AWS Documento técnico

Implementación de microservicios en AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Implementación de microservicios en AWS: AWS Documento técnico

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon, de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

Resumen e introducción i
Introducción 1
¿Dispone de Well-Architected? 2
Modernización a microservicios 3
Arquitectura de microservicios activada AWS 4
Interfaz de usuario 5
Microservicios 5
Implementaciones de microservicios 5
CI/CD 6
Red privada 7
Almacén de datos 7
Simplificar las operaciones 8
Implementación de aplicaciones basadas en Lambda 8
Resumiendo las complejidades de la multitenencia 9
Administración de API 10
Arquitectura de microservicios sin servidor 12
Sistemas resilientes y eficientes 15
Recuperación de desastres (DR) 15
Alta disponibilidad (HA) 15
Componentes de sistemas distribuidos 17
Administración de datos distribuidos 19
Administración de la configuración 22
Administración de secretos 22
Optimización de costes y sostenibilidad 23
Mecanismos de comunicación 24
Comunicación basada en REST 24
Comunicación basada en GraphQL 24
Comunicación basada en gRPC 24
Mensajería asíncrona y transmisión de eventos 24
Organización y administración del estado 27
Observabilidad 30
Supervisión 30
Centralización de registros 32
Rastreo distribuido 33

Análisis de registros en AWS 34
Otras opciones de análisis 34
Gestión de la comunicación de microservicios 37
Uso de protocolos y almacenamiento en caché 37
Auditoría 38
Inventario de recursos y administración de cambios 38
Conclusión 40
Colaboradores 41
Historial del documento 42
Avisos 44
AWS Glosario 45
..... xlvi

Implementación de microservicios en AWS

Fecha de publicación: 31 de julio de 2023 () Historial del documento

Los microservicios ofrecen un enfoque simplificado para el desarrollo de software que acelera la implementación, fomenta la innovación, mejora la capacidad de mantenimiento e impulsa la escalabilidad. Este método se basa en servicios pequeños y poco acoplados que se comunican a través de equipos bien definidos APIs y gestionados por equipos autónomos. La adopción de microservicios ofrece beneficios, como una mayor escalabilidad, resiliencia y flexibilidad y ciclos de desarrollo más rápidos.

En este documento técnico se analizan tres patrones de microservicios populares: los basados en las API, los basados en eventos y el streaming de datos. Proporcionamos una visión general de cada enfoque, describimos las características clave de los microservicios, abordamos los desafíos de su desarrollo e ilustramos cómo Amazon Web Services (AWS) puede ayudar a los equipos de aplicaciones a superar estos obstáculos.

Teniendo en cuenta la naturaleza compleja de temas como el almacenamiento de datos, la comunicación asincrónica y la detección de servicios, le recomendamos que evalúe las necesidades específicas y los casos de uso de su aplicación junto con la orientación proporcionada al tomar decisiones arquitectónicas.

Introducción

Las arquitecturas de microservicios combinan conceptos exitosos y comprobados de varios campos, como:

- Desarrollo de software ágil
- Arquitecturas orientadas a servicios
- Diseño centrado en las API
- (Continuo) Integration/Continuous Delivery (CI/CD

A menudo, los microservicios incorporan patrones de diseño de la aplicación Twelve-Factor.

Si bien los microservicios ofrecen muchos beneficios, es fundamental evaluar los requisitos únicos de su caso de uso y los costos asociados. La arquitectura monolítica o los enfoques alternativos pueden ser más apropiados en algunos casos. La decisión entre microservicios o monolitos debe

hacerse sobre una case-by-case base, teniendo en cuenta factores como la escala, la complejidad y los casos de uso específicos.

En primer lugar, exploramos una arquitectura de microservicios altamente escalable y tolerante a fallos (interfaz de usuario, implementación de microservicios, almacén de datos) y demostramos cómo desarrollarla mediante tecnologías de contenedores. AWS A continuación, sugerimos AWS servicios para implementar una arquitectura típica de microservicios sin servidor, lo que reduce la complejidad operativa.

La tecnología sin servidor se caracteriza por los siguientes principios:

- No hay infraestructura que aprovisionar o administrar
- Escalado automático por unidad de consumo
- Modelo de facturación de «pago por valor»
- Disponibilidad y tolerancia a fallos integradas
- Arquitectura impulsada por eventos (EDA)

Por último, examinamos el sistema en general y analizamos los aspectos interservicios de una arquitectura de microservicios, como la supervisión distribuida, el registro, el rastreo, la auditoría, la coherencia de los datos y la comunicación asincrónica.

Este documento se centra en las cargas de trabajo que se ejecutan en el entorno, excluyendo los escenarios híbridos y las estrategias de Nube de AWS migración. Para obtener información sobre las estrategias de migración, consulte el documento técnico sobre la metodología de migración de contenedores.

¿Dispone de Well-Architected?

El marco de AWS Well-Architected le ayuda a entender las ventajas y desventajas de las decisiones que toma al crear sistemas en la nube. Los seis pilares del marco le permitirán aprender las prácticas recomendadas de arquitectura para diseñar y utilizar sistemas fiables, seguros, eficientes, rentables y sostenibles. Mediante AWS Well-Architected Tool, disponible sin costo alguno en la Consola de administración de AWS, puede comparar las cargas de trabajo con estas prácticas recomendadas respondiendo a una serie de preguntas para cada pilar.

Desde el punto de vista de las aplicaciones sin servidor, nos centramos en las mejores prácticas para diseñar la arquitectura de sus aplicaciones sin servidor. AWS

Para obtener asesoramiento más experto y conocer las prácticas recomendadas para la arquitectura en la nube (implementaciones de arquitectura de referencia, diagramas y documentos técnicos), consulte el Centro de arquitectura de AWS.

Modernización a microservicios

Los microservicios son básicamente unidades pequeñas e independientes que componen una aplicación. La transición de las estructuras monolíticas tradicionales a los microservicios puede seguir varias estrategias.

Esta transición también afecta a la forma en que opera su organización:

- Fomenta un desarrollo ágil, en el que los equipos trabajan en ciclos rápidos.
- Los equipos suelen ser pequeños, y a veces se los describe como dos equipos de pizza, lo suficientemente pequeños como para que dos pizzas puedan dar de comer a todo el equipo.
- Los equipos asumen toda la responsabilidad de sus servicios, desde la creación hasta la implementación y el mantenimiento.

Arquitectura de microservicios sencilla en AWS

Las aplicaciones monolíticas típicas constan de diferentes capas: una capa de presentación, una capa de aplicación y una capa de datos. Las arquitecturas de microservicios, por otro lado, separan las funcionalidades en verticales cohesivas según dominios específicos, en lugar de capas tecnológicas. La figura 1 ilustra una arquitectura de referencia para una aplicación de microservicios típica en. AWS

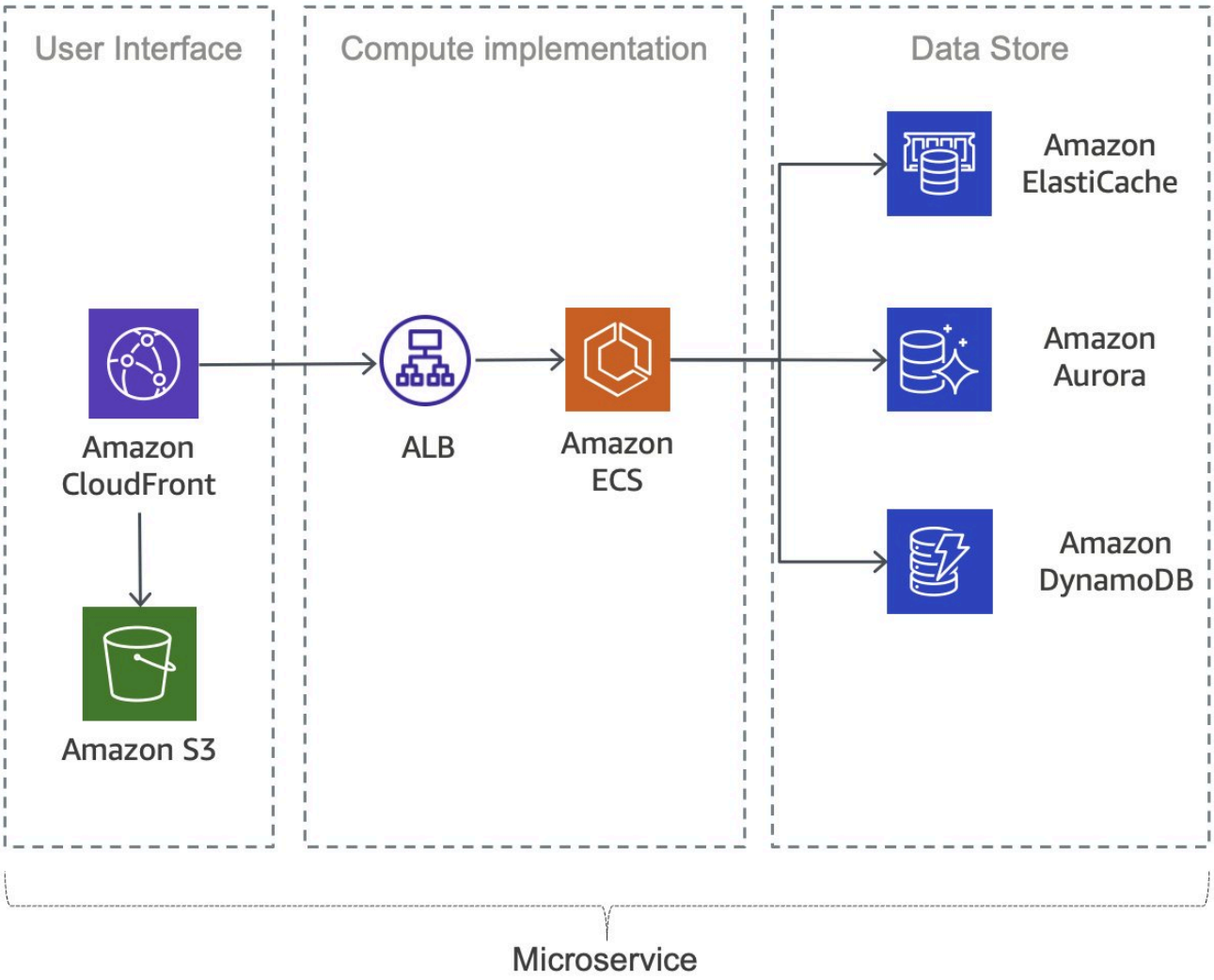

Figura 1: Aplicación de microservicios típica en AWS

Interfaz de usuario

Las aplicaciones web modernas suelen utilizar JavaScript marcos para desarrollar aplicaciones de una sola página que se comunican con el backend APIs. Por lo general, APIs se crean mediante transferencia de estado representacional (REST) o RESTful APIs APIs GraphQL. El contenido web estático se puede servir mediante Amazon Simple Storage Service (Amazon S3) y Amazon CloudFront.

Microservicios

APIs se consideran la puerta principal de los microservicios, ya que son el punto de entrada a la lógica de las aplicaciones. Por lo general, se utilizan la API de servicios RESTful web o GraphQL APIs . Estos APIs gestionan y procesan las llamadas de los clientes y gestionan funciones como la gestión del tráfico, el filtrado de solicitudes, el enrutamiento, el almacenamiento en caché, la autenticación y la autorización.

Implementaciones de microservicios

AWS ofrece componentes básicos para desarrollar microservicios, incluidos Amazon ECS y Amazon EKS como opciones para los motores de organización de contenedores AWS Fargate y EC2 como opciones de alojamiento. AWS Lambda es otra forma de crear microservicios sin servidores. AWS La elección entre estas opciones de alojamiento depende de los requisitos del cliente para administrar la infraestructura subyacente.

AWS Lambda le permite cargar su código, escalar automáticamente y gestionar su ejecución con alta disponibilidad. Esto elimina la necesidad de administrar la infraestructura, por lo que puede avanzar con rapidez y centrarse en la lógica empresarial. Lambda admite varios lenguajes de programación y se puede activar mediante otros AWS servicios o se puede llamar directamente desde aplicaciones web o móviles.

Las aplicaciones basadas en contenedores han ganado popularidad debido a su portabilidad, productividad y eficiencia.AWS ofrece varios servicios para construir, implementar y administrar contenedores.

- App2Container, una herramienta de línea de comandos para migrar y modernizar aplicaciones web Java y .NET al formato de contenedor. AWS A2C analiza y crea un inventario de las aplicaciones que se ejecutan en equipos físicos, máquinas virtuales, instancias de Amazon Elastic Compute Cloud (EC2) o en la nube.

- Amazon Elastic Container Service (Amazon ECS) y Amazon Elastic Kubernetes Service (Amazon EKS) administran la infraestructura de contenedores, lo que facilita el lanzamiento y el mantenimiento de aplicaciones en contenedores.
  - Amazon EKS es un servicio de Kubernetes gestionado para ejecutar Kubernetes en la AWS nube y en centros de datos locales (Amazon EKS Anywhere). Esto amplía los servicios en la nube a los entornos locales para el procesamiento de datos local de baja latencia, los altos costos de transferencia de datos o los requisitos de residencia de datos (consulte el documento técnico sobre la ejecución de cargas de trabajo en contenedores híbridos con Amazon EKS Anywhere). Con EKS, puede utilizar todos los complementos y herramientas existentes de la comunidad de Kubernetes.
  - Amazon Elastic Container Service (Amazon ECS) es un servicio de organización de contenedores totalmente gestionado que simplifica la implementación, la administración y el escalado de las aplicaciones en contenedores. Los clientes eligen ECS por su simplicidad y su profunda integración con los servicios. AWS

Para obtener más información, consulte el blog Amazon ECS vs Amazon EKS: making sense of AWS container services.

- AWS App Runneres un servicio de aplicaciones de contenedores totalmente gestionado que le permite crear, implementar y ejecutar aplicaciones web y servicios de API en contenedores sin necesidad de experiencia previa en infraestructura o contenedores.
- AWS Fargate, un motor de cómputo sin servidor, funciona tanto con Amazon ECS como con Amazon EKS para administrar automáticamente los recursos de cómputo de las aplicaciones de contenedores.
- Amazon ECR es un registro de contenedores totalmente gestionado que ofrece alojamiento de alto rendimiento, por lo que puede implementar imágenes y artefactos de aplicaciones de forma fiable en cualquier lugar.

