Documento técnico de AWS

Implementación de microservicios en AWS

Implementación de microservicios en AWS: Documento técnico de AWS

Table of Contents

Implementación de microservicios en AWS

Fecha de publicación: 9 de noviembre de 2021 (Historial de revisión y colaboradores)

Resumen

Los microservicios son una estrategia organizativa y de arquitectura para el desarrollo de software creada para agilizar los ciclos de implementación, promover el nivel de innovación y distribución de responsabilidades, mejorar la capacidad de mantenimiento y ajuste de escala de las aplicaciones de software y aumentar la escala de las organizaciones que ofrecen software y servicios mediante el uso de un método ágil que ayuda a los equipos a trabajar de manera independiente. Cuando se usa una estrategia basada en microservicios, el software está compuesto por pequeños servicios que se comunican mediante interfaces de programa de aplicación (API) bien definidas que se pueden implementar de manera independiente. Los responsables de estos servicios son equipos autónomos reducidos. Esta estrategia ágil es clave para que pueda aumentar la escala de su organización correctamente.

Se han observado tres patrones habituales cuando los clientes de AWS crean microservicios: basado en API, basado en eventos y secuencia de datos. Este documento técnico presenta las tres estrategias y resume las características comunes de los microservicios, detalla los principales desafíos de crear microservicios y describe cómo los equipos de productos pueden aprovechar las soluciones de Amazon Web Services (AWS) para superar esos desafíos.

Debido a la naturaleza bastante compleja de varios temas que se tratan en este documento técnico, incluidos el almacenamiento de datos, la comunicación asíncrona y la detección de servicios, se recomienda que tenga en cuenta los requisitos específicos y los casos de uso de sus aplicaciones, además de la orientación proporcionada, antes de realizar opciones arquitectónicas.

Introducción

Las arquitecturas basadas en microservicios no son una estrategia completamente nueva en la ingeniería de software, sino más bien una combinación de varios conceptos consolidados y exitosos, como:

- Desarrollo de software ágil
- Arquitecturas orientadas a servicios

- Diseño que prioriza las API
- Integración continua y entrega continua (CI/CD)

En muchos casos, los patrones de diseño de metodología de la Aplicación de Doce Factores se aprovechan para los microservicios.

En primer lugar, el documento técnico describe los diferentes aspectos de una arquitectura de microservicios altamente escalable y tolerante a errores (interfaz de usuario, implementación de microservicios y almacén de datos) y cómo desarrollarla en AWS mediante el uso de tecnologías de contenedores. A continuación, recomienda los servicios de AWS para implementar una arquitectura de microservicios sin servidor típica para reducir la complejidad operativa.

La modalidad sin servidor se define como un modelo operativo debido a los siguientes principios:

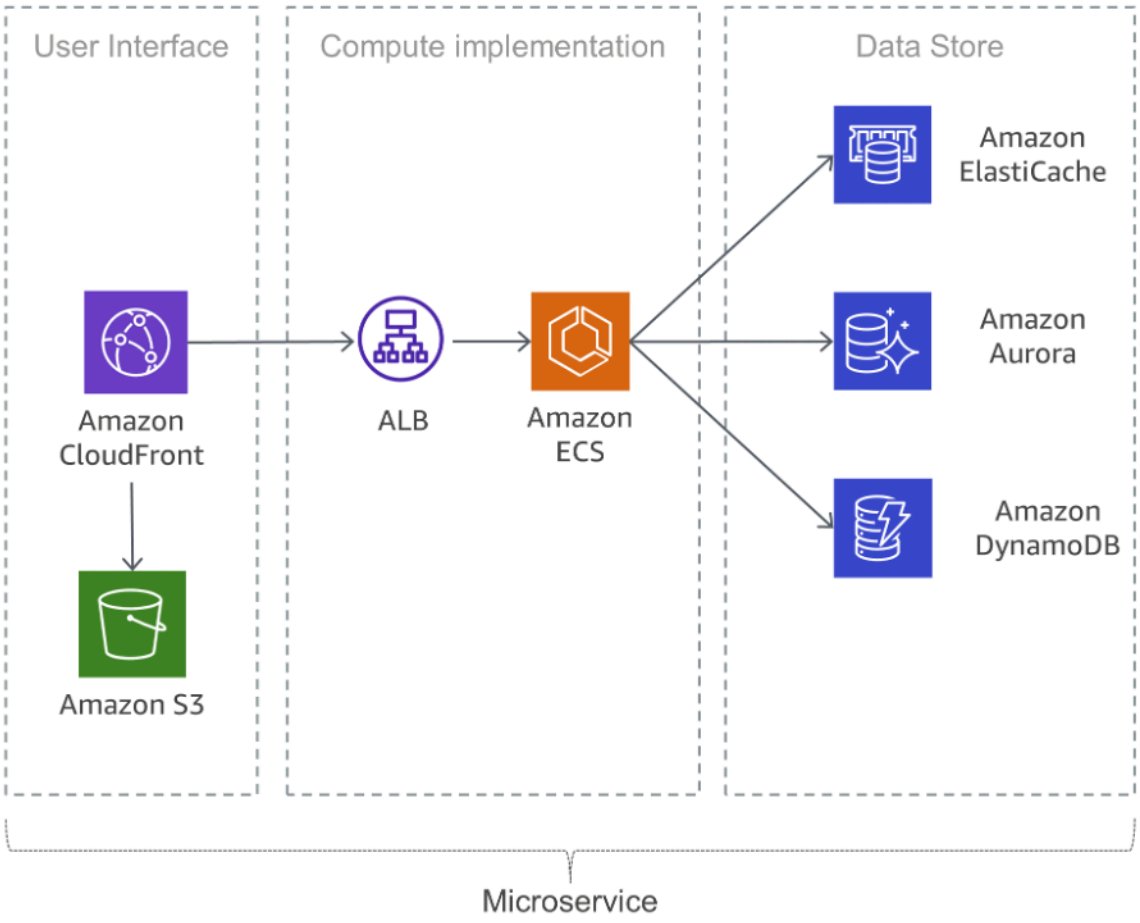
- Inexistencia de aprovisionamiento y administración de infraestructura
- Ajuste de escala automático por unidad de consumo
- Modelo de facturación de pago por valor
- Disponibilidad y tolerancia a errores integradas

Finalmente, el documento técnico observa el sistema en general y analiza los aspectos de las interrelaciones entre los servicios de una arquitectura de microservicios, como la supervisión distribuida y la auditoría, la coherencia de los datos y la comunicación asíncrona.

Este documento técnico solo se centra en las cargas de trabajo que se ejecutan en la nube de AWS. No cubre escenarios híbridos ni estrategias de migración. Para obtener más información sobre la migración, consulte el documento técnico Container Migration Methodology.

Arquitectura de microservicios en AWS

Las aplicaciones monolíticas típicas se crean mediante el uso de diferentes capas: una capa de interfaz de usuario (UI), una capa de negocio y una capa de persistencia. La idea central de una arquitectura basada en microservicios es dividir las funcionalidades en unidades verticales cohesivas, no mediante capas tecnológicas, sino a través de la implementación de un dominio específico. La siguiente figura representa una arquitectura de referencia para una aplicación de microservicios típica en AWS.



Aplicación de microservicios típica en AWS

Temas

- Interfaz de usuario
- Microservicios
- Almacén de datos

Interfaz de usuario

Las aplicaciones web modernas a menudo utilizan marcos de JavaScript para implementar una aplicación de una sola página que se comunica con una API de transferencia de estado representacional (REST) o RESTful. El contenido web estático se puede entregar mediante Amazon Simple Storage Service (S3) y Amazon CloudFront.

Dado que a los clientes de un microservicio se les prestan servicios desde la ubicación de borde más cercana y obtienen respuestas de una caché o un servidor proxy con conexiones optimizadas al origen, las latencias pueden reducirse significativamente. Sin embargo, los microservicios que se ejecutan cerca no se benefician de una red de entrega de contenido. De hecho, en algunos casos, esta estrategia puede aumentar la latencia. Una práctica recomendada consiste en implementar otros mecanismos de almacenamiento en caché para reducir la inestabilidad y minimizar las latencias. Para obtener más información, consulte el tema the section called “Ruido de datos innecesarios”.

Microservicios

Las API son la puerta principal de los microservicios, lo que significa que las API funcionan como el punto de entrada para la lógica de aplicaciones detrás de un conjunto de interfaces de programación, normalmente una API de servicios web RESTful. Esta API acepta y procesa las llamadas de los clientes y puede implementar funcionalidades como la administración del tráfico, el filtrado de solicitudes, el enrutamiento, el almacenamiento en caché, la autenticación y la autorización.

Implementación de microservicios

AWS ha integrado bloques de creación que respaldan el desarrollo de microservicios. Dos estrategias populares usan AWS Lambda y los contenedores de Docker con AWS Fargate.

Con AWS Lambda, simplemente cargue su código y deje que Lambda se encargue de todo lo necesario para ejecutar y escalar la implementación a fin de satisfacer la curva de demanda real con alta disponibilidad. No es necesario administrar infraestructura. Lambda es compatible con varios lenguajes de programación y puede invocarse desde otros servicios de AWS o recibir llamadas directamente desde cualquier aplicación web o móvil. Una de las principales ventajas de AWS Lambda es que puede trabajar con mayor rapidez: puede enfocarse en su lógica empresarial porque AWS se encarga de la seguridad y el ajuste de escala. La persistente estrategia de Lambda respalda la plataforma de escala ajustable.

Un enfoque común para reducir los esfuerzos operativos durante una implementación es basarla en contenedores. Las tecnologías de contenedores como Docker han aumentado su popularidad en los últimos años debido a los beneficios como portabilidad, productividad y eficiencia. La curva de aprendizaje con los contenedores puede ser abrupta y debe pensar en correcciones de seguridad para sus imágenes de Docker y en monitorización. Con Amazon Elastic Container Service (Amazon ECS) y Amazon Elastic Kubernetes Service (Amazon EKS), ya no hay necesidad de instalar, operar ni escalar una infraestructura de administración de clústeres propia. Con llamadas a la API, puede iniciar y detener las aplicaciones habilitadas para Docker, consultar el estado completo de su clúster y acceder a muchas características conocidas, como grupos de seguridad, equilibrio de carga, volúmenes de Amazon Elastic Block Store (Amazon EBS) y roles de AWS Identity and Access Management (IAM).

AWS Fargate es un motor de informática sin servidor para contenedores que funciona con Amazon ECS y Amazon EKS. Con Fargate, ya no deberá preocuparse por aprovisionar la cantidad suficiente de recursos de cómputo para sus aplicaciones en contenedores. Fargate puede lanzar decenas de miles de contenedores y puede ajustar su escala fácilmente para ejecutar las aplicaciones más importantes.

Amazon ECS admite estrategias y restricciones de asignación de contenedores para personalizar la manera en la que Amazon ECS asigna y finaliza las tareas. Una restricción de asignación de tareas es una regla que se define durante la asignación de tareas. Puede asociar atributos, que son esencialmente pares clave-valor, a sus instancias de contenedor y luego usar una restricción para asignar tareas basadas en los atributos. Por ejemplo, puede usar restricciones para asignar ciertos microservicios según el tipo de instancia o la capacidad de la instancia, como las instancias alimentadas por GPU.

Amazon EKS ejecuta versiones actualizadas del software de código abierto Kubernetes, lo que le permite utilizar todos los complementos y las herramientas existentes de la comunidad de Kubernetes. Las aplicaciones que se ejecutan en Amazon EKS son totalmente compatibles con las aplicaciones que se ejecutan en cualquier entorno de Kubernetes estándar, independientemente de si se ejecutan en nubes públicas o centros de datos locales. Amazon EKS integra IAM con Kubernetes, lo que le permite registrar entidades de IAM con el sistema de autenticación nativo en Kubernetes. No es necesario configurar credenciales manualmente para autenticarlas con los nodos de plano de control de Kubernetes. La integración con IAM le permite utilizar IAM para realizar una autenticación directa con el plano de control mismo y ofrecer acceso pormenorizado al punto de conexión público de sus planos de control de Kubernetes.

Las imágenes de Docker utilizadas en Amazon ECS y Amazon EKS se pueden almacenar en Amazon Elastic Container Registry (Amazon ECR). Amazon ECR elimina la necesidad de operar y escalar la infraestructura que se requiere para alimentar el registro de contenedores.

La integración y entrega continuas (CI/CD) son prácticas recomendadas y una parte fundamental de una iniciativa de DevOps que permite realizar cambios en el software rápidamente y, al mismo tiempo, mantener un nivel de estabilidad y seguridad en el sistema. Sin embargo, esto está fuera del alcance de este documento técnico. Para obtener más información, consulte el documento técnico Practicing Continuous Integration and Continuous Delivery on AWS.

Enlaces privados

AWS PrivateLink es una tecnología escalable y de alta disponibilidad que le permite conectar su nube privada virtual (VPC) de forma privada a servicios de AWS compatibles, servicios alojados en otras cuentas de AWS (servicios de punto de conexión de la VPC) y servicios compatibles de socios de AWS Marketplace. No necesita una puerta de enlace de Internet, un dispositivo de traducción de direcciones de red, una dirección IP pública, una conexión de AWS Direct Connect ni una conexión de VPN para comunicarse con el servicio. El tráfico entre su VPC y el servicio no sale de la red de Amazon.

