Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Securización y Monitorización de una Solución Cloud con AWS

Autor: Julio Manso Sánchez-Tornero

Tutora: Sonia Valentina De Frutos Cid

Madrid, Julio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en ingeniería informática
Título: Securización y Monitorización de una Solución Cloud con AWS
Julio 2024

Autor: Julio Manso Sánchez-Tornero

Tutora:
Sonia Valentina De Frutos Cid
Departamento de Lenguajes y sistemas informáticos e ingeniería de software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado (TFG) titulado "Securización y Monitorización de una Solución Cloud con AWS" se centra principalmente en implementar medidas de seguridad y monitoreo en una infraestructura Cloud utilizando los servicios de Amazon Web Services (AWS).

Actualmente, el aumento exponencial del uso de Cloud Computing ha provocado la necesidad de garantizar la seguridad y el desempeño de los sistemas alojados en la nube. Este proyecto busca abordar estos desafíos mediante la configuración y gestión adecuada de servicios como Amazon VPC, IAM, KMS, CloudWatch y CloudTrail.

Por otro lado, el caso práctico desarrollado en este TFG se centra en la implementación de una biblioteca digital. Para ello, primero, se ha creado una base de datos en DynamoDB para almacenar la información de los libros y un bucket S3 para el almacenamiento de archivos digitales. Luego, se han configurado funciones Lambda para manejar las operaciones de la biblioteca, como la búsqueda y recuperación de libros, y por último, se ha establecido una API Gateway para gestionar las solicitudes externas. Esta infraestructura se securiza utilizando Amazon VPC para aislar los recursos en una red privada virtual, asegurando el acceso a través de endpoints específicos y enlaces VPC.

Para la gestión de identidades y accesos, se han utilizado políticas IAM que restringen el acceso a los recursos únicamente a usuarios y servicios autorizados. Además, se ha implementado el cifrado de datos con KMS para proteger la información almacenada en S3 y otros servicios. Asimismo, la monitorización se ha realizado configurando Amazon CloudWatch para supervisar el rendimiento y la disponibilidad de los servicios, y CloudTrail para registrar todas las llamadas a la API y cambios en la infraestructura.

Para finalizar, los resultados del caso práctico demuestran una mejora significativa en la seguridad y el control sobre los recursos Cloud, así como una mayor visibilidad de la actividad del sistema. La integración de CloudWatch y CloudTrail permite una monitorización continua y una respuesta eficiente a los incidentes de seguridad.

Abstract

This Final Degree Project (TFG) entitled "Securing and Monitoring a Cloud Solution with AWS" focuses mainly on implementing security and monitoring measures in a Cloud infrastructure using Amazon Web Services (AWS).

Currently, the exponential increase in the use of Cloud Computing has led to the need to ensure the security and performance of systems hosted in the cloud. This project seeks to address these challenges by properly configuring and managing services such as Amazon VPC, IAM, KMS, CloudWatch and CloudTrail.

On the other hand, the case study developed in this TFG focuses on the implementation of a digital library. For this, first, a DynamoDB database has been created to store the information of the books and an S3 bucket for the storage of digital files. Then, Lambda functions have been configured to handle library operations, such as book search and retrieval, and finally, an API Gateway has been set up to handle external requests. This infrastructure is secured using Amazon VPC to isolate resources in a virtual private network, securing access through specific endpoints and VPC links.

For identity and access management, IAM policies are used to restrict access to resources to authorized users and services only. In addition, data encryption with KMS has been implemented to protect the information stored in S3 and other services. Monitoring has also been done by configuring Amazon CloudWatch to monitor the performance and availability of services, and CloudTrail to log all API calls and infrastructure changes.

Finally, the results of the case study demonstrate a significant improvement in security and control over cloud resources, as well as increased visibility of system activity. The integration of CloudWatch and CloudTrail enables continuous monitoring and efficient response to security incidents.

Tabla de contenidos

1 Introducción

Hoy en día, el auge de las nuevas tecnologías, la globalización de los mercados y el aumento exponencial del uso de datos ha provocado un aumento exponencial de las necesidades de cómputo en las empresas, provocando que las capacidades de sus centros de datos (CPDs) no estén a la altura.

Por ello, muchas empresas se ven forzadas a cambiar de forma integral o parcial sus centros de datos, esto supone un gran gasto de capital, es decir, gastos que están dedicados para la compra de bienes de capital de una empresa. Ante la necesidad de reducir estos gastos, los cuales suponen grandes pérdidas para las empresas, nace una tecnología que permite el acceso remoto a software, máquinas de cómputo, almacenamiento de archivos y procesamiento de datos por medio de Internet, ofreciendo así, una alternativa a la ejecución en una computadora personal o un servidor local, lo que actualmente conocemos como cloud computing.

Es de vital importancia destacar un gran problema al que se enfrentan las empresas que desean migrar sus sistemas 'on-premise' a la nube.

El término 'on-premise' significa en español 'en las instalaciones propias' y se refiere a la utilización de servidores y entorno informático propios de la empresa. [1]

Como he explicado anteriormente, unos de los mayores problemas es la falta de seguridad en las infraestructuras de la nube por ello, es esencial que la dicha arquitectura sea altamente segura y se encuentre correctamente monitorizada. Con securizar nos referimos a asegurarnos que los sistemas informáticos situados en la nube se encuentren debidamente protegidos ante vulnerabilidades, además de que los datos mantengan su confidencialidad evitando posibles filtraciones. [2]

Por otro lado, respecto a la monitorización nos referimos a tener los datos siempre disponibles en tiempo real en la infraestructura para poder gestionarla de forma eficaz y poder adelantarnos a los posibles problemas que puedan surgir. [3]

Así pues, nos surge la pregunta; ¿Cómo monitorizamos y securizamos una solución cloud con AWS?. Este es el principal objetivo de este TFG.

En los últimos tiempos, el número de ciberataques que sufren las empresas ha aumentado de forma significativa. Se conoce que el número de ciberataques a la nube aumentó en un 95% en 2022. [4]

Asimismo, para poder securizar correctamente una solución cloud debemos de tener en cuenta varios aspectos como la seguridad de los datos para evitar posibles filtraciones de contenido sensible, la gestión de identidades y acceso, ya que es importante asegurarnos de que las entidades correctas (personas o cosas) accedan a los recursos adecuados cuando sean necesarios y exclusivamente si se les permite el acceso.

Y por último, la configuración en la nube debido a que es el moyor riesgo de todos ya que deja la puerta abierta a posibles amenazas que se podrían evitar con una correcta configuración.

1.1 Motivación y objetivos

El uso de la nube para adoptar soluciones por parte de las empresas está en auge debido a las facilidades que ofrece. Entre las que destacan la flexibilidad y fiabilidad, la reducción de costes para las empresas y además, mejora la innovación debido a que las empresas reducen el tiempo de lanzamiento de sus soluciones.

Sin embargo, usar la nube no resulta tan sencillo como parece y requiere una serie de retos a los que se enfrentan las empresas en términos de monitorización y securización de la nube. Ya que, detectar amenazas de forma previa y poder asegurar la protección de datos sensibles, son unas de las principales preocupaciones de las empresas.

Por ello, este TFG tratará de abarcar todos los aspectos que se deben de tener en cuenta para cumplir los requisitos de seguridad y monitorización en la nube.

Algunos de los objetivos más concretos son:

- Estudio de servicios cloud AWS de seguridad y monitorización.
- Acceso seguro a objetos almacenados en cloud
- Acceso de red seguro a la red privada virtual
- Cifrado de datos en volúmenes de datos
- Gestión de claves de cifrado
- Creación de sistema de monitorización y respuesta a incidentes
- Automatización de la infraestructura y servicios cloud
- Estimación de costes

1.2 Planificación

Para realizar los objetivos explicados anteriormente se han establecido nueve tareas principales para el plan de trabajo.

T1: Estudiar servicios cloud AWS de seguridad y monitorización.

T2: Planificar el diseño de la arquitectura y estimar los costes.

T3: Implementación de la arquitectura necesaria.

T4: Securizar datos en Amazon S3.

T5: Securizar la red privada virtual.

T6: Securizar el acceso a los recursos AWS usando servicios de gestión de claves.

T7: Monitorizar y registrar los accesos.

T8: Automatizar la creación de infraestructura y servicios de AWS utilizados.

T9: Documentación del proyecto.

2 Estado del arte

Este capítulo se centrará en afianzar el concepto de cloud computing así como sus principales características, así como los distintos roles y tipologías que pueden tener los proveedores con respecto a la clasificación de los modelos operativos basados en el paradigma del Cloud Computing. También, se entrará en detalle acerca de los conceptos básicos y las buenas prácticas que debemos tener en cuenta a la hora de securizar y monitorizar nuestro entorno en la nube. Por último, se hará hincapié en los recursos de la nube utilizados.

Tras explicar en el apartado anterior, el planteamiento del problema que abarca este TFG nos surge un nuevo concepto, el cloud computing. Lo podemos definir como un paradigma en el que un tercero ofrece una serie de recursos a disposición de los usuarios a través de internet. Esto permite no tener que usar hardware ni infraestructura propia sino que podemos acceder a máquinas virtuales, cuentas de almacenamiento a través de muestreo proveedor de nube. [5]

Tomando como base la definición anterior, de forma mucho más sencilla podemos definir la nube como un espacio físico de almacenamiento, por ejemplo un disco duro, pero no está en nuestro ordenador o en nuestro centro de datos sino que podemos acceder al mismo a través de internet. Además, nos ofrece servicios muy variados, desde un 'backup' para tus archivos, servicios para ejecutar aplicaciones como por ejemplo un programa de ofimática, como el Word, pero sin tener que estar instalado en tu ordenador.[6]

Por otro lado, existen muchos los proveedores que te ofrecen "servicios cloud". Los proveedores nos proporcionan infraestructura, aplicaciones y almacenamiento a través de una red basada en la nube. Los más conocidos son Microsoft Azure, AWS (Amazon Web Services) y Google Cloud. Para este TFG se ha decidido usar AWS ya que es un proveedor que tiene mucha experiencia en el mercado gracias a la variedad de servicios que ofrece.

2.1 Propiedades básicas del cloud computing

En primer lugar, para poder entender mejor el concepto del cloud computing, se va a detallar aquellas características que diferencian el cloud computing de otras metodologías. De acuerdo con el NIST (National Institute of Standards and Technology) consta de cinco características esenciales:

- **Demanda de autoservicio:** Podemos abastecer nuestras necesidades de cómputo, de redes o almacenamiento siempre que deseemos, sin necesidad de intervención manual del proveedor del servicio.

  Por ejemplo, si tenemos una página web alojada en la nube, existen periodos del día donde el tráfico de red en la web es bastante más

elevado por lo que podemos asignar más cómputo a la web para subsanar esos momentos. O por ejemplo, si tenemos una cuenta en la nube con 1TB de almacenamiento pero llegamos al límite de capacidad, siempre podremos migrar a un plan superior donde exista más almacenamiento.

- **Amplio acceso de red:** Con la existencia de la computación en la nube ya no será necesario tener equipos informáticos o centros de datos propios ya que podremos acceder a las capacidades de la nube a través de internet. También, podemos acceder a la nube a través de distintas plataformas, como el móvil o el ordenador, así como desde cualquier lugar del mundo con acceso a internet, ya que la nube nos proporcionará una consola de comandos o una interfaz web para poder acceder.
- **Pool de recursos:** El proveedor dispone de una serie de recursos informáticos que atienden a múltiples consumidores a través de un modelo donde los recursos virtuales y físicos se asignan dinámicamente y se reasignan según las demandas individuales de cada cliente.
- **Rápida elasticidad:** Con el término de elasticidad en la nube nos referimos a que los servicios son capaces de agregar o eliminar recursos según necesidad, ajustándose siempre a la cantidad de recursos justa que necesitamos en cada instante. [7]

  Por lo que, los recursos de cómputo no están limitados a una capacidad estática.

- **Medición de servicios:** Hace alusión a la capacidad que mantienen los servicios de nube de controlar y optimizar el uso de los recursos a través de la supervisión, medición e información de atributos en la nube.

  Por ejemplo, la capacidad de procesamiento y almacenamiento, el ancho de banda de la red y la cantidad de cuentas de usuario activas que se controlan automáticamente.

  Por lo tanto, esta característica ayuda a determinar el rendimiento de los recursos y es crucial en este modelo tecnológico ya que todos los recursos están preparados para obtener el resultado final optimizado.

Otras de las características que también podemos considerar son las siguientes:

- **Seguridad:** Los proveedores de la nube tienen la capacidad de dedicar recursos a la seguridad en aspectos que no se cubrirán si fuese la responsabilidad del cliente.

  Por otro lado, podemos afirmar que el proveedor de nube será responsable de la seguridad física y el cliente de la seguridad a nivel de aplicación.

- **Costo:** Este se reduce considerablemente ya que eliminamos los gastos de capital, el dinero gastado en la compra de bienes de capital de una empresa como por ejemplo, las máquinas de un centro de datos.

Por otro lado, añadimos los gastos operativos, el capital utilizado para mantener o mejorar los activos físicos de una empresa.

- **Mantenimiento:** El mantenimiento necesario es el personal capacitado para manejar los servicios de nube. Eliminando así otros gastos de personal como por ejemplo, el personal encargado del mantenimiento físico de los servidores.

[8]

2.1.1 Modelos de computación existentes

Por otro lado, existen tres tipos de modelos diferentes en los que se despliegan las soluciones cloud. Estos son: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS), y Software como Servicio (SaaS).

En la siguiente figura podemos observar los tres modelos mencionados previamente. El primero de ellos (IaaS) podría asemejarse a la infraestructura, el segundo (Paas) podría tratarse de la plataforma y el último de ellos (SaaS) sería una aplicación funcional y terminada, como Microsoft Word.

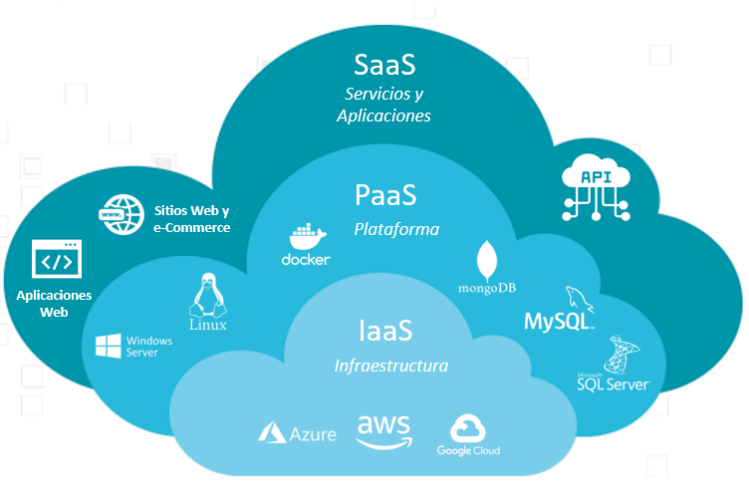

Figura 1. Modelos de la nube existentes

Algunas de las características de los modelos son las siguientes:

Infraestructura como servicio (IaaS):

Este tipo de servicio ofrece a las empresas recursos de infraestructura según demanda, como computación, almacenamiento, redes y virtualización, todos accesibles a través de la nube. Aunque los clientes son responsables del sistema operativo, middleware, máquinas virtuales y cualquier aplicación o dato específico, no tienen la carga de administrar, mantener y actualizar su propia infraestructura de centro de datos.

Por ejemplo, AWS proporciona máquinas virtuales donde los usuarios deben instalar el sistema operativo y el software necesario para sus aplicaciones.

Plataforma como servicio (PaaS):

Para el desarrollo de aplicaciones en la nube, la plataforma como servicio (PaaS) proporciona y gestiona todos los recursos de hardware y software necesarios. Con PaaS, tanto los equipos de operaciones de TI como los desarrolladores pueden crear, ejecutar y gestionar aplicaciones sin tener que construir y mantener la infraestructura o plataforma por sí mismos. El proveedor de servicios en la nube se encarga del entorno de desarrollo y despliegue de aplicaciones, mientras que los clientes son responsables de escribir el código y administrar sus datos y aplicaciones.

Software como servicio (SaaS):

Para ofrecer a los clientes una aplicación completa que se encuentre basada en la nube, el software como servicio (SaaS) proporciona toda la pila de aplicaciones como por ejemplo, Microsoft Excel. El proveedor de servicios gestiona completamente los productos de SaaS y los ofrece para su uso. Además, incluyen todos los mantenimientos, correcciones de errores y actualizaciones. Los clientes no necesitan descargar ni instalar aplicaciones en sus dispositivos para acceder directamente a la mayoría de las aplicaciones SaaS.

[9]

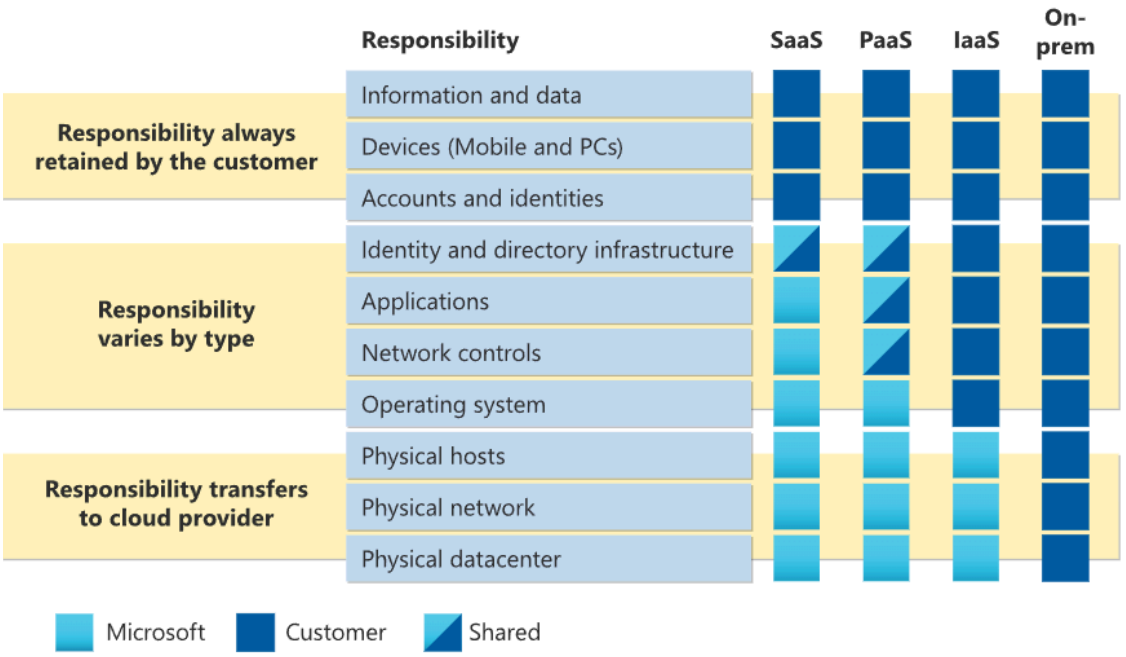

Figura 2. Modelo de responsabilidad compartida

Para finalizar este apartado, como se puede observar en la figura anterior, existe un modelo de responsabilidad compartida entre el cliente y el proveedor. El modelo SaaS es el que sitúa la mayor responsabilidad en el proveedor de nube y la menor responsabilidad en el usuario, sin embargo, este será

responsable de la información y tratamiento de datos, de los dispositivos que le permitirán conectarse al sistema y de los usuarios que tendrán acceso.

Por otra parte, el modelo PaaS se encuentra en un punto intermedio, donde el cliente tendrá responsabilidad en las aplicaciones que instale o en el tratamiento de los datos. Pero, el proveedor de nube se encargará de todo el resto.

Por último, el modelo IaaS supone la mayor responsabilidad al cliente ya que este se encargará del sistema operativo y de su mantenimiento y de las aplicaciones, entre otros.

2.1.2 Modelos de despliegue en la nube

Podemos diferenciar diversas formas de integrar y explotar los servicios ofrecidos por la nube:

Nube pública:

Las nubes públicas son el tipo más común de implementación de informática en la nube. En este modelo, un proveedor de servicios en la nube gestiona y distribuye recursos como servidores y almacenamiento. El proveedor de la nube pública posee y administra todo el hardware, software y otros componentes de la infraestructura subyacente. Algunos ejemplos de proveedores de nube pública son Microsoft Azure, Google Cloud y AWS.

En una nube pública, el mismo hardware, almacenamiento y dispositivos de red se comparten entre varias organizaciones o "inquilinos" de la nube. Además, puedes administrar tu cuenta a través de un navegador web o una aplicación móvil. Las implementaciones de nube pública se utilizan frecuentemente para servicios como correos electrónicos web, aplicaciones de Office en línea, almacenamiento y entornos de desarrollo y prueba.

Ventajas de las nubes públicas:

- **Menores costos:** No se requiere comprar hardware ni software, y se paga solo por el servicio utilizado.
- **Sin necesidad de mantenimiento:** El proveedor de servicios se encarga de todas las tareas de mantenimiento.
- **Escalabilidad casi ilimitada:** Los recursos están disponibles bajo demanda para satisfacer las necesidades empresariales, permitiendo una expansión prácticamente sin límites.
- **Alta fiabilidad:** Debido a la extensa red de servidores del proveedor, se garantiza una alta fiabilidad y disponibilidad, minimizando posibles problemas.

Nube privada:

Una nube privada consiste en recursos informáticos en la nube que son exclusivamente utilizados por una sola empresa u organización. Esta infraestructura puede estar ubicada físicamente en el centro de datos de la propia organización o alojada en una nube privada proporcionada por un proveedor de servicios externo. En ambos casos, los recursos de la nube privada operan en una red privada dedicada, donde tanto el hardware como el software son de propiedad y uso exclusivo de la organización.

La principal característica de una nube privada es que permite a la empresa personalizar y adaptar los recursos para cumplir con requisitos específicos de TI y de seguridad. Por esta razón, las agencias gubernamentales, las instituciones financieras y cualquier organización mediana o grande que realice operaciones críticas y busque aumentar el control sobre su entorno suelen optar por implementar nubes privadas..

Las ventajas de una nube privada incluyen la capacidad de adaptar el entorno en la nube según las necesidades específicas de la empresa, ofreciendo mayor flexibilidad. Además, proporciona un nivel superior de control y privacidad, dado que los recursos no se comparten con otros usuarios. En términos de escalabilidad, las nubes privadas generalmente ofrecen una capacidad de crecimiento más amplia en comparación con las infraestructuras locales tradicionales.

Nube híbrida:

Una nube híbrida integra tanto la nube pública como la nube privada, permitiendo el movimiento de datos y aplicaciones entre ambos entornos. Muchas organizaciones eligen la nube híbrida para satisfacer necesidades comerciales específicas, como el cumplimiento normativo, la soberanía de los datos y la gestión de problemas de latencia.

Las cargas de trabajo perimetrales también forman parte de la estrategia de nube híbrida. Este enfoque lleva la capacidad informática de la nube a los dispositivos de Internet de las cosas (IoT), acercando el procesamiento al lugar donde se generan y residen los datos. Esta proximidad reduce la latencia al minimizar el tiempo que los dispositivos necesitan para comunicarse con la nube. Además, permite que los dispositivos funcionen de manera confiable durante períodos prolongados sin conexión.

[10]

2.2 Securización en AWS

Una vez explicados los conceptos básicos de cloud computing. En este presente capítulo se va a explicar la securización en cloud.

