Salud, Ciencia y Tecnología. 2025; 5:1865
doi: 10.56294/saludcyt20251865

ORIGINAL

Microservices architecture in Azure for automated incident recovery

Arquitectura de microservicios en Azure para la recuperación automatizada de incidentes

Juan Camilo Giraldo Mejia[1], Fabio Alberto Vargas Agudelo[1], Alejandro Restrepo Correa[1], Alicia Martínez Rebollar[2]

[1]Tecnológico de Antioquia – Institución Universitaria, Antioquia. Medellín, Colombia.
[2]TECNM /Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México

Citar como: 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

Enviado: 13-01-2025 **Revisado:** 28-03-2025 **Aceptado:** 08-07-2025 **Publicado:** 09-07-2025

Editor: Prof. Dr. William Castillo-González

Autor para la correspondencia: 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.

INTRODUCCIÓN

En la actualidad son más las organizaciones que incorporan microservicios en la implementación de sus soluciones.(1) Se resalta Microsoft Azure para mejor la gestión de la infraestructura.(2)

Sin embargo, a pesar de los beneficios inherentes a estas tecnologías, la creciente interconexión y distribución de los componentes del sistema introducen nuevos desafíos, particularmente en el ámbito de la recuperación de incidentes. Un fallo en un microservicio individual o en un componente de infraestructura subyacente puede propagarse rápidamente, afectando la disponibilidad y el rendimiento de la aplicación en su totalidad.(3) La capacidad de responder eficazmente a estos incidentes, minimizando el tiempo de inactividad y asegurando la continuidad del negocio, se convierte en una prioridad crítica para las organizaciones modernas.

Frente a esta problemática se busca optimizar el procedimiento para la recuperación de incidentes en infraestructuras. Con soluciones que faciliten el seguimiento, gestión y automatización. De esta manera las empresas contaran con una arquitectura capaz de superar las dificultades que se originen en la nube.

Según Lenk[4], es crucial que las organizaciones planifiquen y establezcan estrategias para la recuperación ante incidentes y desastres tecnológicos. Proteger la infraestructura de TI es esencial para asegurar su funcionamiento eficiente. Complementando esto, Aksakalli IK et al.[5], señalan que la escasez de recursos puede complicar las operaciones y aumentar los tiempos de respuesta y la carga de trabajo.

Aftab et al.[6] presentan un marco basado en modelos para automatizar la provisión de los servicios y aplicaciones virtualizadas en la infraestructura de la nube. Cuenta con un orquestador maestro de servicios, un controlador de red definido por software, y un controlador de aplicaciones para recuperar archivos de almacenamiento y extraer plantillas de configuración para garantizar la adecuada configuración y asignación de los recursos en centros de datos, abordando así el primer operador de tiempo responsable.

Ray[7] Presenta una integración de componentes para crear el flujo de atención de incidentes, estableciendo los activos necesarios para el proceso. Todo esto administrado desde un repositorio de información que permite gestionar los insumos necesarios para que el flujo de trabajo corra de manera eficiente.

Guo et al.[8] proponen un sistema automático que elimina la necesidad de intervención manual en la configuración y generación de recursos virtuales. Esto se logra mediante la llamada de información de máquinas virtuales, almacenamiento compartido y direcciones IP flotantes, lo que, en resumen, mejora la seguridad y la integridad de los datos.

Li et al.[9] presentan un sistema de reparación automática diseñado para mejorar la capacidad de recuperación de las aplicaciones en la nube. Este sistema, adaptable y personalizable, se integra con las herramientas de gestión de la nube, permitiendo su funcionamiento con diversas aplicaciones, desde las más modernas hasta las que utilizan tecnologías antiguas como la virtualización de funciones de red (NFV). Diseñado para operar con *OpenStack*, busca ofrecer una solución robusta para que empresas, especialmente del sector de telecomunicaciones, gestionen eficientemente el estado y la recuperación de sus aplicaciones. El objetivo principal es optimizar los servicios en la nube y reducir la carga de trabajo de los administradores al automatizar la resolución de problemas, mejorando así la confiabilidad. En esencia, este método de reparación automática subraya la importancia de sistemas que detecten y corrijan fallos de manera efectiva, lo que, según los autores, incrementará significativamente la confiabilidad y el rendimiento de las aplicaciones en la nube.