Los enlaces privados son una excelente manera de aumentar el aislamiento y la seguridad de la arquitectura de microservicios. Un microservicio, por ejemplo, podría implementarse en una VPC totalmente independiente, encabezada por un equilibrador de carga y expuesta a otros microservicios a través de un punto de conexión de AWS PrivateLink. Con esta configuración, gracias al AWS PrivateLink, el tráfico de red hacia y desde el microservicio nunca atraviesa la Internet pública. Un caso de uso para dicho aislamiento incluye el cumplimiento normativo de los servicios que manejan datos confidenciales como PCI, HIPPA y el Escudo de Privacidad UE-EE. UU. Además, AWS PrivateLink permite conectar microservicios en diferentes cuentas y VPC de Amazon, sin necesidad de reglas de firewall, definiciones de rutas ni tabla de enrutamiento, lo que simplifica la administración de la red. Al utilizar PrivateLink, los proveedores de software como servicio (SaaS) y los ISV pueden ofrecer sus soluciones basadas en microservicios con aislamiento operativo completo además de acceso seguro.

Almacén de datos

El almacén de datos se utiliza para conservar los datos que necesitan los microservicios. Las tiendas populares para datos de sesión son los cachés en memoria como Memcached o Redis. AWS ofrece ambas tecnologías como parte del servicio administrado Amazon ElastiCache.

Poner una caché entre los servidores de aplicaciones y una base de datos es un mecanismo común para reducir la carga de lectura de la base de datos, lo que, a su vez, puede habilitar el uso de recursos a fin de admitir más escrituras. Los cachés también pueden mejorar la latencia.

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

Sin embargo, las bases de datos relacionales no están diseñadas para escalas interminables, lo que puede hacer que la aplicación de técnicas tendientes a admitir un gran número de consultas resulte muy difícil y use demasiado tiempo.

Las bases de datos NoSQL se han diseñado para favorecer la escalabilidad, el rendimiento y la disponibilidad por sobre la coherencia de las bases de datos relacionales. Un aspecto importante de las bases de datos NoSQL es que, generalmente, no imponen 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 una sola cosa bien, normalmente tienen un modelo de datos simplificado que podría ser adecuado para la persistencia NoSQL. Es importante saber que las bases de datos NoSQL tienen diferentes patrones de acceso en comparación con las bases de datos relacionales. Por ejemplo, no es posible unir tablas. Si esto es necesario, se debe implementar lógica en la aplicación. Puede utilizar Amazon DynamoDB para crear una tabla de base de datos capaz de almacenar y recuperar cualquier cantidad de datos, así como de satisfacer cualquier nivel de tráfico de solicitud. DynamoDB ofrece un rendimiento inferior a un milisegundo. Sin embargo, hay ciertos casos de uso que requieren tiempos de respuesta de microsegundos. Amazon DynamoDB Accelerator (DAX) ofrece capacidades de almacenamiento en caché para obtener acceso a los datos.

DynamoDB también ofrece una característica de escalado automático para ajustar de manera dinámica la capacidad de procesamiento en respuesta al tráfico real. Sin embargo, existen casos en los que la planificación de la capacidad resulta compleja o imposible debido a la existencia de incrementos de actividad pronunciados de duración breve en las aplicaciones. Para dichos casos, DynamoDB ofrece una opción simple bajo demanda que permite pagar por solicitud. DynamoDB con modalidad bajo demanda es capaz de atender miles de solicitudes por segundo de manera instantánea sin planificación de capacidad.

Reducción de la complejidad operativa

La arquitectura descrita en este documento técnico ya utiliza servicios administrados, pero aún debemos administrar instancias de Amazon Elastic Compute Cloud (Amazon EC2). Podemos reducir aún más los esfuerzos operativos necesarios para ejecutar, mantener y supervisar los microservicios mediante el uso de una arquitectura completamente sin servidor.

Temas

- Implementación de API
- Microservicios sin servidor
- Recuperación de desastres
- Alta disponibilidad
- Implementación de aplicaciones basadas en Lambda

Implementación de API

Diseñar, implementar, supervisar, mejorar continuamente y mantener una API pueden llevar mucho tiempo. A veces es necesario ejecutar diferentes versiones de la API a fin de garantizar la compatibilidad con versiones anteriores a todos los clientes. Las diferentes etapas del ciclo de desarrollo (por ejemplo, desarrollo, pruebas y producción) multiplican aún más los esfuerzos operativos.

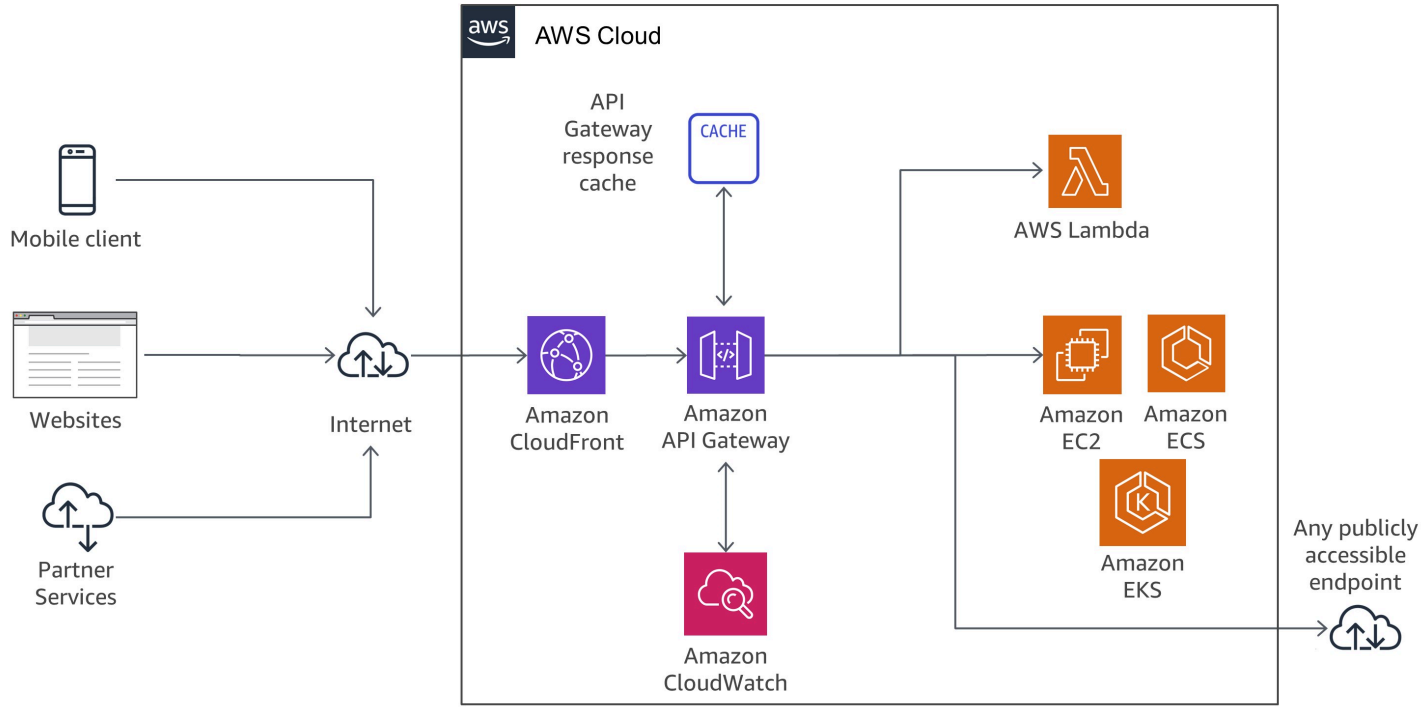
La autorización es una característica clave para todas las API, pero generalmente es complejo compilarla e implica trabajo repetitivo. Cuando una API se publica y se comienza a utilizar masivamente, el siguiente desafío es administrar, supervisar y monetizar el ecosistema de desarrolladores terceros que utilizan la API.

Otras características y desafíos importantes incluyen las solicitudes de limitación controlada para proteger servicios de backend, el almacenamiento en caché de las respuestas de la API, la gestión de transformaciones de solicitudes y respuestas, y la generación de definiciones de API y documentación con herramientas como Swagger.

Amazon API Gateway aborda esos desafíos y reduce la complejidad operativa de crear y mantener las API RESTful. API Gateway le permite crear sus API mediante programación al importar las definiciones de Swagger mediante el uso de la API de AWS o la consola de administración de AWS.

API Gateway funciona como puerta de entrada para cualquier aplicación web que se ejecute en Amazon EC2, Amazon ECS, AWS Lambda o en cualquier entorno local. Básicamente, API Gateway le permite ejecutar las API sin necesidad de administrar servidores.

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



Flujo de llamadas de API Gateway

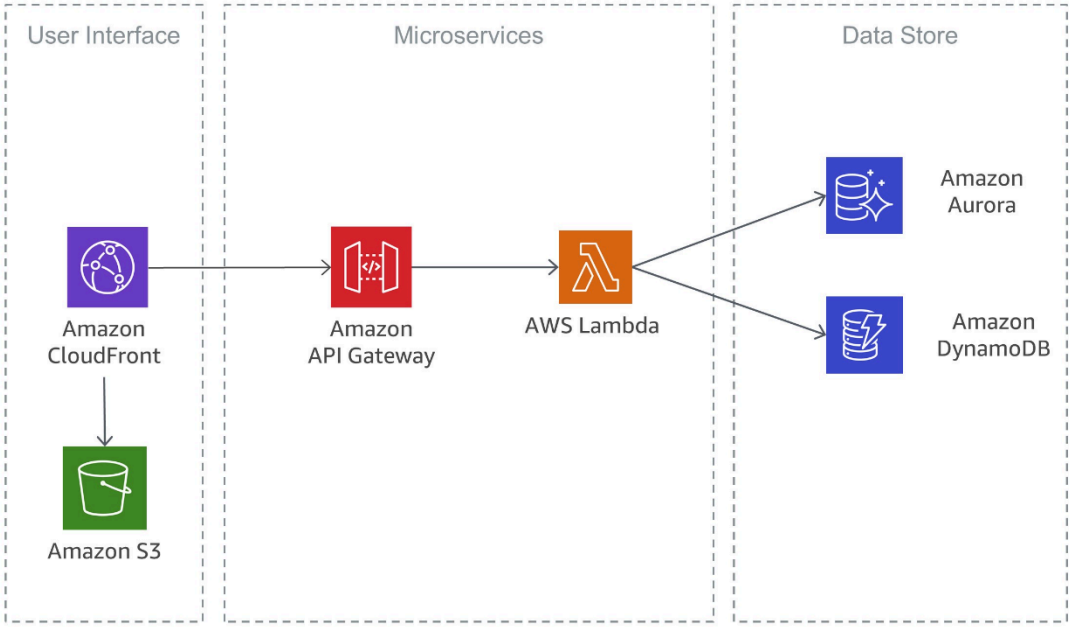
Microservicios sin servidor

«No existe servidor más fácil de administrar que ningún servidor».

Deshacerse de los servidores es una excelente manera de eliminar la complejidad operativa.

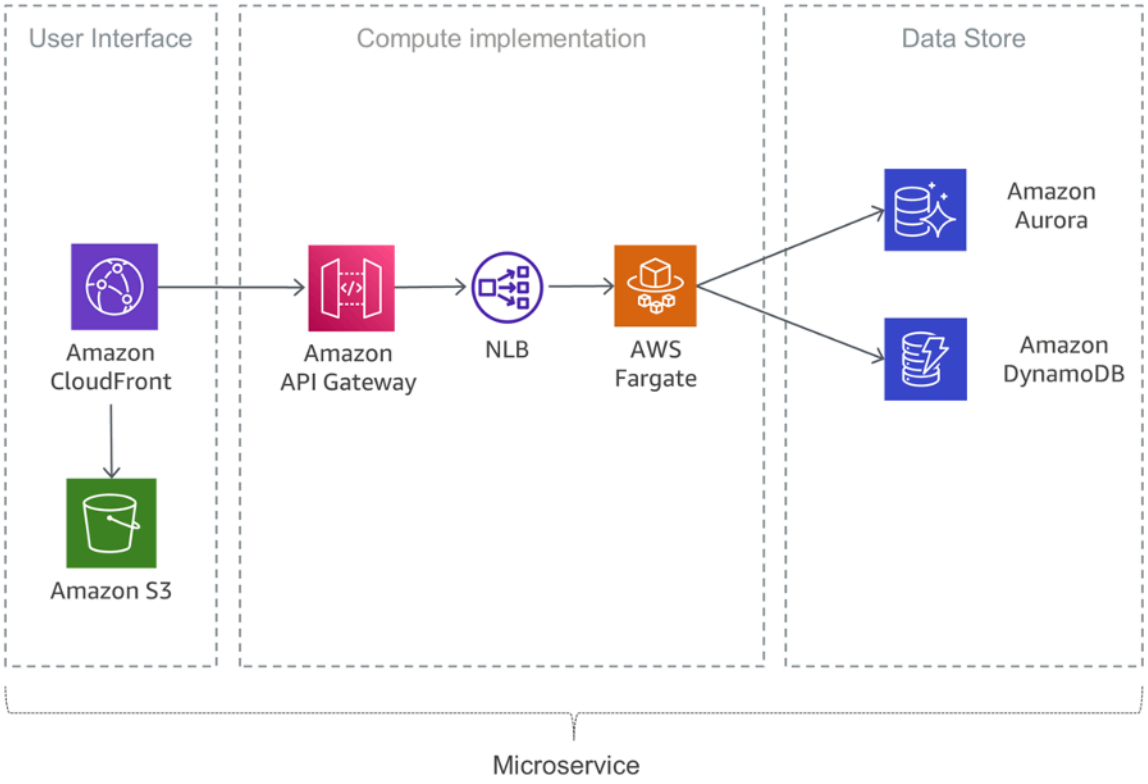
Lambda está estrechamente integrado con API Gateway. La capacidad para realizar llamadas sincrónicas desde API Gateway a Lambda permite crear aplicaciones sin servidor y está descrita en detalle en la Guía del desarrollador de Amazon API Gateway.