Para empezar, la seguridad en la nube es similar a la de los centros de datos locales, pero sin el coste y la complejidad de proteger las instalaciones y el hardware. Dado que en la nube no hay servidores físicos ni dispositivos de almacenamiento, se utilizarán herramientas de seguridad basadas en software para monitorizar y proteger el flujo de información que entrada y salida de los recursos de AWS.

Además, en la nube se pueden utilizar los mismos modelos de seguridad que se usan actualmente en su entorno. Esto incluye proporcionar visibilidad, capacidad de auditoría y control de sus recursos en la nube.

Los principales objetivos en la seguridad cloud en AWS son los siguientes:

- **Controllability (Controlabilidad):**

  AWS proporciona métodos y herramientas para administrar el control de acceso de usuarios, grupos y roles, proporcionar credenciales de seguridad temporales y controlar las claves de cifrado.

  Para ello, existe el servicio de AWS Identity and Access Management (IAM) que permite controlar quién puede utilizar sus recursos de AWS (autenticación), qué recursos pueden utilizar y de qué manera (autorización).

- **Auditiability (Auditoría):**

  Es esencial probar y validar las defensas para garantizar el cumplimiento de los requisitos de conformidad. Las organizaciones suelen realizar auditorías periódicas de sus entornos, tanto internas como externas, para garantizar el cumplimiento de las políticas y normativas.

  Además, existen servicios de AWS como CloudTrail que nos ayudan a responder preguntas como; ¿Qué acciones realizó un usuario específico en un periodo de tiempo determinado? o ¿Cuál es la dirección IP de origen de una actividad concreta?

- **Visibility (Visiblidad):**

  Conocer nuestros activos es el primer paso para poder protegerlos.

  Para tener visibilidad instantánea de la actividad de los usuarios y las aplicaciones, AWS proporciona herramientas para rastrear y monitorizar los recursos. Por ejemplo, AWS Config nos permite descubrir los recursos de AWS existentes, exportar un inventario completo de sus recursos con todos los detalles de configuración y determinar en cómo se configuró un recurso.

  Estas habilidades pueden ayudarnos con la auditoría de conformidad, el análisis de seguridad, el seguimiento y la solución de problemas de recursos.

- **Agility and automation (Agilidad y automatización):**

  AWS nos proporciona agilidad bajo demanda con escalado automático para garantizar una entera disponibilidad durante posibles ataques de seguridad.

  También, se encarga de diseñar centros de datos con exceso de ancho de banda para equilibrar la carga de tráfico en caso de interrupciones y minimizar el impacto en los clientes.

  Asimismo, los clientes pueden usar diferentes estrategias multiregión y multizona para crear aplicaciones resistentes, replicar datos y aplicar controles de seguridad globales de forma consistente.

  AWS permite automatizar tareas rutinarias de seguridad, permitiendo que sus expertos se concentren en mejorar la seguridad del entorno en la nube. Además, permite a los clientes automatizar la arquitectura mediante AWS CloudFormation, para crear entornos de forma segura y repetible.

Una vez explicados los principales objetivos que debemos de cumplir cuando desarrollamos cualquier tipo de infraestructura en AWS. A continuación, se explicarán los principios de diseño de seguridad que debemos de seguir. También, conocidos como AWS Well-Architected Framework.

Estos principios son los siguientes:

- **Implementar una base de identidad sólida:** Implementar el principio de mínimo privilegio y aplicar la separación de funciones con la autorización adecuada para cada interacción con los recursos de AWS. Centralizar la gestión de identidades y eliminar la dependencia de credenciales estáticas a largo plazo.
- **Mantener la trazabilidad:** Para ello es necesario supervisar y alertar las acciones y los cambios en el entorno en tiempo real. Además, de integrar la recopilación de registros y métricas con sistemas para investigar y tomar medidas de forma automática.
- **Aplicar la seguridad en todas las capas:** Es necesario aplicar un enfoque de defensa en profundidad con múltiples controles de seguridad, a todas las capas.
- **Automatizar las prácticas de seguridad:** Los mecanismos de seguridad automatizados mejoran la capacidad para escalar de forma segura con mayor rapidez. Crear arquitecturas seguras, incluida la implementación de controles que se definen y gestionan como código en plantillas controladas por versiones.
- **Proteger los datos en tránsito y en reposo:** Se dividen los datos en niveles de sensibilidad y se utilizan mecanismos como el cifrado, la tokenización y el control de acceso.
- **Mantener a otros usuarios alejados de los datos:** Para ello, se utilizan mecanismos y herramientas para reducir o eliminar la necesidad de acceso directo o procesamiento manual de los datos para reducir el riesgo de manipulación o modificación indebidas y de error humano al manejar datos sensibles.

- **Preparar posibles incidentes de seguridad:** Para ello es aconsejable realizar simulaciones de respuesta a incidentes y utilizar herramientas con automatización para aumentar la velocidad de detección, investigación y recuperación.

[11]

Para finalizar este capítulo, en la siguiente figura se puede apreciar cada uno de los principios del AWS Well-Architected Framework con los respectivos servicios que nos ofrece AWS para cada uno de ellos. Algunos de estos servicios se explicarán con detalle más adelante en los siguientes apartados.

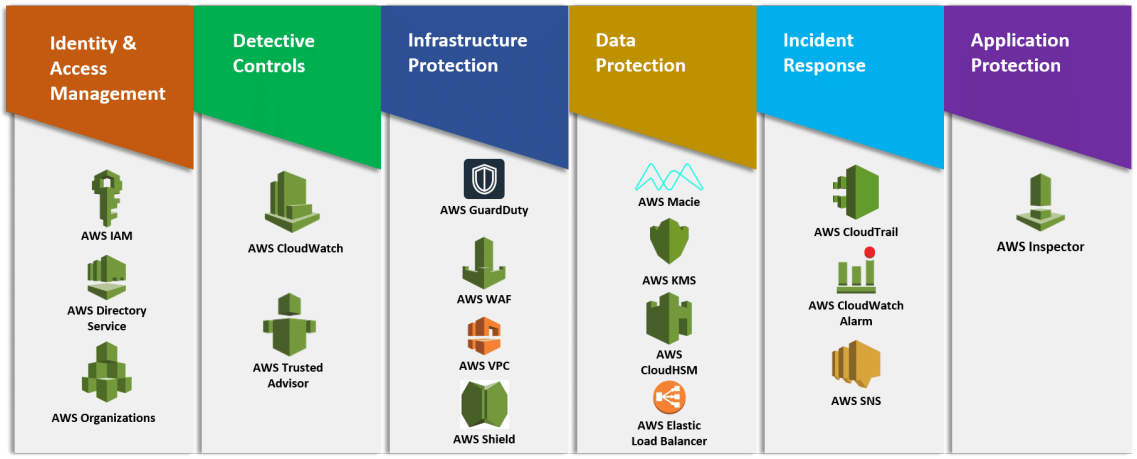

Figura 3. Servicios Well Architected Framework

2.2.1 Identity and Access Management IAM

El servicio de AWS Identity and Access Management (IAM) es esencial para administrar y controlar el acceso a los servicios y recursos de AWS dentro de una organización.

IAM se fundamenta en varios conceptos clave:

- El usuario raíz AWS es la identidad inicial cuando se crea una cuenta de AWS, con acceso total a todos los recursos mediante la consola utilizando una dirección de correo electrónico y contraseña.
- Los usuarios IAM representan personas o servicios dentro de la organización que interactúan con AWS. Pueden acceder a los recursos a través de la consola de administración con nombre y contraseña, o mediante API y CLI con claves de acceso generadas.
- Los grupos de usuarios IAM son conjuntos lógicos de usuarios que simplifican la administración al aplicar permisos de forma colectiva. Por ejemplo, un grupo "admins" puede tener todos los permisos de administración, y los usuarios dentro del grupo heredan estos permisos automáticamente.
- Los roles IAM son similares a los usuarios IAM, sin embargo, no tienen credenciales propias. Se utilizan para definir qué acciones pueden realizar las identidades dentro de AWS, y pueden ser asumidos

temporalmente por usuarios para realizar tareas específicas con permisos específicos.

- Las políticas de IAM definen los permisos precisos sobre recursos de AWS. Estas políticas, escritas en formato JSON, especifican las acciones permitidas (por ejemplo, GetUser) y los recursos a los que se aplican, estableciendo reglas claras para el acceso y la gestión de recursos en AWS.

[12]

Por finalizar, en la siguiente figura se aprecia la jerarquía de los servicios que ofrece IAM.

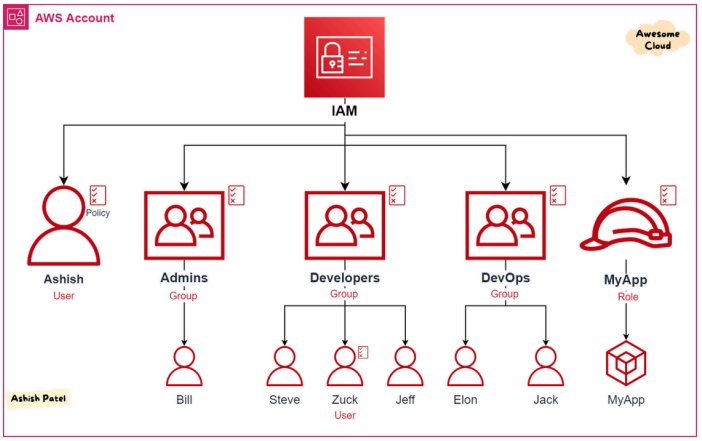

Figura 4. Componentes IAM

2.2.2 Amazon Virtual Private Cloud VPC

Amazon Virtual Private Cloud (Amazon VPC) es un servicio central de AWS que proporciona control completo sobre entornos de redes virtuales en la nube. Al configurar una VPC, se pueden definir y gestionar secciones aisladas de la infraestructura de AWS para desplegar recursos de manera segura.

Una de las funcionalidades destacadas de Amazon VPC es la capacidad de especificar rangos de direcciones IP privadas para los recursos dentro de la VPC. Esto facilita la segmentación lógica de los recursos, permitiendo la comunicación mediante direcciones IP privadas mientras se mantiene el aislamiento de otros recursos y VPCs.

Amazon VPC ofrece varias capacidades esenciales para configurar y administrar redes:

- **Segmentación de Red:** Posibilita la creación de subredes dentro de la VPC, lo que mejora la organización y segmentación de los recursos.
- **Control de Acceso:** Utiliza grupos de seguridad y listas de control de acceso de red (Network ACLs) para administrar el tráfico permitido o restringido hacia y desde las instancias dentro de la VPC.
- **Conexiones Privadas y Públicas:** Permite configurar subredes públicas con acceso a Internet y subredes privadas sin acceso directo a la red

pública, utilizando servicios NAT (Network Address Translation) para la conectividad desde las subredes privadas.

- **Conexiones VPN y Direct Connect:** Permite establecer conexiones seguras entre la VPC y la red local a través de VPN o AWS Direct Connect, facilitando la integración híbrida entre infraestructuras locales y la nube de AWS.
- **Escalabilidad y Disponibilidad:** Diseñado para ser altamente escalable y disponible, Amazon VPC admite la creación de múltiples VPCs en diferentes regiones de AWS para garantizar redundancia y alta disponibilidad.

[13]

2.2.3 Key Management Service KMS

KMS es un servicio administrado de AWS que permite crear y administrar claves para el cifrado de los datos. Además, KMS se encuentra integrado en mayoría de servicios AWS como por ejemplo, DynamoDB o Bucket S3.

Por otro lado, se pueden crear, editar y eliminar claves de cifrado tanto simétricas como asimétricas. Asimismo, se puede habilitar políticas de acceso a las claves para verificar que servicios y usuarios tienen acceso a ellas. También permite habilitar y deshabilitar las claves según convenga. [14]

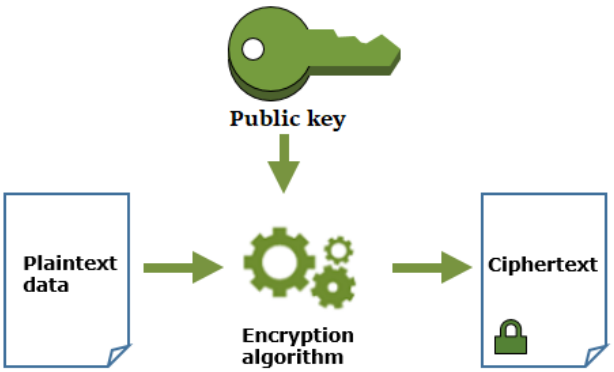

Figura 5. Modelo de uso KMS

Por último, en la figura anterior se aprecia que la forma de utilizar una clave simétrica en KMS es la siguiente: Primero, se genera una clave de cifrado simétrico dentro de AWS KMS, donde se almacena de forma segura. Esta clave se utiliza luego para cifrar datos enviados al servicio KMS mediante una solicitud de cifrado. Una vez cifrados, los datos resultantes (texto cifrado) pueden almacenarse en sistemas como Amazon S3 o bases de datos, manteniendo la información original protegida. Cuando se requiere acceder a estos datos cifrados, se envía el texto cifrado junto con una solicitud de descifrado a AWS KMS, que utiliza la misma clave simétrica para descifrar los datos y devolver la información en su formato original.

2.3 Monitorización en AWS

Este capítulo tratará de entender la importancia de la monitorización en la nube y ayudará a comprender cuáles son las mejores prácticas de monitorización.

A lo largo de la historia, el ser humano ha tratado de documentar, registrar y grabar todas las acciones y progresos realizados y existen una principal razón de ello; Poder tomar decisiones en base a los datos recopilados.

Usando los conocimientos explicados anteriormente como pretexto, los sistemas informáticos que usamos actualmente no son 100% eficientes, existen ocasiones en las que estos sistemas fallan o no funcionan de forma prevista. Y la única forma de evitar estos problemas sería monitorizar estos sistemas informáticos permitiéndonos así predecir y entender posibles errores y de esta forma, minimizar los fallos del sistema al máximo. [15]

Una vez dicho esto, existen dos tipos de monitorización;

En primer lugar, tenemos la monitorización **proactiva**. En ocasiones podemos averiguar si existe algún problema de antemano ya que la infraestructura AWS nos proporcionará avisos o señales de advertencia. De esta forma, con este tipo de monitorización podremos anticiparnos a los fallos antes de que ocurran, usando por ejemplo, alertas por correo electrónico y permitiéndonos así, actuar en base a estas alertas. Por ejemplo, si tenemos una página web y recibimos una alerta de que la unidad de cómputo (CPU) está con una carga de trabajo superior al 90%, podremos concluir que hay una anomalía con esa CPU y gracias a ello, podremos investigar y encontrar el origen del problema.

Por otra parte, tenemos la monitorización **reactiva**. Este tipo de monitorización actúa una vez ya se ha producido el problema, ignorando de esta forma cualquier tipo de alerta o aviso que nos haya podido proporcionar la infraestructura con anterioridad. Asimismo, si usamos este tipo de monitorización, provocaremos fallos, tanto de breve o larga duración.

Por todo ello, este tipo de monitorización, reactiva, no es la más adecuada para su uso en una infraestructura en la nube, la más adecuada será la monitorización proactiva, ya que nos permitirá usar herramientas proporcionadas por AWS, como Amazon CloudWatch, para poder generar alertas y anticiparnos ante posibles problemas.

[16]

Los componentes necesarios para poder realizar una buena monitorización son los siguientes:

- **Alertas:** Son eventos que se activan generalmente cuando hay un problema. Esto es debido a que las alertas se configuran en base a una métrica y en el caso de que se cumpla esa métrica se manda un notificación a la persona correspondiente. Estas alertas, se suelen notificar vía correo electrónico, y en el caso de AWS nos proporciona Amazon SNS que es un servicio el cuál nos permitirá mandar mensajes de los publicadores a los suscriptores. Además, la comunicación entre

los publicadores y consumidores es mediante el envío mensajes a un punto de acceso lógico que actúa como tema y un canal de comunicación de forma asíncrona. [17] Este proceso se puede visualizar en la figura adjunta abajo.

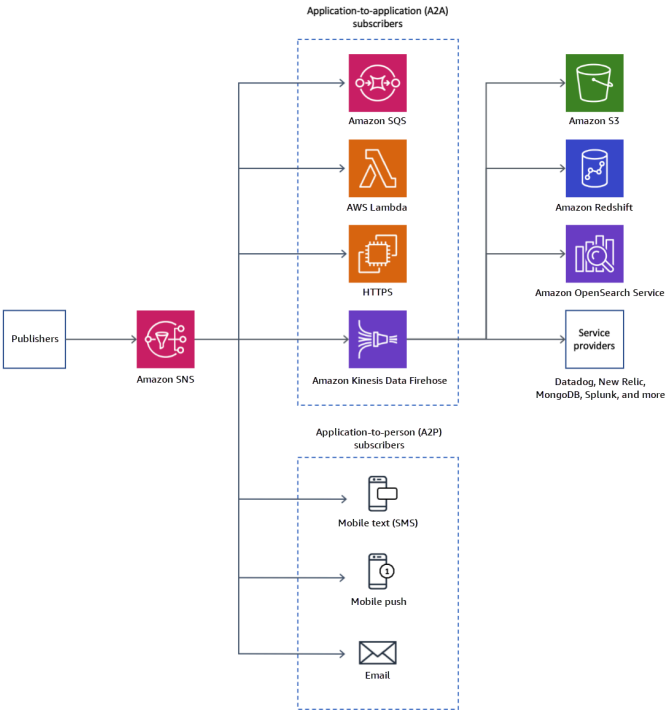

Figura 6. Arquitectura Servicios SNS

- **Eventos:** *"cualquier tipo de acción, actividad o grupos de actividades en un sistema se denomina evento"*. [15] Por ejemplo, un reloj ya que siempre habrá un proceso en segundo plano para poder marcar la hora correctamente.
- **Registros:** En un registro tenemos anotaciones de los acontecimiento que se han producido. Un registro no sólo contiene los eventos y detalles del evento, sino también el momento en que se produjo.
- **Métricas:** *"Una métrica es la unidad más pequeña de información obtenida de un registro"* [15]. Las métricas son capaces de dar sentido a los registros que son recopilados por el sistema. Además, proporcionan un estándar de medida para los distintos componentes del sistema.

[15]

En el caso de AWS, proporciona una serie de herramientas para poder monitorizar la infraestructura existente, permitiendo comprobar el estado y rendimiento de la misma en cualquier momento.

Asimismo, AWS nos proporciona la capacidad de monitorizar el estado de nuestros recursos, dándonos una vista completa del estado en una sola interfaz. Además, nos permite detectar de forma automática cualquier cambio que se realice en nuestra infraestructura AWS como por ejemplo la adición o eliminación de recursos. También nos permite filtrar por etiquetas, es decir, solo monitorizar los recursos que tenga cierta etiqueta, como por ejemplo EC2,

que sería en este caso máquinas virtuales. Además, nos permite hacer una análisis de rendimiento, nos permite detectar anomalías y desviaciones de las cargas de trabajo de nuestros recursos de esta forma podremos planificar de forma eficaz la adjudicación de recursos para mejorar las cargas de trabajo. Gracias a esto podremos utilizar de forma rentable los recursos en AWS.

Por último, tenemos la capacidad de no solo poder monitorizar la infraestructura en AWS sino también en otras plataformas como Microsoft Azure.

[18]

2.3.1 Amazon CloudWatch

Para resolver el problema de cómo medir una infinidad de métricas en tiempo real de los recursos que tengamos desplegados en AWS, surge Amazon CloudWatch, un servicio que ofrece Amazon, en que se permite recopilar y realizar un seguimiento de las métricas de los recursos.

Algunos de los recursos que podemos monitorizar son, instancias de Amazon EC2, balanceadores de carga y bases de datos de Amazon RDS, entre otros. [18]

En la siguiente figura, se pueden observar los pasos de funcionamiento de Amazon CloudWatch.

Figura 7. Funcionalidades Amazon CloudWatch

Comenzando por una recolección de métricas y registros, seguido de una monitorización en la que podemos observar de forma gráfica dichas métricas recolectadas. Posteriormente, podemos automatizar algunas acciones antes cambios que puedan surgir, como una aumento en la demanda de computo. Por último, nos permite retener datos a largo plazo lo que nos permite realizar un análisis histórico y la identificación de patrones a lo largo del tiempo. En la siguiente figura se aprecia la metodología de CloudWatch.

Además, Amazon CloudWatch presenta cuadros de mando (dashboards) en los que podemos visualizar todos nuestros recursos en un mismo sitio, aunque los recursos estén ubicados en diferentes regiones del mundo. Asimismo, nos permite poder crear vistas personalizadas de las métricas que más nos convengan. Por ejemplo, cambiando el color de cada gráfico o cambiando su

distribución. Gracias a ello, nos permite evaluar el estado de las métricas de forma eficaz y sencilla. [19]

Por otro lado, otra de las funciones que tiene CloudWatch es el uso de alarmas para supervisar métricas. Tenemos dos tipos de alarmas:

- **Alarmas de métricas:** Nos permiten inspeccionar una sola métrica o una expresión matemática que sea resultado de varias métricas diferentes. Y además, podemos definir un umbral, y dependiendo del valor de la métrica o expresión respecto al umbral hará una acción u otra.
- **Alarmas compuestas:** Componen una alarma de alarmas, es decir, una expresión de regla que tiene en cuenta otros estados de alarmas ya creadas. Hay que tener en cuenta de que esta alarma solo se activa si se activan todas las alarmas subyacentes. Y además, solo son útiles si queremos recibir una notificación cuando se activen todas las alarmas y no solo cuando se activen algunas o una.

Las alarmas explicadas anteriormente, solo pueden tener tres estados:

- **OK:** Estará presente cuando la métrica o expresión se encuentra dentro de los límites establecidos
- **ALARM:** Estará presente cuando la expresión o métrica se encuentra fuera de los límites.
- **INSUFICCENT_DATA:** Existen varios casos, por ejemplo que no existan suficientes datos o que se esté iniciando la alarma.

Para finalizar este apartado, Amazon CloudWatch evalúa las métricas para activar la alarma de la siguiente forma;

Para comenzar, se tienen que rellenar tres parámetros. El primero es el **periodo**, el tiempo que necesitamos para evaluar la métrica o expresión, el segundo es el periodo de **evaluación**, el número de periodos que se tienen en cuenta para hacer la evaluación y por último, el tercero son los **puntos de datos** para alarmas, el número de datos en periodo de evaluación que se deben de estar por encima de umbral para que se active la alarma.

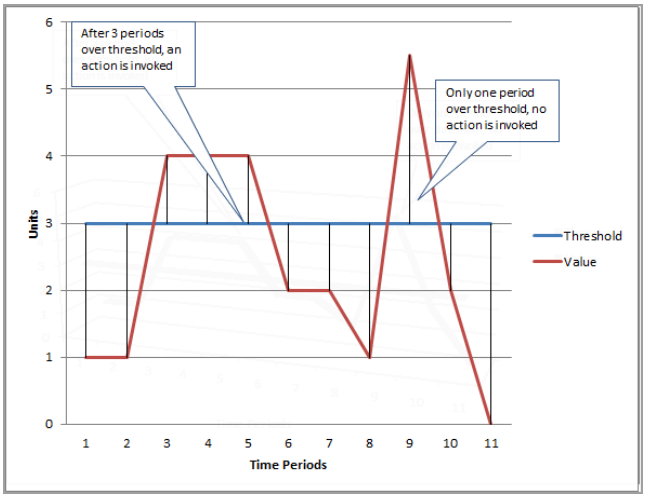

Figura 8. Gráfica de una métrica

Como se aprecia en este caso, el periodo de evaluación es tres por lo que solo se evaluarán los tres últimos periodos. Los puntos de datos también son tres, por lo que, la alarma solo se activará cuando haya tres periodos seguidos que superen el umbral.

