UNIVERSIDAD NACIONAL DE SAN MARTIN

RCSI
https://revistas.unsm.edu.pe/index.php/rcsi
e-ISSN: 2709-992X

Original article
Artículo original
Jul-Dec, 2025

Systems monitoring platform integrating artificial intelligence for incident response in servers

Plataforma de monitoreo de sistemas integrando inteligencia artificial para respuesta a incidentes en servidores

Espinosa-Luna, Bruno Hiroshi[1*]
Castillo-Oliva, Johann[1]
García-Gutiérrez, Willy Francisco[1]
Mendoza-de-los-Santos, Alberto[1]

[1]Universidad Nacional de Trujillo, Trujillo, Perú

Received: 17 Nov. 2024 | **Accepted:** 21 Mar. 2024 | **Published:** 20 Jul. 2025
Corresponding author*: t033300120@unitru.edu.pe
How to cite this article: Espinosa-Luna, B. H., Castillo-Oliva, J., García-Gutiérrez, W. F. & Mendoza-de-los-Santos, A. (2025). Systems monitoring platform integrating artificial intelligence for incident response in servers. *Revista Científica de Sistemas e Informática, 5*(2), e811. https://doi.org/10.51252/rcsi.v5i2.811

ABSTRACT

The increasing complexity of IT management and the need to monitor critical infrastructure metrics, such as CPU usage, memory, storage, and service logs, detect failures, and respond quickly to alerts, imply the adoption of advanced technologies that enable comprehensive monitoring and efficient response. This work developed a server monitoring system with alerts sent via Telegram. Additionally, it integrates artificial intelligence to provide immediate solutions to server incidents, using tools such as Grafana and Prometheus for metric collection and Grafana Loki for log management. The OpenAI API was incorporated to analyze the logs and enhance alerts with a detailed diagnosis. A total of 311 tests were conducted, where the results showed that the system notified incidents in an average of 1.02 seconds, while the GPT model completed the analysis in an average of 2.17 seconds, allowing root causes of problems to be identified and timely recommendations for resolution to be generated. It is concluded that the integration of artificial intelligence and proactive monitoring improves incident management, suggesting future applications in IoT environments to enrich monitoring.

Keywords: alerts; IT governance; Grafana; LLM; Loki; observability

RESUMEN

La creciente complejidad en la gestión de TI y la necesidad de monitorear métricas críticas de infraestructura, como uso de CPU, memoria, almacenamiento y logs de servicios, detectar fallas y responder rápidamente a alertas, implican la adopción de tecnologías avanzadas que permitan una supervisión integral y una respuesta eficiente. El presente trabajo desarrolló un sistema de monitoreo de servidores con alertas enviadas mediante Telegram. Además, integra inteligencia artificial para proporcionar soluciones inmediatas a los incidentes en los servidores, utilizando herramientas como Grafana y Prometheus para la recolección de métricas, y Grafana Loki para la gestión de logs. La API de OpenAI se incorporó para analizar los registros y enriquecer las alertas con un diagnóstico detallado. En total se realizaron 311 pruebas, donde los resultados mostraron que el sistema notificó las incidencias en un promedio de 1,02 segundos, mientras que el modelo GPT completó el análisis en 2,17 segundos en promedio, permitiendo identificar causas raíz de los problemas y generar recomendaciones oportunas para su resolución. Se concluye que la integración de inteligencia artificial y monitoreo proactivo mejora la gestión de incidentes, sugiriendo futuras aplicaciones en entornos IoT para enriquecer la supervisión.

Palabras clave: alertas; gobierno de TI; Grafana; LLM; Loki; observabilidad

1. INTRODUCTION

Monitoring platforms play an important role in the control, management, and optimization of technological resources. The growing complexity of IT services due to virtualization, cloud computing, Big Data, and microservices has created a need for constant improvement in monitoring and alert management to maintain stable operations (Yu et al., 2024). Different IT management and governance frameworks such as TOGAF or ITIL version 4 place special emphasis on monitoring because it is a practice that enables IT capacity management by defining baselines for resources and tracking them throughout their lifecycle within the IT architectures proposed by each framework (Santosa & Mulyana, 2023). Additionally, it's not just about collecting random data, but also considering data quality for proper use in monitoring (Ehrlinger & Wöß, 2022).

The diversity of services to evaluate, such as infrastructure, networks, databases, operating systems, IoT sensors, among others, has posed a significant challenge, leading to the development of specialized monitoring systems for each type of device. For example, Sun et al. (2020) implemented a surveillance system for cloud clusters, tracking resources such as CPU, RAM, and web service logs.

The variety of situations in IT monitoring implies the existence of non-uniform data and difficulty in finding patterns in logs or alert bodies. Therefore, tools have been created in recent years to facilitate the capture of relevant information in alerts to understand the cause of incidents and the possibility of predicting risk based on them, with artificial intelligence being one of the most adopted technologies to effectively solve this problem (Ahmed et al., 2022).

Machine learning, large language models, or pre-trained natural language models are applied in the constant monitoring of IT infrastructure. Bhanage et al. (2023) analyze software event logs from web servers and operating systems for fault detection using the pre-trained BERT model, showing high performance in correctly detecting event content despite the different formats handled by each system. Similarly, Kuang et al. (2024) performed alert content analysis using pre-trained models and their own hybrid model COLA, obtaining the context of an alert dataset in less than 10 seconds, which the authors consider as the maximum time for production deployment.

Additionally, there is a growing trend of integrating instant messaging with monitoring or supervision systems for real-time alert notifications, improving operator response capability (Penkov & Taneva, 2021), such as the monitoring system developed by Butarbutar et al. (2023), which implemented an interactive Telegram bot for receiving alerts.

Given the above, the question arises: How can a system monitoring platform that integrates artificial intelligence support incident response in servers? To answer this question, the general objective was to develop a system monitoring platform that integrates artificial intelligence to explore its applicability in server incident response. The specific objectives were: (i) design a monitoring system that integrates artificial intelligence for the detection and analysis of server incidents, (ii) evaluate the response times of alerts sent through messaging applications, and (iii) evaluate the response time of the GPT model for incident tracking and log analysis.