Integración e implementación continuas (CI/CD)

La integración y la entrega continuas (CI/CD) son una parte crucial de una DevOps iniciativa para realizar cambios rápidos en el software. AWS ofrece servicios CI/CD para implementar microservicios, pero un análisis detallado va más allá del alcance de este documento. Para obtener más información, consulte el documento AWS técnico Practicar la integración continua y la entrega continua en formato práctico.

Red privada

AWS PrivateLink es una tecnología que mejora la seguridad de los microservicios al permitir conexiones privadas entre su Nube Privada Virtual (VPC) y los servicios compatibles. AWS Ayuda a aislar y proteger el tráfico de microservicios, garantizando que nunca atraviese la Internet pública. Esto es particularmente útil para cumplir con normativas como la PCI o la HIPAA.

Almacén de datos

El almacén de datos se utiliza para conservar los datos que necesitan los microservicios. Los almacenes más populares de datos de sesión son las cachés en memoria, como Memcached o Redis. AWS ofrece ambas tecnologías como parte del ElastiCache servicio gestionado de Amazon.

Colocar una memoria caché entre los servidores de aplicaciones y una base de datos es un mecanismo común para reducir la carga de lectura en la base de datos, lo que, a su vez, puede permitir que los recursos se utilicen para soportar más escrituras. Las cachés también pueden mejorar la latencia.

Las bases de datos relacionales siguen siendo muy populares para almacenar datos estructurados y objetos empresariales. AWS ofrece seis motores de bases de datos (Microsoft SQL Server, Oracle, MySQL, MariaDB, PostgreSQL y Amazon Aurora) como servicios gestionados a través de Amazon Relational Database Service (Amazon RDS).

Sin embargo, las bases de datos relacionales no están diseñadas para una escala infinita, lo que puede dificultar y llevar mucho tiempo aplicar técnicas que admitan un gran número de consultas.

Las bases de datos NoSQL se diseñaron para favorecer la escalabilidad, el rendimiento y la disponibilidad por encima de la coherencia de las bases de datos relacionales. Un elemento importante de las bases de datos NoSQL es que, por lo general, no aplican un esquema estricto. Los datos se distribuyen en particiones que se pueden escalar horizontalmente y se recuperan mediante claves de partición.

Como los microservicios individuales están diseñados para hacer bien una cosa, suelen tener un modelo de datos simplificado que podría adaptarse bien a la persistencia de NoSQL. Es importante entender que las bases de datos NoSQL tienen patrones de acceso diferentes a los de las bases de datos relacionales. Por ejemplo, no es posible unir tablas. Si es necesario, la lógica debe implementarse en la aplicación. Puede usar Amazon DynamoDB para crear una tabla de base de datos que pueda almacenar y recuperar cualquier cantidad de datos y atender cualquier nivel

de tráfico de solicitudes. DynamoDB ofrece un rendimiento de milisegundos de un solo dígito; sin embargo, hay algunos casos de uso que requieren tiempos de respuesta en microsegundos. DynamoDB Accelerator (DAX) proporciona capacidades de almacenamiento en caché para acceder a los datos.

DynamoDB también ofrece una función de escalado automático para ajustar dinámicamente la capacidad de rendimiento en respuesta al tráfico real. Sin embargo, hay casos en los que la planificación de la capacidad resulta difícil o no es posible debido a los grandes picos de actividad de corta duración de la aplicación. Para estas situaciones, DynamoDB ofrece una opción bajo demanda, que ofrece precios sencillos. pay-per-request DynamoDB on-demand es capaz de atender miles de solicitudes por segundo de forma instantánea sin necesidad de planificar la capacidad.

Para obtener más información, consulte Administración de datos distribuidos Cómo elegir una base de datos.

Simplificar las operaciones

Para simplificar aún más los esfuerzos operativos necesarios para ejecutar, mantener y monitorear los microservicios, podemos usar una arquitectura totalmente sin servidores.

Implementación de aplicaciones basadas en Lambda

Puede implementar el código Lambda cargando un `zip` archivo o creando y cargando una imagen de contenedor a través de la interfaz de usuario de la consola con un URI de imagen de Amazon ECR válido. Sin embargo, cuando una función de Lambda se vuelve compleja, lo que significa que tiene capas, dependencias y permisos, la carga a través de la interfaz de usuario puede resultar difícil de manejar para los cambios de código.

El uso de AWS CloudFormation and the AWS Serverless Application Model (AWS SAM) o Terraform agiliza Kit de desarrollo de la nube de AWS (AWS CDK) el proceso de definición de aplicaciones sin servidor. AWS SAM, compatible de forma nativa con CloudFormation, ofrece una sintaxis simplificada para especificar recursos sin servidor.AWS Lambda Las capas ayudan a administrar las bibliotecas compartidas en varias funciones de Lambda, lo que minimiza el espacio de funciones, centraliza las bibliotecas con reconocimiento de inquilinos y mejora la experiencia del desarrollador. Lambda SnapStart para Java mejora el rendimiento de inicio de las aplicaciones sensibles a la latencia.

Para realizar la implementación, especifique las políticas de recursos y permisos en una CloudFormation plantilla, empaquete los artefactos de implementación e implemente la plantilla.

SAM Local, una AWS CLI herramienta, permite el desarrollo, las pruebas y el análisis locales de las aplicaciones sin servidor antes de cargarlas a Lambda.

La integración con herramientas como AWS Cloud9 IDE y AWS CodePipeline agiliza la creación AWS CodeBuild AWS CodeDeploy, las pruebas, la depuración y la implementación de aplicaciones basadas en SAM.

El siguiente diagrama muestra la implementación de AWS Serverless Application Model recursos mediante CloudFormation herramientas de CI/CD. AWS

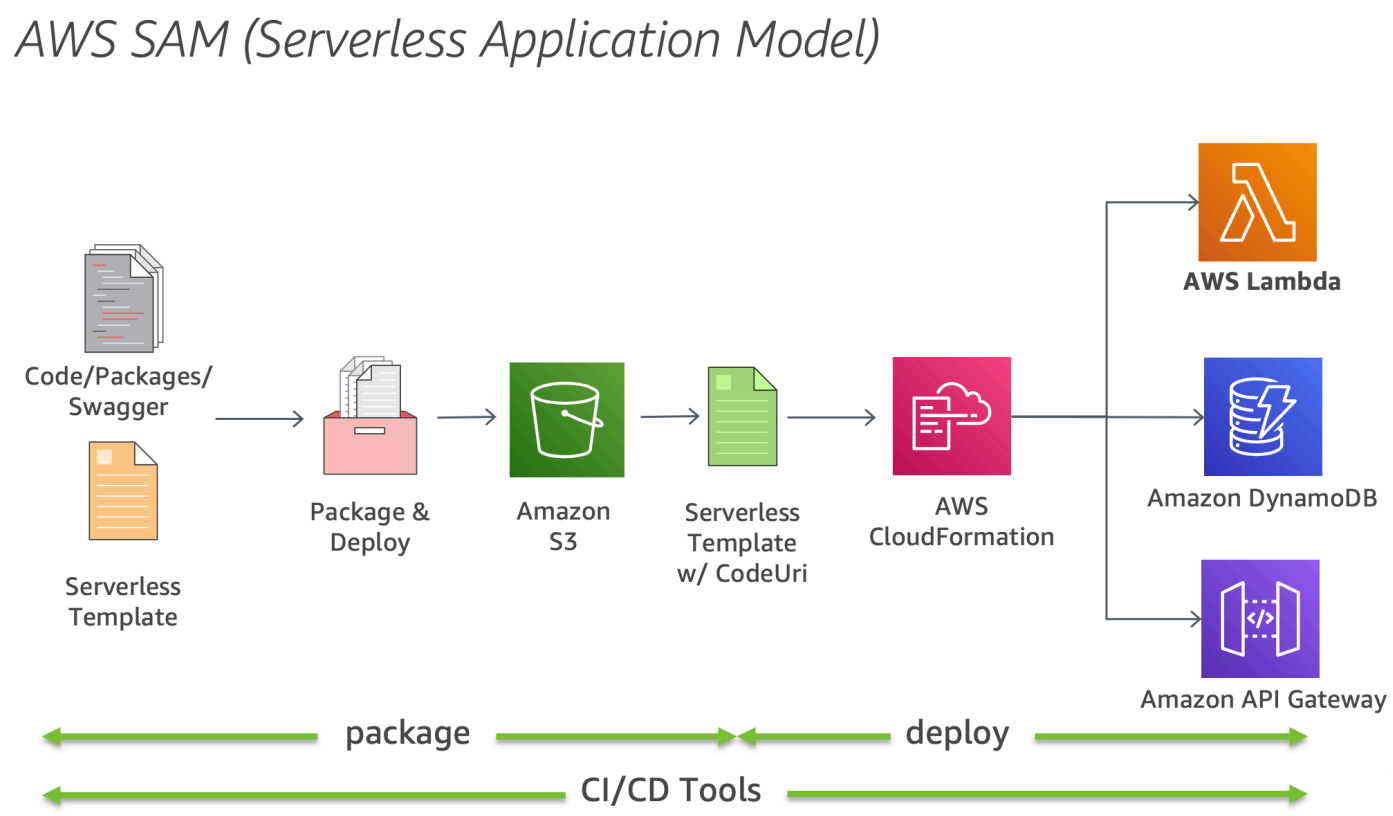

Figura 2: AWS Serverless Application Model (AWS SAM)

Resumiendo las complejidades de la multitenencia

En un entorno multiusuario como las plataformas SaaS, es crucial simplificar las complejidades relacionadas con la multitenencia, lo que permite a los desarrolladores concentrarse en el desarrollo de características y funcionalidades. Esto se puede lograr utilizando herramientas como AWS Lambda Layers, que ofrecen bibliotecas compartidas para abordar problemas transversales. La razón de ser de este enfoque es que las bibliotecas y herramientas compartidas, cuando se utilizan correctamente, gestionan de forma eficiente el contexto de los inquilinos.

Sin embargo, no deberían extenderse a la encapsulación de la lógica empresarial debido a la complejidad y el riesgo que pueden suponer. Un problema fundamental de las bibliotecas compartidas es la creciente complejidad que rodea a las actualizaciones, lo que dificulta su administración en comparación con la duplicación de código estándar. Por lo tanto, es esencial lograr un equilibrio entre el uso de bibliotecas compartidas y la duplicación en la búsqueda de la abstracción más eficaz.

Administración de API

La administración APIs puede llevar mucho tiempo, especialmente si se tienen en cuenta las múltiples versiones, las etapas del ciclo de desarrollo, la autorización y otras funciones, como la limitación y el almacenamiento en caché. Además de API Gateway, algunos clientes también utilizan ALB (Application Load Balancer) o NLB (Network Load Balancer) para la administración de las API. Amazon API Gateway ayuda a reducir la complejidad operativa de la creación y el mantenimiento RESTful APIs. Le permite crear APIs mediante programación, sirve como «puerta de entrada» para acceder a los datos, la lógica empresarial o la funcionalidad desde sus servicios de backend, el control de acceso y autorización, la limitación de velocidad, el almacenamiento en caché, la supervisión y la gestión del tráfico, y funciona sin APIs necesidad de administrar servidores.

La figura 3 ilustra cómo API Gateway gestiona las llamadas a la API e interactúa con otros componentes. Las solicitudes de dispositivos móviles, sitios web u otros servicios de backend se envían al CloudFront punto de presencia (PoP) más cercano para reducir la latencia y ofrecer una experiencia de usuario óptima.

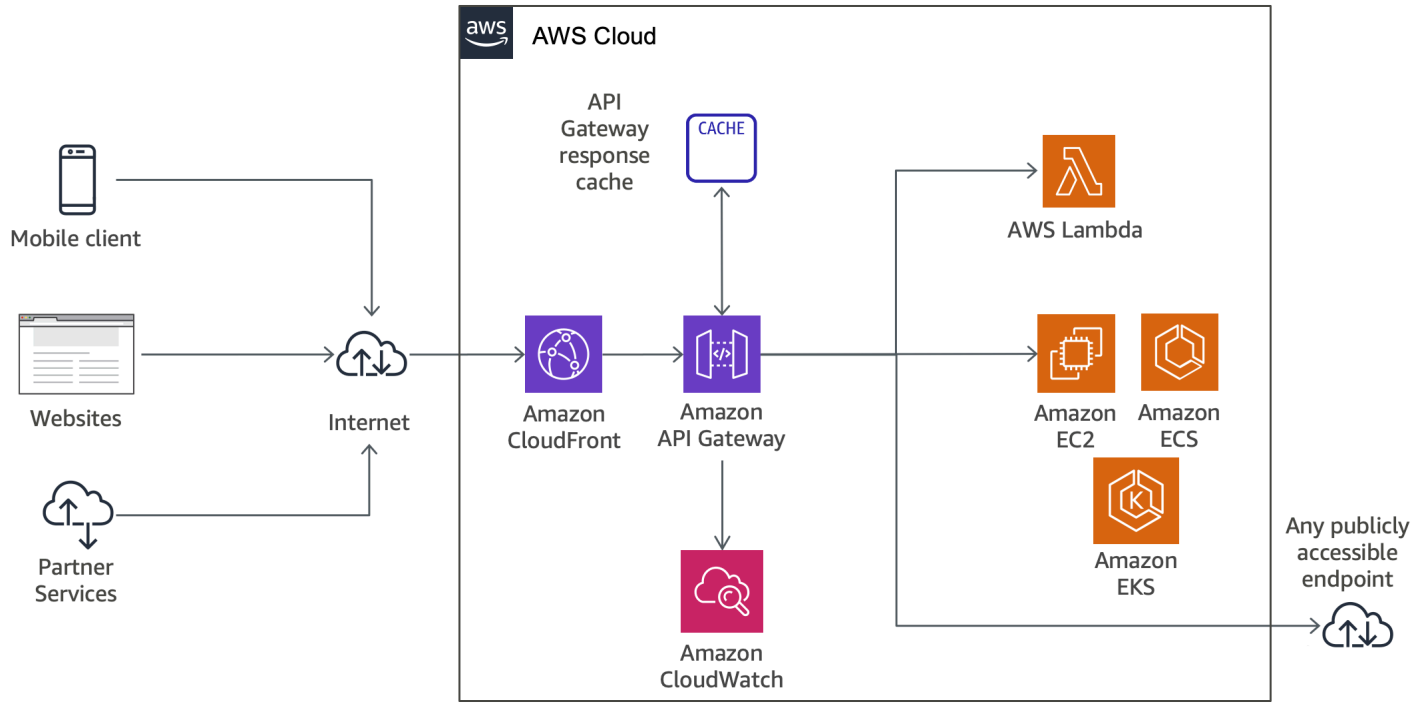

Figura 3: Flujo de llamadas a API Gateway

Microservicios en tecnologías sin servidor

El uso de microservicios con tecnologías sin servidor puede reducir considerablemente la complejidad operativa. AWS Lambda y AWS Fargate, integrado con API Gateway, permite la creación de aplicaciones totalmente sin servidor. A partir del 7 de abril de 2023, las funciones de Lambda pueden transmitir progresivamente las cargas útiles de respuesta al cliente, lo que mejora el rendimiento de las aplicaciones web y móviles. Antes de esto, las aplicaciones basadas en Lambda que utilizaban el modelo tradicional de invocación de solicitud-respuesta tenían que generar y almacenar en búfer la respuesta antes de devolverla al cliente, lo que podía retrasar el tiempo hasta el primer byte. Con la transmisión de respuestas, las funciones pueden enviar respuestas parciales al cliente a medida que estén listas, lo que mejora considerablemente el tiempo transcurrido hasta el primer byte, al que son especialmente sensibles las aplicaciones web y móviles.