[20]

2.3.2 Amazon CloudTrail

Es un servicio en el que podemos registrar de forma detallada toda la información sobre actividades realizadas por los usuarios, servicios y recursos dentro de la cuenta de AWS. Estos eventos pueden ser la creación o modificación de instancias EC2 o creación de bases de datos, entre otros.

Además, Amazon CloudTrail almacena todos los registros de auditoría en bucket de Amazon S3. Estos registros están organizados por regiones y se pueden observar en formato JSON. [21]

En la siguiente figura se aprecia un ejemplo del contenido de estos ficheros.

Figura 9. Registro de Amazon CloudTrail

El significado de cada uno de los componentes es el siguiente:

- **Records:** Supone el arreglo principal que contiene todos los registros de eventos. En este caso, solo hay un registro dentro del arreglo.
- **EventVersion:** La versión del formato del evento.

- **UserIdentity:** Contiene información sobre la identidad del usuario que realizó la acción.
- **Type:** El tipo de identidad. En este caso, es un usuario IAM.
- **PrincipalId:** Un identificador único para la identidad principal.
- **Arn:** El ARN (Amazon Resource Name) de la identidad del usuario.
- **AccessKeyId:** El ID de clave de acceso utilizado por el usuario.
- **AccountId:** El ID de cuenta de AWS del usuario.
- **UserName:** El nombre del usuario IAM.
- **EventTime:** La marca de tiempo en la que ocurrió el evento.
- **EventSource:** La fuente del evento. En este caso, es el servicio EC2 de AWS.
- **EventName:** El nombre del evento. Aquí, "StartInstances" indica que se iniciaron instancias EC2.
- **AwsRegion:** La región de AWS donde se realizó el evento.
- **SourceIPAddress:** La dirección IP desde la que se originó la solicitud.
- **UserAgent:** El agente de usuario que realizó la solicitud.
- **RequestParameters:** Los parámetros de la solicitud. Aquí, se especifica la instancia que se inició.
- **ResponseElements:** Los elementos de respuesta del evento. Aquí, se proporciona información sobre el estado actual y el estado anterior de la instancia iniciada.

Para concluir, Cloudtrail también nos permite una visibilidad multi-región, capturando así eventos en todas las regiones de AWS.

También, permite centralizar registros de múltiples cuentas en un solo bucket de S3, simplificando la gestión y el análisis. Además de registrar acciones, CloudTrail documenta datos de recursos afectados, como el ID de una instancia EC2. Se integra con servicios como CloudWatch e IAM para alertas y control de acceso. Los registros se almacenan en S3 y pueden retenerse según necesidades de cumplimiento, con seguridad garantizada mediante encriptación. Los registros JSON permiten búsqueda y análisis fácil, compatible con AWS Organizations para una trazabilidad organizacional completa. [21]

2.4 Otros Recursos utilizados

En este capítulo se explicarán otros servicios que se usarán para realizar el caso práctico.

2.4.1 Bucket S3

El servicio de almacenamiento de objetos Amazon Simple Storage Service (Amazon S3) se distingue por su capacidad para escalar, su alta disponibilidad de datos, su seguridad robusta y su excelente rendimiento.

Además de que Amazon S3 puede ser utilizado por cualquier usuario para almacenar y proteger información o para una gran variedad de usos, como sitios web y aplicaciones móviles.

Asimismo, gracias a las funciones de gestión de Amazon S3 que permiten optimizar, organizar y configurar el acceso a los datos se consigue satisfacer las necesidades de los usuarios.

En cuanto a los datos que se almacenan en los bucket S3 se conocen como objetos. Un objeto es la unidad de almacenamiento fundamental, que consta de un fichero con un identificador y metadatos asociados.

Por otro lado, Amazon S3 ofrece varias funciones para auditar y administrar el acceso a sus recursos. Los buckets y objetos de S3, por defecto, son privados. Las siguientes características permiten conceder y auditar permisos detallados para los recursos de Amazon S3:

- **S3 Block Public Access**: Limita el acceso público a buckets y objetos. A nivel de bucket, la configuración de bloqueo del acceso público se activa por defecto.
- **AWS Identity and Access Management (IAM)**: Es un servicio de AWS que facilita el manejo seguro del acceso a los recursos de AWS, como los recursos de Amazon S3. Permite administrar los permisos que controlan el acceso a los recursos de AWS de forma centralizada. IAM se utiliza para monitorear quién está autorizado para utilizar los recursos.
- **Políticas de Buckets**: Permite establecer permisos basados en recursos para los buckets y objetos de S3 utilizando el lenguaje de políticas de IAM.
- **Puntos de Acceso de Amazon S3**: Para ello, se configuran puntos de acceso de red con nombre y políticas de acceso dedicadas, con el propósito de gestionar el acceso a conjuntos de datos compartidos a gran escala.
- **Listas de Control de Acceso (ACL)**: Son un mecanismo que permite gestionar los permisos y accesos a los objetos y buckets en S3.
- **S3 Object Ownership**: Permite a los propietarios de buckets asumir la propiedad de todos los objetos que se cargan en sus buckets, independientemente de quién cargue los objetos. Esto facilita la administración de permisos y garantiza que los propietarios del bucket puedan gestionar de manera eficiente los datos almacenados.
- **Analizador de Acceso de IAM para S3**: Revisa y monitorea las políticas de acceso a los buckets de S3 para garantizar que sólo se otorgue acceso a los recursos autorizados.

Por otra parte, otra forma de administrar el acceso al bucket S3 es mediante los puntos de acceso de Amazon S3, también conocidos como endpoints de red. Son una característica diseñada para gestionar y restringir el acceso a los datos almacenados en S3. Cada punto de acceso tiene una política de acceso que define cómo se puede acceder a los datos mediante ese endpoint. Dichos puntos pueden utilizarse para realizar operaciones como GetObject y PutObject.

Finalmente, como se aprecia en la siguiente figura, las instancias EC2 realizan sus conexiones al S3 a través del Gateway. Gracias a esto se establece una conexión directa entre las VPC y el S3, sin tener que pasar por la internet

pública, lo que mejora la seguridad de los datos sensibles.

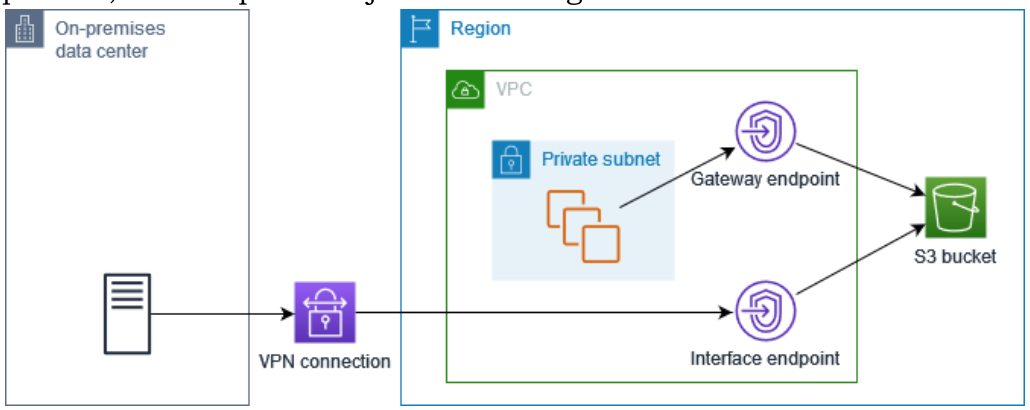

Figura 10. Uso de endpoints para Bucket S3

[22]

2.4.2 DynamoDB

Amazon DynamoDB es un servicio de base de datos NoSQL completamente gestionado que ofrece un rendimiento rápido, escalabilidad predecible y una administración simplificada de la infraestructura subyacente.

Con DynamoDB, los usuarios pueden crear y escalar tablas de bases de datos sin preocuparse por la configuración del hardware, el aprovisionamiento o la replicación de datos.

Las principales características del DynamoDB son las siguientes:

- **Rendimiento y Escalabilidad:** DynamoDB ofrece acceso rápido y predecible a los datos con capacidad para manejar cualquier nivel de tráfico de solicitudes. Además, permite ajustar la capacidad de rendimiento de las tablas de manera dinámica para adaptarse a las necesidades cambiantes sin tiempos de inactividad.
- **Cifrado y Seguridad:** Proporciona cifrado en reposo para proteger la información confidencial sin añadir complejidad operativa adicional. Por ejemplo, se puede añadir las claves de Amazon KMS para cifrar los datos.
- **Backup y Recuperación:** Permite realizar copias de seguridad completas de las tablas para el cumplimiento normativo y la recuperación a un momento dado, que facilita la restauración de tablas a cualquier punto de los últimos 35 días.
- **Eliminación Automática de Datos Obsoletos:** Ofrece la funcionalidad de Time-To-Live (TTL) para eliminar automáticamente los elementos expirados de las tablas, lo que ayuda a reducir el almacenamiento y los costos asociados con datos obsoletos.
- **Alta Disponibilidad y Durabilidad:** DynamoDB distribuye automáticamente los datos a través de múltiples zonas de disponibilidad dentro de una región de AWS, asegurando alta

disponibilidad y durabilidad de los datos. Además, admite tablas globales para la replicación de datos entre diferentes regiones de AWS.

[23]

2.4.3 API Gateway

Amazon API Gateway es un servicio integral de AWS que simplifica la creación, publicación, mantenimiento, monitoreo y protección de APIs REST, HTTP y WebSocket a gran escala. Este servicio permite a los desarrolladores crear APIs para acceder a datos almacenados en la nube y a otros servicios de AWS, como AWS Lambda.

Las principales características de un API Gateway son las siguientes:

- **Creación de APIs:** Permite crear APIs RESTful basadas en HTTP que facilitan la comunicación sin estado entre clientes y servidores. Soporta métodos estándar como GET, POST, PUT, PATCH y DELETE.
- **APIs de WebSocket:** Proporciona APIs que cumplen con el protocolo WebSocket, permitiendo comunicación bidireccional y con estado entre clientes y servidores, adecuadas para aplicaciones en tiempo real.
- **Características de Seguridad:** Ofrece mecanismos de autenticación como políticas IAM, funciones de autorizador Lambda y grupos de usuarios de Amazon Cognito.
- **Despliegue Seguro:** Implementa despliegues de versión Canary para cambios seguros y controlados en las APIs.
- **Monitoreo y Registro:** Integra CloudTrail y CloudWatch para monitorear el uso y los cambios en las APIs, además de permitir la configuración de alarmas para métricas específicas.
- **Integración y Escalabilidad:** Se integra con servicios como AWS Lambda para ejecutar código sin servidor y con infraestructura escalable que garantiza alta disponibilidad y rendimiento.

Por último, en la siguiente figura se puede apreciar la arquitectura de un API Gateway así como todos los servicios que ofrece. [24]

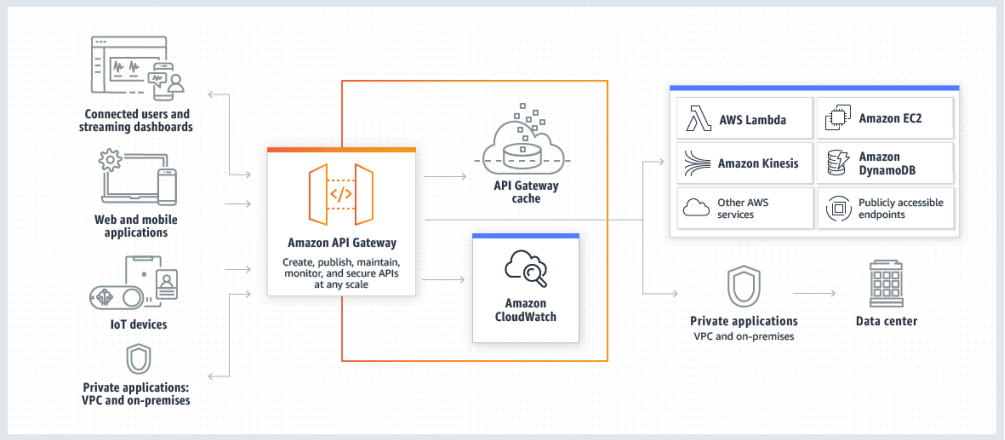

Figura 11. Arquitectura API Gateway

2.4.4 Amazon Lambda

Lambda es un servicio de AWS basado en funciones que elimina la necesidad de gestionar una infraestructura compleja, ya que AWS se encarga de aprovisionar y escalar automáticamente los recursos según la demanda de las funciones Lambda, esto permite un desarrollo más rápido y ágil de las aplicaciones.

Las funciones Lambda solo se ejecutan cuando reciben un evento, que puede provenir de API Gateway, CloudWatch, entre otros.

Las ventajas de utilizar AWS Lambda son las siguientes:

- **Eliminación de la gestión de infraestructura:** AWS maneja la infraestructura subyacente, permitiendo a los desarrolladores centrarse únicamente en el código de la función.
- **Escalabilidad automática:** Permite una escala automáticamente para manejar cualquier carga de trabajo.
- **Pago por uso:** Únicamente se paga por el tiempo de computación utilizado, cuando el código no está en ejecución no tiene costos.
- **Integración con servicios AWS:** Este servicio se integra fácilmente con otros servicios de AWS como S3, DynamoDB y API Gateway, facilitando la creación de aplicaciones complejas y escalables.

Las principales desventajas de utilizar AWS Lambda es que exige límites en el tiempo de ejecución de las funciones, la memoria disponible, el tamaño del código y la cantidad de solicitudes simultáneas que pueden manejar.

Por último, AWS Lambda funciona de la siguiente forma:

- **Eventos y funciones:** Las funciones Lambda se activan por eventos como cargas en S3 o cambios en DynamoDB. Cada función Lambda se ejecuta en un entorno aislado en respuesta a estos eventos.
- **Configuración de funciones:** Se define una función Lambda con un controlador que especifica el código a ejecutar y se configuran las variables de entorno necesarias.
- **Capas y versiones:** AWS permite gestionar diferentes versiones de funciones Lambda y utilizar alias para dirigir el tráfico a versiones específicas.
- **Seguridad y permisos:** Lambda utiliza políticas de ejecución para controlar el acceso a recursos AWS y garantizar la seguridad de las funciones.

[25]

2.4.5 CloudFormation

El servicio de AWS CloudFormation forma parte del paradigma de la infraestructura como código (IaC) que es un enfoque para la gestión y definición de forma automática de los recursos en la nube mediante un código.

Gracias a CloudFormation se ahorra mucho tiempo a la hora de de administrar recursos por lo que permite centrarse ampliamente en las

aplicaciones que se ejecutan en AWS. Para ello, se pueden crear plantillas que describen todos los recursos que queremos implementar ya sean bases de datos, instancias EC2…

[26]

El funcionamiento de CloudFormation se puede observar en la siguiente figura:

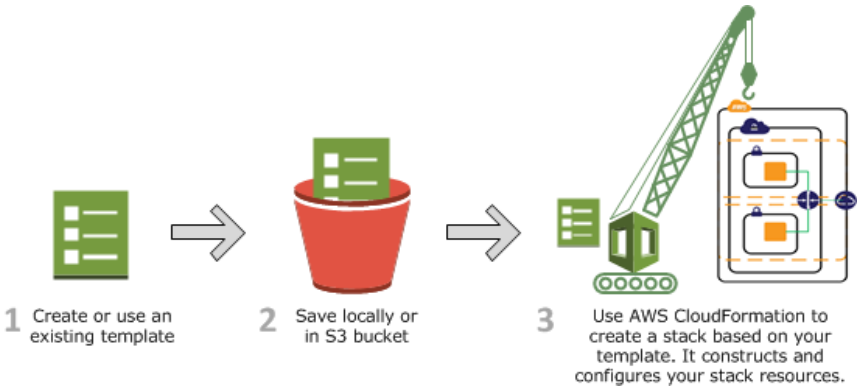

Figura 12. Etapas CloudFormation

Por otro lado, se puede crear una nueva pila (stack) desplegando recursos según una plantilla específica. Además, es posible actualizar una pila existente para realizar cambios mediante modificaciones en la plantilla o ajustando los parámetros según sea necesario.

Asimismo, cuando ya no se necesite, es posible eliminar una pila junto con todos los recursos asociados de forma rápida y completa.

Finalmente, en la parte derecha de la siguiente figura se aprecian las principales características de una plantilla entre las que destaca: **Resources**, el objeto que se va a crear y **parameters**, los parámetros que introducimos a la plantilla a la hora de ejecutarla.

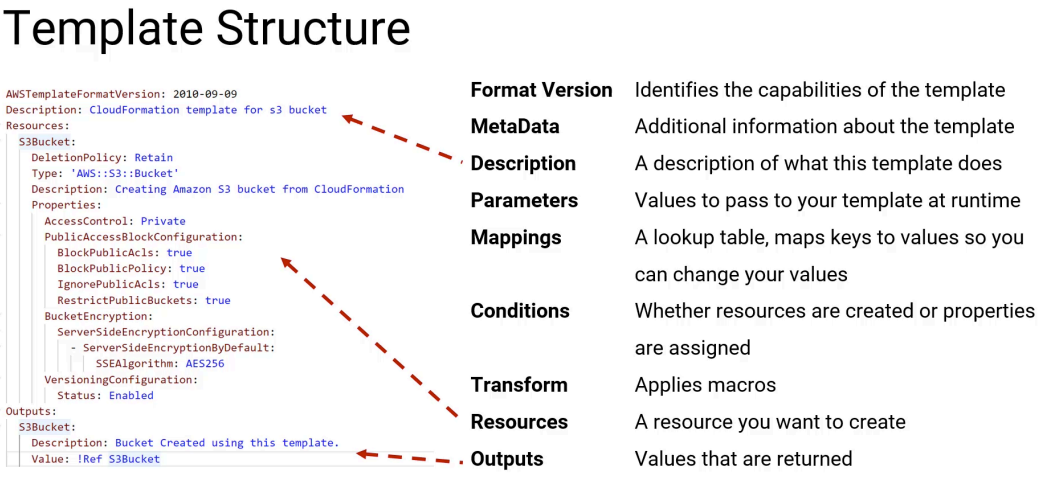

Figura 13. Estructura de un template

3 Desarrollo de la Solución

Continuando con la hoja de ruta definida, para cumplimentar los objetivos específicos planteados al inicio de este proyecto y, después de analizar teóricamente los principales aspectos que rodean el paradigma del Cloud Computing, el presente capítulo describe paso a paso cómo crear y ejecutar un caso práctico real.

El objetivo es demostrar de manera práctica el uso de la metodología de computación en nube en un entorno operativo real, codificando e implementando una API sencilla.

Es importante destacar que el principal objetivo buscado en este capítulo no es realizar simplemente un caso práctico, el verdadero propósito es asegurar que dicho caso esté correctamente securizado y monitorizado. Para ello se hará uso de los conceptos explicados previamente.

3.1 Diseño de la Arquitectura

El objetivo de este capítulo, es explicar en detalle la solución cloud desplegada. Para ello, en la siguiente figura, se aporta el detalle en torno a la representación gráfica de la arquitectura de servicios que intervendrán a lo largo del caso práctico, así como de las interrelaciones establecidas entre cada uno de los recursos. Los servicios incluidos, implementarán una serie de funcionalidades integradas para garantizar el correcto funcionamiento de la API.

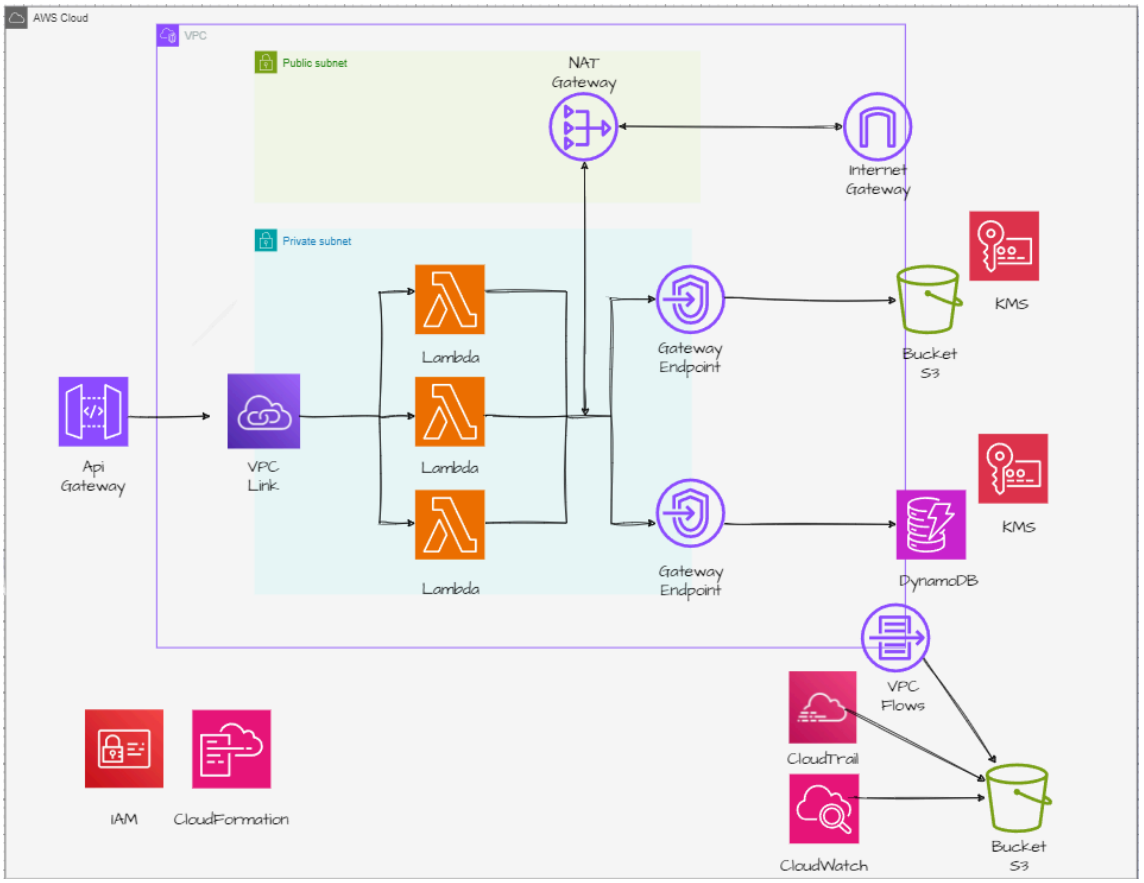

Figura 14. Arquitectura caso práctico

Como se ha nombrado previamente, la arquitectura representará una API que girará en torno al concepto "serverless". Es decir, nos permitirá crear aplicaciones sin la necesidad de tener que provisionar ningún tipo de infraestructura, gracias a ello, el coste de mantener las aplicaciones será menor. [27] Esto podrá ser posible ya que este tipo de aplicaciones se ejecutan en respuestas a eventos como Amazon CloudWatch, Amazon SQS o API Gateway.