2. MATERIALS AND METHODS

The study was conducted in a cloud environment using a set of 4 instances deployed on Google Cloud Platform's Compute Engine. The research was of a non-experimental descriptive type with a transactional design. For the statistical analysis of incident response, descriptive analysis of mean, maximum, and minimum values of the collected data was applied. A total of 311 alert records were collected and stored in a database. Regarding the monitoring system, programming was used to collect response times for both the constructed system and the OpenAI model through the API.

2.1. Software

In the development of this platform, various software tools were used, which contributed to the construction of the monitoring system. Primarily, Grafana was used, which is an open-source platform designed for creating interactive graphs and dashboards and is widely used in applications that seek to display real-time data (Broniewski et al., 2023). Additionally, it is a highly valued tool for its ability to integrate with multiple data sources, as its flexibility and ease of extension make it suitable for a wide range of applications, from deployment infrastructure monitoring to Internet of Things solutions (Anam et al., 2023; Hadikusuma et al., 2023).

Among the most common integrations in monitoring systems are Node Exporter, Grafana Loki, and Prometheus (Chan et al., 2022). Node Exporter is an exporter for Prometheus that collects operating system metrics, such as CPU, memory, and disk usage. Grafana Loki is a log management solution, designed to work efficiently with large volumes of log data (Simili et al., 2021). Its query language, LogQL, allows users to perform quick searches and precise filtering of logs.

Prometheus, on the other hand, is an open-source monitoring and alerting system that collects metrics from systems and services. It stands out for its ability to collect and store metrics from instrumented services in a time-series database (Erdei & Toka, 2023). For this purpose, it uses PromQL, a specialized query language for data extraction and alert creation based on specific metrics (Adamyaa et al., 2022).

APIs are interfaces that allow the integration of multiple pre-existing functionalities for developers, facilitating communication between different software modules (Lamothe et al., 2021). There are various APIs with multiple purposes; in this research, integration was carried out with the OpenAI API, which provides a series of models with multiple purposes, from chatbot creation through natural language processing (NLP) to image generation (Santoso et al., 2023). The Telegram API, for its part, facilitates the integration of real-time messaging services, allowing for notification sending and user interaction through messages for the creation of systems with specific purposes (Alia et al., 2024).

2.2. Architecture design

The proposed architecture, as illustrated in Figure 1, consists of establishing a central server responsible for monitoring the status and performance of the rest of the organization's servers. Using Prometheus, metrics from each server will be stored, which will be collected through Node Exporter, an agent that exposes basic operating system statistics. These metrics were useful for evaluating performance, detecting possible anomalies in resource usage (such as CPU, memory, and disk), and anticipating failures before they affect normal service operation. With this information, it will be possible to implement real-time alert strategies and proactively optimize the infrastructure.

Grafana Agent was used to send logs to Grafana Loki, installed on the central server. These logs detailed events and activities on each monitored server. By centralizing logs in Loki, performance metrics could be correlated with specific events, identifying the root cause of problems such as bottlenecks, service failures, or anomalous behaviors. Additionally, log analysis enriched the alerts through the OpenAI model, providing a complete view of the infrastructure's state.

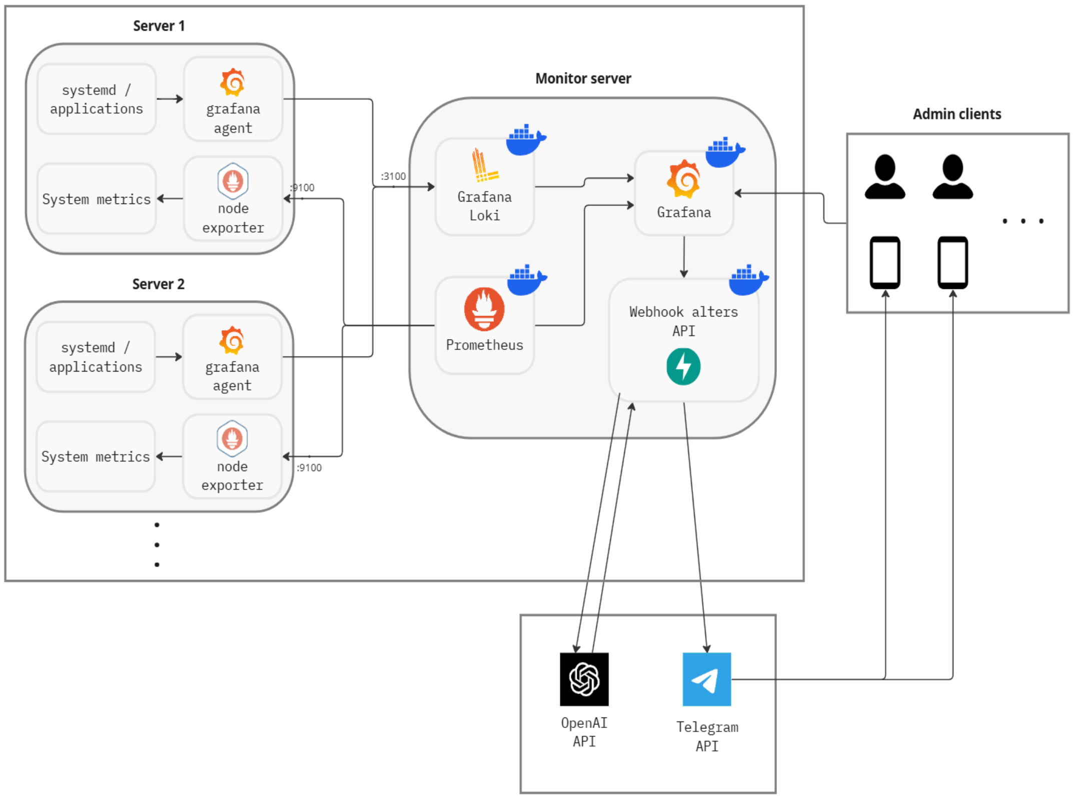


Figure 1. AI-integrated monitoring system architecture

The integration of these tools allows teams to obtain detailed information about the status and performance of their microservices, quickly detect errors, and ensure optimal functioning of the architecture (Jani, 2024).