La siguiente figura muestra la arquitectura de un microservicio sin servidor con AWS Lambda, donde el servicio completo se crea a partir de servicios administrados, lo que elimina la sobrecarga vinculada con la arquitectura que implica diseñar para lograr escala y alta disponibilidad. Además, también elimina los esfuerzos operativos relacionados con la ejecución y supervisión de la infraestructura subyacente del microservicio.



Microservicio sin servidor que utiliza AWS Lambda

La siguiente figura muestra una implementación similar que también está basada en servicios sin servidor. En esta arquitectura, los contenedores Docker se utilizan con Fargate, por lo que no debe ocuparse de la infraestructura subyacente. Además de DynamoDB, también se utiliza Amazon Aurora Serverless, que es una configuración de escalado automático bajo demanda para Amazon Aurora (edición compatible con MySQL), donde la base de datos se iniciará, se cerrará y ajustará la escala de su capacidad automáticamente en función de las necesidades de su aplicación.



Microservicio sin servidor con Fargate

Recuperación de desastres

Como se mencionó anteriormente en la introducción de este documento técnico, las aplicaciones de microservicios típicas se implementan utilizando los patrones de aplicación de doce factores. La sección Procesos establece que «los procesos de doce factores no tienen estado y no comparten nada. Cualquier dato que deba conservarse debe almacenarse en un servicio de respaldo con estado, normalmente una base de datos».

Para una arquitectura de microservicios típica, esto significa que el enfoque principal para la recuperación de desastres debe estar en los servicios descendentes que mantienen el estado de la aplicación. Por ejemplo, pueden ser sistemas de archivos, bases de datos o colas, por ejemplo. Al crear una estrategia de recuperación de desastres, las organizaciones suelen planificar el objetivo de tiempo de recuperación y el objetivo de punto de recuperación.

El objetivo de tiempo de recuperación es el retraso máximo aceptable entre la interrupción del servicio y la restauración del servicio. Este objetivo, definido por la organización, determina lo que se considera una ventana de tiempo aceptable mientras el servicio no está disponible.

El objetivo de punto de recuperación es la cantidad de tiempo máxima aceptable desde el último punto de recuperación de datos. Este objetivo, definido por la empresa, determina lo que se considera una pérdida aceptable de datos entre el último punto de recuperación y la interrupción del servicio.

Para obtener más información, consulte el documento técnico Disaster Recovery of Workloads on AWS: Recovery in the Cloud.

Alta disponibilidad

En esta sección se analiza en detalle la alta disponibilidad para las diferentes opciones de computación.

Amazon EKS ejecuta instancias de plano datos y control de Kubernetes en varias zonas de disponibilidad para garantizar una alta disponibilidad. Amazon EKS detecta y reemplaza automáticamente las instancias de plano de control en mal estado y proporciona actualizaciones automatizadas de versiones y parches para ellas. Este plano de control consta de al menos dos nodos de servidor de API y tres nodos etcd que se ejecutan en tres zonas de disponibilidad dentro de una región. Amazon EKS utiliza la arquitectura de Regiones de AWS para mantener una alta disponibilidad.

Amazon ECR aloja imágenes en una arquitectura de alta disponibilidad y alto rendimiento, lo que le permite implementar imágenes de manera fiable para aplicaciones de contenedores en todas las zonas de disponibilidad. Amazon ECR trabaja con Amazon EKS, Amazon ECS y AWS Lambda, lo que simplifica el flujo de trabajo del desarrollo a la producción.

Amazon ECS es un servicio regional que simplifica la ejecución de contenedores con alta disponibilidad en diversas zona de disponibilidad dentro de una Región de AWS. Amazon ECS incluye varias estrategias de programación que colocan los contenedores en clústeres en función de las necesidades de sus recursos (por ejemplo, CPU o RAM) y requisitos de disponibilidad.

AWS Lambda ejecuta la función en varias zonas de disponibilidad para asegurarse de que está disponible para procesar eventos en caso de una interrupción del servicio en una sola zona. Si configura la función para conectarse a una nube privada virtual (VPC) en su cuenta, especifique subredes en varias zonas de disponibilidad para garantizar una alta disponibilidad.

Implementación de aplicaciones basadas en Lambda

Puede utilizar AWS CloudFormation para especificar, implementar y configurar aplicaciones sin servidor.

El AWS Serverless Application Model (AWS SAM) es una forma conveniente de definir aplicaciones sin servidor. CloudFormation admite AWS SAM de forma nativa y define una sintaxis simplificada para expresar recursos sin servidor. Para implementar su aplicación, especifique los recursos que necesita como parte de su aplicación, junto con las políticas de permisos asociadas en una plantilla de CloudFormation, empaquete los dispositivos de implementación e implemente la plantilla. SAM Local es una herramienta de la AWS Command Line Interface (AWS CLI) que se basa en AWS SAM y que proporciona un entorno para desarrollar, probar y analizar localmente aplicaciones sin servidor antes de cargarlas en el tiempo de ejecución de Lambda. Puede usar AWS SAM Local para crear un entorno de pruebas local que simule el entorno en tiempo de ejecución de AWS.

Componentes de los sistemas distribuidos

Después de analizar la manera en la que AWS puede resolver los desafíos relacionados con los microservicios individuales, la atención se desplaza a los desafíos entre servicios, como la detección de servicios, la coherencia de los datos, la comunicación asíncrona y la supervisión distribuida y la auditoría.

Temas

Detección de servicios

Uno de los principales desafíos de las arquitecturas de microservicios es permitir que los servicios se detecten e interactúen entre sí. Las características distribuidas de las arquitecturas de microservicios no solo dificultan la comunicación de los servicios, sino que también presentan otros desafíos, tal como verificar el estado de dichos sistemas y anunciar cuándo comienzan a estar disponibles aplicaciones nuevas. También debe decidir cómo y dónde almacenar la información del almacén de metadatos, como los datos de configuración que se pueden usar en las aplicaciones. En esta sección, analizamos varias técnicas a fin de detectar servicios en AWS para arquitecturas basadas en microservicios.

Detección de servicios basada en DNS

Amazon ECS ahora incluye la detección de servicios integrada, lo que facilita que sus servicios en contenedores puedan detectarse y conectarse entre sí.

Con anterioridad, a fin de garantizar que los servicios pudiesen detectarse y conectarse entre sí, debía configurar y ejecutar un sistema de detección de servicios propio basado en Amazon Route 53, AWS Lambda y secuencias de eventos ECS, o bien conectar cada servicio a un equilibrador de carga.

Amazon ECS crea y administra un registro de nombres de servicios mediante el uso de la API Auto Naming de Route 53. Los nombres se asignan automáticamente a un conjunto de registros de DNS para que pueda hacer referencia a un servicio por el nombre en su código y escribir consultas de DNS para lograr que el nombre se resuelva en el punto de enlace del servicio en el tiempo de ejecución. Puede especificar condiciones de verificación de estado en una definición de tarea de servicio y Amazon ECS garantizará que únicamente puntos de enlace del servicio que funcionen de manera correcta sean devueltos por una búsqueda de servicios.

Además, también puede utilizar la detección de servicios unificada para los servicios administrados por Kubernetes. Para lograr esta integración, AWS contribuyó al proyecto External DNS, un proyecto de incubadora de Kubernetes.

Otra opción es aprovechar las capacidades de AWS Cloud Map. AWS Cloud Map amplía las capacidades de las API Auto Naming mediante el suministro de un registro de servicios para recursos, como protocolos de Internet (IP), localizadores uniformes de recursos (URL) y nombres de recurso de Amazon (ARN), y el aprovisionamiento de un mecanismo de detección de servicios basado en API con una propagación de cambios más ágil y la capacidad para utilizar atributos a fin de reducir el conjunto de recursos detectados. Los recursos de Auto Naming de Route 53 existentes se actualizan automáticamente a AWS Cloud Map.

Software de terceros

Un enfoque diferente para implementar la detección de servicios es usar software de terceros como HashiCorp Consul, etcd o Netflix Eureka. Los tres ejemplos son almacenes de clave-valor fiables y distribuidos. Para HashiCorp Consul, hay un AWS Quick Start que configura un entorno de nube de AWS flexible y escalable, y que lanza HashiCorp Consul automáticamente con la configuración que usted elija.

Mallas de servicios

En una arquitectura de microservicios avanzada, la aplicación en sí puede estar conformada por cientos, o inclusive miles, de servicios. La parte más compleja de la aplicación no son los servicios en sí, sino la comunicación entre ellos. Las mallas de servicios son una capa adicional para encargarse de la comunicación entre los servicios, responsable de supervisar y controlar el tráfico en las arquitecturas de microservicios. De esta manera, es posible que esta capa se encargue por completo de tareas, como la detección de servicios.

Normalmente, una malla de servicios se divide en un plano de datos y un plano de control. El plano de datos consiste en un conjunto de proxies inteligentes que se implementan con el código de la

aplicación como un proxy sidecar especial que intercepta todas las comunicaciones de red entre los servicios. El plano de control es responsable de la comunicación con los proxies.

Las mallas de servicios son transparentes, por lo que pasa desapercibida para los desarrolladores, que no deben realizar cambios en el código existente de la aplicación. AWS App Mesh es una malla de servicios que ofrece redes a nivel de las aplicaciones para permitir la comunicación entre sus servicios en varios tipos de infraestructura de cómputo. App Mesh estandariza la manera en la que sus servicios se comunican, lo que le otorga una visibilidad completa y garantiza un nivel de disponibilidad alto en sus aplicaciones.

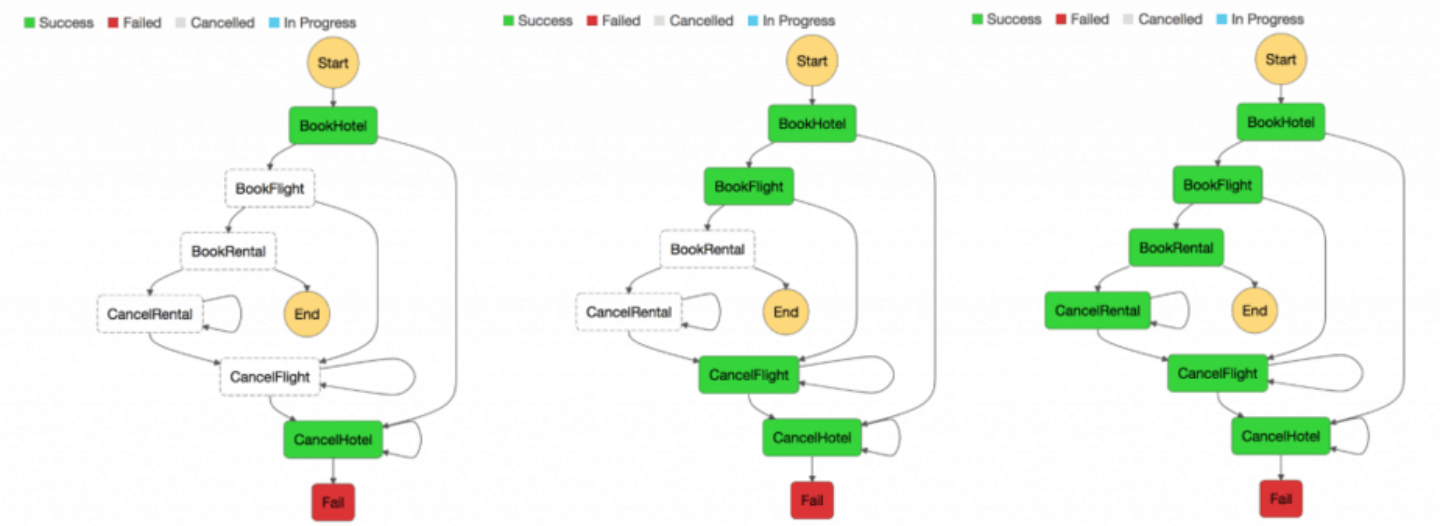
Puede utilizar AWS App Mesh con microservicios nuevos o existentes que se ejecutan en AWS Fargate, Amazon ECS, Amazon EKS y Kubernetes autoadministrados en AWS. App Mesh puede supervisar y controlar comunicaciones de microservicios que se ejecuten en clústeres, sistemas de orquestación o VPC como una aplicación única sin realizar modificaciones en el código.

Administración de datos distribuidos

Las aplicaciones monolíticas generalmente están respaldadas por una gran base de datos relacional, que define un único modelo de datos común a todos los componentes de la aplicación. En un enfoque de microservicios, una base de datos central así evitaría el objetivo de crear componentes descentralizados e independientes. Cada componente del microservicio debe tener su propia capa de persistencia de datos.

La administración de datos distribuidos, sin embargo, plantea nuevos retos. Como consecuencia del Teorema CAP, las arquitecturas de microservicios distribuidos intercambian la consistencia por el rendimiento y necesitan adoptar la consistencia final.