En la figura 4 se muestra una arquitectura de microservicios sin servidor que utiliza servicios AWS Lambda gestionados. Esta arquitectura sin servidor mitiga la necesidad de diseñar pensando en la escalabilidad y la alta disponibilidad, y reduce el esfuerzo necesario para ejecutar y monitorear la infraestructura subyacente.

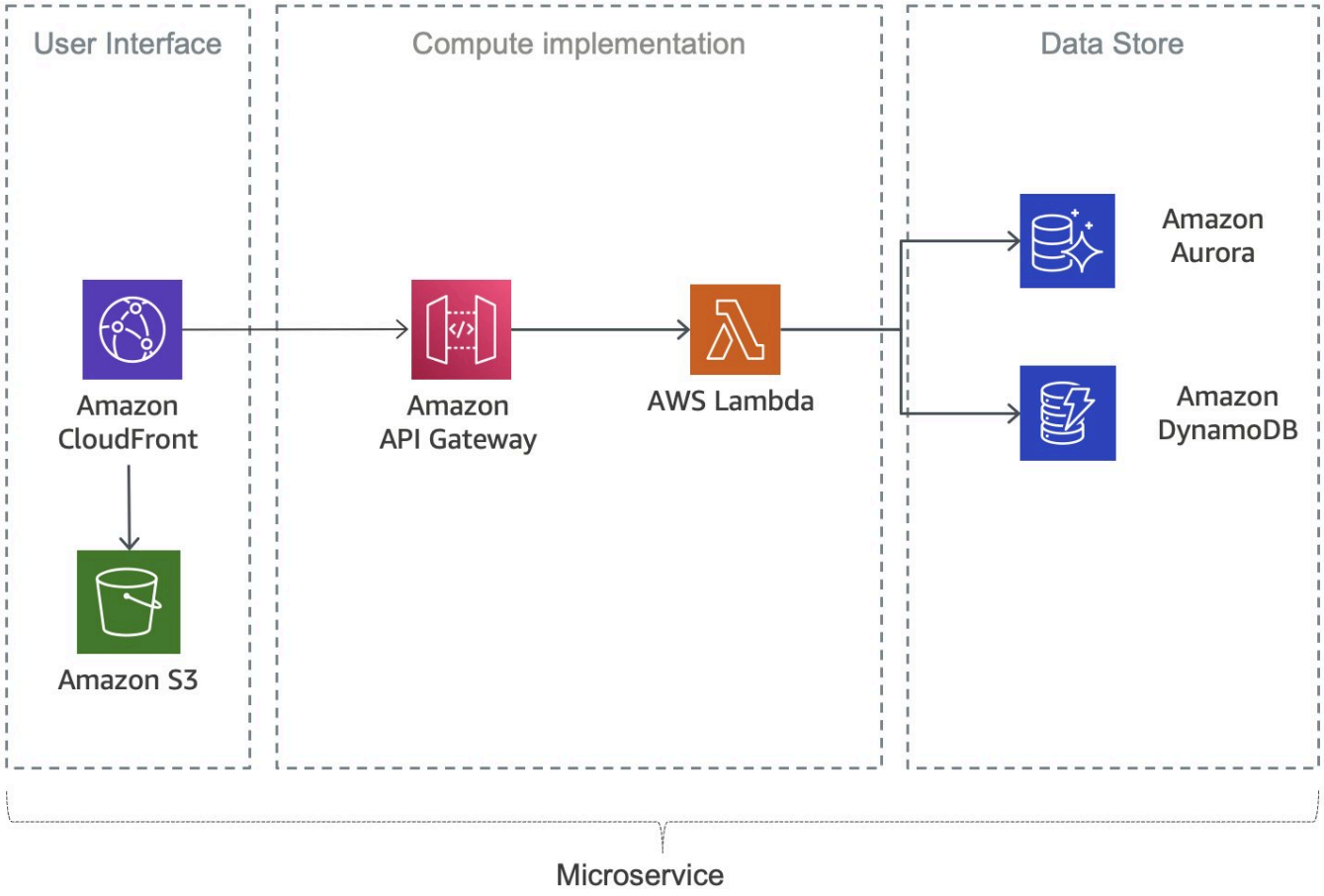

Figura 4: Microservicio sin servidor que utiliza AWS Lambda

La figura 5 muestra una implementación sin servidor similar en la que se utilizan contenedores con AWS Fargate, lo que elimina las preocupaciones sobre la infraestructura subyacente. También incluye Amazon Aurora Serverless, una base de datos bajo demanda y autoscalable que ajusta automáticamente la capacidad en función de los requisitos de la aplicación.

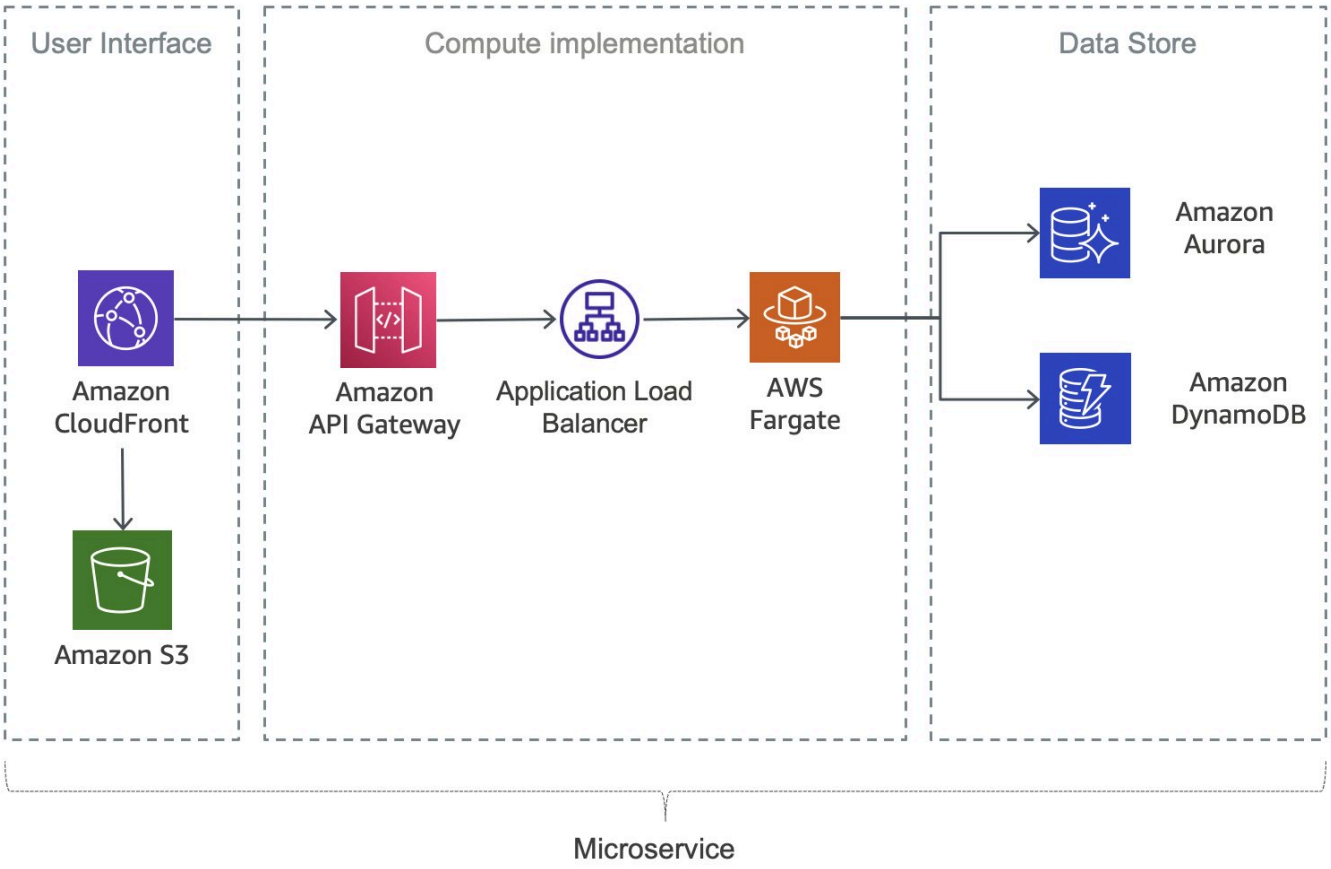

Figura 5: Microservicio sin servidor que utiliza AWS Fargate

Sistemas resilientes y eficientes

Recuperación de desastres (DR)

Las aplicaciones de microservicios suelen seguir los patrones de aplicación de doce factores, en los que los procesos no tienen estado y los datos persistentes se almacenan en servicios de respaldo con estado, como las bases de datos. Esto simplifica la recuperación ante desastres (DR), ya que si un servicio falla, es fácil lanzar nuevas instancias para restaurar la funcionalidad.

Las estrategias de recuperación ante desastres para los microservicios deben centrarse en los servicios secundarios que mantienen el estado de la aplicación, como los sistemas de archivos, las bases de datos o las colas. Las organizaciones deben planificar el objetivo de tiempo de recuperación (RTO) y el objetivo de punto de recuperación (RPO). El RTO es el tiempo máximo aceptable entre la interrupción del servicio y la restauración, mientras que el RPO es el tiempo máximo transcurrido desde el último punto de recuperación de datos.

Para obtener más información sobre las estrategias de recuperación ante desastres, consulte el documento técnico sobre recuperación ante desastres de cargas de trabajo en AWS: Recuperación en la nube.

Alta disponibilidad (HA)

Examinaremos la alta disponibilidad (HA) de varios componentes de una arquitectura de microservicios.

Amazon EKS proporciona alta disponibilidad mediante la ejecución de instancias de plano de datos y control de Kubernetes en varias zonas de disponibilidad. Detecta y reemplaza automáticamente las instancias del plano de control en mal estado y proporciona actualizaciones de versiones y parches automatizados.

Amazon ECR utiliza Amazon Simple Storage Service (Amazon S3) como almacenamiento para que las imágenes de sus contenedores estén altamente disponibles y sean accesibles. Funciona con Amazon EKS, Amazon ECS y AWS Lambda simplifica el flujo de trabajo del desarrollo a la producción.

Amazon ECS es un servicio regional que simplifica la ejecución de contenedores de una manera altamente disponible en varias zonas de disponibilidad dentro de una región, y ofrece múltiples

estrategias de programación que colocan los contenedores en función de las necesidades de recursos y los requisitos de disponibilidad.

AWS Lambda opera en varias zonas de disponibilidad, lo que garantiza la disponibilidad durante las interrupciones del servicio en una sola zona. Si conecta su función a una VPC, especifique las subredes en varias zonas de disponibilidad para obtener una alta disponibilidad.

Componentes de sistemas distribuidos

En una arquitectura de microservicios, el descubrimiento de servicios se refiere al proceso de localizar e identificar dinámicamente las ubicaciones de red (direcciones IP y puertos) de los microservicios individuales dentro de un sistema distribuido.

Al elegir un enfoque AWS, tenga en cuenta factores como:

- Modificación del código: ¿Puede obtener los beneficios sin modificar el código?
- Tráfico entre VPC o entre cuentas: si es necesario, ¿su sistema necesita una gestión eficiente de la comunicación entre diferentes o? VPCs Cuentas de AWS
- Estrategias de implementación: ¿Su sistema usa o planea usar estrategias de implementación avanzadas, como las implementaciones «azul-verde» o «Canary»?
- Consideraciones sobre el rendimiento: si su arquitectura se comunica con frecuencia con servicios externos, ¿cuál será el impacto en el rendimiento general?

AWS ofrece varios métodos para implementar la detección de servicios en su arquitectura de microservicios:

- Amazon ECS Service Discovery: Amazon ECS admite la detección de servicios mediante su método basado en DNS o mediante la integración con AWS Cloud Map (consulte ECS Service Discovery). ECS Service Connect mejora aún más la administración de la conexión, lo que puede resultar especialmente beneficioso para aplicaciones más grandes con varios servicios que interactúan.
- Amazon Route 53: Route 53 se integra con ECS y otros AWS servicios, como EKS, para facilitar la detección de servicios. En el contexto de ECS, Route 53 puede usar la función ECS Service Discovery, que aprovecha la API de nombres automáticos para registrar y anular el registro de los servicios automáticamente.
- AWS Cloud Map: Esta opción ofrece una detección dinámica de servicios basada en una API, que propaga los cambios entre los servicios.

Para necesidades de comunicación más avanzadas, Amazon VPC Lattice es un servicio de redes de aplicaciones que conecta, monitorea y protege de manera uniforme las comunicaciones entre sus servicios, lo que ayuda a mejorar la productividad para que sus desarrolladores puedan centrarse en crear funciones que sean importantes para su empresa. Puede definir políticas de administración,

acceso y monitoreo del tráfico de red para conectar los servicios de cómputo de manera simplificada y uniforme en todas las instancias, contenedores y aplicaciones sin servidor.

Si ya utilizas software de terceros, como HashiCorp Consul o Netflix Eureka para la detección de servicios, tal vez prefieras seguir utilizándolos a medida que vayas migrando a ellos AWS, lo que permitirá una transición más fluida.