2.3. Alert system

Grafana offers a configurable alert system based on defined rules for both logs and server metrics. These alerts are triggered when certain indicator values exceed established thresholds, allowing for a quick response to possible incidents. For this research, the following alerts have been been considered:

Table 1. Alert rules configured for server monitoring

Alert name	Summary	Condition/Threshold	Evaluation Frequency
CPU Usage	CPU usage is over 90%	CPU > 90% for 10 sec	Every 1 min
RAM Usage	RAM usage is over 90%	Memory > 95% for 10 sec	Every 1 min
Server Down Detection	Instance has recently shut down	Instance state < 0.5	Every 1 min
SSH Authentication Failure	More than 4 SSH authentication failures have been detected on the server in the last 10 minutes	More than 4 failed attempts since last 10 minutes	Every 1 min

When any of the previously configured conditions are met, an event is generated and sent to the webhook programmed in the API. This webhook is responsible for interacting with the Telegram API to immediately notify all administrators through their mobile devices about the occurred event. In parallel, the logs are analyzed through the OpenAI API, with the aim of automatically diagnosing the root cause of the problem. The analysis results are also sent to administrators through Telegram, providing useful and detailed information so they can take immediate corrective actions.

2.4. Development framework

Scrum is an agile framework for project management based on collaboration, self-organization, and delivery of functional products in incremental iterations (Sassa et al., 2023). It operates through sprints, which consist of time-boxed planned events. As illustrated in Figure 2, it encompasses four events during development, whose fulfillment, according to the Scrum guide, influences the perception of project success (Kadenic et al., 2023): Sprint Planning, where the team defines what work will be done during the sprint and how it will be achieved; Daily Scrum, a brief daily meeting to synchronize activities and plan the next 24 hours; Sprint Review, where completed work is presented at the end of the sprint and feedback is received from stakeholders; and Sprint Retrospective, a meeting to reflect on the sprint and improve future iterations (Behutiye et al., 2024; Sassa et al., 2023).

Additionally, these events are supported by Scrum artifacts, which are essential in the success of agile software development, where the Product Backlog is a prioritized list of all work needed in the project, managed by the Product Owner. The Sprint Backlog is a subset of the Product Backlog that the development team commits to completing during a specific sprint (Garcia et al., 2021). Finally, the Burndown Chart is a visual tool that shows the team's progress toward completing work in the sprint (Ghimire & Charters, 2022).

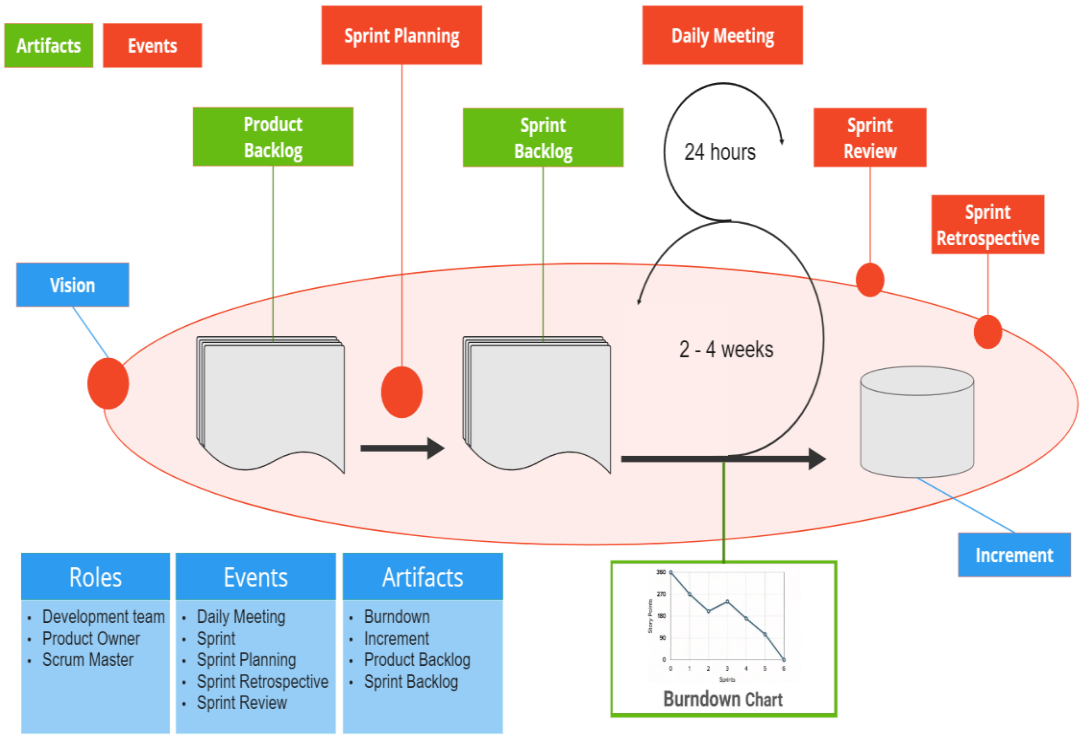


Figure 2. Scrum life cycle by Sassa et al. (2023)

To obtain a clear initial vision of the project, user stories were listed and an estimated Product Backlog was created to guide the development of the monitoring system. According to Table 1, 20 user story points were estimated to be completed in 14.5 days.