Un ejemplo de computación sin servidor o "serverless" es Amazon Lambda explicada anteriormente de la cual se va hacer uso para este caso práctico.

La arquitectura expuesta actúa como una biblioteca en la que podemos realizar diferentes operaciones respecto a los libros almacenados.

La API cuenta con las siguientes funcionalidades:

- **Añadir libro:** Mediante una petición POST podemos añadir libros a la biblioteca pasando la información en formato JSON. El formato del JSON será el siguiente:

Figura 15. Ejemplo fichero JSON para crear un libro

Como se aprecia el objeto 'book' contendrá la información acerca del libro. Entre esta información destaca el título y el autor del libro.

Por otra parte, tenemos el objeto 'pdf_data' que contendrá una secuencia de datos codificados en Base64 que representa el contenido del archivo en formato PDF del libro.

- **Obtener libro:** Gracias a una petición GET podemos obtener la información sobre cualquier libro almacenado en la biblioteca. Para ello se debe pasar el nombre del libro en cuestión mediante el Query param 'libro'.
- **Eliminar libro:** Mediante una petición DELETE podemos eliminar la información sobre cualquier libro almacenado en la biblioteca. La forma de uso es idéntica al del método 'Obtener Libro'.

A continuación, se describe detalladamente cada componente de la arquitectura.

- **API Gateway:** Supondrá la puerta para las peticiones HTTP/HTTPS ya que todas las peticiones llegarán al mismo. Posteriormente el gateway se encargará de mandar los eventos a las funciones lambda.
- **Lambda:** Son funciones sin servidor que se activan al recibir un evento por parte del API Gateway. En el diagrama de arquitectura anterior, se observa que dispone de tres funciones Lambda. Una de estas funciones está dedicada al método POST y presenta diversas responsabilidades. La primera consta de almacenar los datos del objeto book, que contiene información sobre el libro, en una tabla de DynamoDB. Cada registro de la tabla representará las características del libro, como el título y el autor. Además, la función se encarga de convertir el contenido de 'pdf_data', que es una cadena codificada en Base64, a formato PDF. Posteriormente, este archivo PDF se almacena en un bucket de Amazon S3.
  Otra función lambda se encarga de realizar el método GET. Tiene las siguientes funciones: Se encarga de buscar en el DynamoDB el libro que contenga el título que le hemos pasado por el Query Param 'libro'. Además, nos proporciona el enlace al PDF contenido en el S3. Esta lambda se encarga de buscar el id que contiene la información del libro en DynamoDB y busca el PDF en el S3 que tenga ese id como nombre. Por último, la función lambda que realiza el método DELETE se encarga de borrar los datos del libro tanto en el DynamoDB como en el S3.

Además, el método de búsqueda de la información del libro es igual que el de la lambda anterior.

- **Amazon VPC (Virtual Private Cloud):** Desempeña un papel crucial a la hora de proporcionar un entorno de red seguro y aislado para los recursos. Además, consta de dos subredes, una pública y una privada, la pública presenta acceso a internet, sin embargo, la privada no dispone de este acceso. De esta forma, las funciones lambda contenidas en la subred privada no son accesibles desde internet lo que mejora su seguridad. Asimismo, disponemos de un NAT Gateway que se encarga de proporcionar acceso a internet a las funciones lambda pero no proporciona acceso desde internet a las funciones lambda. Por último, se dispone de un VPC Endpoint que proporciona acceso seguro desde las funciones lambda tanto al DynamoDB como al S3.
- **S3 Bucket:** Se utiliza para almacenar objetos como archivos estáticos (PDF, videos, imágenes…). En este proyecto se utilizará para almacenar el PDF de cada libro.
- **DynamoDB:** Es un servicio de base de datos NoSQL (son bases de datos que utilizan modelo de datos diferentes a los tradicionales)[28]. En este caso se utilizará para almacenar la información referente a cada libro.
- **AWS KMS (Key Management Service):** Es un servicio que nos proporciona claves para cifrar datos almacenados. Se utilizará para cifrar los datos almacenados en el S3.
- **Amazon CloudWatch:** Es un servicio que nos permite monitorizar recursos, registrar métricas y poner alarmas entre otras funciones. En este caso se utilizará para monitorear tanto las lambda, como el S3 y DynamoDB. Además, se han implementado alarmas para las lambda. Todos los registros se almacenan en un S3.
- **Amazon CloudTrail:** Se utiliza para registrar acciones que hagan las cuentas de AWS sobre los diferentes recursos. También captura llamadas a la API. Todos los registros se almacenan en un S3.
- **IAM:** Es un servicio que nos permite gestionar las identidades y accesos. Por ejemplo, se usará para otorgar permisos a las funciones lambda.

Por último, todos los recursos están desplegados sobre la región de ‘eu-west-3’ (París) ya que nos permite proximidad geográfica al estar cerca de España, mejorando la latencia de la conexiones así como el rendimiento general de la infraestructura al tener tiempos de respuesta menores.

A conjunción, en los próximos capítulos se realizará el desarrollo del despliegue de la infraestructura paso por paso.

3.2 Análisis de Costes

Este capítulo se centra en realizar un análisis del costo que supone mantener la infraestructura del caso práctico. Para ello, AWS proporciona una serie de herramientas para medir los costos de la infraestructura desplegada. Entre estas herramientas destaca 'Administración de facturación y costos', que permite ver un análisis real de los costos de cada servicio desplegado, así como visualizar facturas. Además, nos permite ajustar la línea temporal.

Por otro lado, para realizar la estimación mensual, se ha tenido en cuenta los costos de mantener la infraestructura durante una semana. Como se aprecia en las siguientes figuras, este costo asciende a 16,62 USD, por lo que en un mes el costo sería de aproximadamente 66,48 USD.

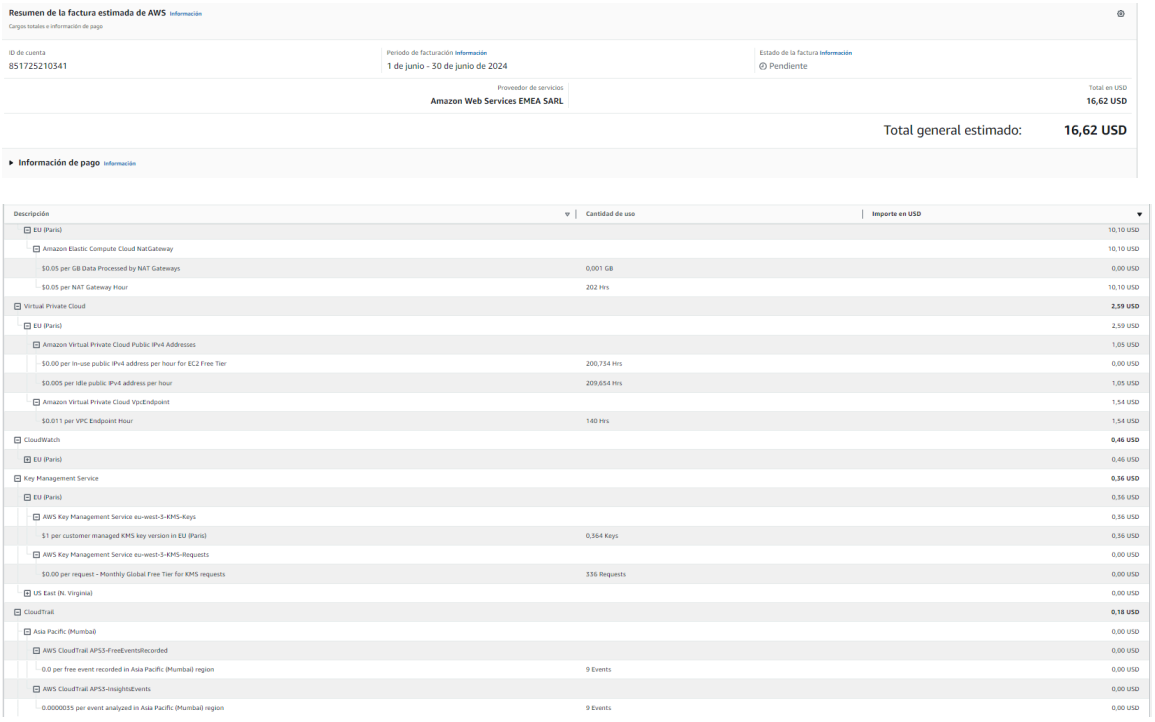

Figura 16. Análisis de costes

3.3 Creación de la Arquitectura

En este capítulo, se comienza a desplegar los principales servicios de esta infraestructura como lo son: DynamoDB, Bucket S3, Funciones Lambda y API Gateway. Para ello, se realizará la implementación a través de la consola de AWS ya que esta, ofrece una interfaz gráfica muy fácil de usar y altamente intuitiva.

3.3.1 Creación DynamoDB

Primeramente, haciendo uso del servicio DynamoDB, se generará una nueva tabla que servirá para almacenar las características de los libros como se ha mencionado anteriormente. Además, los elementos de la tabla poseerán una clave de partición que será única para cada elemento de la tabla como se puede ver en la figura adjunta. Esta clave forma parte de la clave principal de la tabla. Se trata de un valor hash que se utiliza para recuperar elementos de la tabla, así como para asignar datos entre hosts por cuestiones de escalabilidad y disponibilidad. En la siguiente figura se puede observar los primeros pasos para la creación de la tabla así como el nombre de la clave de partición que será 'id'.

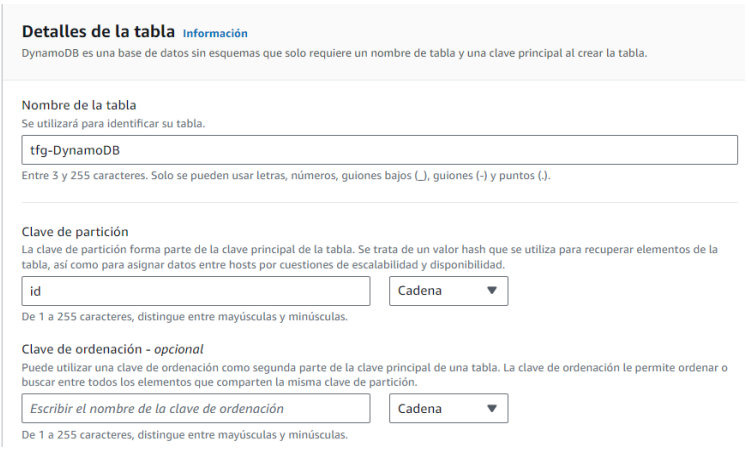

Figura 17. Creación DynamoDB

En cuanto a otra configuración, se activa la protección contra eliminaciones, esto evita que se pueda eliminar la tabla de forma involuntaria. Una vez configurada la tabla se procede a su despliegue.

3.3.2 Creación Bucket S3

Para poder almacenar los archivos en formato PDF se debe crear un bucket S3. En cuanto la configuración de este, se marcará la opción de ACL desactivada para el apartado de propiedad de objetos. Gracias a esto, el control de acceso a los objetos no se gestionará a través de listas de control de acceso (ACL) sino a través de políticas asociadas al cubo o a los objetos, como se aprecia en la figura de abajo.

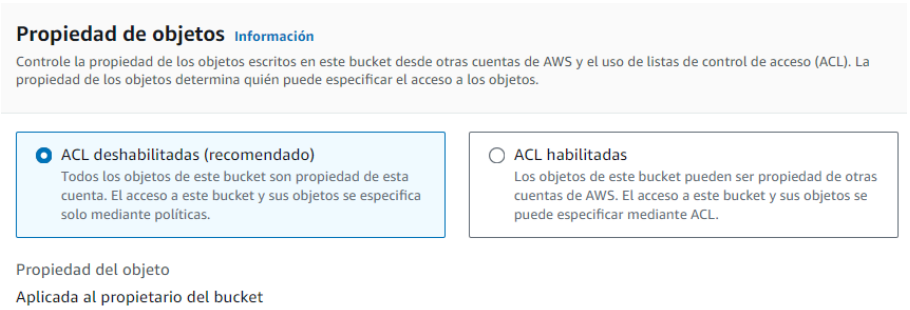

Figura 18. Configuración ACL Bucket S3

Además, para asegurar que los objetos contenidos en el S3 se encuentren protegidos y que no sean accesibles al público se debe desactivar todo el acceso público.

Figura 19. Configuración acceso público Bucket S3

Como se aprecia en la figura anterior, al activar esta opción se garantiza que no se creen nuevas políticas de bucket, puntos de acceso públicos y que no se aplique nuevas ACL que concedan acceso público a los buckets u objetos recién creados. También, se encarga de que S3 ignore todas las ACL, políticas de bucket y puntos de acceso que concedan acceso público. [29]

Asimismo, se ha activado el control de versiones para mantener múltiples variantes de un objeto dentro del mismo bucket y de esta forma poder conservar, recuperar y restaurar todas las versiones de los objetos almacenados en su bucket de Amazon S3. [30]

3.3.3 Creación Funciones Lambda

Una vez se ha desplegado el DynamoDB y el Bucket S3 se continuará desplegando las funciones lambda en AWS que supondrán el núcleo principal de esta arquitectura.

En primer lugar, se realizarán las primeras configuraciones para las funciones lambda, para ello es necesario elegir el lenguaje de programación para el

código de la función lambda. Se ha escogido el lenguaje Python 3.12 ya que es sencillo de utilizar y posee todas las librerías necesarias para realizar consultas a los servicios de AWS.

Cuando se despliega la función lambda, se genera con un código por defecto que nos imprime por consola 'Hello from Lambda!'. Se puede realizar una prueba para comprobar el correcto funcionamiento de la función lambda.

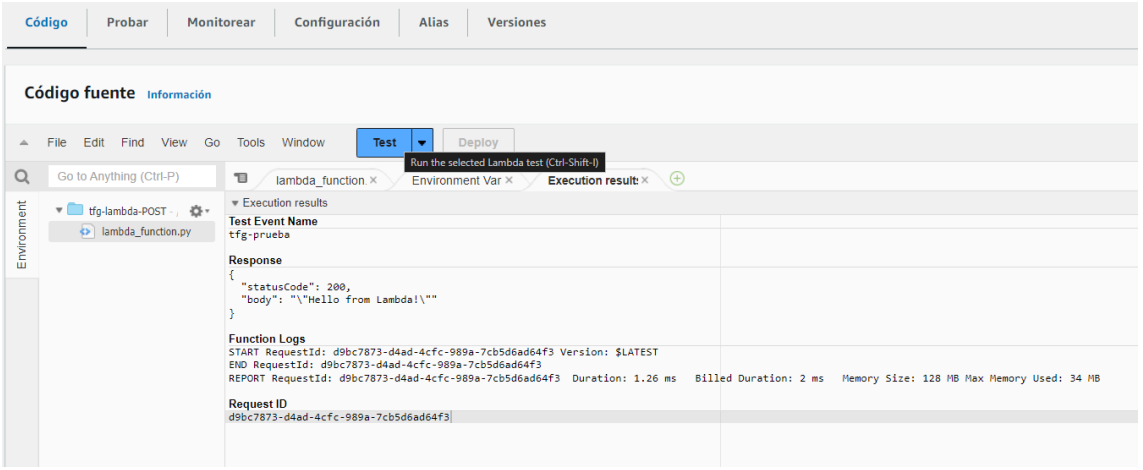

Figura 20. Prueba función lambda

Como se aprecia en la figura anterior, la función devuelve 'Hello from Lambda!' con el código de respuesta 200 OK, esto significa que la petición se ha procesado correctamente.

A continuación, se debe programar el código Python para cada una de las tres funciones lambda. Para ello, se hará uso de 'boto3' que es el AWS SDK para Python. [31]

En cuanto al código función POST, se encarga de iniciar el cliente de DynamoDB y S3 mediante boto3.resource('dynamodb') y boto3.client('s3') respectivamente. Seguidamente, se encarga de obtener los datos del libro del JSON y convertir la cadena en base64 a PDF. Luego, se crea un id para el libro y lo introduce en la tabla de DynamoDB e introduce el PDF con el mismo id al bucket S3. Finalmente, devuelve 201 CREATED en caso de que se haya realizado la petición correctamente o 500 si ha detectado algún error.

```
import json
import uuid
import boto3
import base64
from botocore.exceptions import ClientError

# Inicializamos el cliente de DynamoDB
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('tfg-DynamoDB')

#  Inicializamos el cliente de S3
s3_client = boto3.client('s3')
bucket_name = 'tfg-bucket-s3'
```

```
def lambda_handler(event, context):
    try:

        body = json.loads(event['body'])
        book = body['book']
        pdf_data = body['pdf_data']

        # Decode the base64-encoded PDF
        pdf_content = base64.b64decode(pdf_data)

        # Create a new book with a UUID
        book_id = str(uuid.uuid4())  # Generate a unique UUID
        new_book = {
            **book,
            'id': book_id
        }

        # Put the new book into the DynamoDB table
        table.put_item(Item=new_book)

        # Upload the PDF to S3 with the same UUID
        pdf_key = f"{book_id}.pdf"
        s3_client.put_object(Bucket=bucket_name, Key=pdf_key, Body=pdf_content,
ContentType='application/pdf')

        # Add the PDF URL to the book information
        new_book['pdf_url'] = f"https://{bucket_name}.s3.amazonaws.com/{pdf_key}"

        # Return success response
        return {
            'statusCode': 201,
            'body': json.dumps(new_book)
        }

    except ClientError as error:
        # Log the error and return failure response
        print(error)
        return {
            'statusCode': 500,
            'body': json.dumps({'message': str(error)})
        }
```

Código 1. Función lambda POST

Por otro lado, el código función GET se encarga de iniciar los clientes del bucket S3 y del DynamoDB. Posteriormente se obtiene el título del libro a partir de query param 'libro' proporcionado mediante query_parameters = event['queryStringParameters']. Si no se proporciona un query param devolvemos el error 400. Luego, se realiza un escaneo de la tabla de DynamoDB para encontrar elementos que tengan el mismo título. En el caso de encontrar coincidencia, se devuelve un JSON con la información del libro y el código 200 OK. Si no se encuentra ningún elemento, se devuelve el código 404. En caso de error, se devuelve el código de error 500.

```python
import json
import boto3
from botocore.exceptions import ClientError
from decimal import Decimal

# Configuración de recursos y clientes de AWS
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('tfg-DynamoDB')
s3_client = boto3.client('s3')
bucket_name = 'tfg-bucket-s3'

# Clase de codificación personalizada para manejar Decimals
class DecimalEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, Decimal):
            return str(o)  # Convertir Decimal a string
        return super().default(o)

def lambda_handler(event, context):
    try:
        # Obtener parámetros de consulta
        query_parameters = event.get('queryStringParameters', None)
        if not query_parameters or 'libro' not in query_parameters:
            return {
                'statusCode': 400,
                'body': json.dumps({'message': 'Query parameter "libro" is required'})
            }

        title = query_parameters['libro']

        # Escanear la tabla DynamoDB en busca del libro
        response = table.scan(
            FilterExpression='title = :title',
            ExpressionAttributeValues={':title': title}
        )

        if 'Items' in response and len(response['Items']) > 0:
            # Si se encuentra el libro, construir la respuesta con la URL del PDF
            book = response['Items'][0]
            book_id = book['id']
            pdf_key = f"{book_id}.pdf"
            book['pdf_url'] = f"https://{bucket_name}.s3.amazonaws.com/{pdf_key}"

            # Devolver la respuesta con el estado 200 y el libro serializado a JSON
            return {
                'statusCode': 200,
                'body': json.dumps(book, cls=DecimalEncoder)
            }
        else:
            # Si el libro no se encuentra, devolver estado 404 y mensaje
            return {
                'statusCode': 404,
                'body': json.dumps({'message': 'Book not found'})
            }

    except ClientError as e:
        # Capturar errores de cliente (por ejemplo, permisos insuficientes)
        print(f"Error scanning DynamoDB table: {e.response['Error']['Message']}")
```

```
        return {
            'statusCode': 500,
            'body': json.dumps({'message': 'Internal server error'})
        }
```

Código 2. Función lambda GET

Finalmente, el código de la función DELETE funciona de manera similar al código anterior, pero en vez de devolver un JSON con la información del libro, se encarga de eliminar el elemento de la tabla de DynamoDB y el PDF del bucket S3 mediante table.delete_item(Key={'id': book_id}) y s3_client.delete_object(Bucket=bucket_name, Key=pdf_key).

```
import json
import boto3
from botocore.exceptions import ClientError


dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('tfg-DynamoDB')
s3_client = boto3.client('s3')
bucket_name = 'tfg-bucket-s3'

def lambda_handler(event, context):
    query_parameters = event['queryStringParameters']
    if not query_parameters or 'libro' not in query_parameters:
        return {
            'statusCode': 400,
            'body': json.dumps({'message': 'Query parameter "libro" is required'})
        }

    title = query_parameters['libro']

    try:
        response = table.scan(
            FilterExpression='title = :title',
            ExpressionAttributeValues={':title': title}
        )

        if 'Items' in response and len(response['Items']) > 0:
            book = response['Items'][0]
            book_id = book['id']
            pdf_key = f"{book_id}.pdf"

            table.delete_item(Key={'id': book_id})

            s3_client.delete_object(Bucket=bucket_name, Key=pdf_key)

            return {
                'statusCode': 200,
                'body': json.dumps({'message': 'Book deleted successfully'})
            }
        else:
            return {
                'statusCode': 404,
                'body': json.dumps({'message': 'Book not found'})
            }
```

```
except ClientError as error:
    print(error)
    return {
        'statusCode': 500,
        'body': json.dumps({'message': str(error)})
    }
```

Código 3. Función lambda DELETE

Una vez programado el código de cada función lambda, se procede a actualizar el código y se despliegan con el botón 'Deploy', como se aprecia en la siguiente figura.

Figura 21. Despliegue código función lambda

3.3.4 Creación API Gateway

Una vez implementadas las funciones lambda, es necesario desplegar un recurso que haga de intermediario. En este caso, API Gateway se encarga de gestionar todas las peticiones HTTP y HTTPS, convirtiéndolas en eventos que las dichas funciones puedan controlar, de esta forma cada función lambda atenderá a las peticiones realizadas. Asimismo, API Gateway es capaz de procesar una gran variedad de peticiones y posteriormente distribuirlas de forma eficiente a las funciones lambda, esto permite que la infraestructura sea escalable.

Existen diferentes tipos de API Gateway, tanto API HTTP o API REST. En este caso, se ha utilizado API HTTP ya que es más sencillo de configurar y usar que API Rest, además, de que el coste que supone desplegar API HTTP es notablemente menor.

Una vez desplegado el API Gateway, se deben configurar las rutas esenciales para segmentar y gestionar el tráfico de solicitudes HTTP y HTTPS dirigido al backend. Asimismo, se puede crear diferentes rutas para dirigir hacia parte especificas del backend, de esta forma podemos manejar peticiones POST, GET, DELETE… todas ellas en diferentes rutas.

Como se puede apreciar en la siguiente figura, para configurar una ruta se debe seleccionar el método a manejar y definir ruta partiendo siempre de la URI base del API Gateway.

