









ORIGINAL

## Microservices architecture in Azure for automated incident recovery

### Arquitectura de microservicios en Azure para la recuperación automatizada de incidentes

Juan Camilo Giraldo Mejia<sup>1</sup>  , Fabio Alberto Vargas Agudelo<sup>1</sup>  , Alejandro Restrepo Correa<sup>1</sup>  , Alicia Martínez Rebollar<sup>2</sup>  

<sup>1</sup>Tecnológico de Antioquia - Institución Universitaria, Antioquia. Medellín, Colombia.

<sup>2</sup>TECNM /Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México.

**Cite as:** Giraldo Mejia JC, Vargas Agudelo FA, Restrepo Correa A, Martínez Rebollar A. Microservices architecture in Azure for automated incident recovery. Salud, Ciencia y Tecnología. 2025; 5:1865. <https://doi.org/10.56294/saludcyt20251865>

**Submitted:** 13-01-2025

**Revised:** 28-03-2025

**Accepted:** 08-07-2025

**Published:** 09-07-2025

**Editor:** Prof. Dr. William Castillo-González 

**Corresponding author:** Juan Camilo Giraldo Mejia 

#### ABSTRACT

Nowadays, more organizations incorporate microservices in the implementation of their solutions. However, despite its benefits, this technological strategy poses challenges in incident recovery, since a failure in one component can quickly affect the entire system, making response capacity crucial to reduce downtime. This article proposes a microservices structure in Azure with the objective of optimizing incident recovery. The findings indicate that this structure makes it possible to significantly reduce the recovery time of critical incidents and improves the availability of the services offered through API Management.

**Keywords:** Azure; Microservices; Incident Recovery; Microservices Architecture.

#### RESUMEN

En la actualidad son más las organizaciones que incorporan microservicios en la implementación de sus soluciones. No obstante, a pesar de sus beneficios, esta estrategia tecnológica plantea desafíos en la recuperación de incidentes, dado que una falla en un componente puede afectar de manera rápida todo el sistema, lo que hace crucial la capacidad de respuesta para reducir el tiempo de inactividad. Este artículo propone una estructura de microservicios en Azure con el objetivo de optimizar la recuperación de incidentes. Los hallazgos señalan que esta estructura posibilita disminuir significativamente el tiempo de recuperación de incidentes críticos y mejora la disponibilidad de los servicios ofrecidos mediante la Administración de API.

**Palabras clave:** Azure; Microservicios; Recuperación de Incidentes; Arquitectura de Microservicios.

#### INTRODUCTION

Nowadays, more organizations are incorporating microservices into the implementation of their solutions.

<sup>(1)</sup> Microsoft Azure stands out for its improvements in infrastructure management.<sup>(2)</sup>

However, despite the inherent benefits of these technologies, the growing interconnection and distribution of system components introduce new challenges, particularly in the area of incident recovery. A failure in an individual microservice or underlying infrastructure component can spread quickly, affecting the availability and performance of the entire application.<sup>(3)</sup> The ability to respond effectively to these incidents, minimizing downtime and ensuring business continuity, becomes a critical priority for modern organizations.

Faced with this problem, the goal is to optimize the incident recovery procedure in infrastructures,

providing solutions that facilitate monitoring, management, and automation. In this way, companies will have an architecture capable of overcoming the difficulties that arise in the cloud.

According to Lenk A.<sup>(4)</sup>, organizations must plan and establish strategies for recovery from technological incidents and disasters. Protecting the IT infrastructure is essential to ensure its efficient operation. Complementing this, Aksakalli et al.<sup>(5)</sup> note that resource shortages can complicate operations, thereby increasing response times and workload.

The article is structured as follows: Section 1 presents an overview of related work on cloud incident mitigation and management. Section 2 presents the methodology applied to create the architecture. Section 3 presents the results obtained. Section 4 presents the discussion and concludes with the results.

## Related work

Aftab SA et al.<sup>(6)</sup> present a model-based framework for automating the provision of virtualized services and applications in cloud infrastructure. It features a master service orchestrator, a software-defined network controller, and an application controller to retrieve storage files and extract configuration templates, ensuring proper configuration and allocation of resources in data centers and addressing the first operator's time responsibility.