Table 2. Estimated Product Backlog

Sprint	#US	Description	Size	Points	Time (days)
First Sprint	US1	Collect performance metrics (CPU, memory, disk, and network) from each server using Prometheus and Node Exporter.	XS	1	1
	US2	Collect logs from all servers using Grafana Loki for event analysis on the servers.	XS	1	1
	US3	Develop a dashboard to monitor Prometheus performance metrics in real-time.	M	3	2
Second Sprint	US4	Develop a dashboard to monitor system logs from the servers in real-time.	M	3	2
	US5	Configure alerts to automatically notify when servers exceed resource usage thresholds.	S	2	1.5
	US6	Configure alerts based on log analysis to automatically notify unusual behaviors on the servers.	S	2	1.5
	US7	Send resource usage alerts to organization users via Telegram.	L	5	3
Third Sprint	US9	Send log analysis-based alerts to organization users via Telegram.	XS	1	1
	US10	Integrate an OpenAI model into the alerts to provide detailed tracking of possible causes and action suggestions.	S	2	1.5
User Story Points / Estimated Time				20	14.5

3. RESULTS AND DISCUSSION

3.1. Metrics and Logs Visualization

The system allows visualizing resource usage metrics in the different servers connected in real time through Grafana, integrating the data coming from Prometheus and Node Exporter. As shown in figure 3, the dashboard displays detailed information about the CPU, memory and storage usage of the servers. This integration aims to provide a continuous monitoring tool that is easy to manage without direct interaction with the servers.

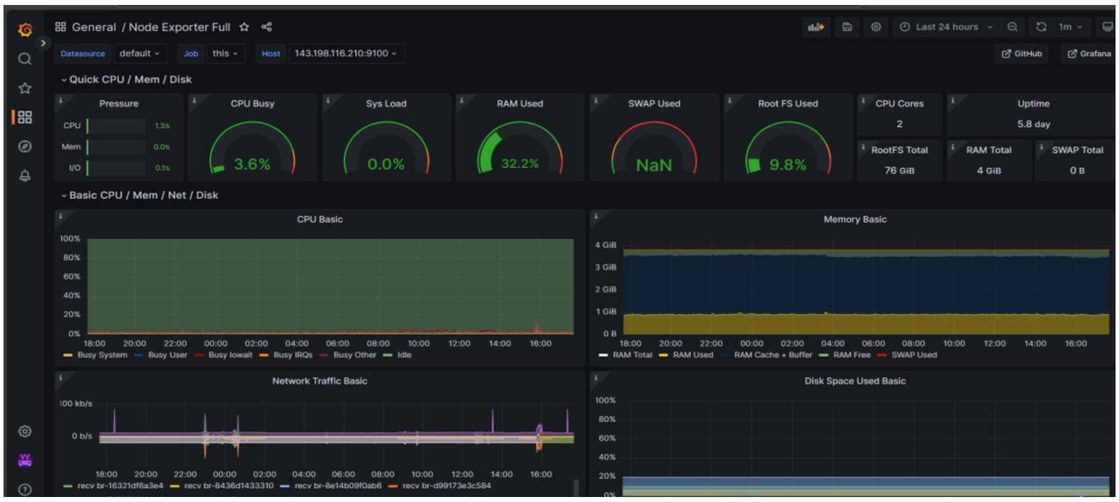
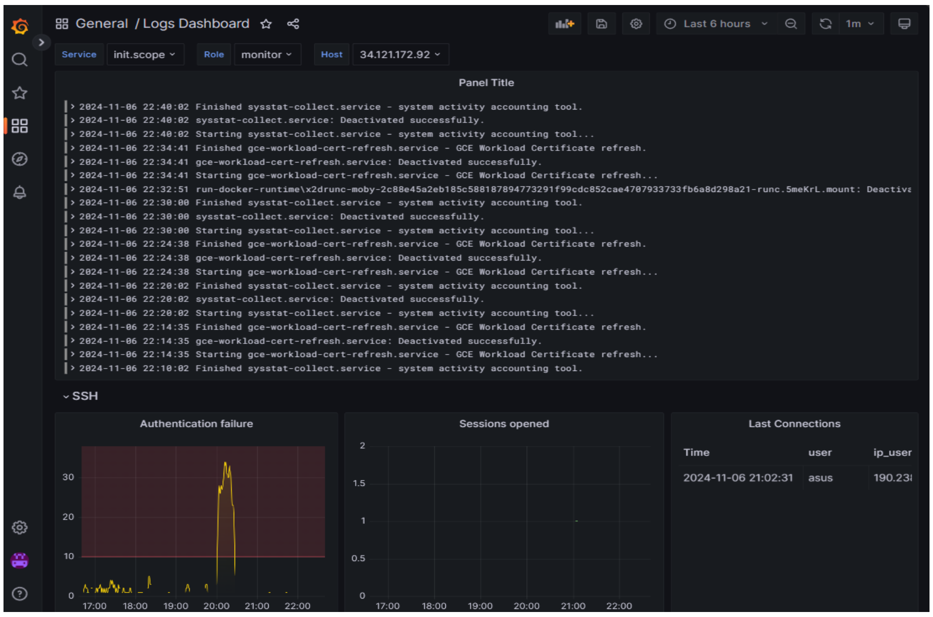


Figure 3. Resource usage dashboard in Grafana

The integration with Loki made it possible to collect and store the logs generated by the servers. Figure 4 shows how the logs were visualized in the Grafana dashboard, which allowed the operations team to access the events recorded in a defined time range. This functionality provides additional context to the analysis of unusual behavior and server interactions.