La elección entre estas opciones debe ajustarse a tus necesidades específicas. Para requisitos más simples, soluciones basadas en DNS como Amazon ECS o AWS Cloud Map podrían ser suficientes. Para sistemas más complejos o más grandes, las mallas de servicio como Amazon VPC Lattice podrían ser más adecuadas.

En conclusión, el diseño de una arquitectura de microservicios AWS consiste en seleccionar las herramientas adecuadas para satisfacer sus necesidades específicas. Si tiene en cuenta las consideraciones expuestas, puede asegurarse de que está tomando decisiones informadas para optimizar la detección de servicios y la comunicación entre servicios de su sistema.

Administración de datos distribuidos

En las aplicaciones tradicionales, todos los componentes suelen compartir una única base de datos. Por el contrario, cada componente de una aplicación basada en microservicios mantiene sus propios datos, lo que promueve la independencia y la descentralización. Este enfoque, conocido como gestión de datos distribuidos, presenta nuevos desafíos.

Uno de estos desafíos surge de la disyuntiva entre consistencia y rendimiento en los sistemas distribuidos. Suele ser más práctico aceptar pequeños retrasos en las actualizaciones de los datos (coherencia eventual) que insistir en las actualizaciones instantáneas (coherencia inmediata).

A veces, las operaciones empresariales requieren varios microservicios para funcionar juntos. Si una parte falla, es posible que tengas que deshacer algunas tareas completadas. El patrón Saga ayuda a gestionar esta situación mediante la coordinación de una serie de acciones compensatorias.

Para ayudar a que los microservicios se mantengan sincronizados, se puede utilizar un almacén de datos centralizado. Esta tienda, gestionada con herramientas como AWS Lambda Amazon EventBridge, puede ayudar a limpiar y deduplicar datos. AWS Step Functions

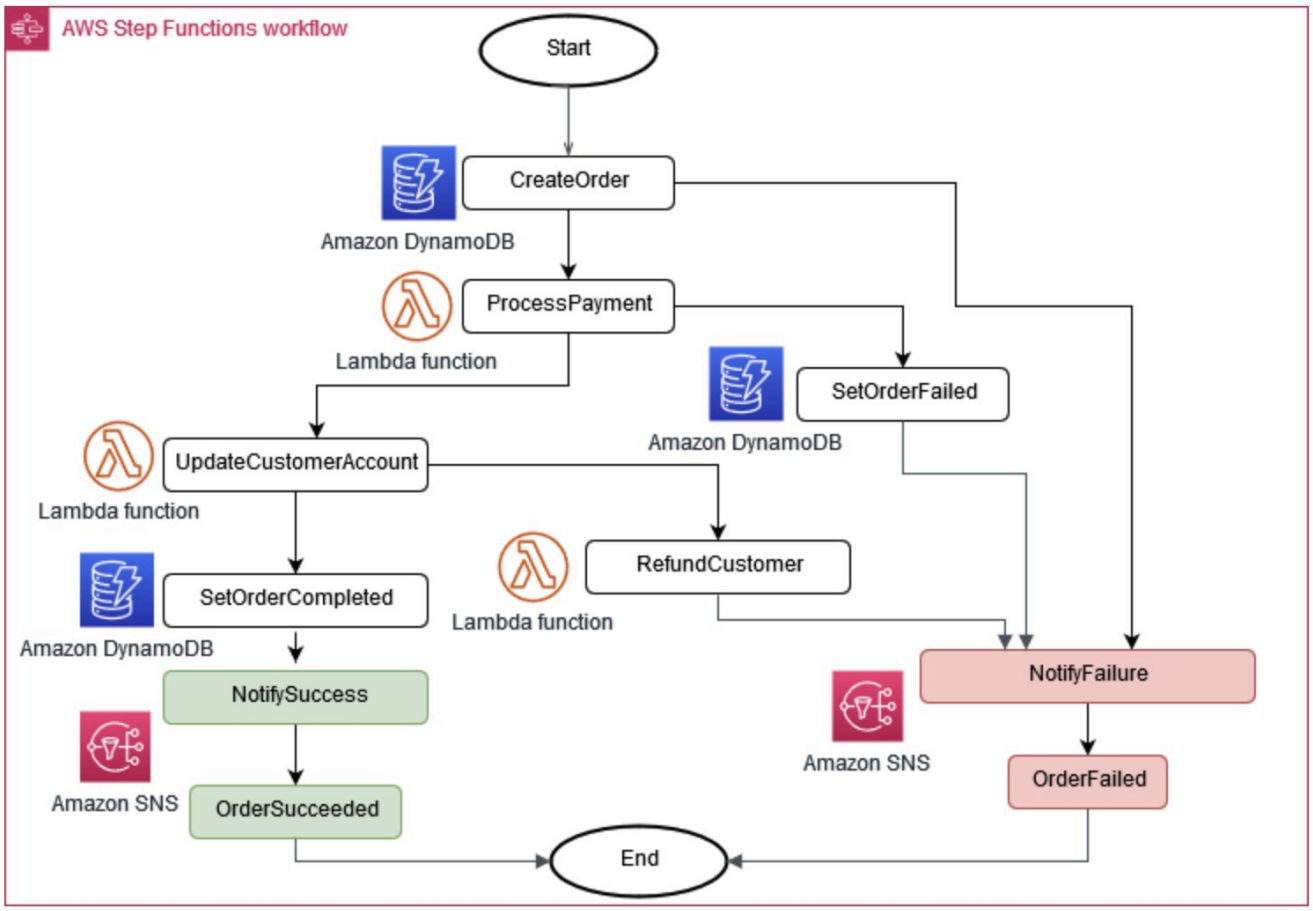

Figura 6: Coordinador de ejecución de Saga

Un enfoque común para gestionar los cambios en los microservicios es el abastecimiento de eventos. Cada cambio en la aplicación se registra como un evento, lo que crea una cronología del estado del sistema. Este enfoque no solo ayuda a depurar y auditar, sino que también permite que diferentes partes de una aplicación reaccionen ante los mismos eventos.

El abastecimiento de eventos suele funcionar hand-in-hand con el patrón de segregación de responsabilidades por consultas de comandos (CQRS), que separa la modificación de datos de la consulta de datos en distintos módulos para mejorar el rendimiento y la seguridad.

También AWS puede implementar estos patrones mediante una combinación de servicios. Como puede ver en la figura 7, Amazon Kinesis Data Streams puede funcionar como almacén central de eventos, mientras que Amazon S3 proporciona un almacenamiento duradero para todos los registros de eventos. AWS Lambda, Amazon DynamoDB y Amazon API Gateway trabajan juntos para gestionar y procesar estos eventos.

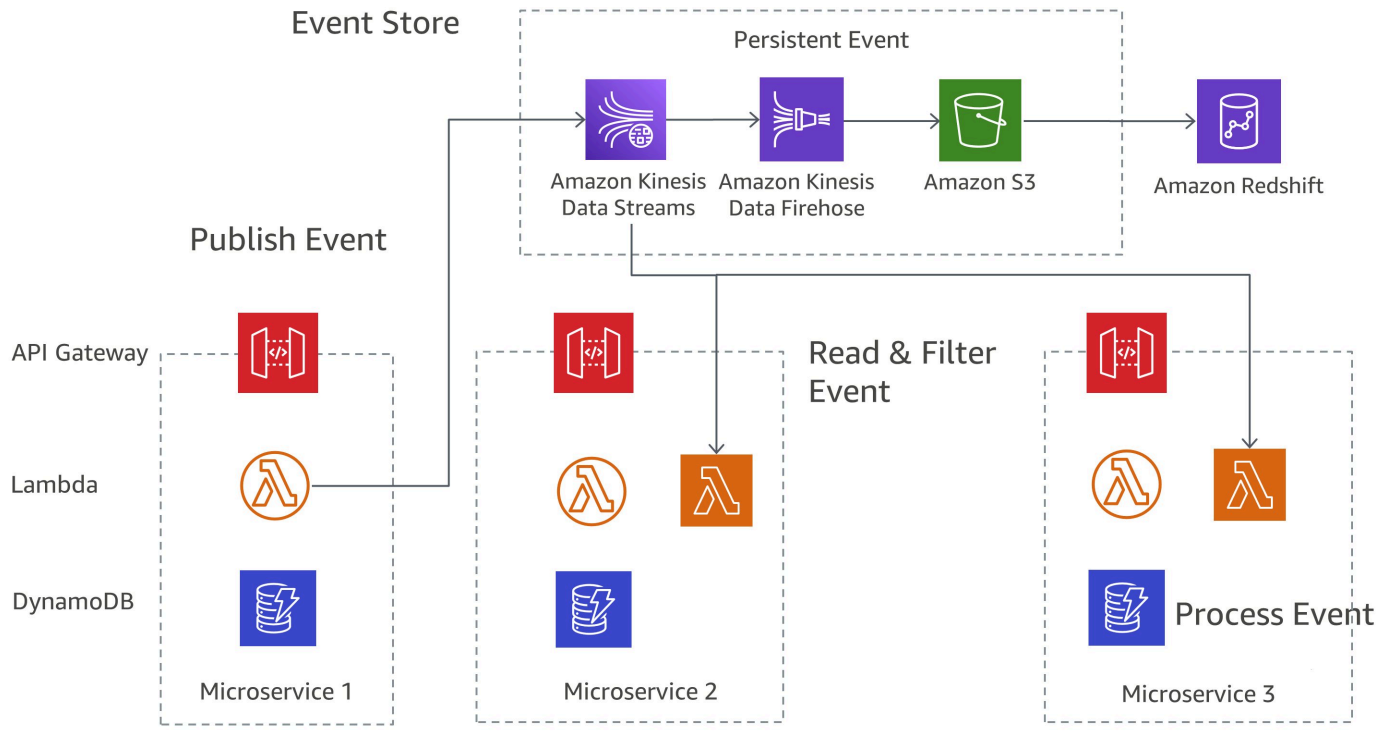

Figura 7: El patrón de aprovisionamiento de eventos está activado AWS

Recuerde que, en los sistemas distribuidos, es posible que los eventos se entreguen varias veces debido a los reintentos, por lo que es importante diseñar las aplicaciones para gestionarlos.

Administración de la configuración

En una arquitectura de microservicios, cada servicio interactúa con varios recursos, como bases de datos, colas y otros servicios. Es fundamental disponer de una forma coherente de configurar las conexiones y el entorno operativo de cada servicio. Lo ideal es que una aplicación se adapte a las nuevas configuraciones sin necesidad de reiniciarla. Este enfoque forma parte de los principios de las aplicaciones de doce factores, que recomiendan almacenar las configuraciones en variables de entorno.

Un enfoque diferente es usar AWS App Config. Es una función de AWS Systems Manager que permite a los clientes configurar, validar e implementar de forma rápida y segura los indicadores de características y la configuración de las aplicaciones. El indicador de características y los datos de configuración se pueden validar sintáctica o semánticamente en la fase previa a la implementación, y se pueden monitorear y revertir automáticamente si se activa una alarma que haya configurado. AppConfig se puede integrar con Amazon ECS y Amazon EKS mediante el AWS AppConfig agente. El agente funciona como un contenedor asociado que se ejecuta junto con las aplicaciones de contenedores de Amazon ECS y Amazon EKS. Si utiliza indicadores de AWS AppConfig características u otros datos de configuración dinámica en una función de Lambda, le recomendamos que añada la extensión de AWS AppConfig Lambda como una capa a la función de Lambda.

GitOpses un enfoque innovador para la administración de la configuración que utiliza Git como fuente de información para todos los cambios de configuración. Esto significa que cualquier cambio realizado en los archivos de configuración se rastrea, versiona y audita automáticamente a través de Git.

Administración de secretos

La seguridad es fundamental, por lo que las credenciales no deben pasarse en texto plano. AWS ofrece servicios seguros para ello, como AWS Systems Manager Parameter Store y AWS Secrets Manager. Estas herramientas pueden enviar secretos a los contenedores de Amazon EKS como volúmenes o a Amazon ECS como variables de entorno. En AWS Lambda, las variables de entorno se ponen a disposición del código automáticamente. Para los flujos de trabajo de Kubernetes, el operador de secretos externos obtiene los secretos directamente de los servicios, por ejemplo AWS Secrets Manager, creando los secretos de Kubernetes correspondientes. Esto permite una integración perfecta con las configuraciones nativas de Kubernetes.

Optimización de costes y sostenibilidad

La arquitectura de microservicios puede mejorar la optimización de costes y la sostenibilidad. Al dividir una aplicación en partes más pequeñas, puede ampliar solo los servicios que necesitan más recursos, lo que reduce los costos y el desperdicio. Esto resulta especialmente útil cuando se trata de tráfico variable. Los microservicios se desarrollan de forma independiente. De este modo, los desarrolladores pueden realizar actualizaciones más pequeñas y reducir los recursos que se gastan en las pruebas integrales. Durante la actualización, tendrán que probar solo un subconjunto de las funciones, a diferencia de los monolitos.

Los componentes sin estado (servicios que almacenan el estado en un almacén de datos externo en lugar de en un almacén de datos local) de su arquitectura pueden utilizar Amazon EC2 Spot Instances, que ofrecen EC2 capacidad no utilizada en la AWS nube. Estas instancias son más rentables que las instancias bajo demanda y son perfectas para cargas de trabajo que pueden gestionar las interrupciones. Esto puede reducir aún más los costos y, al mismo tiempo, mantener una alta disponibilidad.

Con los servicios aislados, puede usar opciones de procesamiento con costos optimizados para cada grupo de autoscalamiento. Por ejemplo, AWS Graviton ofrece opciones informáticas rentables y de alto rendimiento para cargas de trabajo que se adaptan a las instancias basadas en ARM.

La optimización de los costos y el uso de los recursos también ayuda a minimizar el impacto ambiental, alineándose con el pilar de sustentabilidad del Well-Architected Framework. Puede supervisar su progreso en la reducción de las emisiones de carbono mediante la herramienta de huella de carbono del AWS cliente. Esta herramienta proporciona información sobre el impacto ambiental de su AWS uso.

Mecanismos de comunicación

En el paradigma de los microservicios, varios componentes de una aplicación deben comunicarse a través de una red. Los enfoques comunes para esto incluyen la mensajería asíncrona, basada en REST, GraphQL y GRPC.

Comunicación basada en REST

El HTTP/S protocolo, que se usa ampliamente para la comunicación sincrónica entre microservicios, suele funcionar de forma automática. RESTful APIs API Gateway ofrece una forma simplificada de crear una API que sirva como punto de acceso centralizado a los servicios de backend y gestione tareas como la gestión del tráfico, la autorización, la supervisión y el control de versiones.