Ray K<sup>(7)</sup> presents an integration of components to create the incident response flow, establishing the necessary assets for the process. All of this is managed from an information repository that allows the inputs needed to be managed, ensuring the workflow runs efficiently.

Guo D et al.<sup>(8)</sup> propose an automatic system that eliminates the need for manual intervention in the configuration and generation of virtual resources. This is achieved by calling virtual machine information, shared storage, and floating IP addresses, which, in short, improves data security and integrity.

Li X et al.<sup>(9)</sup> present an automatic repair system designed to enhance the resilience of cloud applications. This adaptable and customizable system integrates with cloud management tools, enabling it to work seamlessly with a wide range of applications, from the most modern to those utilizing older technologies, such as network function virtualization (NFV). Designed to operate with *OpenStack*, it aims to provide a robust solution for companies, particularly in the telecommunications sector, to manage the status and recovery of their applications efficiently. The primary objective is to optimize cloud services and reduce administrator workload by automating problem resolution, thereby enhancing reliability. In essence, this automatic repair method highlights the importance of systems that effectively detect and correct faults, which, according to the authors, will significantly increase the reliability and performance of cloud applications.

Shamsuddeen et al.<sup>(10)</sup> propose an autonomous system that distributes workloads among microservices, mimicking decentralized management behavior. This system aims to enhance performance compared to centralized orchestrations by optimizing resource utilization and overall efficiency. Additionally, it emphasizes the significance of automated orchestration for deploying, scaling, and testing microservices. In summary, the study proposes an innovative approach to managing workloads in microservices, emphasizing autonomy and performance improvements in cloud environments to address the challenges posed by centralized systems.

As a result of this research, an algorithm is proposed, consisting of a two-step configuration cycle: creation (establishing initial connections) and Link/Unlink (connecting or disconnecting services as needed). To demonstrate their proposal, they created a tool that generates plans for real microservice architectures, modeled with the Abstract Behavior Specification (ABS) programming language. In conclusion, the authors simplify the practices of automatic microservice implementation and demonstrate that, despite the complexity of the problem, it can be effectively solved with the proposed methods and tools, helping to optimize resources and prevent issues during implementation.<sup>(11)</sup>

Beloki's book,<sup>(12)</sup> "The Art of Site Reliability Engineering," is an essential guide to the principles and practices of Site Reliability Engineering (SRE), highlighting the importance of reliability for user satisfaction. It describes best practices for designing robust software architectures that support failures. The book focuses on implementing applications using Microsoft Azure's Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). It also discusses the use of microservices, valued for their scalability and flexibility. A central theme is the creation of resilient applications that can handle unexpected problems without significant interruptions, which is a fundamental pillar of SRE. In summary, the book aims to educate readers on the fundamentals of SRE, architectural strategies for resilience, and their practical application through Azure cloud services.

Chaplia O and Klym H<sup>(13)</sup> propose an architecture for automatic self-healing based on microservices. Its purpose is to strengthen the fault tolerance of cloud microservices and ensure process flow.

Karn et al.<sup>(14)</sup> focus on the growing popularity of microservices, which, by dividing applications into small, connected components, complicate the network. They highlight that a failure in one microservice can spread and compromise the entire application, making rigorous testing of connection resilience essential. The authors propose a language- and architecture-independent system for testing and improving this resilience, helping administrators identify the cause of failures. The study demonstrates how *Istio*, a service mesh, is utilized to

control communication and simulate failures, while *Locust* is employed to simulate high user loads. For fault detection, tools such as *Jaeger* and *Grafana* are used, and for resolution, temporary backup connections, microservice scaling, and the use of “circuit breakers” are proposed to prevent saturation. Finally, a practical demonstration with the e-commerce application *Hipster Shop* on *Kubernetes* validates the effectiveness of the proposed system. In summary, the paper outlines the challenges of microservices, presents solutions using *Istio*, and explains how this research enhances application resilience.

## METHOD

Phases for creating the *Azure API Management* Recovery and Backup Architecture.

