pipeline may include tools like PagerDuty, Opsgenie, and ServiceNow that can be used for the automation and elevation of incident notifications, escalation, and resolution. The deployment of such instruments can drastically shorten the recovery time and make the response to a huge failure better coordinated and, thus, more effective.

Essentially, the use of multi-cloud SRE processes stresses the need for automation, standardization, and interoperability as the main factors leading to operational resiliency. As a result of the combination of observability-driven monitoring practices with automated response practices, organizations notice lower fault detection latencies, better reliability, and more reliable service continuity across very different cloud-based infrastructures.

Monitoring and Logging Tools Integration

A mix of open-source and cloud-native tools were used to keep an eye on the web app:
Prometheus: The Google Cloud Kubernetes cluster has it installed so it can scrape metrics from both cloud environments. Node Exporter and CloudWatch Exporter were used to get AWS EC2 and database metrics.
Grafana: These were the main tools for live display and to offer a single management panel showing the metrics from AWS, GKE, and Prometheus. The custom

alerting rules were created to generate a local alarm upon the crossing of the critical threshold of the key performance indicators (KPIs), like response time, error rates, and CPU utilization.

ELK Stack (Elasticsearch, Logstash, Kibana): The records of a web application, load balancers, and databases were combined to a centralized ELK stack for cross-cloud log analysis. Logstash was set up to consume logs from AWS CloudWatch and Google Cloud's Stack driver.

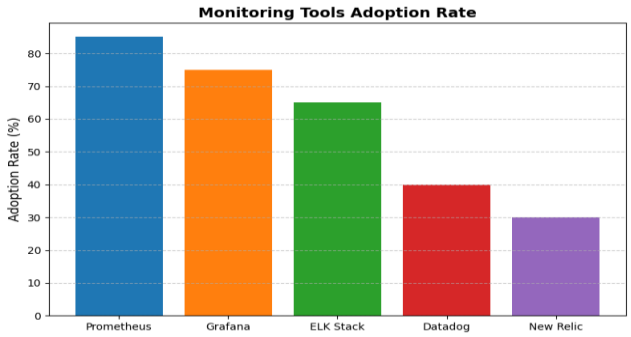


Fig.4 Monitoring Tools Adoption Rate

Figure 4 illustrates that Prometheus is at the forefront with more than 85% utilization, next is Grafana with 75% and the ELK Stack with 65%. Datadog and New Relic are at a lesser adoption level with 40% and 30%, respectively, showing that most users opt for open-source monitoring tools in cloud-native environments.

Resolution and Response Time

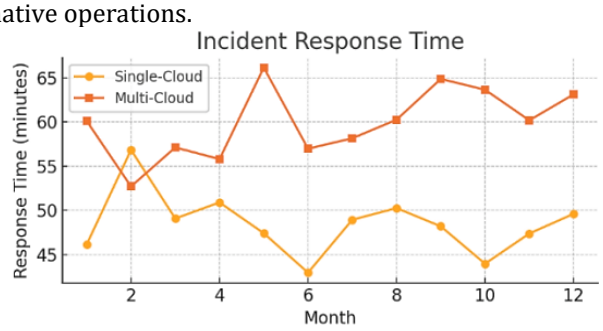
In cloud-native SRE (Site Reliability Engineering) practices, Resolution Time and Response Time are two keys, closely related, but different, performance metrics that, when taken together, outline the operational efficiency of incident management and the overall reliability of services.

Resolution Time: Response Time is the delay that is measured from the time an incident happens until the system or the SRE team acknowledges or reacts to it for the first time. It is the time interval during which the monitoring or alerting systems detect anomalies and issue the corresponding alerts. The response time in cloud-native environments depends a lot on the use of observability tools, real-time telemetry, and automated alerting mechanisms that are integrated across microservices and distributed infrastructures [22]. A low response time shows that the monitoring is very sensitive and that the detection is done in a proactive manner.