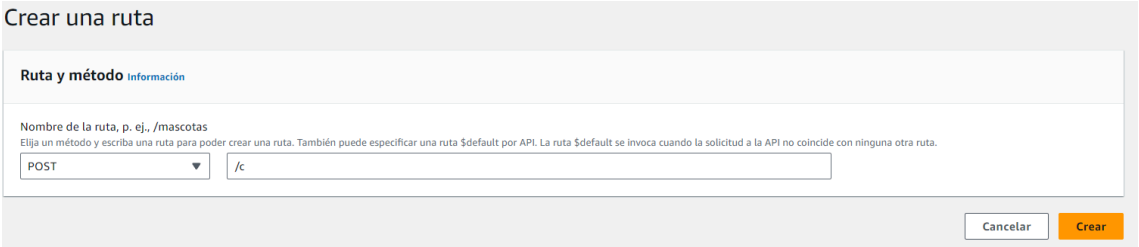

Figura 22. Configuración rutas API Gateway

Una vez se han creado todas las rutas, el API Gateway quedarán de la siguiente forma:

Figura 23. Rutas API Gateway

- **Ruta para POST:**
  https://hsgu4f0frj.execute-api.eu-west-3.amazonaws.com/c
- **Ruta para GET y DELETE:**
  https://hsgu4f0frj.execute-api.eu-west-3.amazonaws.com/libros?libro='titulo'

A continuación, para poder asociar cada ruta con su respectiva función lambda se debe configurar las integraciones. Para ello, en cada ruta se crea una integración en la que se debe elegir qué función lambda va a atender a las peticiones de dicha ruta.

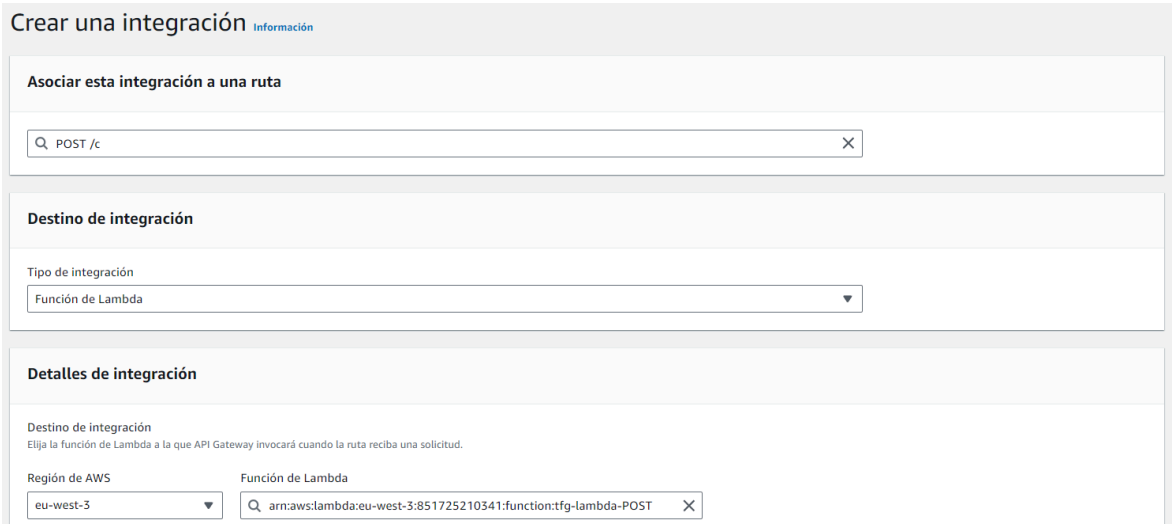

Figura 24. Creación integración API Gateway

Una vez configurado todos los elementos necesarios del API Gateway la arquitectura en este punto debe ser funcional por lo que se podrán realizar varias pruebas a través de Postman.

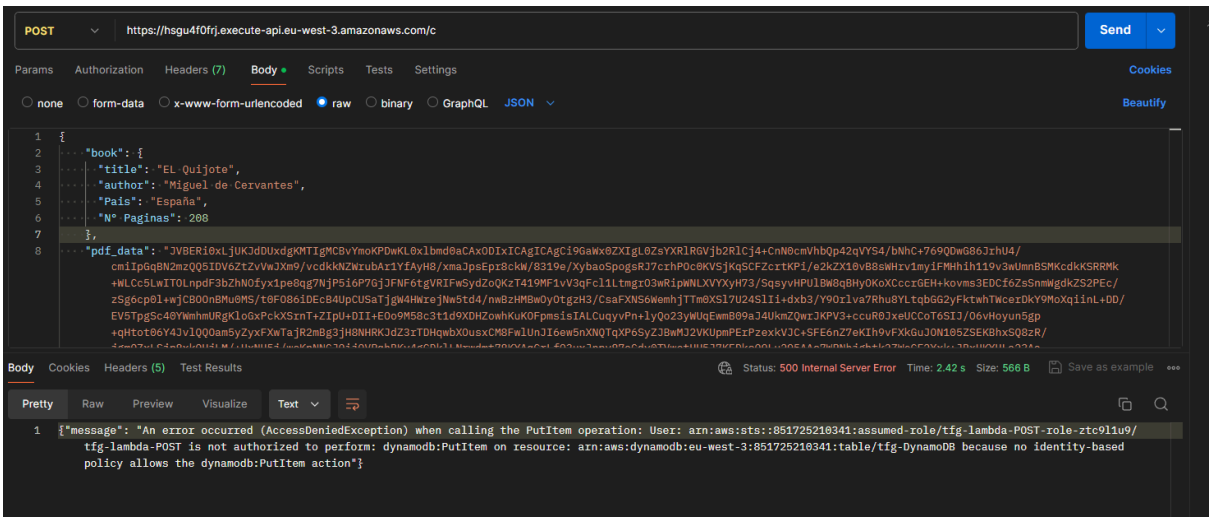

Figura 25. Prueba de ejecución POST

Como se aprecia en la figura anterior, al realizar la prueba del método POST, se da el siguiente error:

{"message": "An error occurred (AccessDeniedException) when calling the PutItem operation: User: arn:aws:sts::851725210341:assumed-role/tfg-lambda-POST-role-ztc9l1u9/tfg-lambda-POST is not authorized to perform: dynamodb:PutItem on resource: arn:aws:dynamodb:eu-west-3:851725210341:table/tfg-DynamoDB because no identity-based policy allows the dynamodb:PutItem action"}

Esto significa que la función lambda no está autorizada para introducir elementos en el DynamoDB ('PutItem'). Por lo que, se debe agregar la siguiente política de permisos al rol de IAM asociado a la función lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:PutItem",
            "Resource": "arn:aws:dynamodb:eu-west-3:851725210341:table/tfg-DynamoDB"
        }
    ]
}
```

Política 1. PutItem lambda-DynamoDB

Por otro lado, al ejecutar de nuevo se genera el siguiente error:

{"message": "An error occurred (AccessDenied) when calling the PutObject operation: Access Denied"}

En este caso, sucede algo similar a lo anterior pero con el Bucket S3. Para subsanar dicho error se debe añadir la siguiente política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::tfg-bucket-s3/*"
        }
    ]
}
```

Política 2. PutObject lambda-Bucket S3

Subsanados dichos errores, la petición POST funciona correctamente como se aprecia en la siguiente figura ya que nos devuelve 201 CREATED.

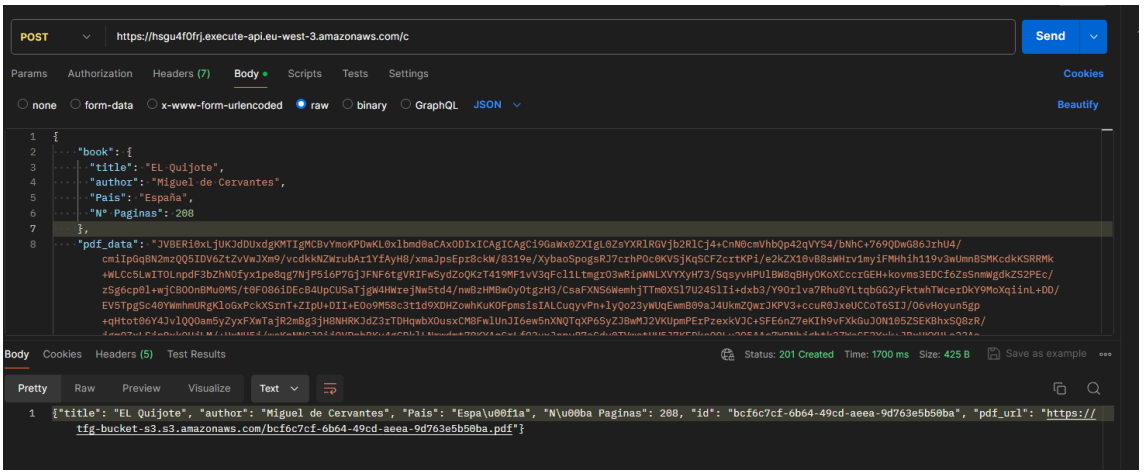

Figura 26. Prueba de ejecución POST corregida

Finalmente, para los métodos GET y DELETE surgen problemas muy parecidos a los que se menciona anteriormente, ya que la función lambda que maneja el método GET no tiene permiso para realizar un escaneo del Dynamodb ('Scan action') ni para obtener objetos del Bucket S3 ('GetObject'). Para subsanar esto, se deben agregar las siguientes políticas a la función lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "dynamodb:Scan",
            "Resource": [
                "arn:aws:dynamodb:eu-west-3:851725210341:table/tfg-DynamoDB",
                "arn:aws:dynamodb:*:851725210341:table/*/index/*"
            ]
        }
    ]
}
```

Política 3. Scan lambda-DynamoDB

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tfg-bucket-s3 /*"
        }
    ]
}
```

Política 4. GetObject lambda-Bucket S3

Por otro lado, respecto a la función DELETE será necesario agregar las siguientes políticas para realizar escaneos al Dynamodb ('Scan action') como el de la figura *Política 3. Scan lambda-DynamoDB*, eliminar elementos de las tablas del DynamoDB ('DeleteItem') y eliminar objetos del Bucket S3 ('DeleteObject').

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "dynamodb:DeleteItem",
            "Resource": "arn:aws:dynamodb:eu-west-3:851725210341:table/tfg-DynamoDB"
        }
    ]
}
```

Política 5. DeleteItem lambda-DynamoDB

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:DeleteObject",
            "Resource": "*"
        }
    ]
}
```

Política 6. DeleteObject lambda-Bucket S3

Para concluir este capítulo, tras agregar dichas políticas los métodos GET y DELETE. Los métodos funcionan correctamente como se aprecia en las siguientes figuras.

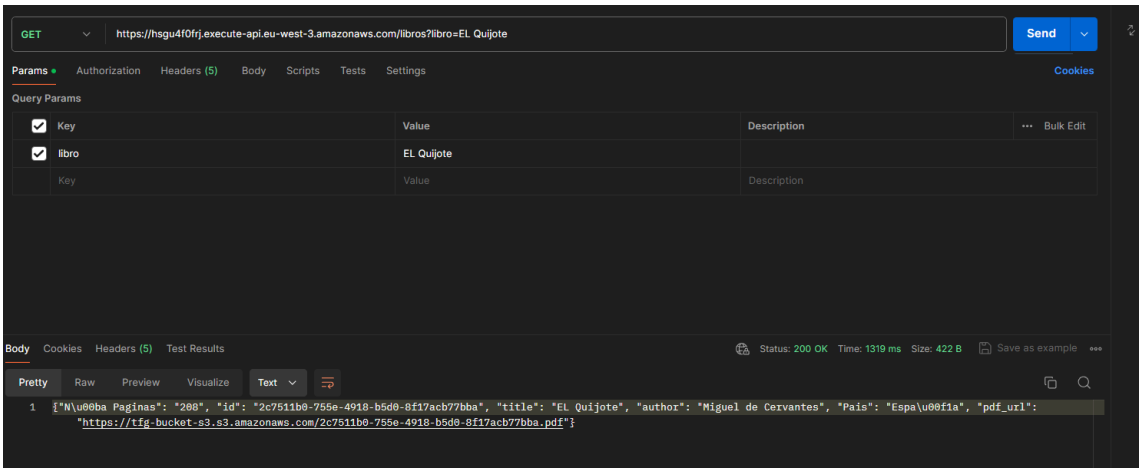

Figura 27. Prueba de ejecución GET

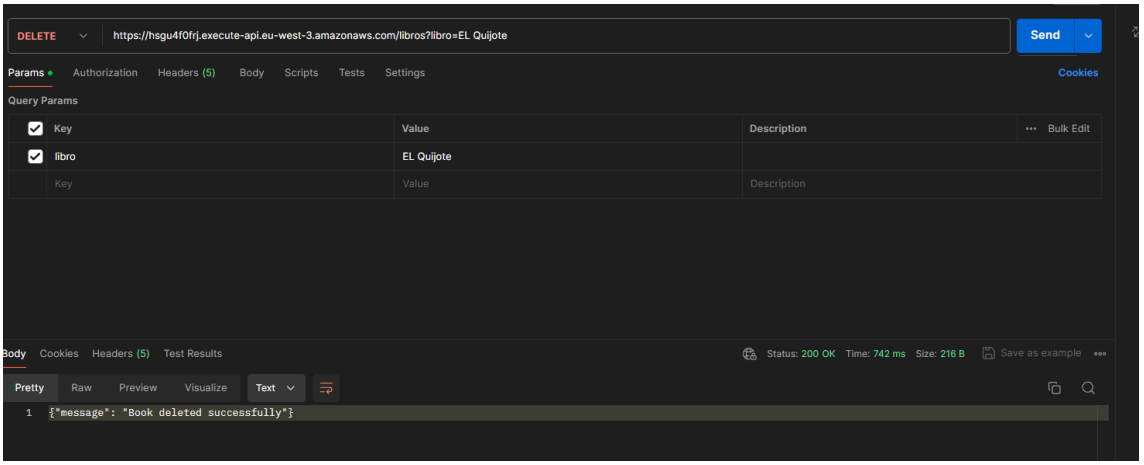

Figura 28. Prueba de ejecución DELETE

3.4 Securizar la Infraestructura con VPC

A pesar de que se han desplegado todos los elementos esenciales para que la infraestructura sea funcional, esto no garantiza que se haya desplegado de forma segura. Al no tener implementado una virtual private cloud (VPC), todos los servicios están expuestos a la internet pública, lo que provoca que los recursos sean más vulnerables a posibles ataques. Además, sin una VPC, los recursos de AWS que quieran acceder a internet serán accesibles desde fuera, ya que no pueden usar NAT Gateways.

Bajo esta premisa, se procederá a desplegar una Virtual Private Cloud (VPC) para mejorar la seguridad y el control de acceso de la infraestructura.

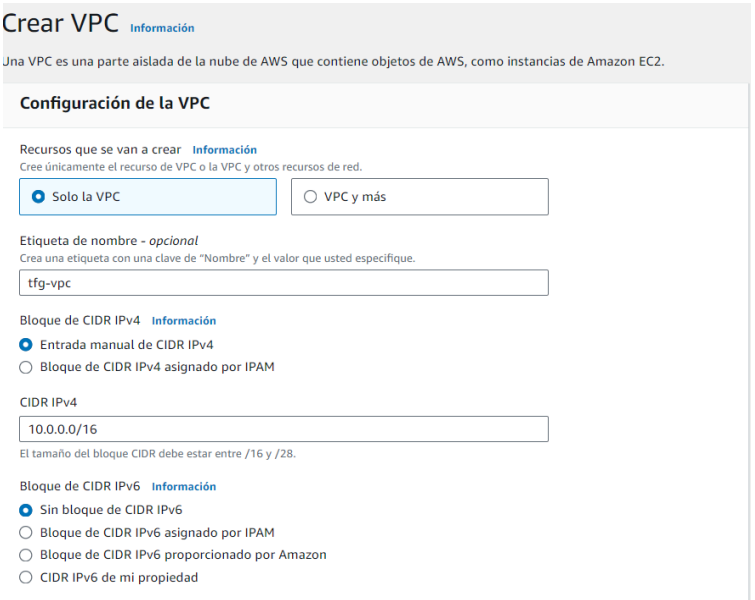

Figura 29. Creación VPC

Como se aprecia en la figura anterior, se despliega la VPC con el rango de direcciones 10.0.0.0/16 lo que permite 65536 host que es más que suficiente para este proyecto.

Atendiendo a la topología de red de la *Figura 14. 'Arquitectura caso práctico'*, se deben desplegar dos subredes. En primer lugar, una subred privada para las lambdas que no tendrá acceso a internet. Asimismo, una subred pública con NAT Gateway para proporcionar conectividad a internet a las funciones lambda. Específicamente, la subred publica tendrá el rango de direcciones 10.0.1.0/24 que permite 256 host. Además, la subred privada tendrá el rango de direcciones 10.0.0.0/24 que también permite 256 host.

Para permitir la conectividad entre la subred privada y pública es crucial configurar las tablas de rutas de cada subred ya que estas se encargan de gestionar el tráfico interno de la VPC y, hacia y desde internet.

Se comienza configurando la tabla de rutas para la primera subred pero antes es necesario crear un internet gateway para tener salida a internet desde la VPC. Como se aprecia en la siguiente figura.

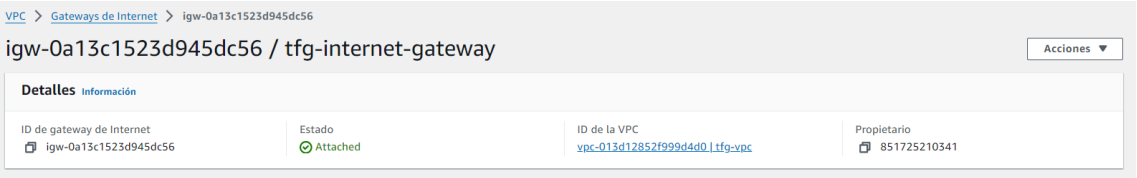

Figura 30. Creación Internet Gateway

Una vez creado el internet gateway, se configura la tabla de rutas como se refleja en la siguiente figura.

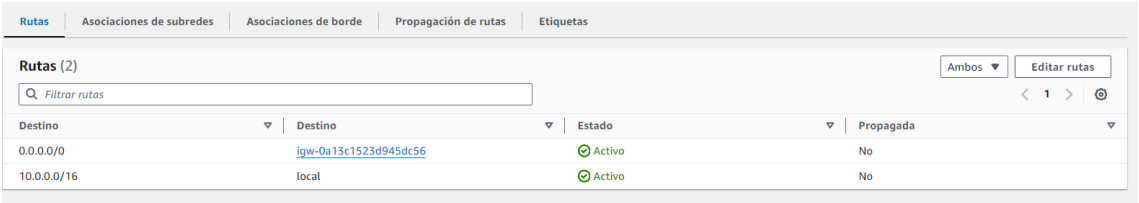

Figura 31. Tabla de rutas subred pública

Consta de dos rutas:

- **Destino 0.0.0.0/0 Destino: Internet gateway:** Se encarga de dirigir todo el tráfico destino a internet a través de un Internet gateway.
- **Destino: 10.0.0.0/16 Destino: local:** Se encarga de que todo el tráfico dirigido a las direcciones ip dentro del rango de la VPC se enrute localmente.

A continuación, se realizará la tabla de rutas para la subred privada, para lo cual se debe crear un NAT Gateway asociado a la subred pública. En la siguiente figura se puede apreciar las tabla de rutas.

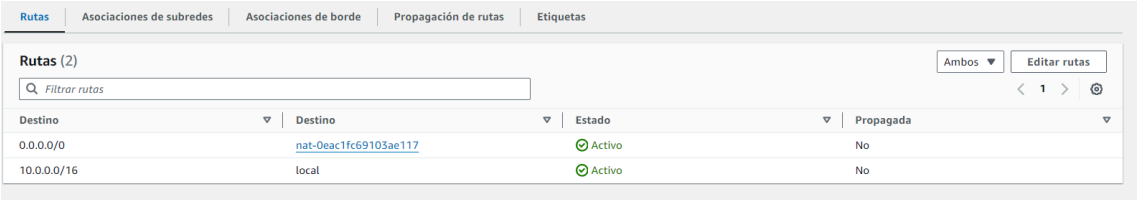

Figura 32. Tabla de rutas subred privada

Se compone de dos rutas:

- **Destino: 10.0.0.0/16 Destino: local:** Explicada anteriormente.
- **Destino 0.0.0.0/0 Destino: NAT Gateway:** Se encarga de que todo el tráfico que se dirija a Internet y con origen cualquier instancia dentro de la VPC se dirija al NAT Gateway.

3.4.1 Endpoint para DynamoDB y Bucket S3

En esta sección, se desplegarán los endpoints para DynamoDB y Bucket S3. Estos endpoints sirven para mejorar la seguridad y rendimiento para acceder a estos recursos desde dentro de una VPC ya que se establece una conexión directa entre los VPC y cada recurso sin tener que pasar por la internet pública favoreciendo la seguridad de los datos sensibles.

Primeramente, se configurará el Endpoint para el bucket S3. Para eso, se debe seleccionar el servicio *com.amazonaws.eu-west-3.s3* de tipo gateway. Además, se añade a la tablas de rutas de la subred privada como se aprecia en la siguiente figura.

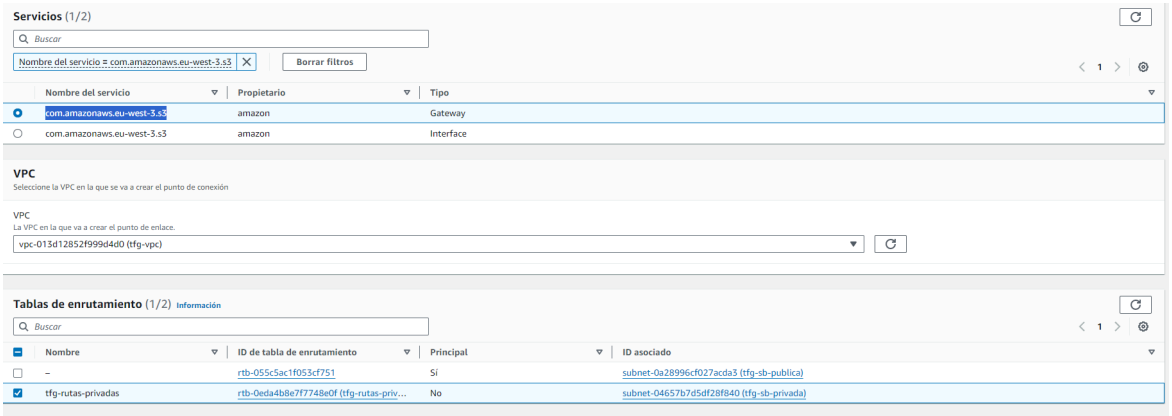

Figura 33. Creación Endpoint Bucket S3

Para el DynamoDB se sigue el mismo procedimiento pero escogiendo el servicio *com.amazonaws.eu-west-3.dynamodb* de tipo gateway como se aprecia en la siguiente figura.