Shamsuddeen et al.[10] proponen un sistema autónomo que distribuye las cargas de trabajo entre microservicios, imitando el comportamiento para una gestión descentralizada. Este sistema busca mejorar el rendimiento en comparación con las orquestaciones centralizadas al optimizar el uso de recursos y la eficiencia general. Además, destaca la importancia de la orquestación automatizada para el despliegue, escalado y prueba de microservicios. En síntesis, el estudio propone una forma innovadora de gestionar las cargas de trabajo en microservicios, enfatizando la autonomía y las mejoras de rendimiento en entornos de nube para superar los desafíos de los sistemas centralizados.

Como resultado de esta investigación, se propone un algoritmo, que consiste en un ciclo de configuración de dos pasos: creación (establecer conexiones iniciales) y Vincular/Desvincular (conectar o desconectar servicios según sea necesario). Para demostrar su propuesta, crearon una herramienta que genera planes para arquitecturas de microservicios reales, modeladas con el lenguaje de programación Abstract Behavior Specification (ABS). En conclusión, los autores simplifican las prácticas de implementación automática de microservicios y demuestran que, a pesar de la complejidad del problema, puede resolverse eficazmente con los métodos y herramientas propuestos, lo que ayuda a optimizar los recursos y prevenir problemas durante la implementación.[11]

El libro de Beloki[12] "The Art of Site Reliability Engineering", es una guía esencial sobre los principios y prácticas de la Ingeniería de Confiabilidad del Sitio (SRE), destacando la importancia de la confiabilidad para la satisfacción del usuario. Describe las mejores prácticas para diseñar arquitecturas de software robustas que soportan fallos. El libro se enfoca en implementar aplicaciones usando Infraestructura como Servicio (IaaS) y Plataforma como Servicio (PaaS) de Microsoft Azure. Además, analiza el uso de microservicios, valorados por su escalabilidad y flexibilidad. Un tema central es la creación de aplicaciones resilientes, capaces de manejar problemas inesperados sin interrupciones significativas, lo cual es un pilar fundamental de SRE. En resumen, el libro tiene como objetivo educar a los lectores sobre los fundamentos de SRE, las estrategias arquitectónicas para la resiliencia y su aplicación práctica a través de los servicios en la nube de Azure.

Chaplia et al.[13] proponen una arquitectura para la autorrecuperación automática basada en microservicios. Tiene como propósito fortalecer la capacidad de solución ante fallos de los microservicios en nube, y garantizar el flujo del proceso.

Karn et al.[14] se centran en la creciente popularidad de los microservicios, que, al dividir las aplicaciones en componentes pequeños y conectados, complican la red. Resalta que un fallo en un microservicio puede propagarse y comprometer toda la aplicación, haciendo esenciales pruebas rigurosas de la resiliencia de las

conexiones. Los autores proponen un sistema independiente del lenguaje y la arquitectura para probar y mejorar esta resiliencia, ayudando a los administradores a identificar la causa de los fallos. El estudio muestra cómo se utiliza *Istio*, una malla de servicios, para controlar la comunicación y simular fallos, mientras que *Locust* se emplea para simular cargas elevadas de usuarios. Para la detección de fallos, se usan herramientas como *Jaeger* y *Grafana*, y para la resolución, se plantean conexiones temporales de respaldo, escalado de microservicios y el uso de "interruptores" para evitar saturación. Finalmente, una demostración práctica con la aplicación de comercio electrónico *Hípster Shop* en *Kubernetes* valida la efectividad del sistema propuesto. En resumen, el documento explica los desafíos de los microservicios, las soluciones con *Istio* y cómo esta investigación mejora la resiliencia de las aplicaciones.

MÉTODO

Fases para la creación de la Arquitectura de Recuperación y Respaldo de *Azure API Management*

Fase I: análisis comparativo de los trabajos revisados

La recuperación tecnológica y la automatización son pilares fundamentales en la gestión moderna de TI, especialmente en entornos de computación en la nube y arquitecturas de microservicios. Aunque los diferentes trabajos presentados abordan estas áreas desde diversas perspectivas, comparten un objetivo común: garantizar la continuidad del negocio y la resiliencia de los sistemas.

Similitudes clave entre los trabajos

Una de las similitudes más destacadas es el enfoque en la automatización para la recuperación ante desastres y la gestión de servicios. Trabajos[15,16,9], proponen marcos y sistemas que eliminan la intervención manual en procesos críticos como el aprovisionamiento de servicios, el despliegue de sistemas de recuperación y la reparación de aplicaciones. Esta automatización es vista como un medio para mejorar la eficiencia, reducir errores humanos y acelerar los tiempos de respuesta.