Comunicación basada en GraphQL

Del mismo modo, GraphQL es un método generalizado para la comunicación sincrónica, que utiliza los mismos protocolos que REST pero limita la exposición a un único punto final. Con AWS AppSync, puede crear y publicar aplicaciones GraphQL que interactúan directamente con AWS los servicios y almacenes de datos, o incorporar funciones de Lambda para la lógica empresarial.

Comunicación basada en gRPC

gRPC es un protocolo de comunicación RPC síncrono, ligero, de alto rendimiento y de código abierto. gRPC mejora sus protocolos subyacentes mediante el uso de HTTP/2 y habilitando más funciones, como la compresión y la priorización de flujos. Utiliza el lenguaje de definición de interfaces (IDL) de Protobuf, que está codificado en binario y, por lo tanto, aprovecha el marco binario HTTP/2.

Mensajería asíncrona y transmisión de eventos

La mensajería asíncrona permite que los servicios se comuniquen enviando y recibiendo mensajes a través de una cola. Esto permite que los servicios permanezcan acoplados de forma flexible y promueva el descubrimiento de servicios.

La mensajería se puede definir de los tres tipos siguientes:

- Colas de mensajes: una cola de mensajes actúa como un búfer que separa a los remitentes (productores) de los receptores (consumidores) de los mensajes. Los productores colocan los mensajes en cola y los consumidores los dejan en cola y los procesan. Este patrón es útil para la comunicación asíncrona, la nivelación de carga y la gestión de ráfagas de tráfico.
- Publicar-suscribirse: en el patrón de publicación-suscripción, se publica un mensaje en un tema y varios suscriptores interesados reciben el mensaje. Este patrón permite transmitir eventos o mensajes a varios consumidores de forma asíncrona.
- Mensajería basada en eventos: la mensajería basada en eventos implica capturar los eventos que ocurren en el sistema y reaccionar ante ellos. Los eventos se publican en un intermediario de mensajes y los servicios interesados se suscriben a tipos de eventos específicos. Este patrón permite un acoplamiento flexible y permite a los servicios reaccionar ante los eventos sin dependencias directas.

Para implementar cada uno de estos tipos de mensajes, AWS ofrece varios servicios gestionados como Amazon SQS, Amazon SNS, Amazon, EventBridge Amazon MQ y Amazon MSK. Estos servicios tienen características únicas adaptadas a necesidades específicas:

- Amazon Simple Queue Service (Amazon SQS) y Amazon Simple Notification Service (Amazon SNS): Como puede ver en la figura 8, estos dos servicios se complementan entre sí: Amazon SQS proporciona un espacio para almacenar mensajes y Amazon SNS permite la entrega de mensajes a varios suscriptores. Son eficaces cuando es necesario entregar el mismo mensaje a varios destinos.

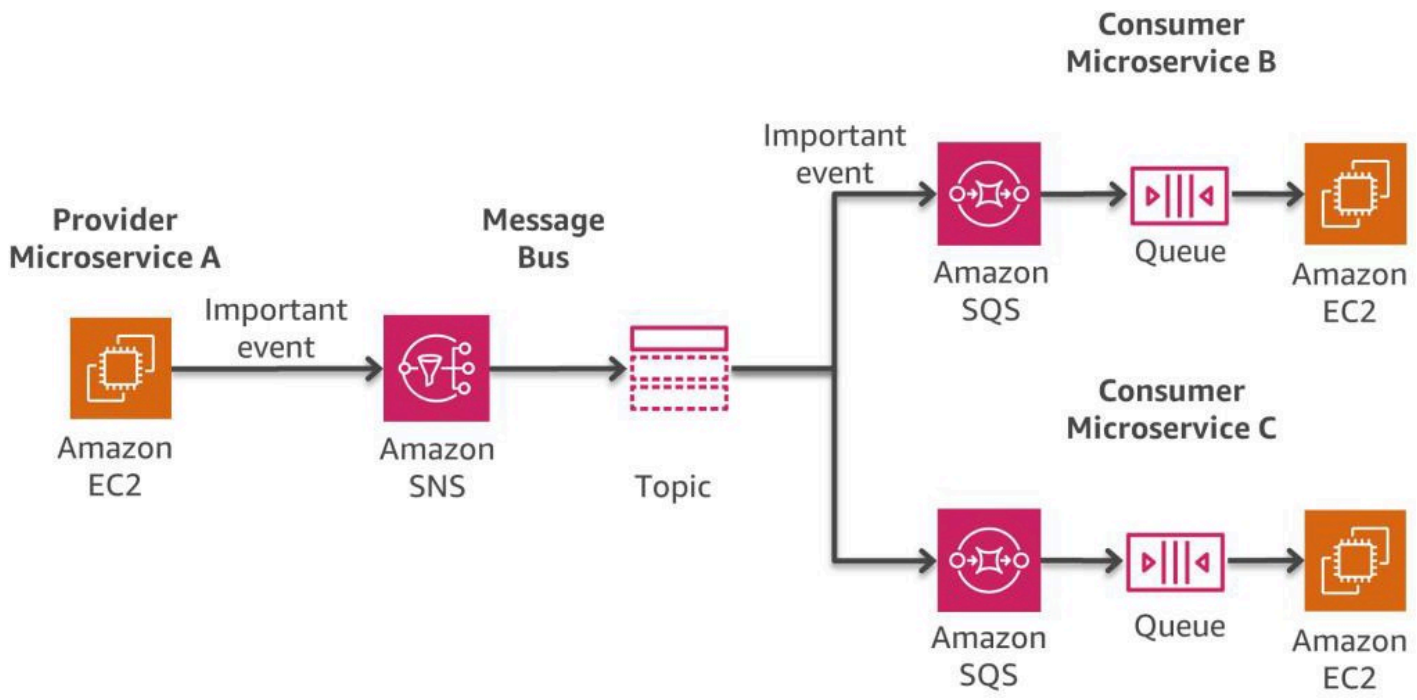

Figura 8: El patrón del bus de mensajes está activado AWS

- Amazon EventBridge: un servicio sin servidor que utiliza eventos para conectar los componentes de la aplicación entre sí, lo que le facilita la creación de aplicaciones escalables basadas en eventos. Úselo para enrutar eventos desde fuentes como aplicaciones propias, AWS servicios y software de terceros a aplicaciones de consumo en toda su organización. EventBridge proporciona una forma sencilla y coherente de incorporar, filtrar, transformar y distribuir eventos para que pueda crear nuevas aplicaciones rápidamente. EventBridge Los buses de eventos son ideales para many-to-many enrutar eventos entre servicios basados en eventos.
- Amazon MQ: una buena opción si tienes un sistema de mensajería preexistente que utiliza protocolos estándar como JMS, AMQP o similares. Este servicio gestionado reemplaza su sistema sin interrumpir las operaciones.
- Amazon MSK (Managed Kafka): un sistema de mensajería para almacenar y leer mensajes, útil para los casos en los que los mensajes deben procesarse varias veces. También admite la transmisión de mensajes en tiempo real.
- Amazon Kinesis: procesamiento y análisis en tiempo real de datos de streaming. Esto permite el desarrollo de aplicaciones en tiempo real y proporciona una integración perfecta con el AWS ecosistema.

Recuerde que el mejor servicio para usted depende de sus necesidades específicas, por lo que es importante entender lo que ofrece cada uno y cómo se alinea con sus requisitos.

Organización y administración del estado

La orquestación de microservicios se refiere a un enfoque centralizado, en el que un componente central, conocido como orquestador, es responsable de gestionar y coordinar las interacciones entre los microservicios. Organizar los flujos de trabajo en varios microservicios puede resultar difícil. No se recomienda incrustar el código de orquestación directamente en los servicios, ya que introduce un acoplamiento más estrecho y dificulta la sustitución de los servicios individuales.

Step Functions proporciona un motor de flujo de trabajo para gestionar las complejidades de la orquestación de servicios, como la gestión de errores y la serialización. Esto le permite escalar y cambiar las aplicaciones rápidamente sin añadir código de coordinación. Step Functions forma parte de la plataforma AWS sin servidor y es compatible con las funciones de Lambda, Amazon EC2, Amazon EKS, Amazon ECS SageMaker , AI AWS Glue y más.

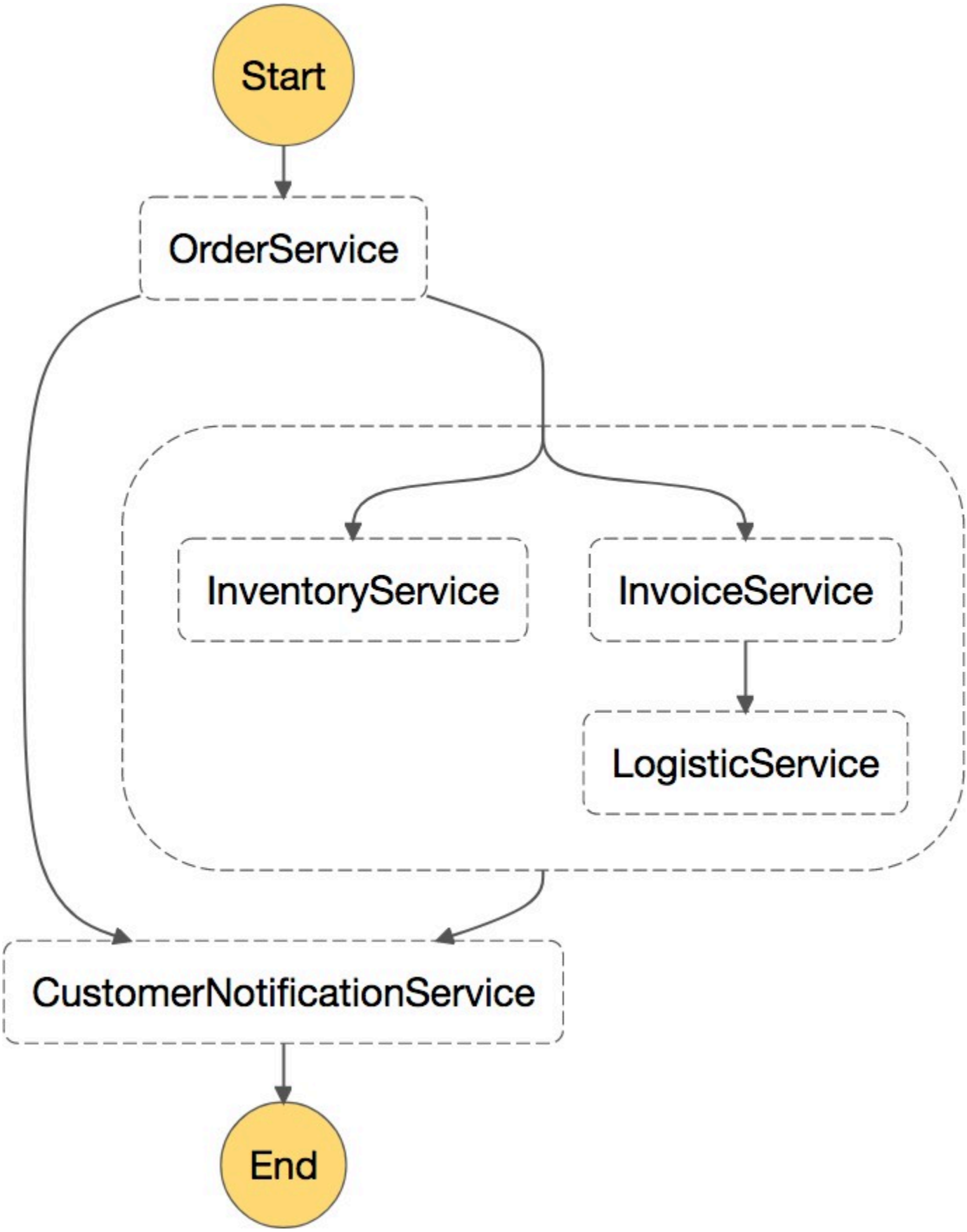
Start
OrderService
InventoryService
InvoiceService
LogisticService
CustomerNotificationService
End

Figura 9: Un ejemplo de un flujo de trabajo de microservicios con pasos paralelos y secuenciales invocados por AWS Step Functions

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) es una alternativa a Step Functions. Debe utilizar Amazon MWAA si prioriza el código abierto y la portabilidad. Airflow cuenta con una comunidad de código abierto amplia y activa que aporta nuevas funcionalidades e integraciones de forma periódica.

Observabilidad

Dado que las arquitecturas de microservicios se componen intrínsecamente de muchos componentes distribuidos, la observabilidad de todos esos componentes se vuelve fundamental. Amazon lo CloudWatch permite, recopilando y rastreando métricas, monitoreando los archivos de registro y reaccionando a los cambios en su AWS entorno. Puede monitorear AWS los recursos y las métricas personalizadas generadas por sus aplicaciones y servicios.

Temas

- Supervisión
- Centralización de registros
- Rastreo distribuido
- Análisis de registros en AWS
- Otras opciones de análisis

Supervisión

CloudWatch ofrece visibilidad en todo el sistema sobre la utilización de los recursos, el rendimiento de las aplicaciones y el estado operativo. En una arquitectura de microservicios, la supervisión personalizada de las métricas CloudWatch es beneficiosa, ya que los desarrolladores pueden elegir qué métricas recopilar. El escalado dinámico también se puede basar en estas métricas personalizadas.

CloudWatch Container Insights amplía esta funcionalidad y recopila automáticamente métricas para muchos recursos, como la CPU, la memoria, el disco y la red. Ayuda a diagnosticar los problemas relacionados con los contenedores y agiliza la resolución.

Para Amazon EKS, la opción preferida con frecuencia es Prometheus, una plataforma de código abierto que ofrece capacidades integrales de monitoreo y alertas. Por lo general, se combina con Grafana para una visualización intuitiva de las métricas. Amazon Managed Service for Prometheus (AMP) ofrece un servicio de monitorización totalmente compatible con Prometheus, que te permite supervisar las aplicaciones en contenedores sin esfuerzo. Además, Amazon Managed Grafana (AMG) simplifica el análisis y la visualización de las métricas, lo que elimina la necesidad de gestionar la infraestructura subyacente.