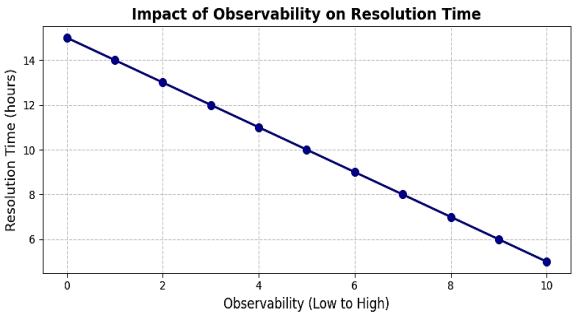
Response Time: Resolution Time is an indicator that tracks the entire time period that starts with the moment an incident is detected and ends with the moment the incident is completely fixed and the service is restored. It reflects the efficiency of the recovery processes that includes analysis of the root cause, decision-making, and implementation. Cloud-native SRE automated incident management systems use AIOps, runbook automation, and self-healing capabilities to greatly shorten the mean time to resolve (MTTR).

Response Time as discussed, is a measure of how quickly a change or incident is detected. Resolution Time, on the other hand, is a measure of how quickly the change or incident is recovered from. Cloud-native architectures that are typically made up of container resented (for instance, Kubernetes), auto-scaling, and service meshes features can facilitate faster responses and resolutions by way of event-driven automation and AI-based diagnostics.

In general, it is necessary to optimize both metrics to be able to achieve high system reliability, less downtime, and a better user experience. In well-developed SRE pipelines, quick response time is a way to rapid anomaly detection, and shortened resolution time is a means to fast, automated restoration, thus, together, they constitute the basis of resilient cloud-native operations.



(a)



(b)

Fig.5 (a) & (b) Show the Resolution & Response Time of the Data in the Study

Figures 5(a) and 5(b) show how cloud deployment strategies and observability influence incident management metrics. The incident response times for single-cloud and multi-cloud environments over a 12 months period are compared by (a), which reveals that single-cloud systems are generally capable of achieving quicker response times, whereas multi-cloud configurations indicate higher fluctuations and longer response durations. While (b) is mainly concerned with the impact of observability on the turnaround time, it also hints at a clear negative correlation: as observability varies from low to high, the incident resolution time decreases accordingly, so it can be inferred that improved monitoring and visibility are the key enablers of quicker problem-solving.

Key Metrics Measured for Cloud SRE

The proposed structure was subjected to a variety of operational efficiency and reliability metrics to ensure that it was completely validated.

- Mean Time to Detection (MTTD): The amount of time it took for the monitoring systems to find the event.
- Mean Time to Recovery (MTTR): The amount of time it takes to fix the problem and get the system back to normal.
- Alert Accuracy: Evaluated based on the accuracy of warnings generated by Grafana Alert Manager.
- Log Aggregation Latency: The ELK stack from AWS and Google Cloud takes a long time to acquire logs and put them together.

Discussion on the Scalability, Reliability, and Adaptability of Automation Models

Cloud-native SRE pipelines for automated incident management have been shown in recent studies over and over again to significantly improve the scalability, reliability, and flexibility of the system as compared to the traditional manual incident management methods. Such architectures as event-driven and microservice-oriented allow the system to be fault-tolerant in a distributed manner and to self-heal in a particular location. Even when workloads grow horizontally, a system can keep working well. Predictive analytics and automated anomaly detection technologies assist keep systems running smoothly by reducing unplanned downtime and service interruptions [23]. Automated incident management systems can spot failure occurrences ahead of time and start automated remedial workflows. This makes services more available and operations more resilient.

ML-driven automated systems are always evolving through feedback loops and hence become better at classifying, prioritizing, and solving incidents gradually. The self-improving features of automation in incident management guarantee that incident management changes along with the transformation of dynamic infrastructures or workloads. There are still challenges in getting smooth scalability at different levels such as multi-cluster and hybrid cloud environments [24]. To illustrate, inefficiency in data synchronization, delay in model retraining, and the inter-dependency of microservice interactions are some of the factors that hinder cross-platform scalability and can thus be considered as bottlenecks in the execution of a certain task.