La optimización de recursos y la eficiencia operativa son objetivos transversales[17] para la asignación de recursos, y para la distribución de carga de trabajo.

Diferencias y enfoques específicos

- Alcance y granularidad de la recuperación: Lenk[4] y Ray[7] proponen una arquitectura de atención a incidentes, que articula especificación de recursos para una adecuada ejecución de recuperar el sistema en caso de presentarse situaciones que afecten el flujo del proceso. Se resalta la capacidad de las aplicaciones para repararse de forma automática.[9]
- Tipo de automatización: Aftab et al.[15] proponen un marco basado en modelos para la automatización del aprovisionamiento y orquestación de servicios virtualizados, centrándose en la configuración adecuada de recursos.[16] Se enfoca en la implementación automática de sistemas de recuperación en plataformas en la nube, eliminando la intervención manual en la configuración de recursos virtuales. La diferencia radica en si la automatización se aplica al aprovisionamiento inicial y la orquestación, o específicamente a la activación de un sistema de recuperación.
- Contexto de aplicación: mientras que algunos trabajos se centran en la recuperación de TI en un sentido amplio,[4,7] otros se especializan en entornos de nube[6,8,9,13,14] y microservicios[10,11,12,13,14]. Esto refleja la evolución de la infraestructura de TI y la necesidad de soluciones específicas para estas arquitecturas.
- Mecanismos de resiliencia: Beloki[12] explora los principios de Ingeniería de Confiabilidad del Sitio (*SRE*) y arquitecturas resilientes en *Azure*, proporcionando una guía integral sobre cómo construir aplicaciones robustas.[14] Se centran en pruebas automatizadas y la resiliencia de redes de microservicios con *Istio*, utilizando herramientas específicas para simular fallos y mejorar la comunicación entre microservicios. Las diferencias radican en la metodología para lograr la resiliencia, ya sea a través de principios de diseño o de herramientas de prueba y control de tráfico.
- Gestión de carga de trabajo: Shamsuddeen et al.[10] proponen un sistema autónomo para la distribución de carga de trabajo en microservicios contenerizados, imitando el comportamiento de un enjambre para una gestión descentralizada. Esta es una preocupación más específica dentro del ámbito de la resiliencia y el rendimiento de los microservicios.

Todos los trabajos convergen en la importancia de la recuperación y la resiliencia en TI, pero difieren en la amplitud de su alcance, el tipo de automatización propuesta, el contexto tecnológico específico (nube, microservicios) y los mecanismos particulares empleados para lograr sus objetivos. La tendencia general es hacia sistemas cada vez más automatizados y auto-recuperables, especialmente en el dinámico entorno de la computación en la nube y las arquitecturas de microservicios.

Fase II: problemáticas identificadas

Gestión de Fallos y Resiliencia en Aplicaciones en la Nube y Microservicios

En arquitecturas de microservicios, un fallo en un componente puede propagarse y comprometer toda la aplicación, haciendo esenciales las pruebas rigurosas de la resiliencia de las conexiones de red.[14] Las aplicaciones en la nube, especialmente en entornos de telecomunicaciones, necesitan sistemas robustos de reparación automática para detectar y corregir fallos eficazmente y optimizar los servicios.[9]

Gestión Descentralizada de Cargas de Trabajo

Los sistemas centralizados de orquestación pueden ser ineficientes, la distribución autónoma de cargas de trabajo en microservicios contenerizados busca mejorar el rendimiento y la eficiencia en comparación con las orquestaciones centralizadas.[10]

Desafíos en la Implementación de Microservicios

Aunque los microservicios ofrecen escalabilidad y flexibilidad, su implementación automática es un problema complejo.[11] Esto requiere métodos y herramientas especializadas para optimizar los recursos y prevenir problemas durante el despliegue.

Garantizar la Confiabilidad y Alta Disponibilidad

La necesidad crítica de alta disponibilidad y autorrecuperación en aplicaciones en la nube es un desafío importante para asegurar la continuidad del servicio, especialmente en sectores críticos como la banca y la salud.[13]

Fase III: propuesta de solución a la problemática

Teniendo en cuenta las necesidades de las organizaciones en miras de contar con soluciones sólidas para atender y recuperar de forma eficiente incidentes, se propone una arquitectura para API Management soportada en Azure. La solución está pensada desde las problemáticas identificadas en la revisión de literatura, y se proponen los siguientes elementos que permitirán estructurarla.