### Phase I: Comparative analysis of the reviewed work

Technology recovery and automation are fundamental pillars of modern IT management, especially in cloud computing environments and microservice architectures. Although the different works presented address these areas from different perspectives, they share a common goal: to ensure business continuity and system resilience.

#### *Key similarities between the works*

One of the most notable similarities is the focus on automation for disaster recovery and service management. Works such as <sup>(9,15,16)</sup> propose frameworks and systems that eliminate manual intervention in critical processes, including service provisioning, recovery system deployment, and application repair. This automation is viewed as a means to enhance efficiency, minimize human error, and expedite response times.

Resource optimization and operational efficiency are cross-cutting objectives <sup>(17)</sup> for resource allocation and workload distribution.

#### *Differences and specific approaches*

- Scope and granularity of recovery: Lenk A and Ray K<sup>(4,7)</sup> propose an incident response architecture that articulates resource specifications for the proper execution of system recovery in the event of situations that affect process flow. The ability of applications to repair themselves automatically is highlighted.<sup>(9)</sup>
- Type of automation: Aftab SA et al.<sup>(15)</sup> propose a model-based framework for automating virtualized service provisioning and orchestration, focusing on the proper configuration of resources.<sup>(16)</sup> It focuses on the automatic implementation of recovery systems in cloud platforms, eliminating manual intervention in the configuration of virtual resources. The difference lies in whether automation is applied to initial provisioning and orchestration, or specifically to the activation of a recovery system.
- Application context: while some work focuses on IT recovery in a broad sense,<sup>(4,7)</sup> others specialize in cloud environments <sup>(6,8,9,13,14)</sup> and microservices.<sup>(10,11,12,13,14)</sup> This reflects the evolution of IT infrastructure and the need for specialized solutions tailored to these architectures.
- Resilience mechanisms: Beloki U<sup>(12)</sup> explores the principles of Site Reliability Engineering (SRE) and resilient architectures in *Azure*, providing comprehensive guidance on how to build robust applications.<sup>(14)</sup> focus on automated testing and the resilience of microservice networks with *Istio*, using specific tools to simulate failures and improve communication between microservices. The differences lie in the methodology for achieving resilience, whether through design principles or testing and traffic control tools.
- Workload management: Shamsuddeen R et al.<sup>(10)</sup> propose an autonomous system for workload distribution in containerized microservices, mimicking the behavior of a swarm for decentralized management. This is a more specific concern within the realm of microservice resilience and performance.

All of the work converges on the importance of recovery and resilience in the IT sector. Still, it differs in its scope, the type of automation proposed, the specific technological context (cloud, microservices), and the particular mechanisms employed to achieve its goals. The general trend is toward increasingly automated and self-healing systems, especially in the dynamic environment of cloud computing and microservice architectures.

### Phase II: identified issues

#### *Fault Management and Resilience in Cloud Applications and Microservices*

In microservice architectures, a failure in one component can propagate and compromise the entire application, making rigorous testing of network connection resilience essential.<sup>(14)</sup> Cloud applications, particularly in telecommunications environments, require robust self-healing systems to effectively detect and correct failures, as well as optimize services.<sup>(9)</sup>

### Decentralized Workload Management

Centralized orchestration systems can be inefficient; autonomous distribution of workloads across containerized microservices seeks to improve performance and efficiency compared to centralized orchestrations.<sup>(10)</sup>

### Challenges in Microservice Implementation

Although microservices offer scalability and flexibility, their automatic implementation is a complex problem.

<sup>(11)</sup> This requires specialized methods and tools to optimize resources and prevent issues during deployment.

### Ensuring Reliability and High Availability

The critical need for high availability and self-healing in cloud applications poses a significant challenge to ensuring service continuity, particularly in vital sectors such as banking and healthcare.<sup>(13)</sup>

### Phase III: Proposed solution to the problem

Taking into account the needs of organizations seeking robust solutions to efficiently address and recover from incidents, we propose an API Management architecture supported by Azure. The solution is designed based on the problems identified in the literature review, and the following elements are proposed to structure it.