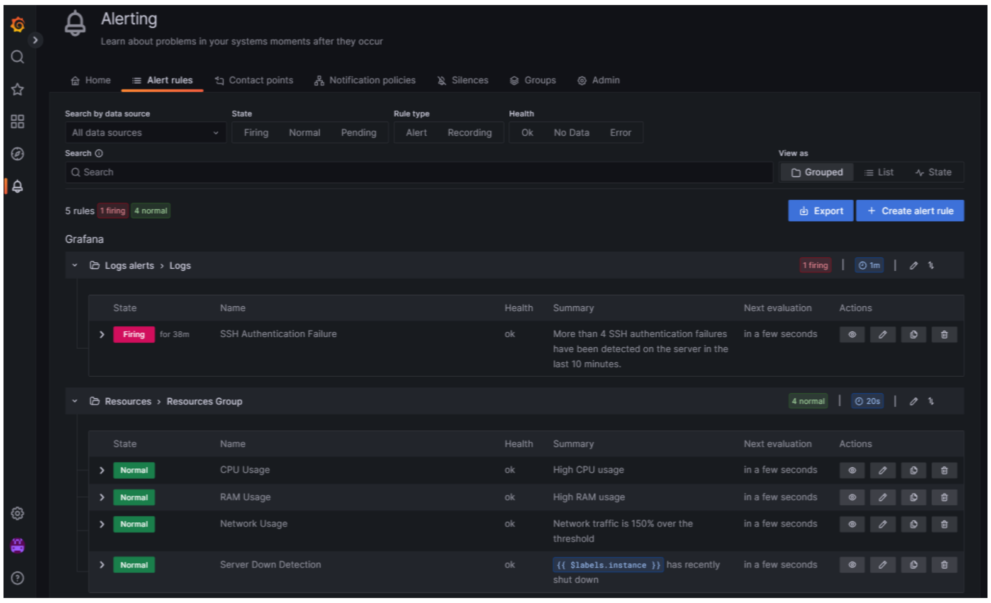
```
> 2024-11-06 22:40:02 Finished sysstat-collect.service - system activity accounting tool.
> 2024-11-06 22:40:02 sysstat-collect.service: Deactivated successfully.
> 2024-11-06 22:40:02 Starting sysstat-collect.service - system activity accounting tool...
> 2024-11-06 22:34:41 Finished gce-workload-cert-refresh.service - GCE Workload Certificate refresh.
> 2024-11-06 22:34:41 gce-workload-cert-refresh.service: Deactivated successfully.
> 2024-11-06 22:34:41 Starting gce-workload-cert-refresh.service - GCE Workload Certificate refresh...
> 2024-11-06 22:32:51 run-docker-runtime\x2drunc-moby-2c88e45a2eb185c588187894773291f99cdc852cae4707933733fb6a8d298a21-runc.5meKrL.mount: Deactiva
> 2024-11-06 22:30:00 Finished sysstat-collect.service - system activity accounting tool.
> 2024-11-06 22:30:00 sysstat-collect.service: Deactivated successfully.
> 2024-11-06 22:30:00 Starting sysstat-collect.service - system activity accounting tool...
> 2024-11-06 22:24:38 Finished gce-workload-cert-refresh.service - GCE Workload Certificate refresh.
> 2024-11-06 22:24:38 gce-workload-cert-refresh.service: Deactivated successfully.
> 2024-11-06 22:24:38 Starting gce-workload-cert-refresh.service - GCE Workload Certificate refresh...
> 2024-11-06 22:20:02 Finished sysstat-collect.service - system activity accounting tool.
> 2024-11-06 22:20:02 sysstat-collect.service: Deactivated successfully.
> 2024-11-06 22:20:02 Starting sysstat-collect.service - system activity accounting tool...
> 2024-11-06 22:14:35 Finished gce-workload-cert-refresh.service - GCE Workload Certificate refresh.
> 2024-11-06 22:14:35 gce-workload-cert-refresh.service: Deactivated successfully.
> 2024-11-06 22:14:35 Starting gce-workload-cert-refresh.service - GCE Workload Certificate refresh...
> 2024-11-06 22:10:02 Finished sysstat-collect.service - system activity accounting tool.
```

Time	user	ip_user
2024-11-06 21:02:31	asus	190.23

Figure 4. Viewing logs through Loki in Grafana Dashboard

3.2. Alert Configuration

The system was configured to manage two types of alerts: resource usage alerts and alerts based on system logs. Figure 5 shows the control panel that allows tracking the status of the alerts. On the other hand, Figures 6 (a) and 6 (b) show alert messages that warn about excessive RAM and CPU resource usage analysis, together with a follow-up message, which is linked to the alert identifier previously received and provides information about the impact of the alert and suggestions for action, thus facilitating incident response.



5 rules 1 firing 4 normal

Grafana

Logs alerts › Logs — 1 firing | 1m

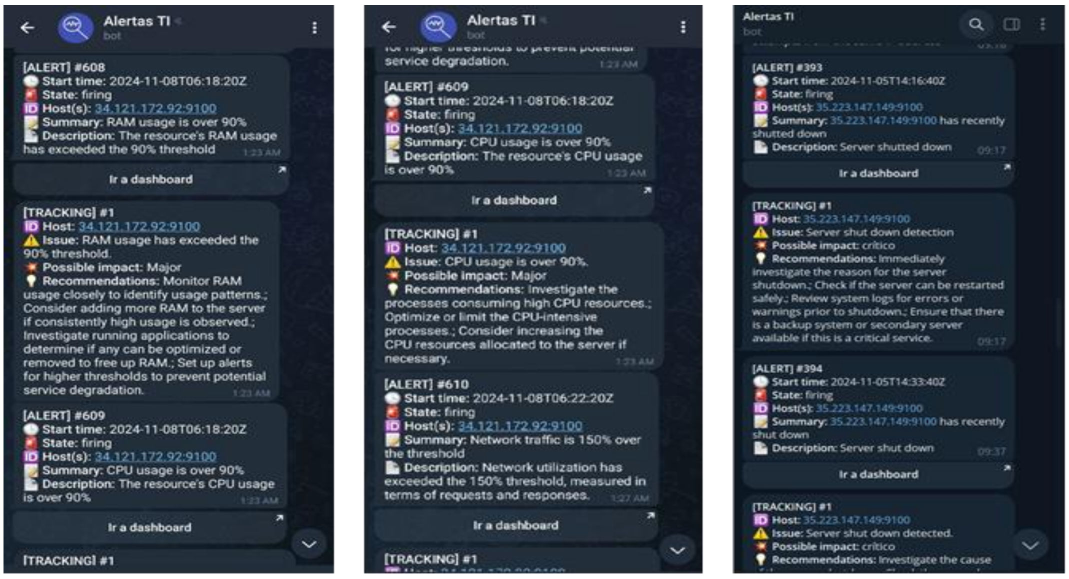
State	Name	Health	Summary	Next evaluation
Firing for 38m	SSH Authentication Failure	ok	More than 4 SSH authentication failures have been detected on the server in the last 10 minutes.	in a few seconds