- Plataforma que soporte servicios en la nube.
- Servicio que administre los accesos a la plataforma.
- Servicio que automatice y controle las tareas.
- Servicio para crear y controlar flujos de trabajo.
- Servicio de supervisión y análisis de estados de recursos.
- Servicio para crear y administrar Apis.
- Servicio de almacenamiento de Datos.

RESULTADOS Y DISCUSIÓN

Arquitectura propuesta

En esta solución la tecnología Azure se utiliza como un sistema centralizado de gestión de identidades y políticas empresariales, permitiendo el acceso unificado de los empleados a las aplicaciones, incluyendo la autorización a los componentes de recuperación de desastres de las *APIs*.

Su objetivo principal es hacer copias de seguridad de los contenedores de las *APIs* y, si hay algún problema o error humano, volver a crear la infraestructura usando código y poner en marcha las *APIs* rápidamente para que el servicio vuelva a funcionar lo antes posible.

La solución planteada se basa en tecnologías de pago por uso ofrecidas por Microsoft Azure, véase la figura 1.

Elementos y Funcionalidades de la arquitectura

Azure Cloud: plataforma que soporta los servicios en la nube de Microsoft.
Azure AD Manage Identity: servicio que administra las identidades y accesos en Azure.
Key Vault: servicio de gestión de claves y secretos en Azure.
Automation account: Servicio de automatización de tareas en Azure.
Logic Apps: servicio para crear flujos de trabajo automatizados en Azure.
Azure Monitor: servicio de supervisión y análisis de recursos en Azure.
Storage account: servicio de almacenamiento de datos en Azure.
API Management: servicio para crear, publicar y administrar Apis en Azure.

Azure EntraID (Antes Azure Active Directory) es un proveedor de identidades corporativas muy usado en el ámbito empresarial para el manejo centralizado de usuarios y asignación de políticas, esto permite que todos los empleados accedan a las diferentes aplicaciones de la compañía con su mismo usuario y contraseña corporativo, en este caso es usado para tener acceso y autorización a los componentes de recuperación de

desastres de apis.