- Platform that supports cloud services
- Service that manages access to the platform
- Service that automates and controls tasks
- Service to create and control workflows
- Resource status monitoring and analysis service
- Service for creating and managing APIs
- Data storage service

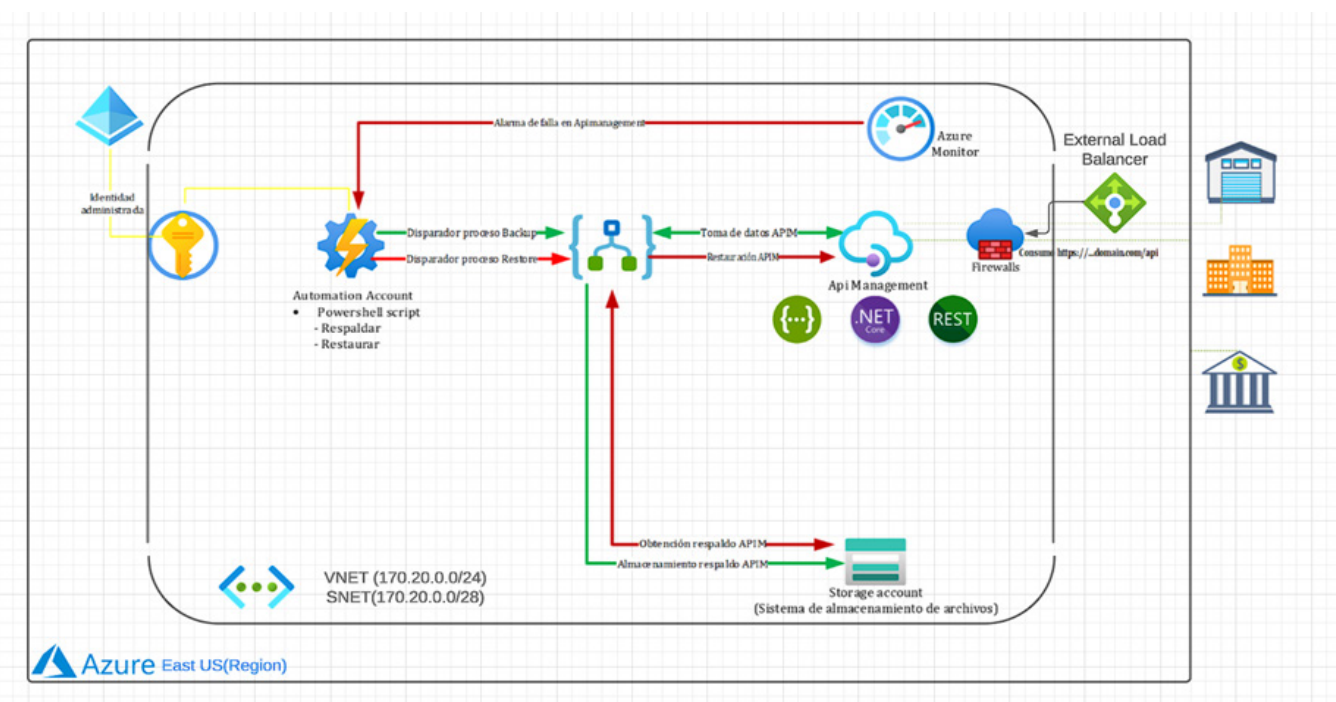
## RESULTS

### Proposed architecture

In this solution, Azure technology is used as a centralized identity and enterprise policy management system, enabling unified employee access to applications, including authorization to *API* disaster recovery components.

Its main objective is to back up *API* containers and, if there is a problem or human error, recreate the infrastructure using code and quickly launch the *APIs* so that the service is back up and running as soon as possible.

The proposed solution is based on pay-per-use technologies offered by Microsoft Azure, see figure 1.



**Figure 1.** Architecture components

## Architecture elements and functionalities

**Azure Cloud:** a platform that supports Microsoft cloud services.

**Azure AD Manage Identity:** a service that manages identities and access in Azure.

**Key Vault:** key and secret management service in Azure.

**Automation account:** Task automation service in Azure.

**Logic Apps:** A Service for creating automated workflows in Azure.

**Azure Monitor:** Resource monitoring and analysis service in Azure.

**Storage account:** Data storage service in Azure.

**API Management:** Service for creating, publishing, and managing APIs in Azure.

Azure EntraID (formerly Azure Active Directory) is a corporate identity provider widely used in the business world for centralized user management and policy assignment. This enables all employees to access the company's various applications using the same corporate username and password. In this case, it is used to access and authorize API disaster recovery components.