Literature Review

These literatures highlight major technological moves in mechanized incident handling and cloud-native reliability engineering which mainly features computer-assisted detection, up to date system extensibility, and adherence to regulatory provisions as a means of enhancing system scalability, security, and an altogether more agile kind of incident response. Lin et al. (2022) A cloud-native light-cone model is suggested for analyzing service stack extensibility issues; this model takes a viewpoint view that reflects stakeholder concerns and centres on the application, infrastructure, tenant, and workflow aspects. This approach is used to categorize the different difficulties that arise when creating cloud-native service stacks that may be extended. A comprehensive design and set of critical technologies are developed to address these issues; these include application-specific controllers, cluster bootstrapped construction, unified runtime abstraction, and more. Also, a set of PaaS and SaaS services for container clusters (OMCC), artificial intelligence (OMAI), big data (OMBD), and other related technologies are made available through the implementation of the OM Stack (Oriental Mind Stack) [25].

Li et al. (2021) provide Warden, an incident detection system that operates automatically, as a component of the Incident Management (IMCO) platform. To detect problems from a global perspective, Warden receives warnings from numerous providers. Warden alerts the appropriate on-call engineers for each possible event so that they may set priorities and begin collaborating across teams. When large-scale cloud computing platforms like AWS, Azure, and GCP experience incidents or outages, their availability is significantly reduced. Currently, incident response teams only have a limited picture of the whole system, which leads to challenges like "fog of war" detection, longer mitigation times, and higher financial losses [26].

Ozer et al. (2020) makes an effort to accommodate cloud computing by including an additional step into the conventional incident response procedure. Further, previous research has identified sophisticated cloud computing architecture and scattered data as the primary obstacles to successful cloud incident response; this study offers answers to these problems. For well-known reasons including cost-effectiveness and adaptability, many businesses are moving their infrastructure from on-premises to the cloud. But here's the rub: the reported number of cyber-attacks hasn't gone down, and in fact has gone up phenomenally, even if cloud computing has become more popular among small and medium-sized firms [27].

Joshi, Elluri and Nagar (2020) created a knowledge graph that incorporates all of these data compliance rules and is rich in semantic information. Data dangers and the necessary security procedures tao reduce such risks are part of it. This knowledge graph and the evaluation mechanism built are both detailed in this publication. They checked their knowledge graph with the privacy regulations of several cloud providers, including Rack space, Amazon Web Services, Google Cloud, and IBM. Organizations may automate

compliance processes and define business Cloud security rules using this publicly available knowledge graph [28].

Bustillo, Patrimonio and Mateo (2020). The survey questionnaires were filled out by 40 individuals, including both BCPO personnel and residents. The reliability coefficient was 0.96, and the yield was 100%. Based on values ranging from 4.64 to 4.76, the study's results show that RAD may create information systems with software attributes that are highly ISO/IEC 9126-1 compliant. The thorough evaluation of system software quality is unaffected by user gender or type. On the other hand, ISO/IEC 9126-1 software quality rises in tandem with the general assessment level of software quality. A recommendation to implement the created information systems was made based on the study's positive results [29].

Torkura et al. (2019) intended to evaluate the current research on the security, hazards, and challenges of cloud computing technology. Seven significant security risks to cloud computing services were identified by this SLR's results. Data leaking and manipulation were found to be among the most talked-about subjects in the selected literature. There was additional security concerns linked to data intrusion and cloud computing storage. Both cloud service providers and their customers continue to have difficulties with consumer data outsourcing, according to this SLR. The results of the SLR analysis provide some recommendations for future research that can improve data availability, integrity, and confidentiality [30].

Silva, Pereira and Ribeiro (2018) enhance the efficiency of the issue management teams by implementing a module to automatically classify incident tickets. If an incident ticket system (ITS) is already in place, this module can be added as an addition to make the process even more efficient and less prone to mistakes when classifying incidents. In order to get the system back up and running as soon as possible with as little disruption to the company and its customers as feasible, the IT incident management process relies on precise classification of incident tickets to assign them to the correct resolution group [31].