Resources › Resources Group — 4 normal | 20s

State	Name	Health	Summary	Next evaluation
Normal	CPU Usage	ok	High CPU usage	in a few seconds
Normal	RAM Usage	ok	High RAM usage	in a few seconds
Normal	Network Usage	ok	Network traffic is 150% over the threshold	in a few seconds
Normal	Server Down Detection	ok	{{ $labels.instance }} has recently shut down	in a few seconds

Figure 5. Alert management screen

Figure 6 (c) shows a similar dynamic, but displaying an alert related to logs, specifically unauthorized access attempts through the SSH service of a server.



```
[ALERT] #608
Start time: 2024-11-08T06:18:20Z
State: firing
Host(s): 34.121.172.92:9100
Summary: RAM usage is over 90%
Description: The resource's RAM usage
has exceeded the 90% threshold                1:23 AM

[TRACKING] #1
Host: 34.121.172.92:9100
Issue: RAM usage has exceeded the
90% threshold.
Possible impact: Major
Recommendations: Monitor RAM
usage closely to identify usage patterns.;
Consider adding more RAM to the server
if consistently high usage is observed.;
Investigate running applications to
determine if any can be optimized or
removed to free up RAM.; Set up alerts
for higher thresholds to prevent potential
service degradation.                          1:23 AM

[ALERT] #609
Start time: 2024-11-08T06:18:20Z
State: firing
Host(s): 34.121.172.92:9100
Summary: CPU usage is over 90%
Description: The resource's CPU usage
is over 90%                                   1:23 AM

[TRACKING] #1
```

(a)

```
service degradation.                          1:23 AM

[ALERT] #609
Start time: 2024-11-08T06:18:20Z
State: firing
Host(s): 34.121.172.92:9100
Summary: CPU usage is over 90%
Description: The resource's CPU usage
is over 90%                                   1:23 AM

[TRACKING] #1
Host: 34.121.172.92:9100
Issue: CPU usage is over 90%.
Possible impact: Major
Recommendations: Investigate the
processes consuming high CPU resources.;
Optimize or limit the CPU-intensive
processes.; Consider increasing the
CPU resources allocated to the server if
necessary.                                    1:23 AM

[ALERT] #610
Start time: 2024-11-08T06:22:20Z
State: firing
Host(s): 34.121.172.92:9100
Summary: Network traffic is 150% over
the threshold
Description: Network utilization has
exceeded the 150% threshold, measured in
terms of requests and responses.              1:27 AM

[TRACKING] #1
```

(b)

```
[ALERT] #393
Start time: 2024-11-05T14:16:40Z
State: firing
Host(s): 35.223.147.149:9100
Summary: 35.223.147.149:9100 has recently
shutted down
Description: Server shutted down              09:17

[TRACKING] #1
Host: 35.223.147.149:9100
Issue: Server shut down detection
Possible impact: critico
Recommendations: Immediately
investigate the reason for the server
shutdown.; Check if the server can be restarted
safely.; Review system logs for errors or
warnings prior to shutdown.; Ensure that there
is a backup system or secondary server
available if this is a critical service.      09:17

[ALERT] #394
Start time: 2024-11-05T14:33:40Z
State: firing
Host(s): 35.223.147.149:9100
Summary: 35.223.147.149:9100 has recently
shut down
Description: Server shut down                 09:37

[TRACKING] #1
Host: 35.223.147.149:9100
Issue: Server shut down detected.
Possible impact: critico
Recommendations: Investigate the cause
```

(c)

Figure 6. Receiving alerts and tracking via Telegram

3.3. Evaluation of Alert Times and System Analysis

A total of 311 tests were conducted, where the values illustrated in Table 3 were obtained. It can be observed that the average initial alert time was 1.02 seconds, indicating that, on average, the system took just over a second to generate an alert from the moment the alert condition was detected.

Table 3. Evaluation of initial alert time

Parameter	Value (seg)
Mean	1,02
Maximum	1,97
Minimum	0,70

Similarly, Table 4 shows the evaluation of the response time of the gpt-4o-mini model when performing the analysis. The average response time was 2.17 seconds, indicating that, on average, the model takes just over two seconds to complete the data analysis.

Table 4. Evaluation of GPT model response time in analysis

Parameter	Value (seg)
Mean	2,17
Maximum	8,85
Minimum	1,13

Finally, the analysis is sent once the alerts have been processed by the AI model. Similar to the initial alert time, the values remain low and with little variability, averaging 0.55 seconds, as shown in Table 5.

Table 5. Evaluation of follow-up message sending time

Parameter	Value (seg)
Mean	0.55
Maximum	0.92
Minimum	0.19

Figure 7 presents a summary of the values obtained in all the tests conducted for acknowledged time, completion time, and send analysis time. It can be observed that the alert sending times are consistently low and show little variability. However, the completion time shows greater variability, with significant peaks.