The key store is used to securely store and consume the secret (password) of an identity (service principal) used to take copies of APIs. This store also stores connection strings that can be used to consume other APIs or database components.

Figure 2 shows the business solution to a massive failure, characterized by an automated flow of activities. By eliminating manual intervention, the risks of errors associated with service provider actions are minimized. Additionally, the comprehensive recovery of all components in a single process is ensured, resulting in a significant reduction in restoration times and, consequently, increased application availability.

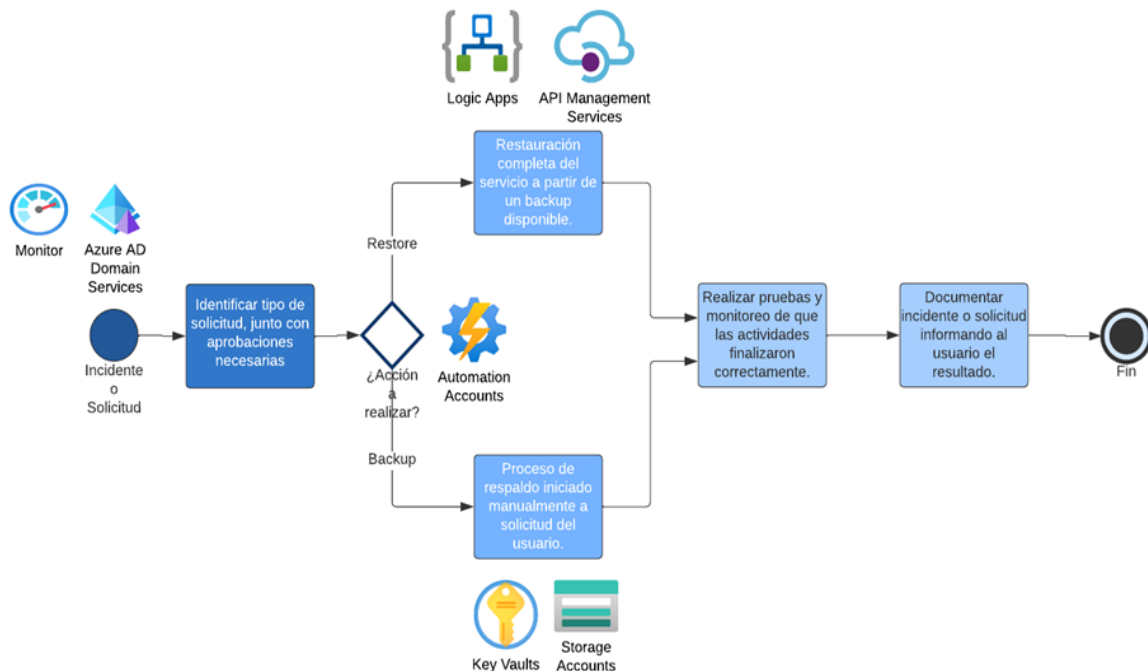


Figure 2. Automated business flow

**Access and authorization:** access is granted at the Azure AD level (Sign-in ID) with a corporate username and password with multi-factor authentication, an access module, and modules depending on the assigned permission level.

**Security:** includes a permissions module and storage for sensitive application data.

**Storage:** module where backup files are physically stored.

**Automation:** automation account module to view the scripts executed by the tool, manual execution, or code modification.

### Flow specification

Figure 3 illustrates the user's journey when interacting with the solution. Access is possible from any device connected to the internet, such as computers, tablets, or mobile phones. Login is performed with the client's corporate credentials, adhering to established access policies. The permissions assigned vary according to the user profile and can be read-only, collaboration, or full administration. Within the portal, the user can interact with the APIs (at the container level), manage automation, configure monitoring alerts, review performance graphs, or verify the correct frequency of data storage.