Table I covers research on cloud-native incident management, focusing on AI-driven detection, automation, and extensibility. It also points out problems with real-time self-healing and multi-cloud interoperability.

Table 1 Literature on Automated Incident Management within Cloud-Native SRE

Authors (Year)	Study Focus	Methodology / Approach	Tools / Data Sources	Strengths	Limitations
Lin et al.(2022)	Extensibility of cloud-native service stacks	Cloud-native light-cone model; holistic architecture; OMStack implementation	Container clusters, PaaS/SaaS (OMCC, OMAI, OMBD)	Provides a systematic view for stakeholders; integrates multiple cloud services; addresses stack extensibility	Implementation focus: limited empirical evaluation of performance under real workloads
Li et al. (2021)	Automatic incident detection	Warden system for global incident detection and prioritization	Azure alerts, IcM platform	Proactive incident detection; cross-team collaboration; real-world deployment	Limited to Azure ecosystem; may not generalize to multi-cloud
Ozer et al. (2020)	Analysis of increasing cyber-attacks amid cloud adoption by small and midsize enterprises (SMEs).	Empirical study using statistical analysis of reported cyber incidents across SMEs adopting cloud services.	Public cybersecurity datasets and survey data from SMEs.	Highlights paradox of increased vulnerabilities despite migration to the cloud; provides quantitative insights into security risks.	Does not propose mitigation mechanisms; limited geographic and sectoral scope.
Joshi, et.al.(2020)	Cloud data compliance and security	Semantically rich knowledge graph; compliance evaluation	Privacy policies of major CSPs	Automates compliance processes; publicly available knowledge graph	Focused on compliance; not directly on incident detection or SRE performance
Cloyd et.al. (2020)	Information system quality	Survey-based evaluation of RAD systems	40 respondents (BCPO personnel, citizens)	High reliability coefficient; validated ISO/IEC 9126-1 quality metrics	Small sample size; limited to RAD systems; not cloud-specific
Torkura et al. (2019)	Cloud computing security threats	Systematic Literature Review (2010–2020)	80 selected papers	Comprehensive SLR; identifies major threats and gaps	Mostly descriptive; does not provide technical solutions or implementations
Silva, et.al. (2018)	Incident ticket categorization	Automatic categorization module for ITS	Incident ticket systems	Improves productivity; reduces categorization errors; practical integration	Focused on ticketing; limited scope in large-scale distributed or cloud-native systems

Conclusion and Future Work

Organizations are turning to cloud-native infrastructures in an increasingly digital world to maintain their service delivery and be operationally resilient. Site Reliability Engineering (SRE) has become the main subject to handle the complexity of distributed systems by means of automation, observability, and performance optimization. This research determines that automated incident management processes in cloud-native SRE have a major effect on the capacity, the reliability, and the flexibility of the system. The SRE teams can have real-time monitoring, intelligent alerting, and predictive maintenance through the use of Prometheus, Grafana, and Open Telemetry, which eventually results in the reduction of both Mean Time to Detection (MTTD) and Mean Time to Resolution (MTTR). The research results also emphasize the role of event-driven and serverless architectures in the development of self-healing systems as well as in the effortless recovery from errors. Therefore, automation along with observability essentially transforms incident management into a pro-active, intelligent, and data-driven approach which is at the core of cloud-native operations that are modern and resilient.

The next agenda for research encompasses facilitating cross-cloud interoperability, using federated learning for predictive analytics, and designing adaptive feedback loops to increase automation and resilience to large-scale cloud-native SRE deployments. Simultaneously, the study of AI-based self-optimization models and autonomous orchestration agents might also propel system intelligence, scalability, and fault tolerance in dynamic cloud environments.

References

[1] S. Palacios Chavarro, P. Nespoli, D. Díaz-López, and Y. Niño Roa, "On the Way to Automatic Exploitation of Vulnerabilities and Validation of Systems Security through Security Chaos Engineering," *Big Data Cogn. Comput.*, vol. 7, no. 1, p. 1, Dec. 2022, doi: 10.3390/bdcc7010001.