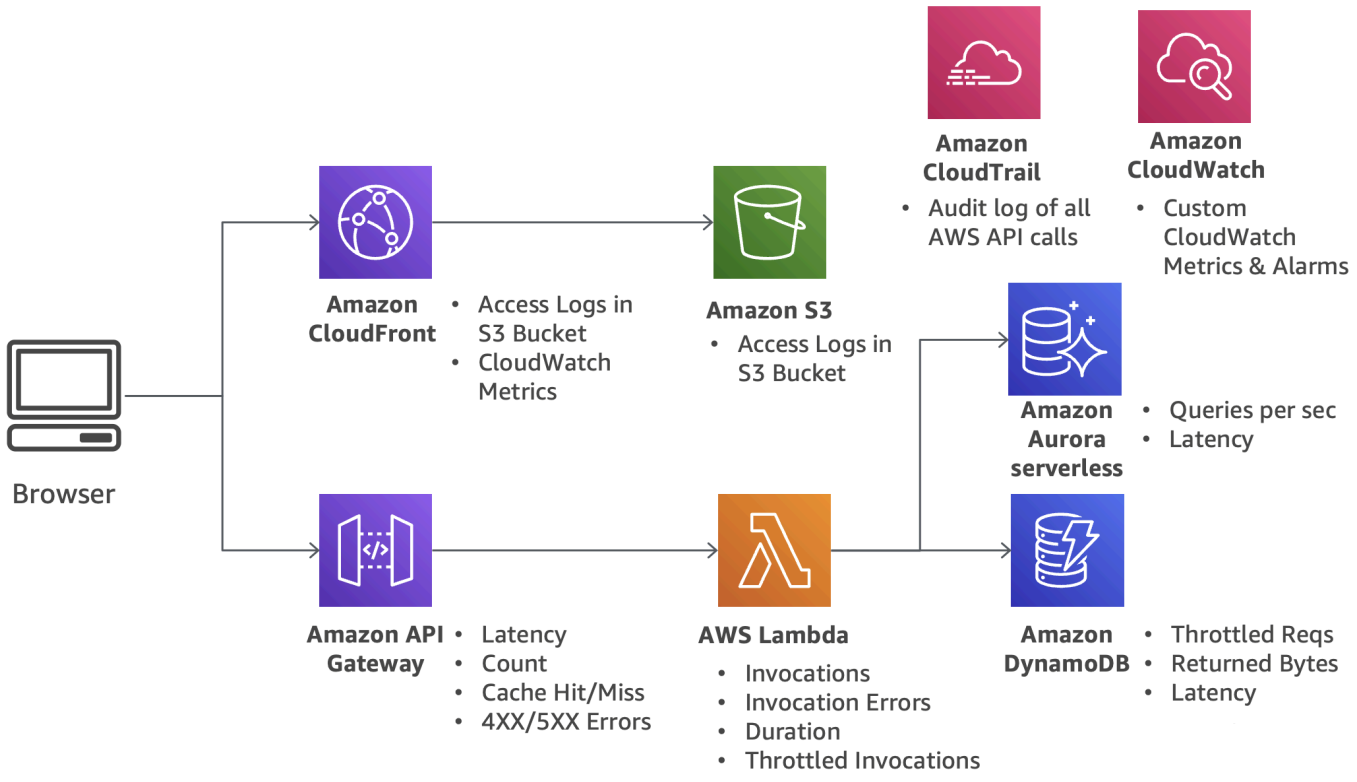

Figura 10: Una arquitectura sin servidor con componentes de monitoreo

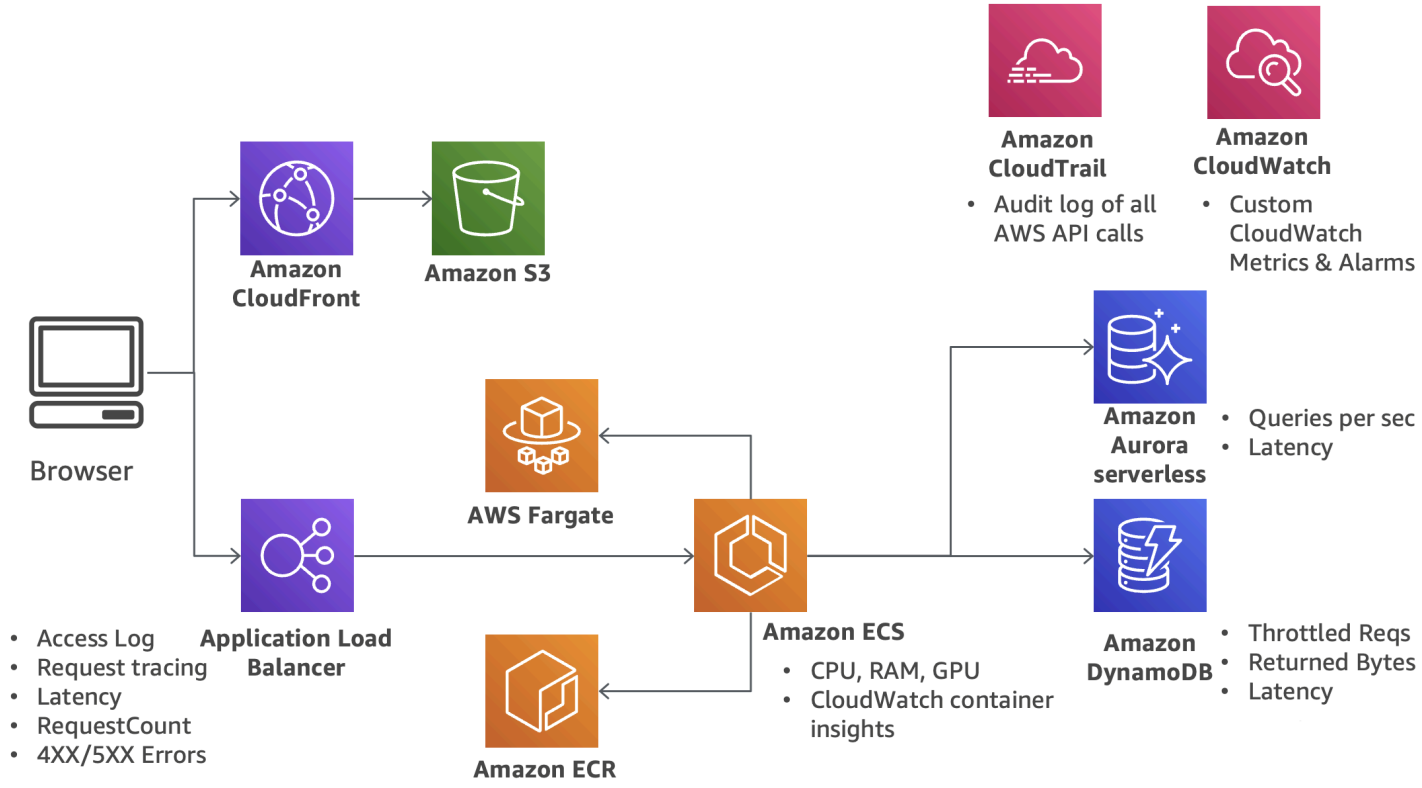

Figura 11: Una arquitectura basada en contenedores con componentes de monitoreo

Centralización de registros

El registro es clave para identificar y resolver problemas. Con los microservicios, puedes lanzar con más frecuencia y experimentar con nuevas funciones. AWS proporciona servicios como Amazon S3, CloudWatch Logs y Amazon OpenSearch Service para centralizar los archivos de registro. Amazon EC2 utiliza un daemon para enviar los registros, CloudWatch mientras que Lambda y Amazon ECS envían allí sus resultados de registro de forma nativa. En Amazon EKS, se pueden usar Fluent Bit o Fluentd para reenviar los registros a CloudWatch los informes mediante Kibana OpenSearch . Sin embargo, debido a su menor tamaño y a las ventajas de rendimiento, se recomienda Fluent Bit en lugar de Fluentd.

La figura 12 ilustra cómo se dirigen los registros de varios AWS servicios a Amazon S3 y CloudWatch. Estos registros centralizados se pueden analizar más a fondo mediante Amazon OpenSearch Service, incluido Kibana, para la visualización de datos. Además, Amazon Athena se puede utilizar para consultas ad hoc en los registros almacenados en Amazon S3.

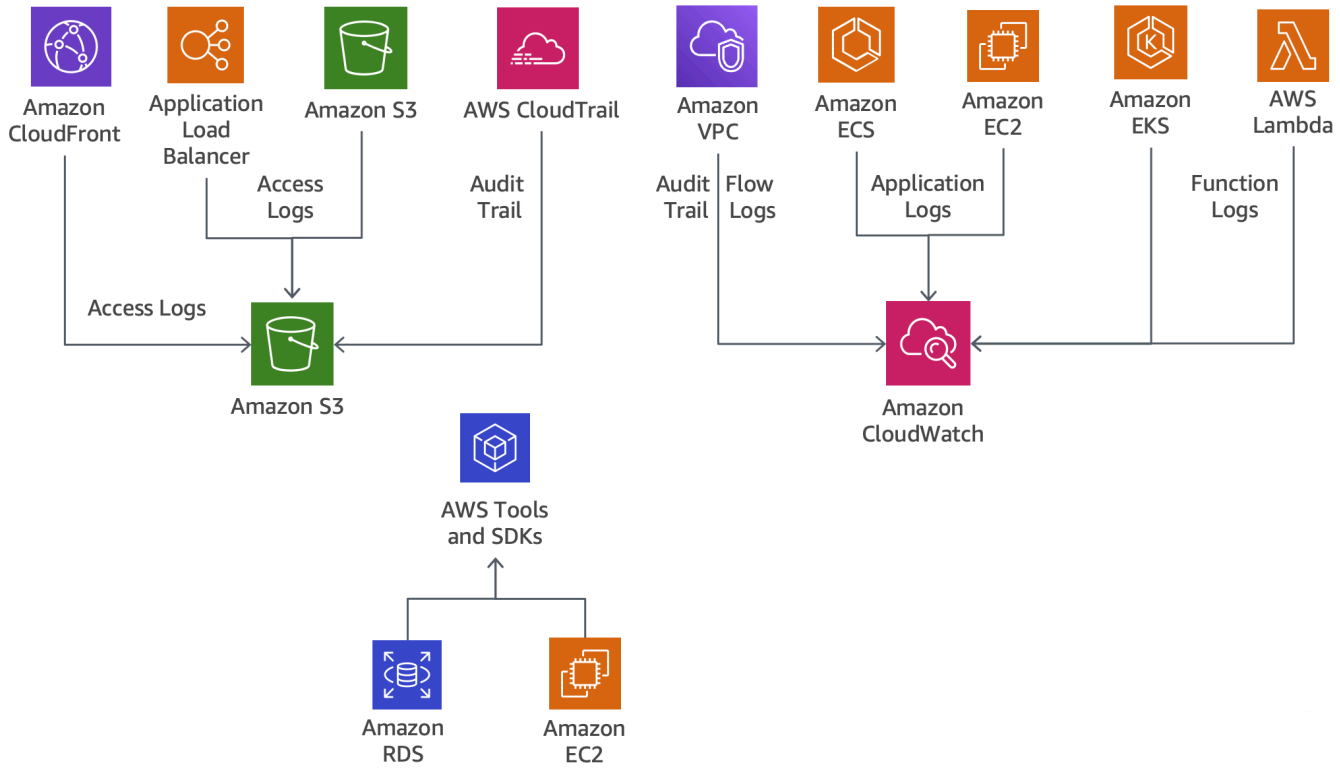

Figura 12: Capacidad de registro de los AWS servicios

Rastreo distribuido

Los microservicios suelen trabajar juntos para gestionar las solicitudes. AWS X-Ray utiliza los identificadores de correlación para realizar un seguimiento de las solicitudes en estos servicios. X-Ray funciona con Amazon EC2, Amazon ECS, Lambda y Elastic Beanstalk.

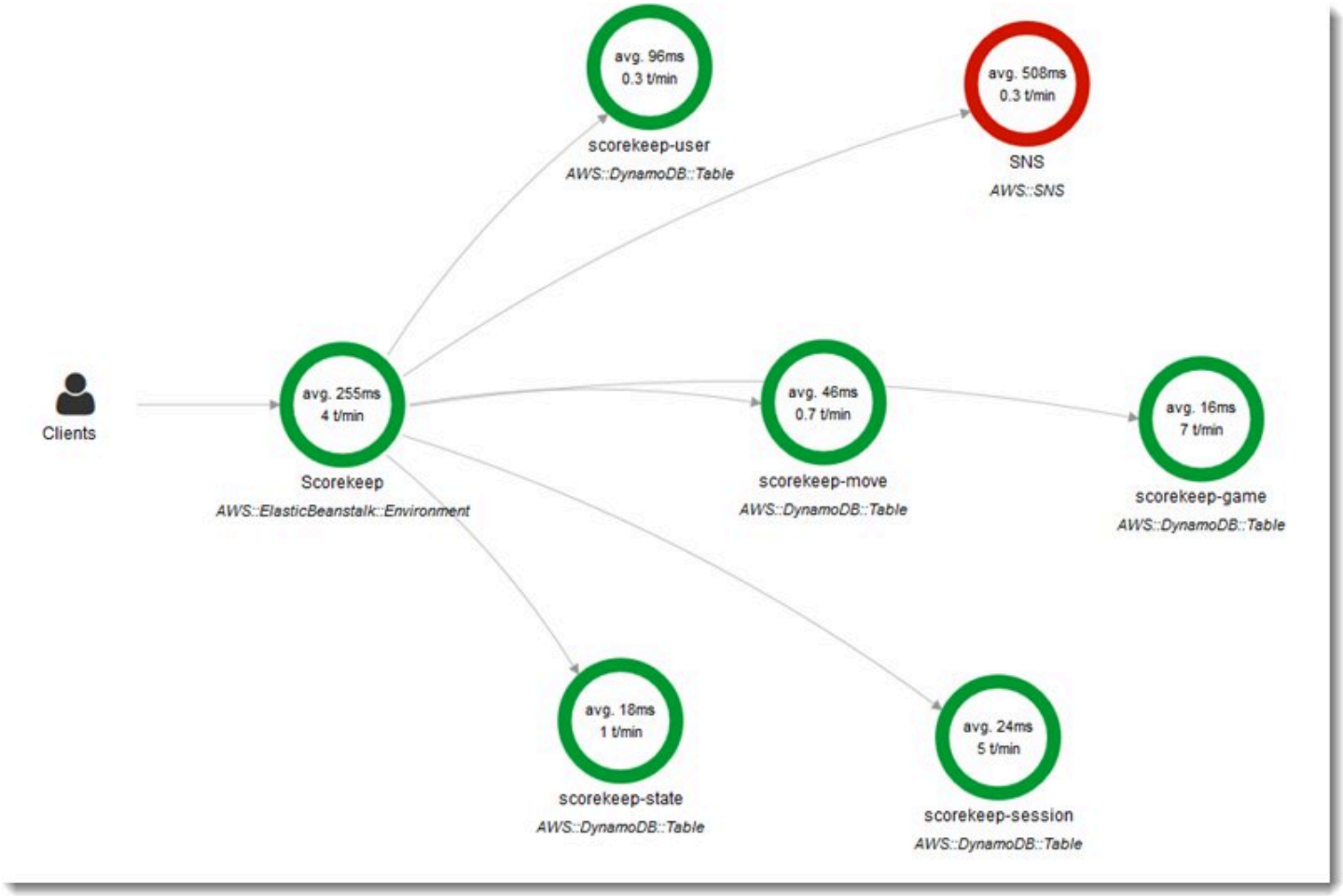

Figura 13: mapa AWS X-Ray de servicios

AWS Distro for OpenTelemetry forma parte del OpenTelemetry proyecto y proporciona agentes y de código abierto para recopilar rastreos APIs y métricas distribuidos, lo que mejora la supervisión de las aplicaciones. Envía métricas y rastreos a múltiples soluciones de monitoreo AWS y asociadas. Al recopilar metadatos de sus AWS recursos, alinea el rendimiento de las aplicaciones con los datos de la infraestructura subyacente, lo que acelera la resolución de problemas. Además, es compatible con una variedad de AWS servicios y se puede utilizar de forma local.