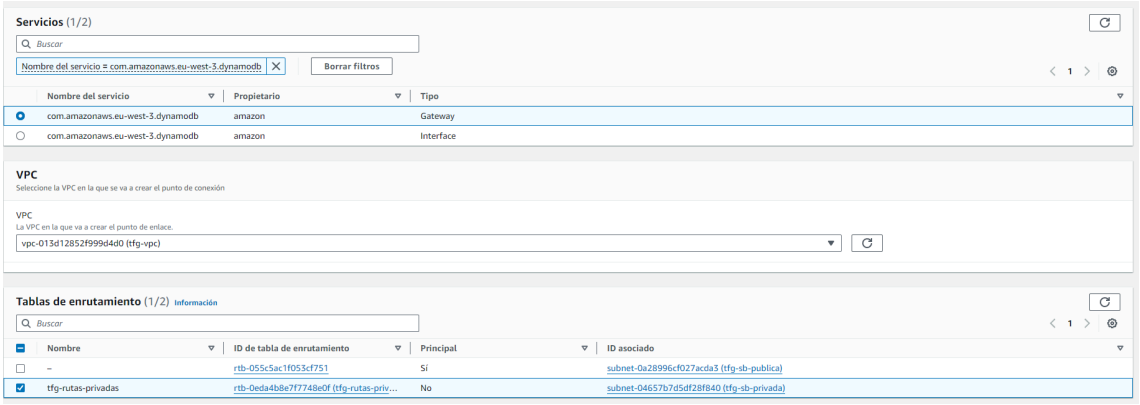

Figura 34. Creación Endpoint DynamoDB

Finalmente, en la tabla de rutas de la subred privada quedarían añadidas las dos rutas para los endpoint como se aprecia en la siguiente figura.

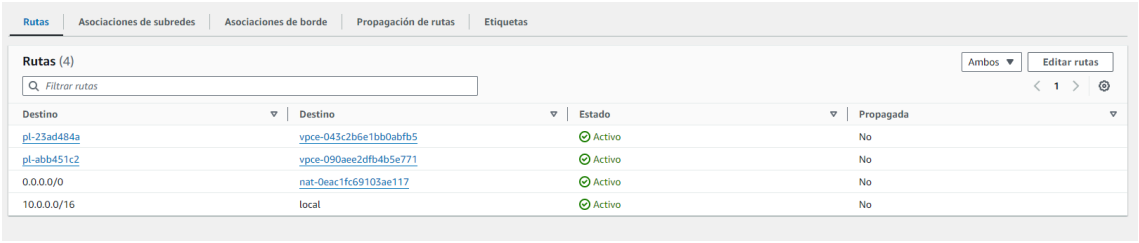

Figura 35. Tabla de rutas privada actualizada

3.4.2 VPC Link para API Gateway

En este capítulo se procederá a integrar el API Gateway con los recursos que se encuentran dentro de una VPC. Para ello, se hará uso de VPC Link.

Como se aprecia en la siguiente figura para configurarlo se debe seleccionar en que subred se integrará en este caso, en la subred privada como se aprecia en la siguiente figura.

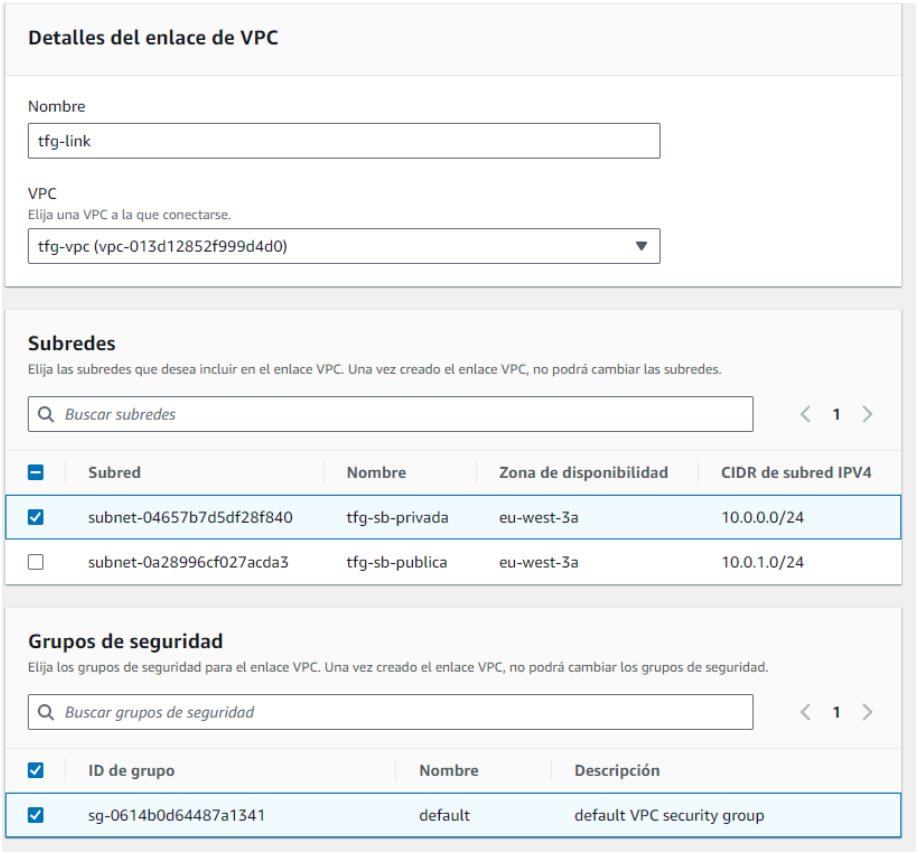

Figura 36. Creación VPC Link

3.4.3 Integración de la Función lambda con VPC

Por último, para asegurar que las funciones lambda no sean accesibles desde internet se debe integrar con una VPC. Cuando se realiza esa integración AWS le asigna a la función lambda una interfaz de red elástica que actúa como punto de entrada y salida para el tráfico que gestiona la función lambda.

Para comenzar, se debe agregar permisos a la función lambda para crear una interfaz de red para ello se agrega la siguiente política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": "*"
        }
    ]
}
```

Política 7. CreateNetwork, DeleteNetwork, DescribeNetwork lambda

Posteriormente, como se aprecia en la siguiente figura debemos integrar la función lambda en la subred privada.

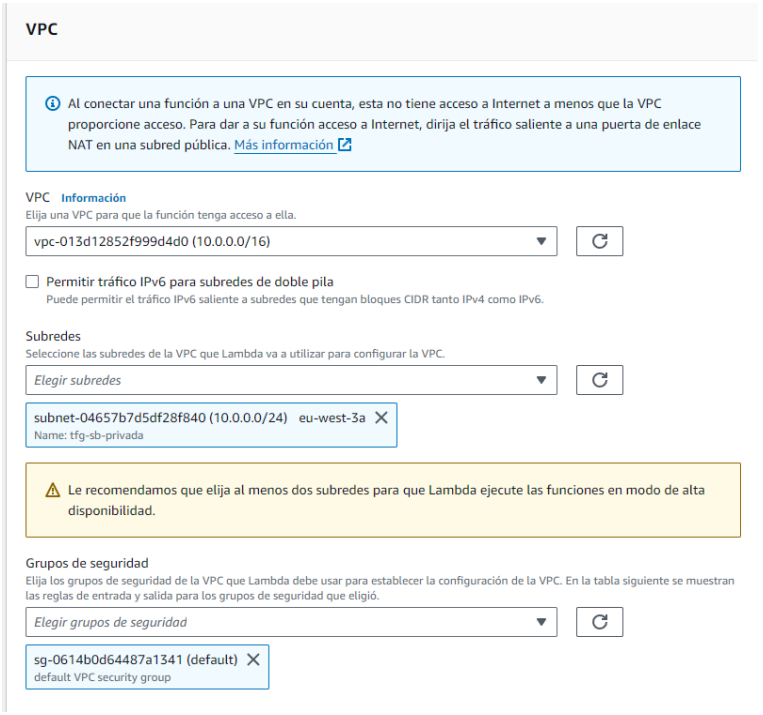

Figura 37. Integración Lambda-VPC

3.5 Uso de KMS para Cifrar Recursos

El objetivo de este capítulo será proteger los datos sensibles que se encuentran alojados en el DynamoDB y el Bucket S3 para evitar accesos no autorizados. Gracias a Key Management Service, se podrá hacer uso del cifrado en reposo para cifrar los datos.

En primer lugar, se crearán dos claves gestionadas por el cliente. Una para el DynamoDB y otra para el bucket S3. Como se aprecia en la siguiente figura, las claves serán de tipo simétrico, es decir, una única clave que se usa tanto para cifrar y descifrar mensajes entre el receptor y el emisor. Además, el uso de la clave también será simétrico por lo que sólo se usará para cifrar y descifrar.

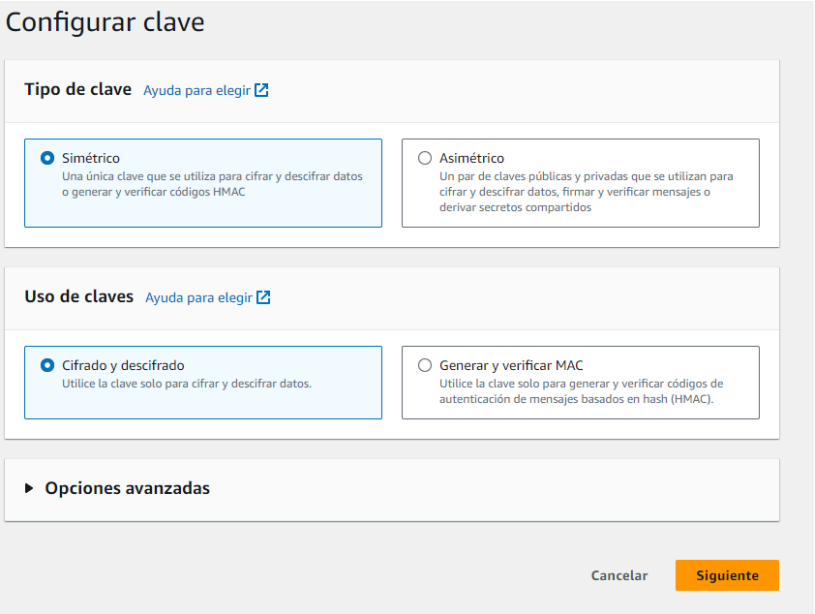

Figura 38. Creación clave KMS

Una vez implementadas ambas llaves, es necesario vincular dichas llaves con los servicios. Para ello, se debe elegir la siguiente opción de tipo de cifrado *Cifrado del servidor con claves de AWS Key Management Service (SSE-KMS).* De esta forma, cada vez que se introduce un objeto en el DynamoDB o Bucket se cifra automáticamente.

Por otro lado, cuando se ejecute el código de la función lambda POST surgirá los siguientes mensajes de error:

{"message": "An error occurred (AccessDeniedException) when calling the PutItem operation: KMS key access denied error: com.amazonaws.services.kms.model.AWSKMSException: User: arn:aws:sts::851725210341:assumed-role/tfg-lambda-POST-role-ztc9l1u9/tfg-lambda-POST is not authorized to perform: kms:Decrypt on resource: arn:aws:kms:eu-west-3:851725210341:key/25544ac0-ac44-48fa-815e-668c9d063977 because no identity-based policy allows the kms:Decrypt action (Service: AWSKMS; Status Code: 400; Error Code: AccessDeniedException; Request ID: 0f5eaee9-1a0c-4dde-8b95-fa06c1358778; Proxy: null)"}

{"message": "An error occurred (AccessDenied) when calling the PutObject operation: User: arn:aws:sts::851725210341:assumed-role/tfg-lambda-POST-role-ztc9l1u9/tfg-lambda-POST is not authorized to perform: kms:GenerateDataKey on resource:

arn:aws:kms:eu-west-3:851725210341:key/ca8f0c24-fc39-4cf1-b9f5-6c41f729fe4f because no identity-based policy allows the kms:GenerateDataKey action"}

Esto sucede porque la función lambda no tiene permisos para realizar kms:Decrypt ni para kms:GenerateDataKey, por lo que se agregarán las siguientes políticas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:eu-west-3:851725210341:key/*"
        }
    ]
}
```

Política 8. Kms:decrypt lambda-KMS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey",
            "Resource": "arn:aws:kms:eu-west-3:851725210341:key/*"
        }
    ]
}
```

Política 9. Kms:GenerateDataKey lambda-KMS

En cuanto a las funciones lambda para GET y DELETE se agregará también la política de kms:Decrypt.

Por último, el único usuario que puede hacer uso de la clave, es el usuario root por lo que cualquier otro IAM user o entidad no tendrá acceso a menos que se le asigne los permisos necesarios. Por lo que, si se intenta descargar un objeto del s3 sin ser root se producirá el siguiente error.

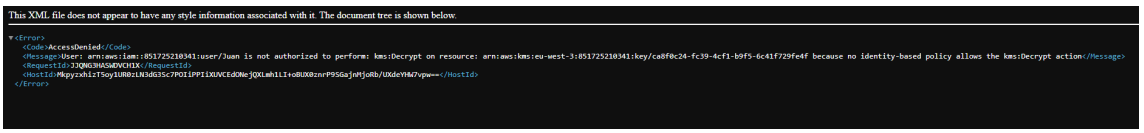

Figura 39. Acceso denegado KMS

3.6 Uso de IAM para las Identidades y Accesos

En este capítulo, se hará uso de IAM para gestionar de forma eficiente las identidades y accesos en la infraestructura desplegada anteriormente.

Con este fin, se hará uso de usuarios IAM, que representan personas o servicios que interactúan con AWS directamente. Además, se hará uso de grupos IAM para gestionar permisos a un grupo de usuarios IAM de forma conjunta. Asimismo, se emplearán roles IAM, a los cuales podemos asignar permisos de forma temporal. Por último, se hará uso de policías IAM, que se adjuntan sobre los usuarios, roles y grupos para definir qué acciones puede realizar en qué recursos.

En primer lugar, se crearán usuarios con roles que pueden incluir un desarrollador, administrador o lector. Los usuarios podrán iniciar sesión con sus respectivas credenciales y realizar las tareas que tengan asignadas para el proyecto en función de su rol. Esto garantiza que los usuarios tengan los permisos adecuados para sus responsabilidades, sin acceso a recursos que no necesitan modificar.

Como se aprecia en la siguiente figura se han creado cuatro usuarios sin permisos asignados.

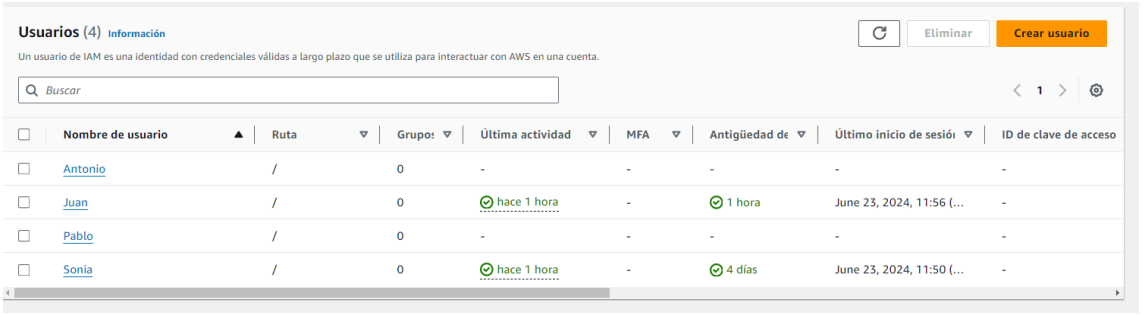

Figura 40. Usuarios IAM

Se hará uso de los grupos IAM para asignar permisos a varios usuarios a la vez.

Primeramente, se creará un grupo IAM dirigido a los desarrolladores para que tengan permisos para modificar los recursos implementados en capítulos anteriores, se agregará a este grupo los usuarios Pablo y Sonia.

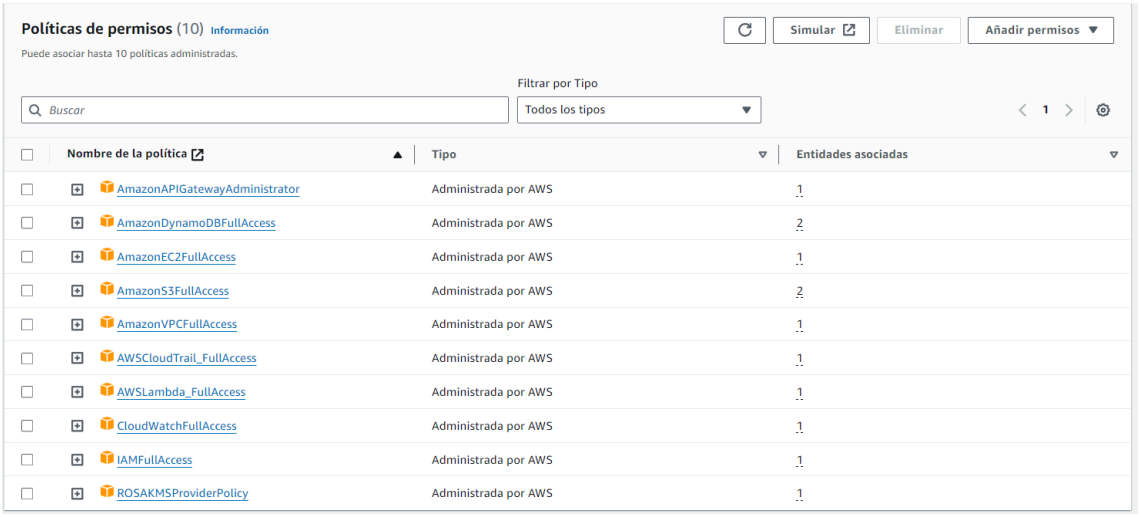

Figura 41. Grupo IAM desarrolladores

Como se aprecia en la figura anterior, se han asignado diez políticas a este grupo:

- AmazonAPIGatewayAdministrator
- AmazonDynamoDBFullAccess
- AmazonEC2FullAccess
- AmazonS3FullAccess
- AmazonVPCFullAccess
- AWSCloudTrail_FullAccess
- AWSLambda_FullAccess
- CloudWatchFullAccess
- IAMFullAccess
- ROSAKMSProviderPolicy

Estos roles permiten acceso completo a los recursos que van asociados.

En segundo lugar, se creará un grupo para que los usuarios, Juan y Antonio solo tengan permisos de lectura sobre los recursos.

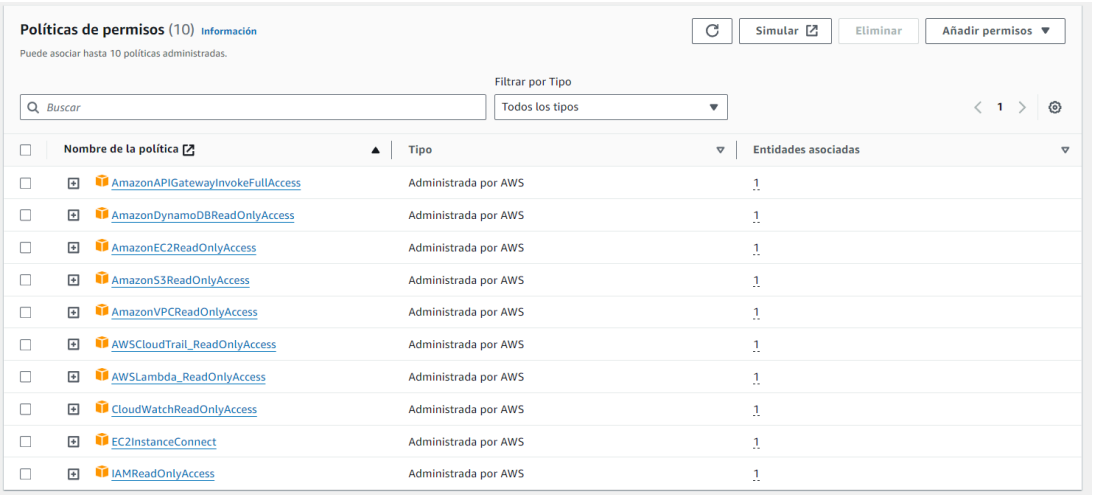

Figura 42. Grupo IAM lectores

Como se observa en la figura anterior, se han adjuntado las siguientes diez políticas a este grupo:

- AmazonAPIGatewayInvokeFullAccess
- AmazonDynamoDBReadOnlyAccess
- AmazonEC2ReadOnlyAccess
- AmazonS3ReadOnlyAccess
- AmazonVPCReadOnlyAccess
- AWSCloudTrail_ReadOnlyAccess
- AWSLambda_ReadOnlyAccess
- CloudWatchReadOnlyAccess
- EC2InstanceConnect
- IAMReadOnlyAccess

Por otro lado, se han creado los siguientes roles IAM para asignar permisos de forma temporal:

- API-Gateway
- CloudTrail
- CloudWatch
- DynamoDB
- EC2-VPC
- Lambda
- S3

Estos roles permiten acceso completo a los servicios relacionados.

Como se observa en la siguiente figura se debe introducir el id de la cuenta AWS y el nombre del rol para poder hacer uso del rol.

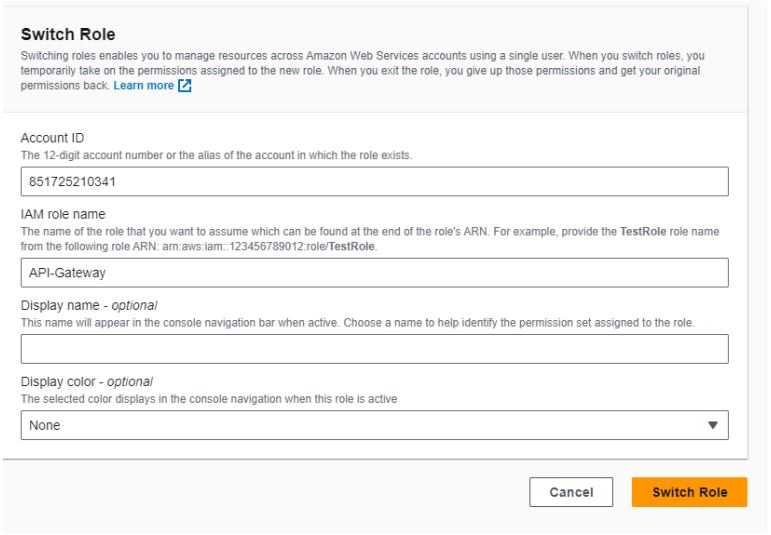

Figura 43. Cambio de rol IAM

Por último, hay que tener en cuenta que en capítulos anteriores se han utilizado políticas IAM para conceder permisos específicos. Por ejemplo, se ha permitido a una función Lambda insertar objetos en un bucket S3.

3.7 Monitorización de Recursos en AWS

En este capítulo, se hará uso de Amazon CloudTrail, Amazon CloudWatch y VPC Flow Logs para garantizar la monitorización de los recursos desplegados. De esta forma, se asegurará la disponibilidad, el rendimiento y la seguridad de los servicios.

3.7.1 Uso de VPC Flow Logs

Los VPC Flow Logs son una característica de AWS que permite capturar toda la información detallada sobre el tráfico IP que entra y que sale de los VPC. Por ende, gracias a estos registros se podrá establecer posibles problemas de conectividad de red, reglas de seguridad mal configuradas...[32]

Para crear un Flow Logs es necesario configurar qué tipo de tráfico se va a capturar, en esta ocasión se ha escogido el tráfico independientemente de si ha sido aceptado o no.

Además, el intervalo máximo de agregación está configurado en un minuto. Esto representa el tiempo máximo durante el cuál se captura y agrega un paquete en una historia de registro de flujo. Por último, los registros de flujo se almacenarán en un bucket S3 denominado "tfg-s3-logs".

En la siguiente figura se observa la configuración de los Flow Logs.

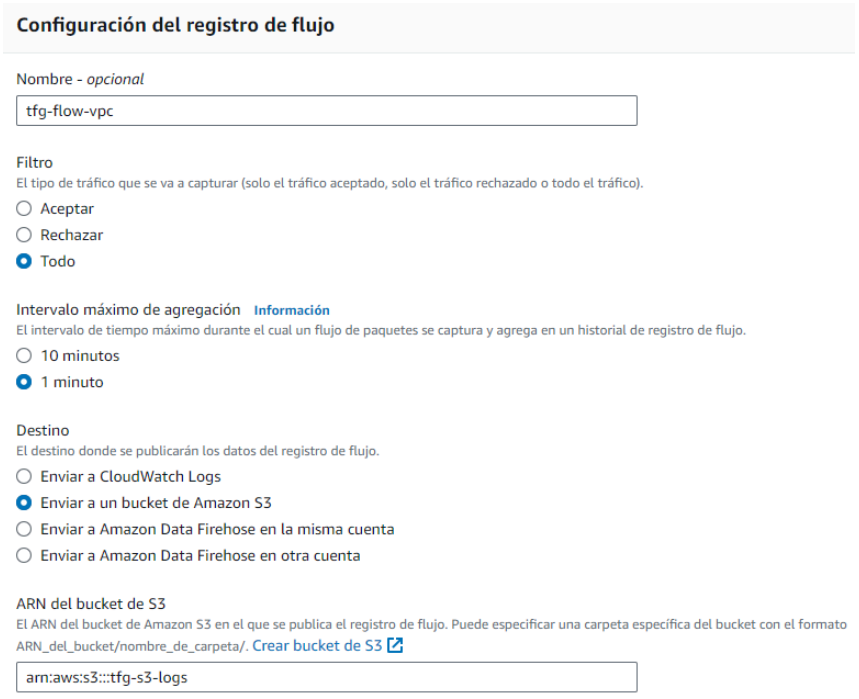

Figura 44. Creación VPC Flow Logs

Por último, en la figura adjunta abajo, se aprecia un ejemplo de un registro en el que podemos ver información muy valiosa como la dirección IP de inicio, la

dirección IP de destino, la interfaz de red, el puerto de origen y el destino, entre otros.