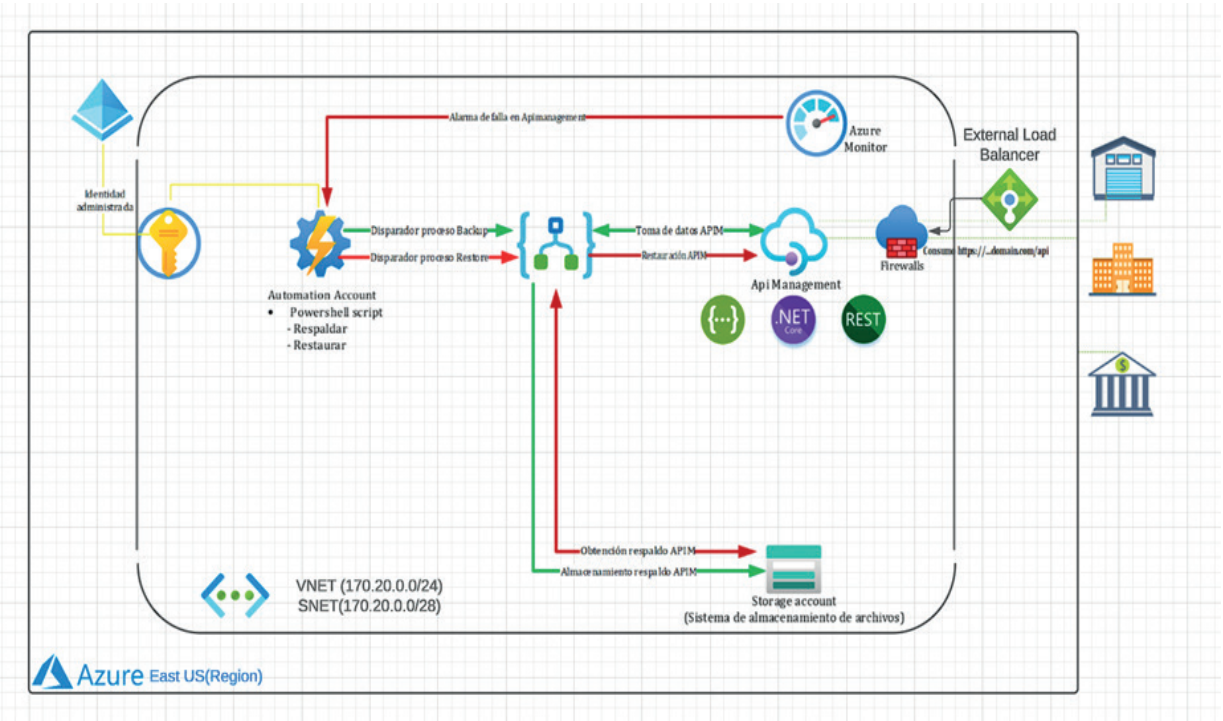

Figura 1. Componentes de la arquitectura

El almacén de claves se usa para guardar y consumir de forma segura el secreto (contraseña) de una identidad (service principal) usada para tomar las copias de las apis, en este almacén también se guardan cadenas de conexión que pueden ser para consumir otras apis o componentes de bases de datos.

La figura 2 representa la solución de negocio ante una falla masiva, caracterizada por un flujo de actividades automatizado. Al eliminar la intervención manual, se minimizan los riesgos de errores asociados a las acciones de los proveedores de servicios. Además, se asegura la recuperación integral de todos los componentes en un solo proceso, lo que conlleva una reducción significativa en los tiempos de restauración y, por ende, una mayor disponibilidad de las aplicaciones.

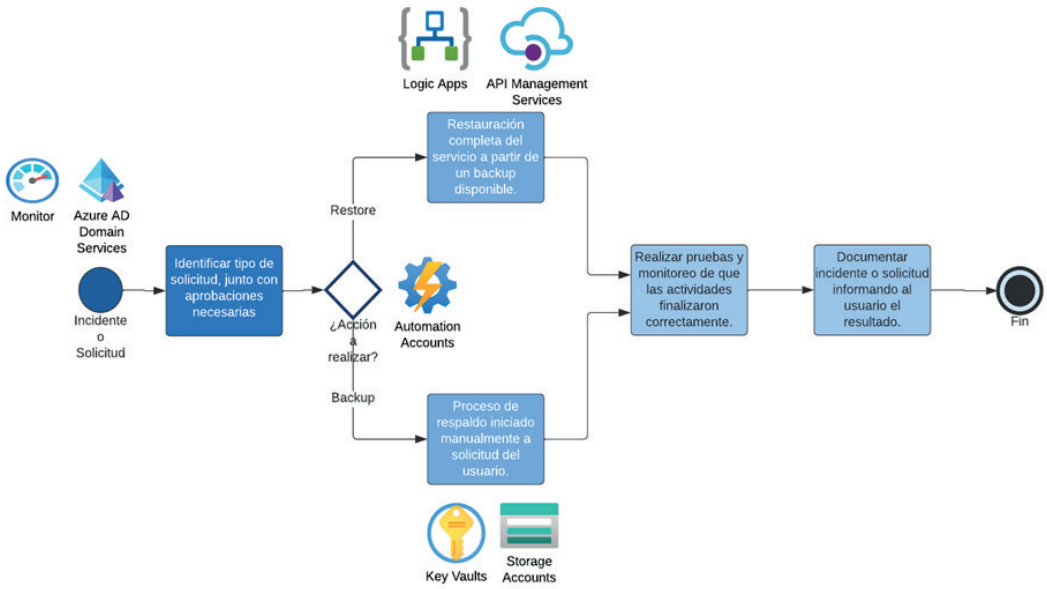

Figura 2. Flujo de negocio automatizado

Acceso y autorización: se da a nivel de Azure AD (Entra ID) con usuario y contraseña corporativo con autenticación multifactor, módulo de acceso y brinda módulos dependiendo el nivel de permisos asignado.

Seguridad: cuenta con un módulo de permisos y almacenamiento de datos sensibles de la aplicación.

Almacenamiento: módulo donde se guardan físicamente los archivos de copias de seguridad.

Automatización: módulo de cuenta de automatización, para visualizar los scripts que ejecuta la herramienta, ejecución manual o modificación de código.

Especificación del flujo

La figura 3 ilustra el recorrido del usuario al interactuar con la solución. El acceso es posible desde cualquier dispositivo conectado a internet, como computadoras, tabletas o teléfonos móviles. El inicio de sesión se realiza con las credenciales corporativas del cliente, adhiriéndose a las políticas de acceso establecidas. Los permisos asignados varían según el perfil del usuario, pudiendo ser de solo lectura, colaboración o administración total. Dentro del portal, el usuario puede interactuar con las *APIs* (a nivel de contenedor), administrar la automatización, configurar alertas de monitoreo, revisar gráficos de rendimiento o verificar la correcta periodicidad del almacenamiento de datos.

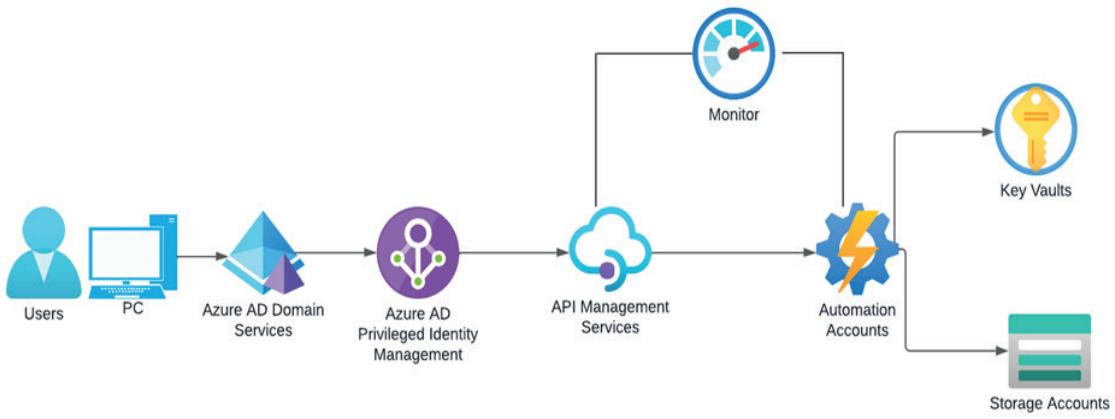

Figura 3. Flujo de interacción con el usuario

Las pruebas a la arquitectura propuesta logran evidenciar la reducción de tiempos necesarios para la atención de incidentes, superando en la gestión descentralizada altas cargas de trabajo. Es una solución escalable y flexible que supera problemáticas presentadas en trabajos como el de Bravetti.[11] Además, demostró alta confiabilidad y disponibilidad, adaptándose fácilmente a cualquier alcance, y sector o área, por ejemplo, los presentados en el trabajo de Chaplia et al.[13]

CONCLUSIONES

Para las organizaciones es muy importante desplegar sistemas de microservicios en la nube para la recuperación de desastres, sistemas que sean eficientes y confiables. Deben tener la capacidad para detectar y responder rápidamente a las fallas, permitiendo una recuperación rápida y minimizando el impacto en los usuarios y en la operación de la organización.

Los microservicios Azure presentan ventajas inherentes de los microservicios, como el aislamiento de fallos, la redundancia y el despliegue aislado. Esto facilita el seguimiento, automatización y disponibilidad, generando sistemas más capaces, garantizando la dinámica de la organización y sus aplicaciones disponible.

La automatización se establece como un pilar fundamental para la mejora de la resiliencia operativa y la eficiencia en la recuperación de incidentes, especialmente en entornos de nube y microservicios. La propuesta arquitectónica demuestra cómo la eliminación de la intervención manual en procesos críticos —desde el aprovisionamiento y la orquestación hasta la reparación y el respaldo de *APIs*— reduce significativamente los errores humanos y acelera los tiempos de respuesta. La orquestación automatizada de tareas de respaldo y restauración, facilitada por servicios como *Azure Automation Account y Logic Apps*, es esencial para lograr una recuperación integral y rápida ante fallas masivas.

REFERENCIAS BIBLIOGRÁFICAS

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, Rabiu 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.

FINANCIACIÓN

Ninguna.

CONFLICTO DE INTERESES

Ninguno.

CONTRIBUCIÓN DE AUTORÍA

Conceptualización: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alicia Martínez Rebollar.
Obtención y análisis de datos: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.
Análisis formal: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.
Investigación: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa, Alicia Martínez Rebollar.
Metodología: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.
Administración del proyecto: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.
Recursos: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.
Arquitectura: Fabio Alberto Vargas, Juan Camilo Giraldo Mejia, Alejandro Restrepo Correa
Supervisión: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía.
Validación: Fabio Alberto Vargas, Juan Camilo Giraldo Mejia, Alejandro Restrepo Correa, Alicia Martínez Rebollar.
Redacción – borrador original: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa.
Redacción – revisión y edición: Fabio Alberto Vargas, Juan Camilo Giraldo Mejía, Alejandro Restrepo Correa.