Análisis de registros en AWS

Amazon CloudWatch Logs Insights permite la exploración, el análisis y la visualización de registros en tiempo real. Para un análisis más detallado de los archivos de registro, Amazon OpenSearch Service, que incluye Kibana, es una herramienta poderosa. CloudWatch Los registros pueden transmitir las entradas de registro al OpenSearch Servicio en tiempo real. Kibana, que se integra perfectamente OpenSearch, visualiza estos datos y ofrece una interfaz de búsqueda intuitiva.

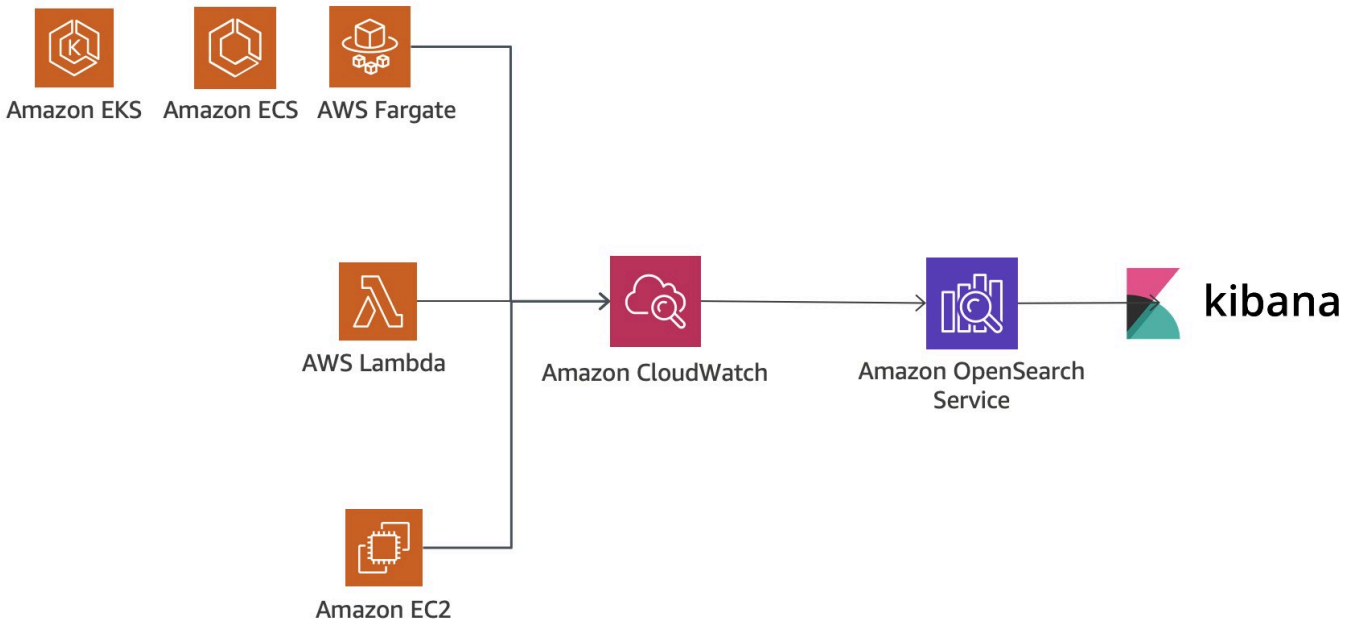

Figura 14: Análisis de registros con Amazon OpenSearch Service

Otras opciones de análisis

Para un análisis más detallado de los registros, Amazon Redshift, un servicio de almacenamiento de datos totalmente gestionado, y Quick, un servicio de inteligencia empresarial escalable, ofrecen soluciones eficaces. QuickSight proporciona una conectividad sencilla a varios servicios de AWS datos, como Redshift, RDS, Aurora, EMR, DynamoDB, Amazon S3 y Kinesis, lo que simplifica el acceso a los datos.

CloudWatch Los registros pueden transmitir las entradas de registro a Amazon Data Firehose, un servicio para entregar datos de streaming en tiempo real. QuickSight a continuación, utiliza los datos almacenados en Redshift para realizar análisis, informes y visualizaciones exhaustivos.

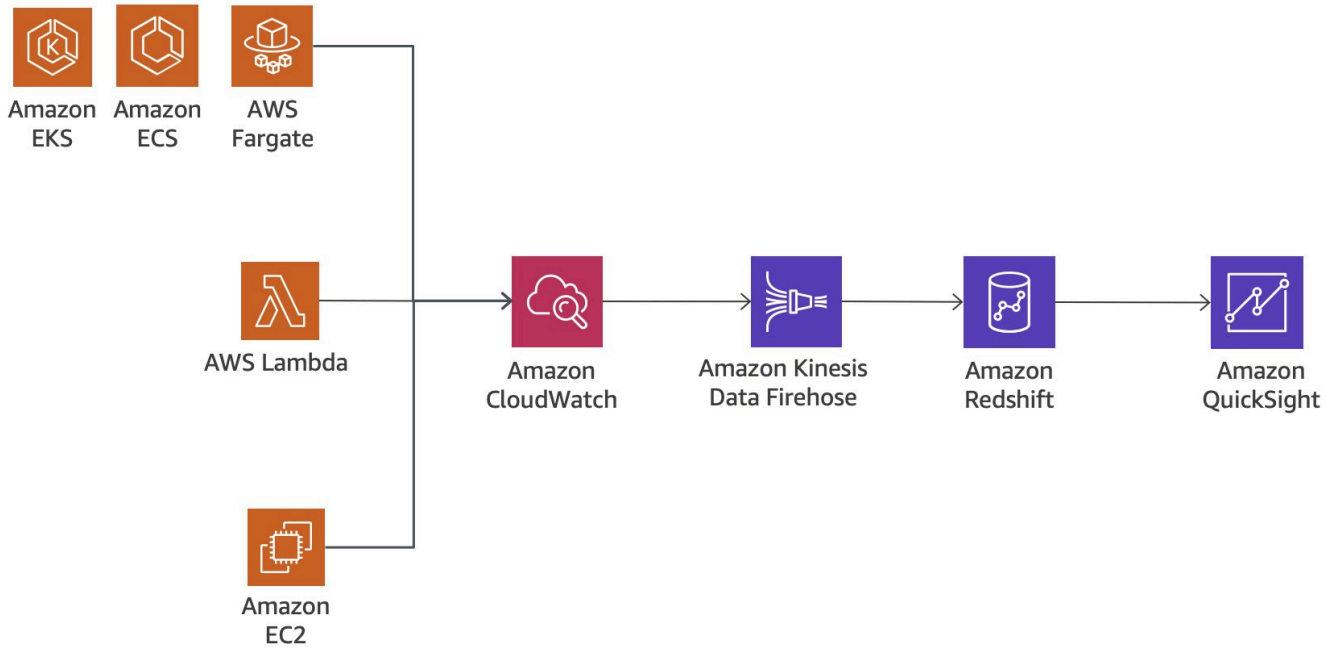

Figura 15: Análisis de registros con Amazon Redshift y Quick

Además, cuando los registros se almacenan en cubos de S3, un servicio de almacenamiento de objetos, los datos se pueden cargar en servicios como Redshift o EMR, una plataforma de macrodatos basada en la nube, que permite un análisis exhaustivo de los datos de registro almacenados.

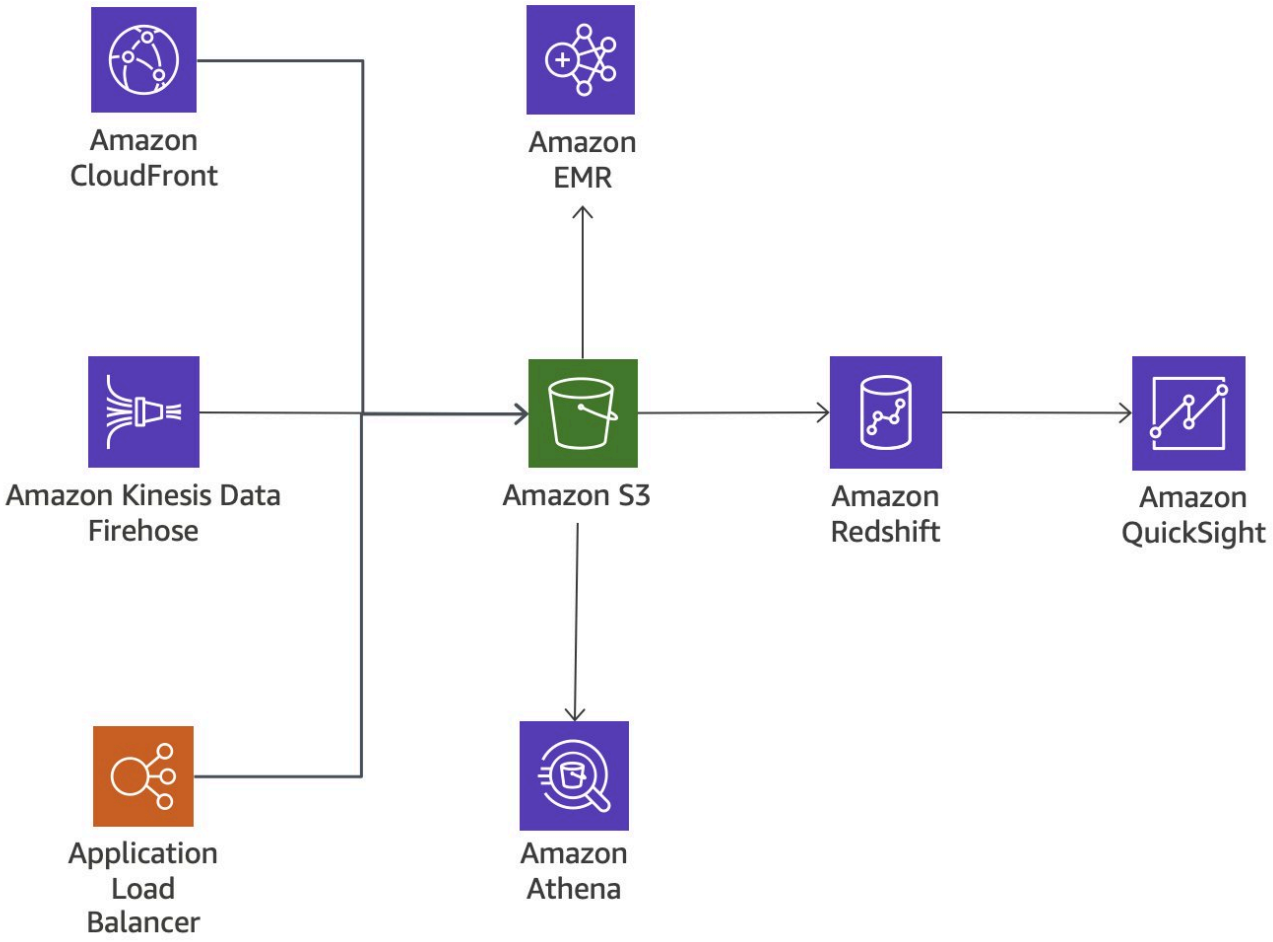

Figura 16: Optimización del análisis de registros: de los servicios a AWS QuickSight

Gestión de la conversación en la comunicación de microservicios

La charlatanería se refiere a una comunicación excesiva entre microservicios, que puede provocar ineficiencia debido al aumento de la latencia de la red. Es esencial gestionar la conversación de forma eficaz para que el sistema funcione correctamente.

Algunas herramientas clave para gestionar la conversación son REST APIs, HTTP y APIs gRPC. APIs REST APIs ofrece una gama de funciones avanzadas, como las claves de API, la regulación por cliente, la validación de solicitudes, la AWS WAF integración o los puntos finales de API privados. APIs Los HTTP están diseñados con funciones mínimas y, por lo tanto, tienen un precio más bajo. Para obtener más información sobre este tema y obtener una lista de las funciones principales que están disponibles en REST APIs y HTTP APIs, consulte Elegir entre REST APIs y HTTP APIs.

A menudo, los microservicios utilizan REST en lugar de HTTP para la comunicación debido a su uso generalizado. Sin embargo, en situaciones de gran volumen, la sobrecarga de REST puede provocar problemas de rendimiento. Esto se debe a que la comunicación utiliza el protocolo de enlace TCP, que es necesario para cada nueva solicitud. En esos casos, la API gRPC es una mejor opción. gRPC reduce la latencia, ya que permite múltiples solicitudes a través de una sola conexión TCP. gRPC también admite la transmisión bidireccional, lo que permite a los clientes y servidores enviar y recibir mensajes al mismo tiempo. Esto permite una comunicación más eficiente, especialmente para transferencias de datos grandes o en tiempo real.