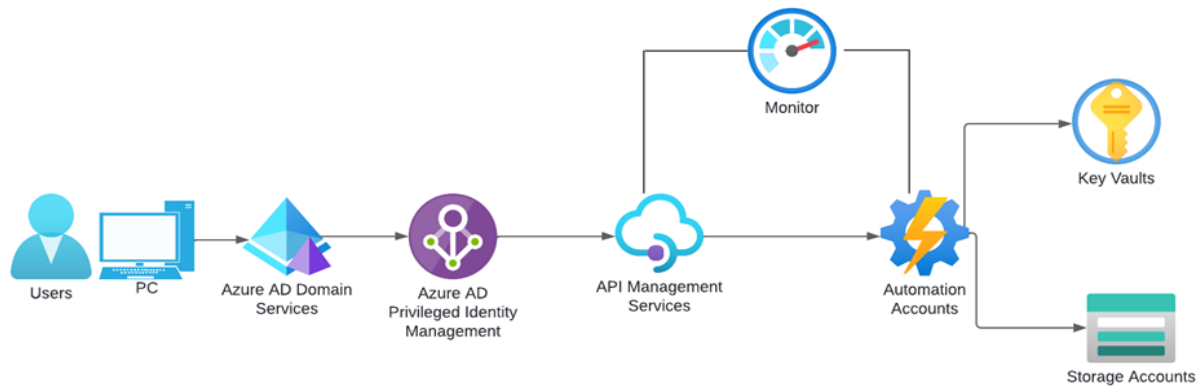


Figure 3. User interaction flow

## DISCUSSION

Tests on the proposed architecture demonstrate a reduction in the time required to respond to incidents, overcoming high workloads in decentralized management. It is a scalable and flexible solution that overcomes problems presented in works such as that of Bravetti M.<sup>(11)</sup> In addition, it demonstrated high reliability and availability, easily adapting to any scope, sector, or area, as seen in the work of Chaplia O and Klym H.<sup>(13)</sup>

## CONCLUSIONS

- Organizations must deploy efficient and reliable microservice systems in the cloud for disaster recovery purposes. They must have the ability to detect and respond to failures quickly, enabling rapid recovery and minimizing the impact on users and the organization's operations.
- Azure microservices offer the inherent advantages of microservices, such as fault isolation, redundancy, and isolated deployment. This facilitates monitoring, automation, and availability, resulting in more capable systems that ensure the organization's dynamics and its applications remain available.
- Automation is established as a fundamental pillar for improving operational resilience and incident recovery efficiency, especially in cloud and microservice environments. The architectural proposal demonstrates how eliminating manual intervention in critical processes—from provisioning and orchestration to repair and *API* backup—significantly reduces human error and speeds up response times. Automated orchestration of backup and restore tasks, facilitated by services such as *Azure Automation Account and Logic Apps*, is essential for achieving comprehensive and rapid recovery from massive failures.

## REFERENCES

1. Newman S. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media; 2020.
2. Microsoft. Azure Well-Architected Framework [Internet]. 2024. Disponible en: <https://learn.microsoft.com/en-us/azure/architecture/framework/>
3. Vogels W. A Decade of Amazon's CTO: The 10 Lessons. ACM Queue. 2016;14(7).
4. Lenk A. Cloud Standby deployment: a model-driven deployment method for disaster recovery in the cloud. In: 2015 IEEE 8th International Conference on Cloud Computing; 2015 Jun 27-Jul 2; New York, NY, USA. Piscataway (NJ): IEEE; 2015. p. 933-40. doi: 10.1109/CLOUD.2015.127.
5. Aksakalli IK, Celik T, Can AB, Tekinerdogan B. A model-driven architecture for automated deployment of microservices. Applied Sciences. 2021;11(20):9617. doi: 10.3390/app11209617.
6. Aftab SA, Jana R, Farooqui K, Murray JF, Gilbert ME, inventores; U.S. Patent and Trademark Office, asignatario. U.S. Patent No. 11,223,536. 11 de enero de 2022.
7. Ray K, inventor. Automatic system disaster recovery. European Patent Office, applicant. Patent WO2017066383A1. 2017 Apr 20.
8. Guo D, Wang W, Zeng G, Wei Z. Microservices architecture based cloudware deployment platform for service computing. In: 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE); 2016 Mar 27-Apr

1; Oxford, UK. Piscataway (NJ): IEEE; 2016. p. 358-63. doi: 10.1109/SOSE.2016.22.