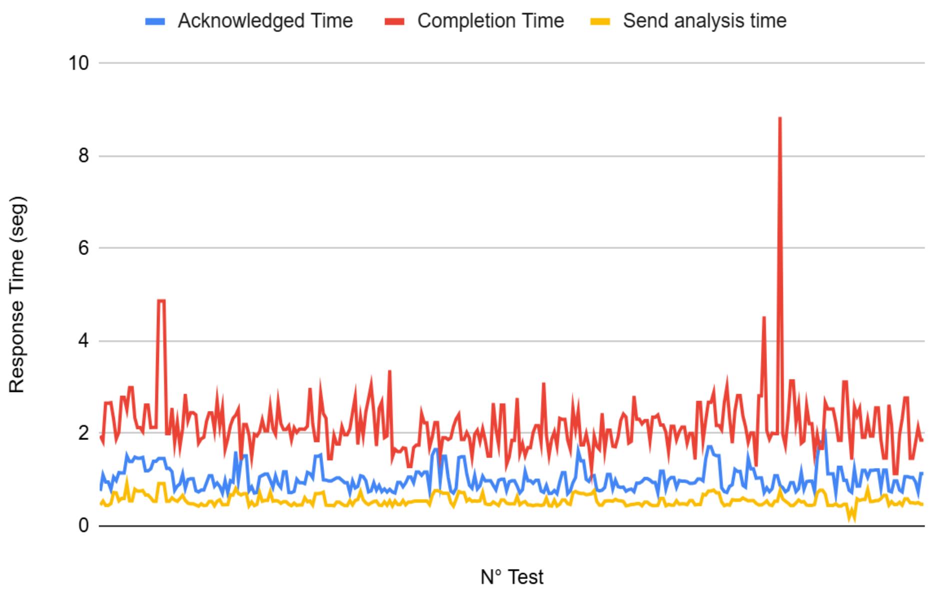


Figure 7. Response times of AI-integrated monitoring system

The monitoring system developed in this study incorporates tools for the detection and analysis of server incidents, complemented by an alert system through messaging applications. This approach reflects practices seen in previous studies, where solutions like Grafana and instant messaging applications, such as Telegram, have been implemented for sending automatic alerts (Butarbutar et al., 2023; Iqromullah et al., 2023). Additionally, the use of Prometheus for metric collection has been highlighted, allowing the creation of dashboards with real-time data updates for system monitoring (Gaol et al., 2022).

Similarly, for effective server monitoring, it is essential to define critical monitoring points. Igromullah et al. (2023) and Sun et al. (2020) mention that the most common problems in servers include excessive resource usage, such as hard disk, memory saturation, and the interruption of essential services for the operation of applications, such as databases and other services.

The evaluation of the initial alert time showed an average time of 1.02 seconds, with a result similar to the monitoring system of Sun et al. (2020), whose average was 8.79 seconds, both being quick in sending alerts.

On the other hand, the evaluation of the response time of the GPT model used for the analysis presented an average time of 2.17 seconds. The models evaluated by Kuang et al. (2024) such as Alert Storm or LiDAR, analyzed an alert in an average time of 1.82 and 0.90 seconds respectively, being faster than GPT; however, according to the authors, a response time of less than 10 seconds by artificial intelligence models is acceptable for production.

CONCLUSIONS

This work has presented an approach for developing a system monitoring platform that integrates artificial intelligence, exploring its applicability in responding to server incidents. A system was designed that incorporates artificial intelligence for the detection and analysis of server incidents, and the response times of alerts sent by the Telegram messaging application were evaluated, as well as the response time of the GPT model in incident tracking and log analysis. Out of a total of 311 tests, the results indicate that the system can generate alerts with an average time of 1.02 seconds and that the GPT model provides analysis with an average time of 2.17 seconds, adding value by offering better context and recommendations for the alerts. It is suggested that future research should focus on field studies in network operations centers to evaluate the effectiveness and adaptability of the platform in practical conditions. Additionally, it is recommended to explore integration with technologies such as the Internet of Things (IoT) to include physical monitoring, which could enrich incident management by covering both hardware and the operating environment.

FINANCING

The authors did not receive any sponsorship to carry out this study-article.

CONFLICT OF INTEREST

There is no type of conflict of interest related to the subject of the work.

AUTHORSHIP CONTRIBUTION

Conceptualization: Castillo-Oliva, J. Data curation: Castillo-Oliva, J., García-Gutiérrez, W. F. Formal analysis: García-Gutiérrez, W. F. Funding acquisition: Espinosa-Luna, B. H. Investigation: Espinosa-Luna, B. H., Castillo-Oliva, J. Methodology: García-Gutiérrez, W. F. Project administration: Espinosa-Luna, B. H. Software: Espinosa-Luna, B. H., Castillo-Oliva, J., García-Gutiérrez, W. F. Resources: García-Gutiérrez, W. F. Supervision & Validation: Mendoza-de-los-Santos, A. Visualization: Castillo-Oliva, J., García-Gutiérrez, W. F. Writing – original draft: Espinosa-Luna, B. H., Castillo-Oliva, J., García-Gutiérrez, W. F. Writing – review & editing: Espinosa-Luna, B. H., Castillo-Oliva, J., García-Gutiérrez, W. F., Mendoza-de-los-Santos, A.

AVAILABILITY OF DEPOSITED DATA

The dataset analyzed in this research is accessible through the Mendeley Data repository at https://data.mendeley.com/datasets/md7x42rcbm/1, identified by DOI: 10.17632/md7x42rcbm.1. It is shared under the Creative Commons Attribution 4.0 International (CC BY 4.0) license, permitting usage, sharing, and reproduction with appropriate attribution to the original creator.

REFERENCES

Adamyaa, D., Dinesh, S., Priya, & Soumya. (2022). Building a Monitoring Framework for a Distributed Cloud Application Using Prometheus and Chef—An Overview. International *Journal for Research in Applied Science and Engineering Technology*, 10(7), Article 7. https://doi.org/10.22214/ijraset.2022.45716

Ahmed, S., Singh, M., Doherty, B., Ramlan, E., Harkin, K., & Coyle, D. (2022). AI for Information Technology Operation (AIOps): A Review of IT Incident Risk Prediction. *2022 9th*

International Conference on Soft Computing & Machine Intelligence (ISCMI), 253-257. https://doi.org/10.1109/ISCMI56532.2022.10068482

Alia, P. A., Prayogo, J. S., Kriswibowo, R., & Setyadi, A. T. (2024). Implementation Open Artificial Intelligence ChattGPT Integrated With Whatsapp Bot. *Advance Sustainable Science, Engineering and Technology*, 6(1), Article 1. https://doi.org/10.26877/asset.v6i1.17909

Anam, K., Rofi, D. N., & Meiyanti, R. (2023). Monitoring System for Temperature and Humidity Sensors in the Production Room Using Node-Red as the Backend and Grafana as the Frontend. *Journal of Systems Engineering and Information Technology (JOSEIT)*, 2(2), Article 2. https://doi.org/10.29207/joseit.v2i2.5222

Behutiye, W., Tripathi, N., & Isomursu, M. (2024). Adopting Scrum in Hybrid Settings, in a University Course Project: Reflections and Recommendations. *IEEE Access*, 12, 105633-105650. Scopus. https://doi.org/10.1109/ACCESS.2024.3434662