```
version account-id interface-id srcaddr dstaddr srcport dstport protocol packets bytes start end action log-status
2 851725210341 eni-06131c4faeb0c05ce - - - - - - - 1719157600 1719157631 - NODATA
2 851725210341 eni-0d9f2623810bbc366 - - - - - - - 1719157585 1719157616 - NODATA
2 851725210341 eni-0ec01b9f0c3edf71c 10.0.0.201 52.95.155.90 42106 443 6 2 80 1719157604 1719157604 ACCEPT OK
2 851725210341 eni-0ec01b9f0c3edf71c - - - - - - - 1719157582 1719157582 - NODATA
2 851725210341 eni-0ec01b9f0c3edf71c - - - - - - - 1719157582 1719157582 - SKIPDATA
2 851725210341 eni-0d9f2623810bbc366 111.19.191.28 10.0.1.113 0 0 1 1 34 1719157571 1719157572 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 165.154.182.92 10.0.1.113 43585 995 6 1 60 1719157622 1719157636 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 158.255.7.157 10.0.1.113 40164 2080 6 1 40 1719157622 1719157636 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 158.255.7.157 10.0.1.113 40164 62911 6 1 40 1719157620 1719157634 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 87.251.67.180 10.0.1.113 52291 8433 6 1 40 1719157620 1719157634 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 139.144.239.72 10.0.1.113 59723 91 6 1 44 1719157620 1719157634 ACCEPT OK
2 851725210341 eni-06131c4faeb0c05ce - - - - - - - 1719157612 1719157643 - NODATA
2 851725210341 eni-0ec01b9f0c3edf71c 3.5.226.172 10.0.0.201 443 55692 6 12 7013 1719157628 1719157630 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 - - - - - - - 1719157594 1719157625 - NODATA
2 851725210341 eni-0ec01b9f0c3edf71c 10.0.0.201 3.5.226.172 55692 443 6 2 104 1719157652 1719157654 ACCEPT OK
2 851725210341 eni-0ec01b9f0c3edf71c - - - - - - - 1719157628 1719157659 - NODATA
2 851725210341 eni-0ec01b9f0c3edf71c - - - - - - - 1719157640 1719157671 - NODATA
2 851725210341 eni-0d9f2623810bbc366 104.237.156.209 10.0.1.113 56999 7771 6 1 44 1719157671 1719157671 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 - - - - - - - 1719157624 1719157655 - NODATA
2 851725210341 eni-0d9f2623810bbc366 - - - - - - - 1719157654 1719157685 - NODATA
2 851725210341 eni-06131c4faeb0c05ce - - - - - - - 1719157660 1719157691 - NODATA
2 851725210341 eni-06131c4faeb0c05ce - - - - - - - 1719157625 1719157655 - NODATA
2 851725210341 eni-06131c4faeb0c05ce - - - - - - - 1719157623 1719157654 - NODATA
2 851725210341 eni-0d9f2623810bbc366 158.255.7.157 10.0.1.113 40164 1660 6 1 40 1719157639 1719157651 ACCEPT OK
2 851725210341 eni-0d9f2623810bbc366 64.62.197.117 10.0.1.113 52829 5901 6 1 40 1719157639 1719157651 ACCEPT OK
2 851725210341 eni-06131c4faeb0c05ce - - - - - - - 1719157611 1719157643 - NODATA
```

Figura 45. Registro VPC Flow Logs

3.7.2 Uso de Amazon CloudTrail

Gracias a CloudTrail será posible tener un registro detallado de las llamadas a la API de todos los recursos desplegados anteriormente. De este modo, se permite el monitoreo de la actividad y los cambios de la cuenta de AWS ya que CloudTrail registra quién hizo qué, cuándo y desde donde.

Para ello, los registros se almacenarán en un bucket S3 llamado “tfg-s3-logs”. Además, se puede integrar CloudTrail con CloudWatch permitiendo visualizar los registros en Amazon CloudWatch Logs, cómo se aprecia en la siguiente figura.

Figura 46. Creación CloudTrail

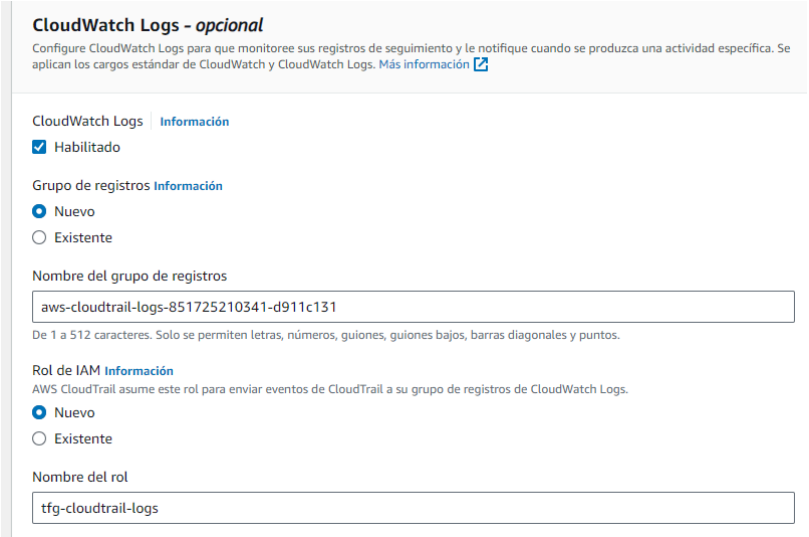

Figura 47. Creación integración CloudTrail-CloudWatch

Por otro lado, se seleccionan los tres tipos de eventos que existen:

- **Eventos de Administración:** Estos eventos registran todas las actividades realizadas sobre los recursos a nivel de administración.

  Por ejemplo, la operación de crear un bucket S3 ('CreateBucket') o la operación de listar buckets ('listBuckets').

- **Eventos de Datos:** Ofrecen registros a nivel de datos de servicios específicos de AWS. Son más detallados y se enfocan en las operaciones que acceden o modifican los datos dentro de los recursos. Por ejemplo, crear un objeto en un S3 ('PutObject') o eliminar un objeto en un S3 ('DeleteObject').
- **Eventos de Información General (Insight Events):** Son un subconjunto de los eventos de administración que se centran en identificar actividades inusuales.

Como se aprecia en la siguiente figura, se deben seleccionar en eventos de datos los servicios de S3 y DynamoDB y en los Insight Events se selecciona la tasa de llamadas a la API y tasa de error de la API.

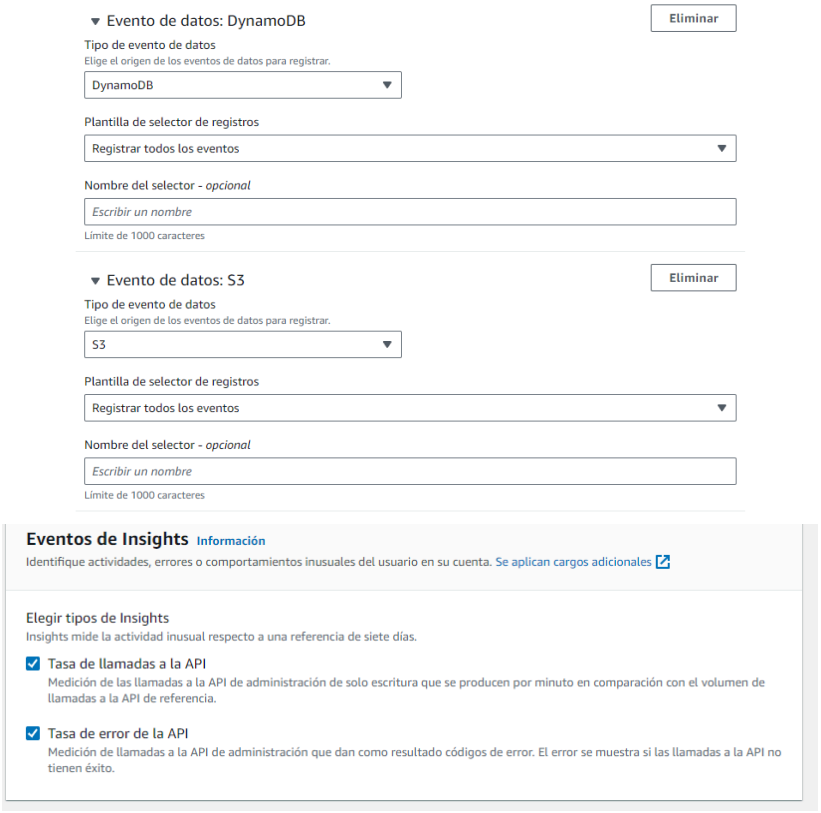

Figura 48. Eventos CloudTrail

Por último, podemos apreciar un registro de CloudTrail que representa la acción de lista buckets por parte del usuarios root. Algunos de los componentes son los siguientes: "arn": "arn:aws:iam::851725210341:root", "accountId": "851725210341", "awsRegion": "eu-west-3", "awsRegion": "eu-west-3", "eventSource": "s3.amazonaws.com" y "eventName": "ListBuckets".

Figura 49. Registro CloudTrail

3.7.3 Uso de Amazon CloudWatch

Por último, Amazon CloudWatch permite obtener una perspectiva global de toda la infraestructura ya que con ella, se puede recopilar y visualizar métricas, configurar alarmas y monitorizar archivos de registro.

Al crear una función Lambda, automáticamente se crea un grupo de registros donde se pueden visualizar todas las acciones basadas en cada una de las funciones cuando se recibe un evento mediante los registros. Estos grupos de registros son muy útiles a la hora encontrar fallos y excepciones, ya que en el caso de que se produzca un error quedará registrado.

A continuación, se muestra un ejemplo de un evento de registro:

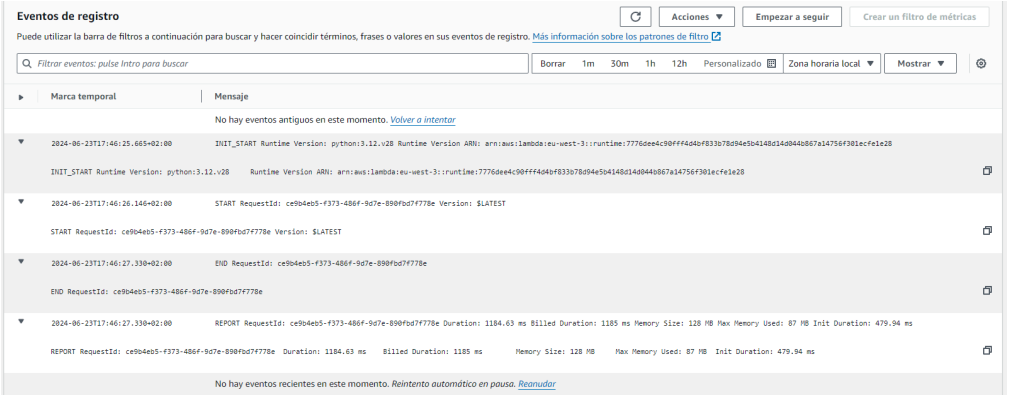

Figura 50. Registro CloudWatch

Por otro lado, como se ha nombrado anteriormente, CloudWatch permite la creación de alarmas y existen tres tipos de alarmas para cada función Lambda:

- **Invocaciones:** Esta alarma se activará cuando el número de invocaciones (cuando se ejecuta la función) supere un umbral específico.
- **Errores:** Esta alarma se disparará cuando el número de errores exceda un valor determinado.
- **Duración:** Esta alarma se activará cuando el tiempo de duración de una función Lambda supere un tiempo límite preestablecido.

Para ello, es necesario configurar el periodo y la condición de activación para cada tipo de alarma. En el caso de la alarma de **invocaciones**, se establece un periodo de 30 segundos y se activa si la función es invocada más de 5 veces en ese intervalo.

En cuanto a la alarma de **errores**, se define un periodo de 1 minuto y se dispara si se produce un error en más de una ocasión durante ese tiempo.

Y por último, para la alarma de **duración**, se configura un periodo de 1 minuto y se activa si la función permanece en funcionamiento durante más de 10 segundos.

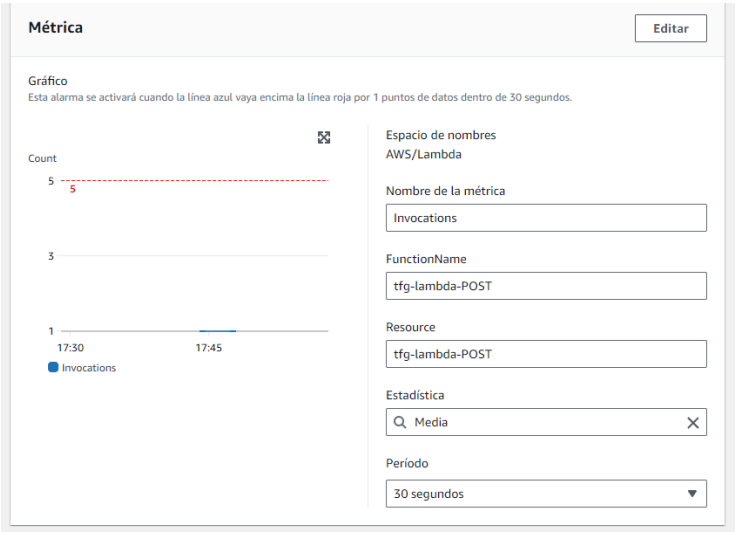

Figura 51. Creación alarma

Por último, se hará uso de Amazon SNS para recibir un correo electrónico en el caso de que se cumplan las condiciones de alarma.

3.8 Automatización de la Infraestructura Usando CloudFormation

Para finalizar, en este capítulo se utilizará AWS CloudFormation, un servicio proporcionado por AWS que ayuda a definir y aprovisionar recursos de infraestructura de manera predecible y automatizada. De esta forma, se podrá generar una plantilla de toda la infraestructura desplegada anteriormente, como se muestra en la siguiente figura.

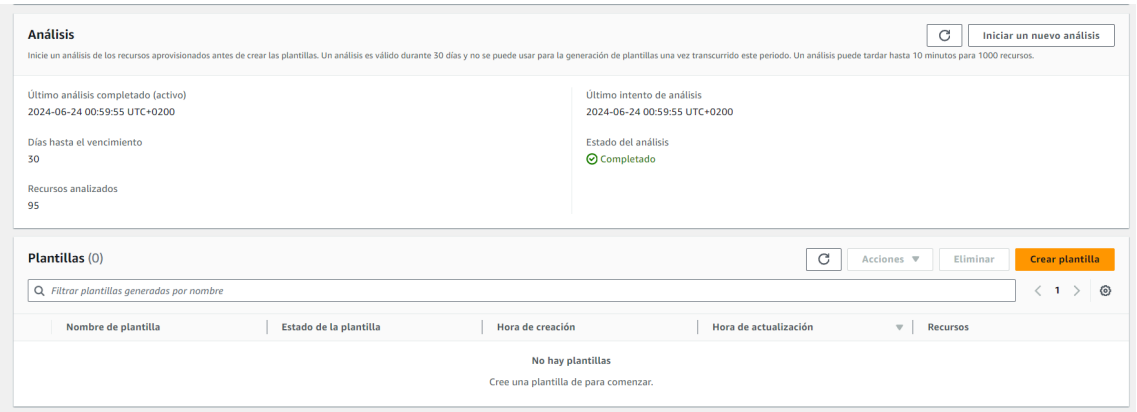

Figura 52. IaC Generator

Para ello, se utilizará la herramienta IaC Generator, un servicio de AWS que permite generar de forma automática una plantilla de toda la infraestructura desplegada.

En primer lugar, para crear la plantilla, se debe realizar un análisis que escanee todos los recursos existentes en la infraestructura. Una vez completado este análisis, podremos generar la plantilla seleccionando todos los recursos identificados.

Una vez que obtengamos la plantilla, se podrá crear un "stack" (pila), una colección de recursos de AWS que se pueden gestionar como una sola unidad. Todos los recursos en un stack se definen mediante una plantilla de CloudFormation. Los stacks facilitan la administración y el despliegue de grupos de recursos relacionados, permitiendo operaciones como la creación, actualización y eliminación de manera consistente y ordenada.

Como se aprecia en la siguiente figura, se pueden cargar plantillas ya creadas desde el ordenador. Posteriormente, se introducen los parámetros que requiere la plantilla y se despliega la pila. La pila se encarga de desplegar todos los recursos automáticamente.

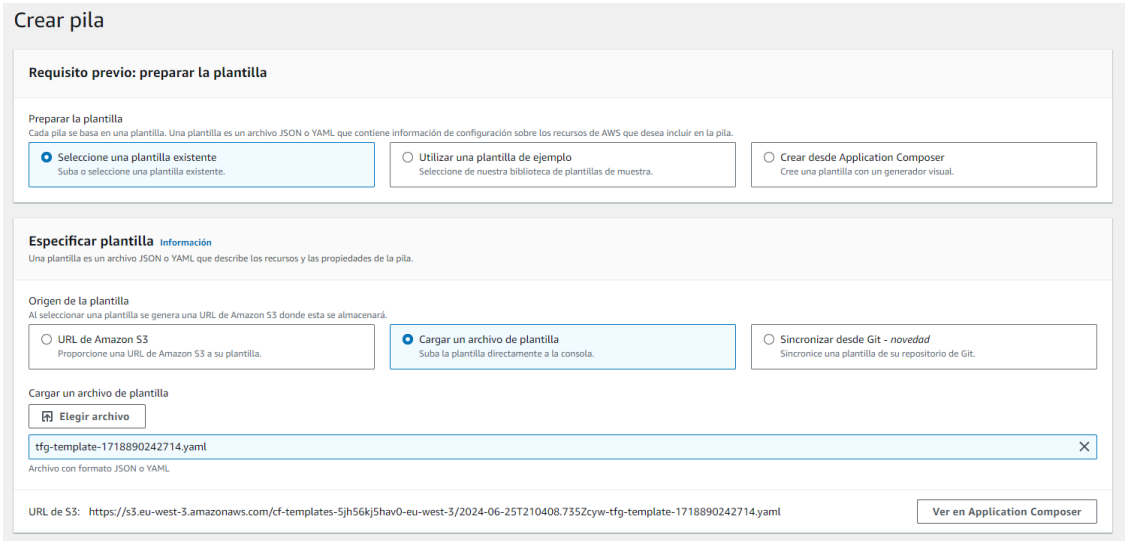

Figura 53. Crear pila

4 Casos Prueba

El objetivo de este capítulo es desarrollar una serie de casos prueba para garantizar el correcto funcionamiento de la infraestructura.

En primer lugar, se realizará la creación de dos libros. El primero de ellos “El Quijote” de “Miguel de Cervantes” y el segundo “El fantasma de la Ópera” de “Gaston Leroux”. Como se aprecia en las siguientes figuras las dos pruebas han devuelto 201 CREATED por lo que se han creado correctamente.

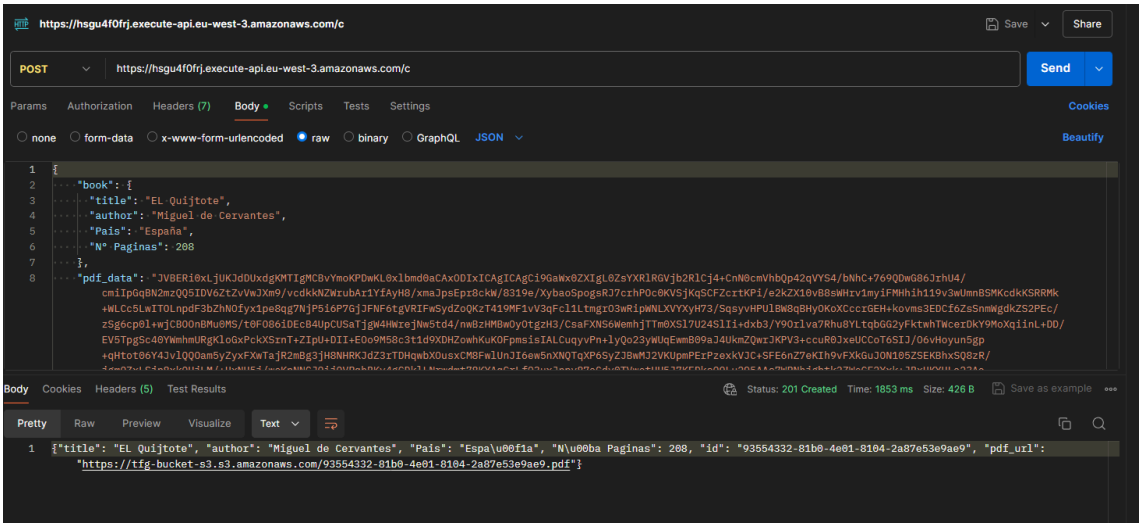

Figura 54. Creación El Quijote

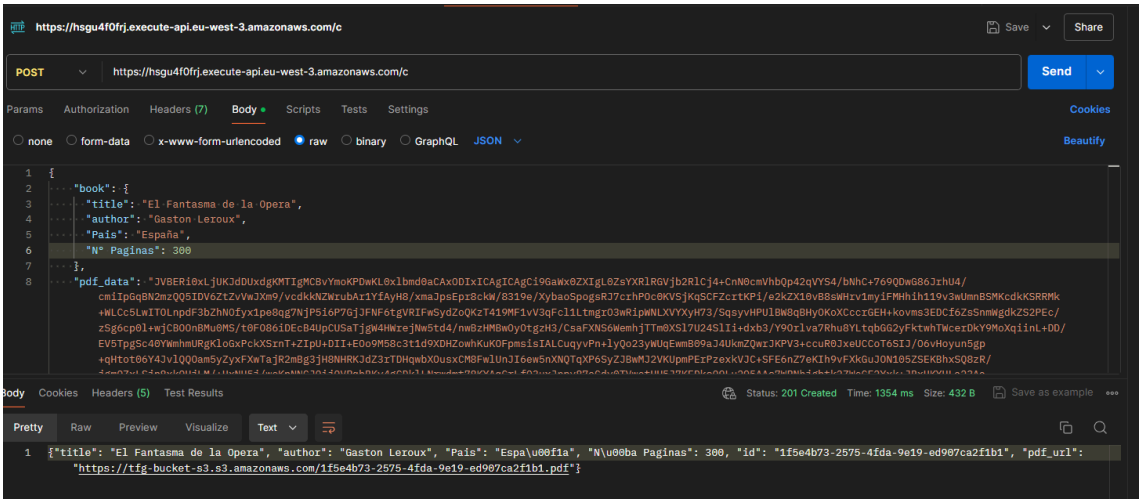

Figura 55. Creación El Fantasma de la Opera

Para comprobar que verdaderamente se han creado se ha revisado el contenido del Bucket S3 y del DynamoDB como se aprecia en las siguiente figuras.

Figura 56. Contenido DynamoDB

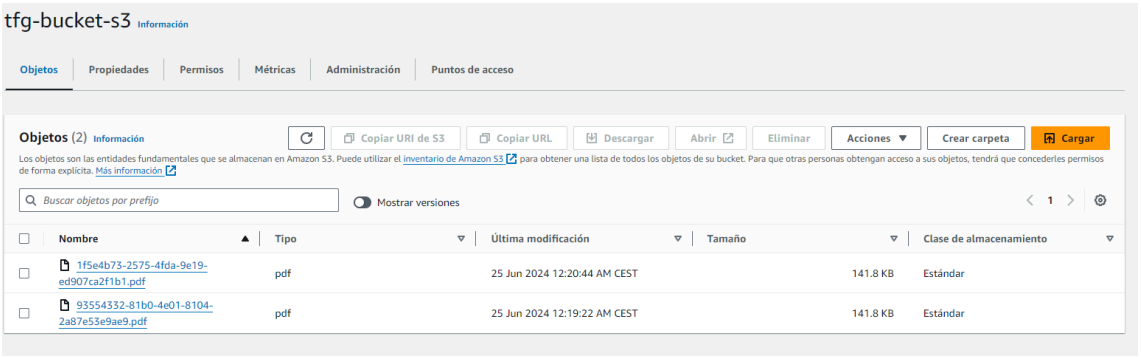

Figura 57. Contenido Bucket S3

Por otro lado, se ha probado la funcionalidad de obtener libros de la biblioteca. Como se aprecia en las siguientes figuras se han obtenido correctamente ya que ha devuelto 200 OK junto a la información en JSON.

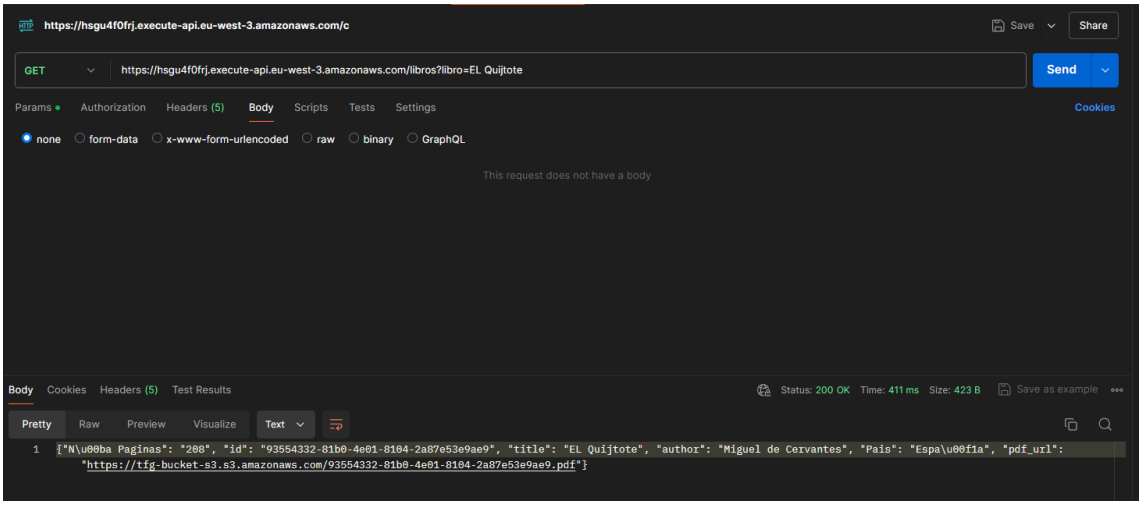

Figura 58. Prueba GET El Quijote

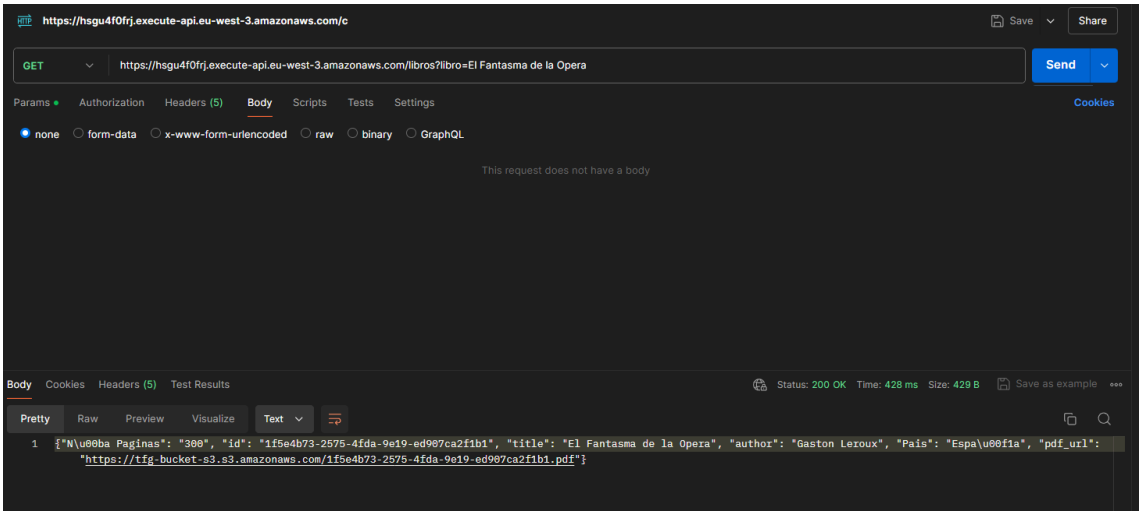

Figura 59. Prueba GET El Fantasma de la Opera

Además, se ha comprobado que la función de borrar libros funcione correctamente. Como se aprecia en la siguiente figuras ha funcionado de forma correcta ya que nos ha devuelto 200 OK.

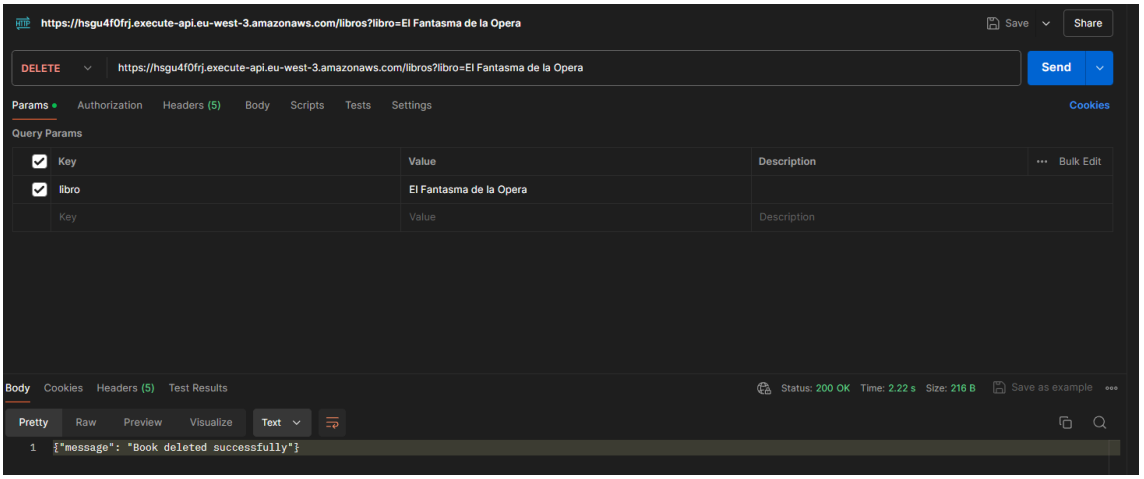

Figura 60. Prueba DELETE El Fantasma de la Opera

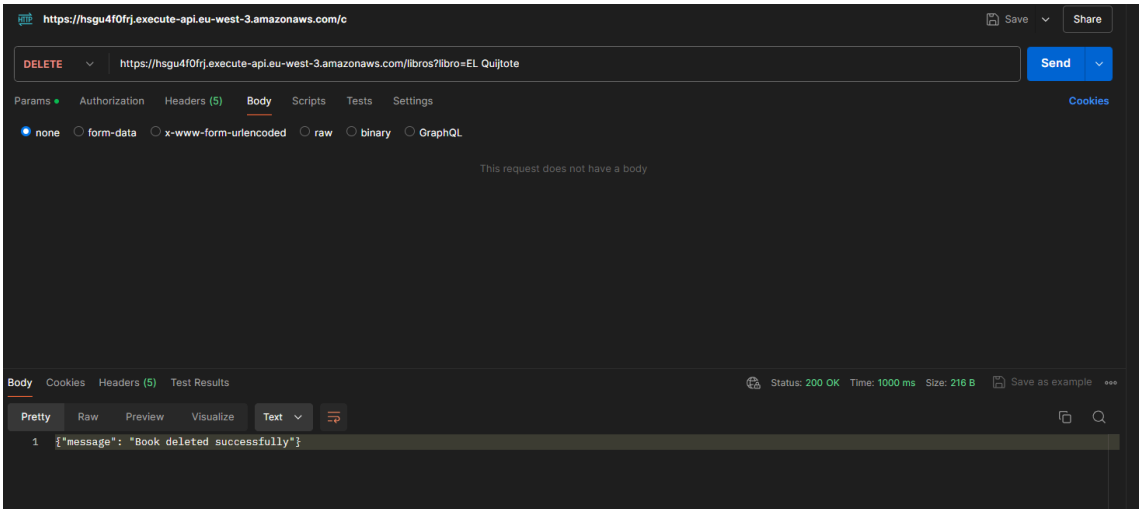

Figura 61. Prueba DELETE El Quijote

Finalmente, se ha comprobado que en efecto el Bucket S3 y el DynamoDB están vacíos.

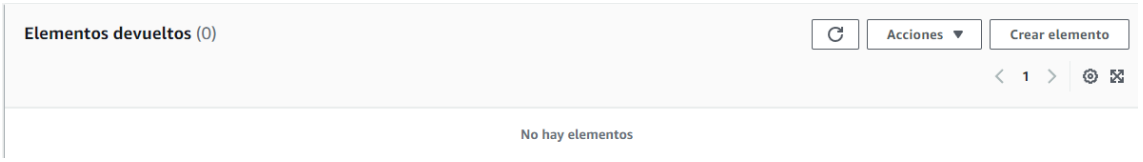

Figura 62. Contenido DynamoDB método borrar

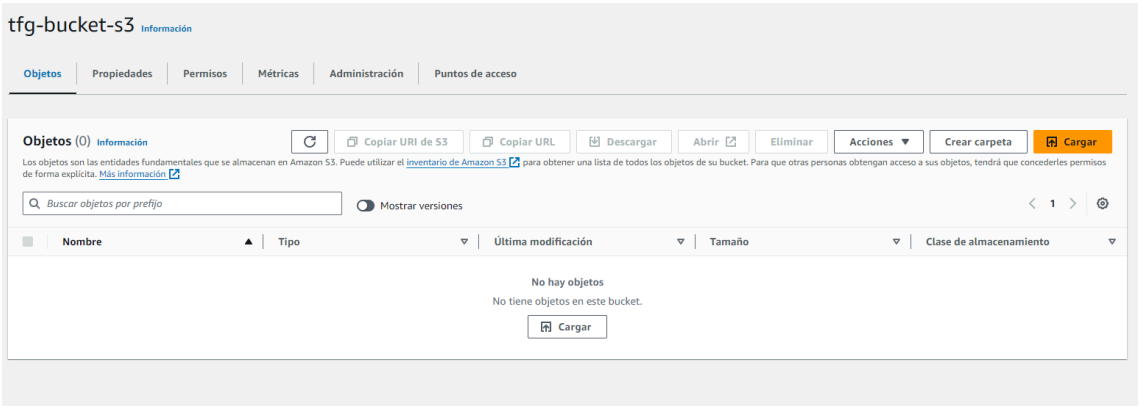