En un sistema distribuido, las transacciones empresariales pueden abarcar varios microservicios. Como no pueden utilizar una transacción ACID única, puede terminar con ejecuciones parciales. En este caso, necesitaríamos algún tipo de lógica de control para volver a realizar las transacciones que ya fueron procesadas. Para este fin, normalmente se utiliza el patrón Saga distribuido. Cuando ocurre un error en una transacción comercial, Saga orquesta una serie de transacciones de compensación para anular los cambios que fueron realizados por las transacciones anteriores. AWS Step Functions facilita la implementación de un coordinador de ejecución de Saga, como se muestra en la siguiente figura.



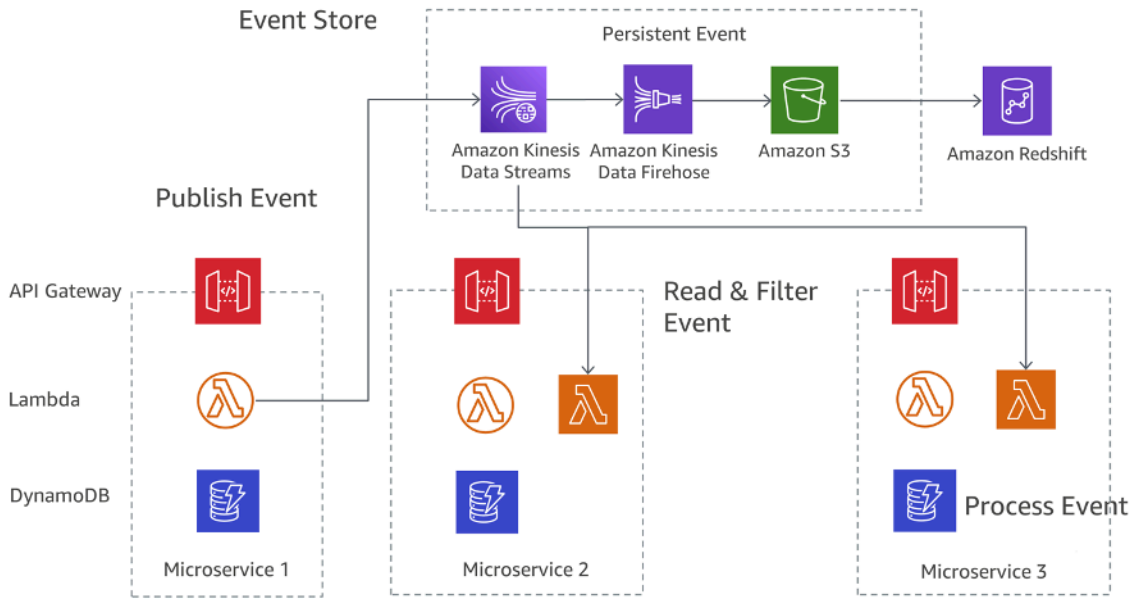
Coordinador de ejecución de Saga

La creación de un almacén centralizado de datos de referencia clave que se depuran mediante herramientas y procedimientos de administración de datos básicos proporciona un medio para que los microservicios puedan sincronizar sus datos clave y posiblemente restaurar el estado anterior. El uso de Lambda con eventos programados de Amazon CloudWatch Events puede crear un mecanismo simple de limpieza y desduplicación.

Es muy común que los cambios de estado afecten a más de un microservicio. En esos casos, el abastecimiento de eventos ha demostrado ser un patrón útil. La idea central detrás del abastecimiento de eventos es representar y mantener cada cambio de la aplicación como un registro de eventos. En lugar de mantener el estado de la aplicación, los datos se almacenan como un stream de eventos. Los sistemas de control de versiones y registro de transacciones de la base de datos son dos ejemplos bien conocidos para el abastecimiento de eventos. El abastecimiento de eventos tiene un par de beneficios: el estado puede determinarse y reconstruirse en cualquier momento. Naturalmente produce una pista de auditoría persistente y también facilita la depuración.

En el contexto de las arquitecturas de microservicios, el abastecimiento de eventos permite desacoplar diferentes partes de una aplicación mediante el uso de un patrón de publicación o suscripción, y alimenta los mismos datos de eventos en diferentes modelos de datos para microservicios separados. El abastecimiento de eventos se usa con frecuencia junto con el patrón Comando, Consulta, Responsabilidad, Segregación (CQRS) para desacoplar la lectura de las cargas de trabajo de escritura y optimizar tanto el rendimiento como la escalabilidad y la seguridad. En los sistemas de administración de datos tradicionales, los comandos y las consultas se ejecutan en el mismo repositorio de datos.

En la siguiente figura se muestra cómo se puede implementar el patrón de abastecimiento de eventos en AWS. Amazon Kinesis Data Streams es el componente principal del almacén central de eventos que registra los cambios de las aplicaciones como eventos y los conserva en Amazon S3. La figura representa tres microservicios diferentes compuestos por Amazon API Gateway, AWS Lambda y Amazon DynamoDB. Las flechas indican el flujo de los eventos: cuando el Microservicio 1 experimenta un cambio de estado de evento, escribe un mensaje en Kinesis Data Streams para publicar un evento. Todos los microservicios ejecutan su propia aplicación de Kinesis Data Streams en AWS Lambda, que lee una copia del mensaje, la filtra según la relevancia para el microservicio y, posiblemente, la reenvía para su procesamiento posterior. Si la función devuelve un error, Lambda volverá a intentar ejecutar el lote hasta que el procesamiento se realice correctamente o los datos caduquen. Para evitar particiones detenidas, puede configurar la asignación del origen de eventos para que vuelva a intentarlo con un tamaño de lote menor, para que limite el número de reintentos o para que se descarten los registros que sean muy antiguos. Si desea conservar los eventos descartados, puede configurar la asignación del origen de eventos para que envíe información sobre los lotes con errores a una cola de Amazon Simple Queue Service (Amazon SQS) o a un tema de Amazon Simple Notification Service (Amazon SNS).



Patrón de abastecimiento de eventos en AWS

Amazon S3 almacena de forma duradera todos los eventos en todos los microservicios y es la única fuente de la verdad cuando se trata de depurar, recuperar el estado de la aplicación o auditar los cambios de la aplicación. Hay dos principales razones por las que registros podrían entregarse más

de una vez en su aplicación Kinesis Data Streams: reintentos de los productores y reintentos de los consumidores. Su aplicación debe prever y administrar de forma adecuada el procesamiento de registros individuales varias veces.

Administración de la configuración

En una arquitectura típica de microservicios con docenas de servicios diferentes, cada servicio necesita acceso a varios servicios descendentes y componentes de infraestructura que exponen los datos al servicio. Algunos ejemplos pueden ser colas de mensajes, bases de datos y otros microservicios. Uno de los desafíos clave es configurar cada servicio de manera coherente para proporcionar información sobre la conexión a la infraestructura y los servicios descendentes. Además, la configuración también debe contener información sobre el entorno en el que funciona el servicio, y no debería ser necesario reiniciar la aplicación para usar nuevos datos de configuración.

El tercer principio de los patrones de la aplicación de doce factores cubre este tema: «La aplicación de doce factores almacena la configuración en variables de entorno (a menudo abreviadas como env vars o env)». Para Amazon ECS, las variables de entorno se pueden transferir al contenedor mediante el parámetro de definición de contenedor de entorno que se asigna a la opción `--env` de ejecución de docker. Las variables de entorno se pueden transferir a sus contenedores en bloque, mediante el parámetro de definición de contenedor de `environmentFiles` para enumerar uno o más archivos que contienen las variables de entorno. El archivo debe estar alojado en Amazon S3. En AWS Lambda, el tiempo de ejecución hace que las variables de entorno estén disponibles para el código y establece variables de entorno adicionales que contienen información sobre la función y la solicitud de invocación. Para Amazon EKS, puede definir variables de entorno en el campo env-field del manifiesto de configuración del pod correspondiente. Una forma diferente de usar variables env-es utilizar un ConfigMap.

Comunicación asíncrona y mensajería liviana

La comunicación en las aplicaciones tradicionales y monolíticas es simple: una parte de la aplicación usa llamadas a métodos o un mecanismo interno de distribución de eventos para comunicarse con las demás partes. Si la misma aplicación se implementa con microservicios desacoplados, la comunicación entre las diferentes partes de la aplicación debe implementarse con comunicación en red.

Comunicación basada en REST

El protocolo HTTP/S es la forma más popular de implementación de comunicación síncrona entre los microservicios. En la mayoría de los casos, las API de REST utilizan el protocolo HTTP como una capa de transporte. El estilo de la arquitectura de REST se basa en la comunicación sin estado, las interfaces uniformes y los métodos estándar.

Con API Gateway puede crear una API que actúe de puerta de entrada para que las aplicaciones obtengan acceso a datos, lógica de negocio o funcionalidades desde sus servicios de backend. Los desarrolladores pueden crear API que accedan a AWS o a otros servicios web, así como los datos almacenados en la nube de AWS. Un objeto API definido con el servicio API Gateway es un grupo de recursos y métodos.

Un recurso es un objeto escrito dentro del dominio de una API y puede tener asociado un modelo de datos o relaciones con otros recursos. Cada uno de ellos se puede configurar para responder a uno o más métodos, es decir, verbos HTTP estándar como GET, POST o PUT. Es posible implementar las API REST en diferentes etapas, y controlar sus versiones y clonarlas en nuevas versiones.

API Gateway maneja todas las tareas involucradas en la aceptación y el procesamiento de cientos de miles de llamadas simultáneas a la API, que incluye la administración del tráfico, la autorización y el control de acceso, la supervisión y la administración de la versión de la API.

Mensajería asíncrona y transferencia de eventos

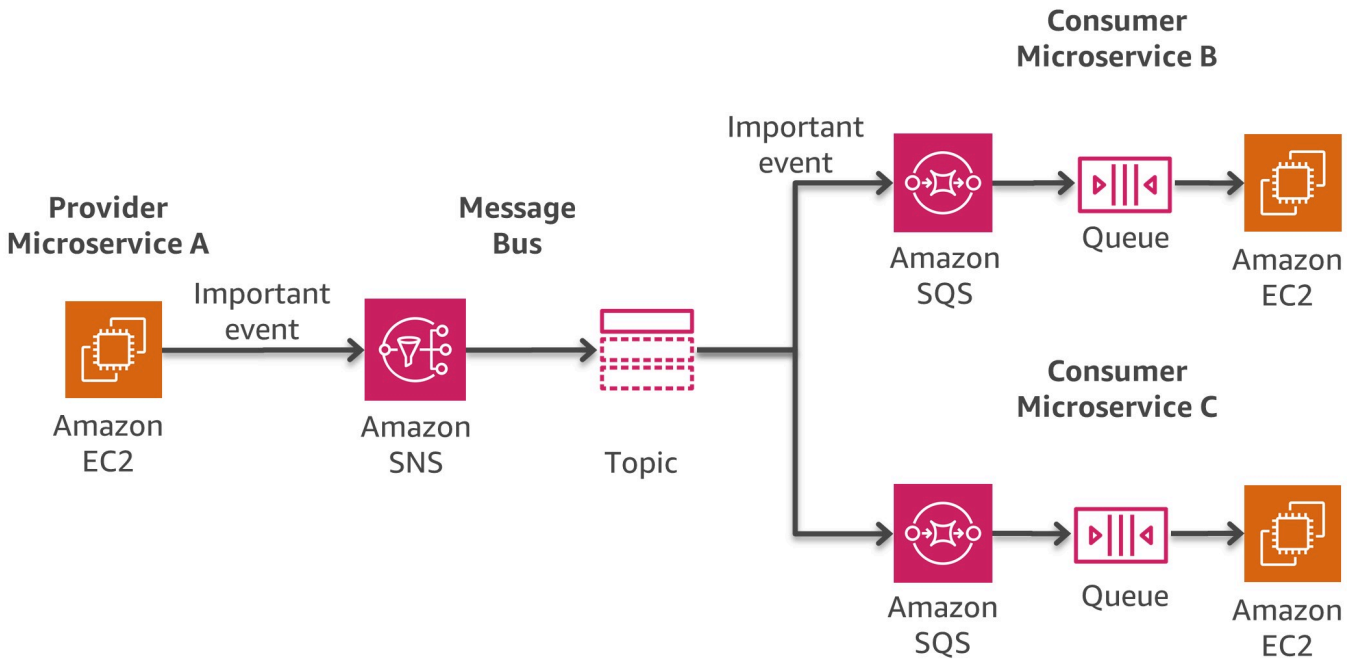
Un patrón adicional usado para implementar la comunicación entre microservicios es la transferencia de mensajes. Los servicios se comunican mediante el intercambio de mensajes a través de una cola. Un beneficio importante de este estilo de comunicación es que no es necesaria la detección de servicios y que los servicios son de estructura flexible.

Los sistemas síncronos son de estructura rígida, lo que significa que cuando ocurre un problema en una dependencia descendente síncrona existe un impacto inmediato en los intermediarios ascendentes. Los reintentos de los intermediarios ascendentes pueden distribuir y aumentar los problemas rápidamente.

En función de requisitos específicos, como protocolos, AWS ofrece diferentes servicios que ayudan a implementar este patrón. Una implementación posible utiliza una combinación de cola de Amazon Simple Queue Service (Amazon SQS) o Amazon Simple Notification Service (Amazon SNS).

Ambos servicios trabajan de manera conjunta: Amazon SNS permite que las aplicaciones envíen mensajes a múltiples suscriptores a través de un mecanismo «push». Al utilizar Amazon SNS y

Amazon SQS juntos, se puede enviar un mensaje a varios consumidores. La siguiente figura muestra la integración de Amazon SNS y Amazon SQS.



Patrón de bus de mensajes en AWS

Cuando suscribe una cola de Amazon SQS a un tema de SNS, puede publicar un mensaje en el tema y Amazon SNS envía un mensaje a la cola de Amazon SQS suscrita. El mensaje incluye el asunto y el mensaje publicado en el tema junto con información de metadatos en formato JSON.

Otra opción para crear arquitecturas basadas en eventos con orígenes de eventos que abarquen aplicaciones internas, aplicaciones SaaS de terceros y servicios de AWS, a escala, es Amazon EventBridge. EventBridge, un servicio de bus de eventos completamente administrado, recibe eventos de orígenes dispares, identifica un destino en función de una regla de enrutamiento y entrega datos casi en tiempo real a ese destino, incluidos AWS Lambda, Amazon SNS y Amazon Kinesis Streams, entre otros. Un evento entrante también se puede personalizar, mediante un transformador de entrada, antes de la entrega.

Para desarrollar aplicaciones basadas en eventos significativamente más rápido, los registros de esquemas de EventBridge recopilan y organizan esquemas, incluidos los esquemas para todos los eventos generados por los servicios de AWS. Los clientes también pueden definir esquemas personalizados o utilizar una opción de deducir el esquema para detectar esquemas

automáticamente. Sin embargo, en general, una posible contrapartida para todas estas funciones es un valor de latencia relativamente más alto para la entrega de EventBridge. Además, el rendimiento y las cuotas predeterminados de EventBridge pueden requerir un aumento, a través de una solicitud de soporte, según el caso de uso.

Una estrategia de implementación diferente se basa en Amazon MQ, que se puede utilizar si el software existente usa protocolos y API de estándar abierto para mensajería, incluidos JMS, NMS, AMQP, STOMP, MQTT y WebSocket. Amazon SQS exhibe una API personalizada, lo que significa que si quiere migrar una aplicación existen desde, por ejemplo, un entorno local a AWS, es necesario implementar cambios en el código. Con Amazon MQ, en muchos casos esto no es necesario.

Amazon MQ gestiona la administración y el mantenimiento de ActiveMQ, un agente de mensajes de código abierto de amplio uso. La infraestructura subyacente se aprovisiona automáticamente para lograr un nivel de disponibilidad alto y mayor durabilidad de los mensajes con el fin de respaldar la fiabilidad de sus aplicaciones.

Administración de estados y orquestación

El carácter distribuido de los microservicios dificulta la orquestación de flujos de trabajo cuando existen varios microservicios involucrados. Los desarrolladores pueden sentirse tentados a agregar código de orquestación directamente en sus servicios. Esto debe evitarse porque introduce un acoplamiento más estrecho y dificulta el reemplazo rápido de los servicios individuales.

Puede usar AWS Step Functions para crear aplicaciones a partir de componentes individuales, cada uno de los cuales realiza una función separada. Step Functions proporciona una máquina de estados que oculta las complejidades de la organización del servicio, como el manejo de errores, la serialización y la paralelización. Esto le permite escalar y cambiar aplicaciones rápidamente evitando códigos de coordinación adicionales dentro de los servicios.

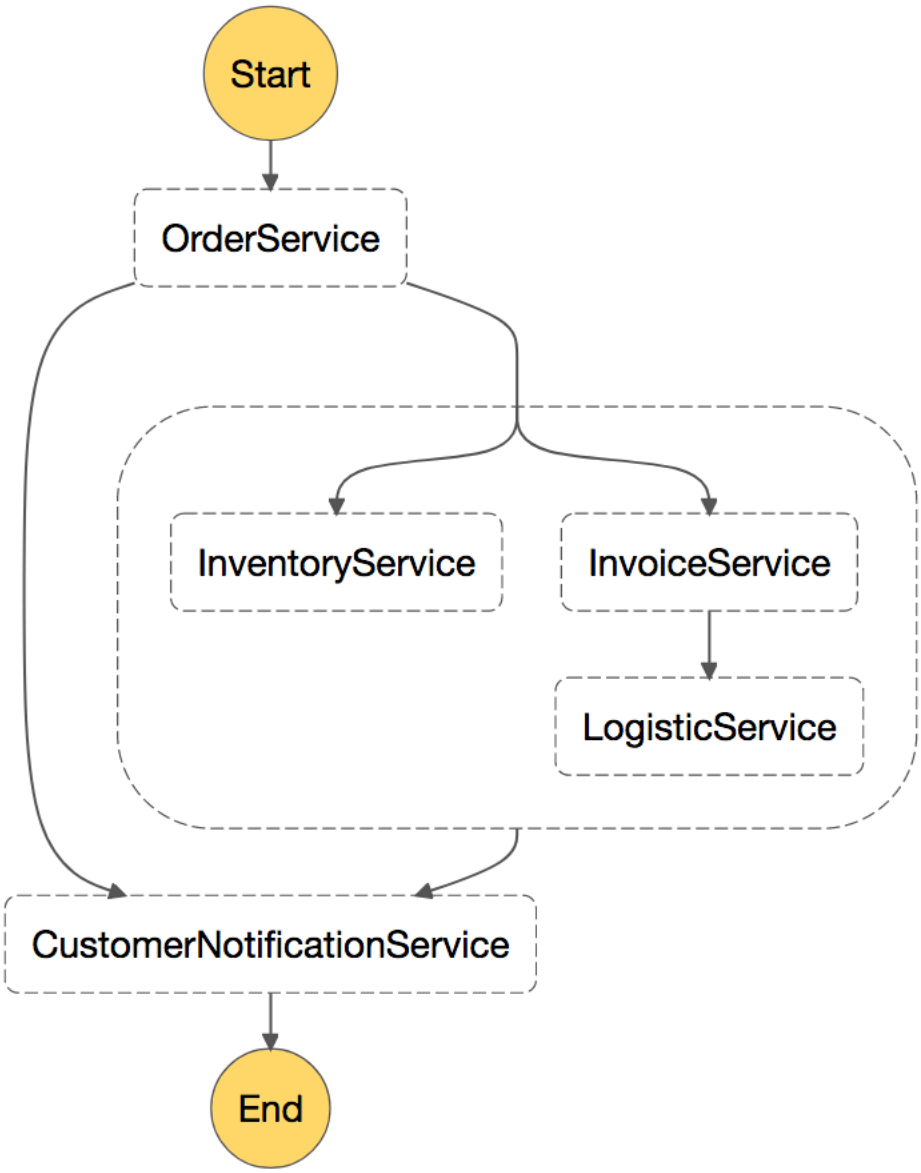
Step Functions es una manera fiable de coordinar los componentes y procesar las funciones de su aplicación. Step Functions proporciona una consola gráfica con la que ordenar y visualizar los componentes de su aplicación en varios pasos. De este modo, crear y ejecutar servicios distribuidos resulta fácil.

Step Functions inicia y sigue cada paso de manera automática, y realiza reintentos cuando se producen errores, por lo que su aplicación se ejecuta en orden y según lo previsto. Step Functions registra el estado de cada paso, de manera que, cuando algo sale mal, puede diagnosticar y depurar los problemas con rapidez. Puede cambiar y agregar pasos sin escribir código para que la aplicación evolucione con facilidad e innovar con mayor velocidad.

Step Functions forma parte de la plataforma sin servidor de AWS y admite la orquestación de funciones de Lambda, así como también aplicaciones basadas en recursos de cómputo, como Amazon EC2, Amazon EKS, Amazon ECS, y servicios adicionales como Amazon SageMaker y AWS Glue. Step Functions administra las operaciones y la infraestructura subyacente para ayudarle a garantizar que su aplicación esté disponible en cualquier escala.

Para crear flujos de trabajo, Step Functions usa el lenguaje de estados de Amazon. Los flujos de trabajo pueden contener pasos secuenciales o paralelos, así como pasos que se bifurcan.

La siguiente figura muestra un ejemplo de flujo de trabajo para una arquitectura de microservicios que combina pasos secuenciales y paralelos. La invocación de dicho flujo de trabajo se puede hacer a través de la API de Step Functions o con API Gateway.



Un ejemplo de un flujo de trabajo de microservicios invocado por Step Functions

Supervisión distribuida

Una arquitectura de microservicios consta de muchas partes distribuidas diferentes que se deben supervisar. Puede usar Amazon CloudWatch para recopilar y hacer un seguimiento de las métricas, centralizar y supervisar los archivos de registro, configurar alarmas y reaccionar automáticamente ante cambios que ocurran en su entorno de AWS. CloudWatch puede supervisar recursos de AWS como, por ejemplo, instancias de Amazon EC2, tablas de DynamoDB e instancias de base de datos de Amazon RDS, así como métricas personalizadas generadas por las aplicaciones y los servicios, y los archivo de registro generados por las aplicaciones.

Monitorización

Puede usar CloudWatch para lograr la visualización de todo el sistema con respecto a la utilización de recursos, el rendimiento de las aplicaciones y el estado operativo. CloudWatch proporciona una solución de monitorización confiable, escalable y flexible que puede comenzar a usar en minutos. Ya no necesita configurar, administrar y escalar sus propios sistemas de monitorización e infraestructura. En una arquitectura de microservicios, la capacidad de monitorizar métricas personalizadas con CloudWatch es un beneficio adicional porque los desarrolladores pueden decidir qué métricas se deben recopilar para cada servicio. Además, el escalado dinámico se puede implementar basándose en métricas personalizadas.

Además de Amazon CloudWatch, puede utilizar CloudWatch Container Insights para recopilar, agregar y resumir las métricas y los registros de sus aplicaciones y microservicios en contenedores. CloudWatch Container Insights recopila automáticamente métricas para muchos recursos, como CPU, memoria, disco y red, y las agrega como métricas de CloudWatch en el nivel de clúster, nodo, pod, tarea y servicio. Con CloudWatch Container Insights, puede obtener acceso a las métricas del panel de CloudWatch Container Insights. También proporciona información de diagnóstico, como, por ejemplo, errores de reinicio de contenedores, para ayudarle a aislar problemas y solucionarlos rápidamente. También puede establecer alarmas de CloudWatch en las métricas que recopila Container Insights.

Container Insights está disponible para las plataformas Amazon ECS, Amazon EKS y Kubernetes en Amazon EC2. El soporte de Amazon ECS incluye soporte para Fargate.

Otra opción muy utilizada (especialmente para Amazon EKS) es recurrir a Prometheus. Prometheus es un conjunto de herramientas de supervisión y alertas de código abierto que, a menudo, se utiliza

junto con Grafana para visualizar las métricas recopiladas. Muchos componentes de Kubernetes almacenan métricas en `/metrics` y Prometheus puede extraerlas a intervalos regulares.

Amazon Managed Service for Prometheus (AMP) es un nuevo servicio de supervisión compatible con Prometheus que permite supervisar aplicaciones en contenedores a escala. Con AMP, puede utilizar el lenguaje de consulta de Prometheus (PromQL) de código abierto para supervisar el rendimiento de las cargas de trabajo en contenedores sin necesidad de administrar la infraestructura subyacente para lidiar con la ingesta, el almacenamiento, la consulta y las métricas operativas. Puede recopilar métricas de Prometheus de entornos Amazon EKS y Amazon ECS, utilizando AWS Distro for OpenTelemetry o servidores Prometheus como agentes de recopilación.

AMP se usa a menudo en combinación con Amazon Managed Service for Grafana (AMG). Con AMG es fácil consultar, visualizar, alertar y comprender las métricas sin importar dónde se encuentren almacenadas. Con AMG, puede analizar sus métricas, registros y seguimientos sin tener que suministrar servidores, configurar ni actualizar software, y sin tener que lidiar con las arduas tareas administrativas que conlleva la seguridad y el escalado de Grafana en la producción.

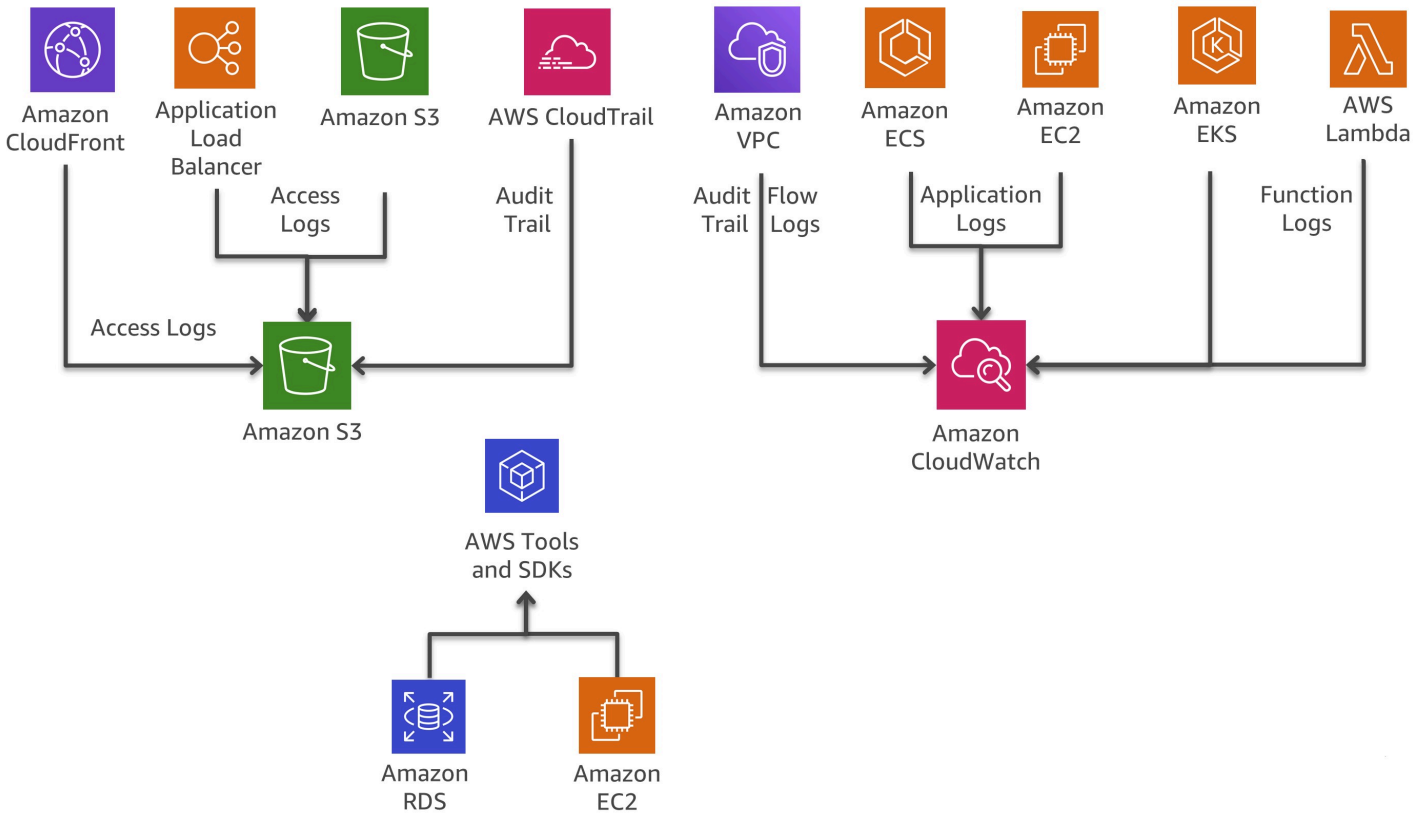
Centralización de registros

El registro sistemático es crucial para resolver problemas e identificar problemas. Los microservicios permiten que los equipos envíen una cantidad de versiones nuevas sin precedentes e instan a los equipos de ingeniería a realizar pruebas con las nuevas características en producción. Conocer el impacto en el cliente es fundamental para mejorar una aplicación gradualmente.

De forma predeterminada, la mayoría de los servicios de AWS centraliza los archivos de registro de manera predeterminada. Los principales destinos para los archivos de registro en AWS son Amazon S3 y Amazon CloudWatch Logs. Para las aplicaciones que se ejecutan en instancias de Amazon EC2, hay un daemon disponible para enviar archivos de registro a CloudWatch Logs. Las funciones de Lambda envían de forma nativa los resultados de registro a CloudWatch Logs y Amazon ECS admite el controlador de registros awslogs, lo que permite la centralización de registros de contenedores en CloudWatch Logs. Para Amazon EKS, Fluent Bit o Fluentd pueden enviar registros desde las instancias individuales en el clúster a un CloudWatch Logs de registro centralizado, donde se combinan para lograr una generación de informes de alto nivel mediante el uso de Amazon OpenSearch Service y Kibana. Debido a sus ventajas de rendimiento y tamaño más reducido, se recomienda Fluent Bit en lugar de FluentD.

La siguiente figura ilustra las capacidades de registro de algunos de los servicios. Luego, los equipos pueden buscar y analizar estos registros mediante herramientas como Amazon OpenSearch Service

y Kibana. Amazon Athena se puede utilizar para ejecutar consultas únicas en archivos de registro centralizados en Amazon S3.



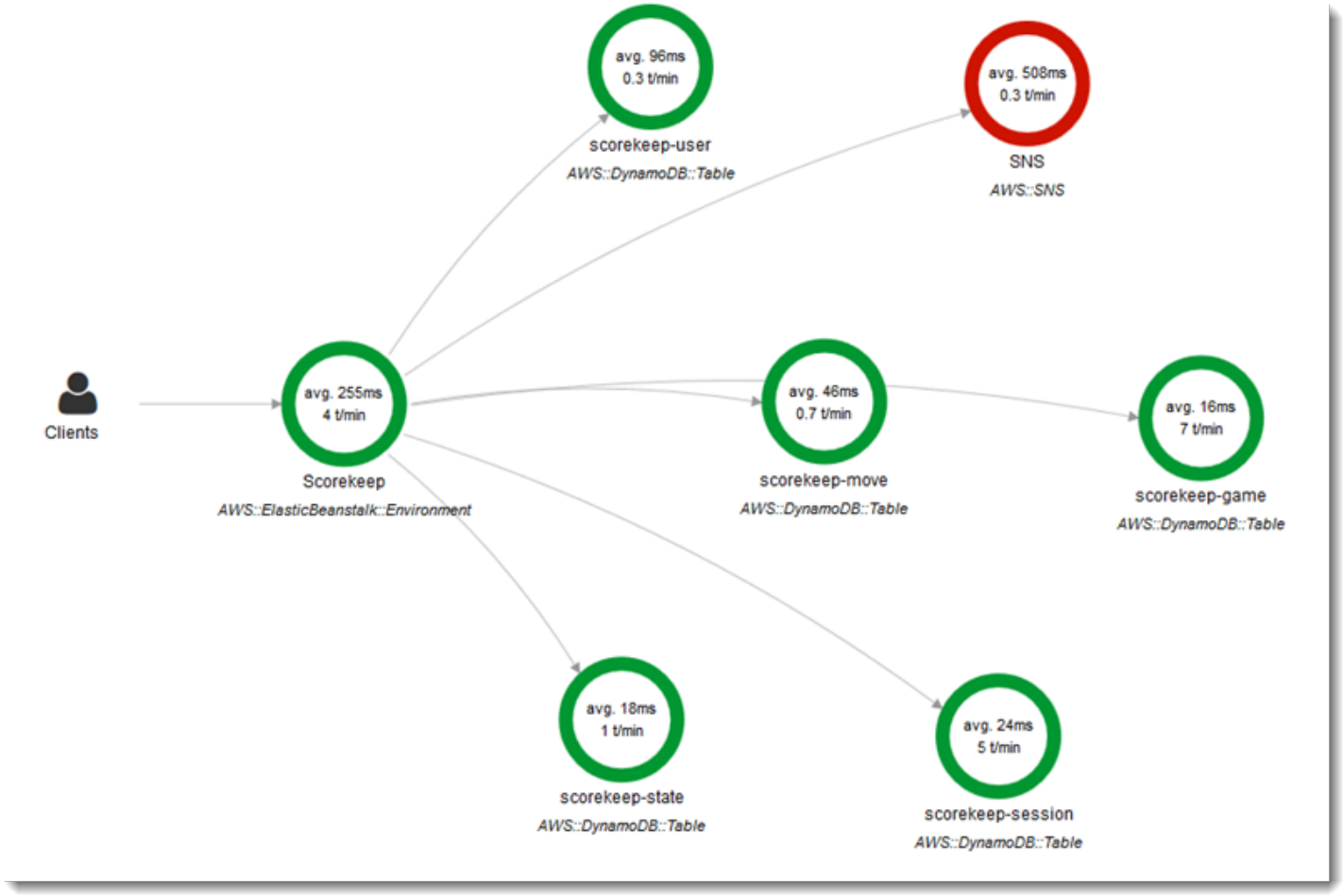
Capacidades de registro de los servicios de AWS

Seguimiento distribuido

En muchos casos, un conjunto de microservicios trabaja conjuntamente para manejar una solicitud. Imagine un sistema complejo que consta de decenas de microservicios en los que se produce un error en uno de los servicios de la cadena de llamadas. Aunque cada microservicio se registre correctamente y los registros se consoliden en un sistema central, puede resultar difícil encontrar todos los mensajes de registro relevantes.

La idea central de AWS X-Ray es el uso de los ID de correlación, que son identificadores únicos asociados con todas las solicitudes y los mensajes relacionados con una cadena de eventos específica. El ID de seguimiento se agrega a las solicitudes HTTP en encabezados de seguimiento específicos llamados `X-Amzn-Trace-Id` cuando la solicitud llega al primer servicio integrado de X-Ray (por ejemplo, Application Load Balancer o API Gateway) y se incluye en la respuesta. A través del SDK de X-Ray, cualquier microservicio puede leer, pero también puede agregar o actualizar este encabezado.

X-Ray funciona con Amazon EC2, Amazon ECS, Lambda y AWS Elastic Beanstalk. Puede usar X-Ray con aplicaciones escritas en Java, Node.js y .NET que se implementan en estos servicios.



Mapa de servicios de AWS X-Ray

Epsagon es un SaaS completamente administrado que incluye el seguimiento de todos los servicios de AWS, API de terceros (a través de llamadas HTTP) y otros servicios comunes como Redis, Kafka y Elastic. El servicio Epsagon incluye capacidades de supervisión, alertas de los servicios más habituales y visibilidad de la carga útil en todas y cada una de las llamadas que realiza su código.

AWS Distro for OpenTelemetry es una distribución segura, lista para la producción y compatible con AWS del proyecto OpenTelemetry. Como parte de Cloud Native Computing Foundation, AWS Distro for OpenTelemetry proporciona las API de código abierto, bibliotecas y agentes a fin de recopilar métricas y seguimiento distribuidos para la supervisión de aplicaciones. Con AWS Distro for OpenTelemetry, puede equipar las aplicaciones una vez, enviar los seguimientos y métricas correlativos a diferentes soluciones de supervisión de AWS y los socios. Use agentes de equipamiento automático para recopilar seguimientos sin cambiar el código. Además, AWS Distro

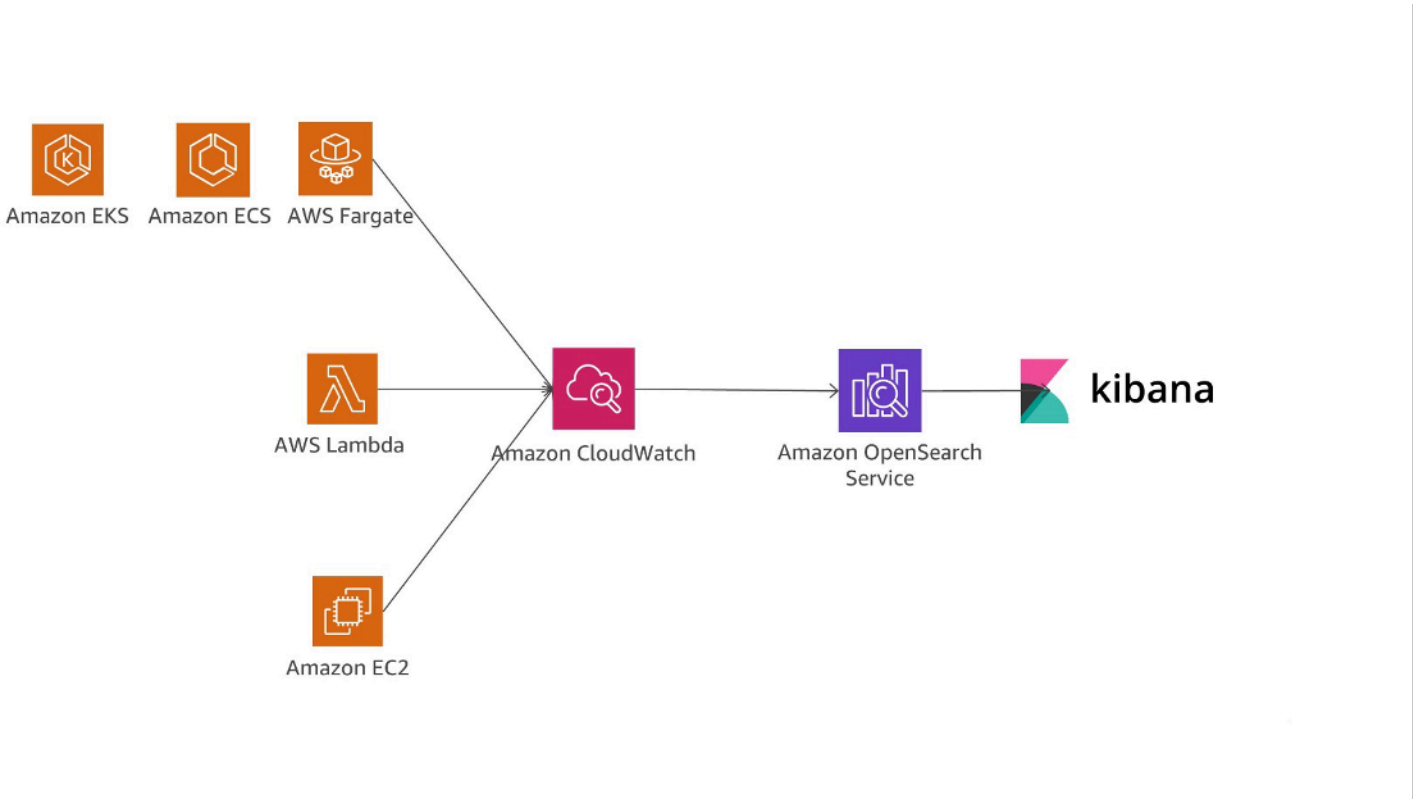
of OpenTelemetry recopila metadatos de los servicios administrados y los recursos de AWS, para que usted pueda establecer la correlación de los datos de rendimiento de la aplicación con los datos de la infraestructura subyacente, lo que reduce el tiempo promedio de resolución de problemas. Utilice AWS Distro for OpenTelemetry para equipar las aplicaciones que se ejecutan en Amazon EC2, Amazon ECS, Amazon EKS en Amazon EC2, Fargate y AWS Lambda, así como localmente.

Opciones para el análisis de registros en AWS

La búsqueda, el análisis y la visualización de datos de registro es un aspecto importante para comprender los sistemas distribuidos. Amazon CloudWatch Logs Insights le permite explorar, analizar y visualizar registros de forma instantánea. Esto le permite resolver problemas operativos. Otra opción muy utilizada para analizar los archivos de registro es usar Amazon OpenSearch Service junto con Kibana.

Amazon OpenSearch Service se puede utilizar para realizar búsquedas de texto completo, búsquedas estructuradas, análisis y una combinación de los tres. Kibana es un complemento de visualización de datos de código abierto que se integra perfectamente con Amazon OpenSearch Service.

La siguiente figura muestra el análisis de registros con Amazon OpenSearch Service y Kibana. A través de una suscripción a CloudWatch Logs, se puede configurar Amazon OpenSearch Service para transmitir entradas de registro a Amazon OpenSearch Service casi en tiempo real. Kibana visualiza los datos y expone una interfaz de búsqueda práctica para los almacenes de datos en Amazon OpenSearch Service. Esta solución se puede utilizar en combinación con un software como ElastAlert con el fin de implementar un sistema de alertas para enviar correos electrónicos y notificaciones de SNS, crear tiques JIRA, etc., si se detectan anomalías, consumos abruptos u otros patrones de interés en los datos.



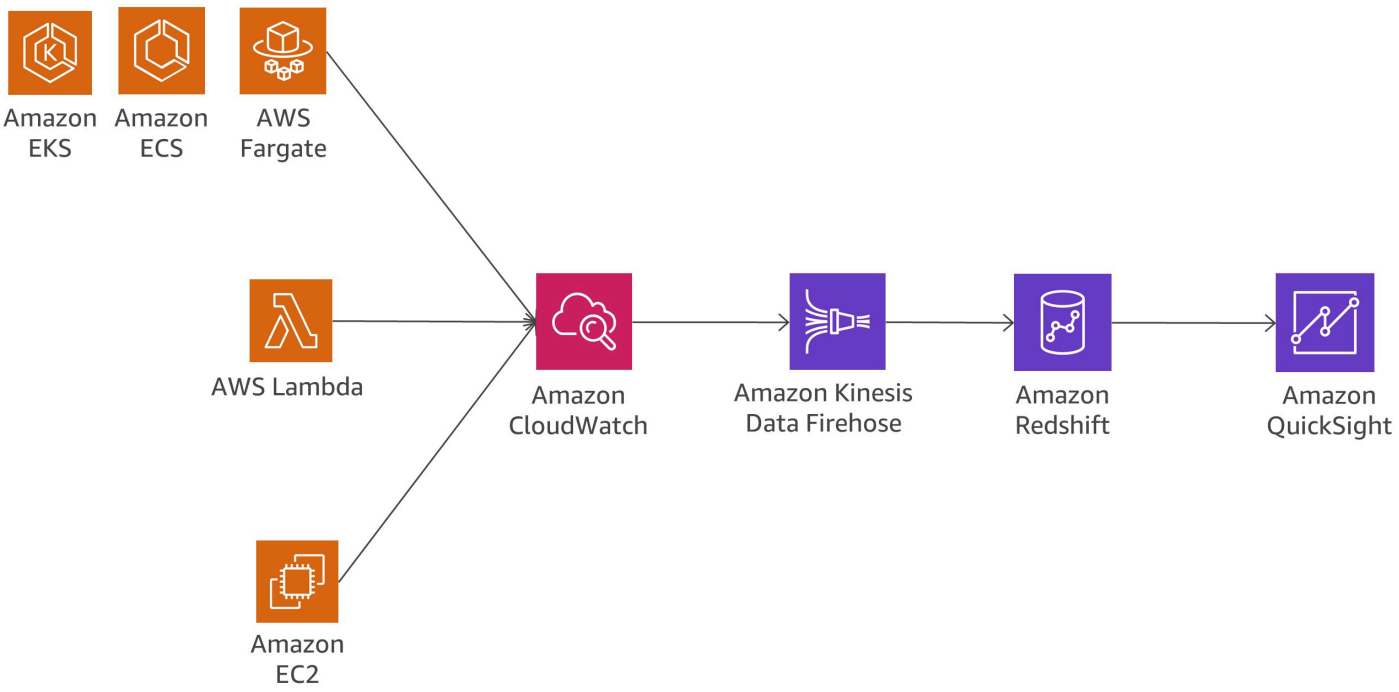
Análisis de registros con Amazon OpenSearch Service y Kibana

Otra opción para analizar los archivos de registro es usar Amazon Redshift con Amazon QuickSight.

QuickSight se puede conectar fácilmente a los servicios de datos de AWS, incluidos Amazon Redshift, Amazon RDS, Amazon Aurora, Amazon EMR, DynamoDB, Amazon S3 y Amazon Kinesis

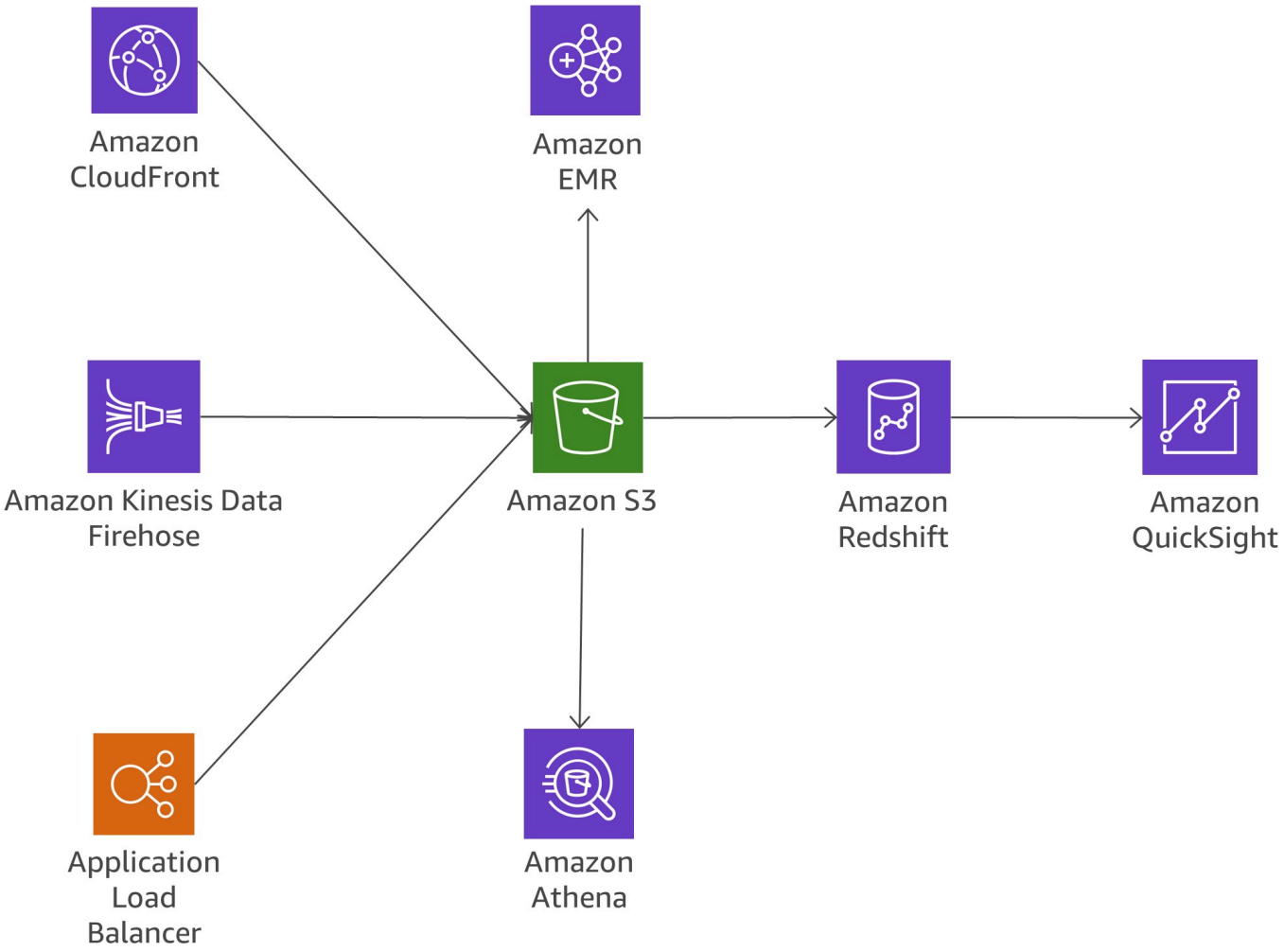
CloudWatch Logs puede funcionar como un almacén centralizado para los datos de registro y, además de solo almacenar los datos, es posible transmitir entradas de registro a Amazon Kinesis Data Firehose.

La siguiente figura presenta un caso donde se transmiten entradas de registro desde diferentes orígenes a Amazon Redshift con CloudWatch Logs y Kinesis Firehose. Amazon QuickSight usa los datos almacenados en Amazon Redshift para análisis, generación de informes y visualización.



Análisis de registros con Amazon Redshift y Amazon QuickSight

La siguiente figura representa una situación de análisis de registros en Amazon S3. Cuando los registros se almacenan en buckets de Amazon S3, los datos de registro se pueden cargar en diferentes servicios de datos de AWS, como Amazon Redshift o Amazon EMR, para analizar los datos almacenados en la transmisión de registros y detectar anomalías.



Análisis de registros en Amazon S3

Ruido de datos innecesarios

Al dividir las aplicaciones monolíticas en microservicios pequeños, la sobrecarga de comunicación aumenta porque los microservicios tienen que comunicarse entre sí. Si bien en muchas implementaciones se utiliza REST a través de HTTP porque es un protocolo de comunicación ligero, los volúmenes de mensajes elevados pueden generar problemas. En algunos casos, podría considerar el uso de servicios consolidados que envíen muchos mensajes de ida y vuelta. Si se encuentra en una situación en la que consolida cada vez más servicios solo para reducir el ruido de datos innecesarios, debe revisar sus dominios con problemas y su modelo de dominio.

Protocolos

Anteriormente en este documento técnico, en la sección the section called “Comunicación asíncrona y mensajería liviana”, se examinan diferentes protocolos posibles. Para los microservicios es común usar protocolos simples como HTTP. Los mensajes que intercambian los servicios se pueden codificar de diferentes maneras, como en un formato legible para las personas como JSON o YAML o en un formato binario eficiente como Avro o Protocol Buffers.

Almacenamiento en caché

Los cachés son una excelente manera de reducir la latencia y el ruido de las arquitecturas de microservicios. Varias capas de almacenamiento en caché son posibles según el caso de uso real y los cuellos de botella. Muchas aplicaciones de microservicios que se ejecutan en AWS utilizan ElastiCache para reducir el volumen de llamadas a otros microservicios mediante el almacenamiento local en caché de los resultados. API Gateway proporciona una capa de almacenamiento en caché integrada para reducir la carga en los servidores de backend. Además, el almacenamiento en caché también es útil para reducir la carga de la capa de persistencia de los datos. El desafío para cualquier mecanismo de almacenamiento en caché es encontrar el equilibrio adecuado entre una buena tasa de aciertos de caché y la puntualidad y la coherencia de los datos.

Auditoría

Otro desafío que se debe abordar en las arquitecturas de microservicios, que potencialmente podrían tener cientos de servicios distribuidos, es garantizar la visibilidad de las acciones de los usuarios en cada servicio y poder obtener una buena visión general de todos los servicios a nivel de la organización. Para ayudar a hacer cumplir las políticas de seguridad, es importante auditar tanto el acceso a los recursos como las actividades que generan cambios en el sistema.

Los cambios deben rastrearse a nivel del servicio individual así como también en los servicios que se ejecutan en el sistema más amplio. Es normal que ocurran cambios frecuentes en las arquitecturas de microservicios, lo que aumenta la importancia de auditar los cambios. En esta sección, examinaremos los servicios y las características clave dentro de AWS que pueden ayudarle a auditar su arquitectura de microservicios.

Traza de auditoría

AWS CloudTrail es una herramienta útil para rastrear cambios en microservicios porque permite que todas las llamadas a la API realizadas en la nube de Nube de AWS se registren y se envíen a CloudWatch Logs en tiempo real o a Amazon S3 en unos minutos.

Todas las acciones de los usuarios y los sistemas automatizados se pueden buscar y se pueden analizar con el fin de detectar comportamientos inesperados, infracciones de políticas de la empresa o depuración. La información registrada incluye una marca de tiempo, información del usuario y la cuenta, el servicio al que se llamó, la acción de servicio solicitada, la dirección IP de la persona que llama, así como parámetros de solicitud y elementos de respuesta.

CloudTrail posibilita la definición de múltiples trazas para la misma cuenta, lo que permite a las diferentes partes interesadas, como administradores de seguridad, desarrolladores de software o auditores de TI, crear y administrar su propia traza. Si los equipos de microservicios tienen diferentes cuentas de AWS, es posible agregar trazas en un solo bucket de S3.

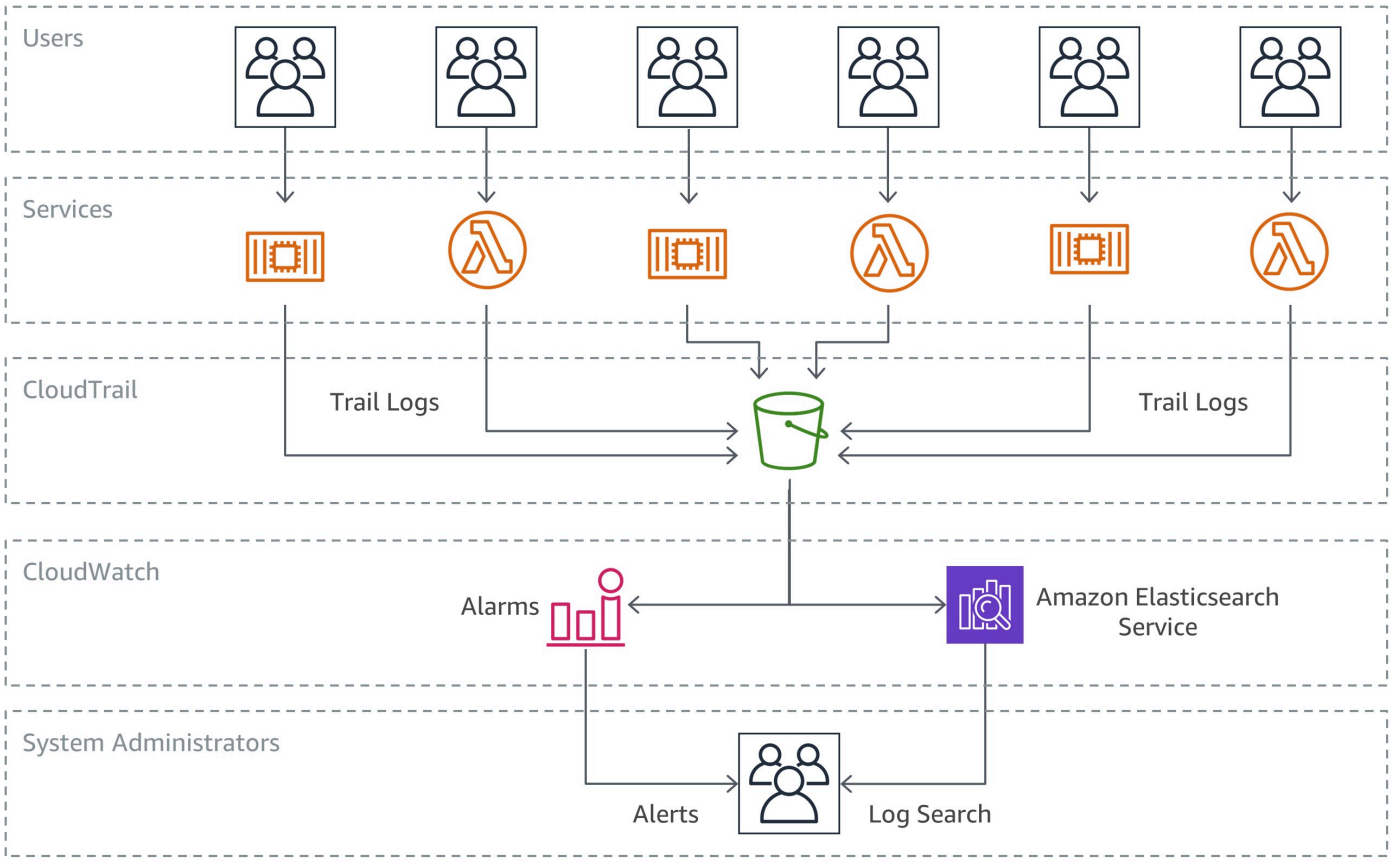
Las ventajas de almacenar las trazas de auditoría en CloudWatch son que los datos de la traza de auditoría se registran en tiempo real y que se facilita el direccionamiento de la información a Amazon OpenSearch Service para fines de búsqueda y visualización. Puede configurar CloudTrail para iniciar sesión en Amazon S3 y en CloudWatch Logs.

Eventos y acciones en tiempo real

Ciertos cambios en las arquitecturas de los sistemas deben responderse rápidamente y se debe tomar una acción de resolución o deben seguirse procedimientos específicos de gobernanza para autorizar el inicio del cambio. La integración de Amazon CloudWatch Events con CloudTrail le permite generar eventos para todas las llamadas a la API mutantes en todos los servicios de AWS. También es posible definir eventos personalizados o generar eventos basados en un cronograma fijo.

Cuando se activa un evento y coincide con una regla definida, se puede notificar inmediatamente a un grupo predefinido de su organización para que tome las medidas oportunas. Si la acción requerida se puede automatizar, la regla puede activar automáticamente un flujo de trabajo integrado o invocar una función de Lambda para resolver el problema.

La siguiente figura muestra un entorno donde CloudTrail y CloudWatch Events trabajan juntos para abordar los requisitos de auditoría y resolución dentro de una arquitectura de microservicios. CloudTrail realiza un seguimiento de todos los microservicios y la traza de auditoría se almacena en un bucket de Amazon S3. CloudWatch Events conoce los cambios operativos a medida que se producen. CloudWatch Events responde a estos cambios operativos y toma medidas correctoras según sea necesario, enviando mensajes para responder al entorno, activando funciones, realizando cambios y captando información de estado. CloudWatch Events se ubica sobre CloudTrail y activa alertas cuando se realiza un cambio específico en su arquitectura.



Auditoría y resolución

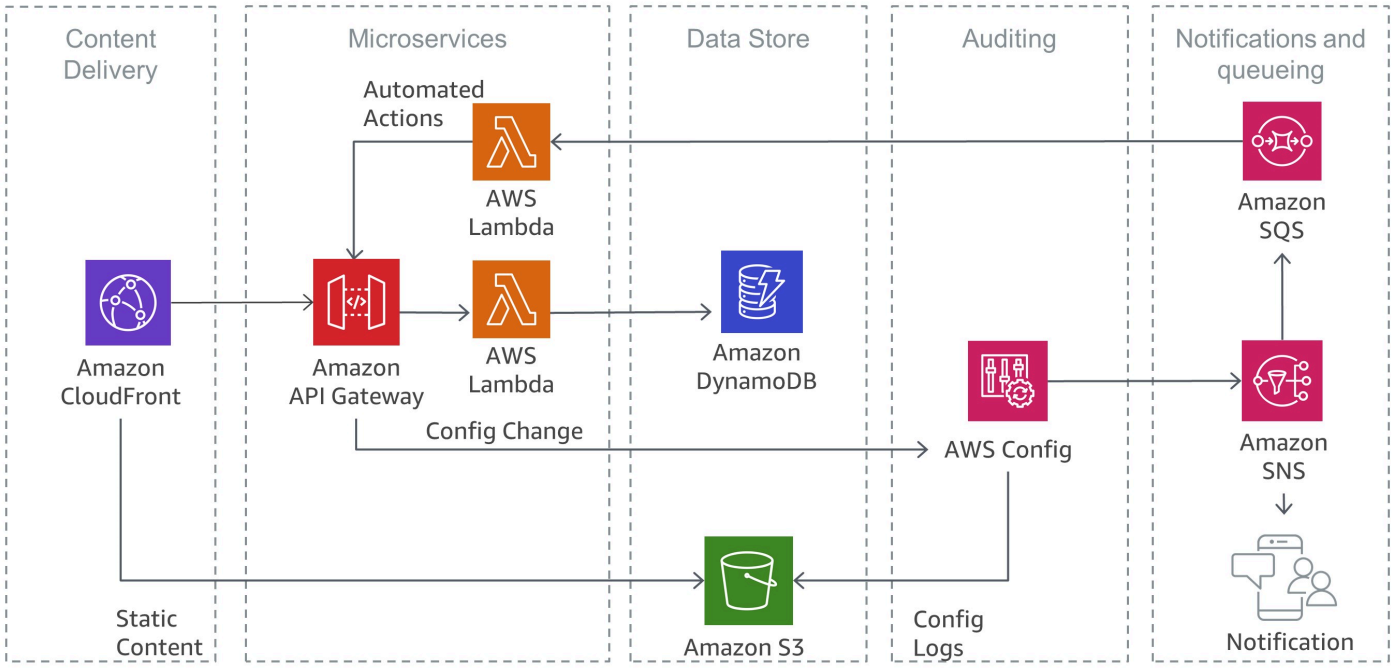
Inventario de recursos y administración de cambios

Para mantener el control sobre las configuraciones de infraestructura que cambian rápidamente en un entorno de desarrollo ágil, es clave tener un enfoque administrado más automatizado con respecto a la auditoría y el control de su arquitectura.

Si bien CloudTrail y CloudWatch Events son bloques de creación importantes para seguir y responder a los cambios de la infraestructura en los microservicios, las reglas de AWS Config permiten que una empresa defina políticas de seguridad con reglas específicas con el fin de detectar, seguir y notificar automáticamente las transgresiones de estas políticas.

En el siguiente ejemplo se muestra cómo es posible detectar, informar y reaccionar automáticamente a los cambios de configuración no conformes dentro de su arquitectura de microservicios. Un miembro del equipo de desarrollo realizó un cambio en la gateway de API en un microservicio para permitir que el punto de enlace acepte tráfico HTTP entrante, en vez de permitir únicamente solicitudes HTTPS.

Como con anterioridad la organización había identificado esta situación como un problema de conformidad de seguridad, una regla de AWS Config ya se encuentra monitorizando esta condición. La regla identifica el cambio como una infracción de seguridad y ejecuta dos acciones: crea un registro del cambio detectado en un bucket de Amazon S3 para fines de auditoría y crea una notificación de SNS. Amazon SNS se utiliza con dos propósitos en esta situación: para enviar un correo electrónico a un grupo específico a fin de informar una infracción de seguridad y para agregar un mensaje a una cola de SQS. A continuación, el mensaje se recoge y se vuelve al estado de conformidad al cambiar la configuración de API Gateway.



Detección de infracciones de seguridad con AWS Config

Conclusión

La arquitectura de microservicios es una estrategia de diseño distribuido creado para superar las limitaciones de las arquitecturas monolíticas tradicionales. Los microservicios ayudan a ajustar la escala de aplicaciones y organizaciones mientras se mejoran los tiempos del ciclo. Sin embargo, también implican un par de desafíos que pueden aumentar la complejidad de la arquitectura y la carga operativa.

AWS ofrece una gran cartera de servicios administrados que pueden ayudar a los equipos de productos a crear arquitecturas de microservicios y minimizar la complejidad operativa y de la arquitectura. Este documento técnico le guía por los servicios de AWS relevantes y le muestra cómo implementar patrones típicos, como la detección de servicios o el abastecimiento de eventos de forma nativa con los servicios de AWS.

Recursos

- Centro de arquitectura de AWS
- Documentos técnicos de AWS
- Arquitectura mensual de AWS
- Blog de arquitectura de AWS
- Vídeos de This Is My Architecture
- AWS Answers
- Documentación de AWS

Historial de revisión y colaboradores

Historial de revisión

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

update-history-change	update-history-description	update-history-date
Documento técnico actualizado	Integración de Amazon EventBridge, AWS OpenTelemetry, AMP, 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 de texto menores.	1 de agosto de 2019
Documento técnico actualizado	Integración de Amazon EKS, AWS Fargate, Amazon MQ, AWS PrivateLink, AWS App Mesh, AWS Cloud Map	1 de junio de 2019
Documento técnico actualizado	Integración de AWS Step Functions, AWS X-Ray y secuencia de eventos de ECS.	1 de septiembre de 2017
Publicación inicial	Implementación de microservicios en AWS.	1 de diciembre de 2016

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

Colaboradores

Las siguientes personas y organizaciones contribuyeron a redactar este documento:

- Sascha Möllering, Arquitectura de soluciones, AWS
- Christian Müller, Arquitectura de Soluciones, AWS
- Matthias Jung, Arquitectura de soluciones, AWS
- Peter Dalbhanjan, Arquitectura de soluciones, AWS
- Peter Chapman, Arquitectura de soluciones, AWS
- Christoph Kassen, Arquitectura de soluciones, AWS
- Umair Ishaq, Arquitectura de soluciones, AWS
- Rajiv Kumar, Arquitectura de soluciones, AWS

Avisos

Los clientes son responsables de realizar sus propias evaluaciones de la información contenida en este documento. Este documento: (a) solo tiene fines informativos, (b) representa las prácticas y las ofertas de productos vigentes de AWS, que están sujetas a cambios sin previo aviso, y (c) no crea ningún compromiso ni garantía de AWS y sus empresas afiliadas, proveedores o concesionarios de licencias. Los productos o servicios de AWS se proporcionan “tal cual”, sin garantías, representaciones ni condiciones de ningún tipo, ya sean explícitas o implícitas. Las responsabilidades y obligaciones de AWS en relación con sus clientes se rigen por los acuerdos de AWS, y este documento no modifica ni forma parte de ningún acuerdo entre AWS y sus clientes.