Si las conversaciones persisten a pesar de haber elegido el tipo de API correcto, puede que sea necesario reevaluar la arquitectura de microservicios. La consolidación de los servicios o la revisión del modelo de dominio podrían reducir la cantidad de conversaciones y mejorar la eficiencia.

Uso de protocolos y almacenamiento en caché

Los microservicios suelen utilizar protocolos como gRPC y REST para la comunicación (consulte la sección Mecanismos de comunicación anterior). gRPC utiliza HTTP/2 para el transporte, mientras que REST suele utilizar HTTP/1.1. gRPC emplea búferes de protocolo para la serialización, mientras que REST suele utilizar JSON o XML. Para reducir la latencia y la sobrecarga de comunicación, se puede aplicar el almacenamiento en caché. Servicios como Amazon ElastiCache o la capa de almacenamiento en caché de API Gateway pueden ayudar a reducir el número de llamadas entre microservicios.

Auditoría

En una arquitectura de microservicios, es fundamental tener visibilidad de las acciones de los usuarios en todos los servicios. AWS proporciona herramientas como AWS CloudTrail la que registra todas las llamadas a la API realizadas y AWS CloudWatch la que se utiliza para capturar los registros de las aplicaciones. AWS Esto le permite realizar un seguimiento de los cambios y analizar el comportamiento en todos sus microservicios. Amazon EventBridge puede reaccionar rápidamente a los cambios del sistema, notificando a las personas adecuadas o incluso iniciando automáticamente los flujos de trabajo para resolver los problemas.

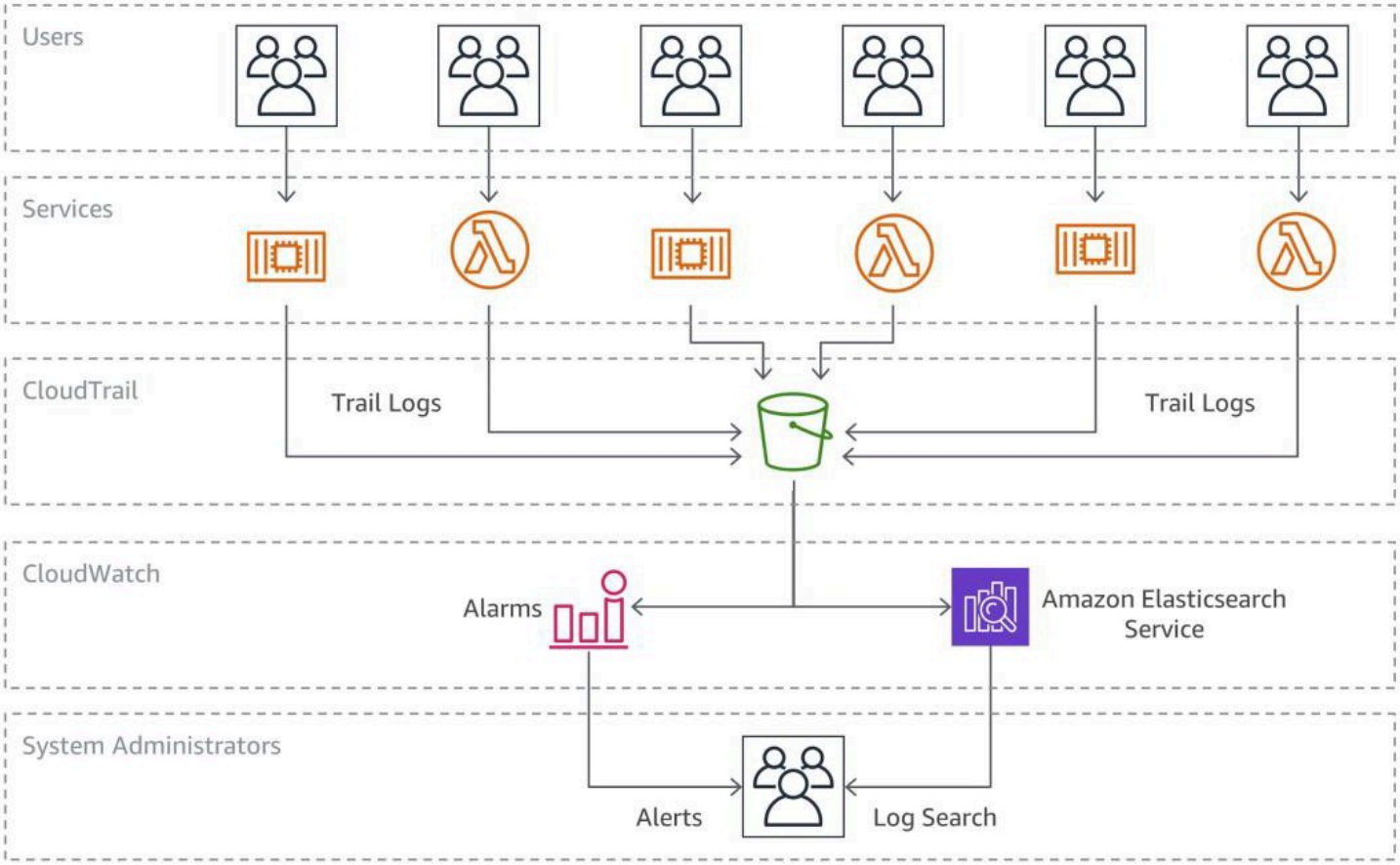

Figura 17: Auditoría y corrección en todos sus microservicios

Inventario de recursos y administración de cambios

En un entorno de desarrollo ágil con configuraciones de infraestructura en rápida evolución, la auditoría y el control automatizados son fundamentales. Reglas de AWS Config proporcionan un enfoque gestionado para supervisar estos cambios en todos los microservicios. Permiten definir

políticas de seguridad específicas que detectan, rastrean y envían alertas automáticamente sobre las infracciones de las políticas.

Por ejemplo, si se modifica la configuración de una API Gateway en un microservicio para aceptar el tráfico HTTP entrante en lugar de solo las solicitudes HTTPS, una AWS Config regla predefinida puede detectar esta infracción de seguridad. Registra el cambio para su auditoría y activa una notificación de SNS, lo que restablece el estado de conformidad.

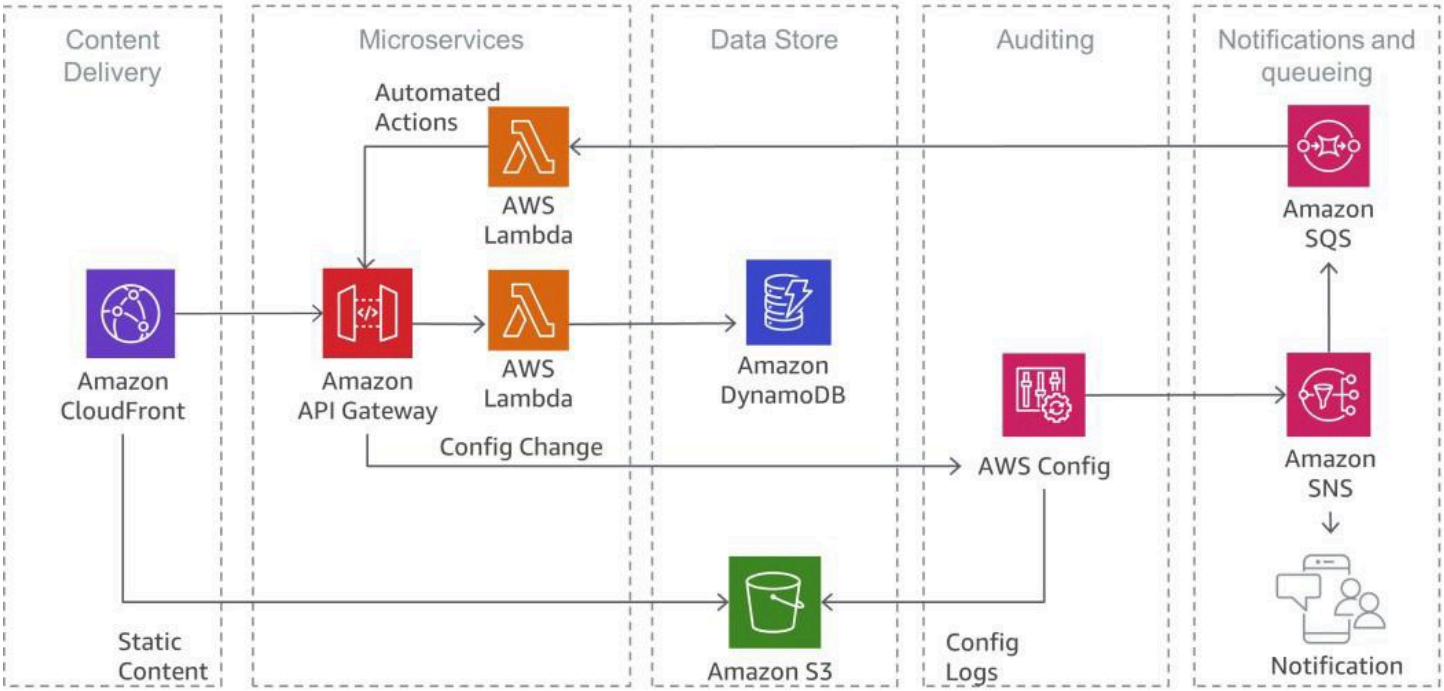

Figura 18: Detectar violaciones de seguridad con AWS Config

Conclusión

La arquitectura de microservicios, un enfoque de diseño versátil que ofrece una alternativa a los sistemas monolíticos tradicionales, ayuda a escalar las aplicaciones, a acelerar el desarrollo y a fomentar el crecimiento de la organización. Gracias a su adaptabilidad, se puede implementar mediante contenedores, enfoques sin servidor o una combinación de ambos, adaptándose a las necesidades específicas.

Sin embargo, no es una solución. one-size-fits-all Cada caso de uso requiere una evaluación meticulosa, dado el posible aumento de la complejidad arquitectónica y de las exigencias operativas. Sin embargo, si se abordan estratégicamente, los beneficios de los microservicios pueden superar con creces estos desafíos. La clave está en la planificación proactiva, especialmente en las áreas de observabilidad, seguridad y gestión de cambios.

También es importante tener en cuenta que, más allá de los microservicios, existen marcos arquitectónicos completamente diferentes, como las arquitecturas de IA generativa, como la Retrieval Augmented Generation (RAG), que ofrecen una gama de opciones que se adaptan mejor a sus necesidades.

AWS, con su sólido conjunto de servicios gestionados, permite a los equipos crear arquitecturas de microservicios eficientes y minimizar la complejidad de forma eficaz. El objetivo de este documento técnico es guiarlo a través de los AWS servicios relevantes y la implementación de los patrones clave. El objetivo es proporcionarle los conocimientos necesarios para aprovechar el potencial de los microservicios AWS, lo que le permitirá aprovechar sus beneficios y transformar su proceso de desarrollo de aplicaciones.

Colaboradores

Las siguientes personas y organizaciones han colaborado en este documento:

- Sascha Möllering, Arquitectura de soluciones, Amazon Web Services
- Christian Müller, Arquitectura de soluciones, Amazon Web Services
- Matthias Jung, Arquitectura de soluciones, Amazon Web Services
- Peter Dalbhanjan, Arquitectura de soluciones, Amazon Web Services
- Peter Chapman, Arquitectura de soluciones, Amazon Web Services
- Christoph Kassen, Arquitectura de soluciones, Amazon Web Services
- Umair Ishaq, Arquitectura de soluciones, Amazon Web Services
- Rajiv Kumar, Arquitectura de soluciones, Amazon Web Services
- Ramesh Dwarakanath, Arquitectura de soluciones, Amazon Web Services
- Andrew Watkins, Arquitectura de soluciones, Amazon Web Services
- Yann Stoneman, Arquitectura de soluciones, Amazon Web Services
- Mainak Chaudhuri, Arquitectura de soluciones, Amazon Web Services
- Gaurav Acharya, Arquitectura de soluciones, Amazon Web Services

Historial del documento

Para recibir notificaciones sobre las actualizaciones de este documento técnico, suscríbase a la fuente RSS.

Cambio	Descripción	Fecha
Actualización importante	Se agregó información sobre AWS Customer Carbon Footprint Tool EventBridge, Amazon AWS AppSync (GraphQL), AWS Lambda Layers, Lambda SnapStart , Large Language Models ()LLMs, Amazon Managed Streaming for Apache Kafka (MSK), Amazon Managed Workflows for Apache Airflow (MWAA), Amazon VPC Lattice,. AWS AppConfig Se agregó una sección separada sobre optimización de costos y sustentabilidad.	31 de julio de 2023
Actualizaciones menores	Se agregó Well-Architected al resumen.	13 de abril de 2022
Documento técnico actualizado	Integración de Amazon EventBridge, AWS, AMP OpenTelemetry, AMG, Container Insights, cambios de texto menores.	9 de noviembre de 2021
Actualizaciones menores	Diseño de página ajustado	30 de abril de 2021
Actualizaciones menores	Cambios menores en el texto.	1 de agosto de 2019

Documento técnico actualizado	Integración de Amazon EKS, AWS Fargate, Amazon MQ, PrivateLink AWS, AWS App Mesh y AWS Cloud Map	1 de junio de 2019
Documento técnico actualizado	Integración de las transmisiones de eventos de AWS Step Functions, AWS X-Ray y ECS.	1 de septiembre de 2017
Publicación inicial	Publicado sobre la implementación de microservicios en AWS.	1 de diciembre de 2016

> **Note**
>
> Para suscribirse a las actualizaciones de RSS, debe tener un complemento de RSS habilitado para el navegador que esté utilizando.

Avisos

Es responsabilidad de los clientes realizar su propia evaluación independiente de la información que contiene este documento. Este documento: (a) tiene únicamente fines informativos, (b) representa las ofertas y prácticas de AWS productos actuales, que están sujetas a cambios sin previo aviso, y (c) no crea ningún compromiso ni garantía por parte de AWS sus filiales, proveedores o licenciantes. AWS los productos o servicios se proporcionan «tal cual» sin garantías, representaciones o condiciones de ningún tipo, ya sean expresas o implícitas. Las responsabilidades y obligaciones de AWS sus clientes están reguladas por AWS acuerdos, y este documento no forma parte de ningún acuerdo entre sus clientes AWS y sus clientes ni lo modifica.

Copyright © 2023 Amazon Web Services, Inc. o sus filiales.

AWS Glosario

Para obtener la AWS terminología más reciente, consulte el AWS glosario de la Glosario de AWS Referencia.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.