Figura 63. Contenido Bucket S3 método borrar

5 Resultados y conclusiones

La planificación y elaboración de infraestructura en la nube utilizando AWS me ha permitido familiarizarme con diversos servicios en la nube, enfocándome especialmente en aquellos que se encargan de monitorear y securizar la infraestructura en AWS, tales como AWS CloudWatch, AWS CloudTrail y VPC. Estos servicios son cruciales para mantener una infraestructura segura y eficiente, ya que proporcionan herramientas para la vigilancia constante y la protección de los recursos en la nube.

A través de este proyecto, he comprendido la importancia de implementar fuertes medidas de seguridad en la nube. En un entorno donde las amenazas cibernéticas están en constante evolución, asegurar la infraestructura en la nube es vital para prevenir posibles brechas de seguridad. La configuración de políticas de Identity and Access Management (IAM) y el uso de Key Management Service (KMS) para el cifrado de datos han demostrado ser esenciales para proteger la información sensible.

La realización del caso práctico ha sido una experiencia reveladora. Configurar y desplegar una API utilizando servicios como DynamoDB, S3, Lambda y API Gateway me ha permitido explorar las capacidades de AWS para construir soluciones seguras. Este proceso me ha enseñado cómo diseñar arquitecturas que no solo cumplen con los requisitos funcionales, sino que también incorporan mejores prácticas de seguridad y monitorización.

Además, la automatización de la infraestructura con CloudFormation ha simplificado la gestión y el despliegue de recursos. Esta herramienta ha facilitado la creación de plantillas para replicar la infraestructura, asegurando consistencia y reduciendo el margen de error humano.

La monitorización ha sido muy importante en este proyecto. A través de AWS CloudWatch y CloudTrail, he podido observar en tiempo real las actividades y el rendimiento de los recursos desplegados. Esta visibilidad me ha permitido identificar y resolver problemas de manera proactiva, garantizando así un funcionamiento óptimo de la infraestructura.

Finalmente, este proyecto ha reforzado mi interés en el campo de la computación en la nube. La nube ofrece innumerables oportunidades para innovar y mejorar la eficiencia operativa tanto para individuos como para empresas. La experiencia adquirida me ha motivado a seguir desarrollándome en esta área y a explorar nuevas soluciones para los desafíos emergentes en la seguridad y la monitorización en la nube.

5.1 Futuras líneas de desarrollo

En este capítulo se explorarán las posibles mejoras futuras que puede tener la infraestructura desplegada anteriormente. A continuación, se detallan algunas de las propuestas más relevantes:

En primer lugar, se podría añadir el método PUT para permitir la actualización de los libros en la biblioteca digital. Esta funcionalidad mejoraría significativamente la gestión y mantenimiento de la base de datos de libros, permitiendo cambios y actualizaciones de manera eficiente.

Por otro lado, es fundamental reforzar la seguridad en el acceso a los recursos en la nube. Para ello, se puede activar la autenticación multifactor (MFA), lo cual proporcionará una capa adicional de seguridad, asegurando que solo los usuarios autorizados puedan acceder a los recursos.

6 Análisis de Impacto

En este apartado tendrá como objetivo analizar el potencial impacto de este trabajo de fin de grado en las siguientes áreas.

Contexto personal:

La realización de este TFG ha sido de gran ayuda para mi desarrollo personal y académico, ya que me ha permitido adquirir nuevos conocimientos sobre una tecnología emergente como es la nube. A través del desarrollo del caso práctico, he analizado y tomado conciencia de las vulnerabilidades a las que se enfrentan las empresas en sus infraestructuras diariamente. Y además, me ha proporcionado los conocimientos básicos y necesarios para poder mitigar estas vulnerabilidades.

Contexto empresarial:

Este caso práctico beneficiará a muchas empresas ayudándoles a asegurar y monitorear sus infraestructuras en la nube, área que se ha convertido, en los últimos años, de gran interés para las organizaciones empresariales.

Contexto económico:

Este trabajo contribuirá significativamente a la reducción de las pérdidas económicas ocasionadas por posibles ciberataques, proporcionando varias soluciones efectivas para proteger las infraestructuras empresariales.

Contexto medioambiental:

Por último, este trabajo podrá ayudar a disminuir la huella de carbono al fomentar el uso compartido de centros de datos en la nube, disminuyendo que las empresas tengan que mantener sus propias infraestructuras.

7 Bibliografía

[1] S. Agencia, «On-premise», *YUV TV*, 8 de mayo de 2021. https://www.yuvtv.com/glosario/on-premise/#:~:text=On%2Dpremises%20significa%20en%20espa%C3%B1ol,inform%C3%A1tico%20propios%20de%20la%20empresa

[2] «¿Qué es la seguridad en la nube?», www.kaspersky.es, 27 de noviembre de 2023. https://www.kaspersky.es/resource-center/definitions/what-is-cloud-security

[3] NexusAdmistraIntegra, «Cloud monitoring, un sistema de monitorización remota», Nexus Integra, 28 de diciembre de 2021. https://nexusintegra.io/es/cloud-monitoring/

[4] E. C. Perú y R. el C. Perú, «Ciberataques a la nube aumentaron un 95% y se triplicó el número de atacantes online en 2022», El Comercio Perú, 6 de marzo de 2023. [En línea]. Disponible en: https://elcomercio.pe/tecnologia/actualidad/ciberataques-a-la-nube-aumentaron-un-95-y-se-triplico-el-numero-de-atacantes-online-en-2022-ciberdelincuentes-malware-espana-mexico-estados-unidos-noticia

[5] T. Nassi, "Introducción a cloud computing," Akamai, Aug. 01, 2023. https://www.linode.com/es/blog/cloud-computing/introduction-to-cloud-computing/

[6] https://www.areatecnologia.com, «Cloud Computing Todo lo que Necesitas Saber Facil». https://www.areatecnologia.com/informatica/cloud-computing.html

[7] Beservices, «El concepto de Elasticidad y el Cloud Computing», *El concepto de Elasticidad y el Cloud Computing*, 19 de octubre de 2022. https://blog.beservices.es/blog/el-concepto-de-elasticidad-el-cloud-computing

[8] Next U, «Computación en la nube: Conoce sus características», *Blog | NextU LATAM*, 9 de septiembre de 2022. https://www.nextu.com/blog/computacion-en-la-nube-rc22/

[9] «PaaS, IaaS y SaaS: ¿en qué se diferencian? | Google Cloud», *Google Cloud*. https://cloud.google.com/learn/paas-vs-iaas-vs-saas?hl=es

[10] "Diferencias entre nube pública, nube privada y nube híbrida | Microsoft Azure." https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-are-private-public-hybrid-clouds

[11] «Design principles - AWS Well-Architected Framework». https://docs.aws.amazon.com/wellarchitected/latest/framework/sec-design.html

[12] Diego.Coder, “Fundamentos de AWS IAM (Identity and Access Management),” *Medium*, Dec. 09, 2023. [Online]. Available: https://medium.com/@diego.coder/introducci%C3%B3n-a-aws-iam-9d4d8a89fa46

[13] Diego.Coder, «Fundamentos de Amazon VPC (Virtual Private Cloud) - diego.coder26 - Medium», *Medium*, 19 de diciembre de 2023. [En línea]. Disponible en: https://medium.com/@diego.coder/introducci%C3%B3n-a-aws-vpc-amazon-virtual-private-cloud-a8e8bd614e24

[14] «AWS Key Management Service - AWS Key Management Service». https://docs.aws.amazon.com/es_es/kms/latest/developerguide/overview.html#:~:text=AWS%20Key%20Management%20Service%20(AWS,utilizan%20para%20proteger%20sus%20datos.

[15] «Infrastructure Monitoring with Amazon CloudWatch», *Google Books*. https://books.google.es/books?hl=es&lr=&id=oI0kEAAAQBAJ&oi=fnd&pg=PP1&dq=monitoring+in+aws&ots=QLNFilwu5u&sig=UcqNeApRrEj3iO9s4WFNzR44DJc#v=onepage&q&f=false

[16] C. Inc, “Monitorización de Amazon Web Services,” May 24, 2022. https://techdocs.broadcom.com/es/es/ca-enterprise-software/it-operations-management/dx-apm-saas/SaaS/implementing-agents/infrastructure-agent/Amazon-Web-Services-Monitoring.html

[17] «¿Qué es Amazon SNS? - Amazon Simple Notification Service». https://docs.aws.amazon.com/es_es/sns/latest/dg/welcome.html

[18] «¿Qué es Amazon CloudWatch? - Amazon CloudWatch». https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

[19] «Using Amazon CloudWatch dashboards - Amazon CloudWatch». https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Dashboards.htm

[20] «Uso de las alarmas de Amazon CloudWatch - Amazon CloudWatch». https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html

[21] Diego.Coder, «Fundamentos de AWS CloudTrail - diego.coder26 - Medium», *Medium*, 9 de diciembre de 2023. [En línea]. Disponible en: https://medium.com/@diego.coder/introducci%C3%B3n-a-aws-cloudtrail-e242c4ee5e65

[22] «¿Qué es Amazon S3? - Amazon Simple Storage Service». https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/Welcome.html

[23] «¿Qué es Amazon DynamoDB? - Amazon DynamoDB». https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/Introduction.html

[24] «¿Qué es Amazon API Gateway? - Amazon API Gateway». https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html

[25] «Qué es AWS Lambda y cómo funciona», *IfgeekthenNTTdata*, 23 de agosto de 2023. https://ifgeekthen.nttdata.com/es/que-es-aws-lambda-y-como-funciona

[26] «¿Qué es AWS CloudFormation? - AWS CloudFormation». https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/Welcome.html

[27] «Líderes en formación tecnológica, reskilling y upskilling | OpenWebinars», *OpenWebinars.net*. https://openwebinars.net/blog/que-es-serverless-ventajas-y-servicios/

[28] «¿Qué es una base de datos NoSQL? | IBM». https://www.ibm.com/es-es/topics/nosql-databases

[29] «Bloquear el acceso público a su almacenamiento de Amazon S3 - Amazon Simple Storage Service». https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/access-control-block-public-access.html

[30] «Using versioning in S3 buckets - Amazon Simple Storage Service». https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html?icmpid=docs_amazons3_console

[31] «Boto3 1.34.131 documentation». https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[32] «Registro del tráfico de IP con registros de flujo de la VPC - Amazon Virtual Private Cloud». https://docs.aws.amazon.com/es_es/vpc/latest/userguide/flow-logs.html

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Sun Jun 30 21:09:05 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)