[2] N. Kavyashree, S. M. C, and L. M. R, "Site reliability engineering for IOS mobile application in small-medium scale industries," *Glob. Transitions Proc.*, vol. 2, no. 2, pp. 137–144, Nov. 2021, doi: 10.1016/j.gltp.2021.08.065.

[3] E. Grünewald, "Cloud Native Privacy Engineering through DevPrivOps," in *IFIP Advances in Information and Communication Technology*, vol. 644 IFIP, 2022, pp. 122–141. doi: 10.1007/978-3-030-99100-5_10.

[4] V. Shah, "Managing Security and Privacy in Cloud Frameworks : A Risk with Compliance Perspective for Enterprises," *Int. J. Curr. Eng. Technol.*, vol. 12, no. 6, pp. 606–618, 2022.

[5] M. Simić, G. Sladić, M. Zarić, and B. Markoski, "Infrastructure as Software in Micro Clouds at the Edge," *Sensors*, vol. 21, no. 21, p. 7001, Oct. 2021, doi: 10.3390/s21217001.

[6] S. Gupta, N. Agrawal, and S. Gupta, "A Review on Search Engine Optimization: Basics," *Int. J. Hybrid Inf. Technol.*, vol. 9, no. 5, pp. 381–390, May 2016, doi: 10.14257/ijhit.2016.9.5.32.

[7] A. Tanikonda, S. R. Katragadda, S. R. Peddinti, and B. K. Pandey, "Integrating AI-Driven Insights into DevOps Practices," *SSRN Electron. J.*, vol. 2, no. 1, pp. 318–339, 2021, doi: 10.2139/ssrn.5102369.

[8] R. Zhang, Y. Li, H. Li, and Q. Wang, "Evolutionary Game Analysis on Cloud Providers and Enterprises' Strategies for Migrating to Cloud-Native under Digital Transformation," *Electronics*, vol. 11, no. 10, p. 1584, May 2022, doi: 10.3390/electronics11101584.

[9] M. Szalay, P. Mátray, and L. Toka, "State Management for Cloud-Native Applications," *Electronics*, vol. 10, no. 4, p. 423, Feb. 2021, doi: 10.3390/electronics10040423.

[10] A. P. Perumal and P. Chintale, "Improving operational efficiency and productivity through the fusion of DevOps and SRE practices in multi-cloud operations," *Int. J. Cloud Comput. Database Manag.*, vol. 3, no. 2, pp. 49–53, Jul. 2022, doi: 10.33545/27075907.2022.v3.i2a.51.

[11] G. Modalavalasa and S. Pillai, "Exploring Azure Security Center : A Review of Challenges and Opportunities in Cloud Security," *ESP J. Eng. Technol. Adv.*, vol. 2, no. 2, 2022, doi: 10.56472/25832646/JETA-V2I2P120.

[12] A. R. P. Reddy, "Automating Incident Response: Ai-Driven Approaches To Cloud Security Incident Management," *Chelonian Conserv. Biol.*, vol. 15, no. 2, pp. 323–325, 2020, doi: 10.18011/2020.04(1).73.79.

[13] M. Usman, S. Ferlin, A. Brunstrom, and J. Taheri, "A Survey on Observability of Distributed Edge & Container-Based Microservices," *IEEE Access*, vol. 10, pp. 86904–86919, 2022, doi: 10.1109/ACCESS.2022.3193102.

[14] V. M. L. G. Nerella, "Observability-Driven SRE Practices for Proactive Database Reliability and Rapid Incident Response," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 7, no. 8, pp. 32–38, Aug. 2019, doi: 10.17762/ijritcc.v7i8.11710.

[15] N. Kratzke, "Cloud-Native Observability: The Many-Faceted Benefits of Structured and Unified Logging—A Multi-Case Study," *Futur. Internet*, vol. 14, no. 10, p. 274, Sep. 2022, doi: 10.3390/fi14100274.