9. Li X, Li K, Pang X, Wang Y. An orchestration based cloud auto-healing service framework. In: 2017 IEEE International Conference on Edge Computing (EDGE); 2017 Jun 25-30; Honolulu, HI, USA. Piscataway (NJ): IEEE; 2017. p. 190-3. doi: 10.1109/IEEE.EDGE.2017.33.

10. Shamsuddeen R, Rabiou S, Abba A, Abubakar MA. Autonomous workload distribution for container-based micro services environments. World Journal of Advanced Engineering Technology and Sciences. 2023 [cited 2025 Jun 3];9(2):[about 7 p.]. Available from: <https://doi.org/10.30574/wjaets.2023.9.2.0226>

11. Bravetti M, Giallorenzo S, Mauro J, Talevi I, Zavattaro G. Optimal and Automated Deployment for Microservices. In: Service-Oriented Computing - ICSOC 2018 Workshops: WESOA, CSB, DC4SCC, FOCAS, IFCT, IWSOA, SMGS, and WoC. Proceedings; 2019. Cham (Switzerland): Springer; 2019. p. 351-68. (Lecture Notes in Computer Science; vol 11424). doi: 10.1007/978-3-030-16722-6\_21.

12. Beloki U. The art of site reliability engineering (SRE) with Azure: building and deploying applications that endure. New York (NY): Apress; 2022. doi: 10.1007/978-1-4842-8704-0.

13. Chaplia O, Klym H. An approach for automatic self-recovery for a Node.js microservice. In: 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT); 2023 Oct; Athens, Greece. Piscataway (NJ): IEEE; 2023. p. 1-4. doi: 10.1109/dessert61349.2023.10416461.

14. Karn RR, Das R, Pant DR, Heikkonen J, Kanth RK. Automated Testing and Resilience of Microservice's Network-link using Istio Service Mesh. In: 2022 31st Conference of Open Innovations Association (FRUCT); 2022 Apr 27-29; Helsinki, Finland. Piscataway (NJ): IEEE; 2022. p. 79-88. doi: 10.23919/FRUCT54823.2022.9770890.

15. Aftab SA, Jana R, Farooqui K, Murray JF, Gilbert ME, inventors; U.S. Patent and Trademark Office, assignee. Single, logical, multi-tier application blueprint used for deployment and management of multiple physical applications in a cloud environment. US Patent 11,223,536. 2022 Jan 11.

16. Guo D, Wang W, Zeng G, Wei Z. Microservices architecture based cloudware deployment platform for service computing. In: 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE); 2016 Mar 29-Apr 2; Oxford, UK. Piscataway (NJ): IEEE; 2016. p. 358-63. doi: 10.1109/SOSE.2016.22.

17. Aksakalli IK, Celik T, Can AB, Tekinerdogan B. A model-driven architecture for automated deployment of microservices. Applied Sciences. 2021;11(20):9617.

18. Lenk A. Cloud Standby deployment: a model-driven deployment method for disaster recovery in the cloud. In: 2015 IEEE 8th International Conference on Cloud Computing (CLOUD); 2015 Jun 27-Jul 2; New York, NY, USA. Piscataway (NJ): IEEE; 2015. p. 933-40. doi: 10.1109/CLOUD.2015.127.

## FUNDING

None.

## CONFLICT OF INTEREST

None.

## AUTHOR CONTRIBUTION

*Conceptualization:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alicia Martínez Rebollar.

*Data collection and analysis:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.

*Formal analysis:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.

*Research:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa, Alicia Martínez Rebollar.

*Methodology:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.

*Project management:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.

*Resources:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.

*Architecture:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa.

*Supervision:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.

*Validation:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa, Alicia Martínez

Rebollar.

*Writing - original draft:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa.

*Writing - review and editing:* Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa.