ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

Máster Universitario en Ingeniería Industrial

Framework de Automatización FinOps: Mejorando la eficiencia de Costos en Despliegues de Cloud

Autor
Álvaro Ruiz Cabrera

Dirigido por
Luis Francisco Sánchez Marchante

Madrid
Junio 2024

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor **D. Álvaro Ruiz Cabrera DECLARA** ser el titular de los derechos de propiedad intelectual de la obra: **Framework de Automatización FinOps: Mejorando la eficiencia de Costos en Despliegues de Cloud**, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, los derechos de digitalización, de archivo, de reproducción, de distribución y de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

(a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.

(d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.

(e) Asignar por defecto a estos trabajos una licencia Creative Commons.

(f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

(a) Que la Universidad identifique claramente su nombre como autor de la misma

(b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

(c) Solicitar la retirada de la obra del repositorio por causa justificada.

(d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

(a) El autor se compromete a:

(b) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

(c) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

(d) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

(e) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplica-

ble, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 12 de Julio de 2024

ACEPTA

Fdo.: . . . . . .

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Framework de Automatización FinOps: Mejorando la eficiencia de Costos en Despliegues de Cloud

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2023/2024 es de mi autoría, original e inédito y

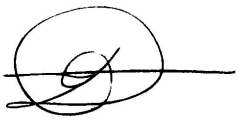
no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Álvaro Ruiz Cabrera Fecha: 19 / 07 / 2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Luis Francisco
Sánchez Merchante
2024.07.19 11:08:40
+02'00'

Fdo.: Luis Francisco Sánchez Merchante Fecha: 19 / 07 / 2024

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

Máster Universitario en Ingeniería Industrial

Framework de Automatización FinOps: Mejorando la eficiencia de Costos en Despliegues de Cloud

Autor
Álvaro Ruiz Cabrera

Dirigido por
Luis Francisco Sánchez Merchante

Madrid
Junio 2024

Resumen

Este Trabajo Fin de Máster (TFM) presenta el desarrollo de un framework de automatización FinOps para optimizar la eficiencia de costos en despliegues de infraestructura cloud. La solución propuesta aborda la creciente necesidad de las organizaciones de gestionar de manera eficiente y rentable sus recursos en la nube, alineando el gasto tecnológico con los objetivos de negocio.

El framework desarrollado se basa en la integración de varias tecnologías clave:

- Terraform para la definición y despliegue automatizado de infraestructura como código (IaC).
- Azure para el aprovisionamiento de recursos cloud, incluyendo Virtual Machine Scale Sets, redes virtuales y servicios de monitoreo.
- GitHub Actions para la implementación de flujos de trabajo de integración y despliegue continuo (CI/CD).

La solución permite el escalado dinámico de recursos basado en métricas internas y consideraciones financieras, utilizando Azure Monitor para la recolección de datos y Azure Functions para la ejecución de acciones automatizadas.

El framework desarrollado permite la creación de un modelo de optimización que considera variables técnicas como el uso de CPU y memoria, así como parámetros financieros definidos por la organización. Esto favorece el ajuste de infraestructura de forma proactiva, maximizando la eficiencia operativa y minimizando costos innecesarios.

Este trabajo contribuye al campo emergente de FinOps, ofreciendo una solución práctica y escalable para organizaciones que buscan optimizar sus operaciones en la nube. El framework desarrollado sienta las bases para futuras investigaciones en la automatización de la gestión financiera de infraestructuras cloud.

Abstract

This Master's Thesis (TFM) presents the development of a FinOps automation framework to optimize cost efficiency in cloud infrastructure deployments. The proposed solution addresses the growing need for organizations to efficiently and cost-effectively manage their cloud resources, aligning technological spending with business objectives.

The developed framework is based on the integration of several key technologies:

- Terraform for the definition and automated deployment of infrastructure as code (IaC).
- Azure for the provisioning of cloud resources, including Virtual Machine Scale Sets, virtual networks, and monitoring services.
- GitHub Actions for implementing continuous integration and continuous deployment (CI/CD) workflows.

The solution enables the dynamic scaling of resources based on internal metrics and financial considerations, using Azure Monitor for data collection and Azure Functions for executing automated actions.

The developed framework allows for the creation of an optimization model that considers technical variables such as CPU and memory usage, as well as financial parameters defined by the organization. This facilitates proactive infrastructure adjustments, maximizing operational efficiency and minimizing unnecessary costs.

This work contributes to the emerging field of FinOps, offering a practical and scalable solution for organizations seeking to optimize their cloud operations. The developed framework lays the groundwork for future research in the automation of financial management for cloud infrastructures.

Agradecimientos

Miércoles, doce de la noche, cansado de todo un día de redacción de este TFM, fuente para mi de agobios y alegrías. Aún quedan partes por retocar, matices que pulir, algún que otro desarrollo y alguna que otra hora por delante; pero, creo necesario pararme a agradecer a todos los que han hecho posible que, si todo sale bien, en poco más de una semana pueda dar por concluido este proyecto.

Quiero agradecer a los hicieron posible empezar a dar luz a este trabajo, a Adrián, Paco y Sergio. Gracias por estar ahí y por prestarme vuestra ayuda en los comienzos del TFM.

Quiero agradecer a mi familia y amigos. A mis padres y mis hermanas, por su constante preocupación por que todo saliera bien; por sus incesantes recordatorios que, aunque añadían más presión, me han ayudado a mantenerme activo y han dado un motivo por el que esforzarse. A Juan, Marina, Pablo, Antonio, Andrés, José, Lío, María, Juan, Enrique, Roberto... (paro porque, si no, no voy a terminar esto nunca) y la infinidad de personas que han tenido que escuchar mis lloros, que me han mostrado su apoyo y ofrecido su ayuda en el transcurso del proyecto. Habéis sido para mi luz en la ocuridad.

Quiero agradecer a Luis, tutor de este TFM, persona clave para la realización de este trabajo. Gracias por ayudar a idearlo, gracias por las constantes reuniones, gracias por tu disponibilidad, gracias por seguir ayudándome aun estando de baja, gracias por tu orientación y tu apoyo. Gracias por todo lo que has hecho, no hubiera sido posible sin ti.

Gracias a todas las personas que tendría que agradecer y no lo he hecho.

Pero sobre todo gracias a Dios por haber puesto a todas estas personas en mi camino y porque por fin pueda entregar ya el proyecto.

Gracias, gracias y gracias.

Índice

Listado de figuras

Listado de tablas

Lista de listings

Acrónimos

ICAI	Insitituto Católico de Artes e Industrias
TFM	Trabajo Fin de Máster
ETS	Escuela Técnica Superior
FinOps	Operaciones Financieras (Financial Operations)
IaC	Infraestructura como Código (Infrastructure as Code)
CI/CD	Integración Continua/Despliegue Continuo (Continuous Integration/Continuous Deployment)
VM	Máquina Virtual (Virtual Machine)
NSG	Grupo de Seguridad de Red (Network Security Group)
CIDR	Enrutamiento entre dominios sin clase (Classless Inter-Domain Routing)
IP	Protocolo de Internet (Internet Protocol)
SKU	Unidad de Mantenimiento de Existencias (Stock Keeping Unit)
LRS	Almacenamiento con Redundancia Local (Locally Redundant Storage)
ZRS	Almacenamiento con Redundancia de Zona (Zone Redundant Storage)
GRS	Almacenamiento con Redundancia Geográfica (Geo-Redundant Storage)
RA-GRS	Almacenamiento con Redundancia Geográfica con Acceso de Lectura (Read-Access Geo-Redundant Storage)
GZRS	Almacenamiento con Redundancia de Zona Geográfica (Geo-Zone-Redundant Storage)
RA-GZRS	Almacenamiento con Redundancia de Zona Geográfica con Acceso de Lectura (Read-Access Geo-Zone-Redundant Storage)
NAT	Traducción de Direcciones de Red (Network Address Translation)
HTTP	Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol)
HTTPS	Protocolo Seguro de Transferencia de Hipertexto (Hypertext Transfer Protocol Secure)
SSH	Secure Shell
DNS	Sistema de Nombres de Dominio (Domain Name System)
VPN	Red Privada Virtual (Virtual Private Network)
TI	Tecnologías de la Información
MVP	Producto Mínimo Viable (Minimum Viable Product)
ODS	Objetivos de Desarrollo Sostenible

DSL	Lenguaje Específico de Dominio (Domain-Specific Language)
AWS	Amazon Web Services
GCP	Google Cloud Platform
XaaS	Todo como Servicio (Anything as a Service)
PaaS	Plataforma como Servicio (Platform as a Service)
SaaS	Software como Servicio (Software as a Service)
IaaS	Infraestructura como Servicio (Infrastructure as a Service)
SMB	Bloque de Mensajes del Servidor (Server Message Block)
NFS	Sistema de Archivos de Red (Network File System)
E/S	Entrada/Salida
VCS	Sistema de Control de Versiones (Version Control System)
DVCS	Sistema de Control de Versiones Distribuido (Distributed Version Control System)
KPI	Indicador Clave de Desempeño (Key Performance Indicator)
JSON	Notación de Objetos de JavaScript (JavaScript Object Notation)
TIR	Tasa Interna de Retorno
VPN	Valor Presente Neto
CPU	Central Processing Unit (Unidad central de procesamiento)
VMSS	Virtual Machine Scale Set (Conjunto de escalado de máquinas virtuales)

Capítulo 1

Introducción

En el contexto de la transformación digital y la creciente adopción de soluciones en la nube, la gestión eficiente de recursos se ha convertido en un desafío crítico para las organizaciones. La escalabilidad y el desempeño óptimo de los sistemas son fundamentales para garantizar la continuidad del negocio y la satisfacción del cliente.

El gasto en inversión de las empresas españolas en la nube ('cloud') ha crecido un 39% en 2023, según los expertos de IPM, empresa de soluciones y servicios transversales de tecnologías de la información (TI). El 'Cloud' se ha convertido en una de las estrategias más demandadas dentro de la inversión en tecnología, ya que tan solo un 6% de las organizaciones españolas no tiene presencia en la nube. Según los últimos datos, el 40% de los ingresos de las empresas serán generados gracias a su digitalización [5].

En este contexto, ha surgido un nuevo concepto llamado FinOps, que consiste en la práctica de introducir un cambio cultural de responsabilidad financiera en el modelo de gasto variable de la nube, lo que permite a los equipos empresariales y de ingeniería distribuidos hacer concesiones entre velocidad, coste y calidad en sus decisiones de inversión y arquitectura de la nube [6]. Dentro de esta práctica, tiene vital importancia la gestión y automatización de la carga de trabajo de recursos. Su objetivo es ofrecer a los equipos de FinOps la posibilidad de ajustar la oferta a la demanda de la forma más eficiente, y optimizar eficazmente el uso de la nube mediante la medición de la demanda de cargas de trabajo y el aprovisionamiento de capacidad de forma dinámica [7].

Empresas de todo el mundo han visto en la digitalización y en la automatización de procesos la única fórmula para poder sobrevivir en un contexto cambiante y de una competencia feroz. En este sentido, el 28% de los responsables de TI de

España considera que su empresa perderá dinero si no adopta la automatización total en el futuro [8].

La necesidad de este proyecto surge de la complejidad creciente en la gestión de infraestructuras en la nube. Automatizar este proceso permite una asignación más precisa y dinámica de recursos, adaptándose en tiempo real a las necesidades del sistema y optimizando tanto el rendimiento como los costos asociados.

En este contexto, se plantea una solución integral que no solo considera el consumo directo de recursos, como la CPU, memoria y espacio de almacenamiento, sino que también incorpora datos contextualizados, como la pertenencia a departamentos o áreas específicas. Esto permite una asignación más estratégica y una optimización basada en el rendimiento y los beneficios obtenidos.

A través de un enfoque estratégico y dinámico, este proyecto busca contribuir al desarrollo de soluciones innovadoras en el ámbito de la gestión de recursos en la nube, con el fin de mejorar la eficiencia operativa y económica de las organizaciones.

1.1 Motivación del proyecto

La realización del presente proyecto se fundamenta en la necesidad de abordar los desafíos asociados con la gestión eficiente de recursos en entornos cloud. Dicha motivación surge de diversas consideraciones y tendencias observadas en el ámbito tecnológico y empresarial, las cuales se detallan a continuación:

1. Complejidad de la Gestión en la Nube: Con el crecimiento exponencial de la adopción de servicios en la nube, las organizaciones se enfrentan a desafíos cada vez mayores en términos de gestión y optimización de recursos. La diversidad de servicios y la necesidad de mantener costos controlados hacen que la gestión de recursos en la nube sea una tarea compleja y crítica para el éxito operativo y financiero de las organizaciones.

2. Necesidad de Eficiencia Operativa y Económica: En un entorno empresarial altamente competitivo, la eficiencia operativa y económica es esencial para mantenerse relevante y competitivo en el mercado. La capacidad de adaptarse dinámicamente a las demandas fluctuantes de recursos en entornos cloud es clave para optimizar costos, mejorar la agilidad operativa y garantizar la disponibilidad de recursos cuando sea necesario.

3. Impacto de la Tecnología en la Estrategia Empresarial: La tecnología de la nube ha transformado fundamentalmente la forma en que las organizaciones operan y ofrecen servicios a sus clientes. La capacidad de gestionar eficientemente recursos en la nube no solo tiene implicaciones operativas, sino que

también puede tener un impacto significativo en la estrategia empresarial, permitiendo la innovación, la escalabilidad y la capacidad de respuesta a las demandas del mercado.

4. Enfoque en Resultados y Rentabilidad: En un entorno empresarial cada vez más orientado hacia los resultados y la rentabilidad, es crucial alinear la asignación de recursos en la nube con los objetivos financieros y empresariales de la organización. La capacidad de medir y optimizar el rendimiento y los beneficios obtenidos a partir de la asignación de recursos en la nube es esencial para garantizar un retorno de la inversión óptimo y maximizar el valor para la organización.

1.2 Objetivos

El presente proyecto tiene como objetivo principal diseñar e implementar un framework destinado a automatizar y optimizar el escalado y desescalado de recursos en entornos cloud. Para lograr este objetivo general, se desglosan los siguientes objetivos específicos:

1. Desarrollar un framework robusto y flexible: El primer objetivo es desarrollar un framework que sea capaz de gestionar de manera eficiente el escalado y desescalado de recursos en entornos cloud. Esto implicará la definición de una arquitectura sólida y modular, que permita adaptarse a diferentes entornos y requisitos de los usuarios.

2. Integrar fuentes de datos relevantes: Se buscará integrar fuentes de datos relevantes para la toma de decisiones automatizada en el ajuste dinámico de recursos. Esto incluirá datos sobre el consumo de recursos (CPU, memoria, almacenamiento), así como datos contextualizados, como la pertenencia a departamentos o áreas específicas, que permitan una asignación más estratégica de los recursos.

3. Establecer métricas de evaluación y seguimiento: Se definirán métricas de evaluación y seguimiento para medir el rendimiento y los beneficios obtenidos a partir de la asignación de recursos. Esto permitirá evaluar la eficacia del framework y realizar ajustes o mejoras según sea necesario para maximizar el valor para la organización.

4. Validar el framework en entornos de prueba: Finalmente, se llevará a cabo la validación del framework en entornos de prueba, utilizando datos simulados o entornos de desarrollo controlados. Esto permitirá identificar posibles problemas o limitaciones y realizar ajustes antes de su implementación en

entornos de producción.

1.3 Metodología de trabajo y planificación

La metodología de trabajo para el desarrollo de este proyecto se fundamentará en un enfoque sistemático y estructurado que permita alcanzar los objetivos establecidos de manera eficiente y efectiva. En la Tabla 1.1 se presenta la distribución de tareas que se ha seguido para la elaboración de este proyecto. A continuación, se describe la metodología propuesta:

- Estudio y pruebas de despliegue de infraestructura virtual en cloud: En primer lugar, se estudiarán distintos métodos para desplegar infraestructura virtual en entornos cloud, por ejemplo, en Azure. Se usarán distintas tecnologías con el fin de desplegar infraestructura, modificarla y eliminarla, además de distintos métodos posibles para añadir reglas al escalado de estos recursos.
- Estudio de las distintas variables a incluir en el framework: Después de la primera toma de contacto con los distintos métodos para automatizar la infraestructura, se procederá a investigar las distintas variables que será necesario incluir para realizar el escalado dinámico. Se estudiarán distintos KPIs para poder realizar una optimización acorde a FinOps.
- Diseño del Framework: Con base en los requisitos identificados, se procederá al diseño del framework para automatizar el escalado y desescalado de recursos en entornos cloud. Esto implicará la definición de la arquitectura del sistema, la identificación de componentes clave y la elaboración de diagramas de flujo y modelos conceptuales que describan el funcionamiento del framework.
- Integración de variables y reglas de automatización en el framework: Se integrarán las distintas variables identificadas anteriormente, en el framework. Asimismo, se implementarán algoritmos para tomar decisiones automatizadas sobre el escalado y desescalado de recursos, basadas en las métricas de rendimiento y los objetivos de optimización definidos.
- Validación del Framework: Finalmente, se llevará a cabo la validación del framework en entornos de prueba, utilizando datos simulados o entornos de desarrollo controlados. Se realizarán pruebas exhaustivas para identificar posibles problemas o limitaciones y realizar ajustes antes de su implementación en entornos de producción.
- Redacción de la memoria: La redacción de la memoria del proyecto se re-

alizará a lo largo de toda la duración de este, de forma paralela a la realización de los apartados anteriores.

	Marzo	Abril	Mayo	Junio
Estudio y pruebas de despliegue de infraestructura virtual en Cloud				
Estudio de las distintas variables a incluir en el framework				
Diseño del Framework				
Integración de variables y reglas de automatización en el framework				
Validación del framework				
Redacción de la memoria				

Table 1.1: Cronograma del proyecto

Es importante destacar que el cronograma propuesto puede estar sujeto a modificaciones durante el desarrollo del proyecto, dependiendo de los avances y los resultados obtenidos en cada etapa.

1.4 Recursos a emplear

Recursos a emplear:

- Plataformas de Cloud Computing: Se utilizarán plataformas de cloud computing, como Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform (GCP), para desarrollar, probar y desplegar el framework. Estas plataformas proporcionarán los recursos de infraestructura necesarios, como servidores virtuales, almacenamiento y servicios de bases de datos.
- Herramientas de Desarrollo: Se emplearán herramientas de desarrollo de software, sistemas de control de versiones (como Git), y frameworks de desarrollo de aplicaciones, para facilitar el desarrollo del proyecto.

Capítulo 2

Estado de la cuestión

El cloud computing es una evolución de la tecnología de la información y un modelo de negocio dominante para ofrecer recursos de TI. Con la computación en la nube, tanto individuos como organizaciones pueden obtener acceso a la red bajo demanda a un conjunto compartido de recursos de TI gestionados y escalables, como servidores, almacenamiento y aplicaciones. Dependemos en gran medida de los servicios en la nube en nuestra vida diaria, por ejemplo, para almacenar datos, escribir documentos, gestionar negocios y jugar juegos en línea. La computación en la nube también proporciona la infraestructura que ha impulsado tendencias digitales clave como el Internet de las cosas, big data e inteligencia artificial, acelerando así la dinámica de la industria, alterando los modelos de negocio existentes y alimentando la transformación digital [9].

En un entorno en el que las tecnologías cloud están obtenido cada vez una mayor relevancia, se ha hecho imprescindible optimizar al máximo el empleo de este tipo de recursos para poder subsistir en un mercado tremendamente competitivo. En su impulso por migrar a la nube, las organizaciones están tropezando con costosos obstáculos. Si bien a menudo hay una amplia gama de razones para estos contratiempos, muchos pueden atribuirse a capacidades inmaduras de gestión financiera en la nube, conocidas como FinOps. Como resultado, las organizaciones a menudo toman decisiones costosas sobre su consumo de nube. Esto es particularmente problemático en las condiciones macroeconómicas actuales, donde las organizaciones tienen aún menos margen para cometer errores. Si bien FinOps es más efectivo cuando las organizaciones lo implementan desde el principio, utilizarlo en casi cualquier momento durante el viaje de migración a la nube de una empresa proporciona beneficios significativos. Las organizaciones que utilizan FinOps de manera efectiva pueden reducir los costos de la nube hasta en un 20% a 30% [10]. Aunque es un término muy nuevo todavía, está adquiriendo cada vez una mayor

relevancia; y una gran cantidad de empresas están empezando a adoptarlo.

Existen numerosas herramientas que facilitan la adopción de FinOps a las empresas [11]. Estas herramientas en su mayoría tratan de dar visibilidad a los costes derivados de los servicios cloud.

- Apptio Cloudability, una plataforma multi-cloud que incluye detección de anomalías respaldada por inteligencia artificial, reglas de asignación personalizables, dashboards, presupuestos y previsiones, análisis de contenedores y recomendaciones de optimización [12].
- Flexera es una plataforma de gestión de TI basada en SaaS que respalda la visibilidad de TI, la gestión de activos de TI, FinOps y la optimización y migración al cloud [13].
- Nordcloud es una plataforma multi-cloud para asignar costos, identificar recursos no gestionados, mostrar datos consolidados en tiempo real sobre costos en la nube y proporcionar recomendaciones de optimización [14].

Dentro del marco de este proyecto, se pretende aportar a esta línea de trabajo gracias al empleo de técnicas que permitan automatizar el despliegue de infraestructura virtual en entornos cloud y el ajuste de sus recursos, teniendo en cuenta los principios de FinOps. Esto permitirá disponer de una configuración óptima de los recursos, logrando así no incurrir en costes innecesarios y maximizar el beneficio.

Dentro del ámbito de la automatización, se han realizado numerosos avances. Conceptos clave como la gestión de la configuración y la infraestructura como código permiten una entrega continua y automatizada de componentes de software durante todo el ciclo de vida, por ejemplo, para instalar, iniciar, modificar, detener o terminar componentes. Al describir los componentes y la infraestructura de una aplicación en modelos de implementación mantenibles y reutilizables, se puede establecer una automatización de implementación de extremo a extremo repetible. Dichos modelos de implementación pueden ser de naturaleza declarativa o imperativa: los modelos declarativos expresan el estado deseado en el que se transfieren una aplicación o partes de ella. Por otro lado, los modelos imperativos describen los pasos de implementación de manera secuencial y a través de procesos.

En la industria y la investigación, los modelos de implementación declarativos son ampliamente aceptados como el enfoque más adecuado para la implementación de aplicaciones y la gestión de la configuración [15]. Como resultado, se han desarrollado una gran cantidad de tecnologías diferentes siguiendo este enfoque, como Chef [16], Puppet [17], AWS CloudFormation [18], Jenkins[19], Terraform[20], Ansible[21] o Kubernetes[22].

Todas estas tecnologías tienen como objetivo automatizar la implementación de aplicaciones, pero difieren en las características y mecanismos admitidos. Por ejemplo, Terraform admite la implementación en múltiples proveedores de servicios en la nube y puede dirigirse a diferentes ofertas de servicios en la nube (XaaS). Mientras tanto, existen tecnologías específicas de proveedores de servicios en la nube, co mo AWS CloudFormation, que permiten la implementación solo en los servicios en la nube de Amazon. Además, existen tecnologías específicas de plataformas, como Kubernetes, que admiten solo paquetes de implementación específicos (imágenes de contenedores) u ofertas de servicios en la nube (por ejemplo, restringidas a PaaS). Además, la mayoría de las tecnologías utilizan su propio lenguaje de modelado con su propia sintaxis y expresividad.

A continuación, se muestra una breve descripción algunas de estas tecnologías:

- Puppet permite escribir definiciones de configuración reutilizables que describen los recursos del sistema y su estado para múltiples proveedores y servicios utilizando su propio lenguaje específico de dominio (DSL) [23].
- Chef utiliza un DSL basado en Ruby. Basado en una arquitectura cliente-servidor, se puede utilizar para mantener y configurar sistemas en varias plataformas o proveedores de servicios en la nube [16].
- Ansible utiliza un DSL declarativo basado en YAML para describir configuraciones del sistema para varias plataformas y servicios en la nube [21].
- Kubernetes es una plataforma para automatizar la orquestación de aplicaciones contenerizadas y multi-servicio. Implementa automáticamente la aplicación especificada en un clúster y garantiza que se alcance y mantenga su configuración deseada [22].
- OpenStack Heat es un motor de orquestación que permite la descripción de aplicaciones basadas en XaaS utilizando una sintaxis YAML. Gestiona todo el ciclo de vida de una implementación y proporciona interfaces para extensiones personalizadas [24].
- Terraform es un orquestador que proporciona interfaces de complementos para extensiones personalizadas. Utiliza su propio DSL y se dirige principalmente a implementaciones de aplicaciones multi-nube [20].
- AWS CloudFormation utiliza un DSL (JSON y YAML) para describir, implementar y gestionar todos los recursos de infraestructura en los servicios en la nube de Amazon [18].
- SaltStack es un sistema de orquestación y gestión de configuración que utiliza su propio DSL para implementar y gestionar todo tipo de pilas de aplica-

ciones dirigidas a diferentes proveedores y servicios en la nube, recientemente adquirido por vmware [25].

- Juju es una herramienta de orquestación de aplicaciones basada en topología. Permite el modelado de implementaciones de aplicaciones utilizando un DSL basado en YAML y admite múltiples ofertas y servicios en la nube [26].
- CFEngine es una herramienta de gestión de configuración de código abierto que proporciona funcionalidades empresariales mediante una versión comercial. Su función principal es proporcionar configuración y gestión automatizadas sobre los recursos informáticos existentes [27].
- Similar a AWS CloudFormation, Azure Resource Manager es el servicio de implementación y gestión de Azure para administrar recursos en el entorno de nube de Microsoft [28].
- Docker Compose es un marco para definir y ejecutar aplicaciones Docker multi-contenedor. Proporciona un lenguaje basado en YAML para especificar los contenedores que forman una aplicación y su configuración [29].
- Cloudify es un marco de orquestación de aplicaciones de código abierto. Admite implementaciones de nubes híbridas para todo tipo de servicios en la nube basados en una sintaxis YAML personalizada inspirada en el estándar TOSCA [30].

La computación en la nube ha revolucionado la forma en que las organizaciones gestionan y despliegan sus recursos tecnológicos. La capacidad de escalar dinámicamente los recursos según la demanda es una de las características más valiosas de los servicios en la nube. A continuación, se describen algunas funcionalidades que ofrecen los principales proveedores de servicios en la nube para el escalado de clusters de máquinas virtuales.

Microsoft Azure

- **Azure Virtual Machine Scale Sets (VMSS)[31]:** Permiten crear y gestionar un grupo de máquinas virtuales idénticas con balanceo de carga automático. Los VMSS pueden escalar automáticamente el número de instancias en función de la demanda o según una programación definida. Azure VMSS también integra balanceadores de carga para asegurar una distribución eficiente de los recursos.
- **Azure Kubernetes Service (AKS)[32]:** Proporciona un entorno gestionado para desplegar y gestionar aplicaciones en contenedores utilizando Kubernetes. AKS permite el escalado automático de pods y nodos en función

de las métricas de uso y demanda.

- **Azure Functions[33]:** Ofrece una plataforma de computación sin servidor que se escala automáticamente en respuesta a eventos. Azure Functions es ideal para tareas que se ejecutan en respuesta a eventos y no requieren una infraestructura subyacente constante.

Amazon Web Services (AWS)

- **Auto Scaling Groups[34]:** Permiten escalar automáticamente el número de instancias EC2 en función de las políticas definidas por el usuario. AWS Auto Scaling ajusta dinámicamente la capacidad para mantener un rendimiento estable y predecible al menor costo posible.
- **Elastic Kubernetes Service (EKS)[35]:** Proporciona un entorno gestionado para Kubernetes, permitiendo el escalado automático de pods y nodos en función de las métricas de uso.
- **AWS Lambda[36]:** Ofrece computación sin servidor que se escala automáticamente en respuesta a la cantidad de solicitudes entrantes, ideal para aplicaciones que requieren ejecución en respuesta a eventos.

Google Cloud Platform (GCP)

- **Google Kubernetes Engine (GKE)[37]:** Proporciona un entorno gestionado para desplegar, gestionar y escalar aplicaciones en contenedores utilizando Kubernetes. GKE permite el escalado automático de pods y nodos basado en métricas de uso.
- **Instance Groups[38]:** Permiten gestionar un grupo de instancias de máquinas virtuales idénticas con balanceo de carga y escalado automático en función de las políticas definidas.
- **Cloud Functions[39]:** Ofrece una plataforma de computación sin servidor que se escala automáticamente en respuesta a eventos, ideal para tareas que se ejecutan en respuesta a eventos.

Capítulo 3

Fundamentos teóricos

3.1 FinOps

FinOps es una práctica de gestión financiera de la nube que combina las capacidades de las finanzas, las operaciones y la ingeniería para optimizar el gasto en la nube y maximizar su valor empresarial. Esta disciplina emergente se ha convertido en una necesidad para las empresas que utilizan servicios en la nube, ya que les permite tomar decisiones informadas y basadas en datos sobre sus inversiones en la nube.

El término "FinOps" proviene de la combinación de "finanzas" y "operaciones", destacando la importancia de la colaboración entre los equipos financieros y técnicos. La práctica de FinOps no solo busca reducir costos, sino también mejorar la eficiencia y la efectividad del uso de los recursos en la nube, asegurando que todo lo gastado en la nube aporte al máximo al éxito del negocio.[40]

A medida que la adopción de la nube ha crecido exponencialmente, también lo ha hecho la necesidad de una disciplina que pueda abordar y optimizar de manera proactiva el gasto en la nube. FinOps ofrece un marco para esto, proporcionando estructuras y procesos que ayudan a las empresas a entender, gestionar y prever sus gastos en la nube. Esto incluye la implementación de políticas de uso eficiente, la negociación de tarifas y descuentos con proveedores de servicios en la nube, y la promoción de una cultura de responsabilidad del gasto en toda la organización.

En su núcleo, FinOps es una práctica cultural. Es la forma en que los equipos gestionan sus costos en la nube, donde todos asumen la responsabilidad de su uso de la nube con el apoyo de un grupo central de mejores prácticas. Equipos multifuncionales en ingeniería, finanzas, producto, adquisiciones, etc., trabajan

juntos para permitir una entrega de productos más rápida, al mismo tiempo que obtienen un mayor control financiero y previsibilidad.

FinOps es la práctica de aportar un cambio cultural de responsabilidad financiera al modelo de gasto variable de la nube, permitiendo a los equipos de ingeniería y negocio distribuidos hacer concesiones entre velocidad, costo y calidad en sus decisiones de arquitectura e inversión en la nube. FinOps consiste en eliminar los bloqueos; capacitar a los equipos de ingeniería para ofrecer mejores funciones, aplicaciones y migraciones con mayor rapidez; y permitir una conversación inter-funcional sobre dónde invertir y cuándo.

La FinOps Foundation [41] es la organización, perteneciente a la Linux Fundation [42], que se encarga de recopilar, estandarizar y compartir las mejores prácticas de gestión financiera en la nube, así como proporcionar formación y certificación a las profesionales cloud. FinOps ofrece un catálogo de buenas prácticas que permite un mayor control de gastos y una gestión más eficiente de los recursos.

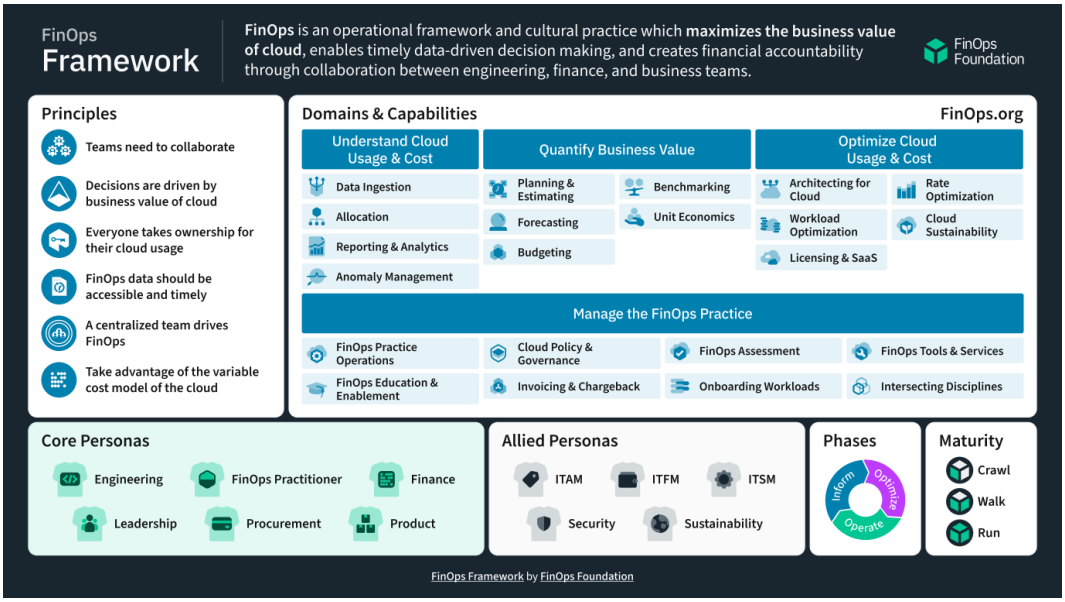


Figure 3.1: Marco de Trabajo de la FinOps Foundation. [1]

El objetivo está claro: conseguir que cada euro invertido en cloud empuje la rentabilidad de nuestro negocio. Contar con una buena práctica FinOps supone una ventaja competitiva, ya que "cuando las actividades en el cloud están gestionadas de forma centralizada podemos alcanzar el 38% de ahorros en los costes en la nube."[43]

3.1.1 Principios fundamentales de FinOps

Los principios fundamentales de FinOps son los pilares que guían a las organizaciones en la gestión y optimización de sus costos en la nube. Estos principios promueven la colaboración interdepartamental, la responsabilidad distribuida y el uso eficiente de los recursos en la nube. A continuación, se detallan estos principios clave [44]:

- Los equipos necesitan colaborar.
  - Los equipos de finanzas, tecnología, producto y negocio trabajan juntos en tiempo casi real, ya que la nube opera por recurso y por segundo.
  - Los equipos trabajan juntos para mejorar continuamente la eficiencia y la innovación.
- Las decisiones se basan en el valor comercial de la nube
  - Las métricas económicas unitarias y basadas en el valor demuestran el impacto comercial mejor que el gasto agregado.
  - Tomar decisiones conscientes de la compensación entre costo, calidad y velocidad.
  - Considerar la nube como un motor de innovación.
- Todos asumen la responsabilidad de su uso de la nube
  - La responsabilidad del uso y costo se descentraliza al máximo, con los ingenieros asumiendo la responsabilidad de los costos desde el diseño de la arquitectura hasta las operaciones.
  - Los equipos están capacitados para gestionar su propio uso de la nube dentro de su presupuesto.
  - Descentralizar la toma de decisiones en torno a la rentabilidad de la arquitectura, el uso de recursos y la optimización.
  - Los equipos técnicos deben comenzar a considerar el costo como una nueva métrica de eficiencia desde el inicio del ciclo de vida del desarrollo de software.
- Los datos de FinOps deben ser accesibles y llegar a tiempo
  - Procesar y compartir los datos de costos en cuanto estén disponibles.
  - La visibilidad en tiempo real impulsa de manera autónoma una mejor utilización de la nube.

  - Los ciclos de feedback rápido resultan en un comportamiento más eficiente.
  - Se proporciona visibilidad constante del gasto en la nube a todos los niveles de la organización.
  - Crear, monitorear y mejorar la previsión y planificación financiera en tiempo real.
  - El análisis de tendencias y variaciones ayuda a explicar el aumento de costos.
  - La evaluación comparativa interna de equipos impulsa las mejores prácticas y celebra los logros.
  - La evaluación comparativa a nivel de pares de la industria evalúa el desempeño de la empresa.
- Un equipo centralizado impulsa FinOps
  - El equipo central fomenta, promueve y facilita las mejores prácticas en un modelo de responsabilidad compartida, de manera similar a la seguridad, que tiene un equipo central pero en el que todos siguen siendo responsables de su parte.
  - Se requiere el apoyo ejecutivo para FinOps y sus prácticas y procesos.
  - La optimización de tarifas, compromisos y descuentos se centraliza para aprovechar las economías de escala.
  - Eliminar la necesidad de que los ingenieros y los equipos de operaciones piensen en las negociaciones de tarifas, permitiéndoles centrarse en la optimización del uso de sus propios entornos.
- Aprovechar el modelo de costo variable de la nube
  - El modelo de costo variable de la nube debe verse como una oportunidad para ofrecer más valor, no como un riesgo.
  - Adoptar la predicción, planificación y adquisición de capacidad justo a tiempo.
  - Preferir la planificación ágil e iterativa sobre los planes estáticos a largo plazo.
  - Adoptar un diseño de sistemas proactivo con ajustes continuos en la optimización de la nube en lugar de limpiezas reactivas infrecuentes.

3.1.2 Personas involucradas en FinOps

La implementación de FinOps requiere que muchas partes interesadas de una organización trabajen en colaboración con el equipo de FinOps. No sólo el equipo de FinOps realiza actividades de FinOps; sino que existe un gran número de personas involucradas en el uso, seguimiento, gestión o dirección del uso de la nube se beneficiarán del trabajo conjunto utilizando el Marco FinOps como modelo operativo.[45]

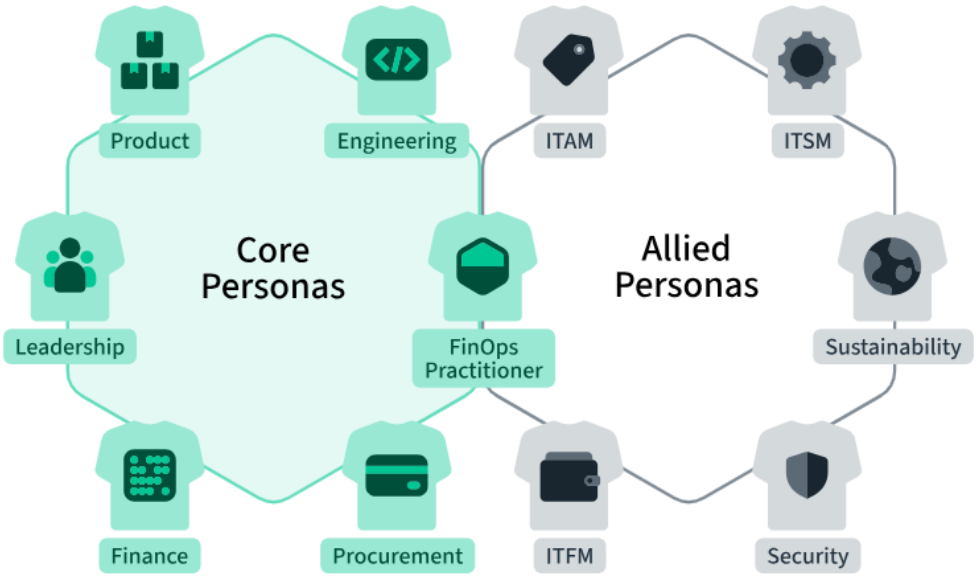


Figure 3.2: Personas clave en FinOps.

3.1.2.1 Personas Clave

Dentro de la mayoría de las organizaciones, hay personas clave que siempre estarán involucradas en la práctica de FinOps. Estas personas clave proporcionan todas las disciplinas organizativas para el uso eficaz de la nube. Cada grupo tiene un papel que desempeñar en la organización, así como en la práctica de FinOps, y cada uno tiene perspectivas, retos, métricas y resultados diferentes.[45]

- **FinOps Practitioner:**

  Sirve de puente entre los equipos empresariales, de ingeniería y financieros mediante la aplicación de un conocimiento práctico del marco de FinOps con el fin de establecer una cultura de FinOps y permitir la toma de decisiones basadas en pruebas para maximizar el valor empresarial de la nube.

Las responsabilidades de FinOps incluyen:

  - Competencia técnica
  - Habilidades analíticas
  - Gestión y optimización de costes
  - Colaboración y comunicación
  - Práctica de FinOps y Mejora continua
  - Resolución de problemas
  - Gestión del cambio
  - Colaboración con las personas involucradas en FinOps

- **Liderazgo:**

  Fomenta la alineación organizativa para priorizar las iniciativas de FinOps, permite actuar para superar los retos de FinOps y conecta las decisiones de tecnología de nube con los objetivos empresariales.

  Las responsabilidades de FinOps incluyen:

  - Supervisión de la planificación estratégica
  - Autoridad de toma de decisiones y priorización
  - Compromiso y alineación de las partes interesadas
  - Crecimiento de los ingresos
  - Cumplimiento y gobernanza

- **Producto:**

  Impulsar el valor empresarial definiendo requisitos y priorizando iniciativas que alineen FinOps con los objetivos empresariales, implicando a las partes interesadas y articulando la propuesta de valor de las decisiones tecnológicas en la nube.

  Las responsabilidades de FinOps incluyen:

  - Alineación estratégica
  - Compromiso de las partes interesadas
  - Definición de requisitos
  - Entrega de valor

  - Gestión del cambio

- **Ingeniería:**

  Responsable de diseñar, gestionar y optimizar la infraestructura de la nube para lograr rentabilidad, rendimiento y fiabilidad, garantizando al mismo tiempo la seguridad y el cumplimiento de los entornos de nube.

  Las responsabilidades de FinOps incluyen:

  - Gestión de la infraestructura de la nube
  - Despliegue de aplicaciones y servicios
  - Gestión de costes y optimización de recursos
  - Supervisión y alerta
  - Seguridad y conformidad
  - Automatización y herramientas
  - Arquitectura sostenible para la nube

- **Finanzas:**

  Proporciona conocimientos financieros y trabaja en estrecha colaboración con los profesionales de FinOps para conciliar las facturas de los proveedores de la nube con los datos de facturación de la nube para prever, presupuestar y facturar con precisión los costes de la nube.

  Las responsabilidades de FinOps incluyen:

  - Experiencia financiera
  - Definición de presupuestos y previsión
  - Análisis de asignación de costes
  - Informes financieros
  - Cumplimiento y gobernanza

- **Adquisiciones:**

  Responsable de la adquisición de servicios en la nube, la optimización de las relaciones con los proveedores y la garantía de compromisos rentables y conformes con los proveedores, colaborando con los profesionales de FinOps para garantizar que las tarifas contribuyan a la gestión satisfactoria de las finanzas en la nube.

Las responsabilidades de FinOps incluyen

- Gestión de riesgos de proveedores
- Gestión y negociación de contratos con proveedores
- Gestión y negociación de descuentos
- Colaboración con TI y Operaciones
- Supervisión y evaluación de licencias
- Cumplimiento y gobernanza

3.1.2.2 Personas aliadas

Las organizaciones pueden tener roles que no están directamente involucrados en la práctica de FinOps. Estas Personas Aliadas trabajan dentro de disciplinas tradicionales o emergentes - incluyendo Sostenibilidad, ITAM, ITFM/TBM, Seguridad e ITSM/ITIL - y pueden necesitar coordinarse con los Profesionales de FinOps. Sus funciones se alinean con la Capacidad Marco "Intersección de Disciplinas", que describe dónde se cruzan sus actividades con las FinOps.[45]

- **ITSM / ITIL**

  La gestión de servicios de TI (IT Service Management) es responsable de colaborar con los profesionales de FinOps para estandarizar y agilizar las operaciones de servicios de TI, mejorar la calidad y fiabilidad de los servicios y garantizar que los servicios de TI cumplen los niveles de servicio acordados y los objetivos de rendimiento se equilibran con las prioridades de gestión de costes de la nube.

  Las responsabilidades que se cruzan con FinOps incluyen:

  - Diseño de servicios
  - Operación y mejora del servicio
  - Supervisión y gestión del nivel de servicio
  - Gestión del cambio
  - Análisis y optimización de costes
  - Documentación e informes
  - Colaboración con las partes interesadas

- **ITAM**

  La gestión de activos de TI (IT Asset Management) colabora con los profesionales de FinOps para lograr eficiencia, transparencia y valor en la gestión de los activos de TI que afectan al uso de la nube, aprovechando la experiencia y los datos de ambas disciplinas para optimizar los costes, garantizar el cumplimiento y respaldar los objetivos empresariales estratégicos en equilibrio con las prioridades de gestión de costes de la nube.

  Las responsabilidades que se cruzan con FinOps incluyen:

  - Descubrimiento e inventario de activos
  - Auditoría y cumplimiento de activos
  - Gestión de licencias
  - Análisis y optimización de costes
  - Documentación e informes
  - Colaboración con las partes interesadas

- **Sostenibilidad**

  El equipo de sostenibilidad colabora con los profesionales de FinOps para garantizar que el uso de la nube optimiza el impacto medioambiental, impulsa la responsabilidad y acelera el progreso hacia objetivos de sostenibilidad más amplios de forma equilibrada con las prioridades de gestión de costes de la nube.

  Las responsabilidades que se cruzan con FinOps incluyen:

  - Optimización para iniciativas sostenibles
  - Reducción de residuos
  - Política y cumplimiento
  - Análisis de eficiencia y optimización
  - Documentación e informes
  - Colaboración con las partes interesadas

- **Seguridad**

  El equipo de seguridad de IT colabora con los profesionales de FinOps para aprovechar su experiencia y conocimientos en FinOps con el fin de optimizar el gasto en seguridad en la nube, mejorar la gobernanza financiera de IT

Security y reforzar la postura general de seguridad en la nube de la organización.

Las responsabilidades que se cruzan con FinOps incluyen:

- Supervisión y respuesta a anomalías
- Investigación y análisis de anomalías
- Política y cumplimiento
- Gestión de identidades y accesos
- Documentación e informes
- Colaboración con las partes interesadas

- **ITFM / TBM**

  El equipo de gestión financiera de TI es responsable de colaborar con los profesionales de FinOps para proporcionar transparencia en el gasto de TI, permitir una toma de decisiones informada, garantizar que las inversiones en TI tradicionales y en la nube estén alineadas con las prioridades empresariales, sean rentables y ofrezcan un valor cuantificable a la organización.

  Las responsabilidades que se cruzan con FinOps incluyen:

  - Presupuestos
  - Contabilidad y optimización de costes
  - Análisis financiero
  - Priorización de inversiones
  - Informes financieros
  - Mejoras continuas de los procesos
  - Documentación e informes
  - Colaboración con las partes interesadas

3.1.3 Áreas de acción FinOps

Los áreas de acción del marco de trabajo de la FinOps Foundation describen los resultados empresariales fundamentales que las organizaciones deben obtener de la práctica de FinOps. En otras palabras, la práctica de FinOps permitirá a las

organizaciones comprender su uso y coste de la nube, cuantificar su valor empresarial, optimizar tanto el uso como las tarifas pagadas y gestionar una práctica eficaz.[46]

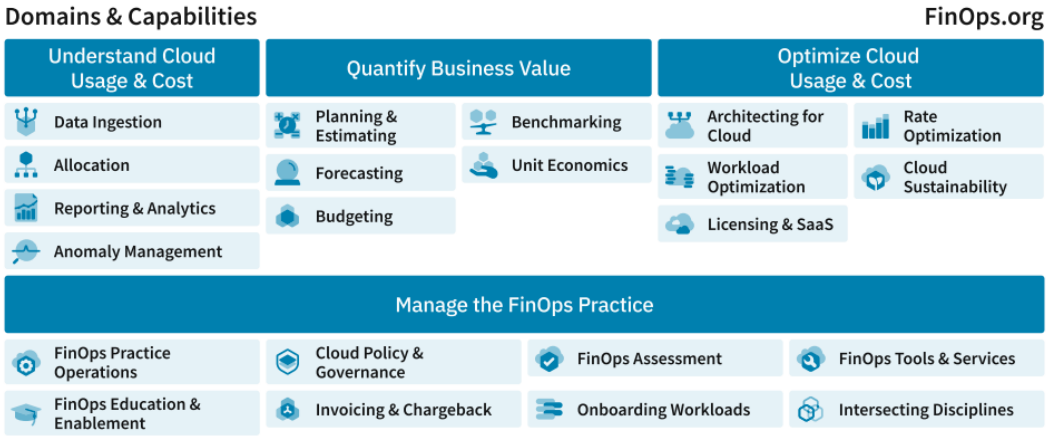


Figure 3.3: Áreas de trabajo de FinOps. [1]

Los dominios en los que la FinOps Foundation propone trabajar son los siguientes:

- **Comprender el uso y el coste de la nube**

  El resultado de este área es una mejor comprensión del uso de la nube por parte de una organización. En este ámbito, las organizaciones trabajan para recopilar toda la información necesaria para llevar a cabo la práctica de FinOps. Esto incluye el coste directo e imputado de la nube, el uso de la nube, la observabilidad, la utilización y los datos de sostenibilidad, así como otros conjuntos de datos requeridos por cualquier dominio de FinOps. Las actividades en este área también definen los metadatos organizativos para categorizar, asignar y resumir el coste y el uso de la nube, y definen los procesos de elaboración de informes y análisis que hacen que los datos estén disponibles para su uso por parte de todas las personas involucradas en FinOps.

  Dentro de este área, se proponen las siguientes competencias:

  - Ingesta de datos
  - Atribución de costos
  - Informes y análisis
  - Gestión de anomalías

- **Comprender el uso y el coste de la nube**

  Las organizaciones desarrollan este área para conectar los datos de uso y coste con el valor de negocio que crea, ayudando a asegurar que el valor es transparente y está dentro de las expectativas. Dentro de este área, las organizaciones asignan costes monetarios y no monetarios de la nube a presupuestos, utilizan información histórica y planes futuros para prever, establecer y medir KPI técnicos y organizativos, y realizar evaluaciones comparativas entre equipos, unidades de negocio y con otras organizaciones.

  Dentro de este área, se proponen las siguientes competencias:

  - Planificación y estimación
  - Previsión
  - Presupuestos
  - Evaluación comparativa
  - Economía unitaria

- **Optimizar el uso y coste de la nube**

  Este área se centra en la eficiencia de la nube, garantizando que las organizaciones sólo utilicen los recursos cuando proporcionen valor a la organización; y que los recursos utilizados se adquieran al menor coste e impacto aceptables para cumplir los objetivos de la organización. Las organizaciones medirán la eficiencia de varias formas, incluyendo el coste monetario, el uso de carbono o medidas más tradicionales de eficiencia operativa de TI. Las capacidades en este área permiten a la organización gestionar los tipos, tiempos y cantidades de recursos de nube utilizados, y las tarifas que se pagan por esos recursos. Las capacidades aquí también abordan la modernización de la arquitectura, las consideraciones de sostenibilidad para los equipos de FinOps y el uso de productos SaaS con licencia y basados en el consumo.

  Dentro de este área, se proponen las siguientes competencias:

  - Arquitectura para la nube
  - Optimización de tarifas
  - Optimización de la carga de trabajo
  - Sostenibilidad de la nube
  - Licencias y SaaS

- **Gestionar la práctica de FinOps**

  Este área permite la mejora continua para cambiar y alinear a toda la organización (sus personas, procesos y tecnología) para adoptar FinOps y utilizar la nube de forma que cree valor para la empresa. Las acciones aquí se centran en el funcionamiento eficaz de FinOps, la habilitación de toda la organización, la mejora de la interacción con todas las demás personas y funciones empresariales para apoyar y representar el uso de la nube de forma más eficaz.

  Dentro de este área, se proponen las siguientes competencias:

  - Operaciones de la Práctica FinOps
  - Política y gobernanza de la nube
  - Evaluación de FinOps
  - Herramientas y servicios de FinOps
  - Formación y capacitación en FinOps
  - Facturación y Contracargo
  - Incorporación de cargas de trabajo
  - Interrelacionar disciplinas

  Cada área describe un conjunto de competencias que una organización puede llevar a cabo para lograr estos resultados. Las organizaciones deben desarrollar las competencias que les aporten valor, en función de sus necesidades y del nivel de madurez actual de FinOps de su organización. Algunas competencias no son apropiadas para algunas organizaciones. Aunque una organización no invierta en todas las competencias; cada organización que utilice la nube y adopte FinOps realizará actividades en cada uno de las áreas definidas.

3.1.4 Fases de FinOps

FinOps se lleva a cabo trabajando de forma iterativa en las competencias definidas anteriormente a través de tres fases: Informar, Optimizar y Operar.[2]

Los equipos de una organización pueden estar trabajando en diferentes fases de FinOps en cualquier momento, y los profesionales de FinOps deberían estar siempre analizando el uso de la nube por parte de la organización, identificando formas de mejorarlo y creando documentación para capacitar a las personas responsables de promulgar los cambios que generarán más valor.

Figure 3.4: Fases de FinOps. [2]

Actuar con rapidez y regularidad puede ayudar a los equipos a evitar la parálisis por análisis y puede reforzar la buena práctica de empezar poco a poco e ir aumentando el tamaño y el alcance de las acciones del equipo a medida que madura gracias a la experiencia y al impulso.

El objetivo es desarrollar continuamente estrategias y perfeccionar flujos de trabajo que incluyan las actividades abarcadas por las competencias del marco, midiendo los resultados, introduciendo mejoras graduales y madurando el proceso para reducir el tiempo necesario para pasar por estas fases.

3.1.4.1 Informar: Visibilidad y asignación

En la fase de Informar, las actividades de FinOps implican la identificación de fuentes de datos sobre costes, uso y eficiencia de la nube. El uso de estos datos para la asignación, el análisis y la elaboración de informes permite a los equipos desarrollar capacidades de presupuesto, previsión de tendencias, creación de KPI para la evaluación comparativa y desarrollo de métricas que revelarán el valor empresarial del gasto en la nube de una organización.

La asignación precisa de los gastos en la nube basada en etiquetas, cuentas o reglas empresariales permite la elaboración de informes precisos. Los equipos empresariales y financieros deben asegurarse de que están impulsando el rendimiento de la inversión sin salirse del presupuesto, previendo con precisión los gastos y los costes de carbono, y evitando sorpresas. La evaluación comparativa con otros o entre equipos proporciona a las organizaciones métricas para comprender la eficacia con la que están operando. Al combinar todos los datos de costes de la nube con otros datos sobre sostenibilidad, eficiencia, utilización y los puntos de referencia de rendimiento de la organización, los equipos deberían ser capaces de ver los in-

dicadores clave de rendimiento y las métricas unitarias relacionadas con el uso de la nube por parte de la organización.

La naturaleza elástica y bajo demanda de la nube, junto con los complejos descuentos de precios, requiere que las organizaciones revisen continuamente las actividades que informan sus objetivos de negocio a través de decisiones basadas en datos utilizando una visibilidad precisa y oportuna de su uso de la nube.

3.1.4.2 Optimizar: Tarifas y uso

En la fase Optimizar, las actividades de FinOps implican la identificación de oportunidades para mejorar la eficiencia de la nube utilizando los datos y capacidades desarrollados en la fase Informar.

Los proveedores de la nube ofrecen múltiples opciones para optimizar los recursos de la nube. Esto implica la creación de capacidades para adaptar los recursos infrautilizados de la nube, aprovechar las arquitecturas modernas, gestionar las cargas de trabajo y automatizar la eliminación de residuos de los recursos no utilizados.

Además, los proveedores de nube ofrecen opciones para optimizar las tarifas de nube. Esto implica capacidades de visibilidad, análisis y elaboración de informes para potenciar la compra y gestión de modelos de precios con descuento por compromiso y descuento por uso comprometido, como las instancias reservadas (RI), los planes de ahorro (SP) y los descuentos por uso comprometido (CUD).

Esta fase también trata de la colaboración entre equipos para optimizar la visibilidad, la elaboración de informes y los procesos de gestión para las áreas en las que las métricas unitarias indican que el rendimiento de la nube no está alineado con los objetivos de valor de la nube de la organización.

Las opciones de optimización pueden dar lugar a caminos contrapuestos, pero el objetivo subyacente es desarrollar un buen conjunto de oportunidades que ayuden a la organización a obtener más valor de su inversión en la nube.

3.1.4.3 Operar: Mejora continua y uso

En la fase Operar, las actividades de FinOps implican implementar cambios organizativos para hacer operativas las FinOps utilizando los datos y capacidades desarrollados en la fase Informar y Optimizar. Esto incluye el establecimiento de políticas de gobierno de la nube, la supervisión del cumplimiento y la capacitación de las personas mediante el desarrollo de programas de formación, directrices de equipo y políticas de automatización que estén alineadas con los objetivos de la organización.

El éxito de FinOps requiere que las organizaciones construyan una cultura de responsabilidad en la que los equipos de ingeniería, finanzas y negocio colaboren en una acción continua e incremental basada en los datos generados en la fase Informar, seleccionando las mejores oportunidades identificadas en la fase Optimizar y utilizando una predisposición a la acción desarrollada en toda la organización.

Al trabajar en esta fase, hay que tener en cuenta el objetivo de desarrollar estrategias de forma iterativa y perfeccionar los flujos de trabajo; esto implica volver a las fases de Informar y Optimizar para madurar las actividades adoptadas a partir de las competencias del marco, evaluar la introducción de nuevas competencias y hacer evolucionar las operaciones FinOps para la organización.

3.2 Cloud Computing

En los últimos años, la computación en nube ha surgido como una fuerza disruptiva en el sector de las tecnologías de la información (TI). Su impacto es de la misma magnitud que Internet. Para los desarrolladores, arquitectos y responsables de operaciones, la computación en la nube ha provocado un cambio importante en la forma de diseñar, desarrollar, implantar y mantener servicios de software.

La computación en nube democratiza las TI, de forma similar a como Internet democratizó la industria de consumo. Internet abrió a los consumidores un vasto océano de recursos accesibles, que van desde la búsqueda gratuita basada en la publicidad hasta la banca en línea. La computación en nube está trayendo tendencias similares a las pequeñas y grandes empresas. Las empresas pueden ahora cosechar los beneficios de la agilidad simplemente desplegando su software en el centro de datos de otro por una tarifa de consumo. Los costes de hardware quedan fuera de la ecuación gracias a los proveedores de servicios en la nube.[47]

3.2.1 ¿Qué es la computación en la nube?

Cloud computing es la disponibilidad bajo demanda de recursos de computación como servicios a través de Internet. Esta tecnología evita que las empresas tengan que encargarse de aprovisionar, configurar o gestionar los recursos y permite que paguen únicamente por los que usen [48].

Existen tres categorías principales de modelos de servicio en la computación en la nube: infraestructura como servicio, que suministra servicios de almacenamiento y procesamiento; plataforma como servicio, que brinda un entorno para el desarrollo y la implementación de aplicaciones en la nube; y software como servicio, que ofrece aplicaciones disponibles como servicios.

3.2.2 ¿Cómo funciona el computación en la nube?

Los modelos de servicio de computación en la nube se fundamentan en la idea de compartir recursos informáticos, software e información bajo demanda a través de Internet. Tanto empresas como individuos pagan para acceder a un conjunto virtual de recursos compartidos, que incluyen servicios de computación, almacenamiento y redes, ubicados en servidores remotos que son propiedad y están gestionados por proveedores de servicios.

Una de las muchas ventajas que ofrece la computación en la nube es el pago por uso. Así, las organizaciones pueden escalar de manera más rápida y eficiente sin la necesidad de adquirir y mantener sus propios centros de datos físicos y servidores.

En otras palabras, la computación en la nube utiliza una red (generalmente, Internet) para conectar a los usuarios con una plataforma en la nube donde solicitan y acceden a servicios informáticos bajo alquiler. Un servidor central se encarga de toda la comunicación entre los dispositivos y los servidores del cliente para facilitar el intercambio de datos. Las funciones de seguridad y privacidad son componentes habituales para mantener la protección de esta información.

A la hora de adoptar una arquitectura de computación en la nube, no existe una solución única para todos. Lo que funciona para una empresa puede no ser adecuado para otra y sus necesidades empresariales específicas. De hecho, esta flexibilidad y versatilidad son dos de los aspectos distintivos de la nube, permitiendo a las empresas adaptarse rápidamente a los cambios del mercado o de las métricas.

Existen tres modelos distintos de despliegue de la computación en la nube: nube pública, nube privada y nube híbrida.

- Las nubes públicas son gestionadas por proveedores externos de servicios en la nube. Proveen recursos de computación, almacenamiento y red a través de Internet, permitiendo a las empresas acceder a estos recursos bajo demanda, según sus necesidades específicas y objetivos empresariales.
- Las nubes privadas son creadas, gestionadas y propiedad de una sola organización, y se alojan de manera privada en sus propios centros de datos, a menudo denominados "on-premise". Estas nubes ofrecen mayor control, seguridad y gestión de datos, al tiempo que permiten a los usuarios internos beneficiarse de un conjunto compartido de recursos de computación, almacenamiento y redes.
- Las nubes híbridas combinan los modelos de nube pública y privada, permitiendo a las empresas aprovechar los servicios de la nube pública mientras

mantienen las funciones de cumplimiento y seguridad propias de las arquitecturas de nube privada. Este es un enfoque muy usado en grandes empresa, ya que permite aprovechar la flexibilidad y escalabilidad que ofrecen los servicios cloud, con la seguridad y los costes inferiores de los elementos on-premise.

3.2.3 Tipos de computación en la nube

Existen tres modelos principales de servicios de computación en la nube que puedes elegir según el nivel de control, flexibilidad y gestión que necesite tu empresa:

- Infraestructura como Servicio (IaaS): Proporciona acceso bajo demanda a servicios de infraestructura informática, como computación, almacenamiento, redes y virtualización. Este modelo ofrece el máximo control sobre todos tus recursos de TI y se asemeja más a los recursos informáticos tradicionales.
- Plataforma como Servicio (PaaS): Ofrece todos los recursos de hardware y software necesarios para desarrollar aplicaciones en la nube. Con PaaS, las empresas pueden enfocarse completamente en el desarrollo de aplicaciones sin la carga de gestionar y mantener la infraestructura subyacente.
- Software como Servicio (SaaS): Proporciona una solución completa de aplicaciones como servicio, abarcando desde la infraestructura subyacente hasta el mantenimiento y las actualizaciones del software de la aplicación. Este modelo a menudo se presenta como una aplicación para el usuario final, donde tanto el servicio como la infraestructura son gestionados y mantenidos por el proveedor de servicios en la nube.

La Figura 3.5 presenta gráficamente las diferencias entre IaaS, PaaS y SaaS.

3.2.4 Proveedores de infraestructura cloud

Actualmente, existen tres grandes empresa dedicadas a ofrecer servicios de cloud computing y que han revolucionado la forma en que las empresas gestionan su infraestructura tecnológica y desarrollan aplicaciones en la era digital. Estas empresas son Amazon Web Services (AWS)[18], Microsoft Azure[28] y Google Cloud Platform (GCP)[49]. AWS, lanzado en 2006, fue el pionero en ofrecer servicios de computación en la nube a escala global. Comenzó como una iniciativa interna de Amazon para mejorar la eficiencia de sus propios sistemas, y rápidamente se convirtió en un negocio independiente que domina el mercado hasta el día de hoy[50].

Microsoft Azure, originalmente conocido como Windows Azure, entró en el mercado en 2010. Aprovechando la fuerte presencia de Microsoft en el mundo empresarial, Azure ha crecido rápidamente para convertirse en el segundo proveedor de

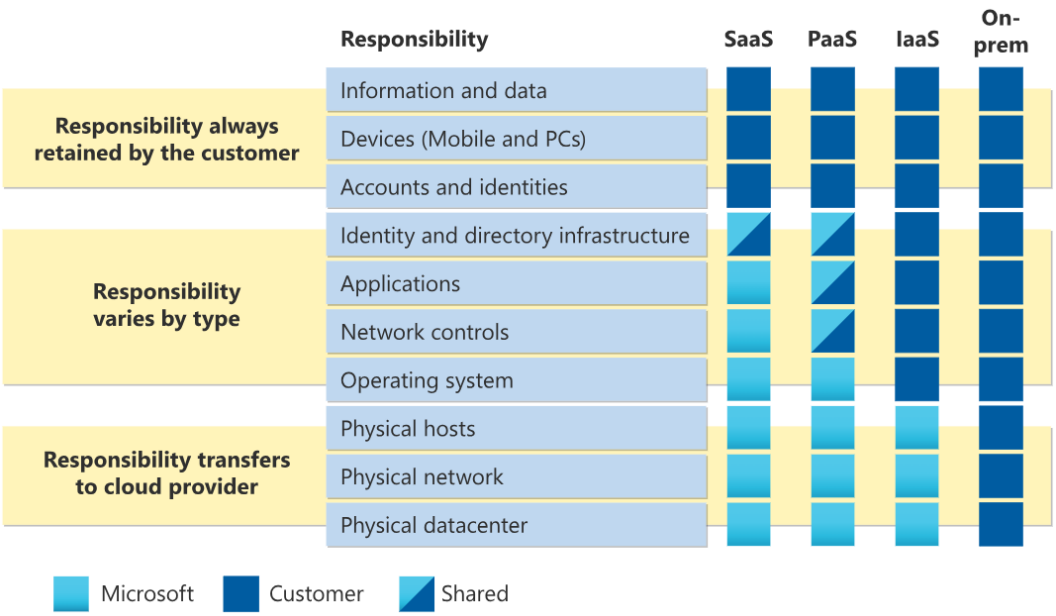


Figure 3.5: Tipos de computación en la nube. [3]

servicios cloud más grande[50].

Google Cloud Platform (GCP), lanzado oficialmente al público en 2011, se basa en la vasta infraestructura tecnológica de Google. Aunque llegó más tarde al mercado, GCP ha ganado terreno rápidamente gracias a sus innovaciones en áreas como el aprendizaje automático y la analítica de datos.

Estos tres proveedores ofrecen una amplia gama de servicios que abarcan desde la infraestructura básica (como almacenamiento y computación) hasta herramientas avanzadas de inteligencia artificial y aprendizaje automático. Cada uno tiene sus propias fortalezas y características únicas, lo que permite a las empresas elegir la plataforma que mejor se adapte a sus necesidades específicas[50].

La competencia entre estos gigantes ha impulsado la innovación continua y la reducción de costos en el sector, beneficiando a empresas de todos los tamaños que buscan aprovechar el poder de la nube para impulsar su transformación digital.

3.2.5 Recursos de Microsoft Azure

En este TFM se ha optado por hacer especial mención a Microsoft Azure, siendo todos ellos igualmente válidos, por el hecho de tener una mayor familiaridad con este entorno. Dentro de este, al igual que en el resto de proveedores, se pueden encontrar una gran variedad de servicios a la disposición de empresas y particulares,

que permitiran desplegar la infraestructura requerida. En los siguientes apartados se explican algunos de ellos.

3.2.5.1 Microsoft Azure Compute

Máquinas virtuales de Azure: Azure Virtual Machines (VM) proporciona la capacidad de crear y utilizar máquinas virtuales en la nube. Estas máquinas virtuales proporcionan infraestructura como servicio (IaaS) en forma de servidores virtualizados y pueden ser empleadas de diversas maneras. Similar a un equipo físico, se puede personalizar todo el software que se ejecuta en la máquina virtual [51]. Las máquinas virtuales son ideales cuando se necesita:

- Control total sobre el sistema operativo (SO).
- Capacidad de ejecutar software personalizado.
- Usar configuraciones de hospedaje personalizadas.

Una máquina virtual de Azure brinda la flexibilidad de la virtualización sin la necesidad de adquirir y mantener el hardware físico que la ejecuta. Sin embargo, como una oferta de IaaS, el usuario será responsable de configurar, actualizar y mantener el software que se ejecuta en la máquina virtual.

Además, es posible crear o utilizar una imagen preexistente para aprovisionar rápidamente máquinas virtuales. El seleccionar una imagen de máquina virtual preconfigurada, permite crear y aprovisionar una máquina virtual en cuestión de minutos. Una imagen es una plantilla utilizada para crear una máquina virtual y puede incluir un sistema operativo y otro software, como herramientas de desarrollo o entornos de hospedaje web.

Conjuntos de escalado de máquinas virtuales: Los conjuntos de escalado de máquinas virtuales permiten crear y gestionar un grupo de máquinas virtuales idénticas con equilibrio de carga. Si se crean varias máquinas virtuales con el mismo propósito, sería necesario que asegurarse de que todas estén configuradas de forma idéntica y configurar los parámetros de enrutamiento de red para garantizar la eficacia. Además, ser necesitaría supervisar el uso para determinar si es necesario aumentar o disminuir el número de máquinas virtuales.

En cambio, con los conjuntos de escalado de máquinas virtuales, Azure automatiza gran parte de ese trabajo. Estos conjuntos permiten administrar, configurar y actualizar centralmente un gran número de máquinas virtuales en cuestión de minutos. El número de instancias de máquinas virtuales puede aumentar o disminuir automáticamente según la demanda, o ajustarse para escalarse en función de una programación definida. Los conjuntos de escalado de máquinas virtuales también

implementan automáticamente un equilibrador de carga para asegurar que los recursos se utilicen de manera eficaz. Estos conjuntos permiten crear servicios a gran escala para áreas como procesamiento, macrodatos y cargas de trabajo de contenedores.[51]

Conjuntos de disponibilidad de máquinas virtuales: Los conjuntos de disponibilidad de máquinas virtuales son una herramienta clave para crear un entorno más resiliente y con alta disponibilidad. Estos conjuntos están diseñados para asegurar que las máquinas virtuales puedan manejar actualizaciones y mantener una conectividad de red y potencia diversificada, evitando la pérdida de todas las máquinas virtuales debido a un solo fallo de energía o red.

Los conjuntos de disponibilidad logran esto mediante la agrupación de las máquinas virtuales en dos formas: dominio de actualización y dominio de error[51].

- Dominio de actualización: Agrupa las máquinas virtuales que pueden reiniciarse simultáneamente. Esto permite aplicar actualizaciones sabiendo que solo un grupo de dominios de actualización estará fuera de línea a la vez. Todas las máquinas de un dominio de actualización se actualizarán juntas. Se asigna un tiempo de recuperación de 30 minutos a un grupo de actualizaciones antes de iniciar el mantenimiento en el siguiente dominio de actualización.
- Dominio de error: Agrupa las máquinas virtuales por fuente de alimentación común y conmutador de red. Por defecto, un conjunto de disponibilidad dividirá las máquinas virtuales en un máximo de tres dominios de error. Esto protege contra fallos físicos de energía o red al distribuir las máquinas virtuales en diferentes dominios de error, conectándolas a recursos de energía y red distintos.

Configurar un conjunto de disponibilidad no tiene costo adicional. Solo se paga por las instancias de máquinas virtuales creadas.

Funciones de Azure: Azure Functions es una opción de procesamiento sin servidor basada en eventos que no requiere el mantenimiento de máquinas virtuales ni contenedores. Si se desarrolla una aplicación utilizando máquinas virtuales o contenedores, estos recursos deben estar en funcionamiento para que la aplicación funcione. Con Azure Functions, un evento activa la función, reduciendo la necesidad de mantener recursos aprovisionados cuando no hay eventos.

El uso de Azure Functions es ideal si solo es necesario el código que ejecuta el servicio y no la infraestructura o plataforma subyacente. Las funciones se utilizan comúnmente para realizar tareas en respuesta a un evento (a menudo mediante

una llamada API REST), un temporizador o un mensaje de otro servicio de Azure, y cuando esas tareas pueden completarse rápidamente, en cuestión de segundos o menos[52].

Algunas características de Azure Functions son:

- Azure Functions se escala automáticamente según la demanda, lo que lo convierte en una opción adecuada cuando la demanda es variable.
- Azure Functions ejecuta el código cuando se desencadena y desasigna recursos automáticamente cuando la función finaliza. En este modelo, solo se paga por el tiempo de CPU utilizado mientras se ejecuta la función.
- Las funciones pueden ser sin estado o con estado. Cuando son sin estado (por defecto), se comportan como si se reiniciaran cada vez que responden a un evento. Cuando son con estado (llamadas Durable Functions), se pasa un contexto a través de la función para realizar un seguimiento de la actividad.

Las funciones son un componente clave de la informática sin servidor y también son una plataforma de procesamiento general para ejecutar cualquier tipo de código. Si cambian las necesidades de la aplicación del desarrollador, se puede implementar el proyecto en un entorno que no sea sin servidor. Esta flexibilidad permite gestionar el escalado, operar en redes virtuales e incluso aislar completamente las funciones.

3.2.5.2 Microsoft Azure Networking

Redes virtuales de Azure: Las redes virtuales y las subredes virtuales de Azure permiten que los recursos de Azure, como máquinas virtuales, aplicaciones web y bases de datos, se comuniquen entre sí, con los usuarios de Internet y con los equipos cliente en entornos locales. Una red de Azure puede considerarse una extensión de la red local que conecta otros recursos de Azure[53]. Las redes virtuales de Azure admiten puntos de conexión públicos y privados para facilitar la comunicación entre recursos externos o internos con otros recursos internos.

- Puntos de conexión públicos: Tienen una dirección IP pública y son accesibles desde cualquier parte del mundo.
- Puntos de conexión privados: Existen dentro de una red virtual y tienen una dirección IP privada en el espacio de direcciones de esa red virtual.

Las redes virtuales de Azure ofrecen varias funcionalidades de red importantes:

- **Aislamiento y segmentación.**

  La red virtual de Azure permite la creación de múltiples redes virtuales aisladas. Al configurar una red virtual, se define un espacio de direcciones IP

privadas utilizando intervalos de direcciones IP públicas o privadas. Este intervalo IP existe únicamente dentro de la red virtual y no es enrutado a través de Internet. Posteriormente, puedes subdividir ese espacio de direcciones IP en subredes y asignar partes del espacio definido a cada subred con nombres específicos.

Para la resolución de nombres, Azure ofrece un servicio integrado de resolución de nombres. También tienes la opción de configurar la red virtual para que utilice un servidor DNS interno o externo.

- **Comunicación con Internet.**

  Puede permitir conexiones entrantes desde Internet mediante la asignación de una dirección IP pública a un recurso de Azure o la colocación del recurso detrás de un equilibrador de carga público.

- **Comunicación entre recursos de Azure.**

  Para asegurar la comunicación entre los recursos de Azure de manera segura, se puede proceder mediante dos enfoques principales:

  - Conexión dentro de redes virtuales: Las redes virtuales no solo permiten conectar máquinas virtuales, sino también otros recursos de Azure como App Service Environment para Power Apps, Azure Kubernetes Service y conjuntos de escalado de máquinas virtuales de Azure. Esta integración dentro de una red virtual facilita la comunicación segura y eficiente entre diversos servicios de Azure bajo un entorno controlado y aislado.

  - Puntos de conexión de servicio: Estos puntos permiten la conexión de otros tipos de recursos de Azure, como cuentas de almacenamiento y bases de datos Azure SQL, a las redes virtuales. Este enfoque permite vincular varios recursos de Azure directamente a las redes virtuales, mejorando la seguridad al evitar la exposición pública de estos servicios. Además, proporciona un enrutamiento optimizado y controlado del tráfico entre los recursos conectados.

- **Comunicación con recursos locales.**

  Las redes virtuales de Azure permiten conectar entre sí los recursos tanto del entorno local como dentro de la suscripción de Azure. Es posible crear una red que abarque ambos entornos, utilizando tres mecanismos principales para lograr esta conectividad:

  - Conexiones de Red Privada Virtual (VPN) de punto a sitio: Estas

conexiones se establecen desde un dispositivo fuera de la organización hacia la red corporativa. El dispositivo cliente inicia una conexión VPN cifrada para conectarse a la red virtual de Azure. Esta opción es útil cuando se necesita acceso seguro a recursos de Azure desde dispositivos individuales fuera de la red corporativa.

  - Redes Virtuales Privadas de sitio a sitio: Este mecanismo vincula el dispositivo o la puerta de enlace de VPN local con la puerta de enlace de VPN de Azure dentro de una red virtual. Esto permite que los dispositivos en la red local se comuniquen de forma segura con los recursos de Azure como si estuvieran en la misma red local. La conexión se cifra y opera a través de Internet.

  - Azure ExpressRoute: Proporciona una conectividad privada dedicada a Azure que no transita por Internet público. ExpressRoute es ideal para entornos que requieren alto ancho de banda y niveles elevados de seguridad. Esta opción garantiza una conexión más confiable y predecible entre la infraestructura local y los recursos de Azure, sin necesidad de pasar por Internet.

- **Enrutamiento del tráfico de red**

  De manera predeterminada, Azure gestiona la dirección del tráfico entre las subredes de todas las redes virtuales conectadas, las redes locales e Internet. Sin embargo, es posible modificar y personalizar este enrutamiento de la siguiente manera:

  - Las tablas de enrutamiento permiten establecer reglas específicas para dirigir el tráfico. Es factible crear tablas de enrutamiento personalizadas que determinen cómo se encaminan los paquetes entre las subredes.

  - El Protocolo de puerta de enlace de borde (BGP) se utiliza con las puertas de enlace de VPN de Azure, Azure Route Server o Azure ExpressRoute para propagar las rutas BGP locales hacia las redes virtuales de Azure.

- **Filtrado del tráfico de red**

  Las redes virtuales de Azure ofrecen métodos para filtrar el tráfico entre las subredes de la siguiente manera:

  - Los grupos de seguridad de red son recursos de Azure que contienen reglas de seguridad configurables. Estas reglas determinan si el tráfico debe permitirse o bloquearse según el protocolo, puerto, y direcciones IP de origen y destino.

  - Las aplicaciones virtuales de red son máquinas virtuales especializadas que desempeñan roles similares a dispositivos de red protegidos. Estas aplicaciones pueden funcionar como firewalls o realizar optimización de redes de área extensa (WAN), entre otras funciones específicas de red.

- **Conexión de redes virtuales**

  Se puede conectar redes virtuales entre sí mediante el emparejamiento de redes virtuales. Esta función permite establecer una conexión directa entre dos redes virtuales. El tráfico de red entre redes emparejadas se mantiene privado y se transmite a través de la infraestructura central de Microsoft, sin pasar por la red pública de Internet. El emparejamiento facilita la comunicación entre los recursos de cada red virtual, incluso si se encuentran en diferentes regiones, lo que posibilita la creación de una red global interconectada dentro de Azure.

  Por otro lado, las Rutas Definidas por el Usuario (UDR) permiten gestionar las tablas de enrutamiento dentro de una red virtual o entre redes virtuales. Esto proporciona un control más detallado sobre cómo se dirige el tráfico de red.

DNS de Azure: Azure DNS es un servicio de hospedaje para dominios DNS que ofrece resolución de nombres utilizando la infraestructura de Microsoft Azure. Al utilizar Azure para hospedar dominios, puedes gestionar los registros DNS utilizando las mismas credenciales, API, herramientas y esquema de facturación que con otros servicios de Azure[54].

Las ventajas de Azure DNS incluyen:

- Confiabilidad y rendimiento: Los dominios DNS de Azure DNS se hospedan en la red global de servidores de nombres DNS de Azure, y proporcionan resistencia y alta disponibilidad.

- Seguridad: Azure DNS se basa en Azure Resource Manager, que proporciona características tales como control de acceso basado en rol de Azure (Azure RBAC), registros de actividad y Bloqueo de recursos para bloquear una suscripción, un grupo de recursos o un recurso.

- Facilidad de uso: Azure DNS puede administrar registros DNS para los servicios de Azure y también proporciona el servicio de nombres de dominio para los recursos externos. Azure DNS está integrado en Azure Portal y usa las mismas credenciales, la misma facturación y el mismo contrato de soporte técnico que los demás servicios de Azure.

- Redes virtuales personalizables: Azure DNS es compatible con dominios DNS privados. Esta característica permite usar nombres de dominio personalizados propios en las redes virtuales privadas, en lugar de limitarse a los nombres proporcionados por Azure.
- Registros de alias: Azure DNS también admite conjuntos de registros de alias. Puede usar un conjunto de registros de alias que haga referencia a un recurso de Azure, como una dirección IP pública de Azure, un perfil de Azure Traffic Manager o un punto de conexión de Azure Content Delivery Network (CDN).

3.2.5.3 Microsoft Azure Storage

Cuentas de almacenamiento de Azure: Una cuenta de almacenamiento proporciona un espacio de nombres único para los datos de Azure Storage, accesible globalmente a través de HTTP o HTTPS. Los datos en esta cuenta son seguros, altamente disponibles, duraderos y pueden escalar de manera masiva[55].

Al crear una cuenta de almacenamiento, el primer paso es seleccionar el tipo de cuenta. Este tipo determina los servicios de almacenamiento disponibles, las opciones de redundancia y tiene impacto en los casos de uso específicos. A continuación se enumeran las opciones de redundancia que se explorarán más adelante:

- Almacenamiento con redundancia local (LRS)
- Almacenamiento con redundancia geográfica (GRS)
- Almacenamiento con redundancia geográfica con acceso de lectura (RA-GRS)
- Almacenamiento con redundancia de zona (ZRS)
- Almacenamiento con redundancia de zona geográfica (GZRS)
- Almacenamiento con redundancia de zona geográfica con acceso de lectura (RA-GZRS)

Servicios de almacenamiento de Azure: La plataforma de Azure Storage ofrece varios servicios de datos[56]:

- **Blobs de Azure:** Azure Blob Storage es una solución de almacenamiento de objetos diseñada para la nube. Este servicio puede manejar grandes volúmenes de datos, como archivos de texto o binarios. No tiene una estructura definida, lo que significa que puede almacenar cualquier tipo de

dato sin restricciones. Azure Blob Storage es capaz de gestionar simultáneamente miles de cargas, grandes cantidades de datos de vídeo, archivos de registro en constante expansión y es accesible desde cualquier ubicación con conexión a Internet.

- **Azure Files:** Azure Files proporciona recursos compartidos de archivos completamente administrados en la nube, accesibles mediante los protocolos estándar del sector como SMB (Server Message Block) o NFS (Network File System). Estos recursos compartidos de archivos pueden ser montados simultáneamente en implementaciones locales o en la nube. Los recursos compartidos de archivos SMB de Azure Files son accesibles desde clientes Windows, Linux y macOS, mientras que los recursos compartidos de archivos NFS de Azure Files son accesibles desde clientes Linux y macOS. Además, los recursos compartidos de archivos SMB de Azure Files pueden ser almacenados en caché en servidores de Windows Server utilizando Azure File Sync, lo que permite un acceso rápido a los datos dondequiera que se utilicen.

- **Colas de Azure:** Azure Queue Storage es un servicio diseñado para almacenar una gran cantidad de mensajes que pueden ser accedidos desde cualquier parte del mundo mediante llamadas autenticadas con HTTP o HTTPS. Cada cola puede contener tantos mensajes como permita el espacio disponible en la cuenta de almacenamiento, lo que puede ascender a millones de mensajes. Cada mensaje individual en la cola puede tener un tamaño máximo de hasta 64 KB. Las colas se utilizan típicamente para gestionar trabajos pendientes que deben procesarse de manera asincrónica.

- **Azure Disks:** Azure Disk, también conocido como discos administrados de Azure, son volúmenes de almacenamiento a nivel de bloque gestionados por Azure para su uso con máquinas virtuales de Azure. Aunque son conceptualmente similares a los discos físicos, están virtualizados, lo que proporciona una mayor resistencia y disponibilidad. Con los discos administrados, solo necesitas aprovisionar el disco; Azure se encarga de todas las operaciones adicionales, como la replicación para alta disponibilidad, la gestión de fallos y la recuperación automatizada.

- **Tablas de Azure:** Azure Table Storage proporciona capacidad para almacenar grandes volúmenes de datos estructurados. Se trata de un almacén de datos NoSQL que acepta llamadas autenticadas desde dentro y fuera de la infraestructura de Azure. Esto facilita su uso en soluciones híbridas o en entornos multi-nube, asegurando la disponibilidad continua de los datos. Azure Table Storage es especialmente adecuado para el almacenamiento de datos estructurados no relacionales.

3.3 Automatización de infraestructura

Antes de la llegada de la computación en nube, la mayoría de los entornos de infraestructura se construían manualmente. La construcción de todos los entornos, incluido el de producción, corría a cargo de ingenieros de infraestructuras que utilizaban técnicas y procesos manuales. Había algo de scripting y automatización, pero incluso esos scripts se ejecutaban manualmente. Por lo general, la infraestructura no sufría cambios y era creada y mantenida exclusivamente por ingenieros de infraestructuras. No había mucha necesidad de escalar la infraestructura, ni tampoco se producían cambios perturbadores en ella, aparte del hardware defectuoso. Este mantenimiento manual funcionaba bien en la mayoría de los casos[57].

Las cosas han cambiado rápidamente a lo largo de los años, especialmente después de que la nube ganara una tracción significativa con cada vez más organizaciones trasladando sus aplicaciones y despliegues a la nube. La nube aporta agilidad y flexibilidad tanto al aprovisionamiento como al mantenimiento de las aplicaciones. Esta agilidad se traduce en despliegues más rápidos, frecuentes, coherentes y predecibles.

La arquitectura de las aplicaciones y los modelos de despliegue también han cambiado en los últimos años. Las aplicaciones se están dividiendo en microservicios, cada uno con su propio ciclo de vida de desarrollo y despliegue, y cada uno con sus propios requisitos de infraestructura en cuanto a tamaño, rendimiento, escalabilidad, recuperación ante desastres y disponibilidad.

Con este aumento del tamaño, la complejidad y el número de implantaciones de infraestructura, cada vez resultaba más difícil continuar con las formas manuales tradicionales de administrar, gestionar y configurar la infraestructura. Además, los pasos manuales para crear el entorno de infraestructura provocaban retrasos debido a la menor coherencia, previsibilidad y estandarización.

Era necesario automatizar el aprovisionamiento, la gestión y la evolución de la infraestructura con mayor coherencia. Además, era necesario que los despliegues fueran más predecibles para obtener un mayor nivel de confianza y que el aprovisionamiento se estandarizara para que dependiera más de los procesos que de las personas.

También empezó a hacerse evidente durante este tiempo que la infraestructura es una responsabilidad compartida entre desarrolladores y operaciones, similar a una aplicación, en lugar de ser propiedad de un grupo ingenieros de infraestructura.

Un nuevo paradigma llamado infraestructura como código (IaC) surgió debido a todos los retos a los que se enfrentaba la implantación y configuración manual

de la infraestructura. La IaC ayuda a tratar la infraestructura como parte de la solución global, convirtiéndola en código y haciéndola pasar por el mismo ciclo de vida, pasos y procesos que seguiría una aplicación. Garantiza que la infraestructura pase a formar parte de las prácticas de ingeniería de software de forma similar a una aplicación y que se le apliquen todos los principios y procesos de ingeniería. Esto incluye la creación del código de la infraestructura, el control de versiones, distintos niveles y tipos de pruebas (unitarias, de integración, de aceptación), linting, etc.[57]

3.3.1 Infraestructura como Código

La infraestructura como código (IaC) permite aprovisionar y respaldar la infraestructura de computación mediante código en lugar de configuraciones y procesos manuales. Cualquier entorno de aplicaciones necesita diversos componentes de infraestructura, como sistemas operativos, conexiones a bases de datos y almacenamiento. Los desarrolladores deben configurar, actualizar y mantener la infraestructura periódicamente para desarrollar, probar y desplegar aplicaciones.

La administración manual de la infraestructura consume tiempo y es propensa a errores, especialmente al manejar aplicaciones a gran escala. La infraestructura como código permite definir el estado deseado de la infraestructura sin detallar todos los pasos para alcanzarlo. Automatiza la gestión de la infraestructura, permitiendo a los desarrolladores centrarse en crear y mejorar aplicaciones en lugar de gestionar entornos. Las organizaciones utilizan la infraestructura como código para controlar costos, reducir riesgos y responder rápidamente a nuevas oportunidades de negocio[58].

3.3.1.1 Enfoque declarativo frente a imperativo

Al elegir una solución de IaC, también es importante comprender la diferencia entre un enfoque declarativo o imperativo de la automatización de infraestructuras[59].

- **Enfoque declarativo:** En la mayoría de las organizaciones, el enfoque declarativo es el más adecuado. En el enfoque declarativo, se especifica el estado final de la infraestructura que se desea aprovisionar y el software IaC se encarga del resto: poner en marcha la máquina virtual (VM) o el contenedor, instalar y configurar el software necesario, resolver las interdependencias de sistemas y software y gestionar el control de versiones. El principal inconveniente del enfoque declarativo es que suele requerir un administrador cualificado para su instalación y gestión, y estos administradores suelen estar especializados en su solución preferida.

- **Enfoque imperativo:** En el enfoque imperativo la solución consiste en preparar scripts de automatización que aprovisionan la infraestructura paso a paso. Aunque esto puede suponer más trabajo de gestión a medida que se escala, puede ser más fácil de entender para el personal administrativo existente y puede aprovechar las secuencias de comandos de configuración que ya usadas.

3.3.2 Herramientas de IaC

Aunque existe una gran variedad de opciones para poder desarrollar infraestructura como código, las dos herramientas más usadas para ello son Terraform y Ansible.[59]

- **Ansible**[60] es un proyecto comunitario de código abierto patrocinado por Red Hat y diseñado para ayudar a las organizaciones a automatizar el aprovisionamiento, la gestión de la configuración y el despliegue de aplicaciones. Ansible es una herramienta de automatización declarativa que permite crear "playbooks" (escritos en el lenguaje de configuración YAML) para especificar el estado deseado de la infraestructura y, a continuación, realizar el aprovisionamiento. Ansible es una opción popular para automatizar el aprovisionamiento de contenedores Docker y despliegues Kubernetes.

- **Terraform**[20] es otra herramienta declarativa de aprovisionamiento de infraestructuras que permite a los ingenieros automatizar el aprovisionamiento de todos los aspectos de su infraestructura empresarial basada en la nube y en las instalaciones. Funciona con los principales proveedores de nube y permite automatizar la creación de recursos en varios proveedores en paralelo, independientemente de dónde residan los servidores físicos, los servidores DNS o las bases de datos. A diferencia de Ansible, Terraform no ofrece capacidades de gestión de la configuración, pero trabaja mano a mano con herramientas de gestión de la configuración para aprovisionar automáticamente la infraestructura en el estado descrito por los archivos de configuración y para cambiar automáticamente el aprovisionamiento de actualización cuando sea necesario en respuesta a los cambios de configuración.

3.3.3 Terraform

En este apartado se ahondará en la herramienta Terraform. Aunque existen otras herramientas que permiten la ejecuación de código IaC, se ha elegido esta debido a su sencillez de aplicación y a la capacidad que tiene de integrarse con proveedores de infreaestructura cloud, en este caso Microsoft Azure.

3.3.3.1 ¿Qué es Terraform?

HashiCorp Terraform es una herramienta de infraestructura como código que permite definir recursos en la nube y locales en archivos de configuración legibles por humanos que se pueden versionar, reutilizar y compartir. Permite ejecutar procesos para aprovisionar y gestionar infraestructura a lo largo de su ciclo de vida. Terraform puede gestionar componentes de bajo nivel como recursos informáticos, de almacenamiento y de red, así como componentes de alto nivel como entradas DNS y funciones SaaS.[4]

3.3.3.2 ¿Cómo funciona terraform?

Terraform crea y gestiona recursos en plataformas en la nube y otros servicios a través de sus APIs. Los proveedores permiten que Terraform funcione con prácticamente cualquier plataforma o servicio con una API accesible.

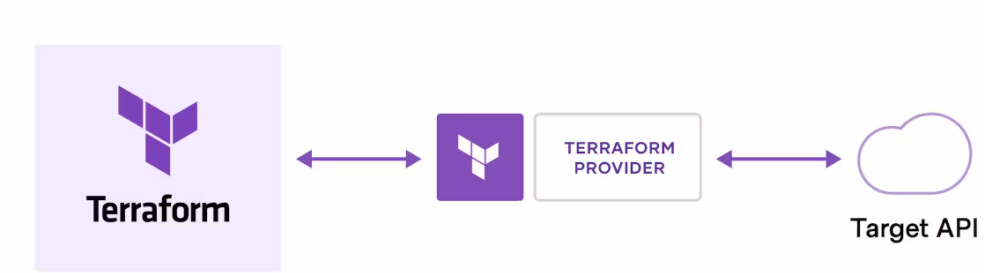


Figure 3.6: Esquema de funcionamiento de Terraform. [4]

HashiCorp y la comunidad de Terraform han desarrollado un gran número de proveedores para gestionar diferentes tipos de recursos y servicios. Estos se pueden encontrar públicamente disponibles en el Registro de Terraform[61], incluidos Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog, y muchos más.

El flujo de trabajo central de Terraform consta de tres etapas:

- **Escribir:** Definir recursos que pueden estar distribuidos en múltiples proveedores de nube y servicios. Por ejemplo, se podría crear una configuración para desplegar una aplicación en máquinas virtuales en una red Virtual Privada Cloud (VPC) con grupos de seguridad y un balanceador de carga.
- **Planificar:** Terraform crea un plan de ejecución que describe la infraestructura que creará, actualizará o eliminará basándose en la infraestructura existente y la configuración definida.

- **Aplicar:** Una vez aprobado, Terraform lleva a cabo las operaciones propuestas en el orden correcto, respetando las dependencias entre recursos. Por ejemplo, si se actualizan las propiedades de una VPC y se cambian el número de máquinas virtuales en esa VPC, Terraform recreará primero la VPC antes de escalar las máquinas virtuales.

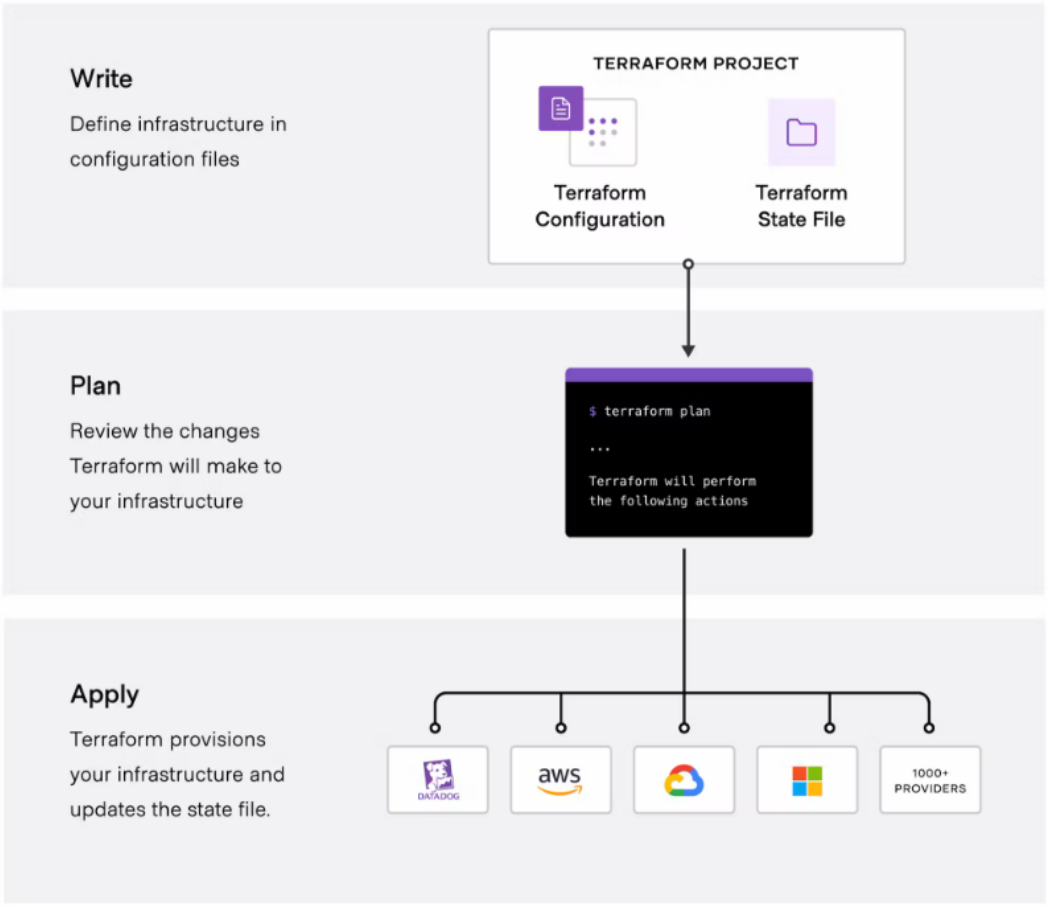


Figure 3.7: Proceso de ejecución de Terraform. [4]

3.3.4 Git

Un sistema de control de versiones (VCS, por sus siglas en inglés) es una herramienta que registra el historial de modificaciones realizadas en un proyecto, facilitando la colaboración entre individuos y equipos. A medida que los desarrolladores efectúan cambios en el proyecto, cualquier versión previa puede ser recuperada en cualquier momento.[62]

Los desarrolladores pueden consultar el historial del proyecto para determinar:

- ¿Qué modificaciones se realizaron?
- ¿Quién efectuó dichas modificaciones?
- ¿Cuándo fueron realizadas?
- ¿Cuál fue la razón detrás de estos cambios?

En un sistema de control de versiones distribuido (DVCS), cada desarrollador posee una copia completa del proyecto y su historial. A diferencia de los sistemas de control de versiones centralizados, los DVCS no requieren una conexión constante a un repositorio central. Git, el sistema de control de versiones distribuido más utilizado, es ampliamente empleado tanto en proyectos de código abierto como en desarrollos comerciales, ofreciendo ventajas significativas a individuos, equipos y organizaciones.

Git permite a los desarrolladores acceder a la línea de tiempo completa de los cambios, decisiones y evolución de cualquier proyecto en un solo lugar. Al revisar el historial del proyecto, los desarrolladores obtienen todo el contexto necesario para comprenderlo y comenzar a contribuir de inmediato.

Dado que los desarrolladores pueden estar ubicados en diferentes zonas horarias, un DVCS como Git permite colaboraciones en cualquier momento, manteniendo la integridad del código fuente. Utilizando ramas, los desarrolladores pueden proponer cambios al código de producción de manera segura.

Las empresas que utilizan Git pueden eliminar barreras de comunicación entre equipos y centrarse en maximizar su rendimiento. Además, Git facilita la colaboración de expertos en distintos proyectos principales dentro de la organización.

3.3.4.1 Repositorios de Git

Un repositorio o proyecto en Git abarca toda la colección de archivos y carpetas relacionados con un proyecto, junto con el historial de revisiones de cada archivo. El historial se presenta como instantáneas en el tiempo, conocidas como "confirmaciones". Estas confirmaciones pueden organizarse en múltiples líneas de desarrollo denominadas "ramas". Dado que Git es un sistema de control de versiones distribuido (DVCS), los repositorios son unidades autónomas y cualquiera que posea una copia del repositorio puede acceder a todo el código fuente y su historial. Utilizando la línea de comandos o cualquier otra interfaz amigable, un repositorio de Git también permite interactuar con el historial, clonar el repositorio, crear ramas, confirmar cambios, fusionar y comparar versiones del código, entre otras acciones.

A través de plataformas como GitHub, Git ofrece aún más oportunidades para la transparencia y la colaboración en los proyectos. Los repositorios públicos per-

miten que los equipos trabajen conjuntamente para crear el mejor producto final posible.[62]

3.3.4.2 Funcionamiento de Github

GitHub aloja repositorios de Git y proporciona a los desarrolladores herramientas para mejorar la calidad del código a través de características como la línea de comandos, debates en hilo (propuestas), solicitudes de cambio, revisión de código, y el uso de una variedad de aplicaciones gratuitas y de pago disponibles en GitHub Marketplace. Con capas de colaboración como el flujo de trabajo de GitHub, una comunidad de 100 millones de desarrolladores y un ecosistema con cientos de integraciones, GitHub transforma la forma en que se desarrolla el software.[62]

GitHub integra la colaboración directamente en el proceso de desarrollo. El trabajo se organiza en repositorios, donde los desarrolladores pueden especificar los requisitos, orientar el proyecto y establecer expectativas para los miembros del equipo. Utilizando el flujo de trabajo de GitHub, los desarrolladores crean una rama para trabajar en las actualizaciones, confirman cambios para guardarlos, abren una solicitud de cambios para proponer y debatir modificaciones, y finalmente fusionan las solicitudes de cambio una vez que se ha alcanzado un consenso.

3.3.4.3 Comandos básicos de Git

Para utilizar Git, los desarrolladores emplean comandos específicos para copiar, crear, modificar y combinar el código. Estos comandos pueden ejecutarse directamente desde la línea de comandos o mediante una aplicación como GitHub Desktop. A continuación, se presentan algunos comandos comunes para el uso de Git[63]:

- **git init** inicializa un nuevo repositorio de Git y comienza a supervisar el directorio existente. Este comando agrega una subcarpeta oculta dentro del directorio que contiene la estructura de datos necesaria para el control de versiones.
- **git clone** crea una copia local de un proyecto que ya existe de forma remota. El clon incluye todos los archivos, historial y ramas del proyecto.
- **git add** almacena provisionalmente un cambio. Git rastrea los cambios realizados en la base de código, pero es necesario preparar estos cambios y tomar una instantánea de ellos para incluirlos en el historial del proyecto. Este comando realiza la preparación, la primera parte del proceso de dos pasos. Cualquier cambio preparado se convertirá en parte de la siguiente instantánea y del historial del proyecto. La preparación y la confirmación

por separado otorgan a los desarrolladores un control total sobre el historial y el proyecto sin alterar su forma de codificar y trabajar.

- **git commit** guarda la instantánea del historial del proyecto y completa el proceso de seguimiento de cambios. En resumen, una confirmación funciona como tomar una fotografía: todo lo que se haya almacenado provisionalmente con git add se incluye en la instantánea con git commit.
- **git status** permite mostrar el estado de los cambios, indicando si están sin seguimiento, modificados o almacenados provisionalmente.
- **git branch** muestra las ramas en las que se está trabajando localmente.
- **git merge** combina las líneas de desarrollo. Este comando se utiliza habitualmente para fusionar los cambios realizados en dos ramas distintas. Por ejemplo, un desarrollador podría utilizar una fusión para combinar los cambios de una rama de características en la rama de desarrollo principal.
- **git pull** actualiza la línea de desarrollo local con las actualizaciones de sus contrapartes remotas. Los desarrolladores emplean este comando si un compañero de equipo ha realizado confirmaciones en una rama de un repositorio remoto y desean reflejar esos cambios en su entorno local.
- **git push** actualiza el repositorio remoto con las confirmaciones realizadas localmente en una rama.

3.3.4.4 Github Actions

GitHub Actions es una plataforma de integración y despliegue continuos (CI/CD) la cual permite automatizar la compilación, pruebas y despliegue de proyectos. Se pueden crear flujos de trabajo que construyan y prueben cada solicitud de cambio en el repositorio, así como desplegar solicitudes de cambio fusionadas a producción.[64]

Más allá del ámbito de DevOps, GitHub Actions permite ejecutar flujos de trabajo en respuesta a diversos eventos en tu repositorio. Por ejemplo, se puede configurar un flujo de trabajo para que agregue automáticamente etiquetas adecuadas cada vez que alguien cree una nueva propuesta en el repositorio.

GitHub proporciona máquinas virtuales con sistemas operativos Linux, Windows y macOS para ejecutar flujos de trabajo. Además, permite la opción de alojar ejecutores propios en un centro de datos o infraestructura en la nube.

A continuación, se describen algunos de los componentes principales de Github Actions.[64]

Workflows: Un flujo de trabajo es un proceso automatizado configurable que ejecuta uno o más trabajos (jobs). Los flujos de trabajo se definen mediante archivos YAML que se registran en tu repositorio y se ejecutan cuando son activados por un evento dentro del mismo, o bien, pueden activarse manualmente o según una programación definida.

Estos flujos de trabajo se ubican en el directorio .github/workflows de un repositorio. Un repositorio puede contener múltiples flujos de trabajo, cada uno destinado a realizar diferentes tareas. Por ejemplo, puedes tener un flujo de trabajo para construir y probar las solicitudes de cambio, otro para desplegar tu aplicación cada vez que se crea una versión, y otro para agregar etiquetas automáticamente cada vez que alguien abra una nueva propuesta.

Eventos: Un evento es una acción específica que ocurre en un repositorio y que puede desencadenar la ejecución de un flujo de trabajo. Estas acciones incluyen actividades como la creación de una solicitud de cambios, la apertura de una propuesta, o la realización de una confirmación en un repositorio en GitHub. Además, los flujos de trabajo pueden activarse según una programación definida, mediante la invocación de una API REST, o de manera manual.

Trabajos: Un trabajo (job) en el contexto de GitHub Actions es un conjunto de pasos dentro de un flujo de trabajo que se ejecuta en un mismo entorno o ejecutor. Cada paso puede consistir en un script de shell o una acción que realiza una tarea específica. Los pasos se ejecutan en secuencia y pueden depender unos de otros, lo que permite compartir datos entre ellos. Por ejemplo, se puede tener un paso que compile una aplicación y luego otro paso que realice pruebas sobre esa misma aplicación compilada.

Los trabajos por defecto no tienen dependencias entre ellos y se ejecutan en paralelo. Sin embargo, es posible configurar dependencias entre trabajos para controlar el orden de ejecución. Cuando un trabajo tiene dependencia de otro trabajo, esperará a que el trabajo dependiente finalice satisfactoriamente antes de comenzar su ejecución. Por ejemplo, podrías tener múltiples trabajos de compilación para diferentes arquitecturas que se ejecutan en paralelo y un trabajo de empaquetado que depende de estos trabajos. Los trabajos de compilación se ejecutarán simultáneamente y, una vez que todos hayan completado exitosamente, se ejecutará el trabajo de empaquetado.

Acciones: Una acción en GitHub Actions es una aplicación personalizada diseñada para la plataforma, que realiza tareas complejas y recurrentes de manera

automatizada. Utilizar acciones permite reducir la cantidad de código repetitivo que se debe escribir en los archivos de flujo de trabajo.

Se pueden crear acciones personalizadas según las necesidades específicas de un proyecto, o bien, es posible buscar y utilizar acciones disponibles en el GitHub Marketplace que ya han sido desarrolladas por otros usuarios.

Ejecutores: Un ejecutor es un servidor que se encarga de ejecutar flujos de trabajo cuando son activados. Cada ejecutor puede manejar la ejecución de un job individual a la vez. GitHub ofrece ejecutores que soportan Ubuntu Linux, Microsoft Windows y macOS para ejecutar los flujos de trabajo. Cada flujo de trabajo se ejecuta en una máquina virtual nueva y recién provisionada según las necesidades del job.

Además de los ejecutores estándar, GitHub también dispone de ejecutores más grandes que están configurados con capacidades extendidas y están disponibles en configuraciones más robustas. Si se necesita utilizar otro sistema operativo o una configuración de hardware específica que no está disponible en los ejecutores estándar, cabe la opción de hospedar ejecutores propios para satisfacer tus requisitos particulares.

Capítulo 4

Implementación

En el contexto actual de la computación en la nube, la capacidad de escalar de manera eficiente los recursos de infraestructura es fundamental para optimizar costos y garantizar un rendimiento óptimo de las aplicaciones desplegadas. Este Trabajo Fin de Máster (TFM) presenta una solución orientada a la gestión automatizada del escalado y desescalado de máquinas virtuales dentro de un Scale Set de Azure, integrando consideraciones financieras para la empresa o proyecto en cuestión.

La solución desarrollada se basa en el uso de Terraform para definir y configurar la infraestructura virtual necesaria. Esta herramienta permite la creación dinámica y la gestión de recursos en la nube de manera reproducible y escalable. A través de archivos de configuración específicos, se generan tanto el Scale Set como las redes necesarias para asegurar su funcionamiento adecuado. Además, se implementa Azure Monitor y Azure Functions para monitorizar el uso de los recursos y enviar alertas cuando se superan o se quedan por debajo de ciertos umbrales predefinidos.

El repositorio de código, alojado en GitHub, juega un papel central en el despliegue automatizado de la infraestructura mediante GitHub Actions. Esta integración continua permite gestionar y controlar de forma eficiente el ciclo de vida de la infraestructura, garantizando una operación continua y minimizando el riesgo de errores manuales.

En este capítulo de la memoria del Trabajo de Fin de Máster, se detallará el diseño de la solución propuesta, su implementación técnica utilizando herramientas como Terraform y Azure, así como su funcionamiento en términos de escalabilidad y gestión financiera. Además, se discutirán los beneficios esperados de esta solución, tanto en términos de eficiencia operativa como de optimización de costos.

4.1 Arquitectura de la solución

La arquitectura de la solución desarrollada para gestionar el escalado y desescalado de máquinas virtuales dentro de un Scale Set de Azure se fundamenta en varios componentes clave que interactúan de manera coordinada para garantizar un funcionamiento eficiente y automatizado. A continuación, se detallan los principales elementos y su interacción dentro del contexto del proyecto.

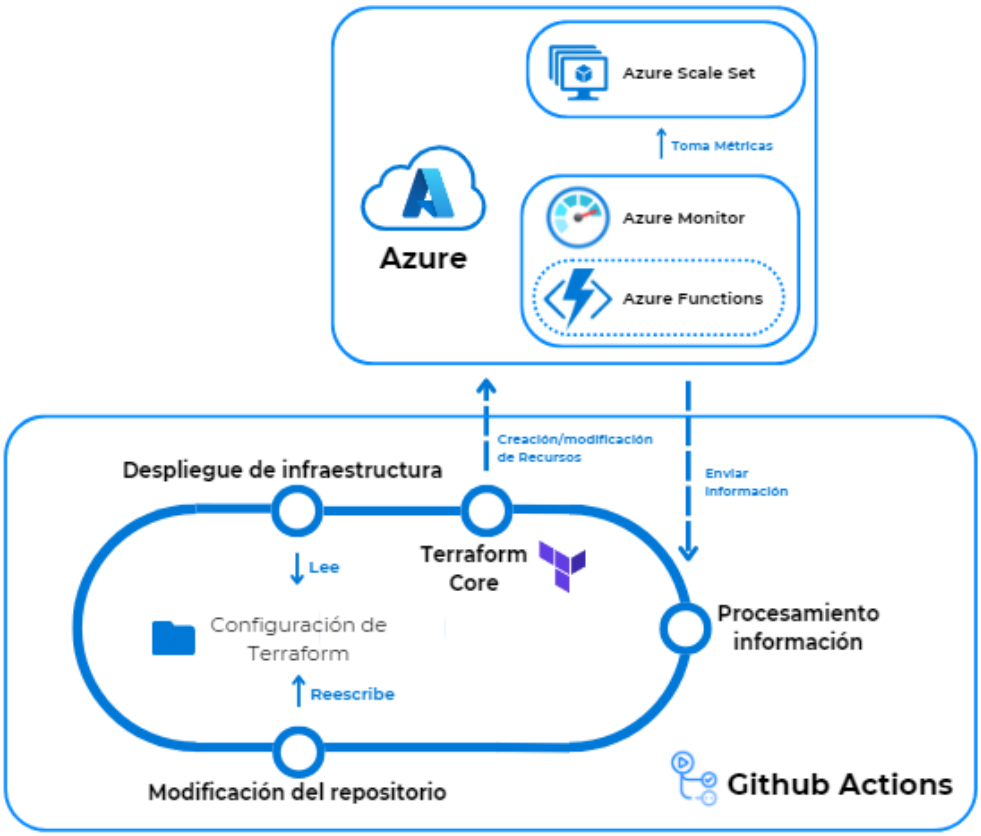


Figure 4.1: Arquitectura general de la solución.

En la figura 4.1, se muestra el flujo de trabajo automatizado para la gestión de infraestructura en Azure utilizando Terraform y GitHub Actions. A continuación se explica cada componente y su interacción:

- **Terraform:** se utiliza para definir y desplegar la infraestructura necesaria de manera consistente y reproducible. Mediante archivos de configuración en formato HCL (HashiCorp Configuration Language), se especifican los recursos de Azure que componen el entorno del Scale Set y las redes asociadas.

Esto incluye la configuración de redes virtuales (VNet), subredes, reglas de seguridad, y el propio Scale Set con las configuraciones de máquinas virtuales.

- **Azure:**
  - Azure Scale Set: servicio de Azure que permite administrar y escalar un conjunto de máquinas virtuales (VMs) idénticas.
  - Azure Monitor: se utiliza para monitorear el uso de recursos dentro del Scale Set. Se configuran alertas personalizadas que se activan cuando se alcanzan umbrales críticos de utilización de CPU, memoria o cualquier métrica relevante para el rendimiento de las aplicaciones. Estas alertas permiten responder ante el estado del scale set.
  - Azure Functions: se emplea para implementar funciones serverless que responden a eventos específicos dentro del entorno del Scale Set. Se han desarrollado funciones encargadas de enviar información a github en función de las alertas generadas por Azure Monitor. Esto permite una gestión automatizada de la infraestructura en tiempo real.
- **GitHub Actions:**
  - Despliegue de infraestructura: flujo de trabajo encargado ejecutar terraform con la configuración incluida dentro del repositorio de github y deplegar los recursos especificados en ella.
  - Procesamiento de información: Recibe información del conjunto de escalado de Azure y define la configuración a implementar.
  - Modificación del repositorio: se modifica el repositorio de github con los parámetros definidos del paso anterior.

4.2 Componentes en Azure

En la implementación de la solución para gestionar el escalado y desescalado de máquinas virtuales dentro de un conjunto de escalado en Azure, se utilizan varios componentes y servicios de Azure. El código de Terraform se puede desglosar en cuatro archivos, cada uno con una función concreta: 'main.tf', 'compute.tf', 'network.tf' y 'monitor.tf'. A continuación, se detallan estos componentes y su rol en la arquitectura de la solución.

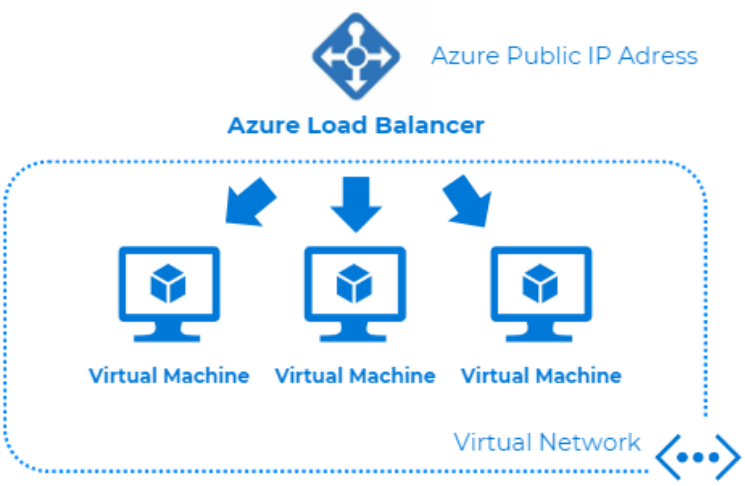


Figure 4.2: Arquitectura del conjunto de escalado.

4.2.1 Archivo principal (main.tf)

El archivo **'main.tf'** es el archivo principal en una configuración de Terraform. Se utiliza para definir los recursos que Terraform va a gestionar en un proveedor de nube, en este caso, Azure. En **'main.tf'**, se especifican los diferentes componentes que se desean desplegar y configurar en Azure. A continuación, se describe para qué se emplea cada sección del archivo main.tf desarrollado:

4.2.1.1 Backend para Terraform

Se especifica el uso de azurerm como backend, lo que permite almacenar el estado de Terraform en una cuenta de almacenamiento de Azure. Este backend es crucial para gestionar el estado de la infraestructura, ya que, al ejecutarse este código dentro de un runner de azure, si no se especifica un backend se perderá el archivo encargado de controlar el estado de los recursos.

Un ejemplo de configuración de un backend de azure se presenta en el Listing 4.1. Se especifica los siguientes parámetros:

- resource_group_name: Define el grupo de recursos en el que se almacenará el estado de Terraform.
- storage_account_name: Especifica la cuenta de almacenamiento de Azure.
- container_name: El contenedor en la cuenta de almacenamiento donde se guardará el estado.

- key: El nombre del archivo de estado.

```
backend "azurerm" {
  resource_group_name = "tfmtfstates"
  storage_account_name = "tfmtf"
  container_name     = "tfstatetfm"
  key                = "terraform.tfstate"
}
```

Listing 4.1: Configuración del backend de azurerm

4.2.1.2 Providers de Terraform

En Terraform, un "provider" (proveedor) es un componente que se encarga de interactuar con APIs externas para gestionar y provisionar recursos. Los proveedores permiten a Terraform gestionar una amplia variedad de servicios y plataformas de nube, infraestructura y aplicaciones. Cada proveedor se configura con una o más credenciales de autenticación y puede gestionar diferentes tipos de recursos según las APIs que soporte.

Los providers requeridos para este proyecto son los siguientes:

- azurerm: Proveedor principal para gestionar recursos de Azure, con una versión específica (3.51.0) para asegurar compatibilidad.
- random: Utilizado para generar datos aleatorios, como contraseñas o identificadores únicos.
- azapi: Proveedor para acceder a las API de Azure.

Un ejemplo de código para su definición se presenta en el Listing 4.2.

```
required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "3.51.0"
    }
    random = {
      source = "hashicorp/random"
      version = "3.5.1"
    }
    azapi = {
      source = "Azure/azapi"
    }
}
```

Listing 4.2: Providers de terraform

4.2.1.3 Creación del Grupo de Recursos

Un grupo de recursos es un contenedor que almacena los recursos relacionados con una solución de Azure. El grupo de recursos puede incluir todos los recursos de la solución o solo aquellos que se desean administrar como grupo.[65]

Un ejemplo de código para su definición se presenta en el Listing 4.3.

Contiene los siguiente componentes:

- name: Nombre del grupo de recursos, en este caso "terraform-tfm".
- location: Región de Azure donde se ubicará el grupo de recursos, en este caso "westus3".

```
resource "azurerm_resource_group" "rg" {
  name     = "terraform-tfm"
  location = "westus3"
}
```

Listing 4.3: Creación del grupo de recursos

4.2.2 Archivo de definición de recursos de cómputo (compute.tf)

El archivo compute.tf configura un Virtual Machine Scale Set en Azure [66], definiendo aspectos críticos como el tipo de máquina virtual, la autenticación, la imagen del sistema operativo, el disco del sistema, y la interfaz de red. Además, incluye configuraciones adicionales para diagnósticos de inicio y gestión del ciclo de vida del recurso, asegurando una implementación eficiente y administrable del VMSS. Esta configuración permite el escalado y desescalado automatizado de instancias según las necesidades del proyecto, garantizando rendimiento y disponibilidad.

A continuación, se explica cada sección del código incluido en este archivo.

4.2.2.1 Configuración del Virtual Machine Scale Set

Se definen distintos parámetros para la configuración de las máquinas virtuales que componen el scale set. Estos parámetros son:

- name: Nombre del scale set, en este caso "vmss-terraform".
- resource_group_name: Nombre del grupo de recursos, utilizando el definido en main.tf.
- location: Ubicación del grupo de recursos, también reutilizado de main.tf.
- sku_name: Especifica el tipo de máquina virtual que se utilizará (Standard_D2s_v4).
- instances: Número inicial de instancias del scale set (2 en este caso).
- platform_fault_domain_count: Especifica el número de dominios de fallo que utiliza este conjunto de escalado de máquina virtual. Cambiar esto obliga a crear un nuevo recurso.
- zones: Zonas de disponibilidad en Azure, especificadas como ["1"].

Un ejemplo de código para su definición se presenta en el Listing 4.4.

```
resource "azurerm_orchestrated_virtual_machine_scale_set"
    "vmss_terraform_tfm" {
  name                        = "vmss-terraform"
  resource_group_name         = azurerm_resource_group.rg.name
  location                    = azurerm_resource_group.rg.location
  sku_name                    = "Standard_D2s_v4"
  instances                   = 2
  platform_fault_domain_count = 1
  zones                       = ["1"]
```

Listing 4.4: Configuración del Virtual Machine Scale Set

4.2.2.2 Configuración de Usuario y Autenticación

Define un script de inicio codificado en base64 y la configuración del sistema operativo, incluyendo la desactivación de la autenticación por contraseña y el uso de claves SSH.

Contiene los siguientes parámetros:

- user_data_base64: Especifica un script de inicio codificado en base64.
- os_profile: Configuración del sistema operativo.
  - linux_configuration: Configuración específica para Linux.
    - disable_password_authentication: Desactiva la autenticación por contraseña.

    - admin_username: Nombre del usuario administrador.
    - admin_ssh_key: Clave SSH pública para autenticación.
        - username: Nombre del usuario para la clave SSH.
        - public_key: Clave pública. Este parámetro se definide como una variable (var.PUB_KEY).

Un ejemplo de código para su definición se presenta en el Listing 4.5.

```
user_data_base64 = base64encode(file("user-data.sh"))
os_profile {
  linux_configuration {
    disable_password_authentication = true
    admin_username = "azureuser"
    admin_ssh_key {
      username  = "azureuser"
      public_key = var.PUB_KEY
    }
  }
}
```

Listing 4.5: Configuración de Usuario y Autenticación

4.2.2.3 Imagen del Sistema Operativo

Se definen los siguientes parámetros:

- publisher: Especifica el proveedor de la imagen utilizada para crear las máquinas virtuales.
- offer: Nombre de un grupo de imágenes relacionadas.
- sku: Especifica el SKU de la imagen usado para crear las máquinas virtuales (22_04-LTS-gen2).
- version: Versión de la imagen (latest).

Un ejemplo de código para su definición se presenta en el Listing 4.6.

```
source_image_reference {
  publisher = "Canonical"
  offer     = "0001-com-ubuntu-server-jammy"
  sku       = "22_04-LTS-gen2"
  version   = "latest"
}
```

Listing 4.6: Configuración de la Imagen del Sistema Operativo

4.2.2.4 Configuración del Disco del Sistema

Se definen los siguientes parámetros:

- storage_account_type:El tipo de cuenta de almacenamiento que debe respaldar el disco interno del sistema operativo. Los valores posibles incluyen Standard_LRS, StandardSSD_LRS, StandardSSD_ZRS, Premium_LRS y Premium_ZRS.
- caching: Tipo de almacenamiento en caché que debe utilizarse para el disco interno del sistema operativo. Los valores posibles son None, ReadOnly y ReadWrite.

Un ejemplo de código para su definición se presenta en el Listing 4.7.

```
os_disk {
  storage_account_type = "Premium_LRS"
  caching              = "ReadWrite"
}
```

Listing 4.7: Configuración del Disco del Sistema

4.2.2.5 Configuración del Interfaz de Red

Se definen los siguientes parámetros:

- name: Nombre que debe utilizarse para esta interfaz de red.
- primary: Define si ésta configuración de la IP es primaria. Los valores posibles son true y false.
- enable_accelerated_networking: Define si esta interfaz de red admite Accelerated Networking. Los valores posibles son true y false.
- ip_configuration: Configuración de IP.
  - name: El nombre que debe utilizarse para esta configuración IP.
  - primary: Define si ésta configuración de la IP es primaria. Los valores posibles son true y false.
  - subnet_id: ID de la subred a la que debe conectarse esta configuración IP.

– load_balancer_backend_address_pool_ids: Una lista de ID de conjuntos de direcciones de backend de un equilibrador de carga al que debe conectarse este scale set.

Un ejemplo de código para su definición se presenta en el Listing 4.8.

```
network_interface {
  name                          = "nic"
  primary                       = true
  enable_accelerated_networking = false

  ip_configuration {
    name                                    = "ipconfig"
    primary                                 = true
    subnet_id                               = azurerm_subnet.subnet.id
    load_balancer_backend_address_pool_ids =
        [azurerm_lb_backend_address_pool.bepool.id]
  }
}
```

Listing 4.8: Configuración del Interfaz de Red

4.2.3 Archivo de definición de recursos de red (network.tf)

El archivo **"network.tf"** configura los recursos de red necesarios para el despliegue de una infraestructura en Azure, incluyendo la creación de una red virtual y subred, un grupo de seguridad de red, direcciones IP públicas, un balanceador de carga, reglas NAT y un gateway NAT para habilitar tráfico de salida. Cada recurso y parámetro está diseñado para asegurar una conectividad y seguridad óptima dentro de la infraestructura en la nube.

4.2.3.1 Creación de la Red Virtual de Azure

Gestiona una red virtual incluyendo cualquier subred configurada. Cada subred puede configurarse opcionalmente con un grupo de seguridad que se asociará a la subred.

Se definen los siguientes parámetros [67]:

- name: Nombre de la red virtual.
- address_space: Espacio de direcciones IP para la red.
- location: Localización donde esta red virtual es creada.

- resource_group_name: Nombre del grupo de recursos donde se crea esta red virtual.

Un ejemplo de código para su definición se presenta en el Listing 4.9.

```
resource "azurerm_virtual_network" "test" {
  name                = "terraformvnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}
```

Listing 4.9: Creación de la Red Virtual de Azure

4.2.3.2 Creación de la Subred de Azure

Gestiona una subred. Las subredes representan segmentos de red dentro del espacio IP definido por la red virtual.

Se definen los siguientes parámetros [68]:

- name: Nombre de la subred.
- resource_group_name: Nombre del grupo de recursos donde se crea esta subred.
- virtual_network_name: El nombre de la red virtual a la que adjuntar la subred.
- address_prefixes: Los prefijos de dirección a utilizar para la subred.

Un ejemplo de código para su definición se presenta en el Listing 4.10.

```
resource "azurerm_subnet" "subnet" {
  name                 = "subnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.test.name
  address_prefixes     = ["10.0.0.0/20"]
}
```

Listing 4.10: Creación de la Subred de Azure

4.2.3.3 Creación de Grupo de Seguridad de Red

Gestiona un grupo de seguridad de red que contiene una lista de reglas de seguridad de red. Los grupos de seguridad de red permiten habilitar o denegar el tráfico

entrante o saliente.

Se definen los siguientes parámetros [69]:

- name: Nombre del NSG.
- location: Localización donde este NSG es creada.
- resource_group_name: Nombre del grupo de recursos donde se crea esta subred.
- security_rule: Reglas de seguridad para permitir tráfico HTTP, HTTPS y SSH.
  - name: Nombre de la regla.
  - priority: Especifica la prioridad de la regla. El valor puede estar comprendido entre 100 y 4096. El número de prioridad debe ser único para cada regla de la colección. Cuanto menor sea el número de prioridad, mayor será la prioridad de la regla.
  - direction: La dirección especifica si la regla se evaluará en el tráfico entrante o saliente. Los valores posibles son Entrante y Saliente.
  - access: Especifica si el tráfico de red está permitido o denegado. Los valores posibles son Permitir y Denegar.
  - protocol: Protocolo de red al que se aplica esta regla. Los valores posibles incluyen Tcp, Udp, Icmp, Esp, Ah o * (que coincide con todos).
  - source_port_range: Puerto de origen o rango. Entero o rango entre 0 y 65535 o * para que coincida con cualquiera. Es obligatorio si no se especifica source_port_ranges.
  - destination_port_range: Puerto o rango de destino. Número entero o rango entre 0 y 65535 o * para que coincida con cualquiera. Es obligatorio si no se especifica destination_port_ranges.
  - source_address_prefix: CIDR o rango de IP de origen o * para que coincida con cualquier IP. También se pueden utilizar etiquetas como VirtualNetwork, AzureLoadBalancer e Internet. Esto es necesario si source_address_prefixes no se especifica.
  - destination_address_prefix: CIDR o rango de IP de destino o * para que coincida con cualquier IP. También se pueden utilizar etiquetas como VirtualNetwork, AzureLoadBalancer e Internet. Esto es necesario si no se especifica destination_address_prefixes.

Un ejemplo de código para su definición se presenta en el Listing 4.11.

```
resource "azurerm_network_security_group" "myNSG" {
  name                = "myNSG"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  security_rule {
    name                       = "allow-http"
    priority                   = 100
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "80"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }

  security_rule {
    name                       = "allow-https"
    priority                   = 101
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "443"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }

  security_rule {
    name                       = "allow-ssh"
    priority                   = 102
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "22"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }
}
```

Listing 4.11: Creación de Grupo de Seguridad de Red

4.2.3.4 Creación de Dirección IP Pública

Gestiona una dirección IP pública.

Se definen los siguientes parámetros [70]:

- name: Nombre de la subred.
- location: Especifica la ubicación de Azure admitida en la que debe existir la IP pública.
- resource_group_name: El nombre del Grupo de Recursos donde debe existir esta IP Pública.
- allocation_method: Define el método de asignación para esta dirección IP. Los valores posibles son Estática o Dinámica.
- sku: El SKU de la IP pública. Los valores aceptados son Básico y Estándar. Por defecto es Básica.
- zones: Una colección que contiene la zona de disponibilidad en la que asignar la IP pública.
- domain_name_label: Etiqueta para el nombre de dominio. Se utilizará para formar el FQDN. Si se especifica una etiqueta para el nombre de dominio, se creará un registro DNS A para la IP pública en el sistema DNS de Microsoft Azure.

Un ejemplo de código para su definición se presenta en el Listing 4.12.

```
resource "azurerm_public_ip" "example" {
  name                = "lb-publicIP"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method = "Static"
  sku                 = "Standard"
  zones               = ["1", "2", "3"]
  domain_name_label =
      "${azurerm_resource_group.rg.name}-${random_pet.lb_hostname.id}"
}
```

Listing 4.12: Creación de Dirección IP Pública

4.2.3.5 Creación del Balanceador de Carga

Load Balancer distribuye flujos de entrada que llegan al front-end del equilibrador de carga a las instancias del grupo de back-end. Estos flujos están de acuerdo con las reglas de equilibrio de carga y los sondeos de estado configurados. Las instancias del grupo de back-end pueden ser instancias de Azure Virtual Machines o de un conjunto de escalado de máquinas virtuales.[71]

Se definen los siguientes parámetros [72]:

- name: Especifica el nombre del balanceador de carga.
- location: Especifica la región de Azure admitida en la que debe crearse el balanceador de carga.
- resource_group_name: Nombre del grupo de recursos en el que se va a crear el balanceador de carga.
- sku: El SKU del Azure Load Balancer. Los valores aceptados son Basic, Standard y Gateway. El valor predeterminado es Basic.
- frontend_ip_configuration: Configura la IP del Front-end
  - name: Nombre de la configuración.
  - public_ip_address_id: ID de una dirección IP pública asociada a este balanceador de carga.

Un ejemplo de código para su definición se presenta en el Listing 4.13.

```
resource "azurerm_lb" "example" {
  name                = "myLB"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  sku                 = "Standard"
  frontend_ip_configuration {
    name                 = "myPublicIP"
    public_ip_address_id = azurerm_public_ip.example.id
  }
}
```

Listing 4.13: Creación del Balanceador de Carga

Además, es necesario crear un pool de direcciones backend para el balanceador.[73] Un ejemplo de código para su definición se presenta en el Listing 4.14.

```
resource "azurerm_lb_backend_address_pool" "bepool" {
```

```
  name           = "myBackendAddressPool"
  loadbalancer_id = azurerm_lb.example.id
}
```

Listing 4.14: Creación del pool de direcciones backend para el balanceador

4.2.3.6 Reglas del Balanceador de Carga

Gestiona una regla del equilibrador de carga.

Se definen los siguientes parámetros [74]:

- name: Nombre de la regla.
- loadbalancer_id: El ID del balanceador de carga en el que crear la regla.
- protocol: El protocolo de transporte para el punto final externo. Los valores posibles son Tcp, Udp o All.
- frontend_port: El puerto para el punto final externo. Los números de puerto de cada regla deben ser únicos dentro del equilibrador de carga. Los valores posibles oscilan entre 0 y 65534, ambos inclusive. Un puerto de 0 significa "Cualquier puerto".
- backend_port: El puerto utilizado para conexiones internas en el endpoint. Los valores posibles oscilan entre 0 y 65535, ambos inclusive. Un puerto de 0 significa "Cualquier puerto".
- frontend_ip_configuration_name: El nombre de la configuración IP del frontend a la que está asociada la regla.
- backend_address_pool_ids: Una lista de referencia a un Grupo de Direcciones Backend sobre el que opera esta Regla de Balanceo de Carga.
- probe_id: Referencia a l Probe utilizado por esta regla del balanceador de carga.

Un ejemplo de código para su definición se presenta en el Listing 4.15.

```
resource "azurerm_lb_rule" "example" {
  name                           = "http"
  loadbalancer_id                = azurerm_lb.example.id
  protocol                       = "Tcp"
  frontend_port                  = 80
  backend_port                   = 80
  frontend_ip_configuration_name = "myPublicIP"
```

```
  backend_address_pool_ids    =
      [azurerm_lb_backend_address_pool.bepool.id]
  probe_id                    = azurerm_lb_probe.example.id
}
```

Listing 4.15: Creación de las Reglas del Balanceador de Carga

4.2.3.7 Probe del Balanceador de Carga

Crea un probe para verificar el estado de las instancias backend.

Se definen los siguientes parámetros [75]:

- name: Nombre del probe.
- loadbalancer_id: El ID del LoadBalancer en el que crear el Probe
- protocol: Especifica el protocolo del punto final. Los valores posibles son Http, Https o Tcp. Si se especifica TCP, se requiere un ACK recibido para que la sonda tenga éxito. Si se especifica HTTP, se requiere una respuesta 200 OK desde el URI especificado para que el Probe tenga éxito. Por defecto es Tcp.
- port: Puerto en el que el Probe consulta el endpoint del backend. Los valores posibles van de 1 a 65535, ambos incluidos.
- request_path: El URI utilizado para solicitar el estado de salud desde el punto final del backend. Obligatorio si el protocolo es Http o Https. De lo contrario, no se permite.

Un ejemplo de código para su definición se presenta en el Listing 4.16.

```
resource "azurerm_lb_probe" "example" {
  name           = "http-probe"
  loadbalancer_id = azurerm_lb.example.id
  protocol       = "Http"
  port           = 80
  request_path   = "/"
}
```

Listing 4.16: Creación del Probe del Balanceador de Carga

4.2.3.8 Reglas NAT del Balanceador de Carga

Una regla NAT de entrada se usa para reenviar el tráfico desde un front-end del equilibrador de carga a una o varias instancias del grupo de back-end.[76]

Se definen los siguientes parámetros [77]:

- name: Nombre de la regla.
- resource_group_name: El nombre del grupo de recursos en el que crear el recurso.
- loadbalancer_id: El ID del balanceador de carga en el que crear la regla NAT.
- protocol: El protocolo de transporte para el punto final externo. Los valores posibles son Udp, Tcp o All.
- frontend_port_start: El inicio del rango de puertos para el extremo externo. Esta propiedad se utiliza junto con BackendAddressPool y FrontendPortRangeEnd. Se crearán asignaciones de puertos de reglas NAT entrantes individuales para cada dirección backend de BackendAddressPool. Los valores posibles van de 1 a 65534, ambos incluidos.
- frontend_port_end: El final del rango de puertos para el extremo externo. Esta propiedad se utiliza junto con BackendAddressPool y FrontendPortRangeStart. Se crearán asignaciones de puertos de reglas NAT entrantes individuales para cada dirección backend de BackendAddressPool. Los valores posibles van de 1 a 65534, ambos incluidos.
- backend_port: El puerto utilizado para conexiones internas en el endpoint. Los valores posibles oscilan entre 1 y 65535, ambos incluidos.
- frontend_ip_configuration_name: El nombre de la configuración IP del frontend que expone esta regla.
- backend_address_pool_id: Especifica una referencia al recurso backendAddressPool.

Un ejemplo de código para su definición se presenta en el Listing 4.17.

```
resource "azurerm_lb_nat_rule" "ssh" {
  name                           = "ssh"
  resource_group_name            = azurerm_resource_group.rg.name
  loadbalancer_id                = azurerm_lb.example.id
  protocol                       = "Tcp"
  frontend_port_start            = 50000
```

```
  frontend_port_end           = 50119
  backend_port                = 22
  frontend_ip_configuration_name = "myPublicIP"
  backend_address_pool_id     = azurerm_lb_backend_address_pool.bepool.id
}
```

Listing 4.17: Creación de las Reglas NAT del Balanceador de Carga

4.2.3.9 NAT Gateway y Asociación de Red y Subred

En primer lugar se crea una IP pública para el NAT gateway.

Se definen los siguientes parámetros [70]:

- name: Nombre de la subred.
- location: Especifica la ubicación de Azure admitida en la que debe existir la IP pública.
- resource_group_name: El nombre del Grupo de Recursos donde debe existir esta IP Pública.
- allocation_method: Define el método de asignación para esta dirección IP. Los valores posibles son Estática o Dinámica.
- sku: El SKU de la IP pública. Los valores aceptados son Básico y Estándar. Por defecto es Básica.
- zones: Una colección que contiene la zona de disponibilidad en la que asignar la IP pública.

Un ejemplo de código para su definición se presenta en el Listing 4.18.

```
resource "azurerm_public_ip" "natgwpip" {
  name                = "natgw-publicIP"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method = "Static"
  sku                 = "Standard"
  zones               = ["1"]
}
```

Listing 4.18: Creación de una IP Pública para NAT Gateway

Crea un NAT gateway para habilitar tráfico de salida. Azure NAT Gateway es un servicio de traducción de direcciones de red (NAT) totalmente administrado y

altamente resistente. Puede usar Azure NAT Gateway para permitir que todas las instancias de una subred privada establezcan conexiones saliente a Internet a la vez que permanecen totalmente privadas.[78]

Se definen los siguientes parámetros [79]:

- name: Nombre del NAT gateway.
- location: Especifica la ubicación de Azure admitida en la que debe existir el NAT Gateway.
- resource_group_name: El nombre del Grupo de Recursos donde debe existir este NAT Gateway.
- sku_name: El SKU que debe utilizarse. En este momento, el único valor admitido es Estándar.
- idle_timeout_in_minutes: El tiempo de espera de inactividad que debe utilizarse en minutos.
- zones: Lista de zonas de disponibilidad en las que se debe ubicar esta puerta de enlace NAT.

Un ejemplo de código para su definición se presenta en el Listing 4.19.

```
resource "azurerm_nat_gateway" "example" {
  name                    = "nat-Gateway"
  location                = azurerm_resource_group.rg.location
  resource_group_name     = azurerm_resource_group.rg.name
  sku_name                = "Standard"
  idle_timeout_in_minutes = 10
  zones                   = ["1"]
}
```

Listing 4.19: Creación del NAT Gateway

Finalmente, se asocia el gateway NAT con la red y la subred.

Se definen los siguientes parámetros para la asociación de la subred[80]:

- subnet_id: ID de la subred.
- nat_gateway_id: ID del NAT gateway.

Y los siguientes parámetros para la asociación de la red[81]:

- public_ip_address_id: ID de la IP pública.
- nat_gateway_id: ID del gateway NAT.

Un ejemplo de código para su definición se presenta en el Listing 4.20.

```
resource "azurerm_subnet_nat_gateway_association" "example" {
  subnet_id      = azurerm_subnet.subnet.id
  nat_gateway_id = azurerm_nat_gateway.example.id
}
resource "azurerm_nat_gateway_public_ip_association" "example" {
  public_ip_address_id = azurerm_public_ip.natgwpip.id
  nat_gateway_id       = azurerm_nat_gateway.example.id
}
```

Listing 4.20: Asociación del gateway NAT con la red y la subred

4.2.4 Archivo de definición de recursos para monitorización (monitor.tf)

El archivo **"monitor.tf"** configura los recursos necesarios para monitorizar la carga de trabajo del Scale Set y, según las alertas, aumentar o reducir el número de máquinas virtuales mediante acciones automatizadas y el envío de alertas a GitHub. Esto incluye la creación de workflows de Logic Apps, triggers y acciones HTTP, alertas de métrica de Azure Monitor y grupos de acción para gestionar las alertas y sus respuestas automáticas. A continuación, se detalla cada recurso y los parámetros que es necesario definir por cada uno de los parámetros que se quiera monitorizar.

4.2.4.1 Creación de un Logic App Workflow

Azure Logic Apps es una plataforma en la nube que permite crear y ejecutar flujos de trabajo automatizados sin apenas código. Mediante la selección de operaciones predefinidas, se pueden crear rápidamente un flujo de trabajo que integre y gestione aplicaciones, datos, servicios y sistemas.[82]

Se definen los siguientes parámetros [83]:

- name: Nombre del workflow.
- location: Ubicación del recurso.
- resource_group_name: Nombre del grupo de recursos.

Un ejemplo de código para su definición se presenta en el Listing 4.21.

```
resource "azurerm_logic_app_workflow" "reduce_vm" {
  name                = "workflow_reduce_vm"
```

```
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}
```

Listing 4.21: Creación de un Logic App Workflow

4.2.4.2 Creación de un Trigger HTTP de la Logic App

A continuación, se define un trigger HTTP para la Logic App.

Se definen los siguientes parámetros [84]:

- name: Nombre del trigger.
- logic_app_id: Especifica el ID del flujo de trabajo de la logic app.
- schema: Un Blob JSON que define el esquema de la solicitud entrante.

Un ejemplo de código para su definición se presenta en el Listing 4.22.

```
resource "azurerm_logic_app_trigger_http_request" "reduce_vm" {
  name         = "http-trigger-reduce-vm"
  logic_app_id = azurerm_logic_app_workflow.reduce_vm.id

  schema = <<SCHEMA
  {
    "type": "object",
    ...
  }
  SCHEMA
}
```

Listing 4.22: Creación de un Trigger HTTP de la Logic App

4.2.4.3 Creación de una Acción HTTP de la Logic App

Se define una acción HTTP para la Logic App.

Se definen los siguientes parámetros [85]:

- name: Nombre de la acción.
- logic_app_id: ID del workflow de Logic App asociado.
- method: Método HTTP (POST).
- body: Cuerpo de la request en formato JSON.

- headers: Encabezados HTTP, incluyendo la autenticación con un token de GitHub.
- uri: URL del webhook de GitHub.

Un ejemplo de código para su definición se presenta en el Listing 4.23.

```
resource "azurerm_logic_app_action_http" "reduce_vm" {
  name         = "webhook"
  logic_app_id = azurerm_logic_app_workflow.reduce_vm.id
  method       = "POST"
  body = jsonencode({
    event_type = "reduce-n-vms"
  })
  headers = {
    "Content-Type" = "application/json"
    "Accept" : "application/vnd.github+json"
    "Authorization" : "Bearer ${var.GH_TOKEN}"
    "X-GitHub-Api-Version" : "2022-11-28"
  }
  uri =
    "https://api.github.com/repos/aruizcab/FinOps_AutoScaling/dispatches"
}
```

Listing 4.23: Creación de una Acción HTTP de la Logic App

4.2.4.4 Creación de una Alerta de Métrica de Azure Monitor

Gestiona una alerta de métrica en Azure Monitor.

Se definen los siguientes parámetros [86]:

- name: Nombre de la alerta.
- resource_group_name: Nombre del grupo de recursos.
- scopes: Conjunto de cadenas de ID de recursos a los que deben aplicarse los criterios de métrica.
- description: La descripción de esta alerta métrica.
- severity: La gravedad de esta alerta métrica. Los valores posibles son 0, 1, 2, 3 y 4. El valor predeterminado es 3.
- enabled: Define si esta alerta está habilitada

- frequency: La frecuencia de evaluación de esta alerta métrica, representada en formato de duración ISO 8601. Los valores posibles son PT1M, PT5M, PT15M, PT30M y PT1H. Por defecto es PT1M.
- window_size: El periodo de tiempo que se utiliza para supervisar la actividad de alerta, representado en formato de duración ISO 8601. Este valor debe ser mayor que la frecuencia. Los valores posibles son PT1M, PT5M, PT15M, PT30M, PT1H, PT6H, PT12H y P1D. Por defecto es PT5M.
- criteria: Criterios de la alerta.
  - metric_namespace: Uno de los metric namespaces que se van a supervisar.
  - metric_name: Uno de los nombres de las métricas a supervisar.
  - aggregation: La estadística que recorre los valores de la métrica. Los valores posibles son Media, Recuento, Mínimo, Máximo y Total.
  - operator: El operador de criterios. Los valores posibles son Igual, Mayor que, Mayor que o igual, Menor que y Menor que o igual.
  - threshold: El valor del umbral de criterios que activa la alerta.
- action: Acción a ejecutar cuando se dispara la alerta.
  - action_group_id: El ID del grupo de acciones puede obtenerse del recurso azurerm_monitor_action_group.

Un ejemplo de código para su definición se presenta en el Listing 4.24.

```
resource "azurerm_monitor_metric_alert" "low_cpu_alert" {
  name                = "vmss-cpu-alert"
  resource_group_name = azurerm_resource_group.rg.name
  scopes              =
    azurerm_orchestrated_virtual_machine_scale_set.vmss_terraform_tfm.id]
  description         = "Alert when VMSS CPU usage is below 10%"
  severity            = 2
  enabled             = true
  frequency           = "PT5M"
  window_size         = "PT5M"

  criteria {
    metric_namespace = "Microsoft.Compute/virtualMachineScalesets"
    metric_name      = "Percentage CPU"
    aggregation      = "Average"
    operator         = "LessThan"
```

```
    threshold       = 10
  }

  action {
    action_group_id = azurerm_monitor_action_group.reduce_vm.id
  }
}
```

Listing 4.24: Creación de una Alerta de Métrica de Azure Monitor

4.2.4.5 Creación de un Grupo de Acción de Azure Monitor

Gestiona un grupo de acciones dentro de Azure Monitor.

Se definen los siguientes parámetros [87]:

- name: Nombre del grupo de acción.
- resource_group_name: Nombre del grupo de recursos.
- short_name: Nombre reducido del grupo de acción.
- logic_app_receiver: Receptor de la Logic App.
  - name: Nombre del receptor.
  - resource_id: El ID de recurso Azure de la aplicación lógica.
  - callback_url: La url de callback a la que se envía la petición HTTP.
  - use_common_alert_schema: Activa o desactiva el esquema común de alertas.

Un ejemplo de código para su definición se presenta en el Listing 4.25.

```
resource "azurerm_monitor_action_group" "reduce_vm" {
  name                = "vmss-action-group"
  resource_group_name = azurerm_resource_group.rg.name
  short_name          = "vmss-ag"

  logic_app_receiver {
    name                    =
        azurerm_logic_app_trigger_http_request.reduce_vm.name
    resource_id             =
        azurerm_logic_app_trigger_http_request.reduce_vm.id
    callback_url            =
        azurerm_logic_app_trigger_http_request.reduce_vm.callback_url
```

```
    use_common_alert_schema = true
  }
}
```

Listing 4.25: Creación de un Grupo de Acción de Azure Monitor

4.3 Parámetros de optimización

4.3.1 Variables que influyen en el costo del scale set

Las variables que se pueden modificar en un Virtual Machine Scale Set (VMSS) de Azure y que influyen en el costo del recurso según la calculadora de precios de Azure incluyen:

- Número de Instancias: Cantidad de VMs en el scale set.
- Tamaño de la Máquina Virtual (VM): El tipo y tamaño de la VM seleccionada afecta directamente el costo. Azure ofrece diferentes tamaños de VM que varían en términos de CPU, memoria y almacenamiento temporal.
- Región: La ubicación geográfica donde se despliega el VMSS puede influir en el costo debido a las variaciones en los precios regionales.
- Tipo de Almacenamiento: La elección entre almacenamiento estándar y premium, así como la cantidad de discos administrados y su tamaño, impacta en el costo total.
- Horas de Uso: El número de horas que las VMs están activas en un mes. Azure define un mes como 730 horas, y se puede configurar la disponibilidad mensual de las VMs.
- Opciones de Pago: Azure ofrece diferentes opciones de pago, como "Pay-as-you-go" y "Compute Pre-Purchase" (CPP), que pueden ofrecer descuentos significativos.
- Prioridad de la VM: Usar Azure Spot Virtual Machines, que son instancias de baja prioridad y pueden ser desalojadas cuando Azure necesita recuperar capacidad, puede reducir costos significativamente.
- Política de Expulsión: La política de expulsión (por ejemplo, "Deallocate" o "Delete") para las Spot VMs también puede influir en los costos, ya que afecta cómo se manejan las VMs desalojadas.
- Configuración de Red: La configuración de red, incluyendo el uso de redes aceleradas y la cantidad de ancho de banda de red, puede afectar los costos.

4.3.2 Parámetros Financieros y Operativos de la Empresa

Para definir los parámetros de la empresa que influirán en las características de un Virtual Machine Scale Set (VMSS) de Azure, es esencial considerar varios aspectos financieros y operativos que se derivan de la planificación y gestión del proyecto. Estos parámetros deben alinearse con los objetivos estratégicos y operativos de la empresa. A continuación, se detallan los parámetros clave que se deben extraer de la empresa:

- **Presupuesto Total del Proyecto:** El presupuesto total asignado al proyecto es fundamental. Este debe incluir todos los costos previstos, como infraestructura, licencias, personal, y contingencias. Un presupuesto bien definido permite establecer límites claros para la inversión en recursos de TI, incluyendo el VMSS.

- **Horizonte Temporal del Proyecto:** La duración del proyecto influye en la planificación de los recursos. Es necesario definir si el proyecto es a corto, mediano o largo plazo, ya que esto afectará la estrategia de escalabilidad y el tipo de instancias a utilizar en el VMSS.

- **Proyecciones de Carga de Trabajo:** Las proyecciones de carga de trabajo, basadas en análisis históricos y previsiones futuras, ayudan a determinar la capacidad necesaria del VMSS. Esto incluye picos de demanda y periodos de baja actividad, lo que permite configurar reglas de escalabilidad automática.

- **Requisitos de Rendimiento:** Definir los requisitos de rendimiento, como la cantidad de CPU, memoria y almacenamiento necesarios, es crucial. Estos requisitos deben alinearse con las necesidades operativas del proyecto para asegurar un rendimiento óptimo sin incurrir en costos innecesarios.

- **Política de Disponibilidad y Tolerancia a Fallos y criticidad del dato:** La política de disponibilidad y la tolerancia a fallos de la empresa determinan la configuración de redundancia y recuperación ante desastres del VMSS. Esto incluye la elección de zonas de disponibilidad y la configuración de backups. También es importante considerar la criticidad del recurso que se va a desplegar y de los datos que este contiene. Esto afectará significativamente a las características de los recursos a implementar.

- **Estrategia de Financiación:** La estrategia de financiación del proyecto, incluyendo fuentes de financiamiento y opciones de pago, influye en la capacidad de inversión en infraestructura de TI. Es importante considerar opciones como "Pay-as-you-go" o "Compute Pre-Purchase" para optimizar costos.

- **Requisitos de Seguridad y Cumplimiento:** Los requisitos de seguridad y cumplimiento normativo de la empresa deben ser considerados para configurar adecuadamente las políticas de seguridad y acceso del VMSS. Esto incluye la implementación de firewalls, encriptación y auditorías.
- **Capacidad de Escalabilidad:** La capacidad de escalabilidad del proyecto, basada en la proyección de crecimiento y expansión, determina la configuración inicial y futura del VMSS.
- **Evaluación de Indicadores de Rentabilidad:** Indicadores como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) ayudan a evaluar la viabilidad financiera del proyecto y a tomar decisiones informadas sobre la inversión en infraestructura de TI.

Definir estos parámetros de manera detallada y precisa permitirá adaptar las características del Virtual Machine Scale Set de Azure a las necesidades específicas del proyecto y de la empresa, optimizando así el uso de los recursos y asegurando la viabilidad financiera y operativa del proyecto.

4.3.3 Relación entre las variables del Scale Set y los Parámetros Financieros y Operativos de la Empresa

A continuación se presentan las distintas relaciones entre las variables que influyen en el costo de un Azure Virtual Machine Scale Set (VMSS) y los parámetros financieros y operativos de la empresa. En cada celda se indica si una variable afecta a la otra. Posteriormente, se proporciona una breve explicación de cada una de las relaciones.

	Presupuesto Total del Proyecto	Horizonte Temporal del Proyecto	Proyecciones de Carga de Trabajo	Requisitos de Rendimiento	Política de Disponibilidad y Tolerancia a Fallos	Estrategia de Financiación	Requisitos de Seguridad y Cumplimiento	Capacidad de Escalabilidad	Evaluación de Indicadores de Rentabilidad
Número de Instancias	Sí	Sí	Sí	Sí	Sí	Sí	No	Sí	Sí
Tamaño de la VM	Sí	Sí	Sí	Sí	No	Sí	No	Sí	Sí
Región	Sí	No	No	No	No	Sí	No	No	Sí
Tipo de Almacenamiento	Sí	No	No	Sí	No	Sí	No	No	Sí
Horas de Uso	Sí	Sí	Sí	No	No	Sí	No	No	Sí
Opciones de Pago	Sí	Sí	No	No	No	Sí	No	No	Sí
Prioridad de la VM	Sí	No	No	No	No	Sí	No	No	Sí
Política de Expulsión	No	No	No	No	Sí	No	No	No	No
Configuración de Red	Sí	No	No	Sí	No	No	Sí	No	No

Figure 4.3: Relación entre las variables del Scale Set y los Parámetros Financieros y Operativos de la Empresa

Número de Instancias

- **Presupuesto Total del Proyecto:** El número de instancias en un VMSS afecta directamente el costo total del proyecto. Cada instancia adicional incrementa el costo total debido a los gastos asociados con la ejecución de máquinas virtuales, almacenamiento y red. Por lo tanto, es crucial ajustar el número de instancias para mantenerse dentro del presupuesto asignado. Un mayor número de instancias puede proporcionar mayor capacidad y redundancia, pero también incrementa los costos operativos.

- **Horizonte Temporal del Proyecto:** La duración del proyecto influye en la cantidad de instancias necesarias a lo largo del tiempo. Para proyectos a corto plazo, puede ser viable mantener un número elevado de instancias para cumplir con los requisitos de rendimiento en un período breve. En cambio, para proyectos a largo plazo, es importante planificar una estrategia de escalabilidad que permita ajustar el número de instancias según las necesidades cambiantes, optimizando así los costos a lo largo del tiempo.

- **Proyecciones de Carga de Trabajo:** Las proyecciones de carga de trabajo determinan la demanda esperada en términos de procesamiento, memoria y almacenamiento. Ajustar el número de instancias según estas proyecciones es esencial para asegurar que el VMSS pueda manejar picos de demanda sin comprometer el rendimiento. Durante periodos de baja actividad, se puede reducir el número de instancias para minimizar costos, mientras que durante picos de demanda, se puede aumentar el número de instancias para asegurar un rendimiento óptimo. Este parámetro también puede afectar al número máximo de instancias que se permita dentro del scale set.

- **Requisitos de Rendimiento:** Los requisitos de rendimiento, como la cantidad de CPU, memoria y almacenamiento necesarios, influyen en el número de instancias requeridas. Si los requisitos de rendimiento son altos, puede ser necesario desplegar más instancias para distribuir la carga de trabajo y evitar cuellos de botella. Esto asegura que la aplicación o servicio mantenga un rendimiento óptimo, pero también incrementa los costos operativos. Este parámetro también puede afectar al número máximo de instancias que se permita dentro del scale set.

- **Política de Disponibilidad y Tolerancia a Fallos y criticidad del dato:** Para asegurar alta disponibilidad y tolerancia a fallos, puede ser necesario desplegar instancias adicionales en diferentes zonas de disponibilidad o regiones. Esto proporciona redundancia y asegura que el servicio permanezca operativo incluso en caso de fallos en una zona o región específica. También influye la criticidad de los recursos que se quieran desplegar. Sin embargo, esta estrategia incrementa el número de instancias y, por ende, los costos

asociados.

- **Estrategia de Financiación:** La estrategia de financiación del proyecto, incluyendo las opciones de pago como "Pay-as-you-go" o "Compute Pre-Purchase", influye en la capacidad de inversión en infraestructura de TI. Diferentes estrategias pueden permitir más o menos instancias. Por ejemplo, "Compute Pre-Purchase" puede ofrecer descuentos significativos para compromisos a largo plazo, permitiendo desplegar más instancias dentro del mismo presupuesto.
- **Capacidad de Escalabilidad:** La capacidad de escalabilidad del proyecto, basada en la proyección de crecimiento y expansión, determina la configuración inicial y futura del VMSS. Es importante prever la necesidad de escalar el número de instancias vertical u horizontalmente para adaptarse a cambios en la demanda. Una estrategia de escalabilidad bien planificada permite ajustar el número de instancias de manera eficiente, optimizando costos y asegurando un rendimiento constante.
- **Evaluación de Indicadores de Rentabilidad:** El número de instancias impacta directamente en los costos y, por ende, en la rentabilidad del proyecto. Indicadores como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) ayudan a evaluar la viabilidad financiera del proyecto. Un mayor número de instancias puede mejorar el rendimiento y la disponibilidad, pero también incrementa los costos operativos, afectando negativamente estos indicadores. Es crucial encontrar un equilibrio que maximice la rentabilidad mientras se cumplen los requisitos operativos.

Tamaño de la VM

- **Presupuesto Total del Proyecto:** El tamaño de las máquinas virtuales (VMs) en un VMSS tiene un impacto directo en el costo total del proyecto. Las VMs más grandes, con más recursos de CPU, memoria y almacenamiento, son más costosas que las VMs más pequeñas. Por lo tanto, es crucial seleccionar el tamaño adecuado de VM para cumplir con los requisitos de rendimiento sin exceder el presupuesto asignado. Un tamaño excesivo de VM puede resultar en costos innecesarios, mientras que un tamaño insuficiente puede comprometer el rendimiento.
- **Horizonte Temporal del Proyecto:** La duración del proyecto puede influir en la elección del tamaño de las VMs. Para proyectos a corto plazo, puede ser viable utilizar VMs más grandes y potentes para cumplir con los requisitos de rendimiento en un período breve, ya que los costos adicionales se compensan con la duración limitada del proyecto. En cambio, para proyec-

tos a largo plazo, puede ser más rentable optar por VMs más pequeñas y escalar horizontalmente según sea necesario, optimizando así los costos a lo largo del tiempo.

- **Proyecciones de Carga de Trabajo:** Las proyecciones de carga de trabajo son fundamentales para seleccionar el tamaño adecuado de las VMs en un VMSS. Si se espera una carga de trabajo intensiva en términos de procesamiento, memoria o almacenamiento, se necesitarán VMs más grandes para manejar esta demanda de manera eficiente. Por otro lado, si la carga de trabajo es relativamente ligera, se pueden utilizar VMs más pequeñas para reducir costos sin comprometer el rendimiento.

- **Requisitos de Rendimiento:** Los requisitos de rendimiento, como la cantidad de CPU, memoria y almacenamiento necesarios, son un factor clave para determinar el tamaño de las VMs. Si los requisitos de rendimiento son altos, se necesitarán VMs más grandes con más recursos para cumplir con estas demandas. Por ejemplo, aplicaciones intensivas en cálculos o bases de datos de gran tamaño pueden requerir VMs con más CPU y memoria, respectivamente.

- **Estrategia de Financiación:** La estrategia de financiación del proyecto puede influir en la elección del tamaño de las VMs. Diferentes opciones de pago, como "Pay-as-you-go" o "Compute Pre-Purchase", pueden ofrecer descuentos significativos para compromisos a largo plazo o para ciertos tamaños de VM. Esto puede permitir la elección de VMs más grandes dentro del mismo presupuesto, mejorando el rendimiento sin incrementar significativamente los costos.

- **Capacidad de Escalabilidad:** La capacidad de escalabilidad del proyecto determina la necesidad de prever la posibilidad de escalar el tamaño de las VMs en el futuro. Si se espera un crecimiento significativo en la carga de trabajo, puede ser necesario comenzar con VMs más pequeñas y planificar una estrategia para escalar verticalmente a VMs más grandes según sea necesario. Esto permite optimizar los costos iniciales y ajustar los recursos según las necesidades cambiantes.

- **Evaluación de Indicadores de Rentabilidad:** El tamaño de las VMs en un VMSS impacta directamente en los costos operativos y, por ende, en la rentabilidad del proyecto. Indicadores como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) se ven afectados por los costos asociados con el tamaño de las VMs. VMs más grandes pueden mejorar el rendimiento, pero también incrementan los costos, lo que puede afectar negativamente estos indicadores. Es crucial encontrar un equilibrio entre el

tamaño de las VMs y la rentabilidad del proyecto.

Región

- **Presupuesto Total del Proyecto:** La región en la que se despliega un Azure Virtual Machine Scale Set (VMSS) afecta directamente el costo total del proyecto debido a las variaciones en los precios regionales. Azure tiene centros de datos en múltiples regiones alrededor del mundo, y los costos de los recursos (como máquinas virtuales, almacenamiento y red) pueden variar significativamente entre estas regiones. Por ejemplo, el costo de ejecutar una VM en la región de EE.UU. Este puede ser diferente al costo en la región de Europa Occidental. Seleccionar una región con costos más bajos puede ayudar a mantenerse dentro del presupuesto total del proyecto, mientras que elegir una región más costosa puede requerir ajustes en otros aspectos del presupuesto para compensar los costos adicionales.
- **Estrategia de Financiación:** La estrategia de financiación del proyecto puede influir en la elección de la región para desplegar el VMSS. Diferentes opciones de pago, como "Pay-as-you-go" o "Compute Pre-Purchase", pueden ofrecer descuentos específicos en ciertas regiones. Además, algunas estrategias de financiación pueden permitir aprovechar incentivos o programas de descuento regionales ofrecidos por Azure. Por ejemplo, comprometerse a un uso a largo plazo en una región específica puede resultar en tarifas más bajas. La elección de la región debe alinearse con la estrategia de financiación para optimizar los costos y maximizar el valor del presupuesto asignado.
- **Evaluación de Indicadores de Rentabilidad:** La región en la que se despliega el VMSS impacta directamente en los costos operativos y, por ende, en la rentabilidad del proyecto. Indicadores financieros como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) se ven afectados por los costos regionales. Desplegar el VMSS en una región con costos más bajos puede mejorar estos indicadores al reducir los gastos operativos, aumentando así la rentabilidad del proyecto. Por el contrario, elegir una región con costos más altos puede disminuir la rentabilidad, a menos que se justifique por otros factores como la proximidad a los usuarios finales, requisitos de cumplimiento normativo o necesidades específicas de latencia.

Tipo de Almacenamiento

- **Presupuesto Total del Proyecto:** El tipo de almacenamiento seleccionado para un Azure Virtual Machine Scale Set (VMSS) afecta directamente el costo total del proyecto. Azure ofrece diferentes tipos de almacenamiento,

como almacenamiento estándar (HDD) y almacenamiento premium (SSD). El almacenamiento premium es más costoso que el estándar debido a su mayor rendimiento y menor latencia. Por lo tanto, la elección del tipo de almacenamiento debe alinearse con el presupuesto total del proyecto. Optar por almacenamiento premium puede incrementar significativamente los costos, lo que puede requerir ajustes en otras áreas del presupuesto para compensar estos gastos adicionales.

- **Requisitos de Rendimiento:** Los requisitos de rendimiento del proyecto influyen en la elección del tipo de almacenamiento. El almacenamiento premium (SSD) ofrece mejor rendimiento en términos de velocidad de lectura/escritura y latencia en comparación con el almacenamiento estándar (HDD). Si el proyecto requiere un alto rendimiento de E/S (entrada/salida), como en el caso de bases de datos transaccionales o aplicaciones de análisis en tiempo real, el almacenamiento premium puede ser necesario para cumplir con estos requisitos. Sin embargo, para cargas de trabajo menos intensivas, el almacenamiento estándar puede ser suficiente y más rentable.

- **Estrategia de Financiación:** La estrategia de financiación del proyecto puede influir en la elección del tipo de almacenamiento. Diferentes opciones de pago, como "Pay-as-you-go" o "Compute Pre-Purchase", pueden ofrecer descuentos específicos para ciertos tipos de almacenamiento. Además, algunas estrategias de financiación pueden permitir la elección de almacenamiento premium dentro del mismo presupuesto, especialmente si se usan programas de descuento o compromisos a largo plazo. La elección del tipo de almacenamiento debe alinearse con la estrategia de financiación para optimizar los costos y maximizar el valor del presupuesto asignado.

- **Evaluación de Indicadores de Rentabilidad:** El tipo de almacenamiento seleccionado impacta directamente en los costos operativos y, por ende, en la rentabilidad del proyecto. Indicadores financieros como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) se ven afectados por los costos asociados con el almacenamiento. El almacenamiento premium, aunque más costoso, puede mejorar el rendimiento y la eficiencia operativa, lo que puede traducirse en beneficios financieros a largo plazo. Sin embargo, es crucial evaluar si los beneficios de rendimiento justifican los costos adicionales. Optar por almacenamiento estándar puede mejorar estos indicadores al reducir los gastos operativos, siempre y cuando cumpla con los requisitos de rendimiento del proyecto.

Opciones de Pago

- **Presupuesto Total del Proyecto:** Las opciones de pago disponibles en Azure para un Virtual Machine Scale Set (VMSS) pueden afectar directamente el costo total del proyecto. Azure ofrece diferentes opciones de pago, como "Pay-as-you-go" y "Compute Pre-Purchase" (CPP). La opción "Pay-as-you-go" implica pagar por los recursos utilizados según las tarifas estándar, mientras que "Compute Pre-Purchase" permite realizar un pago anticipado por un compromiso de uso a largo plazo, lo que puede ofrecer descuentos significativos. La elección de la opción de pago adecuada puede ayudar a optimizar los costos y mantenerse dentro del presupuesto total del proyecto.

- **Horizonte Temporal del Proyecto:** La duración del proyecto influye en la elección de la opción de pago más conveniente. Para proyectos a corto plazo, la opción "Pay-as-you-go" puede ser más adecuada, ya que no requiere un compromiso a largo plazo y permite una mayor flexibilidad en el uso de recursos. Por otro lado, para proyectos a largo plazo, la opción "Compute Pre-Purchase" puede ser más rentable, ya que los descuentos por compromisos a largo plazo pueden compensar los costos adicionales a lo largo del tiempo.

- **Estrategia de Financiación:** La estrategia de financiación del proyecto está estrechamente relacionada con la elección de la opción de pago. Diferentes estrategias de financiación pueden influir en la capacidad de realizar pagos anticipados o compromisos a largo plazo. Por ejemplo, si el proyecto cuenta con un presupuesto limitado y flujos de efectivo restringidos, la opción "Pay-as-you-go" puede ser más adecuada, ya que no requiere un desembolso inicial significativo. Por otro lado, si el proyecto tiene acceso a financiamiento o puede realizar inversiones a largo plazo, la opción "Compute Pre-Purchase" puede ser más conveniente, ya que los descuentos obtenidos pueden mejorar la rentabilidad del proyecto.

- **Evaluación de Indicadores de Rentabilidad:** Las opciones de pago impactan directamente en los costos operativos y, por ende, en la rentabilidad del proyecto. Indicadores financieros como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) se ven afectados por los costos asociados con las opciones de pago. La opción "Compute Pre-Purchase", aunque implica un desembolso inicial mayor, puede mejorar estos indicadores al reducir los costos a largo plazo gracias a los descuentos obtenidos. Por otro lado, la opción "Pay-as-you-go" puede ser más conveniente para proyectos a corto plazo o con flujos de efectivo limitados, pero puede resultar en costos más altos a largo plazo, lo que puede afectar negativamente la rentabilidad del proyecto.

Prioridad de la VM

- **Presupuesto Total del Proyecto:** La prioridad de las máquinas virtuales (VMs) en un Azure Virtual Machine Scale Set (VMSS) afecta directamente el costo total del proyecto. Azure ofrece dos tipos de prioridades: VMs regulares y Azure Spot Virtual Machines (Spot VMs). Las Spot VMs son instancias de baja prioridad que aprovechan la capacidad de cómputo no utilizada en Azure, lo que las hace más económicas que las VMs regulares. Sin embargo, las Spot VMs pueden ser desalojadas cuando Azure necesita recuperar esa capacidad. Utilizar Spot VMs puede reducir significativamente los costos del proyecto, pero también implica un riesgo de interrupción del servicio. Por lo tanto, la elección de la prioridad de las VMs debe alinearse con el presupuesto total del proyecto y la tolerancia al riesgo.
- **Estrategia de Financiación:** La estrategia de financiación del proyecto puede influir en la elección de la prioridad de las VMs. Diferentes estrategias de financiación pueden permitir o limitar el uso de Spot VMs. Por ejemplo, si el proyecto cuenta con un presupuesto limitado y flujos de efectivo restringidos, el uso de Spot VMs puede ser una opción atractiva para reducir costos. Sin embargo, si el proyecto requiere una alta disponibilidad y tiene acceso a financiamiento, puede ser más conveniente utilizar VMs regulares para evitar interrupciones del servicio.
- **Evaluación de Indicadores de Rentabilidad:** La prioridad de las VMs en un VMSS impacta directamente en los costos operativos y, por ende, en la rentabilidad del proyecto. Indicadores financieros como el Valor Presente Neto (VPN) y la Tasa Interna de Retorno (TIR) se ven afectados por los costos asociados con la prioridad de las VMs. El uso de Spot VMs puede mejorar estos indicadores al reducir los costos operativos, lo que puede aumentar la rentabilidad del proyecto. Sin embargo, es crucial evaluar el impacto potencial de los desalojos en la disponibilidad del servicio y los costos asociados con la implementación de estrategias de mitigación, ya que estos factores pueden afectar negativamente la rentabilidad.

Política de Expulsión

- **Política de Disponibilidad y Tolerancia a Fallos:** La política de expulsión seleccionada para las Azure Spot Virtual Machines (Spot VMs) en un Virtual Machine Scale Set (VMSS) influye directamente en cómo se manejan las VMs desalojadas y, por lo tanto, afecta la política de disponibilidad y tolerancia a fallos del proyecto.

  Azure ofrece dos políticas de expulsión principales:

- Deallocate: Cuando una Spot VM es desalojada, se detiene y se libera de la capacidad de cómputo, pero se mantienen los discos asociados. Esto permite reiniciar la VM más tarde cuando haya capacidad disponible.
- Delete: Cuando una Spot VM es desalojada, se elimina completamente, incluyendo los discos asociados.

La política de expulsión "Deallocate" puede ser más adecuada para cargas de trabajo que requieren una alta disponibilidad y tolerancia a fallos, ya que permite reiniciar las VMs desalojadas sin perder datos. Sin embargo, esto implica un tiempo de inactividad mientras se espera a que haya capacidad disponible. Por otro lado, la política "Delete" puede ser más conveniente para cargas de trabajo que no requieren una alta disponibilidad o que pueden tolerar interrupciones del servicio. Esta política puede ser más eficiente en términos de costos, ya que no se incurre en gastos de almacenamiento para los discos detenidos. La elección de la política de expulsión debe alinearse con la política de disponibilidad y tolerancia a fallos del proyecto, considerando el impacto potencial de los desalojos en la continuidad del servicio y la importancia de mantener los datos intactos.

Configuración de Red

- **Presupuesto Total del Proyecto:** La configuración de red para un Azure Virtual Machine Scale Set (VMSS) puede afectar directamente el costo total del proyecto. Elementos como el uso de redes aceleradas, equilibradores de carga, grupos de seguridad de red y la cantidad de ancho de banda de red consumido pueden incrementar los costos. Además, el consumo de ancho de banda de red puede ser un factor significativo en los costos, especialmente para cargas de trabajo que requieren una gran cantidad de transferencia de datos. Es crucial evaluar cuidadosamente la configuración de red necesaria y equilibrar los requisitos de rendimiento y seguridad con el presupuesto total del proyecto.
- **Requisitos de Rendimiento:** La configuración de red adecuada es fundamental para cumplir con los requisitos de rendimiento del proyecto. Una red mal configurada o con recursos insuficientes puede causar cuellos de botella y afectar negativamente el rendimiento de las aplicaciones o servicios que se ejecutan en el VMSS. Por ejemplo, si se espera un alto tráfico de red, puede ser necesario utilizar redes aceleradas o equilibradores de carga para garantizar un rendimiento óptimo. Además, la cantidad de ancho de banda de red disponible debe ser suficiente para manejar la carga de trabajo prevista. Es importante evaluar cuidadosamente los requisitos de rendimiento y selec-

cionar la configuración de red adecuada para cumplir con estos requisitos, incluso si esto implica un costo adicional.

4.4 Integración en Github

La automatización de la infraestructura es un componente esencial en la práctica moderna de DevOps, permitiendo despliegues consistentes, repetibles y escalables en entornos en la nube. Terraform, se ha convertido en una opción popular debido a su capacidad para gestionar recursos en una amplia gama de proveedores de servicios en la nube, incluyendo Azure. La potencia de Terraform aumenta cuando se combina con plataformas de integración y entrega continua (CI/CD) como GitHub Actions. Esta integración permite automatizar no solo el despliegue inicial de la infraestructura, sino también su mantenimiento y actualización continua.

En este apartado, se describe cómo se ha integrado el código de Terraform para el despliegue de esta solución completa en Azure, utilizando GitHub como sistema de control de versiones y GitHub Actions como motor de automatización. A través de esta integración, se logra un flujo de trabajo eficiente y automatizado que simplifica la gestión de la infraestructura y asegura que todos los cambios sean revisables y reproducibles.

4.4.1 Estructura del repositorio

El repositorio de github se ha estructurado en tres carpetas principales:

- **terraform:** esta carpeta contiene todos los ficheros de terraform que permiten definir los recursos de azure que van a ser desplegados y que han sido explicados previamente.
- **.github/workflows:** en esta carpeta se encuentran los ficheros yaml encargados de definir los flujos que posteriormente empleará github actions para desplegar los recursos de terraform y gestionar el escalado de estos.
- **python:** esta carpeta contiene scripts auxiliares que emplearán los flujos de github actions para gestionar el escalado de recursos.

4.4.1.1 Flujos de Github Actions

Dentro de la carpeta .github/workflows, se encuentran los archivos YAML que definen los flujos de trabajo de GitHub Actions. Estos archivos describen las acciones que deben ejecutarse en respuesta a eventos específicos, como commits o accionado manual. Los flujos de trabajo automatizan el despliegue y la gestión

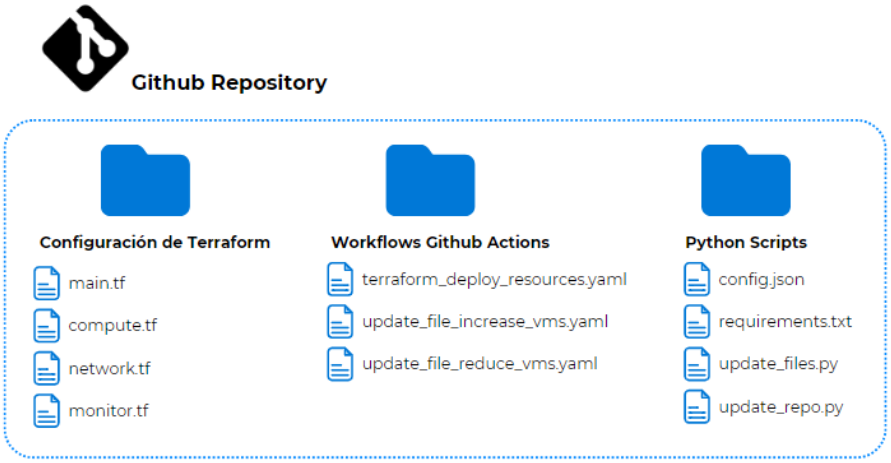


Figure 4.4: Estructura del Repositorio de Github.

de la infraestructura, asegurando que cada cambio en el código de Terraform se refleje de manera consistente en Azure. En esta carpeta se incluten los siguientes archivos archivos:

terraform_deploy_resources.yaml
Este flujo de trabajo automatiza el despliegue de infraestructura, asegurando que los archivos de Terraform estén formateados correctamente, que la configuración sea válida y que los recursos se desplieguen de manera controlada y segura.

Presenta las siguientes características:

- **Disparador del flujo de trabajo:** Se ha configurado el flujo de trabajo para que se active manualmente mediante el disparador workflow_dispatch o cuando haya un push en la rama "main".
- **Variables de entorno:** Se definen las variables de entorno necesarias para la autenticación y configuración de Terraform.
  - **ARM_CLIENT_ID:** Variable de entorno para la autenticación en Azure.
  - **ARM_CLIENT_SECRET** Variables de entorno para la autenticación en Azure.
  - **ARM_SUBSCRIPTION_ID:** Variables de entorno para la autenticación en Azure.
  - **ARM_TENANT_ID:** Variables de entorno para la autenticación en

Azure.

  - **TF_VERSION:** Versión de Terraform a usar.
  - **TF_VAR_PUB_KEY:** Clave pública para la autenticación SSH.
  - **TF_VAR_GH_TOKEN:** Token de GitHub para la autenticación.

- Pasos del trabajo:
  - **Checkout:** Paso para clonar el repositorio en el runner de GitHub Actions.
  - **Terraform Format:** Paso para formatear los archivos de Terraform.
  - **Terraform Init:** Paso para inicializar el entorno de Terraform.
  - **Terraform Validate:** Paso para validar la configuración de Terraform.
  - **Terraform Plan:** Paso para crear el plan de ejecución de Terraform.
  - **Terraform Apply:** Paso para aplicar el plan de Terraform y desplegar los recursos.

update_file_increase_vms.yaml & update_file_increase_vms.yaml
Estos workflows se desencadenan mediante un evento "repository_dispatch" con el tipo increase-n-vms o reduce-n-vms, repectivamente. Se utiliza para aumentar el número de máquinas virtuales en Azure utilizando Terraform y scripts de Python. El trabajo se ejecuta en una máquina ubuntu-latest y configura varias variables de entorno necesarias para la autenticación y configuración de Terraform. Los pasos incluyen clonar el repositorio, configurar Python, instalar dependencias, y ejecutar scripts de Python para modificar archivos y actualizar el repositorio.

Presenta las siguientes características:

- **Disparador del flujo de trabajo:** Este evento se activa cuando se realiza una llamada a la API de GitHub con un repository_dispatch con el tipo especificado. En este caso, el tipo es increase-n-vms.
- **Variables de entorno:** Se definen las variables de entorno necesarias para la autenticación y configuración de Terraform.
  - **ARM_CLIENT_ID:** Variable de entorno para la autenticación en Azure.
  - **ARM_CLIENT_SECRET** Variables de entorno para la autenticación en Azure.

  - **ARM_SUBSCRIPTION_ID:** Variables de entorno para la autenticación en Azure.
  - **ARM_TENANT_ID:** Variables de entorno para la autenticación en Azure.
  - **TF_VERSION:** Versión de Terraform a usar.
  - **TF_VAR_PUB_KEY:** Clave pública para la autenticación SSH.
  - **TF_VAR_GH_TOKEN:** Token de GitHub para la autenticación.
  - **ACTION:** Induca la acción a realizar, en este caso "increase_vm" o "reduce_vm".
- Pasos del trabajo:
  - **checkout:** Este paso usa la acción actions/checkout@v4 para clonar el repositorio en la máquina de ejecución.
  - **Set up Python:** Este paso usa la acción actions/setup-python@v4 para configurar Python 3.x en la máquina de ejecución.
  - **Install dependencies:** Este paso instala las dependencias de Python especificadas en el archivo requirements.txt del directorio python.
  - **Run update_files.py script:** Este paso ejecuta el script de Python llamado update_files.py que se encuentra en la carpeta python. Este script se encarga modificar los archivos terraform con los parámetros indicados.
  - **Run update_compute.py script:** Este paso ejecuta el script de Python llamado update_repo.py que se encuentra en la carpeta python. Este script se encarga de actualizar el repositorio con los cambios realizados en el paso anterior.

terraform_destroy.yaml
Este archivo se encarga de eliminar todos los recursos creados una vez que ya no son necesarios.

Presenta las siguientes características:

- **Disparador del flujo de trabajo:** Se ha configurado el flujo de trabajo para que se active manualmente mediante el disparador workflow_dispatch.
- **Variables de entorno:** Se definen las variables de entorno necesarias para la autenticación y configuración de Terraform.

- **ARM_CLIENT_ID:** Variable de entorno para la autenticación en Azure.
- **ARM_CLIENT_SECRET** Variables de entorno para la autenticación en Azure.
- **ARM_SUBSCRIPTION_ID:** Variables de entorno para la autenticación en Azure.
- **ARM_TENANT_ID:** Variables de entorno para la autenticación en Azure.
- **TF_VERSION:** Versión de Terraform a usar.
- **TF_VAR_PUB_KEY:** Clave pública para la autenticación SSH.
- **TF_VAR_GH_TOKEN:** Token de GitHub para la autenticación.

- Pasos del trabajo:
  - **Checkout:** Paso para clonar el repositorio en el runner de GitHub Actions.
  - **Terraform Init:** Paso para inicializar el entorno de Terraform.
  - **Terraform Destroy:** Ejecuta terraform destroy para eliminar todos los recursos creados anteriormente en Azure.

Capítulo 5

Resultados

En el marco del proyecto "Framework de Automatización FinOps", se han logrado avances significativos que demuestran la viabilidad y efectividad de la solución propuesta. A continuación, se presenta una descripción narrativa de los principales hitos alcanzados y su impacto en la optimización de recursos en la nube.

Se ha desarrollado un framework robusto y flexible utilizando Terraform, una herramienta de infraestructura como código (IaC). Este framework permite la gestión eficiente del escalado y desescalado de recursos en entornos cloud, específicamente en Azure. La arquitectura modular del framework facilita su adaptación a diferentes entornos y requisitos de los usuarios, asegurando una implementación consistente y reproducible de los recursos necesarios.

El framework desarrollado permite una automatización efectiva del despliegue y gestión de la infraestructura en la nube. A través de la integración con GitHub Actions, se ha logrado automatizar no solo el despliegue inicial de la infraestructura, sino también su mantenimiento y actualización continua. Esto se traduce en una reducción significativa de errores manuales y una mejora en la eficiencia operativa.

Adicionalmente, se ha implementado un sistema de monitorización y generación de alertas utilizando Azure Monitor. Este sistema permite supervisar el uso de recursos y activar acciones automatizadas en respuesta a umbrales predefinidos. Por ejemplo, cuando el uso de CPU o memoria supera ciertos límites, se generan alertas que desencadenan procesos de ajuste dinámico de los recursos, optimizando así el rendimiento y los costos asociados.

Se ha realizado un análisis detallado para identificar los parámetros empresariales clave que influyen en la configuración y gestión del Virtual Machine Scale Set

(VMSS). Estos parámetros incluyen el presupuesto total del proyecto, el horizonte temporal, las proyecciones de carga de trabajo, los requisitos de rendimiento, y la política de disponibilidad y tolerancia a fallos. La integración de estos parámetros en el framework permite una asignación estratégica de los recursos, alineando las decisiones técnicas con los objetivos financieros y estratégicos de la organización.

En resumen, el proyecto ha alcanzado un hito significativo con el desarrollo de un Producto Mínimo Viable (MVP) funcional que integra principios de FinOps con la automatización de infraestructura en la nube. Este logro sienta las bases para futuras mejoras y expansiones, y demuestra el potencial del framework para contribuir significativamente a la adopción de prácticas FinOps en la industria, optimizando costos y mejorando la eficiencia operativa de las organizaciones.

Capítulo 6

Discusión y Trabajos Futuros

El desarrollo de este framework de automatización FinOps para la gestión eficiente de recursos en la nube representa un paso significativo hacia la optimización de costos y la alineación de los recursos tecnológicos con los objetivos empresariales. A lo largo de este proyecto, se ha podido constatar cómo la integración de principios FinOps con la automatización de infraestructura puede transformar la manera en que las organizaciones gestionan sus entornos cloud.

Una de las principales revelaciones durante el desarrollo de este proyecto ha sido la importancia crítica de vincular las decisiones de escalado de recursos directamente con las variables de negocio. Este enfoque marca un cambio paradigmático respecto a los sistemas tradicionales de asignación de recursos estáticos o incluso aquellos con capacidades de autoescalado basadas únicamente en métricas operativas. La capacidad de este framework para considerar parámetros empresariales como presupuestos, proyecciones de carga de trabajo y requisitos de rendimiento en tiempo real, permite una gestión de recursos mucho más alineada con las necesidades reales del negocio.

Es probable que en un futuro cercano, plataformas como la desarrollada en este proyecto, que permiten un consumo de recursos mucho más cercano a las variables de negocio de la empresa, sustituyan con los sistemas de asignación de recursos estáticos o incluso aquellos con autoescalado pero que se auto-gestionan en base a métricas puramente operativas de los recursos desplegados. Esta evolución no solo optimizará los costos, sino que también mejorará la agilidad empresarial, permitiendo a las organizaciones adaptarse rápidamente a las cambiantes condiciones del mercado.

La implementación de este framework ha demostrado también la importancia de la automatización en la gestión de infraestructuras cloud. La capacidad de responder

en tiempo real a las fluctuaciones en la demanda, ajustando automáticamente los recursos disponibles, no solo optimiza los costos, sino que también mejora significativamente la experiencia del usuario final al garantizar un rendimiento constante incluso en momentos de alta demanda.

Sin embargo, es este proyecto es solo el comienzo de un camino más largo hacia la optimización completa de los recursos en la nube. A medida que las tecnologías evolucionan y las necesidades empresariales se vuelven más complejas, será necesario seguir desarrollando y refinando estas soluciones. En este contexto, se presentan varias líneas de trabajo futuro que podrían expandir y mejorar significativamente las capacidades del framework desarrollado:

- **Integración con sistemas de inteligencia artificial:** El potencial de integrar algoritmos de inteligencia artificial y aprendizaje automático en este framework es inmenso. Estos sistemas podrían analizar patrones históricos de uso, tendencias de mercado y otros factores externos para predecir con mayor precisión las necesidades futuras de recursos. Por ejemplo, un modelo de IA podría anticipar picos de demanda basados en eventos programados o tendencias estacionales, permitiendo un escalado proactivo en lugar de reactivo. Además, la IA podría detectar anomalías en el uso de recursos que podrían indicar ciberataques o mal funcionamiento del sistema, activando respuestas automatizadas para mitigar riesgos.

- **Expansión a múltiples proveedores de nube:** Aunque el framework actual se centra en Azure, la realidad es que muchas organizaciones operan en entornos multi-cloud. Expandir las capacidades del framework para abarcar otros proveedores como AWS, Google Cloud o IBM Cloud abriría nuevas posibilidades de optimización. Esta expansión permitiría a las organizaciones no solo optimizar recursos dentro de un proveedor, sino también distribuir cargas de trabajo entre diferentes proveedores basándose en costos, rendimiento y disponibilidad en tiempo real. Podría llegar a desarrollarse un sistema capaz de mover automáticamente cargas de trabajo a AWS durante períodos de precios más bajos, y luego volver a Azure al cambiar la condiciones.

- **Incorporación de métricas de sostenibilidad:** Con la creciente importancia de la sostenibilidad en el mundo empresarial, integrar métricas de impacto ambiental en este framework sería un paso natural y valioso. Esto podría incluir el seguimiento y la optimización del consumo de energía, la huella de carbono asociada con el uso de recursos en la nube, e incluso la priorización de centros de datos alimentados por energías renovables. Un framework que pudiera balancear automáticamente entre eficiencia de costos, rendimiento y sostenibilidad representaría un avance significativo en la

gestión responsable de recursos en la nube.

- **Desarrollo de interfaces de usuario más intuitivas:** El desarrollo de interfaces de usuario más intuitivas y visuales podría ampliar significativamente la adopción del framework. Una posible via de desarrollo sería la creación de un dashboard interactivo que no solo muestre métricas en tiempo real, sino que también permita a los usuarios no técnicos ajustar políticas de escalado, ver proyecciones de costos basadas en diferentes escenarios, y entender fácilmente el impacto de sus decisiones. Esto podría incluir visualizaciones 3D de la infraestructura, gráficos de flujo de datos en tiempo real, y herramientas de simulación "what-if" para planificar cambios futuros.

En conclusión, el framework desarrollado en este proyecto representa un avance significativo en la gestión eficiente de recursos en la nube, pero también sirve como punto de partida para innovaciones futuras aún más emocionantes. A medida que las organizaciones continúan su viaje hacia la transformación digital, soluciones como esta serán cada vez más cruciales para mantener la competitividad, la eficiencia y la sostenibilidad en un entorno altamente impulsado por la nube.

Appendix A

Alineación con los ODS

El proyecto propuesto se alinea con varios Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas, especialmente aquellos relacionados con la innovación tecnológica, la eficiencia operativa y económica, y la sostenibilidad empresarial. A continuación, se detallan los principales ODS con los que se alinea el proyecto

- **ODS 9 - Industria, Innovación e Infraestructura:** El proyecto contribuye directamente al ODS 9 al promover la innovación tecnológica en la gestión de recursos en entornos cloud. Al desarrollar un framework para automatizar el escalado y desescalado de recursos, se fomenta la creación de infraestructuras resilientes, sostenibles y adaptables, que son fundamentales para el desarrollo económico y social.
- **ODS 11 - Ciudades y Comunidades Sostenibles:** A través de la optimización de recursos en entornos cloud, el proyecto busca mejorar la eficiencia operativa y económica de las organizaciones, lo que puede tener un impacto positivo en la creación de comunidades sostenibles y resilientes. Al reducir el consumo innecesario de recursos y optimizar los costos asociados, se promueve un uso más eficiente de los recursos y se contribuye a la construcción de ciudades y comunidades más sostenibles.
- **ODS 12 - Producción y Consumo Responsables:** El proyecto fomenta prácticas de producción y consumo responsables al optimizar el uso de recursos en entornos cloud. Al automatizar el escalado y desescalado de recursos en función de la demanda real, se reduce el desperdicio de recursos y se promueve una utilización más eficiente de la infraestructura tecnológica, lo que contribuye a la reducción de la huella ambiental y al uso sostenible de los recursos.

- **ODS 13 - Acción por el Clima:** Al optimizar la gestión de recursos en entornos cloud, el proyecto puede contribuir indirectamente a la mitigación del cambio climático. Al reducir el consumo energético asociado con la infraestructura tecnológica y promover prácticas de eficiencia energética, se puede reducir la emisión de gases de efecto invernadero y minimizar el impacto ambiental de las operaciones empresariales.

Bibliography

[1] FinOps Foundation. Finops framework overview. https://www.finops.org/framework/. Accessed on 11.07.2024.

[2] FinOps Foundation. Finops phases. https://www.finops.org/framework/phases/. Accessed on 11.07.2024.

[3] Training | Microsoft Learn. Describe infrastructure as a service - training. https://learn.microsoft.com/en-us/training/modules/describe-cloud-service-types/2-describe-infrastructure-service. Accessed on 11.07.2024.

[4] Terraform | HashiCorp Developer. What is terraform: Terraform: Hashicorp developer. https://developer.hashicorp.com/terraform/intro. Accessed on 11.07.2024.

[5] Forbes / EP. La inversión en "cloud" de las empresas españolas crece un 39% en 2023, según ipm, Aug 2023. Accessed on 11.07.2024.

[6] FinOps Foundation. What is finops? https://www.finops.org/introduction/what-is-finops/. Accessed on 11.07.2024.

[7] FinOps Foundation. Capability: Workload optimization. https://www.finops.org/framework/capabilities/workload-management-automation/. Accessed on 11.07.2024.

[8] Sergio Delgado Martorell. Solo el 16% de las empresas españolas ha implantado una automatización total. *La Razón*, Nov 2023. Accessed on 11.07.2024.

[9] Ali Sunyaev. *Cloud Computing*, pages 195–236. Springer International Publishing, Cham, 2020. Accessed on 11.07.2024.

[10] Keith Conway, Abdallah Saleme, Bhargs Srivathsan, and Konstantin Tyrman. The finops way: How to avoid the pitfalls to realizing cloud's value, Jan 2023. Accessed on 11.07.2024.

[11] FinOps Foundation. Finops landscape - finops tools, services & training. https://www.finops.org/landscape/?prod_TOOLS_SERVICES%5Bquery%5D=automation&prod_TOOLS_SERVICES%5BrefinementList%5D%5Bcategories%5D%5B0%5D=FinOps+Tool. Accessed on 11.07.2024.

[12] Apptio. Trusted technology investment decisions. https://www.apptio.com/, Apr 2024. Accessed on 11.07.2024.

[13] Flexera. It and cloud management, optimization and solutions. https://www.flexera.com/. Accessed on 11.07.2024.

[14] Nordcloud. Cloud services & training. https://nordcloud.com/, Mar 2024. Accessed on 11.07.2024.

[15] Michael Wurster, Uwe Breitenbücher, Michael Falkenthal, Christoph Krieger, Frank Leymann, Karoline Saatkamp, and Jacopo Soldani. The essential deployment metamodel: A systematic review of deployment automation technologies. *SICS Software-Intensive Cyber-Physical Systems*, 35(1–2):63–75, Aug 2019. Accessed on 11.07.2024.

[16] Chef Software. Configuration management system software - chef infra: Chef. https://www.chef.io/products/chef-infra. Accessed on 11.07.2024.

[17] Puppet by Perforce. Infrastructure automation & compliance at enterprise scale. https://www.puppet.com/. Accessed on 11.07.2024.

[18] Amazon Web Services. Infrastructure as code provisioning tool - aws cloudformation - aws. https://aws.amazon.com/cloudformation/. Accessed on 11.07.2024.

[19] Jenkins. =https://www.jenkins.io/. Accessed on 11.07.2024.

[20] Terraform. Terraform by hashicorp. https://www.terraform.io/. Accessed on 11.07.2024.

[21] Ansible Collaborative et al. How ansible works. https://www.ansible.com/overview/how-ansible-works, Apr 2024. Accessed on 11.07.2024.

[22] Kubernetes. Production-grade container orchestration. https://kubernetes.io/. Accessed on 11.07.2024.

[23] Parker Leach and Clairecadman. What is puppet? https://www.puppet.com/docs/puppet/7/what_is_puppet.html, Apr 2024. Accessed on 11.07.2024.

[24] OpenStack. Heat. https://wiki.openstack.org/wiki/Heat. Accessed on 11.07.2024.

[25] vmware. Support for saltstack. https://www.vmware.com/support/acquisitions/saltstack.html. Accessed on 11.07.2024.

[26] Juju. Juju documentation. https://juju.is/docs/juju. Accessed on 11.07.2024.

[27] CFEngine. Product - overview. https://cfengine.com/product-overview/. Accessed on 11.07.2024.

[28] Microsoft Azure. Azure resource manager. https://azure.microsoft.com/en-us/get-started/azure-portal/resource-manager/. Accessed on 11.07.2024.

[29] Docker. Docker compose overview. https://docs.docker.com/compose/, Jan 2024. Accessed on 11.07.2024.

[30] Cloudify Platform. Cloudify documentation center: Cloudify documentation center. https://cloudify.co/about/. Accessed on 11.07.2024.

[31] Microsoft Learn Ju-Shim. Azure virtual machine scale sets overview - azure virtual machine scale sets. https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/overview. Accessed on 11.07.2024.

[32] Microsoft Learn. What is azure kubernetes service (aks)? - azure kubernetes service. https://learn.microsoft.com/en-us/azure/aks/what-is-aks. Accessed on 11.07.2024.

[33] Microsoft Learn. Azure functions overview. https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview?pivots=programming-language-csharp. Accessed on 11.07.2024.

[34] AWS. Auto scaling groups. https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html. Accessed on 11.07.2024.

[35] AWS. What is amazon eks? - amazon eks. https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html. Accessed on 11.07.2024.

[36] AWS. ¿qué es aws lambda? - aws lambda. https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html. Accessed on 11.07.2024.

[37] Google. Descripción general de gke | google kubernetes engine (gke) | google cloud. https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview?hl=es-419. Accessed on 11.07.2024.

[38] Google. Grupos de instancias | compute engine documentation | google cloud. https://cloud.google.com/compute/docs/instance-groups?hl=es-419. Accessed on 11.07.2024.

[39] Google. Descripción general de cloud functions | cloud functions documentation | google cloud. https://cloud.google.com/functions/docs/concepts/overview?hl=es-419. Accessed on 11.07.2024.

[40] Inesdi. Finops: Qué es y por qué usarlo para ahorrar en la nube. https://www.inesdi.com/blog/finops/. Accessed on 11.07.2024.

[41] FinOps Foundation. The finops foundation. https://www.finops.org/, journal=FinOps Foundation. Accessed on 11.07.2024.

[42] Linux Foundation. Linux foundation - decentralized innovation, built with trust. https://www.linuxfoundation.org/. Accessed on 11.07.2024.

[43] ACKstorm. Cómo empezar en finops con cloud financial management. https://www.ackstorm.com/como-empezar-en-finops-cloud-financial-management/, Feb 2024. Accessed on 11.07.2024.

[44] FinOps Foundation. Finops principles. https://www.finops.org/framework/principles/. Accessed on 11.07.2024.

[45] FinOps Foundation. Finops personas. https://www.finops.org/framework/personas/. Accessed on 11.07.2024.

[46] FinOps Foundation. Finops domains. https://www.finops.org/framework/domains/. Accessed on 11.07.2024.

[47] Tejaswi Redkar and Tony Guidici. *Windows Azure Platform*. Apress, 2011. Accessed on 11.07.2024.

[48] Google. ¿qué es cloud computing? https://cloud.google.com/learn/what-is-cloud-computing?hl=es. Accessed on 11.07.2024.

[49] Google cloud platform. https://console.cloud.google.com/welcome/new?pli=1, journal=Google cloud platform. Accessed on 11.07.2024.

[50] Datademia. La historia de aws, microsoft azure y gcp. https://datademia.es/blog/historia-aws-microsoft-azure-gcp, Aug 2022. Accessed on 11.07.2024.

[51] Training | Microsoft Learn. Descripción de azure virtual machines - training. https://learn.microsoft.com/es-es/training/modules/describe-azure-compute-networking-services/2-virtual-machines. Accessed on 11.07.2024.

[52] Training | Microsoft Learn. Descripción de azure functions - training. https://learn.microsoft.com/es-es/training/modules/describe-

azure-compute-networking-services/6-functions. Accessed on 11.07.2024.

[53] Training | Microsoft Learn. Descripción de las redes virtuales de azure - training. https://learn.microsoft.com/es-es/training/modules/describe-azure-compute-networking-services/8-virtual-network. Accessed on 11.07.2024.

[54] Training | Microsoft Learn. Describir azure dns - training. https://learn.microsoft.com/es-es/training/modules/describe-azure-compute-networking-services/12-domain-name-system. Accessed on 11.07.2024.

[55] Training | Microsoft Learn. Descripción de las cuentas de almacenamiento de azure - training. https://learn.microsoft.com/es-es/training/modules/describe-azure-storage-services/2-accounts. Accessed on 11.07.2024.

[56] Training | Microsoft Learn. Descripción de los servicios de almacenamiento de azure - training. https://learn.microsoft.com/es-es/training/modules/describe-azure-storage-services/4-describe-azure-storage-services. Accessed on 11.07.2024.

[57] Ritesh Modi. *Deep-dive terraform on Azure: Automated delivery and deployment of Azure Solutions*. Apress L.P, 2021. Accessed on 11.07.2024.

[58] AWS. What is infrastructure as code? - iac explained - aws. https://aws.amazon.com/what-is/iac/. Accessed on 11.07.2024.

[59] IBM. What is infrastructure as code (iac)? https://www.ibm.com/topics/infrastructure-as-code, Oct 2021. Accessed on 11.07.2024.

[60] Ansible Collaborative. Homepage | ansible collaborative. https://www.ansible.com/. Accessed on 11.07.2024.

[61] Terraform Registry. Terraform registry. https://registry.terraform.io/. Accessed on 11.07.2024.

[62] GitHub Docs. Acerca de git - documentación de github. https://docs.github.com/es/get-started/using-git/about-git. Accessed on 11.07.2024.

[63] Git. Reference git commands. https://git-scm.com/docs. Accessed on 11.07.2024.

[64] GitHub Docs. Entender las github actions - documentación de github. https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions. Accessed on 11.07.2024.

[65] Microsoft Learn. Administración de grupos de recursos: Azure portal - azure resource manager. https://learn.microsoft.com/es-es/azure/azure-resource-manager/management/manage-resource-groups-portal. Accessed on 11.07.2024.

[66] Terraform registry. Virtual machine scale set in flexible orchestration mode with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/orchestrated_virtual_machine_scale_set. Accessed on 11.07.2024.

[67] Terraform registry. Azure virtual network with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/virtual_network. Accessed on 11.07.2024.

[68] Terraform registry. Azure subnet with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/subnet. Accessed on 11.07.2024.

[69] Terraform registry. Azure network security group with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_security_group. Accessed on 11.07.2024.

[70] Terraform registry. Azure public ip address with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/public_ip. Accessed on 11.07.2024.

[71] Microsoft Learn. ¿qué es azure load balancer? - azure load balancer. https://learn.microsoft.com/es-es/azure/load-balancer/load-balancer-overview. Accessed on 11.07.2024.

[72] Terraform registry. Azure load balancer resource with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/lb#frontend_ip_configuration. Accessed on 11.07.2024.

[73] Terraform registry. Azure load balancer backend address pool with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/3.0.2/docs/resources/lb_backend_address_pool. Accessed on 11.07.2024.

[74] Terraform registry. Azure load balancer rule with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/lb_rule. Accessed on 11.07.2024.

[75] Terraform registry. Azure load balancer probe with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/lb_probe. Accessed on 11.07.2024.

[76] Microsoft Learn. Reglas nat de entrada - azure load balancer. https://learn.microsoft.com/es-es/azure/load-balancer/inbound-nat-rules. Accessed on 11.07.2024.

[77] Terraform registry. Azure load balancer nat rule with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/lb_nat_rule. Accessed on 11.07.2024.

[78] Microsoft Learn. ¿qué es azure nat gateway? https://learn.microsoft.com/es-es/azure/nat-gateway/nat-overview. Accessed on 11.07.2024.

[79] Terraform registry. Azure nat gateway with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/nat_gateway. Accessed on 11.07.2024.

[80] Terraform registry. Azure association of a nat gateway with a subnet within a virtual network using terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/subnet_nat_gateway_association. Accessed on 11.07.2024.

[81] Terraform registry. Azure association between a nat gateway and a public ip using terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/nat_gateway_public_ip_association. Accessed on 11.07.2024.

[82] Microsoft Learn. Overview - azure logic apps. https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview. Accessed on 11.07.2024.

[83] Terraform registry. Azure logic app workflow with terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/logic_app_workflow. Accessed on 11.07.2024.

[84] Terraform registry. Azure http request trigger within a logic app workflow using terraform. https://registry.terraform.io/providers/hashicorp/

azurerm/latest/docs/resources/logic_app_trigger_http_request. Accessed on 11.07.2024.

[85] Terraform registry. Azure http action within a logic app workflow using terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/logic_app_action_http. Accessed on 11.07.2024.

[86] Terraform registry. Azure metric alert within azure monitor using terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/monitor_metric_alert.html. Accessed on 11.07.2024.

[87] Terraform registry. Azure action group within azure monitor using terraform. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/monitor_action_group.html. Accessed on 11.07.2024.