Bhanage, D. A., Pawar, A. V., Kotecha, K., & Abraham, A. (2023). Failure Detection Using Semantic Analysis and Attention-Based Classifier Model for IT Infrastructure Log Data. *IEEE Access*, 11, 108178-108197. https://doi.org/10.1109/ACCESS.2023.3319438

Broniewski, A., Tirmizi, M. I., Zimányi, E., & Sakr, M. (2023). *Using MobilityDB and Grafana for Aviation Trajectory Analysis*. undefined-undefined. https://doi.org/10.3390/engproc2022028017

Butarbutar, R. T. B. D., Sasmita, G. M. A., & Pratama, I. P. A. E. (2023). Development of a Notification-Based Network Security Monitoring System Using Network Development Life Cycle (NDLC). *JITTER : Jurnal Ilmiah Teknologi dan Komputer*, 4(3), 1933. https://doi.org/10.24843/JTRTI.2023.v04.i03.p01

Chan, Y. W., Fathoni, H., Yen, H. Y., & Yang, C. T. (2022). Implementation of a Cluster-Based Heterogeneous Edge Computing System for Resource Monitoring and Performance Evaluation. *IEEE Access*, 10, 38458-38471. https://doi.org/10.1109/ACCESS.2022.3166154

Ehrlinger, L., & Wöß, W. (2022). A Survey of Data Quality Measurement and Monitoring Tools. Frontiers in Big Data, 5, 850611. https://doi.org/10.3389/fdata.2022.850611

Erdei, R., & Toka, L. (2023). Minimizing Resource Allocation for Cloud-Native Microservices. *Journal of Network and Systems Management*, 31(2), 35. https://doi.org/10.1007/s10922-023-09726-3

Gaol, F. L., Santoso, S., & Matsuo, T. (2022). Design and development of the application monitoring the use of server resources for server maintenance. *Open Engineering*, 12(1), 524-538. Scopus. https://doi.org/10.1515/eng-2022-0055

Garcia, L. A., OliveiraJr, E., Leal, G. C. L., & Morandini, M. (2021). *A Unified Feature Model for Scrum Artifacts from a Literature and Practice Perspective*. 296-305. https://doi.org/10.5753/eres.2020.13740

Ghimire, D., & Charters, S. (2022). The Impact of Agile Development Practices on Project Outcomes. *Software*, 1(3), Article 3. https://doi.org/10.3390/software1030012

Hadikusuma, R. S., Lukas, L., & Bachri, K. O. (2023). Survey Paper: Optimization and Monitoring of Kubernetes Cluster using Various Approaches. *Sinkron*, 8(3), Article 3.

https://doi.org/10.33395/sinkron.v8i3.12424

Iqromullah, R., Khairil, K., & Suryana, E. (2023). Security System Implementation And Monitoring Networks At Sma N 10 City Of Bengkulu. *Jurnal Media Computer Science*, 2(2), Article 2. https://doi.org/10.37676/jmcs.v2i2.4431

Jani, Y. (2024). Unified Monitoring for Microservices: Implementing Prometheus and Grafana for Scalable Solutions. *Journal of Artificial Intelligence, Machine Learning and Data Science*, 2(1), 848-852. https://doi.org/10.51219/JAIMLD/yash-jani/206

Kadenic, M. D., Koumaditis, K., & Junker-Jensen, L. (2023). Mastering scrum with a focus on team maturity and key components of scrum. *Information and Software Technology*, 153, 107079. https://doi.org/10.1016/j.infsof.2022.107079

Kuang, J., Liu, J., Huang, J., Zhong, R., Gu, J., Yu, L., Tan, R., Yang, Z., & Lyu, M. R. (2024). Knowledge-aware Alert Aggregation in Large-scale Cloud Systems: A Hybrid Approach. *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, 369-380. https://doi.org/10.1145/3639477.3639745

Lamothe, M., Guéhéneuc, Y.-G., & Shang, W. (2021). A Systematic Review of API Evolution Literature. *ACM Computing Surveys*, 54, 1-36. https://doi.org/10.1145/3470133

Penkov, St., & Taneva, A. (2021). Chat Programs in the Frame of Control System. *IFAC-PapersOnLine*, 54(13), 52-56. https://doi.org/10.1016/j.ifacol.2021.10.417

Santosa, I., & Mulyana, R. (2023). The IT Services Management Architecture Design for Large and Medium-sized Companies based on ITIL 4 and TOGAF Framework. *JOIV : International Journal on Informatics Visualization*, 7(1), 30. https://doi.org/10.30630/joiv.7.1.1590

Santoso, G., Setiawan, J., & Sulaiman, A. (2023). Development of OpenAI API Based Chatbot to Improve User Interaction on the JBMS Website. *G-Tech: Jurnal Teknologi Terapan*, 7(4), Article 4. https://doi.org/10.33379/gtech.v7i4.3301

Sassa, A. C., Almeida, I. A. de, Pereira, T. N. F., & Oliveira, M. S. de. (2023). Scrum: A Systematic Literature Review. *International Journal of Advanced Computer Science and Applications*, 14(4), Article 4. https://doi.org/10.14569/IJACSA.2023.0140420

Simili, E., Stewart, G., Roy, G., Skipsey, S., & Britton, D. (2021). A hybrid system for monitoring and automated recovery at the Glasgow Tier-2 cluster. *EPJ Web of Conferences*, 251. https://doi.org/10.1051/epjconf/202125102047

Sun, Y., Ye, K., & Xu, C.-Z. (2020). PLMSys: A Cloud Monitoring System Based on Cluster Performance and Container Logs. En Q. Zhang, Y. Wang, & L.-J. Zhang (Eds.), Cloud *Computing – CLOUD 2020* (Vol. 12403, pp. 111-125). Springer International Publishing. https://doi.org/10.1007/978-3-030-59635-4_8

Yu, Q., Zhao, N., Li, M., Li, Z., Wang, H., Zhang, W., Sui, K., & Pei, D. (2024). A survey on intelligent management of alerts and incidents in IT services. *Journal of Network and Computer Applications*, 224, 103842. https://doi.org/10.1016/j.jnca.2024.103842