[16] B. R. Cherukuri, "Microservices and containerization: Accelerating web development cycles," *World J. Adv. Res. Rev.*, vol. 6, no. 1, pp. 283–296, Apr. 2020, doi: 10.30574/wjarr.2020.6.1.0087.

[17] A. Gogineni, "Observability Driven Incident Management for Cloud-native Application Reliability," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 9, no. 2, 2021.

[18] Q. Duan, "Intelligent and Autonomous Management in Cloud-Native Future Networks—A Survey on Related Standards from an Architectural Perspective," *Futur. Internet*, vol. 13, no. 2, p. 42, Feb. 2021, doi: 10.3390/fi13020042.

[19] C. Mokkapati and S. Jain, "Enhancing Site Reliability Engineering (SRE) Practices In Large-Scale Retail Enterprises," *Int. J. Creat. Res. Thoughts (*, vol. 9, no. 11, pp. 870–886, 2021.

[20] V. Verma, "Big Data and Cloud Databases Revolutionizing Business Intelligence," *TIJER*, vol. 9, no. 1, pp. 48–58, 2022.

[21] N. Singla, Chahat, Nisha, and Harnoor, "A Review Paper on Cloud Computing," *Proc. - 2022 2nd Int. Conf. Innov. Sustain. Comput. Technol. CISCT 2022*, vol. 9, no. 2, pp. 109–114, 2022, doi: 10.1109/CISCT55310.2022.10046572.

[22] R. Tandon and D. Patel, "Evolution of Microservices Patterns for Designing Hyper- Scalable Cloud-Native Architectures," *ESP J. Eng. Technol. Adv.*, vol. 1, no. 1, pp. 288–297, 2021, doi: 10.56472/25832646/JETA-V1I1P131.

[23] P. Somasundaram, "Trends and Challenges in Cloud-Native Architectures," *J. Sci. Eng. Res.*, vol. 7, no. 11, pp. 214–219, 2020.

[24] P. Ganesan, "Observability In Cloud-Native Environments Challenges And Solutions," *Int. J. Core Eng. Manag.*, vol. 7, no. 4, 2022.

[25] J. Lin, D. Xie, J. Huang, Z. Liao, and L. Ye, "A multi-dimensional extensible cloud-native service stack for enterprises," *J. Cloud Comput.*, vol. 11, no. 1, p. 83, Nov. 2022, doi: 10.1186/s13677-022-00366-7.

[26] L. Li *et al.*, "Fighting the fog of war: Automated incident detection for cloud systems," in *2021 USENIX Annual Technical Conference*, 2021.

[27] M. Ozer, S. Varlioglu, B. Gonen, V. Adewopo, N. Elsayed, and S. Zengin, "Cloud Incident Response: Challenges and Opportunities," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2020, pp. 49–54. doi: 10.1109/CSCI51800.2020.00015.

[28] K. P. Joshi, L. Elluri, and A. Nagar, "An Integrated Knowledge Graph to Automate Cloud Data Compliance," *IEEE Access*, vol. 8, pp. 148541–148555, 2020, doi: 10.1109/ACCESS.2020.3008964.

[29] J. C. M. Bustillo, G. A. Patrimonio, and J. T. I. Mateo, "Automated Incident Reporting Management System Using Mobile Technology," *Int. J. Innov. Manag. Technol.*, vol. 11, no. 1, pp. 18–26, 2020, doi: 10.18178/ijimt.2020.11.1.870.

[30] K. Torkura, M. I. H. Sukmana, F. Cheng, and C. Meinel, "SlingShot - Automated Threat Detection and Incident Response in Multi Cloud Storage Systems," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, IEEE, Sep. 2019, pp. 1–5. doi: 10.1109/NCA.2019.8935040.

[31] S. Silva, R. Pereira, and R. Ribeiro, "Machine learning in incident categorization automation," in *Iberian Conference on Information Systems and Technologies, CISTI*, 2018. doi: 10.23919/CISTI.2018.8399244.